Reservoir computing

Karol Bednarz

October 24, 2025

Contents

1 ESN - Echo State Network

1.1 Principles of reservoir computing

Based on the [1]: The state of reservoir dynamics can be expressed as:

$$h_{\mathrm{t}} = f\big((1-k)\cdot u_{\mathrm{t}} \cdot W_{\mathrm{in}} + k \cdot h_{\mathrm{t-1}} \cdot W_{\mathrm{h}} + y_{\mathrm{t-1}} \cdot W_{\mathrm{1b}} + b\big) \tag{1}$$

Where:

$h_{\mathrm{t-1}}$ – are the reservoir state respectively, from the previous time step,
u_{t} – is the observed data at time step t,
$y_{\mathrm{t-1}}$ – is the the predicted output state $t-1$,
$W_{\mathrm{in}} \in \mathbb{R}^{N_{\mathrm{u}} \times N_{\mathrm{h}}}$ – is the input weight matrix,
$W_{\mathrm{h}} \in \mathbb{R}^{N_{\mathrm{h}} \times N_{\mathrm{h}}}$ – is the internal weight matrix,
$W_{\mathrm{1b}} \in \mathbb{R}^{N_{\mathrm{h}} \times N_{\mathrm{y}}}$ – is the output fedback weight matrix,
$b \in \mathbb{R}^{N_{\mathrm{h}}}$ – is the bias vector.
f – is the activation function, typically tanh or sigmoid,
k – is the leaking rate, typically $k \in [0.1, 0.3]$.

With the computed reservoir dynamics, the output can be then obtained by:

$$y_{\mathrm{t}} = h_{\mathrm{t}} \cdot W_{\mathrm{out}} \tag{2}$$

Where:

$W_{\mathrm{out}} \in \mathbb{R}^{N_{\mathrm{h}} \times N_{\mathrm{y}}}$ – is the output weight matrix.

1.2 Echo state property

Any system that changes in a nonlinear way can work as the reservoir. However, starting a nonlinear system with random connection strengths creates problems. The reservoir is a system that feeds its outputs back into itself. This can make it unstable if the connection strengths aren't set up correctly. For example, if the internal connections are too strong, the system might get stuck giving the same output regardless of what input it receives. The random connection strengths must be chosen so the system doesn't grow out of control. For the system to work well, it must follow something called the "echo state property." This rule ensures that the reservoir's behavior eventually depends on the input signal rather than just its starting conditions. To meet this requirement, the internal connection matrix W_{h} is first set up using random values between -1 and 1. This matrix is then adjusted one time according to the echo state property rule:

$$W_{\mathrm{h}}' = \alpha \odot W_{\mathrm{h}} \tag{3}$$

$$W_{\mathrm{h}}^{\dagger} = \frac{\rho W_{\mathrm{h}}}{|\lambda_{\max}(W_{\mathrm{h}})|} \tag{4}$$

Where:

$\rho \in (0,1)$ – is the spectral radius, typically $\rho \in [0.9, 1]$
$\lambda_{\max}(W_{\mathrm{h}})$ – is the largest eigenvalue of W_{h}.
$\alpha \in (0,1)$ – is the sparsity coefficient, typically $\alpha \in [0.1, 0.3]$.

The spectral radius is a parameter that determines the amount of nonlinear interaction of input components through time.

Due to the recursive nature of the reservoir layer, such dynamics reflect trajectories of the past historical inputthe short-term memory (known as the fading memory). As another critical property for computing the RC principle, short-term memory can be quantitative measured by the coefficient of memory capacity

$$MC = \sum_{\mathrm{k}=1}^{\infty} MC_{\mathrm{k}} = \sum_{\mathrm{k}=1}^{\infty} d^2(u_{\mathrm{t-k}}, y_{\mathrm{t}}) = \sum_{\mathrm{k}=1}^{\infty} \frac{\mathrm{cov}^2(u_{\mathrm{t-k}}, y_{\mathrm{t}})}{\sigma^2(u_{\mathrm{t}})\sigma^2(y_{\mathrm{t}})} \tag{5}$$

Where:

$d^2(u_{\mathrm{t-k}}, y_{\mathrm{t}})$ – is the square of the correlation coefficient between the output y_{t} and the input $u_{\mathrm{t-k}}$ with a delay of k time steps,

According to the Lyapunov stability analysis, a large memory capacity is needed to compute the RC principle, which can be achieved at the asymptotically stable region.

