White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of York.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/241069/

Version: Accepted Version

Proceedings Paper:
Griffin, David, Stovold, James, O'Keefe, Simon et al. (2026) Evaluating ESNs Against Lagged Input Regression Computation. In: Formenti, Enrico and Manzoni, Luca, (eds.) Unconventional Computation and Natural Computation - 22nd International Conference, UCNC 2025, Proceedings. 22nd International Conference on Unconventional Computation and Natural Computation, UCNC 2025, 01-05 Sep 2025 Lecture Notes in Computer Science. Springer Science and Business Media Deutschland GmbH, FRA, pp. 262-276.

https://doi.org/10.1007/978-3-032-15641-9_18

Reuse
This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here:
https://creativecommons.org/licenses/

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

UNIVERSITY OF LEEDS
University of Sheffield
UNIVERSITY of York

Evaluating ESNs against Lagged Input Regression Computation

David Griffin[1 [0000−0002−4077−0005]], James Stovold[2 [0000−0002−0708−2630]], Simon O'Keefe[1 [0000−0001−5957−2474]], and Susan Stepney[1 [0000−0003−3146−5401]]

[1] Department of Computer Science, University of York, UK
{david.griffin,simon.okeefe,susan.stepney}@york.ac.uk
[2] Lancaster University Leipzig, Germany j.stovold@lancaster.ac.uk

Abstract. Echo State Networks (ESNs) are stated in literature to use a random structure to project an input sequence into a higher dimensional space where the input becomes linearly separable. However, the linear mathematics used for this projection are incapable of increasing the dimensionality of the input, and the commonly used tanh() activation function tends not to produce much nonlinearity. Therefore, any increase in dimensionality is due to the echoes of the ESN. We introduce Lagged Input Regression Computation to investigate what types of ESN can be replaced with simpler non-randomised structures. We show that tanh()-based ESNs behave as simple linear memory systems, whereas LeakyReLU provides a more effective non-linearity. We also show that the use of certain orthogonal polynomials in defining nonlinear memory capacity benchmarks gives a misleading impression of nonlinearity, due to the relevant high order polynomials nevertheless containing a linear term.

Keywords: Reservoir Computing · Echo State Networks · Memory Systems.

1 Introduction

Artificial Neural Networks (ANNs) [4] have become a popular tool: they are widely used in Computer Science, and have wide ranging applications from Pattern Recognition to Large Language Models. Of the various types of ANNs, Recurrent Neural Networks (RNNs) [2] arguably have the most complexity, allowing arbitrary connections between simulated neurons, rather than the more structured approach seen in Feed Forward Neural Networks [4]. RNNs have the primary advantage that they capture data within their recurrent connections, making them suitable for temporal data or pattern recognition [2]. However, this also comes with a disadvantage: they are more complex to train. For example, to apply the backpropagation training techniques, it is necessary to unravel the recurrent connections by applying backpropagation through time [17].

In Section 3 we examine the various parts of the intuition behind ESNs and show that, as commonly used in literature, the random recurrence of ESNs may

not be as interesting as previously thought. In particular, we compare common configurations of ESNs using the tanh() activation function with a completely linear lagged input system. The result shows both that many ESNs do not exhibit particularly nonlinear behaviour, as well as highlighting the potential power of linear regression techniques on simple memory.

In Section 4, we introduce Lagged Input Regression Computation (LIRC), which uses simple linear memory with a regression layer. We evaluate ESNs against LIRC to show that tanh() ESNs can be approximated with a purely linear system. We also show that the alternative LeakyReLU activation function does not have the same behaviour, and is only approximable by a LIRC system under specific, favourable conditions. Given that other authors [8] have shown that ESNs exhibit nonlinear memory, we revisit this definition of nonlinear memory and show that linear effects unintentionally dominate this measure.

2 Related Work

An ESN [14] is a randomly connected, sparse, recurrent neural network with a fixed activation function. In the literature, sigmoidal functions are typically recommended, with the tanh() function normally being assumed[14]. Although other authors have suggested using a rectified linear unit (ReLU) [12]. Input is supplied to the ESN nodes through a randomly generated input weight matrix. The internal connections of an ESN are given by a randomly generated matrix that specifies the connections and their weights. When generating the weight matrix, typically both the sparsity and spectral radius (the maximum of the absolute values of the eigenvalues of the weight matrix) are specified. It is desirable for ESNs to have the *echo state property*, which can be obtained with a sigmoidal activation function and a spectral radius < 1 [6]. ESNs can also have feedback, which is typically specified by a third randomly generated matrix. For simplicity, here we consider only the case where feedback is not present.

A simple, non-feedback ESN is typically defined by the equation:

$$\mathbf{x}(n+1) = f(\mathbf{W}\mathbf{x}(n) + \mathbf{W}^{in}\mathbf{u}(n)) \tag{1}$$

where $\mathbf{W}$ is the random weight matrix, $\mathbf{W}^{in}$ is the random input weight matrix, $\mathbf{x}(n)$ is the state of the ESN at time n, $\mathbf{u}(n)$ is the input at time n, and f is the activation function. While this equation has no output feedback, the ESN may have internal feedback through the graph described by $\mathbf{W}$.

