Artificial Neural Networks and Deep Learning

- Recurrent Neural Networks-

Matteo Matteucci, PhD (matteo.matteucci@polimi.it)
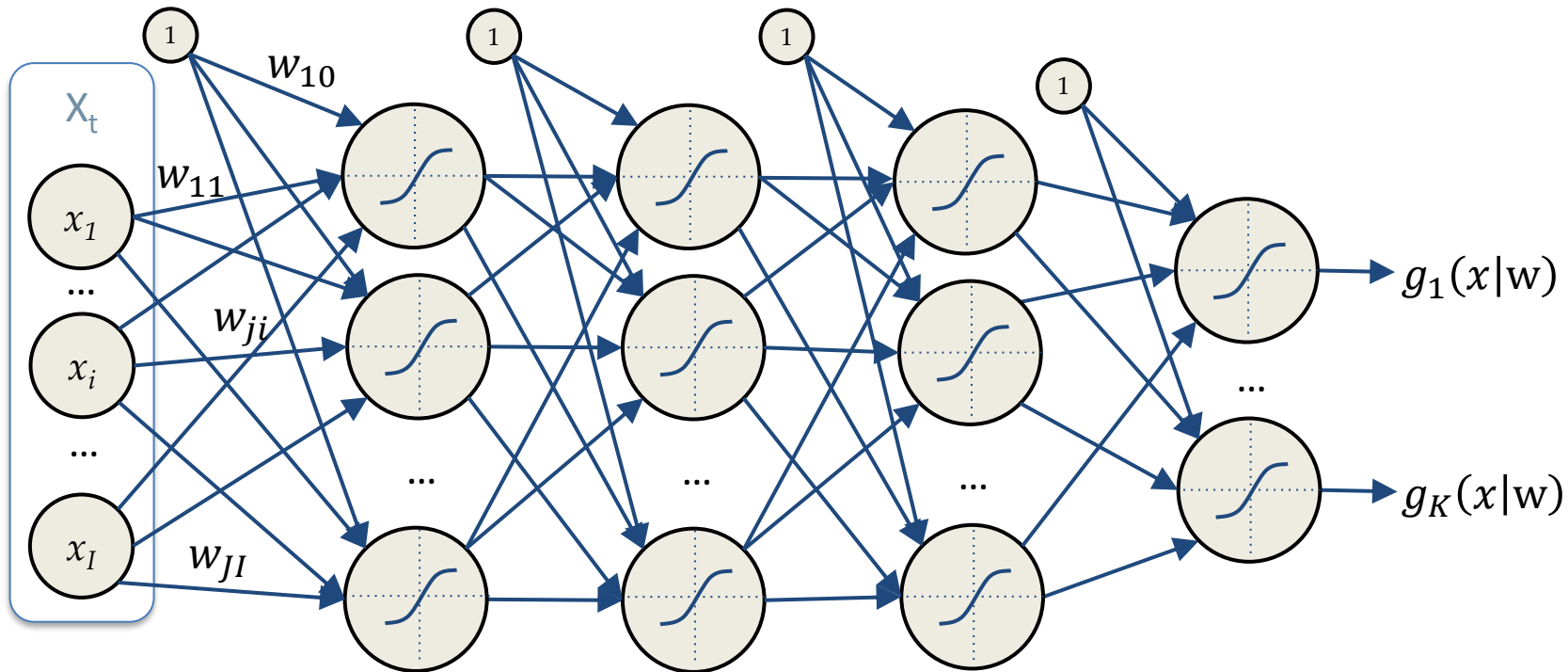*Artificial Intelligence and Robotics Laboratory*
Politecnico di Milano

AIRLAB
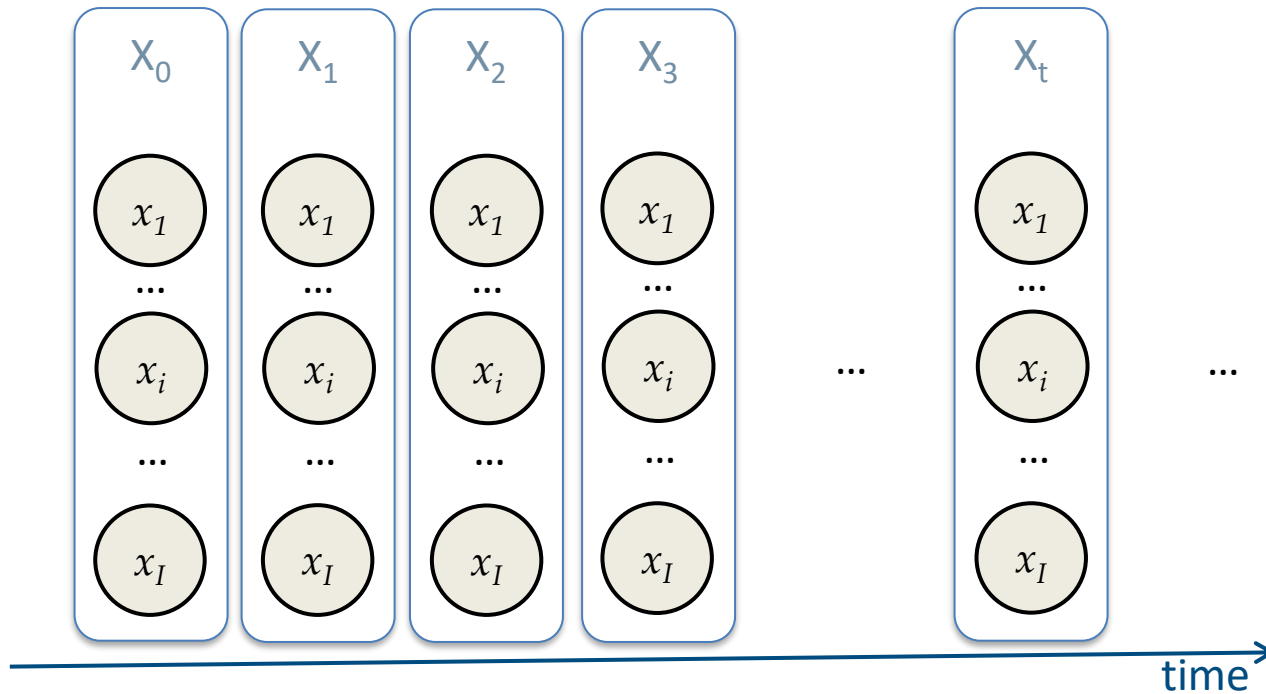ARTIFICIAL INTELLIGENCE AND ROBOTICS LAB

Sequence Modeling

So far we have considered only «static» datasets

Sequence Modeling

So far we have considered only «static» datasets
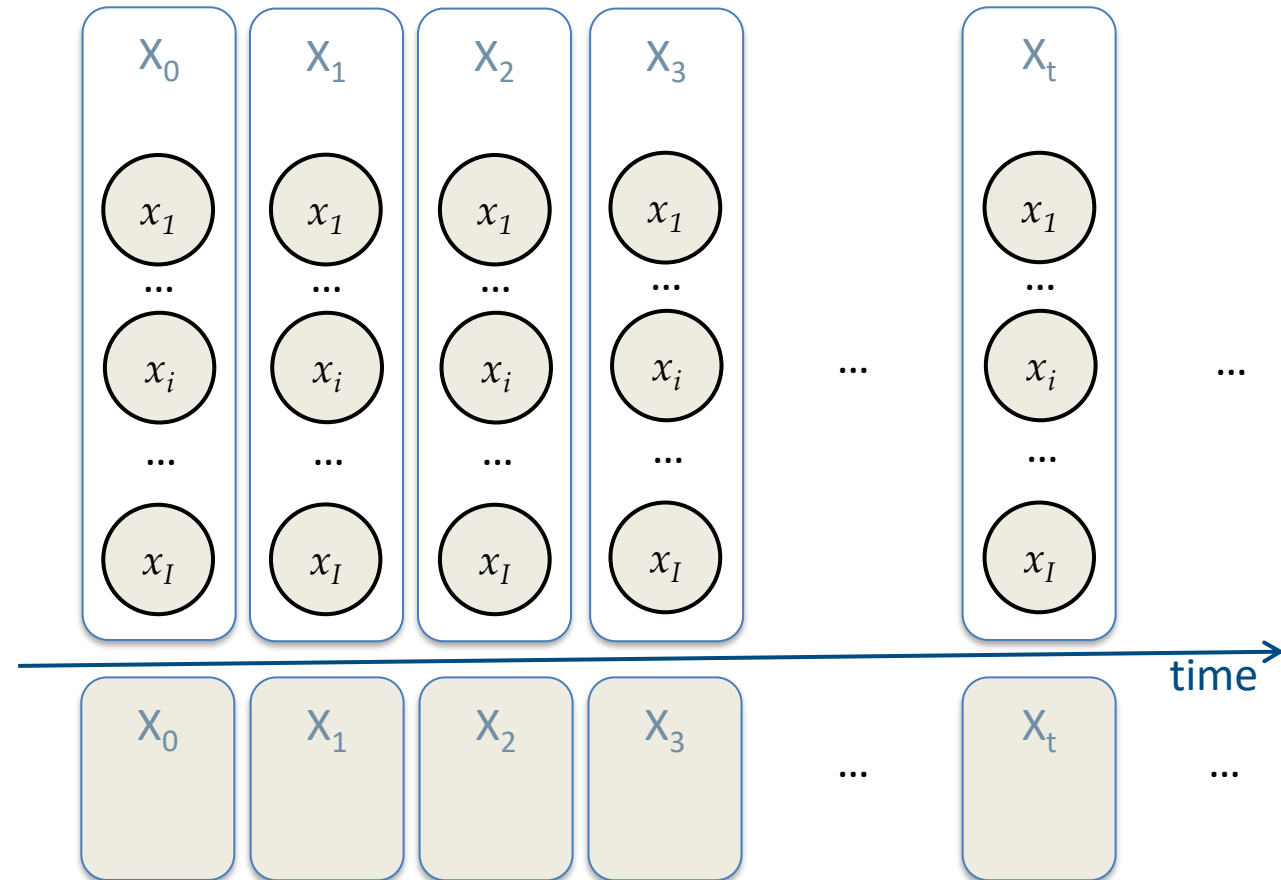
Sequence Modeling

Different ways to deal with «dynamic» data:

Memoryless models (fixed lag):
- Autoregressive models
- Feedforward neural networks
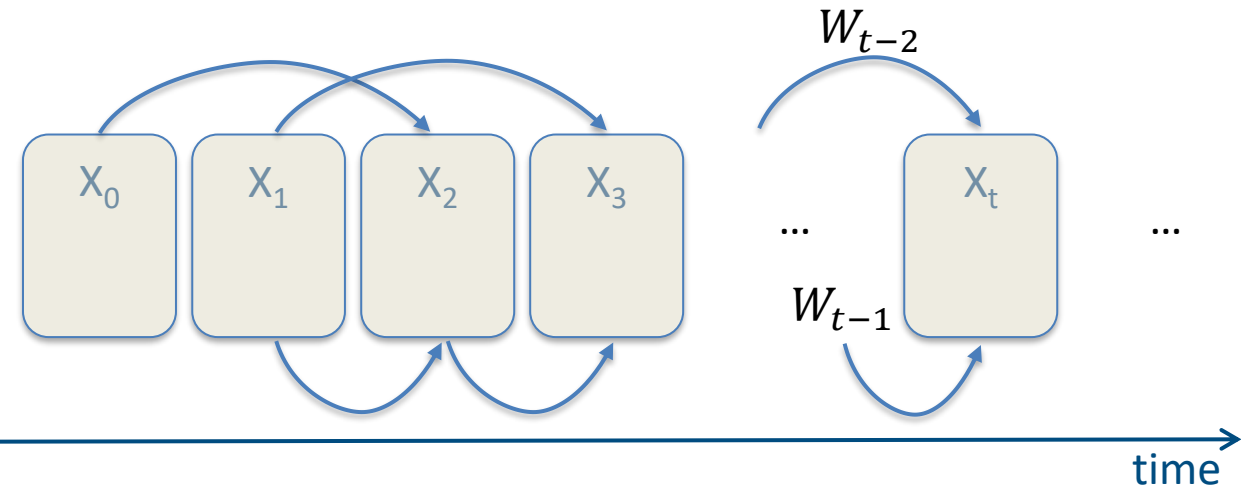
Models with memory (unlimited):
- Linear dynamical systems
- Hidden Markov models
- Recurrent Neural Networks
- …

