Ordinal Partitions as a Predictive Feature for Echo State Networks

by

Hector Morlet

BSc

Supervisors: Professor Michael Small

Doctor Eugene Tan

This thesis is presented for the partial requirements of the degree of Bachelor of Science with honours of the University of Western Australia.

May 16, 2025

Abstract

An echo state network (ESN) is a recurrent neural network whose fixed internal network make it inexpensive to train, and effective for nonlinear time series forecasting. Ordinal partition analysis is a symbolic technique for characterising time series data that encodes the rank order of successive observations, yielding a scale invariant representation of time series that is robust to noise. This thesis explores whether ordinal partitions can be used explicitly as features within ESN architectures to extend their forecast horizon and stability. Two models are developed: (1) The Ordinal partition Readout Switching ESN (ORSESN) keeps a single reservoir but fits a separate readout vector for each observed ordinal pattern. (2) The Ordinal Partition ESN (OPESN) partitions the reservoir itself into sub-reservoirs associated with individual ordinal patterns and routes the input accordingly.

The proposed models are evaluated against a standard ESN on the Lorenz, Rössler, and Mackey-Glass systems, using both iterative and direct multi-step prediction tasks with additive Gaussian noise. The ORSESN consistently achieves lower root mean squared error (RMSE) than the baseline for prediction horizons greater than approximately five steps. These improvements are robust to noise, remaining effective even when the noise's standard deviation is as large as that of the signal. The OPESN provides more modest improvements and demonstrates notable predictive performance on the Mackey-Glass time series.

The results indicate that integrating ordinal partitions through readout switching can extend the practical forecast horizon of ESNs without increasing training cost, whereas structural partitioning of the reservoir offers limited benefit. Ordinal features therefore represent a viable route to stronger reservoir computing models for chaotic time series prediction.

Acknowledgements

This thesis is the result of the many patient meetings and helpful guidance provided by Eugene and Michael, and I am very grateful for their support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The prediction of time series data is a challenge found across scientific and engineering disciplines, from forecasting weather patterns and financial markets to understanding physiological signals and controlling industrial processes. Many real-world systems exhibit chaotic dynamics, making their future behaviour inherently difficult to predict. The limitations of traditional time series models in predicting chaotic dynamics has spurred the development of more adaptive and computationally efficient approaches.

1.1 Reservoir computing and echo state networks

Reservoir computing (RC) models for processing sequential data offer an attractive balance between performance and computational cost. As reviewed in Chapter 2, echo state networks (ESNs) have a large, fixed, and randomly generated recurrent neural network (RNN) known as the "reservoir." Fitting the parameters of the ESN is efficient because only the final parameters mapping the reservoir to the output (the "readout") are trained, typically through simple linear regression. By training only the readout layer, ESNs do not require the computationally expensive and often unstable process of fitting every weight in a traditional RNN, avoiding issues such as vanishing or exploding gradients [15].

The dynamics of an ESN's reservoir are designed to possess the "echo state property," which ensures that the reservoir's current state is a unique function of the input history, with the influence of past inputs gradually fading over time. This property allows the reservoir to act as a non-linear feature extractor, transforming the input time series into a high-dimensional state space where the underlying temporal patterns can be more easily discerned and predicted by the linear readout. ESNs have demonstrated considerable success in various time series prediction tasks [29], yet there remains scope for enhancing their ability to model particularly complex dynamics and to effectively incorporate data-derived features.

1.2 The potential of modularity and feature integration

To improve the predictions of ESNs, researchers have explored modular ESN architectures. As reviewed in Chapter 2, these approaches often involve dividing the reservoir into multiple sub-reservoirs or introducing novel readout mechanisms. The motivation behind modularity is to allow different components of the network to specialise in capturing different aspects or scales of the input dynamics, potentially leading to improved overall performance. Other strategies involve context-sensitive input routing to different sub-reservoirs or switching between multiple readout functions based on the input or reservoir state.

Alongside architectural innovations, the integration of informative features derived from the time series itself presents another avenue for enhancing ESN performance. If features can effectively summarise recent dynamics, their incorporation into the ESN architecture could improve its predictive capability.

1.3 Ordinal partition analysis: a novel feature for ESNs

This thesis focuses on incorporating ordinal partition analysis, a technique introduced by Bandt and Pompe (2002), as a novel feature selected for ESNs. Ordinal partition analysis, detailed in Chapter 3, transforms a time series into a sequence of discrete symbols based on the relative order of values within sliding windows of the series. Each symbol, or "ordinal pattern," represents the local rank ordering of data points, capturing the underlying temporal structure without relying on the absolute magnitudes of the values.

This method offers several advantages: it is robust to noise, invariant to monotonic transformations (making it less sensitive to scaling and offsets), and computationally efficient. Ordinal patterns and their derived statistics, such as permutation entropy, have been applied in tasks like analysis of physiological signals (e.g., EEG [20]) and change-point detection in financial time series [48]. The simplicity of ordinal patterns makes them an attractive candidate for enhancing the predictions of ESNs. Moreover the way ordinal patterns inherently partition the input space makes them particularly suitable for models employing gating or input routing mechanisms.

1.4 Problem statement and motivation

Despite the demonstrated utility of both ESNs and ordinal partitions for predictive tasks on chaotic data, the integration of ordinal partition information directly into the architectural design of ESNs for time series prediction remains unexplored. Standard ESNs process input signals without explicit guidance from such data-derived features.

This thesis investigates the hypothesis that by incorporating ordinal partition information into ESNs—either to guide the selection of a readout mechanism or to structure the reservoir and route input signals—we can develop ESN models that better represent the dynamics of chaotic time series to improve prediction accuracy and stability. The motivation is to create ESN architectures that not only perform well on prediction tasks but also leverage an interpretable, data-derived feature to potentially offer insights into the system's dynamics.

1.5 Aims and contributions

The primary aims of this research are:

1. **To design and implement novel ESN architectures** that use ordinal partition information derived from the input time series. This involves:
   - Developing an **Ordinal partition-based Readout Switching ESN (ORS-ESN)**, where different readout vectors are activated based on the ordinal symbols of the input.
   - Developing an **Ordinal Partition Echo State Network (OPESN)**, which employs ordinal partitions to route input to distinct sub-reservoirs and potentially to structure the connections between these sub-reservoirs.
2. **To rigorously evaluate the performance** of these proposed architectures against traditional ESNs for relevant prediction tasks. This evaluation will be conducted on synthetic data generated from well-known chaotic attractors (Lorenz, Rössler, and Mackey-Glass), which exhibit the nonlinear dynamics these models aim to predict.

3. **To investigate the impact of key design parameters** on the performance of the proposed models. This includes examining the influence of ordinal partition parameters (embedding dimension m, delay τ) and specific architectural choices within the ORSESN and OPESN.

The main contributions of this thesis are the development and empirical validation of these novel ESN architectures. By demonstrating the potential benefits of integrating ordinal partitions, this work aims to broaden the set of tools available for time series prediction with reservoir computers.

1.6 Thesis outline

The remainder of this thesis is structured as follows:

- **Chapter 2: Modular Echo State Networks** provides a review of the foundational concepts of reservoir computing, the properties of ESNs, and surveys existing approaches to modularity in ESNs, including hierarchical and parallel sub-reservoirs, input routing strategies, and readout switching mechanisms.
- **Chapter 3: Ordinal Partitions** details the methodology of ordinal partition analysis. It covers the generation of ordinal patterns, the selection of embedding parameters (m and τ), derived concepts such as permutation entropy and forbidden patterns, and discusses various applications of ordinal analysis for prediction tasks.
- **Chapter 4: Testing Methods** describes the experimental framework used to evaluate the proposed ESN architectures. This includes a description of the generation of training and testing data (Lorenz, Rössler, Mackey-Glass) with varying noise levels, the types of multi-step prediction tasks (iterative and direct), the testing procedure, and the primary evaluation metric (Root Mean Squared Error, RMSE).
- **Chapter 5: Echo State Network with Ordinal Partition Based Readout Switching** introduces the first proposed architecture, the ORSESN. It details its implementation, including how ordinal partitions of the input are used to select among multiple readout vectors. The chapter then presents and analyses the experimental results, comparing the ORSESN's performance to traditional ESNs across different datasets, prediction horizons, and parameter settings.
- **Chapter 6: Echo State Network with Ordinal Partition Based Sub-Reservoirs** introduces the second proposed architecture, the OPESN. This model uses ordinal

partitions to route input to specific sub-reservoirs and explores different strategies for connecting these sub-reservoirs based on ordinal transition probabilities. The implementation details and experimental results are presented and discussed.

- **Chapter 7: Conclusion** summarises the key findings of the research, discusses the implications of the results for time series prediction using ESNs, acknowledges the limitations of the current study, and proposes potential directions for future work in this area.

Chapter 2

Modular echo state networks

2.1 Reservoir computing

Reservoir computing separates computation into a fixed, randomly initialized dynamical system - the reservoir - and a trained linear readout. An input sequence drives the reservoir's state through time, and at each step the readout maps that state to a corresponding output sequence (commonly the following datapoint within the same sequence). The readout will then yield a prediction when the reservoir is driven with unseen data. Because only the readout weights are learned, training is more efficient than similar models for computing with sequential data, which often require compute-intensive parameter tuning algorithms [15]. While the reservoir is usually simulated, reservoir computing can act as a model for *in materio* computing [9], in which a reservoir can be created from any material or substrate with sufficiently rich dynamics. An early example of *in materio* computing involved using a bucket of water as a reservoir [11].

In the discrete case, the evolution of the reservoir's internal state is governed by a state evolution function f_{RC}:

$$\mathbf{s}(t+1) = f_{RC}(\mathbf{s}(t), \mathbf{x}(t)).$$

Here $\mathbf{s}(t)$ represents the internal state of the reservoir at time t and $\mathbf{x}(t)$ is the input signal at time t. The output of the reservoir can be computed by a 'readout' function of the states of the reservoir f_{out}:

$$y(t) = f_{out}(\mathbf{s}(t)).$$

Echo state networks

Introduced by Jaeger (2001), echo state networks (ESNs) are a form of reservoir computing which uses a recurrent neural network (RNN) as the reservoir [15]. RNNs were designed for sequence-to-sequence prediction [29], and are often comprised of a single 'hidden layer'

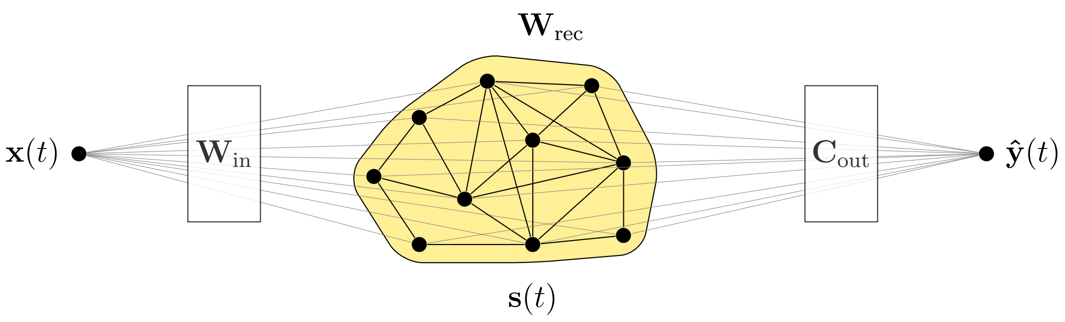


Figure 2.1: A diagram of an echo state network. The points labeled $\mathbf{x}(t)$ and $\hat{\mathbf{y}}(t)$ are the input and prediction respectively. The other points represent the states $\mathbf{s}(t)$ of the network. The lines represent the multiplications of the weighting vectors with $\mathbf{x}(t)$ and $\mathbf{s}(t)$ in equation 2.1.

of states, which undergo the state update equation

$$\mathbf{s}(t+1) = f_{act}\big(\mathbf{W}_{in}\mathbf{x}(t) + \mathbf{W}_{rec}\mathbf{s}(t) + \mathbf{W}_{bias}\big).$$

The input matrix $\mathbf{W}_{in}$ maps the input signal $\mathbf{x}(t)$ to the network states at each time step t, while the recurrence matrix $\mathbf{W}_{rec}$ governs the interactions between states of the network at each time step. The bias vector $\mathbf{W}_{bias}$ is added at each time step, and the resulting values are passed through a nonlinear activation function, f_{act}, where it is common to use the *tanh* function.

One can generate a prediction $\hat{\mathbf{y}}(t)$ at time t by multiplying the scalar product of the state vector $\mathbf{s}(t)$ and the readout vector $\mathbf{C}_{out}$,

$$\hat{\mathbf{y}}(t) = \mathbf{C}_{out}\mathbf{s}(t).$$

After we have used the input time series $\mathbf{x}(t)$ to create the state $\mathbf{s}(t)$ at each time step t, it is convenient to generate predictions over the whole time series at once. By combining the states into a matrix $\mathbf{S}$ we can derive all predictions $\hat{\mathbf{y}}(t)$ combined into a single vector $\hat{\mathbf{Y}}$:

$$\hat{\mathbf{Y}} = \mathbf{C}_{out}\mathbf{S}.$$

A diagram of an ESN can be found in Figure 2.1.

In conventional RNNs, the parameters $\mathbf{W}_{in}$, $\mathbf{W}_{rec}$, and $\mathbf{C}_{out}$ are typically optimised during training; however, for echo state networks, this is not the case. ESNs will randomly instantiate $\mathbf{W}_{in}$ and $\mathbf{W}_{rec}$ and keep them fixed, while simply fitting the readout vector $\mathbf{C}_{out}$. One could do so using ordinary least squares regression, where the coefficients of $\mathbf{C}_{out}$ are chosen to minimise the sum of squared errors

$$\min_{\mathbf{C}_{out}} \left\|\mathbf{S}\,\mathbf{C}_{out} - \mathbf{Y}\right\|_2^2,$$

however it is beneficial to use ridge regression instead [28], augmenting the objective with an ℓ_2 penalty on the coefficients yielding the loss function

$$\min_{\mathbf{C}_{out}} \left\|\mathbf{S}\,\mathbf{C}_{out} - \mathbf{Y}\right\|_2^2 \;+\; \beta \left\|\mathbf{C}_{out}\right\|_2^2.$$

This regularises the readout vector, making the predictions stable for unseen reservoir states and preventing numerical instability. The regularisation parameter β controls the penalty imposed on large-magnitude coefficients in the output matrix $\mathbf{C}_{out}$ [28]. Because the loss function is strictly convex, there is a closed-form solution:

$$\mathbf{C}_{out}^T = (\mathbf{S}^T\mathbf{S} + \beta\mathbf{I})\;\mathbf{S}^T\mathbf{Y}.$$

It is generally fast to find the solution to this linear equation and it yields a readout that generalises stably to unseen reservoir states.

The bias vector $\mathbf{W}_{bias}$ and input weight matrix $\mathbf{W}_{in}$ are simply instantiated with values drawn from a random distribution such as a Normal distribution. The recurrence matrix $\mathbf{W}_{rec}$ is generated using a random graph generation algorithm such as an Erdős-Rényi sparsely connected random graph with nodes k and average degree d, producing a $k \times k$ matrix. To rescale the magnitudes of the recurrence matrix based on the 'spectral radius' parameter ρ, one would find the maximum absolute eigenvalue of the matrix λ_{max}, and scale the matrix like so:

$$\mathbf{W}_{rec} \leftarrow \mathbf{W}_{rec}\frac{\rho}{\lambda_{max}}.$$

The states $\mathbf{s}(0)$ are initialised as a vector of values drawn from a random distribution such as the normal distribution $\mathcal{N}(0,1)$.

Computational efficiency

Increasing the number of states (k) in a traditional RNN significantly escalates the computational load during training due to the quadratic increase in the number of parameters that must be optimised. In contrast, ESNs avoid this issue as the vast majority of their parameters remain fixed. Thus, ESNs have a computational efficiency during parameter optimisation, although this advantage does not extend to the inference phase, when predictions are generated.

Fitting the parameters of an RNN requires accounting for errors propagating through time, often using the Backpropagation Through Time algorithm [29]. However, this training method is costly and the successive multiplication of extreme numbers (large and small) causes the gradients calculated through this algorithm to either 'vanish' or 'explode' [29]. While some RNNs such as Long Short-Term Memory (LSTM) networks encode a memory buffer to mitigate this problem [13], reservoir computers offer another alternative. Since ESNs avoid fitting the recurrence matrix and the input vector, they can take advantage of a large number and depth of parameters without suffering from the issues of vanishing gradients. While ESNs have the added benefit of requiring far less training computation than traditional RNNs, they require some care to be taken in establishing the properties of the reservoir.

The echo state property

The defining characteristic of reservoir computers is the *echo state property.* This property ensures that, given a sufficiently long input sequence, the current state of the reservoir is determined uniquely by the intervening input and is independent of its initial state [15]. In other words, the further past inputs are in time from the current state, the less they impact the current state. More formally, a network possesses echo states if its states are uniquely determined by the input sequence $\mathbf{x}(t)$ for some sufficiently large t, i.e., $\mathbf{s}(t) = \mathbf{s}'(t)$ for any initial states $\mathbf{s}(0)$ and $\mathbf{s}'(0)$, where $\mathbf{s}(t)$ represents the reservoir state at time t and the states are evolved with the same weights $\mathbf{W}_{in}$ and $\mathbf{W}_{rec}$.

For a network to exhibit the echo state property, it must satisfy three key conditions [6]:

- **Uniform State Contraction:** The network must be contractive, meaning that differences in initial states decay over time.
- **State Forgetting:** The network must 'forget' its initial state after a finite amount of time, ensuring that only recent inputs affect the current state.
- **Fading Memory:** The influence of past inputs diminishes over time, making the network's state primarily a function of recent inputs.

With each condition satisfied, the ESN will exhibit the echo state property and thus holds a limited *memory capacity.* Each state of the network can be viewed as a function of a finite sequence of prior inputs:

$$\begin{aligned}\mathbf{s}(t) &= \mathcal{F}(\mathbf{s}(t-1), \mathbf{s}(t-2), ...\mathbf{s}(0))\\ &\approx \mathcal{F}(\mathbf{s}(t-1), \mathbf{s}(t-2), ...\mathbf{s}(t-\tau_{MC})),\end{aligned}$$

where the number of time steps τ_{MC} corresponds to some memory capacity of the network. The states that are beyond τ_{MC} steps in the past practically do not contribute to $\mathbf{s}(t)$. The memory capacity for a given time delay τ_{max} can be quantified by the following method as proposed by Jaeger (2001) [15]:

$$MC(\tau_{max}) = \sum_{\tau=1}^{\tau_{max}} Corr\left[\left\{\mathbf{x}(n-\tau), \hat{\mathbf{y}}_{\tau}(n)\right\}_{n=\tau}^{n=T}\right]^2.$$

The input sequence $\mathbf{x}$ is I.I.D. noise, and the series $\hat{\mathbf{y}}_{\tau}$ refers to the outputs of a reservoir computer trained to reproduce the input sequence after τ time steps have elapsed. We are calculating the Pearson correlation of these 'remembered' values with the input values over the whole sequence, where T is the final time step of the sequence. To measure the length of the reservoir computer's memory τ_{MC}, we should find the lowest value of τ_{max} for which the memory capacity has approximately converged. This value will represent how many time steps in the past a reservoir computer can recall patternless inputs. In later publications, the calculation of memory capacity is performed by summing the correlations directly, without squaring them first [29].

In most prediction tasks, the most recent inputs are more relevant than those received further into the past. This is the intuitive justification for why the echo state property is important to creating a useful model. With a large enough memory capacity, the echo state property ensures that the reservoir provides a rich, yet stable and decaying, representation of the input signal over time.

By scaling the recurrence matrix $\mathbf{W}_{rec}$ so that its spectral radius equals our parameter ρ, we are controlling the effect of the states $s(t)$ on the states $s(t+1)$ relative to the input signal $x(t+1)$. By scaling $\mathbf{W}_{rec}$ by ρ, we are regulating the reservoir's decay rate of past activations and thus its memory capacity. When $0 < \rho < 1$, the past states are propagated through the network but the current input is still the dominant effect on the activation states. As ρ increases, the reservoir begins to exhibit chaotic behavior. As we increase $\rho > 1$, the ESN approaches a tipping point (often referred to as the 'edge of chaos') beyond which the ESN memory dominates the input signal. With values $\rho >> 1$, the ESN becomes unstable and uninformative as the input signal has little effect on its dynamics. A commonly used heuristic is to set $\rho \approx 1$ to balance the memory and the stability of the ESN and testing small adjustments (e.g. up and down by 0.05) on validation data [29] [28]. Generally, tasks requiring a longer memory will require a larger ρ.

2.2 Mixture-of-experts

The concept of the 'Adaptive Mixture of Local Experts' was originally introduced by Jacobs et al. in 1991 [14]. This approach involves training several neural network modules, referred to as 'expert' modules, alongside a 'gating network' that uses a softmax function to weight the outputs of these experts. The final output of the system is a weighted combination of the outputs from the expert modules. A general diagram of this model is shown in Figure 2.2. Both the experts and the gating function are trained concurrently on a supervised task, allowing the system to automatically partition the problem among the experts. This method was shown to achieve better generalization compared to a monolithic network [14].

Mixture of experts (MOEs) can be viewed as a specialised form of ensemble learning [36]. Ensemble learning methods such as bagging, boosting and stacking also combine the outputs of multiple models to improve the prediction of any single model. However the MOE does so with an adaptive and input dependent weighting supplied by the gating function rather than fixed or globally learned weights. This conditional routing lets the MOE weight experts more in the areas of the data where they are most competent. Bias-variance-covariance analysis shows that the selection of local experts trims bias and covariance, whereas the averaging of experts suppresses variance, and the best generalisation arises when the gate balances the two strategies [36]. Jointly training the gating function and the experts encourages negatively correlated errors among the experts, whose mutual cancellation reduces ensemble error more effectively than uncorrelated (but unbiased) predictors [36].

Furthermore, MOEs may be seen on a spectrum that spans deterministic piece-wise models, rule-based gating schemes, and fully adaptive probabilistic gates. At one extreme, the gate is specified explicitly by decision rules and each expert is tied to a fixed partition of the input space so that the model behaves as a piece-wise approximator [36]. At the other extreme, the gate and the experts are trained jointly, where the gate supplies smooth, input-dependent weights that encourage negatively correlated errors and an arbitrarily complex partitioning of input space.

While fully adaptive gates trained alongside the experts provide a flexible and generally effective method of partitioning the input space, rule based gating allows the modeller to encode a priori knowledge of the time series directly into the gate. This prior decomposition avoids the common difficulty of fitting the gating function on complex and nested boundaries between experts by tying one expert to each predefined region [36]. Because the gate's decision logic is deterministic and transparent, the mapping from the input region to the expert remains human readable. A practical bonus is computational: once the split is set, every expert can be trained independently and in parallel. For ESNs this means the model as a whole still enjoys the original efficient training afforded by a handful of ridge regressions.

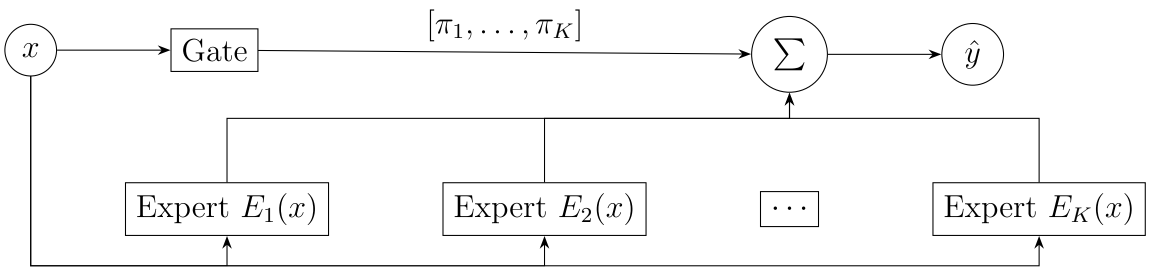


Figure 2.2: A general diagram of a MOE model. Each expert $E_i(x)$ is weighted by π_i for a given input x for $i \in 1...K$.

The structure of this approach can be visualised as a tree, comprising three main components:

- multiple expert models, which can be either regression functions or classifiers;
- a gate that creates partitions of the input space, defining regions for each individual expert; and
- a probabilistic model that combines the outputs of the experts and the gate.

This model introduces three important properties: it allows individual experts to specialize in smaller parts of the larger problem, uses soft partitions of the data, and enables splits of the input space along hyperplanes at arbitrary orientations [18]. This approach is particularly beneficial for handling nonstationary data, where the time series switch their dynamics in different regions of the input space [47].

Jordan (1994) formalised the 'Hierarchical Mixture of Experts' (HME) model, and provided an Expectation-Maximisation (EM) algorithm for its training [18]. The EM algorithm is an iterative two-step procedure for fitting models with latent variables, and alternates between the Expectation (E) step and Maximisation (M) step. For MOE models, the E step computes a set of normalised posterior weights for each data point that reflect how well each expert (given the current gate and expert models) explains that data point. Then the M step updates the gate to predict those posterior weights and retrains each expert, weighting the importance of each data point to each expert by the posterior weights. Iterating these steps leads to convergence: the gating network assigns higher weights to the most accurate experts for each input and each expert's parameters are optimised with respect to the data weighted by those assignments. The HME model allows gating to be organised in a tree like hierarchy of multiple levels of experts, enabling a more complex partitioning of the input space and can lead to improved performance on tasks where the data exhibits hierarchical or multi-scale characteristics [18].

MOE models have been applied to various types of data, including the task of predicting time series. Tabuse, Kinouchi & Hagiwara (1997) proposed a novel approach that uses a mixture of RNNs specifically for time series processing [43]. This method uses the Expectation-Maximisation (EM) algorithm for training, and has been shown to result in faster training times and improved performance on benchmark sequences when compared to a single RNN. A resurgence of interest in the mixture of RNN Experts was marked by a notable paper by Shazeer et al. (2017) titled "Outrageously Large Neural Networks" [40], which demonstrated the effectiveness of using a mixture of experts within a stacked Long Short-Term Memory (LSTM) network - a type of RNN - for machine translation tasks.

In the context of ESNs, Babinec and Pospíchal (2009) investigated the concept of a modular ESN. This approach involves using several unconnected ESN reservoirs in parallel as expert models, each possessing different internal dynamics and memory capacities [3]. The different dynamics were created by varying the spectral radii and internal connection patterns of the networks. A separate gating network - another ESN - was trained to determine the appropriate weighting of the different reservoir outputs. The modular ESN was tested on a chaotic laser intensity time series, where the dynamics exhibited both fast and slow oscillatory components. The results indicated that the ESN MOE outperformed a single ESN, demonstrating that the gating network could dynamically assign higher weight to the expert whose "memory length" best matched the current behaviour of the series, thus dynamically adapting the model's time scale.

2.3 Modular echo state networks

Since they were introduced by Jaeger (2001), a taxonomy of different ESNs has grown, from SHESN [10], HESN [17], and ConvESN [32] to Deep-ESN [12], Mod-DeepESN [7], GD-ESN [23], gdESN [45], MH-ESN [30], IR-ESN [2] and MSSESN [33]. Many of these proposals use multiple reservoirs - or ‘sub-reservoirs’ - in their design, whether they are arranged hierarchically, in parallel or in some other configuration.

Jaeger (2007) himself proposed the concept of a ‘Dynamic Feature Discoverer (DFD)’, in which multiple ESNs are components of a larger system [16]. Jaeger arranged multiple ESN modules in a hierarchy, each receiving input from lower level ESNs, with the lowest level ESN receiving the standard input $\mathbf{x}(t)$. This allowed the DFD to capture latent features on different temporal or spatial scales, where the higher level ESNs represent a longer time scale than the lower level ESNs. Each ESN layer produces high dimensional features at each time step, and then finds the scalar multiple between its predicted features and the prediction of the layer above. The outputs are combined and passed down in this way until the final layer produces the overall prediction for that time step.

Modular echo state networks have since been included in larger ensembles of models, for example Ma et al. (2021) proposed a multi-reservoir solution where the reservoirs run in parallel at different time scales [32]. They proposed using convolutional neural networks to analyse the states of the reservoirs and extract relevant features across time. They demonstrated that the ConvESN could successfully perform a variety of classification tasks for sequence based data.

Restricted echo state networks

While there are many ‘modular’ ESNs, Wringe et al. (2024) describes a further niche of ‘restricted’ ESNs [46]. Restricted ESNs have idiosyncrasies within their internal structure, and are not simply assemblies of traditional ESNs. Restricted ESNs divide their internal reservoir states into several smaller sub-reservoir states simply by restricting certain connections within the recurrence matrix $\mathbf{W}_{rec}$. There may also be other deviations from the traditional ESN, but the restriction of the recurrence weights is what makes them modular. The state vector can be seen as the concatenation of sub-reservoir state vectors,

$$\mathbf{s} = (\mathbf{s}_1, \mathbf{s}_2 ... \mathbf{s}_n)$$

where $\mathbf{s}_i$ represents the states within the sub-reservoir i. Thus the recurrence matrix $\mathbf{W}_{rec}$ can be defined in terms of the recurrence matrices of the sub-reservoirs $\mathbf{W}_i$ and the connections between these submatrices $\mathbf{W}_{i,j}$:

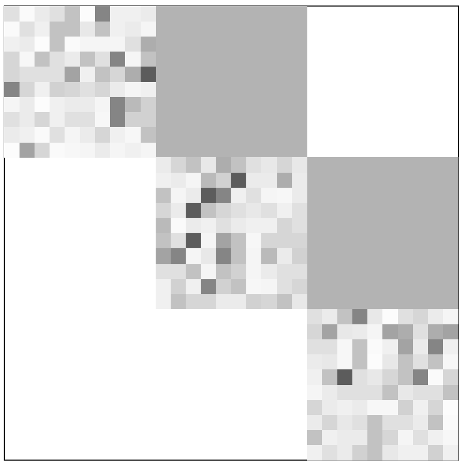
$$\mathbf{W}_{rec} = \begin{pmatrix} \mathbf{W}_1 & \mathbf{W}_{1,2} & \cdots & \mathbf{W}_{1,n} \\ \mathbf{W}_{2,1} & \mathbf{W}_2 & \cdots & \mathbf{W}_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{W}_{n,1} & \mathbf{W}_{n,2} & \cdots & \mathbf{W}_n \end{pmatrix}.$$

An early example of a restricted ESN is the Scale-Free Highly Clustered ESN (SHESN) [10], where each sub-reservoir consists of a number of local neurons clustered around a backbone neuron. These backbone neurons connect the sub-reservoirs to one another. The SHESN outperformed the traditional ESN at approximating complex nonlinear dynamics (the Mackey-Glass system) and displayed an enhanced echo state property [10]. Another architecture, the Hierarchically Clustered ESN (HESN), built upon the SHESN by making the sub-reservoirs randomly connected rather than fully connected and allowing for multiple backbone neurons per sub-reservoir [17].

Malik et al. (2017) proposed the MultiLayered Echo State Machine (ML-ESM), where the reservoir is arranged into sequential sub-reservoirs [35]. Each sub-reservoir is fully connected to neighbouring sub-reservoirs using fixed weights within the reservoir's recurrence matrix - visualised in Figure 2.3a They demonstrated that the ML-ESM outperforms traditional ESNs in tasks requiring the modelling of complex temporal dependencies.

