Artificial Neural Networks
Recurrent Neural Networks

Jan Drchal
drchajan@fel.cvut.cz

Computational Intelligence Group
Department of Computer Science and Engineering
Faculty of Electrical Engineering
Czech Technical University in Prague

Outline

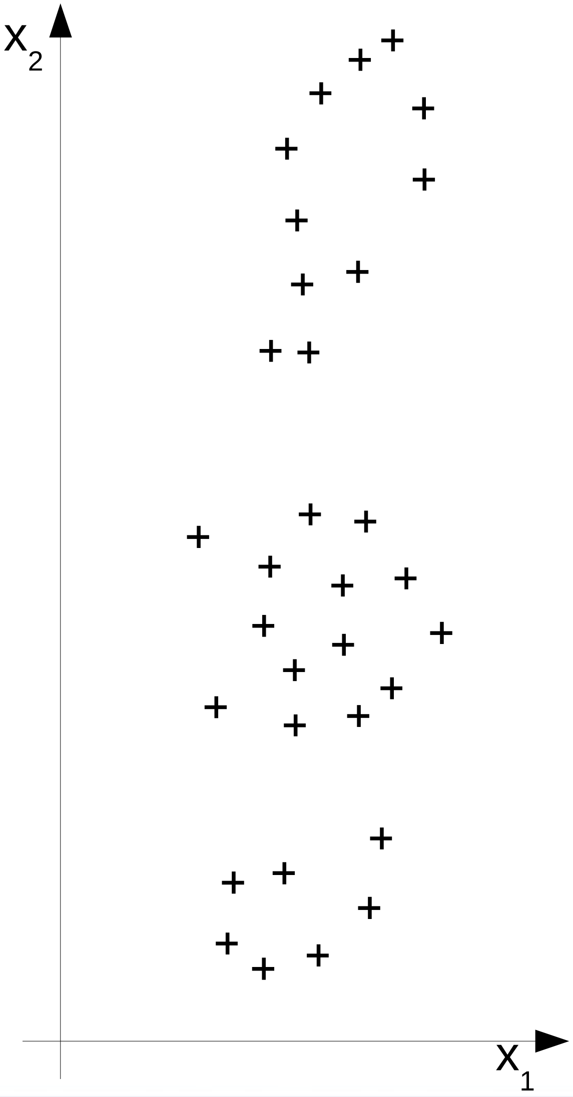
- Time & dynamics – motivation.
- Time-series with feed-forward ANNs.
- Recurrent ANNs:
    - architectures,
    - recall (evaluation),
    - training.

Motivation

- Real world can be understood as a collection of different signals:
  Let's focus on **time & dynamics.**
- Tasks:
  - prediction (economy, weather, ...),
  - recognition (speech, video, ...),
  - modelling (text – grammar, automata description, ...),
  - filtration.
- Working with **time-series**.

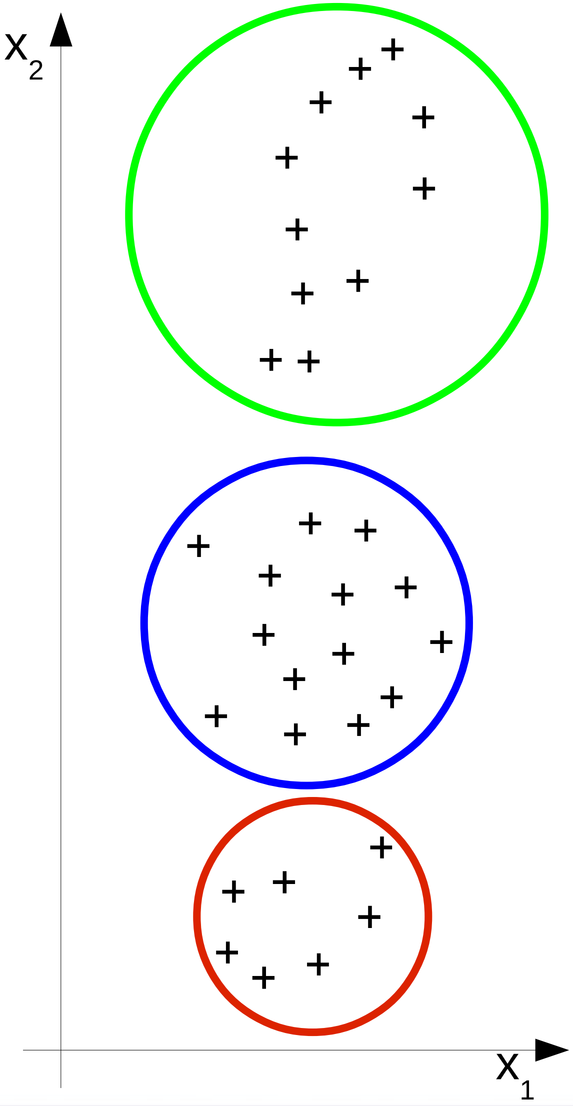
Motivation, Time & Signals

- **Static data**, independent, isolated data vectors.
- What do you see in the figure?



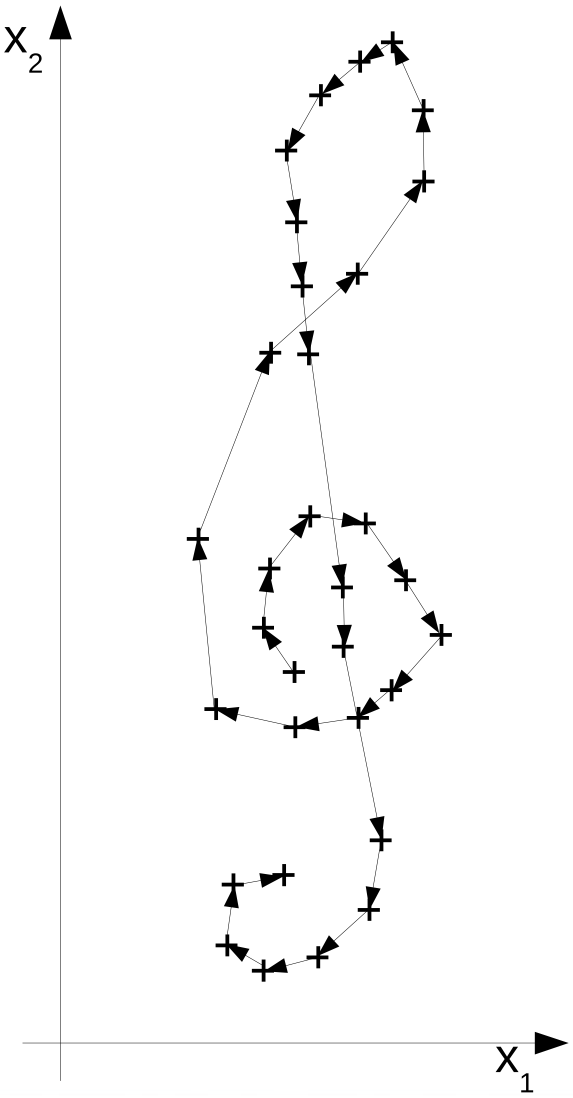
Motivation, Time & Signals

- **Static data**, independent, isolated data vectors.
- What do you see in the figure?
  - 3 clusters in Euclidean space.



Motivation, Time & Signals II

- **Dynamic (sequential) data.**
- Temporal order of vectors utilized.
- Proper meaning of the data can be now recognized - **G-clef**.



