



**POLITECNICO**  
MILANO 1863



POLITECNICO  
MILANO 1863

# Artificial Neural Networks and Deep Learning

## - Recurrent Neural Networks-

Matteo Matteucci, PhD ([matteo.matteucci@polimi.it](mailto:matteo.matteucci@polimi.it))

*Artificial Intelligence and Robotics Laboratory*

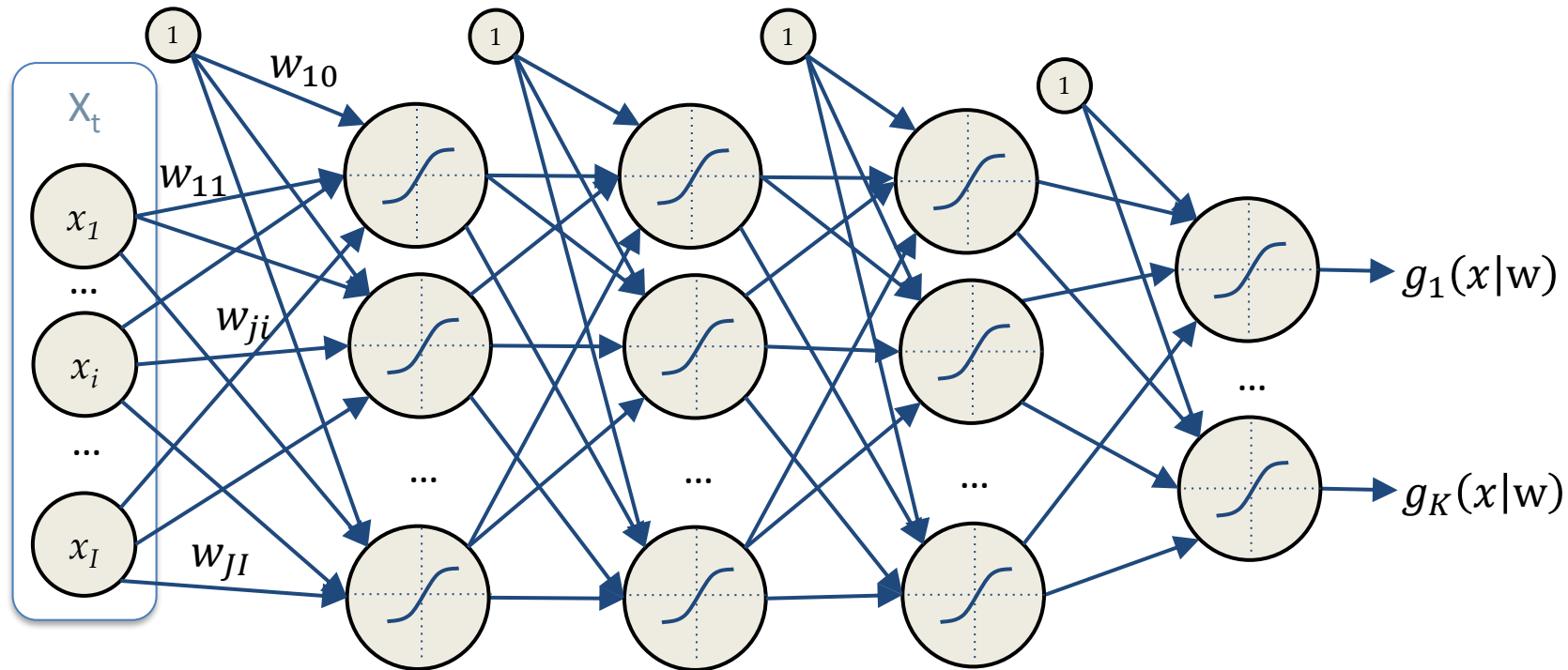
*Politecnico di Milano*

**AIRLAB**  
ARTIFICIAL INTELLIGENCE AND ROBOTICS LAB



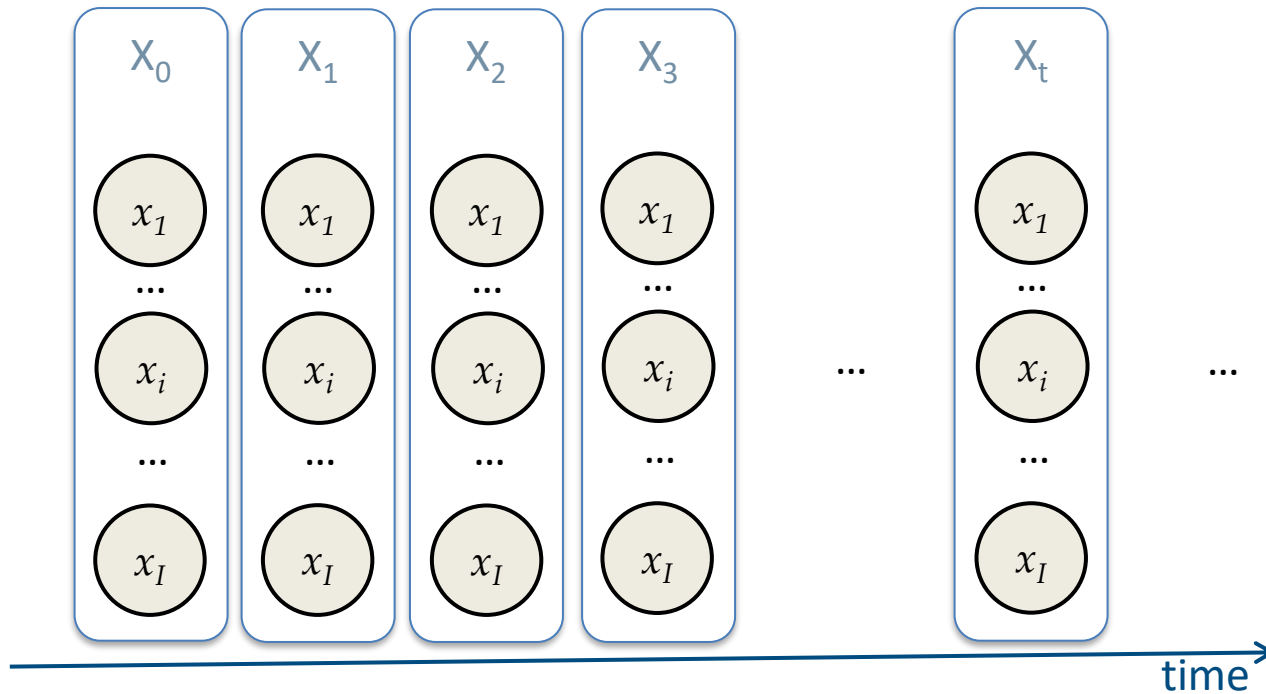
# Sequence Modeling

So far we have considered only «static» datasets



# Sequence Modeling

So far we have considered only «static» datasets



# Sequence Modeling

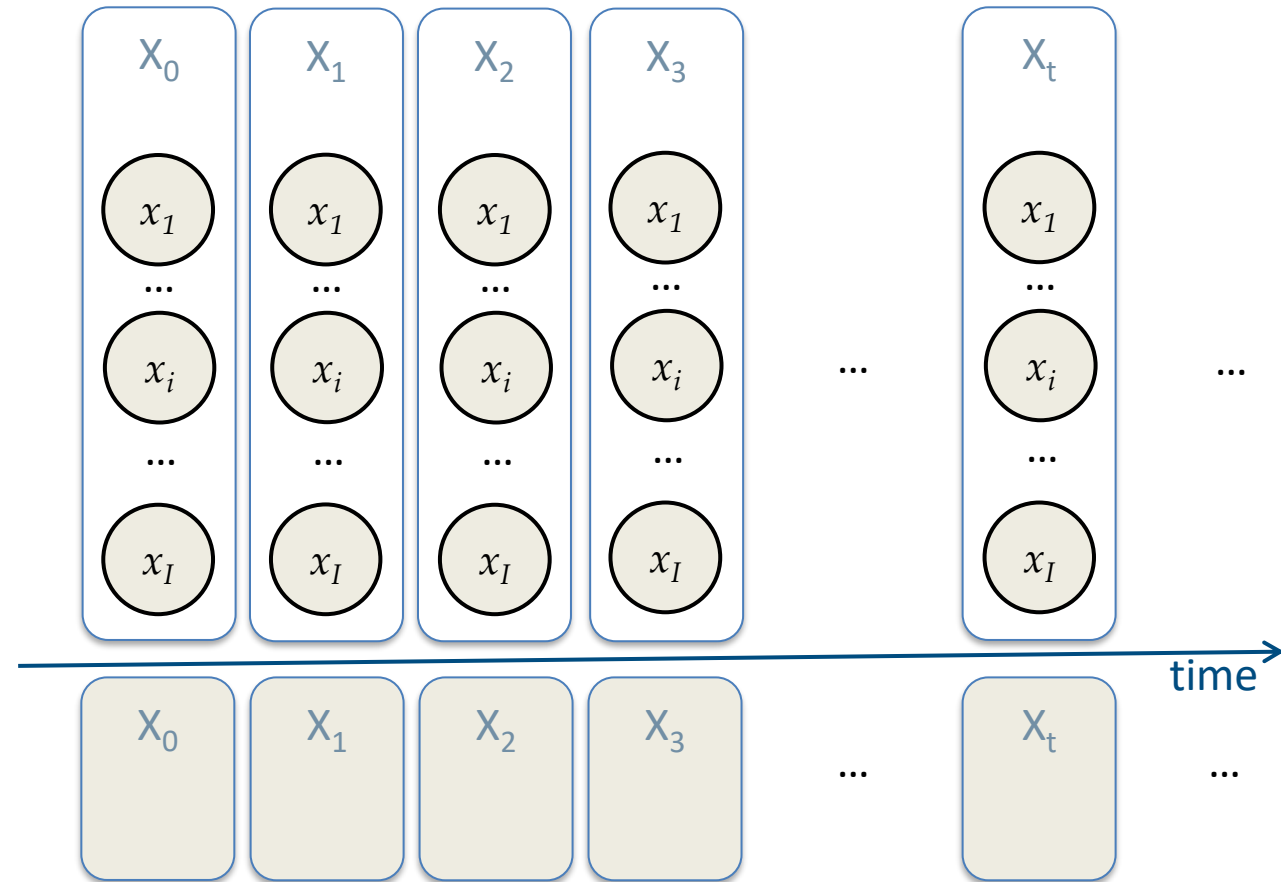
Different ways to deal with «dynamic» data:

Memoryless models (fixed lag):

- Autoregressive models
- Feedforward neural networks

Models with memory (unlimited):

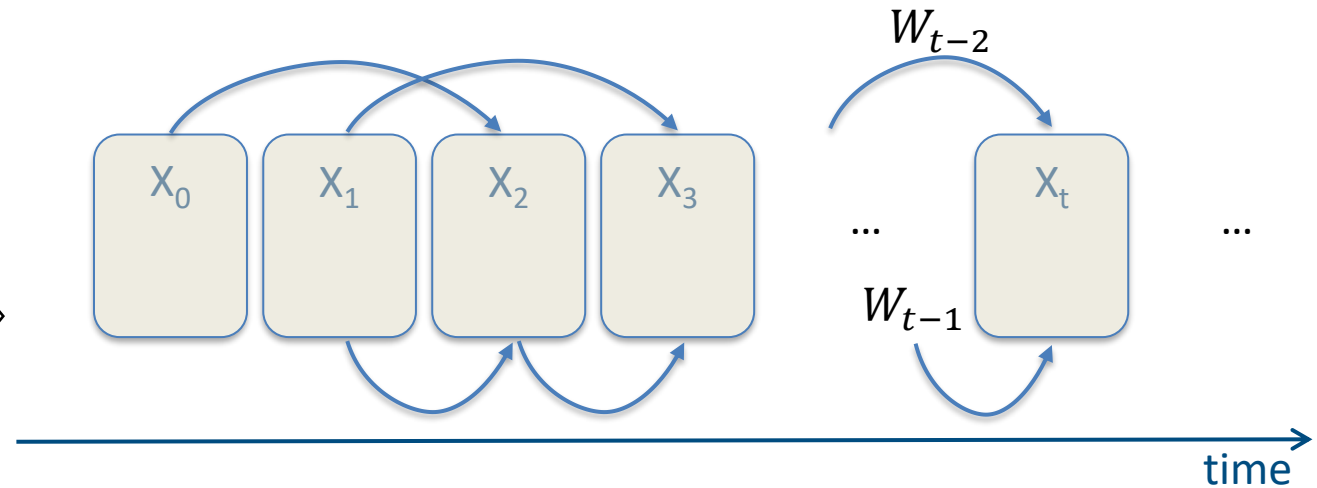
- Linear dynamical systems
- Hidden Markov models
- Recurrent Neural Networks
- ...



