Week 6 – Lesson 2: Long Short Term Memory (LSTM) Networks

Dr. Hongping Cai

Department of Computer Science

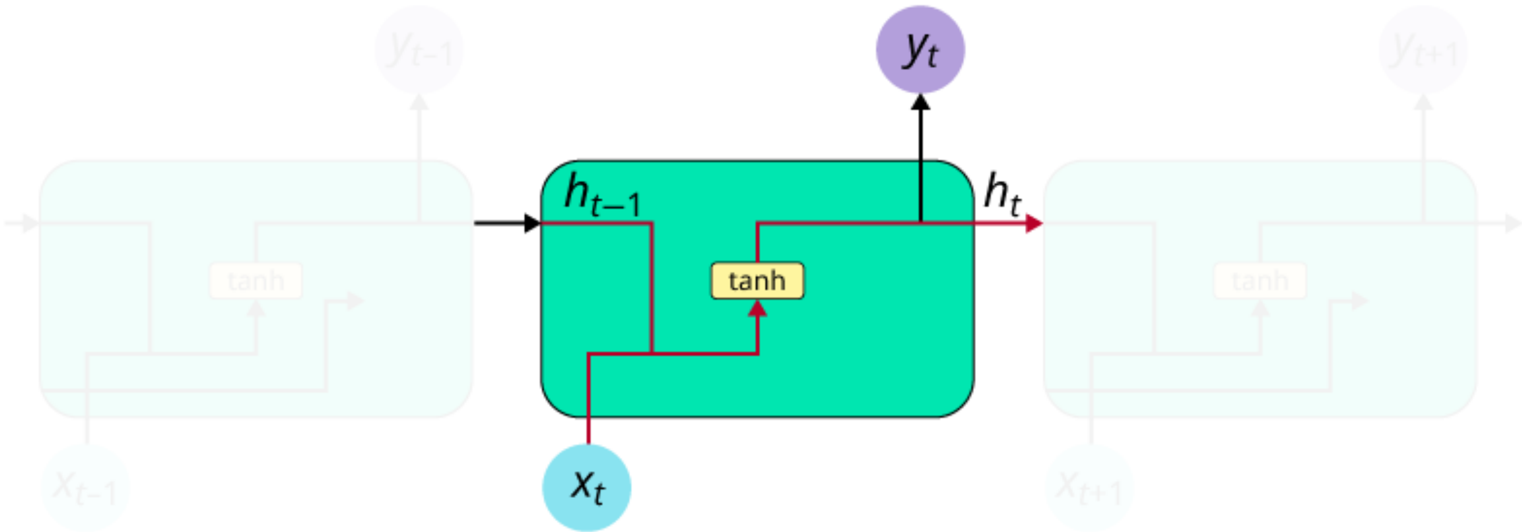
University of Bath

Topic 1: LSTM-1

Standard/Vanilla RNNs

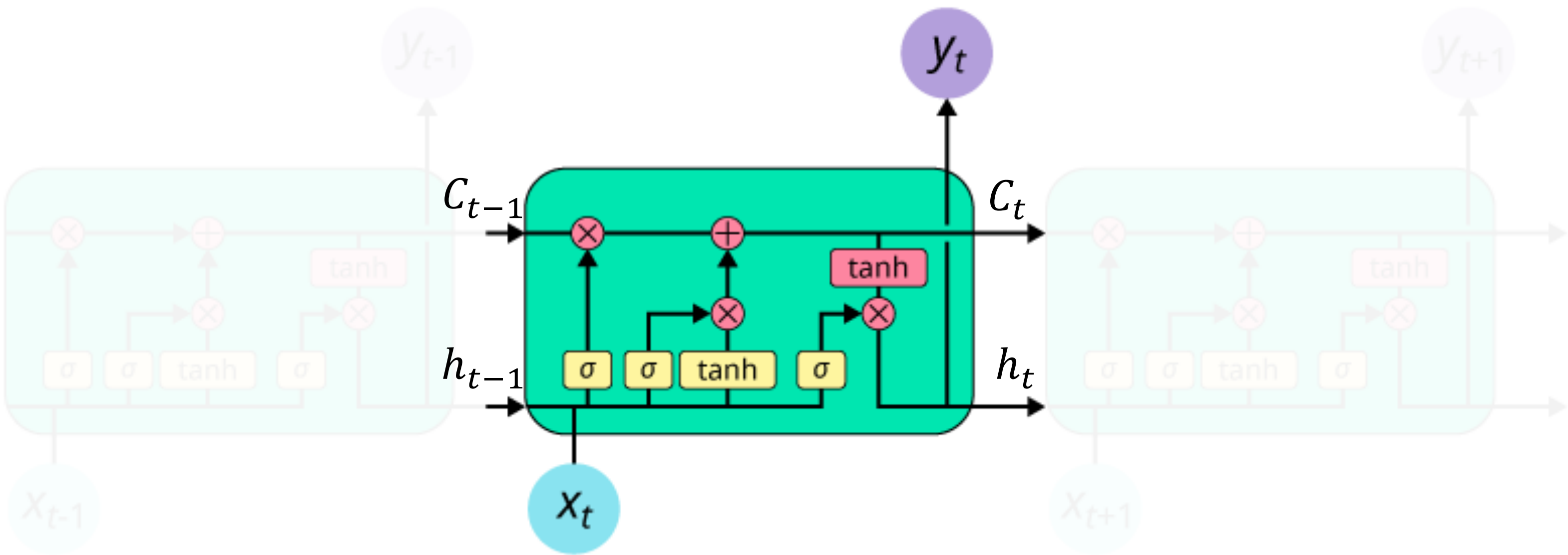
- In a standard RNN, repeating modules contain a **simple computation** node.

