# A stochastic optimization technique for hyperparameter tuning in reservoir computing

Nickson Mwamsojo, Frederic Lehmann, Kamel Merghem, Yann Frignac, Badr-Eddine Benkelfat

# A stochastic optimization technique for hyperparameter tuning in reservoir computing

Nickson Mwamsojo[a], Frederic Lehmann[*,a], Kamel Merghem[a], Yann Frignac[b], Badr-Eddine Benkelfat[a]

[a] *SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, 19 Place Marguerite Perey, 91120 Palaiseau, France.*
[b] *Huawei Technologies France, 18 Quai du Point du Jour, 92100 Boulogne-Billancourt, France.*

**Abstract**

This paper presents a new approach to reservoir computing (RC) optimization. Reservoir computing (RC) is a framework for building recurrent neural networks (RNNs) that alleviates the well-known learning difficulties in such neural networks by training only the output layer. While the weights of the input and the nonlinear hidden layers are randomly generated, the adjustment of critical hyperparameters, such as the input and the feedback scaling factor, is essential for optimal performance in RC. While recent hardware implementations of RC are crucial for high-speed processing, standard gradient-based hyperparameter optimization is often irrelevant due to potentially uncertain or time-varying internal functions and parameters. In this work, we propose and analyze a stochastic optimization approach using gradient approximations based solely on noisy measurements of the loss function to circumvent this problem. Our numerical and experimental results confirm that the proposed method can provide near-optimal RC hyperparameters with substantial complexity reduction compared to competing methods, validating its potential for RC optimization.

*Key words:* Reservoir computing (RC), hyperparameters, stochastic optimization, gradient approximation, RC hardware implementation.

## 1. Introduction

### *1.1. Hyperparameter tuning for hardware Reservoir Computers*

Recurrent neural networks (RNNs) are artificial neural networks with a feedback-loop useful for classifying and predicting temporal series [1]. However, training all RNN parameters is notoriously a difficult task [2]. A discrete-time nonlinear dynamical framework to circumvent this issue, known as reservoir computing (RC), has been introduced in [3] and [4]. Referring to Fig. 1, the multi-dimensional internal (or

*Corresponding author - Phone: (+33) 1 60 76 46 33. Fax: (+33) 1 60 76 44 33
*Email addresses:* nickson_mwamsojo@telecom-sudparis.eu (Nickson Mwamsojo), frederic.lehmann@telecom-sudparis.eu (Frederic Lehmann), kamel.merghem@telecom-sudparis.eu (Kamel Merghem), yann.frignac@huawei.com (Yann Frignac), badr-eddine.benkelfat@telecom-sudparis.eu (Badr-Eddine Benkelfat)

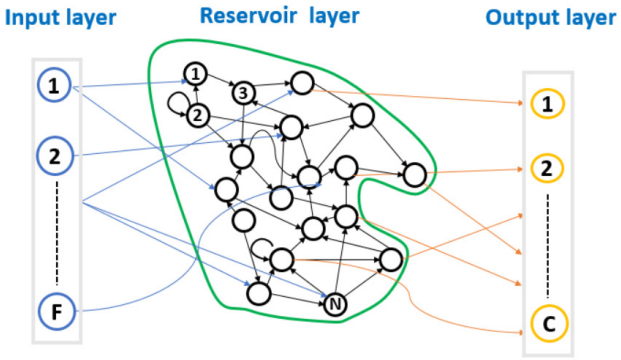



Figure 1: Basic architecture of a reservoir computer showing the three layers.

hidden) state, known as the reservoir, is a nonlinear function of the current input layer vector and the value of the reservoir at the previous time instant. A linear output layer is connected to the reservoir and delivers the classification or prediction outputs. Unlike classical RNNs, the connections to the input layer nodes and the interconnections between reservoir nodes are random but fixed weights. Consequently, only the output layer needs to be trained using standard learning algorithms [5]. Note that the validity of the RC approach has been analyzed in [6] and it has also been used successfully in practical applications such as speech and handwritten recognition [7], robot control, financial forecasting [8], nonlinear equalization [9] and electroencephalography [10]. While practical guidelines exist for selecting many RC hyperparameters (such as the number of nodes in the reservoir, the fixed random weights, etc.) [11], other hyperparameters, such as the input and feedback scaling factors, have a critical effect on the properties of the RC's nonlinear dynamical behavior and thus need to be fine-tuned.

Considering the increasing need for artificial intelligence at high speed and low cost, we focus on hardware implementations having the potential to process data at up to giga-samples per second [12]. We refer the reader to [13] for a recent review of photonic, memristive, spintronic, and mechanical architectures, among others. However, such devices' real-time processing capability comes with several issues nonexistent in digital implementations. The built-in characteristics of such physical devices are not always perfectly known and can even drift with time. Hyperparameter estimation methods addressing the specific constraints related to hardware RCs are thus needed.

### *1.2. Related work*

We survey existing hyperparameter estimation types and discuss their applicability to hardware RCs.

First, let us consider deterministic hyperparameter optimization methods. The simplest method, applicable to both digital and hardware RC, uses brute force parameter selection with a grid-search [14],

thus requiring fine-grained resolution along with an exponential complexity growth in the parameter's dimension. Other methods, including the well-known backpropagation through time [15], are instances of gradient-descent [16, 17]. The later performing a guided search in the parameter space, they have typically much lower complexity than exhaustive search. But in hardware RC the partial or even lack of knowledge regarding the reservoir's feedback weights and nonlinearity prevents their direct application.

Secondly, stochastic optimization [18] has been proposed for digital RC and could also be applied to hardware RC. These techniques include probabilistic methods such as simulated annealing (SAN) [19], evolutionary methods such as genetic algorithms [20] or more recently sequential design strategies such as Bayesian optimization (BO) [21] requiring some form of random search to maximize a query function. These methods, nevertheless, are usually very time-consuming because they need to repeat similar steps a large number of times.

Alternatively, a third class of methods in the form of iterative stochastic algorithms [18], tries to strike a balance between the fast convergence of deterministic gradient-based methods and stochastic approximations. These methods are suitable for optimizing a $p$-dimensional parameter vector, since they need exclusively noisy loss function evaluations. Building on the Robbins-Monro algorithm for root finding [22], procedures for finding the optimum of a loss function based on stochastic finite-difference gradient approximations have been proposed. The original Kiefer-Wolfowitz algorithm [23], which needs $2p$ noisy function evaluations per iteration, has been later simplified to the random direction method [24, p. 58-59] using only a pair of function evaluations per iteration. At each iteration, the latter method creates a two-sided gradient approximation by applying a random perturbation vector that is sampled uniformly and independently at random from a $p-$hypersphere in the parameter space. Almost sure convergence of such procedures usually requires strong assumptions on the perturbations, on the decreasing adaptive step-sizes, and on the decreasing finite-difference steps [25].

### *1.3. Contributions*

We study an iterative stochastic gradient approximation algorithm able to tune an hyperparameter vector, based exclusively on noisy loss function evaluations. This feature is useful in contexts where the exact functional relationship between the parameters and loss function values is unavailable such as in hardware implementations of RC. Moreover, we simplify the random direction method in several ways in order to extend the applicability by

- generating perturbations as random standard unit direction vectors in the hyperparameter space,
- keeping the finite-difference step constant, and
- keeping the adaptation step-size constant.

The first modification is motivated by low implementation complexity wrt the aforementioned original random direction method. The second modification is an adaptation to hardware RC, where increased numerical robustness wrt loss function computations that are noise-ridden is needed. The third modification allows to track parameter drifts (i.e. to account for non stationary loss functions) due to slow hardware time variations. We derive a local convergence analysis based on quadratic approximations of the loss function. Although these convergence results are weaker than for the original method, our results show they hold true in practice. In particular, successful hyperparameter optimization will be demonstrated for simulated and experimental RC.

The notations used throughout this paper for vectors, matrices, probability and statistics are defined in Tab. 1.

The paper is organized as follows. First, Sec. 2 defines the RC system model. Then, the proposed stochastic gradient approximation optimization method is introduced in Sec. 3. Then, Sec. 4 tackles the performance analysis problem by providing a convergence criterion and the expression of the residual error covariance. Finally, in Sec. 5, the effectiveness of the proposed hyperparameter optimization method is investigated on realistic applications, both for a simulated RC and in an experimental optoelectronic hardware implementation.

## 2. System model

Reservoir computers nonlinearly expand input data to higher dimensional state variables that are subsequently recombined to produce the desired output. To achieve this, the time-dependent input layer is allowed to drive the internal (or hidden) layer, while the output layer accomplishes the targeted classification or prediction task (see Fig. 1). Assuming that the input layer receives the feature vector $\mathbf{u}(n) \in \mathbb{R}^F$ at discrete time instant $n$, the nonlinear expansion with memory generates the hidden state in $\mathbb{R}^N$ as [3]

$$\mathbf{x}(n) = f_{NL}(\alpha \mathbf{W}^{in}\mathbf{u}(n) + \beta \mathbf{W}\mathbf{x}(n-1)), \tag{1}$$

Table 1: Notations

| | |
|---|---|
| Vectors and matrices: | |
| $s$ | scalar |
| $\mathbf{v}$ | column vector with components $\mathbf{v}(l)$ |
| $\text{diag}\,(\mathbf{v})$ | diagonal matrix whose principal diagonal is $\mathbf{v}$ |
| $\mathbf{v}(m:n)$ | sequence of vectors $\{\mathbf{v}(t)\}_{t=m}^{n}$, stacked columnwise in a matrix |
| $\mathbf{M}$ | matrix with components $\mathbf{M}(l,c)$ |
| $\text{diag}(\mathbf{M})$ | diagonal matrix, with diagonal elements the principal diagonal of $\mathbf{M}$, while off-diagonal components are zero |
| $\mathbf{I}_m$ | $m \times m$ identity matrix |
| Matrix operators: | |
| $(.)^T$ | transposition |
| $\otimes$ | Kronecker product |
| $\circ$ | Hadamard (elementwise) product |
| vec(.) | vectorization |
| $\rho(.)$ | spectral radius |
| $\Lambda(.)$ | set of eigenvalues |
| $\|\|.\|\|_F$ | Frobenius norm |
| Probability: | |
| $P(.)$ | probability of event |
| $E[.]$ | expectation |

where $f_{NL}$ denotes the nonlinearity applied componentwise. $\mathbf{W}^{in} \in \mathbb{R}^{N\times F}$ and $\mathbf{W} \in \mathbb{R}^{N\times N}$ are sparse random (but fixed) matrices, respectively called the input mask and the network connectivity matrices. The hyperparameters $\alpha$ and $\beta$ standing for the input and feedback scaling factors significantly impact the system's dynamics and must therefore be fine-tuned [3] [1]. Let us collect all hyperparameters in the vector $\boldsymbol{\theta}$. A natural choice for $\boldsymbol{\theta}$ is $[\alpha, \beta]^T$, but more hyperparameters could also be included if needed. The linear output layer subsequently generates the $n$-th read-out as

$$\hat{\mathbf{y}}(n) = \mathbf{W}^{out}\mathbf{x}(n). \tag{2}$$

When RC performs a prediction (resp. a classification) task, $\hat{\mathbf{y}}(n)$ is typically in $\mathbb{R}$ (resp. in $\mathbb{R}^C$, where $C$ stands for the number of classes).

