Recurrent Neural Networks

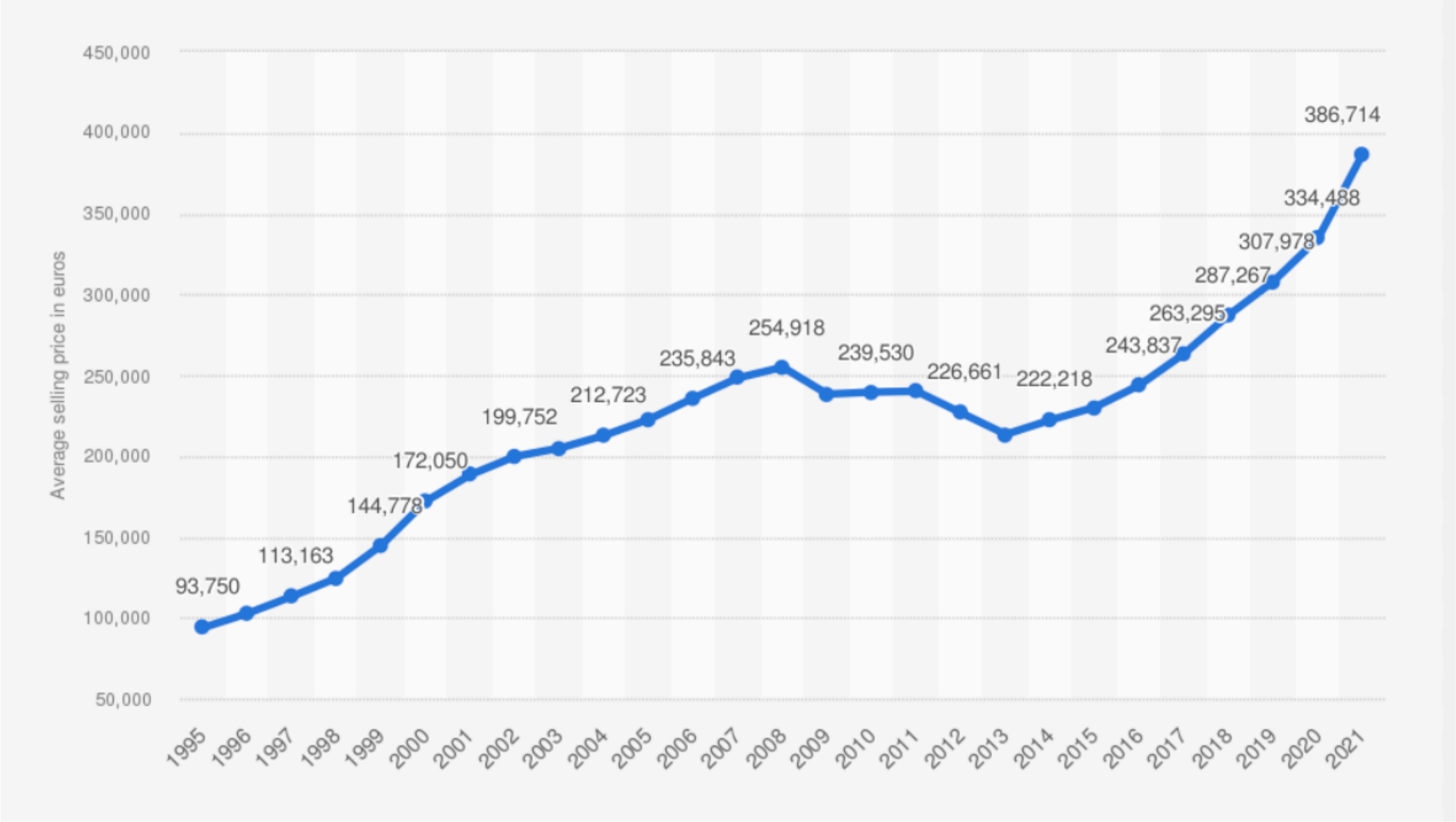
Elena Congeduti, 4-12-2024

TU Delft

Lecture’s Agenda

- Modeling sequences
- Memory and Recurrent Neural Networks (RNNs)
- Gradient in RNNs
- Long Short Term Memory (LSTM)

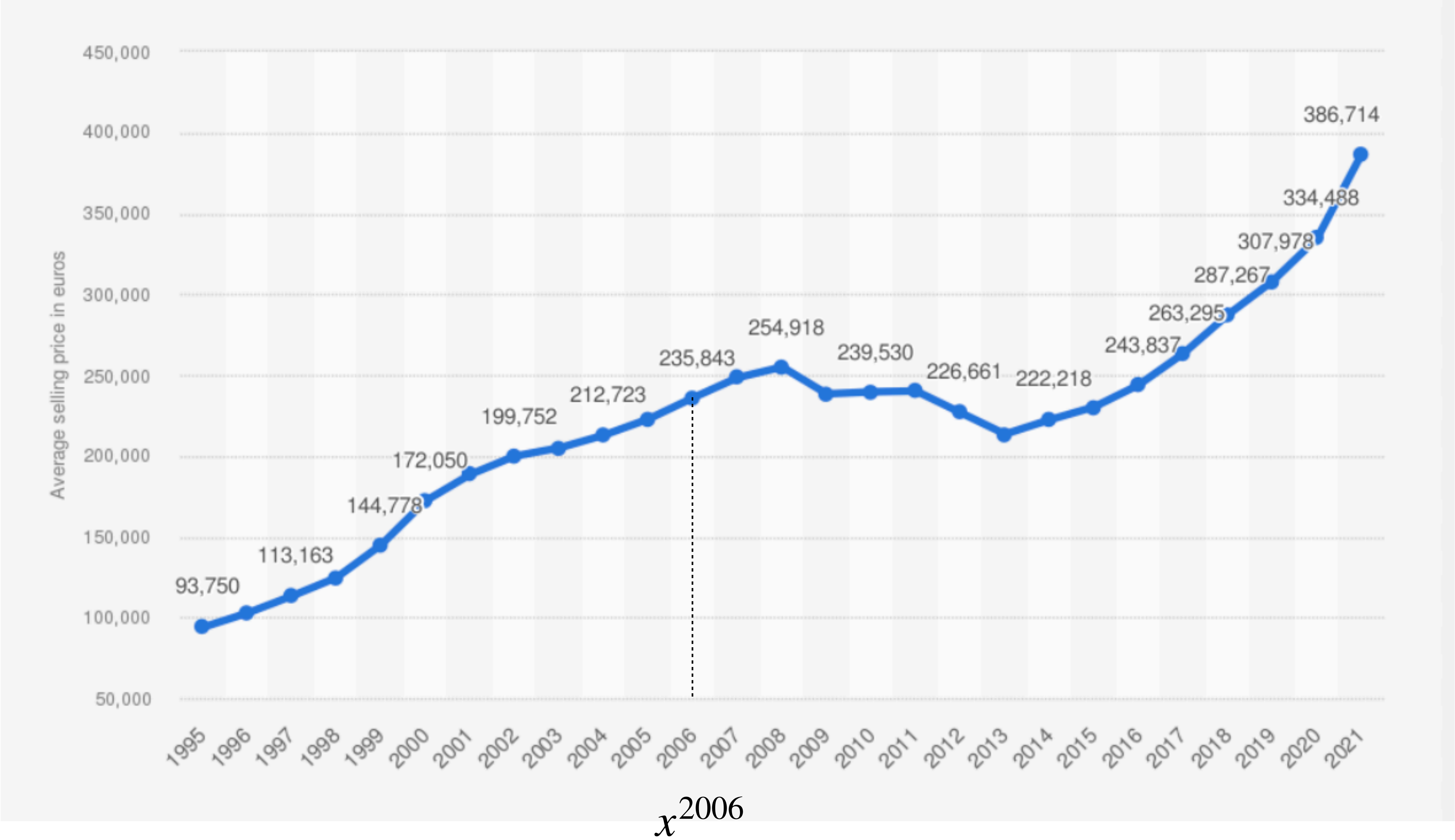
Sequential data



Observation x^t each time step t

$$\ldots, x^{2005}, x^{2006}, x^{2007}, \ldots, x^{2010}, \ldots$$

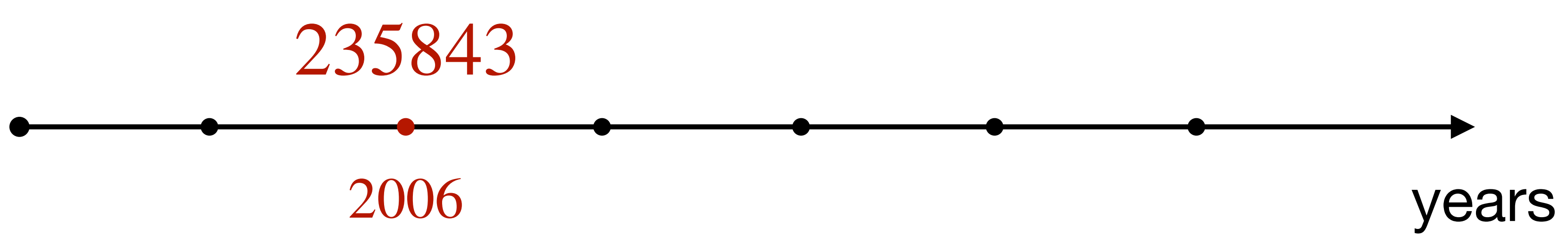
Sequential data



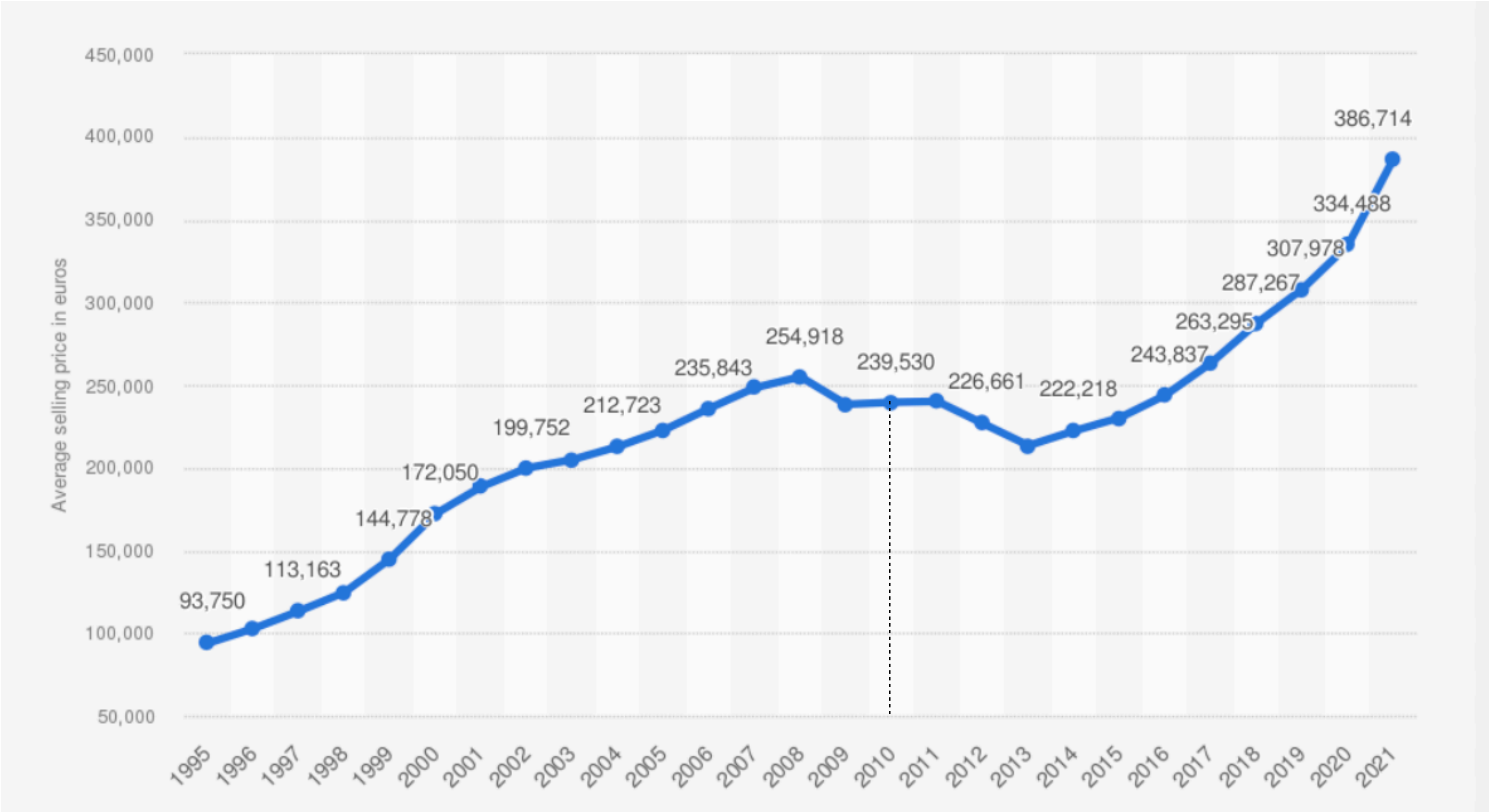
x^{2006}

Observation x^t each time step t

$$\ldots, x^{2005}, x^{2006}, x^{2007}, \ldots, x^{2010}, \ldots$$



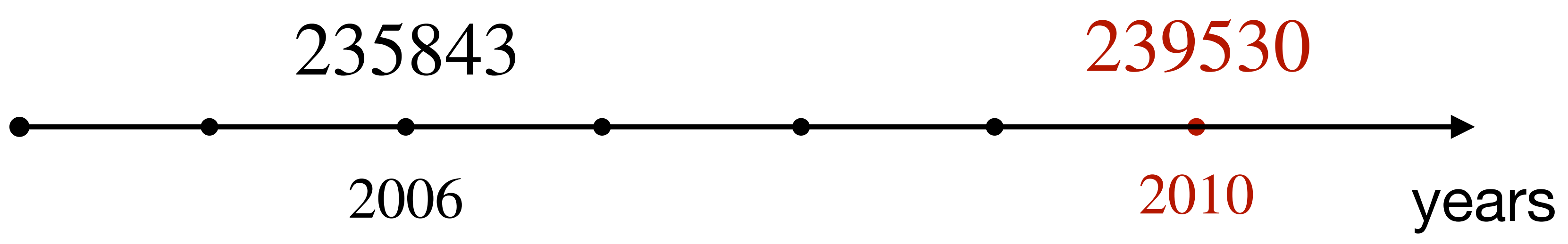
Sequential data



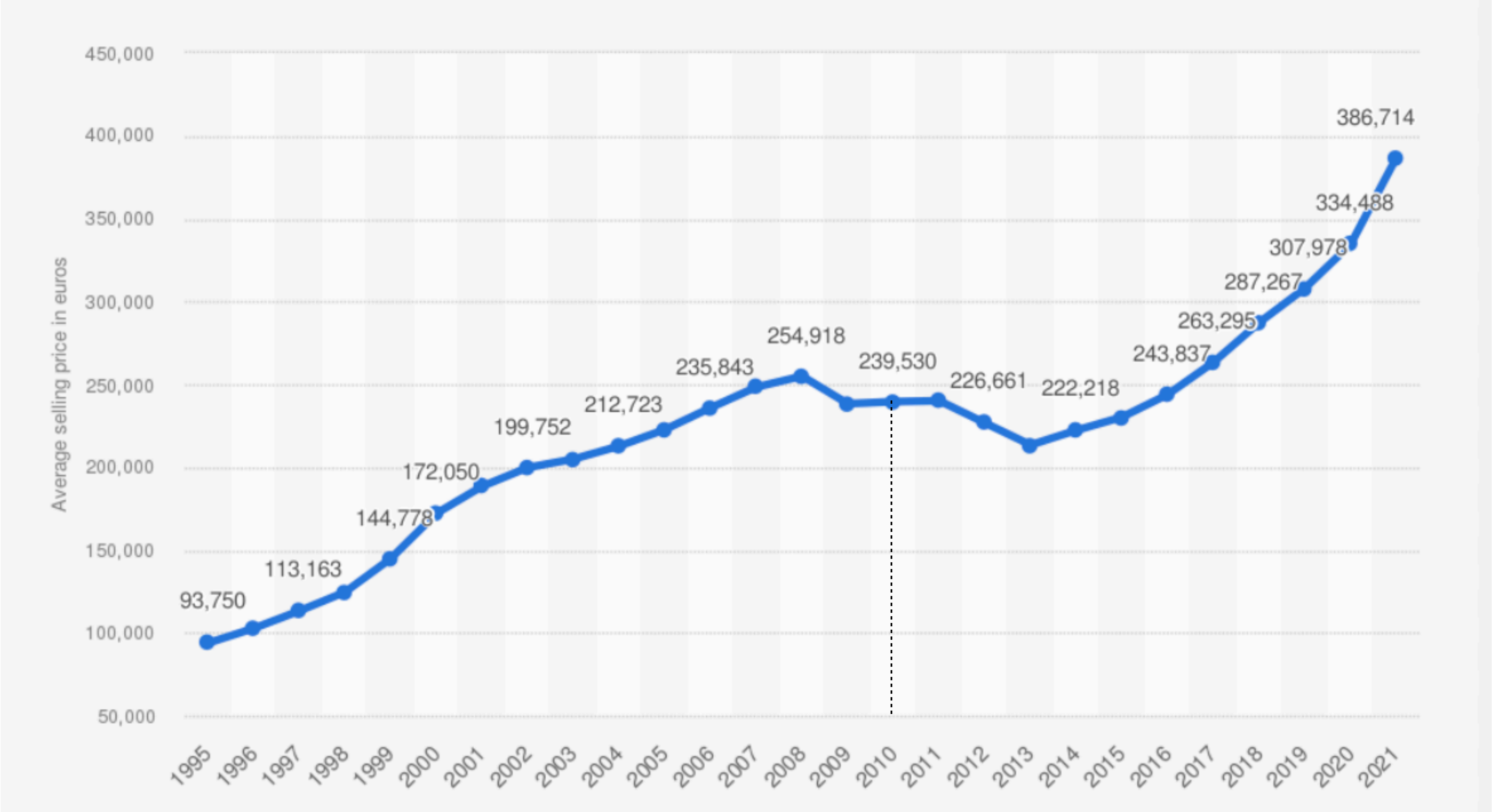
x^{2010}

Observation x^t each time step t

$$\ldots, x^{2005}, x^{2006}, x^{2007}, \ldots, x^{2010}, \ldots$$



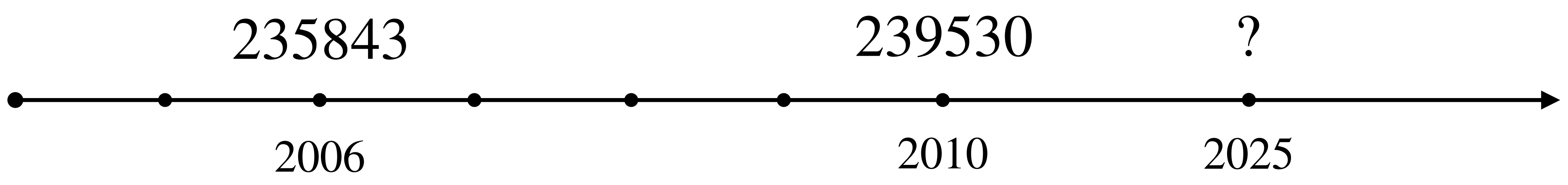
Sequential data



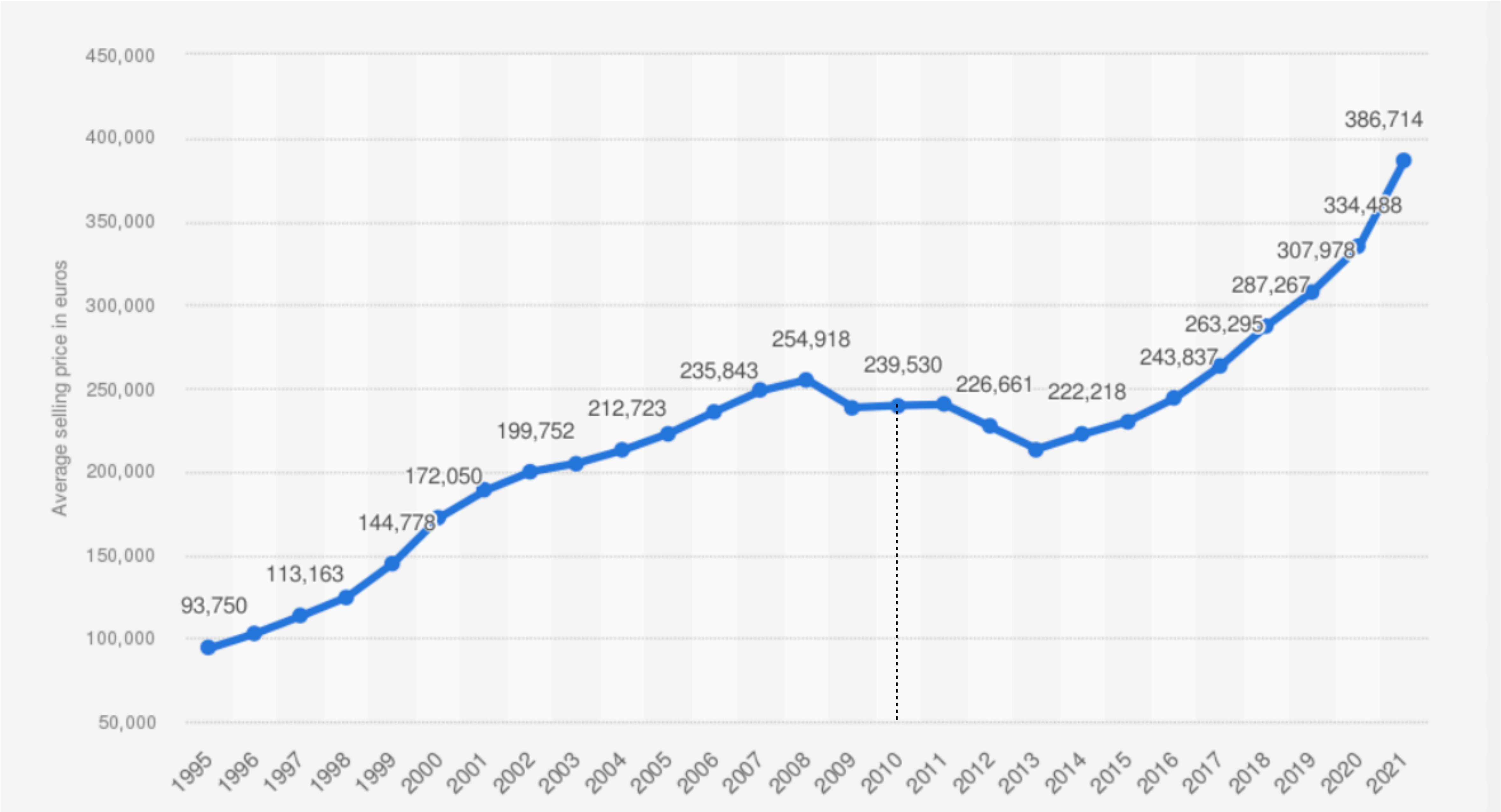
x^{2010}

Observation x^t each time step t

$$\ldots, x^{2005}, x^{2006}, x^{2007}, \ldots, x^{2010}, \ldots$$



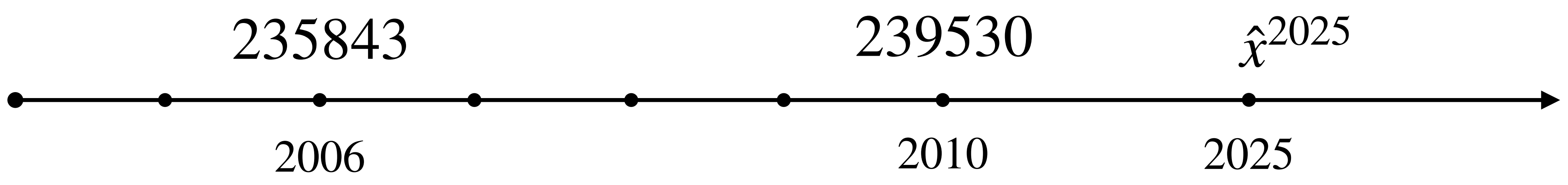
Sequential data



x^{2010}

Observation x^t each time step t

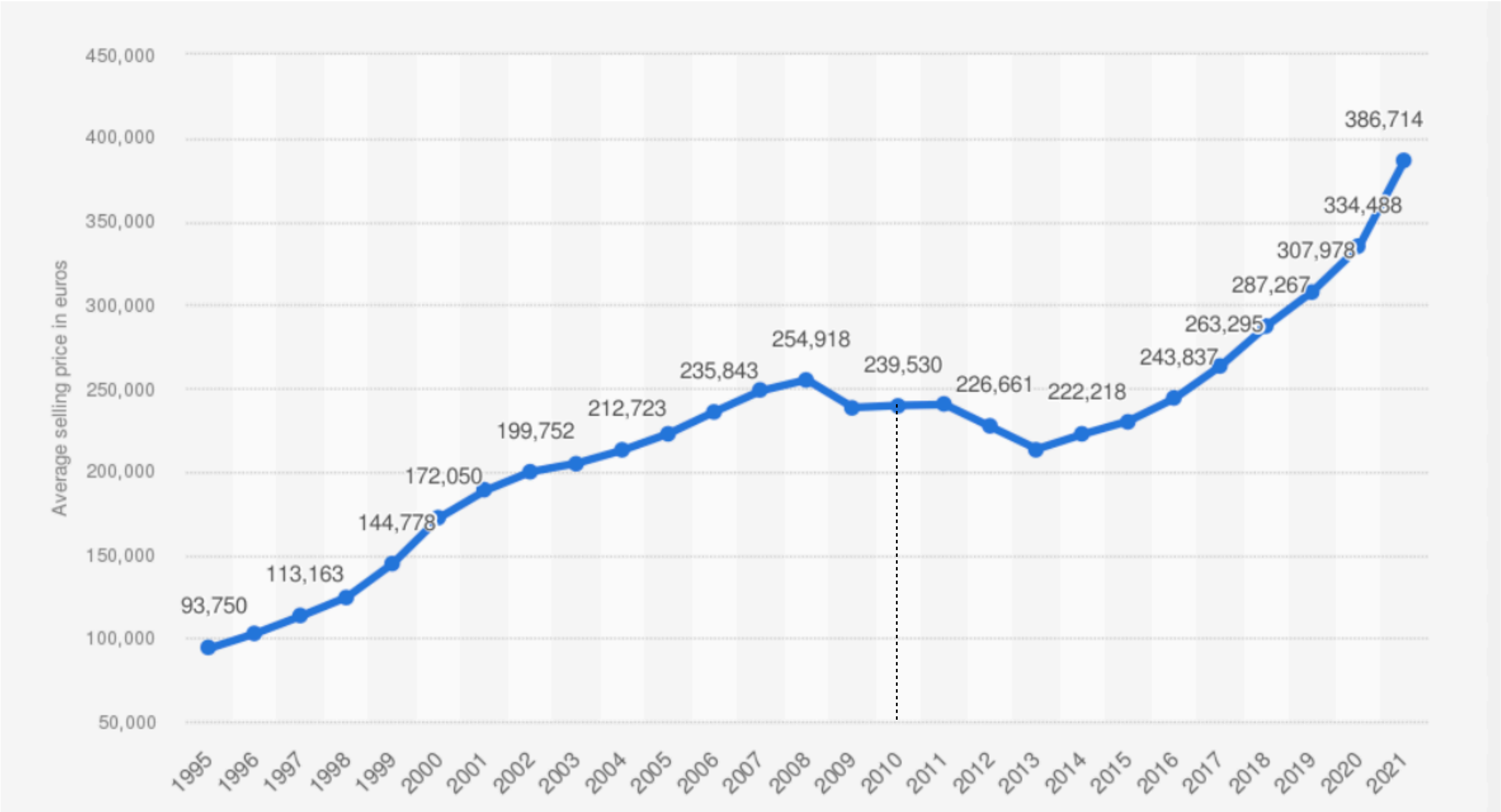
$$\ldots, x^{2005}, x^{2006}, x^{2007}, \ldots, x^{2010}, \ldots$$



Input output

$$x^{1995}, \ldots, x^{2024} \longrightarrow \hat{x}^{2025}$$

Sequential data



x^{2010}

Observation x^t each time step t

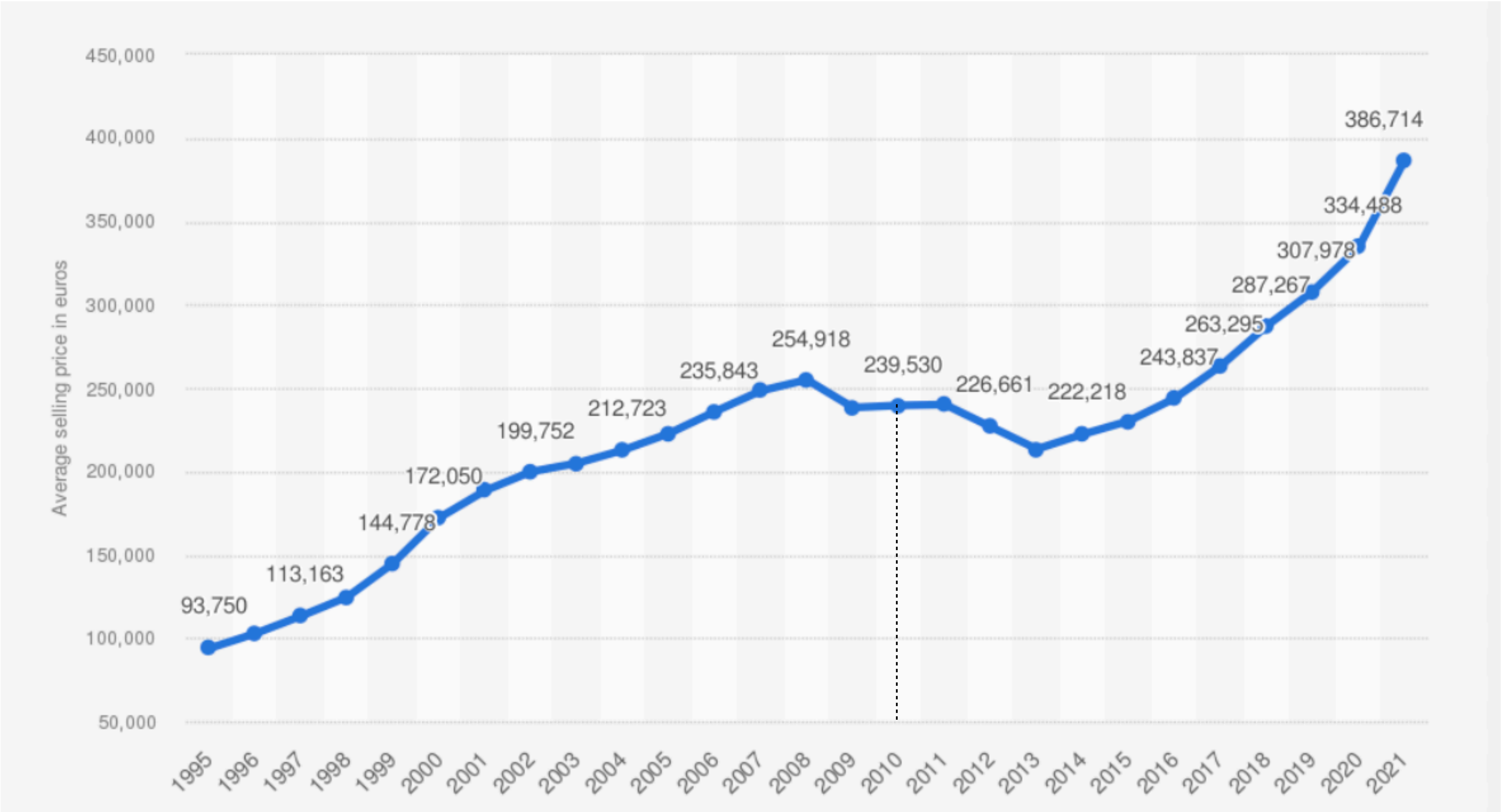
$$\ldots, x^{2005}, x^{2006}, x^{2007}, \ldots, x^{2010}, \ldots$$

235843   239530 x^{2025} ?

2006   2010   2025   2026

Input		output
$x^{1995}, \ldots, x^{2024}$	$\longrightarrow$	$\hat{x}^{2025}$
$x^{1995}, \ldots, x^{2024}, x^{2025}$	$\longrightarrow$	$\hat{x}^{2026}$

Sequential data



x^{2010}

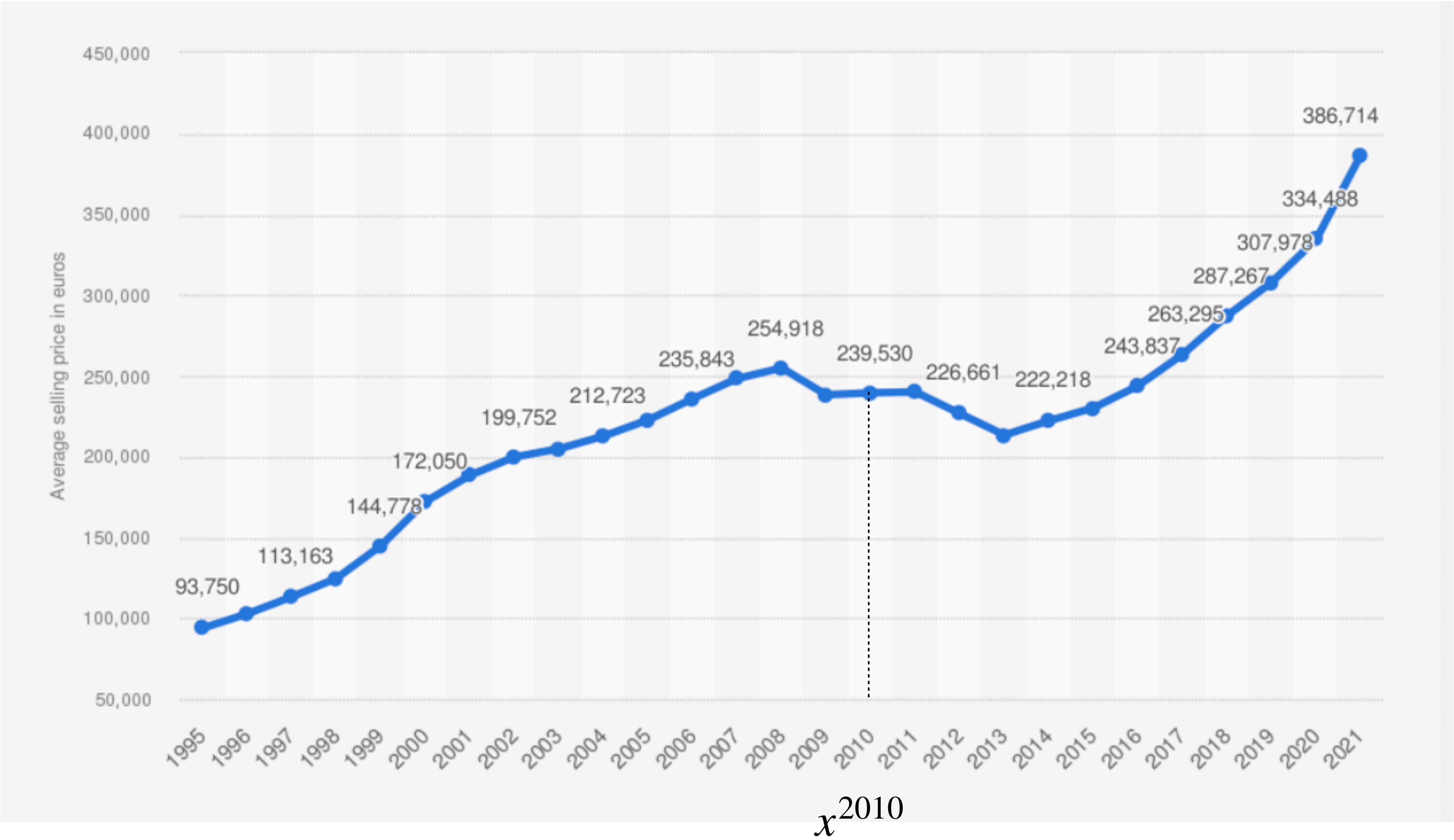
Observation x^t each time step t

$$\dots, x^{2005}, x^{2006}, x^{2007}, \dots, x^{2010}, \dots$$

Predict the next observation x^{t+1} given the past ones

Input	output
$x^1, \dots, x^t$ $\longrightarrow$	x^{t+1}

Sequential data



x^{2010}

Observation x^t each time step t

$$\ldots, x^{2005}, x^{2006}, x^{2007}, \ldots, x^{2010}, \ldots$$

Predict the next observation x^{t+1} given the past ones

Input		output
$x^1, \ldots, x^t$	$\longrightarrow$	x^{t+1}

Other examples of data with sequence structure?

Modeling Sequences

Examples

- Time series forecasting: predict x^{t+1} given $x^{t}, x^{t-1}, \dots$

- Extract the year in which I went to Rome from "In **2022**, I went to Rome"

- Determine the parity of a binary input sequence, whether the number of 1's is odd:

Input		Output
$0\,1\,0\,0\,1\,1\,0$	$\longrightarrow$	1
$0\,0\,1\,1$	$\longrightarrow$	0

Modeling Sequences

Examples

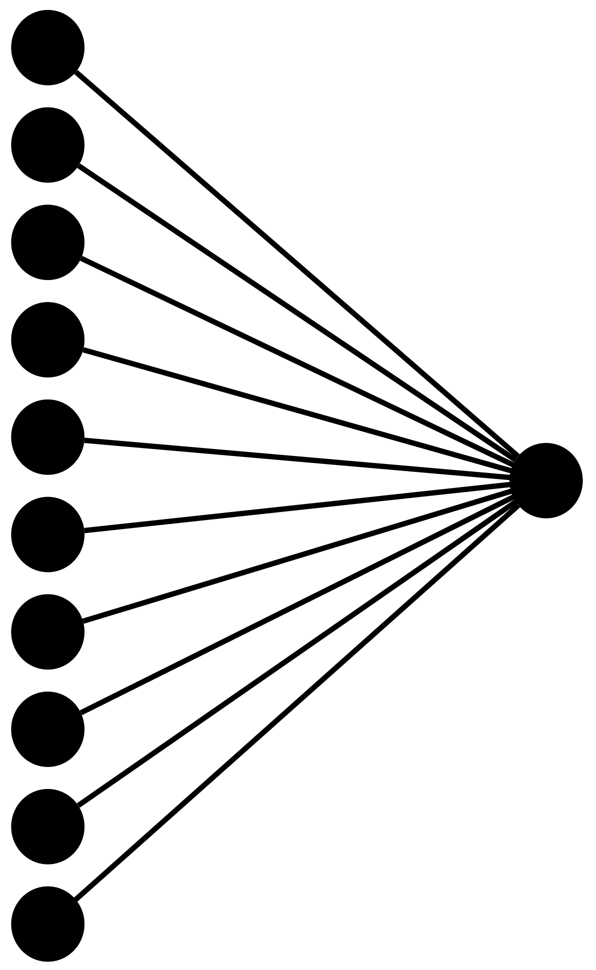
- Time series forecasting: predict x^{t+1} given $x^t, x^{t-1}, \dots$

- Extract the year in which I went to Rome from “In **2022**, I went to Rome”

- Determine the parity of a binary input sequence, whether the number of 1’s is odd:

Input	Output
$0\,1\,0\,0\,1\,1\,0 \longrightarrow$	1
$0\,0\,1\,1 \longrightarrow$	0

Why not using feed forward network?



Modeling Sequences

Examples

- Time series forecasting: predict x^{t+1} given $x^t, x^{t-1}, \ldots$

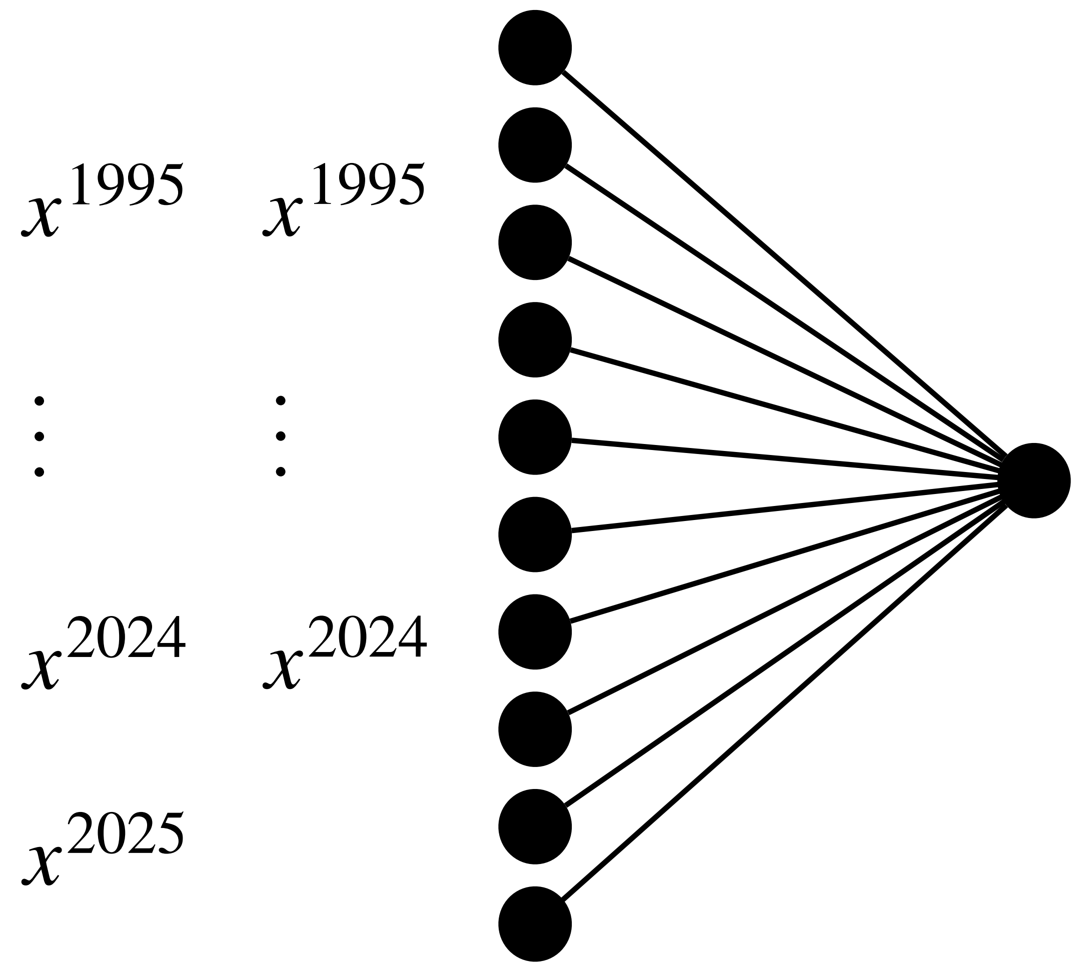
- Extract the year in which I went to Rome from "In **2022**, I went to Rome"

- Determine the parity of a binary input sequence, whether the number of 1's is odd:

Input	Output
$0\,1\,0\,0\,1\,1\,0 \longrightarrow$	1
$0\,0\,1\,1 \longrightarrow$	0

Why not using feed forward network?

- Variable length of sequences (potentially infinite)



How to input sequences with different length?

Modeling Sequences

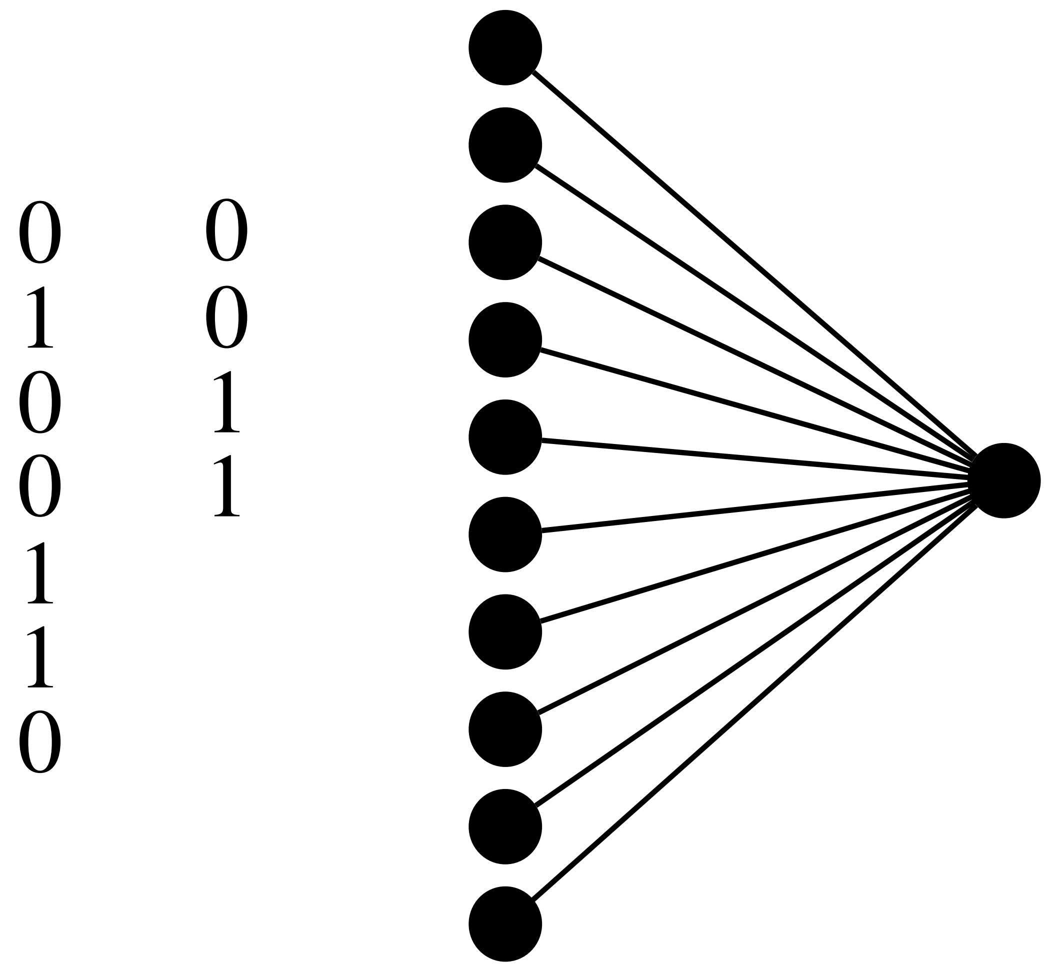
Examples

- Time series forecasting: predict x^{t+1} given $x^t, x^{t-1}, \ldots$
- Extract the year in which I went to Rome from "In **2022**, I went to Rome"
- Determine the parity of a binary input sequence, whether the number of 1's is odd:

Input	Output
$0\,1\,0\,0\,1\,1\,0 \longrightarrow$	1
$0\,0\,1\,1 \longrightarrow$	0

Why not using feed forward network?

- Variable length of sequences (potentially infinite)



How to input sequences with different length?

