Recurrent Neural Networks

(Many figures adapted from Stanford CS230, MIT 6.S191, and Illinois CS 498)

Outline

What Are Recurrent Neural Networks?

When to Use RNN?

Types of RNNs

A Simple Example of RNN Model

How to Train Recurrent Neural Networks?

The Problem of Long-Term Dependencies

Long Short Term Memory (LSTMs)

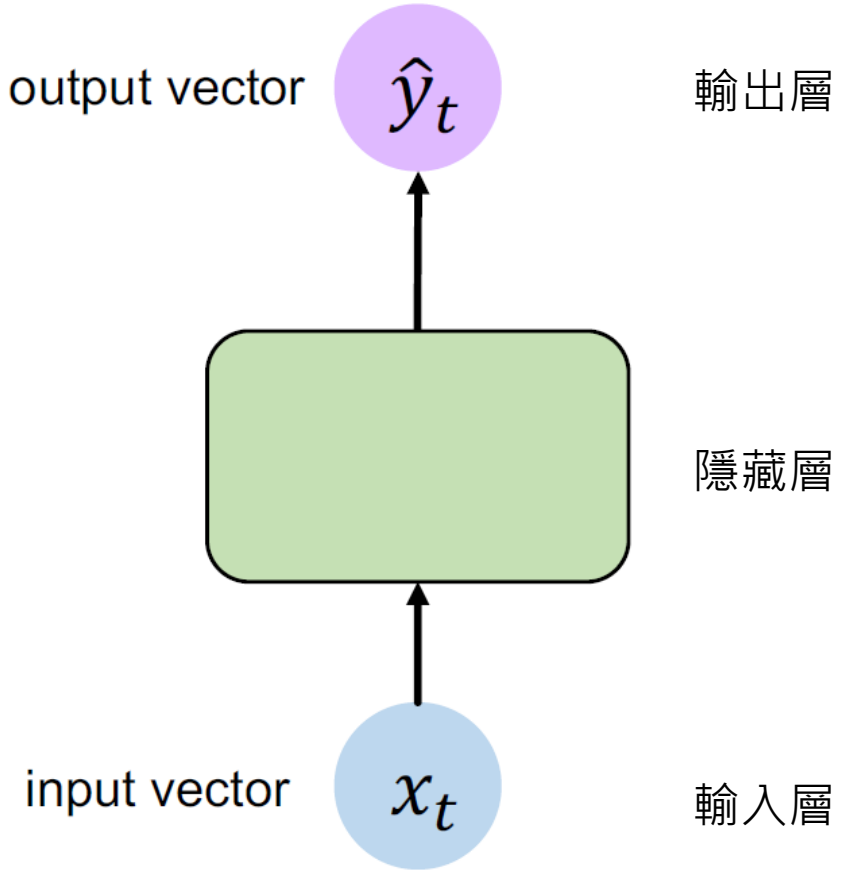
Summary

有順序(sequence)的概念
ex: types的字有順序的概念，如果順序調換就沒有types的意義

CNN主要用在空間上
RNN主要用在時間上(例如文字順序的資訊)的資料

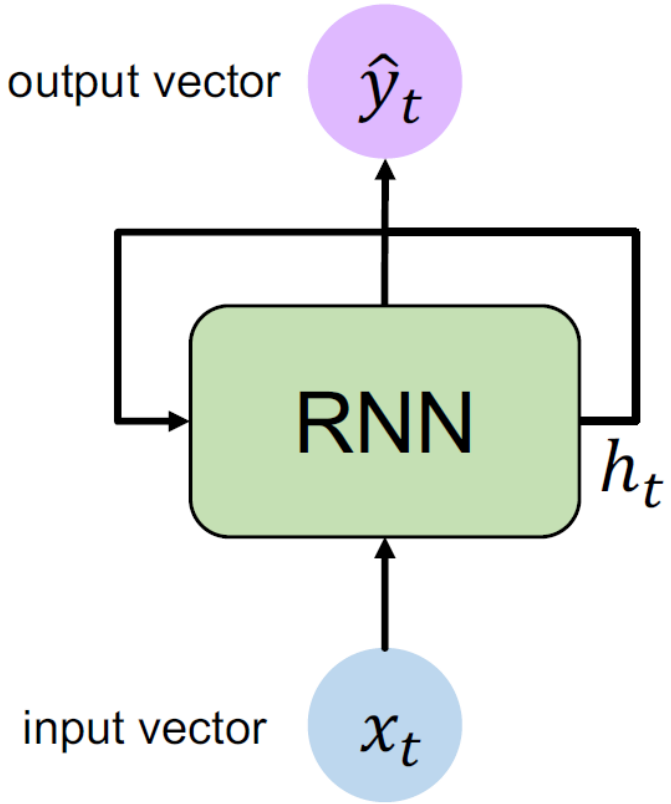
What Are Recurrent Neural Networks? (1/4)

- Feedforward neural networks
  - The data flow in one direction from the input layer to the outputs.
  - They have no feedback loops.



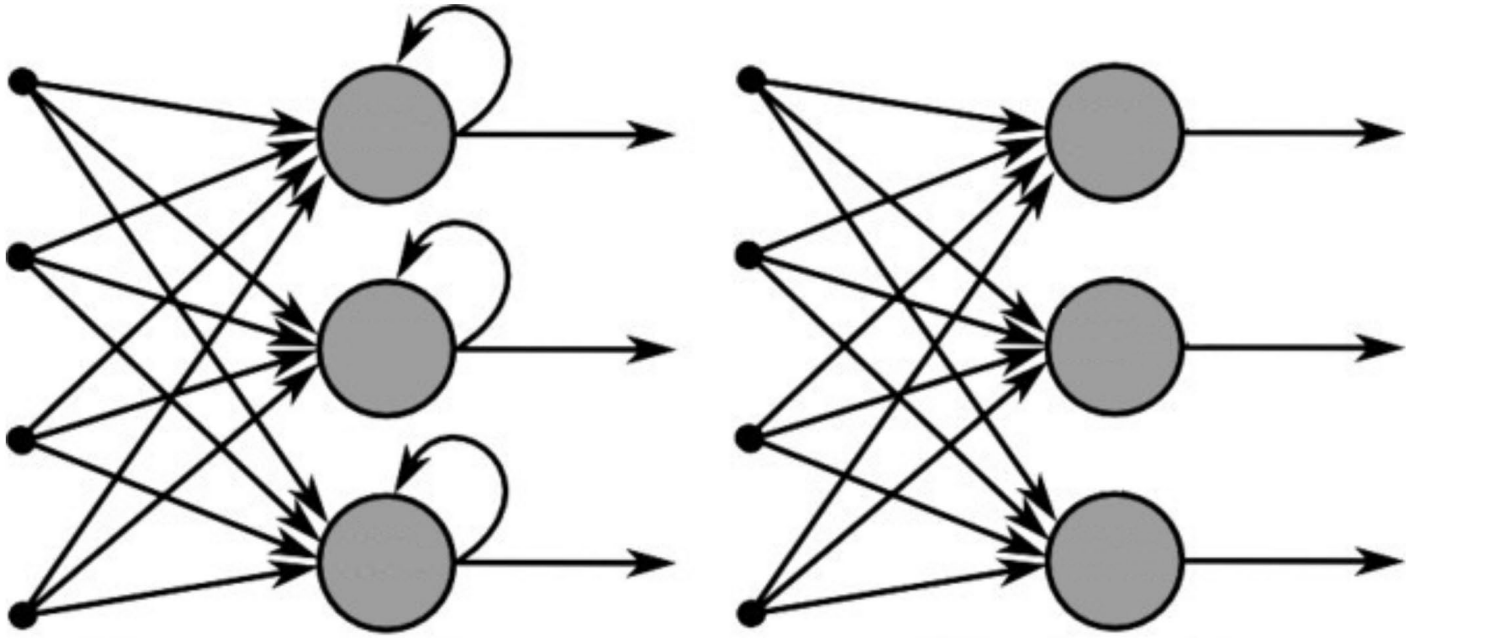
What Are Recurrent Neural Networks? (2/4)

- Recurrent neural networks (RNNs)
  - RNNs use inputs from previous stages to help a model remember its past.
  - This is usually shown as a feedback loop on hidden units.



隱藏層還會再回傳

What Are Recurrent Neural Networks? (3/4)



Recurrent Neural Network

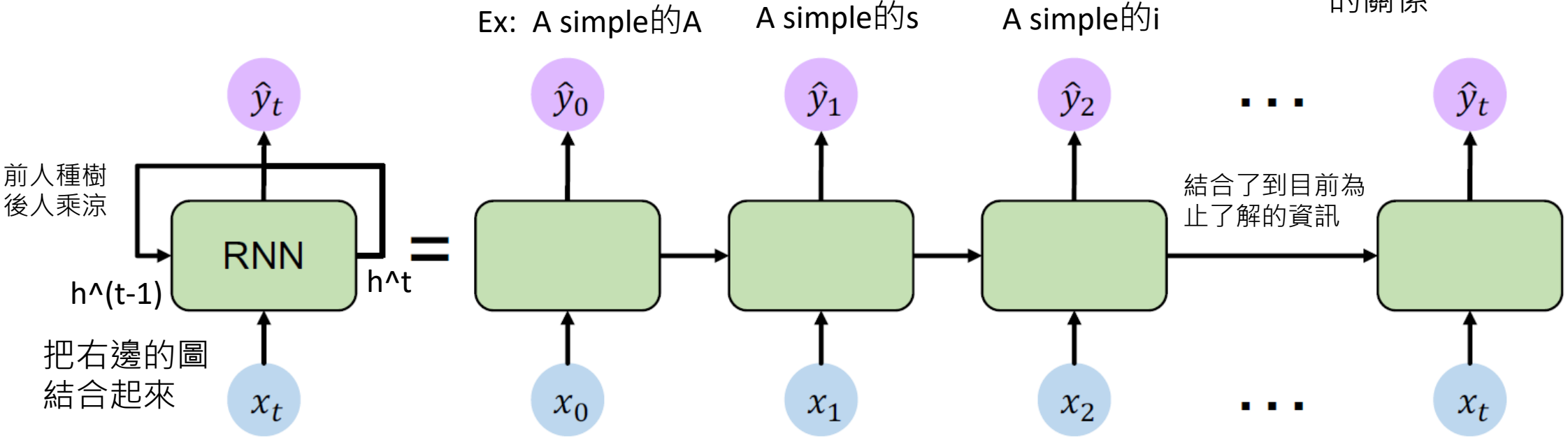
Feed-Forward Neural Network

Feeds the results back into the network

What Are Recurrent Neural Networks? (4/4)

- The output of a layer is added to the next input and fed back into the same layer.

可以更了解A跟simple的關係



第一個人沒有前面的資訊，我們寫h0的初始值為0(一般來說是向量)

輸入層不是一口氣餵進去

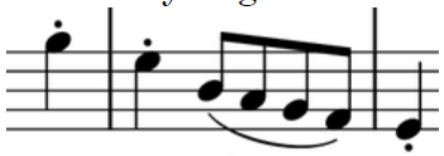
An unrolled recurrent neural network

When to Use RNN? (1/3)

- If the patterns in the data changes with time(有時序性), then RNN is the choice.
- More general, a problem requires to process a signal with a sequence structure.

