EE-508: Hardware Foundations for Machine Learning

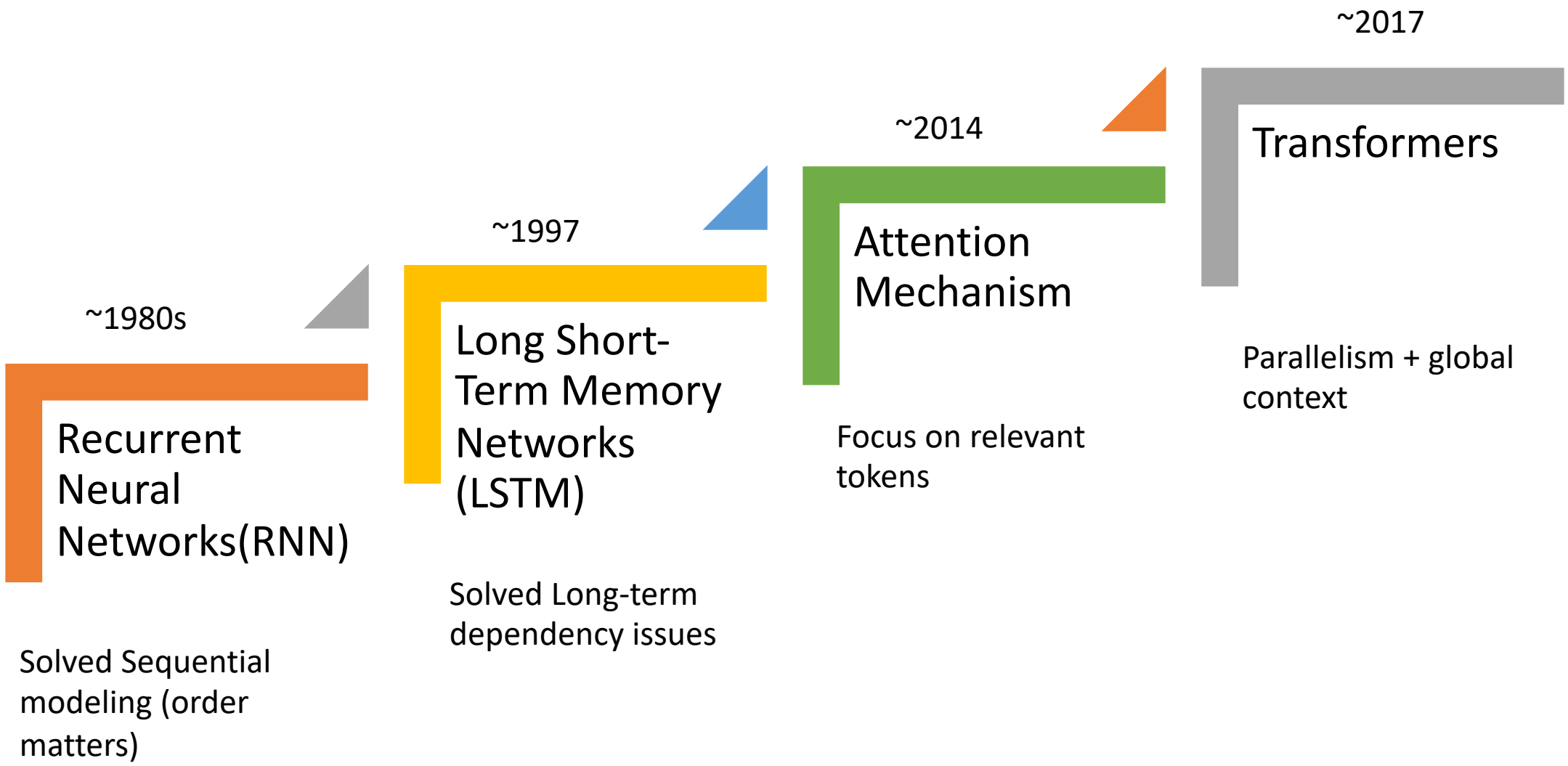
RNNs

University of Southern California

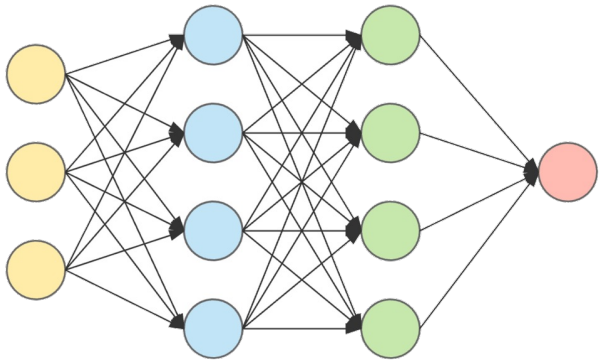
Ming Hsieh Department of Electrical and Computer Engineering

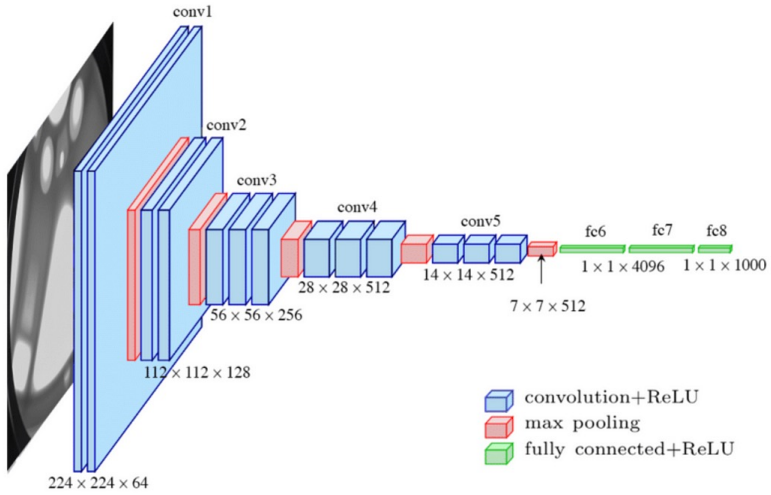
Instructors:
Arash Saifhashemi

Evolution Toward Transformers

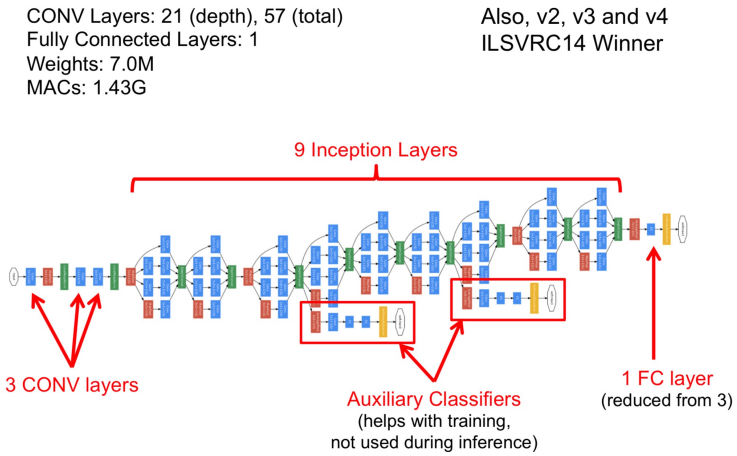


What we have seen so far...





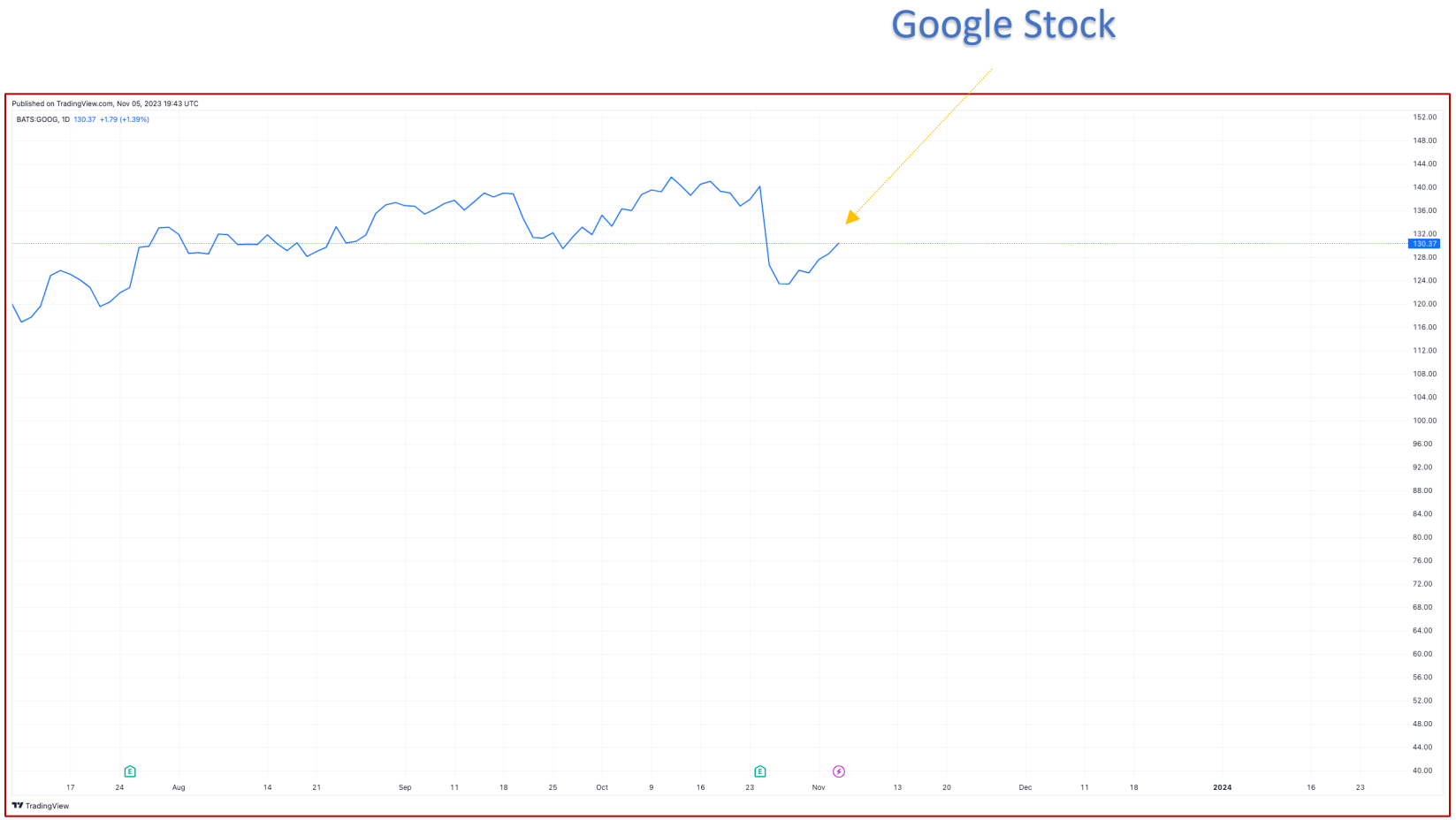
VGG-16



GoogLeNet

Constant number of inputs

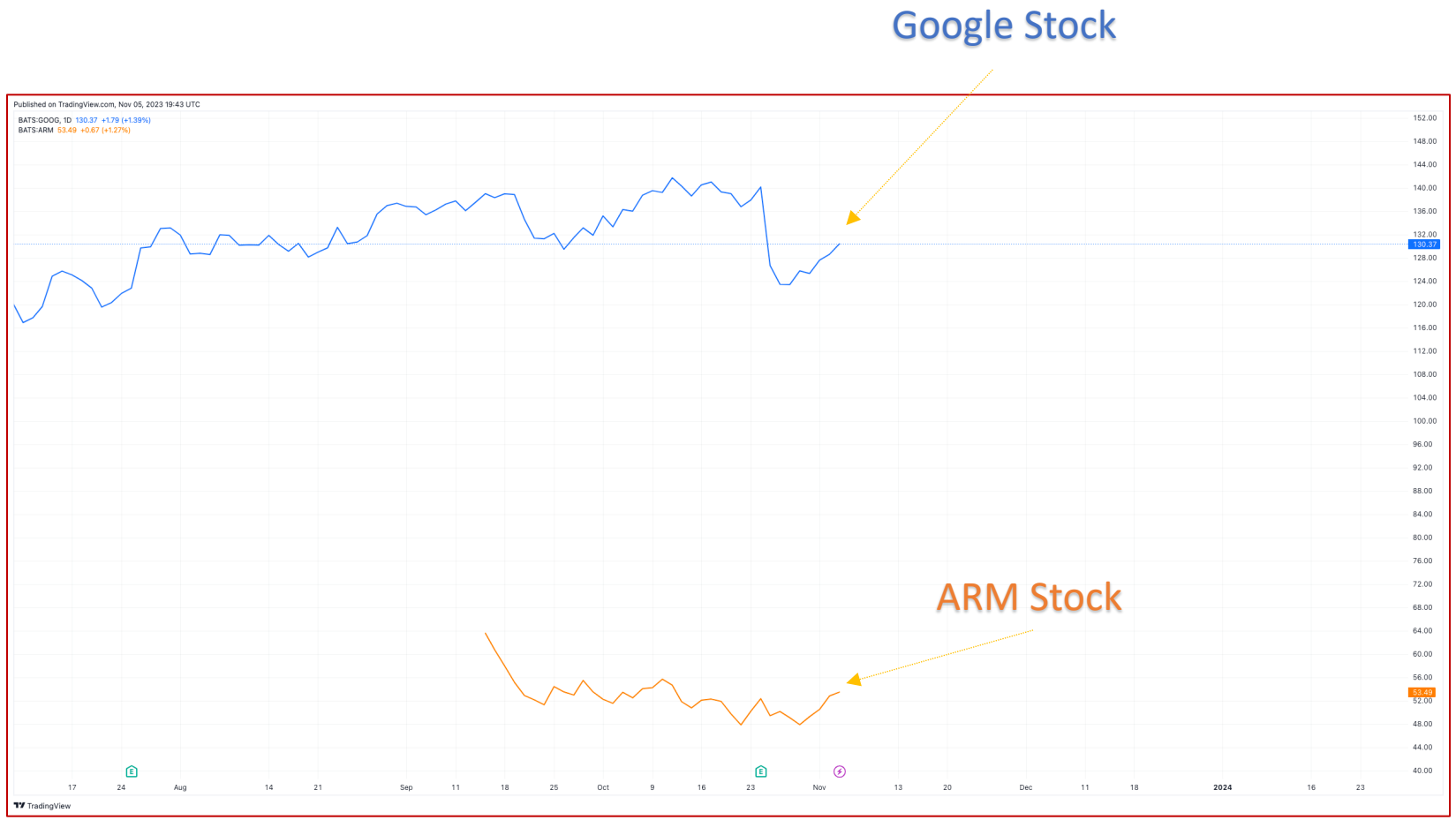
Processing Sequences



Predicting the stock value based on the history

Source: https://www.tradingview.com/x/ufByhJCh/

Processing Sequences

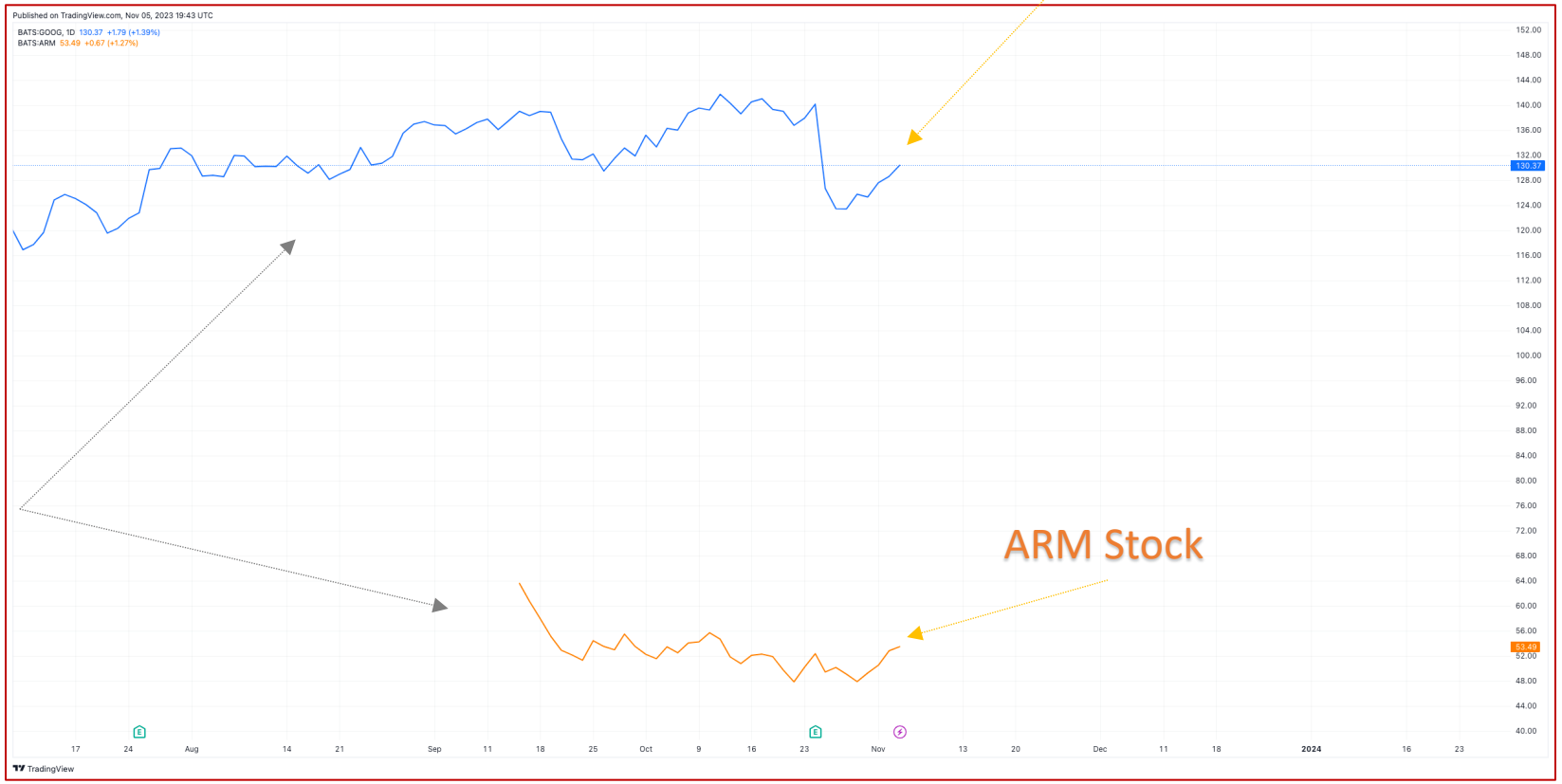


Predicting the stock value based on the history

Source: https://www.tradingview.com/x/ufByhJCh/

Processing Sequences

Different amount of input values



Predicting the stock value based on the history

Source: https://www.tradingview.com/x/ufByhJCh/

Processing Sequences in Movie Review

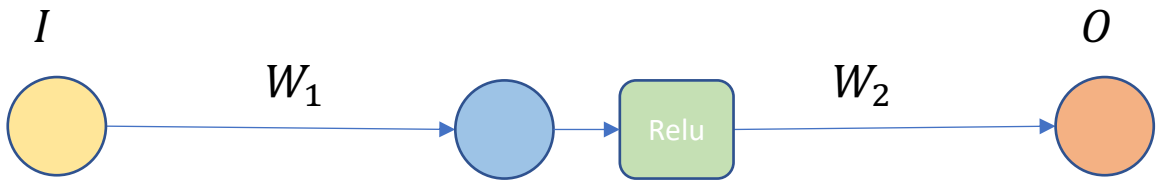
In NLP, we have variable number of inputs

Source: CBS via Getty Images

Review 1: *"Great Movie!"*

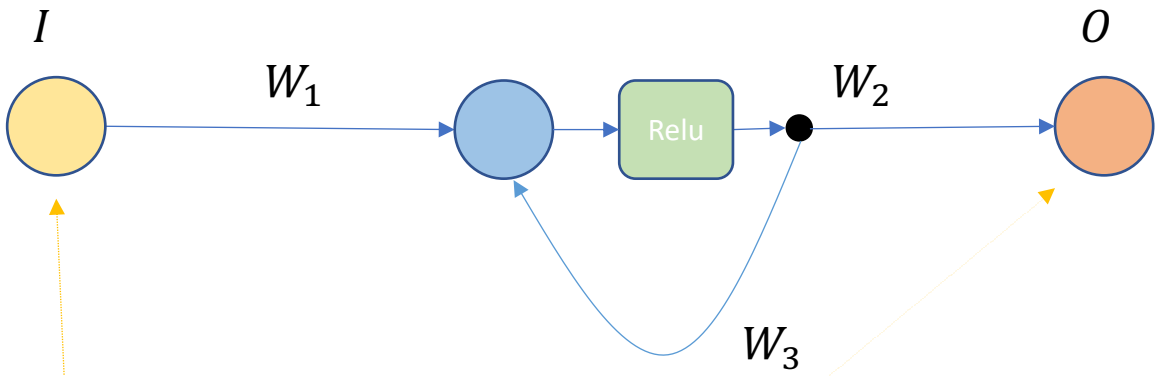
Review 2: ***"****Terminator 2: Judgment Day' elevates the sci-fi action genre to new heights. James Cameron's visionary sequel blends groundbreaking special effects with a profound narrative. Arnold Schwarzenegger reprises his role as the Terminator, this time as a protector. His transition from villain to hero is both convincing and compelling. Linda Hamilton gives a gritty performance as Sarah Connor, transforming into a formidable warrior. The film's commentary on technology and fate is thought-provoking. Its action sequences are iconic, setting a high bar for future films. 'T2' is not just a movie; it's a cultural milestone that remains influential and relevant."*