$$h_t = \tanh(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h)$$



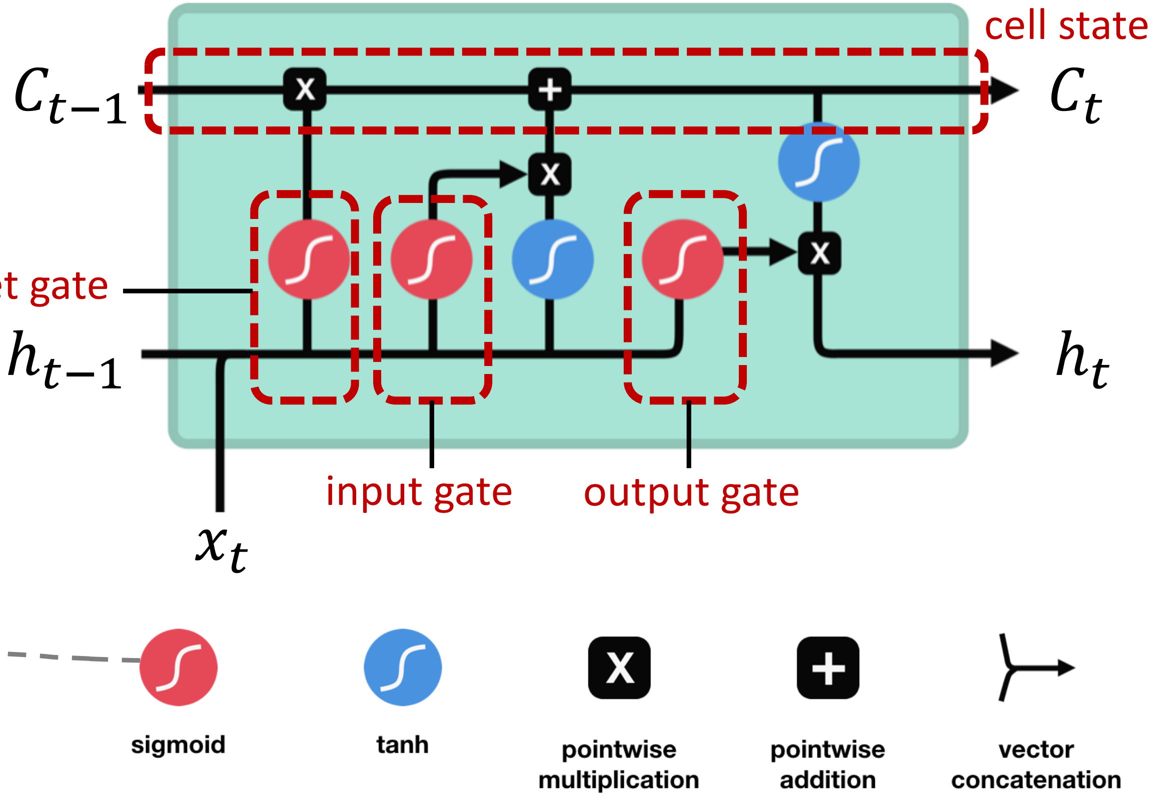
LSTM

- LSTMs are explicitly designed to deal with the long-term dependency problem.
- LSTM modules contain computational blocks that control information flow.



Two core concepts of LSTMs

- **Gates**: to control what information is to keep and forget.
- **Cell state**: act as a transport highway that transfers relative information all way down the sequence chain, thus store long-term information.



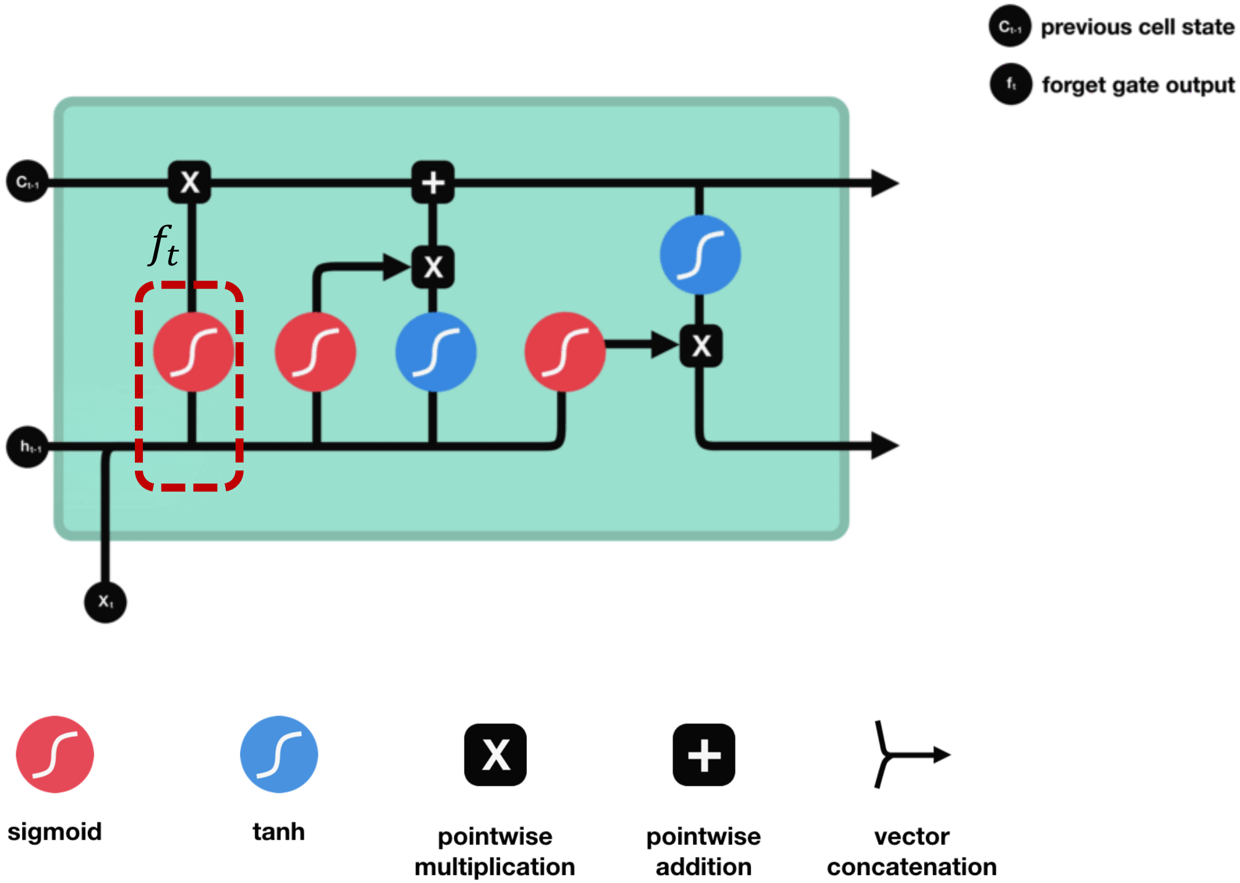
All gate values are between 0 (discard) and 1 (keep).

