LSTMs

Feb 17, 2026

*Acknowledgment: Figures based on materials by CS224N @ Stanford University; Dr. Kilho Shin.

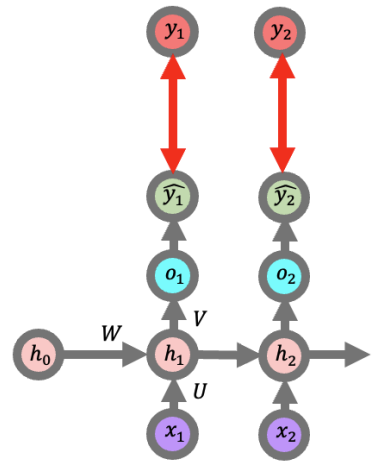
Outline

Outline

Problem with RNN 1: Vanishing gradient

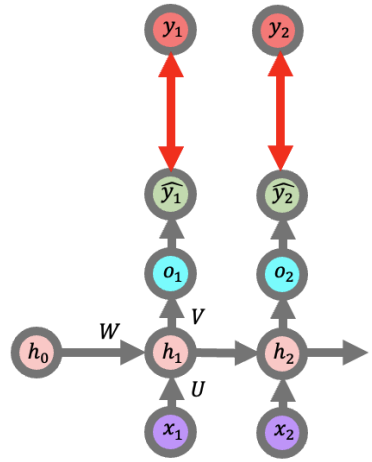
tldr: **If each step's gradient is too small**, multiplying across many steps makes it shrink exponentially. The overall gradient $\rightarrow 0$, so the model cannot learn long-range dependencies.

More explanation: long-term dependency



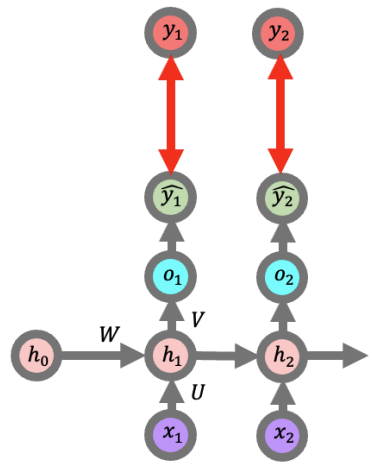
More explanation: long-term dependency and chain rule

$$\frac{\partial L_2}{\partial W}=\frac{\partial L_2}{\partial \widehat{y_2}}\frac{\partial \widehat{y_2}}{\partial o_2}\frac{\partial o_2}{\partial h_2}\frac{\partial h_2}{\partial W}+\frac{\partial L_2}{\partial \widehat{y_2}}\frac{\partial \widehat{y_2}}{\partial o_2}\frac{\partial o_2}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}$$



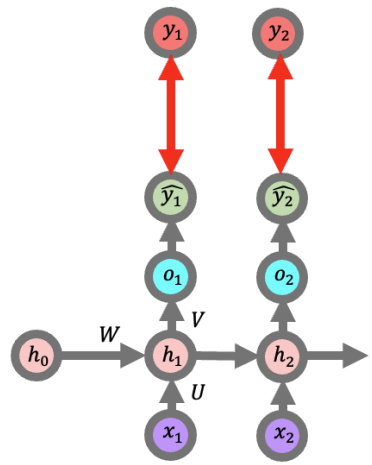
As we can see here, as time increases (as embedding nodes increase)

$$\frac{\partial L_2}{\partial W} = \frac{\partial L_2}{\partial \widehat{y_2}} \frac{\partial \widehat{y_2}}{\partial o_2} \frac{\partial o_2}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L_2}{\partial \widehat{y_2}} \frac{\partial \widehat{y_2}}{\partial o_2} \frac{\partial o_2}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W}$$



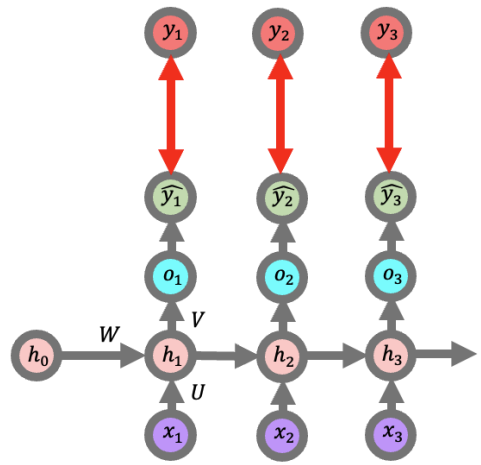
As we can see here, as time increases (as embedding nodes increase)

$$\frac{\partial L_2}{\partial W} = \frac{\partial L_2}{\partial \widehat{y_2}} \frac{\partial \widehat{y_2}}{\partial o_2} \frac{\partial o_2}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L_2}{\partial \widehat{y_2}} \frac{\partial \widehat{y_2}}{\partial o_2} \frac{\partial o_2}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W}$$



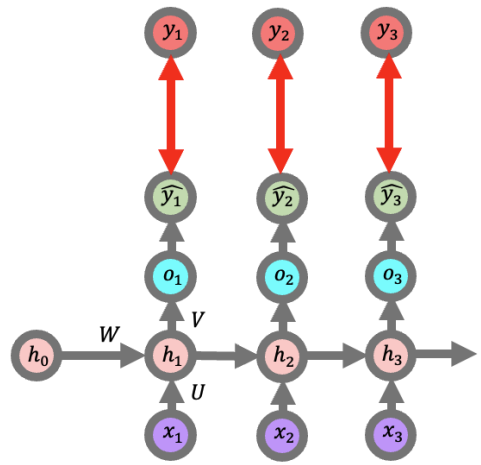
As we can see here, as time increases (as embedding nodes increase), the part that needs to be calculated by the chain rule

$$\frac{\partial L_3}{\partial W}=\frac{\partial L_3}{\partial \widehat{y_3}}\frac{\partial \widehat{y_3}}{\partial o_3}\frac{\partial o_3}{\partial h_3}\frac{\partial h_3}{\partial W}+\frac{\partial L_3}{\partial \widehat{y_3}}\frac{\partial \widehat{y_3}}{\partial o_3}\frac{\partial o_3}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W}+\frac{\partial L_3}{\partial \widehat{y_3}}\frac{\partial \widehat{y_3}}{\partial o_3}\frac{\partial o_3}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}$$



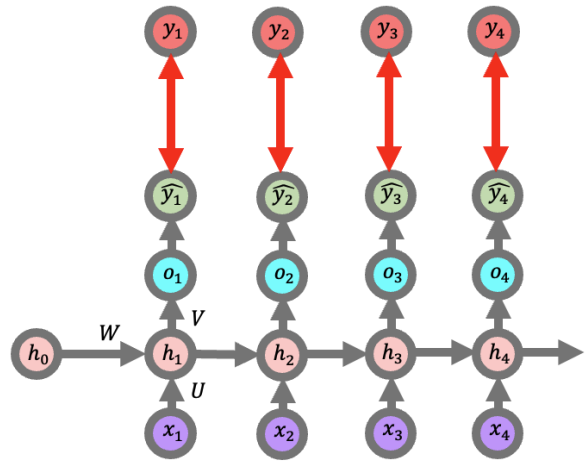
As we can see here, as time increases (as embedding nodes increase), the part that needs to be calculated by the chain rule

$$\frac{\partial L_3}{\partial W}=\frac{\partial L_3}{\partial \widehat{y_3}}\frac{\partial \widehat{y_3}}{\partial o_3}\frac{\partial o_3}{\partial h_3}\frac{\partial h_3}{\partial W}+\frac{\partial L_3}{\partial \widehat{y_3}}\frac{\partial \widehat{y_3}}{\partial o_3}\frac{\partial o_3}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W}+\frac{\partial L_3}{\partial \widehat{y_3}}\frac{\partial \widehat{y_3}}{\partial o_3}\frac{\partial o_3}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}$$



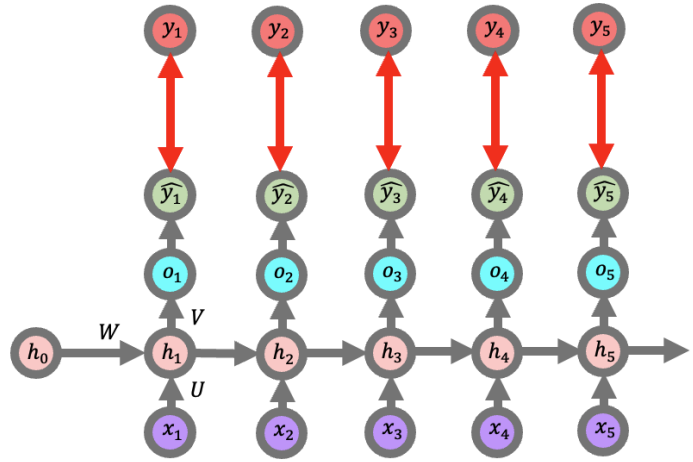
As we can see here, as time increases (as embedding nodes increase), the part that needs to be calculated by the chain rule keeps

$$\frac{\partial L_4}{\partial W}=\frac{\partial L_4}{\partial \widehat{y_4}}\frac{\partial \widehat{y_4}}{\partial o_4}\frac{\partial o_4}{\partial h_4}\frac{\partial h_4}{\partial W}+\frac{\partial L_4}{\partial \widehat{y_4}}\frac{\partial \widehat{y_4}}{\partial o_4}\frac{\partial o_4}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial W}+\frac{\partial L_4}{\partial \widehat{y_4}}\frac{\partial \widehat{y_4}}{\partial o_4}\frac{\partial o_4}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W}$$
$$+\frac{\partial L_4}{\partial \widehat{y_4}}\frac{\partial \widehat{y_4}}{\partial o_4}\frac{\partial o_4}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}$$



As we can see here, as time increases (as embedding nodes increase), the part that needs to be calculated by the chain rule keeps increasing

$$\frac{\partial L_5}{\partial W}=\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial W}$$
$$+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial W}$$

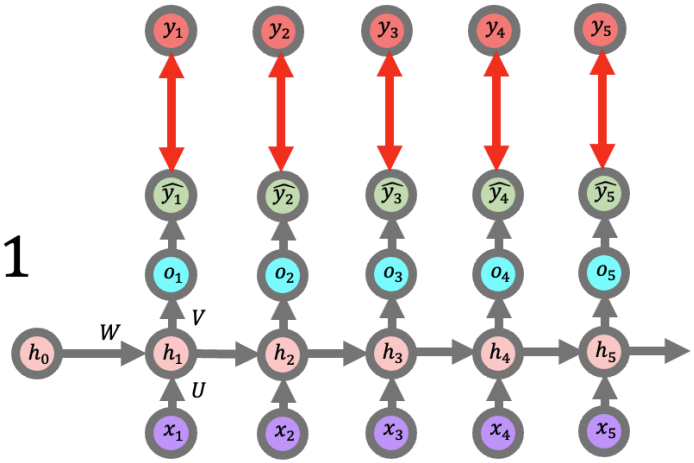


If these parts are smaller than 1

$$\frac{\partial L_5}{\partial W}=\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial W}$$

$$+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial W}+\frac{\partial L_5}{\partial \widehat{y_5}}\frac{\partial \widehat{y_5}}{\partial o_5}\frac{\partial o_5}{\partial h_5}\frac{\partial h_5}{\partial h_4}\frac{\partial h_4}{\partial h_3}\frac{\partial h_3}{\partial h_2}\frac{\partial h_1}{\partial W}$$

< 1

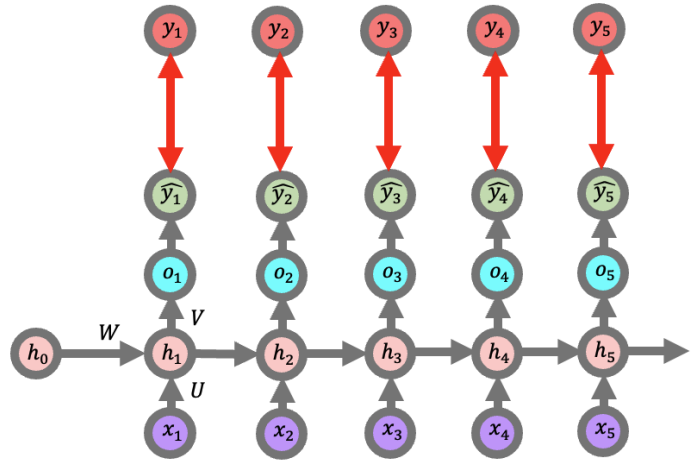


Then, as we keep multiplying through the chain rule, the gradient value for distant parts becomes smaller

$$\frac{\partial L_5}{\partial W} = \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial W} + \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial W} + \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial h_3} \frac{\partial h_3}{\partial W}$$

$$+ \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W}$$

e.g., 0.1x0.3x0.2x0.1 = 0.0006

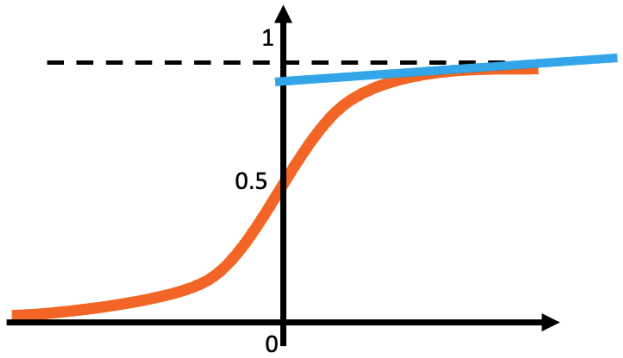


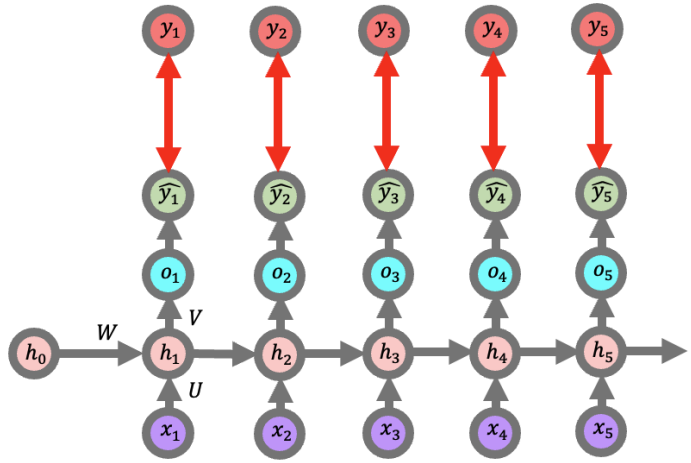
A smaller gradient means that its effect on learning is negligible,

$$\frac{\partial L_5}{\partial W} = \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial W} + \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial W} + \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial h_3} \frac{\partial h_3}{\partial W}$$

$$+ \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W}$$

e.g., 0.1x0.3x0.2x0.1 = 0.0006



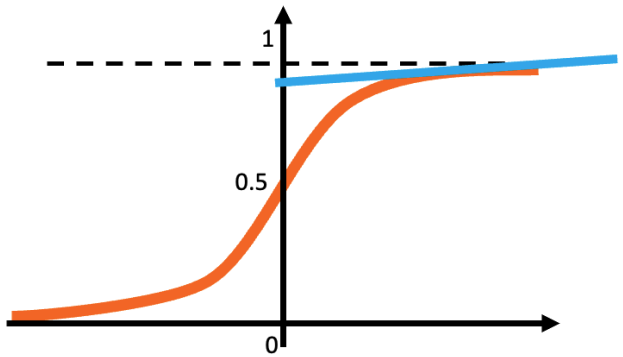


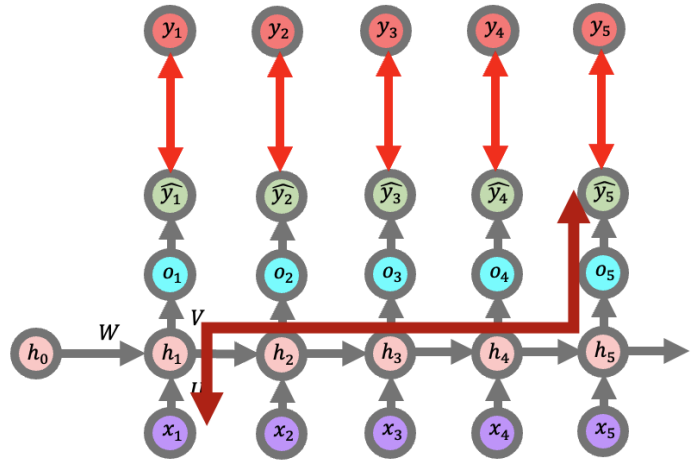
As a result, the farther back in time the input is, the smaller its effect on learning becomes

$$\frac{\partial L_5}{\partial W} = \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial W} + \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial W} + \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial h_3} \frac{\partial h_3}{\partial W}$$

$$+ \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial W} + \frac{\partial L_5}{\partial \widehat{y_5}} \frac{\partial \widehat{y_5}}{\partial o_5} \frac{\partial o_5}{\partial h_5} \frac{\partial h_5}{\partial h_4} \frac{\partial h_4}{\partial h_3} \frac{\partial h_3}{\partial h_2} \frac{\partial h_2}{\partial h_1} \frac{\partial h_1}{\partial W}$$

e.g., 0.1x0.3x0.2x0.1 = 0.0006





Problem with RNN 2: Exploding gradient

- When gradients become very large:
  - A single update step can overshoot the minimum
  - and destabilize or even blow up the model..!

Problems: Summary

- **Parameter sharing:** the same weight matrices are multiplied at each time step.
- If each multiplication factor:
  - is < 1, gradients shrink exponentially (vanishing).
  - is > 1, gradients grow exponentially (exploding).

Vanishing problem: Solution

Solutions explored:

- Separate memory cell (e.g., **LSTM**) with gating mechanisms to add/erase information.

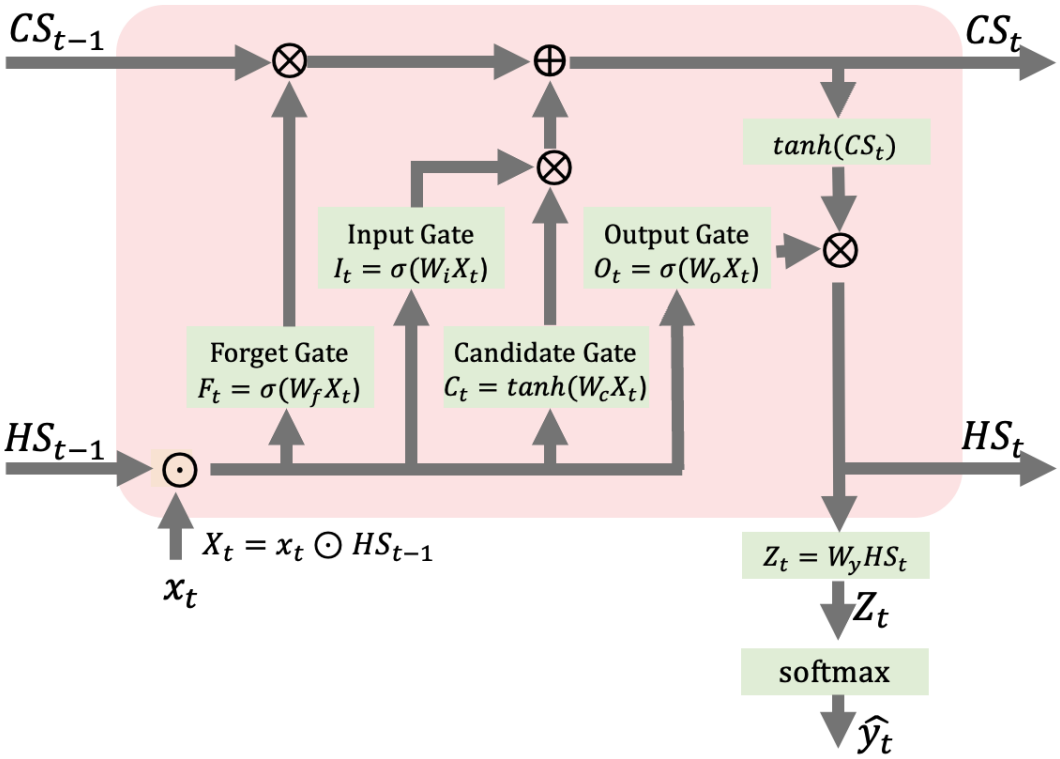
Outline

Overview

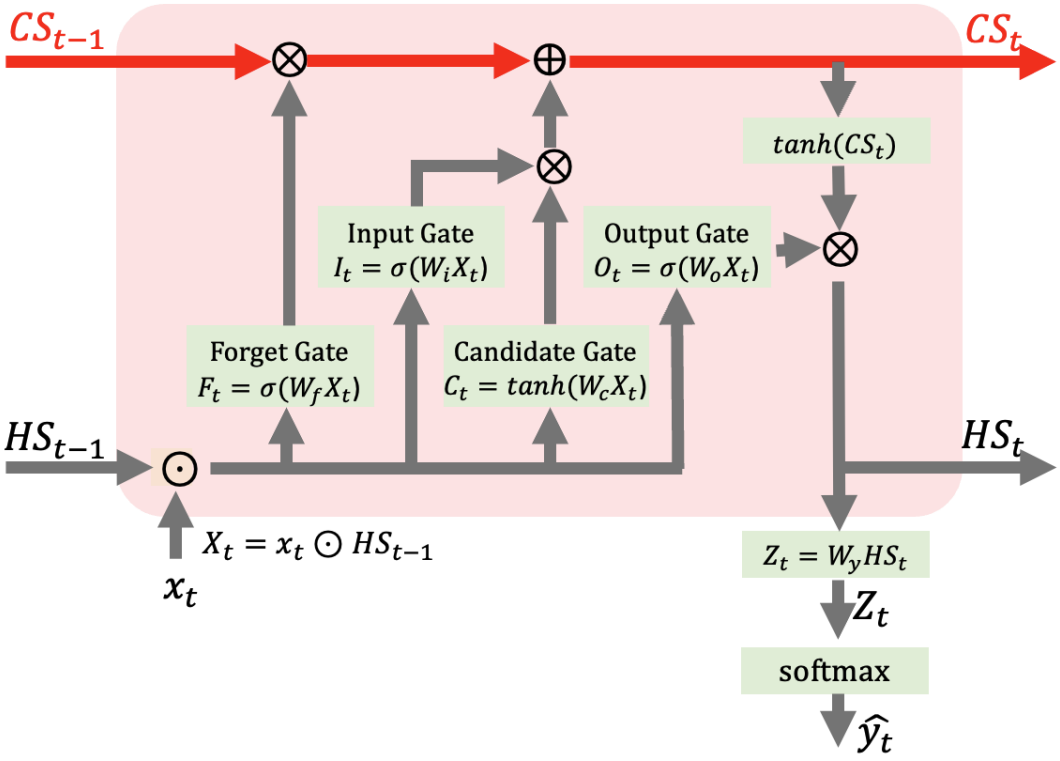
Separate memory cell with gating mechanisms to add/erase information.

