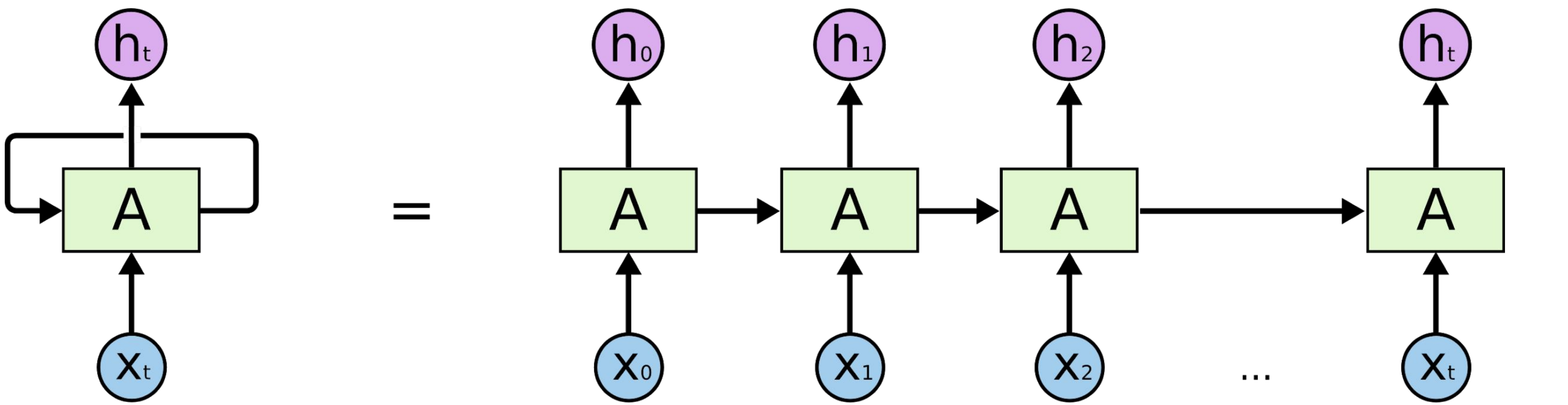
Simple RNN

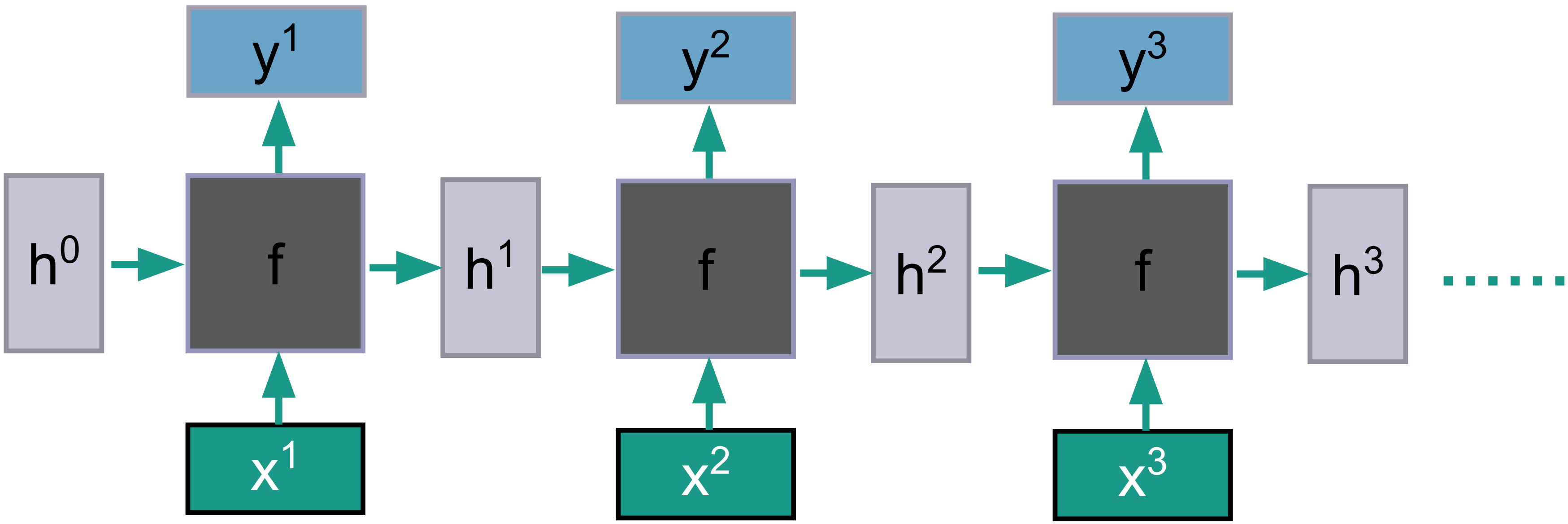
Recurrent Neural Network



How does RNN reduce complexity?

- Given function f: h',y=f(h,x)

h and h' are vectors with the same dimension

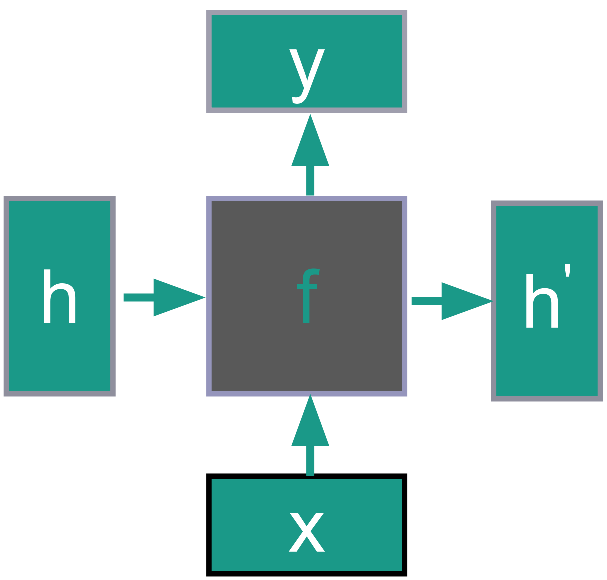


$x^n = vector$

Only need one function f.

