Echo State Networks

Paul Jason Mello
Department of Computer Science and Engineering
University of Nevada, Reno
pmello@unr.edu

Abstract

Echo State Networks (ESNs) are a specialized class of Recurrent Neural Networks (RNNs) that replace gradient based training with a fixed but randomly connected ”reservoir” of neurons and a trainable linear readout. This style of reservoir computing yields and input driven, fading memory dynamics system governed by the Echo State Property (ESP). ESP is the critical property which makes ESNs possible and states that for any bounded input sequence, the reservoir's state will eventually converge to a unique function of the input history that is independent of initial conditions. ESNs are characterized by their spectral radius, leak rate, input scaling, and sparsity which together yield a rich and expressive reservoir that serves both as a nonlinear, high dimensional expansion of the input $u(n)$ into states $x(n)$ and as a fading memory of $u(n)$. More succinctly this means that ESNs comprise a reservoir, which contain many subnetworks, of dynamics which are ”tapped” by trained output units. In this work, we explore the structure and methodology of ESNs then experiment on a custom Mackey-Glass glass dataset which consists of learning chaotic dynamics. Through experimental tests we attain an NRMSE of 0.028 with an R^2 of 0.999. These results confirm that ESNs deliver high quality chaotic forecasting while collapsing recurrent training to what is essentially linear regression. The code and results can be found here: `https://github.com/pauljmello/Echo-State-Networks`

“Every model is just an RNN with a mask, and RNNs are just linear regression with a clock.”

1 Introduction

RNNs have become a corner stone of modern artificial intelligence with mounting evidence corroborating their fundamental nature in efficient and effective sequence modeling. While RNNs theoretically possess the ability to universally approximate complex temporal dynamic systems, the practicality of such training has proven very difficult. The vanish gradient problem through time series forecasting has proven to be unstable and computationally expensive.

In light of these challenges and limitations, ESNs, introduced in [1], have offered an elegant and extremely efficient alterantive to traditional RNN approaches. Rather than consistently updating the network weights through gradient descent, ESNs leverage a fixed, randomly initialized reservoir of recurrent connections to generate the capture of rich dynamical representations of the input history through subnetworks. In this approach, only the linear readout weights from the reservoir to the output layer require training, which ultimately transforms the inherently nonlinear problem of RNN training into a linear regression task.

ESNs thus provide a profound insight into the structure of RNNs and artificial intelligence models in general:

A large, randomly connected recurrent network can serve as a reservoir of dynamics that is rich and expressive enough to be linearly combined into any desired temporal pattern.

Here the reservoir acts both as a nonlinear and high dimensional expansion of the input signal and as a fading memory of the prior input history. This simultaneous nature provides the model with the necessary temporal context for sequence modeling like learning chaotic dynamical systems.

With these concepts in mind, we explore the theoretical underpinnings and fundamental mechanisms of ESNs through a theoretical lens and empirical results on Mackey-Glass chaotic time series data. Our experimental results achieve a strong NRMSE and even stronger correlation coefficient between the predictions and ground truth. We offer an understanding of ESNs, a reservoir computing approach, as an effective and efficient paradigm for time series prediction tasks.

2 Background

2.1 Recurrent Neural Networks

RNNs arouse as an architecture to extend feedforward network architectures by introducing a cyclical connections which transition necessary contextual information across timesteps. These networks are defined by a hidden state $x(t)$ and an input $u(t)$ which evolves according to the following equation:

$$x(t+1) = f(W^{\text{in}}u(t) + Wx(t) + b) \tag{1}$$

where W^{in} represents input to hidden weights, W represents recurrent hidden to hidden weights, b is a bias term, and f is a nonlinear activation function. For this system we typically compute $y(t)$ as:

$$y(t) = f^{\text{out}}(W^{\text{out}}x(t)) \tag{2}$$

Training these networks requires backpropogation through time (BPTT) [3] which itself requires unrolling the network across each timestep. This process has the unfortunate issue where when the spectral radius of $W < 1$ the gradients will exponentially decay as they propagate their signal through each layer. This processes irreversible destroys the long term learn capabilities of these models and subsequent sequencing issues. While in the inverse case, where $W > 1$ the gradients grow exponentially and become unstable. Both of these issues have been a significant challenge of RNN based training that has made the training process computationally expensive and difficult to tune parameters for efficiency on the task of long term sequence modeling like that of chaotic dynamical systems.