1. Structure

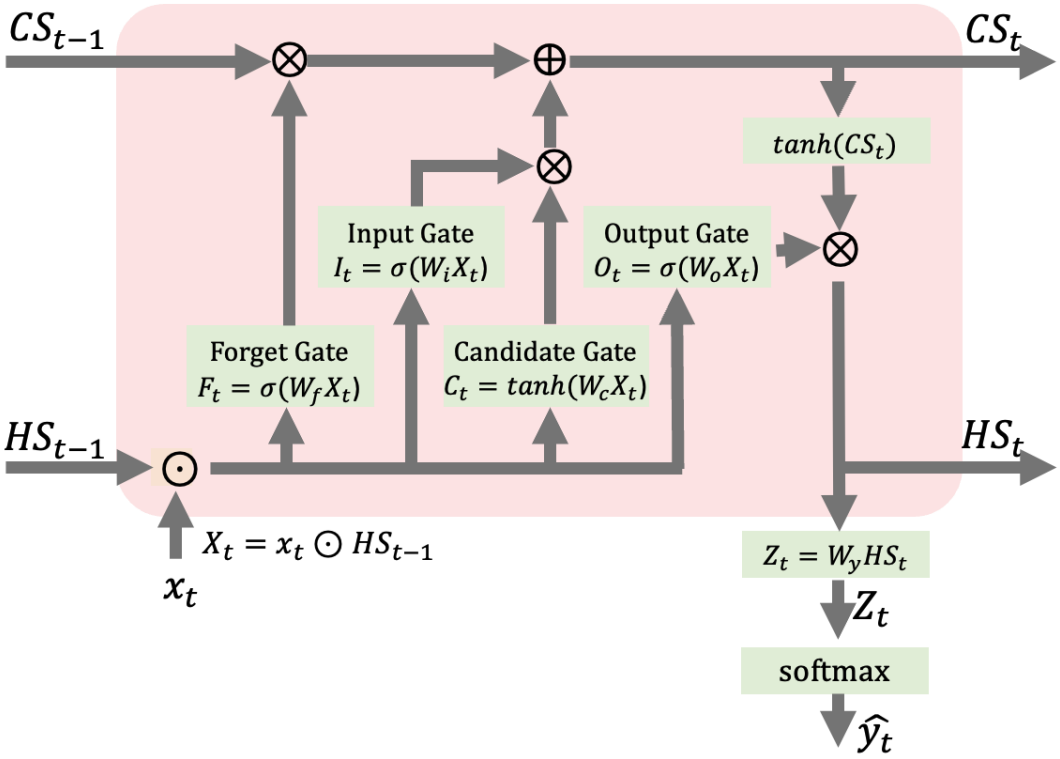
Let's understand LSTM's separate memory cell.



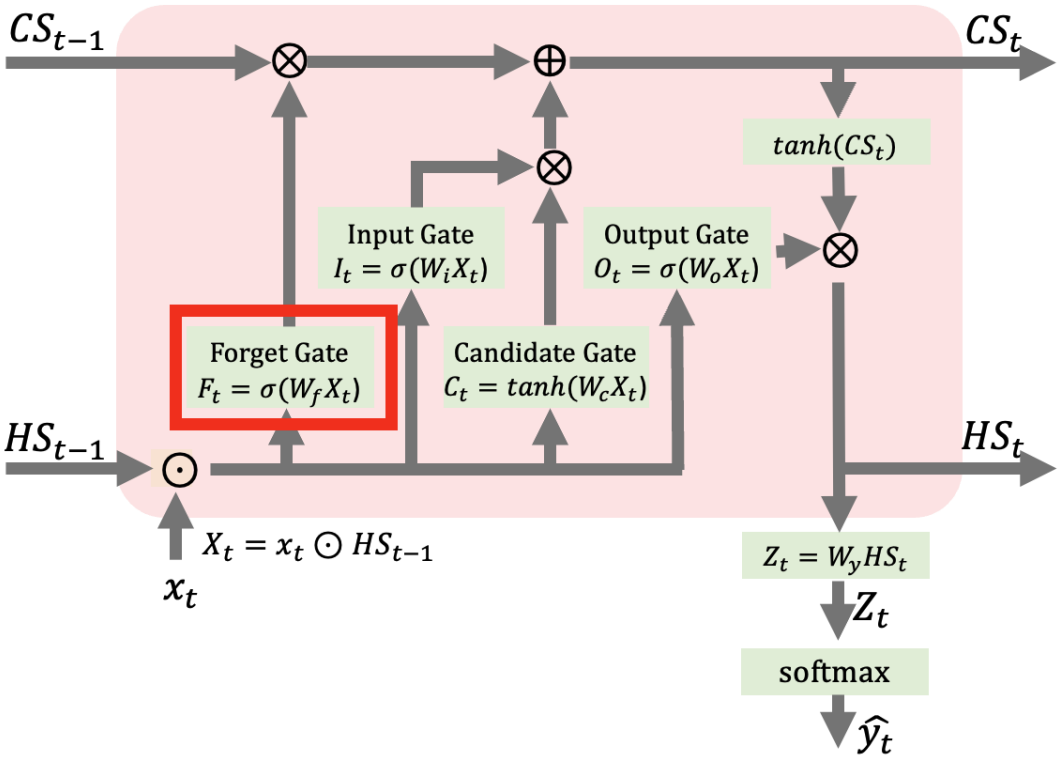
The secret lies in the information called the **cell state (CS)**.



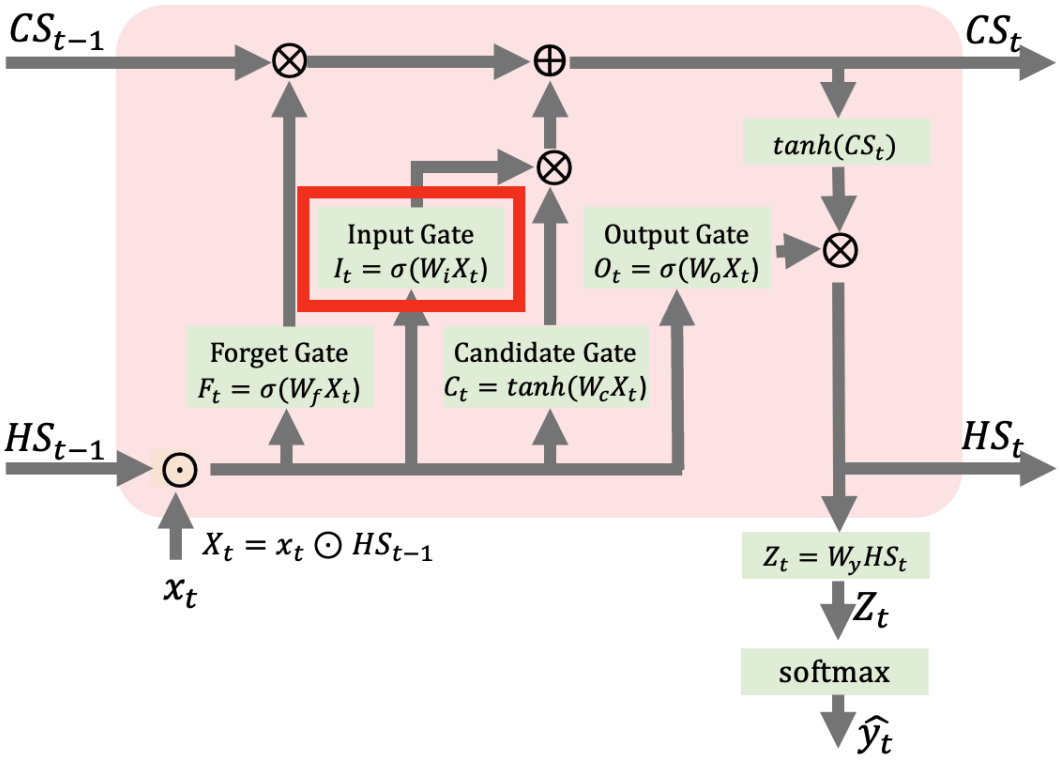
And LSTM has four gates.



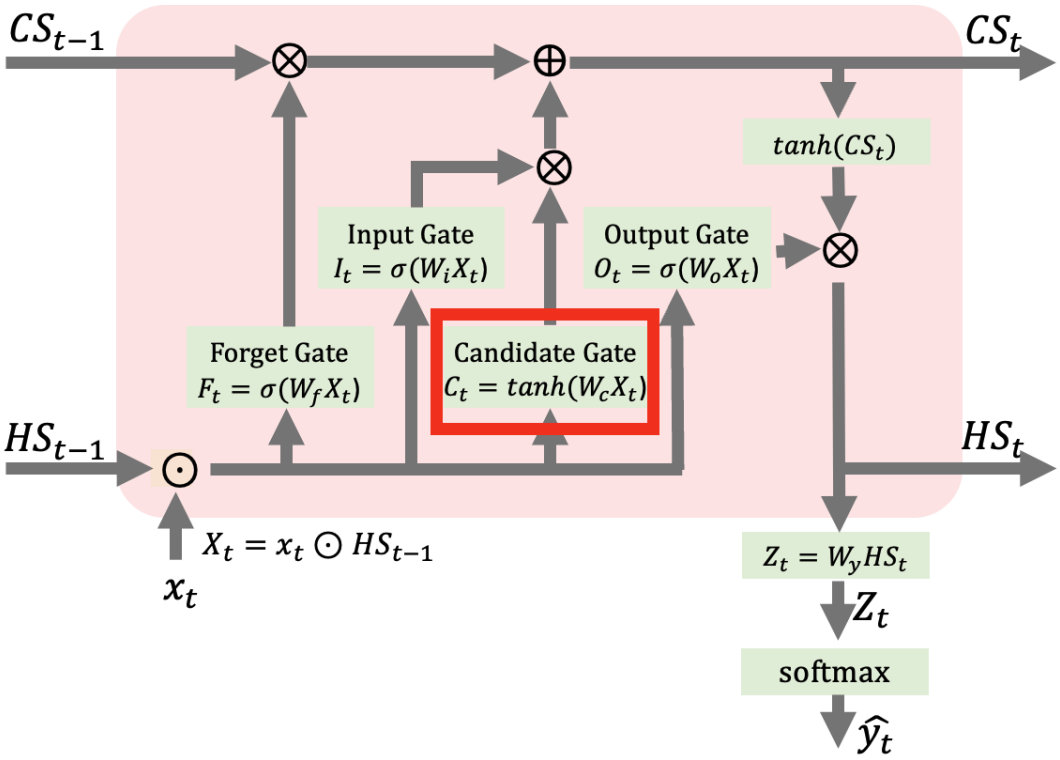
1. Forget gate



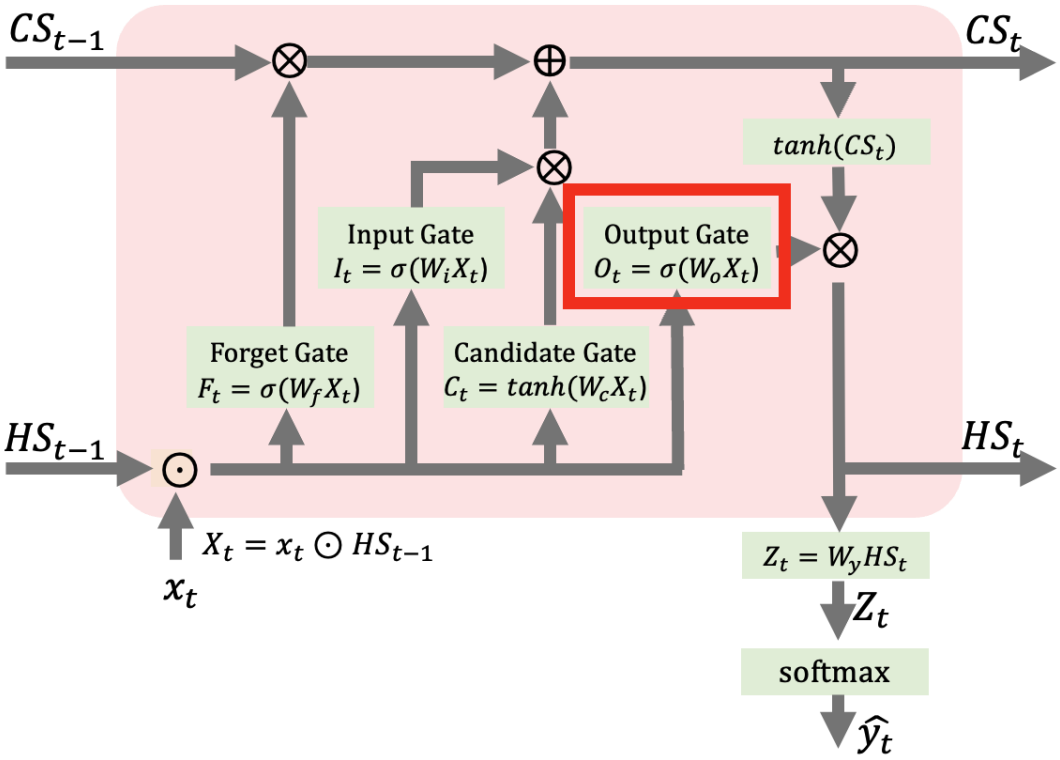
2. Input gate


CS_{t-1}
CS_t
$tanh(CS_t)$
Input Gate
$I_t = \sigma(W_i X_t)$
Output Gate
$O_t = \sigma(W_o X_t)$
Forget Gate
$F_t = \sigma(W_f X_t)$
Candidate Gate
$C_t = tanh(W_c X_t)$
HS_{t-1}
HS_t
$X_t = x_t \odot HS_{t-1}$
x_t
$Z_t = W_y HS_t$
Z_t
softmax
$\hat{y}_t$

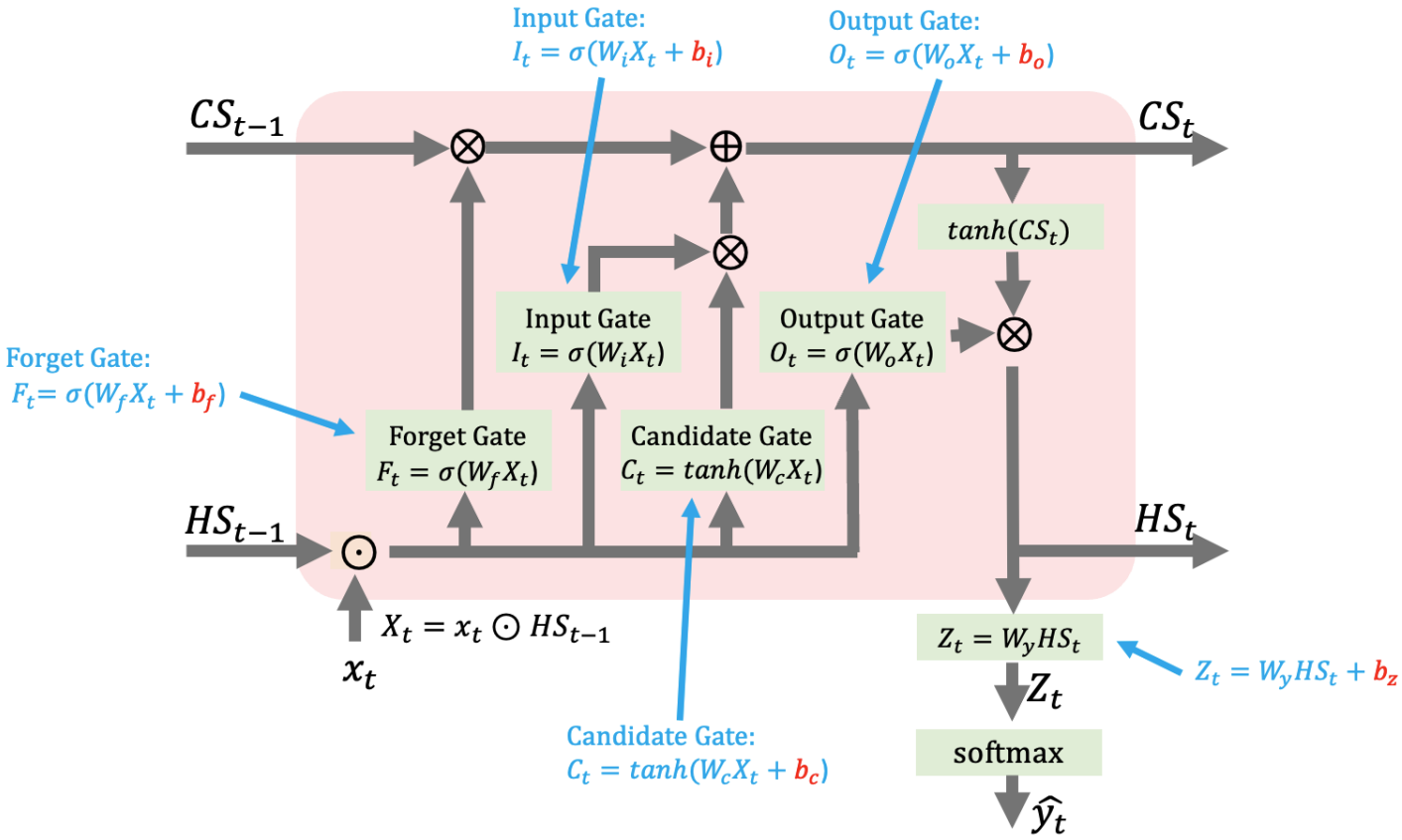
3. Candidate gate



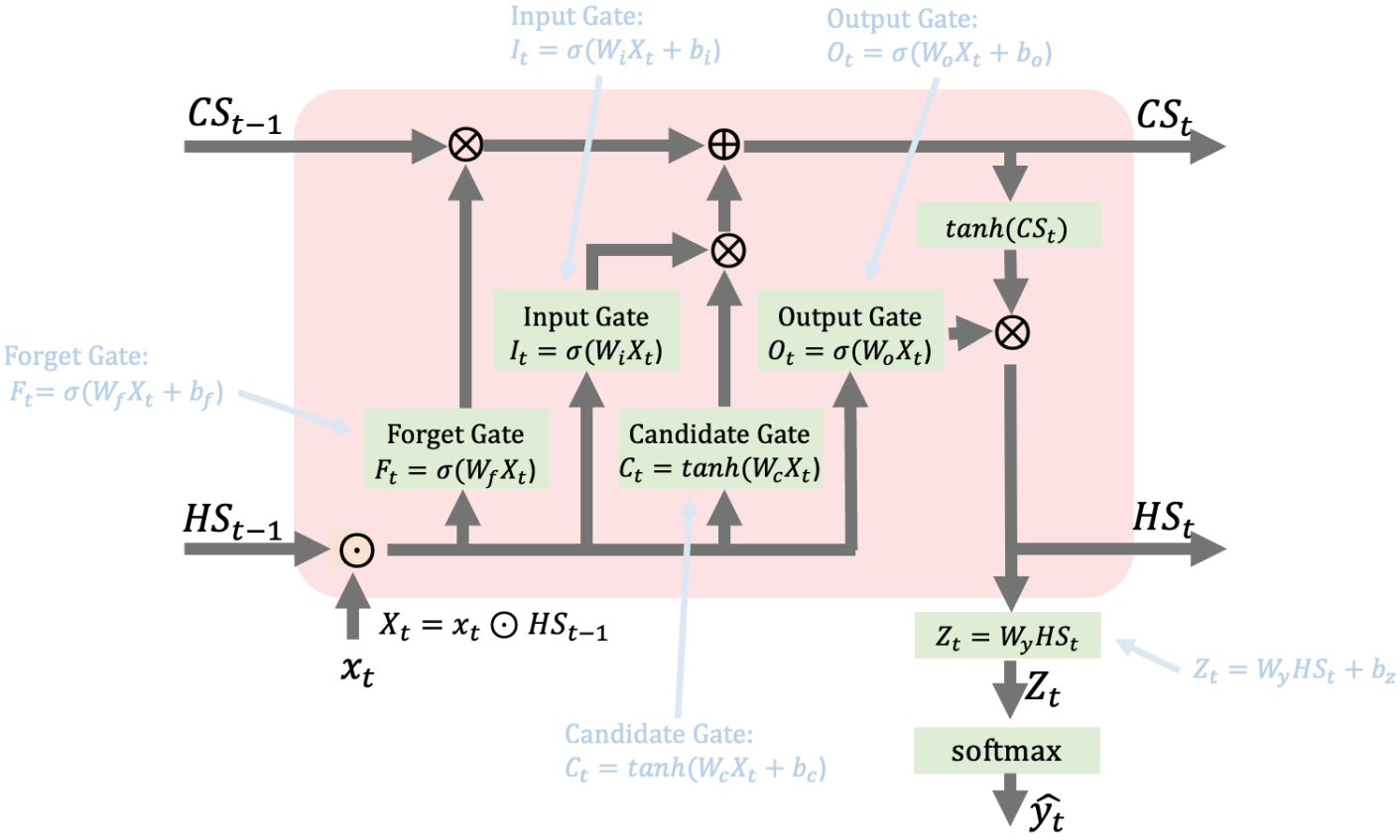
4. Output gate



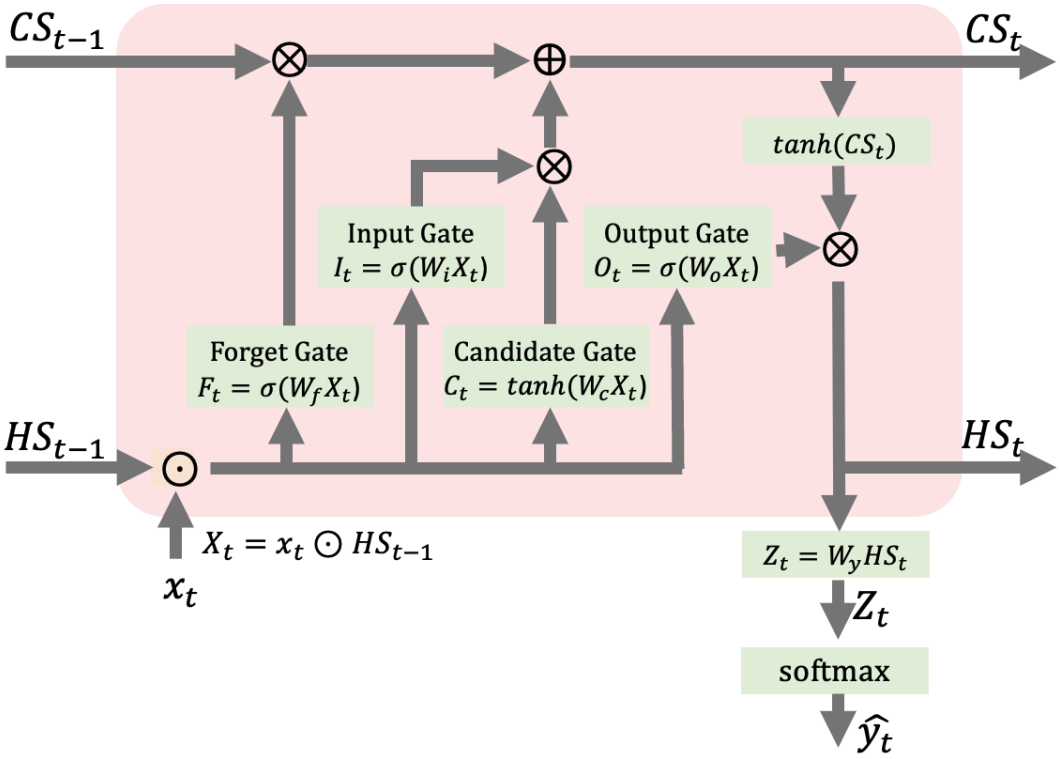
Originally, each gate and layer should include a bias term.