When to Use RNN? (2/3)

Examples of sequence data

Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation	Ø 空集合	→	
Sentiment classification	"There is nothing to like in this movie."	→	
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

情緒

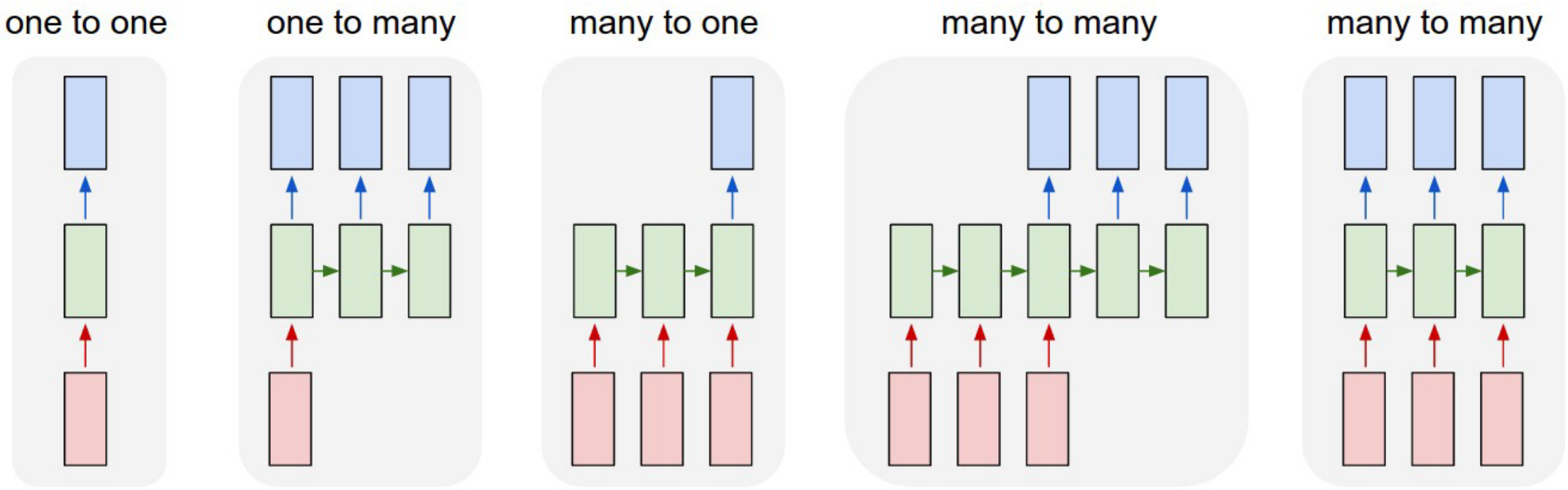
When to Use RNN? (3/3)

- Types of inputs and outputs

Task	Input	Output
Machine translation	Text	Text
Speech recognition	Audio	Text
Speech synthesis	Text	Audio
Sentiment analysis	Text	Categorical variable

影音的合成
(ex: input是文字，變成影音)

Types of RNNs (1/5)



Feed-forward Network

Music generation

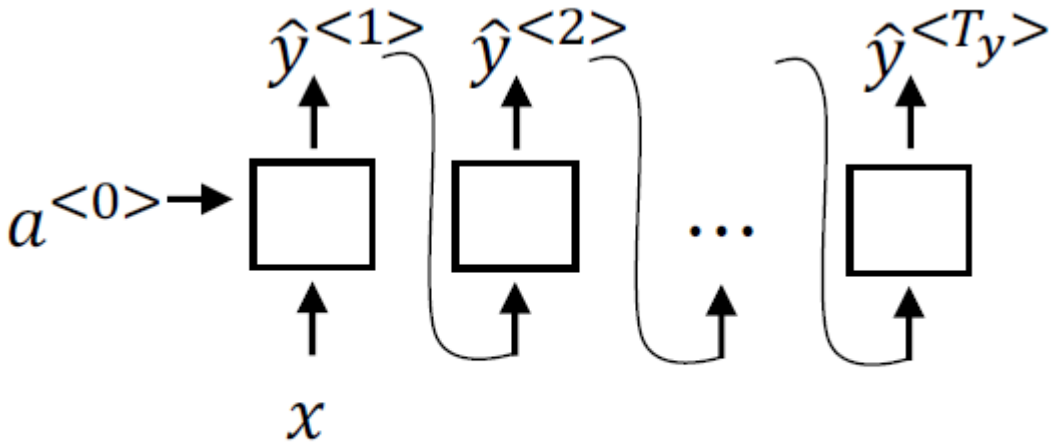
Sentiment classification

Machine translation

Name entity recognition

Types of RNNs (2/5)

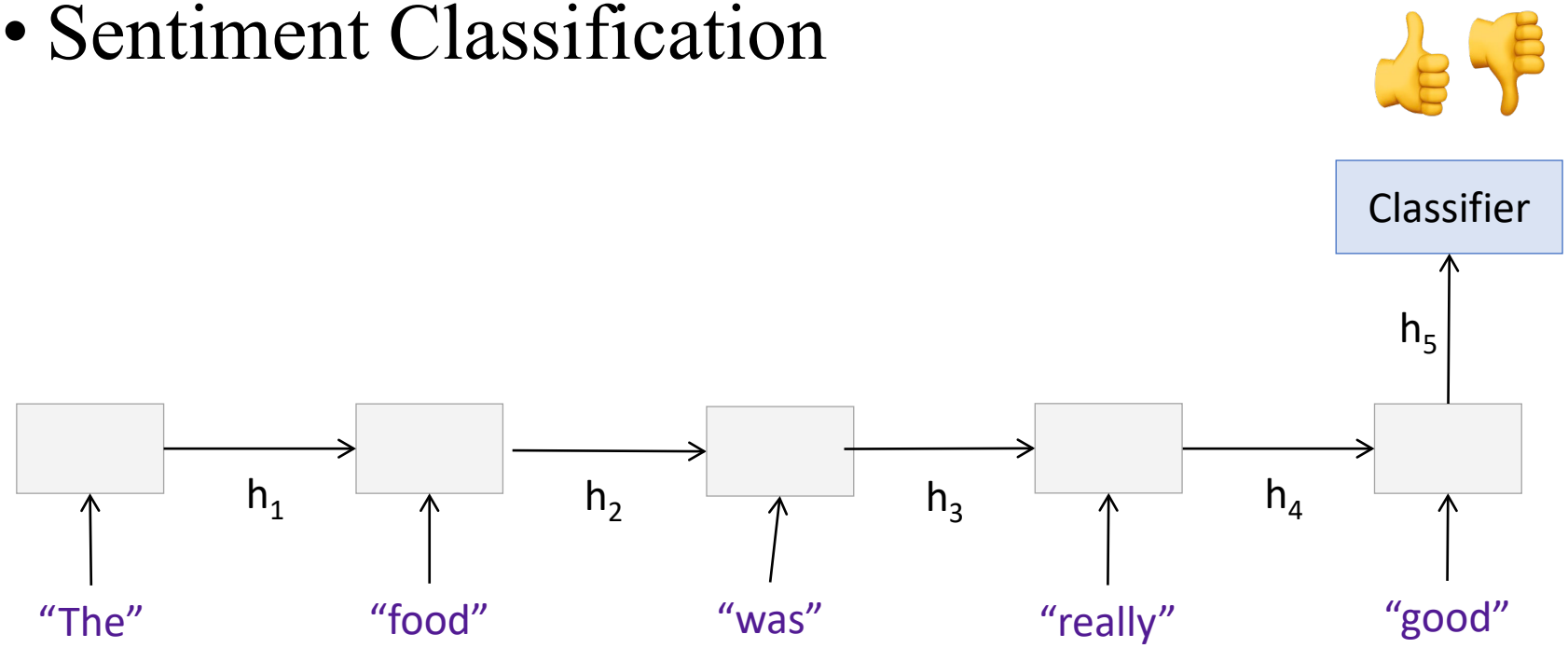
- Music generation



One to many (RNN)

Types of RNNs (3/5)

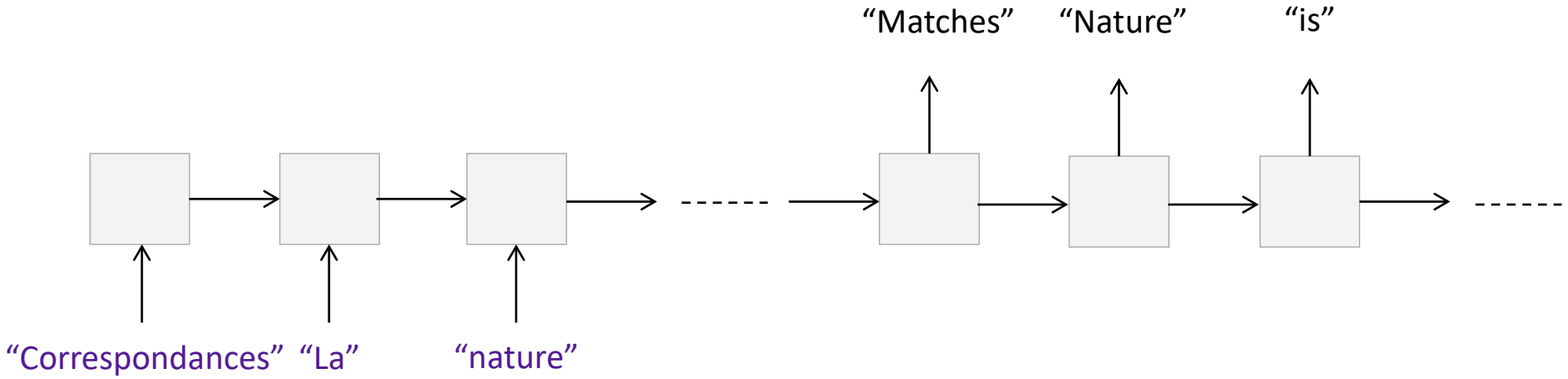
- Sentiment Classification



Many to one (RNN)

Types of RNNs (4/5)

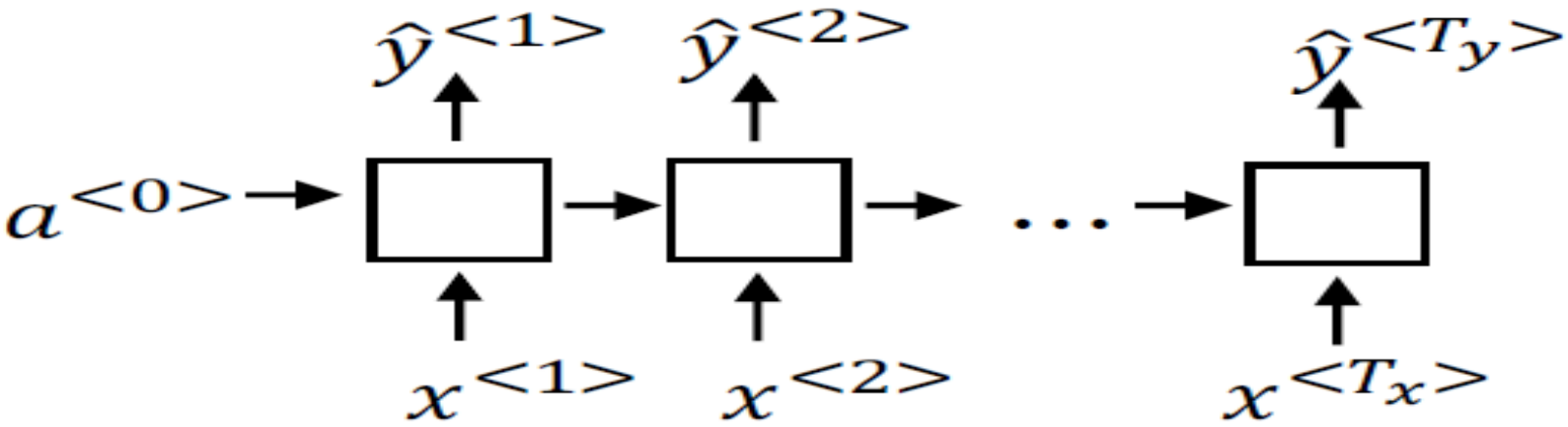
- Machine translation



Many to many (RNN)
(or sequence to sequence)

Types of RNNs (5/5)

- Name entity recognition
  - Harry Potter and Hermione Granger invented a new spell



Many to many (RNN)

A Simple Example of RNN Model (1/8)

- Name Entity Recognition

x: Luke Skywalker and Darth Vader fought an epic battle.

$x^{\langle 1 \rangle}$ $x^{\langle 2 \rangle}$ $x^{\langle 3 \rangle}$ … $x^{\langle t \rangle}$ … $x^{\langle 9 \rangle}$

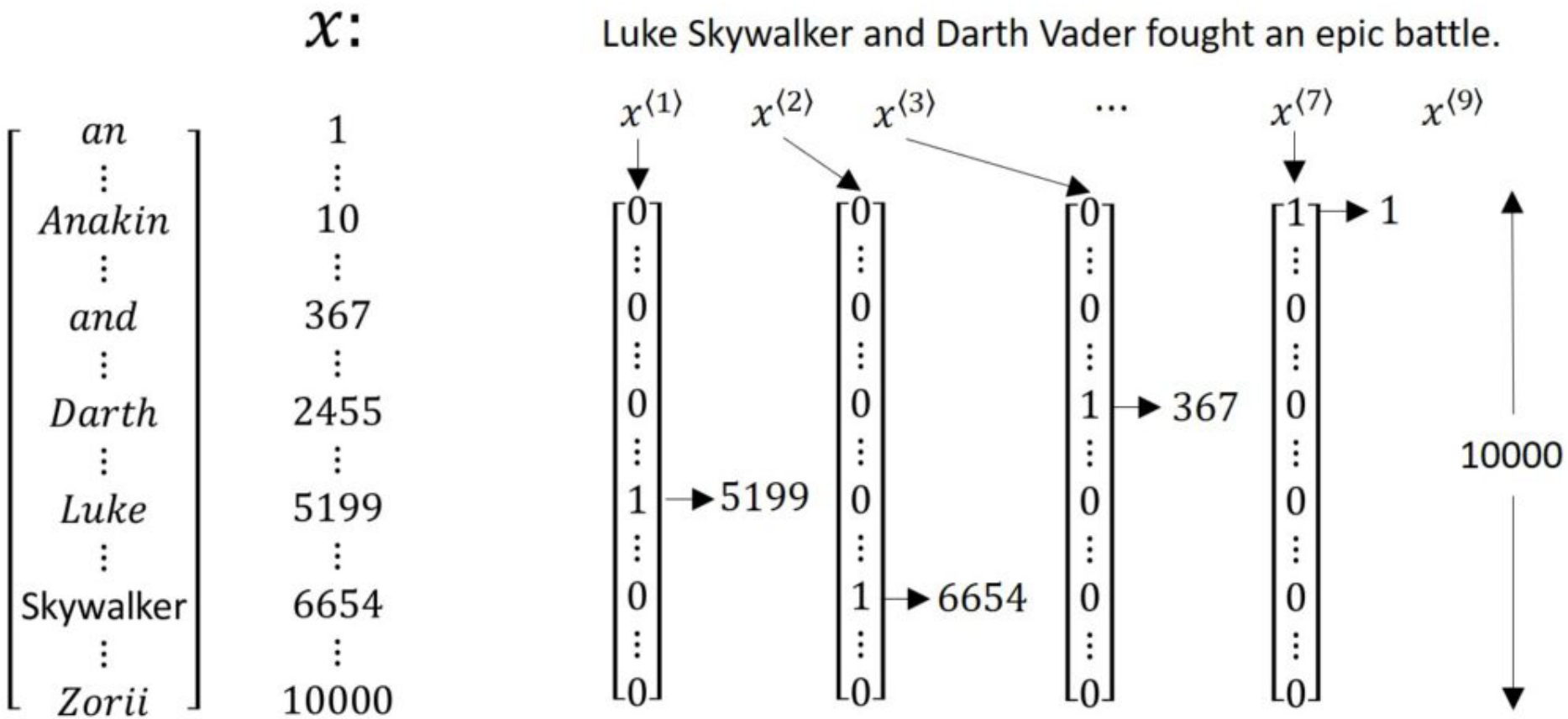
$T_x = 9$

y: 1 1 0 1 1 0 0 0 0

$y^{\langle 1 \rangle}$ $y^{\langle 2 \rangle}$ $y^{\langle 3 \rangle}$ … $y^{\langle t \rangle}$ … $y^{\langle 9 \rangle}$