To extract output, the values stored in the nodes of the ESN are passed through a trained linear output layer.

$$\mathbf{y}(n+1) = \mathbf{W}^{out}\mathbf{x}(n) \tag{2}$$

where $\mathbf{W}^{out}$ is the trained output weight matrix, and $\mathbf{y}(n)$ is the output of the ESN at time n. Typically, the output layer is trained through either linear regression [23] or ridge regression [13] methods, which allow the output layer to be trained in a relatively trivial amount of time.

ESNs have found use in predicting or classifying temporal data [14], where the inherent recurrence of the ESN is most useful. Wringe et al. [22] present a review of benchmarks used in the broader Reservoir Computing literature; the majority of benchmarks used are indeed temporal, such as NARMA (Nonlinear AutoRegressive Moving Average) [21] or Mackey–Glass [16].

Recent work has called into question the way in which ESNs are generated and used. Gauthier et al. [9] found that many of the tasks ESNs excel could also be accomplished by a nonlinear vector autoregression (NVAR) machine [5]. NVAR machines make predictions based on (a) a number of previous timesteps and (b) a number of nonlinear transformations. Those authors were not able to explain which nonlinearites are most useful to which tasks, other than observing that polynomials seem to have broad applicability.

Ma et al. [15] continued that work, introducing a non-random RC that cast inputs into a substantially higher dimensional space, and using a nonlinear readout instead of an activation function. It resulted in an increase in model accuracy, albeit with a larger state space than an NVAR machine. That work did not identify the cause of the result as being either the changes to the model readout or the increase in dimensionality.

One pattern in prior research is a belief that the projection an ESN's input into a higher dimensional space leads to linear separability. If the dimensionality of the ESN is increased in some way, then this leads to an increase in separability. This appears to be similar to a well known result in the study of ANNs, Cover's Theorem [7]. However, Cover's Theorem specifically utilises *nonlinear* projections to achieve linear separability of nonlinear properties. As we show in this paper, the linearity of the input projection $\mathbf{W}^{in}$, and the subsequent use of the sigmoid activation function in its linear regime render the overall behaviour of many ESNs to be predominantly linear.

3 Dissecting an ESN

Using the non-feedback ESN of Eqn.1, we investigate how they achieve the linear separability property. The ESN equation has three main components: the activation function f, the projection of input into a higher dimensional space given by $\mathbf{W}^{in}$, and the movement of information within the ESN structure given by $\mathbf{W}$. We examine these three components in turn.

3.1 Activation Function

The activation function f is commonly defined to be a sigmoidal function such as tanh() or the logistic function $1/(1-e^x)$, see Figure 1. tanh() is close to the identity function in the range $[-1, 1]$, and virtually identical to it in the range $[-0.5, 0.5]$. The Maclaurin series for $\tanh(x)$ [1] is $x - x^3/3 + O(x^5)$, and so in the latter range $\frac{|\tanh(x)-x|}{|x|} \approx \frac{0.5^3}{3 \cdot 0.5} = 0.083$, or a less than 10% difference. Shrinking the range further reduces this magnitude further.

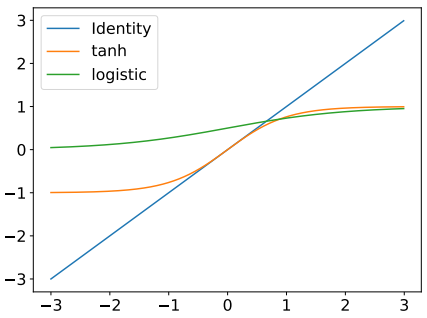

Fig. 1: Sigmoidal activation functions

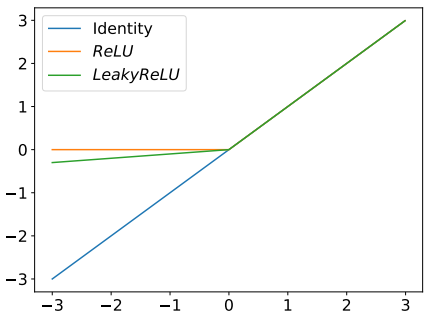

Fig. 2: Rectifier activation functions

Within an echo state network, tanh() activation provides two properties: a nonlinearity, and the ability to constrain values in the range $[-1, 1]$. The latter is seen as an important reminder of physicality; if an ESN approximates a physical system, then it cannot encode infinity. However, the similarity of tanh() to the identity within the region where input is typically normalised may limit its nonlinearity contribution. This combination of constraining to $[-1, 1]$ and the near-linearity in this region seems a major drawback to the use of tanh() as an activation function in the context of an ESN.

Does the internal dynamics of the ESN counter this, and move the value into the nonlinear region of tanh()? We argue no. Consider the internal state $\mathbf{x}$ of inputs to a given node. Each component is multiplied by a random weight as given by the internal weight matrix $\mathbf{W}$. If these weights have a mean of 0 (such as for uniform weights, or Gaussian weights), then approximately equal numbers of these weights have a positive and negative sign. The internal state values stored in an ESN are all within the image of the activation function, $[-1, 1]$. In literature, the weights of $\mathbf{W}$ are typically drawn from the uniform distribution of $[-1, 1]$ and then scaled by ρ, the spectral radius which is typically < 1, resulting in weights in the interval $[-\rho, \rho]$.