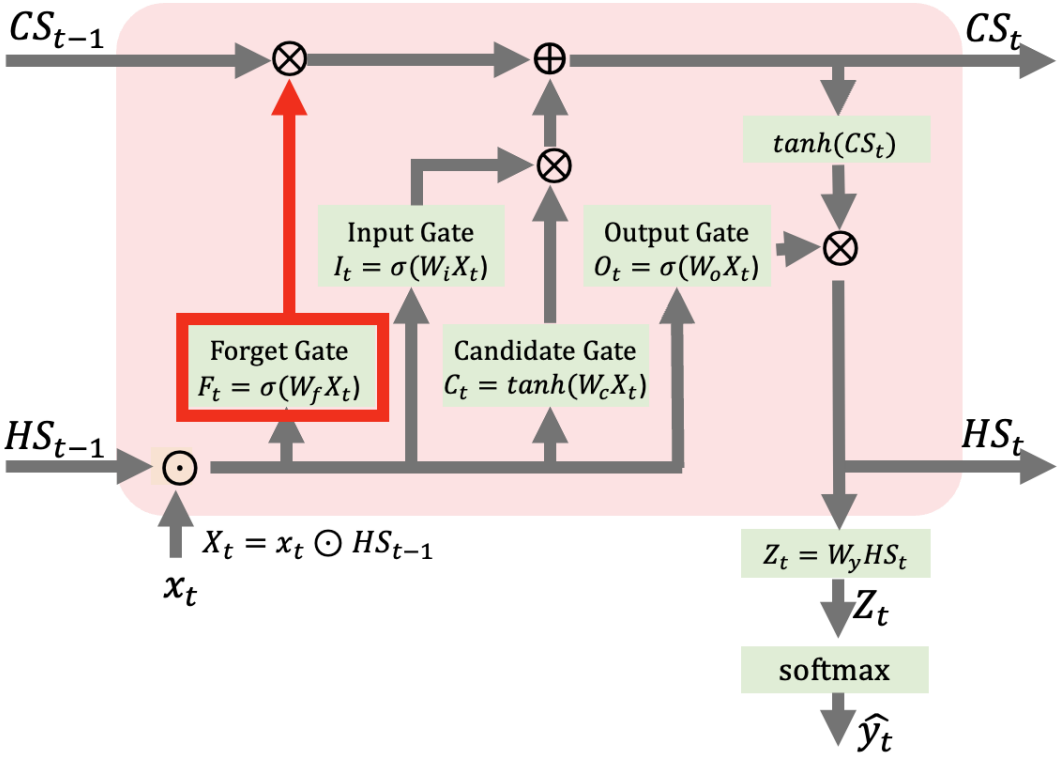
But for convenience, we'll omit them for now.



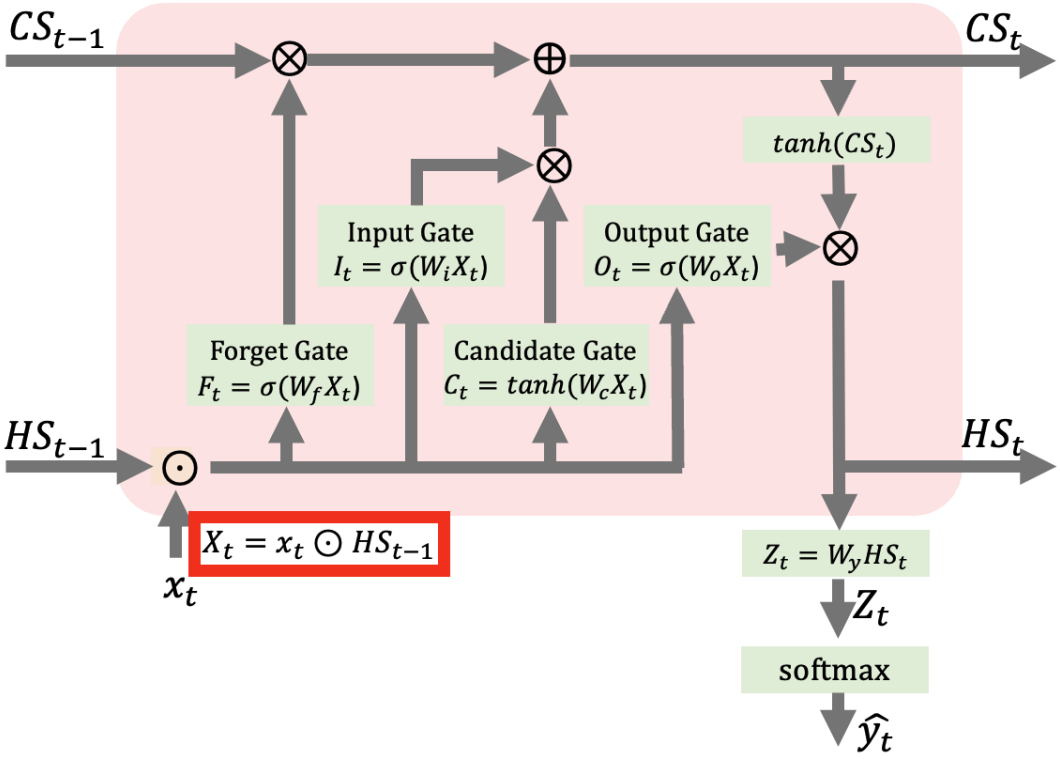
Now, let's see how each gate processes information.



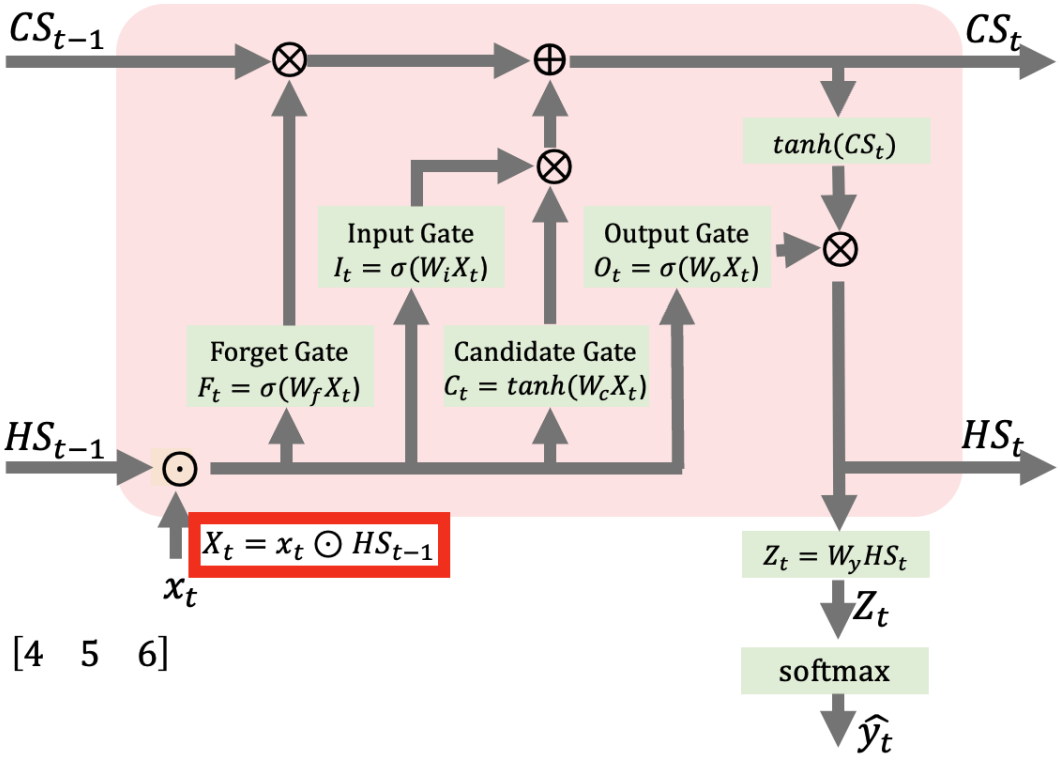
First, as the name suggests, the Forget Gate decides which information to erase (forget).



The input to the the Forget Gate is the concatenation of the previous hidden state (HS_{t-1}) and the current input (x_t).

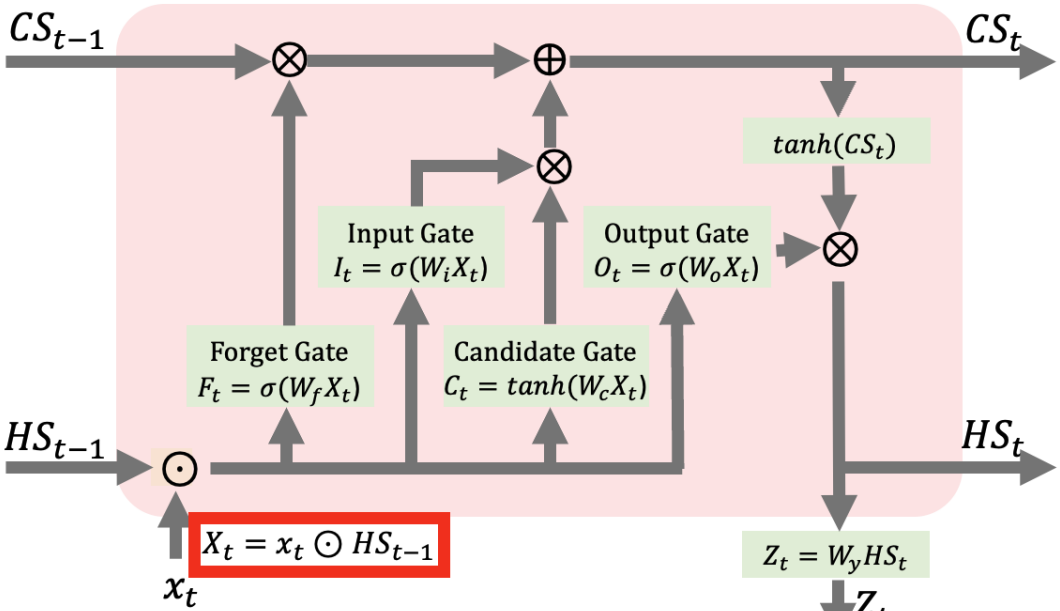


Concatenate: Joining two vectors/matrices end-to-end.



$[1 \quad 2 \quad 3] \odot [4 \quad 5 \quad 6]$

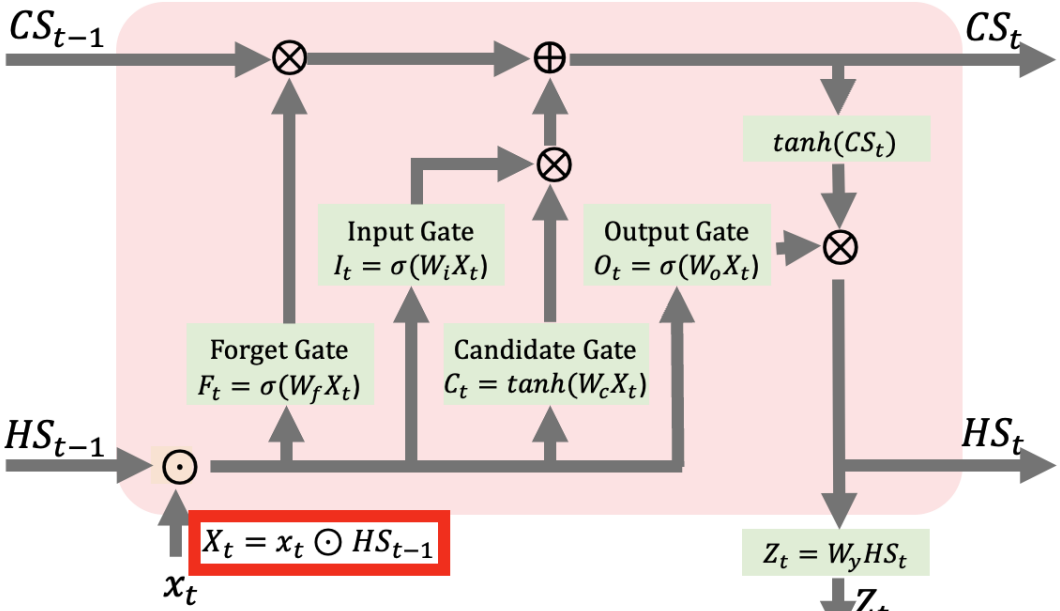
Concatenate: Joining two vectors/matrices end-to-end.



$[1 \quad 2 \quad 3] \odot [4 \quad 5 \quad 6]$

$= [1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6]$

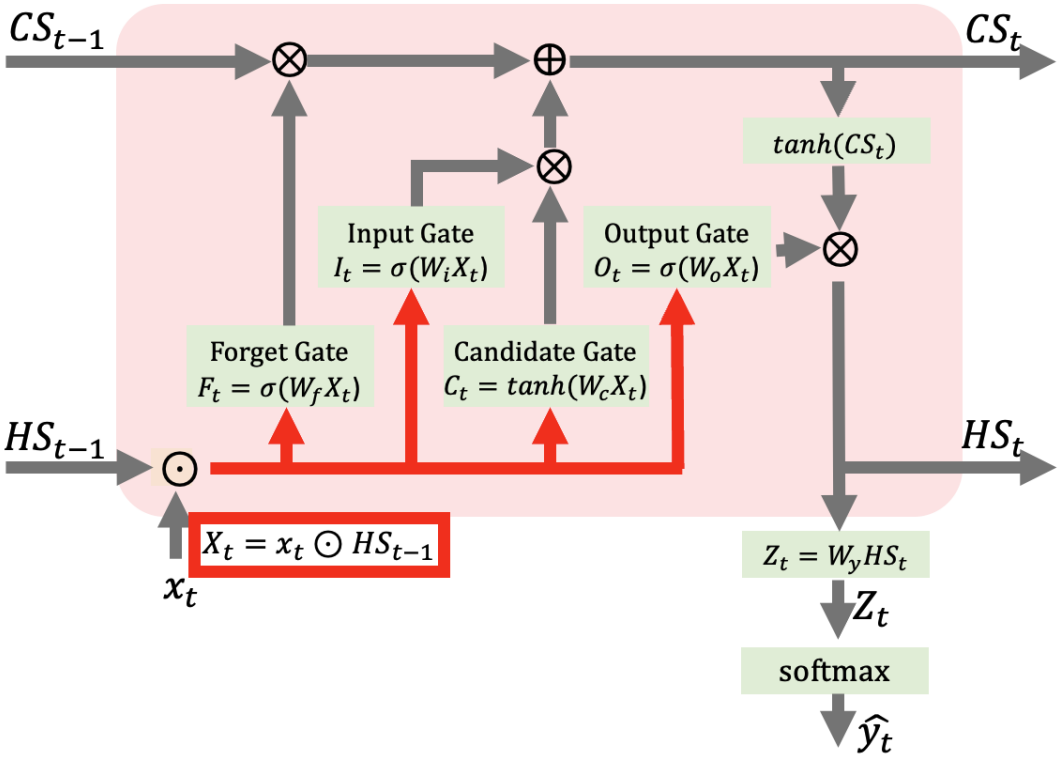
By doing this, the concatenated x_t becomes a kind of short-term memory that bundles the previous hidden state and the current input together.



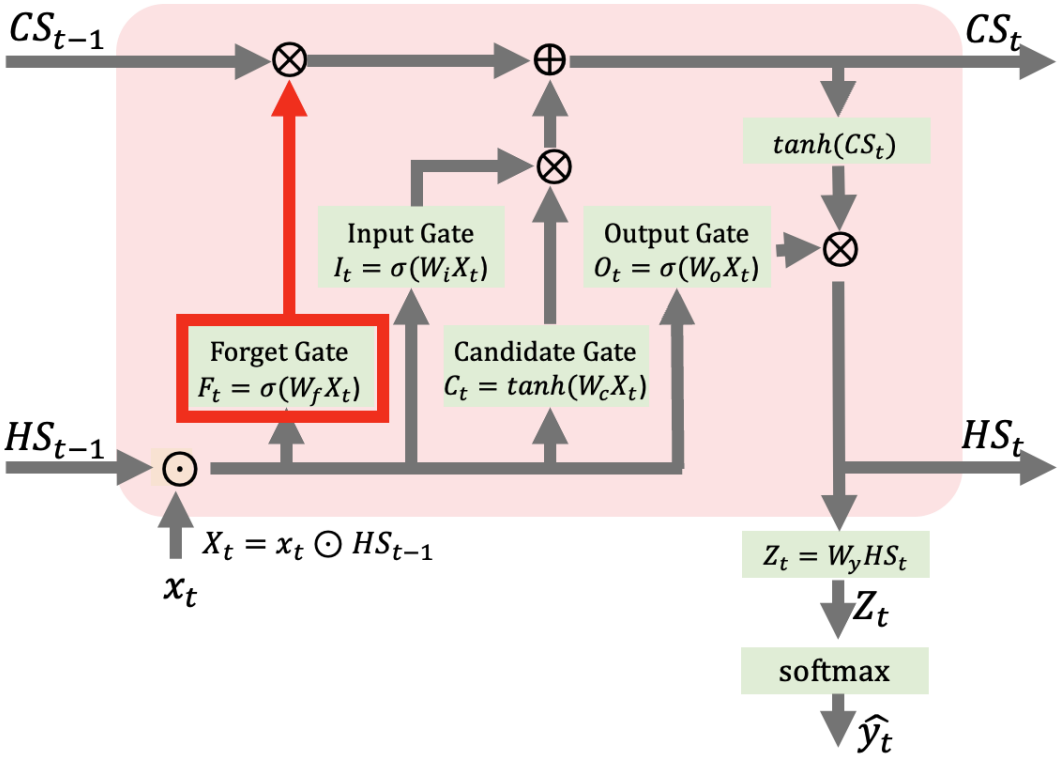
$[1 \quad 2 \quad 3] \odot [4 \quad 5 \quad 6]$

$= [1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6]$

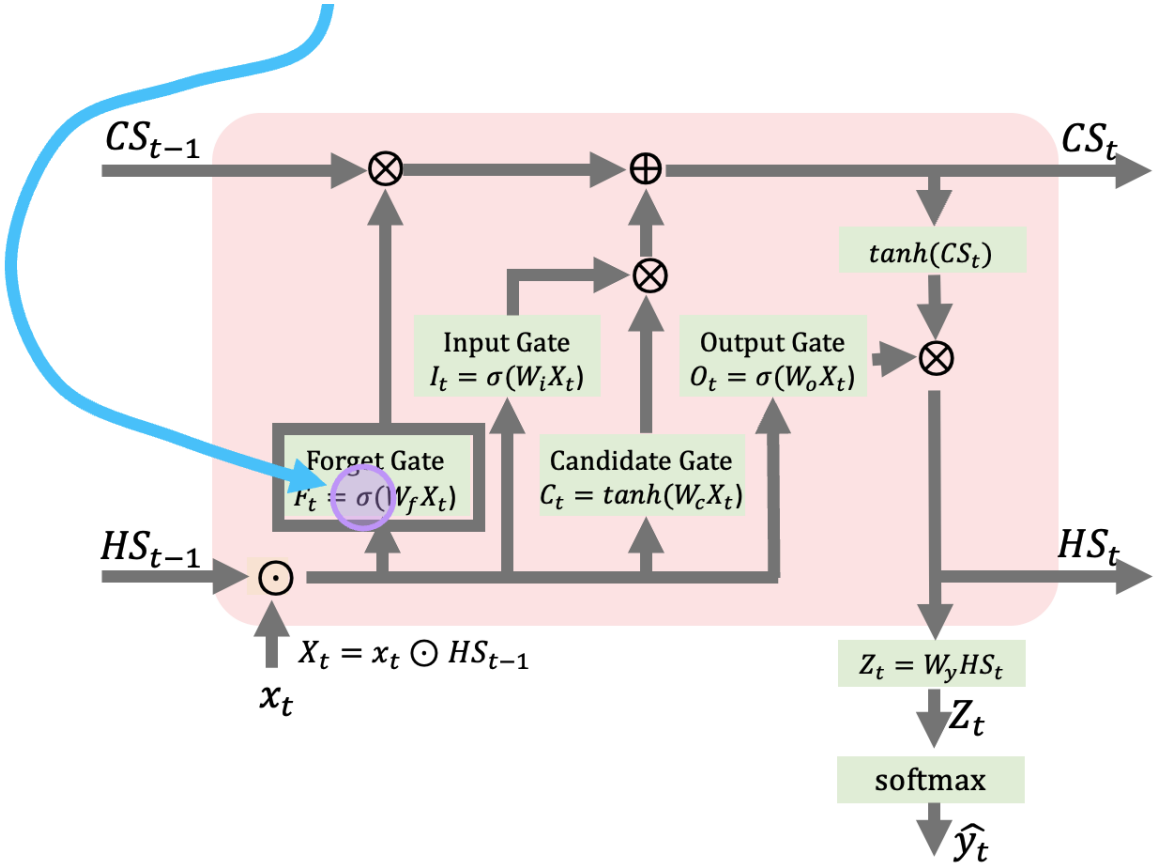
Remember: this x_t serves as the input to all gates in the LSTM.



