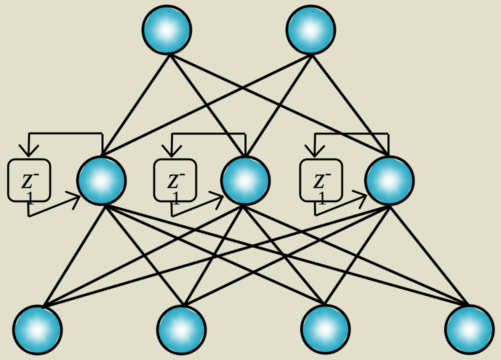
Recurrent neural networks

- Neural networks applied to tasks involving dynamic I/O like time series prediction:
  - Time delay neural networks (TDNNs)
  - Recurrent neural networks (RNNs)
    - Locally recurrent neural networks
    - Partially recurrent neural networks
    - Fully recurrent neural networks
    - Long short-term memory networks (LSTM)

Locally recurrent neural networks

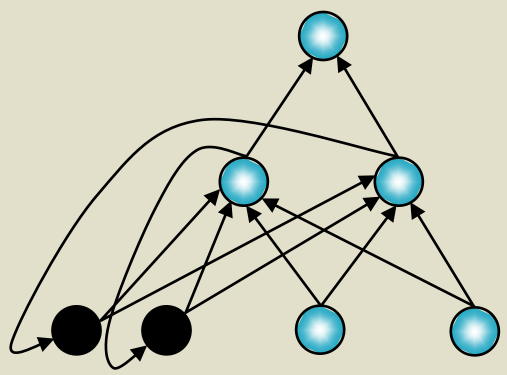
- Neurons are connected in layers like in feedforward nets

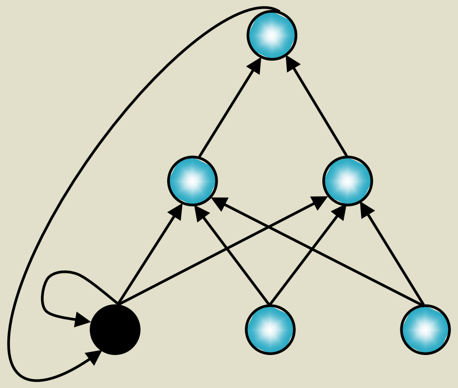
- No feedback connections between different units exist



Partially recurrent neural networks

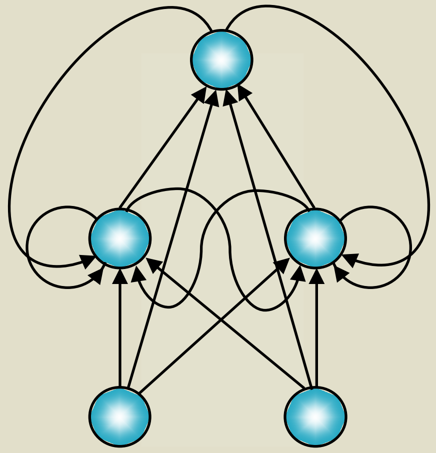
- Are mainly feedforward nets that include feedback connections to a set of units called context unit


Elman


Jordan

Fully recurrent neural networks

- Fully RNNs make use of global recurrences
- Do not impose any condition on the way the neurons are connected
- Each neuron has a connection to every other unit including itself



Vanishing gradient

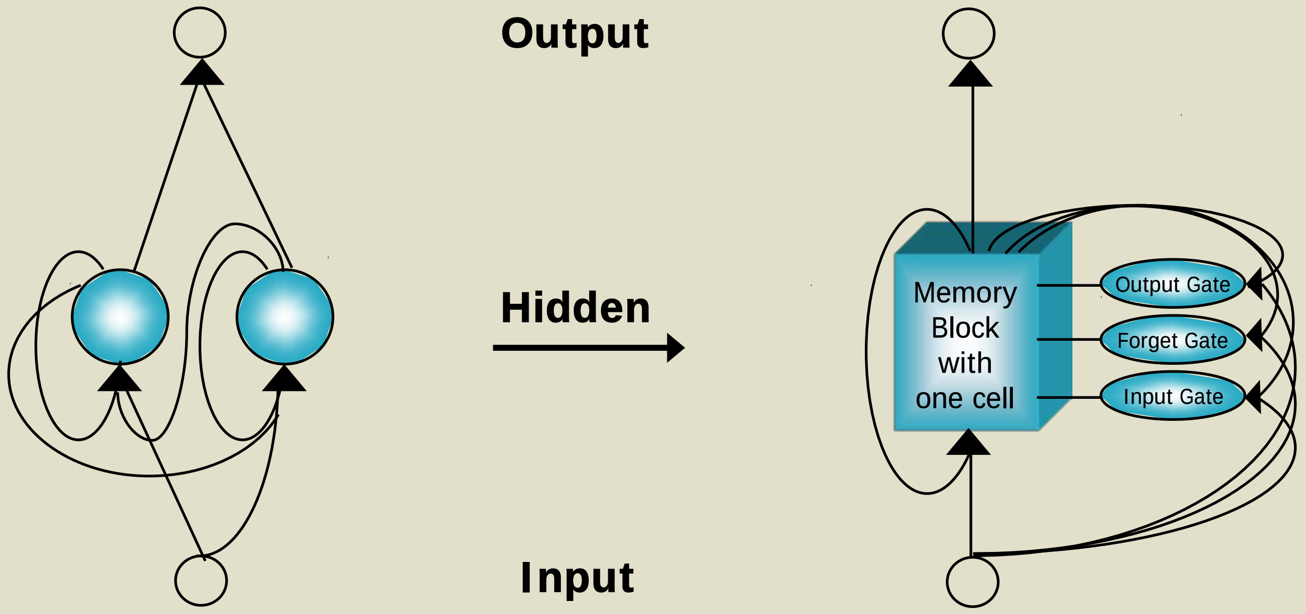
- Memory capacity in traditional recurrent networks is theoretically unlimited
- However these networks suffer from the vanishing gradient problem
- So they are unable to deal with common long-term dependences
- Long short-term memory (LSTM), a novel recurrent architecture overcomes this problem