1.3 Learning algorithm

The training of the reservoir computing model involves adjusting only the output weights W_{out}. The input weights W_{in}, internal weights W_{h}, and feedback weights W_{1b} are typically initialized randomly and remain fixed during training. The training process can be summarized in the following steps:

$$Y = H \cdot W_{\mathrm{out}} \tag{6}$$

Where:

$Y \in \mathbb{R}^{T \times N_{\mathrm{y}}}$ – is the matrix of target outputs for all time steps,
$H \in \mathbb{R}^{T \times N_{\mathrm{h}}}$ – is the matrix of reservoir states for all time steps,

T – is the total number of time steps.

In general, the W_{out} can be directly obtained by calculating the Moore-Penrose pseudoinverse of the reservoir states matrix H with respect to the target outputs matrix Y:

$$W_{\text{out}} = Y \cdot H^T \cdot (H \cdot H^T + \eta I)^{-1} \tag{7}$$

Where:

H^T – is the transpose of matrix H,
η – is the regularization parameter, typically $\eta \in [10^{-6}, 10^{-2}]$,
I – is the identity matrix of size $N_{\text{h}} \times N_{\text{h}}$.

1.4 Model development

The echo state network (ESN) and the liquid state machine (LSM) are the two representations of RC. While ESN and LSM are topographically equivalent, the former adopts the actual numerical data from the input to compute the network dynamics, and the latter adopts the spiking signal to represent the spatiotemporal pattern. **The LSM can be also seen as a SNN, where the reservoir has numerous leaky integrate-and-fire (LIF) neurons interconnected with the same recursive nature as in ESN**. As the spiking event is involved, the training of LSM generally relies on the spike-timing-dependent plasticity (STDP) and short-term plasticity (STP), among other training methods for spiking networks.

2 Time-delayed reservoir computing

Based on [4, 7, 6].

The model of reservoir can be written in the following general form:

$$\mathbf{r}(n) = F\big(\gamma \mathbf{W}_{\text{in}}\mathbf{x}(n) + \beta \mathbf{W}\mathbf{r}(n-1)\big), \tag{8}$$
$$\mathbf{y}(n) = \mathbf{W}_{\text{out}}\mathbf{r}(n). \tag{9}$$

