2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Learning the Deterministically Constructed Echo State Networks

Fengzhen Tang
School of Computer Science
University of Birmingham
Edgbaston, Birmingham, B15 2TT, UK
Email: fxt126@cs.bham.ac.uk

Peter Tiňo
School of Computer Science
University of Birmingham
Edgbaston, Birmingham, B15 2TT, UK
Email: P.Tino@cs.bham.ac.uk

Huanhuan Chen
UBRI, School of Computer Science and Technology, University of Science and Technology of China,
Hefei, 230027, China
Email: hchen@ustc.edu.cn

***Abstract*—Echo State Networks (ESNs) have shown great promise in the applications of non-linear time series processing because of their powerful computational ability and efficient training strategy. However, the nature of randomization in the structure of the reservoir causes it be poorly understood and leaves room for further improvements for specific problems. A deterministically constructed reservoir model, Cycle Reservoir with Jumps (CRJ), shows superior generalization performance to standard ESN. However, the weights that govern the structure of the reservoir (reservoir weights) in CRJ model are obtained through exhaustive grid search which is very computational intensive. In this paper, we propose to learn the reservoir weights together with the linear readout weights using a hybrid optimization strategy. The reservoir weights are trained through non-linear optimization techniques while the linear readout weights are obtained through linear algorithms. The experimental results demonstrate that the proposed strategy of training the CRJ network tremendously improves the computational efficiency without jeopardizing the generalization performance, sometimes even with better generalization performance.**

I. Introduction

Reservoir Computing [1] represents a class of new approaches of designing Recurrent Neural Networks [2] for the purpose of accelerating the training process. The training techniques applied in reservoir computing methods make a conceptual and computational separation of the whole process into two parts [1], [3]: representation of temporal structure in the input stream through a non-adaptable dynamic reservoir and a recurrence-free readout mapping that produces the desired output from the reservoir. In these methods, such as Echo Sate Networks (ESNs) [4] and Liquid State Machines (LSMs) [5], the dynamic reservoir is randomly constructed and fixed through the training process, only the readout weights need to be obtained through training. Therefore these methods are very computational efficient.

ESN [6], [7] is one of the pioneering reservoir computing methods. It is of simple form but very effective. Essentially, ESN is a recurrent neural network with a randomly generated and fixed sparse recurrent part (the reservoir) and a very simple linear readout. The reservoir connection weights and the input weights are randomly generated and fixed through the training process. The reservoir connection weights will be scaled in the way that the spectral radius of the reservoir connection weight matrix is less than one. This is to ensure the network has ”Echo State Property” or ” fading memory”, where the network depends far more on the recent history of the input, works efficiently as a Markovian classifier [8] without explicitly making any Markovian assumption. The networks with fading memory have theoretically been proved to possess powerful computational capabilities [9]. In ESN, only the linear readout weights need to train, therefore it is very computational efficient compared with fully trained RNN. Thus, it has been successfully applied in many time-series prediction task, such as speech recognition, dynamic pattern prediction and language modelling [1], [3]. However, it has several limitations. Since the network structure of the reservoir in ESN is randomly constructed, there might be some properties of the reservoir that are poorly understood and reservoir specification requires numerous trials and even luck [10]. Moreover random connectivity and weight structure of the reservoir is unlikely to be optimal [11].

The randomization of generating the reservoir in ESN cause it to be poorly understood, as a result leaving the room for further investigation on what exactly a reservoir structure leads to good performance for a given problem [11]–[13]. A Simple Cycle Reservoir (SCR) introduced in [3] shows comparable performances to the traditional randomized ESN. One extension of SCR by adding regular bidirectional shortcuts (Jumps) (CRJ) introduced in [14] has shown superior performance to those of the traditional randomized reservoir models in non-linear system identification, real time series prediction and speech recognition [14]. Besides, the characterizations of the selected reservoir model is well understood.

Recently, CRJ has been employed for cognitive fault diagnosis [15]. The ability of CRJ to approximate the dynamic system has been verified in this work. Besides, the CRJ has been used for efficient time series classification and its promising performance has been reported in [16].

However, in the original CRJ model introduced in [14], the parameters that govern the design of reservoir are tuned by costly and potentially unstable cross-validation technique using exhaustive grid search. Furthermore, the selected parameters from grid search may not be the optimal, because of the discretization of the continuous parameters.

