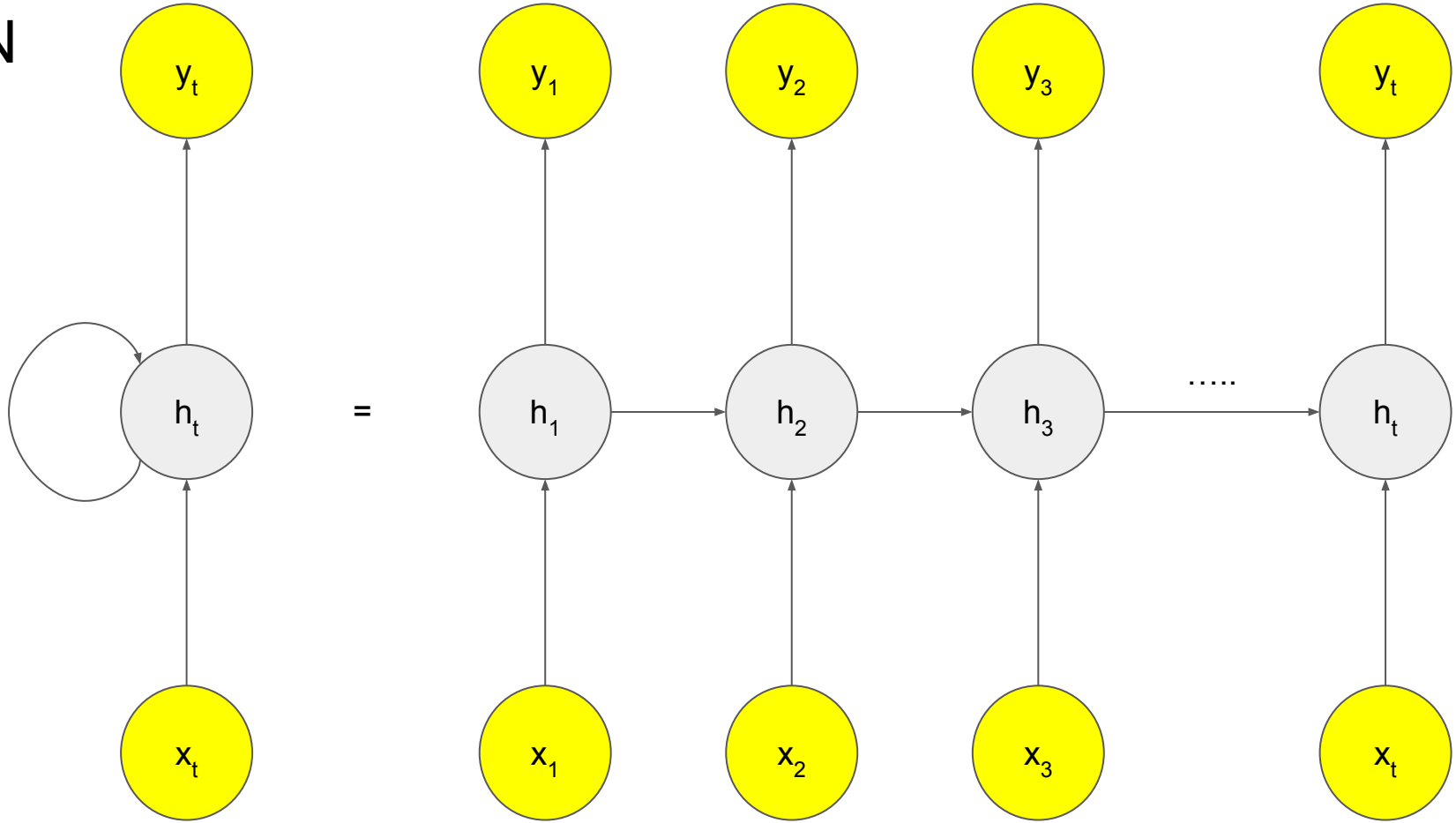
Recurrent Neural Networks

Intro to Deep Learning

RNN



RNN

When the first input (x_1) comes in, the first memory (h_1) is created. When the second input (x_2) comes in, the existing memory (h_1) is referenced along with the new input to create a new memory (h_2). This process can be repeated for any length of inputs.