Naïve RNN

- Given function f: h',y=f(h,x)



$$h' = \sigma(W^h \, h + W^i \, x)$$

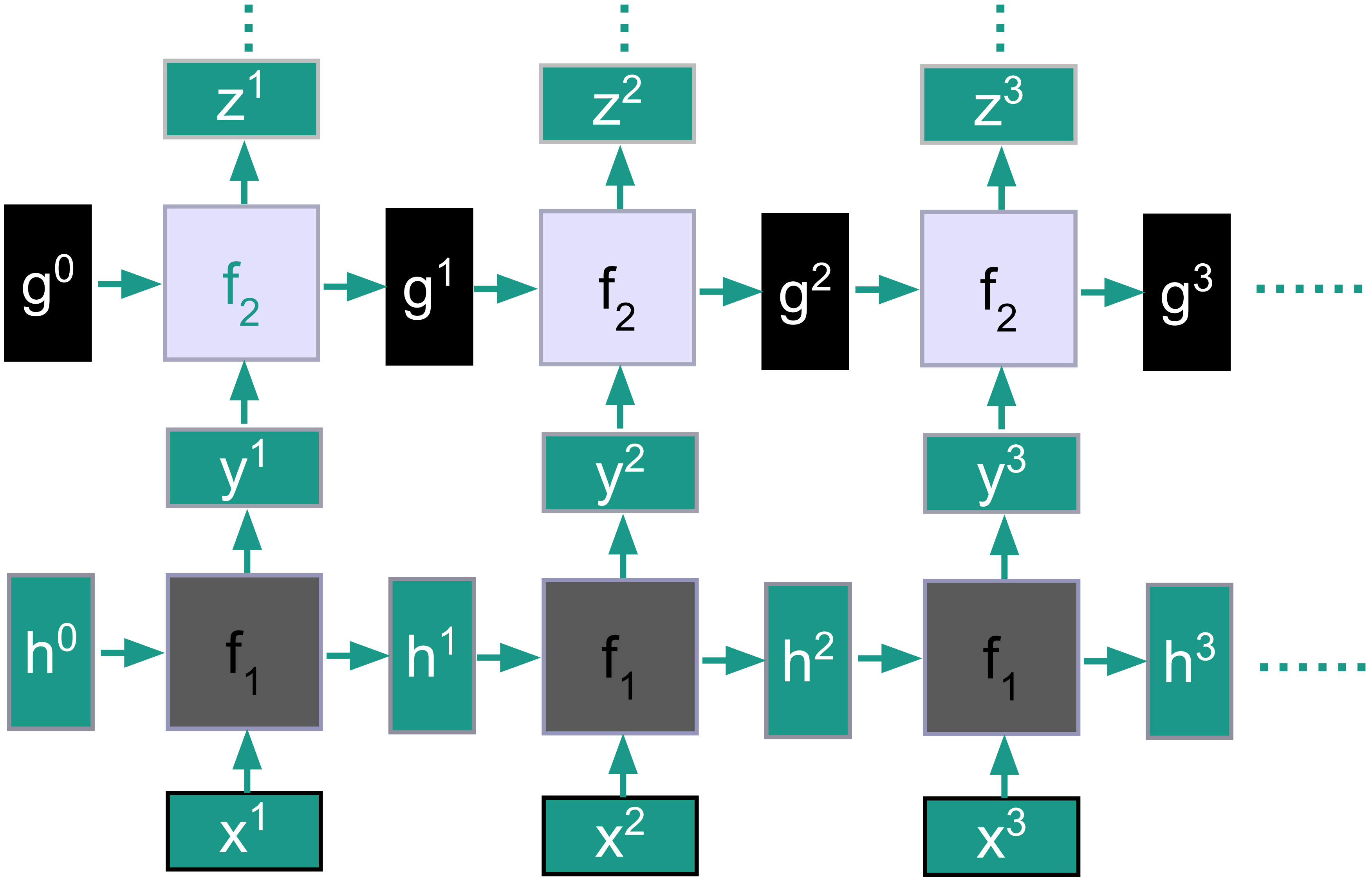
$$y = \sigma(W^o \, h')$$

Note:y is computed from h'

We have ignored the bias

Deep RNN

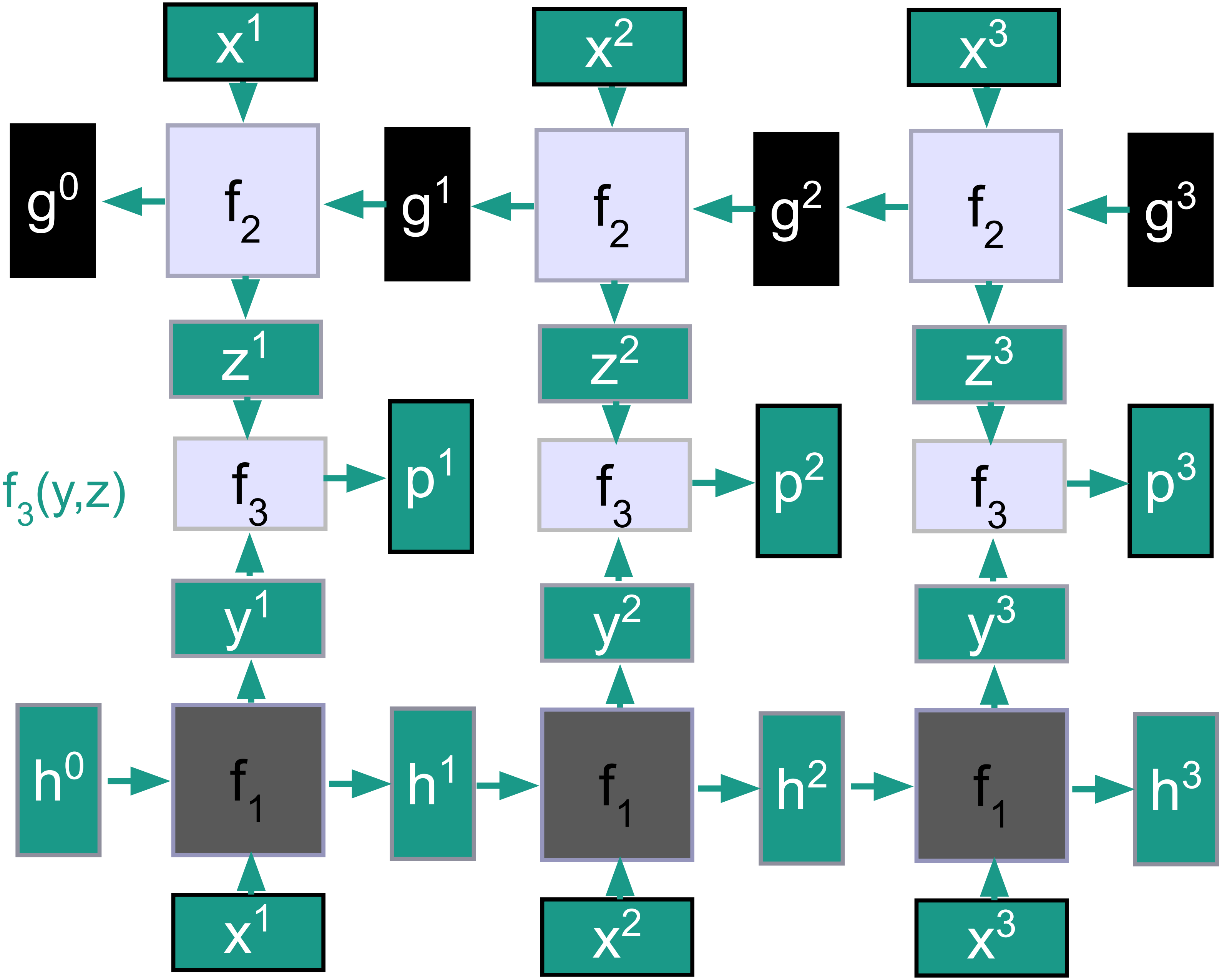
$h', y = f_1(h, x)$, $g', z = f_2(g, y)$ …



