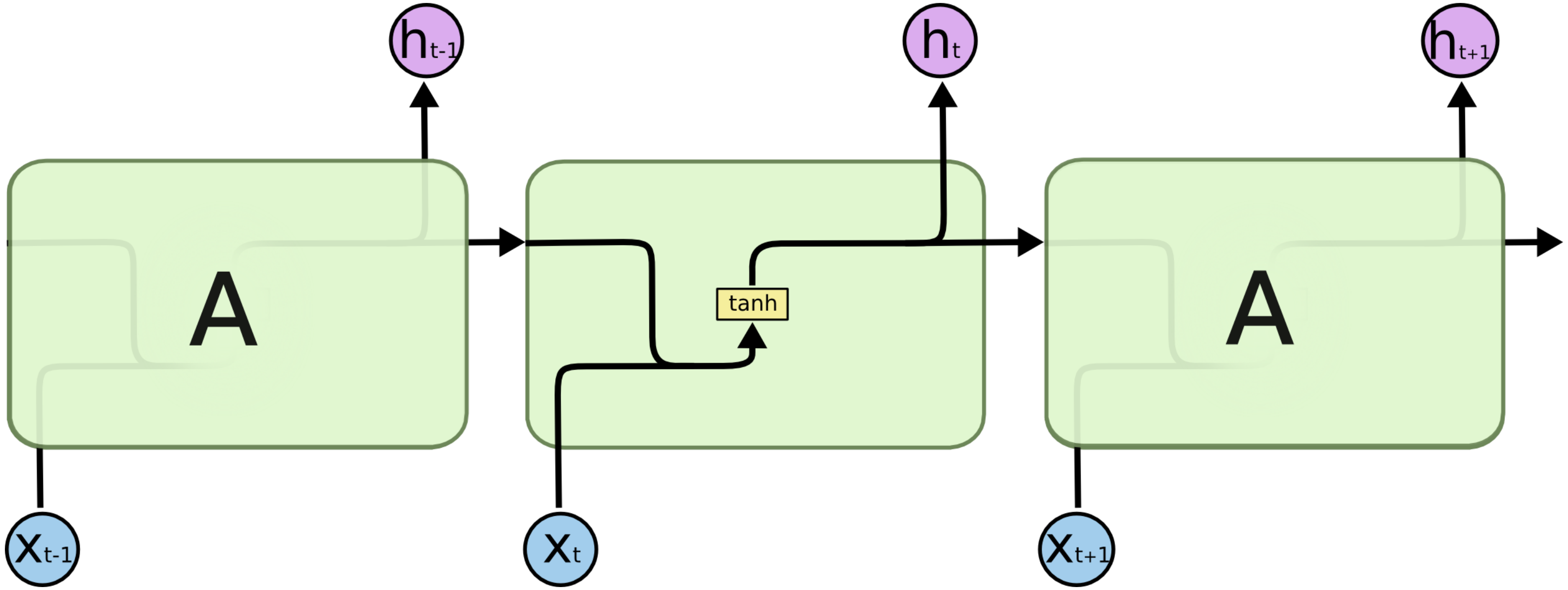
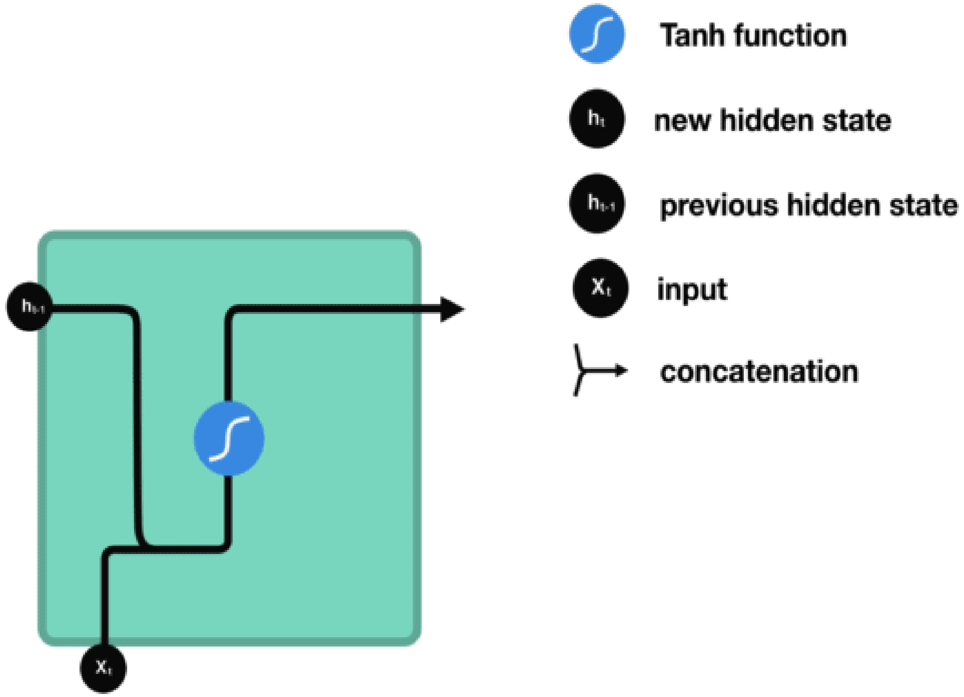
LSTM - Long Short-Term Memory

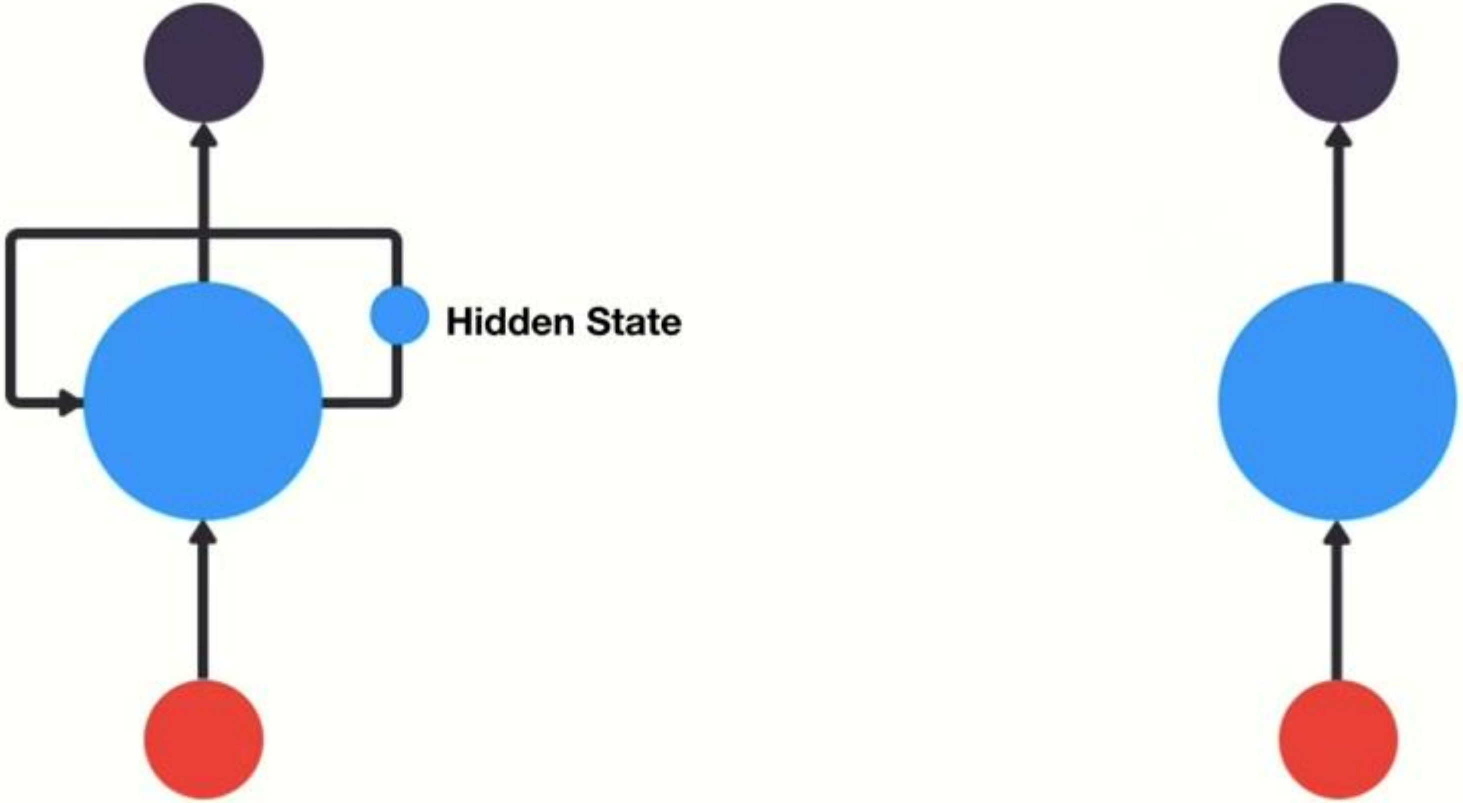
- **Recap: RNN**
  - All recurrent neural networks have the form of a chain of repeating modules of neural network (Recurrent Units).
  - In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