In short, the input is (x) from the data and memory (h), and the output becomes y along with the memory (h).

RNN type

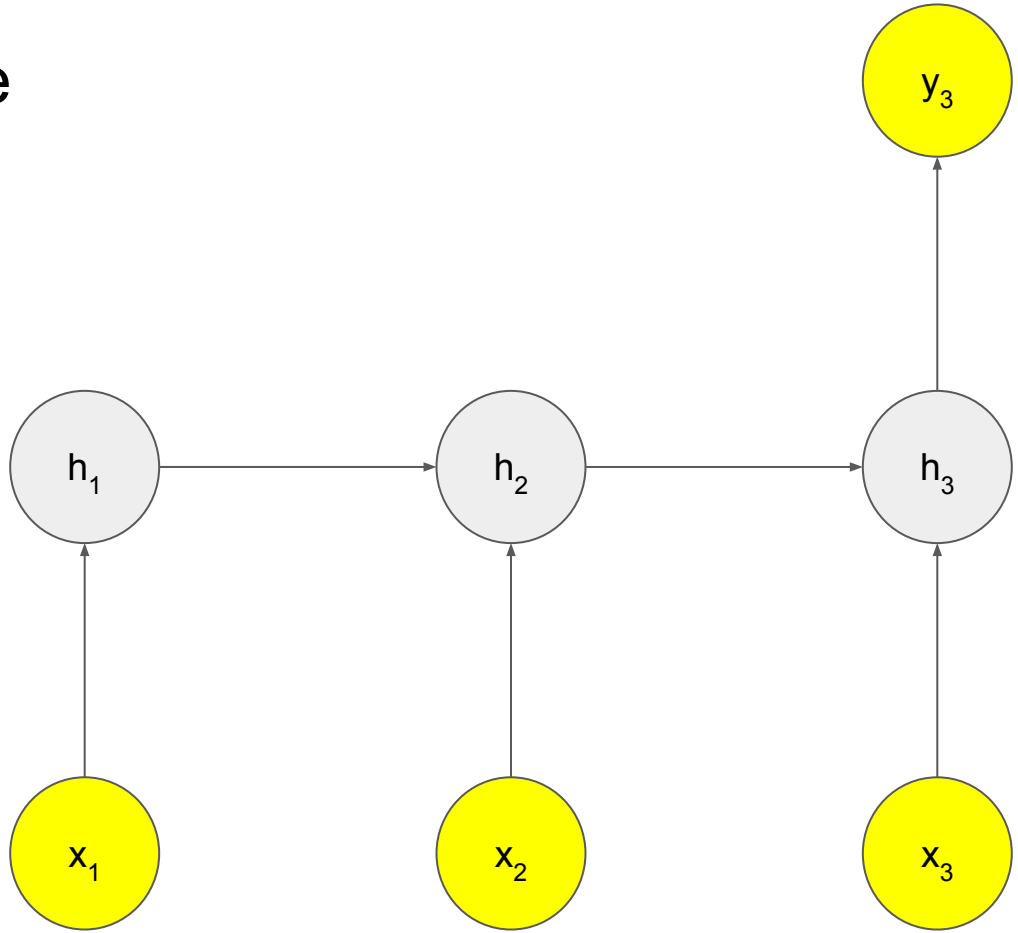
One-to-One: It is difficult to call it an RNN because there is no recurrence. It is a basic neural network structure where each input produces one output, without any recurrent connections.

One-to-Many: It is a structure where one input produces multiple outputs. A typical example is **image captioning**, where an image is given as input, and a sentence describing the image is generated as the output.

