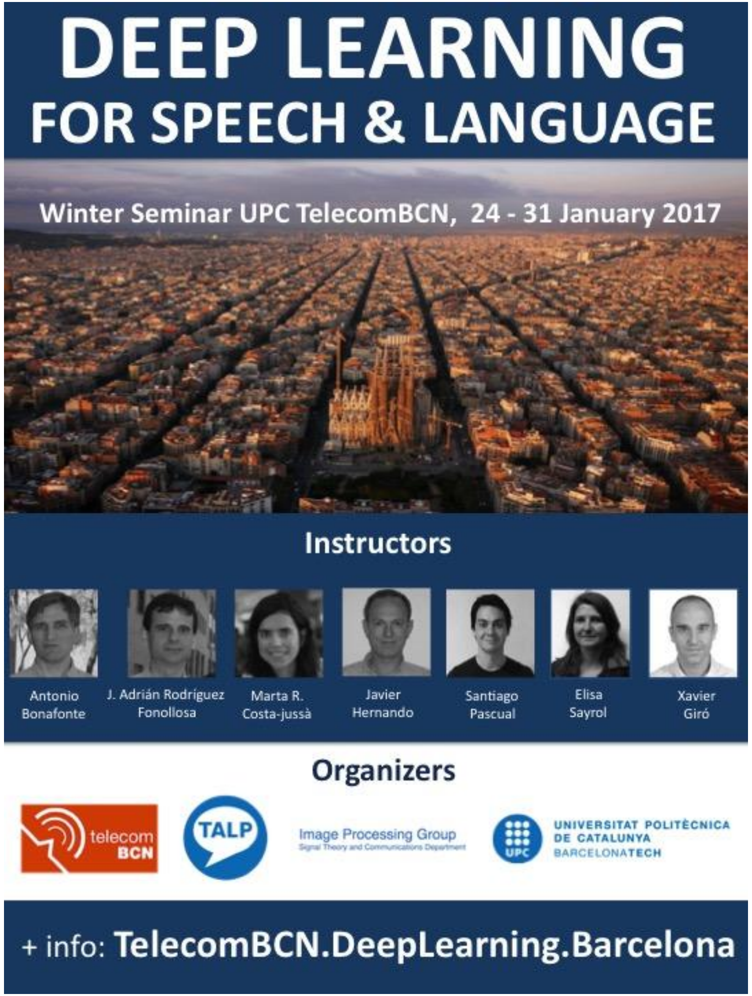


[course site]

Day 2 Lecture 2

Recurrent Neural Networks I

Santiago Pascual

Outline

1. The importance of context
2. Where is the memory?
3. Vanilla RNN
4. Problems
5. Gating methodology
    a. LSTM
    b. GRU

The importance of context

- Recall the 5th digit of your phone number
- Sing your favourite song beginning at third sentence
- Recall 10th character of the alphabet

Probably you went straight from the beginning of the stream in each case…

because in sequences order matters!

Idea: retain the information preserving the importance of order

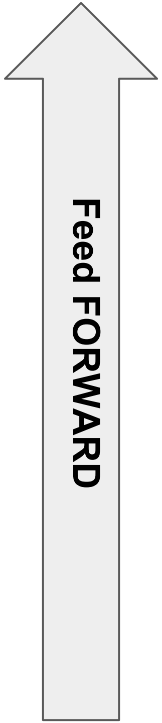
Recall from Day 1 Lecture 2...

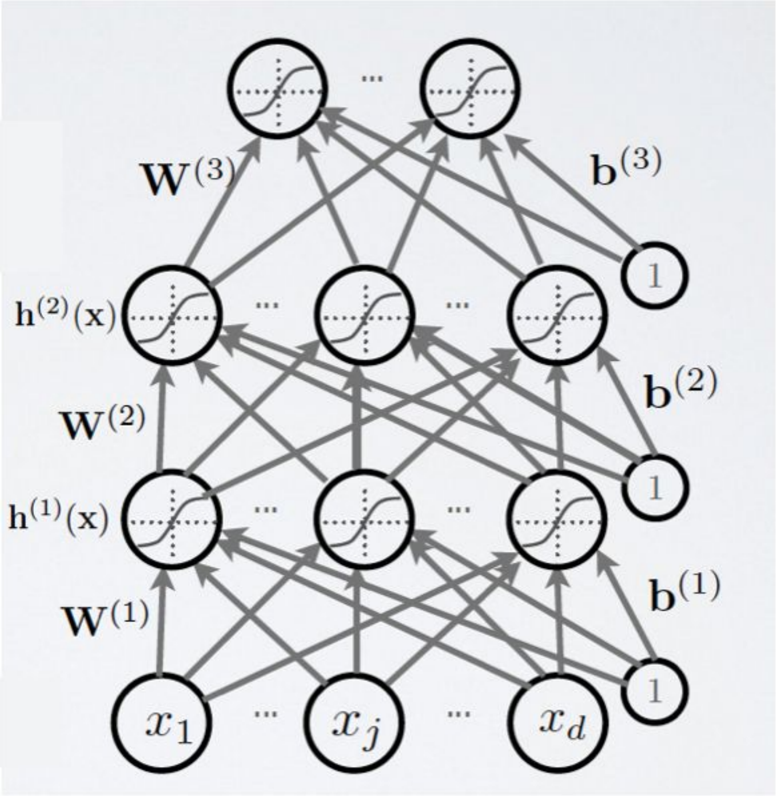
Every **y/hi** is computed from the sequence of forward activations out of input **x**.

$$y = f(W_3 \cdot h_2 + b_3)$$

$$h_2 = f(W_2 \cdot h_1 + b_2)$$

$$h_1 = f(W_1 \cdot x + b_1)$$

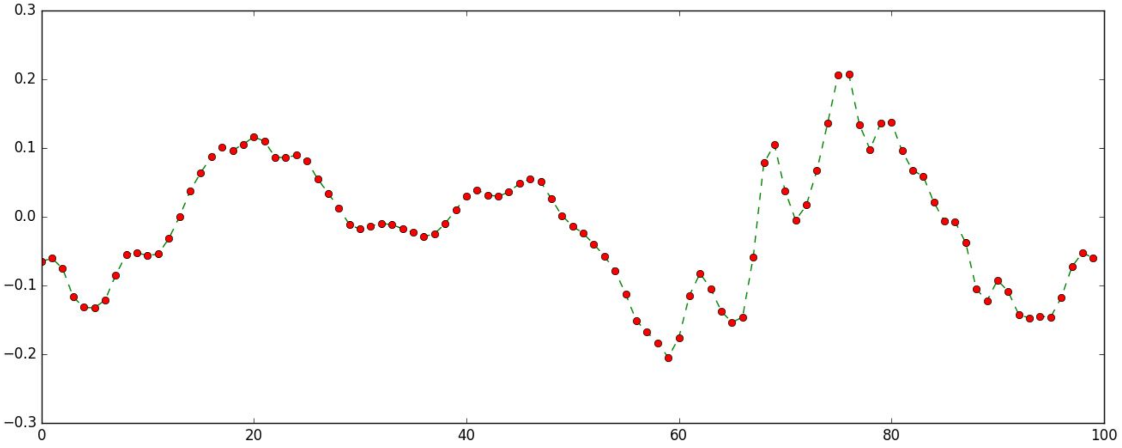




Slide Credit: Hugo Laroche NN course

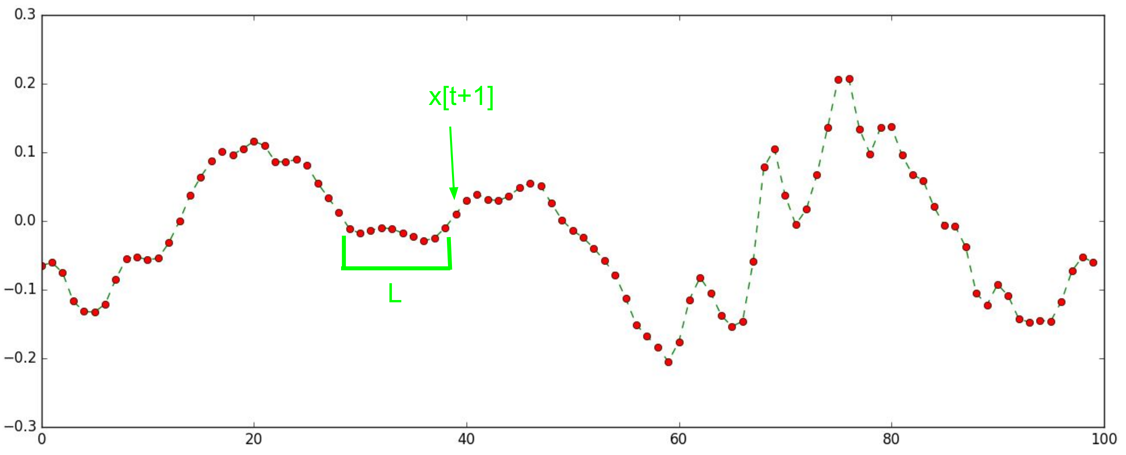
Where is the Memory?

