Gated Recurrent Units

Gated Recurrent Unit (GRU)

Paying attention to a sequence

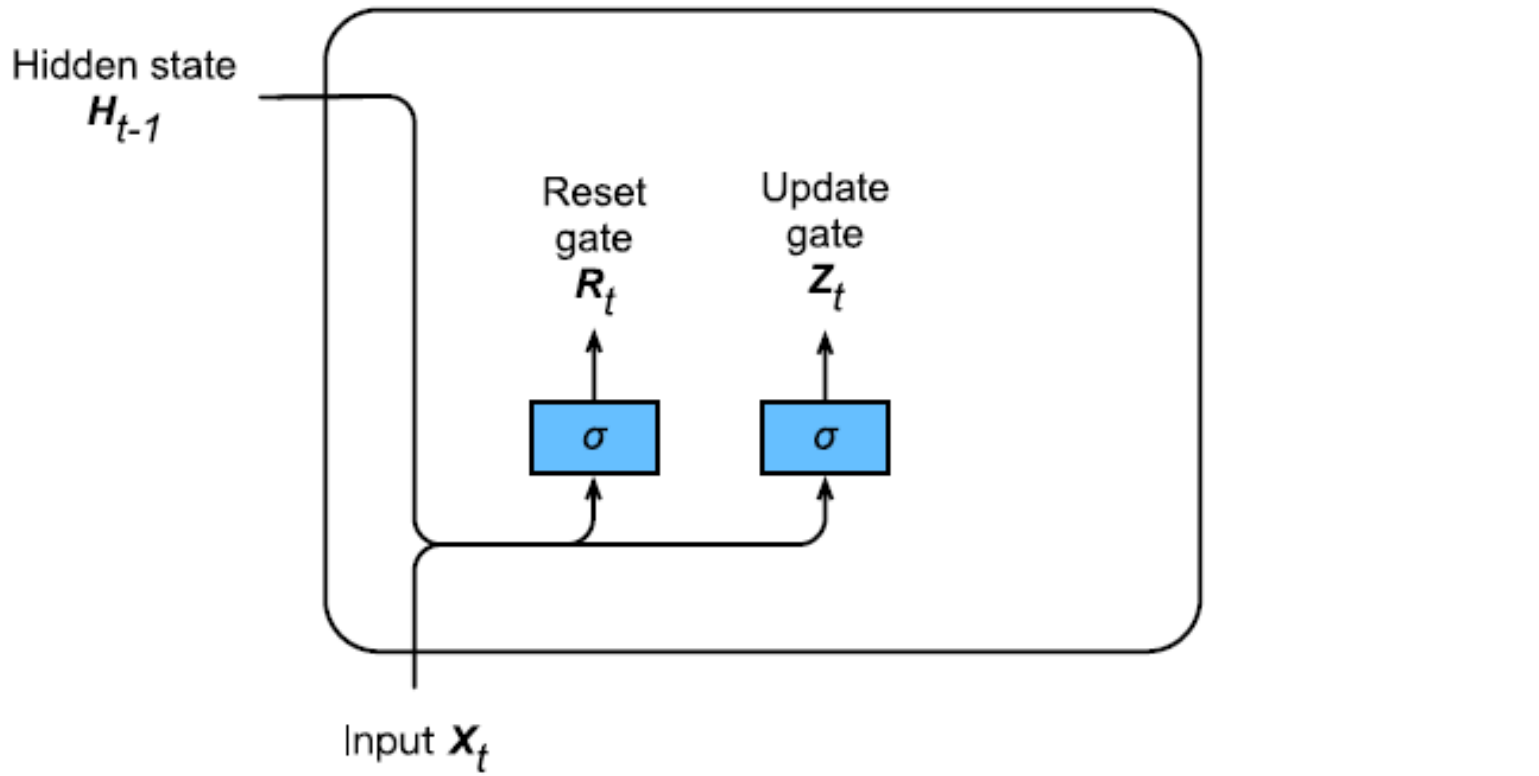
- Not all observations are equally relevant

- Only remember the relevant ones
  - Need mechanism to **pay attention (update gate)**
  - Need mechanism to **forget (reset gate)**

Gating

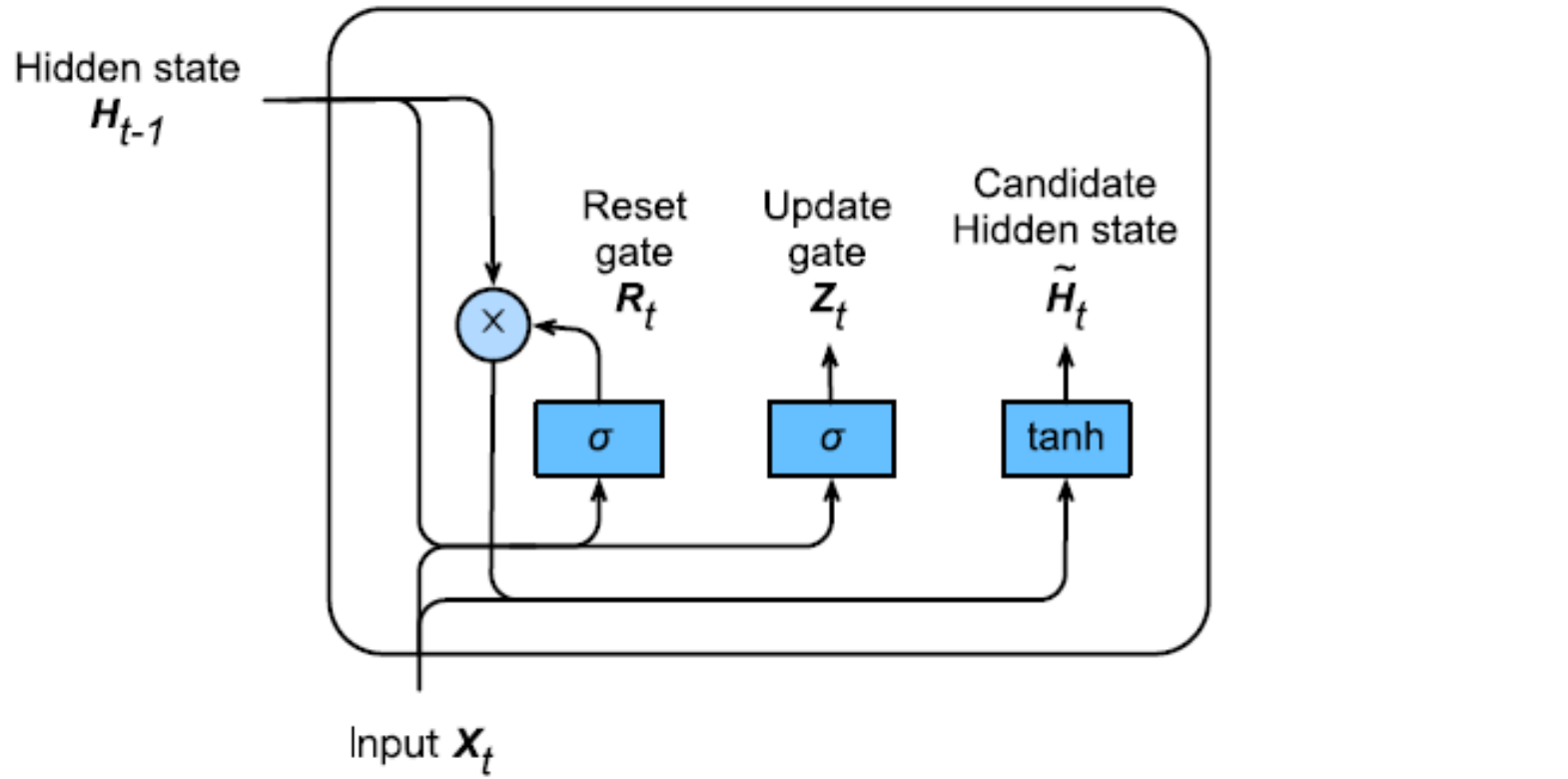
$$\boldsymbol{R}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xr} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hr} + \boldsymbol{b}_r),$$