Long short-term memory architecture

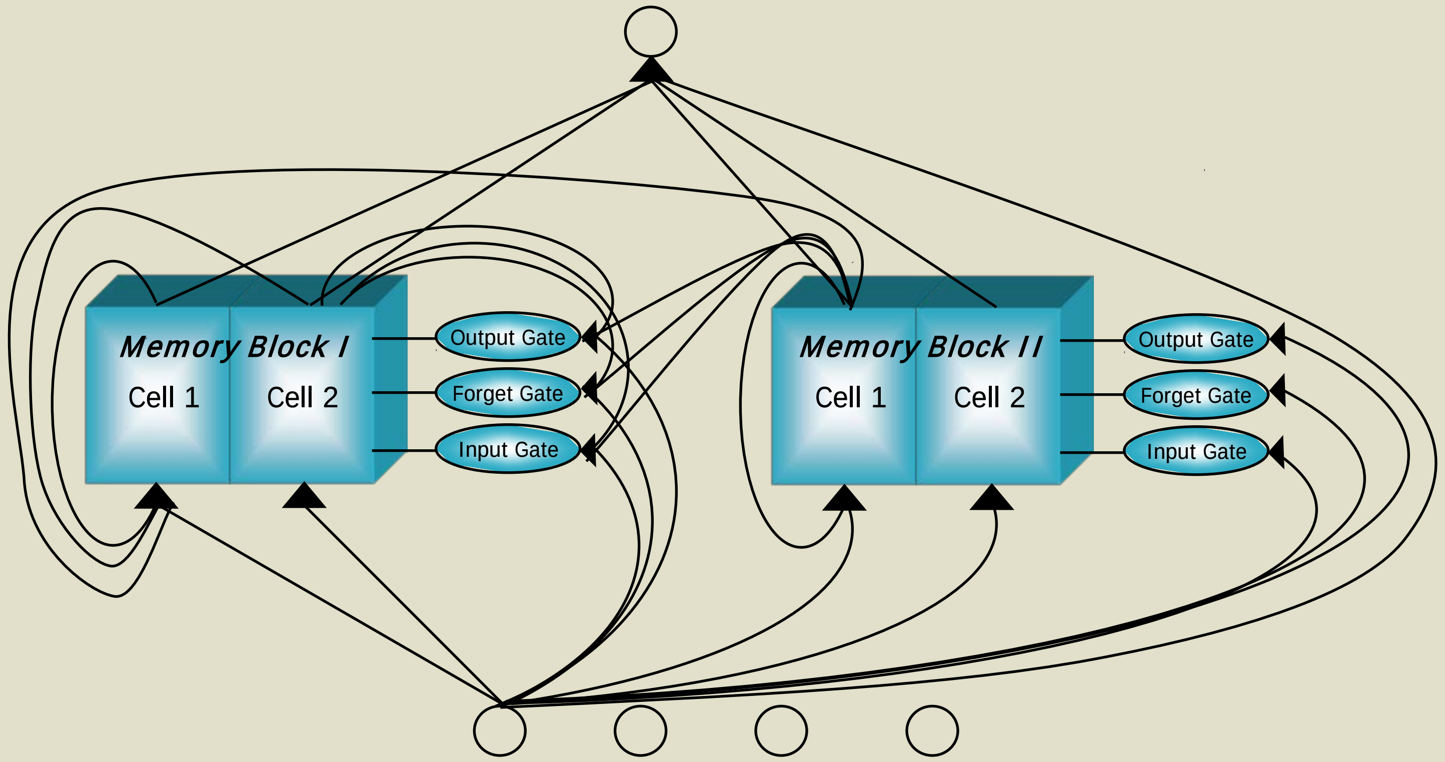
- Long short-term memory (LSTM) [Hochreiter & Schmidhuber, 1997] :
  - A RNN designed to solve the vanishing gradient problem
  - LSTM allows the unit activations to retain important information over longer periods of time
  - Remarkable results in previous experimental studies on artificial long-time lag tasks

LSTM basic unit

- The hidden units of a conventional recurrent neural network have been replaced by memory blocks
- Each memory block has one or more memory cells and three gating units shared by all cells in the block

