

Structuring Echo State Networks using Ordinal Partitions

Hector Morlet

22247737

Supervised by Eugene Tan, Michael Small

March 14, 2025

Introduction

- ▶ Echo State Networks
- ▶ Ordinal Partitions
- ▶ 2 Main Research Proposals

Echo State Networks

- ▶ First introduced by Jaeger (2001).
- ▶ Type of recurrent neural network.
- ▶ Commonly studied form of reservoir computing.
- ▶ A form of supervised learning.
- ▶ Most weights are fixed and only some are fitted.
- ▶ Is computationally efficient compared to other neural networks.

Echo State Networks

State update equation:

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

- ▶ \mathbf{W}_{in} , \mathbf{W}_{rec} , \mathbf{W}_{bias} and $\mathbf{s}(t)$ are randomly generated according to hyper-parameters.
- ▶ Echo state property
- ▶ \mathbf{W}_{in} , \mathbf{W}_{rec} and \mathbf{W}_{bias} are fixed.
- ▶ Only the readout vector weights is fitted.

Echo State Networks

Output equation:

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

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

Output regression with regularisation:

$$\mathbf{C}_{out} = (\mathbf{S}^T \mathbf{S} + \beta \mathbf{I})^{-1} \mathbf{S}^T \mathbf{Y}$$

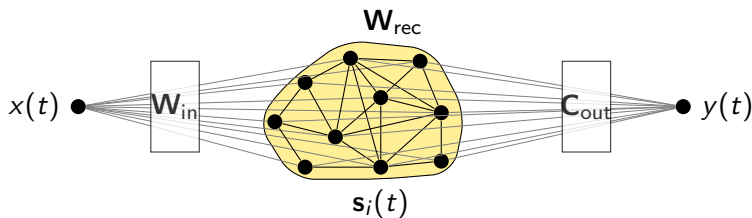


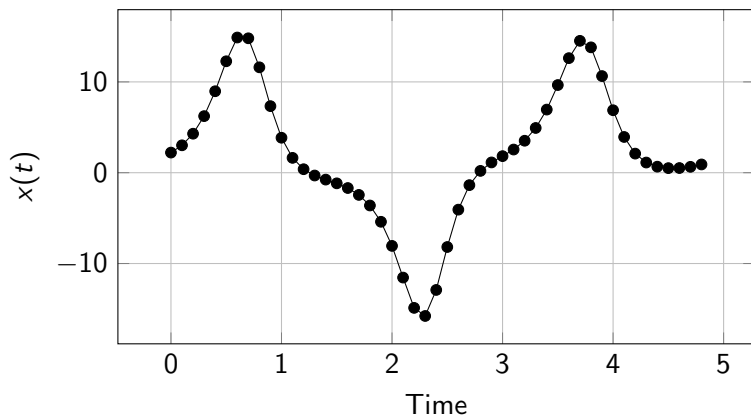
Figure: A diagram of an Echo State Network.

Ordinal Partitions

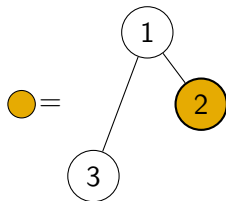
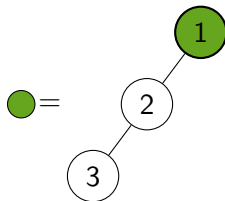
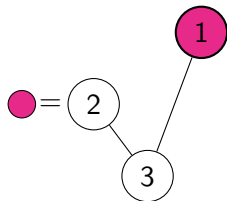
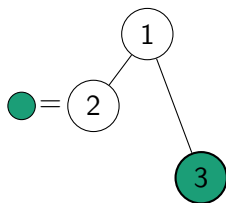
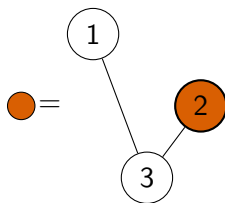
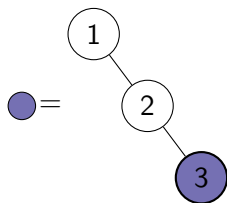
- ▶ Bandt and Pompe (2002).
- ▶ Each data point in a timeseries is partitioned by the ordering of its preceding points.
- ▶ Each partition has a unique 'ordinal symbol' (e.g. numbers from 1 to 6).
- ▶ The probability of the timeseries transitioning from one partition to another can be calculated, giving the 'ordinal transition probabilities'.

