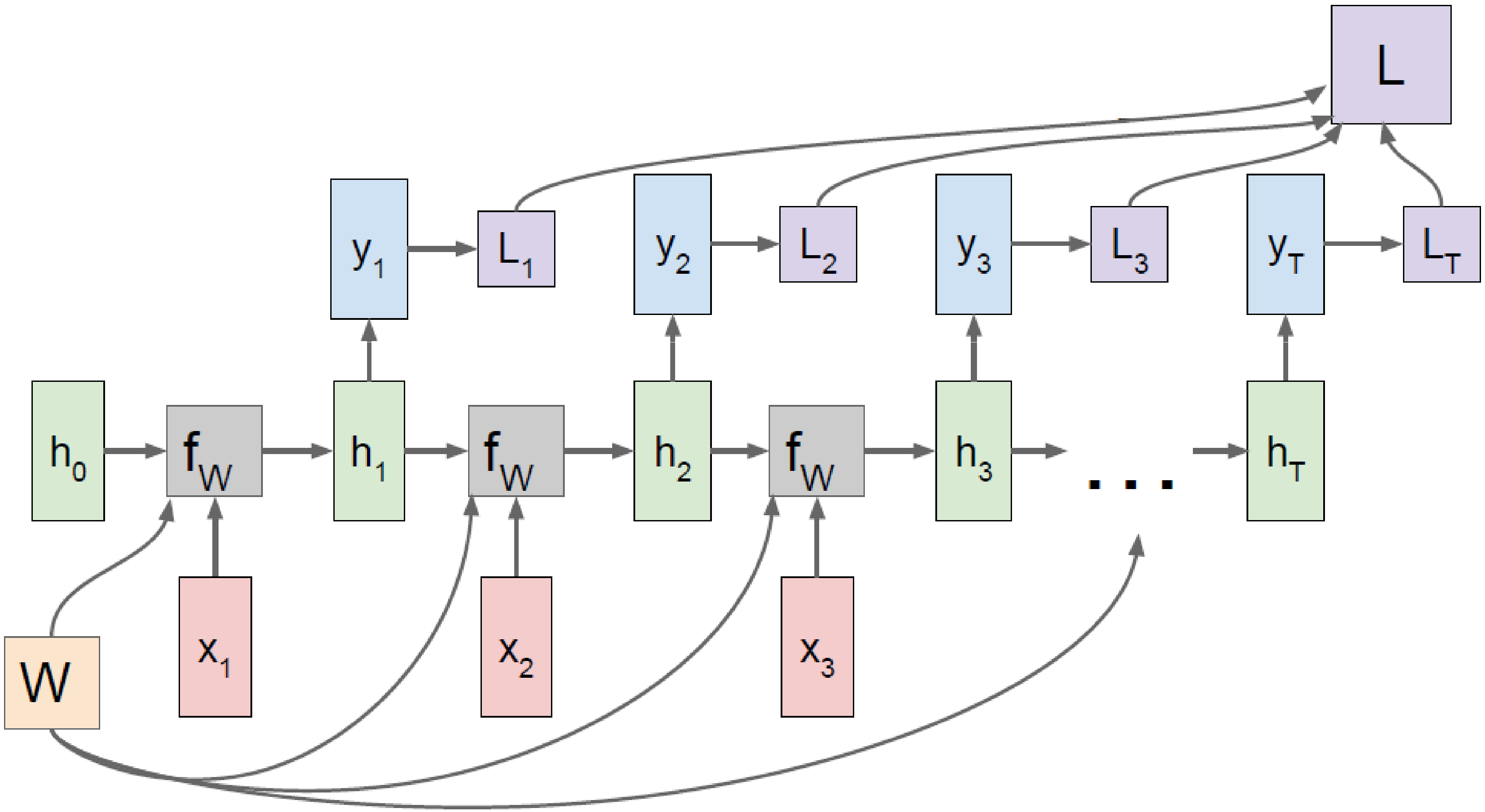
Gated RNN and LSTM

Continue…

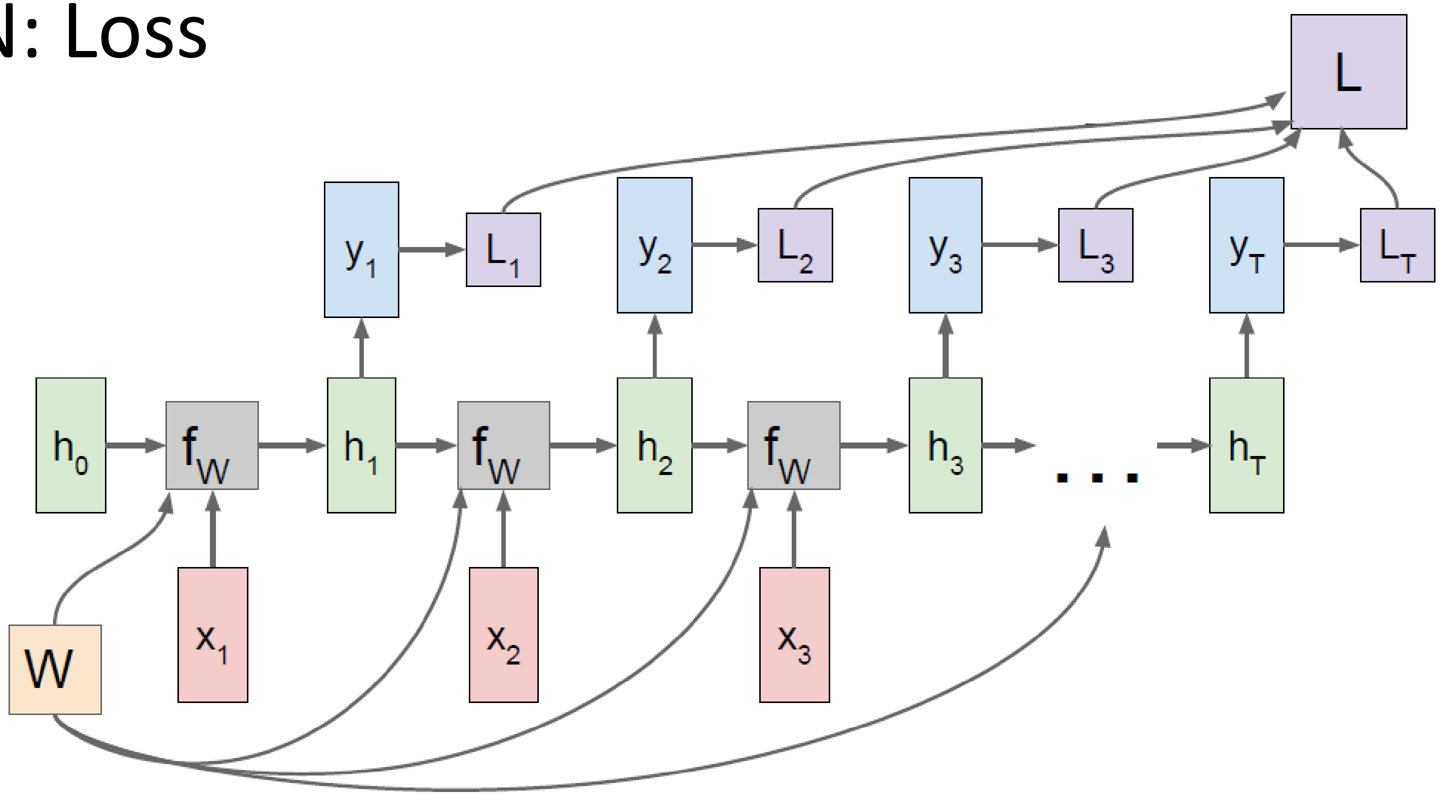
Some slides were adated/taken from various sources, including Andrew Ng’s Coursera Lectures, CS231n: Convolutional Neural Networks for Visual Recognition lectures, Stanford University CS Waterloo Canada lectures, Aykut Erdem, et.al. tutorial on Deep Learning in Computer Vision, Ismini Lourentzou's lecture slide on "Introduction to Deep Learning", Ramprasaath's lecture slides, and many more. We thankfully acknowledge them. Students are requested to use this material for their study only and NOT to distribute it.

