CS 4104
APPLIED MACHINE LEARNING

Dr. Hashim Yasin

National University of Computer and Emerging Sciences,

Faisalabad, Pakistan.

LSTM

LSTM

Long Short-Term Memory

- To solve the problem of Vanishing Gradient, LSTM (a modified versions of RNN) can be used.
- LSTM can remove or add information to the cell state, carefully regulated by structures called **gates**.
  - Gates are a way to optionally let information to go through.
  - They are composed of a sigmoid layer and a point wise multiplication operation.

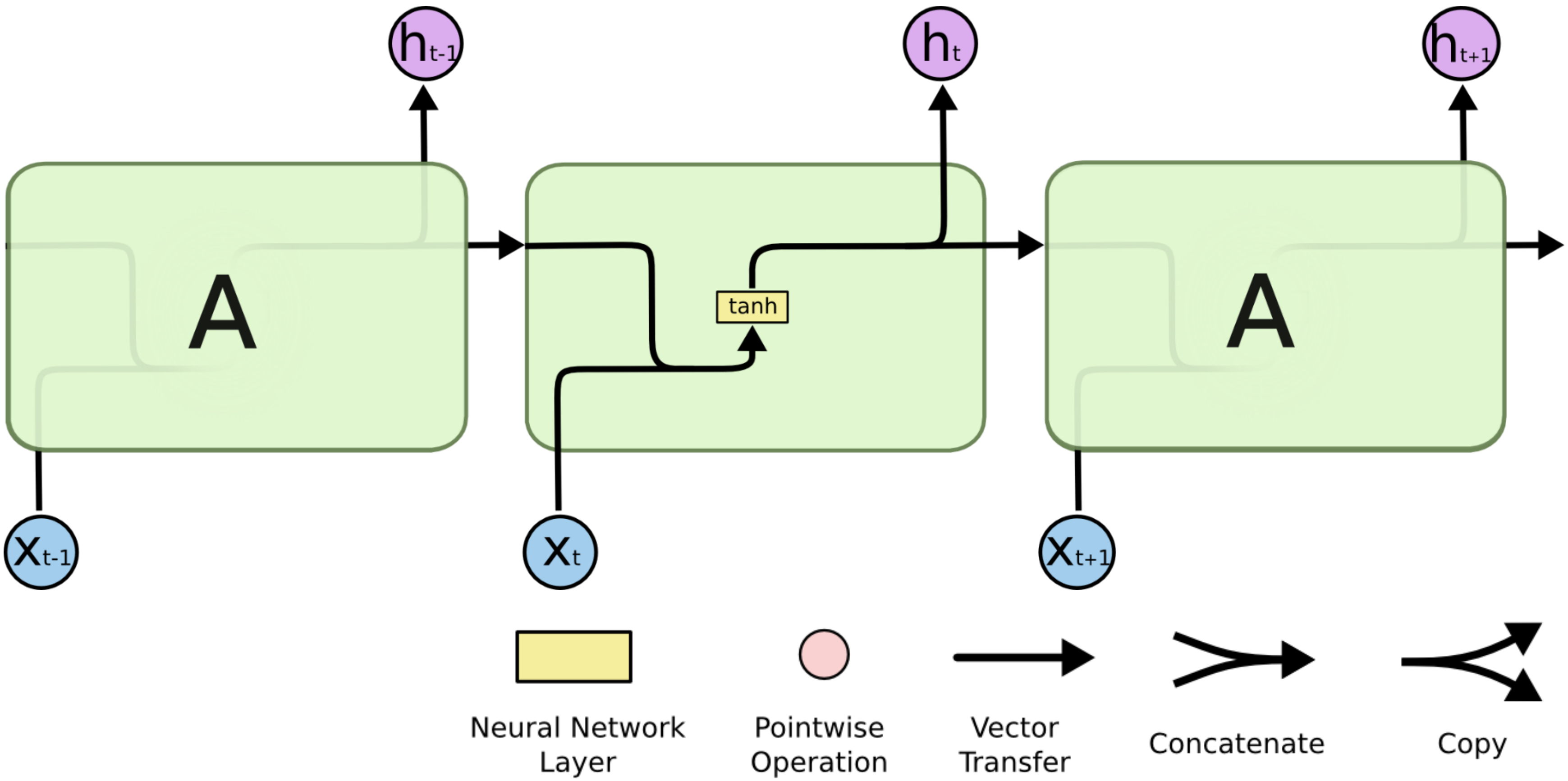
LSTM

Long Short-Term Memory

- An LSTM has three gates.
  - An "**input**" gate controls the extent to which *a new value flows into the memory*;
  - a "**forget**" gate controls the extent to which *a value remains in memory*;
  - an "**output**" gate controls the extent to which the *value in memory is used to compute the output activation of the block*, to *protect and control the cell state* (information flows along it).