Ordinal Partitions

Lorenz Attractor: x -component

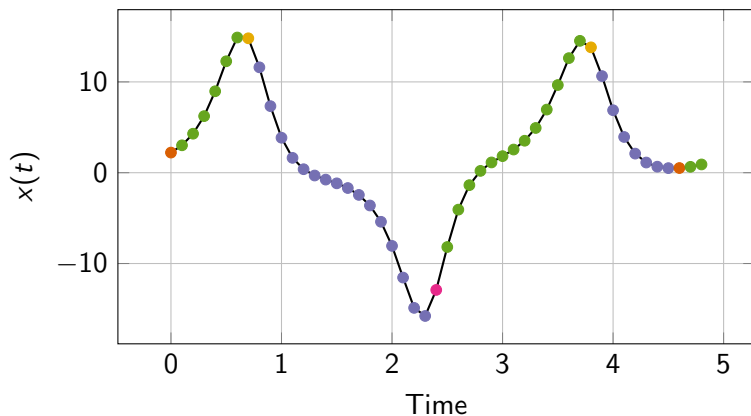


Ordinal Partitions



Ordinal Partitions

Lorenz Attractor: x -component



Ordinal Partitions













						
	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: Transition probabilities between ordinal partitions.

Research Questions

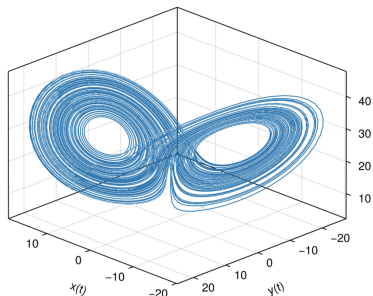
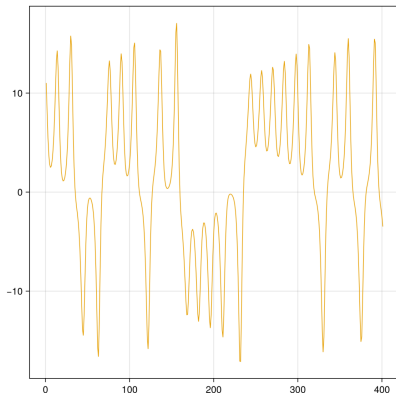
- ▶ Can ordinal partitions be used to improve forecasting accuracy?
- ▶ 'Feature selection'

Input data: Lorenz

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

$$\frac{dz}{dt} = xy - \beta z$$



Research Proposal: Approach 1

Switch the readout (C_{out}) vector based on the ordinal partition.

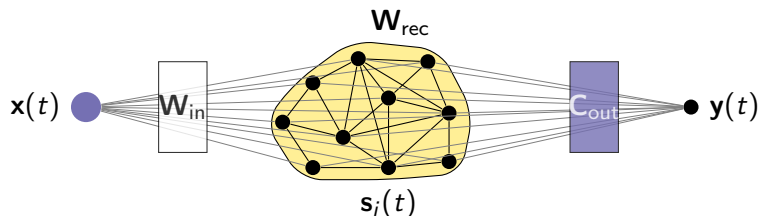


Figure: Echo State Network with readout switching.

Research Proposal: Approach 1

Switch the readout (C_{out}) vector based on the ordinal partition.

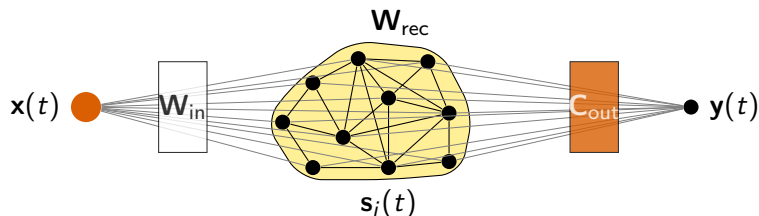


Figure: Echo State Network with readout switching.

