4. Long Short-Term Memory RNNs (LSTMs)

- A type of RNN proposed by Hochreiter and Schmidhuber in 1997 as a solution to the vanishing gradients problem.
  - Everyone cites that paper but really a crucial part of the modern LSTM is from Gers et al. (2000) 💜

- On step *t*, there is a hidden state $\boldsymbol{h}^{(t)}$ *and* a cell state $\boldsymbol{c}^{(t)}$
  - Both are vectors length *n*
  - The cell stores long-term information
  - The LSTM can read, erase, and write information from the cell
    - The cell becomes conceptually rather like RAM in a computer

- The selection of which information is erased/written/read is controlled by three corresponding gates
  - The gates are also vectors length *n*
  - On each timestep, each element of the gates can be open (1), closed (0), or somewhere in-between
  - The gates are dynamic: their value is computed based on the current context

"Long short-term memory", Hochreiter and Schmidhuber, 1997. https://www.bioinf.jku.at/publications/older/2604.pdf
"Learning to Forget: Continual Prediction with LSTM", Gers, Schmidhuber, and Cummins, 2000. https://dl.acm.org/doi/10.1162/089976600300015015

Long Short-Term Memory- LSTM

- LSTM divide the context management problem into two sub-problems: removing information no longer needed from the context, and adding information likely to be needed for later decision making.
- **The key to solving both problems is to learn how to manage this context rather than hard-coding a strategy into the architecture.**
- LSTMs accomplish this by first adding an explicit context layer to the architecture (in addition to the usual recurrent hidden layer), and through the use of specialized neural units that make use of gates to control the flow of information into and out of the units that comprise the network layers.
- **These gates are implemented through the use of additional weights that operate sequentially on the input, and previous hidden layer, and previous context layers.**
- The gates in an LSTM share a common design pattern; each consists of a feedforward layer, followed by a sigmoid activation function, followed by a pointwise multiplication with the layer being gated.

Long Short-Term Memory (LSTM)

We have a sequence of inputs $x^{(t)}$, and we will compute a sequence of hidden states $h^{(t)}$ and cell states $c^{(t)}$. On timestep t:

Forget gate: controls what is kept vs forgotten, from previous cell state

Input gate: controls what parts of the new cell content are written to cell

Output gate: controls what parts of cell are output to hidden state

Sigmoid function: all gate values are between 0 and 1

$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{W}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f \boldsymbol{x}^{(t)} + \boldsymbol{b}_f\right)$$

$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{W}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i \boldsymbol{x}^{(t)} + \boldsymbol{b}_i\right)$$

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{W}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o \boldsymbol{x}^{(t)} + \boldsymbol{b}_o\right)$$

New cell content: this is the new content to be written to the cell

Cell state: erase ("forget") some content from last cell state, and write ("input") some new cell content

Hidden state: read ("output") some content from the cell

$$\tilde{\boldsymbol{c}}^{(t)} = \tanh\left(\boldsymbol{W}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_c \boldsymbol{x}^{(t)} + \boldsymbol{b}_c\right)$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)}$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \circ \tanh \boldsymbol{c}^{(t)}$$