If we have a sequence of samples...



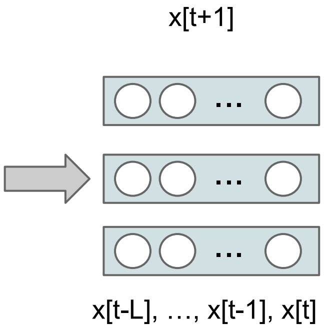
predict sample x[t+1] knowing previous values {x[t], x[t-1], x[t-2], …, x[t-τ]}

Where is the Memory?

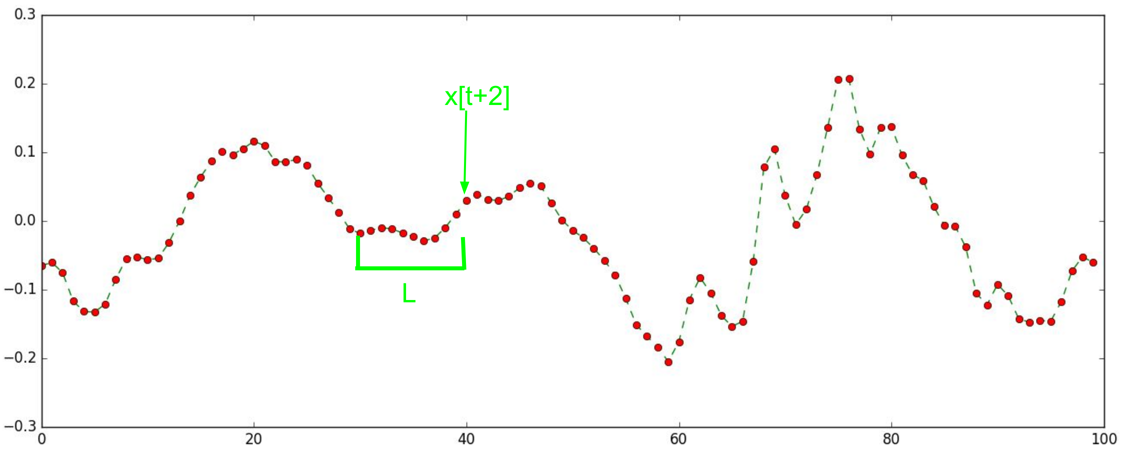


Feed Forward approach:

- static window of size L
- slide the window time-step wise

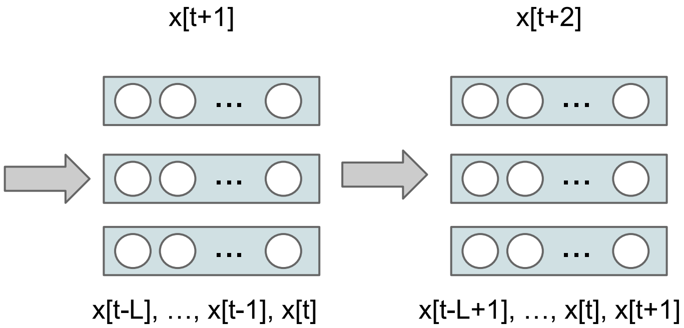


Where is the Memory?

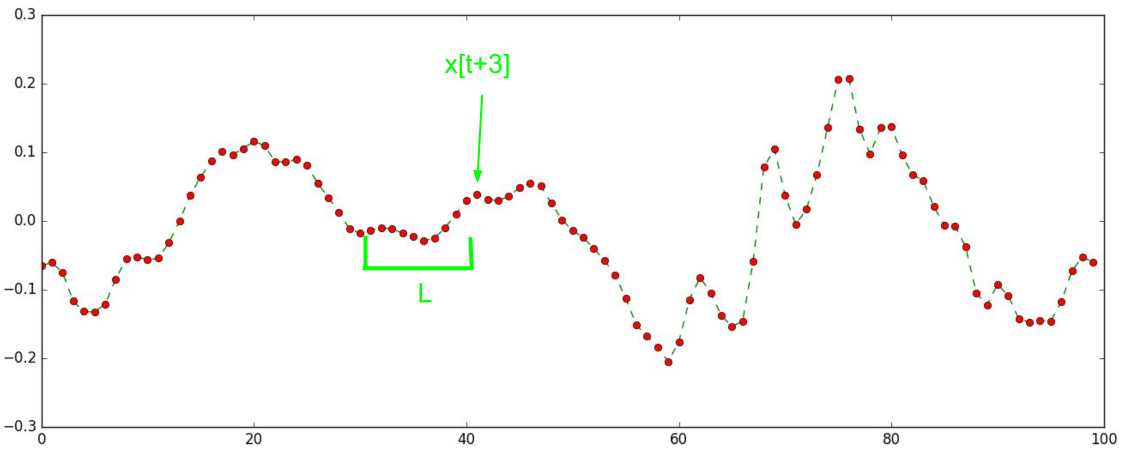


Feed Forward approach:

- static window of size L
- slide the window time-step wise

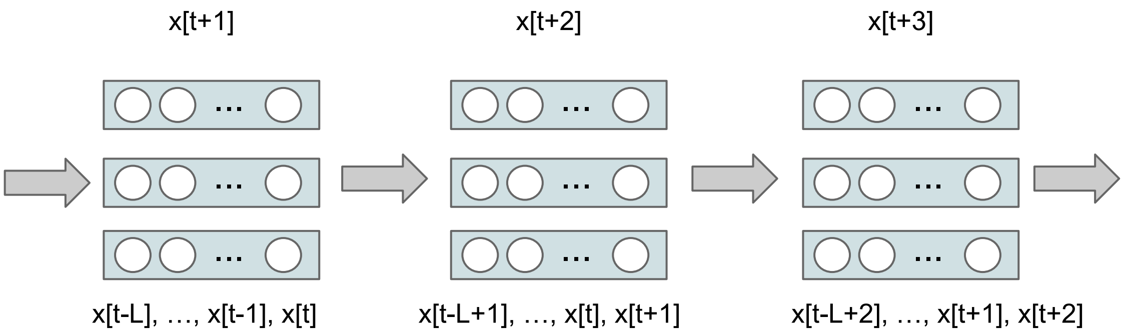


Where is the Memory?



Feed Forward approach:

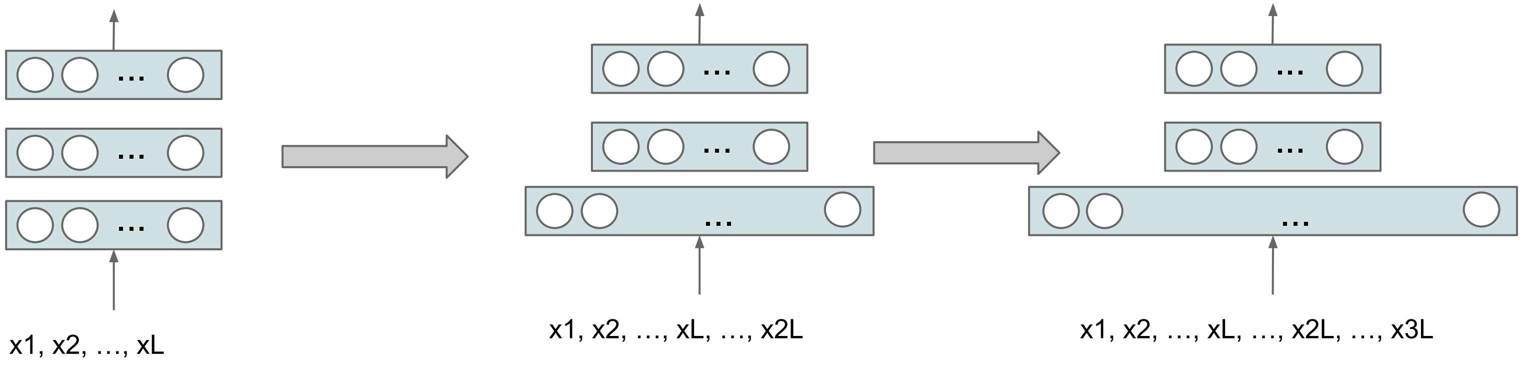
- static window of size L
- slide the window time-step wise



Where is the Memory?