Results: Approach 1

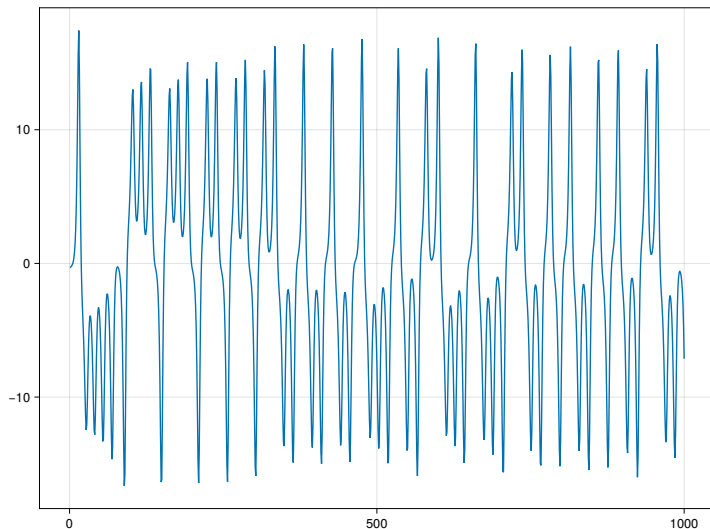


Figure: Freerun prediction for Approach 1 (Readout Switching)

Results: Approach 1

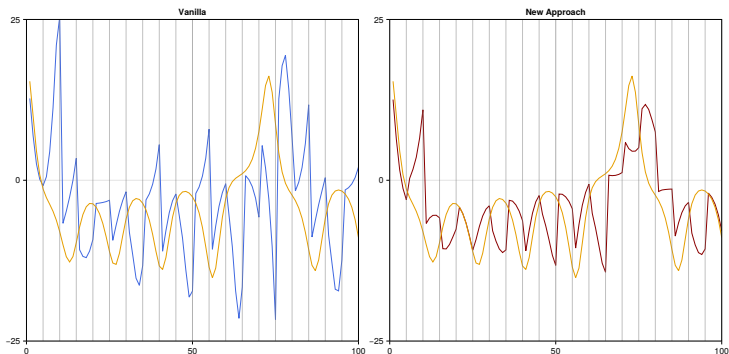


Figure: Multistep prediction for Approach 1 (Readout Switching)

Results: Approach 1

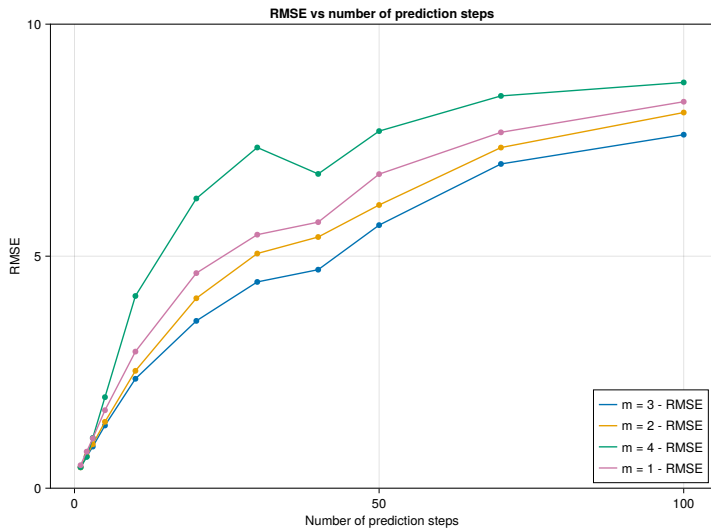


Figure: Root Mean Square Error (RMSE) vs. Steps for Readout Switching

Research Proposal: Approach 2

- ▶ Restructure the reservoir into 'subreservoirs' for each partition.
- ▶ Feed the input based on the ordinal partition.
- ▶ Weight the connections between subreservoirs according to the ordinal transition probabilities.

