Restricted Reservoirs on Heterogeneous Timescales

Chester Wringe[1][0000−0002−5764−2181], Susan Stepney[1][0000−0003−3146−5401], and Martin A. Trefzer[2][0000−0002−6196−6832]

[1] Department of Computer Science, University of York, UK
[2] School of Physics, Engineering and Technology, University of York, UK
{chester.wringe,susan.stepney,martin.trefzer}@york.ac.uk

Abstract. The Reservoir Computing model is one that is suited to computing with physical materials. However, the limitations of those materials can lead to low computational power. To address this, we plan to combine reservoirs operating on different timescales together to create a heterogeneous reservoir. We simulate this using a new multiple timescale ESN model. We also introduce "mock materials" so that future works may focus on combining different materials to study the effect. We benchmark our multi-timescale ESN on a multiple timescale problem, MSO.

Keywords: Echo State Networks · Reservoir Computing · multiple timescales · heterogeneous reservoirs · Multiple Superimposed Oscillators.

1 Introduction

Reservoir Computing (RC) is a computational model that is frequently used in low-power *in materio* computing. The model arises from Artificial Neural Networks, where two RC models, Echo State Networks (ESNs) [16] and Liquid State Machines [24] are proposed as a way to efficiently train Recurrent Neural Networks. The RC model relies on training only the output weights of a given system, leaving the inner state (or "reservoir") as a black box. As such, the inner state can be replaced by various physical systems, so long as they have sufficiently rich dynamics [4, 7, 11]. These *in materio* reservoirs can make ideal low-power devices that excel at time-series recognition.

The computational properties of materials do not always scale well with the physical size of the reservoir [8]. In order to more fully exploit the properties of physical materials, we investigate combining multiple reservoirs. Combining multiple homogeneous ESNs together can lead to similar performance to a larger ESN with the same total number of nodes on simple tasks [35]; however, this does not appear to extend to more complicated tasks, particularly those on multiple timescales.

Here, we study whether this performance can be improved by running the component ESNs on multiple timescales. To do this, we introduce a multi–timescale ESN model, built on our previous Restricted ESN model. We also refine our simulations, so that our simulated ESNs more closely resemble the kinds of physical materials that might be used.

2 Background: Combining ESNs

There are a number of works investigating combining multiple Reservoir Computers, particularly ESNs. One of the more popular ones is the deep-ESN model [5, 14, 15, 22], based on deep learning networks. Other multi-ESN networks include the dual-reservoir network (DRN) [21], dual-reservoir model [33], and the multilayered echo state machine (ML-ESM) [25]. The Reservoir with Random Static Projections (R^2SP) [3] and the ϕESN [13] are models that combine ESNs with Extreme Learning Machines (ELMs).

Some multi-reservoir models have per-subreservoir input and output layers, forming a Modular ESN [35]. Some examples of this in the literature are the Dynamic Feature Discoverer (DFD) [18], the modular ESNs are also used in acoustic modelling [32, 33], and the ConvESN [23]. The majority of multi-reservoir networks, however, combine reservoirs by combining the reservoir state and weights using a direct sum, and then treating these as the inner state of the larger reservoir, with a single input and output layer. These Restricted ESNs [35] are the focus of this paper.

The ESNs studied in our work are restricted ESNs, and are based on the Reservoir of Reservoirs (RoR) [6]. They also resemble the scale-free highly clustered ESN (SHESN) [10], the hierarchically clustered ESN (HESN) [20], and the modular ESN [29]. We also base much of our work on the Decoupled ESN (DESN) [36], as it explores the idea of decoupling ESNs from each other as a method of solving tasks which are too difficult for standard ESNs.

Of the models described above, the ConvESN [23] and Dynamic Feature Discoverer (DFD) [18] both work on multiple timescales. Multi-timescale Restricted ESNs can also be simulated using leakage rates [26].

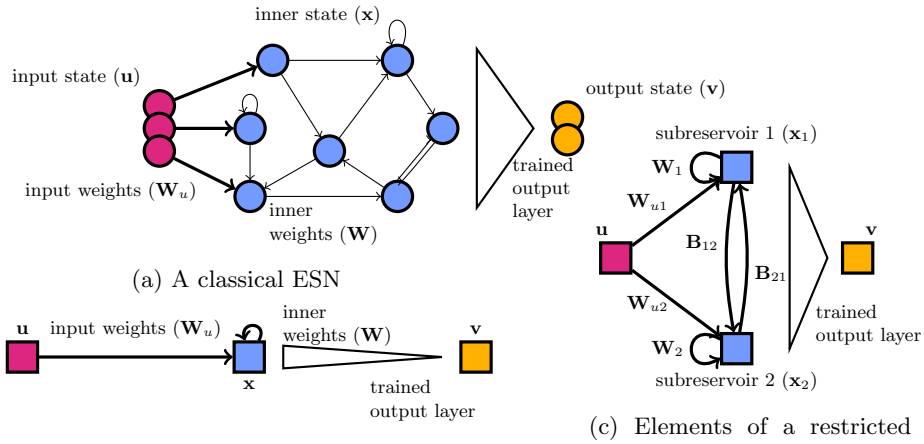
3 Restricted ESNs on Multiple Timescales