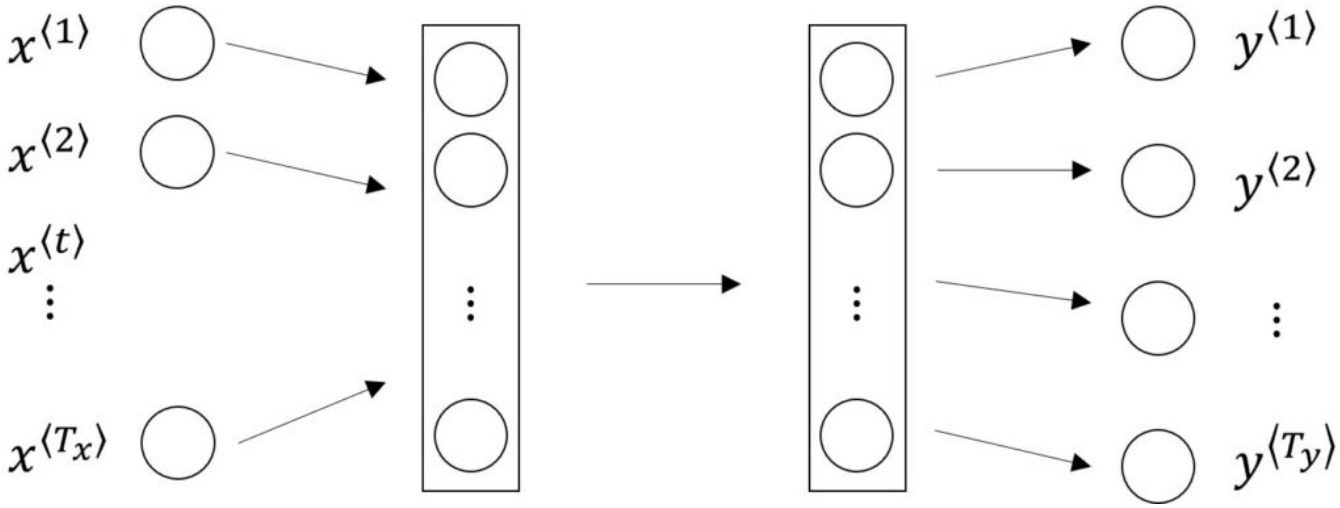
$T_y = 9$

A Simple Example of RNN Model (2/8)



A Simple Example of RNN Model (3/8)

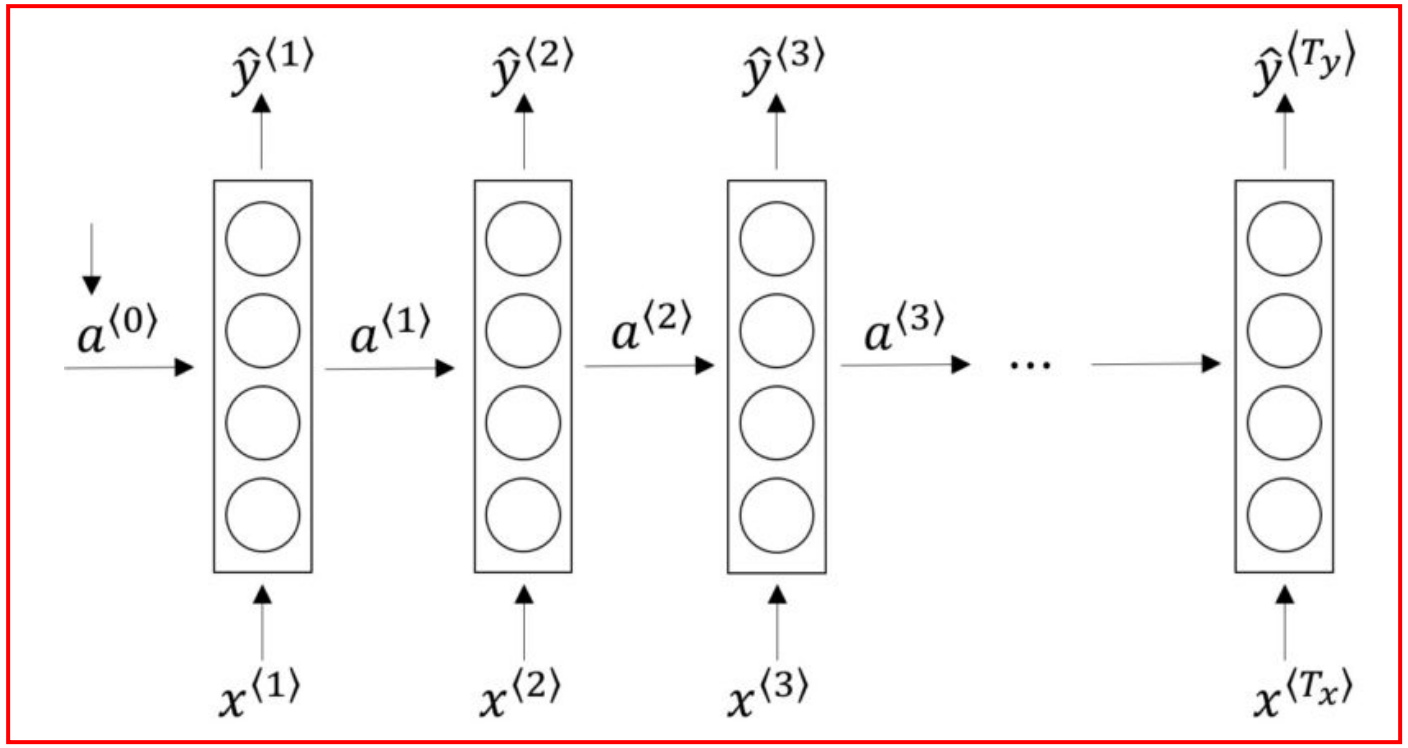
- Why not a standard network?



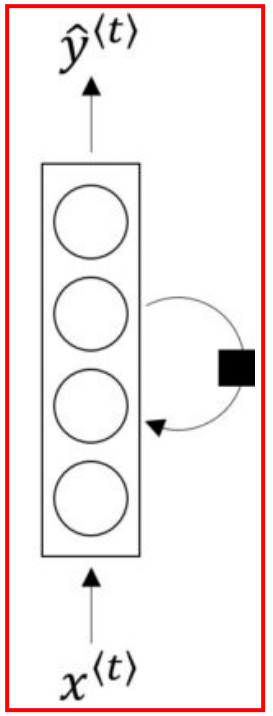
- Problems:
  - Inputs, outputs can be different lengths in different examples.
  - Doesn't share features learned across different positions

A Simple Example of RNN Model (4/8)

- The architecture of an RNN model



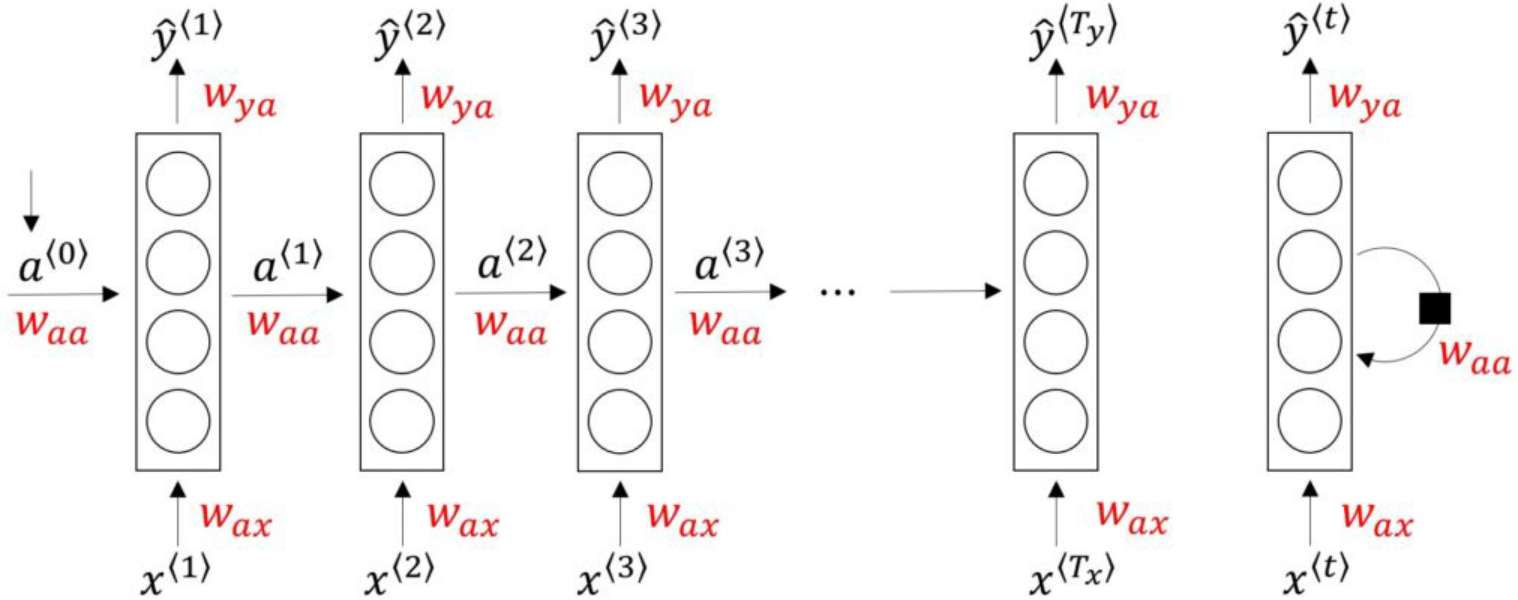
An unrolled recurrent neural network



RNN with loop

A Simple Example of RNN Model (5/8)

Forward Propagation



$$a^{\langle 1 \rangle} = g\left(w_{aa} \times a^{\langle 0 \rangle} + w_{ax} \times x^{\langle 1 \rangle} + b_a\right)$$

