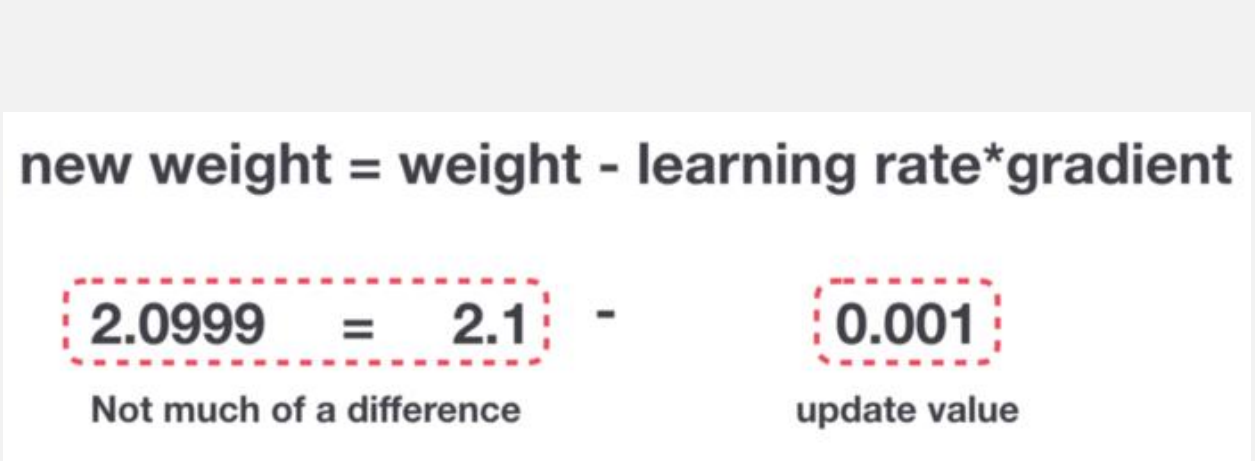
Week 10: LSTM

The Problem, Short-term Memory

Recurrent Neural Networks suffer from short-term memory. If a sequence is long enough, they'll have a hard time carrying information from earlier time steps to later ones. So if you are trying to process a paragraph of text to do predictions, RNN's may leave out important information from the beginning.

During backpropagation, recurrent neural networks suffer from the vanishing gradient problem. Gradients are values used to update the weights of a neural network. The vanishing gradient problem is when the gradient shrinks as it propagates through time. If a gradient value becomes extremely small, it doesn't contribute too much learning.

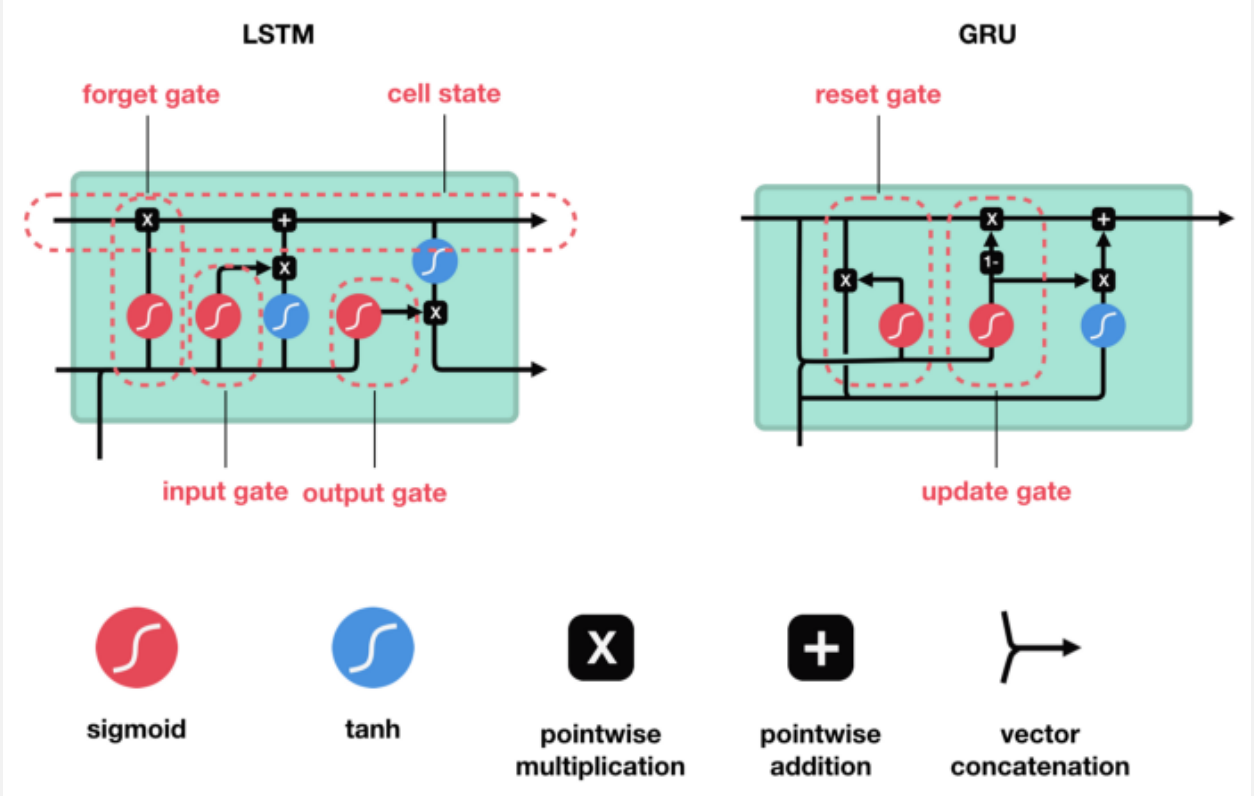


Gradient Update Rule

So in recurrent neural networks, layers that get a small gradient update stop learning. Those are usually the earlier layers. So because these layers don't learn, RNN's can forget what is seen in longer sequences, thus having a short-term memory.

LSTM and GRU as a solution

LSTM 's and GRU's were created as the solution to short-term memory. They have internal mechanisms called gates that can regulate the flow of information.



These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions. Almost all state of the art results based on recurrent neural networks are achieved with these two networks. LSTM and GRU's can be found in speech recognition, speech synthesis, and text generation. You can even use them to generate captions for videos.

Intuition

Ok, Let's start with a thought experiment. Let's say you're looking at reviews online to determine if you want to buy Life cereal (don't ask me why). You'll first read the review then determine if someone thought it was good or if it was bad.