Gallicchio and Micheli (2017) introduced the Deep-ESN, in which reservoirs are arranged sequentially and inputs are only sent to the first sub-reservoir [12]. The sub-reservoirs are sparsely connected via the recurrence matrix, but only connected in one direction to the following sub-reservoir - visualised in Figure 2.3b. They showed that the Deep-ESN allowed for an effective diversification of temporal representations between the sub-reservoirs. The Deep-ESN has been successfully applied to the diagnosis of machinery faults, showing superior results to several other deep learning models [26]. The Deep-ESN was further expanded with other variations, such as feeding the input to all sub-reservoirs (Deep-ESN IA), adding projection and encoding layers between the sub-reservoirs [31], incorporating the Deep-ESN into modular architectures (Mod-DeepESN) [7] or the Growing Deep ESN (GD-ESN), which incrementally adds sub-reservoirs during training to find the optimal depth to capture temporal dependencies in time series data [23].

To compare against their Deep-ESN, Gallicchio and Micheli (2017) also introduced the Grouped-ESN in which the input signal is fed into several unconnected sub-reservoirs [12]. The sub-reservoirs all contribute to the output, which is generated by a readout vector


(a) $\mathbf{W}_{rec}$ of a ML-ESM

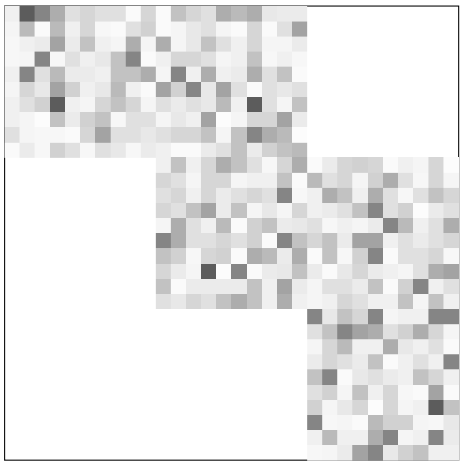

(b) $\mathbf{W}_{rec}$ of a Deep-ESN

Figure 2.3: $\mathbf{W}_{rec}$ of ESN variants with sequential, feed-forward sub-reservoirs.

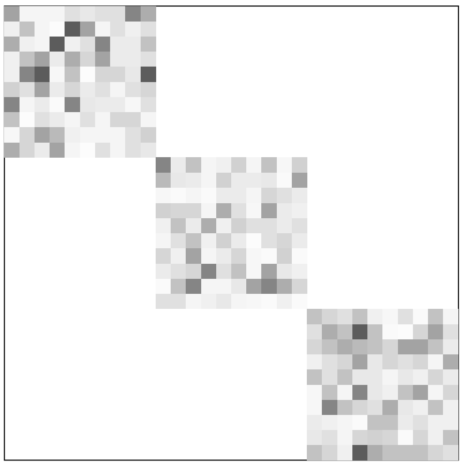
multiplied by the state of all of the sub-reservoirs. Its recurrence matrix is visualised in Figure 2.4a. They created the Grouped-ESN to isolate the effect of the sequential stacking of sub-reservoirs in their Deep-ESN, and gauge the relative effect of splitting the reservoir into sub-reservoirs. Wcisło & Czech (2021) built upon this idea by stacking multiple layers of parallel sub-reservoirs, forming what they called a Grouped Deep ESN (gdESN) [45]. Each sub-reservoir passes its states to its following sub-reservoir in the next layer. Where the regular ESN (with no bias vector) would update its state like so:

$$\mathbf{s}(t+1) = f_{act}(\mathbf{W}_{in}\mathbf{x}(t) + \mathbf{W}_{rec}\mathbf{s}(t)).$$

Any sub-reservoir after the first sub-reservoir $\mathbf{s}_{i+1}(t)$ in a gdESN would update its states using the previous layer's states $\mathbf{s}_i(t)$ rather than the input $\mathbf{x}(t)$:

$$\mathbf{s}^{(i+1)}(t+1) = f_{act}(\mathbf{W}_{in}^{(i)}\mathbf{s}^{(i)}(t) + \mathbf{W}_{rec}^{(i)}\mathbf{s}^{(i+1)}(t)).$$

The nonlinearity introduced by the f_{act} means the structure of gdESN can't be represented simply by restricting the recurrence weight matrix $\mathbf{W}_{rec}$, but rather as an ensemble of restricted ESNs. Another conception of the multi-reservoir ESN is the 'Multiple-Reservoir Hierarchical ESN' (MH-ESN) proposed by Lun et al. (2023), which can be represented by restrictions to $\mathbf{W}_{rec}$ [30]. The 'layers' of the MH-ESN are not ordered sequentially, but instead represent groups of sub-reservoirs with similar parameter settings (number of nodes and sparsity). Each sub-reservoir in the MH-ESN has a single 'principal' neuron, representing the average states of the neurons in that sub-reservoir. A visualisation of its recurrence matrix can be found in Figure 2.4b. MH-ESN differs to


(a) $\mathbf{W}_{rec}$ of a Grouped-ESN. The sub-reservoirs are not connected to one another, but internally are sparsely connected.

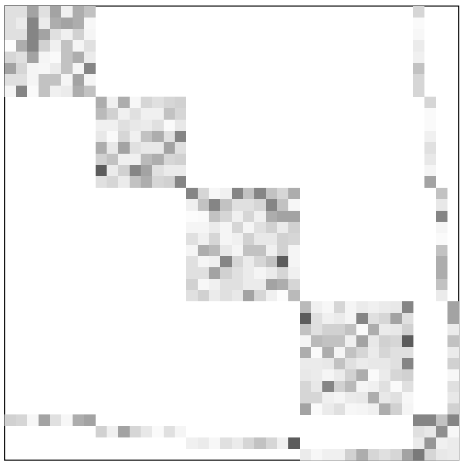

(b) $\mathbf{W}_{rec}$ of the MH-ESN. The principal node weights are in the bottom right. The sub-reservoirs are only connected to one another by their principal nodes.

Figure 2.4: $\mathbf{W}_{rec}$ of ESN variants with parallel sub-reservoirs.

the Grouped-ESN because it sparsely connects the sub-reservoirs within each layer only using these principal neurons, where all other neurons are unconnected.

Throughout much of this literature, the authors stipulate that the advantage of multiple sub-reservoirs is to have a broad range of dynamics for the same input sequence, allowing the ESN to represent a broader and richer set of temporal features. In some cases, this is achieved by the **structure of the network**, as in the case of Jaeger's DFD, in which the input cascades through the different modules over time. This is similarly the case for the Deep-ESN, where each sub-reservoir passed the input signal along to the next sub-reservoir over time. The advantage afforded by a broad range of dynamics is achieved more explicitly by Babinec and Pospíchal's modular ESN and for the Grouped-ESN, in which each sub-reservoir is **instantiated with different parameters** (for example, the spectral radius) to create differing properties such as memory capacity. Other modular ESNs use sequence resampling of the input signal (skipping, overlapping or segmenting) to supply each sub-reservoir with **different variations of the input time series** [24]. The different reservoirs are able to respond at multiple timescales for fast or slow dynamics and thus represent phenomena that requires more or less memory.

Context sensitive input routing

By changing the structure of the recurrence matrix $\mathbf{W}_{rec}$, Dale (2018) introduced a 'Reservoir of Reservoirs' architecture, comprising multiple sparsely connected sub-reservoirs linked by extremely sparse (≈1%) interconnections [8]. They evaluated two routing schemes: directing the input exclusively to a single sub-reservoir versus 'broadcasting' it to all sub-reservoirs. On the NARMA benchmark, neither input routing scheme improved upon the traditional ESN, but the author showed that these architectures had a better transfer performance when retrained on unseen tasks (Santa Fe laser, noisy Hénon map). These ideas show the potential of input routing policies to shape reservoir performance and motivate the exploration of more flexible, context-sensitive input routing architectures.

It is possible to route the input signal to different sub-reservoirs depending on contextual information. Argentieri et al. (2022) proposed the Input Routed Echo State Network (IR-ESNs) to improve the prediction of ESNs on multi-dimensional signals [2]. When the input signal $\mathbf{x}(t)$ is a vector rather than a scalar, traditional ESNs process all input dimensions within a single reservoir. Argentieri et al. argued that this can lead to the mixing of distinct input features and potentially obscure the individual characteristics of each input dimension.

To overcome this limitation, the authors proposed an architecture that routes different dimensions of the input signal to dedicated sub-reservoirs within a multi-reservoir model. Each sub-reservoir receives only a specific component of the input and so develops dynamics reflecting those components separately. Additionally, the sub-reservoirs are interconnected, enabling the modelling of interactions between different input dimensions as required by the task. The authors evaluated their IR-ESNs on both synthetic and real-world time series classification problems. They demonstrated that IR-ESNs outperform standard ESNs in predicting multi-dimensional signals, showing that the proposed routing mechanism leverages the distinct characteristics of each input dimension.

2.4 Readout switching

While there are many examples of ESNs with a partitioning of the reservoir, researchers have also explored ESNs where partitioning focuses on the readout function. For example, Ma & Chen (2013) conceived of a 'Modular State Space ESN (MSSESN)' by dividing the reservoir in the high-dimensional coordinate space of its states (state space), rather than dividing the internal weight matrix of the reservoir [33]. They achieved this by fitting a piecewise readout function, with a separate linear readout for each module. They partition the state space by partitioning predictions derived from the reservoir, arguing that the Euclidean norm of any two states is likely proportional to the Euclidean norm of any two predictions. The state space is divided by first fitting an ordinary linear readout on the reservoir, $\mathbf{C}_{out}$, and then performing an equal partitioning of the predictions generated from this linear readout such that $h_i \leqslant \mathbf{C}_{out}\mathbf{s}(t) < h_{i+1}$ where h_i and h_{i+1} are the bounds of each partition. The partitioning of the predictions gives a partitioning of their state space values, since any unseen reservoir state can be allocated a partition based on its evaluation of $\mathbf{C}_{out}\mathbf{s}(t)$. Then a new readout $\mathbf{C}_{out}^{(k)}$ can be fitted for each partition k, providing a specific readout for each partition of state space. Thus, the piecewise linear readout function is driven by the states of the reservoir itself and was shown to improve prediction performance on non-stationary time series.

Laan & Vicente (2015) also proposed an ESN with multiple linear readout modules, where a module is chosen depending on the states of the reservoir [21]. Each module has two vectors: a readout vector and a weighting vector. The readout vector is fitted via regression while the weighting vector is randomly instantiated. Each module makes its own prediction by taking the scalar product of the readout vector and the states of the network - as a traditional ESN would. Each module also makes its weighting by taking the scalar product of its weighting vector and the states of the network. The predictions from each module are then combined in a softmaxed weighted sum using their weights. Thus the relative weights of each module change according to how the states change over time, emphasising some modules at some times and others at other times.

The architectures reviewed in this chapter illustrate the diverse ways in which modularity can be incorporated into ESNs. Components such as input routing, restricted reservoirs and readout switching offer options for creating novel model architectures. These components not only increase the variety of models we can apply to any prediction task, but also allow us to incorporate specific features of data into the architecture of ESNs. In the following chapters, we treat each of these architectural elements as potential building blocks for developing ESNs that leverage ordinal partitions as a key feature, with the aim of enhancing predictive performance.

Chapter 3

Ordinal partitions

Ordinal partition analysis, introduced by Bandt and Pompe [4], provides a method for transforming a time series into a sequence of discrete symbols based on the relative order of values within segments of the series. This technique focuses on the inherent order relations present in the data, making it invariant to monotonic transformations and less sensitive to observational noise compared to methods relying on absolute values. The core idea involves mapping sliding windows of the time series data onto *ordinal patterns*.

3.1 Ordinal partitions and their measures

Generating the ordinal partition

The process begins by choosing two key parameters: the embedding dimension $m \geqslant 2$, which sets the length of each segment (or vector) to compare, and the embedding delay $\tau \geqslant 1$, which specifies the time gap between successive points in that segment. For a time series $\{x_t\}_{t=1}^{N}$, we form overlapping vectors

$$\mathbf{x}_t = \big(x_{t-(m-1)\tau},\, \dots, x_{t-\tau},\, x_t\big), \quad t = 1 + (m-1)\tau, \dots, N-1, N$$

which we can use to create a rank vector

$$\rho_t = (\rho_{t,1}, \dots, \rho_{t,m}),$$

where $x_{t+(i-1)\tau}$ is the $\rho_{t,i}$-th largest of $\mathbf{x}_t$ for each $i \in [1, ..., m]$.

Ties are broken by giving the lower rank to the earlier index (i.e. if $x_{t+i\tau} = x_{t+j\tau}$ with $i < j$, then $\rho_{t,i} < \rho_{t,j}$). Bandt & Pompe originally introduced the method using an inverse representation, where each $\rho_{t,i}$ is the index of $\mathbf{x}_t$ at which the value has rank i. The two representations carry equivalent information for the purpose of creating an ordinal partitioning of a time series. By this definition a data point's partition is determined by its preceding points ('backward') while most definitions have it determined by its

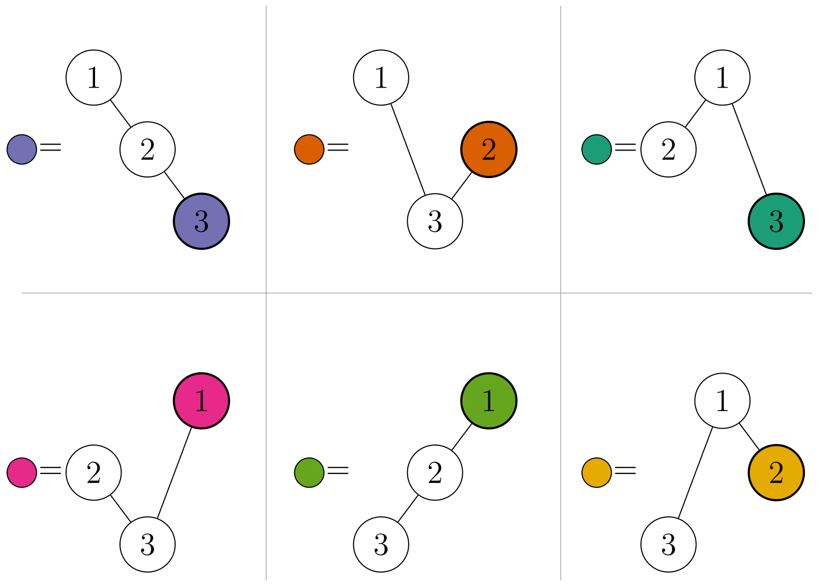


Figure 3.1: A visualisation of all 6 possible ordinal partitions when $m = 3$.

following points ('forward'). Here we have used the backward definition because each point will be used for prediction, where future data points are unknown. While the forward definition is generally assumed, some literature has used the backward definition for similar reasons [20].

For example, consider $m = 3$ and $\tau = 1$. Figure 3.1 gives a visual representation of all 6 possible rank orderings any three data points could form. If we consider a segment of a time series $(x_t, x_{t+1}, x_{t+2}) = (4, 7, 2)$ then the values ranked are $x_{t+2} \leqslant x_t \leqslant x_{t+1}$. The corresponding rankings are $(2, 1, 3)$, so the ordinal pattern is $(2, 1, 3)$. By applying this procedure across the entire time series, a sequence of ordinal patterns is generated, effectively converting the original continuous or discrete-valued series into a symbolic sequence drawn from the $m!$ possible permutations. An example of a partitioned time series is given in Figure 3.2, where each data point has been colour coded depending on its ordinal partition.

To provide a concise reference to these ordinal partitions, each partition is allocated an 'ordinal symbol'. Where we have used colours in our diagrams, one might usually use integers to refer to each ordinal partition, although the allocation of these symbols is arbitrary. Table 3.1 gives an example of how one might allocate ordinal symbols to the partitions in our $m = 3$ example. Once a rank vector ρ_t has been found for each data point in our time series $\{x_t\}_{t=1}^{N}$, allocating an ordinal symbol π_t to each rank vector will yield a time series of ordinal symbols $\Pi = (\pi_1, ..., \pi_N)$.

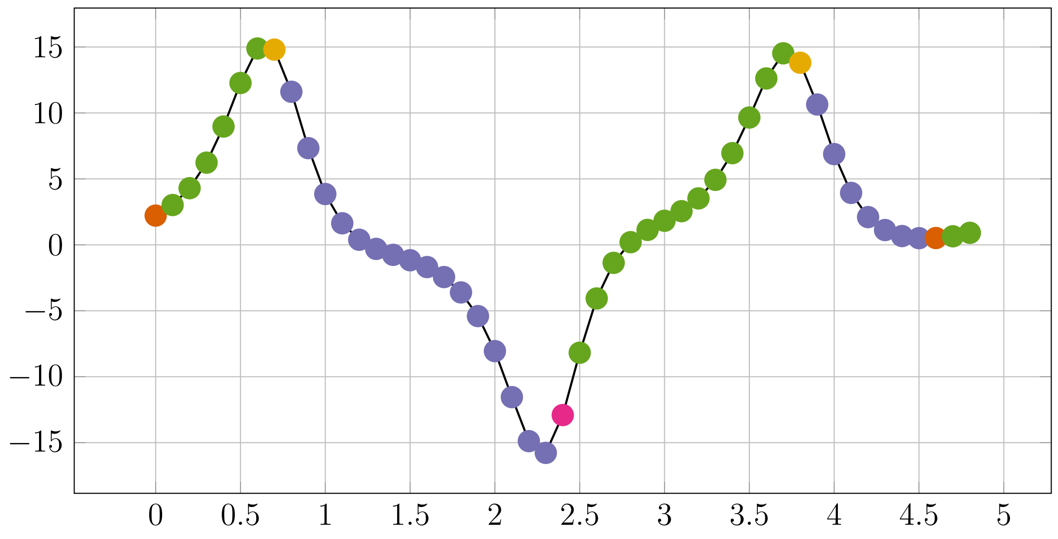


Figure 3.2: A time series with data points colour-coded by ordinal partition.

Ordinal symbol	Ordinal patterns
1	[1, 2, 3]
2	[1, 3, 2]
3	[2, 1, 3]
4	[2, 3, 1]
5	[3, 2, 1]
6	[3, 1, 2]

Table 3.1: Example: Ordinal patterns and their allocated ordinal symbols for $m = 3$.

While the standard Bandt-Pompe method uses discrete ordinal patterns, generalizations have been proposed. Zanin [49] introduced the concept of *Continuous Ordinal Patterns*, where patterns are defined in a continuous space, allowing for optimization tailored to specific problems.

Probabilities

From an ordinal symbolic sequence, a probability distribution can be estimated by calculating the relative frequency of each unique ordinal pattern π_i. This distribution, referred to as the *ordinal probability distribution*, captures essential information about the temporal structure and dynamics of the underlying system. A key measure derived from this distribution is the *Permutation Entropy* (PE), defined as the Shannon entropy of the ordinal probabilities [4]:

$$H(m, \tau) = -\sum_{i=1}^{m!} p(\pi_i) \log_2 p(\pi_i).$$

	🟣	🟠	🔵	🔴	🟢	🟡
🟣	0.7	0.1	0.1	0.05	0.05	0.0
🟠	0.0	0.0	0.0	0.0	0.9	0.1
🔵	0.7	0.1	0.1	0.05	0.05	0.0
🔴	0.0	0.0	0.0	0.0	0.9	0.1
🟢	0.0	0.0	0.0	0.0	0.9	0.1
🟡	0.7	0.05	0.05	0.05	0.05	0.1

Table 3.2: Ordinal transition probabilities for the example time series in Figure 3.2. Each entry $p(\pi_j|\pi_i)$ gives the probability of transitioning from ordinal partition π_i at time t to π_j at time $t+1$.

Permutation entropy quantifies the complexity of the time series as reflected in the diversity and predictability of its ordinal patterns [4]. Low PE values indicate a high degree of regularity, such as in periodic or highly predictable series where only a few patterns dominate. Conversely, high PE values (approaching the maximum of $\log_2(m!)$) suggest more complex, potentially stochastic-like dynamics where patterns occur with closer to uniform probability.

Beyond the static distribution of patterns, further dynamical information can be extracted by examining the transitions between consecutive ordinal patterns in the sequence Π. The *ordinal transition probabilities*, $p(\pi_j|\pi_i)$, quantify the likelihood that pattern π_j occurs at time $t+1$, given that pattern π_i occurred at time t. These probabilities are estimated by counting the frequency of adjacent pairs (π_i, π_j) in the sequence and normalising appropriately. Analysing these transitions provides insights into the temporal dependencies and correlations within the series at the level of ordinal structure, offering a higher-order characterisation compared to the simple pattern frequencies alone. The ordinal transition probabilities for our example time series can be found in Table 3.2.

Forbidden patterns

A significant aspect of ordinal analysis is the concept of *forbidden patterns*. If certain ordinal patterns out of the $m!$ possibilities never appear in the symbolic sequence generated from a sufficiently long time series, they are termed forbidden patterns [1]. The existence of forbidden patterns is a strong indicator that the underlying dynamics are not purely stochastic, but possess some deterministic structure. This is because purely random processes are expected to eventually exhibit all possible ordinal patterns (given stationarity and sufficient length). The identification of forbidden patterns has been explored in various contexts, including financial time series [48].

Choosing τ and m

The selection of τ and m is important to accurately represent the dynamics of the time series. If a τ or m is selected too small, there may be many repeated monotonic patterns and this can underestimate complexity as measured by PE [38]. If τ or m is selected too large, there may be many patterns with very few observations or a spuriously high number of forbidden patterns [38].

Common methods for selecting τ include picking the first zero of the autocorrelation, or the first minimum of average mutual information [38]. However, in this research we have performed an RMSE based search, selecting an optimal τ for the purpose of time series prediction with the proposed models. For most common purposes it is recommended to increase m until the PE stops rising [38], but for this research we have tested all practical values where $m \leqslant 4$, for reasons explained in Chapter 4.

3.2 Applications of ordinal analysis

Ordinal patterns and their derived measures have proven to be effective features in various time series analysis tasks, including classification and change-point detection.

Classification

Liu et al. demonstrated the utility of ordinal representations for classifying EEG time series to detect epileptic seizures [25], building from previous efforts that use permutation entropy to analyse EEG data [20]. They ordinally partitioned EEG signals and extracted features based on both the distribution of individual ordinal patterns and the distribution of adjacent pattern pairs (ordinal transitions). Using these features as input to a k-Nearest Neighbour (k-NN) classifier, they achieved high accuracy, sensitivity, and specificity (over 90%) on benchmark seizure datasets, outperforming several contemporary methods. These results demonstrate that ordinal patterns can be an effective feature to discriminate between physiological signals. Similarly, Boaretto et al. used ordinal-pattern probability vectors as inputs for Artificial Neural Networks (ANNs) to effectively distinguish between chaotic and stochastic time series [5]. Both of these examples demonstrate that ordinal representations are an effective feature for classification tasks.

Change point detection

The sensitivity of the ordinal pattern distribution to the underlying dynamics of a time series makes it suitable for detecting regime changes. Sinn et al. proposed methods for change point detection based on the principle that a shift in dynamics will manifest as a statistically significant change in the observed ordinal distribution [41]. They developed two algorithms: one employing a kernel-based distance metric, the Maximum Mean Discrepancy (MMD), to compare ordinal distributions from consecutive time windows, and another based on clustering ordinal distributions over time. This methodology has been successfully applied to detect changes in real-world data, including physiological recordings like ECG and EEG [41] and financial time series [48]. Although these examples employ the ordinal distributions rather than the raw symbols themselves, they nonetheless demonstrate that ordinal symbols provide an effective representation of the underlying dynamics for time series analysis tasks.

Analysis of reservoir computing systems

Ordinal analysis has also been applied to understand the internal dynamics of reservoir computing (RC) systems, particularly ESNs. Thorne et al. compared permutation entropy calculated from ESN output signals with complexity estimates derived directly from the reservoir's internal state dynamics [44]. Their findings indicated a clear link between the complexity of the driving signal (measured by PE) and the emergent structure and dynamics within the reservoir, suggesting PE could inform choices about ESN hyperparameters like connectivity or spectral radius. Sun et al. applied ordinal analysis directly to the time series of individual reservoir neuron states [42]. They defined an "Instantaneous State Entropy (ISE)" based on the diversity of ordinal patterns across all neurons at a single time step, and a "Global State Entropy (GSE)" reflecting the variability in entropy across different neurons over time. They found correlations between these reservoir entropy measures and the network's forecasting performance, providing insights into the interpretability of reservoir dynamics.

Ordinal partitions as a predictive feature

The preceding examples demonstrate the effectiveness of ordinal partitions and their derived features for analysing time series. This motivates further exploration of ordinal partitions as an input to models for time series prediction. While most prior work has focused on using ordinal pattern probabilities as features, there are compelling reasons to consider the ordinal partitions themselves. By segmenting the input space into distinct ordinal patterns, these partitions naturally lend themselves to piecewise or mixture-of-experts modeling approaches, where different model components can specialize in autoregressive prediction conditioned on particular ordinal patterns.

While direct integration of ordinal partitions into ESN architectures as proposed in this thesis is novel, in their review Leyva et al. suggest that ordinal probabilities have the potential to inform model decisions in machine learning contexts [22]. Ordinal features offer several practical advantages. They are inherently scale-invariant and robust to noise, making them representative features for real-world time series data that may be chaotic or noisy [4]. Unlike more complex feature extraction methods such as wavelet decompositions, empirical mode decomposition, or deep autoencoder representations, ordinal partitions are simple and computationally efficient to compute. Their interpretability is another major strength: ordinal patterns simply represent the local ordering and thus recent dynamics of the data, so improvements in model performance by including these features can be attributed to the model's ability to exploit meaningful temporal structure rather than obscure or uninterpretable transformations. Taken together, these properties make ordinal partitions an attractive predictive feature for time series prediction, warranting further experimentation in predictive MOE models.

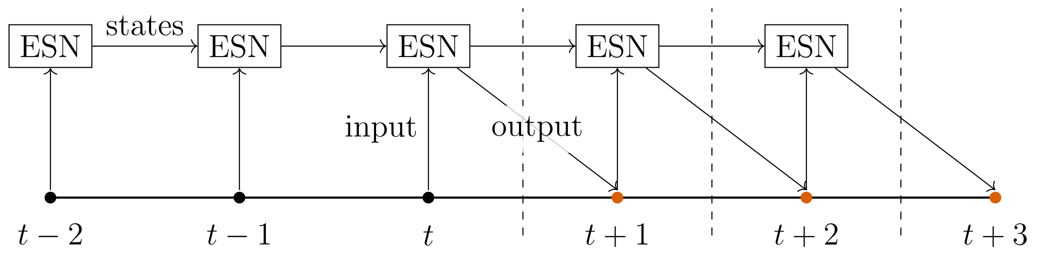
In the following chapters, we apply ordinal partitions as a predictive feature in novel ESN architectures. In Chapter 5 we use the ordinal partition to drive the gating of a piecewise readout function, and in Chapter 6 we use ordinal partitions to drive the input routing of a modular ESN. Both of these models receive the time series as the input series while the contemporaneous ordinal symbols are used in routing/gating. This hybrid input retains the ESN's lightweight training while making use of ordinal partitions.

Chapter 4

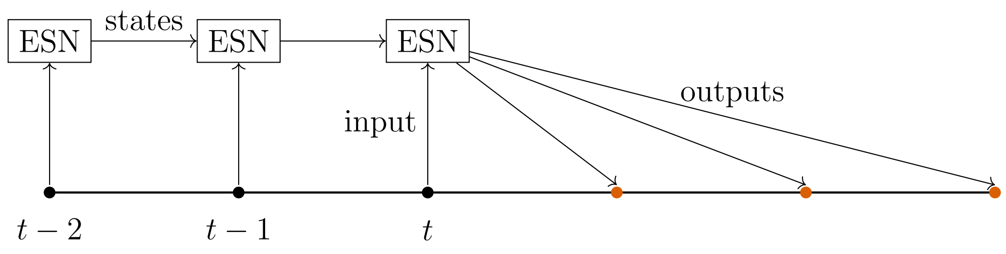
Testing methods

Prediction and error

To evaluate each proposed architecture, we tested its ability to predict multiple steps of a time series and measured the resulting degree of error. The predictions take two forms: iterative prediction (a.k.a. generative, closed-loop) and direct prediction (a.k.a. teacher forced, open-loop) [28]. **Iterative prediction** has the readout of the ESN fitted to predict a single step into the future, and the predicted output is used as the input to derive a prediction for the following time step. **Direct prediction** has the ESN predict n time steps in advance in a single shot. This can be achieved by fitting a readout matrix rather than a readout vector to predict a vector of values rather than a single value. Each of the values in the vector represent the prediction for a different value of n time steps into the future. The two types of multi-step prediction are visualised in the diagrams in Figure 4.1.



(a) Iterative Prediction.



(b) Direct Prediction.

Figure 4.1: Illustration of (a) Iterative and (b) Direct Prediction using an ESN, where the input data points are coloured black and the predictions are orange.

To test the ability for an architecture to perform multi-step prediction, we will split the time series into non-overlapping chunks and perform multi-step predictions from the beginning of each chunk. For recursive predictions, the ESN will change from being driven by the true signal to being driven by its own predictions at the start of each chunk. For direct predictions, all of the values within the chunk will be predicted in one shot after being driven by the true signal up to the start of the chunk. Concatenating each of these chunks will produce discontinuous predictions over the whole time series. To achieve this, we first feed the time series into the ESN and collect the reservoir states at each time step. To generate the predictions for each chunk, we look up the reservoir state at the beginning of the chunk from the collected reservoir states. We then derive predictions for the length of the chunk starting from this state by iterative or direct prediction, yielding a series of predictions $\hat{\mathbf{y}}$. An example of a true time series $\mathbf{y}$ and a predicted time series $\hat{\mathbf{y}}$ are shown in Figure 4.2. Comparing these predictions with the true values, we can calculate the Root Mean Squared Error (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}.$$

Using prediction error metrics such as RMSE provide a quantified means to compare the efficacy of different ESN architectures and parameter settings. Although some measurements such as memory capacity are helpful, prediction error directly reflects how well a model accomplishes the task of forecasting future time series values. A lower error signifies a more accurate prediction, indicating that the model has more effectively captured the underlying dynamics of the data. RMSE represents an average magnitude of prediction error but penalizes larger errors more heavily due to the squaring operation, making it sensitive to both the magnitude of error and the variance of the predictions.

Testing data

The different ESN architectures were tested on single-component time series data generated from three well-known dynamical systems: the Lorenz [27], Rössler [39], and Mackey-Glass [34] systems. Each of these systems exhibits distinct dynamical characteristics, making them suitable benchmarks for evaluating prediction performance.

