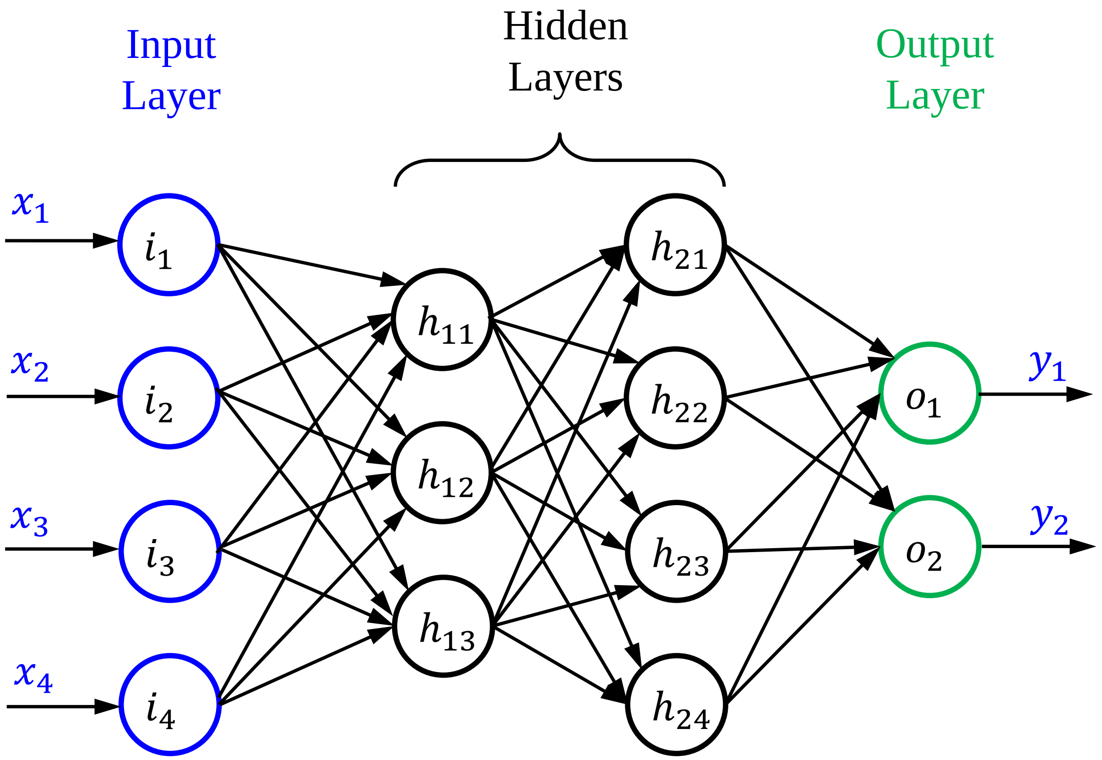
Module 5
Reservoir Computing Architecture

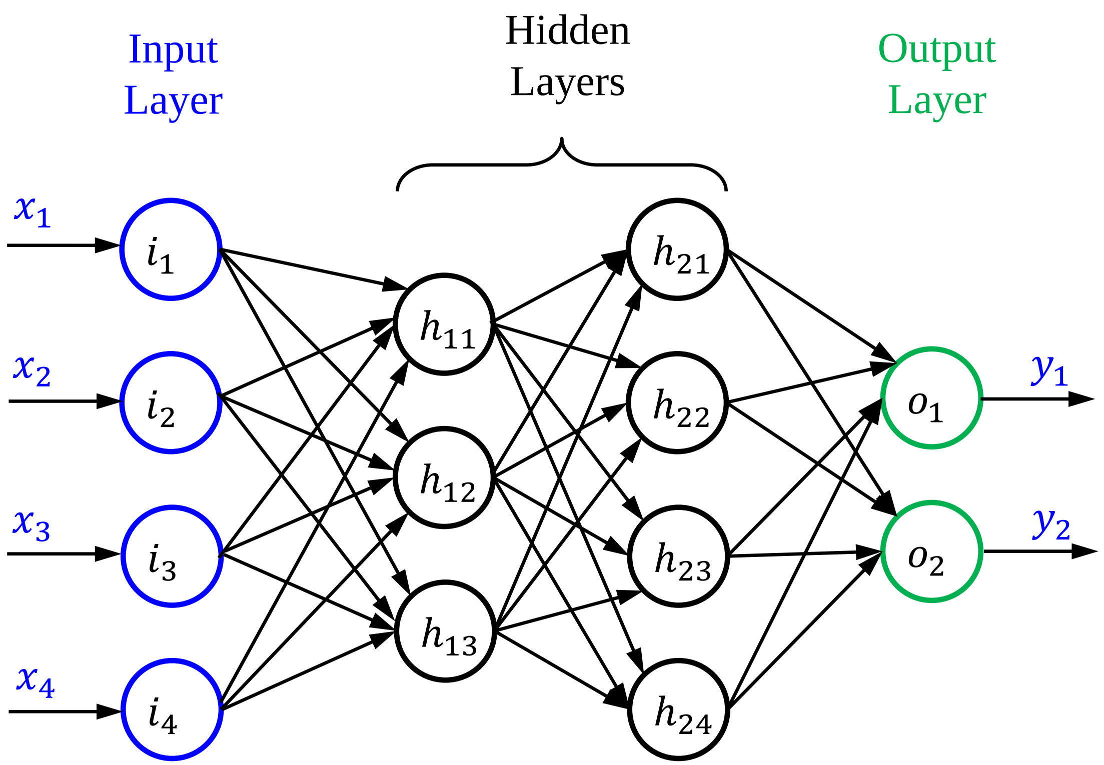
Santa Clara University

Feedforward Neural Networks (FNNs)



- It underlines the flow direction of information between its layers.
- The information flows in one direction from the input layer, through hidden layers, and to the output layer.
- FFNs can easily be trained with the backpropagation technique.
- FNNs can process well spatial information but not temporal information.

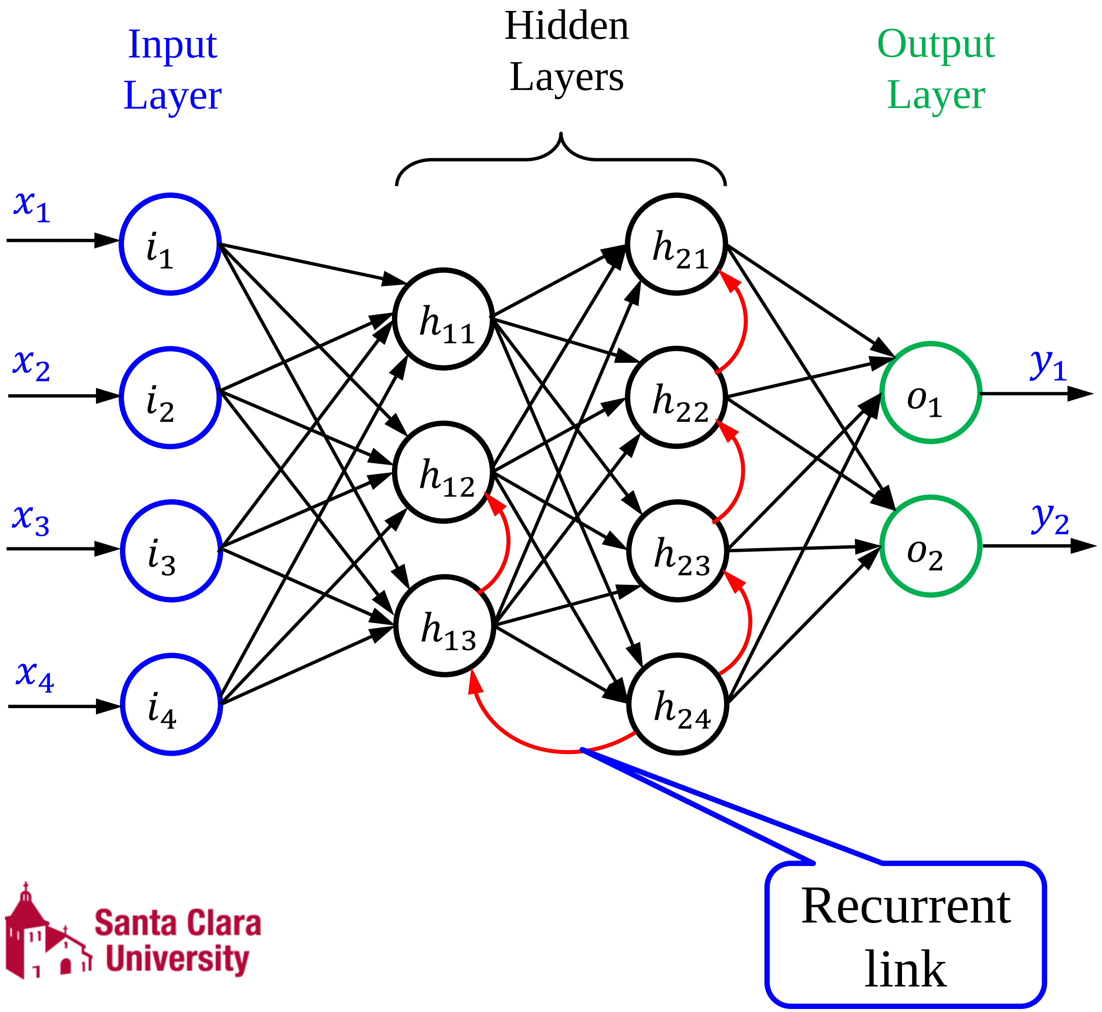
Recurrent Neural Networks (RNNs)



