ROB 537
Learning-Based Control

Week **x**, Lecture **y**

Sequence Modeling

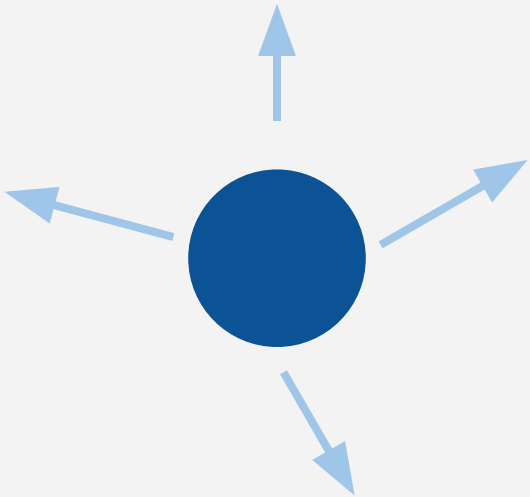
HW **m** due on **n/o 11:59 PM**

Robotics Seminars: 10AM Fridays

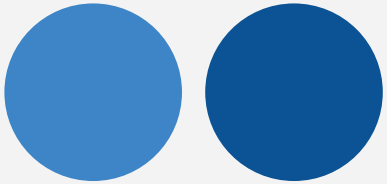
Oregon State University

Given this image of a ball, can you predict where it goes next?

Oregon State University



Given this image of a ball, can you predict where it goes next?



Given this image of a ball, can you predict where it goes next?

Oregon State University

Given this image of a ball, can you predict where it goes next?

Given this image of a ball, can you predict where it goes next?

Given this image of a ball, can you predict where it goes next?

Audio

Audio

Text is also a sequence

Text

character

```
T  e  x  t
```

```
Text is also a sequence
```

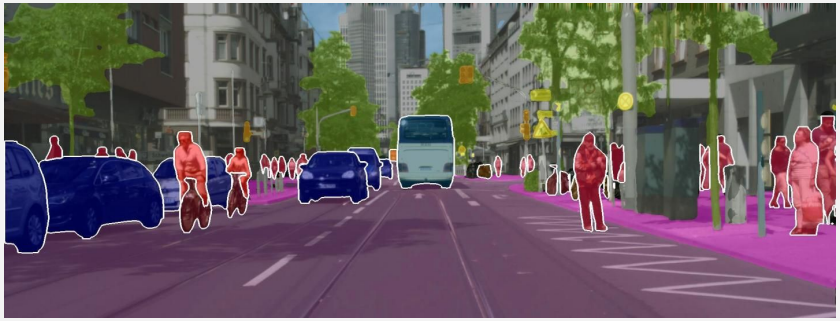
Text

character `T e x t`

word `Text is also a sequence`

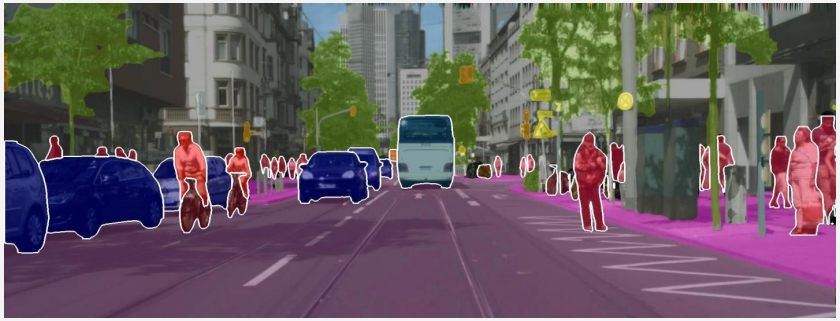
Text

Sequences in the Wild

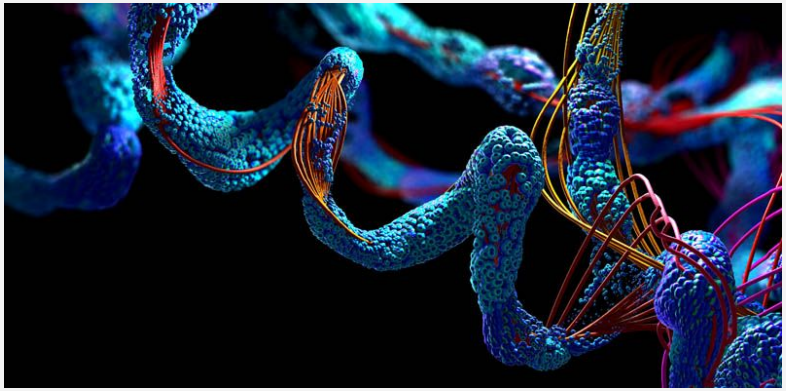


eurekalert

Sequences in the Wild



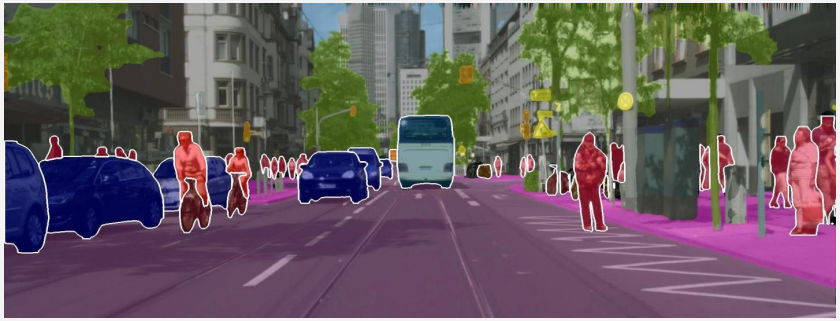
eurekalert



AWS - BERT

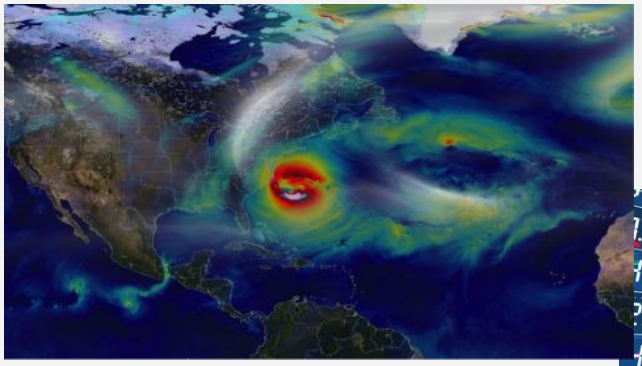
Sequences in the Wild

Oregon State University



eurekalert

AWS - BERT



NASA

Adobe

Adobe

Sequences in the Wild

Oregon State University

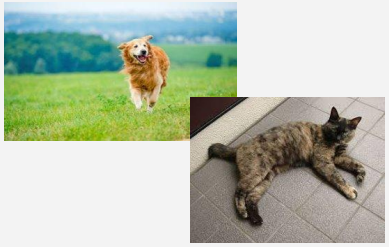
Sequence Modeling Applications

One to One

Sequence Modeling Applications

One to One

Image Classification

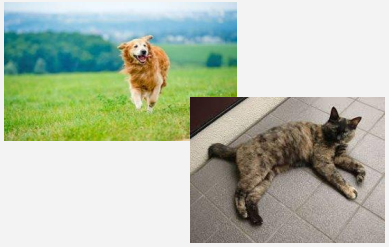


Is the input image a Dog or a Cat?

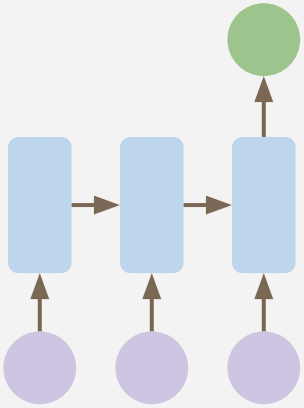
Sequence Modeling Applications

One to One

Image Classification



Is the input image a Dog or a Cat?

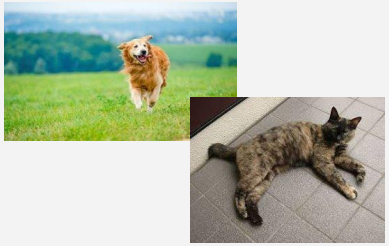


Many to One

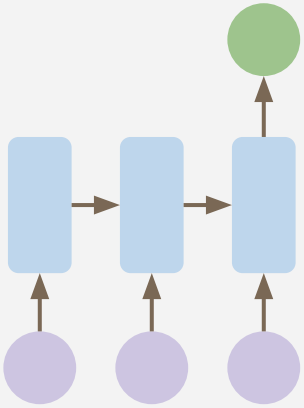
Sequence Modeling Applications

One to One

Image Classification



Is the input image a Dog or a Cat?



Many to One

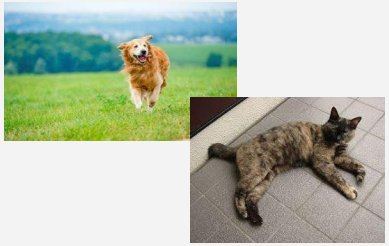
Sentiment Analysis

Oregon State University

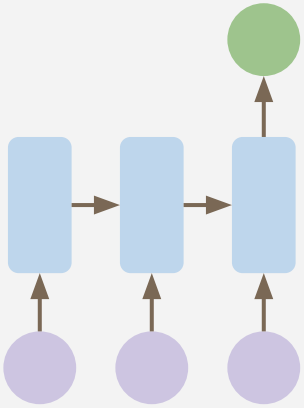
Sequence Modeling Applications

One to One

Image Classification

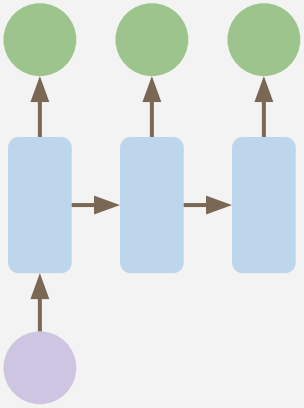


Is the input image a Dog or a Cat?



Many to One

Sentiment Analysis

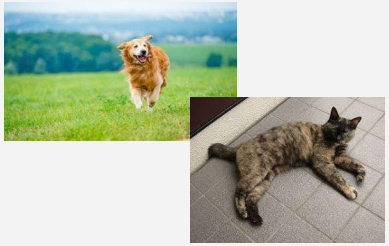


One to Many

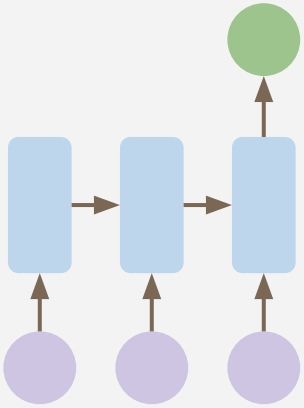
Sequence Modeling Applications

One to One

Image Classification

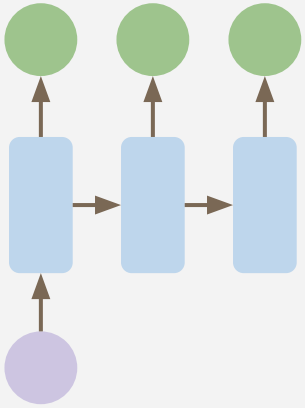


Is the input image a Dog or a Cat?



Many to One

Sentiment Analysis



One to Many

Image Captioning

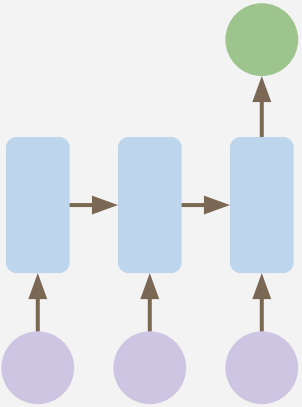
A duck sitting on top of a heap of snow

Sequence Modeling Applications

One to One

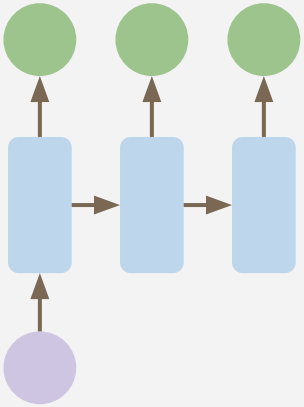
Image Classification

Is the input image a Dog or a Cat?



Many to One

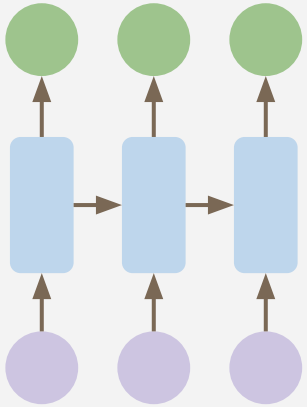
Sentiment Analysis



One to Many

Image Captioning

A duck sitting on top of a heap of snow

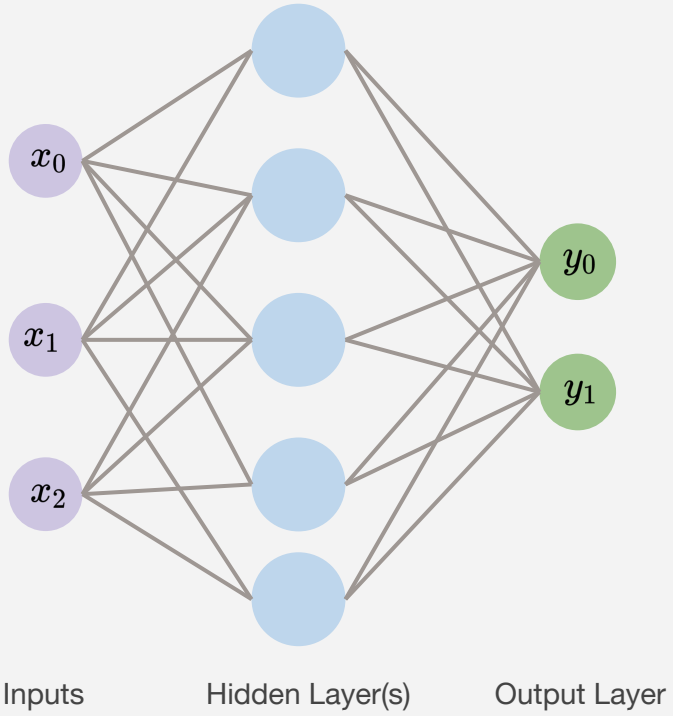


Many to Many

Translation

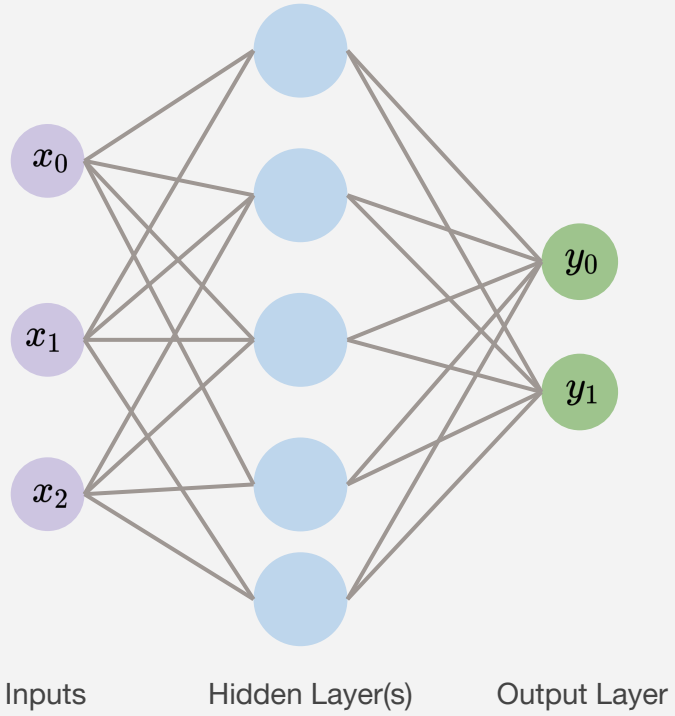
Neural Networks

- Input-output mapping
- Universal function approximators
- Key Assumptions
  - Static I/O mapping
  - Input spans all necessary information



Neural Networks: Sequential Modeling

- Can we feed data sequentially?
  - Immediate state is not Markov!

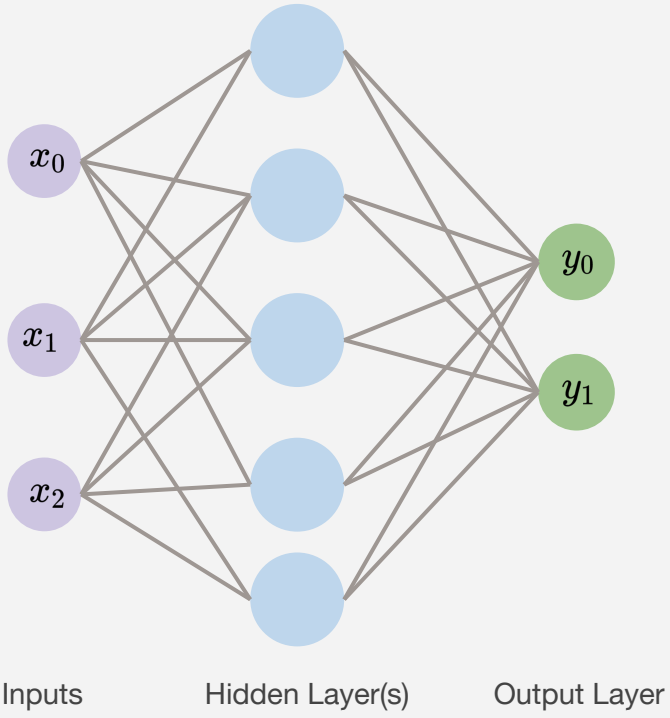


Simple feed forward neural networks cannot process sequences effectively

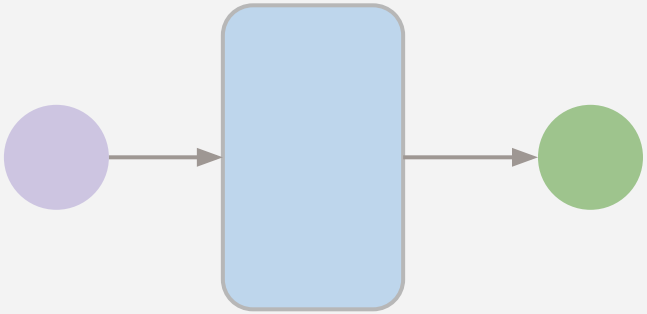
Neural Networks: Sequential Modeling

Neurons with Recurrence

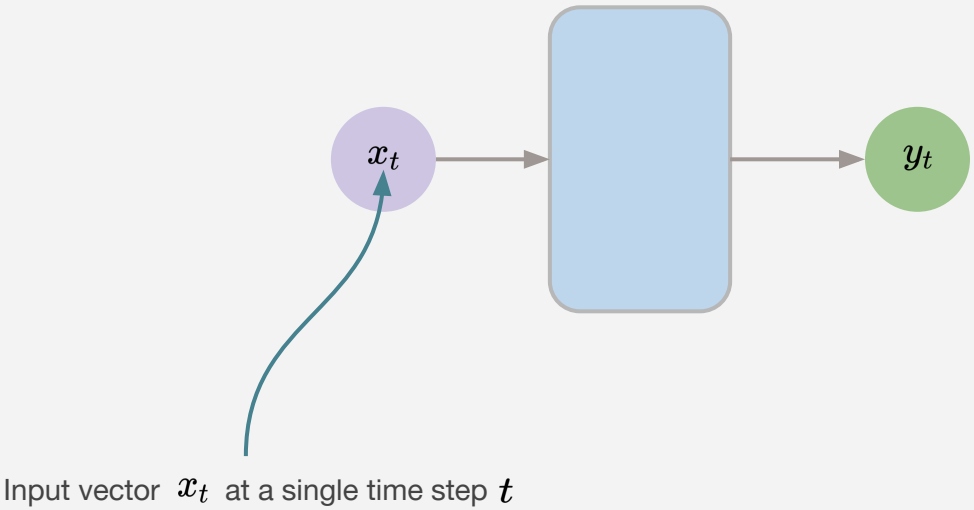
Neural Networks: Abstraction



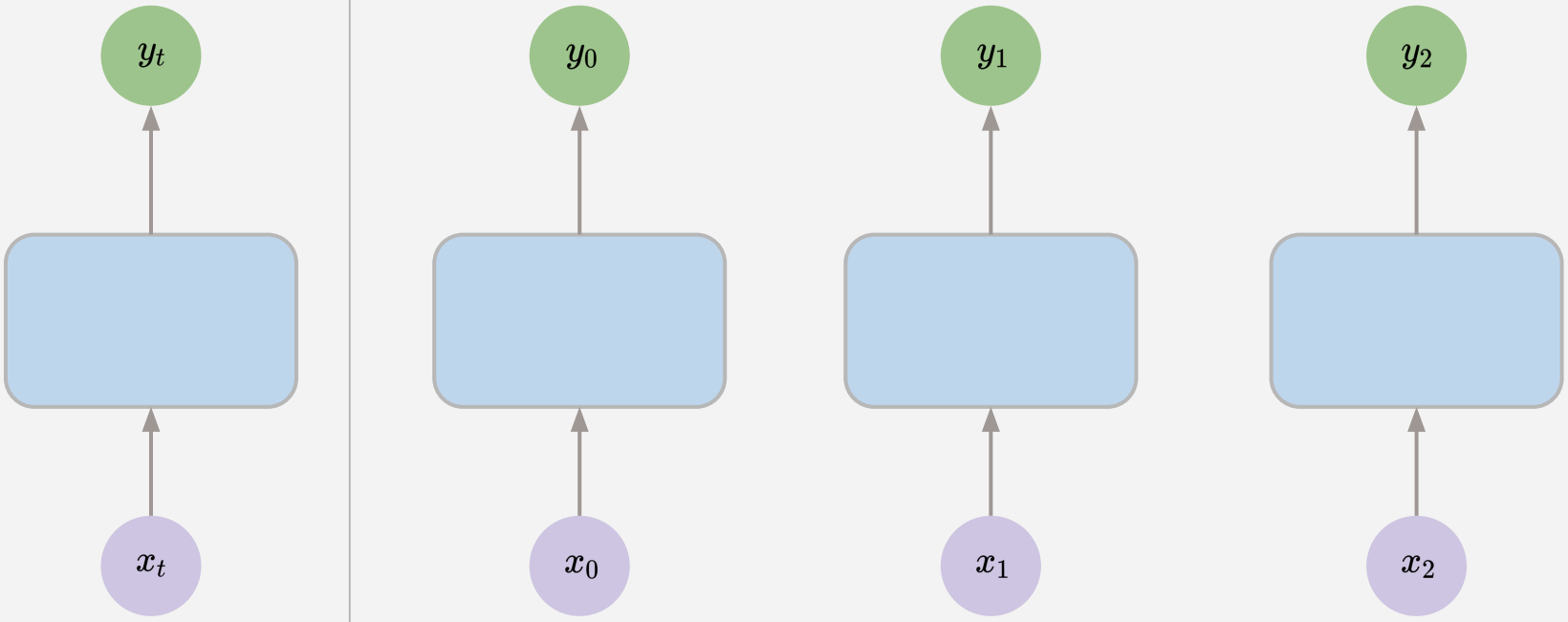
Neural Networks: Abstraction



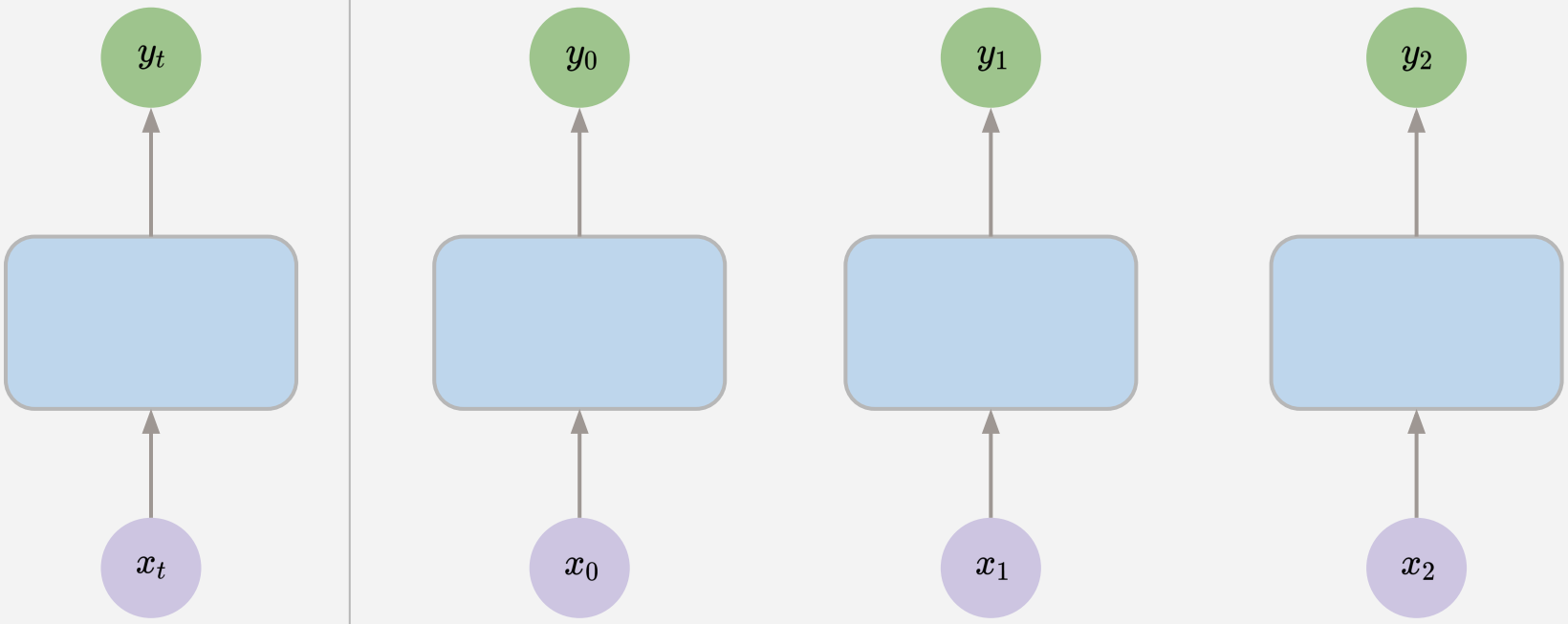
Neural Networks: Handling Time Steps



Handling Individual Time Steps



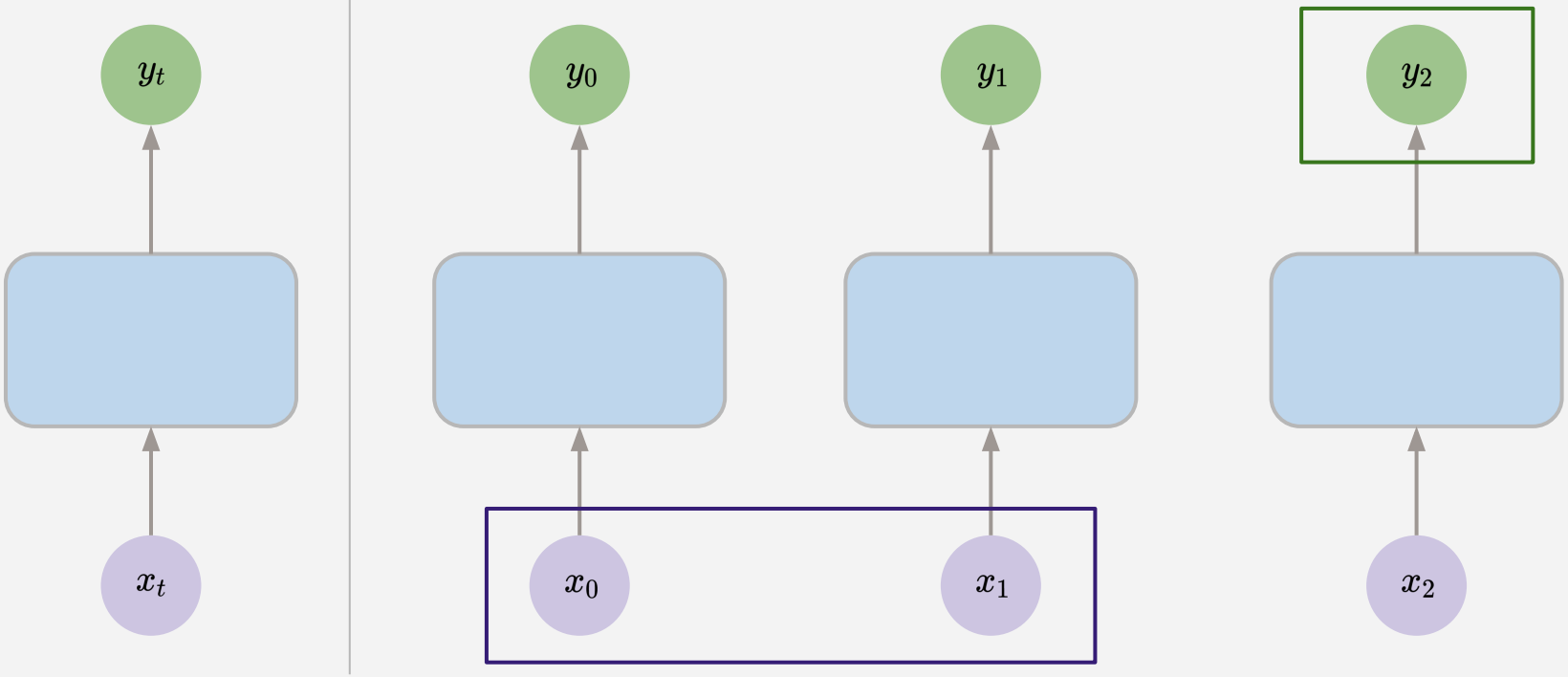
Handling Individual Time Steps



In feedforward networks, the output is the weighted input passed through an activation function