# Memoryless Models for Sequences (1/2)

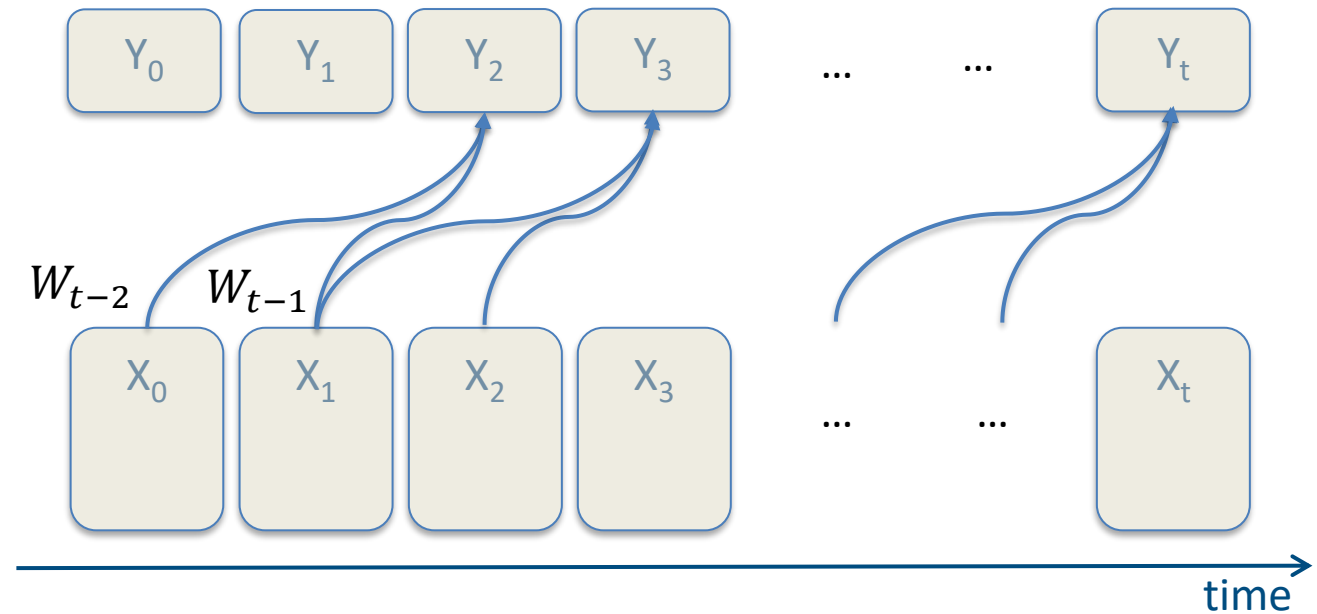
## Autoregressive models

- Predict the next input from previous ones using «delay taps»



## Linear models with fixed lag

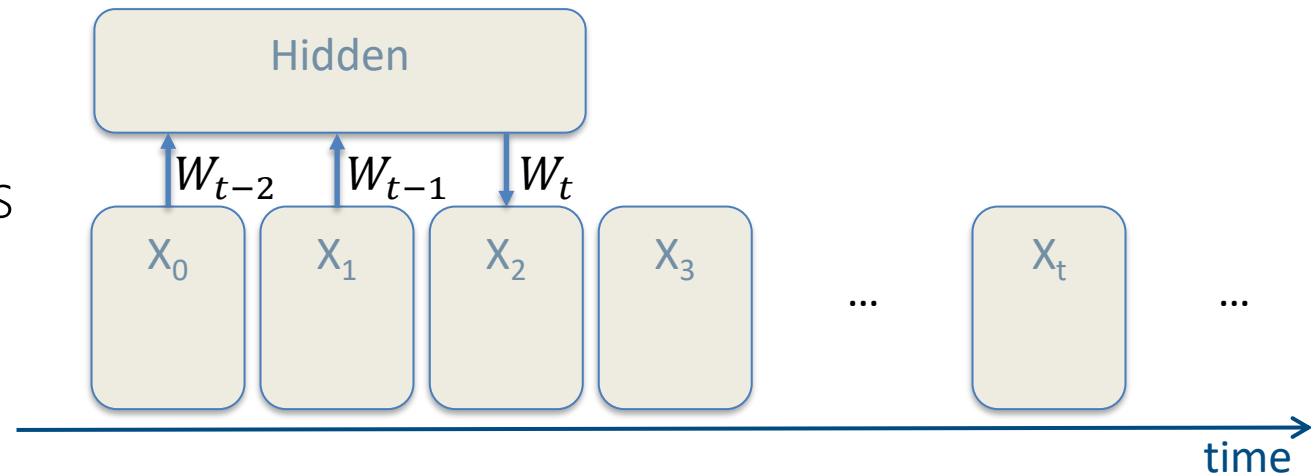
- Predict the next output from previous inputs using «delay taps»



# Memoryless Models for Sequences (2/2)

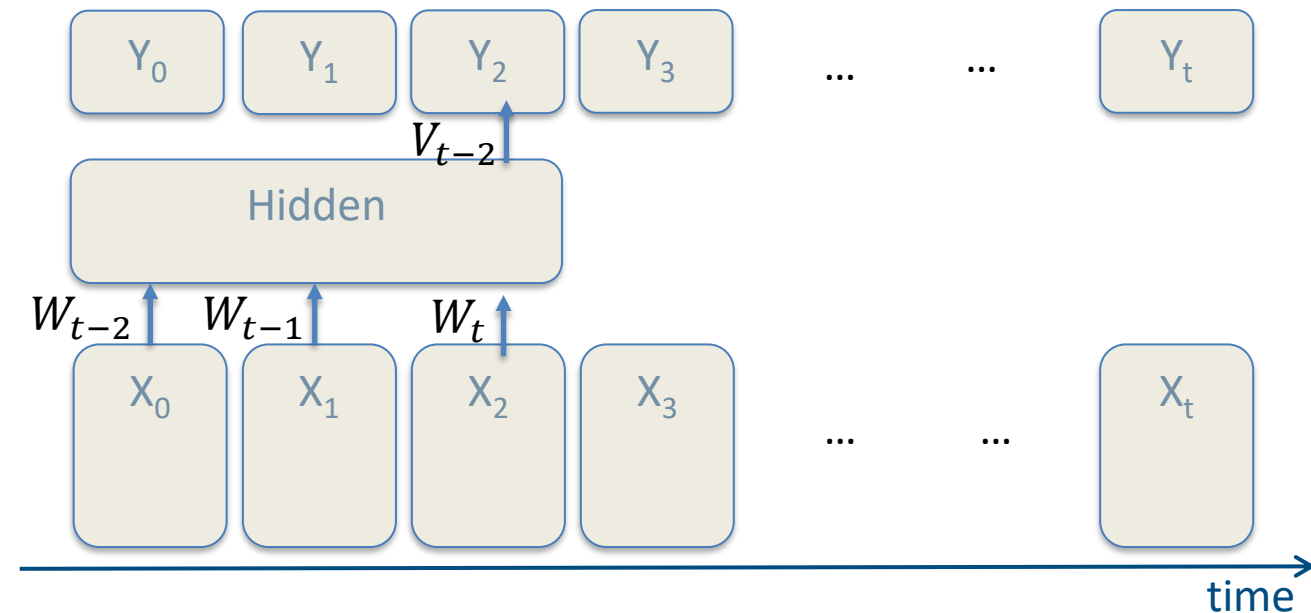
## Feed forward neural networks

- Generalize autoregressive models using non linear hidden layers



## Feed forward neural networks with delays

- Predict the next output from previous inputs and previous outputs using «delay taps»



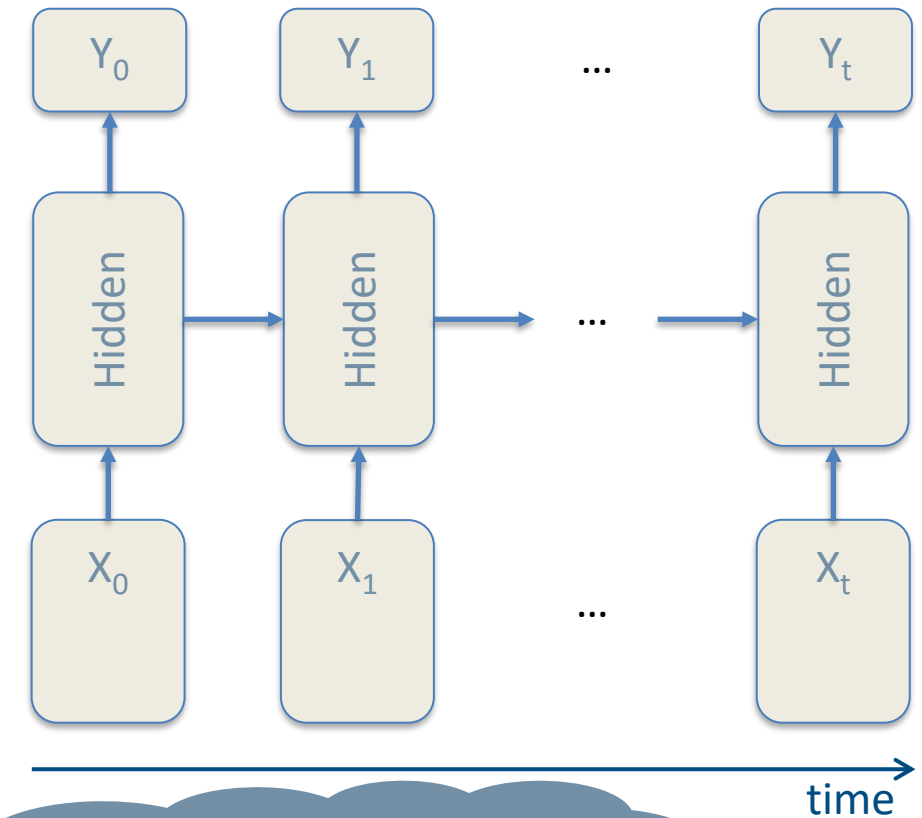
# Dynamical Systems (Models with Memory)

Generative models with a hidden state which cannot be observed directly

- The hidden state has some dynamics possibly affected by noise and produces the output
- To compute the output need to infer hidden state
- Input are treated as driving inputs

In linear dynamical systems this becomes:

- State continuous with Gaussian uncertainty
- Transformations are assumed to be linear
- State can be estimated using *Kalman filtering*

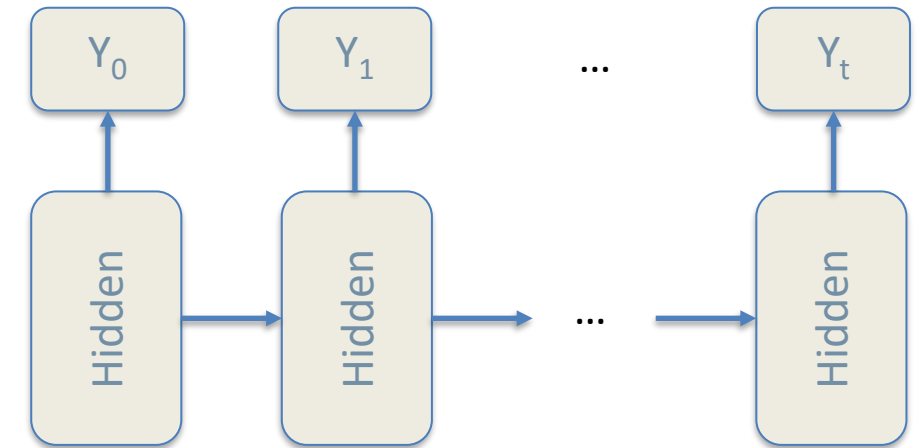


Stochastic systems ...

# Dynamical Systems (Models with Memory)

Generative models with a hidden state which cannot be observed directly

- The hidden state has some dynamics possibly affected by noise and produces the output
- To compute the output need to infer hidden state
- Input are treated as driving inputs



In hidden Markov models this becomes:

- State assumed to be discrete, state transitions are stochastic (transition matrix)
- Output is a stochastic function of hidden states
- State can be estimated via *Viterbi algorithm*.

time →

Stochastic systems ...