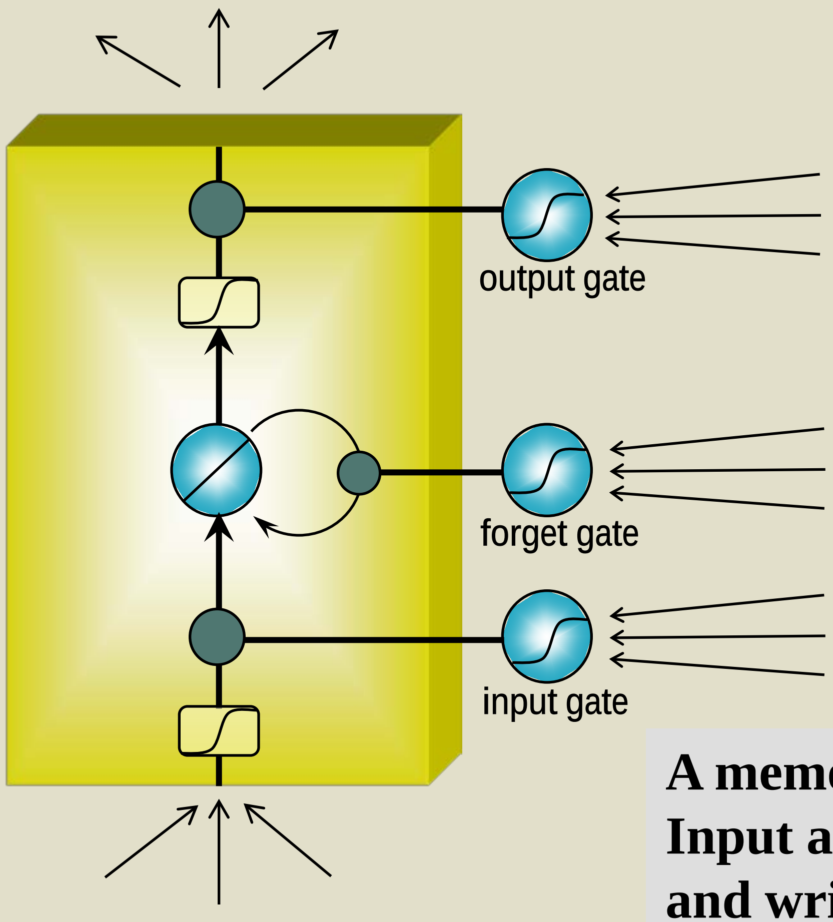


LSTM topology



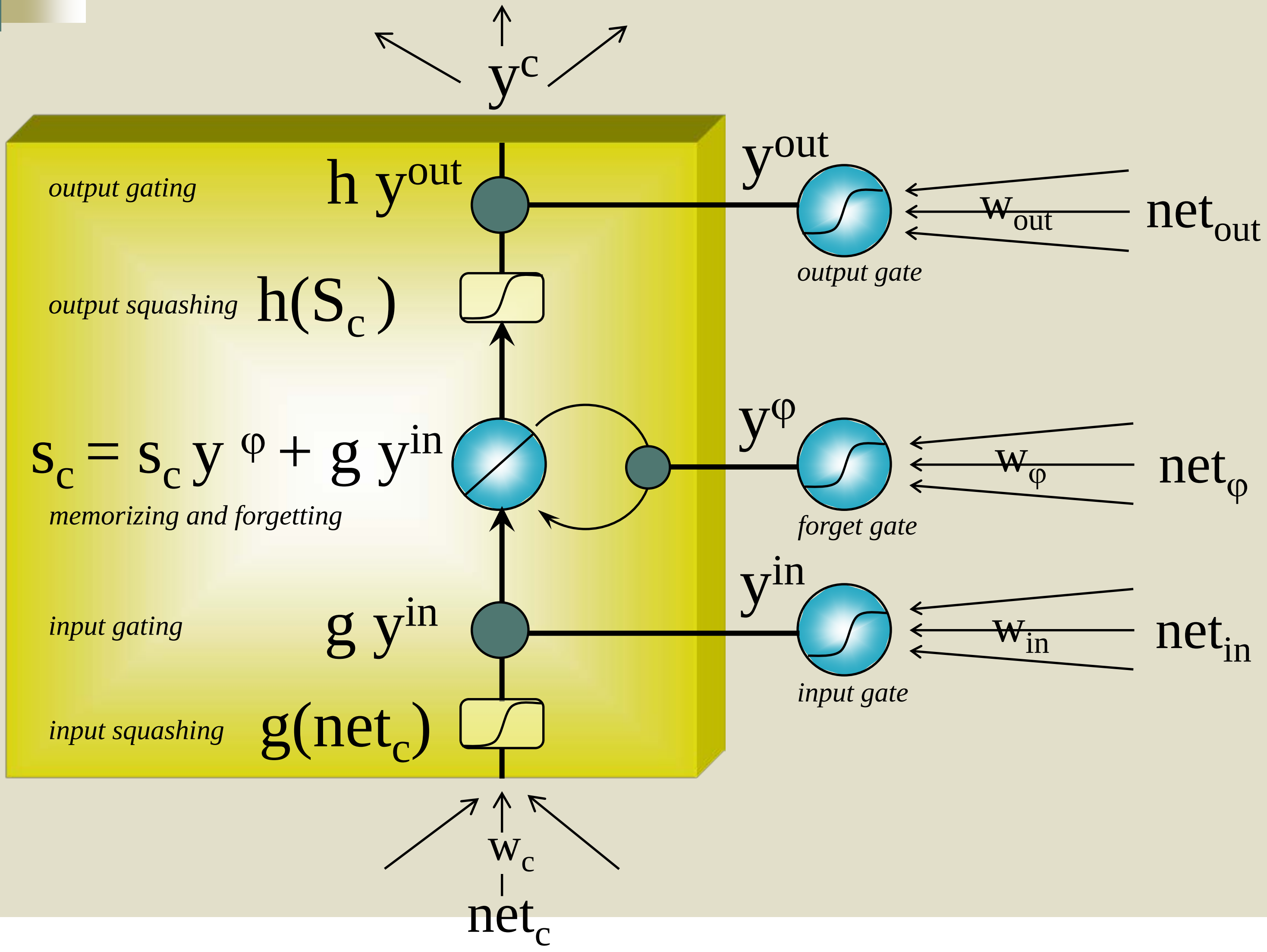
Input units feed all the memory cells and gates, whereas the output activations of memory cells feed the output units as well as all the memory cells and gates in the hidden layer.

Memory block schema

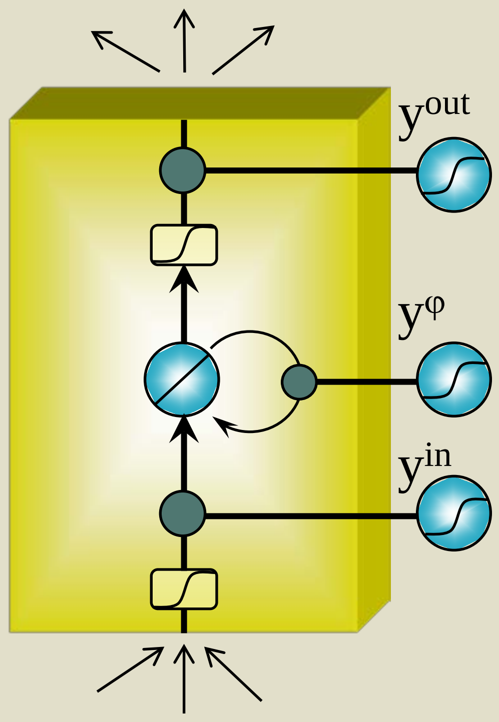


A memory block with only one cell. Input and output gate regulate read and write access to the cell internal state.