Processing Dynamic Data

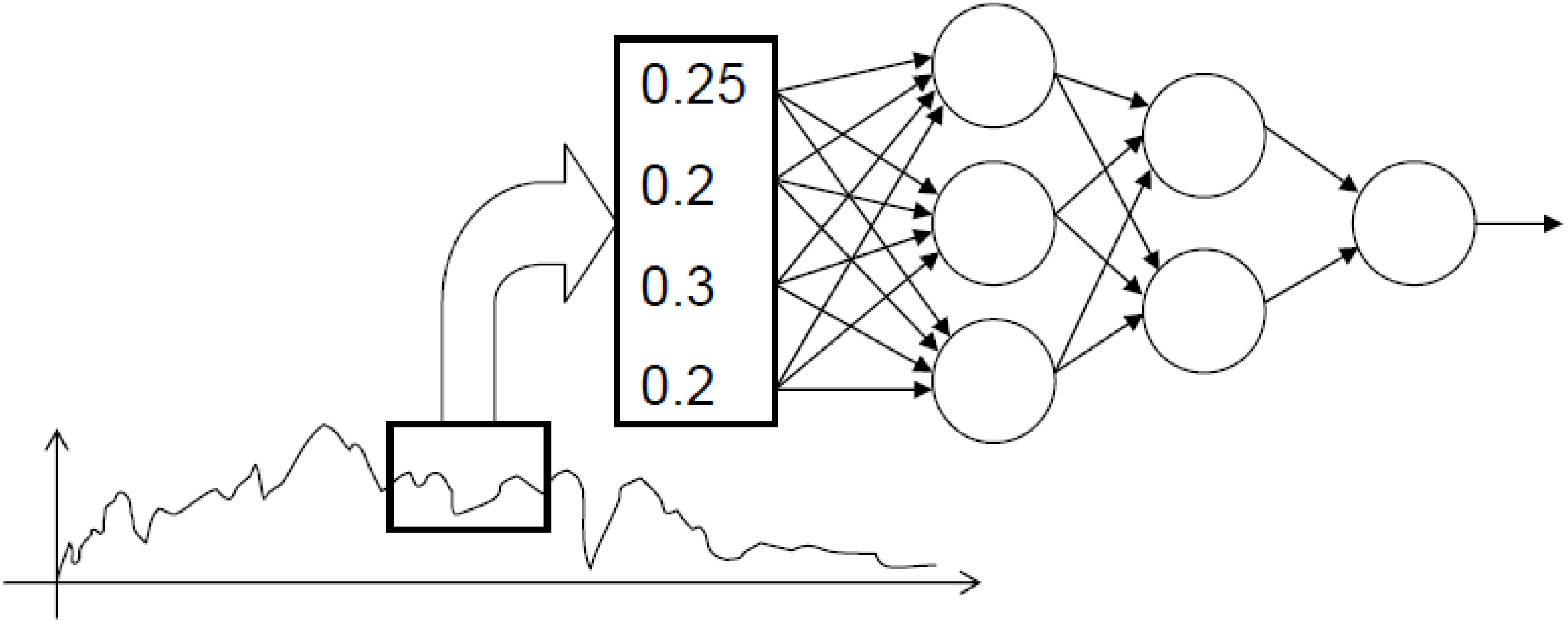
- Architectures:
  - feed-forward networks,
  - **recurrent networks** (RNNs):
    - partially/fully recurrent networks.
- Dynamics of recurrent networks:
  - **discrete time dynamics**,
  - continuous time dynamics.

Dynamics

- Discrete time:
  - the network state in time *t+1* depends on the network state in time *t.*
- Continuous time:
  - special types of neurons: i.e. leaky-integrator neurons, spiking neurons.

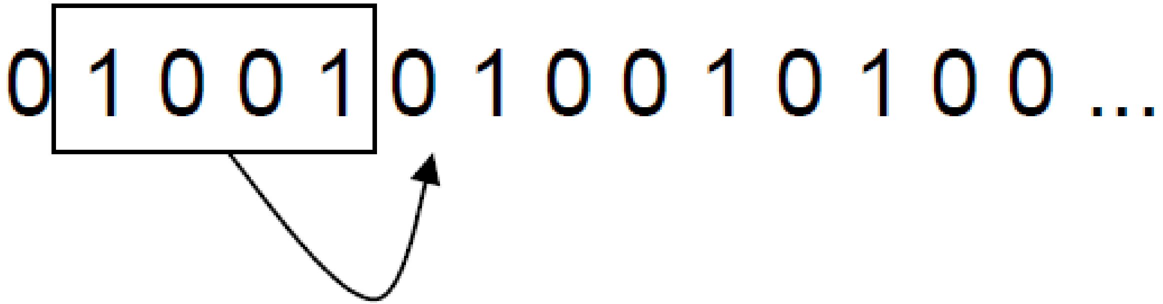
Processing Sequence by Feedforward Neural Network

- Simplest way: **sliding window → Time Delay Neural Networks (TDNN)**
- Feedforward ANN → *combination circuit*.

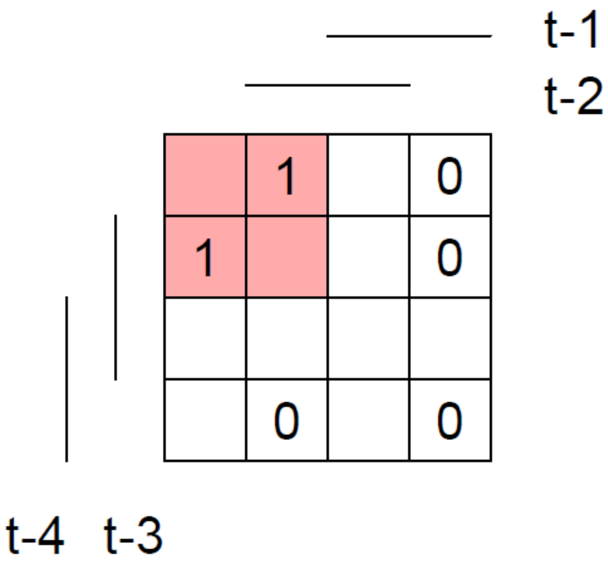


Input values *x(t-1), x(t-2), x(t-3)* or *x(t-1), x(t-2), x(t-4), x(t-8)*...

Time Delay Neural Network (TDNN) Example



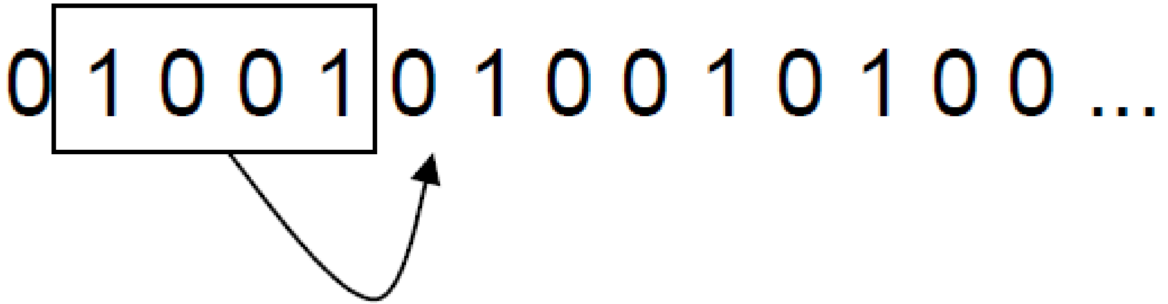
0 1 0 0 1 0 1 0 0 1 0 1 0 0 ...