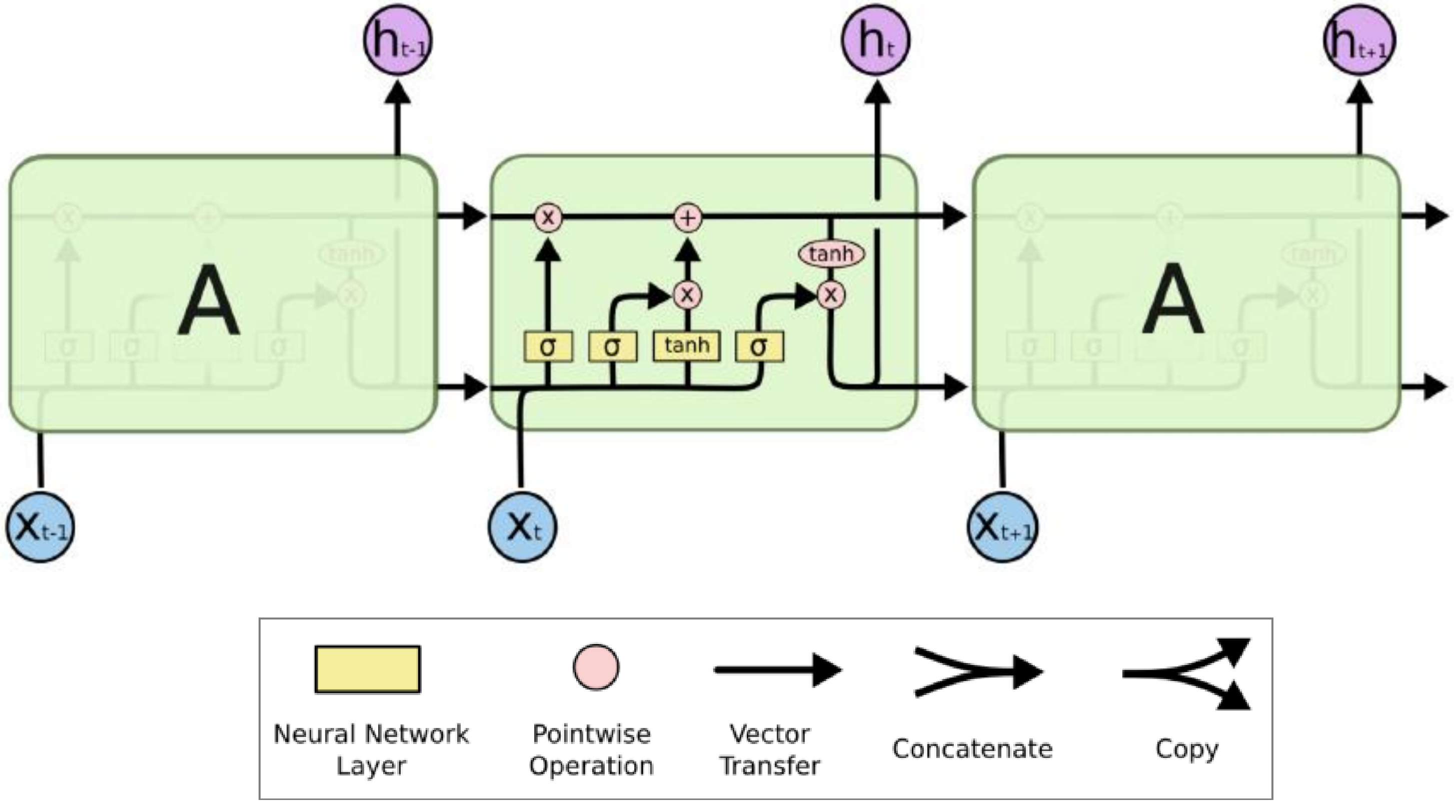
All these are vectors of same length n

Gates are applied using element-wise (or Hadamard) product: $\odot$

Long Short-Term Memory (LSTM)

A: a chunk of neural network

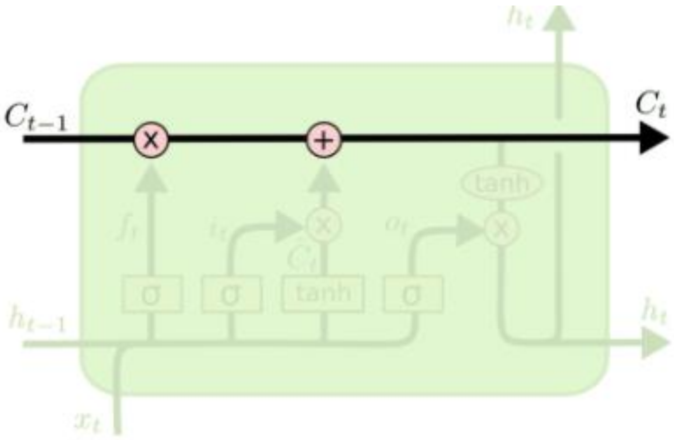
You can think of the LSTM equations visually like this:



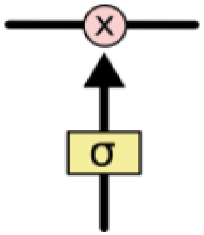
Source :http://colah.github.io/posts/2015-08-Understanding-LSTMs/

The Core Idea Behind LSTMs

- The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.
- The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.
- The LSTM does have 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.

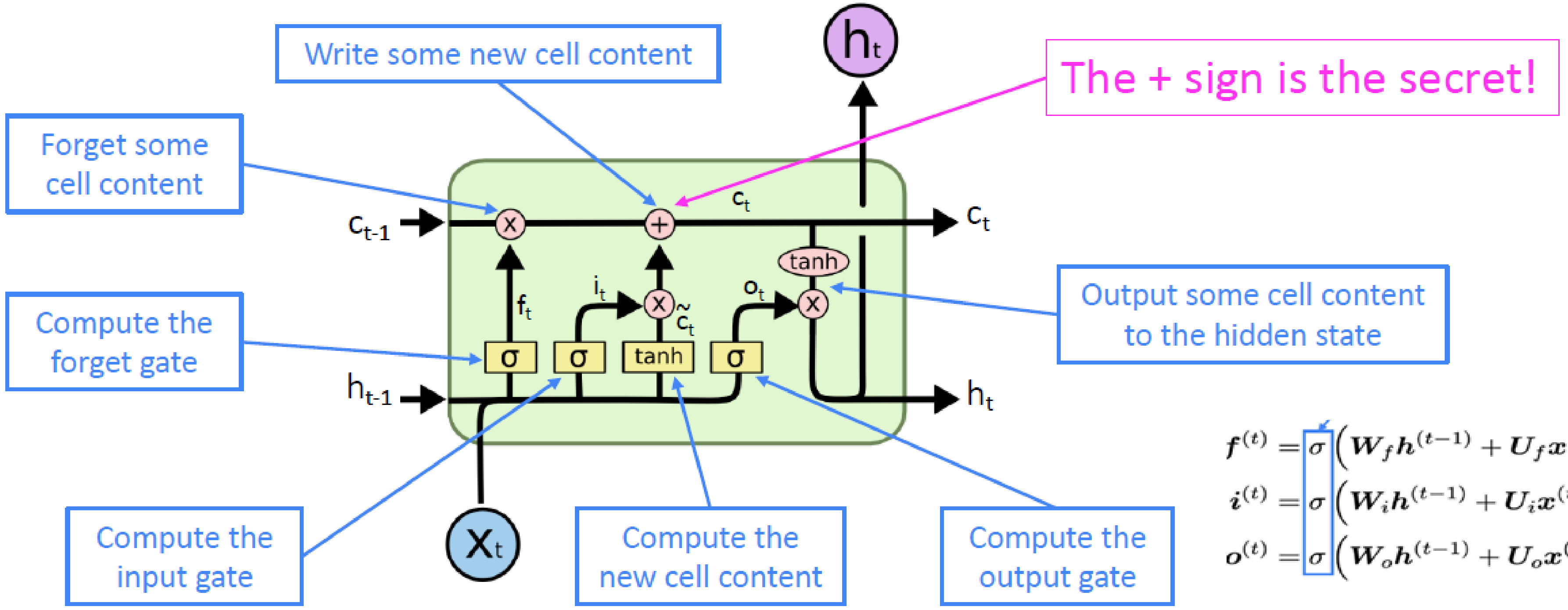


Gate



Long Short-Term Memory (LSTM)

You can think of the LSTM equations visually like this:



Write some new cell content

The + sign is the secret!