Learning a RC model based on a training dataset $\{\mathbf{u}_{train}(n), \mathbf{y}_{train}(n)\}_{n=1}^{T}$ consists in minimizing the

[1] Note that $\beta$ is a weighting factor for the previous internal state, not to be confused with the notion of output (or readout) feedback in [3] which is not used here.

loss function [3]

$$L(\boldsymbol{\theta}, \mathbf{W}^{out}) = ||\mathbf{y}_{train}(1:T) - \mathbf{W}^{out}\mathbf{x}(1:T)||_F^2 + \lambda||\mathbf{W}^{out}||_F^2, \tag{3}$$

where Tikhonov regularization with parameter $\lambda$ is used to limit the effect of noise and overfitting [5]. Note that in a classification context, ouput layer training typically relies on the fact that if $l \in \{0, 1, \ldots, C-1\}$ is the ground-truth class corresponding to the $n$-th training feature vector $\mathbf{u}_{train}(n)$, the corresponding target RC readout $\mathbf{y}_{train}(n) \in \mathbb{R}^C$ will be $+1$ in its $l$-th coordinate and 0 elsewhere. This learning problem consists in solving:

$$\frac{\partial L(\boldsymbol{\theta}, \mathbf{W}^{out})}{\partial \mathbf{W}^{out}} = \mathbf{0} \tag{4}$$

and admits a unique hyperparameter-dependent closed-form solution expressed as

$$\hat{\mathbf{W}}^{out}\left(\boldsymbol{\theta}, \{\mathbf{u}_{train}(n), \mathbf{y}_{train}(n)\}_{n=1}^{T}\right) = \mathbf{y}_{train}(1:T)\mathbf{x}(1:T)^T\left[\mathbf{x}(1:T)\mathbf{x}(1:T)^T + \lambda\mathbf{I}_N\right]^{-1}, \tag{5}$$

provided that the Echo State Property (ESP) is satisfied (i.e. after some washout period, the reservoir dynamics become independent of the initial condition $\mathbf{x}(0)$ [3, 26]).

## 3. The proposed stochastic gradient approximation method

This section introduces the proposed stochastic approximation method to estimate a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^p$, to minimize the loss function. Each time we evaluate an RC hidden layer on a limited training dataset under new hyperparameters, we obtain a noisy value of the loss function. Therefore, a cross-validation scheme, detailed in Sec. 3.1, is introduced to cope with this variability. In the context of hardware RC, where the function to be minimized for tuning hyperparameters can be seen as a black-box accessible only via noisy evaluations, an iterative stochastic approximation approach is one of the few remaining options to solve the problem efficiently. While the random direction method [24] uses only a pair of loss function evaluations per iteration for the sake of finite-difference gradient approximation, almost sure convergence results need specific conditions on the perturbations, the decreasing adaptation step-size, and the finite-difference step [25]. These conditions cannot always be met in practice. In particular, letting the finite-difference step converge to zero with an increasing iteration index is not advisable for the sake of numerical stability. Also, decreasing the adaptation step size is unsuitable for applications where adaptive tracking of a drifting solution is needed. Therefore, in the proposed hyperparameter estimation method in Sec. 3.2, we let both steps remain constant while also updating the coordinates of $\boldsymbol{\theta}$ one at a time to reduce the

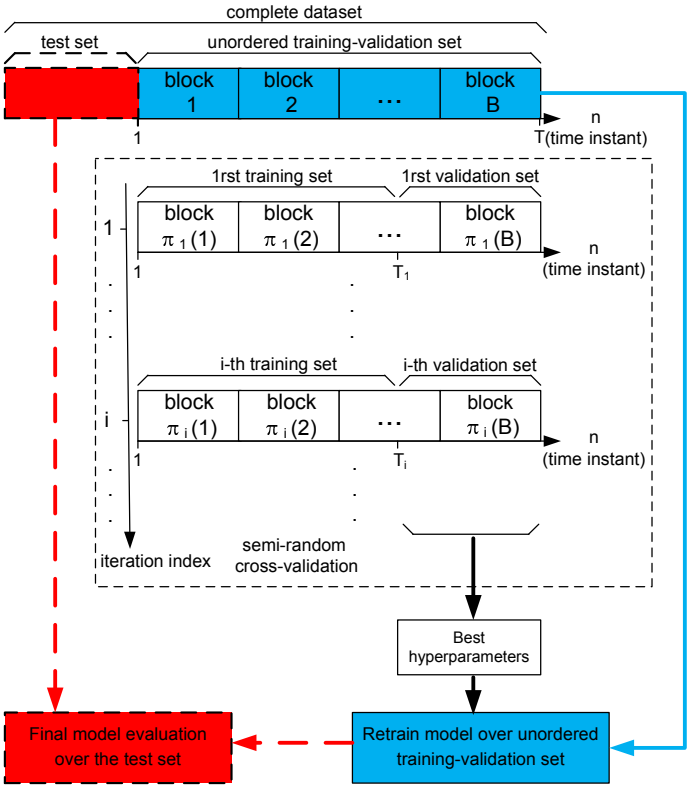



Figure 2: Semi-random cross-validation scheme: training-validation data (blue), test data (red), reshuffled data during hyperparameter estimation (dashed box). $B$: number of homogeneous blocks in the training-validation set. $\pi_i$ : pseudo-random permutation of the data blocks during the $i$-th hyperparameter re-estimation.

complexity by a factor of $p$ during each update. Finally, since stochastic methods are iterative in nature, we also explain how the hidden states and the output weights of the RC are updated at each stage.

### *3.1. Cross-validation scheme*

Since stochastic hyperparameter estimation relies on a succession of loss function evaluations over limited training sets, the induced bias and variance must be mitigated. We use the semi-random cross-validation mechanism illustrated in Fig. 2, similar in spirit to the resampling strategies advocated in [27].

First, the complete dataset is split once and for all into disjoint test and training-validation sets. During hyperparameter initialization, the training-validation set is partitioned into $B$ consecutive blocks of homogeneous time instants. Thus, each block contains a single time instant for a prediction task, and consecutive time instants having the same class label (though its value is assumed *a priori* unknown for validation data) in the context of classification. After hyperparameter update at the $i$-th stochastic approximation iteration, a new split into training and validation data is obtained by applying a pseudo-random permutation $\pi_i$ to the block index set $\{1, 2 \ldots, B\}$. Without loss of generality, the validation data can be selected at each stage among the last blocks.

The final hyperparameter estimates correspond to the minimal loss function on the validation data (this typically occurs for an iteration index close to convergence of stochastic hyperparameter estimation). The RC model is then retrained on the entire unordered training-validation set using the final hyperparameter estimates. Terminal RC model evaluation takes place over the test set. In addition to the loss function in equation (3), for classification tasks we also use the Classification Error Rate (CER) to gauge the model performance:

$$CER(\%) = \frac{\text{Number of wrongly classified entries}}{\text{Cardinality of the test set}} \times 100. \tag{6}$$

*3.2. Proposed RC parameter estimation*

Let $\mathbf{e}_c$ be the standard unit vector in the direction of the $c$-th coordinate (i.e. $\mathbf{e}_c$ is the length-$p$ vector containing 1 in its $c$-th coordinate and 0 elsewhere), for $c = 1, \ldots, p$. Starting from an initial guess $\boldsymbol{\theta}_0$, the proposed RC parameter estimation method alternates between hyperparameter re-estimation (followed by RC hidden state update) and output-layer re-estimation at each iteration.

To be specific, consider the $i$-th iteration where RC hidden states and output layer weights have been trained over the training set of the $(i-1)$-th iteration. We now introduce our hyperparameter re-estimation scheme. Direction vectors $\{\mathbf{d}_i\}$ are uniformly, independently and identically distributed (u.i.i.d.) over $\{\mathbf{e}_1 = [1, 0, \ldots, 0]^T, \mathbf{e}_2 = [0, 1, 0, \ldots, 0]^T, \ldots, \mathbf{e}_p = [0, \ldots, 0, 1]^T\}$, for all iteration indices $i$. Gradient descent, commonly used for optimization in machine learning [1], is not feasible in our setting since gradient calculation needs complete knowledge of the functional relationship between $\boldsymbol{\theta}$ and $L(.,.)$ that was assumed to be unavailable. Instead, a stochastic approximation of the gradient of the loss function in the random direction $\mathbf{d}_i$, is obtained over the $(i-1)$-th training set as

$$\Delta_i = \frac{L(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i, \hat{\mathbf{W}}^{out}_{i-1}) - L(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i, \hat{\mathbf{W}}^{out}_{i-1})}{2h}, \tag{7}$$

where a central finite-difference approach with constant step $h$ has been used. Note that on top of the error inherent to the finite-difference method, the numerator in (7) involves two evaluations of the RC loss function using only a limited number of training data, thus giving rise to noisy evaluations. For the sake of simplicity, the iterative procedure to minimize the loss function updates each component of $\boldsymbol{\theta}$ one at a time according to

$$\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \mu \mathbf{d}_i \Delta_i, \tag{8}$$

where $\mu$ is a constant adaptation step size.

The hidden states are then recomputed over the $i$-th training data, followed by of the output layer weight re-evaluation according to (5).

The proposed parameter optimization procedure is summarized in Algorithm 1.

**Algorithm 1** Parameter optimization procedure

**Require:** $\mu, h, \boldsymbol{\theta}_0$

Initialize the RC hidden states over the initial training set according to Eq. (1)

Initialize $\hat{\mathbf{W}}_0^{out}$ by minimizing $L(\boldsymbol{\theta}_0, \mathbf{W}^{out})$ according to Eq. (5)

**for** $i = 1, 2, \ldots,$ **do**

Pick uniformly at random $c \in \{1, 2, \ldots, p\}$

Set the direction vector $\mathbf{d}_i = \mathbf{e}_c$

Compute $\Delta_i$ according to Eq. (7)

Parameter update: $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{i-1} - \mu \mathbf{d}_i \Delta_i$

Recompute the RC hidden states over the $i$-th training set according to Eq. (1)

Update $\hat{\mathbf{W}}_i^{out}$ by minimizing $L(\boldsymbol{\theta}_i, \mathbf{W}^{out})$ according to Eq. (5)

## 4. Performance analysis

We now analyze the local behavior of the proposed optimization algorithm around the optimal hyperparameter value, $\boldsymbol{\theta}_{opt}$ under standard assumptions. In particular, assuming continuity of the loss function (3) in its two arguments, we study the approximate dynamics of the hyperparameters by fixing the output layer weights to $\hat{\mathbf{W}}^{out}(\boldsymbol{\theta}_{opt})$, i.e. their value taken at the optimal hyperparameters. This simplification is validated via simulations in Sec. 5. The recursion for the proposed method is first approximated around the optimal value of $\boldsymbol{\theta}$ in Sec. 4.1, based on a quadratic approximation of the loss function. By taking into account the stochastic nature of the proposed method, Sec. 4.2 first analyzes the convergence in probability of the transient, provided that the adaptation step size is smaller than some upper bound. Then, Sec. 4.3 derives the steady-state covariance of the parameter vector due to noise terms affecting the stochastic approximation of the gradient.