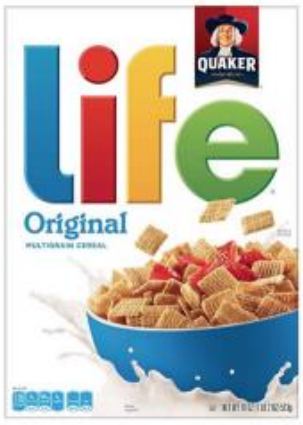
Customers Review 2,491

Thanos

September 2018

Verified Purchase

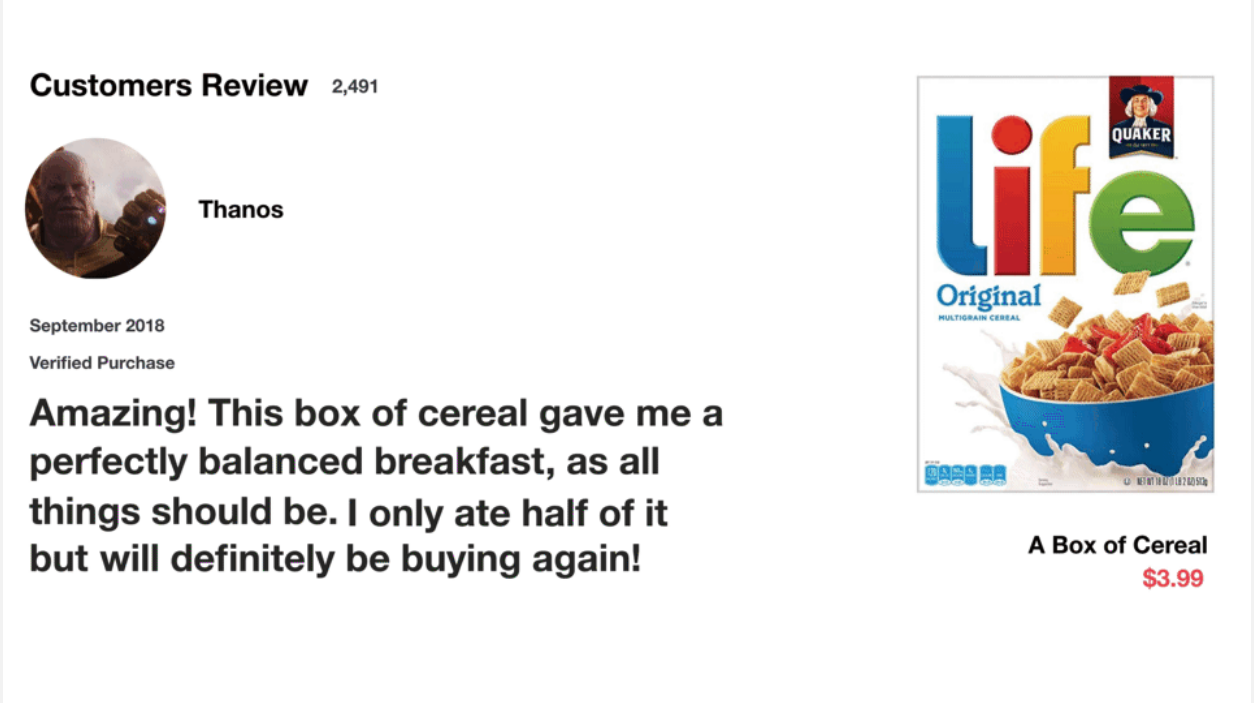
Amazing! This box of cereal gave me a perfectly balanced breakfast, as all things should be. I only ate half of it but will definitely be buying again!



A Box of Cereal

$3.99

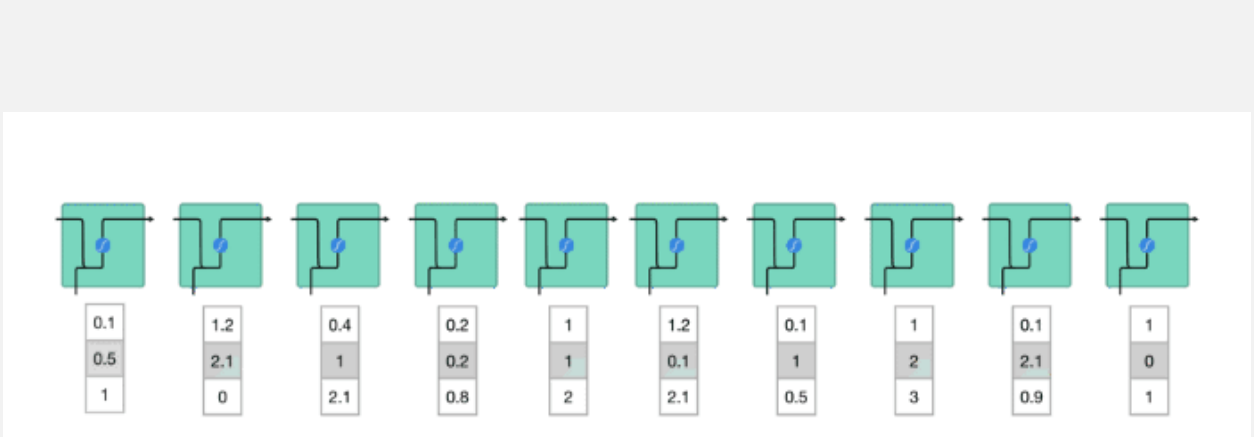
When you read the review, your brain subconsciously only remembers important keywords. You pick up words like “amazing” and “perfectly balanced breakfast”. You don’t care much for words like “this”, “gave“, “all”, “should”, etc. If a friend asks you the next day what the review said, you probably wouldn’t remember it word for word. You might remember the main points though like “will definitely be buying again”. If you’re a lot like me, the other words will fade away from memory.



And that is essentially what an LSTM or GRU does. It can learn to keep only relevant information to make predictions and forget non-relevant data. In this case, the words you remembered made you judge that it was good.

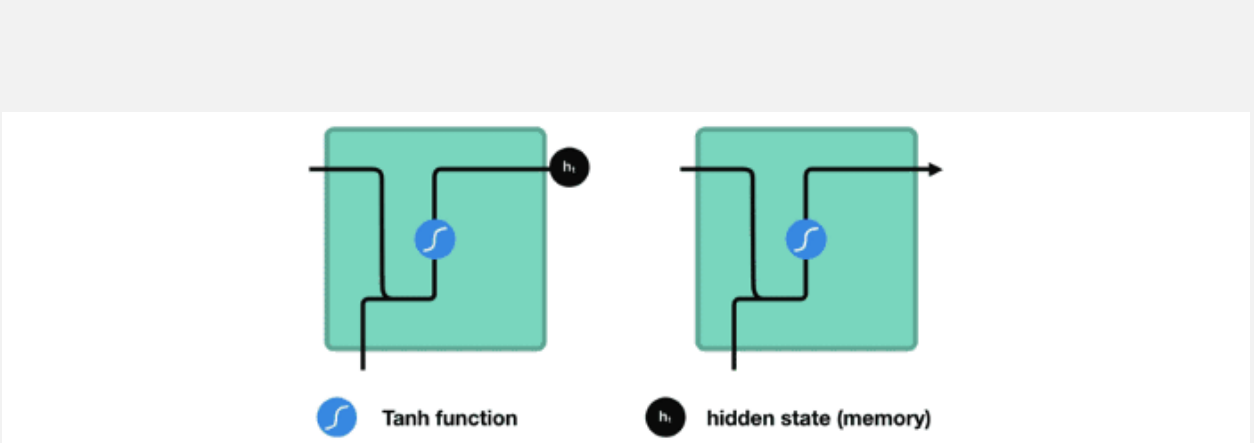
Review of Recurrent Neural Networks

To understand how LSTM's or GRU's achieves this, let's review the recurrent neural network. An RNN works like this; First words get transformed into machine-readable vectors. Then the RNN processes the sequence of vectors one by one.