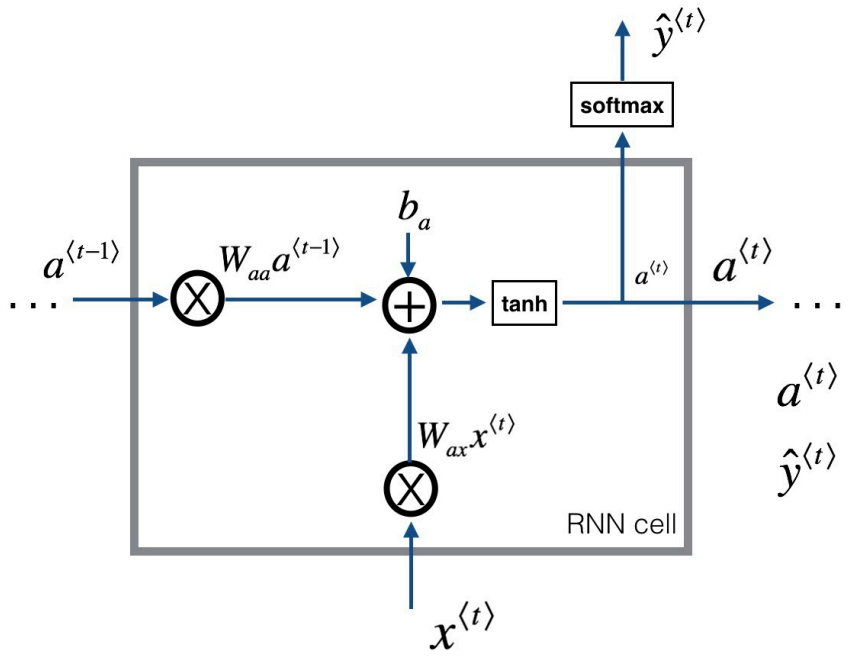
$$\hat{y}^{\langle 1 \rangle} = g\left(w_{ya} \times a^{\langle 1 \rangle} + b_y\right)$$

$$a^{\langle t \rangle} = g\left(w_{aa} \times a^{\langle t-1 \rangle} + w_{ax} \times x^{\langle t \rangle} + b_a\right)$$

$$\hat{y}^{<t>} = g\left(W_{ya} a^{<t>} + b_y\right)$$

A Simple Example of RNN Model (6/8)

An example of RNN cell: $g_1 = \tanh$, $g_2 = \text{softmax}$



$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = soft\max(W_{ya}a^{\langle t \rangle} + b_y)$$

A Simple Example of RNN Model (7/8)

- RNN notation

$$a^{\langle t \rangle} = g\left(w_{aa} \times a^{\langle t-1 \rangle} + w_{ax} \times x^{\langle t \rangle} + b_a\right)$$

g 是 activation function

$$\hat{y}^{<t>} = g\left(W_{ya} a^{<t>} + b_y\right)$$

- Simplified RNN notation for $a^{<t>}$:

$$a^{<t>} = g(w_a[a^{<t-1>}, x^{<t>}] + b_a)$$

$$\overset{100}{\updownarrow}\underset{\leftrightarrow\ 100}{[W_{aa}}\quad \underset{\leftrightarrow\ 10000}{W_{ax}}] = \underset{(100,10100)}{w_a} \qquad [a^{\langle t-1 \rangle} \quad x^{\langle t \rangle}] = \begin{bmatrix} a^{\langle t-1 \rangle} \\ x^{\langle t \rangle} \end{bmatrix} \begin{matrix} \updownarrow 100 \\ \updownarrow 10000 \end{matrix} \updownarrow 10100$$

A Simple Example of RNN Model (8/8)

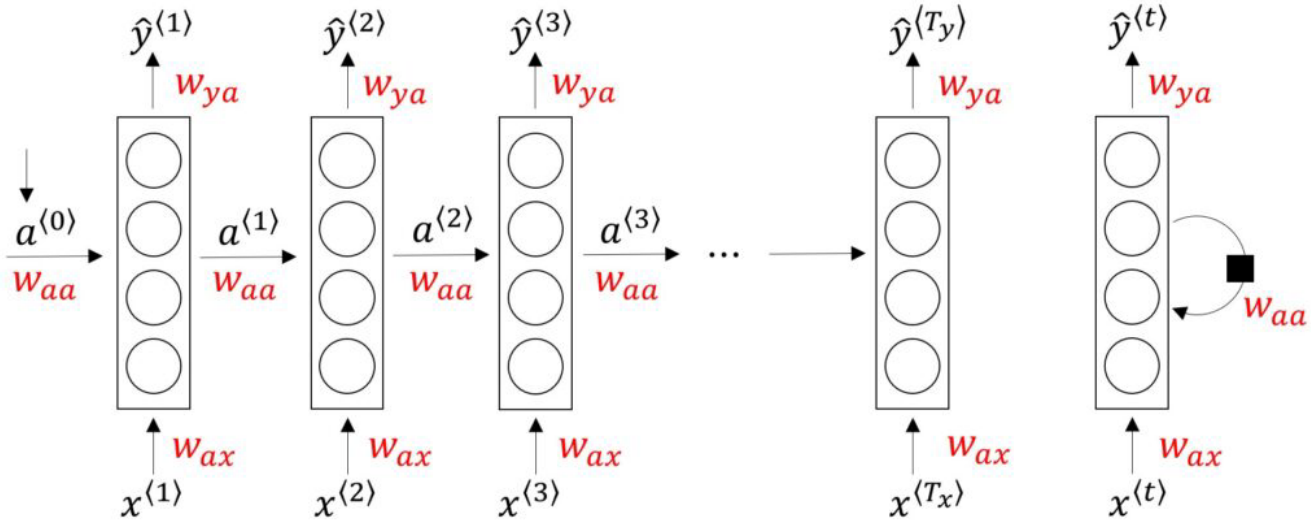
- Summary

  Forward Propagation

$$a^{<t>} = g_1\left(w_a[a^{<t-1>}, x^{<t>}] + b_a\right)$$

$$\underset{100}{\updownarrow}\left[\underset{100}{W_{aa}} \quad \underset{10000}{W_{ax}}\right] = \underset{(100,10100)}{W_a}$$

$$\hat{y}^{<t>} = g_2(w_y a^{<t>} + b_y)$$

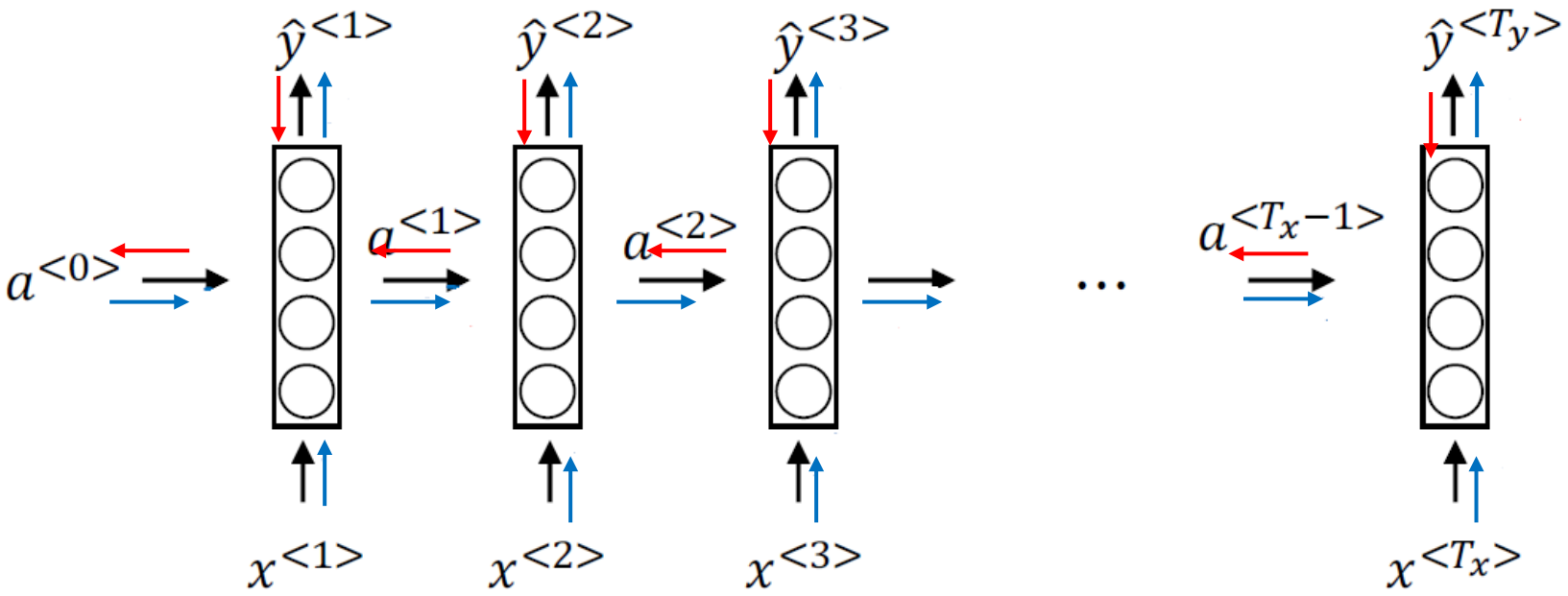


How to Train Recurrent Neural Networks? (1/3)

- Recurrent neural networks use backpropagation algorithm for training, but it is applied for every timestamp.
- It is commonly known as Backpropagation Through Time (BPTT).

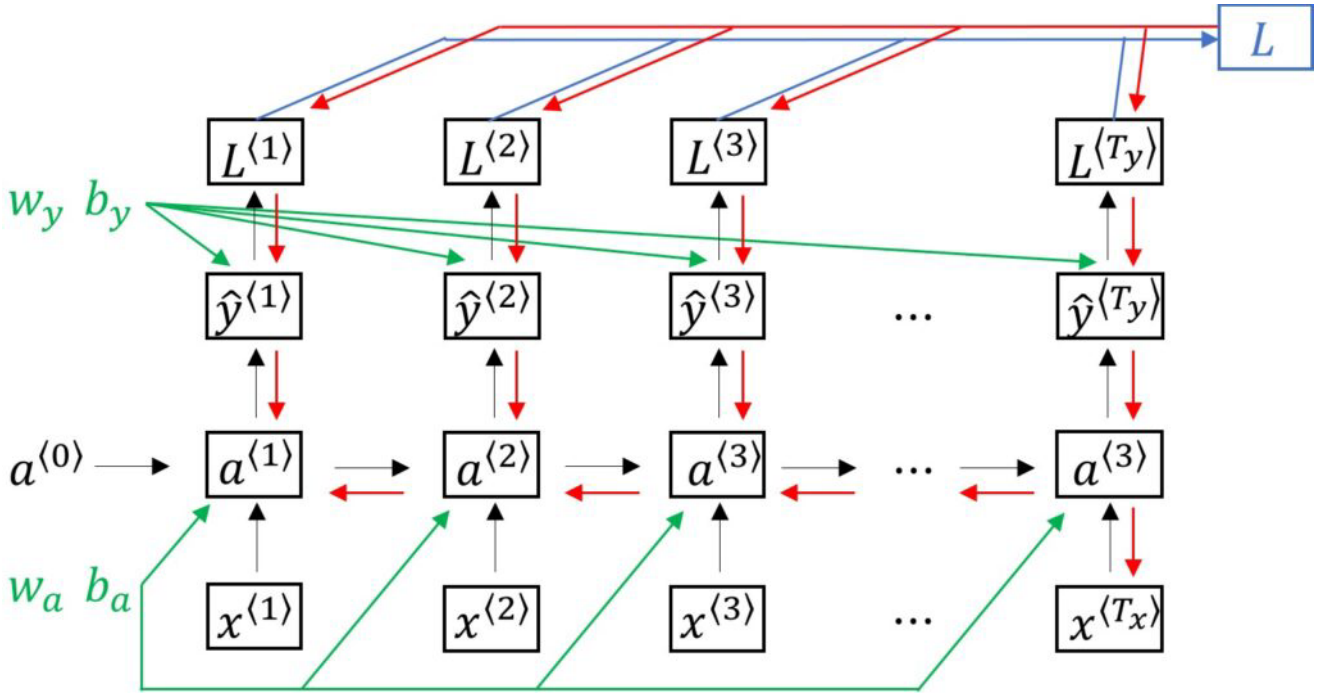
How to Train Recurrent Neural Networks? (2/3)

- Forward propagation (→) and backpropagation (→)



How to Train Recurrent Neural Networks? (3/3)

- BackpropagationThroughTime (BPTT)



$$L^{\langle t \rangle}\left(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}\right) = -y^{\langle t \rangle} log\hat{y}^{\langle t \rangle} - \left(1 - y^{\langle t \rangle}\right) log\left(1 - \hat{y}^{\langle t \rangle}\right)$$

$$L(\hat{y}, y) = \sum_{t=1}^{T_x} L^{\langle t \rangle}\left(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}\right)$$

The Problem of Long-Term Dependencies (1/3)

Why are vanishing gradients a problem?