RNN: Computational Graph: Many to Many



Slide Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

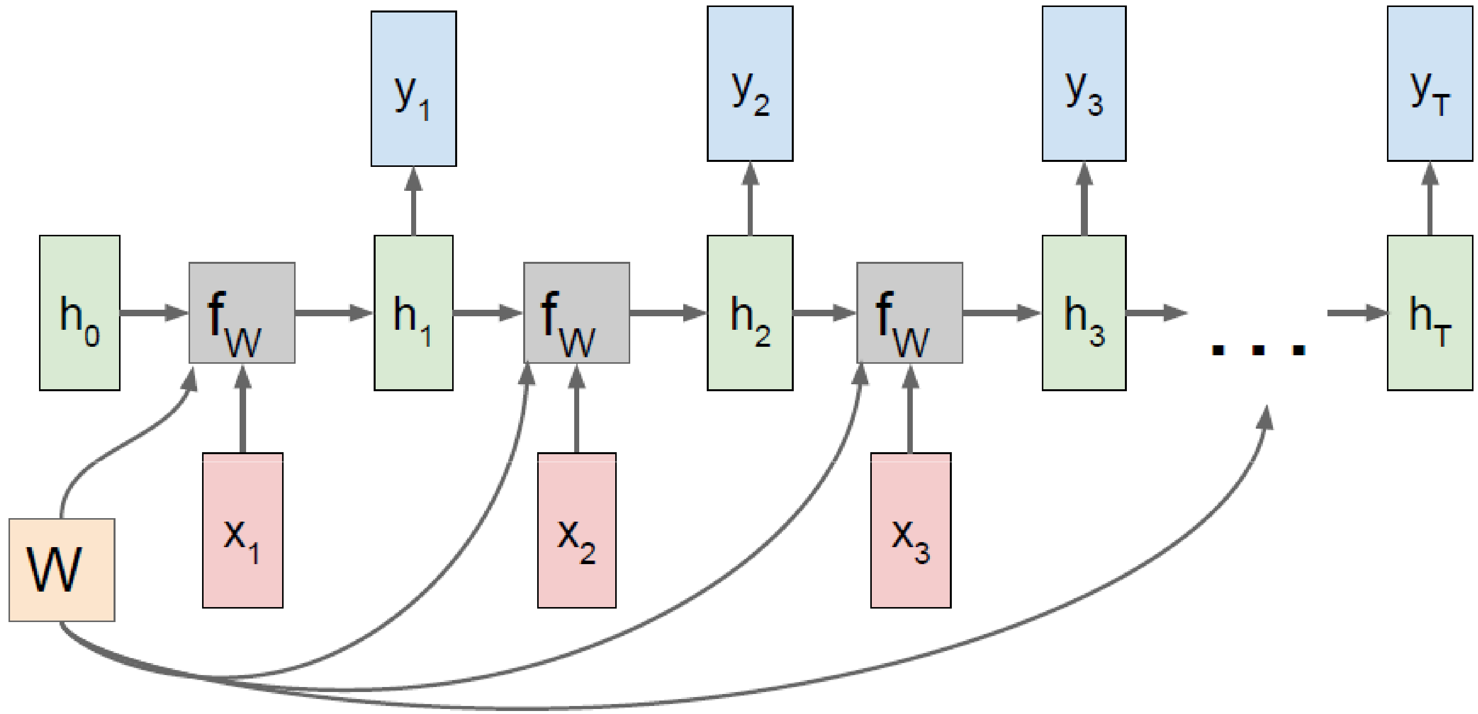
RNN: Loss



$$L^t\left(y^t, x^t\right) = -x^t \log(y^{(t)}) - \left(1 - x^{(t)}\right)\log(1 - y^t)$$

$$L = L(Y, X) = \sum_{t=1}^{T} L^t\left(y^t, x^t\right)$$

RNN: Vanishing Gradient Problem

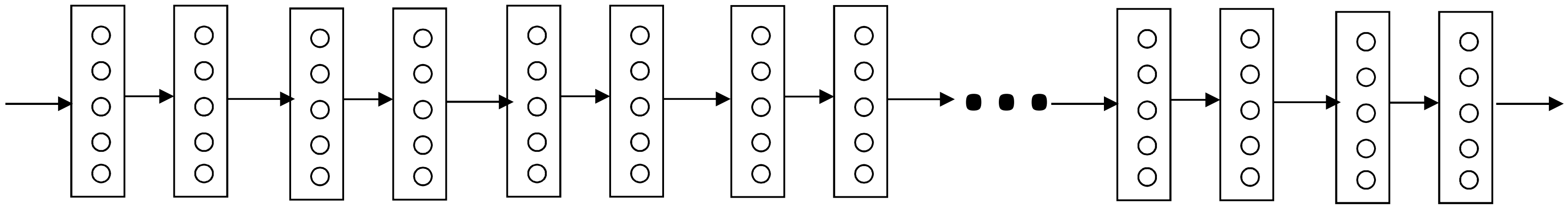


The **kid** watching the cartoon channel in the TV, **is** very happy.

The **kids** watching the cartoon channel in the TV, **are** very happy.

RNN is not capable of representing very long term contextual dependencies within a sentence

Why?



Very deep neural network (≥ 100 layers)

- Initial layer parameters are less effected due to Vanishing gradient problem.

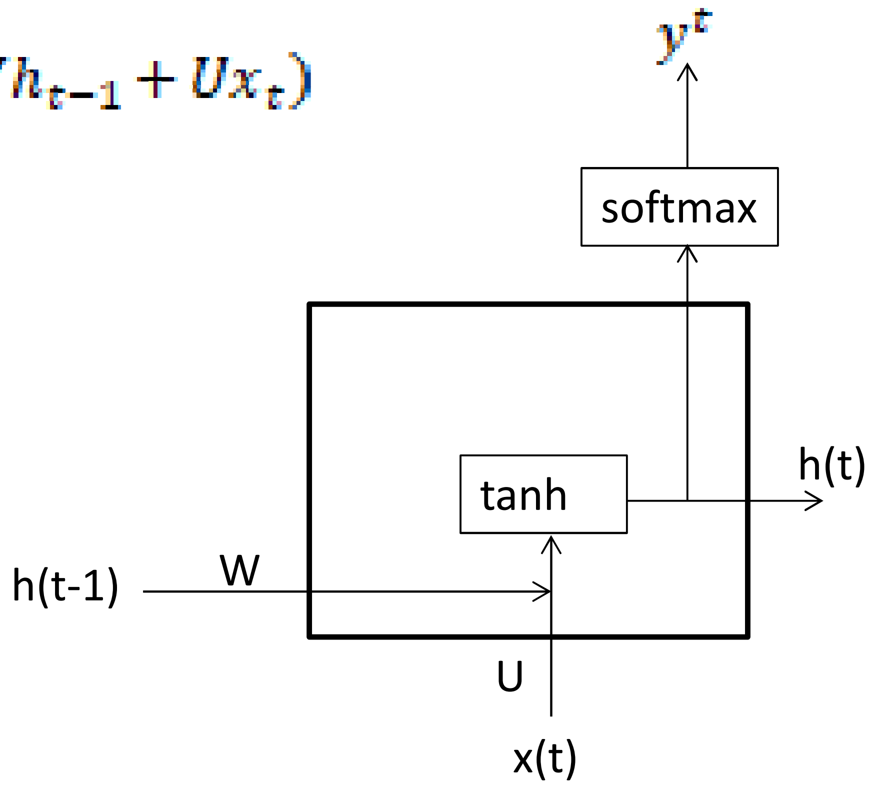
- For RNN, due to this Vanishing gradient problem, for very large sequence, later positioned word are less influenced by very early occurring words. Thus the problem of previous example is happened.

- For the same reason, RNN has some local influences.