$$\boldsymbol{Z}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xz} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hz} + \boldsymbol{b}_z)$$



Candidate Hidden State

$$\tilde{\boldsymbol{H}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xh} + \left(\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}\right) \boldsymbol{W}_{hh} + \boldsymbol{b}_h)$$



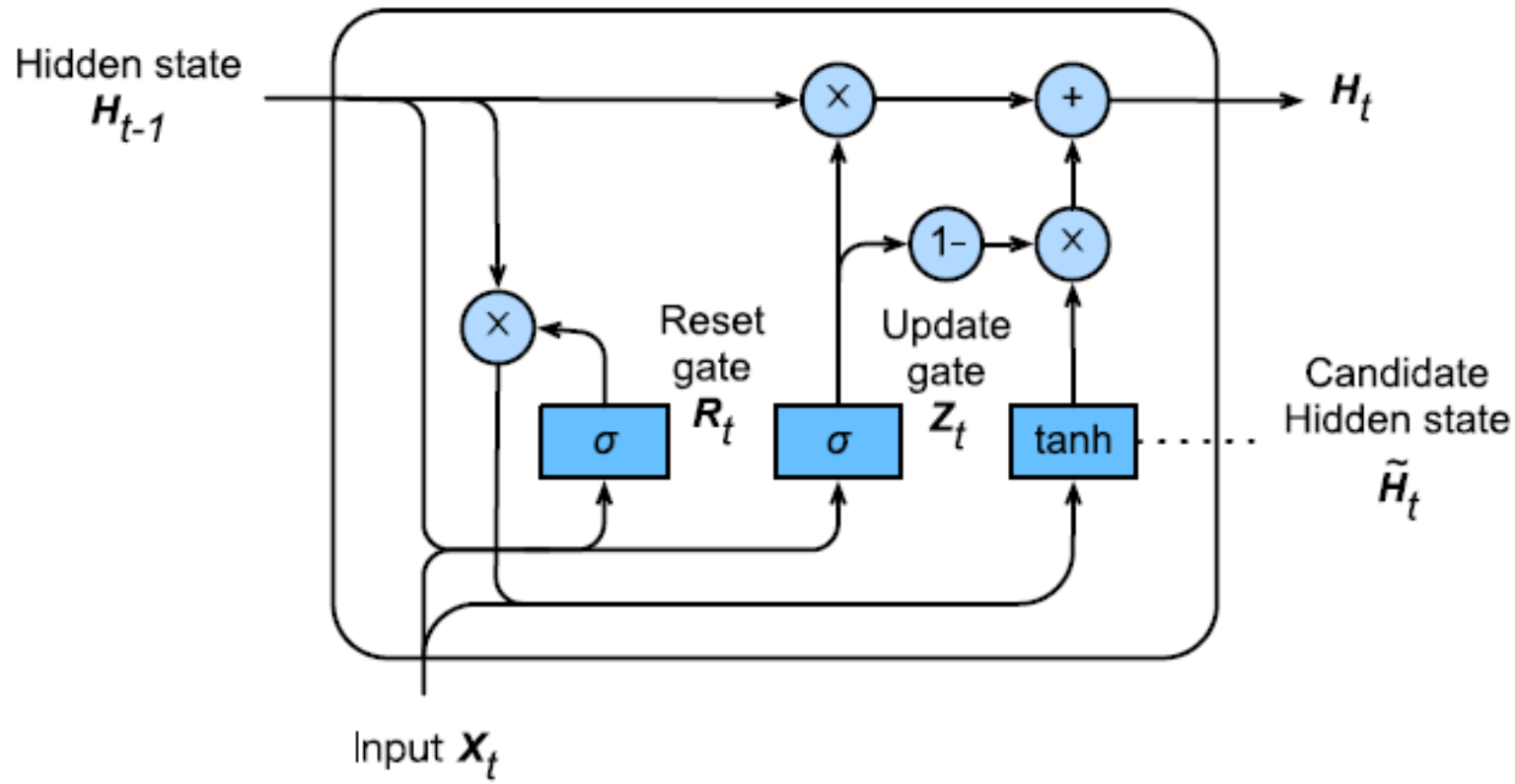
Gate

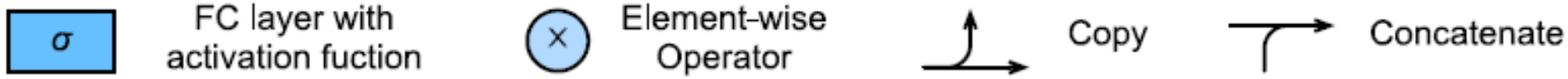
$$R_t \times H_{t-1}$$

$$\begin{bmatrix} 0.3 \\ 0 \\ 0.6 \\ 0.4 \\ 0.1 \\ 1 \\ 0 \\ 0.5 \end{bmatrix} \times \begin{bmatrix} 2 \\ 3 \\ 1 \\ 5 \\ 3 \\ 0.4 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.6 \\ 0 \\ 0.6 \\ 2 \\ 0.3 \\ 0.4 \\ 0 \\ 0.5 \end{bmatrix}$$