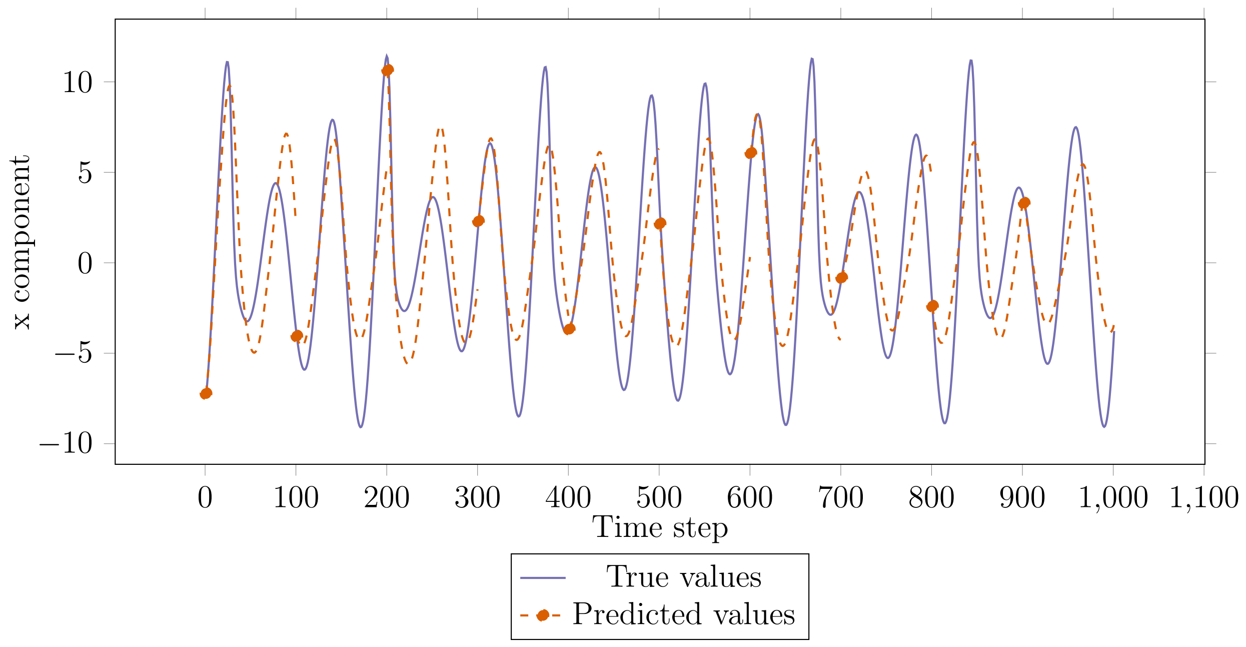


Figure 4.2: Time series plot of some true and predicted values. The start of each chunk is marked with a dot, where the ESN changes from being driven by the true signal to being driven by its own predictions.

The Lorenz system is a set of three coupled ordinary differential equations

$$\frac{dx}{dt} = \sigma\,(y - x),$$
$$\frac{dy}{dt} = x\,(\rho - z) - y,$$
$$\frac{dz}{dt} = x\,y - \beta\,z,$$

for parameters $\sigma, \rho, \beta > 0$. We used the common chaotic values $\sigma = 10$, $\rho = 28$ and $\beta = \frac{8}{3}$.

The Rössler system is described by

$$\frac{dx}{dt} = -\,y - z,$$
$$\frac{dy}{dt} = x + a\,y,$$
$$\frac{dz}{dt} = b + z\,(x - c),$$

with parameters a, b and c. We adopt the commonly used chaotic setting $a = 0.2$, $b = 0.2$ and $c = 5.7$.

The Mackey-Glass system is determined by a scalar delay-differential equation

$$\frac{dx}{dt} \;=\; \beta\,\frac{x(t-\tau)}{1 + [x(t-\tau)]^n} - \gamma\,x(t),$$

where τ is the delay, n determines the non-linearity, and $\beta, \gamma > 0$ control the production and decay rates respectively. We used the common parameters $\beta = 0.2$, $\gamma = 0.1$, $n = 10$ and $\tau = 17$.

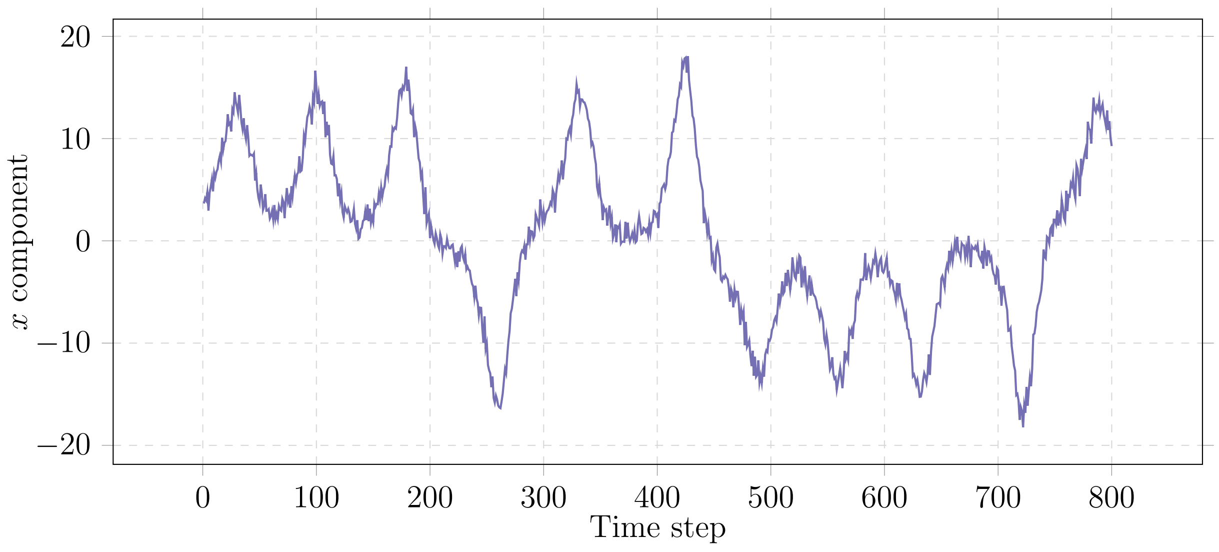
The size of the time step used to integrate each time series was chosen to make the frequency of the oscillations over the time steps comparable between the three attractors. For example, Figure 4.3 shows the three attractors with time steps chosen so that the time series have around 12 oscillations over 800 time steps. We also tested the architectures with a time series simulated from the Lorenz attractor with a time step 5 times as long ($\Delta t = 0.05$). The sequences were each $50,000$ time steps and were split into a $40,000$ time step training set and a $10,000$ time step validation set. Although we were limited to this dataset size due to computational restraints, we also tested both proposed models using a Lorenz dataset ($\Delta t = 0.05$) with $500,000$ time steps to assess the effect of increasing training data. This cursory testing showed consistent patterns to those found for the time series of length $50,000$, however more testing may be beneficial in future research.

To assess robustness against noise, Gaussian noise was introduced into the time series. Specifically, the standard deviation of each original time series was computed, and Gaussian noise with a standard deviation equal to a prescribed multiple of the original series' standard deviation was added. This procedure allowed for systematic evaluation of the ESN architectures' performance under varying levels of noise contamination. Figure 4.3 shows time series simulated from the three systems with additive noise multiple $\alpha = 0.1$.

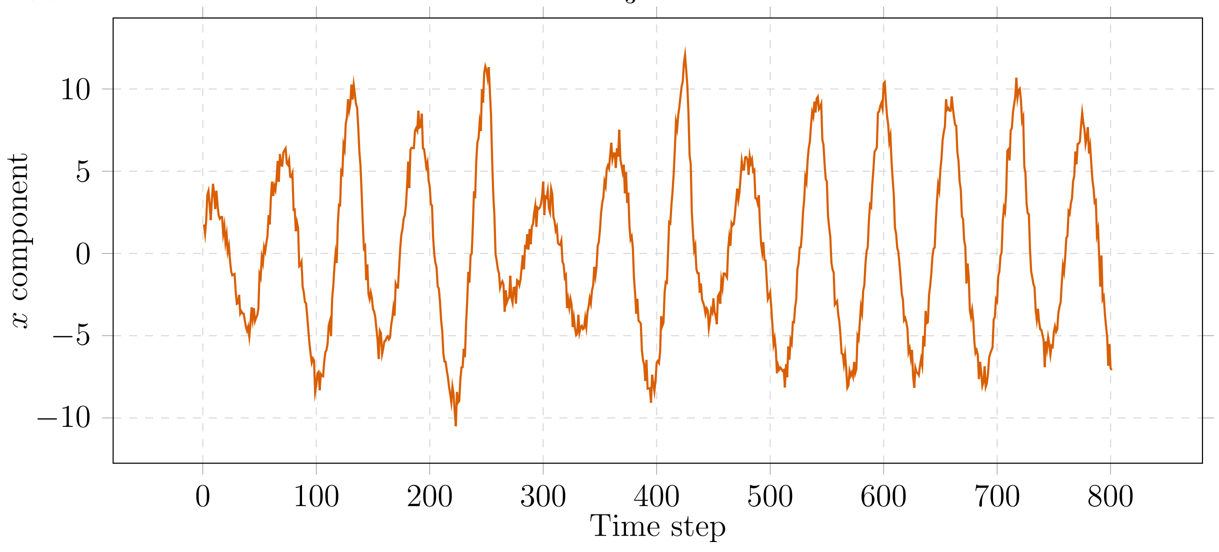
In the forthcoming error vs prediction length plots we superimpose a vertical dashed line located at $\frac{1}{\lambda}$, where λ is the largest Lyapunov exponent of the noiseless system. The quantity $\frac{1}{\lambda}$, often called the Lyapunov time, is the characteristic time over which two trajectories that start arbitrarily close diverge by a factor of e. While this timescale is not a strict limit on prediction accuracy, it signals when sensitivity to initial conditions becomes pronounced [19].

ESN instantiation and readout fitting

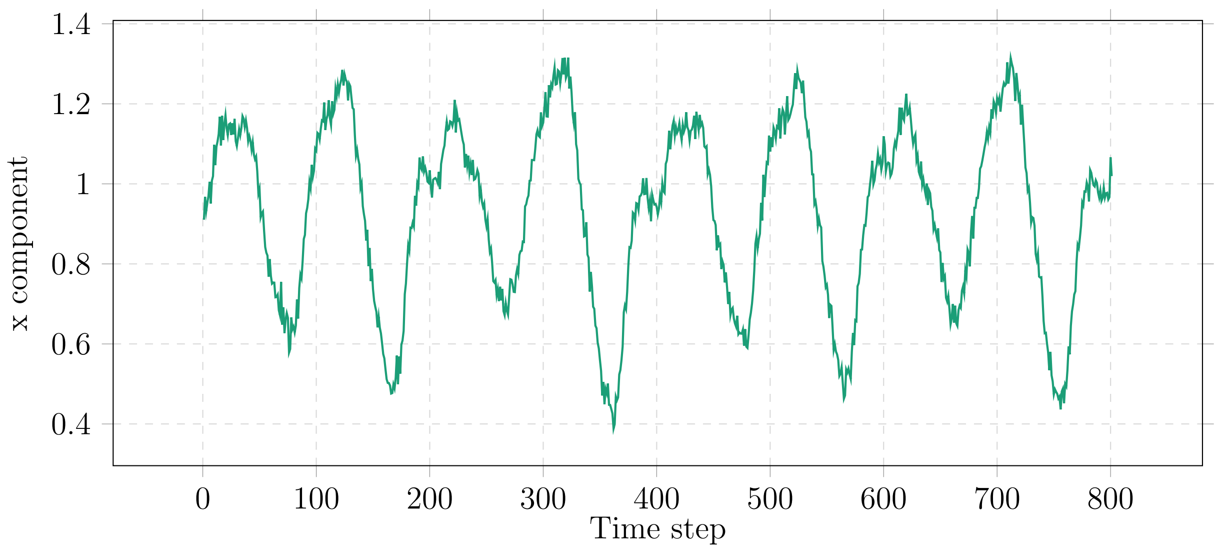
Each recurrence matrix $\mathbf{W}_{rec}$ was initialised as an Erdős-Rényi random graph with connection probability 0.05, a sparsity that Jaeger found sufficient to provide "rich" dynamics [15]. Subsequent surveys note that sparser reservoirs tend to give slightly better performance but note that the benefit of optimising the sparsity is minimal [28]. After instantiation, each recurrence matrix $\mathbf{W}_{rec}$ was rescaled to a spectral radius $\rho = 1.1$, which allows the reservoir to maintain a memory of past inputs while still being responsive to new inputs [15]. A cursory sweep over $\rho \in [0.8, 1.7]$ showed that smaller values minimised



(a) Lorenz system simulated with $\alpha = 10$, $\beta = \frac{8}{3}$ and $\rho = 28$, and time step size $\Delta t = 0.01$.



(b) Rössler system with parameters $a = 0.2$, $b = 0.2$ and $c = 5.7$, and time step size $\Delta t = 0.1$.



(c) Mackey-Glass system with parameters $\beta = 0.2$ and $\gamma = 0.1$, and time step size $\Delta t = 0.5$.

Figure 4.3: Time series data used in the prediction experiments, each simulated and then corrupted by additive noise $\epsilon \sim \mathcal{N}(0, (0.1\,\sigma)^2)$, where σ is the measured standard deviation of the time series component.

RMSE for short predictions whereas larger values were preferable for longer predictions. While absolute RMSE varied with ρ, the ranking of models appeared consistent. We used a normalisation parameter of $\beta = 0.001$ in the ridge regression used to fit the readout vector [28].

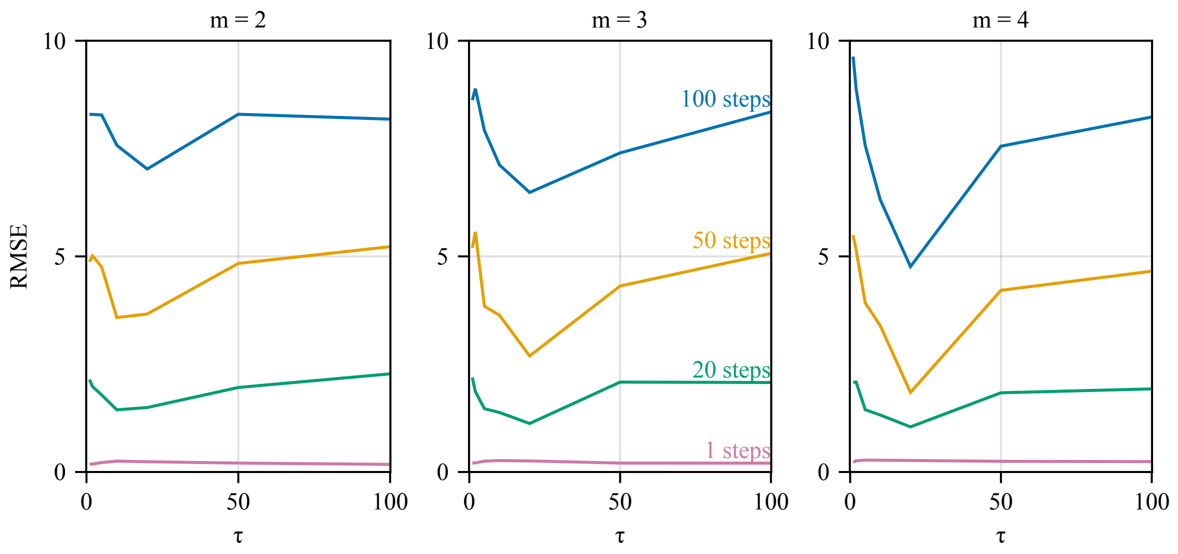
Selection of ordinal pattern delay τ

As explained in Chapter 3, the ordinal partition is created by two main parameters: the number of datapoints considered m and the time step delay between each of the datapoints τ. By varying the value of τ used to partition the data and calculating the mean prediction RMSE over multiple trials, we can determine a near-optimal selection of τ for use in our models. Figure 4.4 shows the ORSESN's prediction RMSE over the Lorenz time series at a time step size of $\Delta t = 0.01$ for iterative and direct prediction. The figure shows that the RMSE is at a minimum or close to a minimum at $\tau = 20$ for all tested values of τ for each of $m = 2, 3, 4$. This is true for both iterative and direct prediction, thus we have chosen $\tau = 20$ for our tests on the Lorenz time series at a time step size of $\Delta t = 0.01$. By the same methods we chose ordinal pattern delays for each of the time series, for each value of m, which can be found in Table 4.1.

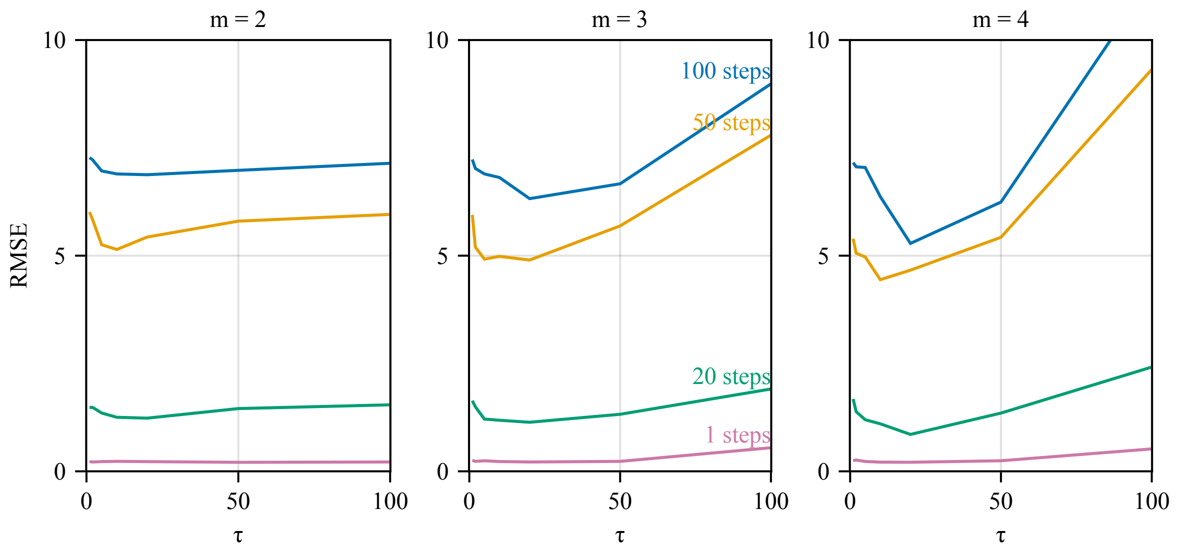
Practical considerations for the ordinal pattern dimension

The number of possible partitions for a given value of m is given by $m!$. Without additive noise the time series will visit a subset of these possible partitions, for example where $m = 4$ the x component of the simulated Lorenz system showed data points in 13 of the 24 possible partitions. Unsurprisingly however, with additive noise the time series will visit many more possible partitions, and for a sufficiently large noise level it will visit all possible partitions. For example where $m = 4$ and with noise multipliers of $\alpha = [0.1, 0.2, 0.3]$ the x component of the simulated Lorenz system showed data points in $[21, 22, 24]$ of the 24 possible partitions respectively.

For the architecture described in chapter 6, the size of the reservoir is dependent on the number of expected partitions, calculated by $k_{total} = kN_{obs}$ where N_{obs} is the number of observed partitions in the time series and k is the desired number of nodes in each sub-reservoir. Varying the value m for the same k will thus produce a different total number of nodes k_{total}. ESNs have generally been found to produce a better prediction for a



(a) Iterative prediction RMSE for the ORSESN as a function of the partition delay τ.



(b) Direct prediction RMSE for the ORSESN as a function of the partition delay τ.

Figure 4.4: Prediction RMSE for the ORSESN as a function of the partition delay size τ, with training data additive noise $\alpha = 0.1$ and reservoir size $k = 468$. Subfigures (a) and (b) correspond to iterative and direct prediction, respectively. Curves correspond to different numbers of prediction steps, and each axis represents an ORSESN with a different value for m.

Model	Dataset	Prediction Type	$m = 2$	$m = 3$	$m = 4$
ORSESN	Lorenz 0.01	Iterative	20	20	20
ORSESN	Lorenz 0.05	Iterative	10	10	4
ORSESN	Rossler 0.1	Iterative	20	20	20
ORSESN	MG 0.5	Iterative	20	20	20
ORSESN	Lorenz 0.01	Direct	20	20	20
ORSESN	Lorenz 0.05	Direct	4	4	4
ORSESN	Rossler 0.1	Direct	20	20	20
ORSESN	MG 0.5	Direct	20	20	20
OPESN	Lorenz 0.01	Iterative	20	20	20
OPESN	Lorenz 0.05	Iterative	10	10	4
OPESN	Rossler 0.1	Iterative	20	20	10
OPESN	MG 0.5	Iterative	20	20	20
OPESN	Lorenz 0.01	Direct	20	20	20
OPESN	Lorenz 0.05	Direct	4	4	4
OPESN	Rossler 0.1	Direct	10	20	10
OPESN	MG 0.5	Direct	20	20	20

Table 4.1: Optimal partition delay values τ selected for ORSESN and OPESN models across different datasets and prediction types. Values are shown for each tested m (ordinal pattern dimension), and were optimise to minimise RMSE.

larger reservoir size [29] [28], so to compare between different values of m we have chosen values of k based on the number of observed partitions N_{obs} for each time series and value of m in order to produce comparable values of k_{total}. It is important to approximately match the total size of reservoirs to delineate the effect of dividing the reservoir into sub-reservoirs and the effect of increasing the total reservoir size. The reader should note that an OPESN with $m = 5$ has up to $5! = 120$ possible patterns. To allow each sub-reservoir of the OPESN to have at least $k = 20$ nodes could require $k_{total} \geqslant 2000$, which is not feasible to test under our compute resource constraints.

For the architecture described in chapter 5, the readout matrix will be of size $k \times N_{obs}$. For a reservoir of size $k = 500$ and $m = 5$, the readout matrix could be up to $500(5!) = 60000$ parameters. An ESN reservoir with connection probability 0.05 will have an expected number of non-zero parameters $\mathbb{E}[\|\mathbf{W}_{rec}\|_0] = 0.05(500^2) = 12500$. The reader can see that as we increase m, the ORSESN's readout size will scale factorially and that for $m > 4$ the size of the ORSESN's readout will be far larger than the number of non-zero parameters of the reservoir, forming a degenerate kind of ESN. For this reason we have not tested the ORSESN model for $m > 4$.

Testing procedure

To test each particular set of parameters we performed 30 trials, creating a new ESN each time. Creating a new ESN includes randomly instantiating the weights and the reservoir states, and fitting a new readout vector, ensuring that the trials are independent of one another. We aggregate the trials by finding the mean, standard deviation, minimum and maximum of the RMSE of their predictions [28]. We report means $\pm 1.96\times$ standard error across the 30 trials; a mean outside the baseline CI is considered significant.

A multi-step prediction can extend to an arbitrary number of steps into future $(y_1, ..., y_n)$ and these predictions could then be sub-sampled to a shorter number of prediction steps. Thus the same multistep prediction could yield error metrics for multiple different numbers of steps, but these metrics would not be independent. So for the purposes of our testing, we have regenerated the multistep predictions for each number of steps anew using a new ESN each time.

Chapter 5

Echo state network with ordinal partition based readout switching

The first approach uses a piecewise readout function to incorporate ordinal partition information into the structure of an ESN. In this architecture there is a different readout vector for each observed ordinal partition and at each time step, a readout vector is chosen depending on the partition of the input data. We will refer to this architecture as the Ordinal partition based Readout Switching ESN (ORSESN).

5.1 Implementation

Instantiating the weights and iterating the reservoir

The input weights $\mathbf{W}_{in}$ and recurrence matrix $\mathbf{W}_{rec}$ of the ORSESN are generated as they would be for a traditional ESN. $\mathbf{W}_{in}$ is a vector of length k drawn from a Normal distribution and $\mathbf{W}_{rec}$ is an Erdős-Rényi random network adjacency matrix. The initial states $s(0)$ form a vector of length k and are initialised by drawing from a Normal distribution. The ORSESN can be driven with the input and the reservoir iterated just like a traditional ESN, using the update equation:

$$\mathbf{s}(t+1) = f_{act}(\mathbf{W}_{in}\mathbf{x}(t) + \mathbf{W}_{rec}\mathbf{s}(t) + \mathbf{W}_{bias}).$$

Driving the reservoir with the training sequence and collecting the states into a matrix $\mathbf{S}$ will prepare us to fit the vectors that make up the piecewise readout function. For a training sequence of length n_{train}, let us collect these states into a matrix:

$$\mathbf{S} = [\mathbf{s}(1), ..., \mathbf{s}(n_{train})]^t \in R^{n_{train} \times k}.$$

Fitting the readout vectors

To fit the readout vector for each partition π, we must filter the input time series $\mathbf{Y}$ to those data points belonging to that partition π and filter the states $\mathbf{S}$ to those that arose immediately after that input from that partition π. We can define those inputs $\mathbf{Y}_\pi$ and states $\mathbf{S}_\pi$ that relate to each partition π by indexing $\mathbf{Y}$ and $\mathbf{S}$, where π_t represents the ordinal partition of the data at time t:

$$\mathbf{Y}_\pi = \{\mathbf{Y}_t | \pi_t = \pi\},$$
$$\mathbf{S}_\pi = \{\mathbf{S}_{t,*} | \pi_t = \pi\},$$

and then for each partition π, we can fit a readout vector using Ridge regression:

$$(\mathbf{C}_{out}^T)_\pi = (\mathbf{S}_\pi^T \mathbf{S}_\pi + \beta \mathbf{I})\ \mathbf{S}_\pi^T \mathbf{Y}_\pi.$$

The readout vectors for each partition can then be arranged into an $N_{obs} \times k$ matrix,

$$\mathbf{C}_{out} = [(\mathbf{C}_{out})_1; ...; (\mathbf{C}_{out})_{N_{obs}}],$$

and a vector of predictions for each possible partition $\mathbf{y}_{all}(t)$ at time t can be inferred for any states $\mathbf{s}(t)$:

$$\mathbf{y}_{all}(t) = \mathbf{C}_{out}(t)\mathbf{s}(t).$$

To obtain just one of these predictions, we construct a mask based on the active partition at each time step. This mask e_π will be a 'one-hot encoded' vector of length N_{obs}, where 1 indicates the active partition at that time step and 0 indicates the other inactive partitions. By multiplying our mask and the prediction vector and summing the result, we can select the prediction for a given ordinal partition π_t:

$$\mathbf{y}(t) = \sum \mathbf{y}_{all}(t) e_{\pi_t}.$$

The selection of the prediction does not necessarily need to be one-hot like this, but could use an informed combination of predictions, as do other mixture-of-experts models [3] [21]. One such relevant method could be to weight the predictions of each of the partition readouts depending on the transition probabilities of the active ordinal partition, however we have not trialed this hypothesis.

Here we have defined a method for 'switching' the readout of an ESN based on the ordinal partition of the input data.

5.2 Results

Lorenz - iterative prediction

First we will consider the mean RMSE created by iterative predictions of the Lorenz time series with a $\Delta t = 0.01$. At a single recursive prediction step, the traditional ESN achieves the lowest mean RMSE of 0.173 (± 0.025), outperforming the ORSESN for all ordinal pattern dimensions (m). For $m = 2$, 3, and 4, the mean RMSEs are 0.237 (± 0.015), 0.258 (± 0.012), and 0.268 (± 0.010), respectively. This shows that for short term predictions, increasing the ordinal pattern dimension m leads to a higher mean RMSE for the ORSESN, and that the traditional ESN is optimal for very short-term predictions. However, as the number of recursive prediction steps increases, this relationship changes. After two and three steps, the RMSEs for all models increase, but the gap between the traditional ESN and the ORSESN with higher m begins to narrow. For example, at 3 steps, the traditional ESN has a mean RMSE of 0.334 (± 0.055), while the ORSESN with $m = 4$ has a mean RMSE of 0.341 (± 0.034).

The significant change occurs at longer prediction horizons, as shown in Figure 5.1a. By 10 steps, the traditional ESN's mean RMSE rises to 1.093 (± 0.159), while the ORSESN with $m = 4$ achieves a significantly lower (95% CI) mean RMSE of 0.739 (± 0.086). At 20 steps, the error rates have reversed: the traditional ESN's RMSE increases to 2.072 (± 0.230), whereas the ORSESN achieves significantly lower (95% CI) errors for all m, with $m = 4$ reaching the lowest RMSE of 1.004 (± 0.092). This advantage becomes even more pronounced as the prediction horizon extends further, with the ORSESN with $m = 4$ achieving a mean RMSE of approximately 3.174 (± 0.254) at 70 steps compared to the traditional ESN's 6.915 (± 0.732). However for $m = 2$, the trend does not significantly hold, with the $m = 2$ ORSESN's mean RMSE failing to be significantly lower for most prediction horizons. The $m = 3$ ORSESN ceases to provide a significant benefit at 200 prediction steps, beyond which all four models diverge from the signal. These results demonstrate that, although the traditional ESN is best for immediate predictions, the ORSESN with higher ordinal pattern dimension m provides significantly more accurate and stable predictions for longer prediction horizons.

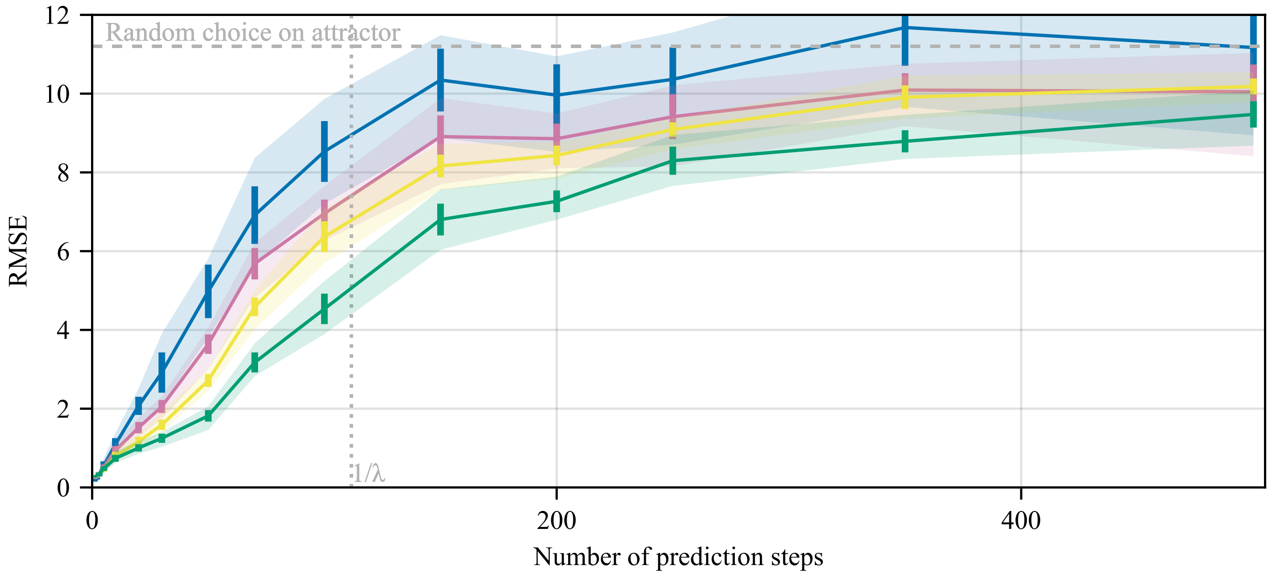
The ORSESN also demonstrates greater consistency in its predictions compared to the traditional ESN, as measured by the standard deviation of RMSE across trials. Since each trial uses a newly instantiated set of states and weights, and regenerates the additive noise for the time series, the standard deviation of the RMSEs generated by each trial can represent how robust each architecture is to its random initial conditions. The ORSESN achieves a lower standard deviation of RMSE across trials, especially at longer prediction horizons. At 10 recursive steps, the traditional ESN shows a standard deviation of 0.159, while the ORSESN with $m = 4$ has a much lower value of 0.086. This pattern continues at 30 steps (0.511 vs. 0.116) and 50 steps (0.679 vs. 0.147). Notably, the variability decreases as the ordinal pattern dimension increases, with $m = 4$ consistently showing the lowest standard deviations in the medium-term prediction range.