3.1 Argument for Temporal Heterogeneity

Simple Restricted ESNs perform as well as standard ESNs at simple tasks such as the NARMA and sunspot prediction benchmarks, but more demanding benchmarks, such as the Multiple Superimposed Oscillators benchmark (MSO), perform worse using restricted ESNs [35]. The worse performance can be addressed by decoupling the subreservoirs from each other [36]. A recent review paper [38] has compared this behaviour to that of sdecoupled neurons communicating with each other [12]. The neurons in [12] are decoupled temporally, using different rhythms; the Decoupled ESN [36] uses physical decoupling, through the weights connecting the subreservoirs. Here, we investigate whether we can recreate this decoupling using multiple rhythms.

3.2 Restricted ESNs

The restricted ESN model [35] is based on the classical Echo State Network model [16,19] (fig.1a). The classical ESN model is a Random Recurrent Network, where the training is performed on only the output weights.



(a) A classical ESN

(b) Elements of a classical ESN

(c) Elements of a restricted ESN

Fig. 1: An example of a possible classical ESN (a) with $n = 7$ nodes, and (b) an abstraction of its different elements. The ESN takes one or more inputs $\mathbf{u}$ which are then sent to the inner state $\mathbf{x}$ through weighted input edges $\mathbf{W}_u$. The weights within the reservoir, $\mathbf{W}$, are recurrent and randomly initialised. The output state $\mathbf{v}$ receives the inner state through the trained output layer. (c) Elements of a restricted ESN with 2 subreservoirs, showing the partitioned state and components of the internal weight matrix.

The ESN at time t takes an input $\mathbf{u}(t)$, has an inner state $\mathbf{x}(t)$, and produces an output $\mathbf{v}(t)$, as defined [31] by the update equation:

$$\begin{aligned} \mathbf{x}(t+1) &= f(\mathbf{W}_u\mathbf{u}(t) + \mathbf{W}\mathbf{x}(t)) \\ \mathbf{v}(t+1) &= \mathbf{W}_v\mathbf{x}(t) \end{aligned} \tag{1}$$

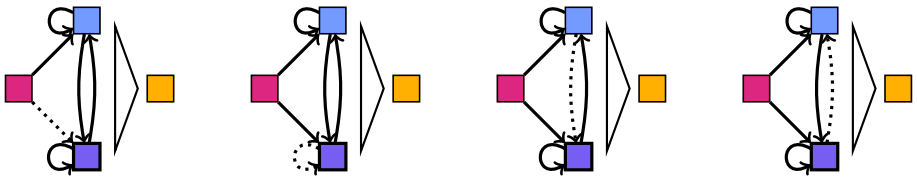
where $\mathbf{W}_u$ is the random input weight matrix, $\mathbf{W}$ is the random internal weight matrix, $\mathbf{W}_v$ is the trained output weight matrix, f is a nonlinear function, typically tanh.

The restricted ESN [26] is a variant where we partition the reservoir state into smaller subreservoirs (resulting in a block diagonal weight matrix), with restricted connections between the subreservoirs (sparse off-block-diagonal components).

The state vector $\mathbf{x}$ of a restricted ESN with n subreservoirs is the concatenation of the subreservoir state vectors:

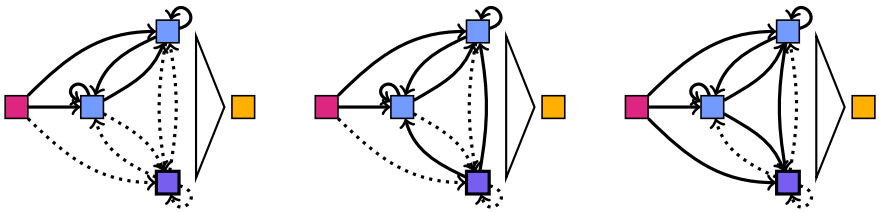
$$\mathbf{x} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_n \end{pmatrix} \tag{2}$$

where $\mathbf{x}_i$ is the state of the subreservoir i. The input weight matrix and internal weight matrix are analogous concatenations of input weights to individual



(a) $\mathbf{W}_{u2} = \mathbf{O}$ (b) $\mathbf{W}_2 = \mathbf{I}$ (c) $\mathbf{B}_{12} = \mathbf{O}$ (d) $\mathbf{B}_{21} = \mathbf{O}$

Fig. 2: Four components of how a subreservoir might "sleep" (notation as in fig.1c, with a dashed line indicating a sleeping communication). In each case, the lower subreservoir is asleep, with one of its connections affected: (a) it receives no external input; (b) it receives no input from its previous state; (c) it receives no input from other reservoirs; (d) it sends no output to other subreservoirs.