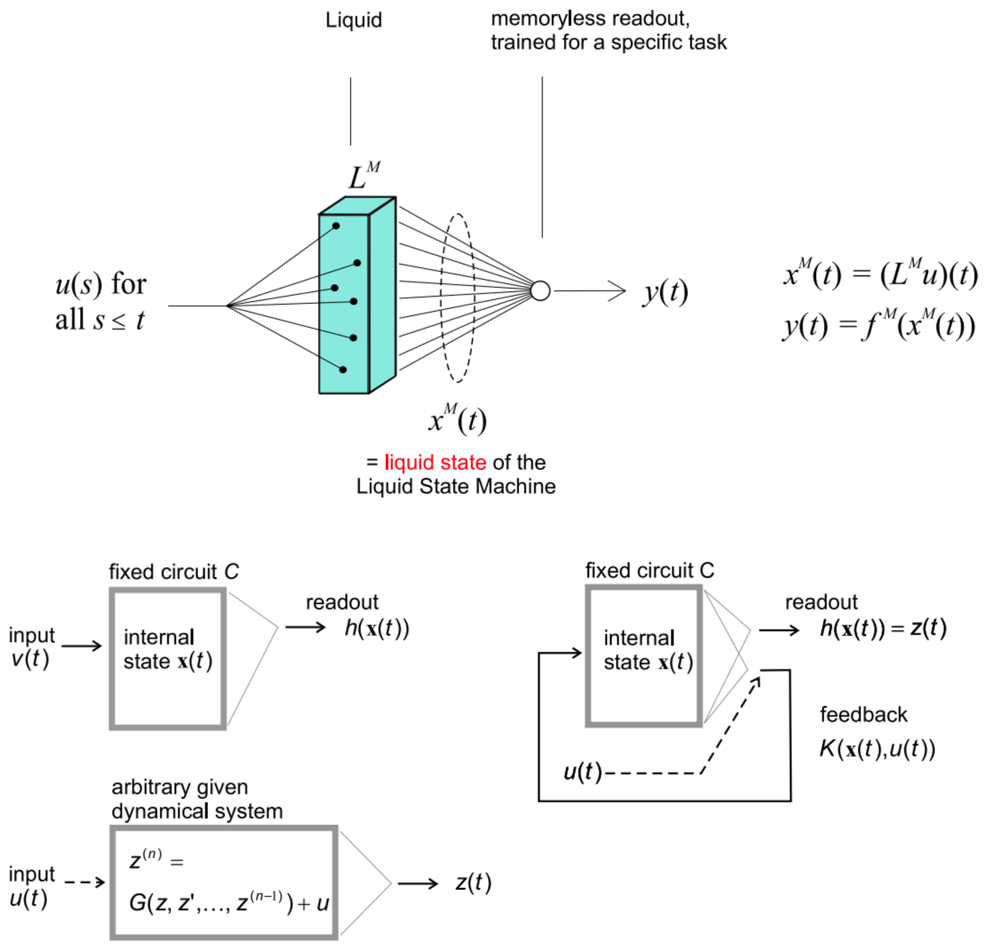
Where:

$\mathbf{r}(n) \in \mathbb{R}^N$ – is the reservoir state at time step n,
$\mathbf{x}(n) \in \mathbb{R}^d$ – is the input vector at time step n,
$\mathbf{y}(n) \in \mathbb{R}^p$ – is the output vector at time step n,
$\mathbf{W}_{\text{in}} \in \mathbb{R}^{N\times d}$ – is the input weight matrix, usually drawn from a random distribution $[-1, 1]$,
$\mathbf{W} \in \mathbb{R}^{N\times N}$ – is the internal weight matrix, usually drawn from a random distribution $[-1, 1]$,
$\mathbf{W}_{\text{out}} \in \mathbb{R}^{p\times N}$ – is the output weight matrix,
$\gamma, \beta \in \mathbb{R}^+$ – are the input and feedback scaling parameters,
$F : \mathbb{R}^N \to \mathbb{R}^N$ – is the nonlinear activation function, typically tanh or sigmoid.

In general, the time-delayed reservoir computing (TDRC) can be implemented using a single nonlinear node with delayed feedback loop instead of a large network of interconnected nodes. The delay dynamics create a high-dimensional state space that can be exploited for computation.

3 Liquid State Machine – Reservoir with spiking neurons

Based on [5, 2].



Liquid
memoryless readout, trained for a specific task
L^M
$u(s)$ for all $s \le t$
$y(t)$
$x^M(t) = (L^M u)(t)$
$y(t) = f^M(x^M(t))$
$x^M(t)$
= liquid state of the Liquid State Machine
fixed circuit C
input $v(t)$
internal state $\mathbf{x}(t)$
readout $h(\mathbf{x}(t))$
fixed circuit C
internal state $\mathbf{x}(t)$
readout $h(\mathbf{x}(t)) = z(t)$
feedback $K(\mathbf{x}(t), u(t))$
$u(t)$
arbitrary given dynamical system
input $u(t)$
$z^{(n)} = G(z, z', \ldots, z^{(n-1)}) + u$
$z(t)$

Figure 1: Liquid State Machine architecture

4 Next generation reservoir computing

Based on [3].