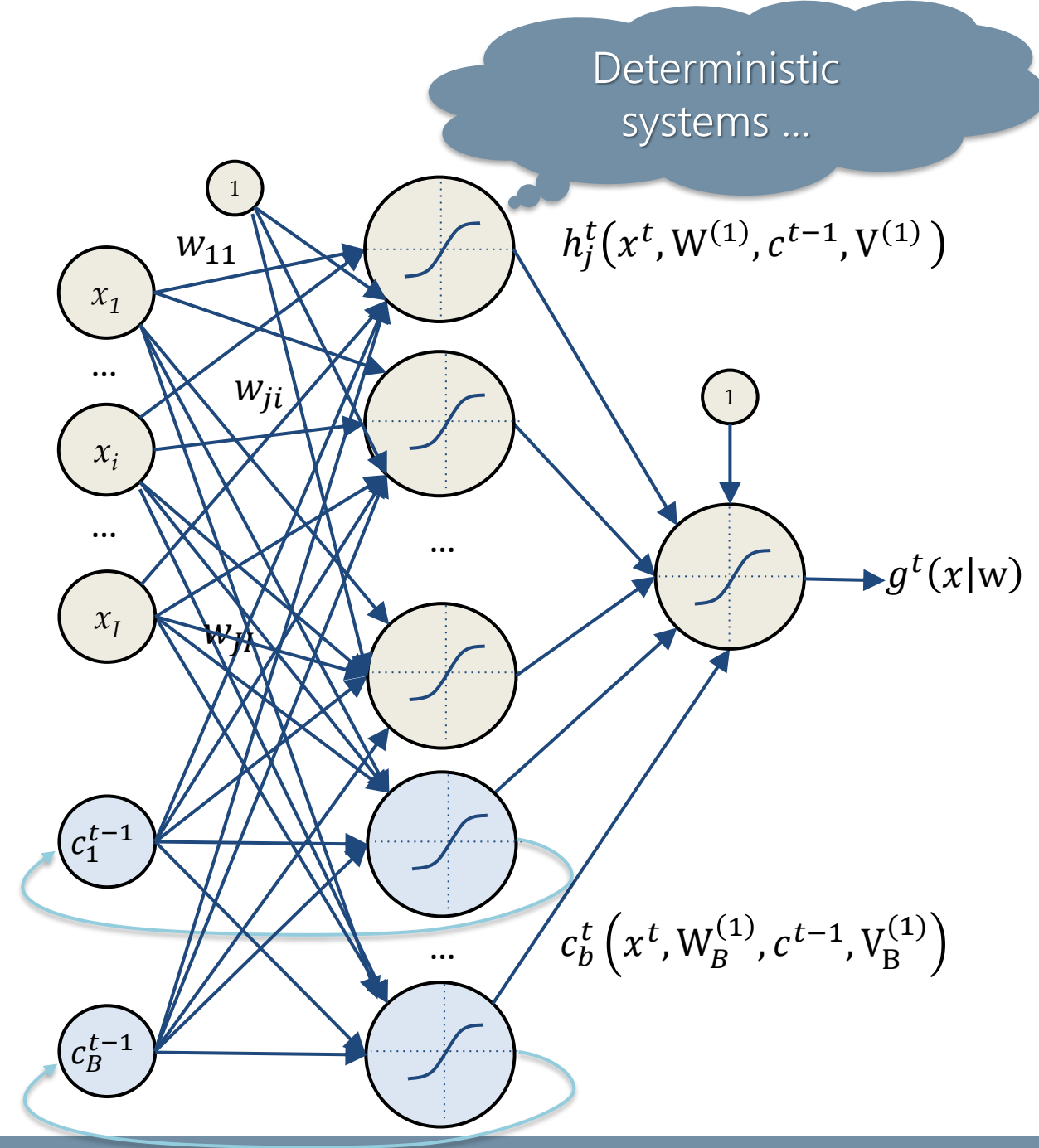
# Recurrent Neural networks

Memory via recurrent connections:

- Distributed hidden state allows to store information efficiently
- Non-linear dynamics allows complex hidden state updates

**“With enough neurons and time, RNNs can compute anything that can be computed by a computer.”**

(Computation Beyond the Turing Limit  
Hava T. Siegelmann, 1995)



# Recurrent Neural networks

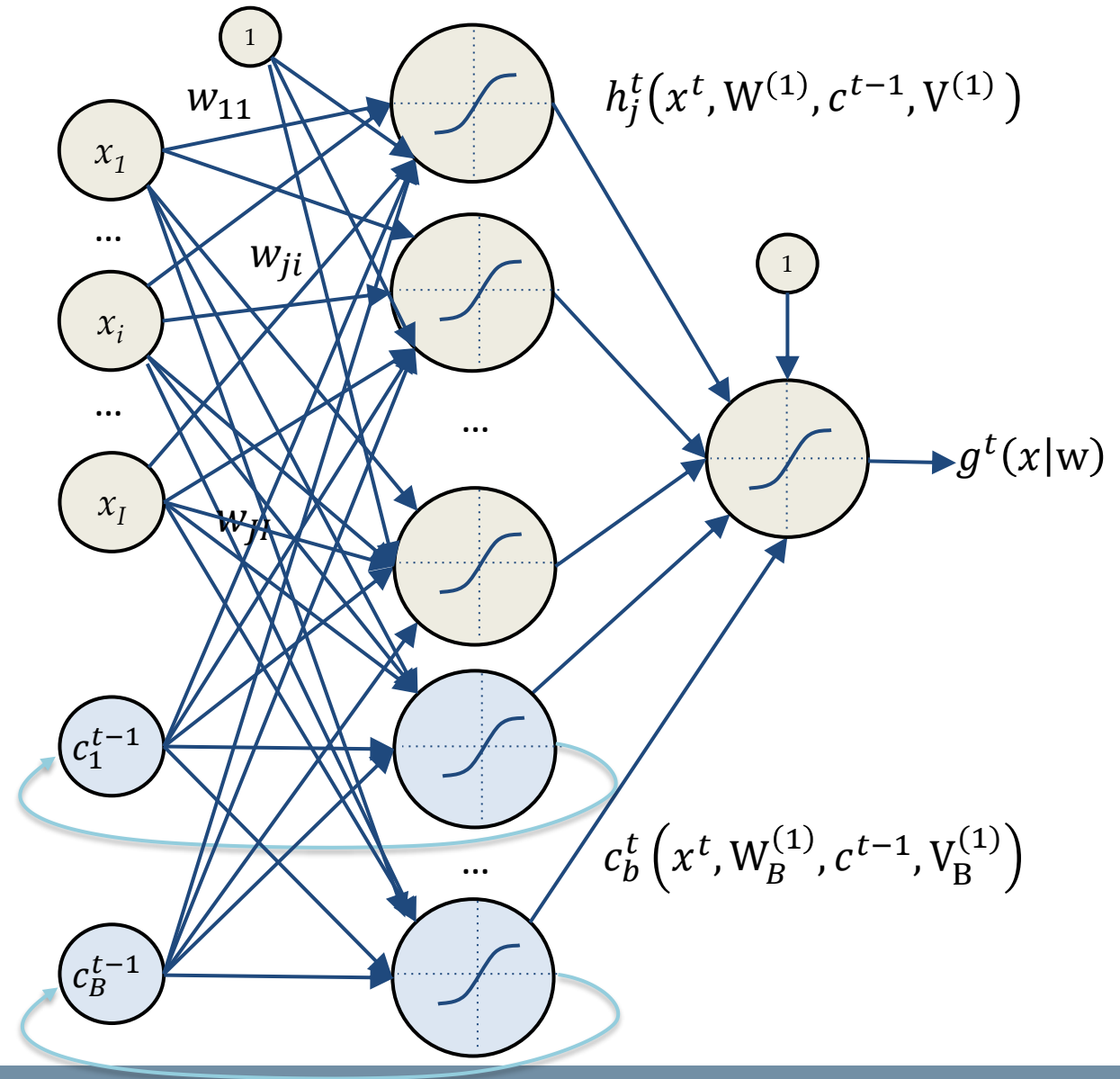
Memory via recurrent connections:

- Distributed hidden state allows to store information efficiently
- Non-linear dynamics allows complex hidden state updates

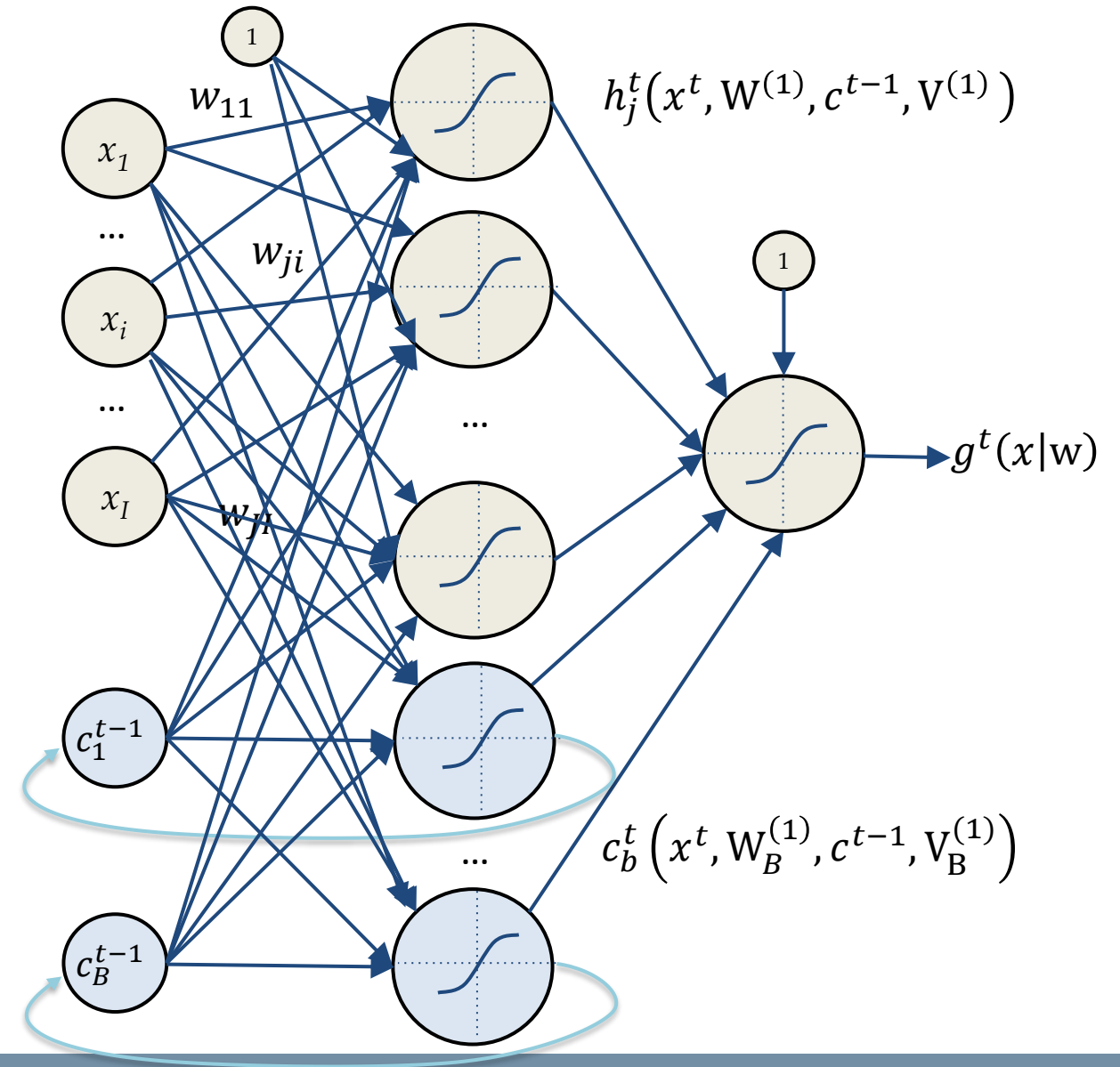
$$g^t(x_n|w) = g \left( \sum_{j=0}^J w_{1j}^{(2)} \cdot h_j^t(\cdot) + \sum_{b=0}^B v_{1b}^{(2)} \cdot c_b^t(\cdot) \right)$$

$$h_j^t(\cdot) = h_j \left( \sum_{i=0}^J w_{ji}^{(1)} \cdot x_{i,n}^t + \sum_{b=0}^B v_{jb}^{(1)} \cdot c_b^{t-1} \right)$$

