
Article:
Florescu, D. and Coca, D. (2019) Learning with precise spike times : a new decoding algorithm for liquid state machines. Neural Computation, 31 (9). pp. 1825-1852. ISSN: 0899-7667

https://doi.org/10.1162/neco_a_01218

Learning with precise spike times: A new decoding algorithm for liquid state machines

Dorian Florescu, Daniel Coca*

Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, S1 3JD, UK.

Keywords: spiking neural network, temporal coding, integrate-and-fire neuron, liquid state machine

Abstract

There is extensive evidence that biological neural networks encode information in the precise timing of the spikes generated and transmitted by neurons, which offers several advantages over rate-based codes. Here we adopt a vector space formulation of spike train sequences and introduce a new liquid state machine (LSM) network architecture and a new forward orthogonal regression algorithm to learn an input-output signal mapping or to decode the brain activity. The proposed algorithm uses precise

spike timing to select the presynaptic neurons relevant to each learning task. We show that using precise spike timing to train the LSM and selecting the Readout presynaptic neurons leads to a significant increase in performance on binary classification tasks, in decoding neural activity from multielectrode array recordings, as well as in a speech recognition task, compared with what is achieved using the standard architecture and training methods.

1 Introduction

It is generally accepted that neurons in the brain encode information not only in their average firing rates - rate coding - but also in the precise timing of spikes - temporal coding (Hirata et al., 2008). The importance of the precise spike timing information has been documented in many studies (Srivastava et al., 2017; Memmesheimer et al., 2014; Kayser et al., 2009; Jones et al., 2004; Gollisch & Meister, 2008; Riehle, 1997). Seth (2015) has argued that the two encoding schemes are in fact complementary.

Neuronal coding is reproducible with a precision of a millisecond (Mainen & Sejnowski, 1995; Izhikevich, 2006). It has been argued that codes that utilise spike timing make better use of the capacity of neural connections than those relying on rate codes (Mainen & Sejnowski, 1995) and that it allows processing information on much shorter time scales allowing to track rapidly changing signals (Gardner & Grüning, 2016).

There is also evidence that during perceptual decisions, learning and behaviour can be driven by a small number of neurons that are trained to read out and interpret very sparse, precisely timed action potentials (Huber et al., 2008; Houweling & Brecht, 2008; Wolfe et al., 2010).

In recent years, a lot of research effort has been expanded to establish a sound theoretical basis for encoding and decoding using the precise timing of the spikes rather than spike-count rates (Lazar & Pnevmatikakis, 2008; Florescu & Coca, 2015; Lazar & Slutskiy, 2015; Florescu, 2017; Florescu & Coca, 2018). A range of supervised learning approaches that utilise temporal coding schemes have been developed for recurrent spiking neural networks (SNNs) with feedforward and feedback connections (Gardner & Grüning, 2016; Gütig, 2014). Some of the popular SNN training algorithms using temporal coding are based on gradient descent (Bohte et al., 2002; Xu et al., 2013; Florian, 2012; Pfister et al., 2006) or on spike timing dependent plasticity (Pfister et al., 2006; Florian, 2007; Izhikevich, 2007; Ponulak & Kasinski, 2010).

Liquid state machines (LSM) (Maass et al., 2002) are a class of recurrent SNNs that consist of a fixed high-dimensional dynamical network of biologically-realistic synapses and spiking neurons that remain unchanged during training, known as reservoir or 'Liquid', followed by a memoryless output or 'Readout' unit with adjustable synaptic weights. The Readout typically combines in a linear fashion the outputs of all the neurons in the Liquid. The LSM model can be viewed as a nonlinear dynamical system where the state vector comprises the states of all neurons in the Liquid, evolving in time according to the internal dynamics and external driving inputs, and the static Readout defines the relationship between the state vector and output (Maass et al., 2002).

The LSMs belong to the general class of reservoir computing approaches, which, compared with high-dimensional recurrent neural networks, have more biologically plausible architectures and simpler training algorithms that only tune the weights of

the connections to the Readout unit (Lukosevicius & Jaeger, 2009).

The reservoir computing approaches also include non-spiking models, as the Echo State Networks (ESNs) (Jaeger, 2001). However, the LSMs are more biologically realistic than ESNs and thus better suited for reproducing the computational properties of biological neural circuits.

The LSM Readout is typically trained by performing linear regression using the spike train outputs of the Liquid converted to continuous signals with exponential filters (Maass et al., 2002). Other proposed LSM models have feedback connections from the Readout, and are trained with recursive least squares using the filtered outputs of the Liquid (Nicola & Clopath, 2017). This leads to losing the information of the exact spike times generated by the Liquid neurons. The current training methods for LSMs learn target outputs using measurements from all the presynaptic neurons (Maass et al., 2002; Verstraeten et al., 2005) . Numerically, this model contains a large number of parameters which can lead to overfitting for large neural circuits. Moreover, it is known that only a relatively small number of cortical neurons project to different areas of the central nervous system (Häusler & Maass , 2007; Thomson et al., 2002).

In the case of ESNs, Dolinský et al. (2017) used orthogonal forward regression (OFR) to identify the contribution of each individual neuron to the response variable, and concluded that a small number of presynaptic neurons are enough to achieve accurate results.

Here we propose a new liquid state machine (LSM) architecture, and a new training algorithm that outperforms the standard methods (Maass et al., 2002; Verstraeten et al., 2005). The architecture consists of a Liquid, comprising only a SNN, in series with a

spike time based Readout. The new algorithm, called OFR with Spike Trains (OFRST), identifies the best synaptic connectivity for the Readout unit of the LSM. The learning algorithm relies on a distance metric between two spike trains that are elements of an inner product vector space (Carnell & Richardson, 2005).

Theoretical results demonstrate that the proposed architecture can learn any continuous target output by mapping it onto a unique target spike train sequence. We prove that the proposed LSM architecture achieves higher accuracy in training compared with the standard methods.

Numerical simulations are given to show the performance of the proposed method compared to the standard methods for binary and multi-label input classification tasks. Additional numerical examples are used to show separately the benefit of selecting the Readout connectivity using OFR and computing with precise spike times. The advantage of the proposed method is also demonstrated for two problems involving real world data. First we consider the problem of classifying the movement direction of drifting sinusoidal gratings using visually evoked multi-array recordings from the primary visual cortex of the monkey. Second, we test our method against the standard methods on a problem of speech recognition.

The paper is structured as follows. Section 2 introduces the standard architecture and method for training an LSM. Section 3 presents the proposed approach. Numerical simulations are in Section 4. Section 5 presents the conclusion.

2 The Standard LSM Architecture and Training Method

The spike train inputs and outputs of the LSM are elements of space $\mathbb{S}_0$ satisfying

$$\mathbb{S}_0 = \left\{ s | s = \{t_k\}_{k=1}^{P}, t_{k+1} > t_k \geq 0, \forall k = 1, \ldots, P-1 \right\}$$

The Liquid is modelled by an operator $\boldsymbol{\mathcal{L}}$, which maps the vector of input spike trains $\boldsymbol{s}^{in}$ into a vector of continuous functions $\boldsymbol{x}(t)$, also known as the state of the Liquid. The Readout is modelled by operator $\mathcal{R}$, which maps $\boldsymbol{x}(t)$ into the continuous scalar function $y(t)$, which denotes the LSM output. The function $y(t)$ satisfies (Maass et al., 2002)

$$y(t) = \mathcal{R}\left(\boldsymbol{\mathcal{L}}\boldsymbol{s}^{in}\right),$$

where $\boldsymbol{s}^{in} = \left[s_1^{in}, \ldots, s_{N_{in}}^{in}\right]$, $s_k^{in} = \left\{t_{k,1}^{in}, \ldots, t_{k,P_k^{in}}^{in}\right\}$, $\boldsymbol{\mathcal{L}} : \left[\mathbb{S}_0\right]^{N_{in}} \rightarrow \left[L^2(\mathbb{R})\right]^N$, $\mathcal{R} : \left[L^2(\mathbb{R})\right]^N \rightarrow L^2(\mathbb{R})$, where N_{in} and N denote the number of inputs and number of neurons in the SNN, respectively, P_k^{in} denotes the number of spikes in input k, and $\boldsymbol{x}(t) = \left(\boldsymbol{\mathcal{L}}\boldsymbol{s}^{in}\right)(t)$.

The Liquid is represented as the composition of two mathematical operators $\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{F}}\boldsymbol{\mathcal{L}}_{SNN}$, where $\boldsymbol{\mathcal{L}}_{SNN} : \left[\mathbb{S}_0\right]^{N_{in}} \rightarrow \left[\mathbb{S}_0\right]^N$ models a generic SNN and $\boldsymbol{\mathcal{F}} : \left[\mathbb{S}_0\right]^N \rightarrow \left[L^2(\mathbb{R})\right]^N$, $\boldsymbol{\mathcal{F}}\boldsymbol{s} = [\mathcal{F}s_1, \mathcal{F}s_2, \ldots, \mathcal{F}s_N]$, $\forall \boldsymbol{s} \in \left[\mathbb{S}_0\right]^N$, $\boldsymbol{s} = [s_1, \ldots, s_N]$ models a pool of linear filters

$$\mathcal{F}s_n = \sum_{k=1}^{P_n} e^{-\frac{t-t_k^n}{\tau_s}} \cdot 1_{[t_k^n, \infty)}(t), \tag{1}$$

where P_n denotes the number of spikes in s_n, $1_{[t_k^n,\infty)}$ denotes the characteristic function of interval $[t_k^n, \infty)$, and τ_s denotes the time constant of the filter.

Maass et al. (2002) demonstrated that this model has, under idealised conditions, universal real-time computing power. The standard LSM architecture is presented in

Figure 1.

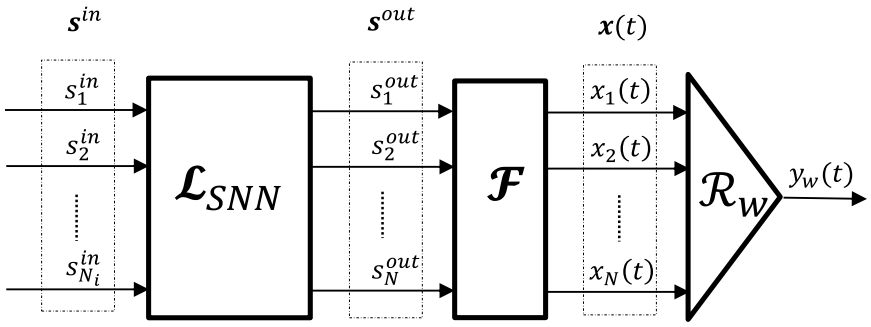

Figure 1: Block diagram of the standard architecture used for training LSMs. It consists of three blocks connected in series: the Liquid $\mathcal{L}_{SNN}$, the pool of filters $\mathcal{F}$ and the readout $\mathcal{R}_w$.