2.2 Reservoir Computing

Reservoir computing address the challenges presented in 2 with a new perspective on RNNs. Rather than training the recurrent connections, which introduce computationally expensive and unstable training, we fix the recurrent connections and randomly initialize them such that our training targets are reduced to only the linear readout function. This approach rectifies the prior described issues of RNNs through two key observations. The first being that a sufficiently large random recurrent network generates a high dimensional, nonlinear projection of its input history. The second being that if the network is stably parametrized, these projections contain enough information about previous inputs which can be linearly combined into a desired output.

From an information theoretic perspective, the reservoir dynamics induce a Markov chain over the state space where the current state $x(n)$ servers as a sufficiently informational statistic for the entire input history. By the Markov property, $I(x(n); u(0), ..., u(n-k)|x(n-1)) = 0$, given the previous state $x(n-1)$, the current state contains no additional information than the immediate past. This intuition that the reservoirs capacity to compress temporal context into a fixed dimensional vector fundamentally bounds what the readout can learn becomes even more profound as we consider the fading memory property. The property implies that while $x(n)$ technically encodes the entire history, the effective memory is finite. The generally means that the reservoir dynamics filter historical inputs, preserving only temporal features relevant to the timescale set by the networks spectral properties.

Notably, ESNs are the quintessential model of reservoir computing, but that also similarly related to Liquid State Models which are both inspired by biologically spiking neurons. They both comprise the idea that a complex, but fixed dynamical system can serve as a strong feature extractor. Additionally, in both cases, these models learn an effective linear combination of these extracted features.

3 Echo State Networks

3.1 ESN Architecture

Similar to the design standards of artificial neural networks, there is an input layer, a hidden layer, and an output layer. However, rather than having n hidden layers, ESNs consist of a single hidden layer which is the reservoir. This design is illustrated in figure 1.

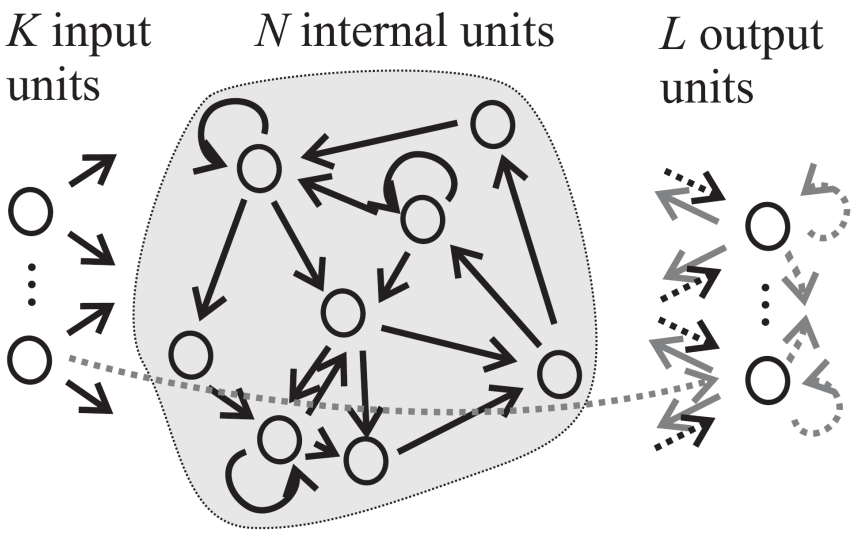

Figure 1: Basic Echo State Network Design. Figure from [1]

Here the diagram illustrates the fundamental connection possibilities of each recurrent unit. We consider discrete time neural networks with K input units, N internal units, and L output units. Activations at a time step n are $u(n)$, internal units are $x(n)$, and output units are $y(n)$. Connections from input to output units are allowed and connections from output to output units are also allowed. Similarly, connections from internal units to internal units and even recurrent connections that are self referential for internal units and output units are allowed. The only connection not allowed is from output units to input units or internal units to input units

ESNs are then fundamentally characterized by three distinct components as follows:

1. Input layer: K input units receiving signals $u(n) \in \mathbb{R}^K$.
2. Reservoir (hidden): N recurrent units with states $x(n) \in \mathbb{R}^N$.
3. Readout (output): L output units producing $y(n) \in \mathbb{R}^L$.