Then the sum of the weighted inputs to a node is also likely to be in the range $[-\rho, \rho]$. This is because the most likely outcome is for roughly an equal number of input weights to be positive and negative, resulting in the expected value of the sum to be close to 0. This can be seen empirically in Figure 3, which shows the distribution of values over time evolves in an example ESN. The state of the ESN is initialised to be uniform random $[-1, 1]$ at time 0. By time 8, the distribution of values has converged to a normal distribution. This experiment has been repeated with a variety of different parameterisations, and provided that the ESN is sufficiently large, this convergence is always seen.

As the number of input connections increases (increasing number of nodes), we expect the central limit theorem to further compress the range of likely values towards 0. While one could construct parameters to maximise the probability of getting a positive or negative bias in the weights to a node – for example, a

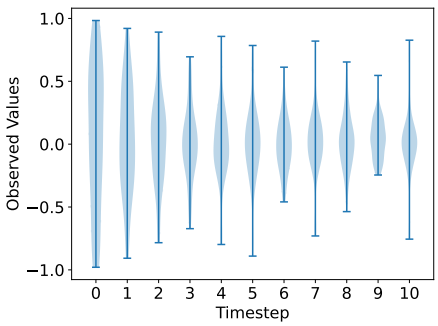

Fig. 3: Distribution of values over time in a 100-node ESN with a spectral radius of 0.8 and connectivity of 0.1 with a tanh() activation function

small value of N, a higher sparsity in W, or a higher spectral radius – reducing the number of values summed *also* reduces the probability of getting an extreme value. And most values in the ESN will still be in the linear region, as this remains the most likely outcome.

An effective example of the apparent ineffectiveness of tanh() as a nonlinearity is to apply the power method [11] to an ESN. The power method iterativly calculates the maximum eigenvalue of a matrix using:

$$\lambda_1 = \lim_{k\to\infty} \frac{||\mathbf{A}^k\mathbf{b}||}{||\mathbf{A}^{k-1}\mathbf{b}||} \tag{3}$$

where $\mathbf{b}$ is an arbitrarily chosen initial input vector, $\mathbf{A}$ is a matrix and $||.||$ is a vector norm. Provided that $\mathbf{b}$ is nonzero in the direction of the associated eigenvector and the maximum eigenvalue is strictly greater in magnitude than all other eigenvalues, this sequence converges on the maximum eigenvalue. If not (for example, in the case of complex conjugate eigenvalues), it oscillates between the greatest eigenvalues.

Using a highly restricted case of ESNs where we can set $\mathbf{u}(0) = \mathbf{b}, \mathbf{u}(n+1) = \mathbf{x}(t)$ with $\mathbf{W}^{in} = \mathbf{1}$ the identity, then $\mathbf{x}(n) = (tanh \otimes \mathbf{W})^n(\mathbf{b})$. If $\tanh \otimes \mathbf{W} \approx \mathbf{W}$, then $\mathbf{x}(n) \approx \mathbf{W}^n(\mathbf{b})$, and the power iteration is applicable. Conversely, if $\tanh \otimes \mathbf{W} \not\approx \mathbf{W}$, then the power iteration would not be applicable. In practice, $\tanh \otimes \mathbf{W} \approx \mathbf{W}$ if the spectral radius of $\mathbf{W}$ lies within the mostly linear region of tanh() between $[-1, 1]$. Experimentally, this approach has been applied to $10,000$ ESNs with unique maximum eigenvalues, with the size of the ESN varying between $[10, 100]$, spectral radius between $[0.1, 0.99]$ and connectivity between $[0.05, 1.0]$, and in every case the power method converged.

Similar arguments as presented on tanh() can also be applied to the standard logistic function, Figure 1. While the logistic function is mostly linear in the region $[-1, 1]$ it is not close to the identity. However, as the image of the standard logistic function is $[0, 1]$, repeated applications once again keep it within a mostly linear section. If a linear transformation is applied to map the standard logistic

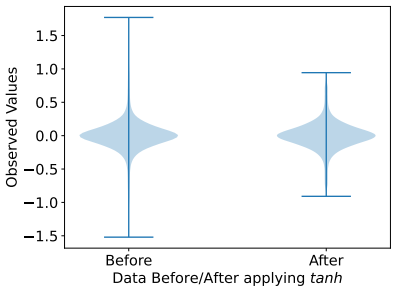

Fig. 4: Effect of tanh() activation function on values in an ESN

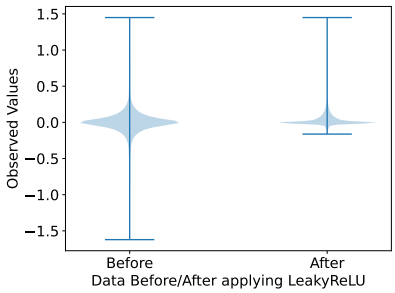

Fig. 5: Effect of LeakyReLU activation function on values in an ESN

sigmoids values back to the range $[-1, 1]$, then the previously mentioned suite of power method tests passes. While this transformation may seem to be adding an additional step, an ESN includes a linear readout layer, which can incorporate this transformation.

The Rectified Linear Unit (ReLU) is an alternative activation function (Figure 2). ReLU maps all negative values to 0, and all positive values to themselves. This mapping of negative values to 0 causes any information represented by those values to be lost. Again, if weights are distributed around 0, one may expect that the sum of values is also distributed around 0. Absent any bias, an ESN with the ReLU activation function may delete around 50% of the information it contains with every timestep, and the remaining 50% is strictly linear. This property makes it very difficult to apply arguments on how it behaves, and typically the power method will not converge if the dominant eigenvector has a negative value. Unlike with the standard logistic sigmoid, due to ReLU deleting information there is no linear transformation that can recover from this. The range of ReLU not being finite is not relevant for the lack of convergence, as the only requirement is that ReLU is mostly linear in its image.