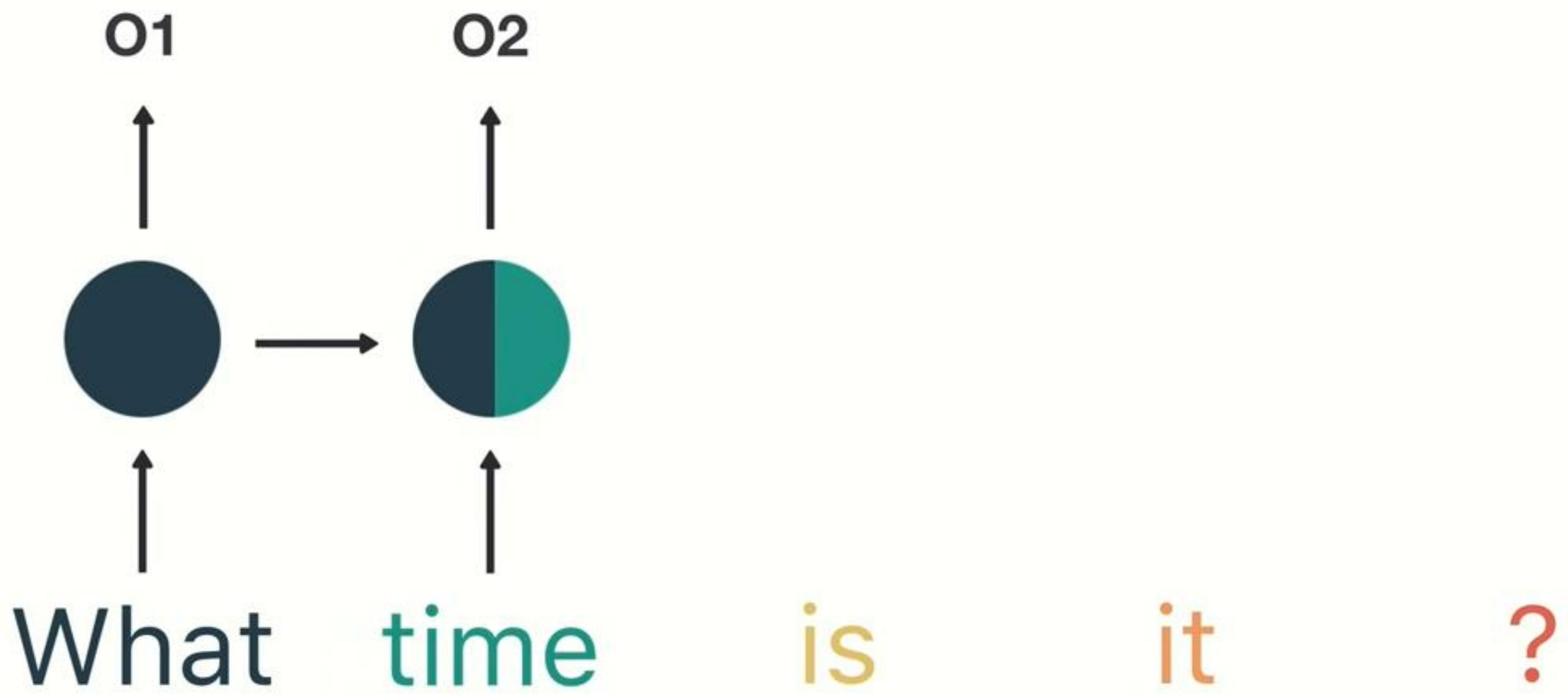


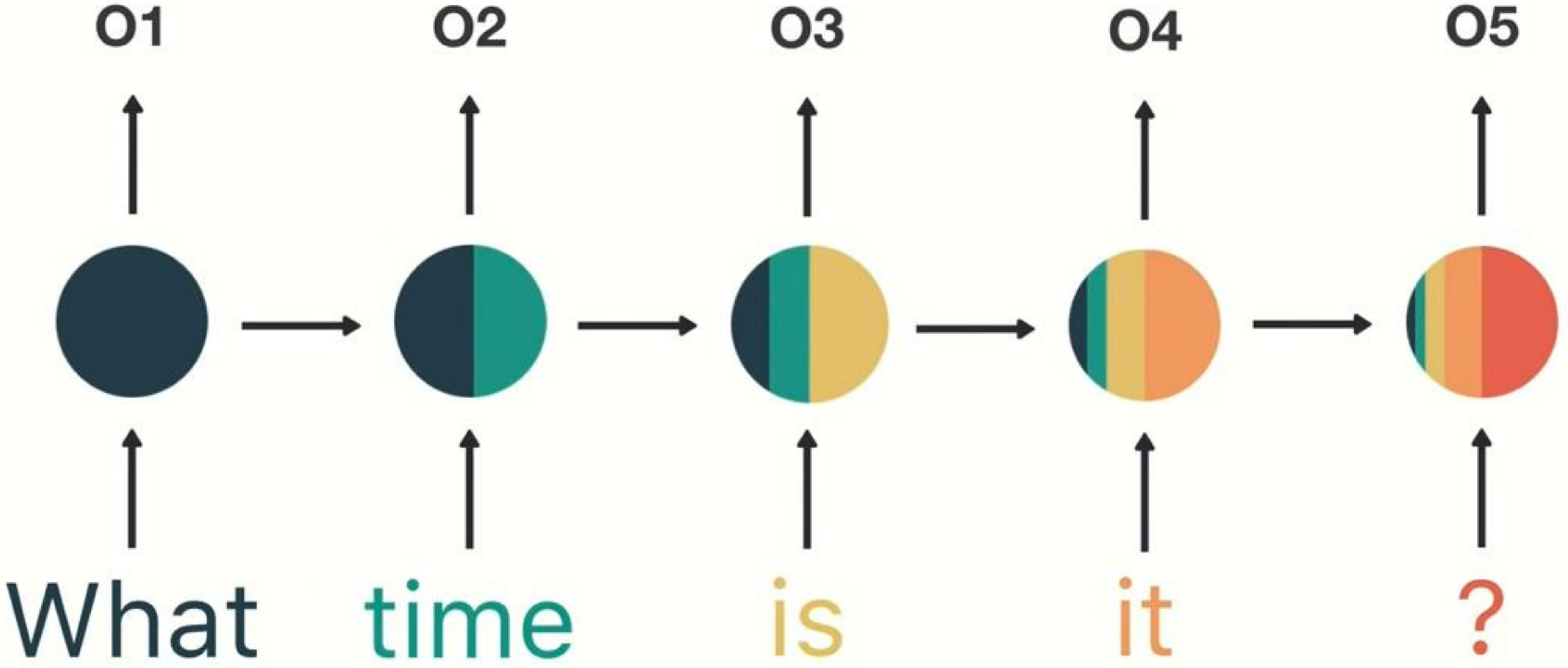


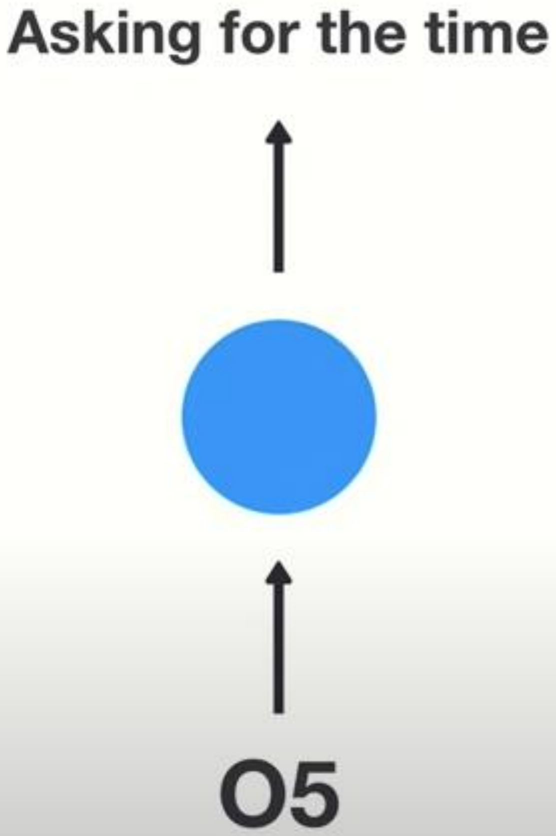
- **Tanh activation**
  - The tanh activation is used to help regulate the values flowing through the network. The tanh function squishes values to always be between -1 and 1.

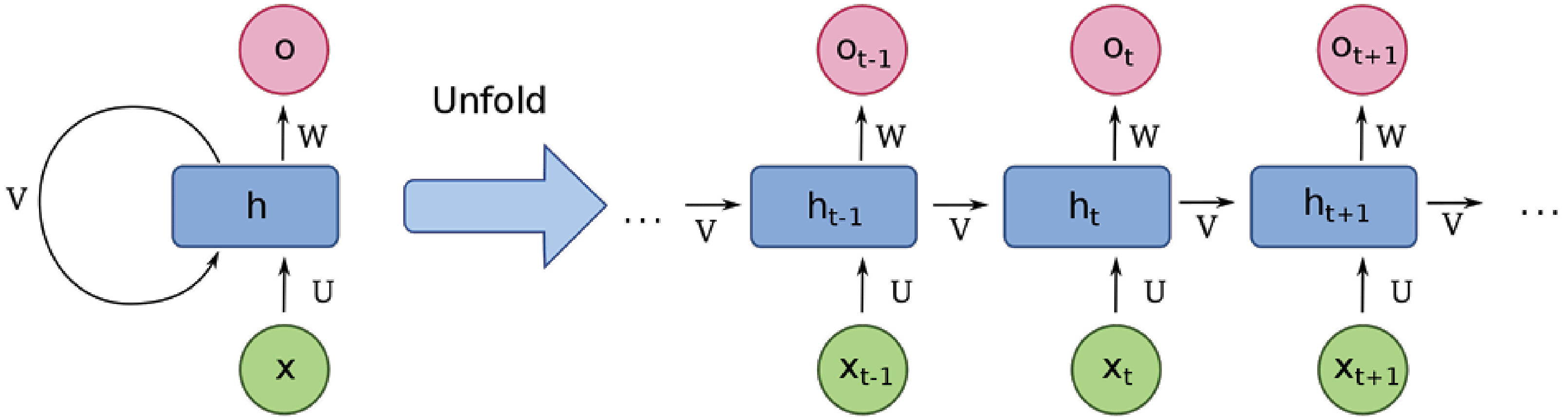
What time is it?