Problems for the feed forward + static window approach:

- What's the matter increasing L? → Fast growth of num of parameters!
- Decisions are independent between time-steps!
    - The network doesn't care about what happened at previous time-step, only present window matters → doesn't look good
- Cumbersome padding when there are not enough samples to fill L size
    - Can't work with variable sequence lengths



Recurrent Neural Network

Solution: Build specific connections capturing the temporal evolution → **Shared weights in time**

- Give volatile memory to the network

$$\boxed{h_t = f(W \cdot x_t + U \cdot h_{t-1} + b)}$$

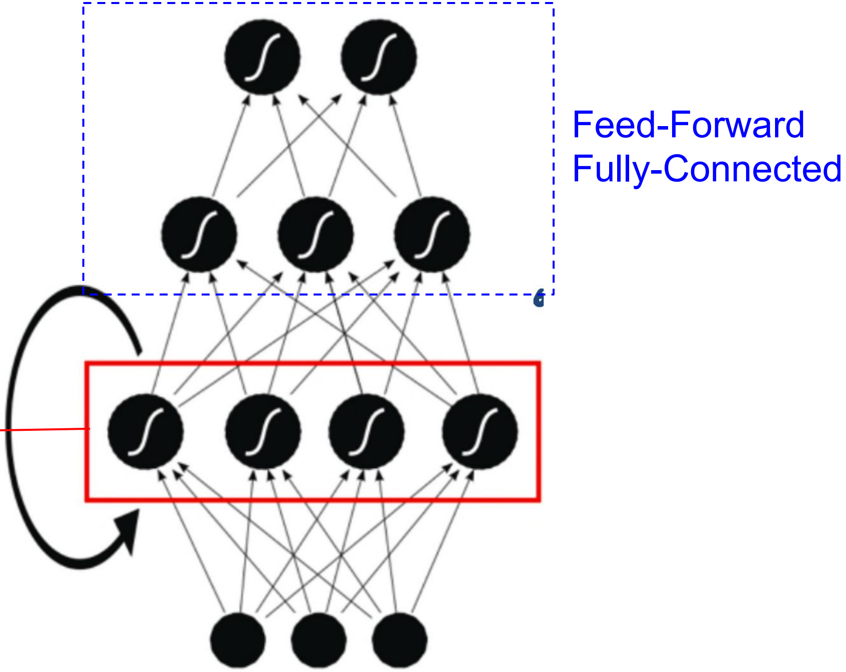


Figure credit: Xavi Giro

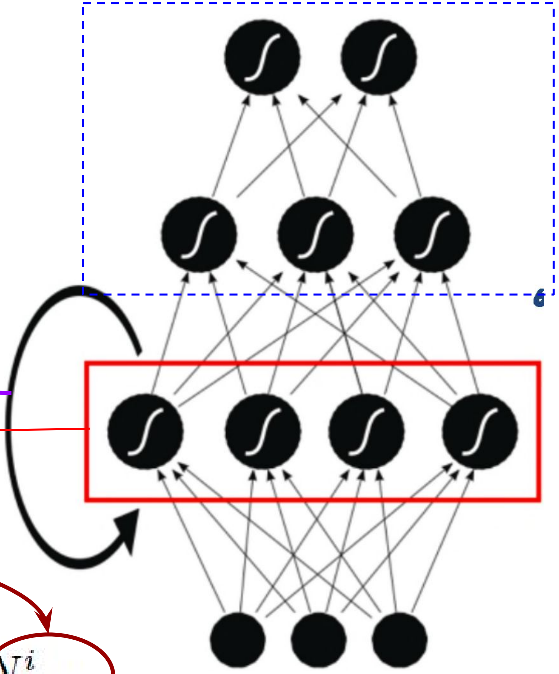
Recurrent Neural Network

Solution: Build specific connections capturing the temporal evolution → **Shared weights in time**

- Give volatile memory to the network

Fully connected in time: recurrent matrix

$$\boxed{h_t = f(W \cdot x_t + U \cdot h_{t-1} + b)}$$

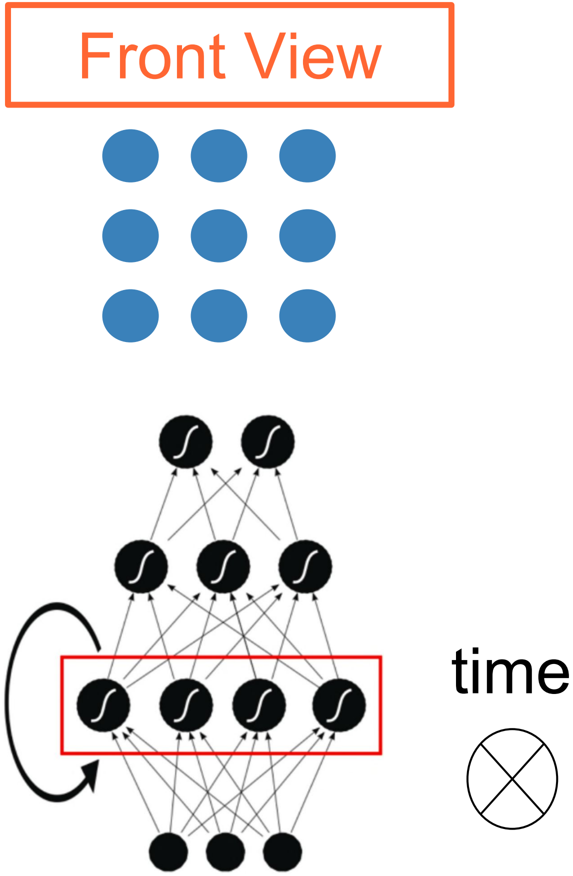


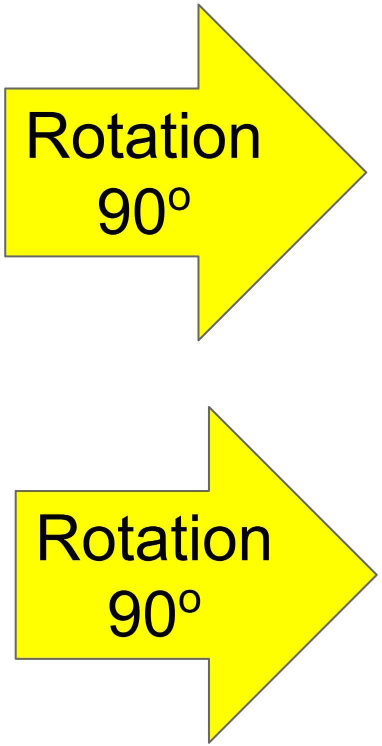
Feed-Forward Fully-Connected

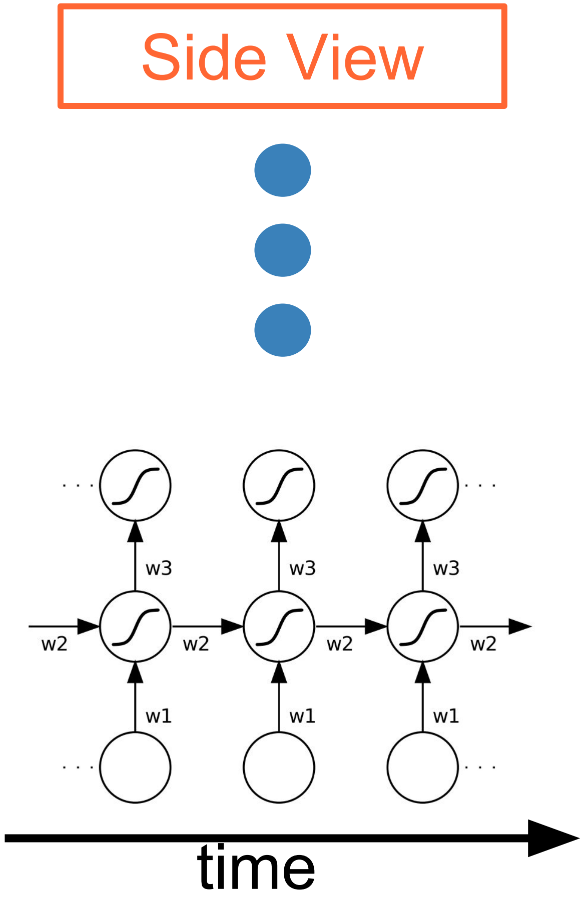
$$N^i_{params} = N^i_{inputs} \times N^i_{units} + N^i_{units} \times N^i_{units} + N^i_{units}$$