- It underlines the flow direction of information between its layers.
- The information flows in one direction from the input layer, through hidden layers, and to the output layer.
- FNNs can easily be trained with the backpropagation technique.
- FFNs can process well spatial information but not temporal information.

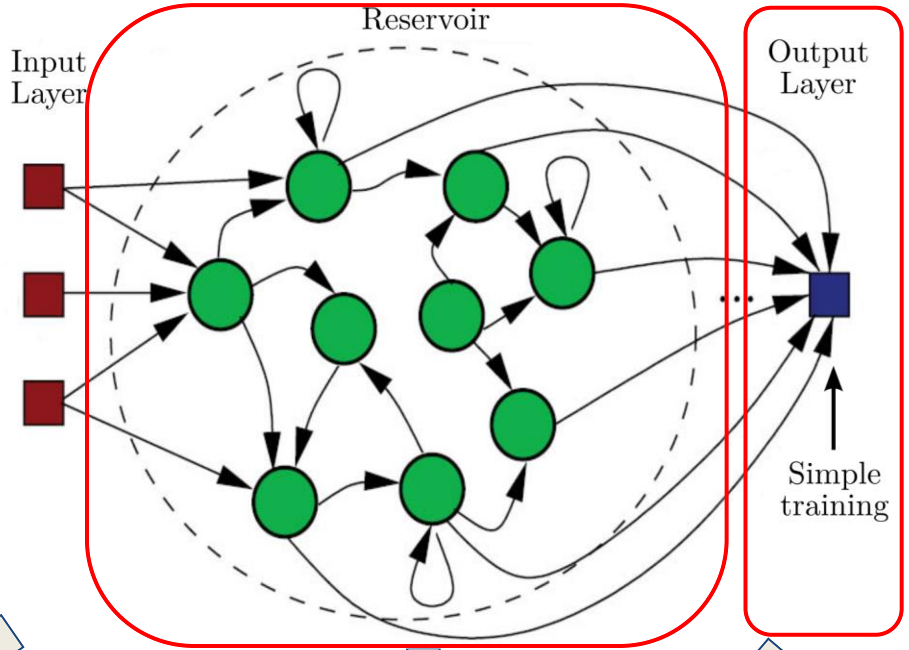
Santa Clara University

Recurrent Neural Networks



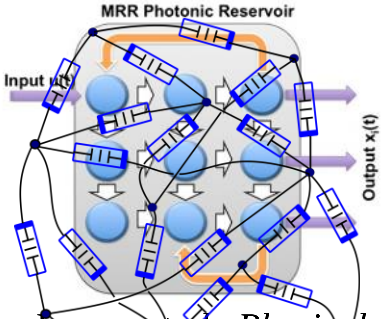
- Recurrent Neural Networks (RNNs) are FNNs with recurrent or feedback connections.
- RNNs overcome the issue of processing temporal information but suffer the complexity in training.
- One of many complex training algorithms for training RNNs is backpropagation through time (BPTT).

Reservoir Computing Architectures (1)



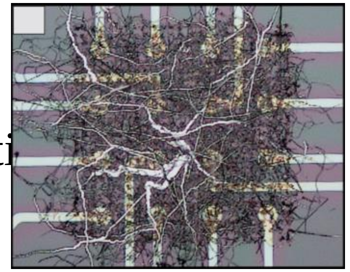
6 weights

Fernando *et al.*, *European Conference on Artificial Life*, 2003.



Larger *et al.*, *Physical Review X* 7.1, 2017

Memcapaciti network

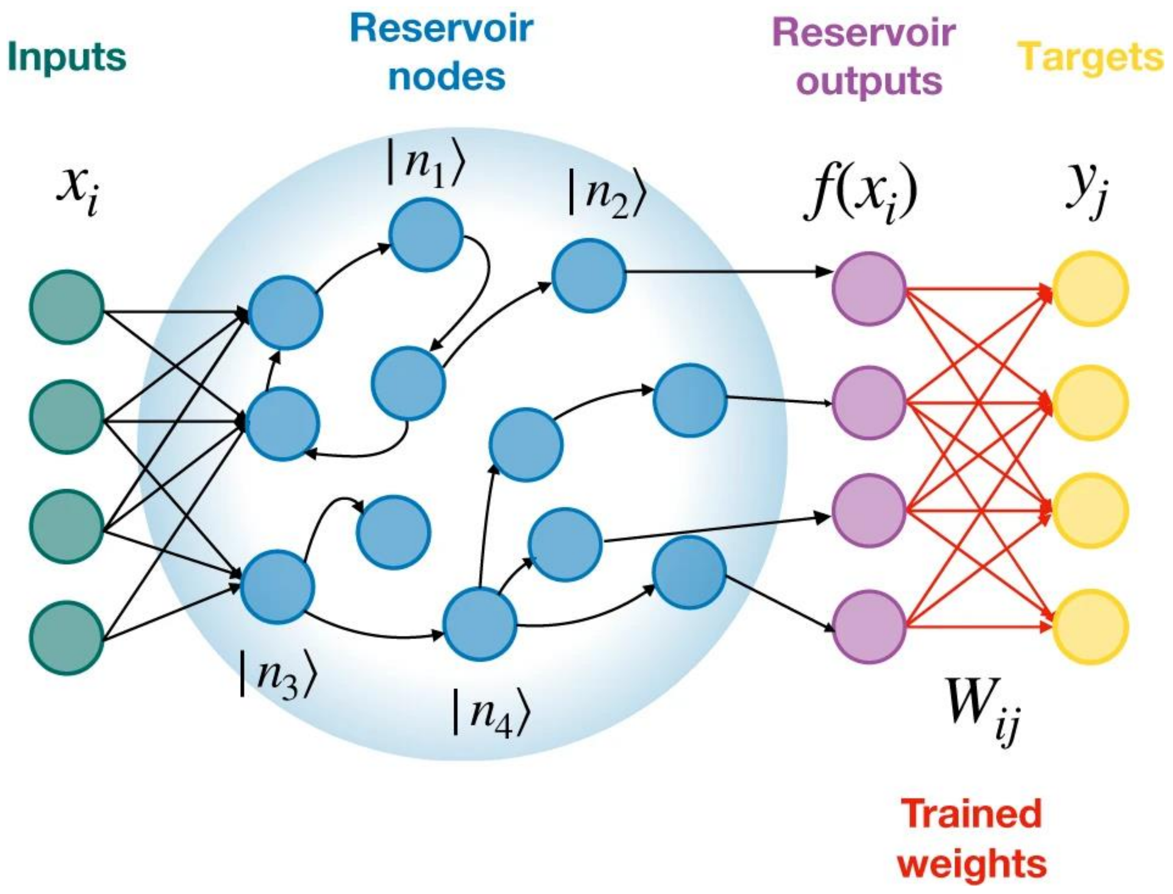


Demis *et al*, *Nanotechnology*, 2015

Santa Clara University

Reservoir Computing Architecture (2)

