Neural Networks 38 (2013) 76–89

Contents lists available at SciVerse ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Reservoir computing and extreme learning machines for non-linear time-series data analysis

J.B. Butcher [a,*], D. Verstraeten [b], B. Schrauwen [b], C.R. Day [a], P.W. Haycock [a]

[a] Institute for the Environment, Physical Sciences and Applied Mathematics (EPSAM), Keele University, Staffordshire, ST5 5BG, United Kingdom
[b] Department of Electronics and Information Systems (ELIS), Ghent University, Sint-Pietersnieuwstraat 41, 9000 Ghent, Belgium

ARTICLE INFO

Article history:
Received 25 May 2012
Received in revised form 8 October 2012
Accepted 21 November 2012

Keywords:
Reservoir computing
Extreme learning machine
Reservoir with random static projections
Non-linearity
Short-term memory
Time-series data

ABSTRACT

Random projection architectures such as Echo state networks (ESNs) and Extreme Learning Machines (ELMs) use a network containing a randomly connected hidden layer and train only the output weights, overcoming the problems associated with the complex and computationally demanding training algorithms traditionally used to train neural networks, particularly recurrent neural networks. In this study an ESN is shown to contain an antagonistic trade-off between the amount of non-linear mapping and short-term memory it can exhibit when applied to time-series data which are highly non-linear. To overcome this trade-off a new architecture, Reservoir with Random Static Projections (R^2SP) is investigated, that is shown to offer a significant improvement in performance. A similar approach using an ELM whose input is presented through a time delay (TD-ELM) is shown to further enhance performance where it significantly outperformed the ESN and R^2SP as well other architectures when applied to a novel task which allows the short-term memory and non-linearity to be varied. The hard-limiting memory of the TD-ELM appears to be best suited for the data investigated in this study, although ESN-based approaches may offer improved performance when processing data which require a longer fading memory.

1. Introduction

Reservoir Computing (RC) (Verstraeten, Schrauwen, D'Haene, & Stroobandt, 2007) and Extreme Learning Machines (ELMs) (Huang, Zhu, & Siew, 2006) are two relatively recent additions to the fields of recurrent neural networks (RNNs) and single layer feedforward neural networks (SLFNNs). Both overcome the problems typically associated with traditional neural network training algorithms, such as local minima and vanishing gradients, by modifying only the output weights using a simple and efficient linear regression algorithm. The main difference between the two approaches is that the reservoir of RC architectures contains recurrent connections, giving it a short-term memory, whereas ELMs, being a feedforward architecture, have no short-term memory. ELMs are introduced briefly in the next section, while RC and the properties of RC architectures are discussed in Section 3.

2. Extreme learning machines

The ELM (Huang et al., 2006) approach is an SLFNN with an input, a hidden and an output layer of neurons in which the input weights and bias to the hidden layer are randomly selected. ELMs, provided that the activation functions of the hidden neurons are infinitely differentiable (for example, sigmoid activation functions), have been shown to outperform other approaches, including multilayer perceptrons (MLPs) and support vector machines (SVMs), when applied to some benchmark tasks (Huang et al., 2006) with the added benefit of fast and efficient training procedures. The activations of the ELM hidden layer neurons are defined by:

$$\mathbf{x}(t) = f\left(\mathbf{W}_{hl}^{inp}\mathbf{u}(t) + \mathbf{b}\right) \tag{1}$$

where $\mathbf{x}(t)$ is the vector of hidden layer neuron activations, $\mathbf{W}_{hl}^{inp}$ is the input to hidden layer weight matrix (note: $\mathbf{W}_b^a$ denotes a weight matrix connecting layer a to layer b), $\mathbf{u}$ is the vector of inputs, $f(\mathbf{z})$ is the activation function (tanh in this work) and $\mathbf{b}$ is the bias vector. The input weights were either -0.1 or $+0.1$ selected randomly from a uniform distribution. The output vector of the ELM is calculated according to:

$$\mathbf{y}(t) = \mathbf{W}_{out}^{hl}\mathbf{x}(t) \tag{2}$$

where $\mathbf{W}_{out}^{hl}$ is the output weight matrix, calculated using ridge regression (Montgomery & Peck, 1982):

$$\mathbf{W}_{out}^{hl} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{Y}_{tgt} \tag{3}$$

* Corresponding author.
E-mail address: j.b.butcher@cs.keele.ac.uk (J.B. Butcher).

0893-6080/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.neunet.2012.11.011

where λ is the regularisation parameter which is found in this work using a grid search with 10-fold cross-validation, $\mathbf{I}$ is the identity matrix and $\mathbf{X}$ is the matrix of all hidden layer neurons' activations over the length of the data.

A thorough overview of ELMs is outside the scope of this article and readers are directed to Huang and Wang (2011) for recent developments in this area.

3. Reservoir computing

Reservoir Computing (RC) (Verstraeten et al., 2007) is an RNN technique that offers a solution to the many problems associated with typical RNN architectures which have prevented their widespread use (Lukoševičius & Jaeger, 2009). One RC technique, Echo State Networks (ESNs) (Jaeger, 2002a) which contain analog neurons (using the tanh activation function) are investigated in this study.

Since its conception RC, and its derivatives such as those discussed in this paper, have been successfully applied to a wide range of real-world domains including time-series prediction (Jaeger, 2001; Jaeger & Haas, 2004), robotics (Antonelo, Schrauwen, & Stroobandt, 2008), speech recognition (Butcher, Verstraeten, Schrauwen, Day, & Haycock, 2010a; Verstraeten, Schrauwen, Stroobandt, & Van Campenhout, 2005), non-linear audio processing (Holzmann, 2009) and defect detection in reinforced concrete (Butcher, Verstraeten, Schrauwen, Day, & Haycock, submitted for publication). In all of these studies, reservoirs were found to equal, if not outperform, the state-of-the-art technique with the added advantage of using a simple and more efficient training procedure.

A general ESN architecture involves an input layer, a randomly interconnected layer, the reservoir, and an output layer, each of which is connected in the forward direction to its neighbouring layers. Fig. 1 shows an ESN architecture.

The activation of the ESN reservoir neurons is evaluated as follows:

$$\mathbf{x}(t) = f\left((1-\delta)\mathbf{x}(t-1) + \delta(\mathbf{W}_{res}^{inp}\mathbf{u}(t-1) + \mathbf{W}_{res}^{res}\mathbf{x}(t-1))\right) \quad (4)$$

where $\mathbf{W}_{res}^{inp}$ is the input to reservoir weight matrix (either -0.1 or $+0.1$ selected randomly from a uniform distribution), $\mathbf{x}(t)$ are the activations of the reservoir neurons at time t, which is the current time step, $t-1$ is the previous time step, $\mathbf{W}_{res}^{res}$ is the reservoir weight matrix (drawn randomly from a Gaussian distribution with zero mean and a variance of 1) and δ is the *leak rate* which determines the extent to which the ESN reservoir neuron's activation decreases over a period of time. The leak rate varies inversely with the extent of the ESN's short term memory where smaller values increase the ability of reservoir neurons to recall inputs presented further in the past, but decrease their ability to recall the most recent inputs (Verstraeten, 2009).

Another parameter which influences the behaviour of the reservoir is the *input scale* (ι) which determines the extent to which the ESN reservoir is driven by the input, in other words the extent to which the current input impacts the next state of the reservoir. The default value for ι is often unity. A small input scale ($\iota < 1$) creates a linear reservoir as most of the reservoir neurons stay within the linear range of their activation functions (assuming other parameters are within their normal range). Increasing the input scale ($\iota > 1$) moves the working point of a reservoir neuron's activation function towards its regions of non-linearity (see Fig. 2 below). A very large input scale leads to the saturation of the reservoir where all of its neurons' activations are close to either -1 or $+1$.

Once the activations have been calculated for all of the data in a particular training dataset, the output weights can be trained using a simple linear regression technique, such as ridge regression (Montgomery & Peck, 1982) as shown above in Eq. (3) where $\mathbf{W}_{out}^{hl}$ is replaced by $\mathbf{W}_{out}^{res}$. Once the output weights have been calculated, the actual output of the network is:

$$\mathbf{y}(t) = f^{out}\left(\mathbf{W}_{out}^{res}(\mathbf{x}(t))\right) \quad (5)$$

where f^{out} is the activation function of the output unit, which is linear. As regression is based on least squares error minimisation, the calculation of the output weights based on the parameters of the network always results in an optimal solution, avoiding the pitfall of local minima from which the more traditional RNNs suffer. Referring back to the ELM introduced in the previous section, since only the hidden layer to output layer weighted connections are trained using linear regression (in most studies the Moore–Penrose pseudoinverse Moore, 1920; Penrose, 1955 is used), the ELM can be viewed as a kind of memoryless reservoir.

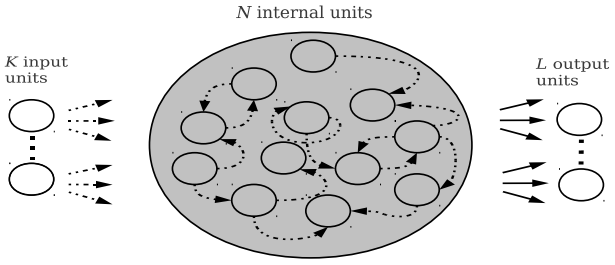


Fig. 1. A schematic representation of an ESN where the dotted lines indicate weighted connections which are left untrained after initialisation and rescaling and solid lines indicate the weighted connections which are modified during training.

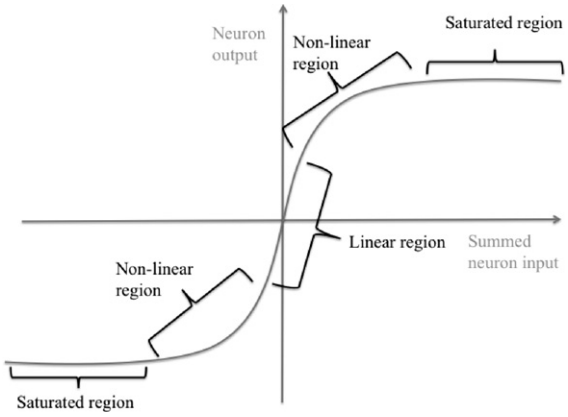


Fig. 2. The tanh activation function and the different regions where a neuron's working point can be found.