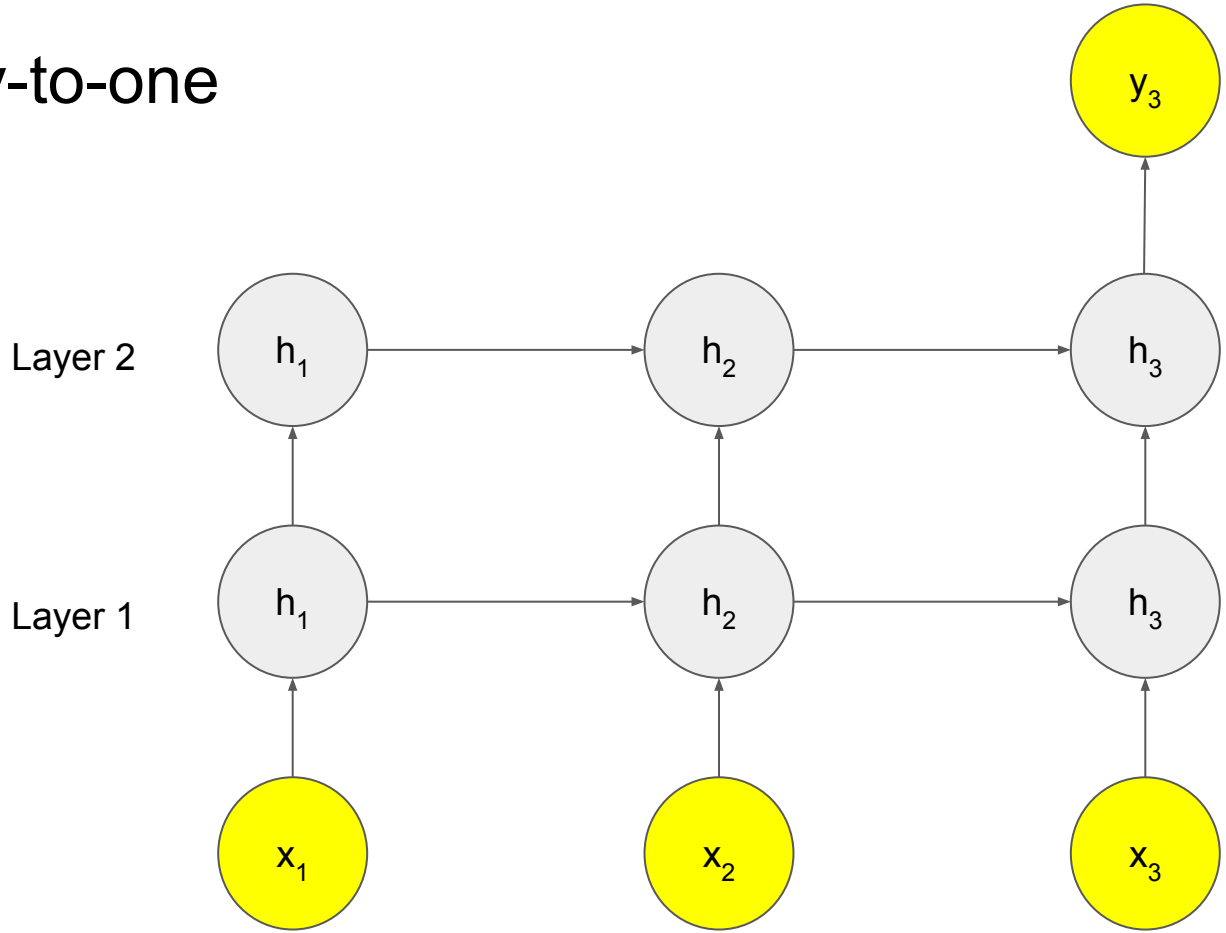
Many-to-One: It is a structure where multiple inputs produce one output. A **sentiment analyzer** is a representative example of this. It takes a sentence as input and outputs the sentiment (positive/negative) of that sentence.