Figure credit: Xavi Giro

Recurrent Neural Network





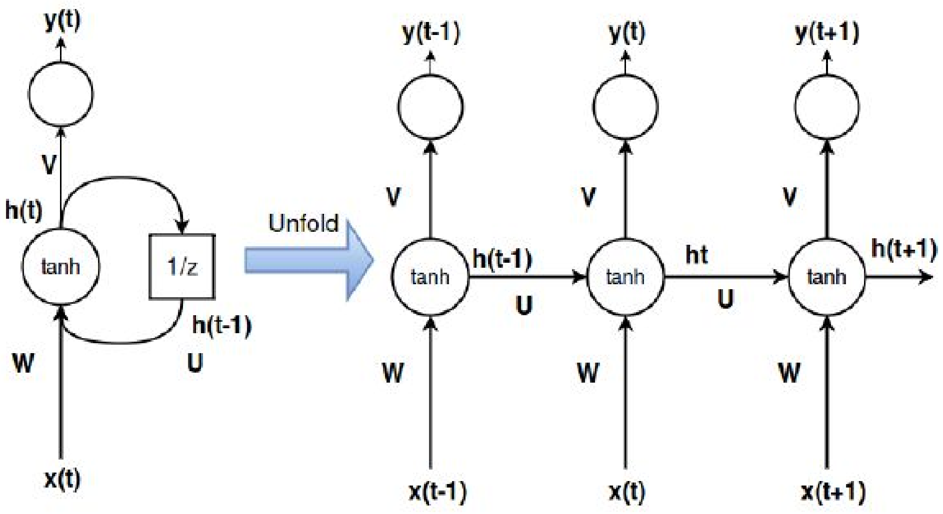


Slide credit: Xavi Giro

Recurrent Neural Network

Hence we have two data flows: **Forward in space + time** propagation: **2 projections per layer activation.**

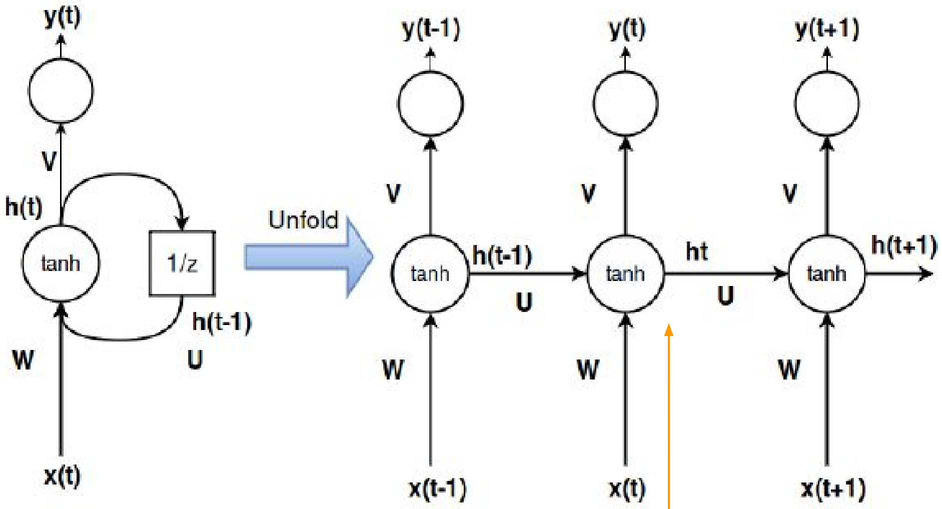
BEWARE: We have extra depth now! Every time-step is an extra level of depth (as a deeper stack of layers in a feed-forward fashion!)



Recurrent Neural Network

Hence we have two data flows: **Forward in space + time** propagation: **2 projections per layer activation**

- Last time-step includes the context of our decisions recursively



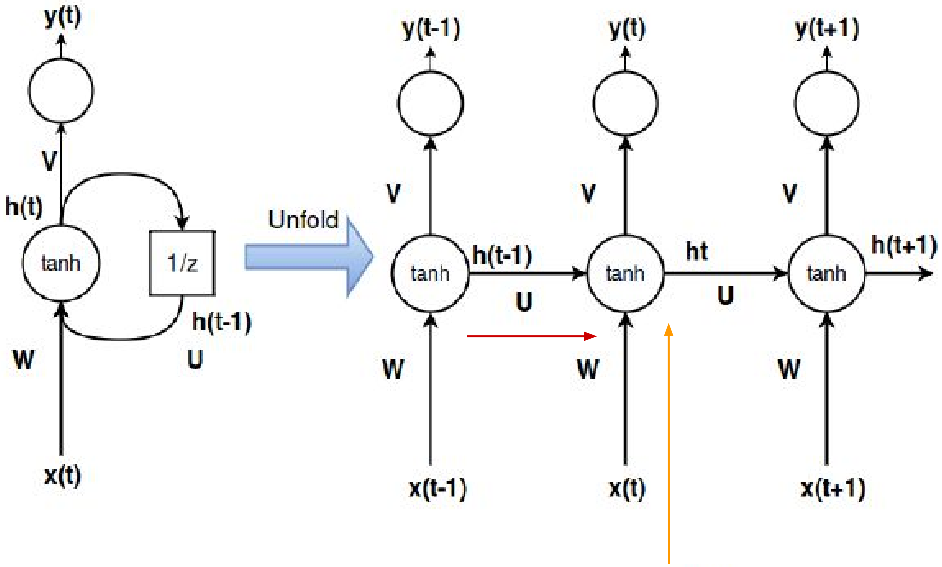
$\mathbf{x}_t \in \mathbb{R}^n$

$$\mathbf{h}_t = g(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h)$$

Recurrent Neural Network

Hence we have two data flows: **Forward in space + time** propagation: **2 projections per layer activation**

- Last time-step includes the context of our decisions recursively



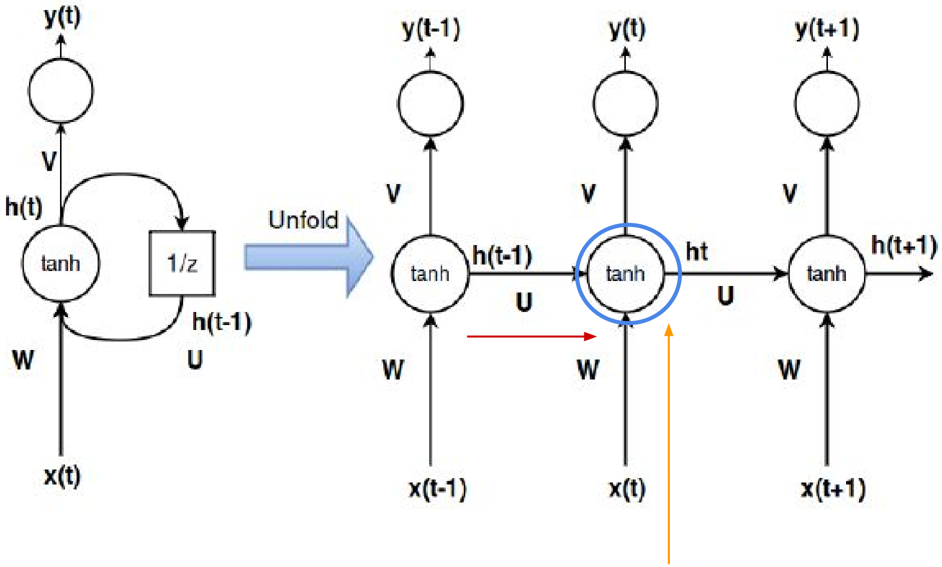
$$\mathbf{x}_t \in \mathbb{R}^n$$

$$\mathbf{h}_t = g(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h)$$

Recurrent Neural Network

Hence we have two data flows: **Forward in space + time** propagation: **2 projections per layer activation**

- Last time-step includes the context of our decisions recursively



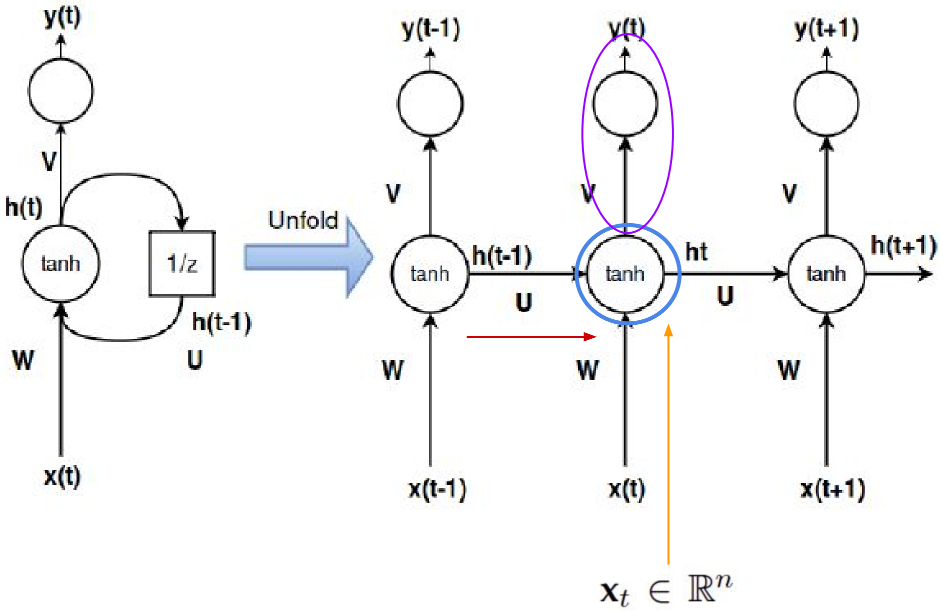
$$\mathbf{h}_t = g(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h)$$

$$\mathbf{x}_t \in \mathbb{R}^n$$

Recurrent Neural Network

Hence we have two data flows: **Forward in space + time** propagation: **2 projections per layer activation**