Remark 1. *Throughout the paper, it will be assumed that* $s_k^{out} \neq s_l^{out}, \forall k, l \in \{1, \ldots, N\}$, $k \neq l$. *In a practical scenario it is very unlikely that two neurons will generate two identical spike trains simultaneously. However, if this happens to be true, only the distinct outputs will be used for training.*

The most common Readout is the linear unit $\mathcal{R}_{\boldsymbol{W}}\boldsymbol{x}(t) = \sum_{n=1}^{N} w_n x_n(t)$, where $\boldsymbol{W} = [w_1, \ldots, w_N]$ and $\boldsymbol{x}(t) = [x_1(t), \ldots, x_N(t)]$. This Readout was shown to classify time-varying inputs with the same power as complex non-linear Readouts, given a large enough Liquid (Häusler et al., 2002). A typical way to train the Readout is by tuning the weights using the least squares (LS) algorithm

$$\boldsymbol{w}_{opt} = \underset{\boldsymbol{w}}{\operatorname{argmin}} \|y^* - y_{\boldsymbol{w}}\|_{L^2}, \tag{2}$$

where $y^* \in L^2(\mathbb{R})$ denotes the target output function, $\|\cdot\|_{L^2}$ denotes the standard norm in $L^2(\mathbb{R})$ and $y_{\boldsymbol{w}} = \mathcal{R}_{\boldsymbol{w}}\mathcal{F}\mathcal{L}_{SNN}\boldsymbol{s}^{in}$ denotes the predicted output.

In practice, the continuous state of the liquid $\boldsymbol{x}(t)$ is sampled uniformly with period $\Delta T > 0$. The function $\boldsymbol{x}(t) = [\mathcal{F}s_1^{out}, \mathcal{F}s_2^{out}, \ldots, \mathcal{F}s_N^{out}]$ is not continuous in a mathematical sense at points $\{t_k^n\}_{k=1}^{P_n}, n = 1, \ldots, N$, due to the expression of operator $\mathcal{F}$ (1). Therefore, for any sequence of spike trains $\{s_1^{out}, \ldots, s_N^{out}\}$, $\boldsymbol{x}(t)$ in not bandlimited. This can also be explained by viewing the values of operator $\mathcal{F}$ as the output of an exponential filter with impulse response $h(t) = e^{-\frac{t}{\tau_s}}$, given a train of Dirac delta pulses $\sum_{k=1}^{P_n} \delta(t - t_k^n)$. Given that the filter is not ideal, its output has arbitrarily large frequency components, and thus the samples $\{\boldsymbol{x}(kT)\}$ are affected by aliasing, due to Shannon's law. This leads to computing weights $\boldsymbol{w}_{opt}$ that are deviated form the theoretical optimal values, as well as an imprecise final output prediction $y_{\boldsymbol{w}_{opt}}(t)$.

Moreover, in practice not all synaptic connections of the Readout are relevant to a particular task, so that training the weights of all possible connections from the Liquid neurons to the Readout can easily lead to overfitting.

There are a few variations of LS that introduce an additional parameter, also known as hyperparameter, in order to control the effective complexity of the model and to reduce overfitting. Some of the standard methods doing this are LS with L^2 regularization, or ridge regression (RR), LS with L^1 regularization, or lasso, and early stopping (ES). The regularization parameter for RR and lasso, and the number of iterations for ES are typically tuned to minimise the prediction error on the validation dataset (Bishop, 2006). These methods can lead to a Readout with smaller weights, or fewer presynaptic connections to the Liquid.

However, computing the Readout weights with RR, lasso or ES is affected by approximation error, as a result of the aliasing effect caused by uniform sampling. This

leads to Readout presynaptic connections to neurons that are less relevant for the computing task. Furthermore, the output spikes of a biological neural network do not lie on a grid of uniformly spaced time points, and therefore are not directly compatible with the training methods above.

3 A New LSM Training Approach using Precise Times

3.1 The Carnell-Richardson Spike Train Space

The space $\mathbb{S}_0$ is not a linear space because it does not allow any operations between spike trains. To overcome this problem, this space is extended to the Carnell-Richardson spike train space (Carnell & Richardson, 2005)

$$\mathbb{S} = \left\{ s = \{(a_k, t_k)\}_{k=1}^{P}, P \geq 1, t_k, a_k \in \mathbb{R}, t_k \neq t_l, \forall k, l \in \{1, \ldots, P\}, k \neq l \right\}.$$

Carnell & Richardson (2005) have proven that $\mathbb{S}$ is an inner product space, where the vector sum, scalar multiplication and inner product of two spike trains $s_1, s_2 \in \mathbb{S}$ are defined as

$$s_1 + s_2 = \{(a_k^1, t_k^1)\}_{k=1}^{M_1} \cup \{(a_k^2, t_k^2)\}_{k=1}^{M_2},$$

$$\alpha \cdot s = \{(\alpha \cdot a_k, t_k)\}_{k=1}^{M}, \forall \alpha \in \mathbb{R},$$

$$\langle s_1, s_2 \rangle_{\mathbb{S}} = \sum_{k_1=1, k_2=1}^{k_1=M_1, k_2=M_2} a_{k_1}^1 a_{k_2}^2 \cdot e^{-\frac{|t_{k_1}^1 - t_{k_2}^2|}{\tau_s}},$$

where $\tau_s > 0$ is a scaling factor. The inner product $\langle \cdot, \cdot \rangle_{\mathbb{S}}$ generates a norm $\| \cdot \|_{\mathbb{S}}$ satisfying $\|s\|_{\mathbb{S}} = \sqrt{\langle s, s \rangle_{\mathbb{S}}}, \forall s \in \mathbb{S}$. Figure 2 illustrates an example of a linear operation between two randomly generated spike trains $s_1, s_2 \in \mathbb{S}$, presented comparatively with the equivalent operation in $L^2(\mathbb{R})$.

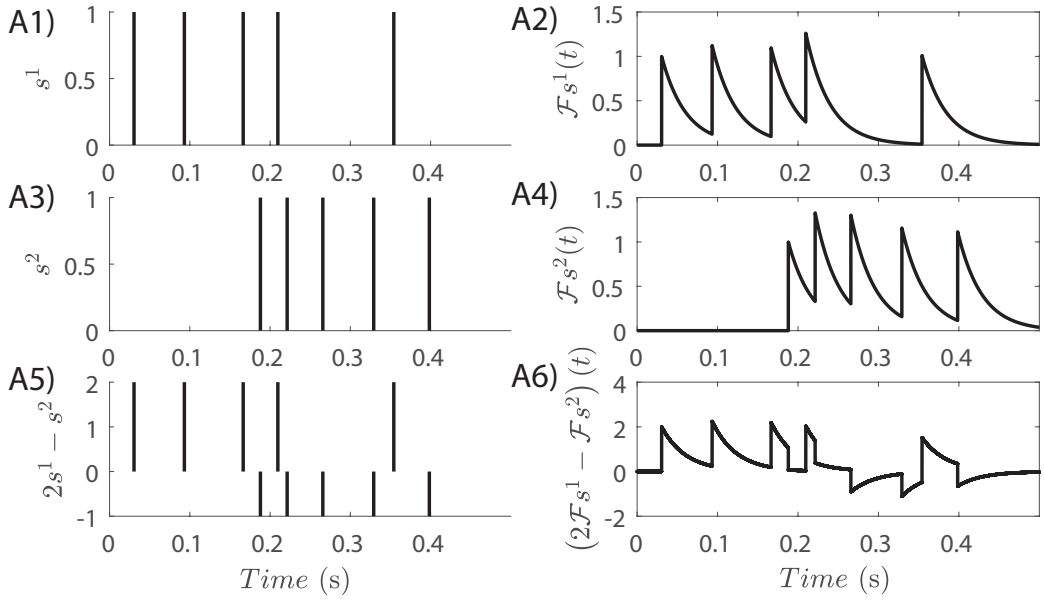

Figure 2: An example of a linear operation in $\mathbb{S}$. Two spike trains s^1, $s^2 \in \mathbb{S}$, and their corresponding elements $\mathcal{F}s^1, \mathcal{F}s^2 \in L^2(\mathbb{R})$, are generated in time interval $[0, 0.5 \text{ s}]$ (A1-4). The equivalent linear operations in the two spaces $2s^1 - s^2$ and $2\mathcal{F}s^1 - \mathcal{F}s^2$ are depicted in (A5-6).

A spike train $s_0 = \{t_k\}_{k=1}^P \in \mathbb{S}_0$, as defined by the standard method, can be mapped uniquely onto an element $s \in \mathbb{S}$, such that $s = \{(1, t_k)\}_{k=1}^P$. Maass et al. (2002) have defined a metric d on $\mathbb{S}_0$

$$d(s_1, s_2) = \left[\int_{\mathbb{R}} \left[\left(\mathcal{F}s_1\right)(t) - \left(\mathcal{F}s_2\right)(t)\right]^2 dt\right]^{1/2},$$

where $\mathcal{F} : \mathbb{S}_0 \to L^2(\mathbb{R}), \mathcal{F}s = \sum_{k=1}^P e^{-\frac{t-t_k}{\tau_s}} \cdot 1_{[t_k^n, \infty)}(t)$ denotes the output of a linear filter with exponential decay and time constant τ_s, given spiking input s. The norm $\|\cdot\|_{\mathbb{S}}$ relates to metric d as follows $\|s_1 - s_2\|_{\mathbb{S}}^2 = 2 \cdot d(s_1, s_2)^2, \forall s_1, s_2 \in \mathbb{S}_0$. However, in a practical setting, the metric d is approximated by $d_{\Delta T}$, computed on a uniform grid with

sampling time ΔT. Then the following holds

$$\lim_{\Delta T \to 0} d_{\Delta T}(s_1, s_2) = \frac{1}{\sqrt{2}} \|s_1 - s_2\|_{\mathbb{S}}.$$

In order to show the disadvantage in computing $d_{\Delta T}$, we generated two random spike trains s_1 and s_2 with 100 spike times each. We then computed $\|s_1 - s_2\|_{\mathbb{S}}$ and $d_{\Delta T}(s_1, s_2)$ for 100 values of ΔT on $[1 \text{ ms}, 100 \text{ ms}]$, and $\tau_s = 30$ ms. The results, depicted in Figure 3, show that the values of $d_{\Delta T}(s_1, s_2)/\sqrt{2}$ oscillate around $\|s_1 - s_2\|_{\mathbb{S}}$ as $\Delta T \to 0$. However, the computing time for $d_{\Delta T}$ increases exponentially with $1/\Delta T$. Thus, at the sampling interval of 2 ms, which is used to simulate the LSM, the spike based metric results in a similar value to the standard metric, but runs three times faster.