(a) total sleep (b) input sleep (c) output sleep

Fig. 3: The sleep modes used for our experiments, which determine the behaviour of reservoirs when asleep. In each figure, the lower subreservoir sleeps when $t = t_{\text{sleep}}$. The weight matrices that are changed are indicated by the dotted lines.

subreservoirs, of internal subreservoir weights, and of matrices describing the connections between subreservoirs:

$$\mathbf{W}_u = \begin{pmatrix} \mathbf{W}_{u1} \\ \vdots \\ \mathbf{W}_{un} \end{pmatrix} \qquad \mathbf{W} = \begin{pmatrix} \mathbf{W}_1 & \dots & \mathbf{B}_{1n} \\ \vdots & \ddots & \vdots \\ \mathbf{B}_{n1} & \dots & \mathbf{W}_n \end{pmatrix} \tag{3}$$

where $\mathbf{W}_{ui}$ is the input weight matrix of subreservoir i, $\mathbf{W}_i$ is the weight matrix of subreservoir i, and $\mathbf{B}_{ij}$ gives the connections from subreservoir i to subreservoir j. Note that $\mathbf{B}_{ij}$ is square iff subreservoirs i and j are the same size. The components of a two-substate restricted reservoir are shown in figure 1c.

We could use the same approach to assign a different function f_i to each subreservoir, for example, to model different material properties. This can be set to the identity function, **id**.

3.3 Sleep modes

Here we introduce a model that allows us to define timescales on a per-subreservoir basis. We use the idea of a "sleeping" subreservoir: Instead of updating on every

clock tick, a subreservoir might only update "normally" on some of them. During the rest of the clock ticks, we may change the input and inner weight matrices to any combination of the following effects (see figure 2):

- not receiving input from its previous state ($\mathbf{W}_n = \mathbf{I}$)
- not receiving input from the input layer ($\mathbf{W}_{un} = \mathbf{0}$)
- not receiving communication from other subreservoirs ($\mathbf{B}_{\_n} = \mathbf{0}$)
- not sending communication to other reservoirs ($\mathbf{B}_{n\_} = \mathbf{0}$)

The trained output weights are not affected by the sleeping reservoirs, as these are externally set during the training phase, and do not have any effect on the reservoir state. Instead, the output will always see the last updated state of every subreservoir.

We can compose these individual sleep components in multiple ways. For our experiments here, we define three different sleep modes:

- *total sleep*: when the subreservoir is asleep, no communication takes place between it and other subreservoirs, nor from the external input; figure 3a.
- *input sleep*: no inputs or communication from other reservoirs affect the state of the sleeping reservoir, but communication from it can still be received by the other subreservoirs; figure 3b.
- *output sleep*: the subreservoir reacts to external and internal inputs, but does not send out any communication to other subreservoirs; figure 3c.

In each of the sleep modes, the subreservoir does not react to its own past input. At this time, we keep the transfer function as **tanh** during both sleep and wake states.

3.4 Multiple Timescales with an extended transfer function

Once we have defined what the sleep state for a given subreservoir entails, we can extend our update equation to reflect this. In order to do this, we use time dependent weights in our transfer function. For illustration, consider the case of a single reservoir that is awake every odd timestep, but that sleeps (receives no input, and no internal update) every even timestep. We would have:

$$\mathbf{x}(t+1) = f(\mathbf{W}_u(t)\mathbf{u}(t) + \mathbf{W}(t)\mathbf{x}(t)) \tag{4}$$

$$\mathbf{W}_u(t) = \mathbf{W}_u, \mathbf{W}(t) = \mathbf{W} \quad (t \text{ odd})$$

$$\mathbf{W}_u(t) = \mathbf{0}, \mathbf{W}(t) = \mathbf{I} \qquad (t \text{ even}) \tag{5}$$

For a standard ESN, this model would be overly complicated; we can easily simulate a sleep state by removing every other input value from our input set. In a restricted ESN, however, this model allows us to have one substate sleep while the other substate is awake. For illustration, consider a two subreservoir case, where subsreservoir 1 behaves as above, and also receives no input from,

Algorithm 1 building the restricted weight matrix at time t

```
W_BASE := weight matrix, with all subreservoirs awake
for t in timesteps do
    for i in subreservoirs do
        if subreservoir i is awake then
            C_i^t := C_i^wake
        else
            C_n^t := C_i^sleep
    W^t := ⊙_1^n C_i^t ⊙ W_BASE
```

and sends no output to, the other subreservoir while asleep, while subreservoir 2 is awake all the time. We then have:

$$\mathbf{W}_u(t) = \begin{pmatrix} \mathbf{W}_{u1} \\ \mathbf{W}_{u2} \end{pmatrix}, \mathbf{W}(t) = \begin{pmatrix} \mathbf{W}_1 & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{W}_2 \end{pmatrix} (t \text{ odd})$$
$$\mathbf{W}_u(t) = \begin{pmatrix} \mathbf{0} \\ \mathbf{W}_{u2} \end{pmatrix}, \mathbf{W}(t) = \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_2 \end{pmatrix} \quad (t \text{ even}) \tag{6}$$

This is the simplest multi-transfer equation model: it allows for only one subreservoir to be asleep at a time. Here, we use an ESN with three subreservoirs, each with its own sleep/wake *rhythm*. To allow for this, we must be able to build the weight matrix for the full reservoir at each timestep, a process which we describe in section 3.5.