Modeling Sequences

Examples

- Time series forecasting: predict x^{t+1} given $x^t, x^{t-1}, \ldots$
- Extract the year in which I went to Rome from “In **2022**, I went to Rome”
- Determine the parity of a binary input sequence, whether the number of 1’s is odd:

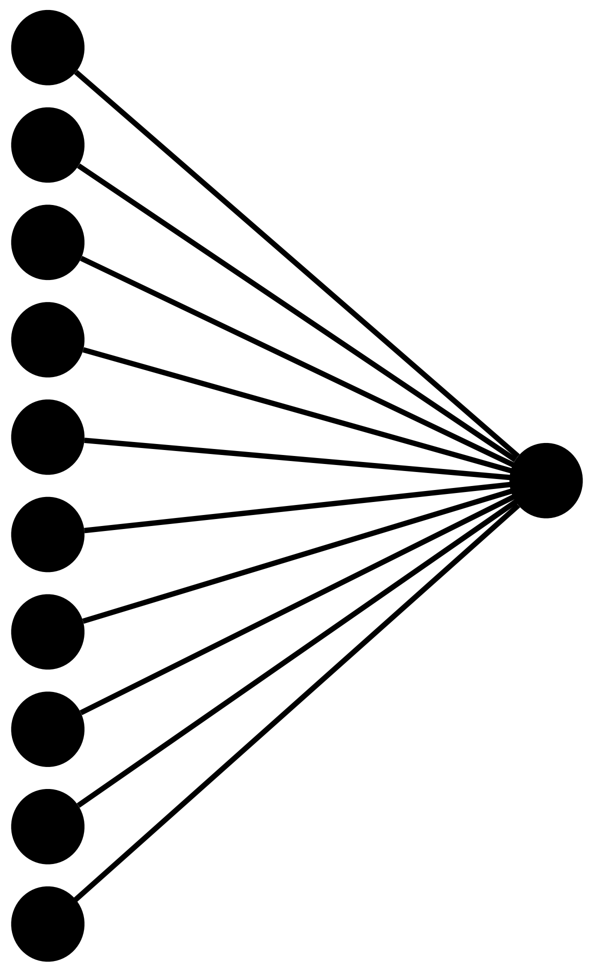
Input		Output
$0\,1\,0\,0\,1\,1\,0$	$\longrightarrow$	1
$0\,0\,1\,1$	$\longrightarrow$	0

Why not using feed forward network?

- Variable length of sequences (potentially infinite)

0	0
1	0
0	1
0	1
1	0
1	0
0	0

Zero padding



Modeling Sequences

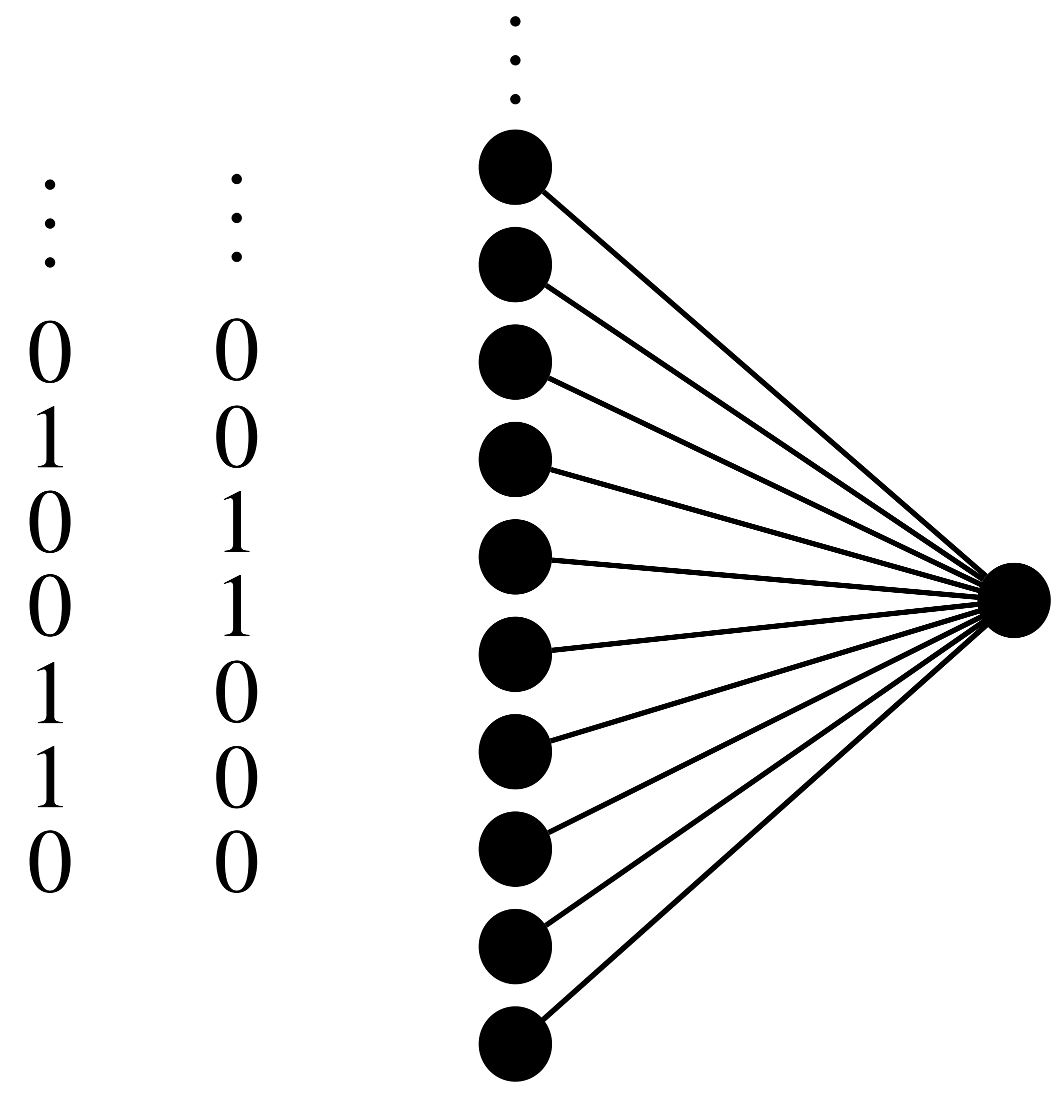
Examples

- Time series forecasting: predict x^{t+1} given $x^t, x^{t-1}, \ldots$
- Extract the year in which I went to Rome from “In **2022**, I went to Rome”
- Determine the parity of a binary input sequence, whether the number of 1’s is odd:

Input	Output
$0\,1\,0\,0\,1\,1\,0 \longrightarrow$	1
$0\,0\,1\,1 \longrightarrow$	0

Why not using feed forward network?

- Variable length of sequences (potentially infinite)



Growing size with sequence length!

Modeling Sequences

Examples

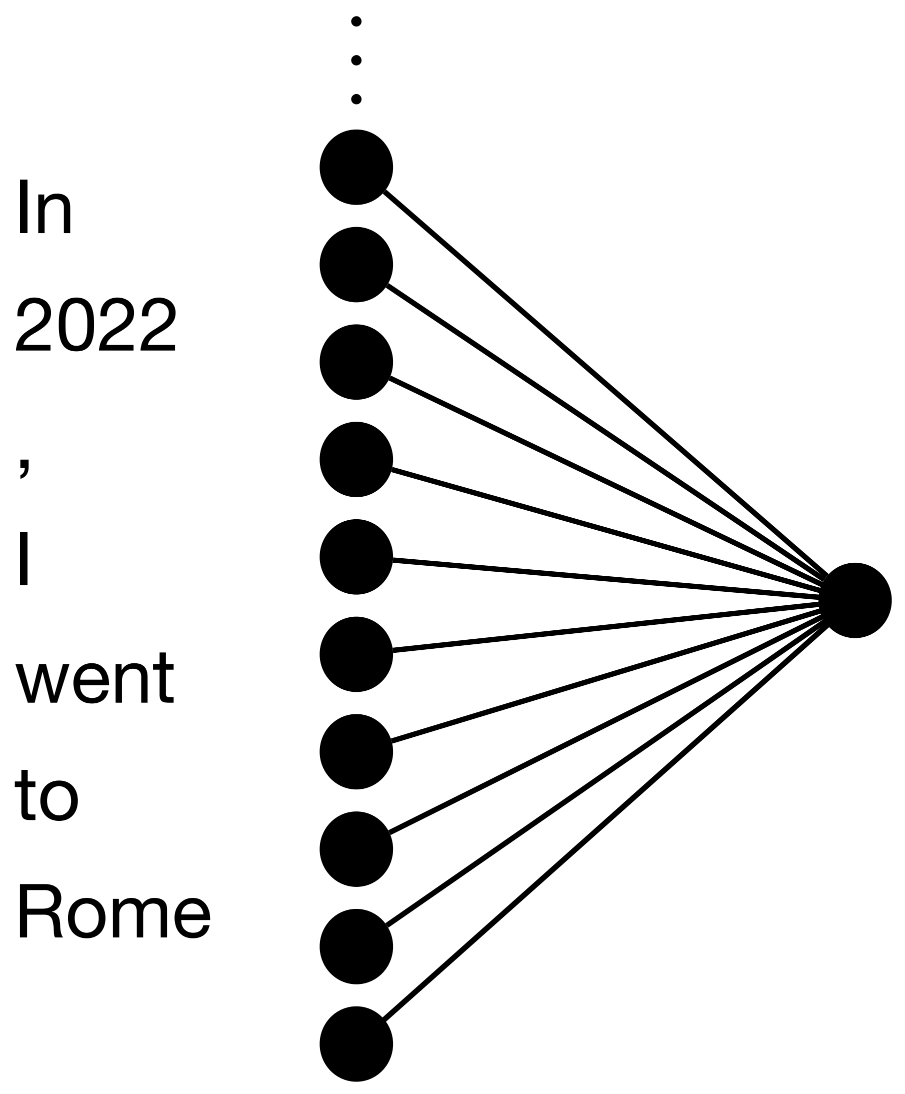
- Time series forecasting: predict x^{t+1} given $x^t, x^{t-1}, \ldots$
- Extract the year in which I went to Rome from “In **2022**, I went to Rome”
- Determine the parity of a binary input sequence, whether the number of 1’s is odd:

Input Output

$$0\,1\,0\,0\,1\,1\,0 \longrightarrow 1$$
$$0\,0\,1\,1 \longrightarrow 0$$

Why not using feed forward network?

- Variable length of sequences (potentially infinite)
- Dependencies within a sequence



Feed forward net uses separate parameters for each input word

Modeling Sequences

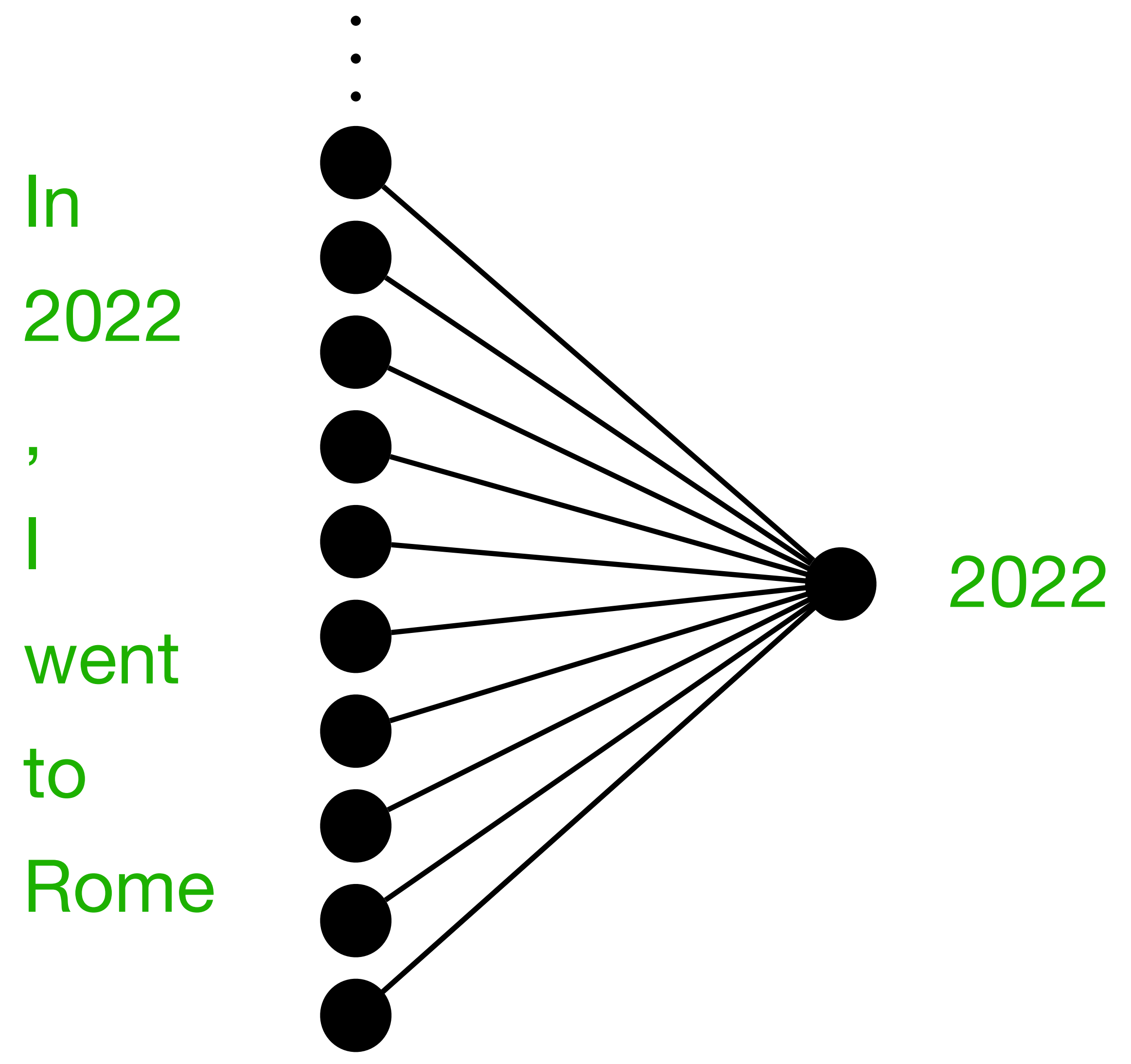
Examples

- Time series forecasting: predict x^{t+1} given $x^t, x^{t-1}, \dots$
- Extract the year in which I went to Rome from “In **2022**, I went to Rome”
- Determine the parity of a binary input sequence, whether the number of 1’s is odd:

Input	Output
$0\,1\,0\,0\,1\,1\,0 \longrightarrow$	1
$0\,0\,1\,1 \longrightarrow$	0

Why not using feed forward network?

- Variable length of sequences (potentially infinite)
- Dependencies within a sequence



Feed forward net correctly classifies the training sequence

Modeling Sequences

Examples

- Time series forecasting: predict x^{t+1} given $x^t, x^{t-1}, \ldots$

- Extract the year in which I went to Rome from "In **2022**, I went to Rome"

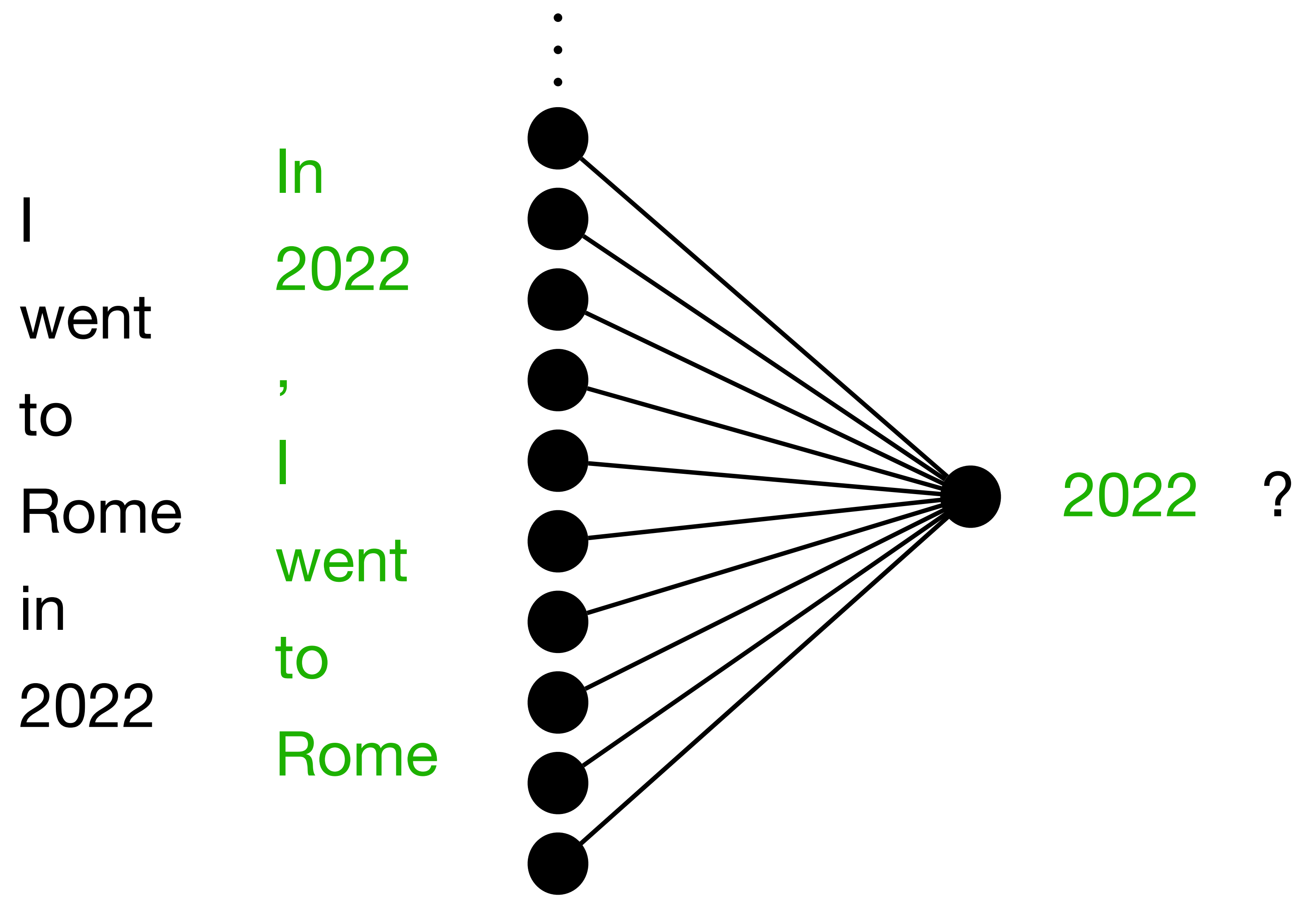
- Determine the parity of a binary input sequence, whether the number of 1's is odd:

Input	Output
$0\,1\,0\,0\,1\,1\,0 \longrightarrow$	1
$0\,0\,1\,1 \longrightarrow$	0

Why not using feed forward network?

- Variable length of sequences (potentially infinite)

- Dependencies within a sequence



Feed forward net correctly classifies the training sequence

Modeling Sequences

Examples

- Time series forecasting: predict x^{t+1} given $x^t, x^{t-1}, \ldots$

- Extract the year in which I went to Rome from "In **2022**, I went to Rome"

- Determine the parity of a binary input sequence, whether the number of 1's is odd:

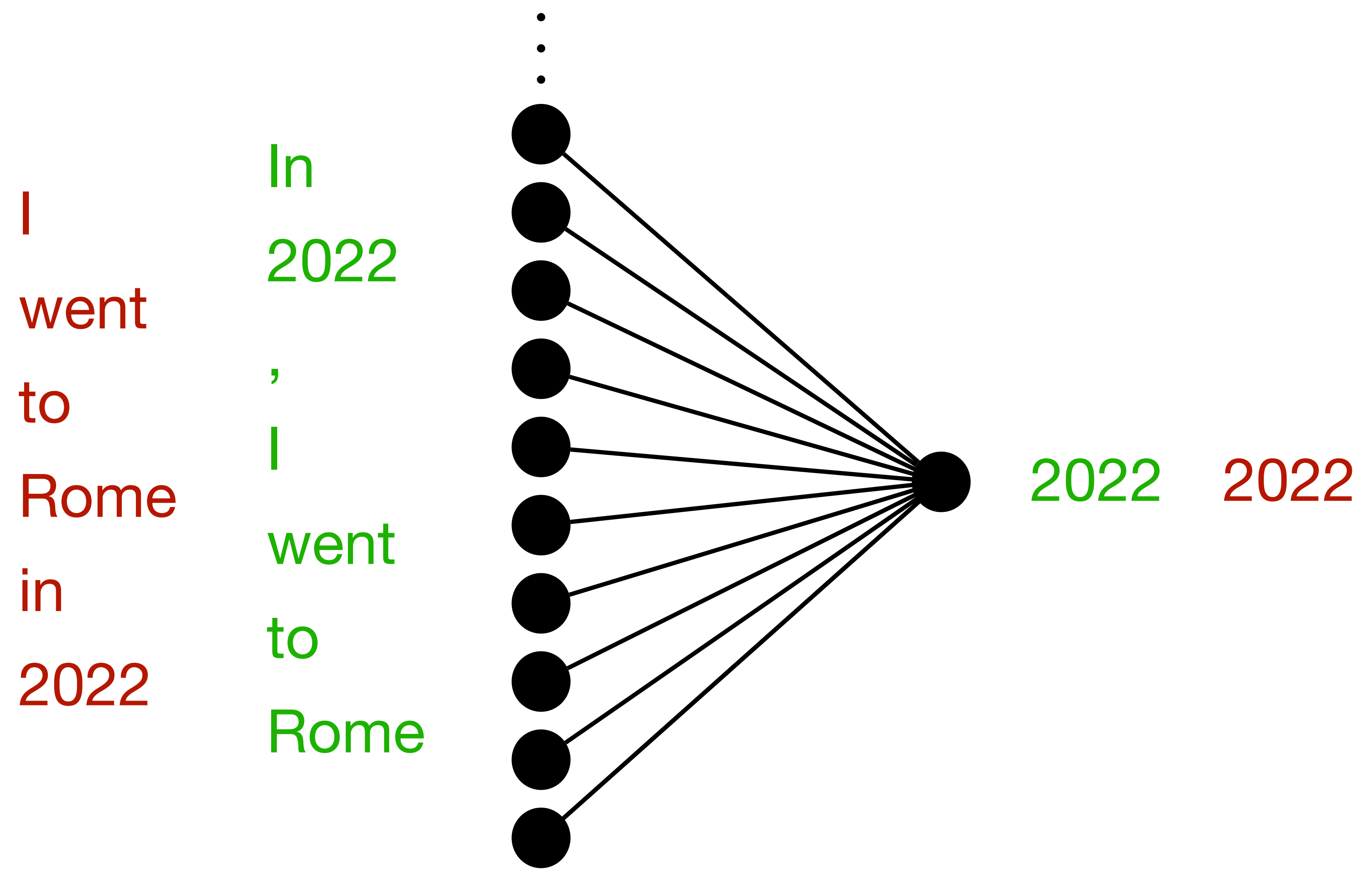
Input Output

$$0\,1\,0\,0\,1\,1\,0 \longrightarrow 1$$
$$0\,0\,1\,1 \longrightarrow 0$$

Why not using feed forward network?

- Variable length of sequences (potentially infinite)

- Dependencies within a sequence



Feed forward net needs to see all possible sentences during training and learn the rules for every position

Modeling Sequences

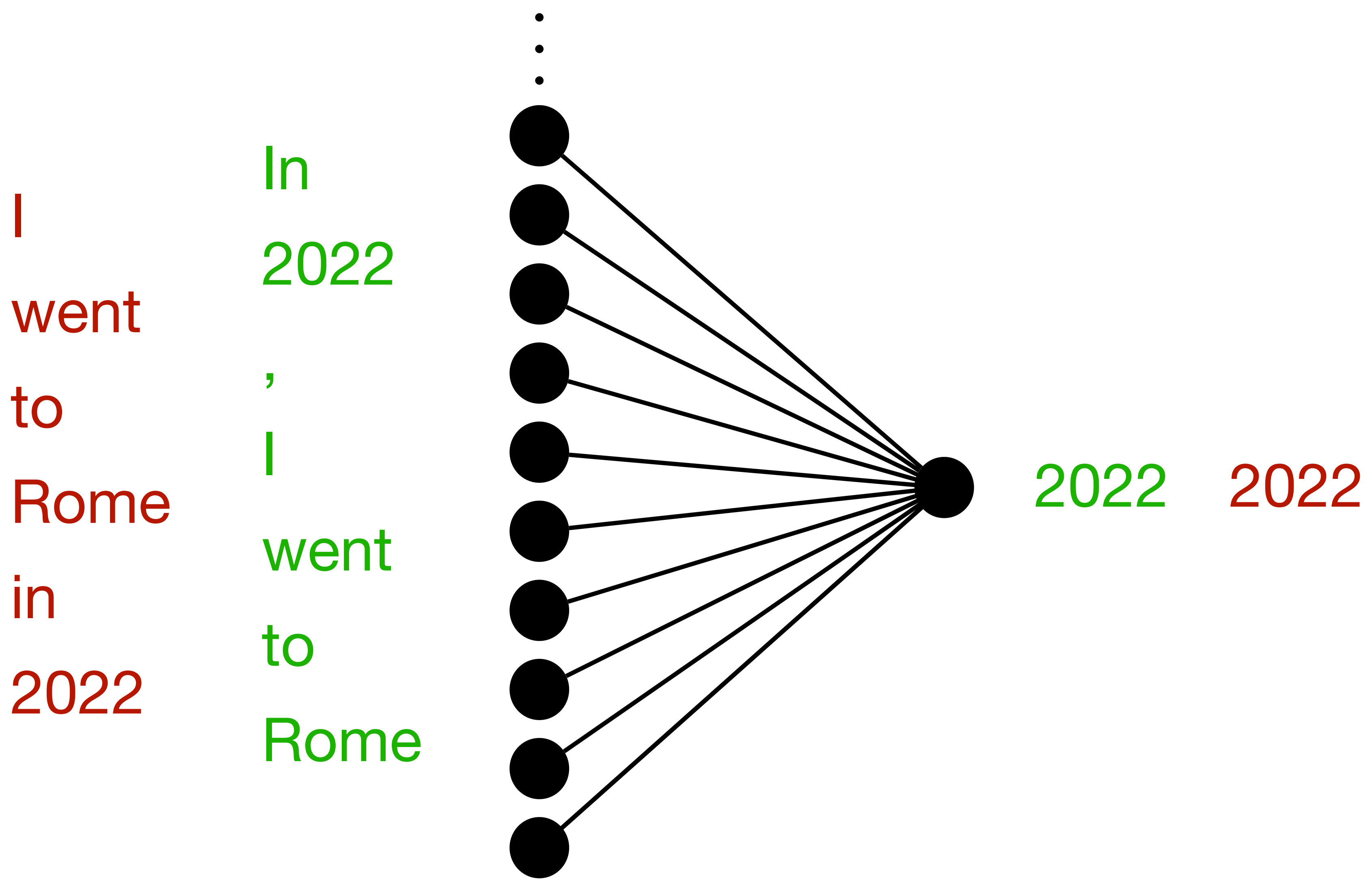
Examples

- Time series forecasting: predict x^{t+1} given $x^t, x^{t-1}, \ldots$
- Extract the year in which I went to Rome from "In **2022**, I went to Rome"
- Determine the parity of a binary input sequence, whether the number of 1's is odd:

Input	Output
$0\,1\,0\,0\,1\,1\,0 \longrightarrow$	1
$0\,0\,1\,1 \longrightarrow$	0

Why not using feed forward network?

- Variable length of sequences (potentially infinite)
- Dependencies within a sequence



Inefficient and practically infeasible!

Modeling Sequences

Examples

- Time series forecasting: predict x^{t+1} given $x^t, x^{t-1}, \ldots$

- Extract the year in which I went to Rome from “In **2022**, I went to Rome”

- Determine the parity of a binary input sequence, whether the number of 1’s is odd:

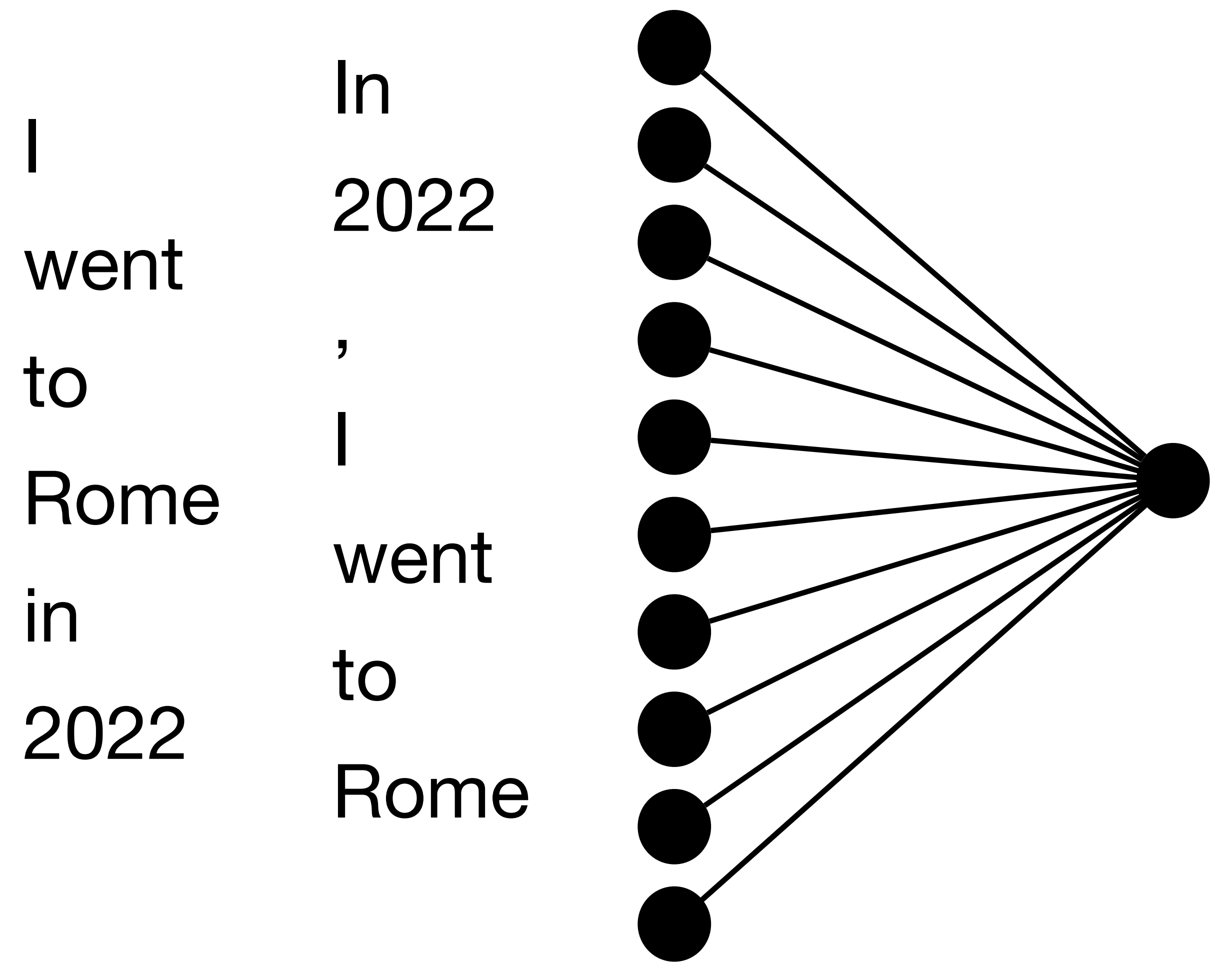
Input Output

$$0\,1\,0\,0\,1\,1\,0 \longrightarrow 1$$
$$0\,0\,1\,1 \longrightarrow 0$$

Why not using feed forward network?

- Variable length of sequences (potentially infinite)

- Dependencies within a sequence



Parameter sharing across sequence steps to form memory of the past

Memory

Learn the parity of a binary input sequence, whether the number of 1’s is odd or even

Strategy: track the parity over the sequence

Output							1
Input	0	1	0	0	1	1	0

Memory

Learn the parity of a binary input sequence, whether the number of 1’s is odd or even

Strategy: track the parity over the sequence

Output 1

0

Input 0 1 0 0 1 1 0

x^1

Memory

Learn the parity of a binary input sequence, whether the number of 1's is odd or even

Strategy: track the parity over the sequence

Output 1

0 ?

$\uparrow$ $\uparrow$

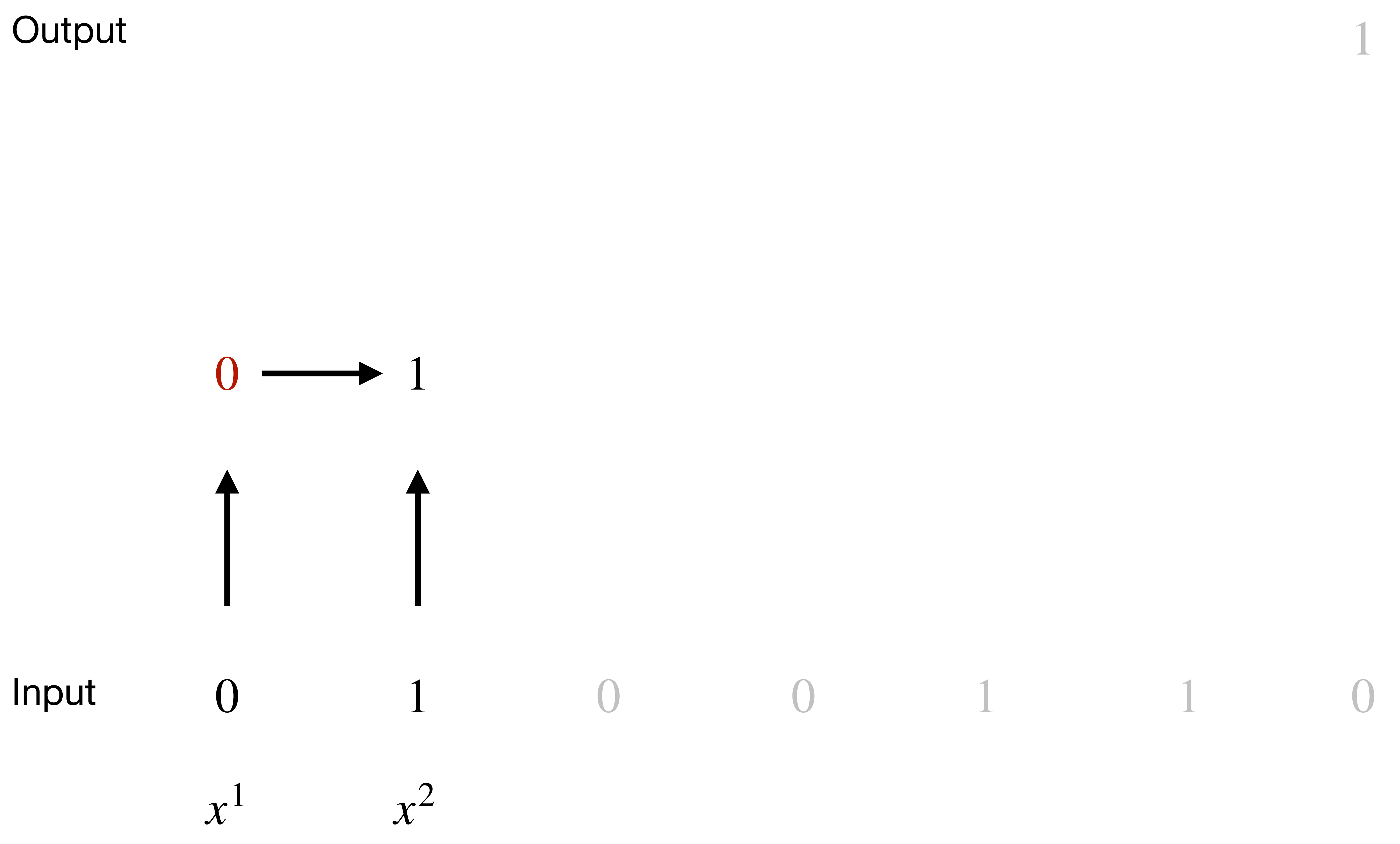
Input 0 1 0 0 1 1 0

x^1 x^2

Memory

Learn the parity of a binary input sequence, whether the number of 1’s is odd or even

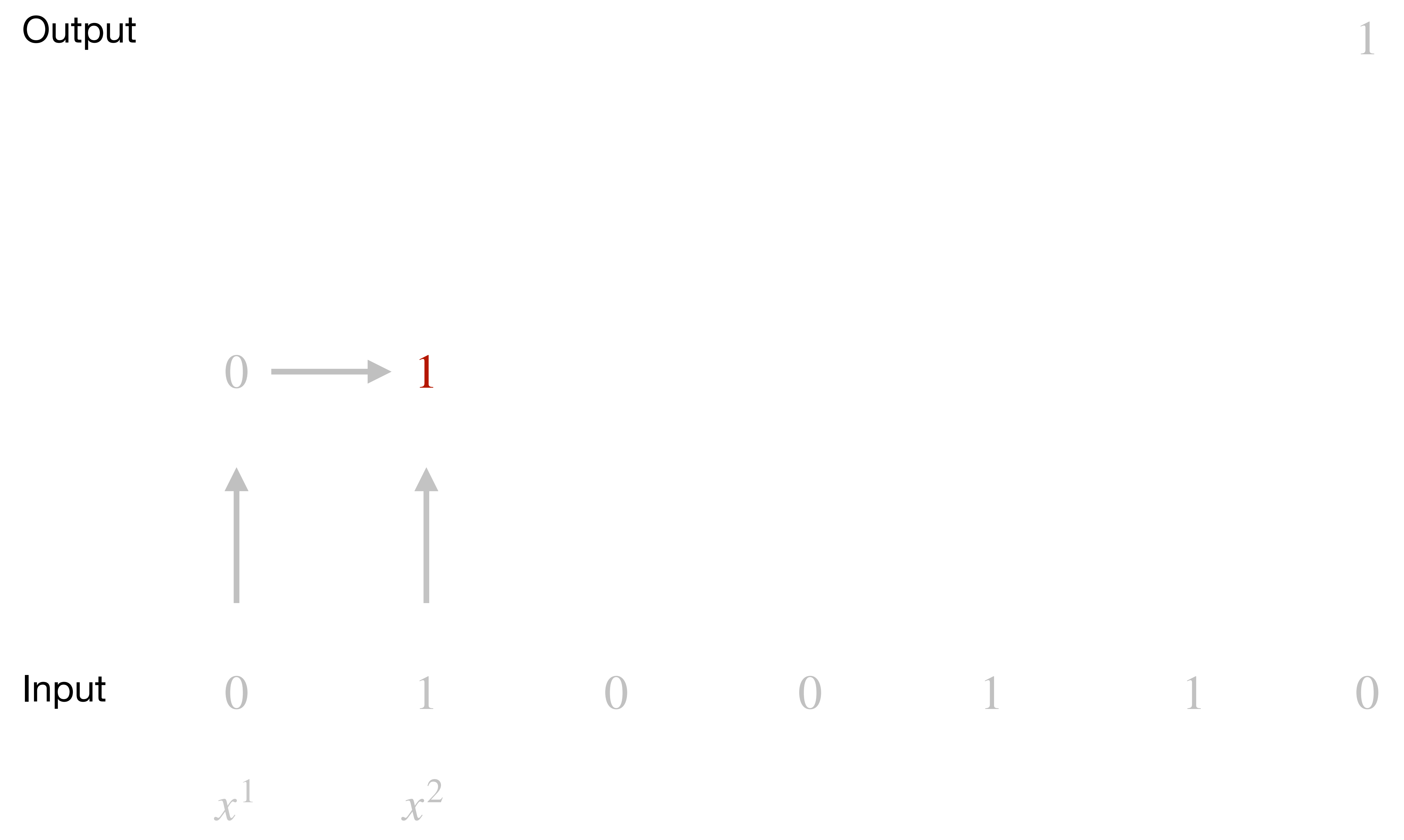
Strategy: track the parity over the sequence



Memory

Learn the parity of a binary input sequence, whether the number of 1's is odd or even

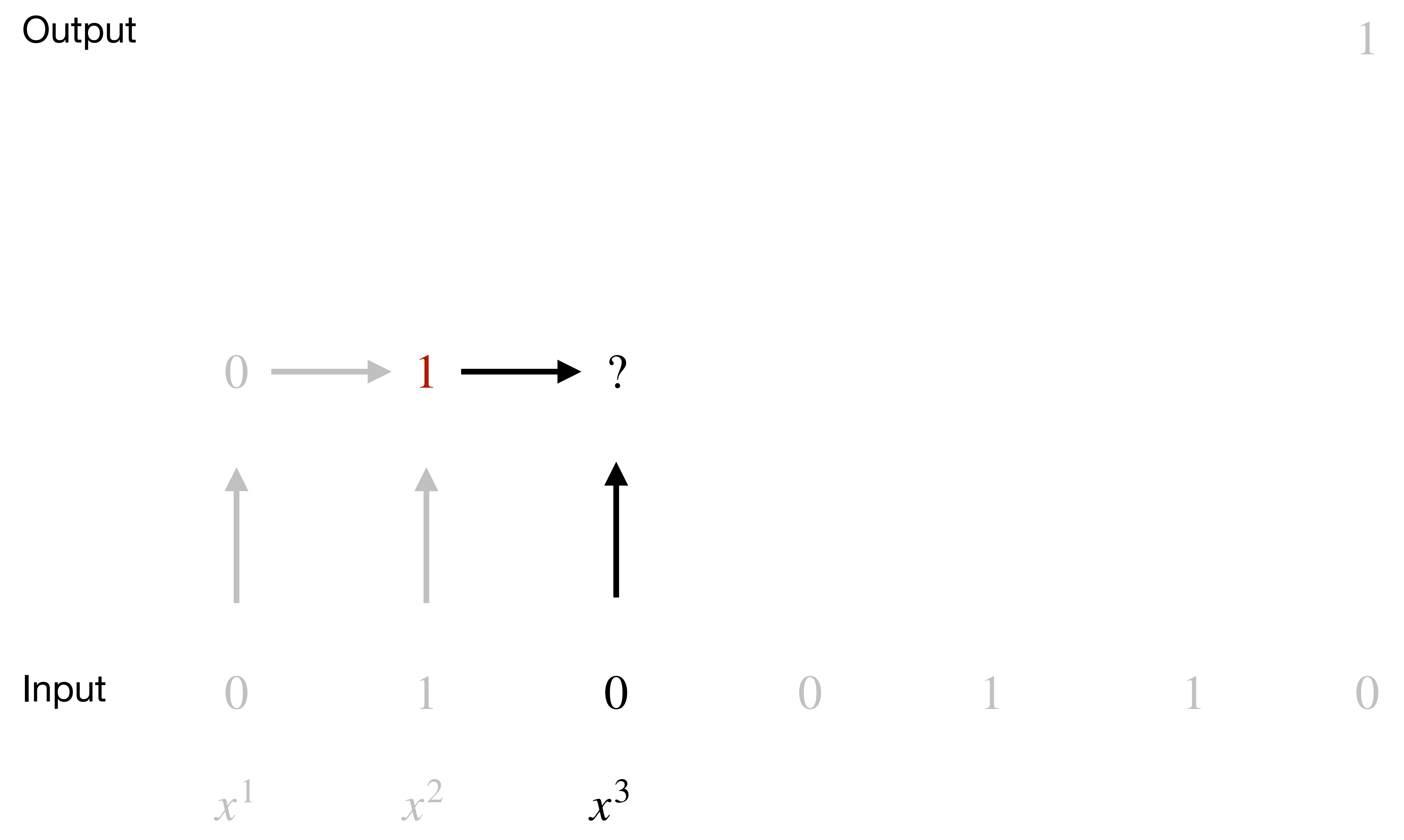
Strategy: track the parity over the sequence



Memory

Learn the parity of a binary input sequence, whether the number of 1's is odd or even

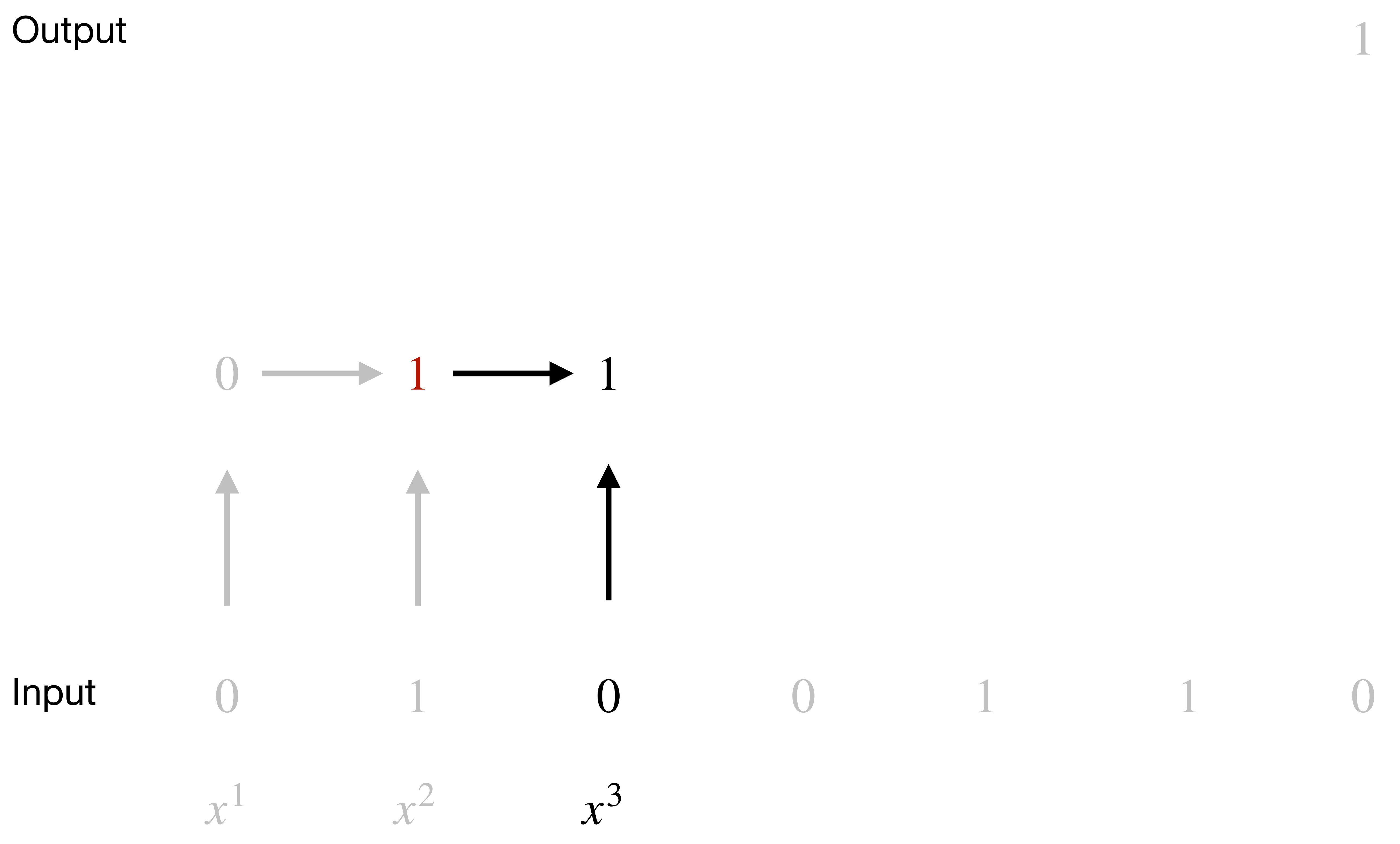
Strategy: track the parity over the sequence



Memory

Learn the parity of a binary input sequence, whether the number of 1's is odd or even