### *4.1. Second-order analysis around the optimum*

Let $\boldsymbol{\theta}_{opt}$ be the optimal hyperparameter vector and let $\hat{\mathbf{W}}^{out}(\boldsymbol{\theta}_{opt})$ be the corresponding output layer weights. We only consider the variations of the loss function wrt $\boldsymbol{\theta}$ around $\boldsymbol{\theta}_{opt}$, which leads to studying the recursion.

$$\boldsymbol{\theta}_i \approx \boldsymbol{\theta}_{i-1} - \mu \mathbf{d}_i \frac{C(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i) - C(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i)}{2h}, \tag{9}$$

where $C(\boldsymbol{\theta}) = L(\boldsymbol{\theta}, \hat{\mathbf{W}}^{out}(\boldsymbol{\theta}_{opt}))$. Assume the Hessian matrix of $C(\boldsymbol{\theta})$ (i.e. the matrix of continuous second-order partial derivatives), $\mathbf{HC}(\boldsymbol{\theta})$ exists. Performing a second-order expansion of (9) around $\boldsymbol{\theta}_{opt}$ leads to the recursion

$$\boldsymbol{\theta}_i - \boldsymbol{\theta}_{opt} = \left(\mathbf{I}_p - \mu \mathbf{d}_i \mathbf{d}_i^T \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{opt}) - \mu \mathbf{d}_i \mathbf{d}_i^T \mathbf{n}_i, \tag{10}$$

where $\mathbf{n}_i$ is a noise vector accounting for potential noisy loss function evaluations in our method. The derivation is postponed to Appendix A. Since (10) has the structure of a linear time-variant system, it can be seen as the superposition of the stochastic transient response

$$\begin{cases} \boldsymbol{\zeta}_0 = \boldsymbol{\theta}_0 \\ \boldsymbol{\zeta}_i = \boldsymbol{\theta}_{opt} + \left(\mathbf{I}_p - \mu \mathbf{d}_i \mathbf{d}_i^T \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt}), \end{cases} \tag{11}$$

and the noise contribution

$$\begin{cases} \boldsymbol{p}_0 = \mathbf{0}_{p\times 1} \\ \boldsymbol{p}_i = \left(\mathbf{I}_p - \mu \mathbf{d}_i \mathbf{d}_i^T \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)\boldsymbol{p}_{i-1} - \mu \mathbf{d}_i \mathbf{d}_i^T \mathbf{n}_i. \end{cases} \tag{12}$$

*4.2. Transient response behavior*

**Lemma 4.1.** *There exists $\mu^* > 0$, such that for all $0 < \mu < \mu^*$, $\lim_{i\to\infty} \operatorname{vec}\left(E[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T]\right) = \mathbf{0}$.*

**Proof 4.2.** *We first establish the matrix difference equation (see Appendix B)*

$$\operatorname{vec}\left(E[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T]\right) = \mathbf{D}_C(\mu) \operatorname{vec}\left(E[(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T]\right), \tag{13}$$

*whose dynamics are governed by the matrix*

$$\mathbf{D}_C(\mu) = \mathbf{I}_{p^2} - \frac{\mu}{p}\left(\mathbf{HC}(\boldsymbol{\theta}_{opt}) \otimes \mathbf{I}_p + \mathbf{I}_p \otimes \mathbf{HC}(\boldsymbol{\theta}_{opt})\right) + \frac{\mu^2}{p} \operatorname{diag}\left(\operatorname{vec}(\mathbf{I}_p)\right)\left(\mathbf{HC}(\boldsymbol{\theta}_{opt}) \otimes \mathbf{HC}(\boldsymbol{\theta}_{opt})\right). \tag{14}$$

*Using [28, Thm. 4.4.5],*

$$\Lambda\left(\mathbf{HC}(\boldsymbol{\theta}_{opt}) \otimes \mathbf{I}_p + \mathbf{I}_p \otimes \mathbf{HC}(\boldsymbol{\theta}_{opt})\right) = \{\lambda' + \lambda'', \forall(\lambda' > 0, \lambda'' > 0) \in \Lambda(\mathbf{HC}(\boldsymbol{\theta}_{opt}))^2\}, \tag{15}$$

*consequently, $\mathbf{HC}(\boldsymbol{\theta}_{opt}) \otimes \mathbf{I}_p + \mathbf{I}_p \otimes \mathbf{HC}(\boldsymbol{\theta}_{opt})$ is also symmetric positive-definite so that in turn*

$$\Lambda\left(\mathbf{I}_{p^2} - \frac{\mu}{p}\left(\mathbf{HC}(\boldsymbol{\theta}_{opt}) \otimes \mathbf{I}_p + \mathbf{I}_p \otimes \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)\right) = \left\{1 - \frac{\mu}{p}(\lambda' + \lambda''), \forall(\lambda' > 0, \lambda'' > 0) \in \Lambda(\mathbf{HC}(\boldsymbol{\theta}_{opt}))^2\right\}. \tag{16}$$

*Applying the Bauer-Fike theorem [29, p. 321] to (14), there exists $\mu^* > 0$, such that for all $0 < \mu < \mu^*$ $\rho(\mathbf{D}_C(\mu)) < 1$.*

*Finally, $\lim_{i\to\infty} \operatorname{vec}\left(E[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T]\right) = \mathbf{0}$ if and only if $\rho(\mathbf{D}_C(\mu)) < 1$ [30, p. 119]. Noting that $0 < \mu < \mu^*$ is a sufficient condition for $\rho(\mathbf{D}_C(\mu)) < 1$ completes the proof.*

We are now ready to state our main convergence result regarding the transient behavior.

**Theorem 4.3.** *When $0 < \mu < \mu^*$, the transient response random vectors $\boldsymbol{\zeta}_i \xrightarrow[i\to\infty]{} \boldsymbol{\theta}_{opt}$ in probability.*

**Proof 4.4.** *Applying the definition of convergence in probability in the multivariate case [18, p. 530], we must show that*

$$\lim_{i\to\infty} P\left(\sqrt{(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})} \geq \epsilon\right) = 0, \quad \forall \epsilon > 0. \tag{17}$$

*Applying Markov's inequality, for any $\epsilon > 0$, we have*

$$\begin{aligned} P\left(\sqrt{(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})} \geq \epsilon\right) &= P\left((\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt}) \geq \epsilon^2\right) \\ &\leq \frac{E\left[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})\right]}{\epsilon^2} \\ &\leq \frac{\operatorname{trace} E\left[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T\right]}{\epsilon^2}. \end{aligned} \tag{18}$$

*Letting $i \to \infty$ when $0 < \mu < \mu^*$, lemma 4.1 leads to the desired result.*

*4.3. Steady-state covariance*

We assume the $\{\mathbf{n}_i\}$ to be white zero-mean with covariance matrix $\boldsymbol{\Sigma}$ and independent of the direction vectors $\{\mathbf{d}_i\}$.

Regarding the mean of $\boldsymbol{p}_i$ conditional on the previous direction vectors, according to (12) we have

$$E[\boldsymbol{p}_i | \ldots, \mathbf{d}_{i-2}, \mathbf{d}_{i-1}] = \left(\mathbf{I}_p - \mu E[\mathbf{d}_i\mathbf{d}_i^T]\mathbf{H}\mathbf{C}(\boldsymbol{\theta}_{opt})\right) E[\boldsymbol{p}_{i-1} | \ldots, \mathbf{d}_{i-2}, \mathbf{d}_{i-1}] - \mu E[\mathbf{d}_i\mathbf{d}_i^T]E[\mathbf{n}_i], \tag{19}$$

using the fact that the $\{\mathbf{d}_i\}$ are u.i.i.d. Applying the law of total expectation and using the fact that $E[\mathbf{n}_i] = \mathbf{0}$ and $E[\mathbf{d}_i\mathbf{d}_i^T] = (1/p)\mathbf{I}_p$, for all $i \geq 0$, we have

$$E[\boldsymbol{p}_i] = \left(\mathbf{I}_p - \frac{\mu}{p}\mathbf{H}\mathbf{C}(\boldsymbol{\theta}_{opt})\right) E[\boldsymbol{p}_{i-1}]. \tag{20}$$

Thus $E[\boldsymbol{p}_i] = \mathbf{0}$, for all $i \geq 0$, since $E[\boldsymbol{p}_0] = \mathbf{0}$.

It can also be shown that the covariance of $\{\boldsymbol{p}_i\}$ is obtained via the recursion (see Appendix C)

$$\operatorname{vec}\left(E[\boldsymbol{p}_i\boldsymbol{p}_i^T]\right) = \mathbf{D}_C(\mu)\operatorname{vec}\left(E[\boldsymbol{p}_{i-1}\boldsymbol{p}_{i-1}^T]\right) + \frac{\mu^2}{p}\operatorname{vec}\left(\operatorname{diag}\left(\boldsymbol{\Sigma}\right)\right). \tag{21}$$

When $0 < \mu < \mu^*$, then $\rho(\mathbf{D}_C(\mu)) < 1$ and the steady-state solution is given by [31, p. 27].

$$\lim_{i\to\infty} \operatorname{vec}\left(E[\boldsymbol{p}_i\boldsymbol{p}_i^T]\right) = \frac{\mu^2}{p}(\mathbf{I}_{p^2} - \mathbf{D}_C(\mu))^{-1}\operatorname{vec}\left(\operatorname{diag}\left(\boldsymbol{\Sigma}\right)\right). \tag{22}$$

## 5. Numerical and experimental results

First, Sec. 5.1 describes the considered generic setup, which is valid for a number of optoelectronic and optical hardware implementations of RC. Then in Sec. 5.2, we introduce three datasets corresponding to challenging real-world classification tasks (with a significant number of features and/or classes). In Sec. 5.3, we introduce benchmark hyperparameter optimization methods. Grid-search and SAN were selected for their ability to reach the global optimum asymptotically [18]. While Bayesian optimization (see the discussion in [32, p. 163]) lacks such performance guarantees in the general case, it was also retained due to its good empirical performances in RC [21]. Sec. 5.4 provides a complexity analysis of the proposed and benchmark algorithms. Finally, the performances of the proposed hyperparameter optimization algorithm are assessed using both a simulated (see Sec. 5.5) and a hardware implementation (see Sec. 5.6) of RC. Note that in the sequel, the value of the parameter $h$ is found via a manual search to find a reasonable tradeoff between the approximation error due to the finite difference method and noise term affecting the loss function evaluations in Eq. (7).

### *5.1. Setup*

We consider the delayed feedback loop optoelectronic and optical RC implementations from [33, 34, 35, 36], which can be modeled as the discrete-time RC model (1) after digital-to-analog conversion (DAC) and analog-to-digital conversion (ADC) at the input and output of the circuit, respectively. In our numerical simulations, we emulate the model in [33]. Since the operating principle of such physical implementations relies on serializing the internal (hidden) nodes using time-multiplexing, the connectivity matrix $\mathbf{W}$ corresponds to causal filtering having non-zero coefficients only on the lower diagonals. To be more specific, considering the continuous-time impulse response of a first order low-pass filter $h(t) = e^{-t/T_R}$ of response time $T_R \approx 240ns$ [33], it's sampled version with sampling period $T_s = 52.18ns$ has coefficients $h_i = h(iT_s)$, where the integer $i \geq 0$ stands for the discrete time index. The connectivity matrix $\mathbf{W}$ can therefore be written as follows :

$$\mathbf{W} = \begin{bmatrix} h_0 & 0 & 0 & 0 & 0 & . & . & . & 0 \\ h_1 & h_0 & 0 & 0 & 0 & . & . & . & 0 \\ h_2 & h_1 & h_0 & 0 & 0 & . & . & . & 0 \\ h_3 & h_2 & h_1 & h_0 & 0 & . & . & . & 0 \\ h_4 & h_3 & h_2 & h_1 & h_0 & . & . & . & 0 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ & . & . & . & h_4 & h_3 & h_2 & h_1 & h_0 \end{bmatrix}. \tag{23}$$

In hardware implementations, these coefficients' exact values are generally not precisely known due to inaccuracies in physical filter characterization.

The elements of the sparse input matrix $\mathbf{W}^{in}$ at positions $(l, c)$ are usually generated from a zero-mean distribution. In this work, they are sampled independently from the probability distribution defined as:

$$P\left(\mathbf{W}^{in}(l,c) = w\right) = \begin{cases} \frac{1}{5}, & \text{if } w = -1, 1. \\ \frac{3}{5}, & \text{if } w = 0. \end{cases} \tag{24}$$

Moreover, $f_{NL}$ in (1) stands for the squared transfer function of a Mach-Zehnder Modulator (MZM) (where squaring is due to the action of a photodetector), i.e

$$f_{NL}(.) = \sin^2(. + \phi). \tag{25}$$

The value of the parameter $\phi$ controls the nonlinearity in play by varying from highly linear operation regimes (e.g., around point C in Fig. 3) to highly non-linear regimes (e.g., point A or point E in Fig. 3). Hence, by varying the value of $\phi$, the nonlinearity can be optimized. We select $\phi = 3$ rad in our simulations, which is located near an extremum of $f_{NL}$ but slightly leaning towards the negative slope, which gave the best results in [33]. We also have two other global parameters to set, namely the size of the reservoir, $N$, and the regularization parameter $\lambda$. In the sequel we use $N = 400$, for the sake of fair comparison with [33]. Regarding $\lambda$, we fix it at $10^{-8}$ after running trials manually.