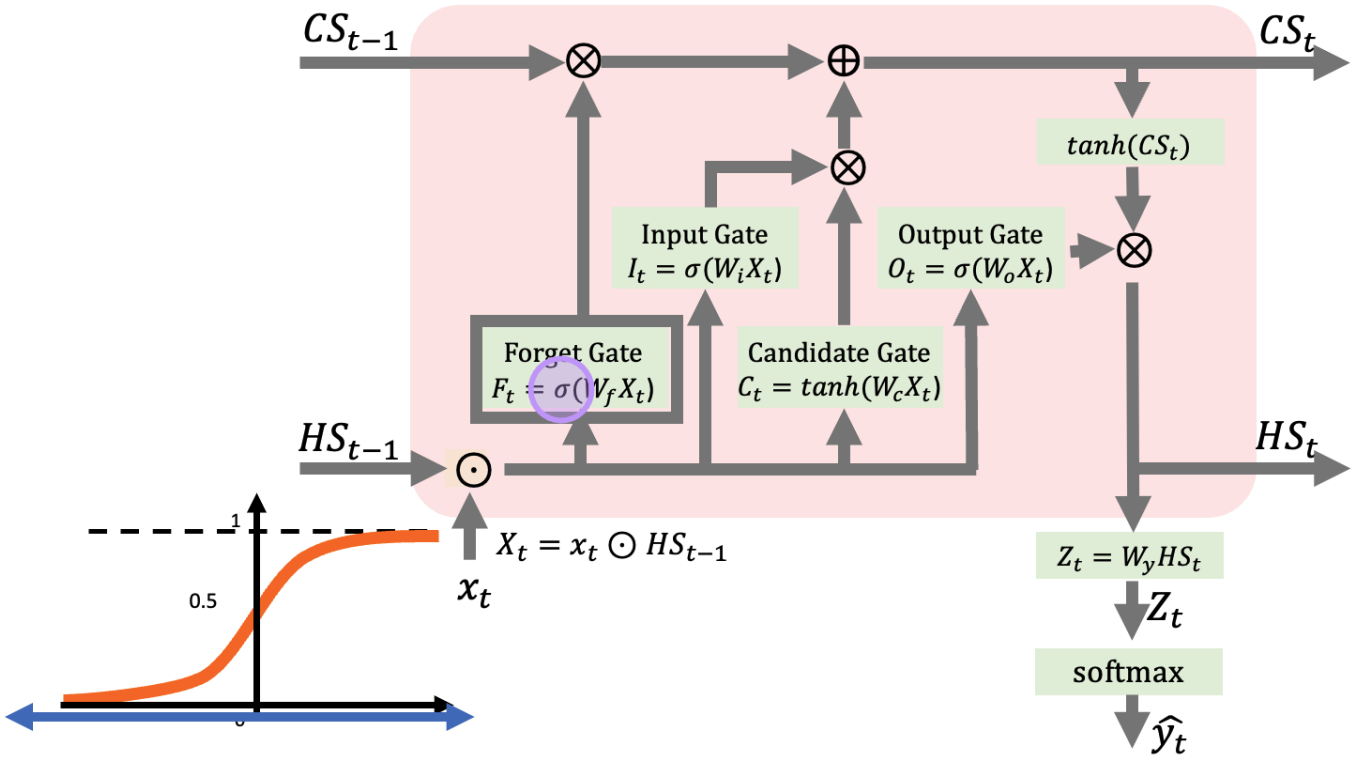
The first thing to note in the Forget Gate is:



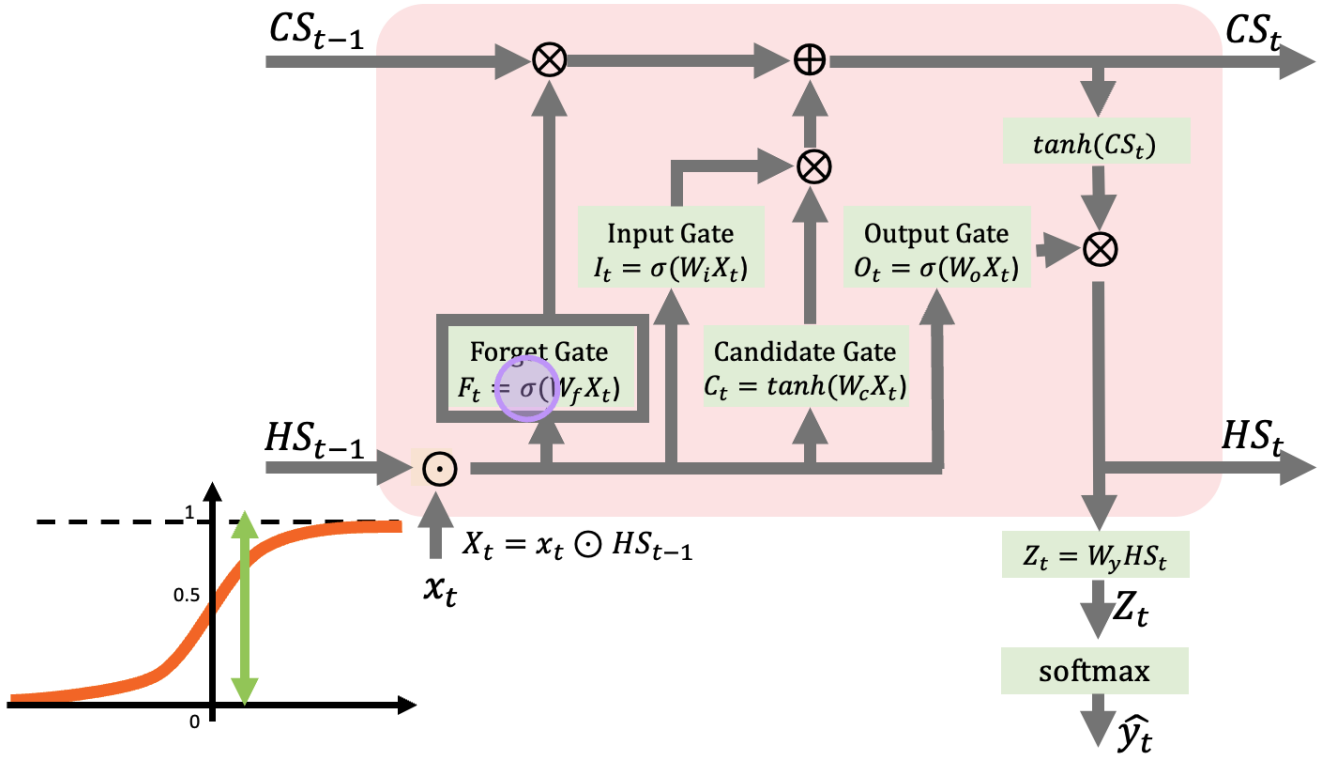
There is a sigmoid function inside the Forget Gate.



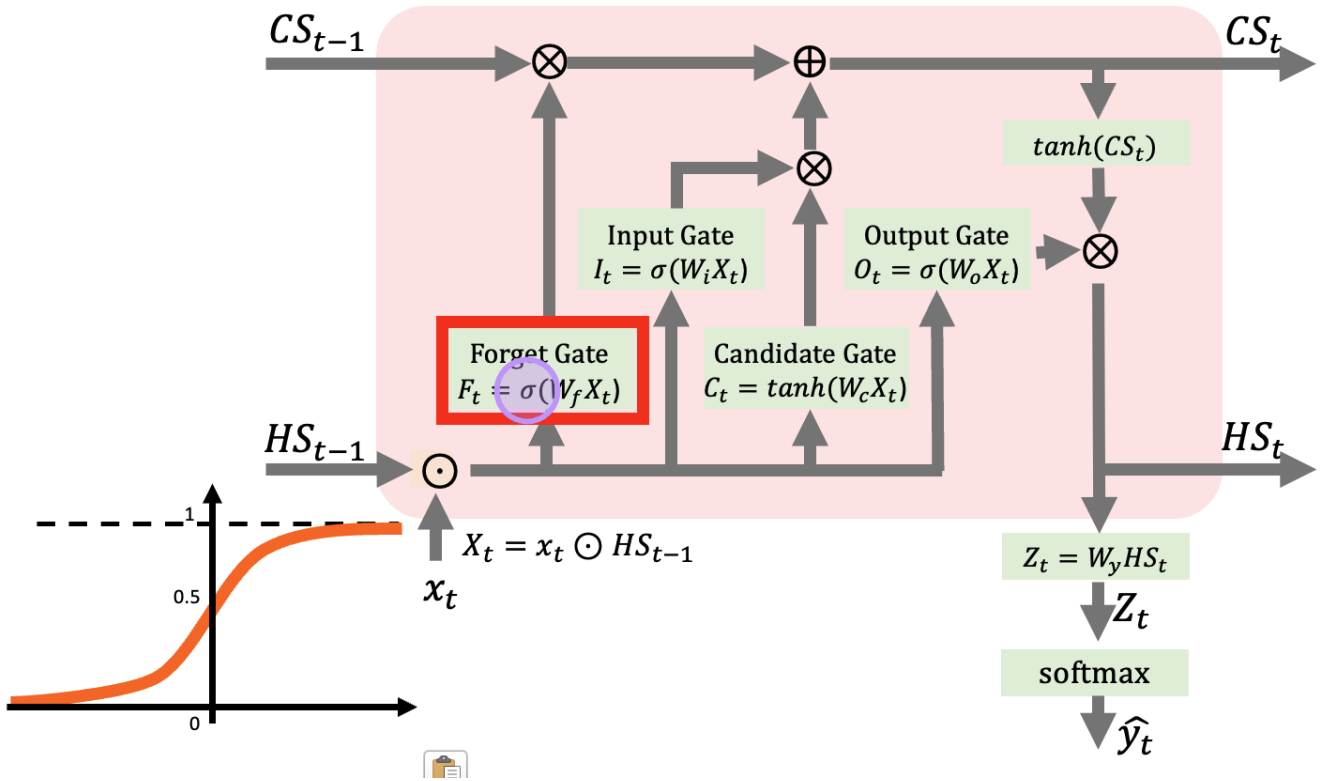
As we learned about the sigmoid, regardless of the input,



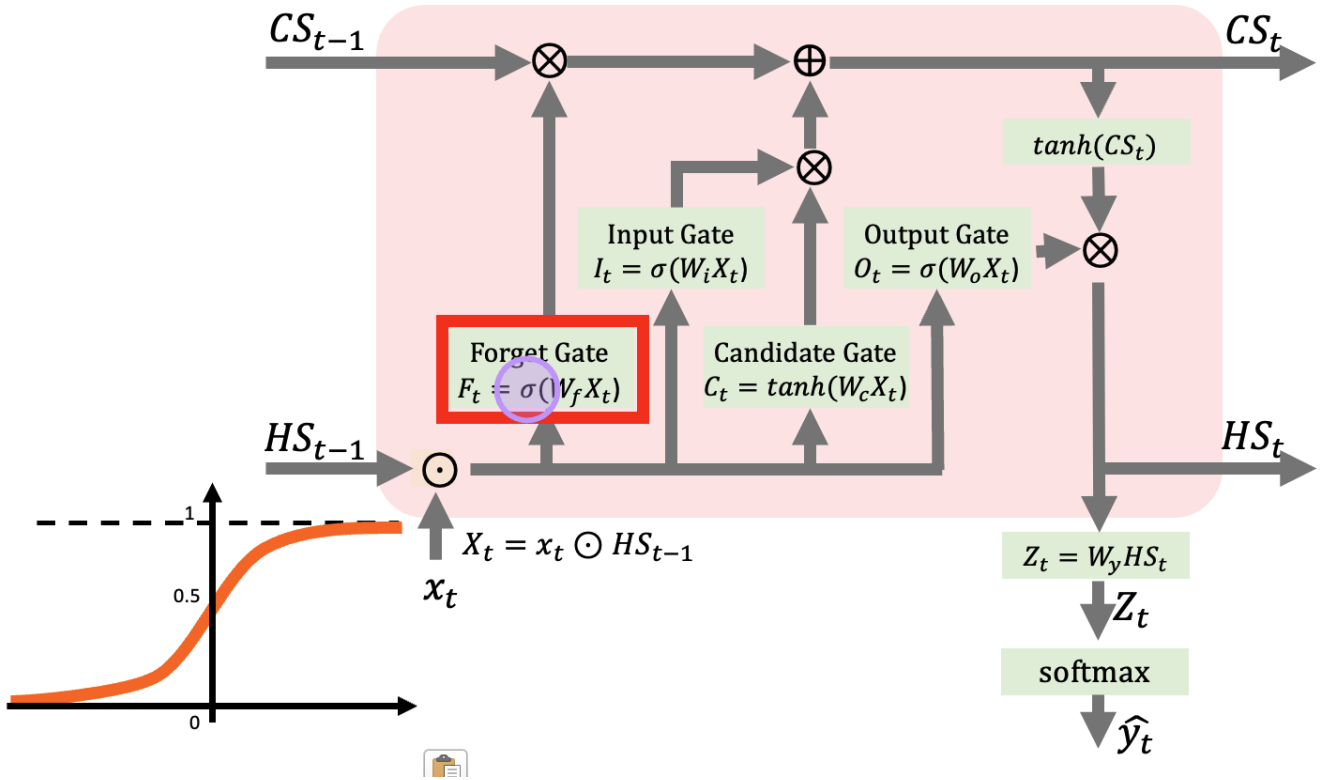
it **returns a value between 0 and 1.**



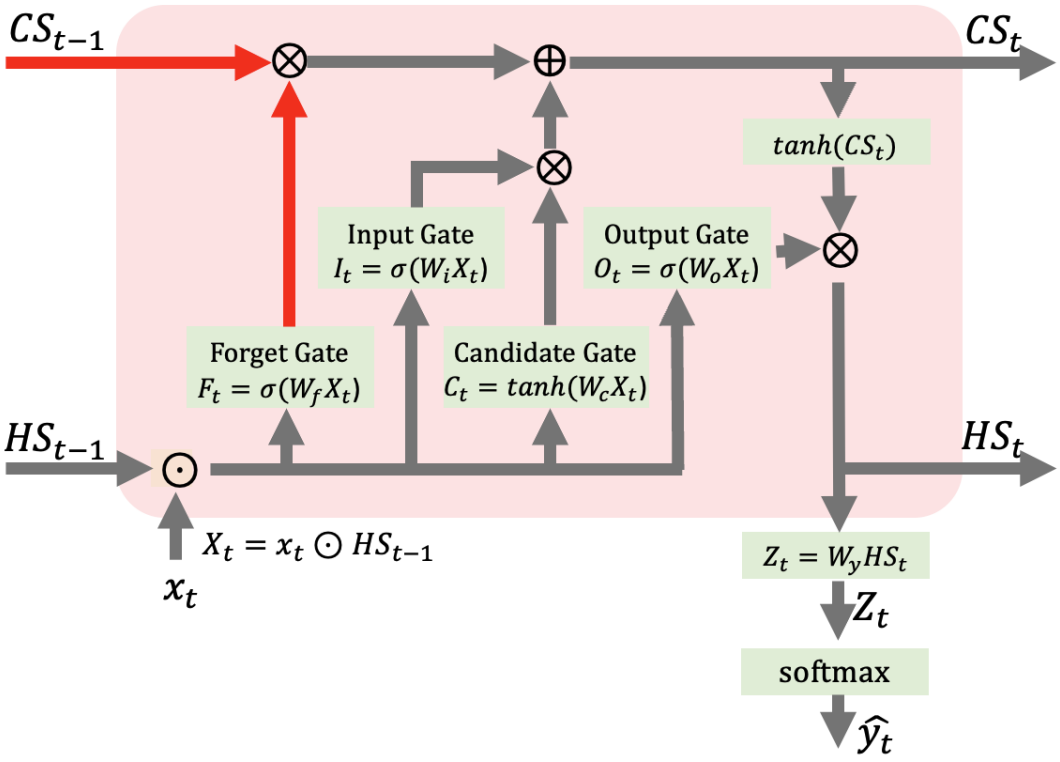
So, what the Forget Gate does is: it takes the (just-prior + current) input, multiplies by weights,



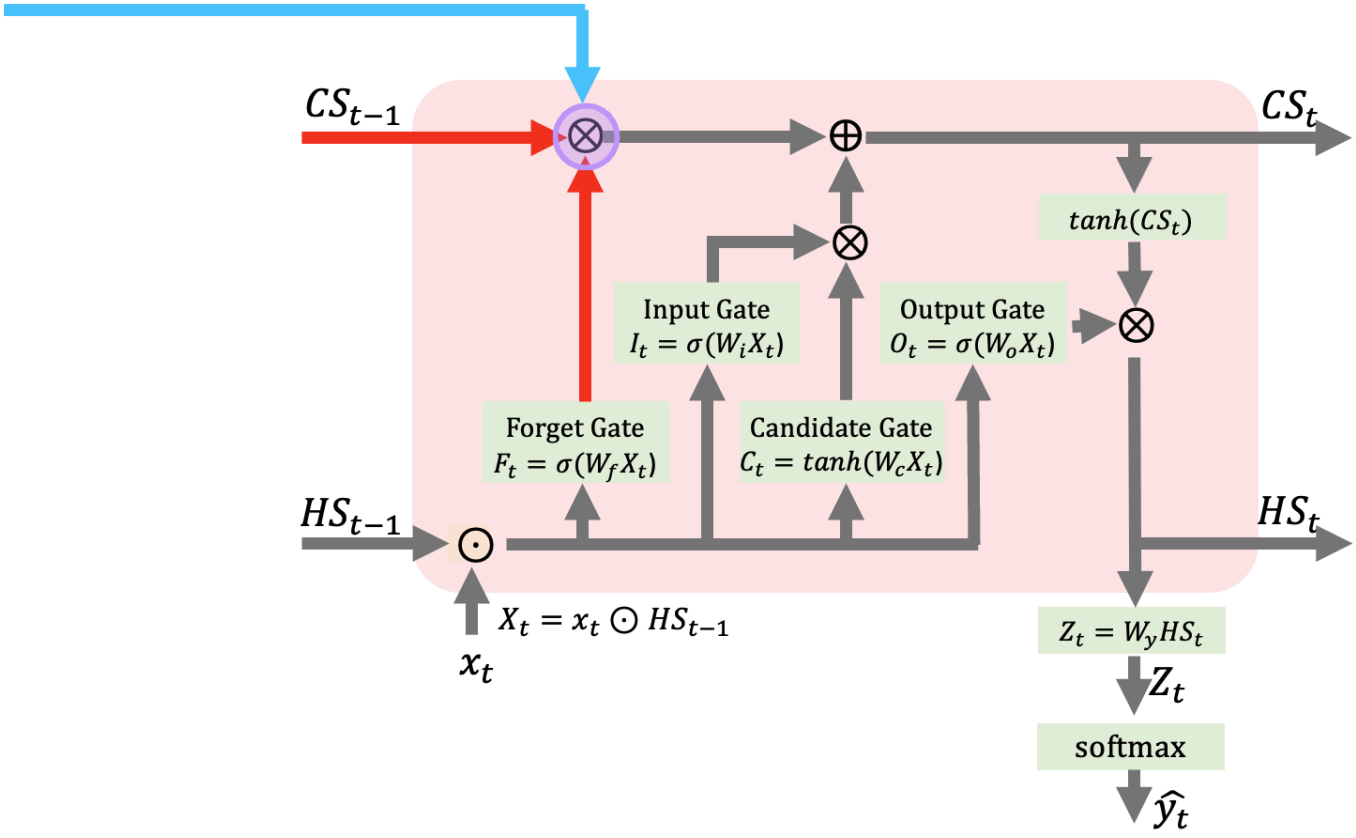
and maps it to values between 0 and 1.



Then, these 0–1 values meet the cell state values



and undergo element-wise multiplication.

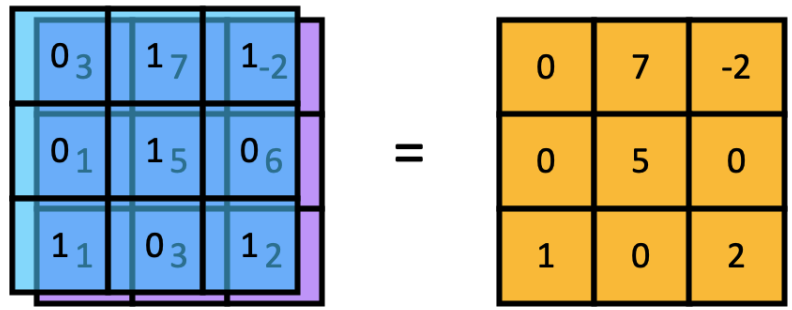


Notes: Element-wise multiplication means **multiplying two matrices by their corresponding elements**.