3.2 Reservoir Dynamics

The reservoir is updated according to the discrete time update equation:

$$x(n+1) = f\big(W^{\text{in}}\, u(n+1) + W\, x(n) + b\big)\,, \tag{3}$$

where,

- $W^{\text{in}} \in \mathbb{R}^{N \times K}$ represents the input weight matrix

- $W \in \mathbb{R}^{N \times N}$ represents the reservoir weight matrix
- $b \in \mathbb{R}^{N}$ represents the bias vector
- $f = (f_1, \ldots, f_N)$ denotes element wise activation functions like $\tanh$

For tasks with output feedback the equation has to be redefined such that:

$$x(n+1) = f\big(W^{\text{in}}\, u(n+1) + W\, x(n) + W^{\text{back}}\, y(n) + b\big)\,, \tag{4}$$

where $W^{\text{back}} \in \mathbb{R}^{N \times L}$ represents the potential for output feedback weights.

The output is then formally computed as follows:

$$y(n+1) = f^{\text{out}}\left(W^{\text{out}} \begin{bmatrix} u(n+1) \\ x(n+1) \\ y(n) \end{bmatrix}\right)\,, \tag{5}$$

Here, $[\, u(n+1)\,;\, x(n+1)\,;\, y(n)\,]$ denotes concatenation, and very notably $W^{\text{out}} \in \mathbb{R}^{L \times (K+N+L)}$ are the only trainable parameters in the whole network.

In the simplified case, where potential connections do not include output feedback nor input units connecting to output units, we can simplify this:

$$y(n+1) = f^{\text{out}}\big(W^{\text{out}} x(n+1)\big)\,, \qquad W^{\text{out}} \in \mathbb{R}^{L \times N}. \tag{6}$$

3.3 Leaky Integrator Neurons

Introducing a leaky integrator is a very useful approach to improving the temporal dynamics of ESNs. The leaky integrator makes each unit in the system a first order low pass filter. This effectively amounts to shrinking the Jacobian down toward the identity matrix, further enlarging the reducing the drift or chaos from W. The consequences of this addition is a reservoir with improved stability, better memory timescale, and aligning the dynamics of the reservoir with the task's temporal structure. This integrator can be implemented with a discretized equation from a continuous time differential equation. The differential update starts with the following equation:

$$\frac{dx}{dt} = -x + \tanh\left(W^{\text{in}} \begin{bmatrix} 1 \\ u(t) \end{bmatrix} + W\, x(t)\right) \tag{7}$$

Discretizing with timestep Δt on the following equation

$$\frac{x(n+1) - x(n)}{\Delta t} \approx \frac{dx}{dt} \tag{8}$$

results in changes to differential equation to introduce the leaky integrator.

$$x(n+1) = (1-\alpha)\, x(n) + \alpha\, \tanh\left(W^{\text{in}} \begin{bmatrix} 1 \\ u(n+1) \end{bmatrix} + W\, x(n)\right) \tag{9}$$

Here, $\alpha \in (0, 1]$ is the leak rate. Where an $\alpha = 1$ corresponds to standard discrete time updates, where as an $\alpha < 1$ is an exponential averaging.

In essence, the integration of a leaky integrator provides significantly better coverage with better generalization by matching the reservoir's memory timescale to the task, shrinking the Jacobian to suppress unstable modes, and reducing the state variance and collinearity for a stronger and more stable solution to ridge regression which we cover in the following section 3.3. Ultimately this addition provides more stable ESP and consistently lower test errors. We will cover ESP in more depth in 4

3.4 Training Readout: Ridge Regression

The beauty of ESNs is illuminated in their training methodology. Particularly, the ESN training can be conceptually reduced to linear regression on reservoir states. First we collect the reservoir states during training $X = \big[x(n_{\min}), \dots, x(n_{\max})\big] \in \mathbb{R}^{N \times T}$, then collect the corresponding outputs $Y = \big[y_{\text{target}}(n_{\min}), \dots, y_{\text{target}}(n_{\max})\big] \in \mathbb{R}^{L \times T}$, which amounts to solving a linear regression problem. In the scenario where the values arent clipped in regression space, we then map the predictions back to a bounded range to maintain stability during training.