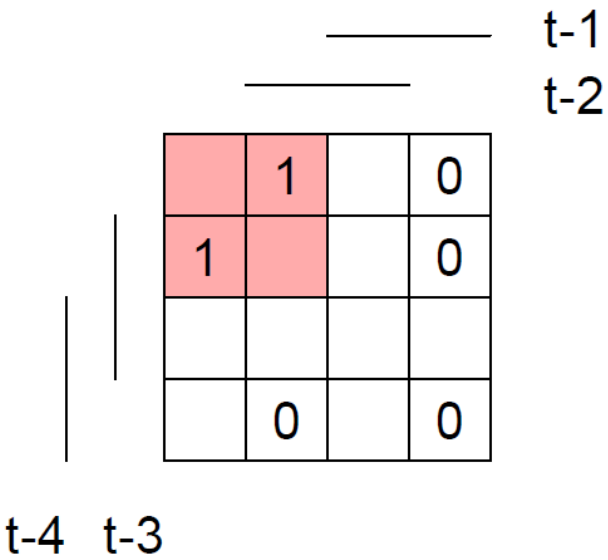
	1		0
1			0
	0		0

Karnaugh map

Time Delay Neural Network (TDNN) Example

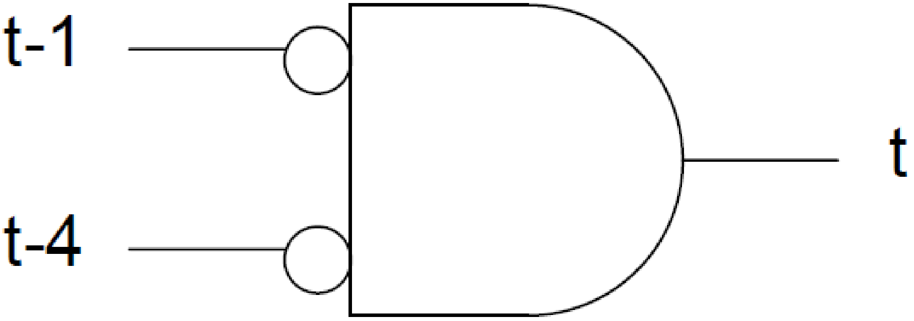


0 1 0 0 1 0 1 0 0 1 0 1 0 0 ...



	1		0
1			0
	0		0

Karnaugh map



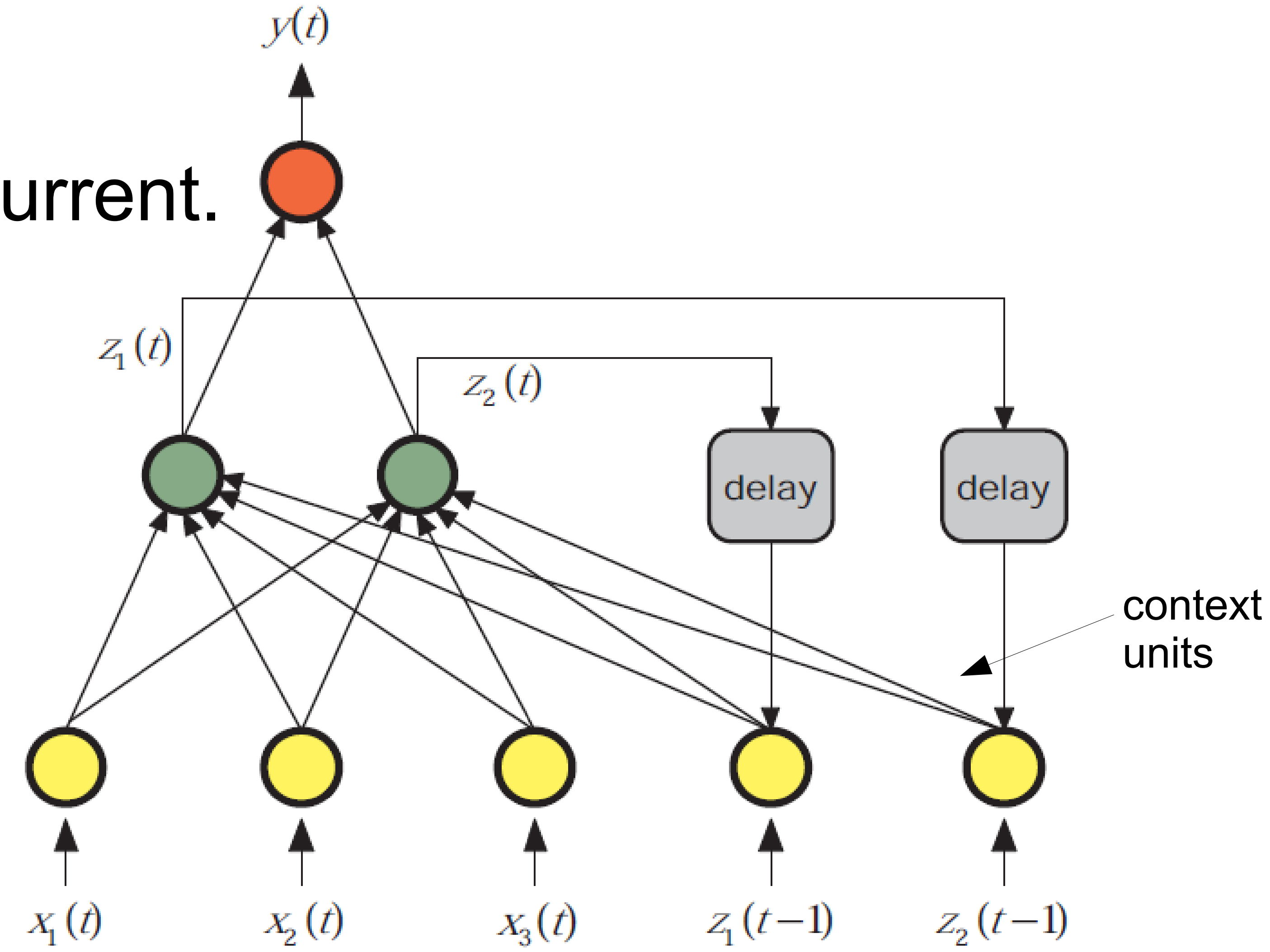
NOR

TDNN Remarks

- We need to know the period (memory depth).
- Disadvantageous for longer periods.
- Large growth of number of neurons.
- Better use sequential circuitry → native approach → states/memory.
- Examples:
  - automatons,
  - grammars,
  - regular expressions,
- **Recurrent connections.**