Forget some cell content

Compute the forget gate

Output some cell content to the hidden state

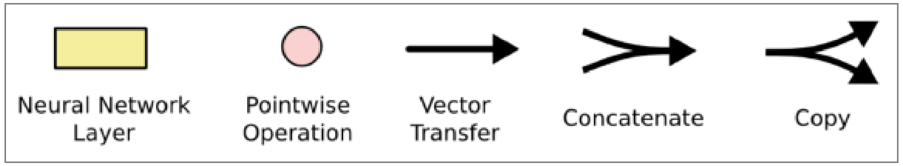
Compute the input gate

Compute the new cell content

Compute the output gate

$$
\begin{aligned}
\boldsymbol{f}^{(t)} &= \boxed{\sigma}\left(\boldsymbol{W}_f \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f \boldsymbol{x}^{(t)} + \boldsymbol{b}_f\right) \\
\boldsymbol{i}^{(t)} &= \boxed{\sigma}\left(\boldsymbol{W}_i \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i \boldsymbol{x}^{(t)} + \boldsymbol{b}_i\right) \\
\boldsymbol{o}^{(t)} &= \boxed{\sigma}\left(\boldsymbol{W}_o \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o \boldsymbol{x}^{(t)} + \boldsymbol{b}_o\right)
\end{aligned}
$$

$$
\begin{aligned}
\tilde{\boldsymbol{c}}^{(t)} &= \tanh\left(\boldsymbol{W}_c \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_c \boldsymbol{x}^{(t)} + \boldsymbol{b}_c\right. \\
\boldsymbol{c}^{(t)} &= \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)} \\
\boldsymbol{h}^{(t)} &= \boldsymbol{o}^{(t)} \circ \tanh \boldsymbol{c}^{(t)}
\end{aligned}
$$

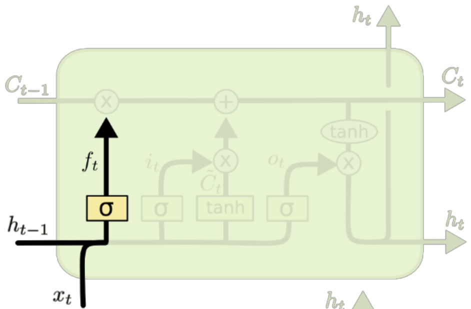


Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

How does LSTM solve vanishing gradients?

- The LSTM architecture makes it easier for the RNN to preserve information over many timesteps
  - e.g., if the forget gate is set to 1 for a cell dimension and the input gate set to 0, then the information of that cell is preserved indefinitely.
  - In contrast, it's harder for a vanilla RNN to learn a recurrent weight matrix W_h that preserves info in the hidden state
  - In practice, you get about 100 timesteps rather than about 7

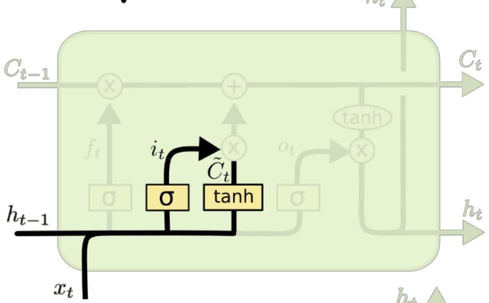
- LSTM doesn't *guarantee* that there is no vanishing/exploding gradient, but it does provide an easier way for the model to learn long-distance dependencies



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

ht-1 →Roufa and Manoj plays well in basket ball.
Xt →Manoj is really good at webdesigning.

- Forget gate realizes that there might be change in the context after encounter its first fullstop.
- Compare with Current Input **Xt.**
- Its important to know that next sentence, talks about Manoj. so information about Roufa is omited.



i_t decides what component to be updated.

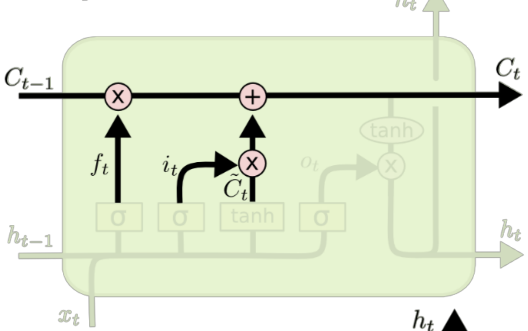
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

C'_t provides change contents

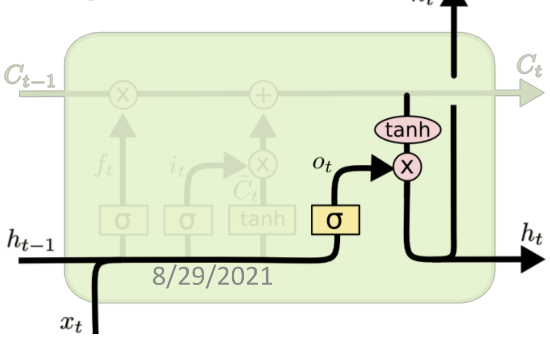
Manoj is good at webdesigining, yesterday he told me that he is a university topper.

- Input gate analysis the important information.
- “Manoj is good at webdesigining, he is university topper” is important.
- “yesterday he told me” that is not important, so forgotten.