When examining the Lorenz system with a larger time step size of $\Delta t = 0.05$, we observe similar patterns although less pronounced. The traditional ESN has a mean RMSE of 1.330 ($\pm$0.040) at a single prediction step, which is comparable to the ORSESN with $m = 3$ (1.337 $\pm$0.055) and $m = 4$ (1.328 $\pm$0.059). Compared to the Lorenz series with $\Delta t = 0.01$, the performance of the ORSESN improves over the traditional ESN at even shorter prediction lengths. After just 2 recursive prediction steps, the ORSESN with $m = 4$ already demonstrates significantly (95% CI) superior performance with a mean RMSE of 1.483 ($\pm$0.040) compared to the traditional ESN's 1.535 ($\pm$0.017). The ORSESN with $m = 3$ demonstrates significantly superior performance at 20 time steps, and $m = 2$ at 30, but by 70 prediction steps, all three models have converged, as shown in Figure 5.1b.

Lorenz - direct prediction

Here we will analyse the ability of the ORSESN to generate direct predictions. For the Lorenz system with time step size $\Delta t = 0.01$, as shown in Figure 5.2a, the traditional ESN initially performs well for short prediction horizons, achieving a mean RMSE of 0.228 ($\pm$0.033) after 1 step. The ORSESN models show slightly higher errors at this prediction length, with mean RMSEs of 0.243 ($\pm$0.020), 0.233 ($\pm$0.017), and 0.228 ($\pm$0.021) for $m = 2$, $m = 3$, and $m = 4$, respectively. But mirroring what we observed in the iterative predictions, the ORSESN models perform better at longer prediction lengths. At 20 prediction steps, all three ORSESN models achieve a statistically significant (95% CI) lower mean RMSE than the traditional ESN, with a higher m producing a lower RMSE over the whole tested range of prediction lengths.

(a) Mean iterative prediction RMSE on the Lorenz series with time step size $\Delta t = 0.01$ for the ORSESN with delay $\tau = 20$, noise level $\alpha = 0.1$, and total reservoir size $k = 500$.



(b) Mean iterative prediction RMSE on the Lorenz series with time step size $\Delta t = 0.05$ for the ORSESN with delays $\tau = 10, 10, 4$ for $m = 2, 3, 4$ respectively, noise level $\alpha = 0.1$, and total reservoir size $k = 500$.

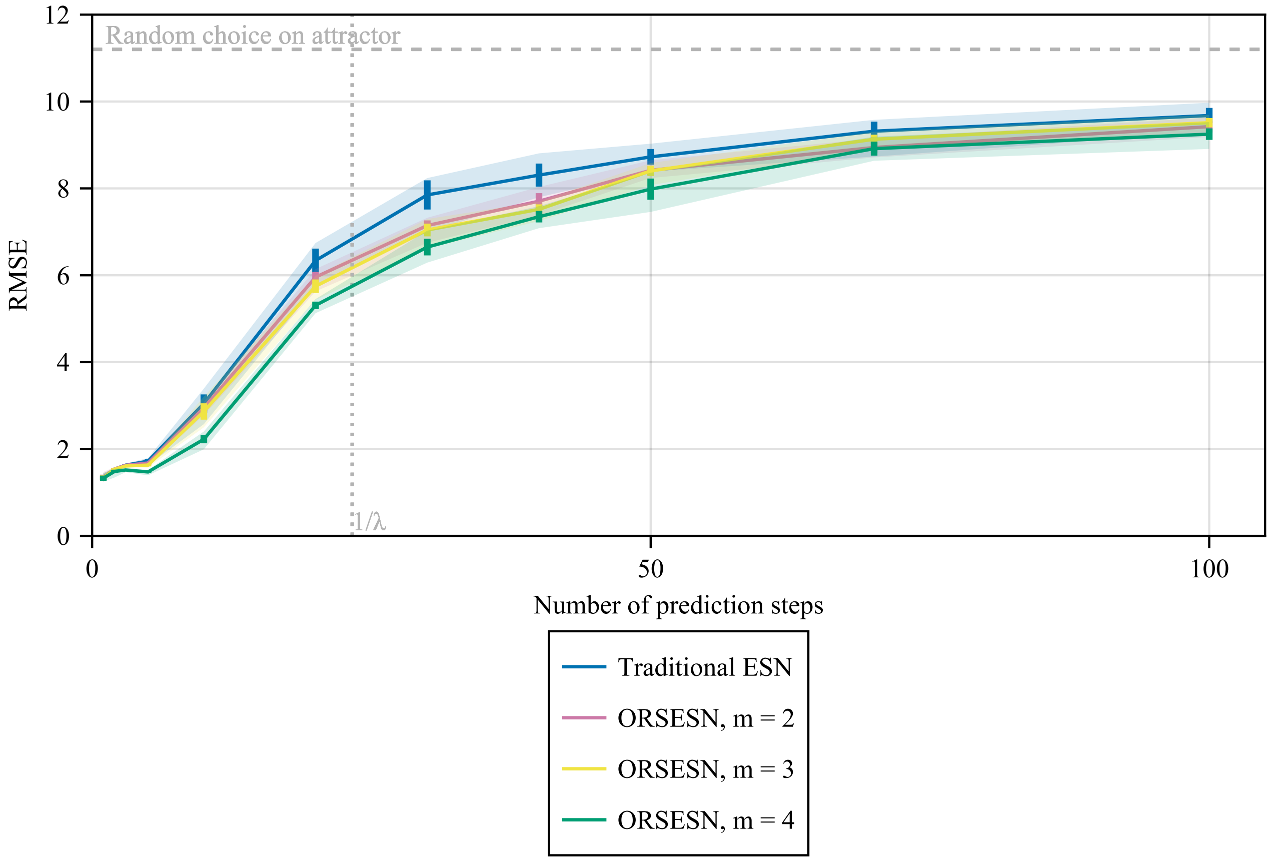


Figure 5.1: Mean iterative prediction RMSE for the ORSESN on the Lorenz x-component time series with additive noise $\alpha = 0.1$ and total reservoir size $k = 500$. Subfigure (a) shows results for time step size $\Delta t = 0.01$, delay $\tau = 20$ for all m; subfigure (b) shows results for time step size $\Delta t = 0.05$, delays $\tau = 10, 10, 4$ for $m = 2, 3, 4$, respectively. Standard deviations between trials are represented by vertical bars and shaded regions indicate the full range of trial results.

For direct prediction on the Lorenz system with time step size $\Delta t = 0.05$ (Figure 5.2b), the ORSESN with $m = 4$ outperforms the traditional ESN from the start (mean RMSE of 0.239 ± 0.019 vs. 0.311 ± 0.025 at 1 step) and maintains this advantage throughout longer horizons (0.759 ± 0.045 vs. 1.175 ± 0.076 at 5 steps; 5.003 ± 0.088 vs. 6.214 ± 0.254 at 20 steps). The ORSESN with $m = 3$ also reliably outperforms the traditional ESN, however the ORSESN with $m = 2$ fails to significantly improve (95% CI) the traditional ESN at many prediction lengths.

Both direct and iterative prediction methods confirm the same conclusion: while traditional ESNs are superior or on-par for short-term forecasts, the ORSESN provides more accurate and stable predictions for medium to long-term horizons, with performance improving as the ordinal pattern dimension increases. The direct prediction results further confirm that the ORSESN's superior performance is not an artifact of error accumulation in iterative prediction but rather reflects a genuine improvement in the model's ability to capture the underlying dynamics of the chaotic system. There is a consistent pattern of lower errors for the ORSESN at longer prediction horizons across both forms of prediction.

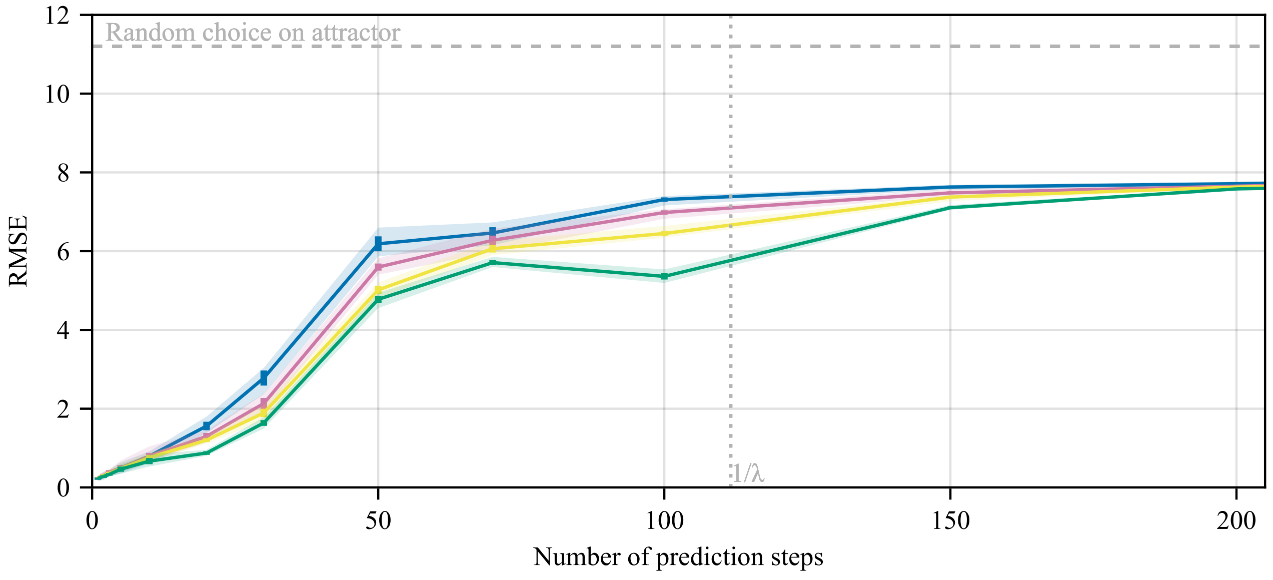
Comparison across attractors

Rössler system

For the Rössler system integrated with a time step size of $\Delta t = 0.1$ (Figure 5.3), the results show a similar pattern to the Lorenz attractor. The traditional ESN exhibits lower error for very short term predictions, but after 5 recursive steps the mean RMSE of the ORSESN with $m = 4$ (0.374 ± 0.0190) and $m = 3$ (0.385 ± 0.022) significantly (95% CI) outperform the traditional ESN (0.482 ± 0.041). Like those predictions for the Lorenz series, this advantage becomes more pronounced at longer horizons; for instance, at 200 steps, ORSESN with $m = 4$ achieves a significantly lower mean RMSE of 2.176 (± 0.250) compared to 4.393 (± 1.126) for the traditional ESN. Furthermore, the ORSESN with $m = 4$ demonstrates remarkable prediction stability at this horizon, with a standard deviation of only 0.250, whereas the traditional ESN has a standard deviation of 1.126.

The ORSESN with $m = 2$ fails to be significantly more accurate at multiple prediction lengths, and the ORSESN with $m = 3$ provides insignificant improvement after 150 prediction steps. After 200 prediction steps, the ORSESN with $m = 4$ no longer signifi-

(a) Mean direct prediction RMSE on the Lorenz series with time step size $\Delta t = 0.01$ for the ORSESN with delay $\tau = 20$, noise level $\alpha = 0.1$, and total reservoir size $k = 500$.



(b) Mean direct prediction RMSE on the Lorenz series with time step size $\Delta t = 0.05$ for the ORSESN with delay $\tau = 4$, noise level $\alpha = 0.1$, and total reservoir size $k = 500$.

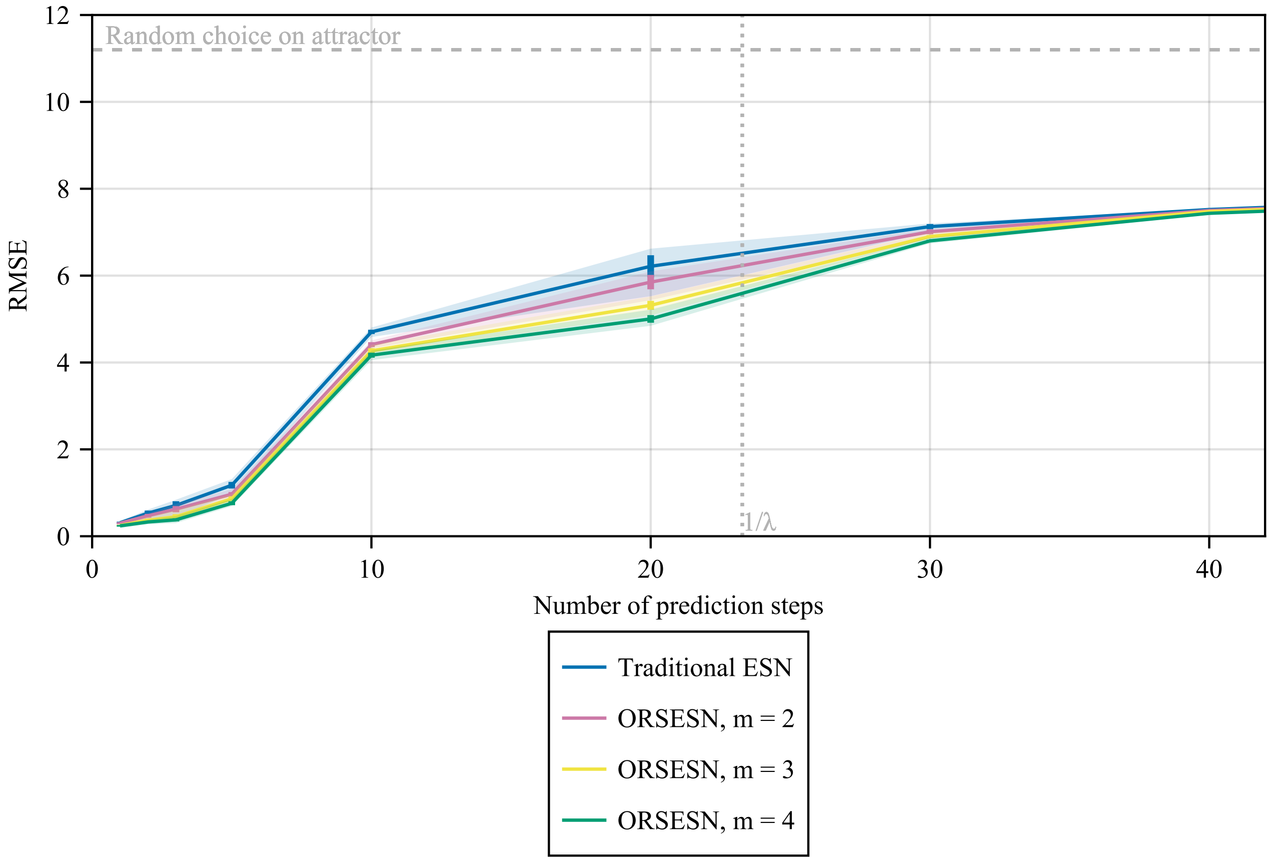


Figure 5.2: Mean direct prediction RMSE for the ORSESN on the Lorenz x-component time series with additive noise $\alpha = 0.1$ and total reservoir size $k = 500$. Subfigure (a) shows results for time step size $\Delta t = 0.01$, delay $\tau = 20$ for all m; subfigure (b) shows results for time step size $\Delta t = 0.05$, delay $\tau = 4$ for all m. Standard deviations between trials are represented by vertical bars and shaded regions indicate the full range of trial results.

cantly outperforms the traditional ESN, but this can be seen to be a result of the higher standard deviation in the results of the traditional ESN.

Comparing these findings with the Lorenz system, the general performance characteristics are consistent. Traditional ESNs tend to be more accurate for shorter prediction lengths, however ORSESN models outperform on accuracy and stability for longer predictions lengths, particularly where $m = 4$. The crossover point, where ORSESN models begin to outperform traditional ESNs, typically occurs within the first 10 prediction steps for both chaotic systems.

What appears peculiar is the comparably low mean RMSE and small variance of the ORSESN with $m = 4$. Figure 5.4 shows iterative predictions over 200 prediction steps (Rossler with $\Delta t = 0.1$) for the ORSESN and traditional ESN. Simply inspecting these predictions, all four models appear to predict the dynamics of only approximately one cycle before reverting to an ‘average’ periodic prediction. However the ORSESN with $m = 4$ appears to be able to predict the dynamics of 2 to 3 cycles to varying degrees of success before reverting to the ‘average’ periodic prediction.

The direct prediction performance on the Rössler system (Figure 5.5) is consistent with the iterative prediction trends. The benefit of the ORSESN appears more pronounced for the Rossler time series than the Lorenz time series. Direct predictions on the Rossler series again shows the traditional ESN with slightly lower error for short predictions but the ORSESN models with higher ordinal pattern dimensions (m) generally achieve better accuracy for longer predictions. At 20 prediction steps, ORSESN with $m = 4$ (mean RMSE 0.701 ± 0.058) is significantly (95%CI) more accurate than the traditional ESN (mean RMSE 1.094 ± 0.155). The mean prediction RMSE for all models converges, similar to the direct predictions for the Lorenz series, at approximately 350 prediction steps.

Mackey-Glass system

The iterative prediction results for the Mackey-Glass system with a time step size $\Delta t = 0.5$ are illustrated in Figure 5.6. Like the Lorenz and Rössler attractors, the traditional ESN demonstrates a clear advantage in single time step prediction for this system.

Similarly again, the ORSESN models generally show significantly (95% CI) improved performance over the traditional ESN as the prediction horizon grows. For instance, at 20 recursive prediction steps, the traditional ESN has a mean RMSE of 0.110 (± 0.010),

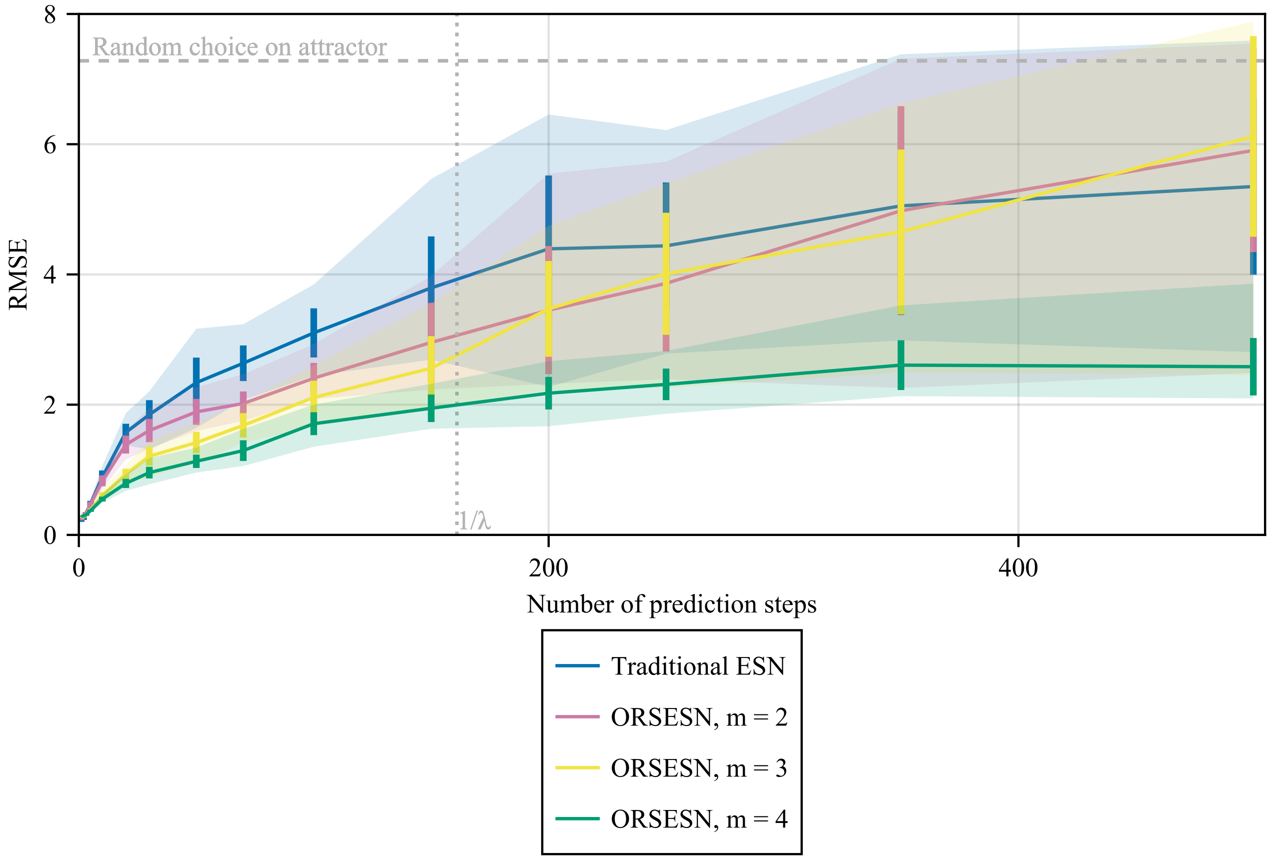


Figure 5.3: Mean iterative prediction RMSE for the ORSESN for Rössler x component time series with time step size $\Delta t = 0.1$ and additive noise $\alpha = 0.1$, delay $\tau = 20$ and total reservoir size $k = 500$. The standard deviations between trials are represented by vertical bars and the range of trial results is represented by the shaded regions.

while the ORSESN with $m = 2$ achieves 0.071 ($\pm$0.003), $m = 3$: 0.070 ($\pm$0.003), and $m = 4$: 0.039 ($\pm$0.001).

Unlike the other attractors, the ORSESN models with $m = 2$ and $m = 3$ maintain significantly (95% CI) improved accuracy only up to around 70 prediction steps, while the ORSESN with $m = 4$ continues to provide improved accuracy for much longer predictions.

The ORSESN with $m = 4$ achieves a consistent significantly (95% CI) improved accuracy for all tested prediction lengths 5 time steps or longer. It maintains a much lower mean RMSE and a smaller standard deviation compared to both the traditional ESN and the other ORSESN configurations ($m = 2, 3$) for longer prediction horizons. For example, the performance at 100 steps is 62% lower in error than the traditional ESN at 100 steps, with standard deviation of 0.002 much lower than the traditional ESN's 0.017. An example of the prediction of all four models can be found in Figure 5.7, however unlike the Rossler system, a simple qualitative explanation for the outperforming ORSESN with

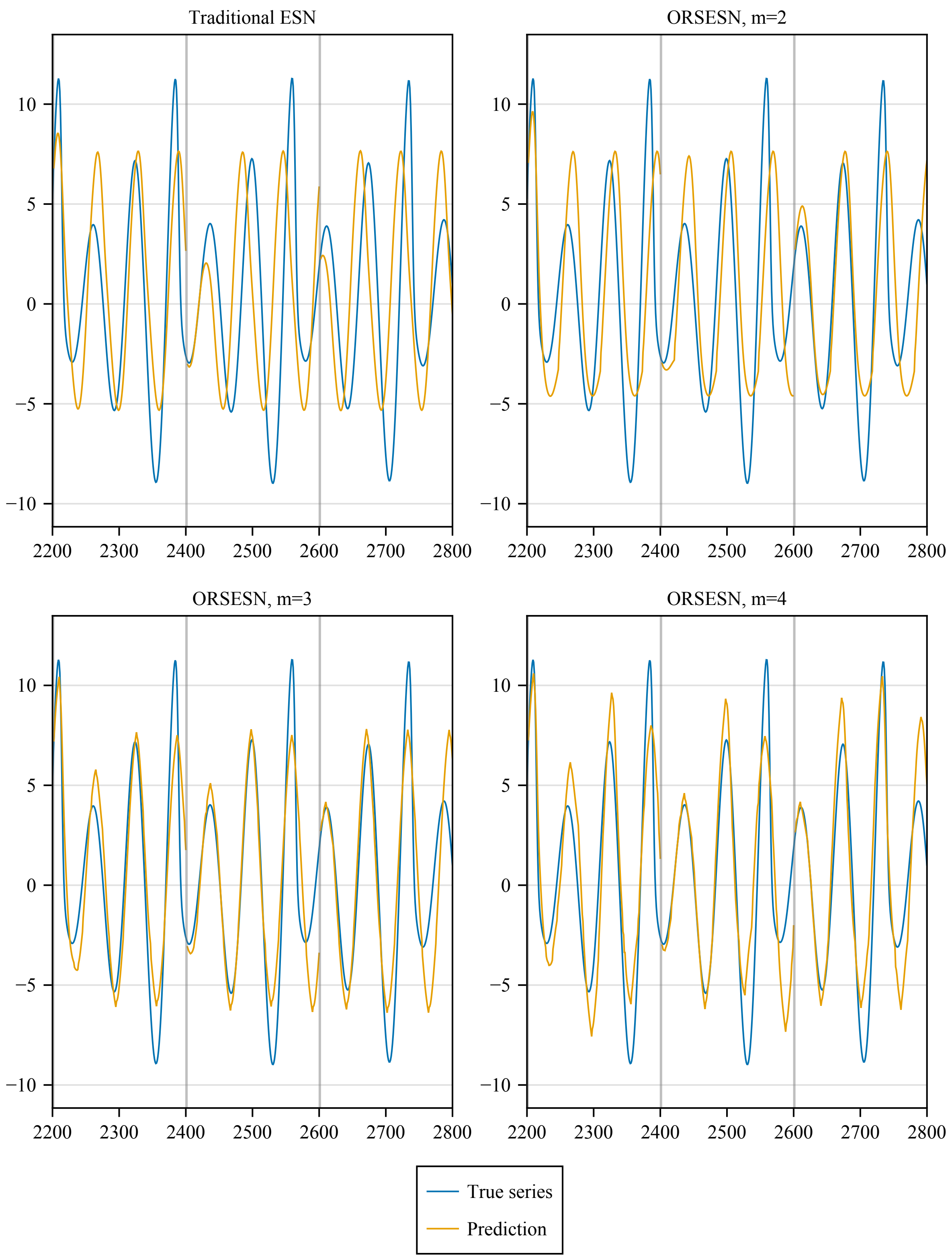


Figure 5.4: Iterative free-run predictions for the Rössler system (x component) with time step size $\Delta t = 0.1$. The figure compares the traditional ESN with ORSESN models of varying ordinal pattern dimensions ($m = 2, 3, 4$) over 200 prediction steps. All models were configured with a delay $\tau = 20$, additive noise $\alpha = 0.1$, and a total reservoir size $k = 500$. Each grey vertical line indicates the beginning of prediction chunk, where the echo state network changes from being driven by the true signal to being driven by its own predictions.

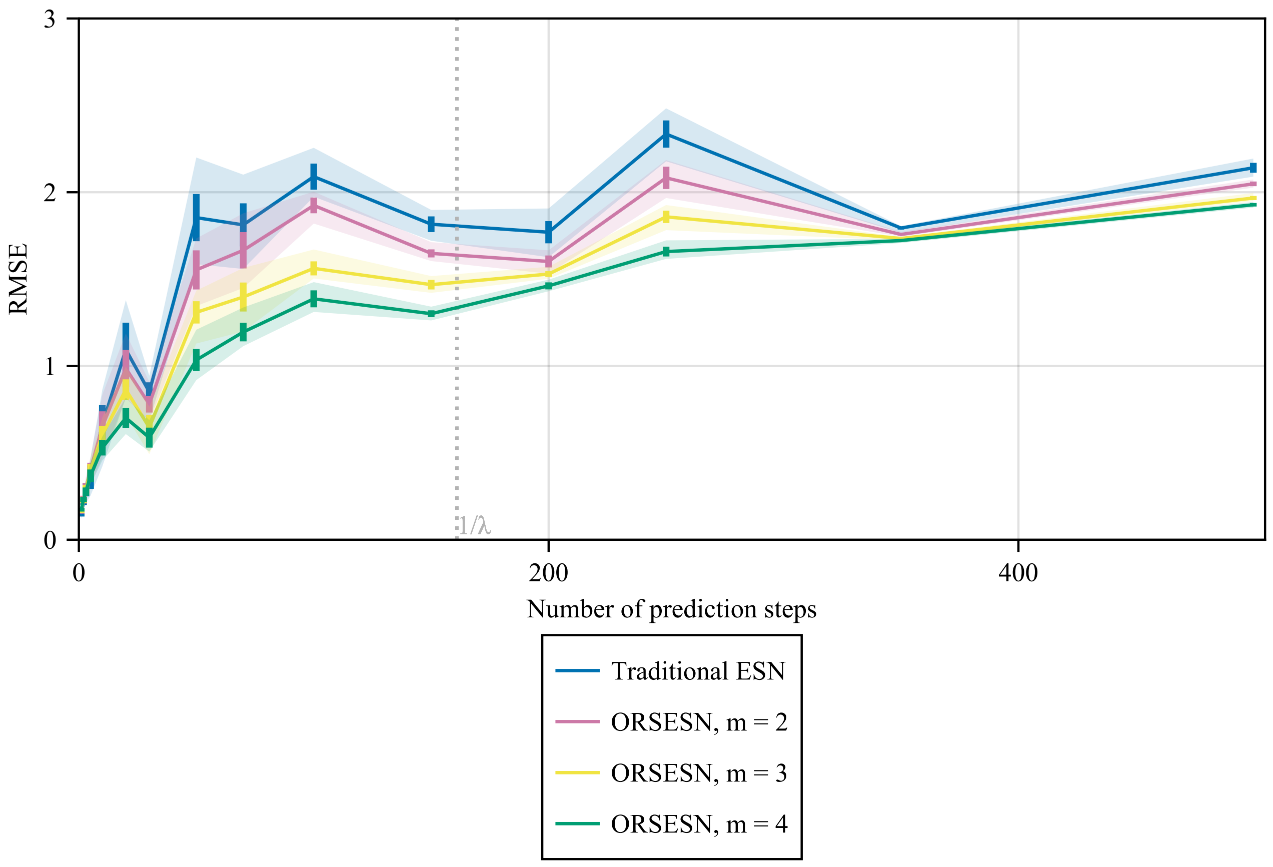


Figure 5.5: Mean direct prediction RMSE for the ORSESN for Rössler x component time series with time step size $\Delta t = 0.1$ and additive noise $\alpha = 0.1$, delay $\tau = 20$ and total reservoir size $k = 500$. The standard deviations between trials are represented by vertical bars and the range of trial results is represented by the shaded regions.

$m = 4$ is not apparent.

The direct prediction results for the Mackey-Glass system (Figure 5.8) show a clear and consistent trend: the ORSESN produces a consistently lower mean RMSE, with $m = 4$ achieving a significantly (95%CI) superior result across all tested prediction lengths. This contrasts with the iterative prediction setting, where the advantage of higher m is limited to shorter horizons and can degrade at longer ones; direct prediction has no crossover or degradation of performance for $m = 2$ or $m = 3$ at longer steps.

These results suggest that while the ORSESN with $m = 2$ and $m = 3$ can initially outperform the traditional ESN for medium length iterative predictions, they may be less effective at correcting for accumulating errors as the prediction length grows. This behavior leads to a crossover where the traditional ESN eventually matches or surpasses their performance.