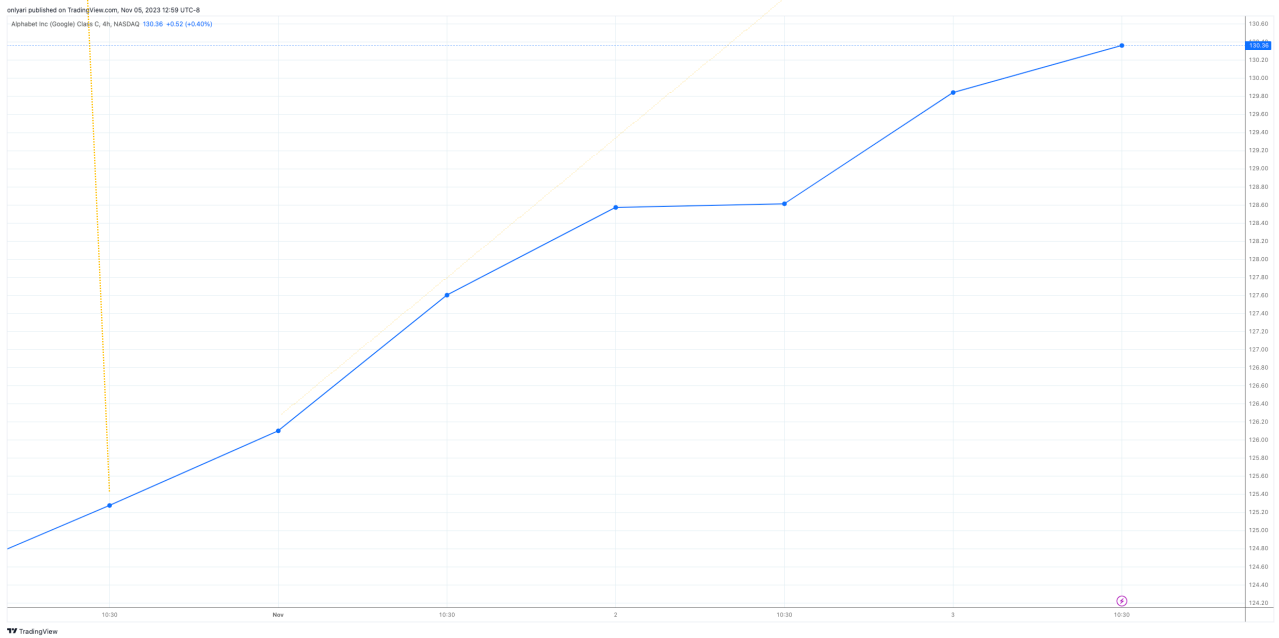
Recurrent Neural Network



Recurrent Neural Network

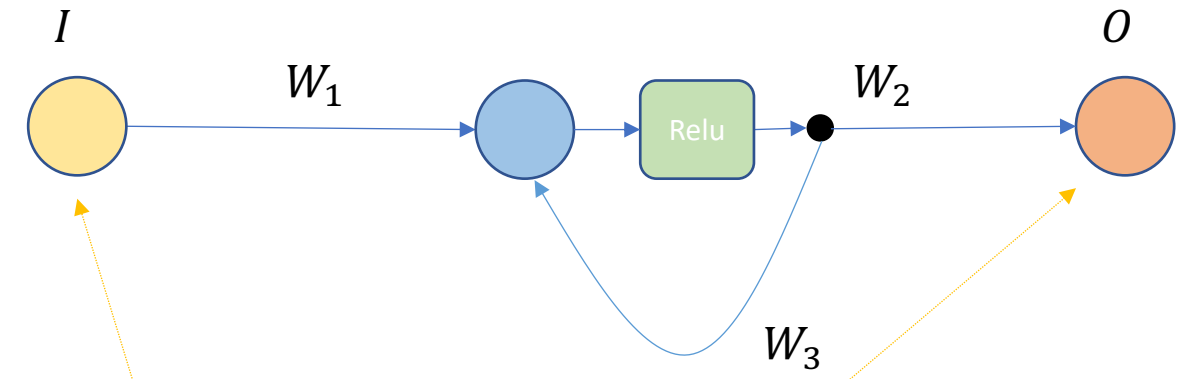
There is a feedback loop

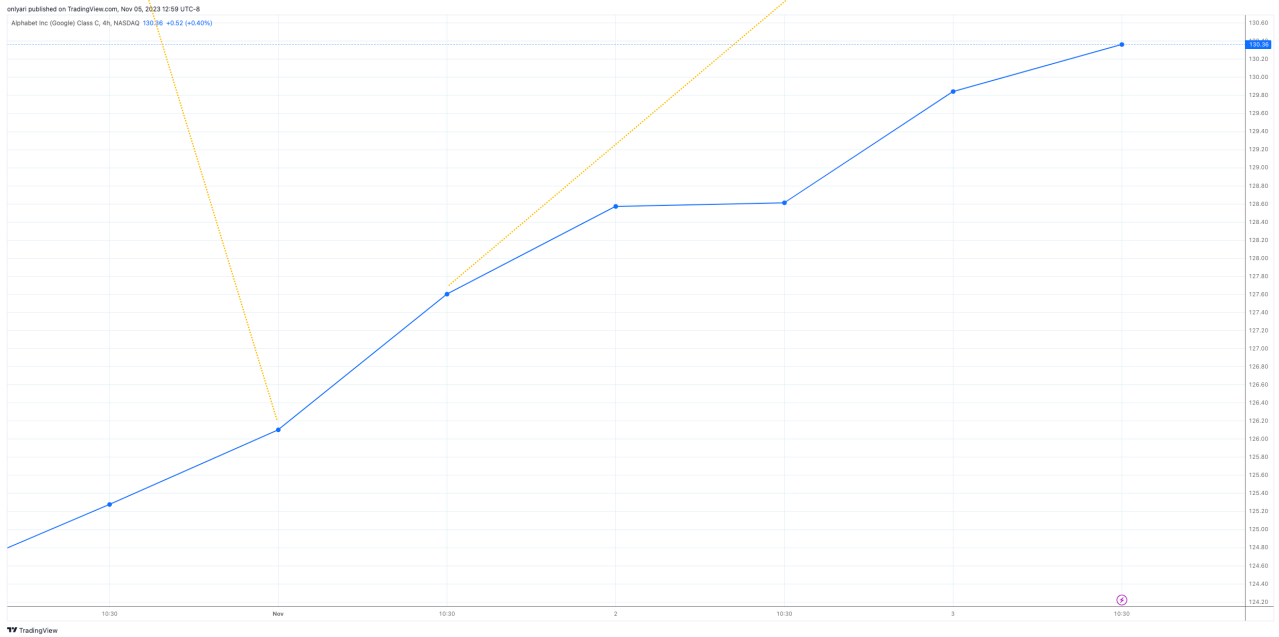




Recurrent Neural Network

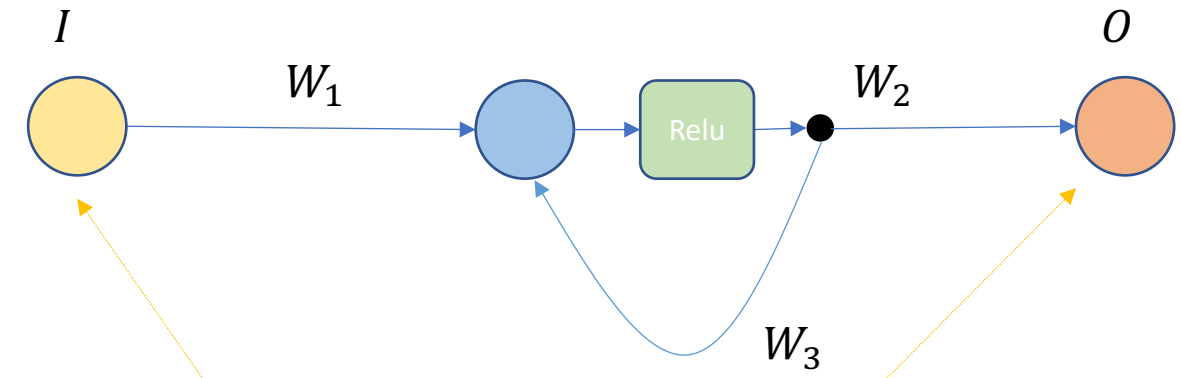
There is a feedback loop

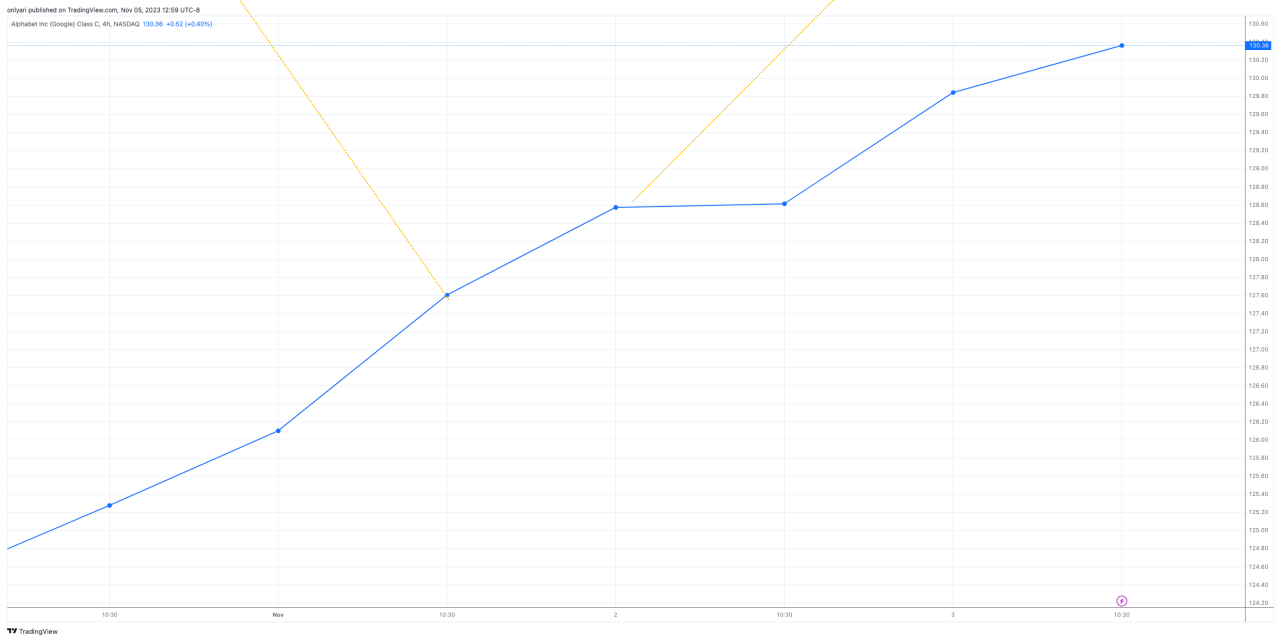




Recurrent Neural Network

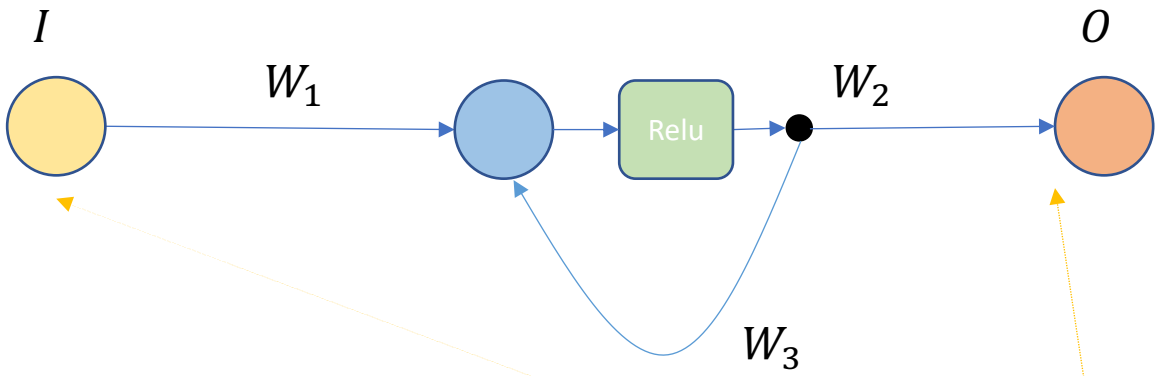
There is a feedback loop

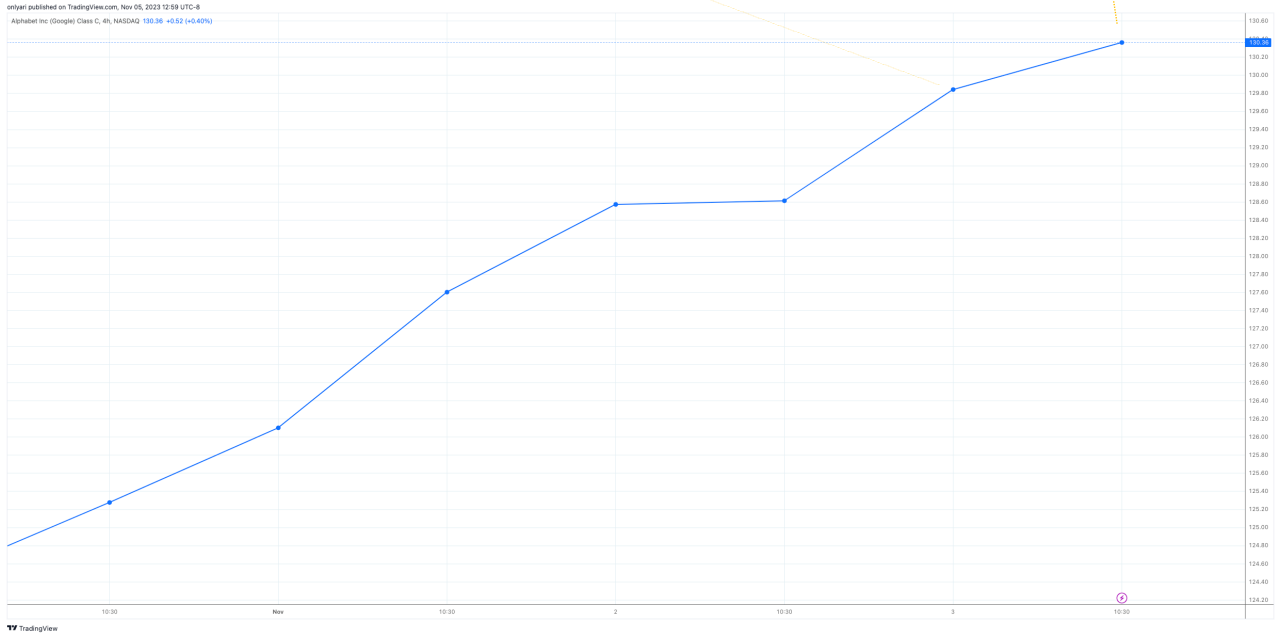




Recurrent Neural Network

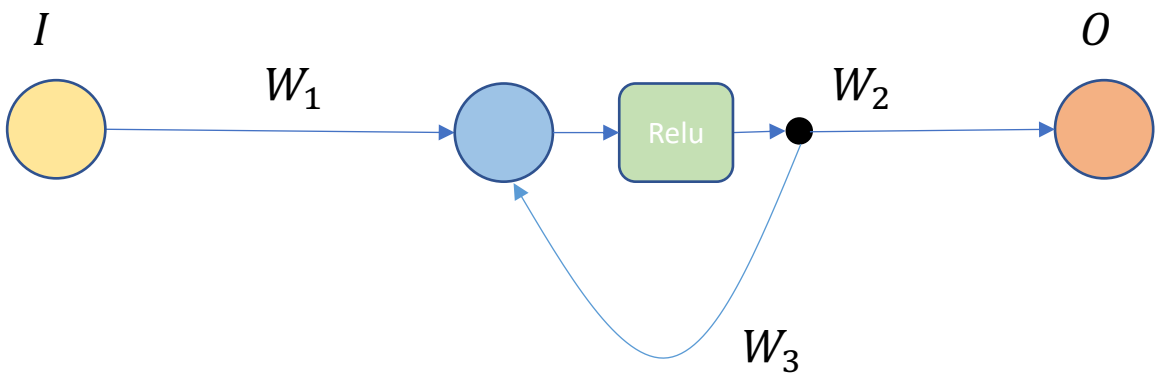
There is a feedback loop





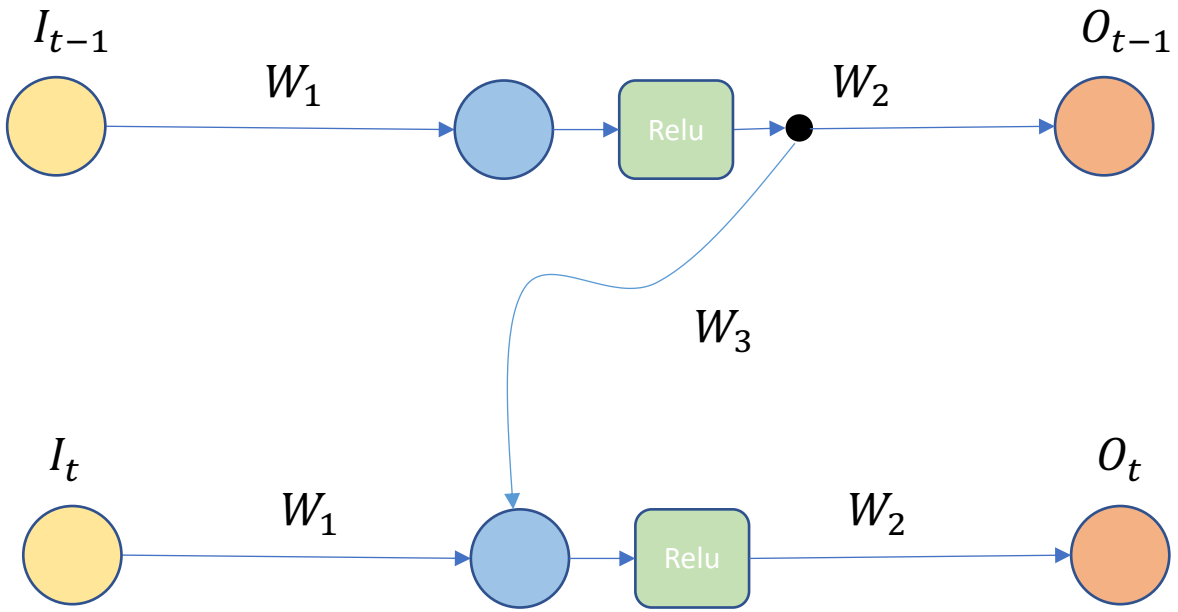
Recurrent Neural Network

Feedback loop can be unrolled

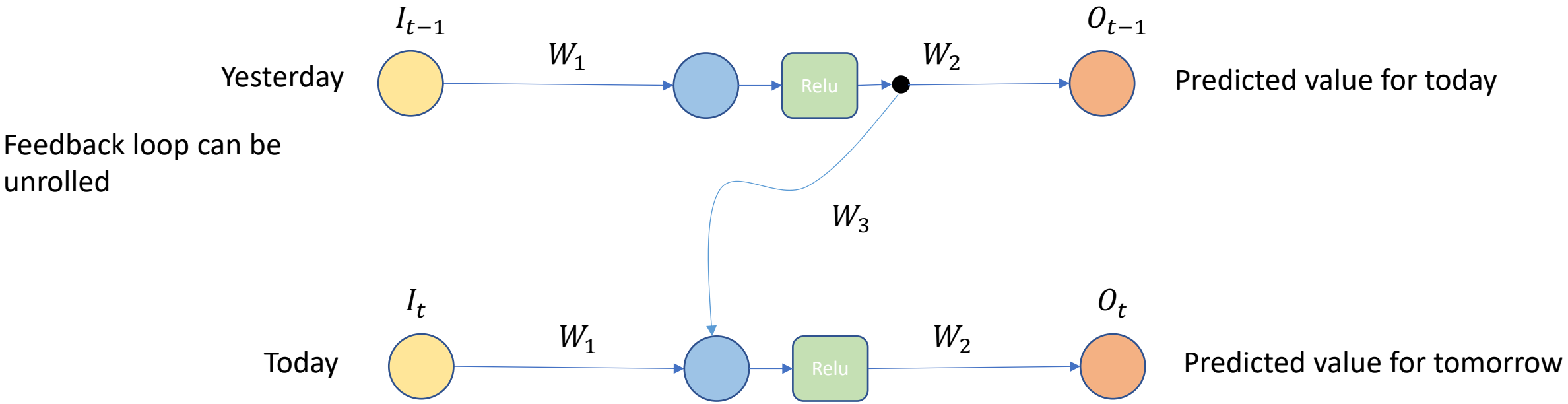


Recurrent Neural Network

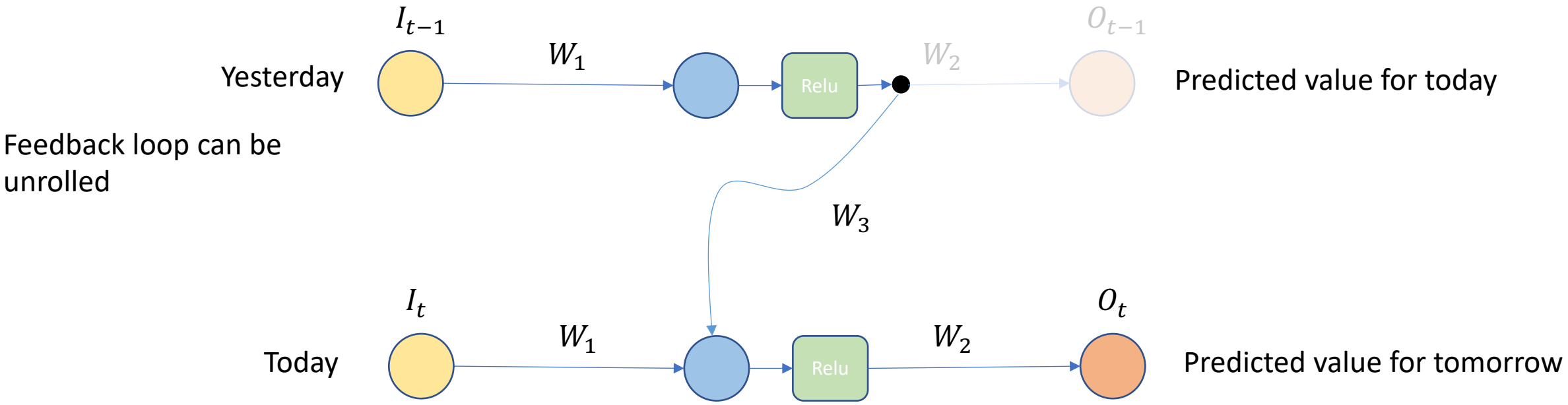
Feedback loop can be unrolled



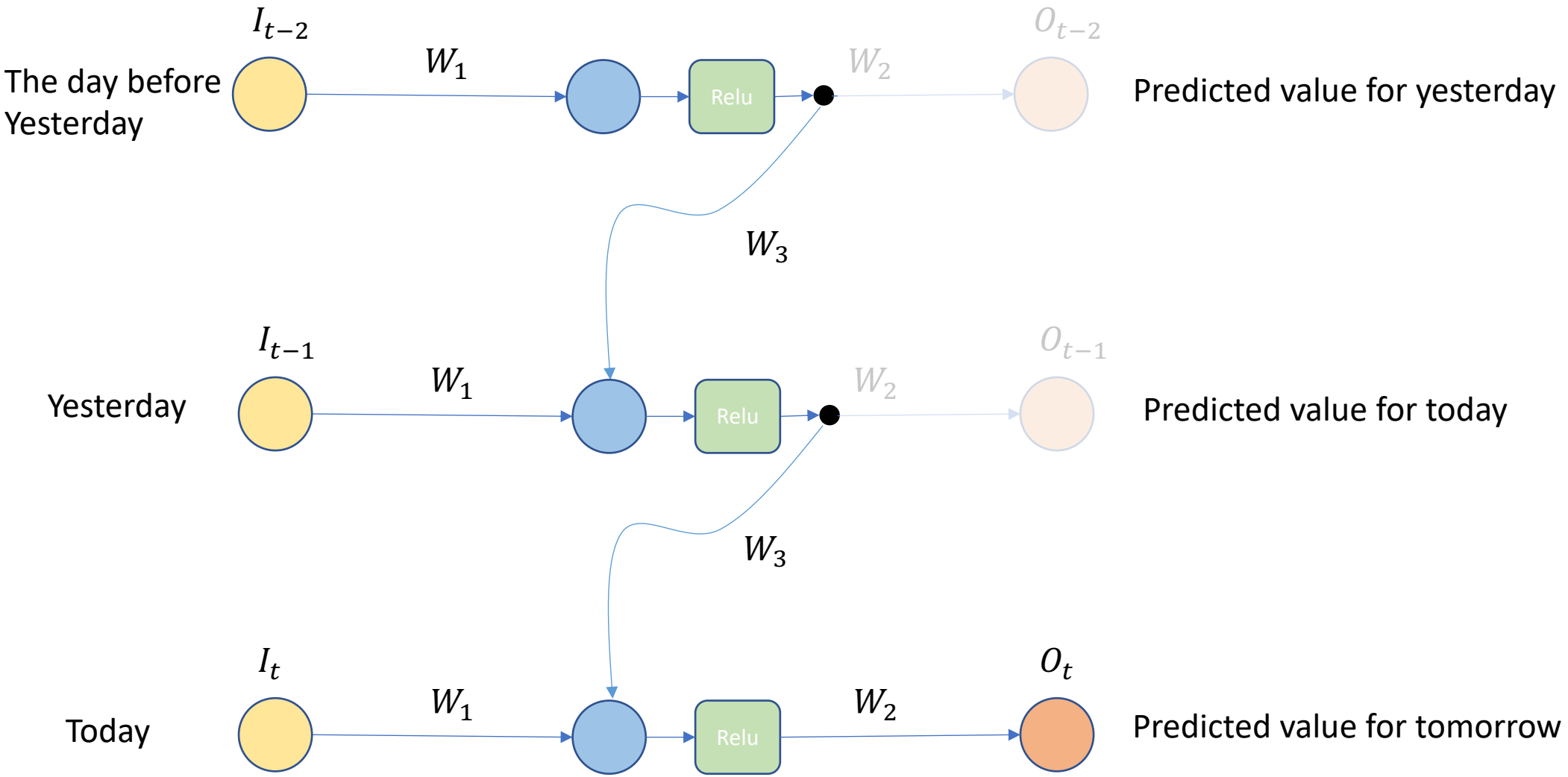
Recurrent Neural Network



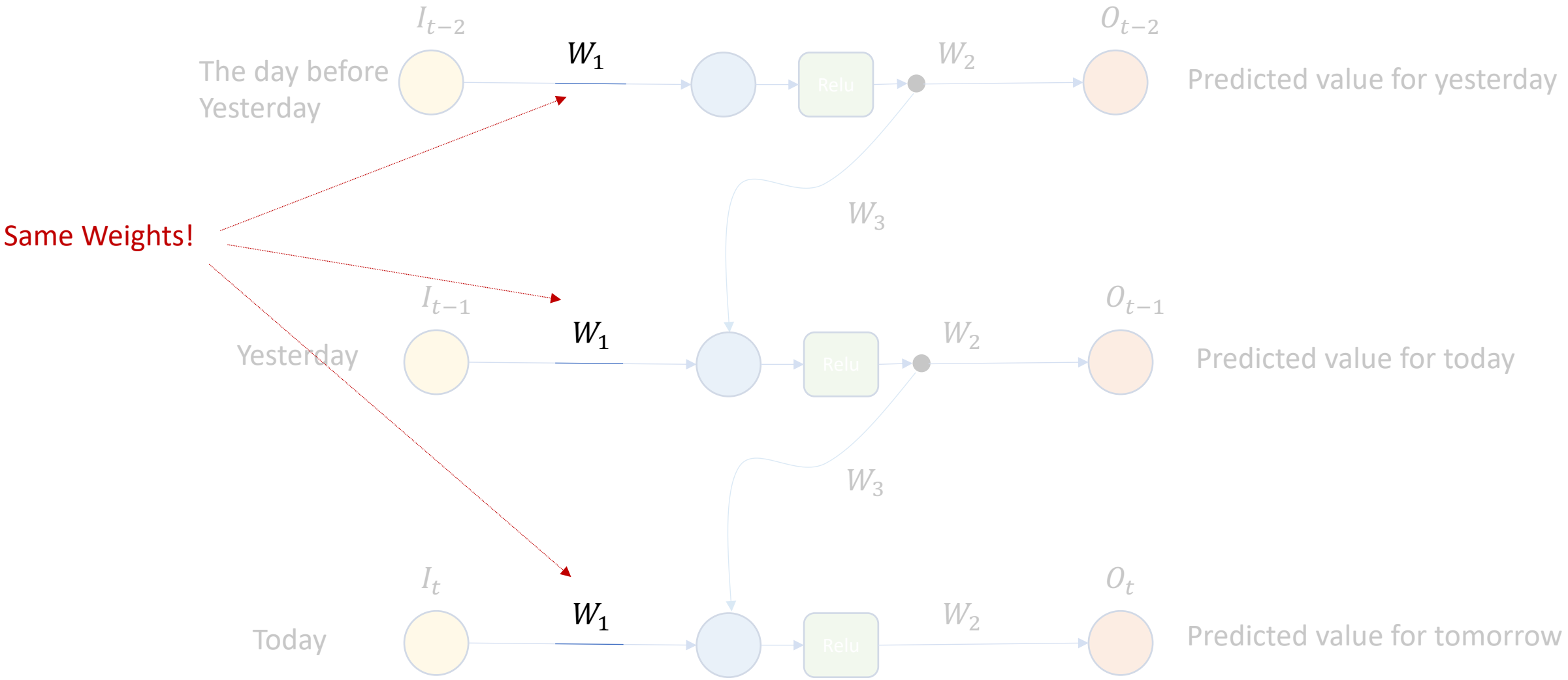
Recurrent Neural Network



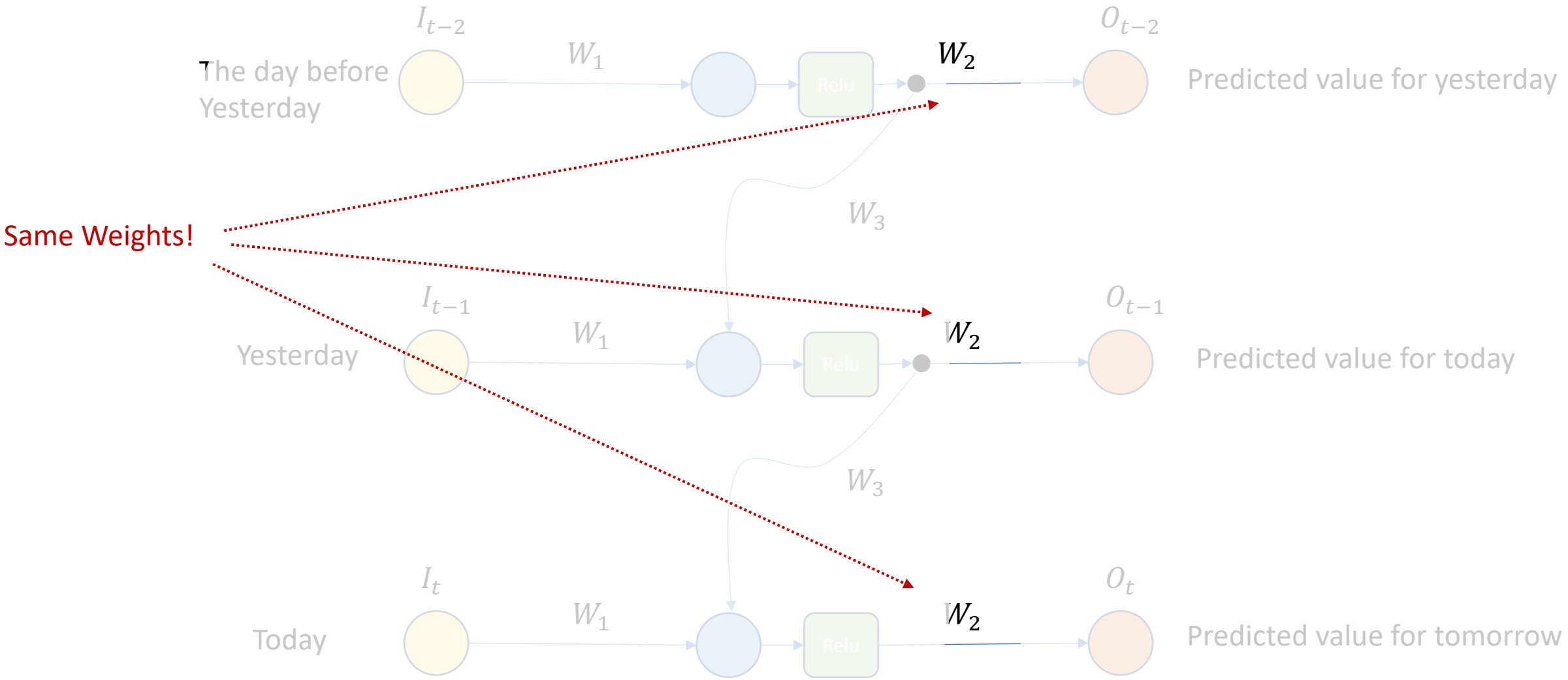
Recurrent Neural Network



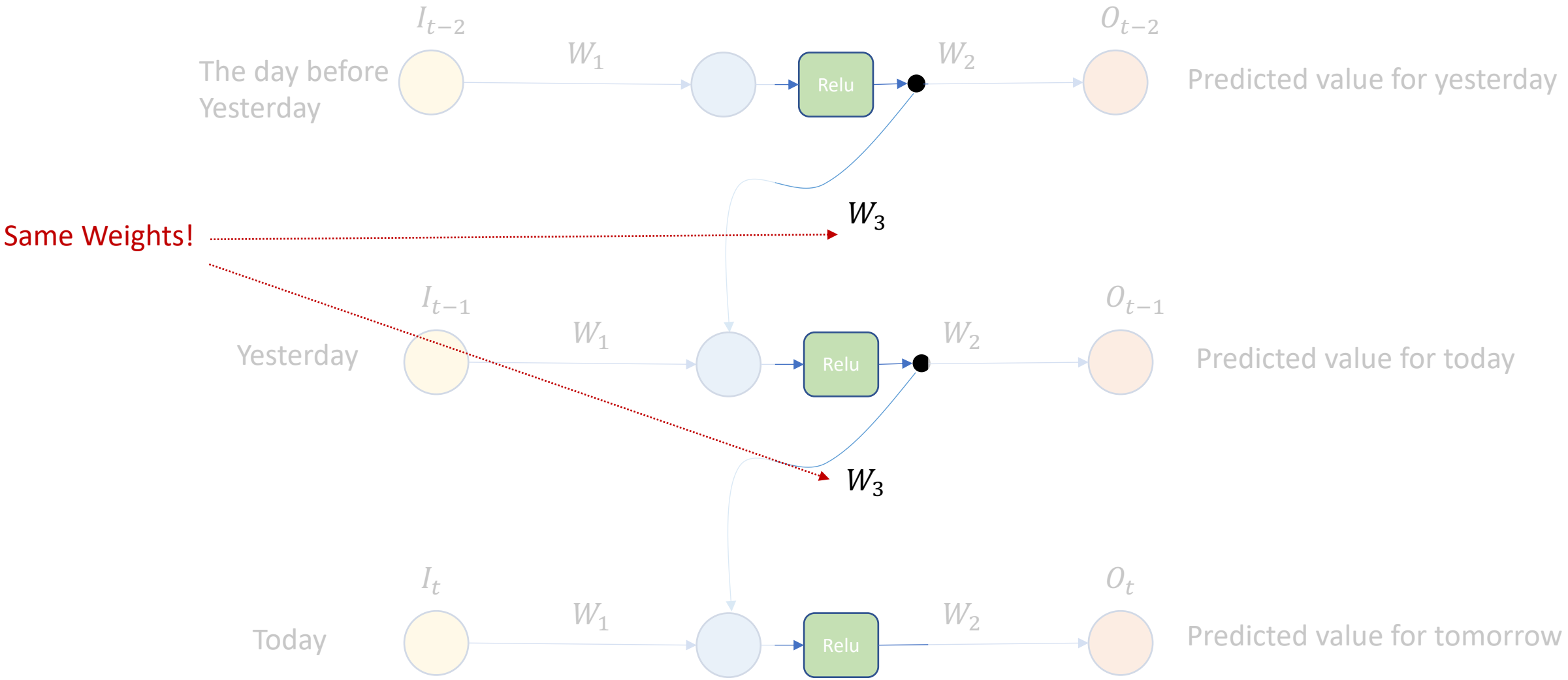
Recurrent Neural Network



Recurrent Neural Network



Recurrent Neural Network

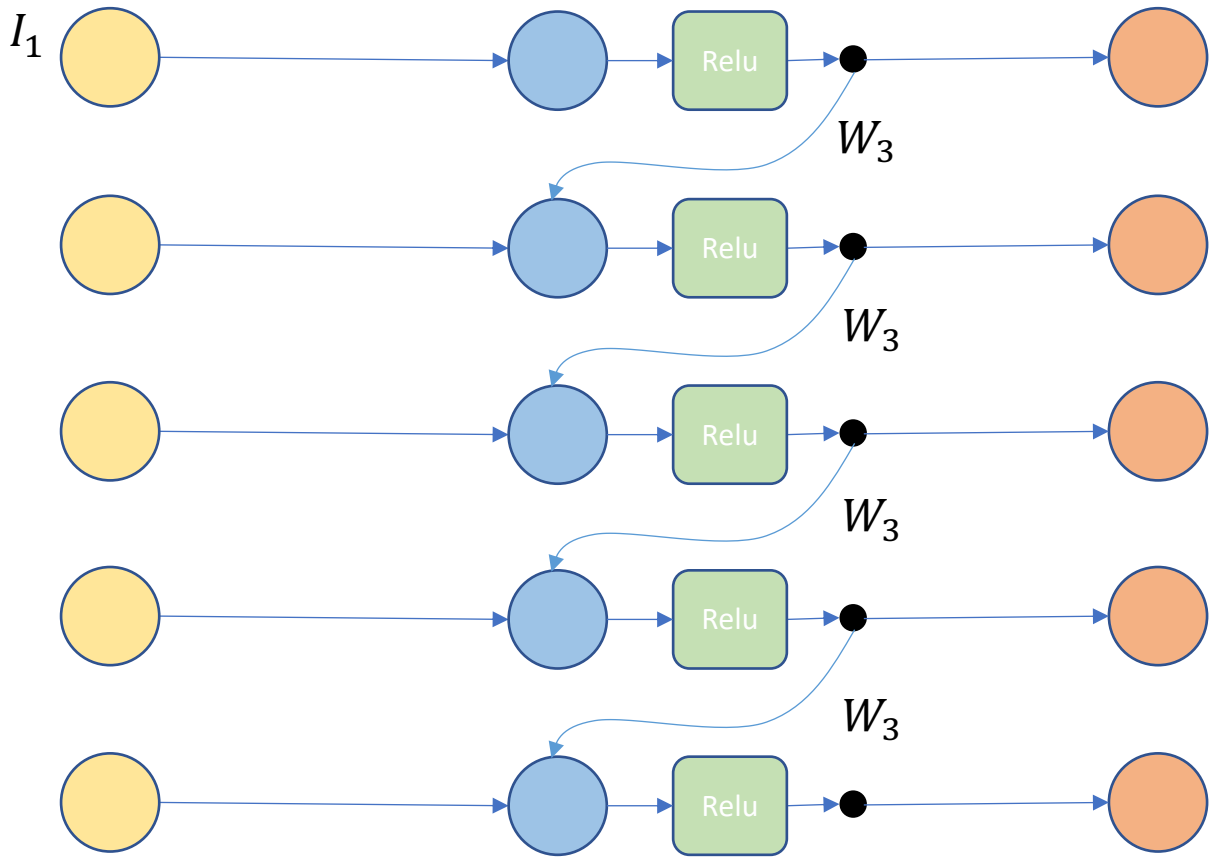


No matter how many times we unroll, we will use the same weights and biases