The LeakyReLU activation function is also somewhat common. Rather than mapping values less than 0 to 0, LeakyReLU instead applies a small multiplier, which means that information is no longer being deleted. For LeakyReLU, the image of the function, $Im(\text{LeakyReLU})$, is not confined to a region where the function is strictly linear across the entire range. Instead, LeakyReLU is strictly linear across exactly half of its image. This means that iterative methods are still likely to succeed, as LeakyReLU is a mostly linear invertible function. However, again, there is no linear transformation that can undo LeakyReLU over its entire range. This results in the power method converging for LeakyReLU, provided that the spectral radius is not close to 0.

An experimental evaluation of the activation functions can be performed by plotting the distribution of values in an ESN before and after an activation function is applied. This is shown in Figures 4 and 5, for a 100 node ESN, although these results are repeatable with many sizes or parameterisations. As

in Figure 3, we see values consistent with a normal distribution. For LeakyReLU the spread of the distribution is lower due to it compressing negative values, thus reducing the average magnitude of values within the ESN. In Figure 4, we see that values outside of the range $[-1, 1]$ are compressed, but otherwise tanh() does not substantially change the distribution, with the new values still being normally distributed. This is also true of the logistic function, although the distribution of values afterwards is shifted to around 0.5. Figure 5 shows LeakyReLU, and in this case the distribution is substantially altered below 0, although in a manner that can be recovered with a linear transformation of the values below 0.

3.2 Linear Projections into Higher Dimensions

Having made an argument that standard sigmoid activation functions may not be sufficiently nonlinear, we move onto the next portion of the ESN equation: the input projection.

In order to examine the claim that an ESN projects input data into a higher dimensional space where it becomes linearly separable, we examine $\mathbf{W}^{in}$, which implements the initial projection.

Recall some properties of functions. A function $f : X \to Y$ maps all points in X onto some set of points in Y. If there is no $x_1 \neq x_2 \in X$ such that $f(x_1) = f(x_2)$, then f is *injective*. If for all $y \in Y$ there exists $x \in X$ such that $f(x) = y$, then f is *surjective*. If f is both injective and surjective, then f is bijective.

In order to project into a higher dimension, we need a function that maps $\mathbb{R}^m \to \mathbb{R}^n$, where $m < n$. There are bijective functions which do this; for example, consider the function $C : \mathbb{R} \to \mathbb{R}^2$ defined by:

$$\begin{aligned} C(x) = (x_1, x_2) \text{ where} \\ x_1 \text{ comprises the odd digits in the decimal expansion of } x \\ x_2 \text{ comprises the even digits in the decimal expansion of } x \end{aligned} \tag{4}$$

However, such functions are rather unusual, and tend to have properties that do not fit the spirit of ESNs as a theoretical model for reservoir computing, such as here being discontinuous everywhere. In particular, no such bijective function can be implemented by linear algebra, such as multiplication by a matrix. Indeed, any linear projection from $\mathbb{R} \to \mathbb{R}^n$ results in a 1-dimensional line within $\mathbb{R}^n$.

We can show that linear algebra cannot map onto the whole higher-dimensional space by applying the Rank-Nullity Theorem [20], which is also known as the Fundamental Theorem of Linear Algebra.If $\mathbf{W}^{in}$ is the matrix projecting the m-dimensional input $\mathbf{u}$ into n-dimensional space, then the Rank-Nullity Theorem gives $rank(\mathbf{W}^{in}) + nullity(\mathbf{W}^{in}) = m$, where $rank$ is the matrix rank function, representing the dimension of the image, and $nullity$ is the nullity of the matrix. The nullity represents the dimension of the kernel of the matrix (unrelated to the Kernel Rank measure in Reservoir Computing) , i.e. the dimension of the

space of vectors where $\mathbf{W}^{in}\mathbf{u}_1 = \mathbf{W}^{in}\mathbf{u}_2$. This trivially implies $dim(Im(\mathbf{W}^{in})) \leq dim(Domain(\mathbf{W}^{in}))$.

Therefore the transformed inputs occupy at most an m-dimensional hyperplane of the n-dimensional echo space. Contrary to the literature, the projection provided by $\mathbf{W}^{in}$ cannot increase the dimensionality of the input, and may even *decrease* it, if not chosen well. If the values of $\mathbf{W}^{in}$ are chosen in a uniform random manner from the range $[-1, 1]$, the probability of a decrease in dimensionality is negligible. However, the decrease in dimensionality is plausible if the the values of $\mathbf{W}^{in}$ are chosen from a discrete set, such as $\{-1, 0, 1\}$.

If $nullity(\mathbf{W}^{in}) = 0$, then $dim(Im(\mathbf{W}^{in})) = dim(Domain(\mathbf{W}^{in})$, and the use of linear algebra guarantees the transformation is surjective. We can therefore define $\mathbf{W}^{in\prime}$ such that $\mathbf{W}^{in\prime}(\mathbf{W}^{in}\mathbf{u}) = \mathbf{u}$. Note that this is *not* a bijection, as there exist points in $\mathbb{R}^n$ which are not in $Im(\mathbf{W}^{in})$.