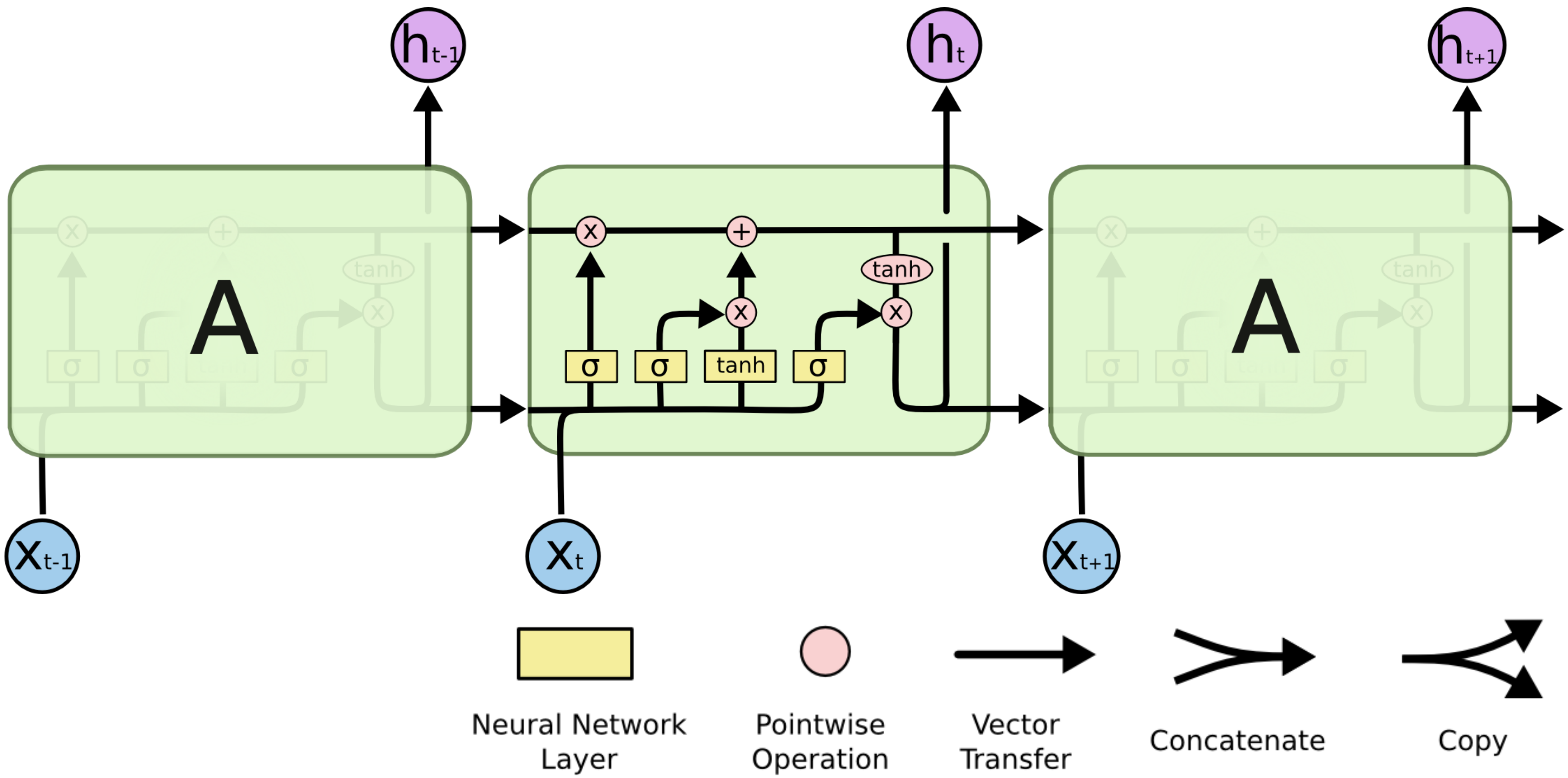
LSTM

A simple RNN

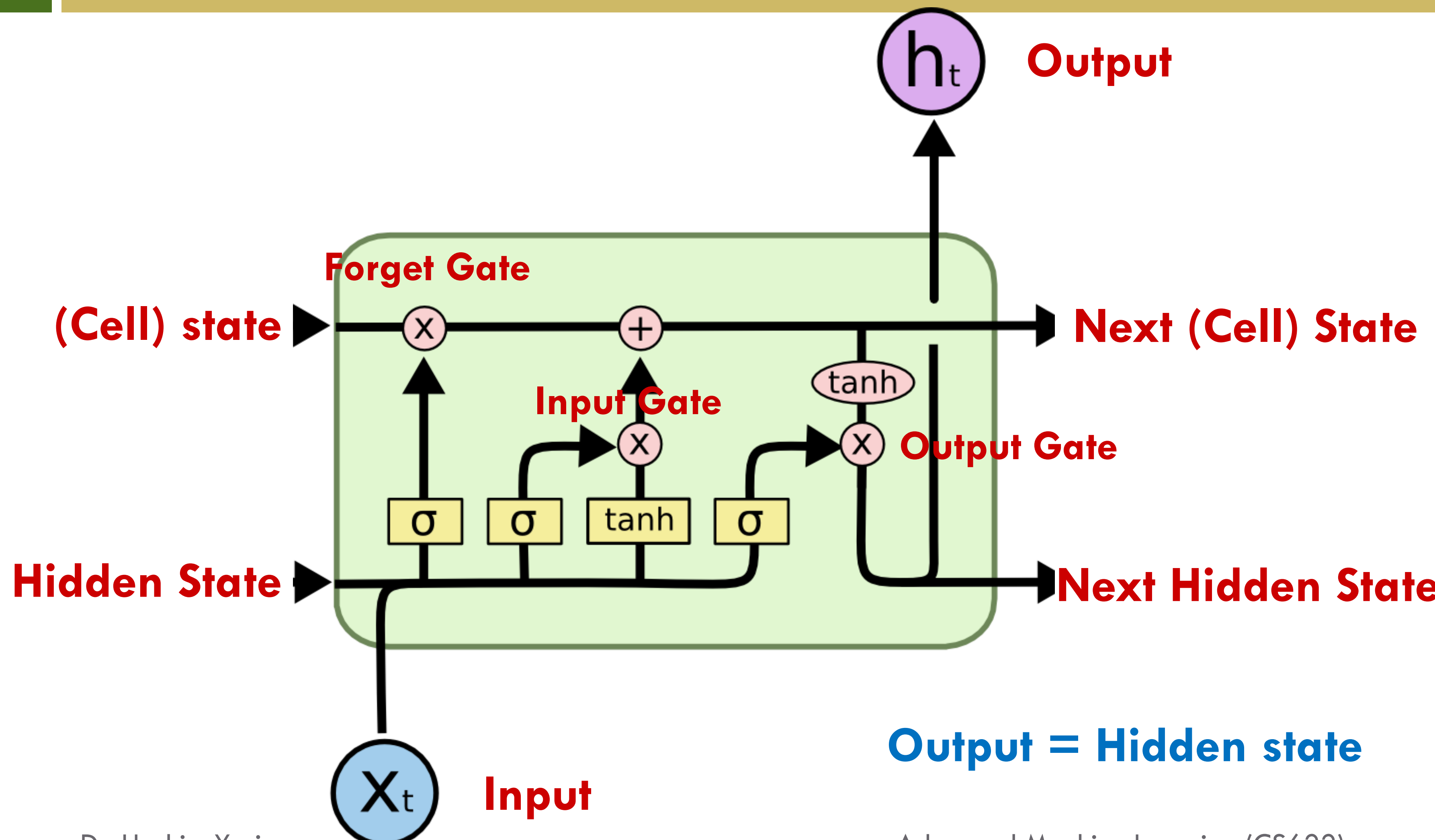


LSTM

A simple LSTM structure



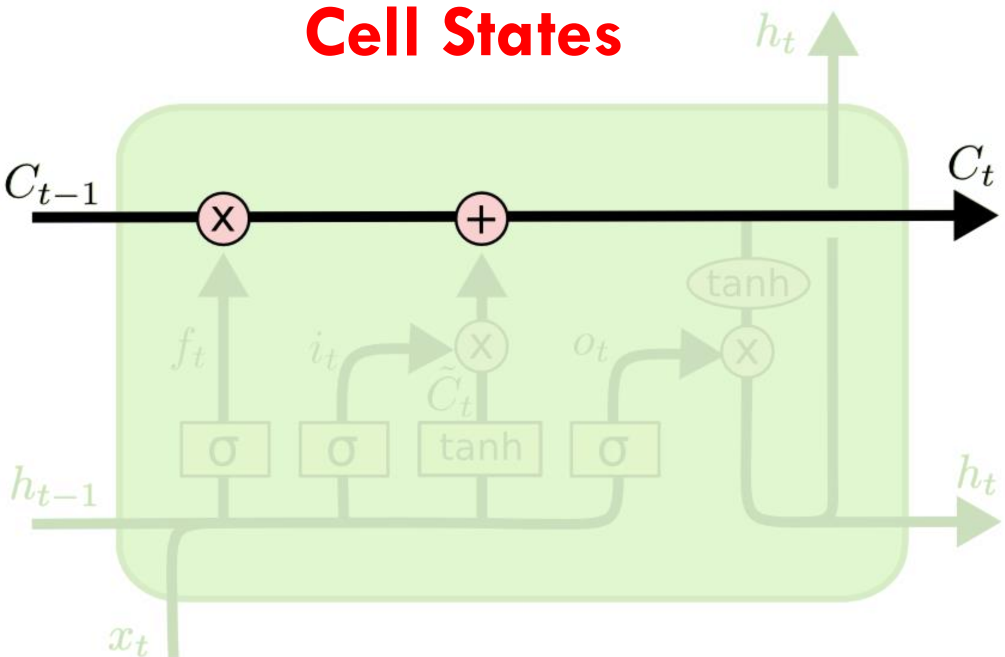
LSTM Architecture