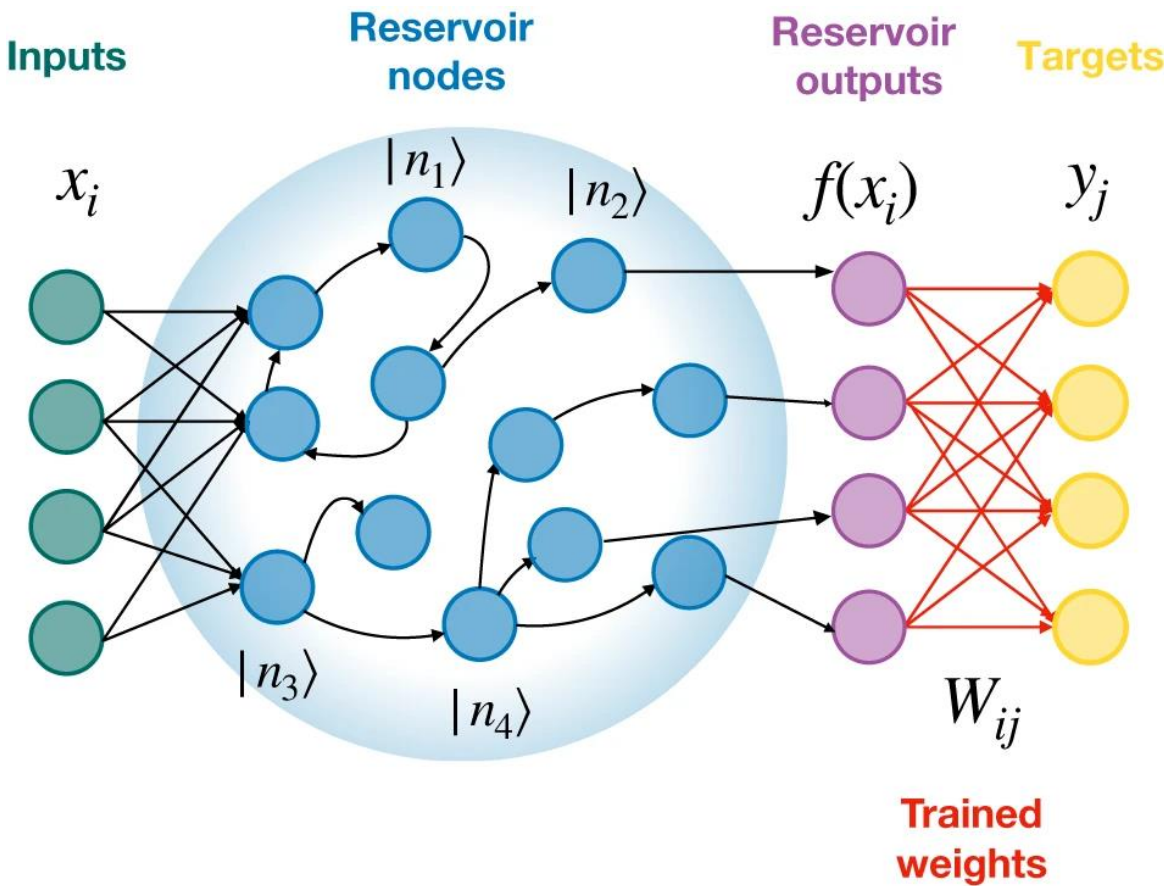
- Reservoir computing is an alternative to RNNs.
- It has a reservoir (fixed and non-linear system) that maps input signals into higher dimensional computational spaces
- An output layer extracts information from reservoir states and is trained with a simple technique.



https://www.nature.com/articles/s41534-023-00734-4

Reservoir Computing Architecture (3)

- In 2004, Jaeger and Haas proved that a nonlinear reservoir could characterize the input signal [1].
- Their work was based on two reservoir models developed earlier by Jaeger and Maass: Echo State Network [2] and Liquid State Machine [3]



https://www.nature.com/articles/s41534-023-00734-4

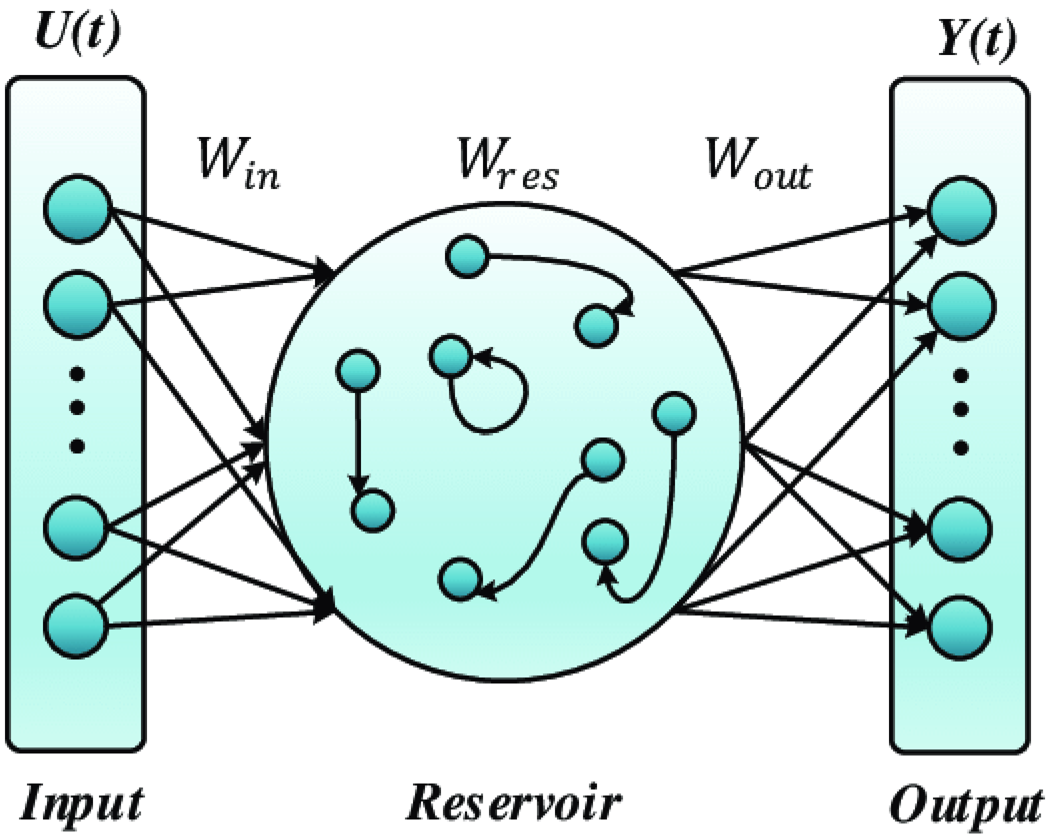
[1] https://www.science.org/doi/full/10.1126/science.1091277
[2] https://www.ai.rug.nl/minds/uploads/EchoStatesTechRep.pdf
[3] https://ieeexplore.ieee.org/abstract/document/6789852

Echo State Network (1)

- Echo State Network (ESN) is a software network: input layer $u(t)$, an RNN as a reservoir, and output layer $y(t)$.
- The input and reservoir weights (W_{in} and W_{res}) are fixed, the output weight W_{out} is trainable with a linear regression technique.
- The state transition equation with an activation function $\varphi()$ is

