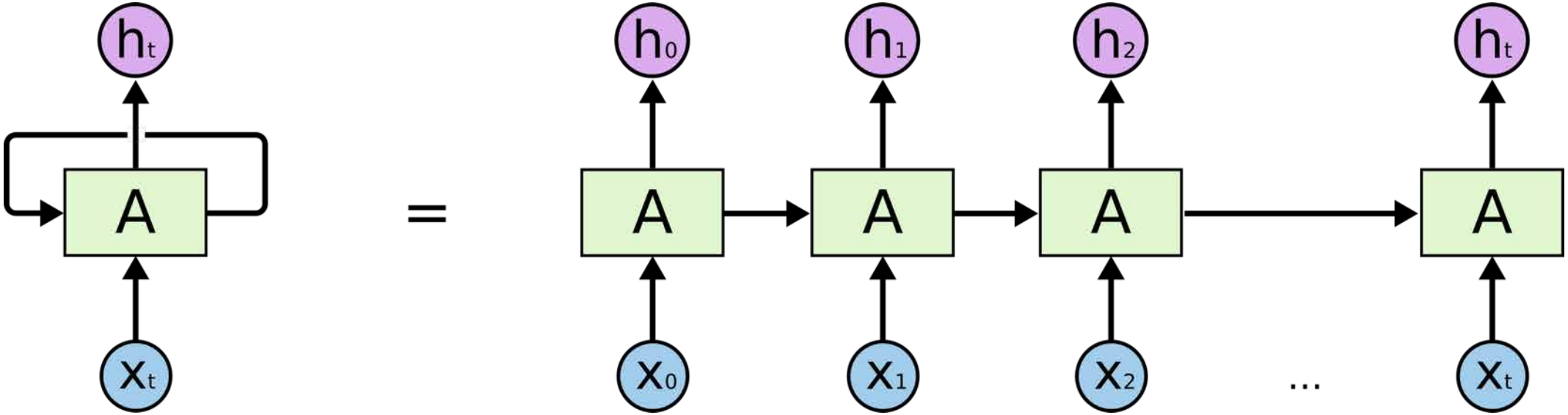
CS 466/566
Introduction to Deep Learning

Lecture 12 – Recurrent Neural Networks – Part 2

Compiled from various resources

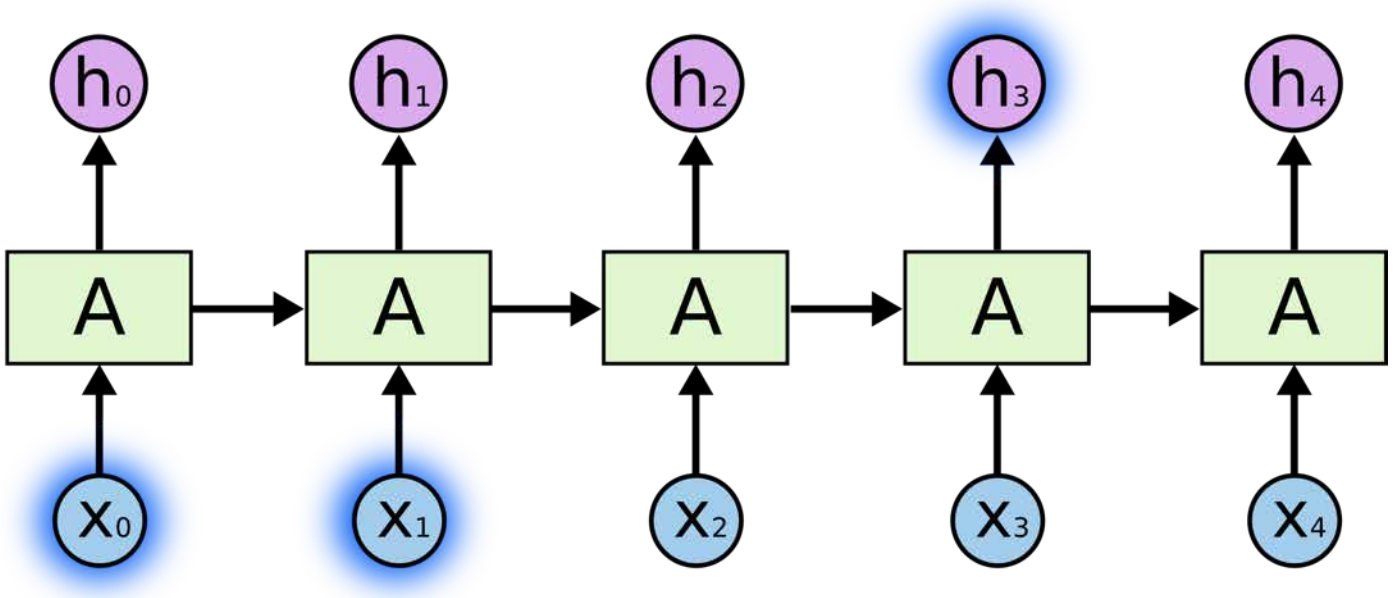
Recurrent neural networks for modeling sequences

- Recurrent neural networks are a very natural way to model sequential data:
  - They are equivalent to very deep nets with one hidden layer per time slice.
  - Except that they use the same weights at every time slice and they get input at every time slice.
- They have the ability to remember information in their hidden states.
  - But its very hard to train them to use this potential.



The Problem of Long-Term Dependencies

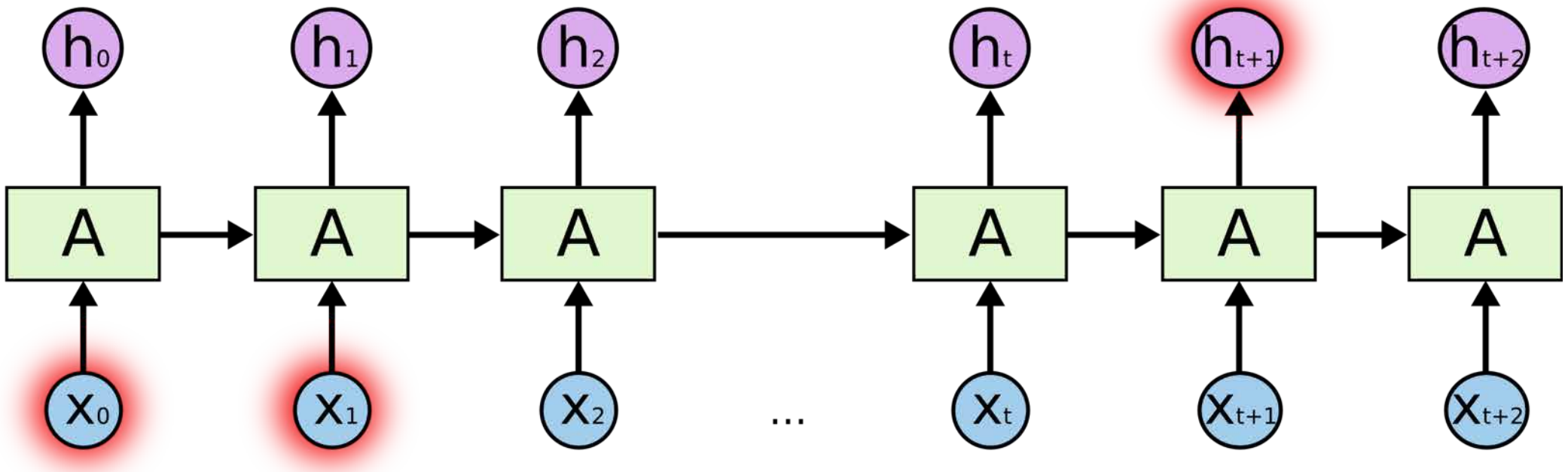
- Sometimes, we only need to look at recent information to perform the present task.
- Predict last word in the sentence:
  - "the clouds are in the *sky*" (relevant information is very close)



Borrowed from Christopher Olah

The Problem of Long-Term Dependencies

- There are also cases where we need more context.
- Predict last word in the sentence:
  - "I grew up in France… I speak fluent *French*." (relevant information is far away)



Borrowed from Christopher Olah

Long Short-Term Memory Networks (LSTM)

Long Short-Term Memory, 1997

Sepp Hochreiter, J. Schmidhuber

http://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735

Learning to Forget: Continual Prediction with LSTM, 2000

*F. A. Gers, **J. Schmidhuber**, F. Cummins*

http://www.mitpressjournals.org/doi/abs/10.1162/089976600300015015

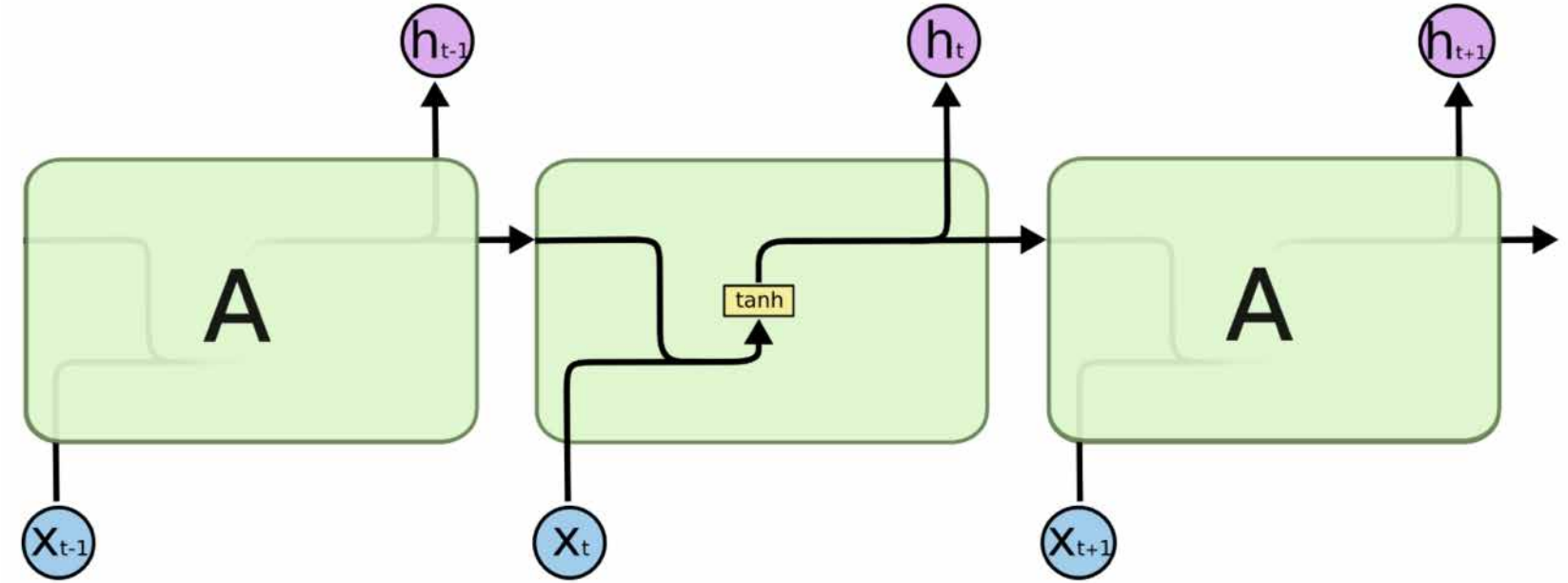
LSTM recurrent networks learn simple context-free and context-sensitive languages, 2001

*F. A. Gers, **J. Schmidhuber***

http://ieeexplore.ieee.org/abstract/document/963769

Vanilla RNNs

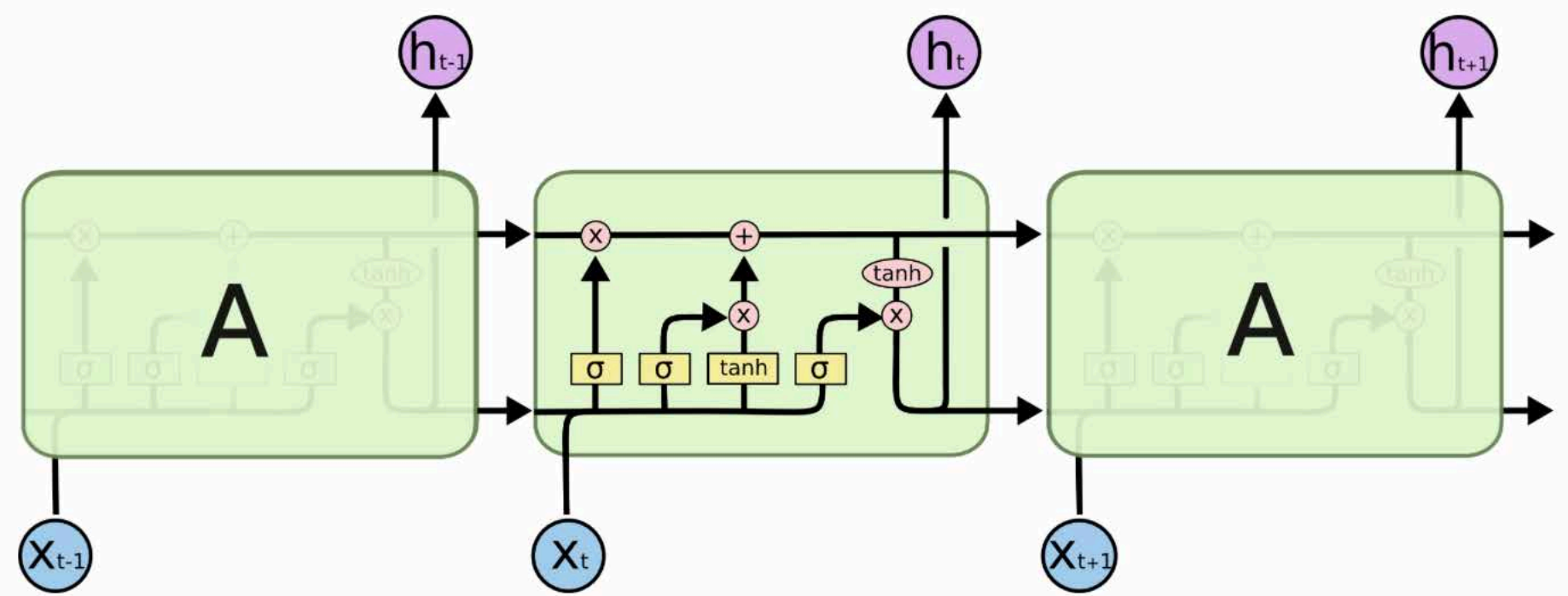
- All recurrent neural networks have the form of a chain of repeating modules of neural network.
- In standard RNNs, this repeating module will have a very simple structure, such as a single ***tanh*** layer.



