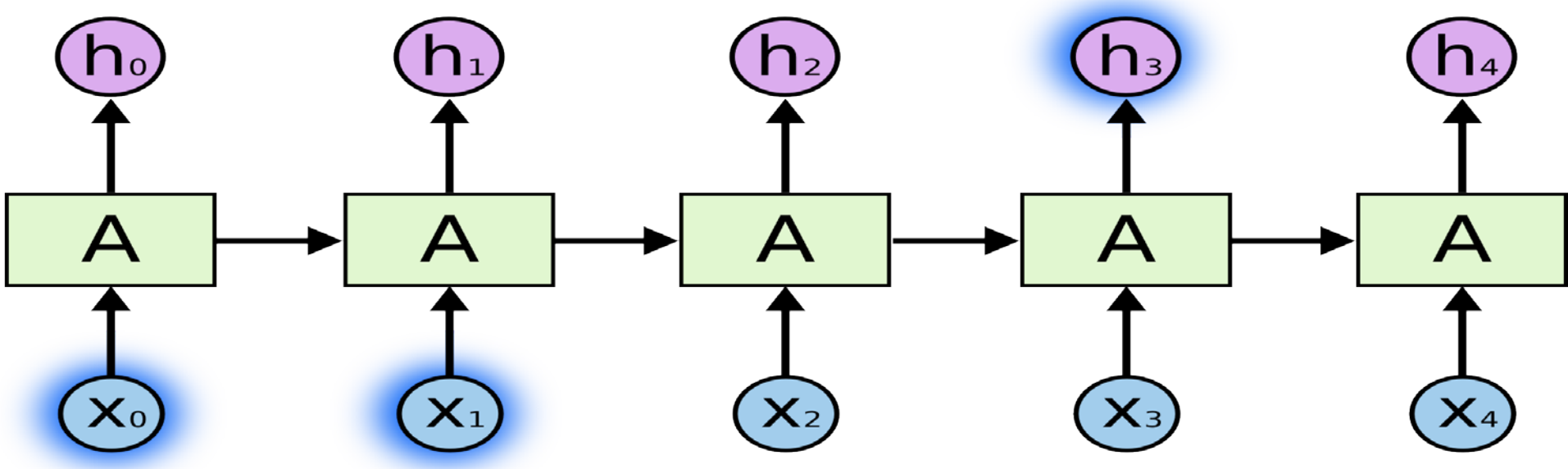
LSTMs



IDENTIFY THE NETWORK

- If we are trying to predict the last word in “the clouds are in the *sky*,” we don’t need any further context

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

In theory, RNNs are absolutely capable of handling such "long-term dependencies."

Sadly, in practice, RNNs don't seem to be able to learn them.

The problem was explored in depth by Hochreiter (1991) and Bengio, et al. (1994), who found some pretty fundamental reasons why it might be difficult.

LONG SHORT-TERM MEMORY

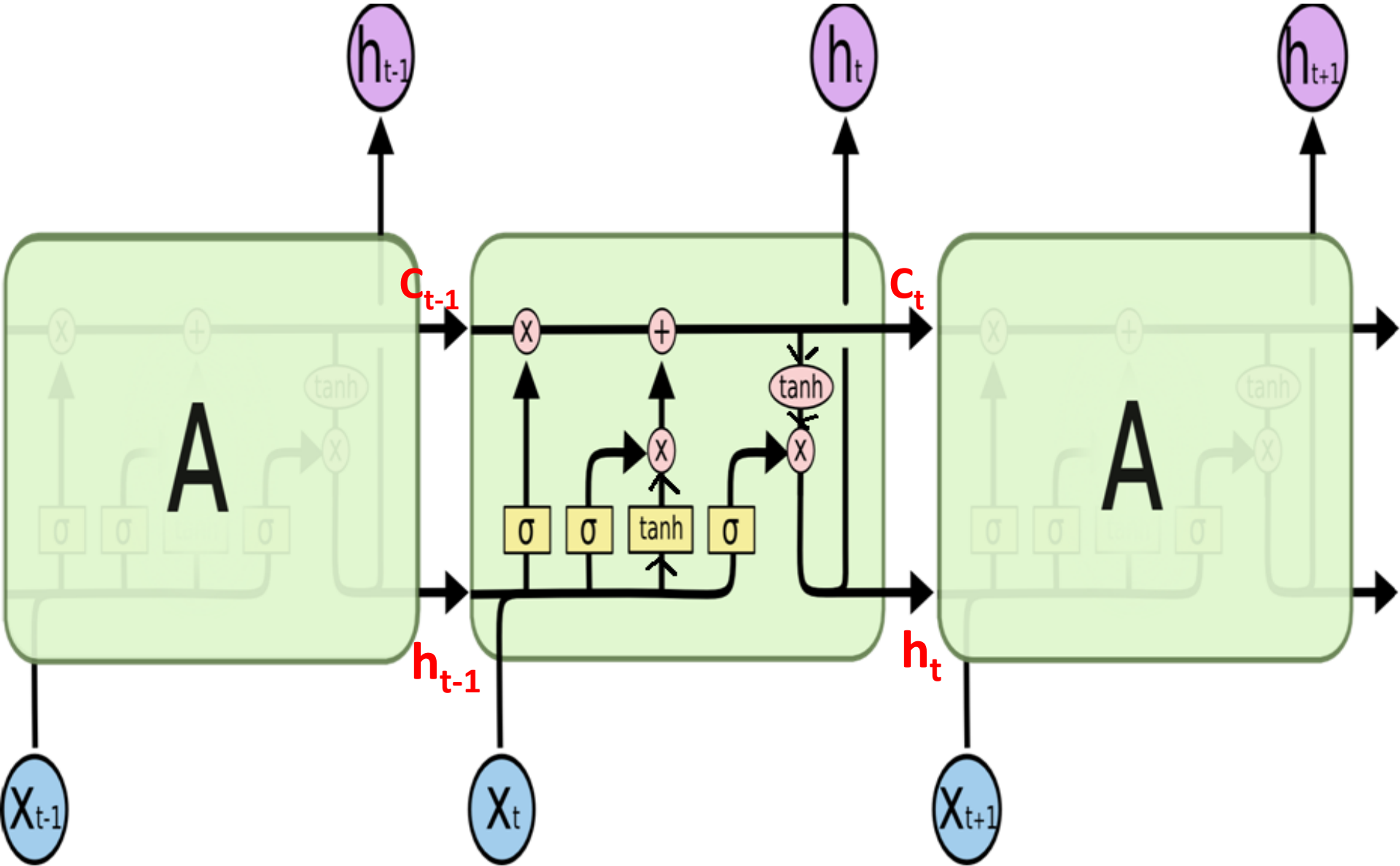
Neural Computation 9(8):1735–1780, 1997

Sepp Hochreiter
Fakultät für Informatik
Technische Universität München
80290 München, Germany
hochreit@informatik.tu-muenchen.de
http://www7.informatik.tu-muenchen.de/~hochreit

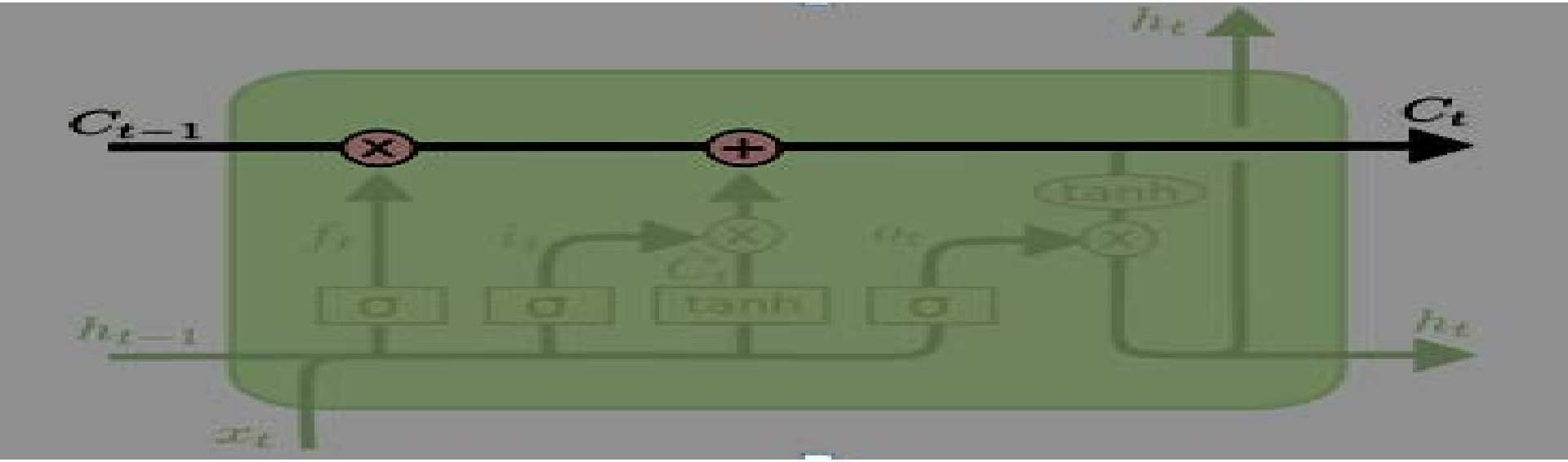
Jürgen Schmidhuber
IDSIA
Corso Elvezia 36
6900 Lugano, Switzerland
juergen@idsia.ch
http://www.idsia.ch/~juergen

https://www.bioinf.jku.at/publications/older/2604.pdf

LSTMs also have chain like structure, but the repeating module has a different structure.