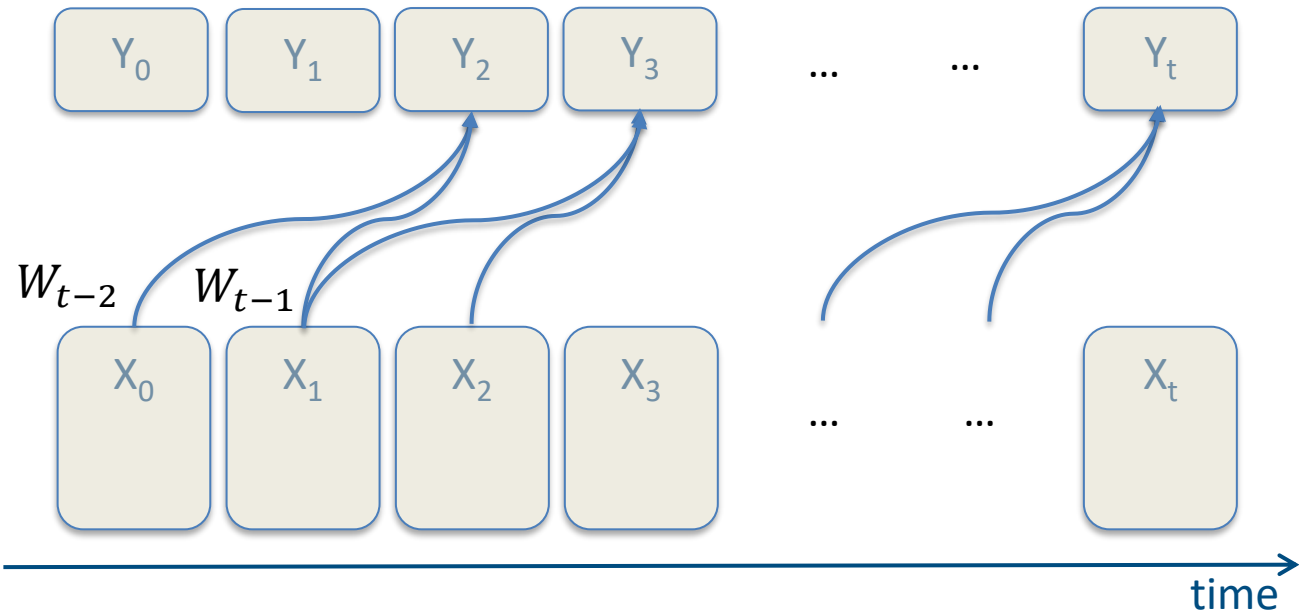# Memoryless Models for Sequences (1/2)

Autoregressive models

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

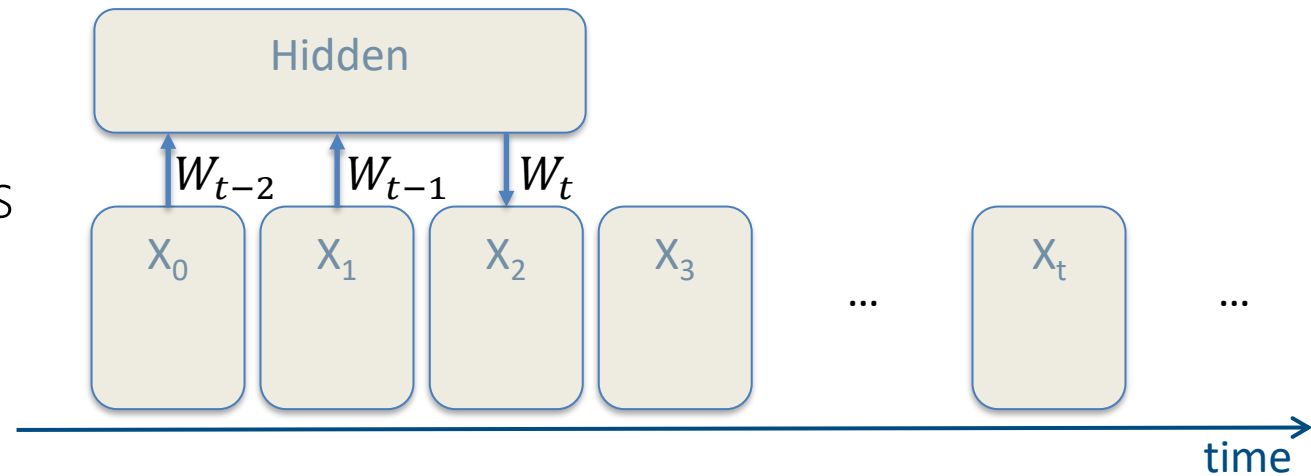## Linear models with fixed lag

- 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

Hidden

W_{t-2} W_{t-1} W_t

X_0 X_1 X_2 X_3 … X_t …

time

Feed forward neural networks with delays

- Predict the next output from previous inputs and previous outputs using «delay taps»
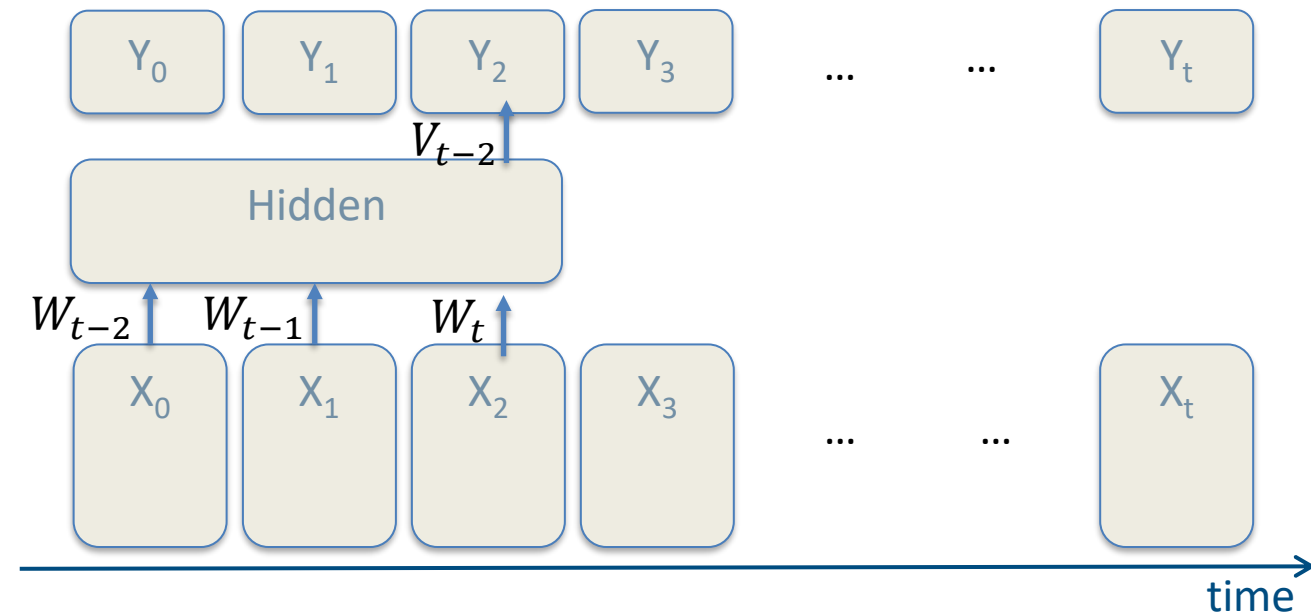
Y_0 Y_1 Y_2 Y_3 …   … Y_t

V_{t-2}

Hidden

W_{t-2} W_{t-1} W_t

X_0 X_1 X_2 X_3 …   … X_t

time

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*
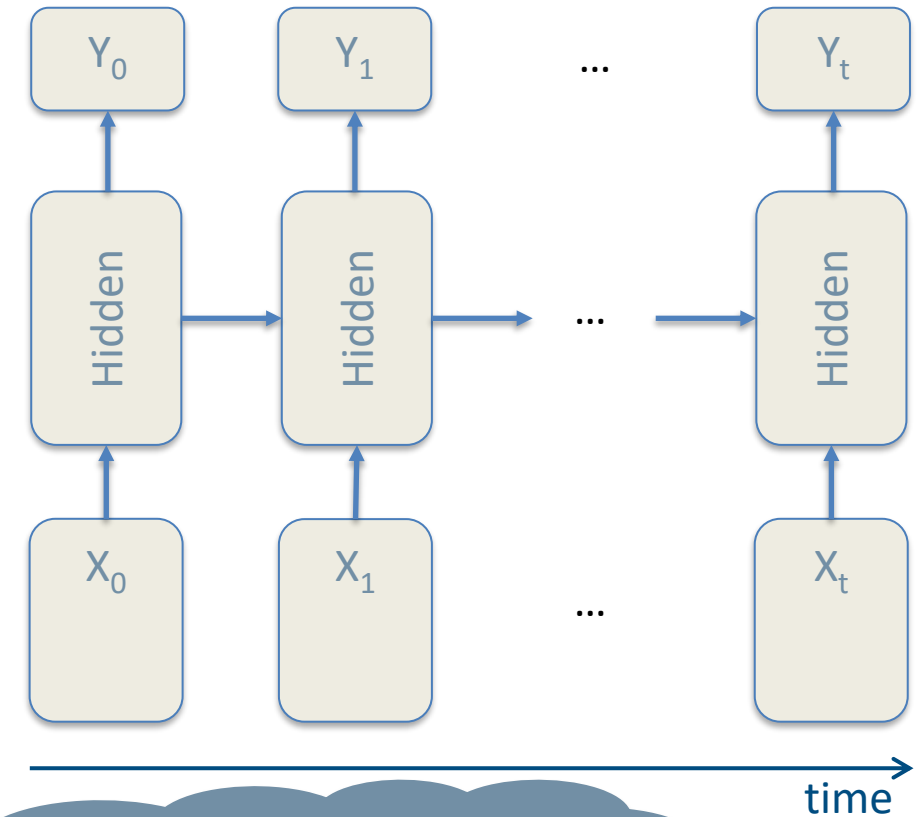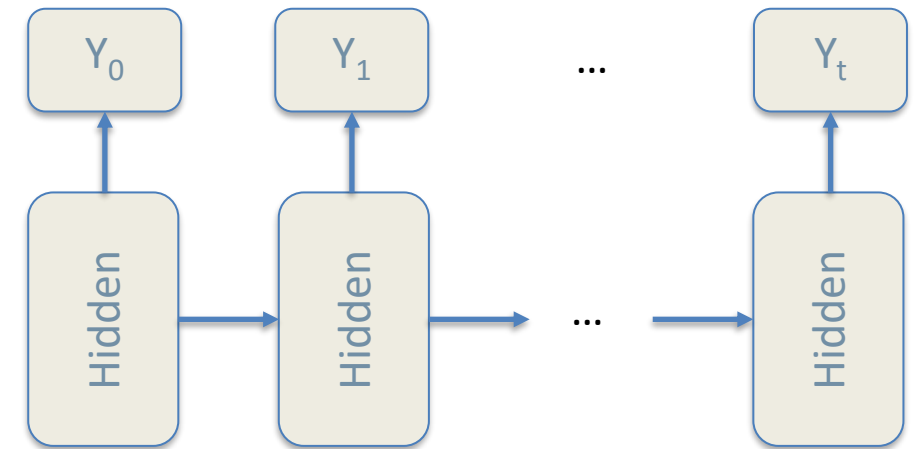
time

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.*

Y_0 Y_1 ... Y_t

Hidden   Hidden   ...   Hidden

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)

Deterministic systems ...