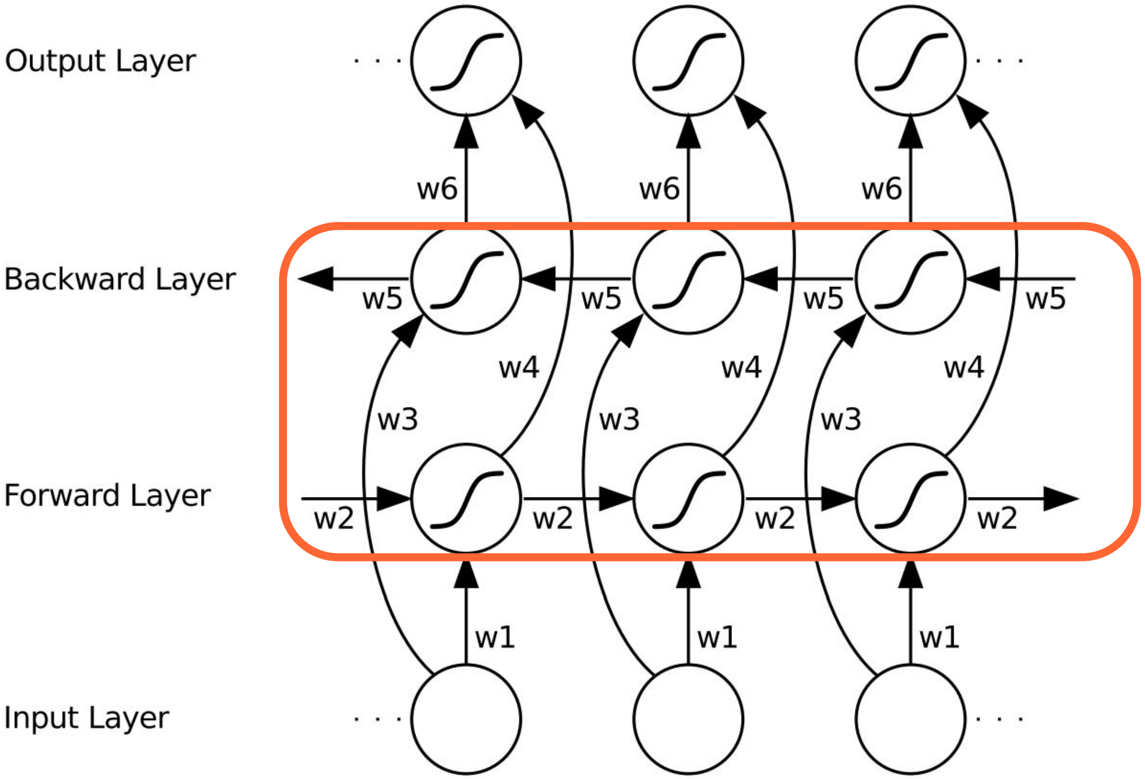
- Last time-step includes the context of our decisions recursively



$$\mathbf{y}_t = f(\mathbf{V} \cdot \mathbf{h}_t + \mathbf{b}_v)$$

$$\mathbf{h}_t = g(\mathbf{W} \cdot \mathbf{x}_t + \mathbf{U} \cdot \mathbf{h}_{t-1} + \mathbf{b}_h)$$

Bidirectional RNN (BRNN)



Must learn weights w2, w3, w4 & w5; in addition to w1 & w6.

Alex Graves, “Supervised Sequence Labelling with Recurrent Neural Networks”

Recurrent Neural Network

Back Propagation Through Time (BPTT): The training method has to take into account the time operations → a cost function ***E*** is defined to train our RNN, and in this case the total error at the output of the network is the sum of the errors at each time-step:

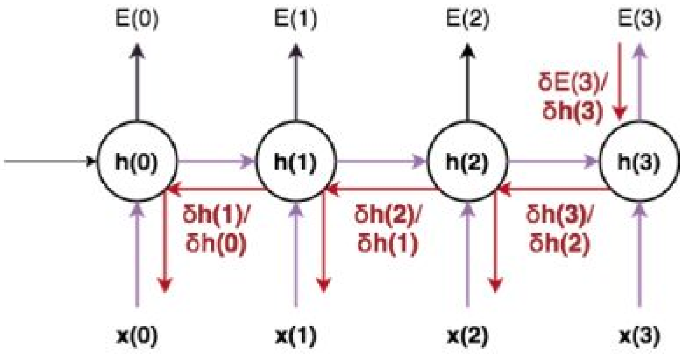
$$E(\mathbf{y}, \hat{\mathbf{y}}) = \sum_{t=1}^{T} E_t(\mathbf{y}_t, \hat{\mathbf{y}}_t)$$

$$\frac{\partial E}{\partial \mathbf{W}} = \sum_{t=0}^{T-1} \frac{\partial E_t}{\partial \mathbf{W}}$$

T: **max amount of time-steps to do back-prop**. In Keras this is specified when defining the "input shape" to the RNN layer, by means of:
(batch size, sequence length (T), input_dim)

Input shape

3D tensor with shape `(nb_samples, timesteps, input_dim)`.



Example back-prop in time with 3 time-steps

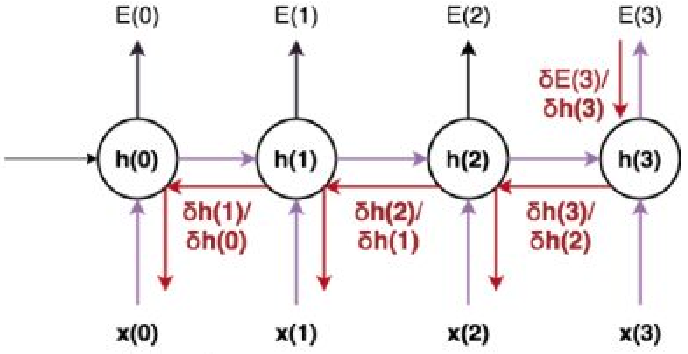
Recurrent Neural Network

Main problems:

- **Long-term memory** (remembering quite far time-steps) **vanishes quickly** because of the recursive operation with **U**

$$\mathbf{h}_t = g(\mathbf{W}\cdot\mathbf{x}_t + \mathbf{U}\cdot g(\cdots g(\mathbf{W}\cdot\mathbf{x}_{t-T} + \mathbf{U}\cdot\mathbf{h}_{t-T} + \mathbf{b}_h)\cdots) + \mathbf{b}_h)$$

- **During training gradients explode/vanish easily because of depth-in-time** → Exploding/Vanishing gradients!