Hidden State

$$H_t = Z_t \odot H_{t-1} + (1 - Z_t) \odot \tilde{H}_t$$





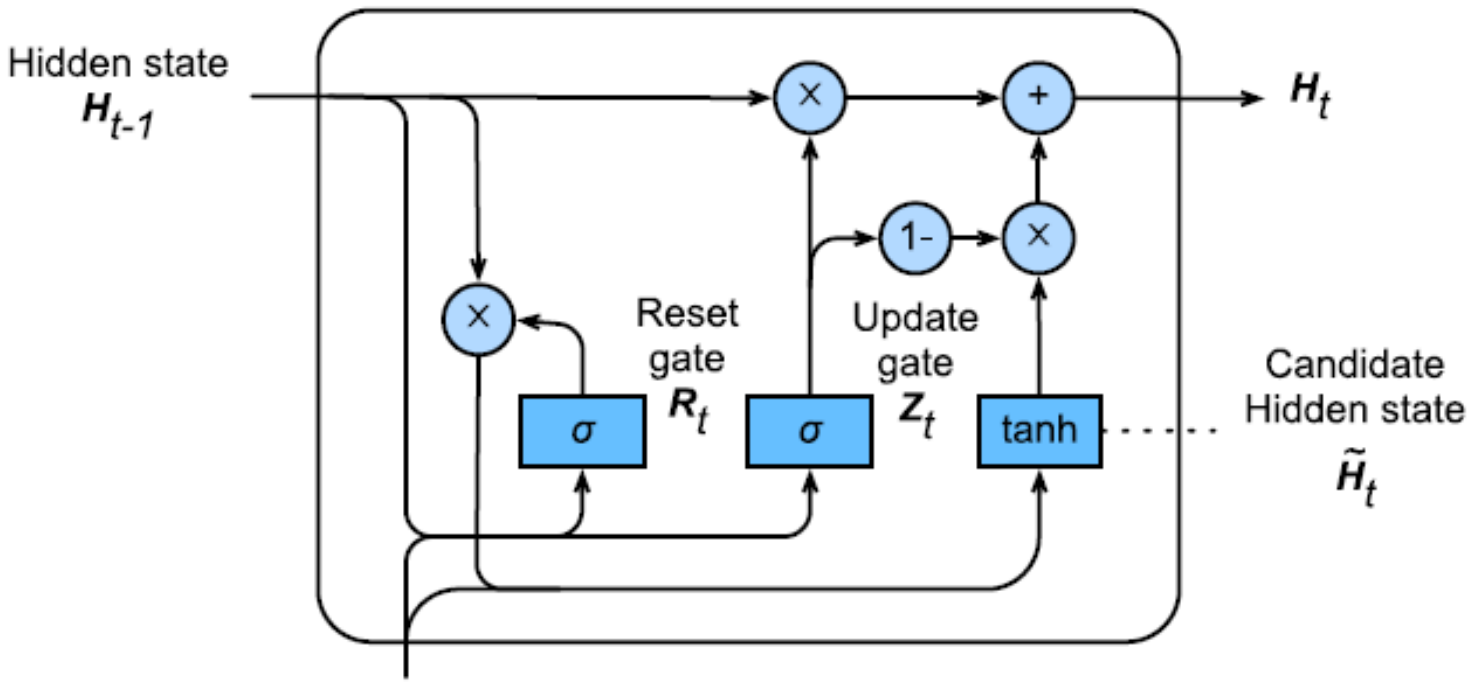
Summary

$$\boldsymbol{R}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xr} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hr} + \boldsymbol{b}_r),$$

$$\boldsymbol{Z}_t = \sigma(\boldsymbol{X}_t \boldsymbol{W}_{xz} + \boldsymbol{H}_{t-1} \boldsymbol{W}_{hz} + \boldsymbol{b}_z)$$

$$\tilde{\boldsymbol{H}}_t = \tanh(\boldsymbol{X}_t \boldsymbol{W}_{xh} + \left(\boldsymbol{R}_t \odot \boldsymbol{H}_{t-1}\right) \boldsymbol{W}_{hh} + \boldsymbol{b}_h)$$

$$\boldsymbol{H}_t = \boldsymbol{Z}_t \odot \boldsymbol{H}_{t-1} + (1 - \boldsymbol{Z}_t) \odot \tilde{\boldsymbol{H}}_t$$



LSTM

Long Short-Term Memory (LSTM)

- A type of RNN proposed by Hochreiter and Schmidhuber in 1997 as a solution to the vanishing gradients problem.

- 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 erase, write and read information from the cell

- 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 (LSTM)

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

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

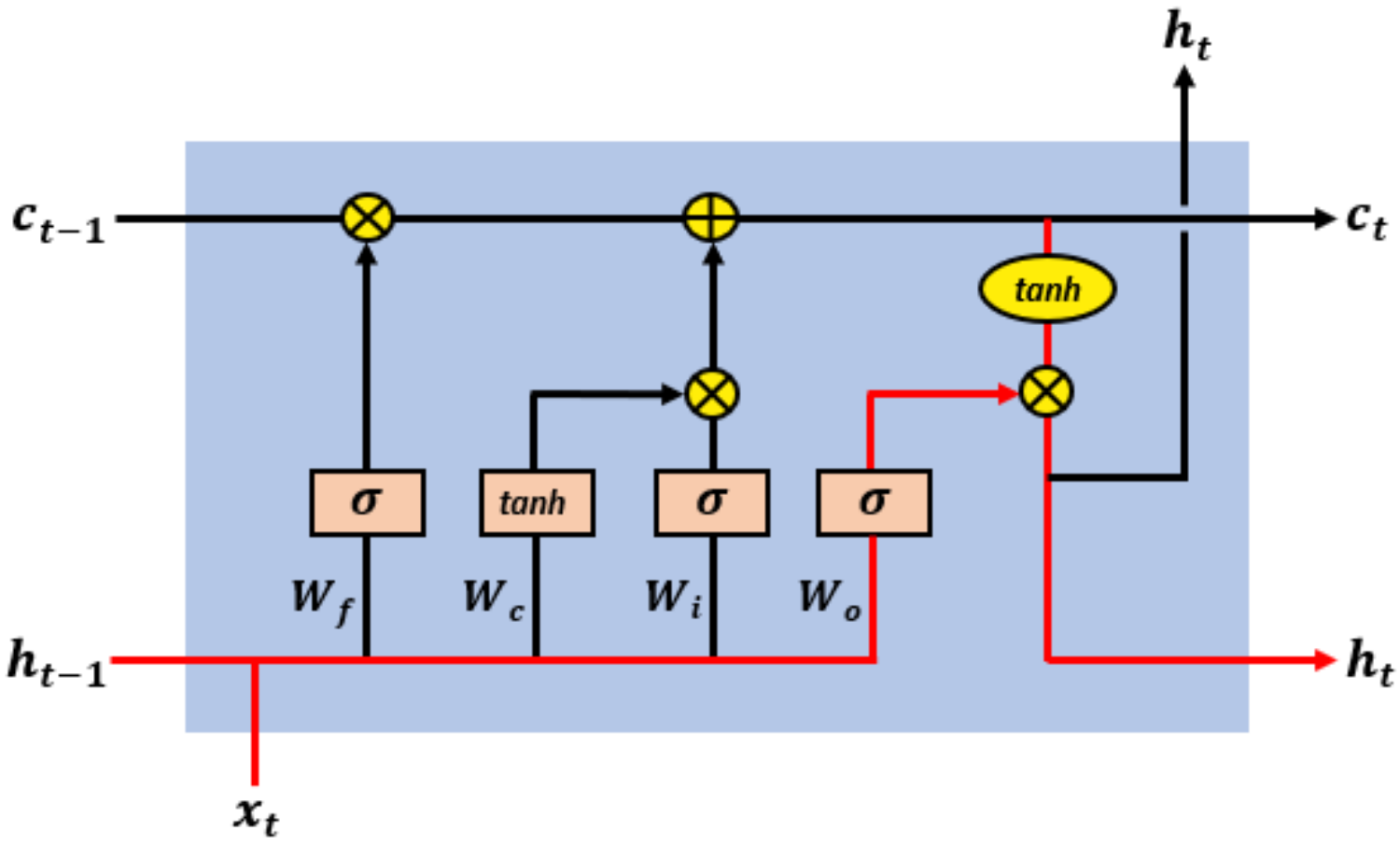
$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)}$$