Bidirectional RNN

$y, h = f_1(x, h)$ $z, g = f_2(g, x)$

$p = f_3(y, z)$

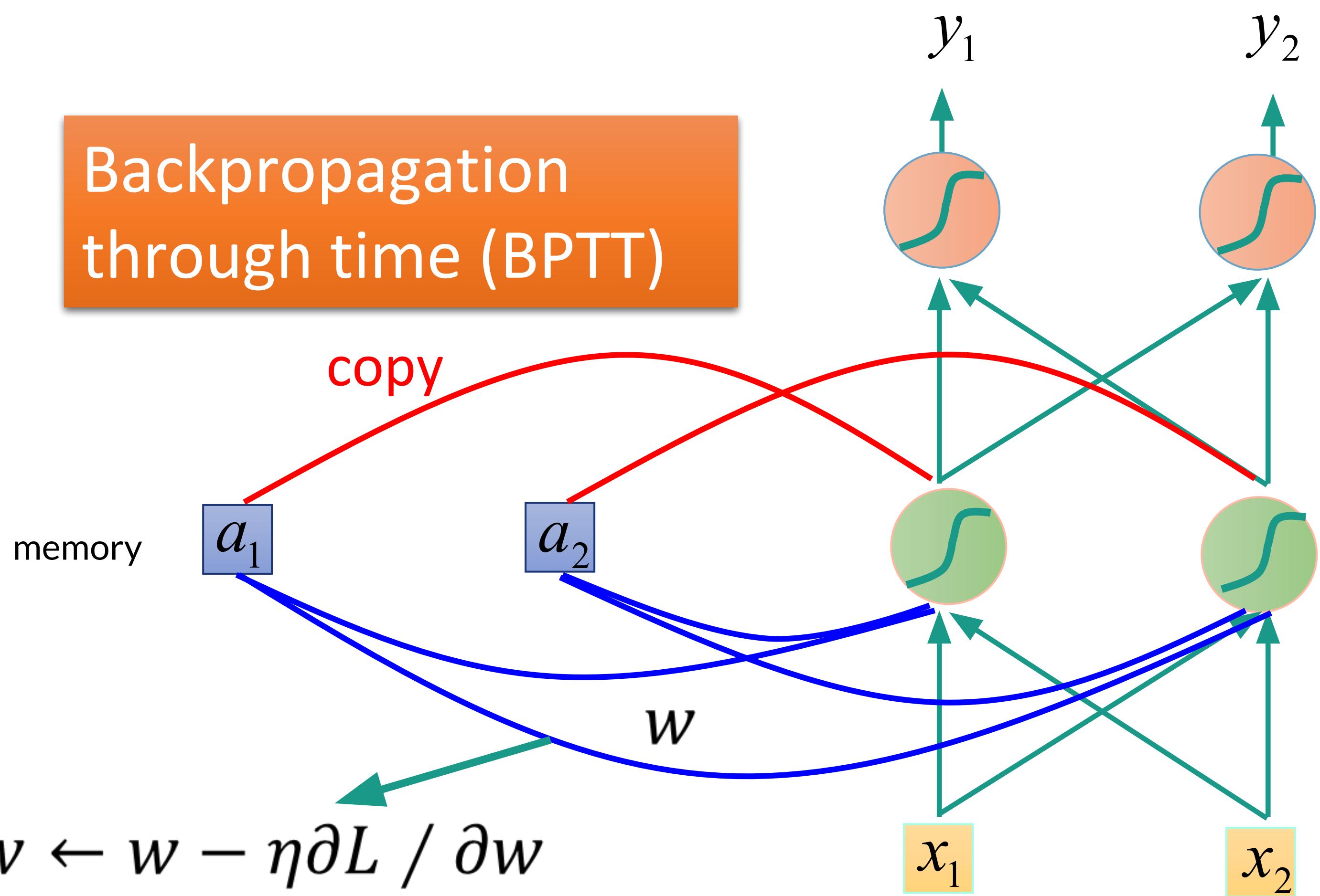


Problems with naive RNN

- “memory” to short.
- Gradient vanishing
- Gradient exploding

How to Training RNN

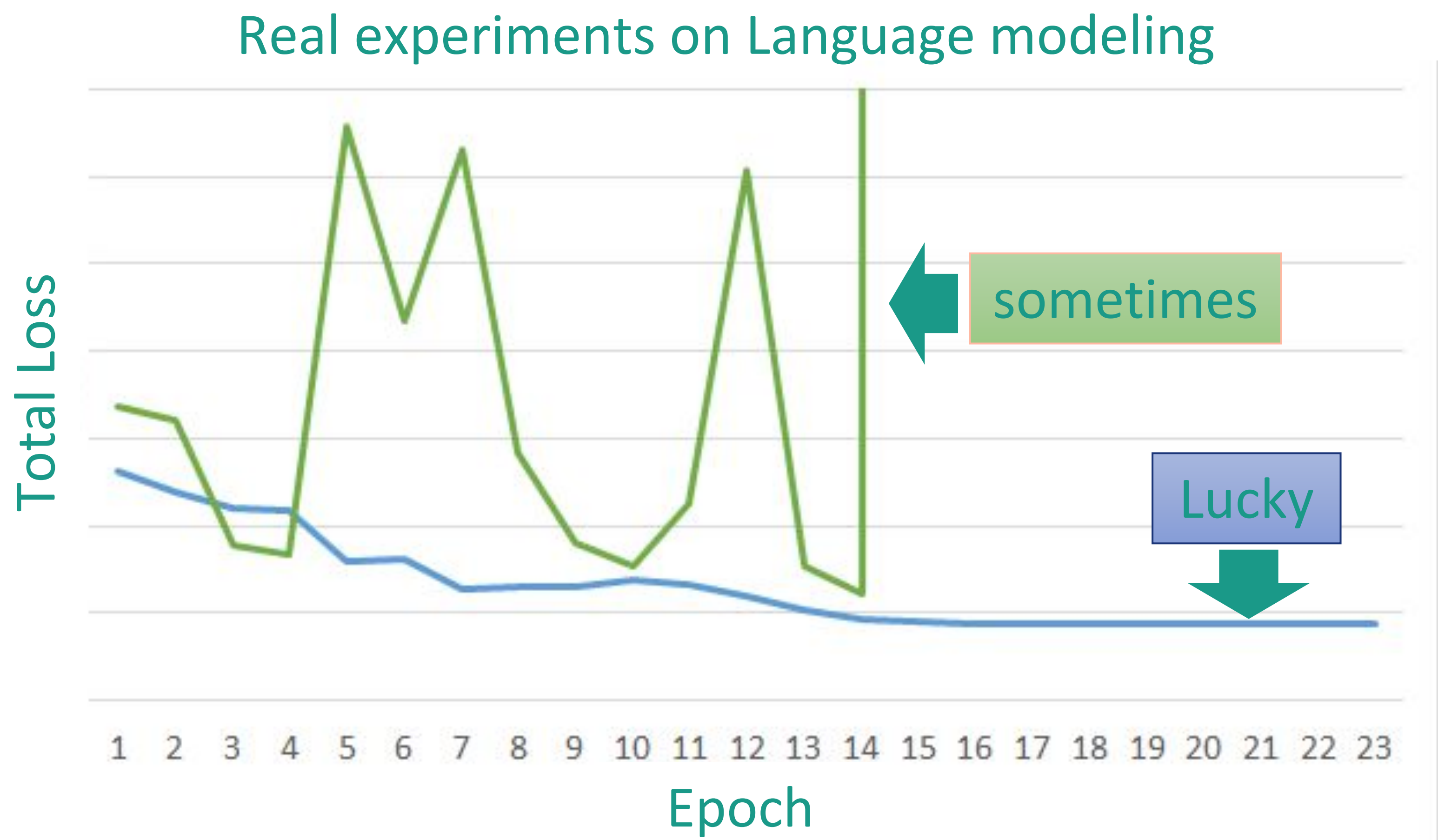