Memory Block Schema

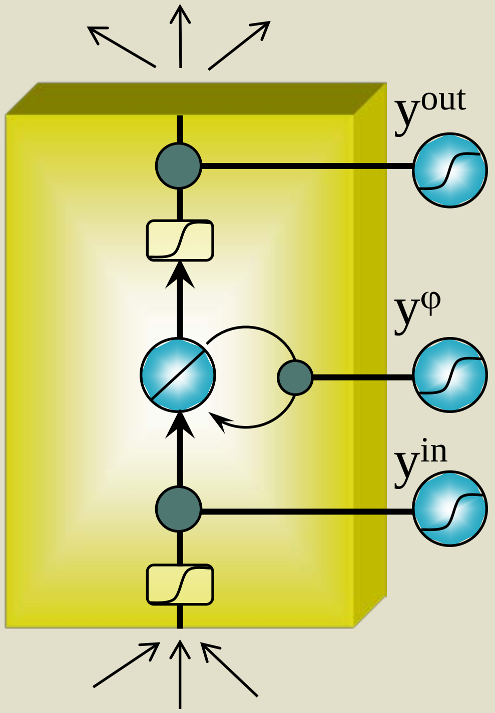


Gates equations



$$net_{in}(t) = \sum_m w_{in\,m} y^m(t-1), \quad y^{in}(t) = f_{in}\left(net_{in}(t)\right)$$

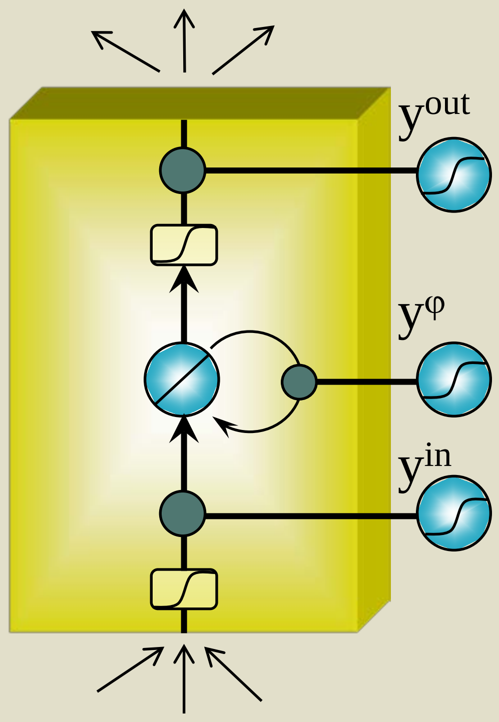
Gates equations



$$net_{\varphi}(t) = \sum_m w_{\varphi m} y^m(t-1), \quad y^{\varphi}(t) = f_{\varphi}(net_{\varphi}(t))$$

$$net_{in}(t) = \sum_m w_{in\,m} y^m(t-1), \quad y^{in}(t) = f_{in}\,(net_{in}(t))$$

Gates equations



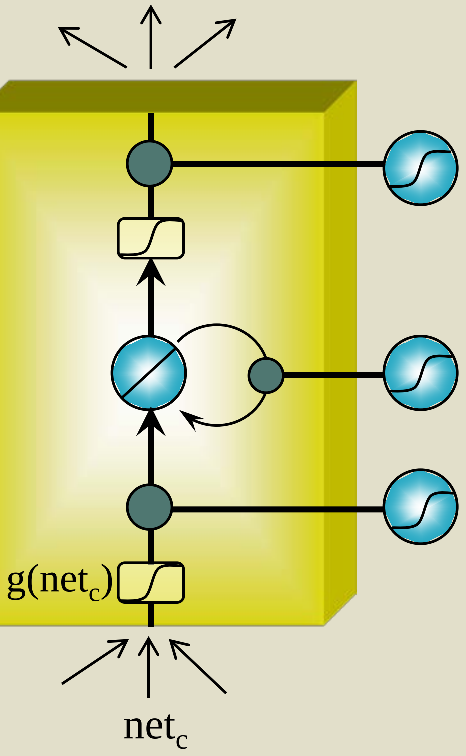
$$net_{out}(t)=\sum_m w_{out\,m}\,y^m(t-1),\quad y^{out}(t)=f_{out}(net_{out}(t))$$

$$net_{\varphi}(t)=\sum_m w_{\varphi\,m}\,y^m(t-1),\quad y^{\varphi}(t)=f_{\varphi}(net_{\varphi}(t))$$