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

Forget gate

- **Forget gate**: forget irrelevant parts of the previous state.

$$f_t = \sigma(W_f \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_f)$$



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

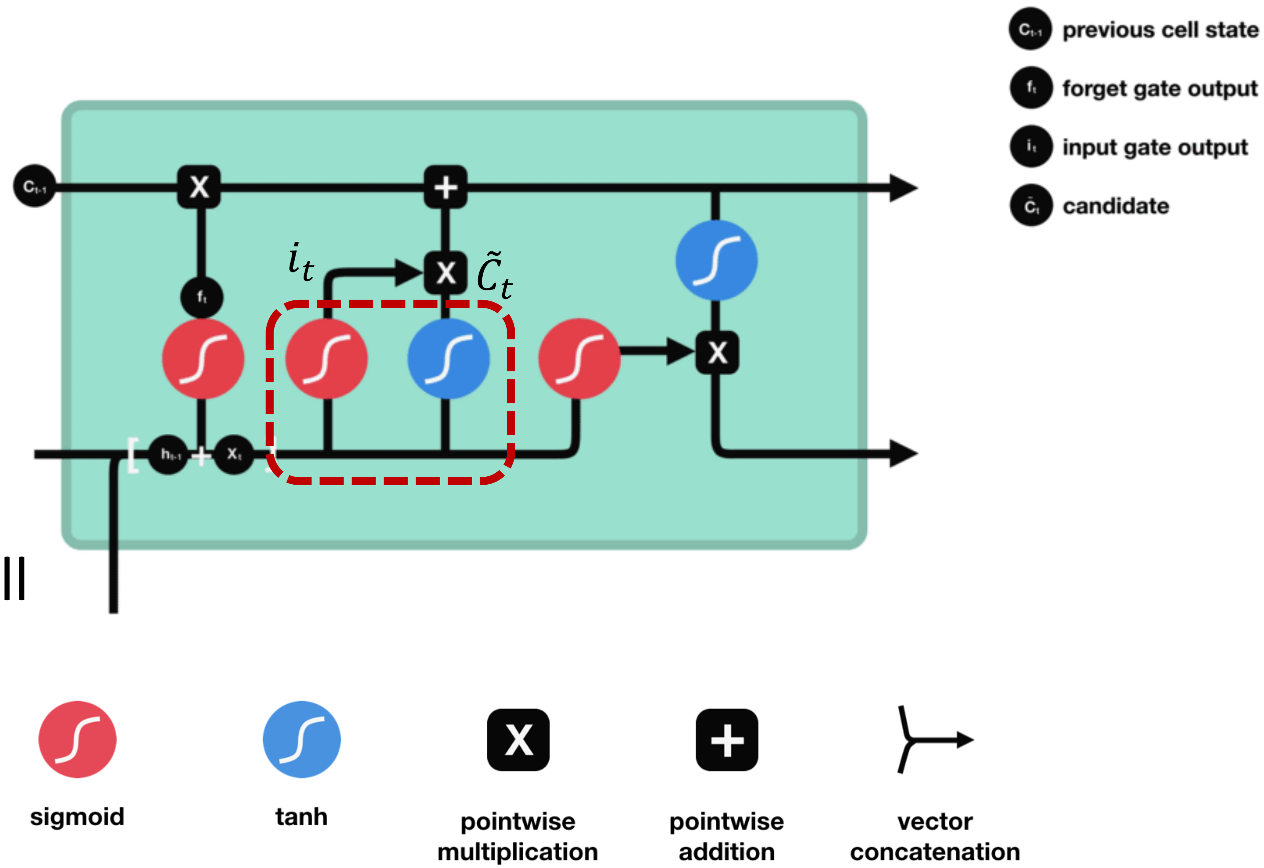
Input gate

- **Input gate**: decides what information is relevant to add from the current step.

$$i_t = \sigma(W_i \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_i)$$

- **New cell content**: the new content to be written to the cell

$$\tilde{C}_t = \tanh(W_C \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_C)$$