Research Proposal: Approach 2

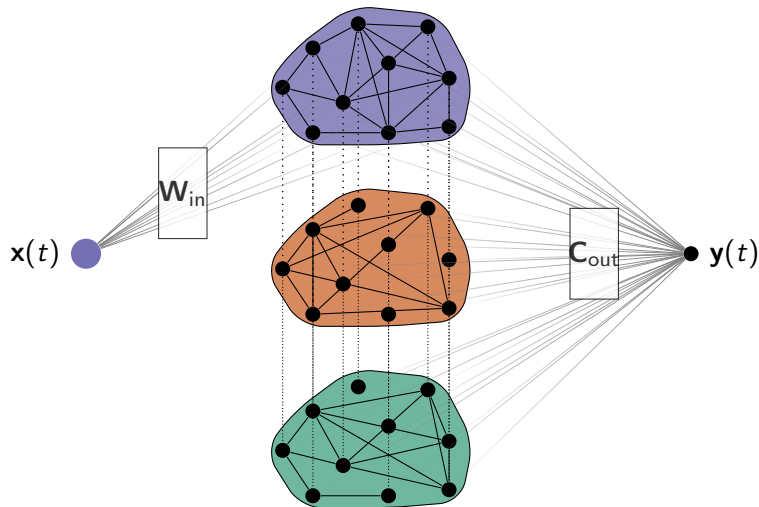


Figure: Ordinally Partitioned Echo State Network.

Results: Approach 2

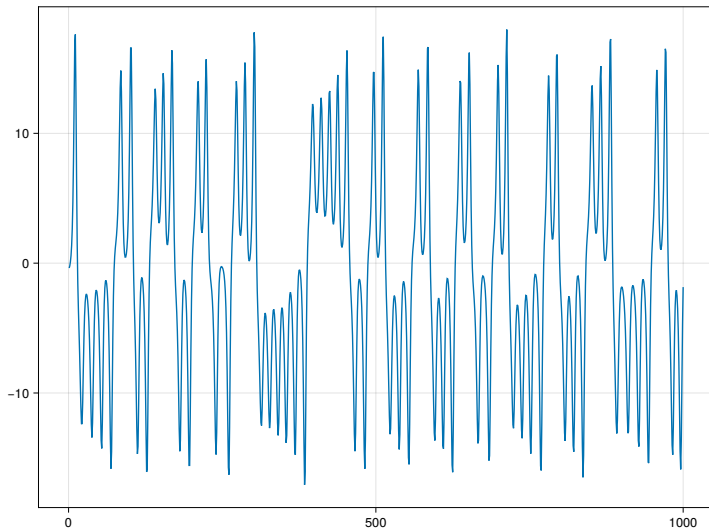


Figure: Freerun prediction for Approach 2 (Sub Reservoirs)

Results: Approach 2

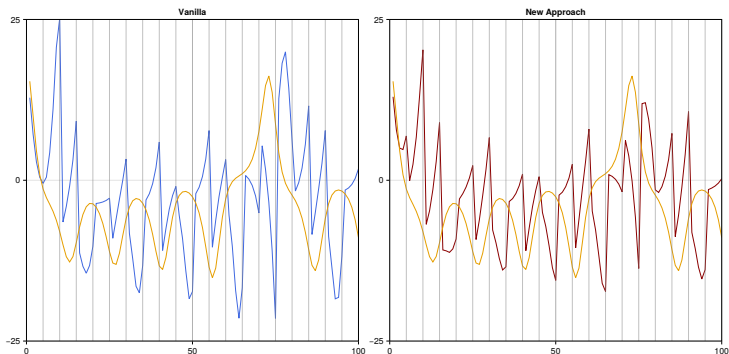


Figure: Multistep prediction for Approach 2 (Sub Reservoirs)