The Core Idea Behind LSTMs is the cell state, the horizontal line running through the top of the diagram.



✓**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 point-wise multiplication operation.**

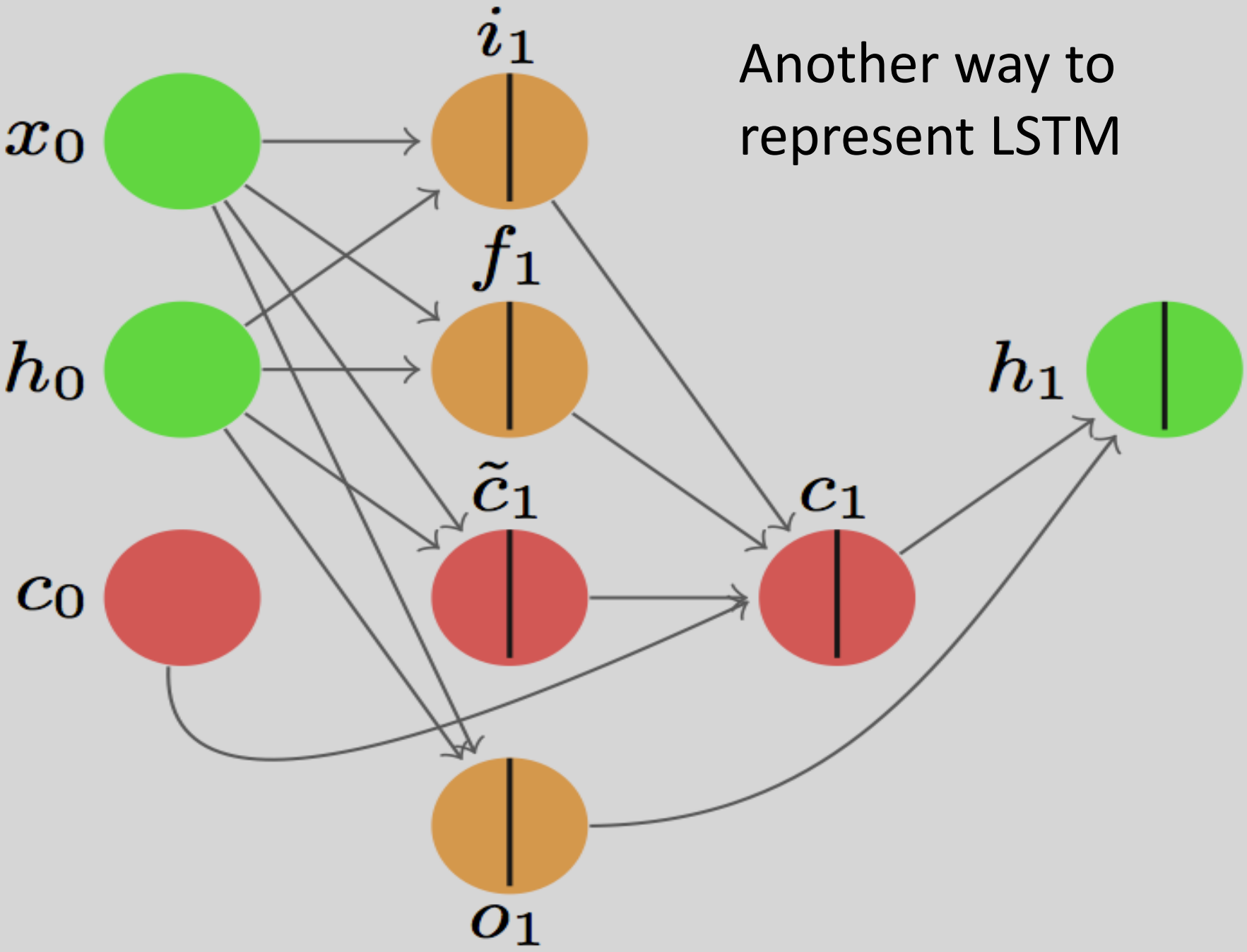
The basic components of LSTM cell

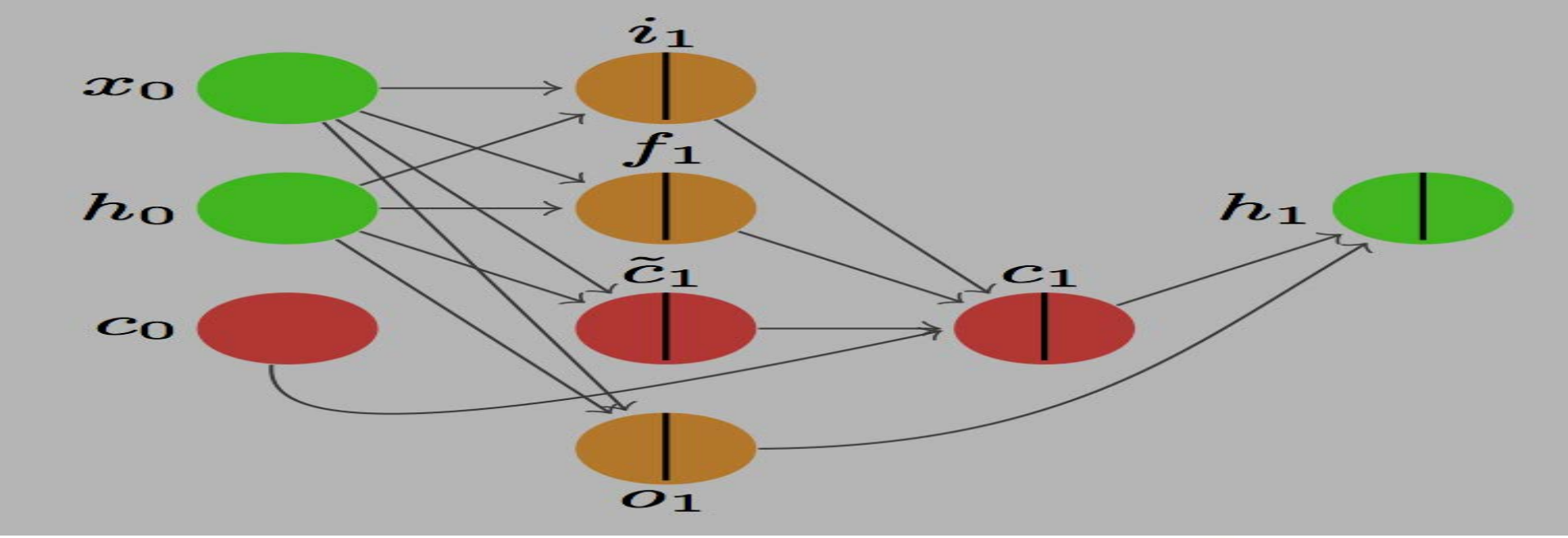
- »an input gate,
- »a forget gate
- »an output gate, and
- »a memory cell