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

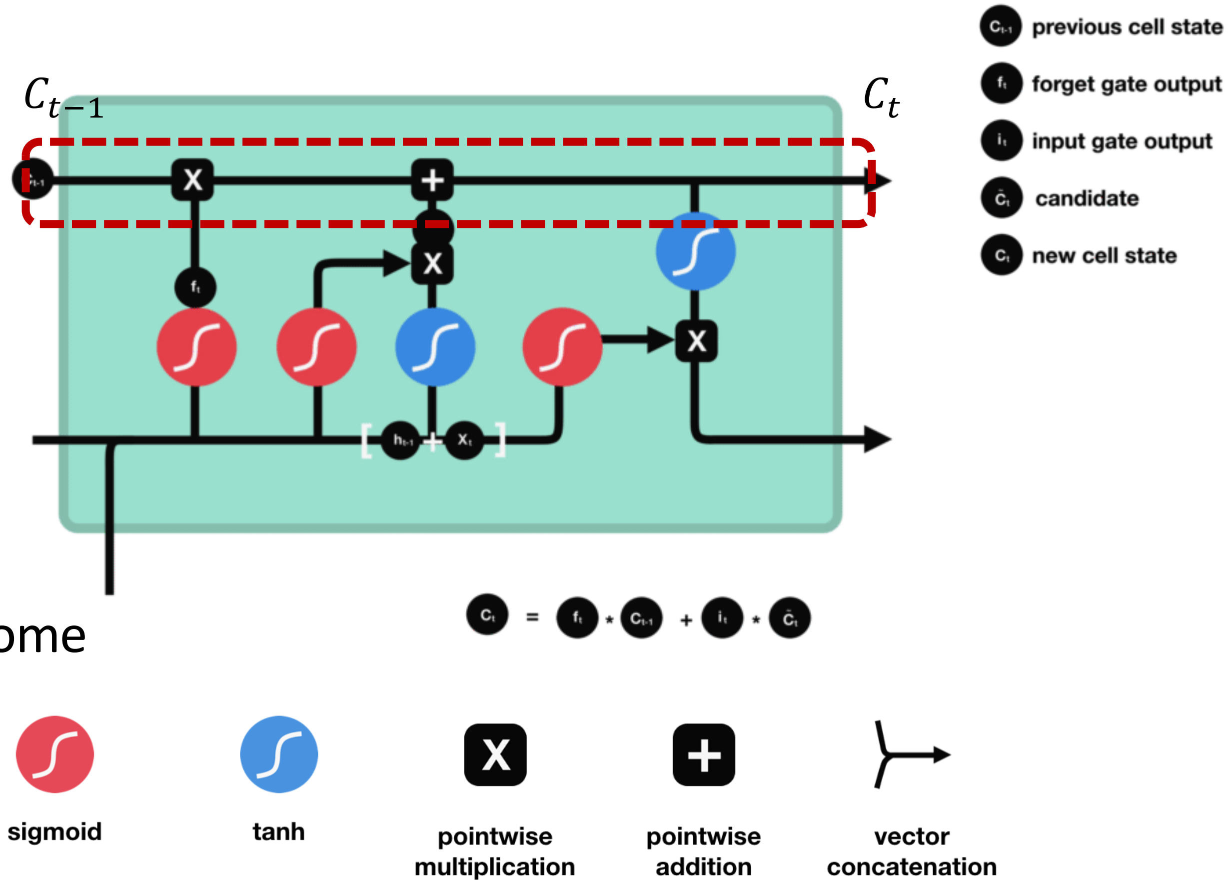
Cell state

- **Cell state**: update the cell state to new values that the network finds relevant.

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

Keep ("input") some new cell content

Erase ("forget") some content from the previous state



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

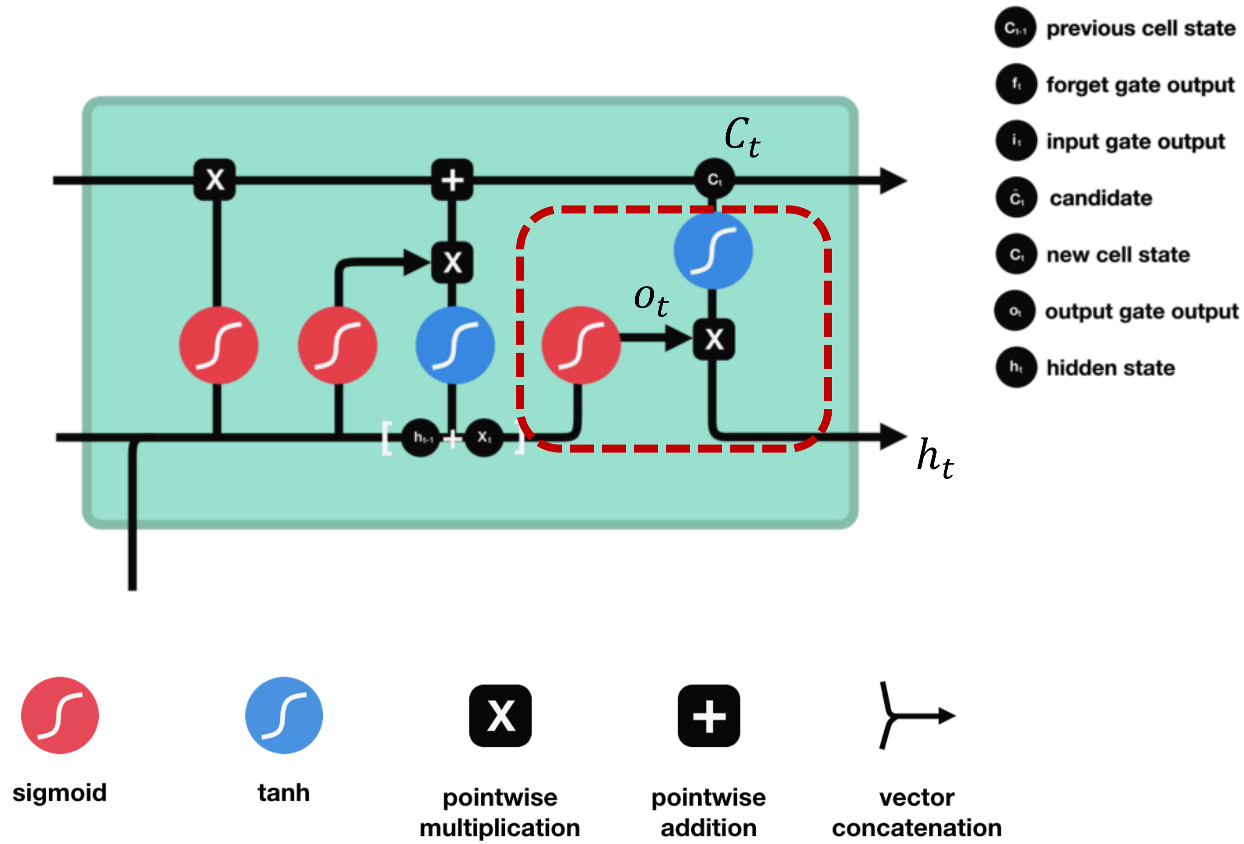
Output gate

- **Output gate**: determines what parts of the cell are output to the hidden state.