Example back-prop in time with 3 time-steps

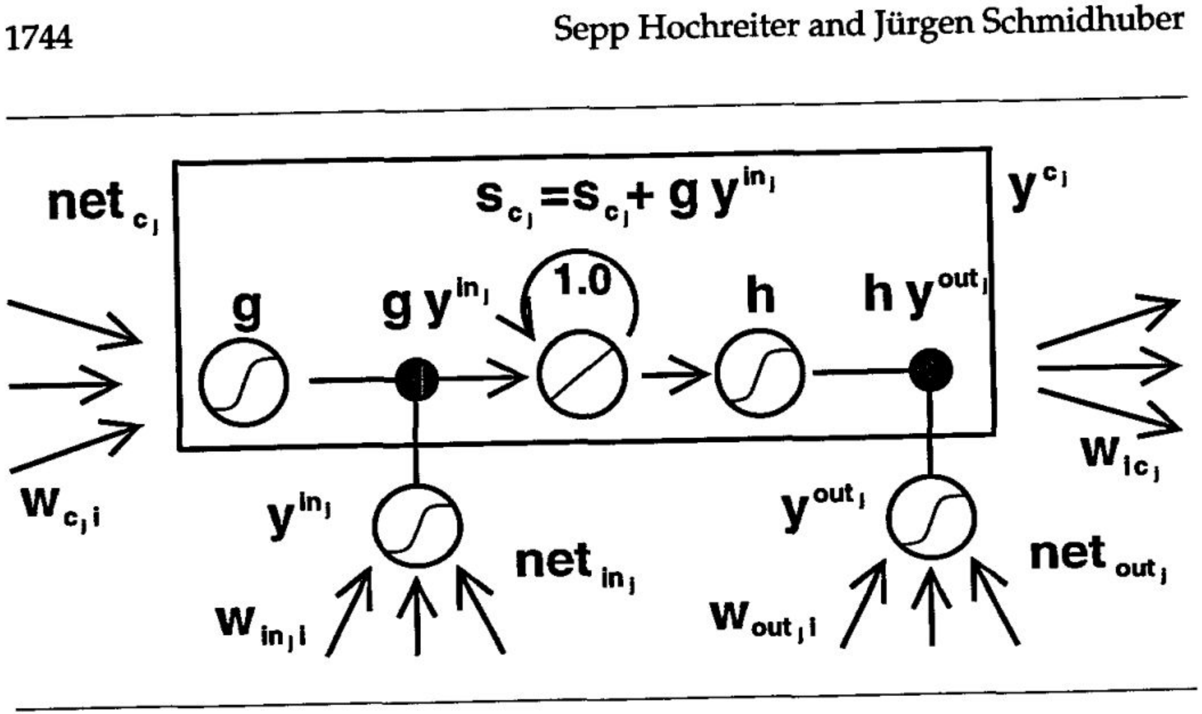
Gating method

Solutions:

1. Change the way in which past information is kept → create the notion of **cell state, a memory unit that keeps long-term information in a safer way by protecting it from recursive operations**
2. **Make every RNN unit able to decide whether the current time-step information matters or not**, to accept or discard (optimized reading mechanism)
3. **Make every RNN unit able to forget whatever may not be useful anymore** by clearing that info from the cell state (optimized clearing mechanism)
4. **Make every RNN unit able to output the decisions whenever it is ready to do so** (optimized output mechanism)

Long Short-Term Memory (LSTM)

slide credit: Xavi Giro

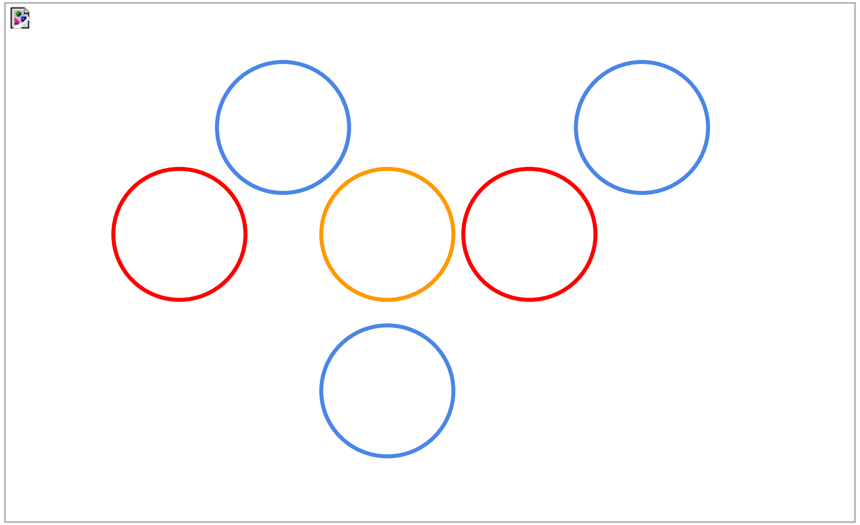


Hochreiter, Sepp, and Jürgen Schmidhuber. ["Long short-term memory."](#) Neural computation 9, no. 8 (1997): 1735-1780.

Long Short Term Memory (LSTM) cell

An LSTM cell is defined by two groups of neurons plus the cell state (memory unit):

1. Gates
2. Activation units
3. Cell state



$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i)$$

$$\hat{\mathbf{C}}_t = \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f)$$

$$\mathbf{C}_t = \mathbf{i}_t \odot \hat{\mathbf{C}}_t + \mathbf{f}_t \odot \mathbf{C}_{t-1}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{C}_t)$$

Computation Flow

Long Short-Term Memory (LSTM)

Three **gates** are governed by *sigmoid* units (btw [0,1]) define the control of in & out information..

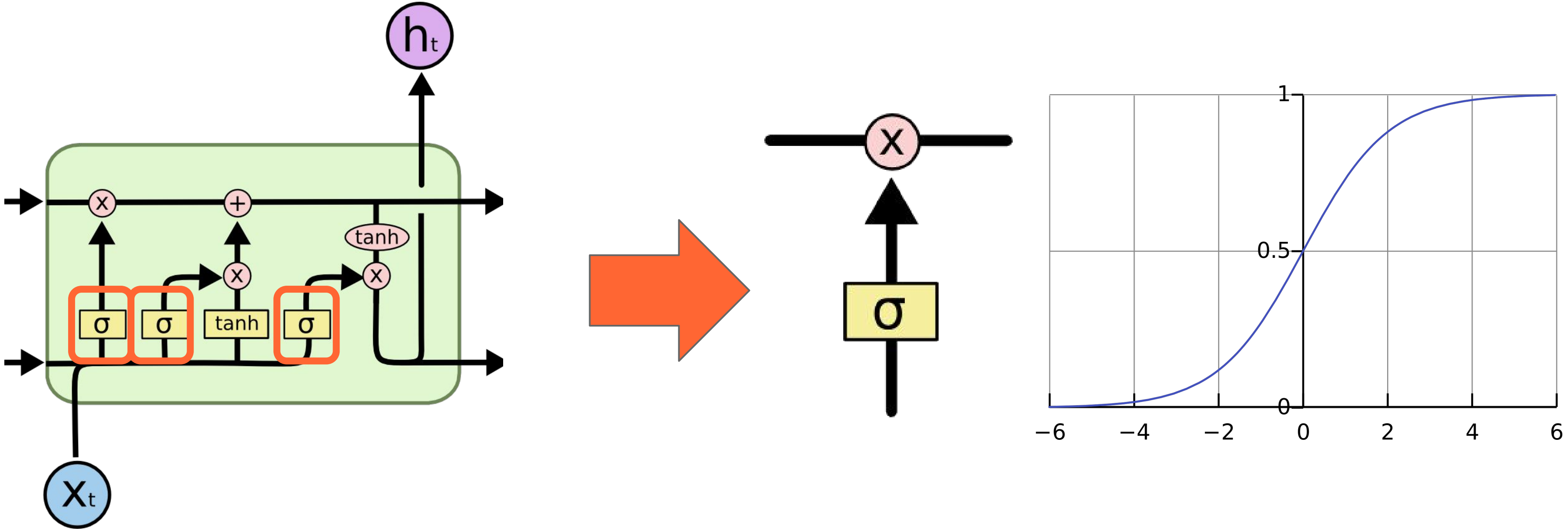
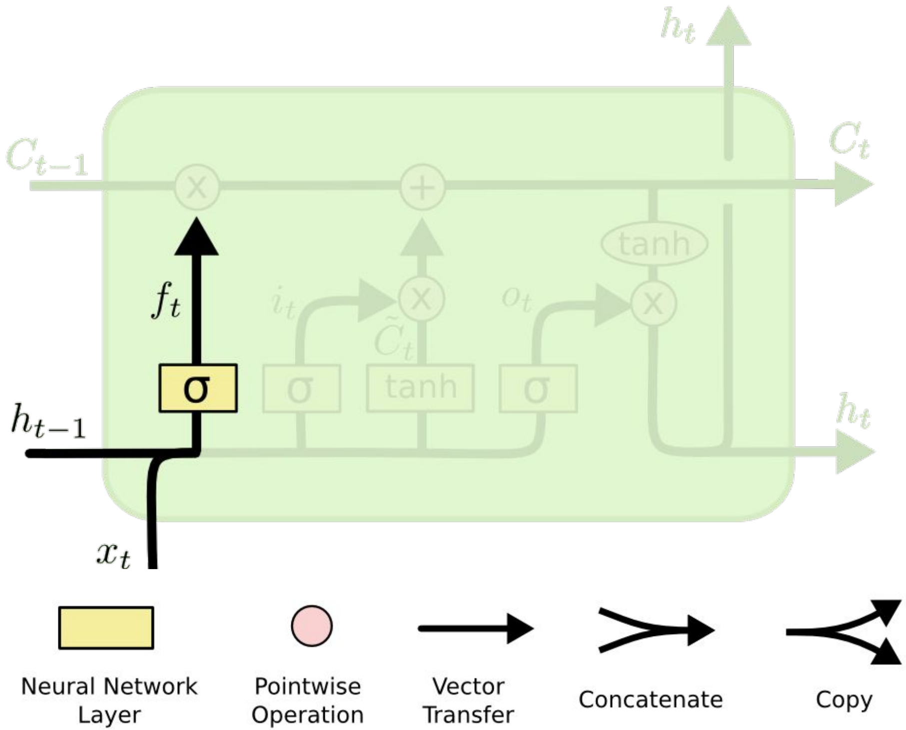