O1
O2
What
time
is
it
?


O1
O2
O3
O4
O5
What
time
is
it
?


Asking for the time
O5

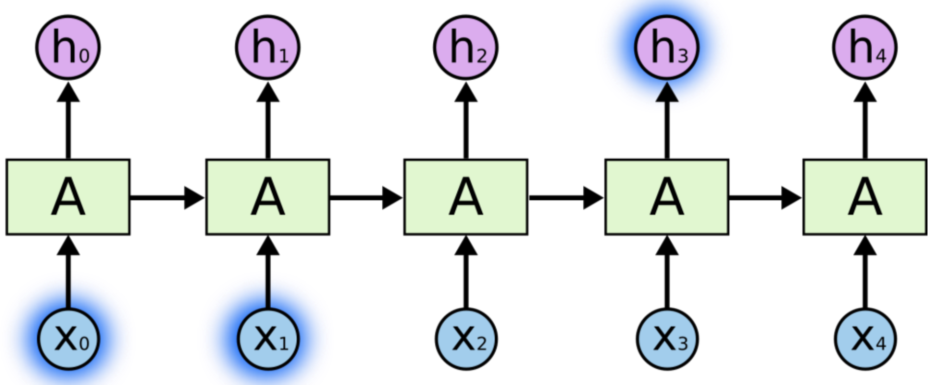


$$h_t = f(\overset{\color{red}{U}}{W_x} x_t + \overset{\color{red}{V}}{W_h} h_{t-1} + b)$$

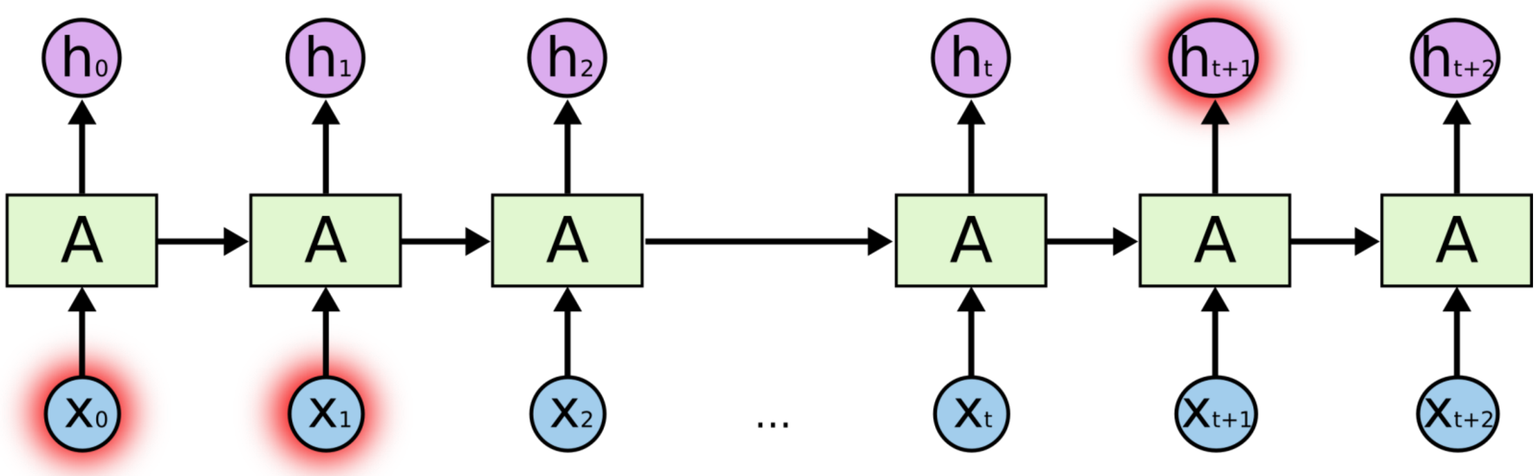
o_t or $y_t = g(W_y h_t + b_y)$

The Problem of Long-Term Dependencies

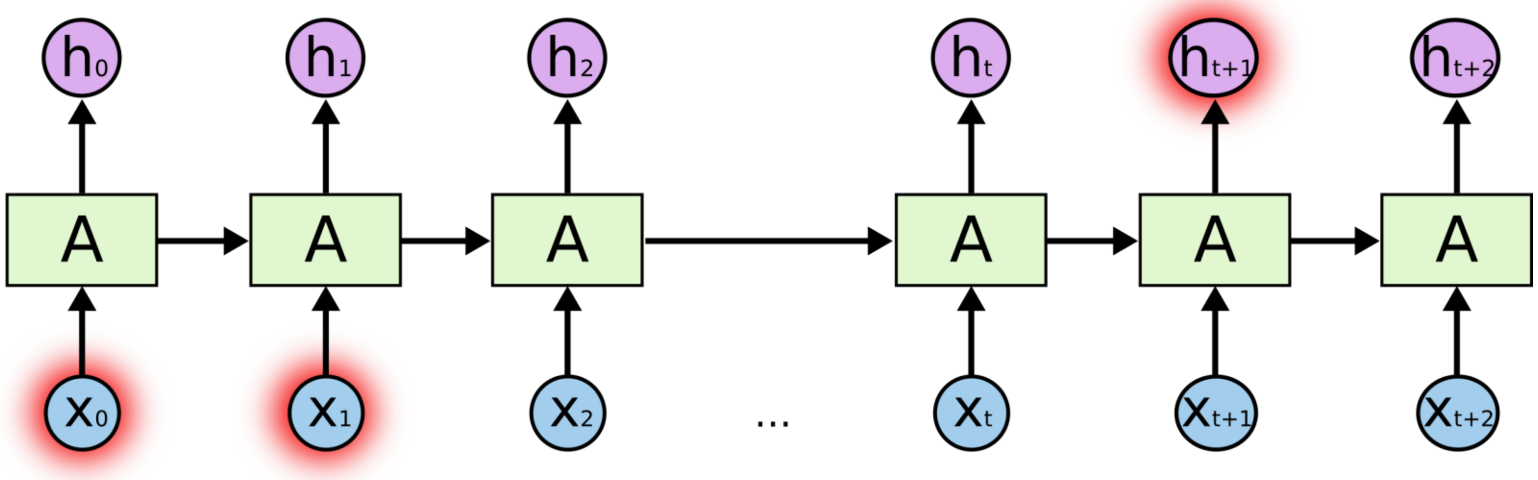
- One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task
- It is useful when we only need to look at recent information to perform the present task.
- For example, consider a language model trying to predict the next word based on the previous ones.
- If we are trying to predict the last word in "the clouds are in the *sky*," we don't need any further context – it's pretty obvious the next word is going to be sky.
- In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information.



- There are also cases where we need more context. Consider trying to predict the last word in the text:
    - "I grew up in France… I speak fluent …………."
- Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back.
- It's entirely possible for the gap between the relevant information and the point where it is needed to become very large.
- Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.

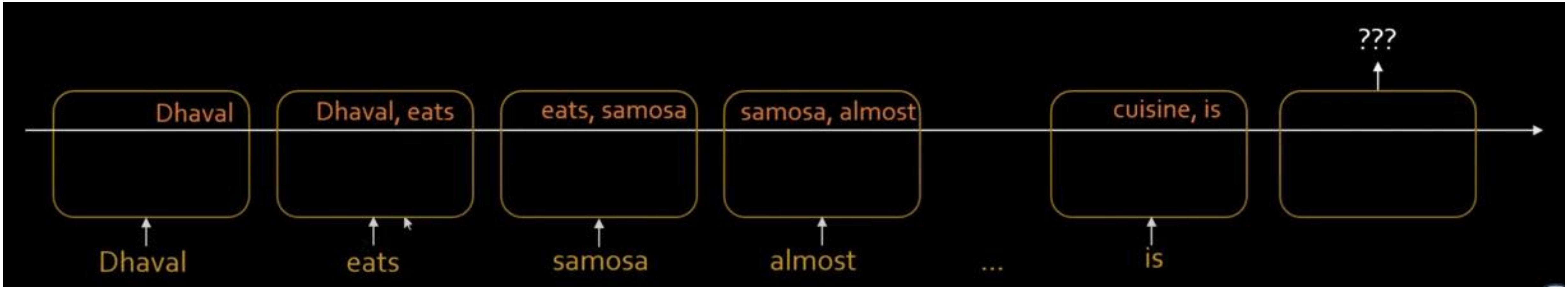


- There are also cases where we need more context. Consider trying to predict the last word in the text:
  - "I grew up in France… I speak fluent *French*."
- Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back.
- It's entirely possible for the gap between the relevant information and the point where it is needed to become very large.
- Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.