In both of these cases, we must construct a design matrix by augmenting the states with a bias column. In this representation the design matrix is a time by feature table of augment states which are augmented with a bias to allow the model to learn more complex dynamics. We can construct it as such:

$$E = \begin{bmatrix} x(n_{\min})^\top & 1 \\ \vdots & \vdots \\ x(n_{\max})^\top & 1 \end{bmatrix} \in \mathbb{R}^{T \times (N+1)} \tag{10}$$

Here, each row of E is $([, x(n)^\top; 1,])$ where the reservoir state exists with an added bias, giving a $(T \times (N+1))$ design matrix that captures the time steps × features.

$$\hat{\Theta} = \left(E^\top E + \lambda I\right)^{-1} E^\top Y \tag{11}$$

$(\hat{\Theta})$ is solved for by $((E^\top E + \lambda I)\hat{\Theta} = E^\top Y)$. The small diagonal term of (λI) results in a stabilization that fits and shrinks the weights toward zero for better generalization. This solution fundamentally amounts to ordinary least squares. We use uniform ridge regularization, i.e., the penalty matrix is I_{N+1} so all parameters in Θ (including the bias) are penalized equally. If one wishes to exclude the bias from regularization, replace I_{N+1} with $D = \operatorname{diag}(1, \dots, 1, 0)$. In general, a uniform distribution is preferred for its continuity of values and boundedness, but many works have also utilized a gaussian distribution with negligible differences in results.

4 Theoretical Foundations

4.1 The Echo State Property

ESP is a fundamental requirement for ESNs to function as we now understand. This property is defined in the following way; that a network will exhibit ESP if the reservoir state $x(n)$ is uniquely determined by the input history, independent of all initial conditions. This intuitively makes sense as the ESP ensures the ESN's current state only "echoes" the input history, without any noise introduced by the initial states. This means that the reservoir is a well defined bank for dynamical systems rather than an autonomous system with its own intrinsic and self-evolving dynamics. The ESP is more formally defined to arise under any one of the following equivalent conditions:

1. State Contracting: The transition mapping of $T(x, u)$ is a uniform contraction of x.
2. State Forgetting: Says that states which are initialized from different starting conditions will ultimately converge exponentially under identical input sequences.
3. Input Forgetting: When input sequences end in the same sufficiently long history, they produce similar states.

Notably, input forgetting demonstrates a universal concept defined by the Markov property. Chained inputs through contractive dynamics naturally induce a Markov property where only the most recent limited history will determine the present state.

A second important consequence of the ESP is that, if a network satisfies any of the ESP propositions above, then it also posses a fading memory continuity. That the input to state mapping depends informally on a limited recent number of the prior history, similar to the input forgetting idea. This sees these networks acting as temporal filters similar to other network types like liquid state networks.

A third and final proposition of ESNs comes from a concept we have used but not touched on yet; spectral radius and more specifically the conditions necessary for ESP. For context, the spectral radius $p(W)$ is the maximum absolute eigenvalue of the weight matrix which helps to contract or expand the vectors. The key here is that the spectral radius in discrete time systems would require a $p < 1$ such that there is an exponential contraction toward the origin. If $p > 1$ then there would exponential divergence. This is critical as within the nonlinear dynamics framework we have built, and utilizing the tanh activation function, the spectral radius of W is bounded by the Jacobian's eigenvalues near equilibrium. This has the interesting consequence of controlling the timescale of information decay in the system and the memory horizon of the network.

4.2 Sparsity and Scaling

When problems become sufficiently large, it becomes necessary to optimize the spectral radius through scaling. This is because we want to optimize the short term memory of the system to extract as much information from prior states as possible, yet ensuring that $p < 1$ to preserve the exponential forgetting and guarantee ESP.