3.1. The echo state property

In order to obtain an ESN which performs well on a given task, its reservoir should satisfy the echo state property (Jaeger, 2001). Informally this means that the reservoir of the ESN should have a fading memory. This means that the current inputs have a greater influence on the current state of the reservoir neurons than inputs presented in the past and that any inputs should eventually fade away in the reservoir after a period of time, allowing the reservoir to settle back to a stable state. An ESN whose reservoir satisfies the echo state property should eventually, for any input, wash out (remove) any initial reservoir conditions created at initialisation (Jaeger, Lukoševičius, Popovici, & Siewert, 2007).

In order to satisfy the echo state property the largest absolute value of the eigenvalues of the ESN reservoir weight matrix, i.e. the *spectral radius* (ρ), should be close to, but not equal to unity (Jaeger, 2001). ρ can be interpreted as the forgetting factor of the network, as it determines the extent to which the network forgets its previous inputs. ρ values above unity mean that the echo state

property disappears for zero input, although a value higher than unity for non-zero data does not result in the disappearance of the echo state property (as is sometimes claimed), because input data actually stabilises the reservoir (see Verstraeten, Schrauwen, & Stroobandt, 2006, for further discussion).

The input scale and spectral radius can have similar effects on the dynamics of the reservoir of an ESN: low values for the input scale and spectral radius create a linear reservoir with a high memory capacity (linear reservoirs were shown to possess the highest MC possible Jaeger, 2002b), whereas values higher than unity for the input scale and spectral radius create a very non-linear reservoir which has a low memory capacity. The main difference between the two parameters is that while the input scale influences the impact of the current input ($\mathbf{u}(t)$) on the next state of the reservoir ($\mathbf{x}(t+1)$), the spectral radius influences the impact of the current state ($\mathbf{x}(t)$, which has been driven by all previous inputs) on the next state of the reservoir ($\mathbf{x}(t+1)$). For further details on the effects of these parameters on the dynamics of an ESN reservoir, see Butcher (2012).

3.2. Interpreting reservoir dynamics

Although ESNs have been shown to perform well in particular tasks, due to their black box nature it can be unclear what characteristics of the network and/or problem domains enable them to perform well. As a result various reservoir measures have been introduced to shed light on certain characteristics of an ESN's dynamics, two of which are presented below.

3.2.1. Deviation from linearity

The deviation from linearity (δ_ϕ) (Verstraeten, Dambre, Dutoit, & Schrauwen, 2010) is a measure of the 'energy' ('energy' here is a measure of the ESN's reservoir activations) present in an ESN's reservoir stimulated by a sine wave of a particular frequency. Any other energy in the reservoir at different frequencies is caused by non-linearities present inside it. The measure is based on the observation that a purely linear ESN reservoir contains energy only at the frequency of the input signal, whereas an ESN reservoir with non-linear characteristics will contain other frequencies. Calculating δ_ϕ for a given ESN reservoir involves the following steps.

- Create a single input dataset containing 100 sine waves with carrier frequencies linearly spaced from 0.01 to 0.5 Hz.
- Present the ESN with the first sine wave which has a frequency f_{c1}.
- Perform a Fast Fourier Transform (FFT) of the ESN reservoir activations for all neurons and average. Calculate $E_{tot} = mean(FFT(\mathbf{X})) - DC_{cmp}(\mathbf{u})$ where $DC_{cmp}(\mathbf{u})$ is the DC component (the average value of the signal) of the input signal.
- Calculate $\delta_{\phi_1} = 1 - \frac{E_{c1}}{E_{tot}}$ where E_{c1} is the energy of the FFT at f_{c1}.
- Repeat for all input signals from f_{c2} to f_{c100} and compute the average δ_ϕ.

3.2.2. Memory capacity

The Memory Capacity task (Jaeger, 2002b) provides a measure of an ESN's short-term memory. This task provides a measure of how well the ESN's reservoir can remember past inputs by forcing it to reproduce the input (randomly selected from a uniform distribution between -0.8 and $+0.8$) which was presented d time steps ago where d is the delay. For example, the 1st output neuron is trained to reproduce the input one timestep ago, the 2nd output neuron two timesteps ago and so on. The activation of each output neuron (d) at time t is, therefore, defined as:

$$\mathbf{y}^d(t) = \mathbf{u}(t-d). \tag{6}$$

The memory capacity for the time delay d is:

$$\mathbf{MC}_d = \frac{\mathrm{Cov}^2([\hat{\mathbf{y}}^d, \mathbf{y}^d_{tgt}])}{\mathrm{Var}[\hat{\mathbf{y}}^d]\mathrm{Var}[\mathbf{y}^d_{tgt}]} \tag{7}$$

where $\hat{\mathbf{y}}^d$ is the actual output of the network for delay d, $\mathbf{y}^d_{tgt}$ is the target output for delay d and Cov and Var are the covariance and variance. The total memory capacity for all delays is:

$$\mathbf{MC} = \sum_{d=1}^{\infty} \mathbf{MC}_d. \tag{8}$$

3.3. Non-linear mapping and memory capacity of ESNs

The attraction of applying RNNs to time-series data comes from their ability to learn the relationship between a train of successive inputs and output data, a relationship that can often be highly non-linear, making linear solutions ill-suited. In RC, these desirable characteristics also apply to a reservoir, but with the added advantage of simple and fast training.

Inspection of Fig. 2 shows that the tanh activation function has effectively five characteristic regions. Reservoirs with neurons mostly operating within each of these regions will possess different emergent properties. The working point will generally be in the linear region of the activation function given a small input signal, an input scale smaller than unity and spectral radius close to but smaller than unity. Populations of neurons whose activations are in this region will create a reservoir with the highest MC: maximised by small input scales and a spectral radius close to unity. For reservoirs of this type, the amount of non-linearity present (as measured by deviation from linearity) will be at its minimum.

Perhaps the two parameters which have the largest influence on reservoir dynamics are the input scale and the spectral radius. Higher input scaling and spectral radius (values over unity) decrease the memory capacity of a reservoir as a result of moving the working points of its neurons to more non-linear regions. Ideally, one would like to obtain an ESN whose reservoir has as high memory capacity as possible, while also containing non-linear mappings of input signals in order to separate non-linear data (e.g. in the case of classification).[1] Currently, however, it is not possible to obtain an ESN with a reservoir which contains high amounts of short-term memory and non-linear transformations simultaneously.

This problem was briefly mentioned in one of Jaeger's first publications on ESNs (Jaeger, 2002b) where, in order to achieve a long short-term memory in a reservoir, the use of small input weights was recommended. Jaeger conjectured that this may conflict with non-linear task characteristics where a larger input scale may be required.

This problem was also commented on by Büsing, Schrauwen, and Legenstein (2010) where the spread of a reservoir's state space and the memory capacity of reservoirs with binary (spiking) and analog (sigmoid) neurons were investigated. Büsing et al. pointed out that linear networks such as networks with delay lines have a high MC but perform poorly on non-linear datasets. The reservoirs reported were able to offer good performance on the datasets studied, but had memory capacity capabilities of the order of $\mathcal{O}\log(N)$ compared to $\mathcal{O}(N)$ of linear networks (Büsing et al., 2010).

The antagonistic trade-off between the non-linear mapping capabilities and the memory capacity of an ESN reservoir was

[1] The same applies for regression problems, but classification is presented for clarity

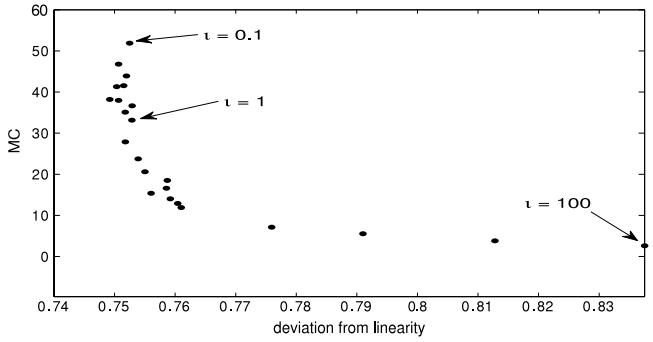


Fig. 3. The effect of varying the input scale on the memory capacity and deviation from linearity of an ESN reservoir containing 200 neurons showing a trade-off between the two characteristics.

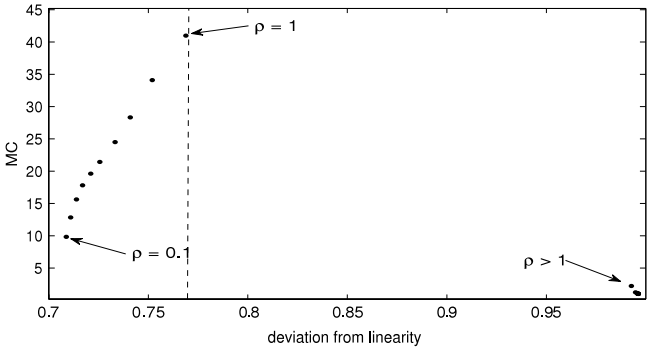


Fig. 4. The effect of varying the spectral radius on the memory capacity and linearity of an ESN reservoir showing a trade-off between the two characteristics.

later investigated by Verstraeten et al. (2010). The memory requirements of a task were often found to predominate over the non-linear mapping capabilities of a reservoir when processing data with high amounts of non-linearities and short-term memory. This is due to the fact that the ESNs applied to this dataset offered better performance when tuned towards a longer short-term memory than an ESN reservoir which contained more non-linearities (Verstraeten et al., 2010).

A highly non-linear time-series dataset can typically be identified if an ESN with an input scale and/or spectral radius much higher than unity is obtained (enabling best performance for some tasks, where little MC is perhaps required). Although when saturated, a neuron's ability to distinguish a difference between small changes in its inputs decreases since small changes in its summed inputs leads to a small change in its output. In the case of highly non-linear data which can change dramatically from one time step to the next, this is not so problematic as the change in input is such to create a substantial change in the neuron's output. In this case the neuron maps one input to an area in its state space, while a different input will be mapped to a completely different area of its state space, enabling the non-linear separation of the highly non-linear time-series data. This may offer an explanation as to why, for datasets which are highly non-linear and require some short-term memory, large input scales and/or spectral radius values may offer best performance.