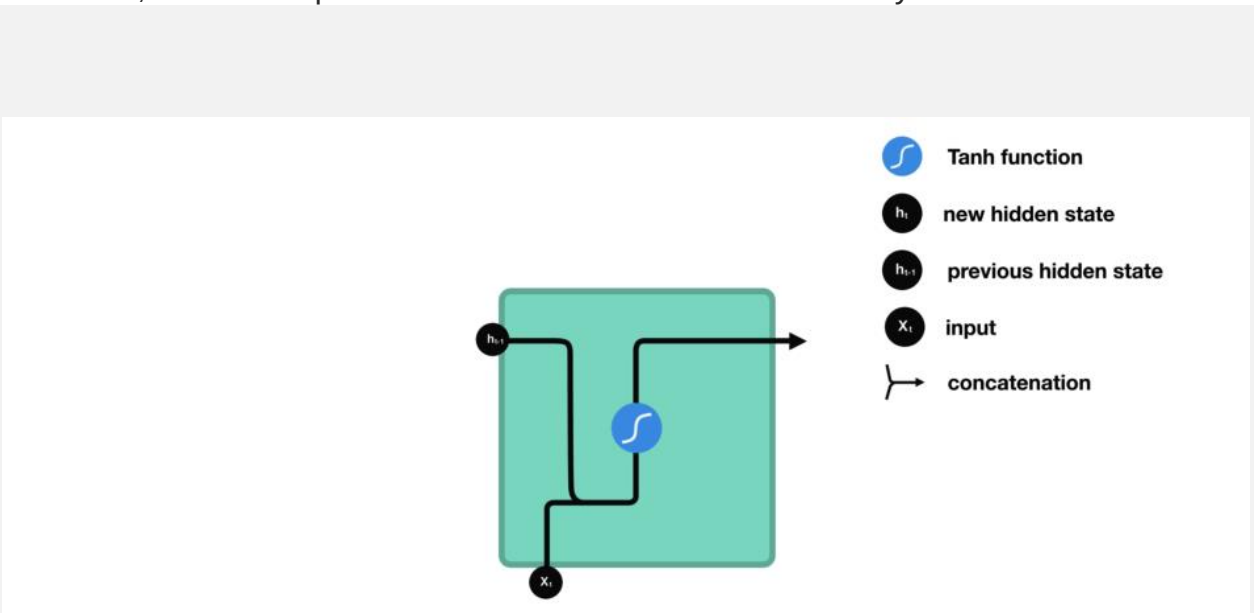
Processing sequence one by one

While processing, it passes the previous hidden state to the next step of the sequence. The hidden state acts as the memory of the neural network. It holds information on previous data the network has seen before.



Passing hidden state to next time step

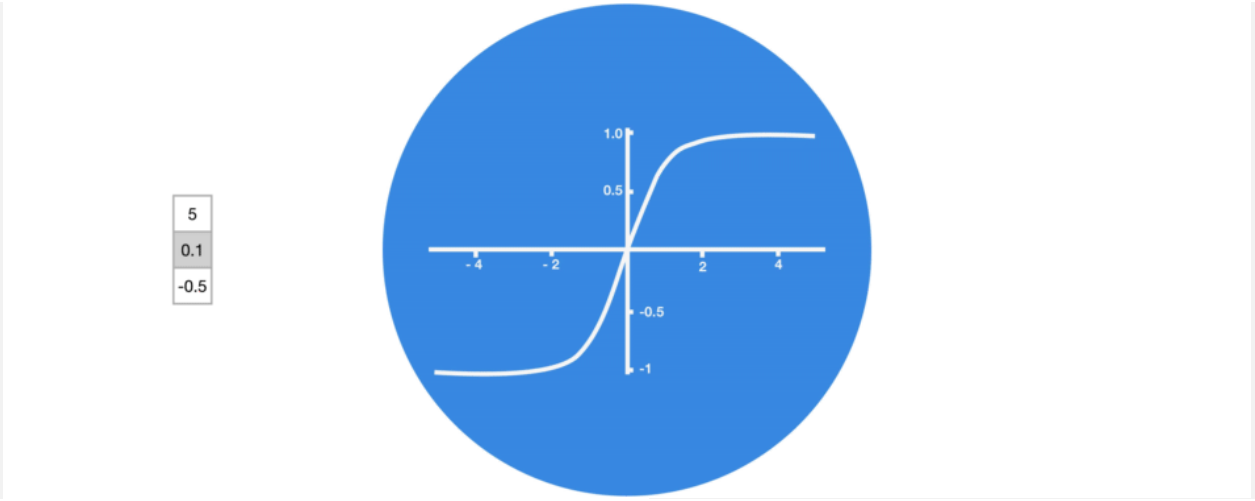
Let’s look at a cell of the RNN to see how you would calculate the hidden state. First, the input and previous hidden state are combined to form a vector. That vector now has information on the current input and previous inputs. The vector goes through the tanh activation, and the output is the new hidden state or the memory of the network.



RNN Cell

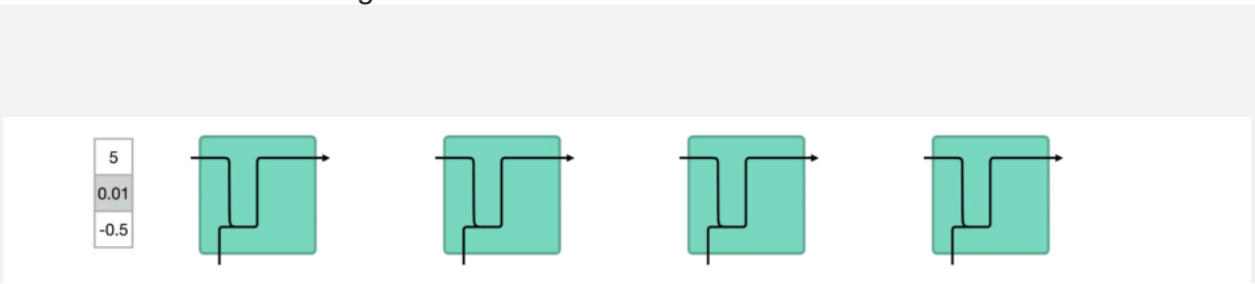
Tanh activation

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.