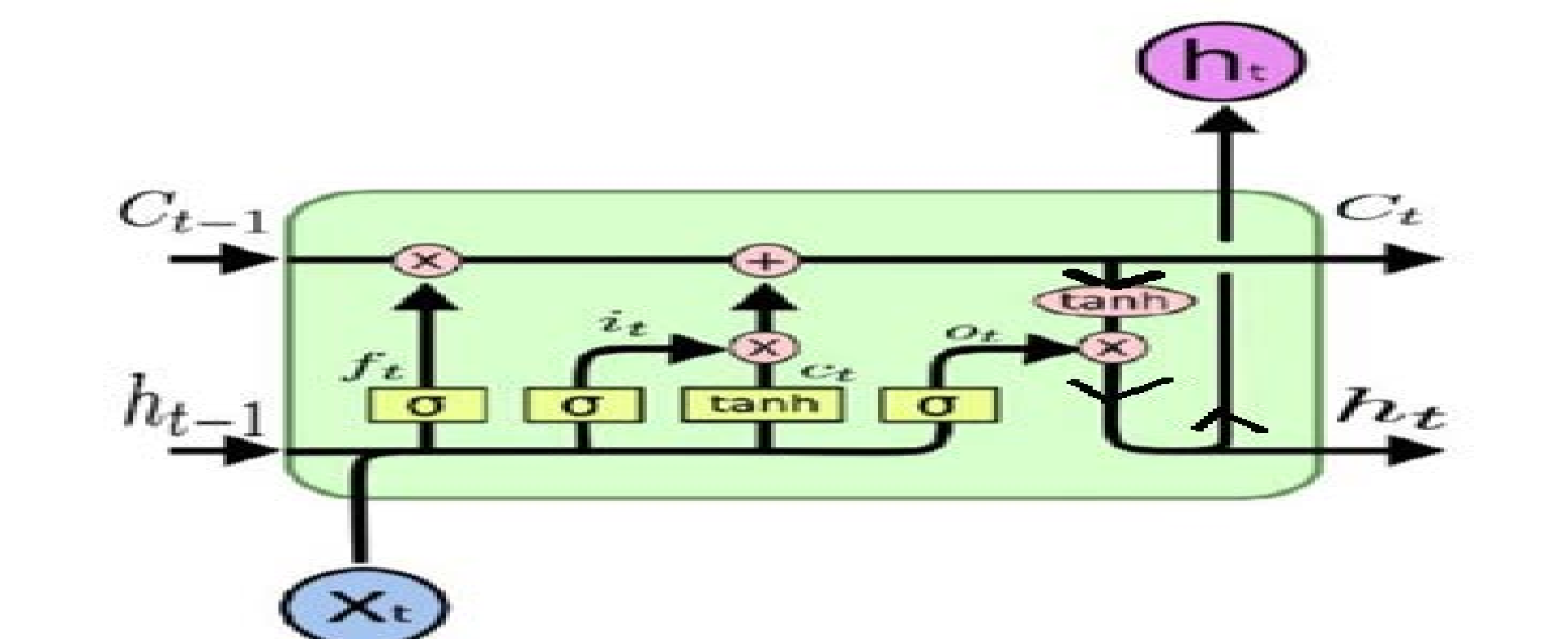
✓**Forget gate** decides what is relevant to keep from prior steps.

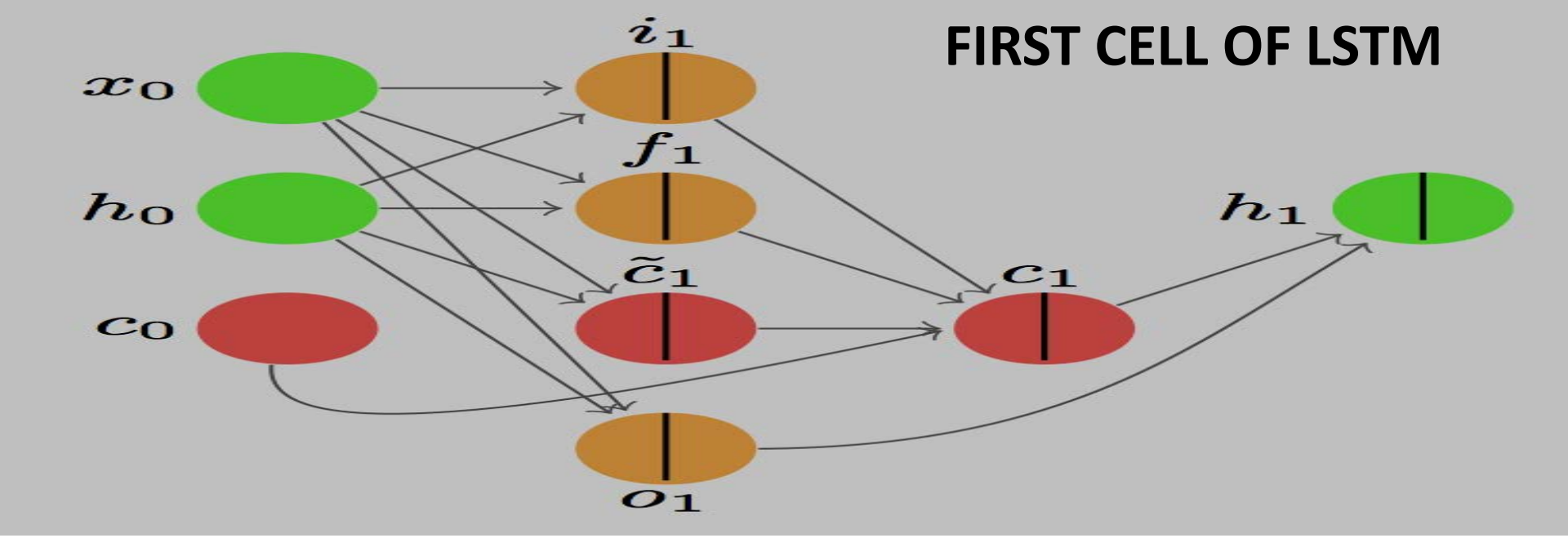
✓**Input gate** decides what information is relevant to add from the current step.

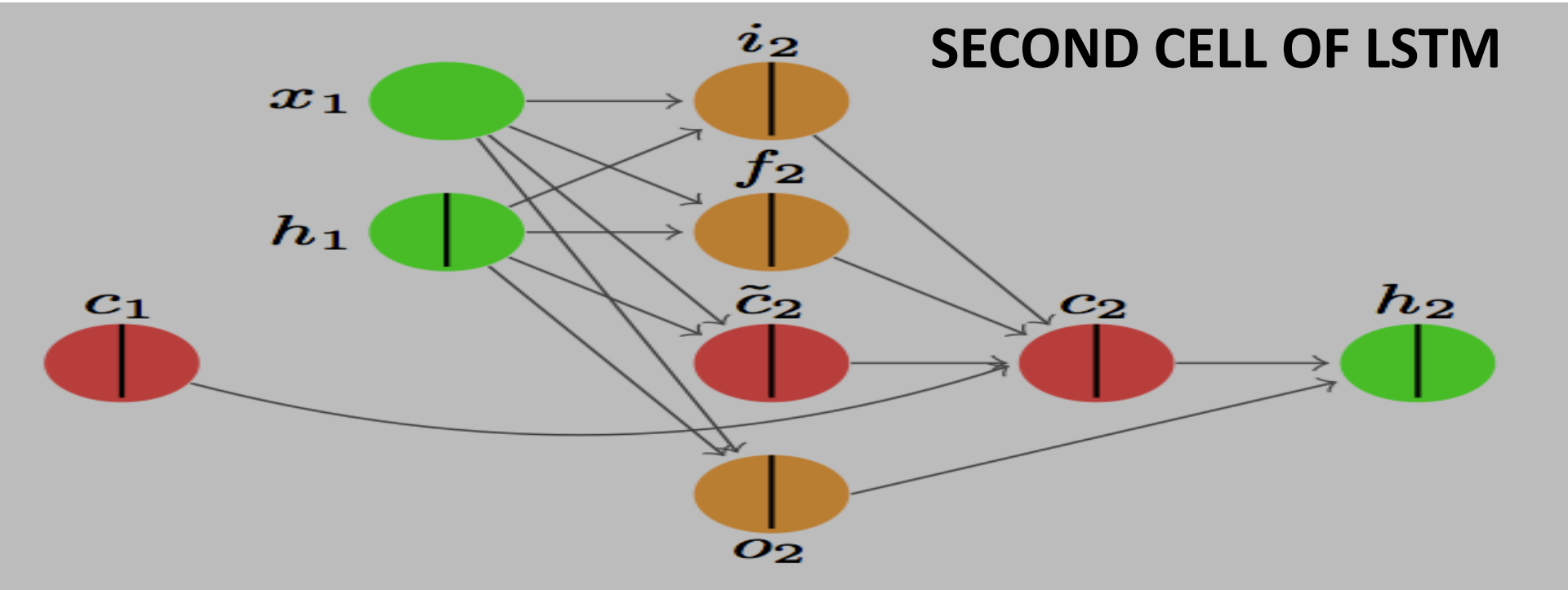
✓**Output gate** determines what the next hidden state should be.