$h_j^t(x^t, W^{(1)}, c^{t-1}, V^{(1)})$

w_{11}

x_1

w_{ji}

x_i

x_I

w_{Ji}

$g^t(x|w)$

c_1^{t-1}

c_B^{t-1}

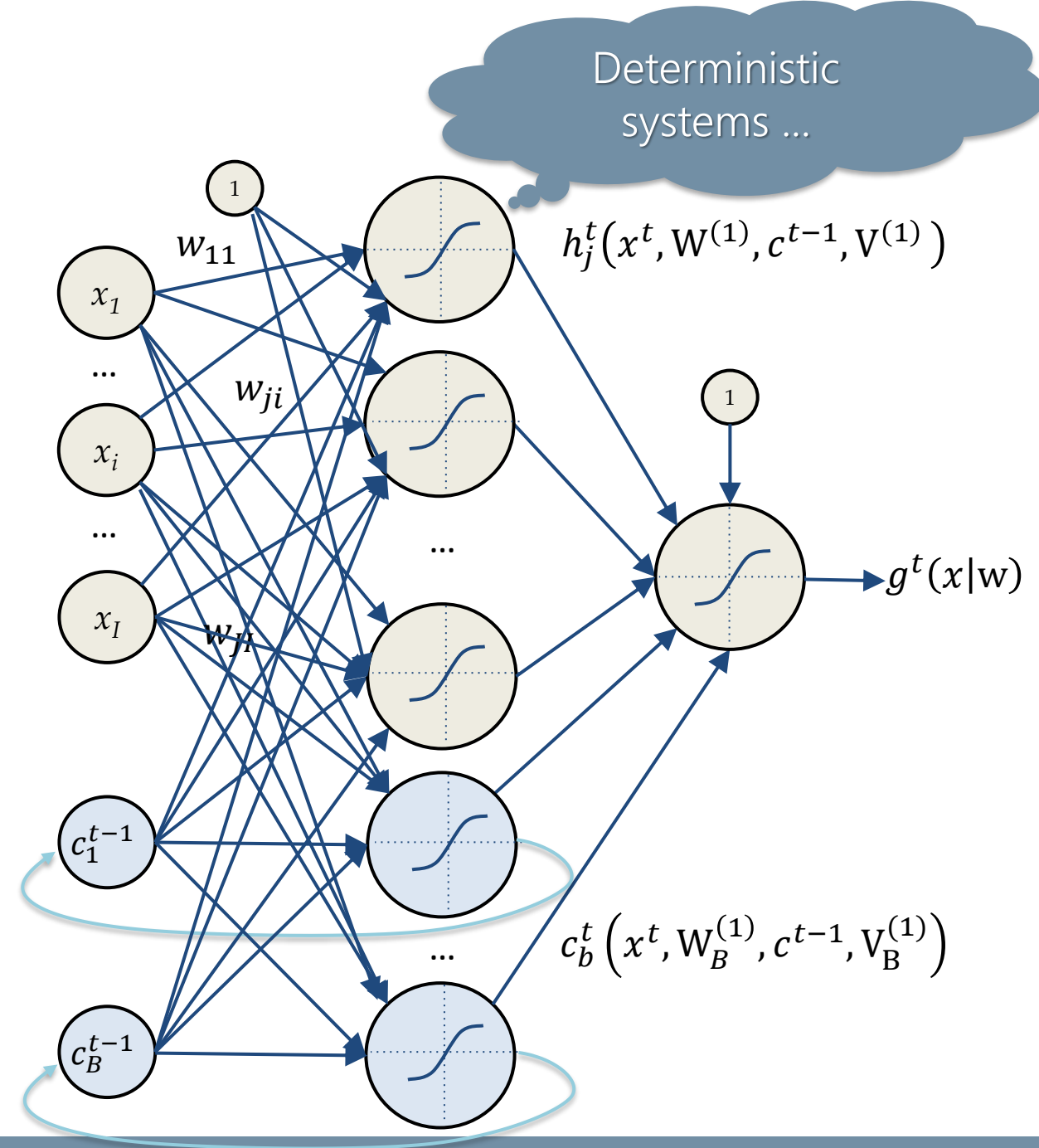$c_b^t\left(x^t, W_B^{(1)}, c^{t-1}, V_B^{(1)}\right)$
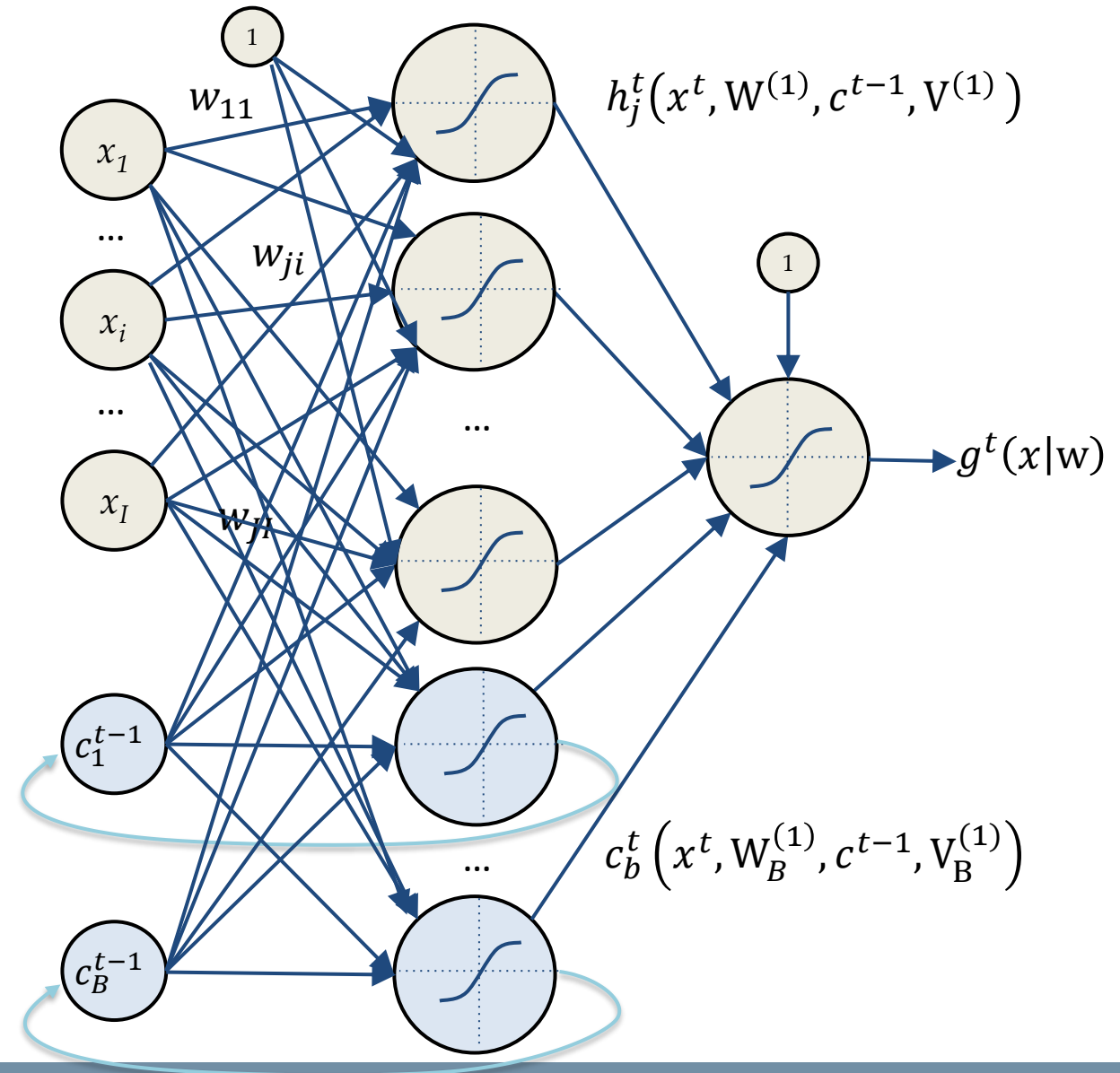
Recurrent Neural networks

Memory via recurrent connections:

- Distributed hidden state allows to store information efficiently

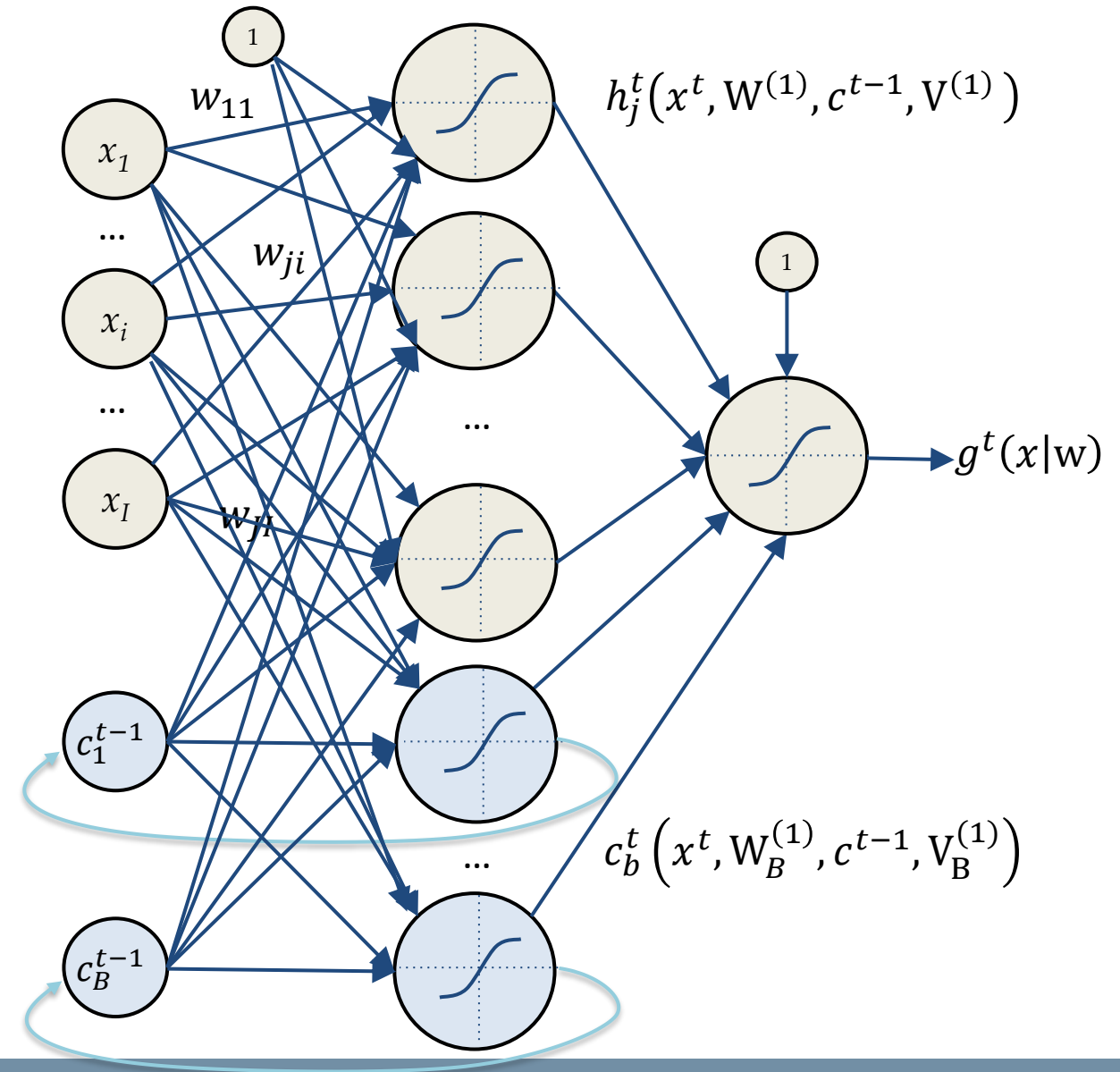- Non-linear dynamics allows complex hidden state updates