$$x(t) = \varphi[W_{in}u(t) + W_{res}x(t-1)]$$

$$y(t) = W_{out}x(t)$$

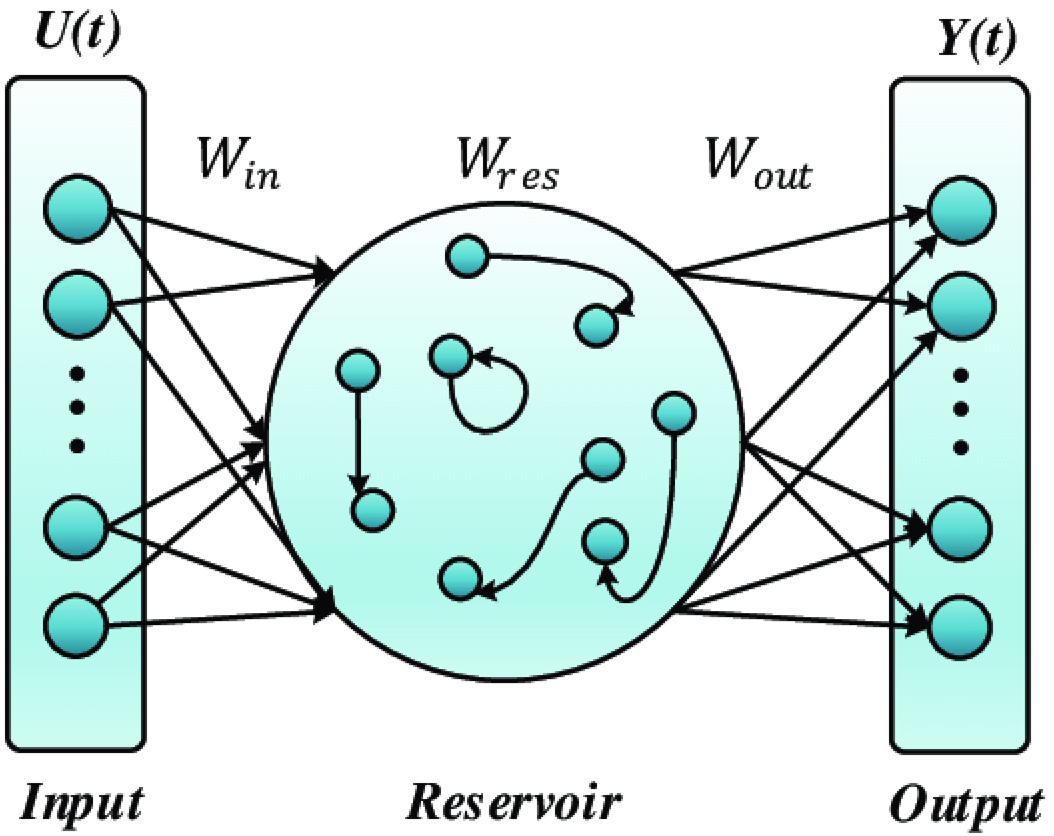


https://ieeexplore.ieee.org/abstract/document/9093897

Echo State Network (2)

- ESN has three hyperparameters that need to be initialized:
- w^{in} is an input-scaling parameter that sets W_{in} to an uniform distribution in $[-w^{in}, w^{in}]$.
- α is a sparsity parameter of W_{res}
- $\rho(W_{res})$ is the spectral radius parameter (the largest eigenvalue) of W_{res} initialized with W in $[-1, 1]$ and the largest eigenvalue $\lambda_{max}(W)$:

$$W_{res} = \rho(W_{res}) * \frac{W}{\lambda_{max}(W)}$$



https://ieeexplore.ieee.org/abstract/document/9093897

Echo State Network (3)

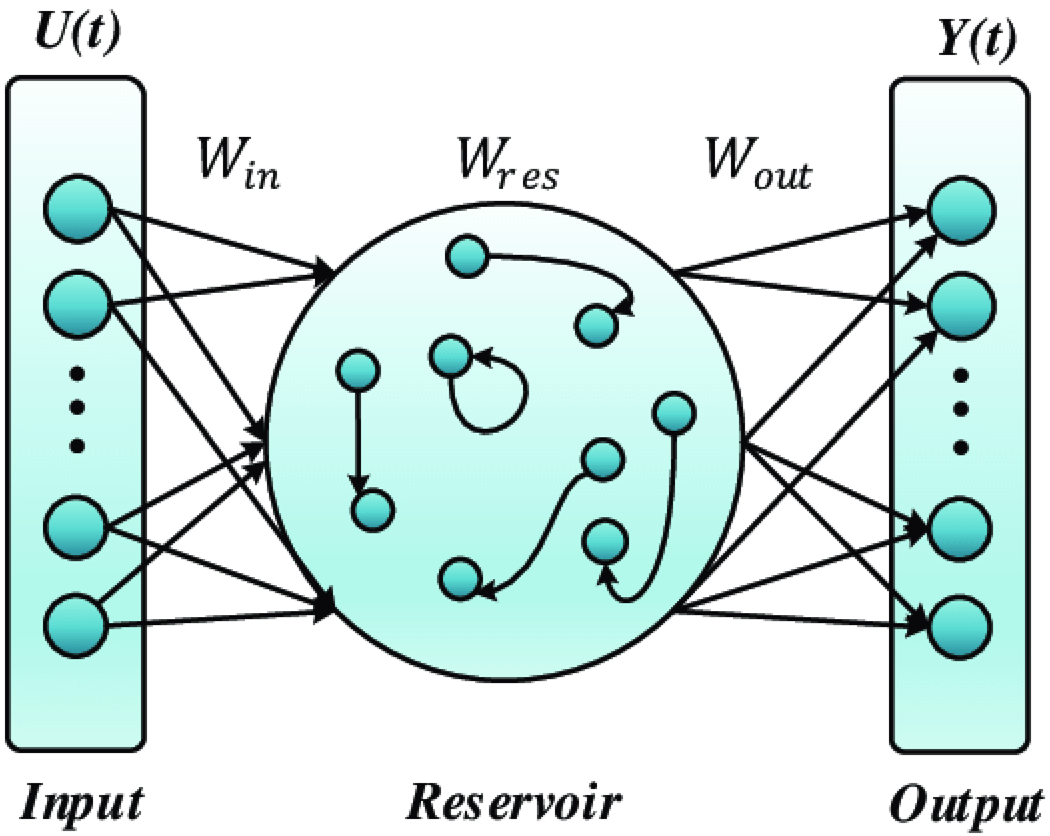
- The output layer is trained with a ridge regression method from the signal $X(t)$ of the reservoir and desired target $Y(t)$:

$$min_{W_{out}} |W_{out} * X(t) - Y(t)|^2$$

- The weight update for the output layer is:

$$W_{out} = Y(t) * X(t)^{-1}$$

- Echo state properties:
  - Spectral radius: $\rho(W_{res}) < 1$
  - Memory capacity is bounded by the size N of the reservoir.



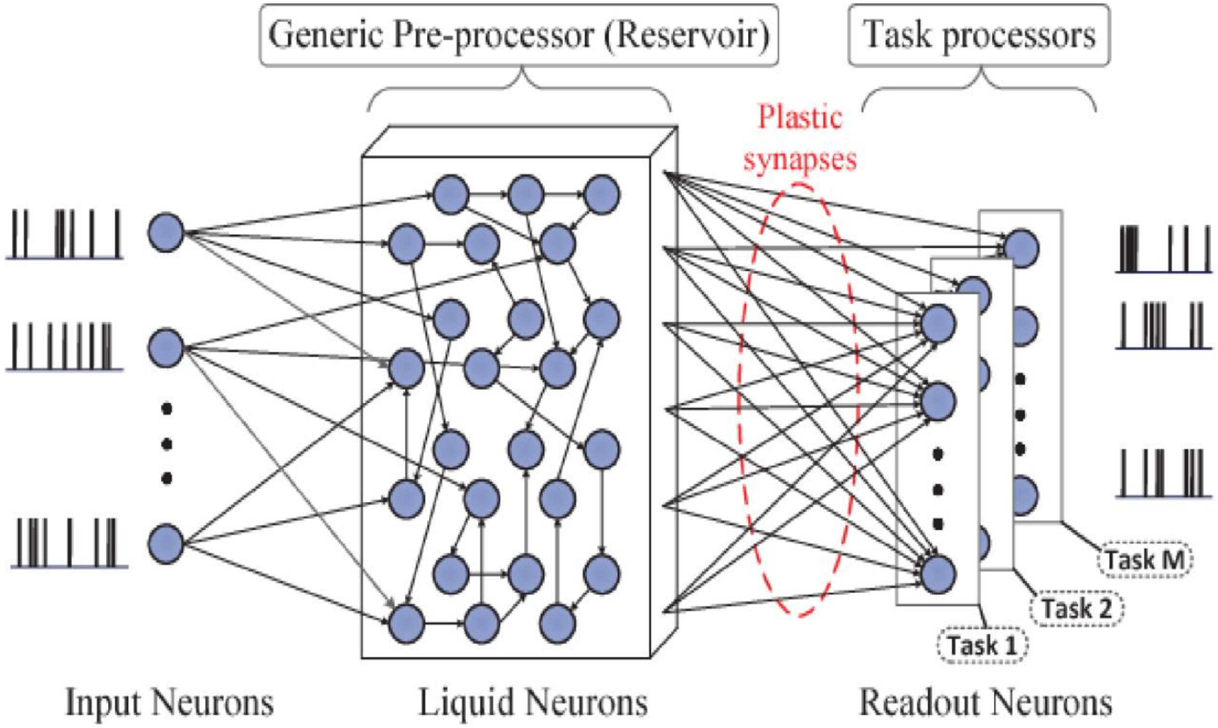
https://ieeexplore.ieee.org/abstract/document/9093897

Santa Clara University

https://www.geeksforgeeks.org/echo-state-network-an-overview/

Liquid State Machine (1)