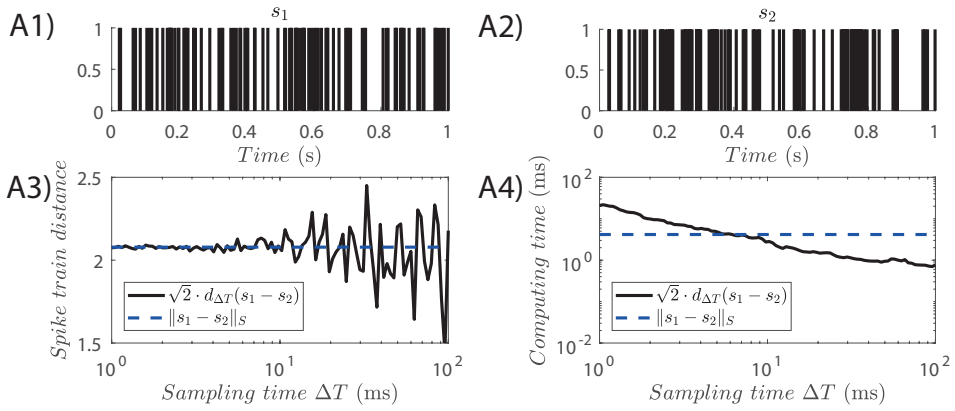

Figure 3: Comparison between the Carnell-Richardson spike train distance $\|s_1 - s_2\|_{\mathbb{S}}$ and the standard metric $d_{\Delta T}(s_1 - s_2)$: two randomly generated spike trains s_1, s_2 (A1,2) and their corresponding distance calculated with the two metrics (A3,4).

3.2 The Proposed LSM Architecture and Training Method

We propose a new spike time based Readout architecture, which does not require the bank of filters $\mathcal{F}$ (Figure 4).

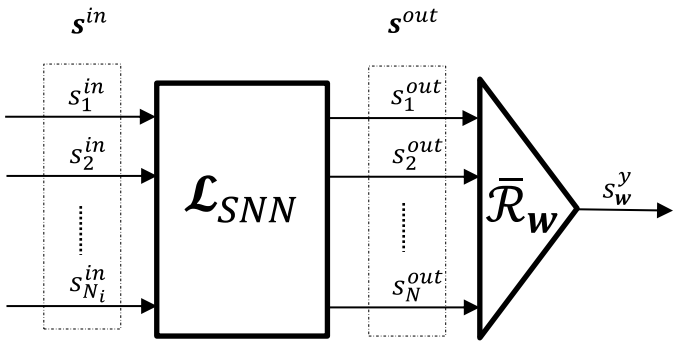

Figure 4: Block diagram of the proposed architecture used for training LSMs , consisting of two blocks connected in series: the Liquid $\mathcal{L}_{SNN}$ and the proposed spike based Readout $\bar{\mathcal{R}}_w$.

The Readout $\bar{\mathcal{R}}_{\boldsymbol{w}}$ is defined using the operations in $\mathbb{S}$ as

$$\bar{\mathcal{R}}_{\boldsymbol{w}} \boldsymbol{s}^{out} = \sum_{n=1}^{P_n^{out}} w_n s_n^{out} = s_{\boldsymbol{w}}^{y}.$$

Let s^{y*} be a target spike train. Then the optimal $\boldsymbol{w}$ in the least squares sense is

$$\bar{\boldsymbol{w}}_{opt} = \underset{\boldsymbol{w}}{\text{argmin}} \|s^{y*} - s_{\boldsymbol{w}}^{y}\|_{\mathbb{S}},$$

where $\|\cdot\|_{\mathbb{S}}$ denotes the standard norm in $\mathbb{S}$.

The proposed architecture can be extended to learn continuous target signals. To this end, the following results demonstrate that any continuous target function $y^* \in L^2(\mathbb{R})$ can be mapped uniquely onto a spike train $s^{y*} \in \mathbb{S}$.

Theorem 1. *Let $\mathbb{S}^{out}$ denote the subset of $\mathbb{S}$ generated by the outputs of the SNN, such that $\mathbb{S}^{out} = \text{span}\{s_1^{out}, \ldots, s_N^{out}\} \subset \mathbb{S}$. Let $\mathcal{F} : \mathbb{S}^{out} \to L^2(\mathbb{R})$ be an operator defined by*

$$\mathcal{F}s = \sum_{k=1}^{P} a_k e^{-\frac{t-t_k}{\tau_s}} \cdot 1_{[t_k,\infty)}(t), \forall s \in \mathbb{S}^{out}, s = \{(a_k, t_k)\}_{k=1}^{P}. \tag{3}$$

Moreover, let $\mathbb{FS}^{out}$ denote the subset of $L^2(\mathbb{R})$ generated by the filtered outputs of the SNN, such that $\mathbb{FS}^{out} = \text{span}\{\mathcal{F}s_1^{out}, \ldots, \mathcal{F}s_N^{out}\}$. Then the following mapping is well defined

$$\mathcal{M} : L^2(\mathbb{R}) \to \mathbb{S}^{out}, \mathcal{M}(y) = \mathcal{F}^{-1}\mathcal{P}_{\mathbb{FS}^{out}}(y), \forall y \in L^2(\mathbb{R}), \tag{4}$$

where $\mathcal{P}$ denotes the projection operator.

Proof. See Appendix 1. □

Theorem 1 defines a mapping that allows converting any continuous target output function $y^*(t)$ into a unique target output spike train s^{y*}. The operator $\mathcal{F}$ in (3) is the extension of the filtering operator in (1) to the more general space $\mathbb{S}$. The following result assesses the prediction accuracy of the proposed method relative to the standard method for continuous target functions.

Theorem 2. *Let $y^* \in L^2(\mathbb{R})$ and let $\boldsymbol{w}_{opt}$ be the vector of weights computed for the standard architecture, such that $\boldsymbol{w}_{opt} = \underset{\boldsymbol{w}}{\text{argmin}} \|y^* - \mathcal{R}_{\boldsymbol{w}}\boldsymbol{\mathcal{F}}\boldsymbol{s}^{out}\|_{L^2}$. It follows that*

$$\boldsymbol{w}_{opt} = \underset{\boldsymbol{w}}{\text{argmin}} \|s^{y*} - \bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{out}\|_{\mathbb{S}} = \bar{\boldsymbol{w}}_{opt},$$

where $s^{y*} = \mathcal{M}(y^*)$, $\mathcal{M}(y^*) = \mathcal{F}^{-1}\mathcal{P}_{\mathbb{FS}^{out}}(y^*)$, $\mathcal{P}$ denotes the projection operator and $\mathbb{FS}^{out} = \text{span}\{\mathcal{F}s_1^{out}, \ldots, \mathcal{F}s_N^{out}\}$.

Proof. See Appendix 1. □

Corollary 1. *Theorem 2 proves that the proposed methodology achieves, in theory, the same accuracy as the state-of-the-art method when learning continuous target signals. In practice, however, the accuracy of the standard method is lower because it is affected by the approximation error introduced when calculating* $\boldsymbol{w}_{opt}$ *and* $y_{\boldsymbol{w}_{opt}}(t)$ *from uniform samples, which doesn't affect the proposed method.*

3.3 The Orthogonal Forward Regression with Spike Trains (OFRST) Algorithm

The optimisation problem addressed by the proposed method is to learn a continuous target output $y^*(t)$ given a SNN of size N. Let $\{s_k^{out}\}_{k=1}^{N}$ denote the outputs of the SNN in response to stimuli $\{s_k^{in}\}_{k=1}^{N_{in}}$. Computing the optimal $\boldsymbol{w}_{opt}$ in the least squares sense (Maass et al., 2002) leads to many non zero weights that are not particularly relevant for the learning task and overfit the data. Furthermore, the standard methods that address this problem using regularization or early stopping lead to weights that are deviated from the theoretical optimal weights as a result of the approximation error.

Theorem 1 demonstrates that the problem addressed here can be reduced to learning a target spike train s^{y*}, uniquely derived from the continuous target $y^*(t)$. This leads to a more precise estimation of weights $\boldsymbol{w}_{opt}$ (Theorem 2). Here we introduce a greedy selection algorithm for the spike trains that are most relevant for the learning task, called Orthogonal Forward Regression with Spike Trains (OFRST). The OFRST algorithm is inspired by the orthogonal forward regression (OFR) for finite dimensional spaces (Chen et al., 1989). The remaining part of this section will first present the classical OFR and then the proposed OFRST algorithm.

Given vectors $\{x_1, \ldots, x_N\}$ and target vector y^*, the OFR algorithm aims to identify

a subset $\{x_{\ell_1} \ldots, x_{\ell_p}\}$ and an estimate of the parameters $\{w_{\ell_1}, \ldots, w_{\ell_p}\}$ that fits the data y^*.

At the first stage, y^* is projected onto basis vectors $\{x_1, \ldots, x_N\}$. Then the error-reduction-ratio (ERR) is calculated for each vector, defined as

$$ERR_k^{(1)} = \frac{\langle x_k, y^* \rangle^2}{\|x_k\|^2 \cdot \|y^*\|^2}.$$

The magnitude of $ERR_k^{(1)}$ represents the proportion of the dependant variable variance explained by x_k. A geometrical interpretation of the ERR is depicted in Figure 5 for the simplified case where $x_k \in \mathbb{R}^2, k = 1, 2$, and $y^* \in \mathbb{R}^2$. The maximum ERR, computed as $ERR_1 = ERR_{\ell_1}^{(1)} = \max_{k=1,\ldots,N}\{ERR_k^{(1)}\}$, leads to the selection of $x_1^{\perp} = x_{\ell_1}$ as the basis for the one-dimensional space E^1.

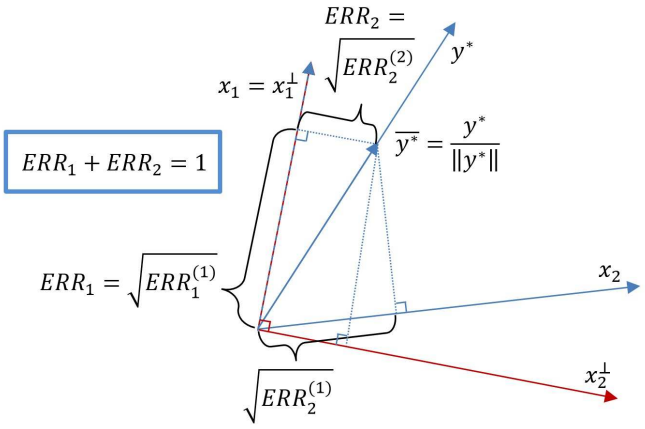

Figure 5: Geometrical interpretation of OFR for the simplified two-dimensional scenario. In this case $ERR_1^{(1)} > ERR_2^{(1)}$ implies that x_1 explains a larger proportion of the variance of target output y^*.

At the second stage, the rest of the vectors $\{x_i\}_{i=1,\ldots,N, i \neq \ell_1}$ are projected, through Gram-Schmidt orthogonalization, into a $(N-1)$-dimensional space orthogonal on E_1.

Subsequently, the vector x_{ℓ_2} is selected and orthogonalised with the Gram-Schmidt procedure to compute $x_2^{\perp}$. The vectors $x_1^{\perp}$ and $x_2^{\perp}$ form the basis for two-dimensional space E_2. Similarly, at stage number p, the vector x_{ℓ_p} is selected, which is used to define the p-dimensional space E_p with orthogonal basis $\{x_i^{\perp}\}_{i=1\ldots,p}$. The detailed algorithm is given in Appendix 2.