3.5 Building the reservoir edge matrices

Our model allows us not only to have subreservoirs be in their wake or sleep state independently of each other, but also for each reservoir to have their own sleep mode for communicating with other subreservoirs when asleep. In order to allow for this, we need to build the weight matrix at every timestep t.

The $N \times N$ full weight matrix is built by taking the Hadamard product (elementwise multiplication, denoted $\odot$) of $\mathbf{W}_{\text{BASE}}$, the inner weight matrix of our reservoir when every subreservoir is awake, and the $N \times N$ connectivity matrix $\mathbf{C}_i^t$ for each subreservoir i. $\mathbf{C}_i^t$ is constructed from:

- an $N_i \times N_i$ matrix $\mathbf{c}_i^t$ corresponding to the sleep state of subreservoir i at time t (**1** if awake, **0** if asleep; see fig.2b)
- horizontal and vertical strips of blocks corresponding to the sleep modes of the connections between i and the other subreservoirs, at time t (**1** if awake, **0** if asleep; see fig.2c,d)
- an all-ones matrix in every other block.

We illustrate this in algorithm 1 and figure 4.

4 Mock Materials

When studying combining heterogeneous reservoirs, we encounter a plethora of possible parameter values, making them difficult to compare over. Some of these

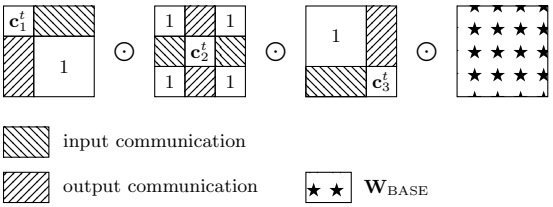


Fig. 4: Building the restricted reservoir's weight matrix $\mathbf{W}^t$ for timestep t, taking into account which reservoirs are awake and which are not.

parameters include: density, spectral radius, architecture/topology, number of nodes. When studying heterogeneous timescales, we introduce the following further parameters: rhythm (pattern of awake/asleep); tempo (number of timesteps that a reservoir is asleep over); sleep mode (behaviour of the reservoir when asleep).

This results in a combinatorial explosion of possibilities. In order to reduce this space, our work focuses on a set of three simulated materials, each with their own fixed properties. These materials are inspired by materials and models used for reservoir computing, but some liberties are taken, both because the ESN model does not fully correspond to existing materials, and to provide a larger range of properties. We also use different sleep modes for each material, both to reflect the fact that materials may have different sleep properties, and to study a range of different modes.

Here, we simulate three materials by constraining various parameters of the ESN, and we focus on the effect of different timescales over multiple subreservoirs made of a single simulated material. In future work, we will focus on combining the materials. As such, this paper does not describe a tempo or rhythm for each individual material, instead, the rhythms for each subreservoir is asleep are described in section 5.

4.1 Ring

The Ring mock material topology [28] is intended to be a spatial representation of delay-line reservoirs [2]. The nodes in this substrate are laid out in a ring, connecting only to themselves and to a single neighbour.

The weight matrix of the ring material has weights drawn from $U[-a, a]$, normalised to a spectral radius $\rho(\mathbf{W}) = 1$.

A ring subreservoir communicates with other subreservoirs via two "spine" nodes: one node within the ring receives any input communications from other subreservoirs, while another transmits all output communications. This communication model is again based on the delay-line reservoir, where the input is fed into the reservoir via a single virtual node at every timestep τ. We use two distinct spine nodes to allow for some processing of information before the state is communicated to the other reservoirs.

For this material, we use total sleep mode for most experiments.

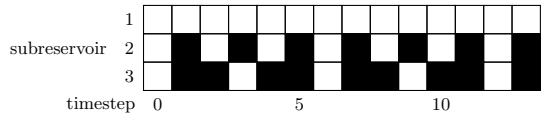


Fig. 5: Sleep rhythms used for the multiple timescale experiment. At every timestep, if the subreservoir is awake, the square is white, and if it is asleep, the square is black.

4.2 Lattice

The Lattice mock material is inspired by reservoirs made out of magnetic ring arrays [1], which can be laid out in a grid formation. As such, the nodes are laid out in a grid, with each node having an edge to itself and to every node in its von Neumann neighbourhood. The edges do not wrap, as this would instead form a torus [9].

The weight matrix of the ring material has weights between $U[-a, a]$, adjusted to ensure the spectral radius $\rho(\mathbf{W}) = 1$.

Communication between a lattice subreservoirs and other subreservoirs in a restricted ESN takes place via "sides" of the grid, in order to reinforce the physical idea of distance between unconnected nodes. In preliminary work, the model was arranged so that the nodes on one side of the lattice received all the input communications from other reservoirs, while the nodes on the opposite side transmitted outputs. However, this model performed poorly, as it took many timesteps for information to propagate through the subreservoir and on to its siblings. Here, we use a single side for both input and output, leading to improved results.