- Similar to ESN, a liquid-state machine has 3 layers:
  - An input layer
  - A reservoir (or liquid layer) is composed of neurons interconnected recurrently with biologically realistic parameters using dynamic synaptic connections.
  - A memoryless readout circuit



https://arxiv.org/ftp/arxiv/papers/1910/1910.03354.pdf

Santa Clara University

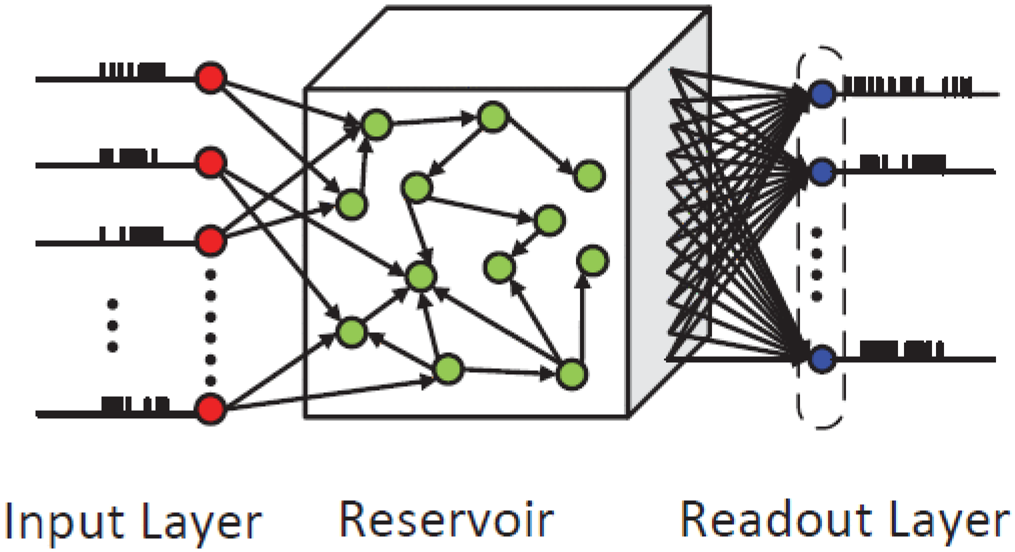
Liquid State Machine (2)

- Since the reservoir is spiking neurons, it is required to update both pre- and post-synapses using the spike-time-dependent plasticity (STDP) rule.
- The reservoir translates spiking train signal $u(t)$ into its high-dimensional state. The output of a readout neuron i at time t from a reservoir neuron k with a response $f(t)$ is:

$$o_i(t) = \sum W_{oj} * f[u(t)]$$

$$\int_0^T o_i(t) = \sum W_{oj} * \int_0^T f[u(t)]$$

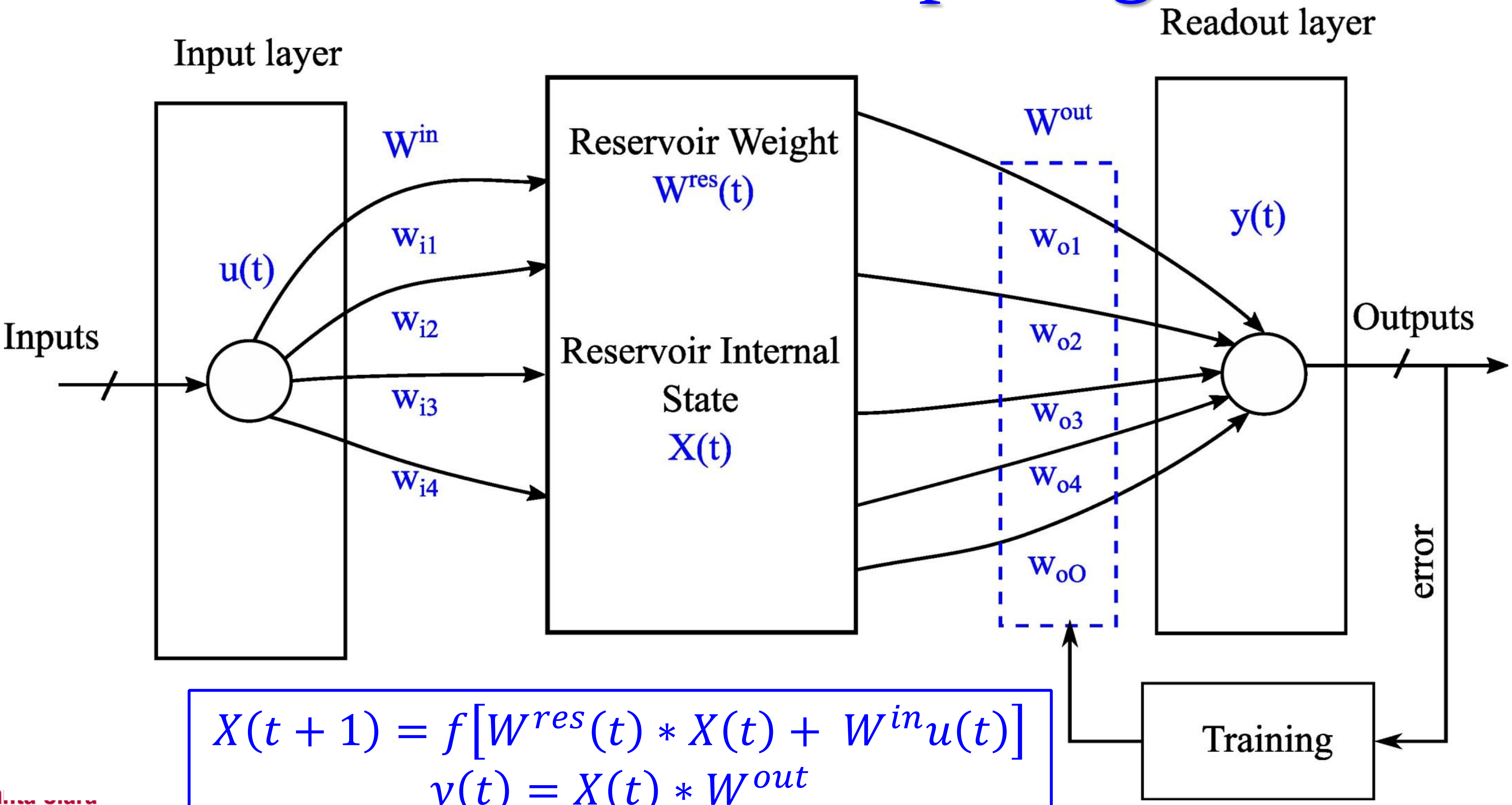
https://arxiv.org/ftp/arxiv/papers/1910/1910.03354.pdf



https://ieeexplore.ieee.org/abstract/document/7966097

Santa Clara University

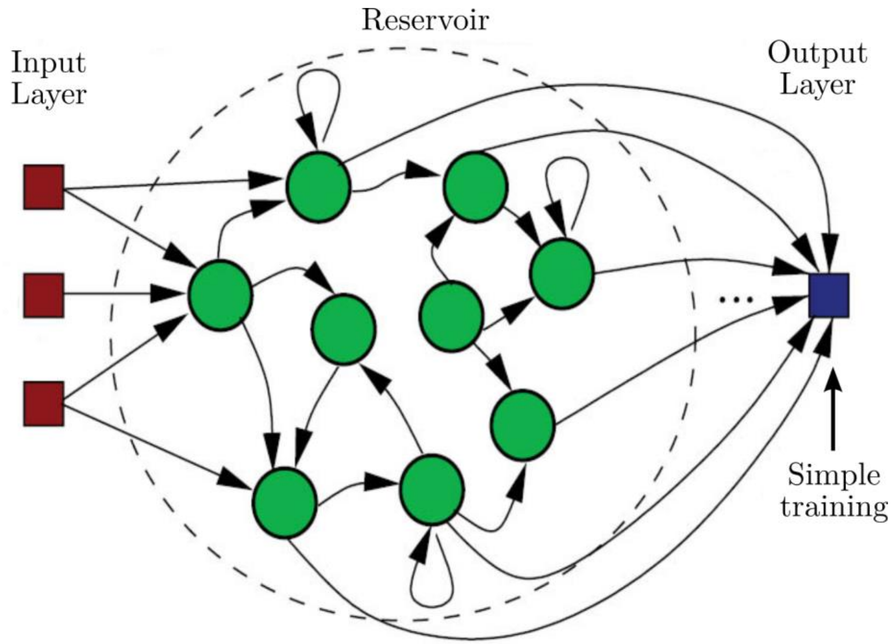
Reservoir Computing

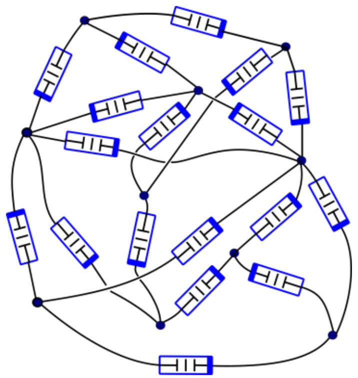


$$X(t+1) = f\left[W^{res}(t) * X(t) + W^{in}u(t)\right]$$
$$y(t) = X(t) * W^{out}$$