For datasets which are highly non-linear and require little MC (or which are very linear and require a long MC), this trade-off is not an issue for a particular ESN as its reservoir neurons can move to their non-linear or saturated regions (or linear regions) and perform the non-linear mapping (memory of previous inputs) required from the task. For datasets which require both properties, a trade-off between MC and non-linear mapping has to be made and MC has been shown to be given priority for particular datasets (Verstraeten et al., 2010). Consequently this reduces an ESN reservoir's ability to perform highly non-linear mappings of the input data, resulting in inferior performance.

To illustrate this trade-off, the deviation from linearity and the MC measures are plotted in Fig. 3. Here the effect of varying the input scale over the range 0.1–100 (using increments of 0.1 until unity was reached, after which the input scale was increased by one until a value of 10 was reached, subsequently input scales of 20, 30, 50 and 100 were used) on these two measures is shown (note that the spectral radius is 1 here). Each point on the figure is an ESN with particular input scales which has been presented with random input.

Fig. 3 shows the trade-off between the memory capacity and the linearity of the reservoir: the higher the MC, the more linear the reservoir; the more non-linear the reservoir, the lower the MC. The figure also shows that the highest MCs are obtained with input scale values smaller than unity. Once the input scale of the reservoir exceeds unity its MC decreases dramatically while the non-linearities present inside its reservoir increase. Fig. 4 below shows the effect of increasing the spectral radius using the same values as used for the input scale in Fig. 3 (note that the input scale is 1 here).

Fig. 4 shows that a value of unity for an ESN reservoir's spectral radius gives the highest memory capacity. The lowest value for the ESN reservoir's deviation from linearity can be observed when its spectral radius is smallest at 0.1. This makes the ESN reservoir activations very contractive, which means that their recurrent activations soon die out, reducing its memory capacity and also the non-linearities it contains. The figure also shows that to obtain the highest memory capacity, the amount of non-linearity the reservoir of the ESN contains must also increase. However, after the non-linearity inside the ESN reservoir exceeds 0.77, it becomes less able to recall inputs from the past, as shown by the vast reduction in its MC. Note the greater effect the spectral radius has on the dynamics of the ESN reservoir, where values larger than unity increase the non-linearities while decreasing the MC of the reservoir more drastically than input scales greater than unity.

When processing highly non-linear data, which changes dramatically from one time step to the next, an ESN's reservoir must be able to map each input to a separate region of its state space in order to classify each data point correctly. This ability is compromised when dealing with data which contains high amounts of non-linearity and short-term memory as the reservoir is tuned towards maximising either short-term memory (as was the case in Verstraeten et al., 2010) or non-linear mapping of the input (as was shown in Butcher et al., submitted for publication). The next section introduces two approaches that may offer a solution to this problem: an extreme learning machine approach whose input is presented as a successive window of inputs; and a novel extension to the ESN architecture, *Reservoir with Random Static Projections*.

4. Overcoming the trade-off between non-linear mapping and memory capacity

4.1. Extreme learning machines and time-delayed inputs

An ELM with a sliding window of time-delayed input (TD-ELM), which has an input layer of $N \times K$ neurons, where N is the length of the window and K is the dimensionality of the input and corresponds to the number of original ELM input layer neurons, was investigated. At each time-step, the window was shifted through the input data by one time-step. A schematic representation of this implementation is shown in Fig. 5 which also depicts a standard ELM, shown for comparison. The TD-ELM approach required the optimisation of the window size: found using an extensive empirical search. Presenting data to a network

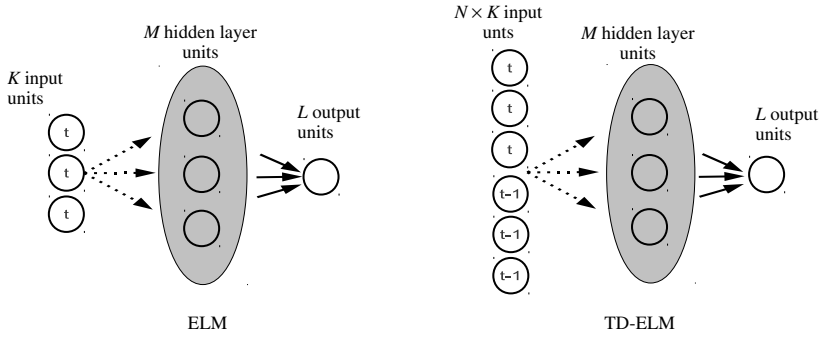


Fig. 5. An ELM (left) and a TD-ELM (right) with a delay line as its input. The delay line transforms the input data into a window of length N, which in this figure is two time steps. The solid arrows indicate connections which are modified during training, while the dashed arrows are weighted connections initialised at network creation and remain unaltered.

in this format resulted in an increase in the size of the input layer, although the fast training of the ELM meant that this was not problematic overall.

By combining an ELM with a time-delayed moving window of the input data, the network was able possess a short-term memory of the input data that extended as far as the window length, while the memoryless hidden layer would transform the data onto a high dimensional state space, easing the antagonistic trade-off presented in the previous section.

Previous studies have investigated the use of a moving window in ELM networks for stationary and non-stationary time-series data analysis. For example, van Heeswijk et al. (2009) used an ensemble of ELMs for time-series prediction. The different ELMs were trained on the original input data (i.e. not using the moving window) and then compared against ensembles of the same configuration using a sliding window and a growing window (i.e. all of the input data presented so far) of the input data. On a stationary dataset a growing window was found to offer the best performance, while on the non-stationary dataset presenting input to the network using a sliding window gave the best performance. Other studies have shown an ELM with a moving window to offer good results also (Chen & Ou, 2011; Singh & Balasundaram, 2007).

4.2. Reservoir with random static projections

A novel RC approach has recently been proposed which combines an ESN with two ELM-inspired hidden layers to give a Reservoir with Random Static Projections (R^2SP) (Butcher et al., 2010a;. Butcher, Verstraeten, Schrauwen, Day, & Haycock, 2010b) architecture. As ELMs use the same simple training approach as RC and have been successfully applied to a variety of problem domains, it was conjectured that ELMs and RC, appropriately conjugated into a single architecture, could usefully complement each other. The two approaches are combined to form the R^2SP architecture: where an ESN reservoir is tuned to perform the memory mapping of the data (due to its recurrent connections) while two separate ELM layers perform an instantaneous non-linear transformation of the input data and the reservoir states, thereby overcoming the antagonistic trade-off between short-term memory and non-linear transformation within a single network.

As the R^2SP combines an ESN reservoir with an ELM, the activations of the reservoir neurons are obtained as outlined in Eq. (4) above (the reservoir of the R^2SP is denoted as R^2SP-reservoir). One set of feedforward neurons of a second ELM were added in parallel with the R^2SP-reservoir to give an instantaneous non-linear mapping of the input from the current time step. Feedforward neurons of an ELM were also placed between the

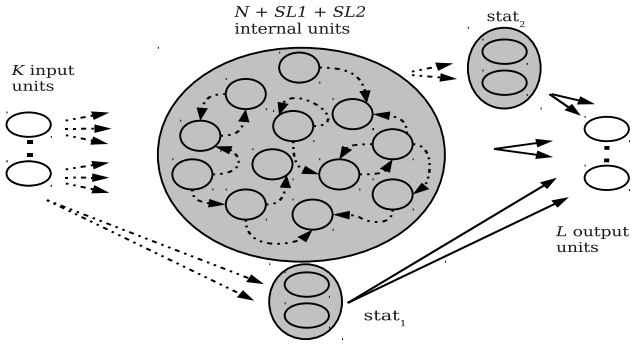


Fig. 6. The R^2SP architecture. SL1 and SL2 denote the number of neurons in the static (feedforward) layers $stat_1$ and $stat_2$ respectively. The solid arrows indicate the weighted connections which are modified during training. The remaining connections (as indicated by the dotted lines) are left unmodified after network initialisation.

R^2SP-reservoir and the output layer to give an instantaneous non-linear mapping of the R^2SP-reservoir at each time step. These additional ELM layers are denoted $stat_1$ and $stat_2$ respectively. The activations of $stat_1$ and $stat_2$ are defined as:

$$\mathbf{x}_{stat_1}(t) = f\left(\mathbf{W}^{inp}_{stat_1}\mathbf{u}(t)\right) \tag{9}$$

$$\mathbf{x}_{stat_2}(t) = f\left(\mathbf{W}^{res}_{stat_2}\mathbf{x}(t)\right) \tag{10}$$

where $\mathbf{x}_{stat_1}(t)$, $\mathbf{x}_{stat_2}(t)$, $\mathbf{W}^{inp}_{stat_1}$ and $\mathbf{W}^{res}_{stat_2}$ are the activations and input weight matrices of $stat_1$ and $stat_2$ respectively. All other symbols are as in Eqs. (4) and (5). The initial weights for the matrices $\mathbf{W}^{inp}_{stat_1}$ and $\mathbf{W}^{res}_{stat_2}$ are set to either -0.1 or $+0.1$ selected randomly from a uniform distribution. The three sets of activations (the R^2SP-reservoir in addition to $stat_1$ and $stat_2$) are combined in order to calculate the output weights using ridge regression. Fig. 6 shows a schematic overview of the R^2SP architecture.

Since $stat_1$ provided an instantaneous non-linear mapping of the input data it was expected to perform the majority of the non-linear mapping of the input data onto a higher dimensional state space. As the reaction of $stat_1$ neurons to a change in input data is instantaneous (compared to an ESN reservoir which can take some time to react to changes in inputs as a result of its recurrent connections), $stat_1$ was also expected to help improve performance of the network when presented with fast changing input data.

The role of $stat_2$ was to map the R^2SP-reservoir activations onto a more non-linear state space than the ESN alone could achieve: if best performance of the R^2SP network were to result in the R^2SP-reservoir being tuned towards maximum MC possible, then its state space would not be highly non-linear. The

advantage of transforming reservoir activations onto a more non-linear state space is that it aids the readout layer in separating datapoints which are mapped onto a larger more non-linear state space, improving performance when compared to a standard ESN architecture as shown in Gallicchio and Micheli (2011). In the study by Gallicchio and Micheli (2011) a static feedforward layer was added between the reservoir layer and the readout layer to give a φ-ESN. All of the reservoir neurons were connected to the added static layer. The output weights were then calculated based on the activations of the static layer. The φ-ESN approach differs from the newly proposed R^{2}SP approach as the R^{2}SP output weights are calculated based on the activations of the R^{2}SP-reservoir neurons, $stat_1$ neurons and $stat_2$'s neurons (which perform the same role as the static layer of the φ-ESN).