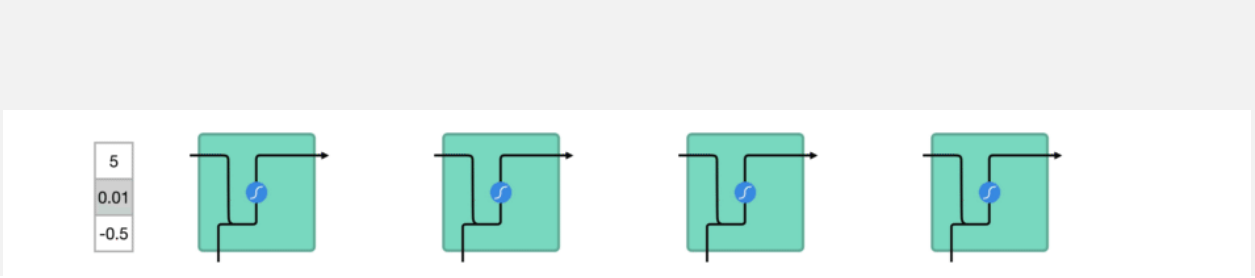
Tanh squishes values to be between -1 and 1

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

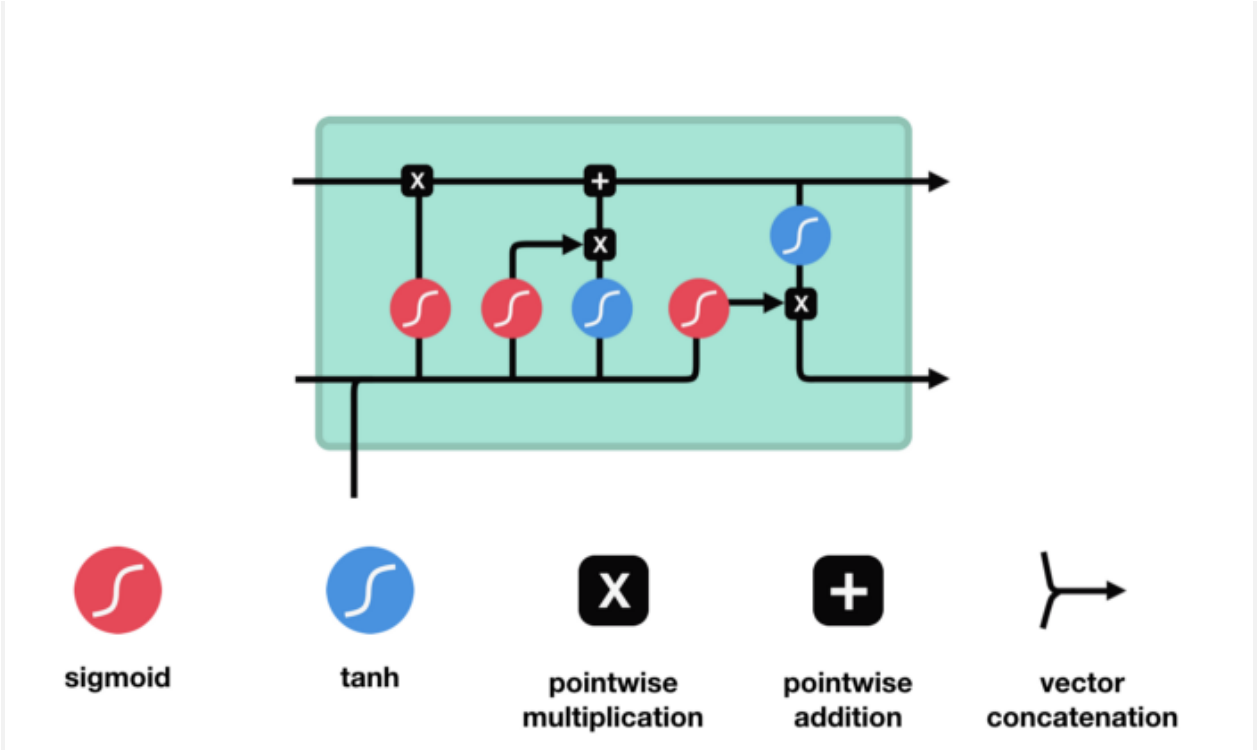
A tanh function ensures that the values stay between -1 and 1, thus regulating the output of the neural network. You can see how the same values from above remain between the boundaries allowed by the tanh function.

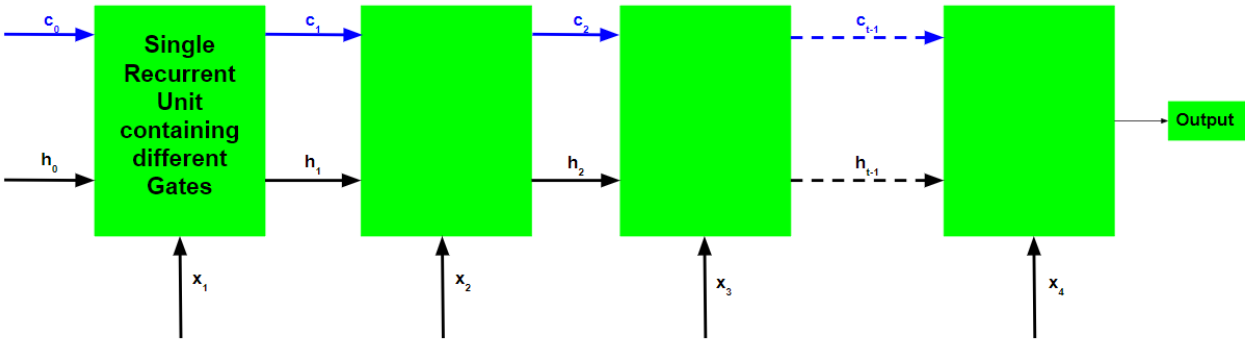


vector transformations with tanh

So that’s an RNN. It has very few operations internally but works pretty well given the right circumstances (like short sequences). RNN’s uses a lot less computational resources than it’s evolved variants, LSTM’s and GRU’s.