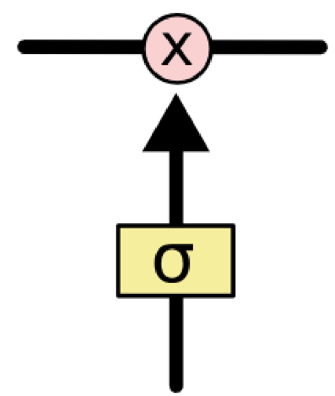
Output = Hidden state

LSTM Architecture

Cell States



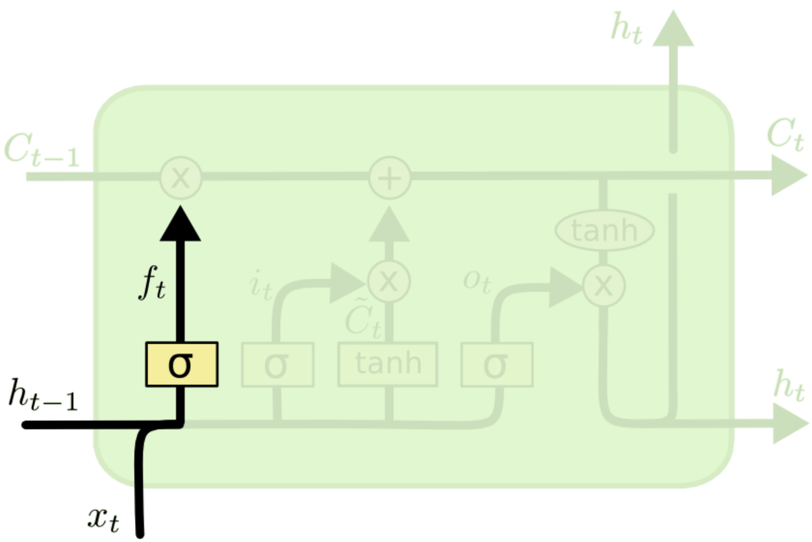
Gates



- **Cell state:** which works like a ***conveyor belt*** runs straight down the entire chain, easy for information to flow along without changes.
- **Gates:** which control or decide *what kind of information could go or throw away from the cell state*.