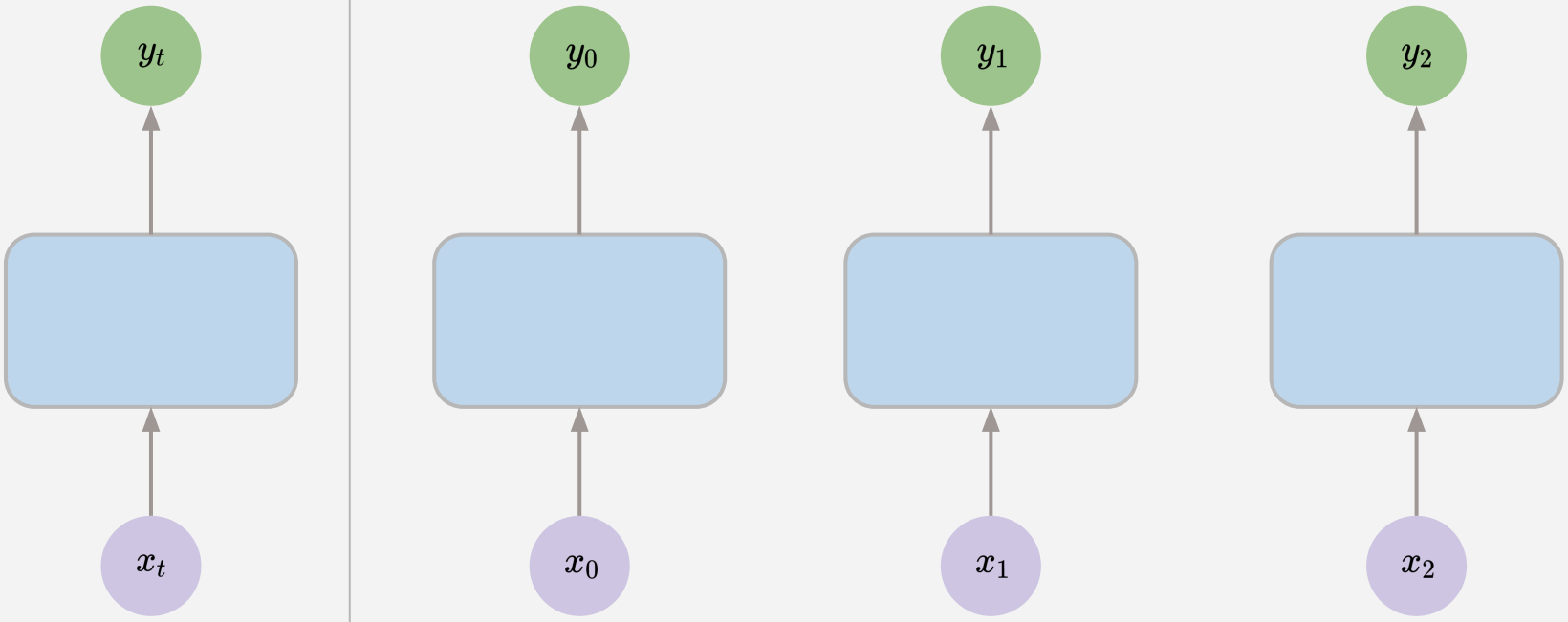
$$y_t = f(x_t)$$

Handling Individual Time Steps

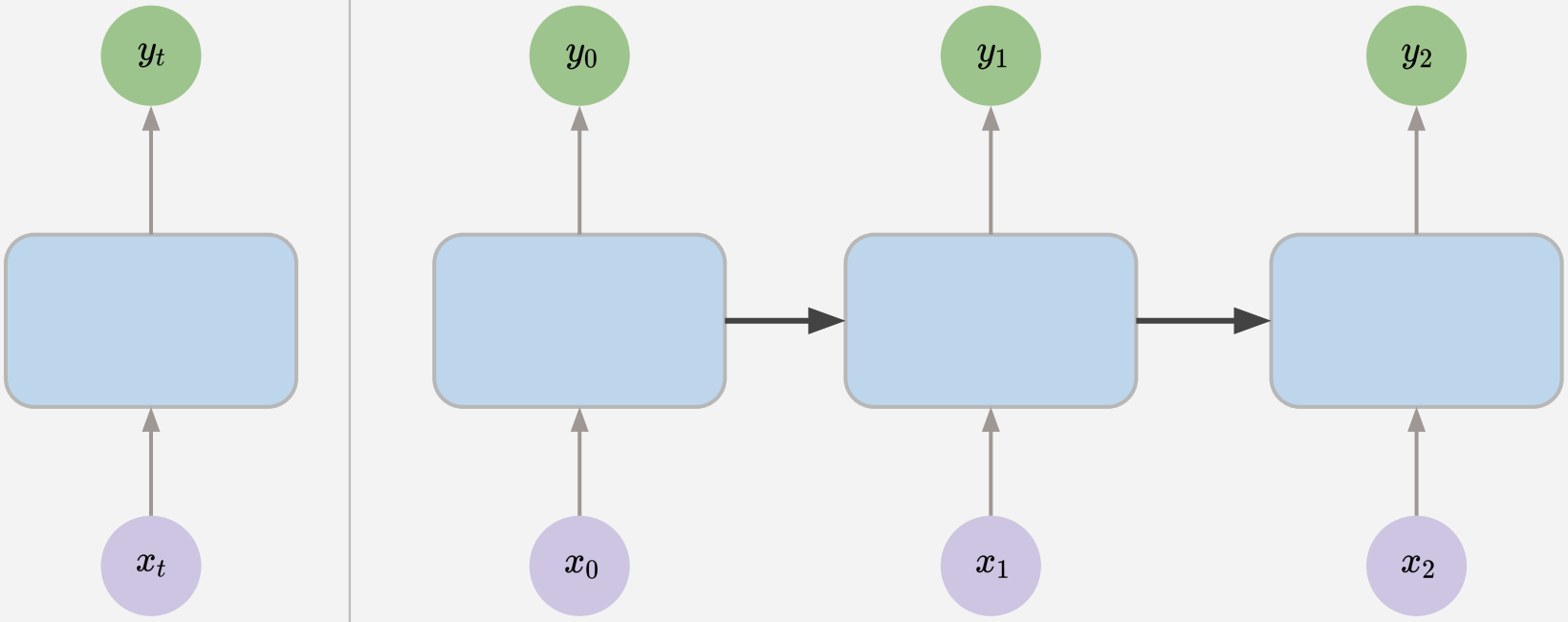


We have to now account for time steps and recurrence

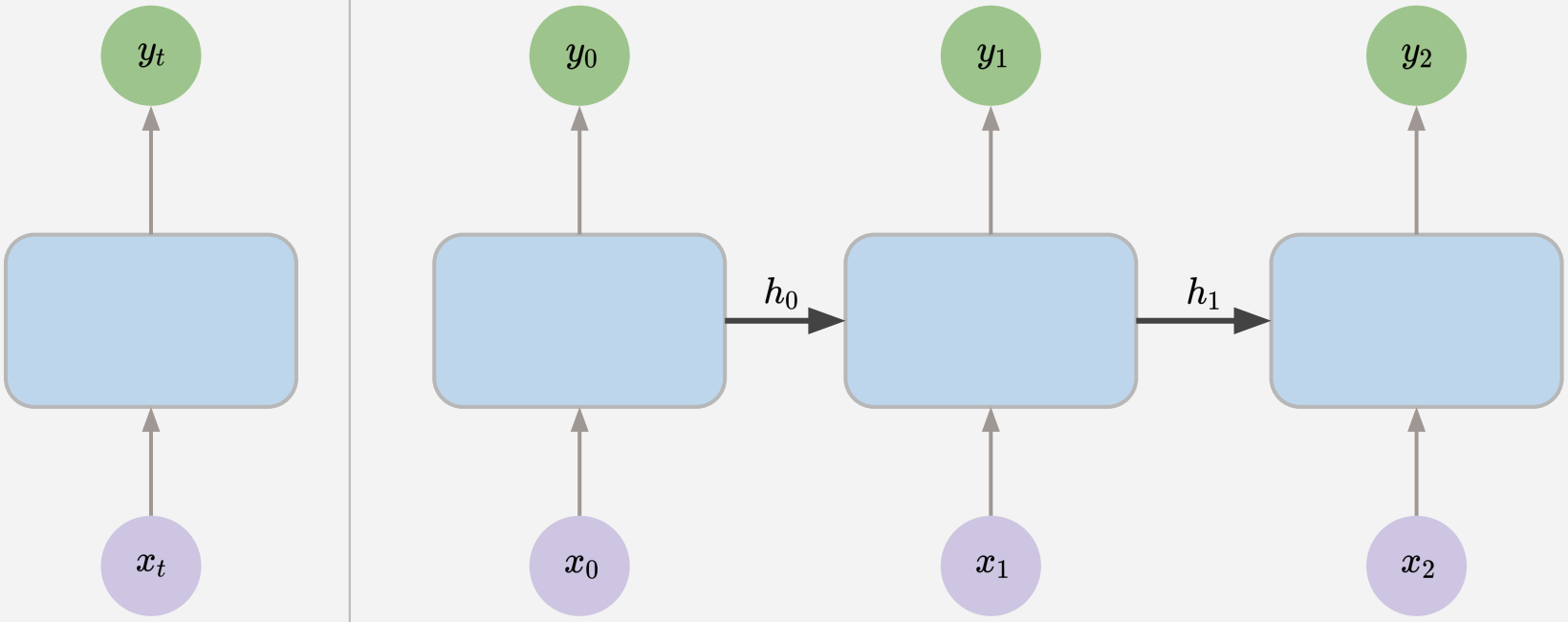
Recurrence



Recurrence

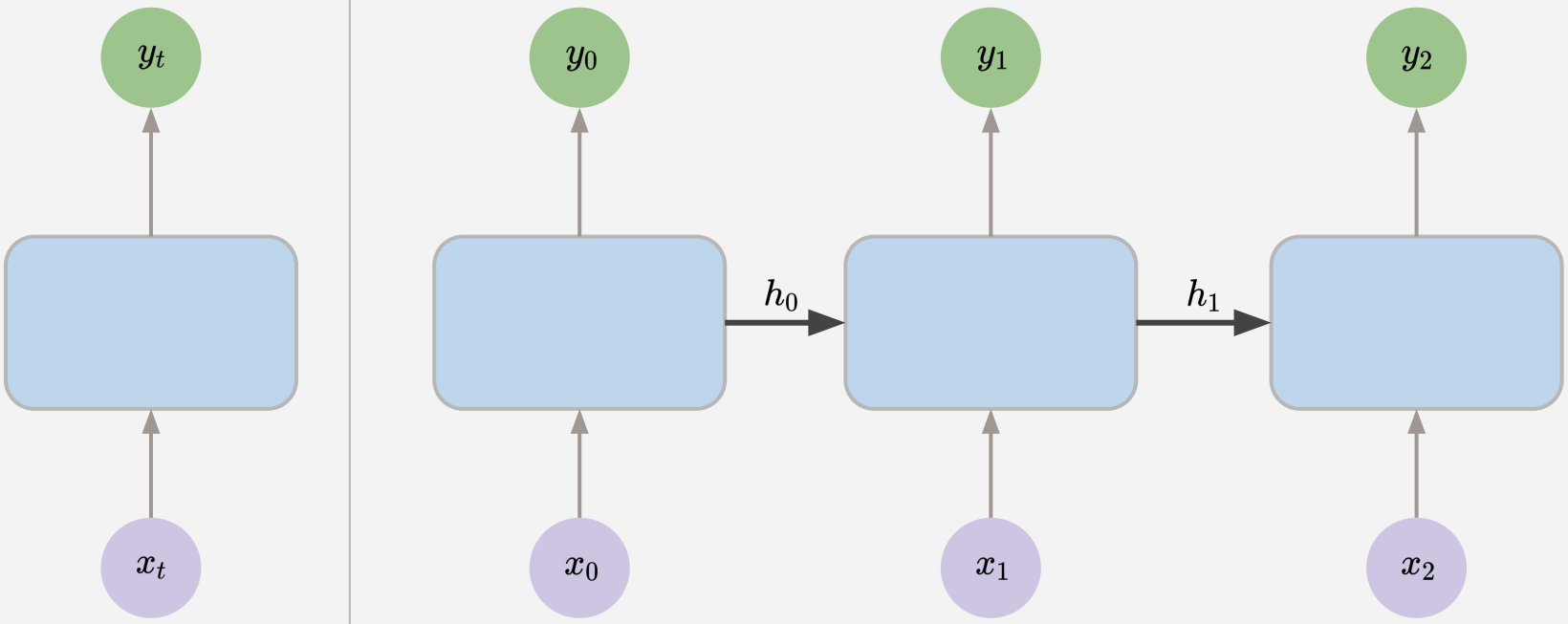


Recurrence



$$y_t = f(x_t, h_{t-1})$$

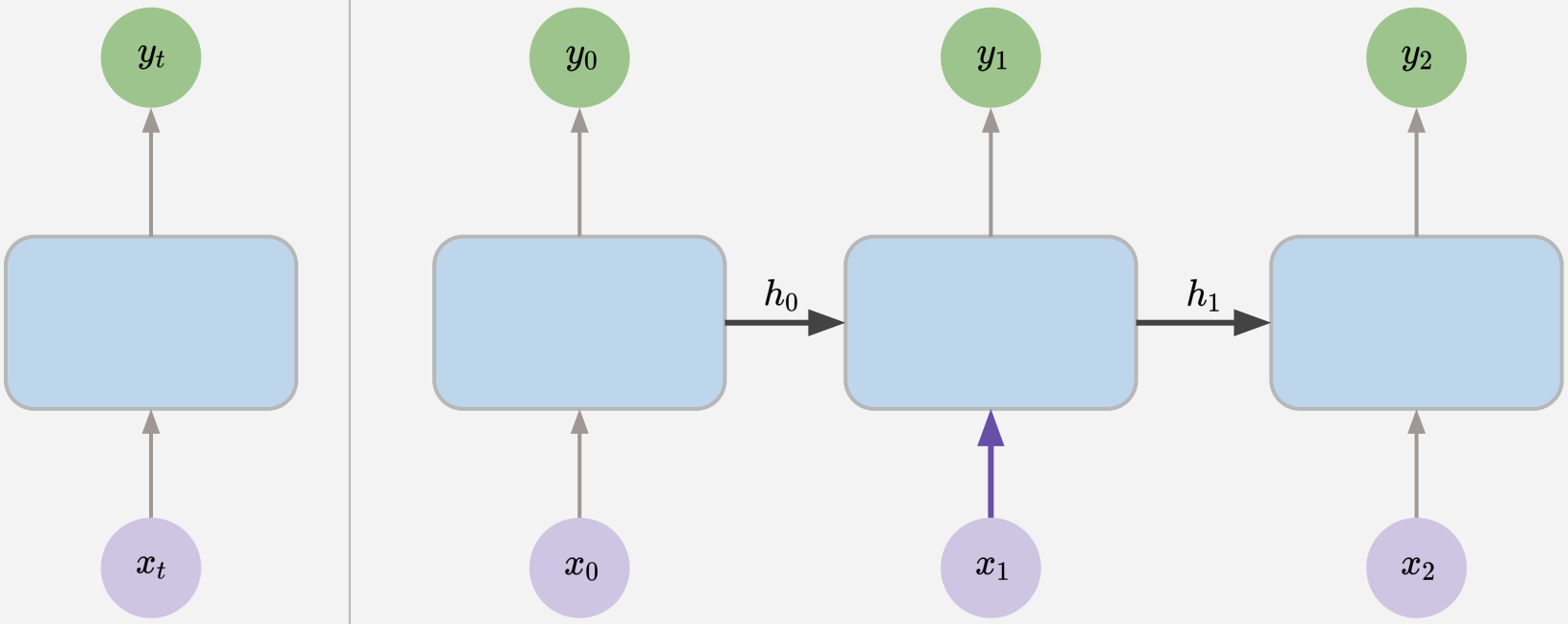
Recurrence



$$y_t = f(x_t, h_{t-1})$$

output

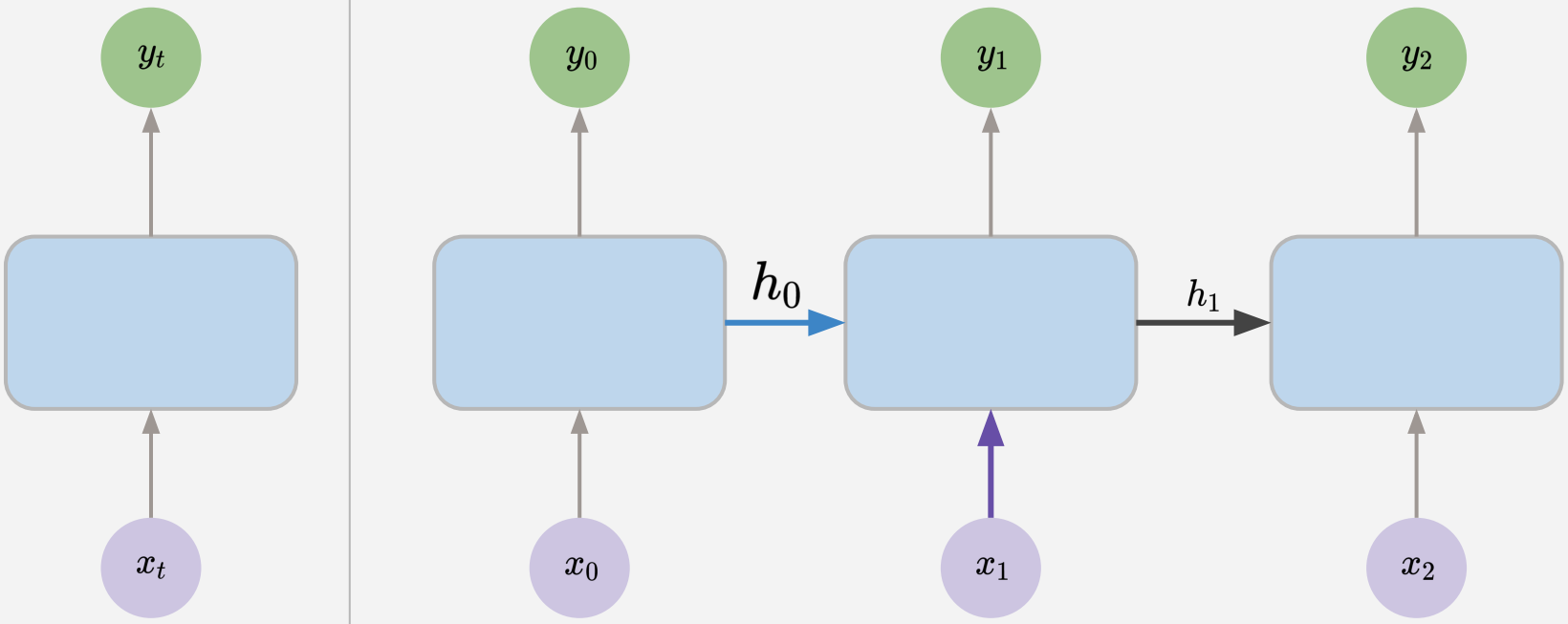
Recurrence



$$y_t = f(x_t, h_{t-1})$$

output input

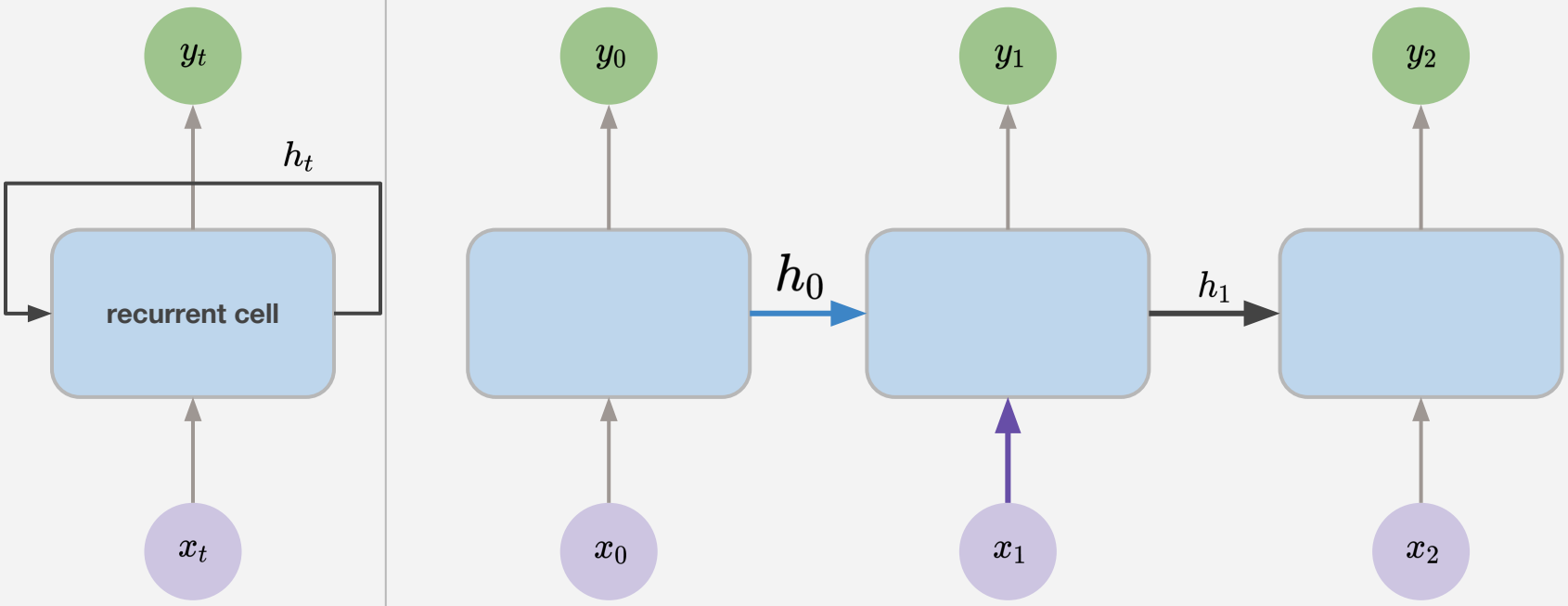
Recurrence



$$y_t = f(x_t, h_{t-1})$$

output input history

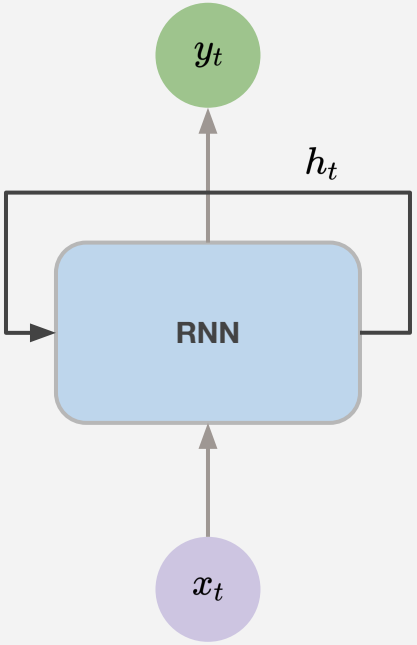
Recurrence



$$y_t = f(x_t, h_{t-1})$$

output input history

Recurrent Neural Networks (RNNs)



Output Vector

$$y_t = W_{hy}^T h_t$$

Hidden State Update

$$h_t = \sigma\left(W_{xh}^T x_t + W_{hh}^T h_{t-1}\right)$$

Input Vector

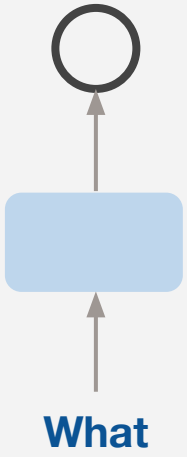
$$x_t$$

Recurrent cells have a state h_t, that is updated at each time step as a sequence is processed

RNNs: Feeding Sequential Data Example

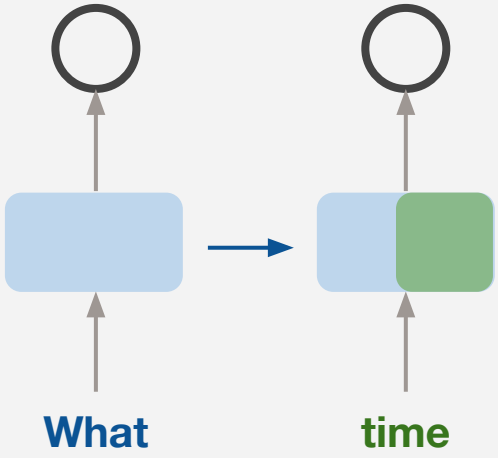
What time is it ?

RNNs: Feeding Sequential Data Example



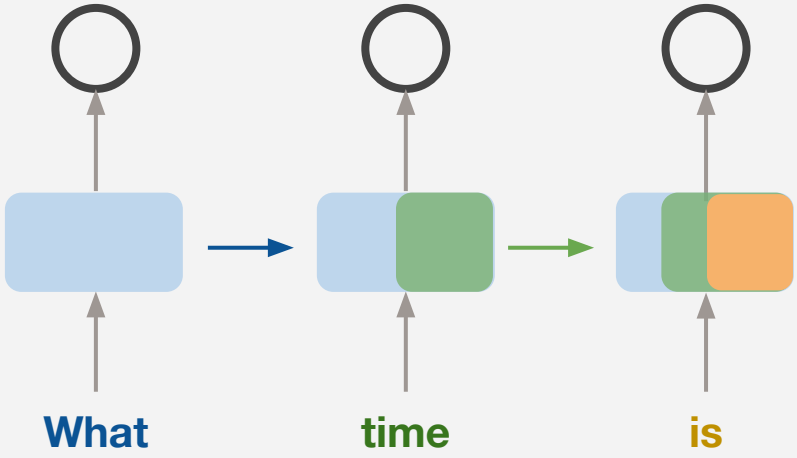
What time is it ?

RNNs: Feeding Sequential Data Example



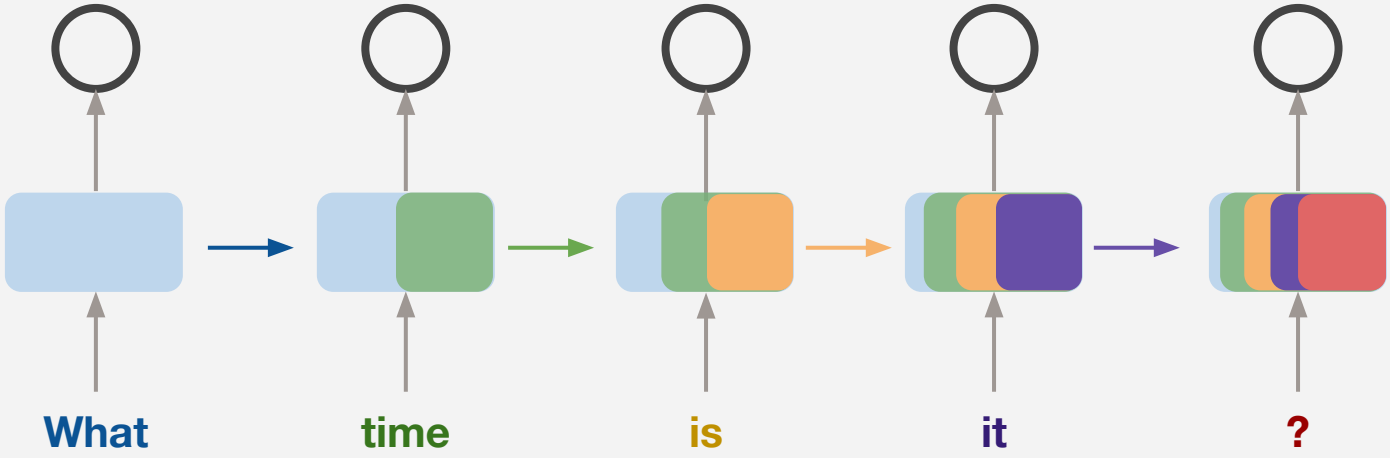
What time is it ?

RNNs: Feeding Sequential Data Example



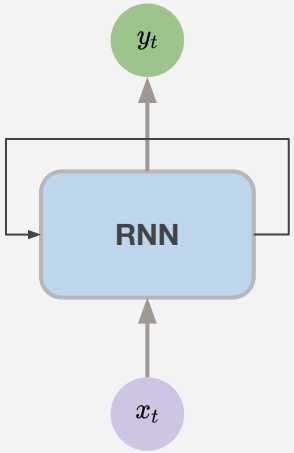
What time is it ?