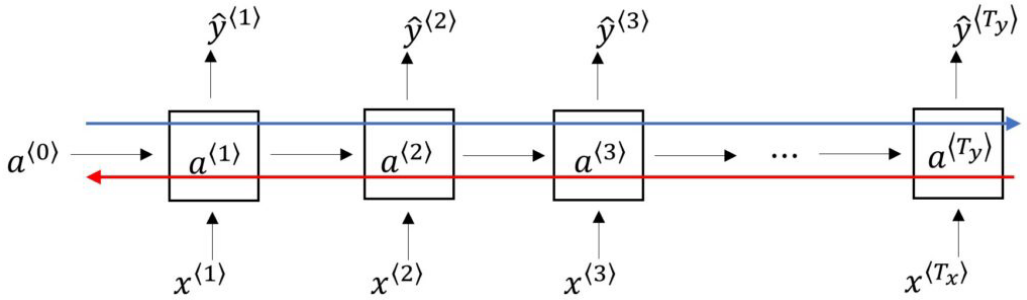
Multiply many small numbers together

↓

Errors due to further back time steps have smaller and smaller gradients

↓

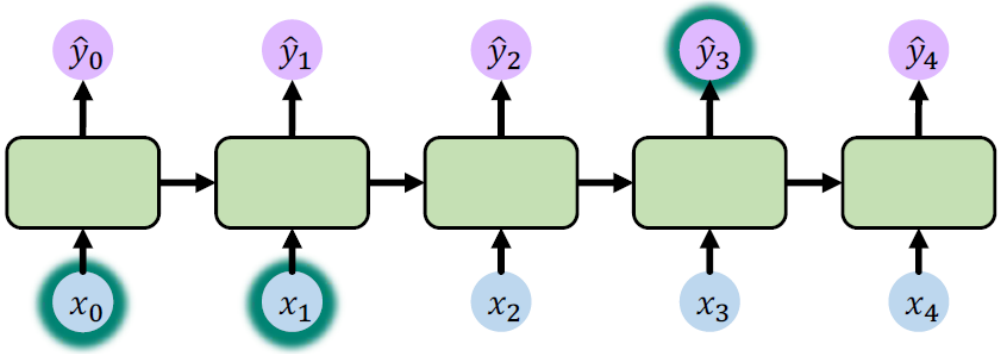
Influence parameters to capture short-term dependencies



Note:
For the exploding gradients problem, use gradient clipping to scale big gradients.

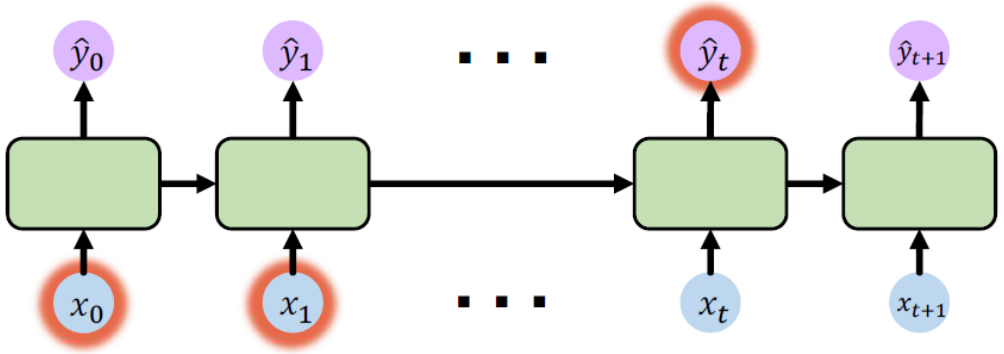
The Problem of Long-Term Dependencies (2/3)

"The clouds are in the ___"



RNN may work out for this short-term dependency case

"I grew up in France, … and I I speak fluent___ "



RNN may cause a problem for this long-term dependency case

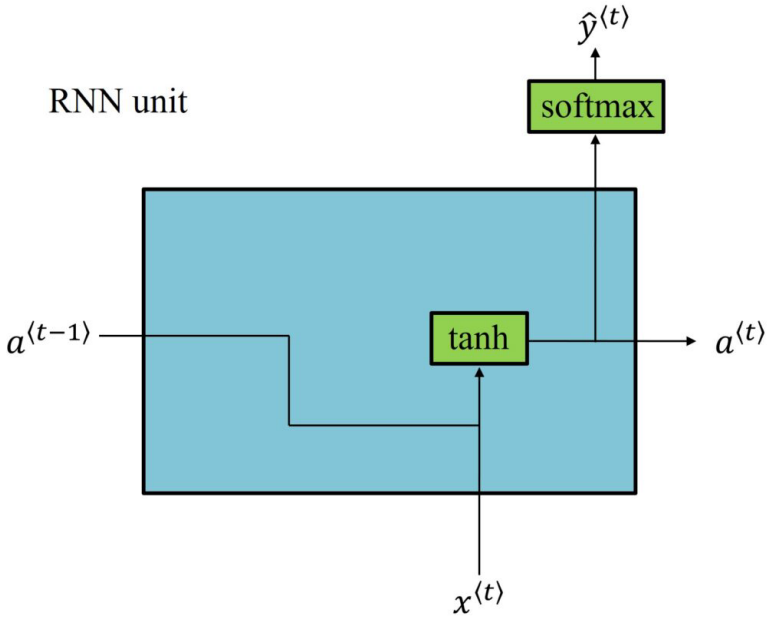
The Problem of Long-Term Dependencies (3/3)

- Two popular RNN architectures:
  - Long Short-Term Memory Networks (LSTMs)
  - Gated Recurrent Units (GRUs)
- Both aim to remember long-term dependencies while alleviating the vanishing and exploding gradient problems.
- These architectures use gated modules to keep what's important in a sequence of data points.

Long Short Term Memory (LSTMs)

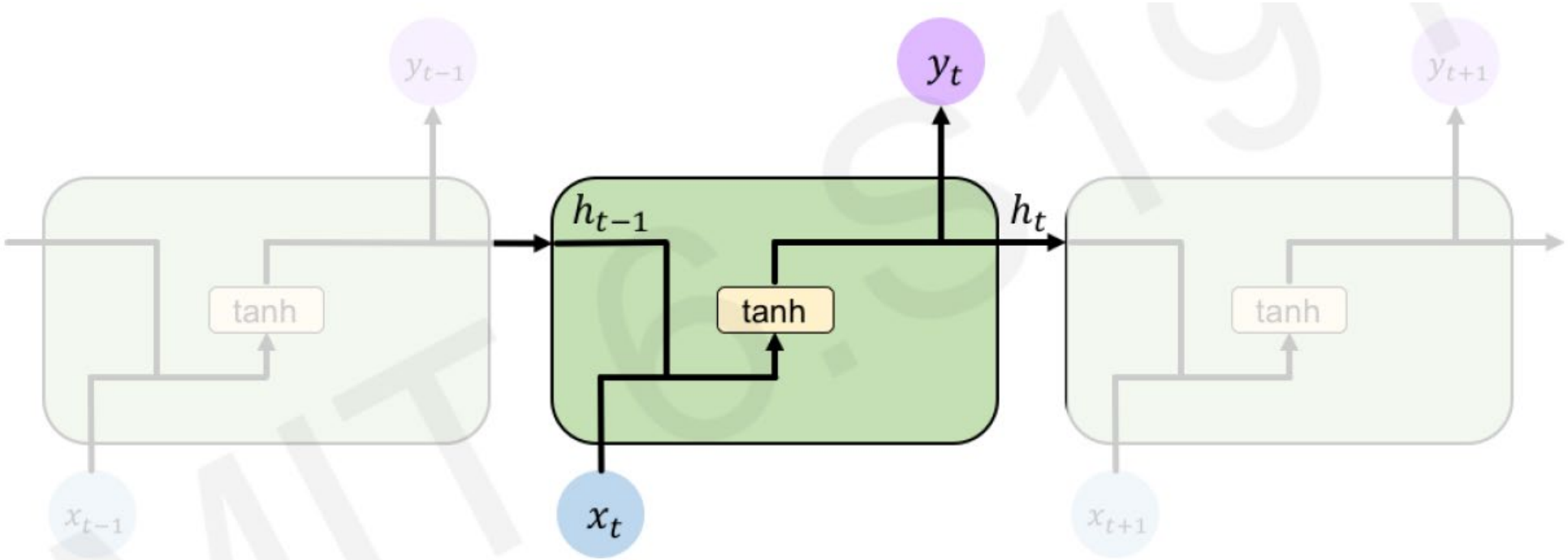
- Basic RNN
  - $a^{<t>} = \tanh(w_a[a^{<t-1>}, x^{<t>}] + b_a)$
  - $\hat{y}^{<t>} = \text{softmax}(w_y a^{<t>} + b_y)$

  : Repeating modules contain a simple computation node



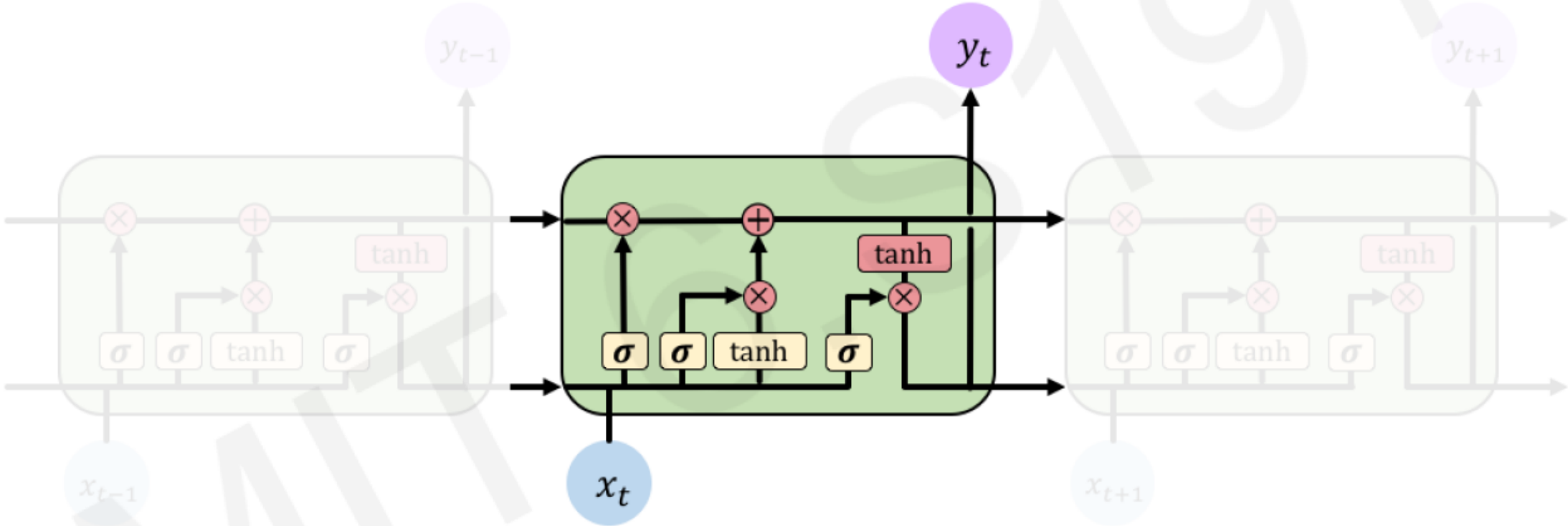
Long Short Term Memory (LSTMs)

- Basic RNN (New Notation)



Long Short Term Memory (LSTMs)

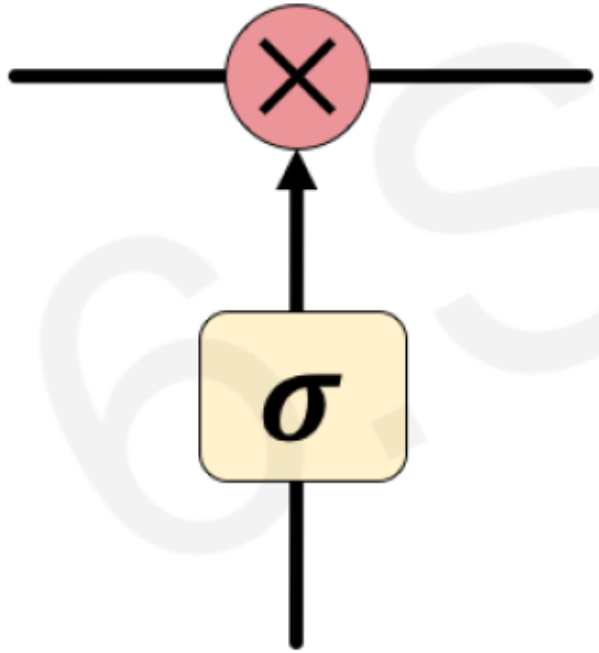
LSTM modules contain **computational blocks** that **control information flow**



LSTM cells are able to track information throughout many timesteps

Long Short Term Memory (LSTMs)