However, this limitation is mitigated when the ordinal pattern dimension is increased

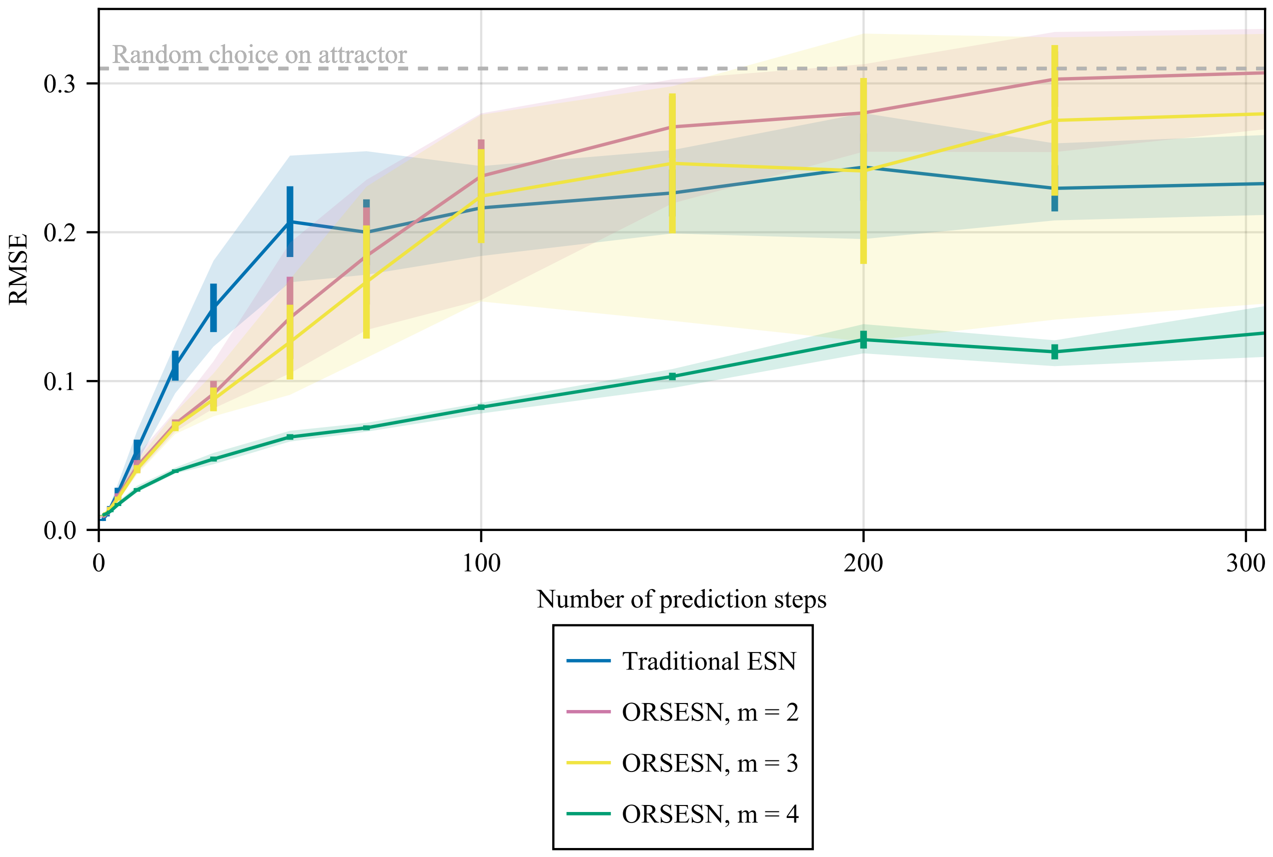


Figure 5.6: Iterative prediction RMSE for the ORSESN for Mackey-Glass time series with $\Delta t = 0.5$, delay $\tau = 20$, additive noise $\alpha = 0.1$ and total reservoir size $k = 500$. The standard deviations between trials are represented by vertical bars and the range of trial results is represented by the shaded regions.

to $m = 4$. The ORSESN with $m = 4$ not only maintains its initial advantage but also demonstrates sustained accuracy and stability across all tested prediction horizons. The mean RMSE for the $m = 4$ ORSESN is 66% lower than the traditional ESN at 10 direct prediction steps, and 51% lower at 100 direct prediction steps. This indicates that a higher ordinal pattern dimension enables the ORSESN to capture more relevant dynamical information, resulting in both improved short term accuracy and greater resilience to error accumulation in long term predictions.

Gating

The ORSESN introduces a new architectural feature, the readout gating mechanism, and drives the gating using ordinal partition information of the incoming data. We would like to confirm whether it is the ordinal partition information that is creating the improved predictions rather than just the gating mechanism itself. To test this, we

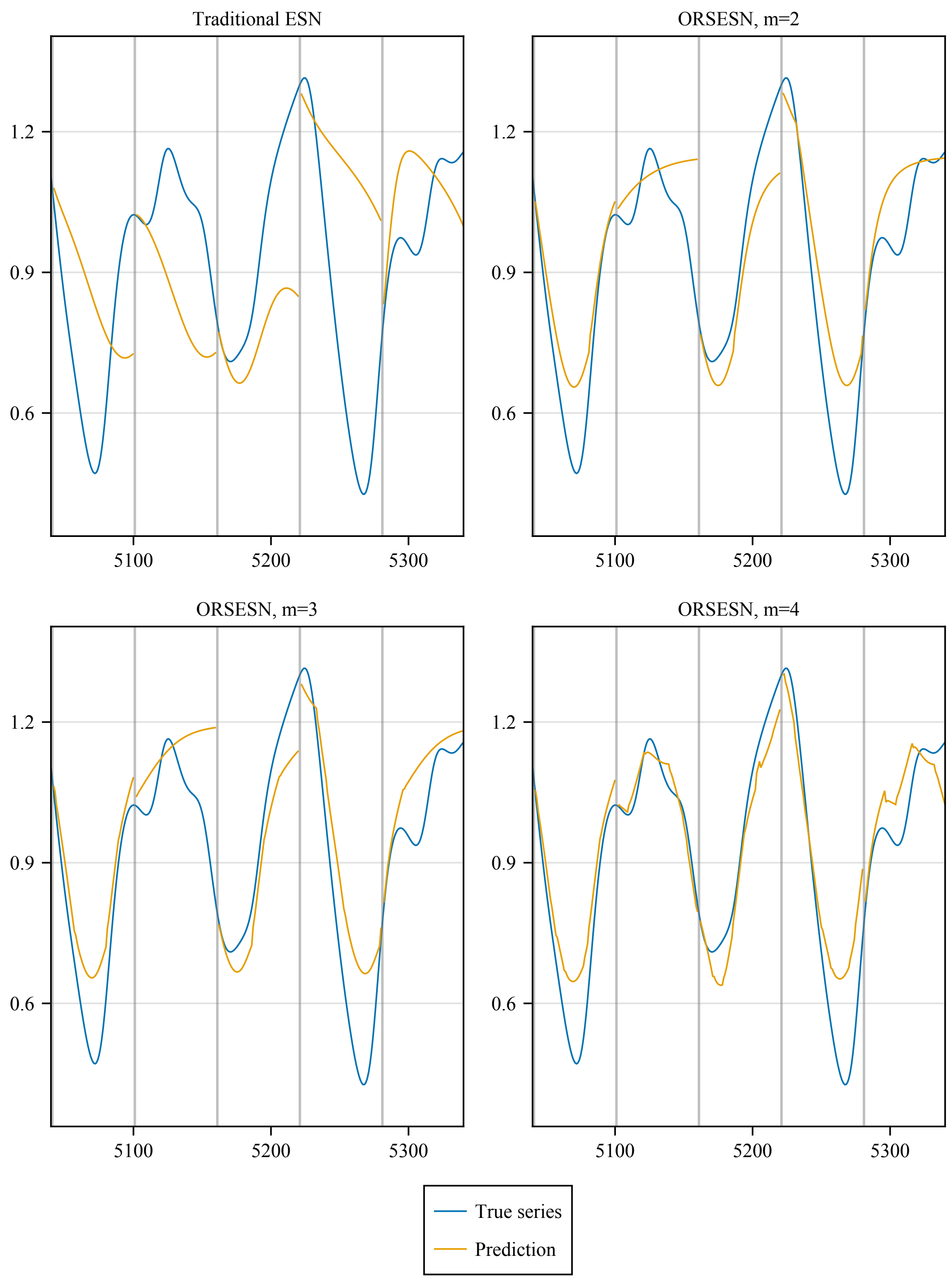


Figure 5.7: Example of iterative prediction for the Mackey-Glass system ($\Delta t = 0.5$, additive noise $\alpha = 0.1$). Predictions from the traditional ESN ($m = 1, \tau = 1$) and ORSESN ($m \in \{2, 3, 4\}, \tau = 20$) are compared against the true series. All models use a total reservoir size $k = 500$. Each grey vertical line indicates the beginning of prediction chunk, where the echo state network changes from being driven by the true signal to being driven by its own predictions.

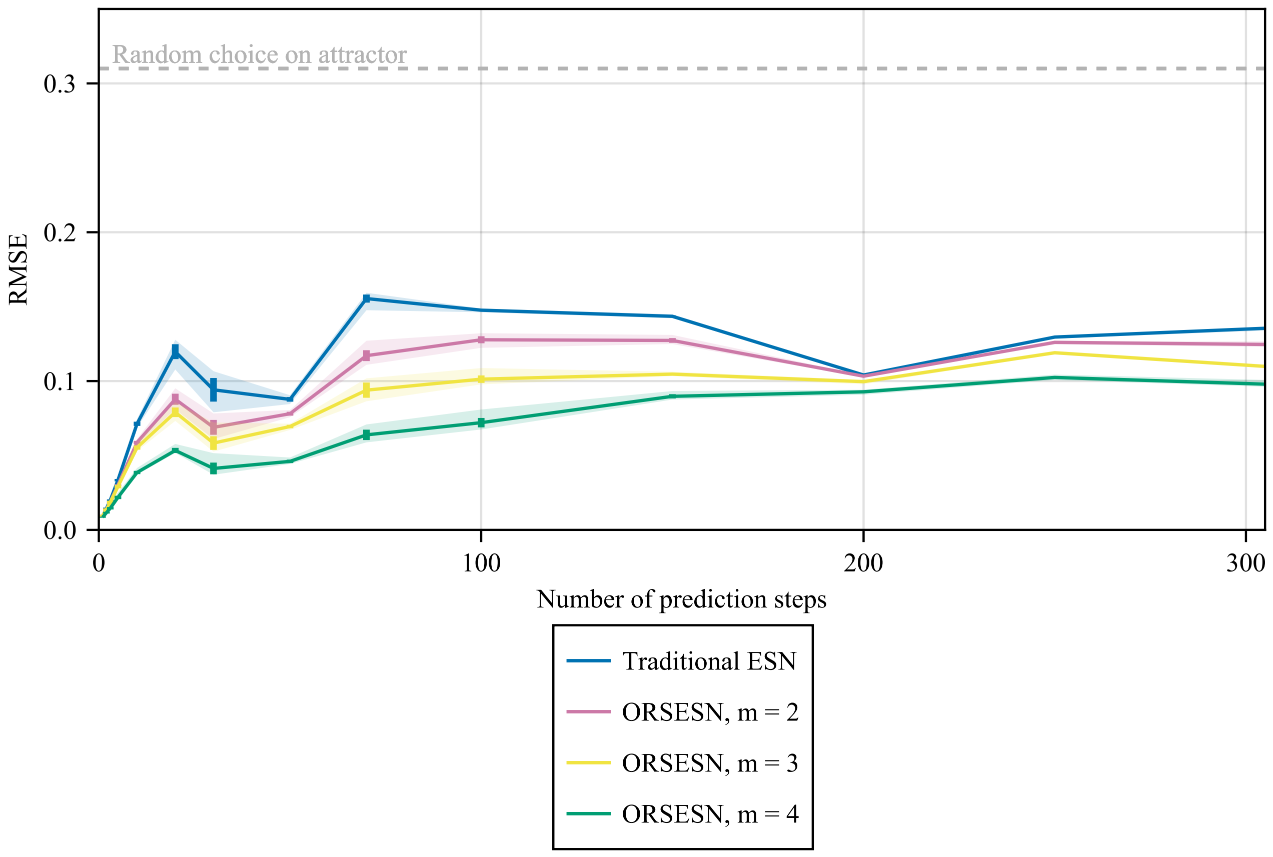


Figure 5.8: Mean direct prediction RMSE for the ORSESN for Mackey-Glass time series with $\Delta t = 0.5$, delay $\tau = 20$, additive noise $\alpha = 0.1$ and total reservoir size $k = 500$. The standard deviations between trials are represented by vertical bars and the range of trial results is represented by the shaded regions.

supplied the ORSESN with the Lorenz time series with $\Delta t = 0.01$ accompanied by a random selection of the readout and compared it to the readout selection driven by the ordinal partition. Figure 5.9 shows the performance of the ORSESN when its readout is gated via the ordinal partitions of the data compared to its performance when the gating of the readout is determined by random selection, for each of $m = 2, 3, 4$. The performance of the ORSESN with random gating is insignificantly (95% CI) different from the performance of our traditional ESN benchmark, with the mean RMSE value across trials lying within one standard deviation of the traditional ESN across all tested lengths of iterative prediction. We can infer from this testing that the predictive advantage of the ORSESN is likely conferred by the ordinal partition information rather than the use of the multiple readout vector gating mechanism by itself.

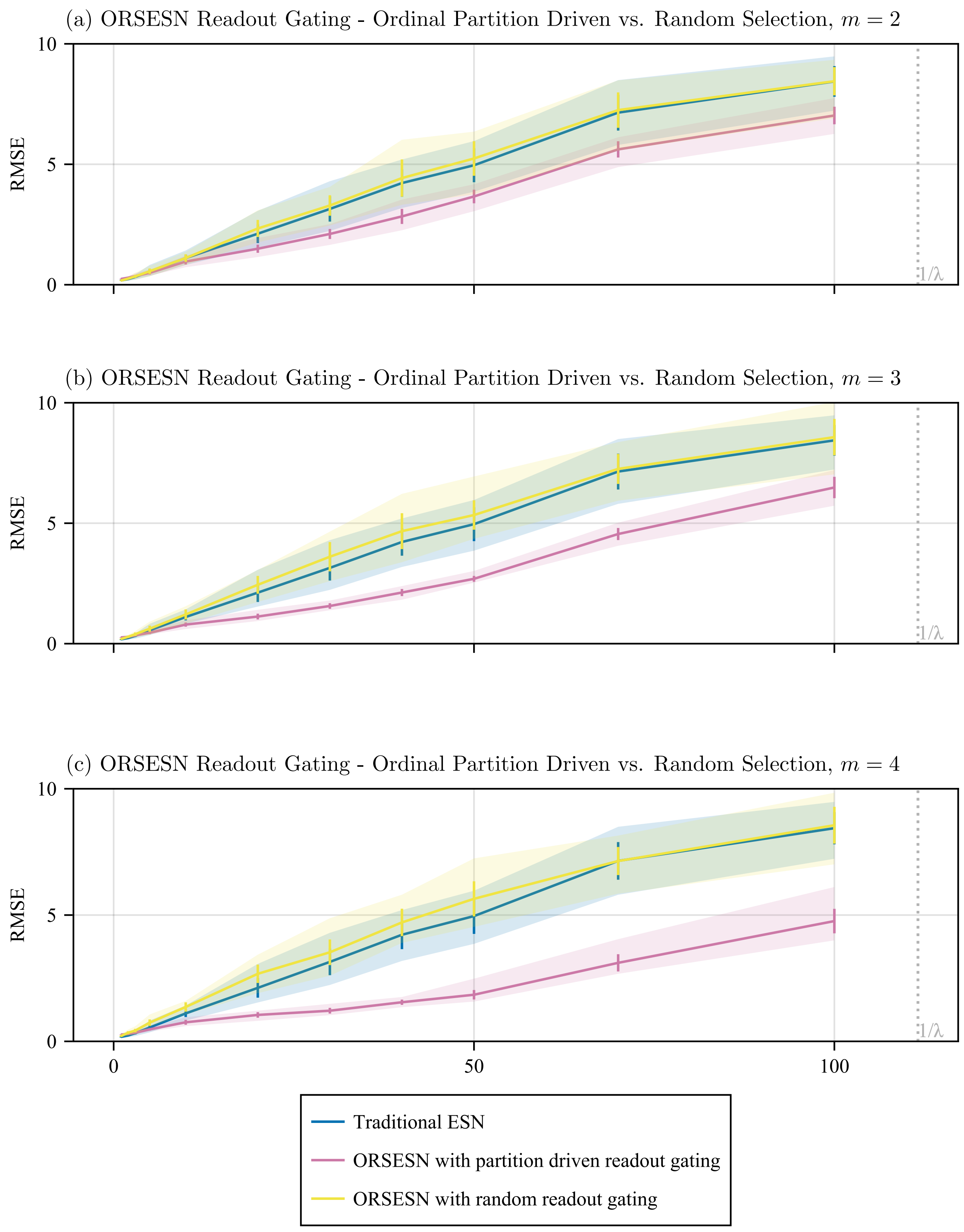


Figure 5.9: Mean iterative prediction RMSE across trials for the ORSESN on the Lorenz x-component time series with time step size $\Delta t = 0.01$, delay $\tau = 20$, additive noise $\alpha = 0.1$, and total reservoir size $k = 468$, compared to a variant that selects the readout randomly rather than by ordinal partition. Subfigures (a), (b), and (c) correspond to $m = 2$, $m = 3$, and $m = 4$, respectively. Standard deviations between trials are represented by vertical bars and shaded regions indicate the full range of trial results.

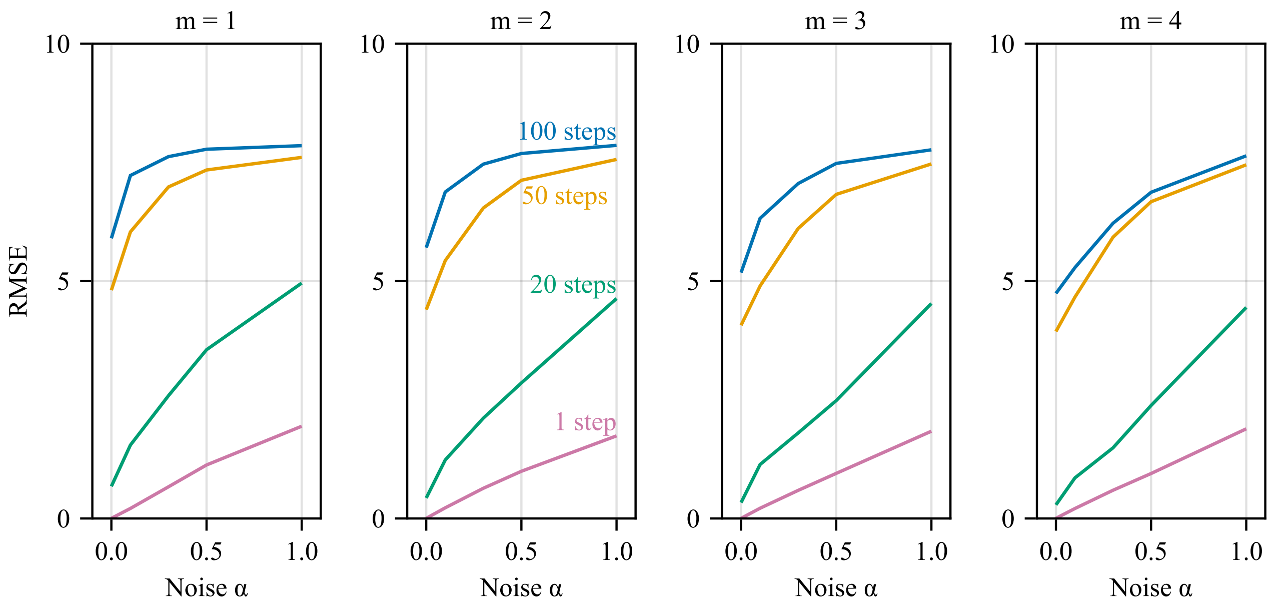


Figure 5.10: Mean direct prediction RMSE across trials for the ORSESN on the Lorenz x-component time series with time step size $\Delta t = 0.01$ as a function of simulated data noise level α, with delay $\tau = 20$ and total reservoir size $k = 468$. Curves correspond to different numbers of prediction steps, and each panel represents an ordinal pattern dimension m (with $m = 1$ denoting the traditional ESN).

Resilience to noise

We can test the resilience of models to noise by adding increasing levels of noise to the training data and testing the resulting prediction performance. We can then compare the effect of noise on the model's ability to create predictions. Figure 5.10 shows the mean direct prediction RMSE of multiple trials across values of noise $\alpha \in [0.0, 1.0]$, with four curves representing different numbers of prediction steps. As to be expected, as the level of noise increases so does the prediction RMSE for both the traditional ESN ($m = 1$) and the ORSESN models ($m = 2, 3, 4$). Compared to the traditional ESN, the ORSESN models appear to be less resilient to noise as the relative increase in RMSE is greater over noise levels 0.0 to 1.0. However the absolute mean RMSE is still lower (or approximately equal) for all $\alpha \in [0.0, 1.0]$ despite this lesser resilience to noise.

Figure 5.11 shows the same noise analysis performed with iterative prediction instead of direct prediction. Iterative prediction yields similar results, where the traditional ESN appears to be more resilient to noise however the ORSESN has a lower (or approximately the same) mean prediction RMSE for $0.0 \leqslant \alpha \leqslant 1.0$. The reader can see that as the noise level increases, the predictive performance initially improves between $\alpha = 0.0$ and $\alpha = 0.1$

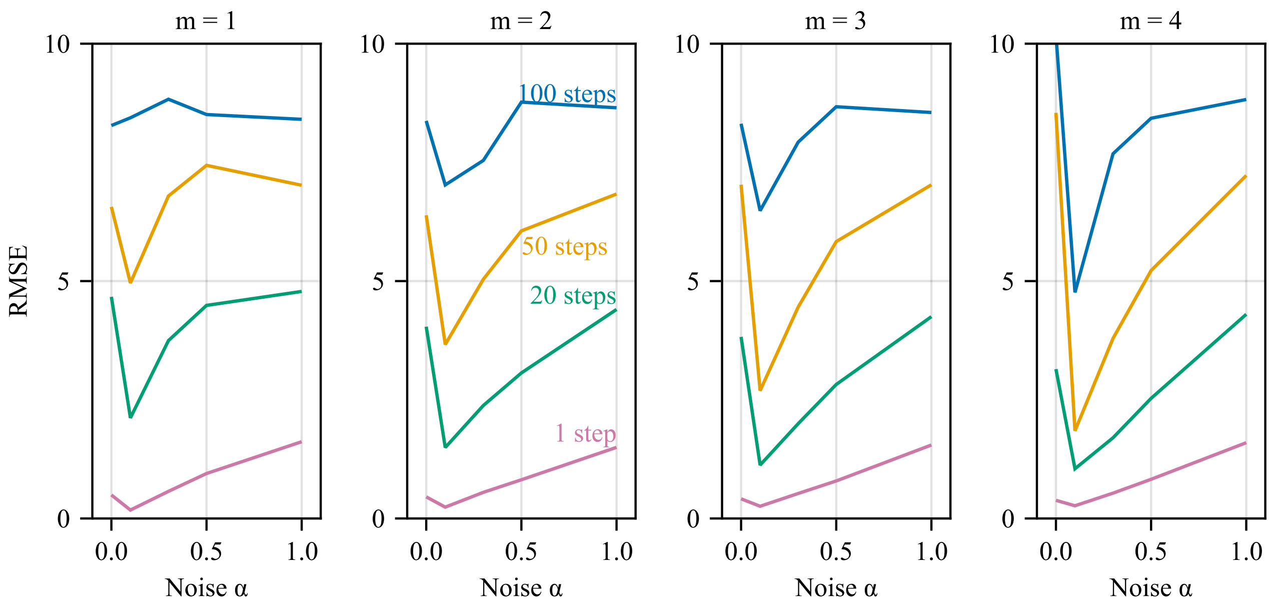


Figure 5.11: Mean iterative prediction RMSE across trials for the ORSESN on the Lorenz x-component time series with time step size $\Delta t = 0.01$ as a function of simulated data noise level α, with delay $\tau = 20$ and total reservoir size $k = 468$. Curves correspond to different numbers of prediction steps, and each panel represents an ordinal pattern dimension m (with $m = 1$ denoting the traditional ESN).

before degrading between $\alpha = 0.1$ and $\alpha = 1.0$. This 'sweet spot' has been observed previously, where a modest amount of noise in training data lowers the prediction error of an ESN when producing iterative predictions [15][29]. The benefit has been interpreted as a 'mild stochastic regularisation', broadening the set of states seen during training so that the readout is less likely to compound small errors during iterative prediction. For the purposes of testing the performance of the traditional ESN and the ORSESN we have chosen the 'sweet spot' value of $\alpha = 0.1$.

In an effort to explain this result, one should note that ordinal partitions are invariant to monotonic transformations but they are not immune to misclassification when the signal is noisy. Figures 5.10 and 5.11 show that prediction error grows faster with α for ORSESN than for the traditional ESN, because additive noise can flip the relative order of nearby samples and therefore trigger an incorrect readout. In other words, as we increase α we are pushing the gating mechanism to become random - the effects of which have been tested above. Nevertheless ORSESN preserves an absolute advantage up to $\alpha = 1.0$ in the present setting, implying that its gains outweigh the extra sensitivity unless measurement noise is extreme. For applications where noise cannot be pre-filtered, a softened gating strategy such as weighting readouts by the posterior probability of each partition rather than enforcing a hard mask may retain accuracy while dampening the noise penalty.

Extended discussion

Across all three benchmark systems, the traditional ESN delivers the lowest error at the one-step horizon, but the ORSESN with ordinal dimension $m = 4$ steadily outperforms it for medium and long-term forecasts. We suggest this arises from the different ways each model embeds past inputs: the ESN's echo state property produces a fading-memory embedding that weights recent samples more than past samples, while a backward ordinal pattern of length $m\tau + 1$ encodes all samples in its window equally. In the ORSESN, separate readouts are trained on states gated by each ordinal partition - reducing the data per readout but ensuring that each specializes on homogeneous dynamical conditions. For short predictions, the ESN's single readout trained on the full set of reservoir states interprets the dynamics of the recency biased reservoir more effectively. By contrast, the ORSESN's partitioned readouts each receive less training data and may underfit, resulting in poorer short-term accuracy. For extended horizons, however, the ORSESN readouts capture longer-timescale structure, counteract the ESN's biased memory, and thus yield more accurate medium and long-term predictions.

The random-gating experiment (Figure 5.9) confirms that the improvement stems from the ordinal structure, not from the mere presence of multiple readouts. Thus the ORSESN not only provides strong evidence that ordinal partitions are an effective predictive feature for ESNs, but is also an effective architecture to take advantage of this feature.

Chapter 6

Echo state network with ordinal partition based sub-reservoirs

The second proposal uses a restricted reservoir and routed input to structurally encode the information given by ordinal partitioning of the data into the model. We will call this model an 'Ordinal Partition Echo State Network' or OPESN. The reservoir consists of N_{obs} sub-reservoirs of size k, where N_{obs} is the number of non-forbidden ordinal partitions of the training sequence. In the primary case, the sub-reservoirs are connected 'one-to-one', with one node of each sub-reservoir possibly connected to one node of each other sub-reservoir. The connections from one sub-reservoir to another sub-reservoir are scaled according to the probability of transitioning between them. We also test multiple other schemes for connecting the sub-reservoirs.

An input is routed to the relevant sub-reservoir depending on the ordinal partition of the observation. If an input observation is from ordinal partition '3' then it will be used to update the states of sub-reservoir '3' and an input value of 0 will be used to update the states of the other sub-reservoirs. We will refer to the sub-reservoir to which input is routed at any time step as the 'active sub-reservoir'. Figure 6.1 provides a diagram of the OPESN.

6.1 Implementation

Restricting the Recurrence Matrix

To construct the recurrence matrix of the OPESN, an ordinal sequence should be derived from the training time series used to fit the readout vector such that each data point has an associated ordinal partition. We must then count the number of unique ordinal partitions that occur in the sequence, a number we will refer to as N_{obs}. Each ordinal partition is assigned an ordinal symbol $\pi \in 1, ...N_{obs}$, which should be unique each between 1 to N_{obs} to

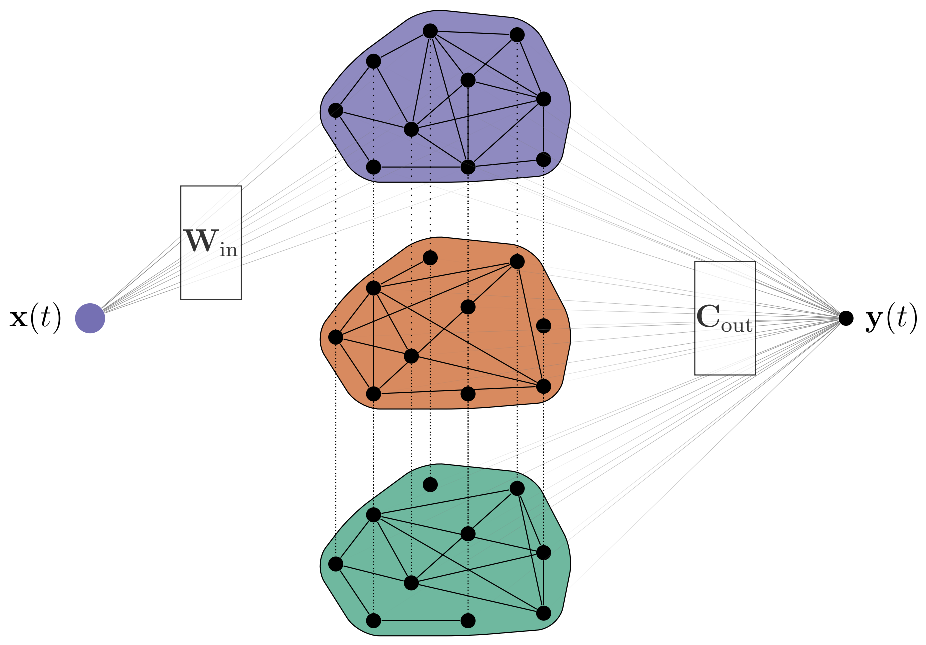


Figure 6.1: Architecture of the ordinal partition echo state network (OPESN). The reservoir is partitioned into N_{obs} sub-reservoirs of size k. Input $\mathbf{x}(t)$ is routed to the active sub-reservoir based on the ordinal partition π_t (colour coded), and a single global readout $\mathbf{C}_{\text{out}}$ produces $\mathbf{y}(t)$.

allow for simple indexation of the recurrence weight matrix. To contruct the connections between sub-reservoirs, we will also need to calculate the transition probabilities between the ordinal partitions of the training sequence. We assume that the training sequence is sufficiently long such that all partitions observed in the testing sequence will have been observed in the training sequence. We acknowledge that this presents a limitation to this architecture, and this is not addressed in this research. It could be resolved by instantiating the network with all possible ordinal partitions or by implementing a 'growing' ESN with multiple sub-reservoirs as described by Qiao et al. [37]. However with a sufficient amount of additive noise, all possible partitions will be visited in any case.

