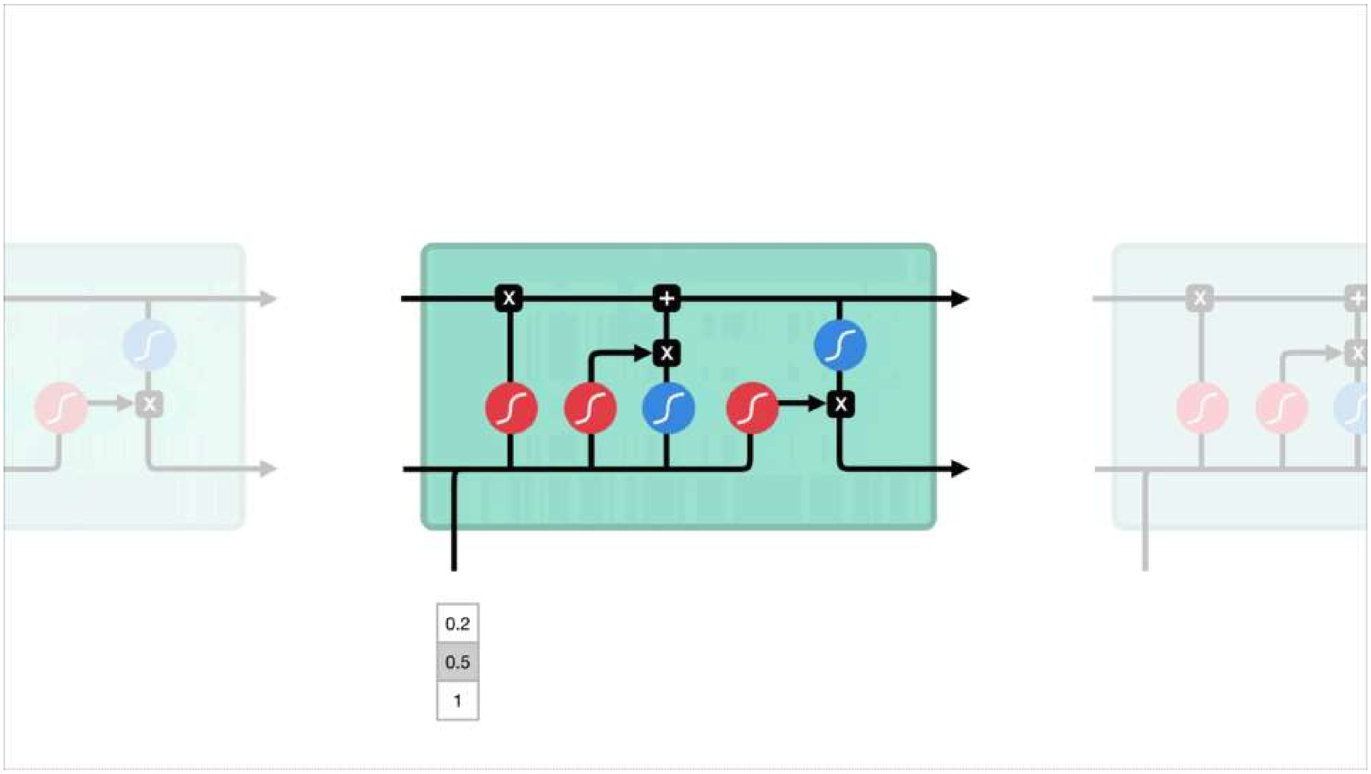
Illustrated Guide to LSTM's and GRU's: A step by step explanation

Michael Phi · Follow

Published in Towards Data Science · 10 min read · Sep 24, 2018



Hi and welcome to an Illustrated Guide to Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). I'm Michael, and I'm a Machine Learning Engineer in the AI voice assistant space.

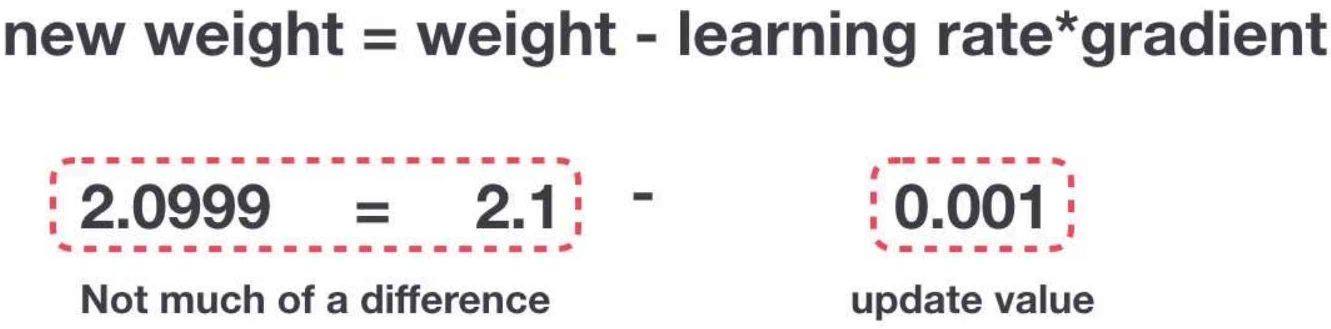
In this post, we'll start with the intuition behind LSTM 's and GRU's. Then I'll explain the internal mechanisms that allow LSTM's and GRU's to perform so well. If you want to understand what's happening under the hood for these two networks, then this post is for you.

You can also watch the video version of this post on youtube if you prefer.

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 back propagation, recurrent neural networks suffer from the vanishing gradient problem. Gradients are values used to update a neural networks weights. The vanishing gradient problem is when the gradient shrinks as it back 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 stops learning. Those are usually the earlier layers. So because these layers don't learn, RNN's can forget what it seen in longer sequences, thus having a short-term memory. If you want to know more about the mechanics of recurrent neural networks in general, you can read my previous post here.