Information is **added** or **removed** through structures called **gates**

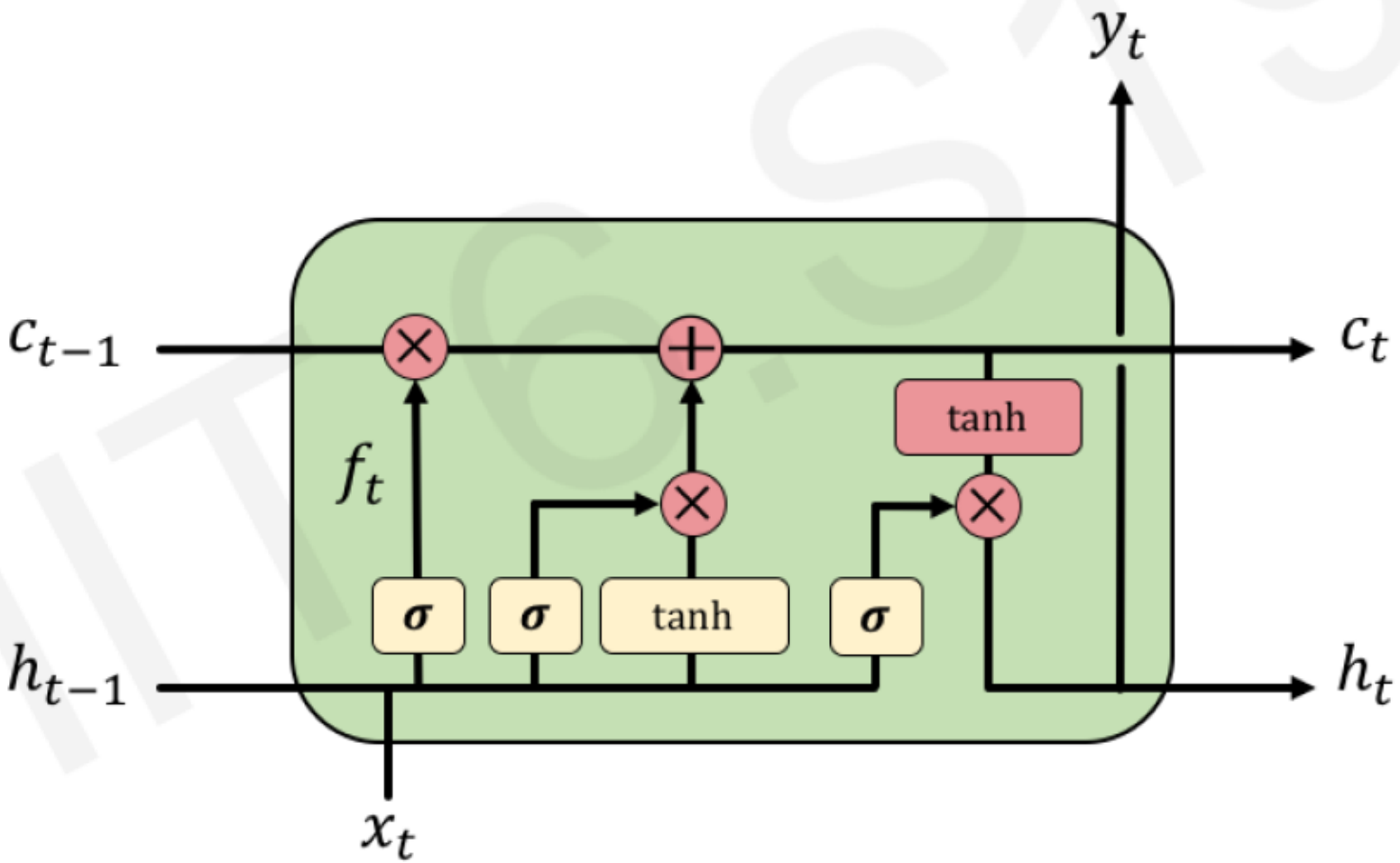


Gates optionally let information through, for example via a sigmoid neural net layer and pointwise multiplication

Long Short Term Memory (LSTMs)

How do LSTMs work?

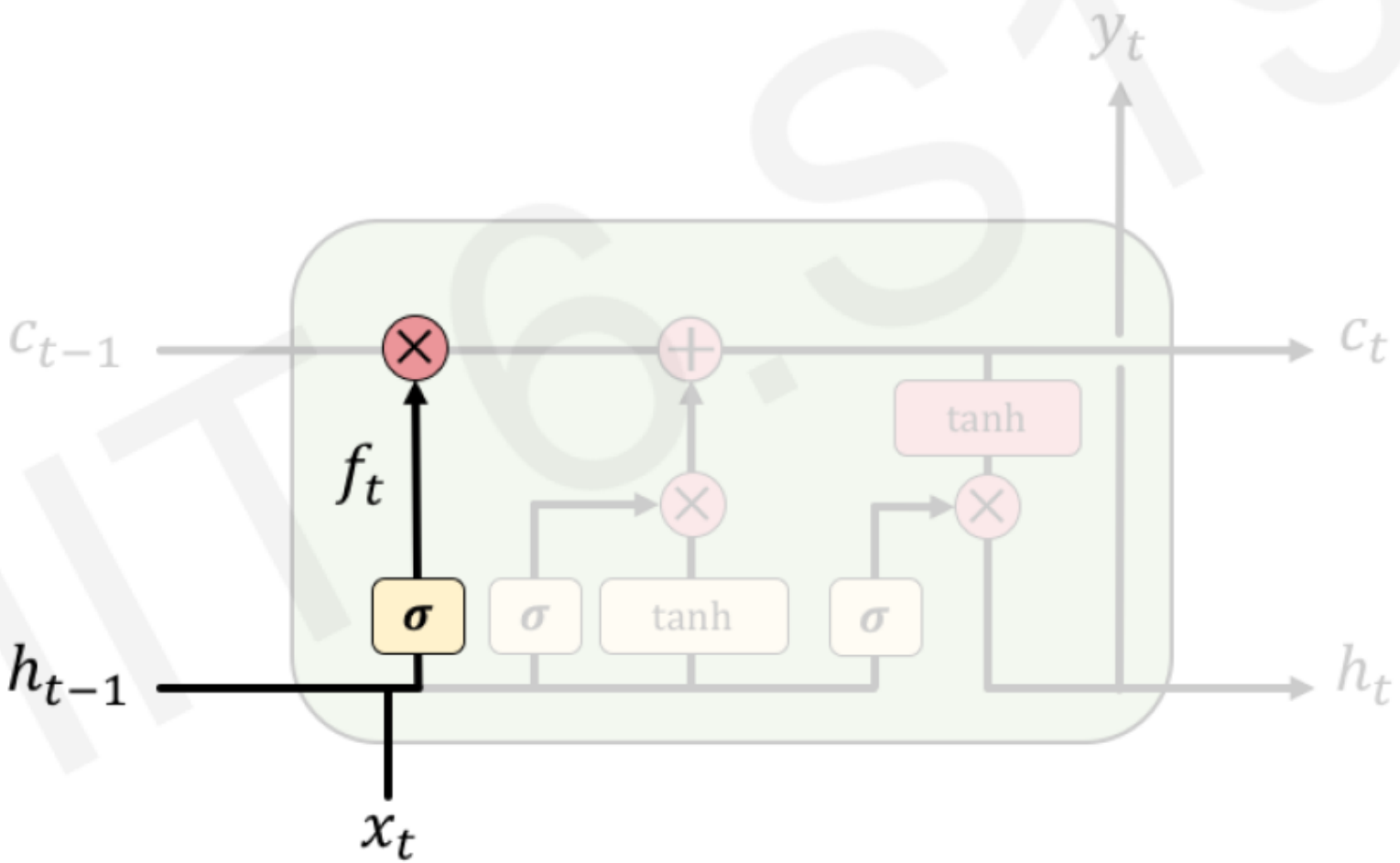
1) Forget 2) Store 3) Update 4) Output



Long Short Term Memory (LSTMs)

1) Forget 2) Store 3) Update 4) Output

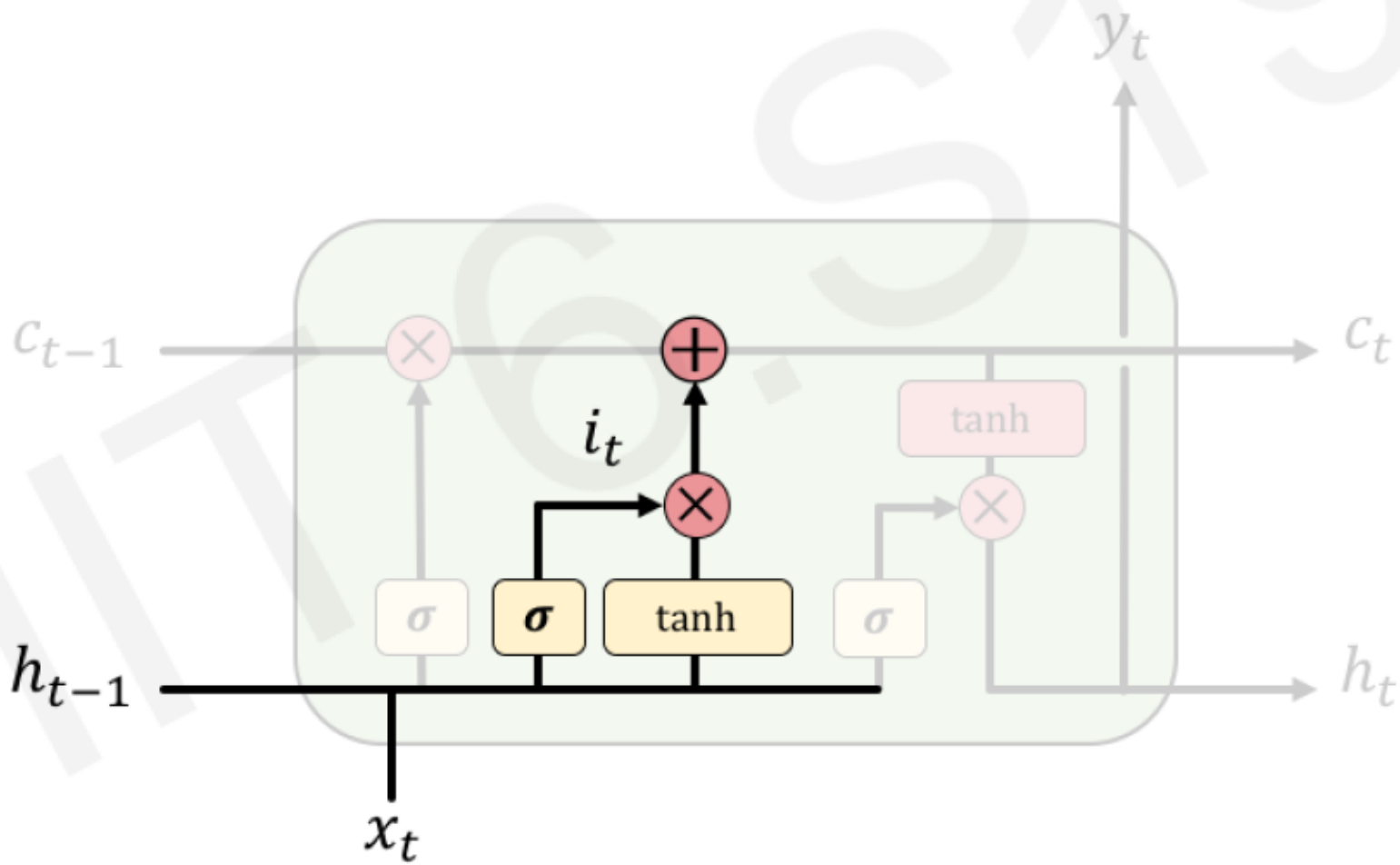
LSTMs **forget irrelevant** parts of the previous state



Long Short Term Memory (LSTMs)

1) Forget **2) Store** 3) Update 4) Output

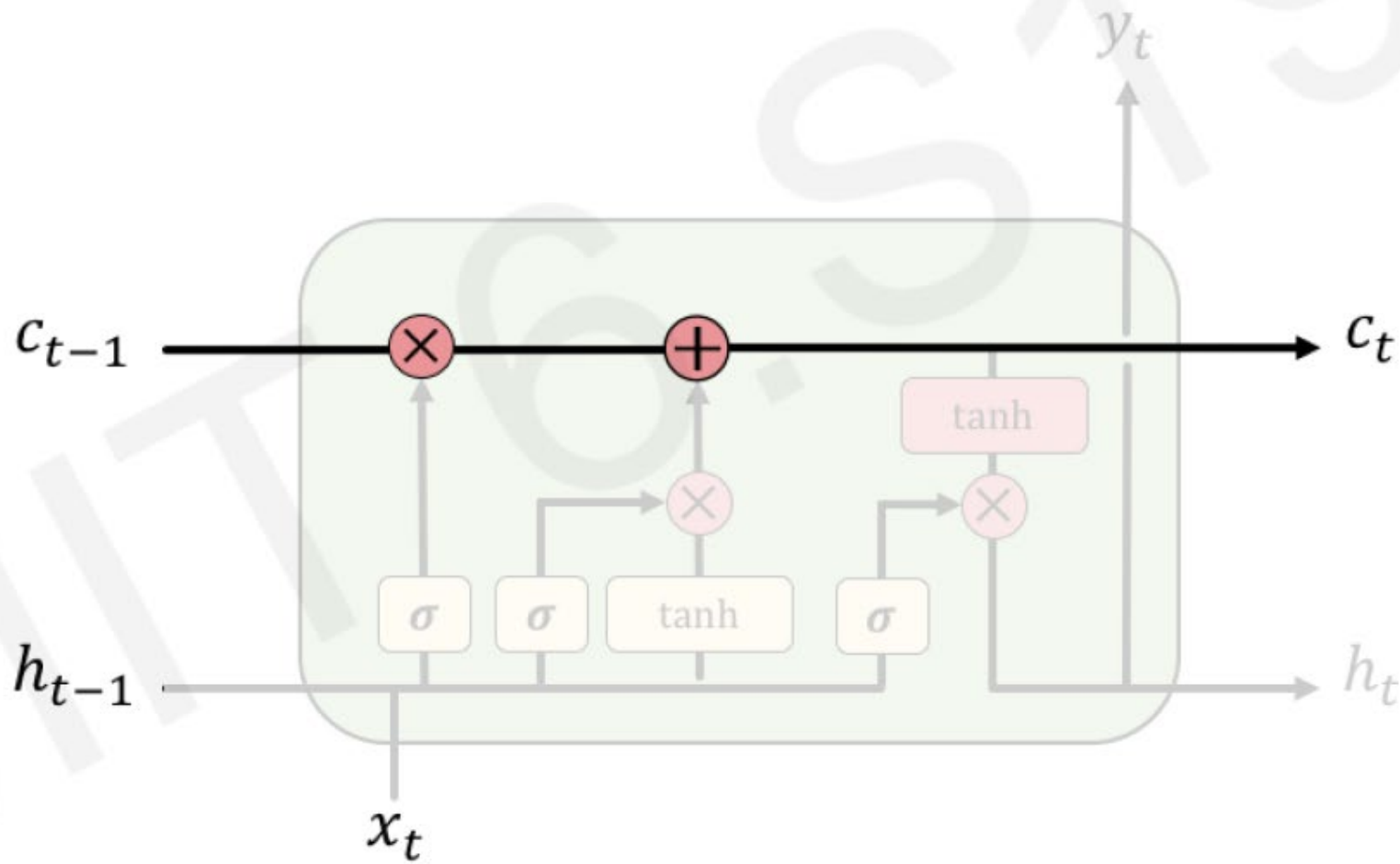
LSTMs **store relevant** new information into the cell state