Santa Clara University

Reservoir Computing Architectures





Memcapacitive
network

Santa Clara
University

Memristive Electrical Node (1)

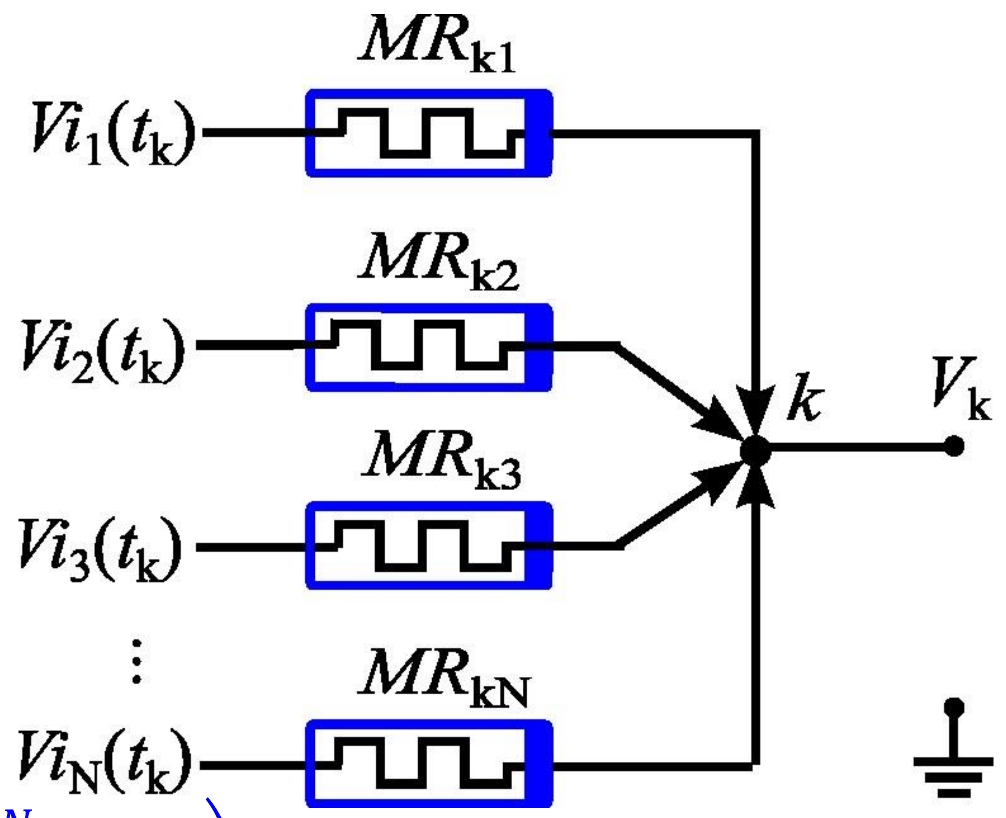
- According to Kirchhoff's Current Law (KCL), the sum of current at node k is:

$$i_{MR_{k1}} + i_{MR_{k2}} + \cdots + i_{MR_{kN}} = 0$$

$$\frac{V_{i1} - V_k}{MR_{k1}} + \frac{V_{i2} - V_k}{MR_{k2}} + \cdots + \frac{V_{iN} - V_k}{MR_{kN}} = 0$$

$$\frac{V_{i1}}{MR_{k1}} + \frac{V_{i2}}{MR_{k2}} + \cdots + \frac{V_{iN}}{MR_{kN}} = \left(\sum_{l=1}^{N} \frac{1}{MR_{kl}}\right) V_k$$

- If we denote $MS_{k1} (MS_{k1} = 1/MR_{k1})$ as the conductance of MR_{k1}, we have:



$$V_{i1}MS_{k1} + V_{i2}MS_{k2} + \cdots + V_{iN}MS_{kN} = \left(\sum_{l=1}^{N} MS_{kl}\right) V_k$$

Santa Clara University

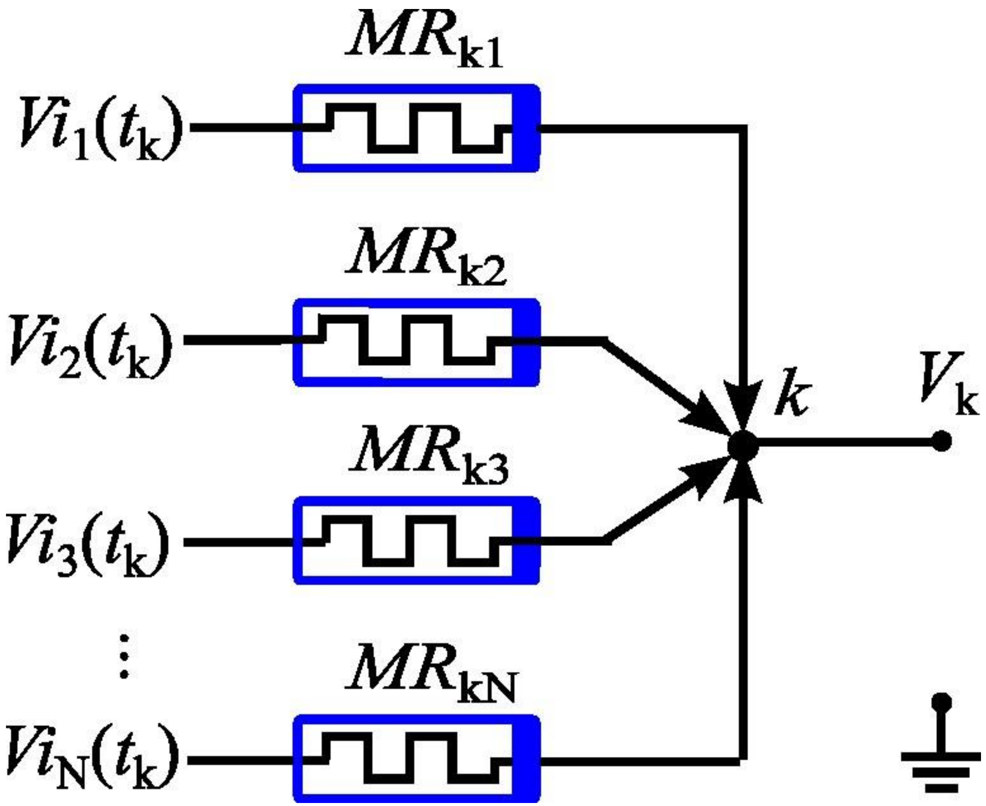
Memristive Electrical Node (2)

- We can rearrange the equation as:

$$V_k = \left(\frac{1}{\sum_{l=1}^{N} MS_{kl}}\right) \sum_{n=1}^{N} MS_{kn} V_{in}$$