$$net_{in}(t)=\sum_m w_{in\,m}\,y^m(t-1),\quad y^{in}(t)=f_{in}\,(net_{in}(t))$$

Cell net-input equation

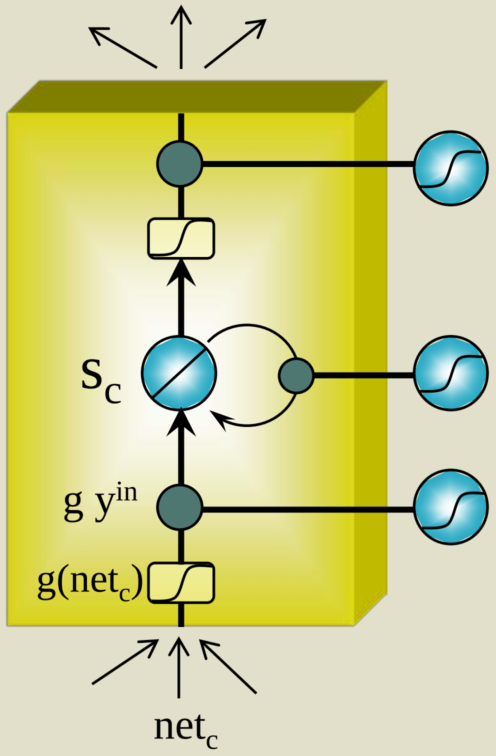
$$net_c(t) = \sum_m w_{cm}\, y^m(t-1)$$



Cell state equation

$$S_c(t) = y^{\varphi}(t)\ S_c(t-1)\ +\ y^{in}(t)\ g\left(net_c(t)\right)$$

$$net_c(t) = \sum_m w_{cm}\, y^m(t-1)$$

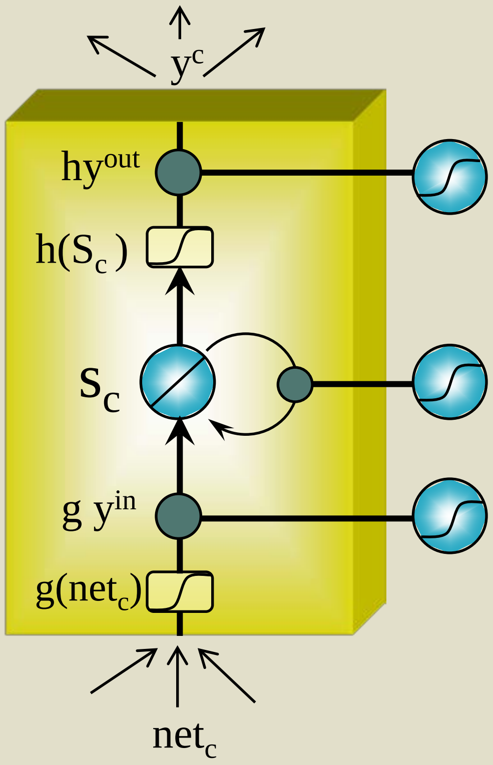


Cell equations

$$y^c(t) = y^{out}(t)\ h\big(S_c(t)\big)$$

$$S_c(t) = y^{\varphi}(t)\ S_c(t-1)\ +\ y^{in}(t)\ g\big(net_c(t)\big)$$

$$net_c(t) = \sum_m w_{cm}\ y^m(t-1)$$



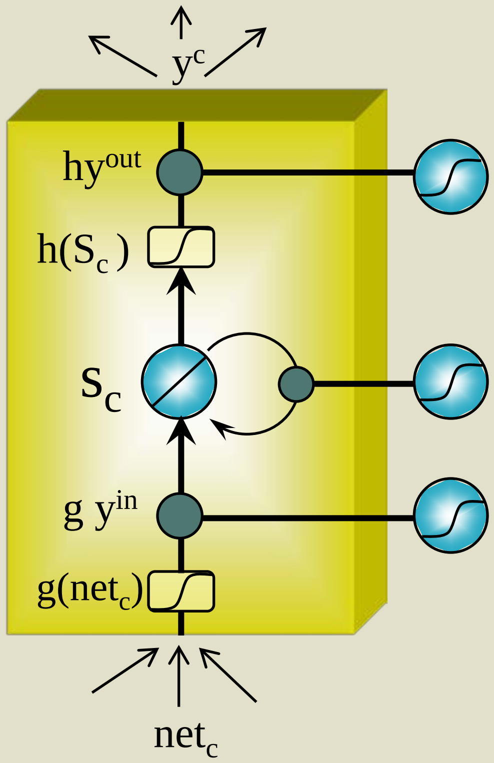
Global net output

$$net_k(t) = \sum_m w_{km} y^m(t-1), \quad y^k(t) = f_k(net_k(t))$$

$$y^c(t) = y^{out}(t)\ h(S_c(t))$$

$$S_c(t) = y^{\varphi}(t)\ S_c(t-1)\ +\ y^{in}(t)\ g(net_c(t))$$

$$net_c(t) = \sum_m w_{cm}\ y^m(t-1)$$



Learning

- Backward pass

  a combination of the truncated back-propagation through time (BPTT) and a modified version of real time recurrent learning (RTRL)

Recurrent neural networks applications

- Channel equalization
- Speech recognition
- Speech coding
- System identification and control
- Natural language processing
- Temporal sequence prediction

Temporal sequence prediction

- The sample $u[t]$ is given to the net
- The desired output is $u[t+1]$
- After a correct learning the output of the network will be an estimation of next sample value

Training algorithms

- Derivative based algorithms
    - Gradient descent
    - Decoupled extended Kalman filter (DEKF)
    - ...
- Non-derivative algorithms
    - Breeder Genetic Algorithm (BGA)
    - ...

Derivative based methods

- Compute the error derivative function with respect to the different net parameters (weights and biases)

$$\frac{\partial E[t]}{\partial w}$$

- Methods for gradient calculation
  - Real time recurrent learning (RTRL)
  - Backpropagation through time (BPTT)
  - Mixed methods (like those used by LSTM nets)
  - ...

Gradient descent algorithms

- Descent through the error hyper-surface in direction opposite to the gradient:

$$\Delta w_i[t] = -\alpha \frac{\partial E[t]}{\partial w_i} \qquad \Delta w_i[t] = -\alpha \frac{\partial E[t]}{\partial w_i} + \gamma \, \Delta w_i[t-1]$$

- Gradient descent algorithms are usually slow because they only use the last estimation of the gradient; they do not take into account information about past history

Kalman filters (KF)

- Kalman filter algorithms recursively and efficiently consider all the information computed until now
  - Extended Kalman filter (EKF)
    - An adaptation of Kalman Filters that allows work with nonlinear systems
  - Decoupled extended Kalman filter
    - An adaptation of EKF suitable for neural network training that reduces global EKF complexity

Breeder genetic algorithm (BGA)

- The breeder genetic algorithm (BGA) is similar to the standard genetic algorithm
  - Truncation selection
  - Recombination
  - Mutation
  - Replacement

Developed research

- **Goals**
  - Investigate the LSTM ability to learn in presence of numerical temporal sequences
  - To improve learning and generalization abilities of LSTM net by applying growing heuristics (GLSTM)

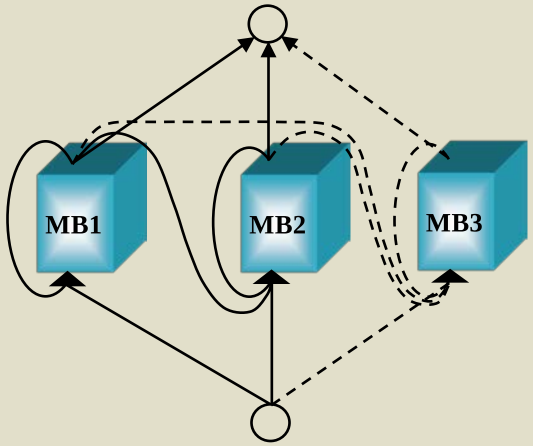
GLSTM growing configuration

- LSTM starts training with 1 Memory block and grows by inserting blocks
- An important problem is how to set the weight connections of a newly added block
- Two ways of inserting blocks:
  - Cascade
    - Cascade freezing
    - Cascade trainable output
    - Cascade not-freezing
  - Fully connected (no weights are frozen)