Finally, the hyperparameter vector $\boldsymbol{\theta} = [\alpha, \beta]^T$ needs to be fine-tuned since it plays a vital role in controlling the dynamical regime in the reservoir by ensuring the so-called echo state property [3].

### *5.2. Datasets*

#### *5.2.1. Spoken digits recognition*

We use TIDIGITS LDC93S10 [37], a dataset consisting of 326 speakers of different ages and gender, uttering the digits from 0 to 9 so that $C = 10$. Each file is recorded at 20kHz, and the utterances are in

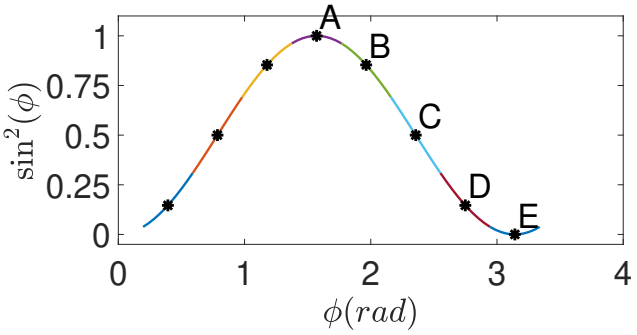



Figure 3: Squared Mach-Zehnder transfer function (Operating point A, E: highly nonlinear regimes. Operating point C: essentially linear regime).

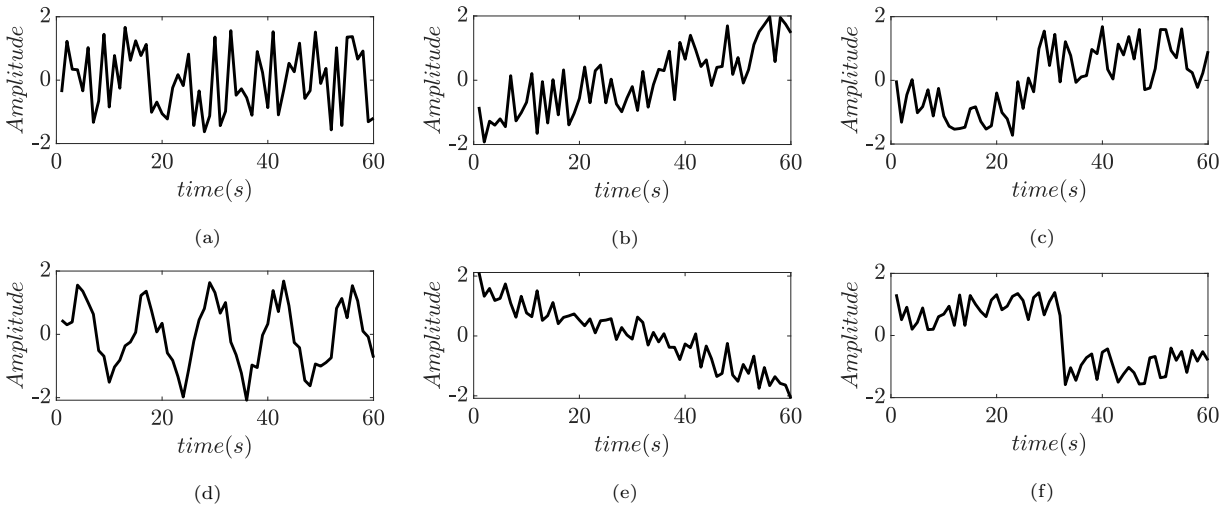



Figure 4: Representative examples of control chart 1-D signals for each class: (a) Normal parttern (b) Increasing trend (c) Upward shift (d) Cyclic pattern (e) Decreasing trend (f) Downward shift.

English. From this dataset, we choose 1500 sound files with equal distribution between gender and age. Raw sound files undergo preprocessing to obtain the Mel-frequency cepstral coefficients (MFCCs) [38] as input features to the RC. The files in the dataset have an average of $10k$ samples, so we trim the silent ends for the longer files and pad the slightly shorter files with zeros. For each sound file, 69 frames of $25.6ms$ are generated every $7.2ms$ (with an overlap of $18.4ms$) and passed to a 512-point FFT after Hanning windowing. Then 128 Mel-spaced filter bank values are generated, followed by the extraction of $F = 13$ cepstral coefficients per frame. It follows that each time-homogeneous block in Fig. 2 is of length 69 (each frame playing the role of a single time instant $n$). Moreover, the parameters of the proposed semi-random cross-validation scheme in Sec. 3.1 are listed in Tab. 2.

*5.2.2. Control chart pattern recognition*

In industrial, financial, and medical settings, control charts are a well-known tool to classify observed time series patterns produced by equipment or sensors to detect anomalies and guarantee continuous operation.

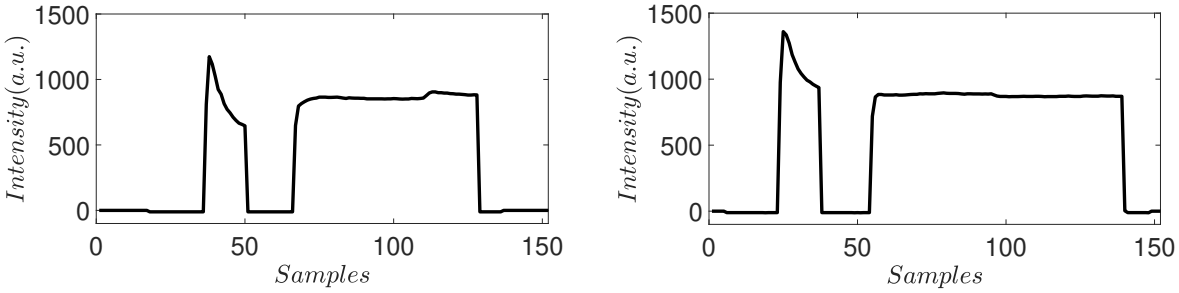
The goal is to establish similarities between the system-generated temporal signals and known normal or abnormal signals to narrow down the necessary mitigative measures. However, determining the similarity between time series is not a trivial task. The dataset [39] introduced in [40] contains 600 synthetically generated control charts consisting of 1-D signals ($F = 1$) with 60 consecutive time instants $n$ having the same class label. It follows that each time-homogeneous block in Fig. 2 is of length 60. Representative examples of the $C = 6$ classes are depicted in Fig 4. Moreover, the parameters of the proposed semi-random cross-validation scheme in Sec. 3.1 are listed in Tab. 2.

*5.2.3. Semiconductor microelectronics fabrication: Wafer classification task*

The process of manufacturing semiconductor microelectronics is prone to many imperfections that may cause malfunctions, unreliability, and performance issues during post-production. In this work, we use the data from a sensor for the plasma emission at 405 *nm*, deemed more efficient for faulty detection [41]. A feature vector consists of $F = 152$ consecutive readings from the sensor having the same class label. Since the feature vector exhausts all the recorded observations, $\beta$ should be forced to zero. Therefore, for this particular case, the RC can be viewed as an Extreme Learning Machine [42]. The task is to discriminate between the normal and abnormal wafers ($C = 2$ classes) referring to non-faulty and faulty wafers, respectively. Unfortunately, the dataset has a significant class imbalance, with the abnormal class accounting for only 10.7% of the train data and 12.1% of the test data, respectively. We employed the SMOTE technique to restrain the overfitting of the majority class, avoid underfitting the minority class, and improve overall generalization [43]. With this tool, we undersample the majority class slightly, and synthetic examples are generated to add more samples for the minority class. Figure 5a and 5b show the readings for normal and abnormal wafers, respectively. Moreover, the parameters of the proposed semi-random cross-validation scheme in Sec. 3.1 are listed in Tab. 2.

| **Task** | **Parameters** | | |
|---|---|---|---|
| | **Training** | **Validation** | **Test** |
| Spoken digit recognition | 1200 (80%) | 150 (10%) | 150 (10%) |
| Control chart pattern recognition | 240 (40%) | 60 (10%) | 300 (50%) |
| Wafer classification | 900 (12.6%) | 100 (1.4%) | 6164 (86%) |

Table 2: Parameters of the proposed semi-random cross-validation: number of blocks (percentage) for training, validation, and test data. Note that the parameter $B$ is obtained by adding up the number of blocks for training and validation.




(a) (b)

Figure 5: Representative examples of sensor readings for wafers : (a) Normal (b) Abnormal

### *5.3. Benchmark methods*

| **Parameter** | **Range** | **Discretization step** |
|---|---|---|
| Input scaling $\alpha$ | $[10^{-2}, 1]$ | Speech & control chart : $6.6 \times 10^{-3}$<br>Wafer : $9.9 \times 10^{-4}$ |
| Feedback scaling $\beta$ | $[0, 1]$ | $10^{-2}$ |

Table 3: Parameters of the grid search for digital RC optimization.

Grid search, simulated annealing (SAN) and Bayesian optimization (BO) are the benchmark methods against which we assess the proposed hyperparameter optimization performance. Grid-search retains the hyperparameter value corresponding to the lowest loss function (3) after an exhaustive search over the discretized variable $\boldsymbol{\theta}$ [14]. Tab. 3 lists the parameters for the grid-search algorithm showing the resolution necessary to find a near-optimal operating point. On the other hand, SAN is a stochastic optimization algorithm that emulates the slow and controlled cooling of materials until the lowest energy state is reached. Our implementation of SAN follows [18, 44, 45, 46], with parameters listed in Tab. 4. Note that to counteract the effect of noisy (instead of deterministic) evaluations of the loss function, we use the acceptance threshold $\tau$ introduced in [18, p. 115], in the sense that we only update when the new loss function improves the previous loss function by not less than $\tau$. Finally, BO iteratively samples the noisy loss function with the hyperparameter value maximizing an acquisition function using a Gaussian process (GP) prior/posterior distribution on functions. In this paper, we use the formalism described in [32] with parameters listed in Tab. 5.

| Parameter | Value |
|---|---|
| Initial temperature ($T_0$) | 1 |
| Cooling scheme | Geometric |
| Cooling rate | - 0.9 for spoken digit and control chart<br>- 0.8 for wafer dataset |
| Epochs | 100 |
| Threshold ($\tau$) | - $1.8 \times 10^{-2}$ for the spoken digits dataset<br>- $5.6 \times 10^{-1}$ for the control chart dataset<br>- $1.3 \times 10^{-1}$ for the wafer dataset |
| Random perturbation of the solution | Lorentzian distribution |

Table 4: Parameters of SAN for digital RC optimization.

| Parameter | Value |
|---|---|
| Gaussian process kernel | Matérn $\nu = \frac{5}{2}$ |
| Observation noise variance | $\sigma^2 = 10^{-5}$ |
| Initial loss function observations | $n_{pre} = 5$ |
| Acquisition function (AF) | Expected improvement (EI) |
| AF Optimization | $(n_{restart}, n_{search}) = (10, 8000)$ |

Table 5: Parameters of Bayesian optimization for the digital RC optimization (see [32] for the notations). At each iteration, optimization of the acquisition function, performs $n_{restart}$ times a random search with $n_{search}$ samples.

### *5.4. Complexity analysis*

Instead of hardware-specific metrics such as execution time, we consider asymptotic complexity as our performance indicator of computational effort, which is commonly used in computer science to compare the complexity of algorithms. Considering two matrices $\mathbf{A}$ and $\mathbf{B}$ having the same dimension $d_1 \times d_2$, the asymptotic complexity of $\mathbf{A} - \mathbf{B}$ and $||\mathbf{A}||_F$ is $\mathcal{O}(d_1 d_2)$ [29]. For a matrix $\mathbf{C}$ with dimension $d_2 \times d_3$, the asymptotic complexity of the matrix multiplication $\mathbf{AC}$ is $\mathcal{O}(d_1 d_2 d_3)$ [29]. For a square matrix $\mathbf{M}$ is dimension $d \times d$, the the asymptotic complexity of the matrix inversion $\mathbf{M}^{-1}$ is $\mathcal{O}(d^3)$ [29].