With $\mathbf{W}^{in\prime}$ defined, we can show that the projection into a higher dimensional space does not contribute to linear separability. Suppose that there exist two sets of inputs A, B such that for all $\mathbf{u}_a \in A$ and $\mathbf{u}_b \in B$, $\mathbf{W}^{in}\mathbf{u}_a$ and $\mathbf{W}^{in}\mathbf{u}_b$ are linearly separable by the hyperplane $H_{sep} \in \mathbb{R}^n$. As $\mathbf{W}^{in}$ is a linear transformation, we know that that all points $\mathbf{W}^{in}\mathbf{u}_a, \mathbf{W}^{in}\mathbf{u}_b$ lie on some hyperplane H_{inp} given by $Im(\mathbf{W}^{in})$. As H_{sep} separates points that lie on H_{inp}, by hypothesis, we know these two hyperplanes are not parallel. Therefore we can intersect H_{inp} and H_{sep} to get H_{int}. Finally, as points on H_{int} trivially lie in $Im(\mathbf{W}^{in})$, we can transform an orthognal basis of H_{int} using $\mathbf{W}^{in\prime}$, and find that $\mathbf{W}^{in\prime}(H_{int})$ is a hyperplane that separates A and B.

From the above it is clear that the ability to linearly separate points in echo space is *not* a function of the projection into the higher dimensional space of the ESN. Combined with the apparent insufficiently nonlinear activation functions, we move onto the final part of the ESN equation – the connectivity of the nodes – to attempt to identify how an ESN achieves linear separability.

3.3 Chains and Recurrence as Memory

Having established that the projection into the echo space of an ESN is not responsible for any enhanced linear separability, the next step is to identify where this capability comes from.

An ESN can be treated as a randomly connected graph. Recall that when doing so, the weight matrix of the ESN can be treated as a weighted adjacency matrix, allowing a matrix to be interpreted as a directed graph. Examples of this are shown in Figures 6 and 7. In these figures, the nodes labelled u_i represent where inputs $\mathbf{u}(t)$ enter the ESN. Other nodes sum the weighted inputs and then apply the activation function.

Assuming that the activation function is not particularly significant, we can identify two clear structures that have the potential for memory. First are terminated chains, as shown in Figure 6. These are a simple chain of nodes within the ESN with a clear end. From an input node, a terminated chain of length t provides information on the past t inputs. If the spectral radius of the ESN is less than 1, then the values in these chains diminish as they progress along the

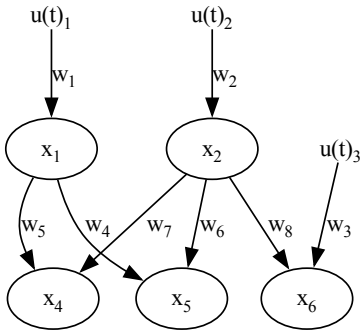

Fig. 6: Multiplexed chains in an ESN

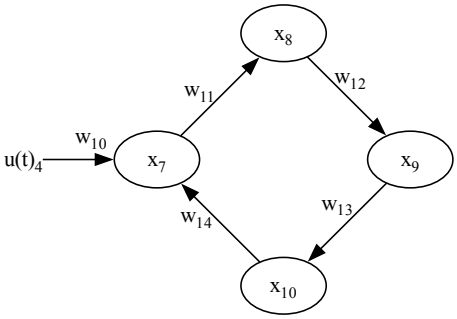

Fig. 7: Loop in an ESN

chain. They diminish by a fixed amount: the product of the weights of the chain to that point in history. Therefore, this diminishment is largely irrelevant, as the linear regression output layer can learn these weights to recover the history.

The information provided by a chain is effectively equivalent to a single dimension of the input. This means that multiple-dimensioned inputs require multiple chains to attain the same memory. These chains can be multiplexed, because the linear regression layer can solve linear algebra equations. For example, in Figure 6, the node values x_4, x_5 and x_6 sum some combination of values from all three inputs, at different stages of time. We can express the values of these nodes at time t as:

$$x_4 = w_1 w_5 u(t-2)_1 + w_2 w_7 u(t-2)_2 \tag{5}$$

$$x_5 = w_1 w_4 u(t-2)_1 + w_2 w_6 u(t-2)_2 \tag{6}$$

$$x_6 = w_2 w_8 u(t-2)_2 + w_3 u(t-1)_3 \tag{7}$$

Given that the weights are typically randomly generated floating point values, they are unlikely to be identical, and so these equations are likely to be independent. Therefore, as we have three equations and the three values $u(t-2)_1$, $u(t-2)_2$ and $u(t-1)_3$ to recover, there is a linear solution, and therefore linear regression can recover all three values. While this example used a sparse input matrix for simplicity, meaning that not all nodes received all inputs, this also holds for ESNs with a densely connected $\mathbf{W}^{in}$.