The OFRST algorithm closely follows the steps of the OFR algorithm, implemented for the Carnell-Richardson spike train space $\mathbb{S}$. Initially, let $s_k^{\perp(1)} = s_k^{out} \in \mathbb{S}, \forall k = 1, \ldots, N$, be the complete set of SNN outputs. The most significant spike train $s_{\ell_1}^{out}$ is defined as the one that maximises $ERR_k^{(1)}$, where $ERR_k^{(i)}$ denotes the error-reduction-ratio (ERR) of term k at iteration i, defined as

$$ERR_k^{(i)} = \frac{\left\langle s_k^{\perp(i)}, s^{y*} \right\rangle_{\mathbb{S}}^2}{\|s_k^{\perp(i)}\|_{\mathbb{S}}^2 \cdot \|s^{y*}\|_{\mathbb{S}}^2}.$$

Subsequently, the set $\{s_k^{\perp(2)}\}_{k=1,k\neq\ell_1}^N$ is computed by orthogonalising the remaining output spike trains against $s_{\ell_1}^{out}$ using the Gram-Schmitt routine.

The process continues iteratively. At every iteration i, the algorithm selects the next most significant spike train $s_{\ell_i}^{out}$ such that $\ell_i = \underset{k}{\text{argmax}} \left(ERR_k^{(i)} \right)$, and generates the set $\{s_{\ell_1}^{out}, \cdots, s_{\ell_i}^{out}\}$ of significant SNN outputs and the corresponding vector of weights $\boldsymbol{w}^{(p)}$. Subsequently, the set $\{s_k^{\perp(i)}\}_{k=1,k\neq\ell_1,\ldots,\ell_i}^N$ is computed from the remaining spike trains through orthogonalisation. The process continues until $p = N$. The final number of presynaptic neurons is selected as the smallest p that leads to the maximum prediction accuracy on the validation dataset. The detailed algorithm is given in Appendix 3.

4 Numerical examples

The proposed new Readout and associated training algorithm is evaluated in comparison with the standard architecture trained with LS, RR, lasso and ES.

Additional numerical examples show the advantage of using a spike based Readout and the advantage of selecting the Readout presynaptic neurons using OFRST. The benefit of the proposed method is also demonstrated for two additional examples with real world data. First, OFRST is compared against the standard methods for a multi-label classification problem using multi-array recordings from the primary visual cortex of the monkey. Second, the advantage of the proposed method on a speech recognition task is shown using data from the TI-46 corpus database of spoken digits.

The LSM was simulated using the toolbox described in (Natschläger et al., 2003). The Liquid consists of leaky integrate-and-fire neurons, 20% of which were randomly selected to be inhibitory (Maass et al., 2002). The connection probability between neurons a and b is defined as $C \cdot e^{-(D(a,b)/L)^2}$, where $D(a,b)$ denotes the Euclidian distance between the neurons, $L = 2$ is a parameter that controls the average number of connections and the average distance between neurons, and C, depending on whether the neurons are excitatory (E) or inhibitory (I), is 0.3 (EE) , 0.2 (EI) , 0.4 (IE) , 0.1 (II). The synaptic transmission is given by the dynamic model proposed in (Markram, Wang & Tsodyks, 1998). The input is injected into 30% randomly chosen neurons in the Liquid with an input gain of 0.1. For the standard Readout architecture, the time constant of the exponential filters is $\tau_s = 30$ms. The LSM was simulated using the default sampling time of 0.2ms (Maass et al., 2002). The simulations were carried out in Matlab Version

8.6 $(R2015b)$ on a 3 GHz Intel Core i7-5960X 8 core PC workstation.

Example 1. Binary classification - comparison with the standard methods.

This example compares the performance achieved by a standard LSM with the Readout parameters estimated using the LS, RR, lasso and ES with that of a LSM comprising a spike-based Readout trained using the proposed OFRST method. The LSM consists of 240 neurons spatially organised as a lattice with dimensions 15x4x4.

The task is to discriminate between two spike train templates using the SNN responses. The templates are two instances of a Poisson point process with rate 20 Hz, depicted in Figure 6.

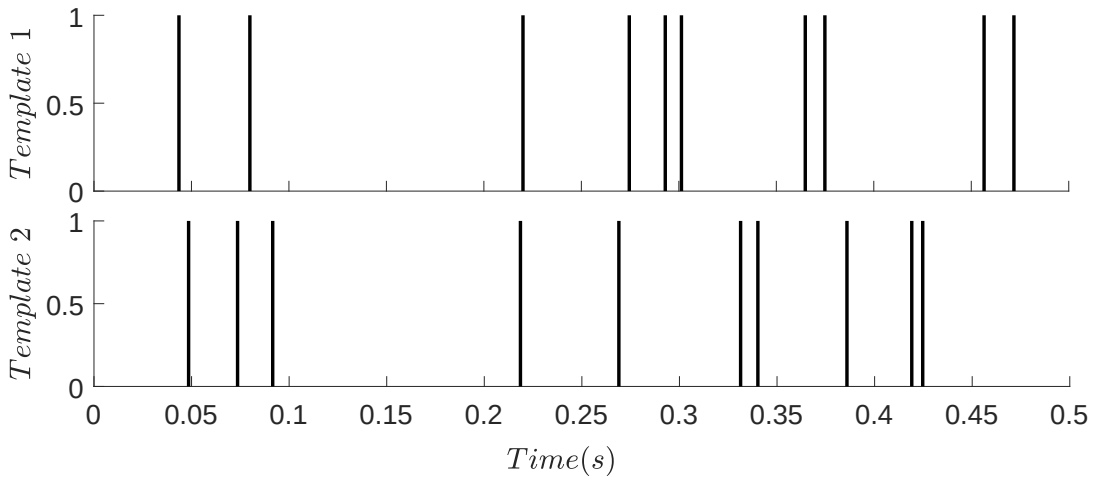

Figure 6: The input templates used for classification, generated as Poisson spike trains with frequency 20 Hz over time interval $[0, 0.5$ s$]$.

The inputs are generated in time interval $[0, 0.5\ s]$ by jittering one of the two templates, where the jitter noise is drawn from the Gaussian distribution with zero mean

and standard deviation 6 ms. A number of 100 jittered templates were generated for each class, of which 50 were used for training and 50 for validation. The two classes of inputs are assigned the target output labels $y(t) = 1$ (template 1) and $y(t) = -1$ (template 2), $t \in [0, 0.5\ s]$.

The input-output mappings are learned with the LSM by estimating the standard Readout parameters using LS, RR, lasso and ES, where the sampling time is $\Delta T = 20$ ms (Maass et al., 2002; Verstraeten et al., 2005). Subsequently, the spike time based Readout is trained using OFRST. The regularization parameter for RR and lasso, the number of steps for ES and the number p of presynaptic neurons for OFRST are computed using a line search that maximises the prediction accuracy on the validation dataset.

The classification accuracies for RR, lasso, ES and OFRST were evaluated as a function of the hyperparameter and averaged over 100 trials. Each trial consisted in a different Liquid and a different instance of jitter applied to the input. The results are depicted in Figure 7.

In the case of the OFRST algorithm the results show that, on average, the accuracy drops when using more than 36 Readout presynaptic connections, or equivalently training for more than 36 iterations. This suggests that, on average, more than 30 Readout presynaptic connections lead to overfitting the data. This result mimics what has been observed experimentally in cortical circuits, where only a small number of cortical neurons project to different areas of the central nervous system (Thomson et al., 2002; Häusler & Maass , 2007).

The accuracy for each method was optimised with a different hyperparameter on

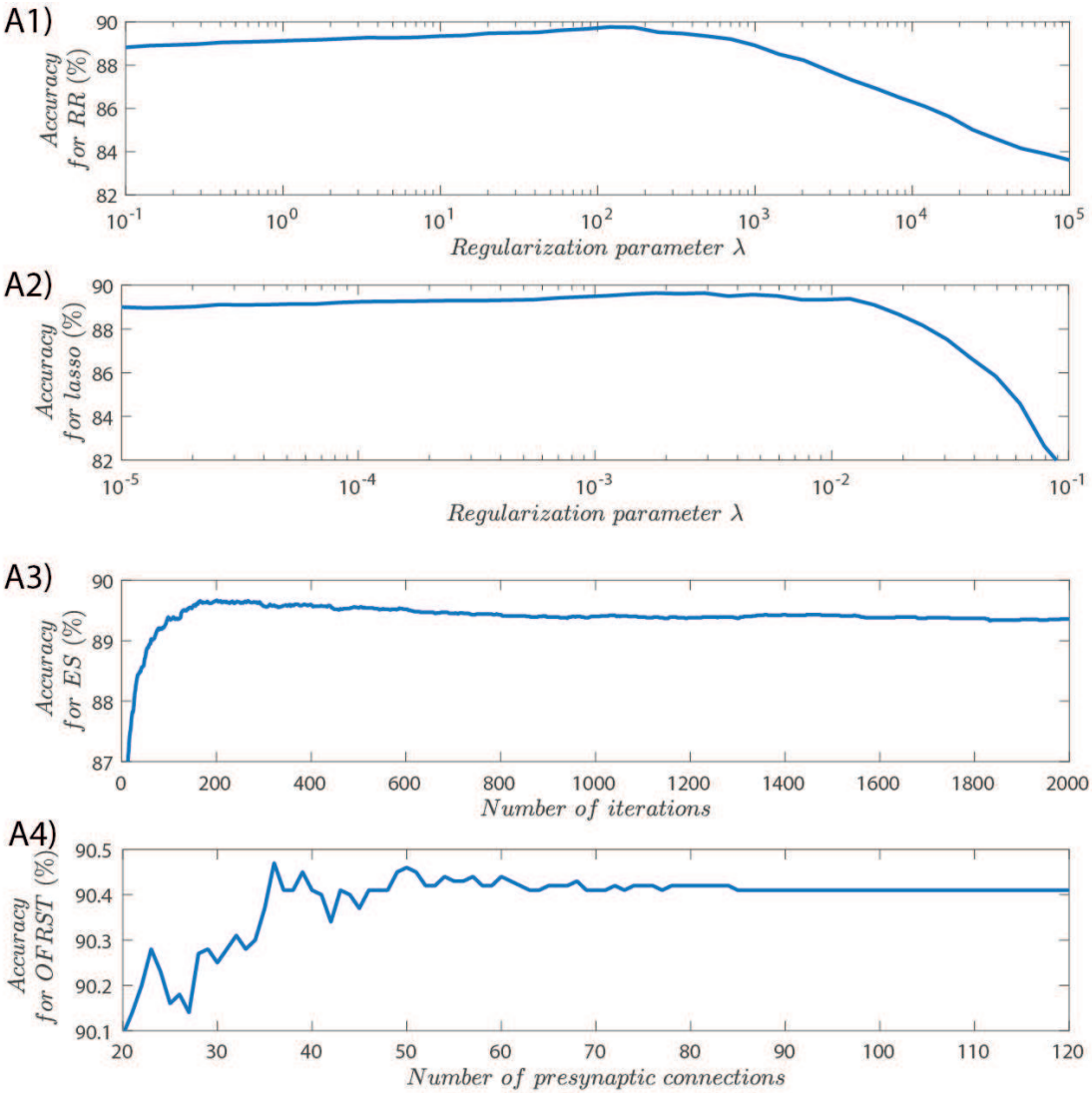

Figure 7: Binary classification with RR (A1), Lasso (A2), ES (A3), and OFRST (A4), as a function of the regularization parameter. The average accuracies were computed for each method on the validation dataset over 100 trials.

each simulation trial. The classification accuracies achieved by all the methods over 100 trials are given in Table 1. The results show that, on average, OFRST has the highest accuracy from all methods while using the smallest number of synapses.