Backpropagation through time (BPTT)



$$w \leftarrow w - \eta \partial L / \partial w$$

RNN-based network

- RNN-based network is not always easy to learn

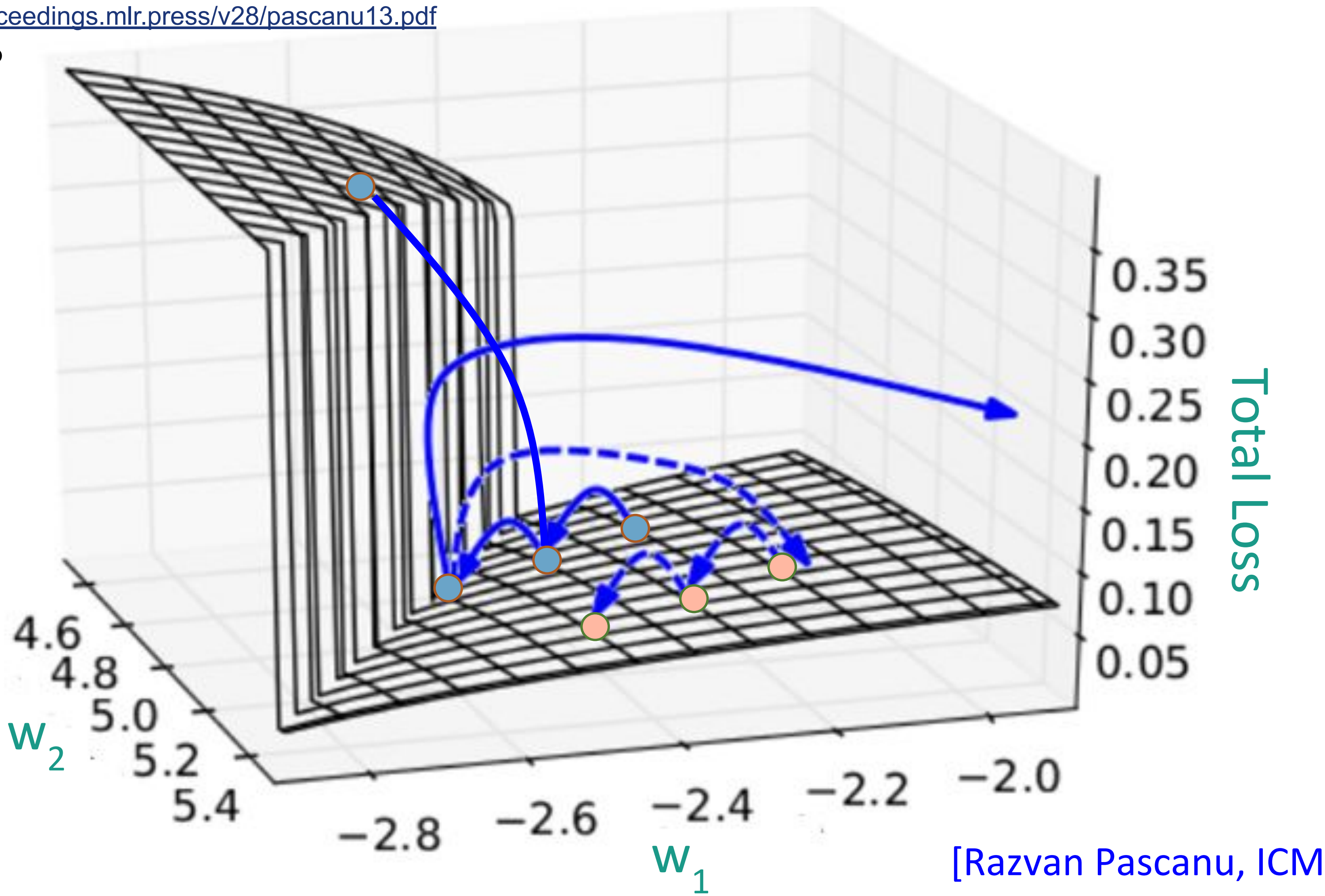


The error surface is rough.

