ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

Reservoir Memory Networks

Claudio Gallicchio and Andrea Ceni [*]

Department of Computer Science, University of Pisa
Largo Bruno Pontecorvo 3 - 56127 Pisa, Italy

Abstract. We introduce Reservoir Memory Networks (RMNs), a novel class of Reservoir Computing (RC) models that integrate a linear memory cell with a non-linear reservoir to enhance long-term information retention. We explore various configurations of the memory cell using orthogonal circular shift matrices and Legendre polynomials, alongside non-linear reservoirs configured as in Echo State Networks and Euler State Networks. Experimental results demonstrate the substantial benefits of RMNs in time-series classification tasks, highlighting their potential for advancing RC applications in areas requiring robust temporal processing.

1 Introduction

Reservoir Computing (RC) [1] represents a powerful paradigm in the design of Recurrent Neural Networks (RNNs), widely recognized for its efficiency and reduced training requirements. This approach is particularly relevant in the realms of pervasive artificial intelligence (AI) and neuromorphic hardware implementations, where RC facilitates low-power, high-speed processing, aligning with the goals of sustainable AI. Traditional RC architectures, however, often struggle with processing sequences in tasks that require long-range retention of information, a critical capability for many advanced AI applications. Recent studies have shown that the integration of linear dynamical systems within RNNs can significantly enhance the propagation of information across long sequences [2, 3], providing a promising direction for overcoming classical RC limitations.

In this paper, we introduce a novel class of RC systems, the Reservoir Memory Networks (RMNs), which combine a linear memory cell with a non-linear processing reservoir. This dual-reservoir approach aims to harness the strengths of both linear and non-linear dynamics to efficiently manage long-term dependencies. The efficacy of our proposed approach is empirically evaluated across diverse time-series classification tasks.

2 Reservoir Computing

We start by introducing the Echo State Network (ESN) model [4], the most widely known and used RC model in the literature. The architecture of an ESN includes a fixed randomized recurrent hidden layer, called *reservoir*, paired with a feed-forward trainable *readout*. To fix our mathematical notation, we use N_h to indicate the number of reservoir neurons, and N_x for the number of input features at each time-step (i.e., the dimensionality of the driving input signal). Moreover, we use $\mathbf{h}(t) \in \mathbb{R}^{N_h}$ and $\mathbf{x}(t) \in \mathbb{R}^{N_x}$, respectively, to denote the state

[*]Work supported by EU-EIC EMERGE (Grant No. 101070918) and NEURONE, a project funded by the Italian Ministry of University and Research (PRIN 20229JRTZA).

of the reservoir and the input at time-step t. Referring to the formulation of leaky integrator neurons from [5], the operation of the reservoir can be described mathematically by the following iterated map:

$$\mathbf{h}(t) = (1-\alpha)\mathbf{h}(t-1) + \alpha \tanh\big(\mathbf{W}_h \mathbf{h}(t-1) + \mathbf{W}_x \mathbf{x}(t) + \mathbf{b}_h\big), \qquad (1)$$

where $\mathbf{W}_\mathbf{h} \in \mathbb{R}^{N_h \times N_h}$ is the recurrent reservoir weight matrix, $\mathbf{W}_\mathbf{x} \in \mathbb{R}^{N_h \times N_x}$ is the input weight matrix, $\mathbf{b}_h \in \mathbb{R}^{N_h}$ is the bias vector, $\alpha \in (0,1]$ is the leaking-rate, and $\tanh(\cdot)$ indicates the element-wise applied non-linearity.

An alternative approach to the design of the reservoir layer is provided by the recently introduced Euler State Network (EuSN) model [6]. In this case, the reservoir is derived by forward Euler discretization of a stable and non-dissipative ODE, and implements the following iterated map:

$$\mathbf{h}(t) = \mathbf{h}(t-1) + \varepsilon \tanh\big((\mathbf{W}_h - \mathbf{W}_h^T - \gamma \mathbf{I})\mathbf{h}(t-1) + \mathbf{W}_x \mathbf{x}(t) + \mathbf{b}_h\big), \qquad (2)$$

where ε and γ are small positive hyper-parameters that respectively indicate the time-step of integration and the diffusion coefficient.