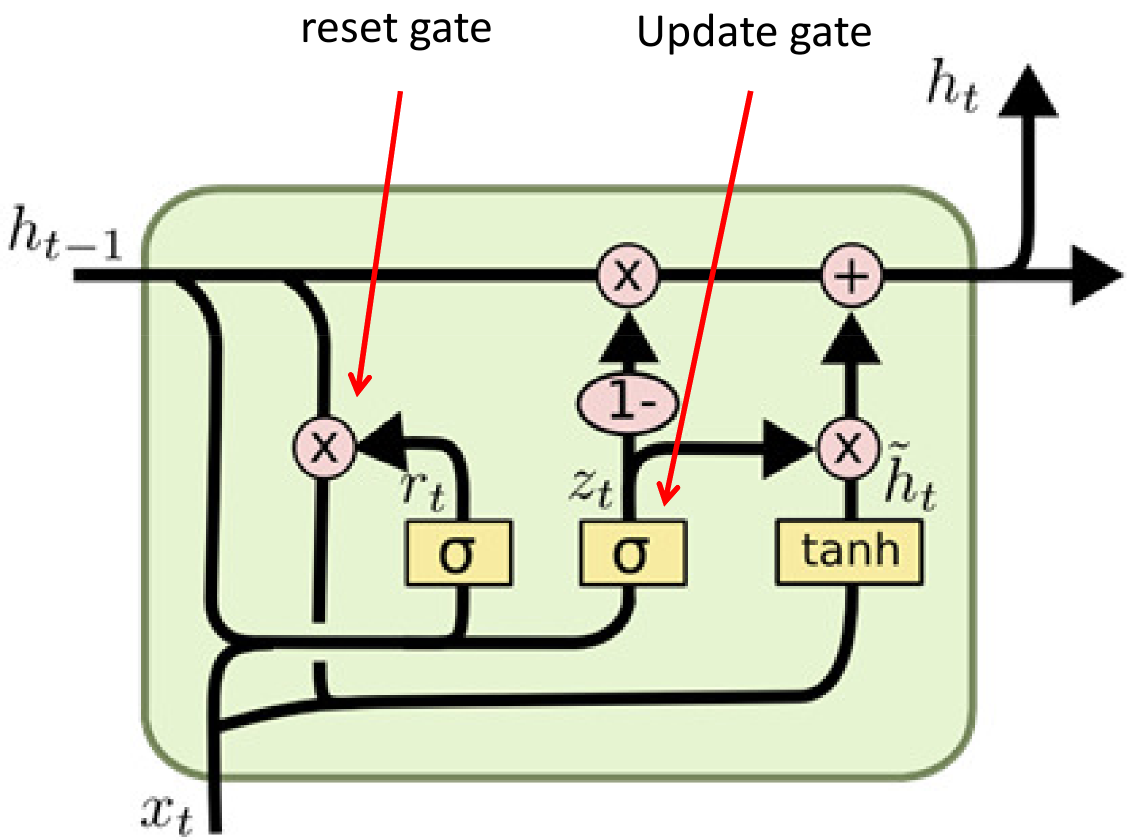
Simple RNN

- Standard RNN computes hidden layer at next time step directly

$$h_t = f(W h_{t-1} + U x_t)$$



Solution: Gated RNN (GRU)



Solution: Gated RNN (GRU)

- GRU first computes an update gate based on current input vector and hidden state

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$ Update gate

- Then it computes the reset gate similarly but with different weights

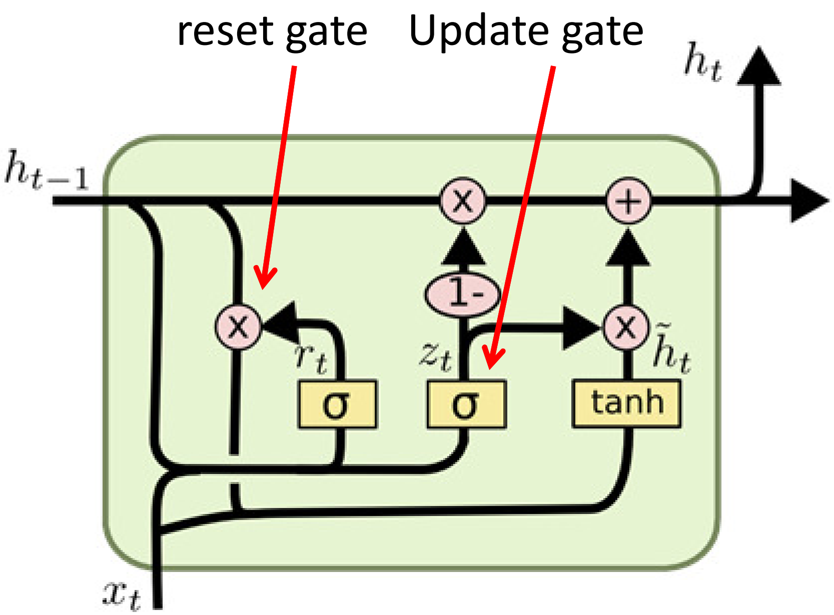
$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$ Reset gate

- The new memory content

$$\tilde{h}_t = \tanh\ (Wx_t + r_t * Uh_{t-1})$$

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

Gated RNN (GRU)



Update gate

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$

Reset gate

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$

The new memory content

$$\tilde{h}_t = \tanh\ (Wx_t + r_t * Uh_{t-1})$$

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

Gated RNN (GRU)

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1})$$ Update gate

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1})$$ Reset gate

$$\tilde{h}_t = \tanh\ (Wx_t + r_t * Uh_{t-1})$$ New memory content

If reset gate is 0 , then this ignores the previous memory and only stores the new information

Final memory at time step combines current and previous time steps:

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

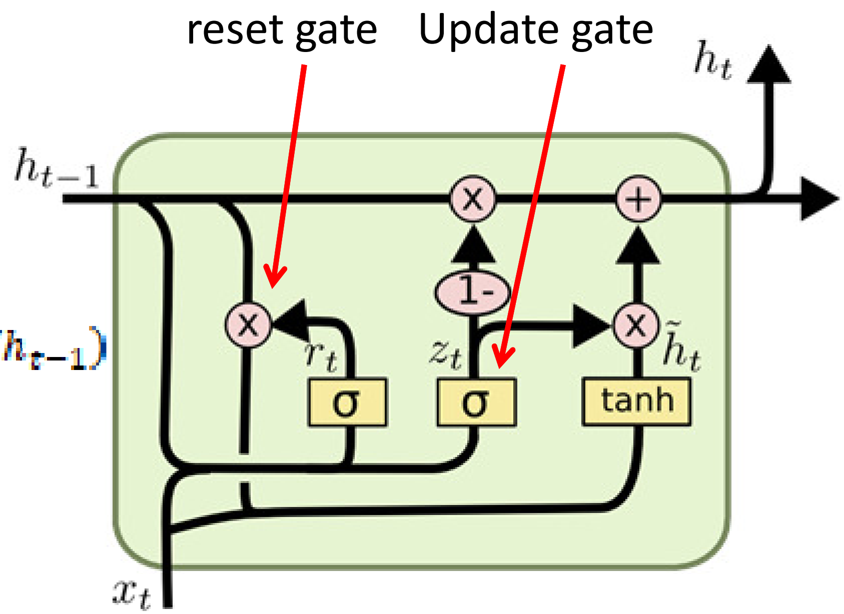
Gated RNN (GRU)

Reset gate: $r_t = \sigma(W^{(r)} x_t + U^{(r)} h_{t-1})$

Update gate: $z_t = \sigma(W^{(z)} x_t + U^{(z)} h_{t-1})$

New memory content: $\tilde{h}_t = \tanh\ (W x_t + r_t * U h_{t-1})$

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

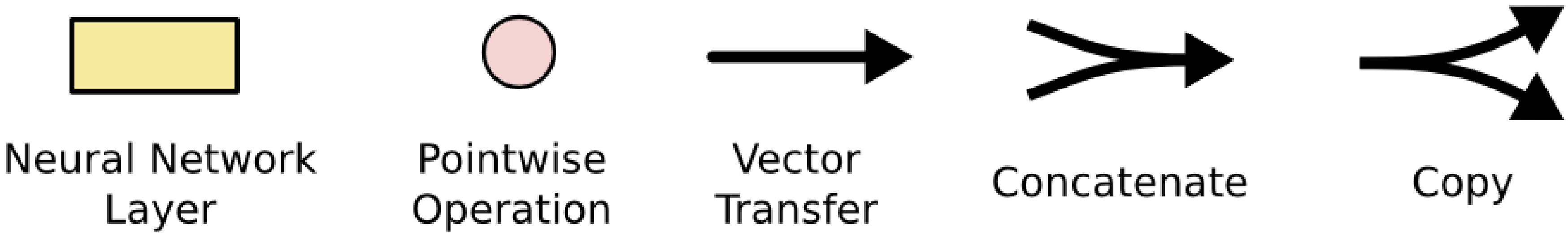


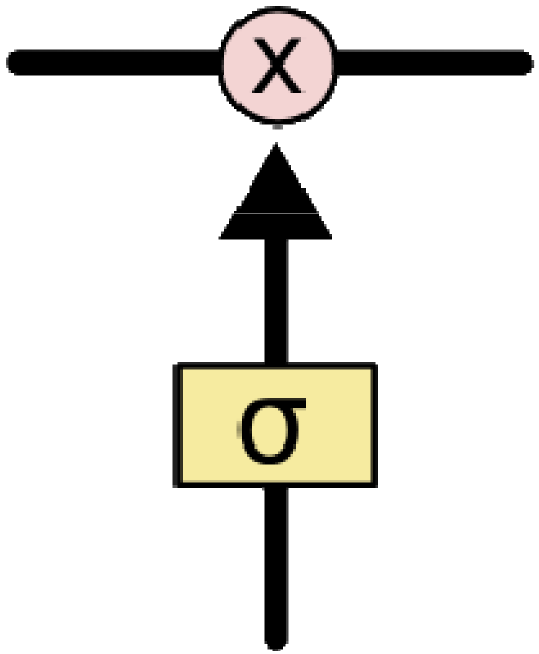
The **kid** watching the cartoon channel in the TV, **is** very happy.