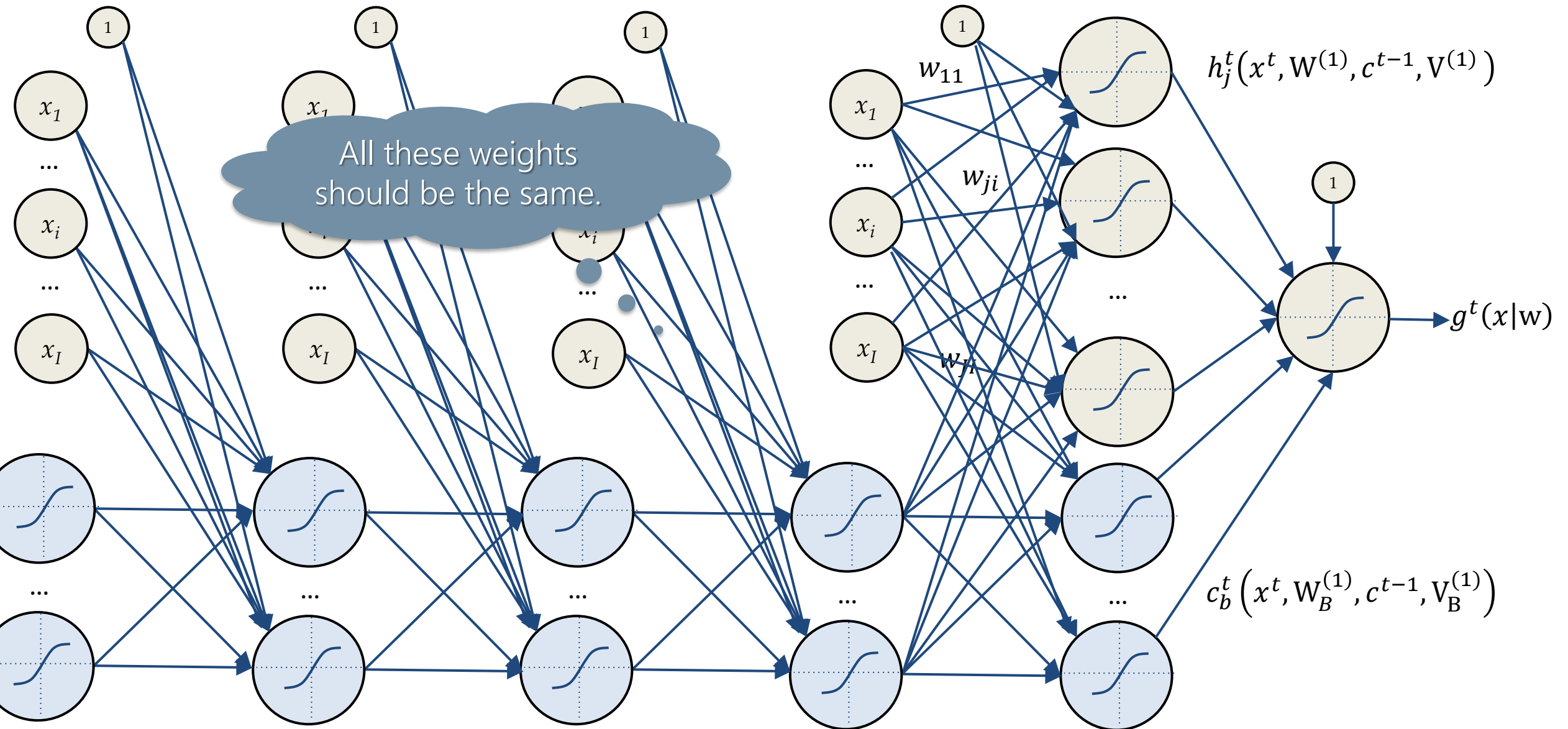
$$c_b^t(\cdot) = c_b \left( \sum_{i=0}^J w_{bi}^{(1)} \cdot x_{i,n}^t + \sum_{b'=0}^B v_{bb'}^{(1)} \cdot c_{b'}^{t-1} \right)$$



# Backpropagation Through Time



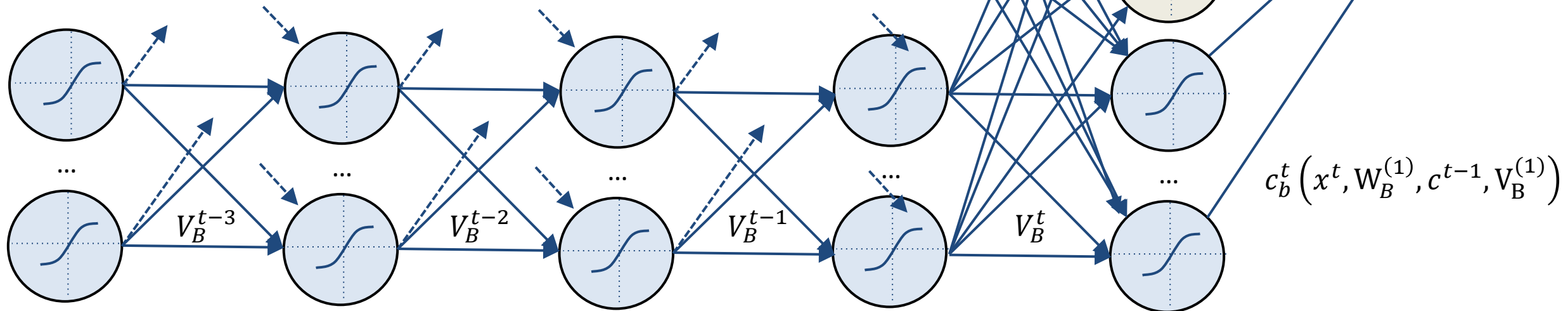
# Backpropagation Through Time



# Backpropagation Through Time

- Perform network unroll for  $U$  steps
- Initialize  $\mathbf{W}_B, \mathbf{V}_B$  replicas to be the same
- Compute gradients and update replicas with the average of their gradients

$$W_B = W_B - \eta \cdot \frac{1}{U} \sum_{u=0}^{U-1} \frac{\partial E}{\partial W_B^{t-u}} \quad V_B = V_B - \eta \cdot \frac{1}{U} \sum_{u=0}^{U-1} \frac{\partial E^t}{\partial V_B^{t-u}}$$

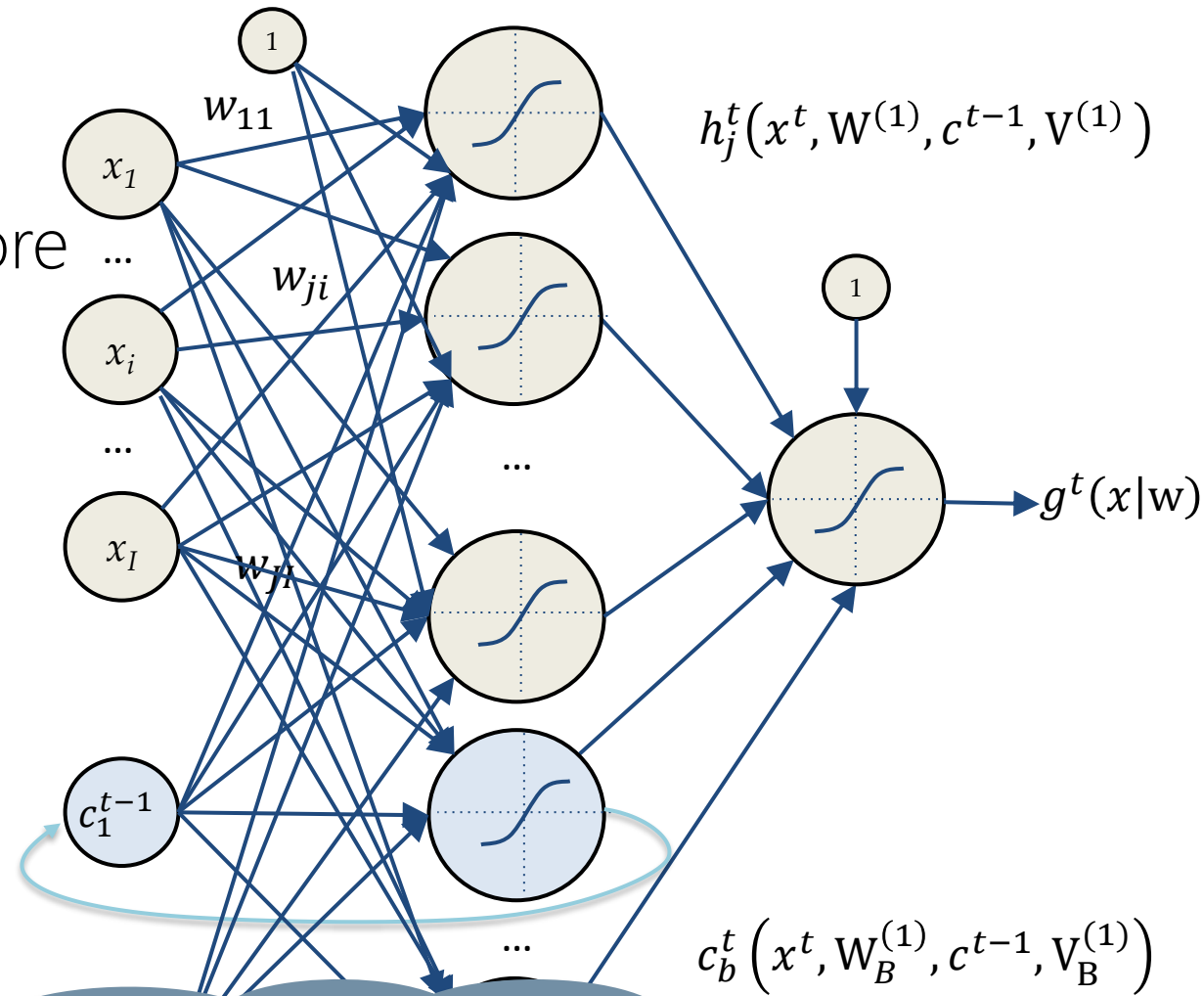


# How much should we go back in time?

Sometime output might be related to some input happened quite long before

Jane walked into the room. John walked in too.  
It was late in the day. Jane said hi to <???

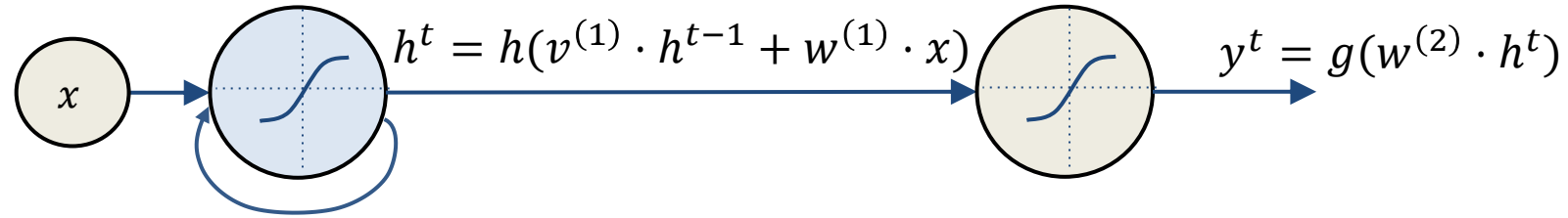
However backpropagation through time was not able to train recurrent neural networks significantly back in time ...



Was due to not being able to backprop through many layers ...

# How much can we go back in time?

To better understand why it was not working consider a simplified case:



Backpropagation over an entire sequence  $S$  is computed as

$$\frac{\partial E}{\partial w} = \sum_{t=1}^S \frac{\partial E^t}{\partial w} \rightarrow \frac{\partial E^t}{\partial w} = \sum_{t=1}^t \frac{\partial E^t}{\partial y^t} \frac{\partial y^t}{\partial h^t} \frac{\partial h^t}{\partial h^k} \frac{\partial h^k}{\partial w} \rightarrow \frac{\partial h^t}{\partial h^k} = \prod_{i=k+1}^t \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k+1}^t v^{(1)} h'(v^{(1)} \cdot h^{i-1} + w^{(1)} \cdot x)$$

If we consider the norm of these terms

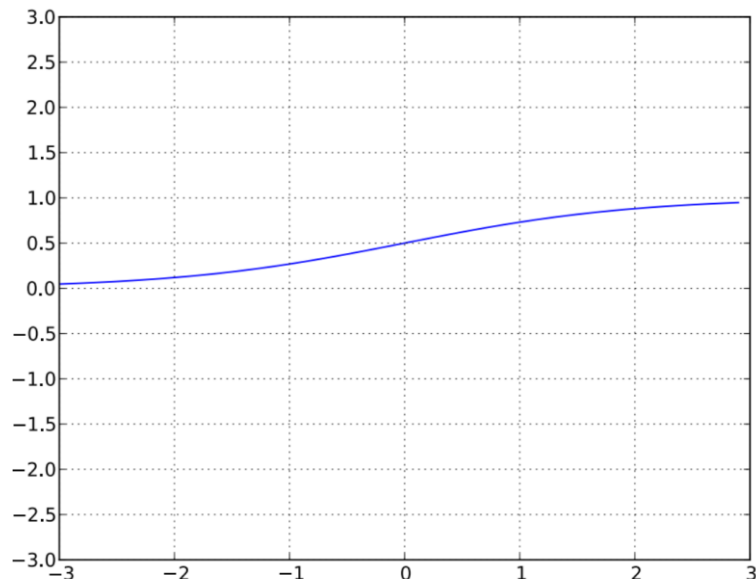
$$\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\| \leq \|v^{(1)}\| \|h'(\cdot)\| \rightarrow \left\| \frac{\partial h^t}{\partial h^k} \right\| \leq (\gamma_v \cdot \gamma_{h'})^{t-k}$$

If  $(\gamma_v \cdot \gamma_{h'}) < 1$  this converges to 0 ...