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

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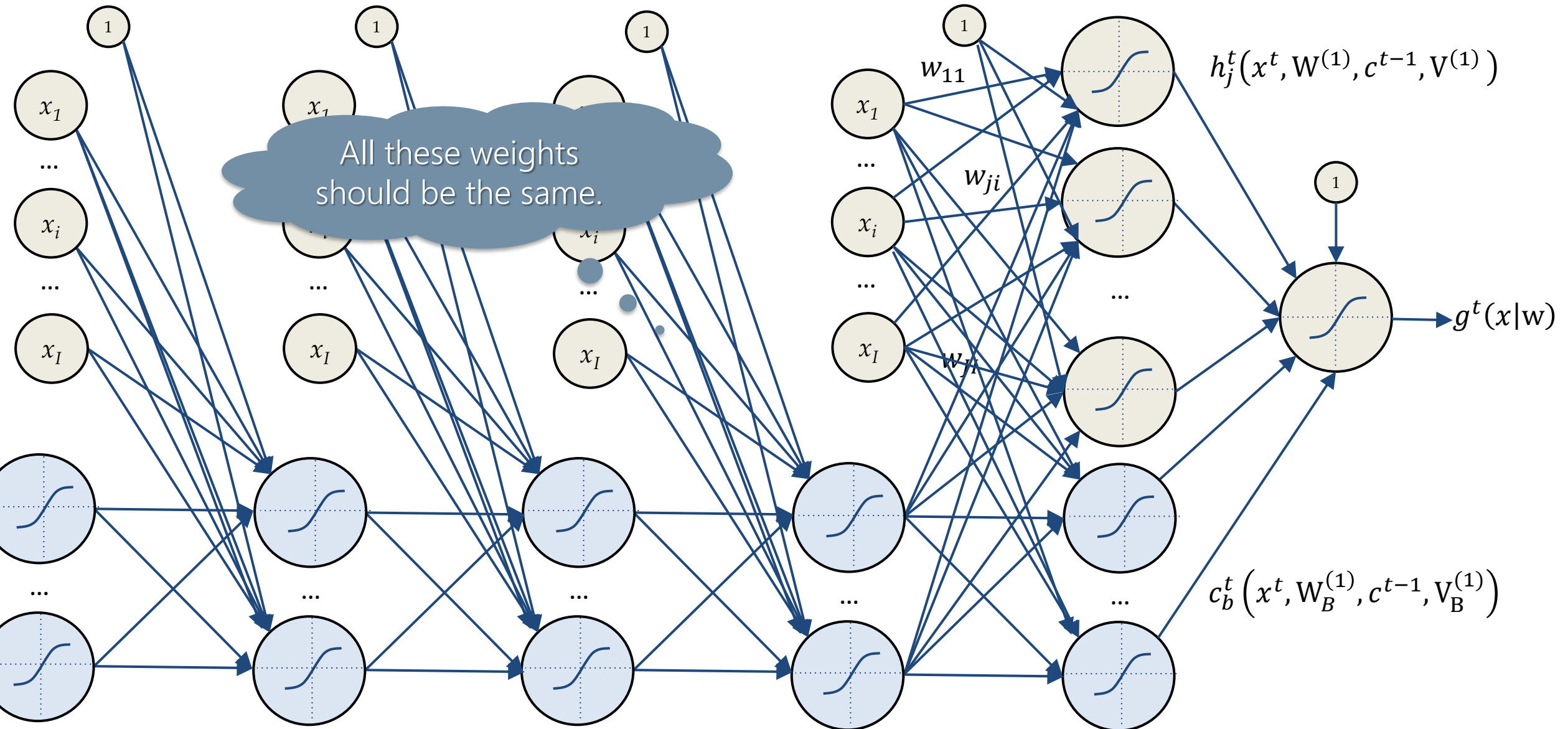
$$c_b^t(\cdot) = c_b\left(\sum_{j=0}^{J} w_{bi}^{(1)} \cdot x_{i,n}^t + \sum_{b'=0}^{B} v_{bb'}^{(1)} \cdot c_{b'}^{t-1}\right)$$
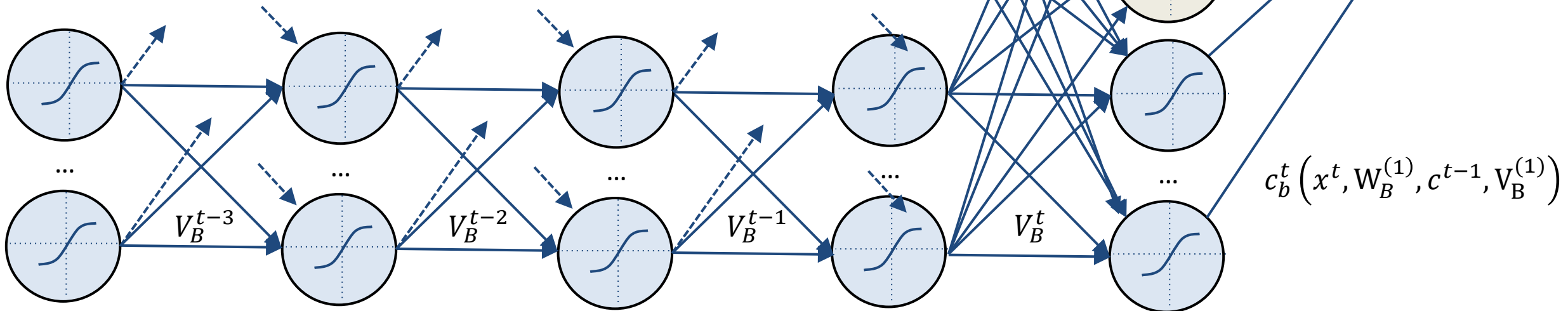

$h_j^t(x^t, \mathrm{W}^{(1)}, c^{t-1}, \mathrm{V}^{(1)})$

$g^t(x|\mathrm{w})$

$c_b^t\left(x^t, \mathrm{W}_B^{(1)}, c^{t-1}, \mathrm{V}_B^{(1)}\right)$

Backpropagation Through Time

Backpropagation Through Time

All these weights should be the same.

$$h_j^t(x^t, W^{(1)}, c^{t-1}, V^{(1)})$$

$$w_{11}$$

$$w_{ji}$$

$$g^t(x|w)$$

$$c_b^t\left(x^t, W_B^{(1)}, c^{t-1}, V_B^{(1)}\right)$$

Backpropagation Through Time

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

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

$$h_j^t(x^t, W^{(1)}, c^{t-1}, V^{(1)})$$

$$g^t(x|w)$$

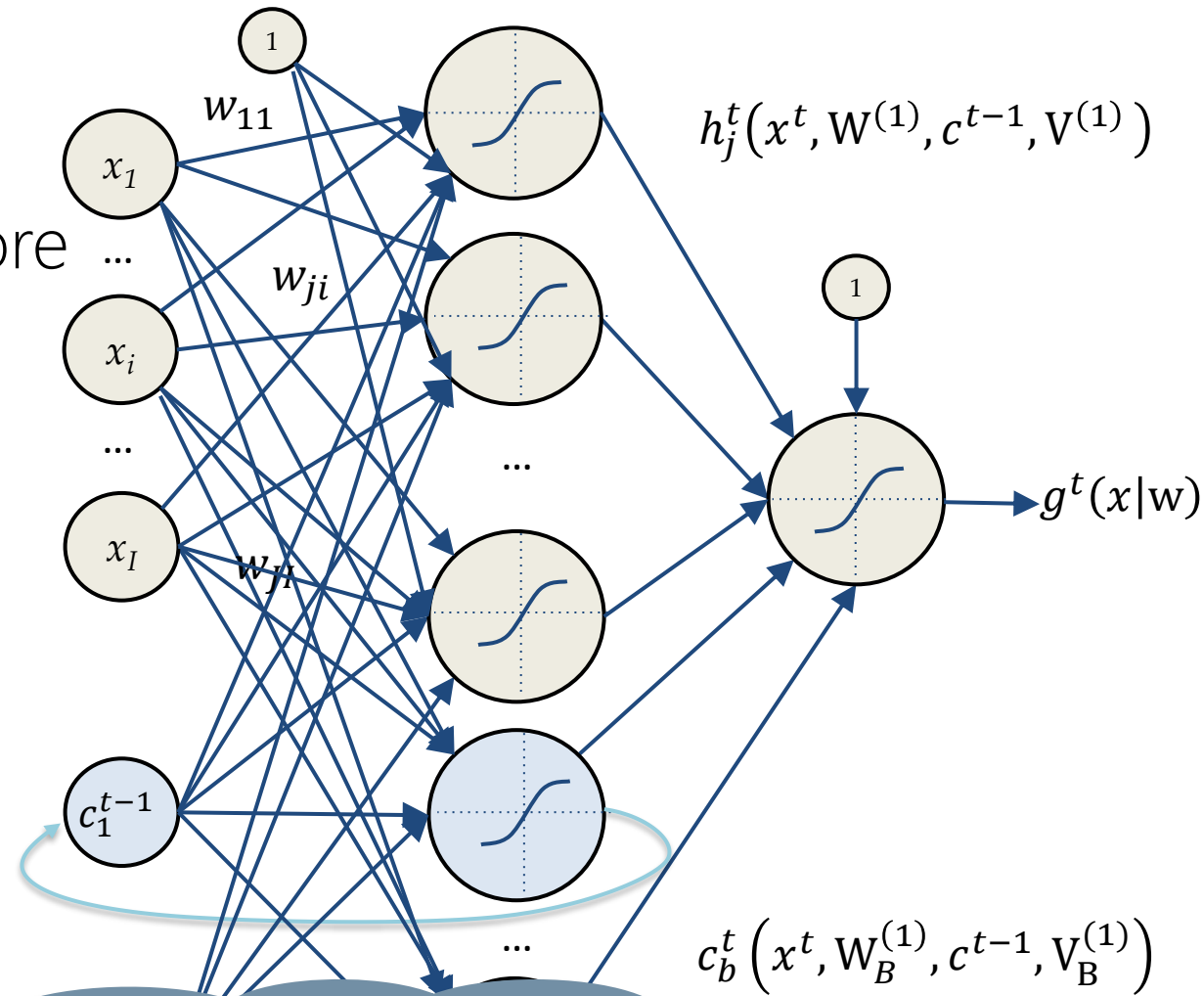$$c_b^t\left(x^t, W_B^{(1)}, c^{t-1}, V_B^{(1)}\right)$$

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 ...

w_{11}

x_1

w_{ji}

x_i

...

x_I

w_{JI}

c_1^{t-1}

...

$h_j^t(x^t, W^{(1)}, c^{t-1}, V^{(1)})$

$g^t(x|w)$

$c_b^t(x^t, W_B^{(1)}, c^{t-1}, V_B^{(1)})$

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:

$$h^t = h(v^{(1)} \cdot h^{t-1} + w^{(1)} \cdot x)$$

$$y^t = g(w^{(2)} \cdot h^t)$$
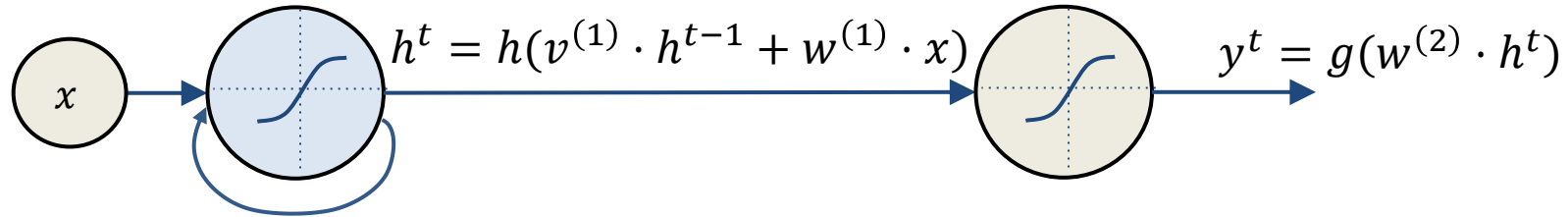
Backpropagation over an entire sequence S is computed as

$$\frac{\partial E}{\partial w} = \sum_{t=1}^{S} \frac{\partial E^t}{\partial w} \quad \Longrightarrow \quad \frac{\partial E^t}{\partial w} = \sum_{t=1}^{t} \frac{\partial E^t}{\partial y^t} \frac{\partial y^t}{\partial h^t} \frac{\partial h^t}{\partial h^k} \frac{\partial h^k}{\partial w} \quad \Longrightarrow \quad \frac{\partial h^t}{\partial h^k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k+1}^{t} v^{(1)} h'(v^{(1)} \cdot h^{i-1} + w^{(1)} \cdot x)$$
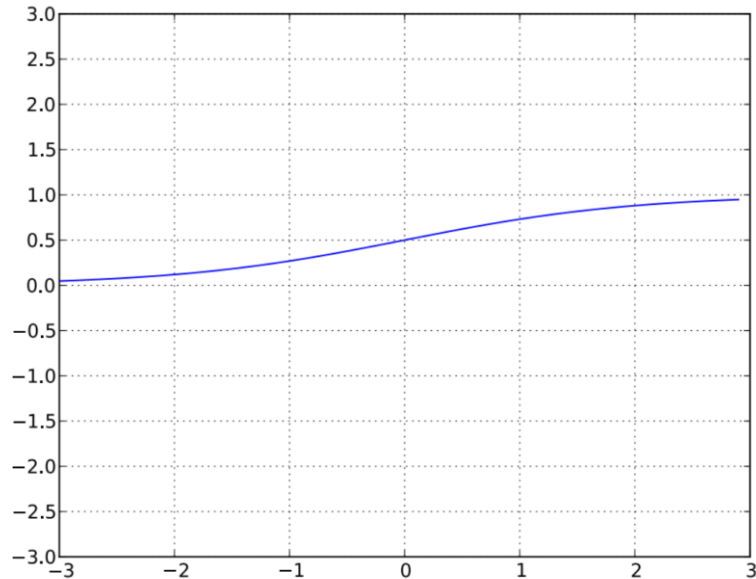
If we consider the norm of these terms

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

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

With Sigmoids and Tanh we have vanishing gradients
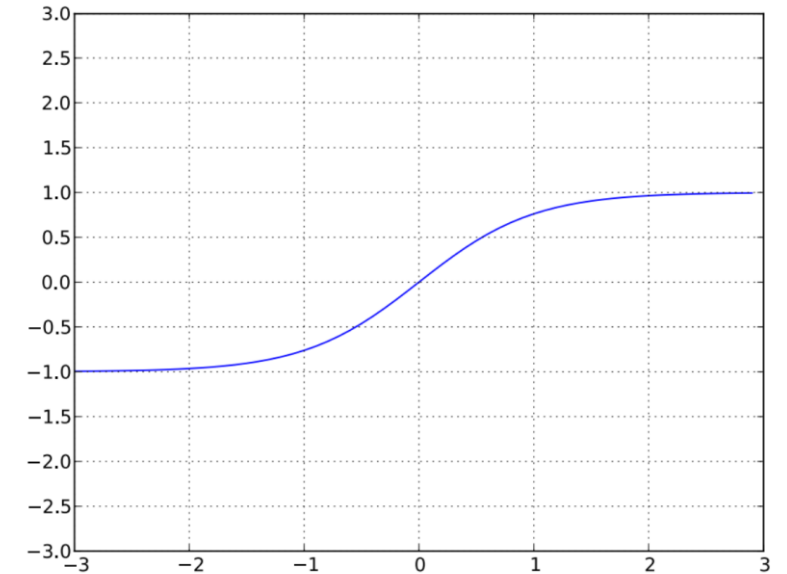
Which Activation Function?