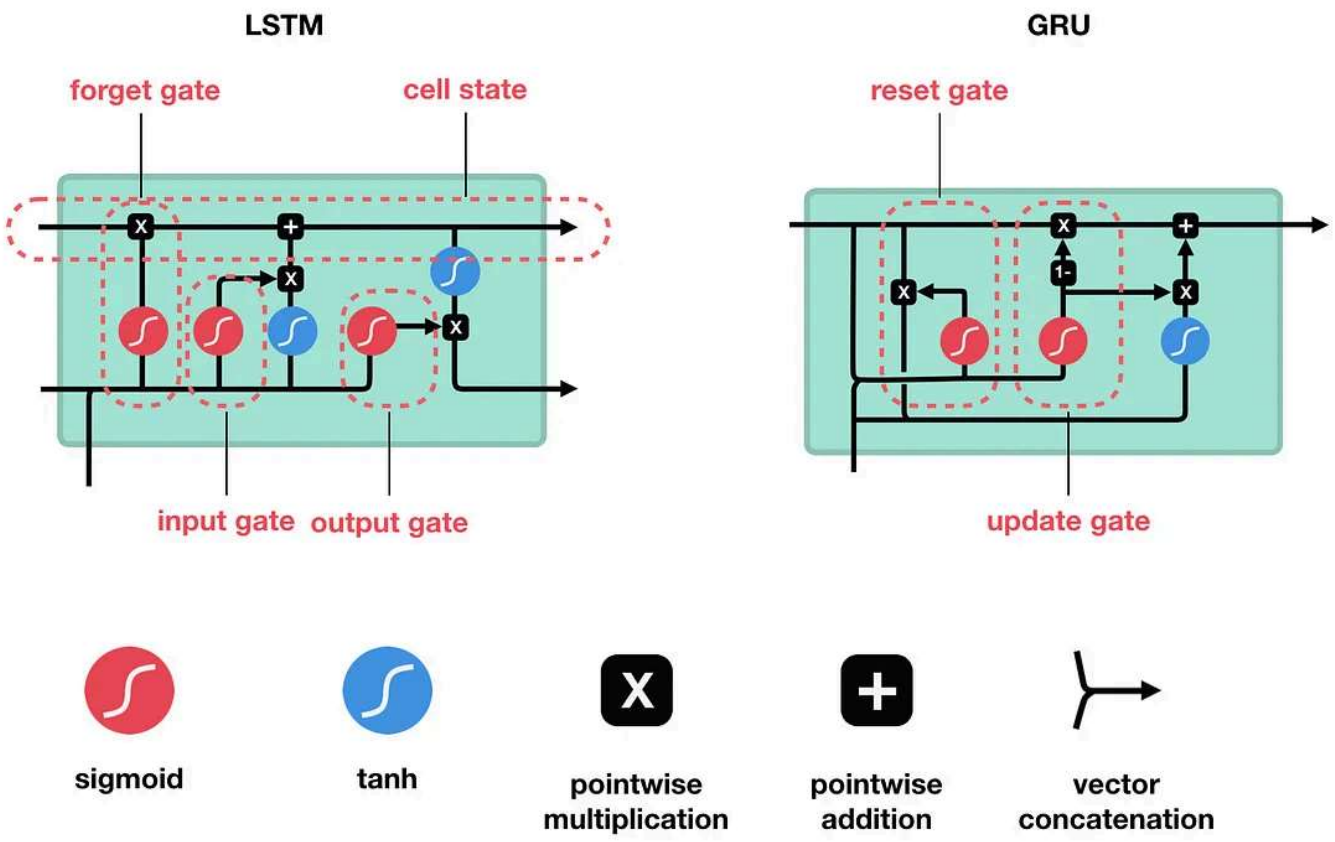
Illustrated Guide to Recurrent Neural Networks

Hi and welcome to an Illustrated guide to recurrent neural networks. I'm Michael also known as LearnedVector. I'm a...

towardsdatascience.com

LSTM's and GRU's 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.

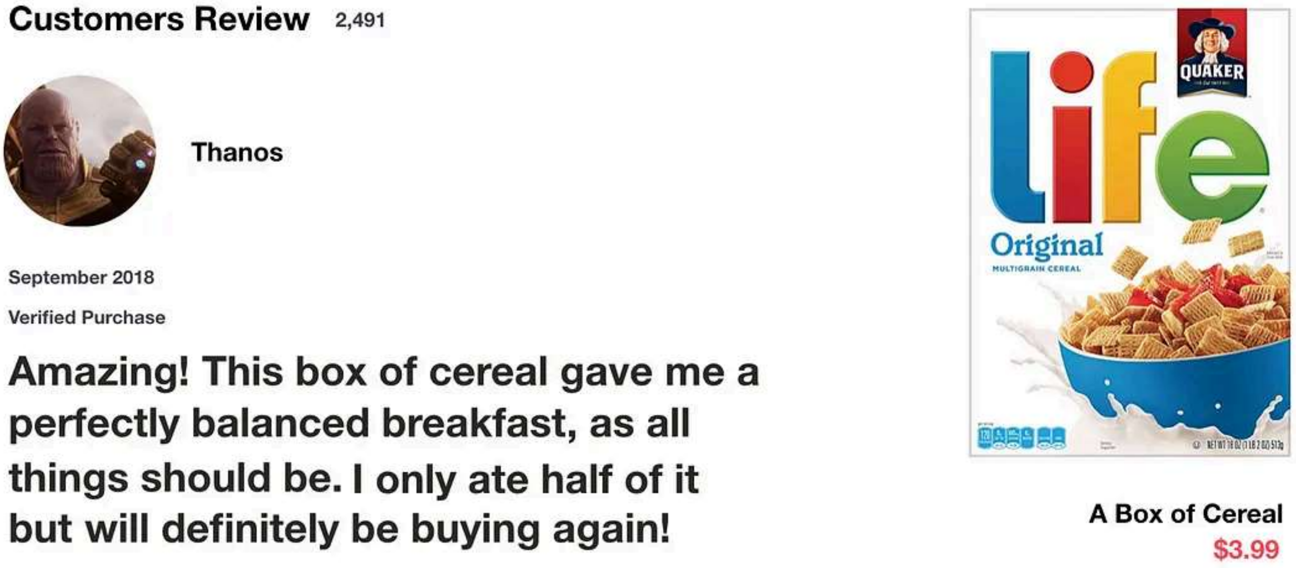


These gates can learn which data in a sequence is important to keep or throw away. By doing that, it can pass relevant information down the long chain of sequences to make predictions. Almost all state of the art results based on recurrent neural networks are achieved with these two networks. LSTM’s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

Ok, so by the end of this post you should have a solid understanding of why LSTM’s and GRU’s are good at processing long sequences. I am going to approach this with intuitive explanations and illustrations and avoid as much math as possible.

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

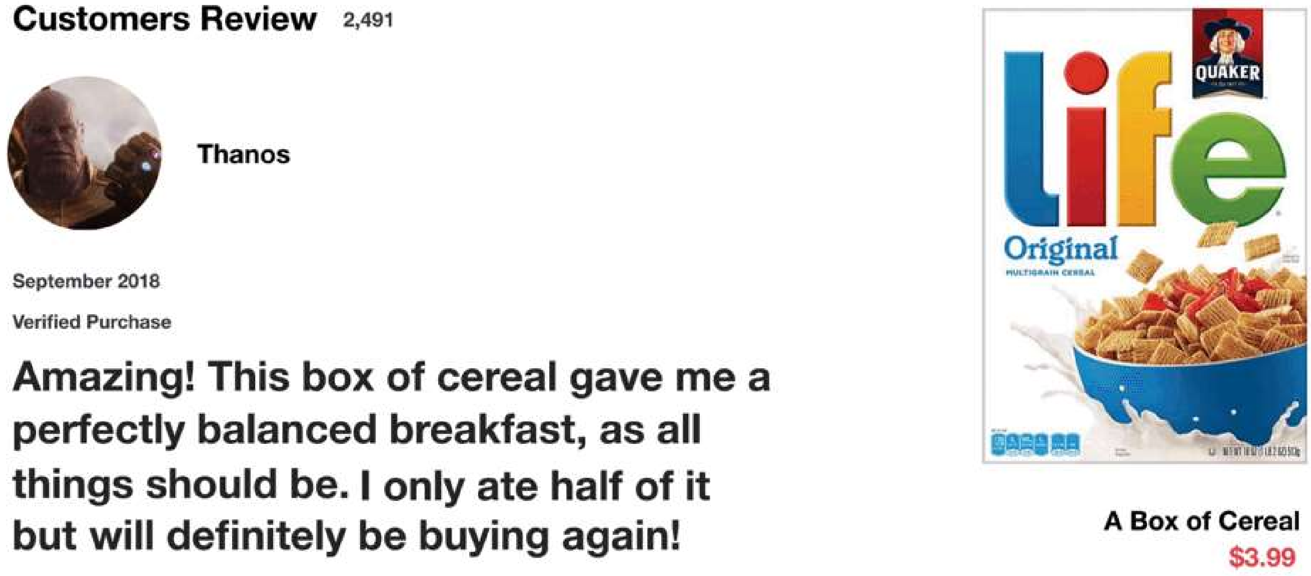
Verified Purchase

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.