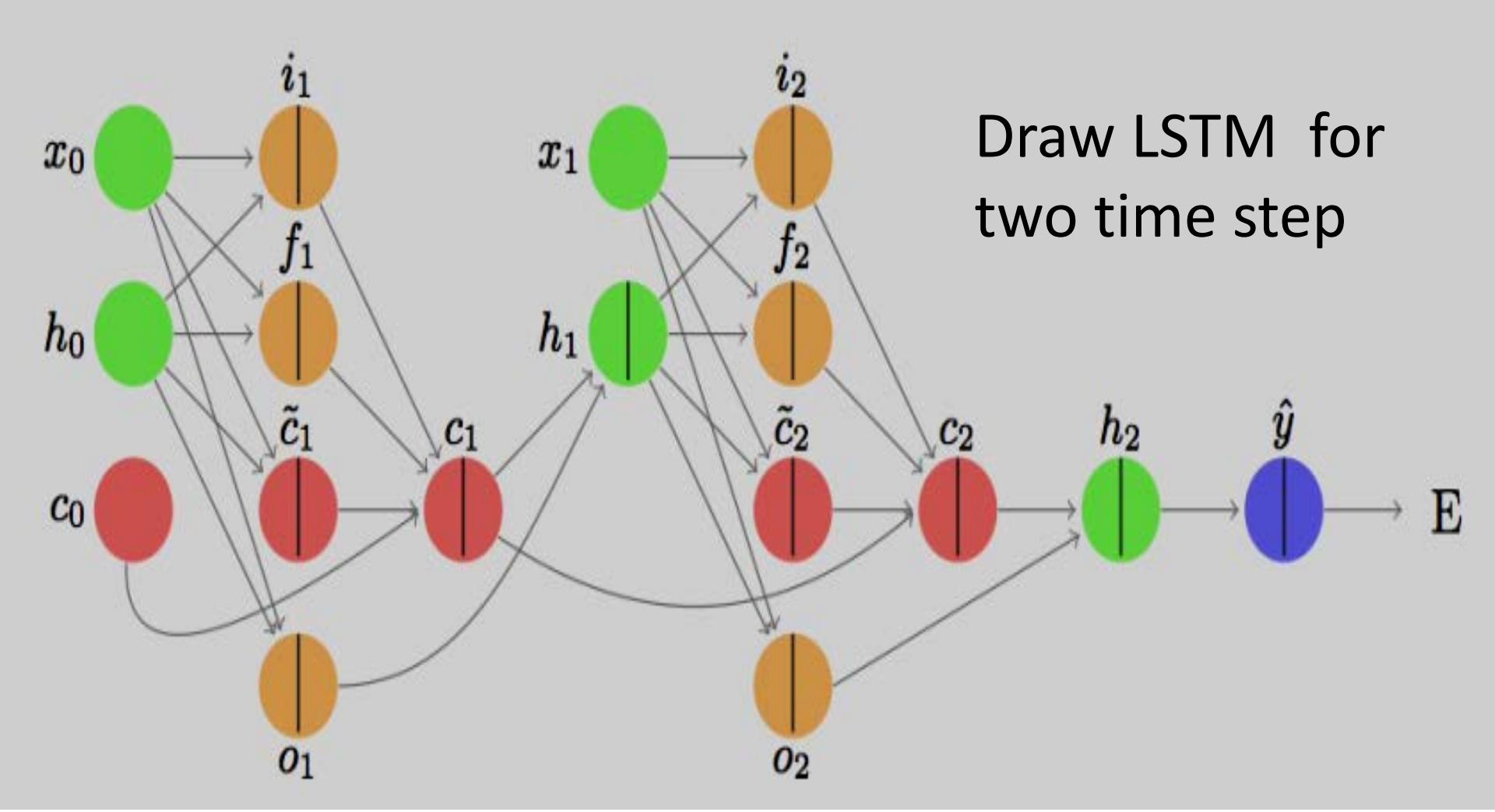

Another way to represent LSTM
i_1
x_0
f_1
h_0
h_1
$\tilde{c}_1$
c_1
c_0
o_1


i_1
x_0
f_1
h_0
h_1
$\tilde{c}_1$
c_1
c_0
o_1


h_t
C_{t-1}
C_t
tanh
i_t
o_t
f_t
c_t
h_{t-1}
h_t
σ
σ
tanh
σ
X_t


FIRST CELL OF LSTM
x_0
h_0
c_0
i_1
f_1
$\tilde{c}_1$
o_1
c_1
h_1


SECOND CELL OF LSTM
x_1
h_1
c_1
i_2
f_2
$\tilde{c}_2$
o_2
c_2
h_2



The hidden state from second cell of LSTM is not input for the next cell because there is no next cell. So, hidden state is used to compute the output of the LSTM network.

- The input gate allows new information to flow into the network. It has parameters W_i , b_i , where 'i' stands for input.

- The memory cell preserves the hidden units information across time steps. It has parameters W_c , b_c, where 'c' stands for cell.

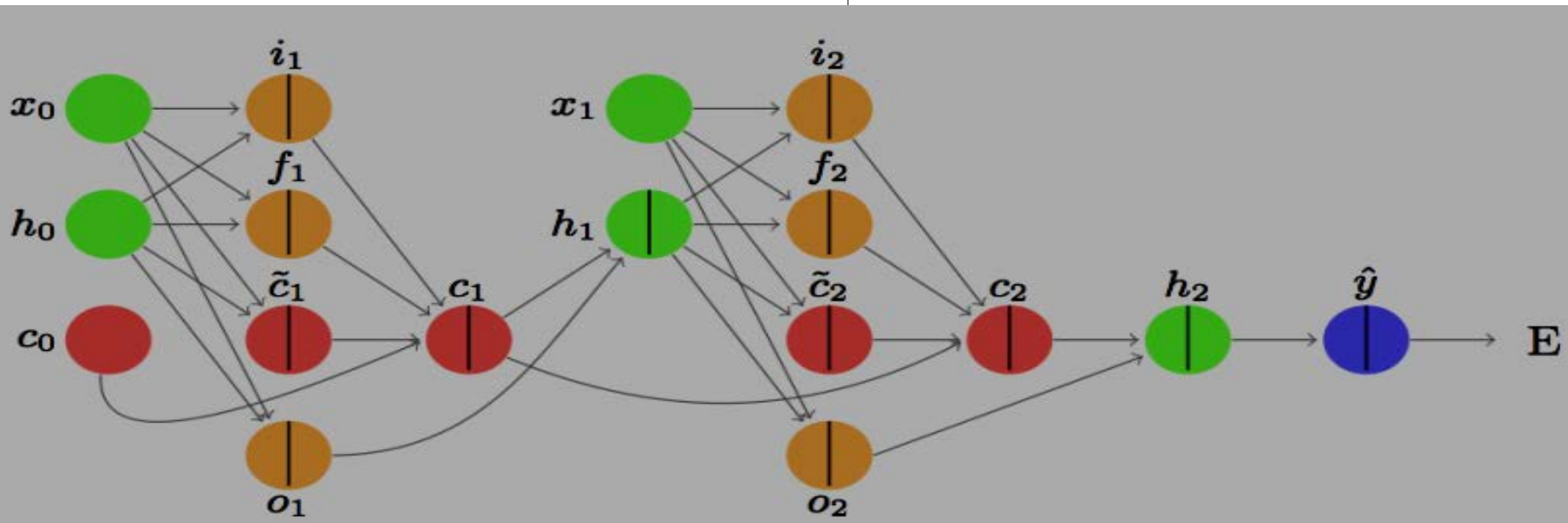
- The forget gate allows information which is no longer pertinent to be discarded. It has parameters W_f , b_f , where 'f' stands for forget.

- The output gate allows what information will be output to the screen/print and what will be propagated forward as part of the new hidden state. It has parameters W_o , b_o, where 'o' stands for output.

- Interestingly, all of the weights have the same dimension.