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

Gates are applied using element-wise product

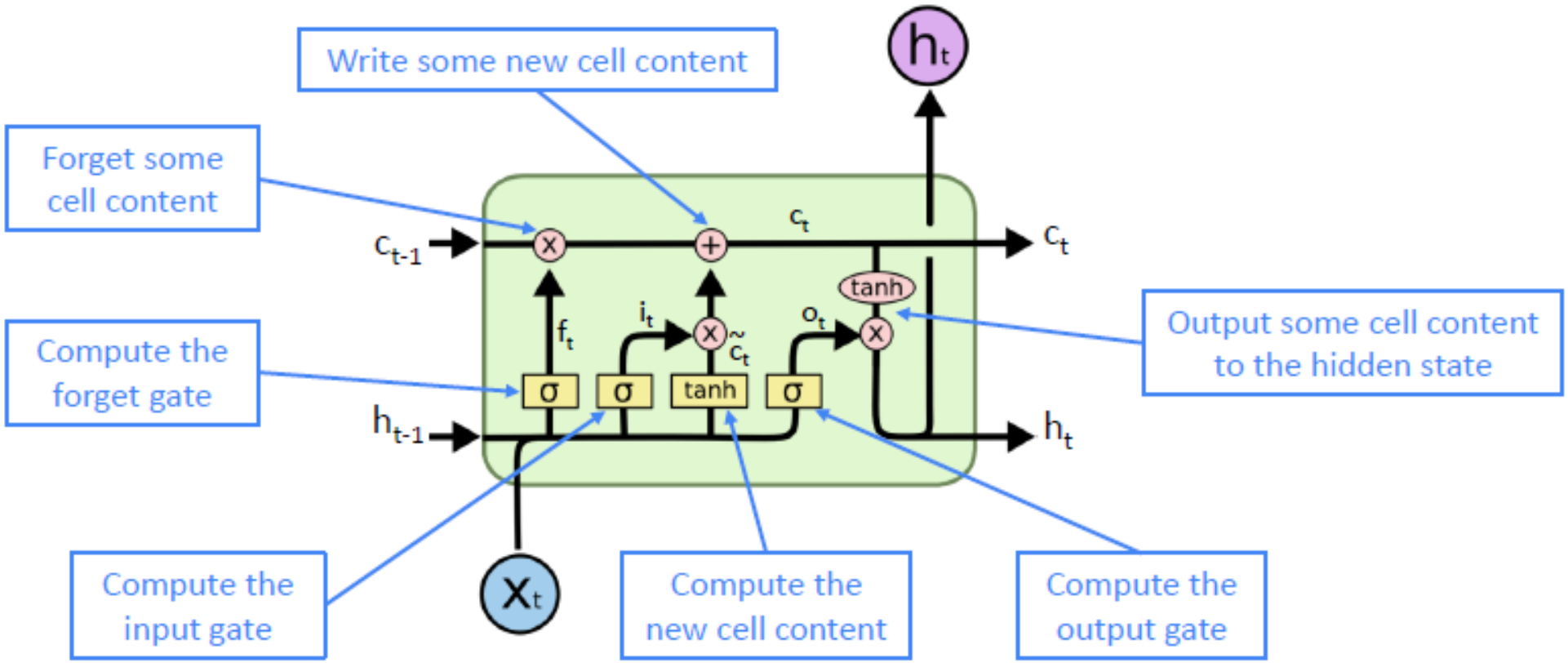
All these are vectors of same length n

The LSTM output gate's action on the cell state



Long Short-Term Memory (LSTM)

You can think of the LSTM equations visually like this:

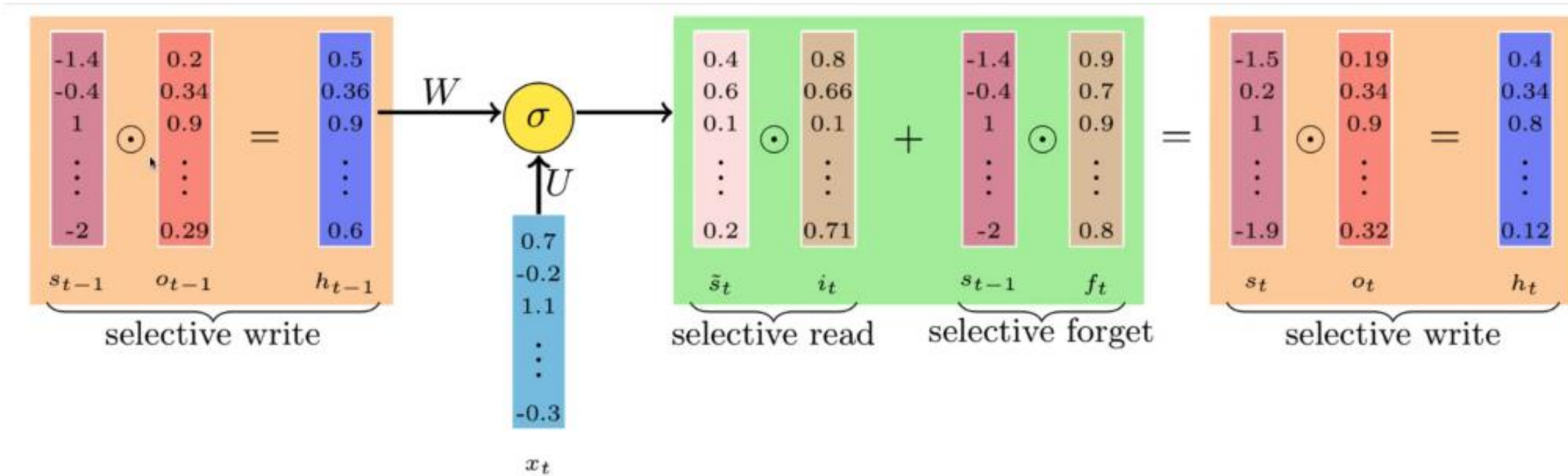


Gates:

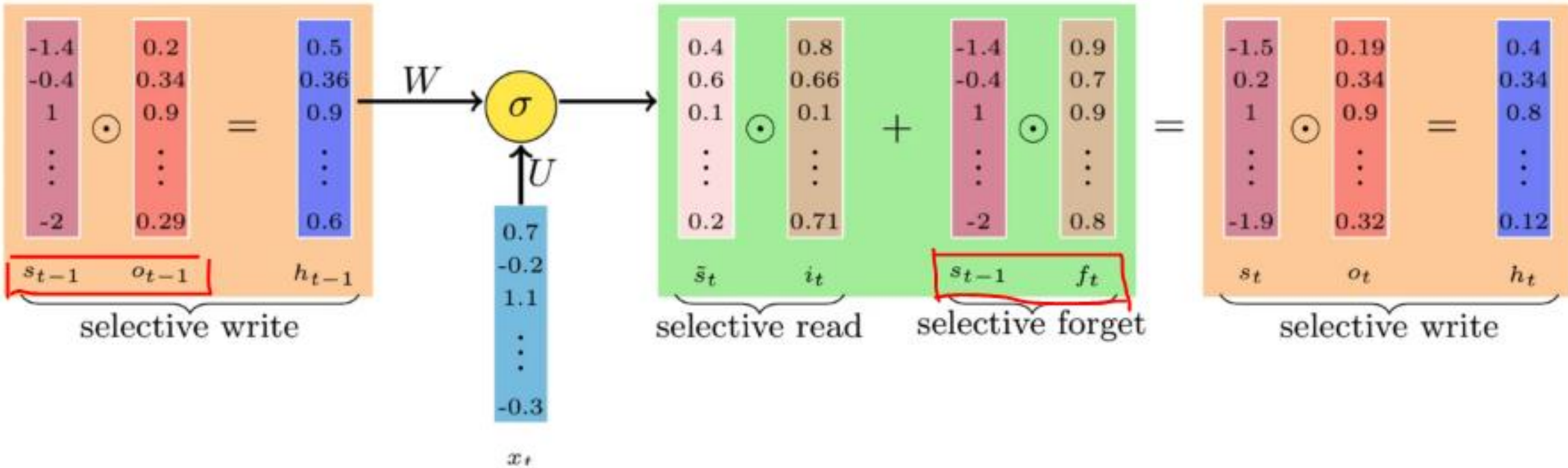
$$o_t = \sigma(W_o h_{t-1} + U_o x_t + b_o)$$
$$i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$$
$$f_t = \sigma(W_f h_{t-1} + U_f x_t + b_f)$$

States:

$$\tilde{s}_t = \sigma(W h_{t-1} + U x_t + b)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t$$
$$h_t = o_t \odot \sigma(s_t)$$



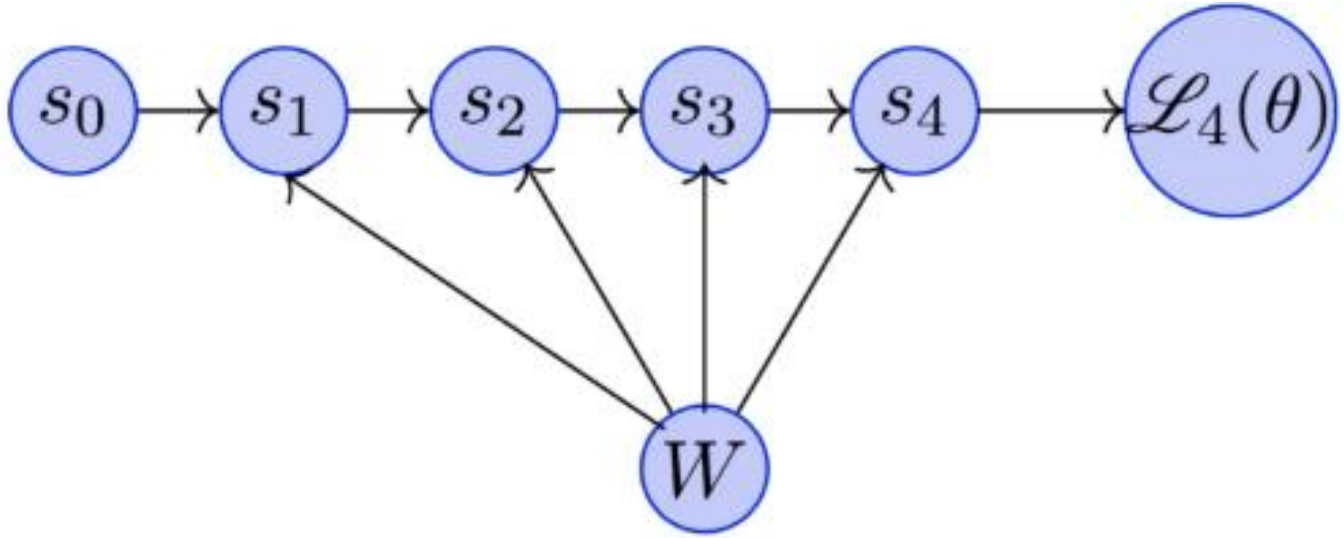
https://medium.com/@prvnk10/how-lstms-solve-the-problem-of-vanishing-gradients-ea88f08c78ca