In this paper, we propose a hybrid optimization strategy to train the CRJ network (TCRJ). The linear output weights of the network are determined by ridge regression, while the reservoir weights are found using a nonlinear optimization technique. Regularization on the output weights and early stopping strategy have been incorporated in this proposed training strategy. The experimental results show that the new learning method tremendously improves the computational efficiency and can achieve comparable, sometimes even better, performance to the original cross-validation CRJ fitting.

This paper is organized as follows. Section II briefly describes Echo State Networks (ESNs). Section III presents the CRJ model. Our proposed algorithm of training the CRJ model are described in details in Section IV. Experimental results and analysis are given in Section V. The main findings are discussed and summarized in Section VI.

II. Echo Sate Networks

Echo State Networks (ESNs) are essentially recurrent neural networks (RNNs) with a randomly generated and fixed sparse recurrent part (the reservoir) and a simple linear readout. ESN can be regarded as a kind of parametric state space models parameterized by input weights V, internal unit connection weights R and output weights W. The input stream $\mathbf{s}$ is mapped into a state vector $\mathbf{x}$ (a vector of reservoir neuron activation), and the output will be described as a linear transformation of the state vector [1]. Given an ESN with d input units, N internal units and O output units, the network architecture can be summarized as Figure 1. The state vector at time t, $\mathbf{x}(t) \in \mathbb{R}^N$, of the network is updated according to:

$$\mathbf{x}(t) = f(R\,\mathbf{x}(t-1) + V\;\mathbf{s}(t)) \tag{1}$$

where $f(\cdot)$ is the activation function, usually $\tanh(\cdot)$ or sigmoid function, applied element-wise, $\mathbf{s}(t) \in \mathbb{R}^d$ is the input vector, matrix $V \in \mathbb{R}^{N\times d}$ describes the connection between the input units and the internal units, matrix $R \in \mathbb{R}^{N\times N}$ collects the connections between internal units. The internal weight matrix R, and the input weight matrix V are randomly generated and fixed through the training. The internal weight matrix R will be scaled in the way that the spectral radius is less than 1, in order to ensure the "Echo state property", where the reservoir state is an "echo" of the input history.

The linear readout is computed follows:

$$\hat{y}(t) = W\mathbf{x}(t) \tag{2}$$

where $\hat{y}(t) \in \mathbb{R}^O$ is the output and matrix $W \in \mathbb{R}^{O\times N}$ represents the connections between internal units and output units[1]. The linear readout parameter W is trained using ridge regression [17]. Collecting the whole state vectors into a matrix $X \in \mathbb{R}^{N\times T}$ and representing the whole true targets in matrix $Y \in \mathbb{R}^{O\times T}$, the output matrix can be obtained:

$$W = YX^T(XX^T + \lambda I)^{-1} \tag{3}$$

where I is the identity matrix and λ is a regularization parameter which can be determined by cross validation.

ESN has been successfully applied in many time-series prediction tasks because of its high computational efficiency. The randomized reservoir construction leads to high computational efficiency (compared with the fully trained RNN) - only the linear readout weights need to be trained through an efficient linear optimization algorithm. Meanwhile, the randomly constructed reservoir also poses limitations. The properties of the reservoir that are poorly understood, and the specification of the reservoir requires numerous trials and even luck [10]. Besides, random construction is quite different from optimal design [11], be it in a constrained setting [14].

[1]As usual, the weight matrices and input/reservoir vectors are extended so that bias terms are accounted for.

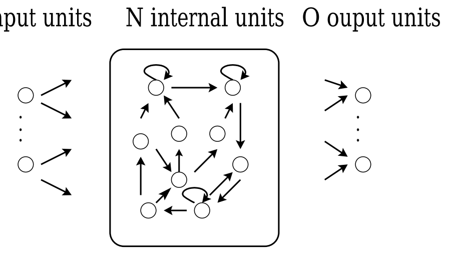

Fig. 1. The network architecture of ESN. In this architecture, the input units to internal units are randomly connected and internal units are randomly connected to each other.