We can then instantiate a random network connection matrix W_{rec} with size kN_{obs} to prepare for N_{obs} sub-reservoirs of size k. This connection matrix can be instantiated with many methods, but we will use an Erdős-Rényi random network. Let $\mathbf{W}_{i,j}$ refer to the submatrix of $\mathbf{W}_{rec}$ with indices

$$\begin{aligned} \text{Rows:} \quad & \{ik+1, ik+2, \ldots, (i+1)k\}, \\ \text{Columns:} \quad & \{jk+1, jk+2, \ldots, (j+1)k\}. \end{aligned}$$

Then the resulting matrix $\mathbf{W}_{rec}$ can be defined in block form where each $\mathbf{W}_{i,j}$ for $i, j \in 1, 2 \ldots n$ represents either the internal weights of each sub-reservoir (when $i = j$) or the connections between these sub-reservoirs (when $i \neq j$):

$$\mathbf{W}_{rec} = \begin{bmatrix} \mathbf{W}_{1,1} & \mathbf{W}_{1,2} & \mathbf{W}_{1,3} & \cdots & \mathbf{W}_{1,N_{obs}} \\ \mathbf{W}_{2,1} & \mathbf{W}_{2,2} & \mathbf{W}_{2,3} & \cdots & \mathbf{W}_{2,N_{obs}} \\ \mathbf{W}_{3,1} & \mathbf{W}_{3,2} & \mathbf{W}_{3,3} & \cdots & \mathbf{W}_{3,N_{obs}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{W}_{N_{obs},1} & \mathbf{W}_{N_{obs},2} & \mathbf{W}_{N_{obs},3} & \cdots & \mathbf{W}_{N_{obs},N_{obs}} \end{bmatrix}.$$

In the initial formulation that we tested, the connections between each node in the sub-reservoirs are created as one-to-one, set equal to the transition probability of the two ordinal partitions. To achieve this, the connection matrices between sub-reservoirs are set to $p_{ij}\mathbf{I}$ where p_{ij} is the probability of transitioning from ordinal partition i to j and $\mathbf{I}$ is the $k \times k$ identity matrix, while the weight matrices are left as the relevant submatrix of the original Erdős-Rényi randomly instantiated network $\mathbf{W}_{ER_{i,j}}$:

$$\mathbf{W}_{rec} = \begin{bmatrix} \mathbf{W}_{ER_{1,1}} & p_{1,2}\mathbf{I} & p_{1,3}\mathbf{I} & \cdots & p_{1,N_{obs}}\mathbf{I} \\ p_{2,1}\mathbf{I} & \mathbf{W}_{ER_{2,2}} & p_{2,3}\mathbf{I} & \cdots & p_{2,N_{obs}}\mathbf{I} \\ p_{3,1}\mathbf{I} & p_{3,2}\mathbf{I} & \mathbf{W}_{ER_{3,3}} & \cdots & p_{3,N_{obs}}\mathbf{I} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ p_{N_{obs},1}\mathbf{I} & p_{N_{obs},2}\mathbf{I} & p_{N_{obs},3}\mathbf{I} & \cdots & \mathbf{W}_{ER_{N_{obs},N_{obs}}} \end{bmatrix}.$$

More succinctly, we can define each submatrix $\mathbf{W}_{i,j}$ in $\mathbf{W}_{rec}$ by:

$$\mathbf{W}_{i,j} = \begin{cases} p_{i,j}\mathbf{I}, & i \neq j, \\ \mathbf{W}_{ER_{i,j}}, & i = j. \end{cases}$$

Secondly, we trialed a 'fully connected' sub-reservoir, where each submatrix $\mathbf{W}_{i,j}$ in $\mathbf{W}_{rec}$ can be defined by

$$\mathbf{W}_{i,j} = \begin{cases} p_{i,j}\mathbf{1}, & i \neq j, \\ \mathbf{W}_{ER_{i,j}}, & i = j, \end{cases}$$

where $\mathbf{1}$ represents a $k \times k$ matrix of 1. To isolate the effect of the connections between sub-reservoirs, we also tested disconnected sub-reservoirs:

$$\mathbf{W}_{i,j} = \begin{cases} \mathbf{0}, & i \neq j, \\ \mathbf{W}_{ER_{i,j}}, & i = j, \end{cases}$$

where $\mathbf{0}$ represents a $k \times k$ matrix of 0. And to isolate the effect of the scaling of the connections between sub-reservoirs according to transition probabilities, we tested constant-value

	🟣	🟠	🟢
🟣	0.7	0.1	0.2
🟠	0.9	0.1	0
🟢	0.1	0	0.8

Table 6.1: Example transition probabilities used to form our restricted recurrence matrix W_{rec}.

connections:

$$\mathbf{W}_{i,j} = \begin{cases} \mathbf{0}, & i \neq j \text{ and } p_{i,j} = 0, \\ c\mathbf{I}, & i \neq j \text{ and } p_{i,j} > 0, \\ \mathbf{W}_{ER_{i,j}}, & i = j, \end{cases}$$

where c is some constant $-1 \leqslant c \leqslant 1$, for which we tested the mean value of $\mathbf{W}_{rec}$ and also values randomly drawn from $\mathcal{N}(0,1)$. We also tested fully connected sub-reservoirs using a constant value:

$$\mathbf{W}_{i,j} = \begin{cases} \mathbf{0}, & i \neq j \text{ and } p_{i,j} = 0, \\ c\mathbf{1}, & i \neq j \text{ and } p_{i,j} > 0, \\ \mathbf{W}_{ER_{i,j}}, & i = j, \end{cases}$$

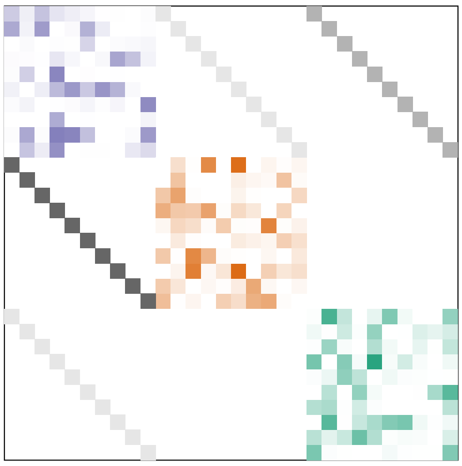
and sparsely connected sub-reservoirs:

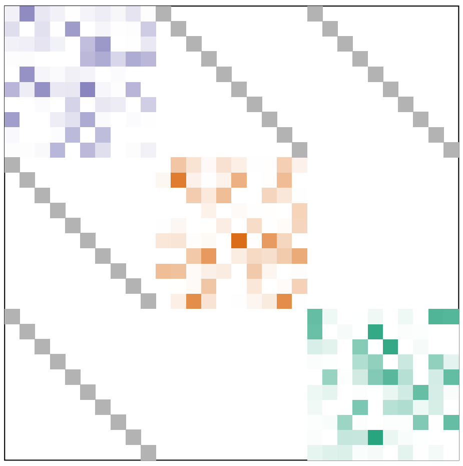
$$\mathbf{W}_{i,j} = \begin{cases} \mathbf{0}, & i \neq j \text{ and } p_{i,j} = 0, \\ \mathbf{W}_{ER_{i,j}}, & \text{otherwise.} \end{cases}$$

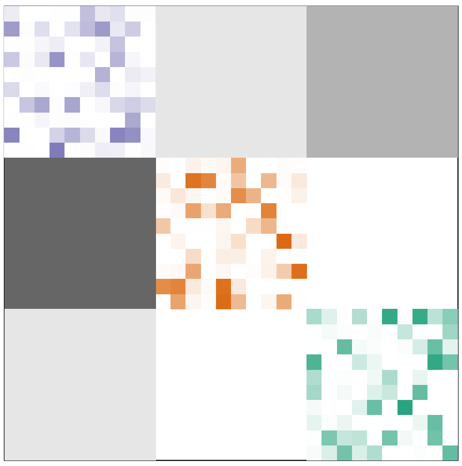
Additionally, we tested a recurrence matrix with constant one-to-one connections between sub-reservoirs that are stochastically activated at each time step, this is described in Appendix A. Figure 6.2 visualises an example of some of these definitions of $\mathbf{W}_{rec}$ using the same method as in chapter 2, using the example ordinal transition probabilities given in Table 6.1.

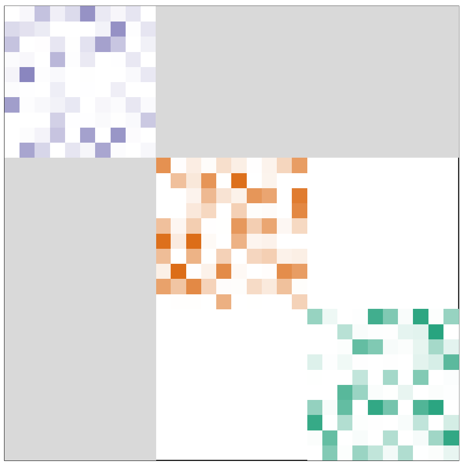
Input Routing

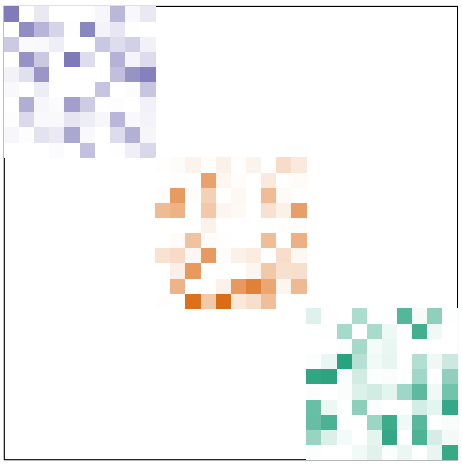
To feed the input into only one relevant layer, we mask all values of $\mathbf{W}_{in}$ except those in the relevant partition. We will randomly instantiated an input vector $\mathbf{W}_{in}$ of size $k_{part}N_{obs}$ by draws from a Normal distribution. Thus, we can define each element of our masked input vector $\mathbf{W}_{masked}$ as a piecewise function of π_t, the ordinal partition of the input at


(a) One-to-one connections between sub-reservoirs scaled according to transition probabilities.


(b) Sub-reservoirs with constant value one-to-one connections.


(c) Fully connected sub-reservoirs with connections scaled according to transition probabilities.


(d) Sub-reservoirs with constant value fully connected connections.


(e) Disconnected sub-reservoirs, running in parallel.

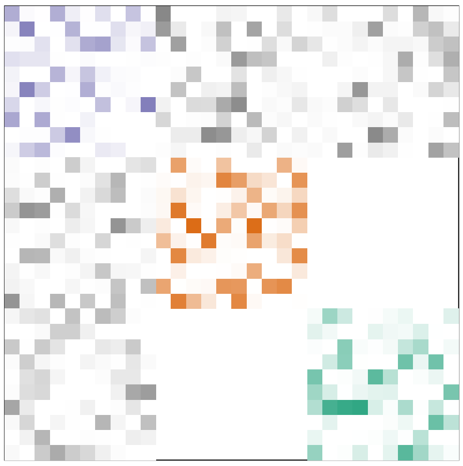

(f) Sub-reservoirs with sparse connections.

Figure 6.2: Visualisations of the recurrence matrix $\mathbf{W}_{rec}$ under six connection schemes for a hypothetical OPESN with three partitions: (a) one-to-one weighted by p_{ij}; (b) constant one-to-one; (c) fully connected weighted by p_{ij}; (d) constant fully connected; (e) disconnected; and (f) sparsely connected. Shading intensity reflects relative weight magnitudes and internal weights for each sub-reservoir are colour-coded.

time t:

$$\mathbf{W}_{masked}(\pi_t)_i = \begin{cases} (\mathbf{W}_{in})_i, & k(\pi_t - 1) < i \leqslant k\pi_t \\ 0, & \text{otherwise.} \end{cases}$$

To apply this to an example, consider the input vector

$$\mathbf{W}_{in} = \begin{bmatrix} 0.12 & 0.5 & 0.34 & -0.67 & 0.1 & -0.43 & 0.98 & -0.64 & 0.2 & 0.45 & 0.2 & -0.45 \end{bmatrix},$$

then masking the input vector for each $\pi \in [\pi_1, \pi_2, \pi_3]$ will yield

$$\mathbf{W}_{masked}(\pi_1) = \begin{bmatrix} 0.12 & 0.5 & 0.34 & -0.67 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{W}_{masked}(\pi_2) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0.1 & -0.43 & 0.98 & -0.64 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{W}_{masked}(\pi_3) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.2 & 0.45 & 0.2 & -0.45 \end{bmatrix}.$$

To isolate the effect of the ordinal partition information, we compared the ordinal partition based input routing against a random routing of the input at each time step. In this random routing model, the active partition is chosen uniformly.

Iterating the reservoir and fitting the readout vector

Once the reservoir has been instantiated and restricted to sub-reservoirs according to ordinal partition, and the piecewise input function has been defined, we can drive the reservoir with the input as we would a traditional ESN:

$$\mathbf{s}(t+1) = f_{act}(\mathbf{W}_{masked}(\pi_t)\mathbf{x}(t) + \mathbf{W}_{rec}\mathbf{s}(t) + \mathbf{W}_{bias}).$$

We can then fit the readout vector and infer predictions using the OPESN with the same method as a traditional ESN:

$$\mathbf{C}_{out}^T = \left(\mathbf{S}^T\mathbf{S} + \beta\,\mathbf{I}\right)^{-1}\mathbf{S}^T\,\mathbf{Y},$$
$$y(t) = \mathbf{C}_{out}\,\mathbf{s}(t).$$

We can also mask the sub-reservoirs belonging to the inactive partitions for each $\mathbf{s}(t)$ prior to fitting and applying the readout vector, and then do the same again at inference. Let $\mathbf{s}_{masked}(t, \pi_t)$ be the masked states of the reservoir as a function of the partition at time t, π_t. We can define each element of $s_{masked}(t, \pi_t)$ similarly to $\mathbf{W}_{masked}(\pi_t)$:

$$\mathbf{s}_{masked}(t, \pi_t)_i = \begin{cases} \mathbf{s}(t)_i, & k(\pi_t - 1) < i \leqslant k\pi_t \\ 0, & otherwise. \end{cases}$$

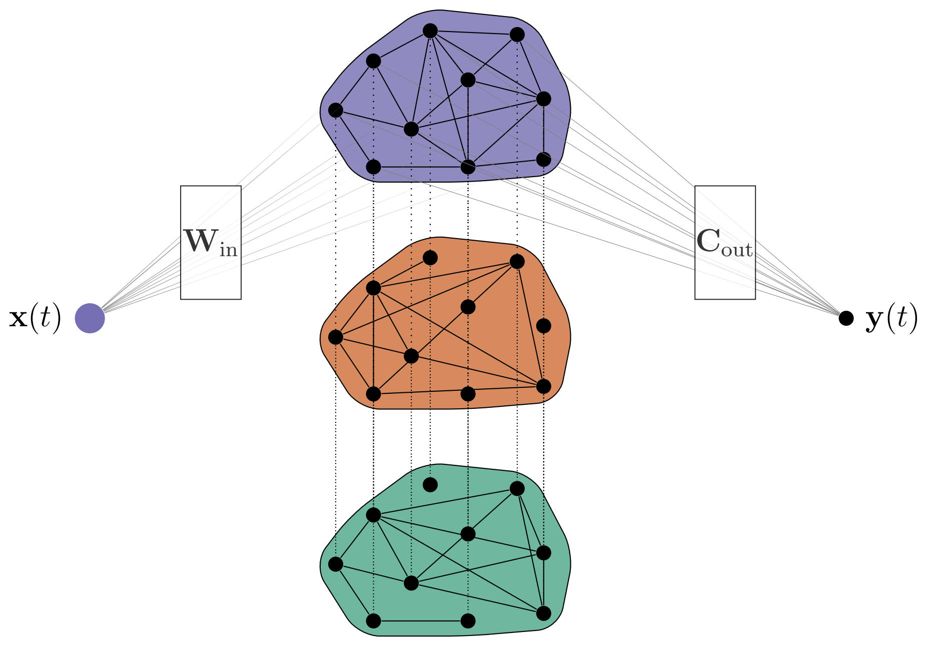


Figure 6.3: OPESN state masking prior to readout. Only the k states of the active sub-reservoir (purple) remain; all other states are zeroed before applying $\mathbf{C}_{\text{out}}$.

Then we can fit the readout vector for the OPESN:

$$y(t) = \mathbf{C}_{out}\mathbf{s}_{masked}(t, \pi_t).$$

Figure 6.3 illustrates the OPESN architecture with state masking. We experimented with the two approaches: masking the reservoir states according to the active ordinal partition before applying the readout vector, versus applying the readout vector to all states as in a standard ESN. Our results showed that the state masking approach led to significantly poorer performance compared to both the non-masked OPESN and the traditional ESN. Consequently, we adopted the non-state masking OPESN for all subsequent experiments.

The reader might notice that the state masking approach would create a structure similar to the ORSESN, with additional switching of $\mathbf{W}_{rec}$. However the input routing into separate sub-reservoirs provides a key difference.

6.2 Results

Lorenz - iterative prediction

The testing of the OPESN for iterative prediction of the Lorenz time series ($\Delta t = 0.01$) showed very limited benefit for longer time horizons, as seen in Figure 6.4a. The OPESN with $m = 2$ showed significant (95% CI) improvement for predictions of 100 or longer time steps. However this improvement appears to reflect a slight improvement in matching the periodicity of the attractor rather than any superior ability to predict performance.

When the time step size is increased to $\Delta t = 0.05$, the mild advantage of the OPESN models becomes insignificant (95% CI). As shown in Figure 6.4b, the traditional ESN and OPESN models show similar performance across prediction lengths, with the mean RMSE values closely aligned. Overall, while the OPESN with $m = 2$ shows some limited benefit for longer prediction on the $\Delta t = 0.01$ time step Lorenz series, its advantage is marginal and becomes insignificant with a longer time step.

Lorenz - direct prediction

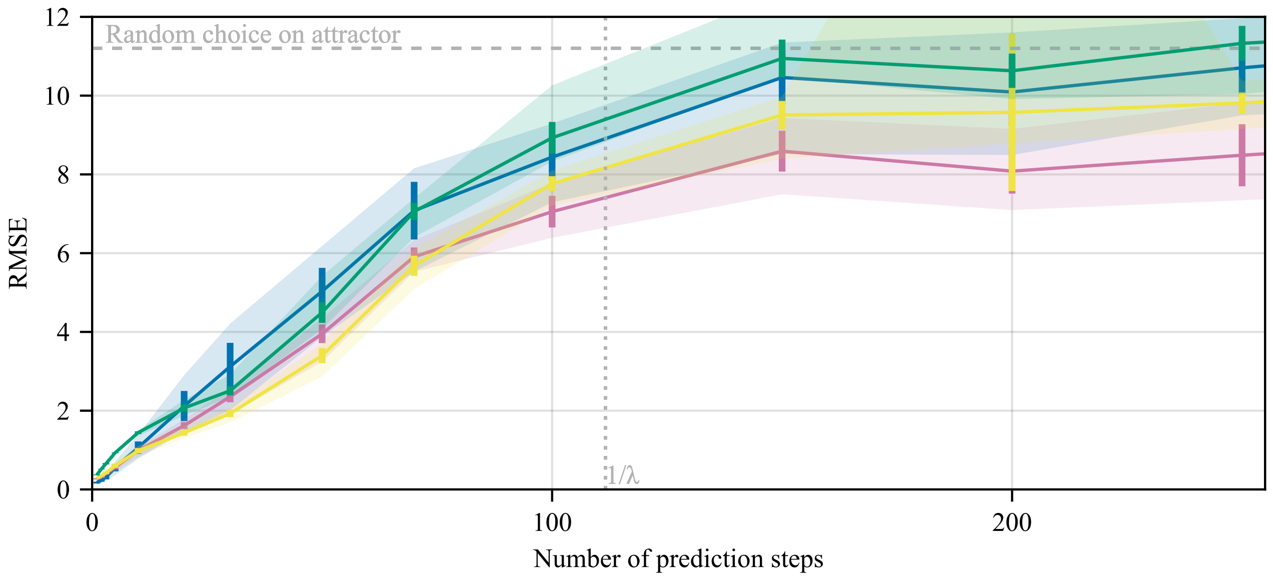
The direct prediction results further confirm the findings that the OPESN only marginally improves upon the traditional ESN at some prediction lengths (Figure 6.5a). For a time step size of $\Delta t = 0.01$, the traditional ESN consistently outperforms the OPESN up to 40 prediction steps. However, it is noteworthy that for 50 and 100 direct prediction steps, the OPESN with $m = 2, 3$ show a statistically significant (95% CI) improvement. Additionally, the OPESN with $m = 4$ shows a significantly (95% CI) lower mean RMSE for 100 prediction steps, with the largest improvement of a mean RMSE of 6.480 (± 0.035) compared to the traditional ESN with 7.318 (± 0.069). Despite these few instances, overall the gains are marginal and do not represent a consistent advantage.

For the Lorenz time series with a time step size of $\Delta t = 0.05$, the traditional ESN shows comparable or better performance than the OPESN (Figure 6.5b). Although, the OPESN with $m = 2, 3$ has a significantly (95% CI) lower mean RMSE for 10 direct prediction steps, the gains in accuracy are marginal: The traditional ESN has a mean RMSE of 4.720 (± 0.046) compared to the OPESN $m = 2$ with 4.618 (± 0.036) and $m = 3$ with 4.610 (± 0.037). Overall, these results indicate that the OPESN does not provide a substantial advantage for the prediction of the Lorenz time series, and at most prediction horizons, the traditional ESN performs similarly or better.

Rössler

The recursive prediction results on the Rössler time series with time step size $\Delta t = 0.1$ are plotted in Figure 6.6. For short prediction horizons, the OPESN tends to have higher RMSEs than the traditional ESN. However, once the prediction length reaches 10

(a) Mean iterative prediction RMSE on the Lorenz series with time step size $\Delta t = 0.01$ for the OPESN with delay $\tau = 20$, noise level $\alpha = 0.1$, and total reservoir size $k \approx 500$.



(b) Mean iterative prediction RMSE on the Lorenz series with time step size $\Delta t = 0.05$ for the OPESN with delays $\tau = 10, 10, 4$ for $m = 2, 3, 4$ respectively, noise level $\alpha = 0.1$, and total reservoir size $k \approx 500$.

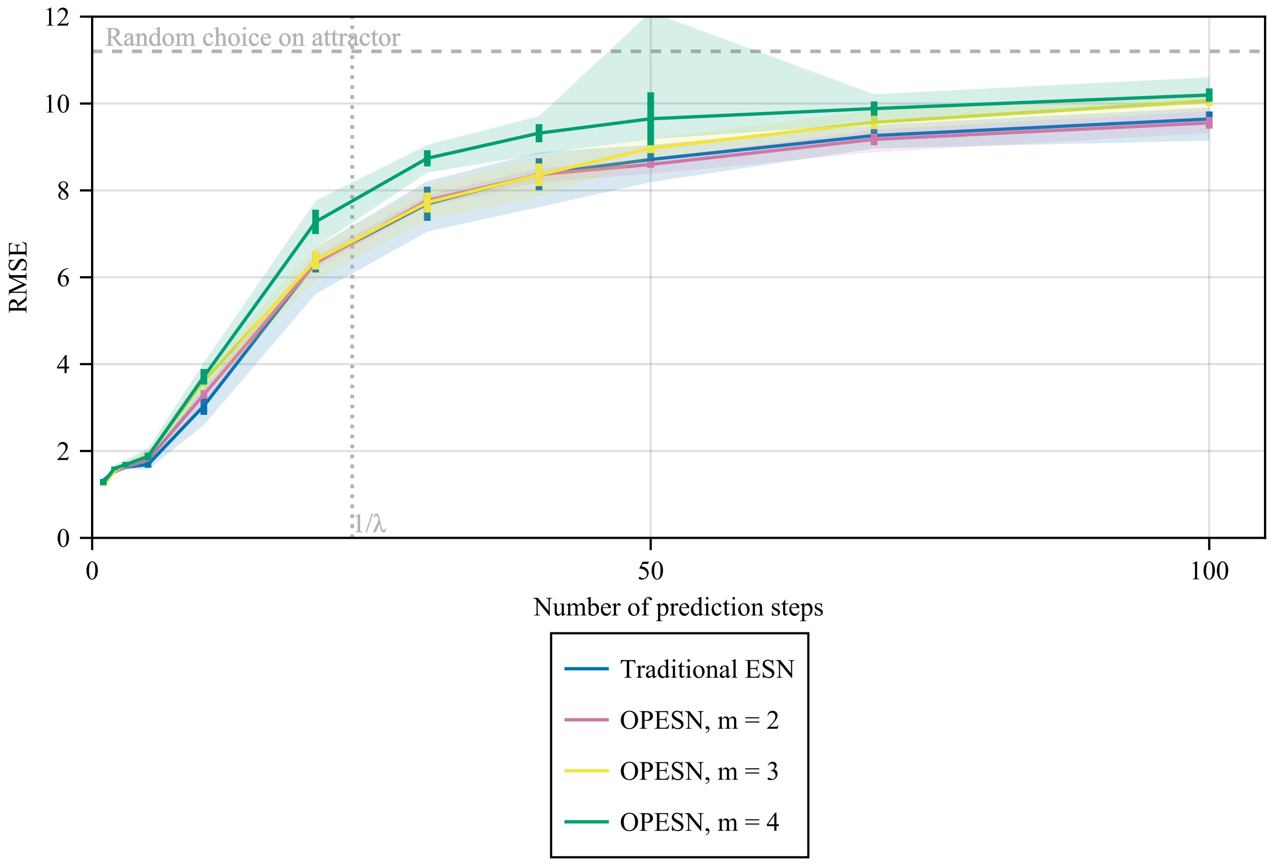
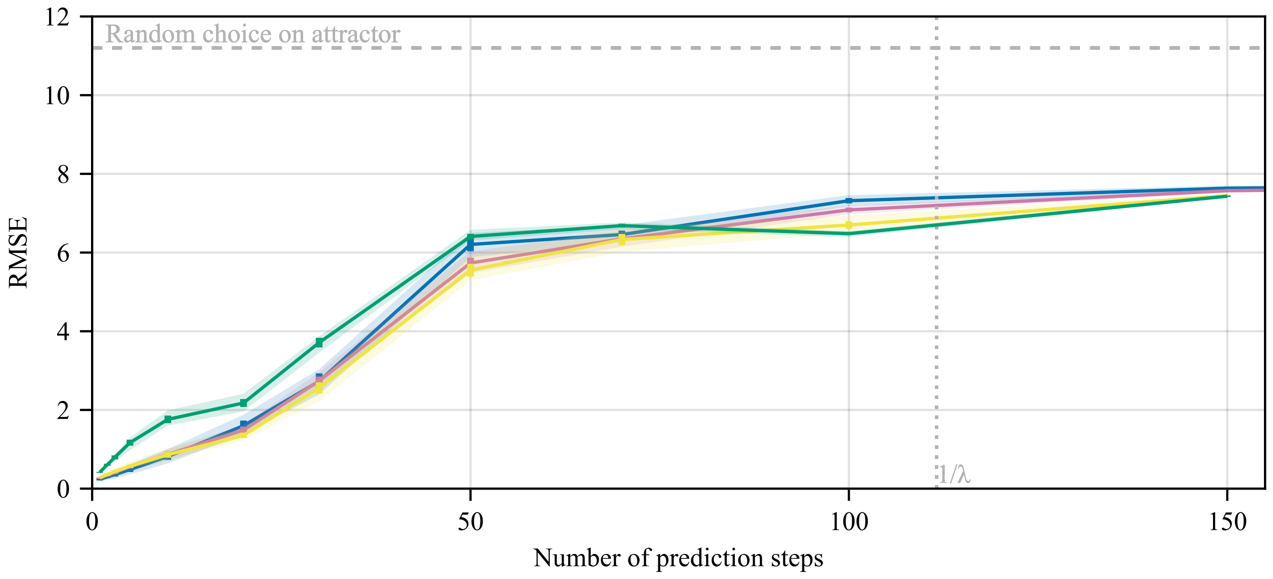


Figure 6.4: Mean iterative prediction RMSE for the OPESN on the Lorenz x component time series with additive noise $\alpha = 0.1$ and reservoir size $k \approx 500$. (a) $\Delta t = 0.01$, delays $\tau = 20$ for $m = 2, 3, 4$; (b) $\Delta t = 0.05$, delays $\tau = 10, 10, 4$ for $m = 2, 3, 4$. Vertical bars show standard deviations; shaded regions show full range.

(a) Mean direct prediction RMSE on the Lorenz series with time step size $\Delta t = 0.01$ for the OPESN with delay $\tau = 20$, noise level $\alpha = 0.1$, and total reservoir size $k \approx 500$.



(b) Mean direct prediction RMSE on the Lorenz series with time step size $\Delta t = 0.05$ for the OPESN with delays $\tau = 4$, noise level $\alpha = 0.1$, and total reservoir size $k \approx 500$.

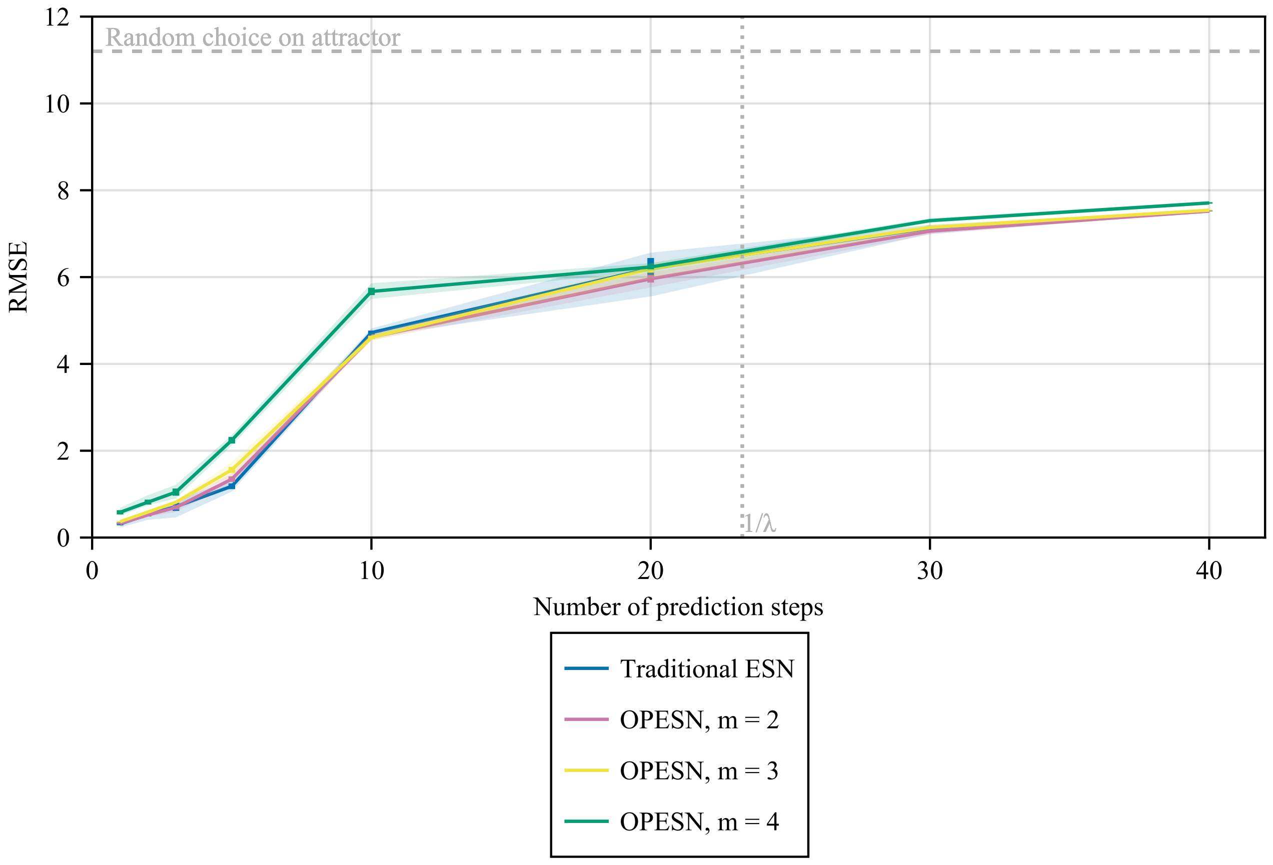


Figure 6.5: Mean direct prediction RMSE for the OPESN on the Lorenz x component time series with additive noise $\alpha = 0.1$ and reservoir size $k \approx 500$. (a) $\Delta t = 0.01$, delays $\tau = 20$ for $m = 2, 3, 4$; (b) $\Delta t = 0.05$, delays $\tau = 4$ for $m = 2, 3, 4$. Vertical bars show standard deviations; shaded regions show full range.