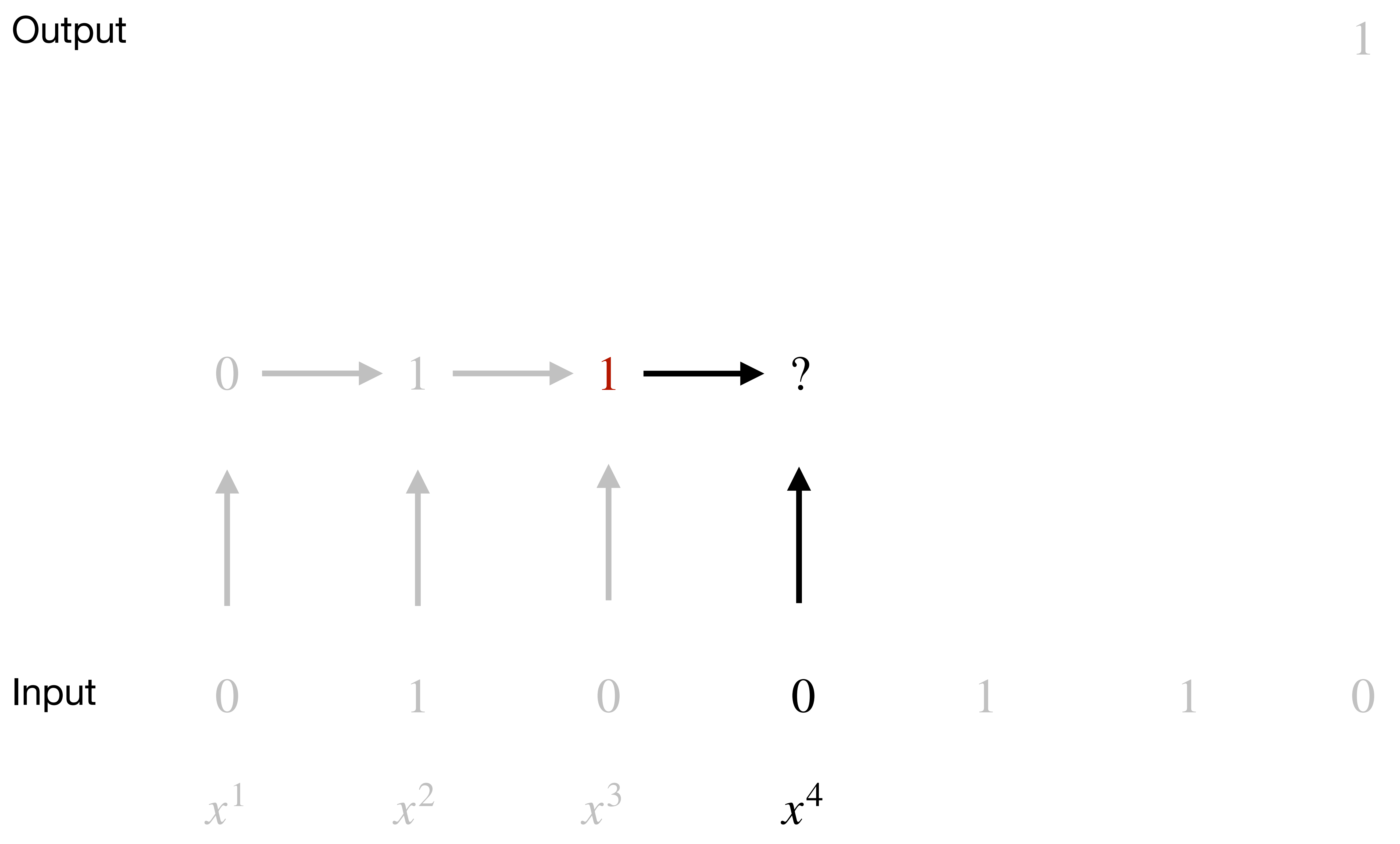
Strategy: track the parity over the sequence



Memory

Learn the parity of a binary input sequence, whether the number of 1’s is odd or even

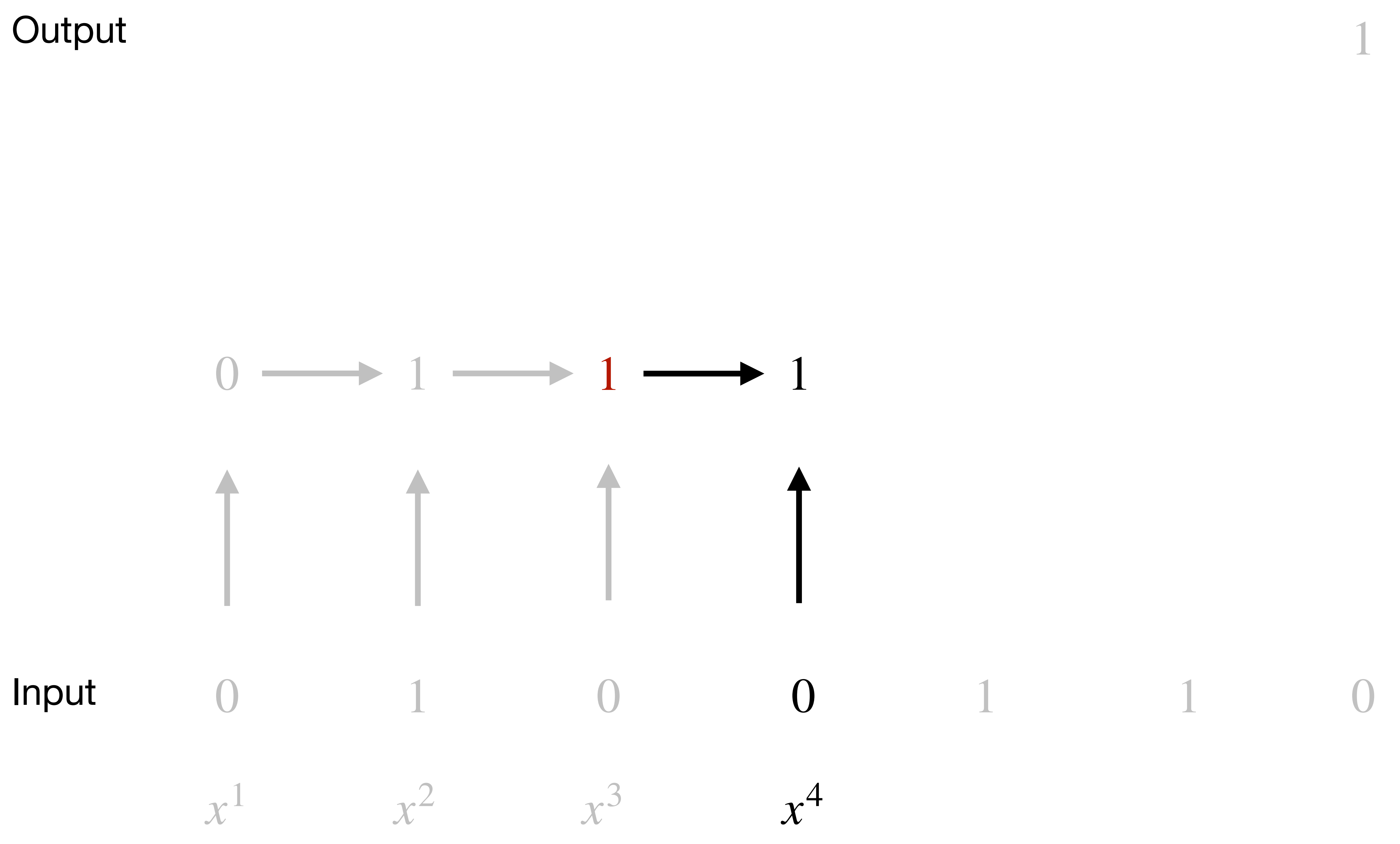
Strategy: track the parity over the sequence



Memory

Learn the parity of a binary input sequence, whether the number of 1's is odd or even

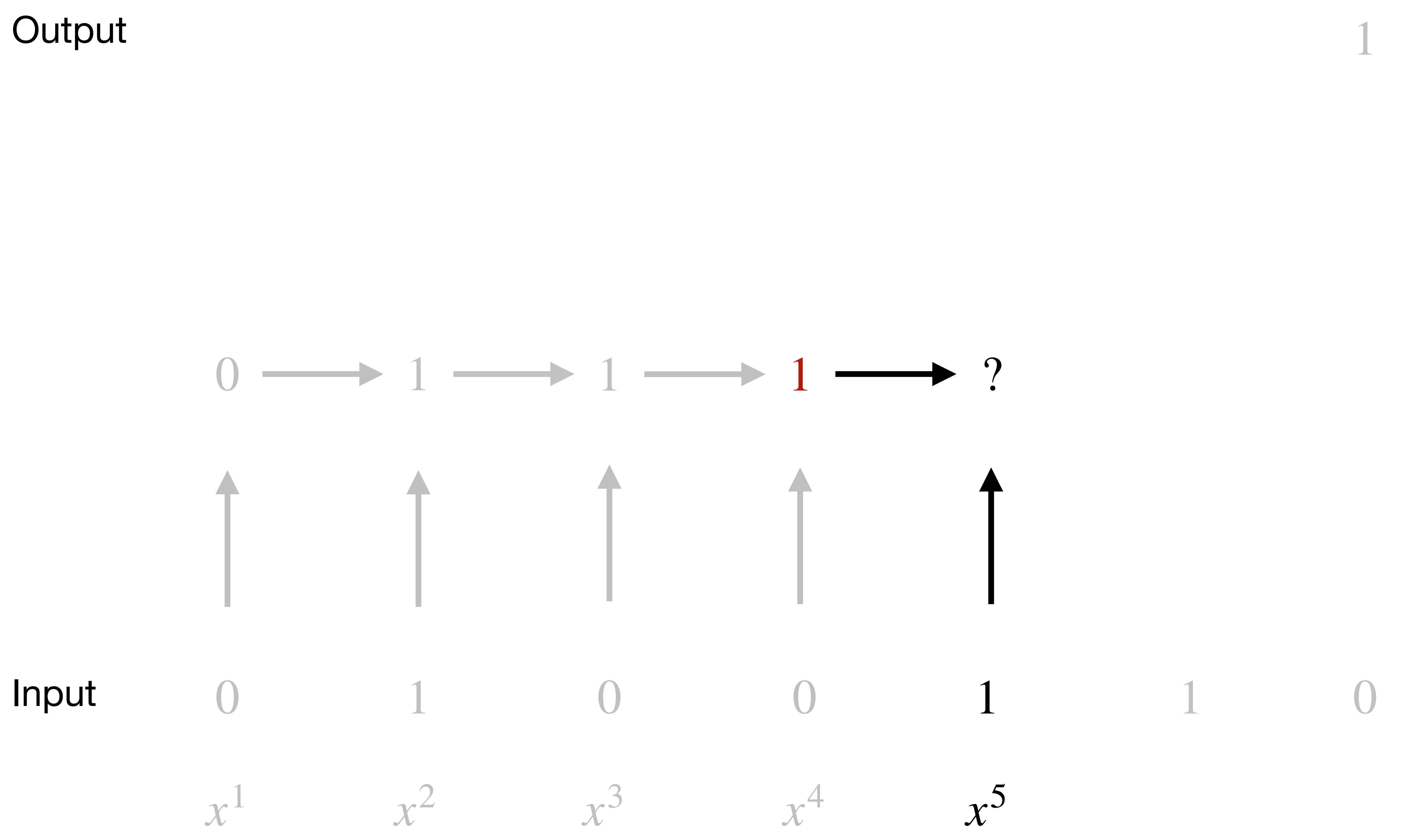
Strategy: track the parity over the sequence



Memory

Learn the parity of a binary input sequence, whether the number of 1's is odd or even

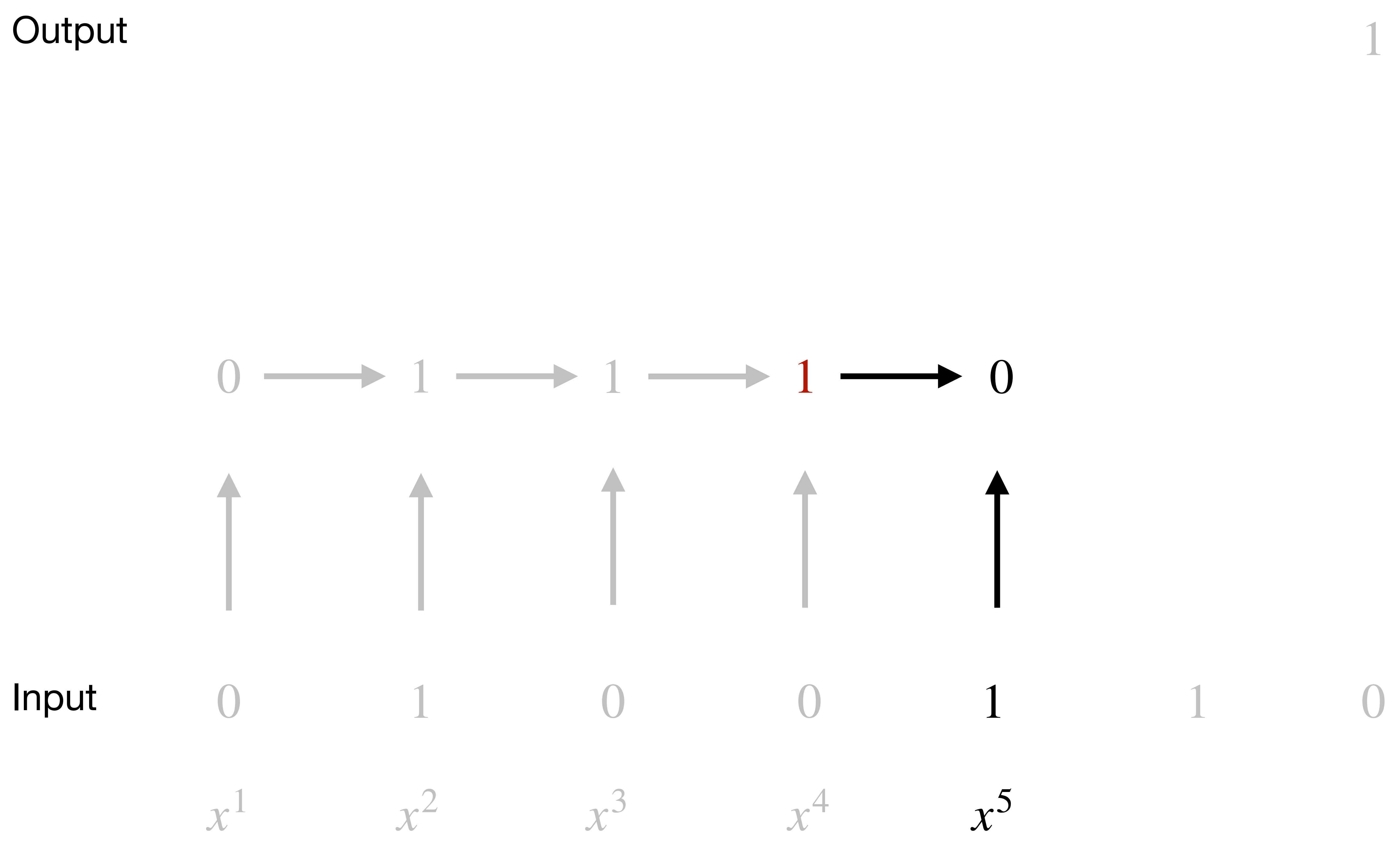
Strategy: track the parity over the sequence



Memory

Learn the parity of a binary input sequence, whether the number of 1's is odd or even

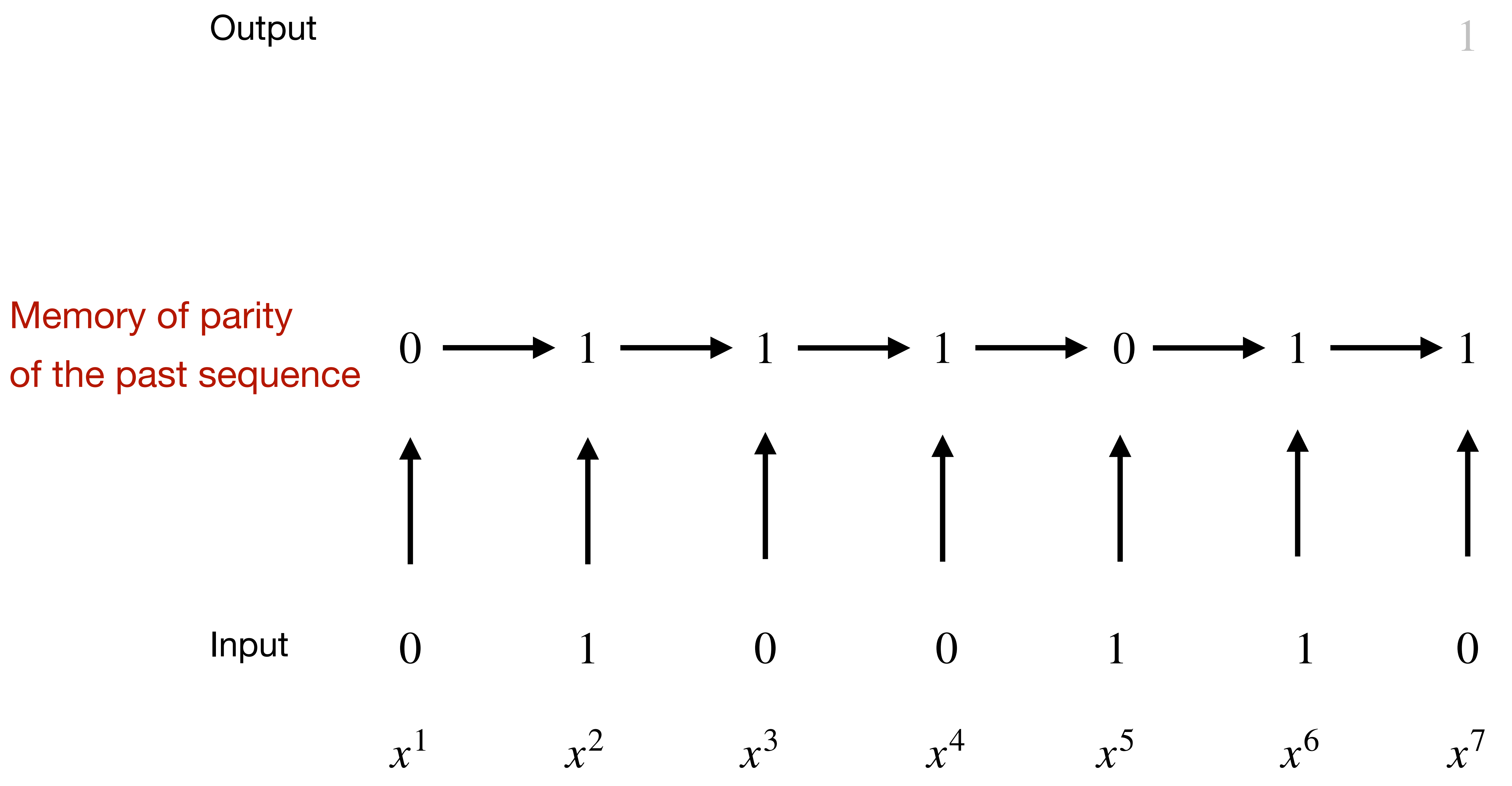
Strategy: track the parity over the sequence



Memory

Learn the parity of a binary input sequence, whether the number of 1's is odd or even

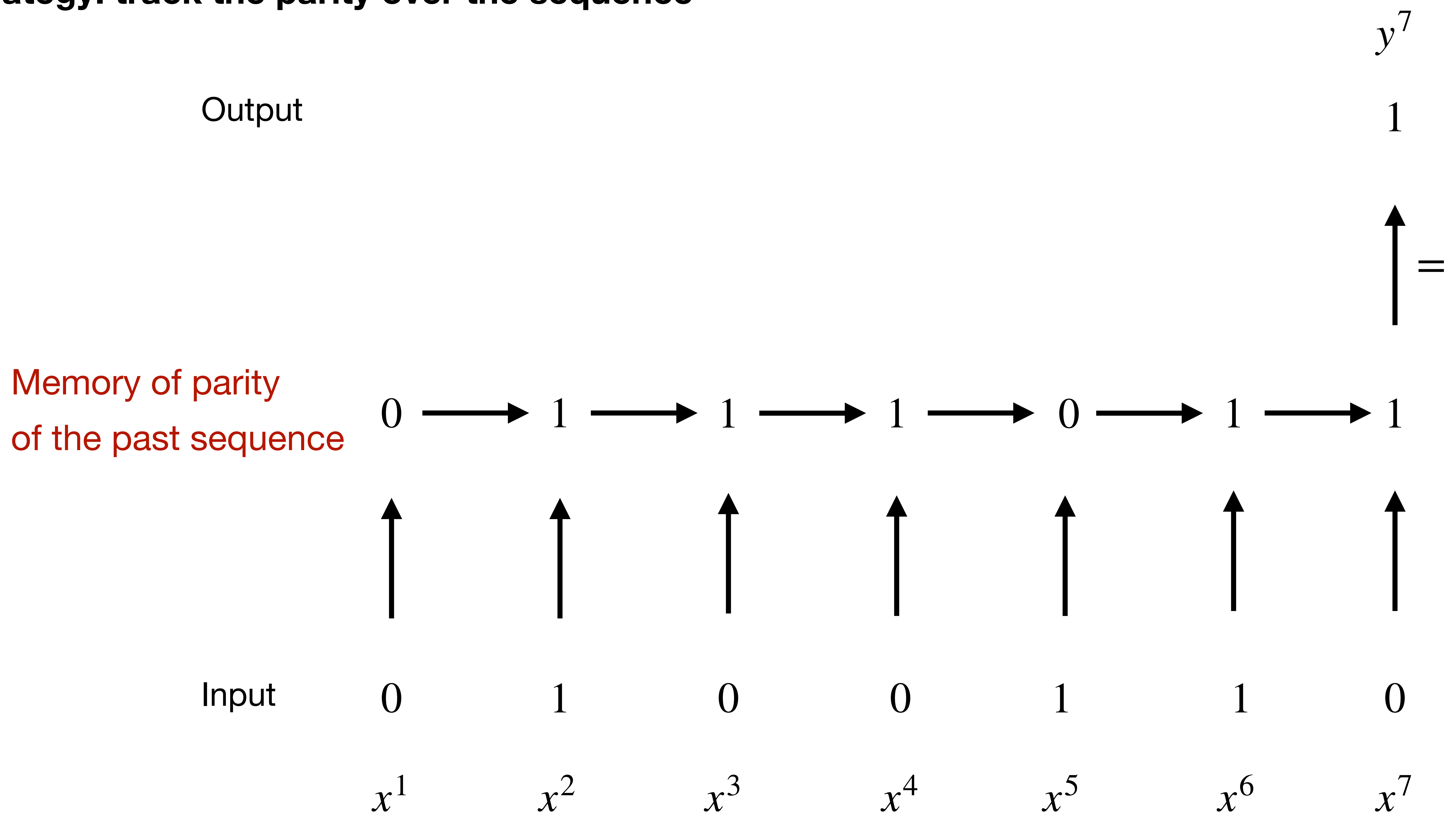
Strategy: track the parity over the sequence



Memory

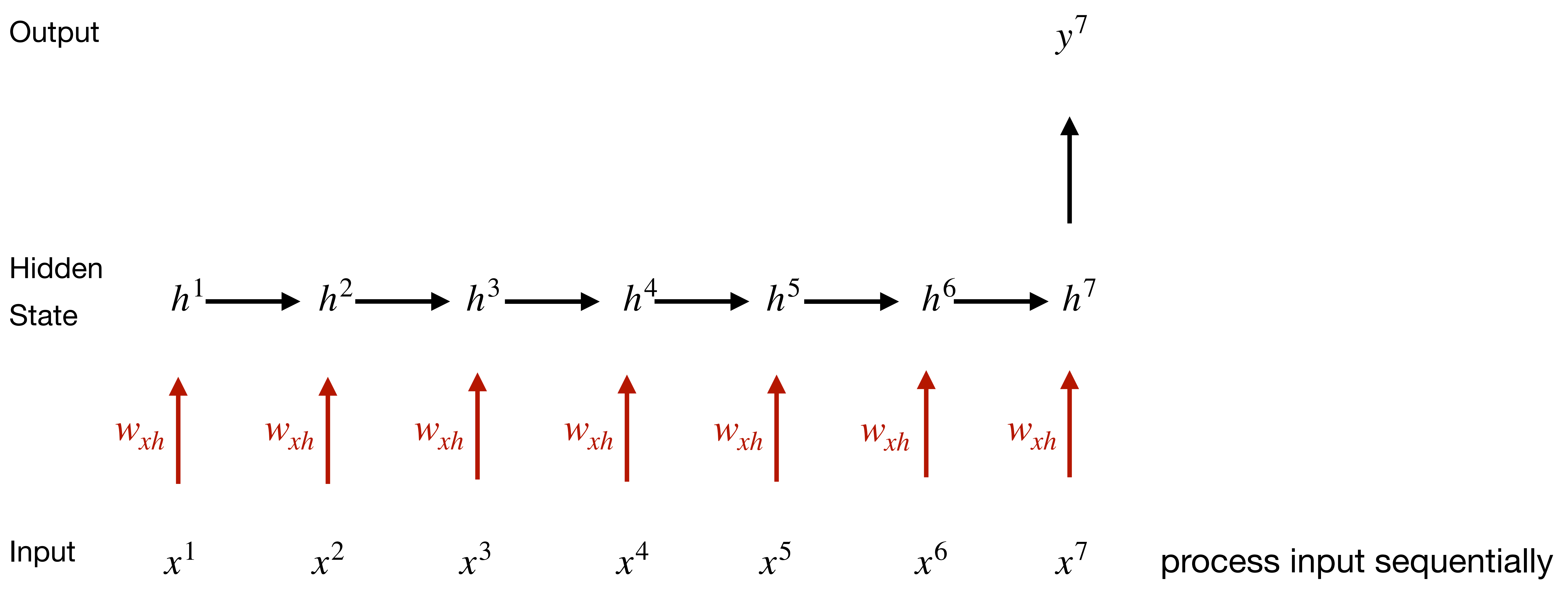
Learn the parity of a binary input sequence, whether the number of 1's is odd or even

Strategy: track the parity over the sequence



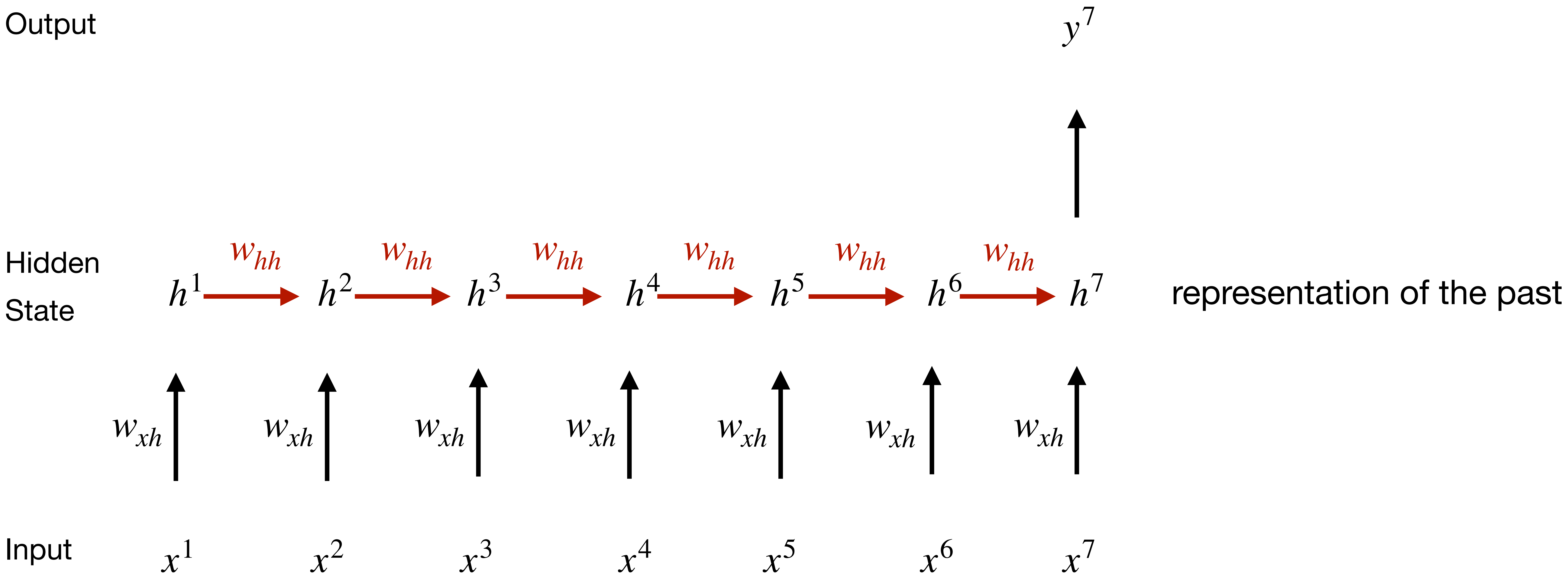
Recurrent Neural Network Cell

Process a sequence in order by sharing the same parameters at each time step



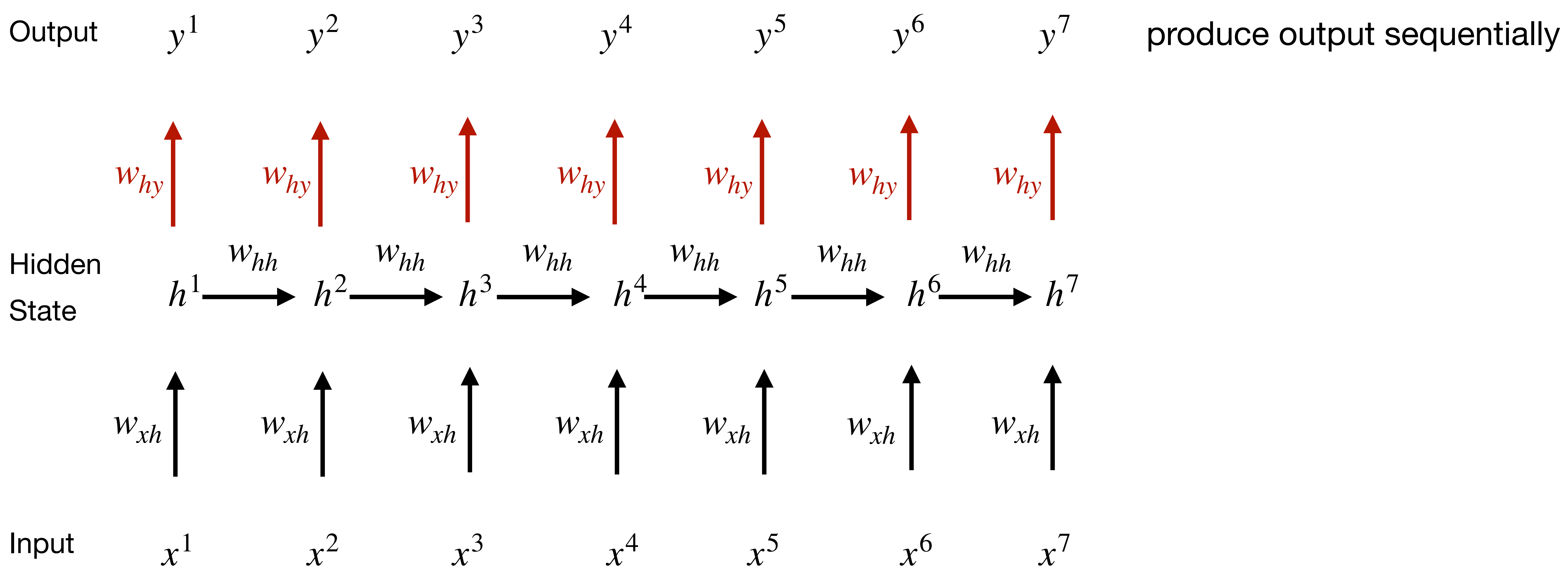
Recurrent Neural Network Cell

Process a sequence in order by sharing the same parameters at each time step



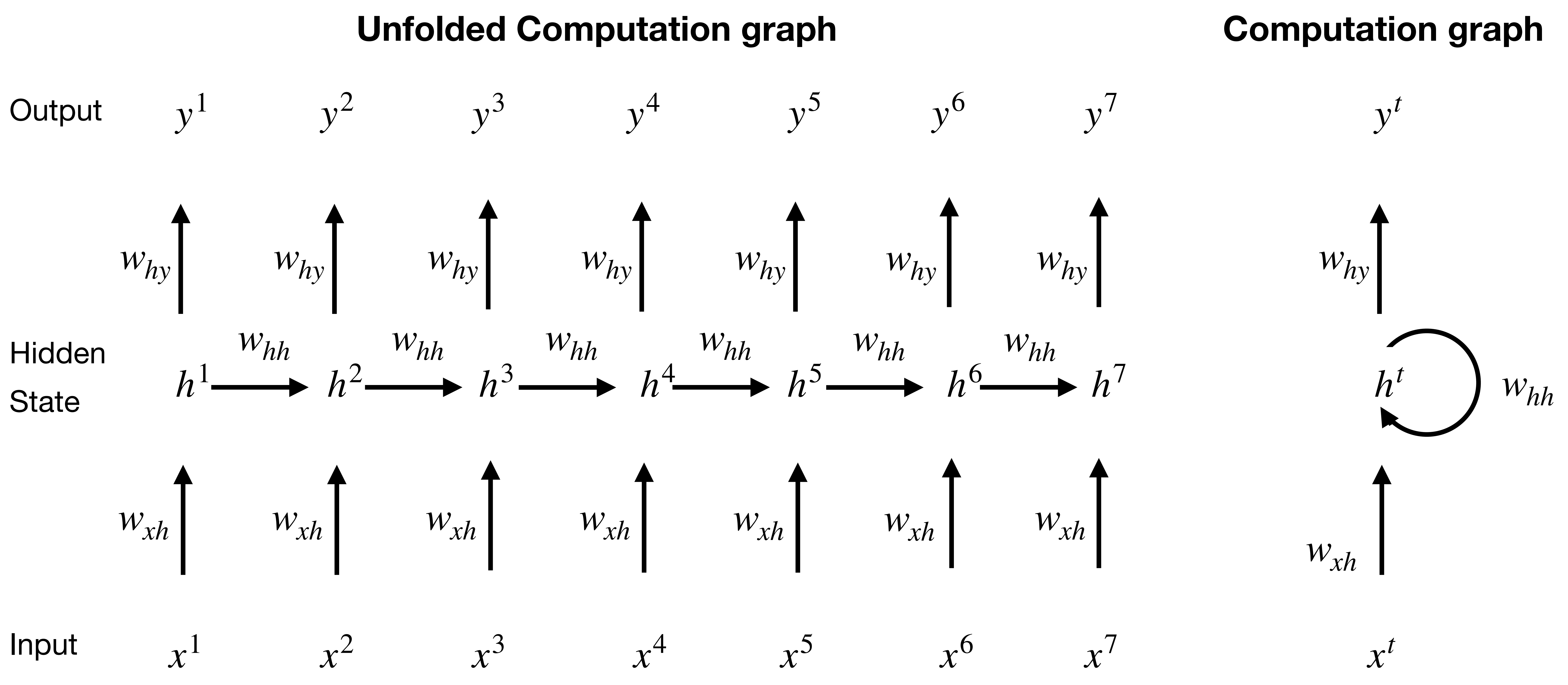
Recurrent Neural Network Cell

Process a sequence in order by sharing the same parameters at each time step



Recurrent Neural Network Cell

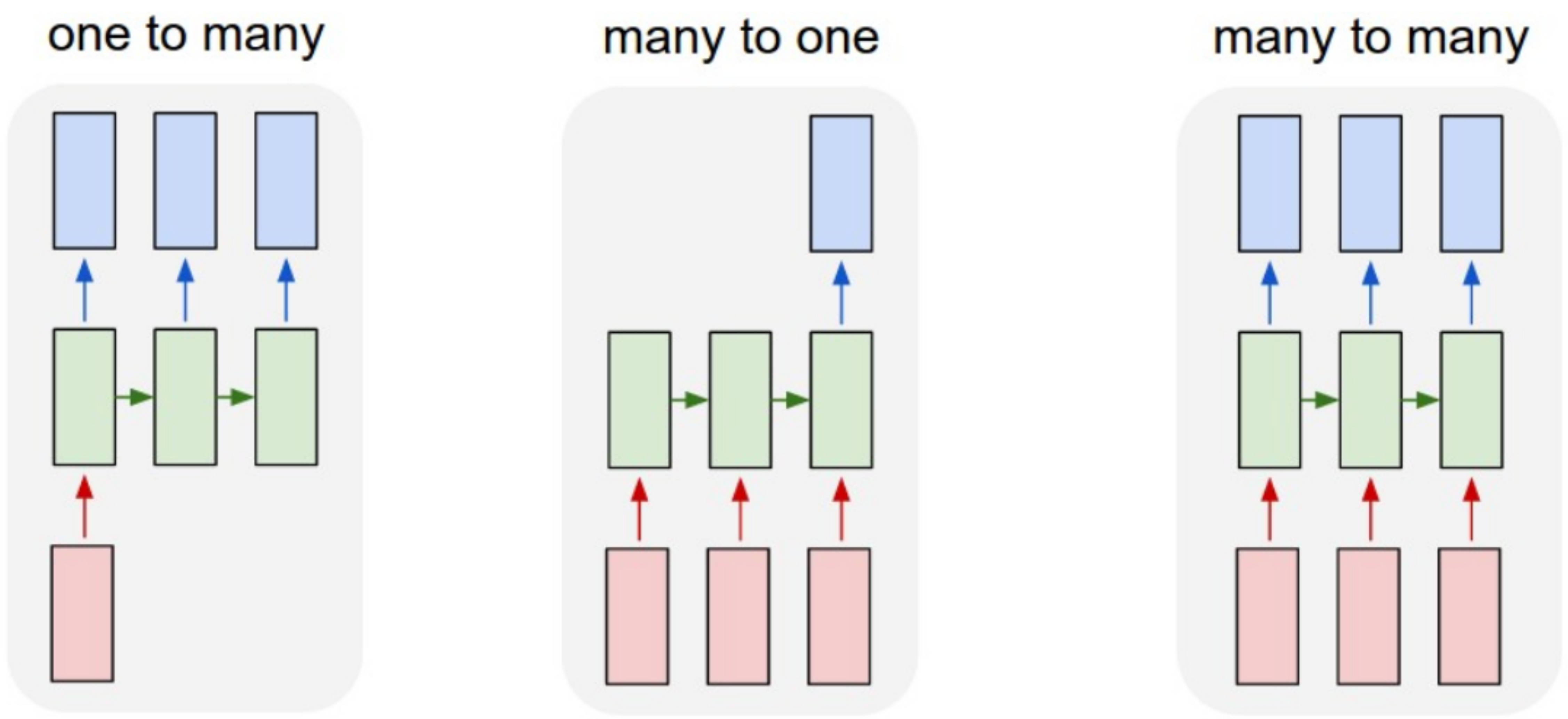
Process a sequence in order by sharing the same parameters at each time step



The amount of computational steps depends on the input sequence length but the network size doesn't

Recurrent Neural Network Cell

Process a sequence in order by sharing the same parameters at each time step



Sequence output (image captioning) | Sequence input (time series forecasting) | Sequence input and output * (action recognition in videos)

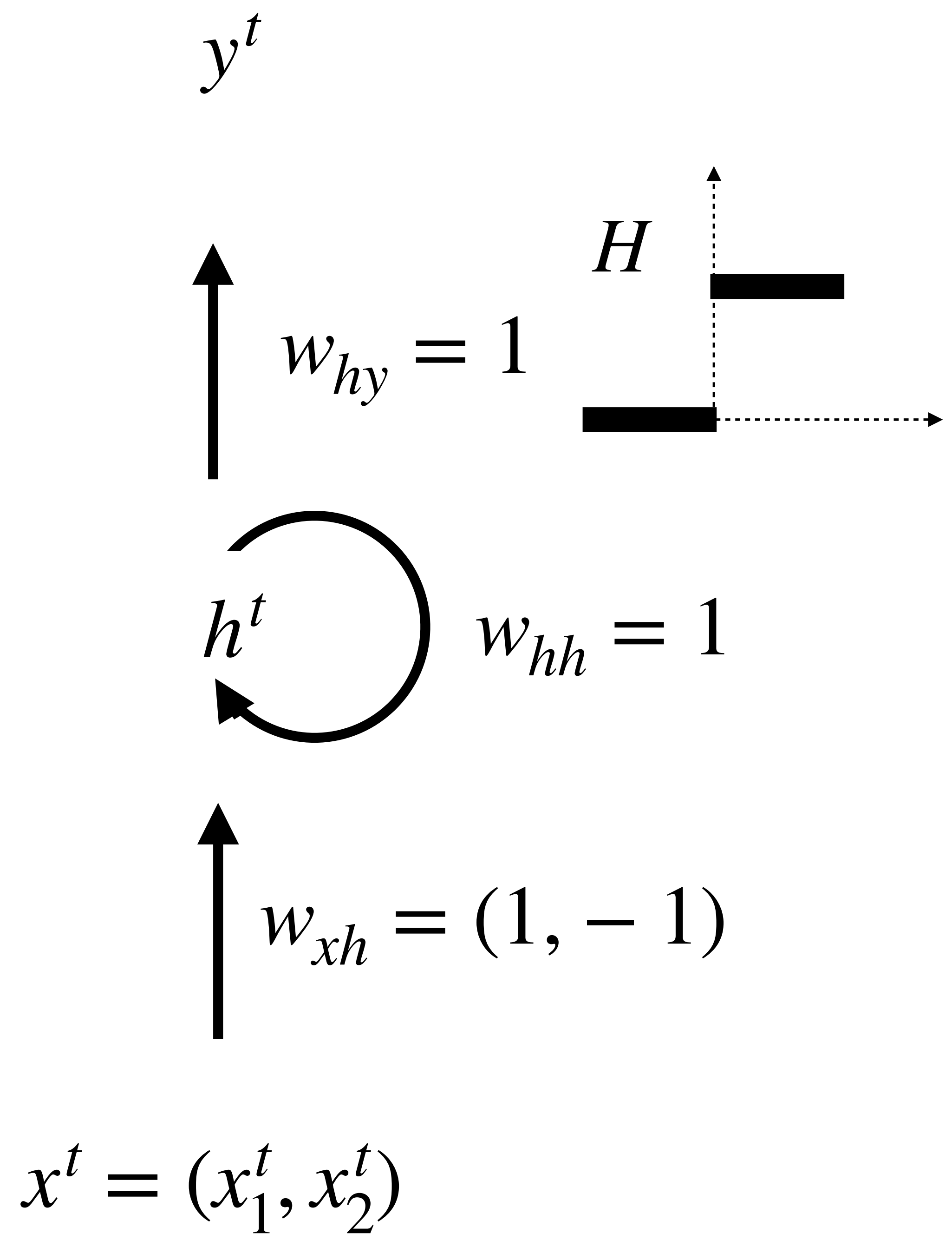
RNNs can process input/ouput sequences

* Andrej Karpathy's blog: "The Unreasonable Effectiveness of Recurrent Neural Networks "

Forward Pass

What does this RNN do?

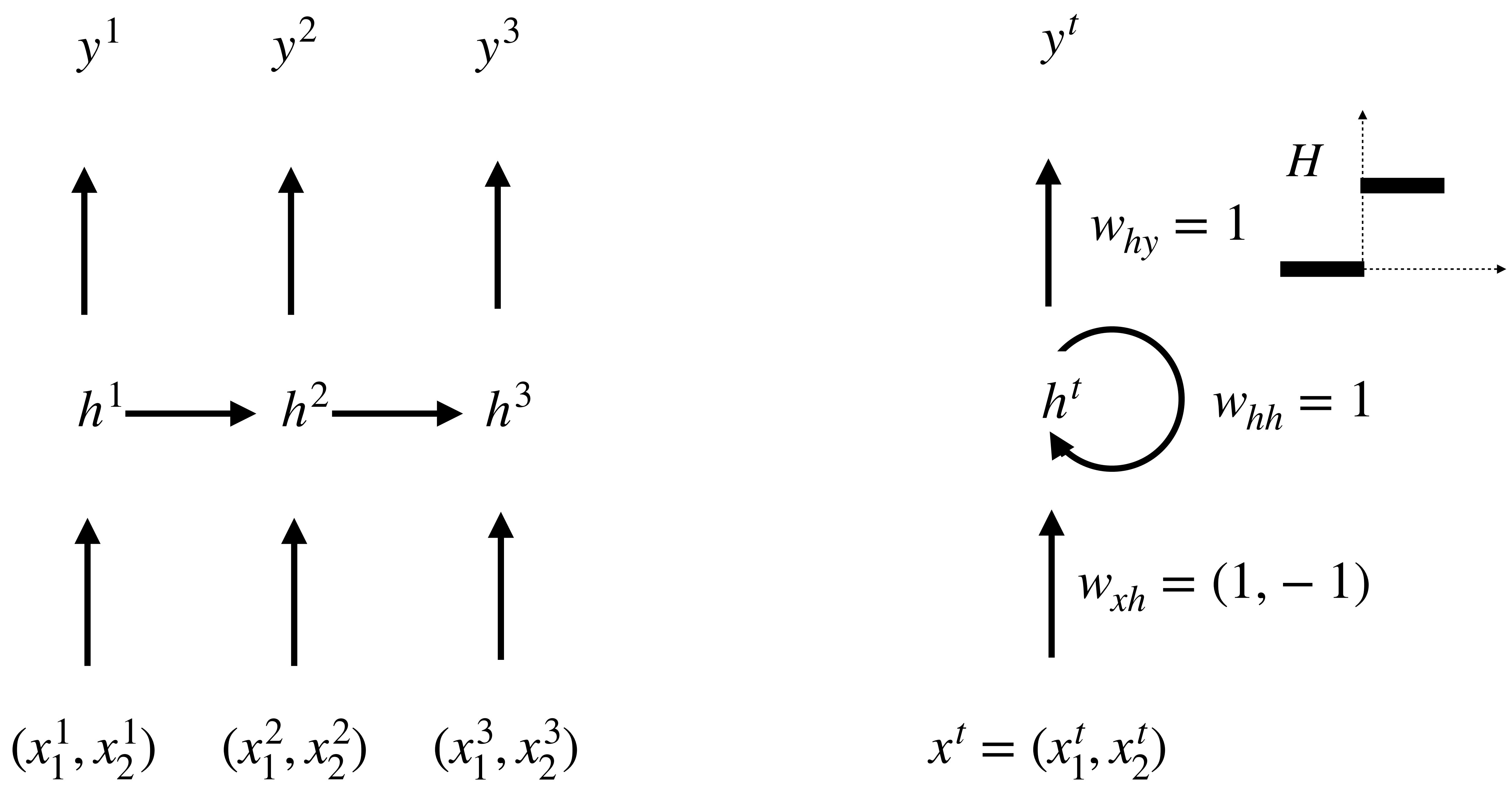
Input sequence $x^1, x^2, \ldots, x^t, \ldots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$



Forward Pass

What does this RNN do?