With Sigmoids and Tanh we have vanishing gradients



# Which Activation Function?

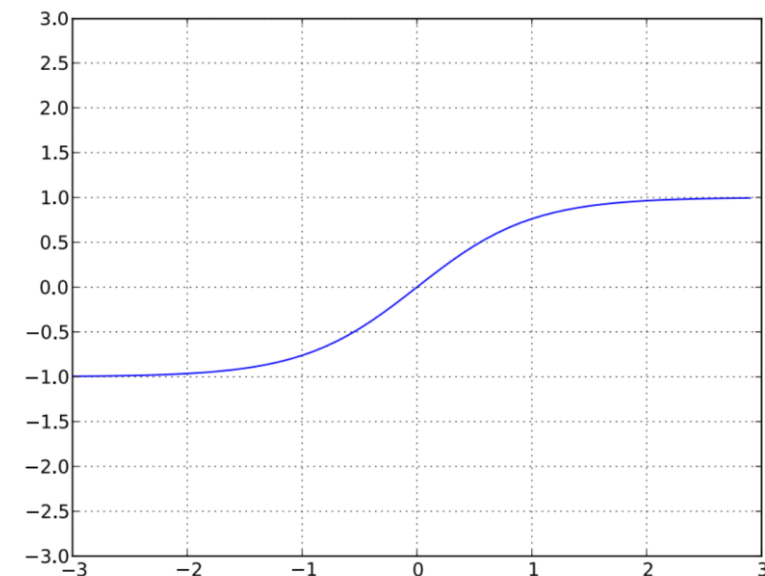


Sigmoid activation function

$$g(a) = \frac{1}{1 + \exp(-a)}$$

$$g'(a) = g(a)(1 - g(a))$$

$$g'(0) = g(0)(1 - g(0)) = \frac{1}{1 + \exp(0)} \cdot \frac{\exp(0)}{1 + \exp(0)} = 0.25$$



Tanh activation function

$$g(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)}$$

$$g'(a) = 1 - g(a)^2$$

$$g'(0) = 1 - g(0)^2 = 1 - \left(\frac{\exp(0) - \exp(0)}{\exp(0) + \exp(0)}\right)^2 = 1$$

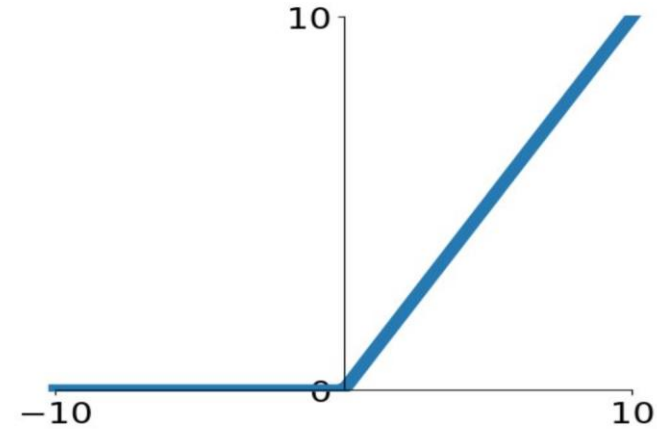


# Dealing with Vanishing Gradient

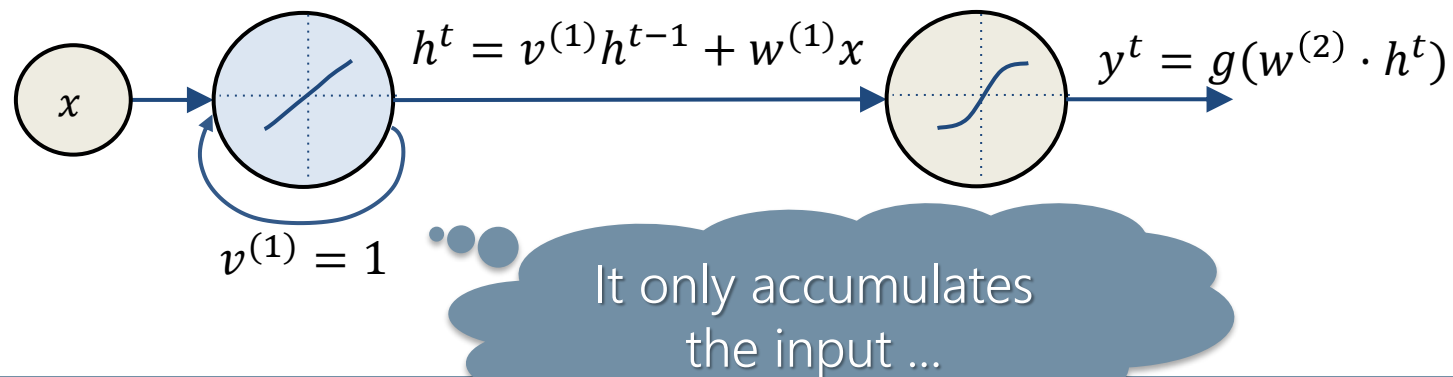
Force all gradients to be either 0 or 1

$$g(a) = \text{ReLu}(a) = \max(0, a)$$

$$g'(a) = 1_{a>0}$$



Build Recurrent Neural Networks using small modules that are designed to remember values for a long time.

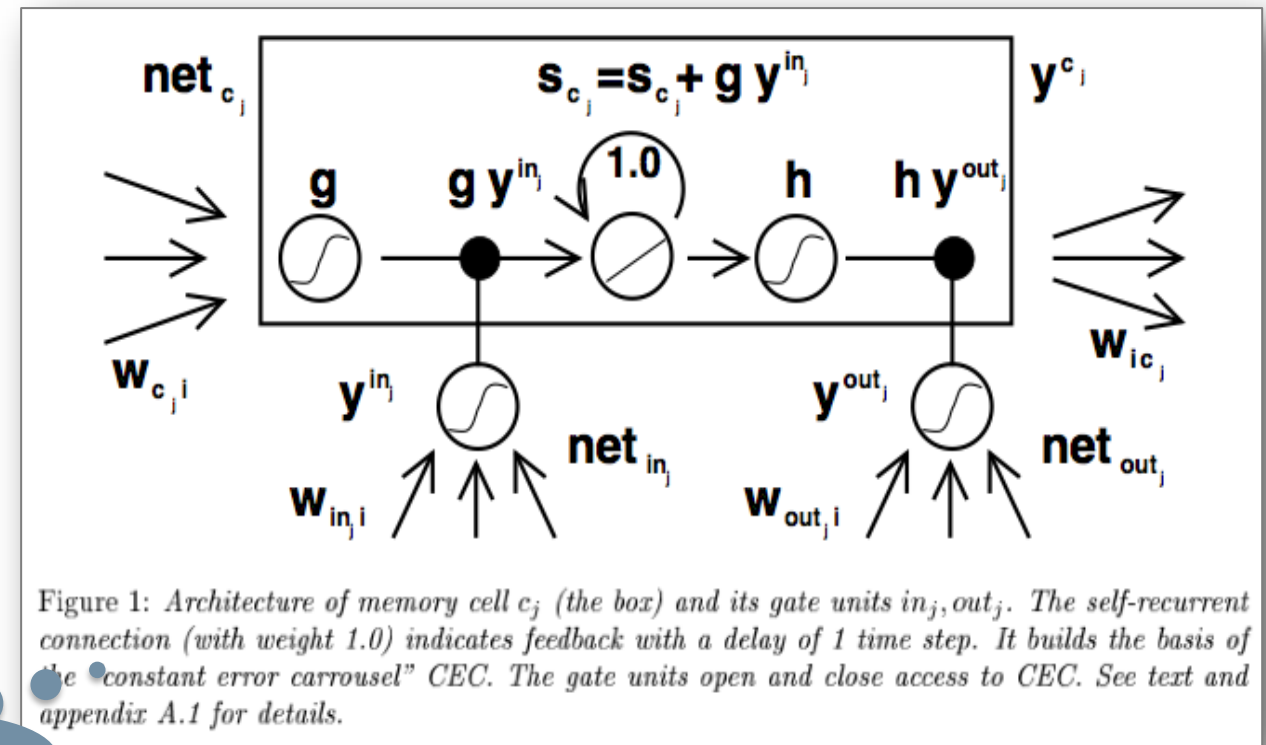


# Long Short-Term Memories (LSTM)

Hochreiter & Schmidhuber (1997) solved the problem of vanishing gradient designing a memory cell using logistic and linear units with multiplicative interactions:

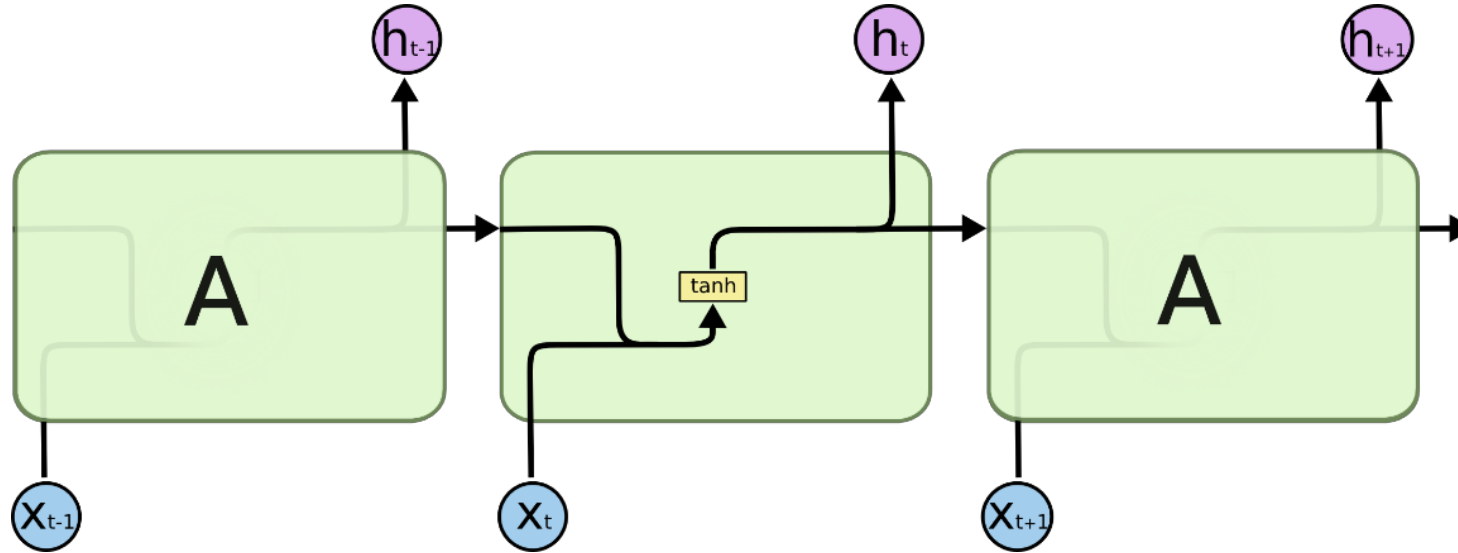
- Information gets into the cell whenever its "write" gate is on.
- The information stays in the cell so long as its "keep" gate is on.
- Information is read from the cell by turning on its "read" gate.

Can backpropagate through this since the loop has fixed weight.

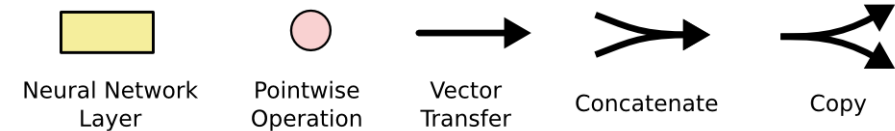
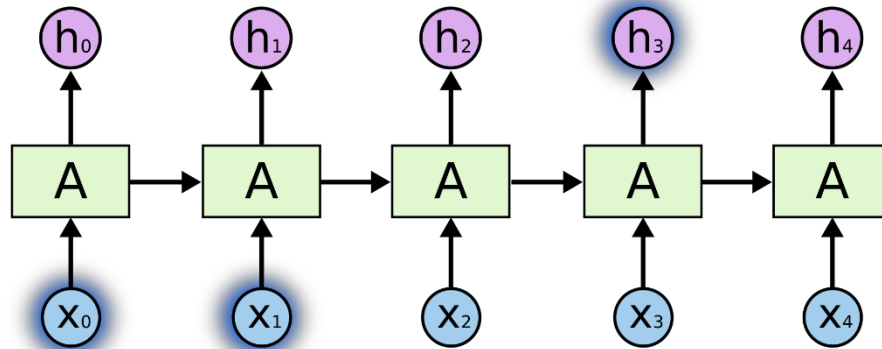