The distinctive aspect of all the RC approaches is that the reservoir weights are not trained, but initialized based on considerations of the nature of the dynamic system being implemented by the reservoir layer. For ESNs, we impose asymptotic stability constraints by means of the Echo State Property condition. In practice, this corresponds to initializing the reservoir by controlling the largest length of an eigenvalue in the recurrent weight matrix $\mathbf{W}_\mathbf{h}$, i.e., its spectral radius ρ, which is typically set to a value smaller than 1, and is treated as a hyper-parameter. The weights in $\mathbf{W}_\mathbf{h}$ are randomly initialized from a uniform distribution in $(-1,1)$ and then properly re-scaled to the desired value of ρ. On the one hand, the ESN initialization provides stable dynamics in the state space, allowing the use of untrained weights in the reservoir. On the other hand, this type of initialization results in fading memory on the input signal and dissipation of state information that makes it difficult to effectively propagate information over time. In the case of EuSNs, the dynamical reservoir is constrained to operate at the edge of stability by its architectural design that leverages the use of an anti-symmetric recurrent weight matrix (i.e., $\mathbf{W}_\mathbf{h} - {\mathbf{W}_\mathbf{h}}^T$), forcing the eigenvalues of the Jacobian in the continuous-time version of eq. 2 to lie on the imaginary axis (see [6] for details). In practice, it is not necessary to control the spectral radius of $\mathbf{W}_\mathbf{h}$ in eq. 2, which is initialized randomly with values from a uniform distribution in $(-\omega_r, \omega_r)$, with ω_r playing the role of recurrent weight scaling. The EuSN reservoir avoids fading memory and facilitates information retention over longer time spans, ultimately resulting in better accuracy in time-series classification tasks compared to standard ESNs [6].

For both ESNs and EuSNs, the weights for $\mathbf{W}_\mathbf{x}$ and $\mathbf{b}_h$ are randomly initialized from a uniform distribution within the intervals $(-\omega_x, \omega_x)$ and $(-\omega_b, \omega_b)$, respectively. Here, ω_x and ω_b serve as hyper-parameters for scaling the input and bias. The output is computed by the trainable readout component, which is usually implemented by a dense linear layer trained by ridge regression. For time-series classification problems, the readout is fed by the last state computed by the reservoir across each input sequence.

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

3 Reservoir Networks with Linear Memory Cell

We introduce our novel *Reservoir Memory Network* (RMN) model. As standard RC networks, the architecture of RMN includes a fixed recurrent component, and a trainable feed-forward readout. The key distinction lies in RMN's dynamical component, which consists of a dual reservoir system: a linear reservoir *memory cell*, and a non-linear reservoir for non-linear processing over time. Conceptually, the reservoir memory cell should explicitly provide the system with a long-range temporal context on the external input driving signal, feeding the operation of the non-linear reservoir, which in turn should focus on the non-linear processing aspects required by the problem at hand. The state of the non-linear reservoir is then used as input to the readout component. The overall architecture of RMN is shown in Figure 1. Note that our design allows handling the input memorization in isolation from the non-linear processing aspects of the system, thereby avoiding a classic weakness of conventional RC models where memory and non-linear processing are tightly intertwined. Moreover, it allows one to decouple the dimensionality of the non-linear reservoir (i.e., N_h) from the size of the memory cell. Here, we use N_m to denote the number of neurons in the linear reservoir, and $\mathbf{m}(t) \in \mathbb{R}^{N_m}$ to denote the memory state at time-step t. The memory cell is updated based on the external driving input signal, as follows:

$$\mathbf{m}(t) = \mathbf{V_m}\mathbf{m}(t-1) + \mathbf{V_x}\mathbf{x}(t), \tag{3}$$

in which $\mathbf{V_m} \in \mathbb{R}^{N_m \times N_m}$ is a recurrent memory weight matrix, $\mathbf{V_x} \in \mathbb{R}^{N_m \times N_x}$ is an input memory weight matrix, and both of them are left untrained after initialization. The role of $\mathbf{V_m}$ is therefore crucial in determining the memorization abilities of the system, and we leverage two relevant design strategies for appropriately structuring the recurrent memory weights. The first entails the use of an orthogonal $\mathbf{V_m}$ matrix, exploiting the optimal short-term memory properties of this type of dynamic neural systems [7, 8]. In particular, as a specific instance of an orthogonal weight matrix with fixed weights, we use a circular shift matrix, containing 1 on the sub-diagonal and on the top-right element, and 0 elsewhere:

$$\mathbf{V_m} = \begin{bmatrix} 0 & 0 & \dots & 1 \\ 1 & 0 & \dots & 0 \\ \vdots & \ddots & \dots & \vdots \\ 0 & \dots & 1 & 0 \end{bmatrix}. \tag{4}$$

Our second approach leverages the properties of Legendre polynomials to orthogonalize the input signal across a sliding window of length θ:

$$(\bar{\mathbf{V}}_\mathbf{m})_{i,j} = \frac{(2i+1)}{\theta} \begin{cases} -1, & i < j \\ (-1)^{i-j+1}, & i \geq j \end{cases}. \tag{5}$$