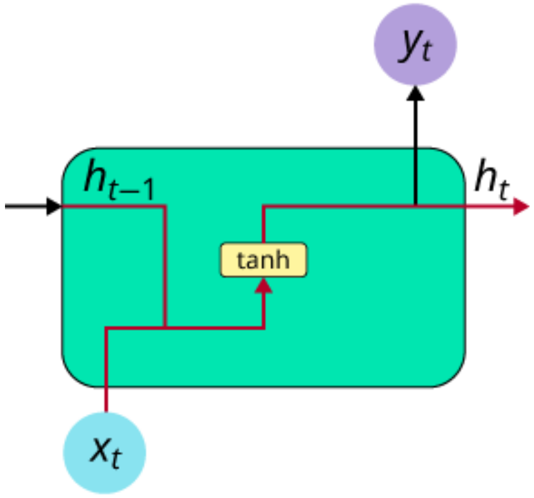
$$o_t = \sigma(W_o \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_o)$$

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

$$h_t = o_t \circ \tanh(C_t)$$

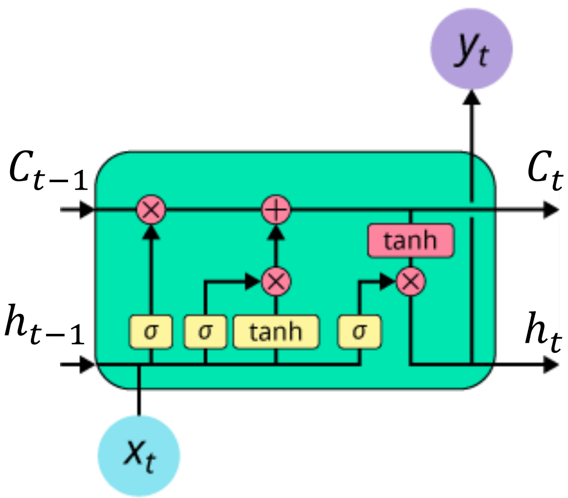


Animation from: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21



Vanilla RNN:

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h\right)$$



LSTM:

$f_t \in \mathbb{R}^H$
$i_t \in \mathbb{R}^H$
$o_t \in \mathbb{R}^H$
$\tilde{C}_t \in \mathbb{R}^H$
$C_t \in \mathbb{R}^H$
$h_t \in \mathbb{R}^H$
$x_t \in \mathbb{R}^M$
$W_f, W_i, W_o, W_C \in \mathbb{R}^{H\times(H+M)}$

$$f_t = \sigma\left(W_f \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_f\right)$$

$$i_t = \sigma\left(W_i \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_i\right)$$

$$o_t = \sigma\left(W_o \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_o\right)$$

$$\tilde{C}_t = \tanh\left(W_C \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_C\right)$$

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

$$h_t = o_t \circ \tanh(C_t)$$

Reference for Topic 1

- Blog by Colah: Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Blog by Michael Phi: Illustrated Guide to LSTM's and GRU's: A step to step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
- Video lecture by Alexander Amini: MIT course on Recurrent Neural Networks, YouTube.
- Lectures from University of Michigan: https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html