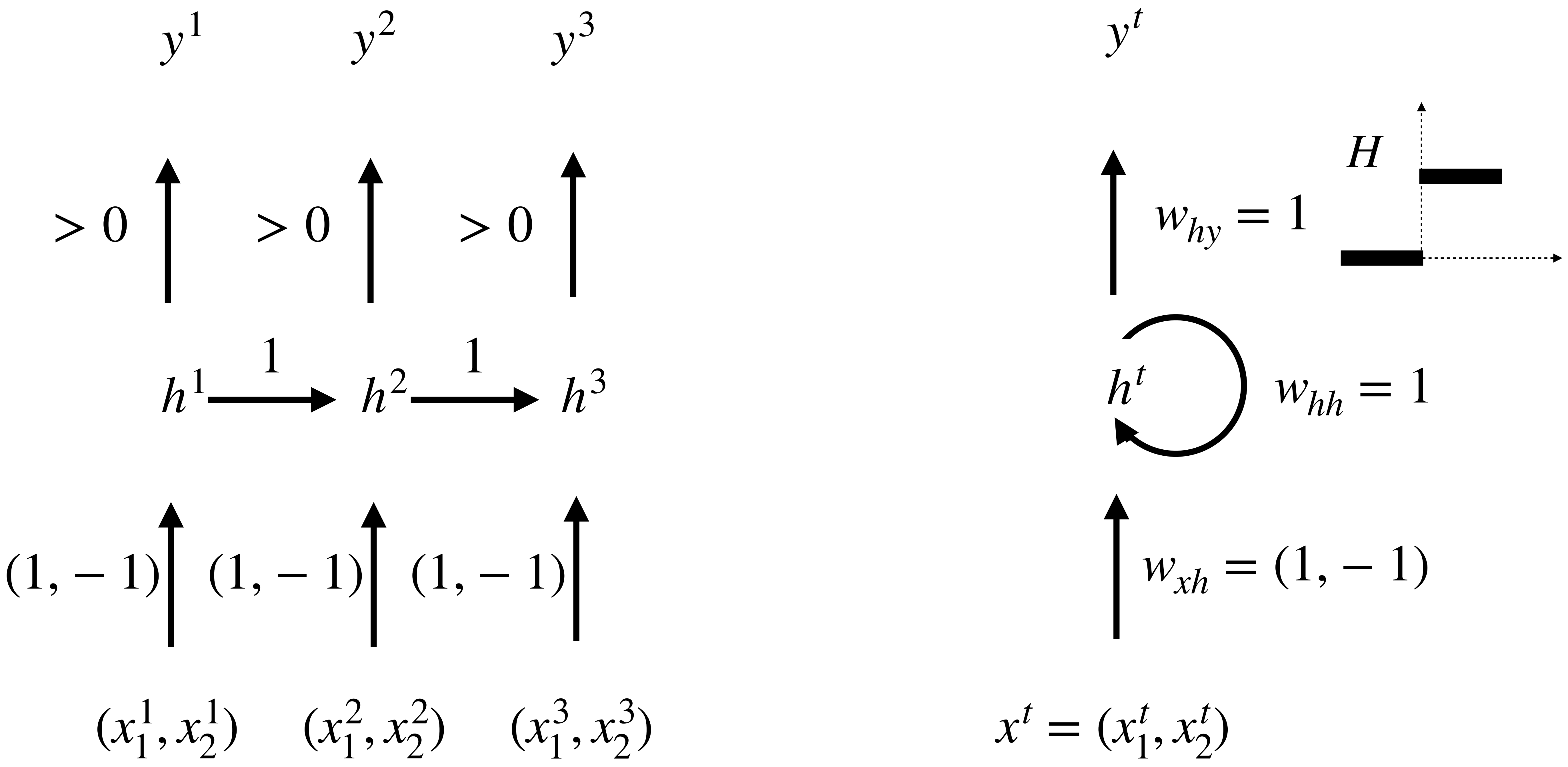
Input sequence $x^1, x^2, \ldots, x^t, \ldots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$



Forward Pass

What does this RNN do?

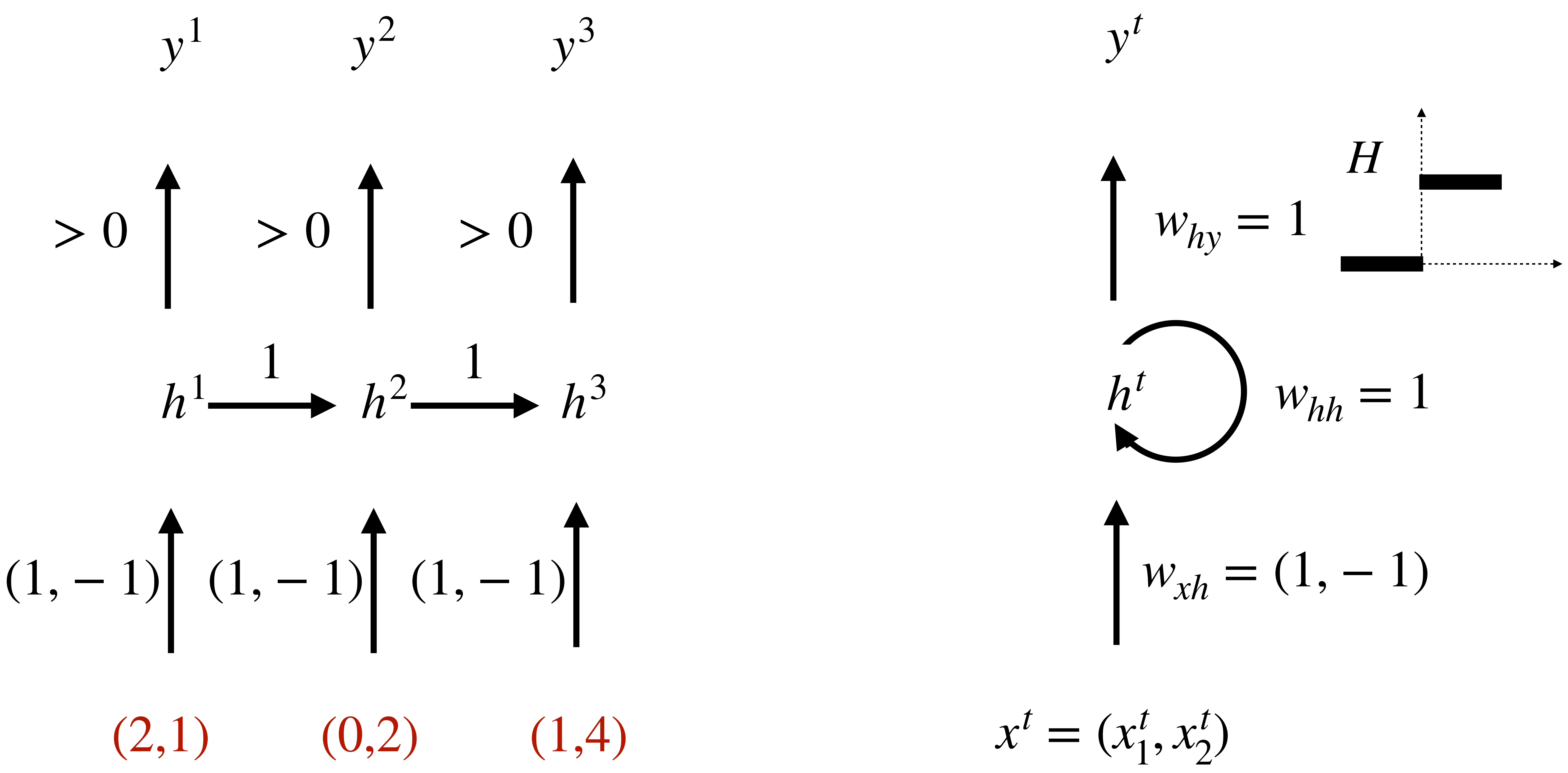
Input sequence $x^1, x^2, \ldots, x^t, \ldots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$



Forward Pass

What does this RNN do?

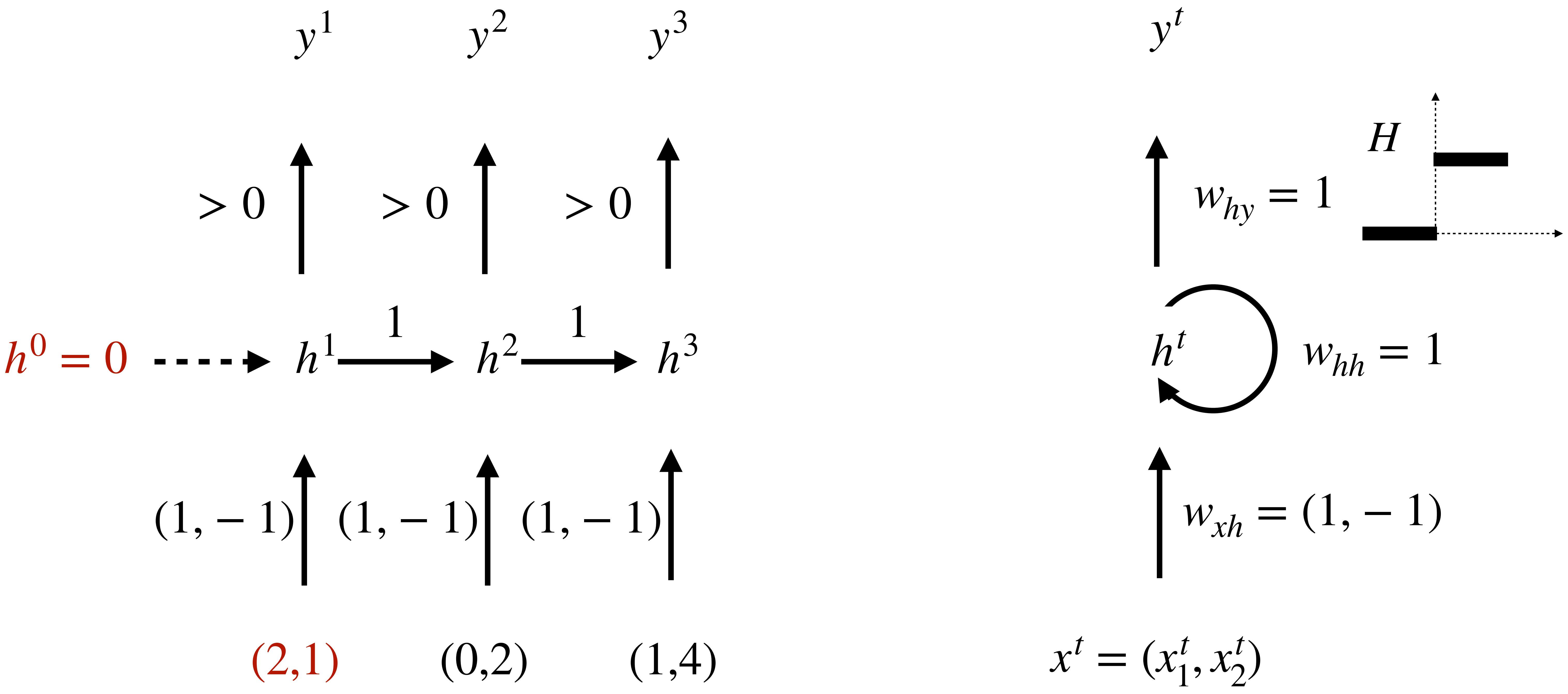
Input sequence $x^1, x^2, \ldots, x^t, \ldots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$



Forward Pass

What does this RNN do?

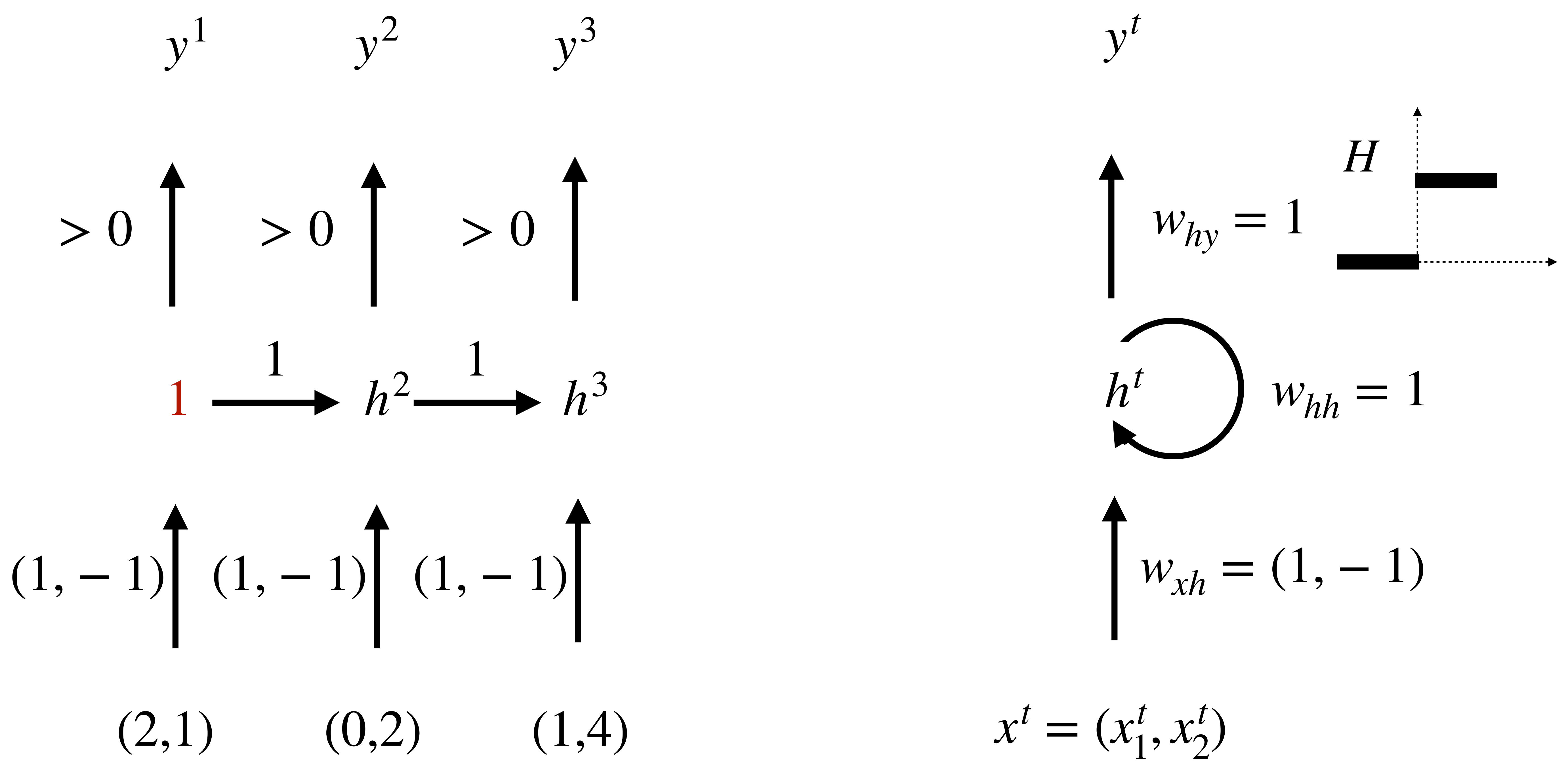
Input sequence $x^1, x^2, \ldots, x^t, \ldots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$



Forward Pass

What does this RNN do?

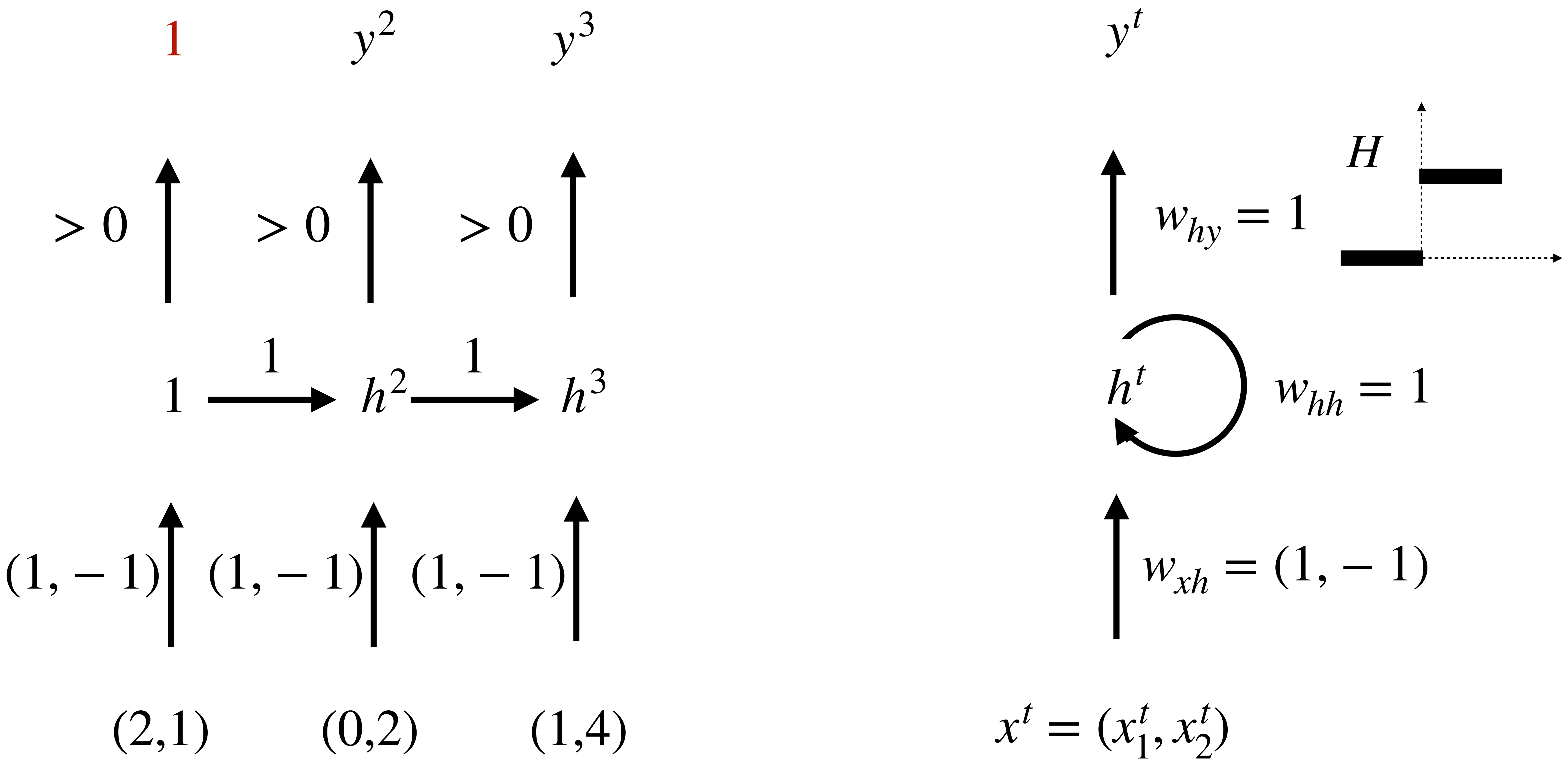
Input sequence $x^1, x^2, \ldots, x^t, \ldots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$



Forward Pass

What does this RNN do?

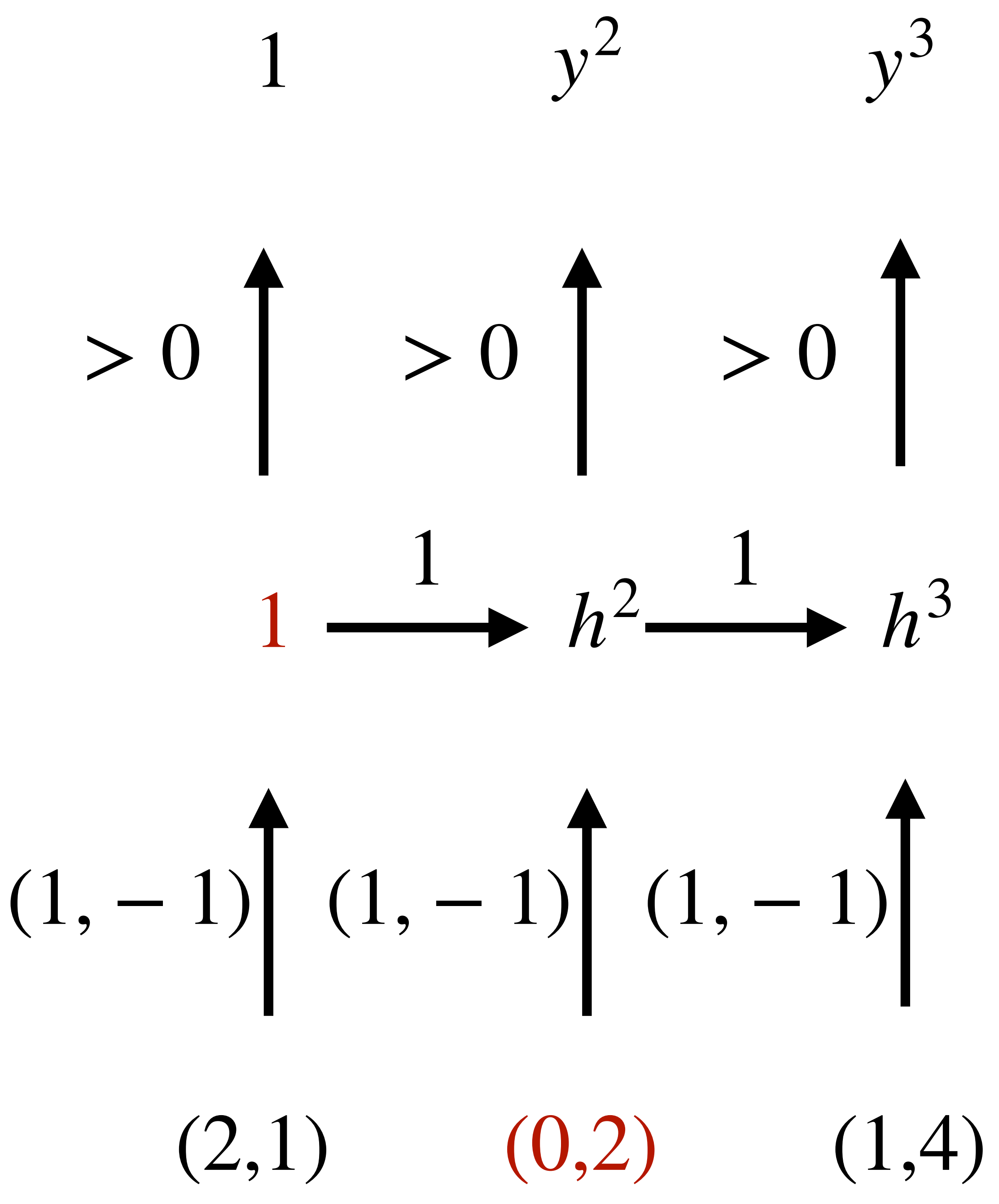
Input sequence $x^1, x^2, \ldots, x^t, \ldots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$

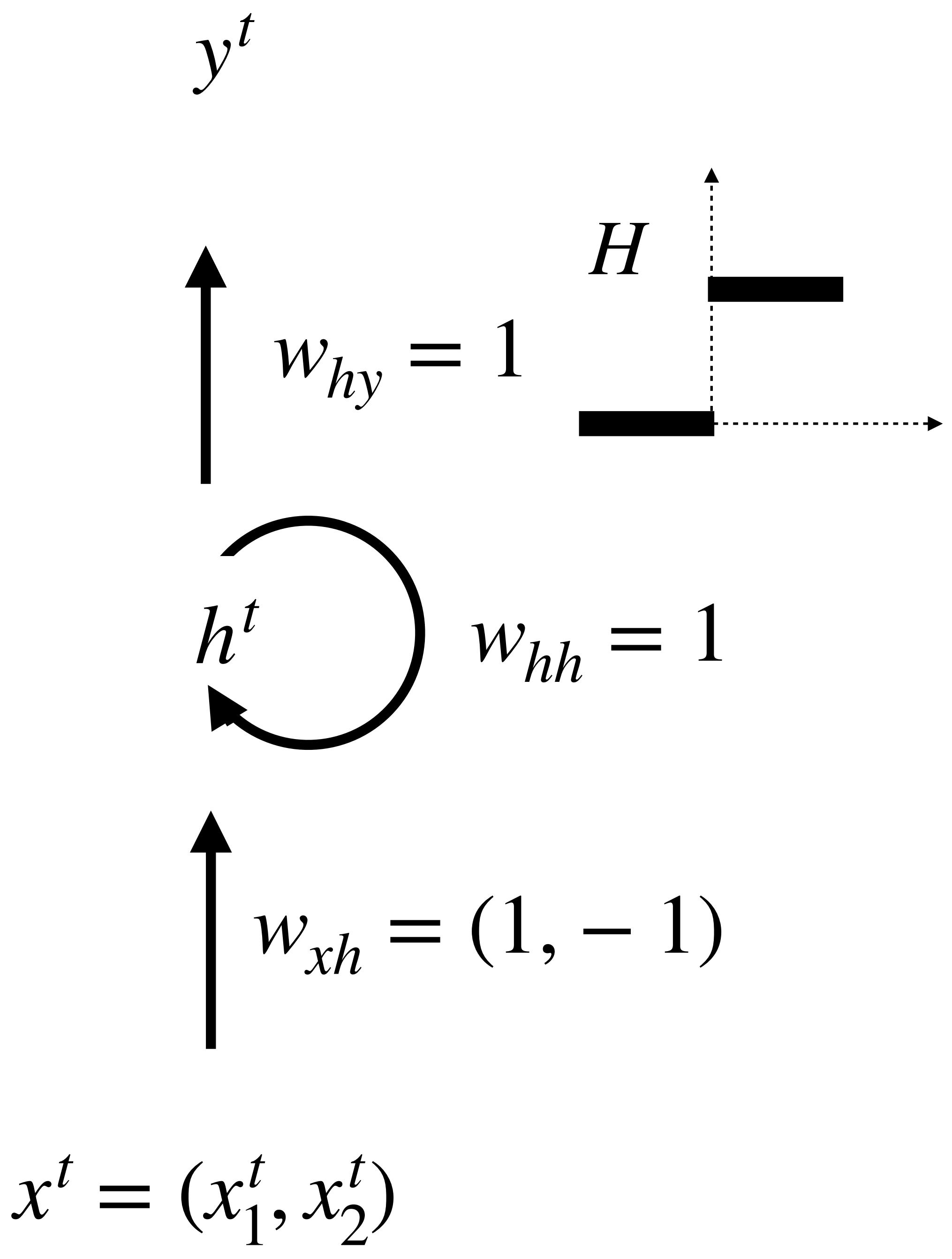


Forward Pass

What does this RNN do?

Input sequence $x^1, x^2, \ldots, x^t, \ldots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$

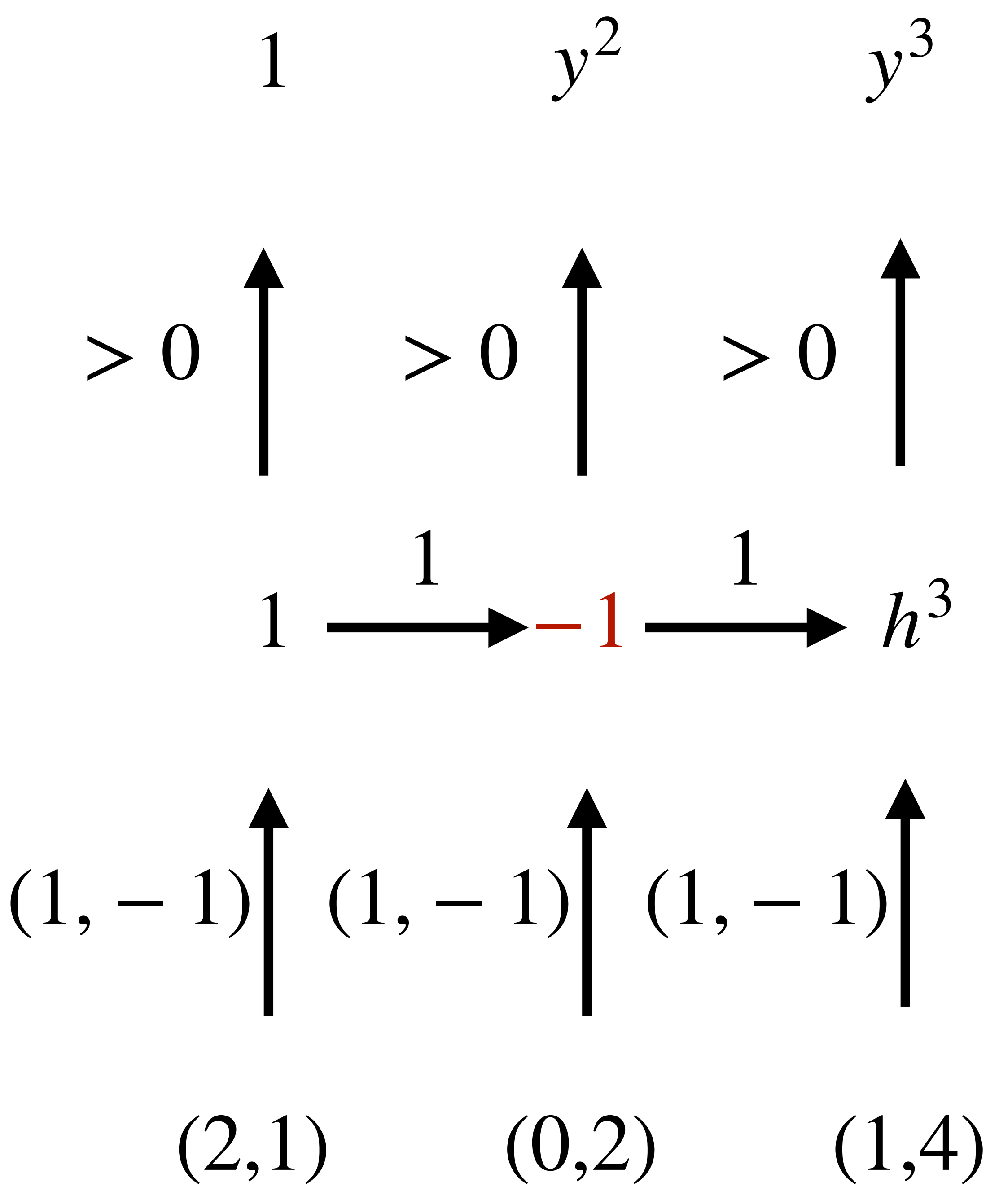


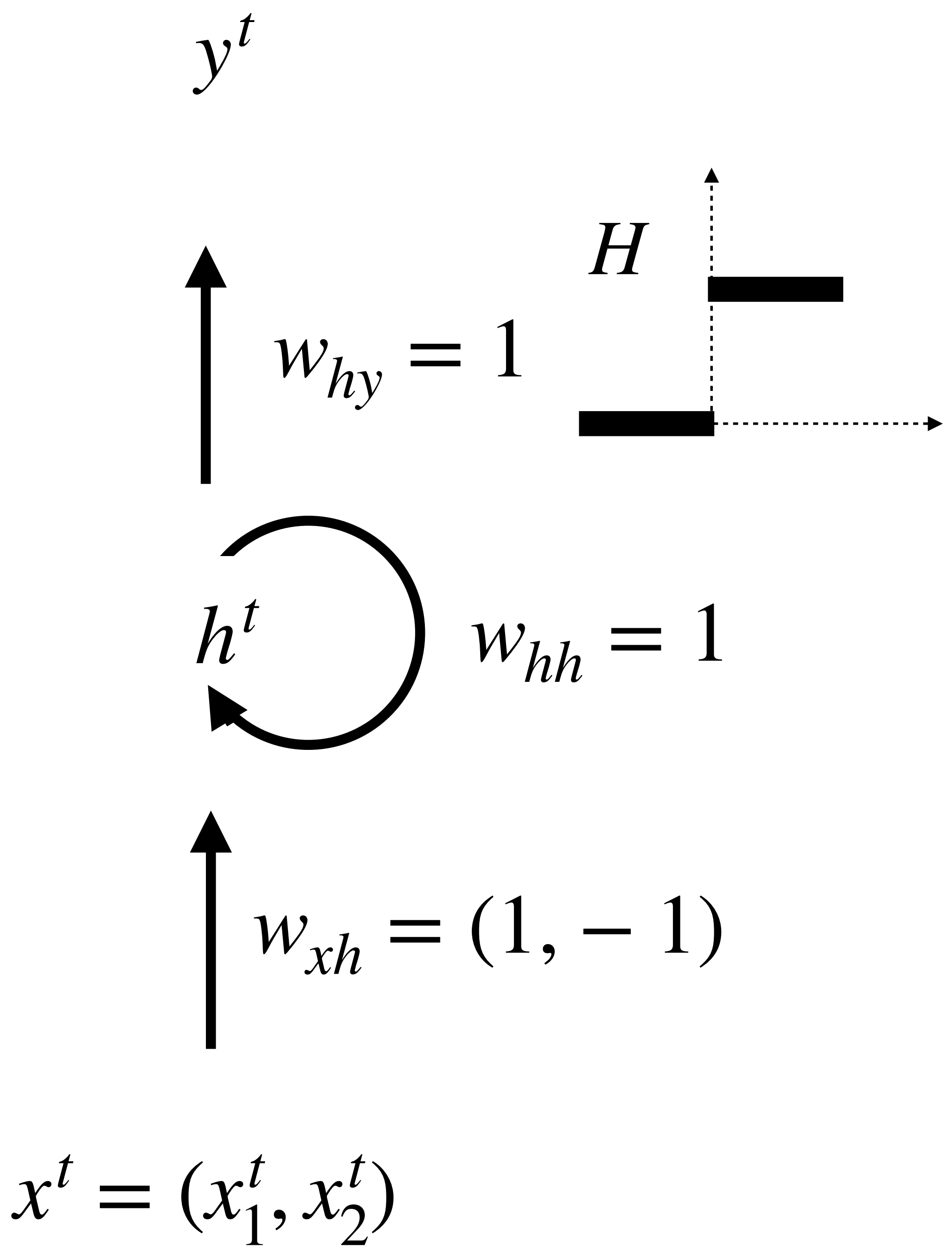


Forward Pass

What does this RNN do?

Input sequence $x^1, x^2, \dots, x^t, \dots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$

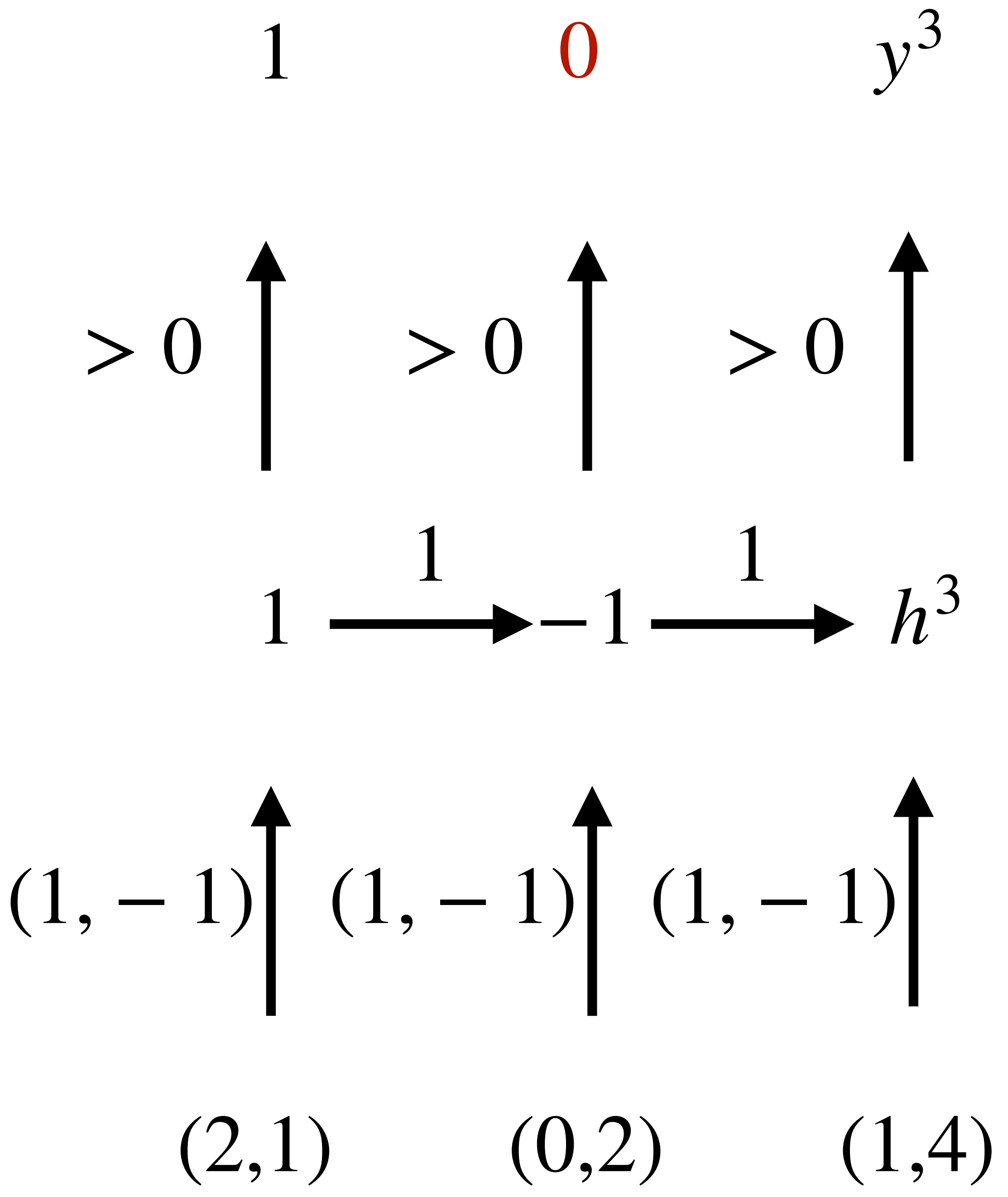


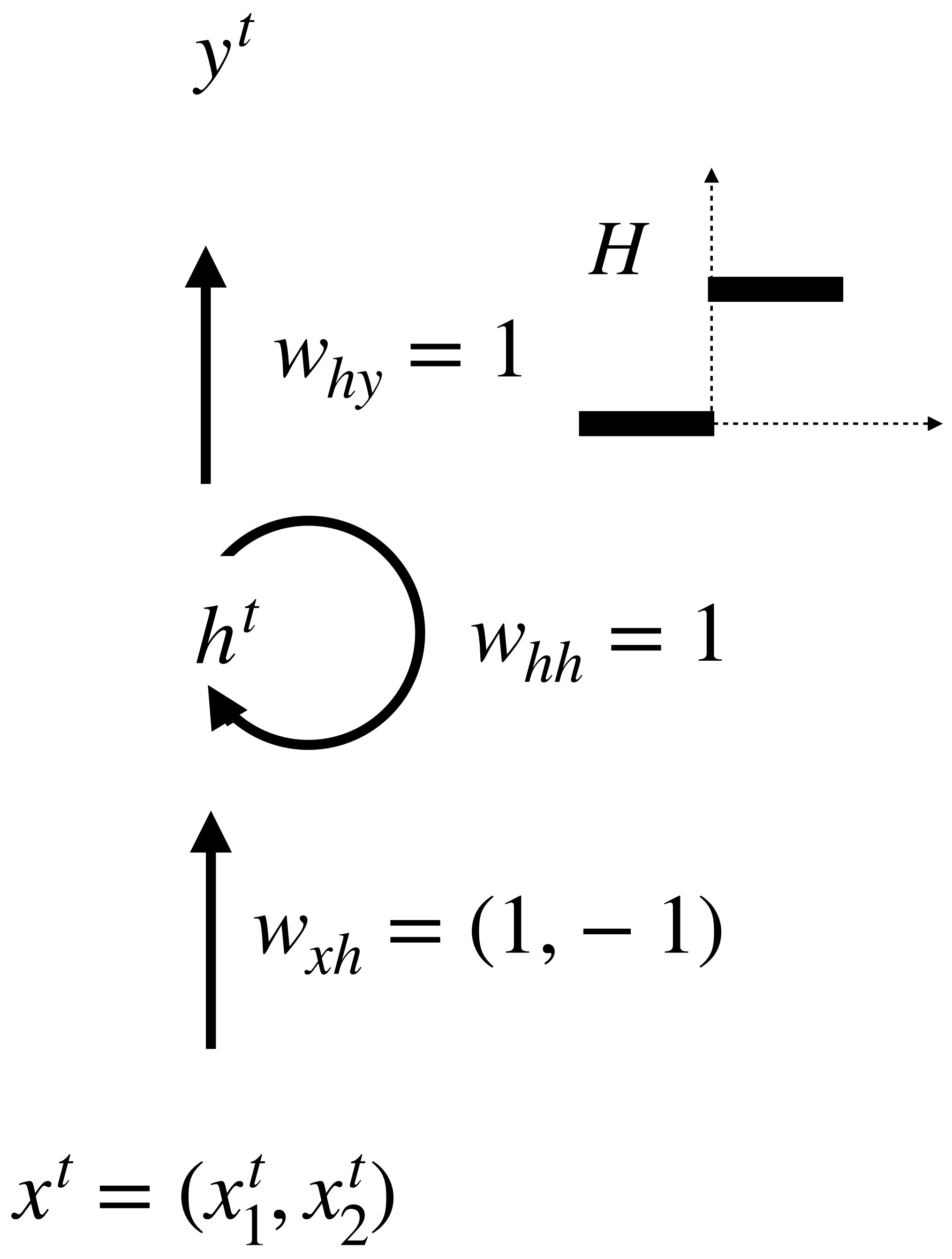


Forward Pass

What does this RNN do?

Input sequence $x^1, x^2, \ldots, x^t, \ldots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$

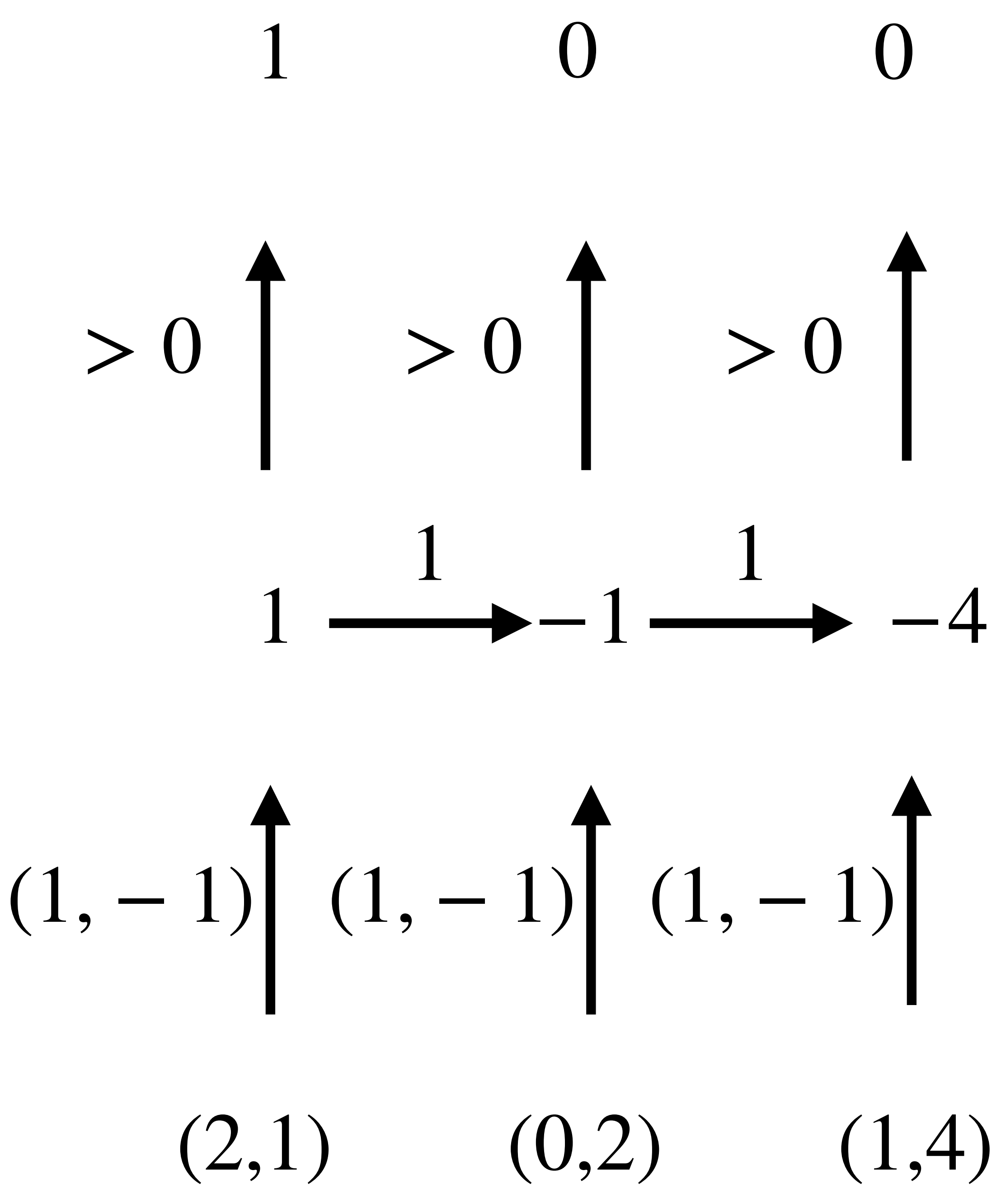


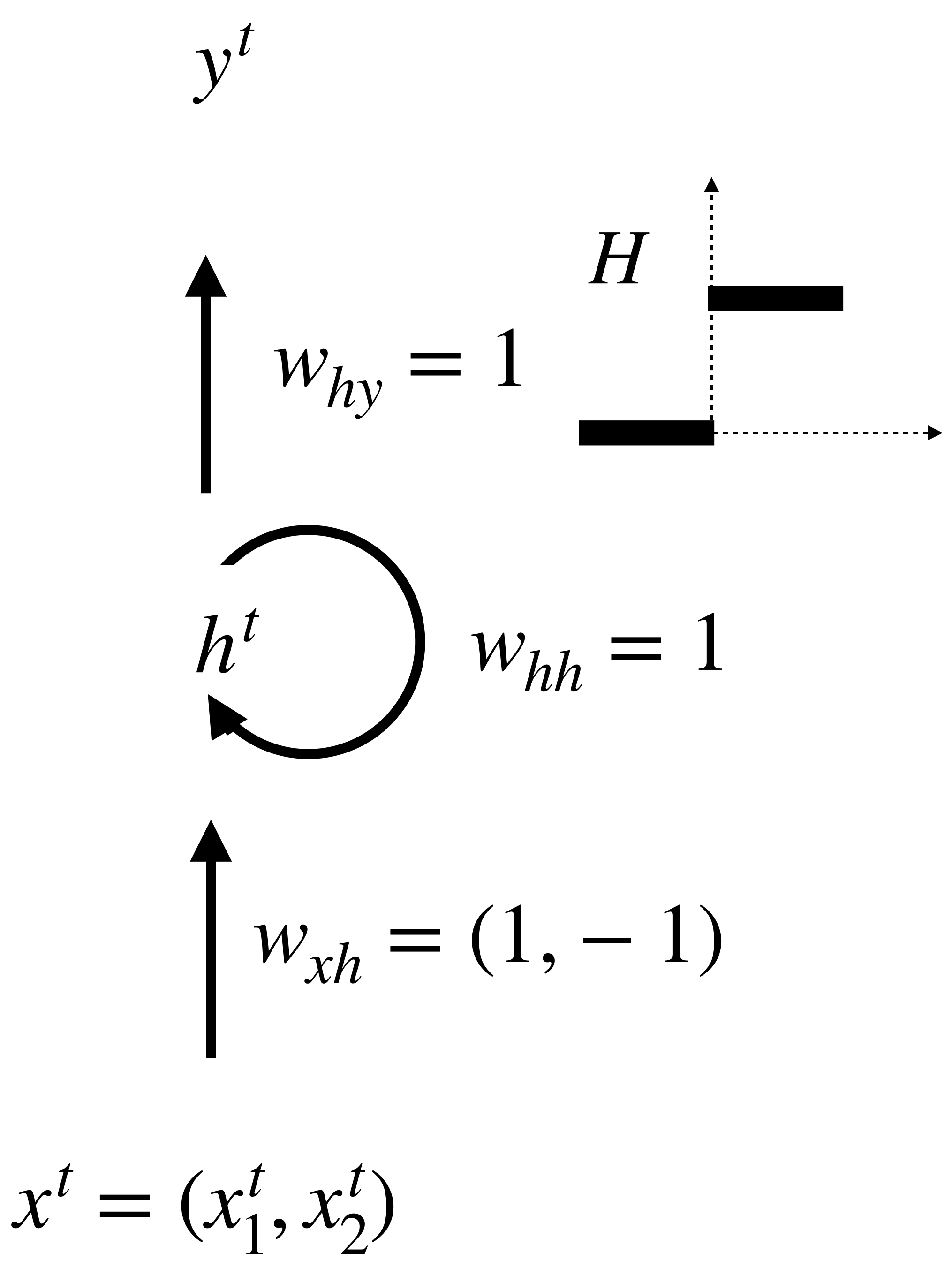


Forward Pass

What does this RNN do?