Dhaval eats samosa almost everyday, it shouldn’t be hard to guess that his favorite cuisine is …

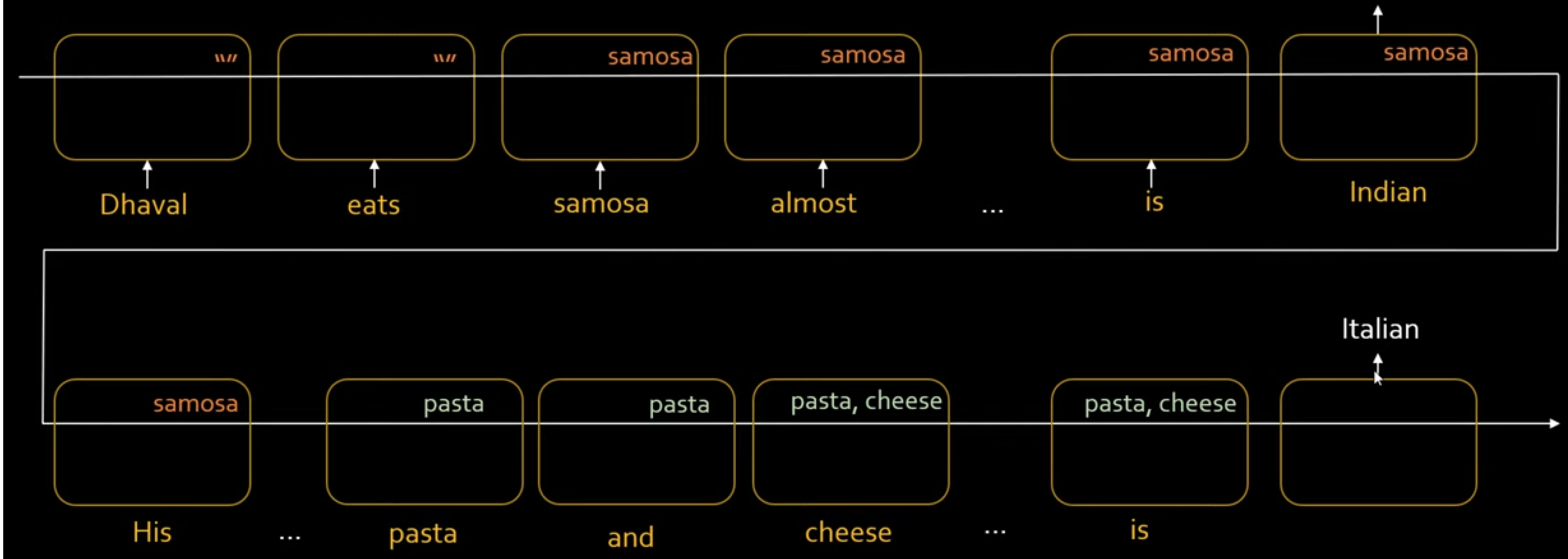
Dhaval eats samosa almost everyday, it shouldn't be hard to guess that his favorite cuisine is Indian



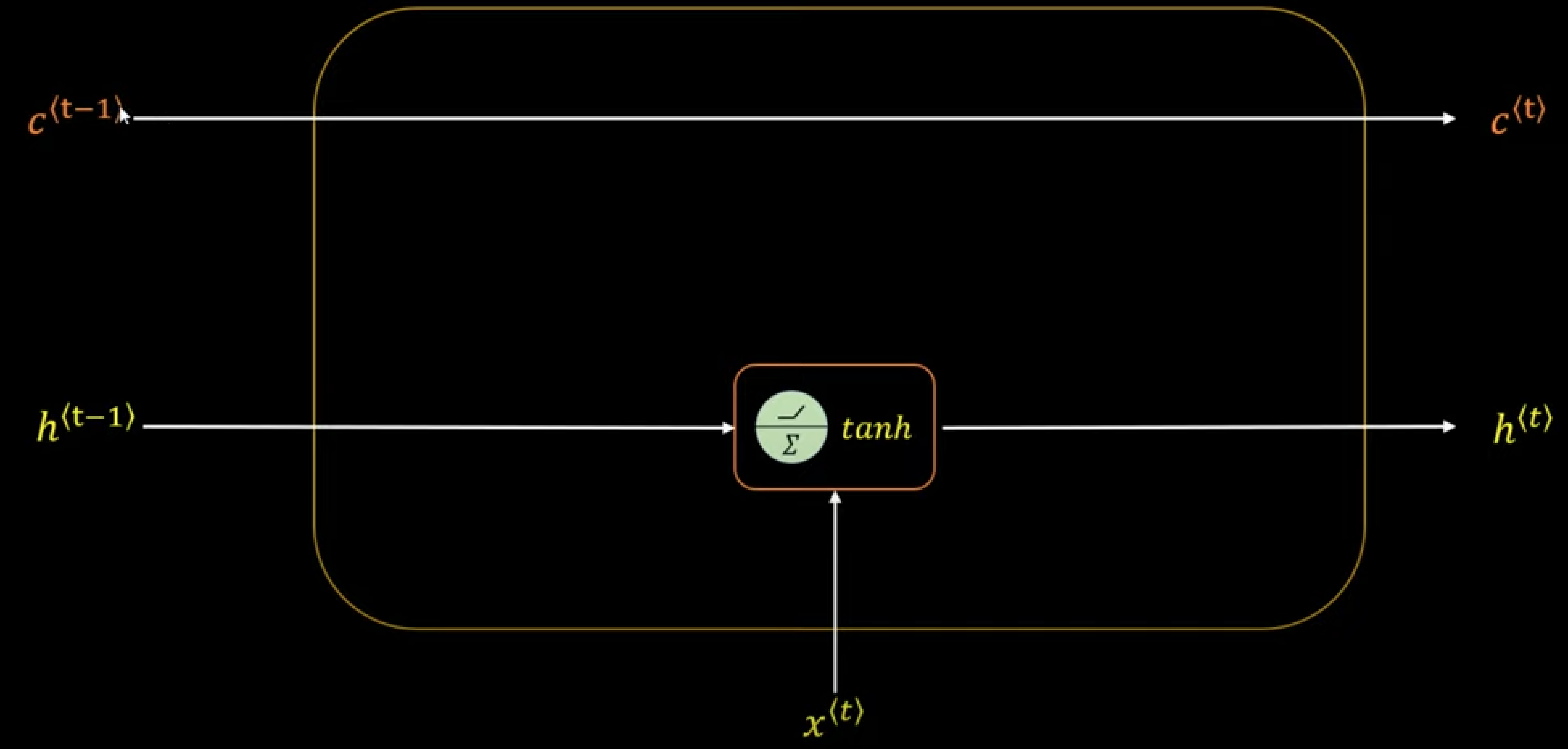
A representative example only

Dhaval eats samosa almost everyday, it shouldn't be hard to guess that his favorite cuisine is Indian. His brother Bhavin however is a lover of pasta and cheese that means Bhavin's favorite cuisine is

Dhaval eats samosa almost everyday, it shouldn't be hard to guess that his favorite cuisine is Indian. His brother Bhavin however is a lover of pasta and cheese that means Bhavin's favorite cuisine is Italian



Short term memory and long term memory

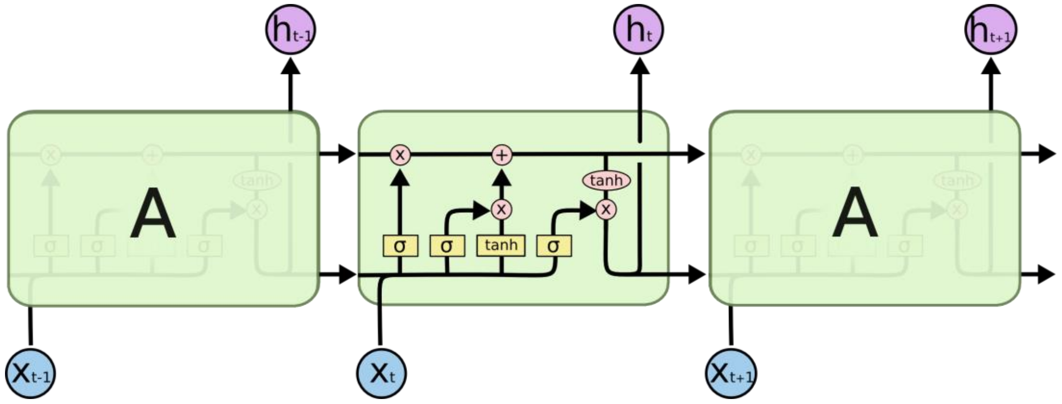


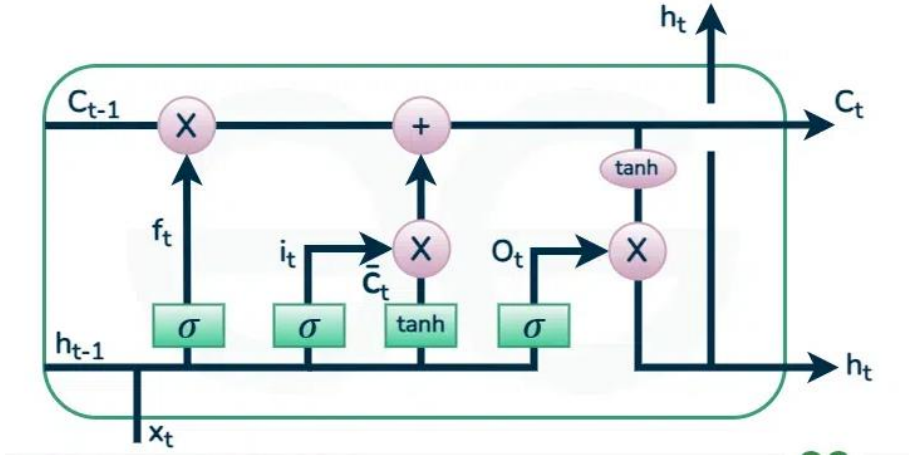
LSTM

- Long Short Term Memory networks – usually called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies.
- Remembering information for long periods of time is practically their default behavior.