Topic 2: LSTM-2

LSTM Architecture

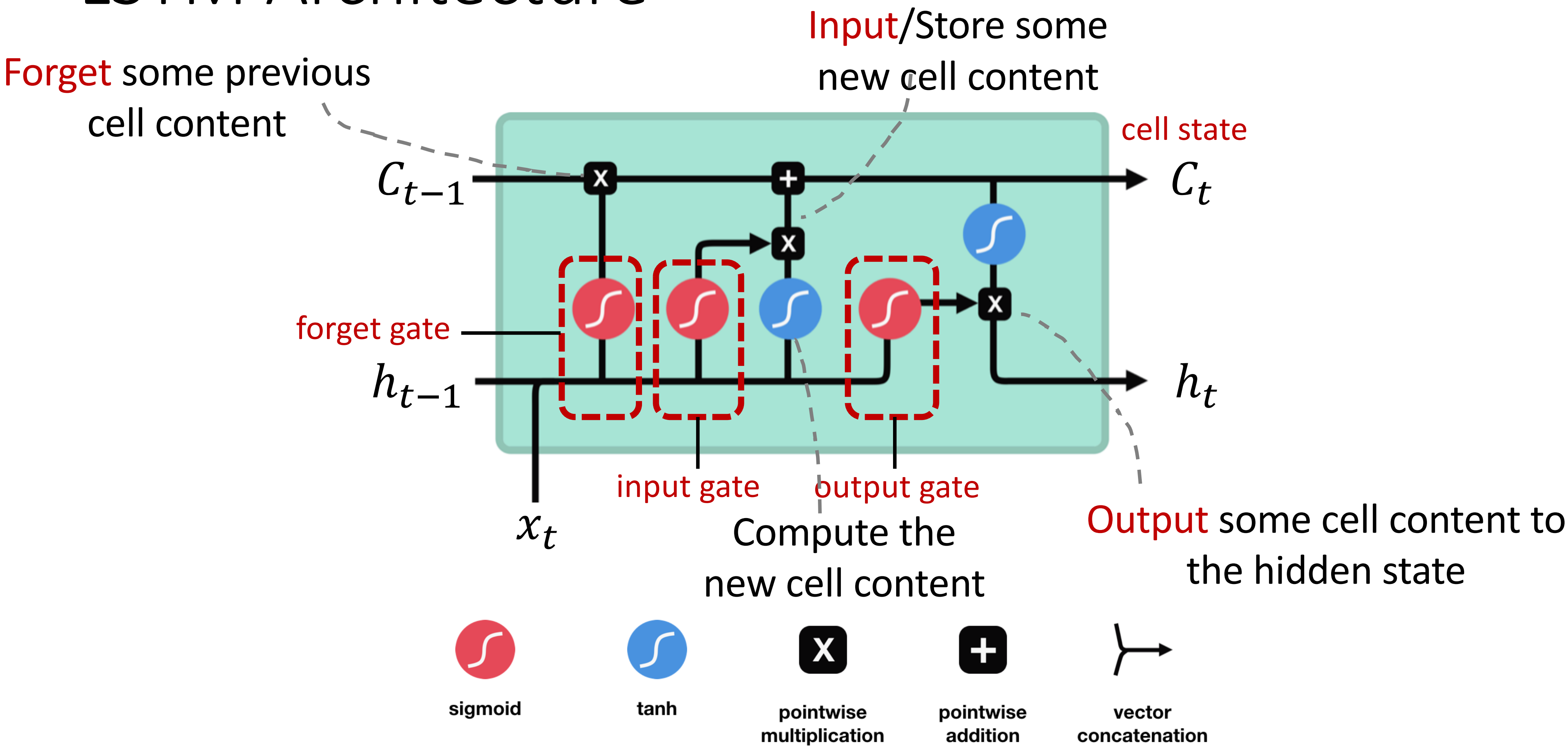
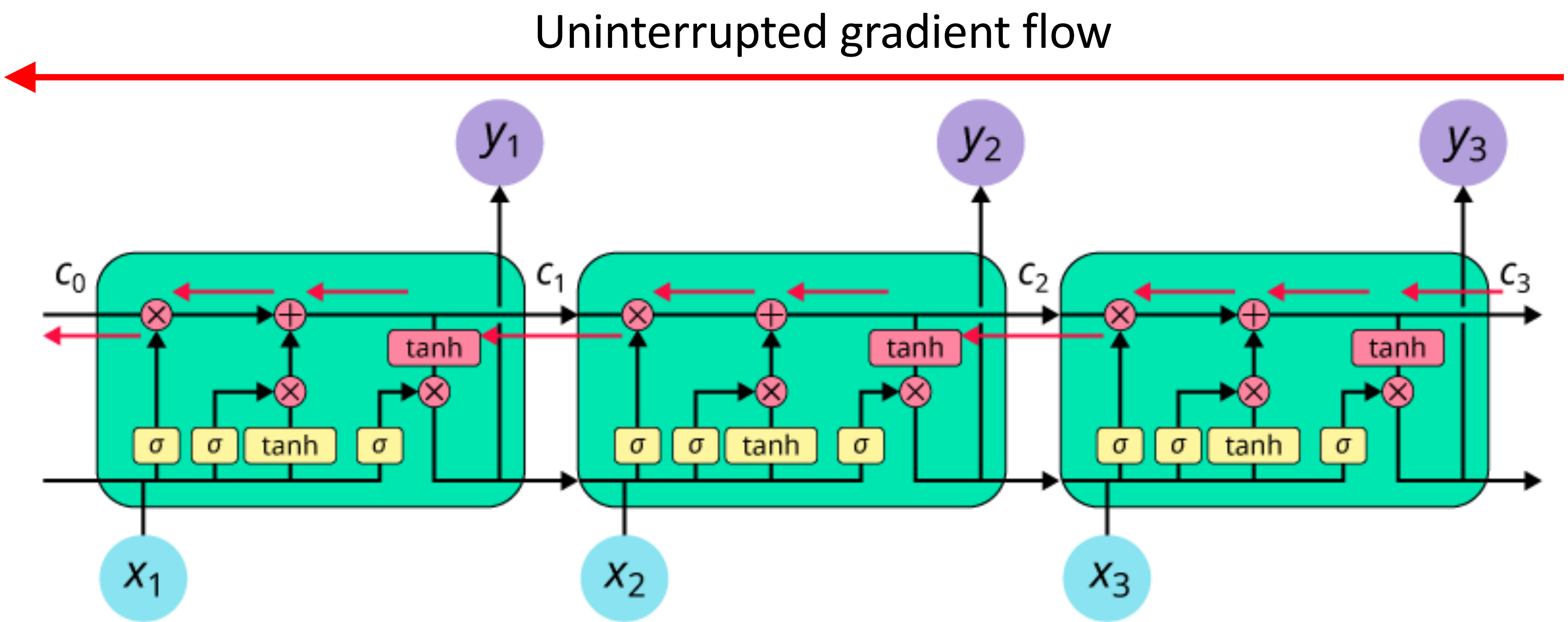