LSTM Architecture

1. Forget gate layer

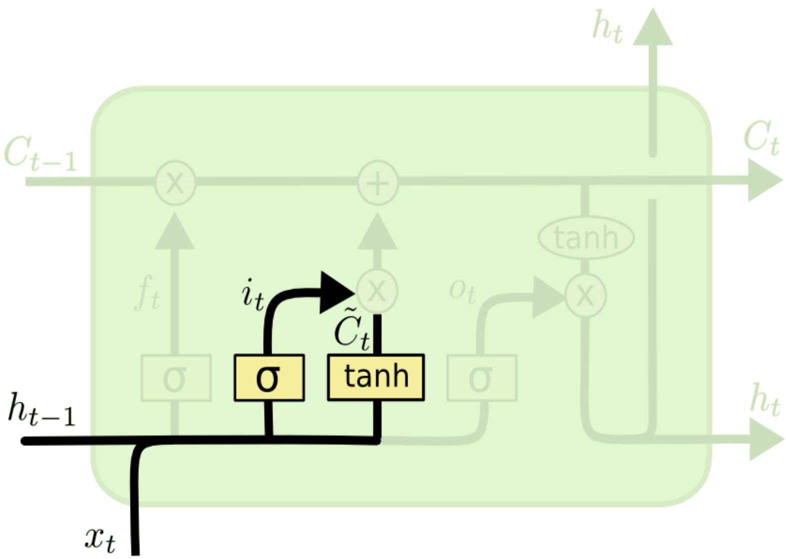


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

- Decide what information we're going to **throw away** from the cell state.
- It gives a value between 0 and 1, where a 1 represents "keep this as it is" while a 0 represents "get rid of this."

LSTM Architecture

2. Input gate layer



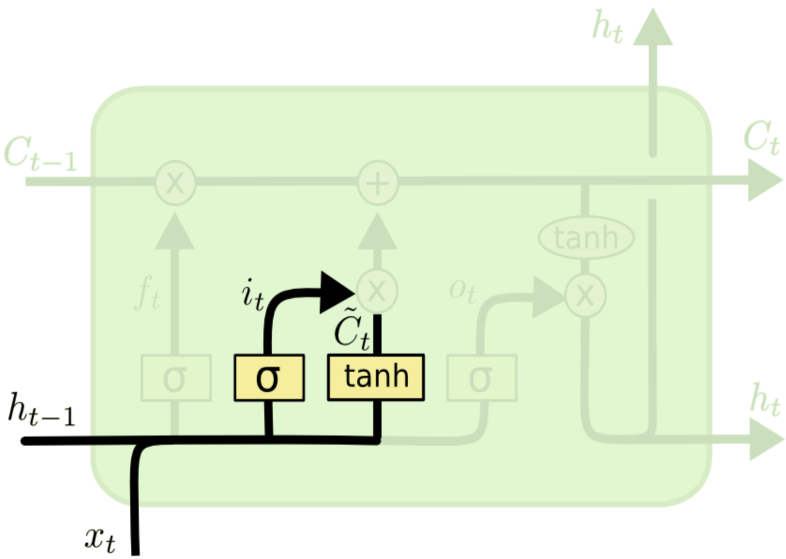
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

- Decide what **new** information we're going to **store** in the cell state.
- This step has two parts:

LSTM Architecture

2. Input gate layer



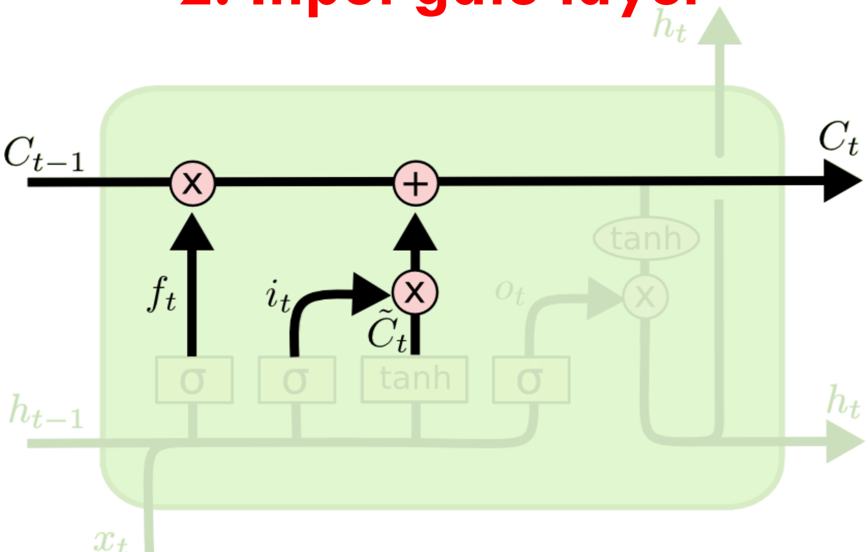
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

- **First**, a *sigmoid layer* called the "input gate layer" decides which values we'll update.
- **Second**, a *tanh layer* creates a vector of new candidate values $\tilde{C}_t$ that could be added to the state.