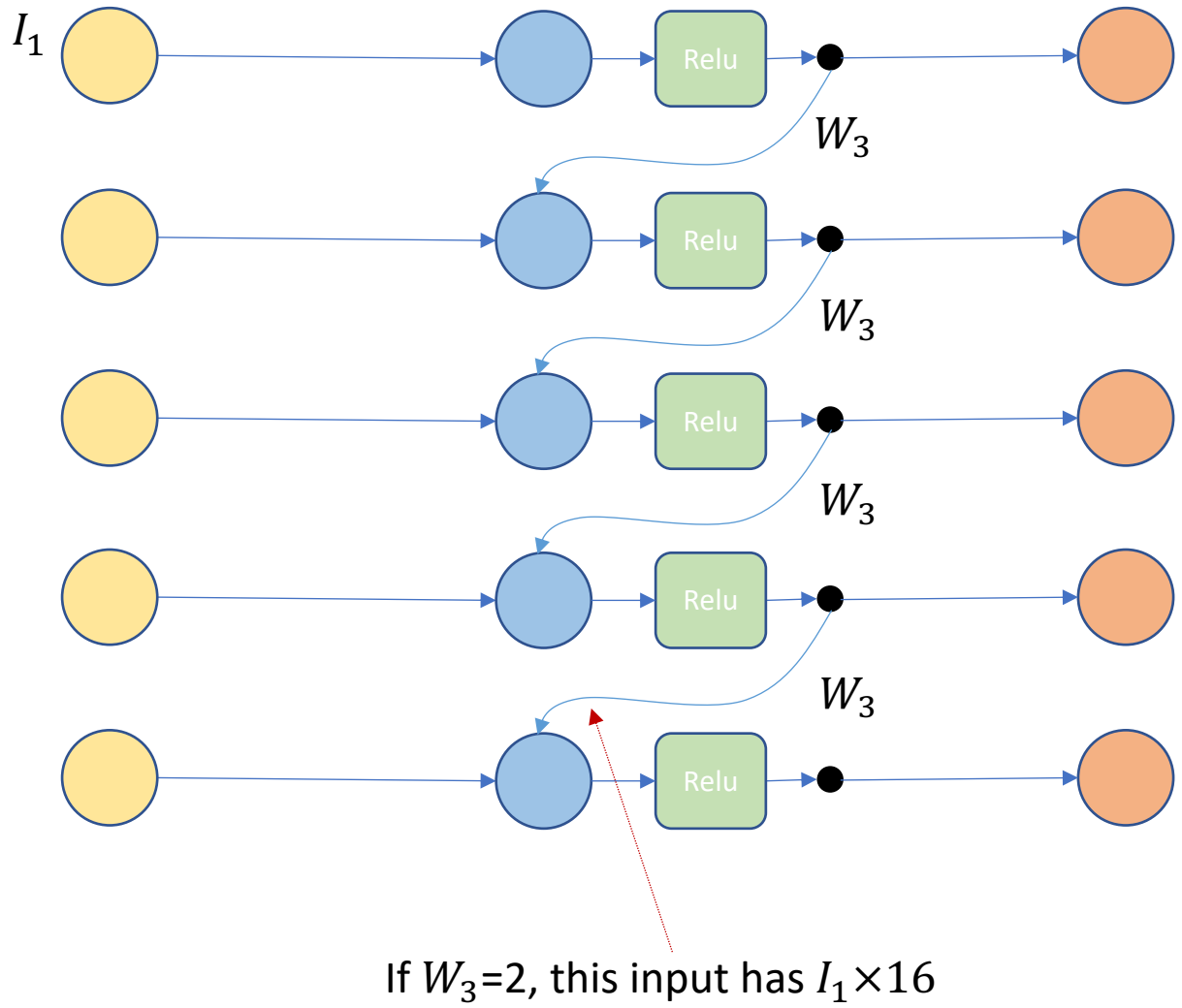
No matter how many times we unroll, we will use the same weights and biases

That was cool, but what are the challenges?

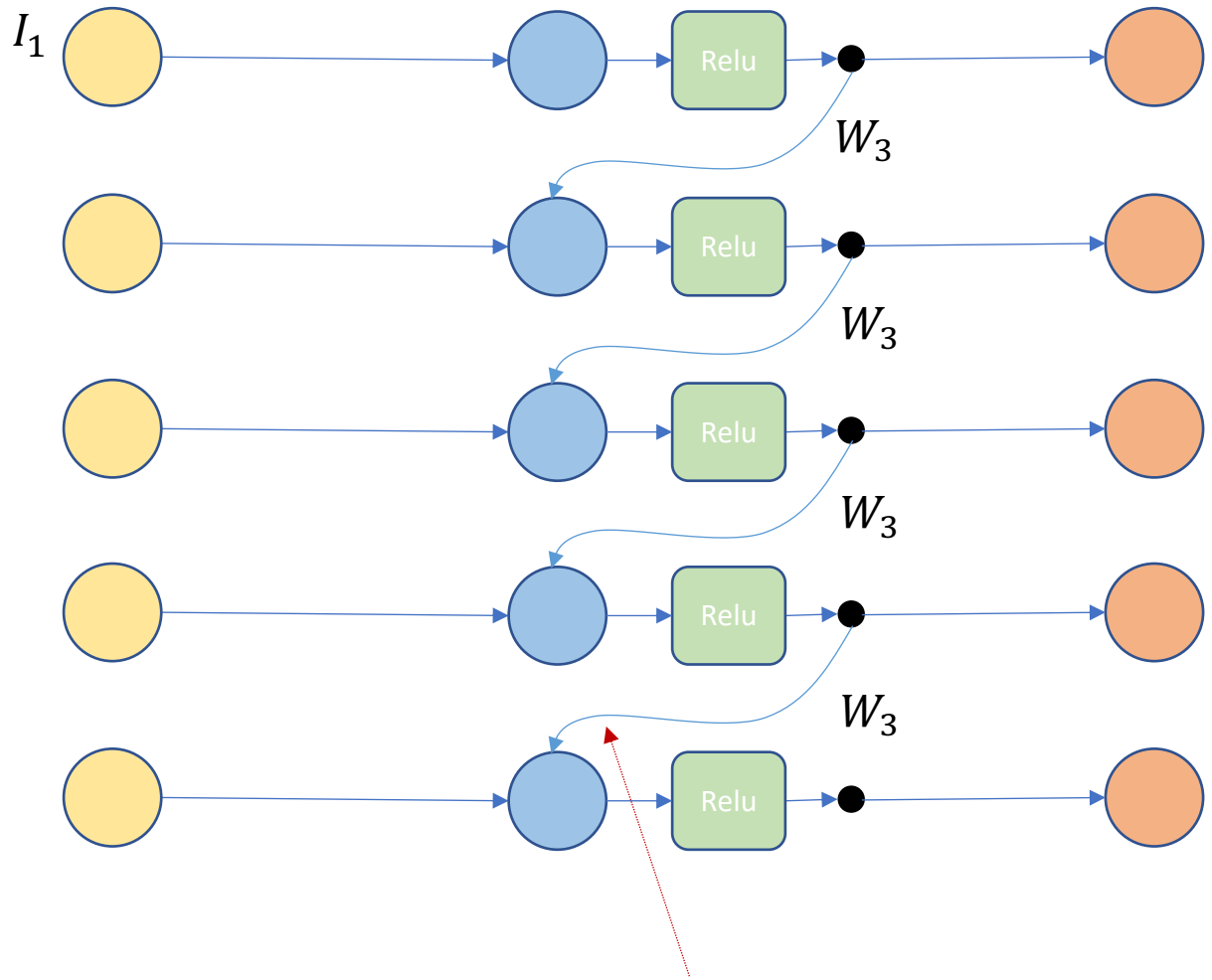
Vanishing/Exploding Gradient Problem



Vanishing/Exploding Gradient Problem



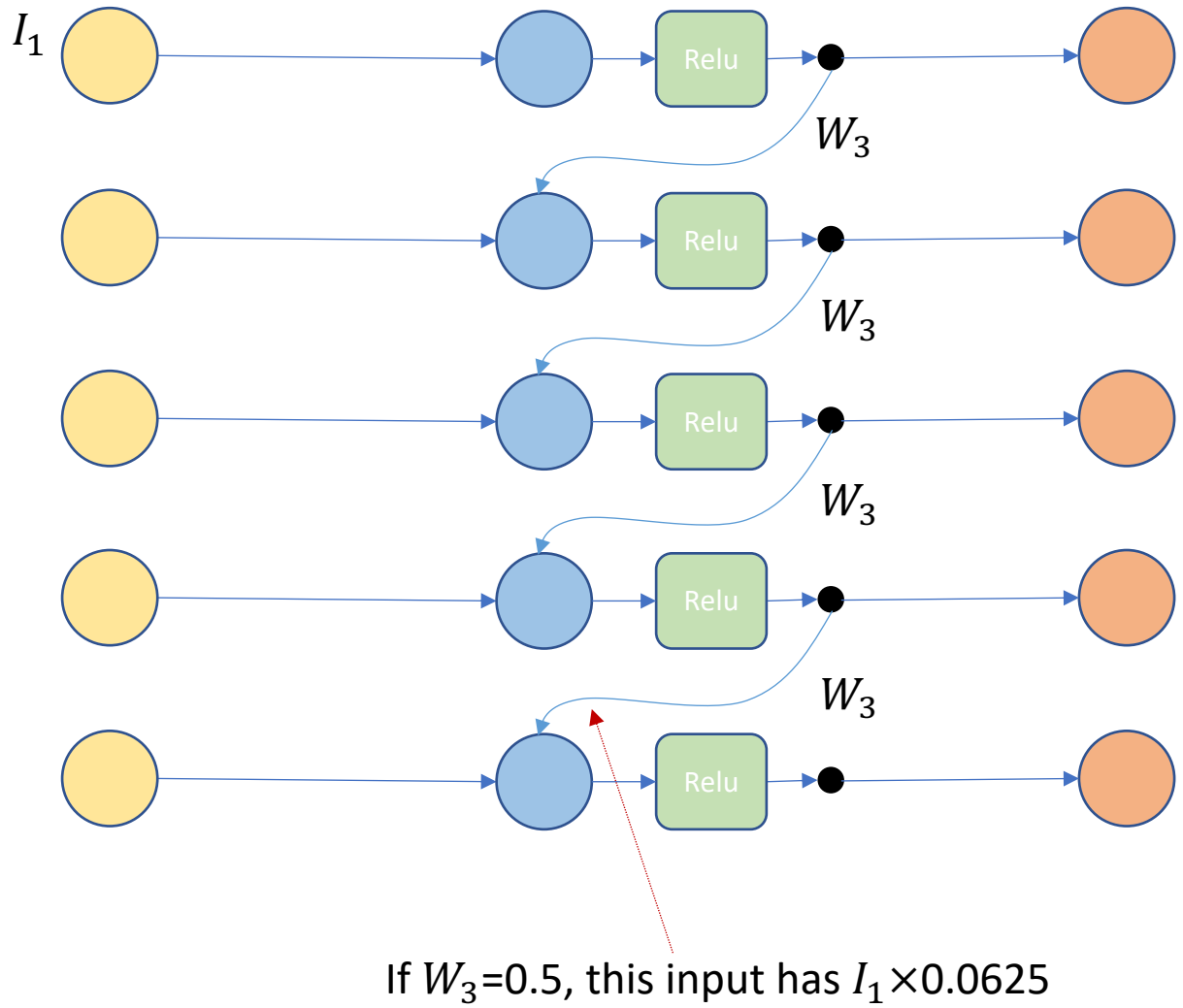
Vanishing/Exploding Gradient Problem



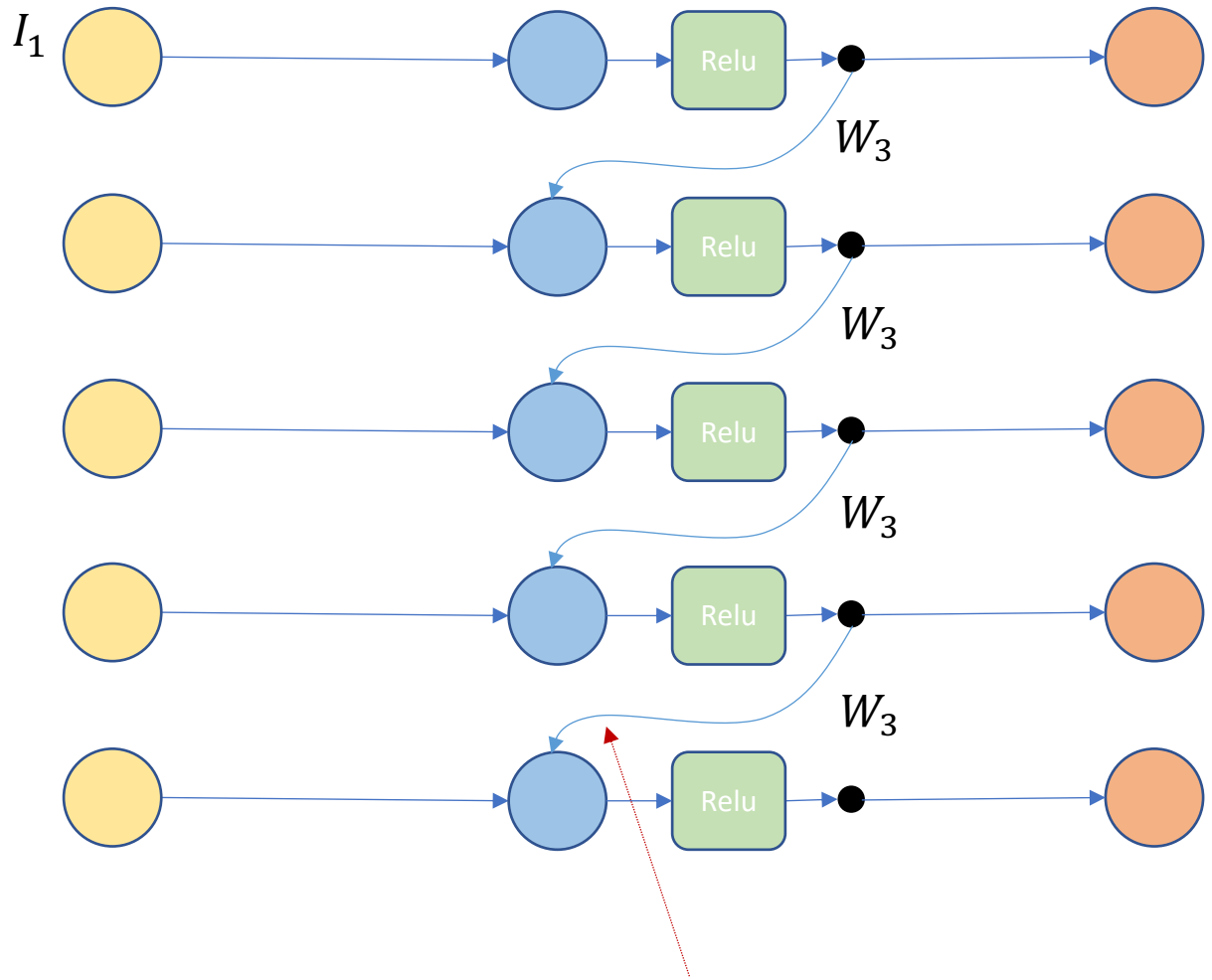
If W_3=2, this input has $I_1 \times 16$

If we use 100 days of data, I_1 will be amplified 2^{100} times!

Vanishing/Exploding Gradient Problem



Vanishing/Exploding Gradient Problem



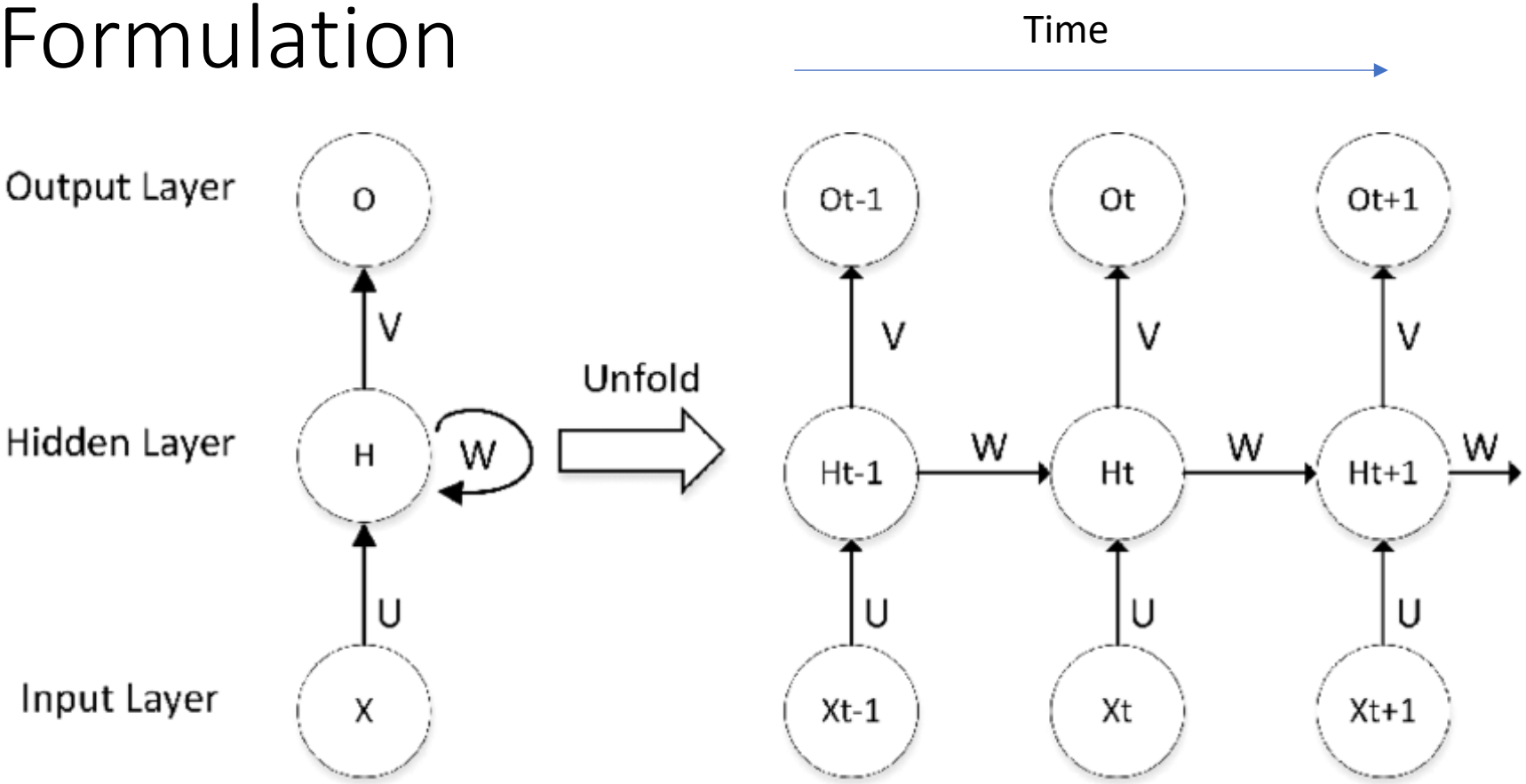
If W_3=0.5, this input has I_1×0.0625

If we use 100 days of data, I_1 will be amplified 2^{-100} times!

Vanishing/Exploding Gradient Problem

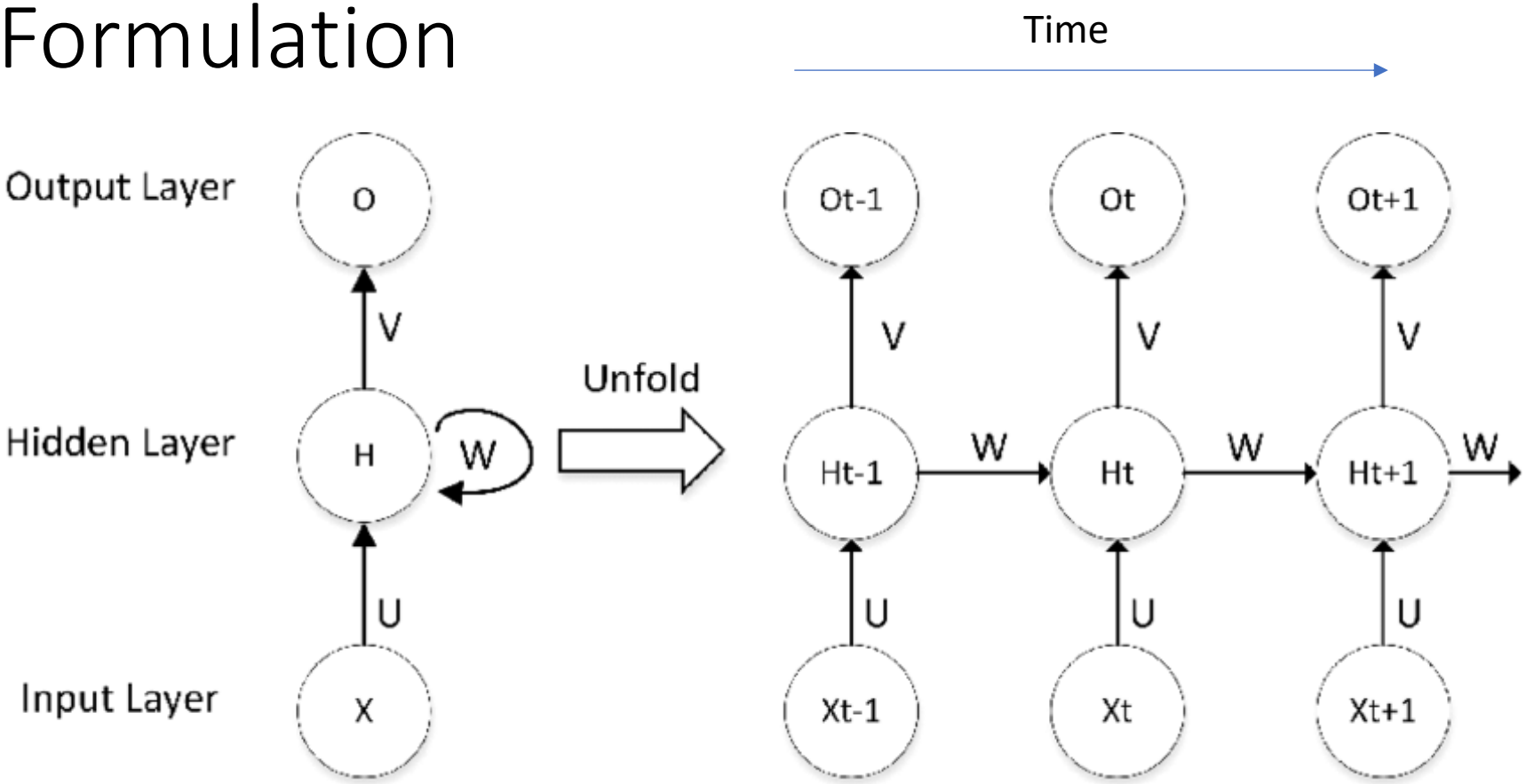
- In RNNs, the same weights are used repeatedly over time. When gradients are backpropagated through many steps, they get multiplied repeatedly by the same weights.
  - If weights are **>1**, gradients explode.
  - If weights are **<1**, gradients vanish.
    This makes training very deep or long-sequence models unstable — hence the need for LSTM, GRU, and eventually Transformers.