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

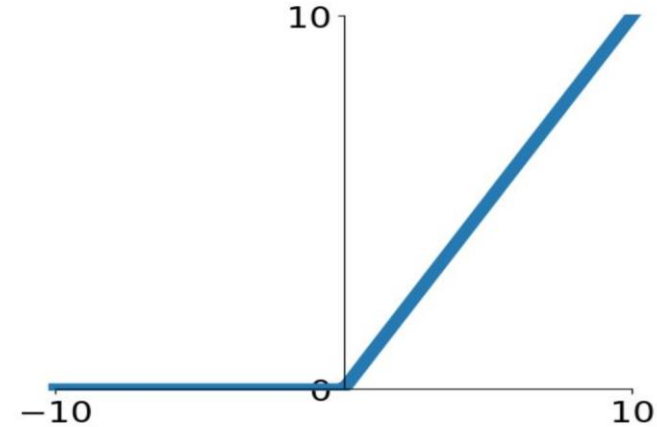$$g(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)}$$
$$g'(a) = 1 - g(a)^2$$

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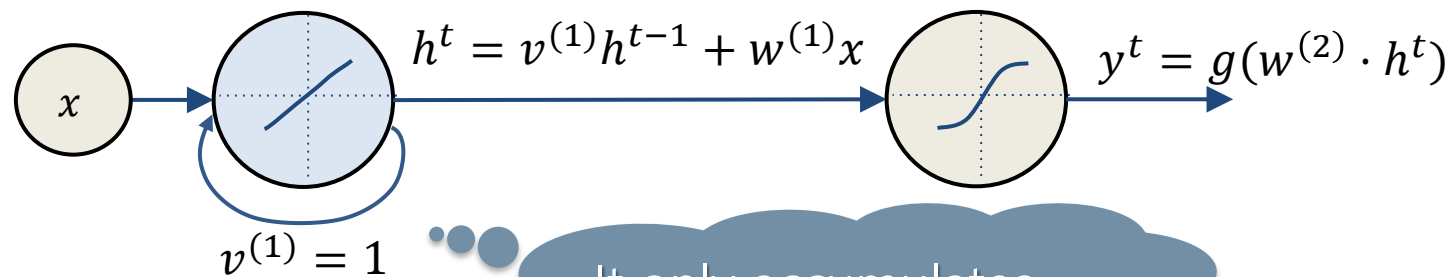
Dealing with Vanishing Gradient

Force all gradients to be either 0 or 1

$$g(a) = ReLu(a) = \max(0, a)$$

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

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

$$h^t = v^{(1)}h^{t-1} + w^{(1)}x \qquad y^t = g(w^{(2)} \cdot h^t)$$
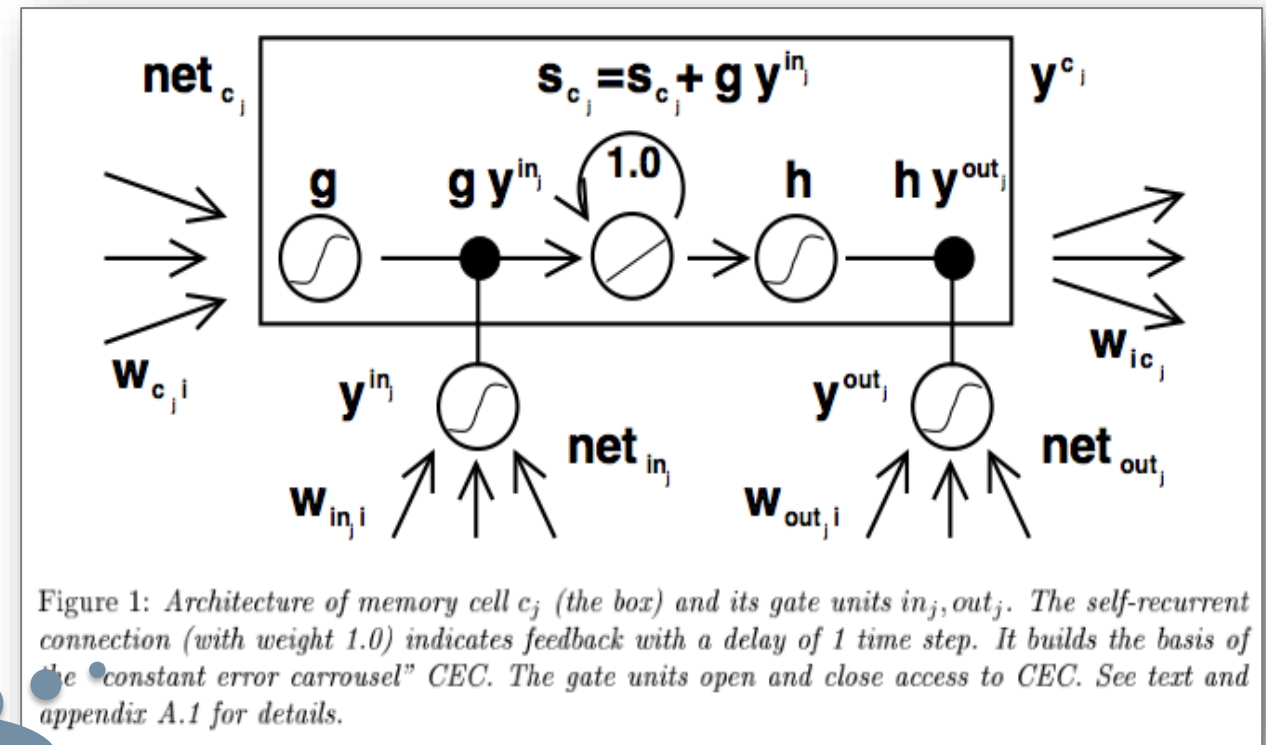
x

$v^{(1)} = 1$

It only accumulates the input ...

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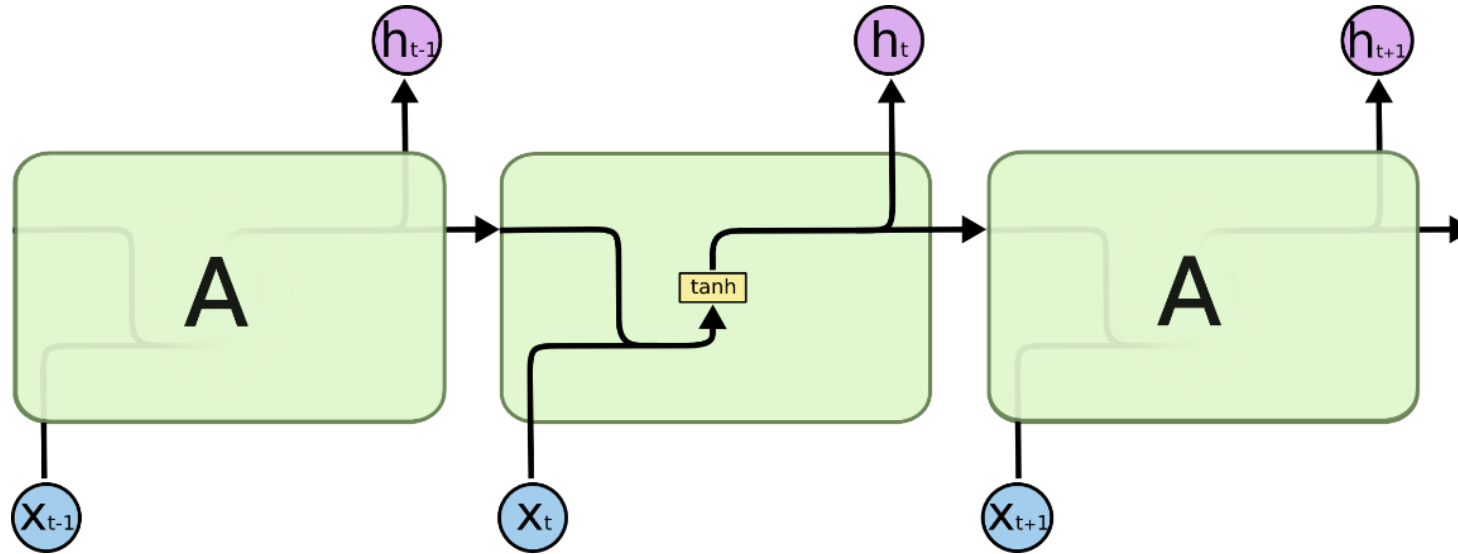
- Information gets into the cell whenever its "*write*" gate is on.
- The information stays in the cell so long as its "*keep*" gate is on.
- Information is read from the cell by turning on its "*read*" gate.

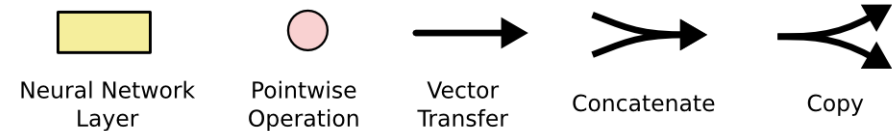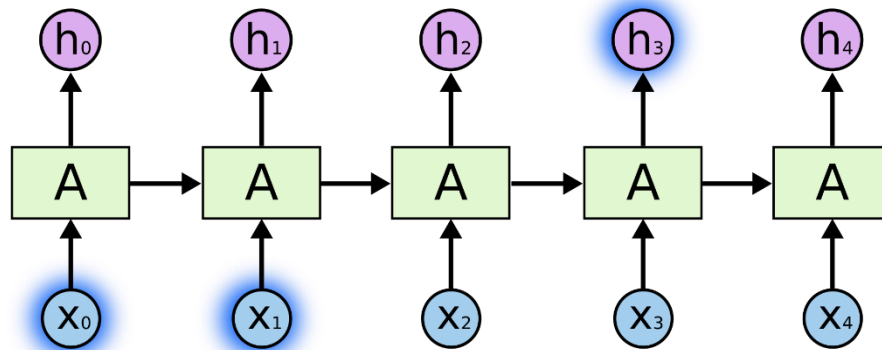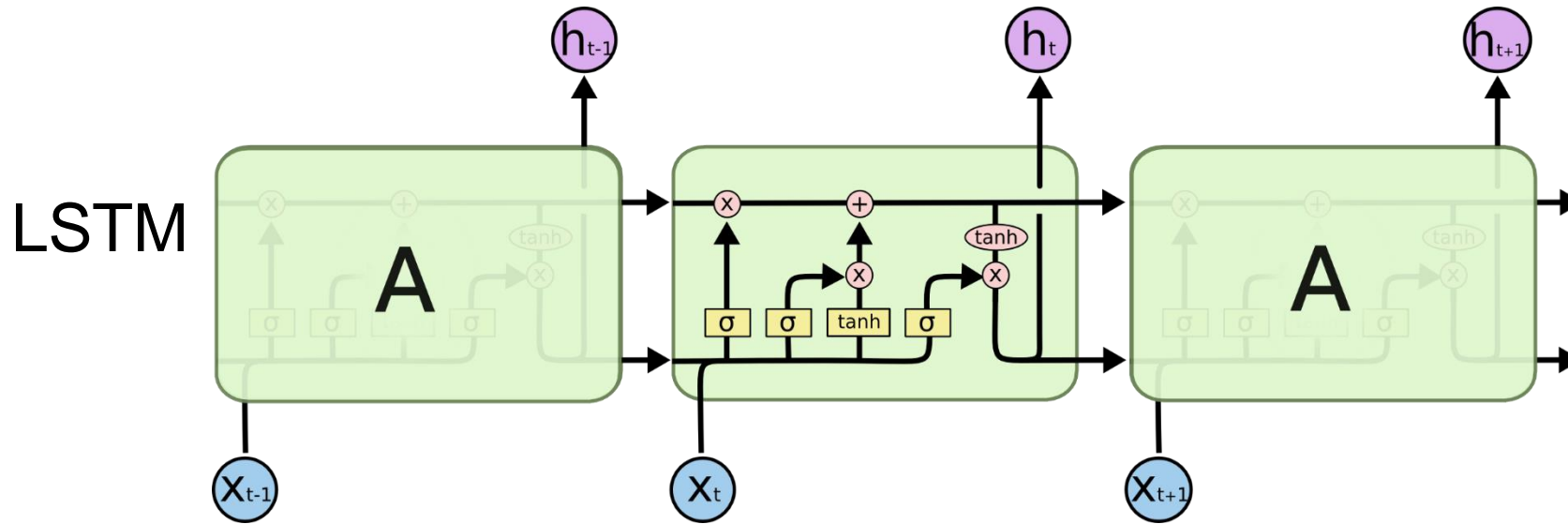Can backpropagate through this since the loop has fixed weight.