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!

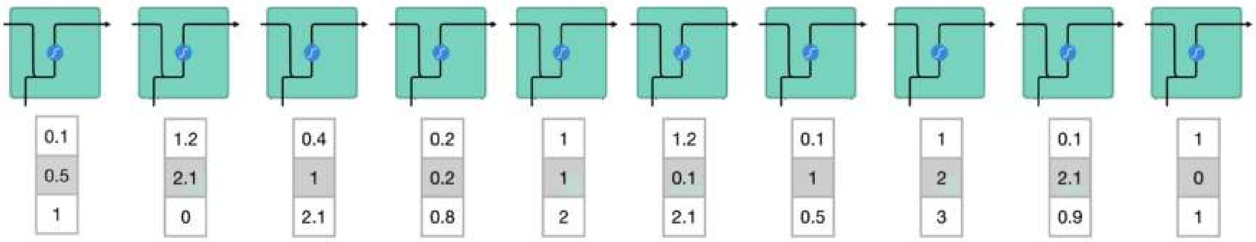
A Box of Cereal

$3.99

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.

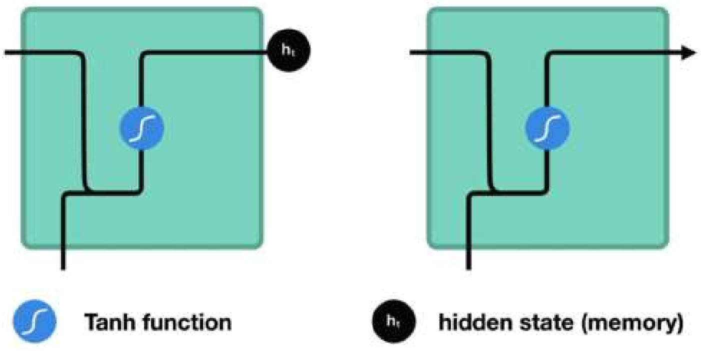
Review of Recurrent Neural Networks

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.



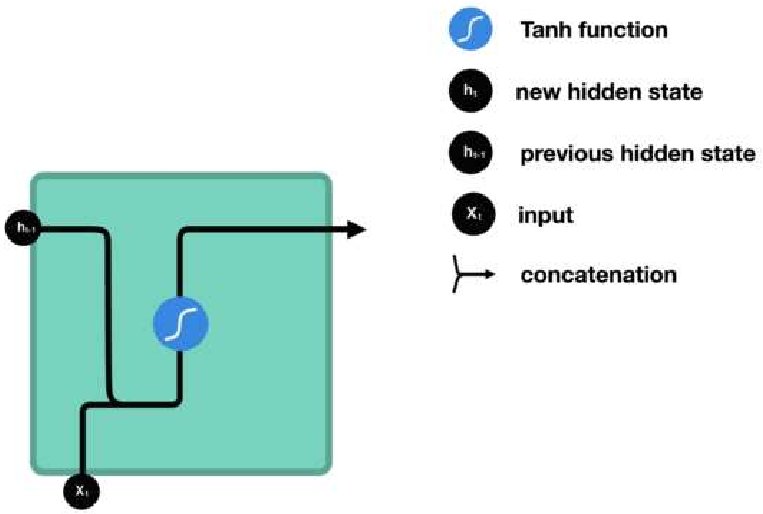
Processing sequence one by one

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



Passing hidden state to next time step

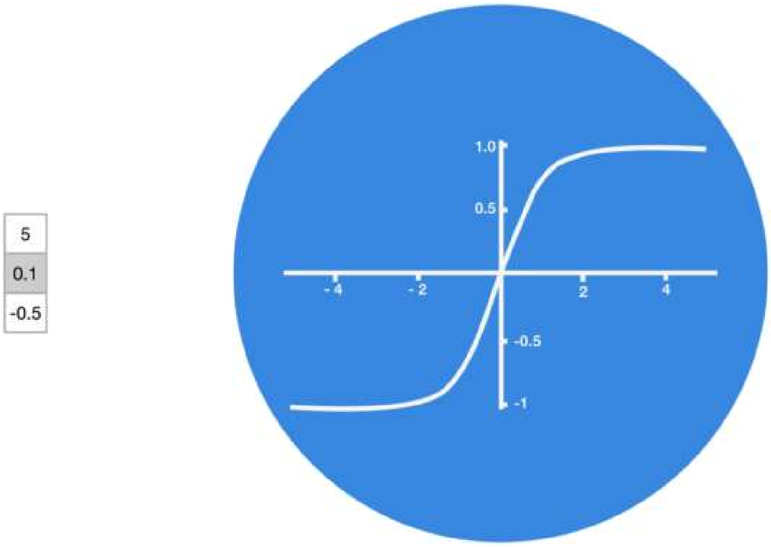
Let's look at a cell of the RNN to see how you would calculate the hidden state. First, the input and previous hidden state are combined to form a vector. That vector now has information on the current input and previous inputs. The vector goes through the tanh activation, and the output is the new hidden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



RNN Cell

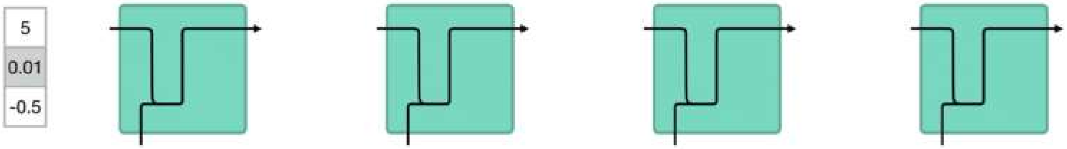
Tanh activation

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.



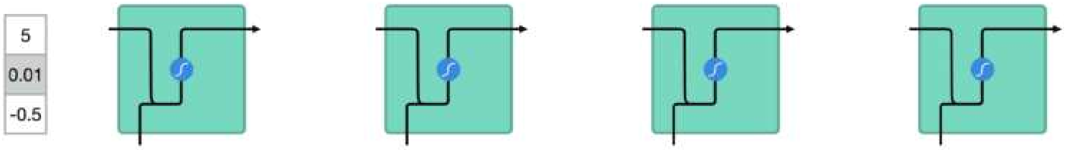
Tanh squishes values to be between -1 and 1

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.