LSTM





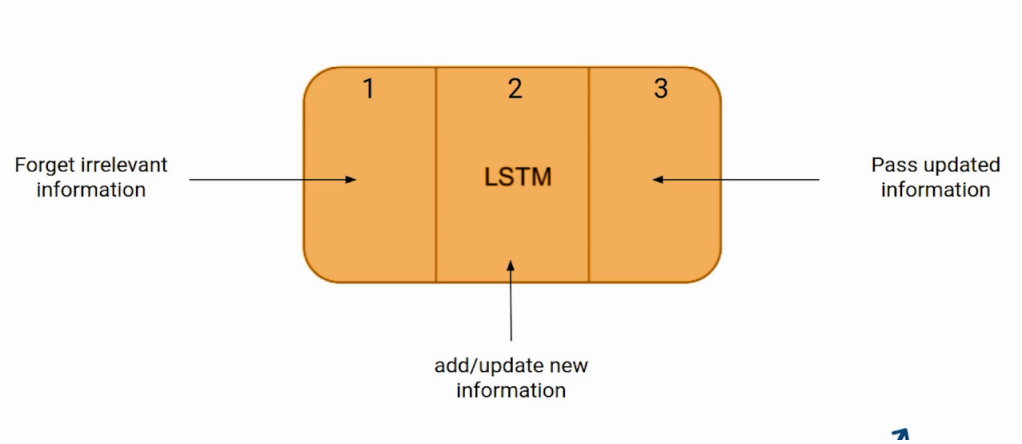
Introduction

Long Short Term Memory Network is an advanced RNN, a sequential network, that allows information to persist. It is capable of handling the vanishing gradient problem faced by RNN. A recurrent neural network also known as RNN is used for persistent memory.

Similarly RNNs work, they remember the previous information and use it for processing the current input. The shortcoming of RNN is, they can not remember Long term dependencies due to vanishing gradients. LSTMs are explicitly designed to avoid long-term dependency problems.

LSTM Architecture

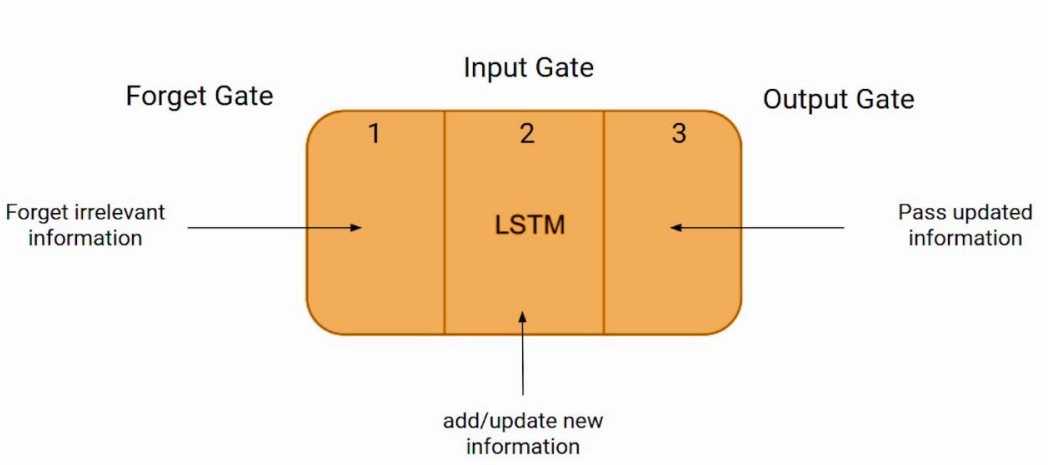
At a high-level LSTM works very much like an RNN cell. Here is the internal functioning of the LSTM network. The LSTM consists of three parts, as shown in the image below and each part performs an individual function.



The first part chooses whether the information coming from the previous timestamp is to be remembered or is irrelevant and can be forgotten. In the second part, the cell tries to

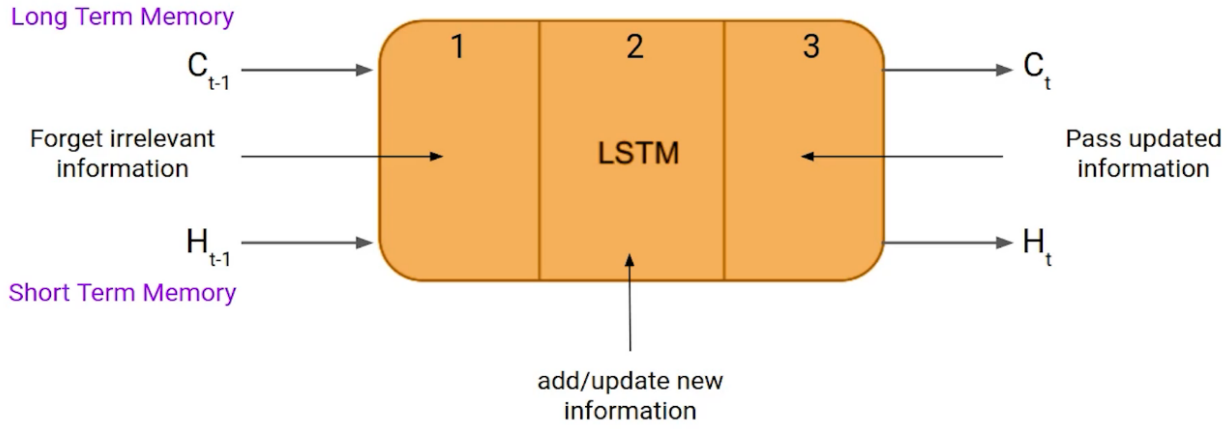
learn new information from the input to this cell. At last, in the third part, the cell passes the updated information from the current timestamp to the next timestamp.

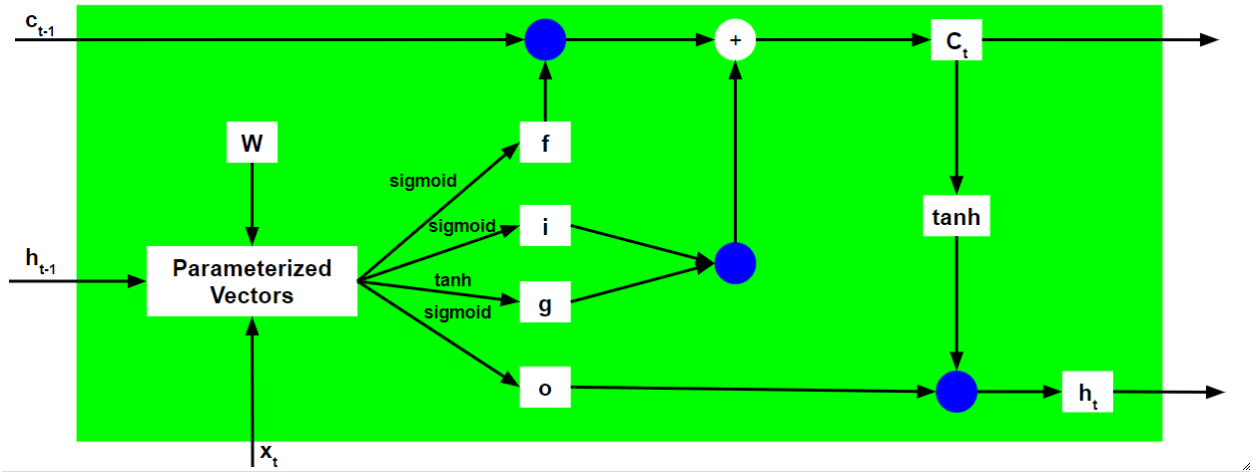
These three parts of an LSTM cell are known as gates. The first part is called Forget gate, the second part is known as the Input gate and the last one is the Output gate.