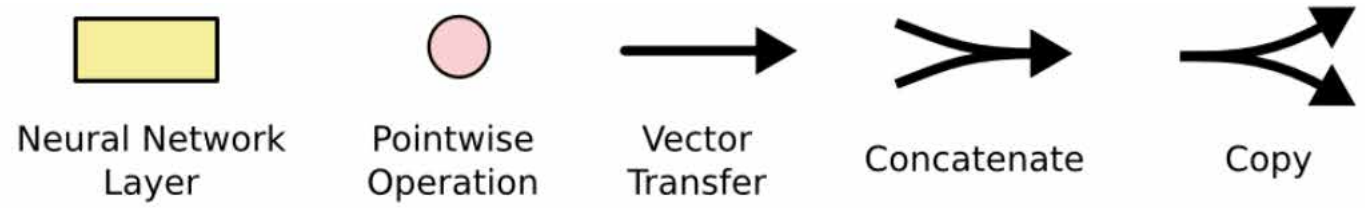
The repeating module in a standard RNN contains a single layer.

Borrowed from Christopher Olah

Long Short-Term Memory Networks (LSTM)



The repeating module in an LSTM contains four interacting layers.



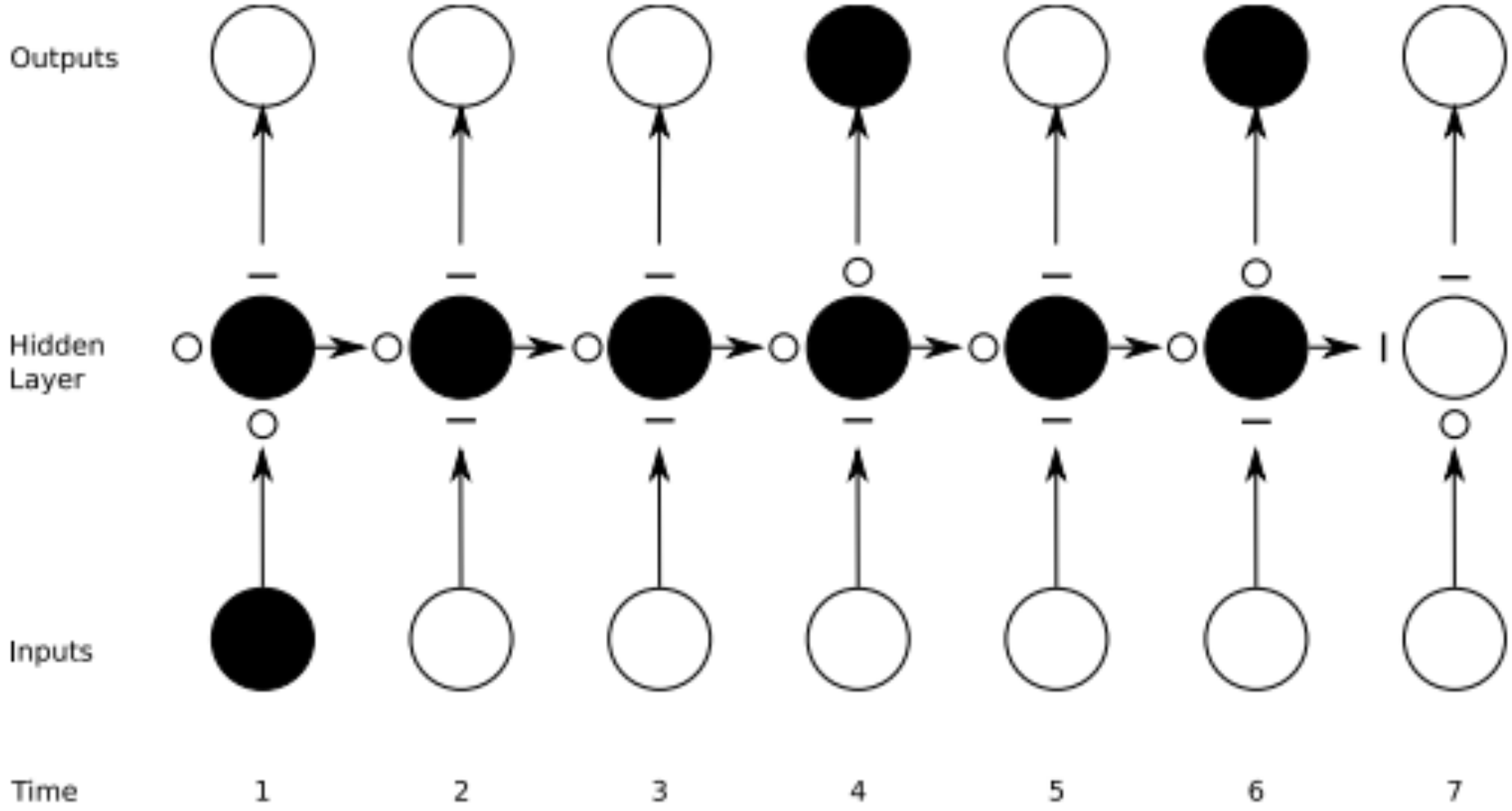
Borrowed from Christopher Olah

Long Short-Term Memory Networks (LSTM)

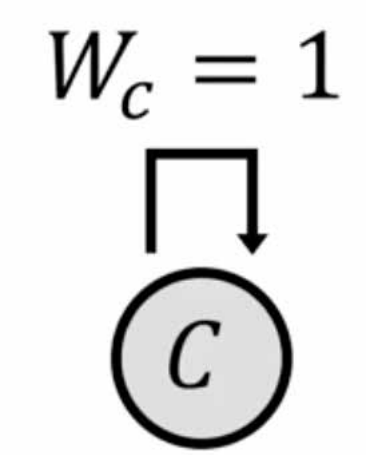
- Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies.
- They work tremendously well on a large variety of problems, and are now widely used.
- LSTMs are explicitly designed to avoid the long-term dependency problem.
- Remembering information for long periods of time is practically their default behavior.

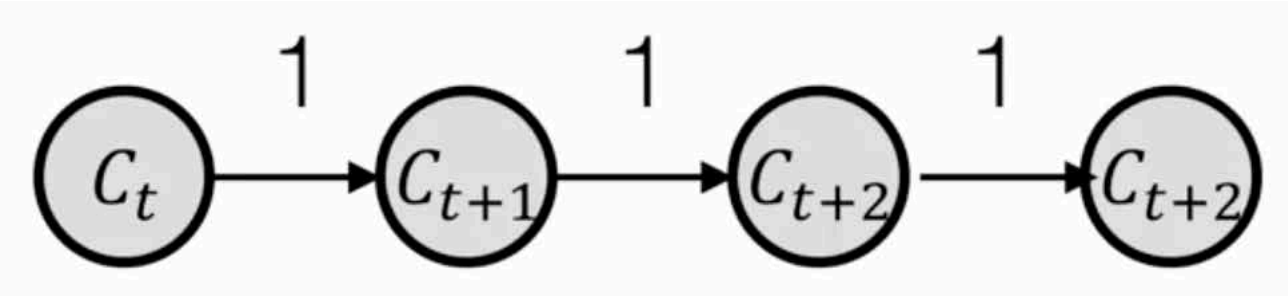
Borrowed from Christopher Olah

Single LSTM Memory Cell Dimension and Its Time Travel



LSTM Internal State = Memory = Conveyor Belt



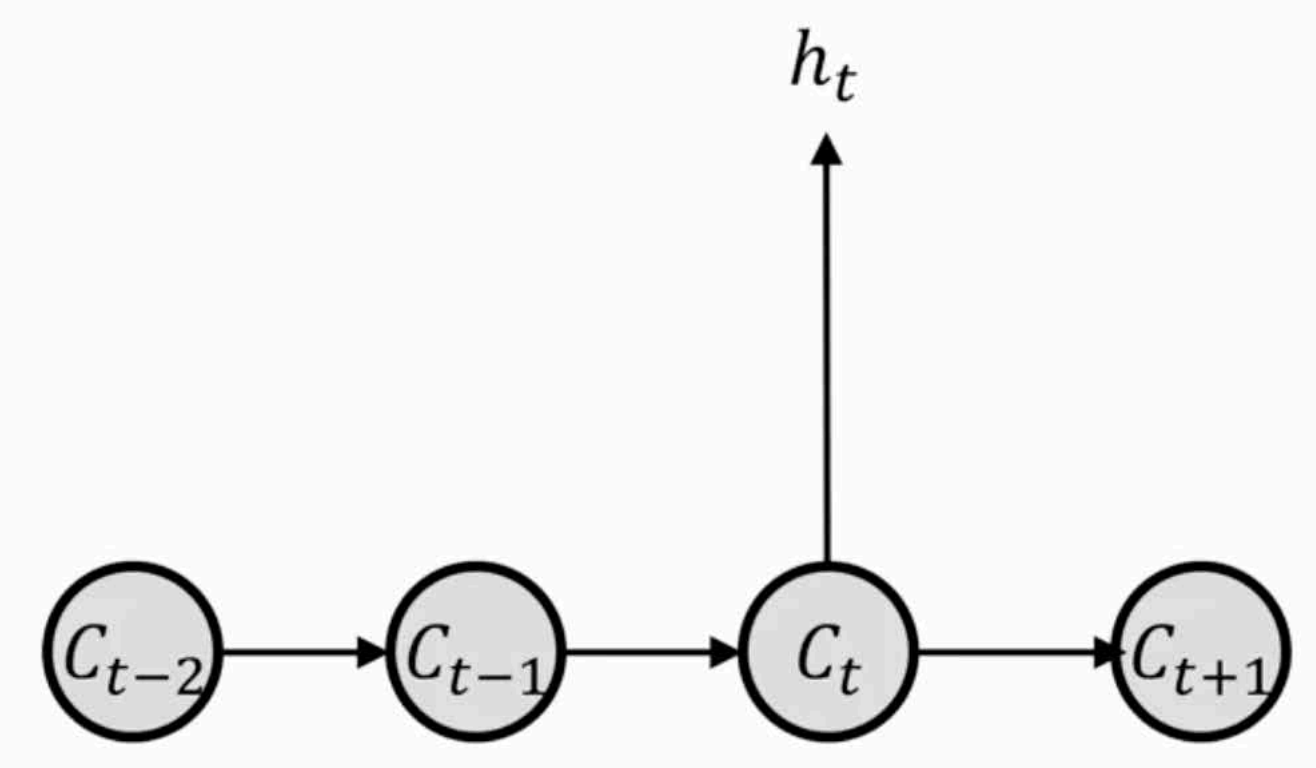


Want to Manipulate Memory Cell

1. Forget (flush the memory)
2. Input (add to memory)
3. Output (get from memory)

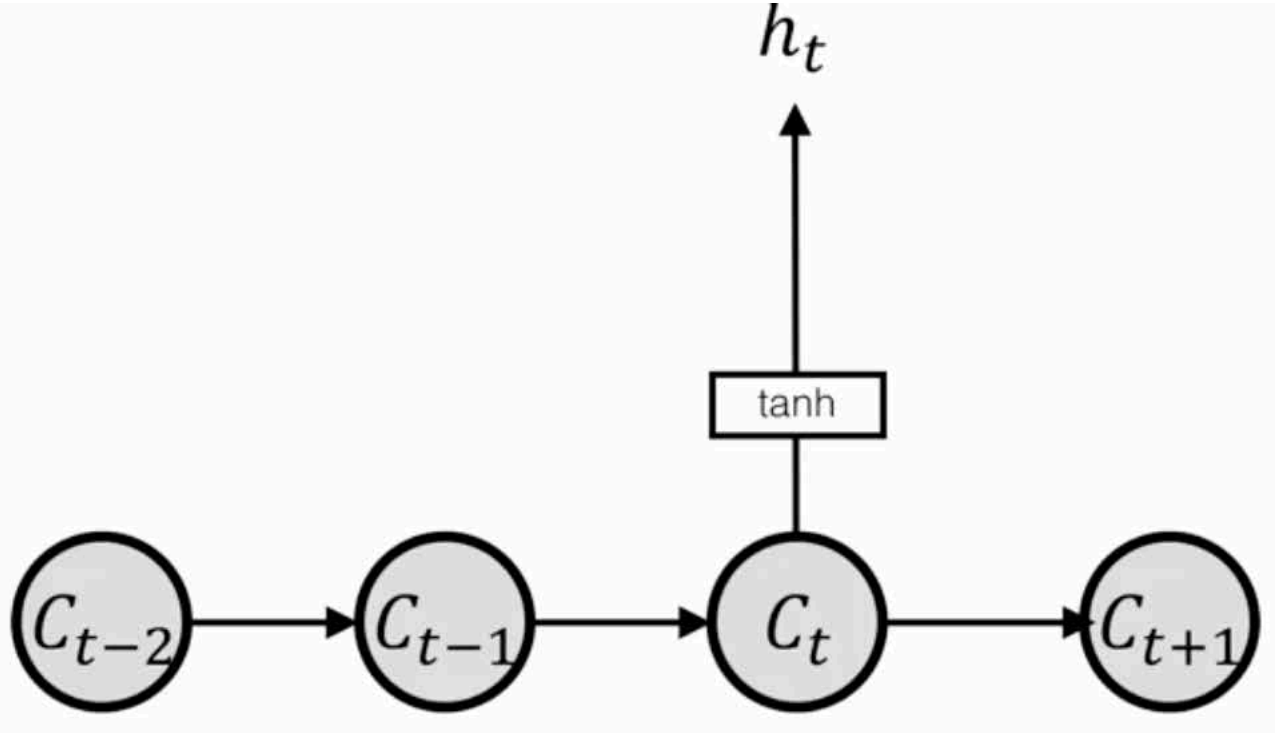
Borrowed from Nervana

LSTM:



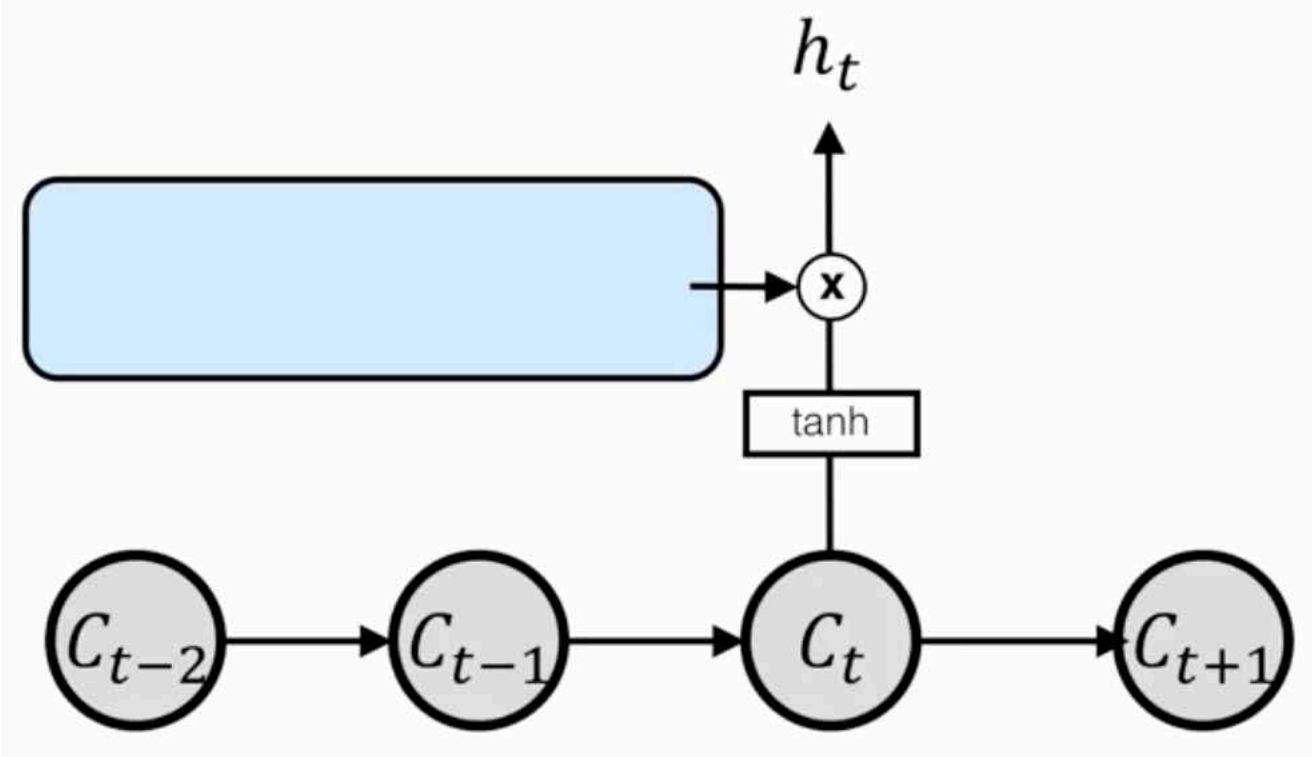
Borrowed from Nervana

LSTM: Filtered Output Gate



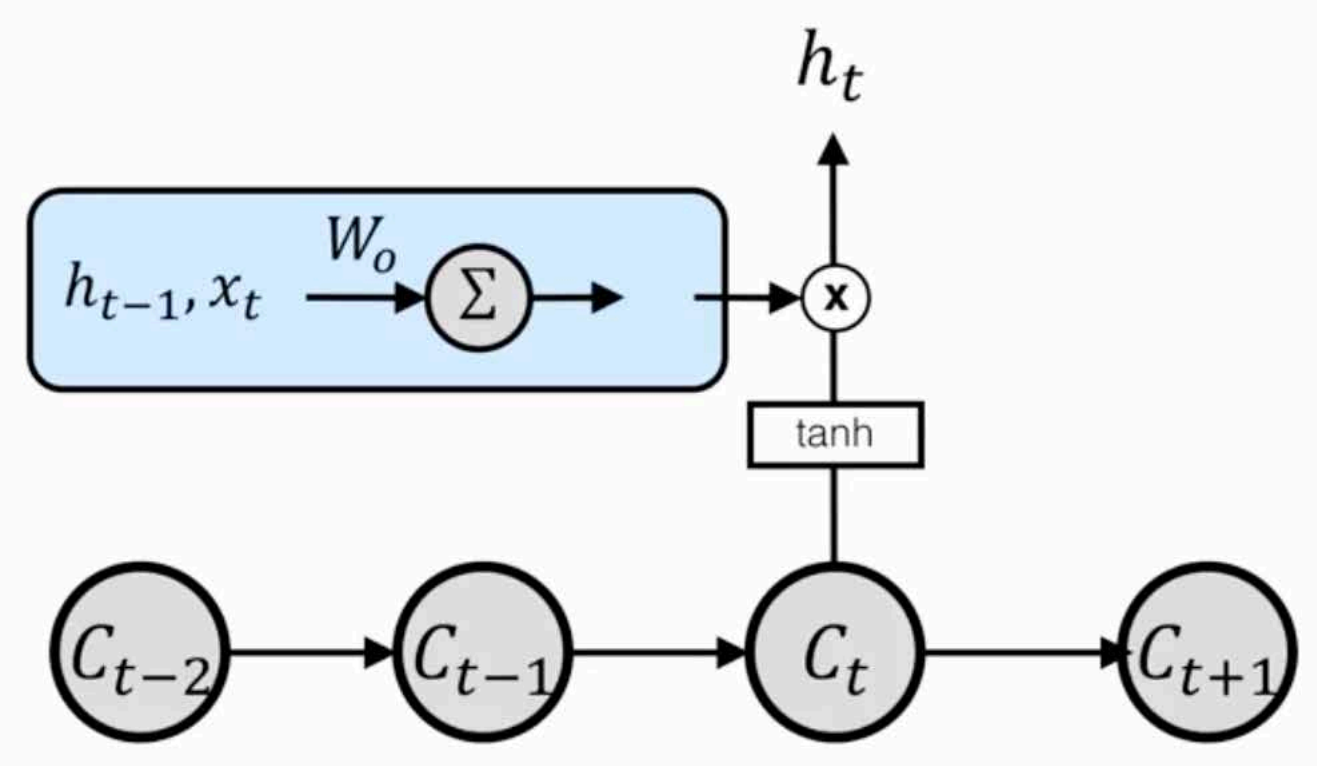
Borrowed from Nervana

LSTM: Filtered Output Gate



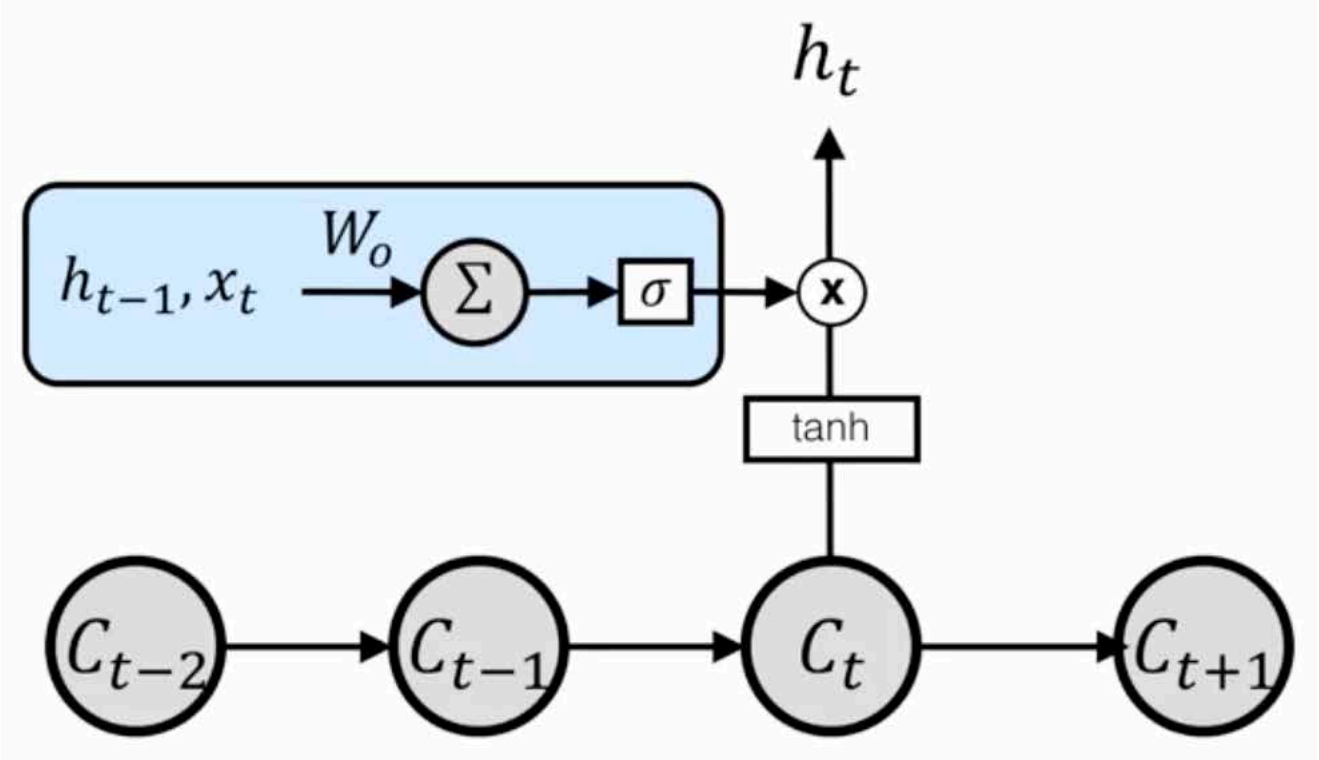
Borrowed from Nervana

LSTM: Filtered Output Gate



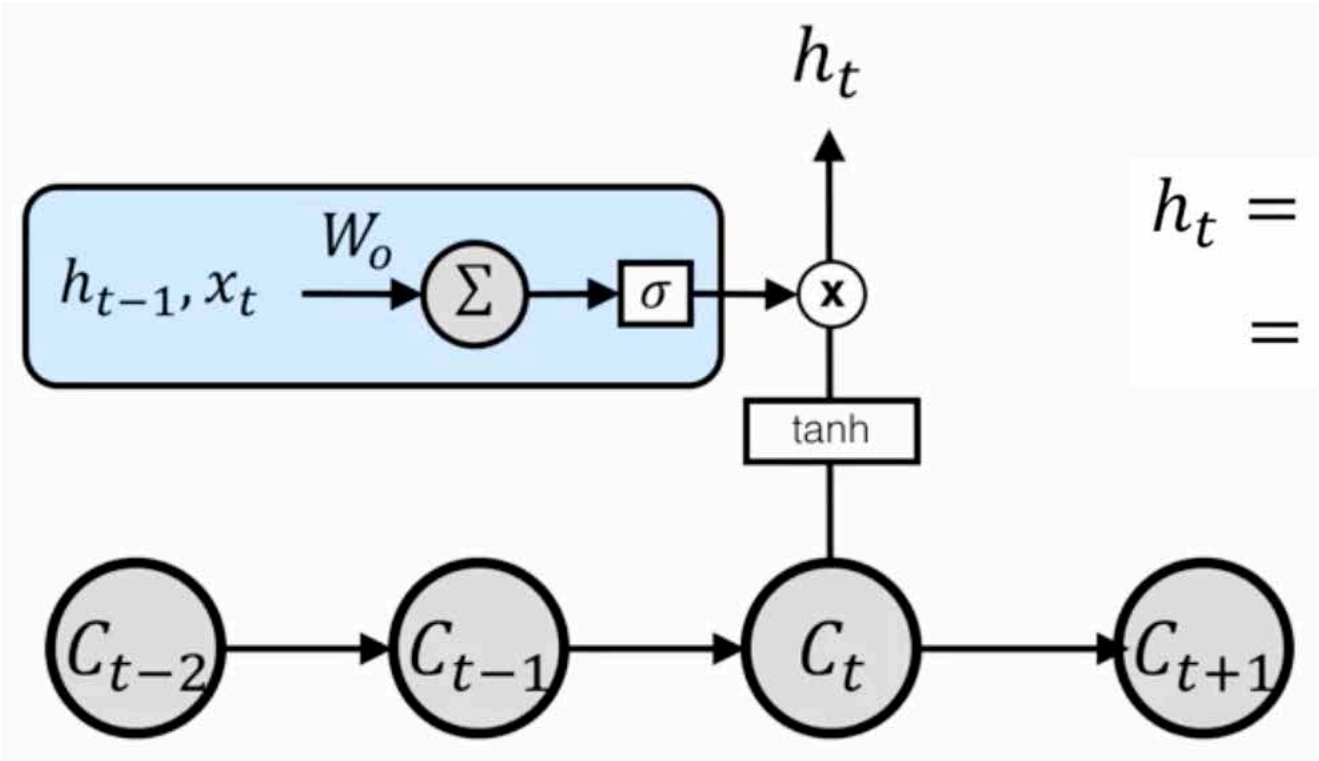
Borrowed from Nervana

LSTM: Filtered Output Gate

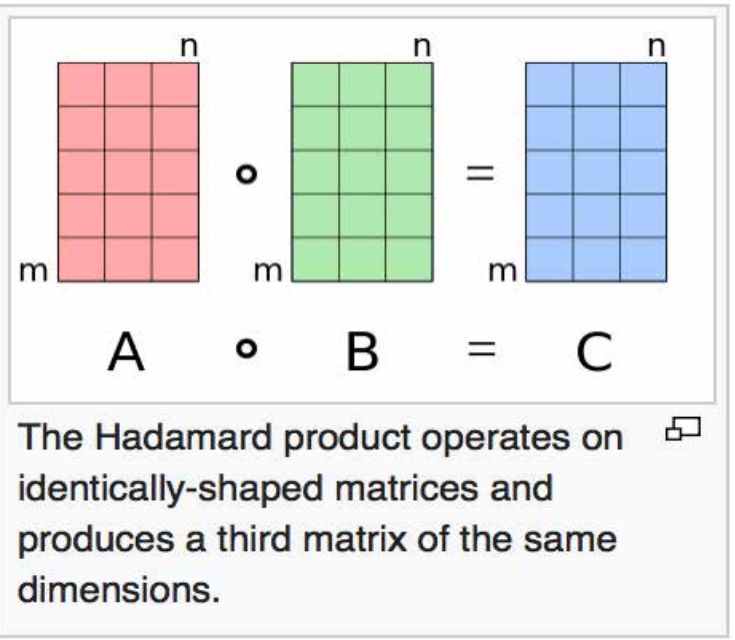


Borrowed from Nervana

LSTM: Filtered Output Gate



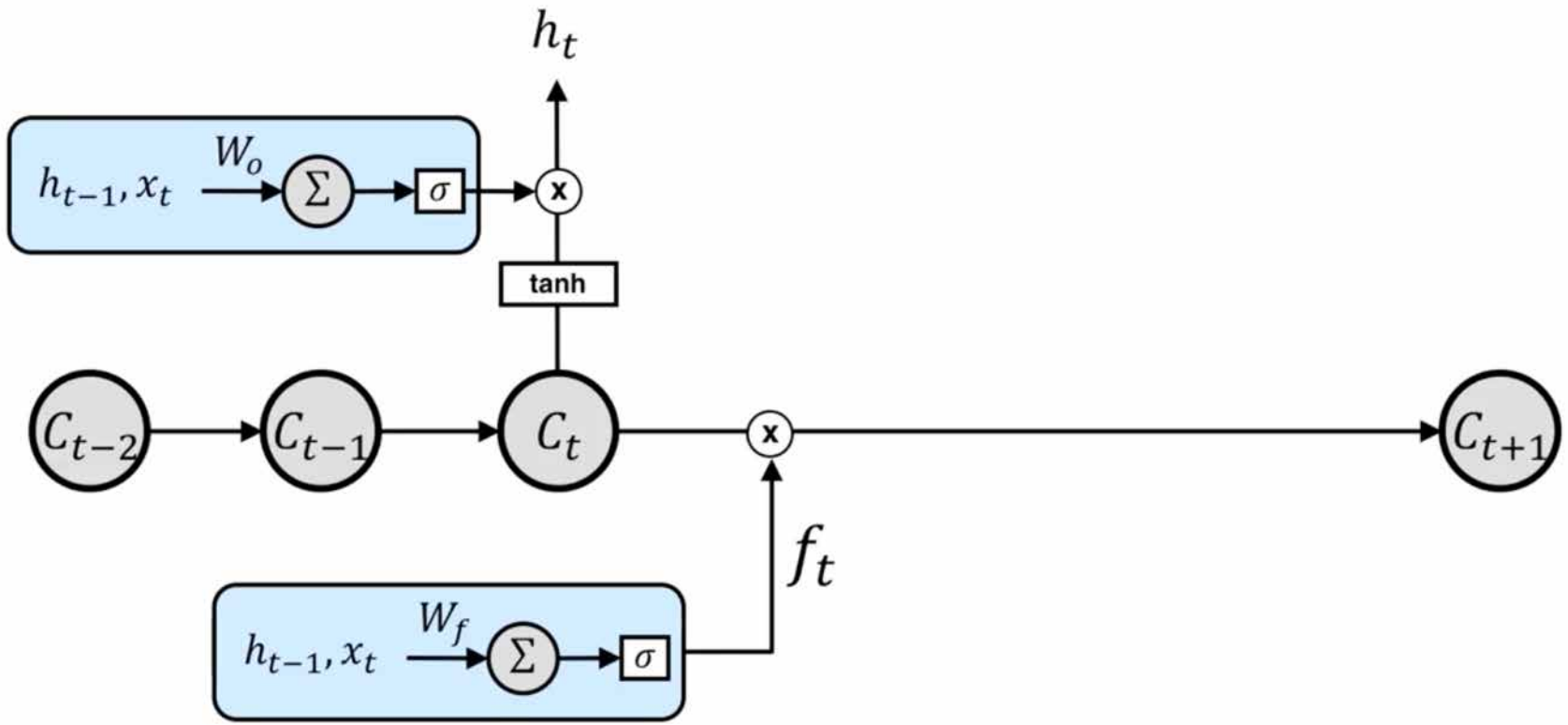
$$h_t = \sigma(W_o \cdot [x_t, h_{t-1}] + b_o) \odot \tanh(C_t)$$
$$= o_t \odot \tanh(C_t)$$



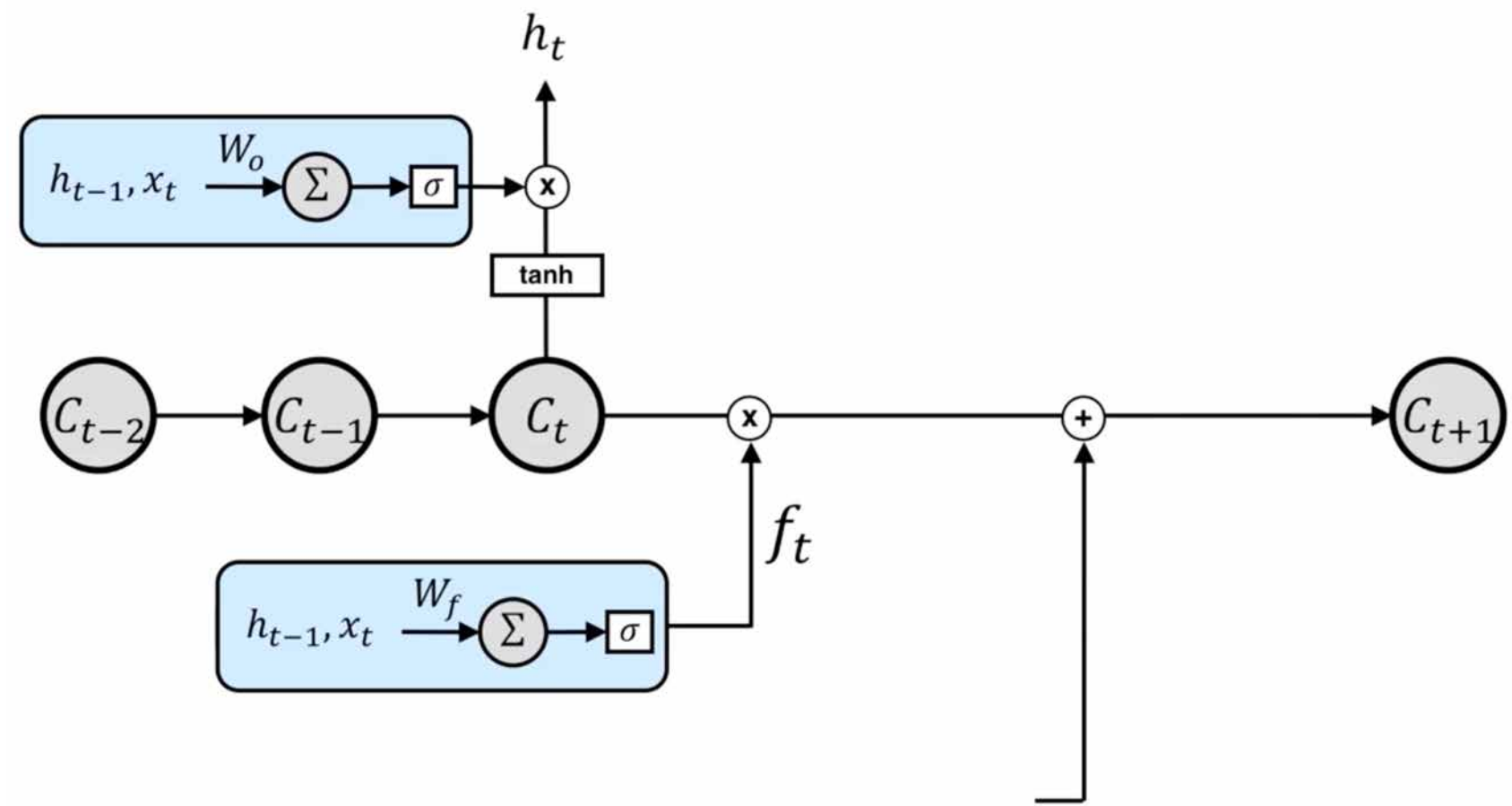
The Hadamard product operates on identically-shaped matrices and produces a third matrix of the same dimensions.