III. Cycle Reservoir with Jumps

Cycle Reservoir with Jumps (CRJ) introduced in [14] deterministically constructs the reservoir in a highly constrained form of unidirectional cycle with regular bidirectional shortcuts. In the CRJ model, the structure of R is particularly simple: reservoir units are connected in a uni-directional cycle with bi-directional short-cuts (jumps) (Figure 2). All cyclic connections have the same weight $r_c > 0$. Likewise, all jumps share the same weight $r_j > 0$. Specifically, R is a very sparse matrix with r_c and r_j spread over, e. g. in a network of 10 internal units with 2 as jump size, the matrix R is of the form as follows:

$$\begin{pmatrix}
0 & 0 & r_j & 0 & 0 & 0 & 0 & 0 & r_j & r_c\\
r_c & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
r_j & r_c & 0 & 0 & r_j & 0 & 0 & 0 & 0 & 0\\
0 & 0 & r_c & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & r_j & r_c & 0 & 0 & r_j & 0 & 0 & 0\\
0 & 0 & 0 & 0 & r_c & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & r_j & r_c & 0 & 0 & r_j & 0\\
0 & 0 & 0 & 0 & 0 & 0 & r_c & 0 & 0 & 0\\
r_j & 0 & 0 & 0 & 0 & 0 & r_j & r_c & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & r_c & 0
\end{pmatrix}$$

The input weight matrix is also highly constrained: the input connections have the same absolute value $r_i > 0$ with an aperiodic sign pattern. The input weight signs are determined by the expansion of the irrational number π (see [14]).

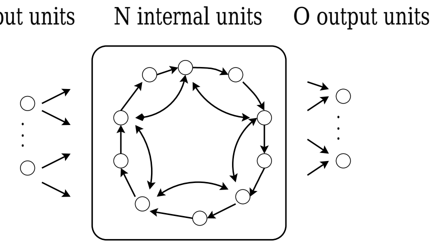

Fig. 2. The network architecture of CRJ. In this architecture, the input units are connected to internal units by the same weight with aperiodic sign. The internal units are connected by directed cycle with regular jumps. All the cycle collection weights are the same and all the jump weights are the same.

The CRJ model has shown superior performance in non-linear system identification, real time series prediction and speech recognition [14], to the randomized ESN. Moreover, the characterizations of the selected reservoir model can be better understood. The selected reservoir meets the "Echo state property".

IV. Training Cycle Reservoir with Jumps

In the original CRJ model introduced in [14], the weights r_c, r_j and r_i are determined by "train-validation-test" approach where the whole time series is divided into three sets, i. e. training set, validation set and test set. The continuous parameter space of r_c, r_j and r_i is discretized into finite combinations. Exhaustive grid search is used to find the best possible parameter setting. For each of the parameter combination, the reservoir will be constructed and the linear readout weights will be obtained on training set, and then the performance of the trained CRJ will be evaluated on the validation set. The best possible parameter combination will be selected on the basis of the minimum error on the validation set. Finally, the reported generalization performance is the error of the network constructed using the selected parameter setting measured on the test set.

In this paper, we propose to learn the parameters r_c, r_j and r_i along with the linear output weights W using a hybrid optimization strategy, where the parameters r_c, r_j and r_i are obtained using non-linear optimization techniques while output weights W are learned using linear optimization methods. The detailed learning algorithm will be presented in the following subsections.

A. Basic Algorithm

The mathematical formulation of the deterministically constructed CRJ model we used in this paper is given as follows:

$$\begin{aligned} \mathbf{x}(t) &= \tanh(R\,\mathbf{x}(t-1) + V\,\mathbf{s}(t)), && (4)\\ \hat{y}(t) &= W\mathbf{x}(t) && (5) \end{aligned}$$

The problem of training the network can be formulated as a problem of the minimization of an error function E. For convenience, r_c, r_j and r_i are grouped together into a single vector $\mathbf{r}$ (spread over coupling matrix R). We choose the sum-of-squares error function (other error functions are equivalent to sum-of-squares error) when training the network. Assuming the network is running from time stamp t_0 up to time step t_1, the sum-of-squares error is given as follows:

$$E(\mathbf{r}, W) = \sum_{t=t_0+1}^{t_1} ||\hat{\mathbf{y}}(t) - \mathbf{y}(t)||^2 \qquad (6)$$

where $\hat{\mathbf{y}}(t)$ is the predicted output and $\mathbf{y}(t)$ is the real output of the sequence at time stamp t and $||\cdot||$ denotes the Euclidean norm. Since the value of the sum-of-squares error function depends on the number of patterns, we consider a normalized error function for assessing the performance of the trained network. In this paper, we choose the normalized mean square error function as follows:

$$\tilde{E}(\mathbf{r}, W) = \frac{\langle ||\hat{\mathbf{y}}(t) - \mathbf{y}(t)||^2 \rangle}{\langle ||\mathbf{y}(t) - \langle \mathbf{y}(t) \rangle||^2 \rangle} \qquad (7)$$

where $\langle \cdot \rangle$ denotes the empirical mean.