For this material, we use input sleep mode for most experiments.

4.3 Bucket

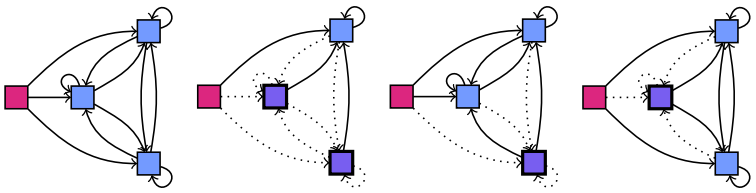
The Bucket mock material is inspired by a bucket of water, one of the first materials in which reservoir computing was performed [11]. Unlike the other two materials, this one has a fully connected weight matrix, giving us a "well-mixed bucket". To reflect the relatively simple dynamics of the system, the spectral radius $\rho(\mathbf{W}) = 0.8$.

The communication between a bucket subreservoir and other subreservoirs is performed by four communication nodes, which both receive input communications and transmit output ones. This allows us to study three restricted reservoirs with different amounts of communication between them.

For this material, we use output sleep mode for most experiments.

5 Experimental setup

For each material, we compare the performance of a restricted ESN with three 64-node subreservoirs on a single timescale, to one on three timescales. For the multi-timescale experiment, each subreservoir has its own sleep/wake rhythm:



(a) $t = 1$ (b) $t = 2$ (c) $t = 3$ (d) $t = 4$

Fig. 6: Connections between subreservoirs during the first four timesteps of our lattice experiment. (a) $t = 1$, all subreservoirs are awake; (b) $t = 2$, both subreservoir 2 and subreservoir 3 sleep; (c) $t = 3$, only subreservoir 2 is asleep; (d) $t = 4$, only subreservoir 3 is asleep. The dotted arrows indicate which connections are affected by the sleeping reservoirs.

one is awake every timestep, one is awake every other timestep, and one is awake one timestep out of three (Fig. 5). No attempt is made here to match the rhythms used to the task; instead, we use the simplest nontrivial set of rhythms.

We illustrate how this rhythm structure affects the connections between subreservoirs in our lattice experiment over four timesteps in figure 6.

5.1 Scaling the Weight Matrix

Scaling the weight matrix is a necessary step when working with Echo State networks, in order to ensure that the Echo State Property is kept and so the reservoir state doesn't diverge. This is often accomplished by scaling the spectral radius ρ of the weight matrix $\mathbf{W}$. The spectral radius is a global parameter of the Echo State Network that has a large effect on the performance. It is commonly accepted that a larger spectral radius can lead to violating the echo state property, while keeping the spectral radius smaller than 1 ensures the echo state property [27].

Our mock materials each have a given spectral radius. However, the spectral radius of the full reservoir weight matrix (composed of three subreservoirs of the same mock material) may be different from the spectral radius of the subreservoirs, because of the extra off-block-diagonal communication edges (matrices $\mathbf{B}_{ij}$). Hence the full weight matrix needs to be rescaled. In our case, preliminary experiments show that scaling the spectral radius of the full weight matrix to one does not give good performance; instead, we use the more restrictive method of scaling the largest singular value, $\bar{\sigma}$, to one [37]. This results in a spectral radius less than one, but here it gives improved performance.

The largest singular value of the full reservoir could be scaled to 1 by uniformly scaling the entire weight matrix. However, in the case of our restricted reservoir model, this would affect the properties of the subreservoirs, by scaling the weights within each of them. This would risk losing any properties that are particular to our mock material based on its weight matrix. This would also not readily transfer to using physical materials, as their effective weight matrix is

Algorithm 2 Scaling the off-diagonals

1: $\mathbf{W}_B := \mathbf{W}$ from eqn.3 with each $\mathbf{W}_i$ replaced by all-ones
2: $\mathbf{W}_B := \mathbf{W}_B \times$ random matrix
3: $\bar{\sigma} :=$ largest singular value of $\mathbf{W}_B$
4: $\mathbf{W}_B := \mathbf{W}_B / \bar{\sigma}$

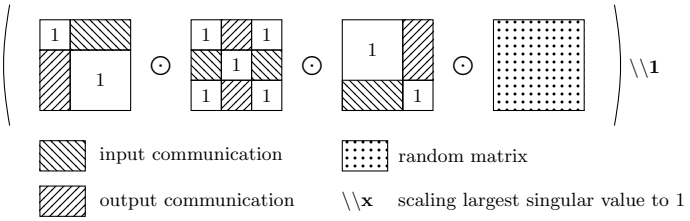


Fig. 7: Representation of Scaling the Off-diagonals

a given. So instead, we scale only the off-diagonal connections between different subreservoirs to ensure the overall scaling, as illustrated in algorithm 2 and figure 7.