Results: Approach 2

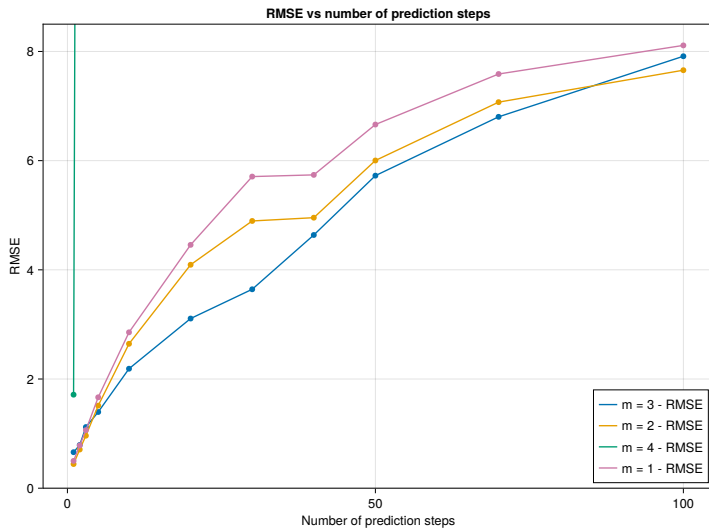


Figure: RMSE vs. Steps for Sub Reservoirs

Results: approach 1 vs. approach 2

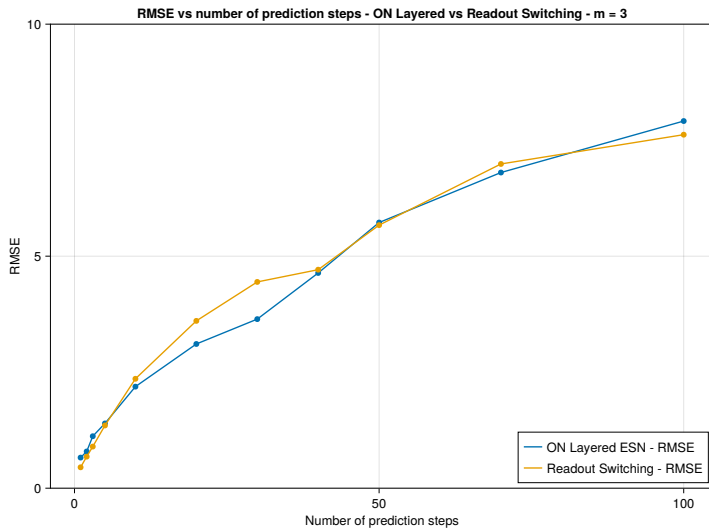


Figure: RMSE vs. Steps for Readout Switching & Sub Reservoirs

Other findings

- ▶ Non-monotonic partition RMSE - no better than overall RMSE
- ▶ Stochastic ordinaly partitioned ESN - very similar results to the deterministic version

Next steps

- ▶ Testing
 - ▶ Qualitative testing including time delay embedding on freerun predictions
 - ▶ Further testing of parameters
 - ▶ Testing for memory capacity
 - ▶ Add noise to the data
 - ▶ Partition the data using a different method and compare
 - ▶ Test using other data including data generated from the Rossler system
- ▶ Building
 - ▶ Prediction n-steps into the future
 - ▶ Different ways of connecting the layers
- ▶ Why?

End

Implementation: Approach 1 (Readout Switching)

When fitting the readout vectors, for each partition p :

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

Where

- ▶ \mathbf{S}_p is \mathbf{S} filtered to the states that results from data points with partition p .
- ▶ \mathbf{Y}_p is \mathbf{Y} filtered to data points that follow immediately from data points with partition p .

And when inferring the prediction at partition p :

$$\mathbf{y}(t) = (\mathbf{C}_{\text{out}})_{P(t)} \mathbf{s}(t)$$

Where $P(t)$ is the partition of $x(t)$.

Implementation: Approach 2 (Sub Reservoirs)

Let k_{part} be the number of nodes in each partition's sub reservoir, and let $\mathbf{W}_{p,q}$ refer to the submatrix of \mathbf{W}_{rec} with indices:

Rows: $\{pk_{part} + 1, pk_{part} + 2, \dots, (p + 1)k_{part}\}$

Columns: $\{qk_{part} + 1, qk_{part} + 2, \dots, (q + 1)k_{part}\}$

Then \mathbf{W}_{rec} is given by:

$$\mathbf{W}_{p,q} = \begin{cases} \mathbf{I}P(p, q), & p \neq q, \\ \mathbf{W}_{ER}, & p = q, \end{cases}$$

where

- ▶ \mathbf{I} is the $k_{part} \times k_{part}$ identity matrix
- ▶ $P(p, q)$ is the probability of transitioning from partition p to partition q
- ▶ \mathbf{W}_{ER} is an Erdos-Renyi randomly instantiated network.

Implementation: Approach 2

Example:

For a k_{layer} of 4 and 3 partitions (not actually possible).