Table 1: Binary classification results using pools of 240 neurons. Comparison between least squares, ridge regression, lasso and early stopping, implemented for the standard Readout, and the proposed OFRST method for the spike based Readout. The mean ($\pm$ standard deviation) is computed for each method over 100 trials.

Training method	Total number of Readout connections	Accuracy
Least squares	56.46 (± 20.3)	88.4% ($\pm 7.28\%$)
Ridge regression	56.46 (± 20.3)	91.27% ($\pm 7.07\%$)
Lasso	40.32 (± 23.07)	91.15% ($\pm 6.8\%$)
Early stopping	56.46 (± 20.31)	91.28% ($\pm 7.07\%$)
OFRST	15.05 (± 11.03)	92.15%($\pm 6.92\%$)

Example 2. Binary classification - benefits of learning with exact spike times.

In this example we compare the classification accuracy of the proposed Readout trained with the OFRST method to that of the standard Readout trained with LS, RR, lasso, ES and classical OFR (Billings et al., 1989) on the same binary classification task as in Example 1, but for different values of the sampling time ΔT.

The training and validation datasets were generated as in Example 1. For the OFR and OFRST methods the number the presynaptic neurons, which represent the regressors in the standard OFR algorithm (Billings et al., 1989), is the smallest number that achieves maximum accuracy on the validation dataset. In order to evaluate the effect of the sampling time ΔT on the performance of the standard Readout, the training was

performed for several sampling times ranging from 0.2ms to 30ms. The accuracies for all the methods, as a function of the sampling time, are depicted in Figure 8. Each data point represents an average value over 10 different Liquids.

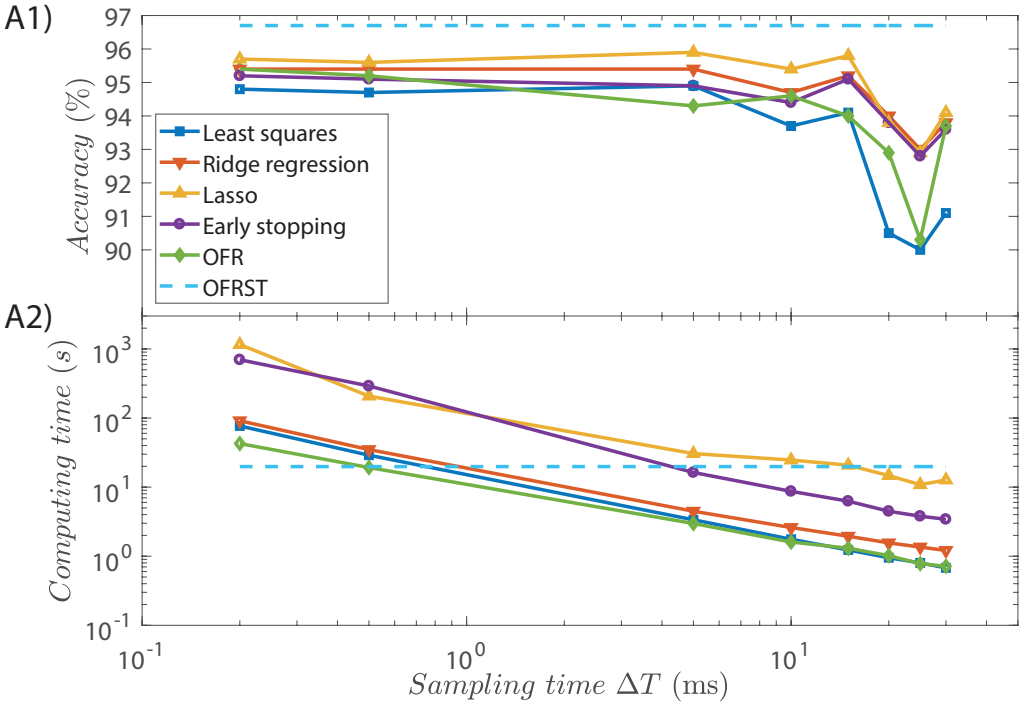

Figure 8: Comparison between the proposed OFRST method and LS, RR, lasso, ES and OFR for different values of the sampling time ΔT: accuracies (A1) and computing times (A2).

The results show that the classification accuracy for the LS, RR, lasso, ES and classical OFR methods can be increased by decreasing the sampling time. However, the performance is still below the one achieved by the OFRST method, which selects presynaptic connections using the exact spike times generated by the Liquid neurons. The difference in accuracy between OFR and OFRST, which is expected to vanish when $\Delta T \to 0$, shows directly advantage in processing exact spike times.

Interestingly, even for $\Delta T = 0.2$ ms, which is the sampling time used for simulating the LSM, OFRST still performs significantly better than the other methods. This is because all the training methods based on the standard Readout architecture are subject to an approximation error when estimating the weights, for any $\Delta T > 0$.

Example 3. Binary classification: selecting relevant presynaptic neurons.

This numerical example evaluates the performance of the OFRST in selecting the relevant presynaptic partners using exact spike timing. The SNN used in this example has a reservoir consisting of two sub-networks that are disconnected from one another, each sub-network consisting of a different pool of 135 spiking neurons generated as in examples 1 and 2. Two templates were generated as Poisson spike trains with frequency of 20 Hz over interval $[0, 0.5\text{s}]$. The first pool $R_1 = \{r_1, \ldots, r_{135}\}$ receives 200 inputs generated by jittering the two spike train templates, 100 for each class, of which 50 were used for training and 50 for validation. The jitter noise is drawn from the Gaussian distribution with zero mean and standard deviation 1 ms. The second pool $R_2 = \{r_{136}, \ldots, r_{270}\}$ receives a number of 200 new jittered inputs generated from the same two templates but in a different order selected at random.

The task is to classify the inputs to sub-network R_1 using the neuron outputs from the full reservoir. The OFRST algorithm is compared with the LS, RR, lasso, ES and the OFR algorithms, which use the standard filtered spike train outputs.

In essence, when solving the binary classification problem, the algorithms should only select neurons from R_1 as pre-synaptic partners of the Readout unit. The training results, computed for 100 different Liquids and instances of jitter, are summarised in

Table 2.

Table 2: Binary classification results for least squares, ridge regression, lasso, early stopping, standard OFR and OFRST using two unconnected sub-networks with 135 neurons each. The reported values represent means ($\pm$ standard deviations) computed over 100 trials.

Training method	Total number of Readout connections	Percentage connections to sub-network R_1	Accuracy
Least squares	67.6 (± 21.72)	49.9% ($\pm 1.6\%$)	95.7% ($\pm 4.4\%$)
Ridge regression	67.6 (± 21.72)	49.9% ($\pm 1.6\%$)	97.1% ($\pm 3.8\%$)
Lasso	49.6 (± 26.5)	58.9% ($\pm 14.19\%$)	97.6% ($\pm 3.4\%$)
Early stopping	67.6 (± 21.72)	49.9% ($\pm 1.6\%$)	96.8% ($\pm 4\%$)
OFR	13.1 (± 10)	86.7% ($\pm 14.6\%$)	96.8% ($\pm 4.2\%$)
OFRST	9.45 (± 9.4)	93.6% ($\pm 10.7\%$)	97.8% ($\pm 3.8\%$)

The results show that OFRST achieves the highest accuracy among all methods using the least number of Readout presynaptic connections, and the highest percentage of connections to the correct sub-network R_1. Only OFR and OFRST achieve a percentage of connections to R_1 of over 90%, while all the other methods result in percentages just above chance.

Example 4. Motion direction decoding using multi-electrode array recordings from the primary visual cortex.

Here we use the proposed methodology to decode stimulus features using simultaneous multi-electrode array recordings of visually evoked activity from the primary visual cortex of three anesthetized macaque monkeys. The data were downloaded from the CRCNS online database (Kohn & Smith, 2016). Here we use the recordings from

monkey number 1.

The stimuli were full-contrast drifting sinusoidal gratings at 12 orientations spaced equally $(0°, 30°, 60°, \ldots, 270°)$. Each stimulus was presented 200 times, for a duration of 1.3 s per trial (Smith & Kohn, 2008; Kelly et al., 2010). The spiking train responses of 106 neurons were simultaneously recorded using a Utah multi-electrode array and spike-sorted offline (Smith & Kohn, 2008; Kelly et al., 2010). In this example we only use the first 200 ms from all recording trials, which is the time reported for visual categorisation tasks in primates (Fabre-Thorpe, 1998; Hung et al., 2005). A recording trial for the $0°$ drifting bar stimulus is depicted in Figure 5.

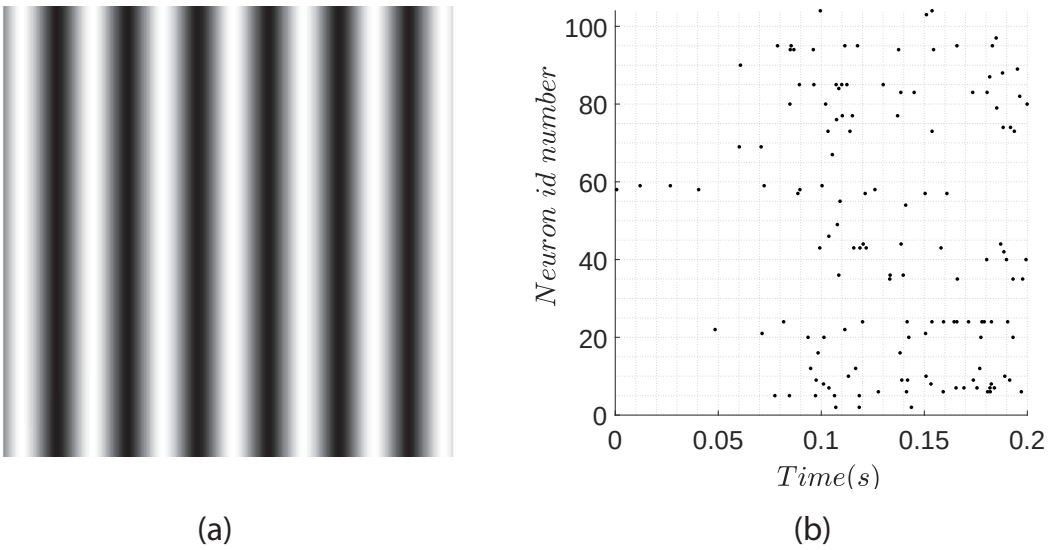

Figure 9: The first sweep of experimental data used in Example 4: a) The first frame of a drifting bar stimulus oriented at $0°$, b) Raster plot showing the response of 106 neurons, as a function of time.