RNNs: Feeding Sequential Data Example



What time is it ?

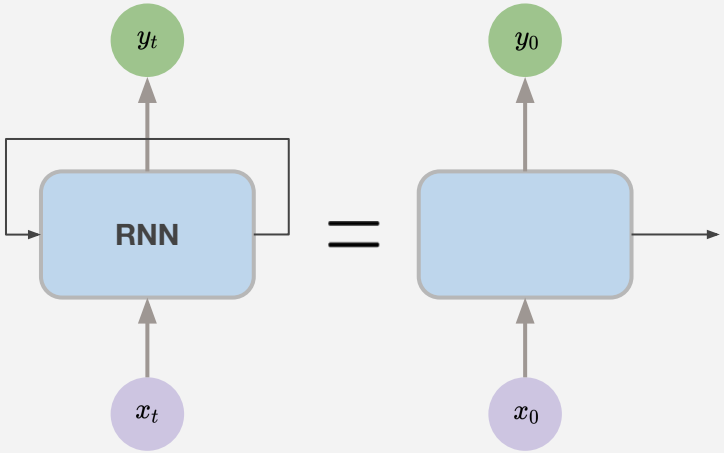
RNNs: Computation Across Time



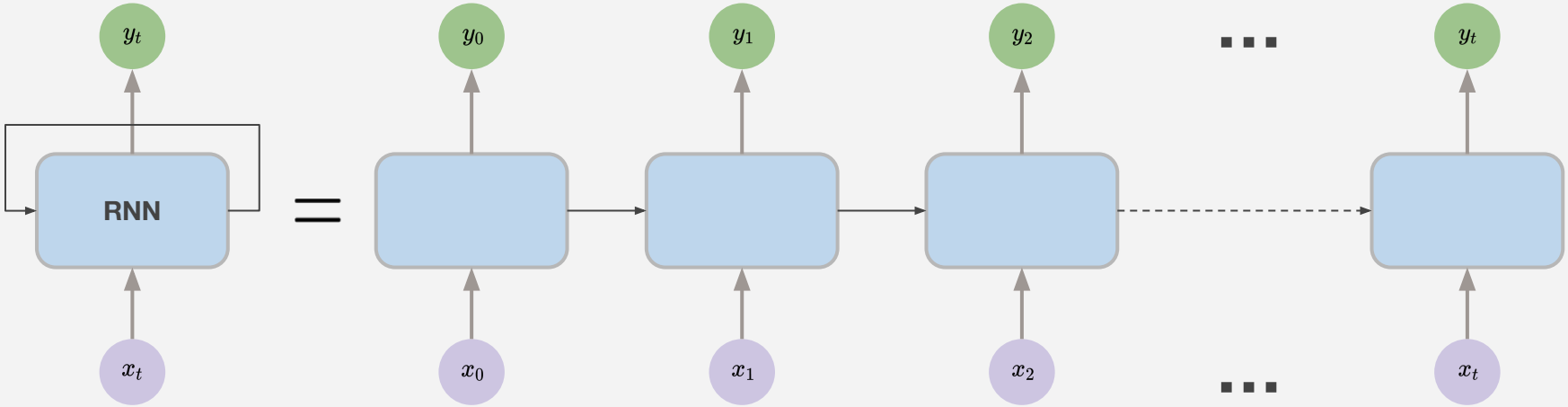
$=$

To process a sequence, unroll across time

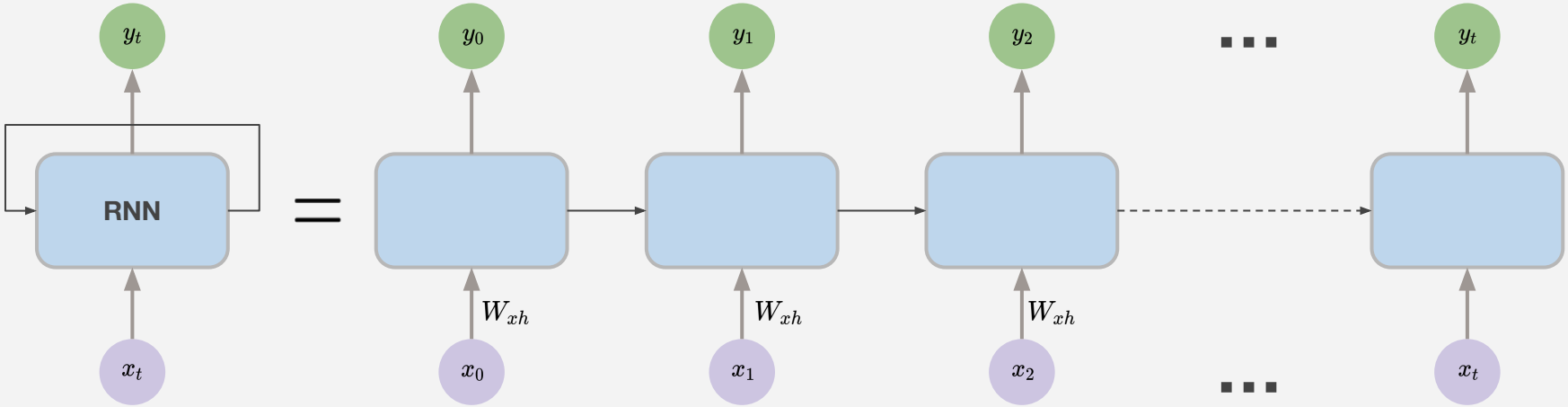
RNNs: Computation Across Time



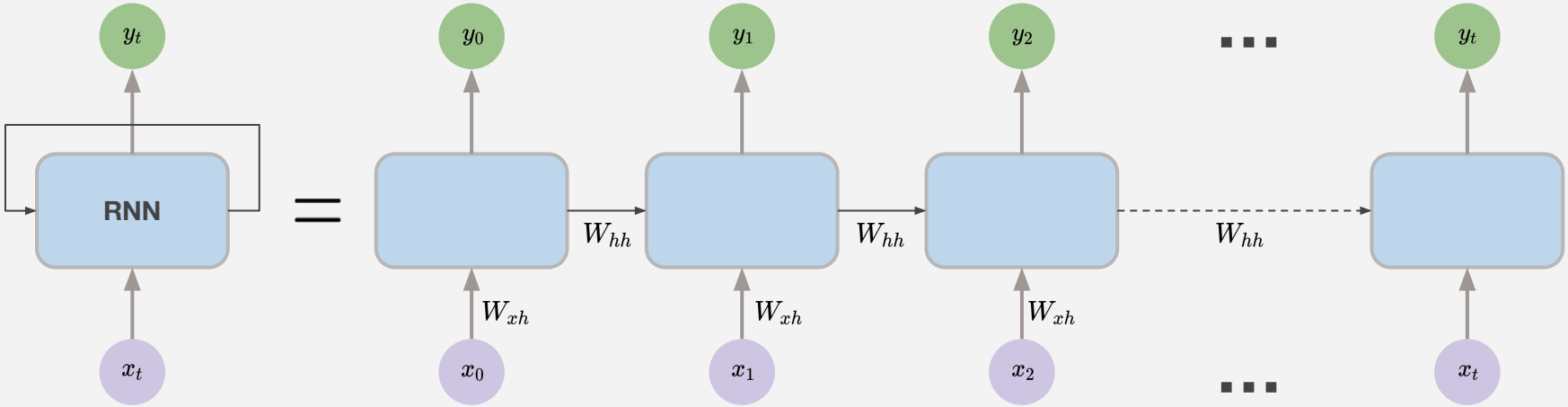
RNNs: Computation Across Time



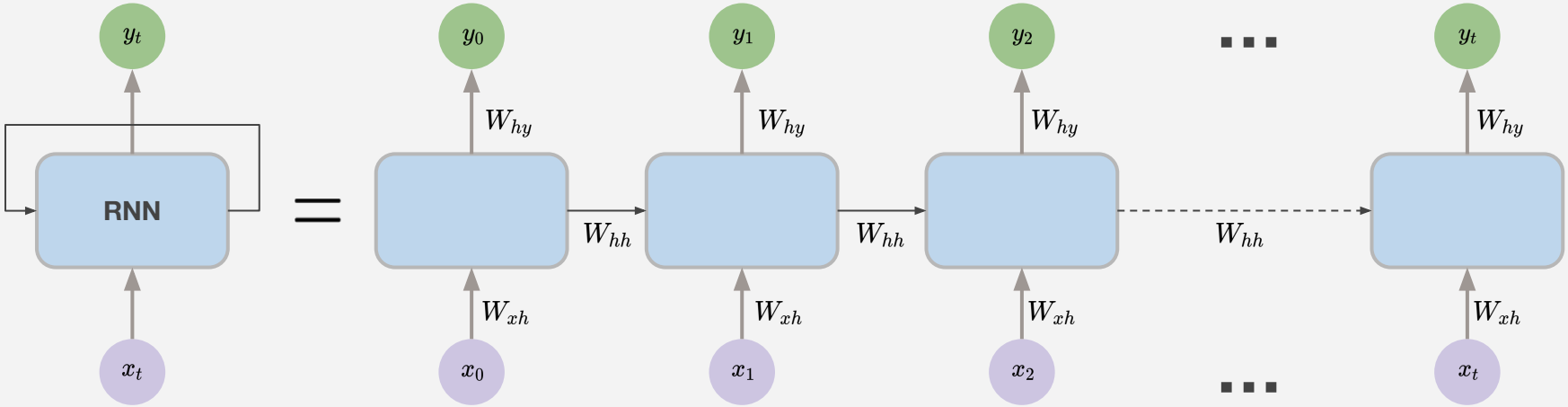
RNNs: Computation Across Time



RNNs: Computation Across Time

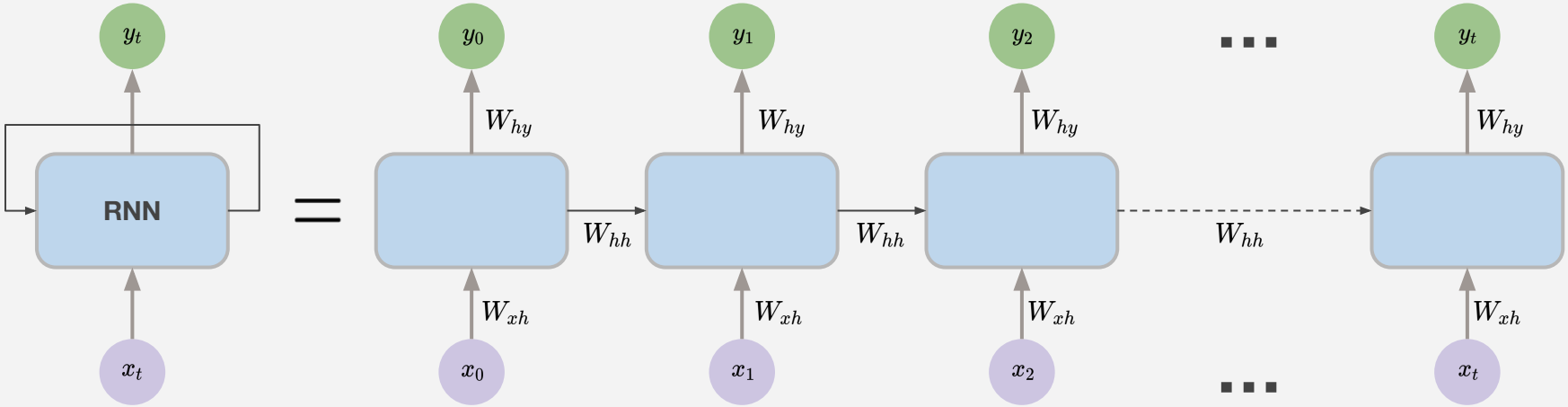


RNNs: Computation Across Time

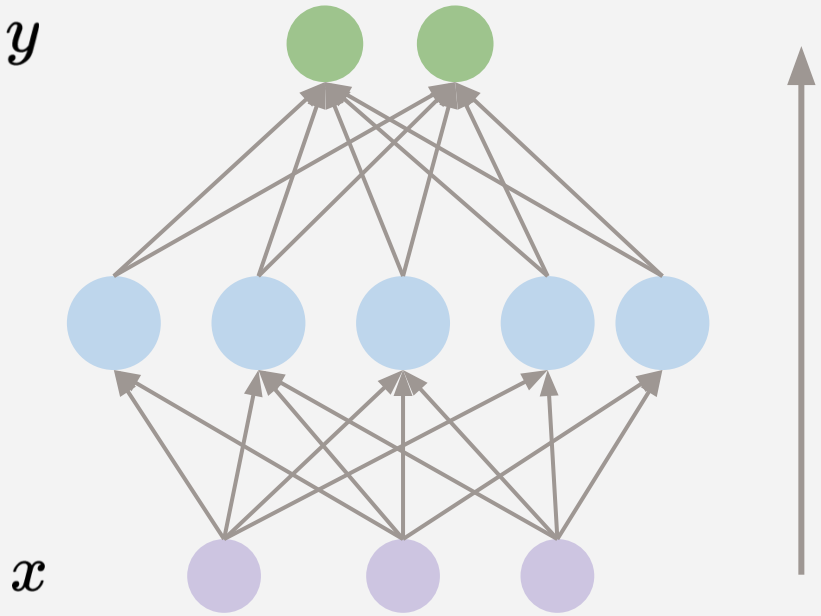


RNNs: Computation Across Time

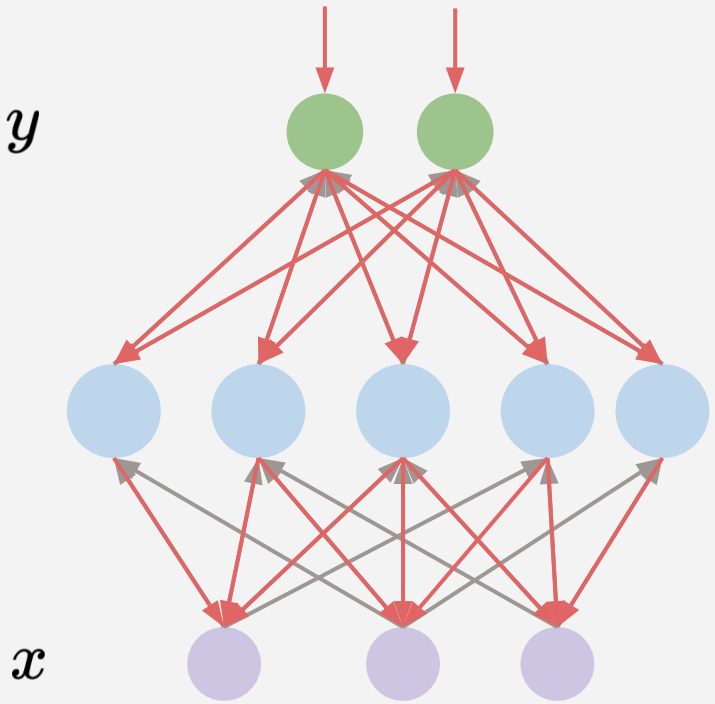
When a sequence is processed, the **same weights** are used at **every time step**



Recap: Forward Pass in Feedforward NNs

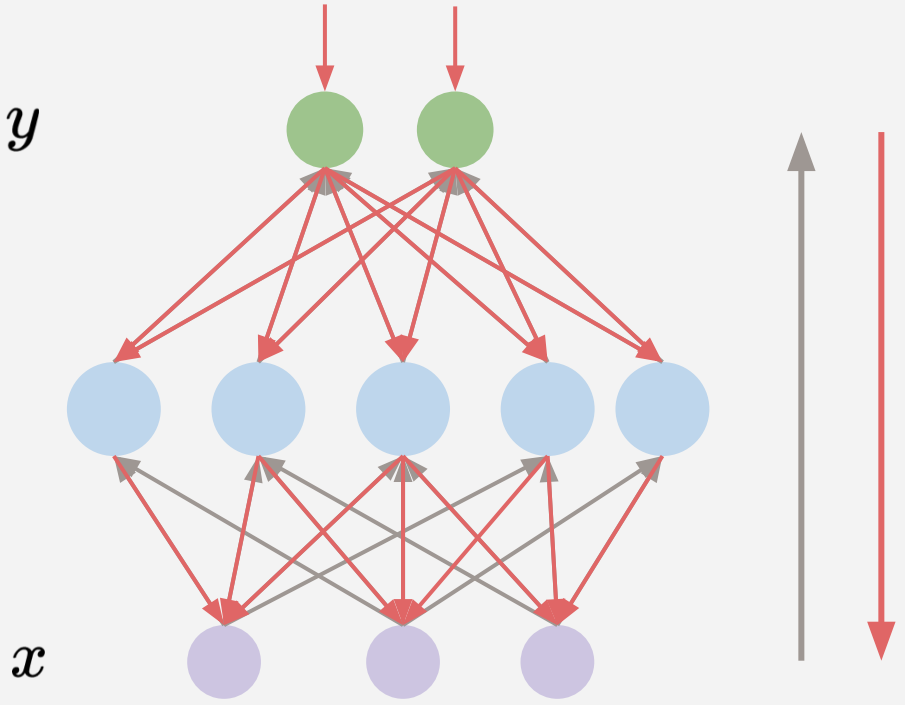


Recap: Backpropagation in Feedforward NNs



Backpropagation

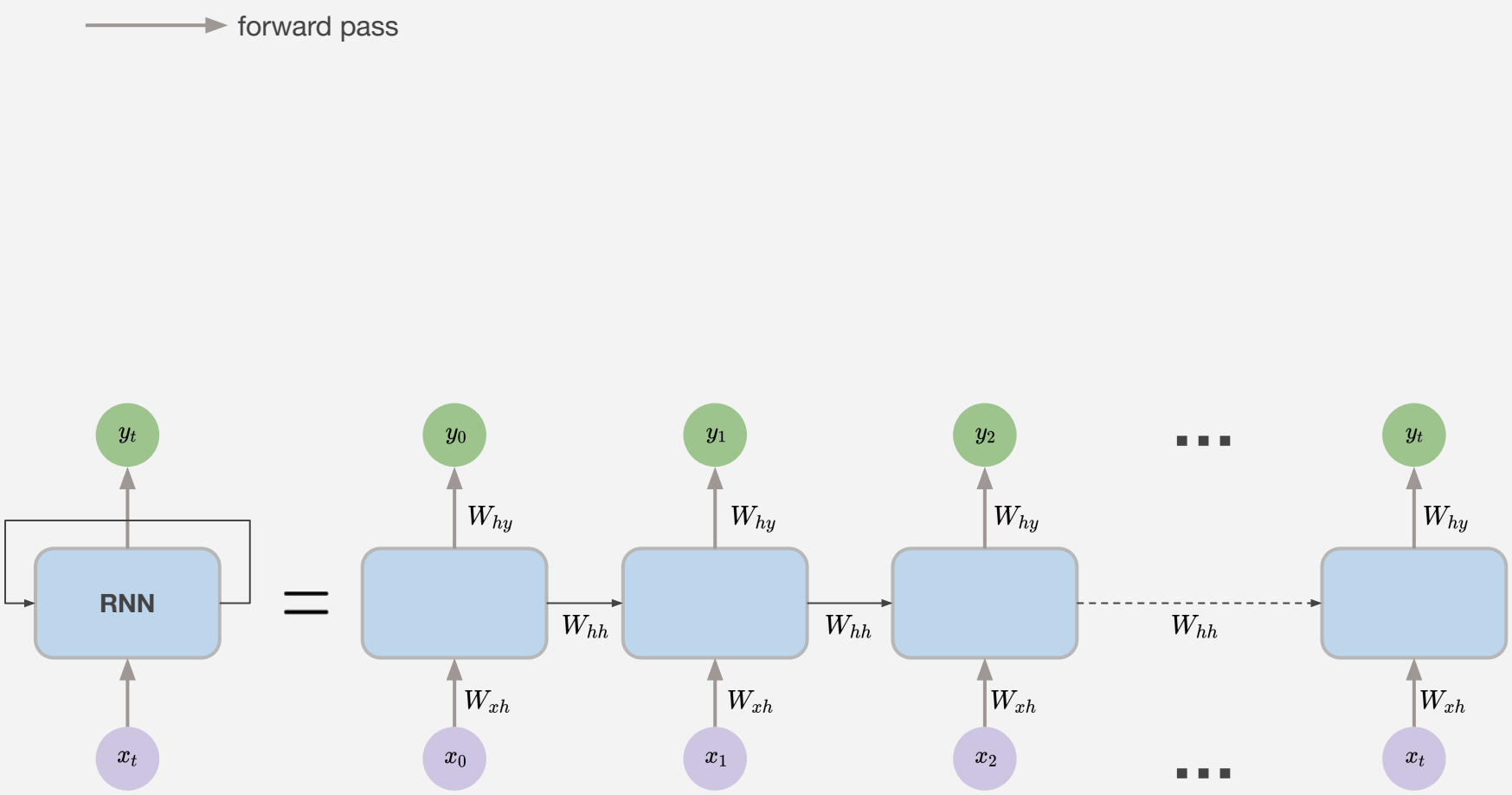
Recap: Backpropagation in Feedforward NNs



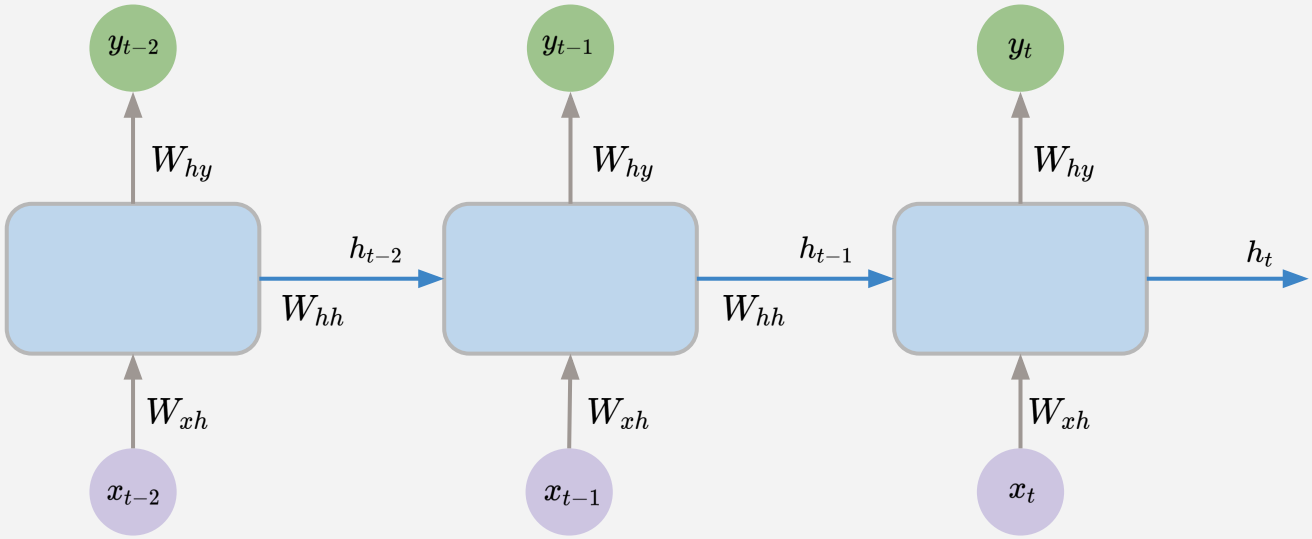
Backpropagation

1. Take the gradient of the loss with respect to each parameter (remember the chain rule!)
2. Update parameters (small nudge) in the direction of the gradient to minimize loss

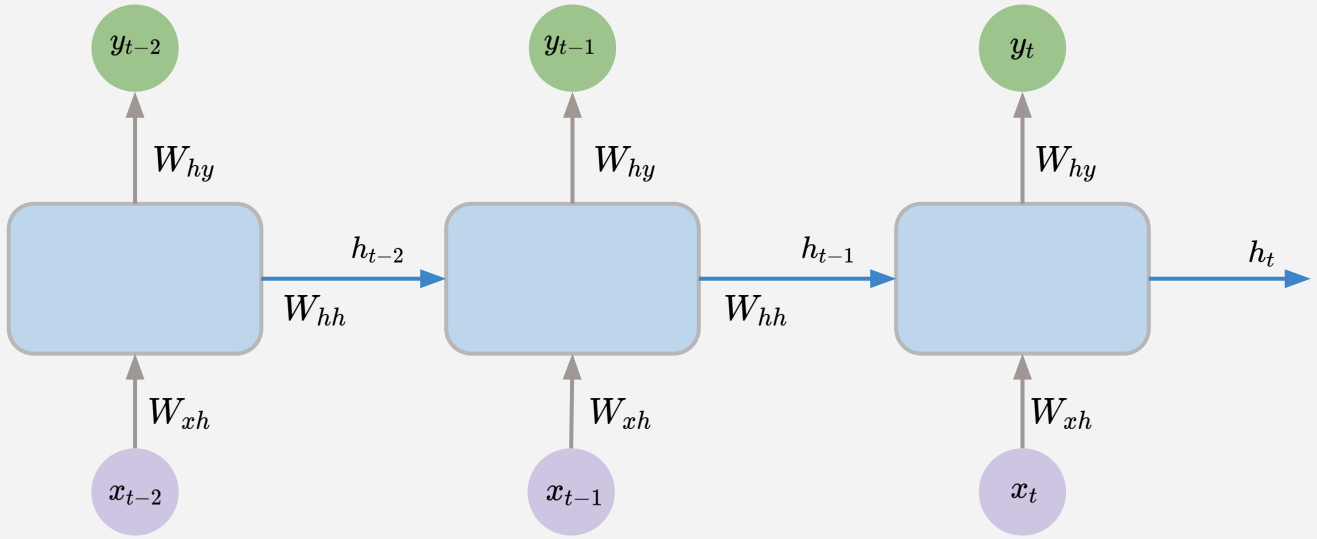
RNNs: Backpropagation Through Time (BPTT)



RNNs: Forward Pass

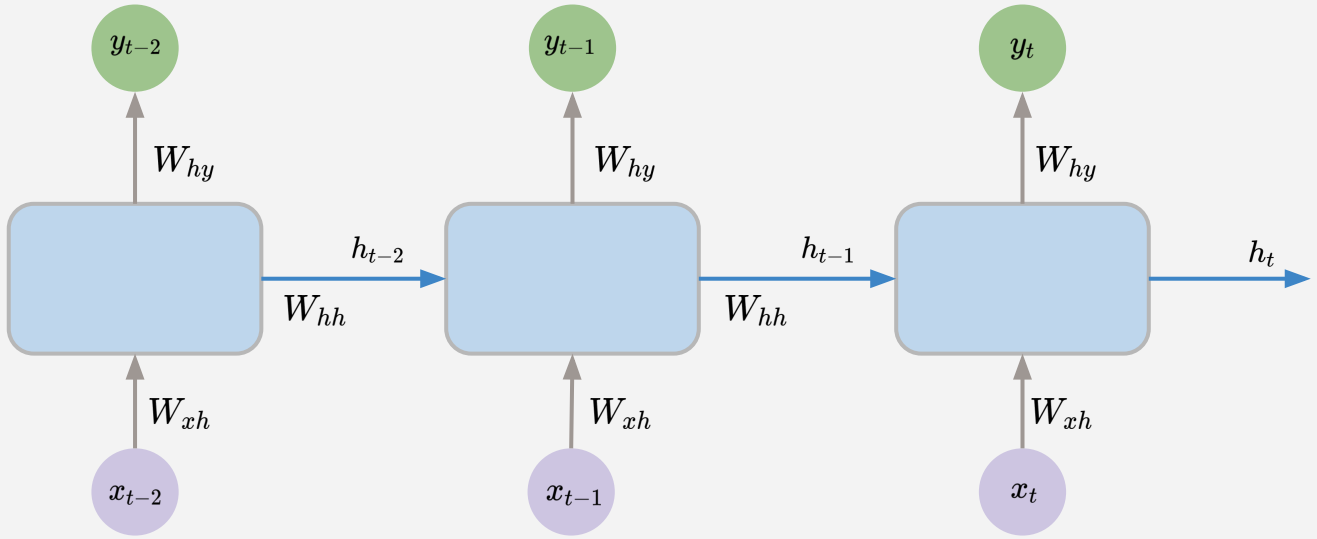


RNNs: Forward Pass



The forward pass **computes hidden states** in each time step.
Like a perceptron, the forward pass is a sum of weighted input, the hidden state and the bias, passed through an activation function