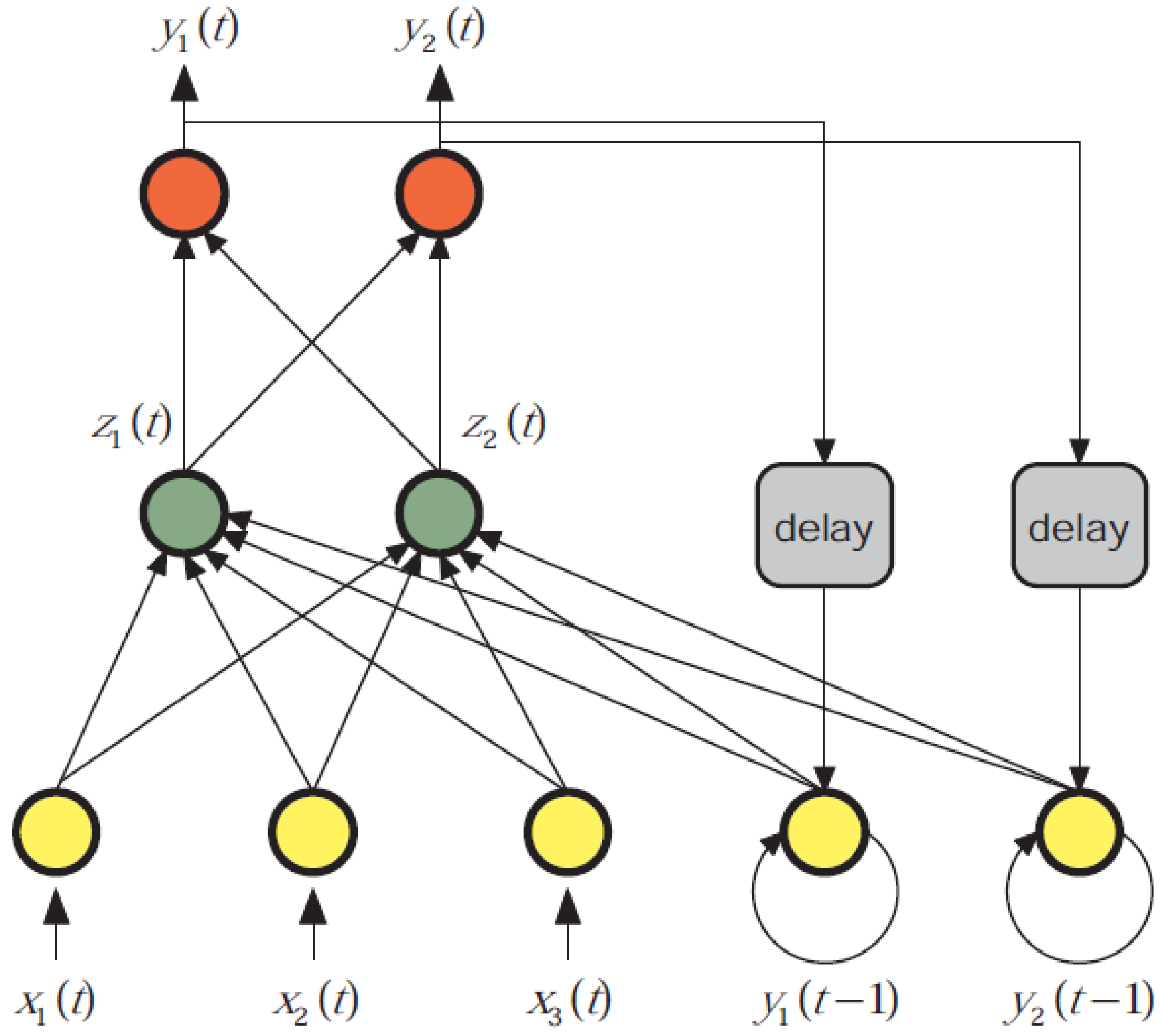
Elman's Network

- 1990.
- Partially recurrent.



Jordan's Network

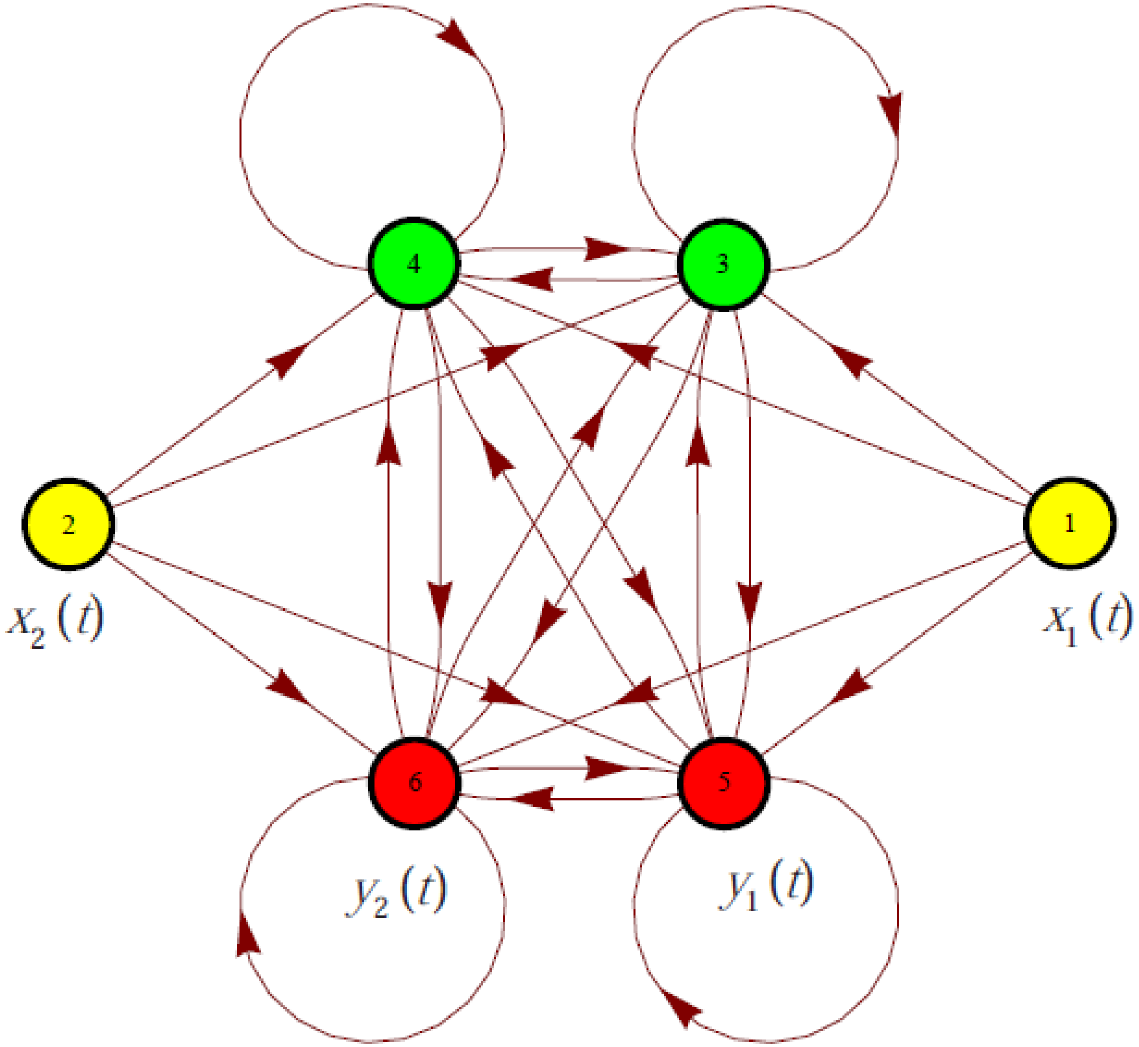
- 1989.