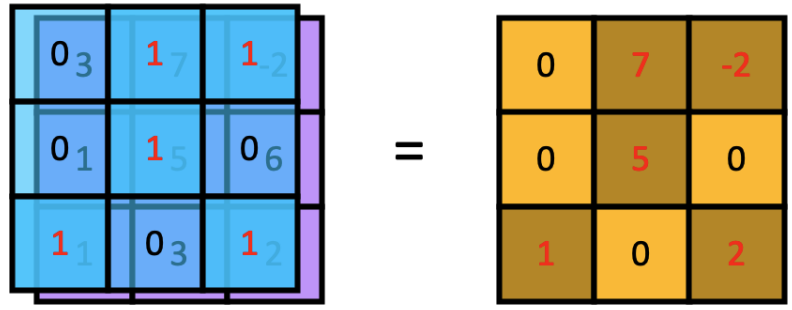
0	1	1
0	1	0
1	0	1

3	7	-2
1	5	6
1	3	2

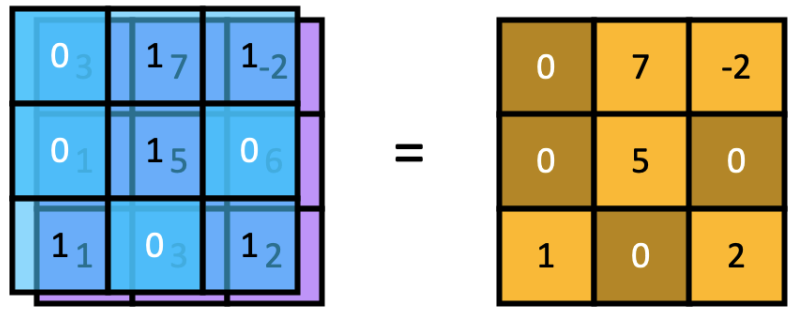
Notes: Element-wise multiplication means **multiplying two matrices by their corresponding elements**.



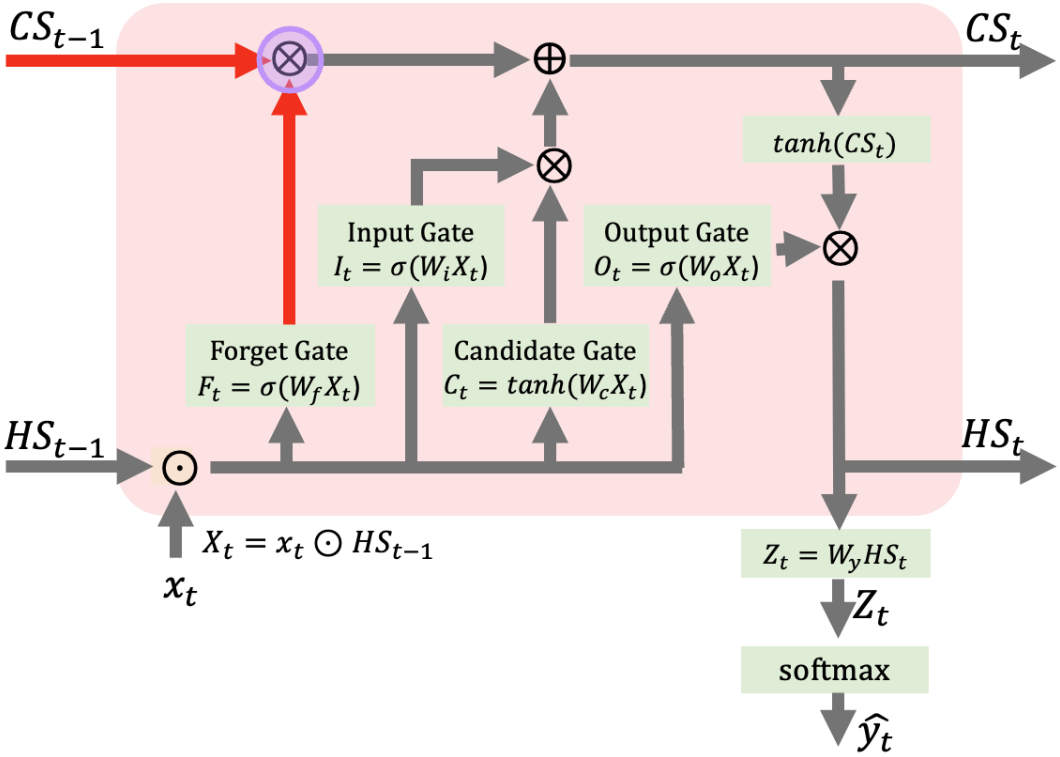
We do this so that entries near 1 are kept



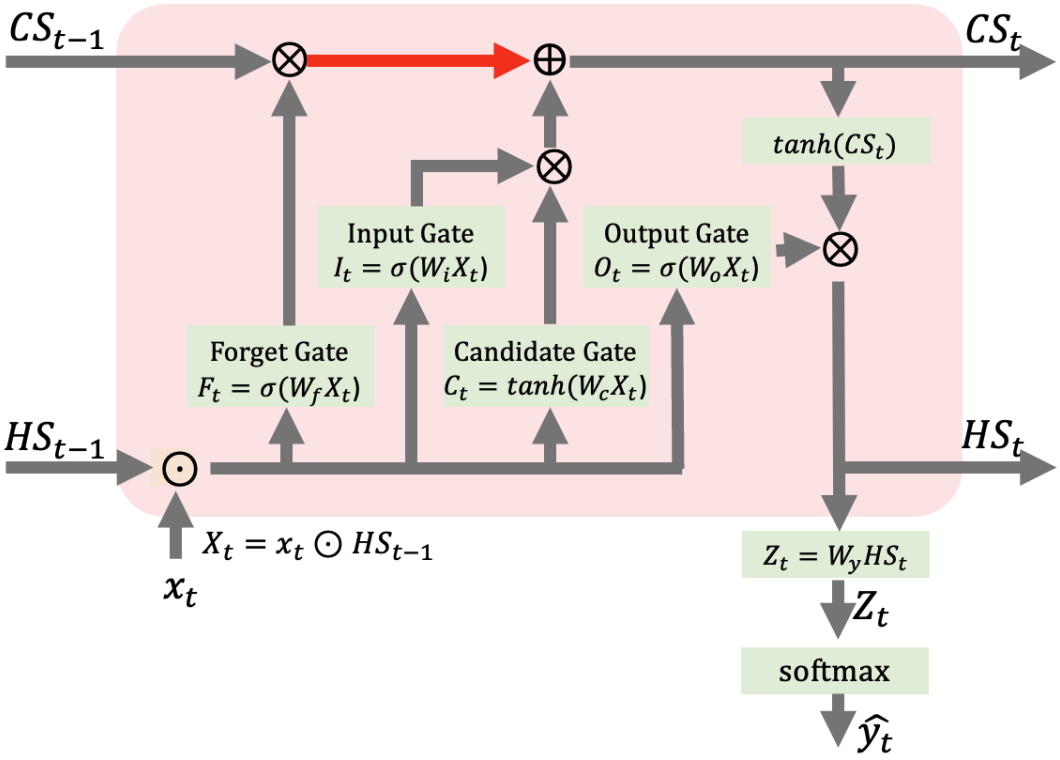
and **entries near 0 are erased (forgotten)**.



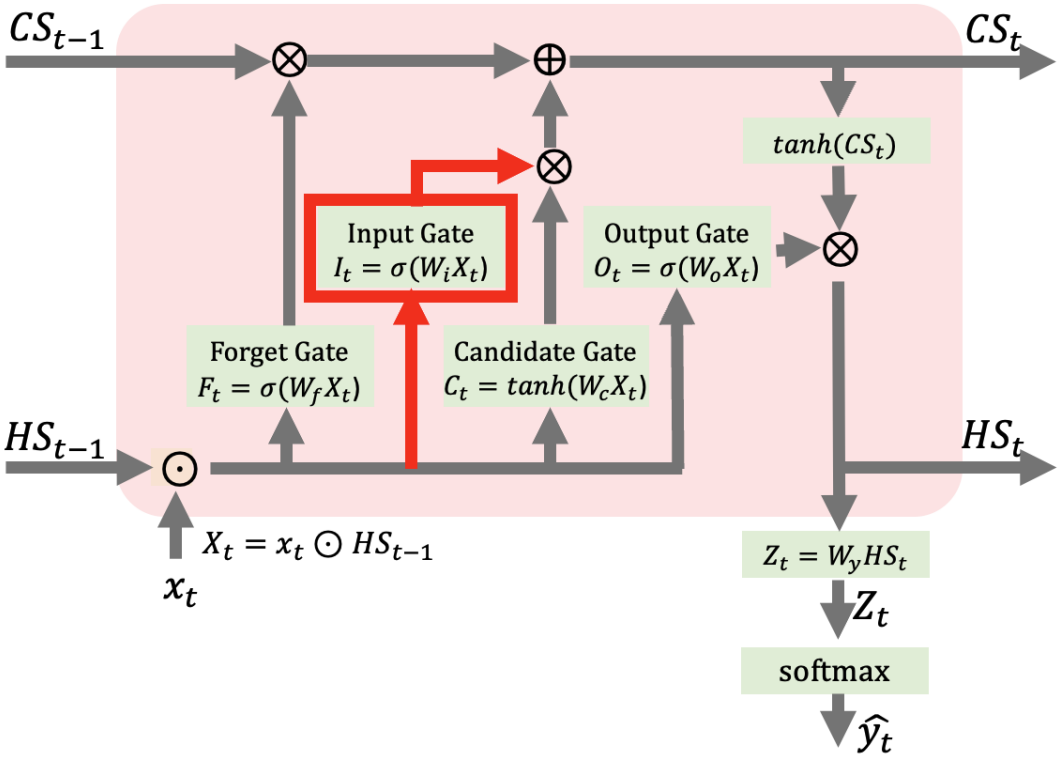
For example, suppose the Forget Gate's output consisted only of 0s and 1s.



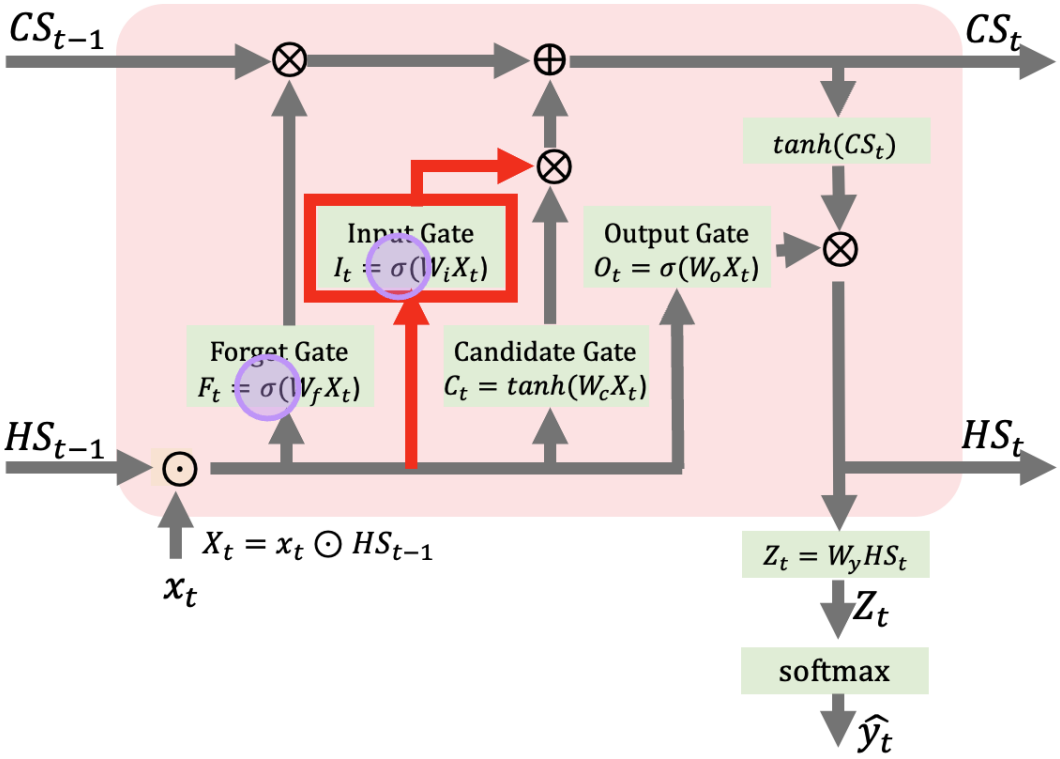
In short, as the cell state (CS) passes through the Forget Gate, it forgets what should be forgotten.



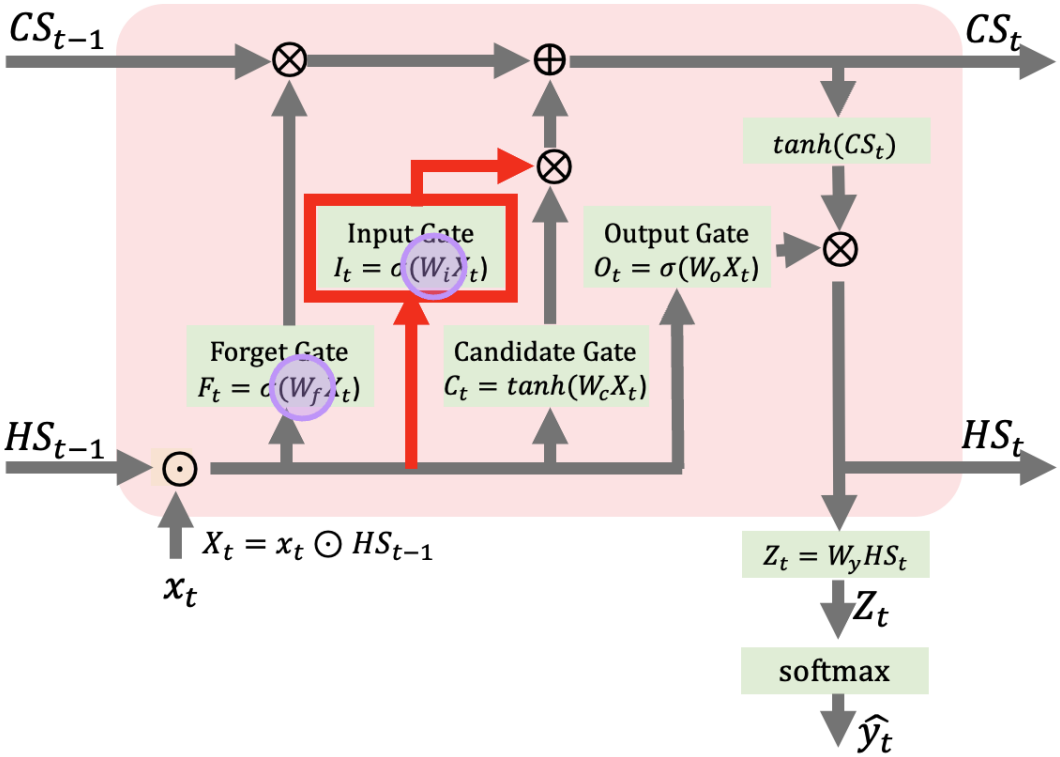
Next, the Input Gate. Its computation is the same pattern as the Forget Gate



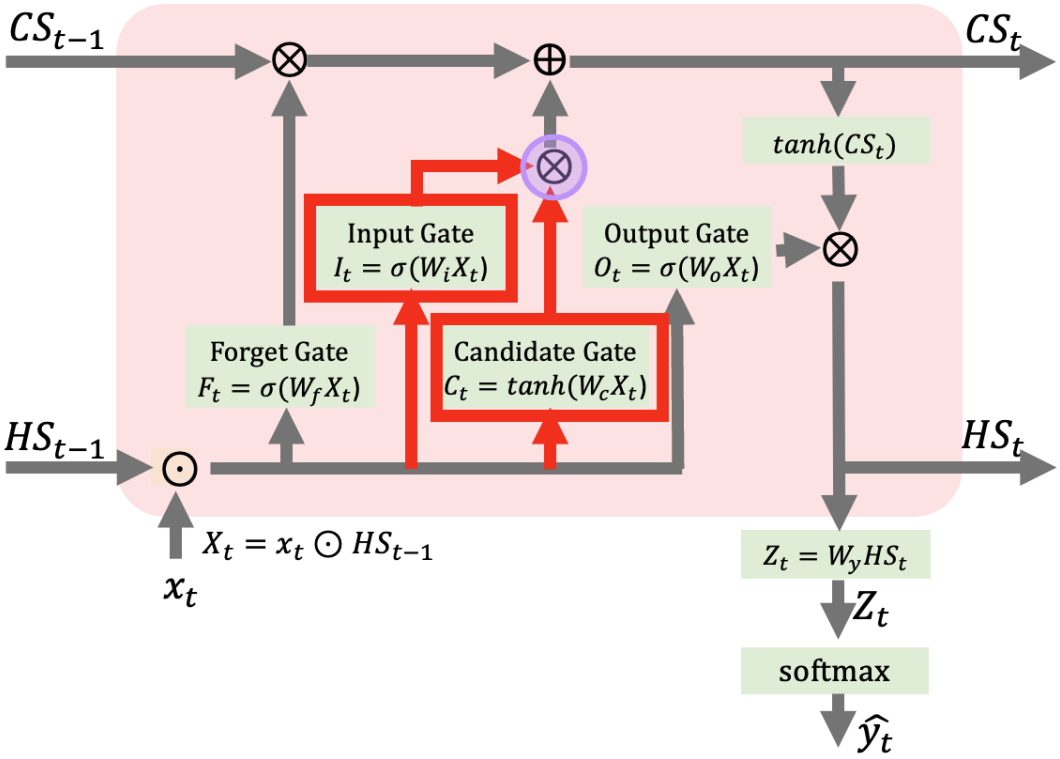
because both use a sigmoid function.



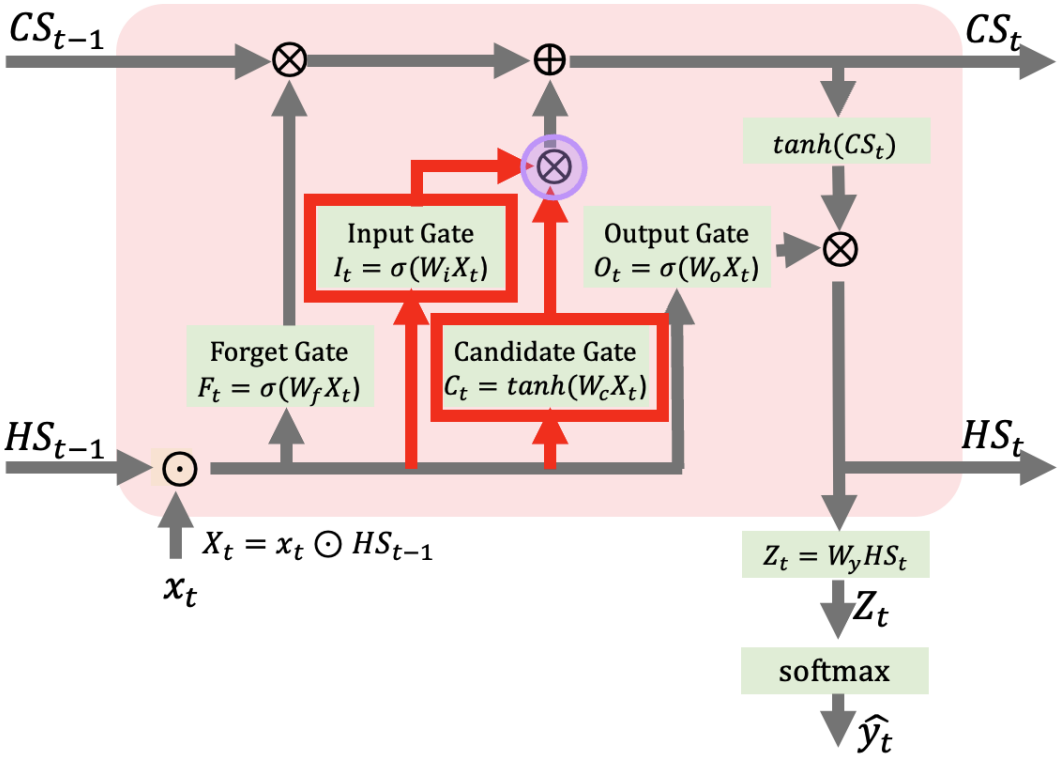
(But) the weights are different.