The network we considered in this paper is a recurrent network with linear output units. Since the dependence of the network mapping on the final-layer weight is linear, the partial optimization of sum-of-squares error function with respect to these weights can be performed by linear methods. The computational effort involved in linear methods is often very much less than that required for general non-linear optimization. Therefore, in this paper, we adopt a hybrid procedure for optimizing the weights in the network, where linear method is used for obtaining the final layer weights, and non-linear method is used for acquiring all the other parameters [18].

The error function $E(\mathbf{r}, W)$ is a quadratic function of W. For any given value of $\mathbf{r}$, we can perform a one-step exact minimization with respect to the W using linear regression, in which $\mathbf{r}$ is held fixed. The gradient of E with respect to W is as follows:

$$\frac{\partial E}{\partial W} = 2 \sum_{t=t_0+1}^{t_1} (\hat{\mathbf{y}}(t) - \mathbf{y}(t))\mathbf{x}(t) \qquad (8)$$

By making the gradient of E with respect to W equal to zero, the output weights can be computed as follows:

$$W = YX^T(XX^T)^{-1} \qquad (9)$$

Since the output weights W are regarded as a function of the weights $\mathbf{r}$ and can be chosen using Equation (9), we can regard E as a nonlinear function of the reservior weights $\mathbf{r}$ only and a nonlinear function optimization method, e.g. conjugate gradient descent [19] in this paper, is employed to find these weights by minimizing E with respect to $\mathbf{r}$. The gradient of E with respect to $\mathbf{r}$ is computed using real-time recurrent learning [20].

Since the total error is the sum of the errors at the each time steps, we can compute the gradient by summing up the gradient at each time step via

$$\begin{aligned} \frac{\partial E}{\partial \theta} &= 2 \sum_{t=t_0+1}^{t_1} (\hat{\mathbf{y}}(t) - \mathbf{y}(t)) W \frac{\partial \mathbf{x}(t)}{\partial \theta} \\ &+ 2 \sum_{t=t_0+1}^{t_1} (\hat{\mathbf{y}}(t) - \mathbf{y}(t))\mathbf{x}(t) \frac{\partial W}{\partial \theta} \qquad (10) \end{aligned}$$

The gradient of $\mathbf{x}(t)$ with respect to θ, where $\theta = r_c, r_j$ or r_i, can be computed iteratively as:

$$\begin{aligned} \frac{\partial \mathbf{x}(t)}{\partial r_c} &= \mathrm{sech}^2(R\mathbf{x}(t-1) + Vs(t)) \\ .* & \left(R \frac{\partial \mathbf{x}(t-1)}{\partial r_c} + \frac{\partial R}{\partial r_c} \mathbf{x}(t-1) \right), \qquad (11) \end{aligned}$$

$$\frac{\partial \mathbf{x}(t)}{\partial r_j} = \operatorname{sech}^2(R\mathbf{x}(t-1)+Vs(t)) \;.*\; \left(R\frac{\partial \mathbf{x}(t-1)}{\partial r_j} + \frac{\partial R}{\partial r_j}\mathbf{x}(t-1)\right), \quad (12)$$

$$\frac{\partial \mathbf{x}(t)}{\partial r_i} = \operatorname{sech}^2(R\mathbf{x}(t-1)+Vs(t)) \;.*\; \left(\frac{\partial V}{\partial r_i}s(t) + R\frac{\partial \mathbf{x}(t-1)}{\partial r_i}\right), \quad (13)$$

where $.*$ is the element-wise product in matrix calculation (MATLAB notation). As usual in real time recurrent learning, the initial conditions for the update equations can be set to:

$$\frac{\partial \mathbf{x}(t_0)}{\partial r_c} = 0, \; \frac{\partial \mathbf{x}(t_0)}{\partial r_j} = 0, \; \frac{\partial \mathbf{x}(t_0)}{\partial r_i} = 0. \quad (14)$$

The gradient of W with respect to θ is very difficult to compute. However, it is very lucky that the coefficients before $\frac{\partial W}{\partial \theta}$, i.e. $\sum_{t=t_0+1}^{t_1}(\hat{\mathbf{y}}(t)-\mathbf{y}(t))\mathbf{x}(t)$ is essentially equal to zero because we compute W by making it so. Therefore, we don't need to compute the gradient of W with respect to θ and the second term in (10) is vanished.