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

Updating the cell state



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

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

Decide what part of the cell state to output

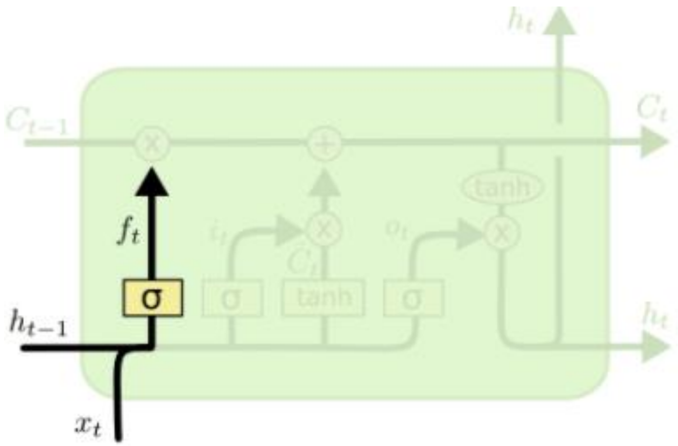
Manoj is good at webdesigining, he is the university topper. So the Merit student _________ was awarded University Gold medal

There could be lot of choices for the empty dash. this final gate replaces it with *Manoj.*

Step-by-Step LSTM Walk Through

- The first step in our LSTM is to decide what information we're going to throw away from the cell state.
- This decision is made by a sigmoid layer called the "forget gate layer."
- It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C$_{t-1}$.
- A 1 represents "completely keep this" while a 0 represents "completely get rid of this."
- Let's consider an example of a language model trying to predict the next word based on all the previous ones.
- In such a problem, the cell state might include the gender of the present subject, so that the correct pronouns can be used. When we see a new subject, we want to forget the gender of the old subject.

If we are trying to predict the last word in
Dan grew up in France. He speak fluent French.



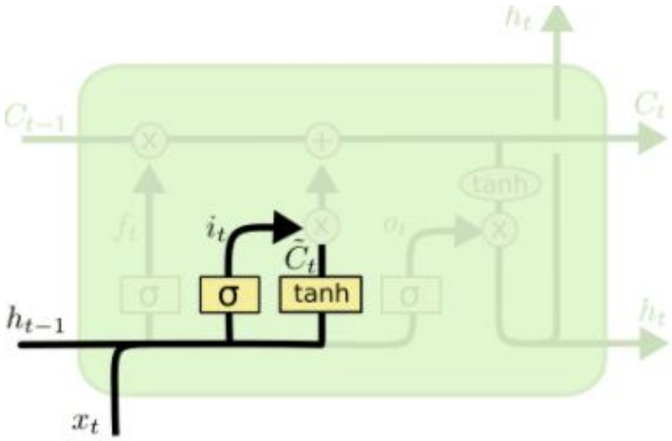
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Forget Gate

Cell State :
Previous information
Subject : Dan
Gender : Male
Inferred : Pronoun - He
Has to Forget the gender
New Subject identified: He

Step-by-Step LSTM Walk Through

- The next step is to decide what new information we’re going to store in the cell state.
- This has two parts.
- First, a sigmoid layer called the “input gate layer” decides which values we’ll update. To let through **0,1**
- Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. Range **-1** to **1**
- In the next step, we’ll combine these two to create an update to the state.
- In the example of our language model, we’d want to add the gender of the new subject to the cell state, to replace the old one we’re forgetting.



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

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

Input Gate

New value

Subject : He

Gender : Male

New Cell Content : Vector

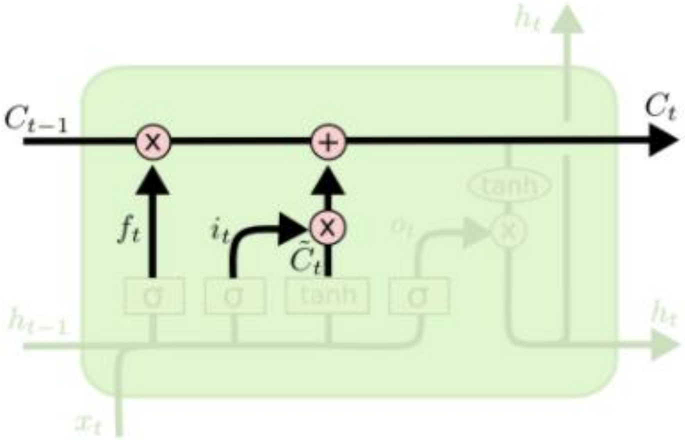
If we are trying to predict the last word in
“**Dan grew up in France. He speak fluent French.**”

Step-by-Step LSTM Walk Through

- It's now time to update the old cell state, C_{t-1}, into the new cell state C_t. The previous steps already decided what to do, we just need to actually do it.

- We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add $i_t * \tilde{C}_t$. This is the new candidate values, scaled by how much we decided to update each state value.

- In the case of the language model, this is where we'd actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps.