LSTM Architecture

2. Input gate layer

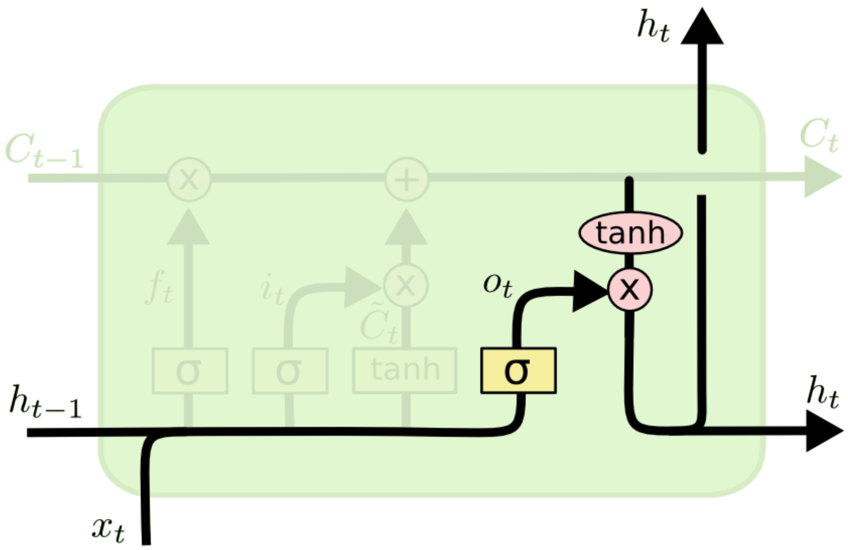


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- It is now time to update the old cell state, C_{t-1}, into the new cell state C_t.