Just like a simple RNN, an LSTM also has a hidden state where H(t-1) represents the hidden state of the previous timestamp and Ht is the hidden state of the current timestamp. In addition to that, LSTM also has a cell state represented by C(t-1) and C(t) for previous and current timestamps respectively.

Here the hidden state is known as Short term memory and the cell state is known as Long term memory. Refer to the following image.





It is interesting to note that the cell state carries the information along with all the timestamps.

LSTM

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

Let's take an example to understand how LSTM works. Here we have two sentences separated by a full stop. The first sentence is "Bob is a nice person" and the second sentence is "Dan, on the Other hand, is evil". It is very clear, in the first sentence we are talking about Bob and as soon as we encounter the full stop(.) we start talking about Dan.

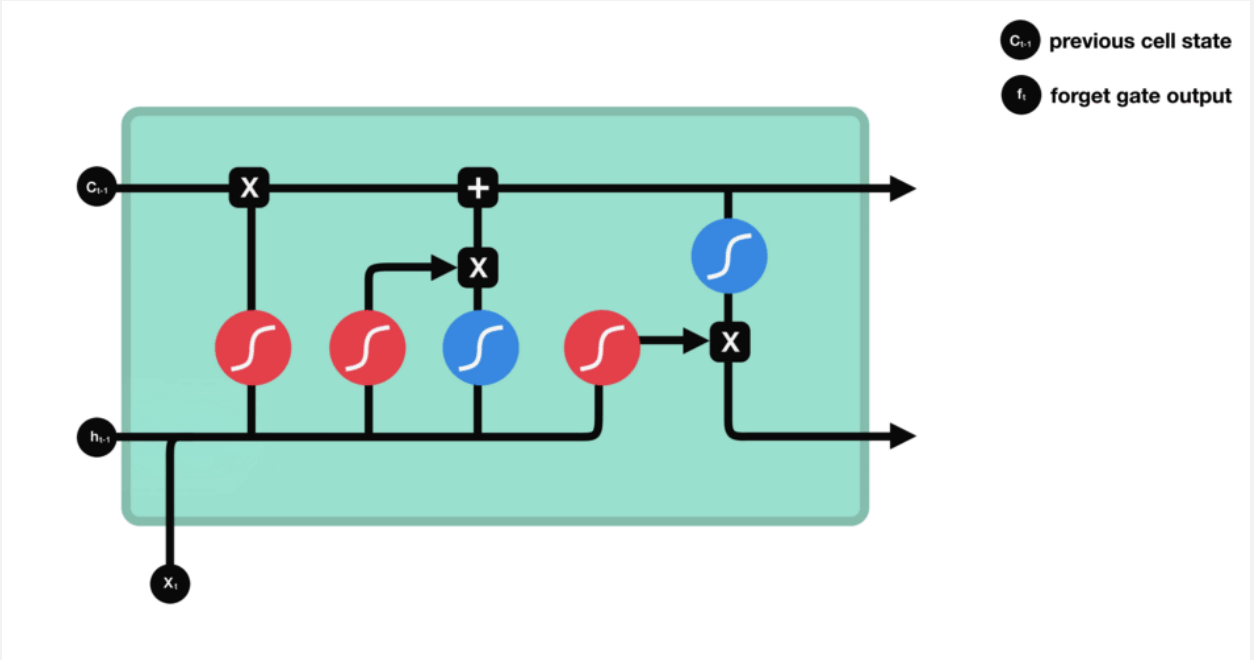
As we move from the first sentence to the second sentence, our network should realize that we are no more talking about Bob. Now our subject is Dan. Here, the Forget gate of the network allows it to forget about it. Let's understand the roles played by these gates in LSTM architecture.

Forget Gate

In a cell of the LSTM network, the first step is to decide whether we should keep the information from the previous timestamp or forget it. Here is the equation for the forget gate.

Forget Gate:

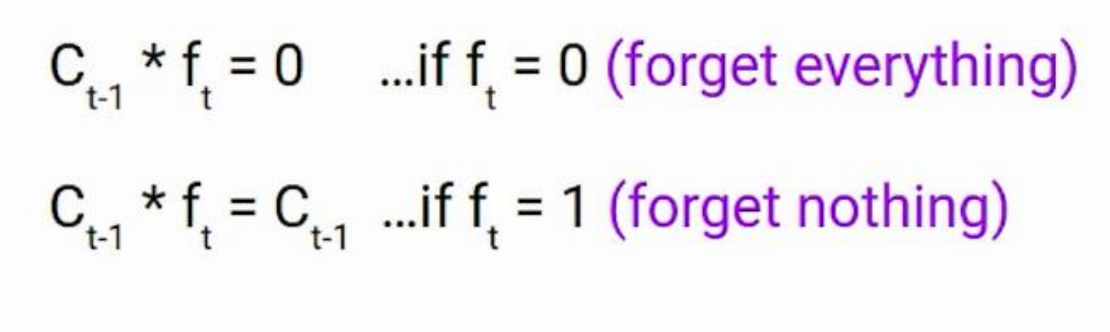
- $f_t = \sigma (x_t * U_f + H_{t-1} * W_f)$



Let's try to understand the equation, here

- Xt: input to the current timestamp.
- Uf: weight associated with the input
- Ht-1: The hidden state of the previous timestamp
- Wf: It is the weight matrix associated with hidden state

Later, a sigmoid function is applied over it. That will make ft a number between 0 and 1. This ft is later multiplied with the cell state of the previous timestamp as shown below.



If ft is 0 then the network will forget everything and if the value of ft is 1 it will forget nothing. Let's get back to our example. The first sentence was talking about Bob and after a full stop, the network will encounter Dan, in an ideal case the network should forget about Bob.