$$w \leftarrow w - \eta \partial L / \partial w$$

http://proceedings.mlr.press/v28/pascanu13.pdf

figure.6



[Razvan Pascanu, ICML’13]

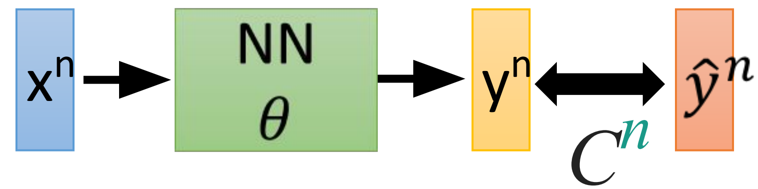
Update Weights

$$w = 1 \quad \Rightarrow \quad y^{1000} = 1$$

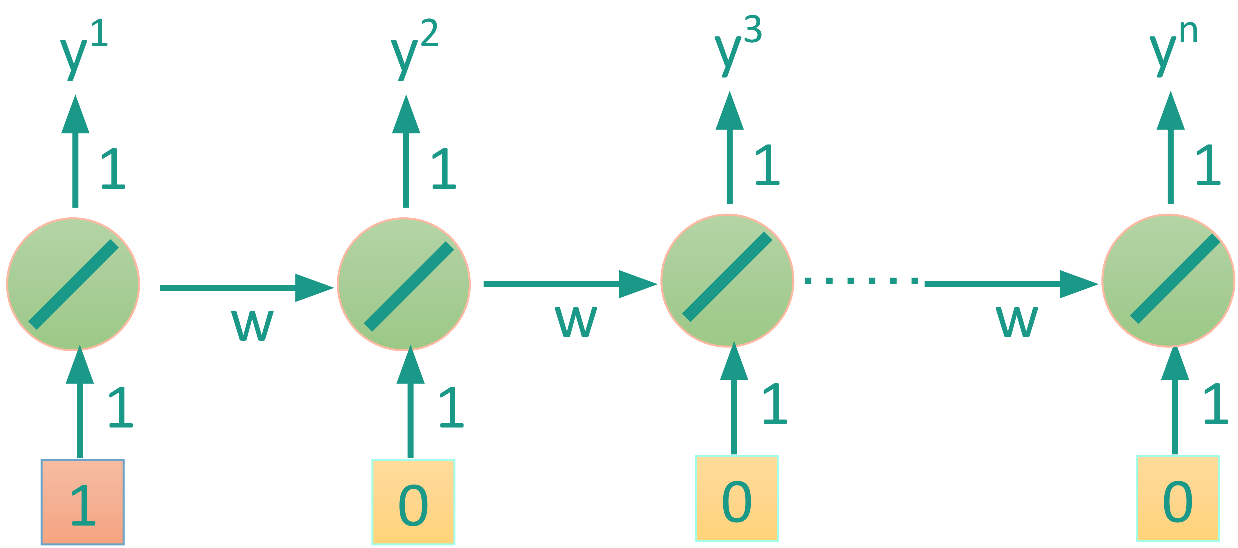
$$w = 1.01 \quad \Rightarrow \quad y^{1000} \approx 20000$$

$$w = 0.99 \quad \Rightarrow \quad y^{1000} \approx 0$$

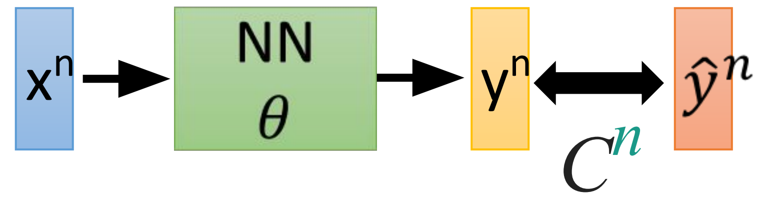
$$w = 0.01 \quad \Rightarrow \quad y^{1000} \approx 0$$