The decoding task is to predict the stimulus orientation based on the recorded neural activity. The task is formulated as a multi-label classification problem. Each of the 12

directions was assigned a target label $(1-12)$ and a Readout. Each Readout processes the outputs of the 106 recorded neurons, which play the role of the Liquid spike train outputs.

The data (2400 trials) was randomly divided into equal datasets for training and validation, such that each dataset comprises 100 trials with each of the 12 inputs. The 12 Readouts were trained using the "one-to-all" method, also known as "1-hot coding", where only one Readout generates an output "1" at any given time. Specifically, the target output for each Readout satisfies $y^*(t) = 1$ when the input direction label matches the Readout label, and $y^*(t) = -1$ for any other direction. The overall prediction is given by the label of the Readout with maximum average value. The training data for each Readout consists of 100 trials from the target class and 100 trials evenly distributed among all other classes. The parameters of the 12 Readouts were tuned using the LS, RR, lasso, ES and the OFRST methods. Considering the large number of possible connections, here the OFRST algorithm for each Readout was stopped when the criterion $ERR_p < \zeta$ was met, where $\zeta = 4 \cdot 10^{-4}$ is a parameter determined using line search and ERR denotes the error reduction ratio (see Appendix 4). Essentially, this means that each Readout only connects to presynaptic neurons whose outputs contribute more than 0.04% to the variance change in the target output. The regularization parameters for RR and lasso, and the number of iterations for ES were selected using line search to maximise the accuracy on the validation dataset. The final accuracy, computed on the validation dataset, is defined as the percentage of correctly decoded input directions.

We compared the decoding performance with standard Readouts, trained with LS, RR, lasso and ES, to the performance with spike time based Readouts, trained with the

OFRST algorithm described in subsection 3.3. The results are summarised in tables 3 and 4.

Table 3: Multi-label classification accuracies with the standard LS, RR, lasso, ES methods and the proposed OFRST method.

Training method	Readout accuracy (%)												Final accuracy
	1	2	3	4	5	6	7	8	9	10	11	12	
LS	86	84	83	81	82	84	85	83	84	87	80	78	58.17%
RR	88	88	85	87	84	80	84	88	84	85	85	82	61.75%
Lasso	83	83	85	87	83	82	83	90	83	86	83	81	59.5%
ES	89	85	87	87	82	82	88	87	88	86	82	83	61.33%
OFRST	88	88	86	86	84	86	88	86	85	89	82	84	67.58%

Table 4: Number of presynaptic connections selected for each Readout with the standard LS, RR, lasso, ES methods and the proposed OFRST method.

Training method	Number of Readout connections											
	1	2	3	4	5	6	7	8	9	10	11	12
LS	106	106	106	106	106	106	106	105	105	106	106	106
RR	106	106	106	106	106	106	106	106	106	106	106	106
Lasso	69	71	62	77	69	70	68	64	75	77	72	72
ES	106	106	106	106	106	106	106	106	106	106	106	106
OFRST	49	61	63	61	66	59	58	61	60	60	61	59

The results show that the proposed spike time based Readout, trained with the OFRST algorithm, performs significantly better than the standard Readout architecture trained with LS, RR, lasso, or ES, while using significantly fewer neuron connections.

Example 5. Speech recognition.

In this example we use the proposed OFRST methodology to perform speech recog-

nition. The data is a subset of the TI-46 corpus of isolated spoken digits, consisting of 500 utterances of digits "zero" to "nine" spoken by 5 different female speakers [1] (Doddington & George, 1981; Schalk, 1982).

The decoding task is to predict the digit number using a LSM, formulated as a multi-label classification problem (Verstraeten et al., 2005). The LSM in this example has 135 neurons, spatially organised as a lattice with dimensions 15x3x3. As before, each digit was assigned a target label (1-10) and a Readout unit.

The data is preprocessed using the Lyon passive ear model, which is a model of the human inner ear, or cochlea. This model consists of three processing stages: a band-pass filter-bank, inspired by the human ear sensitivity to certain frequencies, half way rectification, and automatic gain control, which model the hair cells in the cochlea (Lyon , 1982). Subsequently, the continuous output of the Lyon passive ear model is converted into a spike train using an algorithm called Ben's spiker algorithm (BSA) (Schrauwen, 2003). This preprocessing front-end, consisting of the Lyon passive ear model in series with BSA, has been used successfully to address this type of speech recognition problem using an LSM (Verstraeten et al., 2005, 2007; Yin et al., 2012).

The data was divided in two sets: a training set of size 300 and a validation set of size 200, such that the recordings of each speaker are proportionally distributed between the two sets. As before, the 10 Readouts were trained using the "one-to-all" method. The training data for each Readout consists of 60 recordings from the corresponding target class and 60 recordings evenly distributed among all other classes. The final accuracy

[1] Downloaded from the Linguistic Data Consortium website: http://www.ldc.upenn.edu.

is defined as the percentage of correctly recognised digits in the validation dataset.

The 10 Readout units were trained using LS, RR, lasso, ES and the OFRST method. The stop criterion for the OFRST algorithm is $ERR_{p+1} - ERR_p < \zeta$. The parameter ζ and the regularization parameters for RR, lasso and ES were tuned for each Readout on the validation dataset using line search.

The comparative performance of the spike time based Readouts trained with OFRST, and the standard Readouts trained with LS, RR, lasso and ES are summarised in tables 5 and 6.

Table 5: Multi-label classification accuracies for the LS, RR, lasso, ES methods and the proposed OFRST method, computed as mean ($\pm$ standard deviation) for 10 different Liquid simulations.

Training method	Readout accuracy (%)					
	1	2	3	4	5	6
LS	$88(\pm6)$	$96(\pm3)$	$96(\pm2)$	$95(\pm4)$	$94(\pm4)$	$95(\pm4)$
RR	$95(\pm3)$	$92(\pm3)$	$91(\pm6)$	$93(\pm5)$	$95(\pm3)$	$93(\pm4)$
Lasso	$96(\pm2)$	$90(\pm2)$	$87(\pm7)$	$92(\pm7)$	$95(\pm4)$	$91(\pm4)$
ES	$96(\pm3)$	$92(\pm3)$	$92(\pm5)$	$94(\pm6)$	$95(\pm3)$	$93(\pm5)$
OFRST	$94(\pm4)$	$95(\pm3)$	$100(\pm1)$	$91(\pm4)$	$98(\pm2)$	$93(\pm7)$

Training method	Readout accuracy (%)				Final accuracy
	7	8	9	10	
LS	$97(\pm3)$	$94(\pm1)$	$92(\pm5)$	$95(\pm4)$	$73.4\%(\pm5.5\%)$
RR	$100(\pm1)$	$92(\pm4)$	$89(\pm5)$	$91(\pm2)$	$86.4\%(\pm2.9\%)$
Lasso	$99(\pm1)$	$92(\pm4)$	$83(\pm8)$	$89(\pm4)$	$85.9\%(\pm2.2\%)$
ES	$99(\pm1)$	$91(\pm4)$	$89(\pm4)$	$91(\pm3)$	$86.6\%(\pm2.4\%)$
OFRST	$99(\pm1)$	$92(\pm5)$	$99(\pm2)$	$90(\pm3)$	$88\%(\pm1.9\%)$

The results show that the proposed spike based Readout architecture trained with OFRST leads to the highest final accuracy of correctly recognised spoken digits. More-

Table 6: The average number of presynaptic connections selected for each Readout using the LS, RR, lasso, ES methods and the proposed OFRST method, computed for 10 different Liquids.

Training method	Average number of Readout connections									
	1	2	3	4	5	6	7	8	9	10
LS	112	112	112	112	112	112	112	112	112	112
RR	112	112	112	112	112	112	112	112	112	112
Lasso	87	82	85	86	85	86	90	88	85	85
ES	112	112	112	112	112	112	112	112	112	112
OFRST	61	54	60	59	63	67	66	68	65	58

over, each spike based Readout trained with OFRST has significantly fewer connections to Liquid neurons compared to the corresponding standard Readout trained with LS, RR, lasso and ES. Relative to lasso, which results in the fewest presynaptic connections for the standard Readout, the proposed OFRST method leads to a total reduction of 28% in number of connections to the Liquid.

5 Conclusions

This work proposed a spike based Readout architecture for LSMs and introduced a new training method that uses the exact spike timing information generated by SNN models, or recorded during experimental procedures. The new method implements an orthogonal forward regression algorithm for training the Readout parameters, which exploits a distance metric defined in a spike train space.

The new algorithm, called orthogonal forward regression with spike trains (OFRST), allows the selection of the connectivity between the Liquid and the Readout unit, i.e., the neurons in the Liquid that are particularly relevant for solving a given learning or decoding task.

One advantage is that computations are carried out directly on spike trains. The standard methods filter the spike trains and then perform uniform sampling in order to optimise the weights. It is demonstrated theoretically and shown through numerical simulations, with synthetic and experimental data, that the classification accuracy is improved by using exact spike times.

Specifically, new theoretical results demonstrated that the proposed Readout trained with OFRST outperforms the standard Readout, which combines linearly the uniform samples from the neuron filtered outputs and is trained with ordinary least squares, ridge regression, lasso or early stopping. Numerical simulations with synthetic data confirmed the theoretical findings and also showed that the proposed algorithm leads to a much smaller number of Readout synapses. A numerical study showed that OFRST

outperforms the standard methods on decoding the orientation of drifting gratings using the multi-electrode array recordings of the evoked activity in the primary visual cortex of the monkey. An additional example showed the advantage in using the OFRST method on a speech recognition task.

It is interesting to highlight the fact that typically around less than 20% of the total possible connections between Liquid and Readout are required, and that fully connected Readouts achieve less accuracy on classification tasks. This suggests that, besides decoding stimulus features from the evoked brain activity, the new training method could also be used to characterise the functional specificity of neurons in the brain.

Appendix 1. Proofs of theorems

Proof of Theorem 1. The mapping (4) is well defined if the operator $\mathcal{F} : \mathbb{S}^{out} \to \mathbb{FS}^{out}$ is well defined and invertible.