Borrowed from Nervana

LSTM: Forget Gate

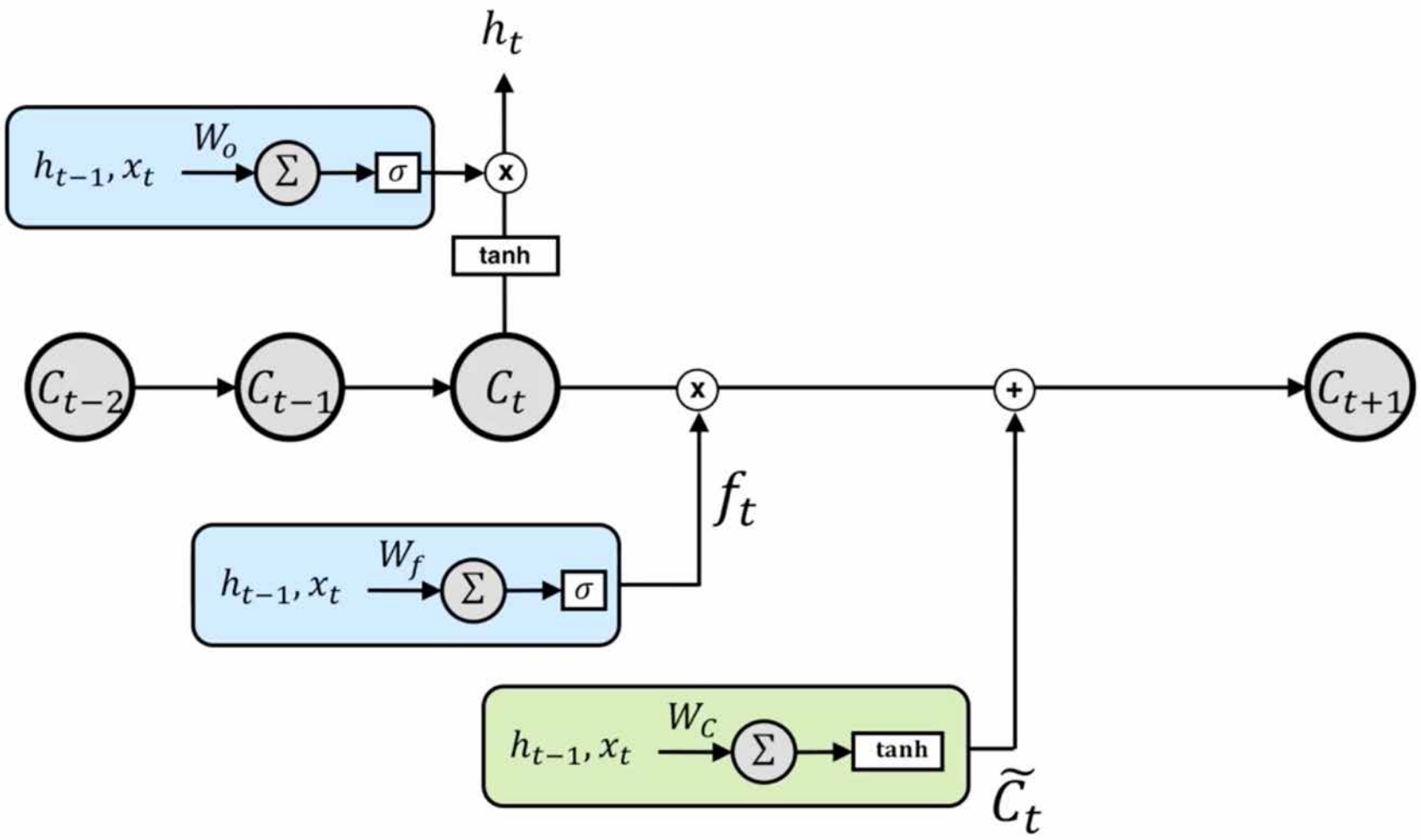


Borrowed from Nervana



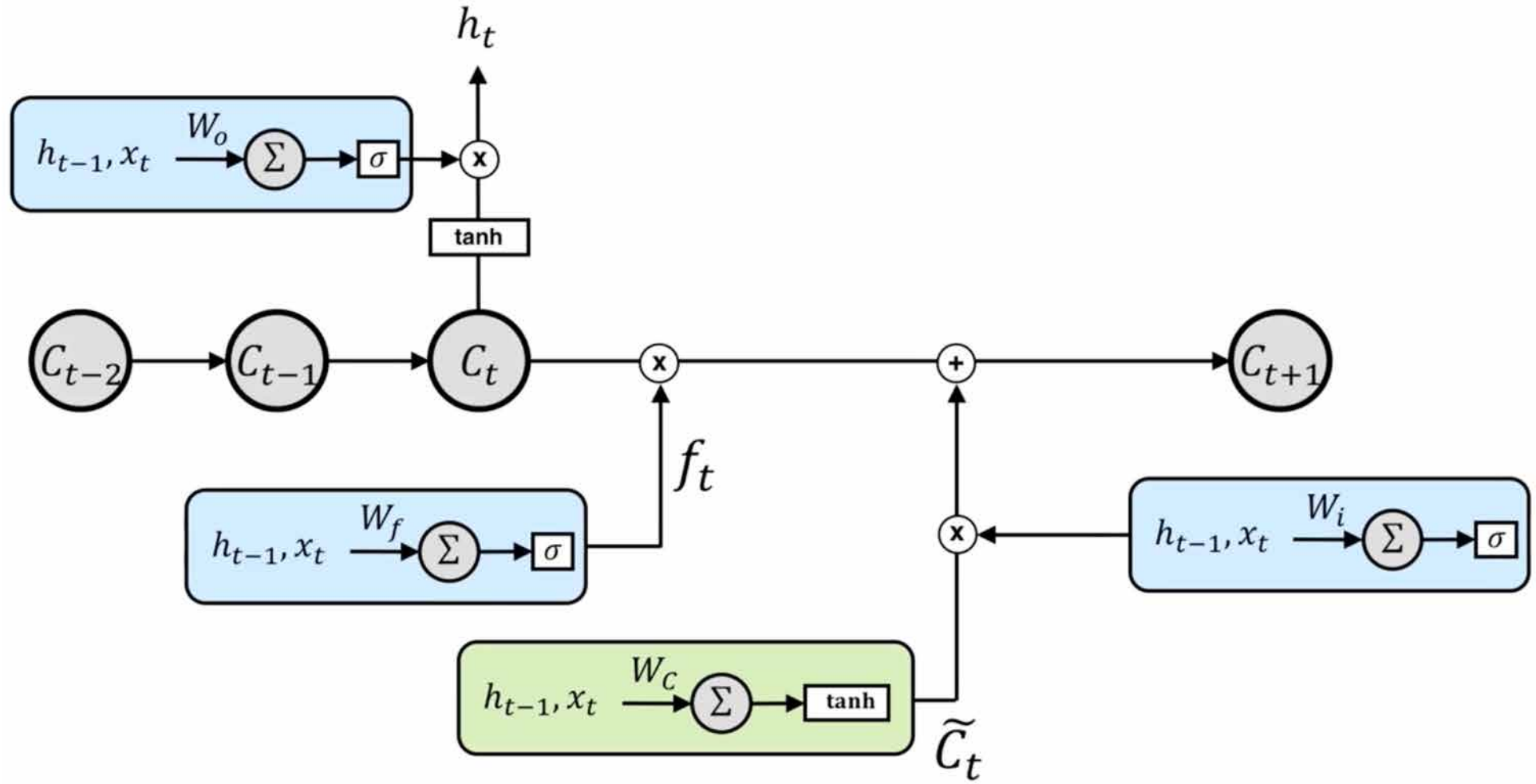
Borrowed from Nervana

LSTM: Input Gate



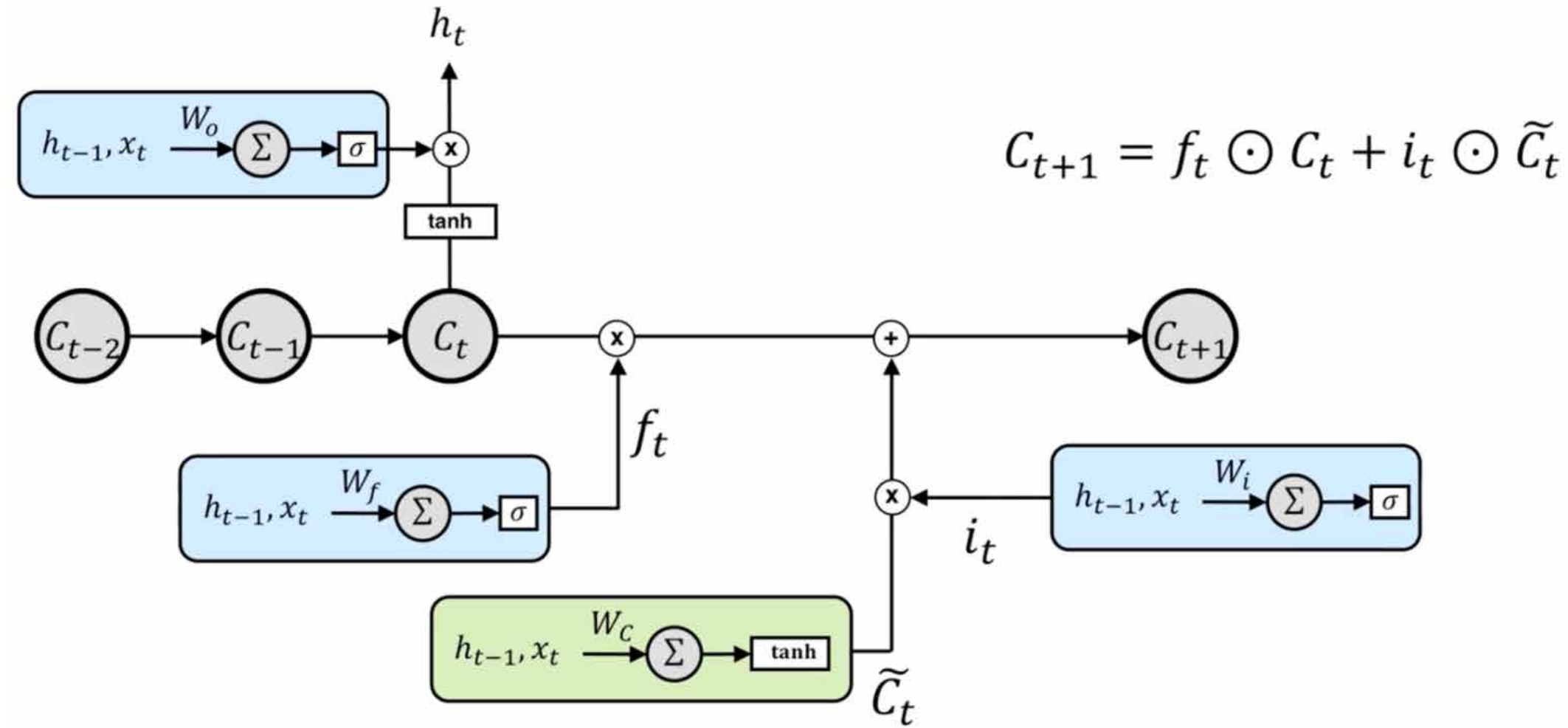
Borrowed from Nervana

LSTM: Input Gate



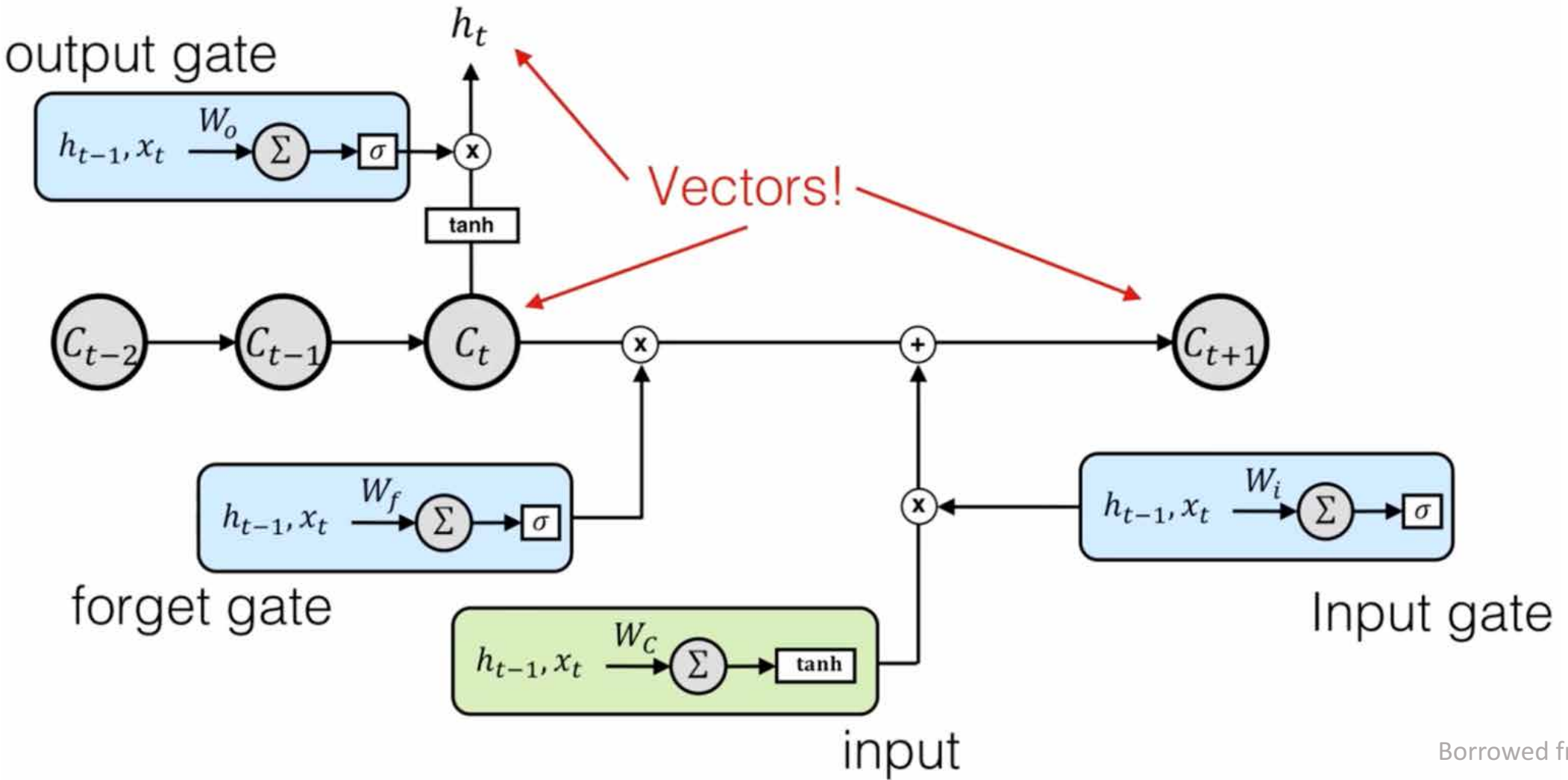
Borrowed from Nervana

LSTM: Input Gate