Example: Input at beginning of the sequence is 0.1, and input at next time step is 0.2. $x_0 = 0.1$, $x_1 = 0.2$[Two cell LSTM] Initial weights and biases are

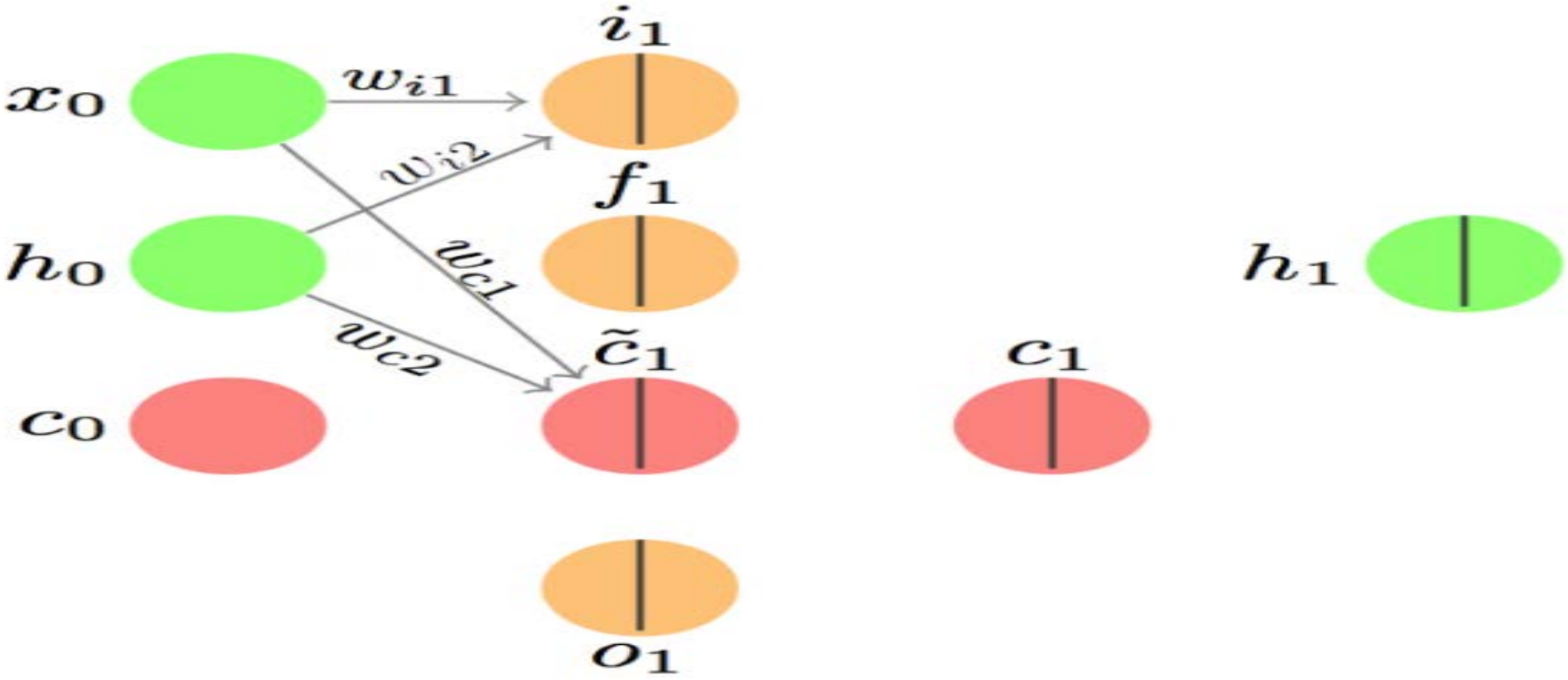
$$W = \begin{bmatrix} w_{i1} & w_{i2} & b_i \\ w_{c1} & w_{c2} & b_c \\ w_{f1} & w_{f2} & b_f \\ w_{o1} & w_{o2} & b_o \\ w_y & 0 & b_y \end{bmatrix} = \begin{bmatrix} 0.5 & 0.25 & 0.01 \\ 0.3 & 0.4 & 0.05 \\ 0.03 & 0.06 & 0.002 \\ 0.02 & 0.04 & 0.001 \\ 0.6 & 0 & 0.025 \end{bmatrix}$$



Example: Input at beginning of the sequence is 0.1, and input at next time step is 0.2. $x_0 = 0.1$, $x_1 = 0.2$[Two cell LSTM] Initial weights and biases are

$$W = \begin{bmatrix} w_{i1} & w_{i2} & b_i \\ w_{c1} & w_{c2} & b_c \\ w_{f1} & w_{f2} & b_f \\ w_{o1} & w_{o2} & b_o \\ w_y & 0 & b_y \end{bmatrix} = \begin{bmatrix} 0.5 & 0.25 & 0.01 \\ 0.3 & 0.4 & 0.05 \\ 0.03 & 0.06 & 0.002 \\ 0.02 & 0.04 & 0.001 \\ 0.6 & 0 & 0.025 \end{bmatrix}$$

If two time step input is two dimensional, what will be the size of intial weight matrix ??



$$net_{i1} = w_{i1}x_0 + w_{i2}h_0 + b_i$$

$$net_{i1} = 0.5(0.1) + 0.25(0) + 0.01 = 0.06$$

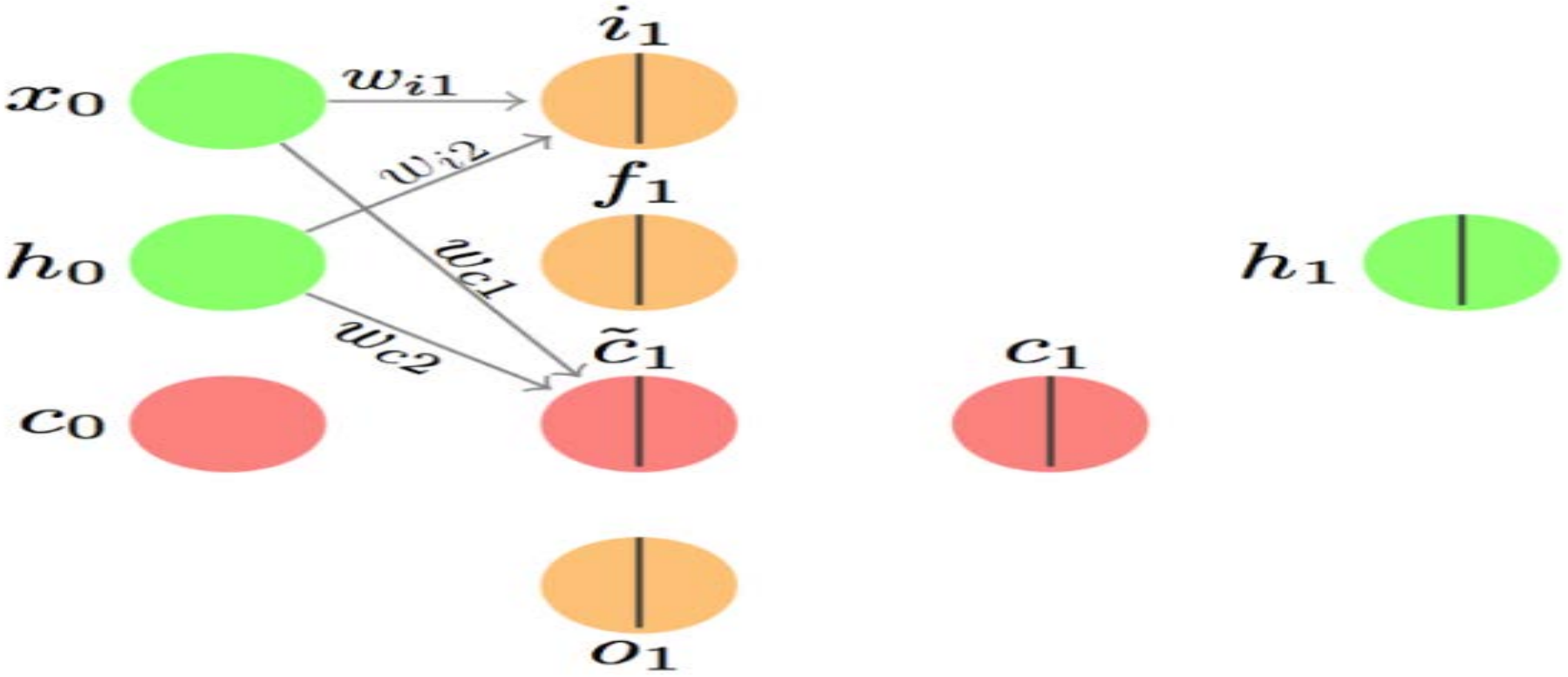
$$i_1 = \frac{1}{1+\exp(-net_{i1})} = \frac{1}{1+\exp(-0.06)} = 0.515$$

$$net_{i1} = 0.06; \qquad i_1 = \frac{1}{1+\exp(-net_{i1})} = \frac{1}{1+\exp(-0.06)} = 0.515$$

This value can be interpreted as the probability that we will allow information from input x to enter the memory cell.