RNN Formulation



Same weights (U, W, V) are shared across time steps

RNN Formulation



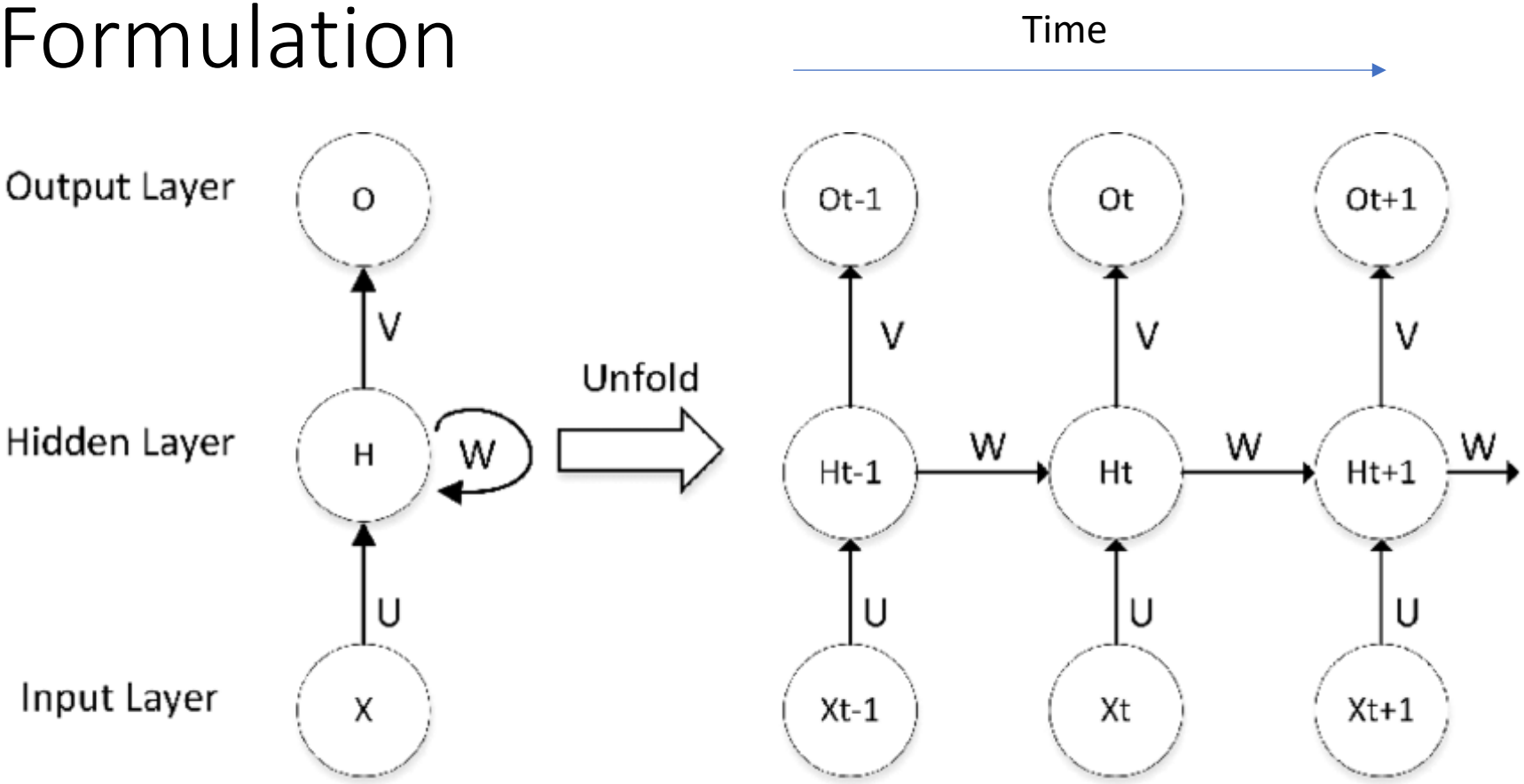
$$h_t = \mathrm{f}(h_{t-1}, x_t)$$

$$h_t = \mathrm{f}(U.x_t + W.h_{t-1} + b_h)$$

$$O_t = \mathrm{g}(V.h_t + b_y)$$

U: input-to-hidden weights
W: hidden-to-hidden weights
V: hidden-to-output weights
f: activation (e.g., tanh or ReLU)
g: output function (e.g., softmax)

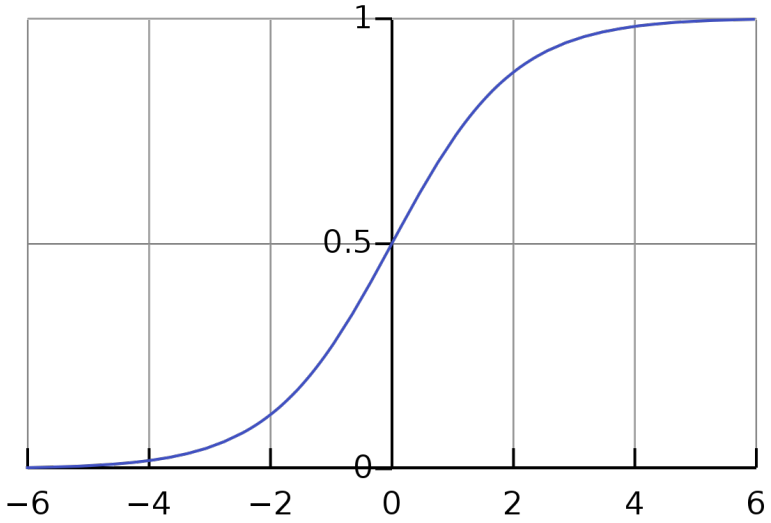
RNN Formulation



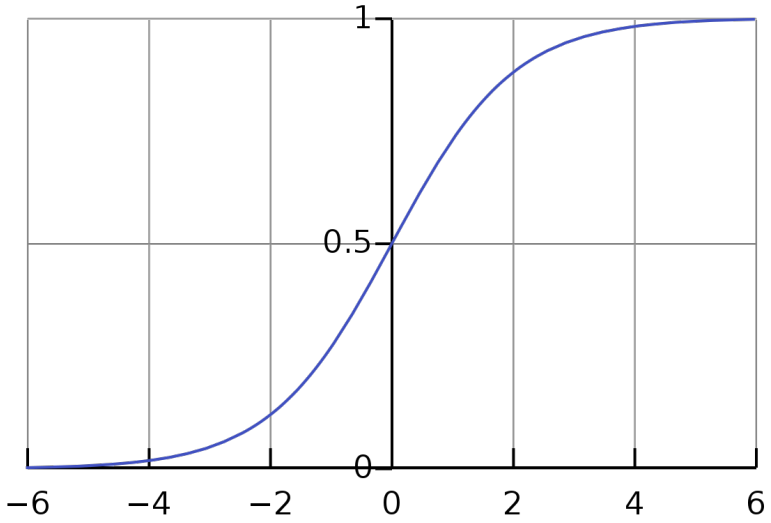
At each time step t, an RNN takes the current input xt and the previous hidden state ht−1 to compute the current hidden state ht.

This hidden state acts as a **memory**, capturing information from the past. The same network is "unrolled" across time, making it ideal for modeling sequences like text or time-series data.

Reminder: Sigmoid Function (Logistic Function)

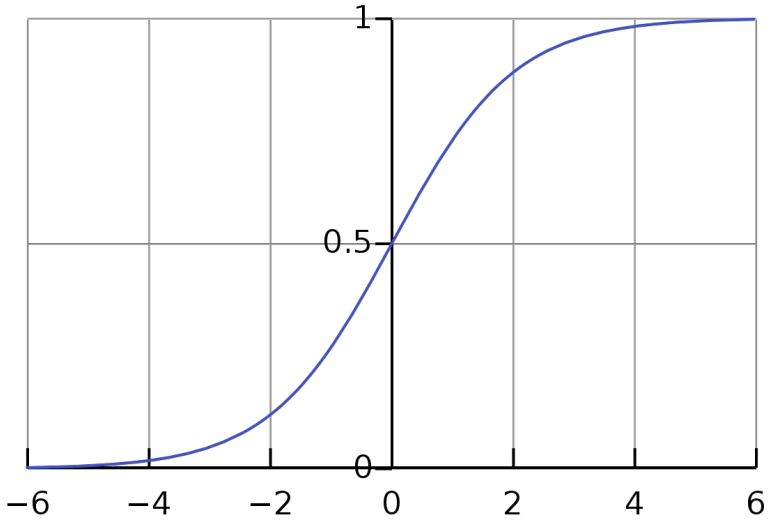


Reminder: Sigmoid Function (Logistic Function)



$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^{x}}{1 + e^{x}}$$

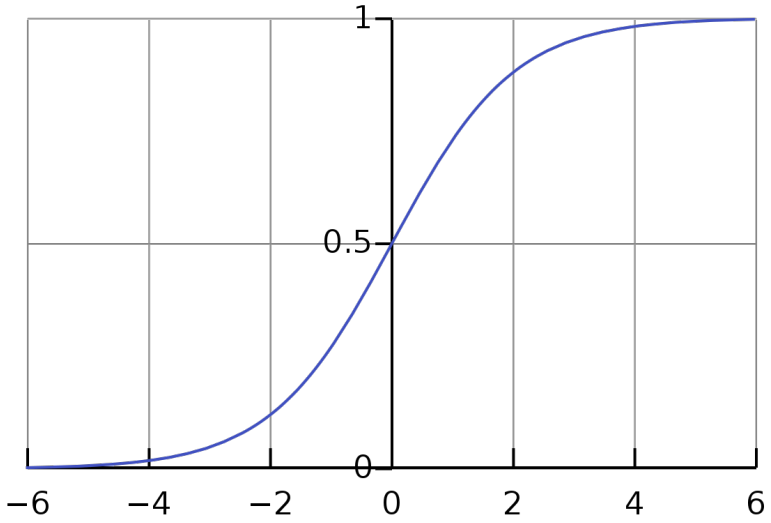
Reminder: Sigmoid Function (Logistic Function)



$$\sigma(x) = \frac{1}{1 + e^{-x}} = \frac{e^{x}}{1 + e^{x}}$$

Range is between 0 and 1

Reminder: Sigmoid Function (Logistic Function)

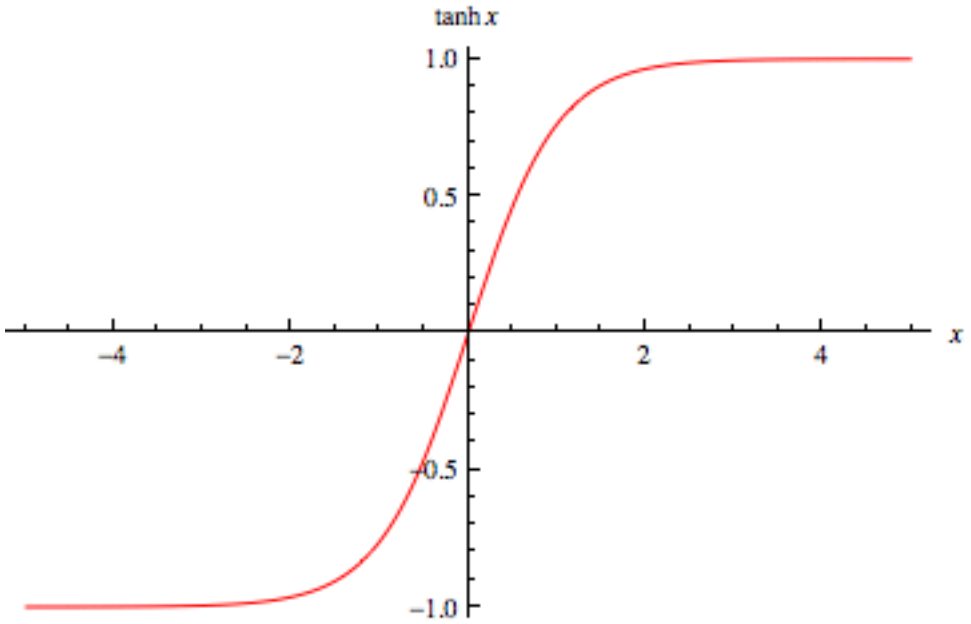


$$\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{1+e^{x}}$$

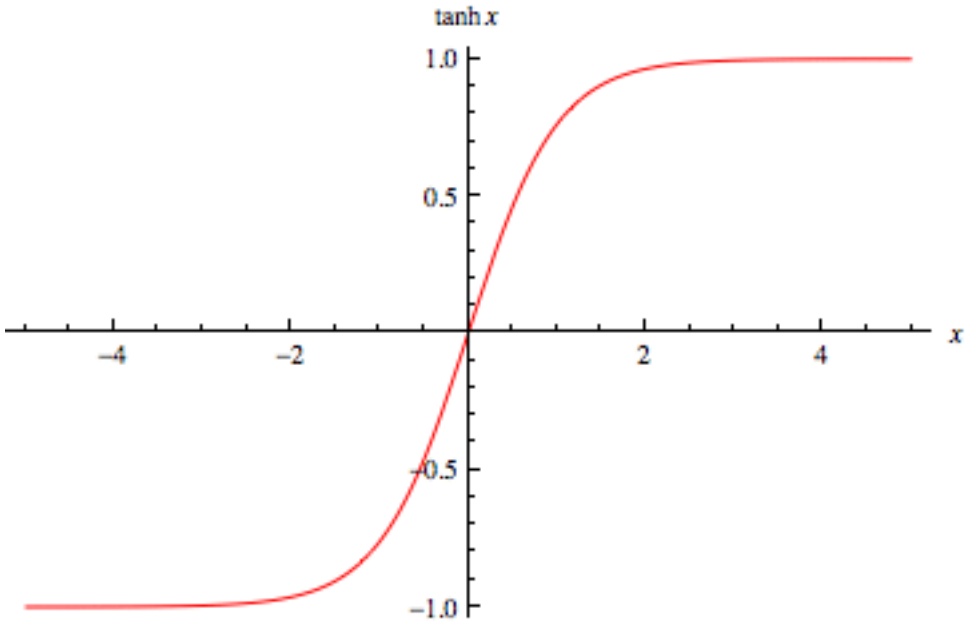
Range is between 0 and 1

$\sigma(f(x))$ maps $f(x)$ to a percentage.

Reminder: tanh Function

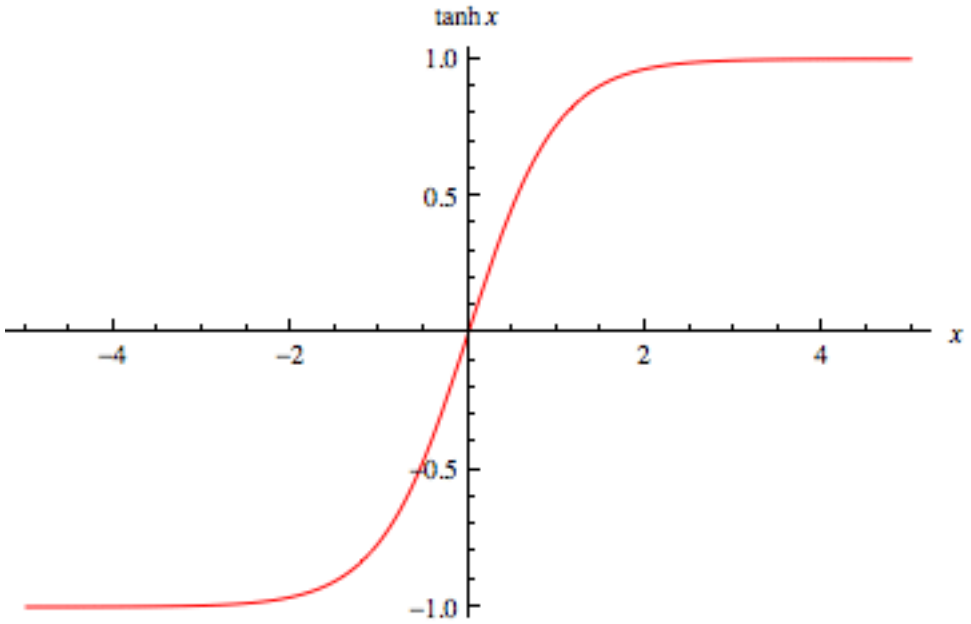


Reminder: tanh Function



$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

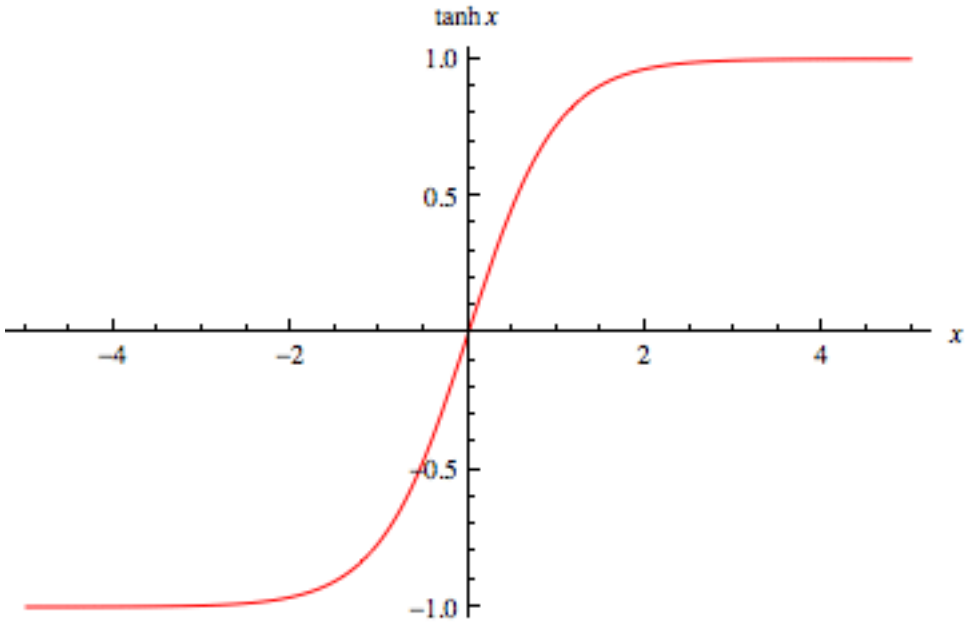
Reminder: tanh Function



$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

Range is between -1 and 1

Reminder: tanh Function

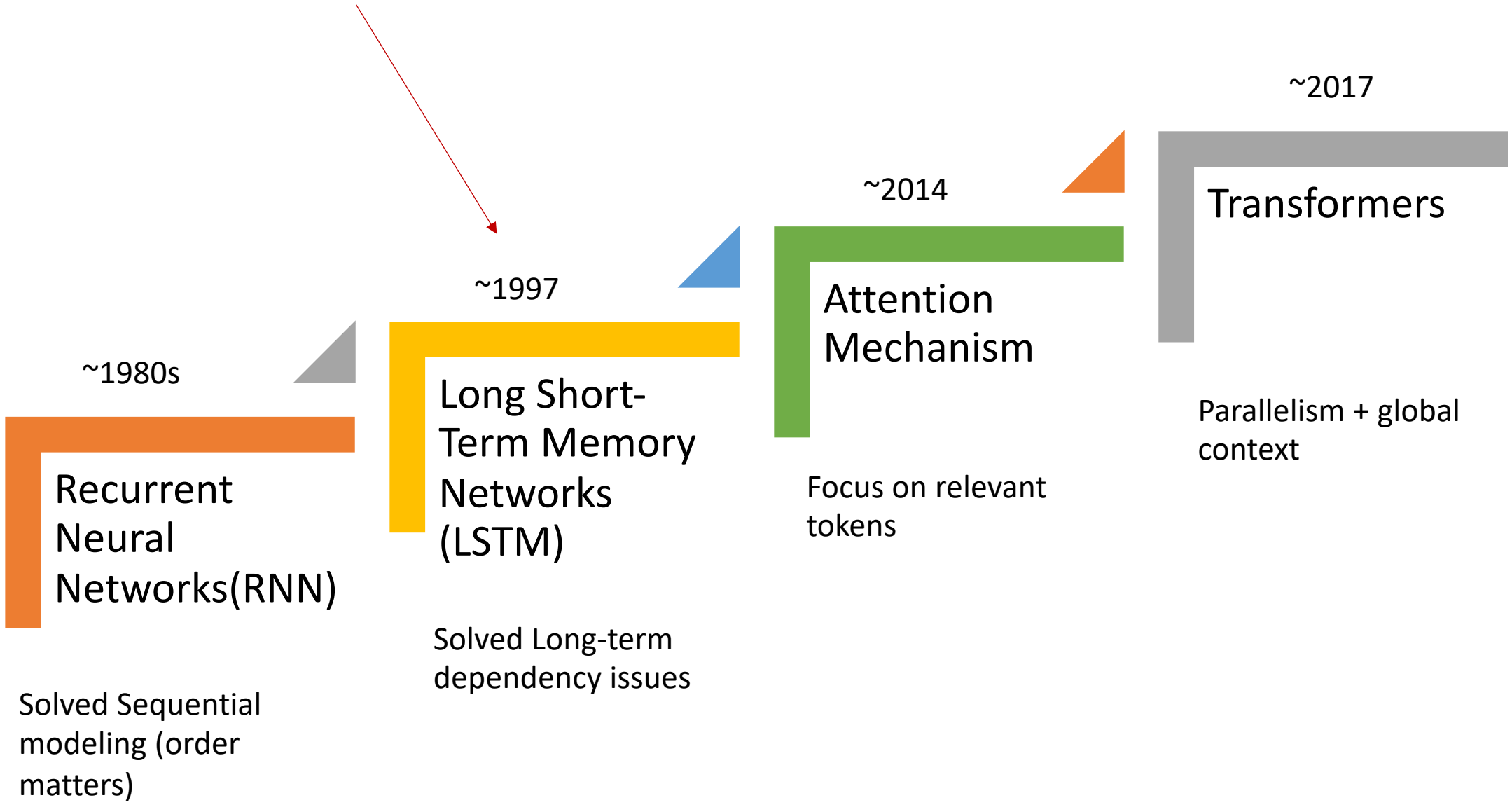


$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$$

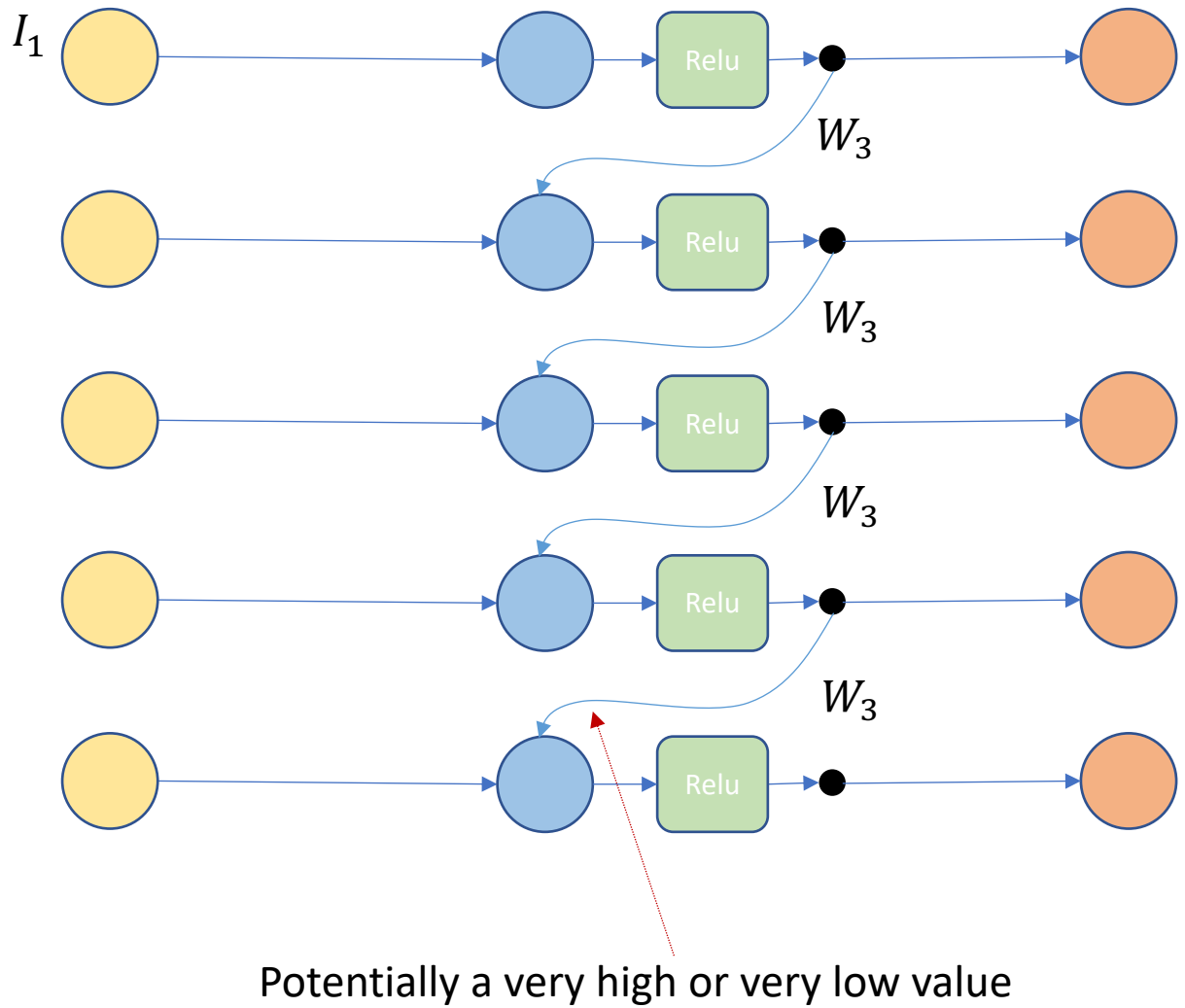
Range is between -1 and 1

$\sigma(f(x))$ maps $f(x)$ to -1 and 1.

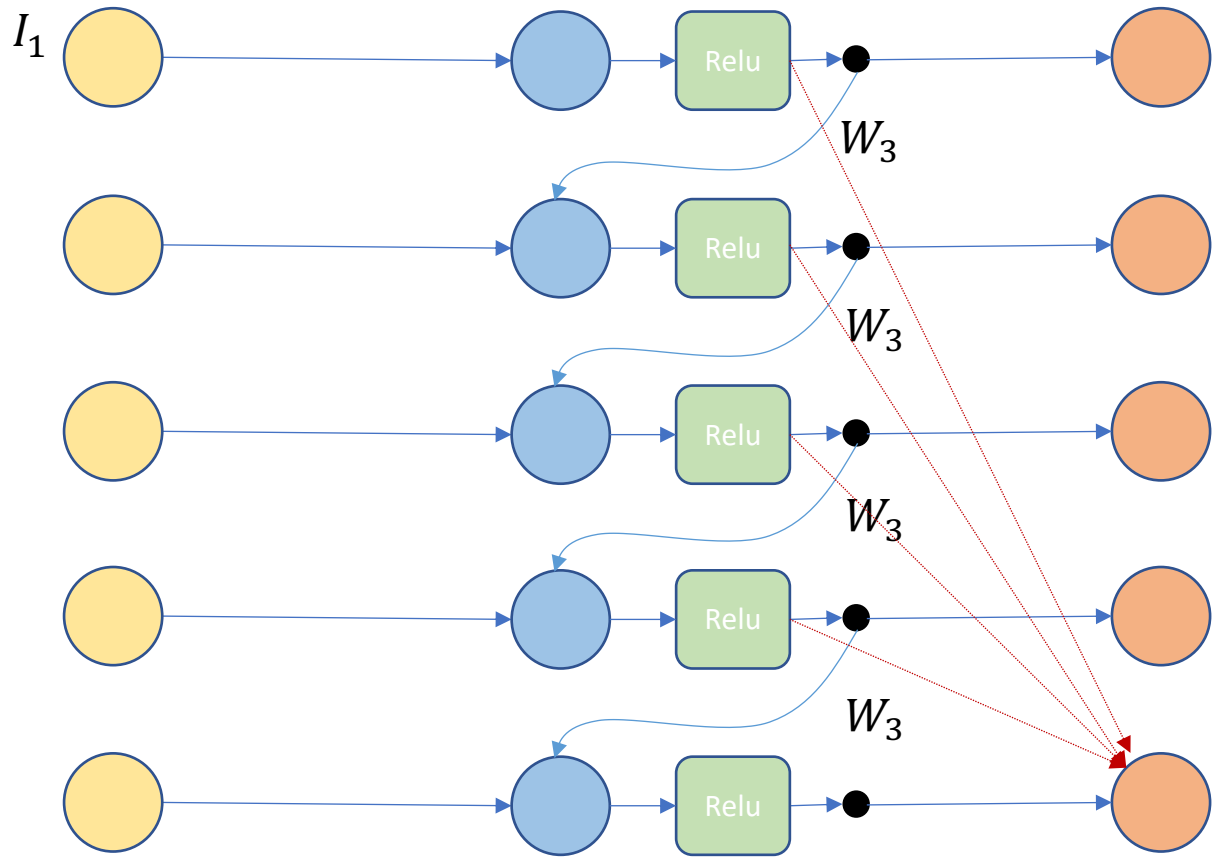
Evolution Toward Transformers



Remember the Vanishing/Exploding Gradient Problem



Remember the Vanishing/Exploding Gradient Problem



Can we have a separate path for the long-term memory?

LSTM

Forget Gate
(Filters past memory)

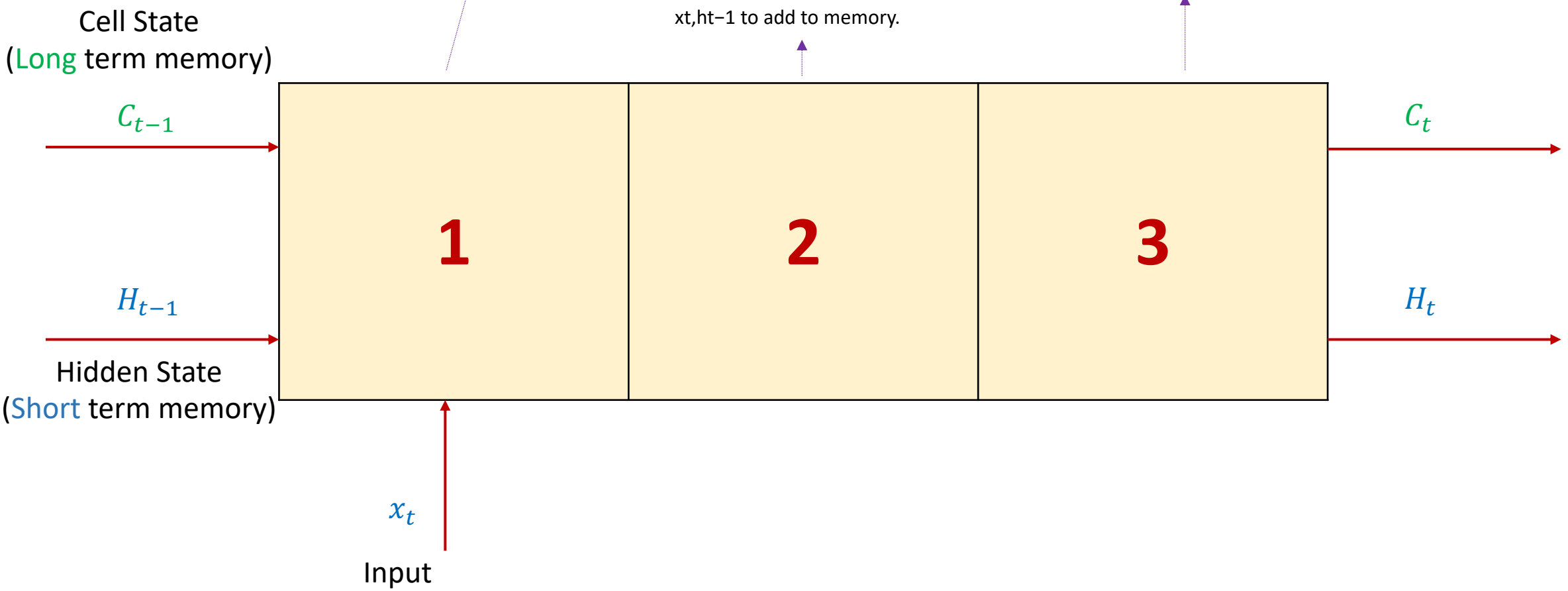
Multiplies Ct−1 by a learned forget factor.