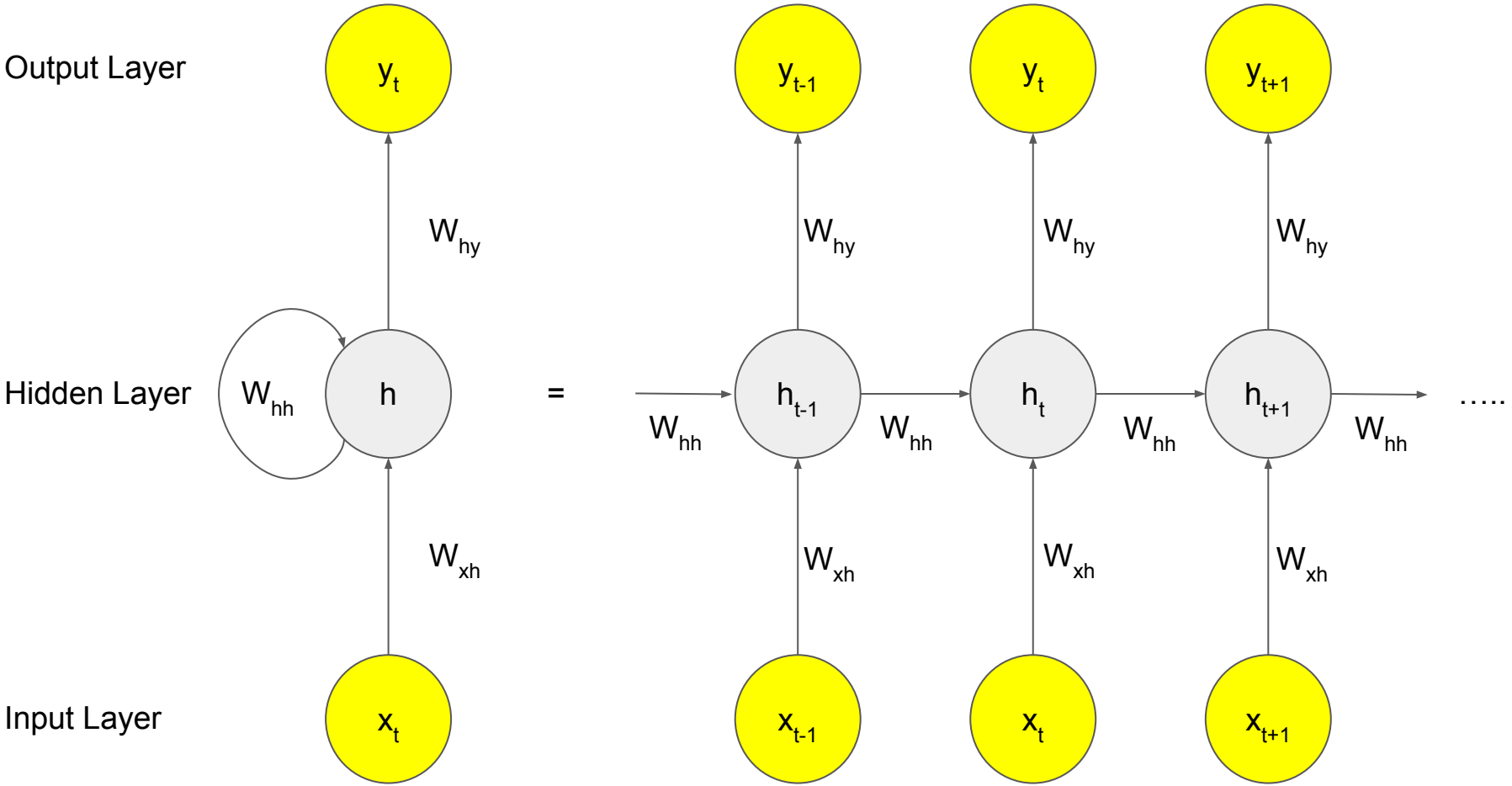
Many-to-Many: It is a structure where multiple inputs produce multiple outputs. The lengths of the input and output sequences can be different. **Machine translation** or speech recognition tasks can be examples of this structure.