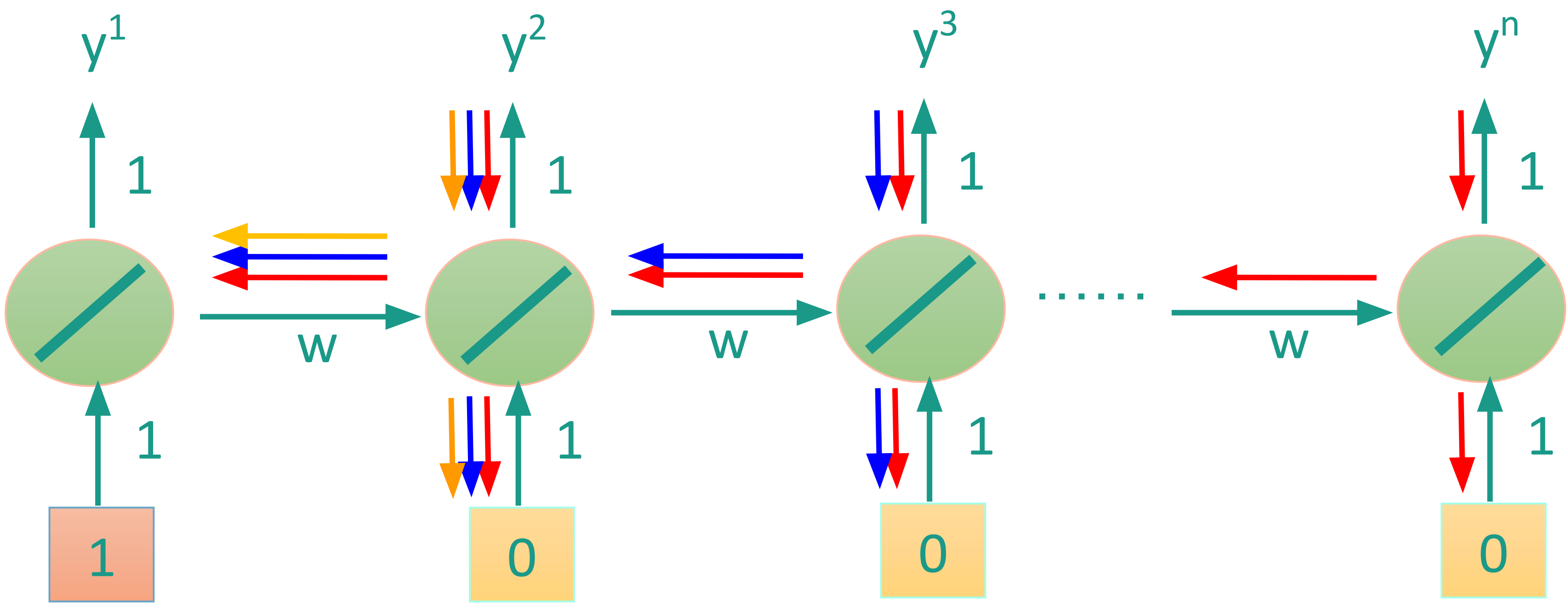
$$y^n = 1 * W^{n-1}$$



Backpropagation

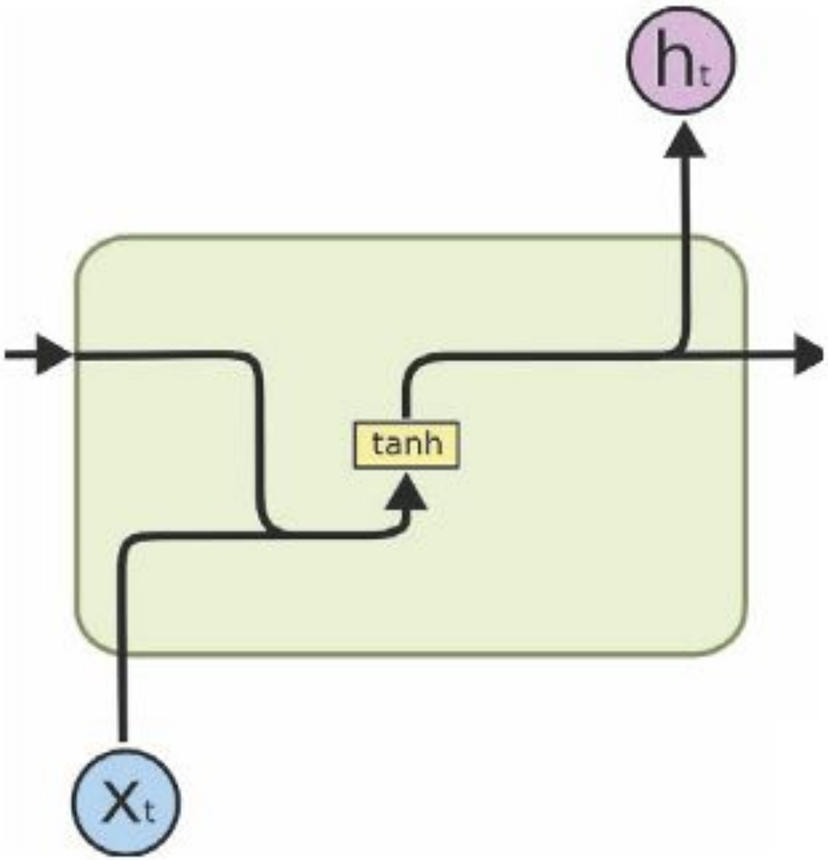


$$\frac{\partial C^n}{\partial W} = \boxed{\frac{\partial C^n}{\partial y^n}} \frac{\partial y^n}{\partial W}$$