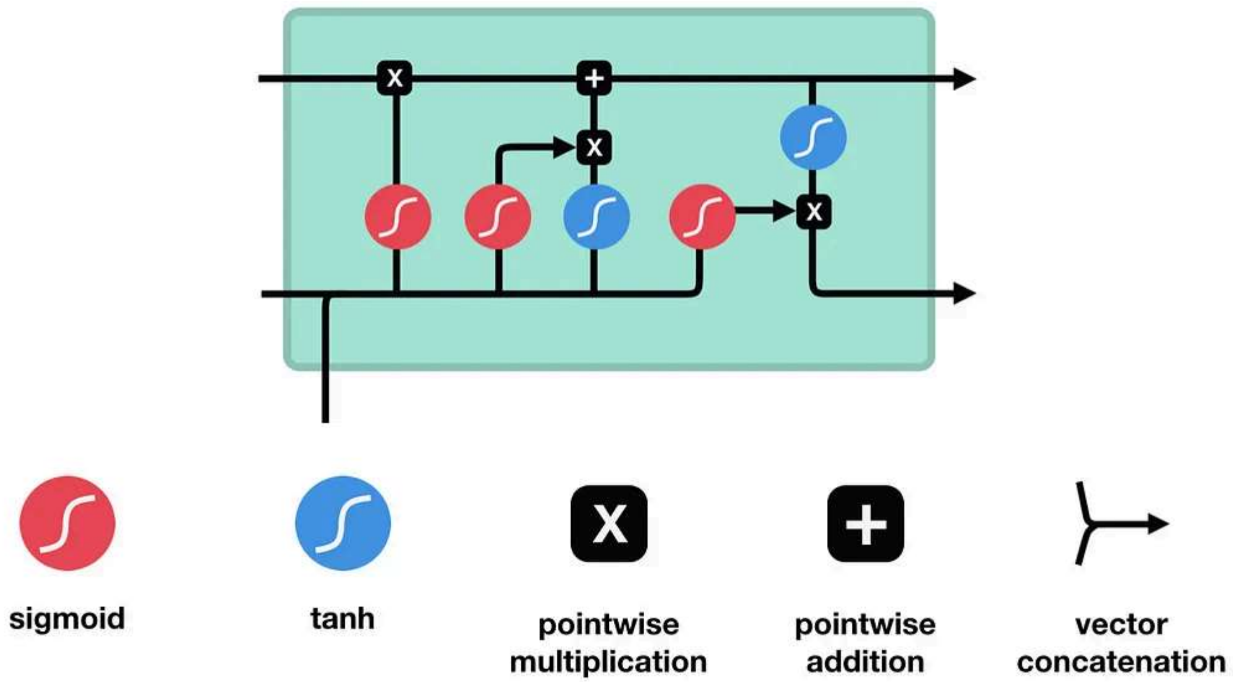


vector transformations with tanh

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

An LSTM has a similar control flow as a recurrent neural network. It processes data passing on information as it propagates forward. The differences are the operations within the LSTM’s cells.



LSTM Cell and It’s Operations

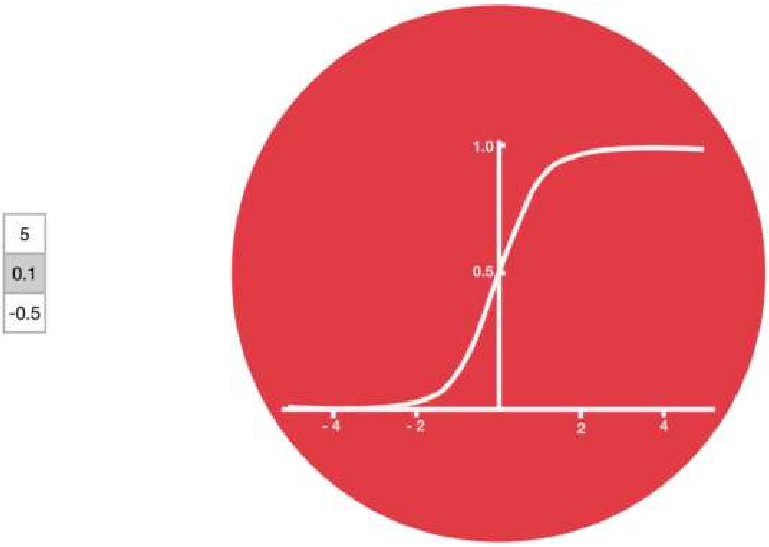
These operations are used to allow the LSTM to keep or forget information. Now looking at these operations can get a little overwhelming so we’ll go over this step by step.

Core Concept

The core concept of LSTM’s are the cell state, and it’s various gates. The cell state act as a transport highway that transfers relative information all the way down the sequence chain. You can think of it as the “memory” of the network. The cell state, in theory, can carry relevant information throughout the processing of the sequence. So even information from the earlier time steps can make it’s way to later time steps, reducing the effects of short-term memory. As the cell state goes on its journey, information get’s added or removed to the cell state via gates. The gates are different neural networks that decide which information is allowed on the cell state. The gates can learn what information is relevant to keep or forget during training.

Sigmoid

Gates contains sigmoid activations. A sigmoid activation is similar to the tanh activation. Instead of squishing values between -1 and 1, it squishes values between 0 and 1. That is helpful to update or forget data because any number getting multiplied by 0 is 0, causing values to disappears or be "forgotten." Any number multiplied by 1 is the same value therefore that value stay's the same or is "kept." The network can learn which data is not important therefore can be forgotten or which data is important to keep.

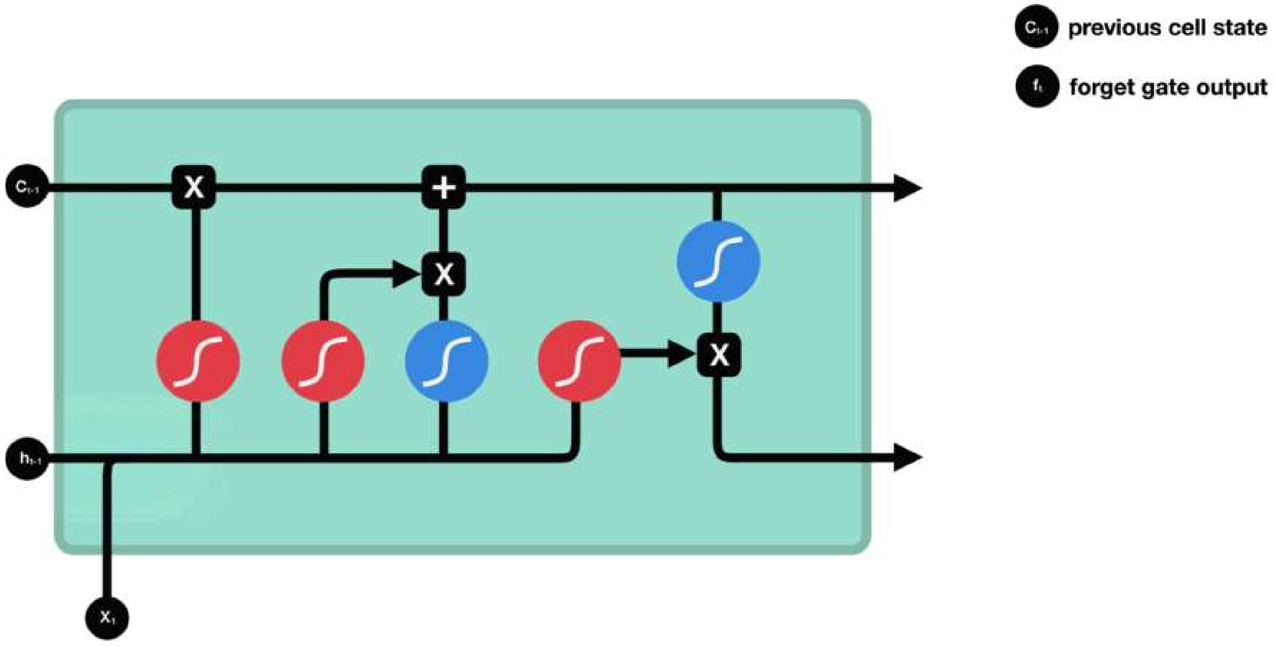


Sigmoid squishes values to be between 0 and 1

Let's dig a little deeper into what the various gates are doing, shall we? So we have three different gates that regulate information flow in an LSTM cell. A forget gate, input gate, and output gate.