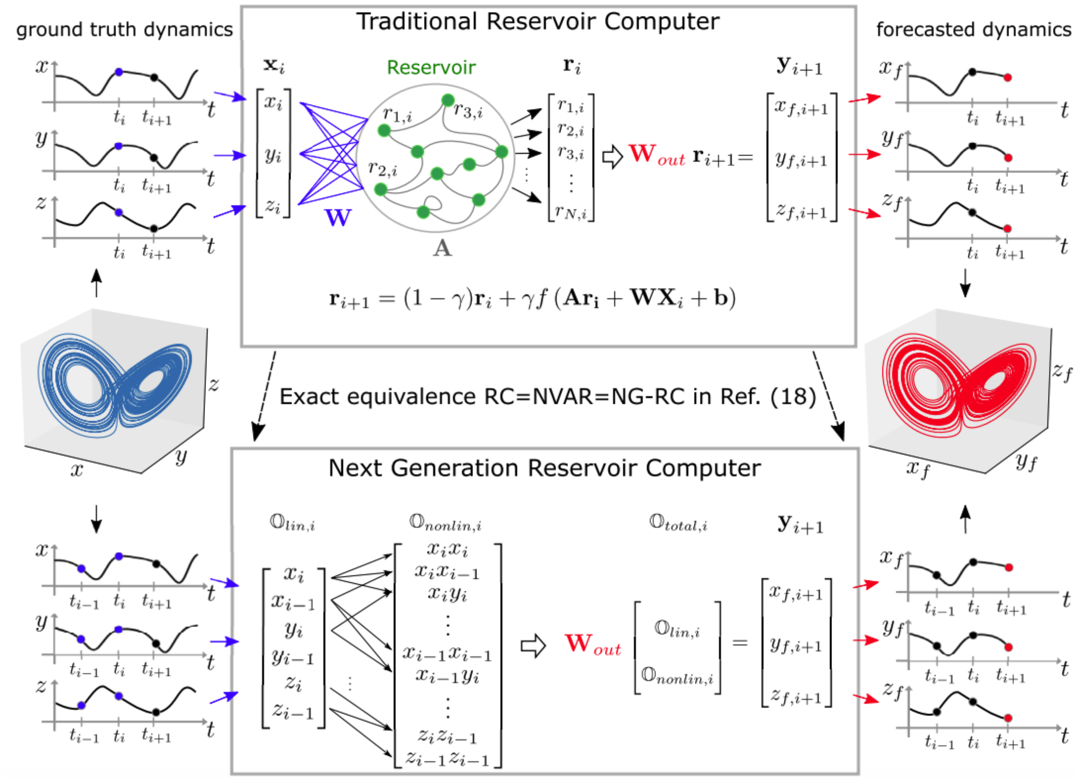

ground truth dynamics
Traditional Reservoir Computer
forecasted dynamics
Reservoir
$\mathbf{W}$
$\mathbf{A}$
$\mathbf{W}_{out}\,\mathbf{r}_{i+1} =$
$\mathbf{r}_{i+1} = (1-\gamma)\mathbf{r}_i + \gamma f(\mathbf{A}\mathbf{r}_i + \mathbf{W}\mathbf{X}_i + \mathbf{b})$
Exact equivalence RC=NVAR=NG-RC in Ref. (18)
Next Generation Reservoir Computer
$\mathbb{O}_{lin,i}$
$\mathbb{O}_{nonlin,i}$
$\mathbb{O}_{total,i}$
$\mathbf{W}_{out}$

Figure 2: A traditional RC is implicit in an NG-RC. (top) A traditional RC processes time-series data associated with a strange attractor (blue, middle left) using an artificial recurrent neural network. The forecasted strange attractor (red, middle right) is a linear weight of the reservoir states. (bottom) The NGRC performs a forecast using a linear weight of time-delay states (two times shown here) of the time series data and nonlinear functionals of this data (quadratic functional shown here).