The **kids** watching the cartoon channel in the TV, **are** very happy.

$r_t = 1$ $\qquad$ $z_t = 1$ $\qquad$ $r_t = 1$

When $z_t = 1$, h(t) = h(t-1), so maintaining the previous h(t) values





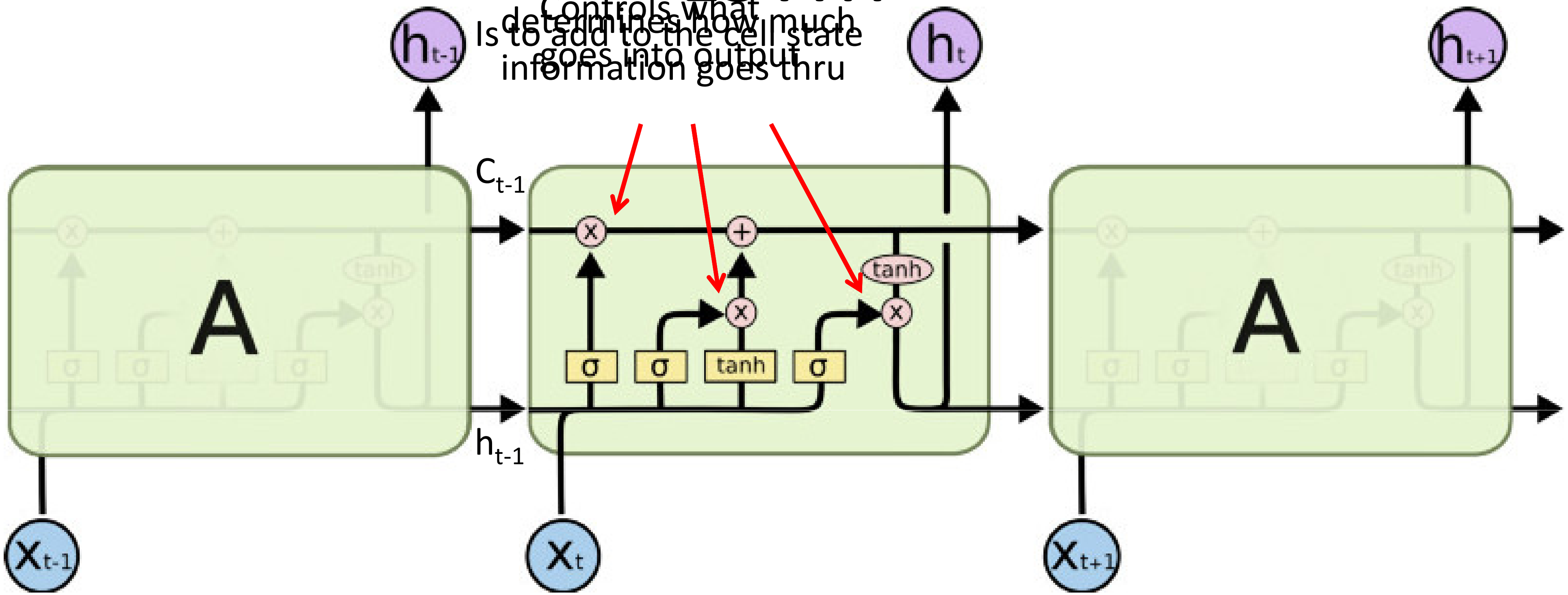
The sigmoid layer outputs numbers between 0-1 determine how much
each component should be let through. Pink X gate is point-wise multiplication.

LSTM

This sigmoid gate determines how much information goes thru

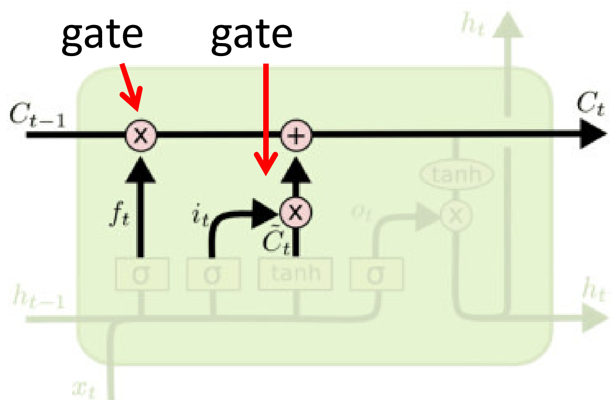
This decides what info Is to add to the cell state

Output gate Controls what goes into output



Forget gate

input gate

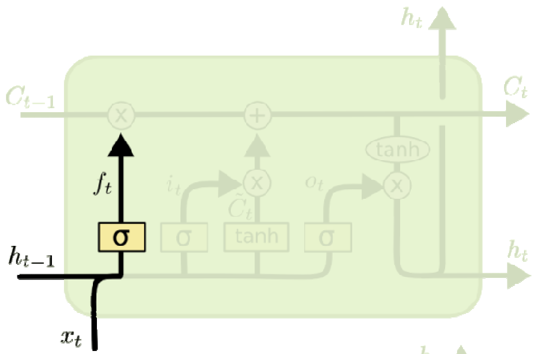


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

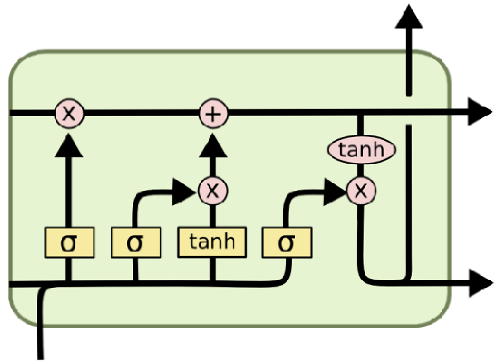
The core idea is this cell state C_t, it is changed slowly, with only minor linear interactions. It is very easy for information to flow along it unchanged.

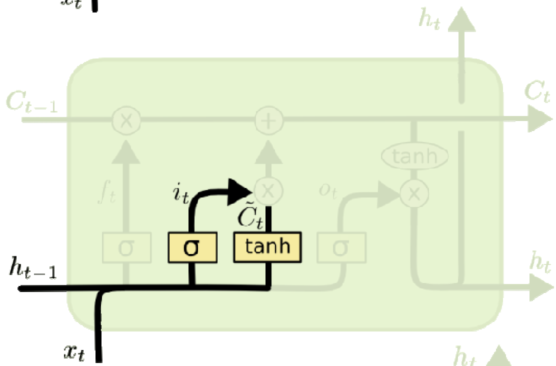
Why sigmoid or tanh:

Sigmoid: 0,1 gating as switch. Vanishing gradient problem in LSTM is handled already. ReLU replaces tanh ok?