Forget gate

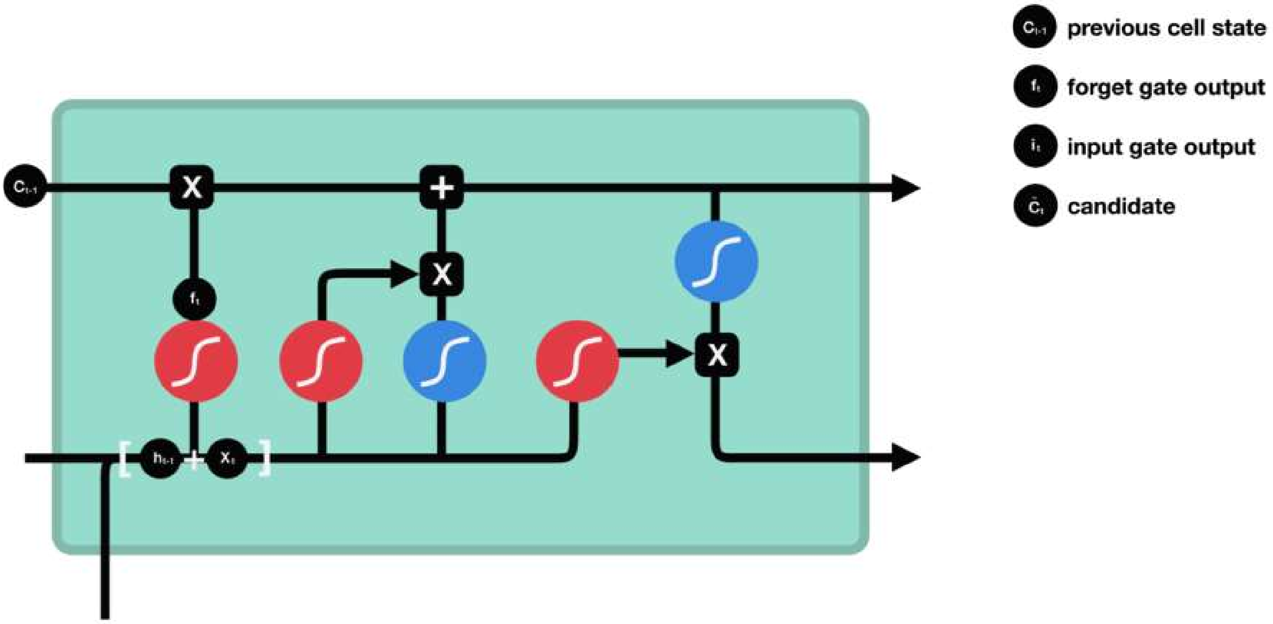
First, we have the forget gate. This gate decides what information should be thrown away or kept. Information from the previous hidden state and information from the current input is passed through the sigmoid function. Values come out between 0 and 1. The closer to 0 means to forget, and the closer to 1 means to keep.



Forget gate operations

Input Gate

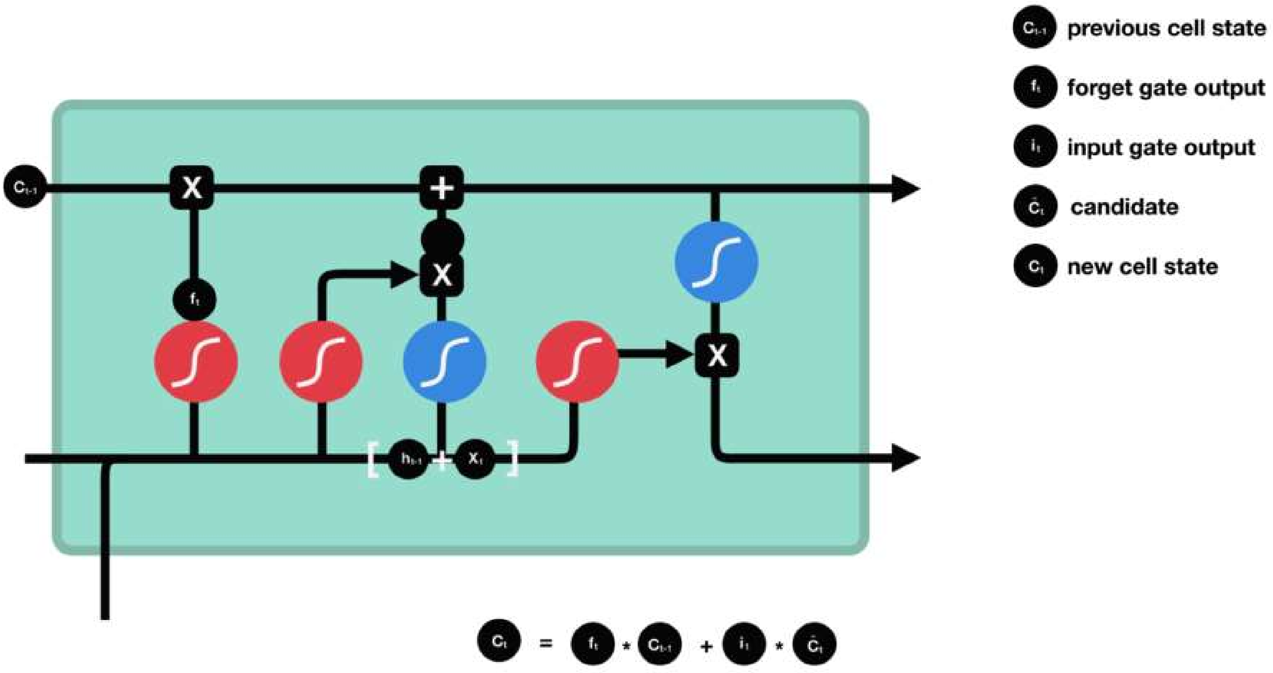
To update the cell state, we have the input gate. First, we pass the previous hidden state and current input into a sigmoid function. That decides which values will be updated by transforming the values to be between 0 and 1. 0 means not important, and 1 means important. You also pass the hidden state and current input into the tanh function to squish values between -1 and 1 to help regulate the network. Then you multiply the tanh output with the sigmoid output. The sigmoid output will decide which information is important to keep from the tanh output.