5. Experimental setup

Previous studies (Butcher et al., 2010a, 2010b, submitted for publication) have shown the R^{2}SP architecture to outperform an ESN on three datasets: a fourth order polynomial, real-world spoken utterances of the digits 0–9 from the TI46 spoken digit dataset (Doddington & Schalk, 1981) and real-world data collected from reinforced concrete structures. One of the important contributions of the work reported here is an extension of the fourth order polynomial dataset to allow the amount of non-linearity and memory present in the output data to be explicitly controlled.

Training of the ESN, TD-ELM and R^{2}SP networks was conducted using 10-fold cross-validation with a grid-search to find the optimal regularisation parameter for ridge regression. Cross-validation removed the danger of training and testing using unrepresentative portions of the dataset, while the grid-search ensured that the networks did not over-fit the dataset.

In order to find the best parameters of an ESN and an R^{2}SP when applied to the dataset (outlined below), a broad parameter sweep was conducted using the RCToolbox,[2] ranging over: the input scale, spectral radius, leak rate, bias to reservoir neurons and reservoir size. The input scale of the TD-ELM was also found using this approach. To reduce the search space of the parameter sweep, the number of neurons used in the hidden layer of each architecture was determined by finding the size of reservoir layer that enabled the best test performance for the ESN when applied to the polynomial task using p and d values of unity. The hidden layer sizes of the R^{2}SP and TD-ELM were then also set at this value to give each architecture the same number of trainable parameters (i.e. output weights) in order to enable as fair a comparison as possible. The size of the reservoir of the ESN was increased by increments of 100 neurons over the range 100–1000 each time using 10-fold cross-validation as described earlier.

As shown in Fig. 7 an ESN reservoir layer size of 600 neurons was found to offer the best performance, which was confirmed as significant according to the two-sided t-test (Rees, 2000). The total number of neurons in the hidden layer of the R^{2}SP and the hidden layer of the TD-ELM were then set at 600 neurons to give the same number of trainable parameters as the ESN, whereas an R^{2}SP-reservoir, $stat_1$ and $stat_2$ size of 200 was chosen. The remaining parameters were then ranged over to find the values which gave best performance, where the input scale was varied over the range 10^{-1}–10^{2} using a power increment of 0.4, the spectral radius was varied over the range 0.1–1.5 using increments of 0.2, the leak rate was varied between 10^{-5} and $10^{2.5}$ using power increments of 0.5 and the bias to reservoir neurons was varied over the range 0–0.5 using increments of 0.05.

[2] An opensource Matlab toolbox available from http://www.elis.ugent.be/rct.

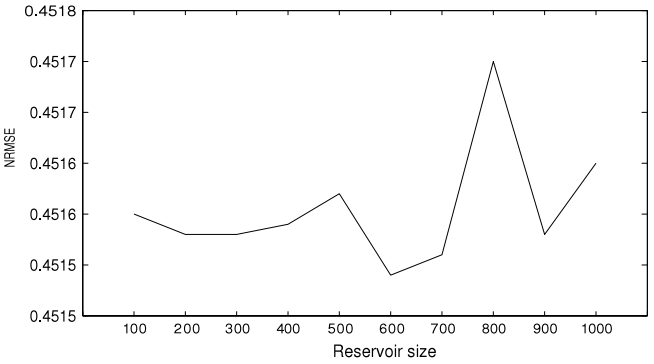


Fig. 7. The test performance of the ESN when applied to the extended polynomial task using p and d values of unity as a function of its reservoir size showing that a reservoir size of 600 neurons enabled best performance.

In order to overcome the variations in performance due to the random initialisation of the ESN, R^{2}SP and TD-ELM networks, 100 networks with the parameter settings that enabled best performance were simulated for each dataset. In the results reported below, the errors were averaged and their standard deviations are shown in parentheses.

6. Extended polynomial dataset

A fourth order polynomial task has been investigated in previous studies (Butcher et al., 2010a, 2010b) which required a network to produce an output according to a fourth order polynomial and the previous input. This dataset is extended here to allow the order and delay of the polynomial to be varied. An increase in the order of the polynomial increases the non-linearity of the target output, while an increase in the delay increases the short-term memory required from a network. The data consists of one input which is a random number, chosen from a uniformly distributed range between -1 and $+1$, for which the network is then required to give one output according to the extended polynomial:

$$\mathbf{y}_{tgt}(t) = \sum_{i=0}^{p} \sum_{j=0}^{p-i} c_{ij} \mathbf{u}^i(t) \mathbf{u}^j(t-d) \quad \text{s.t. } i+j <= p \tag{11}$$

where c_{ij} is a random uniformly distributed coefficient drawn from the same range as the input data. This allowed the non-linearity and memory contained in the output to be varied, where p controlled the order of the polynomial and the non-linearity of the dataset and d controlled the memory required from the network. Increasing both of these parameters at the same time increases the difficulty of the task. While a more simple polynomial (or other type of dataset) could have been chosen to provide further interpretation of the dataset under investigation (for example, the shape of the polynomial presented here) the main aim of this study was to investigate the performance of the three aforementioned architectures when processing complex data which contains high amounts of non-linearity and short-term memory requirements. The performance of each architecture was calculated using the Normalised Root Mean Squared Error (NRMSE) which is defined as:

$$NRMSE = \sqrt{\frac{\langle \left(\hat{\mathbf{y}}(t) - \mathbf{y}_{tgt}(t)\right)^2 \rangle_{t_{tot}}}{\sigma^2_{\mathbf{y}_{tgt}}}} \tag{12}$$

where $\sigma^2_{\mathbf{y}_{tgt}}$ is the variance of the target output, $\langle\rangle_{t_{tot}}$ indicates the average over the whole time series t_{tot}, t_{tot} is the total number of patterns, $\mathbf{y}_{tgt}$ is the target output and $\hat{\mathbf{y}}(t)$ is the actual output from the network.

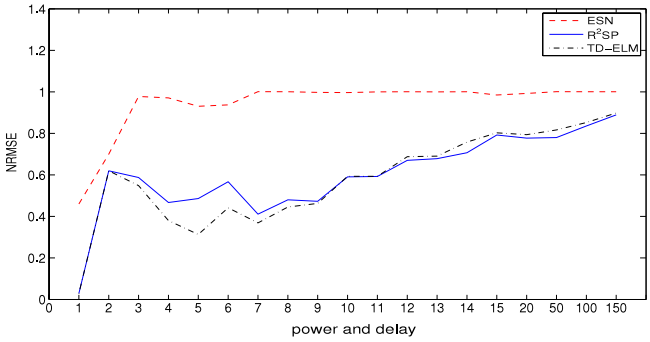


Fig. 8. The NRMSE error of an ESN (red), an R^{2}SP (blue) and TD-ELM (black) obtained when increasing the power and delay simultaneously for the extended polynomial task. Note that the scale along the X axis is non-linear beyond 15. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

7. Results

Three experiments were conducted by incrementing over the degree of non-linearity within the polynomial and the amount of memory required by the target output. The window size of the TD-ELM when applied to this dataset was set at the value of the delay plus one ($d + 1$), as this would provide the TD-ELM with sufficient memory of the current input and the previous input as defined by d. Different values of the leak rate were found to make little difference in performance.

7.0.1. Simultaneously increasing the non-linearity and short-term memory

Fig. 8 shows the performance of the three architectures when simultaneously increasing the polynomial power (i.e. non-linearity) and delay of the extended polynomial task. It can be seen that the R^{2}SP and TD-ELM networks outperform an ESN when applied to the extended polynomial task as the power and delay increase simultaneously. This result was confirmed as significant using the two-sided t-test (Rees, 2000) with a 95% confidence level. R^{2}SP significantly outperformed the TD-ELM architecture for power and delay values of 1 and 2, and for p and d values greater than 9. The TD-ELM significantly outperformed the R^{2}SP architecture for p and d values between 3 and 9 inclusive.

As the values of p and d increase, the ESN's error quickly approaches a value around unity indicating that the error of the network is of the same magnitude as the variance of the signal (the performance of the network is essentially guessing in this event). In contrast, the error of R^{2}SP and TD-ELM never reaches an NRSME of unity, and for the majority of p and d values their errors are approximately half that of the ESN. This remains the case up until the power and delay values are greater than nine, where the task becomes more difficult as both networks are required to recall inputs further into the past and produce a more non-linear output. As the values of p and d increase further the performance of the R^{2}SP and TD-ELM become gradually worse and can reasonably be expected to reach a similar level to the ESN when values of p and d exceed 150. As the number of coefficients in the target polynomial increases dramatically with each increment in p, p values greater than 150 were not investigated in this work.

Fig. 9 shows the NRMSE of an ESN and R^{2}SP when ranging over the input scale and spectral radius when p and d are equal to 1, 5, 15 and 100. Inspection of Fig. 9 reveals that the input scale which enabled best performance for the ESN is generally small, indicating that it prioritises MC over non-linear separation: larger input scale values cause an increase in error. In the first three sub-figures (where p and d are equal to 1, 5 and 15) the input scale of the ESN is generally low, which will aid its short-term memory of the input data. In the case when the power and delay are equal to 100, the input scale of the ESN which enables best performance becomes high, which will significantly decrease the ESN's MC. In comparison, the input scale to the best performing R^{2}SP-reservoir never has a value as high as that of the ESN, allowing the R^{2}SP-reservoir to possess a higher MC.

Inspection of the spectral radius for the ESN shows that the tuned value that enables best performance is again generally below unity, indicating that it is tuned for longer short-term memory over non-linearity. The spectral radius of the R^{2}SP-reservoir is also less than unity as shown in Fig. 9, which will aid its MC also. In order to analyse these results further and to investigate the aforementioned hypotheses regarding the dynamics of both architectures' reservoirs, the MC and deviation from linearity for both architectures using the input scale and spectral radius parameters that enabled best performance for each value of p and d are plotted below in Figs. 10 and 11. The data in these plots and Figs. 14, 15, 18 and 19 shown below show the average MC and deviation from linearity values of ten simulated networks for each architecture along with their standard deviations as well as smoothed data using a moving window of 11 datapoints to show the general trend as the parameters of the extended polynomial are varied.