Usual practice is to keep the value 0.515 (keep gate partially open)

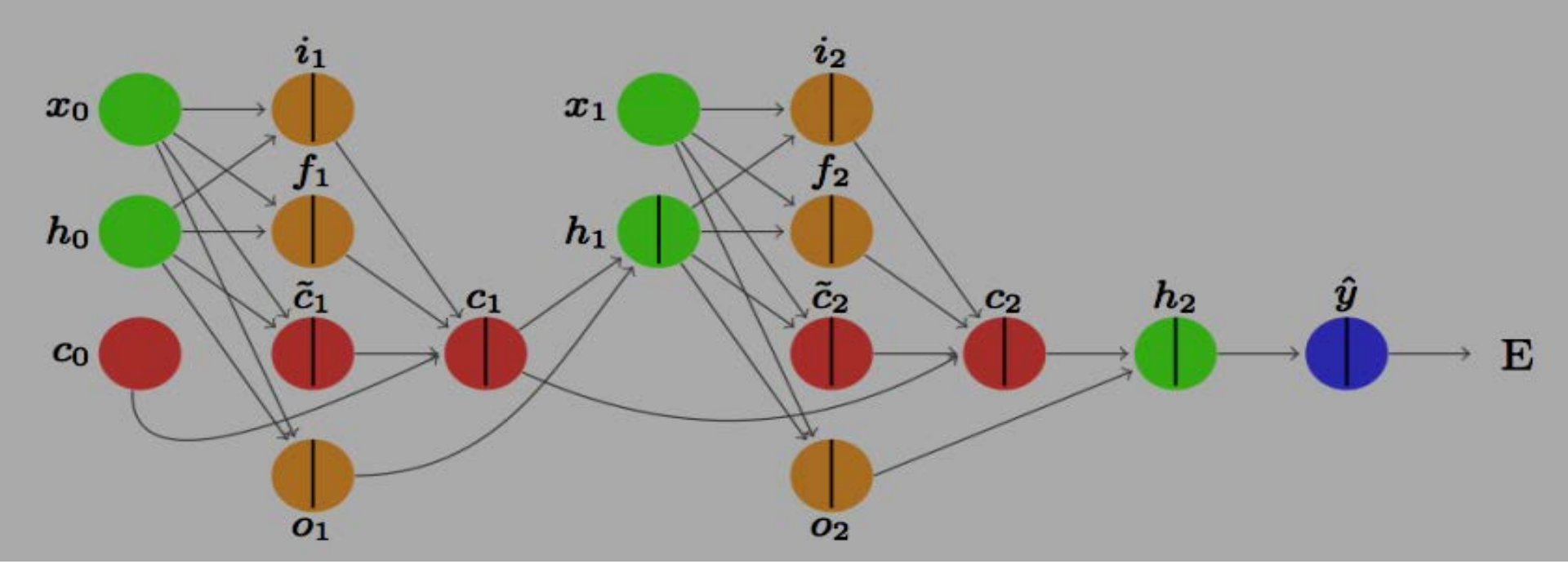
Alternatively, decision is made whether the information will go forward , i.e. open the gate (value 1) of close the gate(value 0)



$$net_{\tilde{c1}} = w_{c1}x_0 + w_{c2}h_0 + b_c$$

$$net_{\tilde{c}1} = 0.3(0.1) + 0.4(0) + 0.05 = 0.08$$

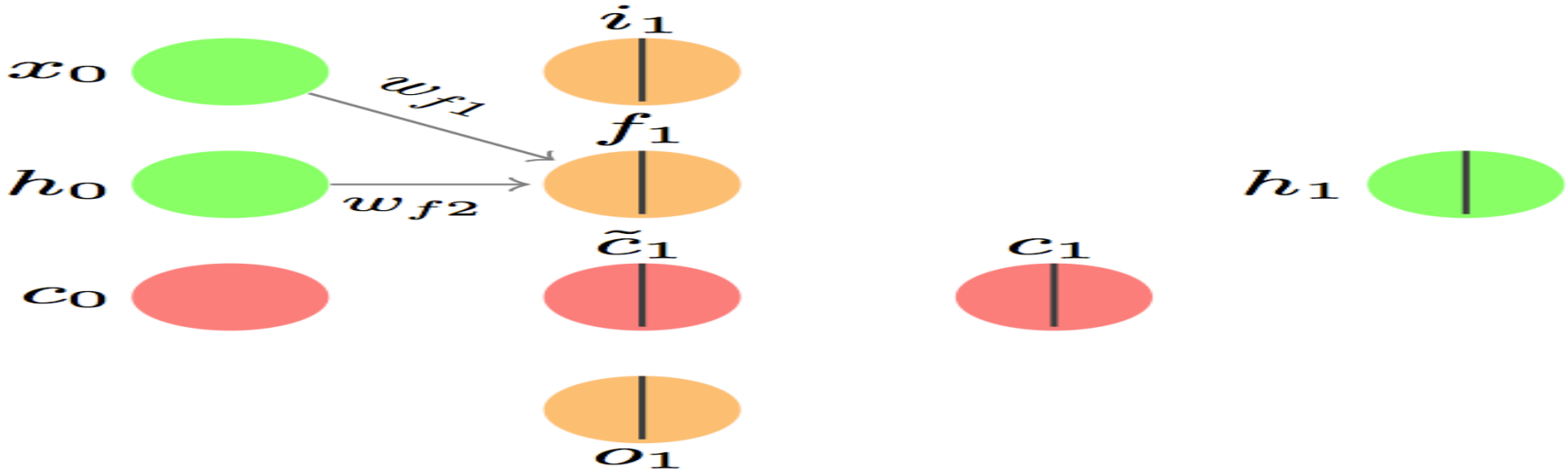
$$\tilde{c}_1 = \tanh(net_{c1}) = 0.0798$$



$$\tilde{c}_1 = \tanh(net_{c1}) = 0.0798$$

Note no stochastic decision is made here – this is the quantity associated with the input that we'll pass to the memory cell.

We'll use these pieces together later when we update the memory cell.



$$net_{f1} = w_{f1}x_0 + w_{f2}h_0 + b_f$$

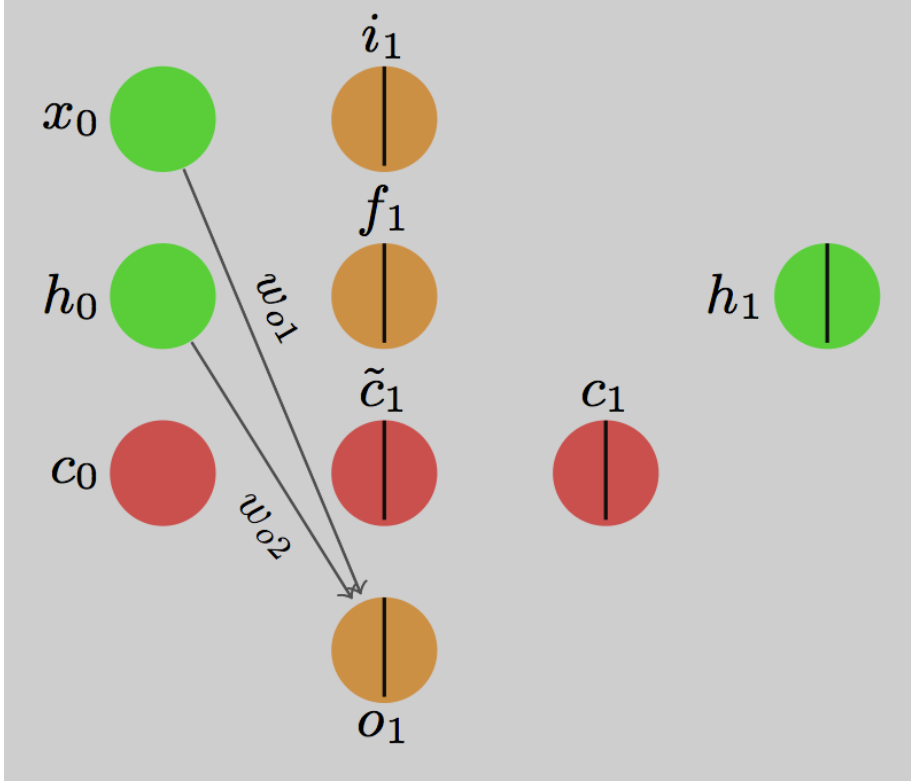
$$net_{f1} = 0.03(0.1) + 0.06(0) + 0.002 = 0.005$$

$$f_1 = \frac{1}{1+\exp(-net_{f1})} = \frac{1}{1+\exp(-0.005)} = 0.5012$$

Again, a stochastic decision could be made here as to whether the previous information should be forgotten(value 0) or allowed through(value 1) or kept as it is.