Many-to-one



Many-to-one




Output Layer
Hidden Layer
Input Layer
y_t
W_{hy}
W_{hh}
h
W_{xh}
x_t
=
y_{t-1}
y_t
y_{t+1}
h_{t-1}
h_t
h_{t+1}
x_{t-1}
x_t
x_{t+1}
.....

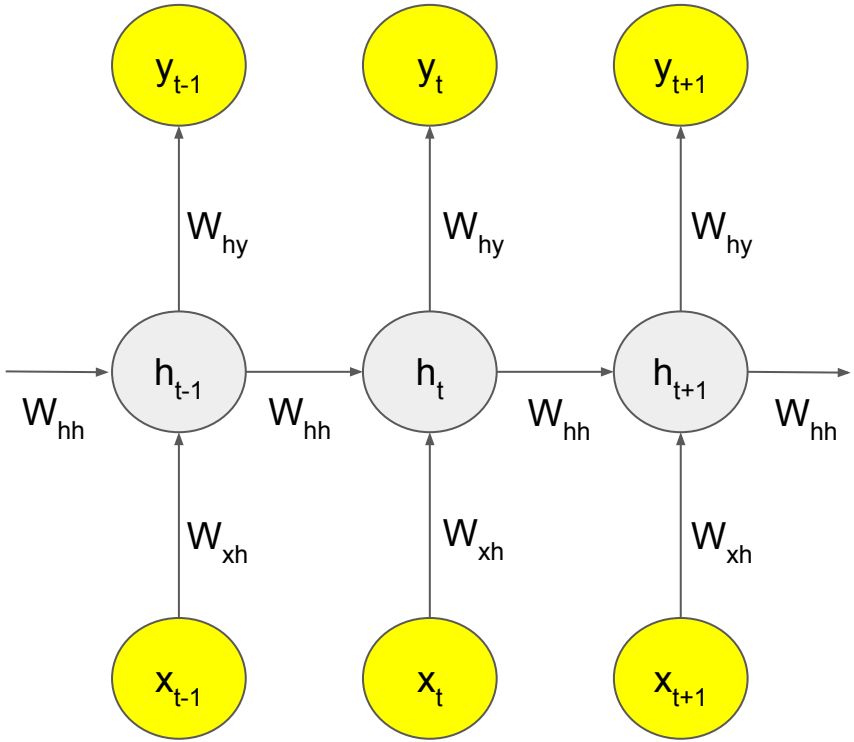
Hidden Layer and Output Layer

- Hidden Layer

$$h_t = tanh(W_{hh} \times h_{t-1} + W_{xh} \times x_t)$$

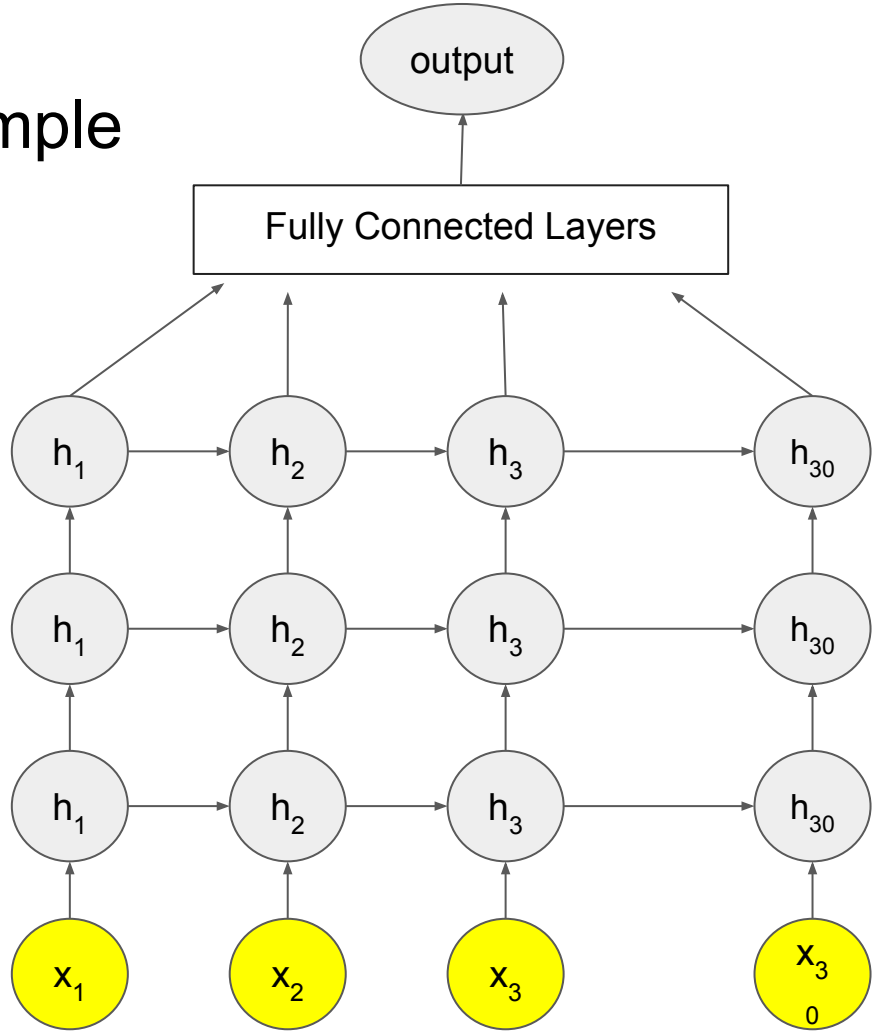
- Output Layer

$$y_t = softmax(W_{hy} \times h_1)$$



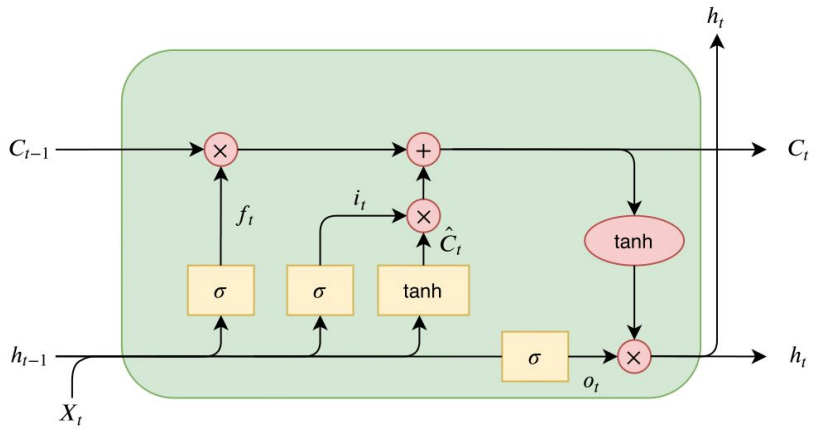
Netflix stock example

5 hidden layers



Long Short-Term Memory (LSTM)