Fully Connected Architecture

Note: every network is a sub-network of a fully-recurrent network having the same # of neurons.

Representation?



RNN Recall

- **Synchronous**:
  - change all neuron states simultaneously,
  - precompute neuron activities for time *t* based on state in *t-1* → change all activities at once.
- **Asynchronous**:
  - compute and immediately set activities for individual neurons,
  - proceed in predefined order (corresponding to signal flow, random).

RNN Recall 2

- When to read RNN output?
  - Perform predefined number of simulation steps.
  - **Wait until relaxed.**
  - **Continuously push new input data to RNN** inputs and read delayed responses at outputs → typical approach for controlling tasks.
    - Note the delay equal to the depth of the network (shortest path from inputs to outputs).

Response of RNNs

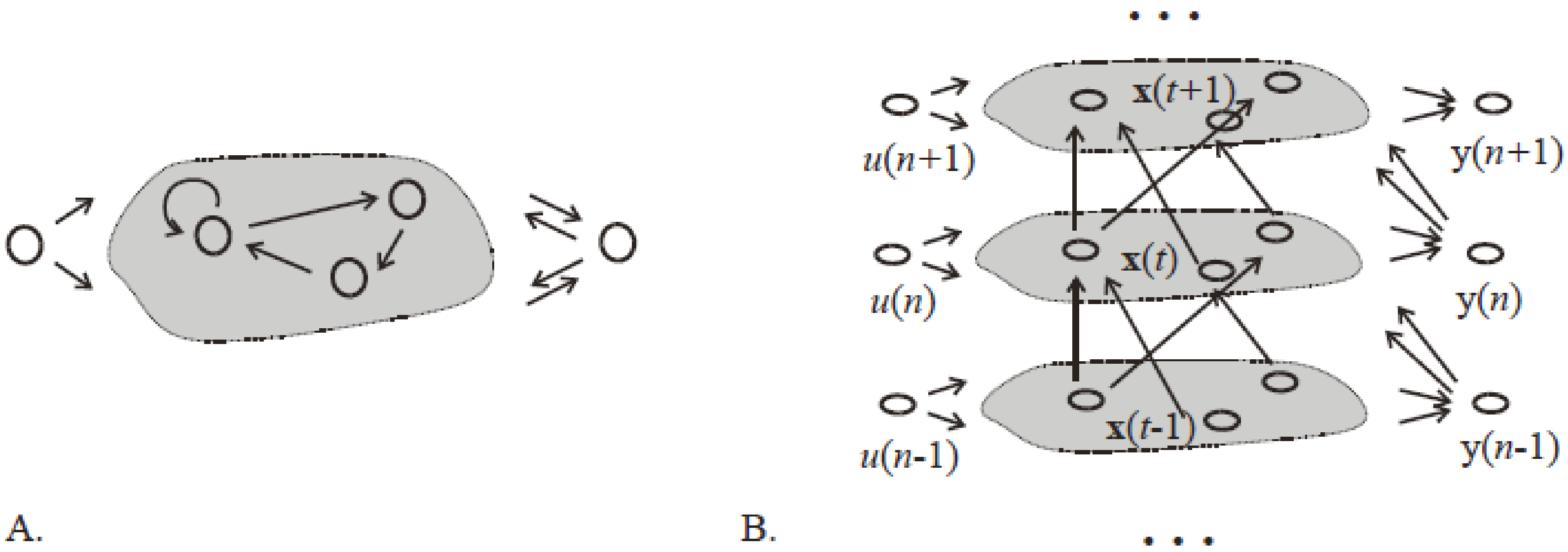
- Possibilities of output behaviour:
  - **convergence to a stable value**,
  - oscillation,
  - chaotic behaviour.
- Let's see demonstration with a polynomial neuron....

Relaxation (Settling)

- Relaxation – wait until system (RNN) settles to a locally optimal state.
- Typical implementation:
    - Repeat evaluation steps of RNN until the difference between outputs of successive steps drops below a certain threshold.

Backpropagation Through-Time (BPTT)

- 1986 - Rumelhart, Hinton, Williams
- Unfold ANN in time & use BP.



Jaeger: A tutorial on training recurrent neural networks, covering BPTT, RTRL, EKF and the "echo state network" approach, 2003

Backpropagation Through-Time 2

- Unfolding *p* times: *p*-BPTT – user must choose *p* :(
- Low time complexity, single epoch *$O(TN^2)$:*
  - *T ... # of training pairs,*
  - *N … # of internal neurons.*
- Slow convergence - same reasons as BP*.*
- Hard to achieve memory longer than 10-20 steps.
- Unlike BP, not guaranteed to converge to a local error minimum!
- Modifications:
  - BBPTT: Batch Backpropagation Through Time – averages weight changes over epoch,
  - QPTT: Quickprop Through Time.

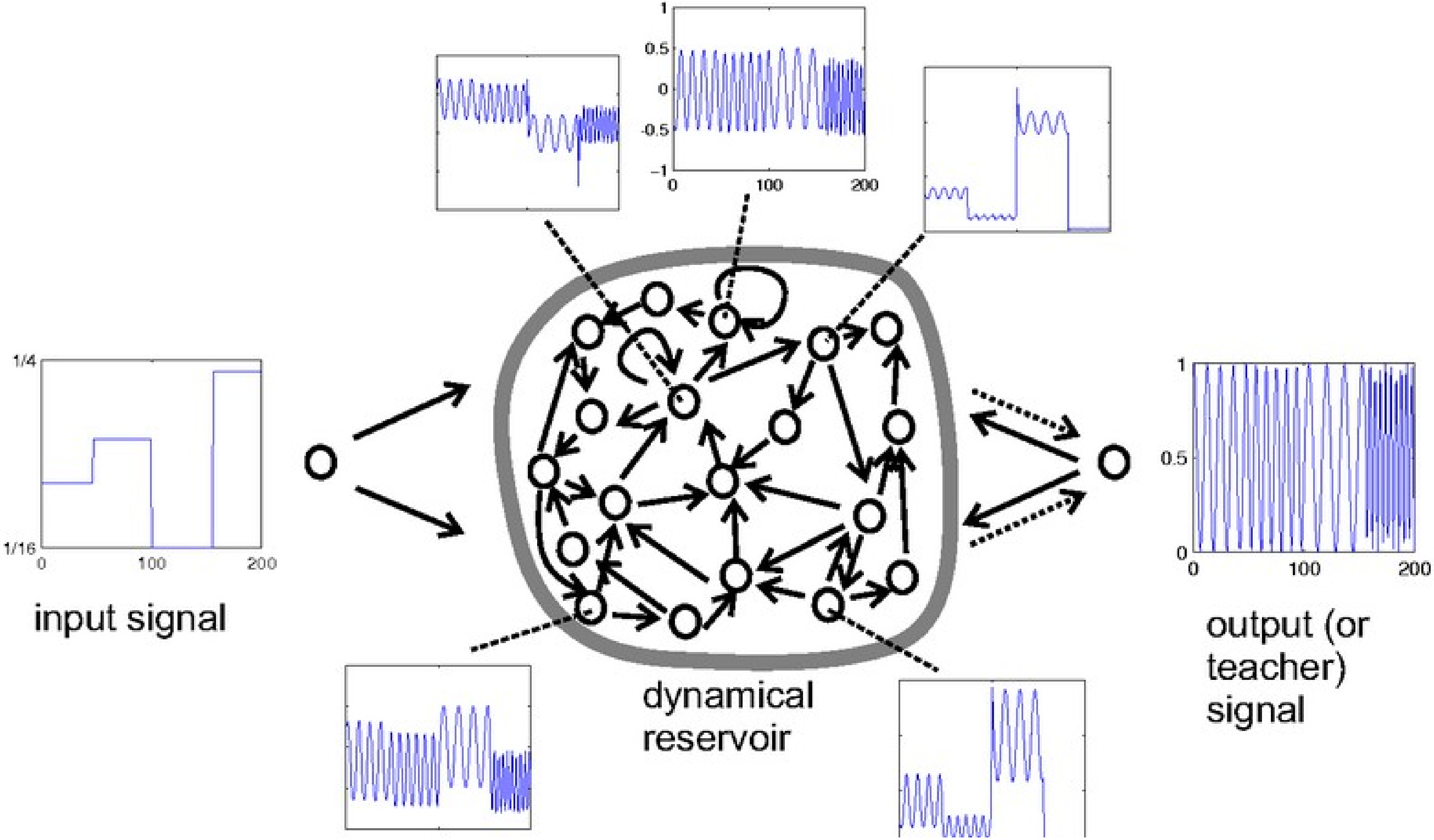
Real-Time Recurrent Learning (RTRL)