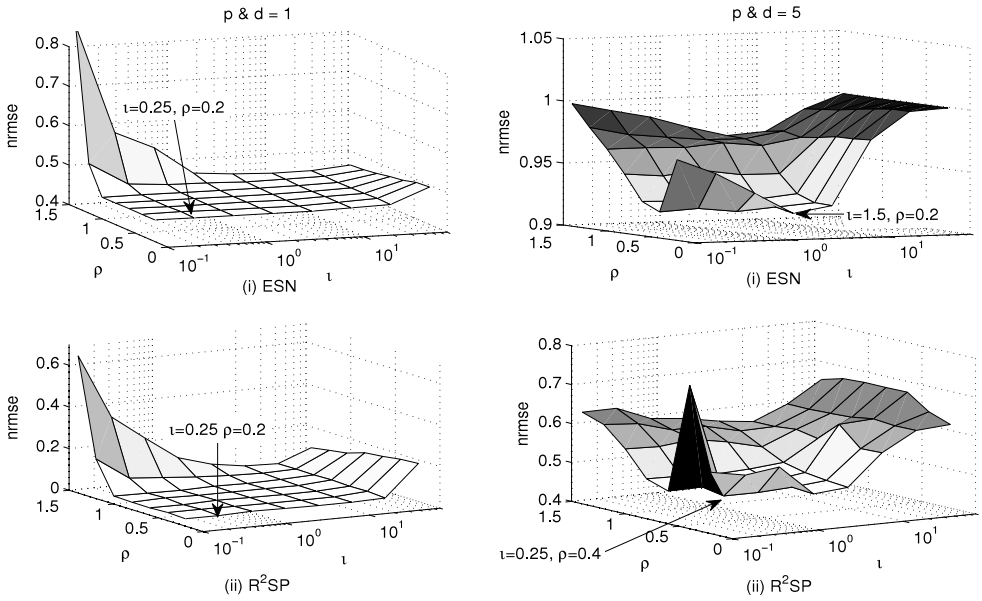
Fig. 10 shows the general trend of the MC of the ESN decreasing as the non-linearity and short-term memory requirements of the task increase, whereas the overall trend of the MC of the R^{2}SP-reservoir increases slightly as p and d increase. As a result of the increase in the input scale and spectral radius values of the ESN as p and d increase, the ESN's deviation from linearity also increases as is shown in Fig. 11. The decrease in MC and increase in deviation from linearity of the ESN is mainly due to large values of input scale (and/or spectral radius) when p and d increase: attributed to the high levels of non-linearity required by the target output.

7.0.2. Increasing the short-term memory

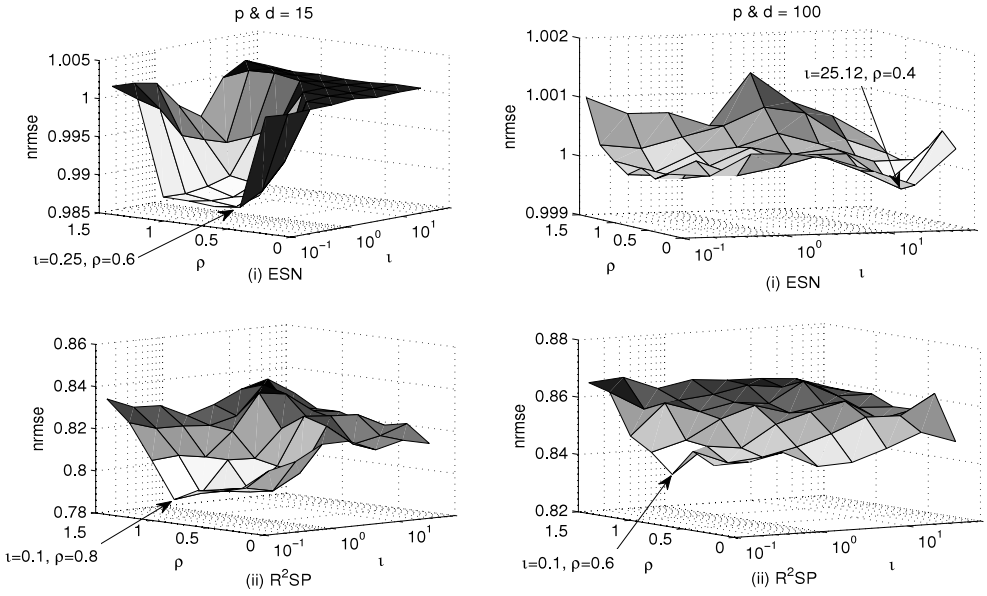
The performance of the three architectures was further compared by increasing the delay required by the task, d, whilst keeping the non-linearity of the task, p, equal to unity. This was to test the hypothesis that by having additional non-linear layers the R^{2}SP-reservoir has an increased MC as a result of its tuned parameters. For the TD-ELM, this also shows whether presenting the input using windows equal to $d + 1$ was sufficient to capture the required input data. The resulting errors of the architectures are plotted in Fig. 12 below.

As Fig. 12 shows, the TD-ELM and R^{2}SP approaches outperform an ESN over all iterations of delay (d). Significance testing using a two-sided t-test (p value $<5\%$) revealed that the R^{2}SP and TD-ELM's improved performance was indeed significant. The error of the TD-ELM and R^{2}SP are very similar up until delay values are greater than 20, after which the TD-ELM offers a significant improvement in performance. The better performance of the TD-ELM for larger values appears to be a result of presenting the input data via a moving window. This approach seems to be sufficient to capture the inputs required for the target output. In the case of the R^{2}SP, whose reservoir has a fading memory, inspection of its error indicates that its fading memory does not stretch as far as required by the polynomial for large values of d, resulting in increased errors.

Fig. 13 shows the test errors of the ESN and R^{2}SP when the delay is equal to 5, 15, 50 and 100 and the power is equal to 1 as a function of the input scale and spectral radius parameters. This figure shows that both architectures have a preference for a larger MC as shown by their relatively small input scales and spectral radii (mostly ≤ 1), where the spectral radius increases ($\rho \longrightarrow 1$) as



(a) Test errors of an ESN (i) and an R^{2}SP (ii) when the power and delay are equal to 1 (left) and 5 (right).



(b) Test errors of an ESN (i) and an R^{2}SP (ii) when the power and delay are equal to 15 (left) and 100 (right).

Fig. 9. Test NRMSE plots of the ESN (upper plots) and R^{2}SP (lower plots) when applied to the extended polynomial task ranging over the input scale (ι) and the spectral radius (ρ) with the combination of parameters that provided best performance shown. Note the different scales on the Z axis.

the delay increases. Larger values of input scale and spectral radius generally cause larger network errors, especially where d is equal to 5, 15 and 50. This behaviour is expected for both architectures, as both should be tuned towards an increasing MC as the delay of the target output is increased.

As before, the MC and non-linearities contained in the ESN and R^{2}SP architectures were calculated using the values of their respective input scale and spectral radius that gave best performance for each value of d, which are shown in Figs. 14 and 15.

Fig. 14 shows that the MC of both architectures is quite similar. Not surprising as the task required both architectures to have a high MC, especially as the delays increased. The increased MC of the R^{2}SP-reservoir may be a result of the small amount of non-linear separation being performed by $stat_1$ rather than the R^{2}SP-reservoir, a facility not available to the ESN. Fig. 15 shows that the deviation from linearity of both architectures follows a similar pattern to their MCs. The R^{2}SP has a higher deviation from linearity for the majority of the increments of d, with noticeable exceptions when d is equal to 100 and 150.

7.0.3. Increasing the polynomial power

As a final comparison for this dataset, the performance of the three architectures when increasing the non-linearity of the output, p, and leaving the delay, d, set at unity was investigated. This was conducted with the aim of analysing the amount of non-linear transformation performed by the R^{2}SP-reservoir: if the feedforward memoryless layers (in particular $stat_1$) performed the majority of the non-linear separation, then with highly non-linear data, its R^{2}SP-reservoir should still be tuned towards a longer MC. In the case of the ESN, its neurons will become more non-linear as required by the task. The instantaneous non-linearities

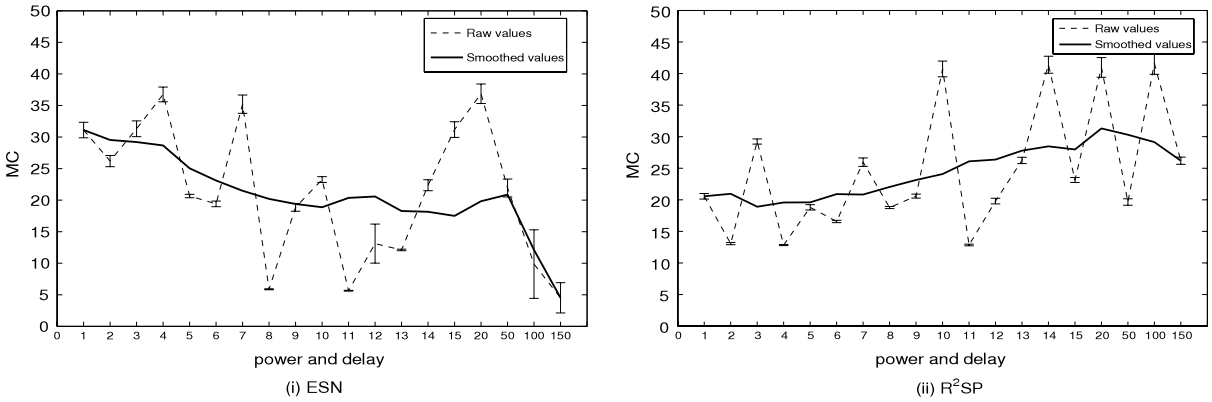


Fig. 10. The memory capacity (MC) of an ESN (left) and the R^{2}SP-reservoir (right) when increasing the power and delay for the extended polynomial task showing original values (dotted line) and their standard deviations as well as smoothed values which show the overall trend (solid line). Note that the power and delay increases are non-linear above values of 15, as shown on the X axis.