# RNN vs. LSTM

RNN

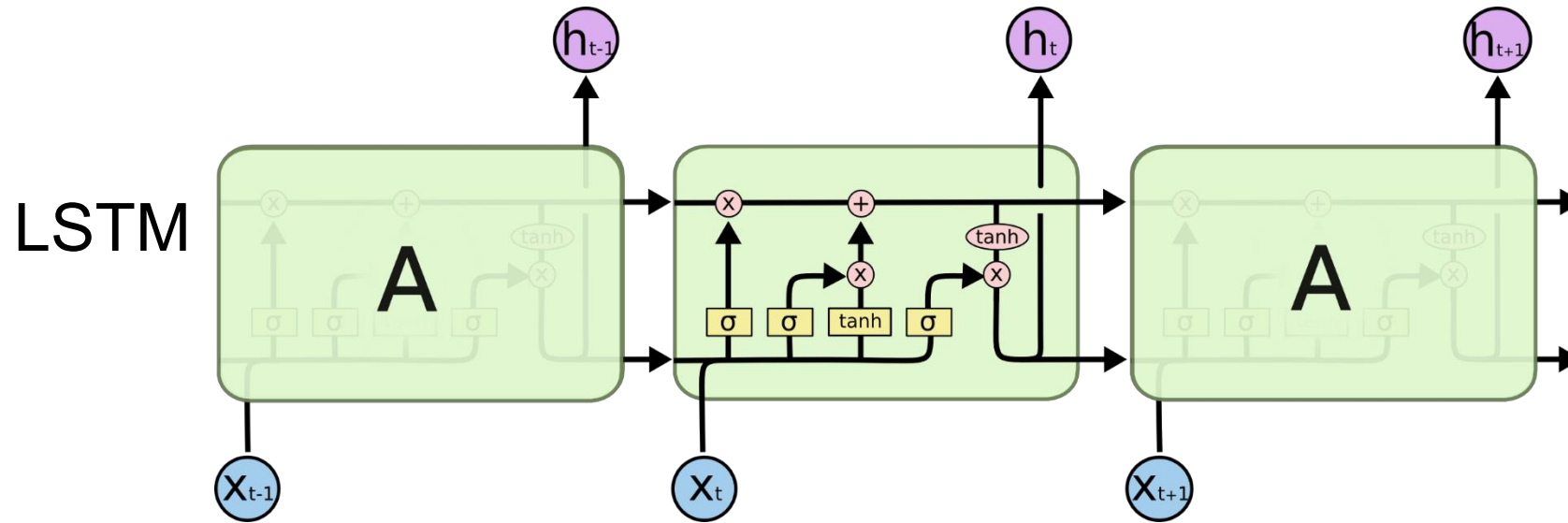


$$a_h^t = \sum_{i=1}^I w_{ih} x_i^t + \sum_{h'=1}^H w_{h'h} b_{h'}^{t-1}$$
$$b_h^t = \theta_h(a_h^t)$$



LSTM Images from: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# Long Short-Term Memory



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

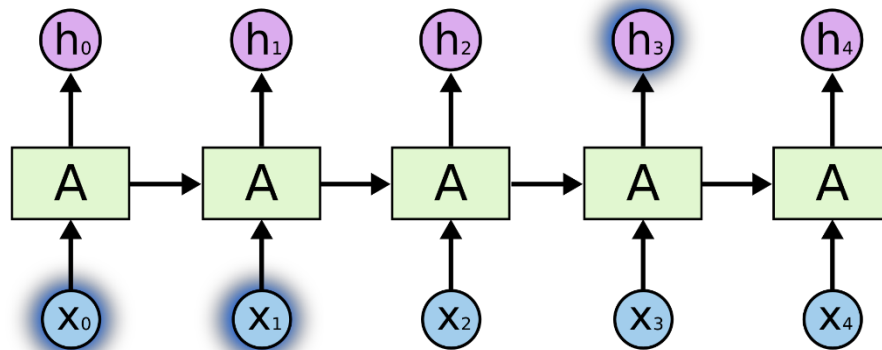
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

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

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

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



Neural Network Layer

Pointwise Operation

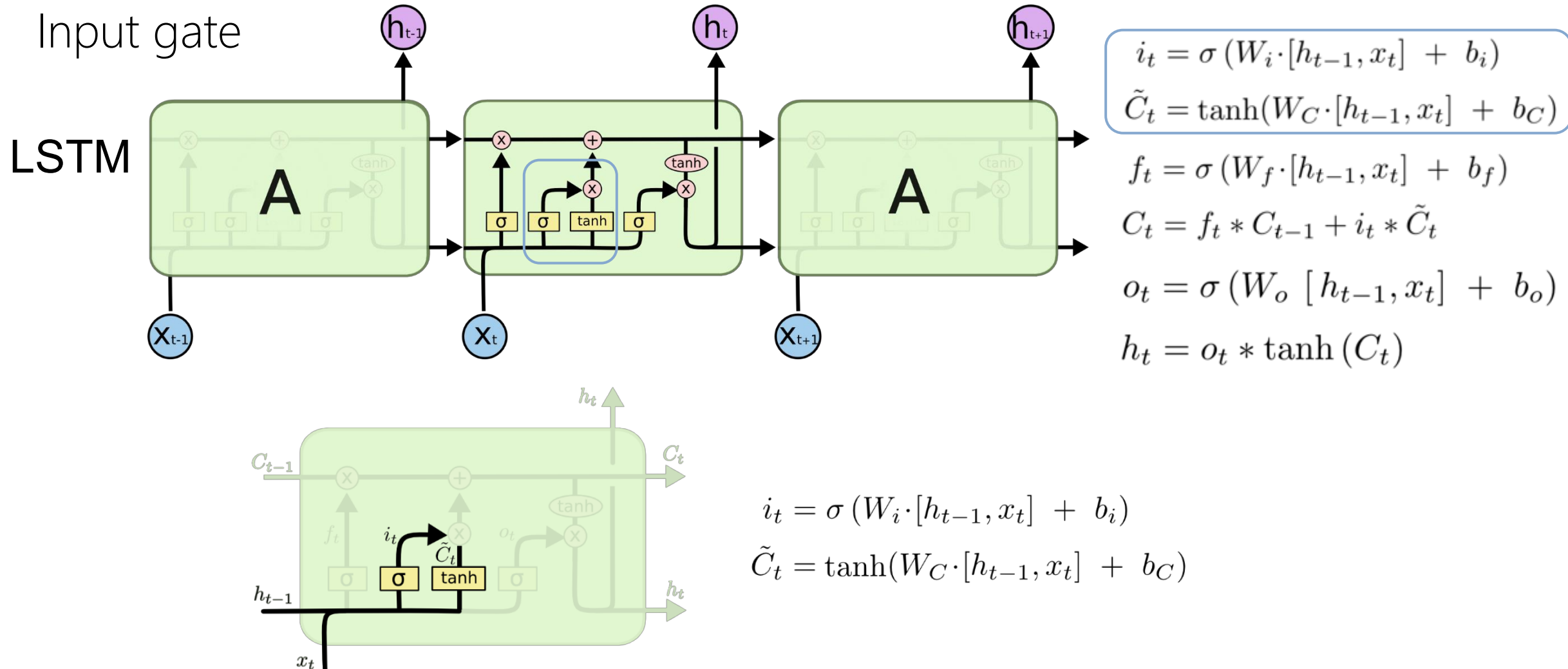
Vector Transfer

Concatenate

Copy

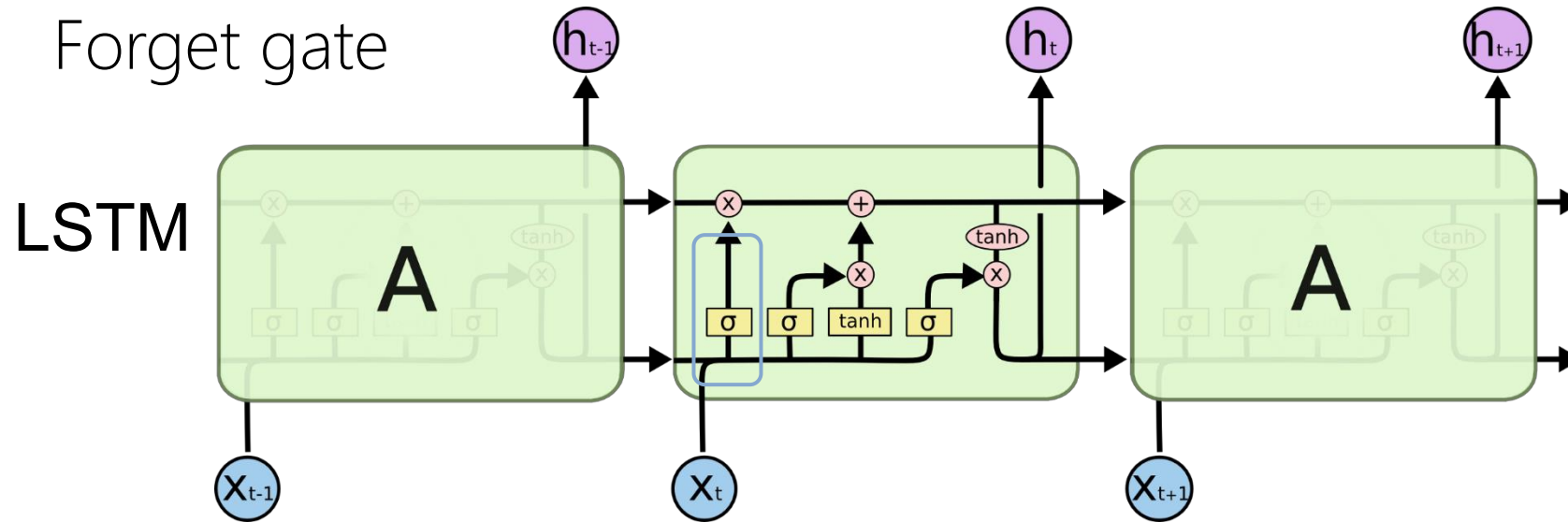
LSTM Images from: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# Long Short-Term Memory



LSTM Images from: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# Long Short-Term Memory



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

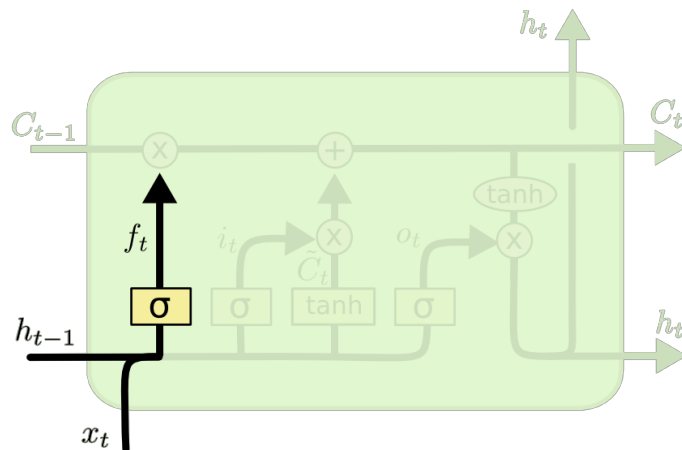
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

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

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

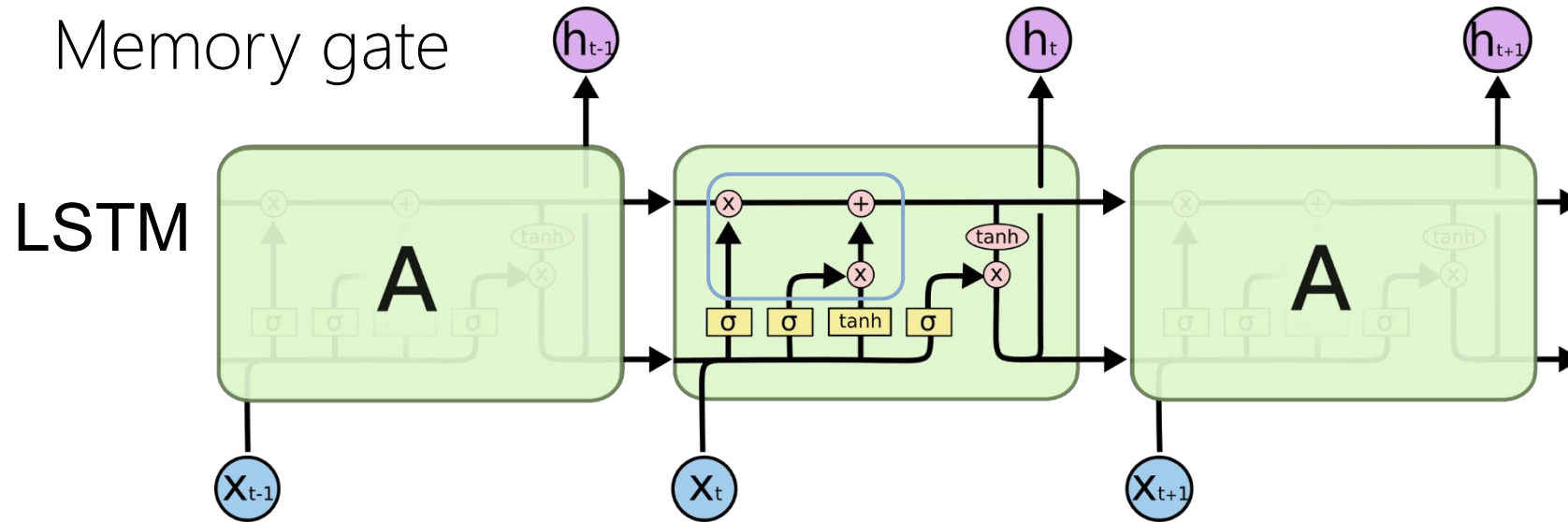
$$h_t = o_t * \tanh(C_t)$$



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

LSTM Images from: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# Long Short-Term Memory