- 1989 - Williams & Zipser
- Different approach to compute gradient.
- Often called “forward-propagation”.
- No *p* to choose :)
- Complexity of a single epoch: $O(N^4)$:(
- Better convergence, but slower epoch than BPTT.

Echo State Networks (ESN)

- 2001, Jaeger
- Observation: **most often, dominant changes happen to output weights while training.**
- Pool of randomly connected neurons with **random** weights → **dynamical reservoir**.
- Connect inputs to reservoir: **random**, fully-connected.
- Connect reservoir to outputs: fully-connected, **these weights will undergo training**.

Echo State Networks Example: Tunable Frequency Generator



Read this: http://www.scholarpedia.org/article/Echo_state_network

Echo State Networks (ESN) 2

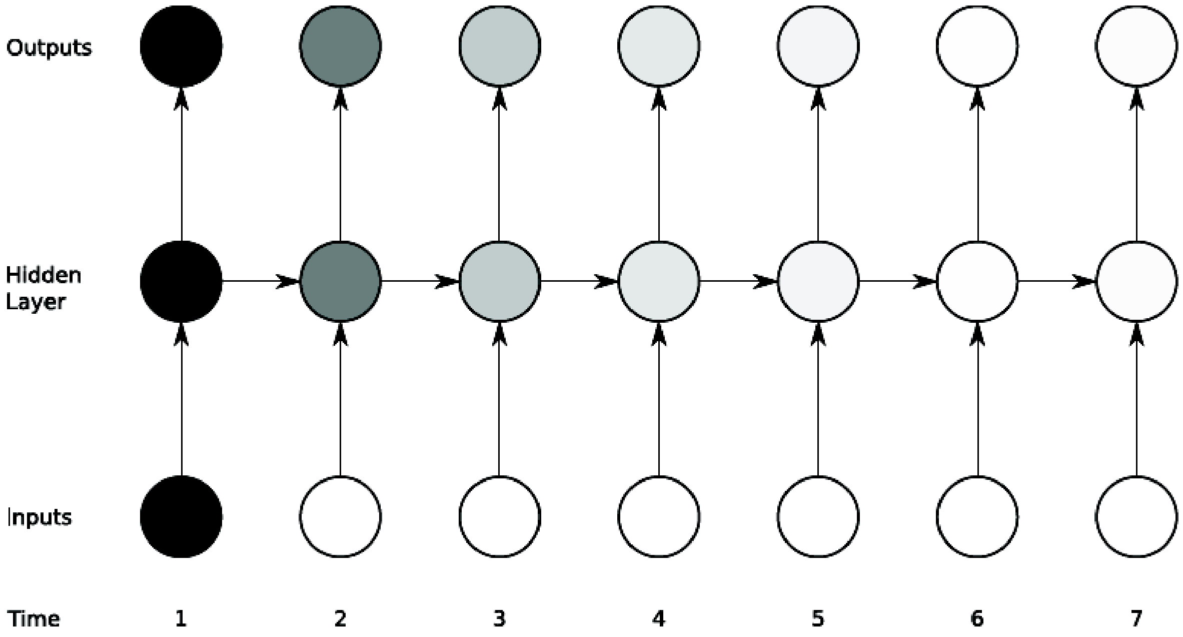
- Training uses Linear Regression → general methods to estimate parameter of linear model (see http://en.wikipedia.org/wiki/Linear_regression)
- Very fast.
- Unlike previous algorithms: **does not suffer from bifurcations** (change in dynamics caused by small changes of system's parameters).
- For more info see:

  http://minds.jacobs-university.de/esn_research

Vanishing Gradient

It is hard to train RNNs with delays > 10 timesteps.

colour - sensitivity to input from time = 1

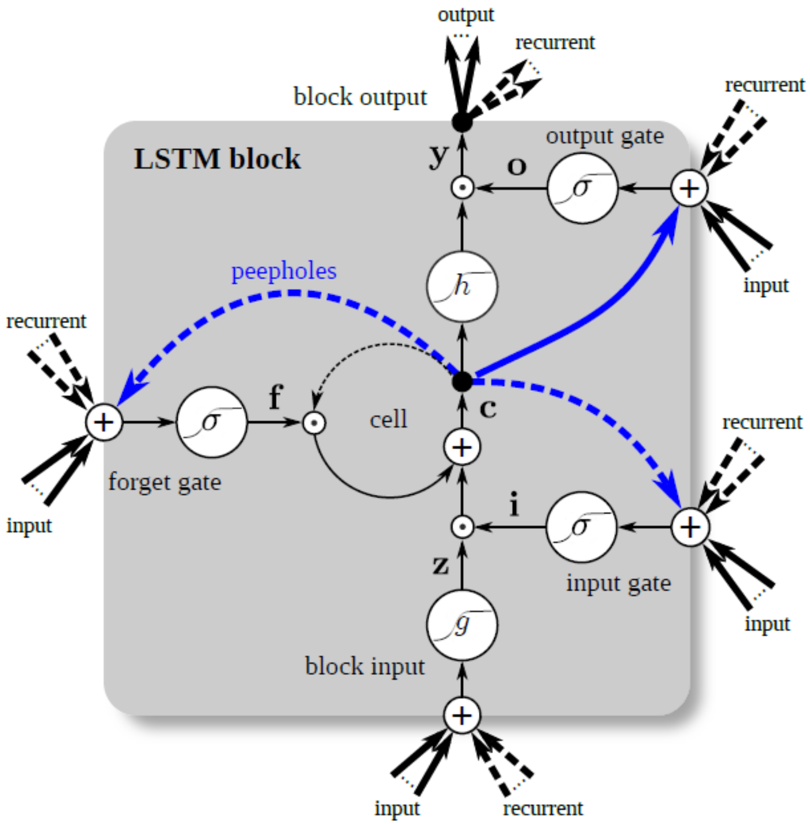


Exponential decay of sensitivity as new inputs overwrite the activation of hidden unit and the network “forgets” the first input.