Long Short Term Memory (LSTMs)

1) Forget 2) Store **3) Update** 4) Output

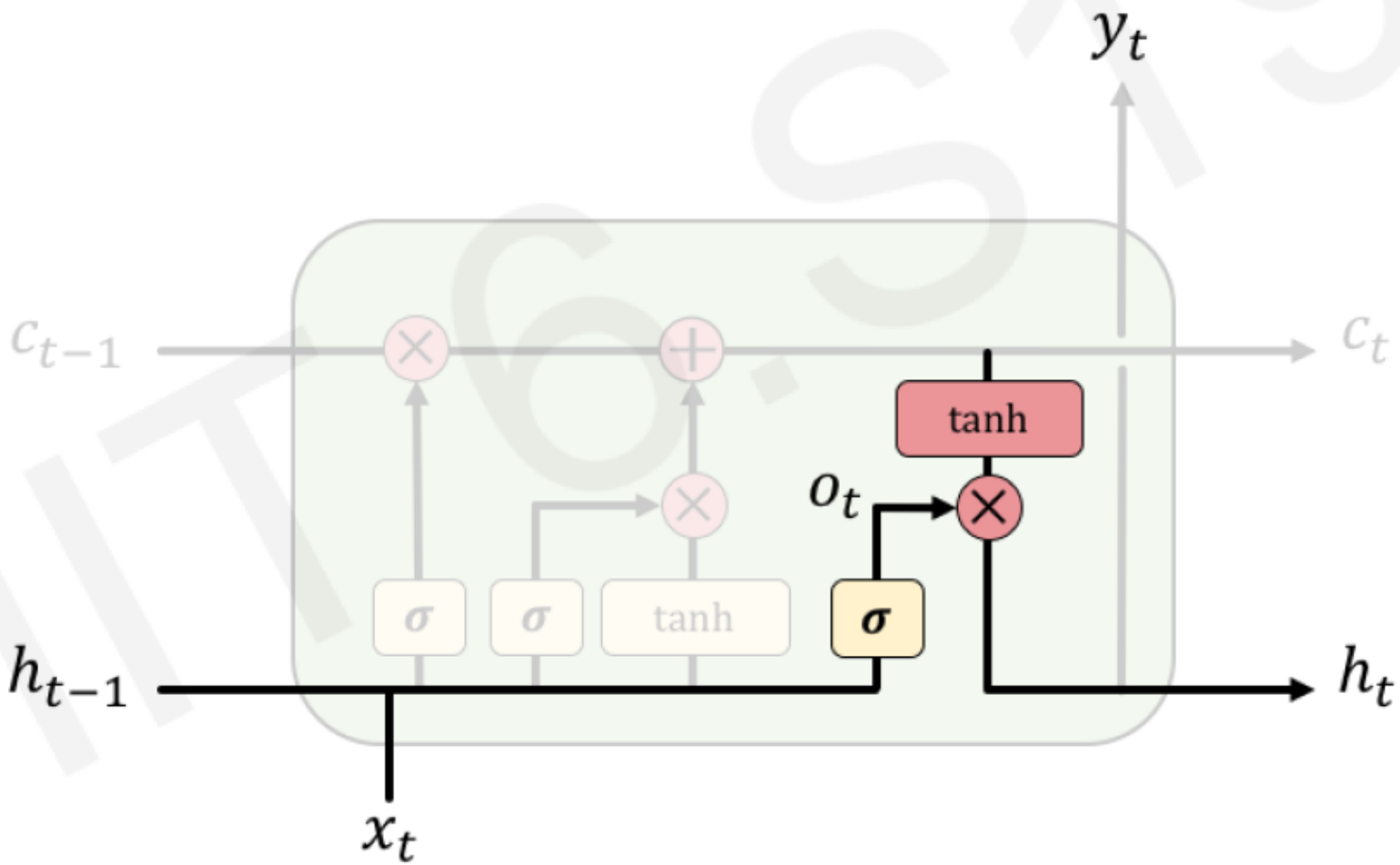
LSTMs **selectively update** cell state values



Long Short Term Memory (LSTMs)

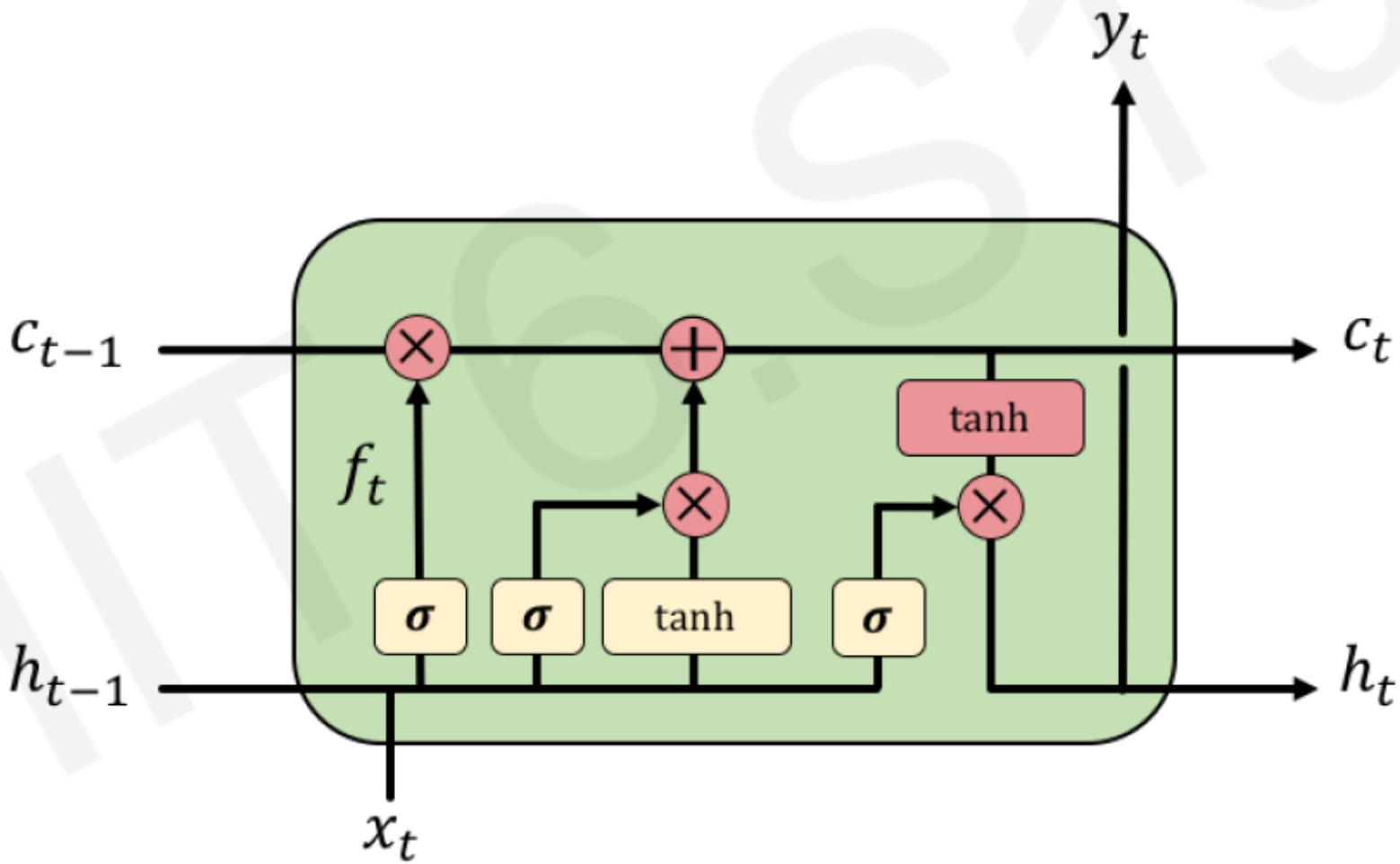
1) Forget 2) Store 3) Update **4) Output**

The **output gate** controls what information is sent to the next time step

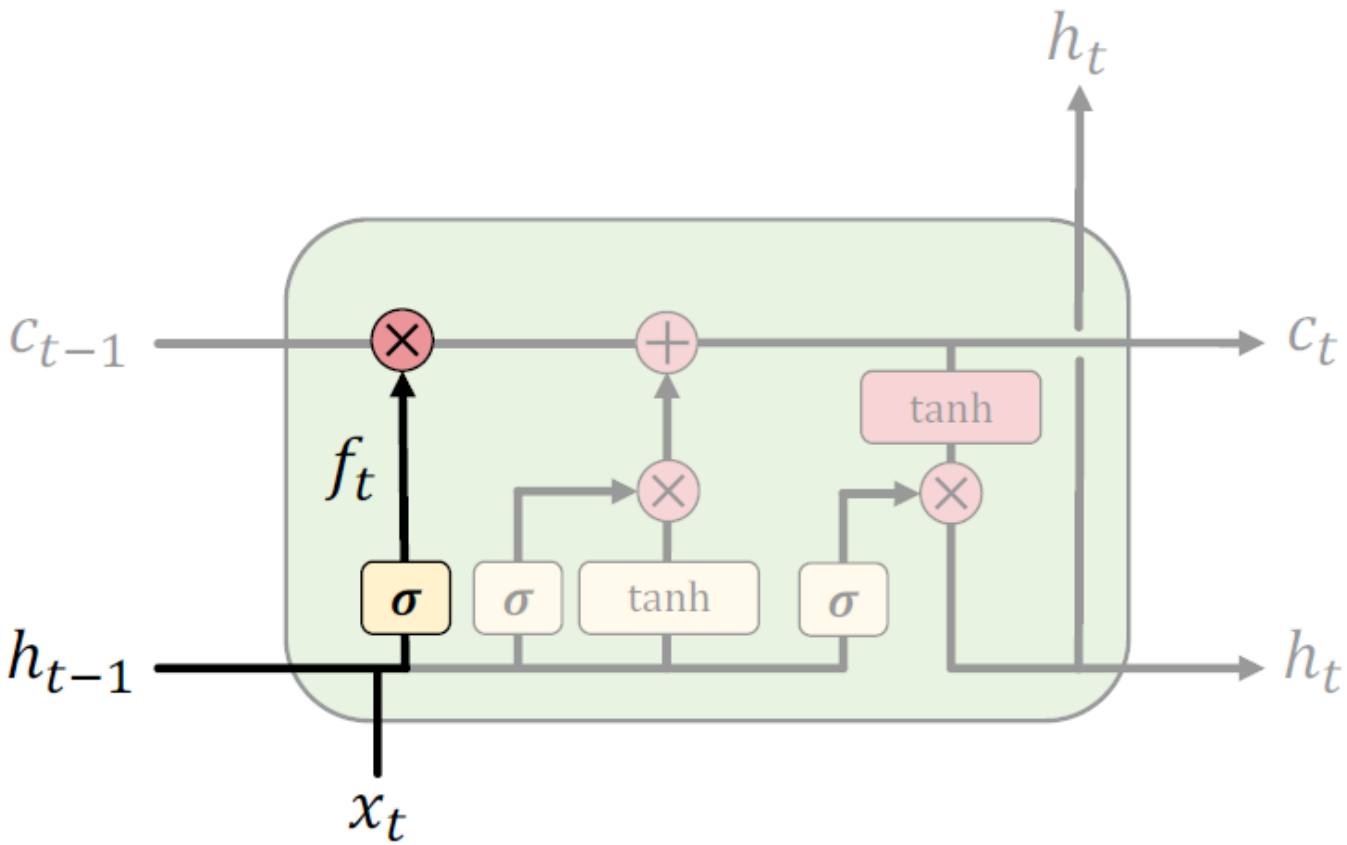


Long Short Term Memory (LSTMs)