In this case, the linear reservoir implements a time-invariant memory system that simulates the derivative operation in the Legendre polynomial basis [2]. To maximize the resulting memory cell capacity, we set θ equal to the maximum length of the input time-series, and define $\mathbf{V}_m = \exp(\bar{\mathbf{V}}_\mathbf{m})$, corresponding to the zero-order hold discretisation method.[1] Here, we dub the RMN model with memory cell based on Legendre polynomials as RMN$_{Le}$. The weights in $\mathbf{V_x}$ are chosen from a uniform distribution over $(-\omega_{x_m}, \omega_{x_m})$.

[1] We denotes with $\exp(\mathbf{M})$ the matrix exponential of $\mathbf{M}$.

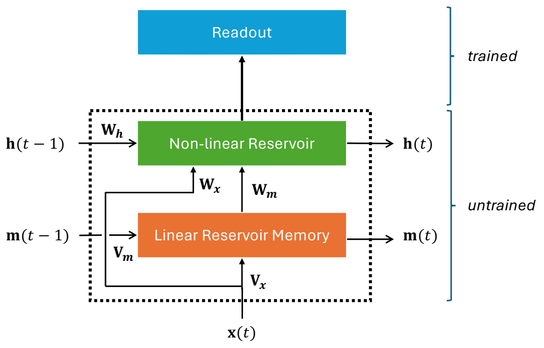

Fig. 1: Time unrolled architecture of a Reservoir Memory Network (RMN). The dynamical component includes two systems: (i) a memory cell implementing a linear reservoir driven by the external input, and (ii) a non-linear reservoir system that is fed by both the external input and the output of the memory cell. The output of the non-linear reservoir component is fed to the readout component, which is the only trained part in the architecture.

The non-linear reservoir receives in input both the external input signal $\mathbf{x}(t)$, modulated by $\mathbf{W_x}$, and the state of the memory cell $\mathbf{m}(t)$, modulated by an newly introduced reservoir memory weight matrix $\mathbf{W_m}$. The weights in $\mathbf{W_m}$ are chosen from a uniform distribution over $(-\omega_m, \omega_m)$. Notice that the non-linear reservoir component of RMN can be in principle implemented as any conventional reservoir architecture. Here, we explore the use of both ESN-like and EuSN-like non-linear reservoirs described in Section 2. In the latter case, we denote the resulting model as RMN$_{Eu}$.

Summarizing, here we consider 3 instances of the proposed approach: RMN, using a circular shift matrix in the memory cell, and an ESN-like reservoir in the non-linear recurrent part; RMN$_{Le}$, which is as RMN, but uses a Legendre polynomial-based recurrent matrix in the memory cell; RMN$_{Eu}$, which is as RMN, but uses a EuSN-like reservoir in the non-linear recurrent part.

4 Experiments

We have experimentally validated the introduced RMN approach on several time-series classification benchmarks of diverse nature, including datasets from the UCR archive (from `http://timeseriesclassification.com/`) and the sequential MNIST (sMNIST). For each dataset, we used the original partition into training and test sets. The details of the datasets used are given in Table 1. Time-series classification represents an inherently challenging class of problems, as it requires accurate retaining of input information over hundreds of steps, making it an ideal domain for evaluating the long-term dependency capabilities.

We ran experiments with RMN and its variants, using $N_h = 500$ units in the non-linear reservoir component, and exploring configurations with a number of units in the reservoir memory cell N_m equal to $1/3\ T$, $1/2\ T$, T, and $2\ T$, where T indicates the maximum length of a time-series in the training set. The other hyper-parameters were explored in the following ranges: ω_x, ω_{x_m}, ω_m, ω_b, and ω_r in $\{5, 2, 1, 0.1, 0.01\}$, ρ in $\{0.8, 0.9, 1.0\}$, α, ϵ, γ in $\{1.0, 0.1, 0.01\}$. For comparison, we ran the same experiments with ESNs and EuSNs, exploring the corresponding hyper-parameters within the same ranges indicated above for the RMN variants. As regards the non-linear reservoir dimensionality in these latter experiments, we considered two different setups. The first with $N_h = 500$,

ESANN 2024 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 9-11 October 2024, i6doc.com publ., ISBN 978-2-87587-090-2. Available from http://www.i6doc.com/en/.