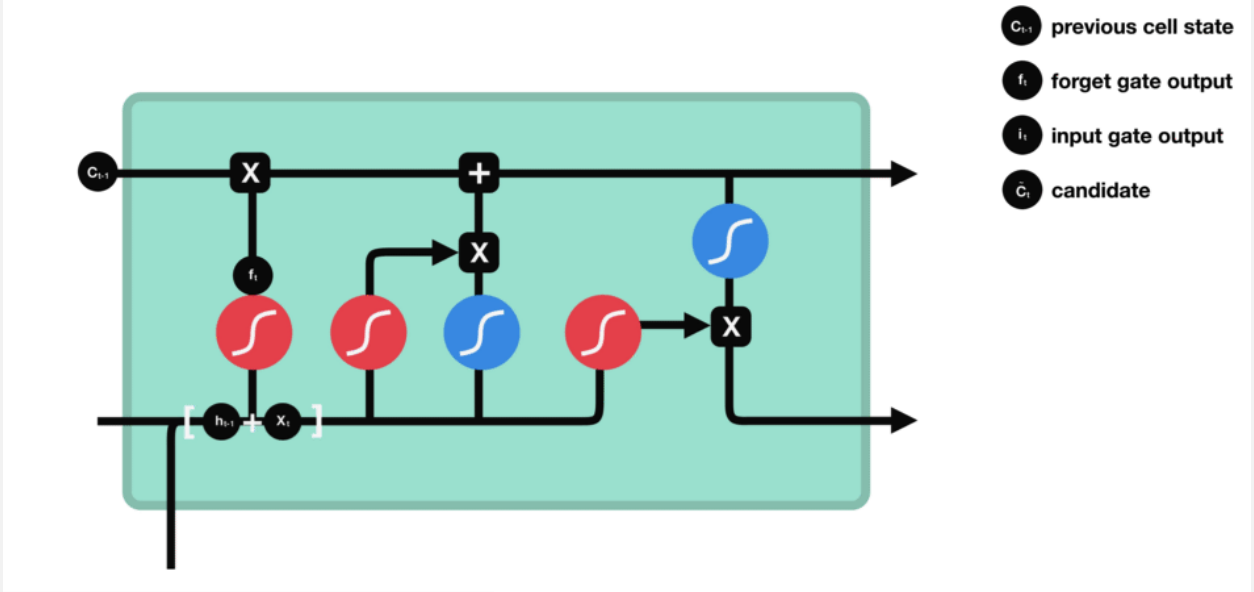
Input Gate

Let's take another example. "Bob knows swimming. He told me over the phone that he had served the navy for four long years."

So, in both these sentences, we are talking about Bob. However, both give different kinds of information about Bob. In the first sentence, we get the information that he knows how to swim. Whereas the second sentence tells he uses the phone and served in the navy for four years.

Now just think about it, based on the context given in the first sentence, which information of the second sentence is critical. First, he used the phone to tell if he served in the navy. In this context, it doesn't matter whether he used the phone or any other medium of communication to pass on the information. 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.

Input gate is used to quantify the importance of the new information carried by the input. Here is the equation of the input gate.



Input Gate:

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

Here,

- Xt: Input at the current timestamp t
- Ui: weight matrix of input
- Ht-1: A hidden state at the previous timestamp
- Wi: Weight matrix of input associated with hidden state

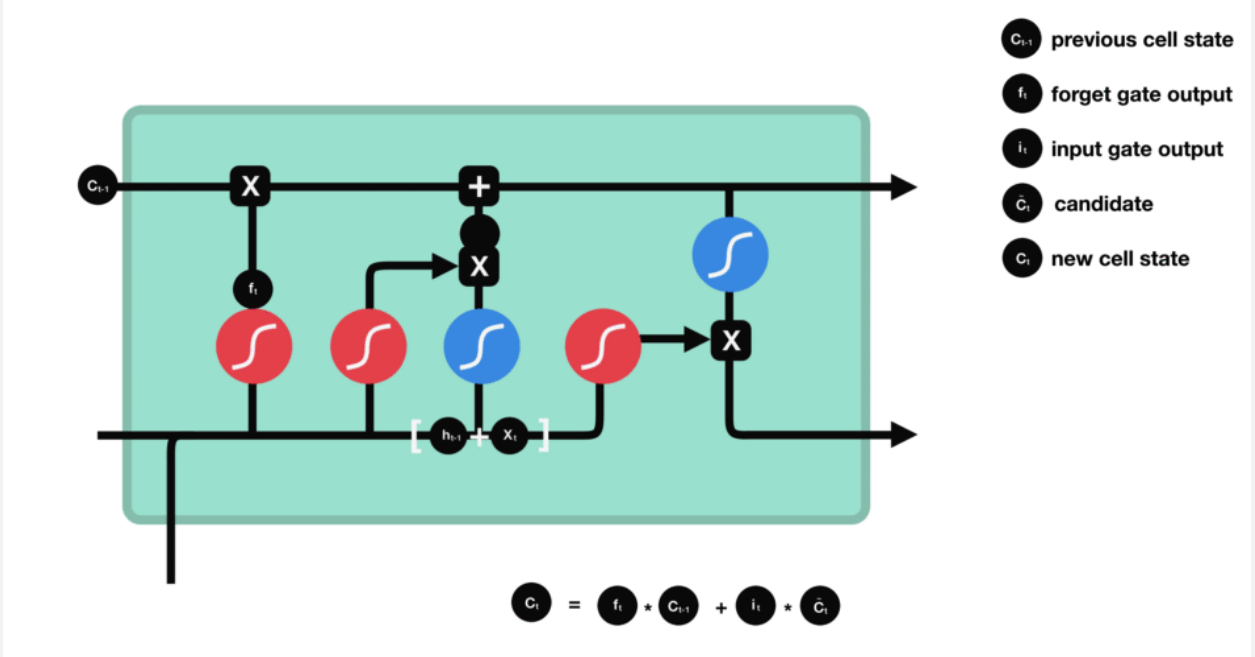
Again we have applied a sigmoid function over it. As a result, the value of I at timestamp t will be between 0 and 1.

New information

- $N_t = \tanh(x_t * U_c + H_{t-1} * W_c)$ (new information)

Now the new information that needs to be passed to the cell state is a function of a hidden state at the previous timestamp t-1 and input x at timestamp t. The activation function here is tanh. Due to the tanh function, the value of new information will be between -1 and 1. If the value of Nt is negative the information is subtracted from the cell state and if the value is positive the information is added to the cell state at the current timestamp.

However, the Nt won't be added directly to the cell state. Here comes the updated equation.



$C_t = f_t * C_{t-1} + i_t * N_t$ (updating cell state)