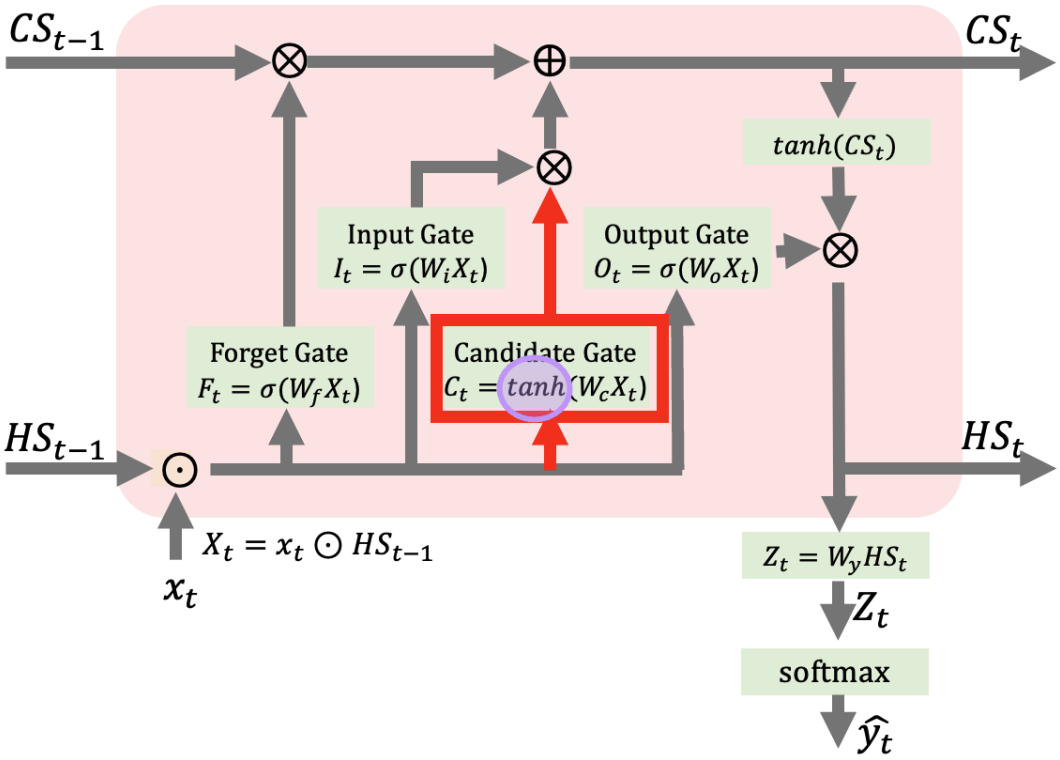
This Input Gate works together with the Candidate Gate



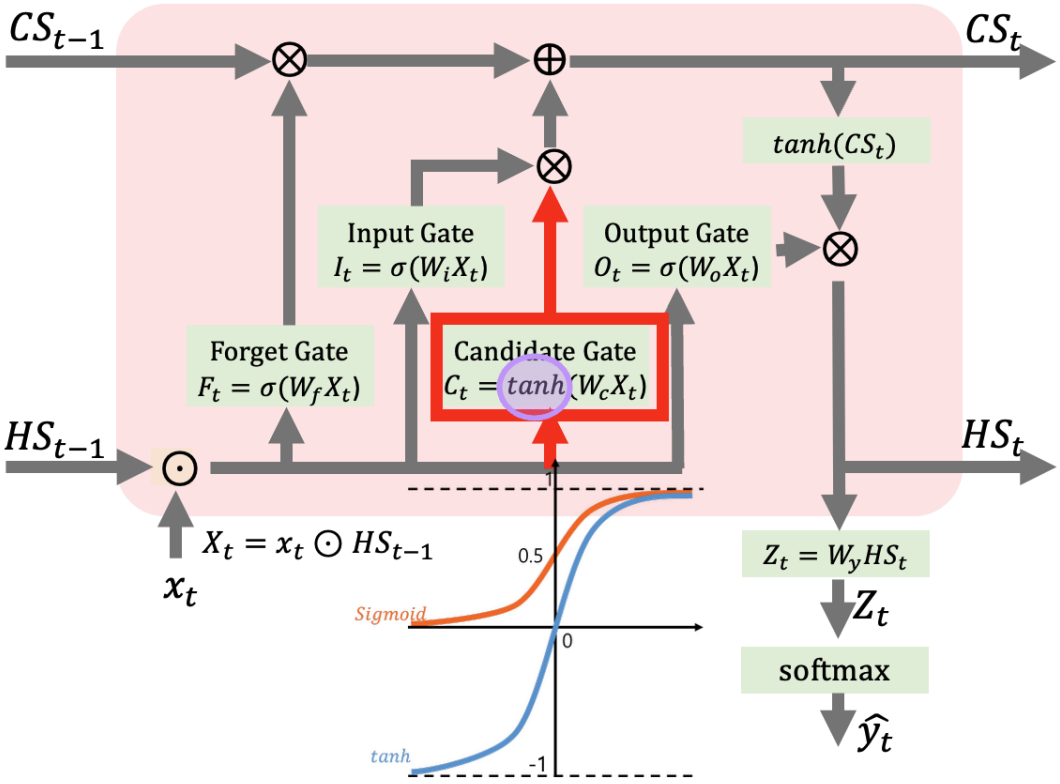
to update the cell state with what should be "remembered."



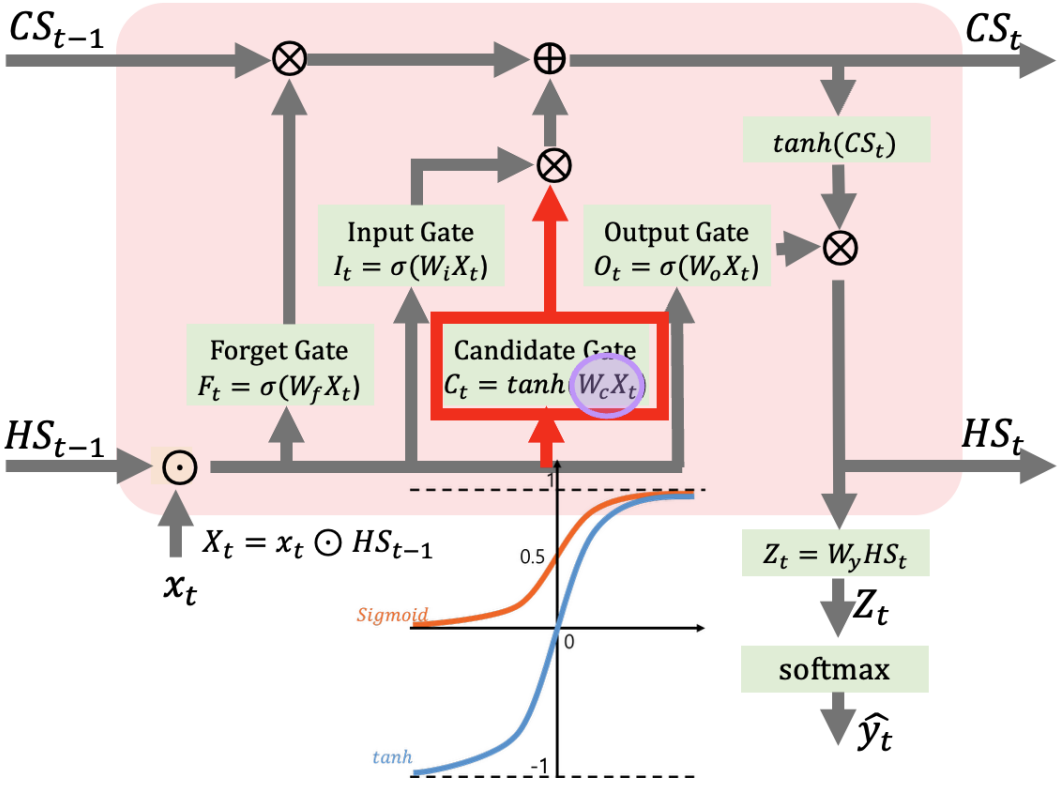
The Candidate Gate uses tanh rather than a sigmoid inside.



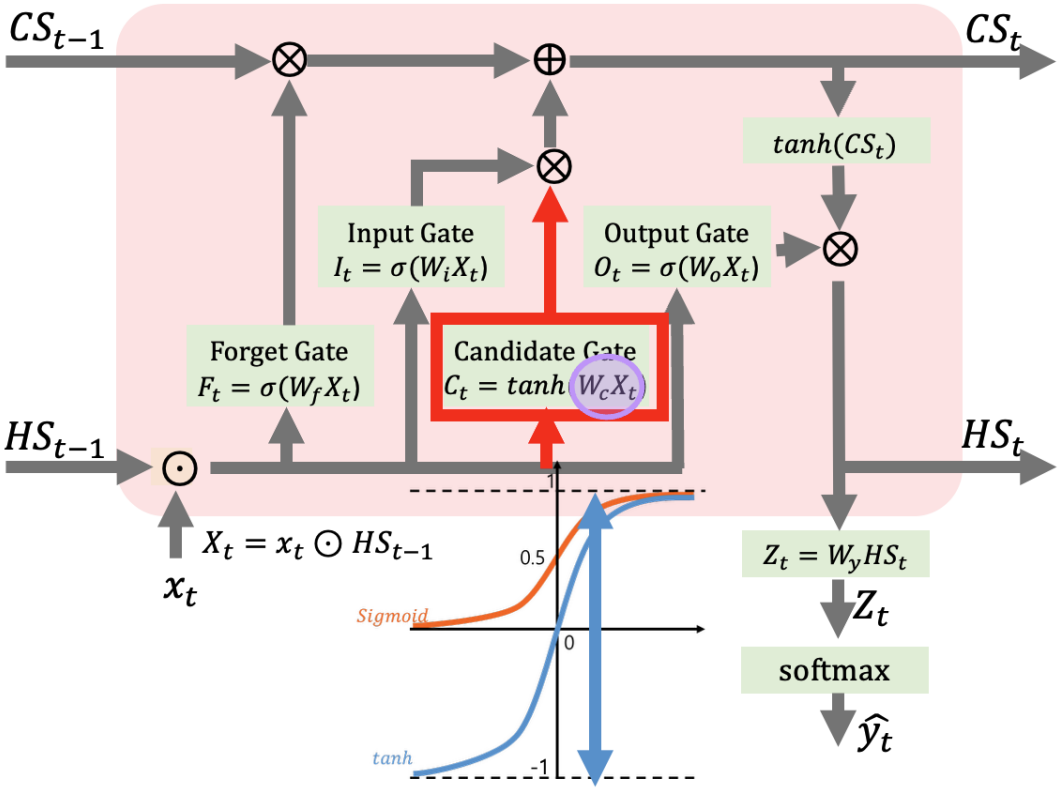
The tanh function maps inputs to values between -1 and 1.



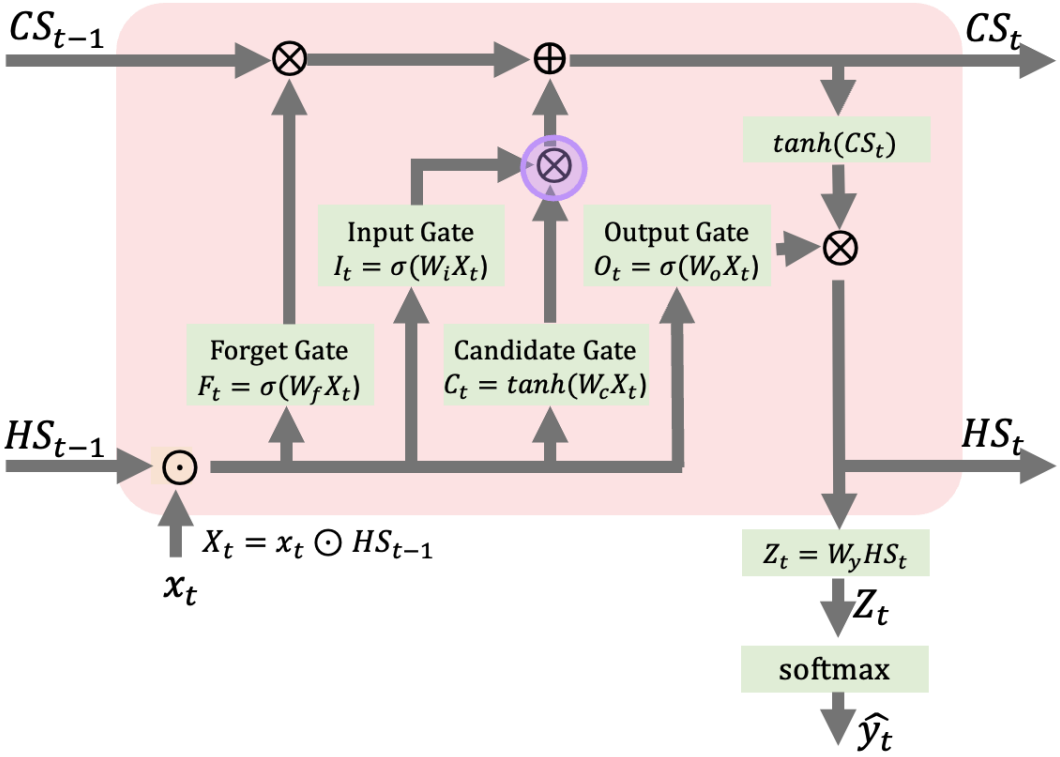
So, what the Candidate Gate does is: multiply the input by weights,



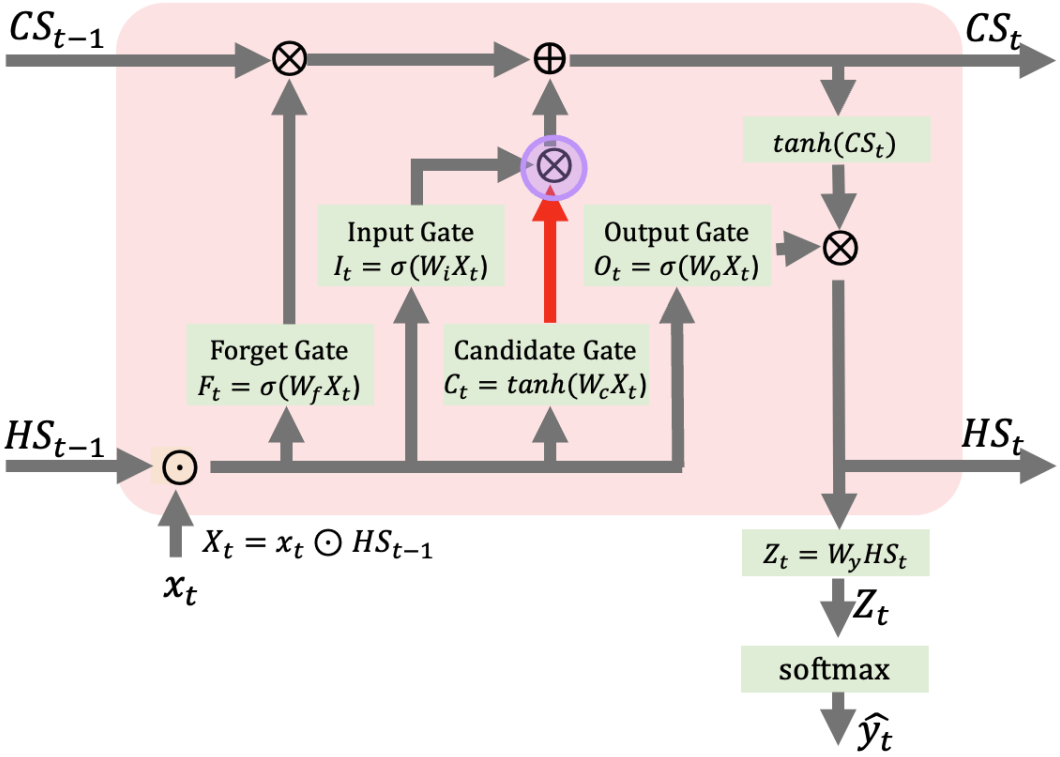
and then preserve the sign while **normalizing the range**.



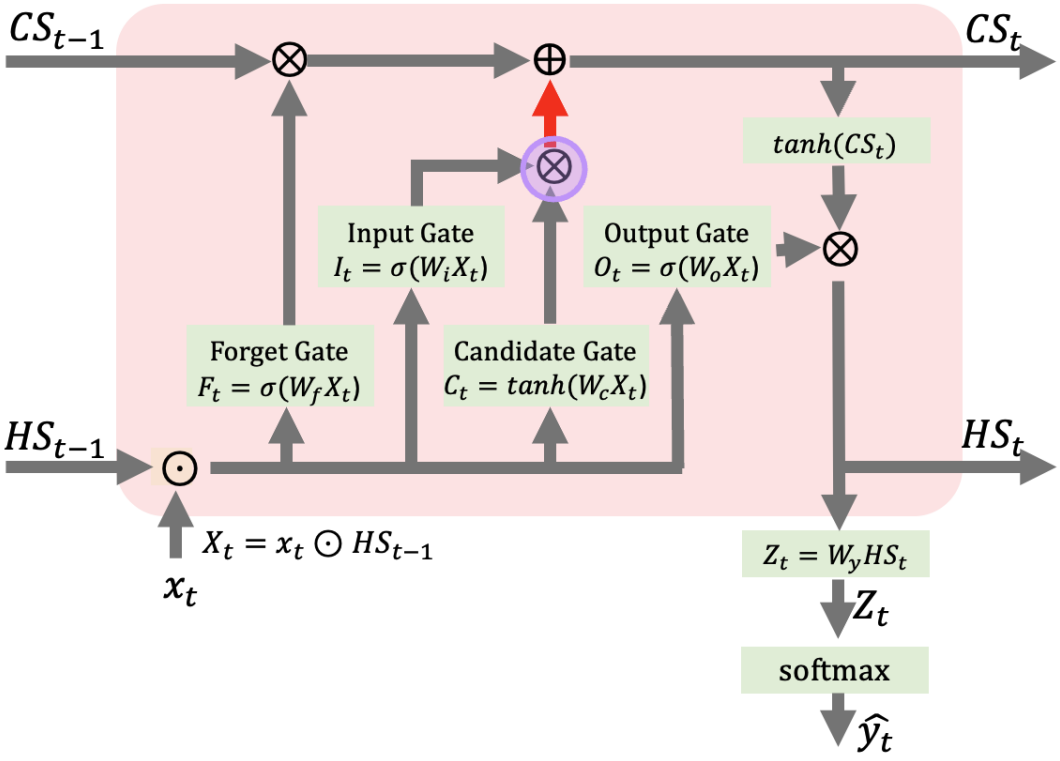
Then, via element-wise multiplication with the 0–1 values from the Input Gate,



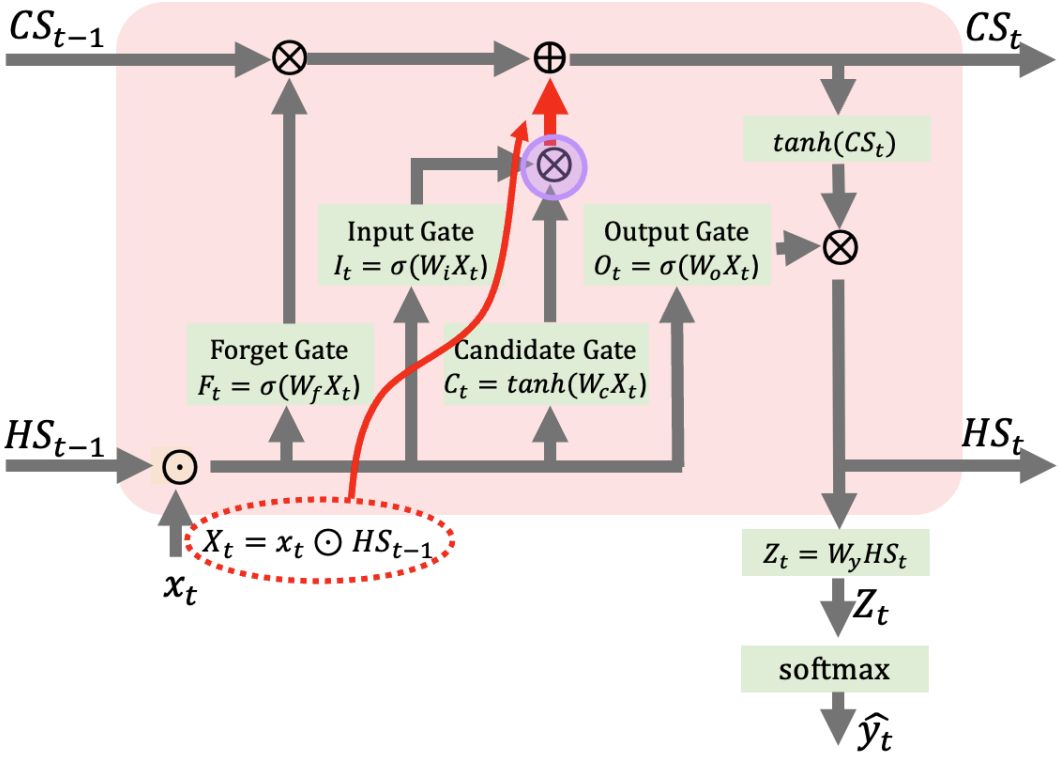
some Candidate outputs are pushed close to 0 while others are kept as they are.



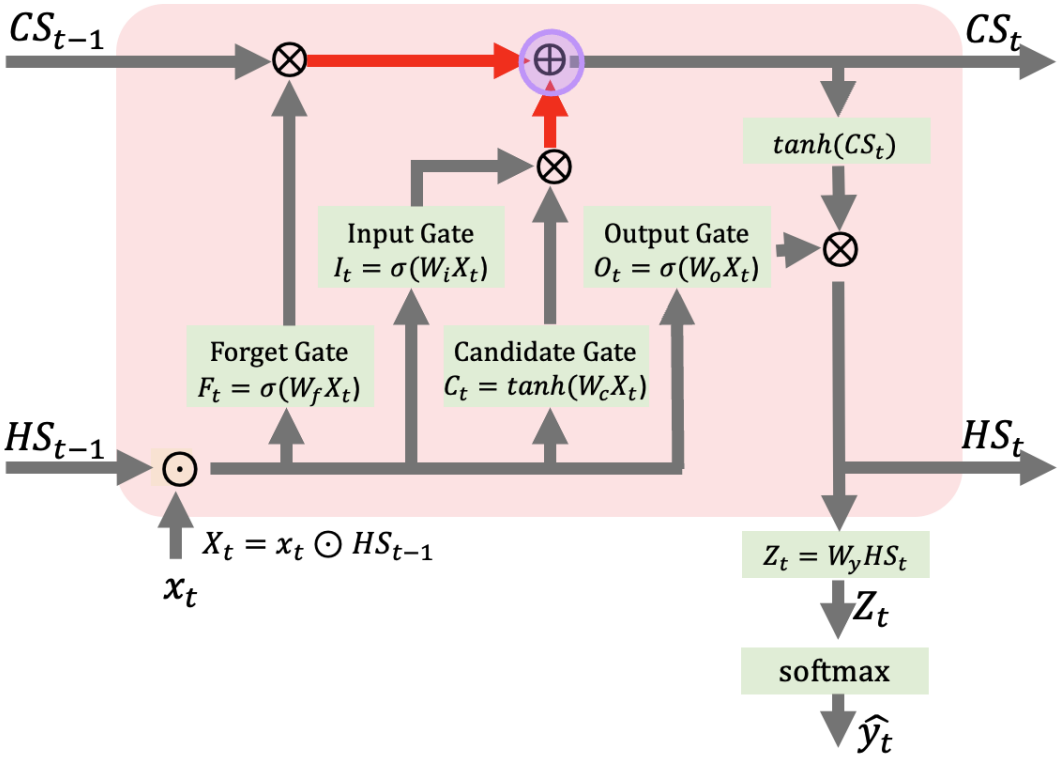
Those kept values



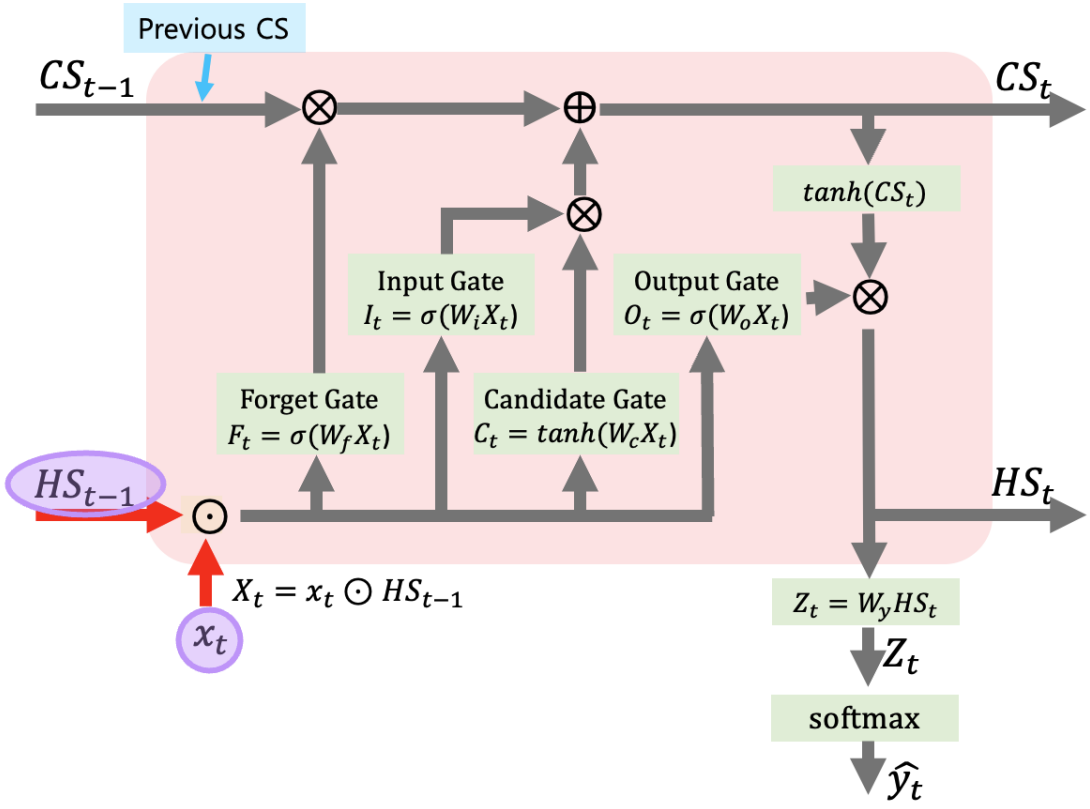
become the parts of the current input (short-term) to be remembered.