Input Gate
(Update new information)

Decides how much new info from xt,ht−1 to add to memory.

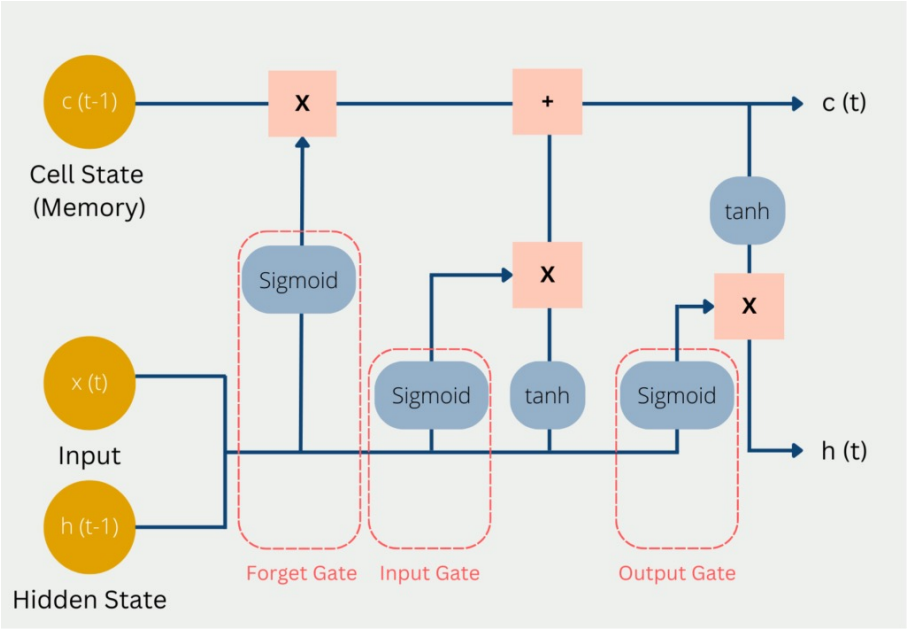
Output Gate
(Pass updated information)

Decides how much of the cell state affects the next hidden state ht.



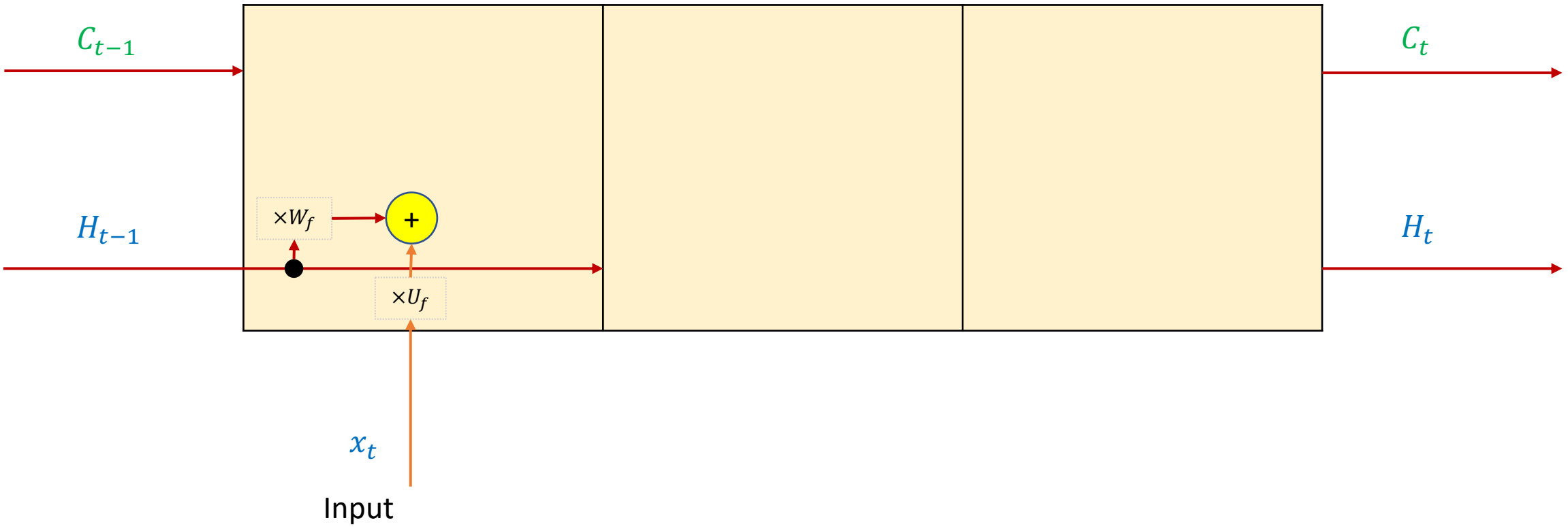
LSTM

- LSTM introduces **gates** to selectively update its memory:
  - The **Forget Gate** decides what past information to discard.
  - The **Input Gate** decides what new information to store.
  - The **Output Gate** controls what information to expose to the next time step. This helps LSTMs handle **long-term dependencies** better than vanilla RNNs.

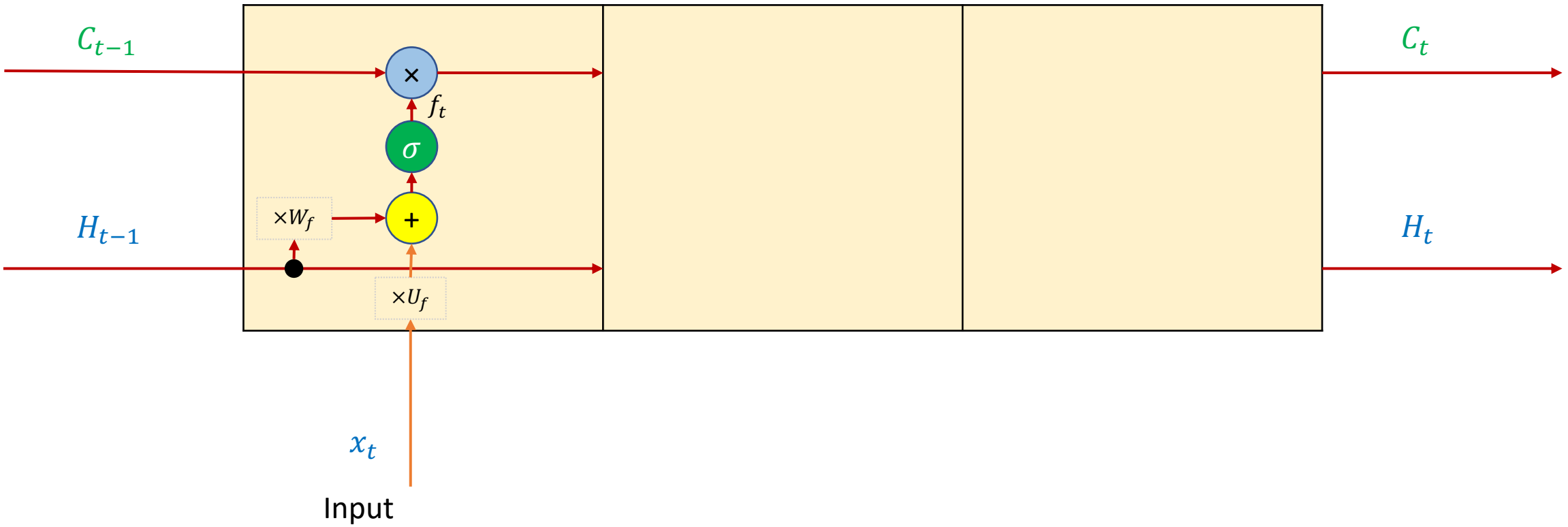


Source: codificandobits https://databasecamp.de/wp-content/uploads/lstm-architecture-1024x709.png.

LSTM – Forget Gate

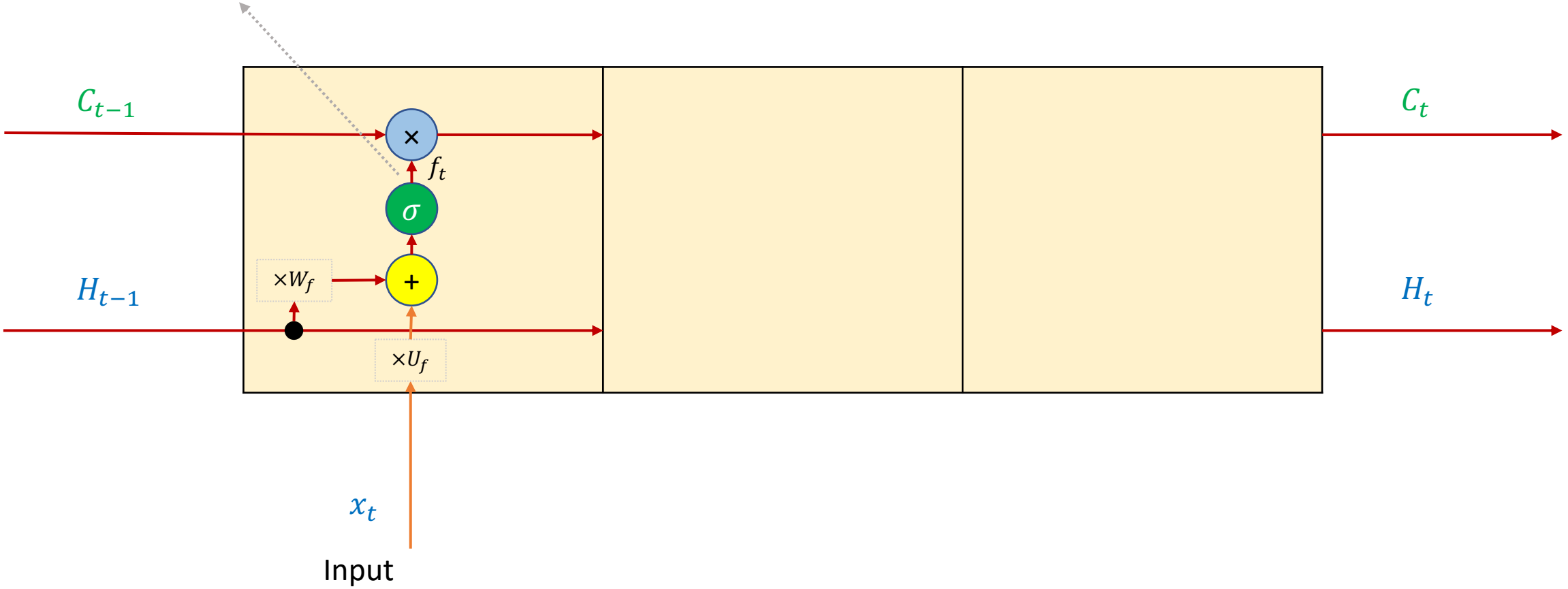


LSTM – Forget Gate



LSTM – Forget Gate

$$f_t = \sigma(W_f H_{t-1} + U_f x_t)$$

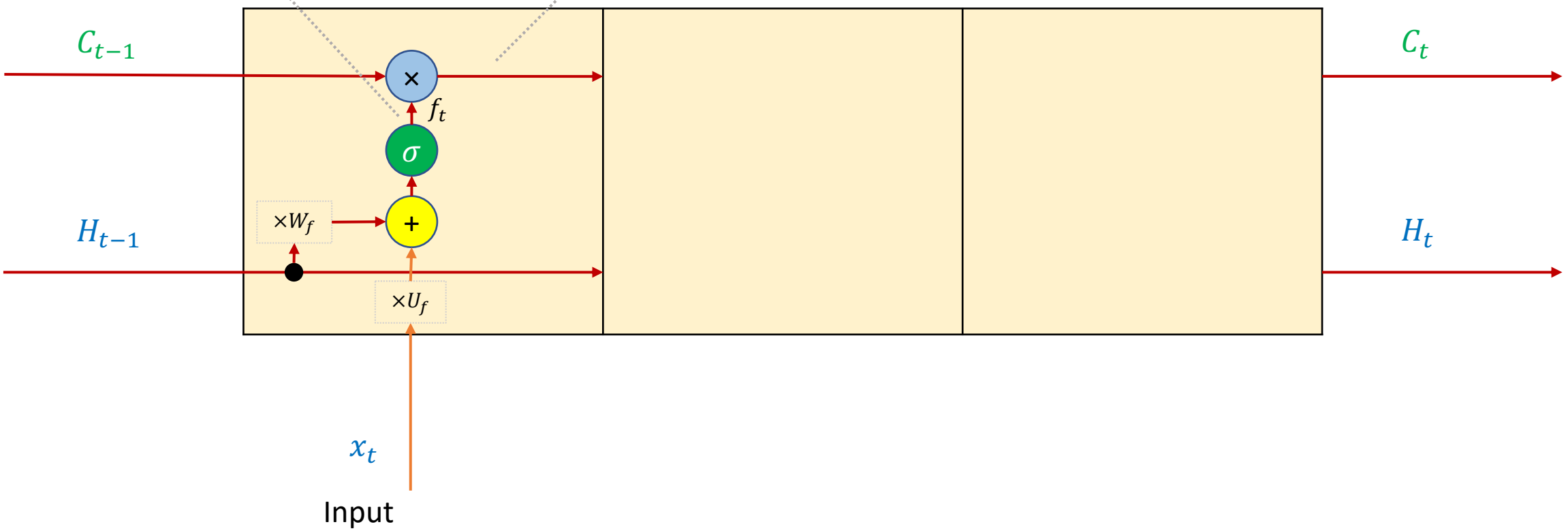


LSTM – Forget Gate

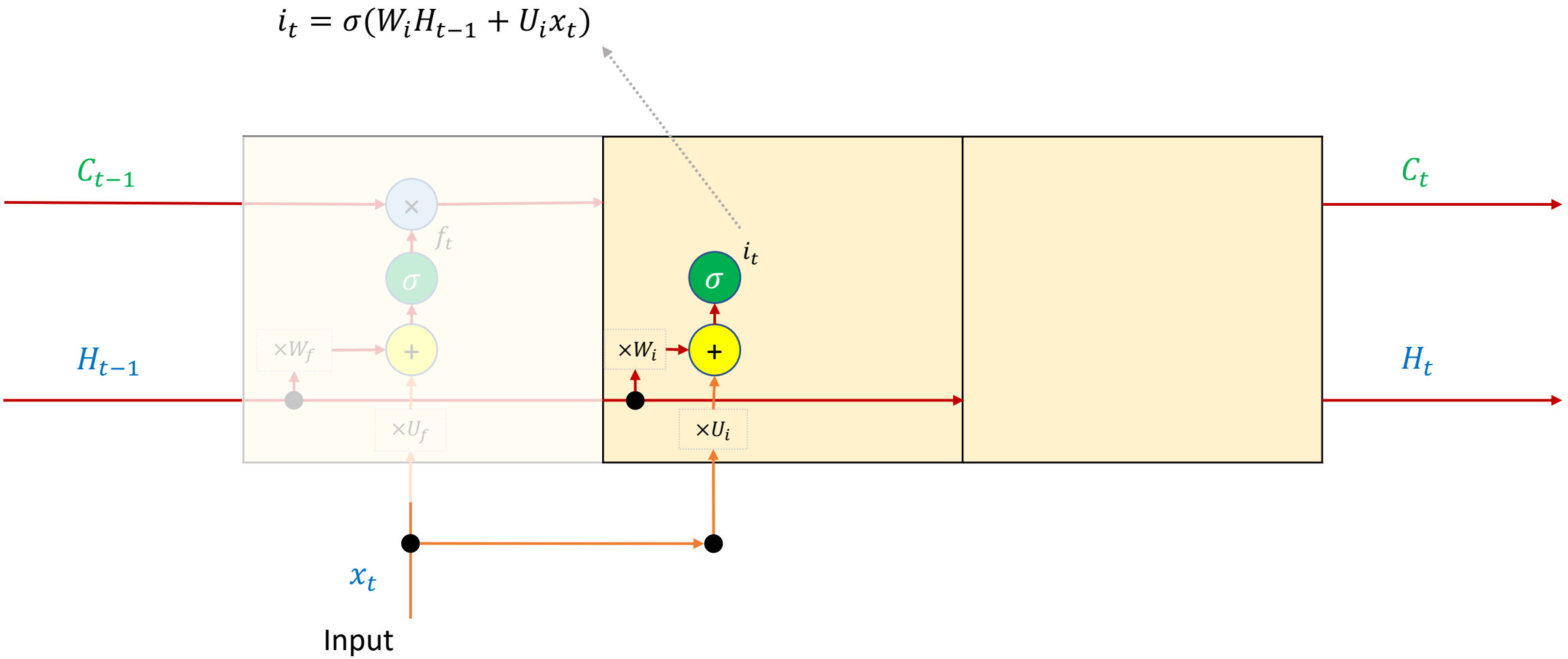
How much of the long-term memory should be forgotten

$$f_t = \sigma(W_f H_{t-1} + U_f x_t)$$

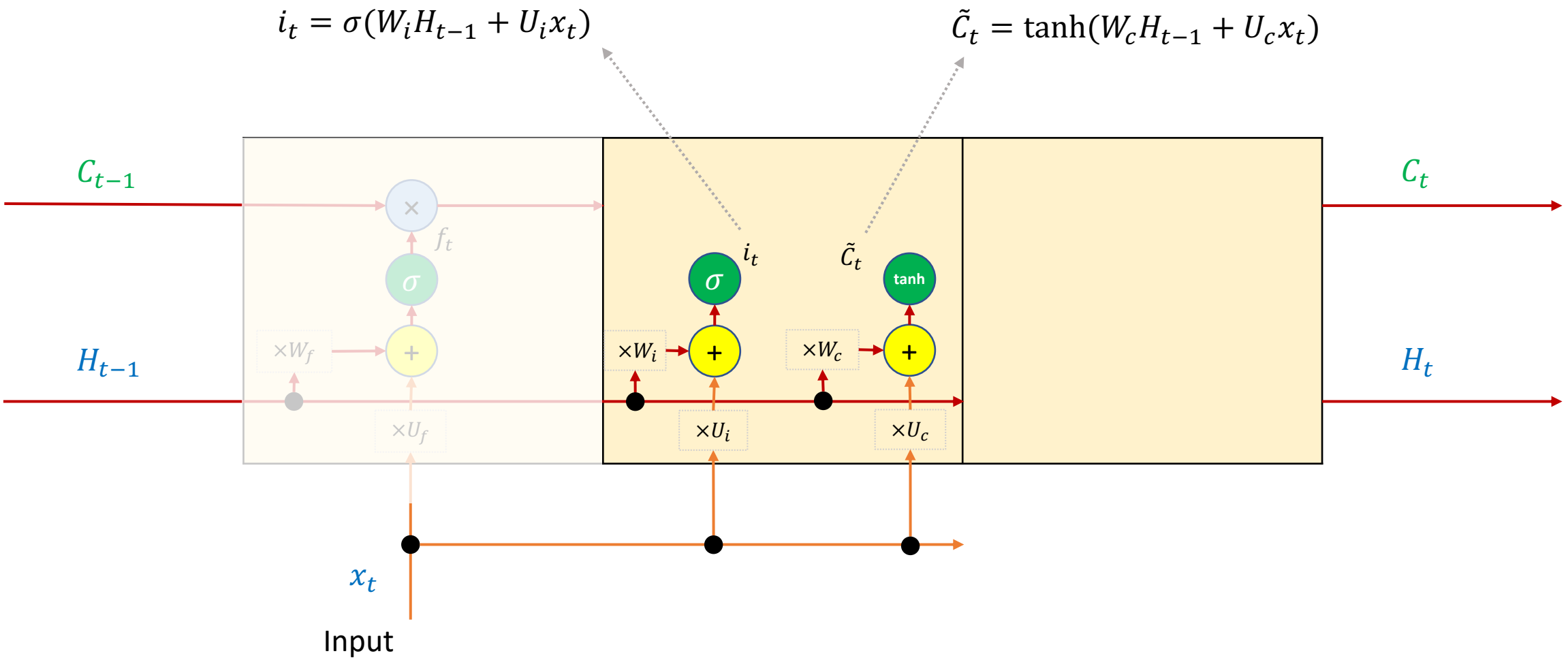
$$= \begin{cases} 0, & f_t = 0 \\ C_{t-1}, & f_t = 1 \end{cases}$$