Once $\mathbf{W}_B$ is scaled, we superimpose the weight matrices for the subreservoirs over the diagonals. This step does change the scaling slightly, but not significantly for our purposes.

5.2 The MSO* benchmark

The Multiple Superimposed Oscillators is a prediction benchmark task. The aim of the task is to predict $y(t+1)$ given $y(t)$ of the function

$$y(t) = \sum_{i=0}^{n-1} \sin \alpha_i t \tag{7}$$

where $\alpha \in [0.2, 0.311, 0.42, 0.51, 0.63, 0.74, 0.85, 0.97]$.

The first use of the MSO-2 benchmark, then referred to as "additive dynamics" was introduced in a presentation by Jaeger [17]. It has subsequently been extended to include higher frequency α values, up to MSO-5 [34] and MSO-8 [30].

We use a variation on this task referred to as MSO* [35]. We use this modification due to the fact that the slowest reservoir in our experiments is only "awake" one out of every three timestep. In order to prevent potential undersampling of the higher frequency tasks (as illustrated in fig. 8), we sample the MSO function 8 times more frequently. The modified equation is given by:

$$y^*(t) = \sum_{i=1}^{n} \sin\left(\frac{\alpha_i t}{8}\right) \tag{8}$$

This input is then scaled to be between -0.5 and 0.5.

We choose this task as it includes multiple incommensurable timescales. The task has previously been approached by physically decoupling the reservoirs

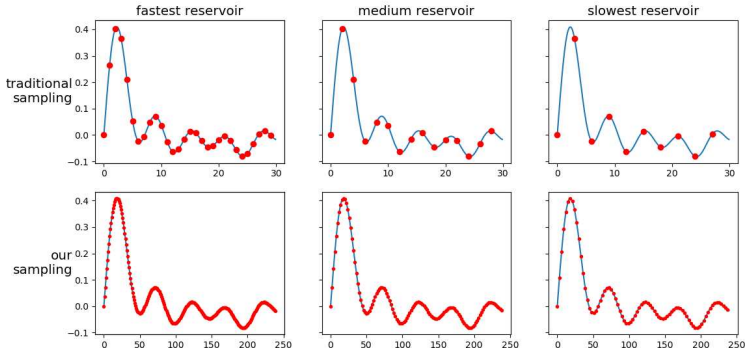


Fig. 8: Illustration of how we sample the MSO*-8 function, compared to the typical MSO-8 sampling. The higher row shows the more commonly used sampling, where each datapoint is $y(t)$. As we can see though, in a multi–timescale reservoir, this would lead to undersampling by the slower reservoirs. Reduce the MSO frequency by a factor of 8, allowing reservoirs of all speeds to see the details of the curve.

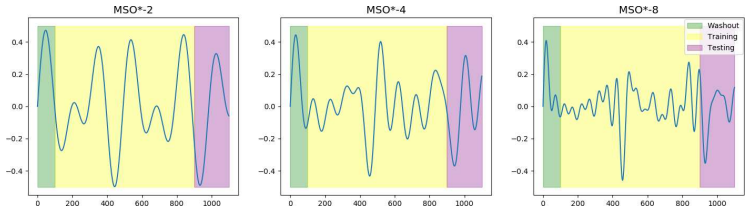


Fig. 9: Data used in our MSO* experiments, which each include 1100 datapoints. The differently coloured zones indicate the washout, training, and testing points.

from each other [36]. In this work, we add a temporal decoupling element to this physical decoupling[3] in order to find out whether this has an impact on the results.

In this paper, we study three MSO* problems: MSO*-2 , MSO*-4, and MSO*-8 (fig. 9). We use the same data lengths as previous work [36]: 100 timesteps for the washout, 600 steps for the training, and 200 steps for testing.

6 Results

The results for all the experiments show that generally, the multi-timescale perform worse than the single timescale ones. This effect does not apply as strongly across materials, however, being more pronounced in the "bucket" material and almost nonexistent with the "ring" material.

[3] The physical decoupling is accomplished here by scaling the largest singular value of the weight matrix to 1.

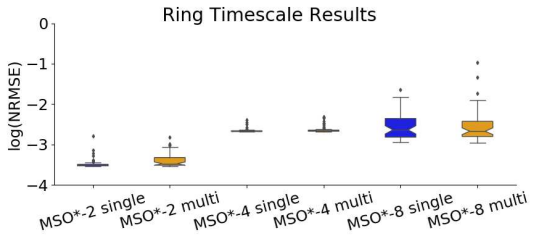


Fig. 10: Ring material results for the MSO* experiments. The results are given as $\log_{10}(NRMSE)$.

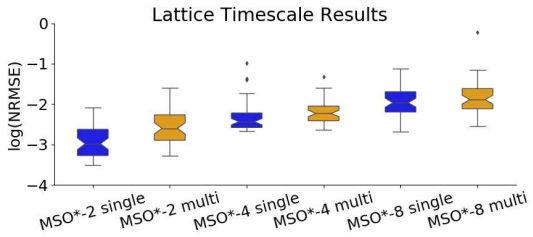


Fig. 11: Lattice material results for the MSO* experiments. The results are given as $\log_{10}(NRMSE)$.