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

Cell State :
Drop Previous information
Subject : Dan
Gender : Male

Add the new information
New Subject : He
Gender : Male

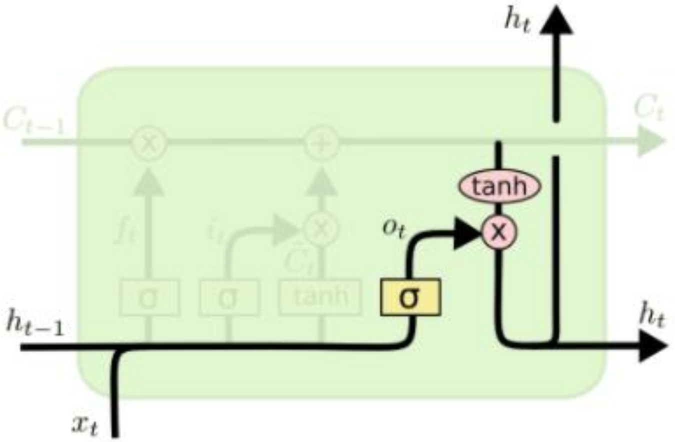
If we are trying to predict the last word in
"**Dan grew up in France. He speak fluent French.**"

Step-by-Step LSTM Walk Through

- Finally, we need to decide what we’re going to output.
- **This output will be based on our cell state, but will be a filtered version.**
- First, we run a sigmoid layer which decides what parts of the cell state we’re going to output.
- Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

- For the language model example, since it just saw a subject, it might want to output information relevant to a verb, in case that’s what is coming next.
- For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that’s what follows next. Subject-Verb agreement !

If we are trying to predict the last word in
“**Dan grew up in France. He speaks fluent French.**”



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

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

Output Gate

Output information
Subject : Singular

ht **Subject : Singular**

Hint for next state

LSTM Gates operation Examples

1. Forget Gate Taking the example of a text prediction problem. Let’s assume an LSTM is fed in, the following sentence:

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

As soon as the first full stop after “person” is encountered, the forget gate realizes that there may be a change of context in the next sentence. As a result of this, the subject of the sentence is forgotten and the place for the subject is vacated. And when we start speaking about “Dan” this position of the subject is allocated to “Dan”.
This process of forgetting the subject is brought about by the forget gate.

2. Input Gate Let’s take another example where the LSTM is analyzing a sentence:

Bob knows swimming. He told me over the phone that he had served the navy for 4 long years.

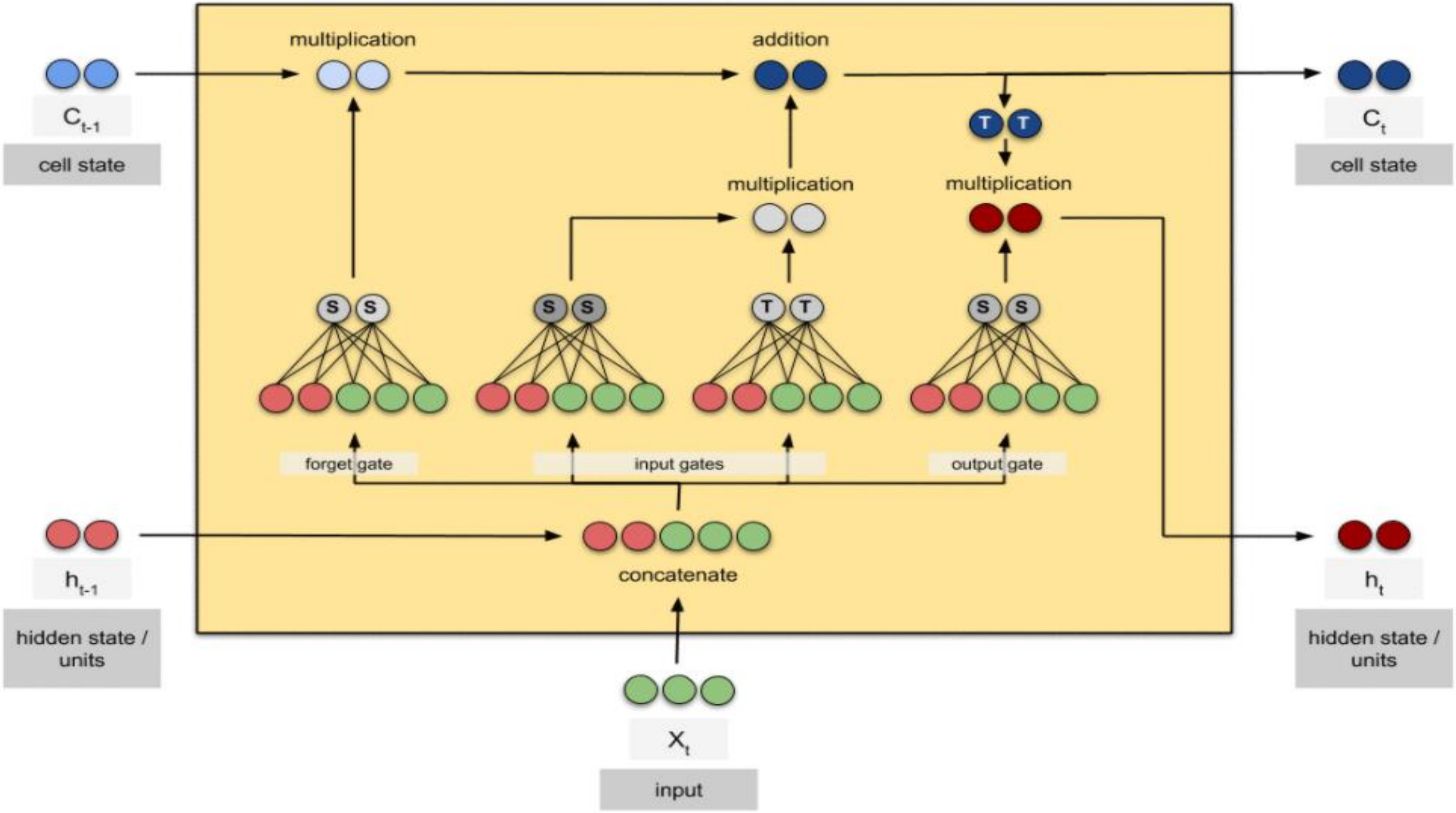
Now the important information here is that “Bob” knows swimming and that he has served the Navy for four years. This can be added to the cell state, however, the fact that he told all this over the phone is a less important fact and can be ignored.
This process of adding some new information can be done via the input gate.