RNNs: Forward Pass



The forward pass **computes hidden states** in each time step.
Like a perceptron, the forward pass is a sum of weighted input, the hidden state and the bias, passed through an activation function

$$h_t = \tanh\left(\boxed{W_{xh} \cdot x_t} + \boxed{W_{hh} \cdot h_{t-1}} + \boxed{b_t}\right)$$

weighted input

weighted previous hidden state

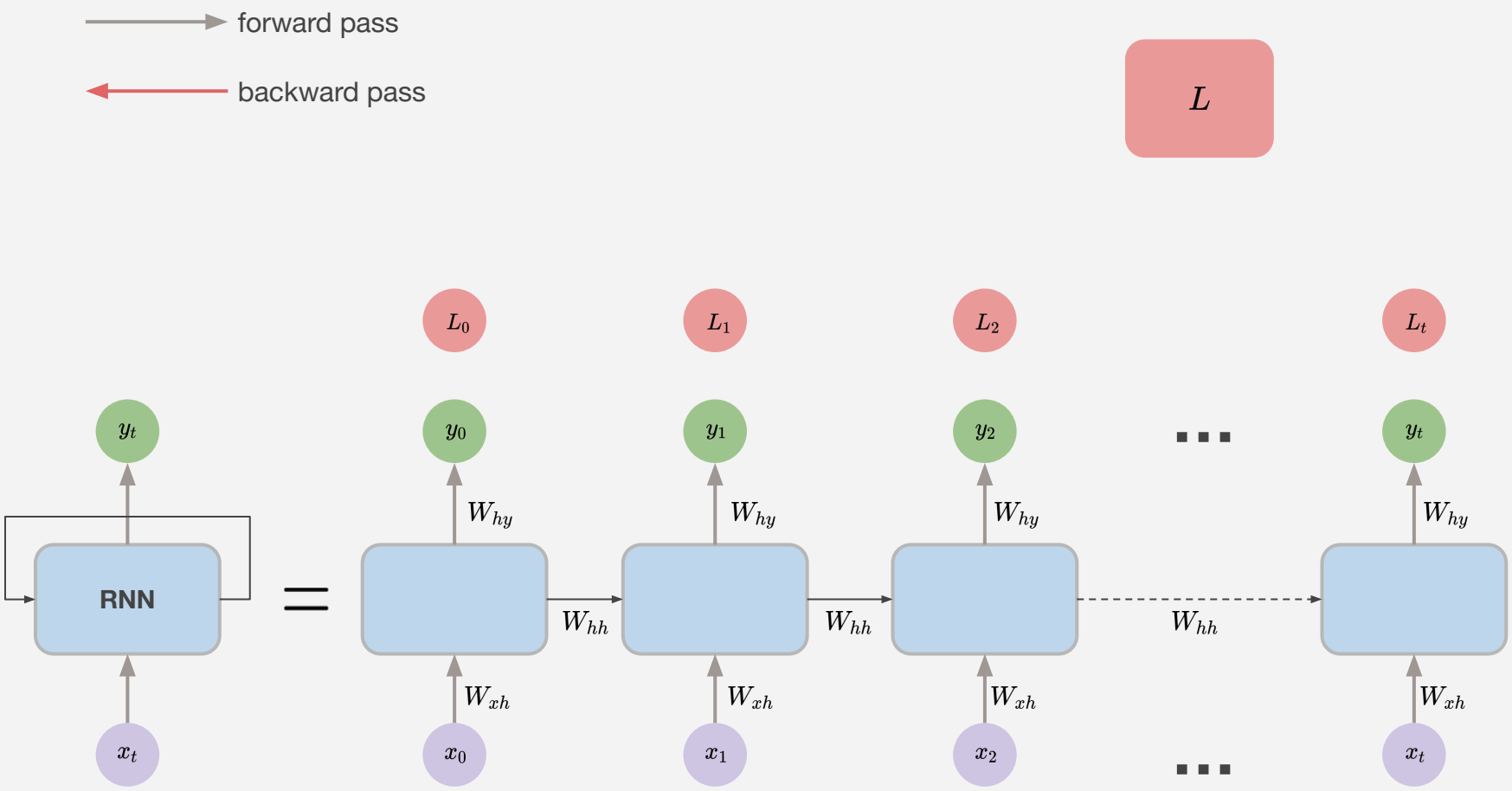
bias

$$y_t = \sigma\left(\boxed{W_{hy} \cdot h_t} + \boxed{b_y}\right)$$

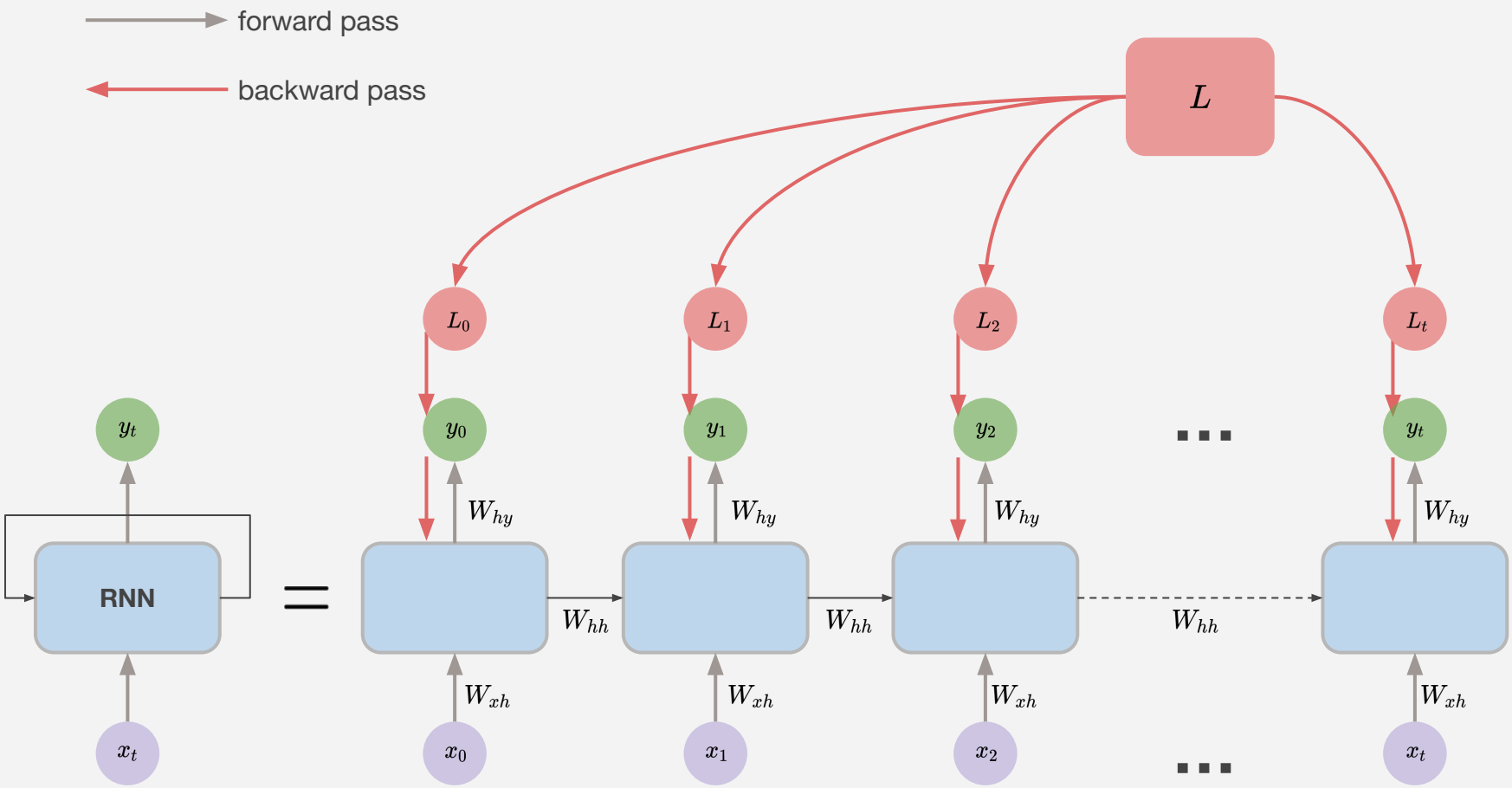
weighted hidden state

bias

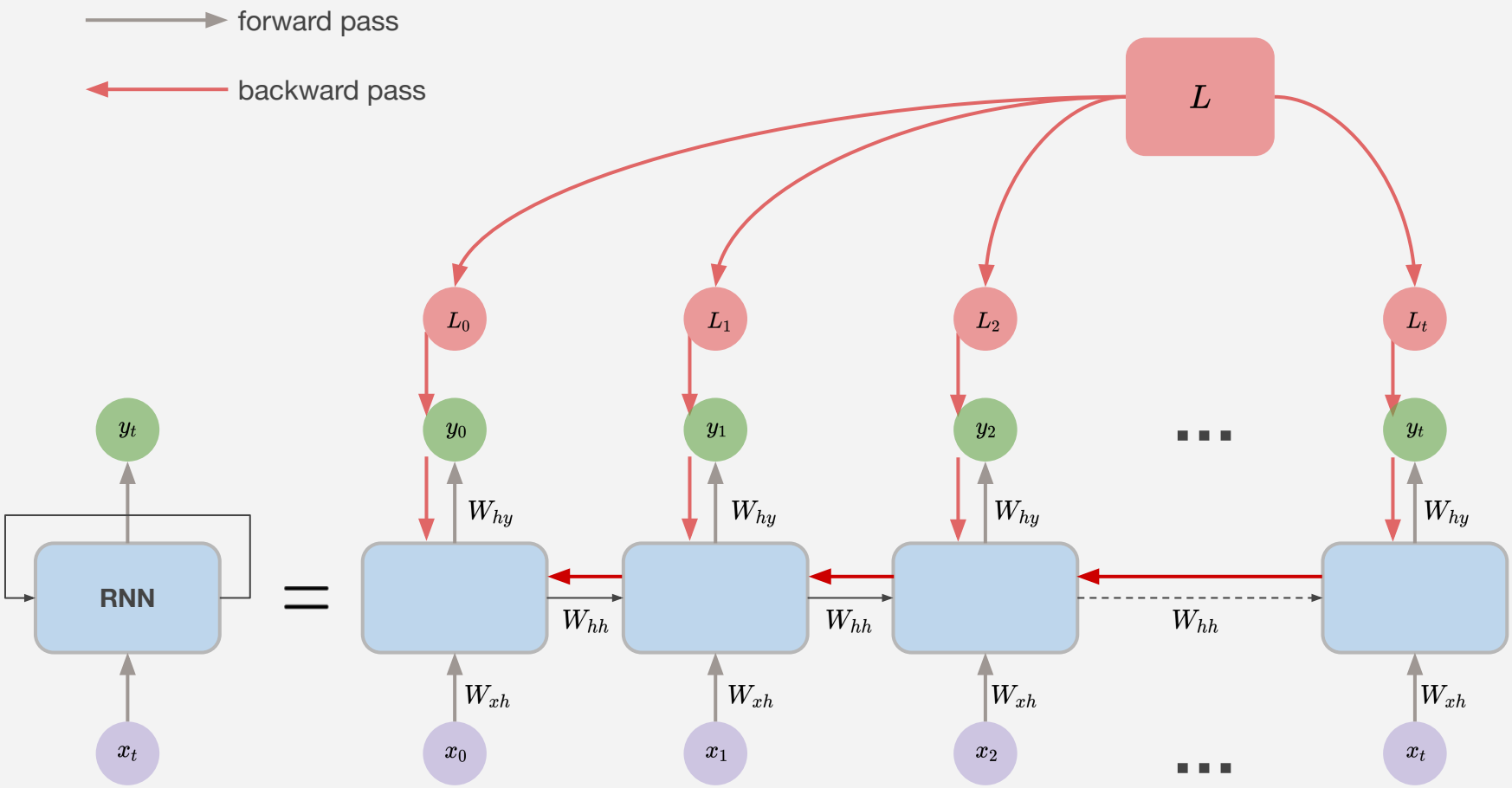
RNNs: Backpropagation Through Time (BPTT)



RNNs: Backpropagation Through Time (BPTT)



RNNs: Backpropagation Through Time (BPTT)



Oregon State University

Backpropagation Through Time (BPTT)

Similar to the feed forward network, we will use Mean Squared Error Loss.

For ***N*** time steps:

$$L = \frac{1}{N}\sum_{t=1}^{N}(y - y_t)^2$$

The derivative for MSE:

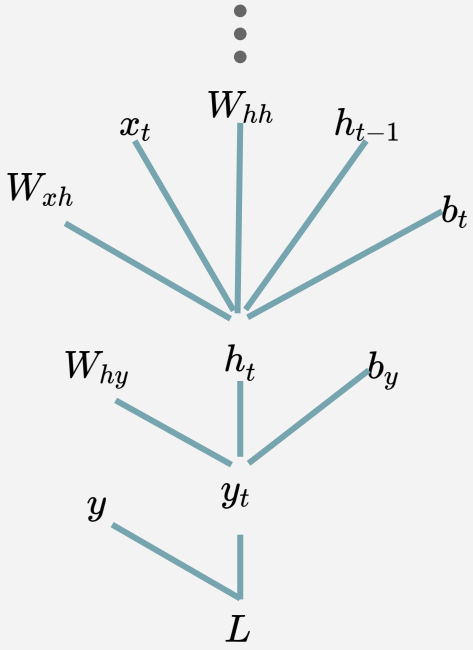
$$\frac{\partial L}{\partial y_t} = \frac{2}{N}\sum_{t=1}^{N}(y - y_t)$$

We have to find the derivative of loss with respect to three weights: W_{xh} W_{hy} W_{hh}

Backpropagation Through Time (BPTT)

We have to find the derivative of loss with respect to three weights: W_{xh}, W_{hy}, W_{hh}.

Use the value dependency tree to build the chain of partials required to compute the derivatives.



For a single time step, the derivative of loss wrt W_{xh} :

$$\frac{\partial L}{\partial W_{xh}} = \frac{\partial L}{\partial y_t}\frac{\partial y_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{dW_{xh}}$$

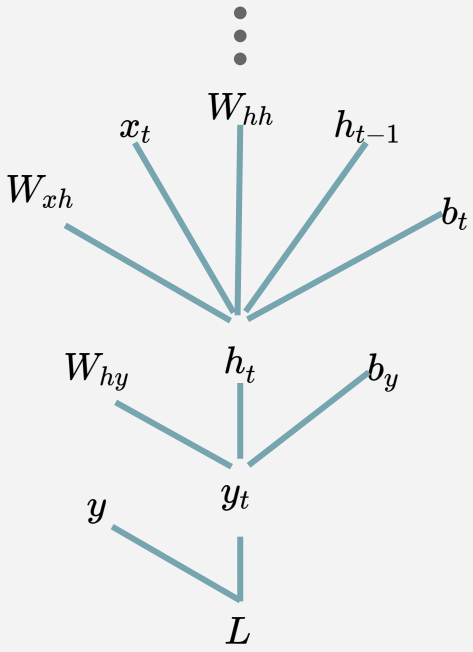
For a two time steps:

$$\frac{\partial L}{\partial W_{xh}} = \frac{\partial L}{\partial y_t}\frac{\partial y_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{dW_{xh}} + \frac{\partial L}{\partial y_{t-1}}\frac{\partial y_{t-1}}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{\partial h_{t-2}}{\partial W xh}$$

Backpropagation Through Time (BPTT)

More generally,

$$\frac{\partial L}{\partial W_{xh}} = \sum_{i=0}^{t} \frac{\partial L}{\partial y_{t-i}} \frac{\partial y_{t-i}}{\partial h_{t-i}} \left(\prod_{j=t-i+1}^{t} \frac{\partial h_{t-j+1}}{\partial h_{t-j}} \right) \frac{\partial h_{t-i-1}}{\partial W_{xh}}$$



Backpropagation Through Time (BPTT)

More generally,

$$\frac{\partial L}{\partial W_{xh}} = \sum_{i=0}^{t} \frac{\partial L}{\partial y_{t-i}} \frac{\partial y_{t-i}}{\partial h_{t-i}} \left(\prod_{j=t-i+1}^{t} \frac{\partial h_{t-j+1}}{\partial h_{t-j}} \right) \frac{\partial h_{t-i-1}}{\partial W_{xh}}$$

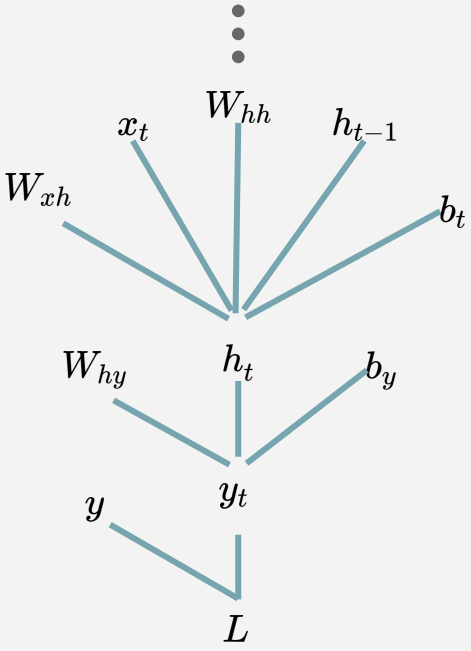
For sequence of length *t*

How sensitive the loss is to change in output

How sensitive the output is to change in the hidden state

Sensitivity to the chain of hidden states from previous time steps

How sensitive the hidden state is to changes in the weight



Oregon State University

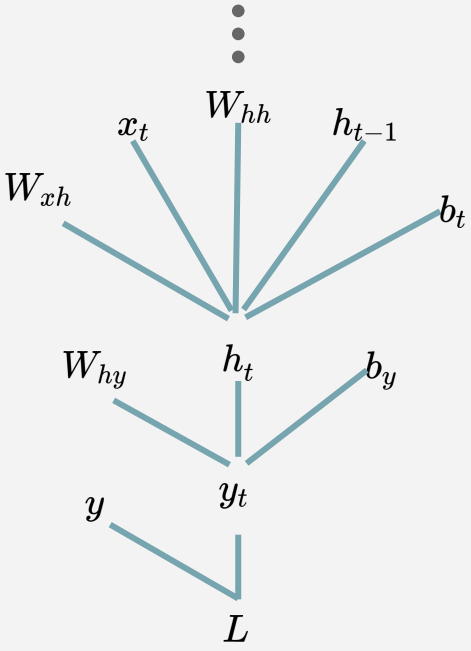
Backpropagation Through Time (BPTT)

The derivative of loss wrt W_{hh} can be computed using the exact same chain of dependencies :

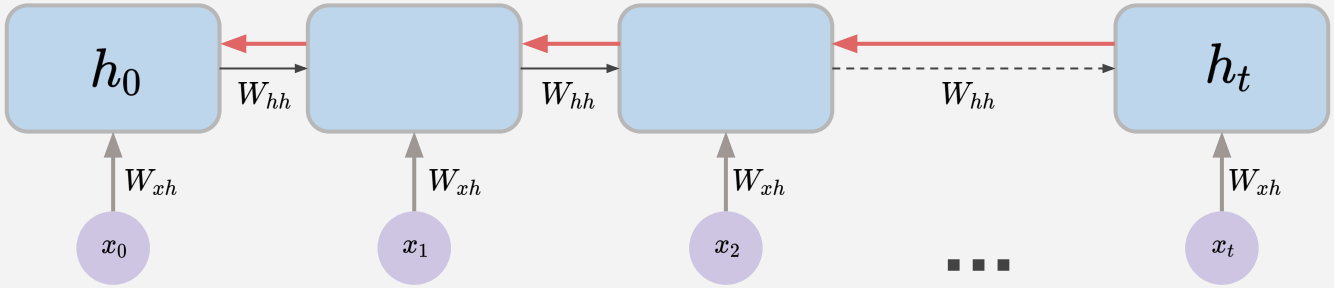
$$\frac{\partial L}{\partial W_{hh}} = \sum_{i=0}^{t} \frac{\partial L}{\partial y_{t-i}} \frac{\partial y_{t-i}}{\partial h_{t-i}} \left(\prod_{j=t-i+1}^{t} \frac{\partial h_{t-j+1}}{\partial h_{t-j}} \right) \frac{\partial h_{t-i-1}}{\partial W_{hh}}$$

Finally, the weights connecting the output are sensitive only to the outputs and output weights for the sequence:

$$\frac{\partial L}{\partial W_{hy}} = \sum_{i=0}^{t} \frac{\partial L}{\partial y_{t-i}} \frac{\partial y_{t-i}}{\partial W_{hy}}$$

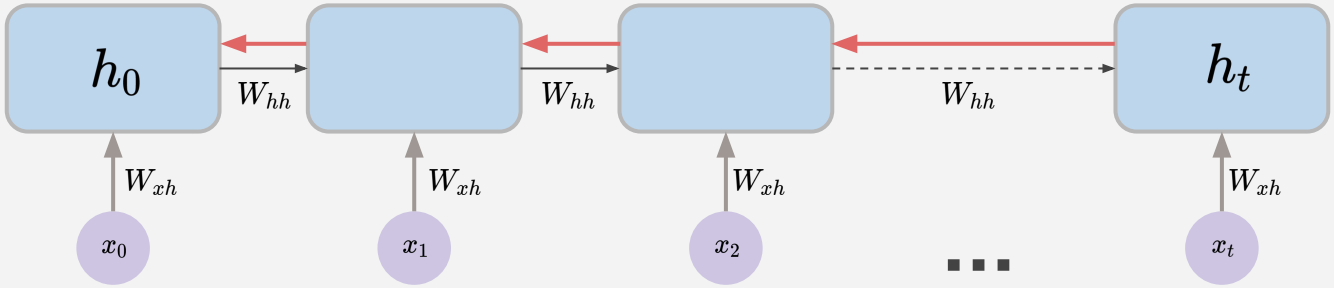


RNNs: Backpropagation Through Time (BPTT)



Remember the chain rule gradient update: computing **gradient** wrt h_0 involves **many factors** of W_{hh}

Backpropagation Through Time: Exploding Gradients

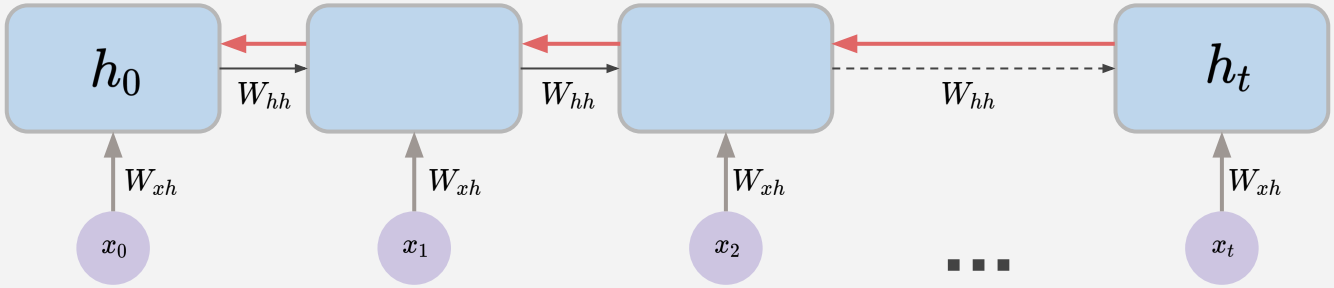


Remember the chain rule gradient update: computing **gradient** wrt h_0 involves **many factors** of W_{hh}

Weights > 1.0 = **exploding gradient**

1. Gradient clipping that scales gradients
2. Batch normalization

Backpropagation Through Time: Vanishing Gradients



Remember the chain rule gradient update: computing **gradient** wrt h_0 involves **many factors** of W_{hh}

Weights > 1.0 = **exploding gradient**

Weights < 1.0 = **vanishing gradient**

1. Gradient clipping that scales gradients
2. Batch normalization

RNNs: Vanishing Gradients

Why are vanishing gradients a problem in RNNs?

RNNs: Vanishing Gradients

Successive products of **small weights**

Errors further back in the **history** have **smaller gradients**

Weights capture **short-term persistence**

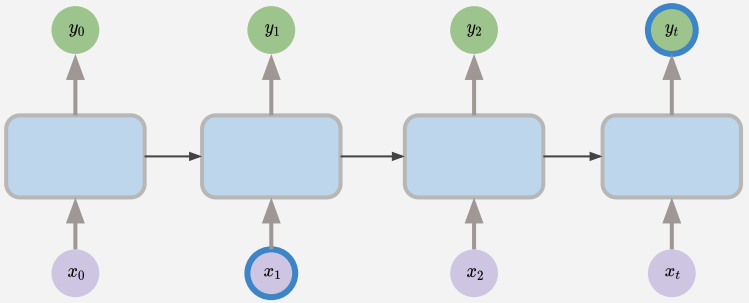
RNNs: Vanishing Gradients

Successive products of **small weights**

Errors further back in the **history** have **smaller gradients**

Weights capture **short-term persistence**

The clouds are in the ___



short-term persistence is adequate

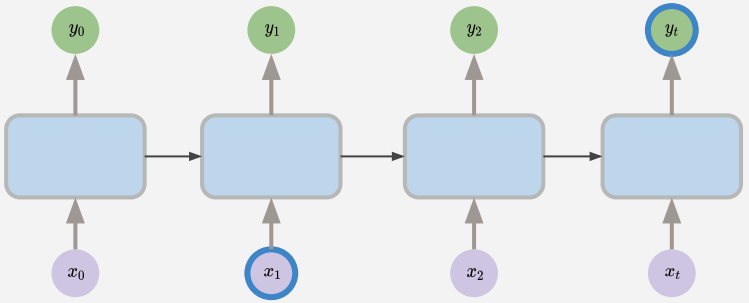
RNNs: Vanishing Gradients

Successive products of **small weights**

Errors further back in the **history** have **smaller gradients**

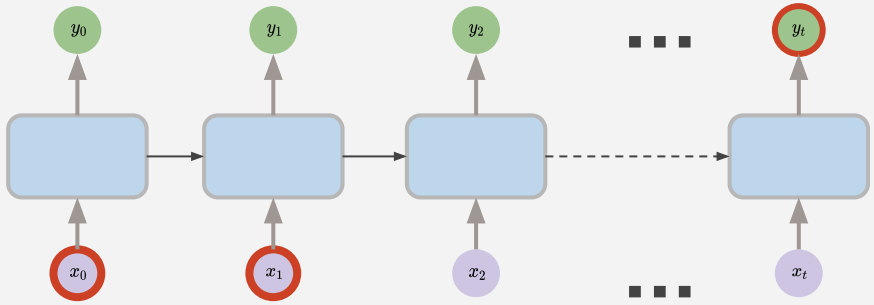
Weights capture **short-term persistence**

The clouds are in the ___



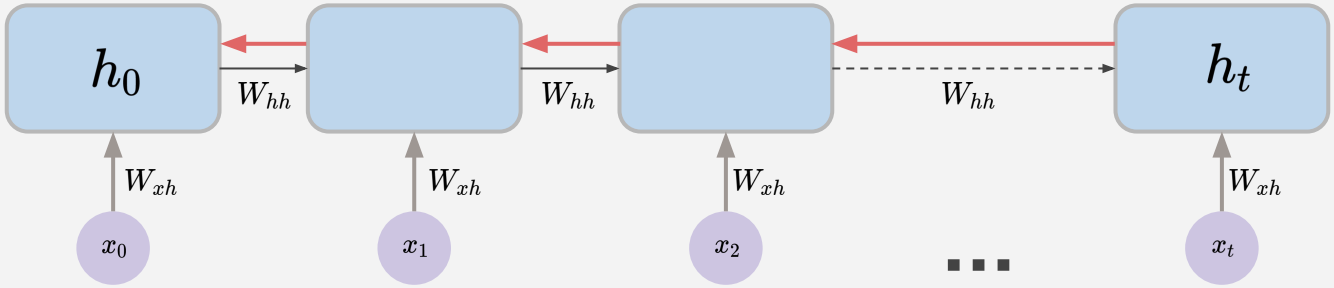
short-term persistence is adequate

I grew up in France. I now live in Seattle. But I speak fluent ___



long-term persistence is **necessary**!