The most common and effective approach to handle this is scaling. In this situation we initialize W randomly, then scale it to achieve some desired spectral radius ρ, which is often close to 1:

$$W = \frac{\rho}{\rho_{\text{init}}} W_{\text{init}} \tag{12}$$

where $\rho_{\text{init}} = \max |\lambda_i(W_{\text{init}})|$. Another effective approach to managing these intractably large networks, is sparsity. Sparsity is defined by setting the weights of certain neurons to exactly zero, so that they do not influence any other neurons. This sparsity introduces highly desirable benefits including the following:

1. Computation efficiency introduced by sparse matrix operations
2. Decoupling of subnetworks leading to diversifying the individual dynamics preserved by the system
3. Reducing the potential degrees of freedom for neurons, which help neurons learn more generalizable patterns

Sparse connectivity is typically defined between the ranges of $1 - 20\%$, and empirical observations have found that the ideal number of neuron to neuron connections are approximately 10 [2].

4.3 Dynamics Perspective

4.3.1 Washout and Transient Dynamics

Since we randomly initialize $x(0)$, these states are considered "contaminated" after being initialized. They are contaminated by transients that reflect the initial conditions rather that the input history. Transients can be considered as random noise or "ringing" from the arbitrary initialization. Since we are learning to readout the trajectories, the initial states have many poorly chosen paths. Washout is the process of dropping these initial state variations so that signal is echoing the proper target sequence. This process is a critical and necessary step to settle the learned network dynamics.

4.3.2 The Big Picture

From a dynamics perspective, and being that ESNs are a special case of RNNs, ESNs flip RNNs on their head by fixing the representational vector field. Rather than training to update the vector field, the vector field is randomly initialized and learns only how to read out the desired trajectories from the field. In other words, we are not training to update the representational space, but training to read the trajectories from a static representational space.

5 Experimental Setup

We implement a simple ESN architecture that was original established in [1]. The system consists of a randomly initialized input weight matrix that projects incoming signals to the reservoir, a sparsely connected recurrent reservoir with weights scaled to control the spectral radius and ensure ESP, and a trainable linear readout layer. We utilize a leaky integrator neuron model where state updates are combined bother with current and prior activations through the $\tanh$ activation function. These choices create a rich temporal representation of the input dynamics. Finally, to train the ESN we follow a two-phase approach where reservoir weights remain fixed while the output layer is trained on ridge regression. This ridge regression solve a least square problem that seeks to map reservoir states to target outputs. We also utilize washout to remove transient noise dynamics and stabilize prediction outputs to only the input history. This implementation also supports both dense and sparse reservoir connection paradigms.

Utilizing a hyperparameter search over 1000 configurations, we found the following configuration and rounded up some values: reservoir size $N = 550$ neurons, spectral radius $\rho = 0.92$, a sparse connection density of 17%, input scaling set to 3.2, a leak rate of $\alpha = 0.79$, ridge regularization $\lambda = 0.19$, a washout period of 250 steps, a training sequence length of $12,000$ steps, and test sequence length of $3,400$ steps. For our experimentation we train our ESN on a standard chaotic time series prediction task: the Mackey-Glass delay differential equation. Here the goal is to predict a single step ahead to prove the model does learn and follow ESP.

5.1 Training Process

The complete training procedure follows these steps:

1. Generate Data: Generate a Mackey-Glass dataset consisting of a sequence of length $\texttt{train length} + \texttt{test length} + \texttt{washout} + 1$
2. Initialize Reservoir: Randomly initialize W^{in}, W, and bias vectors, then scale W to the targeted spectral radius
3. Washout: Run the reservoir with the first `washout` time steps to remove the initial transients
4. State Collection: Collect reservoir states $x(n)$
5. Ridge Regression: Solve for W^{out} using the collected states and their corresponding targets
6. Evaluate: Test the trained ESN on the hold out test data

The beauty of this training process, and the ESN approach more generally, is that it only requires only a single forward pass. This is then followed up with a matrix inversion, completely avoiding the need for a optimization algorithm, and more generally the pit falls described in section 2.

6 Results

6.1 Quantitative Performance

Through our experimentation, and hyperparameter sweep, our ESN achieves an NRMSE = 0.028, correlation $r = 0.9996$, $R^2 = 0.9992$, and MAE = 0.0039 on the Mackey-Glass dataset. Demonstrate a successful ESN process.