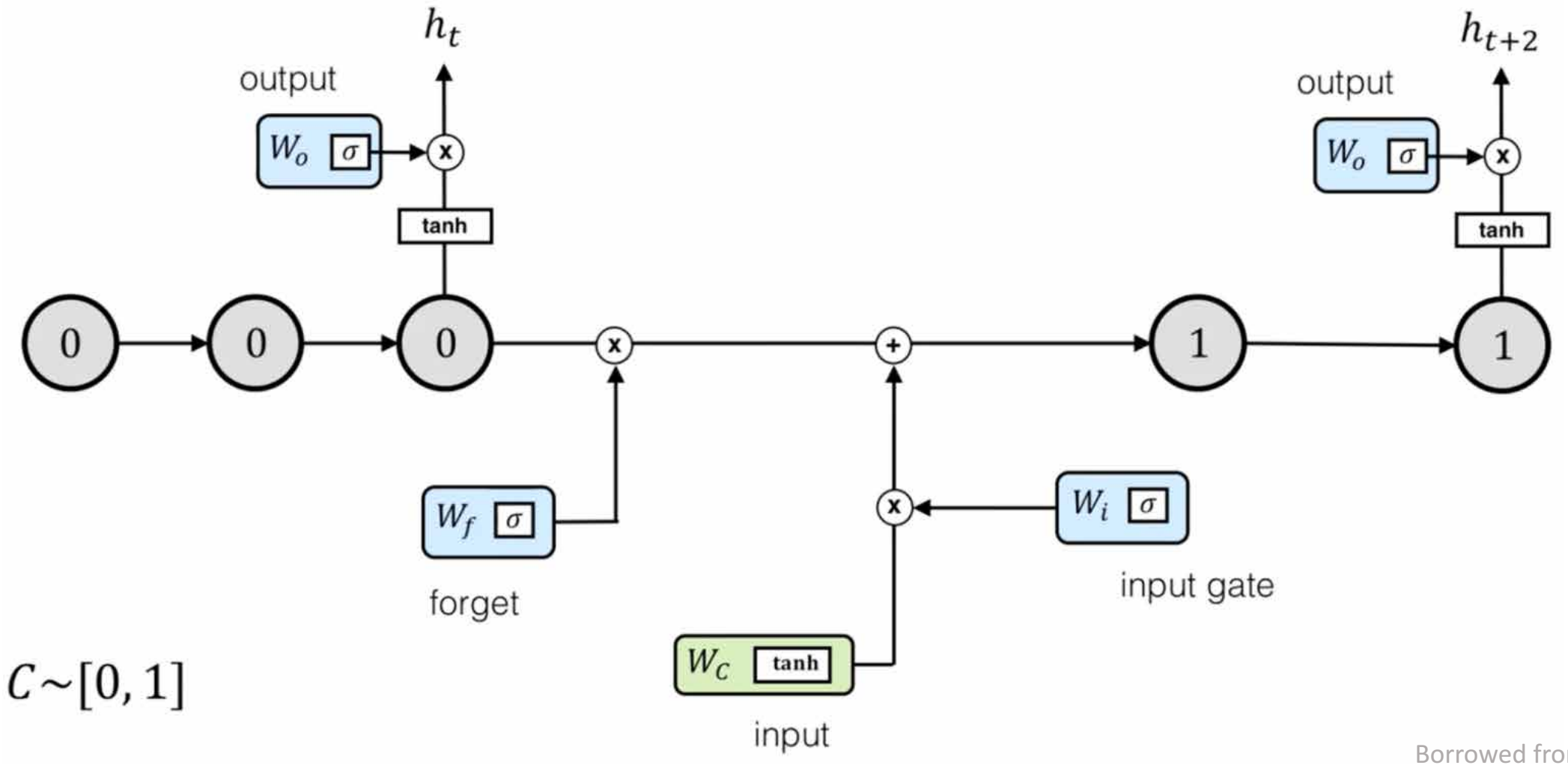
Borrowed from Nervana

LSTM: All Gates



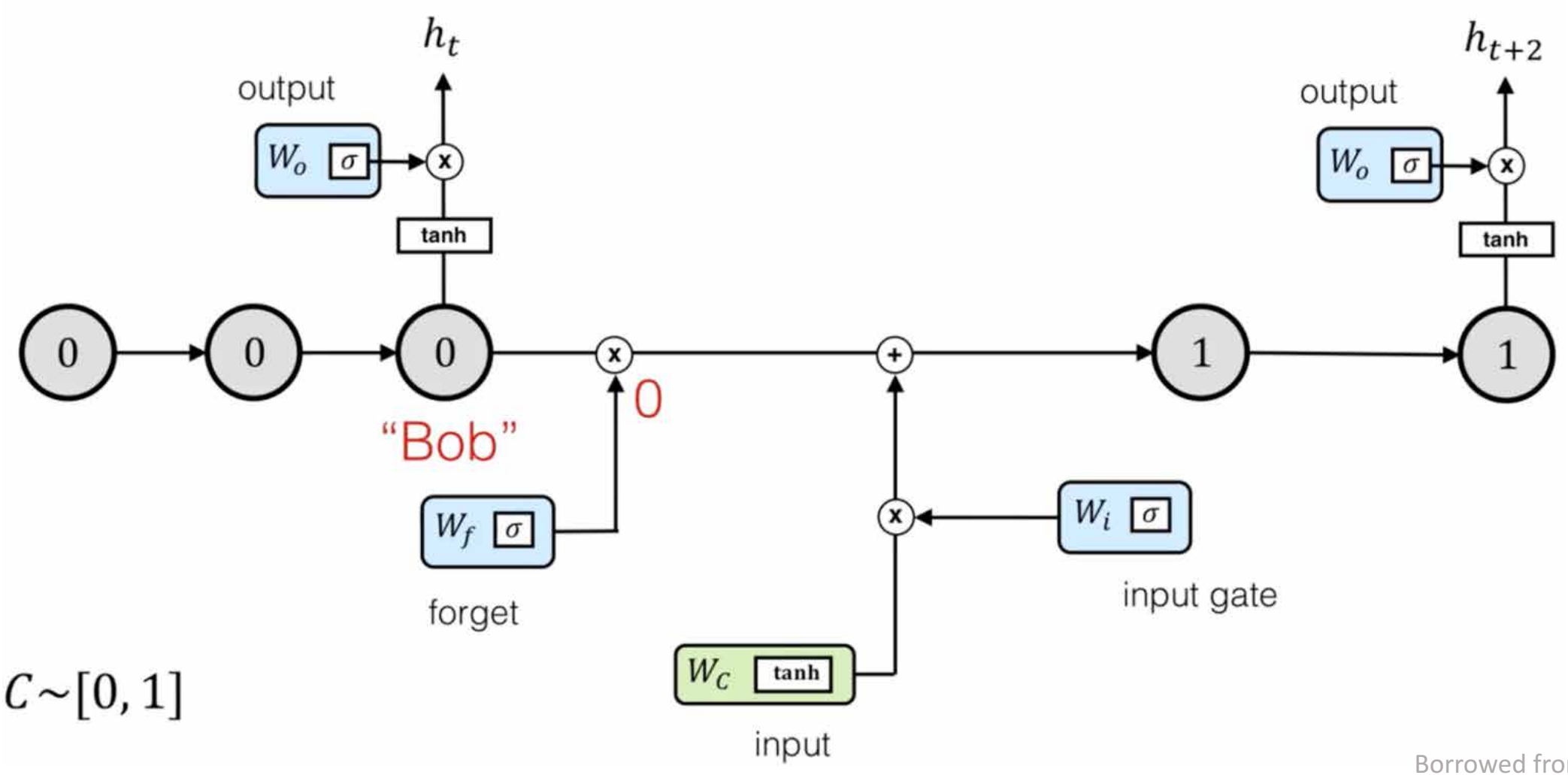
Borrowed from Nervana

LSTM: Example with 1 dimensional Information



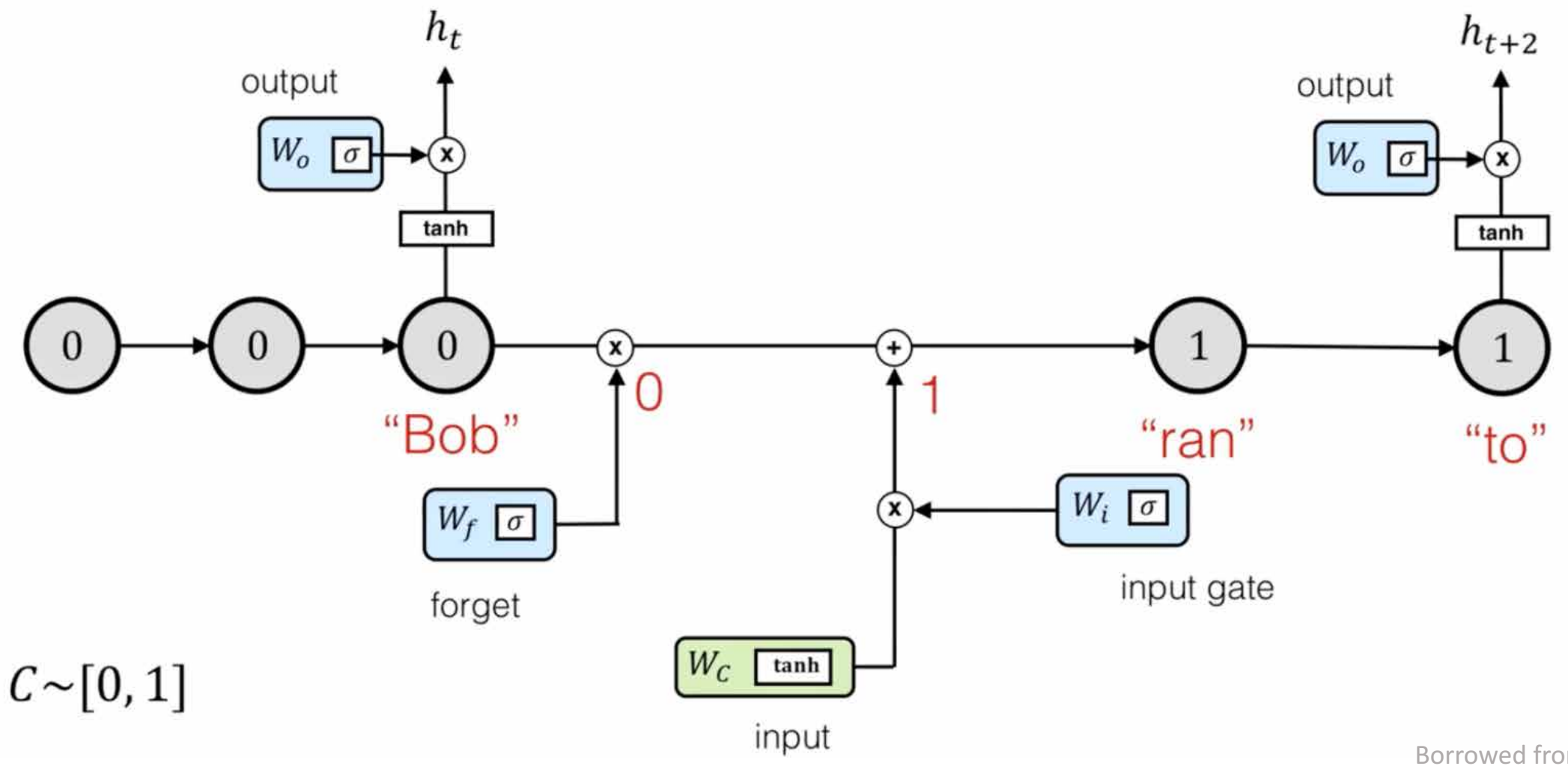
Borrowed from Nervana

LSTM: New Gender “Bob” – Flush Memory



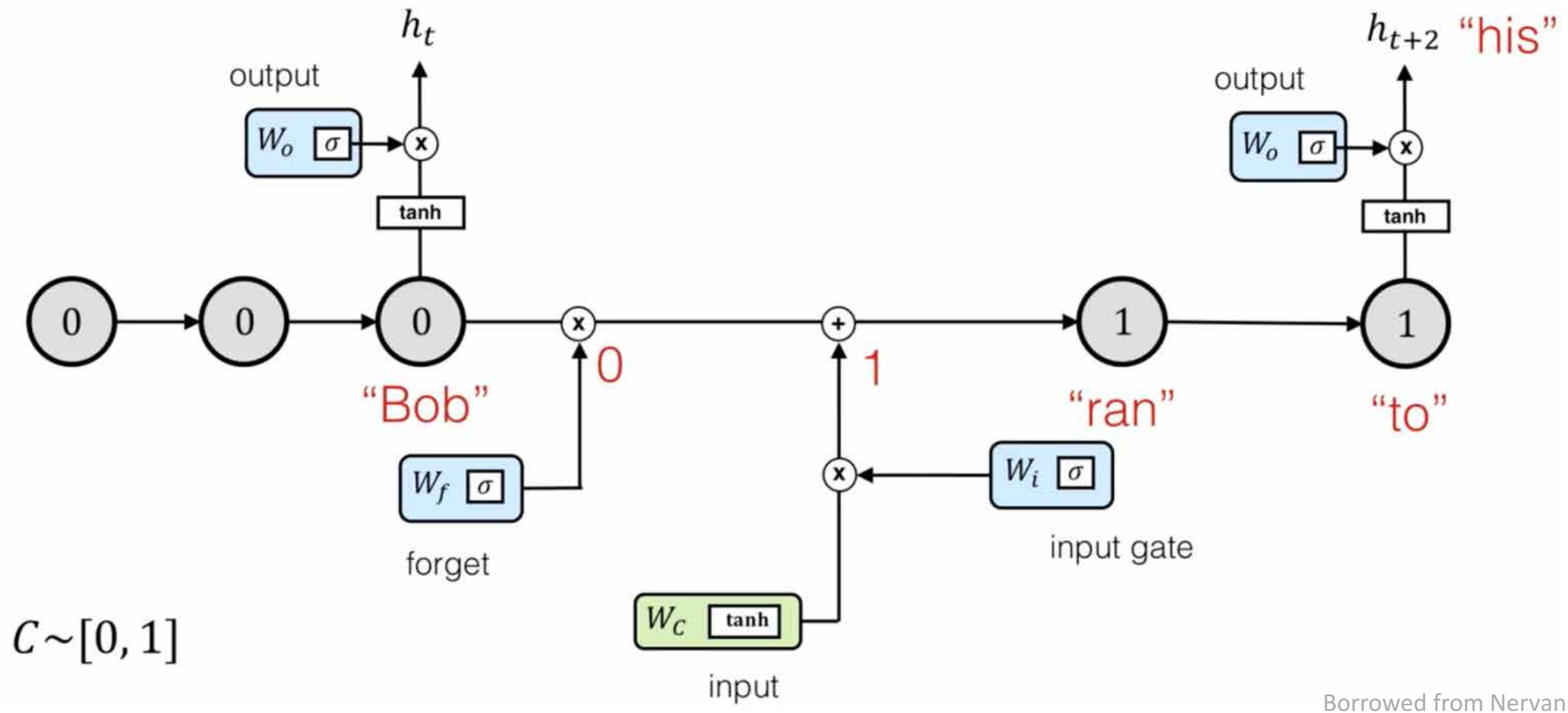
Borrowed from Nervana