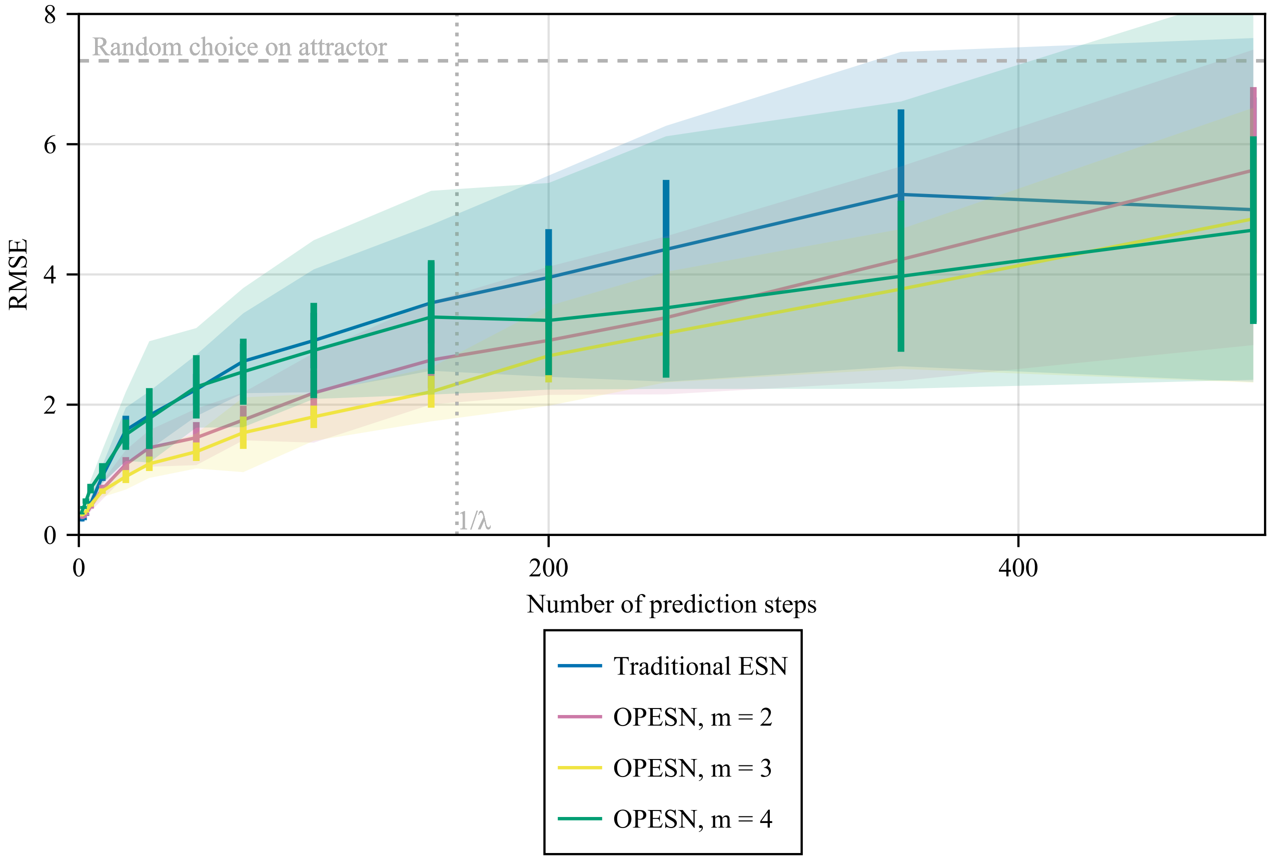


Figure 6.6: Mean iterative prediction RMSE for the OPESN on the Rössler x component time series ($\Delta t = 0.1$, $\alpha = 0.1$, $k \approx 500$). Delays $\tau = 20, 20, 10$ for $m = 2, 3, 4$, respectively. Vertical bars show standard deviations; shaded regions show full range.

steps, the OPESN models with $m = 2, 3$ begin to significantly (95% CI) outperform the traditional ESN, with the advantage being most pronounced for $m = 3$. For example, at 50 steps, the traditional ESN shows a mean RMSE of 2.234 (± 0.238) while the OPESN with $m = 3$ shows a substantial improvement with a mean RMSE of 1.278 (± 0.142). The advantage ceases to be significant after 150 prediction steps.

The direct prediction results (see Figure 6.7) corroborate the advantage of the OPESN with $m = 2, 3$, which are significantly (95% CI) more accurate after 30 prediction steps. While this advantage continues, the OPESN with $m = 4$ only has a significant advantage between 100 and 150 prediction steps. For example, at 100 direct prediction steps, the traditional ESN has a mean RMSE of 2.075 (± 0.059), while the OPESN with $m = 3$ achieves a lower mean RMSE of 1.666 (± 0.037). These results indicate that the OPESN models, particularly with $m = 3$, provide a modest improvement over the traditional ESN at longer prediction lengths on the Rössler series.

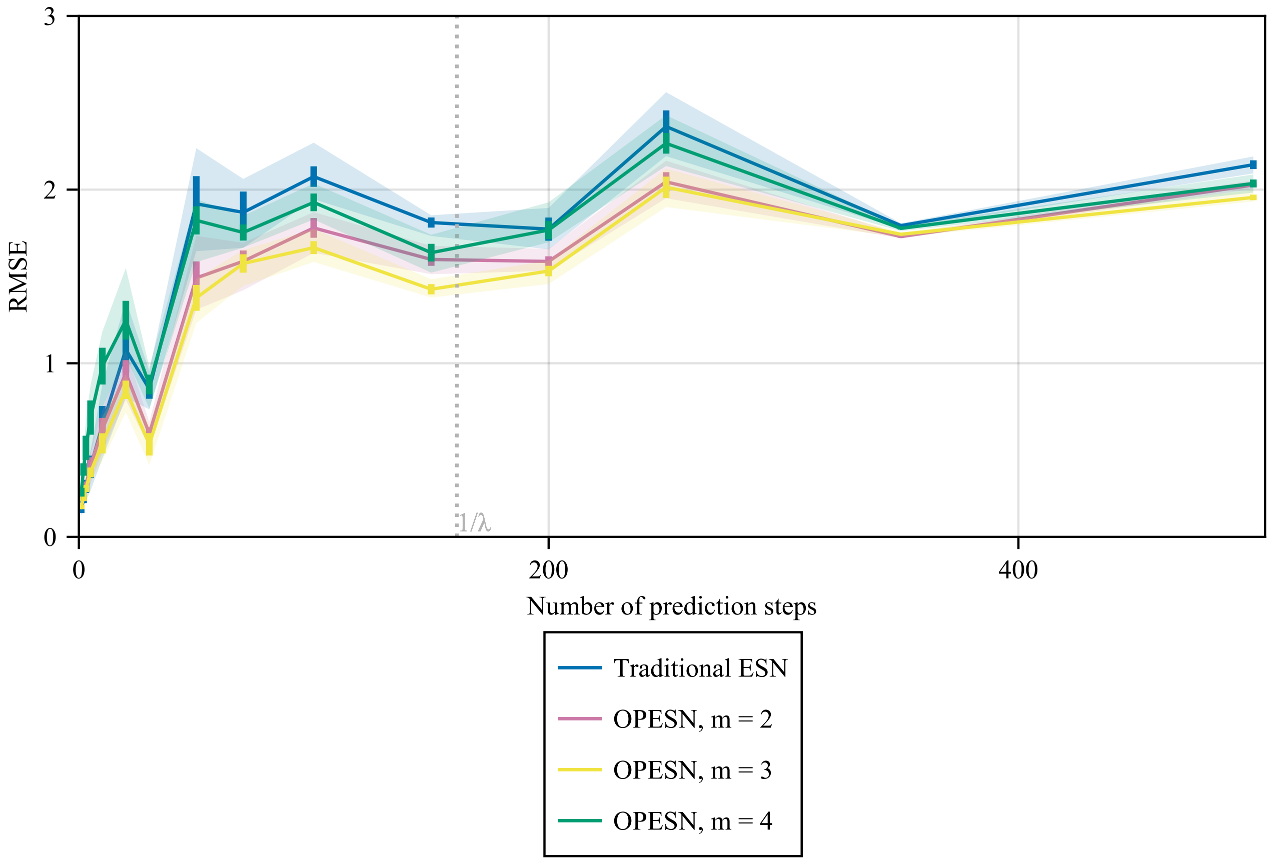


Figure 6.7: Mean direct prediction RMSE for the OPESN on the Rössler x component time series ($\Delta t = 0.1$, $\alpha = 0.1$, $k \approx 500$). Delays $\tau = 10, 20, 10$ for $m = 2, 3, 4$, respectively. Vertical bars show standard deviations; shaded regions show full range.

Mackey-Glass

The results of the OPESN prediction on the Mackey-Glass time series with time step size $\Delta t = 0.5$ are shown in Figure 6.8. The recursive prediction performance closely mirrors the pattern observed for the ORSESN: for short prediction horizons, the traditional ESN achieves the lowest mean RMSE, but as the prediction length increases beyond 10, the OPESN with $m = 2$ and $m = 3$ begin to outperform the traditional ESN. They maintain a significantly (95% CI) lower mean RMSE than the traditional ESN for prediction lengths shorter than 50 and 70 steps, at which they lose any significant advantage.

Surprisingly, and similar to the results seen for the ORSESN, the OPESN with $m = 4$ creates predictions with a substantially lower mean and standard deviation of RMSE than the other models at all prediction lengths longer than 10 steps. For example, the OPESN with $m = 4$ achieves a mean RMSE of 0.060 ($\pm$0.004) at 50 prediction steps compared to the traditional ESN with 0.208 ($\pm$0.030), and maintains this advantage for

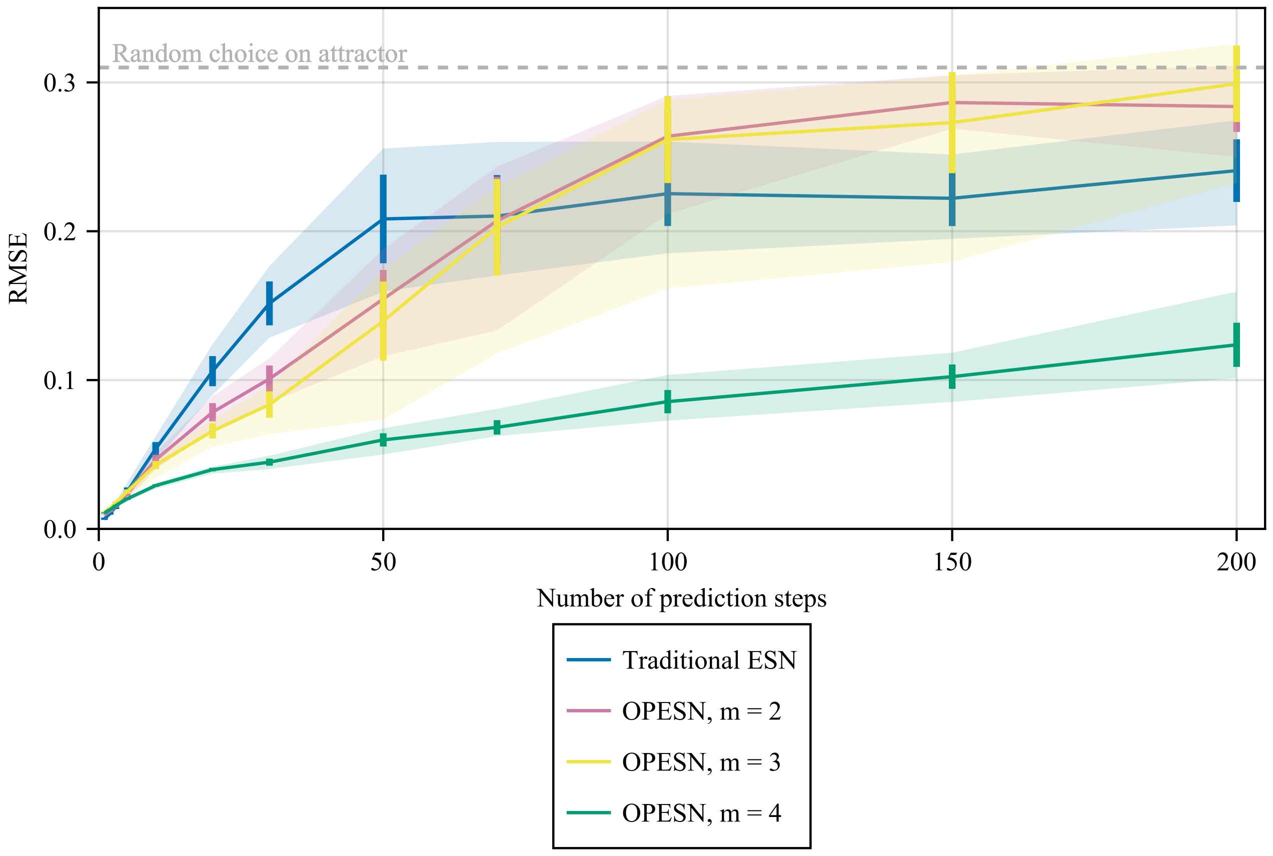


Figure 6.8: Mean iterative prediction RMSE for the OPESN on the Mackey-Glass time series ($\Delta t = 0.5$, $\alpha = 0.1$, $k \approx 500$). Delay $\tau = 20$ for all $m = 2, 3, 4$. Vertical bars show standard deviations; shaded regions show full range.

longer prediction horizons.

The direct prediction results for the OPESN on the Mackey-Glass system (shown in Figure 6.9) show a similar pattern as observed for the ORSESN: increasing the ordinal pattern dimension m consistently leads to lower mean RMSE across tested prediction lengths greater than 2 steps, becoming significant (95% CI) for 5 steps or longer. While a higher m enables the OPESN to achieve improved direct predictions and mid-term iterative predictions, the $m = 2, 3$ models appear to suffer from instability and error accumulation in recursive prediction. By contrast, the $m = 4$ OPESN appears to maintain its advantage, demonstrating stable and accurate predictions at longer horizons. Although the OPESN does not provide a substantial benefit for the Lorenz and Rossler time series, these results indicate that the OPESN can offer benefit for some attractors.

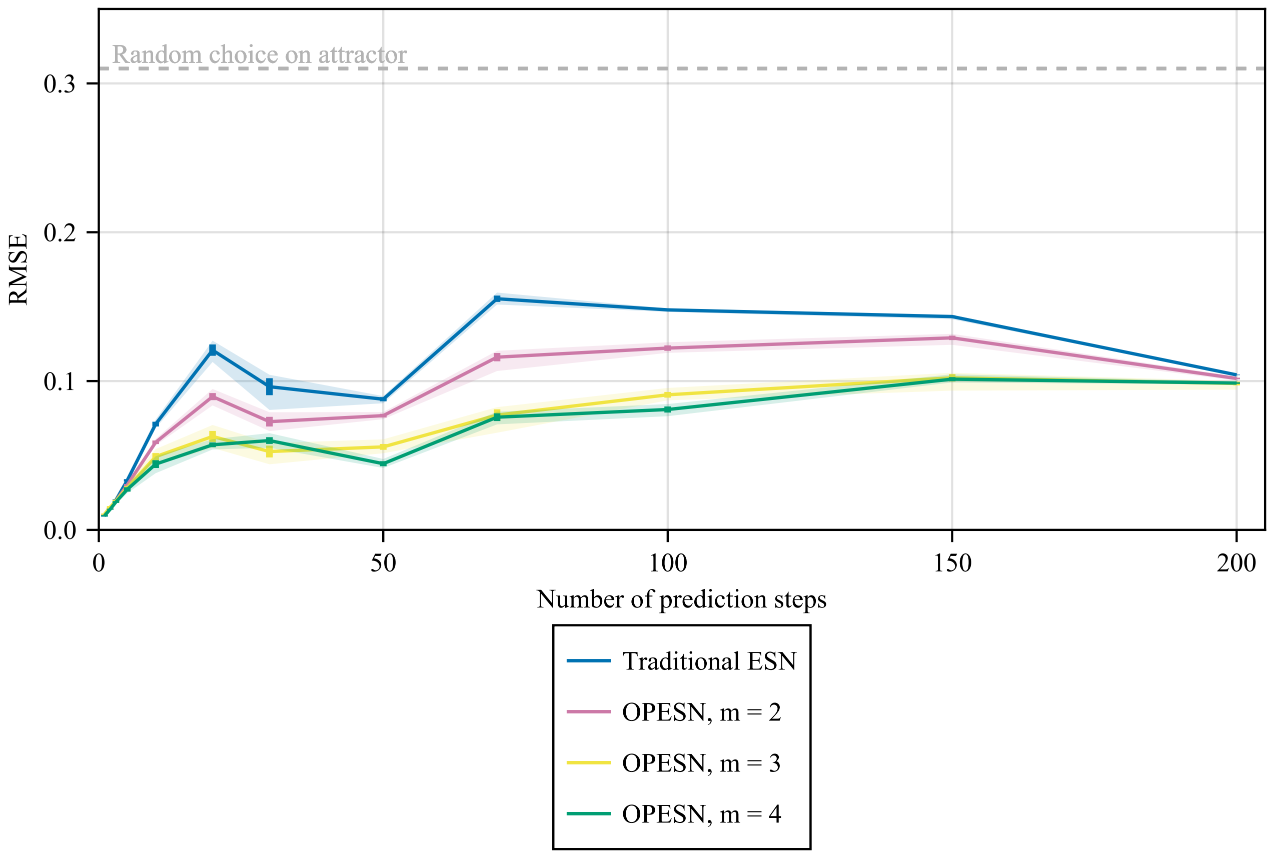


Figure 6.9: Mean direct prediction RMSE for the OPESN on the Mackey-Glass time series ($\Delta t = 0.5$, $\alpha = 0.1$, $k \approx 500$). Delay $\tau = 20$ for all $m = 2, 3, 4$. Vertical bars show standard deviations; shaded regions show full range.

Input Routing

The OPESN introduces two architectural features to the traditional ESN, the input routing mechanism and sub-reservoirs, and drives the input routing using ordinal partition information. Here we will test whether the improved performance - and indeed lack of improved performance - is caused by the new architectural features or the introduction of ordinal partition information. Similar to testing the gating funtion of the ORSESN, we supplied the OPESN with a random selection of sub-reservoir and compared it to the input routing driven by the ordinal partition. Figure 6.10 shows the results for the OPESN with each of $m = 2, 3, 4$. The OPESN with random routing fails to significantly improve upon the traditional ESN at any prediction length, and in fact significantly degrades the performance for some prediction lengths - especially for the OPESN with $m = 4$. We have concluded that the performance of the OPESN is likely hindered by its idiosyncratic architectural features. We can infer from this testing and the results from Chapter 5 that the introduction of ordinal partition information has the potential to improve model performance, however the architecture of the OPESN does not adequately exploit this potential in most cases.

Sub-Reservoir Connections

As described early in this chapter, we trialed multiple methods of connecting the sub-reservoirs of the OPESN. We trialed each of these methods by finding their mean prediction RMSE across 30 trials on the Lorenz ($\Delta t = 0.01$) series. We found that no method of connection between sub-reservoirs significantly (95% CI) improved upon the results of the disconnected sub-reservoirs. Figure 6.11 shows these mean prediction RMSEs for the OPESN with $m = 4$, with the dashed black line representing the 'disconnected sub-reservoirs' OPESN variant. As the figure illustrates, no other variant produces a significantly lower mean RMSE than the disconnected sub-reservoirs. Appendix A contains similar results for recursive and direct predictions and for OPESN models with $m = 3, 4$.

These results lead us to speculate that the modest advantages confered by the OPESN do not result from its modeling of an ordinal transition network, but rather from the specialisation of the sub-reservoirs akin to a mixture of experts model.

Resilience to Noise

As described in Chapter 5, we also tested the sensitivity of the OPESN to noise. Figure 6.12 shows the mean direct prediction RMSE of multiple trials across values of noise $\alpha \in [0.0, 1.0]$, with four curves representing different numbers of prediction steps. As to be expected, as the level of noise increases so does the prediction RMSE for both the traditional ESN ($m = 1$) and the ORSESN models ($m = 2, 3, 4$). The noise resilience of the ORSESN models is not significantly different to the traditional ESN for direct prediction. The same noise analysis performed with iterative prediction (Figure 6.13) shows the OPESN to be mildly more resilient to noise, and shows that the OPESN with $m = 4$ appears to benefit greatly from the regularisation provided by additive noise. For the purposes of testing the performance of the traditional ESN and the OPESN we have chosen the 'sweet spot' value of $\alpha = 0.1$, the same chosen for the ORSESN.

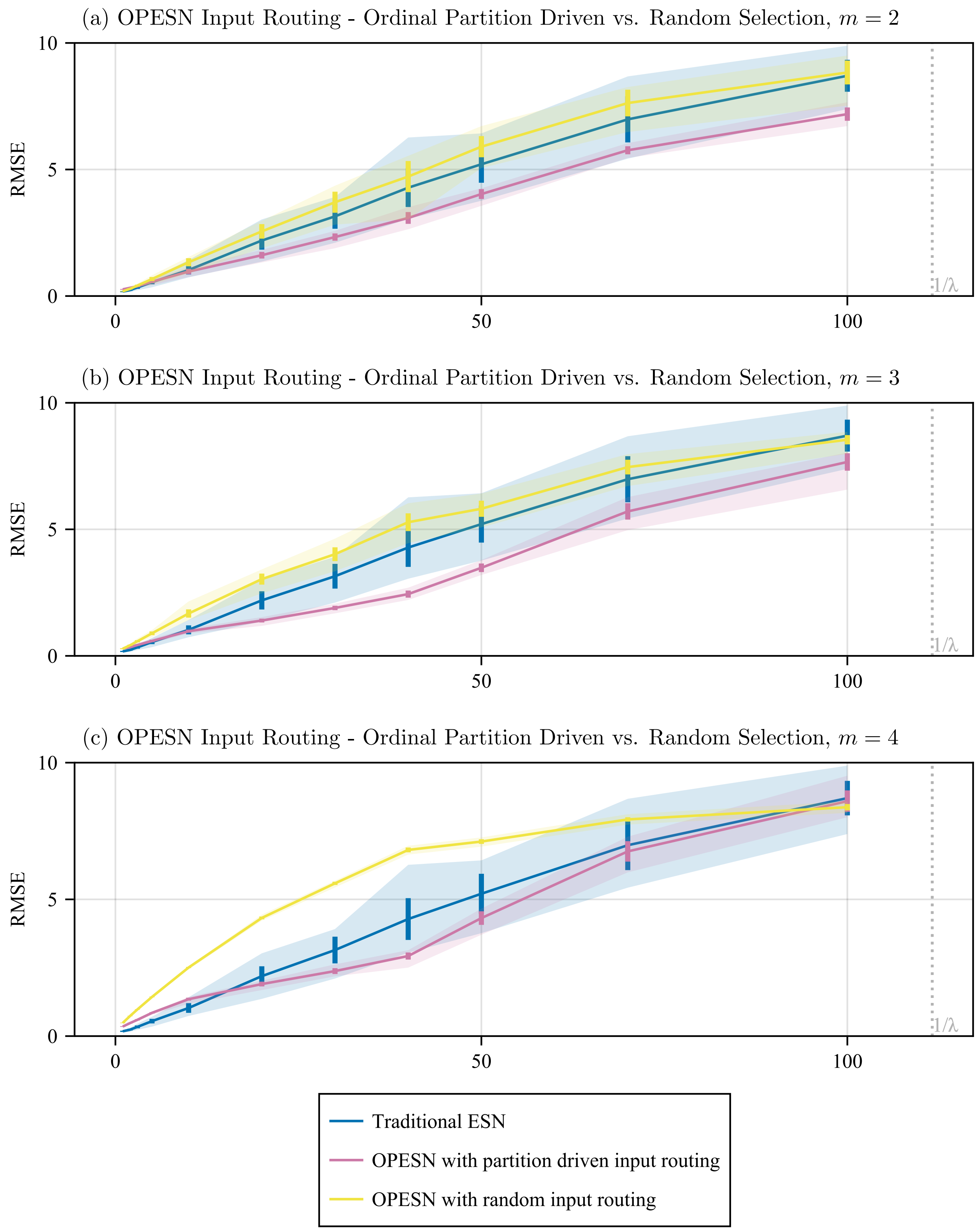


Figure 6.10: Mean iterative prediction RMSE on the Lorenz x component ($\Delta t = 0.01$, $\alpha = 0.1$, $k = 468$) comparing OPESN input routing by ordinal partition versus random routing, for delays $\tau = 20$ and $m = 2, 3, 4$ in subfigures (a)-(c). Vertical bars show standard deviations; shaded regions show full range.

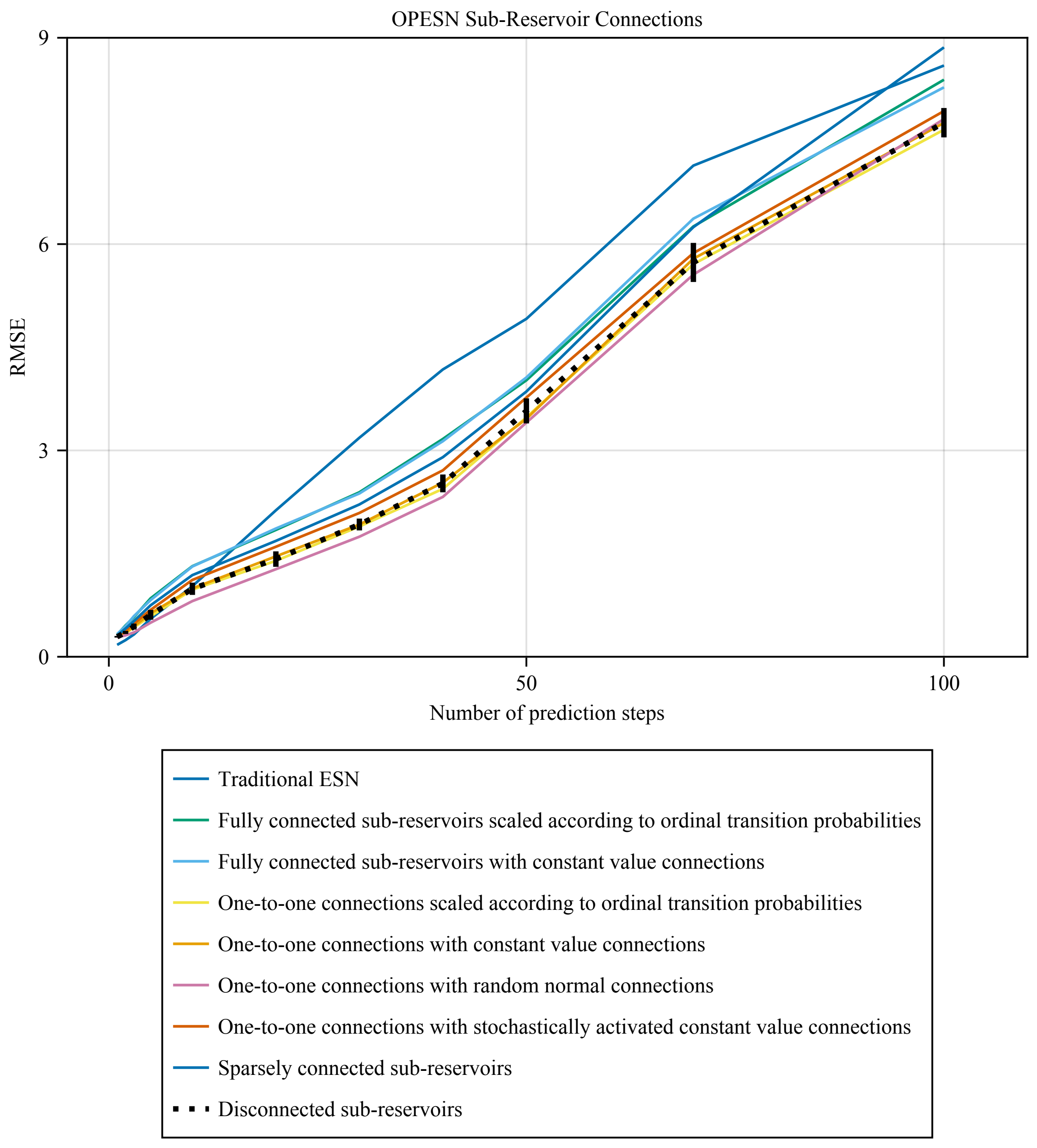


Figure 6.11: Mean iterative prediction RMSE on the Lorenz x component ($\Delta t = 0.01$, $\alpha = 0.1$, $k = 468$) for OPESN variants of sub-reservoir connectivity. Delay $\tau = 20$, $m = 4$. Dashed black line is the disconnected baseline; vertical bars show its standard deviation.

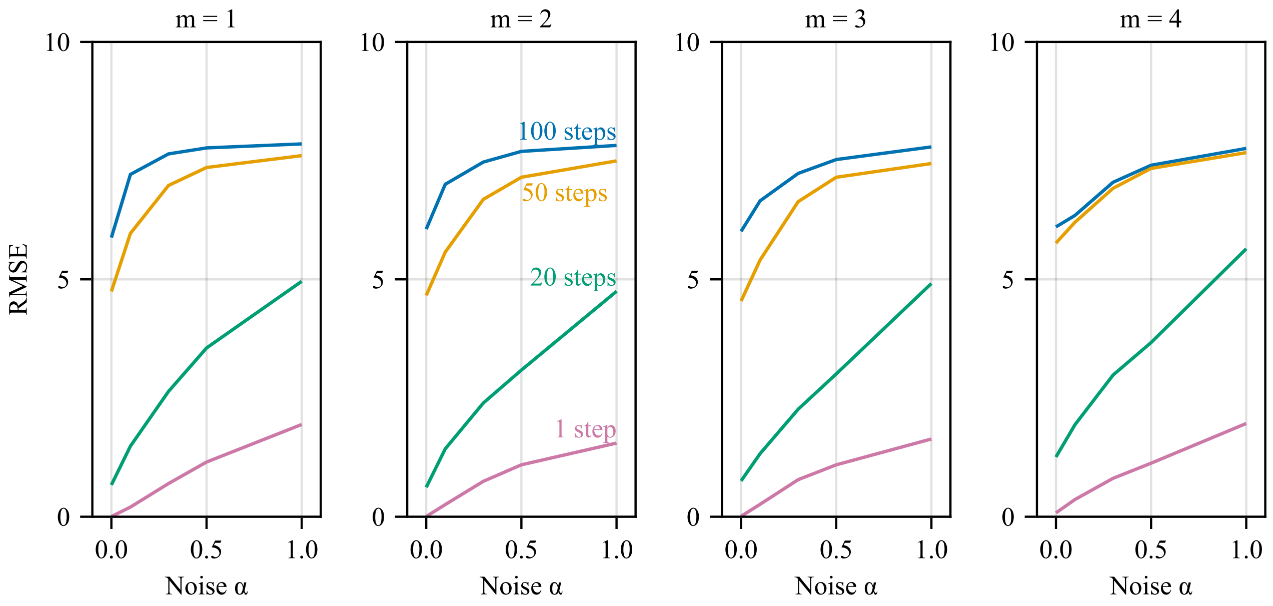


Figure 6.12: Mean direct prediction RMSE as a function of noise level $\alpha \in [0, 1]$ for the OPESN on the Lorenz x component ($\Delta t = 0.01$, delay $\tau = 20$, $k = 468$). Curves are for lengths of prediction 1, 20, 50 and 100 steps; panels correspond to $m =$ 1-4.