The reservoir weights $\mathbf{r}$ are obtained using a non-linear optimization algorithm while the output weights W are regarded as a function of $\mathbf{r}$ and are chosen using Equation (9). Every time the value of $\mathbf{r}$ is changed, the weights W need to be recomputed. Thus, the optimization strategy in this paper proceeds the training process on two timescales: Longer timescale, where the weights $\mathbf{r}$ are adjusted to minimize the error function, and a short timescale, where the output weights are changed to minimize the error as a function of the weights $\mathbf{r}$ alone.

The hybrid optimization strategy of combining linear methods and non-linear methods together is chosen in this paper mainly for two reasons: First, the dimensionality of the effective search space for the non-linear algorithm is reduced. Thus, it is possible that the time taken for the nonlinear optimization scheme to find a minimum of the error will be reduced. Second, the network obtained in this way is always in a state where the error is at a global minimum in the space of output weights. This may help the network to reach a minimum more rapidly and to reach a shallow local minimum less often. The approach can be characterized as a group-coordinate-wise descent on the error function, where the parameters are divided into two groups - output readout weights and the reservoir/input weights.

B. Readout regularization

If the training set contains noise, the network with an access of many free coefficients tends to generate mappings which have a lot of curvature and structure, as a result of over-fitting to the noise on the training data. In order to avoid over-fitting, we introduce a quadratic regularization term in the error function to encourage smoother network mapping, as follows:

$$E(\mathbf{r},W) = \sum_{t=t_0+1}^{t_1} ||\hat{\mathbf{y}}(t)-\mathbf{y}(t)||^2 + \lambda||W||^2 \quad (15)$$

The redundant weights will get smaller as the training proceeds. Ideally, the structure of the network will be simplified while the accuracy remains. Thus, the generalization ability can be improved [21]. We only penalize the output weights, since the input weights and cycle connection weights have already been pruned when the highly constrained reservoir is designed. Hence, addition of regularization term will not change the optimization of the reservoir weights $\mathbf{r}$. The output weights W will be computed as follows:

$$W = YX^T(XX^T+\lambda I)^{-1} \quad (16)$$

which is known as ridge regression. The regularization parameter λ can be tuned via cross-validation.

C. Early Stopping

Another way to improve the generalization ability is the procedure of early stopping [19], [22]. The non-linear optimization process of learning the parameter $\mathbf{r}$ corresponds to an iterative reduction of the error function with respect to the training dataset. For many of the optimization algorithms, e.g. conjugate gradient descent, the value of the error function is a nonincreasing function of the iteration index. However, the error measured with respect to a dataset independent of the training, i.e. a validation set, often decreases first and then increases as the network starts to over-fit. Therefore, in order to have a good generalization ability, training needs to be stopped at the point of the smallest error with respect to the validation dataset, rather than the minimum point with respect to the training set.

In this paper, we leave out a validation set. The training process is terminated when the error measured on the validation set starts to increase, in order to optimize the generalization performance of the network.

The overall structure of the training process is briefly demonstrated in Algorithm 1 .

Algorithm 1 The Algorithm of the Training Process

```
1: Initialize $r_c, r_j$ and $r_i$.
2: repeat
3:     Generate CRJ model using $r_c, r_j$ and $r_i$ to obtain the state matrix $X$, and then compute the linear readout weight $W$ using Equation (16).
4:     Assess the performance of the network on the validation set $D_{validation}$ via the normalized mean square error $\tilde{E}(D_{validation})$.
5:     Update the reservoir parameters $r_c, r_j$ and $r_i$ using non-linear optimization algorithm, such as conjugate gradient descent.
6: until $\tilde{E}(D_{validation})$ starts to increase.
```

V. Experimental Studies

In this section, we evaluate our proposed algorithm on a variety of time series prediction tasks. The topology of the network was fixed, with 50 internal units and with the jump size of 15. For comparison, the original CRJ had the same network topology, but the cycle connection weight r_c, jump weight r_j and input weight r_i were chosen via cross-validation through exhaustive grid search and the linear readout weights were fitted using ridge regression. The range of reservoir weights are: r_c, r_j and $r_i \in \{0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. Not every combination of the parameters leads to a network that satisfies the Echo state property, but the optimum or selected parameters do. Standard randomized ESN with the same number of internal units was considered. The linear readout weights of ESN was also learned using ridge regression. The range of the ridge regression parameter is: $\lambda \in \{0.0001, 0.001, 0.01, 0.1, 1, 10, 100\}$. For ESN, we reported the average performance for 10 trials. Since, CRJ and TCRJ are deterministically methods, we only report one result[2].