A function $y \in \mathbb{FS}^{out}$ satisfies

$$y(t) = \sum_{k=1}^{N} w_k \mathcal{F} s_k^{out}(t) = \mathcal{F}\left(\sum_{k=1}^{N} w_k s_k^{out}\right)(t).$$

According to the definition of $\mathbb{S}^{out}$ it follows that $\sum_{k=1}^{N} w_k s_k^{out} \in \mathbb{S}^{out}$, and therefore $\mathcal{F} : \mathbb{S}^{out} \to \mathbb{FS}^{out}$ is well defined. Moreover, $\mathcal{F}$ is invertible if it is a one-to-one and onto operator. Let $s_1 = \sum_{k=1}^{N} v_k s_k^{out}$ and $s_2 = \sum_{k=1}^{N} w_k s_k^{out}$. Operator $\mathcal{F}$ is one-to-one if

$$\mathcal{F} s_1 = \mathcal{F} s_2 \Rightarrow s_1 = s_2.$$

It follows that

$$\mathcal{F} s_1 = \mathcal{F} s_2 \Leftrightarrow \sum_{k=1}^{N} v_k \mathcal{F} s_k^{out}(t) = \sum_{k=1}^{N} w_k \mathcal{F} s_k^{out}(t) \Leftrightarrow \sum_{k=1}^{N} (v_k - w_k) \mathcal{F} s_k^{out}(t) = 0.$$

The functions $\{\mathcal{F}s_k^{out}\}_{k=1}^N$ are linearly independent according to Remark 1. It follows that $w_k = v_k, \forall k = 1, \ldots, N$, and thus $\mathcal{F}$ is one-to-one. According to the definition of $\mathbb{FS}^{out}$ and due to the linearity of $\mathcal{F}$, it follows that $\mathcal{F}$ is also an onto operator, and thus it is invertible.

□

Proof of Theorem 2.

$$
\begin{aligned}
\|y^* - \mathcal{R}_{\boldsymbol{w}}\boldsymbol{\mathcal{F}}\boldsymbol{s}^{out}\|_{L^2}^2 &= \|y^* - \mathcal{R}_{\boldsymbol{w}}\left[\mathcal{F}_1 s_1^{out}, \ldots, \mathcal{F}_N s_N^{out}\right]\|_{L^2}^2 \\
&= \|y^* - \sum_{n=1}^{N} w_n \mathcal{F} s_n^{out}\|_{L^2}^2 \\
&= \|y^* - \mathcal{F}\sum_{n=1}^{N} w_n s_n^{out}\|_{L^2}^2 \\
&= \|y^* - \mathcal{F}\bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{\boldsymbol{out}}\|_{L^2}^2 \\
&= \|y^*\|_{L^2}^2 + \|\mathcal{F}\bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{\boldsymbol{out}}\|_{L^2}^2 - 2\left\langle y^*, \mathcal{F}\bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{\boldsymbol{out}}\right\rangle_{L^2} \\
&= \|y^*\|_{L^2}^2 + \|\mathcal{F}\bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{\boldsymbol{out}}\|_{L^2}^2 - 2\left\langle \mathcal{P}_{\mathbb{FS}^{out}}(y^*), \mathcal{F}\bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{\boldsymbol{out}}\right\rangle_{L^2} \\
&= \|y^*\|_{L^2}^2 + \|\mathcal{F}\bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{\boldsymbol{out}}\|_{L^2}^2 - 2\left\langle \mathcal{F}s^{y*}, \mathcal{F}\bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{\boldsymbol{out}}\right\rangle_{L^2} \\
&= \|y^*\|_{L^2}^2 + \frac{1}{2}\|\bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{\boldsymbol{out}}\|_{\mathbb{S}}^2 - \left\langle s^{y*}, \bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{\boldsymbol{out}}\right\rangle_{\mathbb{S}} \\
&= \frac{1}{2}\left[\|\bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{\boldsymbol{out}}\|_{\mathbb{S}}^2 + \|s^{y*}\|_{\mathbb{S}}^2 - 2\left\langle s^{y*}, \bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{\boldsymbol{out}}\right\rangle_{\mathbb{S}}\right] \\
&\qquad - \frac{1}{2}\|s^{y*}\|_{\mathbb{S}}^2 + \|y^*\|_{L^2}^2 \\
&= \frac{1}{2}\|s^{y*} - \bar{\mathcal{R}}_{\boldsymbol{w}}\boldsymbol{s}^{out}\|_{\mathbb{S}}^2 - \frac{1}{2}\|s^{y*}\|_{\mathbb{S}}^2 + \|y^*\|_{L^2}^2.
\end{aligned}
$$

□

Appendix 2. The Standard Orthogonal Forward Regression Algorithm (OFR)

Let $ERR_j^{(p)}$ be the error reduction ratio corresponding to term x_j at iteration p defined as

$$ERR_j^{(p)} = \frac{\left\langle x_j^{\perp(p)}, y^* \right\rangle_{L^2}^2}{\|x_j^{\perp(p)}\|_{L^2}^2 \cdot \|y^*\|_{L^2}^2}.$$

The algorithm for training the Readout weights and predicting the input class is given as follows.

- Initialization
  - $x_j^{\perp(1)} = x_j, j = 1, \ldots, N,$
  - $\ell_1 = \underset{j \in \{1,\ldots,N\}}{\text{argmax}}\, ERR_j^{(1)}, L^{(1)} = \{\ell_1\},$
  - $ERR_1 = ERR_{\ell_1}^{(1)},$
  - $x_1^{\perp} = x_{\ell_1}, w_1^{\perp} = \frac{\langle y^*, x_1^{\perp} \rangle_{L^2}}{\|x_1^{\perp}\|_{L^2}^2},$
  - $w_1 = w_1^{\perp}.$
- For $p = 2, \ldots, N$, compute:
  - $x_j^{\perp(p)} = x_j^{\perp(p-1)} - \frac{\langle x_j, x_{p-1}^{\perp} \rangle_{L^2}}{\|x_{p-1}^{\perp}\|_{L^2}^2}, j \in \{1, \ldots, N\} \backslash L^{(p-1)},$
  - $\ell_p = \underset{j \in \{1,\ldots,N\} \backslash L^{(p-1)}}{\text{argmax}}\, ERR_j^{(p)}, L^{(p)} = L^{(p-1)} \cup \{\ell_p\},$
  - $ERR_p = ERR_{\ell_p}^{(p)},$
  - $x_p^{\perp} = x_{\ell_p}, w_p^{\perp} = \frac{\langle y^*, x_p^{\perp} \rangle_{L^2}}{\|x_p^{\perp}\|_{L^2}^2},$
  - $a_{i,p} = \frac{\langle x_i, x_p^{\perp} \rangle_{L^2}}{\|x_p^{\perp}\|_{L^2}^2}, i \in \{1, \ldots, p-1\},$

- $\boldsymbol{A}^{(p)} = \begin{bmatrix} 1 & a_{1,2} & \dots & a_{1,p} \\ 0 & 1 & \dots & a_{2,p} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{p-1,p} \\ 0 & 0 & \dots & 1 \end{bmatrix}$,
- $\boldsymbol{w}^{\perp(p)} = [w_1^{\perp}, \dots, w_p^{\perp}]$,
- $\boldsymbol{w}^{(p)} = \left[\boldsymbol{A}^{(p)}\right]^{-1} \boldsymbol{w}^{\perp(p)}$,

  where $\boldsymbol{w}^{(p)} = [w_1^{(p)}, \dots, w_p^{(p)}]$ denote the Readout weights at iteration p,
- $\hat{y}^{(p)} = \sum_{k=1}^{p} w_k^{(p)} x_{\ell_p}$,

  where $\hat{y}^{(p)}$ is the Readout output,
- $Pred\left(\hat{y}^{(p)}\right) = \text{sign}\left[\int_0^{T_{max}} \hat{y}^{(p)}(t)dt\right]$,

  where $Pred\left(\hat{y}^{(p)}\right)$ is the class prediction based on the Readout activity on time interval $[0, T_{max}]$, and sign() denotes the sign function.

- Select the smallest p that gives the minimum error for validation.

Appendix 3. The Orthogonal Forward Regression with Spike Trains Algorithm (OFRST)

Let $ERR_j^{(p)}$ be the error reduction ratio corresponding to presynaptic neuron j at iteration p defined as

$$ERR_j^{(p)} = \frac{\left\langle s_j^{\perp(p)}, s^{y*} \right\rangle_{\mathbb{S}}^2}{\|s_j^{\perp(p)}\|_{\mathbb{S}}^2 \cdot \|s^{y*}\|_{\mathbb{S}}^2}.$$

The target output spike train s^{y*} is unknown prior to training. However, for $y^*(t) = \pm 1$, the inner product $\langle s, s^{y*} \rangle_{\mathbb{S}}, \forall s \in \mathbb{S}, s = \{(a_k, t_k)\}_{k=1}^{M}$, can be computed on given

time interval $[T_1, T_2]$, representing the total simulation time, as follows

$$\begin{aligned}\langle s, s^{y*}\rangle_{\mathbb{S}} &= 2\langle \mathcal{F}s, \mathcal{F}s^{y*}\rangle_{L^2} \\ &= 2\langle \mathcal{F}s, y^*\rangle_{L^2} \\ &= (\pm 1)\tau_s \sum_{k=1}^{M} a_k \left[e^{-\frac{\max\{T_1,t_k\}-t_k}{\tau_s}} - e^{-\frac{\max\{T_2,t_k\}-t_k}{\tau_s}}\right].\end{aligned}$$

The algorithm for training the Readout weights and predicting the input class is given as follows.

- Initialization
  - $s_j^{\perp(1)} = s_j^{out}, j = 1, \ldots, N,$
  - $\ell_1 = \underset{j\in\{1,\ldots,N\}}{\text{argmax}}\, ERR_j^{(1)}, L^{(1)} = \{\ell_1\},$
  - $ERR_1 = ERR_{\ell_1}^{(1)},$
  - $s_1^{\perp} = s_{\ell_1}^{out}, w_1^{\perp} = \frac{\langle s^{y*}, s_1^{\perp}\rangle_{\mathbb{S}}}{\|s_1^{\perp}\|_{\mathbb{S}}^2},$
  - $w_1 = w_1^{\perp}.$
- For $p = 2, \ldots, N$, compute:
  - $s_j^{\perp(p)} = s_j^{\perp(p-1)} - \frac{\langle s_j^{out}, s_{p-1}^{\perp}\rangle_{\mathbb{S}}}{\|s_{p-1}^{\perp}\|_{\mathbb{S}}^2}, j \in \{1, \ldots, N\}\backslash L^{(p-1)},$
  - $\ell_p = \underset{j\in\{1,\ldots,N\}\backslash L^{(p-1)}}{\text{argmax}}\, ERR_j^{(p)}, L^{(p)} = L^{(p-1)} \cup \{\ell_p\},$
  - $ERR_p = ERR_{\ell_p}^{(p)},$
  - $s_p^{\perp} = s_{\ell_p}^{out}, w_p^{\perp} = \frac{\langle s^{y*}, s_p^{\perp}\rangle_{\mathbb{S}}}{\|s_p^{\perp}\|_{\mathbb{S}}^2},$
  - $a_{i,p} = \frac{\langle s_i^{out}, s_p^{\perp}\rangle_{\mathbb{S}}}{\|s_p^{\perp}\|_{\mathbb{S}}^2}, i \in \{1, \ldots, p-1\},$

- $\boldsymbol{A}^{(p)} = \begin{bmatrix} 1 & a_{1,2} & \dots & a_{1,p} \\ 0 & 1 & \dots & a_{2,p} \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & a_{p-1,p} \\ 0 & 0 & \dots & 1 \end{bmatrix}$,
- $\boldsymbol{w}^{\perp(p)} = [w_1^{\perp}, \dots, w_p^{\perp}]$,
- $\boldsymbol{w}^{(p)} = \left[\boldsymbol{A}^{(p)}\right]^{-1} \boldsymbol{w}^{\perp(p)}$,

  where $\boldsymbol{w}^{(p)} = [w_1^{(p)}, \dots, w_p^{(p)}]$ denote the Readout weights at iteration p,
- $\hat{s}^{(p)} = \sum_{k=1}^{p} w_k^{(p)} s_{\ell_p}^{out}$,

  where $\hat{s}^{(p)}$ is the Readout output,
- $Pred\left(\hat{s}^{(p)}\right) = \text{sign}\left[T_{max} - 2\tau_s \sum_{k=1}^{M_p} a_k^{(p)} \left(e^{-\frac{T_{max}-t_k^{(p)}}{\tau_s}} - 1\right)\right]$,

  where $Pred\left(\hat{s}^{(p)}\right)$ is the class prediction based on the Readout activity on time interval $[0, T_{max}]$, sign() denotes the sign function, and $\hat{s}^{(p)} = \left\{\left(a_k^{(p)}, t_k^{(p)}\right)\right\}_{k=1}^{M_p}$.