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

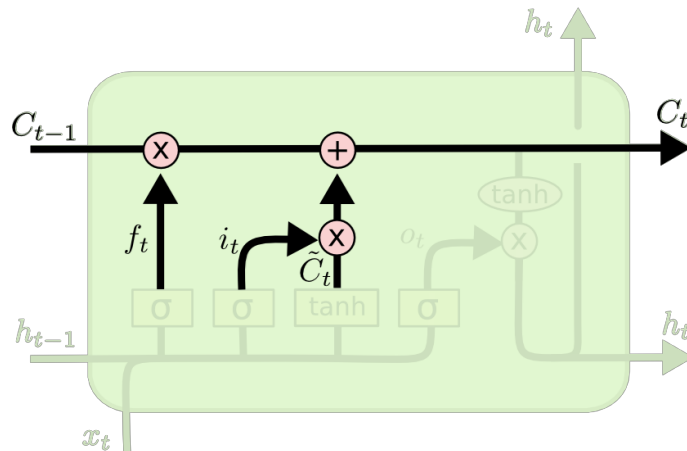
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

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

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

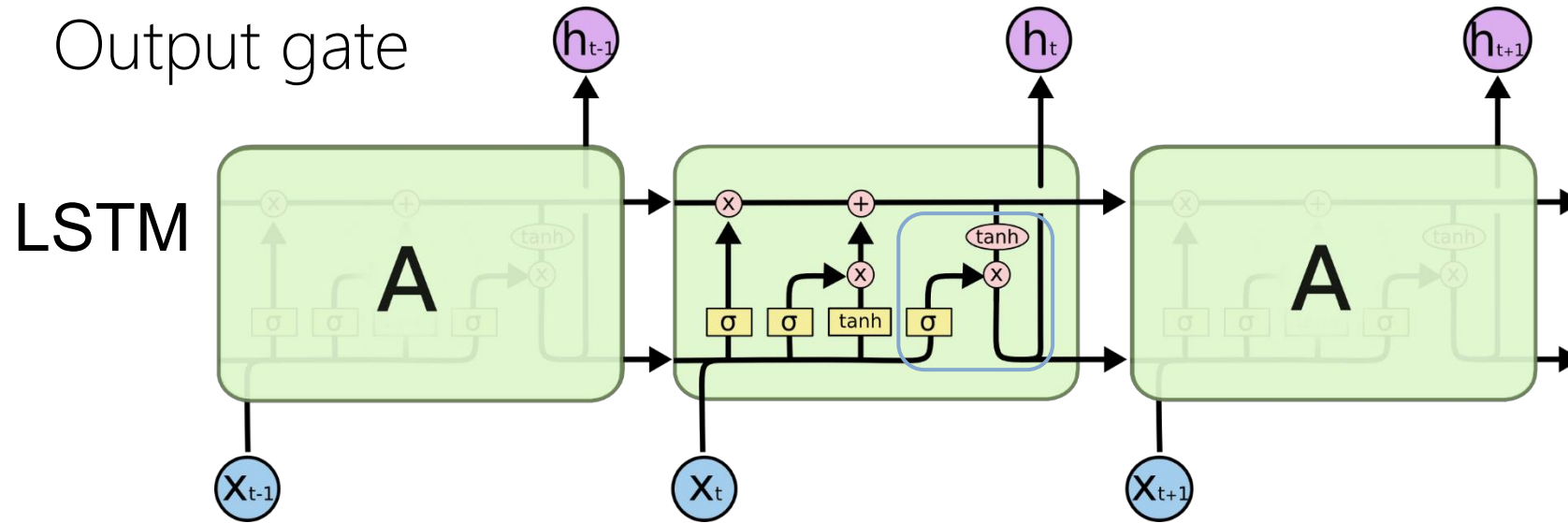
$$h_t = o_t * \tanh(C_t)$$



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

LSTM Images from: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

# Long Short-Term Memory



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

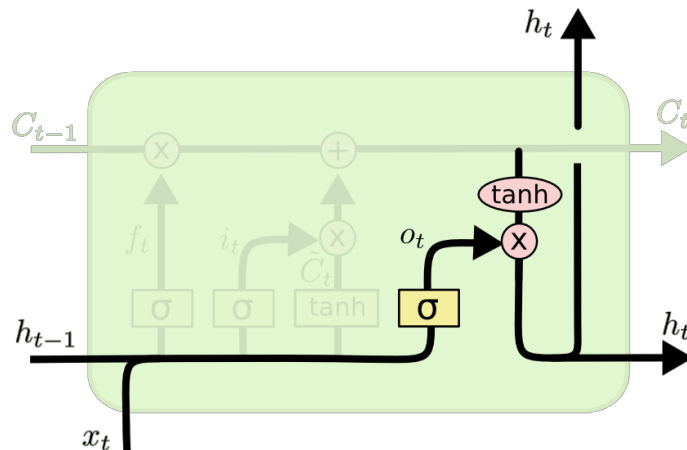
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

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

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

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



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

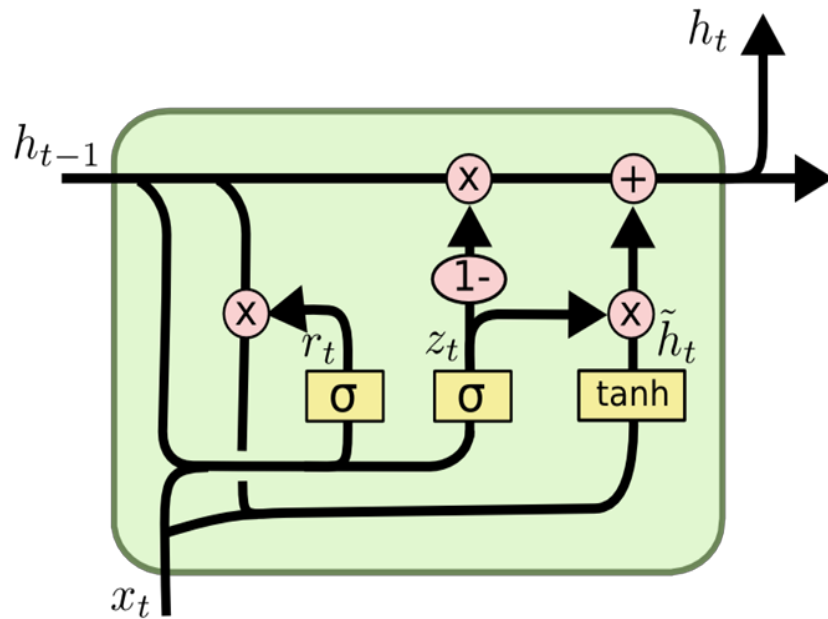
$$h_t = o_t * \tanh(C_t)$$

LSTM Images from: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>



# Gated Recurrent Unit (GRU)

It combines the forget and input gates into a single “update gate.” It also merges the cell state and hidden state, and makes some other changes.



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

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

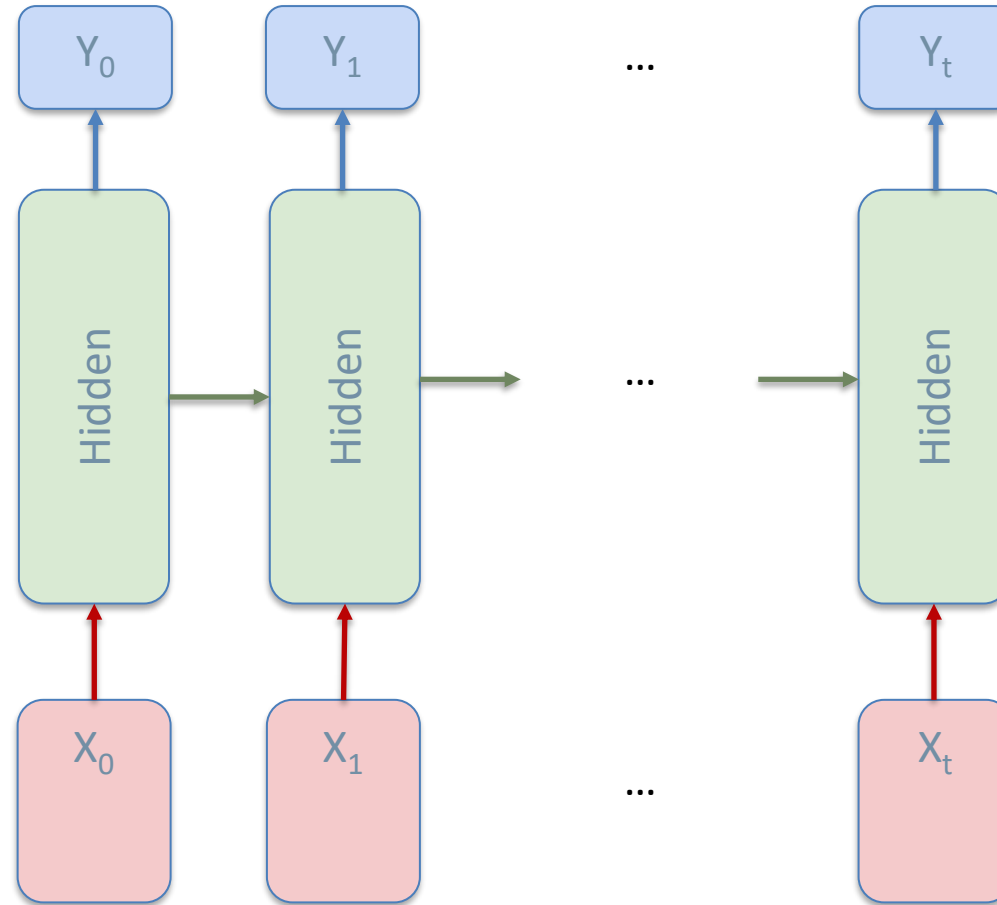
$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

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

LSTM Images from: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

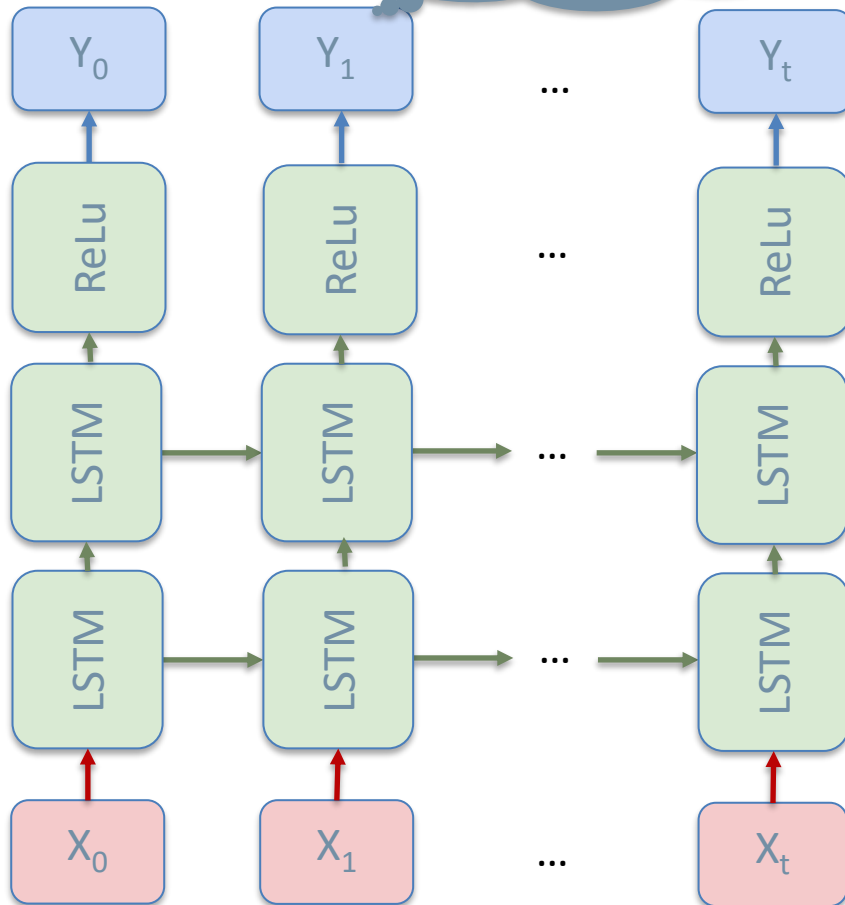
# LSTM Networks

You can build a computation graph with continuous transformations.



# Multiple Layers and Bidirectional LSTM Networks

A computational graph with continuous transformations.