LSTM architecture

- Long Short Term Memory networks, LSTMs, are a special kind of RNN, capable of learning long-term dependencies. They were introduced by [Hochreiter & Schmidhuber (1997)]
- LSTMs architecture has a chain structure similar to RNN, but the repeating module has a different structure. It contains four neural networks and different memory blocks called cells (Long term memory and short term memory).
- Information is retained by the cells and the memory manipulations are done by the gates.



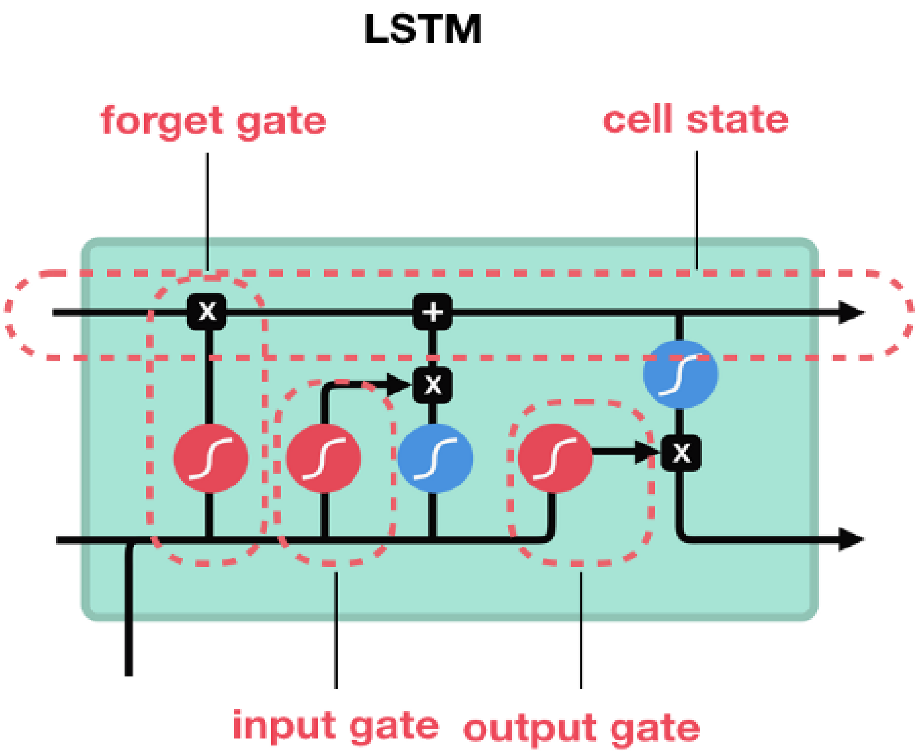


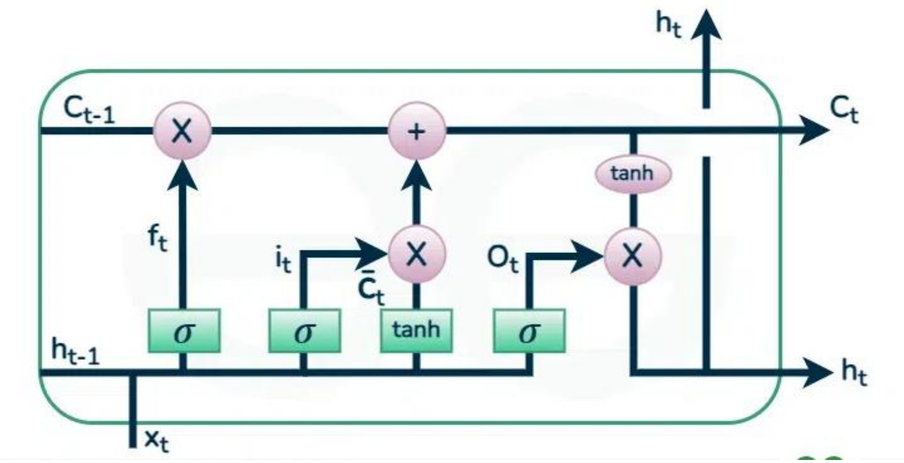
LSTM Cell Architecture

- The core of the LSTM is the **memory cell** that helps store and update information over time. Each LSTM cell contains the following components:
- **Cell State** (C_t), **Long term memory**: This is where the memory of the network is stored. It runs through the entire sequence with minimal linear interactions, making it easy to carry long-term information.
- **Hidden State** (h_t), **Short term memory**: The hidden state is the output of the LSTM unit at any time step, capturing both current and historical context.
- **Gates**: LSTMs have three types of gates that control the flow of information through the network:
  - **Forget Gate** (f_t): Controls how much of the previous cell state should be forgotten.
  - **Input Gate** (i_t): Controls how much new information is written to the cell state.
  - **Output Gate** (o_t): Controls the final output of the cell at each time step.

- **The Core Idea Behind LSTMs**

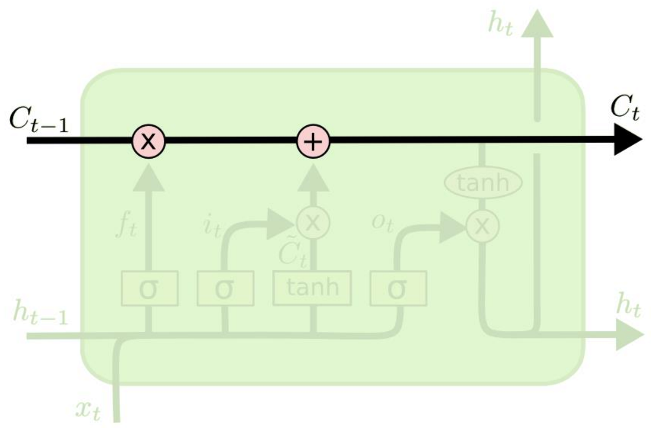
[Illustrated Guide to LSTM's and GRU's: A step by step explanation (youtube.com)]()



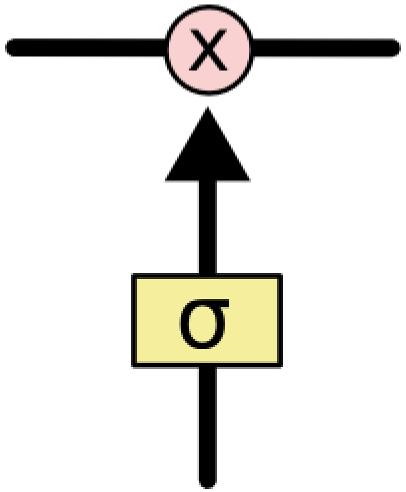


The Core Idea Behind LSTMs

- The key to LSTMs is the cell state, the horizontal line running through the top of the diagram. This is where the long term memory of the network is stored. It runs through the entire sequence with minimal linear interactions, making it easy to carry long-term information.
- The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions.
- The LSTM has the ability to remove or add information to the cell state, carefully regulated by structures called gates



- Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation.
- The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means “let nothing through,” while a value of one means “let everything through!”
- An LSTM has three of these gates, to protect and control the cell state.



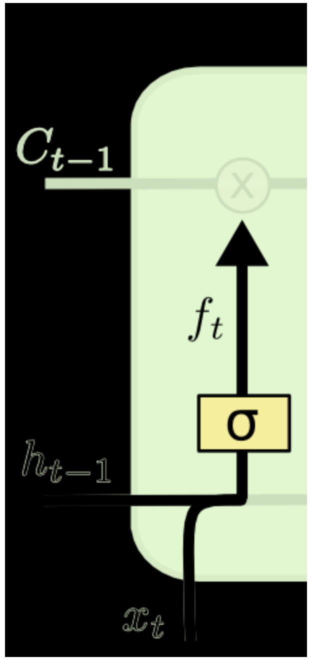
- **Forget Gate**
  - This gate helps the LSTM retain relevant long-term information while discarding irrelevant or outdated information, making it effective for long-term dependencies.
  - It controls which parts of the previous cell state Ct−1 should be carried forward to the next time step.
  - It looks at ht−1 and x_t, and outputs a number between 0 and 1 for each number in the cell state Ct−1. A 1 represents "completely keep this" while a 0 represents "completely get rid of this."

$$f_t = \sigma(W_f^x \cdot x_t + W_f^h \cdot h_{t-1} + b_f)$$

Where:

- f_t is the forget gate output, a vector of values between 0 and 1 that determines how much of the previous cell state C_{t-1} should be forgotten or retained.
- W_f^x is the weight matrix for the current input x_t.
- W_f^h is the weight matrix for the previous hidden state h_{t-1}.
- x_t is the input at time step t.
- h_{t-1} is the hidden state from the previous time step.
- b_f is the bias term for the forget gate.
- σ is the sigmoid activation function, which outputs values between 0 and 1.

Ft
0.5
0.7
0.1
0.9
0.0



The forget gate output f_t is a vector of values between 0 and 1. Each value in f_t corresponds to a part of the cell state C_{t-1}. A value close to 1 means "keep this part of the memory," while a value close to 0 means "forget this part."