Figure 1: *Architecture of memory cell c_j (the box) and its gate units in_j, out_j. The self-recurrent connection (with weight 1.0) indicates feedback with a delay of 1 time step. It builds the basis of the "constant error carrousel" CEC. The gate units open and close access to CEC. See text and appendix A.1 for details.*

RNN vs. LSTM

RNN

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

$$b_h^t = \theta_h(a_h^t)$$

| 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

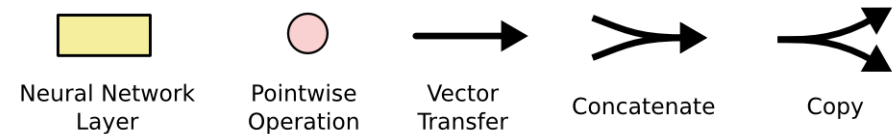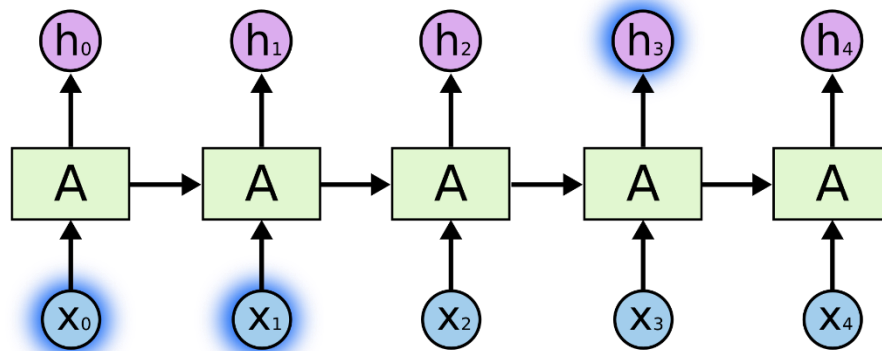$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

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

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$
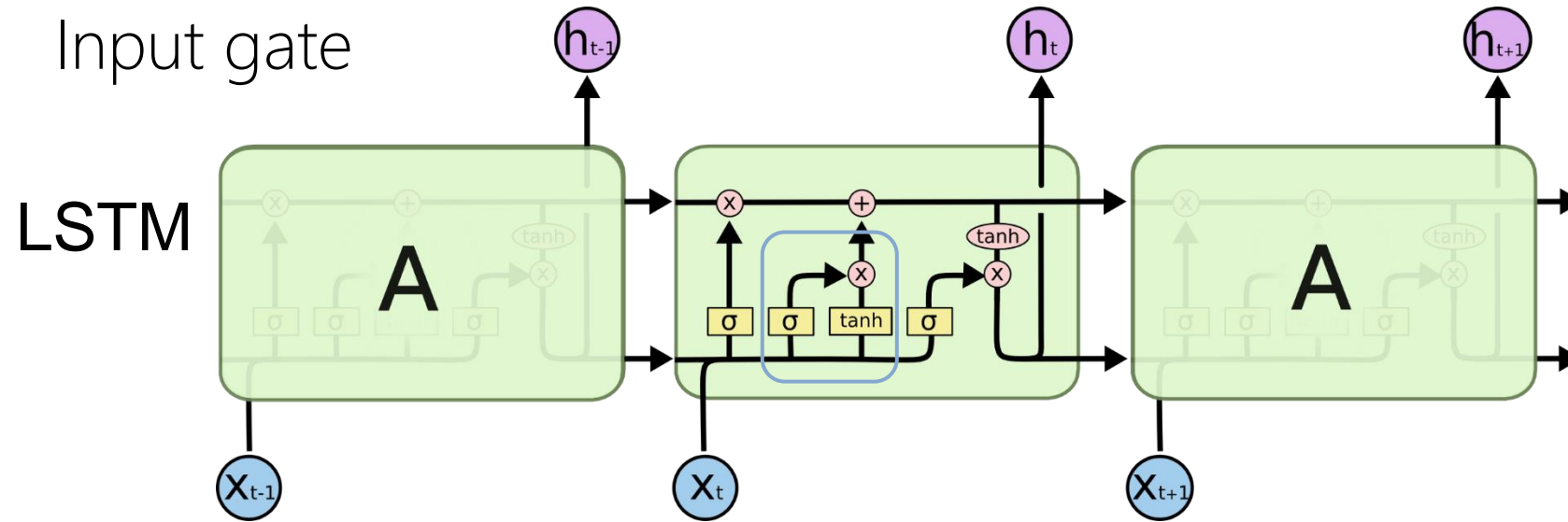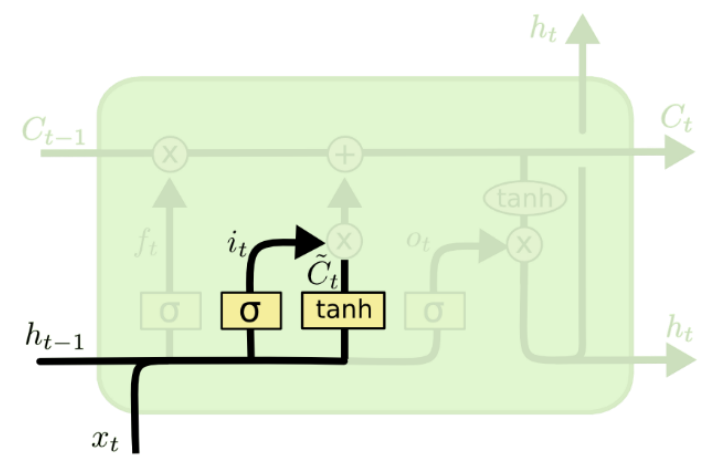
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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

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

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

Long Short-Term Memory

Input gate

LSTM

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

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

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

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

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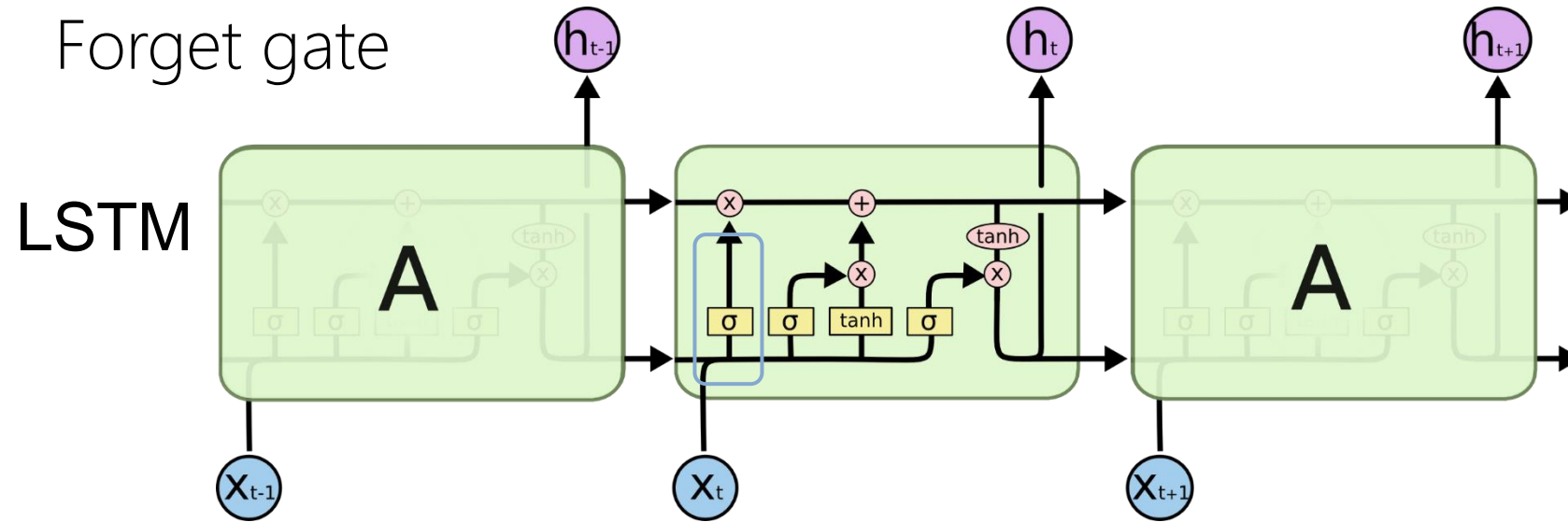
$$h_t = o_t * \tanh\left(C_t\right)$$

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

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

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

Long Short-Term Memory

Forget gate

LSTM

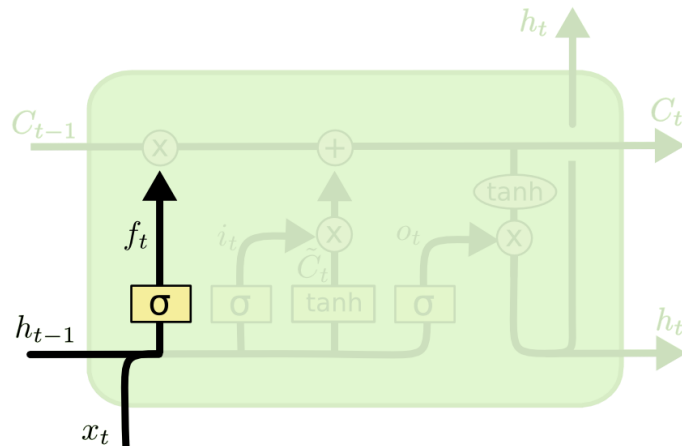$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

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

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

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
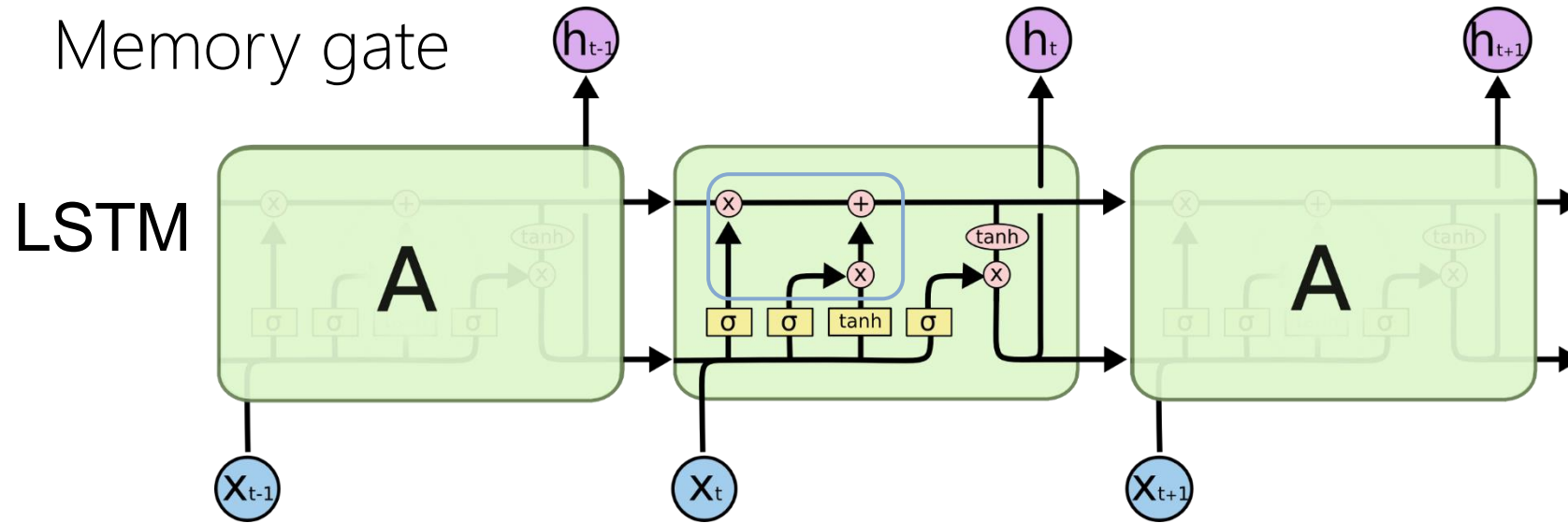
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

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

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

Long Short-Term Memory

Memory gate

LSTM

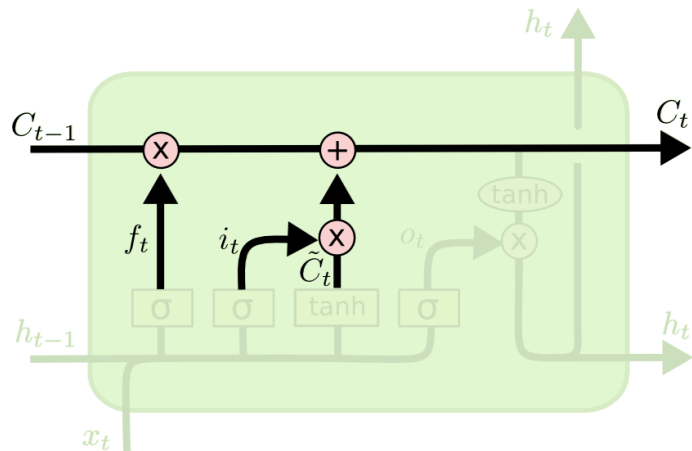$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