Lets assume for this example the value is 1

OUTPUT GATE



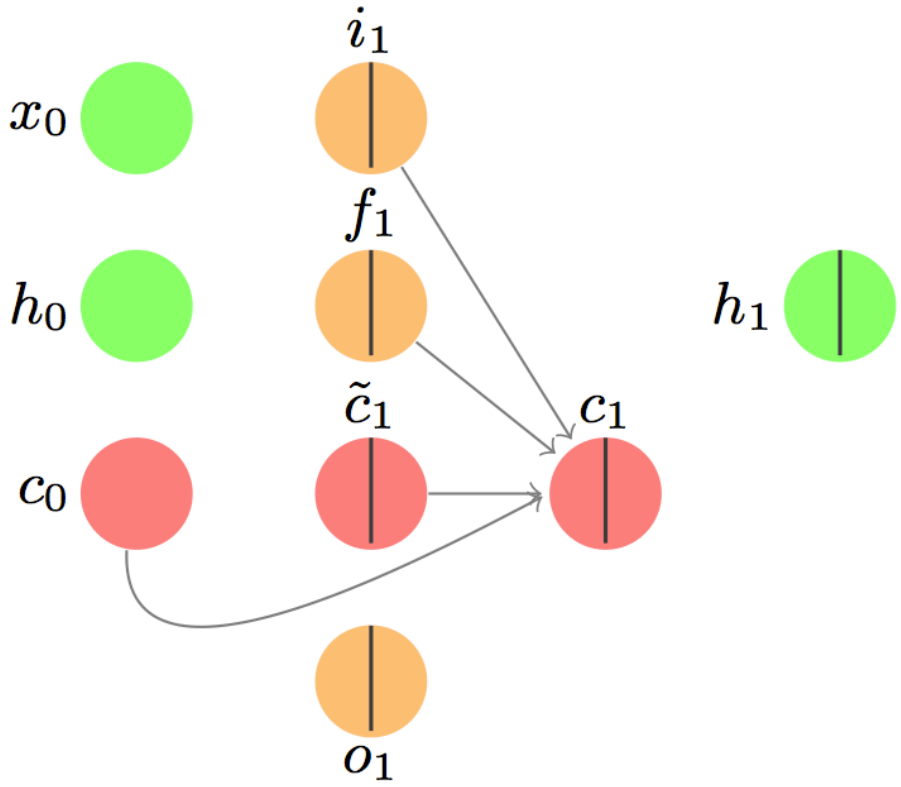
$$net_{o1} = w_{o1}x_0 + w_{o2}h_0 + b_0$$

$$net_{o1} = 0.02(0.1) + 0.04(0) + 0.001 = 0.003$$

$$o_1 = \frac{1}{1+\exp(-net_{01})} = \frac{1}{1+\exp(-0.003)} = 0.5007$$

Again we make a stochastic decision as to whether we pass this output along. Lets assume the stochastic decision results in 1.

MEMORY CELL



MemoryCell

$$c_1 = i_1 \circ \tilde{c}_1 + f_1 \circ c_0$$

$\circ$: *HADAMARD / NOT*

MATRIX – PRODUCT

$$vector \quad a_1 = [1,2,3], b_1 = [9,10,11]$$

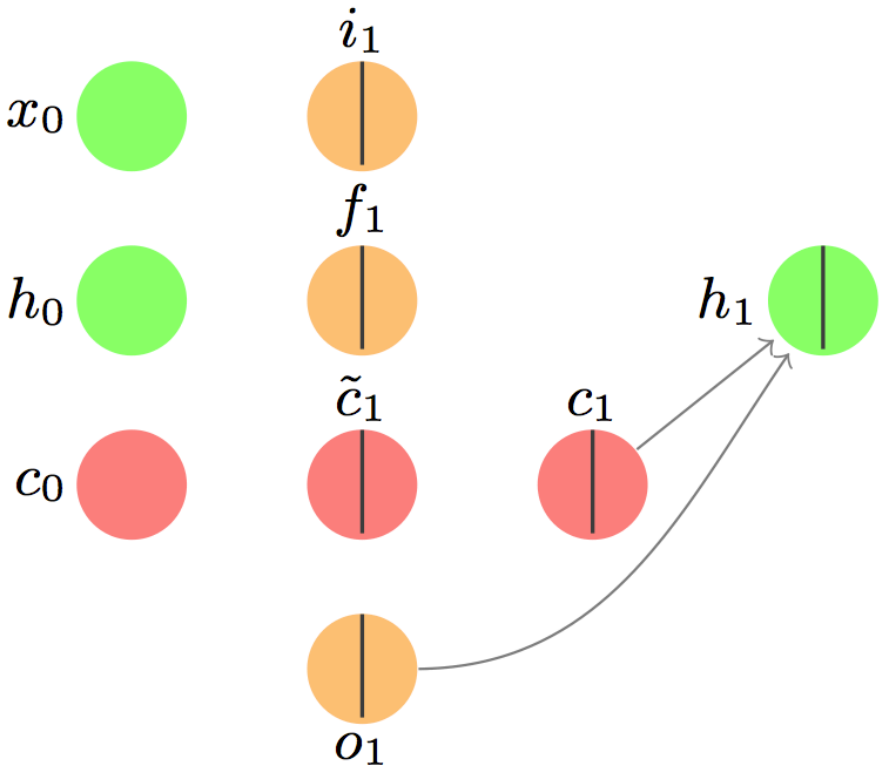
$$p_1 = a_1 \circ b_1 = [(1)(9),(2)(10),(3)(11)]$$

$$= [9,20,33]$$

$$c_1 = i_1 \circ \tilde{c}_1 + f_1 \circ c_0$$

$$c_1 = 1 \circ 0.0798 + 1 \circ 0 = 0.0798$$

Now that we have updated the memory state (another name for memory cell), lets think what we want to output



HIDDEN STATE:

The hidden layer is separate from the memory cell, but very related.

Think of it as part of memory cell that we want to ensure persists.

$$h_1 = o_1 \circ \tanh(c_1)$$

$$h_1 = 1 \circ \tanh(0.0798) = 1 \circ 0.0796 = 0.0796$$

The output gate decides whether the signal from the memory cell gets sent forward as part of the input to next LSTM cell

The Second LSTM cell :Assume Weights are shared across

LSTM cells

$$net_{i2} = w_{i1}x_1 + w_{i2}h_1 + b_i$$

$$i_2 = \frac{1}{1+\exp(-net_{i2})}$$

$$net_{c2} = w_{c1}x_1 + w_{c2}h_1 + b_c$$

$$\tilde{c}_2 = \tanh(net_{c2})$$

$$net_{f2} = w_{f1}x_1 + w_{f2}h_1 + b_f$$

$$f_2 = \frac{1}{1+\exp(-net_{f2})}$$

$$net_{o2} = w_{o1}x_1 + w_{o2}h_1 + b_0$$

$$o_2 = \frac{1}{1+\exp(-net_{02})}$$

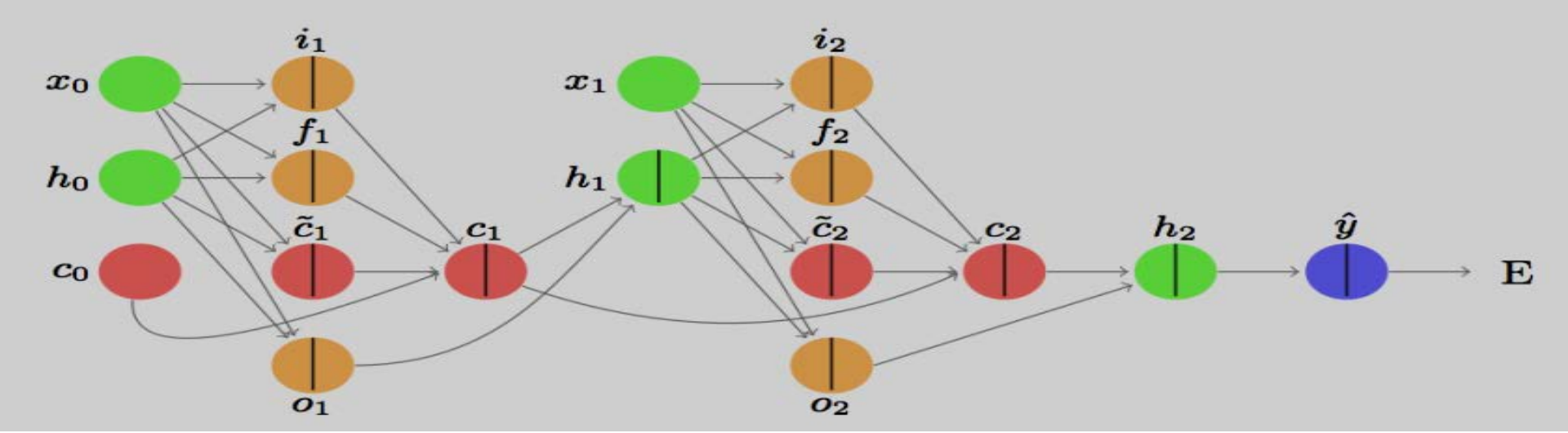
memoryCell

$$c_2 = i_2 \circ \tilde{c}_2 + f_2 \circ c_1$$

Hidden state

$$h_2 = o_2 \circ \tanh(c_2)$$

Final output, to use for the error calculation and apply BPA ??