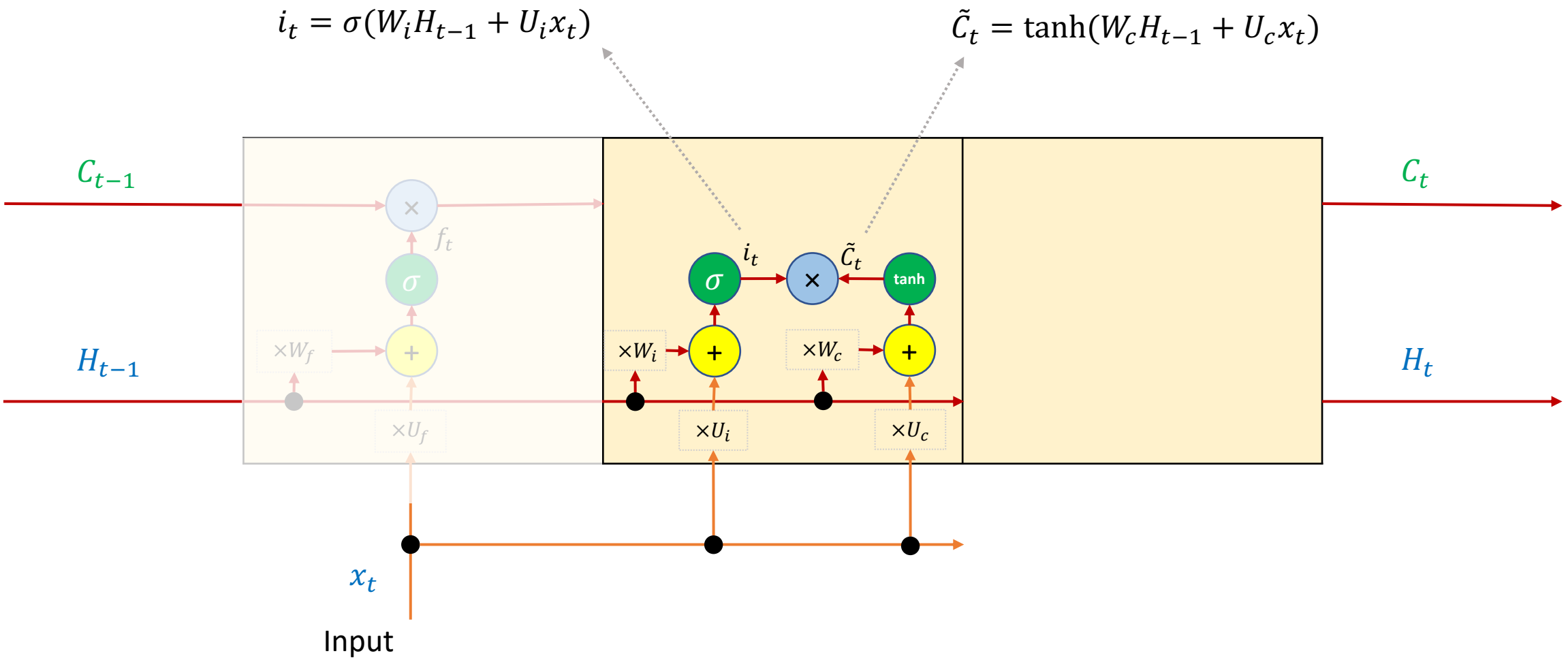
LSTM – Input Gate



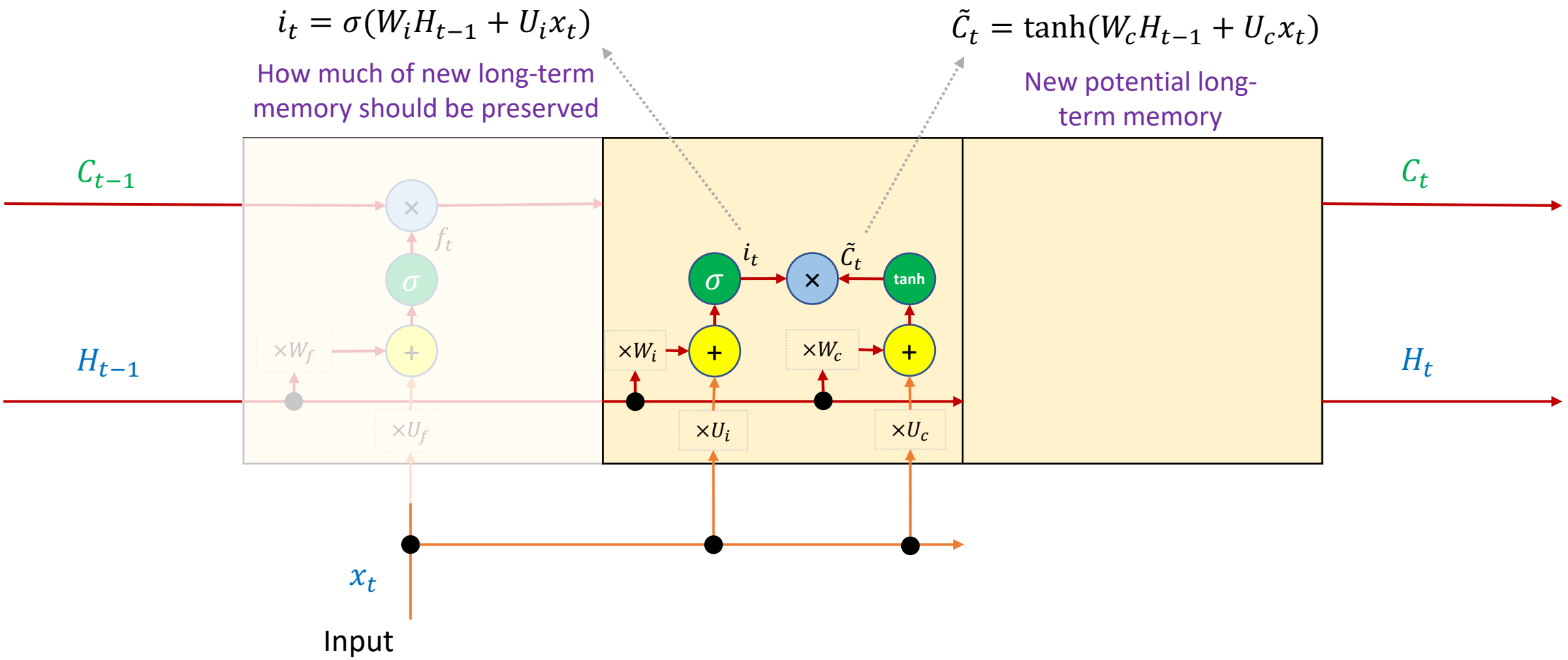
LSTM – Input Gate



LSTM – Input Gate



LSTM – Input Gate



LSTM – Input Gate

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

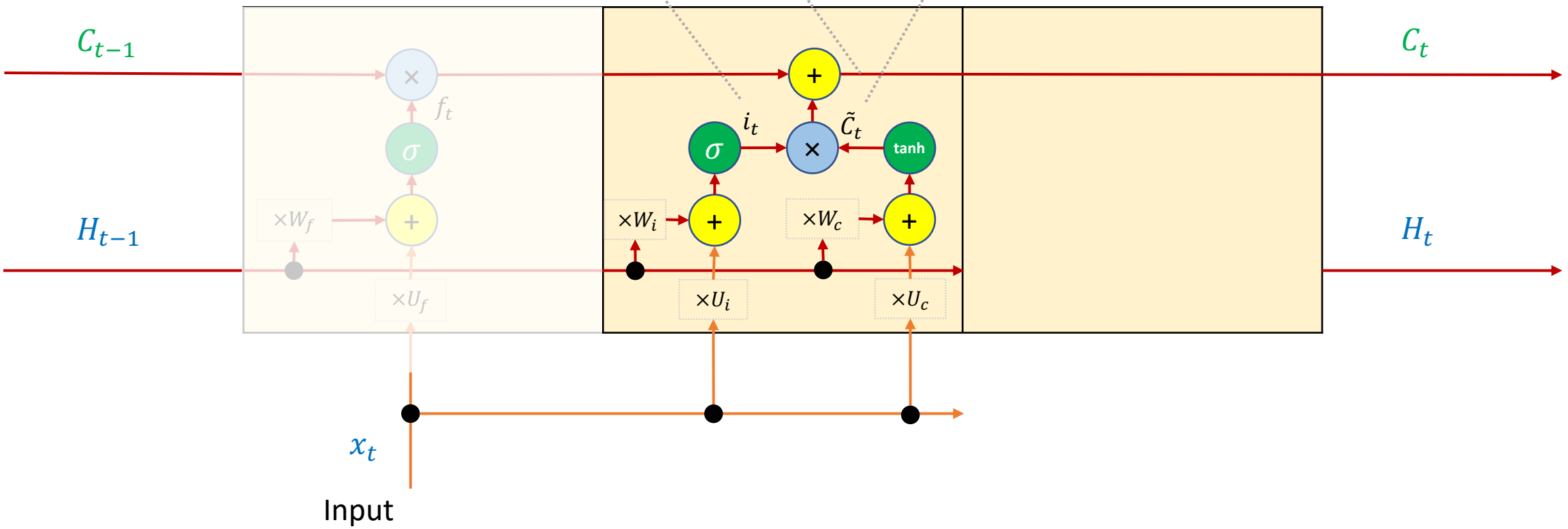
New long-term memory

$$i_t = \sigma(W_i H_{t-1} + U_i x_t)$$

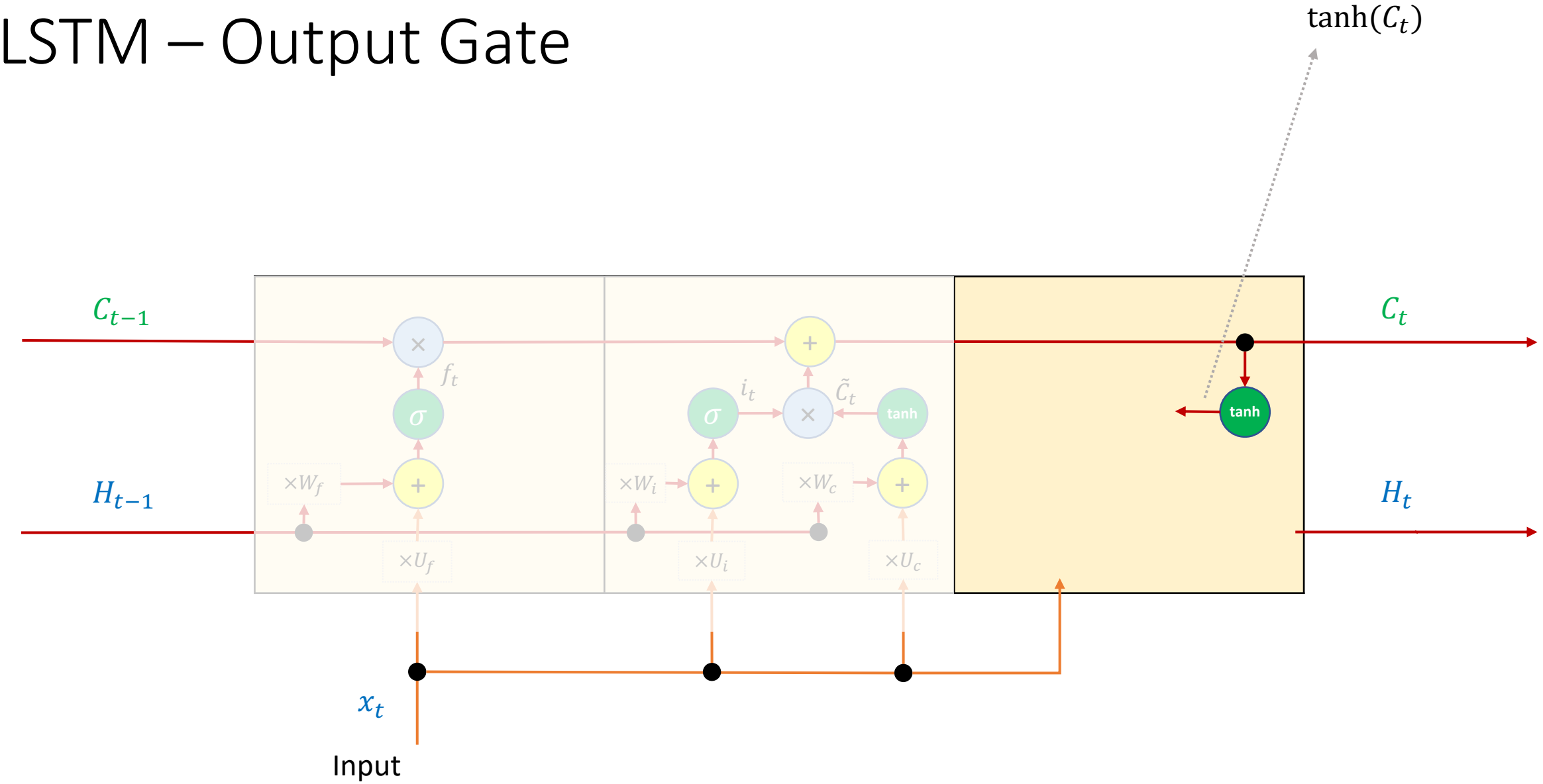
How much of new long-term memory should be preserved

$$\tilde{C}_t = \tanh(W_c H_{t-1} + U_c x_t)$$

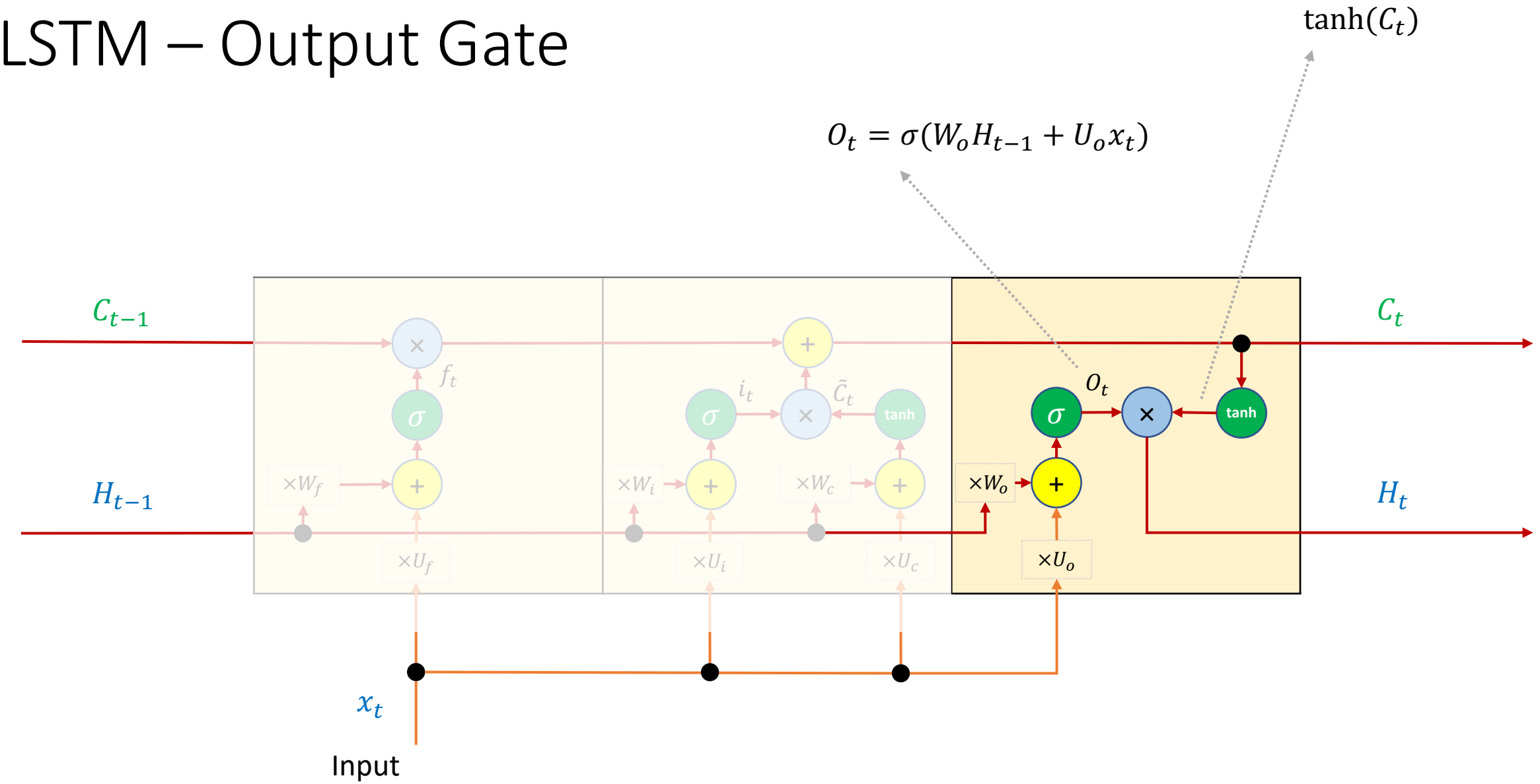
New potential long-term memory



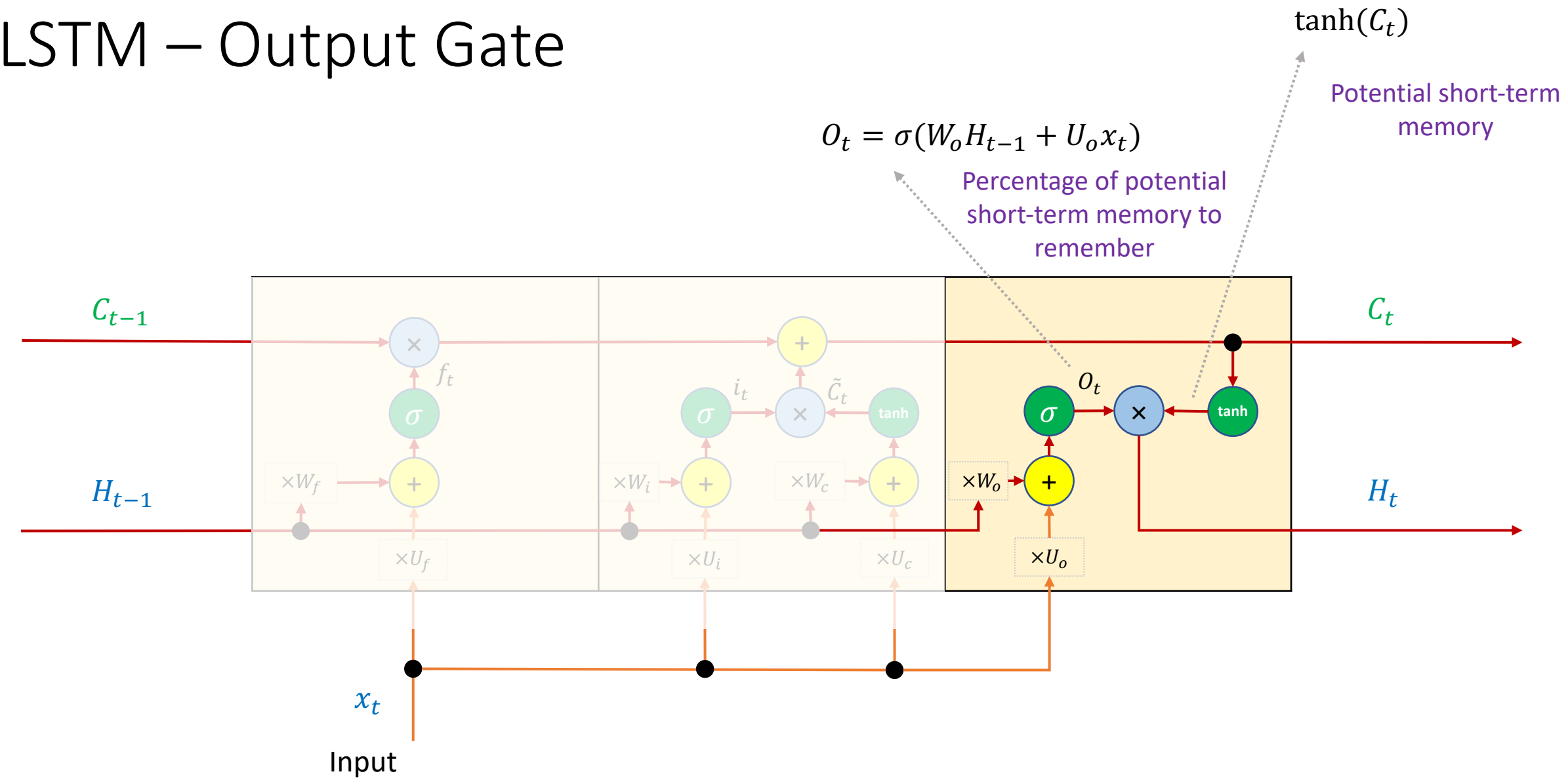
LSTM – Output Gate



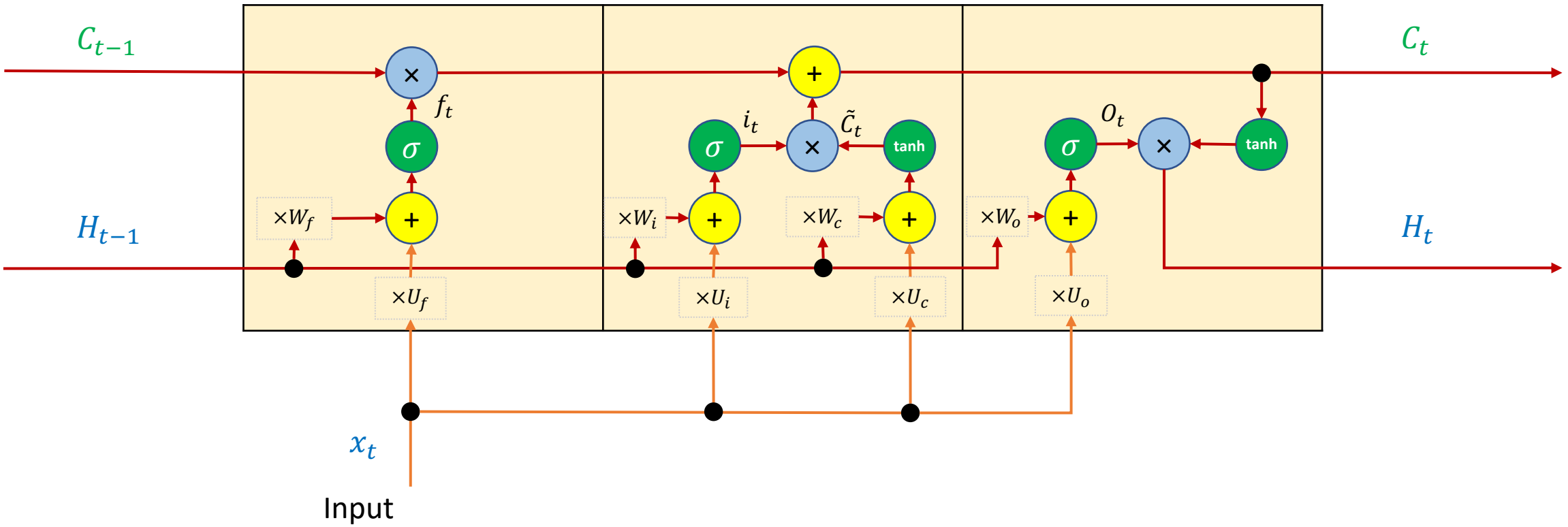
LSTM – Output Gate



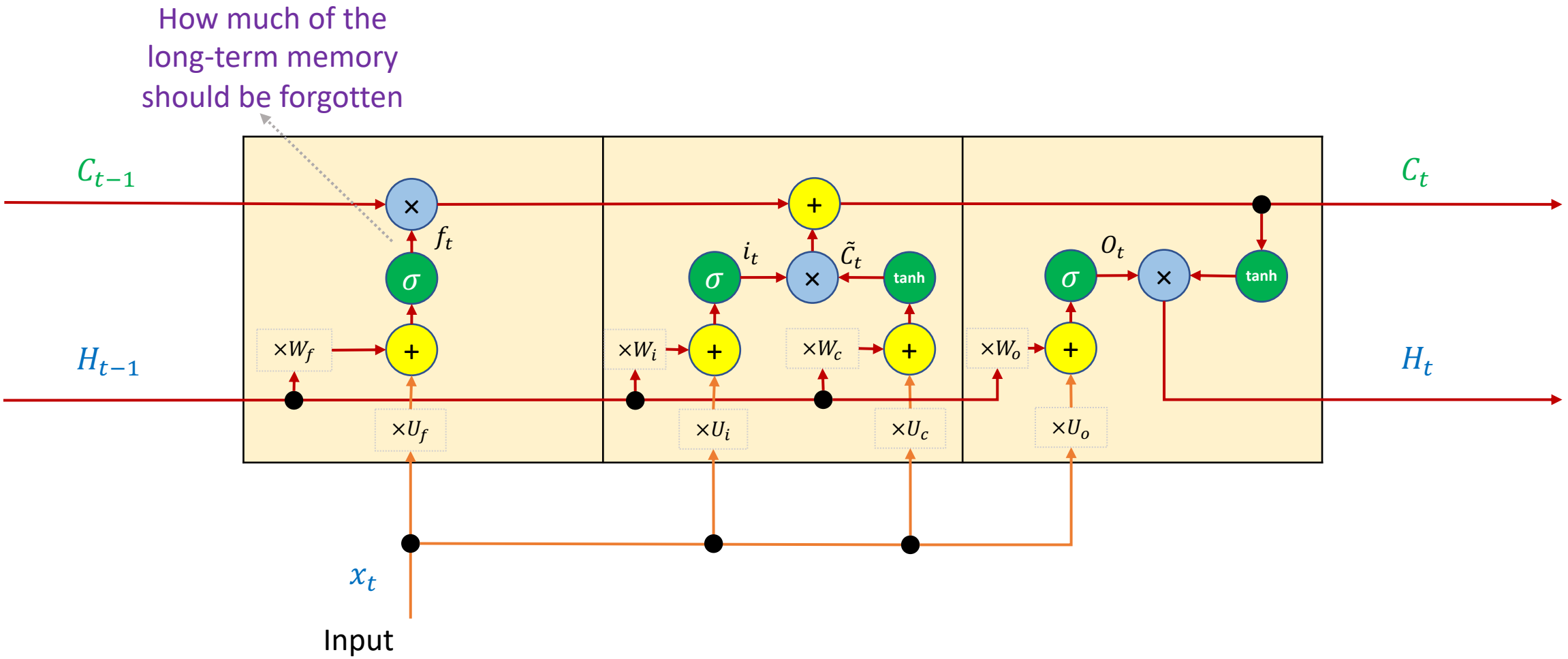
LSTM – Output Gate



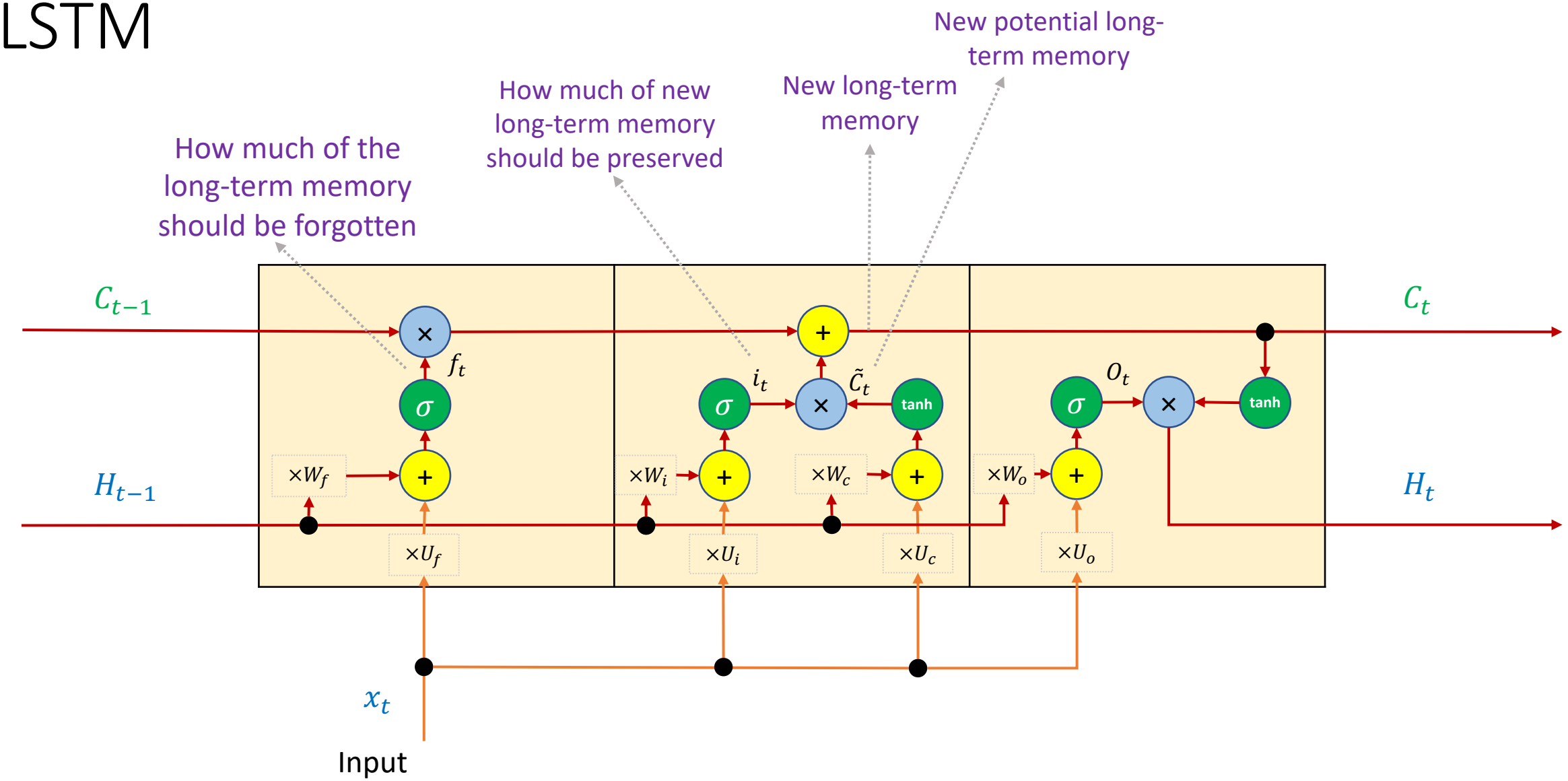
LSTM


C_{t-1}
H_{t-1}
C_t
H_t
f_t
i_t
$\tilde{C}_t$
O_t
σ
tanh
$\times W_f$
$\times W_i$
$\times W_c$
$\times W_o$
$\times U_f$
$\times U_i$
$\times U_c$
$\times U_o$
x_t
Input

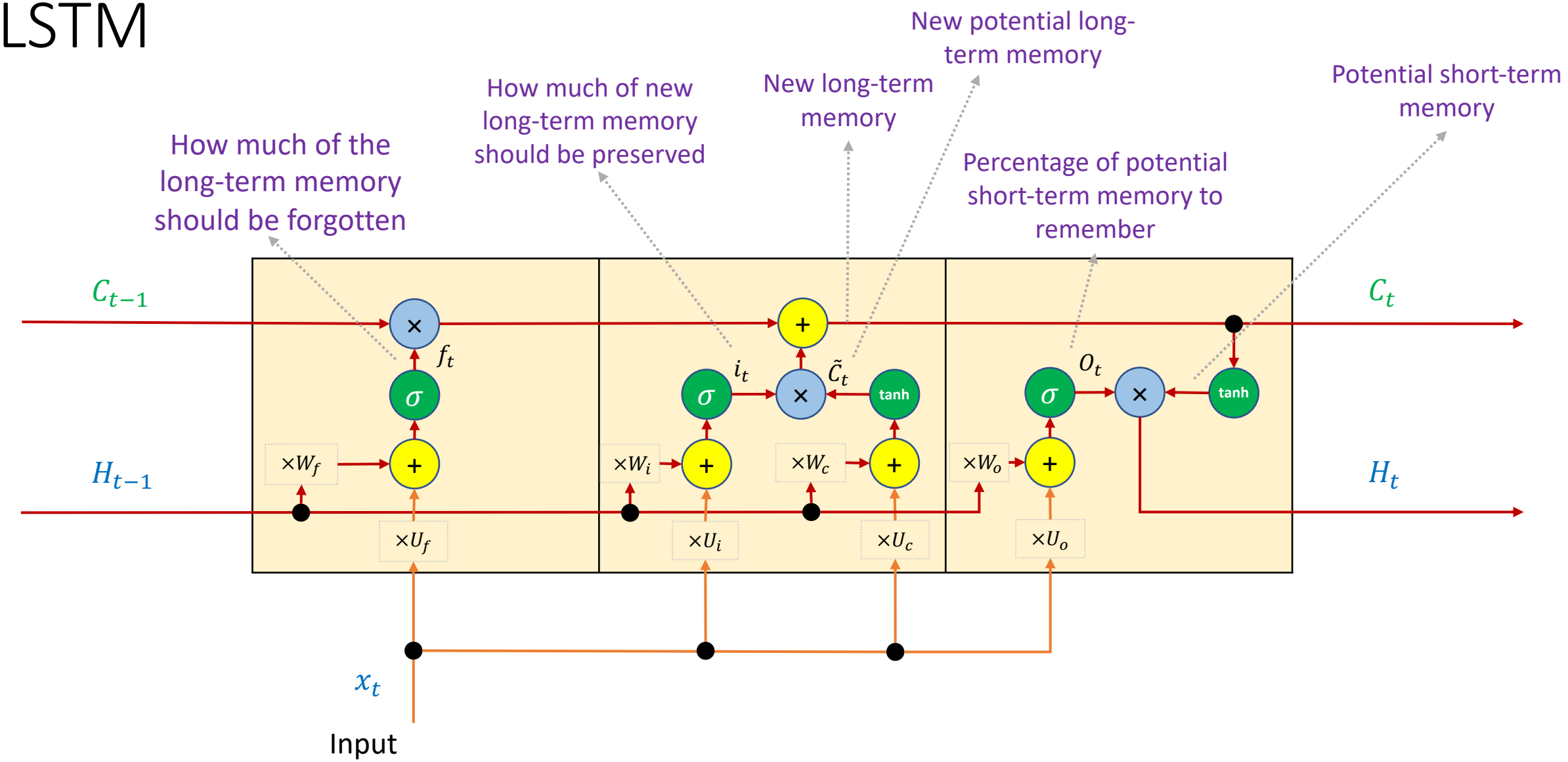
LSTM


How much of the long-term memory should be forgotten
C_{t-1}
C_t
H_{t-1}
H_t
f_t
i_t
$\tilde{C}_t$
O_t
σ
tanh
$\times W_f$
$\times W_i$
$\times W_c$
$\times W_o$
$\times U_f$
$\times U_i$
$\times U_c$
$\times U_o$
x_t
Input

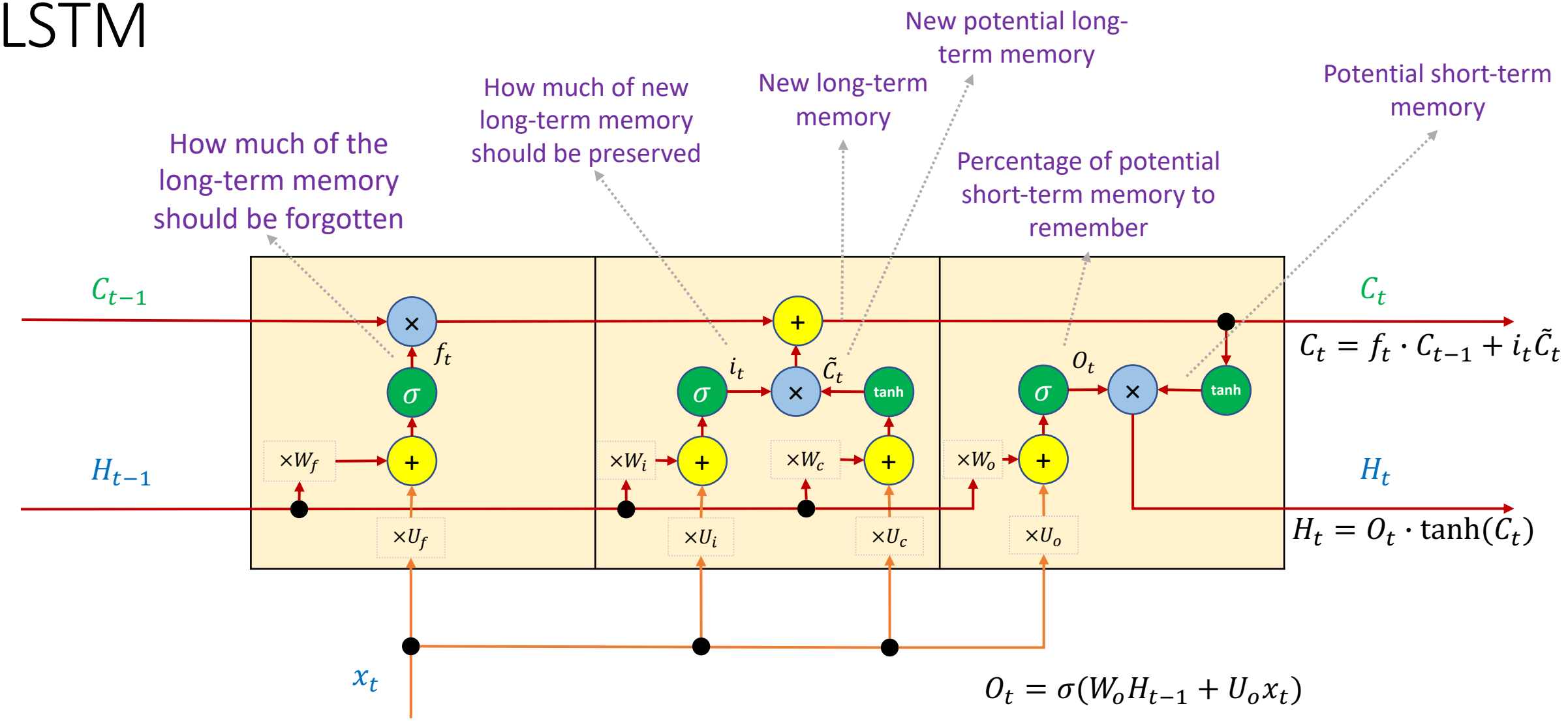
LSTM


How much of the long-term memory should be forgotten
How much of new long-term memory should be preserved
New long-term memory
New potential long-term memory
C_{t-1}
C_t
H_{t-1}
H_t
f_t
i_t
$\tilde{C}_t$
O_t
σ
tanh
$\times W_f$
$\times W_i$
$\times W_c$
$\times W_o$
$\times U_f$
$\times U_i$
$\times U_c$
$\times U_o$
x_t
Input