LSTM has added new elements to the hidden layer: the forget gate, the input gate, and the output gate.

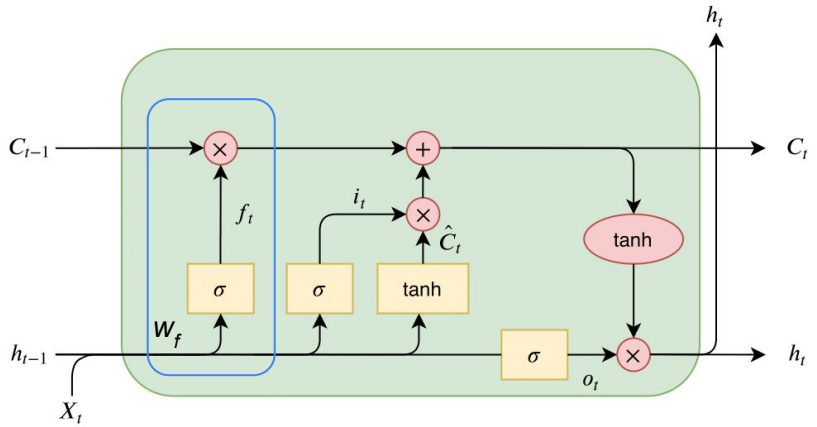


Forget gate

The forget gate in LSTM (Long Short-Term Memory) determines how much of the past information to retain. It takes the past information, i.e., the memory, and the current input data, and after applying the sigmoid function to them, it multiplies the result with the past information. Therefore, if the output of the sigmoid is **0**, the past information is discarded, but if it's **1**, the past information is fully preserved.