LSTM: New Gender “Bob” – Change Gender



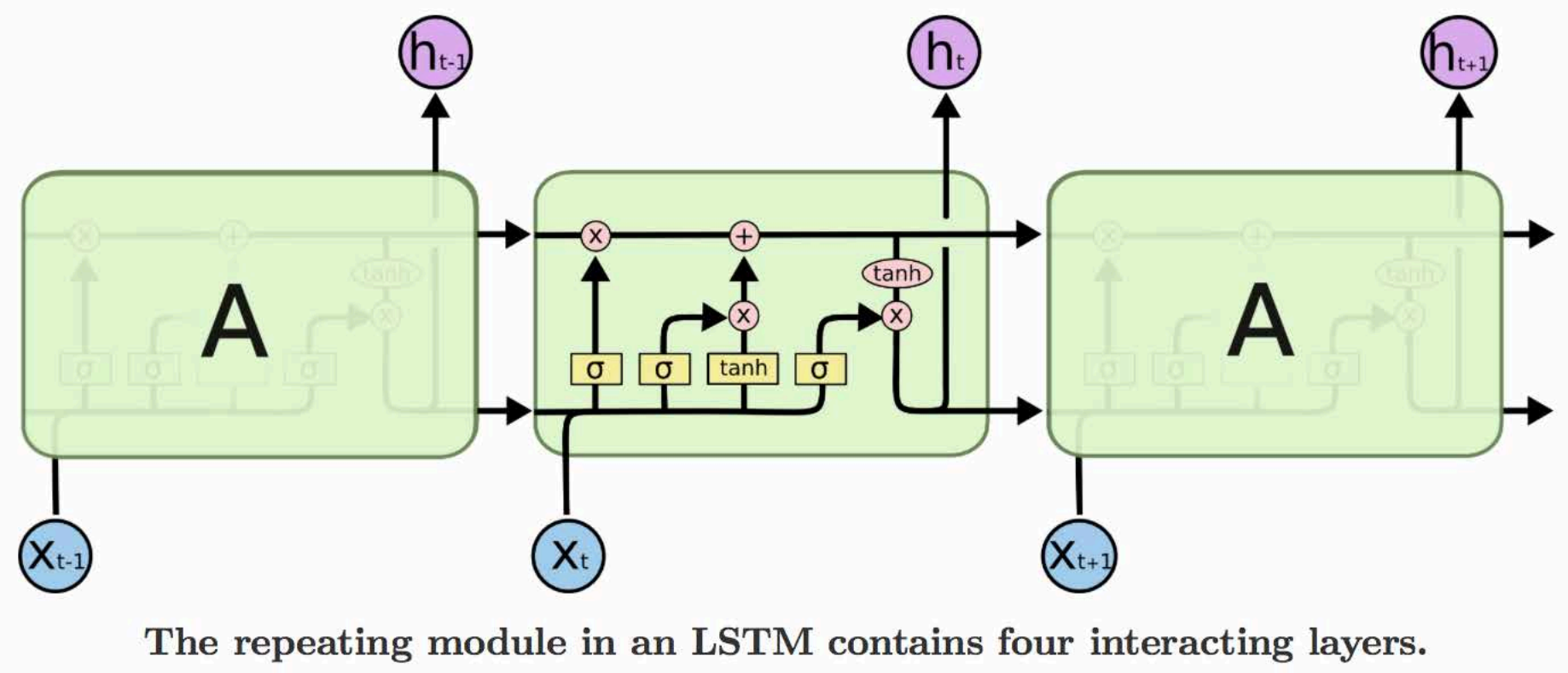
Borrowed from Nervana

LSTM: New Gender “Bob” – Change Gender

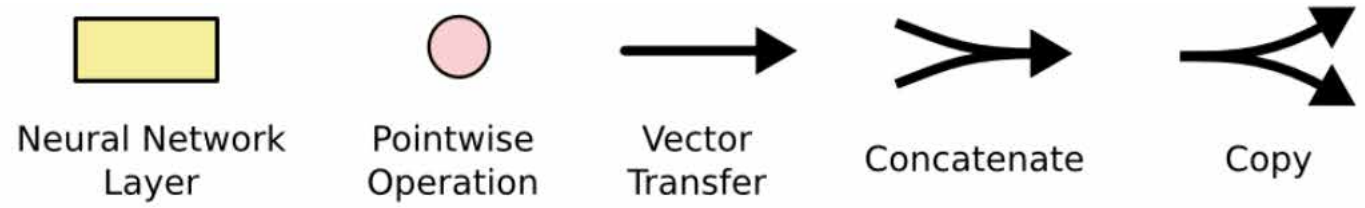


Borrowed from Nervana

LSTM: Let's sum it up again

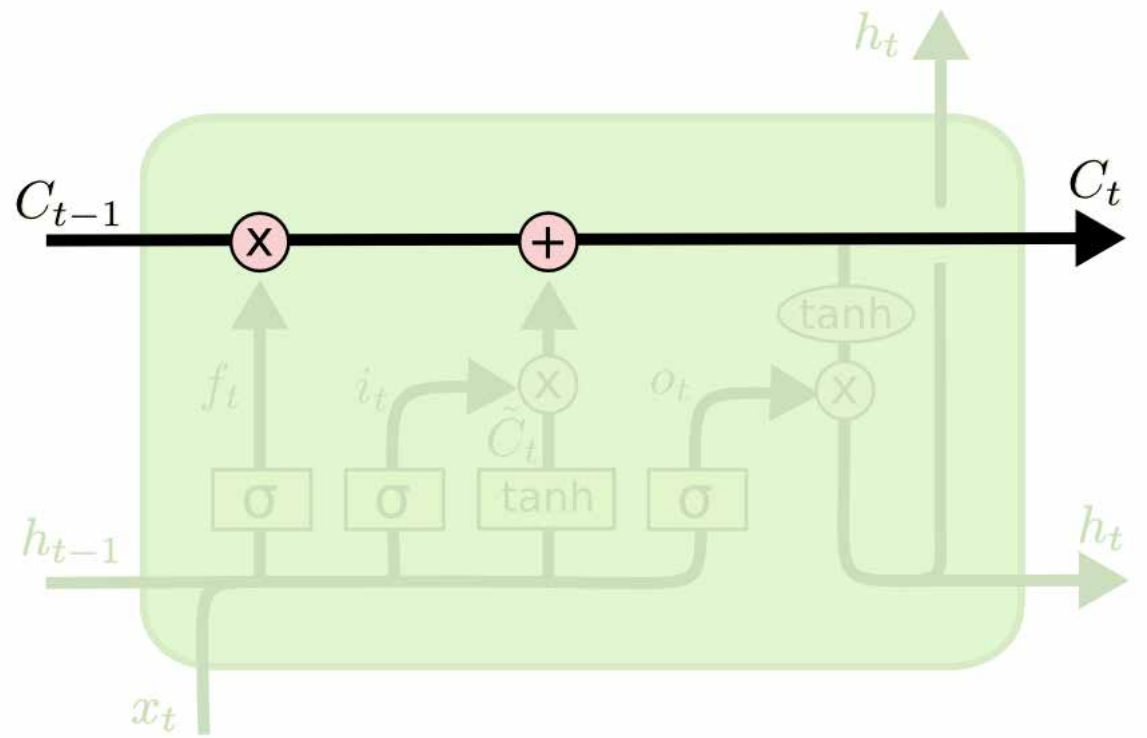


The repeating module in an LSTM contains four interacting layers.