Experimental configurations

- **Cascade freezing**

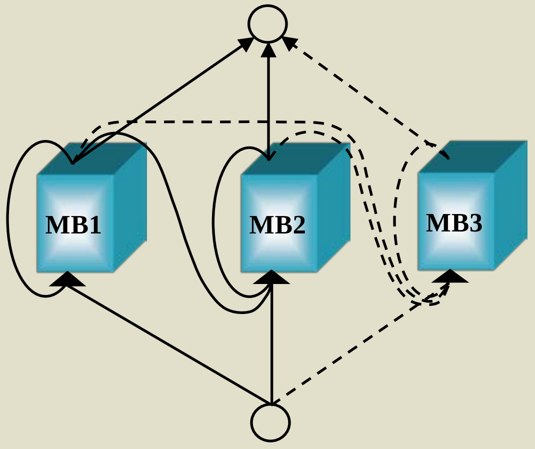
  All pre-existing weights are frozen

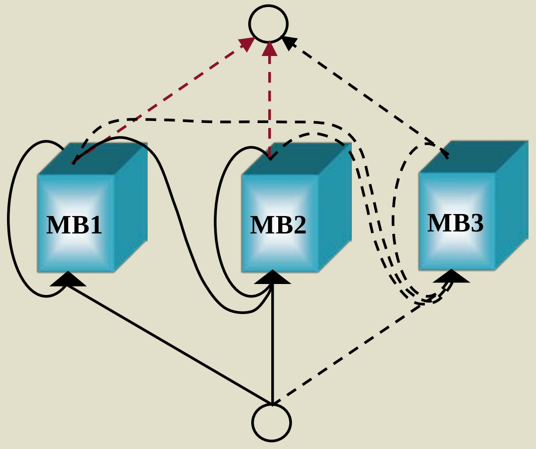


Experimental configurations

- **Cascade freezing but trainable-output**

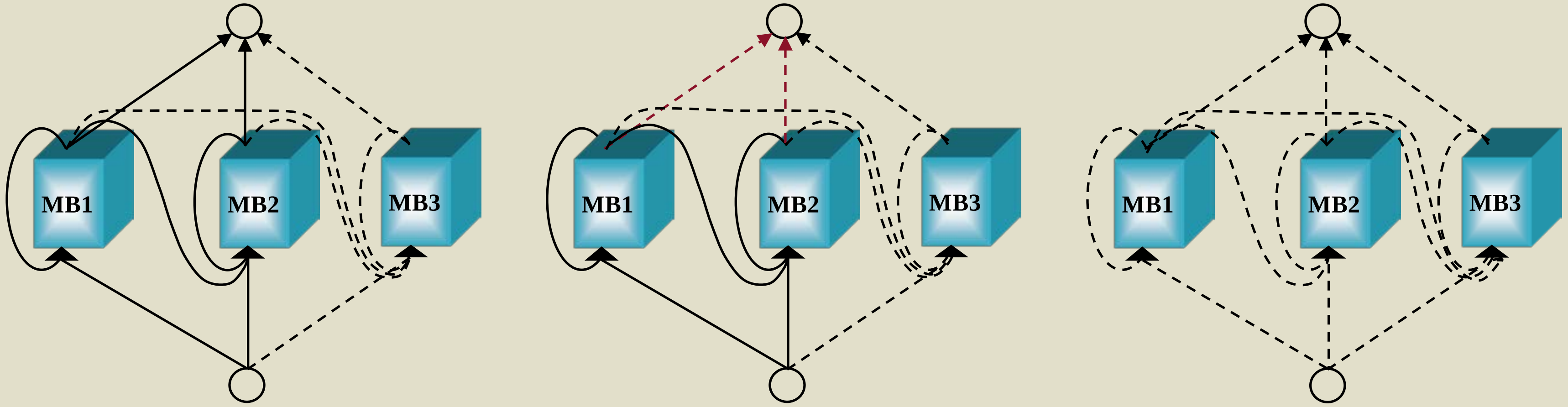
  Only weights arriving at already existing blocks are frozen





Experimental configurations

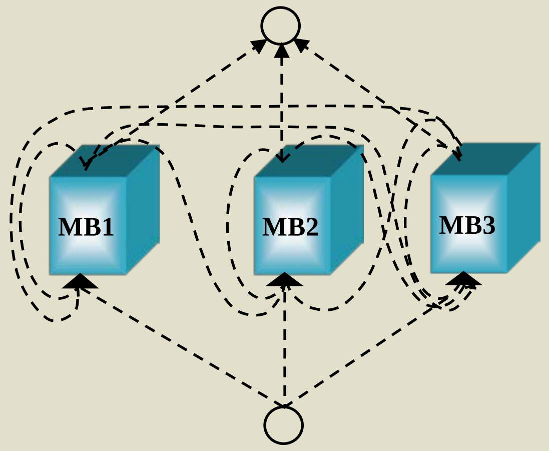
- **Cascade freezing** (all pre-existing weights are frozen)
- **Cascade freezing but trainable output** (only weights arriving at already existing blocks are frozen)
- **Cascade not-freezing** (no weight is frozen)



Experimental configurations

- **Fully connected**

  Receives connections from every pre-existing blocks and also from the new connected block, no weight is frozen



Experimental evaluation

- Objectives
  - Carry out experiments in two different domains
    - Forecasting of maximum ozone concentration
    - Identification of central nervous system controllers (CNSC) using GLSTM
  - Compare the LSTM performance with those obtained previously using TDNNs and RNNs
  - To validate LSTM extended with growing abilities (GLSTM)

Identification of central nervous system controllers (CNSC) using GLSTM

- Signals were extracted from a simulation of the cardiovascular system
- For each one of the 5 controllers:
  - 1 training set: 1,500 points
  - 6 test data sets: 300 points each
- Error measure is given by:

$$NMSE = \frac{E\left[\left(t_g(t) - \hat{y}(t)\right)^2\right]}{t_{g\,\mathrm{var}}} * 100\%$$

Stepwise prediction results

- Train and test set average NMSE errors (in percent) when using the four configurations

	FULLY		CASCADE FREEZING		CASCADE NOT-FREEZING		CASCADE TRAINABLE-OUT	
	Train	Test	Train	Test	Train	Test	Train	Test
HRC	4.85	3.99	3.09	4.11	3.26	4.15	2.11	3.27
PRC	0.12	0.69	0.13	0.66	0.17	1.33	0.09	0.61
MCC	0.39	1.11	0.14	0.99	0.11	1.02	0.09	0.98
VTC	0.08	0.96	0.10	0.96	0.18	1.05	0.08	0.96
CRC	0.22	0.37	1.11	0.35	0.18	0.38	0.09	0.26
Av. Error	1.35	1.42	0.71	1.41	0.78	1.58	0.48	1.21