6.2 Temporal Prediction Analysis

As seen in figure 2, the predictions exactly track the target with minimal deviation, demonstrating that the reservoir's high dimensional state trajectory successfully encodes the system's chaotic attractor structure. Further, the near perfect temporal alignment in sequencing confirms that the linear readout from the learned reservoir features are sufficient in reconstructing the complex nonlinear dynamics.

6.3 Prediction Accuracy

Figure 3 reveals that the reservoir perfectly tracks the target, which is an identity. The linear readout thus successfully pulls the predictions without systematic bias or information loss.

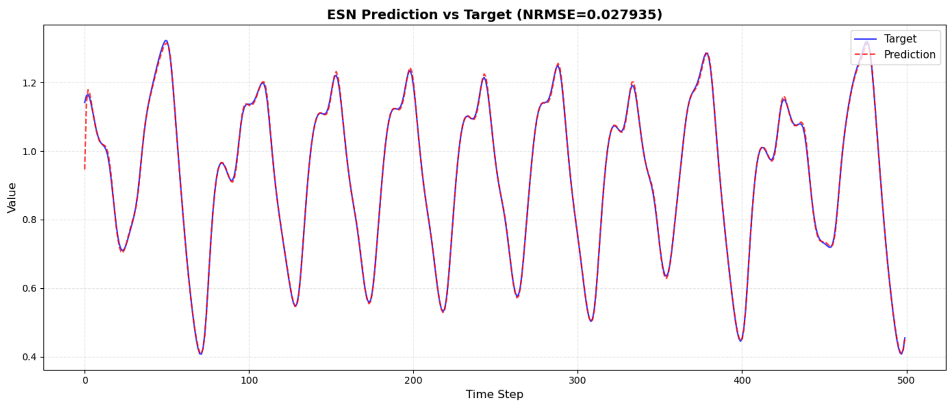

Figure 2: ESN predictions are in red and perfectly track the ground truth in blue over 500 timesteps.

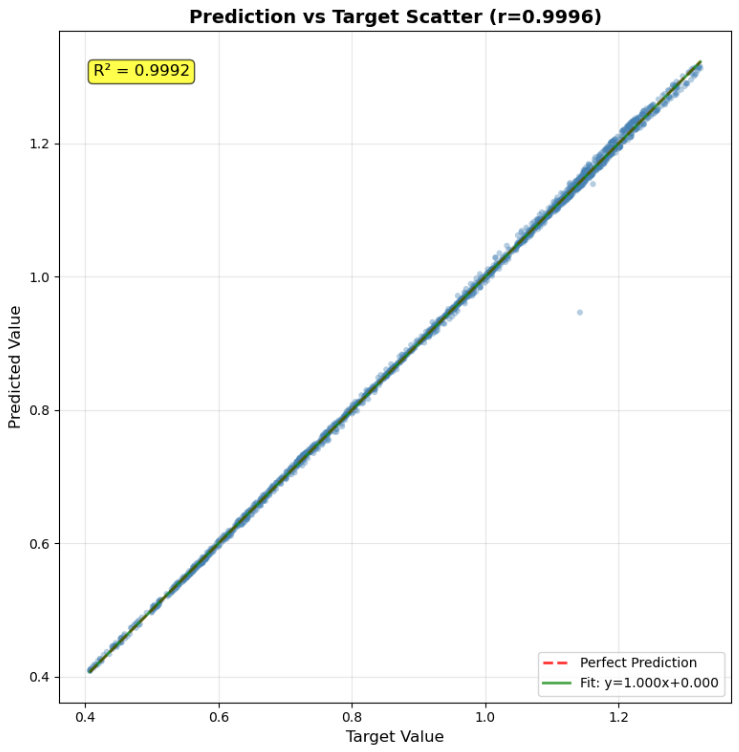

Figure 3: Output prediction compared to scattered targets resulting in an $r = 0.9996$, and $R^2 = 0.9992$.