Alexander Graves, Supervised Sequence Labelling with Recurrent Neural Networks, 2008

Long Short-Term Memory (LSTM)

- 1997, Juergen Schmidhuber
- LSTM cell: subnetwork,
- Supports both long and short-term memories – **thousands timesteps**!
- Combined with classic networks.
- **Constant Error Carrousel** (CEC): with linear transfer function stores the information.
- **Gates** control access to the memory: reminds Write Enable (WE) and Output Enable (OE) on memory ICs.
- Typically learned using BP.



$$\mathbf{z}^t = g(\mathbf{W}_z\mathbf{x}^t + \mathbf{R}_z\mathbf{y}^{t-1} + \mathbf{b}_z) \qquad \textit{block input}$$

$$\mathbf{i}^t = \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{R}_i\mathbf{y}^{t-1} + \mathbf{p}_i \odot \mathbf{c}^{t-1} + \mathbf{b}_i) \qquad \textit{input gate}$$

$$\mathbf{f}^t = \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{R}_f\mathbf{y}^{t-1} + \mathbf{p}_f \odot \mathbf{c}^{t-1} + \mathbf{b}_f) \qquad \textit{forget gate}$$

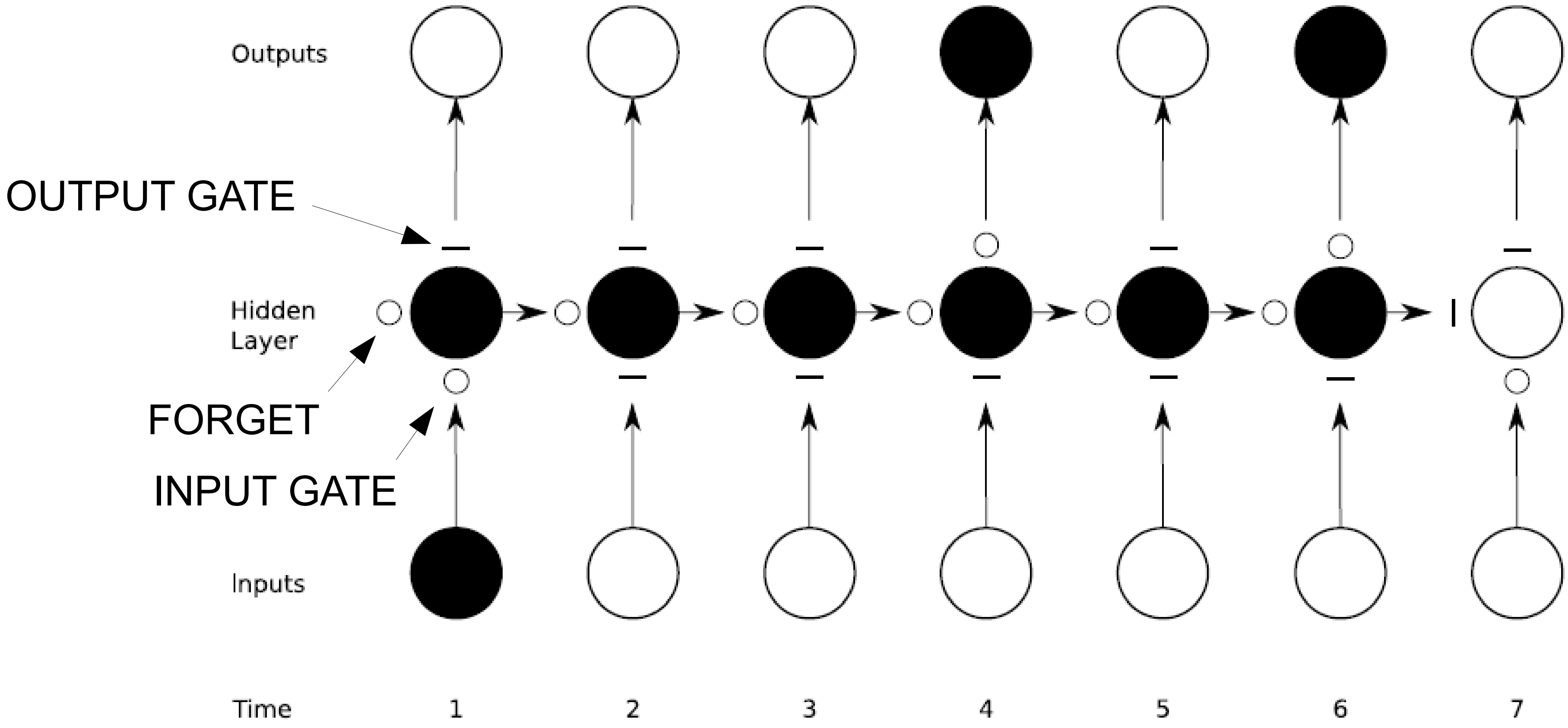
$$\mathbf{c}^t = \mathbf{i}^t \odot \mathbf{z}^t + \mathbf{f}^t \odot \mathbf{c}^{t-1} \qquad \textit{cell state}$$

$$\mathbf{o}^t = \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{R}_o\mathbf{y}^{t-1} + \mathbf{p}_o \odot \mathbf{c}^t + \mathbf{b}_o) \qquad \textit{output gate}$$

$$\mathbf{y}^t = \mathbf{o}^t \odot h(\mathbf{c}^t) \qquad \textit{block output}$$

Long Short-Term Memory 2

Gate states: O (opened), – (closed).