Name	**# Tr**	**# Ts**	***T***	**# Feat.**	**# Classes**
Beef	30	30	470	1	5
Car	60	60	577	1	4
Coffee	28	28	286	1	2
DuckDuckGeese (DDG)	60	40	270	1345	5
FordA	3601	1320	500	1	2
FordB	3636	810	500	1	2
OSULeaf	200	242	427	1	6
Meat	60	60	448	1	3
ShapeletSim	20	180	500	1	2
ShapesAll	600	600	512	1	60
sMNIST	60000	10000	784	1	10
Symbols	25	995	398	1	6
Wine	57	54	234	1	2

Table 1: Summary of the datasets used, reporting: size of the training set (# Tr) and of the test set (# Ts), max length of a time-series (T), dimensionality of the input (# Feat.), number of target classes (# Classes).

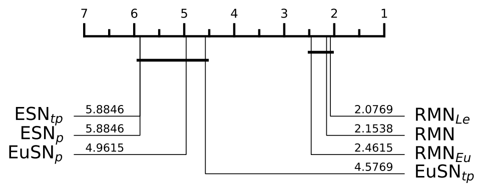

Fig. 2: Critical difference plot summarizing the aggregated scores across all datasets.

leading to a comparison under equal conditions of trainable parameters with the RMN variants. This setting is indicated with a subscript *tp*. A second experimental setup for ESNs and EuSNs involved exploring a number of reservoir units varying within a range such that the total number of internal connections is in the same range as explored for the case of experiments with RMNs. This second setting led to a comparison under equal conditions of total number of reservoir parameters with the RMN variants, and it is indicated with a subscript *p*. In all cases, the readout was applied to the reservoir state computed at the last time-step of each sequence, and was trained by ridge regression with Tikhonov regularization hyper-parameter λ, exploring values in $\{1.0, 0.1, 0.01\}$.

For each model, hyper-parameters were optimized through model selection on a validation set, derived by a further stratified splitting 33% / 67% of the training data, utilizing a random search across 200 trials (or until reaching a maximum of 10 hours of computation for model selection) and averaging results over 3 random guesses. Following model selection, the selected network configuration was trained on the entire training set and then assessed on the test set, with averages and standard deviations calculated from 10 random guesses[2].

The achieved results are given in Table 2, and demonstrate a clear benefit of using the RMN method compared to both ESN and EuSN. The enhancements in performance are uniform across nearly all datasets, with particularly notable improvements in instances such as Beef, FordA, FordB, and sMNIST. The critical difference plot, shown in Figure 2, provides a statistical comparison of the overall performance across all assessed datasets, highlighting the significant differences and rankings among the RMN models and traditional RC approaches.

[2]For sMNIST we used 1 guess for each configuration explored in the model selection phase, and 3 guesses for the final assessment of the models.