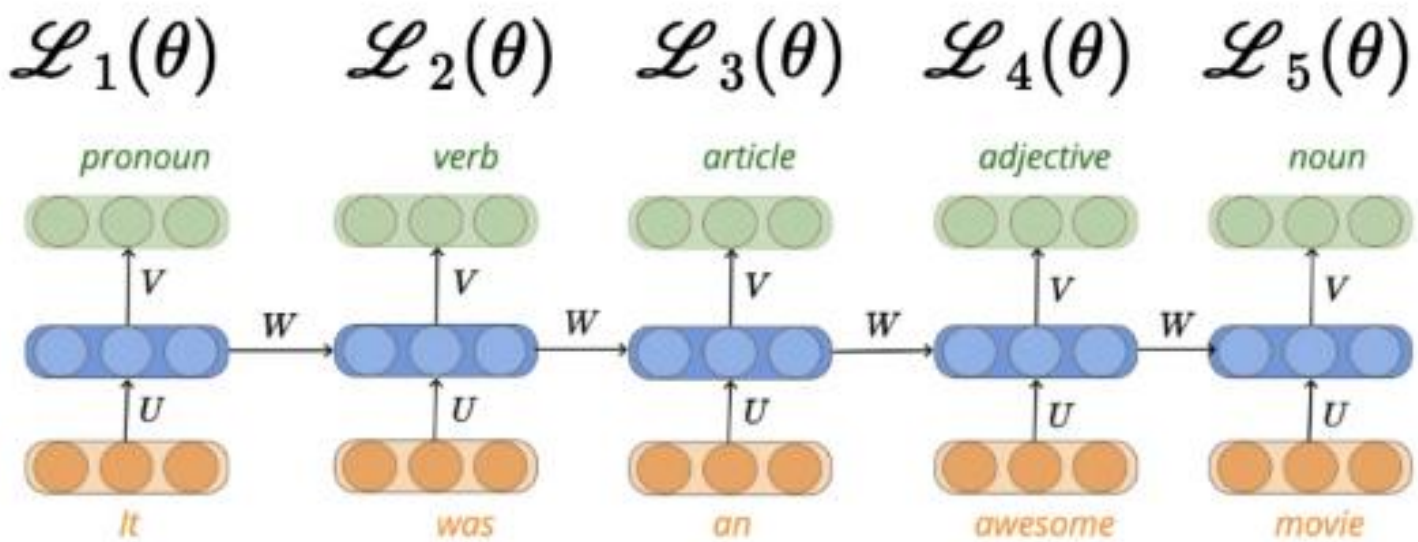


https://medium.com/@prvnk10/how-lstms-solve-the-problem-of-vanishing-gradients-ea88f08c78ca

If the state at time $t-1$ did not contribute much to the state at time t (i.e., if $\|f_t\| \to 0$ and $\|o_{t-1}\| \to 0$) then during backpropagation the gradients flowing into s_{t-1} will vanish

https://medium.com/@prvnk10/how-lstms-solve-the-problem-of-vanishing-gradients-ea88f08c78ca


s_0
s_1
s_2
s_3
s_4
$\mathscr{L}_4(\theta)$
W


$\mathscr{L}_1(\theta)$
$\mathscr{L}_2(\theta)$
$\mathscr{L}_3(\theta)$
$\mathscr{L}_4(\theta)$
$\mathscr{L}_5(\theta)$
pronoun
verb
article
adjective
noun
V
W
U
It
was
an
awesome
movie

In general, the gradient of $\mathscr{L}_t(\theta)$ w.r.t. θ_i vanishes when the gradients flowing through **each and every path** from $L_t(\theta)$ to θ_i vanish.

On the other hand, the gradient of $\mathscr{L}_t(\theta)$ w.r.t. θ_i explodes when the gradient flowing through **at least one path** explodes.

How LSTM Solves Vanishing Gradient Problem

$$o_k = \sigma(W_o h_{k-1} + U_o x_k + b_o)$$

$$i_k = \sigma(W_i h_{k-1} + U_i x_k + b_i)$$

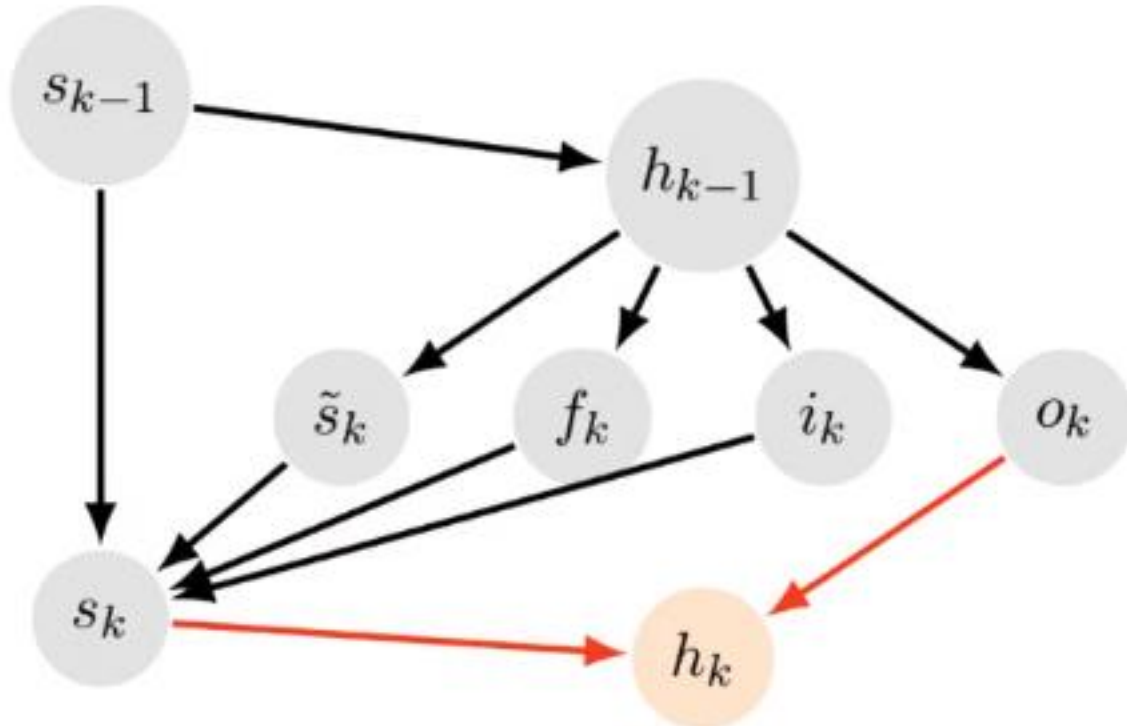
$$f_k = \sigma(W_f h_{k-1} + U_f x_k + b_f)$$