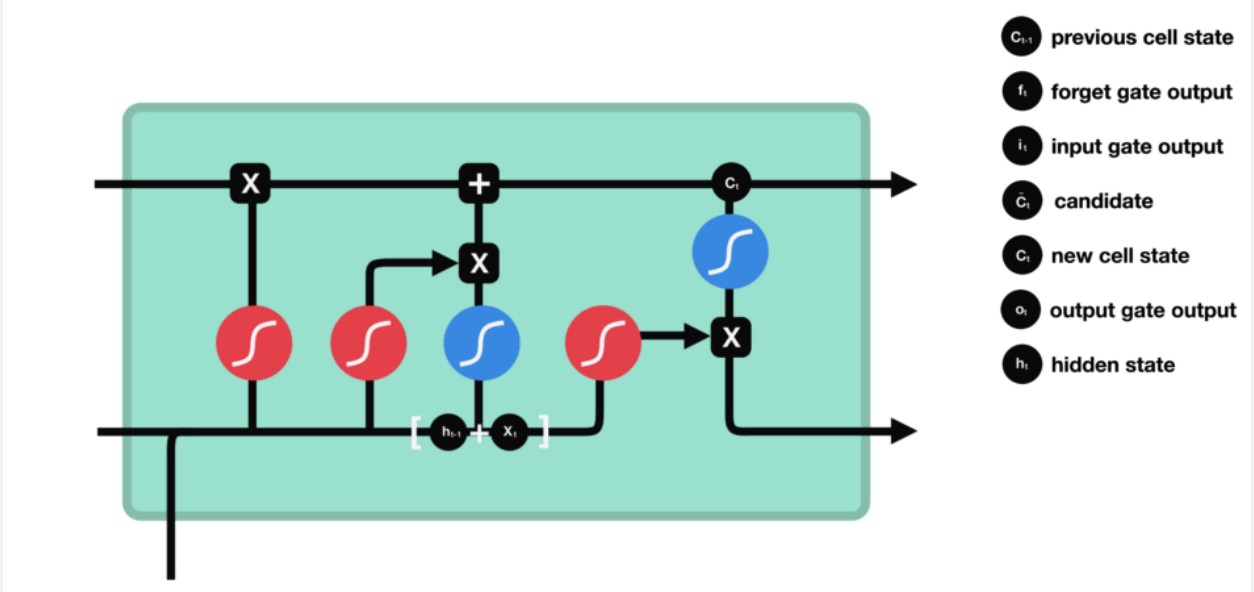
Here, Ct-1 is the cell state at the current timestamp and others are the values we have calculated previously.

Output Gate
Now consider this sentence

“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 can not 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 our Output gate.

Here is the equation of the Output gate, which is pretty similar to the two previous gates.



Output Gate:

- $o_t = \sigma(x_t * U_o + H_{t-1} * W_o)$

Its value will also lie between 0 and 1 because of this sigmoid function. Now to calculate the current hidden state we will use Ot and tanh of the updated cell state. As shown below.

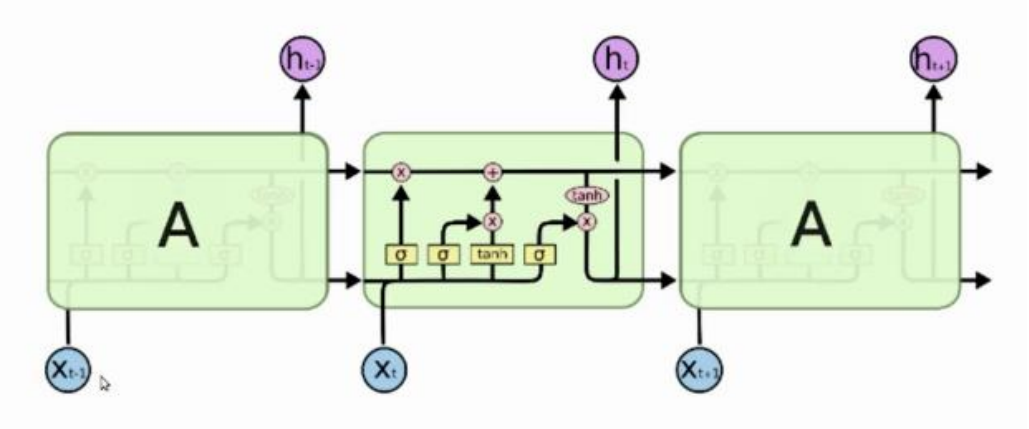
$$H_t = o_t * \tanh(C_t)$$

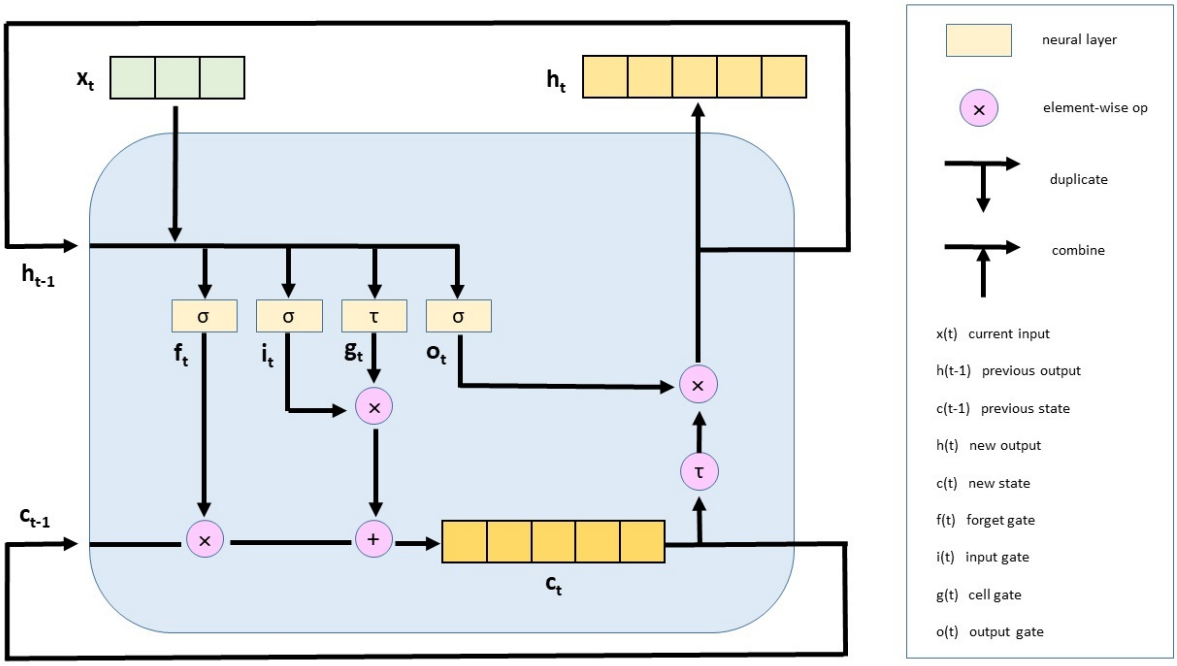
It turns out that the hidden state is a function of Long term memory (Ct) and the current output. If you need to take the output of the current timestamp just apply the SoftMax activation on hidden state Ht.

$$\text{Output} = \text{Softmax}(H_t)$$

Here the token with the maximum score in the output is the prediction.

This is the More intuitive diagram of the LSTM network.


h_{t-1}
h_t
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
X_{t-1}
X_t
X_{t+1}



$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$$

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$

$$g_t = \tau(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$

$$c_t = f_t \circ c_{(t-1)} + i_t \circ g_t$$

$$h_t = o_t \circ \tau(c_t)$$

References

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/

For Lab
https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/

https://towardsdatascience.com/recurrent-neural-networks-by-example-in-python-ffd204f99470

https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

https://www.kaggle.com/thebrownviking20/intro-to-recurrent-neural-networks-lstm-gru

https://machinelearningmastery.com/time-series-forecasting-long-short-term-memory-network-python/