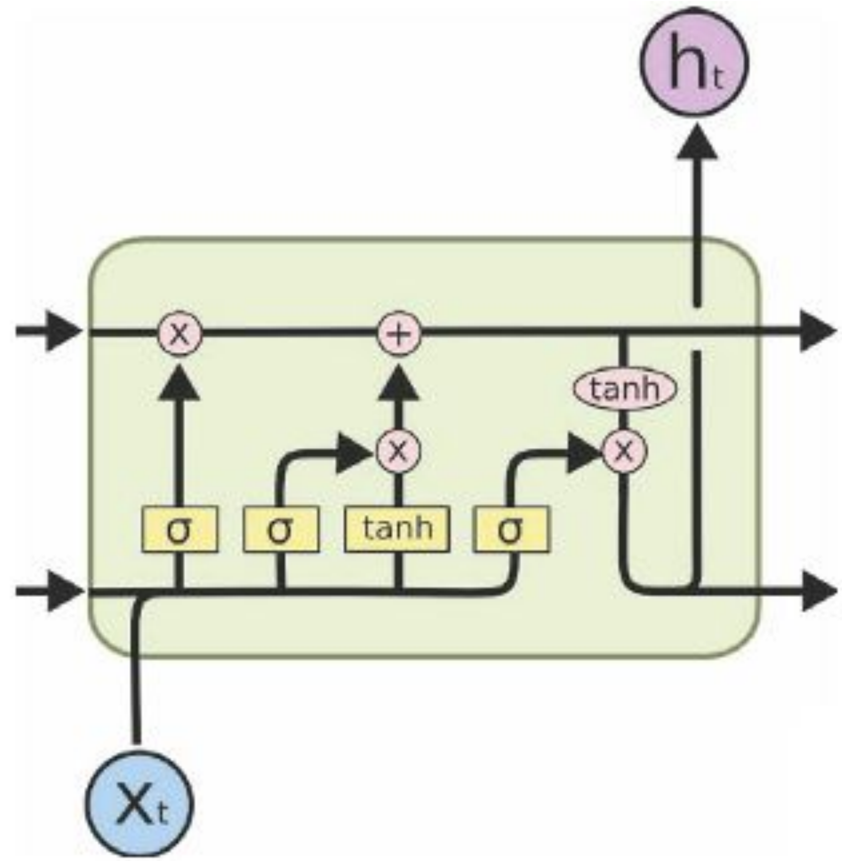


LSTM

RNN vs LSTM

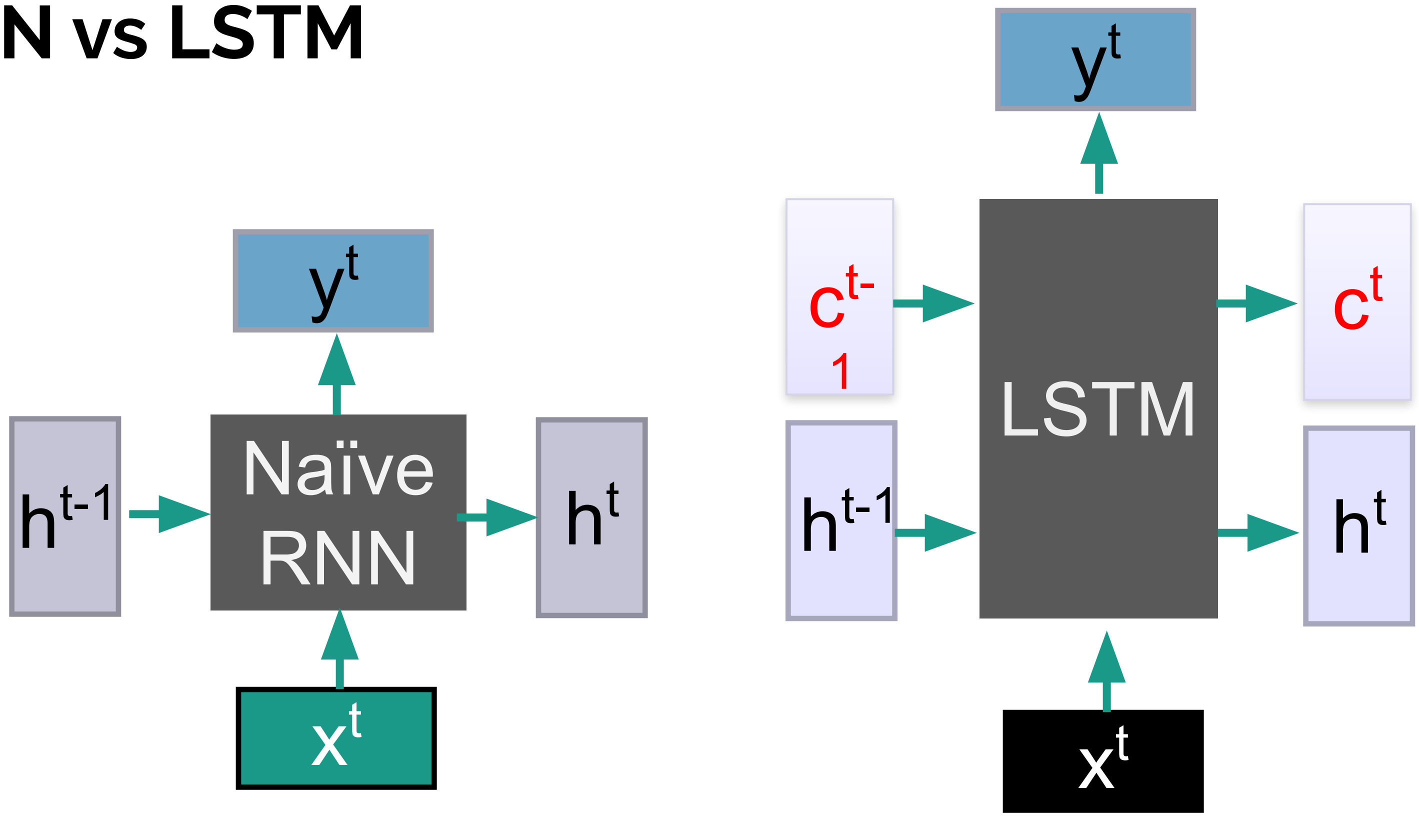


(a) RNN



(b) LSTM

RNN vs LSTM

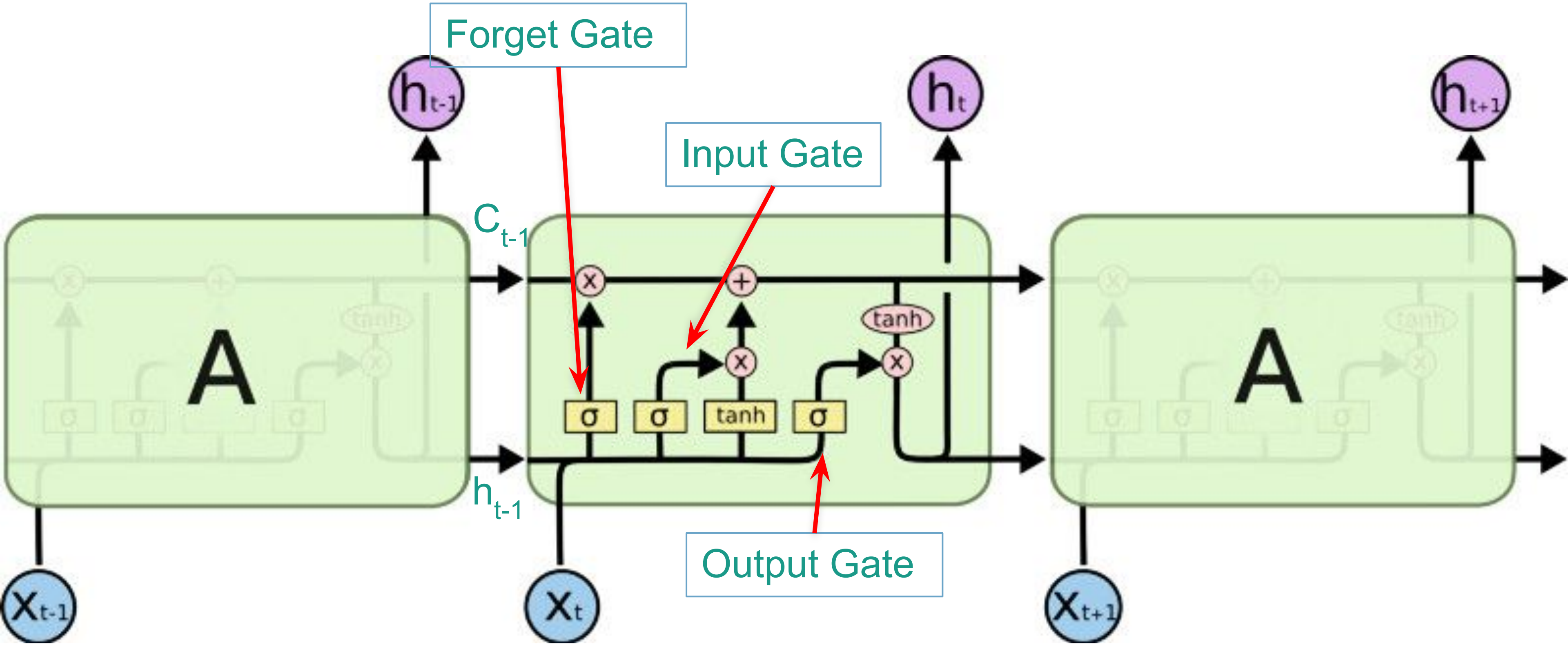


c changes slowly → c^t is c^{t-1} added by something

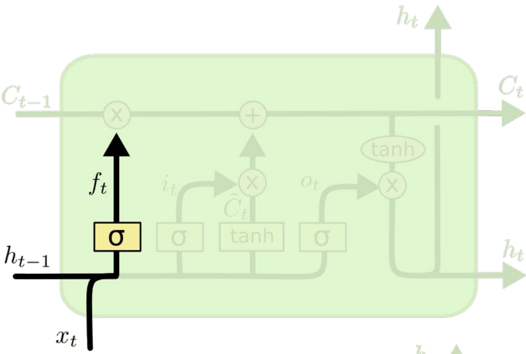
h changes faster → h^t and h^{t-1} can be very different