3. Output Gate Not all information that runs along the cell state, is fit for being output at a certain time. An example:

Bob fought single handedly with the enemy and died for his country. For his contributions brave ______.

In this phrase, there could be a number of options for the empty space. But we know that the current input of ‘brave’, is an adjective that is used to describe a noun. Thus, whatever word follows, has a strong tendency of being a noun. And thus, Bob could be an apt output.
This job of selecting useful information from the current cell state and showing it out as an output is done via the output gate.

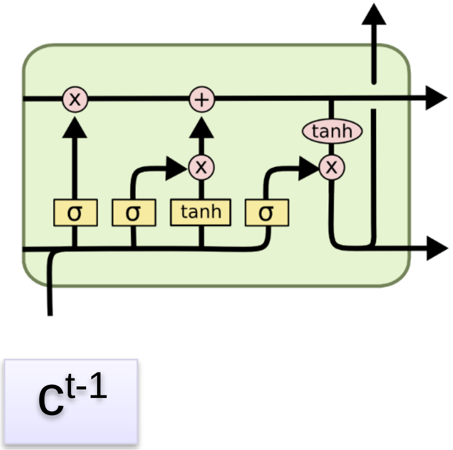
LSTM Gates operation

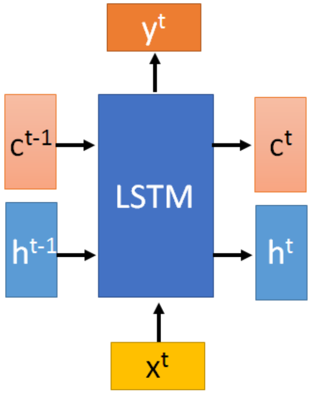


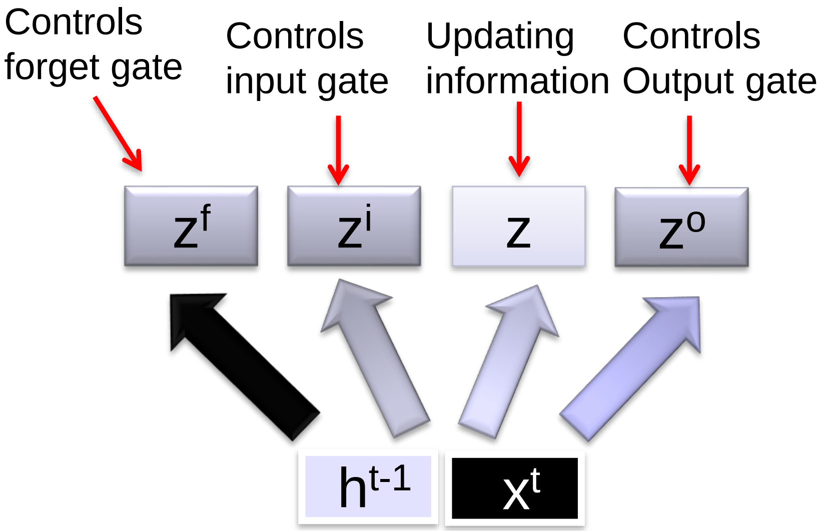
https://purnasaigudikandula.medium.com/recurrent-neural-networks-and-lstm-explained-7f51c7f6bbb9

Information flow of LSTM

These 4 matrix computation should be done concurrently.







Controls forget gate

Controls input gate

Updating information

Controls Output gate

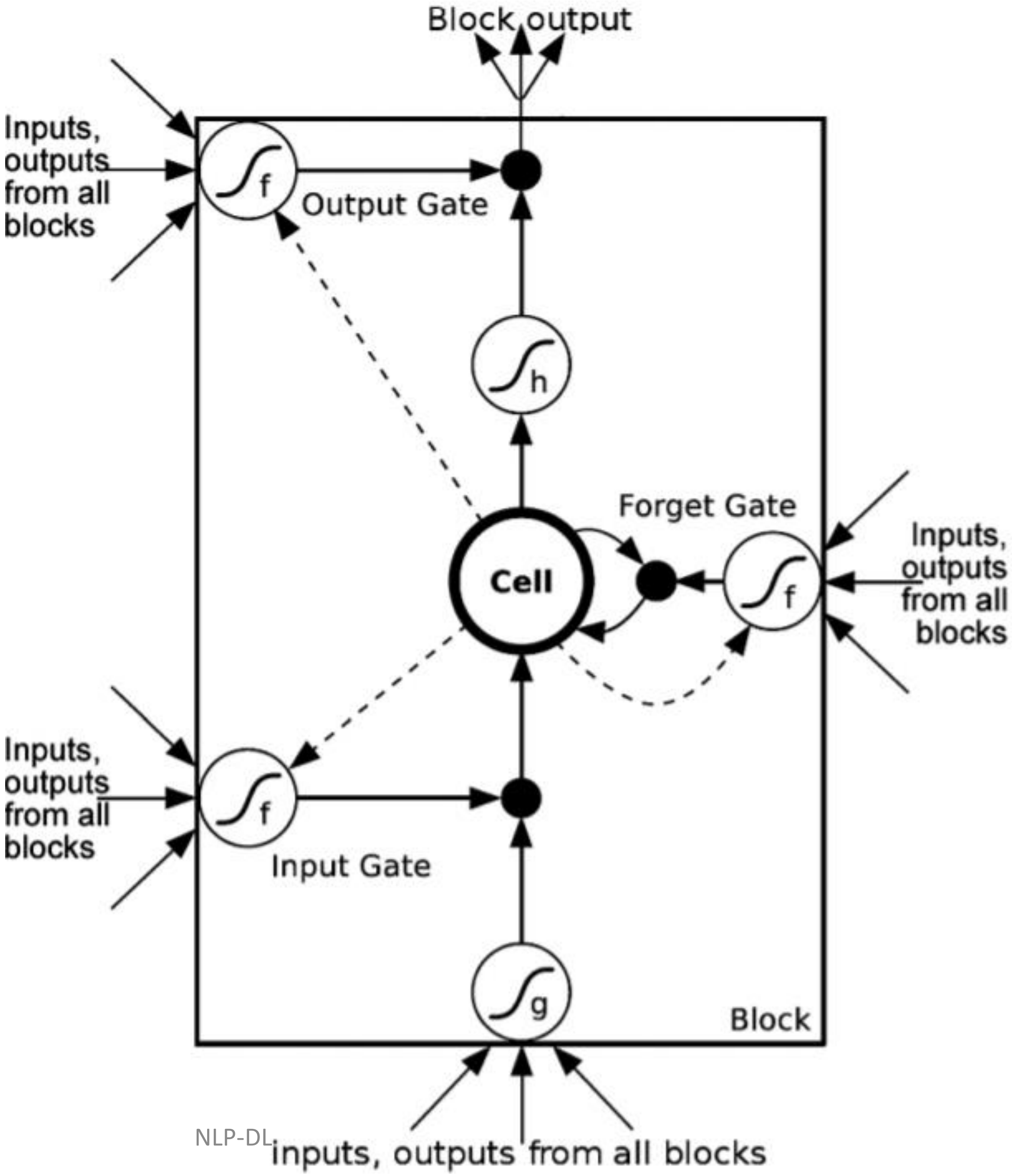
$$z = tanh(W \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix})$$