Stepwise prediction results

- Average NMSE errors for stepwise prediction on training and test sets

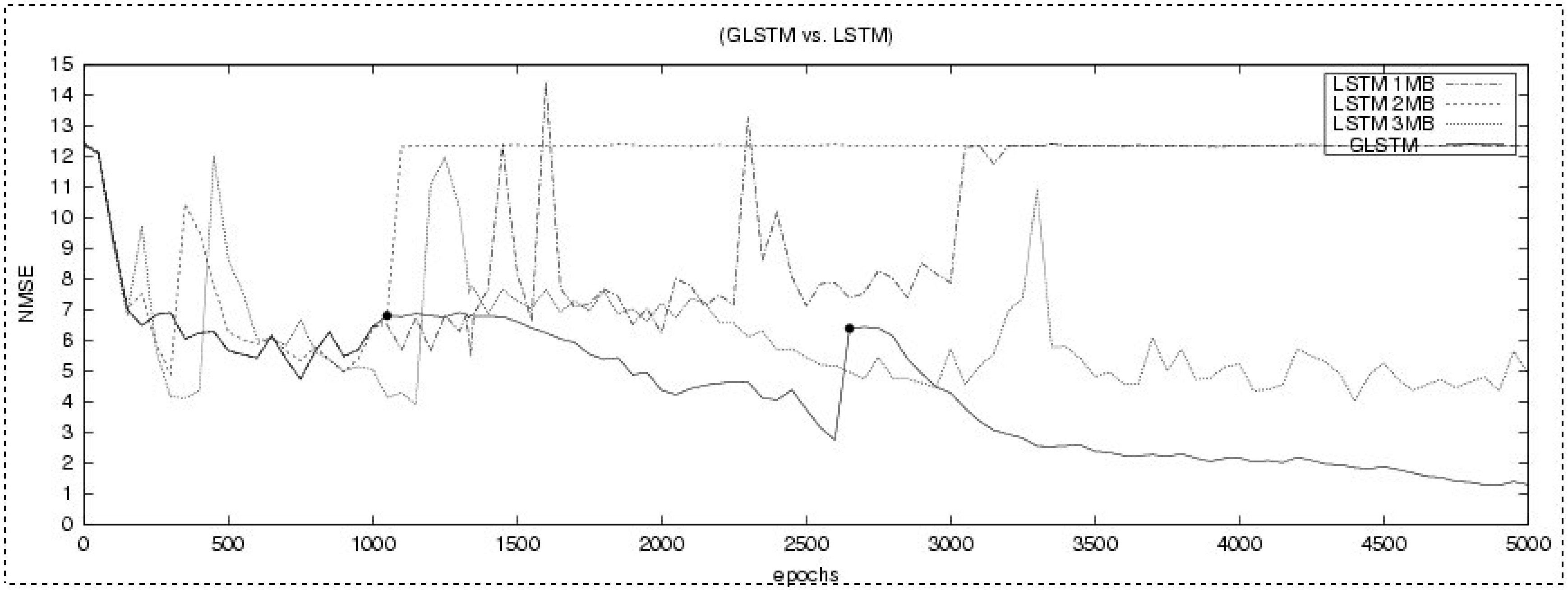
	TD-HNN		TDNN-BP		TDNN-AC		ASLRNN		LSTM		GLSTM	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
HRC	0.11	0.18	1.15	1.52	0.15	0.13	1.63	1.91	2.16	3.41	2.11	3.27
PRC	0.09	0.12	0.94	1.27	0.26	0.14	0.84	1.10	0.25	0.65	0.09	0.61
MCC	0.03	0.06	0.81	1.33	0.09	0.08	0.71	1.18	0.19	1.04	0.09	0.98
VTC	0.03	0.06	0.81	1.33	0.09	0.08	0.71	1.18	0.19	1.01	0.08	0.96
CRC	0.10	0.11	0.47	0.66	0.03	0.04	0.41	0.53	0.18	0.31	0.09	0.26
Av. Error	0.07	0.11	0.84	1.22	0.12	0.09	0.86	1.18	0.59	1.28	0.48	1.21

TD-HNN : Time-Delay Heterogeneous Neural Network
TDNN-BP : Time-Delay Neural Network trained by Back-Propagation
TDNN-AC : Time-Delay Neural Network trained by Annealing and Conjugate Gradient

ASLRNN : Augmented Single Layer Recurrent Neural Network
LSTM : Long-Short Term Memory Recurrent Neural Network
GLSTM : Cascade Trainable Output
* All errors are given in %

Graphical results

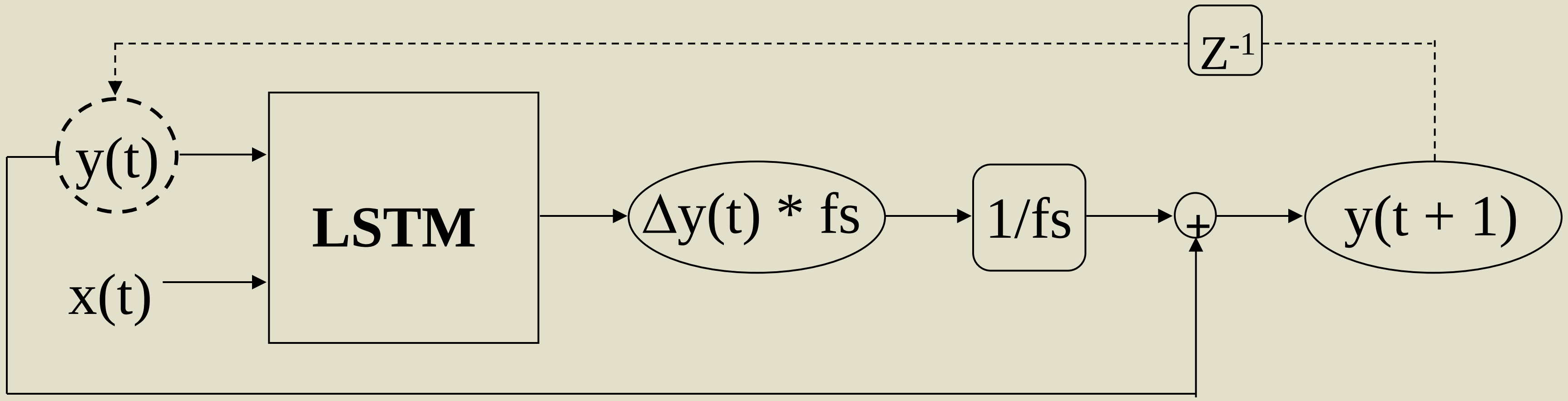
- Error curve of LSTM trained with 1, 2 and 3 memory blocks and trained with cascade growing (GLSTM)