Consequently, evaluating Eq. (3) has overall computational complexity $aCNT + bCT + cCN$, where $a, b, c$ are constants. Similarly, evaluating Eq. (5) has overall computational complexity $a'CN^2 + b'CNT + c'N^3 + d'TN^2$, where $a', b', c', d'$ are constants. Considering that $N \lesssim T$ in practical applications, the

| | ***Complexity*** |
|---|---|
| **Grid Search** | $\mathcal{O}(S(CNT + N^2T))$ |
| **SAN** | $\mathcal{O}(S(CNT + N^2T))$ |
| **BO** | $\mathcal{O}(S(CNT + N^2T) + S^3)$ |
| ***Proposed*** | $\mathcal{O}(S(CNT + N^2T))$ |

Table 6: Asymptotic complexity order of hyperparameter optimization. Note that the definition of the number of steps, $S$, depends on the optimization algorithm (see Sec. 5.4).

asymptotic complexity order of a single loss function evaluation (resp. of recomputing the output weights per hyperparameter vector and training set) is $\mathcal{O}(CNT)$ (resp. $\mathcal{O}(CNT + N^2T)$).

The complexity of all methods is proportional to the number of repeated steps $S$ (see Tab. 6). First for the grid-search approach, $S$ represents the number of grid points for which an output weight update and a loss function evaluation is needed. Secondly for the SAN approach, $S$ represents the number of temperature changes times the number of epochs until convergence, each requiring an output weight update and a loss function evaluation. Thirdly for the proposed approach, $S$ represents the number of iterations, each requiring two loss function evaluations (in Eq. (7)) and re-evaluation of the output weights. Thirdly, for BO, $S$ represents the number of iterations, each requiring an output weight update and a loss function evaluation in addition to the $\mathcal{O}(S^3)$ complexity order of the method itself (see [32]). Finally ignoring all low-order terms and all constant multipliers, the complexity order of the proposed method and all benchmark algorithms is summarized in Tab. 6. In conclusion, for a given task, $C$, $N$, and $T$ are fixed so that the complexity of all algorithms is linear in $S$. The only exception is that when $S$ becomes large (typically many hundreds), the linear complexity trend of BO is dominated by $\mathcal{O}(S^3)$ instead.

### *5.5. Digital RC optimization*

We first consider the optimization of a digital RC using computer simulations of the system described in Sec. 5.1. For the sake of fair comparison, the initial hyperparameter of the proposed method and SAN is set to $\boldsymbol{\theta}_0 = [0.25, 0.25]^T$ (resp. $\boldsymbol{\theta}_0 = [0.25, 0]^T$) for the speech and control chart (resp. wafer) datasets. the finite-difference step $h$ was set manually to the values given in Tab. 7.

*5.5.1. Transient response of the proposed method*

We first wish to check numerically the validity of the convergence of the transient as given by Thm. 4.3, in our approximate analysis of the dynamics of the proposed hyperparameter optimization method. In order to do so, we first find a good approximation of the cost function via a multivariate interpolation of the grid search as a quadratic function of $\alpha$ and $\beta$ of the form of

$$C(\boldsymbol{\theta}) = a_0 + a_1\alpha + a_2\beta + a_3\alpha\beta + a_4\beta^2 + a_5\alpha^2. \tag{26}$$

A good approximation of $\boldsymbol{\theta}_{opt}$ is obtained by minimizing Eq. (26). The following optima approximations for each task were obtained:

- Spoken digit recognition : $\boldsymbol{\theta}_{opt} = [\alpha, \beta]^T = [0.0521, 0.131]^T$,
- Control chart recognition : $\boldsymbol{\theta}_{opt} = [\alpha, \beta]^T = [0.18, 0.139]^T$.
- Wafer classification : $\boldsymbol{\theta}_{opt} = [\alpha, 0]^T = [0.88, 0]^T$.

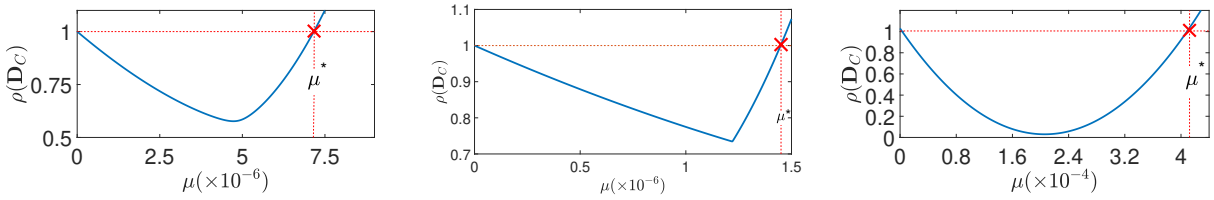
Then we approximate the second-order partial derivatives of $C(\boldsymbol{\theta})$ around $\boldsymbol{\theta}_{opt}$ to obtain elements of the Hessian matrix $\mathbf{HC}(\boldsymbol{\theta}_{opt})$ as follows :

$$\mathbf{HC}(\boldsymbol{\theta}_{opt}) = \left[\begin{array}{cc} \frac{\partial C(\boldsymbol{\theta})}{\partial \alpha^2} & \frac{\partial C(\boldsymbol{\theta})}{\partial \alpha \partial \beta} \\ \frac{\partial C(\boldsymbol{\theta})}{\partial \beta \partial \alpha} & \frac{\partial C(\boldsymbol{\theta})}{\partial \beta^2} \end{array}\right]\Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_{opt}}. \tag{27}$$

Since the Hessian matrix of interest is supposed to be exempt of noise, accurate evaluation is obtained by averaging $20 - 30$ second-order difference approximations of each component. The spectral radius of the matrix (14) is then plotted for all datasets as a function of the adaptation step-size $\mu$ in Fig 6. As predicted by the proof of lemma 4.1, there exists $\mu^*$ such that the spectral radius of (14) is strictly lower than 1 for $0 < \mu < \mu^*$. Our numerical simulations showed that the proposed method's transient indeed converges for $\mu \lesssim \mu^*$ as inferred from Thm. 4.3 and diverges otherwise. Note that the values selected for $\mu$ in Tab. 7 meet this requirement.

| **Parameter** | **Spoken Digit** | **Control Chart** | **Wafer** |
|---|---|---|---|
| Adaptation step-size ($\mu$) | $5 \times 10^{-6}$ | $1 \times 10^{-6}$ | $1 \times 10^{-4}$ |
| Finite-difference step ($h$) | $5 \times 10^{-3}$ | $5 \times 10^{-2}$ | $5 \times 10^{-2}$ |

Table 7: Parameters of the proposed method for digital RC simulation.




(a) (b) (c)

Figure 6: Spectral radius of $\mathbf{D}_C(\mu)$ for the digital RC: (a) spoken digit (b) control chart and (c) wafer dataset.

*5.5.2. Performance comparison with benchmark methods*

Sample trajectories of optimization vs. the number of steps $S$ are shown in Fig. 7 for the spoken digits recognition task. We place the convergence plots for SAN, BO and our method side by side for an easier visual comparison of the evolution for $\alpha$, $\beta$, and the loss function given by Eq. (3). Note that the seemingly erratic behavior of SAN before convergence comes from the non-zero probability of acceptance of hyperparameter values that increase the loss function, especially at high temperatures. In practice, a stopping criterion (for instance, detecting marginal loss function changes) would be implemented to detect the convergence of SAN, BO and the proposed method.

The performance measures of all methods after RC optimization for the spoken digits recognition task are shown in Tab. 8. Note that the proposed method, SAN and BO have similar CERs after convergence. We could also improve the grid-search CER after optimization, but at the expense of higher complexity, by using a finer grid of values. Moreover, assuming that the computational complexity is essentially proportional to the number of steps $S$ defined previously (see the discussion in Sec. 5.4); according to Tab. 8 the proposed algorithm outperforms the three competing methods at least by a factor of 2.4 in this respect. We also compare the variance of the values taken by $\alpha$ and $\beta$ at convergence, as shown in Tab. 11. We observe that the parameters' empirical and theoretical variances (derived from Eq. (22)) have the same order of magnitude at convergence. To emulate this numerically, $\mathbf{\Sigma}$, the covariance of the noise affecting the gradient approximation in all directions around $\boldsymbol{\theta}_{opt}$, is reliably estimated by computing the sample covariance of the vector

$$\begin{bmatrix} \frac{C(\boldsymbol{\theta}_{opt}+h\mathbf{e}_1)-C(\boldsymbol{\theta}_{opt}-h\mathbf{e}_1)}{2h} \\ \vdots \\ \frac{C(\boldsymbol{\theta}_{opt}+h\mathbf{e}_p)-C(\boldsymbol{\theta}_{opt}-h\mathbf{e}_p)}{2h} \end{bmatrix}. \tag{28}$$

using Monte Carlo simulations with 1000 trials [47].

A similar analysis is applied for the other tasks under consideration. The sample trajectories are shown in Fig. 8 and 9. The performance measures of all optimization methods are given in Tab. 9 and 10 for control

| | ***Grid Search*** | ***SAN*** | ***BO*** | ***Proposed*** |
|---|---|---|---|---|
| $\alpha$ | 0.062 | 0.053 | 0.049 | 0.055 |
| $\beta$ | 0.13 | 0.13 | 0.13 | 0.13 |
| **CER** (%) | 2.1 | 1.88 | 1.77 | 1.94 |
| **Number of steps** ($S$) | 15000 | 3400 | 600 | 250 |

Table 8: Digital RC on the spoken digits recognition task: hyperparameter estimates, CER, and number of steps $S$ needed to complete optimization.

chart recognition and wafer classification, respectively. We notice a similar tendency to that of the spoken digit task. Finer resolution may improve the results of grid search at the price of increasing the computational complexity. For control chart (resp. wafer) classification, to attain convergence of down to $CER \approx 2\%$ (resp. $\approx 0.4\%$), SAN requires 3550 and BO requires 1400, (resp. SAN requires 700 and BO requires 1100) steps, whereas our method requires 1300 (resp. 340). This corresponds to a moderate complexity reduction (resp. a reduction by at least a factor of 2.1) for the control chart (resp. wafer) classification. Interestingly, we observe in Fig. 9e that a lower loss function does not always convert to a lower $CER$. Our interpretation is that the optimum obtained by minimizing the chosen loss function in Eq. (3) need not be the same as the one obtained by minimizing some other cost function such as the $CER$. Nevertheless, it can be checked on all used datasets that a low value of the loss function is consistent with a low $CER$. For all tasks, the

| | ***Grid Search*** | ***SAN*** | ***BO*** | ***Proposed*** |
|---|---|---|---|---|
| $\alpha$ | 0.185 | 0.182 | 0.179 | 0.174 |
| $\beta$ | 0.14 | 0.14 | 0.14 | 0.14 |
| **CER** (%) | 3.33 | 1.97 | 2.1 | 2.1 |
| **Number of steps** ($S$) | 15000 | 3550 | 1400 | 1300 |

Table 9: Digital RC on the control charts recognition task: hyperparameter estimates, CER, and number of steps $S$ needed to complete optimization.

variance of $\alpha$ and $\beta$ at convergence are shown in Tab. 11. We observe an agreement between the empirical and the theoretical variances to the same order of magnitude at convergence.