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

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

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

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

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

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

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

Long Short-Term Memory

Output gate

LSTM

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

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

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

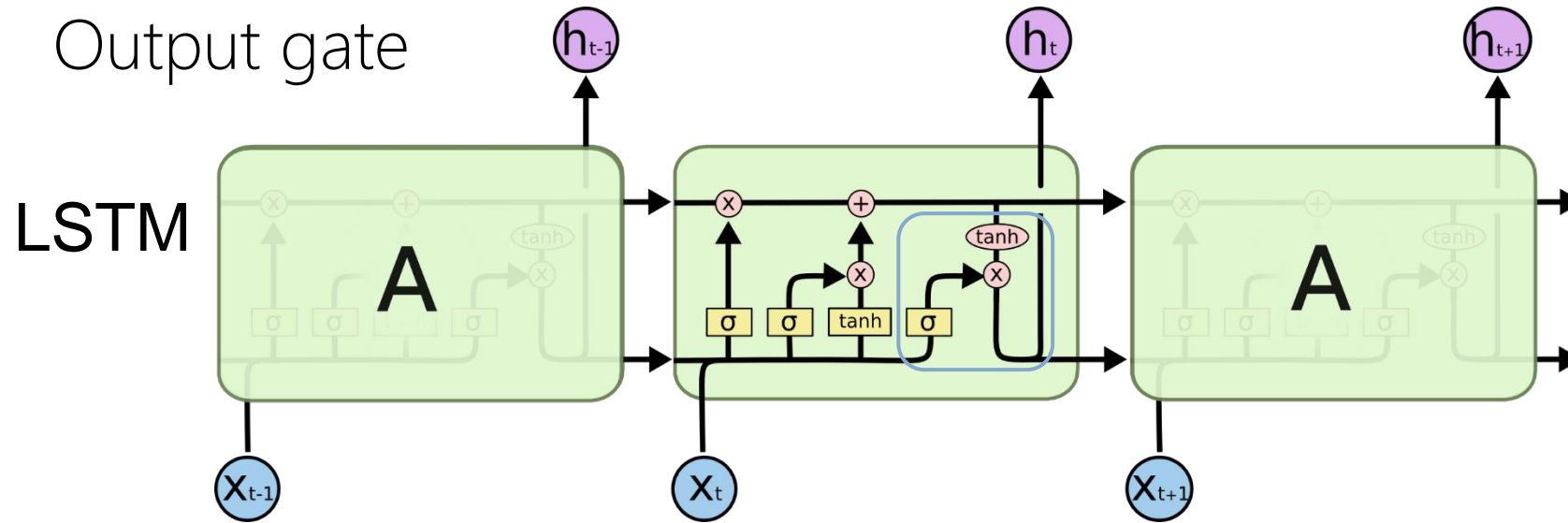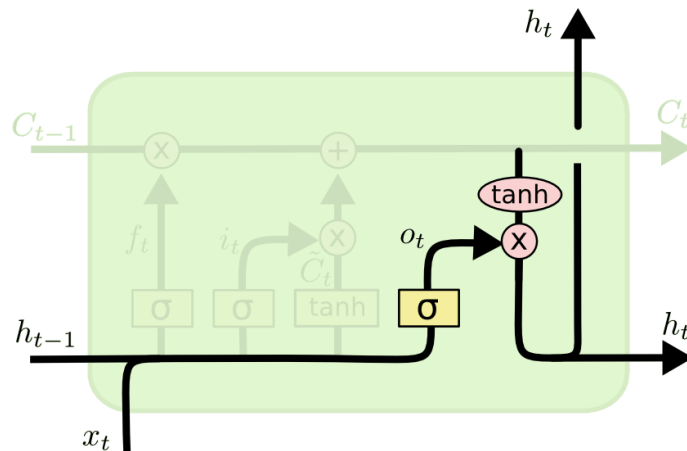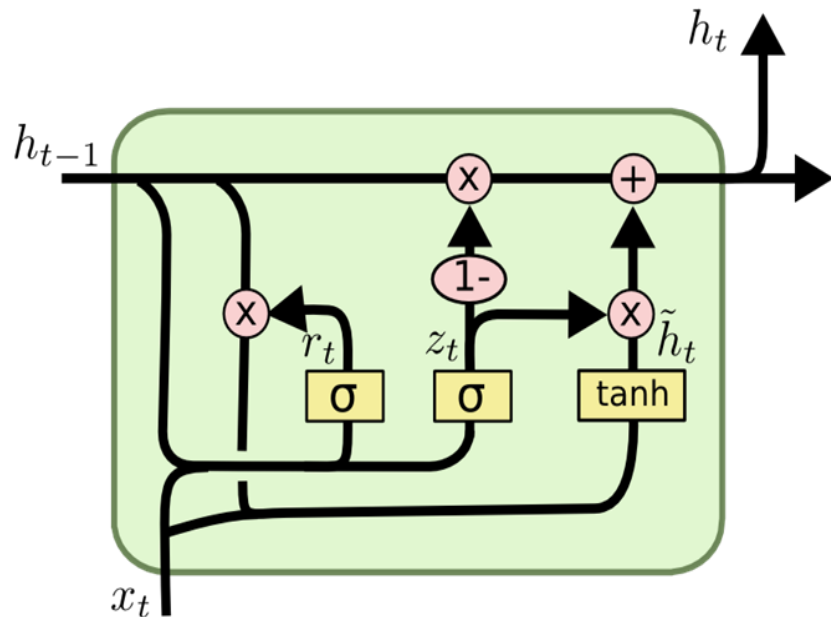$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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

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

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

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

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.

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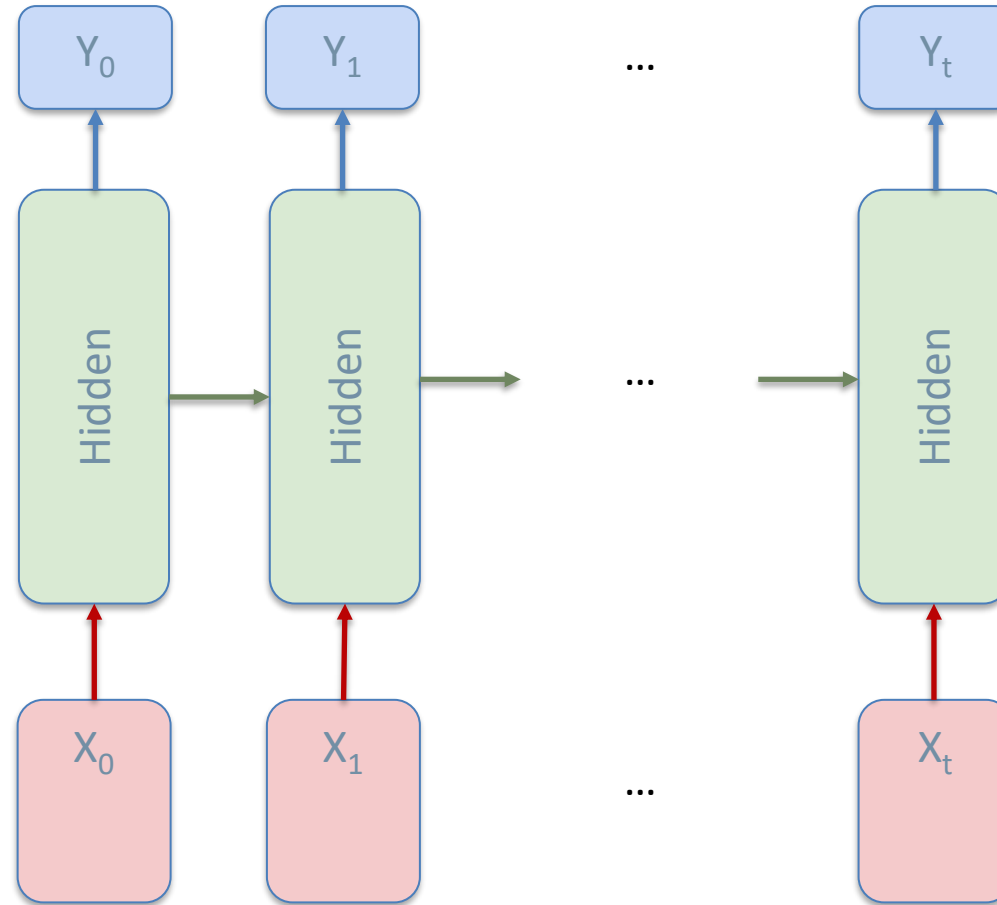
$$r_t = \sigma \left(W_r \cdot [h_{t-1}, x_t] \right)$$

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

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

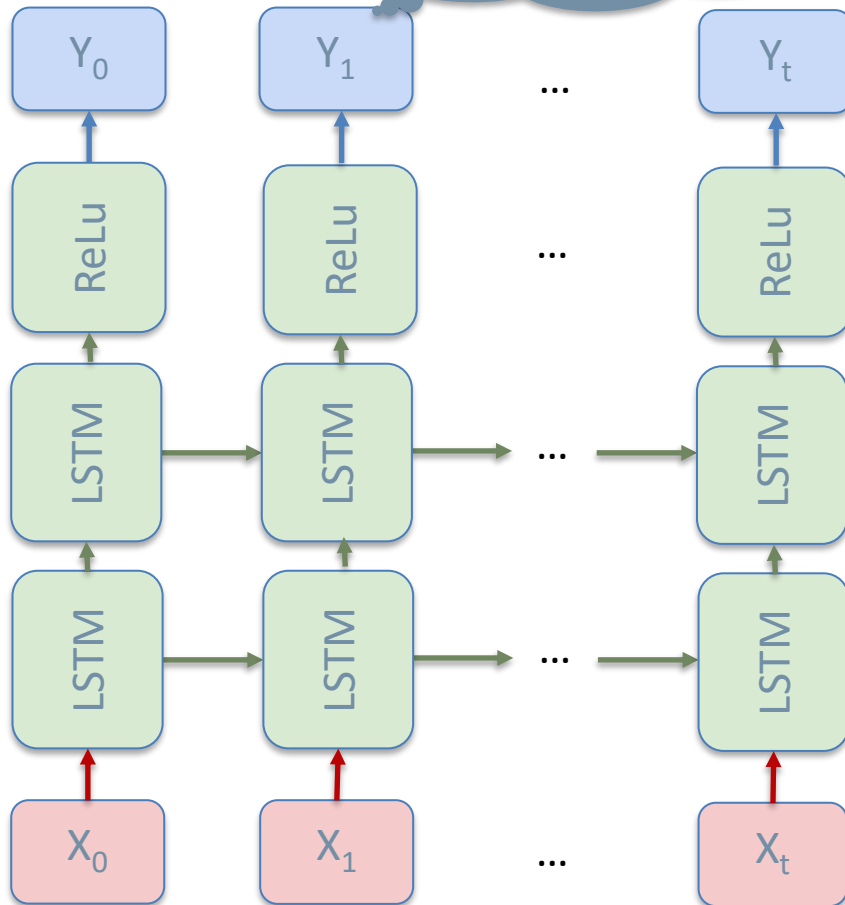LSTM Images from: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Networks

You can build a computation graph with continuous transformations.