This depends exclusively on the hidden state (remember the memory cell is input to the hidden state, so we only need to use the hidden state to take into account the entire memory of our LSTM).

$$\hat{y} = w_y h_2 + b_y$$

$$Error = E = \frac{1}{2}(y_d - \hat{y})^2$$

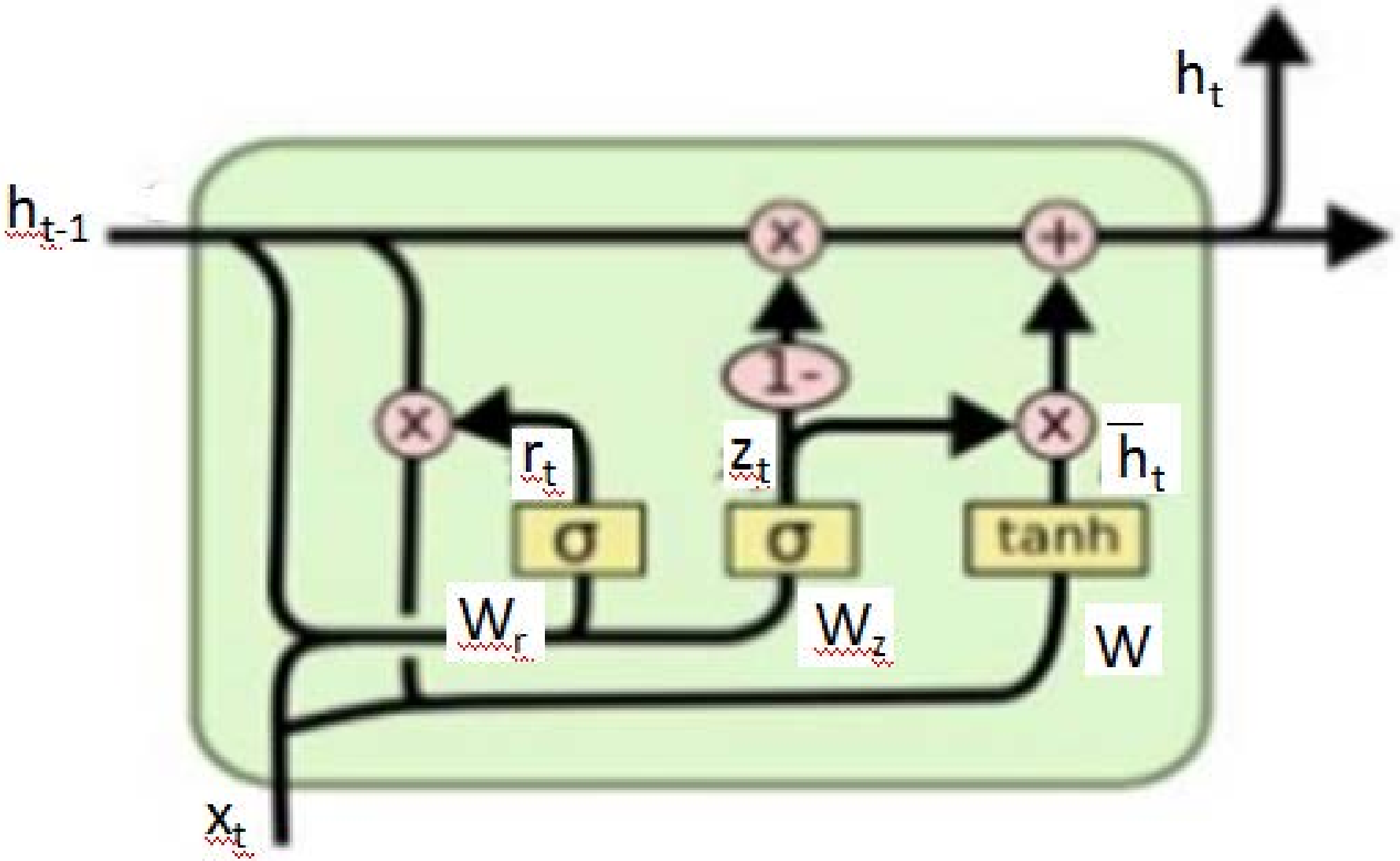
The LSTM followed by the Gated Recurrent Unit (GRU) having the same goal of tracking long-term dependencies

The GRU operates using

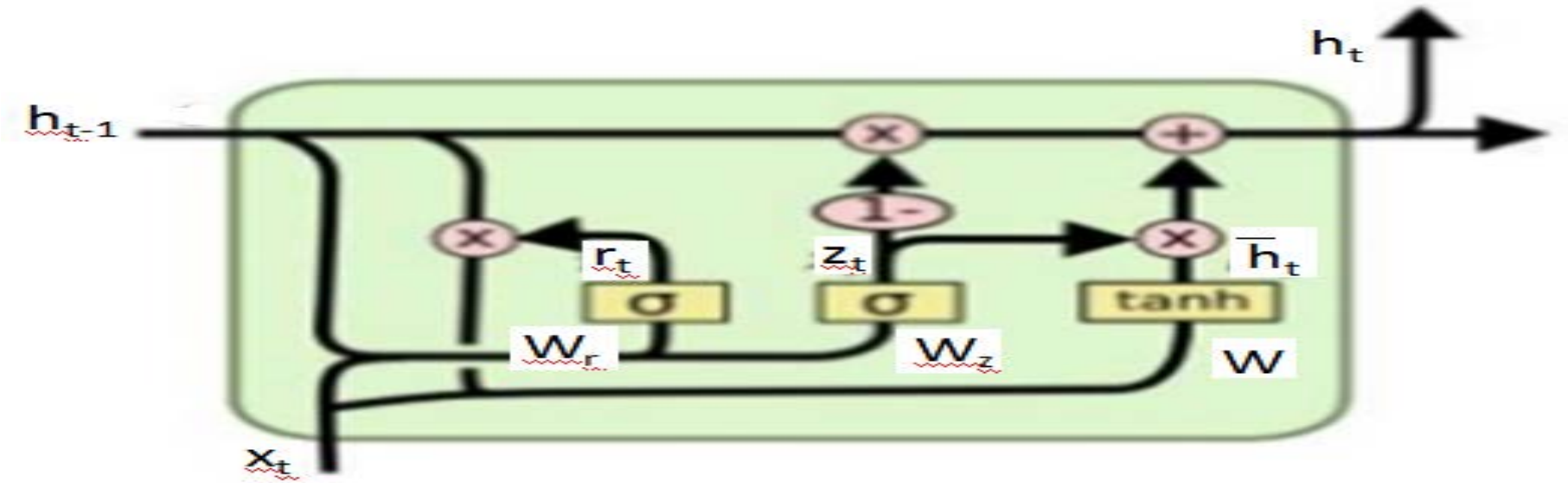
A RESET gate, how much past information to forget

An UPDATE gate (Combination of forget and input Gate) what information to throw away and what new information to add.

✓Merges the Cell state and Hidden state


h_t
h_{t-1}
×
+
1-
×
×
r_t
z_t
$\tilde{h}_t$
σ
σ
tanh
W_r
W_z
W
x_t

Expression for $z_t, r_t, \tilde{h}_t, h_t$



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

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

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

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

✓**Transformer models are essentially attention based models.**

✓**They see the entire sentence as a whole, unlike LSTMs (or in general RNNs) where the sentence is processed sequentially - one word per time step.**

✓ **During training LSTMs need to propagate the error back in time through words one word at a time.**

✓**Transformer sees all words simultaneously - so there is no backpropagation through time.**