Backpropagation Through Time: Vanishing Gradients



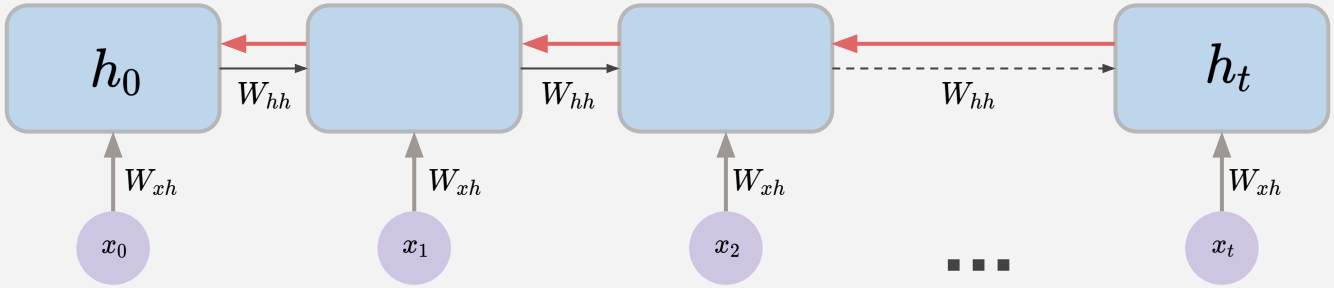
Remember the chain rule gradient update: computing **gradient** wrt h_0 involves **many factors** of W_{hh}

Weights > 1.0 = **exploding gradient**

Weights < 1.0 = **vanishing gradient**

1. Gradient clipping that scales gradients
2. Batch normalization

Backpropagation Through Time: Vanishing Gradients



Remember the chain rule gradient update: computing **gradient** wrt h_0 involves **many factors** of W_{hh}

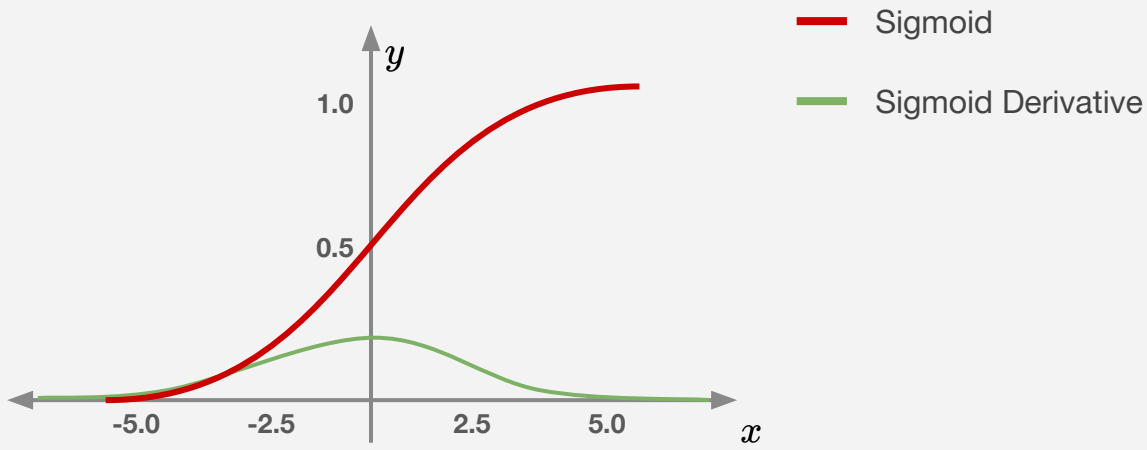
Weights > 1.0 = **exploding gradient**

1. Gradient clipping that scales gradients
2. Batch normalization

Weights < 1.0 = **vanishing gradient**

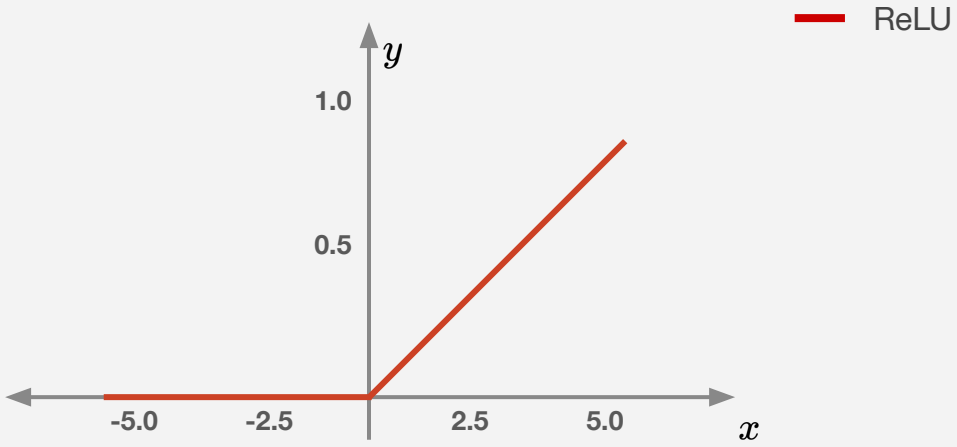
1. Activation functions
2. Weight initialization
3. Network architecture (use gates!)

Vanishing Gradients: Solution 1

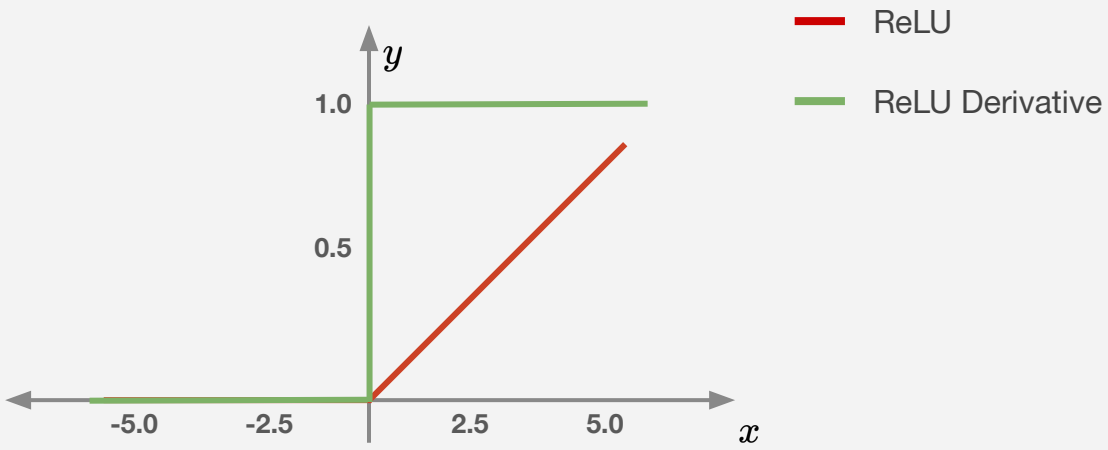


Recap: The derivative of sigmoid plateaus and gets close to zero

Vanishing Gradients: Solution 1



Vanishing Gradients: Solution 1



Using a ReLU activation function prevents the gradients from shrinking

Vanishing Gradients: Solution 2

- Initialize weight matrices to identity matrix
- Reconfigure number of layers

Vanishing Gradients: Solution 3

Use **gates** to regulate how information is passed between successive recurrent cells

Vanishing Gradients: Solution 3

Use **gates** to regulate how information is passed between successive recurrent cells

- Long Short Term Memory (LSTMs) Networks
- Gated Recurrent Networks (GRUs)

These networks use gated cells to control the flow of information and help with **long-term persistence**

LSTMs

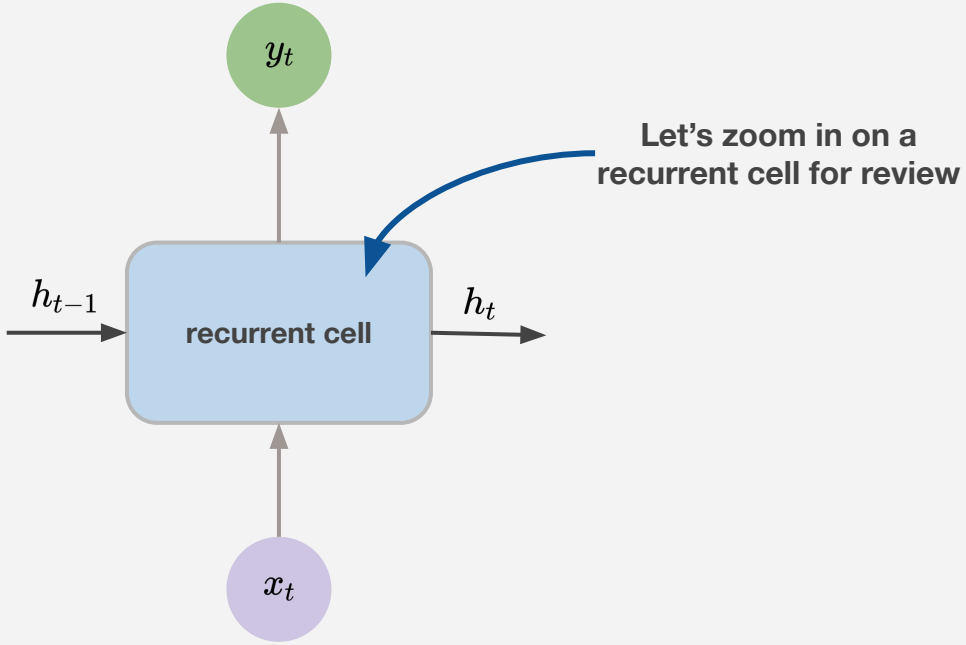
1. Maintain a persistent cell state - robust to gradient updates

2. Gates to regulate information
   a. **Forget gate** to remove unnecessary information
   b. **Input gate** to selectively use the current information
   c. **Output gate** to propagate the modified cell state to the next cell

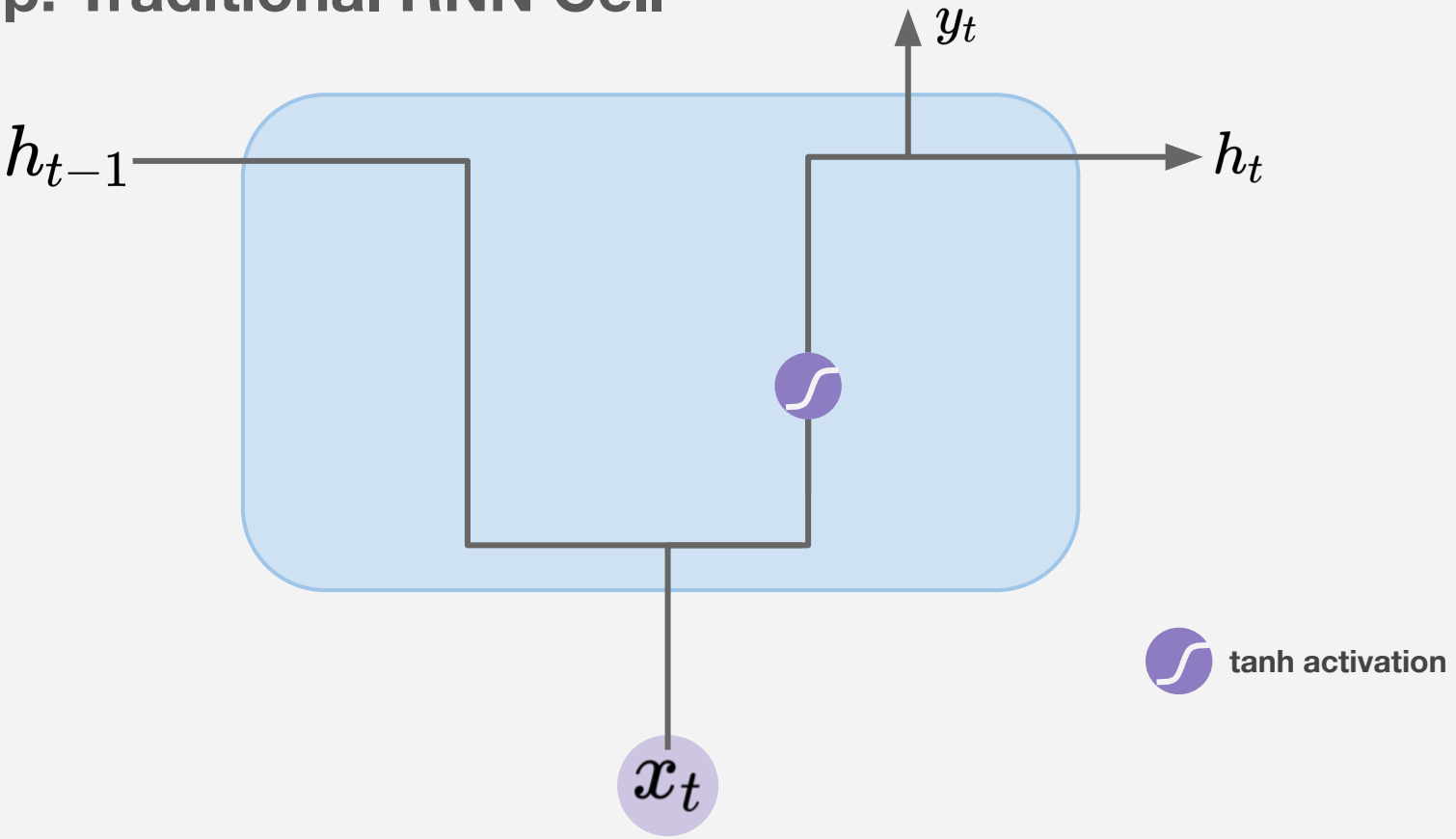
3. Gates are parameterized

   Backpropagation through time can update gradients to **control gates** for **long-term persistence**

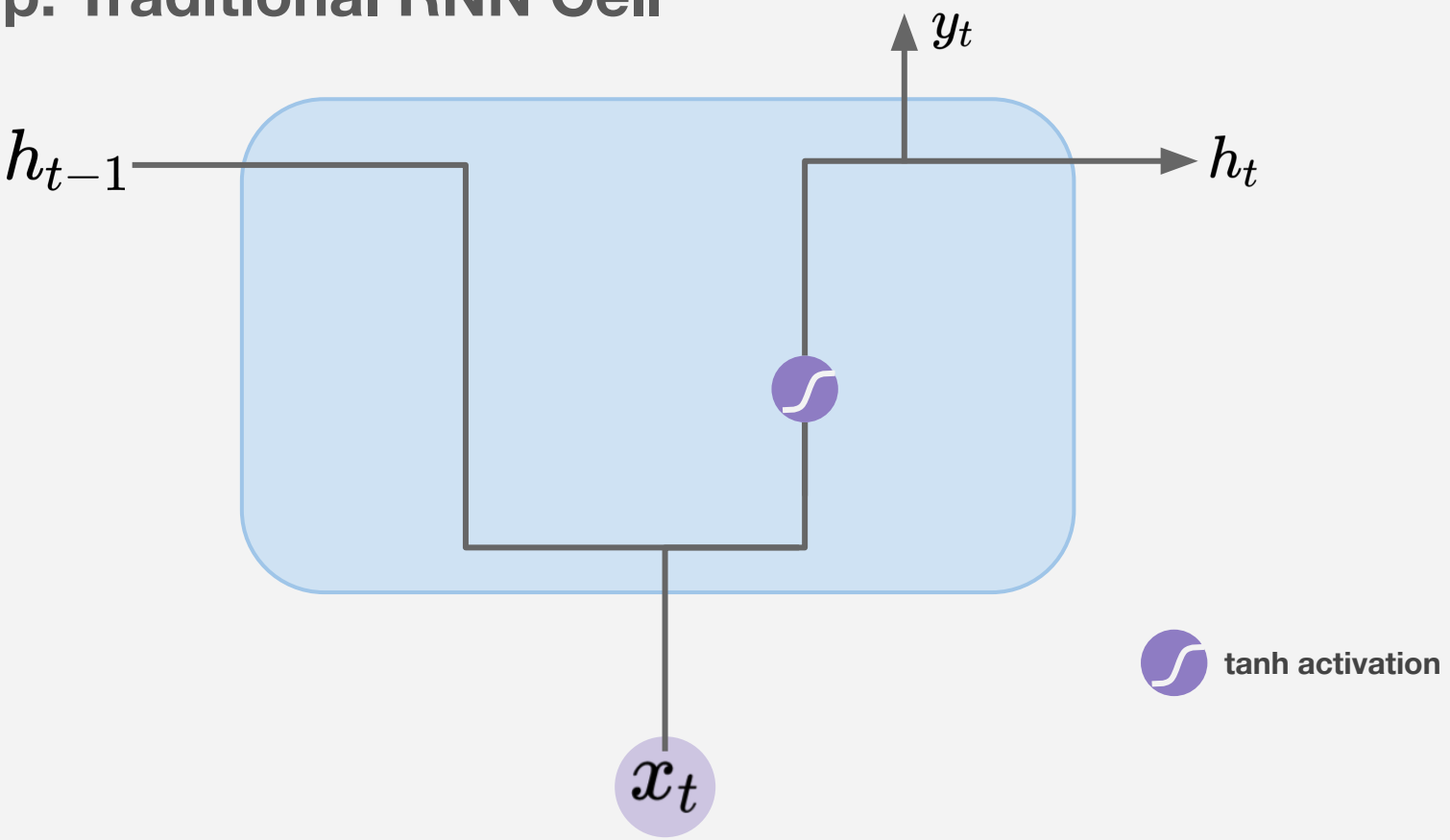
Recap: Traditional RNN Cell



Recap: Traditional RNN Cell



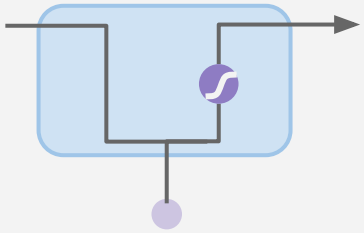
Recap: Traditional RNN Cell



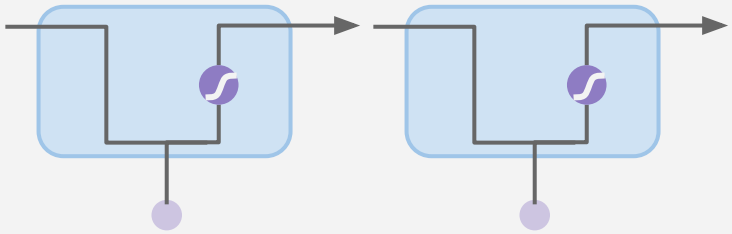
$$h_t = \tanh\left(W.\left[h_{t-1}, x_t\right] + b_t\right)$$

The next hidden state h_t is the sum of the weighted combination of the previous hidden state and input, with the bias, passed through a tanh activation

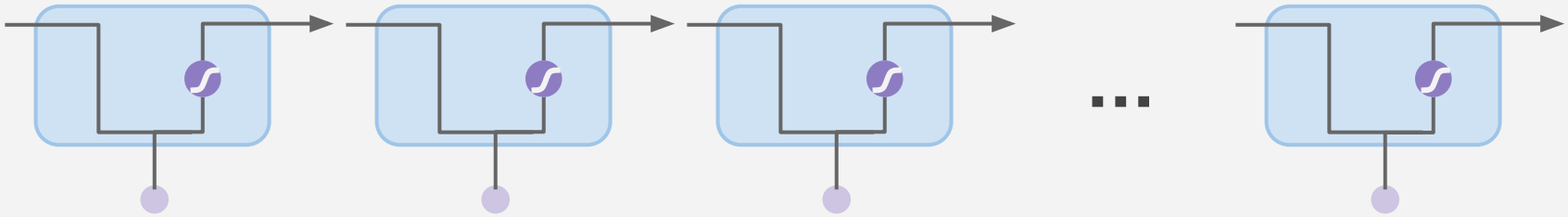
Recap: Traditional RNNs



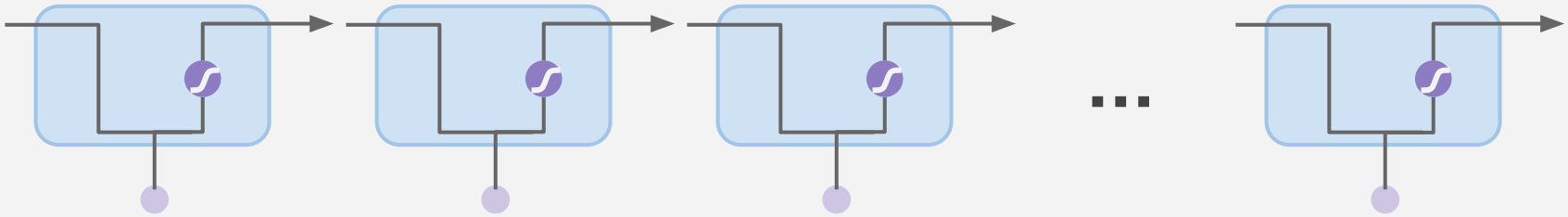
Recap: Traditional RNNs



Recap: Traditional RNNs - Vanishing Gradient



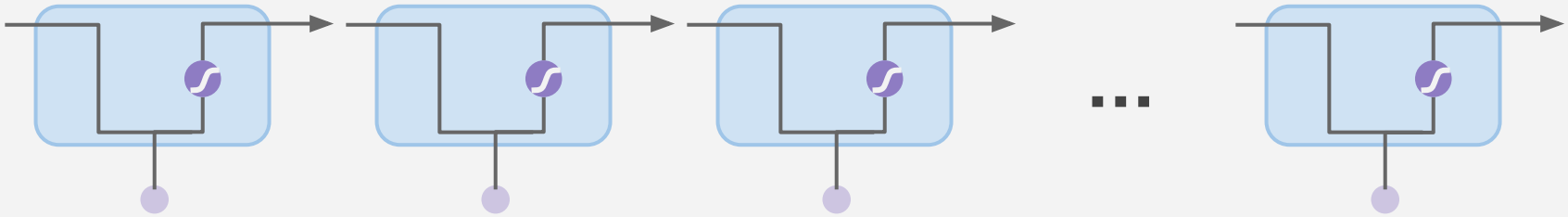
Recap: Traditional RNNs - Vanishing Gradient



RNNs

- effective for capturing **short-term dependencies**
- vanishing gradients eliminate potentially useful information from the the hidden states as you go further back in time

Recap: Traditional RNNs - Vanishing Gradient



RNNs

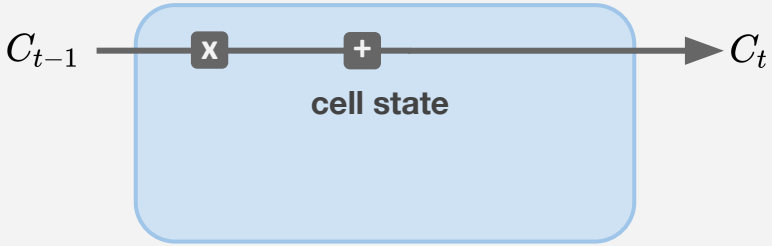
- effective for capturing **short-term dependencies**
- vanishing gradients eliminate potentially useful information from the the hidden states as you go further back in time

Gates in an LSTM will regulate gradient flow and effectively manage the vanishing gradient problem

LSTM Cell: Components

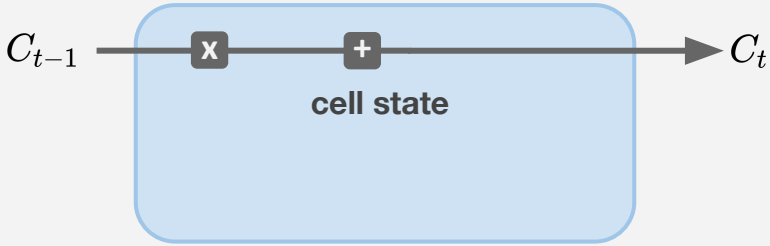
LSTM Cell: Components

- The **cell state** is an uninterrupted information pathway, with only minor linear interactions, that goes across all time steps through all cells



LSTM Cell: Components

- The **cell state** is an uninterrupted information pathway, with only minor linear interactions, that goes across all time steps through all cells



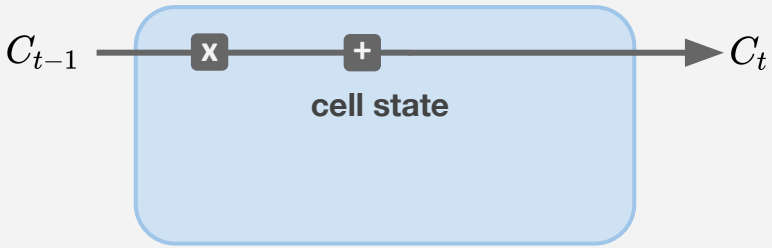
- The **pointwise** vector addition and multiplication allow the gates to propagate information

X **pointwise multiplication**

\+ **pointwise addition**

LSTM Cell: Components

- The **cell state** is an uninterrupted information pathway, with only minor linear interactions, that goes across all time steps through all cells



- The **pointwise** vector addition and multiplication allow the gates to propagate information

X **pointwise multiplication**

\+ **pointwise addition**

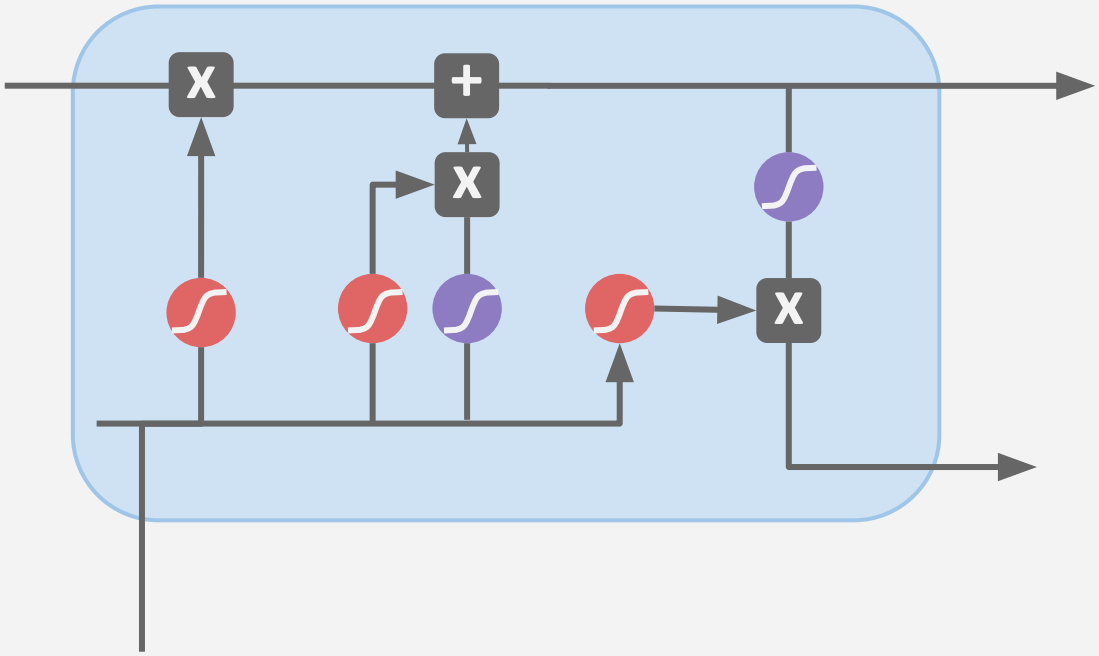
- The **sigmoid** activation squishes values between 0 and 1 which allows gates to control how much information should be let through.
  - a value of zero means “let nothing through”
  - a value of one means “allow everything”

sigmoid activation

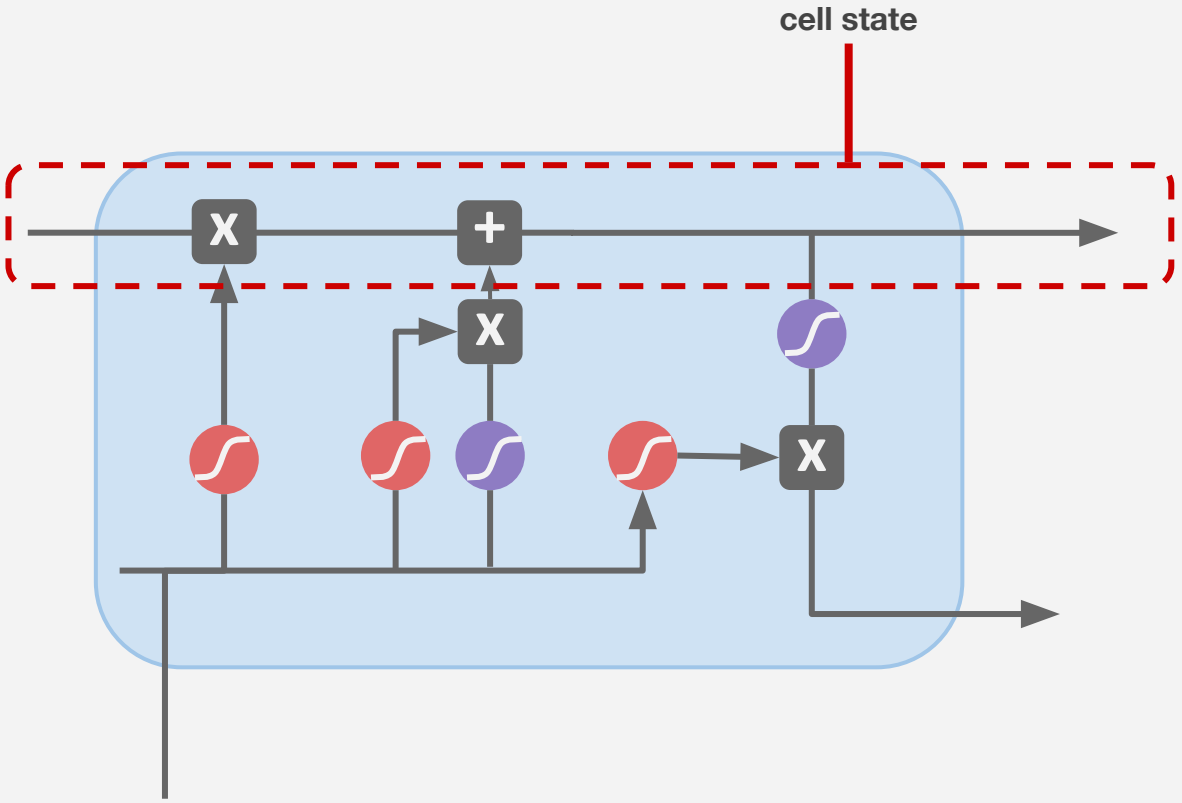
tanh activation

- The **tanh** activation is similar to the sigmoid, except it squishes output values between -1 and 1.