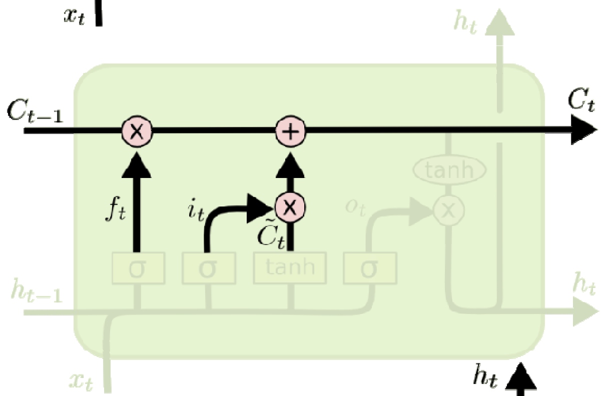
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$





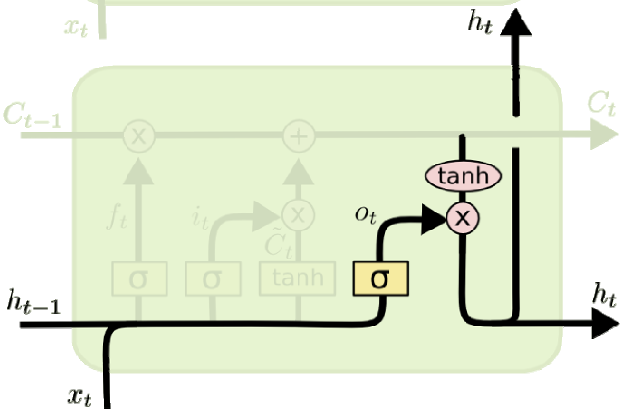
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

i_t decides what component is to be updated.
C'$_t$ provides change contents



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

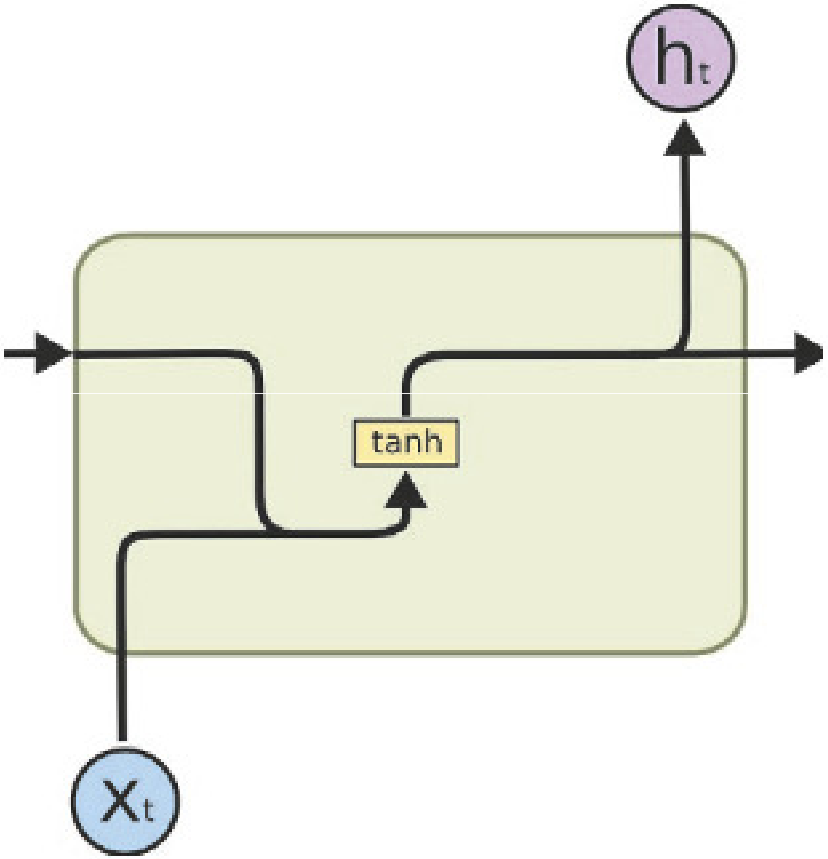
Updating the cell state



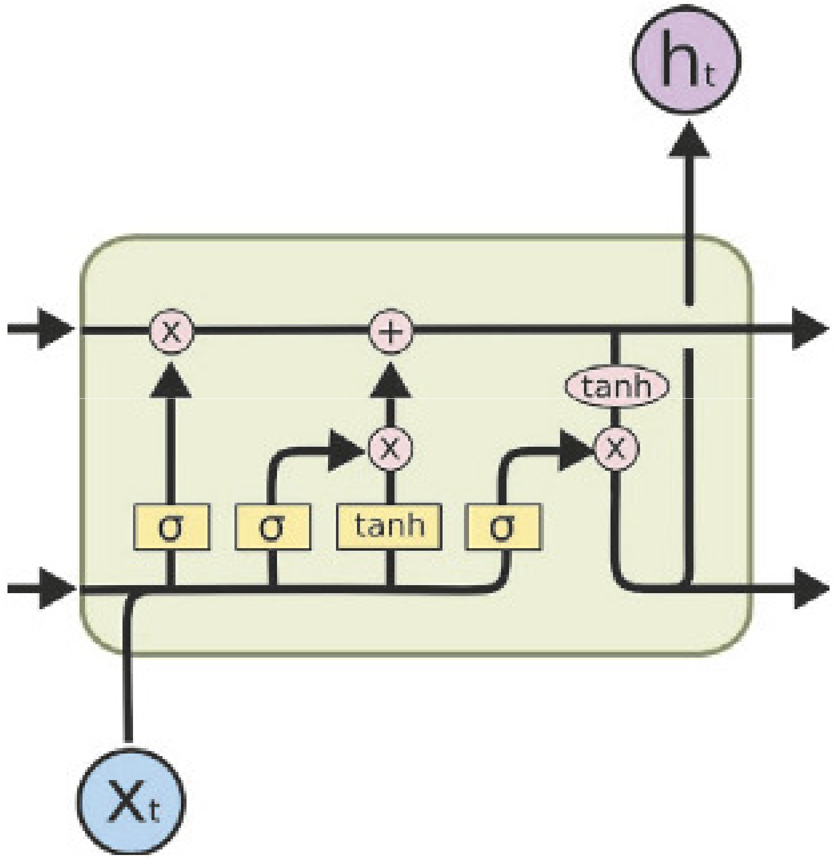
$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Decide what part of the cell state to output

RNN vs LSTM

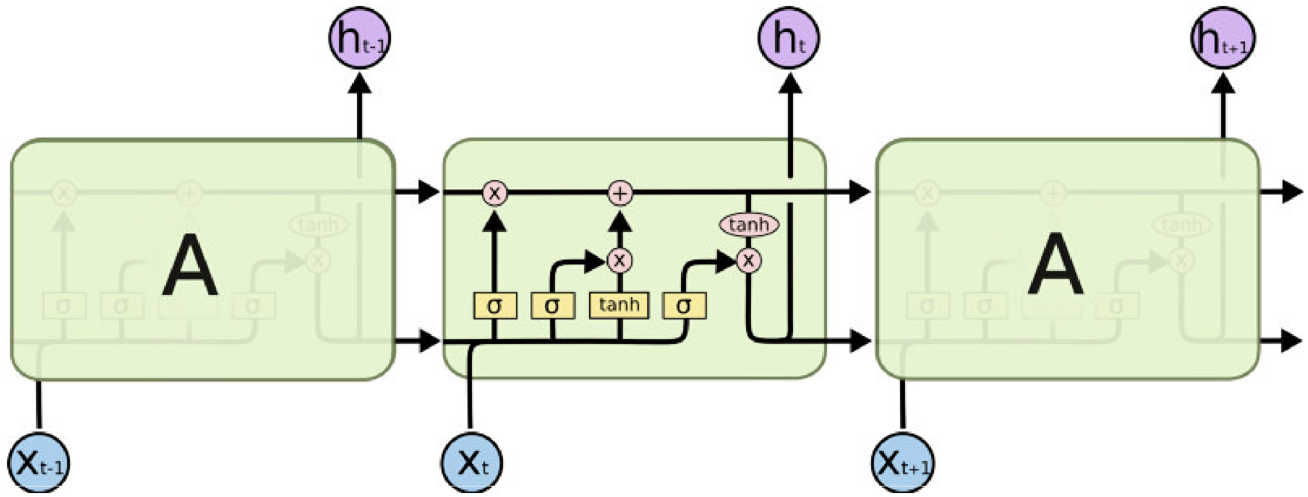


(a) RNN

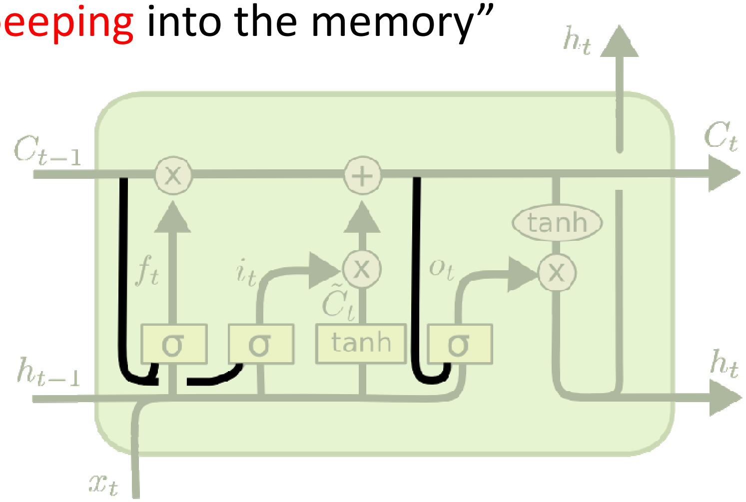


(b) LSTM

Peephole LSTM



Allows "peeping into the memory"