6.4 Reservoir Internal Dynamics

Heterogeneous activation patterns in figure 4 demonstrates some form of specialization emerging from random connectivity. The neurons develop specialized patterns at multiple timescales which collectively, span a rich feature space. The diverse nature of the neuron oscillations are what enables the linear separability of the target signal from reservoir states, or rather the diverse patterns are visualizations of these underlying dynamics.

6.5 Reservoir Weight Properties

The eigenvalue spectrum in figure 5, the bottom right, confirms that $\rho_{\text{actual}} = 0.95 < 1$, which satisfies the necessity for ESP. Combined with the sparsity, shown from the top right graph, sitting

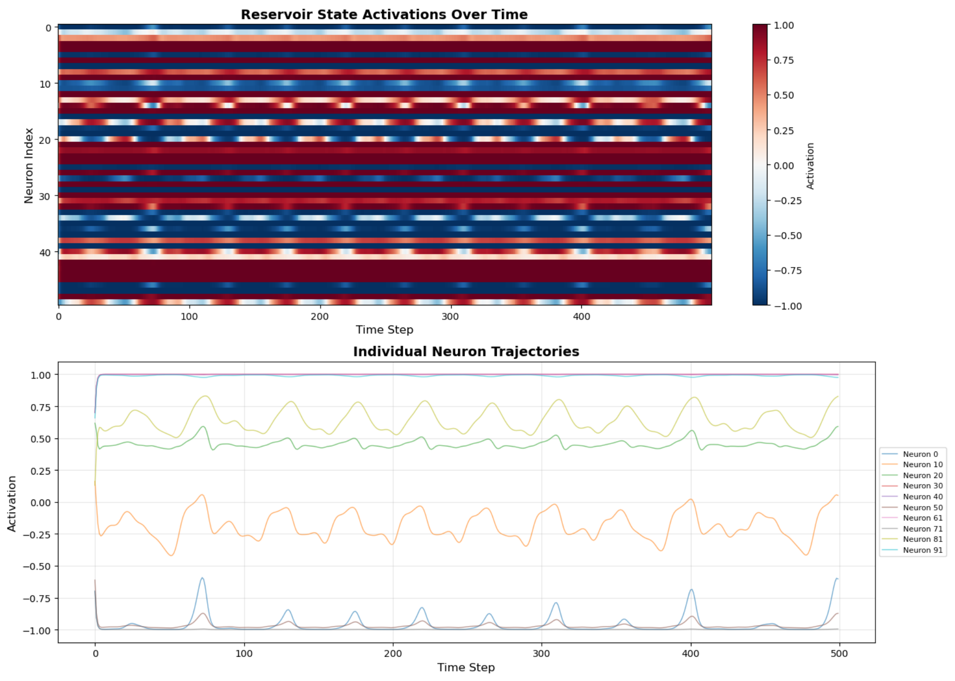

Figure 4: A heatmap reservoir neurons activations. We display the top 50 neurons in the heatmap for clarity. In the bottom graph we visualize the individual trajectories of the top 10 neurons which illustrates some interesting dynamics including some undulated waves.

around 10%, and the uniformly distributed weights from the bottom left graph, the we show the ESN is work as intended. This ESN implementation is thus a contractive high dimensional nonlinear mapping that preserves input dynamics in a temporal structure.

7 Discussion

The beauty of ESNs lie in their differences from RNNs. Where traditional RNNs lie on optimizing the recurrent weights through BPTT, ESNs fix the recurrent dynamics entirely and train only the linear readout. This design decision relies on a profound insight, that a sufficiently large random recurrent network with contractivity $\rho < 1$ naturally generates a rich and expressive feature space where complex nonlinear dynamics become linearly separable. Through all the other design decisions and implementations like spectral radius, leak rate, and sparsity, we develop a computational efficient for chaotic time series prediction without gradient based optimization.

However, as with all modeling methods, there must be a tradeoff. ESNs tradeoff efficiency for expressivity. The fixed reservoir can not adapt to exploit any task specific structure, often requiring more neurons than an RNN for similar performance. However, ESNs can learn these specific dynamics with far less computation. The trick of using readouts to express the dynamics formed by the reservoirs state trajectories, becomes a critical component that simplifies much of the complexities in RNNs. This ESN approach has significant overlaps to many other methods and topics going as far as providing insight into modeling a critical understanding of biological circuits. Future work may utilize ESNs in hierarchical architectures operating at multiple timescales, as initialization strategies for traditional RNNs, and more.