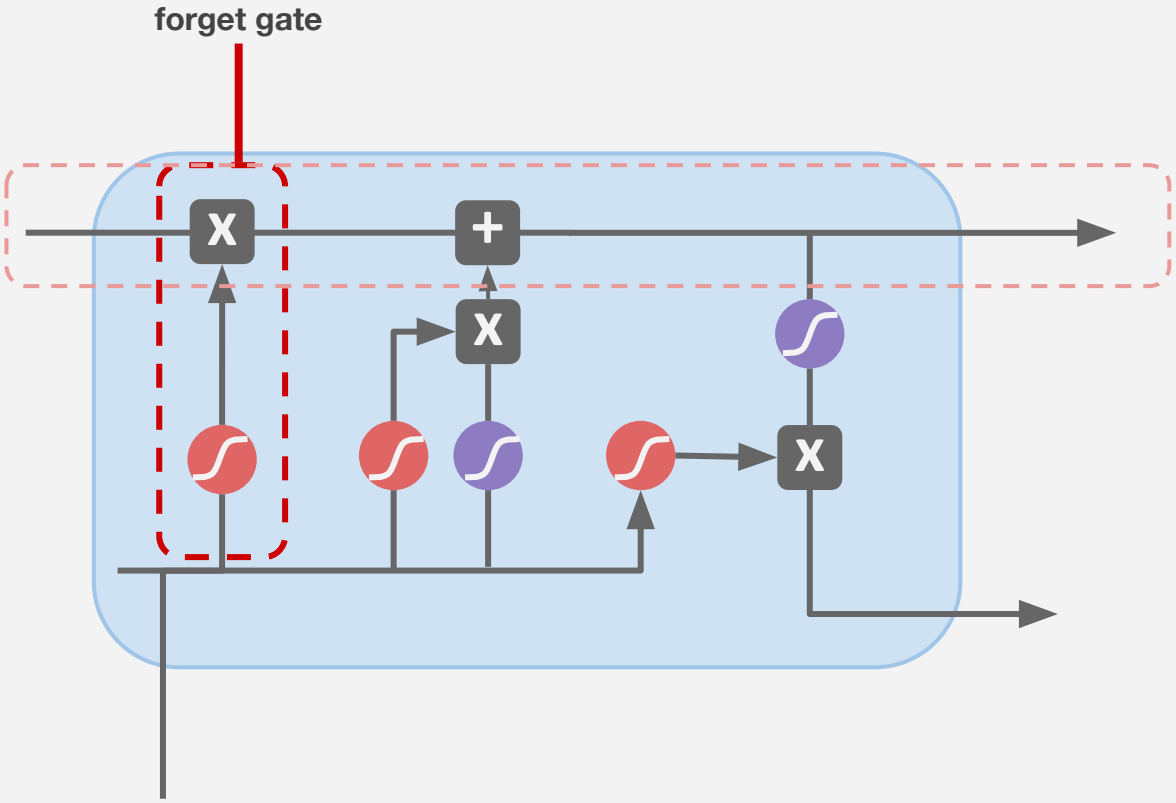
LSTM Cell



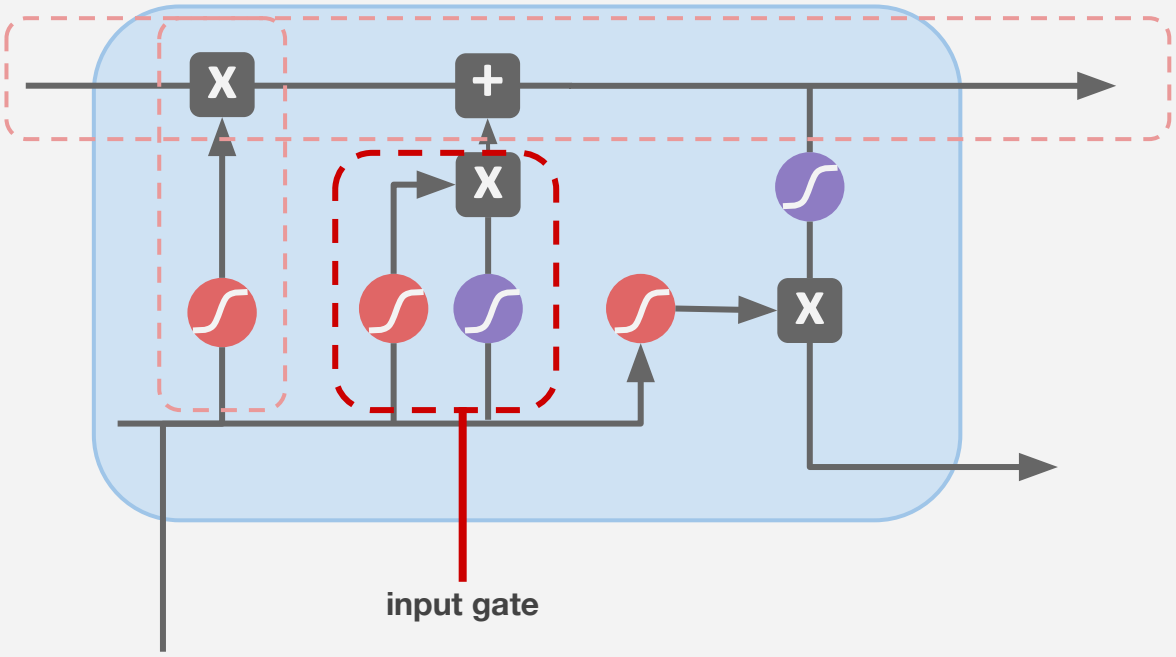
LSTM Cell



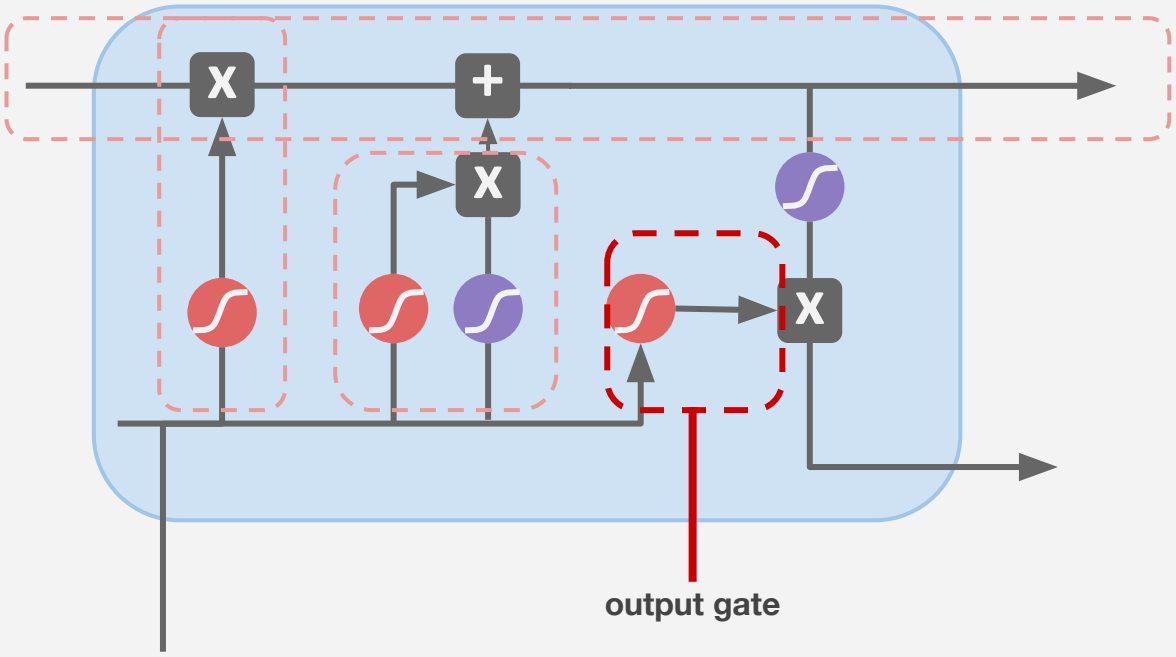
LSTM Cell



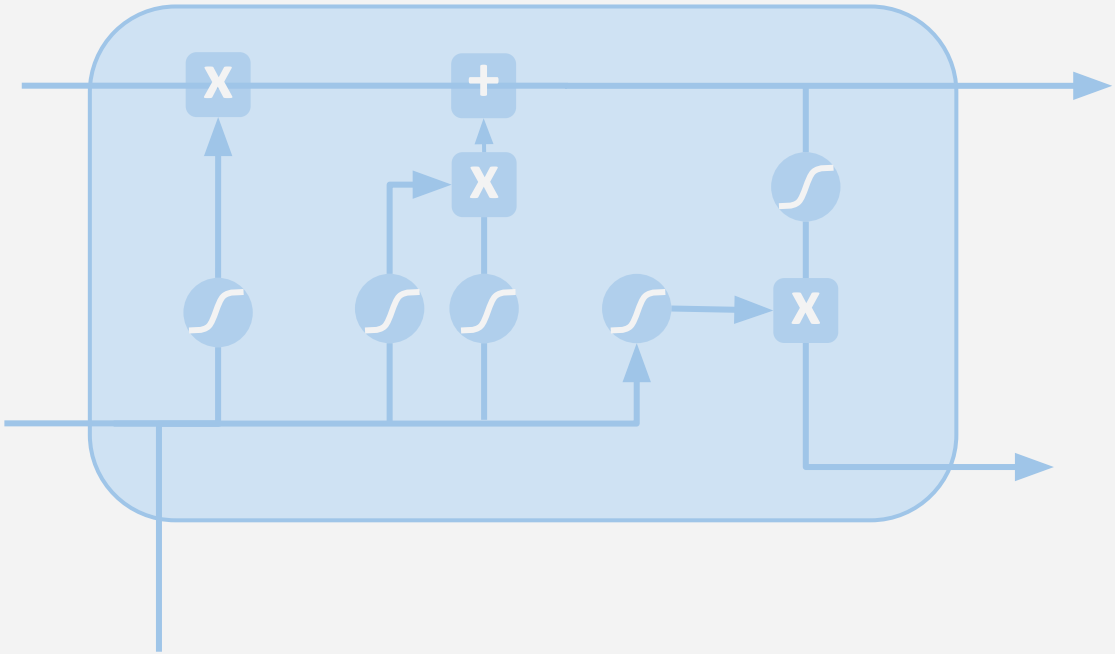
LSTM Cell



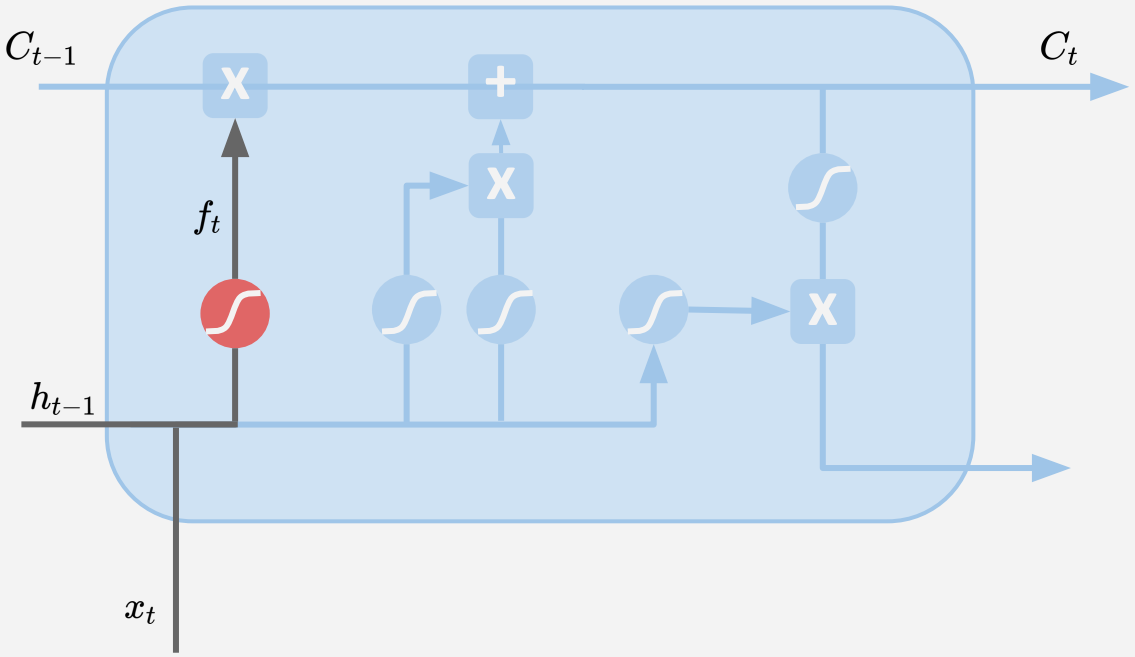
LSTM Cell



LSTM Cell: Forget Gate



LSTM Cell: Forget Gate

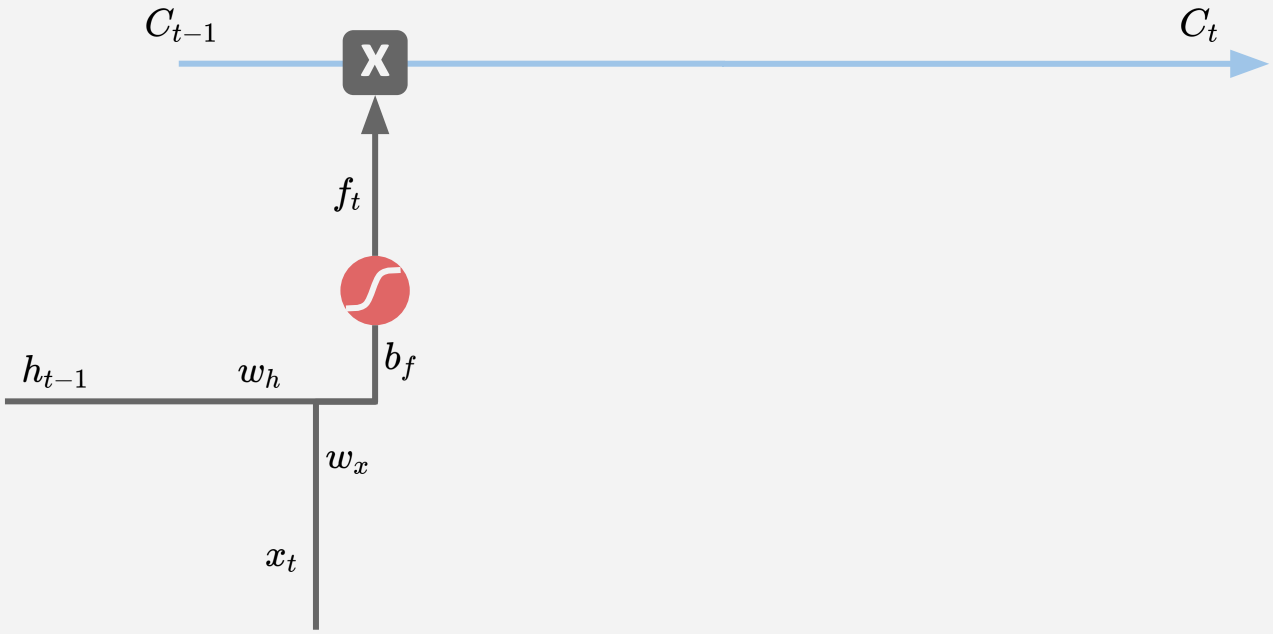


$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

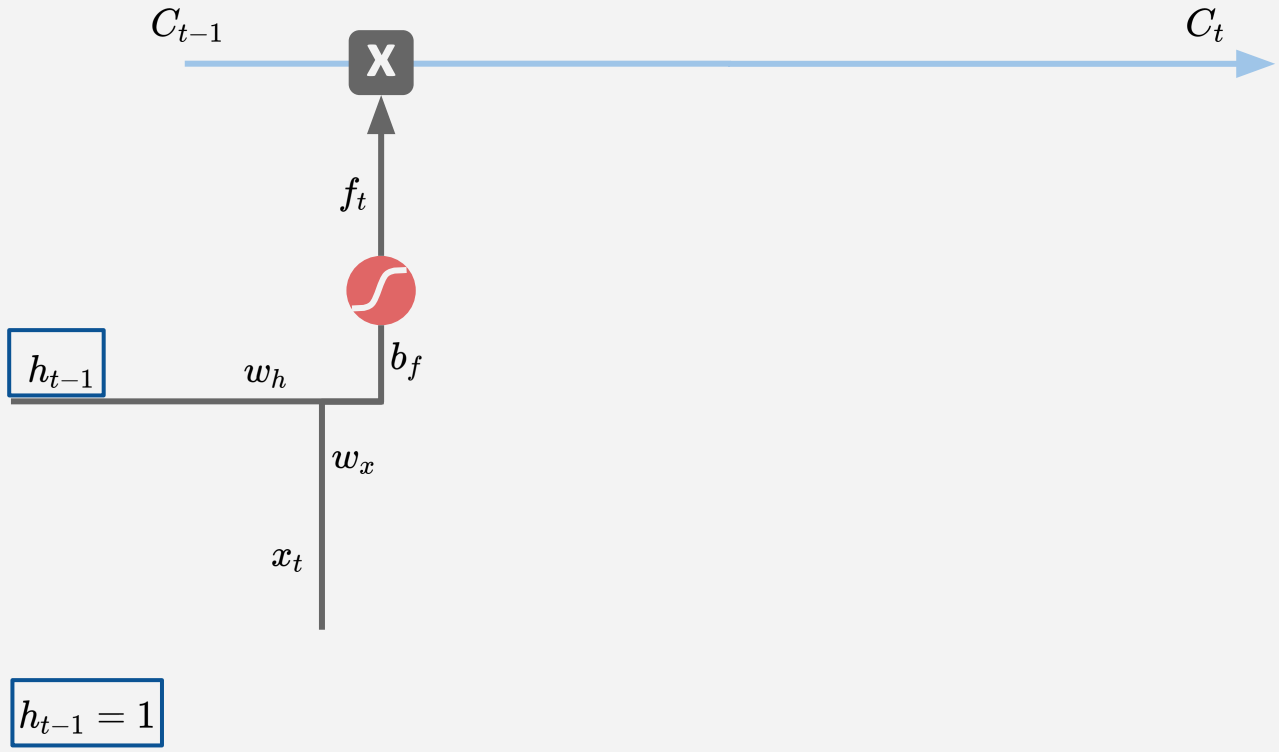
Forget Gate: Conditioned on the current input and the last hidden state, **decide how much information to throw away** from the cell state.

A one through the sigmoid activation means “keep all of f_t in the cell state”, whereas zero means “forget everything from the cell state”.