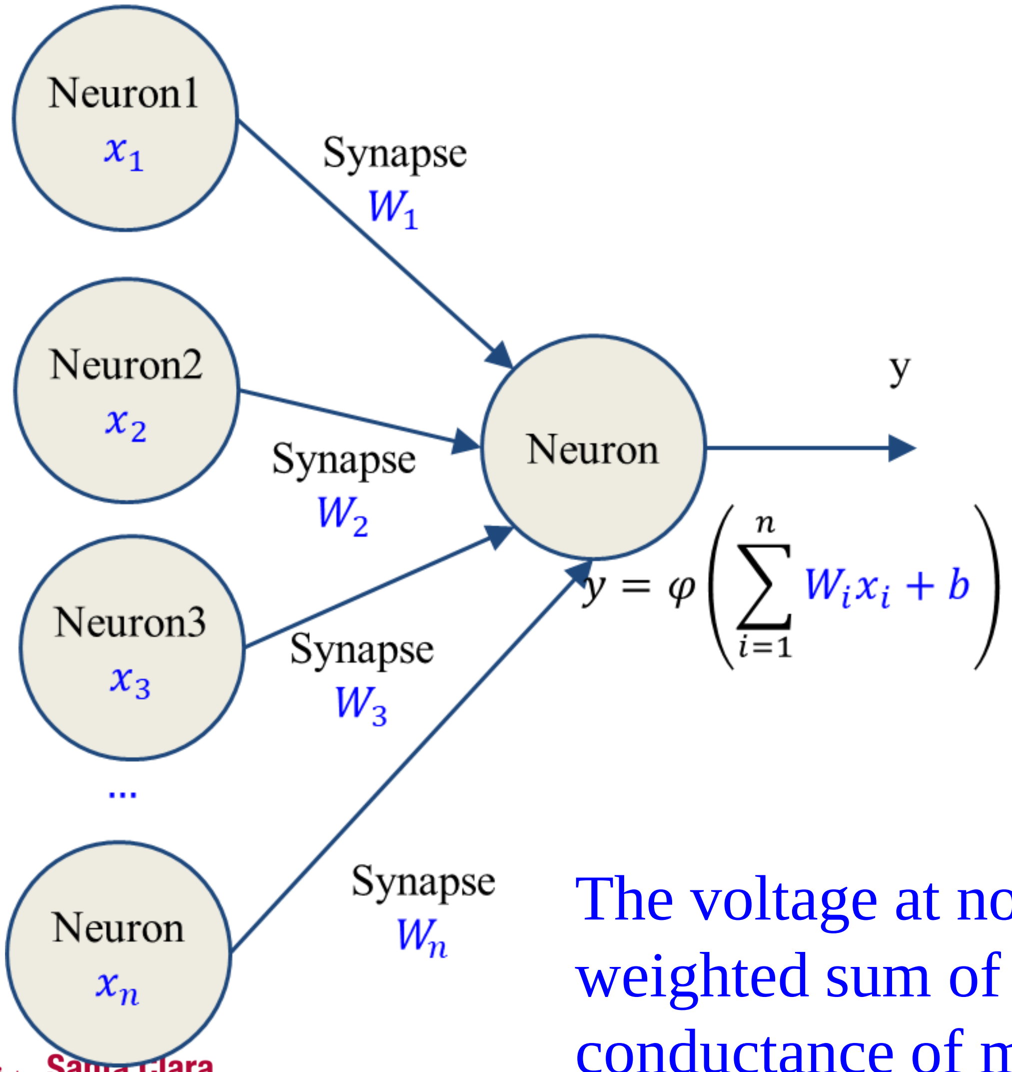
$$V_k = \varphi\left(\sum_{n=1}^{N} MS_{kn} V_{in}\right)$$

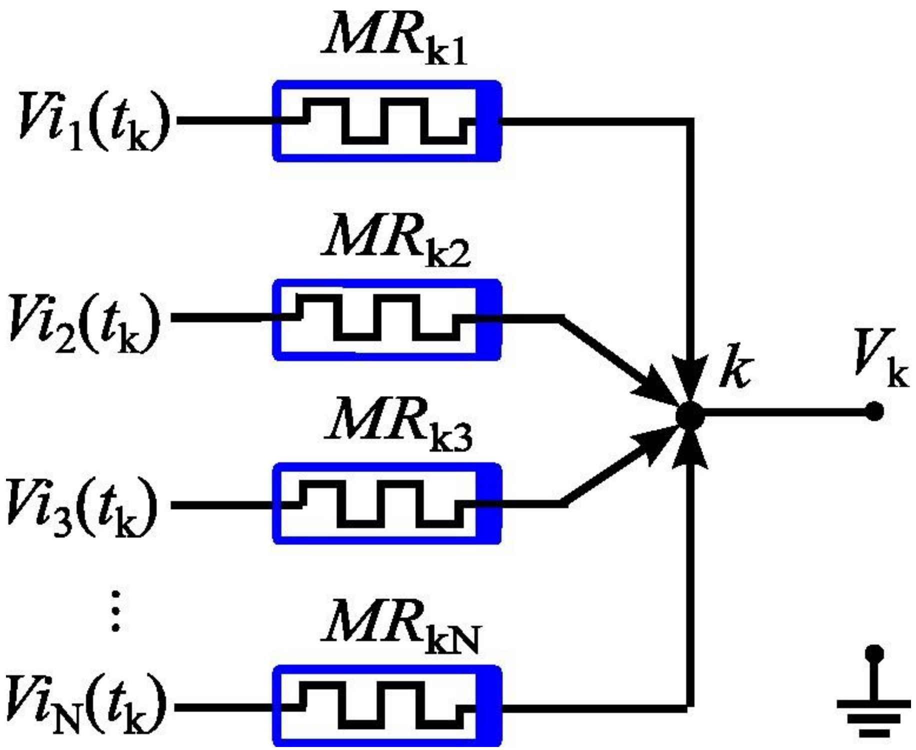
$$\varphi(x) = \frac{x}{\sum_{l=1}^{N} MS_{kl}}$$



Santa Clara University

Memristive Electrical Node (3)





The voltage at node k is the weighted sum of input voltages and conductance of memristive synapses.

$$V_k = \varphi\left(\sum_{n=1}^{N} MS_{kn} V_{in}\right)$$

Santa Clara University

Memcapacitive Electrical Node (1)

- According to Kirchhoff's Current Law (KCL), the sum of current at node k is zero. Therefore, the sum of charge at node k is also zero :

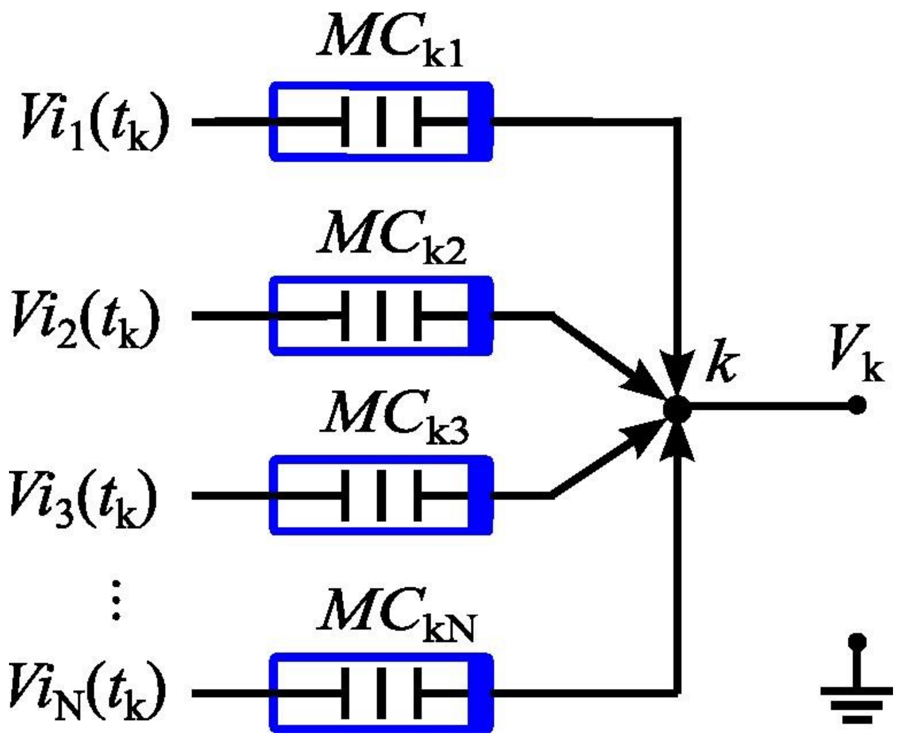
$$q_{MC_{k1}} + q_{MC_{k2}} + \cdots + q_{MC_{kN}} = 0$$

$$(V_{i1} - V_k)MC_{k1} + (V_{i1} - V_k)MC_{k2} + \cdots + (V_{iN} - V_k)MC_{kN} = 0$$

$$V_{i1}MC_{k1} + V_{i2}MC_{k2} + \cdots + V_{iN}MC_{kN} = \left(\sum_{l=1}^{N} MC_{kl}\right)V_k$$

- We can rearrange the equation as:

$$V_k = \frac{1}{\sum_{l=1}^{N} MC_{kl}}\left(\sum_{l=1}^{N} V_{il}MC_{kl}\right)$$