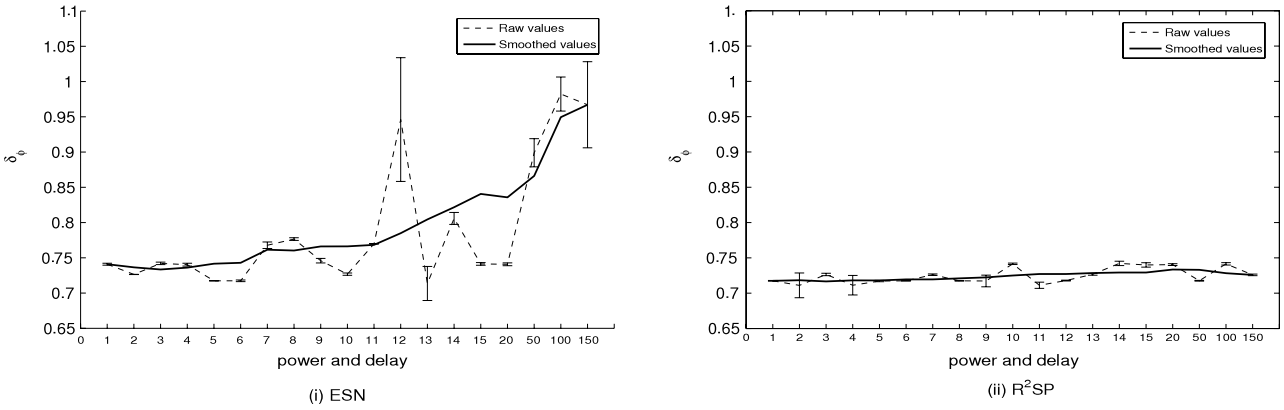


Fig. 11. The deviation from linearity (δ_ϕ) of an ESN (left) and the R^{2}SP-reservoir (right) when increasing the power and delay for the extended polynomial task showing original values (dotted line) and their standard deviations as well as smoothed values which show the overall trend (solid line). Note that the power and delay increases are non-linear above values of 15, as shown on the X axis.

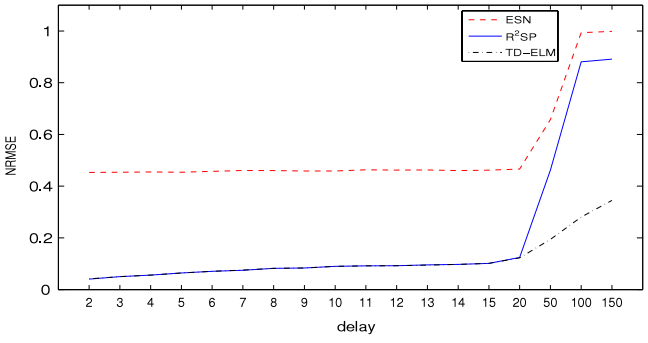


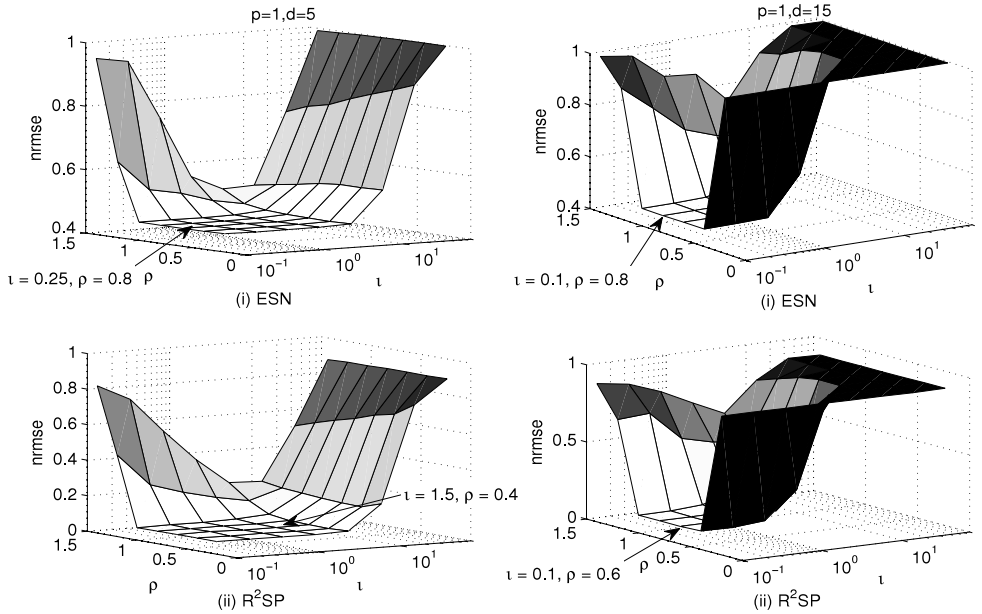
Fig. 12. The NRMSE of an ESN (red), an R^{2}SP (blue) and TD-ELM (black) when increasing the delay for the extended polynomial task. Note that the delay increases are non-linear above values of 15, as shown on the X axis. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

provided by the hidden layer of the TD-ELM should also enable good performance here.

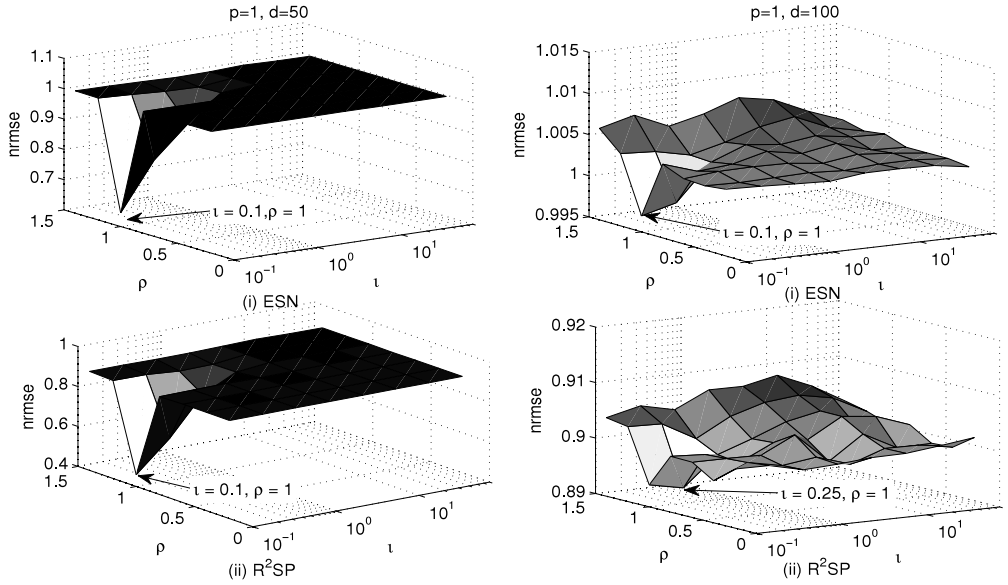
The errors of all three architectures when iterating over p are shown in Fig. 16 below. The TD-ELM and R^{2}SP approaches can be seen to offer superior performance compared to the ESN, which was found to be significant for each value of p and d using two-sided t-test with a 95% confidence interval. The TD-ELM significantly outperforms the R^{2}SP for all values of p. With the exception of when p is equal to 2, the NRSME of the ESN is around unity. The NRMSE of the TD-ELM and R^{2}SP networks never reaches unity, although their errors do approach the same level with high values of p. Fig. 17 shows the error of the ESN and R^{2}SP as a function of different values of input scale and spectral radius when p is equal to 5, 15, 50 and 100 and d is equal to 1.

Fig. 17 shows that the input scale which enables the best performance of the ESN is much higher than the input scale to the R^{2}SP-reservoir. For the ESN, this is expected: as the target output becomes more non-linear, a high input scale is optimal since it moves the working points of the reservoir neurons towards their non-linear regions in order to project the input data onto a high dimensional state space. The input scale to the R^{2}SP-reservoir on the other hand never becomes as high as the input scale of the ESN, confirming that the R^{2}SP-reservoir is required to perform less non-linear transformation as this is mainly taken care of by $stat_1$.

The spectral radius values which give best performance for the ESN are also high in some iterations of p as shown in Fig. 17. Conversely, the spectral radius values of the R^{2}SP-reservoir stay below unity for all iterations over p. While a spectral radius around unity often gives the highest MC, a high MC for this task is not required as only one time step into the past needs to be recalled. This may explain the many small values of spectral radius for the R^{2}SP-reservoir around 0.2 and 0.4. As a result of the high input scale to the ESN, it is expected that its MC will be low, whereas its deviation from linearity will be high. The opposite is expected for the R^{2}SP-reservoir. The values of both of these measures are plotted in Figs. 18 and 19.



(a) Test errors of an ESN (i) and an R^2SP (ii) when the power is equal to unity and the delay is equal to 5 (left) and 15 (right).



(b) Test errors of an ESN (i) and an R^2SP (ii) when the power is equal to unity and the delay is equal to 50 (left) and 100 (right).

Fig. 13. Test NRMSE plots of the ESN (upper plots) and R^2SP (lower plots) when applied to the extended polynomial task ranging over the input scale and the spectral radius with the combination of parameters that provided best performance shown. Note the different scales on the Z axis.

As Fig. 18 shows, the MC of the R^2SP-reservoir is much higher than that of the ESN. Fig. 19 shows that the ESN has a higher deviation from linearity than the R^2SP-reservoir. These two observations are mainly caused by the high input scale of the ESN that enabled best performance when increasing the non-linearity of the dataset. When increasing the non-linearity of the dataset the non-linear transformation of the input is mainly performed by $stat_1$ of the R^2SP architecture, which enables its reservoir to be tuned towards a longer MC.

8. Discussion

The results above for the extended polynomial task showed that the improvement in performance offered by the R^2SP and TD-ELM can be credited to their separation of non-linear mapping and short-term memory: the R^2SP through the combination of two memoryless feedforward layers and an ESN reservoir respectively; and the TD-ELM through the use of a memoryless hidden layer and a moving window. Analysing the R^2SP through the variation of the parameters that control the amount of non-linear separation and memory required by the task, it was shown that the R^2SP-reservoir is tuned towards maximising its MC, with minimal non-linear mapping as this is mainly performed by $stat_1$. Even in the situation where little memory was required by the task, it was shown that the R^2SP-reservoir was still tuned towards a longer MC, as a result of the non-linear separation that was mostly performed by $stat_1$. The analysis of the amount of non-linearity and MC of the R^2SP-reservoir compared to the ESN when applied to the extended polynomial task further confirmed this.