$$z^i = \sigma(W^i \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix})$$

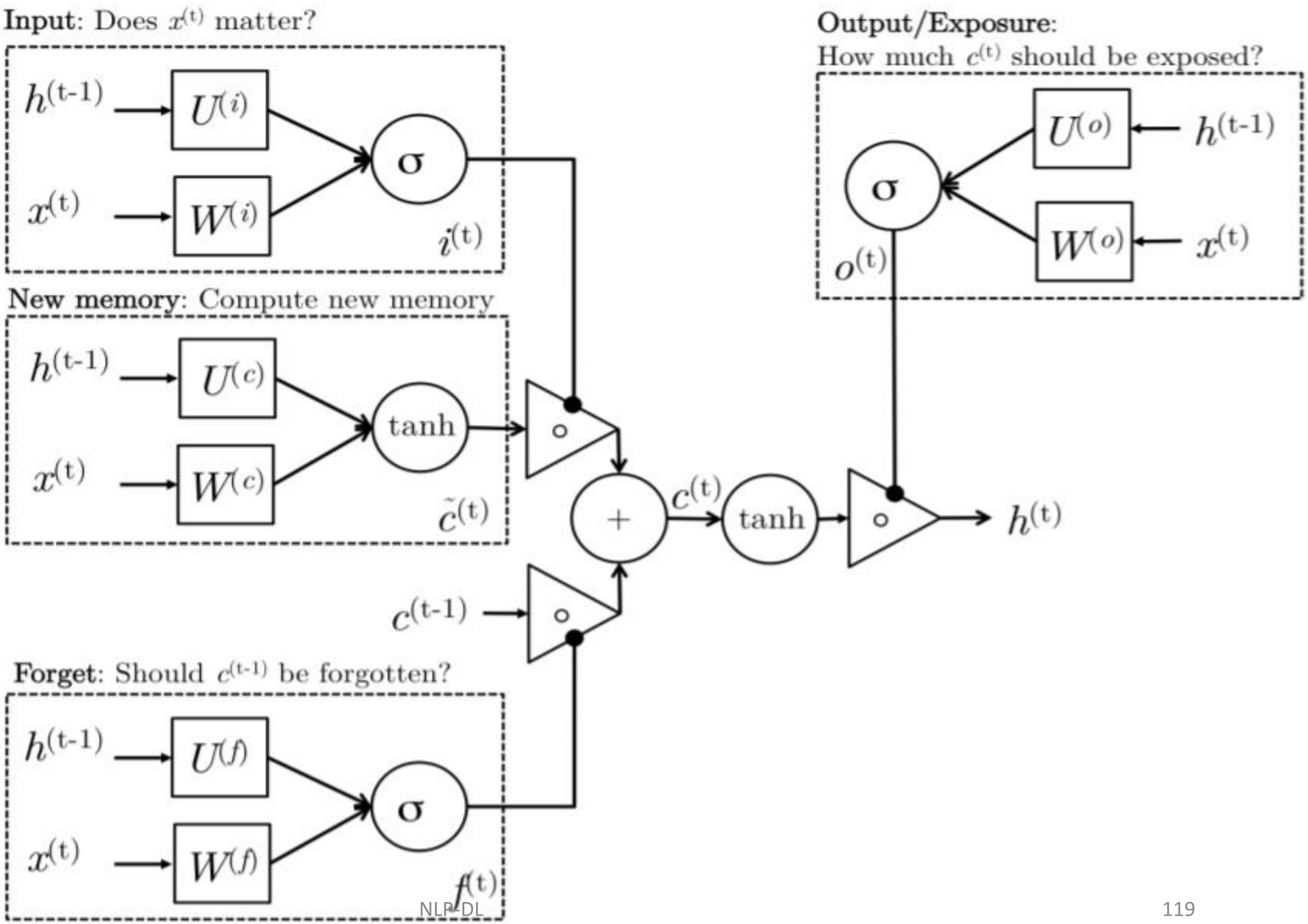
$$z^f = \sigma(W^f \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix})$$