Effect of forget gate on Cell State:

- The forget gate output f_t is multiplied element-wise with the previous cell state Ct−1.

$$C_{t-1} \odot f_t = 0 \quad \text{...if } f_t = 0 \text{ (forget everything)}$$

$$C_{t-1} \odot f_t = C_{t-1} \quad \text{...if } f_t = 1 \text{ (forget nothing)}$$

Ft		**Ct-1**		
0.5		0.8		0.40
0.7	$\odot$	1.0	=	0.7
0.1		2.0		0.2
0.9		0.9		0.81
0.0		0.8		**0.0**

This operation selectively forgets parts of the previous cell state. Only the parts of C_{t-1} where f_t is close to 1 are retained, while the parts where f_t is close to 0 are forgotten.

- **Forget gate function**

  - Eg: Bob is a nice person. Dan on the other hand is evil.

  - As we move from the first sentence to the second sentence, our network should realize that we are no more talking about Bob.
  - Now our subject is Dan.
  - Here, the Forget gate of the network allows it to forget about Bob.

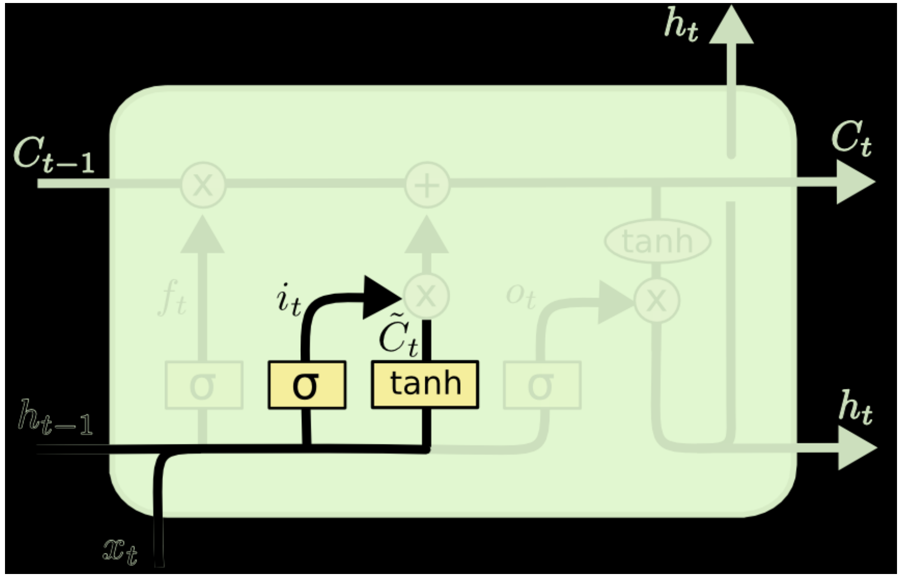
$$C_{t-1} * f_t = 0 \quad \text{...if } f_t = 0 \text{ (forget everything)}$$

$$C_{t-1} * f_t = C_{t-1} \quad \text{...if } f_t = 1 \text{ (forget nothing)}$$

- **Input Gate**

  The **input gate** in an **LSTM (Long Short-Term Memory)** controls how much of the new information from the current input x_t and the previous hidden state h_{t-1} should be **added** to the cell state C_t. In other words, the input gate decides what new information will be stored in the cell state after filtering out irrelevant parts.

- The input gate has two main components:
  1. **Candidate cell state**: Proposes new information to be added to the cell state.
  2. **Input gate activation**: Determines how much new information will be allowed into the cell state.



Candidate Cell State:

$$\tilde{C}_t = \tanh(W_C^x \cdot x_t + W_C^h \cdot h_{t-1} + b_C)$$

Where:

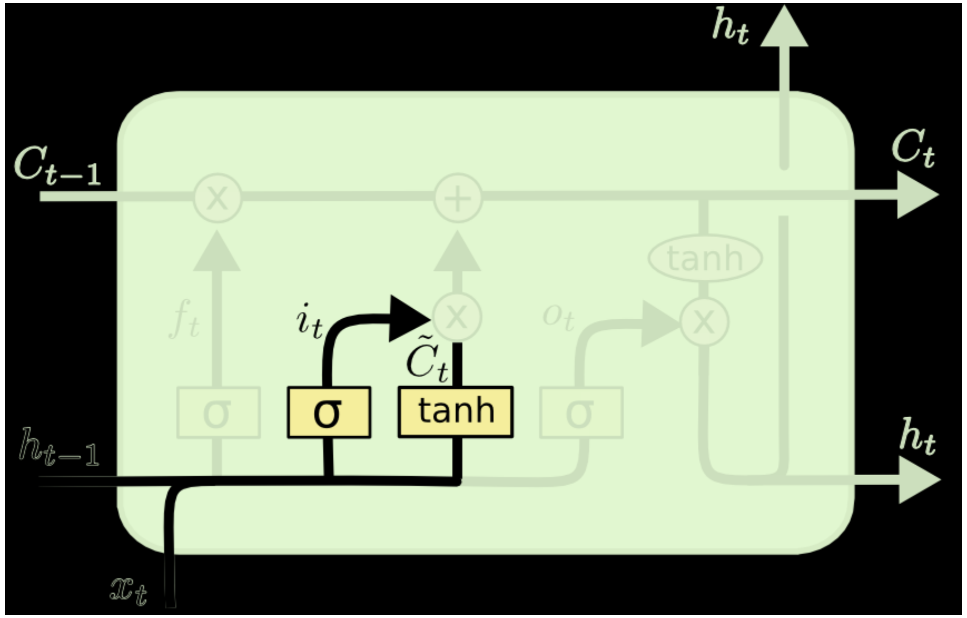
- $\tilde{C}_t$ is the **candidate cell state**, representing the new information that could be added to the cell state.
- W_C^x is the weight matrix for the current input x_t when computing the candidate cell state.
- W_C^h is the weight matrix for the previous hidden state h_{t-1}.
- b_C is the bias for the candidate cell state.
- $\tanh$ is the hyperbolic tangent activation function, which outputs values in the range $[-1, 1]$. This allows the LSTM to propose both positive and negative changes to the cell state.

2. Input gate activation: Determines how much new information will be allowed into the cell state.

$$i_t = \sigma(W_i^x \cdot x_t + W_i^h \cdot h_{t-1} + b_i)$$

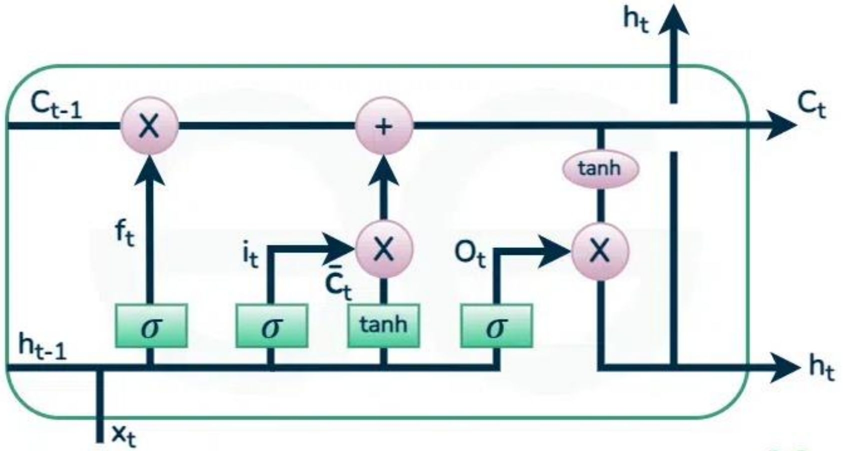
Where:

- i_t is the input gate activation, a vector of values between 0 and 1 that controls how much of the candidate cell state should be added to the current cell state.
- W_i^x is the weight matrix for the input x_t.
- W_i^h is the weight matrix for the hidden state h_{t-1}.
- b_i is the bias term for the input gate.
- σ is the sigmoid activation function, which ensures the output values are between 0 and 1.



Updating the Cell State:

The final update to the cell state C_t is controlled by combining the input gate activation i_t and the candidate cell state $\tilde{C}_t$. The input gate decides how much of $\tilde{C}_t$ should be added to the existing cell state C_t.



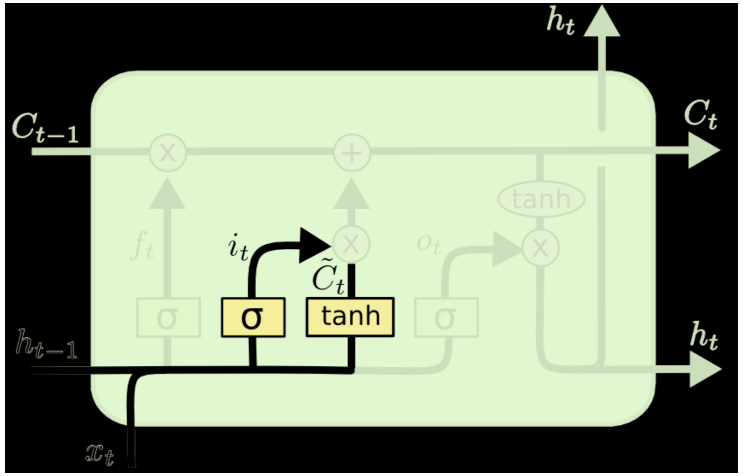
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Where:

- C_t is the updated cell state at time step t.
- f_t is the forget gate activation, which decides how much of the previous cell state C_{t-1} to retain.
- $i_t \odot \tilde{C}_t$ is the new information being added to the cell state, with i_t controlling how much of $\tilde{C}_t$ is used.
- $\odot$ represents element-wise multiplication (Hadamard product).