Input gate operations

Cell State

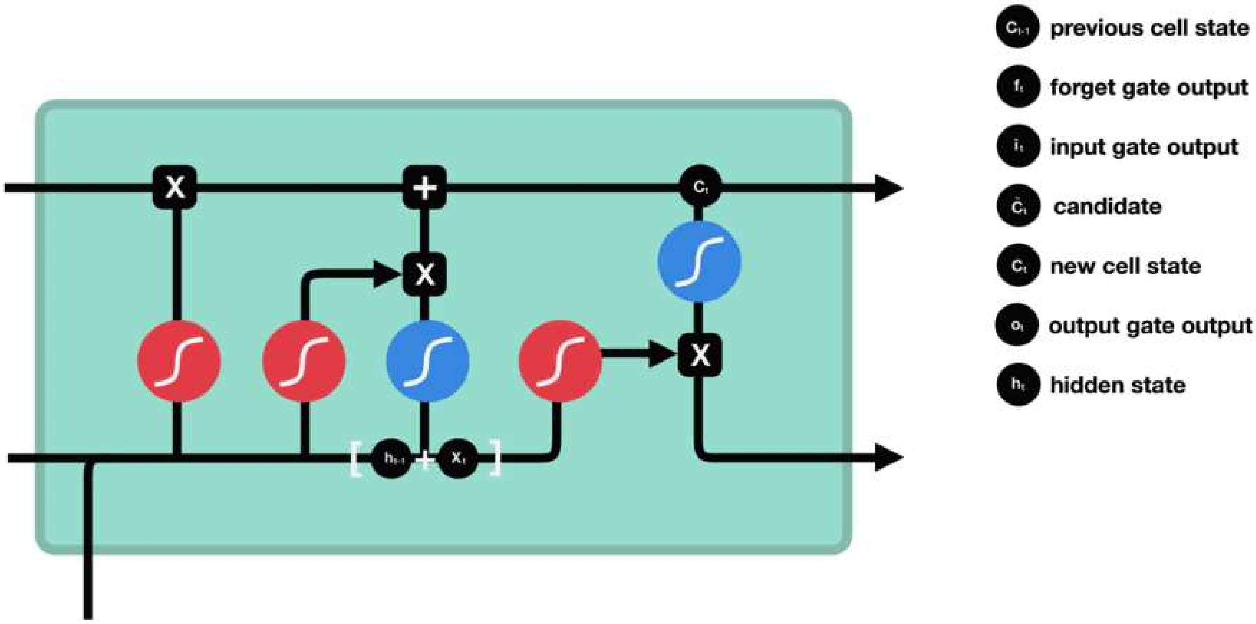
Now we should have enough information to calculate the cell state. First, the cell state gets pointwise multiplied by the forget vector. This has a possibility of dropping values in the cell state if it gets multiplied by values near 0. Then we take the output from the input gate and do a pointwise addition which updates the cell state to new values that the neural network finds relevant. That gives us our new cell state.



Calculating cell state

Output Gate

Last we have the output gate. The output gate decides what the next hidden state should be. Remember that the hidden state contains information on previous inputs. The hidden state is also used for predictions. First, we pass the previous hidden state and the current input into a sigmoid function. Then we pass the newly modified cell state to the tanh function. We multiply the tanh output with the sigmoid output to decide what information the hidden state should carry. The output is the hidden state. The new cell state and the new hidden is then carried over to the next time step.

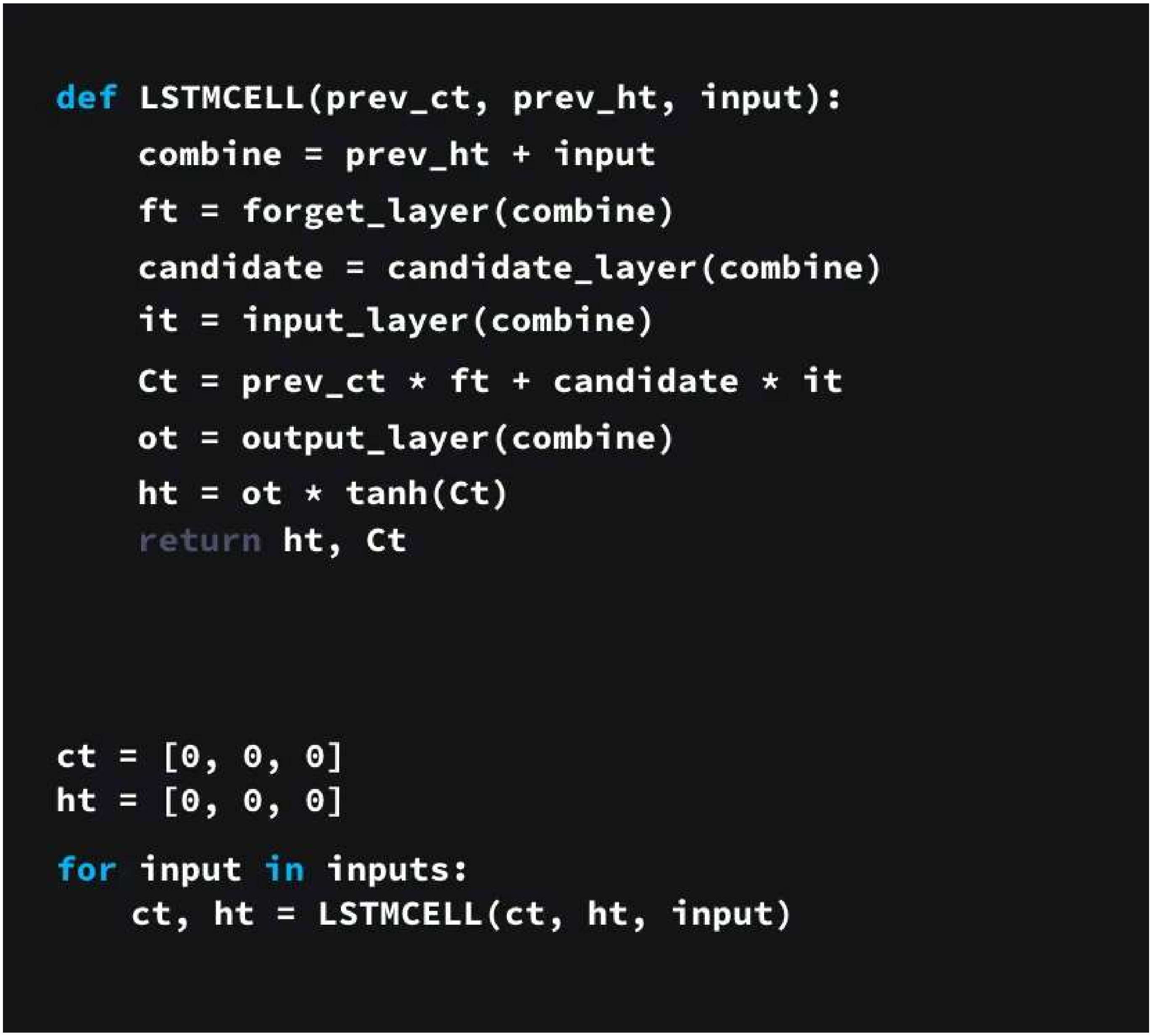


output gate operations

To review, the Forget gate decides what is relevant to keep from prior steps. The input gate decides what information is relevant to add from the current step. The output gate determines what the next hidden state should be.