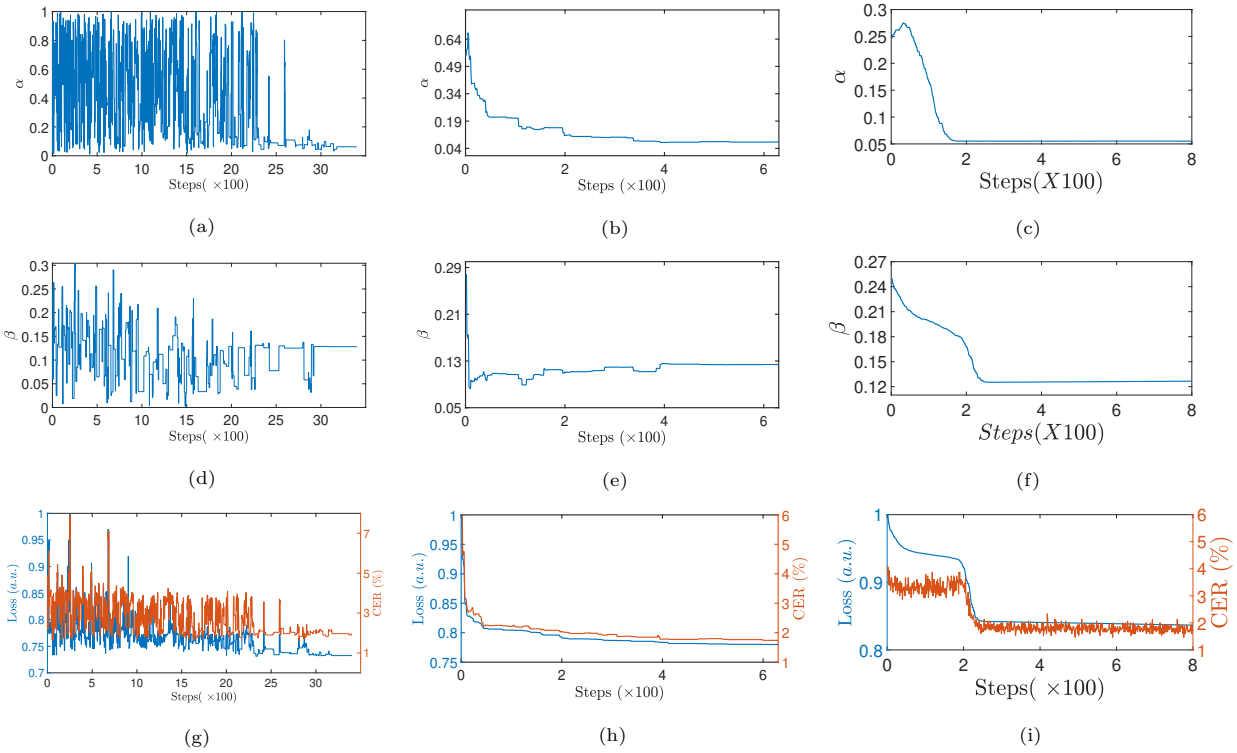



Figure 7: Sample optimization trajectory for digital RC on the spoken digits recognition task. Left column: SAN - Middle column: BO - Right column: proposed method.

| | ***Grid Search*** | ***SAN*** | ***BO*** | **Proposed** |
|---|---|---|---|---|
| $\alpha$ | 0.86 | 0.88 | 0.87 | 0.88 |
| **CER** (%) | 0.6 | 0.4 | 0.35 | 0.41 |
| **Number of steps** ($S$) | 1000 | 700 | 1100 | 340 |

Table 10: Digital RC on the wafer classification task: hyperparameter estimates, CER, and number of steps $S$ needed to complete optimization..

### *5.6. Hardware RC optimization*

The RC system described in Sec. 5.1 now uses a hardware implementation that also needs hyperparameter optimization. To this end, we work with the optoelectronic compact setup provided by Laurent Larger, one of the authors of [33] with FEMTO-ST in Besançon, France. The setup and its components are illustrated in Fig. 10. A distributed feedback (DFB) laser diode (with input power $20mW$) emits light at wavelength $1550nm$. This light enters a Mach-Zehnder Modulator (MZM) with a sinusoidal transfer function modulated by a signal proportional to the continuous-time input signal $u_I(t)$ obtained from the discrete-time samples $\mathbf{W}^{in}\mathbf{u}(n)$ via sample-and-hold DAC. After modulation, the light propagates in the 4.2 km fiber spool (equivalent to a delay line with delay $\tau_D \approx 20.87\mu s$). At the fiber output, a photodiode

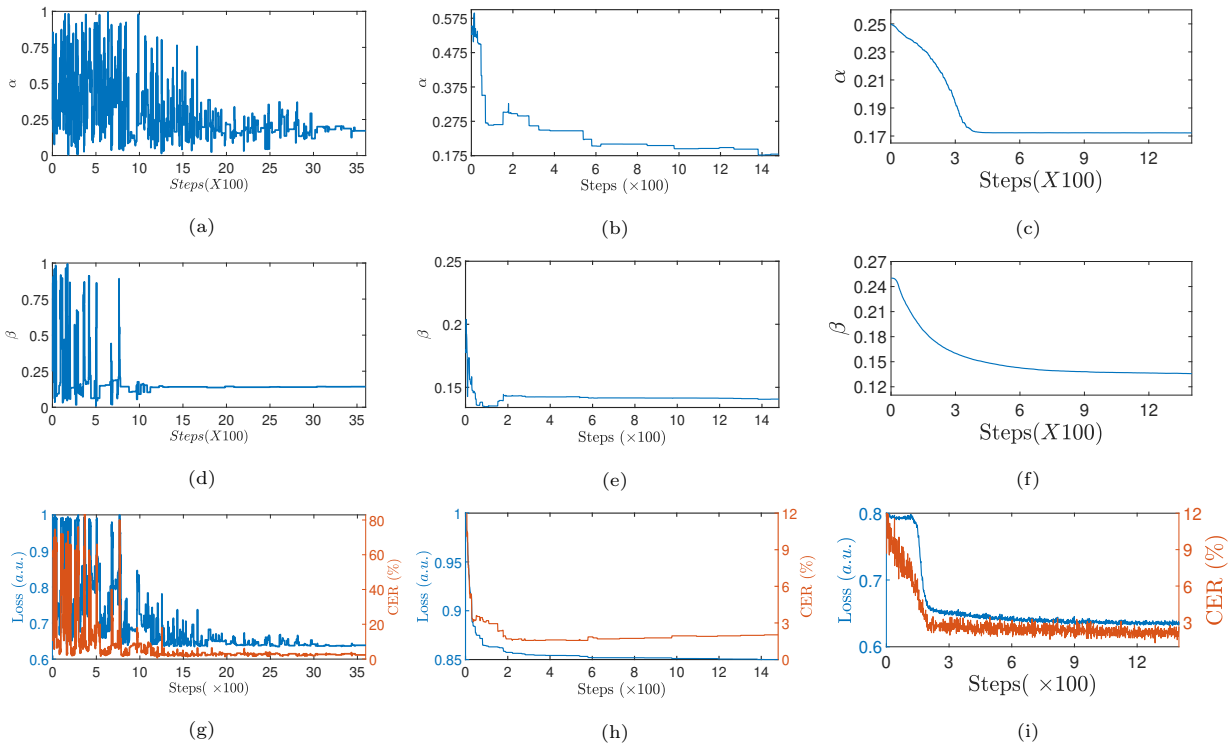



Figure 8: Sample optimization trajectory for digital RC on the control charts recognition task. Left column: SAN - Middle column: BO - Right column: proposed method.

| | **Spoken Digit** | | **Control Chart** | | **Wafer** |
|---|---|---|---|---|---|
| | var($\alpha$) | var($\beta$) | var($\alpha$) | var($\beta$) | var($\alpha$) |
| Theoretical | $7.94 \times 10^{-9}$ | $14.9 \times 10^{-9}$ | $5.8 \times 10^{-8}$ | $1.8 \times 10^{-8}$ | $4.3 \times 10^{-7}$ |
| Empirical | $8.47 \times 10^{-9}$ | $16.4 \times 10^{-9}$ | $6.1 \times 10^{-8}$ | $2.1 \times 10^{-8}$ | $6.55 \times 10^{-7}$ |

Table 11: Theoretical and empirical variances of $\alpha$ and $\beta$ around the optimal hyperparameter vector for the proposed method for digital RC.

converts the optical intensity variations to electric variations, followed by an electronic feedback circuit acting like a lowpass filter with characteristic response time $T_R$, that enables the addition of the feedback photo detected signal $x(t - \tau_D)$ to the input $u_I(t)$. In this way, $N = 400$ virtual hidden nodes are created via time-division-multiplexing, where the delay-spacing between consecutive nodes is $T_s \approx 52.18ns$ [33]. Selecting $T_R = 4.6T_s$ allows for local coupling between nodes emulating the sparse connections in $\mathbf{W}$. The equation that governs this nonlinear delay system is modeled as

$$T_R \frac{dx(t)}{dt} + x(t) = \kappa \sin^2(\alpha u_I(t - \tau_D) + \beta x(t - \tau_D) + \phi), \tag{29}$$

where $\phi$ is the MZM bias and $\kappa$ is the nonlinearity gain. Finally, sampling the electronic feedback circuit output with a ADC $N$ times at the sampling period $T_s$ reconstitutes the hidden state vector $\mathbf{x}(n)$. The

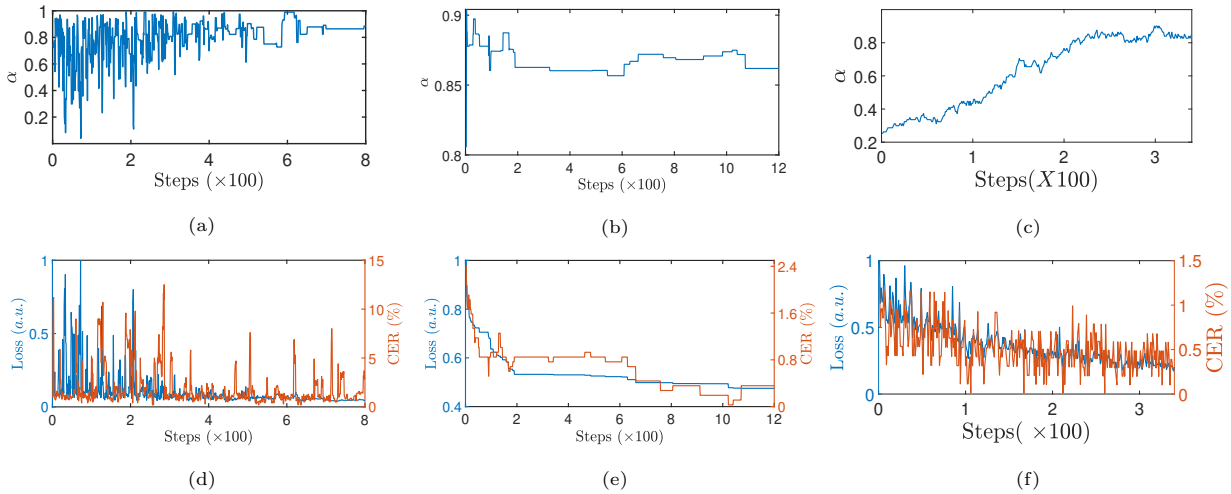



Figure 9: Sample optimization trajectory for digital RC on the wafer classification task. Left column: SAN - Middle column: BO - Right column: proposed method.

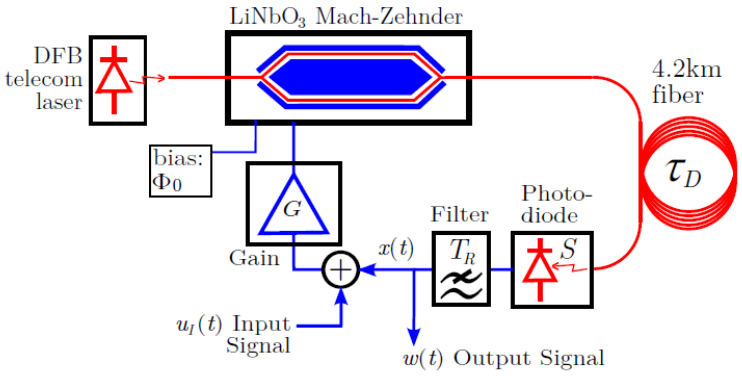



Figure 10: Setup 1 from FEMTO-ST (Reprinted with permission from [33] © The Optical Society).

linear output layer can then be implemented via digital postprocessing.