Summary of input gate :

- The **input gate** in an LSTM controls how much of the new information $\tilde{C}_t$ will be stored in the cell state.
- The gate consists of two components: the input gate activation i_t (deciding how much of the new information should be added) and the candidate cell state $\tilde{C}_t$ (the actual new information proposed for the cell state).
- The LSTM combines the input gate output with the forget gate's output to produce a carefully regulated update to the cell state, enabling the network to retain important long-term information and discard less important short-term information.



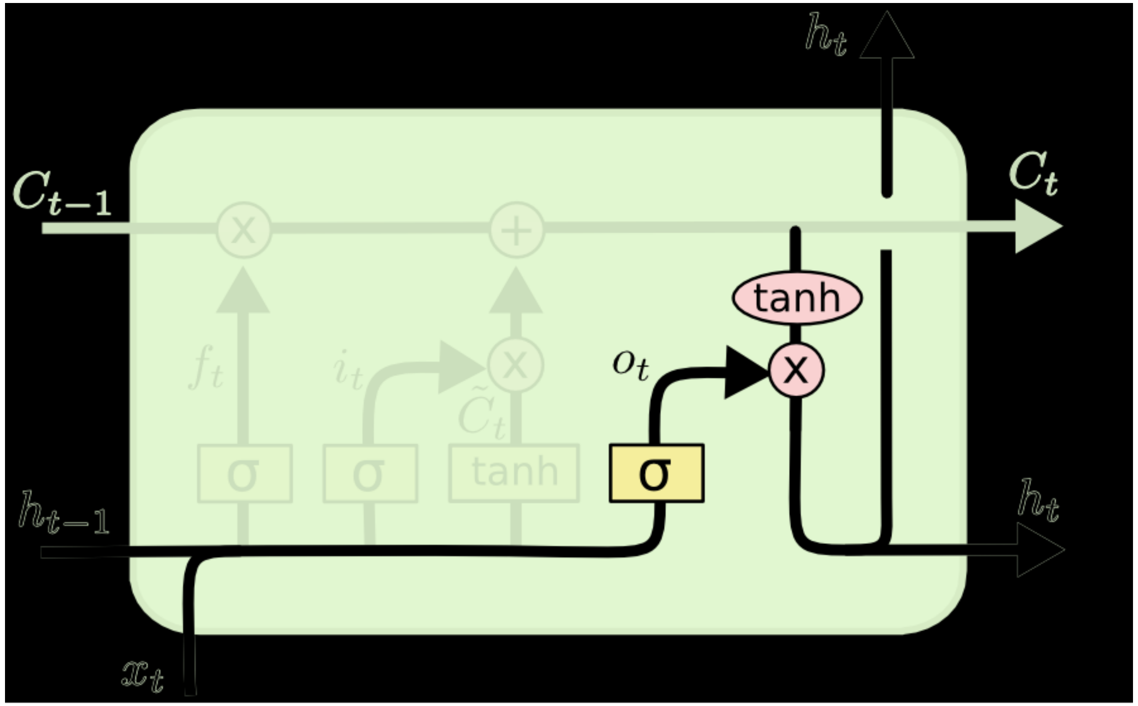
- **Input Gate: Example**

  - Eg: “Bob knows swimming. He told me over the phone that he had served the navy for four long years.”
  - based on the context given in the first sentence, which information of the second sentence is critical?
  - The fact that he was in the navy is important information and this is something we want our model to remember. This is the task of the Input gate.

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

Output gate:

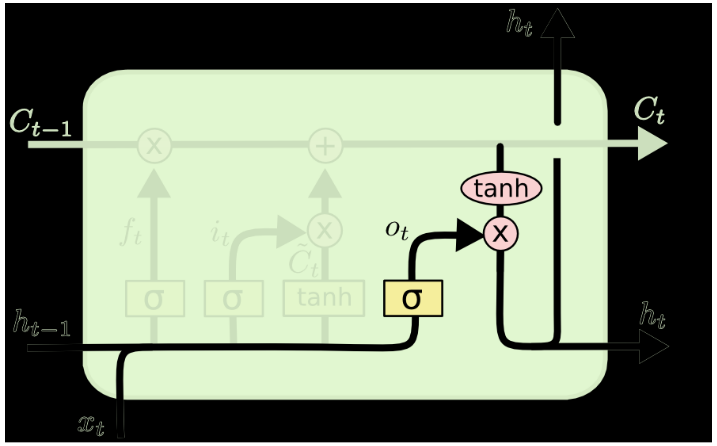
- The **output gate** in an **LSTM (Long Short-Term Memory)** cell controls how much of the current cell state Ct should be exposed as the hidden state ht.
- The hidden state ht is what gets passed to the next time step and is also used for predictions or other tasks in the LSTM's output.
- The output gate effectively determines how much information from the current cell state should influence the next hidden state.



The output gate involves two key components:

1. **Output gate activation**: Controls the flow of information from the cell state to the hidden state.

2. **Computation of the hidden state**: Determines the value of the hidden state h_t, which is a filtered version of the cell state.



Output Gate Activation:

$$o_t = \sigma(W_o^x \cdot x_t + W_o^h \cdot h_{t-1} + b_o)$$

Where:

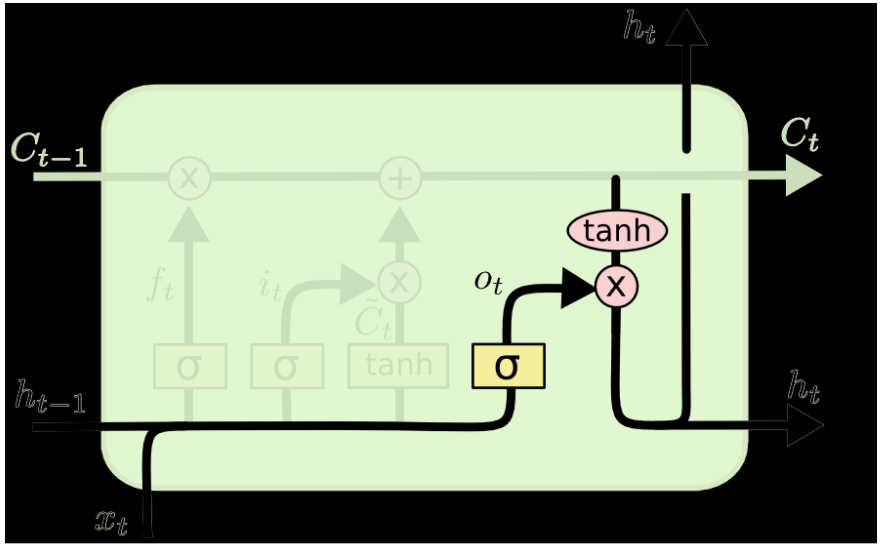
- o_t is the **output gate activation**, a vector of values between 0 and 1 that controls how much of the current cell state C_t is passed to the hidden state h_t.
- W_o^x is the weight matrix for the current input x_t when computing the output gate.
- W_o^h is the weight matrix for the previous hidden state h_{t-1} when computing the output gate.
- b_o is the bias term for the output gate.
- σ is the sigmoid activation function, which outputs values between 0 and 1.

Hidden State Computation:

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

Where:

- h_t is the new **hidden state** at time step t, representing the output of the LSTM at that time step.
- o_t is the output gate activation, determining how much of the cell state's information will be output as the hidden state.
- $\tanh(C_t)$ is the hyperbolic tangent of the current cell state C_t, which scales the cell state's values to a range between -1 and 1.
- $\odot$ represents element-wise multiplication (Hadamard product), meaning that the output gate o_t controls which parts of $\tanh(C_t)$ are passed to the hidden state.



- The output gate, together with the forget and input gates, helps the LSTM maintain a balance between remembering important long-term information and generating meaningful short-term outputs at each time step.
- It turns out that the hidden state is a function of Long term memory (C_t) and the current output.

- Output Gate
  - Eg: “Bob single-handedly fought the enemy and died for his country. For his contributions, brave________ .”
  - During this task, we have to complete the second sentence. Now, the minute we see the word brave, we know that we are talking about a person. In the sentence only Bob is brave, we cannot say the enemy is brave or the country is brave. So based on the current expectation we have to give a relevant word to fill in the blank. That word is our output and this is the function of Output gate.