Figure: Cristopher Olah, [“Understanding LSTM Networks”]() (2015)

slide credit: Xavi Giro

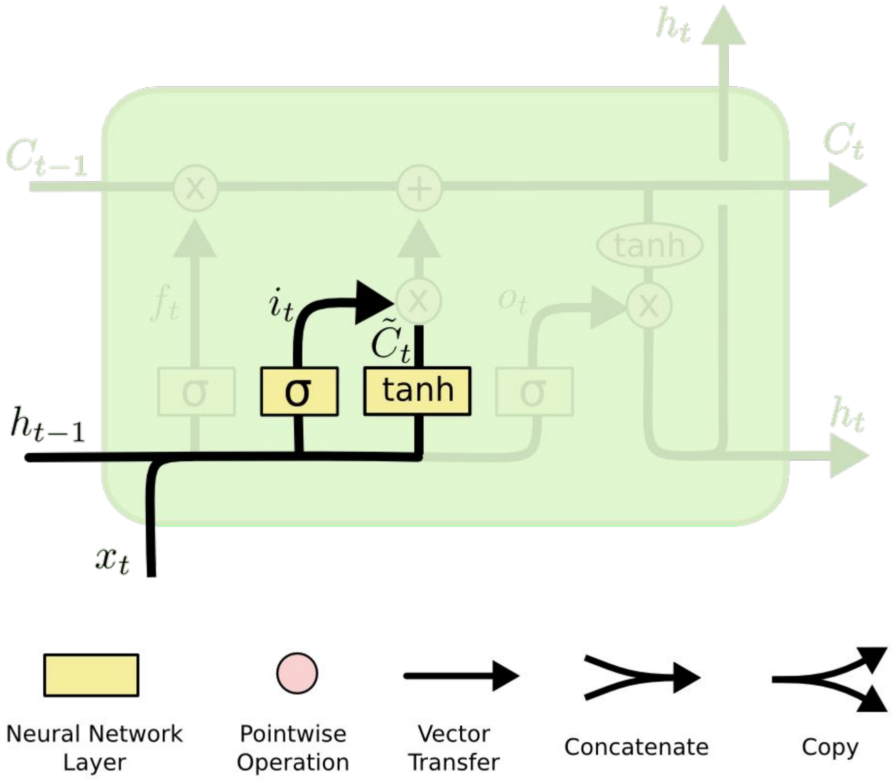
Long Short-Term Memory (LSTM)



Forget Gate:

$$f_t = \sigma\left(W_f \cdot \underbrace{[h_{t-1}, x_t]}_{\text{Concatenate}} + b_f\right)$$

Long Short-Term Memory (LSTM)



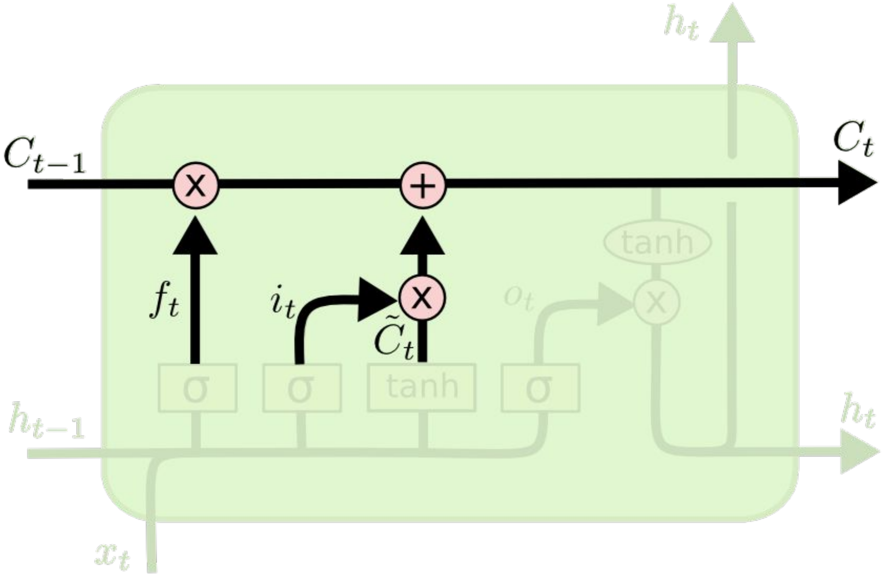
Input Gate Layer

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

New contribution to cell state

$$\tilde{C}_t = \underbrace{\tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)}_{\text{Classic neuron}}$$

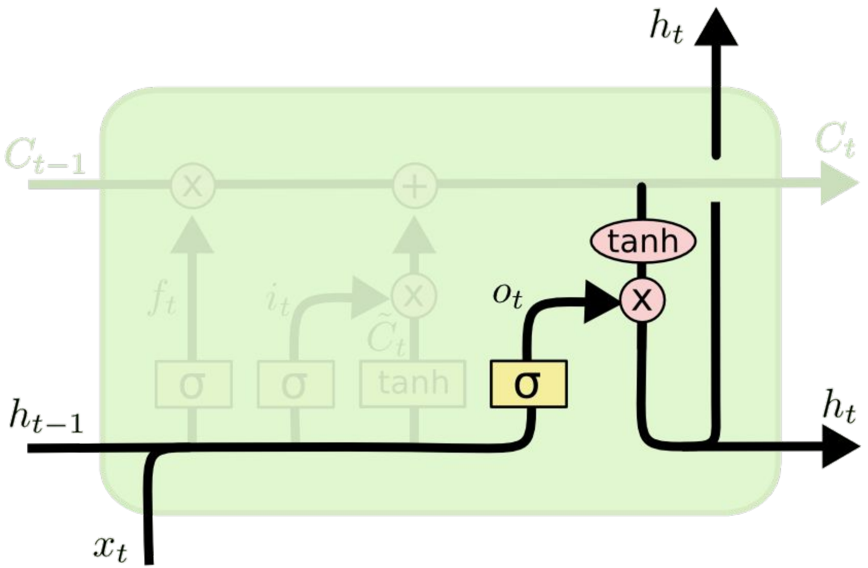
Long Short-Term Memory (LSTM)



Update Cell State (memory):

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Long Short-Term Memory (LSTM)



Output Gate Layer

$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

Output to next layer

$$h_t = o_t * \tanh\left(C_t\right)$$

Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes

Long Short-Term Memory (LSTM)