Code Demo

For those of you who understand better through seeing the code, here is an example using python pseudo code.



```
def LSTMCELL(prev_ct, prev_ht, input):
    combine = prev_ht + input
    ft = forget_layer(combine)
    candidate = candidate_layer(combine)
    it = input_layer(combine)
    Ct = prev_ct * ft + candidate * it
    ot = output_layer(combine)
    ht = ot * tanh(Ct)
    return ht, Ct



ct = [0, 0, 0]
ht = [0, 0, 0]

for input in inputs:
    ct, ht = LSTMCELL(ct, ht, input)
```

python pseudo code

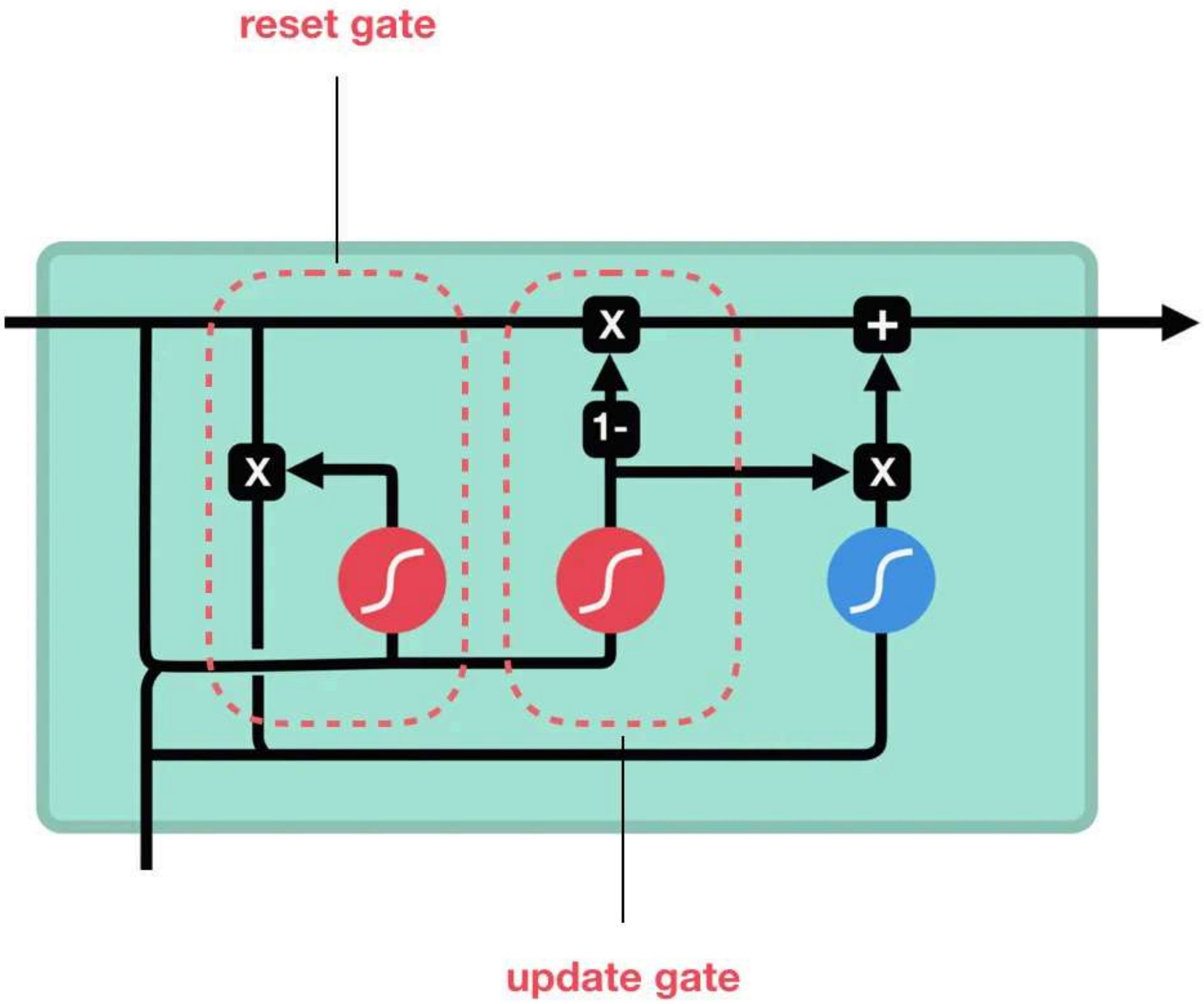
1. First, the previous hidden state and the current input get concatenated. We’ll call it *combine*.
2. *Combine* get’s fed into the forget layer. This layer removes non-relevant data.
4. A candidate layer is created using *combine*. The candidate holds possible values to add to the cell state.
3. *Combine* also get’s fed into the input layer. This layer decides what data from the candidate should be added to the new cell state.
5. After computing the forget layer, candidate layer, and the input layer, the

cell state is calculated using those vectors and the previous cell state.
6. The output is then computed.
7. Pointwise multiplying the output and the new cell state gives us the new hidden state.

That’s it! The control flow of an LSTM network are a few tensor operations and a for loop. You can use the hidden states for predictions. Combining all those mechanisms, an LSTM can choose which information is relevant to remember or forget during sequence processing.

GRU

So now we know how an LSTM work, let’s briefly look at the GRU. The GRU is the newer generation of Recurrent Neural networks and is pretty similar to an LSTM. GRU’s got rid of the cell state and used the hidden state to transfer information. It also only has two gates, a reset gate and update gate.



GRU cell and it's gates

Update Gate

The update gate acts similar to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add.

Reset Gate

The reset gate is another gate is used to decide how much past information to forget.

And that's a GRU. GRU's has fewer tensor operations; therefore, they are a little speedier to train then LSTM's. There isn't a clear winner which one is better. Researchers and engineers usually try both to determine which one works better for their use case.

So That’s it

To sum this up, RNN’s are good for processing sequence data for predictions but suffers from short-term memory. LSTM’s and GRU’s were created as a method to mitigate short-term memory using mechanisms called gates. Gates are just neural networks that regulate the flow of information flowing through the sequence chain. LSTM’s and GRU’s are used in state of the art deep learning applications like speech recognition, speech synthesis, natural language understanding, etc.

If you’re interested in going deeper, here are links of some fantastic resources that can give you a different perspective in understanding LSTM’s and GRU’s. This post was heavily inspired by them.

http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano

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

https://www.youtube.com/watch?v=WCUNPb-5EYI

I had a lot of fun making this post so let me know in the comments if this was helpful or what you would like to see in the next one. And as always, thanks for reading!

Check out michaelphi.com for more content like this.

✍🏾 Want more Content? Check out my blog at ***https://www.michaelphi.com***

📺 Like to watch project-based videos? Check out my **Youtube**!

🥇 Stay up to date on articles and videos by signing up for my **email newsletter**!

Machine Learning | Deep Learning | Artificial Intelligence | Lstm | Neural Networks

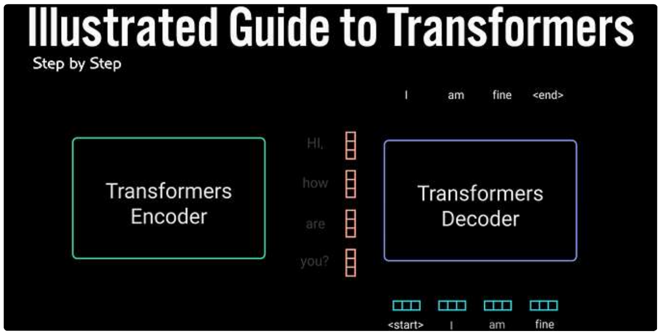
Written by Michael Phi

Follow

4.8K Followers · Writer for Towards Data Science

Software and Machine Learning Research Engineer. Here to make complex things simple. Follow me to stay on top of A.I. www.michaelphi.com

More from Michael Phi and Towards Data Science



Michael Phi in Towards Data Science

Illustrated Guide to Transformers- Step by Step Explanation

Transformers are taking the natural language processing world by storm. These incredible...