$$\tilde{s_k} = \sigma(W h_{k-1} + U x_k + b)$$

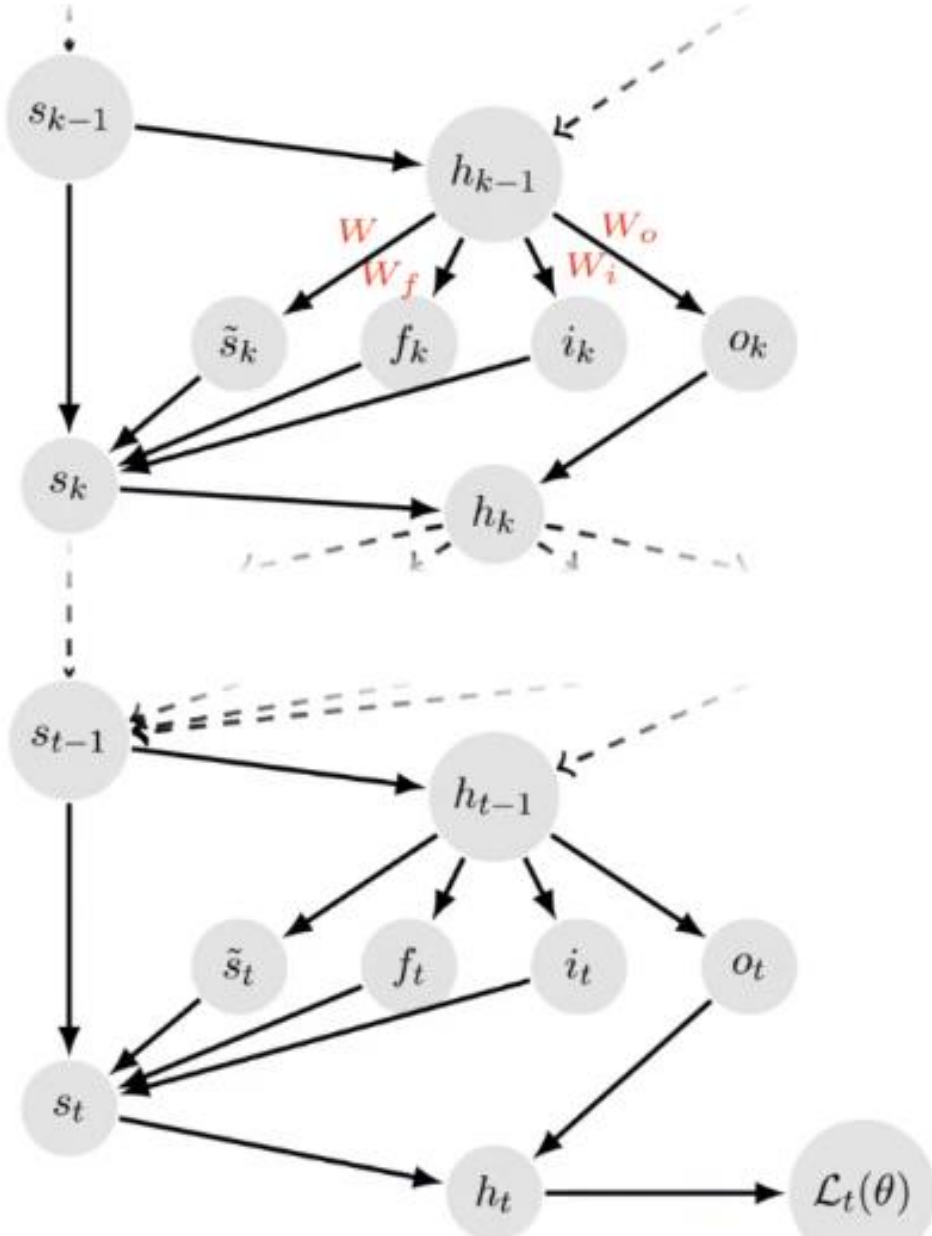
$$s_k = f_k \odot s_{k-1} + i_k \odot \tilde{s_k}$$

$$h_k = o_k \odot \sigma(s_k)$$

https://medium.com/@prvnk10/how-lstms-solve-the-problem-of-vanishing-gradients-ea88f08c78ca

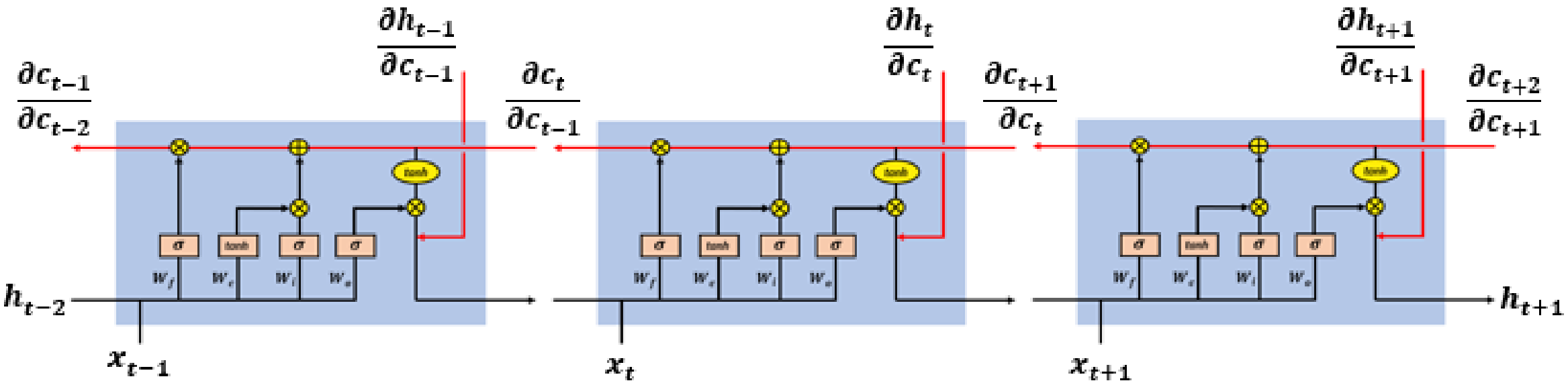


https://medium.com/@prvnk10/how-lstms-solve-the-problem-of-vanishing-gradients-ea88f08c78ca



https://medium.com/@prvnk10/how-lstms-solve-the-problem-of-vanishing-gradients-ea88f08c78ca

Back propagating through time for gradient computation



https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

As in RNNs, the error term gradient is given by the following sum of T gradients:

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W} \qquad (3)$$

For the complete error gradient to vanish, all of these T sub gradients need to vanish. If we think of (3) as a series of functions, then by definition, this series converges to zero if the sequence of its partial sums tends to zero,

$$\sum_{t=1}^{T} \frac{\partial E_t}{\partial W} \to 0$$

so if the series of partial sums

$$(S_1, S_2, S_3, \ldots)$$

where

$$S_n = \sum_{t=1}^{n} \frac{\partial E_t}{\partial W}$$

tends to zero

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W} \qquad (3)$$

So if we want (3) not to vanish, our network needs to increase the likelihood that at least some of these sub gradients will not vanish, in other words, make the series of sub gradients in (3) not converge to zero.

https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

The gradient of the error for some time step k has the form:

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\dots\frac{\partial c_2}{\partial c_1}\frac{\partial c_1}{\partial W}$$

$$= \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k}\frac{\partial c_t}{\partial c_{t-1}}\right)\frac{\partial c_1}{\partial W} \qquad (4)$$

the following product causes the gradients to vanish:

$$\prod_{t=2}^{k}\frac{\partial c_t}{\partial c_{t-1}}$$

https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

$$c_t = c_{t-1} \otimes f_t \oplus \tilde{c}_t \otimes i_t \qquad (5)$$

$$\frac{\partial c_t}{\partial c_{t-1}} = \frac{\partial}{\partial c_{t-1}} [c_{t-1} \otimes f_t \oplus \tilde{c}_t \otimes i_t]$$

$$= \frac{\partial}{\partial c_{t-1}} [c_{t-1} \otimes f_t] + \frac{\partial}{\partial c_{t-1}} [\tilde{c}_t \otimes i_t]$$

$$= \frac{\partial f_t}{\partial c_{t-1}} \cdot c_{t-1} + \frac{\partial c_{t-1}}{\partial c_{t-1}} \cdot f_t + \frac{\partial i_t}{\partial c_{t-1}} \cdot \tilde{c}_t + \frac{\partial \tilde{c}_t}{\partial c_{t-1}} \cdot i_t$$

$$\frac{\partial c_t}{\partial c_{t-1}} = \sigma'\left(W_f \cdot [h_{t-1}, x_t]\right) \cdot W_f \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot c_{t-1}$$

$$+ f_t$$

$$+ \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot \tilde{c}_t$$

$$+ \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot i_t$$

$$A_t = \sigma'\left(W_f \cdot [h_{t-1}, x_t]\right) \cdot W_f \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot c_{t-1}$$

$$B_t = f_t$$

$$C_t = \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot \tilde{c}_t$$

$$D_t = \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot i_t$$

$$\frac{\partial c_t}{\partial c_{t-1}} = A_t + B_t + C_t + D_t \qquad (6)$$

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k}[A_t + B_t + C_t + D_t]\right)\frac{\partial c_1}{\partial W}$$

Preventing the error gradients from vanishing

Notice that the gradient contains the forget gate's vector of activations, which allows the network to better control the gradients values, at each time step, using suitable parameter updates of the forget gate.

$$\frac{\partial c_t}{\partial c_{t-1}} = \sigma'\left(W_f \cdot [h_{t-1}, x_t]\right) \cdot W_f \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot c_{t-1}$$

$$+ f_t$$

$$+ \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot \tilde{c}_t$$

$$+ \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot i_t$$

https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

Say that for some time step k<T, we have that:

$$\sum_{t=1}^{k} \frac{\partial E_t}{\partial W} \rightarrow 0$$

Then for the gradient not to vanish, we can find a suitable parameter update of the forget gate at time step k+1 such that:

$$\frac{\partial E_{k+1}}{\partial W} \not\rightarrow 0$$

It is the presence of the forget gate's vector of activations in the gradient term along with additive structure which allows the LSTM to find such a parameter update at any time step, and this yields:

$$\sum_{t=1}^{k+1} \frac{\partial E_t}{\partial W} \not\rightarrow 0$$

https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

Another important property to notice is that the cell state gradient is an additive function

$$A_t \approx \overrightarrow{0.1},\ B_t = f_t \approx \overrightarrow{0.7},\ \mathrm{C_t} \approx \overrightarrow{0.1},\ \mathrm{D_t} \approx \overrightarrow{0.1}$$

$$\prod_{t=2}^{k}[A_t + B_t + C_t + D_t] \approx \prod_{t=2}^{k}\left[\overrightarrow{0.1} + \overrightarrow{0.7} + \overrightarrow{0.1} + \overrightarrow{0.1}\right]$$

$$\approx \prod_{t=2}^{k} \vec{1} \nrightarrow 0$$

https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W} \qquad (3)$$

In RNNs, the sum in (3) is made from expressions with a similar behavior that are likely to all be in [0,1] which causes vanishing gradients.

In LSTMs, however, the presence of the forget gate, along with the additive property of the cell state gradients, enables the network to update the parameter in such a way that the different sub gradients in (3) do not necessarily agree

$$\sum_{t=1}^{T} \frac{\partial E_t}{\partial W} \nrightarrow 0$$

https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577

Summary

- We have seen that RNNs suffer from vanishing gradients and caused by long series of multiplications of small values, diminishing the gradients and causing the learning process to become degenerate. In a analogues way, RNNs suffer from exploding gradients affected from large gradient values and hampering the learning process.
- LSTMs solve the problem using a unique additive gradient structure that includes direct access to the forget gate’s activations, enabling the network to encourage desired behavior from the error gradient using frequent gates update on every time step of the learning process.

Summary

The additive nature of LSTM has two advantages

- **First,** it is easy for each unit to remember the existence of specific feature in the input stream for a long series of steps.
- Any important feature will not be overwritten but be maintained as it is.
- **Second,** and perhaps more importantly, this addition effectively creates shortcut paths that bypass multiple temporal steps.
- Because of these shortcuts vanishing gradients problem will not occur

Is vanishing/exploding gradient just a RNN problem?

- No! It can be a problem for all neural architectures (including feed-forward and convolutional), especially deep ones.
  - Due to chain rule / choice of nonlinearity function, gradient can become vanishingly small as it backpropagates
  - Thus lower layers are learnt very slowly (hard to train)
  - Solution: lots of new deep feedforward/convolutional architectures that add more direct connections (thus allowing the gradient to flow)

- Conclusion: Though vanishing/exploding gradients are a general problem, RNNs are particularly unstable due to the repeated multiplication by the same weight matrix [Bengio et al, 1994]

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, image captioning
- LSTM became the dominant approach