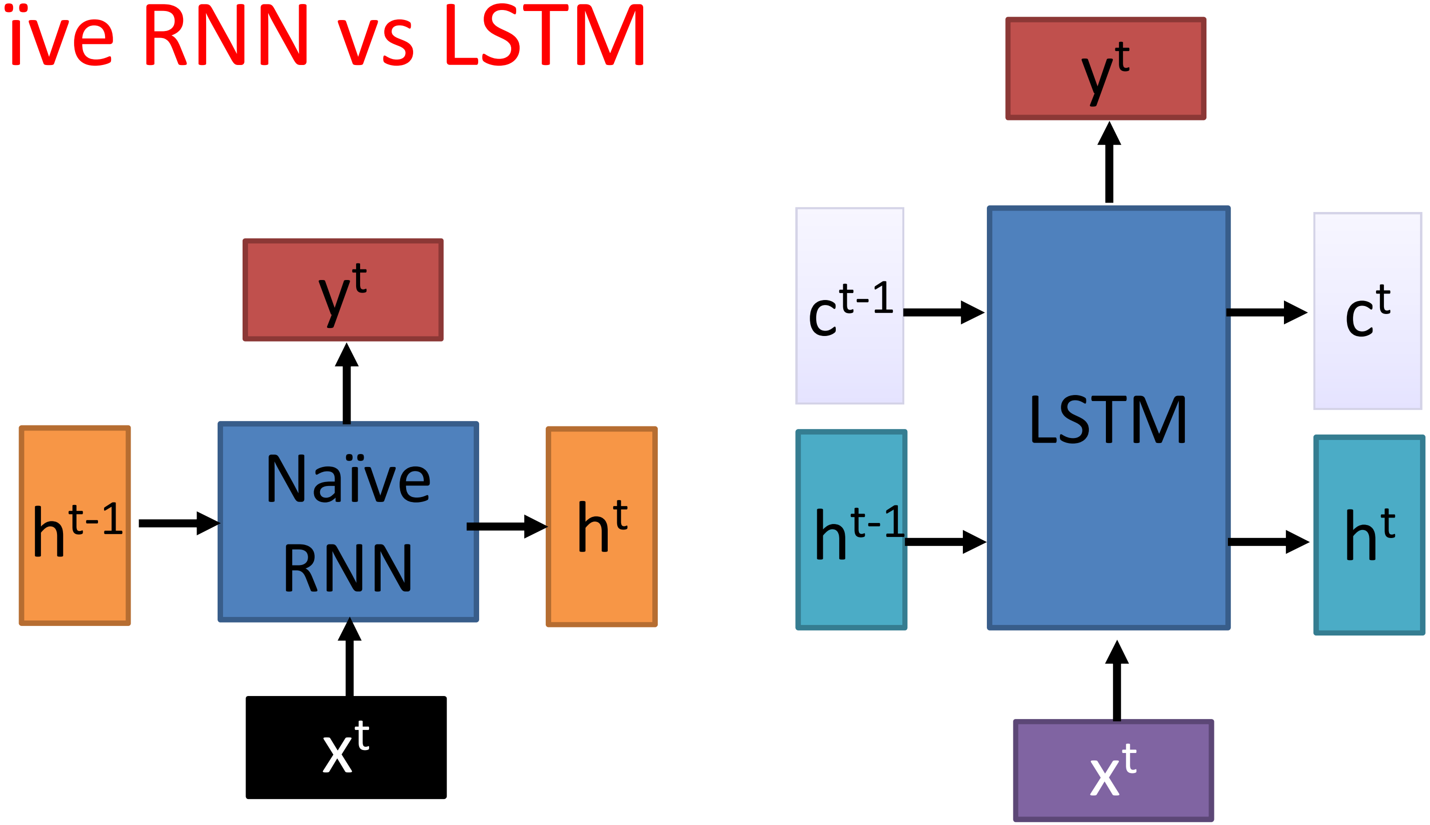


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] + b_f\right)$$

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

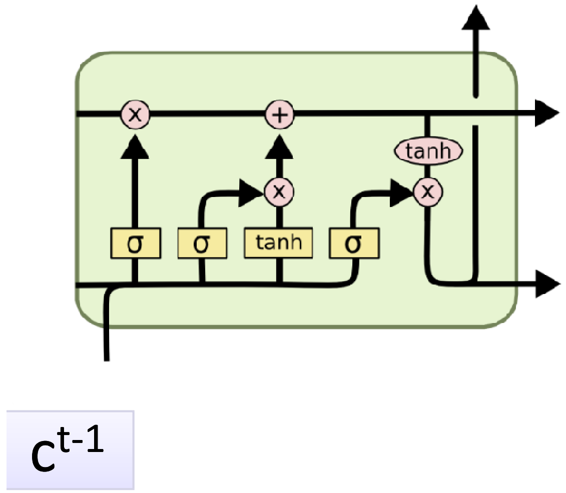
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] + b_o\right)$$