$$W_{rec} =$$

$$\begin{bmatrix} 0.12 & 0 & 0.34 & -0.67 & 0.78 & 0 & -0.23 & 0.11 & -0.56 & 0 & 0.47 & 0 \\ -0.44 & 0.23 & 0 & 0 & -0.76 & 0.54 & 0 & -0.34 & 0 & 0.56 & 0.89 & 0 \\ 0.23 & 0.78 & 0 & 0 & 0.34 & 0.89 & 0 & 0.45 & 0.67 & 0 & -0.89 & -0.75 \\ 0 & 0.34 & 0.89 & 0 & 0.23 & 0 & -0.67 & 0.12 & 0 & 0.56 & 0 & -0.78 \\ 0.67 & 0.12 & 0 & 0.45 & 0 & 0.78 & 0 & 0.89 & 0.12 & 0 & 0.23 & 0 \\ 0 & 0.56 & 0.89 & 0 & 0.23 & 0 & 0.67 & 0 & 0.34 & 0.78 & 0 & 0 \\ 0.45 & 0 & 0.89 & 0.23 & 0 & 0.56 & 0.78 & 0 & 0.12 & 0 & 0.45 & 0.1 \\ 0.78 & 0.12 & 0 & 0 & 0.34 & 0 & 0.45 & 0.56 & 0 & 0.89 & 0 & 0 \\ 0.89 & 0.23 & 0 & 0.56 & 0 & 0.89 & 0 & 0 & 0.34 & 0 & 0.78 & 0.24 \\ 0.34 & 0 & 0.78 & 0 & 0.12 & 0 & 0 & 0.67 & 0 & 0.12 & 0 & 0 \\ 0.78 & 0.12 & 0 & 0.34 & 0 & 0.78 & 0.89 & 0 & 0.23 & 0.56 & 0 & -0.95 \\ 0 & 0.89 & 0.23 & 0 & 0.56 & 0 & 0.89 & 0.12 & 0 & 0.34 & 0 & 0 \end{bmatrix}$$

Implementation: Approach 2

Example:

For a k_{layer} of 4 and 3 partitions (not actually possible).

$$W_{rec} =$$

$$\begin{bmatrix} 0.12 & 0 & 0.34 & -0.67 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.44 & 0.23 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.23 & 0.78 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.34 & 0.89 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.78 & 0 & 0.89 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.23 & 0 & 0.67 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.56 & 0.78 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.34 & 0 & 0.45 & 0.56 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.34 & 0 & 0.78 & 0.24 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.12 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.23 & 0.56 & 0 & -0.95 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.34 & 0 & 0 \end{bmatrix}$$

Implementation: Approach 2

Example:

For a k_{layer} of 4 and 3 partitions (not actually possible).

$$W_{rec} =$$

$$\begin{bmatrix} 0.12 & 0 & 0.34 & -0.67 & 0.1 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 \\ -0.44 & 0.23 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0.2 & 0 & 0 \\ 0.23 & 0.78 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0.2 & 0 \\ 0 & 0.34 & 0.89 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0.2 \\ 0.9 & 0 & 0 & 0 & 0 & 0.78 & 0 & 0.89 & 0 & 0 & 0 & 0 \\ 0 & 0.9 & 0 & 0 & 0.23 & 0 & 0.67 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9 & 0 & 0 & 0.56 & 0.78 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9 & 0.34 & 0 & 0.45 & 0.56 & 0 & 0 & 0 & 0 \\ 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.34 & 0 & 0.78 & 0.24 \\ 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.12 & 0 & 0 \\ 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0.23 & 0.56 & 0 & -0.95 \\ 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0.34 & 0 & 0 \end{bmatrix}$$

Research Proposal: Approach 2

To only feed the input into the relevant layer, we mask all values of \mathbf{W}_{in} except those in the relevant partition.

Given a randomly instantiated input vector \mathbf{W}_{all} of size $k_{part}O$ where O is the number of partitions, we can define \mathbf{W}_{in} as a function of p_t , the partition of the input at time t :

$$\mathbf{W}_{in}(p_t)_i = \begin{cases} (\mathbf{W}_{all})_i, & k_{part}(p_t - 1) < i \leq k_{part}p_t \\ 0, & \text{otherwise.} \end{cases}$$

Example:

$$\mathbf{W}_{all} = [0.12 \quad 0.5 \quad 0.34 \quad -0.67 \quad 0.1 \quad -0.43 \quad 0.98 \quad -0.64 \quad 0.2 \quad 0.45 \quad 0.2 \quad -0.45]$$

$$\mathbf{W}_{in}(1) = [0.12 \quad 0.5 \quad 0.34 \quad -0.67 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$\mathbf{W}_{in}(2) = [0 \quad 0 \quad 0 \quad 0 \quad 0.1 \quad -0.43 \quad 0.98 \quad -0.64 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$\mathbf{W}_{in}(3) = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.2 \quad 0.45 \quad 0.2 \quad -0.45]$$