Multiple Layers and Bidirectional LSTM Networks

A computatio... Hierarchical representation ... with continuous transformations.
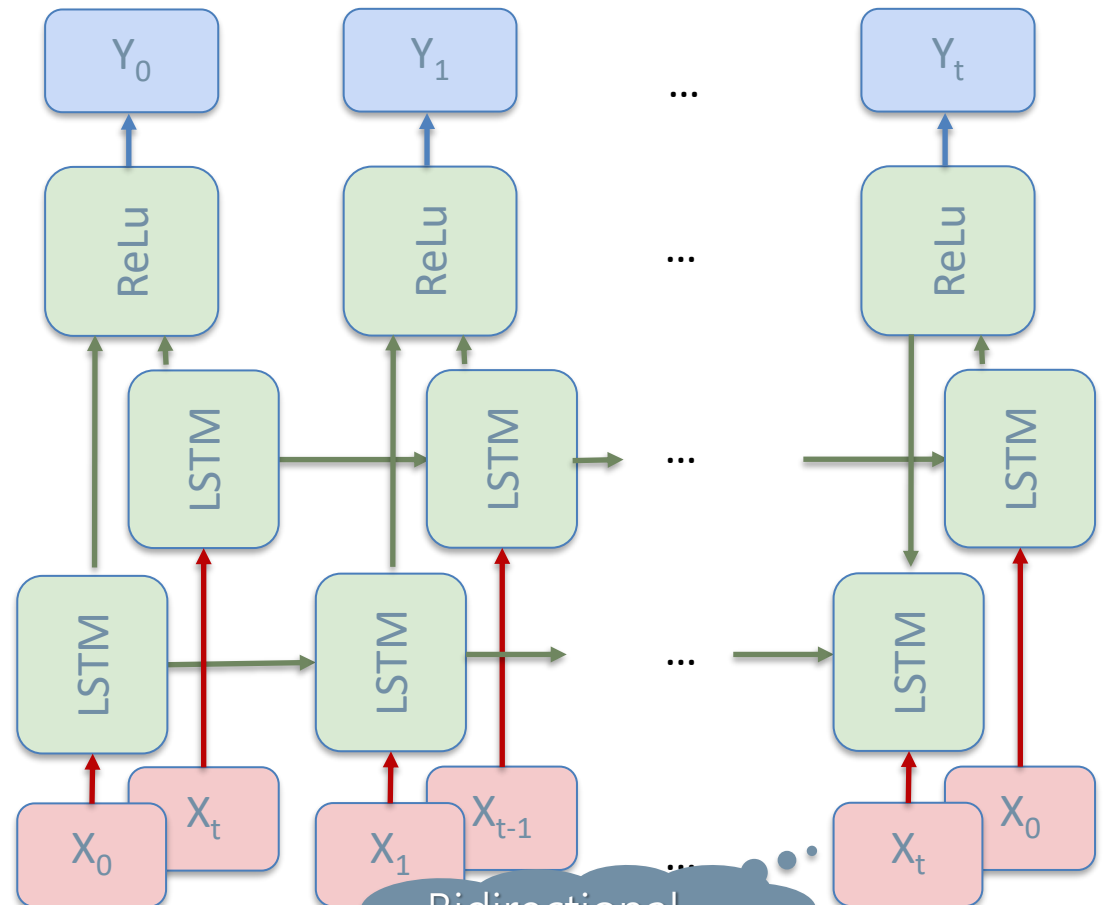
Tips & Tricks

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 with continuous transformations.

Hierarchical representation

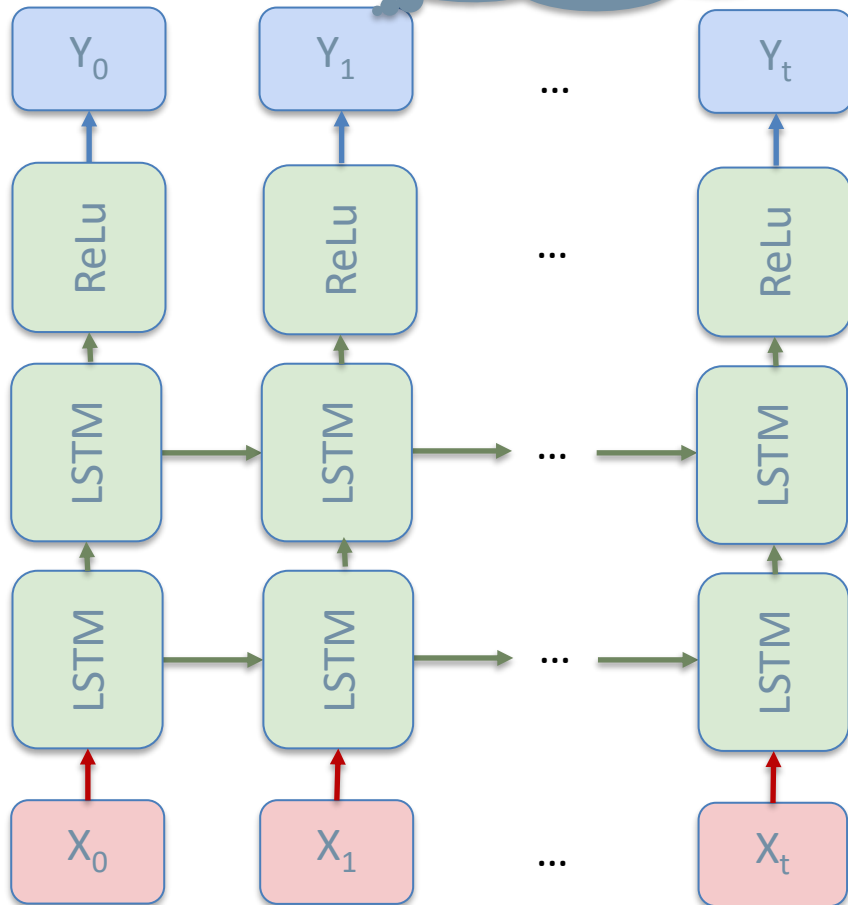Bidirectional processing

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
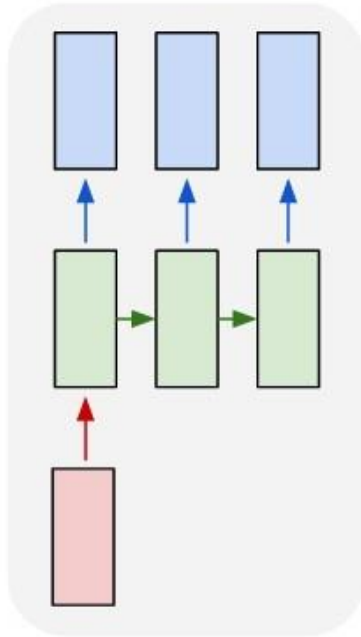 - Update these parameters by gradient descent
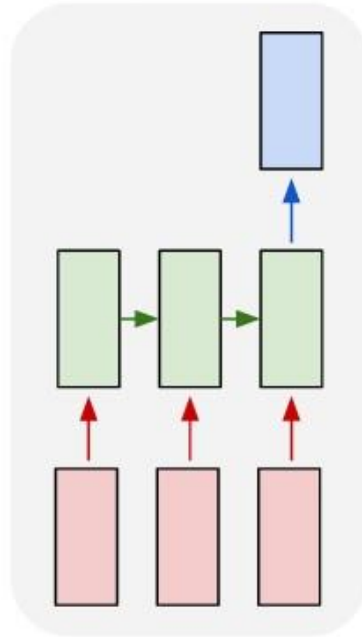
Sequential Data Problems

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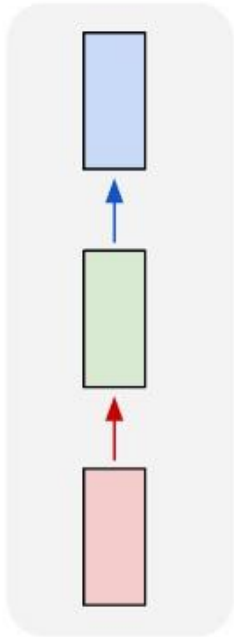
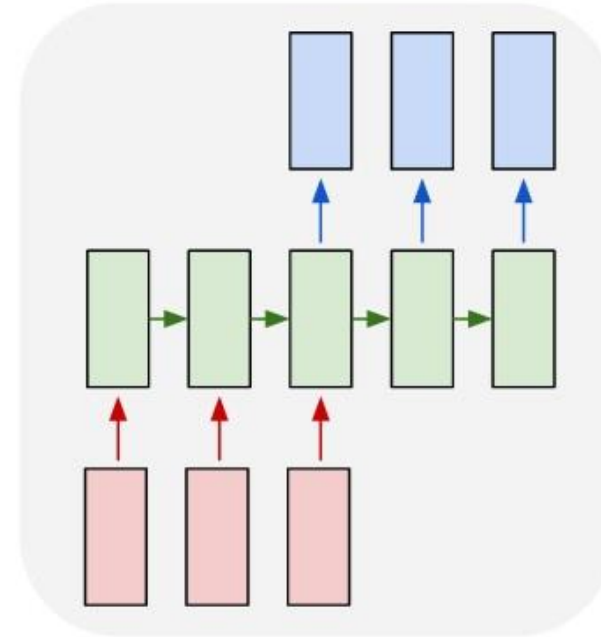**Fixed-sized input to fixed-sized output** (e.g. image classification)

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

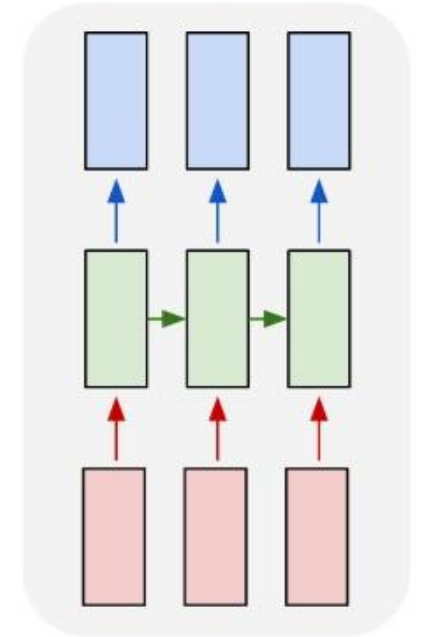**Sequence input** (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).

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

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

LSTM Images Credits: Andrej Karpathy

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.

one to many
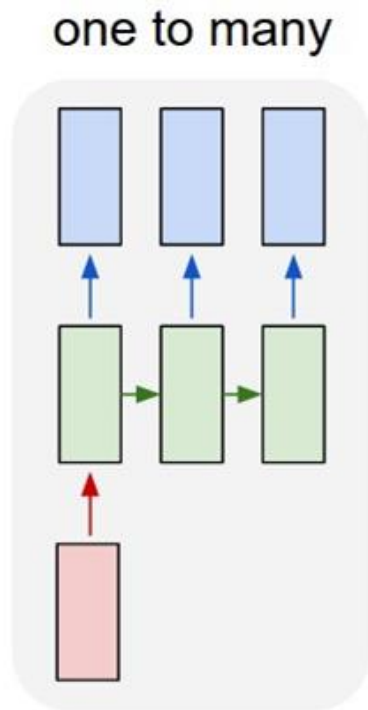
A person riding a motorcycle on a dirt road.

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 → -
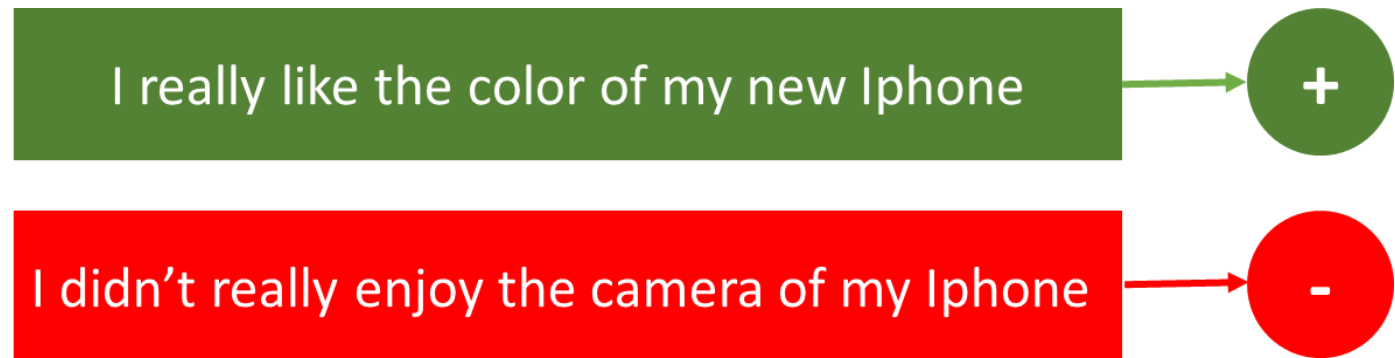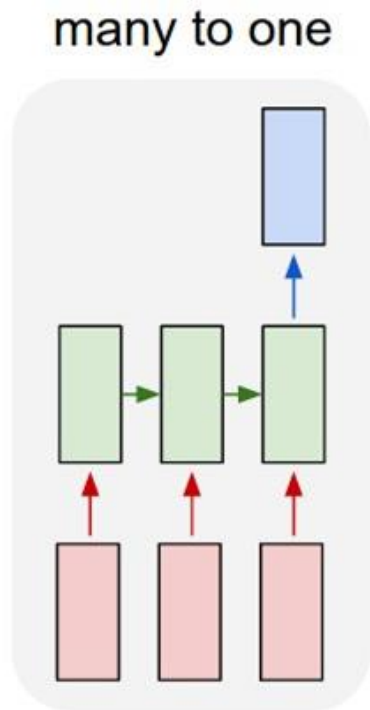
Sequence to Sequence Learning Examples (3/3)

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 it's own semantics and it has varying lengths for the same sentence.

many to many   many to many

French was the official language of the colony of French Indochina, comprising modern-day Vietnam, Laos, and Cambodia. It continues to be an administrative language in Laos and Cambodia, although its influence has waned in recent years.

Le français était la langue officielle de la colonie de l'Indochine française, comprenant le Vietnam d'aujourd'hui, le Laos et le Cambodge. Il continue d'être une langue administrative au Laos et au Cambodge, bien que son influence a décliné au cours des dernières années.