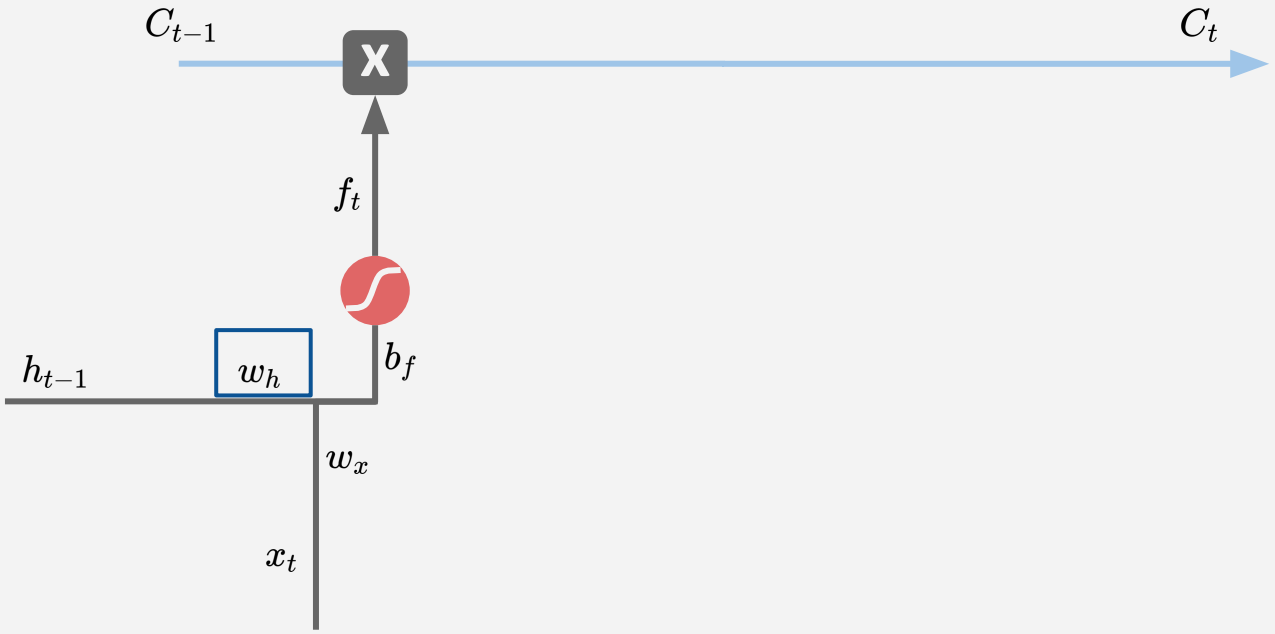
Forget Gate: Forward Pass



Forget Gate: Forward Pass



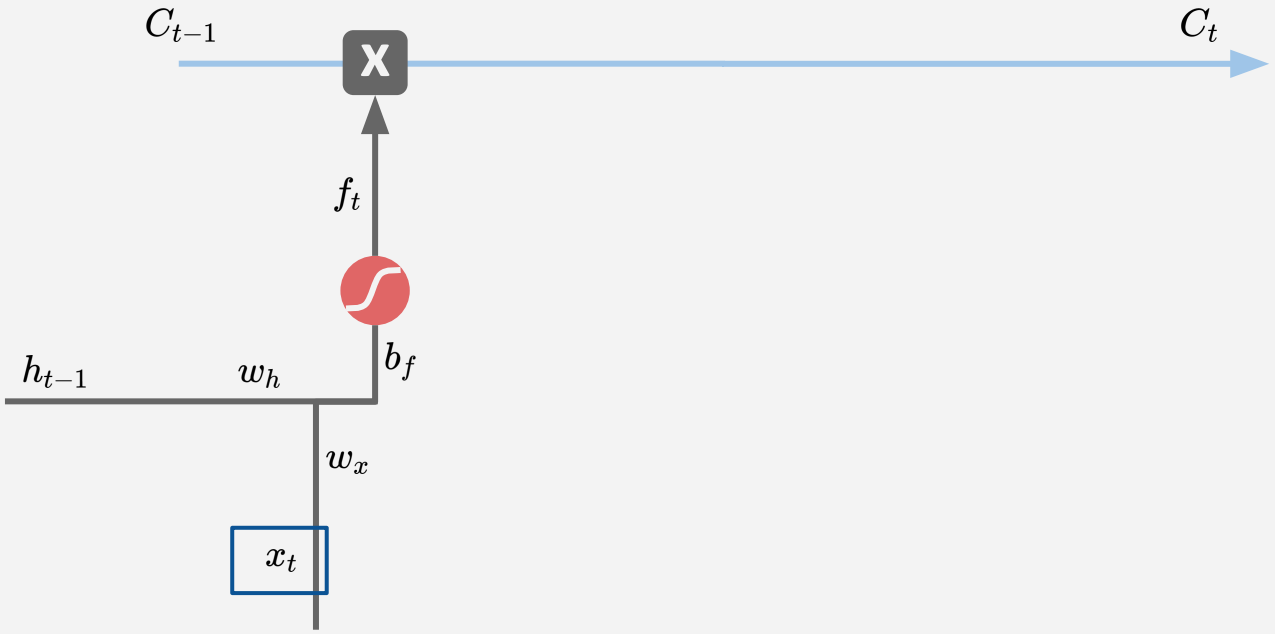
Forget Gate: Forward Pass



$h_{t-1} = 1$

$w_h = 2.7$

Forget Gate: Forward Pass

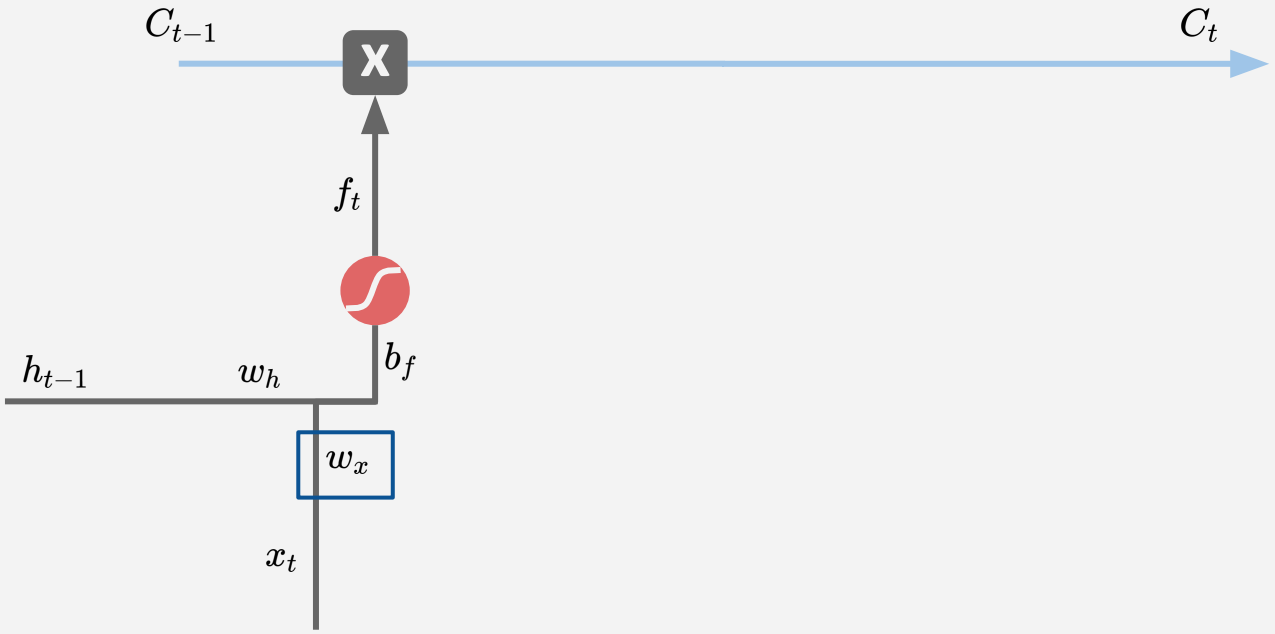


$h_{t-1} = 1$

$w_h = 2.7$

$x_t = 1$

Forget Gate: Forward Pass



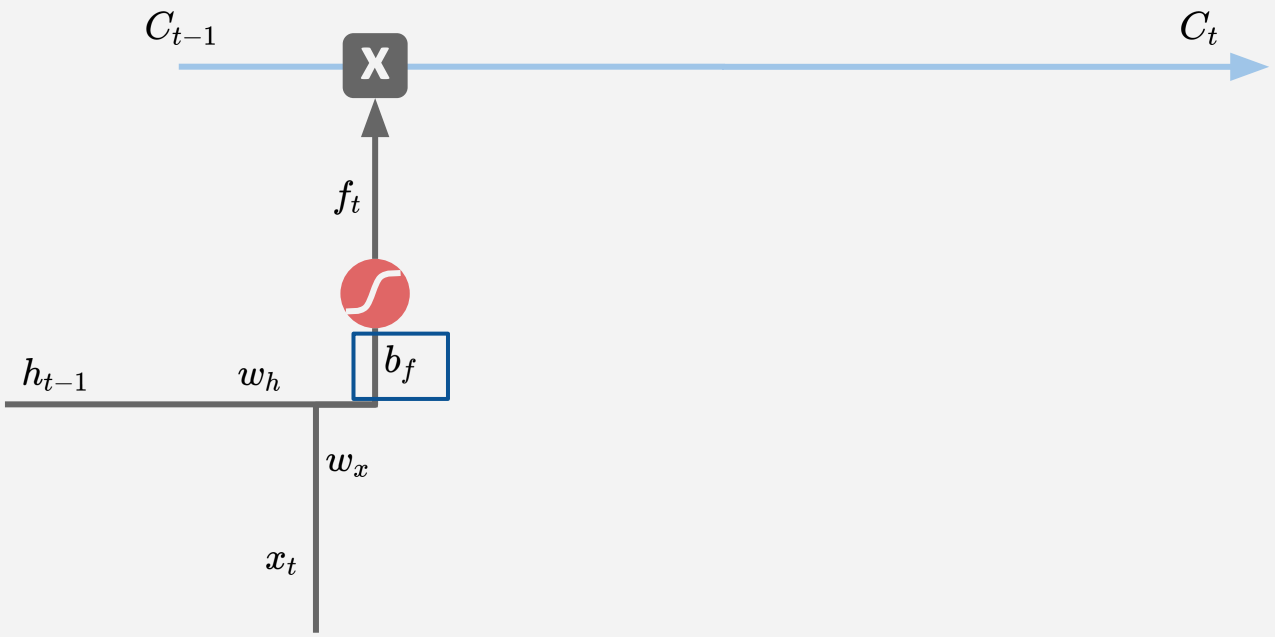
$h_{t-1} = 1$

$w_h = 2.7$

$x_t = 1$

$w_x = 1.7$

Forget Gate: Forward Pass



$h_{t-1} = 1$

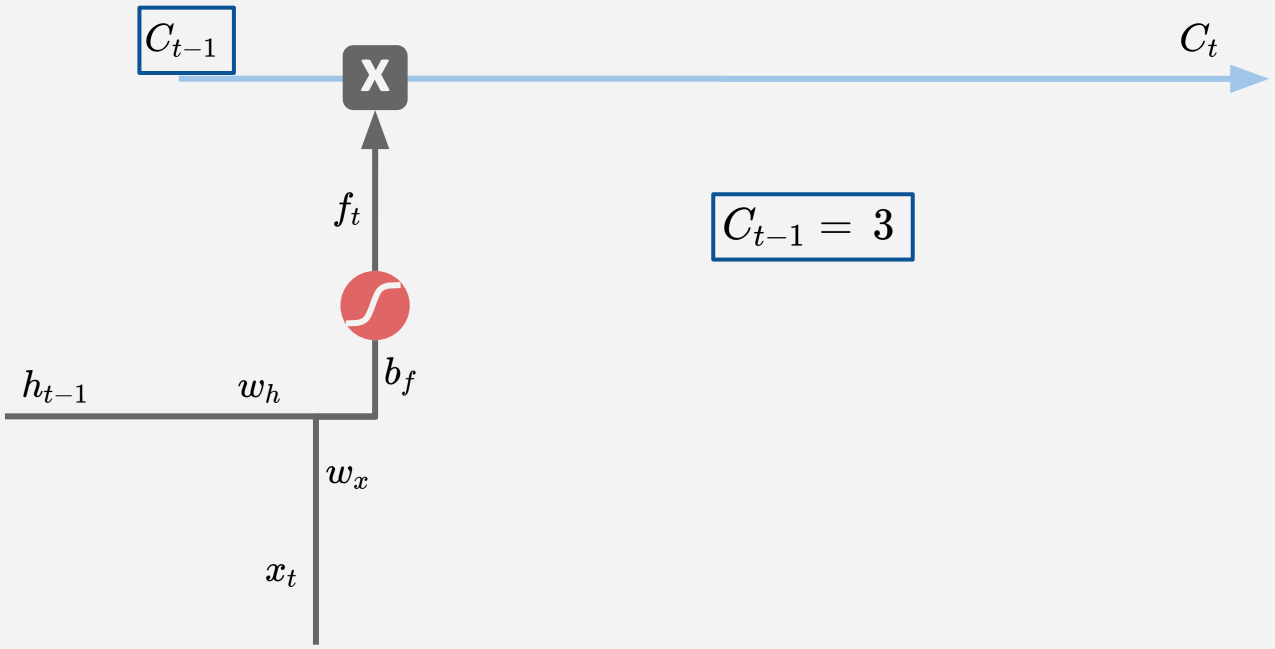
$w_h = 2.7$

$x_t = 1$

$w_x = 1.7$

$b_f = 1.6$

Forget Gate: Forward Pass



$h_{t-1} = 1$

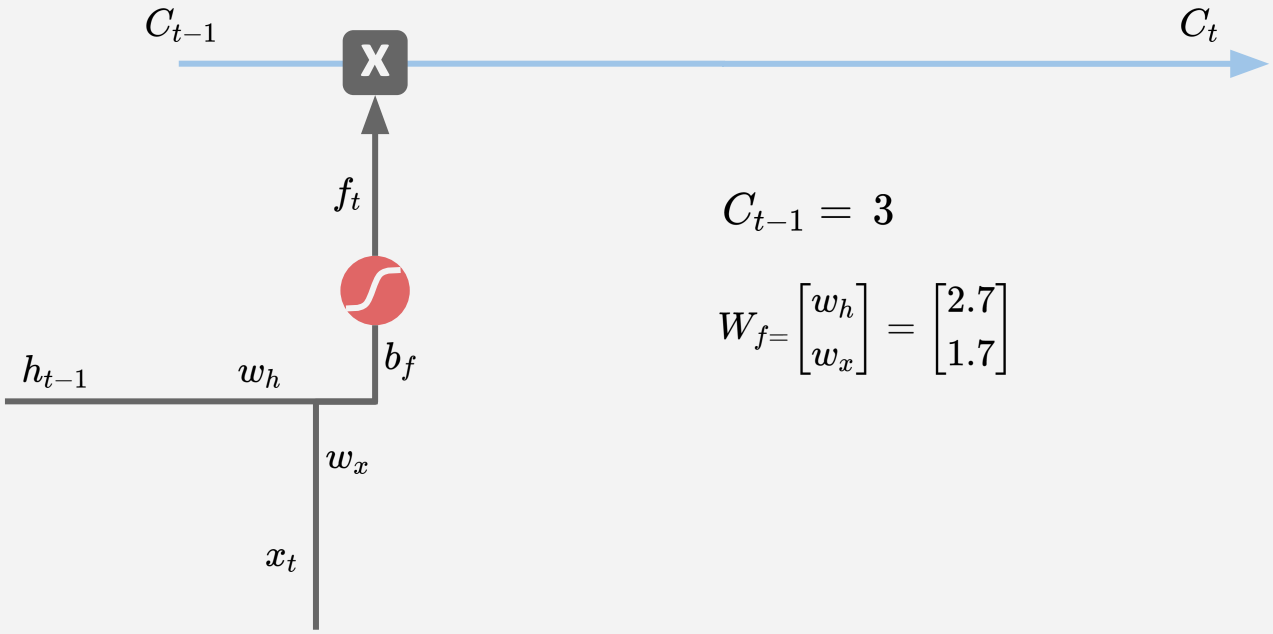
$w_h = 2.7$

$x_t = 1$

$w_x = 1.7$

$b_f = 1.6$

Forget Gate: Forward Pass



$C_{t-1} = 3$

$W_f = \begin{bmatrix} w_h \\ w_x \end{bmatrix} = \begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix}$

$h_{t-1} = 1$

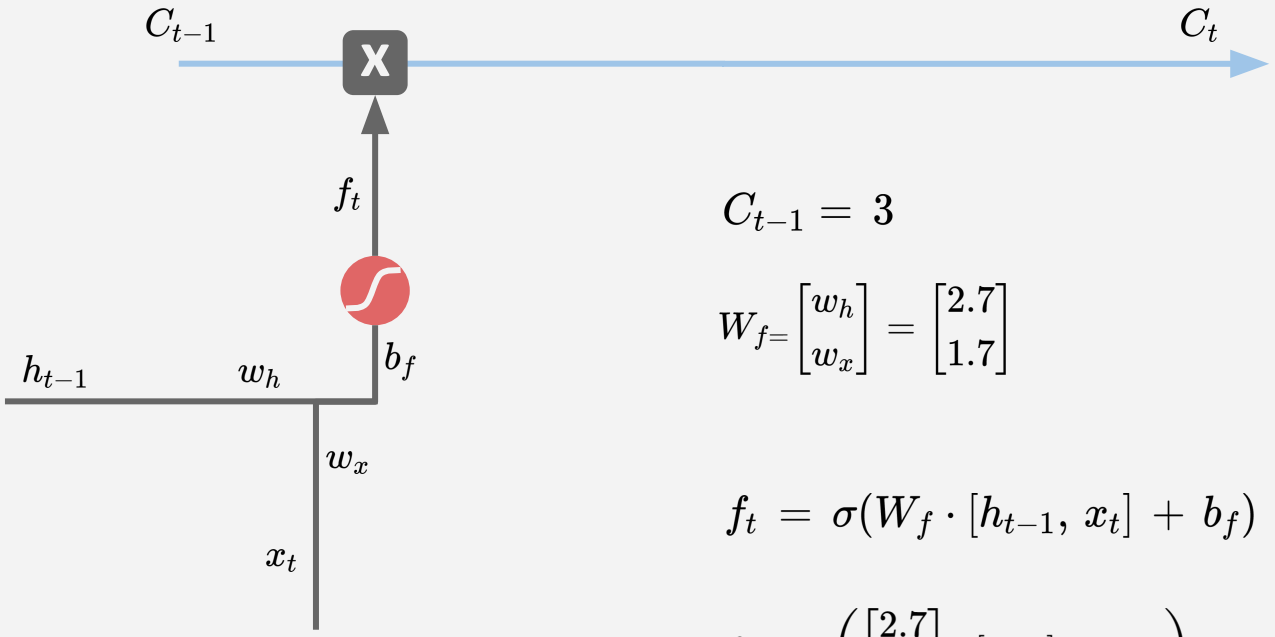
$w_h = 2.7$

$x_t = 1$

$w_x = 1.7$

$b_f = 1.6$

Forget Gate: Forward Pass



$$C_{t-1} = 3$$

$$W_f = \begin{bmatrix} w_h \\ w_x \end{bmatrix} = \begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix}$$

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

$$f_t = \sigma\left(\begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix} \cdot [1, 1] + 1.6\right) = 0.997$$

$$h_{t-1} = 1$$

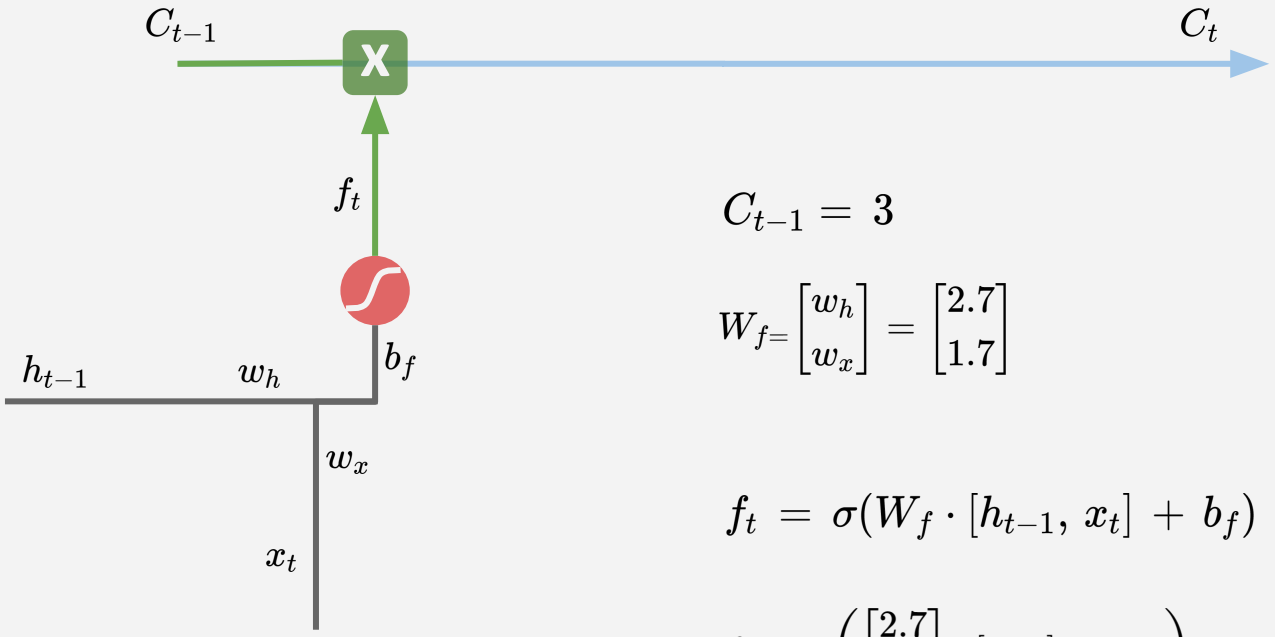
$$w_h = 2.7$$

$$x_t = 1$$

$$w_x = 1.7$$

$$b_f = 1.6$$

Forget Gate: Forward Pass



$$C_{t-1} = 3$$

$$W_f = \begin{bmatrix} w_h \\ w_x \end{bmatrix} = \begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix}$$

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

$$f_t = \sigma\left(\begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix} \cdot [1, 1] + 1.6\right) = 0.997$$

$$h_{t-1} = 1$$

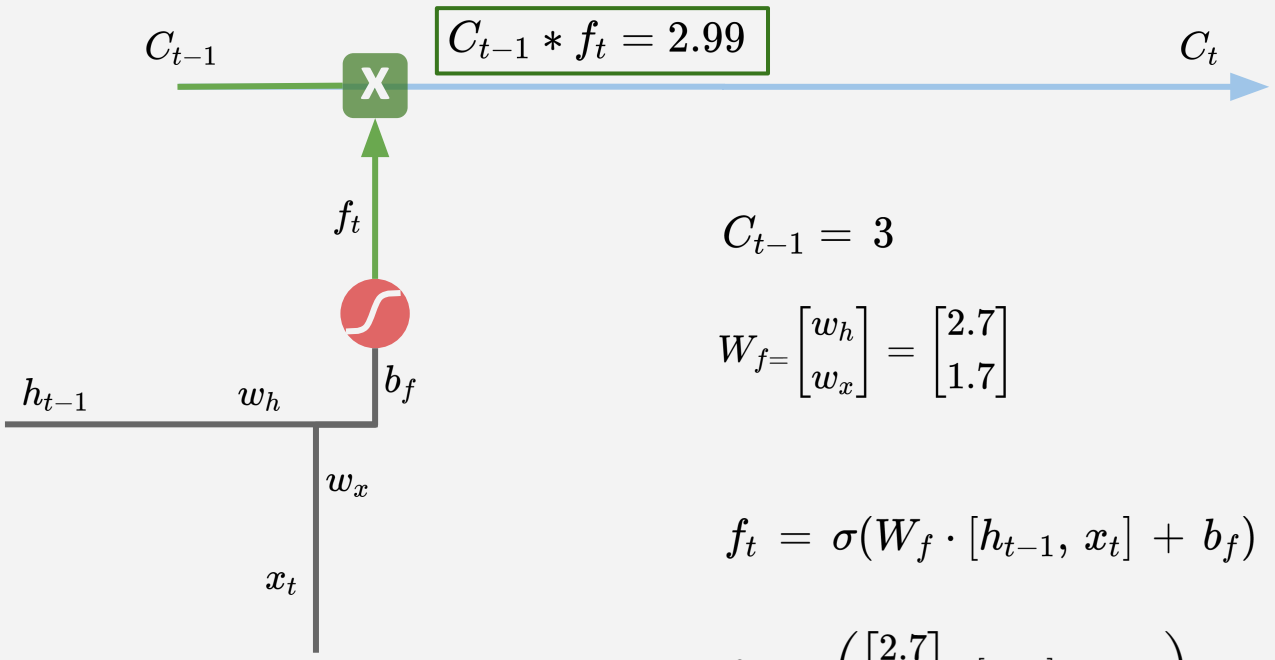
$$w_h = 2.7$$

$$x_t = 1$$

$$w_x = 1.7$$

$$b_f = 1.6$$

Forget Gate: Forward Pass



$$C_{t-1} = 3$$

$$W_f = \begin{bmatrix} w_h \\ w_x \end{bmatrix} = \begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix}$$

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

$$f_t = \sigma\left(\begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix} \cdot [1, 1] + 1.6\right) = 0.997$$

$$h_{t-1} = 1$$

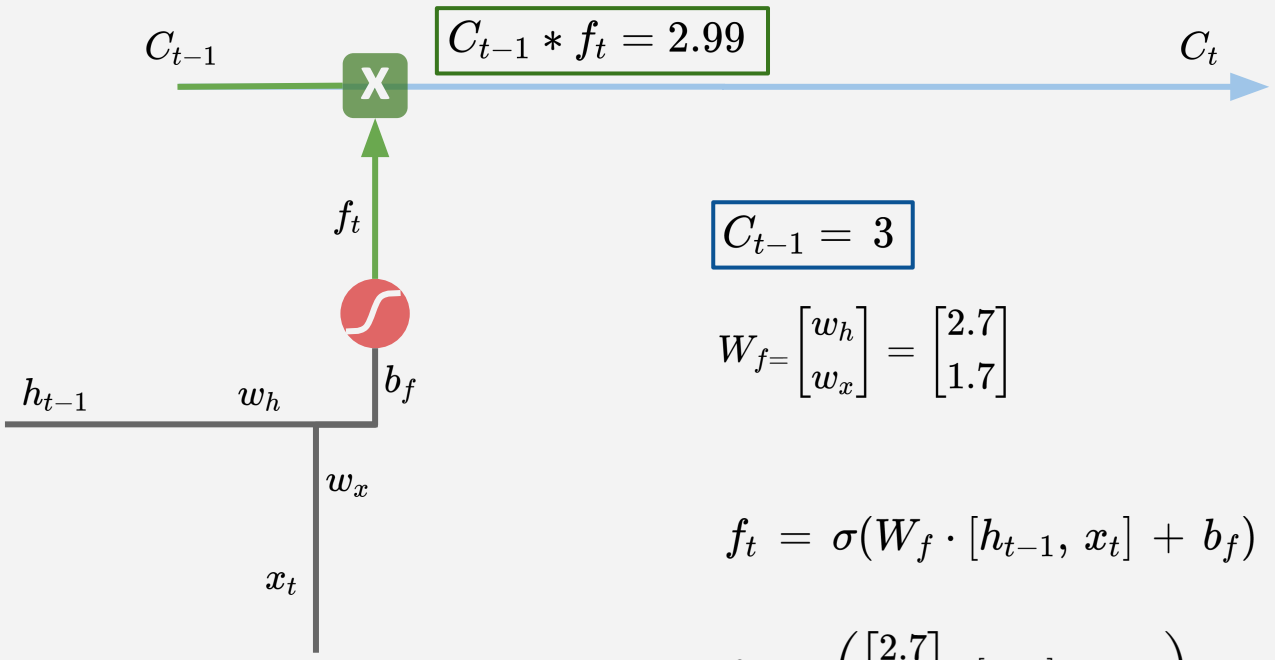
$$w_h = 2.7$$

$$x_t = 1$$

$$w_x = 1.7$$

$$b_f = 1.6$$

Forget Gate: Forward Pass



$C_{t-1} = 3$

$$W_f = \begin{bmatrix} w_h \\ w_x \end{bmatrix} = \begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix}$$

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

$$f_t = \sigma\left(\begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix} \cdot [1, 1] + 1.6\right) = 0.997$$

$h_{t-1} = 1$

$w_h = 2.7$

$w_x = 1.7$

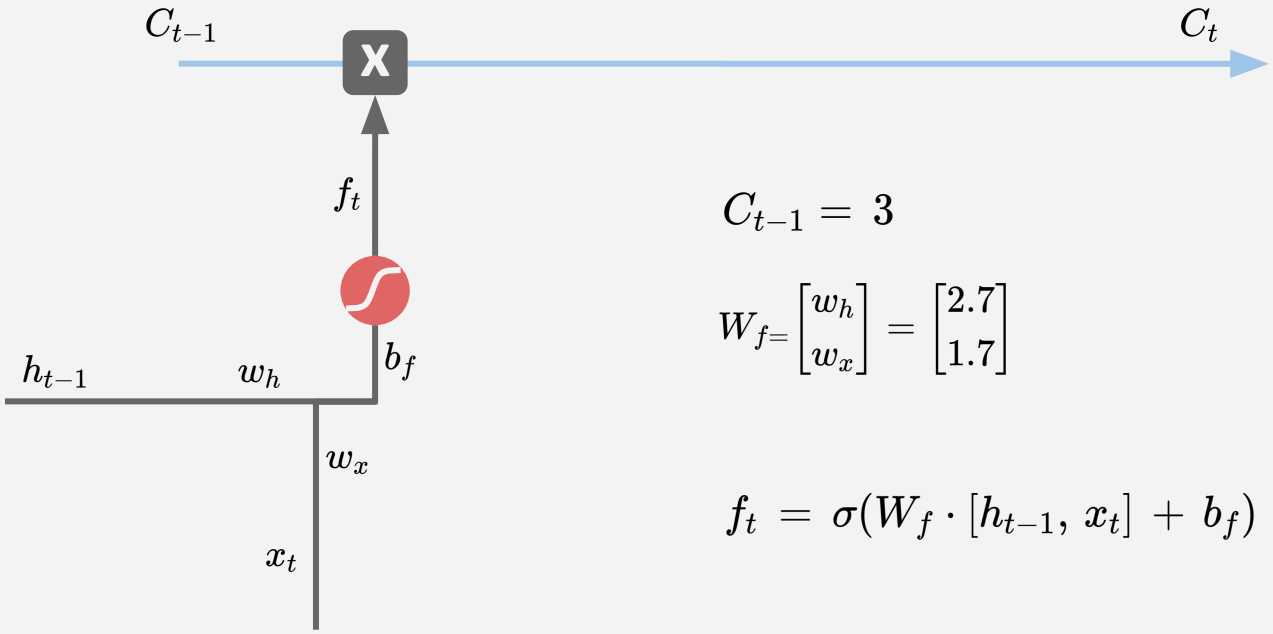
$b_f = 1.6$

The cell state (long-term memory) of the cell reduced slightly from 3.0 to 2.99 for a small input $x_t = 1$

Forget Gate: Forward Pass

Contrast this with a Drastic Change in the Input

Forget Gate: Forward Pass



$C_{t-1} = 3$

$W_f = \begin{bmatrix} w_h \\ w_x \end{bmatrix} = \begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix}$

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

$h_{t-1} = 1$

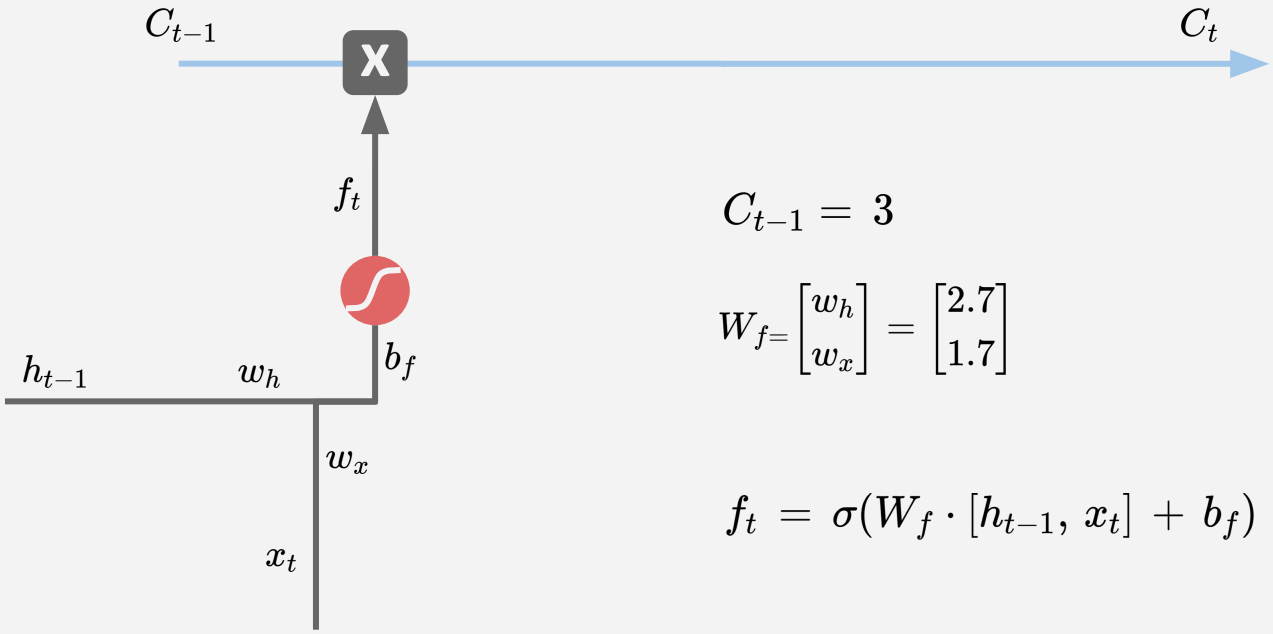
$w_h = 2.7$

$x_t = 1$

$w_x = 1.7$

$b_f = 1.6$

Forget Gate: Forward Pass



$C_{t-1} = 3$

$W_f = \begin{bmatrix} w_h \\ w_x \end{bmatrix} = \begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix}$

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

$h_{t-1} = 1$

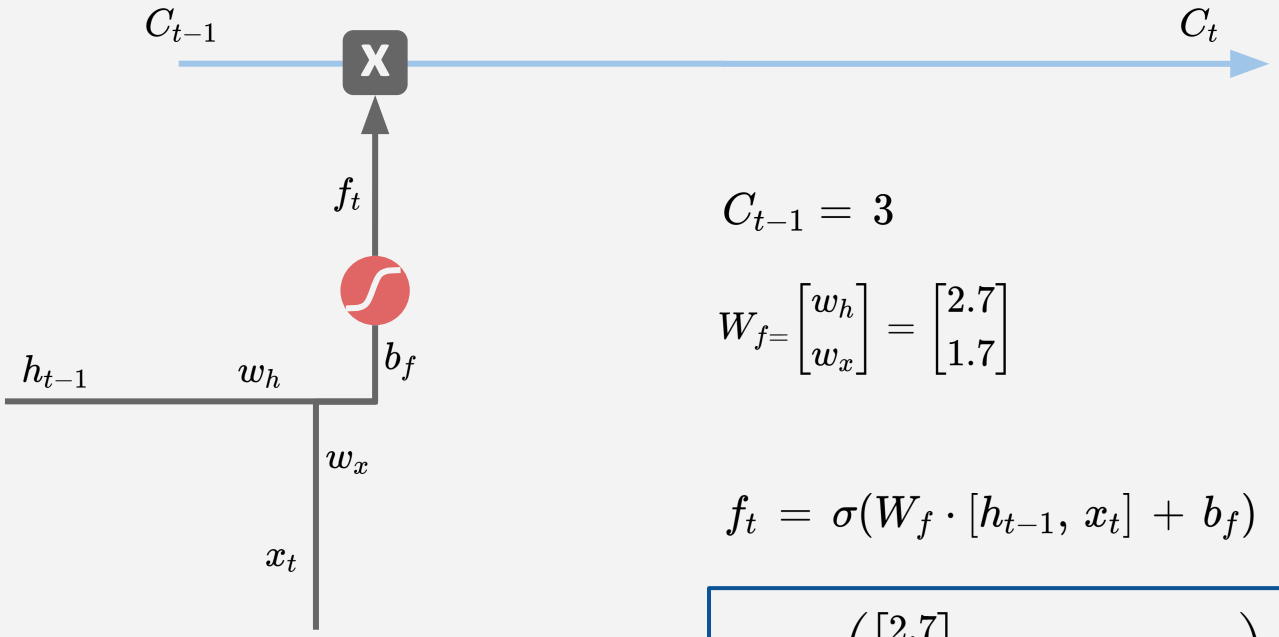
$w_h = 2.7$

$x_t = -10$

$w_x = 1.7$

$b_f = 1.6$

Forget Gate: Forward Pass



$C_{t-1} = 3$

$W_f = \begin{bmatrix} w_h \\ w_x \end{bmatrix} = \begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix}$

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

$$f_t = \sigma\left(\begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix} \cdot [1, -10] + 1.6\right) = 0.00$$

$h_{t-1} = 1$

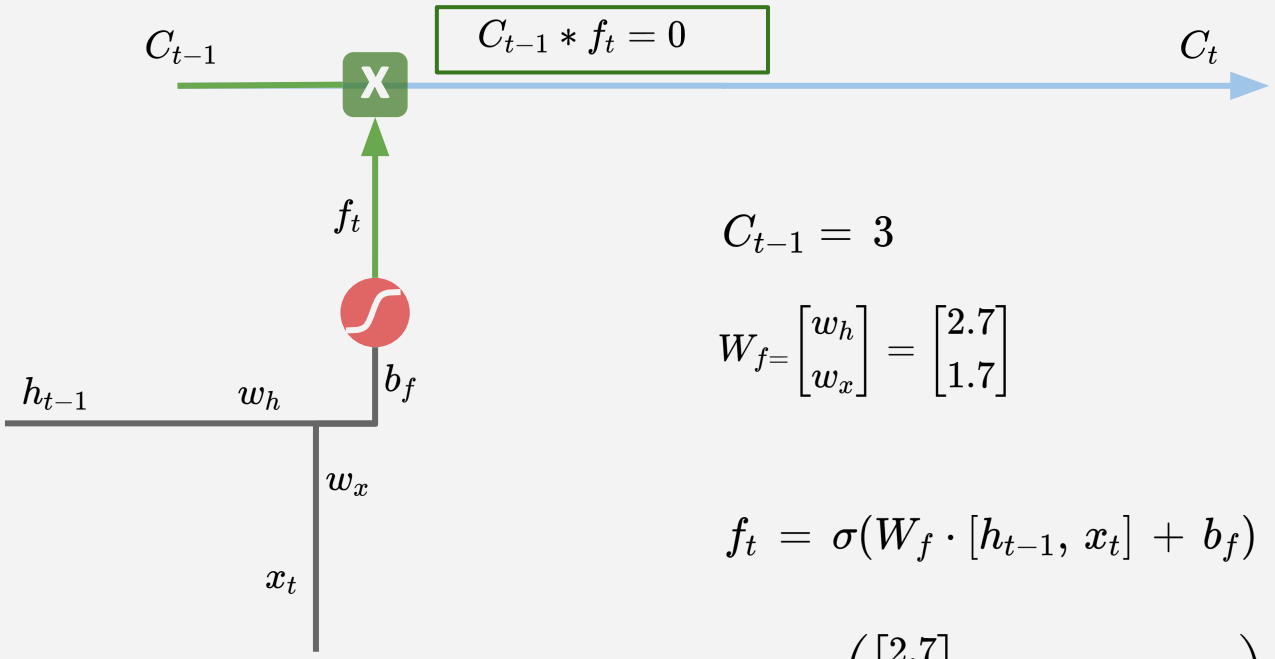
$w_h = 2.7$

$x_t = -10$

$w_x = 1.7$

$b_f = 1.6$

Forget Gate: Forward Pass



$$C_{t-1} = 3$$

$$W_f = \begin{bmatrix} w_h \\ w_x \end{bmatrix} = \begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix}$$