The second structure is a cycle of nodes, such as in Figure 7. A cycle has similar properties to a chain, especially if the number of nodes in the loop is large. In this case, the diminishing of value due to the spectral radius ensures that even a large initial input will be mostly insignificant once it has made its way around the cycle. However, if the size of the loop is small, or the spectral radius is close to 1, then a cycle is noticeably different from a chain. In Figure 7, the cycle exhibits memory modulo 4: the input is added to the residual value from 4 timesteps ago. In practical terms, this means that a cycle can be sensitive to periodic inputs,

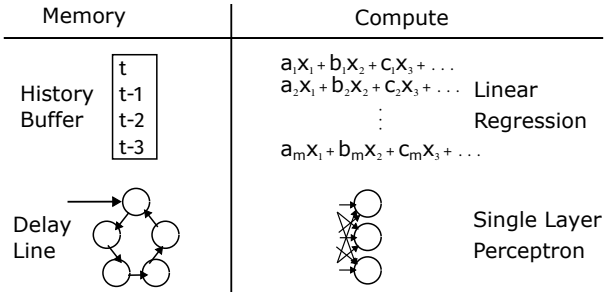

Fig. 8: LIRC Method

provided that the periodic input is of great enough magnitude to overcome the diminishing of being passed through the cycle. However, a similar effect can be accomplished with linear regression, provided that the stored history is long enough.

As an ESN is randomly connected, these chain and loop structures occur at random. They may also be combined with other structures, for example, a chain-like structure with extra inputs. These combinations are not necessarily significant, as they all constitute linear algebra. A combination of values may change the way in which information is recovered, but provided that the history of time t is comprised of weights that form a basis for the input, a linear regression layer can extract that history.

Unlike the cases of input projection and the activation function, the chains of nodes within an ESN do increase the dimensionality of the state of the ESN. By encoding the values of previous inputs, by storing history, the amount of data available increases.

This raises a question: can we increase the dimensionality by simpler means?

4 LIRC: Lagged Input Regression Computation

To recap: we have shown that the tanh() activation function and linear projection into a higher dimensional space do not yield an increase in dimensionality. In this situation, the only increase in dimensionality is given by the memory of the ESN storing past values. One question that arises is can the setup be simplified and memory provided by simply time-lagging inputs?

Time-lagged data is not a new idea in reservoir computing: delay-lines [19] are commonly used to implement feedback in physical systems. An alternative is the so-called "single dynamical node" approach of Appeltant et al. [3], which stores the last n outputs. In both cases, either the virtual nodes of the delay line or output buffer of the single dynamical node are then utilised by a linear regression layer which can be trained to a task. Both of these approaches can be adapted to store information on past inputs.

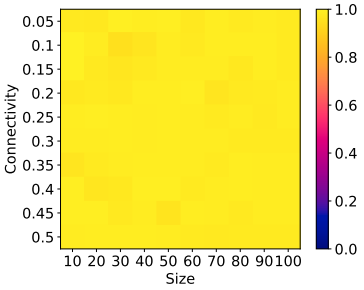

Fig. 9: R^2 of tanh() ESNs approximated by Linear Regression on a Single Dynamical Node of length 10

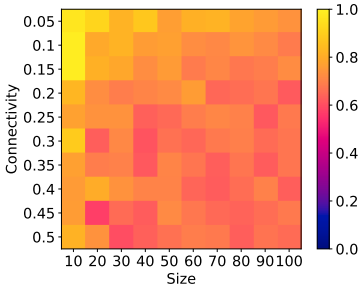

Fig. 10: R^2 of LeakyReLU ESNs approximated by Linear Regression on a Single Dynamical Node of length 10

While delay-lines and the single dynamical node approach use linear regression, more recent work has identified that the capability for nonlinear readout [9,15] can be beneficial. Therefore, in addition to using Linear Regression, we also propose the use of a trained nonlinear readout, in the form of a single layer perceptron (SLP). Use of an SLP sidesteps the need to define a precise nonlinearity, whilst still being easy to train.

The combination of time lagged inputs and regression results is here named Lagged Input Regression Computation (LIRC). LIRC separates the memory and computational aspects of a system (Figure 8). This enhances the understandability of the system, and in some configurations may be simpler to implement.

4.1 Comparison to ESNs

The first evaluation of LIRC as an approach is to compare against ESNs. In this case, randomly generated ESNs are used as a benchmark that LIRC methods attempt to predict. Depending on the LIRC methods with the greatest accuracy, we can make statements about the complexity of the ESN. To ensure a wide variety of configurations were tested, we conducted a parameter sweep on the spectral radius, connectivity, size and activation function of the ESN, with all tests being carried out against all combinations of LIRC approximations. All experiments were repeated 5 times with different random seeds to verify the results. Accuracy is measured using the coefficient of determination R^2 [10]; for the presented heatmaps, values below 0 are deemed sufficiently inaccurate to be unusable and clamped to 0. More parameters were explored than can be represented on a 2D heatmap; each cell represents the average of all experiments with those parameters (100 experiments per cell). The effects of parameters not mentioned in the graphs, such as spectral radius, were observed to be minimal.

Figure 9 shows that ESNs with a tanh() activation function are trivially approximable via linear regression on time lagged inputs. Even in cases where the ESN is small or the connectivity is low, this holds true, with a minimum

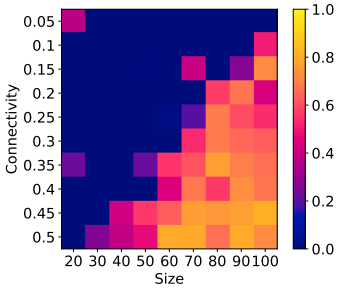

Fig. 11: R^2 of LeakyReLU ESNs approximated by an SLP on a Single Dynamical Node of length 20

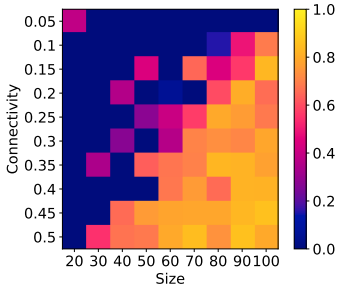

Fig. 12: R^2 of LeakyReLU ESNs approximated by an SLP on a Delay Line of length 20

accuracy of 98% being observed. However, linear regression is unable to handle the LeakyReLU activation function (Figure 10), due to the nonlinearity of LeakyReLU occurring at 0, thus meaning the linear regression layer can train to fit only half of the output data.