LSTM Architecture

3. Output gate layer

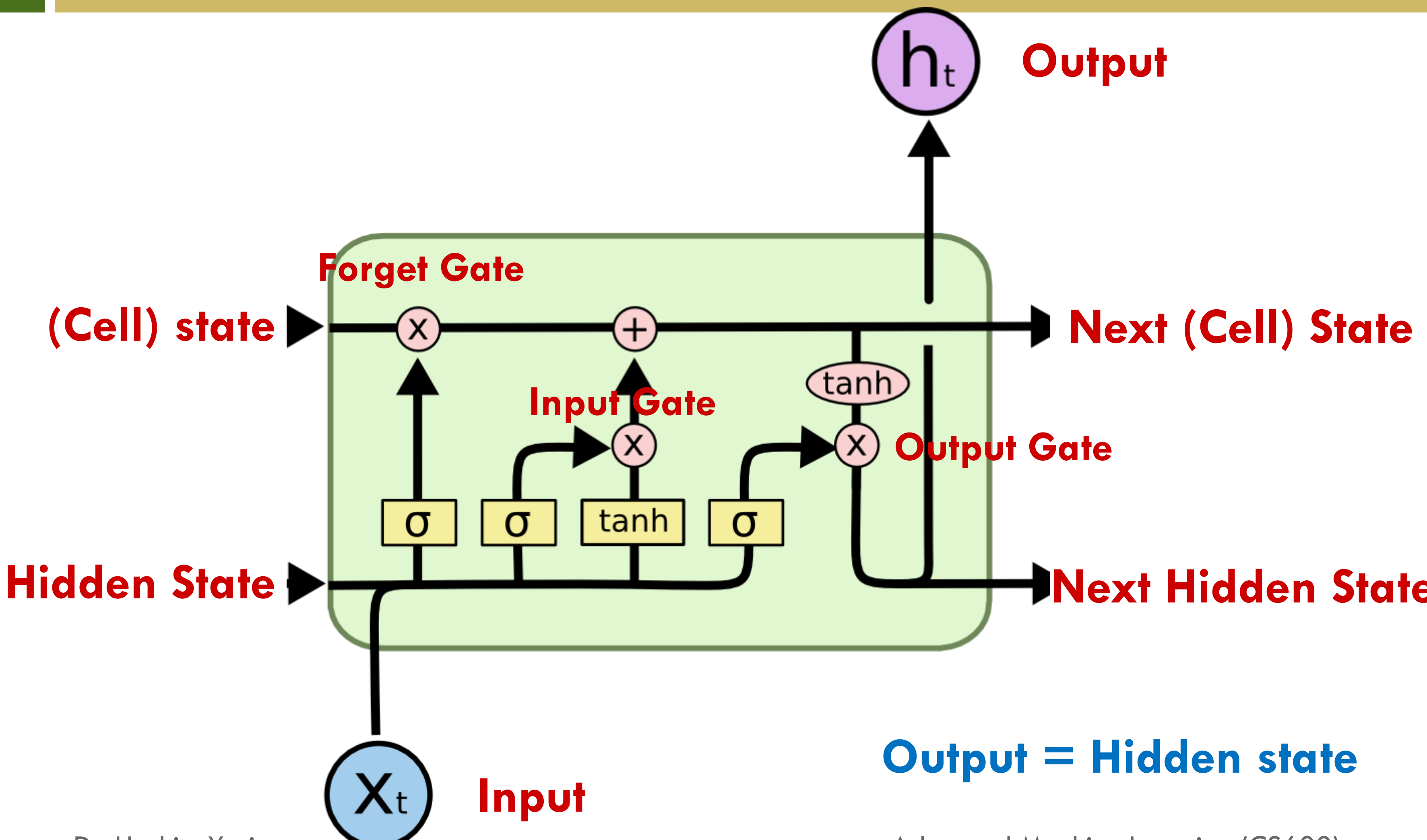


$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

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

- Output based on the updated state

LSTM Architecture

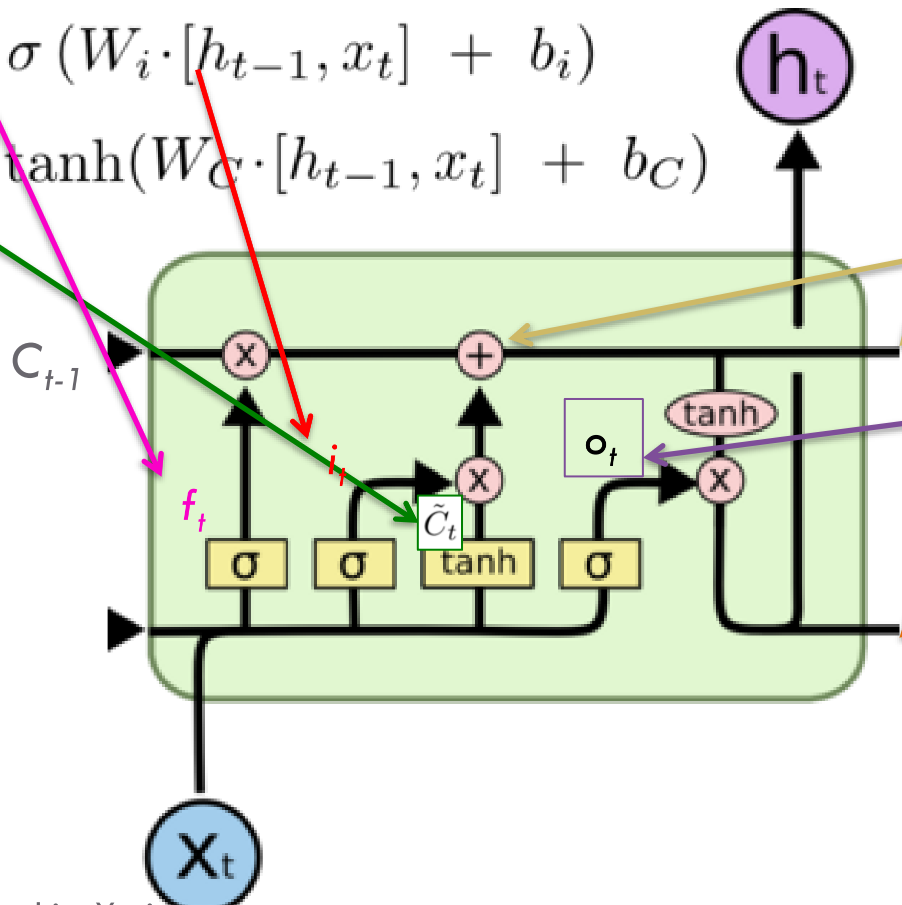


LSTM Architecture

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

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

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



(2)

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

(3)

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

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

Implementation of LSTM

For $t = 1,\ldots,T$:

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

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

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

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

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

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