Input sequence $x^1, x^2, \dots, x^t, \dots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$

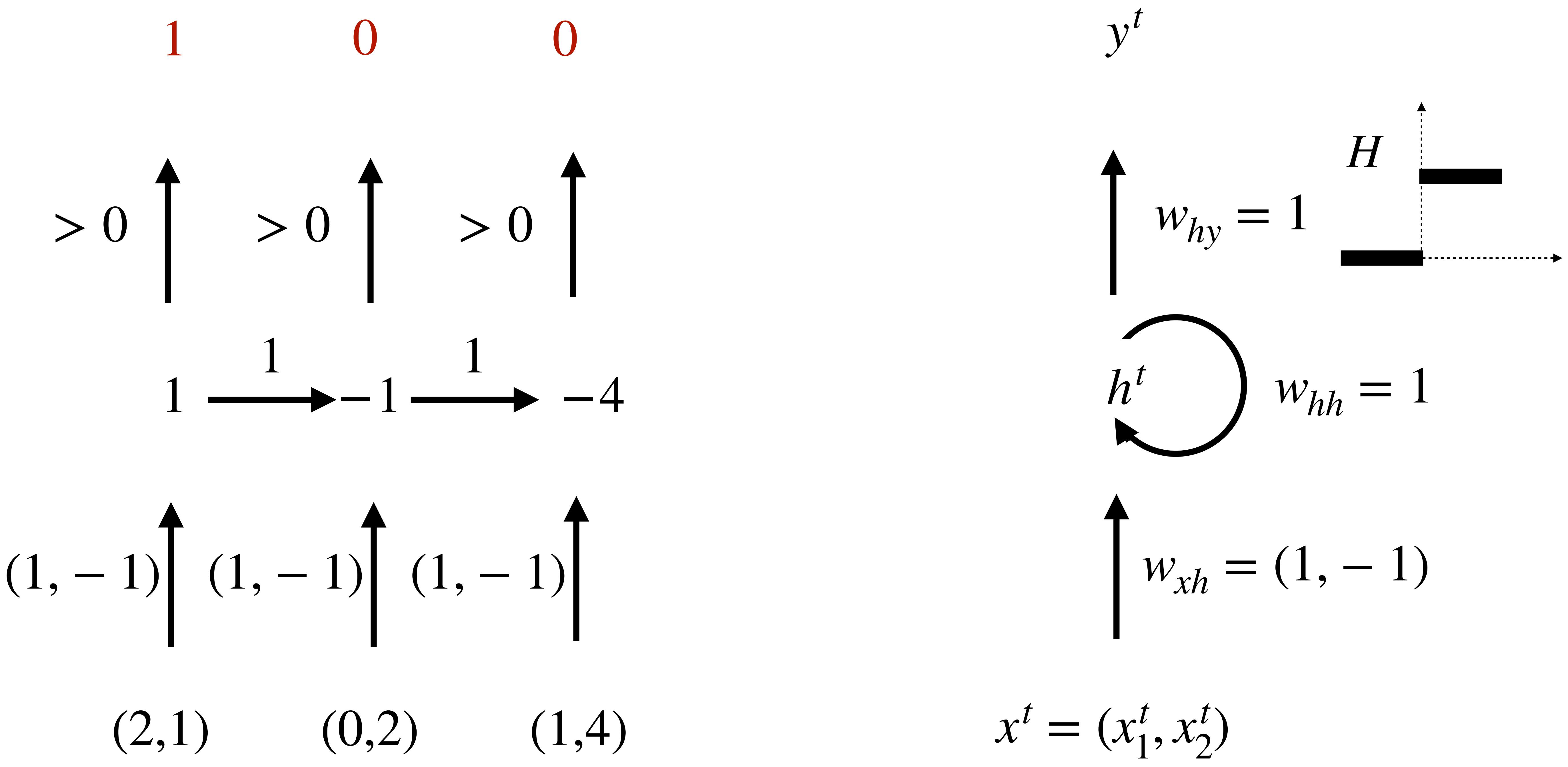




Forward Pass

What does this RNN do?

Input sequence $x^1, x^2, \ldots, x^t, \ldots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$

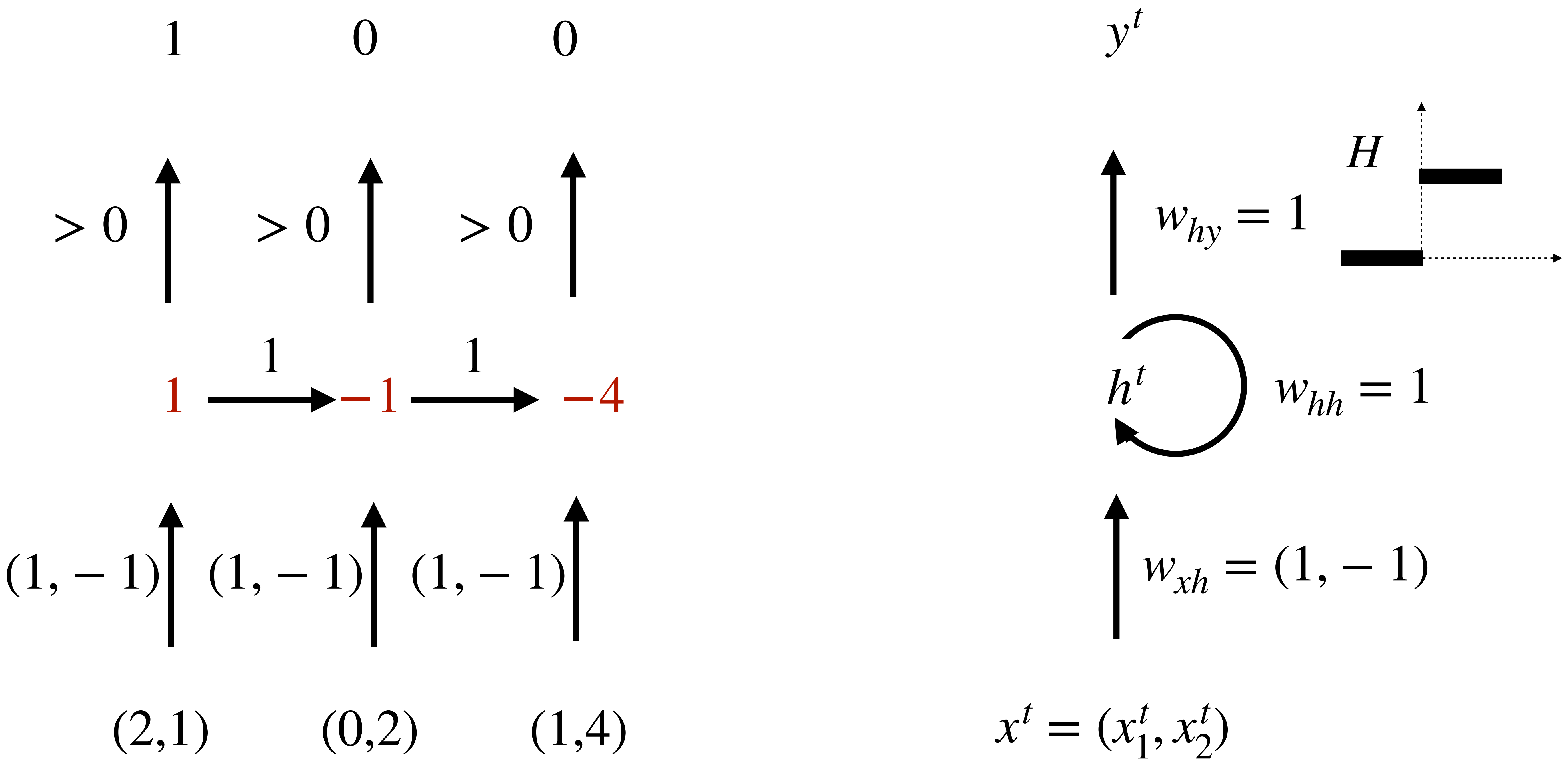


The network determines if the total sum of the first feature values is greater than the sum of the second feature values

Forward Pass

What does this RNN do?

Input sequence $x^1, x^2, \ldots, x^t, \ldots$ Each input element of the sequence $x^t = (x_1^t, x_2^t) \in \mathbb{R}^2$



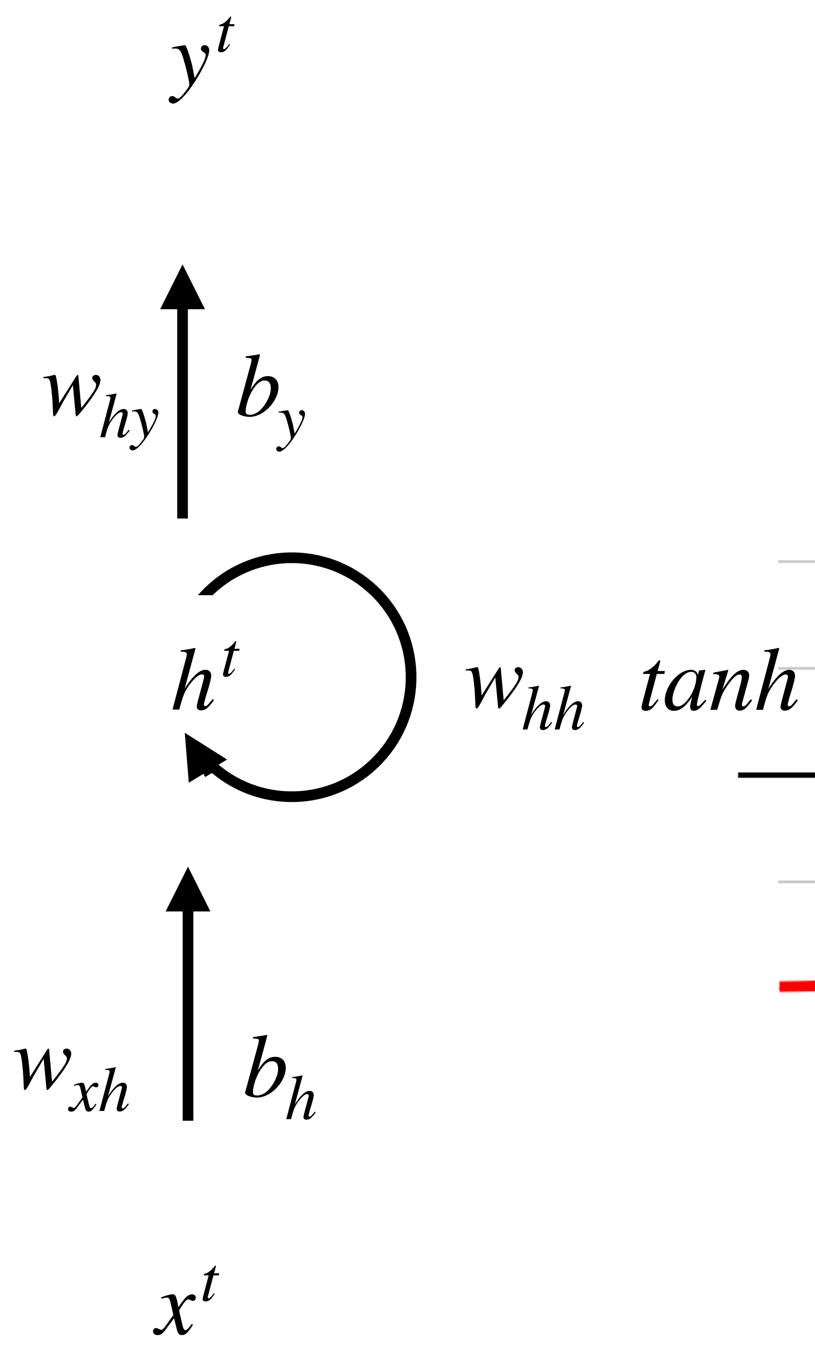
Hidden state keeps track of the total difference between the first and second input feature

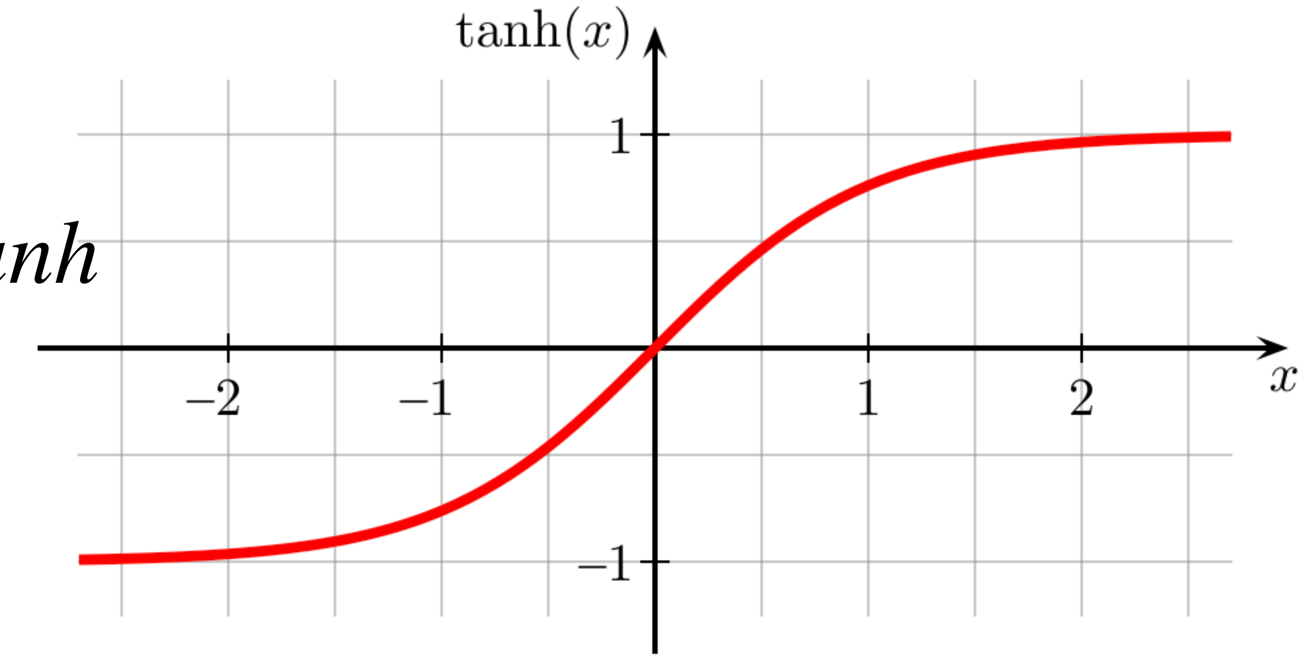
Forward Pass Equations

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\, h^t + b_y$$





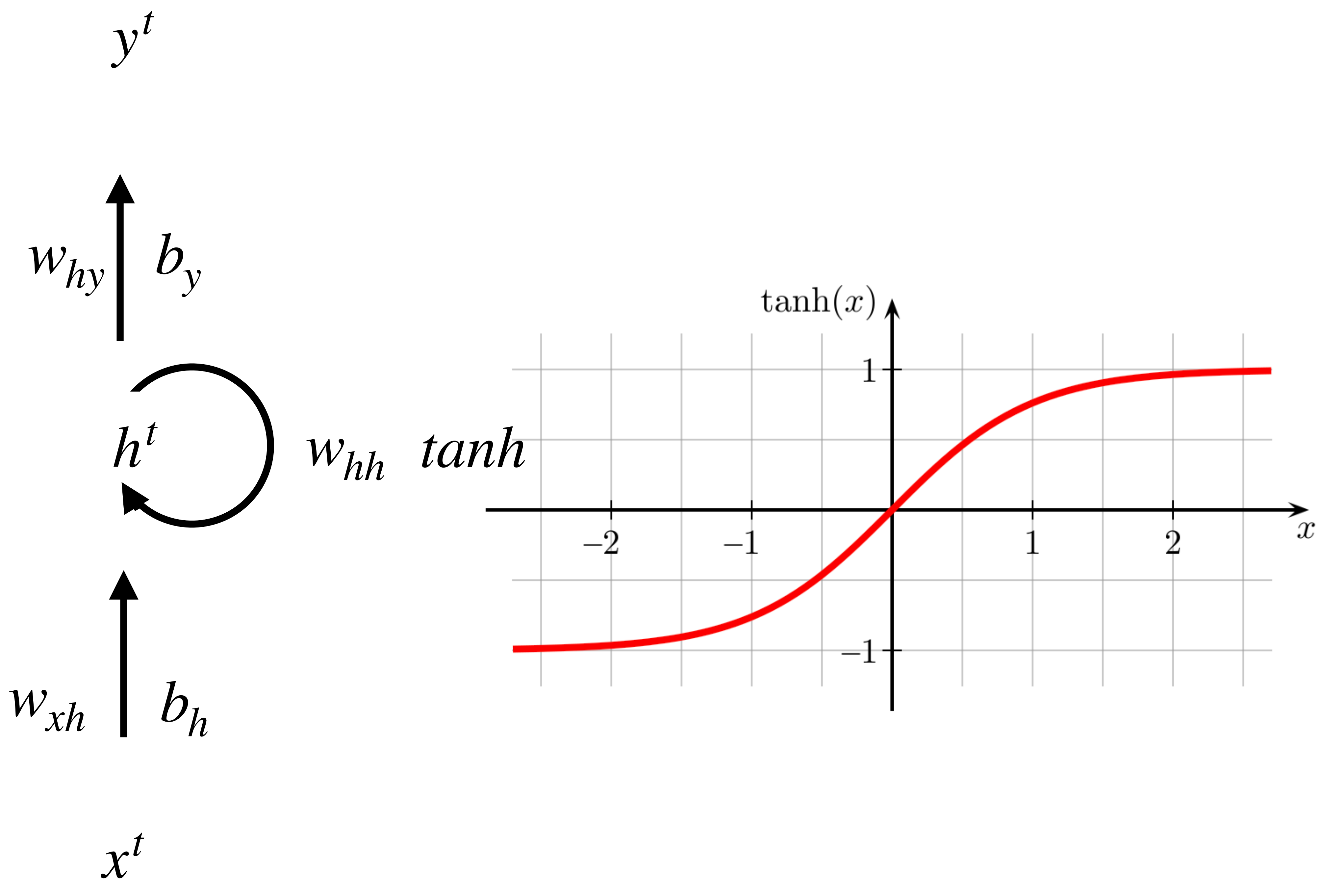
Forward Pass Equations

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\, h^t + b_y$$

Loss function $L = \dfrac{1}{T}\sum_{t=1}^{T} L^t$



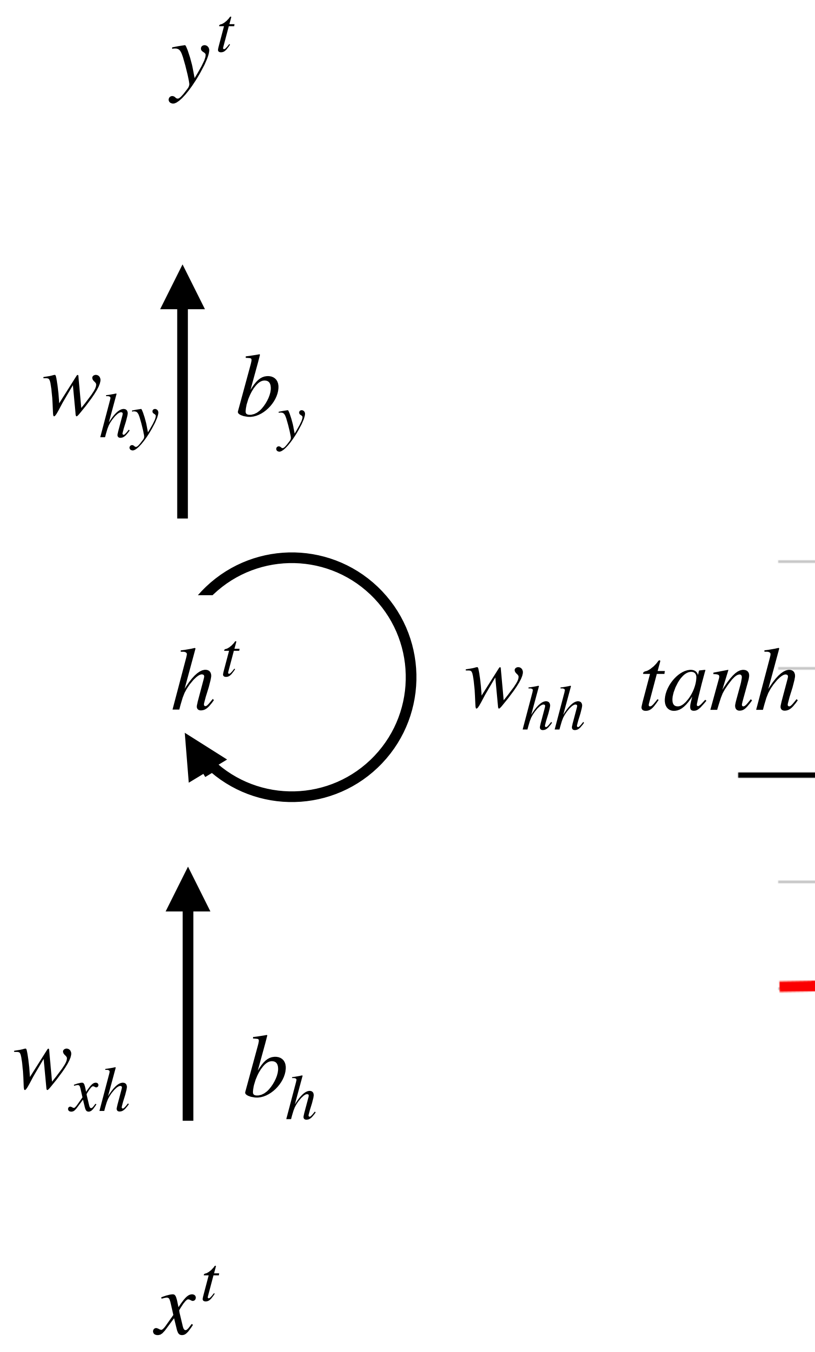
Forward Pass Equations

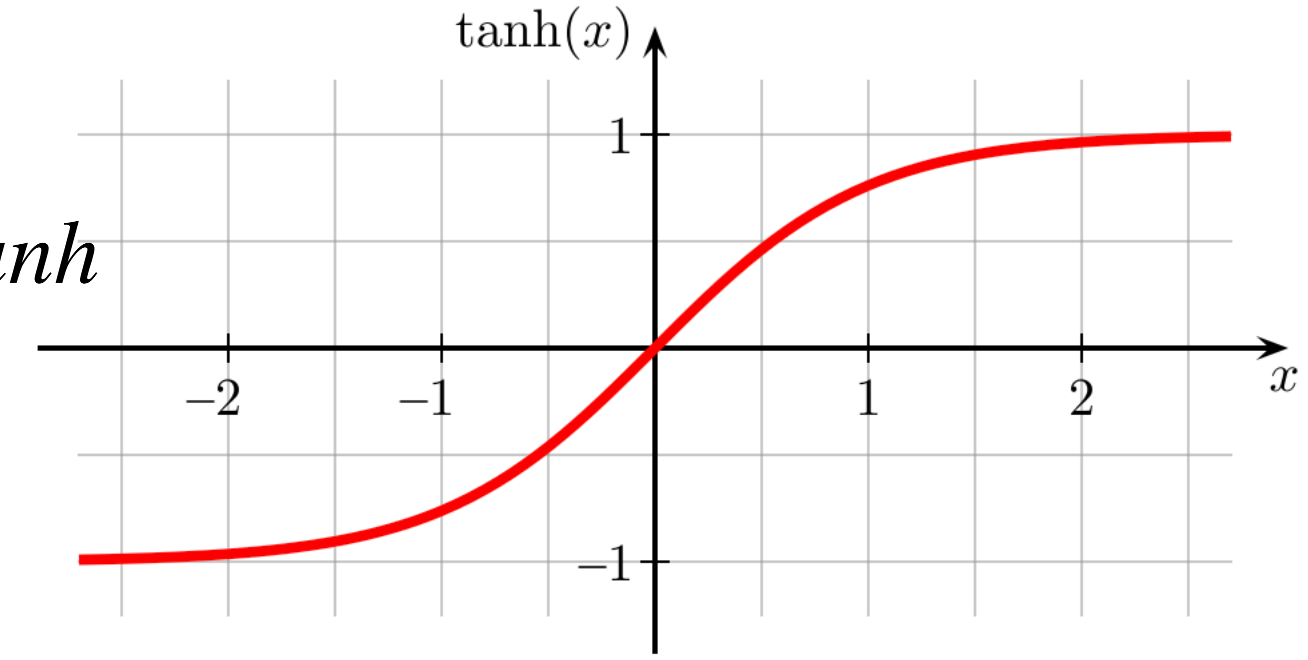
Vanilla RNN

$$h^t = \tanh\left(w_{xh}\,x^t + w_{hh}\,h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\,h^t + b_y$$

Loss function $L = \frac{1}{T}\sum_{t=1}^{T} L^t$





Training sequence x^{1995} x^{1996} $\ldots$ x^{2022}

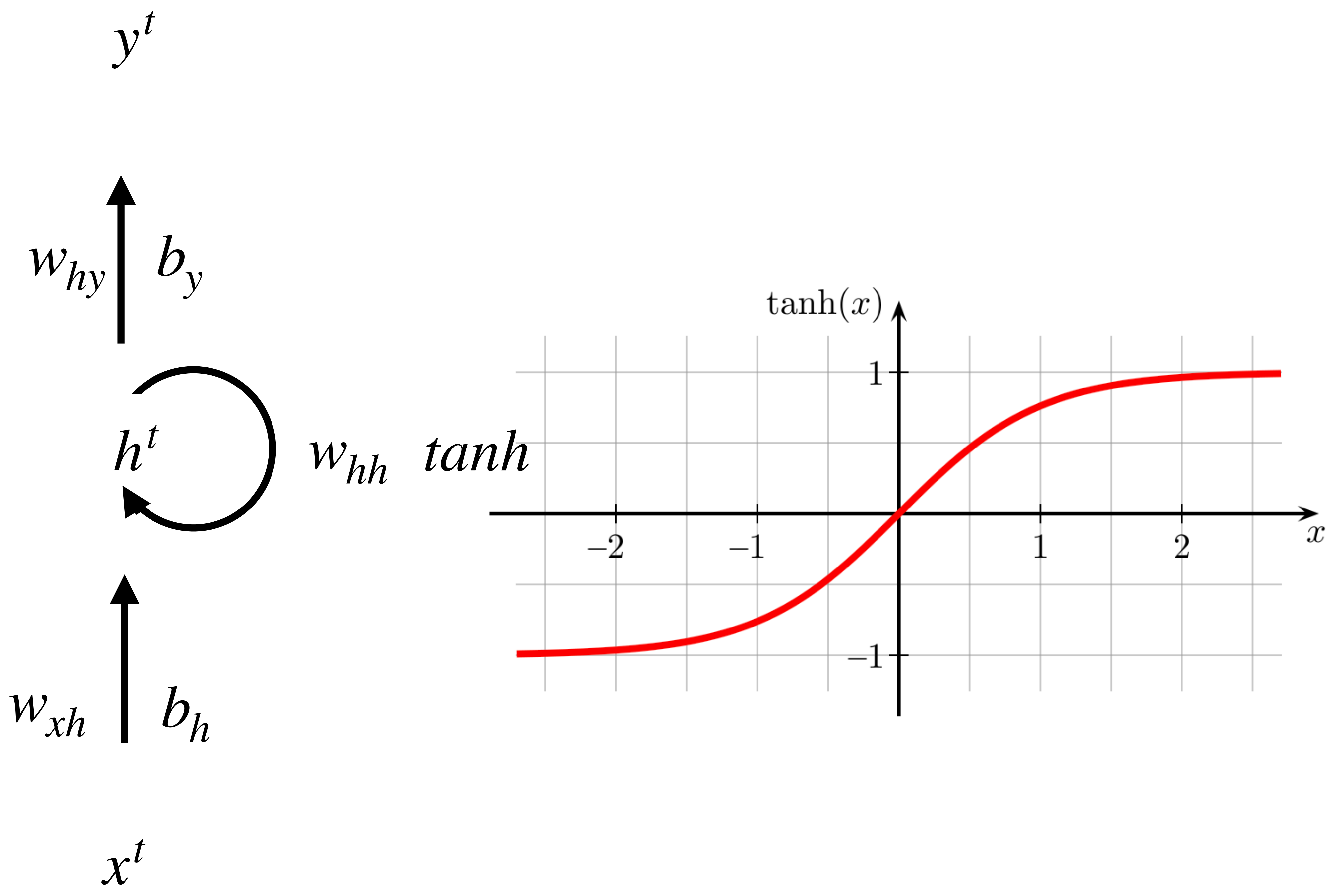
Forward Pass Equations

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\, h^t + b_y$$

Loss function $L = \dfrac{1}{T}\sum_{t=1}^{T} L^t$



What are these? → $\hat{y}^{1995}$ $\hat{y}^{1996}$ $\dots$ $\hat{y}^{2022}$

$\uparrow$ $\uparrow$ $\uparrow$

Training sequence x^{1995} x^{1996} $\dots$ x^{2022}

Forward Pass Equations

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

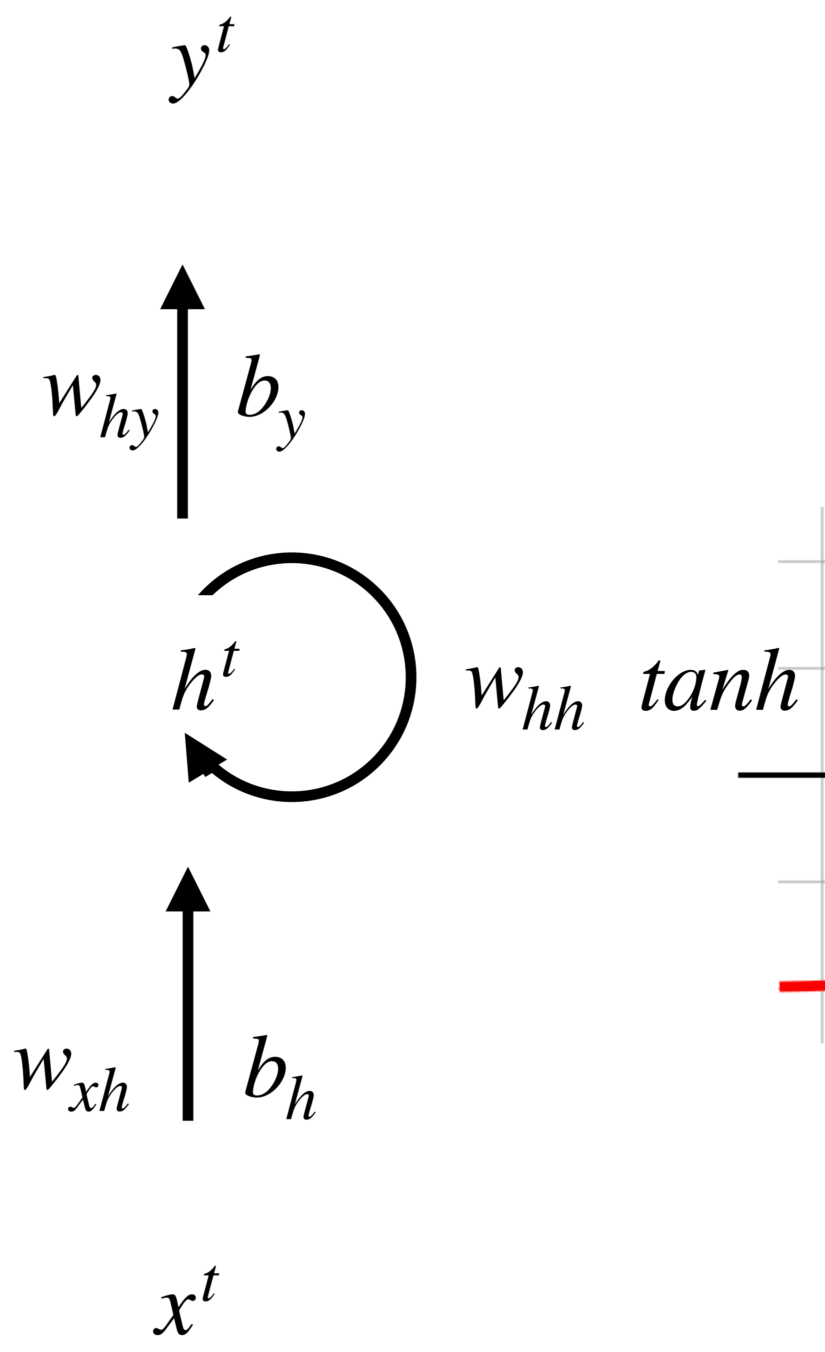
$$y^t = w_{hy}\, h^t + b_y$$

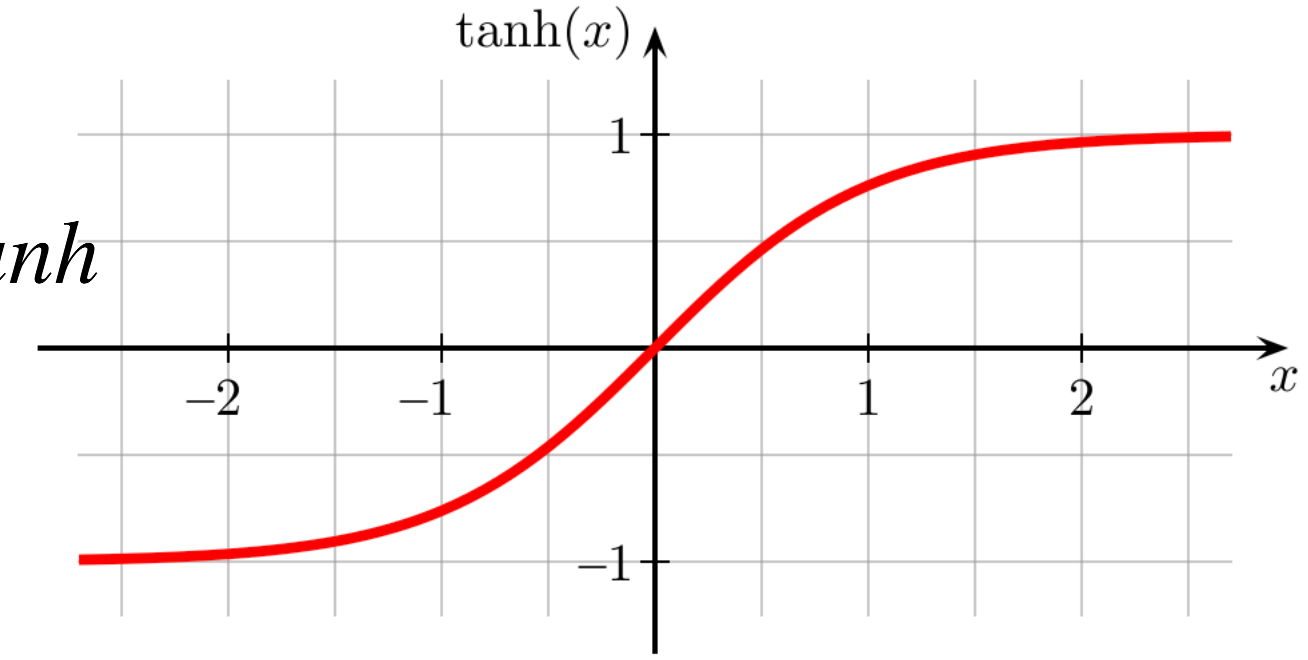
Loss function $L = \dfrac{1}{T}\displaystyle\sum_{t=1}^{T} L^t$

$\hat{x}^{1996}$ $\hat{x}^{1997}$ $\dots$ $\hat{x}^{2023}$

$\uparrow$ $\uparrow$ $\uparrow$

Training sequence x^{1995} x^{1996} $\dots$ x^{2022}





Forward Pass Equations

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\,x^t + w_{hh}\,h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\,h^t + b_y$$

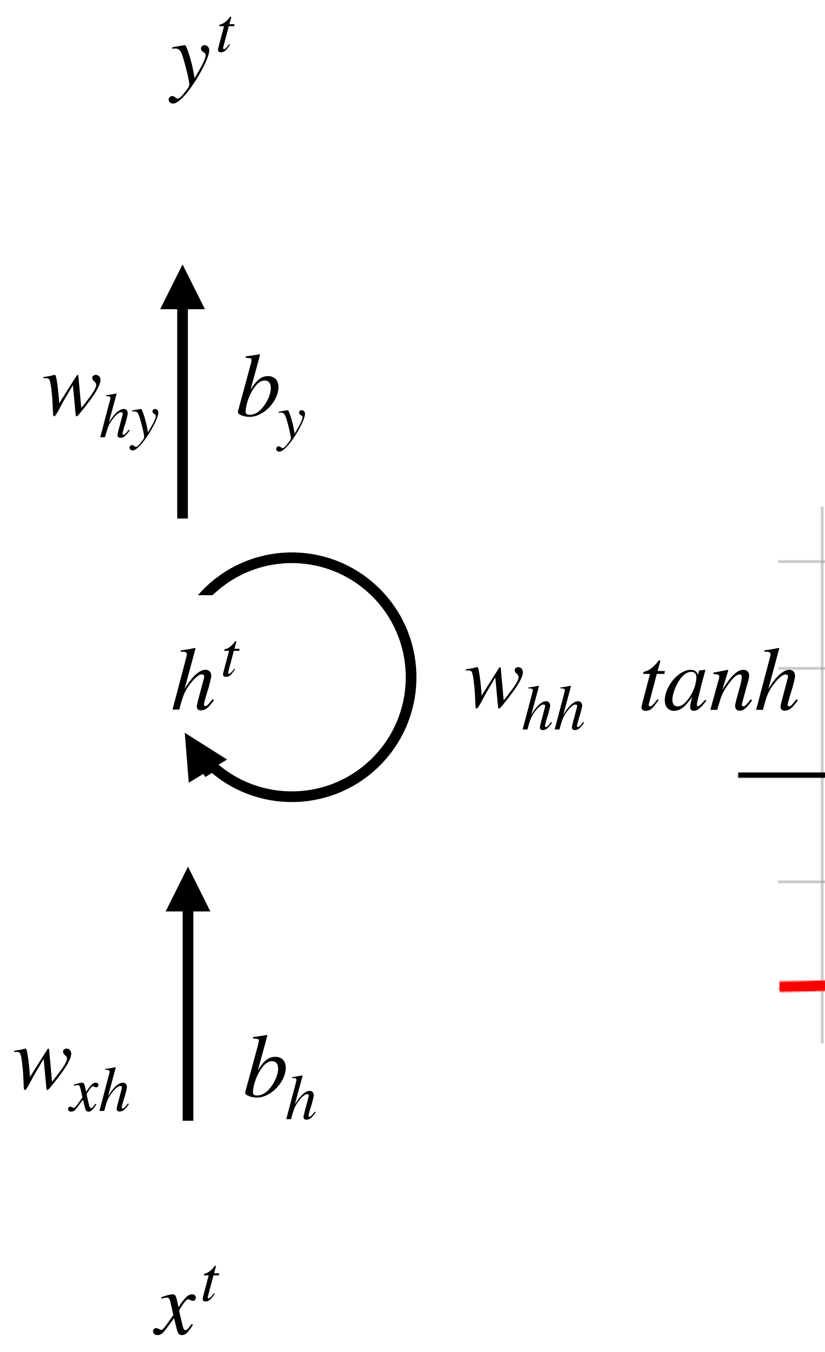
Loss function $L = \dfrac{1}{T}\sum_{t=1}^{T} L^t$

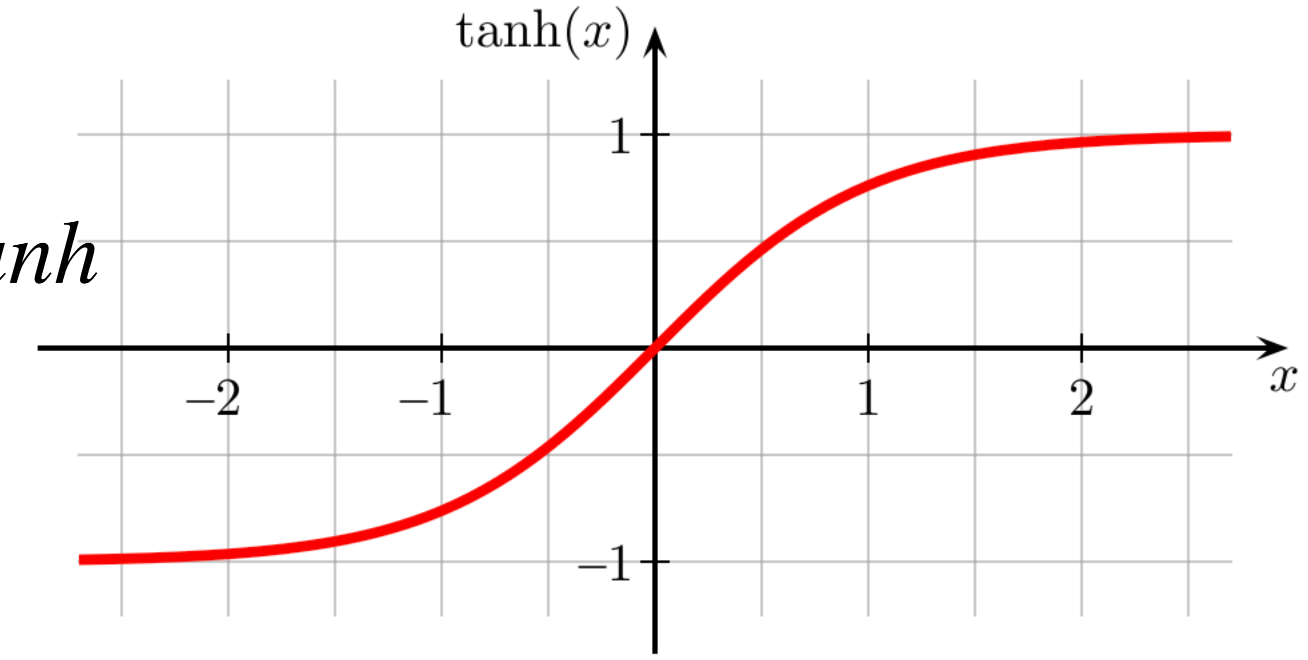
$$L^{1996} = \frac{1}{2}(x^{1996} - \hat{x}^{1996})^2$$

$\hat{x}^{1996}$ $\hat{x}^{1997}$ $\ldots$ $\hat{x}^{2023}$

$\uparrow$ $\uparrow$ $\uparrow$

Training sequence x^{1995} x^{1996} $\ldots$ x^{2022}





Forward Pass Equations

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\, h^t + b_y$$

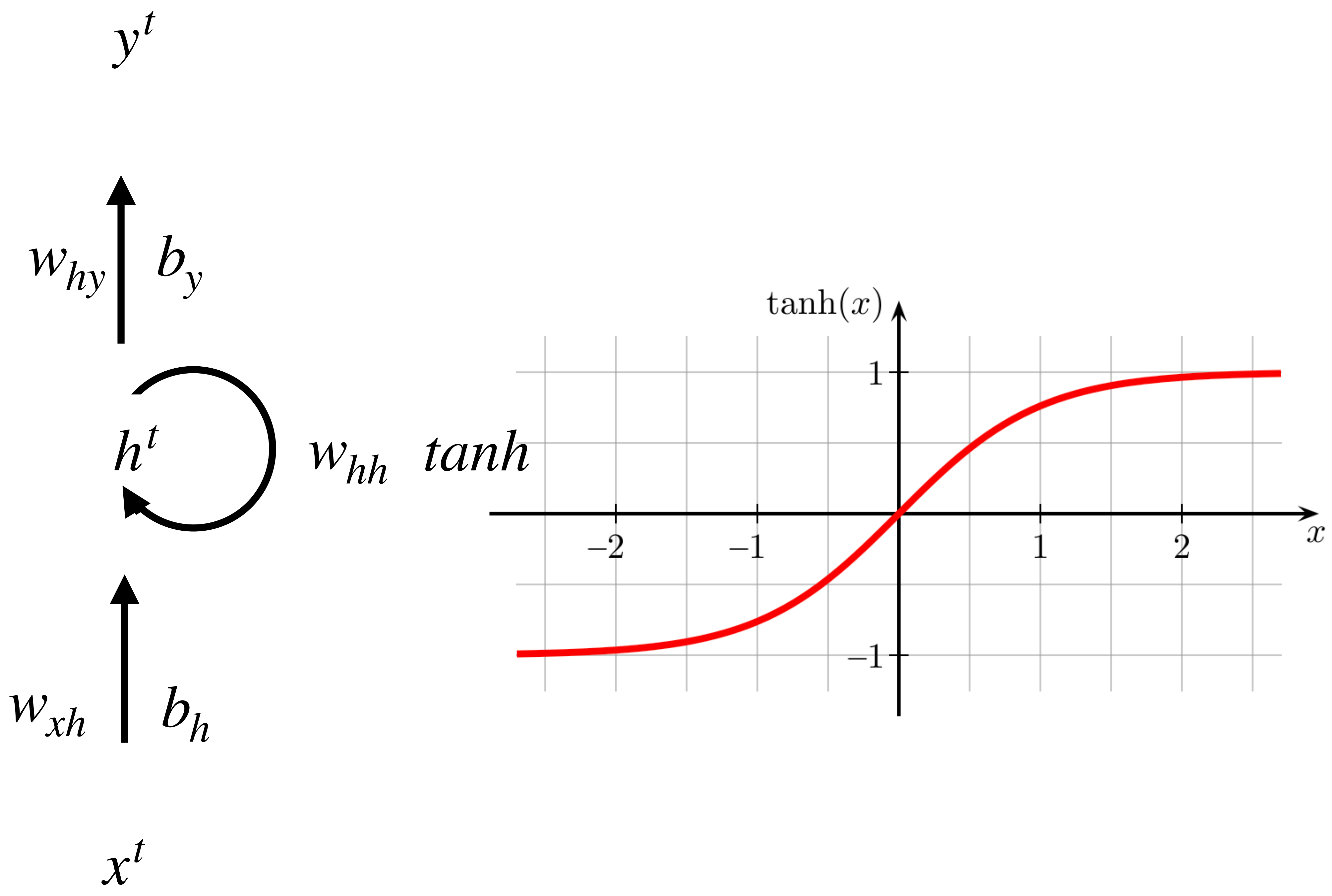
Loss function $L = \dfrac{1}{18} \sum_{t=1996}^{2022} L^t$