1) Forget 2) Store 3) Update 4) Output



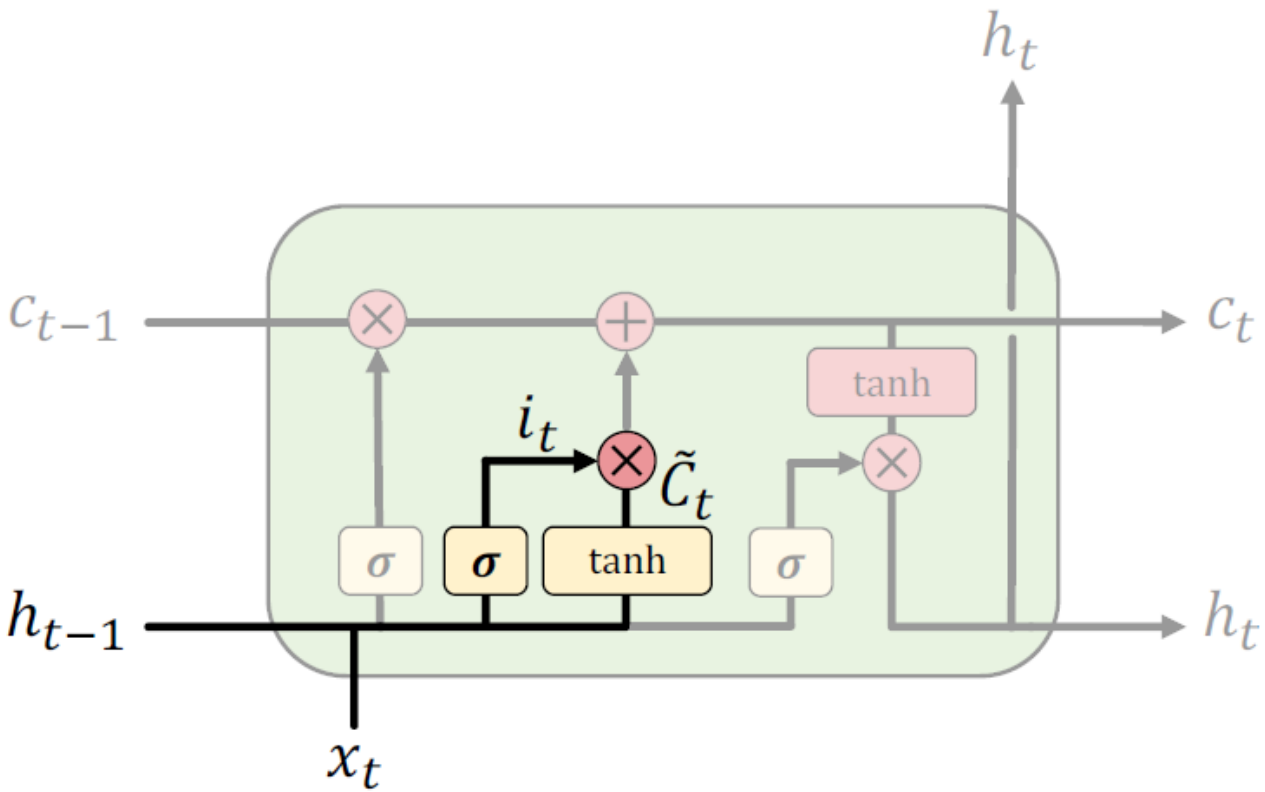
LSTMs: forget irrelevant information



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

- Use previous cell output and input
- Sigmoid: value 0 and 1 – "completely forget" vs. "completely keep"

LSTMs: identify new information to be stored

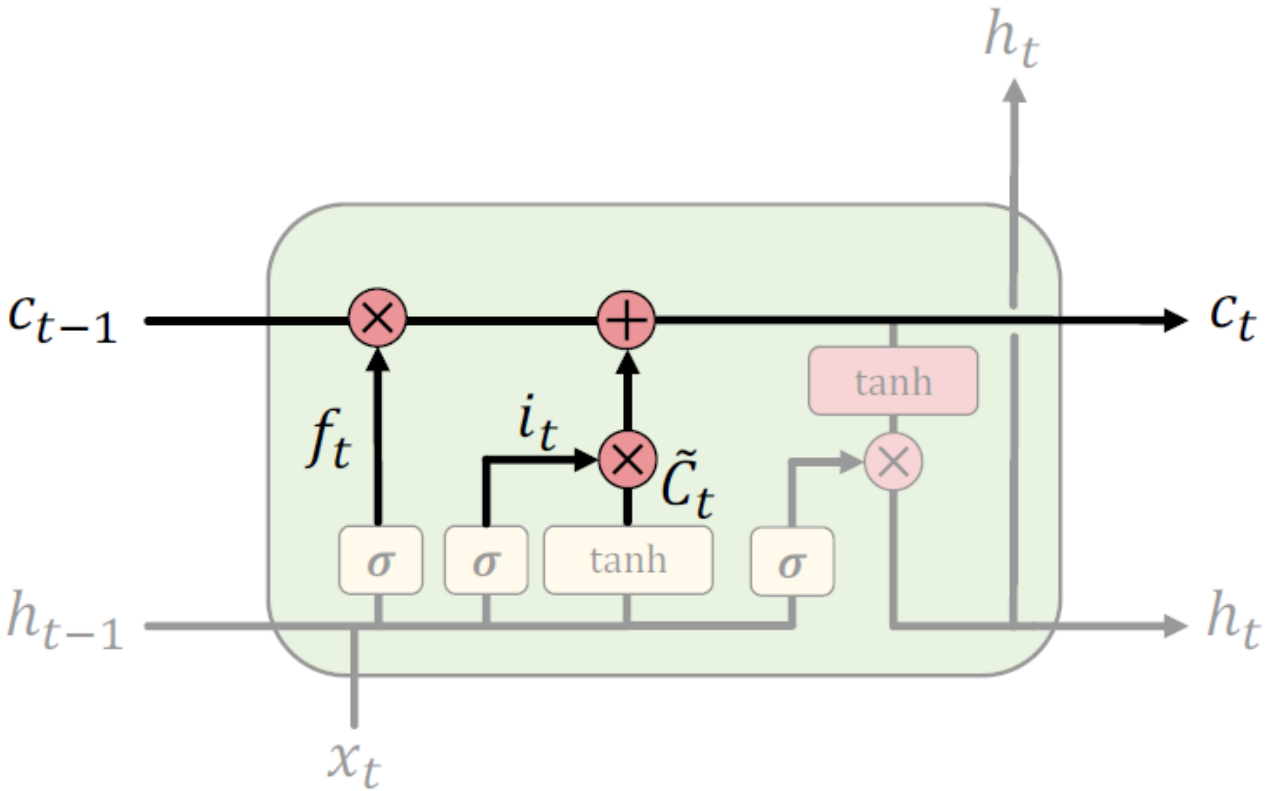


$$i_t = \sigma(\boldsymbol{W}_i[\, h_{t-1}, x_t \,] + b_i)$$

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

- Sigmoid layer: decide what values to update
- Tanh layer: generate new vector of "candidate values" that could be added to the state

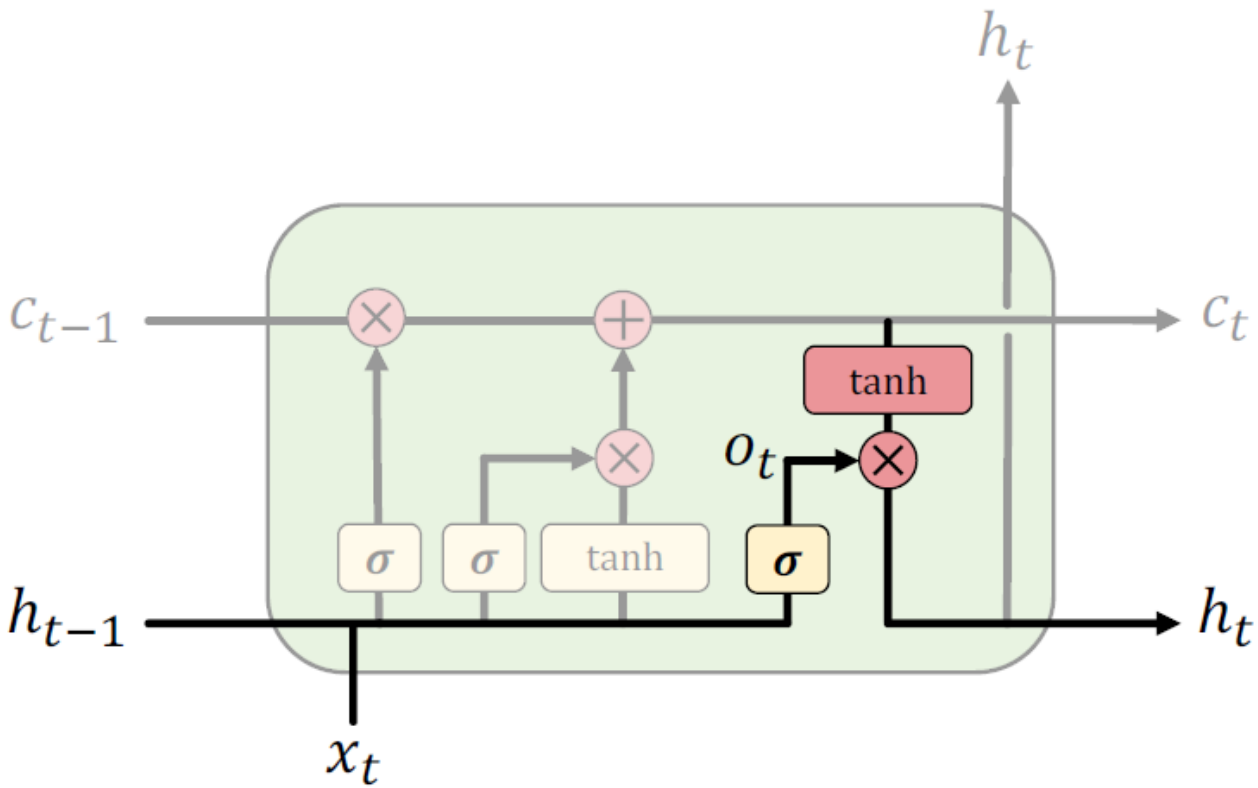
LSTMs: update cell state



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

- Apply forget operation to previous internal cell state: $f_t * C_{t-1}$
- Add new candidate values, scaled by how much we decided to update: $i_t * \tilde{C}_t$

LSTMs: output filtered version of cell state



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

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

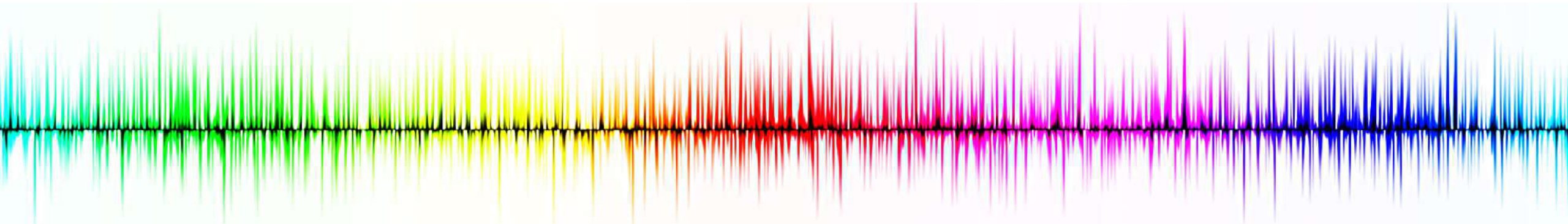
- Sigmoid layer: decide what parts of state to output
- Tanh layer: squash values between -1 and 1
- $o_t * \tanh(C_t)$: output filtered version of cell state

LSTMs: key concepts

1. Maintain a separate cell state from what is outputted
2. Use gates to control the flow of information
   - Forget gate gets rid of irrelevant information
   - Store relevant information from current input
   - Selectively update cell state
   - Output gate returns a filtered version of the cell state

Summary

1. RNNs are well suited for sequence modeling tasks
2. Model sequences via a recurrence relation
3. Training RNNs with backpropagation through time
4. Gated cells like LSTMs let us model long-term dependencies
5. Models for music generation, classification, machine translation



Resources

Stanford CS230: Deep Learning

MIT 6.S191: Introduction to Deep Learning

Illinois CS 498: Introduction to Deep Learning