Santa Clara University

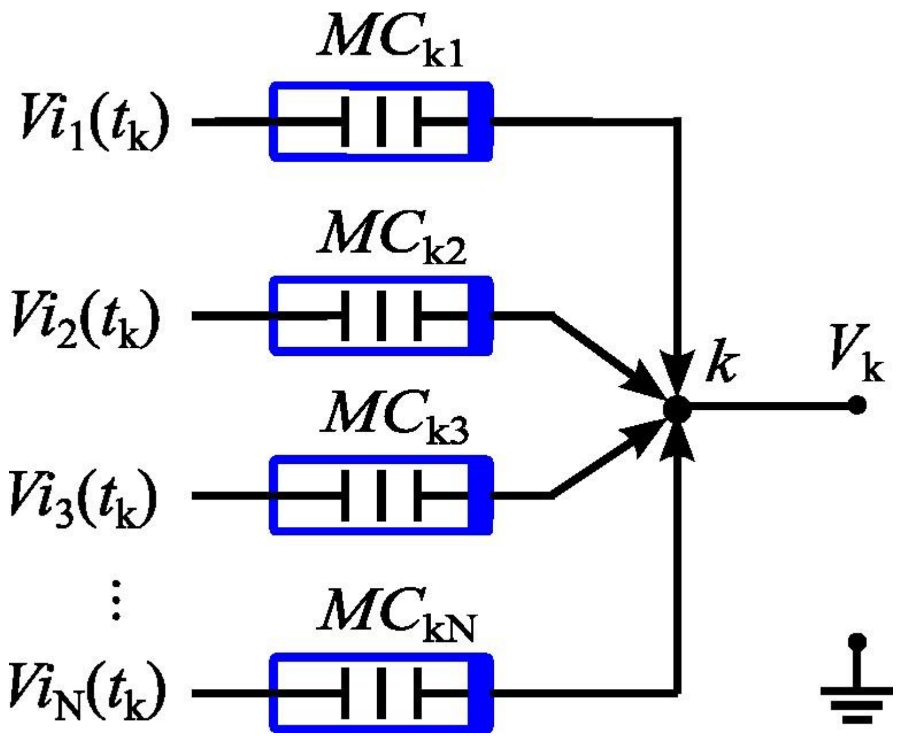
Memcapacitive Electrical Node (2)

- We can rearrange the equation as:

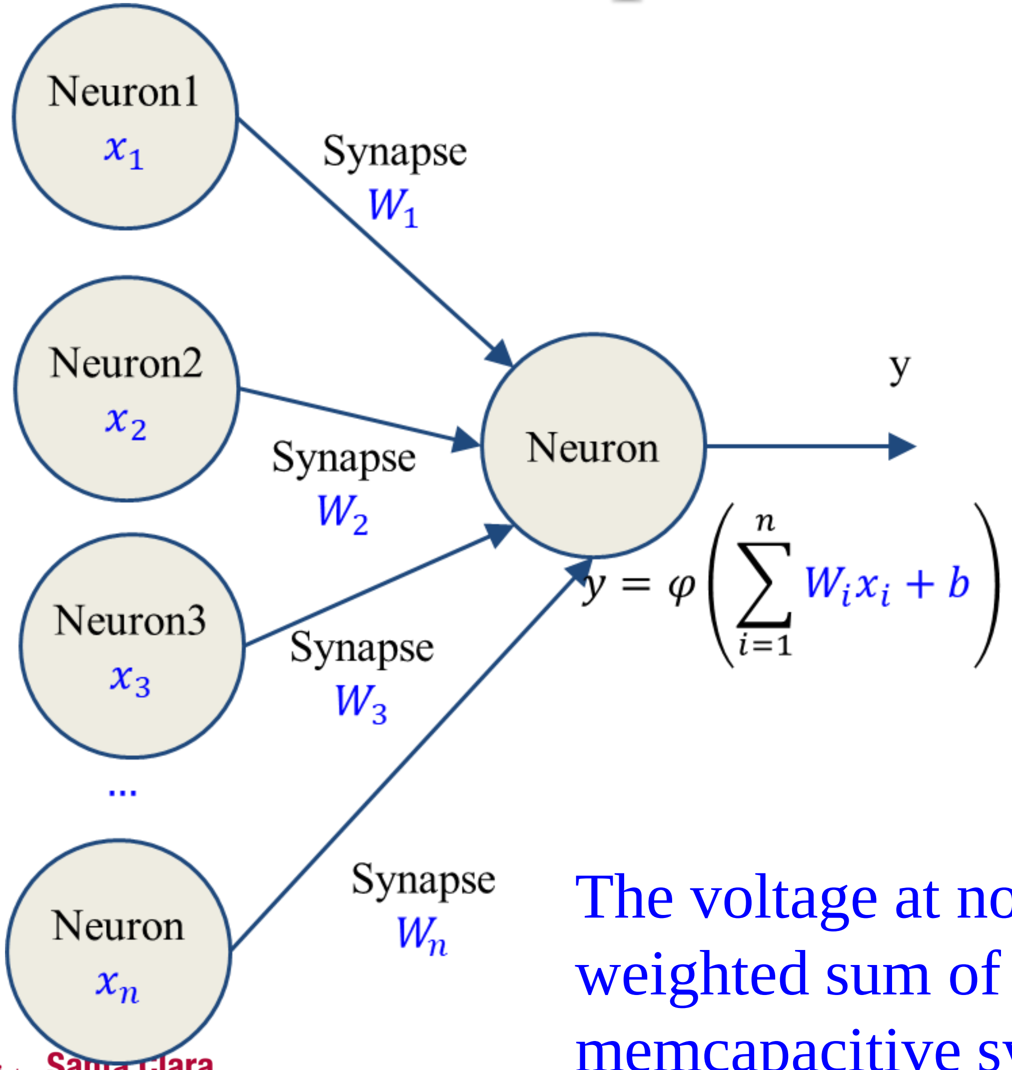
$$V_k = \frac{1}{\sum_{l=1}^{N} MC_{kl}} \left(\sum_{l=1}^{N} V_{il} MC_{kl} \right)$$

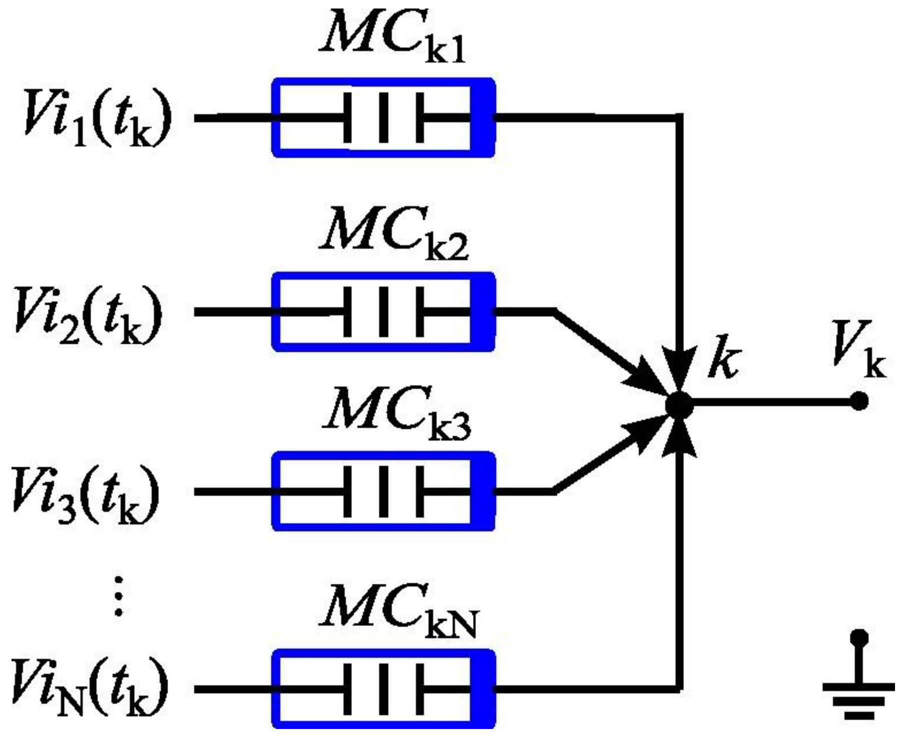
$$V_k = \varphi \left(\sum_{n=1}^{N} V_{il} MC_{kl} \right)$$

$$\varphi(x) = \frac{x}{\sum_{l=1}^{N} MC_{kl}}$$



Memcapacitive Electrical Node (3)





The voltage at node k is the weighted sum of input voltages and memcapacitive synapses.

$$V_k = \varphi\left(\sum_{n=1}^{N} V_{il} MC_{kl}\right)$$

Santa Clara University