$L^{1996} \quad L^{1997} \quad \dots$

$\hat{x}^{1996} \quad \hat{x}^{1997} \quad \dots \quad \hat{x}^{2023}$

$\uparrow \quad \uparrow \quad \quad \uparrow$

Training sequence $x^{1995} \quad x^{1996} \quad \dots \quad x^{2022}$



Backpropagation Through Time

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\, h^t + b_y$$

Loss function $L = \frac{1}{T}\sum_{t=1}^{T} L^t \qquad \frac{\partial L}{\partial w} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial L^t}{\partial w}$

Unrolled computation graph

$$y^1 \longrightarrow L^1$$

$$y^2 \longrightarrow L^2$$

$$y^3 \longrightarrow L^3$$

$$\vdots$$

Backpropagation Through Time

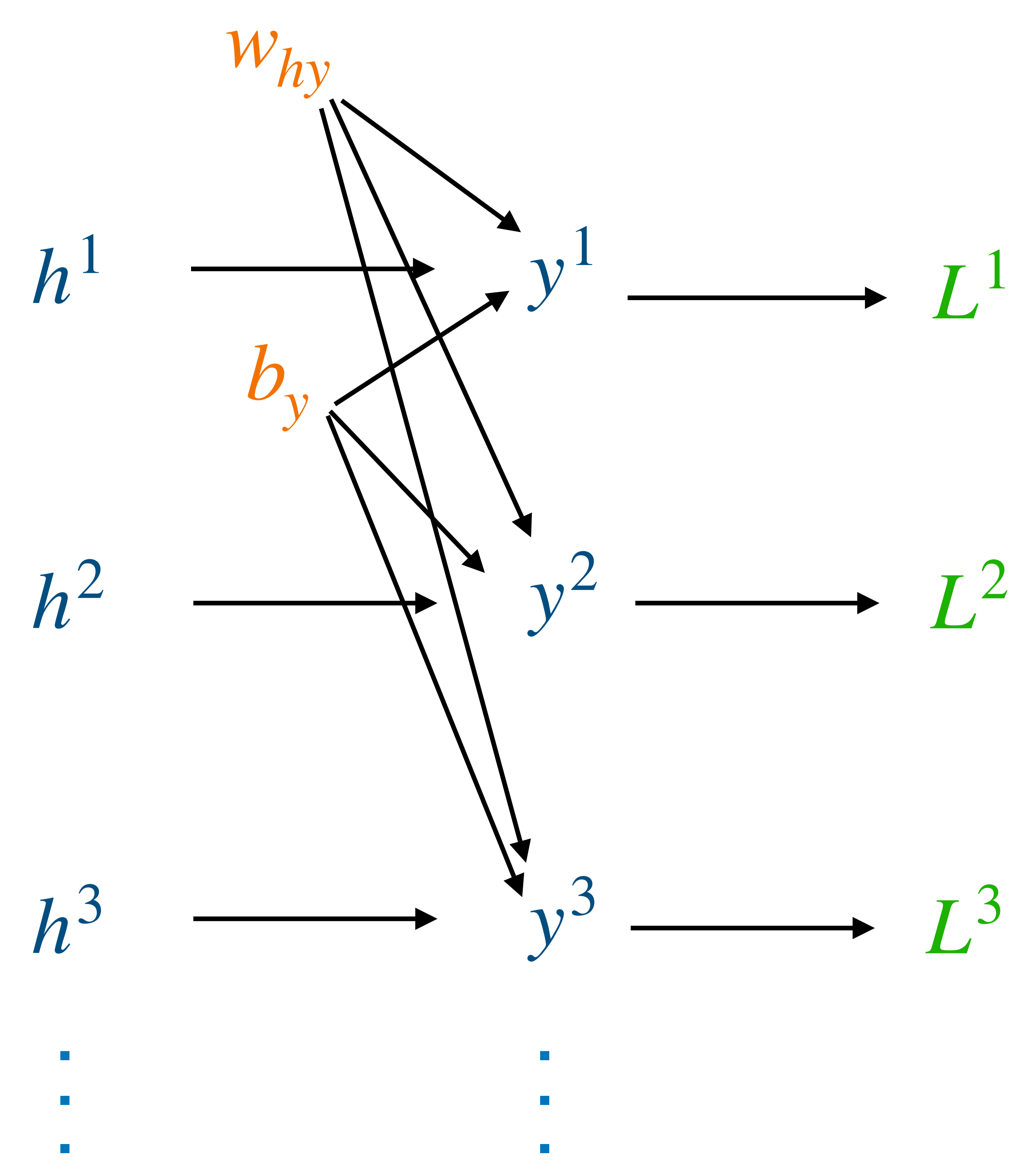
Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\, h^t + b_y$$

Loss function $L = \frac{1}{T}\sum_{t=1}^{T} L^t \qquad \frac{\partial L}{\partial w} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial L^t}{\partial w}$

Unrolled computation graph



Backpropagation Through Time

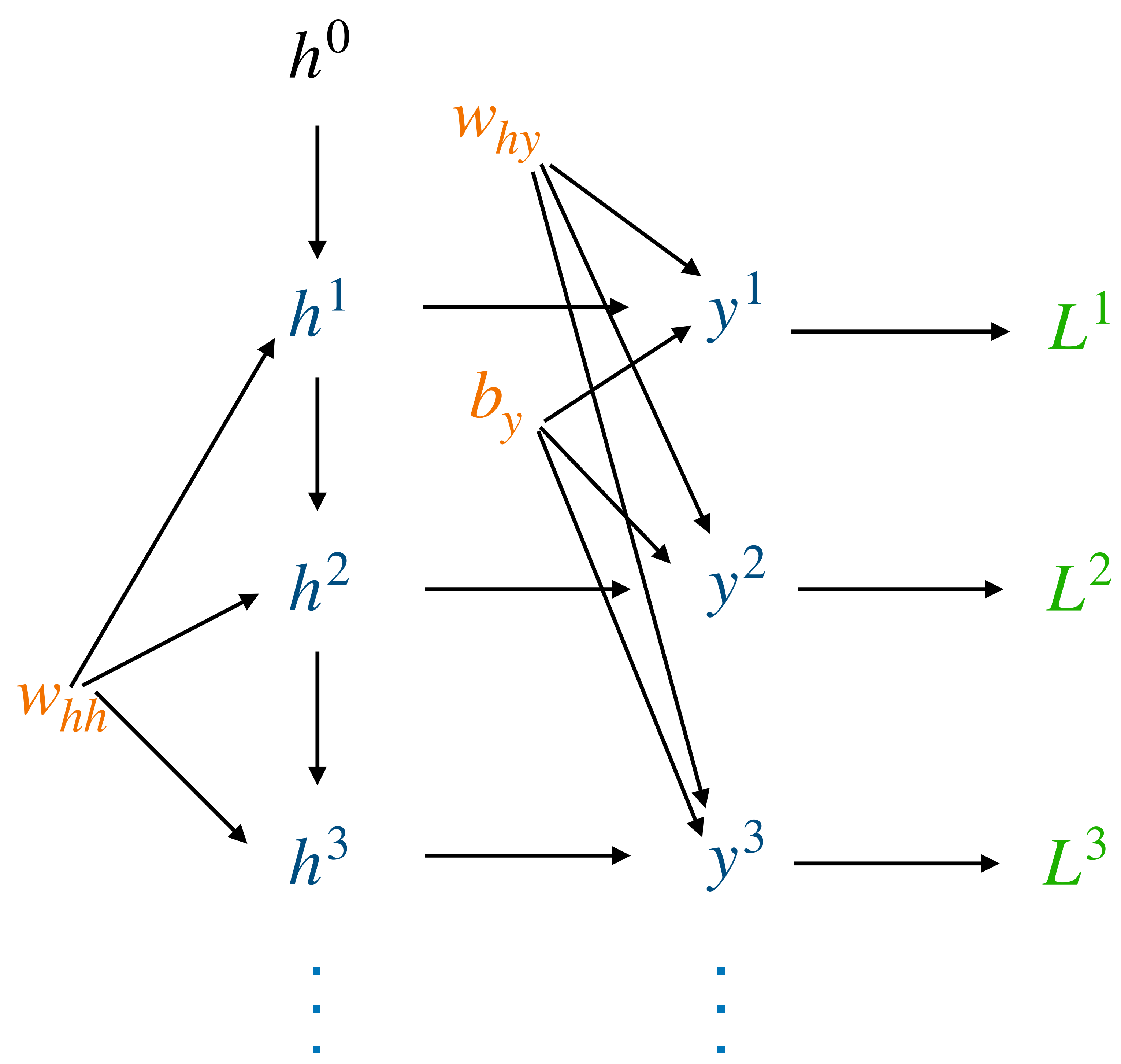
Vanilla RNN

$$h^t = \tanh\left(w_{xh}\,x^t + w_{hh}\,h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\,h^t + b_y$$

Loss function $L = \frac{1}{T}\sum_{t=1}^{T} L^t \qquad \frac{\partial L}{\partial w} = \frac{1}{T}\sum_{t=1}^{T}\frac{\partial L^t}{\partial w}$

Unrolled computation graph



Backpropagation Through Time

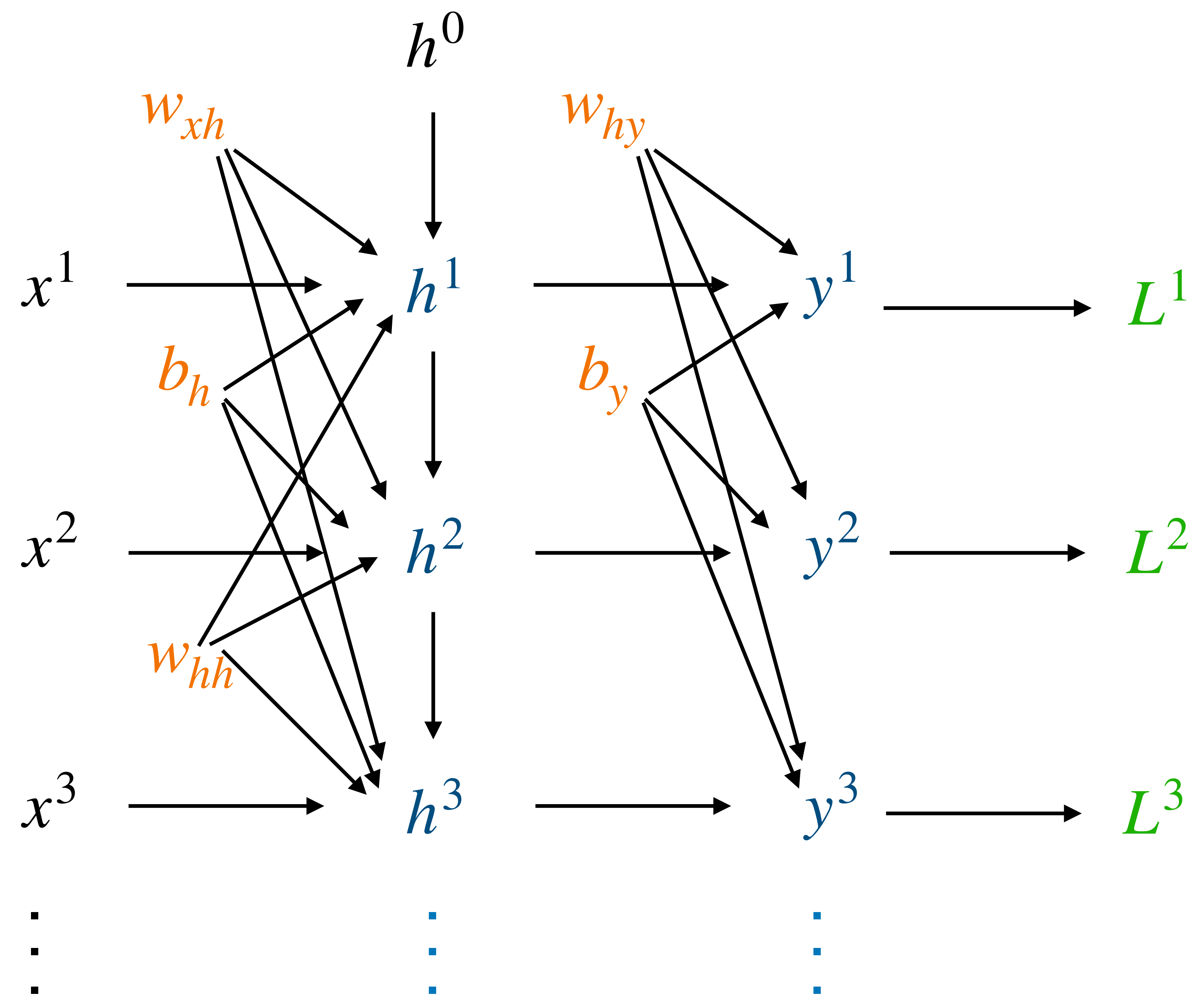
Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\, h^t + b_y$$

Loss function $L = \frac{1}{T}\sum_{t=1}^{T} L^t \qquad \frac{\partial L}{\partial w} = \frac{1}{T}\sum_{t=1}^{T}\frac{\partial L^t}{\partial w}$

Unrolled computation graph



Backpropagation Through Time

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

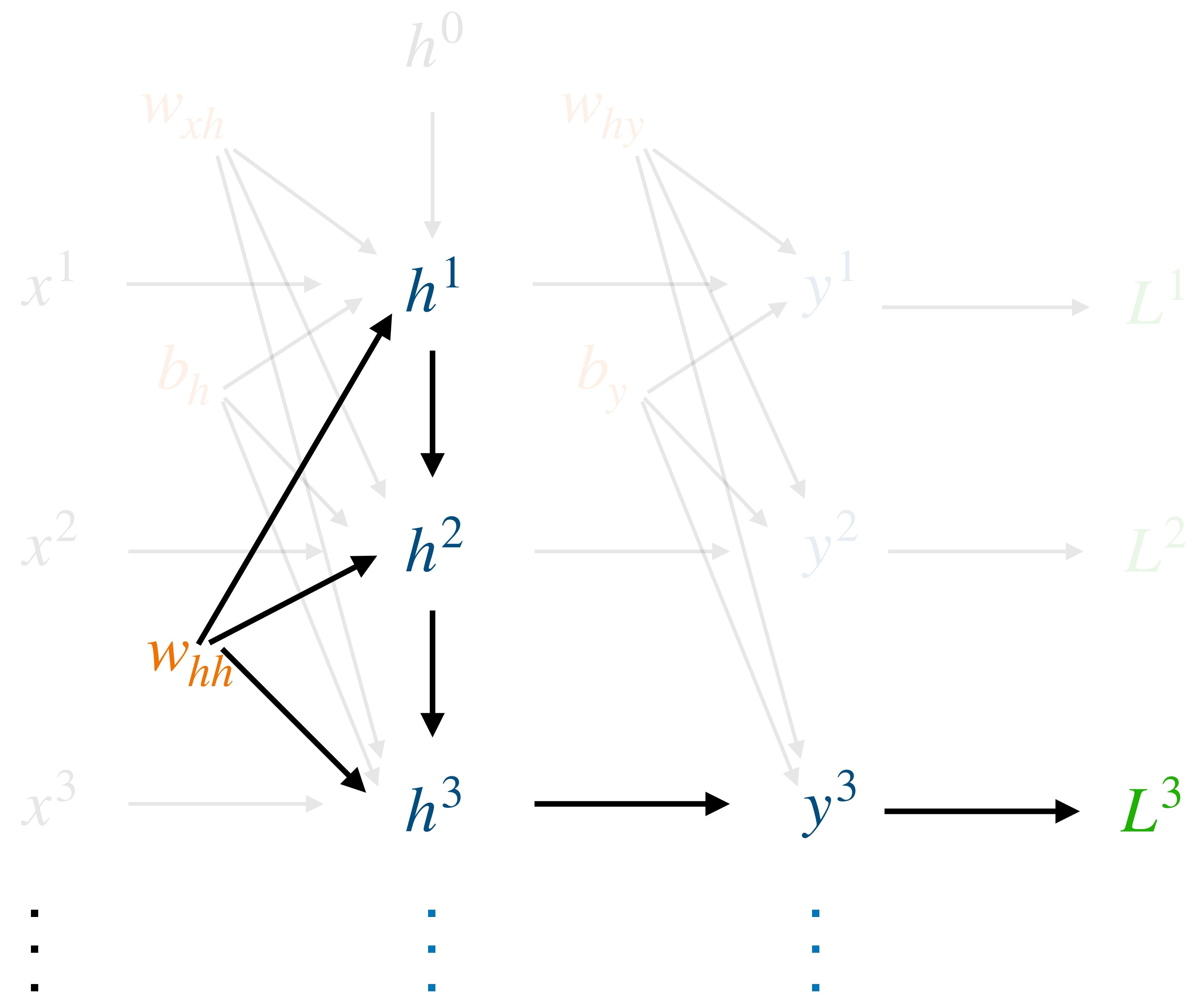
$$y^t = w_{hy}\, h^t + b_y$$

Loss function $L = \frac{1}{T}\sum_{t=1}^{T} L^t \qquad \frac{\partial L}{\partial w} = \frac{1}{T}\sum_{t=1}^{T}\frac{\partial L^t}{\partial w}$

Chain Rule

$$\frac{\partial L^3}{\partial w_{hh}} = \frac{\partial L^3}{\partial y^3}\frac{\partial y^3}{\partial h^3}\frac{\partial h^3}{\partial w_{hh}}$$

Unrolled computation graph



Backpropagation Through Time

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\, h^t + b_y$$

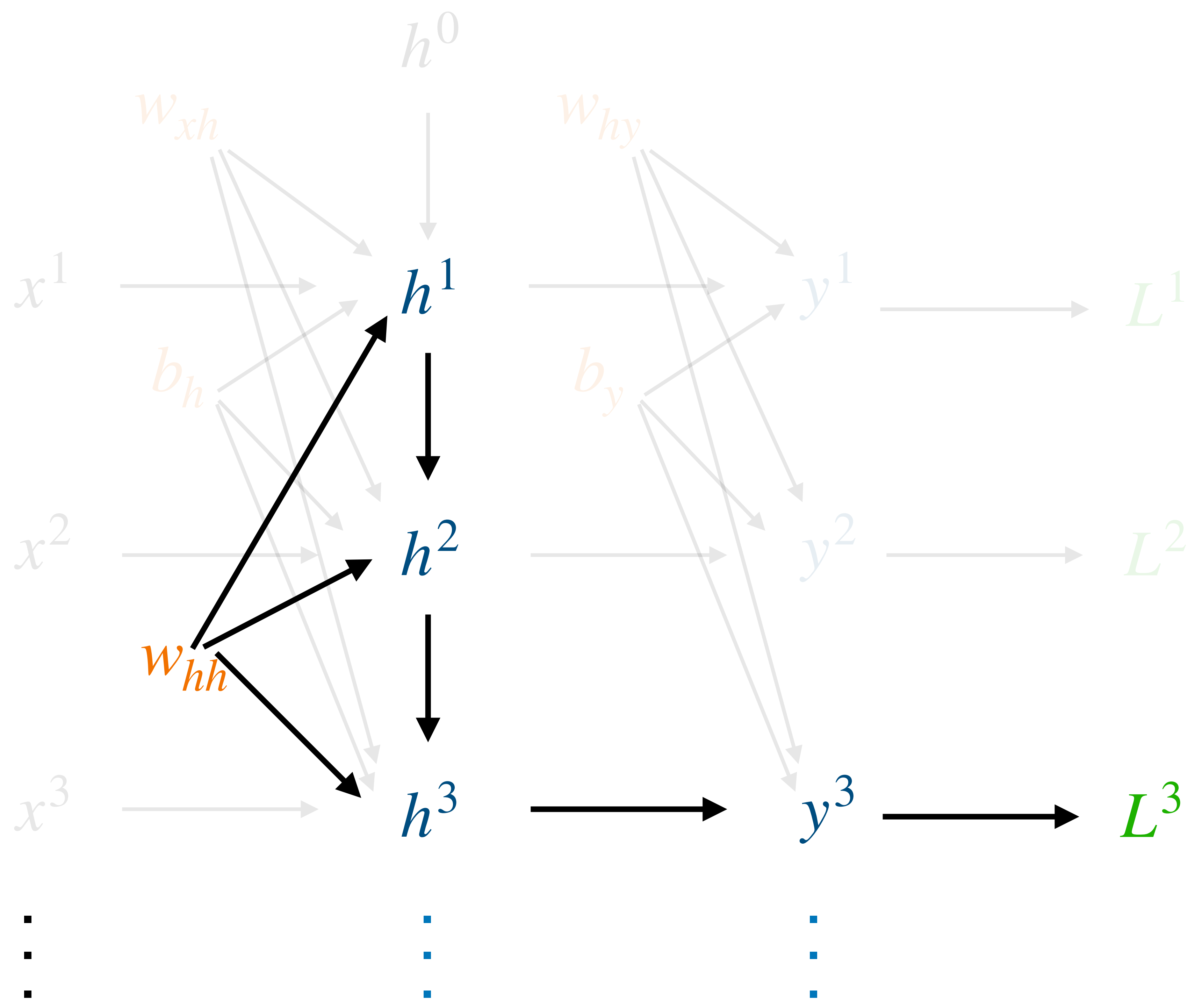
Loss function $L = \dfrac{1}{T}\sum_{t=1}^{T} L^t \qquad \dfrac{\partial L}{\partial w} = \dfrac{1}{T}\sum_{t=1}^{T} \dfrac{\partial L^t}{\partial w}$

Chain Rule

$$\frac{\partial L^3}{\partial w_{hh}} = \boxed{\frac{\partial L^3}{\partial y^3}\frac{\partial y^3}{\partial h^3}}\frac{\partial h^3}{\partial w_{hh}}$$

Nothing new from FeedForward

Unrolled computation graph



Backpropagation Through Time

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\, h^t + b_y$$

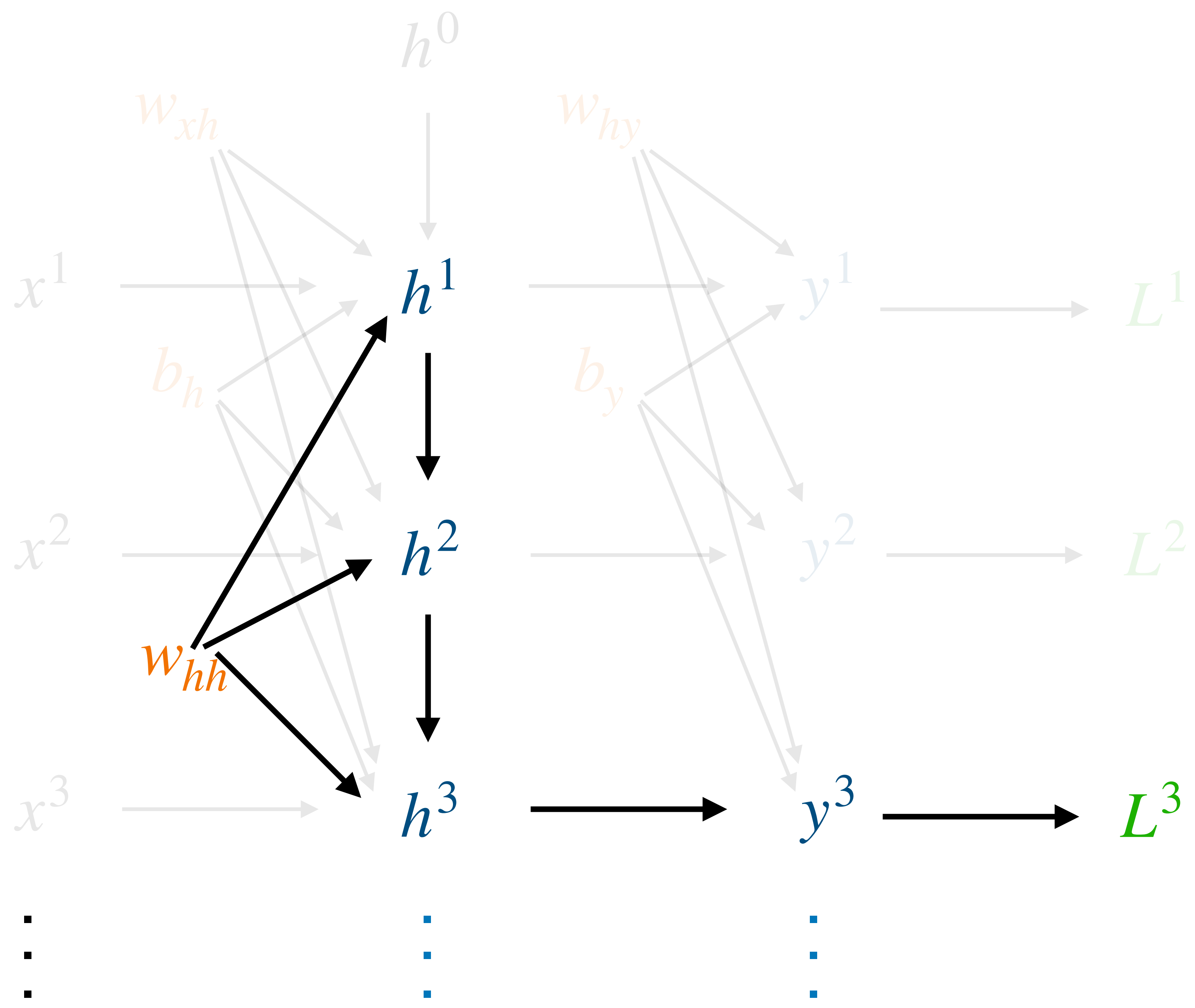
Loss function $L = \frac{1}{T}\sum_{t=1}^{T} L^t \qquad \frac{\partial L}{\partial w} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial L^t}{\partial w}$

Chain Rule

$$\frac{\partial L^3}{\partial w_{hh}} = \frac{\partial L^3}{\partial y^3}\frac{\partial y^3}{\partial h^3}\boxed{\frac{\partial h^3}{\partial w_{hh}}}$$

?

Unrolled computation graph



Backpropagation Through Time

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\,x^t + w_{hh}\,h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\,h^t + b_y$$

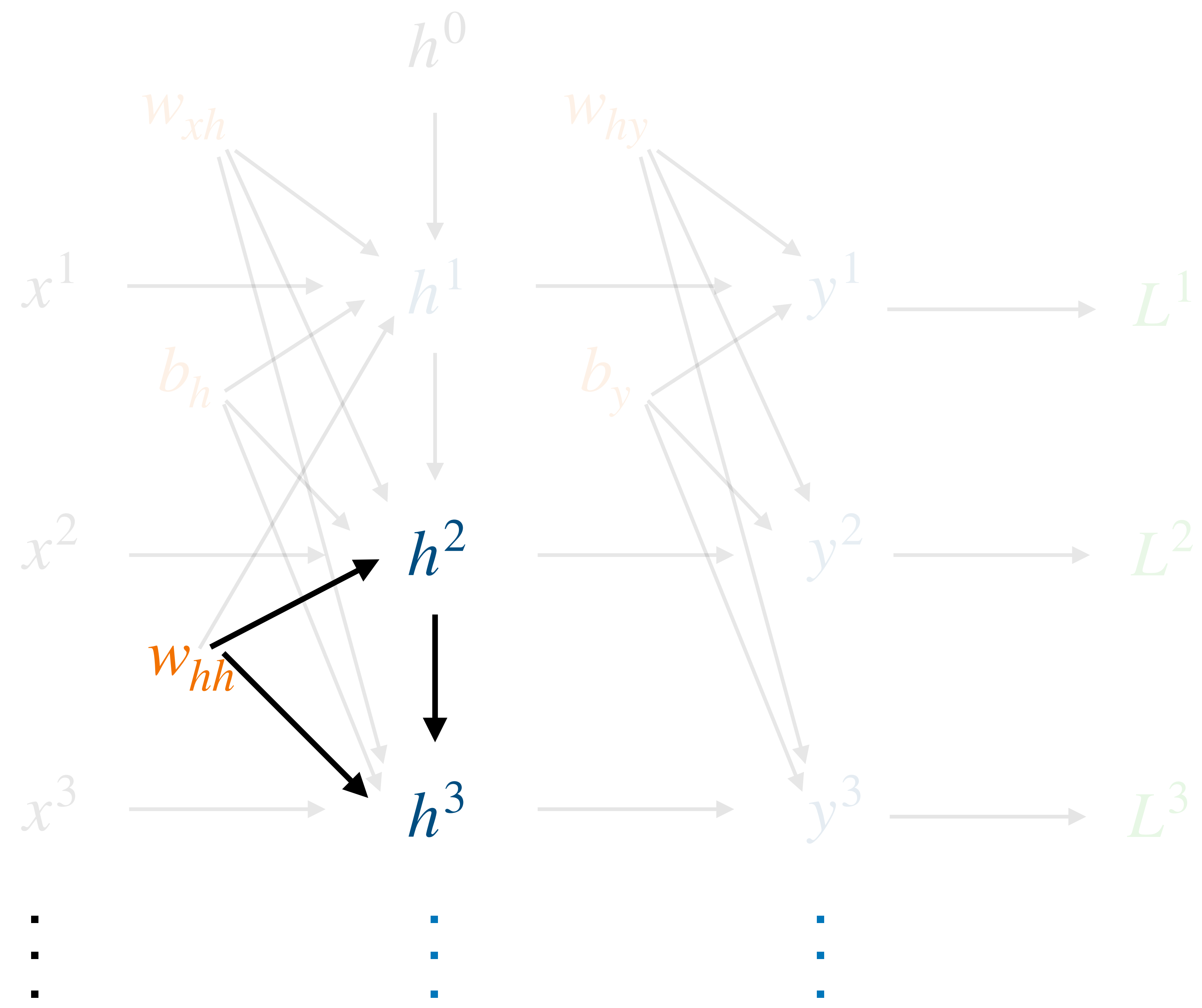
Loss function $L = \frac{1}{T}\sum_{t=1}^{T} L^t \qquad \frac{\partial L}{\partial w} = \frac{1}{T}\sum_{t=1}^{T}\frac{\partial L^t}{\partial w}$

Chain Rule

$$\frac{\partial L^3}{\partial w_{hh}} = \frac{\partial L^3}{\partial y^3}\frac{\partial y^3}{\partial h^3}\boxed{\frac{\partial h^3}{\partial w_{hh}}}$$

$$h^3 = \tanh(w_{hh}h^2 + \ldots)$$

Unrolled computation graph



Backpropagation Through Time

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\, h^t + b_y$$

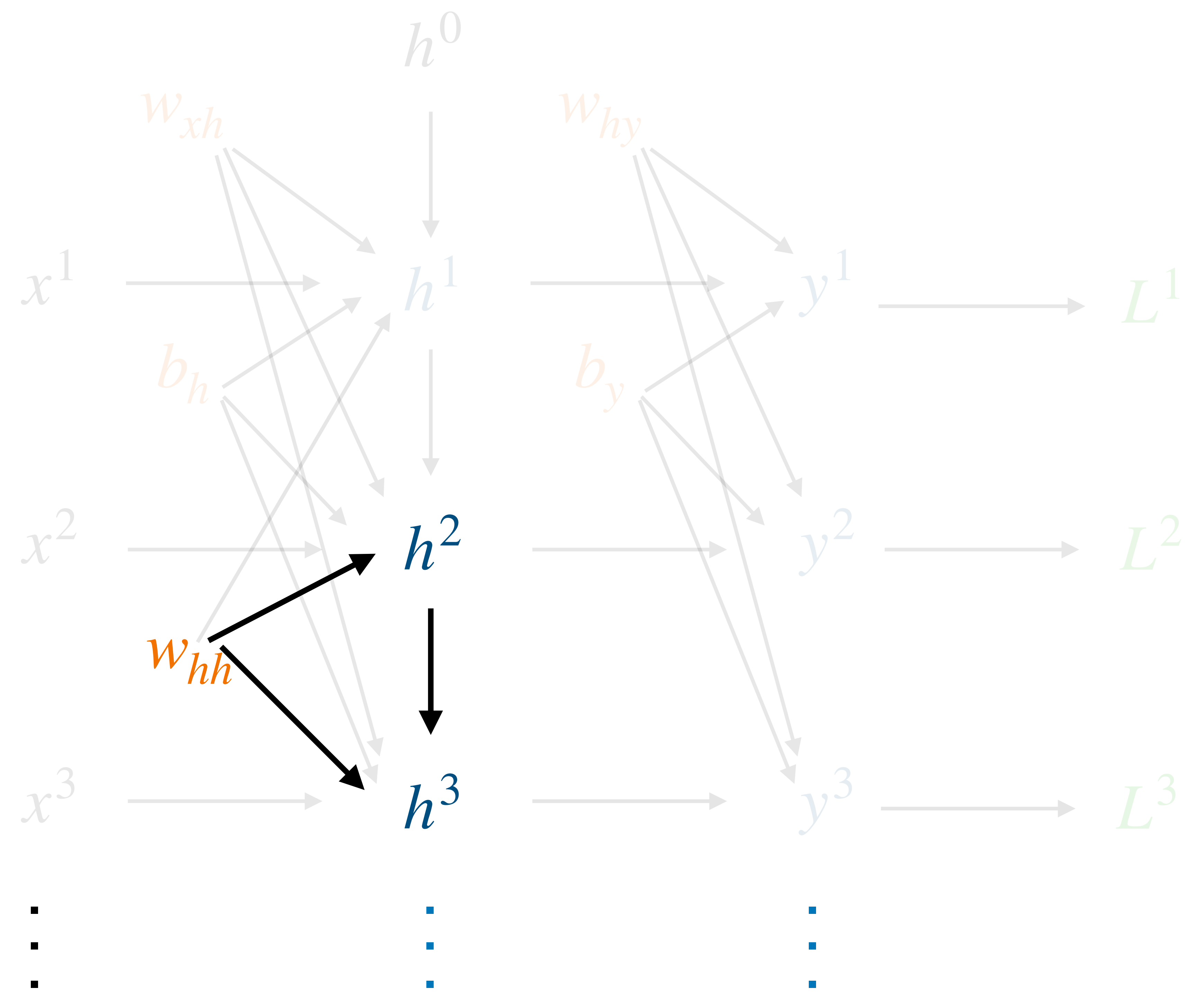
Loss function $L = \frac{1}{T}\sum_{t=1}^{T} L^t \qquad \frac{\partial L}{\partial w} = \frac{1}{T}\sum_{t=1}^{T}\frac{\partial L^t}{\partial w}$

Chain Rule

$$\frac{\partial L^3}{\partial w_{hh}} = \frac{\partial L^3}{\partial y^3}\frac{\partial y^3}{\partial h^3}\boxed{\frac{\partial h^3}{\partial w_{hh}}}$$

$$h^3 = \tanh(w_{hh}\tanh(w_{hh}h^1 + \ldots) + \ldots)$$

Unrolled computation graph



Backpropagation Through Time

Vanilla RNN

$$h^t = \tanh\left(w_{xh}\, x^t + w_{hh}\, h^{t-1} + b_h\right) \qquad h^0 = 0$$

$$y^t = w_{hy}\, h^t + b_y$$

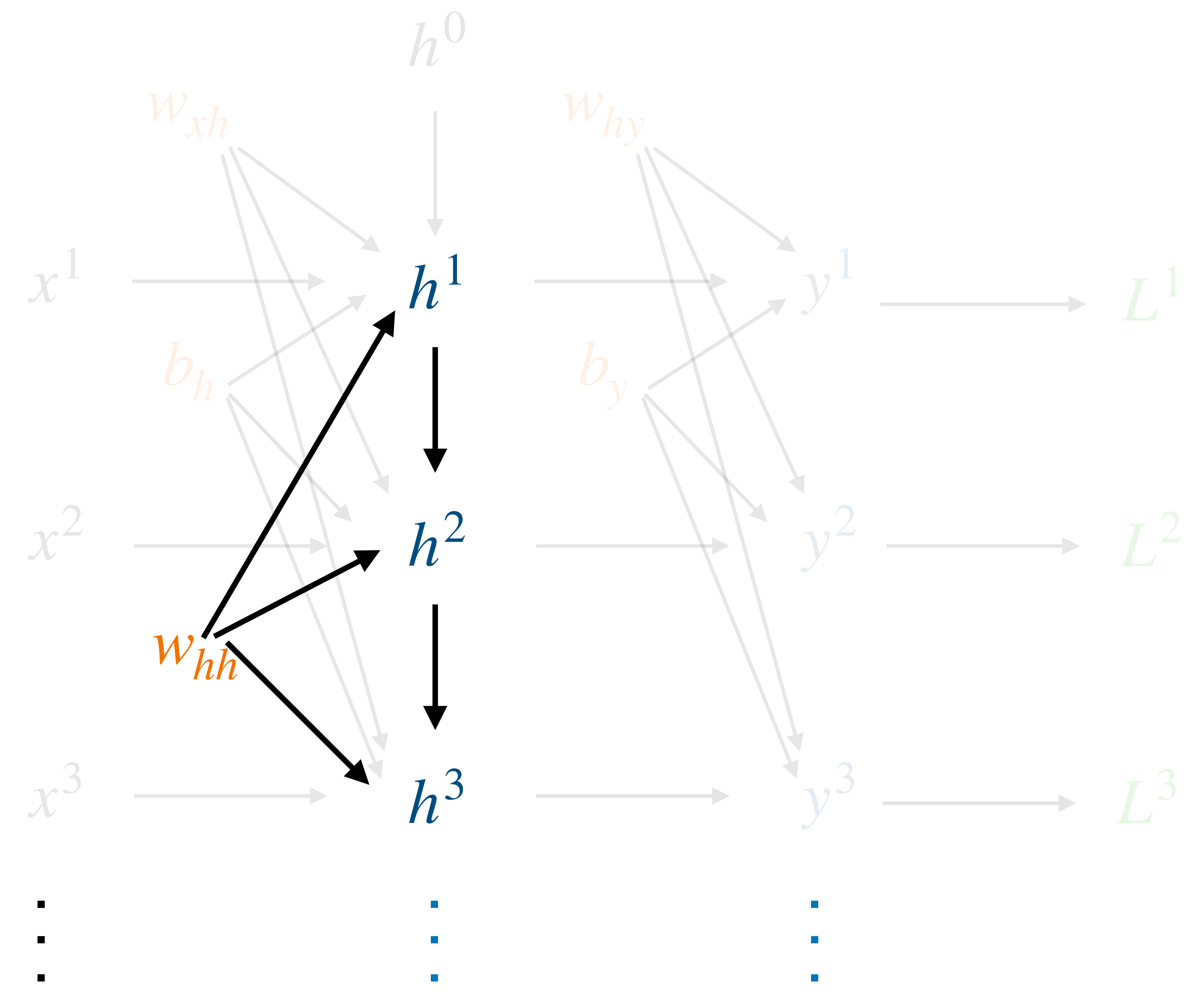
Loss function $L = \frac{1}{T}\sum_{t=1}^{T} L^t \qquad \frac{\partial L}{\partial w} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial L^t}{\partial w}$

Chain Rule

$$\frac{\partial L^3}{\partial w_{hh}} = \frac{\partial L^3}{\partial y^3}\frac{\partial y^3}{\partial h^3}\boxed{\frac{\partial h^3}{\partial w_{hh}}}$$

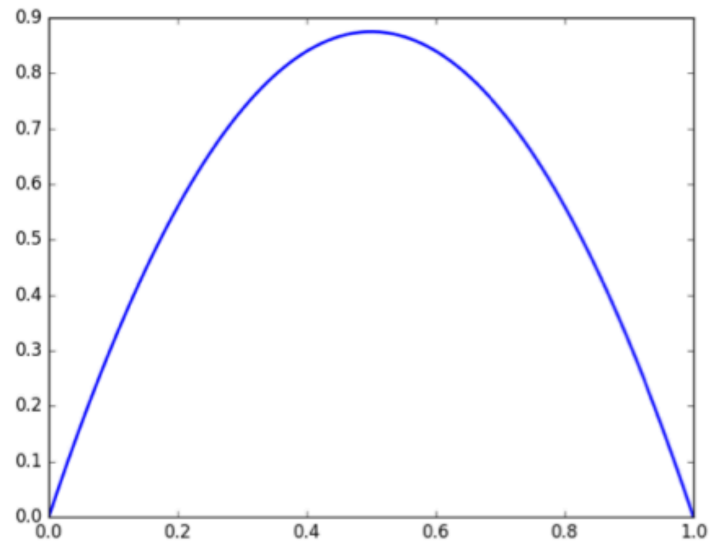
$$h^3 = \tanh(w_{hh} \tanh(w_{hh} \tanh(w_{hh} h^0 + \dots) + \dots) + \dots)$$

Unrolled computation graph

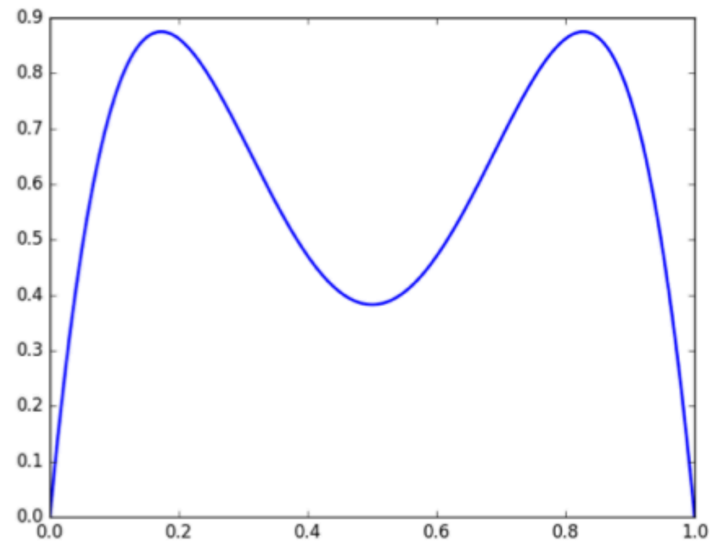


Vanishing and Exploding Gradients

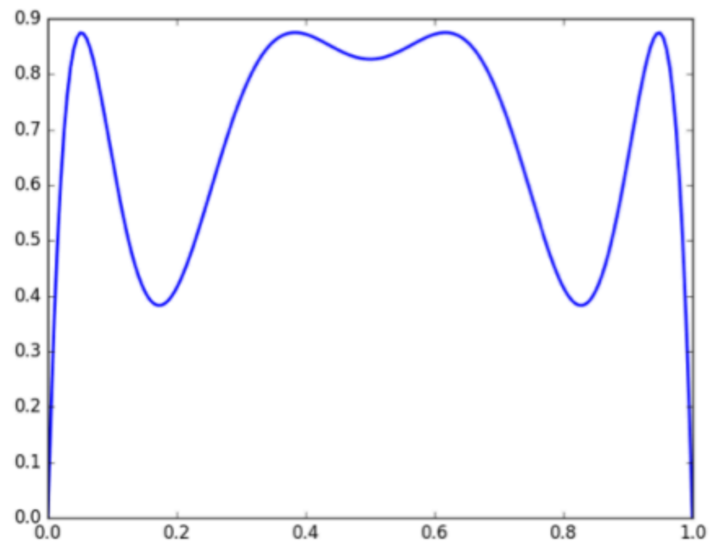
$$h^t = f(h^{t-1}, w_{hh}) = f(f(h^{t-2}, w_{hh}), w_{hh}) = \dots$$



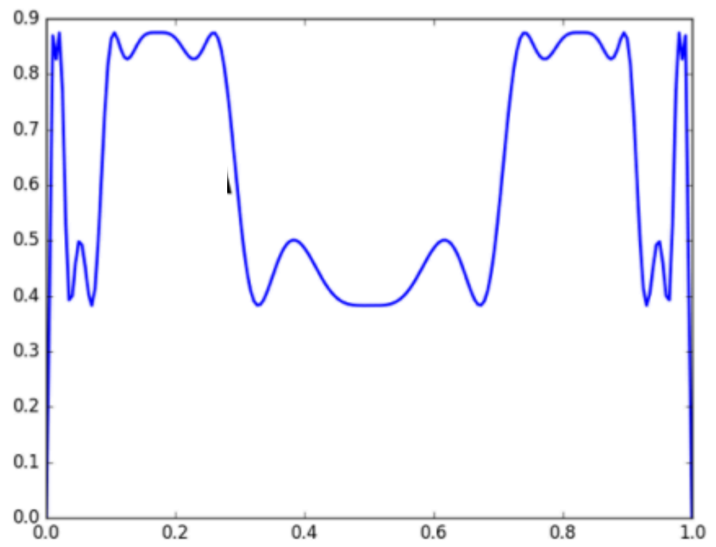
$h^1 = f(h^0, w_{hh})$



$h^2 = f(f(h^0, w_{hh}), w_{hh})$



$h^3 = f(f(f\dots)$



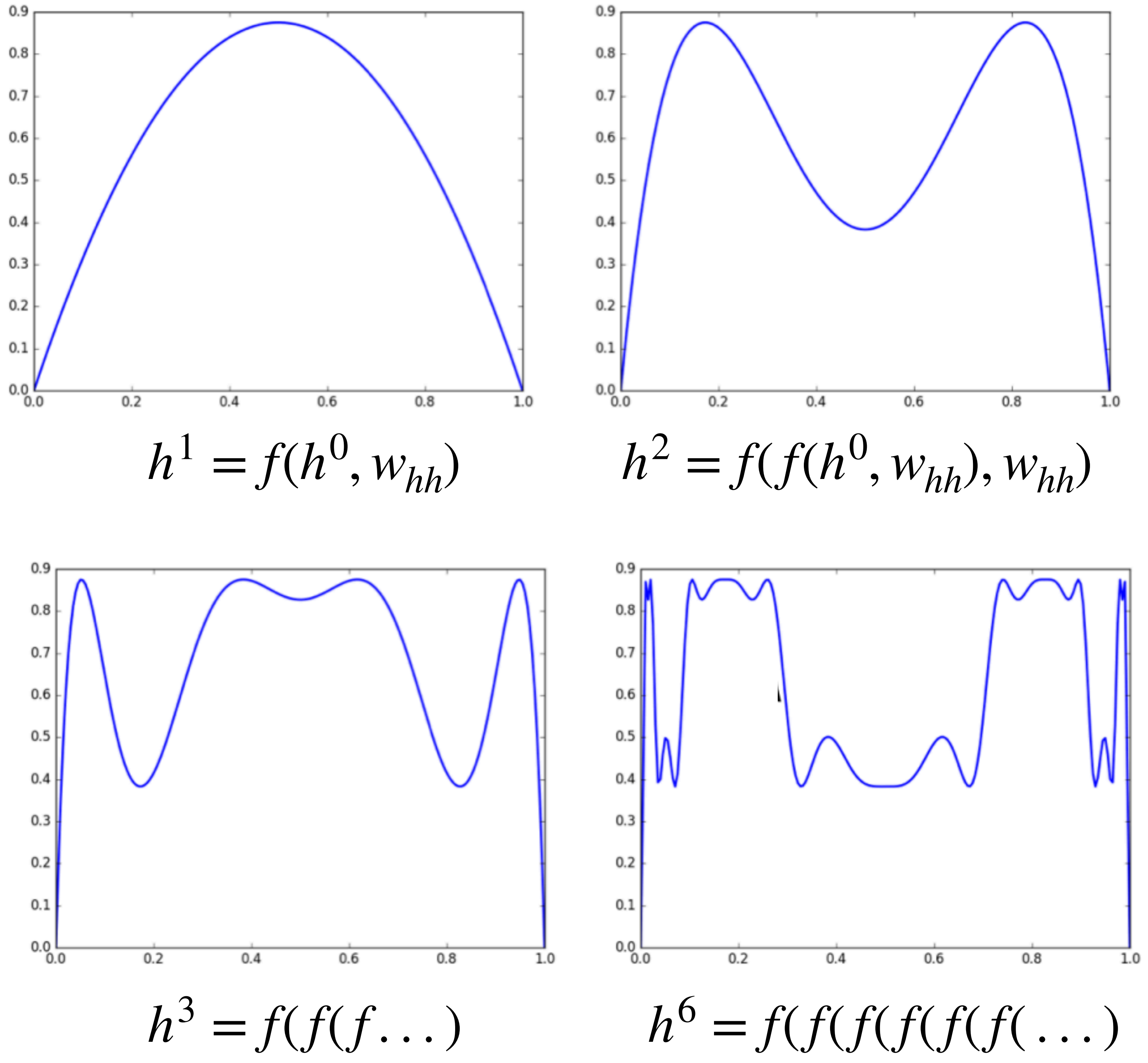
$h^6 = f(f(f(f(f(f(\dots)$

The function behaviour gets more chaotic with iterations

Vanishing and Exploding Gradients

$$h^t = f(h^{t-1}, w_{hh}) = f(f(h^{t-2}, w_{hh}), w_{hh}) = \ldots$$

What is the problem with the derivative $\dfrac{\partial h^t}{\partial w_{hh}}$?