A. Synthetic data

We first tested our algorithm with a nonlinear system identification task, i. e. a 10th-order NARMA system [7], given by:

$$s(t+1) = 0.3s(t)+0.05s(t)\sum_{i=0}^{9} s(t-i)+1.5u(t-9)u(t)+0.1,$$

where $s(t)$ is the output at time t, $u(t)$ is the input at time t. The inputs $u(t)$ form an i. i. d stream generated uniformly in the interval $[0, 0.5]$. The current output depends on both the input and the previous output. The time series is challenging due to non-linearity and long memory. To make the task harder, we added zero mean and 0.01 variance Gaussian noise to the output stream. The networks were trained to predict output $s(t)$ based on $u(t)$. NARMA sequence has a length of 4000 items where first 2000 were used for training, the following 1000 for validation and the remaining 1000 for testing.

Then the nonlinear Mackey Glass chaotic time series model was used to generate a time series on which we evaluated the proposed algorithm. The series is a solution of the following equation:

$$\frac{dx(t)}{dt} = -ax(t) + \frac{bx(t-\tau)}{1+x^{10}(t-\tau)} \tag{17}$$

where a, b and τ are the parameters of the equation. We used the Mackey-Glass series with parameters $a = 0.1$, $b = 0.2$, $\tau = 30$ and the initial condition $x(\tau) = 1$. A length of 4000 items of this time series was generated and zero mean and 0.05 variance Gaussian noise is added to it. As we did for NARMA sequence, the first 2000 items were used for training, the following 1000 for validation and the remaining 1000 for testing.

The experimental results on these two artificial time series are presented in Tables I and II. Table I shows that our proposed algorithm of learning the reservoir parameters in CRJ slightly outperforms the method of obtaining the reservoir parameters in CRJ through exhaustive grid search in terms of generalization error. As shown in Table II, our proposed algorithm is much better than that of the original CRJ in terms of the computational time, decreasing by 70% to 80%.

TABLE I. Normalized mean square error of the artificial time series

Dataset	ESN	CRJ	TCRJ
NARMA	0.2539±0.0308	0.1019	**0.0981**
MG	0.0900±0.0004	0.0900	**0.0884**

TABLE II. Computational time (s) of the artificial time series

Dataset	ESN	CRJ	TCRJ
NARMA	279	4594	1632
MG	280	4648	962

B. Real-world Time Series Datasets

Three real time series downloaded from website [3] have been used to evaluate the proposed algorithm.

Darwin-SLP represents the monthly values of the Darwin sea level pressures from 1882 to 1998. This series is of length 1400 and a key indicator of climatological patterns. The first 1000 patterns were used for training, the following 200 patterns were used for validation and the remaining 200 patterns were used for test.

Oxygen Isotope Level (*OIL*) series contains measurements of relative abundance of oxygen isotope to oxygen from the deep ocean cores from various geographical locations over a period of about 2.5 million years, whose geological time variations relate to patterns of variation in global ice volume and ocean temperature [23]. This series contains 866 patterns. Given the limited size of the dataset, 5-fold cross validation were performed to tune the parameters, instead of "train-validation-test" approach. The first 600 patterns were used for training and the rest 266 patterns were used for test.

SOI is a series of monthly values of the Southern Oscillation Index during 1950-1995. This series consists of 540 observations on the SOI computed as the difference of the departure from the long-term monthly mean sea level pressures at Tahiti in the South Pacific and Darwin in Northern Australia. As we did for *OIL*, 5-fold cross validation were used to tune the parameters. The first 400 observations were used for training and the remaining 140 observations were used for test.

The experimental results on these three real time series are presented in Tables III and IV. As shown in Table III, the proposed methodology appears to have comparable generalization ability to the original CRJ model where the reservoir parameters was obtaining through exhaustive grid search. The computational time of the proposed algorithm is much less than that of the original CRJ model.

Three datasets from UCR Time Series Repository [24] were used for the evaluation of our proposed algorithm. These datasets are briefly described in Table V. They are mainly used for time series classification. Here we use the sequences to demonstrate predictive power of our models, so each sequence

[2]we use coarse grid search to initialize the reservoir parameters in TCRJ. The time for initialization is included when presenting the experiments

[3]http://isds.duke.edu/~mw/ts_data_sets.html

TABLE III. Normalized mean square error of the *Darwin-SLP*, *OIL* and *SOI* time series

Dataset	ESN	CRJ	TCRJ
Darwin-SLP	0.2467(0.0174)	0.1780	0.1843
OIL	0.2214(0.0008)	0.2094	0.2076
SOI	0.5613(0.0039)	0.5374	0.5374

TABLE IV. Computational time (s) of the *Darwin-SLP*, *OIL* and *SOI* time series

Dataset	ESN	CRJ	TCRJ
Darwin-SLP	230	3539	726
OIL	167	2470	781
SOI	122	1622	381

is divided into training, vakidation and test part as indicated in Table V. The results presented in Tables VI and VII are the average results over the test parts of the sequences considered.