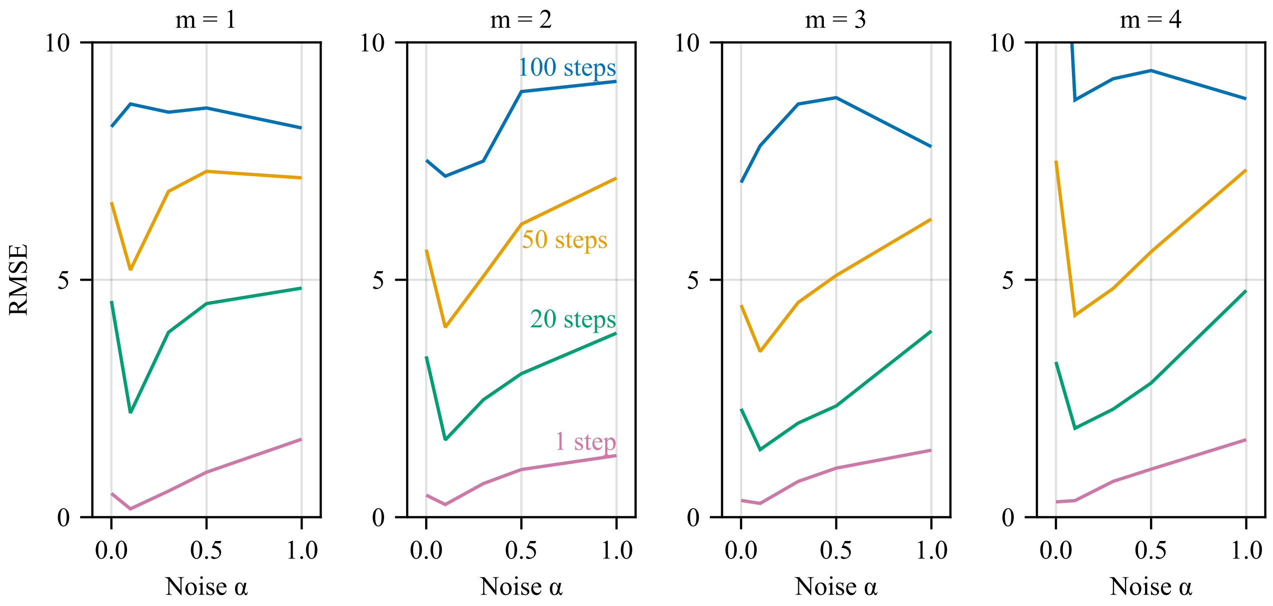


Figure 6.13: Iterative prediction RMSE as a function of noise level $\alpha \in [0, 1]$ for the OPESN on the Lorenz x component ($\Delta t = 0.01$, delay $\tau = 20$, $k = 468$). Curves are for lengths of prediction 1, 20, 50 and 100 steps; panels correspond to $m =$ 1-4.

Chapter 7

Conclusion

This thesis investigated the potential of integrating ordinal partition analysis into ESN architectures to enhance their predictive capabilities for chaotic time series. The core hypothesis was that incorporating ordinal patterns—either to guide readout selection or to structure the reservoir and input routing—could lead to ESN models that more accurately capture and predict complex dynamics. To this end, two novel architectures were proposed and evaluated: the Ordinal Partition-based Readout Switching ESN (ORSESN) and the Ordinal Partition ESN (OPESN).

7.1 Key findings

The ORSESN architecture, which employed ordinal partitions of the input to switch between different readout vectors, demonstrated notable improvements over traditional ESNs, particularly for medium to long-term prediction horizons across various chaotic systems (Lorenz, Rössler, and Mackey-Glass). For iterative prediction, while traditional ESNs often performed better for single-step predictions, the ORSESN, especially with higher ordinal pattern dimensions ($m = 3$ or $m = 4$), consistently achieved lower Root Mean Squared Error (RMSE) and greater prediction stability (lower variance across trials) as the number of prediction steps increased. This was particularly evident in the Lorenz and Rössler systems. For the Mackey-Glass system, ORSESN with m=4 showed substantially superior performance across almost all iterative prediction horizons. Direct prediction results largely mirrored these findings, with ORSESN models (particularly with $m = 4$) outperforming traditional ESNs for longer forecasts. This suggests the ORSESN's advantage is not merely an artefact of error accumulation in iterative predictions but reflects an enhanced ability to model the underlying system dynamics. We are led to believe that the predictive advantage of the ORSESN was indeed conferred by the ordinal partition information guiding the readout selection, rather than the mere presence of a multi-readout gating mechanism. Furthermore, while ORSESN models appeared slightly less resilient to increasing noise levels in terms of relative RMSE increase, their absolute mean RMSE often remained lower than or comparable to traditional ESNs across tested noise levels ($\alpha \in [0.0, 1.0]$).

The OPESN architecture, which used ordinal partitions to route input to distinct sub-reservoirs and explored various inter-sub-reservoir connection strategies, yielded more mixed results. For the Lorenz system, the OPESN showed only marginal and inconsistent improvements over traditional ESNs on the Lorenz system, with benefits diminishing with longer time steps. On the Rössler system, OPESN with $m = 3$ offered some improvement for longer prediction horizons in both iterative and direct prediction; however, increasing m to 4 degraded prediction performance. For the Mackey-Glass system, the OPESN (particularly with m=4) showed promise, achieving substantially lower RMSEs for longer iterative predictions and consistently better direct predictions with increasing m, similar to the ORSESN. Experiments with different sub-reservoir connection strategies (one-to-one, fully connected, scaled by transition probabilities, constant values, or disconnected) within the OPESN did not reveal a significantly superior connection method over simply using disconnected sub-reservoirs, suggesting that inter-sub-reservoir connections, as implemented, did not substantially enhance performance. The input routing mechanism, when driven by random selection instead of ordinal partitions, generally hindered OPESN performance, indicating that while ordinal information is valuable, the specific architectural choices of the OPESN does not optimally exploit this information, or that the architecture itself introduces complexities that can degrade performance.

The particularly strong performance of both ORSESN and OPESN (especially with $m = 4$) on the Mackey-Glass series, when compared to the traditional ESN, may be attributed to the inherent nature of the Mackey-Glass system as a delay-differential equation. The Mackey-Glass equation is a scalar delay differential system whose derivative at time t depends directly on $x(t - \tau)$. Ordinal patterns constructed with embedding delay τ encode whether recent samples are rising, falling, or stationary, thereby summarising the lagged information most relevant to the dynamics. When m and τ are chosen appropriately, ORSESN can switch readouts and OPESN can route activations based on these patterns, effectively specialising on distinct local contexts. A conventional ESN can still capture delay effects through its evolving state, but it does so without an explicit symbolic representation, which may explain the stronger results observed for the variants based on ordinal partitions.

7.2 Contributions and implications

This research makes several contributions to the field of reservoir computing and time series prediction. Firstly, we designed and implemented two novel ESN architectures, ORSESN and OPESN, that explicitly integrate ordinal partition information. Secondly, we empirically validated these architectures against traditional ESNs on well-known chaotic attractors, demonstrating that ordinal-feature-guided ESNs, particularly the ORSESN, can offer significant performance improvements, especially for longer-term predictions. Thirdly, the findings highlight the utility of ordinal patterns as an interpretable, data-derived feature for enhancing ESNs; the ORSESN's success suggests that gating the readout mechanism based on local temporal orderings is a promising strategy. Lastly, the study provides insights into the impact of design parameters, such as ordinal pattern dimension (m) and delay (τ), and architectural choices on model performance.

7.3 Limitations and further research

- The current OPESN architecture assigns a sub-reservoir to each observed partition, which for higher m can lead to a very large number of sub-reservoirs, many potentially associated with rare patterns and trained on limited data. Strategies for handling rare patterns, grouping similar ordinal patterns or growing the network to accomodate unseen ordinal patterns could be beneficial.
- The ORSESN gates the predictions generated by each readout by selecting only the relevant ordinal partition, however a weighted combination based on ordinal transition probabilities could be investigated.
- The thesis explored specific ways of integrating ordinal patterns, but other integration methods - such as using ordinal transition probabilities to modulate reservoir dynamics or directly feeding ordinal symbols as auxiliary inputs to a standard ESN - could be investigated.
- The investigation of OPESN sub-reservoir connections was not exhaustive; more sophisticated connection strategies, potentially learned or dynamically adapted, could be explored.

- The study primarily focused on single-component time series. Extending these architectures to multi-dimensional time series, perhaps by using multi-dimensional ordinal patterns or routing different dimensions to specialized sub-reservoirs in an OPESN-like structure, would be a valuable next step.
- Further theoretical analysis into why and how ordinal-guided architectures improve prediction, particularly the stability observed in ORSESN with $m = 4$, would deepen understanding.
- Testing these architectures on a broader range of real-world datasets from different domains would further validate their generalizability and practical utility.

In conclusion, this thesis has demonstrated that ordinal partitions can serve as an effective predictive feature for enhancing echo state networks. The ORSESN stands out as a particularly effective architecture, offering improved accuracy and stability for chaotic time series forecasting. While the OPESN showed more modest results, the underlying principle of using ordinal information to structure reservoir computation remains a promising area for continued exploration. Future work building upon these foundations has the potential to further advance the capabilities of reservoir computing for complex time series analysis.

Appendix A

OPESN sub-reservoir connections

OPESN with stochastic connections

Here I will explain the implementation of the OPESN with one-to-one sub-reservoir connections that are stochastically activated constant value connections. At every time step each partition chooses exactly one destination partition according to the ordinal transition probabilities. Firstly we will define a stochastic function $F_i(t)$ which selects a destination partition j for each partition i at each time step (t) such that

$$P[F_i(t) = j] = p_{i,j}$$

where $F_i(t) \in [1, ..., N_{obs}]$. Then we can define each submatrix $\mathbf{W}_{i,j}$ in the recurrence matrix $\mathbf{W}_{rec}$ as we did for other connection methods in Chapter 6:

$$\mathbf{W}_{i,j}(t) = \begin{cases} \mathbf{W}_{ER_{i,j}}, & i = j, \\ c\mathbf{I}, & i \neq j \text{ and } F_i(t) = j, \\ \mathbf{0}, & \text{otherwise.} \end{cases}$$

Practically, this involves instantiating the constant one-to-one connections solution

$$\mathbf{W}_{i,j} = \begin{cases} \mathbf{W}_{ER_{i,j}}, & i = j, \\ c\mathbf{I}, & i \neq j \text{ and } p_{i,j} > 0, \\ \mathbf{0}, & \text{otherwise,} \end{cases}$$

and masking each $\mathbf{W}_{i,j}$ at each time step if $F_i(t) \neq j$ by setting $\mathbf{W}_{i,j} = \mathbf{0}$. This masking process makes the evolution of the reservoir extremely inefficient, so it is necessary to cache the possible masked states of the resevoir for each combination of stochastically selected partitions. The stochastic OPESN with $m = 4$ has $(4!)(4!) = 576$ possible combinations of partitions, which creates a memory usage problem for storing the cached masked states of the reservoir. Thus we cached the most common combinations and allowed the remaining to be computed at each time step, including a partition transition pair in the most common combinations according to a transition probability threshold.

Further sub-reservoir connection experiments

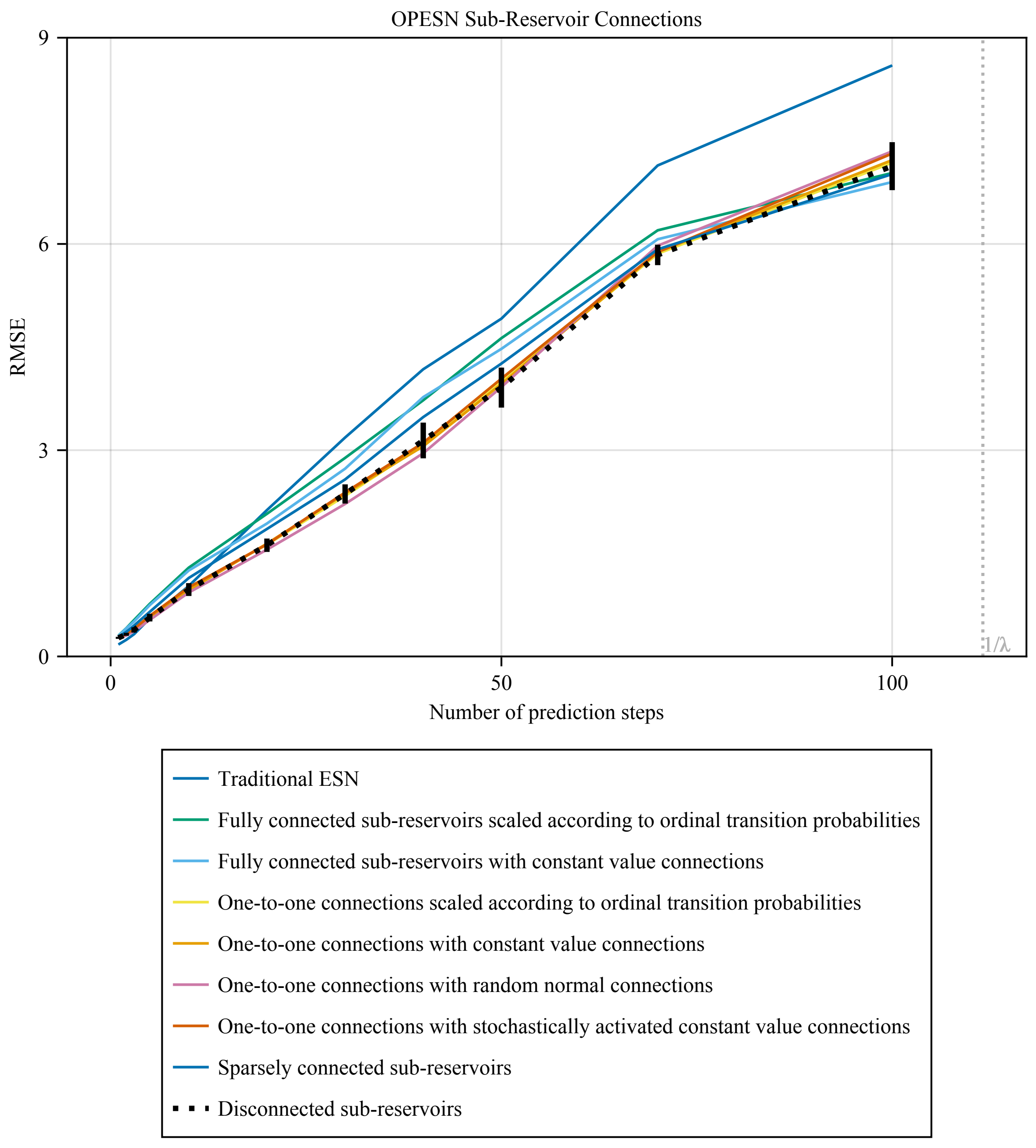


Figure A.1: Recursive prediction RMSE for the OPESN for all variants of sub-reservoir connections. Delay $\tau = 20$, $m = 3$, noise level $\alpha = 0.1$, and total reservoir size $k = 468$.

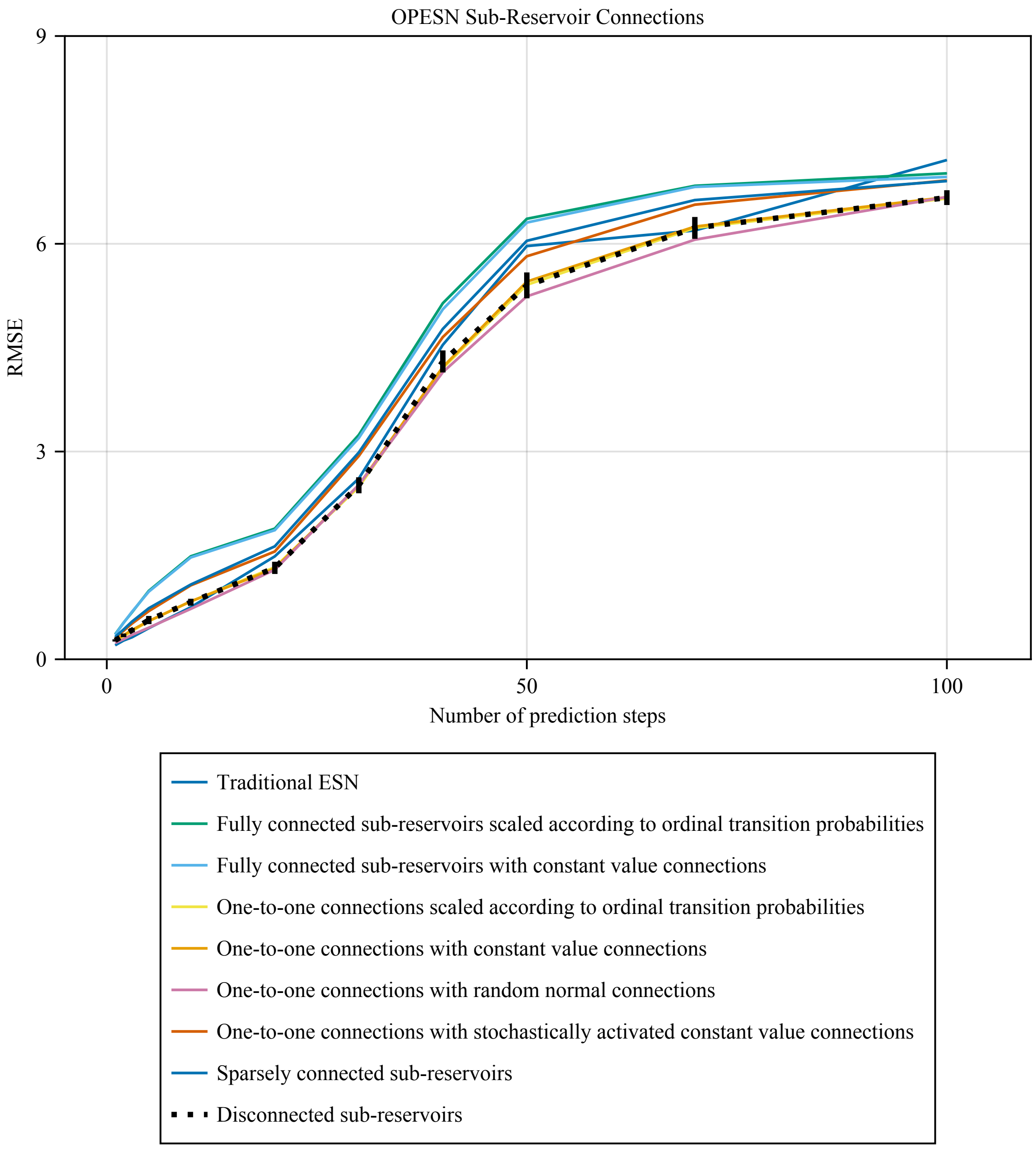


Figure A.2: Direct prediction RMSE for the OPESN for all variants of sub-reservoir connections. Delay $\tau = 20$, $m = 4$, noise level $\alpha = 0.1$, and total reservoir size $k = 468$.

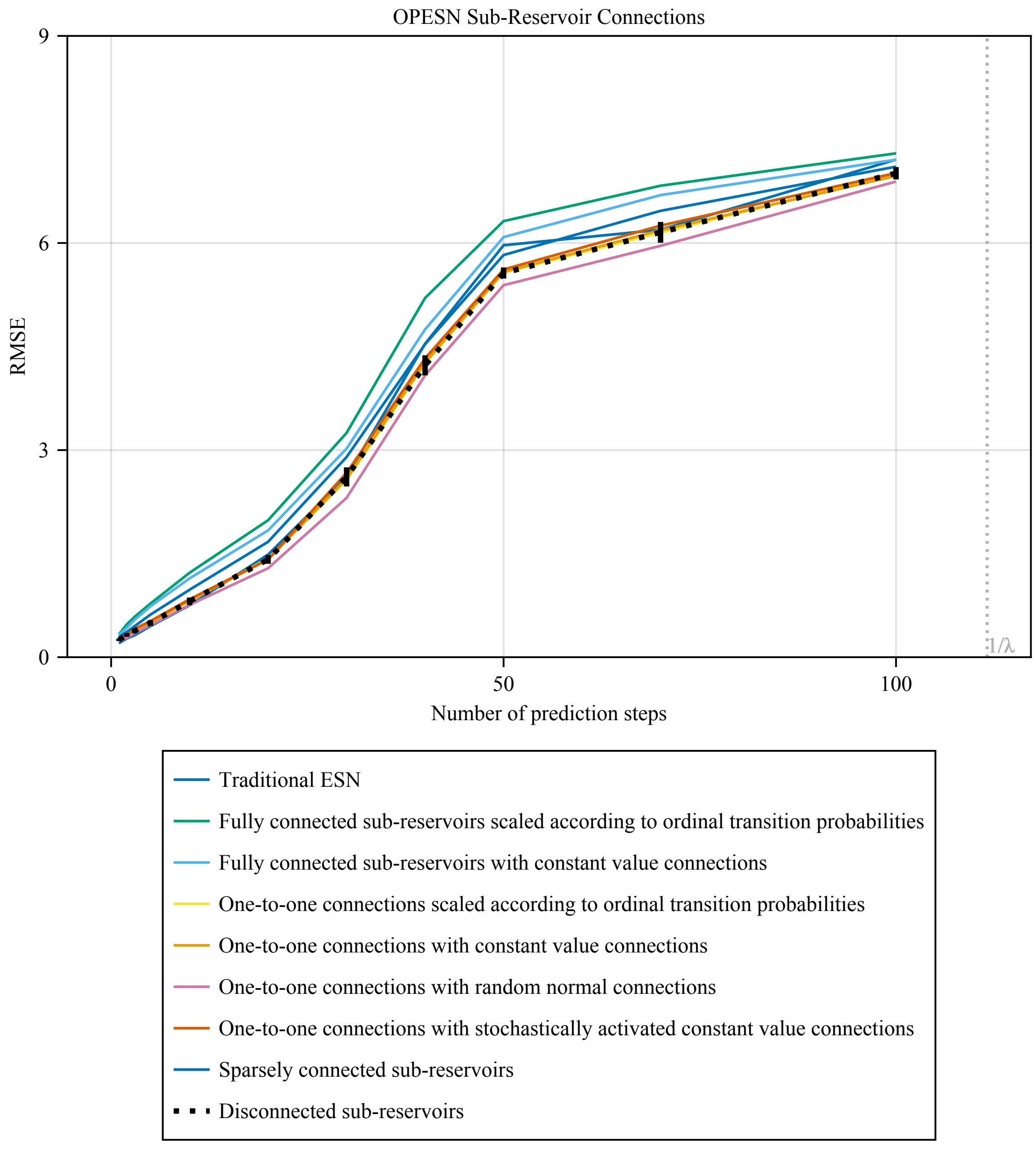


Figure A.3: Direct prediction RMSE for the OPESN for all variants of sub-reservoir connections. Delay $\tau = 20$, $m = 3$, noise level $\alpha = 0.1$, and total reservoir size $k = 468$.

Bibliography

[1] J. M. Amigó, S. Zambrano, and M. A. F. Sanjuán. True and false forbidden patterns in deterministic and random dynamics. *Europhysics Letters*, 79(5):50001, jul 2007.

[2] L. Argentieri, C. Gallicchio, A. Micheli, et al. Input routed echo state networks. In *ESANN 2022 Proceedings - 30th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 405–410, 2022.

[3] Š. Babinec and J. Pospíchal. Gating echo state neural networks for time series forecasting. In M. Köppen, N. Kasabov, and G. Coghill, editors, *Advances in Neuro-Information Processing*, pages 200–207, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[4] C. Bandt and B. Pompe. Permutation entropy: A natural complexity measure for time series. *Phys. Rev. Lett.*, 88:174102, Apr 2002.

[5] B. Boaretto and C. Masoller. Discriminating chaotic and stochastic time series using permutation entropy and artificial neural networks. *Scientific reports*, 11(1):15789, 2021.

[6] M. Buehner and P. Young. A tighter bound for the echo state property. *IEEE Transactions on Neural Networks*, 17(3):820–824, 2006.

[7] Z. Carmichael, H. Syed, S. Burtner, and D. Kudithipudi. Mod-deepesn: modular deep echo state network. *CoRR*, abs/1808.00523, 2018.

[8] M. Dale. Neuroevolution of hierarchical reservoir computers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18, pages 410–417, New York, NY, USA, 2018. Association for Computing Machinery.

[9] M. Dale, J. F. Miller, and S. Stepney. *Reservoir computing as a model for in-materio computing*, pages 533–571. Springer International Publishing, Cham, 2017.

[10] Z. Deng and Y. Zhang. Collective behavior of a small-world recurrent neural system with scale-free distribution. *IEEE Transactions on Neural Networks*, 18(5):1364–1375, 2007.

[11] C. Fernando and S. Sojakka. Pattern recognition in a bucket. In W. Banzhaf, J. Ziegler, T. Christaller, P. Dittrich, and J. T. Kim, editors, *Advances in Artificial Life*, pages 588–597, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[12] C. Gallicchio, A. Micheli, and L. Pedrelli. Deep reservoir computing: A critical experimental analysis. *Neurocomputing*, 268:87–99, 2017. Advances in artificial neural networks, machine learning and computational intelligence.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[14] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[15] H. Jaeger. The “echo state” approach to analysing and training recurrent neural networks. GMD Report 148, German National Research Center for Information Technology, Sankt Augustin, Germany, 2001. With an erratum note.

[16] H. Jaeger. Discovering multiscale dynamical features with hierarchical echo state networks. Technical Report 10, Jacobs University Bremen, 2007.

[17] S. Jarvis, S. Rotter, and U. Egert. Extending stability through hierarchical clusters in echo state networks. *Frontiers in Neuroinformatics*, Volume 4 - 2010, 2010.

[18] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2):181–214, 1994.

[19] H. Kantz and T. Schreiber. *Nonlinear time series analysis*. Cambridge University Press, USA, 2003.

[20] K. Keller, H. Lauffer, and M. Sinn. *Ordinal analysis of EEG time series*, pages 239–249. Nova Science Publishers, Inc., Dec. 2009.

[21] A. Laan and R. Vicente. Echo state networks with multiple readout modules. *bioRxiv*, 2017.

[22] I. Leyva, J. H. Martínez, C. Masoller, O. A. Rosso, and M. Zanin. 20 years of ordinal patterns: perspectives and challenges. *Europhysics Letters*, 138(3):31001, may 2022.

[23] Y. Li and F. Li. Growing deep echo state network with supervised learning for time series prediction. *Applied Soft Computing*, 128:109454, 2022.

[24] Z. Li and G. Tanaka. Multi-reservoir echo state networks with sequence resampling for nonlinear time-series prediction. *Neurocomputing*, 467:115–129, 2022.

[25] Y. Liu, Y. Lin, Z. Jia, Y. Ma, and J. Wang. Representation based on ordinal patterns for seizure detection in eeg signals. *Computers in Biology and Medicine*, 126:104033, 2020.

[26] J. Long, S. Zhang, and C. Li. Evolving deep echo state networks for intelligent fault diagnosis. *IEEE Transactions on Industrial Informatics*, 16(7):4928–4937, 2020.

[27] E. N. Lorenz. Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2):130 – 141, 1963.

[28] M. Lukoševičius. *A practical guide to applying echo state networks*, pages 659–686. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

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

[30] S. Lun, Z. Sun, M. Li, and L. Wang. Multiple-reservoir hierarchical echo state network. *Mathematics*, 11(18), 2023.

[31] Q. Ma, L. Shen, and G. W. Cottrell. Deep-esn: a multiple projection-encoding hierarchical reservoir computing framework. *CoRR*, abs/1711.05255, 2017.

[32] Q. Ma, L. Shen, and G. W. Cottrell. Deepr-esn: A deep projection-encoding echo-state network. *Information Sciences*, 511:152–171, 2020.

[33] Q.-L. Ma and W.-B. Chen. Modular state space of echo state network. *Neurocomputing*, 122:406–417, 2013. Advances in cognitive and ubiquitous computing.

[34] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197(4300):287–289, 1977.

[35] Z. K. Malik, A. Hussain, and Q. J. Wu. Multilayered echo state machine: A novel architecture and algorithm. *IEEE Transactions on Cybernetics*, 47(4):946–959, 2017.

[36] S. Masoudnia and R. Ebrahimpour. Mixture of experts: a literature survey. *Artificial Intelligence Review*, 42:275–293, 2014.

[37] J. Qiao, F. Li, H. Han, and W. Li. Growing echo-state network with multiple subreservoirs. *IEEE Transactions on Neural Networks and Learning Systems*, 28(2):391–404, 2017.

[38] M. Riedl, A. Müller, and N. Wessel. Practical considerations of permutation entropy: A tutorial review. *The European Physical Journal Special Topics*, 222(2):249–262, 2013.

[39] O. Rössler. An equation for continuous chaos. *Physics Letters A*, 57(5):397–398, 1976.

[40] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. V. Le, G. E. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *CoRR*, abs/1701.06538, 2017.

[41] M. Sinn, A. Ghodsi, and K. Keller. Detecting change-points in time series by maximum mean discrepancy of ordinal pattern distributions. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, UAI'12, pages 786–794, Arlington, Virginia, USA, 2012. AUAI Press.

[42] X. Sun, M. Hao, Y. Wang, Y. Wang, Z. Li, and Y. Li. Reservoir dynamic interpretability for time series prediction: A permutation entropy view. *Entropy*, 24(12), 2022.

[43] M. Tabuse, M. Kinouchi, and M. Hagiwara. Recurrent neural network using mixture of experts for time series processing. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 1, pages 536–541 vol.1, 1997.

[44] B. Thorne, T. Jüngling, M. Small, D. Corrêa, and A. Zaitouny. A novel approach to time series complexity via reservoir computing. In H. Aziz, D. Corrêa, and T. French, editors, *AI 2022: Advances in Artificial Intelligence*, pages 442–455, Cham, 2022. Springer International Publishing.

[45] R. Wcisło and W. Czech. Grouped multi-layer echo state networks with self-normalizing activations. In M. Paszynski, D. Kranzlmüller, V. V. Krzhizhanovskaya, J. J. Dongarra, and P. M. A. Sloot, editors, *Computational Science – ICCS 2021*, pages 90–97, Cham, 2021. Springer International Publishing.

[46] C. Wringe, S. Stepney, and M. A. Trefzer. Restricted reservoirs on heterogeneous timescales. In M. Wand, K. Malinovská, J. Schmidhuber, and I. V. Tetko, editors, *Artificial Neural Networks and Machine Learning – ICANN 2024*, pages 168–183, Cham, 2024. Springer Nature Switzerland.

[47] S. E. Yuksel, J. N. Wilson, and P. D. Gader. Twenty years of mixture of experts. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1177–1193, 2012.

[48] M. Zanin. Forbidden patterns in financial time series. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 18(1):013119, 03 2008.

[49] M. Zanin. Continuous ordinal patterns: creating a bridge between ordinal analysis and deep learning. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 33(3):033114, 03 2023.