Borrowed from Christopher Olah

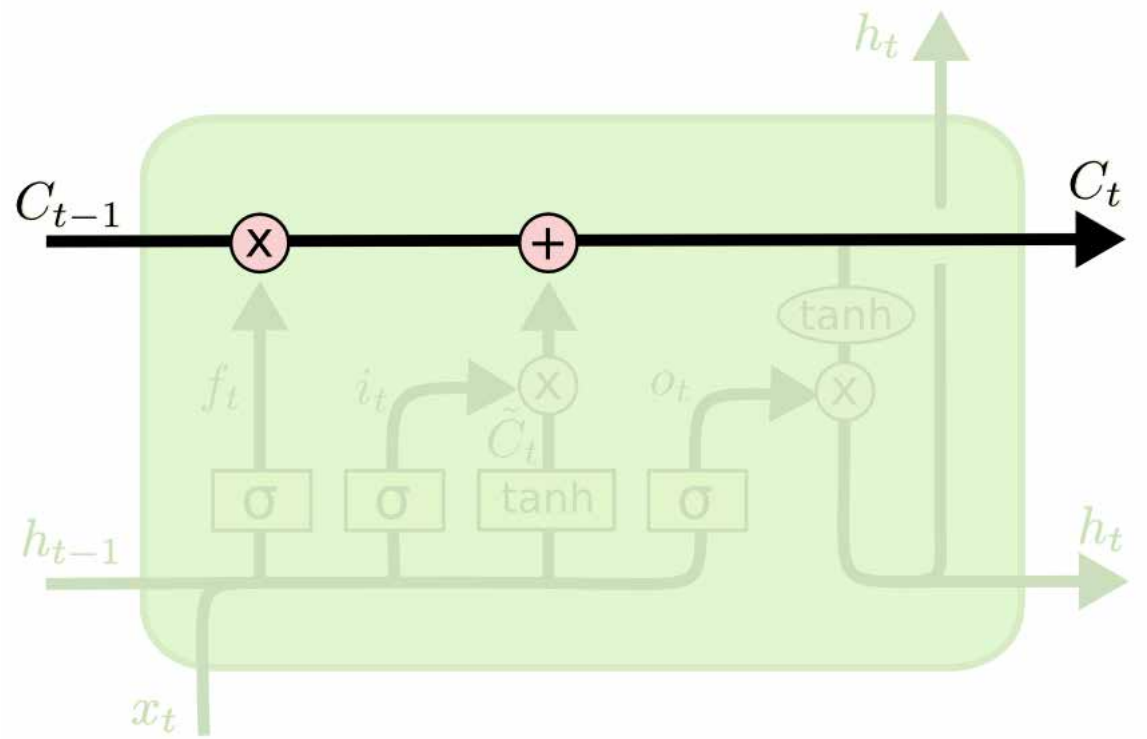
The Core Idea Behind LSTMs: The Cell State

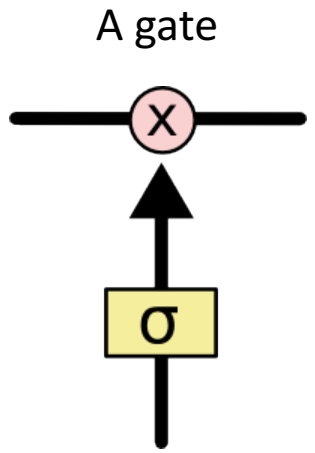


- The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.
- The ***cell state*** is kind of like a ***conveyor belt***. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

Borrowed from Christopher Olah

The Core Idea Behind LSTMs: Gates





- 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.

Borrowed from Christopher Olah

LSTM: Sigmoid Layers

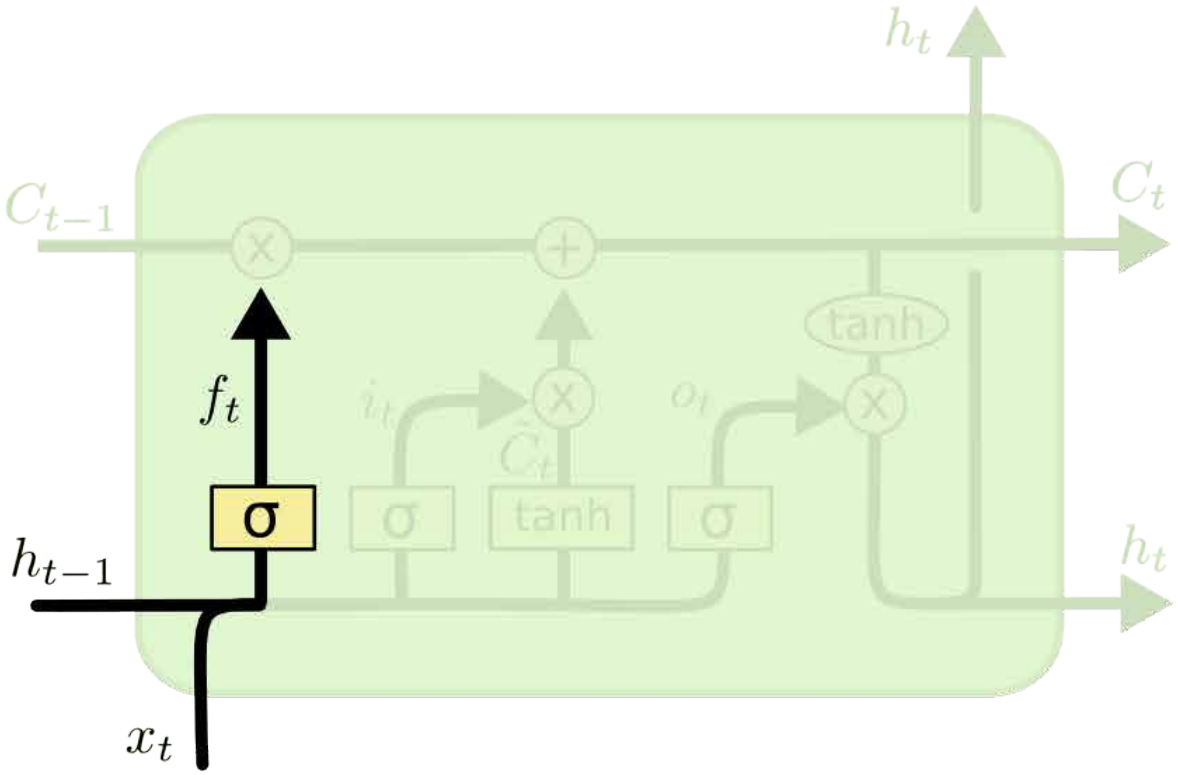
- The sigmoid layer outputs numbers between **zero and one**, describing how much of each component should be let through.
- A value of **zero** means **“let nothing through,”** while a value of **one** means **“let everything through”**

- An LSTM has **three** of these gates, to protect and control the cell state.

Borrowed from Christopher Olah

LSTM Forget Gate (Introduced in 2000)

- 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.

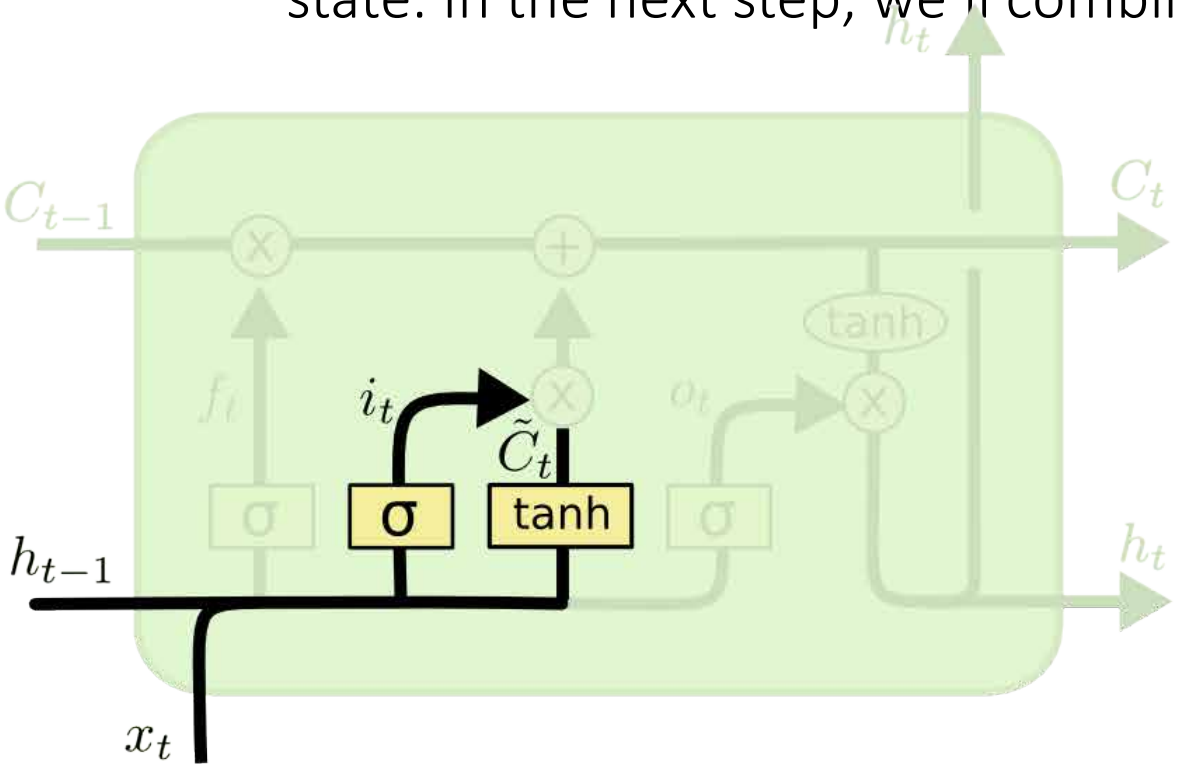


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Borrowed from Christopher Olah

LSTM Input Gate and New Candidate Gate

- New information to store in the cell state:
  - A sigmoid layer called the “input gate layer” decides which values we’ll update.
  - A ***tanh*** layer creates a vector of new candidate values that could be added to the state. In the next step, we’ll combine these two to create an update to the state.



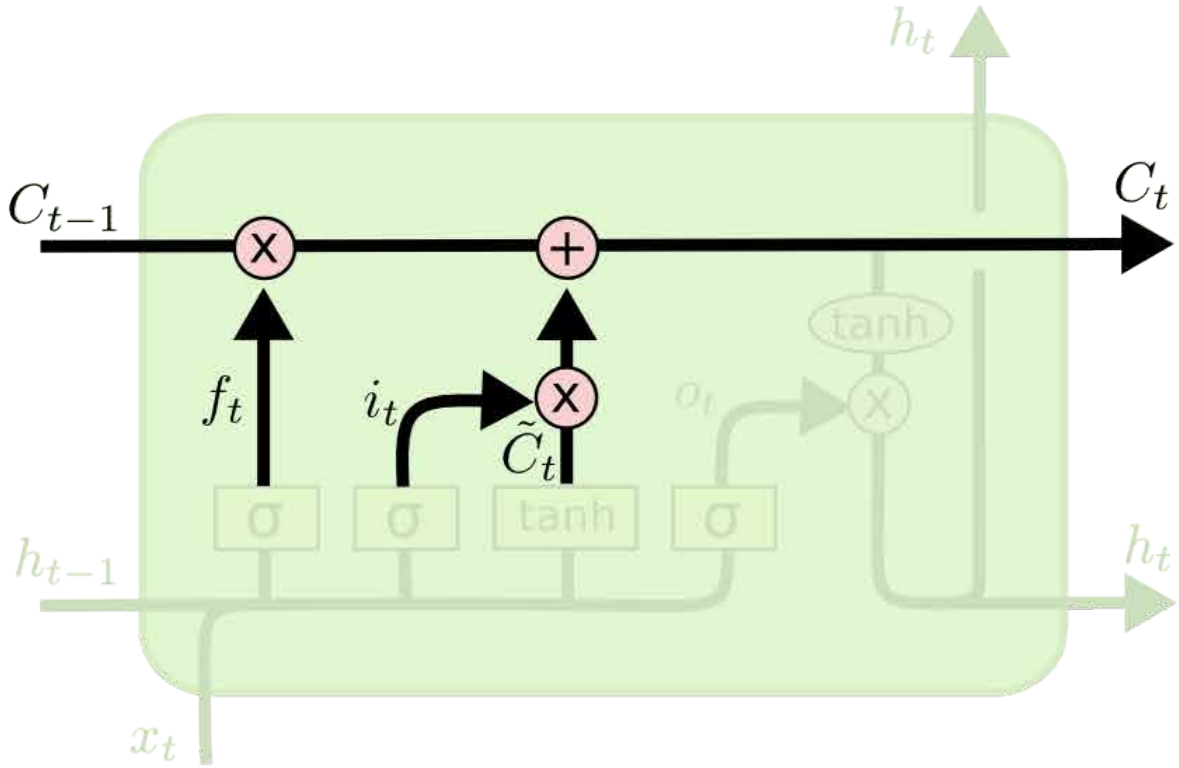
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

Borrowed from Christopher Olah

LSTM Input Gate and New Candidate Gate

- 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.

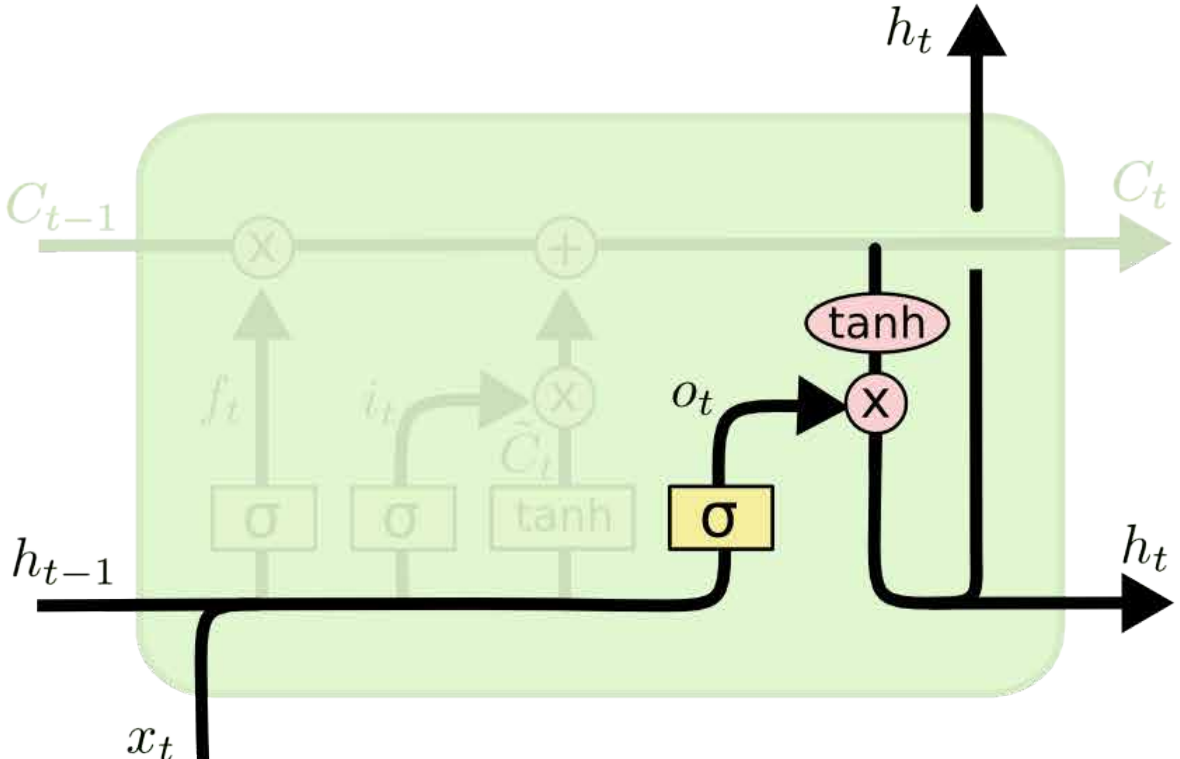


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Borrowed from Christopher Olah

LSTM Filtered Output Gate

- 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.