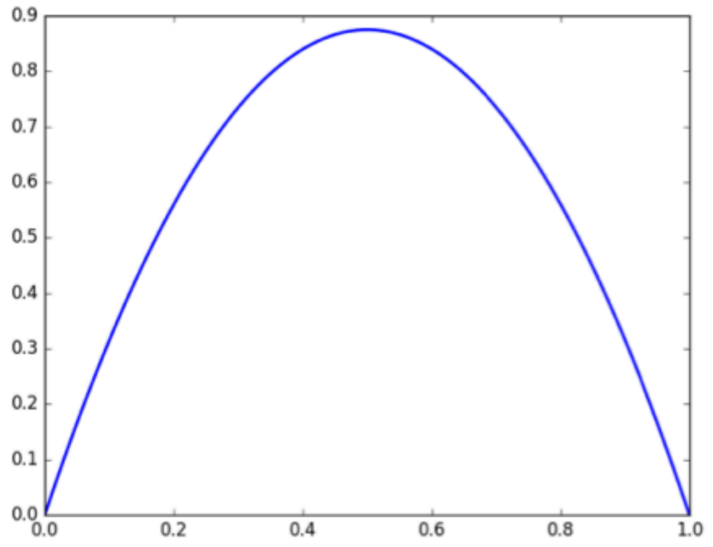
The function behaviour gets more chaotic with iterations

Vanishing and Exploding Gradients

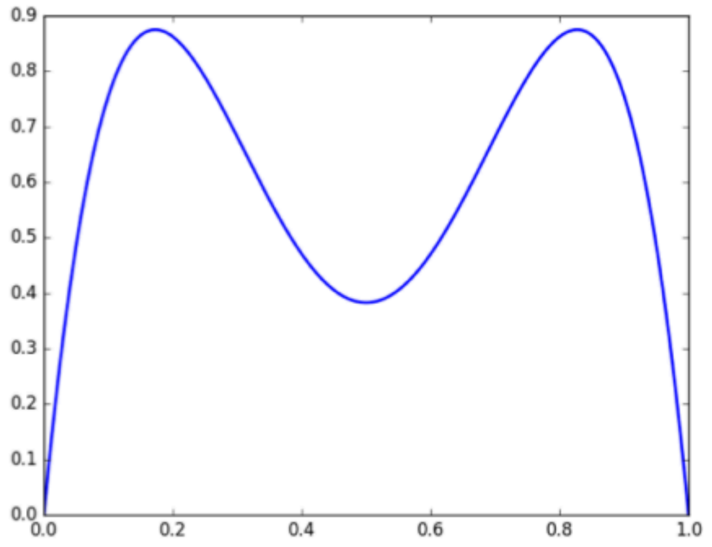
$$h^t = f(h^{t-1}, w_{hh}) = f(f(h^{t-2}, w_{hh}), w_{hh}) = \ldots$$

Especially, for large sequence length T

$$\frac{\partial h^T}{\partial w_{hh}} \approx \infty \longrightarrow \frac{\partial L^T}{\partial w_{hh}} \approx \infty \text{ (explodes)}$$

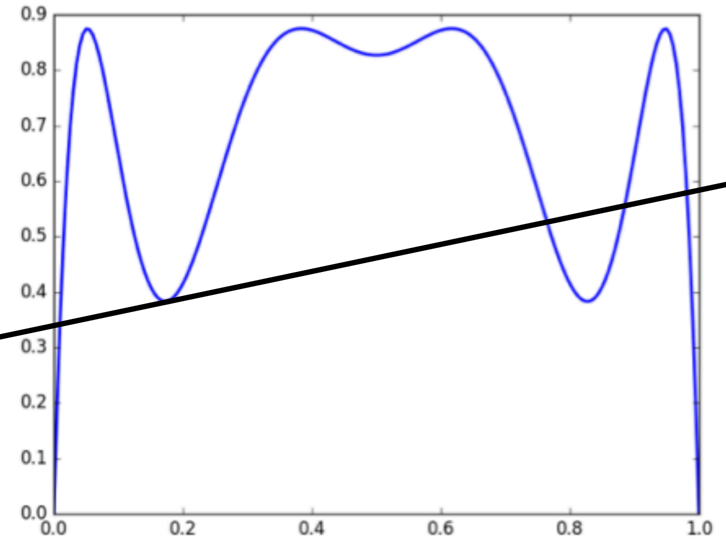


$h^1 = f(h^0, w_{hh})$

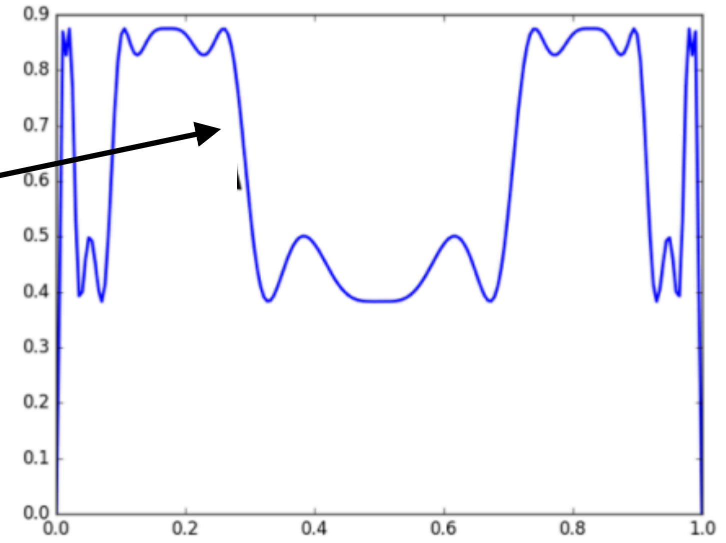


$h^2 = f(f(h^0, w_{hh}), w_{hh})$

Large



$h^3 = f(f(f\ldots)$



$h^6 = f(f(f(f(f(f(\ldots)$

Roger Grosse Lecture notes - Lecture 14: Exploding and Vanishing Gradients

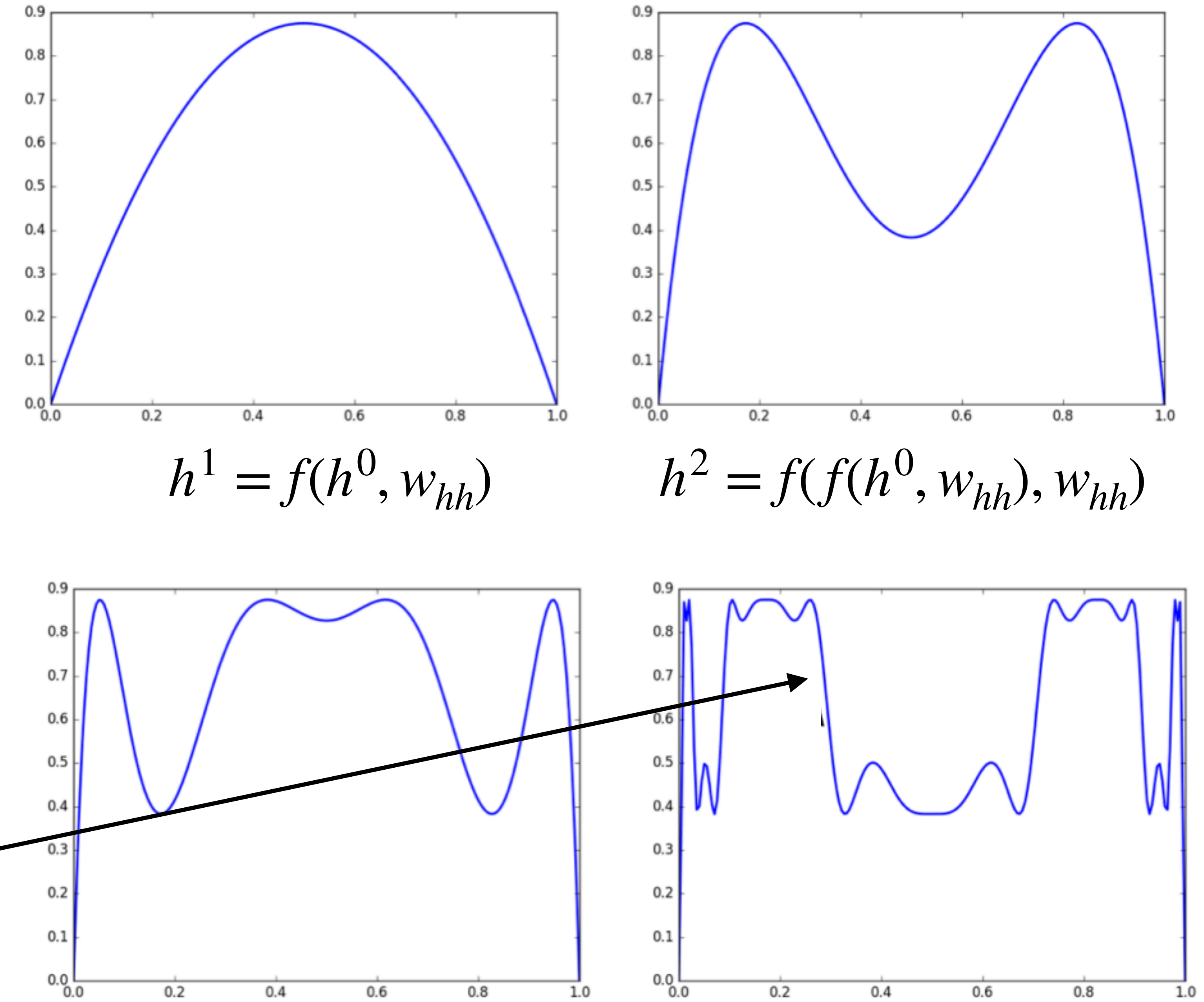
Vanishing and Exploding Gradients

$$h^t = f(h^{t-1}, w_{hh}) = f(f(h^{t-2}, w_{hh}), w_{hh}) = \dots$$

Especially, for large sequence length T

$$\frac{\partial h^T}{\partial w_{hh}} \approx \infty \longrightarrow \frac{\partial L^T}{\partial w_{hh}} \approx \infty \longrightarrow \frac{\partial L}{\partial w_{hh}} \approx \infty \text{ (explodes)}$$

Large



$h^1 = f(h^0, w_{hh})$ $h^2 = f(f(h^0, w_{hh}), w_{hh})$

$h^3 = f(f(f\dots)$ $h^6 = f(f(f(f(f(f(\dots)$

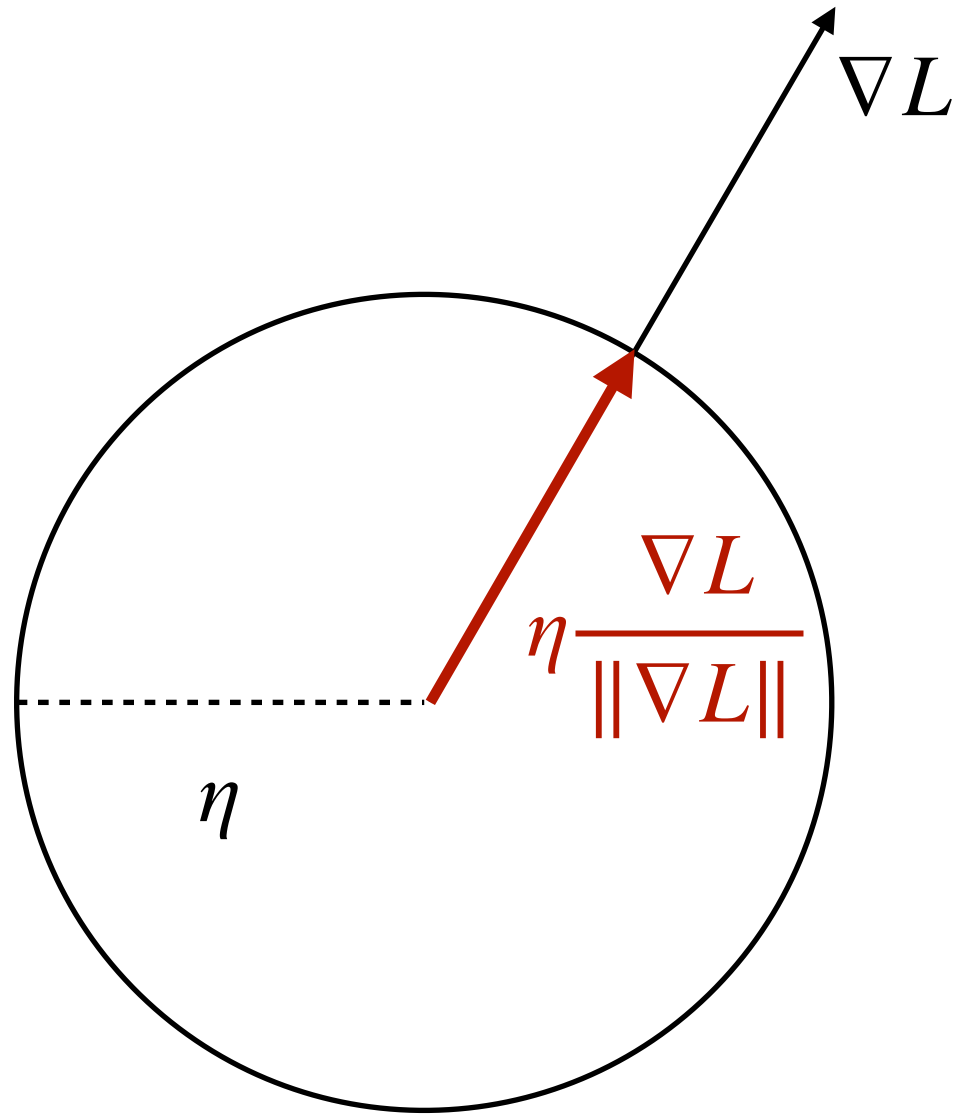
Roger Grosse Lecture notes - Lecture 14: Exploding and Vanishing Gradients

Vanishing and Exploding Gradients

$$h^t = f(h^{t-1}, w_{hh}) = f(f(h^{t-2}, w_{hh}), w_{hh}) = \dots$$

Especially, for large sequence length T

$$\frac{\partial h^T}{\partial w_{hh}} \approx \infty \longrightarrow \frac{\partial L^T}{\partial w_{hh}} \approx \infty \longrightarrow \frac{\partial L}{\partial w_{hh}} \approx \infty \text{ (explodes)}$$



Strategy: gradient clipping

If the norm $\|\nabla L\|$ is larger than a threshold η, rescale the gradient to have same direction but norm η

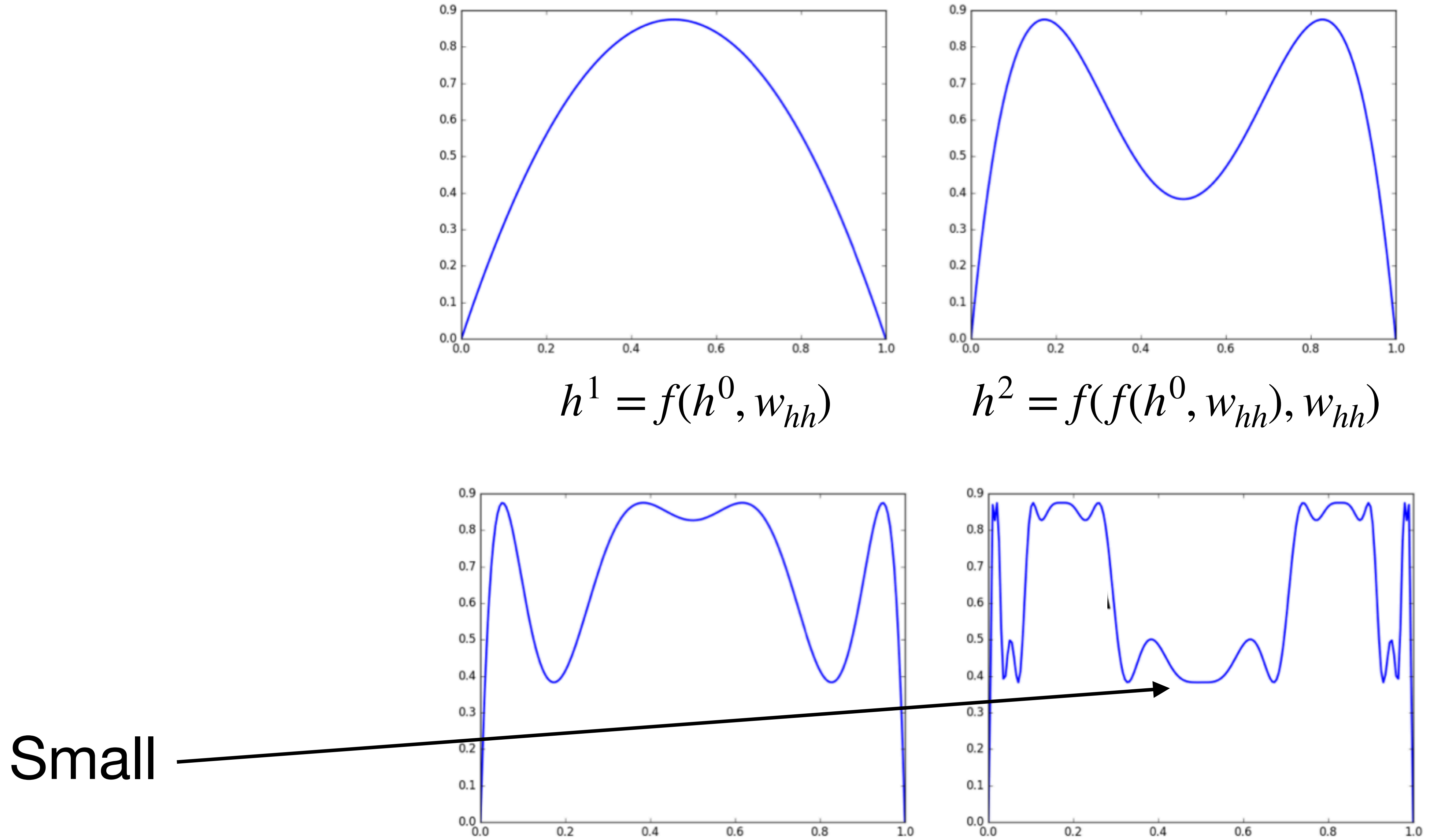
Roger Grosse Lecture notes - Lecture 14: Exploding and Vanishing Gradients

Vanishing and Exploding Gradients

$$h^t = f(h^{t-1}, w_{hh}) = f(f(h^{t-2}, w_{hh}), w_{hh}) = \ldots$$

Especially, for large sequence length T

$$\frac{\partial h^T}{\partial w_{hh}} \approx 0 \longrightarrow \frac{\partial L^T}{\partial w_{hh}} \approx 0 \text{ (vanishes)}$$



$h^1 = f(h^0, w_{hh})$ $h^2 = f(f(h^0, w_{hh}), w_{hh})$

$h^3 = f(f(f\ldots)$ $h^6 = f(f(f(f(f(f(\ldots)$

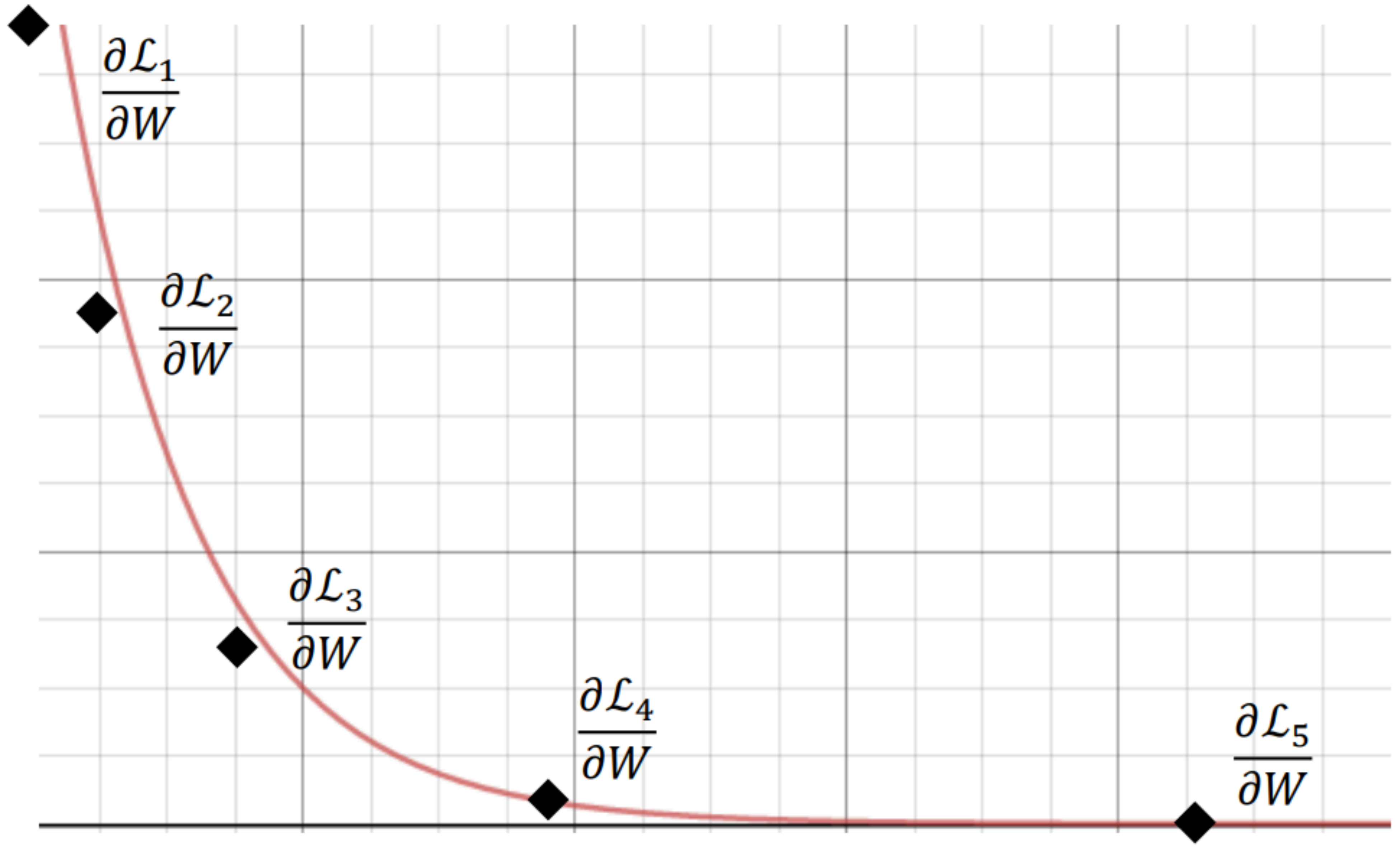
Vanishing and Exploding Gradients

$$h^t = f(h^{t-1}, w_{hh}) = f(f(h^{t-2}, w_{hh}), w_{hh}) = \dots$$

Especially, for large sequence length T

$$\frac{\partial h^T}{\partial w_{hh}} \approx 0 \longrightarrow \frac{\partial L^T}{\partial w_{hh}} \approx 0 \text{ (vanishes)}$$

$$\frac{\partial L}{\partial w_{hh}} = \frac{1}{T}\left(\frac{\partial L^1}{\partial w_{hh}} + \frac{\partial L^2}{\partial w_{hh}} + \frac{\partial L^3}{\partial w_{hh}} + \frac{\partial L^4}{\partial w_{hh}} + \frac{\partial L^5}{\partial w_{hh}} + \dots\right)$$

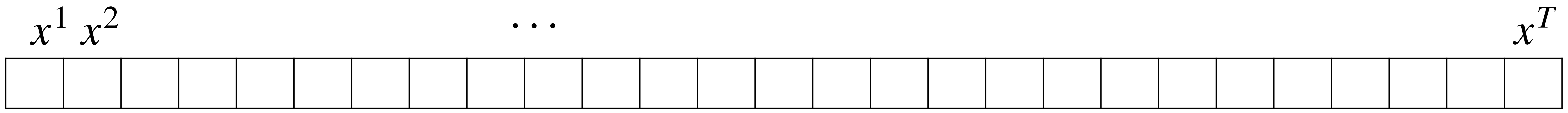


Loss and weights updates focus on early time steps → long-term dependencies are ignored!

Roger Grosse Lecture notes - Lecture 14: Exploding and Vanishing Gradients

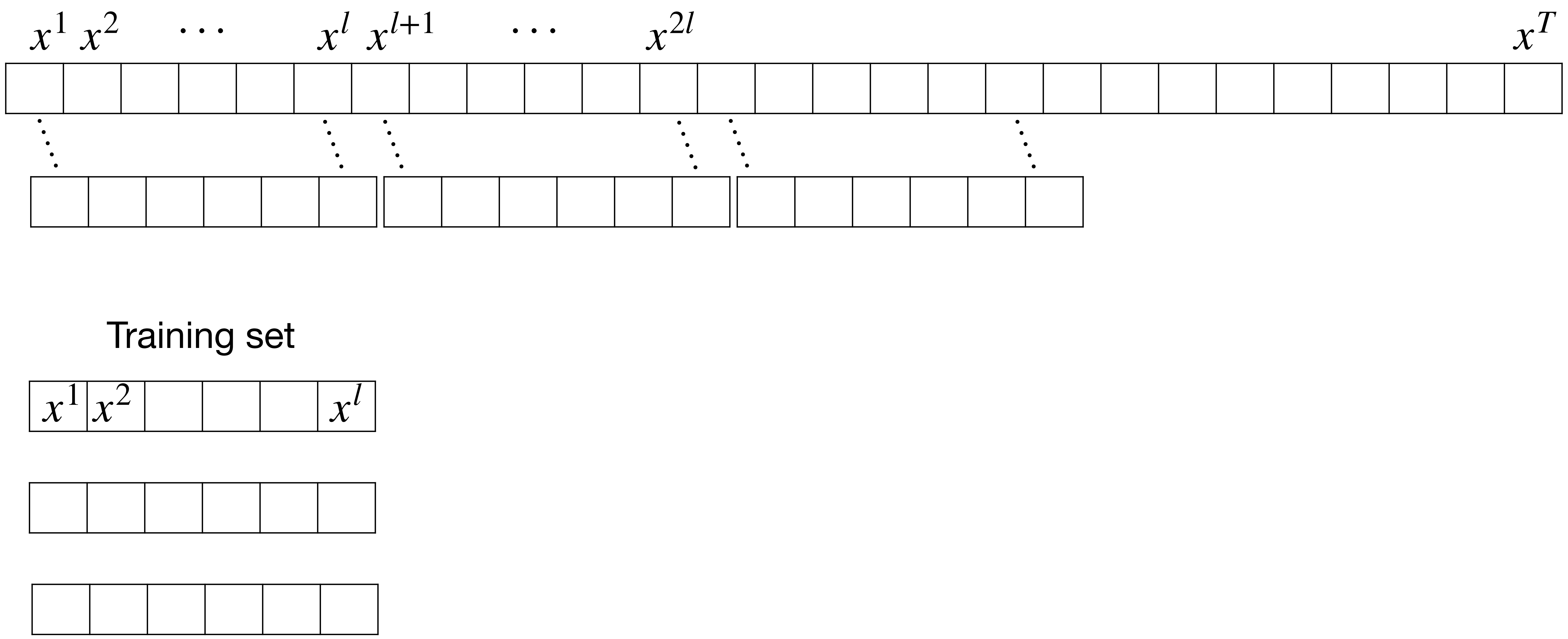
Truncated Backpropagation

How to train for very long sequences? Backpropagation all the way is unfeasible!



Truncated Backpropagation

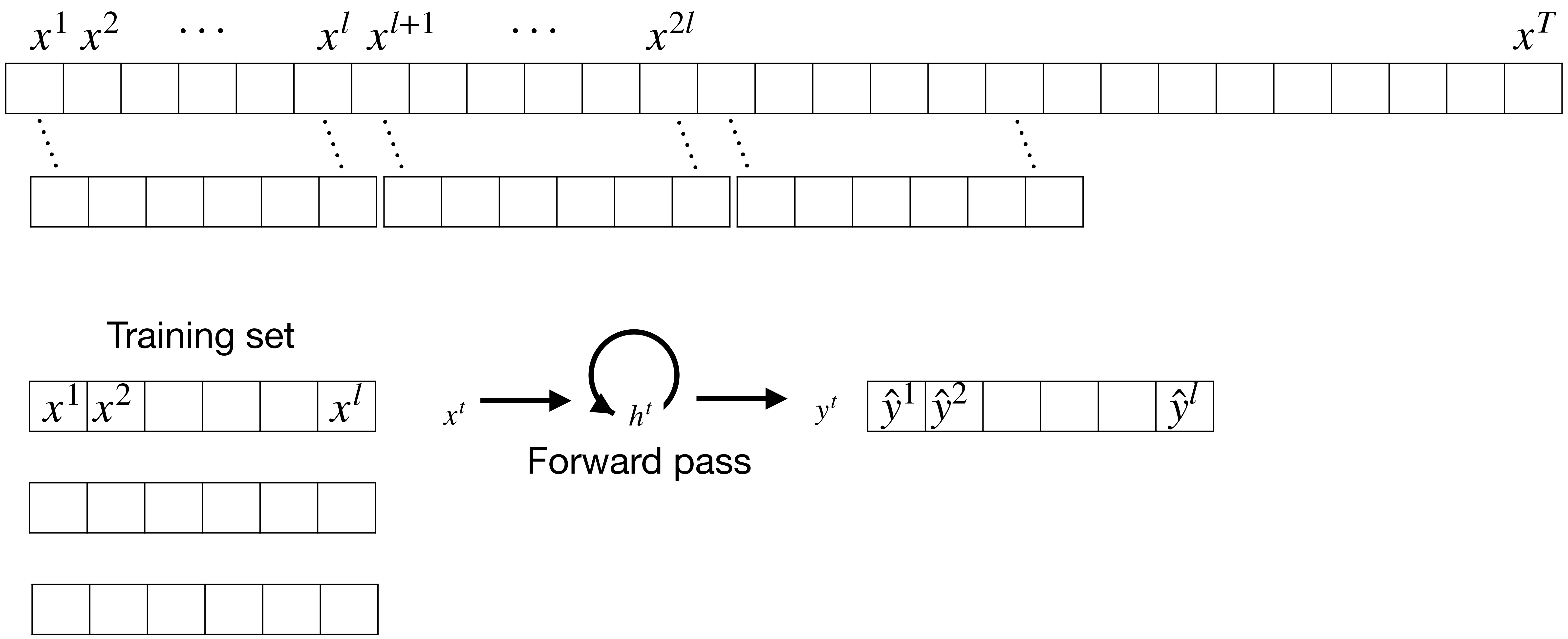
How to train for very long sequences? Backpropagation all the way is unfeasible!



Process the sequence in “chunks” with shorter sequence length l

Truncated Backpropagation

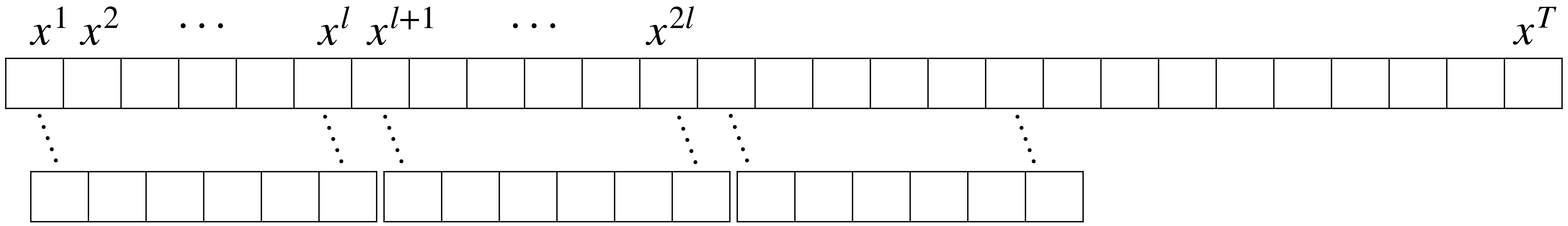
How to train for very long sequences? Backpropagation all the way is unfeasible!

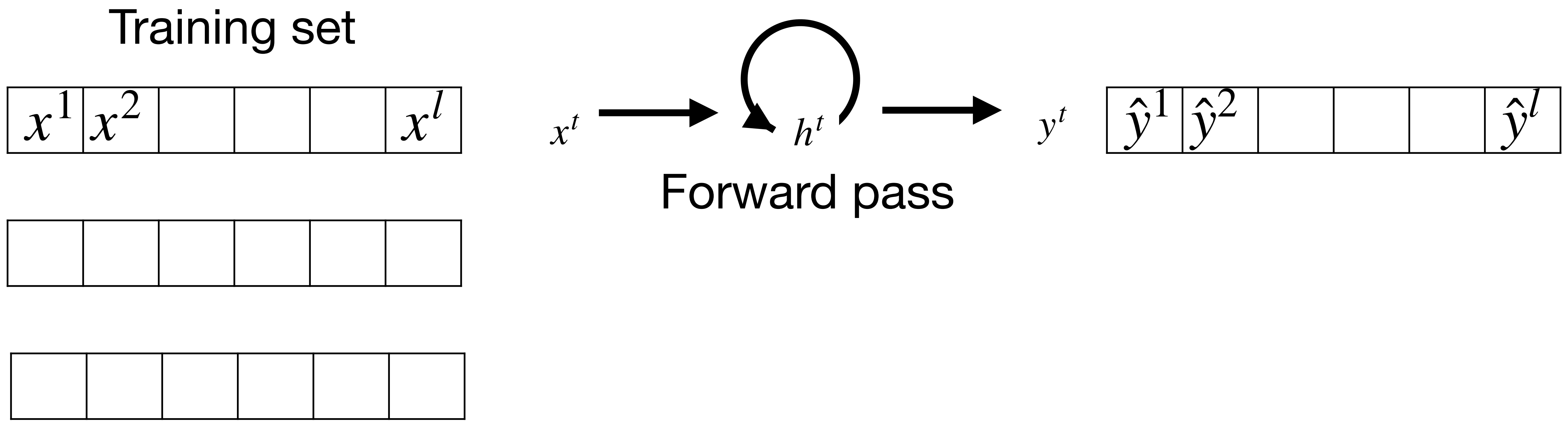


Process the sequence in "chunks" with shorter sequence length l

Truncated Backpropagation

How to train for very long sequences? Backpropagation all the way is unfeasible!





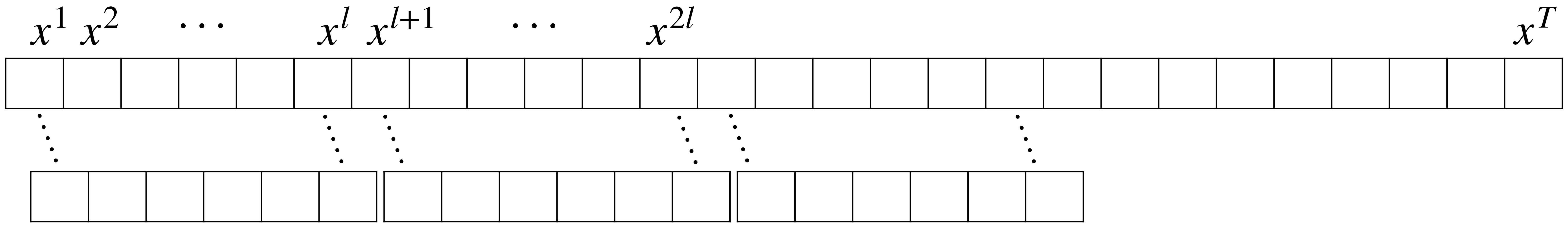
$$L = \frac{1}{l}\sum_{t=1}^{l} L^t \qquad \frac{\partial L}{\partial w} \approx \frac{1}{l}\sum_{t=1}^{l}\frac{\partial L^t}{\partial w}$$

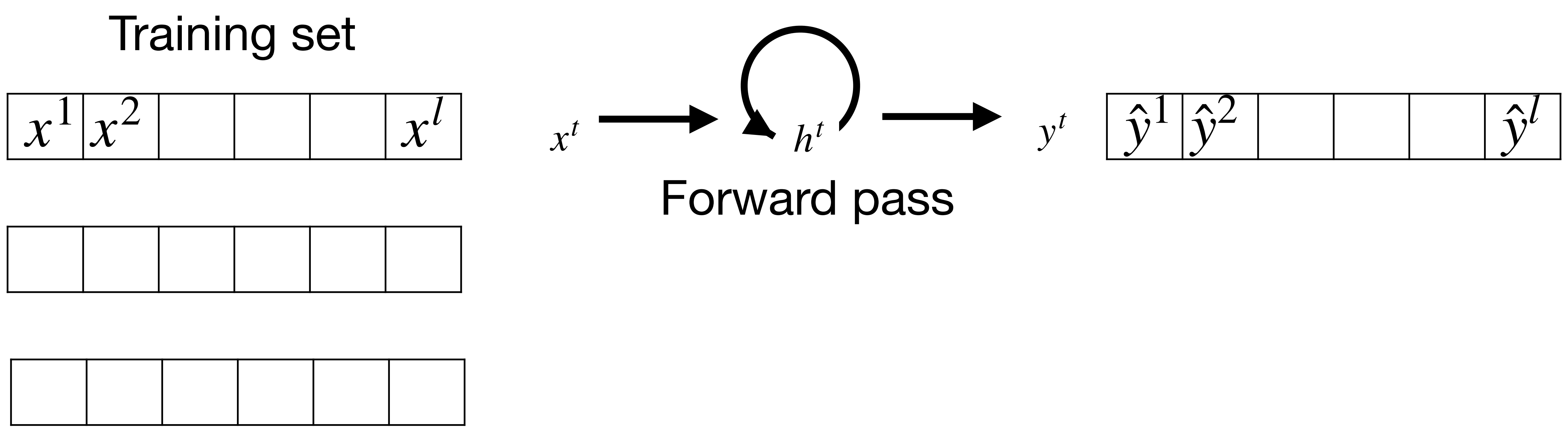
Compute loss and backward pass

Process the sequence in “chunks” with shorter sequence length l

Truncated Backpropagation

How to train for very long sequences? Backpropagation all the way is unfeasible!





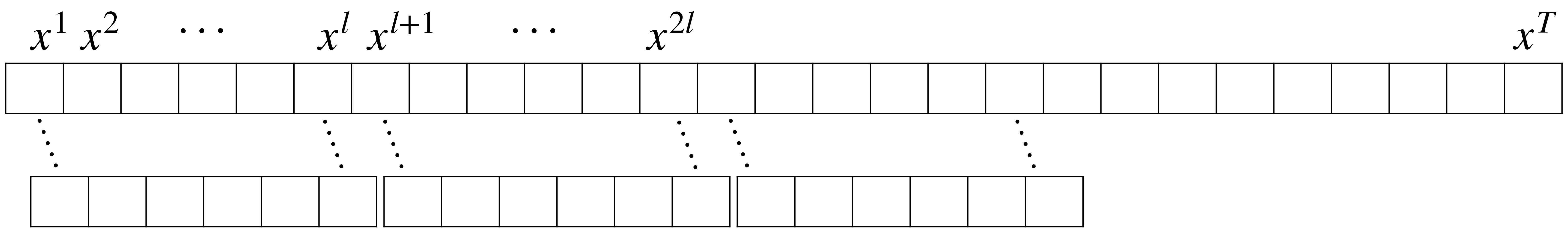
$$L = \frac{1}{l}\sum_{t=1}^{l} L^t \qquad \frac{\partial L}{\partial w} \approx \frac{1}{l}\sum_{t=1}^{l}\frac{\partial L^t}{\partial w}$$

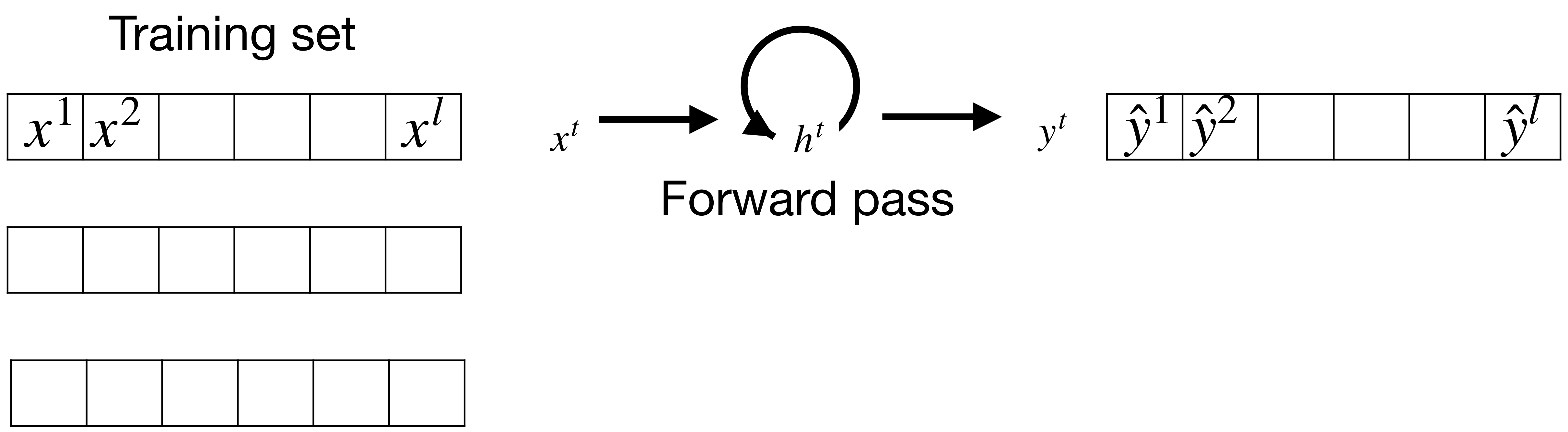
Compute loss and backward pass

l tradeoff computational efficiency and long-term dependencies

Truncated Backpropagation

How to train for very long sequences? Backpropagation all the way is unfeasible!





$$L = \frac{1}{l}\sum_{t=1}^{l} L^t \qquad \frac{\partial L}{\partial w} \approx \frac{1}{l}\sum_{t=1}^{l}\frac{\partial L^t}{\partial w}$$

Compute loss and backward pass

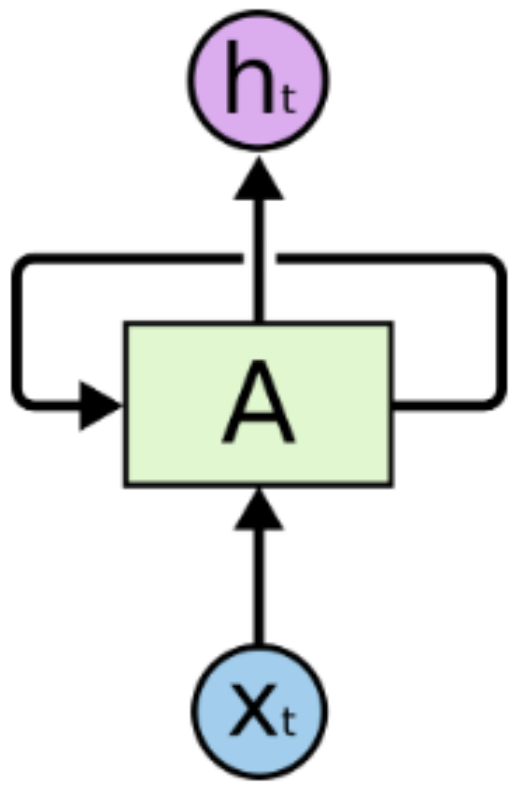
What about testing? Do we need chunks?

Long-Short Term Memory (LSTM)

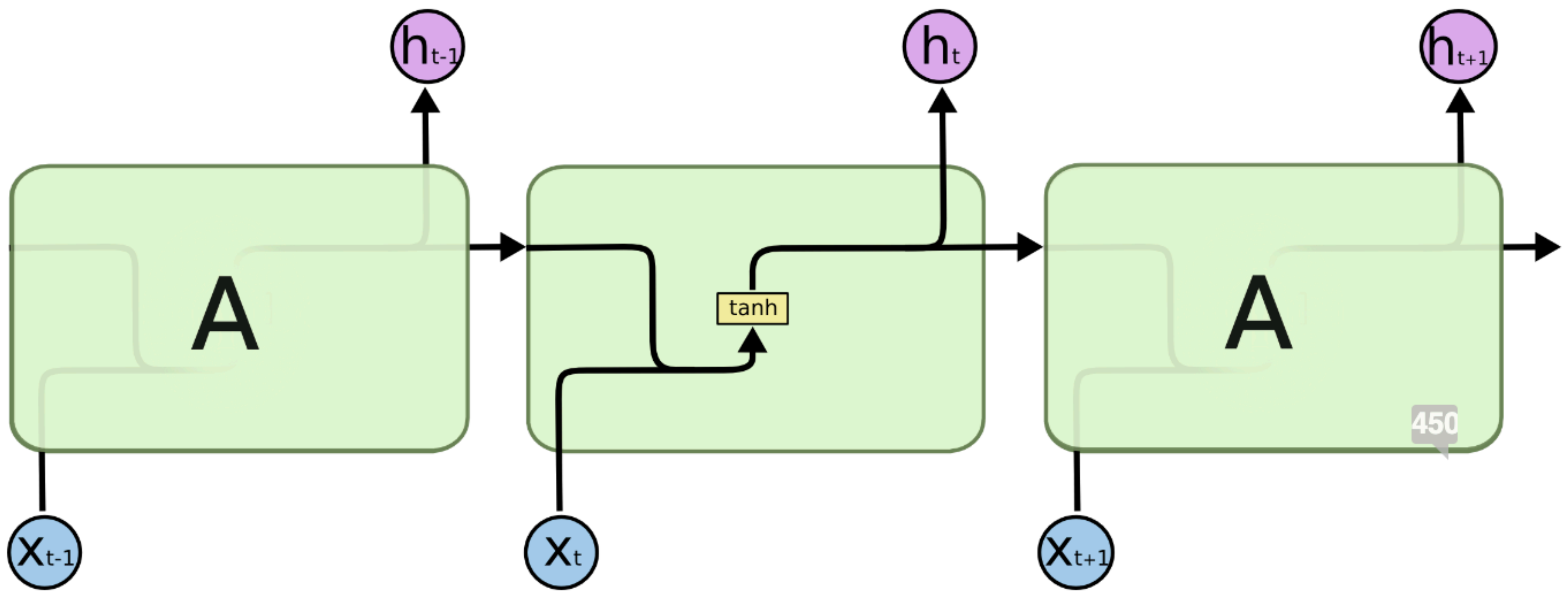
Architecture designed to remember information over many time steps and handle long-term dependencies

Long-Short Term Memory (LSTM)

Architecture designed to remember information over many time steps and handle long-term dependencies



RNN



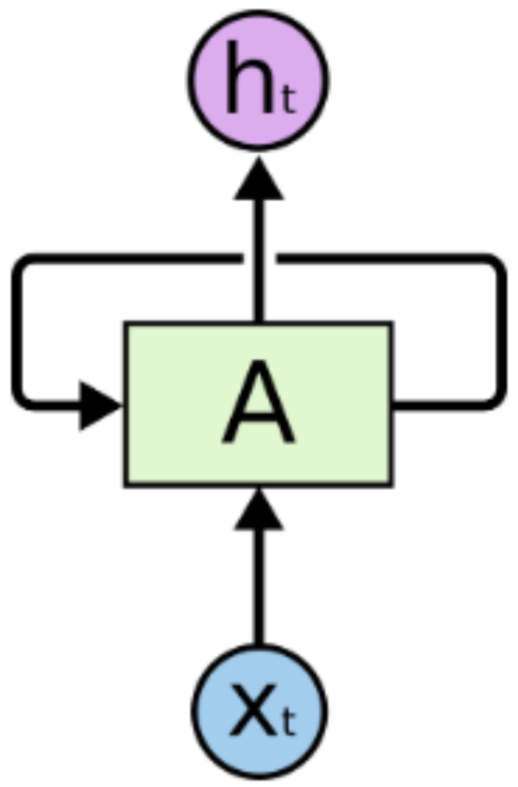
Vanilla RNN architecture

What does the single module A compute?