TABLE V. Description of the UCR time series. l is the length of each sequences in the datasets and m is number of sequences in the datasets.

Datasets	l	m	Train/Validation/Test
CinC-ECG-torso	1639	40	1000/300/339
InlineSkate	1882	100	1000/400/482
MALLAT	1024	55	600/200/224

TABLE VI. Normalized mean square error of the UCR time series

Dataset	ESN	CRJ	TCRJ
CinC-ECG-torso	0.0060(0.0154)	0.0054(0.0149)	0.0054(0.0150)
InlineSkate	0.0310(0.0695)	0.0104(0.0313)	**0.0056**(0.0134)
MALLAT	0.0175(0.0138)	0.0152(0.0166)	**0.0117**(0.0147)

The experimental results on the UCR time series presented in Tables VI and VII shows similar trends as we find before.

In this section, two synthetic time series, three real time series and three UCR time series datasets were used to test our algorithm compared with the original exhaustive search CRJ model and standard randomized ESN model. All the experimental results show the same trend that our TCRJ model has comparable performance (sometimes even better) to the original CRJ model in terms of generalization ability, and our TCRJ model is much more computational efficient than that of the original CRJ model.

Our proposed algorithm does not jeopardize the generalization performance on all the datasets used in this paper with the exception of *Darwin-SLP* time series. It may be because the error surface of this time series contains many shallow local minimums and the learning algorithm was stuck in one of the very bad local minimum, while the exhaustive search method reach the relative global minimum (global minimum on the discretized parameter space). However, the generalization performance of our proposed algorithm on *Darwin-SLP* time series is still much better than that of standard randomized ESN model.

Our proposed algorithm tremendously reduces the computational time in the experiments, compared with the original exhaustive search CRJ model. The proposed hybrid optimization strategy keeps the computational advantage of ESN in terms of efficient computation of linear readout weights by the separation of the optimization of linear readout weights from the weights govern the structure of the reservoir. The readout weights are computed in linear techniques and as a result, the error of the network is always at a global minimum in the space of output weights, helping the network to reach a minimum fast. Since the output weights are solved using linear algorithm, the parameters left for non-linear optimization are reduced. The reduction of the search space for the non-linear optimization methods will lead to less number of iteration to terminate the training. Besides, the early stopping strategy potentially helps to reduce the computational time and improve the generalization performance.

TABLE VII. Computational time (s) of the UCR time series

Dataset	ESN	CRJ	TCRJ
CinC-ECG-torso	155(1.51)	2488(28.14)	398(117.78)
InlineSkate	150(5.10)	2237(11.37)	372(81.56)
MALLAT	165(1.92)	2379(142.64)	353(23.18)

VI. Conclusion

ESN has been proved to be highly promising in the applications of non-linear time series processing, because of its powerful computational ability and efficient training strategy. However, the random connectivity and weight structure of the reservoir cause it be poorly understood and leave room for further improvements on performance. Rodan and Tino showed that a simple unidirectional cycle with fixed weight (SCR) [3] is competitive to traditional randomized ESN. Adding regular bidirectional shortcuts on the basis of SCR model (CRJ) originating and ending in few higher-clustering coefficients nodes potentially brings performance improvements and sometimes significantly beats ESN. However, the weights in the original CRJ model is obtained through cross-validation via computational intensive and potentially unstable exhaustive search. In this paper, we propose to learn these weights and the linear readout weights through a hybrid optimization strategy where the weights governing the reservoir structure are trained through non-linear optimization techniques while the linear readout weights are obtained through linear algorithms. Readout regularization and early stopping strategy are applied in our proposed methodology, in order to improving the generalization performance.

The proposed TCRJ was compared with the exhaustive-search CRJ model and the standard randomized ESN. The results show that, compared with the original CRJ model, TCRJ can be much more computationally efficient, while retaining comparable generalization performance. Interestingly enough, TCRJ performs better than ESN in terms of generalization performance. In the exhaustive search, the dominating computation cost of each grid point involves the inversion of a $N \times N$ matrix. Therefore, the time complexity for CRJ model fitting on cross validation will be of order $O(mN^3)$, where m is the number of grid points, while the time complexity of our method is of order $O(kN^3)$, where $k = k_1 + k_2 << m$, k_1 is the number of grid point for initialization and k_2 is the number of iterations in the hybrid learning.