Naïve RNN vs LSTM

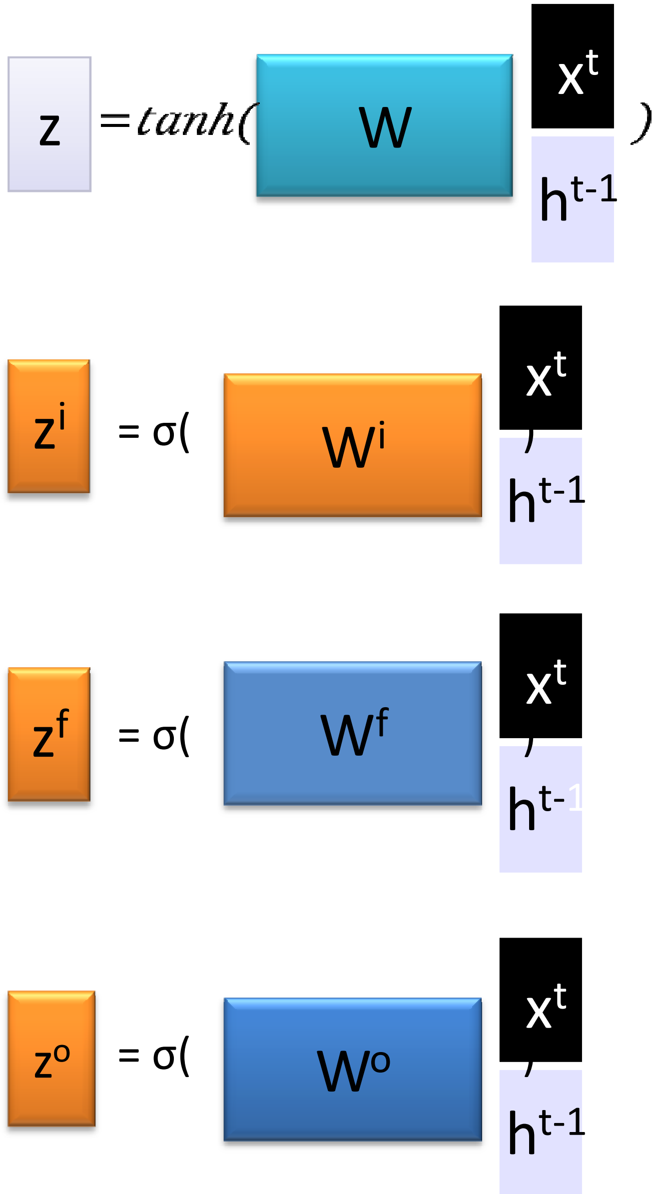


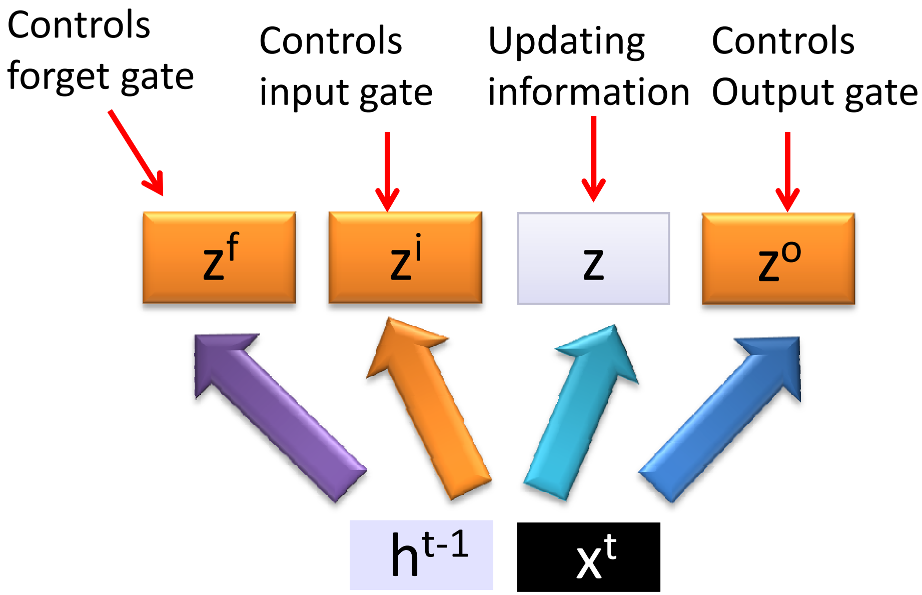
c changes slowly ➡ c^t is c^{t-1} added by something

h changes faster ➡ h^t and h^{t-1} can be very different

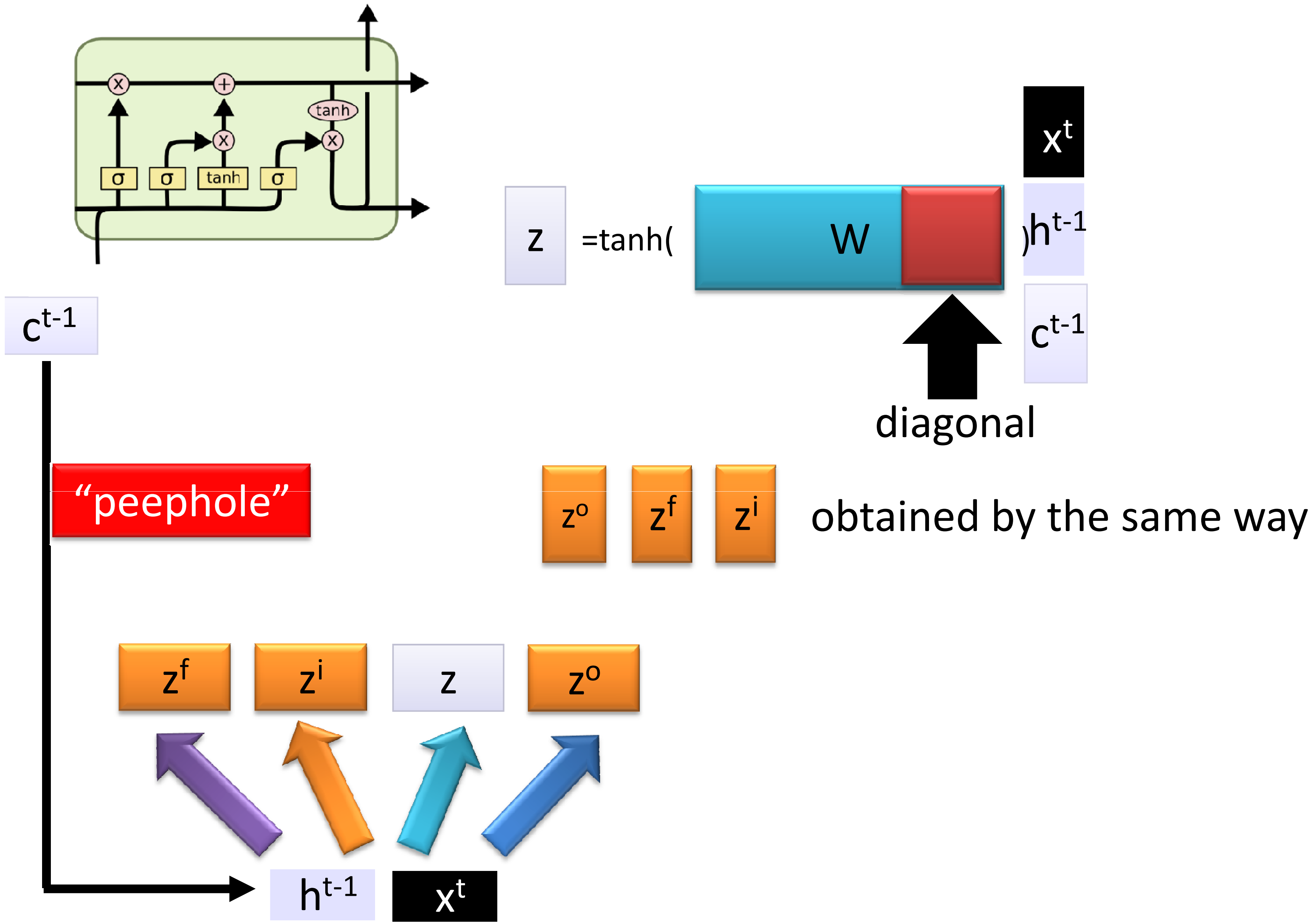


These 4 matrix computation should be done concurrently.