The "ring" material is the one with the best performance overall. The performance of the single timescale reservoir is very similar to the "bucket" single timescale, both of which are slightly better than the lattice material. More promisingly, with this material, the multi-timescale case gives us a similar MSO*-2 performance in the best case (with a higher variance), a very similar performance overall for MSO*-4, and a slightly better performance for MSO*-8. In the MSO*-8 case, this leads to the multi-timescale "ring" reservoir having the best performance at this task overall.

The "lattice" model performs poorly compared to the other materials, in both the single and multi-timescale case. This performance is worsened by adding multiple timescales, although the effect diminishes as the task gets harder.

The "bucket" material has single-timescale results that are similar to that of the "ring" material. In this material though, the multi-timescale reservoir performs consistently worse than the single-timescale one. As with the lattice material, this difference in performance diminishes as the task gets more complex.

Two likely reasons for the difference in the effect of multiple timescales are:

- The sleep mode used has a significant effect on how well a multi–timescale reservoir performs
- the physical properties of our reservoirs may suit them more or less to a multi–timescale approach.

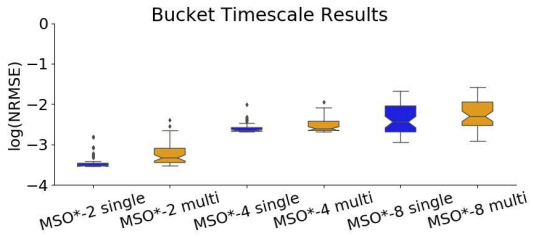
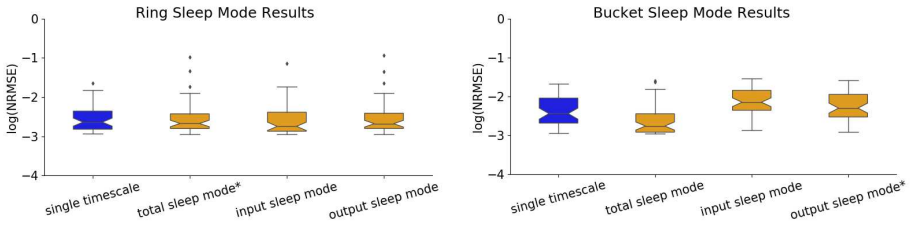


Fig. 12: Bucket material results for the MSO* experiments. The results are given as $\log_{10}(NRMSE)$.

In order to test these hypotheses, we run the MSO*-8 experiment on the "ring" (fig. 13a) and "bucket" (fig. 13b) materials, but this time using all three different sleep modes.



(a) (b)

Fig. 13: Results for the experiments performed on the (a) Ring and (b) Bucket materials, using different sleep modes. The previously used sleep mode is marked by *.

We see in these additional results that both hypotheses play a factor. With the "ring" material, the different types of sleep modes appear to have very little effect on the results. In each case, the multi-timescale model works better than the single–timescale one, but no sleep mode appears to be better than the other. With the "bucket" material, however, the type of sleep mode has a very large effect, with the model using ring-style sleep modes performing as well as the single-timescale model, and the other two performing much worse.

This leads us to conclude that the different sleep modes can have an effect on performance, but that this effect changes depending on the other properties of the reservoir.

7 Conclusions

In this work, we introduce an extension of the ESN model that allows us to operate at different timescales over different regions of the reservoir, which we call subreservoirs. We introduce "sleep states" for the subreservoirs, which let us study different possible implementation of multiple timescales. To evaluate these

timescales, we introduce "simulated materials", which allow us to experiment over a number of different parameter sets and sleep models.

We find that adding multiple timescales can sometimes improve performance of reservoirs at certain multi-timescale tasks, such as the MSO*-8 task. This improvement in performance depends on both the mock material properties, and on the sleep mode used. Future work is needed to explore this in further depth, for which this work can be used as a baseline.

Our model allows for mixing and matching of different materials, as well as sleep modes. In this work, we focus on the heterogeneity of the *timescales* of the different reservoirs. In future work, we will study heterogeneous *materials*.

Acknowledgment

This work was made possible by PhD studentship funding from the Computer Science Department of the University of York.