Long Short-Term Memory 3

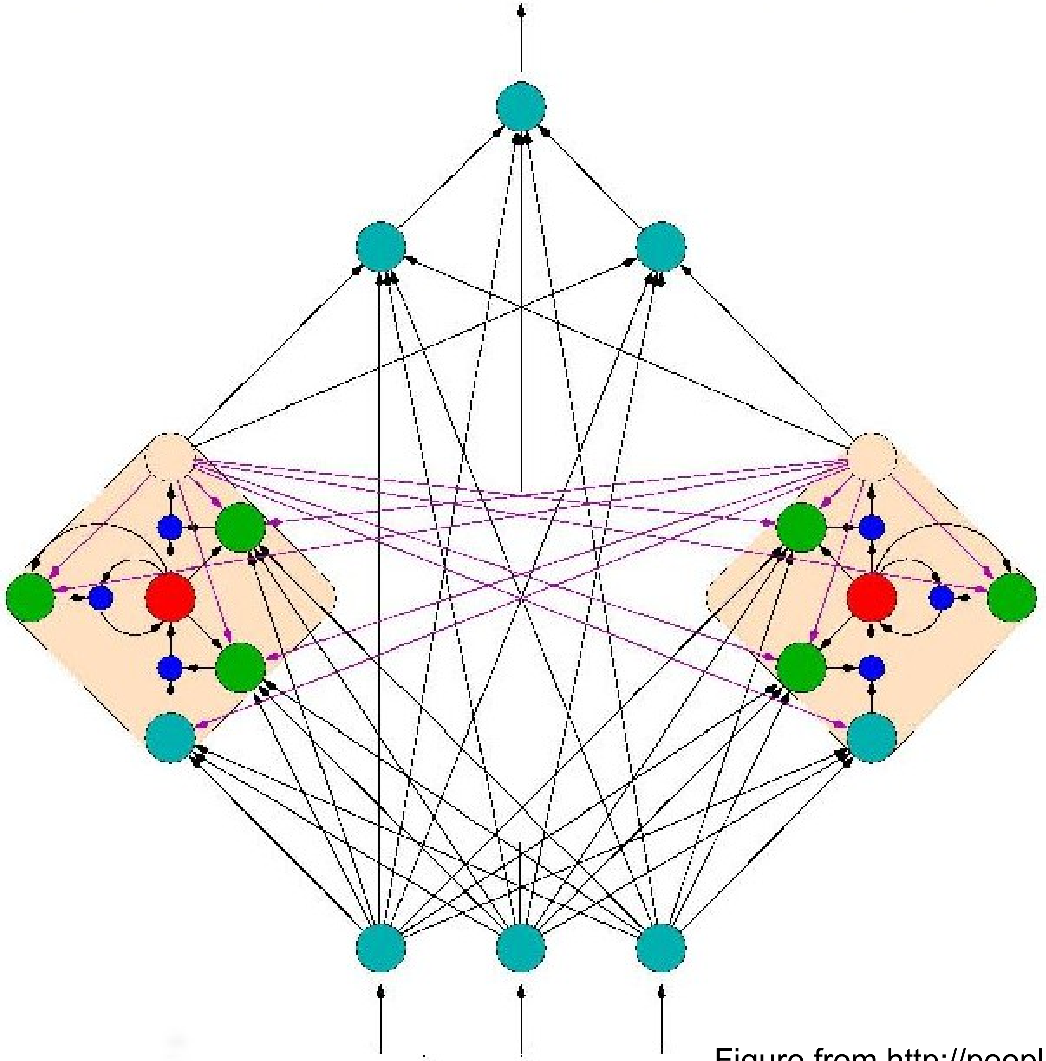


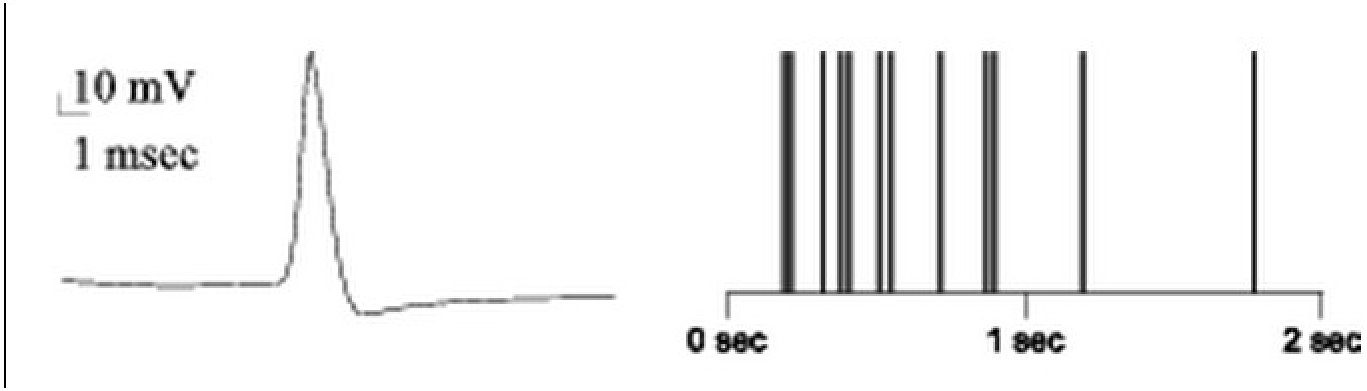
Figure from http://people.idsia.ch/~juergen/lstm/index.htm

Long Short-Term Memory 3

- Read this paper:
  Greff et al.: LSTM: A Search Space Odyssey, 2015
- State-of-the art applications:
  - handwriting recognition,
  - language modelling/translation,
  - speech modelling/synthesis,
  - activity recognition,
  - music composition (e.g., blues improvisation),
  - audio/video analysis,
  - and many others...
- Often combined with other Deep Neural Networks.

Dynamic Neuron Model

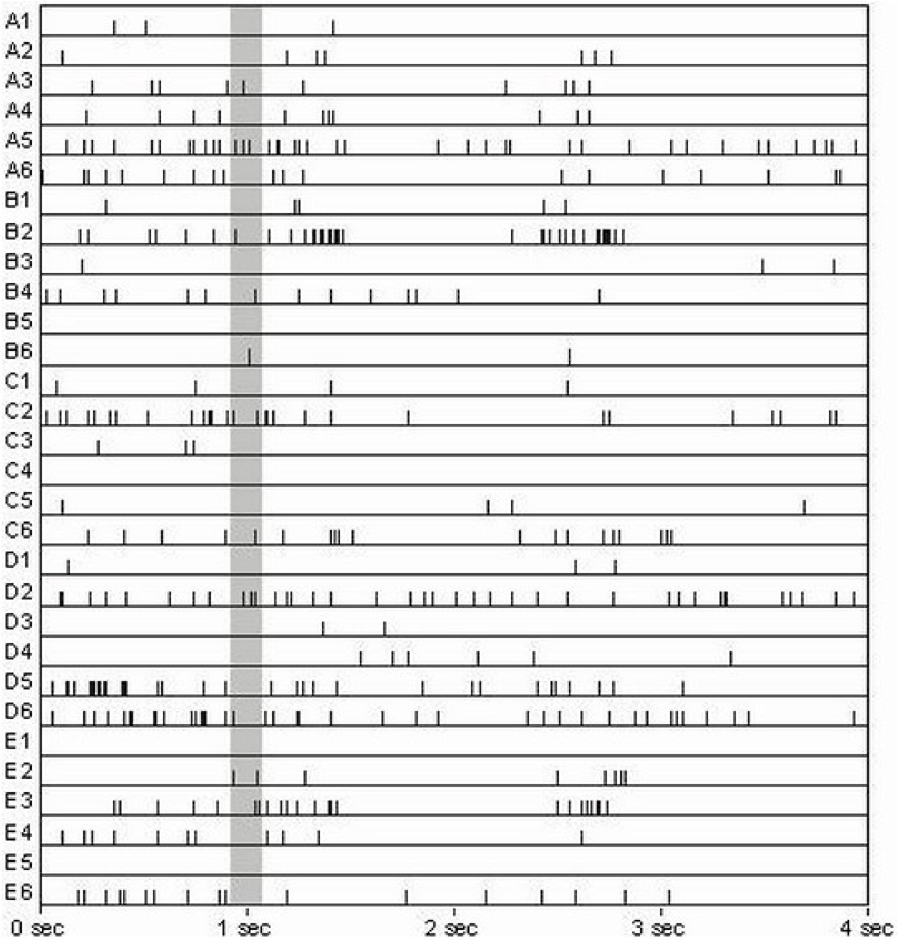
- Real neurons don't work with activation levels → they fire **spike trains**.



We talk about rate: spikes/time unit → **firing rate**.

http://www.igi.tugraz.at/maass/123/node2.html
and K. Stanley's presentation CAP6938: Leaky Integrator Neurons and CTRNNs

Spike Trains



http://www.igi.tugraz.at/maass/123/node2.html
and K. Stanley's presentation CAP6938: Leaky Integrator Neurons and CTRNNs

- 30 neurons firing from monkey striate cortex (Krüger and Aiple, 1988).
- Selected 150ms range shows time needed for complex computations i.e. face recognition.
- **Spiking neural networks**:
  - different models of neurons: integrator accumulates inner potential, then fires...
  - different simulation approach.