Long-term predictions

- Iterated prediction of output using GLSTM

  for ***T* = *n*** the predicted values are fed back ***n* - 1** times



Long-term prediction results

- Average NMSE errors of long-term prediction on test sets

	TDNN-BP	ASLRNN	LSTM	GLSTM
HRC	15.35 %	18.31 %	28.78 %	24.05 %
PRC	33.76 %	31.16 %	14.04 %	21.95 %
MCC	34.04 %	35.16 %	26.35 %	22.36 %
VTC	34.04 %	34.77 %	22.02 %	18.89 %
CRC	55.69 %	57.12 %	14.73 %	13.65 %
Av. Error	34.58 %	35.30 %	21.18 %	20.18 %

CNSC experiments conclusion

- CNSC task
  - Single-step prediction

    similar to the one provided by common RNN approaches, but worse than the one obtained by training TDNNs

  - Iterated long-term prediction

    better than those of ASLRNN and TDNN-BP, but not very satisfactory indeed

Discussion

- LSTM could not achieve as good results as those achieved by other neural nets (TDNNs) in temporal sequence prediction tasks
- Why?
  - Lack of long term dependences
  - Prediction of numerical sequences
- How to improve LSTM performance in temporal prediction tasks?
- Is LSTM really useful for a given problem?

Hybrid Growing TDNN-RNN-LSTM

- To develop a hybrid methodology for the incremental construction of recurrent neural networks
- Aiming to find the optimal topology for a given task
- By combining the benefits of
  - TDNNs (feedforward neurons)
  - RNNs (recurrent neurons)
  - LSTM (memory blocks)

Hybrid Growing TDNN-RNN-LSTM

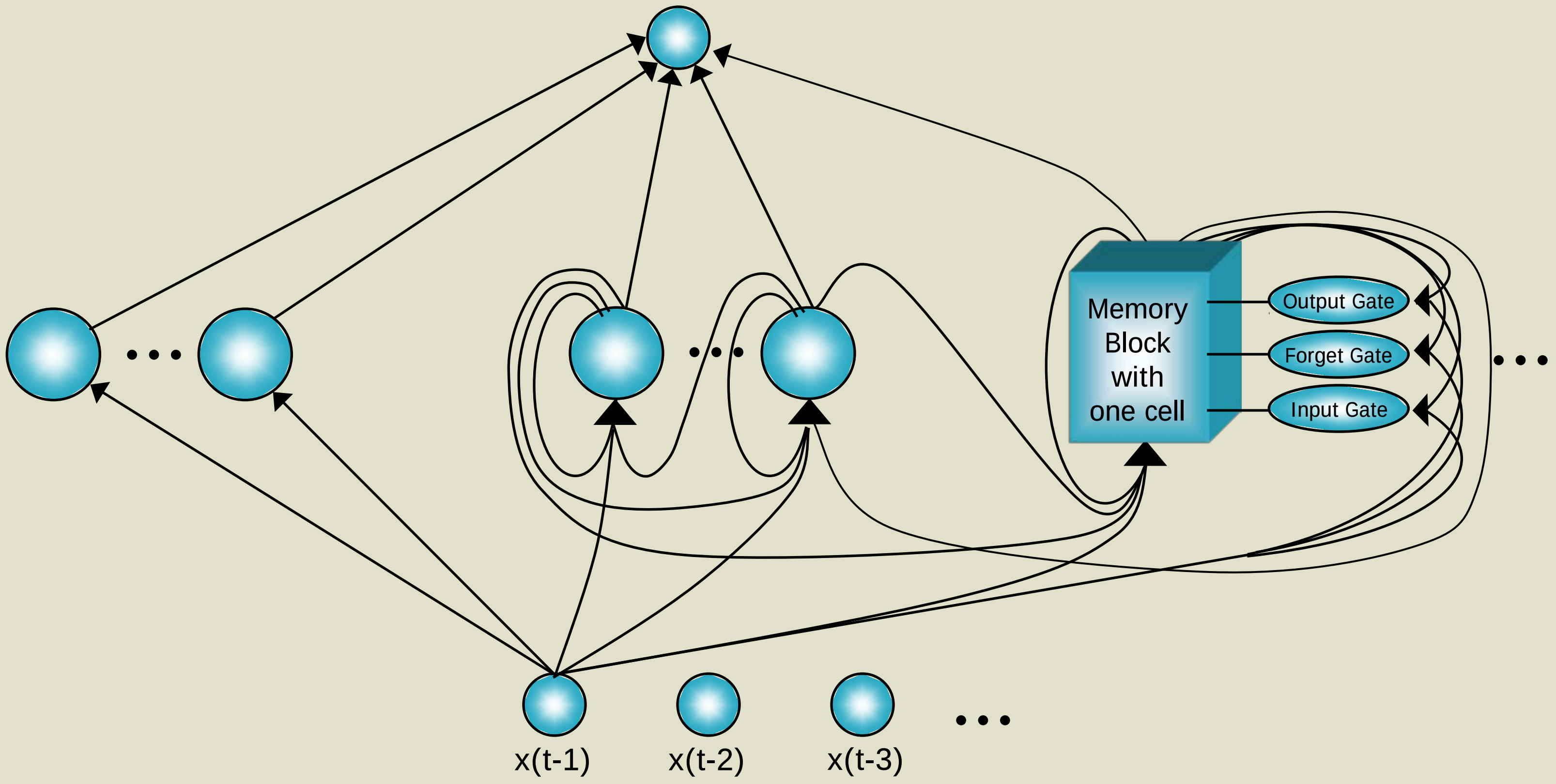
- Selection of the initial TDNN topology by means of:
  - Growing strategies

    study of criteria to decide about the addition of more input units (longer delays)

  - Feature selection

    select the time delays from a given fixed size temporal window of input values

Hybrid Growing TDNN-RNN-LSTM



Training algorithms

- Gradient descent
  - Classical
- Decoupled extended Kalman filter
  - Very good results already achieved with neural networks, including LSTM
- Breeder genetic algorithm
  - Already used in growing strategies with feedforward networks
  - Direct continuous variable representation - no need to codify weights