References

1. Allwood, D.A., et al.: A perspective on physical reservoir computing with nanomagnetic devices. Appl. Phys. Let. **122**(4), 040501 (2023)
2. Appeltant, L., et al.: Information processing using a single dynamical node as complex system. Nat. Commun. **2**, 468 (2011)
3. Butcher, J.B., et al.: Extending reservoir computing with random static projections. In: ESANN 2010. pp. 303–308 (2010)
4. Caluwaerts, K., et al.: Locomotion Without a Brain: Physical Reservoir Computing in Tensegrity Structures. A. Life **19**(1), 35–66 (2013)
5. Canaday, D., et al.: Model-free control of dynamical systems with deep reservoir computing. J. Phys. Complex. **2**(3), 035025 (2021)
6. Dale, M.: Neuroevolution of hierarchical reservoir computers. In: GECCO 2018. pp. 410–417. ACM (2018)
7. Dale, M., et al.: Evolving carbon nanotube reservoir computers. In: UCNC 2016. LNCS, vol. 9726, pp. 49–61. Springer (2016)
8. Dale, M., et al.: Computing with magnetic thin films: Using film geometry to improve dynamics. In: UCNC 2021. LNCS, vol. 12984, pp. 19–34. Springer (2021)
9. Dale, M., et al.: Reservoir computing quality: connectivity and topology. Nat. Comput. **20**(2), 205–216 (2021)
10. Deng, Z., Zhang, Y.: Collective behavior of a small-world recurrent neural system with scale-free distribution. IEEE TNN **18**(5), 1364–1375 (2007)
11. Fernando, C., et al.: Pattern recognition in a bucket. In: Advances in A. Life. pp. 588–597. Springer (2003)
12. Fries, P.: Rhythms for cognition: communication through coherence. Neuron **88**(1), 220–235 (2015)
13. Gallicchio, C., Micheli, A.: Architectural and Markovian factors of echo state networks. Neural Netw. **24**(5), 440–456 (2011)
14. Gallicchio, C., Micheli, A.: Echo state property of deep reservoir computing networks. Cognit. Comput. **9**(3), 337–350 (2017)
15. Gallicchio, C., et al.: Deep reservoir computing: A critical experimental analysis. Neurocomputing **268**, 87–99 (2017)
16. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks – with an erratum note. German Ntnl. Research Ctr. for Info. Tech. GMD Tech. Rep. **148**(34), 13 (2001)

17. Jaeger, H.: The echo state approach to recurrent neural networks (pres.) (2004), https://www.ai.rug.nl/minds/uploads/ESNStandardSlides.pdf, accs. 29 November 2023
18. Jaeger, H.: Discovering multiscale dynamical features with hierarchical echo state networks. Tech. Rep. TR-10, Jacobs University Bremen (2007)
19. Jaeger, H., Maass, W., Principe, J.: Special issue on echo state networks and liquid state machines. Neural Netw. **20**(3), 287–289 (2007)
20. Jarvis, S., et al.: Extending stability through hierarchical clusters in echo state networks. Front. Neuroinform. **4** (2010)
21. Ma, Q., et al.: Decouple adversarial capacities with Dual-Reservoir network. In: ICONIP 2017. pp. 475–483. Springer (2017)
22. Ma, Q., et al.: Deep-ESN: A multiple projection-encoding hierarchical reservoir computing framework. arXiv:1711.05255 [cs.LG] (2017)
23. Ma, Q., et al.: Convolutional multitimescale echo state network. IEEE Trans Cybern **51**(3), 1613–1625 (2021)
24. Maass, W., Natschläger, T., Markram, H.: Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. Neural Comput. **14**(11), 2531–2560 (2002)
25. Malik, Z.K., et al.: Multilayered echo state machine: A novel architecture and algorithm. IEEE Trans Cybern **47**(4), 946–959 (2017)
26. Manneschi, L., et al.: Exploiting multiple timescales in hierarchical echo state networks. Front. Appl. Math. Stat. **6** (2021)
27. Montavon, G., et al.: Neural Networks: Tricks of the Trade. Springer (2012)
28. Rodan, A., Tino, P.: Minimum complexity echo state network. IEEE Trans. Neural Netw. **22**(1), 131–144 (Jan 2011)
29. Rodriguez, N., et al.: Optimal modularity and memory capacity of neural reservoirs. Netw Neurosci **3**(2), 551–566 (Apr 2019)
30. Roeschies, B., Igel, C.: Structure optimization of reservoir networks. Logic J. of the IGPL **18**(5), 635–669 (2010)
31. Stepney, S.: Non-instantaneous information transfer in physical reservoir computing. In: UCNC 2021. pp. 164–176. Springer (2021)
32. Triefenbach, F., et al.: Phoneme recognition with large hierarchical reservoirs. Adv. Neural Inf. Proc. Syst. **23**, 2307–2315 (2010)
33. Triefenbach, F., et al.: Acoustic modeling with hierarchical reservoirs. IEEE TASLP **21**(11), 2439–2450 (2013)
34. Wierstra, D., et al.: Modeling systems with internal state using evolino. In: GECCO 2025. pp. 1795–1802. ACM (2005)
35. Wringe, C., et al.: Modelling and evaluating restricted esns on single-and multi-timescale problems (2023)
36. Xue, Y., et al.: Decoupled echo state networks with lateral inhibition. Neural Netw. **20**(3), 365–376 (2007)
37. Yildiz, I.B., et al.: Re-visiting the echo state property. Neural Netw. **35**, 1–9 (2012)
38. Zhang, H., Vargas, D.V.: A survey on reservoir computing and its interdisciplinary applications beyond traditional machine learning. IEEE Access **11**, 81033–81070 (2023)