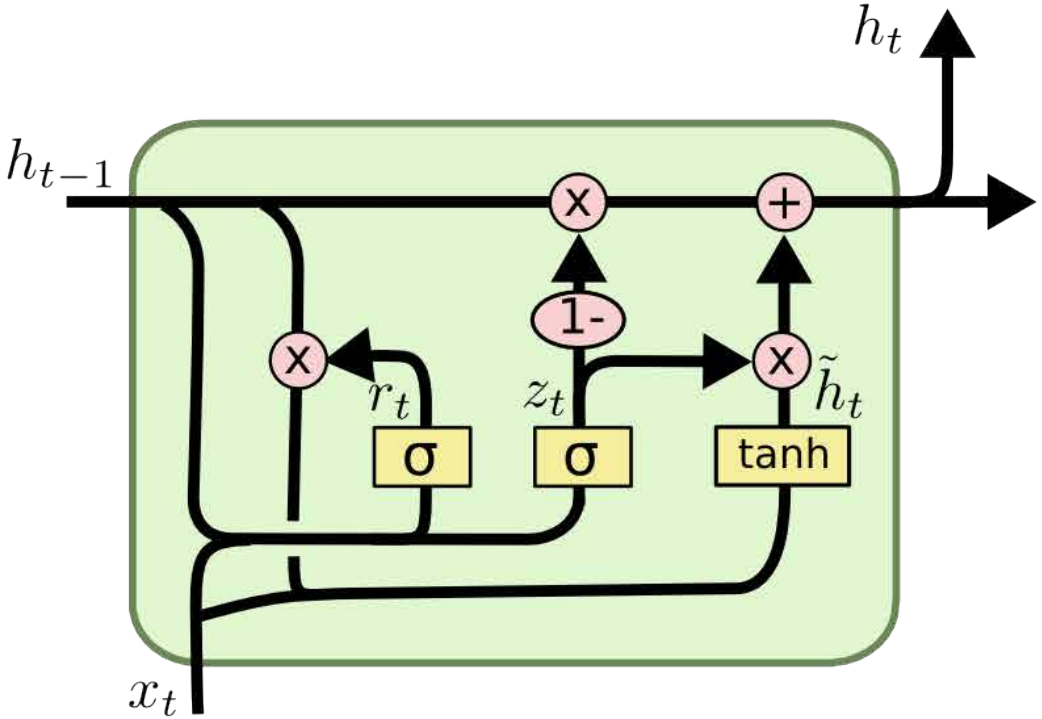


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

Borrowed from Christopher Olah

LSTM Variants: Gated Recurrent Unit



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Deep Recurrent Neural Networks (RNNs)

RNN:

$$h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$h \in \mathbb{R}^n$ $\quad W^l \ [n \times 2n]$

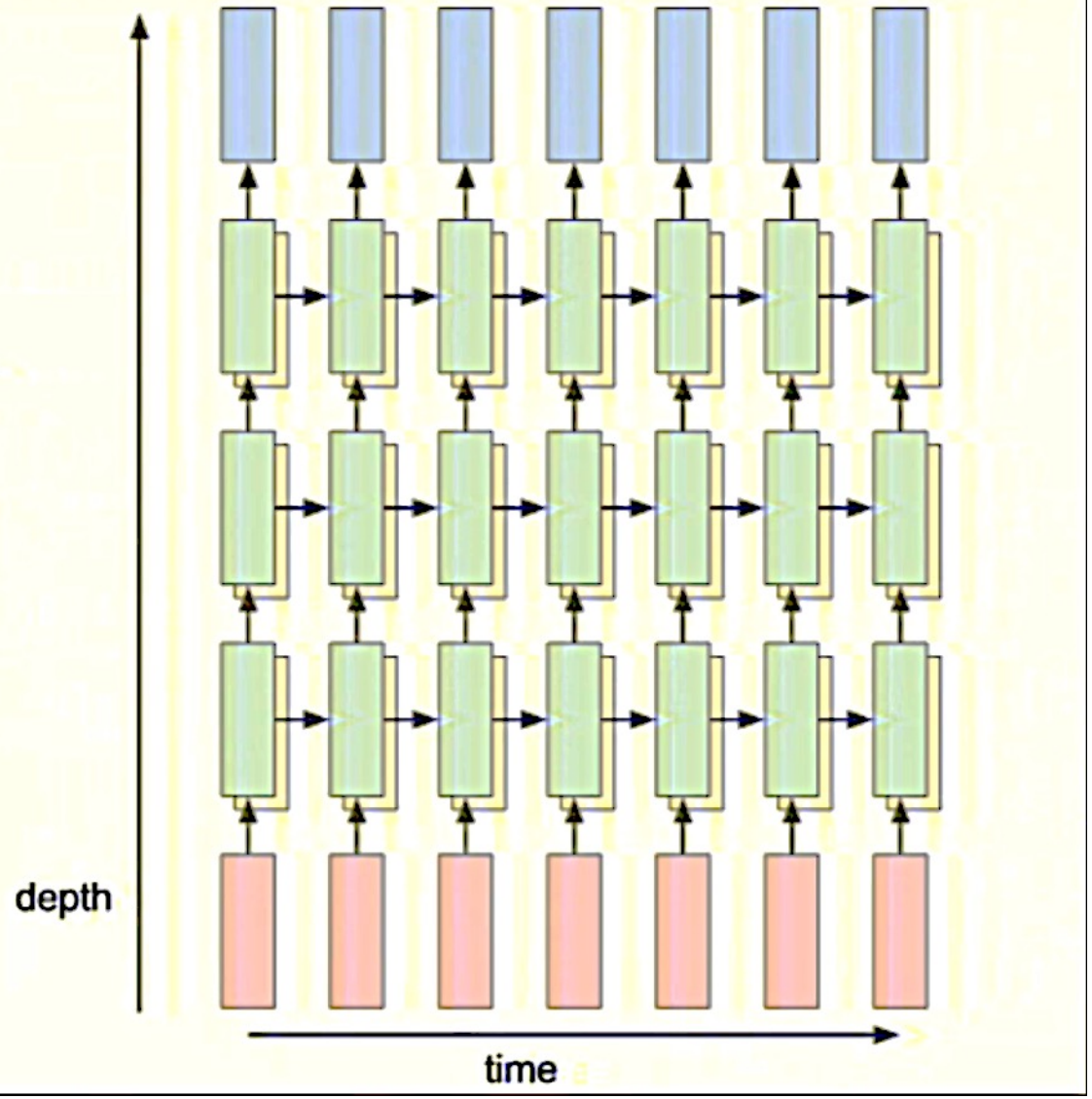
LSTM:

$W^l \ [4n \times 2n]$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

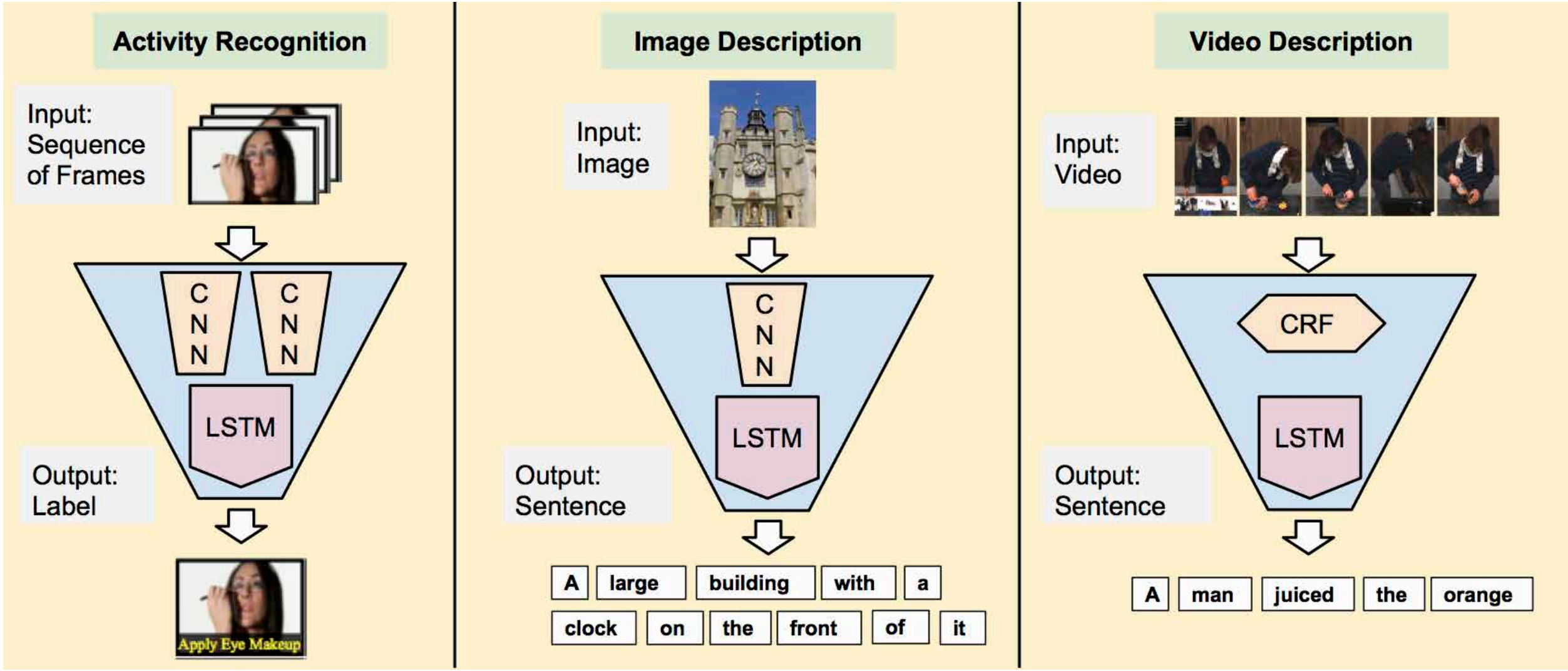
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$



Borrowed from Andrej Karpathy

Teaser #1: Long-term Recurrent Convolutional Networks (LRCN)

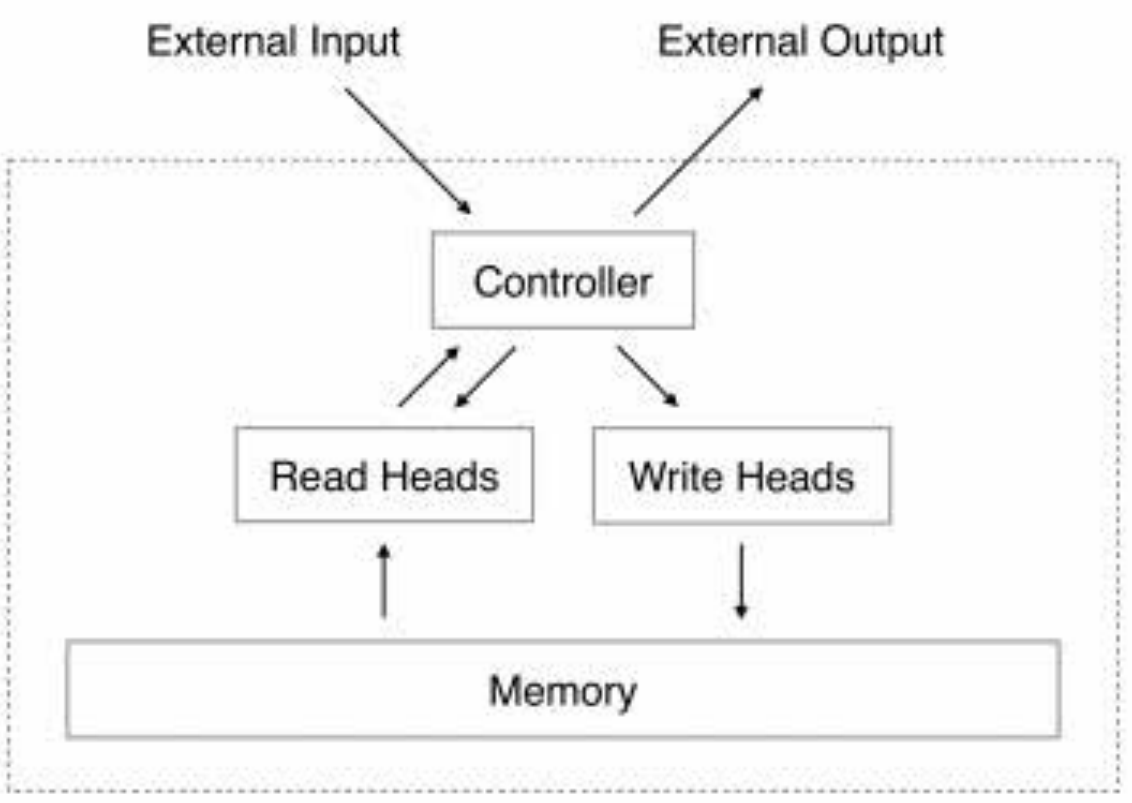


Long-term Recurrent Convolutional Networks (LRCN)

https://www.youtube.com/watch?v=w2iV8gt5cd4

Teaser #2: Neural Turing Machines (Graves et al., 2014)

- A Neural Turing Machine (NTM) also tries to form a memory system, but it actually creates a separate memory outside the network (amazingly simple structure to look at), and the network does *read* and *write* operations on it.



Universal Turing Machine?

- A well-known result is that
  - a finite-sized recurrent neural network with sigmoidal activation functions can simulate a universal Turing machine [Siegelmann and Sontag, 1991].
- The capability of RNNs to perform arbitrary computation demonstrates their expressive power.
- One could argue that the C programming language is equally capable of expressing arbitrary programs.
- What is special about NNs and RNNs in particular compared to other languages such as C, etc…?

Universal Turing Machine?

- Conventional programming languages vs UTM
  - there is no simple way of efficiently exploring the space of C programs.
  - There is no general way to calculate the gradient of an arbitrary C program to minimize a chosen loss function.
- Moreover, given any finite dataset, there exist countless programs which over-fit the dataset, generating desired training output but failing to generalize to test examples.

Differentiable Neural Computer
Family tree inference task
(artistic rendering)

One final note!

- Unfortunately we don’t have a very fast GPU cluster to experiment on, right now, for our students.
- Fortunately, each and every one of you have **”The GPU”**
- **“The GPU”** is your **BRAIN**! ☺
- You know RNNs train very easily with repetition.
- Use it.

- BEYİN BEDAVA!