- Select the smallest p that gives the minimum error for validation.

Acknowledgments

DF and DC gratefully acknowledge that this work was supported by BBSRC under grant BB/M025527/1. We also gratefully acknowledge reviewers' comments, which helped improve the quality of the manuscript.

References

Bishop, C. M. (2006). *Pattern recognition and machine learning.* Springer.

Billings, S. A., Chen, S., & Korenberg, M. J. (1989). Identification of MIMO non-linear systems using a forward-regression orthogonal estimator. *International journal of control*, *46* (6), 2157–2189.

Bohte, S., Kok J. & Poutre H. L. (2002). Errorbackpropagation in temporally encoded networks of spiking neurons. *Neurocomp*, *48* (1-4), 17–37.

Carnell, A., & Richardson, D. (2005). Linear algebra for time series of spikes. In *Proc. European Symp. on Artificial Neural Networks*.

Chen, S., Billings, S. A., & Luo, W. (1989). Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control*, *50* (5), 1873 – 1896.

Doddington, G. R. & Schalk, T. B. (1981). Speech recognition: turning theory to practice. *IEEE Spectrum*, 26 – 32.

Dolinský, J., Hirose, K., & Konishi, S. (2017). Readouts for echo-state networks built using locally-regularized orthogonal forward regression. *Journal of Applied Statistics*, *45* (4), 740–762.

Fabre-Thorpe, M., Richard, G., & Thorpe, S. J. (1998). Rapid categorization of natural images by rhesus monkeys. *Neuroreport*, *9* (2), 303–308.

Florescu, D. (2017). *Reconstruction, identification and implementation methods for spiking neural circuits.* Springer.

Florescu, D., & Coca, D. (2018). Identification of Linear and Nonlinear Sensory Processing Circuits from Spiking Neuron Data. *Neural Computation*, *30* (3), 670–707.

Florescu, D., & Coca, D. (2015). A novel reconstruction framework for time-encoded signals with integrate-and-fire neurons. *Neural Computation*, *27* (9), 1872–1898.

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

Florian, V. (2012). The Chronotron : A Neuron That Learns to Fire Temporally Precise Spike Patterns. *PloS one*, *7* (8).

Gardner, B., & Grüning, A. (2016). Supervised learning in spiking neural networks for precise temporal encoding. *PloS one*, *11* (8).

Gollisch, T., & Meister, M. (2008). Rapid neural coding in the retina with relative spike latencies. *Science*, *319* (5866), 1108–1111.

Gütig, R. (2014). To spike, or when to spike? *Current opinion in neurobiology*, *25*, 134–139.

Häusler, S., & Maass, W. (2007). A statistical analysis of information-processing properties of lamina-specific cortical microcircuit models. *Cerebral cortex*, *17* (1), 149–162.

Häusler, S., Markram, H., & Maass, W. (2002). Perspectives of the high dimensional

dynamics of neural microcircuits from the point of view of low dimensional readouts. *Neural Comput*, *14* (11), 2531–2560.

Hirata, Y., Katori, Y., Shimokawa, H., Suzuki, H., Blenkinsop, T. A., Lang, E. J., & Aihara, K. (2008). Testing a neural coding hypothesis using surrogate data. *Journal of neuroscience methods*, *172* (2), 312–322.

Houweling, A. R., & Brecht, M. (2008). Behavioural report of single neuron stimulation in somatosensory cortex. *Nature*, *451*65–68.

Huber, D., Petreanu, L., Ghitani, N., Ranade, S., Hromádka, T., Mainen, Z., & Svoboda, K. (2008). Sparse optical microstimulation in barrel cortex drives learned behaviour in freely moving mice. *Nature*, *451*, 61–64.

Hung, C. P., Kreiman, G., Poggio, T., & DiCarlo, J. J. (2005). Fast readout of object identity from macaque inferior temporal cortex. *Science*, *451* (5749), 863–866.

Izhikevich, E. M. (2006). Polychronization: Computation with Spikes. *Neural Comput*, *18* (2).

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral cortex*, *17* (10), 2443–2452.

Jaeger, H.(2001). The echo state approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology, 2001.

Jones, L. M., Depireux, D. A., Simons, D. J., & Keller, A. (2004). Robust temporal coding in the trigeminal system. *Science*, *304* (5679), 1986–1989.

Kayser, C., Montemurro, M. A., Logothetis, N. K., & Panzeri, S. (2009). Spike-phase coding boosts and stabilizes information carried by spatial and temporal spike patterns. *Neuron*, *61* (4), 597–608.

Kelly, R. C., Smith, M. A., Kass, R. E., & Lee, T. S. (2010). Local field potentials indicate network state and account for neuronal response variability. *Journal of computational neuroscience*, *29* (3), 567–579.

Kohn, A. & Smith, M. A. (2016). Utah array extracellular recordings of spontaneous and visually evoked activity from anasthetized macaque primary visual cortex (V1). *CRCNS.org*, Retrieved from: http://dx.doi.org/10.6080/KONC5Z4X.

Lukosevicius, M., & Jaeger, H. (2009). Reservoir Computing Approaches to Recurrent Neural Network Training. *Computer Science Review*, *3* (3), 127–149.

Lazar, A. A., & Pnevmatikakis, E. A. (2008). Faithful Representation of Stimuli with a Population of Integrate-and-Fire Neurons. *Neural Computation*, *20* (11), 2715–2744.

Lazar, A. A., & Slutskiy, Y. B. (2015). Spiking neural circuits with dendritic stimulus processors. *Journal of Computational Neuroscience*, *38* (1): 1–24, 2015.

Lyon, R. (1982). A computational model of filtering, detection, and compression in the cochlea. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on, ICASSP'82*, *7*.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Comput*, *14* (11), 2531–2560.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics.*, *5* (4), 115–133.

Mainen, Z. F., & Sejnowski, T. J. (1995). Reliability of spike timing in neocortical neurons. *Science*, *268* (5216), 1503–1506.

Markram, H., Wang, Y., & Tsodyks, M.(1998). Differential signaling via the same axon of neucortical pyramidal neurons. In *Proc. Natl. Acad. Sci.*, 95, 5323–5328.

Memmesheimer, R. M., Rubin, R., Ölveczky, B. P., & Sompolinsky, H. (2014). Learning precisely timed spikes. In *Neuron*, 82(4), 925–938.

Natschläger, T., Markram, H., & Maass, W. (2003). Computer models and analysis tools for neural microcircuits. In *Neuroscience Databases*. Springer, Boston, MA.

Nicola, W., & Clopath, C. (2017). Supervised learning in spiking neural networks with FORCE training. In *Nature communications*, 8(1).

Pfister, J. P., Toyoizumi, T., Barber, D., & Gerstner, W. (2006). Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning. *Neural Comput*, *18* (6), 1318–1348.

Ponulak, F., & Kasinski, A. (2010). Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting. *Neural Comput*, *22* (2), 467–510.

Riehle, A., Grün, S., Diesmann, M., & Aertsen, A. (1997). Spike synchronization and rate modulation differentially involved in motor cortical function. *Science*, *278* (5345), 1950–1953.

Seth, A. K. (2015). Neural coding: rate and time codes work together. *Current Biology*, *25* (3), 357–363.

Schalk, T. B.(1982). The Design and Use of Speech Recognition Data Bases. *Proceedings of the Workshop on Standardization for Speech I/O Technology*, *25* (3), 211 – 214.

Schrauwen, B., & Van Campenhout, J. (2003). BSA, a fast and accurate spike train encoding scheme. *In Proceedings of the international joint conference on neural networks*, *4* (4), Piscataway, NJ: IEEE.

Smith, M. A.,z & Kohn, A. (2008). Spatial and temporal scales of neuronal correlation in primary visual cortex. *Journal of Neuroscience*, *28* (48), 12591–12603.

Srivastava, K. H., Holmes, C. M., Vellema, M., Pack, A. R., Elemans, C. P., Nemenman, I., & Sober, S. J. (2017). Motor control by precisely timed spike patterns. In *Proceedings of the National Academy of Sciences*, *114*(5), 1171-1176.

Thomson, A. M., West, D. C., Wang, Y., & Bannister, A. P. (2002). Synaptic connections and small circuits involving excitatory and inhibitory neurons in layers 2-5 of adult rat and cat neocortex: triple intracellular recordings and biocytin labelling in vitro. *Cerebral cortex*, *12* (9), 936–953.

Verstraeten, D., Schrauwen, B., D'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural networks*, *20* (3), 391-403.

Verstraeten, D., Schrauwen, B., Stroobandt, D., & Van Campenhout, J. (2005). Isolated

word recognition with the liquid state machine: a case study. *Information Processing Letters*, *95* (6), 521-528.

Vigneswaran, G., Philipp, R., Lemon, R. N., & Kraskov, A. (2013). M1 corticospinal mirror neurons and their role in movement suppression during action observation. *Curr Biol*, *23* (3).

Wolfe, J., Houweling, A. R., & Brecht, M. (2010). Sparse and powerful cortical spikes. *Current opinion in neurobiology.*, *20* (3), 306–312.

Xu, Y., Zeng, X., Han, L., & Yang, J. (2013). A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks. *Neural Networks*, *43*, 99–113.

Yin, J., Meng, Y., & Jin, Y. (2012). A developmental approach to structural self-organization in reservoir computing. *IEEE transactions on autonomous mental development*, *4* (4).