$$z^o = \sigma(W^o \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix})$$

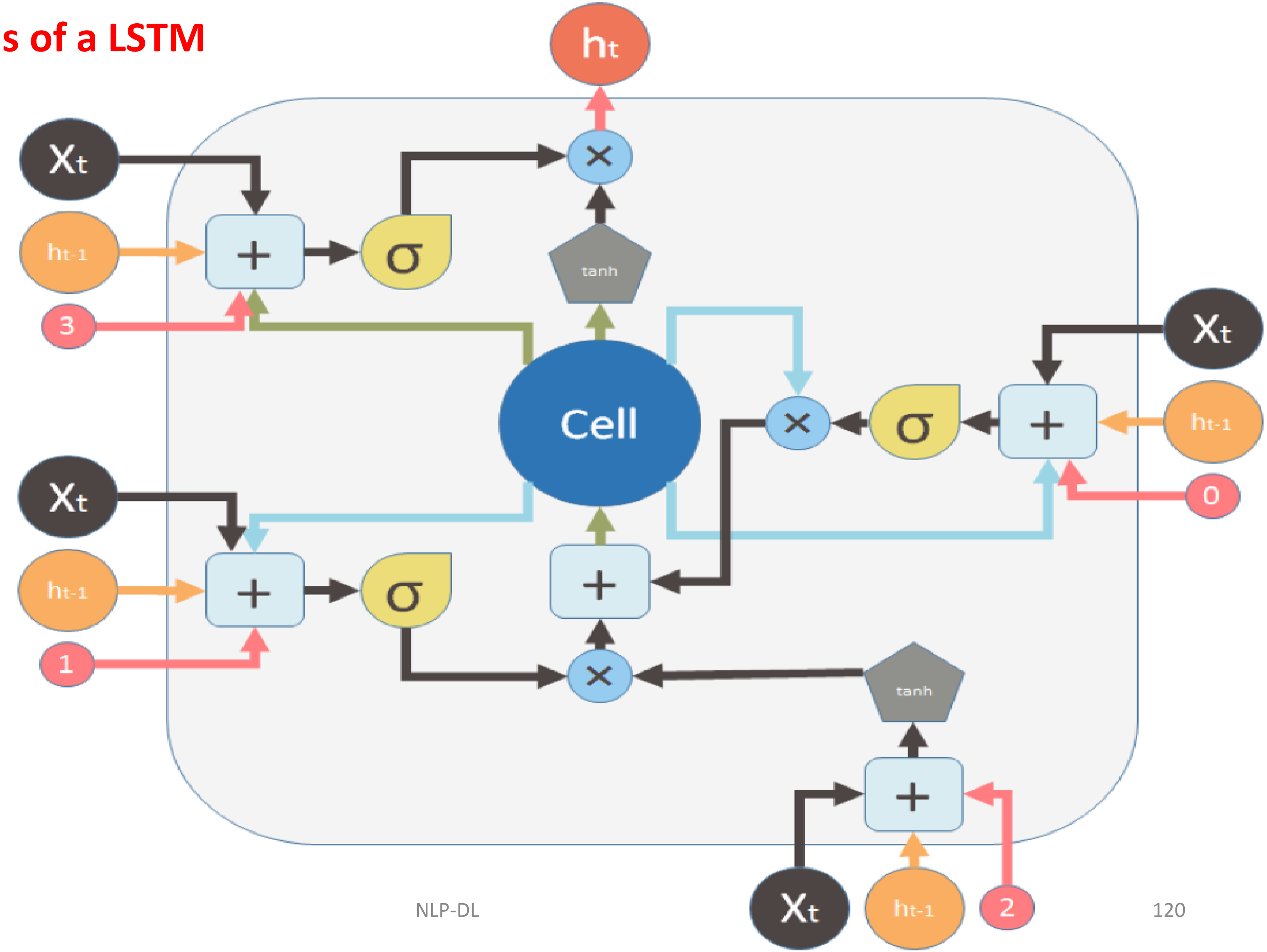
The LSTM unit


Block output
Inputs, outputs from all blocks
f
Output Gate
h
Forget Gate
Cell
Inputs, outputs from all blocks
f
Inputs, outputs from all blocks
f
Input Gate
g
Block
inputs, outputs from all blocks

The detailed internals of a LSTM



The detailed internals of a LSTM



LSTMs: real-world success

- In 2013–2015, LSTMs started achieving state-of-the-art results
  - Successful tasks include handwriting recognition, speech recognition, machine translation, parsing, and image captioning, as well as language models
  - LSTMs became the dominant approach for most NLP tasks

- Now (2021), other approaches (e.g., Transformers) have become dominant for many tasks
  - For example, in **WMT** (a Machine Translation conference + competition):
  - In WMT 2016, the summary report contains "RNN" 44 times
  - In WMT 2019: "RNN" 7 times, "Transformer" 105 times

Source: "Findings of the 2016 Conference on Machine Translation (WMT16)", Bojar et al. 2016, http://www.statmt.org/wmt16/pdf/W16-2301.pdf
Source: "Findings of the 2018 Conference on Machine Translation (WMT18)", Bojar et al. 2018, http://www.statmt.org/wmt18/pdf/WMT028.pdf
Source: "Findings of the 2019Conference on Machine Translation (WMT19)", Barrault et al. 2019, http://www.statmt.org/wmt18/pdf/WMT028.pdf