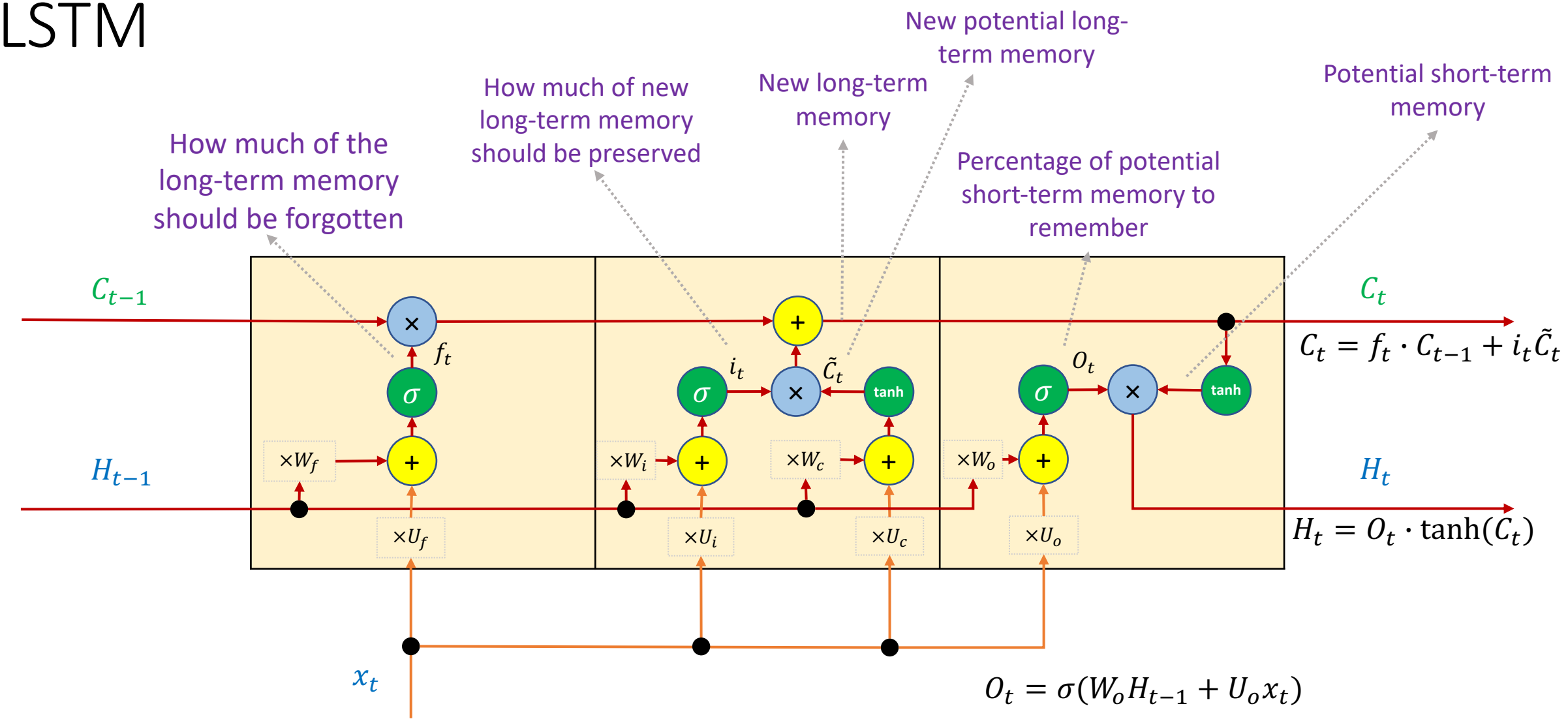
LSTM


How much of the long-term memory should be forgotten
How much of new long-term memory should be preserved
New long-term memory
New potential long-term memory
Percentage of potential short-term memory to remember
Potential short-term memory
C_{t-1}
C_t
H_{t-1}
H_t
f_t
i_t
$\tilde{C}_t$
O_t
σ
tanh
$\times W_f$
$\times W_i$
$\times W_c$
$\times W_o$
$\times U_f$
$\times U_i$
$\times U_c$
$\times U_o$
x_t
Input

LSTM

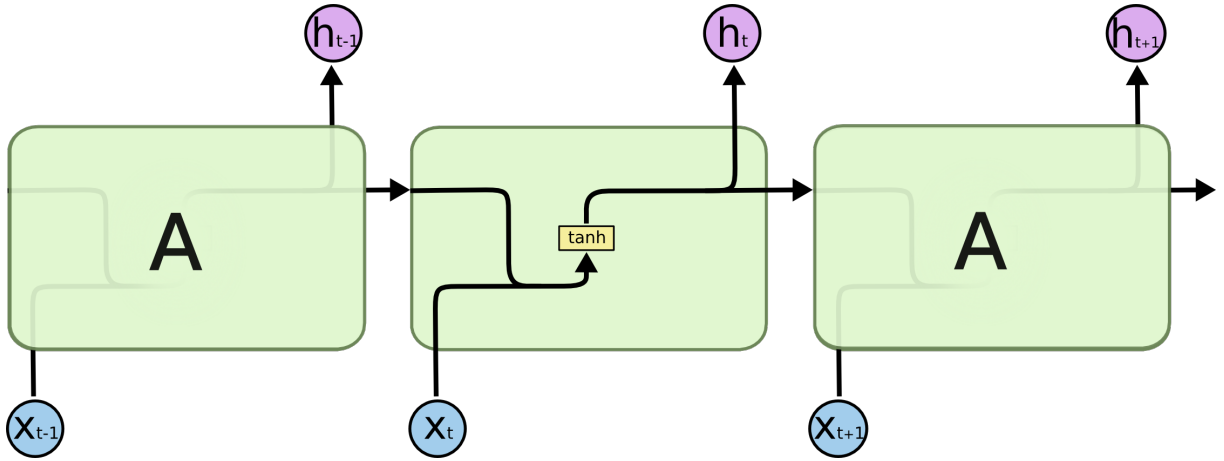

How much of the long-term memory should be forgotten
How much of new long-term memory should be preserved
New long-term memory
New potential long-term memory
Percentage of potential short-term memory to remember
Potential short-term memory
C_{t-1}
H_{t-1}
x_t
f_t
i_t
$\tilde{C}_t$
O_t
$\times W_f$
$\times U_f$
$\times W_i$
$\times U_i$
$\times W_c$
$\times U_c$
$\times W_o$
$\times U_o$
σ
tanh
C_t
H_t
$C_t = f_t \cdot C_{t-1} + i_t \tilde{C}_t$
$H_t = O_t \cdot \tanh(C_t)$
Input
$f_t = \sigma(W_f H_{t-1} + U_f x_t)$
$i_t = \sigma(W_i H_{t-1} + U_i x_t)$
$\tilde{C}_t = \tanh(W_c H_{t-1} + U_c x_t)$
$O_t = \sigma(W_o H_{t-1} + U_o x_t)$

LSTM


How much of the long-term memory should be forgotten
How much of new long-term memory should be preserved
New long-term memory
New potential long-term memory
Percentage of potential short-term memory to remember
Potential short-term memory
C_{t-1}
H_{t-1}
x_t
C_t
H_t
f_t
i_t
$\tilde{C}_t$
O_t
σ
tanh
$\times W_f$
$\times U_f$
$\times W_i$
$\times U_i$
$\times W_c$
$\times U_c$
$\times W_o$
$\times U_o$
Input
$C_t = f_t \cdot C_{t-1} + i_t \tilde{C}_t$
$H_t = O_t \cdot \tanh(C_t)$
$O_t = \sigma(W_o H_{t-1} + U_o x_t)$
$i_t = \sigma(W_i H_{t-1} + U_i x_t)$
$f_t = \sigma(W_f H_{t-1} + U_f x_t)$
$\tilde{C}_t = \tanh(W_c H_{t-1} + U_c x_t)$
$H_t, C_t = \mathrm{LSTM}(H_{t-1}, x_t, C_{t-1})$

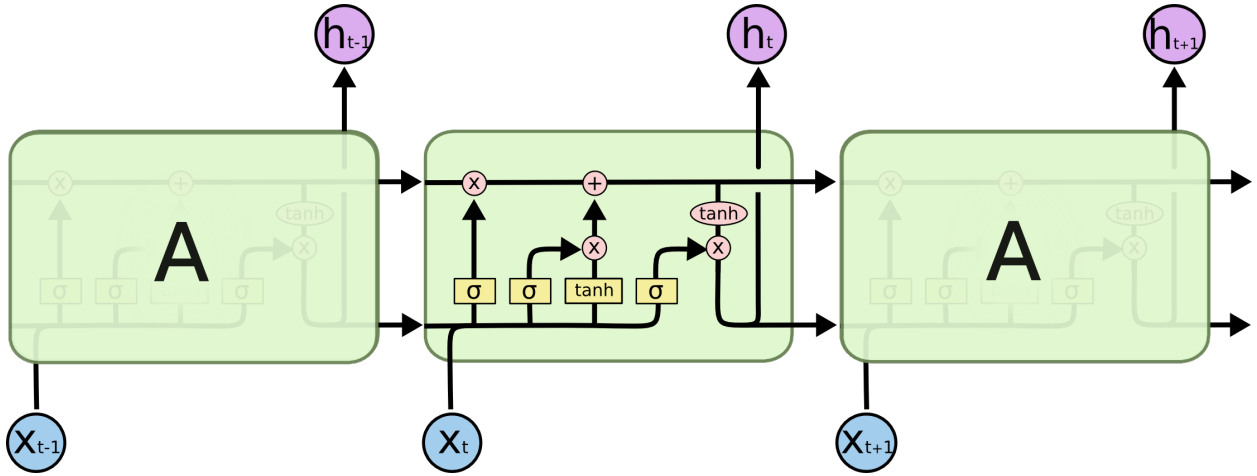
LSTM vs RNN

LSTMs address the vanishing gradient problem in RNNs by introducing a cell state and gating mechanisms, enabling them to capture long-range dependencies.



The Cell State (Memory):

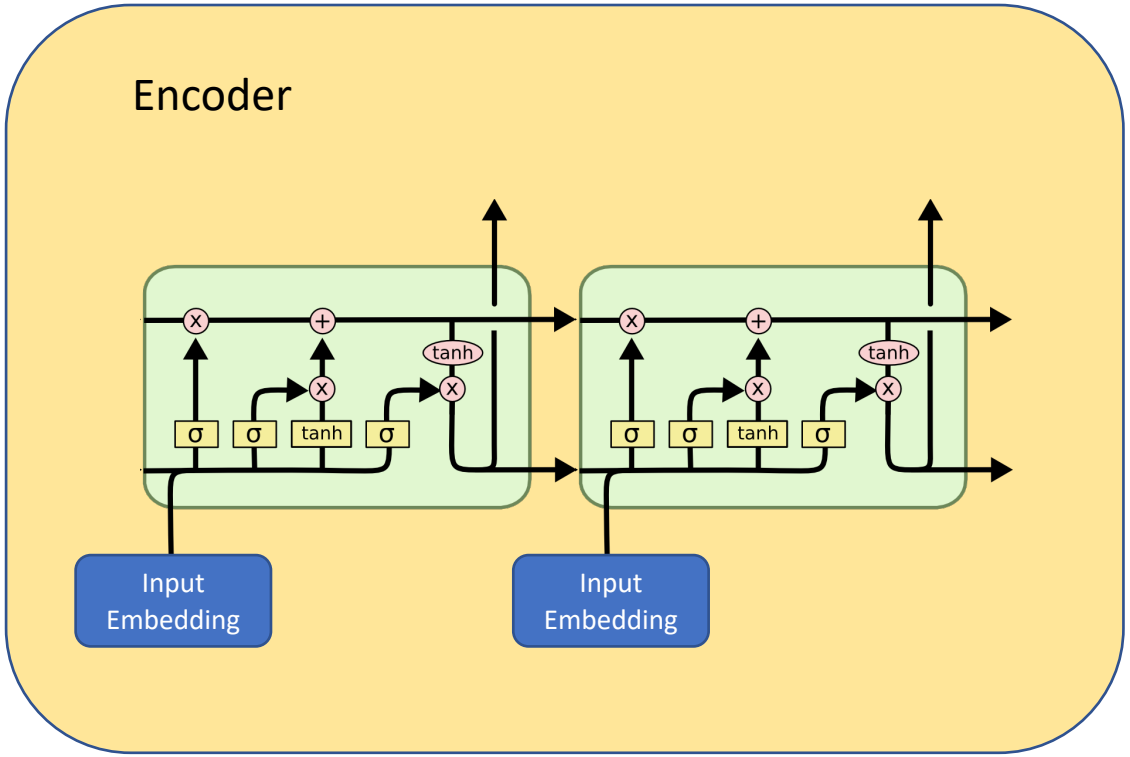
- Instead of just a hidden state that's constantly being overwritten, LSTMs introduce a "cell state" (often denoted as 'C'). This cell state acts like a conveyor belt, carrying information through the entire sequence.
- Information can flow through the cell state relatively unchanged, allowing gradients to propagate backward through many time steps. This helps preserve long-term dependencies.



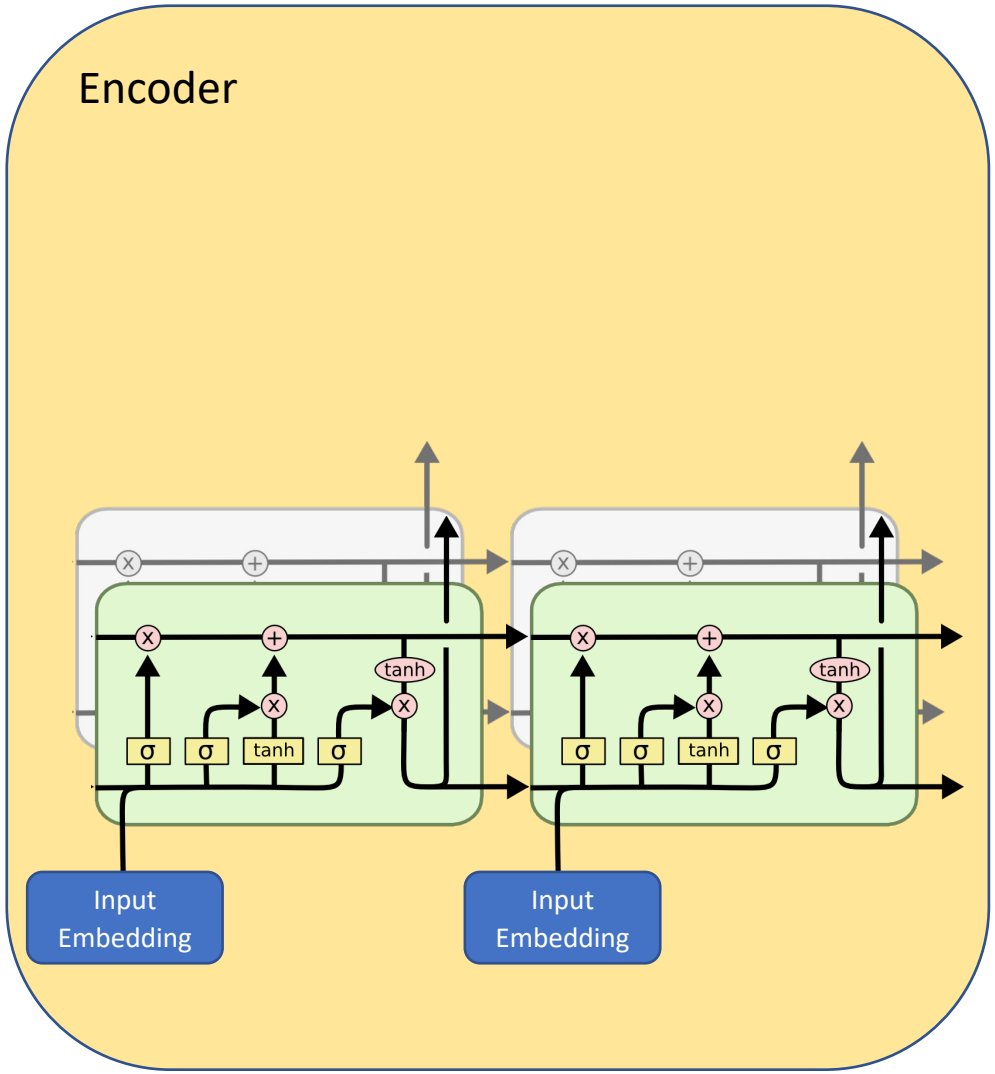
Note: The weights and biases are the same for each level

Source: colah.github.io

Seq2Seq Encoder-Decoder

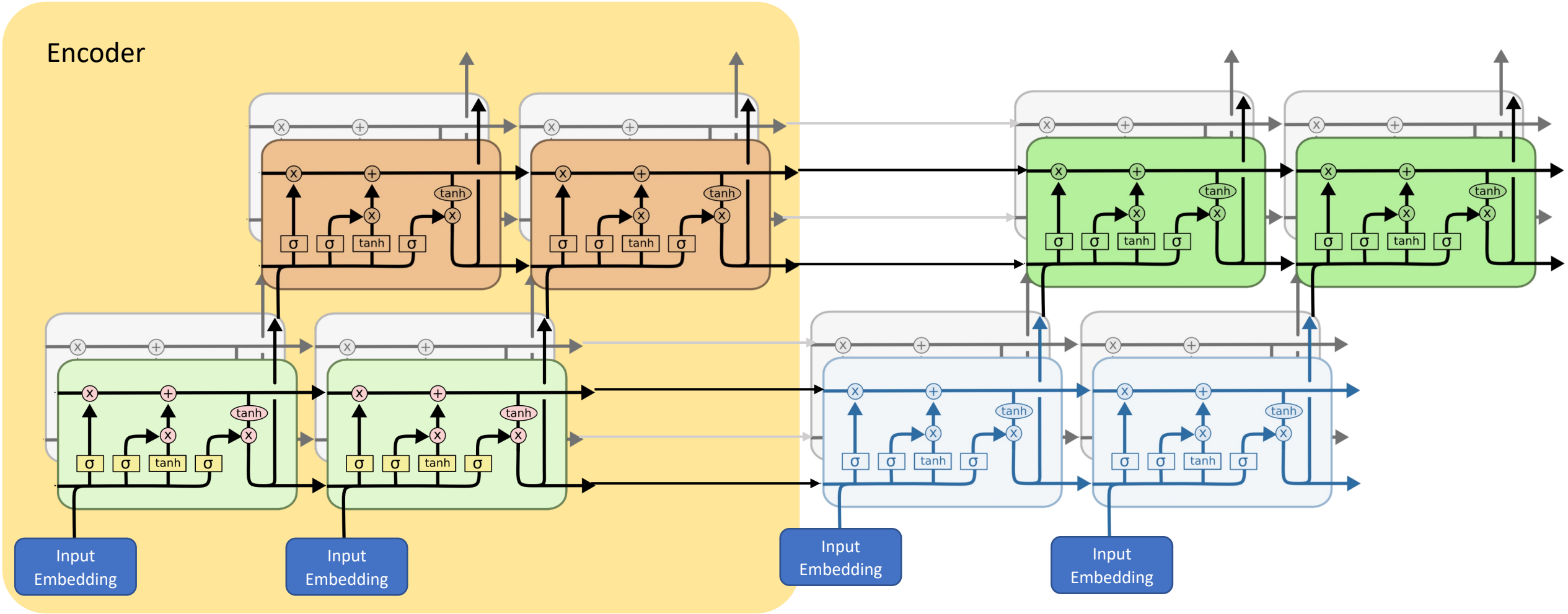


Seq2Seq Encoder-Decoder

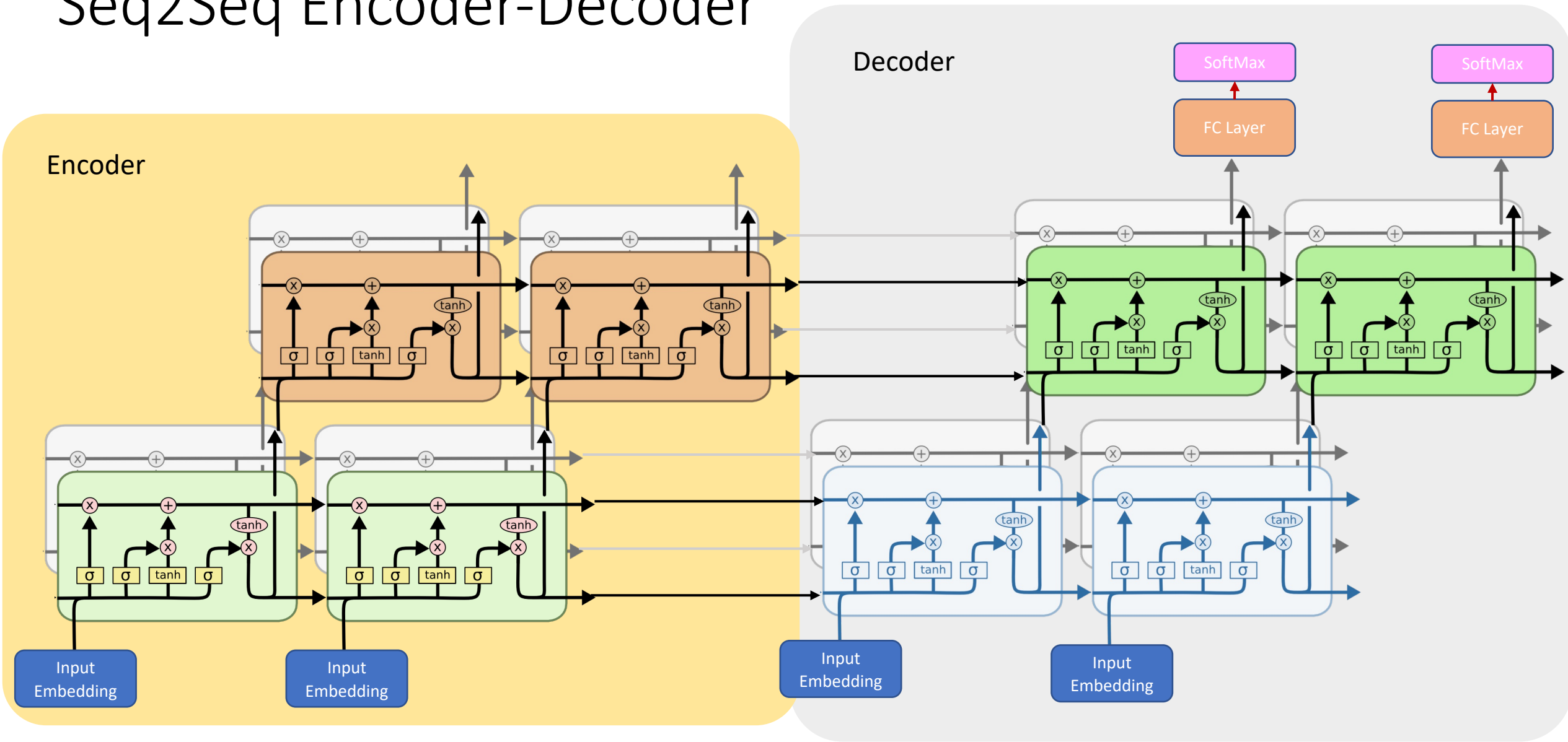


- Add more LSTM cells to get more variables in the network
- The cells in each row have **the same weights** and **biases**