8 Conclusion

In this work, we explored and formalized ESNs and the fundamental property making them work, ESP. Leveraging the fixed randomness of the reservoir and restricting training to a linear readout ESNs perform learning over a rich dynamical reservoir capturing dynamics through subnetworks, effectively inversing the learning process of RNNs. This allows them to have their single forward

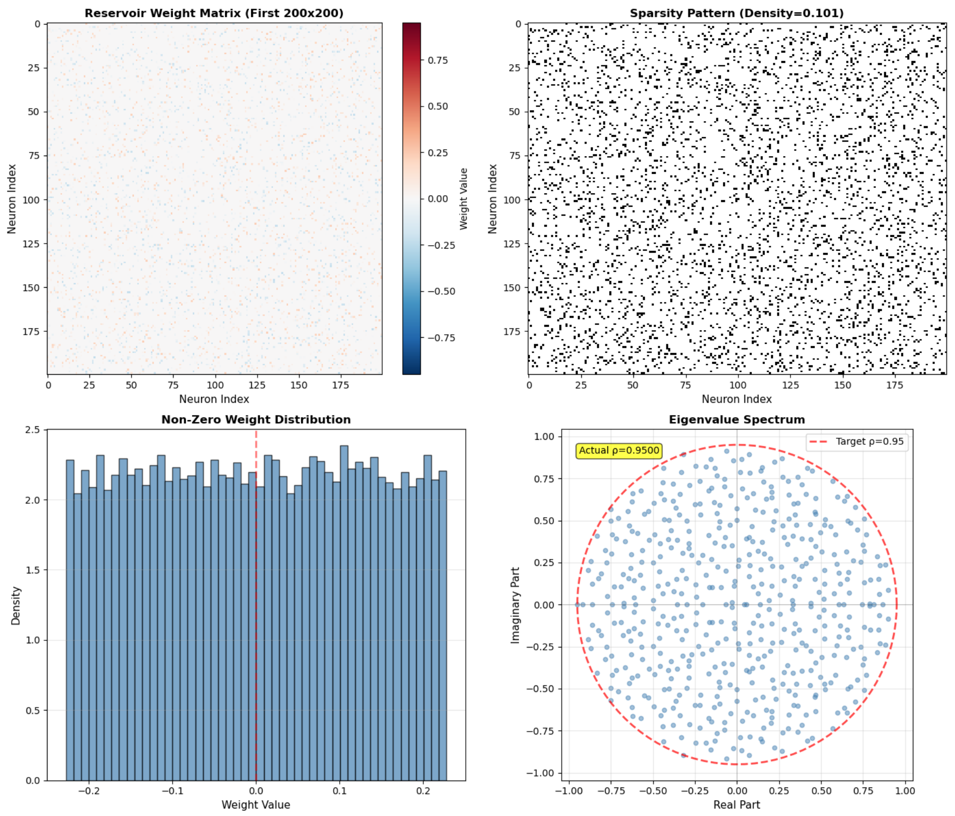

Figure 5: Reservoir weight analysis: values, sparsity, distribution, and eigenvalue spectrum.

pass efficiency. In our implementation we achieve strong performance as demonstrated by our results. Demonstrating that reservoir computing, and specifically ESNs, can serve as powerful temporal feature extractors. Together, the components that comprise the design of ESNs, namely spectral radius, sparse connectivity, and leak rate create a system that is on the edge of chaos. The balance between sensitivity and stability enables the reservoir to naturally create complex temporal representations in the reservoirs states, which is why ESNs achieve strong performance without expensive optimization strategies.

References

[1] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks – with an erratum note. Technical report, Fraunhofer Institute for Autonomous Intelligent Systems, January 2010. Corrected version of GMD Report 148 (2001).

[2] Mantas Lukoševičius. A practical guide to applying echo state networks. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 659–686. Springer, Berlin, Heidelberg, 2012.

[3] A. J. Robinson and Frank Fallside. The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Engineering Department, Cambridge University, Cambridge, UK, 1987.