Dataset	$\mathbf{ESN_p}$	$\mathbf{ESN_{tp}}$	$\mathbf{EuSN_p}$	$\mathbf{EuSN_{tp}}$	**RMN**	$\mathbf{RMN_{Le}}$	$\mathbf{RMN_{Eu}}$
Beef	$0.55 \pm_{0.02}$	$0.53 \pm_{0.00}$	$0.55 \pm_{0.06}$	$0.42 \pm_{0.05}$	$0.82 \pm_{0.05}$	$\mathbf{0.84} \pm_{0.05}$	$0.82 \pm_{0.05}$
Car	$0.70 \pm_{0.00}$	$0.71 \pm_{0.01}$	$0.81 \pm_{0.01}$	$0.80 \pm_{0.01}$	$\mathbf{0.86} \pm_{0.02}$	$0.84 \pm_{0.03}$	$0.85 \pm_{0.04}$
Coffee	$0.98 \pm_{0.02}$	$0.93 \pm_{0.02}$	$0.91 \pm_{0.04}$	$0.93 \pm_{0.00}$	$0.99 \pm_{0.02}$	$0.95 \pm_{0.04}$	$\mathbf{1.00} \pm_{0.00}$
DDG	$0.44 \pm_{0.06}$	$0.49 \pm_{0.05}$	$0.48 \pm_{0.04}$	$0.48 \pm_{0.03}$	$0.50 \pm_{0.05}$	$0.53 \pm_{0.04}$	$\mathbf{0.56} \pm_{0.05}$
FordA	$0.71 \pm_{0.01}$	$0.70 \pm_{0.01}$	$0.72 \pm_{0.01}$	$0.69 \pm_{0.01}$	$0.88 \pm_{0.02}$	$\mathbf{0.89} \pm_{0.01}$	$0.87 \pm_{0.01}$
FordB	$0.63 \pm_{0.01}$	$0.61 \pm_{0.01}$	$0.63 \pm_{0.01}$	$0.64 \pm_{0.01}$	$\mathbf{0.74} \pm_{0.01}$	$\mathbf{0.74} \pm_{0.02}$	$0.69 \pm_{0.02}$
OSULeaf	$0.60 \pm_{0.01}$	$0.60 \pm_{0.01}$	$0.61 \pm_{0.02}$	$0.63 \pm_{0.01}$	$0.65 \pm_{0.02}$	$0.65 \pm_{0.02}$	$\mathbf{0.67} \pm_{0.02}$
Meat	$0.85 \pm_{0.00}$	$0.85 \pm_{0.01}$	$0.89 \pm_{0.01}$	$0.92 \pm_{0.02}$	$0.93 \pm_{0.01}$	$0.94 \pm_{0.04}$	$\mathbf{0.97} \pm_{0.01}$
Symbols	$0.67 \pm_{0.01}$	$0.80 \pm_{0.02}$	$0.81 \pm_{0.02}$	$0.89 \pm_{0.00}$	$0.90 \pm_{0.02}$	$\mathbf{0.92} \pm_{0.01}$	$0.89 \pm_{0.01}$
ShapeletSim	$0.49 \pm_{0.01}$	$0.52 \pm_{0.01}$	$0.48 \pm_{0.01}$	$0.51 \pm_{0.04}$	$0.59 \pm_{0.06}$	$\mathbf{0.78} \pm_{0.05}$	$0.50 \pm_{0.03}$
ShapesAll	$0.76 \pm_{0.00}$	$0.75 \pm_{0.00}$	$0.80 \pm_{0.00}$	$\mathbf{0.82} \pm_{0.01}$	$0.81 \pm_{0.01}$	$0.80 \pm_{0.01}$	$0.80 \pm_{0.01}$
sMNIST	$0.60 \pm_{0.00}$	$0.77 \pm_{0.00}$	$0.87 \pm_{0.00}$	$0.72 \pm_{0.03}$	$0.94 \pm_{0.00}$	$0.95 \pm_{0.01}$	$\mathbf{0.96} \pm_{0.00}$
Wine	$0.46 \pm_{0.00}$	$0.43 \pm_{0.02}$	$0.61 \pm_{0.07}$	$0.71 \pm_{0.03}$	$\mathbf{0.80} \pm_{0.03}$	$0.77 \pm_{0.10}$	$0.64 \pm_{0.09}$

Table 2: Test set accuracy achieved by the proposed RMN variants, compared against literature RC approaches. Best results are highlighted in bold. For the RMN models: subscript *Le* indicates a reservoir memory cell using Legendre polynomials (eq. 5); subscript *Eu* indicates that the non-linear reservoir is implemented as in EuSN (eq. 2). For the literature RC models: subscript *p* indicates comparison with the same total maximum number of parameters in the reservoir; *tp* indicates comparison with the same amount of trainable parameters.

5 Conclusions

We have introduced a novel class of Reservoir Computing systems known as Reservoir Memory Networks (RMNs), which innovatively combine a linear memory cell and a non-linear processing reservoir. This dual-reservoir architecture enhances the handling of long-term dependencies, significantly improving performance on time-series classification tasks over traditional Echo State Networks and Euler State Networks. The results presented in this paper demonstrate the great effectiveness of decoupling between a properly designed memory cell and non-linear processing layer in Recurrent Neural Networks (RNNs), independent of learning the internal connections. In the future, we plan to extend our analysis for the development of efficiently trainable structured state space model-based RNNs, and in neuromorphic hardware implementations.

References

[1] K. Nakajima and I. Fischer. *Reservoir Computing.* Springer, 2021.

[2] A. Voelker, I. Kajić, and C. Eliasmith. Legendre memory units: Continuous-time representation in recurrent neural networks. *NeurIPS*, 32, 2019.

[3] A. Orvieto, S.L. Smith, A. Gu, A. Fernando, C. Gulcehre, R. Pascanu, and S. De. Resurrecting recurrent neural networks for long sequences. In *ICML*, pages 26670–26698, 2023.

[4] H. Jaeger and H. Haas. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *science*, 2004.

[5] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert. Optimization and applications of echo state networks with leaky-integrator neurons. *Neural networks*, 20(3):335–352, 2007.

[6] C. Gallicchio. Euler state networks: Non-dissipative reservoir computing. *Neurocomputing*, page 127411, 2024.

[7] O. L. White, D. D. Lee, and H. Sompolinsky. Short-term memory in orthogonal neural networks. *Physical review letters*, 92(14):148102, 2004.

[8] P. Tino. Dynamical systems as temporal feature spaces. *Journal of Machine Learning Research*, 21(44):1–42, 2020.