In order to achieve more accurate results for LeakyReLU, it is necessary to use the SLP (Figures 11 and 12); however, even this requires favourable conditions to achieve a high level of accuracy, with a large size, unreasonably high connectivity and increased available memory required to achieve greater than 80% accuracy. This illustrates that when choosing a nonlinearity for an ESN, the probability of the nonlinearity having a significant impact must be taken into account.

There is a noticeable improvement in the accuracy of the approximation when using a Delay Line for memory with LeakyReLU, as shown in Figure 12, where the greatest accuracy is 0.92, as opposed to 0.79 in Figure 11. This suggests that the unbounded nature of the LeakyReLU activation may be capable of causing data to persist in cycles more than is the case with tanh. However, an increase in accuracy of the approximation indicates that the ESN is not providing more computational ability than a LIRC-SLP with the same number of nodes. As SLPs are simpler to compute and better studied, this raises questions around the efficacy of large ESNs. Smaller ESNs are less approximable, and therefore give a different computation to an equivalently sized LIRC-SLP.

4.2 Nonlinear memory capacity

Next we turn to evaluating the linear regression based LIRC models for nonlinear memory capacity. This may seem unusual; a linear system should not be able to exhibit nonlinear behaviour. However, we have shown that we can approximate an ESN with the tanh() activation function to a high degree of accuracy with a linear system. This is apparently at odds with the results of Dambre et al. [8], who show that ESNs exhibit nonlinear memory of odd degree, measured using the orthogonal set of Legendre Polynomials (used for uniform random input).

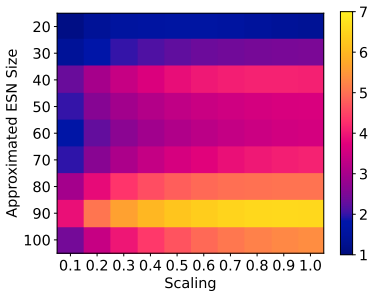

Fig. 13: P_3 Memory Capacity of Lagged Input Linear Regression Approximating a tanh() ESN

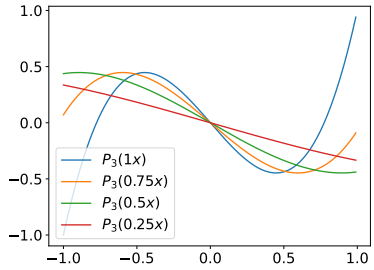

Fig. 14: Legendre Polynomial P_3 with different input scalings

This same nonlinear memory capacity exists in our linear system for odd degrees of Legendre Polynomials (e.g. Figure 13 for P_3). This is because all odd degrees of Legendre Polynomials include a linear term. For example, $P_3(x) = \frac{1}{2}(5x^3 - 3x)$. As $x \to 0$, the linear term dominates. Therefore, there is a region of the odd degree Legendre Polynomials that is approximable by a linear system, with the size of this region decreasing for higher degrees of Legendre Polynomial, explaining the apparent nonlinear memory capacity. This observation also applies to other orthogonal polynomials, such as Hermite Polynomials used with Gaussian inputs [8].

Even degrees of the Legendre Polynomial lack this linear term, and so are not approximable in this manner, but are zero for the reason Dambre et al. [8] give: tanh() is an odd function, which matches the odd degrees.

Dambre et al. [8] also observe that by changing the input scaling, the system appears to become more nonlinear, as the contribution of higher degrees becomes greater. This is correct, but somewhat misleading. Changing the input scaling stetches the polynomial (Figure 14), which increases the range of the linearly approximable region. For $P_3(0.25x)$, the function is almost entirely linear across $[-1, 1]$. If this range exceeds $[-1, 1]$, then the tanh() activation clamps the values, resulting in the linear section becoming nonlinear. The effect of this is that as the scaling changes, so does the Legendre Polynomial that is most approximable by a linear system. While this scaling is not reflected exactly in an ESN or linear approximation due to the summation of values in the ESN, in Figure 13 we do see that the optimal P_3 memory is at a scaling of 0.8.

5 Conclusion

Several of the claims in the literature about how ESNs perform their computations, and what computations they perform, may be overstated.

ESNs with the commonly-used tanh() activation function can be accurately approximated by performing linear regression on time lagged inputs. This is due

to the nonlinearity of tanh() not being substantial for the majority of values likely to occur within such an ESN. tanh() ESNs tend to behave as linear memory systems, with the linear regression layer unscrambling the random weighted connections within an ESN to recover data. Linear regression over simple time-lagged memory structures is sufficient to solve many problems, and should be used as a control case when evaluating ESN performance.

The LeakyReLU activation function is *not* approximable by linear regression to the same degree. Even when using the more powerful SLP-based regression method, LeakyReLU ESNs require favourable conditions to be well approximated. Despite being effectively a choice between two linear functions, LeakyReLU produces more complex dynamics in an ESN by virtue of the nonlinearity being more likely to be exploited.