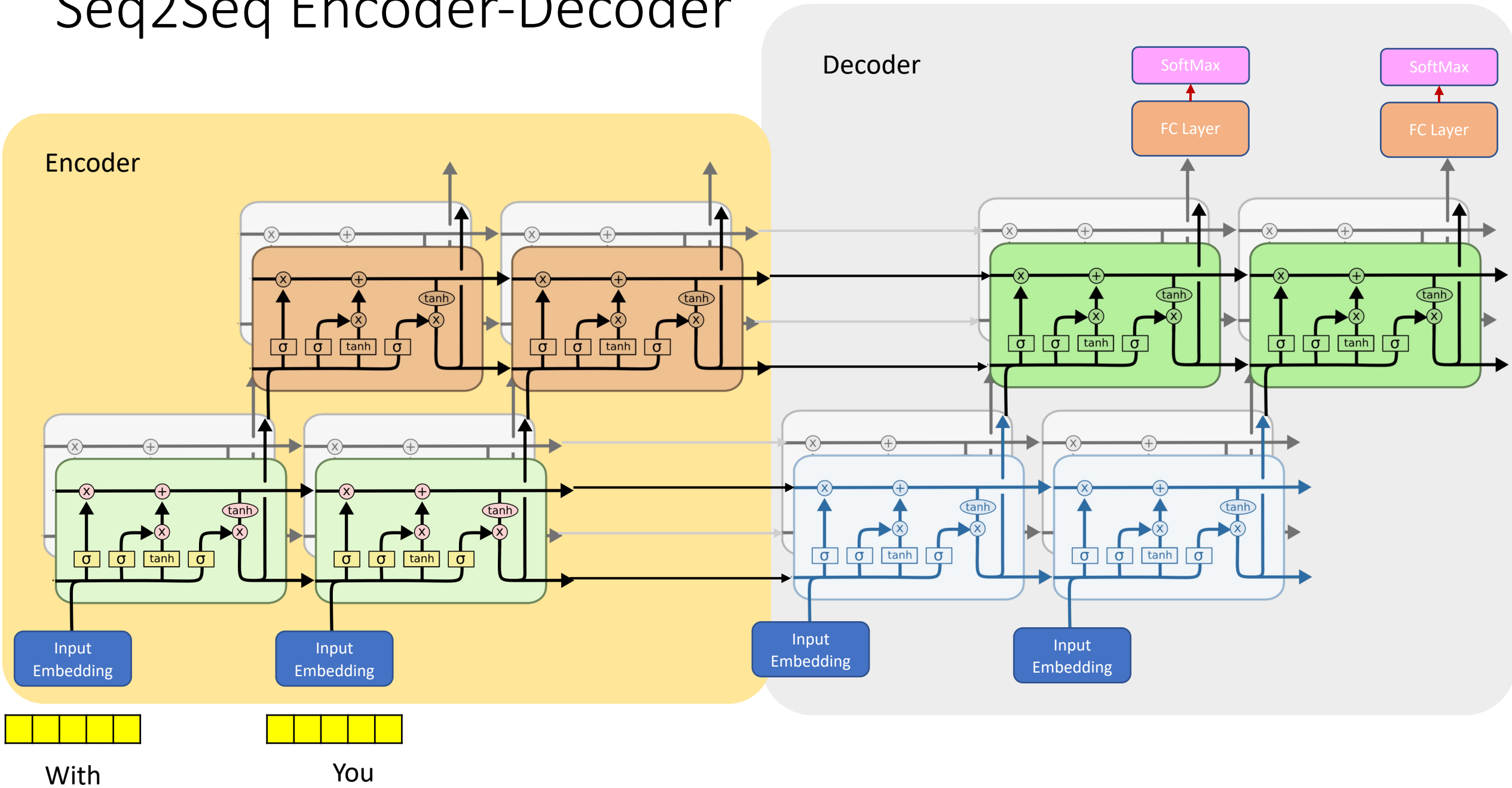
Seq2Seq Encoder-Decoder



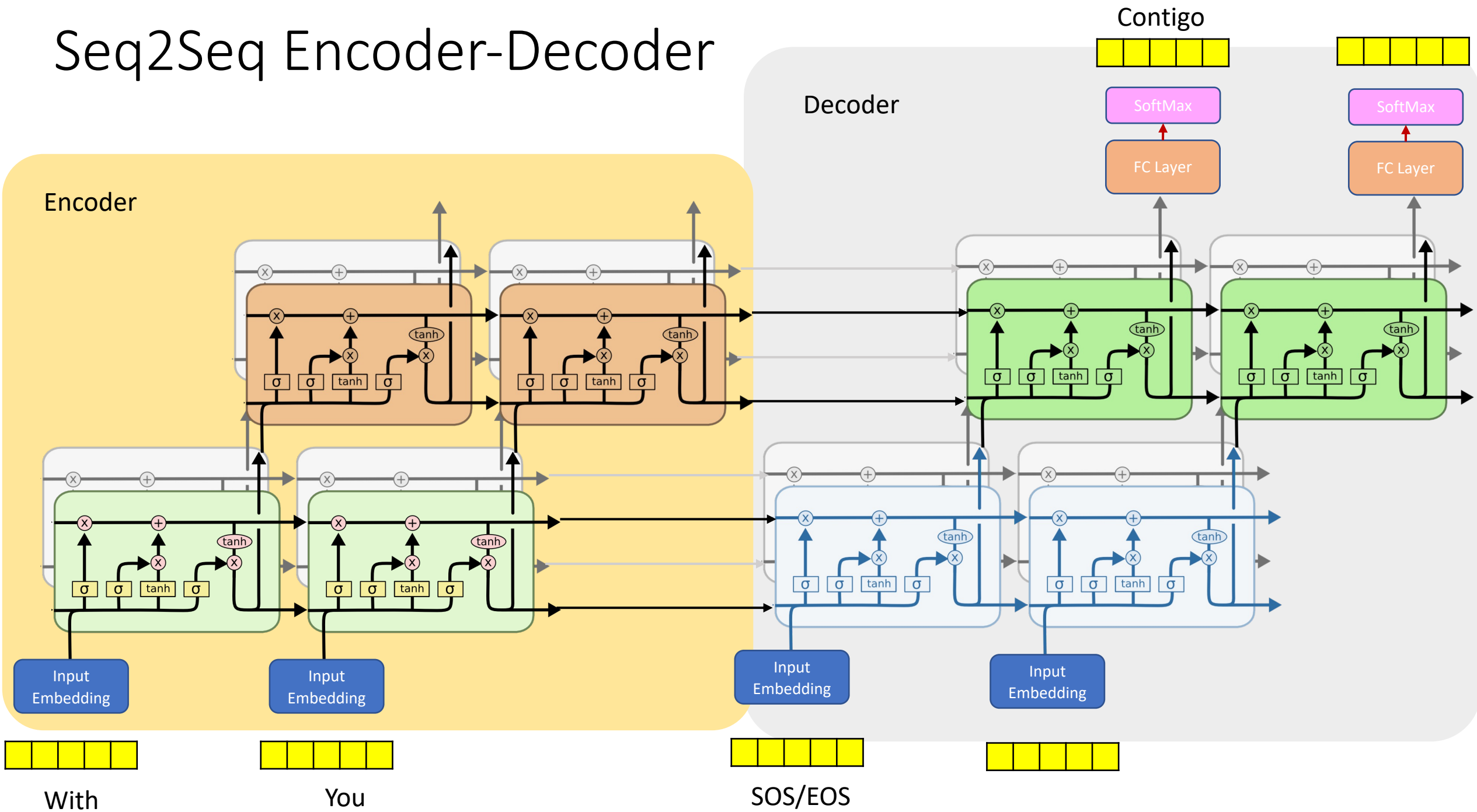
Seq2Seq Encoder-Decoder



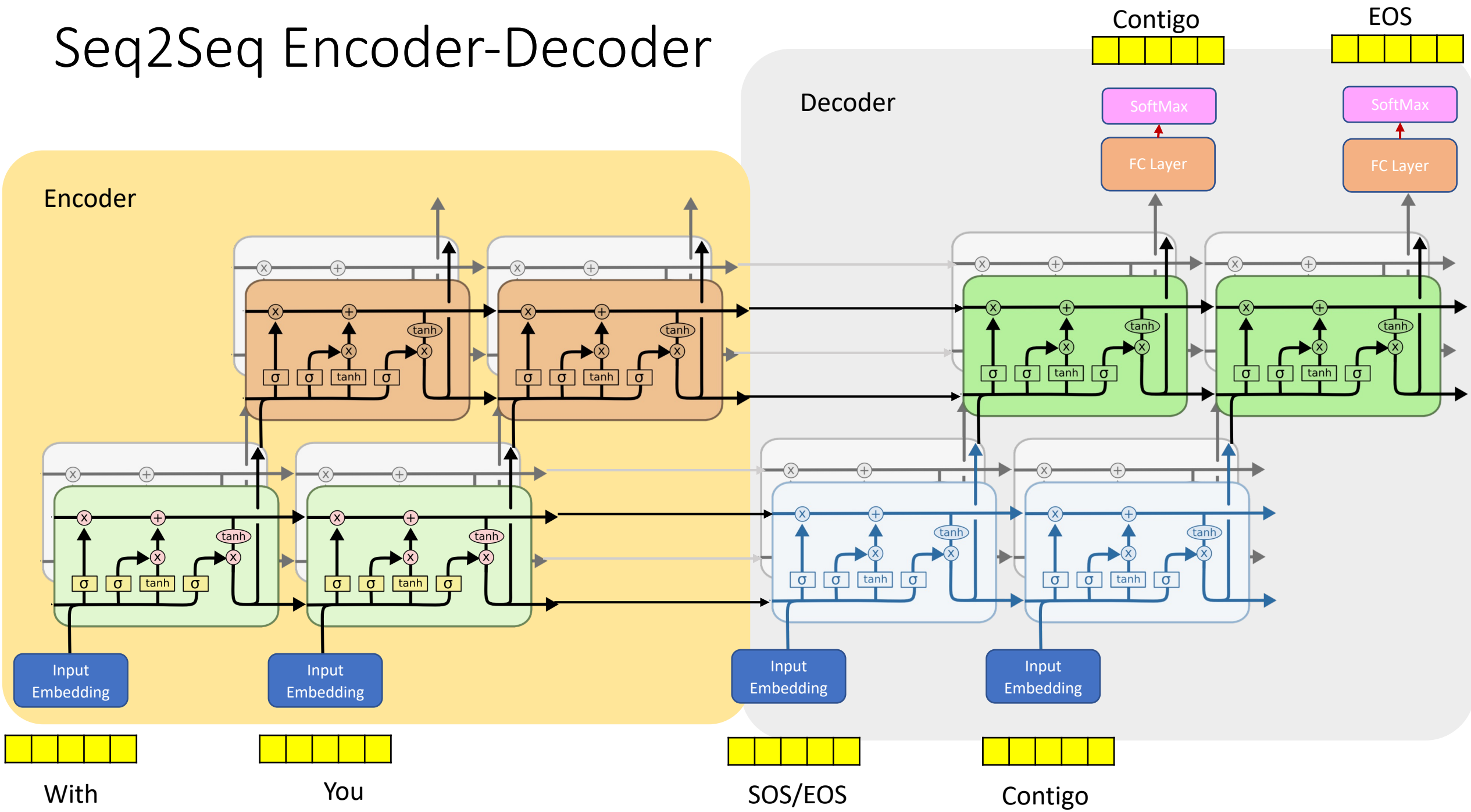
Seq2Seq Encoder-Decoder



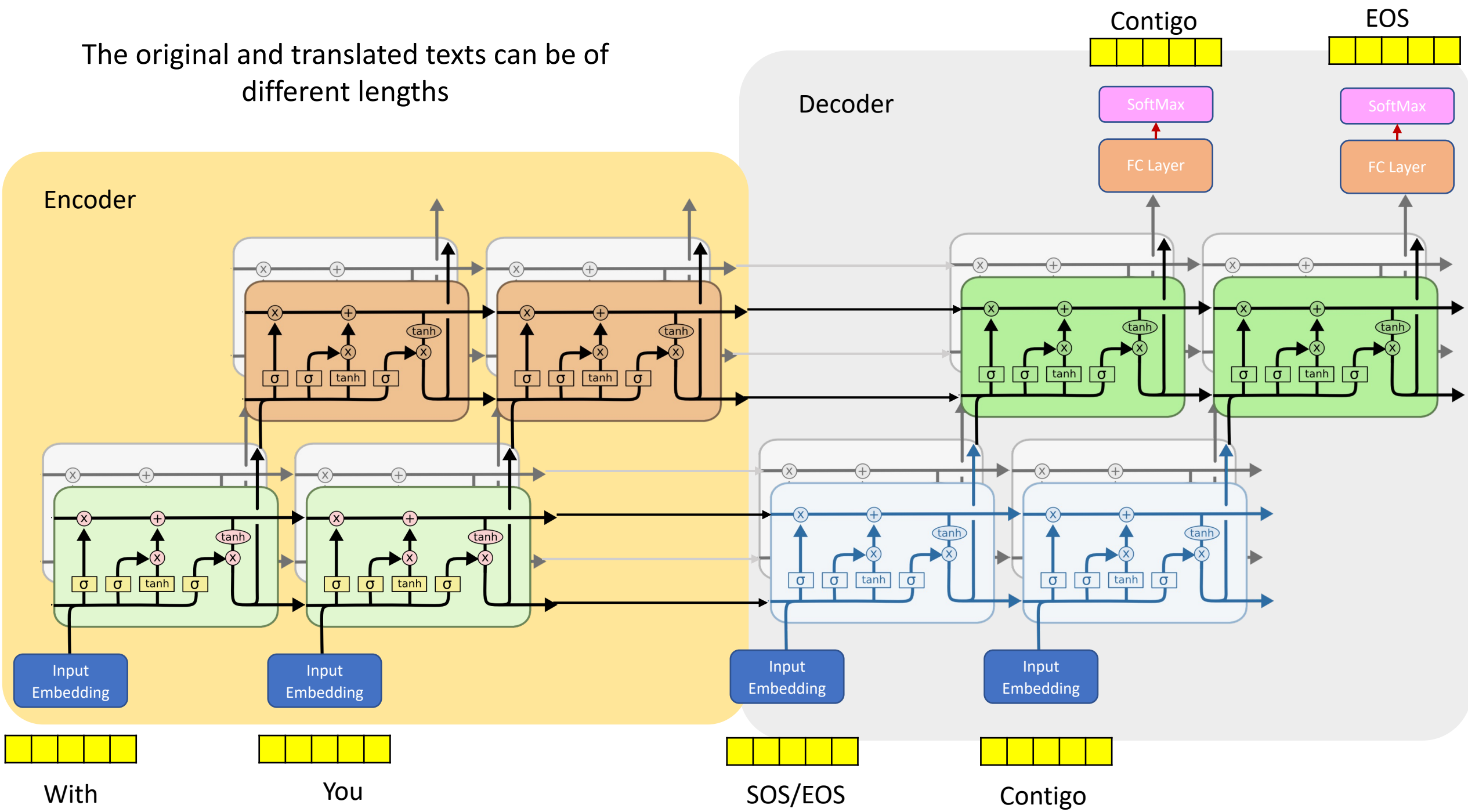
Seq2Seq Encoder-Decoder

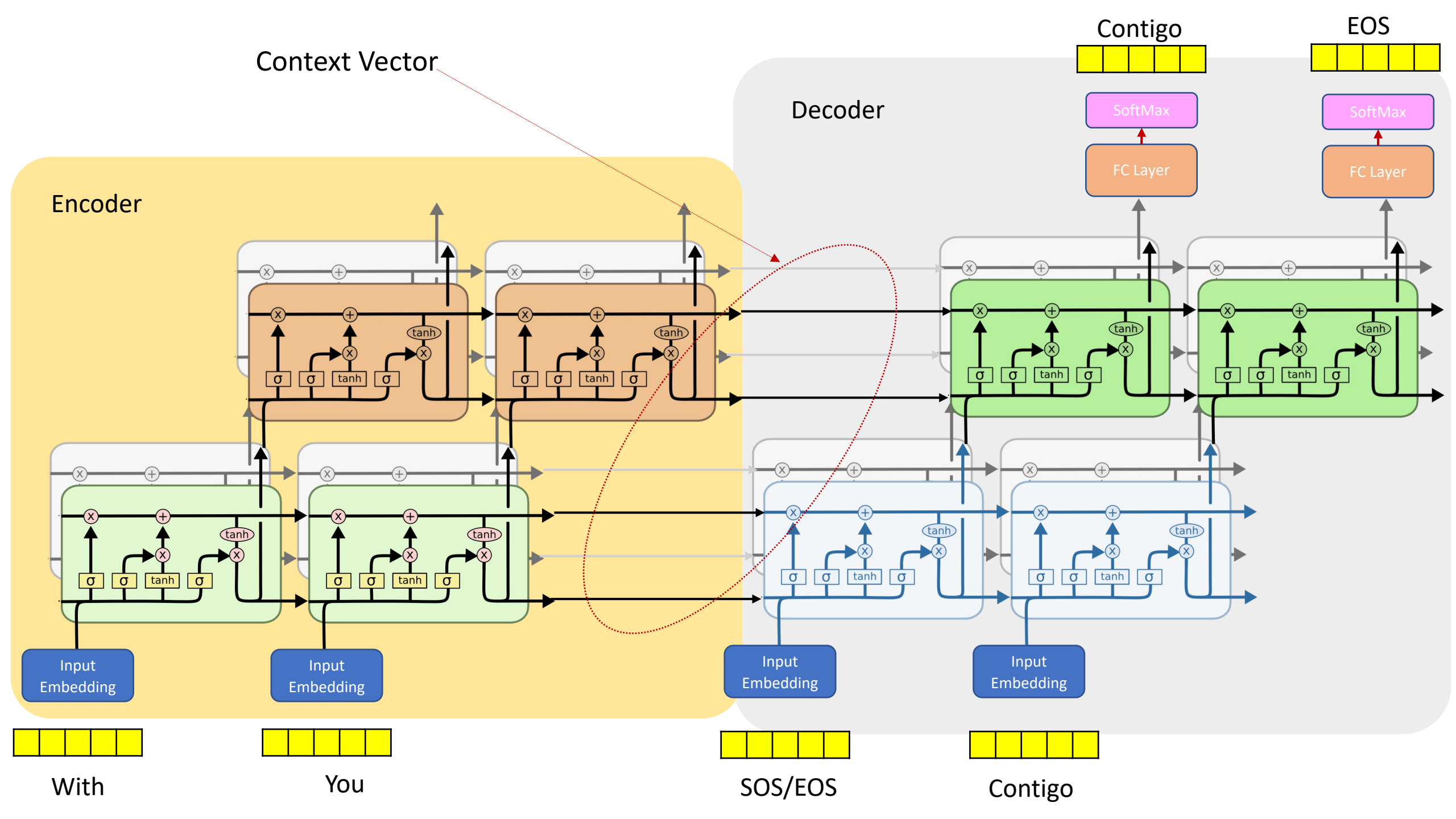


Seq2Seq Encoder-Decoder

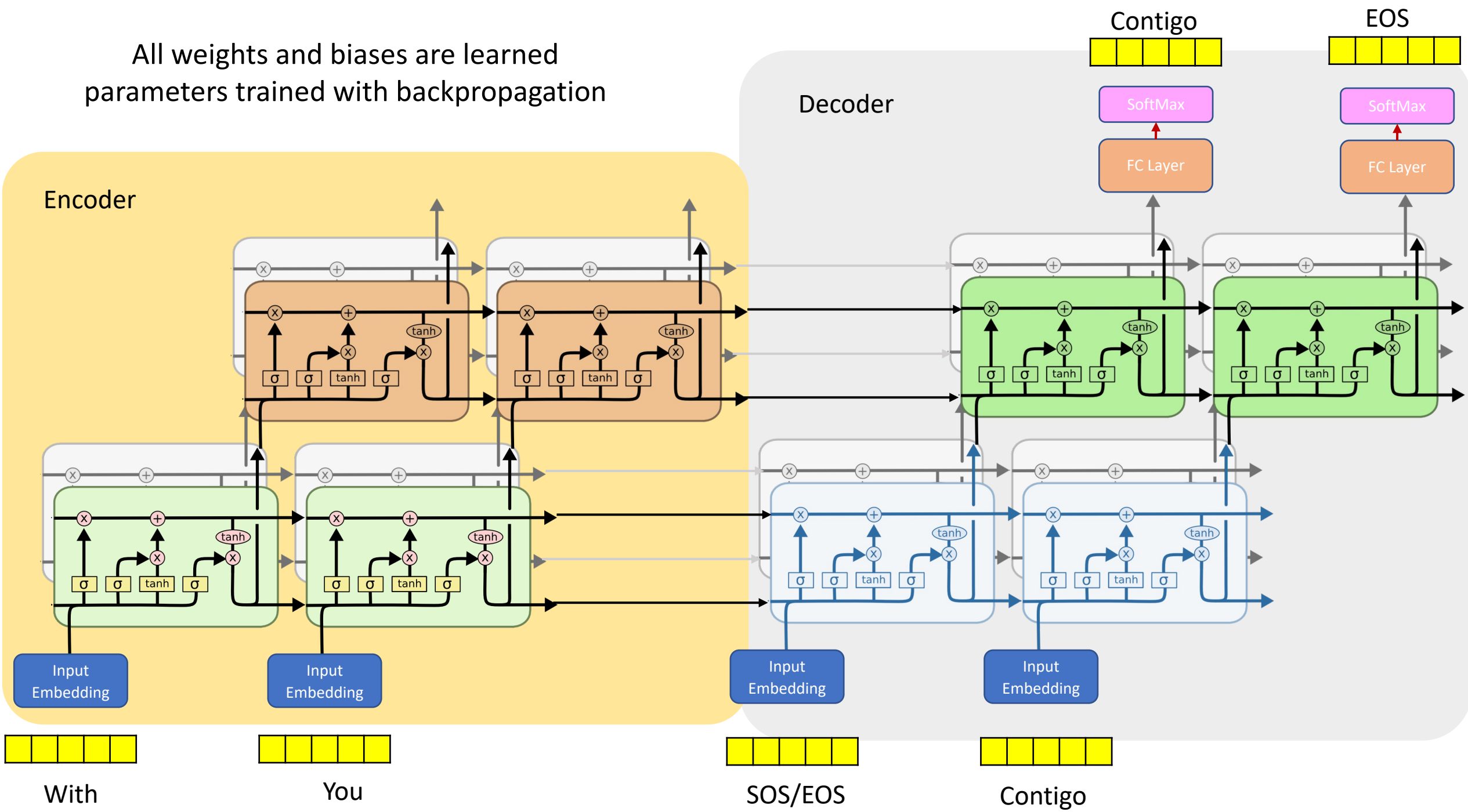


The original and translated texts can be of different lengths

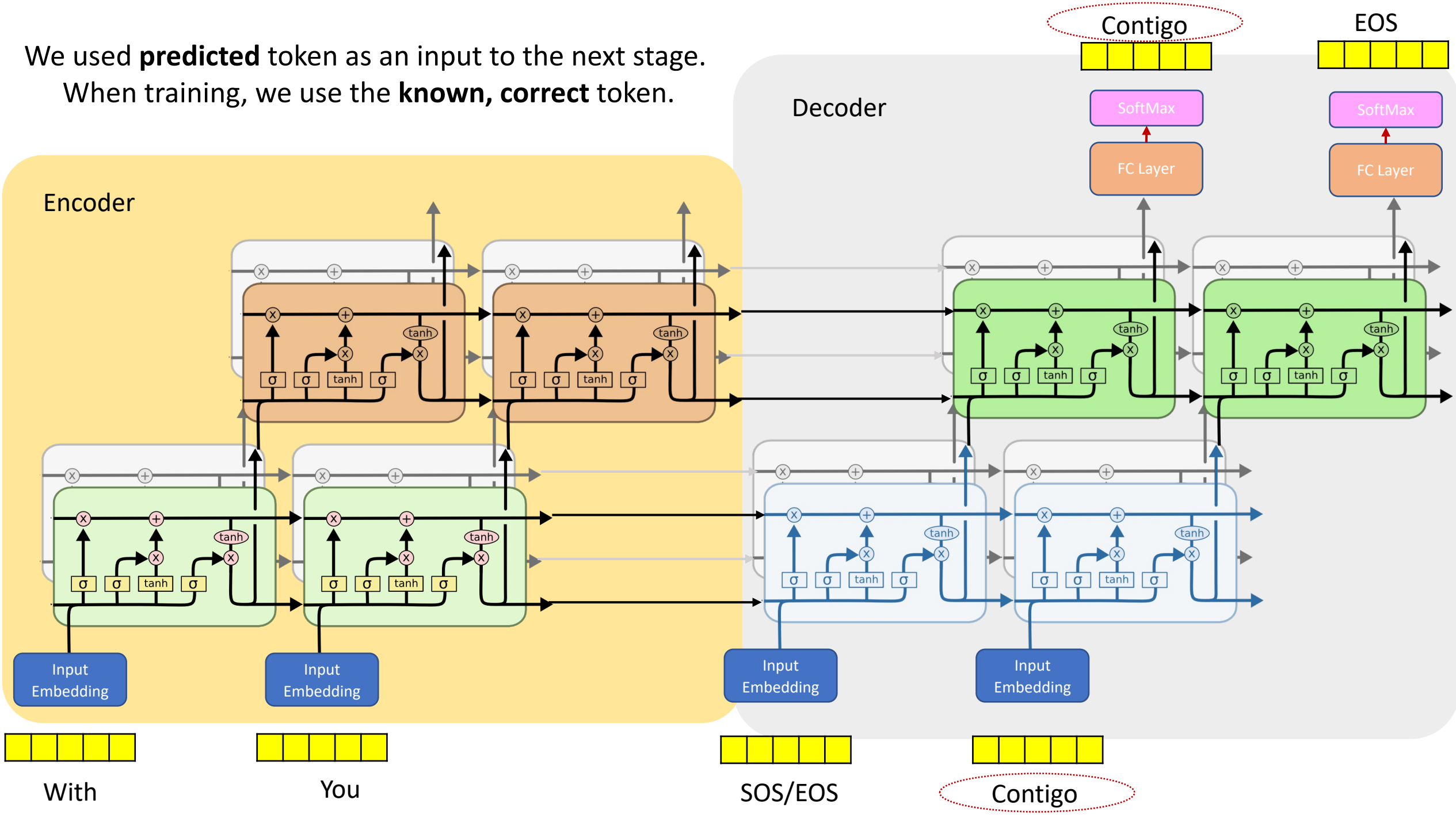



Context Vector
Contigo
EOS
Decoder
Encoder
SoftMax
FC Layer
SoftMax
FC Layer
tanh
σ
Input Embedding
Input Embedding
Input Embedding
Input Embedding
With
You
SOS/EOS
Contigo

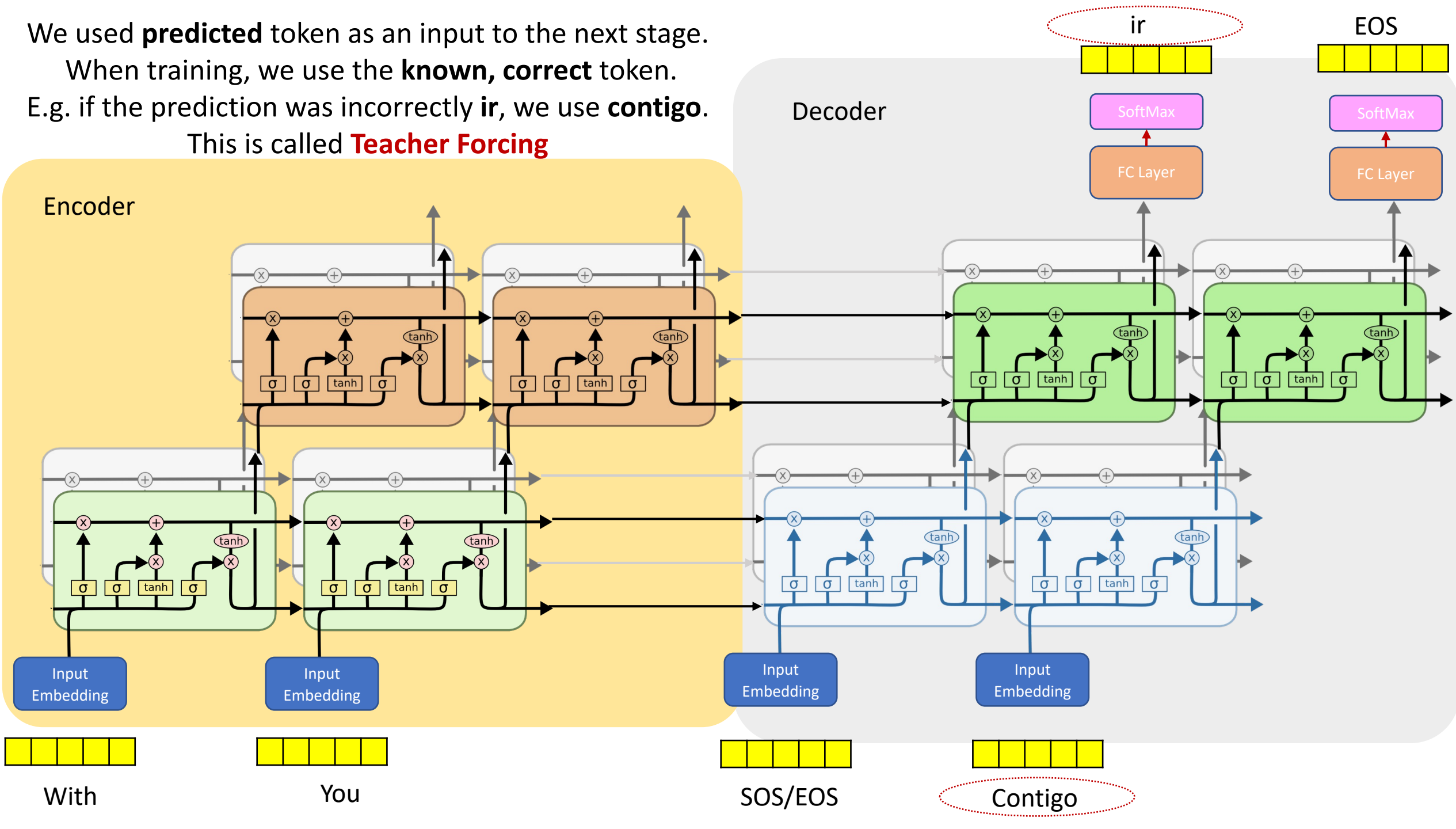
All weights and biases are learned parameters trained with backpropagation



We used **predicted** token as an input to the next stage.
When training, we use the **known, correct** token.



We used **predicted** token as an input to the next stage.
When training, we use the **known, correct** token.
E.g. if the prediction was incorrectly **ir**, we use **contigo**.
This is called **Teacher Forcing**



Variable-Length Inputs in Seq2Seq Models

- **Challenge:** Input and output sequences in real-world scenarios often have varying lengths.
  - Seq2Seq models require fixed-length inputs for efficient processing.
- **Solution: Padding and Masking**
  - **Padding:** Add special padding tokens (e.g., <PAD>) to shorter sequences to match the maximum length.
    - Example: "Hello" becomes "Hello <PAD> <PAD> <PAD> <PAD>" if the max length is 5.
  - **Masking:** Create a mask that indicates which positions in the input/output sequence are actual data and which are padding tokens.
    - The model ignores padding tokens during calculations.
    - Tells the Seq2Seq model which parts of the padded sequence are actual data and which parts are padding.

behavioural patterns between 5 rooms.

Sequence Number	Description	Group
1	Walking in all 5 rooms	1
2	Walking in the corridor, spending almost equal times in rooms 3 and 5	2
3	Similar to 2 but doing more activities in room 3	2
4	Spending time in room 3 with a short period in room 4	2
5	Doing activities between rooms 2, 3 and 5	2
6	Doing activities in all five rooms, mostly walking in room 2.	3
7	Activities in all rooms, although spending more time in room2.	3
8	Activities in all five rooms.	3
9	Moving between rooms 2 and 4 and interact-	4

Variable length input

Applications of LSTM

- **Natural Language Processing (NLP):**
  - Language Modeling: Predicting the next word in a sequence.
  - Text Generation: Creating new text, like articles or poems.
  - Machine Translation: Converting text from one language to another.
  - Sentiment Analysis: Determining the emotional tone of text.
  - Question Answering: Providing answers to questions based on a given context.
- **Speech Recognition:** Converting spoken language into text.
- **Time Series Analysis:**
  - Stock Price Prediction: Forecasting future stock values.
  - Weather Forecasting: Predicting future weather conditions.
  - Sales Forecasting: Estimating future product sales.
- **Image Captioning:** Generating textual descriptions of images.
- **Video Analysis:** Understanding and describing the content of videos.

Products Using LSTMs

- **Google's Products:**
  - Google Translate: For machine translation.
  - Google Assistant: For voice recognition and natural language understanding.
  - YouTube: For video captioning and recommendations.
- **Apple's Products:**
  - Siri: For voice recognition and natural language understanding.
  - QuickType: For predictive text input on iOS devices.
- **Amazon's Products:**
  - Alexa: For voice recognition and natural language understanding.
- **Key Advantage of LSTMs:**
  - LSTMs excel at processing sequential data due to their ability to handle long-term dependencies, making them suitable for tasks where the context of previous elements is crucial.
- **The shift towards LLMs:**
  - While LSTMs have been widely used in the past, Google and other companies are increasingly adopting Large Language Models (LLMs) like BERT, PaLM, and Gemini for many NLP tasks.
  - However, LSTMs are still used in specific applications, especially where computational efficiency and resource constraints are important.