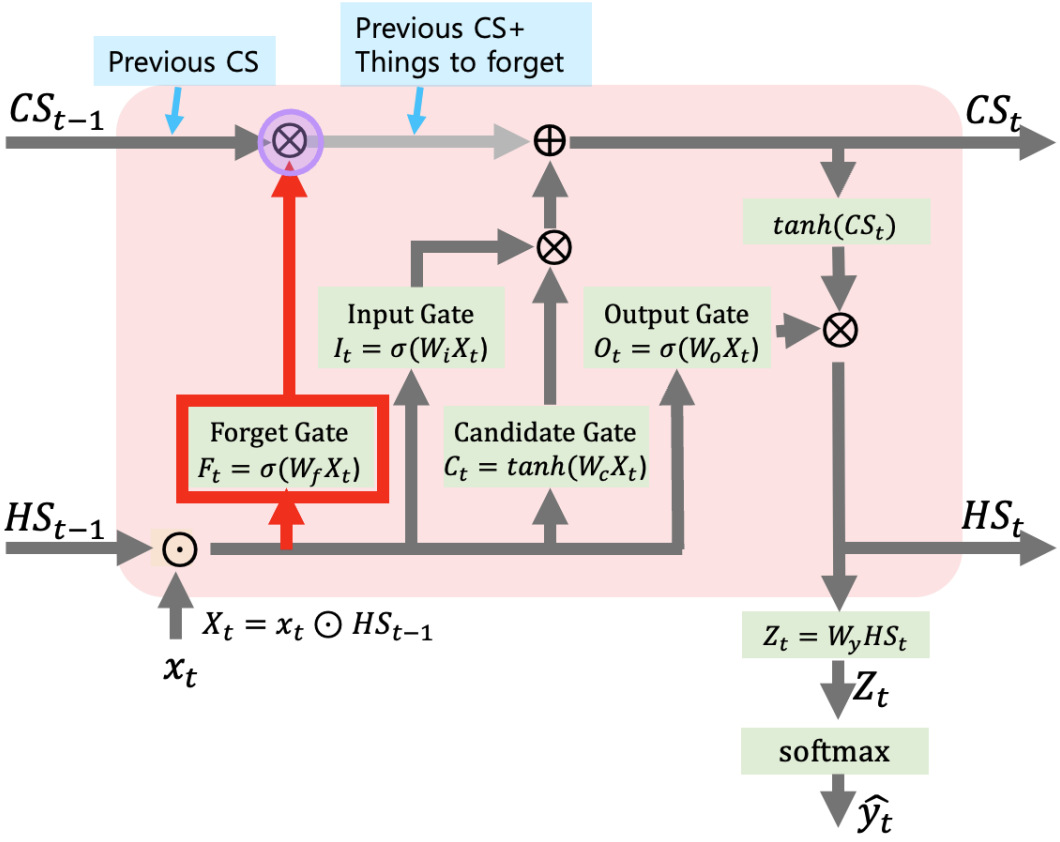
Then the remaining values are added into the cell state to update it.



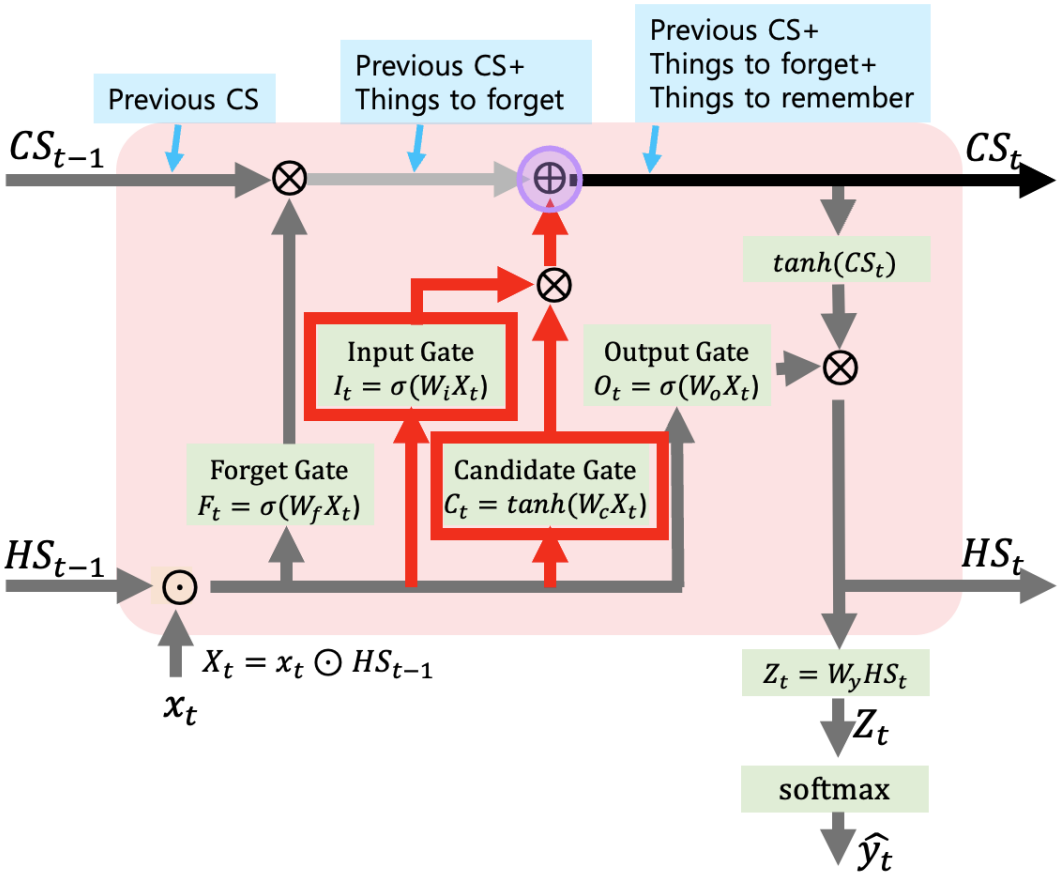
In short, given the previous hidden state and the current input,



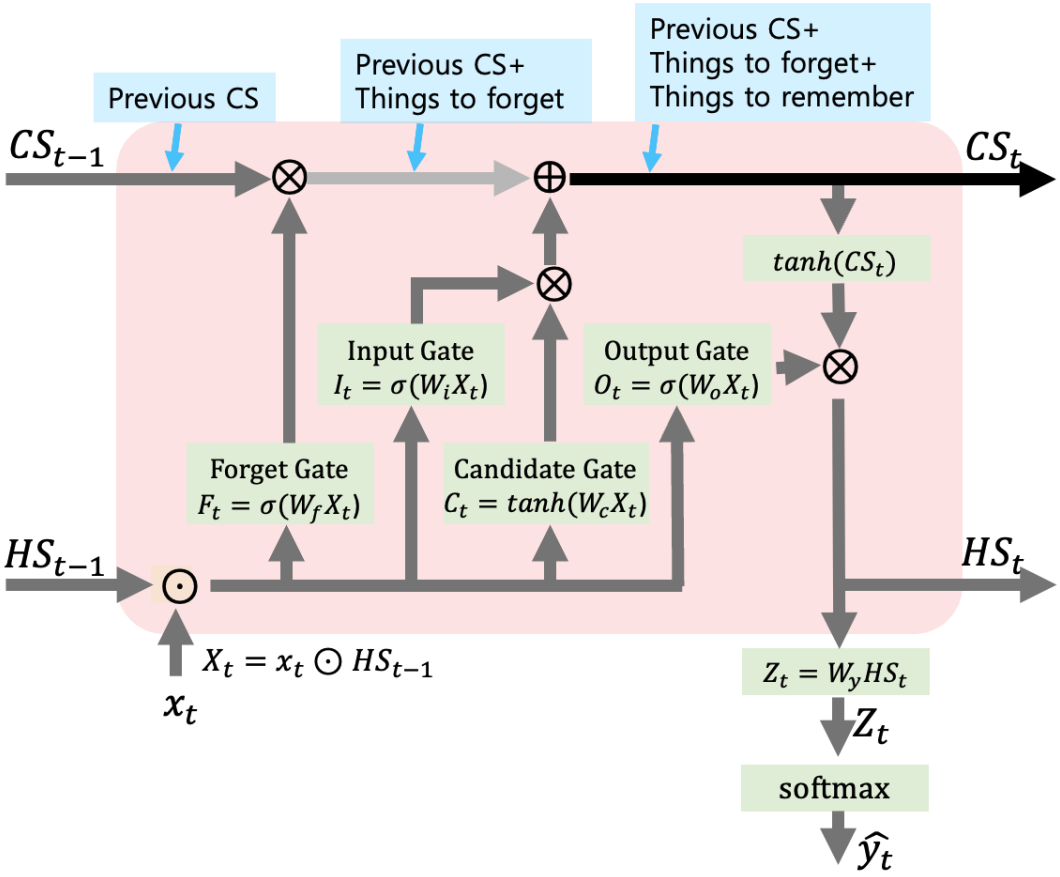
we forget what should be forgotten from the previous cell state,



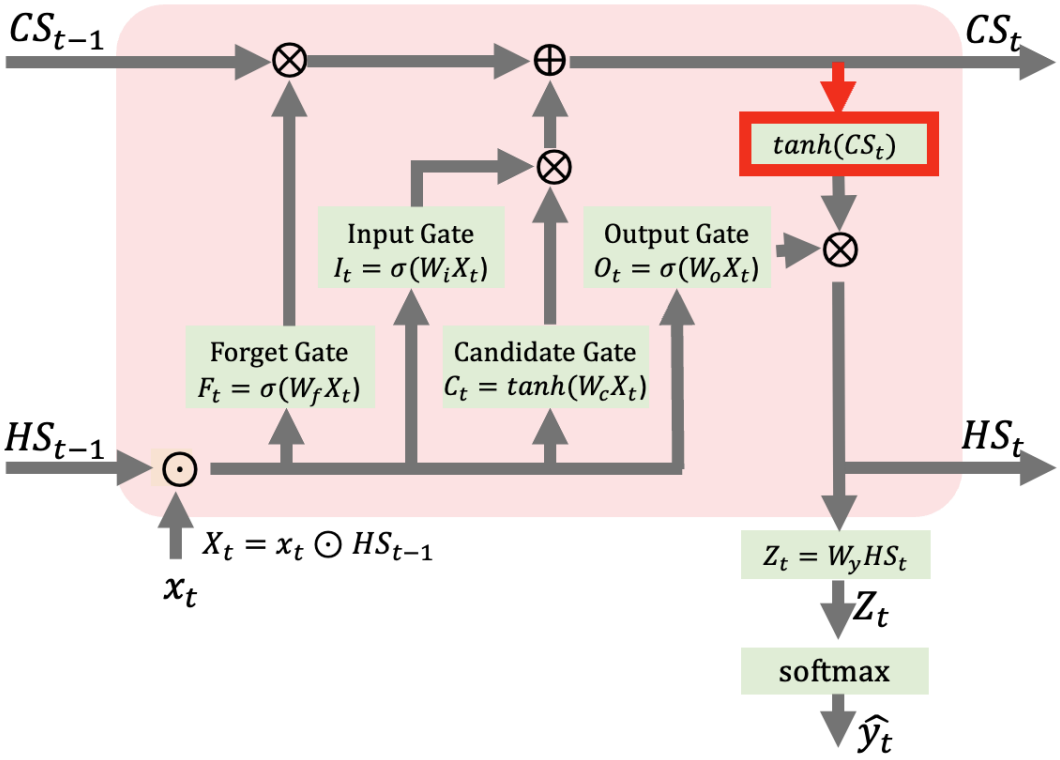
and remember what should be remembered,



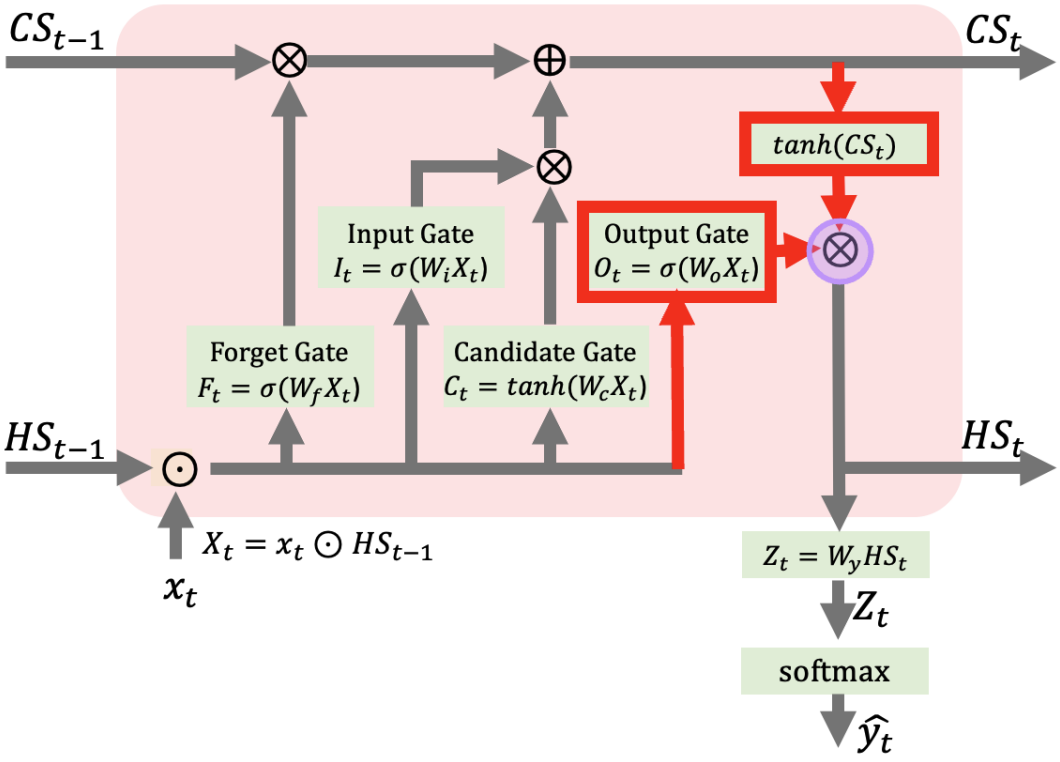
thereby updating LSTM's long-term memory (the cell state).



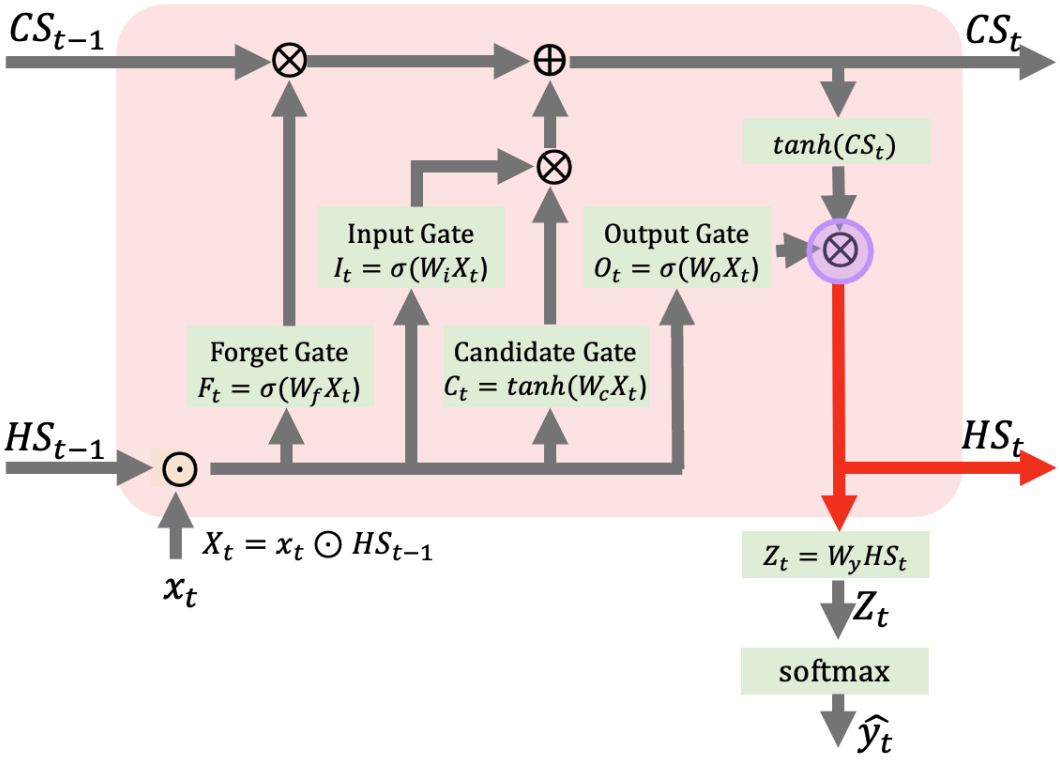
Next, we normalize this long-term state via tanh (to $[-1, 1]$),



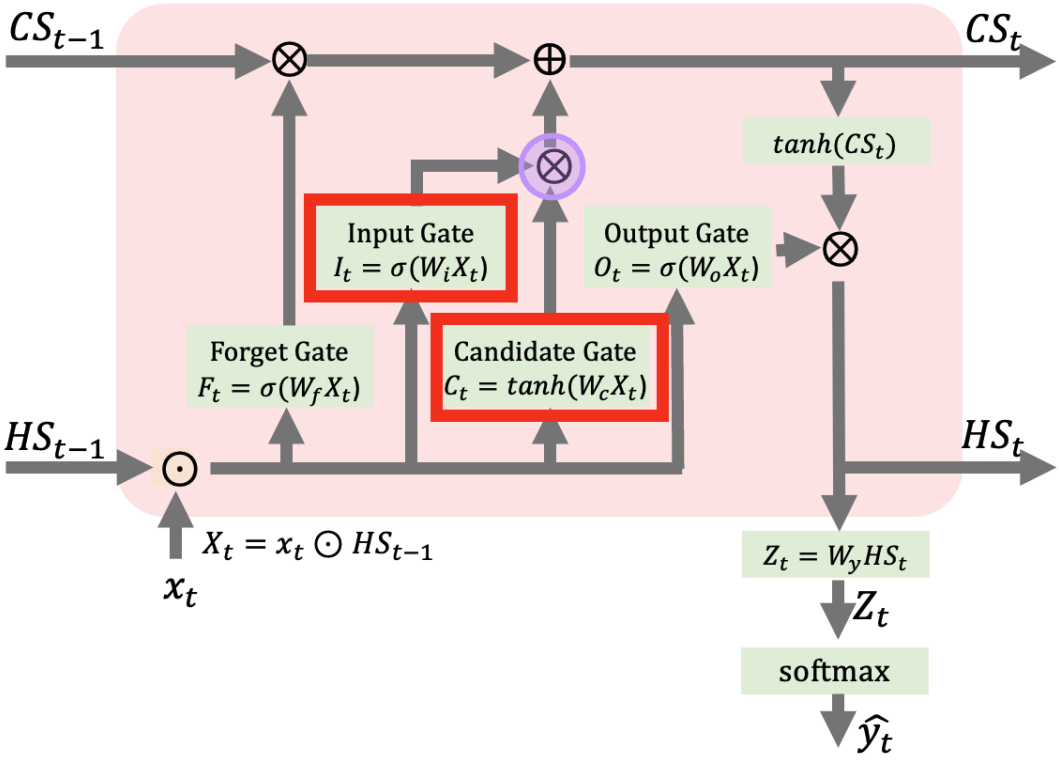
and take an element-wise product with the Output Gate's values.