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

$$f_t = \sigma\left(\begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix} \cdot [1, -10] + 1.6\right) = 0.00$$

$$h_{t-1} = 1$$

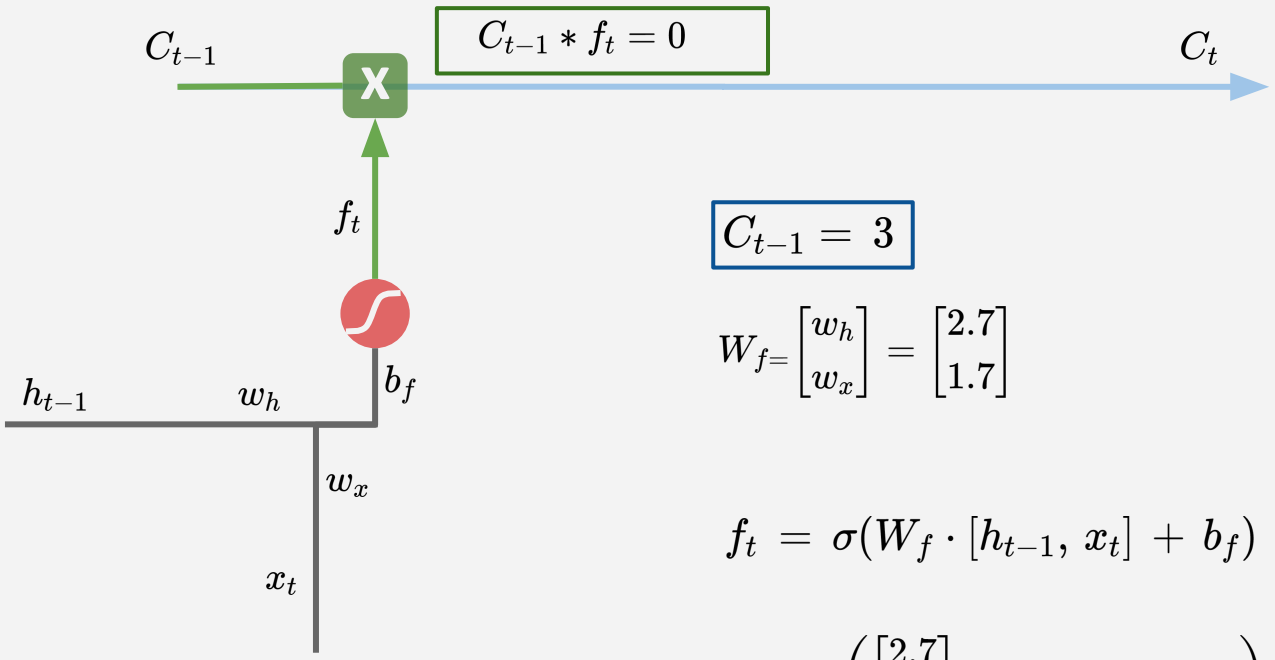
$$w_h = 2.7$$

$$x_t = -10$$

$$w_x = 1.7$$

$$b_f = 1.6$$

Forget Gate: Forward Pass



$C_{t-1} = 3$

$$W_f = \begin{bmatrix} w_h \\ w_x \end{bmatrix} = \begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix}$$

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

$$f_t = \sigma\left(\begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix} \cdot [1, -10] + 1.6\right) = 0.00$$

$h_{t-1} = 1$

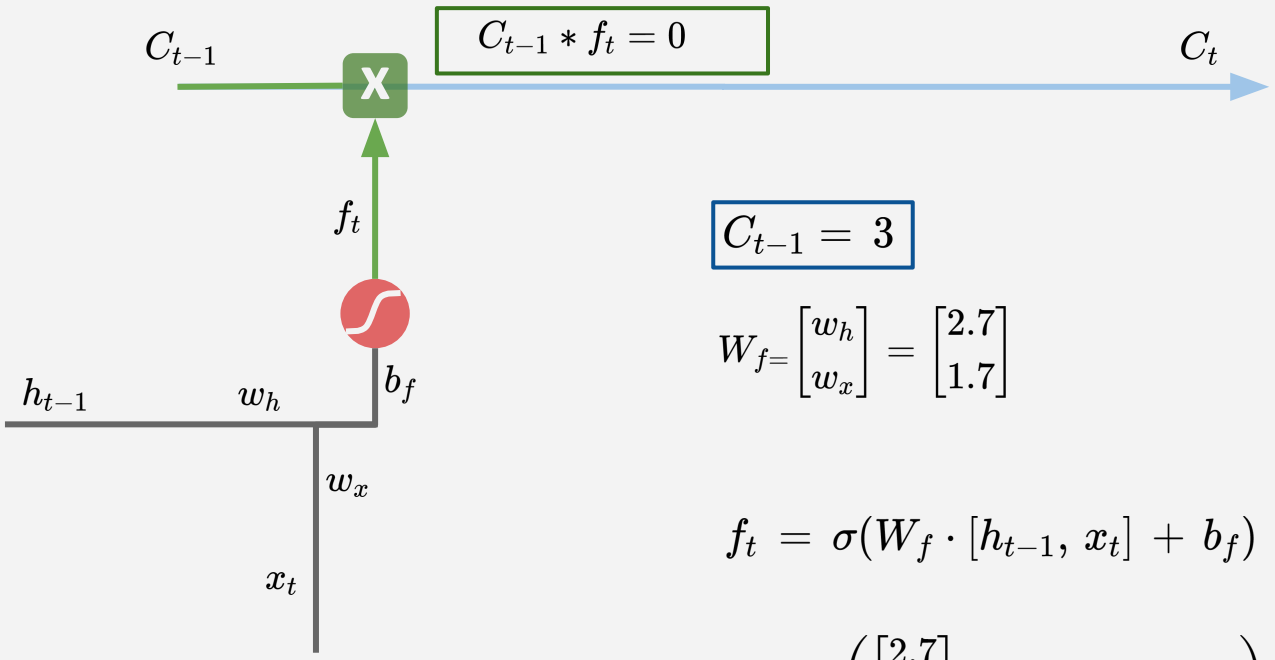
$w_h = 2.7$

$x_t = -10$

$w_x = 1.7$

$b_f = 1.6$

Forget Gate: Forward Pass



$$C_{t-1} = 3$$

$$W_f = \begin{bmatrix} w_h \\ w_x \end{bmatrix} = \begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix}$$

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

$$f_t = \sigma\left(\begin{bmatrix} 2.7 \\ 1.7 \end{bmatrix} \cdot [1, -10] + 1.6\right) = 0.00$$

$h_{t-1} = 1$

$w_h = 2.7$

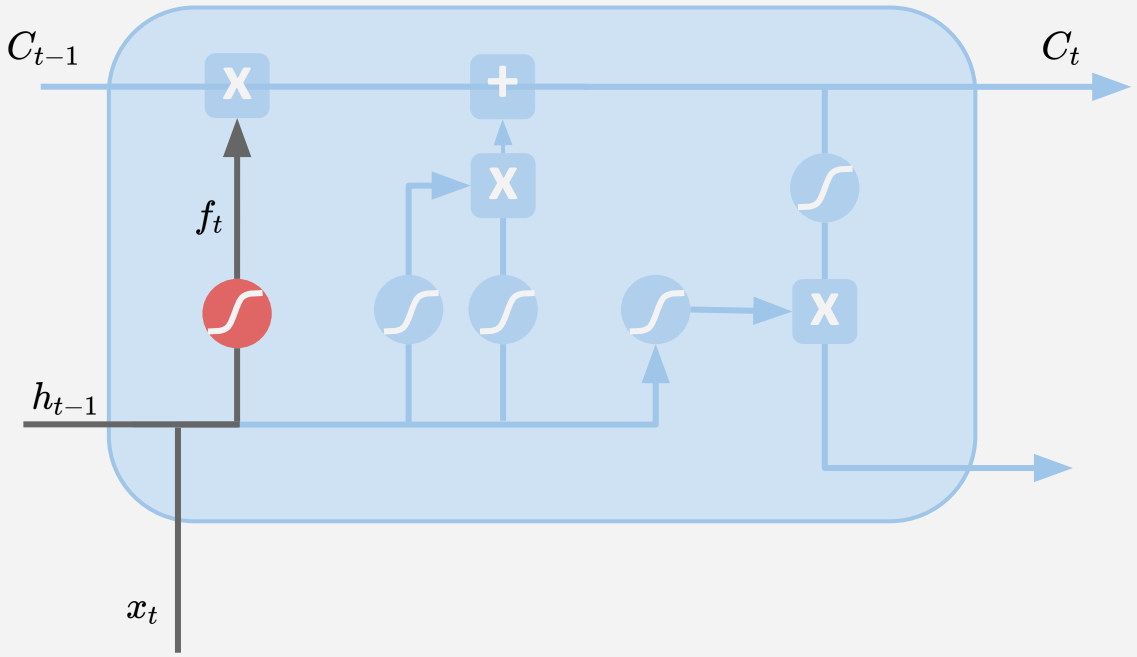
$x_t = -10$

$w_x = 1.7$

$b_f = 1.6$

The cell state (long-term memory) is completely forgotten for a large change in the input (change of context)

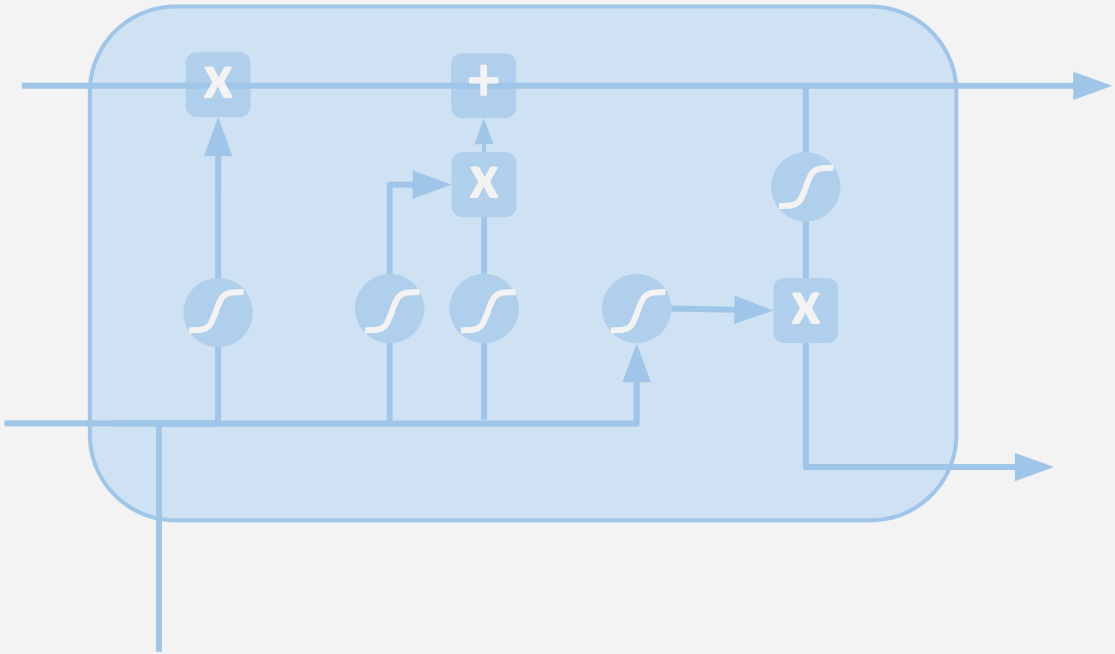
LSTM Cell: Forget Gate



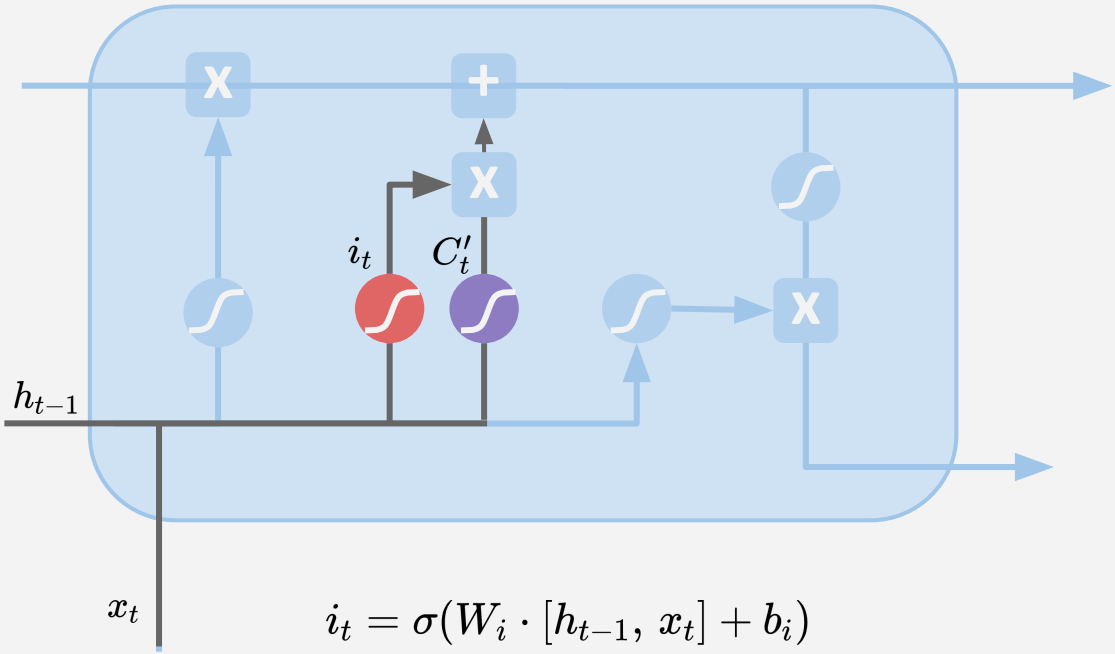
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Forget Gate: Determines what percent of the long-term memory (cell state) is remembered

LSTM Cell: Input Gate



LSTM Cell: Input Gate



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

$$C'_t = \tanh\left(W_C \cdot [h_{t-1},\, x_t] + b_C\right)$$

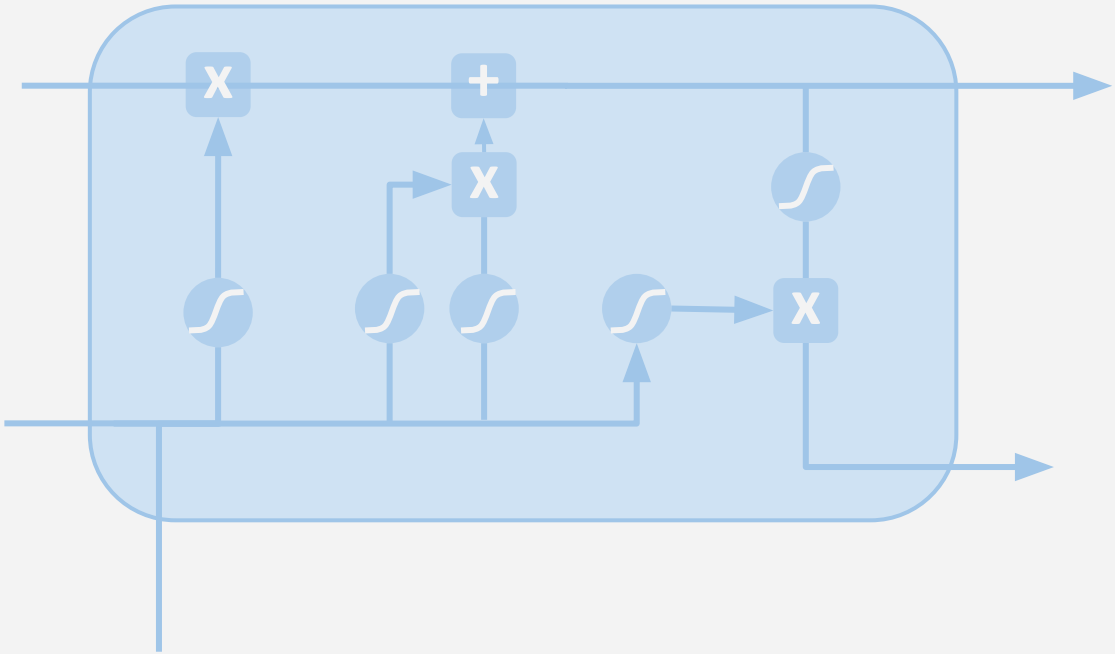
Input Gate: Decide what **new information to store** in the cell state.

C'_t dictates the new information to accumulate from the previous hidden state and input.

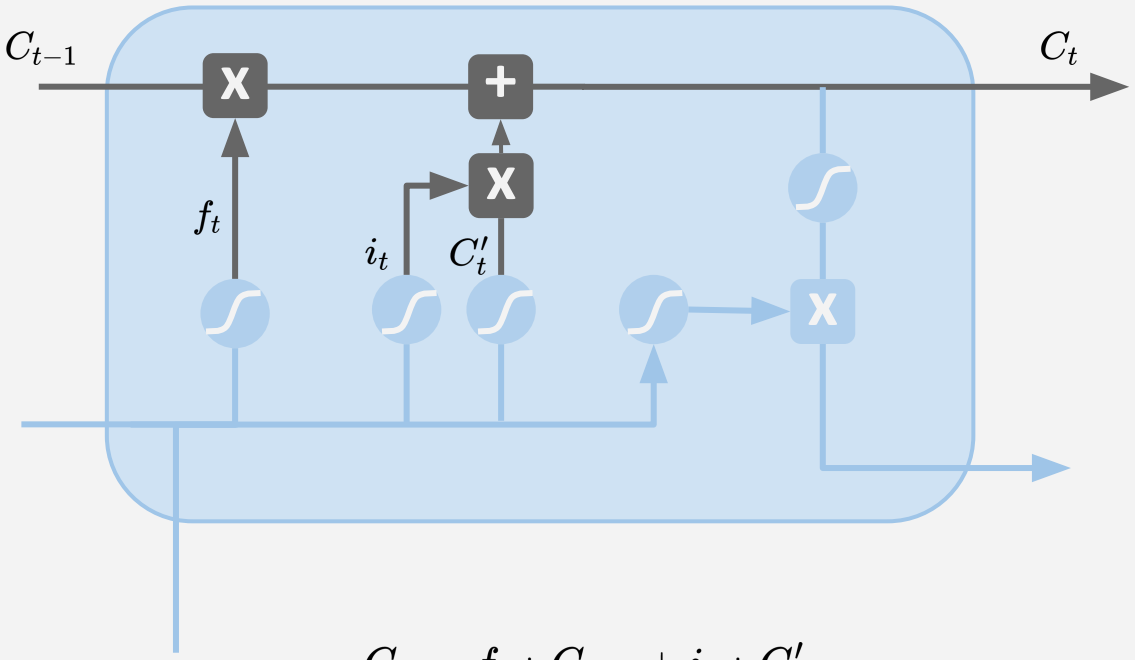
The input gate layer i_t dictates how much percentage (notice the sigmoid activation) of C'_t to propagate.

Oregon State University

LSTM Cell: Cell State Update



LSTM Cell: Cell State Update

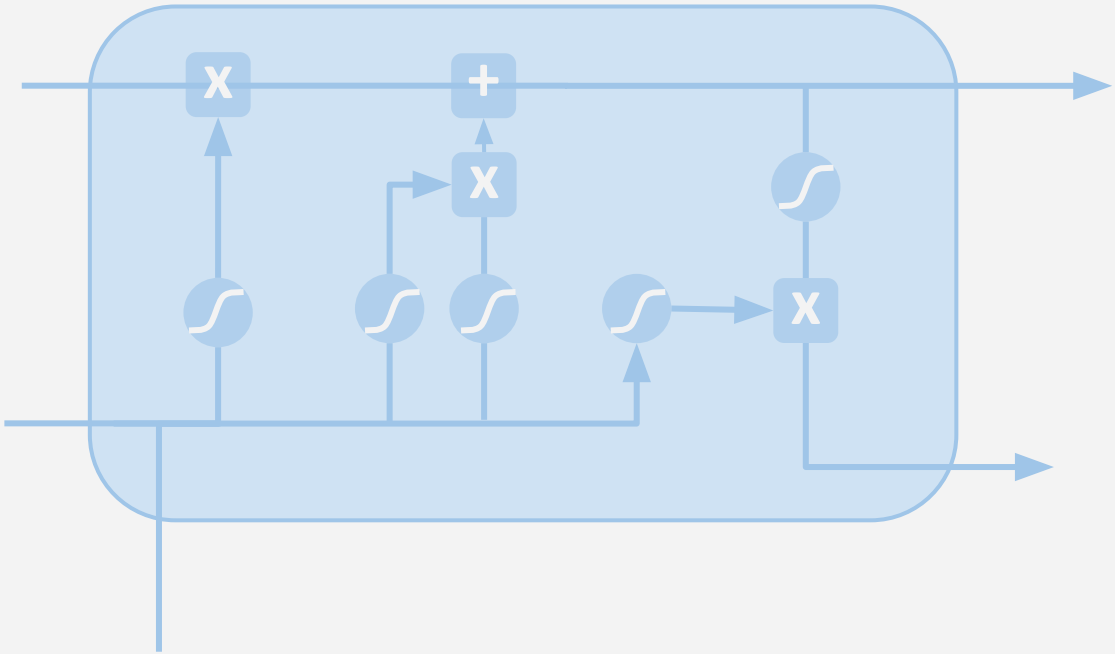


$$C_t = f_t * C_{t-1} + i_t * C'_t$$

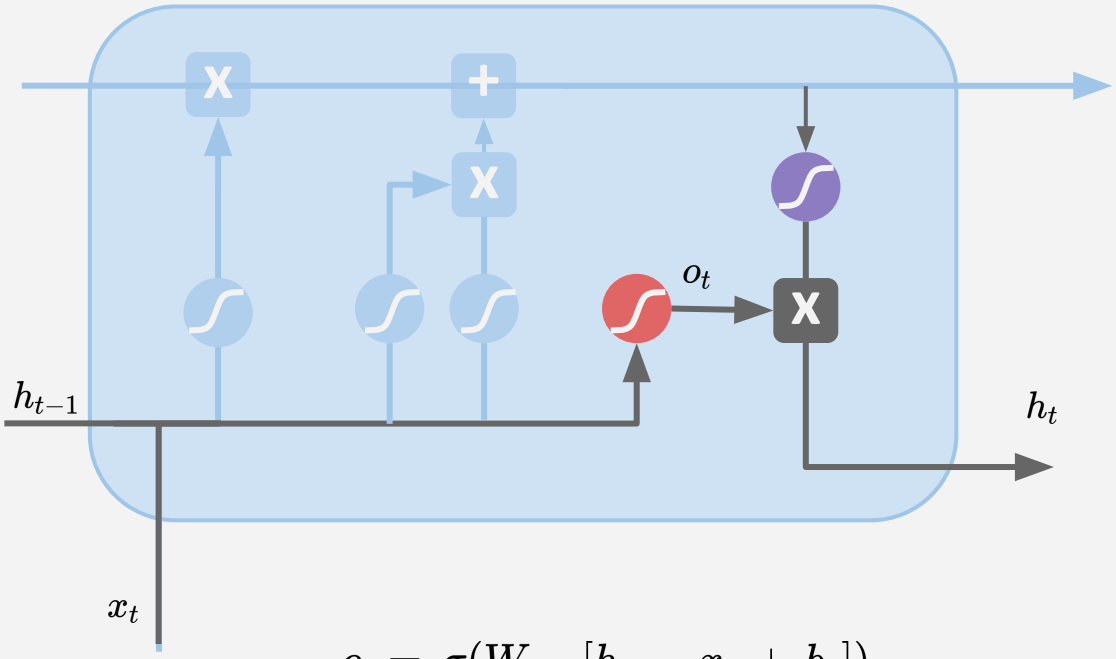
Cell State Update: Incorporate information that has passed through the forget and the input gate to create new cell state C_t

The updated cell state C_t, is a filtered version of the previous cell state, that has potentially incorporated new information relevant to the current input and previous hidden state

LSTM Cell: Output Gate



LSTM Cell: Output Gate



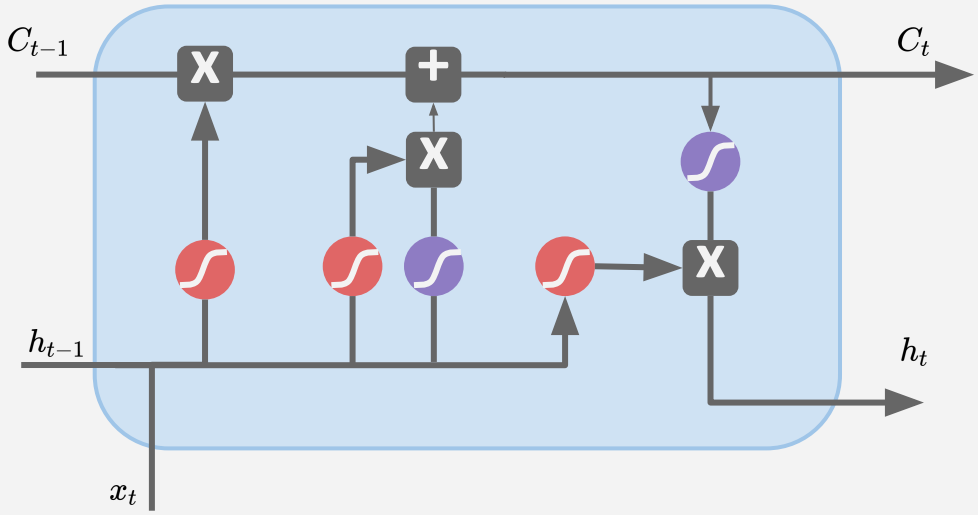
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t + b_o])$$

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

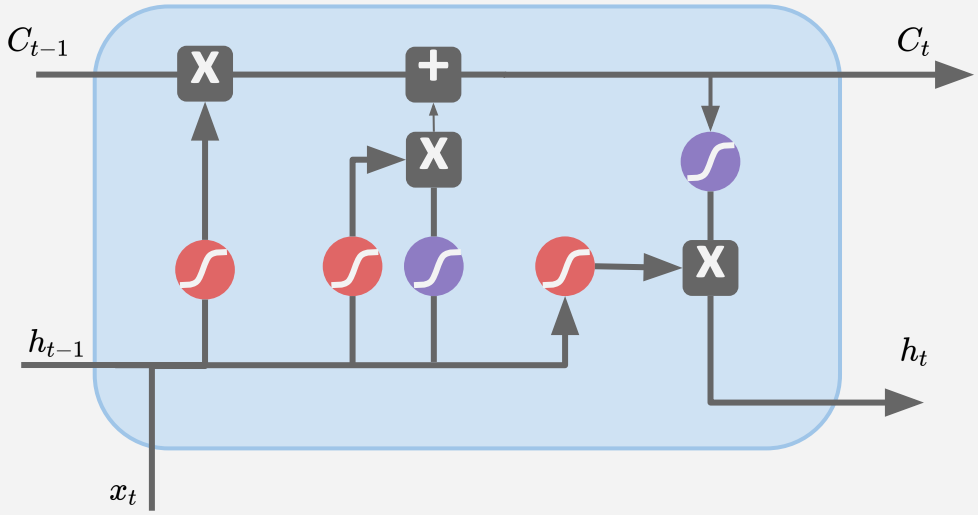
Output Gate: The output is a filtered version of the updated cell state C_t.

The output gate propagates a percentage (notice the sigmoid activation for o_t) of the current input and previous hidden state, along with the updated cell state C_t.

Review: LSTMs



Review: LSTMs



1. Maintains a persistent cell state - robust to gradient updates

2. Gates to regulate information
   a. **Forget gate** to remove unnecessary information from the cell state
   b. **Input gate** to selectively use the current input and hidden state information
   c. **Output gate** to propagate the modified cell state to the next cell

3. Gates are parameterized

Backpropagation through time will update gradients to **control gates** for **long-term persistence**

LSTM Cell