*Figures from *Colah's blog: Understanding LSTM Networks*

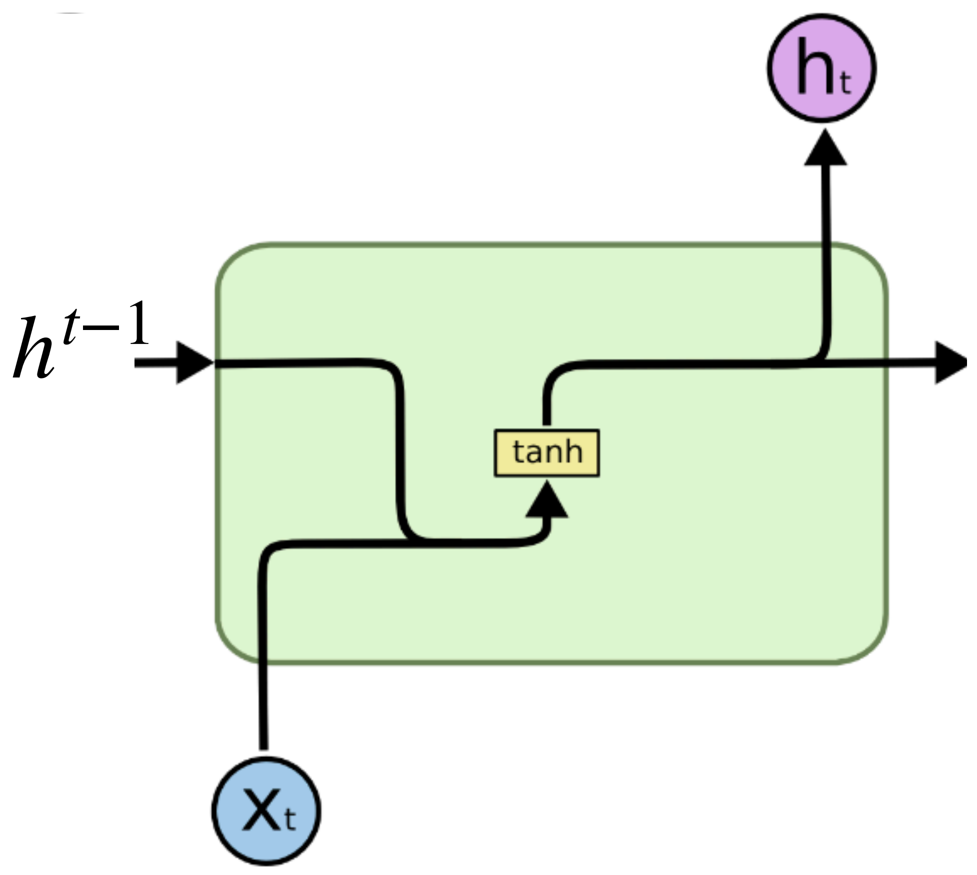
Long-Short Term Memory (LSTM)

Architecture designed to remember information over many time steps and handle long-term dependencies



RNN

$$y^t = w_{hy}\,h^t + b_y$$



$$h^t = \tanh\left(x^t \cdot w_{xh} + h^{t-1} \cdot w_{hh} + b_h\right)$$

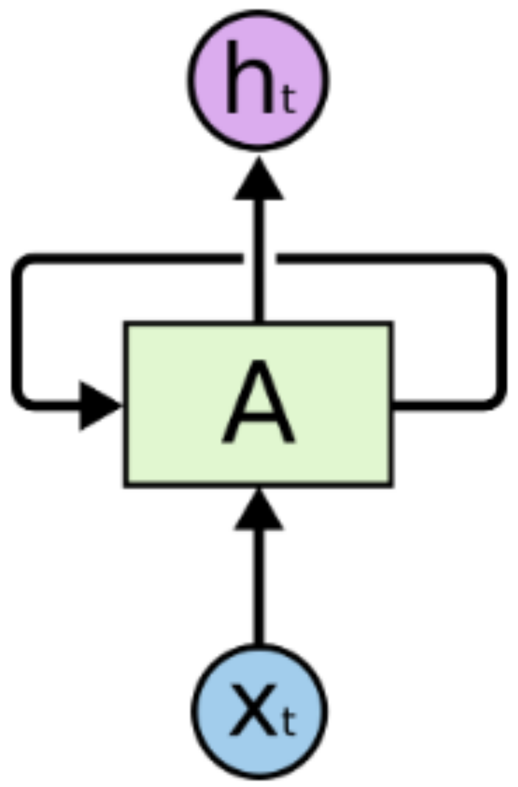
Vanilla RNN architecture

What does the single module A compute?

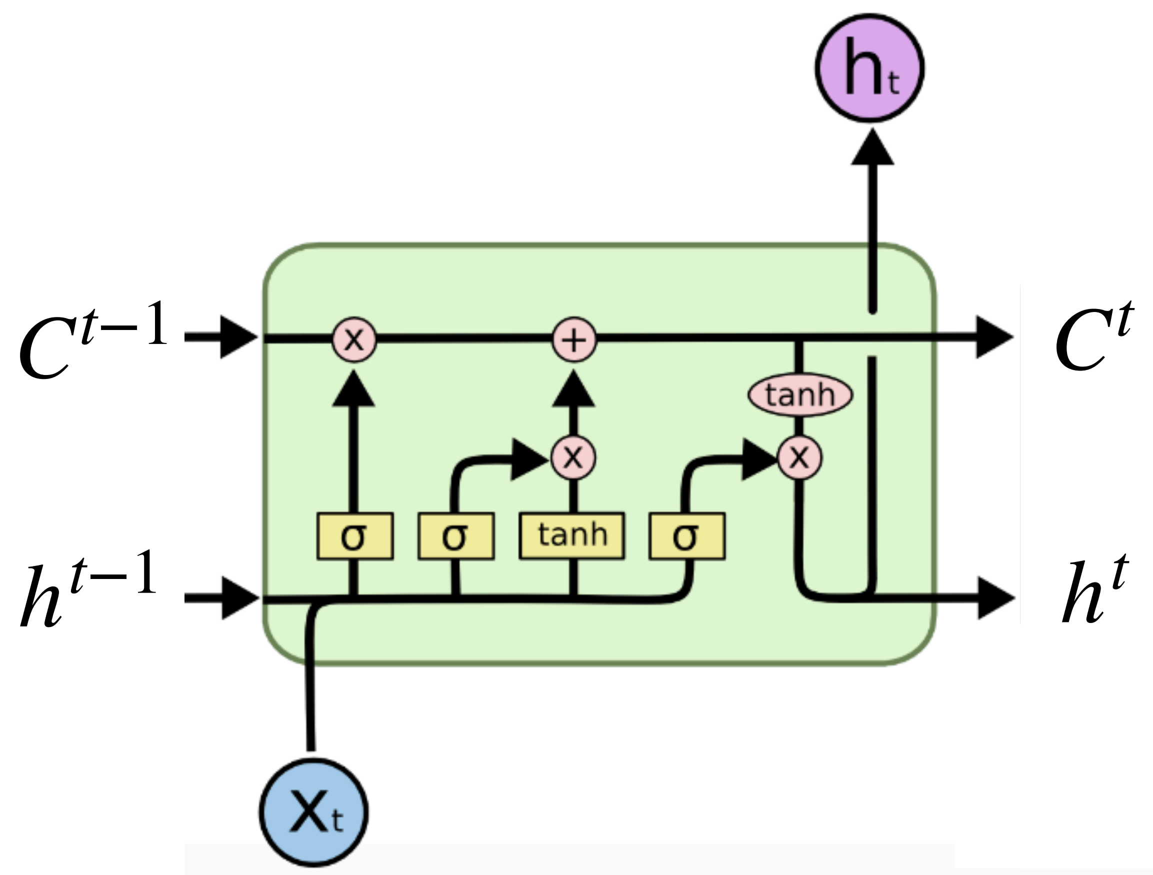
*Figures from *Colah's blog: Understanding LSTM Networks*

Long-Short Term Memory (LSTM)

Architecture designed to remember information over many time steps and handle long-term dependencies



RNN



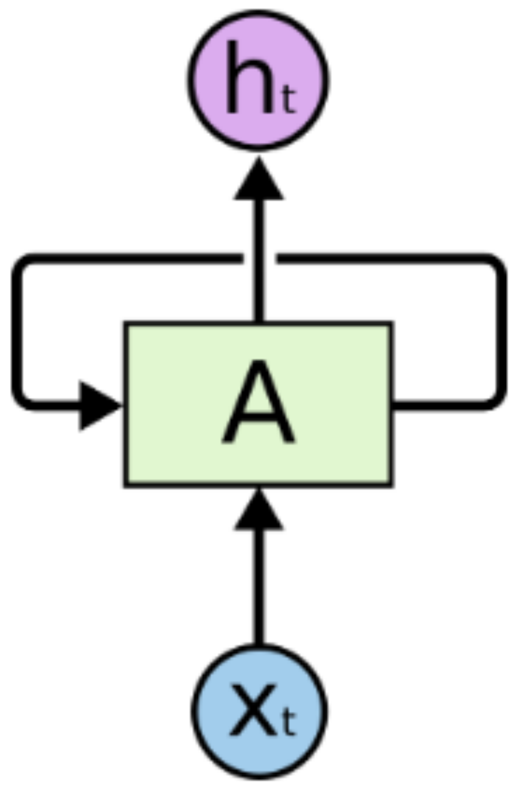
LSTM architecture

Each module has four layers and a cell state, the key component to retain long-term memory

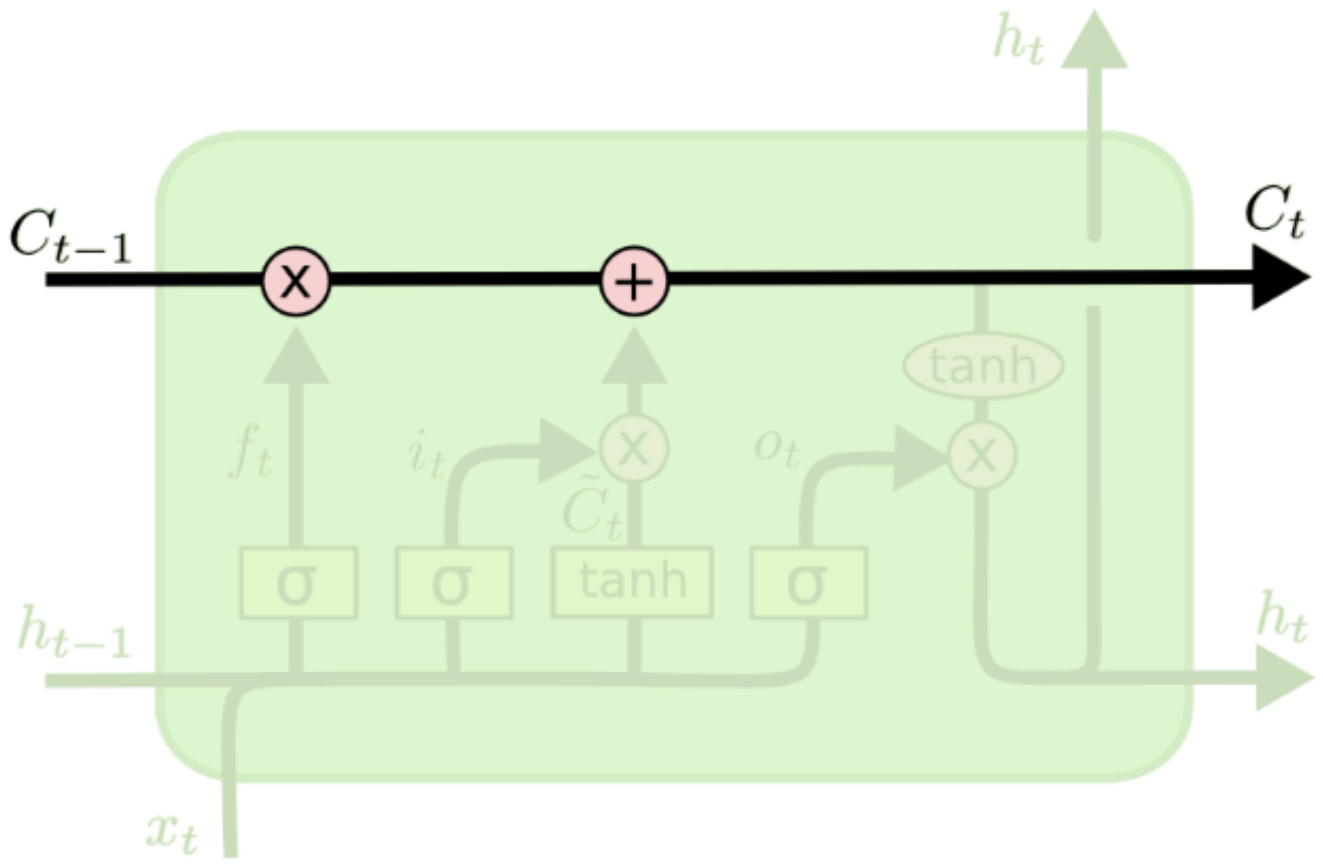
*Figures from *Colah's blog: Understanding LSTM Networks*

Long-Short Term Memory (LSTM)

Architecture designed to remember information over many time steps and handle long-term dependencies



RNN



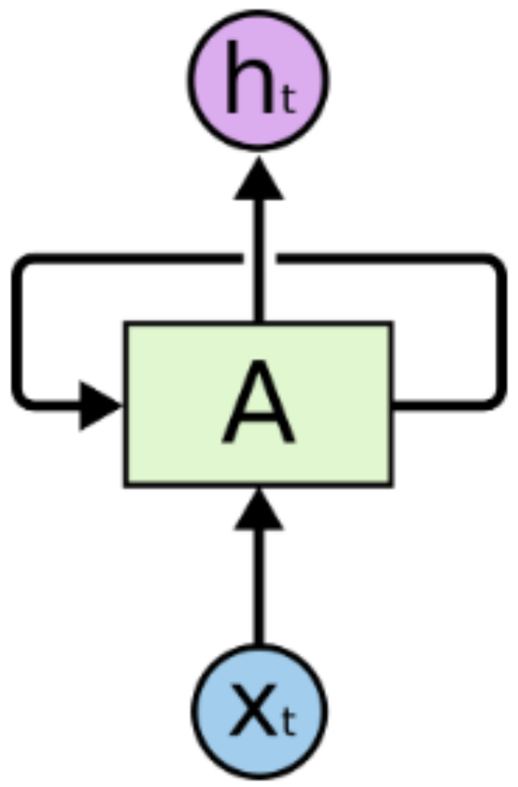
LSTM architecture

Information in cell state passes through the module with only minor linear changes

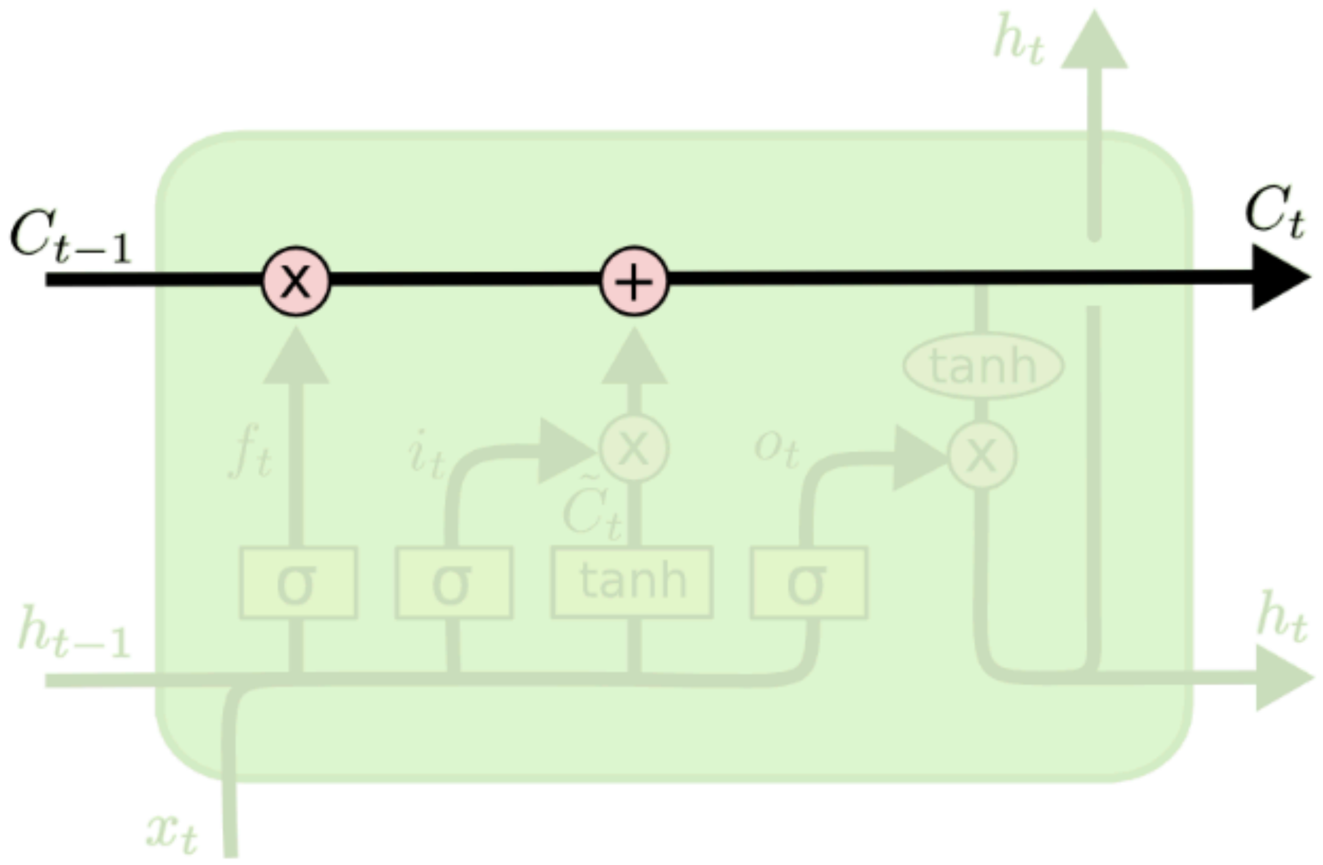
*Figures from *Colah's blog: Understanding LSTM Networks*

Long-Short Term Memory (LSTM)

Architecture designed to remember information over many time steps and handle long-term dependencies



RNN



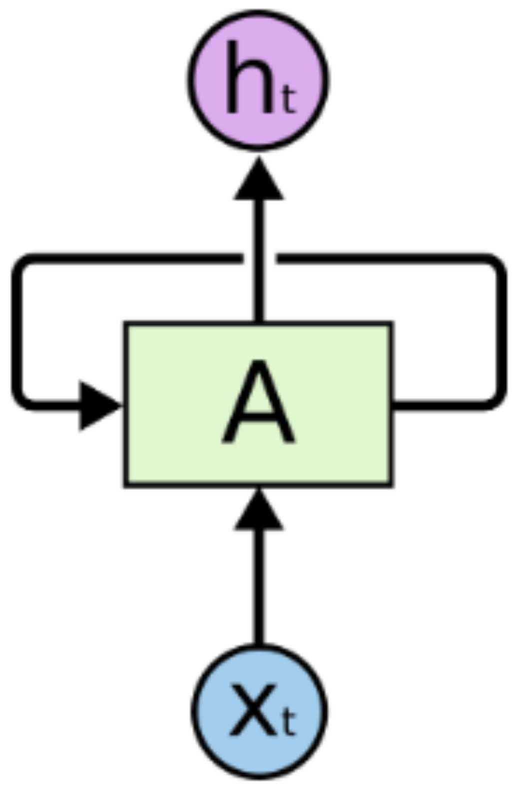
LSTM architecture

“Gates” control the addition or removal of information to the cell state

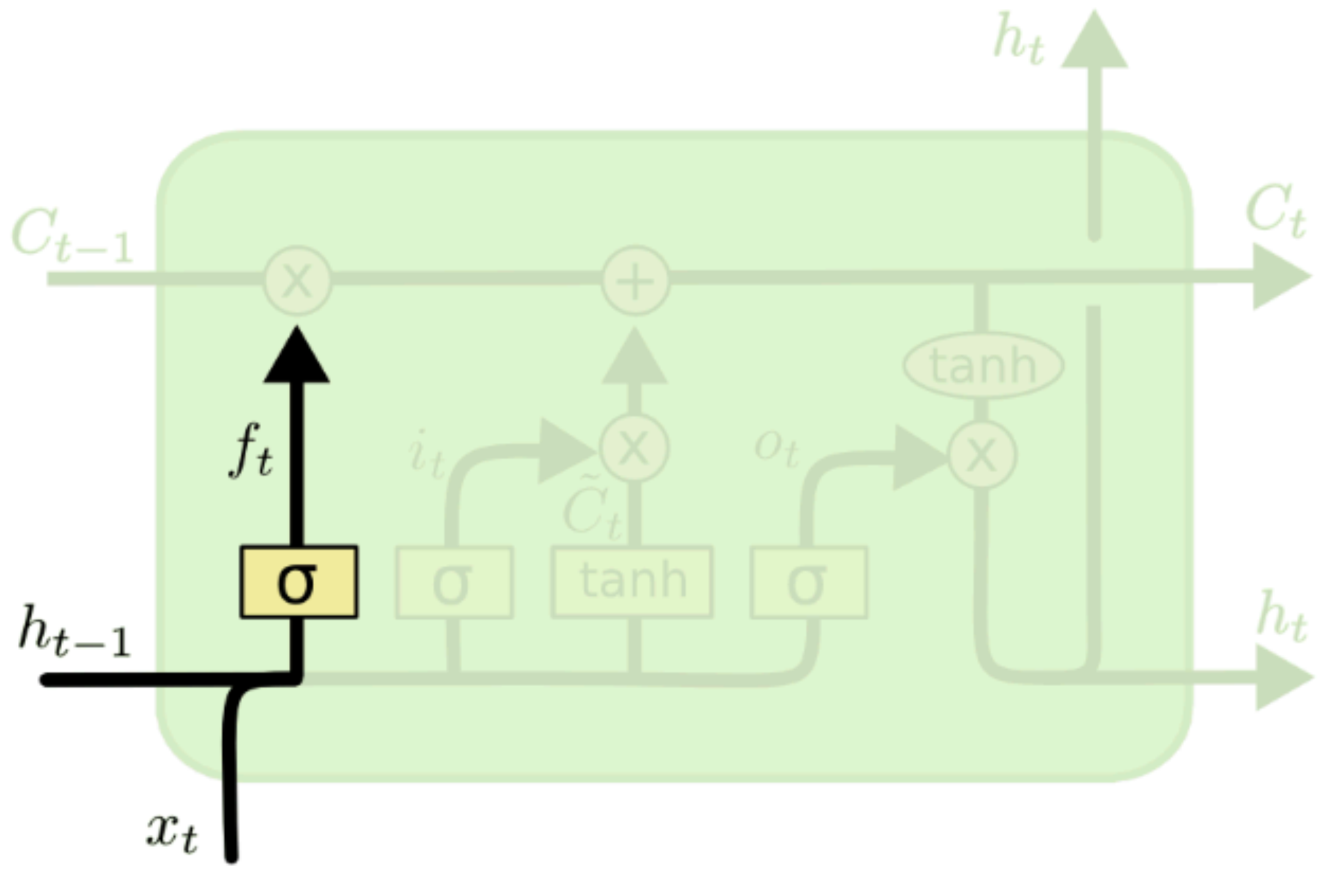
*Figures from *Colah’s blog: Understanding LSTM Networks*

Long-Short Term Memory (LSTM)

Architecture designed to remember information over many time steps and handle long-term dependencies



RNN



LSTM architecture

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

probability/portion to be kept,

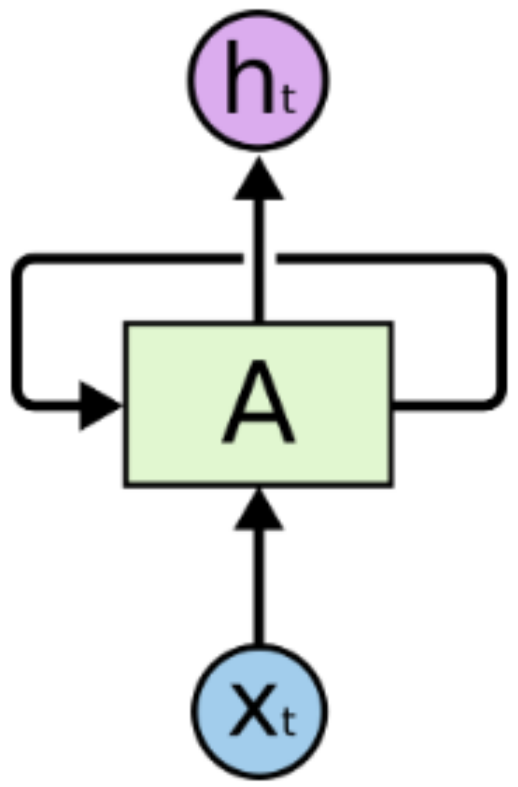
(one for each cell state dimension)

“Forget gate” regulates what to keep and what to discard from the previous cell state

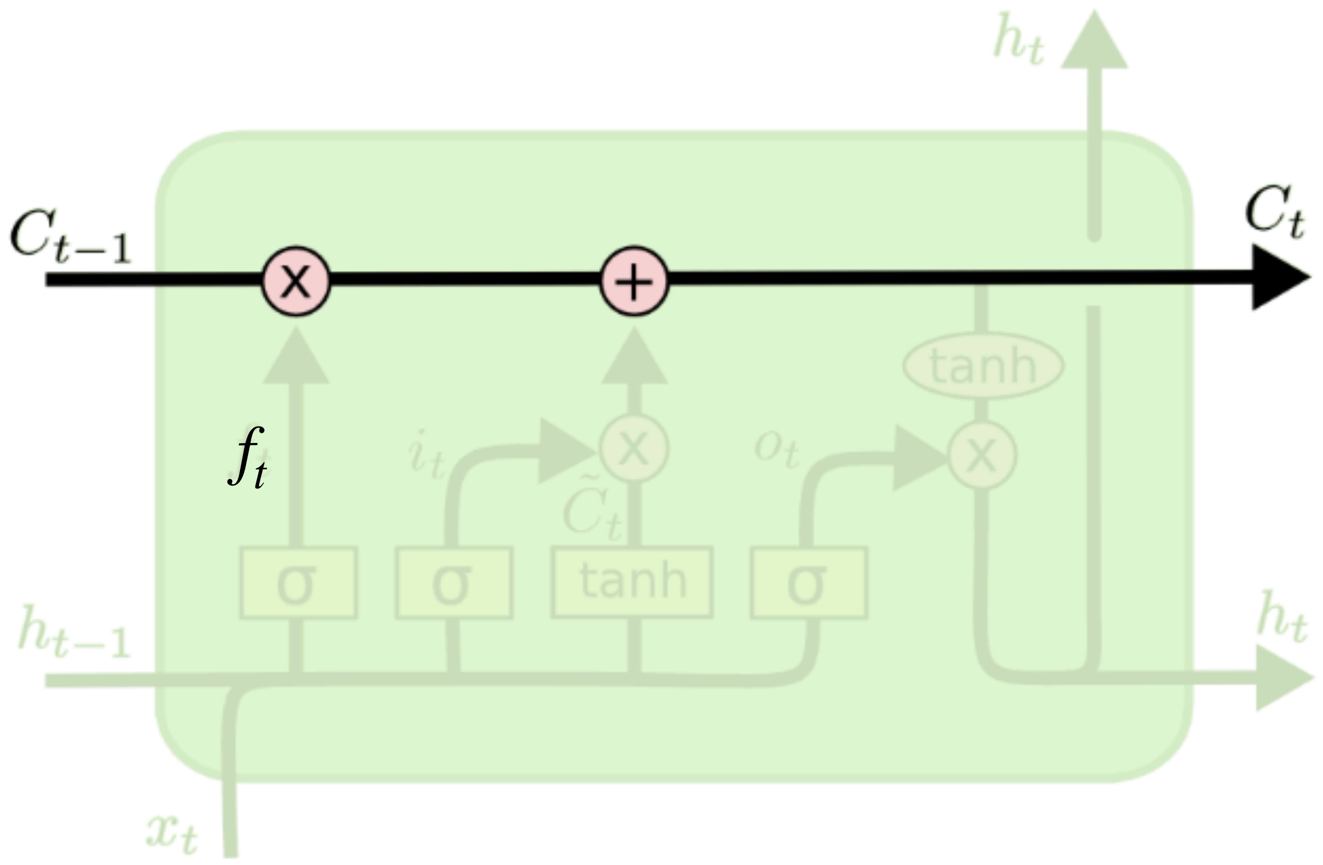
*Figures from *Colah’s blog: Understanding LSTM Networks*

Long-Short Term Memory (LSTM)

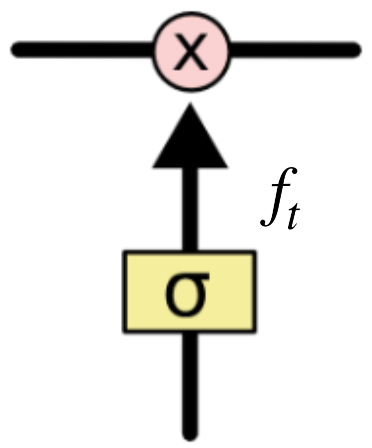
Architecture designed to remember information over many time steps and handle long-term dependencies



RNN



LSTM architecture



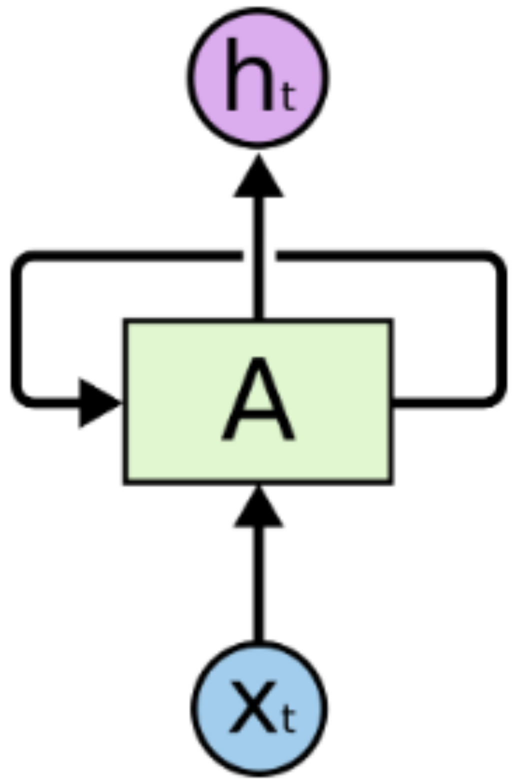
pointwise multiplication ⊙

“Forget gate” regulates what to keep and what to discard from the previous cell state

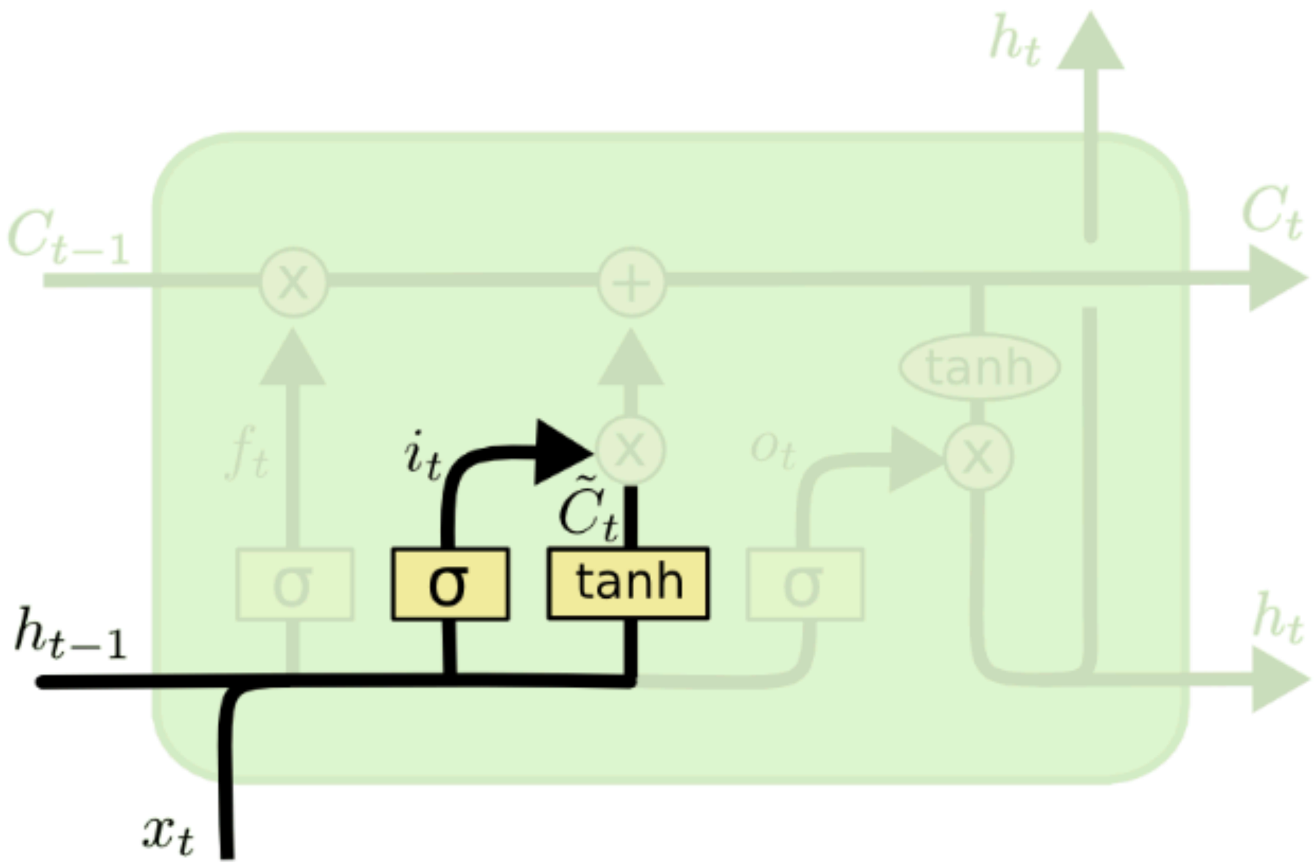
*Figures from *Colah’s blog: Understanding LSTM Networks*

Long-Short Term Memory (LSTM)

Architecture designed to remember information over many time steps and handle long-term dependencies



RNN



LSTM architecture

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

$$\tilde{C}_t = \tanh\left(w_c \cdot (h_{t-1}, x_t) + b_C\right)$$

i —> probability to be update

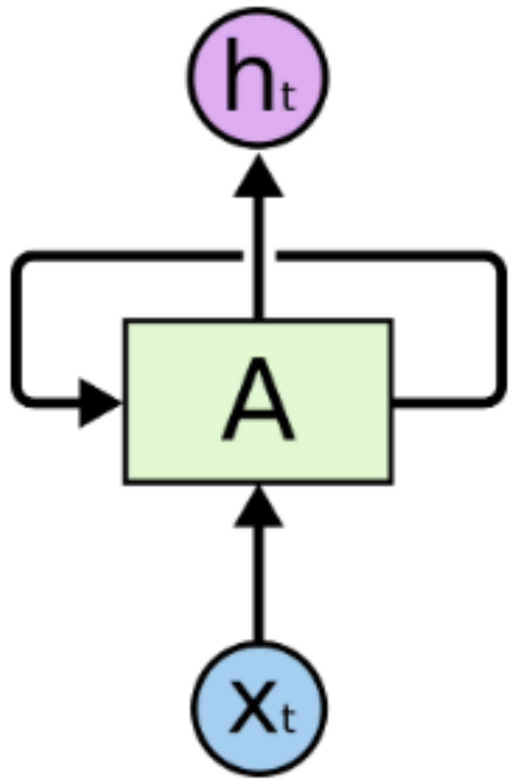
$\tilde{C}$—> candidate update

“Input gate” regulates what new information to add to the cell state

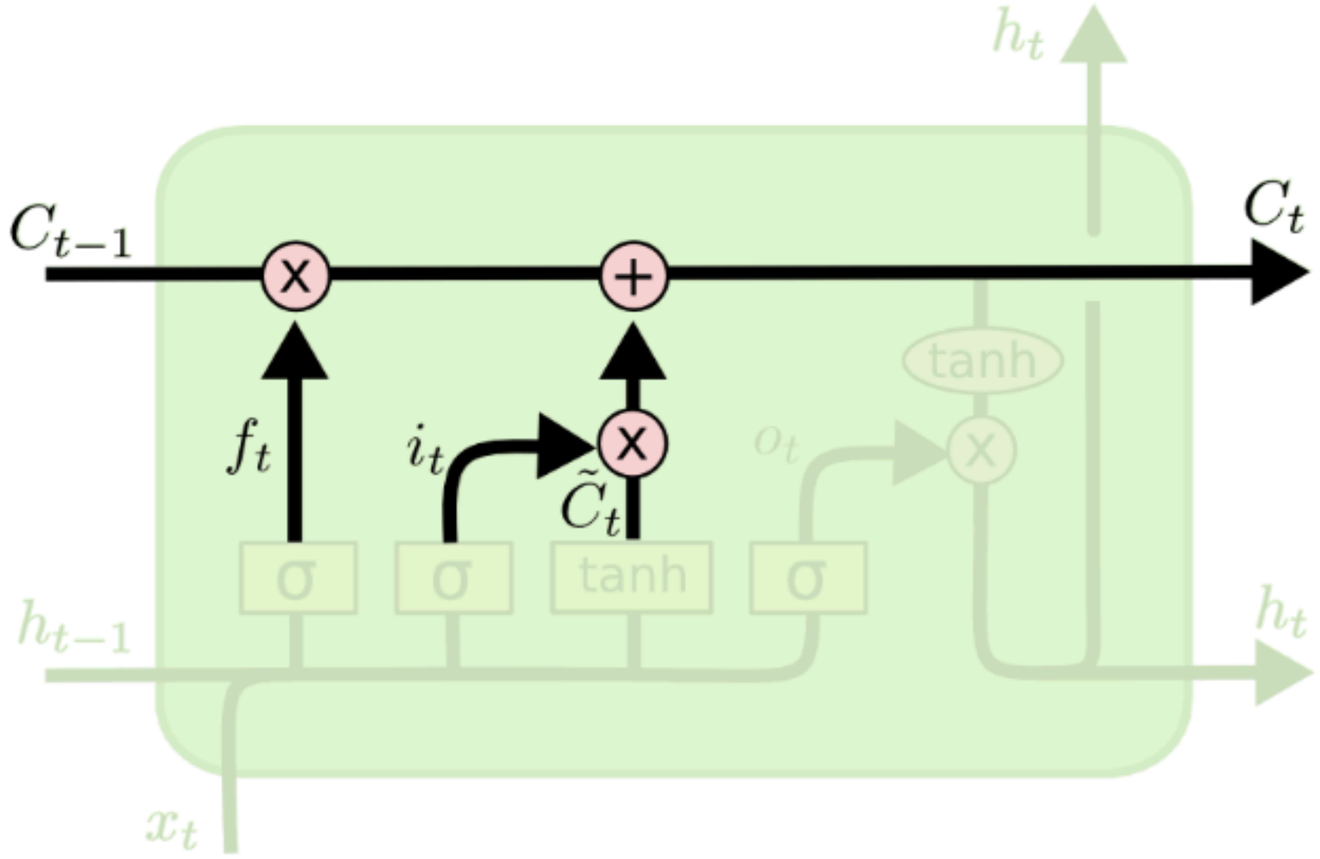
*Figures from *Colah’s blog: Understanding LSTM Networks*

Long-Short Term Memory (LSTM)

Architecture designed to remember information over many time steps and handle long-term dependencies



RNN



LSTM architecture

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

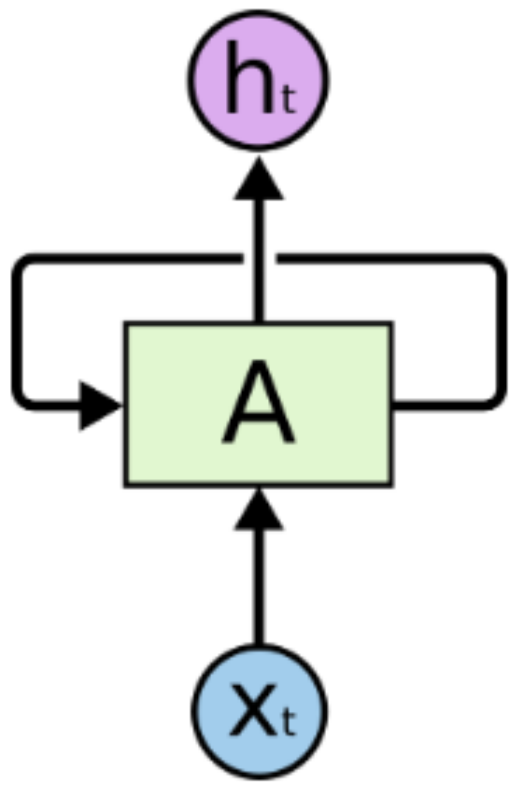
Forget or retain   Update

“Input gate” regulates what new information to add to the cell state

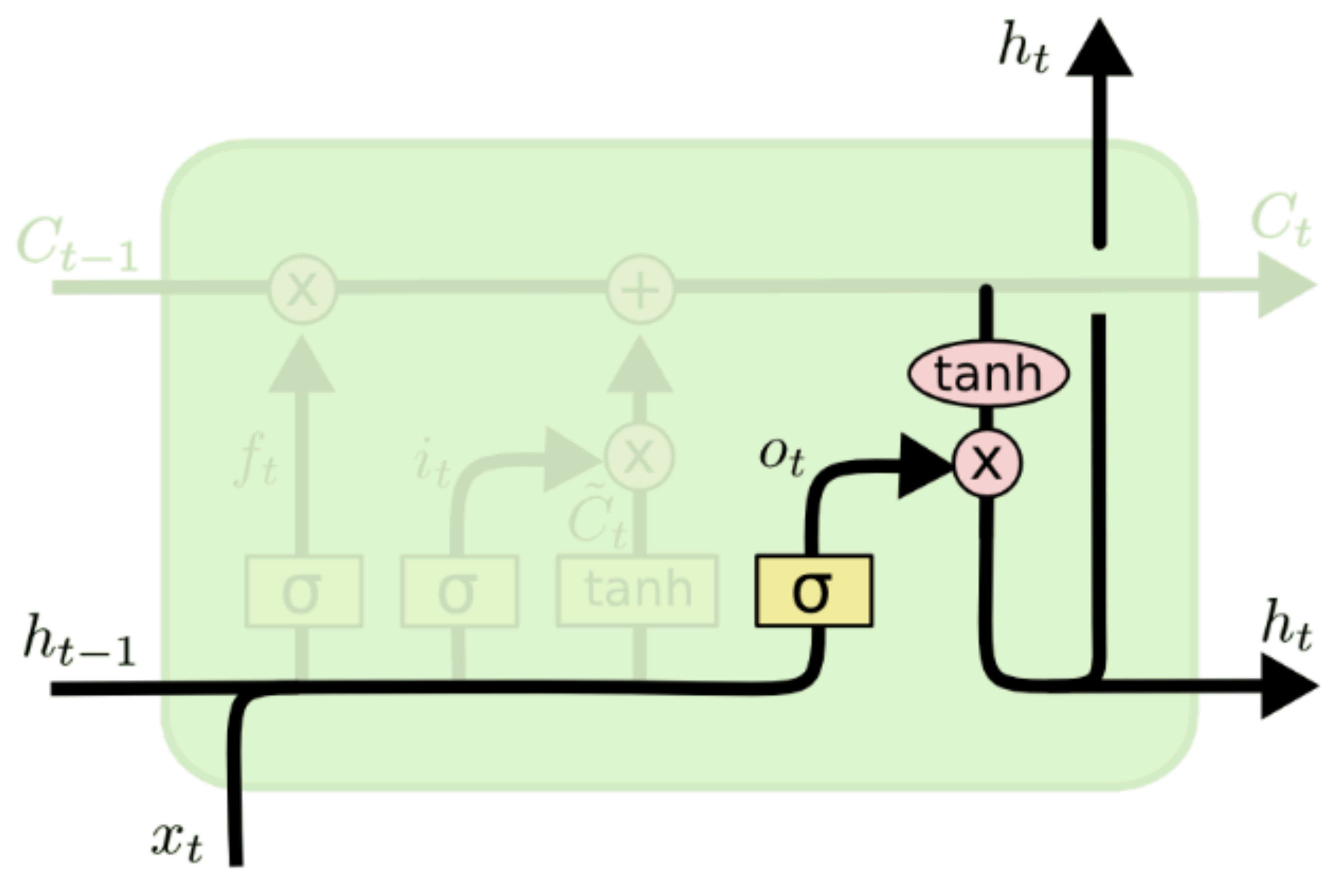
*Figures from *Colah’s blog: Understanding LSTM Networks*

Long-Short Term Memory (LSTM)

Architecture designed to remember information over many time steps and handle long-term dependencies



RNN



LSTM architecture

$o_t = \sigma(w_o \cdot (h_{t-1}, x_t) + b_o)$
$h_t = o_t \odot \tanh(C_t)$

Sigmoid decides what parts of the cell state to output

“Output gate” filters the new cell state to produce the output hidden state

*Figures from *Colah’s blog: Understanding LSTM Networks*

Summary

Topics

- Sequential data
- Recurrent Network Cells
- Backpropagation through time
- Vanishing and exploding gradients
- LSTM

Reading material

- Roger Grosse, Lecture Notes - Lecture 13 (section 2, 3), Lecture 15 (until 3.2 excluded)
- Colah's blog - Understanding LSTM Networks