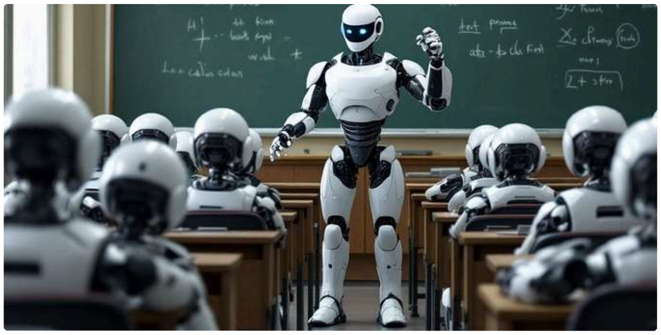
May 1, 2020 3.1K 21

Eric Colson in Towards Data Science

Beyond Skills: Unlocking the Full Potential of Data Scientists

Unlock the hidden value of data scientists by empowering them beyond technical tasks to...

5d ago 127 1

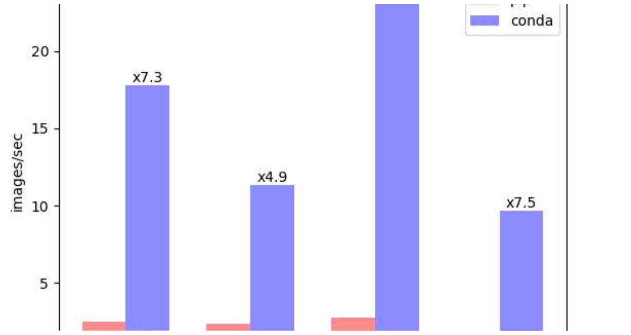


Daniel Kharitonov in Towards Data Science

Unsupervised LLM Evaluations

Practitioners guide to judging outputs of large language models

3d ago 156 1



Michael Phi in Towards Data Science

Stop Installing Tensorflow using pip for performance sake!

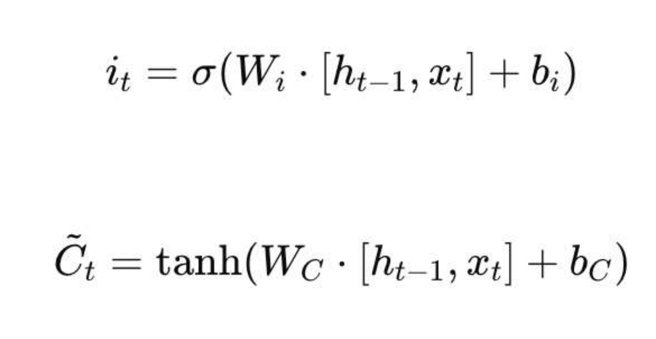
Get 8X the speed boost with the conda installation compared to the pip installation.

Oct 6, 2018 8.1K 50

See all from Michael Phi

See all from Towards Data Science

Recommended from Medium



$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

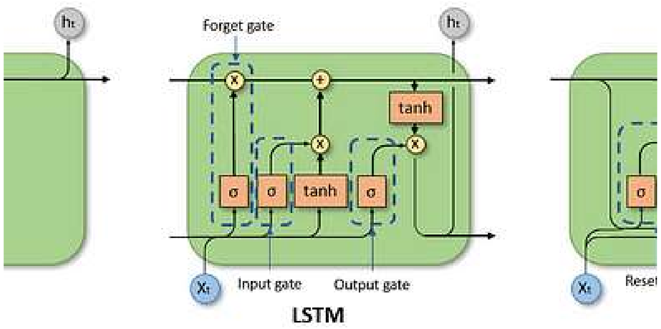
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

noplaxochia

LSTM from scratch

Using PyTorch

Jul 6 8



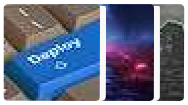
Jonte Dancker in Towards Data Science

A Brief Introduction to Recurrent Neural Networks

An introduction to RNN, LSTM, and GRU and their implementation

Dec 26, 2022 672 7

Lists

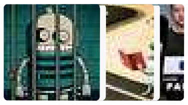


Predictive Modeling w/ Python

20 stories · 1632 saves

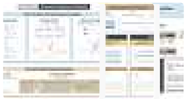
Natural Language Processing

1789 stories · 1400 saves



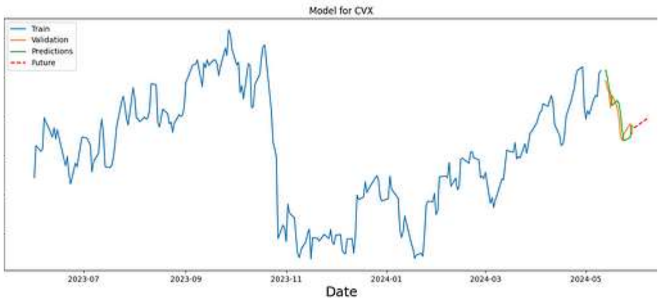
AI Regulation

6 stories · 604 saves



Practical Guides to Machine Learning

10 stories · 1998 saves



Sercan Bugra Gultekin

(1) Stock Price Prediction with ML in Python: LSTM (Long short-ter…

In this series, we will discuss how we can make predictions about stock prices with…

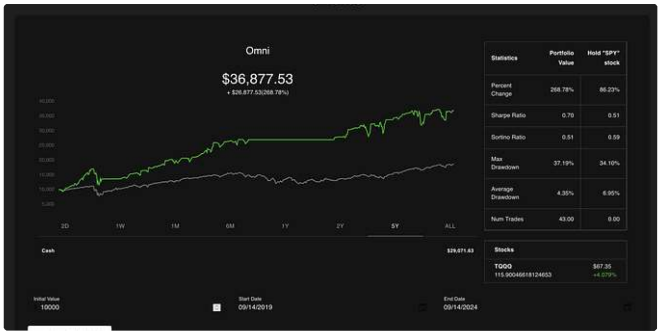
May 30 50

Jo Wang

Deep Learning Part 2—Neural Network and the critical Activatio…

Neural Network Structure

Jun 29 1



Omni

$36,877.53

+ $26,877.53(268.78%)

Statistics	Portfolio Value	Hold "SPY" stock
Percent Change	268.78%	86.23%
Sharpe Ratio	0.70	0.51
Sortino Ratio	0.51	0.59
Max Drawdown	37.19%	34.10%
Average Drawdown	4.35%	6.95%
Num Trades	43.00	0.00

Cash $29,071.63

Stocks

TQQQ 115.90046618124653 $67.35 +4.079%

Initial Value 10000

Start Date 09/14/2019

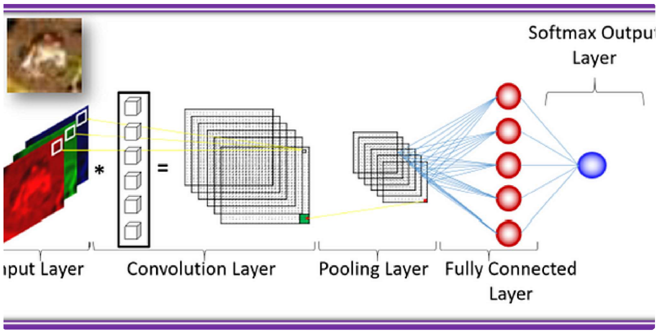
End Date 09/14/2024

Austin Starks in DataDrivenInvestor

I used OpenAI’s o1 model to develop a trading strategy. It is…

It literally took one try. I was shocked.

Sep 16 5.5K 139



Jyoti Dabass, Ph.D. in Python in Plain English

Friendly Introduction to Deep Learning Architectures (CNN, RN…

This blog aims to provide a friendly introduction to deep learning architectures…

Apr 1 1.3K 12

See more recommendations