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

BiLSTM

- Eg:1

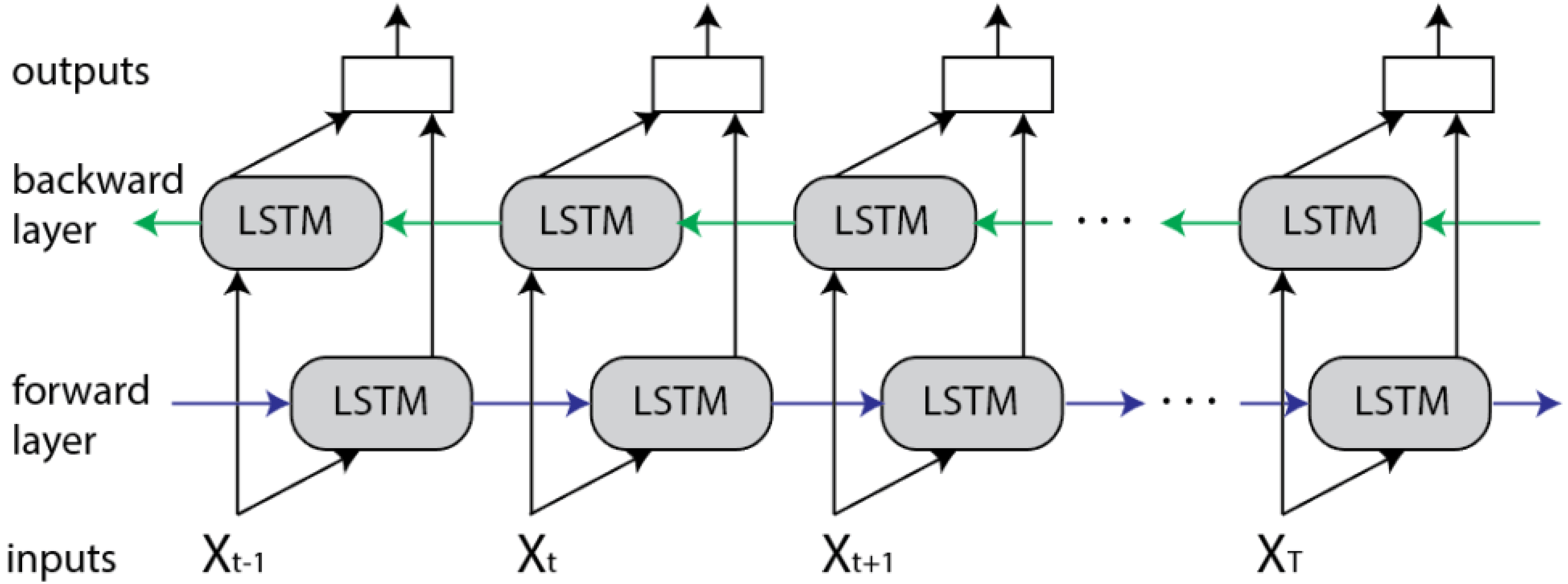
- Eg 2:

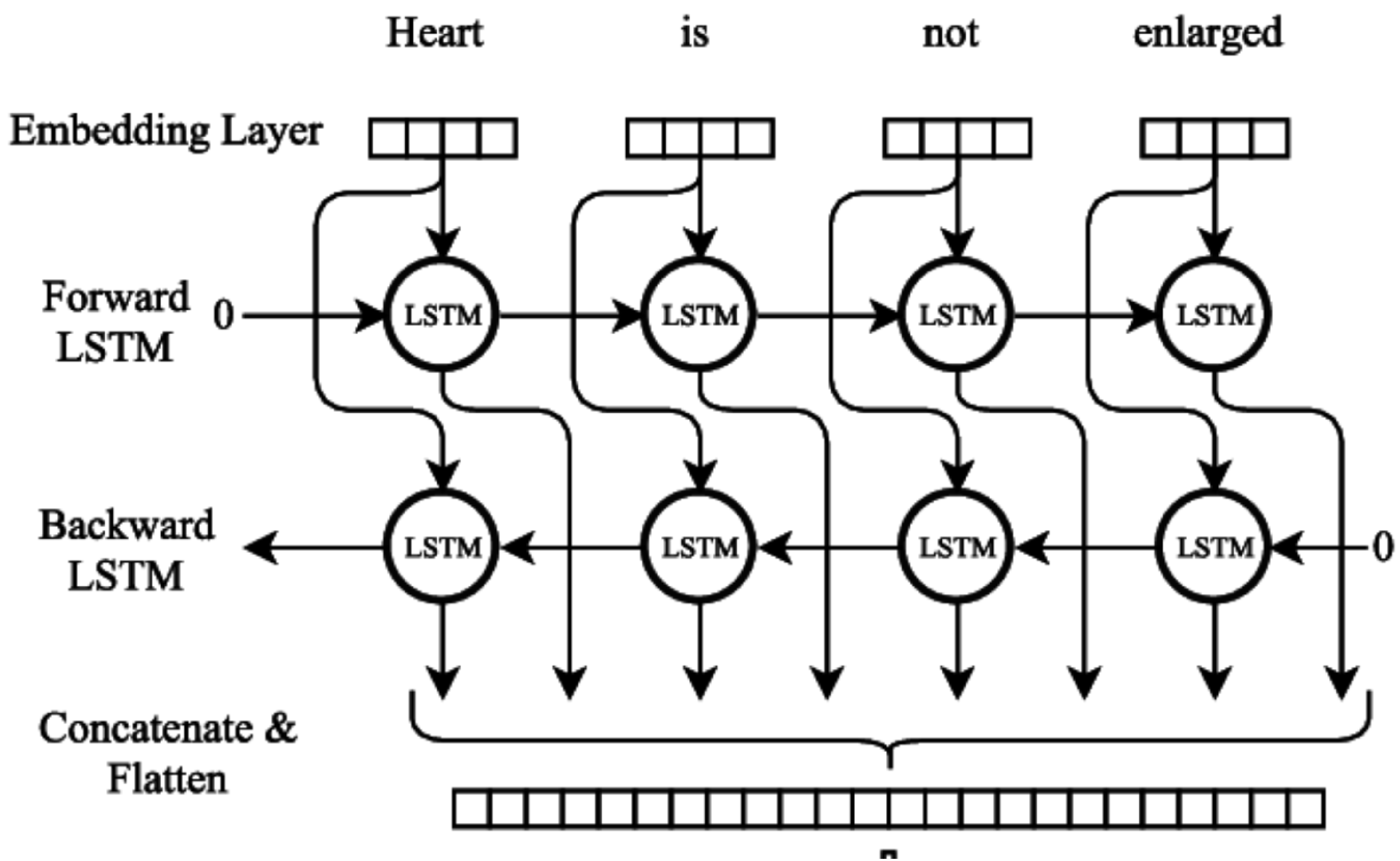
Apple is something that competitors simply cannot reproduce.

Apple is something that I like to eat.

- sometimes to understand a word we need not just to the previous word , but also to the coming word

- BiLSTM architecture




Heart
is
not
enlarged
Embedding Layer
Forward LSTM
0
LSTM
LSTM
LSTM
LSTM
Backward LSTM
LSTM
LSTM
LSTM
LSTM
0
Concatenate & Flatten

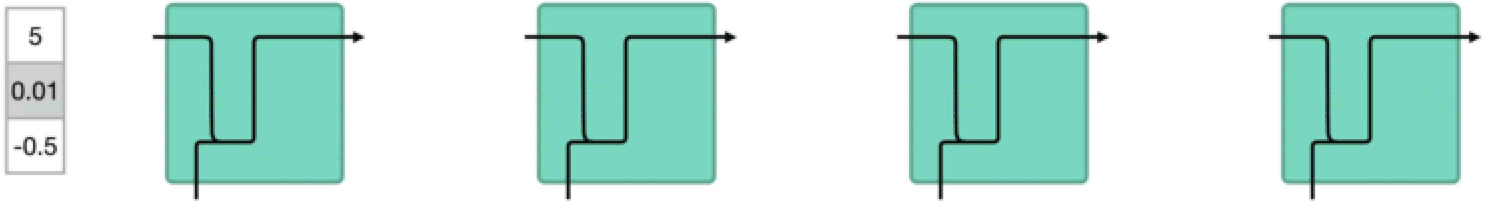
- Unlike standard LSTM, the input flows in both directions, and it is capable of utilizing information from both sides.
- BiLSTM adds one more LSTM layer ; it means that the input sequence flows backward in the additional LSTM layer.
- Then we combine the outputs from both LSTM layers in several ways, such as average, sum, multiplication, or concatenation.

Advantages of BiLSTM

- This type of architecture has many advantages in real-world problems, especially in NLP.
- The main reason is that every component of an input sequence has information from both the past and present.
- For this reason, BiLSTM can produce a more meaningful output, combining LSTM layers from both directions.

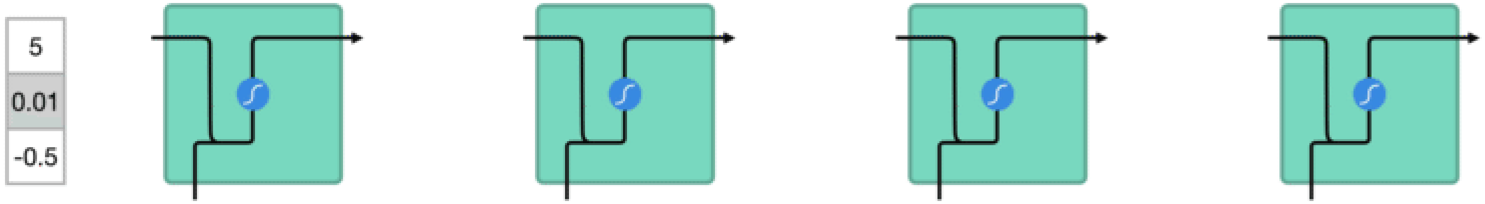
Appendix

- **Tanh activation: What it does**
  - When vectors are flowing through a neural network, it undergoes many transformations due to various math operations.
  - So, imagine a value that continues to be multiplied by let's say ***3***. You can see how some values can explode and become astronomical, causing other values to seem insignificant.



vector transformations without tanh

- A tanh function ensures that the values stay between -1 and 1, thus regulating the output of the neural network.



Reference

Understanding LSTM Networks -- colah's blog

Simple Explanation of LSTM | Deep Learning Tutorial 36 (Tensorflow, Keras & Python) (youtube.com)

Long Short Term Memory: Predict the Next Word (analyticsvidhya.com)

What is LSTM?- Introduction to Long Short-Term Memory (analyticsvidhya.com)