The reservoir is a dynamic system whose dynamics can be represented by:

$$\mathbf{r}_{\mathrm{i}+1} = (1-\gamma)\mathbf{r}_{\mathrm{i}} + \gamma f\big(\mathbf{A}r_{\mathrm{i}} + \mathbf{W}_{\mathrm{in}}\mathbf{X}_{\mathrm{i}} + b\big) \tag{10}$$

Where:
$r_{\mathrm{i}} = [r_{\mathrm{i}}^1, r_{\mathrm{i}}^2, \ldots, r_{\mathrm{i}}^N]^T \in \mathbb{R}^{N\times 1}$ is the reservoir state at time step i,
The output layer expresses the RC output $\mathbf{Y}_{\mathrm{i}+1}$ as a linear transformation of a feature vector $\mathbf{O}_{\mathrm{total,i+1}}$, constructed from the reservoir state $\mathbf{r}_{\mathrm{i}+1}$, through the relation:

$$\mathbf{Y}_{\mathrm{i}+1} = \mathbf{W}_{\mathrm{out}}\mathbf{O}_{\mathrm{total,i+1}} \tag{11}$$

Where:

$$\begin{aligned} \mathbf{O}_{\mathrm{total,i+1}} \in \mathbb{R}^{M\times 1} &\ \text{ – is the feature vector at time step } i+1, \\ \mathbf{W}_{\mathrm{out}} \in \mathbb{R}^{p\times M} &\ \text{ – is the output weight matrix,} \\ M &\ \text{ – is the size of the feature vector.} \end{aligned}$$

The RC is trained using supervised training via regularized least-squares regression. Here, the training data points generate a block of data contained in $\mathbf{O}_{\mathrm{total}}$ and we match $\mathbf{Y}$ to the desired output $\mathbf{Y}_{\mathrm{d}}$ in a least-square sense using Tikhonov regularization so that $\mathbf{W}_{\mathrm{out}}$ is given by

$$\mathbf{W}_{\text{out}} = \mathbf{Y}_{\text{d}}\mathbf{O}_{\text{total}}^{T}(\mathbf{O}_{\text{total}}\mathbf{O}_{\text{total}}^{T} + \alpha\mathbf{I})^{-1} \tag{12}$$

5 Sample implementation in Python

References

[1] Kang Jun Bai et al. “Design Strategies and Applications of Reservoir Computing: Recent Trends and Prospects [Feature]”. In: *IEEE Circuits and Systems Magazine* 23.4 (2023), pp. 10–33. ISSN: 1558-0830. DOI: 10.1109/mcas.2023.3325496. URL: http://dx.doi.org/10.1109/MCAS.2023.3325496.

[2] Lucas Deckers et al. “Extended liquid state machines for speech recognition”. In: *Frontiers in Neuroscience* Volume 16 - 2022 (2022). ISSN: 1662-453X. DOI: 10.3389/fnins.2022.1023470. URL: https://www.frontiersin.org/journals/neuroscience/articles/10.3389/fnins.2022.1023470.

[3] Daniel J. Gauthier et al. “Next generation reservoir computing”. In: *Nature Communications* 12.1 (Sept. 2021). ISSN: 2041-1723. DOI: 10.1038/s41467-021-25801-2. URL: http://dx.doi.org/10.1038/s41467-021-25801-2.

[4] Lyudmila Grigoryeva et al. “Time-Delay Reservoir Computers and High-Speed Information Processing Capacity”. In: *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*. IEEE, Aug. 2016, pp. 492–495. DOI: 10.1109/cse-euc-dcabes.2016.230. URL: http://dx.doi.org/10.1109/CSE-EUC-DCABES.2016.230.

[5] Wolfgang Maass. “Liquid State Machines: Motivation, Theory, and Applications”. In: *Computability in Context*. IMPERIAL COLLEGE PRESS, Feb. 2011, pp. 275–296. ISBN: 9781848162778. DOI: 10.1142/9781848162778_0008. URL: http://dx.doi.org/10.1142/9781848162778_0008.

[6] S. Ortín et al. “A Unified Framework for Reservoir Computing and Extreme Learning Machines based on a Single Time-delayed Neuron”. In: *Scientific Reports* 5.1 (Oct. 2015). ISSN: 2045-2322. DOI: 10.1038/srep14945. URL: http://dx.doi.org/10.1038/srep14945.

[7] Ulrich Parlitz. “Learning from the past: reservoir computing using delayed variables”. In: *Frontiers in Applied Mathematics and Statistics* 10 (Mar. 2024). ISSN: 2297-4687. DOI: 10.3389/fams.2024.1221051. URL: http://dx.doi.org/10.3389/fams.2024.1221051.