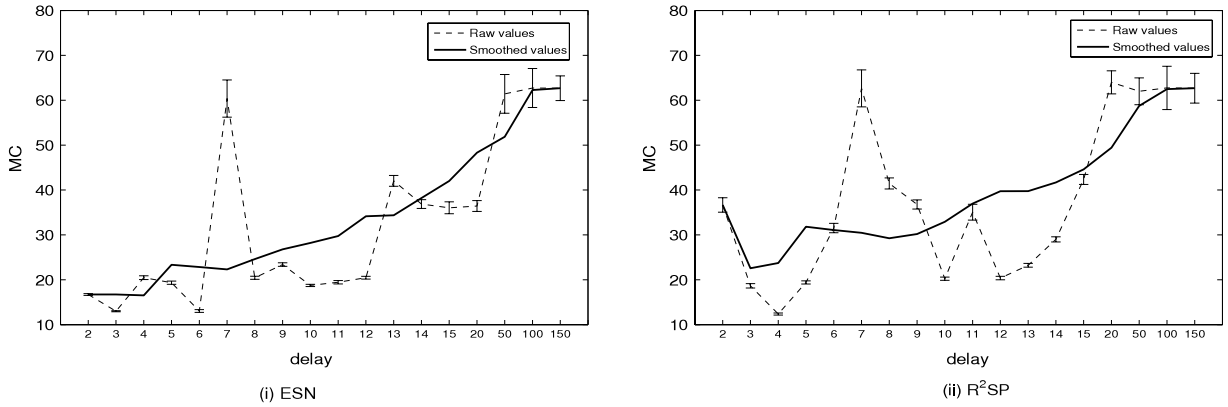


Fig. 14. The memory capacity (MC) of an ESN (left) and the R^{2}SP-reservoir (right) when increasing the delay for the extended polynomial task showing original values (dotted line) and their standard deviations as well as smoothed values which show the overall trend (solid line). Note that the power and delay increases are non-linear above values of 15, as shown on the X axis.

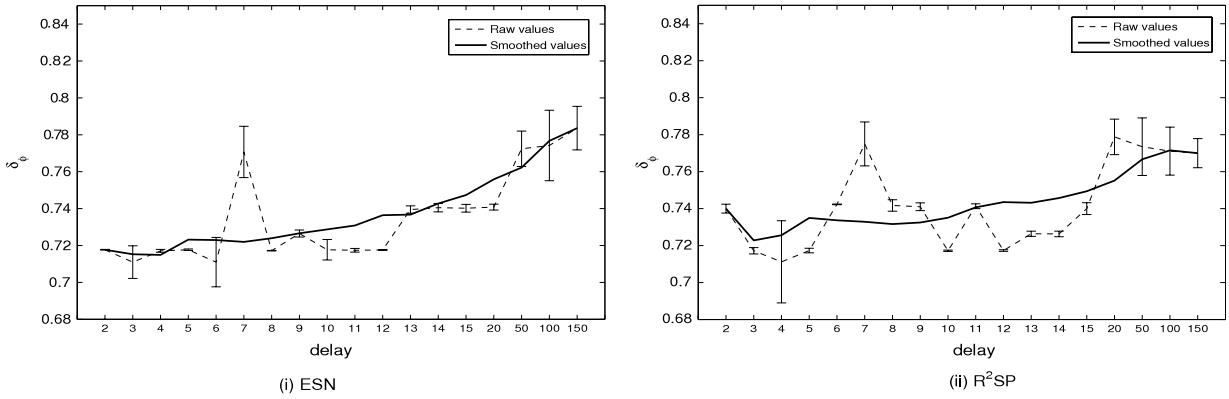


Fig. 15. The deviation from linearity (δ_ϕ) of an ESN (left) and the R^{2}SP-reservoir (right) when increasing the delay for the extended polynomial task showing original values (dotted line) and their standard deviations as well as smoothed values which show the overall trend (solid line). Note that the power and delay increases are non-linear above values of 15, as shown on the X axis.

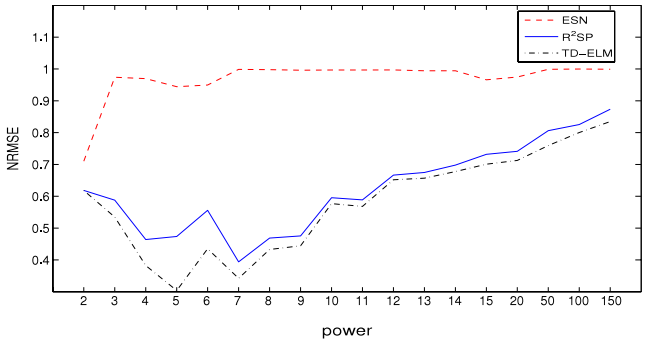


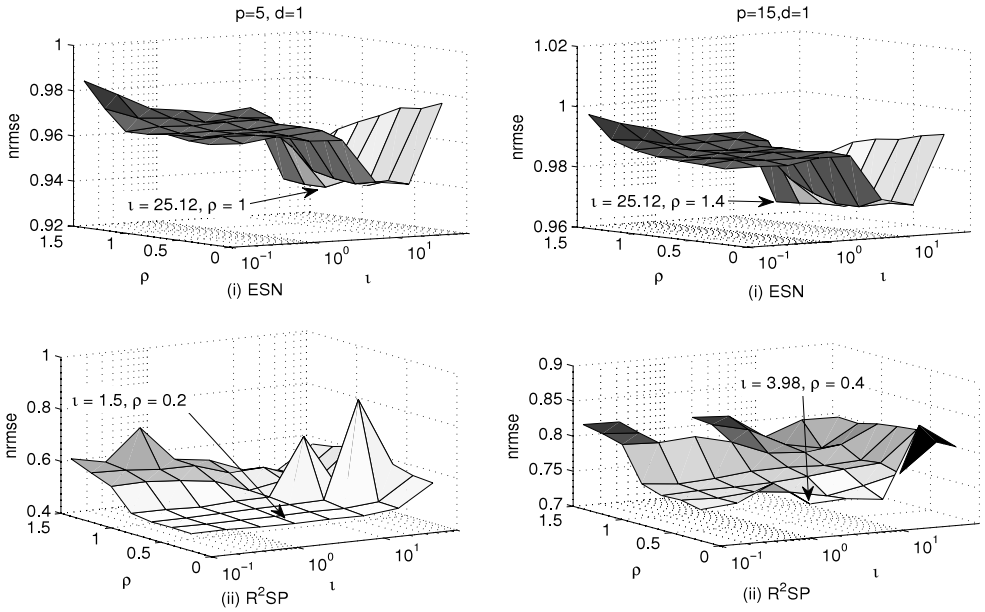
Fig. 16. The test NRMSE of the ESN (red), the R^{2}SP (blue) and a TD-ELM (black) when increasing the power for the extended polynomial task. Note the non-linear increase in the value of p (X axis). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

An interesting observation can be made by inspection of the errors of the three architectures for each experiment when the values of p and d are at their lowest. The R^{2}SP and TD-ELM networks have a much smaller error in comparison to the ESN which, in most cases, is considerably higher. It is conjectured that the instantaneous reaction of $stat_1$'s neurons of the R^{2}SP and the hidden layer neurons of the TD-ELM to a change in input are the reason for this improved performance. In contrast, an ESN's reservoir is slow to react to a change in input due to its recurrent connections. While this is a vital characteristic of an ESN (acting as a short-term memory) in some cases where fast changing input data is present this may hinder its performance.

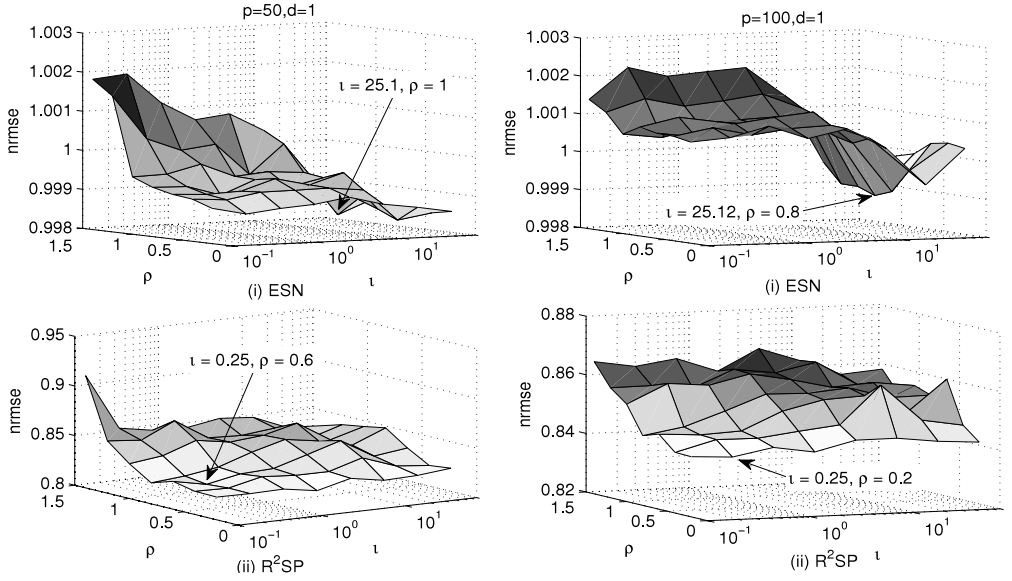
9. Conclusions

Two different architectures which separate the task of transforming input data onto a non-linear state space and possessing a short-term memory of the input data have been presented and investigated here. The first, a Time Delay Extreme Learning Machine (TD-ELM), consists of a memoryless feedforward ANN, where only the output weights are trained, coupled with a moving window of the input data giving it a memory as long as the length of the window. The second, a new addition to Reservoir Computing (RC), Reservoir with Random Static Projections (R^{2}SP), consists of an Echo State Network (ESN) with two added memoryless feedforward layers that provide an instant non-linear transformation of the input and the reservoir states. As with the ELM approach, only the output weights are trained in RC architectures.

Using a novel task, for which the amount of memory and non-linear transformation could be varied, it was shown that the R^{2}SP and TD-ELM offered a significant improvement in performance in comparison to an ESN as a result of containing higher amounts of memory, while at the same time, performing non-linear transformations of the input data. Using reservoir metrics this was confirmed for the R^{2}SP. The TD-ELM was generally found to offer a significant improvement in performance when compared to the



(a) Test errors of an ESN (i) and an R^2SP (ii) when the delay is equal to unity and power is equal to 5 (left) and 15 (right).



(b) Test errors of an ESN (i) and an R^2SP (ii) when the delay is equal to unity and the power is equal to 50 (left) and 100 (right).