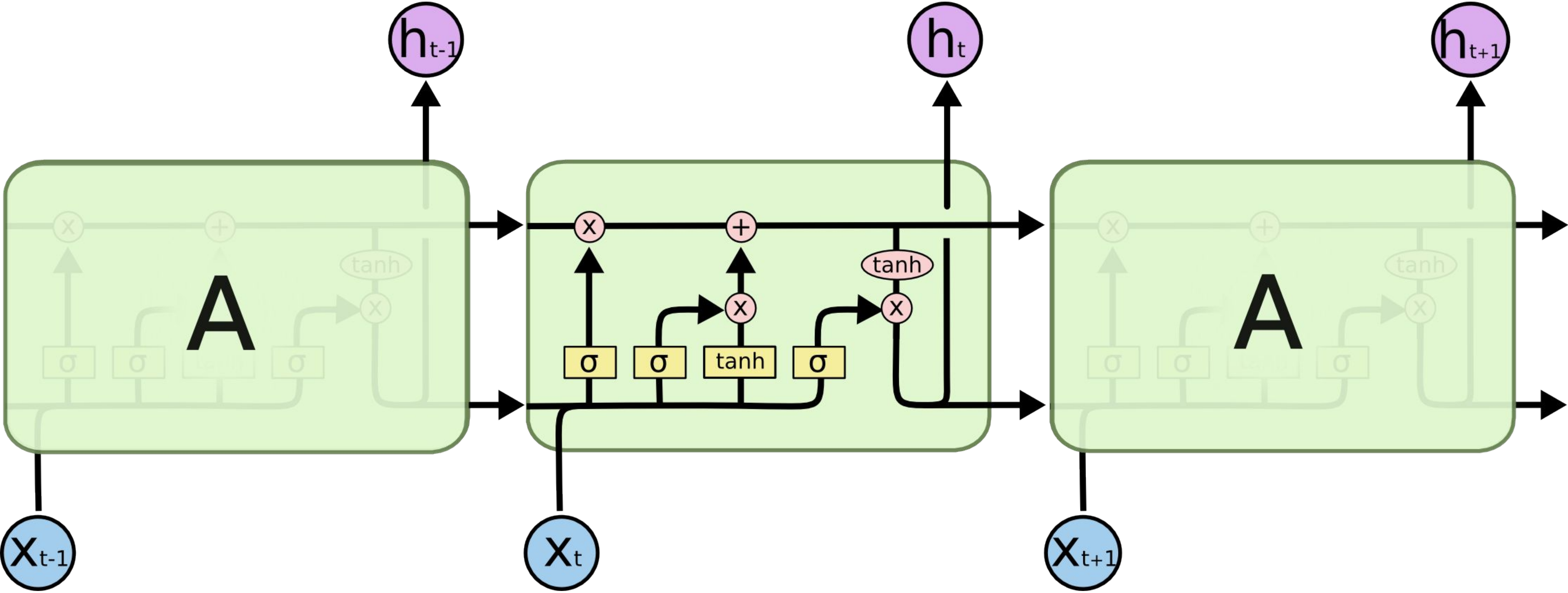


Figure: Cristopher Olah, "Understanding LSTM Networks" (2015) / Slide: Alberto Montes

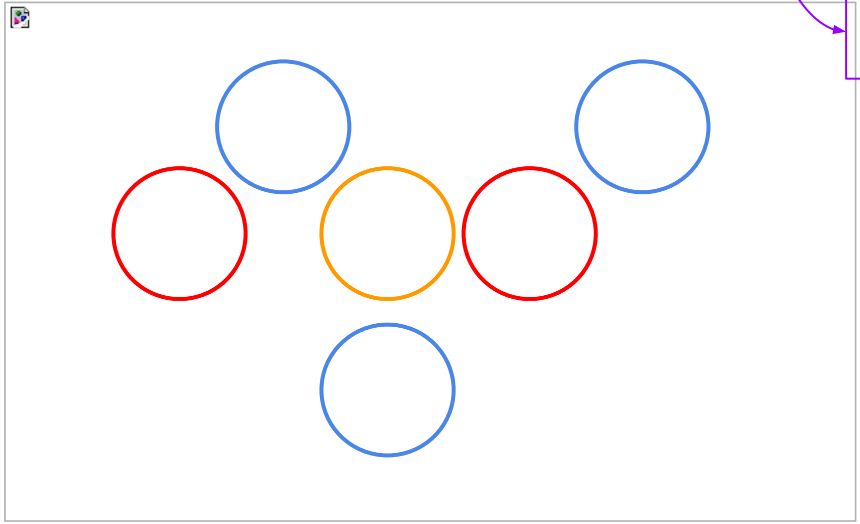
Long Short Term Memory (LSTM) cell

An LSTM cell is defined by two groups of neurons plus the cell state (memory unit):

1. **Gates**
2. **Activation units**
3. **Cell state**

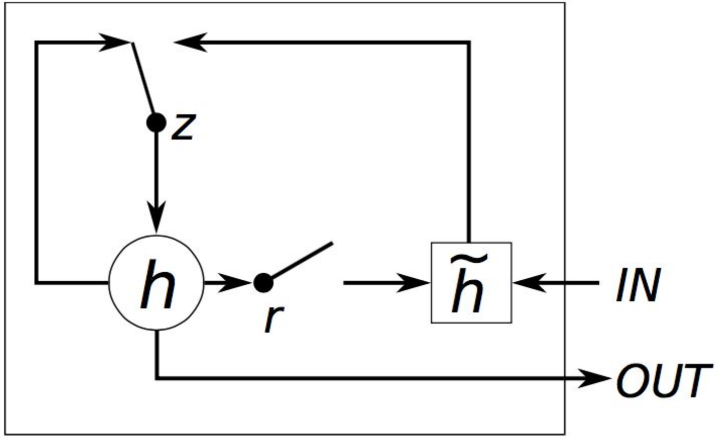
$$N^{i}_{params} = 4 \times (N^{i}_{inputs} \times N^{i}_{units} + N^{i}_{units} \times N^{i}_{units} + N^{i}_{units})$$

3 gates + input activation



Gated Recurrent Unit (GRU)

Similar performance as LSTM with less computation.



$$u_i = \sigma\left(W^{(u)}x_i + U^{(u)}h_{i-1} + b^{(u)}\right) \quad (1)$$

$$r_i = \sigma\left(W^{(r)}x_i + U^{(r)}h_{i-1} + b^{(r)}\right) \quad (2)$$

$$\tilde{h}_i = \tanh\left(Wx_i + r_i \circ Uh_{i-1} + b^{(h)}\right) \quad (3)$$

$$h_i = u_i \circ \tilde{h}_i + (1 - u_i) \circ h_{i-1} \quad (4)$$

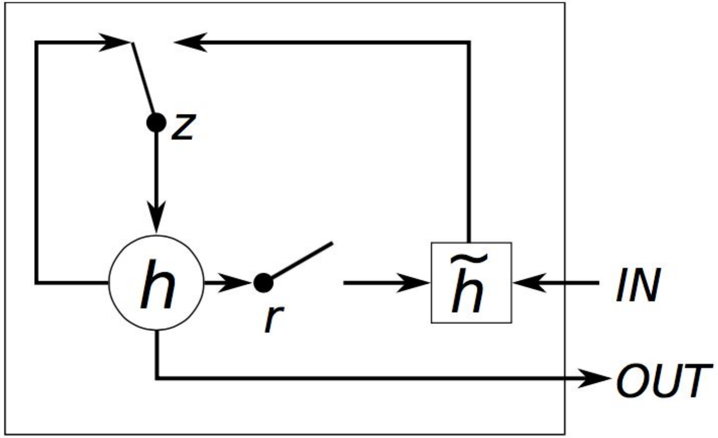
$$N^i_{params} = 3 \times (N^i_{inputs} \times N^i_{units} + N^i_{units} \times N^i_{units} + N^i_{units})$$

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." AMNLP 2014.

slide credit: Xavi Giro

Gated Recurrent Unit (GRU)

Similar performance as LSTM with less computation.



$$u_i = \sigma\left(W^{(u)}x_i + U^{(u)}h_{i-1} + b^{(u)}\right) \quad (1)$$

$$r_i = \sigma\left(W^{(r)}x_i + U^{(r)}h_{i-1} + b^{(r)}\right) \quad (2)$$

$$\tilde{h}_i = \tanh\left(Wx_i + r_i \circ Uh_{i-1} + b^{(h)}\right) \quad (3)$$

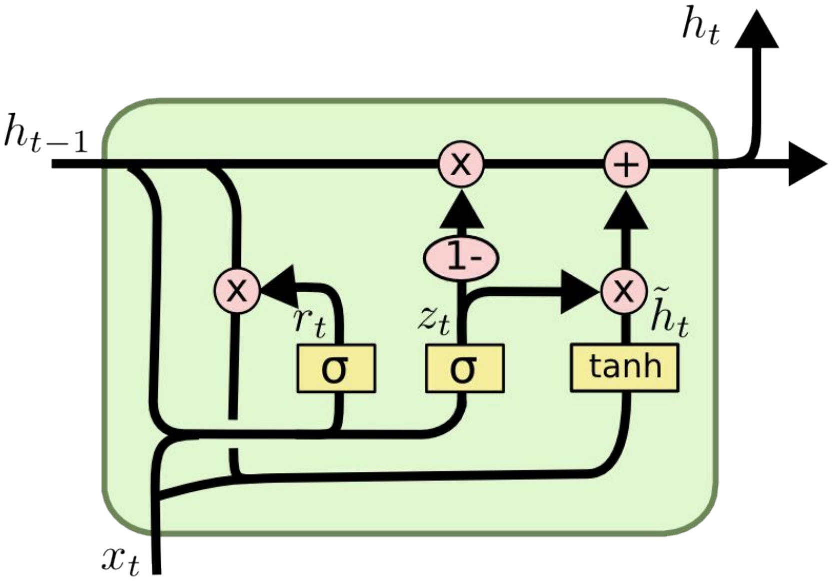
$$h_i = u_i \circ \tilde{h}_i + (1 - u_i) \circ h_{i-1} \quad (4)$$

$$N^i_{params} = 3 \times (N^i_{inputs} \times N^i_{units} + N^i_{units} \times N^i_{units} + N^i_{units})$$

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." AMNLP 2014.

slide credit: Xavi Giro

Gated Recurrent Unit (GRU)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Thanks ! Q&A ?

@Santty128