Other authors have demonstrated nonlinear memory capacity in tanh() ESNs. These results can be explained by the odd order Legendre Polynomials containing a linear term, which dominates the nonlinear terms for sufficiently small input values. When this linear term dominates in the range $[-1, 1]$, the Legendre Polynomial is approximable by linear regression. By varying the scaling factor, it is possible to vary which of the Legendre Polynomials contributes most nonlinear memory capacity. This gives the illusion of nonlinear memory capacity, when in fact only linear memory capacity is being demonstrated.

Further work is needed to evaluate other activation functions, and to determine the desirable properties for an activation function of an ESN, or indeed if an ESN is still a useful tool given recent work that utilised simplified structures to achieve similar results [9,18]. In addition, it may be necessary to revisit the use of orthogonal polynomials for determining nonlinear memory capacity.

Statements and Declarations

The authors acknowledge funding from the MARCH project, EPSRC grant numbers EP/V006029/1. The authors have no relevant financial or non-financial interests to disclose.

References

1. Abramowitz, M., Stegun, I.A.: Handbook of mathematical functions: with formulas, graphs, and mathematical tables, vol. 55. Courier Corporation (1965)
2. Amari, S.I.: Learning patterns and pattern sequences by self-organizing nets of threshold elements. IEEE Transactions on Computers **C-21**(11), 1197–1206 (1972). https://doi.org/10.1109/T-C.1972.223477
3. Appeltant, L., Soriano, M.C., Van der Sande, G., Danckaert, J., Massar, S., Dambre, J., Schrauwen, B., Mirasso, C.R., Fischer, I.: Information processing using a single dynamical node as complex system. Nature communications **2**(1), 468 (2011)
4. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg (2006)

5. Bollt, E.: On explaining the surprising success of reservoir computing forecaster of chaos? the universal machine learning dynamical system with contrast to var and dmd. Chaos: An Interdisciplinary Journal of Nonlinear Science **31**(1) (2021)
6. Buehner, M., Young, P.: A tighter bound for the echo state property. IEEE Transactions on Neural Networks **17**(3), 820–824 (2006). https://doi.org/10.1109/TNN.2006.872357
7. Cover, T.M.: Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. IEEE Transactions on Electronic Computers **EC-14**(3), 326–334 (1965). https://doi.org/10.1109/PGEC.1965.264137
8. Dambre, J., Verstraeten, D., Schrauwen, B., Massar, S.: Information processing capacity of dynamical systems. Scientific reports **2**(1), 514 (2012)
9. Gauthier, D.J., Bollt, E., Griffith, A., Barbosa, W.A.S.: Next generation reservoir computing. Nature Communications **12**(1) (Sep 2021). https://doi.org/10.1038/s41467-021-25801-2
10. Glantz, S.A., Slinker, B.K., Neilands, T.B.: Primer of applied regression & analysis of variance. (No Title) (1990)
11. Gu, B.: Lecture notes: The power method for spectral radius (2023), https://users.wpi.edu/~bgu/sp23/ma3257/lecture_notes/MA3257_L22.pdf
12. Hart, A.G., Olding, K.R., Cox, A.M.G., Isupova, O., Dawes, J.H.P.: Using echo state networks to approximate value functions for control (2021), https://arxiv.org/abs/2102.06258
13. Hoerl, A.E., and, R.W.K.: Ridge regression: Biased estimation for nonorthogonal problems. Technometrics **12**(1), 55–67 (1970). https://doi.org/10.1080/00401706.1970.10488634
14. Jaeger, H., Haas, H.: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. science **304**(5667), 78–80 (2004)
15. Ma, H., Prosperino, D., Räth, C.: A novel approach to minimal reservoir computing. Scientific Reports **13**(1), 12970 (2023)
16. Mackey, M.C., Glass, L.: Oscillation and chaos in physiological control systems. Science **197**(4300), 287–289 (1977)
17. Mozer, M.C.: A focused backpropagation algorithm for temporal pattern recognition. In: Backpropagation, pp. 137–169. Psychology Press (2013). https://doi.org/10.4324/9780203763247
18. Prosperino, D., Ma, H., Räth, C.: Tailored minimal reservoir computing: on the bidirectional connection between nonlinearities in the reservoir and in data (2025), https://arxiv.org/abs/2504.17503
19. Soriano, M.C., Ortín, S., Keuninckx, L., Appeltant, L., Danckaert, J., Pesquera, L., Van der Sande, G.: Delay-based reservoir computing: noise effects in a combined analog and digital implementation. IEEE transactions on neural networks and learning systems **26**(2), 388–393 (2014)
20. Strang, G.: The fundamental theorem of linear algebra. The American Mathematical Monthly **100**(9), 848–855 (1993), http://www.jstor.org/stable/2324660
21. Waheeb, W., Ghazali, R., Shah, H.: Nonlinear autoregressive moving-average time series forecasting using neural networks. In: 2019 International Conference on Computer and Information Sciences. pp. 1–5 (2019). https://doi.org/10.1109/ICCISci.2019.8716417
22. Wringe, C., Trefzer, M., Stepney, S.: Reservoir computing benchmarks: a tutorial review and critique. International Journal of Parallel, Emergent and Distributed Systems pp. 1–39 (2025)
23. Yan, X., Su, X.: Linear regression analysis: theory and computing. World Scientific (2009)