Image from: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

LSTM Gradient Flow

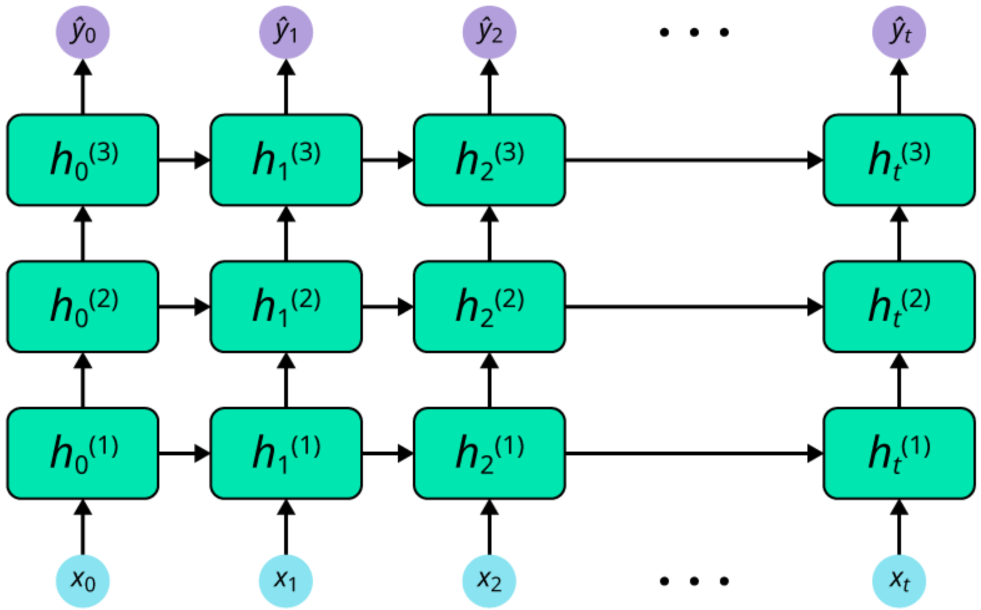


LSTMs **solve the vanishing/exploding gradient problem** using an additive gradient structure.

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

Advanced use of RNNs/LSTMs

- **Multi-layer RNNs (Deep RNNs)**: stack more than one RNN. It increases the representation power of the network, at the cost of higher computational loads.

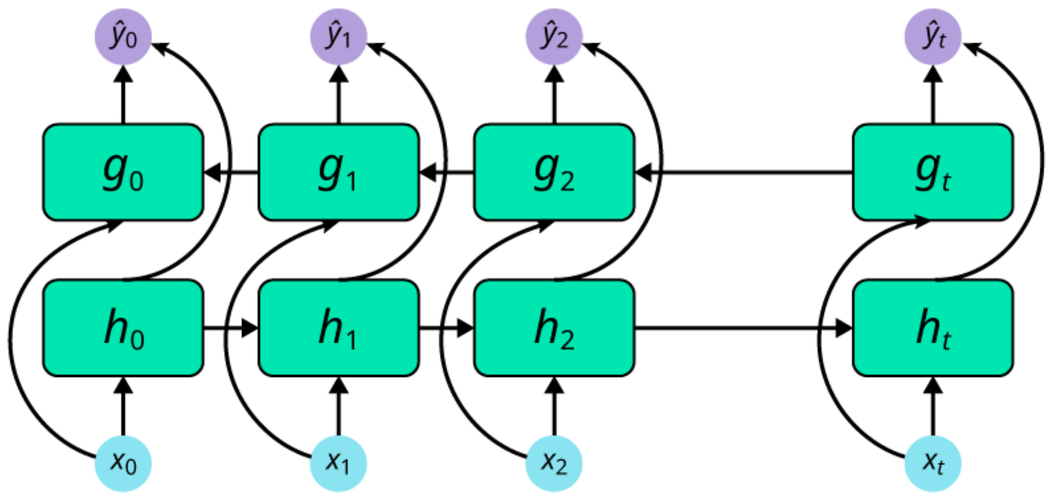



```
from keras.layers import LSTM
...
model.add(LSTM(32), return_sequences=True)
model.add(LSTM(32), return_sequences=True)
model.add(LSTM(32))
...
```

`True` means: output all the hidden states

Advanced use of RNNs/LSTMs

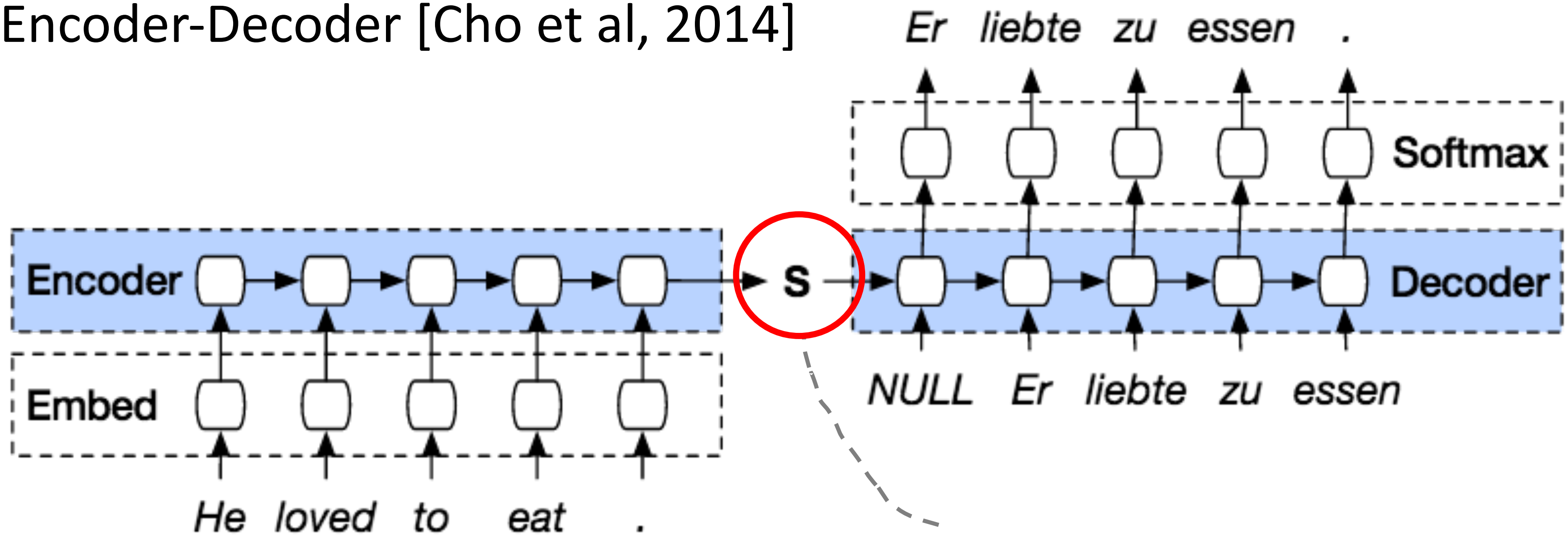
- **Bidirectional RNNs**: process a sequence in both directions, capturing pattens that may be missed by the chronological-order version alone.




```
from keras.layers import Bidirectional, LSTM
...
model.add(Bidirectional(LSTM(32)))
...
```