However, many of the setup parameters are not precisely known, for example, the lowpass filter coefficients, the feedback scaling factor $\beta$, and many other details of our fixed hardware configuration. It follows that the only remaining hyperparameter that can be digitally controlled is the input scaling factor $\alpha$, understood as the range of the signal $u_I(t)$, generated with an Arbitrary Waveform Generator (AWG) [48]. The range scaling variable $\alpha$ is initialized at 1.5 Volts (the maximum output range for the AWG). Tunable global parameters crucial to RC performance are $\phi$ (set as in our previous simulations) and the nonlinearity gain $\kappa$ (set by adjusting the input power of the laser). We manually set $\phi$ and $\kappa$ close to the optimal values obtained in [33]. Note that coarse adjustment of $\kappa$ is achieved using a potentiometer. In contrast, manual adjustment of the MZM bias $\phi$ via a voltage source is subject to drift in time due to changing temperature conditions. This creates further uncertainty regarding the actual value of the nonlinear function $f_{NL}(.)$.

Thus in the context of hardware RC, the need for the automatic tuning algorithm advocated in the

| Parameter | Spoken Digit | Control Chart | Wafer |
|---|---|---|---|
| Adaptation step-size ($\mu$) | $1.5 \times 10^{-4}$ | $1.8 \times 10^{-4}$ | $1 \times 10^{-4}$ |
| Finite-difference step ($h$) | $2 \times 10^{-2}$ | $2 \times 10^{-2}$ | $5 \times 10^{-2}$ |

Table 12: Parameters of the proposed method for hardware RC.

present paper is justified *a posteriori* by the unknown and slowly varying functional relationship between the adjustable hyperparameter $\alpha$ and the loss function. To be specific, an external computer running our algorithm updates $\alpha_i$ at iteration $i$. It communicates the corresponding value to the AWG through an interface to serve as the new amplitude range of the MZM input voltage. In this way, the proposed algorithm updates $\alpha$ without human intervention.

The parameters of the proposed algorithm for the hardware implementation are given in Tab. 12. Again, the adaptation step size being smaller than the upper bound $\mu^*$ deduced from the plots in Fig. 11, the transient response will die out for each task.

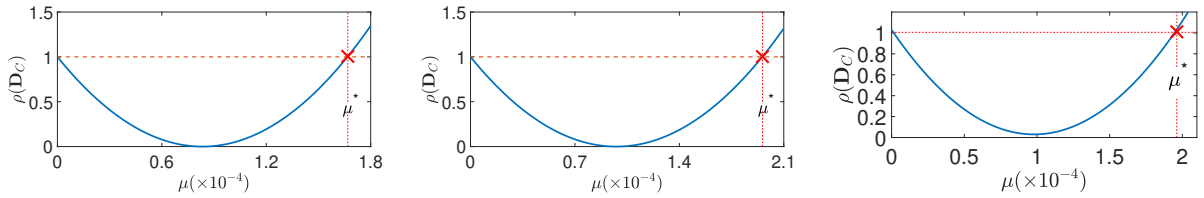



Figure 11: Spectral radius of $\mathbf{D}_C(\mu)$ for the hardware RC: (a) spoken digits recognition (b) control chart recognition and (c) wafer classification.

The results of this experiment are plotted in Fig. 12, showing sample trajectories of $\alpha$ and that of the loss function (and CER) as they converge to the optimal values for all the studied tasks. The result of these experiments for each task in Table 13 shows fast convergence towards a low CER value within a few tens of iterations. The resultant empirical and theoretical variances for $\alpha$ at convergence in Table 14 have again the same order of magnitude for each task.

## 6. Conclusion

In this paper, an algorithm for automatic tuning of the hyperparameters of RC is introduced. The proposed method is a stochastic optimization version of gradient-descent on a loss function, where gradients are approximated with finite differences so that the functional relationship between the desired hyperparameters and the value of the loss function need not be known. Unlike similar methods, the procedure can track the

| **Task** | $\alpha$ | **CER** (%) | **Number of steps** ($S$) |
|---|---|---|---|
| Spoken Digit | 0.32 | 2.6 | 30 |
| Control Chart | 0.18 | 4 | 50 |
| Wafer | 1.36 | 0.3 | 60 |

Table 13: Proposed method for hardware RC: hyperparameter estimate for $\alpha$, CER, and number of steps $S$ needed to complete optimization.

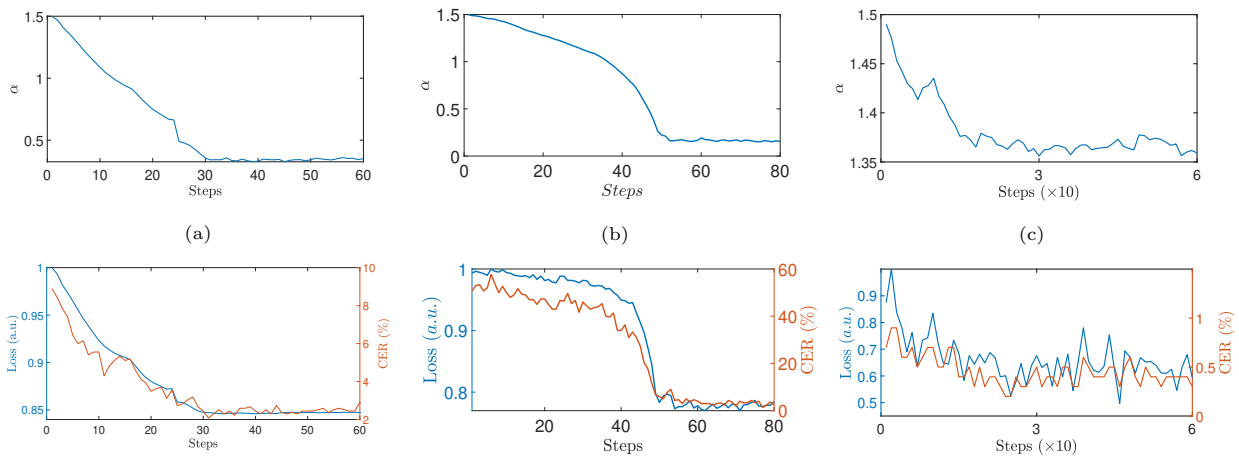



Figure 12: Sample optimization trajectory using the proposed method for hardware RC: Left column: spoken digits dataset, Middle column: control charts dataset - Right column: wafer dataset.

| | var($\alpha$) | | |
|---|---|---|---|
| **Type** | **Spoke Digit** | **Control Chart** | **Wafer** |
| Theoretical | $5.87 \times 10^{-6}$ | $3.77 \times 10^{-7}$ | $1.53 \times 10^{-4}$ |
| Empirical | $5.22 \times 10^{-6}$ | $3.32 \times 10^{-7}$ | $2.73 \times 10^{-4}$ |

Table 14: Theoretical and empirical variances of $\alpha$ around the optimal hyperparameter value for the proposed method for hardware RC.

time fluctuations of the hyperparameters and is numerically robust since it is governed by a constant adaptation step size and a constant finite-difference step. Under suitable hypotheses, the convergence analysis shows that the transient behavior converges to the optimum value. Also, a theoretical analysis of the effect of noise affecting the loss function - due to potential outliers or variations in the training data - provides the variance of the hyperparameters at convergence.

All aforementioned features constitute a valuable advantage in practical RC hardware implementations. Indeed, successful application on an optoelectronic implementation demonstrated the validity of the proposed approach on real-life classification tasks.

Future work will consider extensions to other RC devices and generalizations to various loss functions and machine-learning architectures.

## A. Second-order expansion of the proposed method in (9)

Assuming $C(\boldsymbol{\theta})$ is locally strictly convex close to its optimum, a second order Taylor expansion around $\boldsymbol{\theta}_{opt}$ is obtained as

$$C(\boldsymbol{\theta}) \approx C(\boldsymbol{\theta}_{opt}) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}_{opt})^T \mathbf{HC}(\boldsymbol{\theta}_{opt})(\boldsymbol{\theta} - \boldsymbol{\theta}_{opt}) + \nu(\boldsymbol{\theta}), \tag{A.1}$$

where the Hessian matrix $\mathbf{HC}(\boldsymbol{\theta}_{opt})$ is symmetric positive definite [49], since all second-order partial derivatives were assumed to be continuous. The term $\nu(\boldsymbol{\theta})$ is a noise term accounting for potential noisy loss function evaluations in our method. Replacing $\boldsymbol{\theta}$ alternatively by $\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i$ and $\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i$, we get

$$\begin{aligned} C(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i) &\approx C(\boldsymbol{\theta}_{opt}) + \nu(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i) + \frac{1}{2}(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{opt} + h\mathbf{d}_i)^T \mathbf{HC}(\boldsymbol{\theta}_{opt})(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{opt} + h\mathbf{d}_i), \\ C(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i) &\approx C(\boldsymbol{\theta}_{opt}) + \nu(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i) + \frac{1}{2}(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{opt} - h\mathbf{d}_i)^T \mathbf{HC}(\boldsymbol{\theta}_{opt})(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{opt} - h\mathbf{d}_i). \end{aligned} \tag{A.2}$$

It follows that

$$\frac{C(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i) - C(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i)}{2h} \approx \mathbf{d}_i^T \mathbf{HC}(\boldsymbol{\theta}_{opt})(\boldsymbol{\theta}_{i-1} - \boldsymbol{\theta}_{opt}) + \frac{\nu(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i) - \nu(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i)}{2h}. \tag{A.3}$$

Let us define the random noise vector that would be obtained if a gradient approximation were sought along each direction $\mathbf{e}_c$, $c \in \{1, \ldots, p\}$ (instead of a single randomly chosen $\mathbf{d}_i$ as in the proposed algorithm)

$$\mathbf{n}_i = \begin{bmatrix} \frac{\nu(\boldsymbol{\theta}_{i-1} + h\mathbf{e}_1) - \nu(\boldsymbol{\theta}_{i-1} - h\mathbf{e}_1)}{2h} \\ \vdots \\ \frac{\nu(\boldsymbol{\theta}_{i-1} + h\mathbf{e}_p) - \nu(\boldsymbol{\theta}_{i-1} - h\mathbf{e}_p)}{2h} \end{bmatrix}. \tag{A.4}$$

It follows that the noise term affecting (A.3) can be rewritten as

$$\frac{\nu(\boldsymbol{\theta}_{i-1} + h\mathbf{d}_i) - \nu(\boldsymbol{\theta}_{i-1} - h\mathbf{d}_i)}{2h} = \mathbf{d}_i^T \mathbf{n}_i. \tag{A.5}$$

Now, injecting (A.3) into (9) results in (10).

## B. Proof of the matrix difference equation (13)

Starting from (11) and recalling that the direction vectors $\{\mathbf{d}_i\}$ are u.i.i.d.

$$\begin{aligned} & E\left[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T | \boldsymbol{\zeta}_{i-1}\right] \\ &= \frac{1}{p}\sum_{c=1}^{p} \left(\mathbf{I}_p - \mu \mathbf{e}_c \mathbf{e}_c^T \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T \left(\mathbf{I}_p - \mu \mathbf{e}_c \mathbf{e}_c^T \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)^T \\ &= \left(\mathbf{I}_p - \frac{\mu}{p}\mathbf{HC}(\boldsymbol{\theta}_{opt})\right)(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T \\ &\quad - \frac{\mu}{p}(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T \mathbf{HC}(\boldsymbol{\theta}_{opt})^T \\ &\quad + \frac{\mu^2}{p}\sum_{c=1}^{p} \mathbf{e}_c \mathbf{e}_c^T \mathbf{HC}(\boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T \mathbf{HC}(\boldsymbol{\theta}_{opt})^T \mathbf{e}_c \mathbf{e}_c^T, \end{aligned} \tag{B.1}$$

where we have used the fact that $\sum_{c=1}^{p} \mathbf{e}_c \mathbf{e}_c^T = \mathbf{I}_p$ in the last equality. Applying the law of total expectation,

$$\begin{aligned}
&E\left[(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_i - \boldsymbol{\theta}_{opt})^T\right] \\
&= \left(\mathbf{I}_p - \frac{\mu}{p}\mathbf{HC}(\boldsymbol{\theta}_{opt})\right) E\left[(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T\right] \\
&- \frac{\mu}{p} E\left[(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T\right] \mathbf{HC}(\boldsymbol{\theta}_{opt})^T \\
&+ \frac{\mu^2}{p}\mathbf{I}_p \circ \left(\mathbf{HC}(\boldsymbol{\theta}_{opt}) E\left[(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})(\boldsymbol{\zeta}_{i-1} - \boldsymbol{\theta}_{opt})^T\right] \mathbf{HC}(\boldsymbol{\theta}_{opt})^T\right).
\end{aligned} \tag{B.2}$$

A straightforward application of the rules of vectorization in [30, p. 97-98] leads to the desired result in (13).