Acknowledgment

Peter Tiňo was supported by the EPSRC grant EP/L000296/1. Huanhuan Chen was supported by the National Natural Science Foundation of China under Grants 61203292, 61311130140 and the One Thousand Young Talents Program. The authors would like to thank Professor Xin Yao from University of Birmingham for comments. Fengzhen Tang would like to thank Liyan Song and Chunlei Xu from University of Birmingham and Yu Sun from University of Science and Technology of China for the discussion and comments.

References

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

[2] B. Hammer and P. Tiňo. Recurrent neural networks with small weights implement definite memory machines. *Neural Networks*, 15(8):1897–1926, 2003.

[3] A. Rodan and P. Tiňo. Simple deterministically constructed recurrent neural networks. In *Proceedings of the 11th international conference on Intelligent data engineering and automated learning*, IDEAL'10, pages 267–274, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] H. Jaeger. The echo state approach to analysing and training recurrent neural networks. Technical report, German National Research Center for Information Technology, 2001.

[5] W. Maass, T. Natschlager, and H. Markram. Real-time computing without stable states: a new framework for neural computation based on perturbations. *Neural Computation*, 14(11):2513 – 2560, 2002.

[6] H. Jaeger. Short term memory in echo state networks. Technical report, German National Research Center for Information Technology, 2002.

[7] H. Jaeger. Adaptive nonlinear system identification with echo state networks. In *Advances in Neural Information Processing Systems*, pages 593–600. MIT Press, Cambridge, MA,, 2003.

[8] S. Gallicchio and A. Micheli. Architectural and markovian factors of echo state networks. *Neural Networks*, 24(5):440–456, 2011.

[9] O. L. White, D. D. Lee, and H. Sompolinsky. Short-term memory in orthogonal neural networks. *Physical Review Letters*, 92(14):148102.1–148102.4, 2004.

[10] Y. Xue, L. Yang, and S. Haykin. Decoupled echo state networks with lateral inhibition. *Neural Networks*, 20(3):365–376, 2007.

[11] M. C. Ozturk, D. Xu, and J.C. Principe. Analysis and design of echo state network. *Neural Computation*, 19(1):111–138, 2007.

[12] S. Hausler, H. Markram, and W. Maass. Perspectives of the high-dimensional dynamics of neural microcircuits from the point of view of low-dimensional readouts. *Special Issume on Complex Adaptive Systems*, 8(4):39–50, 2003.

[13] A. Rodan and P. Tiňo. Minimum complexity echo state network. *IEEE Transactions on Neural Networks*, 22(1):131–144, 2011.

[14] A. Rodan and P. Tiňo. Simple deterministically constructed cycle reservoirs with regular jumps. *Neural Computation*, 24(7):1822–1852, 2012.

[15] H. Chen, P. Tiňo, A. Rodan, and X. Yao. Learning in the model space for cognitive fault diagnosis. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):124–136, 2014.

[16] H. Chen, F. Tang, P. Tino, and X. Yao. Model-based kernel for efficient time series analysis. In *Proceedings of the 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'13)*, pages 392–400, 2013.

[17] Z. Shi and M. Han. Ridge regression learning in esn for chaotic time series prediction. *Control and Decision*, 22(3):258–267, 2007.

[18] A. R. Webb and D. Lowe. A hybrid optimization strategy for adaptive feed-forward layered networks. Technical report, RSRE Memorandum 4193, Royal Signals and Radar Establishment, St Andrews Road, Malvern, UK, 1988.

[19] C. M. Bishop. *Neural networks for pattern recognition*. Clarendon Press, Oxford, 1995.

[20] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[21] D. Niu, L. Ji, Xing M., and J. Wang. Multi-variable echo state network optimization by bayesian regulation for daily peak load forecasting. *Journal of Networks*, 7(11):1790–1795, 2012.

[22] C. M. Bishop. *Pattern Recognition and machine learnig*. Springer, 2006.

[23] M. West and J. Harrison. *Bayesian forecasting and dynamic models*. Springer-Verlag New York, Inc, 2nd edition, 1997.

[24] E. Keogh, Q. Zhu, B. Hu, Hao. Y., X. Xi, L. Wei, and C. A. Ratanamahatana. The ucr time series classification/clustering, 2011. http://www.cs.ucr.edu/~eamonn/time_series_data/.