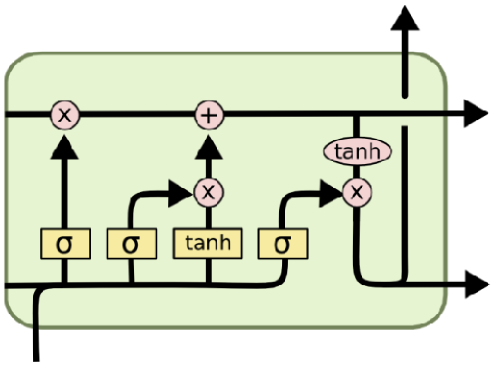




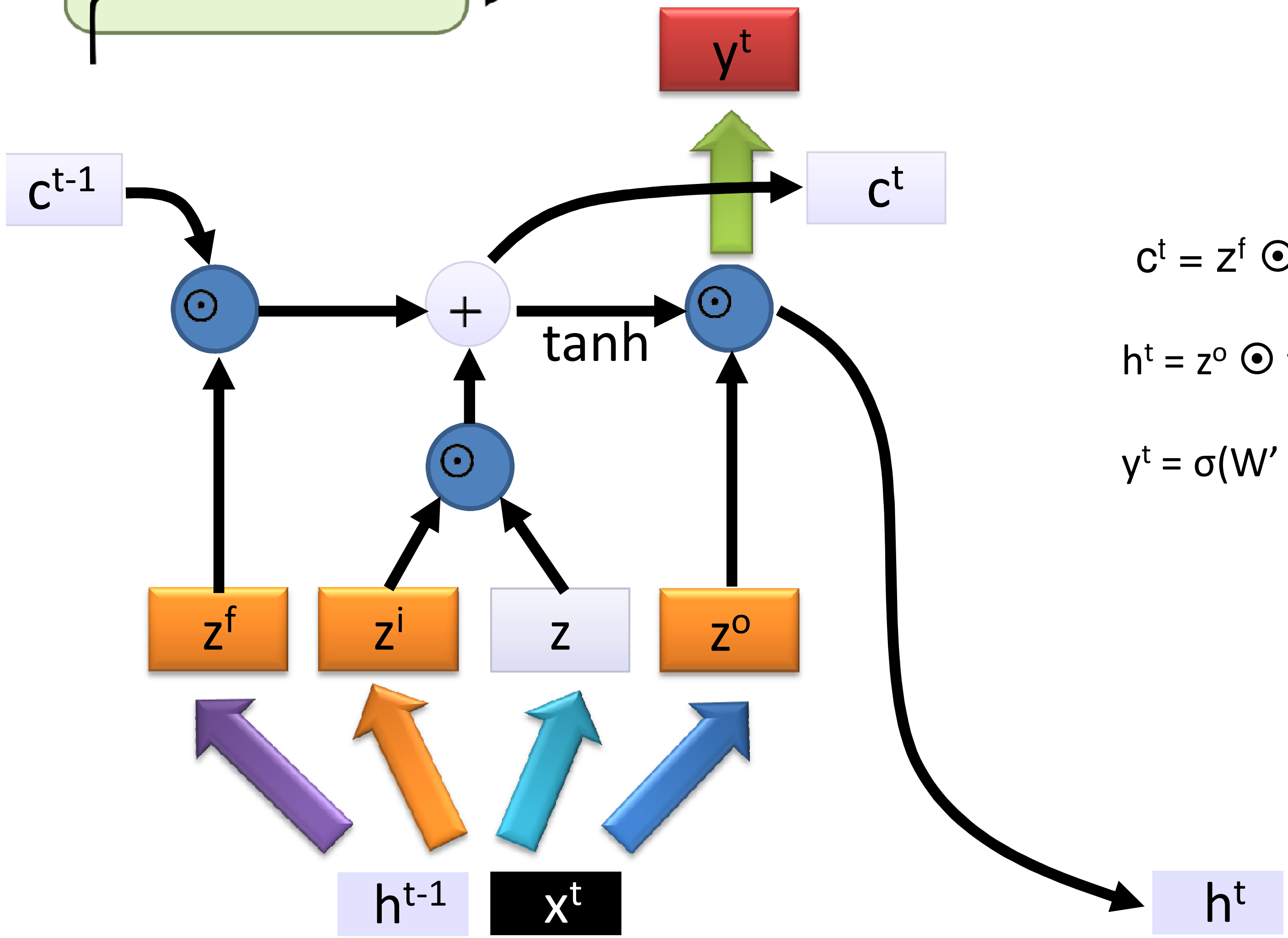
Information flow of LSTM



Information flow of LSTM



$\odot$ Element-wise multiply



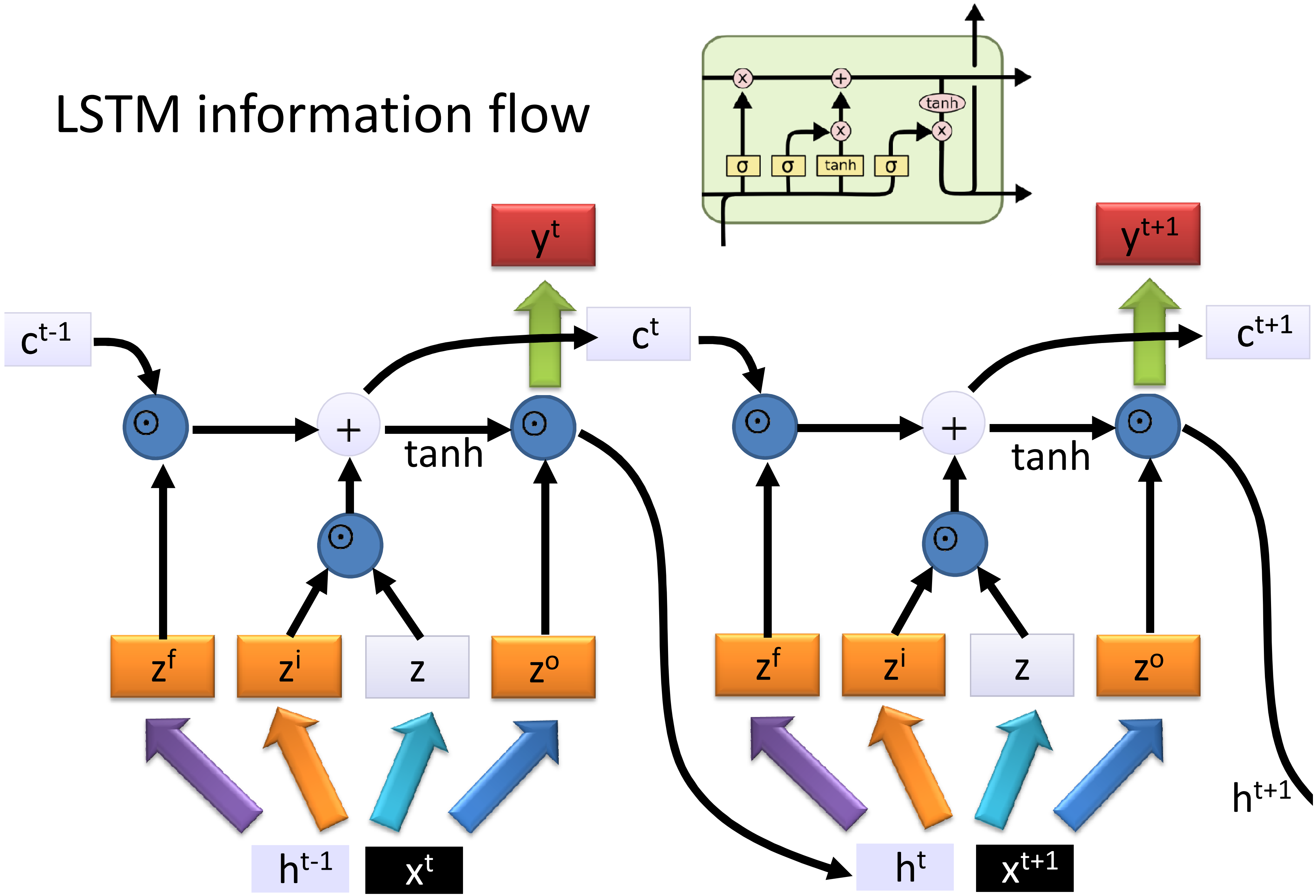
$$c^t = z^f \odot c^{t-1} + z^i \odot z$$

$$h^t = z^o \odot \tanh(c^t)$$

$$y^t = \sigma(W' h^t)$$

Information flow of LSTM

LSTM information flow

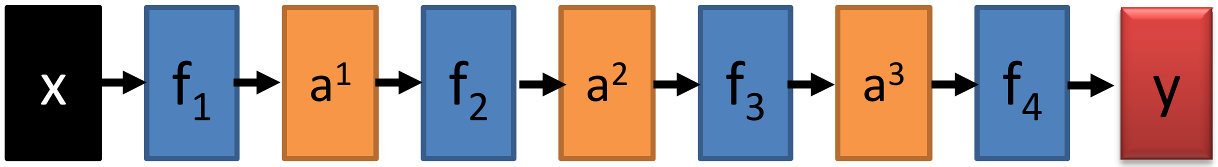


Information flow of LSTM

Feed-forward vs Recurrent Network

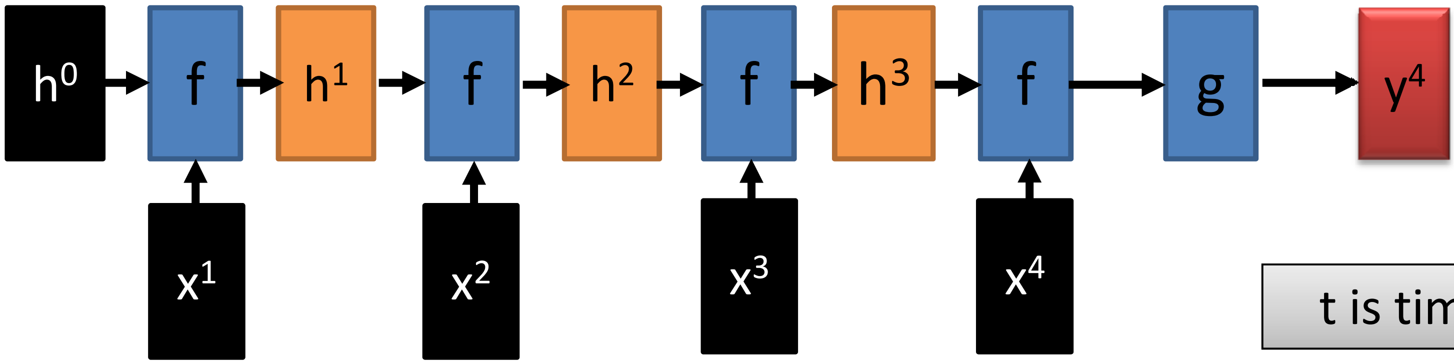
1. Feedforward network does not have input at each step

2. Feedforward network has different parameters for each layer



t is layer

$a^t = f_t(a^{t-1}) = \sigma(W^t a^{t-1} + b^t)$



t is time step

$a^t = f(a^{t-1}, x^t) = \sigma(W^h a^{t-1} + W^i x^t + b^i)$

We will turn the recurrent network 90 degrees.

to continue...