Fig. 17. Test NRMSE plots of the ESN (upper plots) and R^2SP (lower plots) when applied to the extended polynomial task ranging over the input scale and the spectral radius with the combination of parameters that provided best performance shown. Note the different scales on the Z axis.

R^2SP (the R^2SP offered a significant improvement in performance for some variations over the non-linearity and short-term memory of the extended polynomial task).

The overall superior performance of the TD-ELM compared to the two RC approaches could be attributable to the dataset studied in this work, where a network with a hard limited memory appears to offer better performance than a network with a fading memory (as is the case with the ESN and R^2SP approaches). This can be attributed to the nature of the extended polynomial task, where the memory required from a network includes the current time step plus the delay of the target data ($d + 1$). Therefore a hard-limited, rather than fading, memory is sufficient to remember previous inputs as defined by the delay. Further work is required to validate this point, as a reservoir based approach (such as an ESN or R^2SP) is expected to outperform a TD-ELM when applied to data that requires a long fading memory: a problematic case for architectures that use fixed windows for the input since large window sizes will lead to greater training times.

The analysis of the performance of the R^2SP and TD-ELM approaches when applied to other challenging datasets with high amounts of non-linearity and long short-term memory would be an interesting focus for future work. As the extended polynomial dataset is a novel dataset, investigations into performance of other machine learning approaches when applied to this dataset are desirable.

In this work it would appear that the use of a reservoir is ill advised for datasets which do not require a fading memory of inputs. A simple feedforward approach for which only the output weights are trained and the input is presented as a window over time offers the best performance. Further work into the dynamics of both architectures and the characteristics of input datasets is needed to further validate this point.

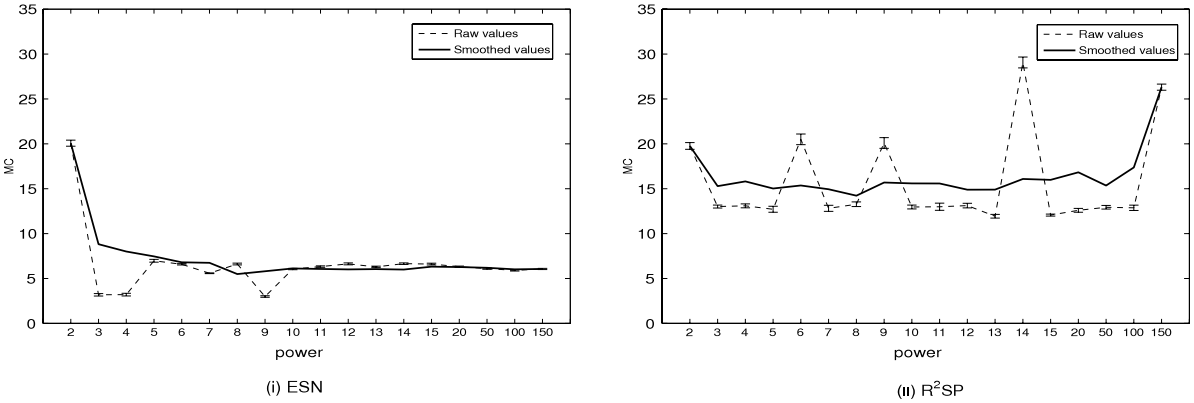


Fig. 18. The memory capacity (MC) of an ESN (left) and the R^2SP-reservoir (right) when increasing the power for the extended polynomial task showing original values (dotted line) and their standard deviations as well as smoothed values which show the overall trend (solid line). Note that the power and delay increases are non-linear above values of 15, as shown on the *X* axis.

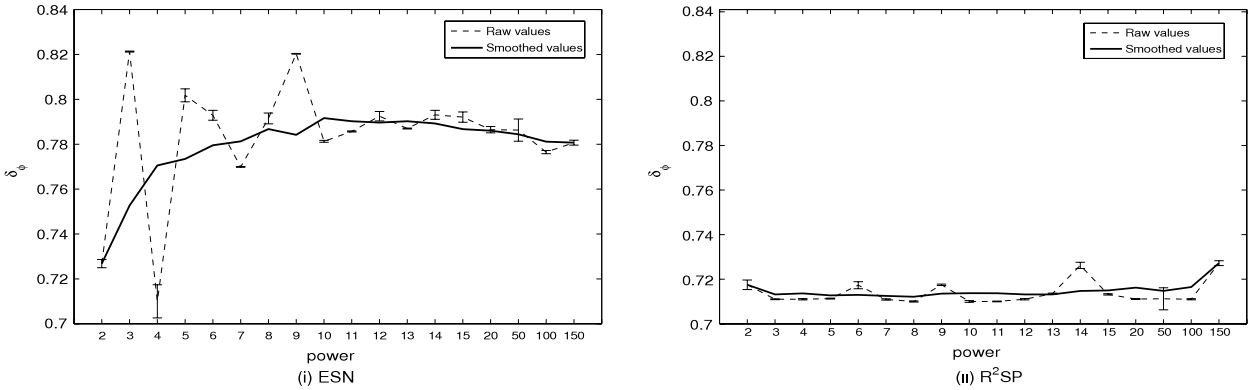


Fig. 19. The deviation from linearity (δ_ϕ) of an ESN (left) and the R^2SP-reservoir (right) when increasing the power for the extended polynomial task showing original values (dotted line) and their standard deviations as well as smoothed values which show the overall trend (solid line). Note that the power and delay increases are non-linear above values of 15, as shown on the *X* axis.

Acknowledgements

The UK EPSRC and SciSite Ltd are gratefully acknowledged for funding this research. The research team from the Department of Electronics and Information Systems from Ghent University are also gratefully acknowledged for their kind and extremely useful feedback. Simulations were conducted using the high-performance computer CUDA at EPSAM Institute at Keele University (UK).

References

Antonelo, E., Schrauwen, B., & Stroobandt, D. (2008). Mobile robot control in the road sign problem using reservoir computing networks. In *Proceedings of the IEEE international conference on robotics and automation* (pp. 911–916).

Büsing, L., Schrauwen, B., & Legenstein, R. (2010). Connectivity, dynamics and memory in reservoir computing with binary and analog neurons. *Neural Computation*, *22*(5), 1272–1311.

Butcher, J.B. (2012). Reservoir computing with high non-linear separation and long-term memory for time-series data analysis, Ph.D. Thesis, Institute for the Environment, Physical Sciences and Applied Mathematics (EPSAM), Keele University, Staffordshire, ST5 5BG, UK.

Butcher, J.B., Verstraeten, D., Schrauwen, B., Day, C.R., & Haycock, P.W. (2010a). Extending reservoir computing with random static projections: a hybrid between OP-ELM and RC. In *European symposium on artificial neural networks*, ESANN (pp. 303–308).

Butcher, J.B., Verstraeten, D., Schrauwen, B., Day, C.R., & Haycock, P.W. (2010b). Pruning reservoirs with random static projections. In *IEEE International workshop on machine learning for signal processing*, MLSP (pp. 250–255). http://dx.doi.org/10.1109/MLSP.2010.5589251.

Butcher, J.B., Verstraeten, D., Schrauwen, B., Day, C.R., & Haycock, P.W. (2012). Defect detection in reinforced concrete using random neural architectures. In *Computer aided civil and infrastructure engineering* (submitted for publication).

Chen, F., & Ou, T. (2011). Sales forecasting system based on Gray extreme learning machine with Taguchi method in retail industry. *Expert Systems with Applications*, *38*(3), 1336–1345.

Doddington, G., & Schalk, T. (1981). Speech recognition: turning theory to practice. *IEEE Spectrum*, *18*(9), 26–32.

Gallicchio, C., & Micheli, A. (2011). Architectural and Markovian factors of echo state networks. *Neural Networks*, *24*(5), 440–456.

Holzmann, G. (2009). Reservoir computing: a powerful black-box framework for nonlinear audio processing. In *12th international conference on digital audio effects*, DAFx-09. Como, Italy.

Huang, G., & Wang, D. (2011). Advances in extreme learning machines (elm2010). *Neurocomputing*, *74*(16), 2411–2412.

Huang, G., Zhu, Q., & Siew, C. (2006). Extreme learning machines: theory and applications. *Neurocomputing*, *70*(1-3), 489–501.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks, Tech. rep., German National Research Institute for Computer Science.

Jaeger, H. (2002a). Adaptive nonlinear system identification with echo state networks. In *Neural information processing systems*, NIPS.

Jaeger, H. (2002b). A tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the "echo state network" approach, Tech. Rep. GMD Report 159, German National Research Center for Information Technology.

Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science*, *304*(5667), 78–80.

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

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

Montgomery, D., & Peck, E. (1982). *Probability and mathematical statistics*, *Introduction to linear regression analysis*. USA: Wiley.

Moore, E. (1920). On the reciprocal of the general algebraic matrix. *Bulletin of The American Mathematical Society*, *26*, 394–395.

Penrose, R. (1955). A generalised inverse for matrices. In *Cambridge philosophical society, Vol. 51* (pp. 406–413)).
Rees, D. (2000). *Essential statistics* (4th Ed.). London, UK: Chapman and Hall.
Singh, R., & Balasundaram, S. (2007). Application of extreme learning machine method for time series analysis. *International Journal of Intelligent Technology*, *2*(4), 256–262.
van Heeswijk, M., Miche, Y., Lindh-Knuutila, T., Hilbers, P., Honkela, T., Oja, E., et al. (2009). Adaptive ensemble models of extreme learning machines for time series prediction. In *Proceedings of the 19th international conference on artificial neural networks*, ICANN. (pp. 305–314). Berlin, Heidelberg: Springer-Verlag.
Verstraeten, D. (2009). Reservoir computing: computation with dynamical systems, Ph.D. Thesis, Electronics and Information Systems department, University of Ghent, Belgium.
Verstraeten, D., Dambre, J., Dutoit, X., & Schrauwen, B. (2010). Memory versus non-linearity in reservoirs. In *International joint conference on neural networks*, IJCNN (pp. 2669–2676).
Verstraeten, D., Schrauwen, B., D'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Networks*, *20*(3), 391–403.
Verstraeten, D., Schrauwen, B., & Stroobandt, D. (2006). Reservoir-based techniques for speech recognition. In *Proceedings of the international joint conference on neural networks*, IJCNN (pp. 1050–1053).
Verstraeten, D., Schrauwen, B., Stroobandt, D., & Van Campenhout, J. (2005). Isolated word recognition with a liquid state machine: a case study. *Information Processing Letters*, *95*(6), 521–528.