## C. Proof of recursion (21)

Starting from (12) and recalling that the direction vectors $\{\mathbf{d}_i\}$ are u.i.i.d. and independent from the zero-mean white noise process $\{\mathbf{n}_i\}$

$$\begin{aligned}
&E\left[\boldsymbol{p}_i \boldsymbol{p}_i^T | \boldsymbol{p}_{i-1}\right] \\
&= \frac{1}{p}\sum_{c=1}^{p} \left(\mathbf{I}_p - \mu \mathbf{e}_c \mathbf{e}_c^T \mathbf{HC}(\boldsymbol{\theta}_{opt})\right) \boldsymbol{p}_{i-1}\boldsymbol{p}_{i-1}^T \left(\mathbf{I}_p - \mu \mathbf{e}_c \mathbf{e}_c^T \mathbf{HC}(\boldsymbol{\theta}_{opt})\right)^T \\
&\quad + \frac{\mu^2}{p}\sum_{c=1}^{p} \mathbf{e}_c \mathbf{e}_c^T E[\mathbf{n}_i \mathbf{n}_i^T] \mathbf{e}_c \mathbf{e}_c^T \\
&= \left(\mathbf{I}_p - \frac{\mu}{p}\mathbf{HC}(\boldsymbol{\theta}_{opt})\right) \boldsymbol{p}_{i-1}\boldsymbol{p}_{i-1}^T \\
&\quad - \frac{\mu}{p}\boldsymbol{p}_{i-1}\boldsymbol{p}_{i-1}^T \mathbf{HC}(\boldsymbol{\theta}_{opt})^T \\
&\quad + \frac{\mu^2}{p}\sum_{c=1}^{p} \mathbf{e}_c \mathbf{e}_c^T \mathbf{HC}(\boldsymbol{\theta}_{opt}) \boldsymbol{p}_{i-1}\boldsymbol{p}_{i-1}^T \mathbf{HC}(\boldsymbol{\theta}_{opt})^T \mathbf{e}_c \mathbf{e}_c^T, \\
&\quad + \frac{\mu^2}{p}\sum_{c=1}^{p} \mathbf{e}_c \mathbf{e}_c^T E[\mathbf{n}_i \mathbf{n}_i^T] \mathbf{e}_c \mathbf{e}_c^T,
\end{aligned} \tag{C.1}$$

where we have used the fact that $\sum_{c=1}^{p} \mathbf{e}_c \mathbf{e}_c^T = \mathbf{I}_p$, in the last equality. Applying the law of total expectation,

$$\begin{aligned}
&E\left[\boldsymbol{p}_i \boldsymbol{p}_i^T\right] \\
&= \left(\mathbf{I}_p - \frac{\mu}{p}\mathbf{HC}(\boldsymbol{\theta}_{opt})\right) E\left[\boldsymbol{p}_{i-1}\boldsymbol{p}_{i-1}^T\right] \\
&\quad - \frac{\mu}{p} E\left[\boldsymbol{p}_{i-1}\boldsymbol{p}_{i-1}^T\right] \mathbf{HC}(\boldsymbol{\theta}_{opt})^T \\
&\quad + \frac{\mu^2}{p}\mathbf{I}_p \circ \left(\mathbf{HC}(\boldsymbol{\theta}_{opt}) E\left[\boldsymbol{p}_{i-1}\boldsymbol{p}_{i-1}^T\right] \mathbf{HC}(\boldsymbol{\theta}_{opt})^T\right) \\
&\quad + \frac{\mu^2}{p}\,\mathrm{diag}(\boldsymbol{\Sigma}).
\end{aligned} \tag{C.2}$$

A straightforward application of the rules of vectorization in [30, p. 97-98] leads to the desired result in (21).

## References


[1] S. Haykin, Neural Networks: A Comprehensive Foundation, Prentice Hall, 1999.

[2] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Transactions on Neural Networks 5 (2) (1994) 157–166.

[3] H. Jaeger, The "echo state" approach to analysing and training recurrent neural networks, GMD-Report 148, German National Research Institute for Computer Science (2001) 1–47.

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

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

[6] L. Grigoryeva, J.-P. Ortega, Echo state networks are universal, Neural Networks 108 (2018) 495–508.

[7] A. Jalalvand, K. Demuynck, W. De Neve, J.-P. Martens, On the application of reservoir computing networks for noisy image recognition, Neurocomputing 277 (2018) 237–248.

[8] H. J. Mantas Lukoševičius, B. Schrauwen, Reservoir computing trends, Künstliche Intelligenz 26 (4) (2012) 365–371.

[9] F. Da Ros, S. M. Ranzini, H. Bülow, D. Zibar, Reservoir-computing based equalization with optical pre-processing for short-reach optical transmission, IEEE Journal of Selected Topics in Quantum Electronics 26 (5) (2020) 1–12.

[10] L. Bozhkov, P. Koprinkova-Hristova, P. Georgieva, Reservoir computing for emotion valence discrimination from eeg signals, Neurocomputing 231 (2017) 28–40, neural Systems in Distributed Computing and Artificial Intelligence.

[11] M. Lukoševičius, A practical guide to applying echo state networks, in: G. Montavon, G. B. Orr, K.-R. Müller (Eds.), Neural Networks: Tricks of the Trade: Second Edition, Springer, Berlin, Heidelberg, 2012, pp. 659–686.

[12] J. Bueno, D. Brunner, M. C. Soriano, I. Fischer, Photonic information processing at 20gs/s rates based on semiconductor lasers with delayed optical feedback, in: 2017 European Conference on Lasers and Electro-Optics and European Quantum Electronics Conference, Optical Society of America, Munich, Germany, 2017, pp. 1–1.

[13] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, A. Hirose, Recent advances in physical reservoir computing: A review, Neural Networks 115 (2019) 100–123.

[14] N. Trouvain, L. Pedrelli, T. T. Dinh, X. Hinaut, ReservoirPy: an Efficient and User-Friendly Library to Design Echo State Networks, in: ICANN 2020 - 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, 2020, pp. 1–19.

[15] M. Hermans, B. Schrauwen, One step backpropagation through time for learning input mapping in reservoir computing applied to speech recognition, in: Proceedings of 2010 IEEE International Symposium on Circuits and Systems, 2010, pp. 521–524.

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

[17] L. A. Thiede, U. Parlitz, Gradient based hyperparameter optimization in echo state networks, Neural Networks 115 (2019) 23–29.

[18] J. C. Spall, Introduction to Stochastic Search and Optimization: estimation, simulation, and control, 1st Edition, John Wiley & Sons, Inc., USA, 2003.

[19] A. T. Sergio, T. B. Ludermir, Reservoir computing optimization with a hybrid method, in: 2014 International Joint Conference on Neural Networks (IJCNN), 2014, pp. 2653–2660.

[20] S. Zhong, X. Xie, L. Lin, F. Wang, Genetic algorithm optimized double-reservoir echo state network for multi-regime time series prediction, Neurocomputing 238 (2017) 191–204.

[21] J. Yperman, T. Becker, Bayesian optimization of hyper-parameters in reservoir computing (2016). doi:10.48550/ARXIV.1611.05193.

[22] H. Robbins, S. Monro, A stochastic approximation method, The Annals of Mathematical Statistics 22 (3) (1951) 400–407.

[23] J. Kiefer, J. Wolfowitz, Stochastic estimation of the maximum of a regression function, The Annals of Mathematical Statistics 23 (3) (1952) 462–466.

[24] H. J. Kushner, D. S. Clark, Stochastic Approximation for Constrained and Unconstrained Systems, Applied Mathematical Sciences 26, Springer, New York, 1978.

[25] D. Chin, Comparative study of stochastic algorithms for system optimization based on gradient approximations, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 27 (2) (1997) 244–249.

[26] I. B. Yildiz, H. Jaeger, S. J. Kiebel, Re-visiting the echo state property, Neural Networks 35 (2012) 1–9.

[27] R. Simon, Resampling Strategies for Model Assessment and Selection, Springer US, Boston, MA, 2007, pp. 173–186.

[28] R. A. Horn, C. R. Johnson, Topics in Matrix Analysis, Cambridge University Press, Cambridge; New York, 1994.

[29] G. H. Golub, C. F. Van Loan, Matrix Computations, 3rd Edition, The Johns Hopkins University Press, Baltimore and London, 1996.

[30] H. Lütkepohl, Handbook of matrices, Wiley, Berlin [u.a.], 1996.

[31] H. Lütkepohl, New introduction to multiple time series analysis, Springer, New York, 2005.

[32] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, N. de Freitas, Taking the human out of the loop: A review of bayesian optimization, Proceedings of the IEEE 104 (1) (2016) 148–175.

[33] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutierrez, L. Pesquera, C. R. Mirasso, I. Fischer, Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing, Opt. Express 20 (3) (2012) 3241–3249.

[34] L. Larger, A. Baylón-Fuentes, R. Martinenghi, V. S. Udaltsov, Y. K. Chembo, M. Jacquot, High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification, Phys. Rev. X 7 (2017) 011015.

[35] D. François, S. Anteo, A. Akram, H. M., M. Serge, Fully analogue photonic reservoir computer, Scientific Reports 6 (2016) 22381.

[36] F. Duport, A. Smerieri, A. Akrout, M. Haelterman, S. Massar, Virtualization of a photonic reservoir computer, Journal of Lightwave Technology 34 (9) (2016) 2085–2091.

[37] R. G. Leonard, G. Doddington, TIDIGITS LDC93S10, Web Download. Philadelphia: Linguistic Data Consortium (1993).

[38] R. Togneri, D. Pullella, An overview of speaker identification: Accuracy and robustness issues, IEEE Circuits and Systems Magazine 11 (2) (2011) 23–61.

[39] R. Alcock, Synthetic control (1999).
URL `http://www.timeseriesclassification.com/description.php?Dataset=SyntheticControl`

[40] R. Alcock, Y. Manolopoulos, Time-series similarity queries employing a feature-based approach, 7th Hellenic Conference on Informatics (1999) 27–29.

[41] R. T. Olszewski, R. A. Maxion, D. P. Siewiorek, Generalized feature extraction for structural pattern recognition in time-series data, Ph.D. thesis, Carnegie Mellon University (2001).

[42] G.-B. Huang, Q.-Y. Zhu, C. K. Siew, Extreme learning machine: Theory and applications., Neurocomputing 70 (1-3) (2006) 489–501.

[43] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, Smote: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research 16 (2002) 321–357.

[44] J. Haddock, J. Mittenthal, Simulation optimization using simulated annealing, Computers and Industrial Engineering 22 (4) (1992) 387–395.

[45] H. Szu, R. Hartley, Fast simulated annealing, Physics Letters A 122 (3) (1987) 157–162.

[46] M.-W. Park, Y.-D. Kim, A systematic procedure for setting parameters in simulated annealing algorithms, Computers and Operations Research 25 (3) (1998) 207–217.

[47] N. T. Thomopoulos, Essentials of Monte Carlo Simulation: Statistical Methods for Building Simulation Models, Springer, 2013.

[48] K. Technologies, M3302A/M3300A PXIe Arbitrary Waveform Generator and Digitizer Combos with Optional Real-Time Sequencing and FPGA Programming, Keysight Technologies (2019).
URL `https://www.keysight.com/fr/en/assets/7018-05403/data-sheets/5992-1809.pdf`

[49] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.