Example Tasks: Machine Translation

- Encoder-Decoder [Cho et al, 2014]



Problem: **Encoding bottleneck**
One solution**: Attention Based Encoder-Decoder**

Image from: https://smerity.com/articles/2016/google_nmt_arch.html

Example Tasks: Image Caption Generation

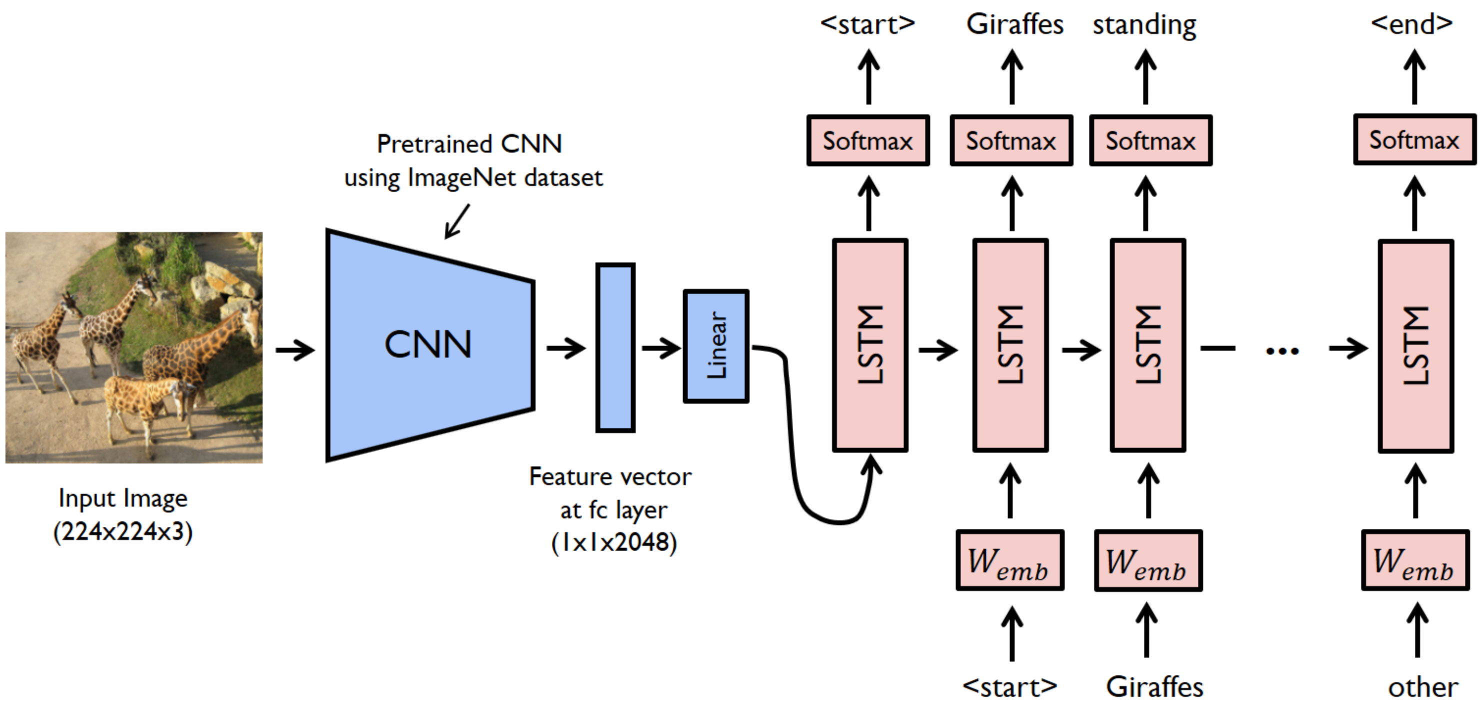


Image from: https://www.analyticsvidhya.com/blog/2018/04/solving-an-image-captioning-task-using-deep-learning/

Summary of LSTMs

- Beside the hidden state, also maintain a **cell state** to store **long-term information.**
- Use **gates** to control the flow of information
  - **Forget** gate: gets rid of irrelevant **old** information
  - **Input** gate**:** stores relevant information from **current** input
  - **Output** gate: **output** a filtered version of the cell state
- Backpropagation through time with **uninterrupted gradient flow**, to avoid the vanishing/exploding gradient problem.

Reference for Topic 2

- Blog by Colah: Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Blog by Michael Phi: Illustrated Guide to LSTM's and GRU's: A step to step explanation. https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21
- Video lecture by Alexander Amini: MIT course on Recurrent Neural Networks, YouTube.
- Lectures from University of Michigan: https://web.eecs.umich.edu/~justincj/teaching/eecs498/FA2020/schedule.html
- Blogs by Nir Arbel: How do LSTM networks solve the problem of vanishing gradients: https://medium.com/datadriveninvestor/how-do-lstm-networks-solve-the-problem-of-vanishing-gradients-a6784971a577