# Tips & Tricks

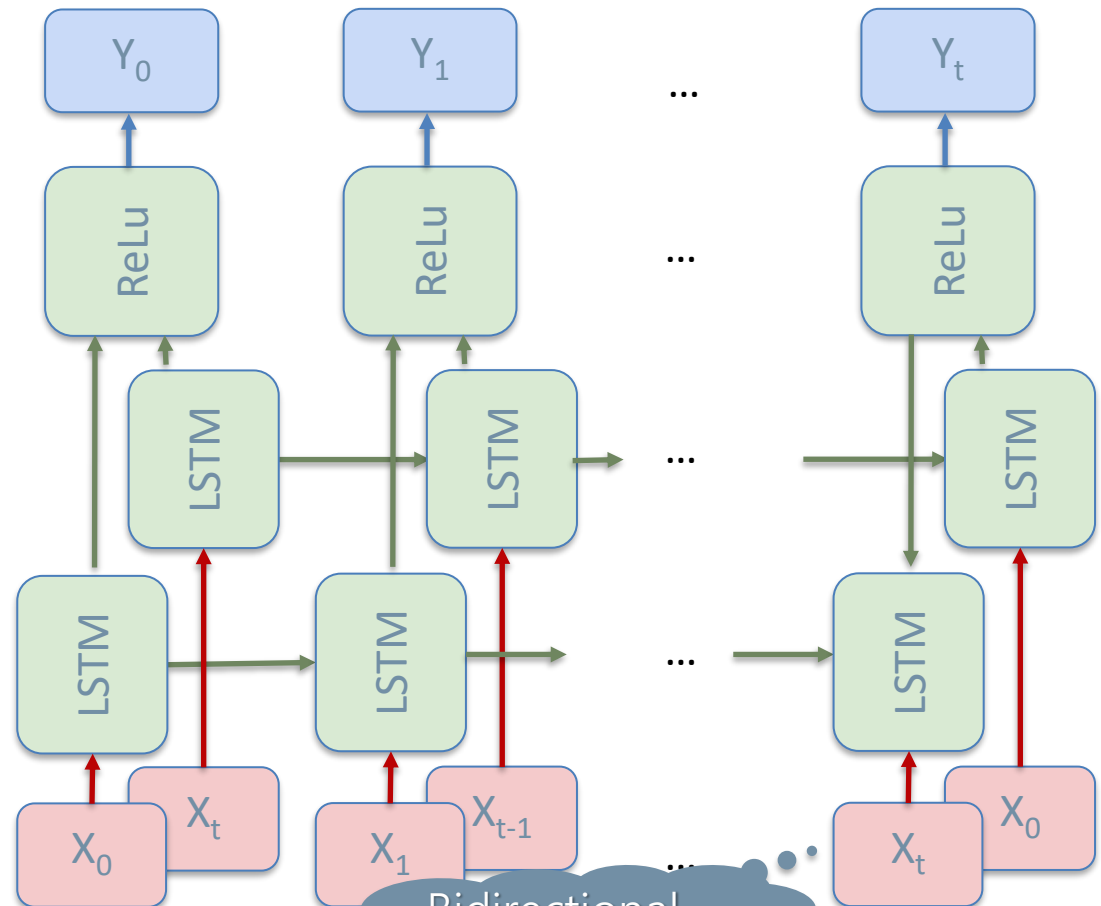
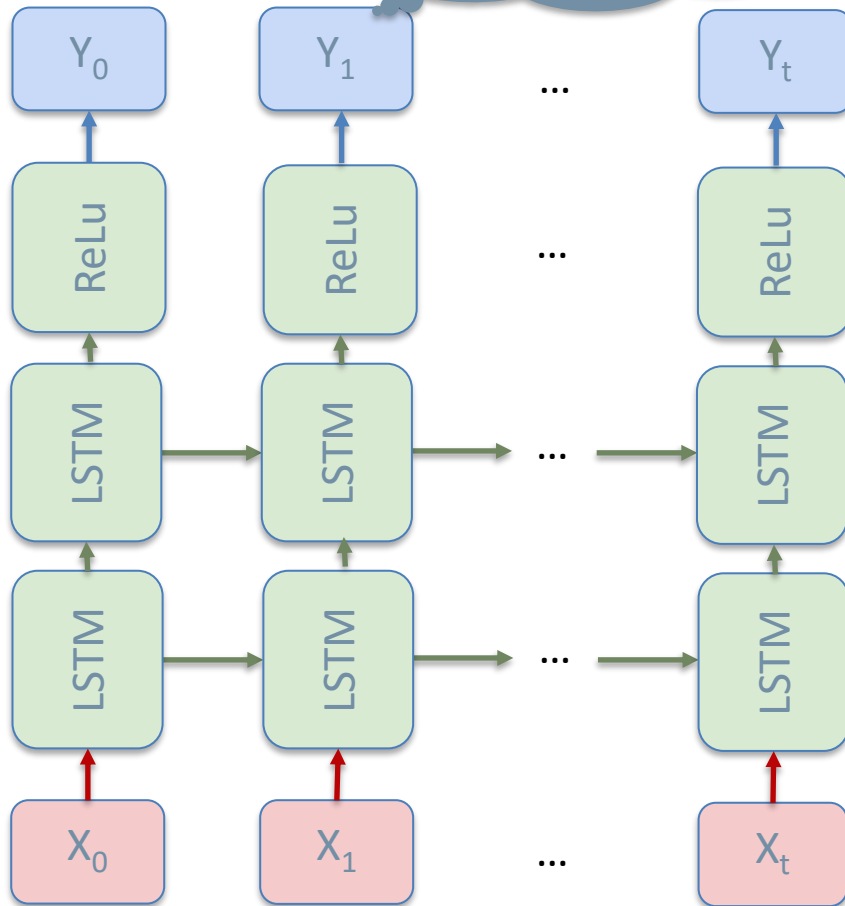
When conditioning on full input sequence Bidirectional RNNs exploit it:

- Have one RNNs traverse the sequence left-to-right
- Have another RNN traverse the sequence right-to-left
- Use concatenation of hidden layers as feature representation

# Multiple Layers and Bidirectional LSTM Networks

A computational graph with continuous transformations.

Hierarchical representation



Bidirectional processing

# Tips & Tricks

When conditioning on full input sequence Bidirectional RNNs exploit it:

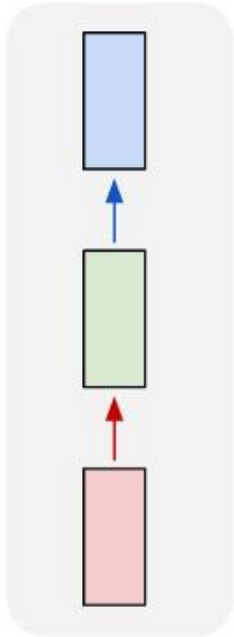
- Have one RNNs traverse the sequence left-to-right
- Have another RNN traverse the sequence right-to-left
- Use concatenation of hidden layers as feature representation

When initializing RNN we need to specify the initial state

- Could initialize them to a fixed value (such as 0)
- Better to treat the initial state as learned parameters
  - Start off with random guesses of the initial state values
  - Backpropagate the prediction error through time all the way to the initial state values and compute the gradient of the error with respect to these
  - Update these parameters by gradient descent

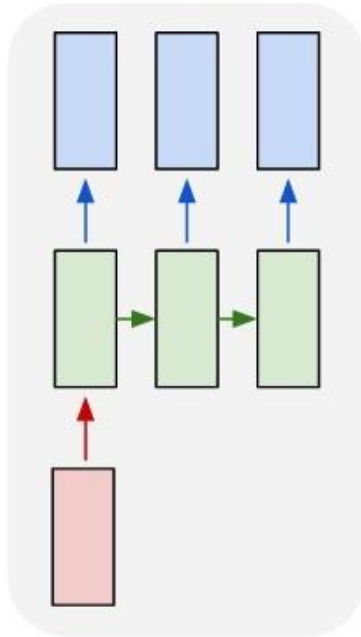
# Sequential Data Problems

one to one



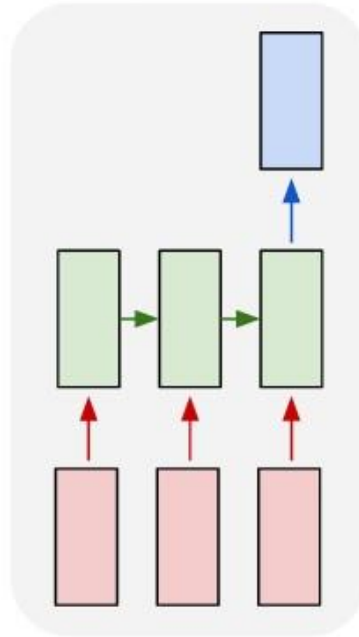
**Fixed-sized input to fixed-sized output** (e.g. image classification)

one to many



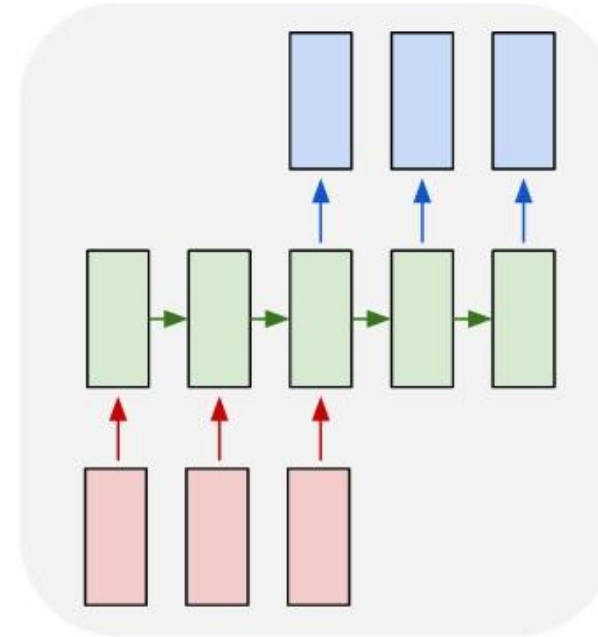
**Sequence output** (e.g. image captioning takes an image and outputs a sentence of words).

many to one



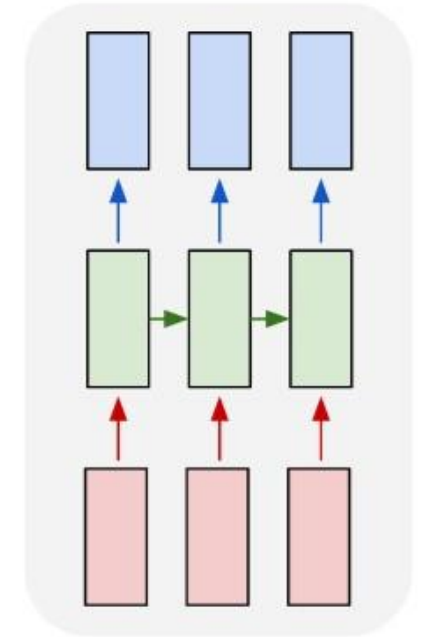
**Sequence input** (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).

many to many



**Sequence input and sequence output** (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French)

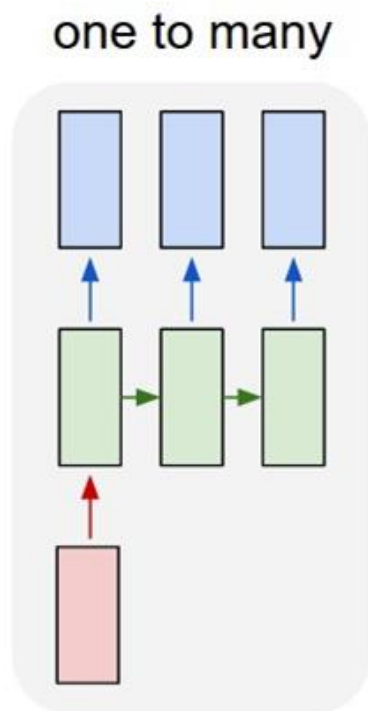
many to many



**Synced sequence input and output** (e.g. video classification where we wish to label each frame of the video)

# Sequence to Sequence Learning Examples (1/3)

Image Captioning: input a single image and get a series or sequence of words as output which describe it. The image has a fixed size, but the output has varying length.



**A person riding a motorcycle on a dirt road.**



**Two dogs play in the grass.**



**A group of young people playing a game of frisbee.**



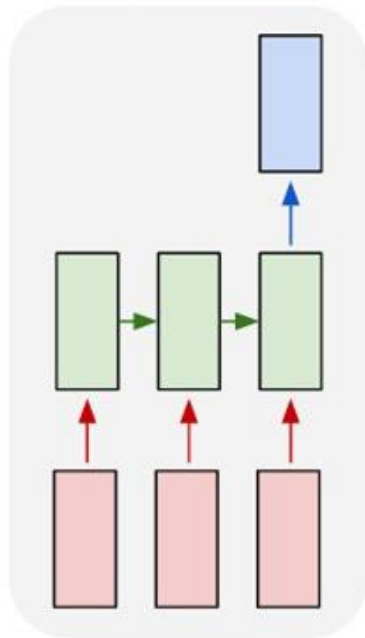
**Two hockey players are fighting over the puck.**



# Sequence to Sequence Learning Examples (2/3)

Sentiment Classification/Analysis: input a sequence of characters or words, e.g., a tweet, and classify the sequence into positive or negative sentiment. Input has varying lengths; output is of a fixed type and size.

many to one



I really like the color of my new Iphone

+

I didn't really enjoy the camera of my Iphone

-

# Sequence to Sequence Learning Examples (3/3)

Language Translation: having some text in a particular language, e.g., English, we wish to translate it in another, e.g., French. Each language has its own semantics and it has varying lengths for the same sentence.