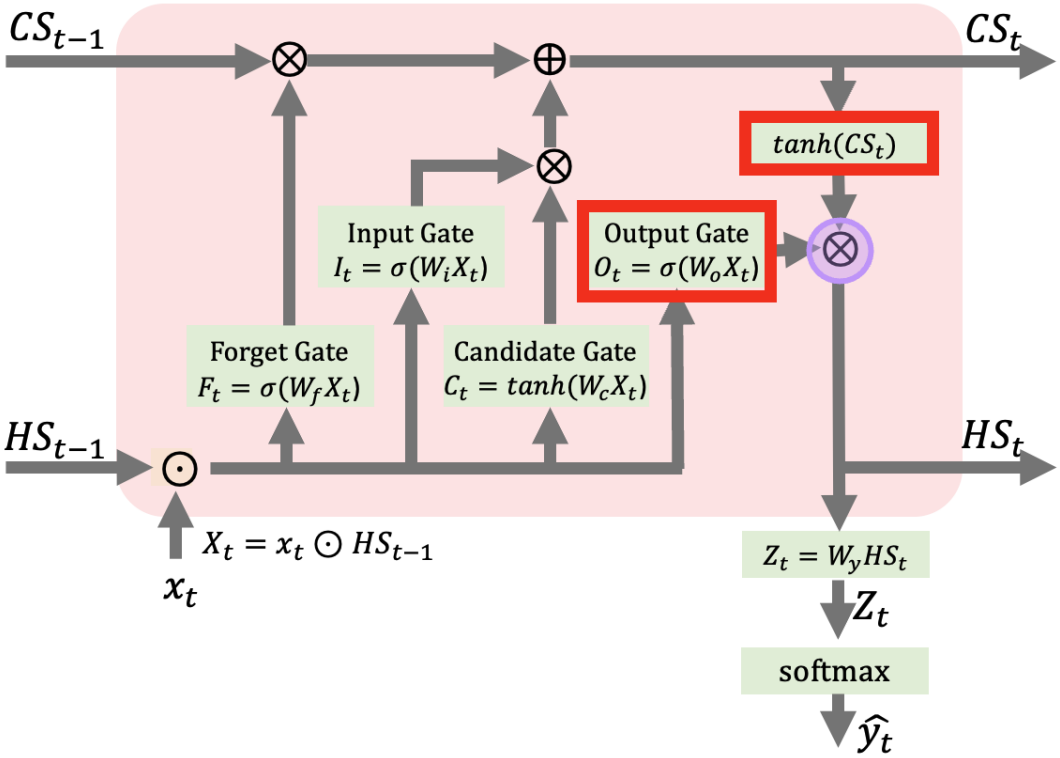
This produces the new hidden state HS_t.



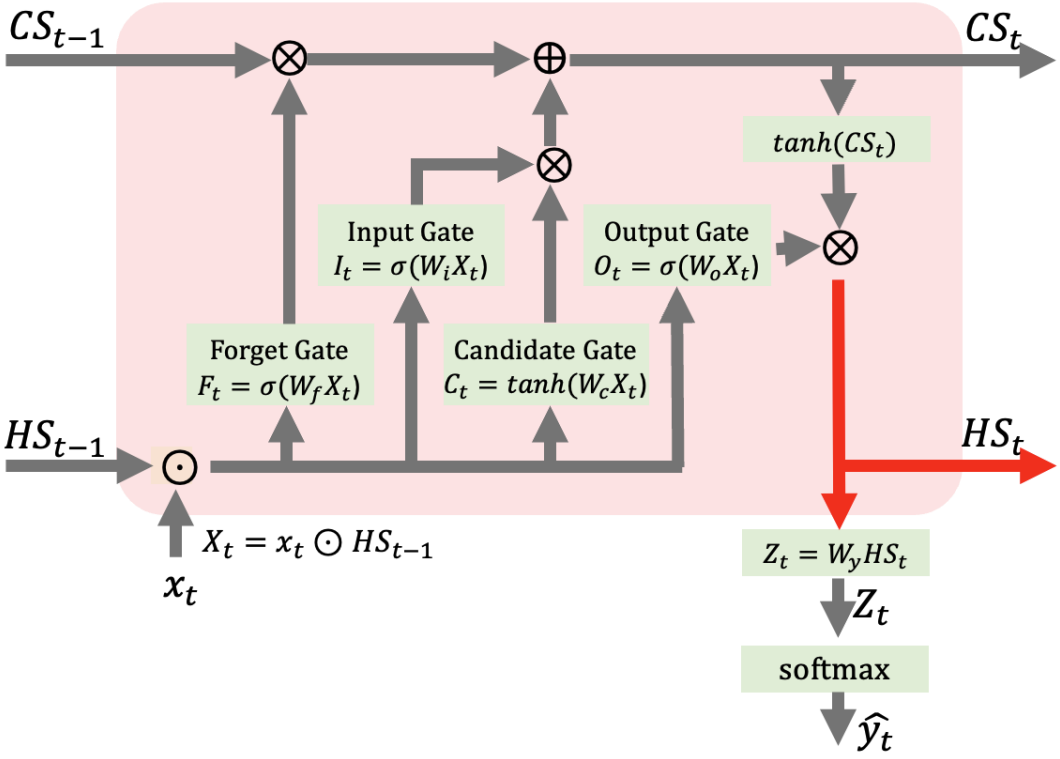
Just as the collaboration of the Input Gate and Candidate Gate keeps the "to-be-remembered" part of the current (short-term) input,



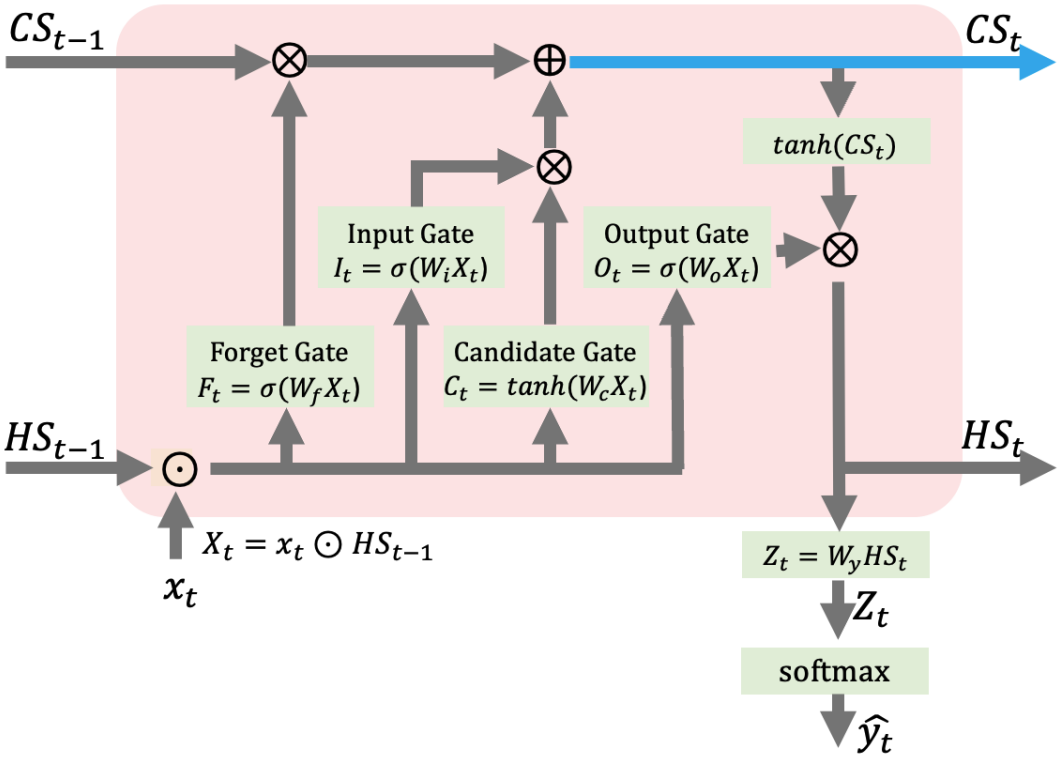
the collaboration of the Output Gate and $\tanh(CS_t)$ creates a new hidden state (HS_t) from the updated cell state (CS_t) that reflects the characteristics of the current input (X_t) more strongly.



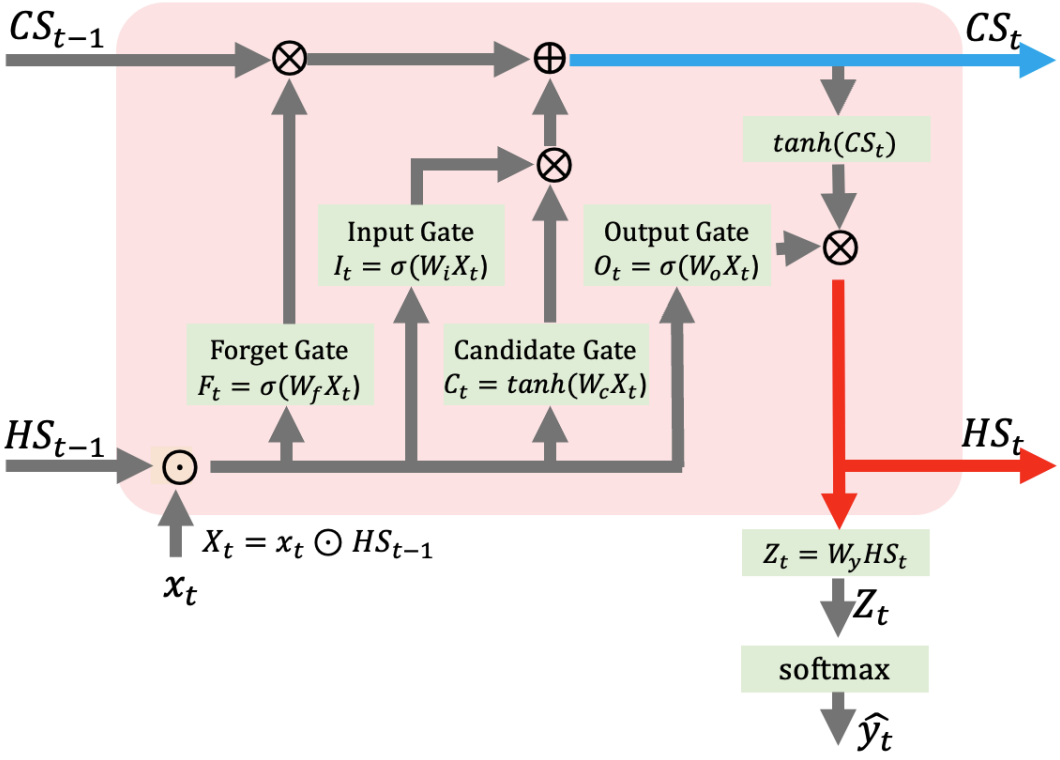
Thus this hidden state (HS_t) tends to show more short-term characteristics than CS_t.



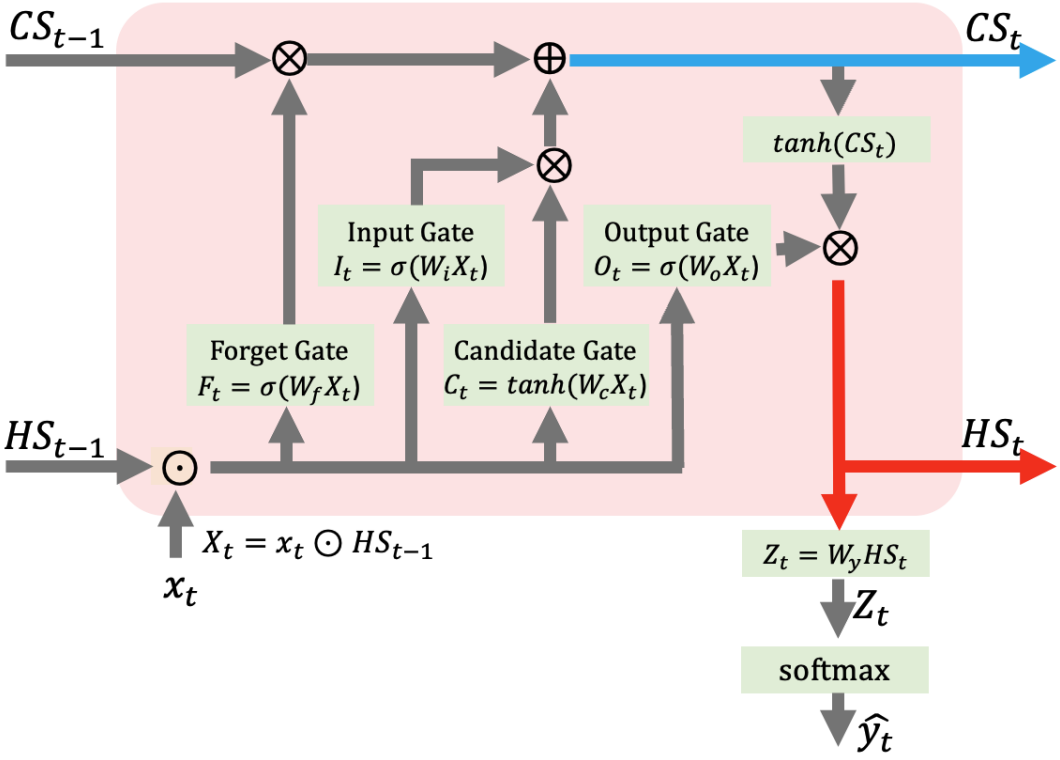
So if CS_t carries more long-term information,



HS_t, given the same inputs, carries information closer to short-term,



and by leveraging these two information flows, LSTM can **handle long-term dependency problems more effectively** than a vanilla RNN.



2. Real-world success in NLP tasks

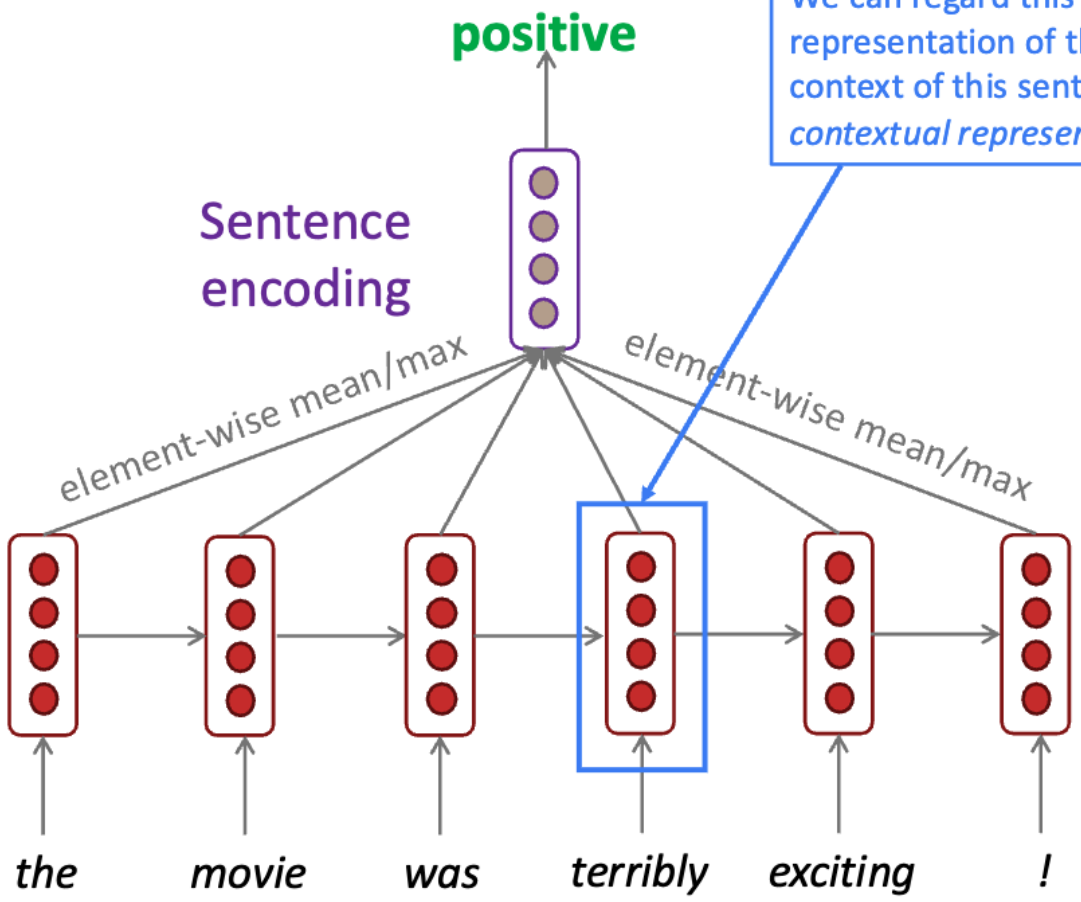
- Introduced by Hochreiter & Schmidhuber (1997)
- In 2013-2025, LSTMs started achieving state-of-the-art results in NLP tasks, including:
  - handwriting recognition, speech recognition, machine translation, parsing, and image captioning, as well as language modeling
- Recently (2019-2025), Transformers have become dominant for all tasks
  - For example, in WMT (a Machine Translation conference/competition):
    - In WMT 2014, there were 0 neural machine translation systems
    - In WMT 2016, the summary report contains RNN 44 times (and these systems won)
    - In WMT 2019: RNN 7 times, Transformer 105 times

Outline

1. Motivation

- A standard RNN only uses **past context**.
- For many NLP tasks (e.g., tagging, parsing, translation), knowing **both previous and future contexts** improves predictions.
- **Bidirectional** RNNs address this by processing the sequence in both directions.

Task: Sentiment Classification

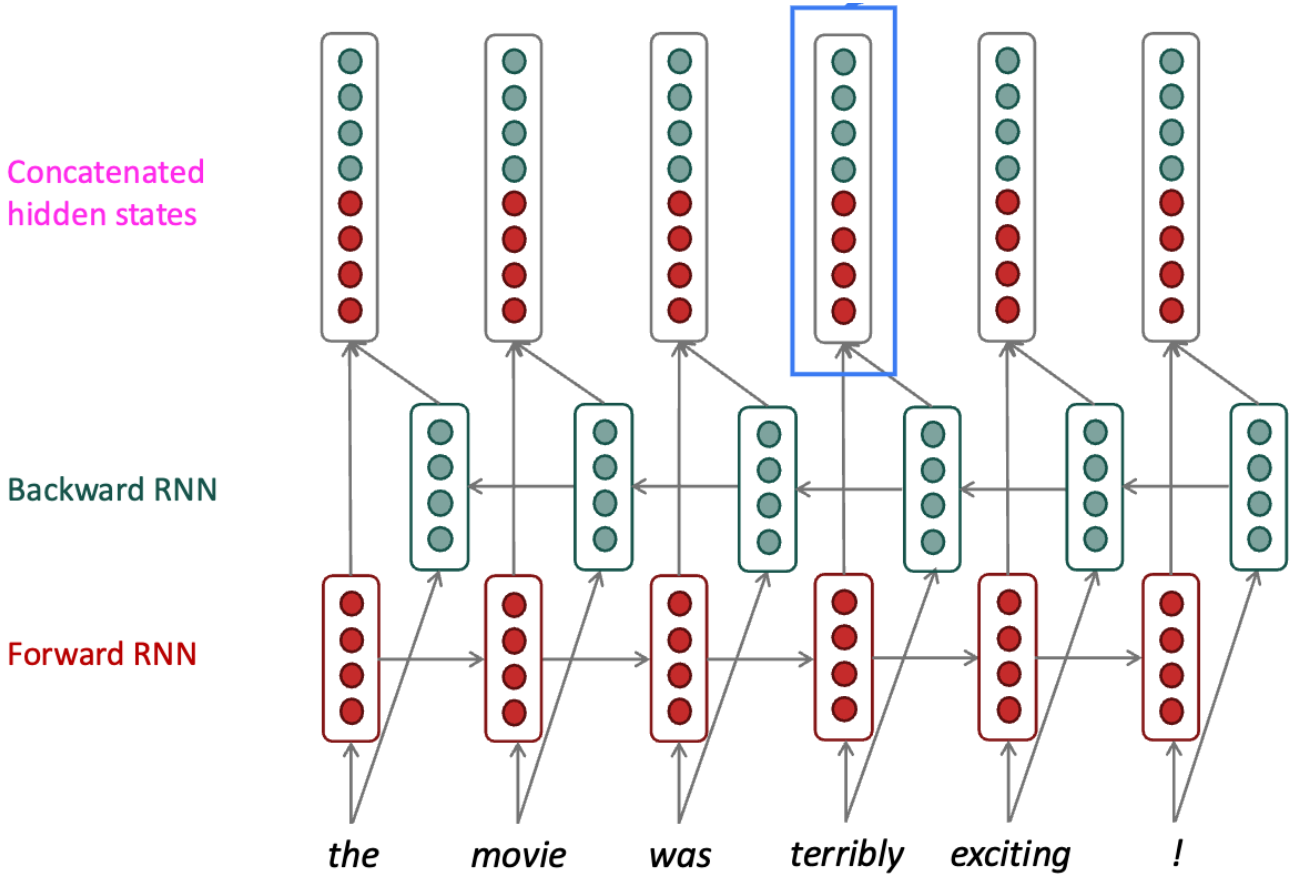


We can regard this hidden state as a representation of the word *"terribly"* in the context of this sentence. We call this a *contextual representation*.

These contextual representations only contain information about the *left* context (e.g. *"the movie was"*).

What about *right* context?

In this example, *"exciting"* is in the right context and this modifies the meaning of *"terribly"* (from negative to positive)



Forward + Backward: The contextual representation of "terribly" has both left and right context.

On timestep *t*:

This is a general notation to mean "compute one forward step of the RNN" – it could be a simple RNN or LSTM computation.

Forward RNN $\overrightarrow{\boldsymbol{h}}^{(t)} = \mathrm{RNN}_{\mathrm{FW}}(\overrightarrow{\boldsymbol{h}}^{(t-1)}, \boldsymbol{x}^{(t)})$

Backward RNN $\overleftarrow{\boldsymbol{h}}^{(t)} = \mathrm{RNN}_{\mathrm{BW}}(\overleftarrow{\boldsymbol{h}}^{(t+1)}, \boldsymbol{x}^{(t)})$

Generally, these two RNNs have separate weights

Concatenated hidden states $\boldsymbol{h}^{(t)} = [\overrightarrow{\boldsymbol{h}}^{(t)}; \overleftarrow{\boldsymbol{h}}^{(t)}]$

We regard this as "the hidden state" of a bidirectional RNN. This is what we pass on to the next parts of the network.

2. More information

- Hidden states combine forward and backward context.
- Require access to the entire sequence (not suitable for language modeling).
- Very effective for encoding tasks (e.g., tagging, parsing, translation).
- Example: BERT (Bidirectional Encoder Representations from Transformers) leverages bidirectionality for powerful contextual embeddings.
- Can be extended by stacking layers (Multi-layer RNNs).

Presentation

Shubh Sehgal - Vaswani et al. (2017). *Attention Is All You Need*.

Outline

On Thursday

- Presentation: Billy - Huang et al. (2018). *Music Transformer*.
- Work on Lab 6 - Hugging face