$$f_t = sigmoid(w_f\,[h_{t-1}, x_t])$$

$$c_t = f_t \times c_{t-1}$$



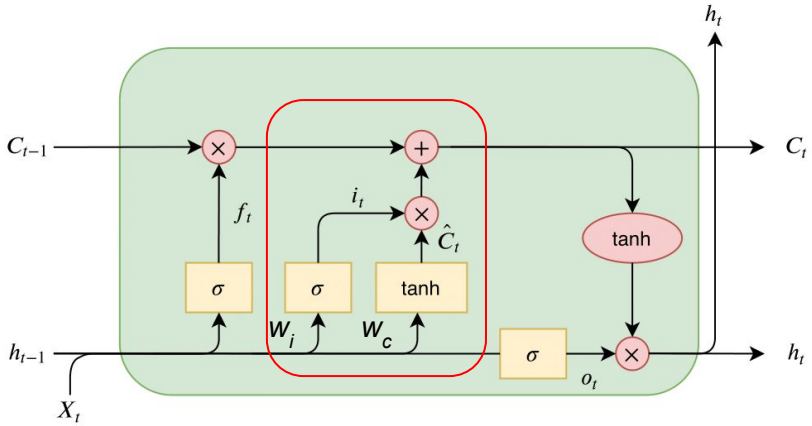
Input gate

The input gate is responsible for preserving the current input information. It uses the sigmoid and tangent functions to determine how much of the current input information should be retained. In other words, it decides how much new information should be added to the memory.

$$i_t = sigmoid(w_i\,[h_{t-1}, x_t])$$

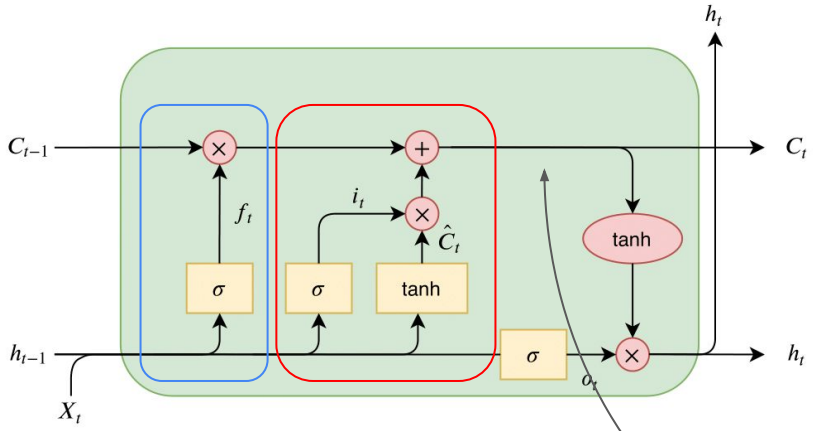
$$c'_t = tanh(w_c\,[h_{t-1}, x_t])$$

$$c_t = c_{t-1} + i_t \times c'_t$$



Cell gate

Update the cell state



$$f_t = sigmoid(w_f\,[h_{t-1}, x_t])$$

$$c_t = f_t \times c_{t-1}$$

$$i_t = sigmoid(w_f\,[h_{t-1}, x_t])$$

$$c'_t = tanh(w_c\,[h_{t-1}, x_t]$$

$$c_t = c_{t-1} + i_t \times c'_t$$

Output gate

Output gate controls memory to output (h_t)

$o_t = sigmoid(w_o\ [h_{t-1}, x_t])$

$h_t = o_t \times tanh(c_{t-1})$