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

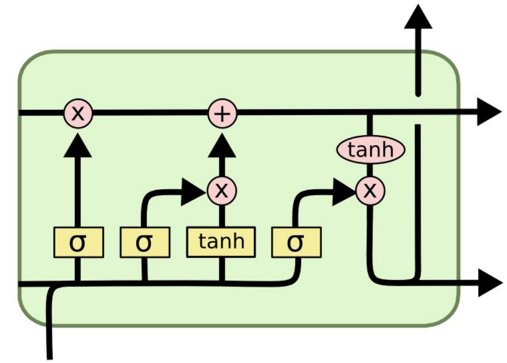
LSTM

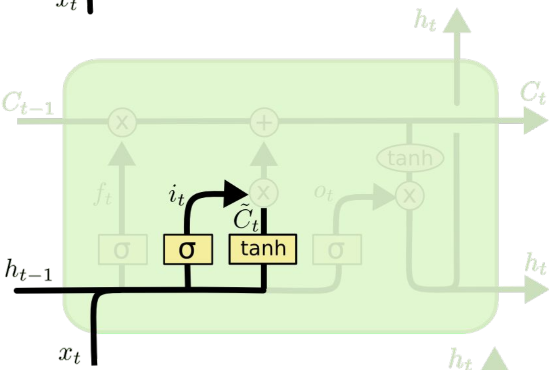


Forget Gate
Input Gate
Output Gate
C_{t-1}
h_{t-1}
h_{t-1}
h_t
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
X_{t-1}
X_t
X_{t+1}



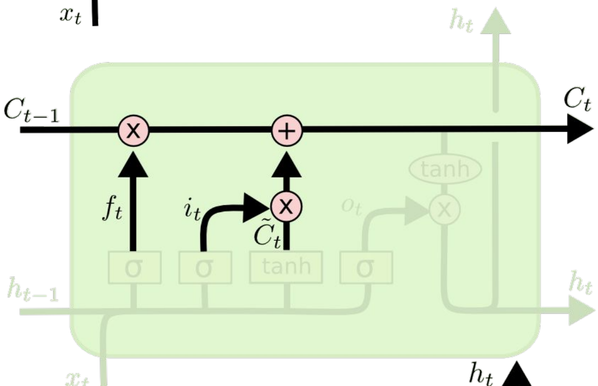
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$





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

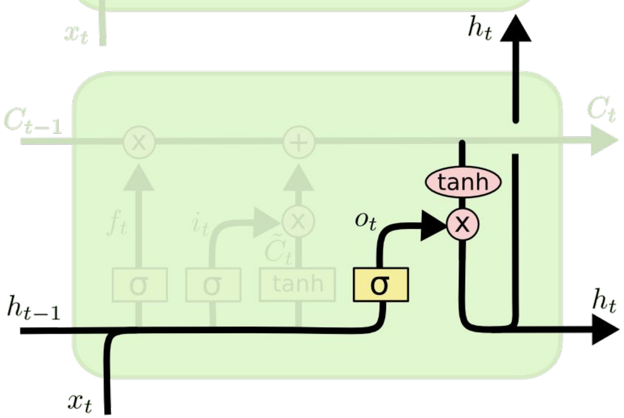
i_t decides what component is to be updated.
C_t provides change contents



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

Updating the cell state

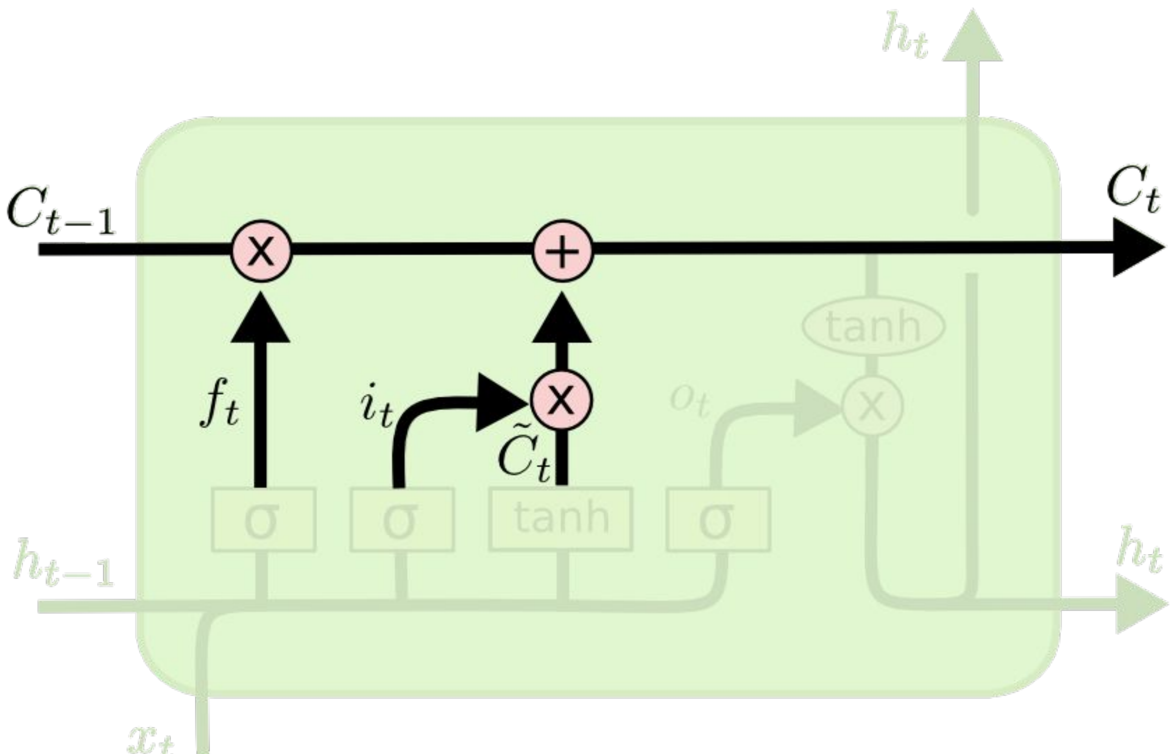
C_t : Cell's memory changed slowly



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

Decide what part of the cell state to output

Cell's memory



Forget Gate

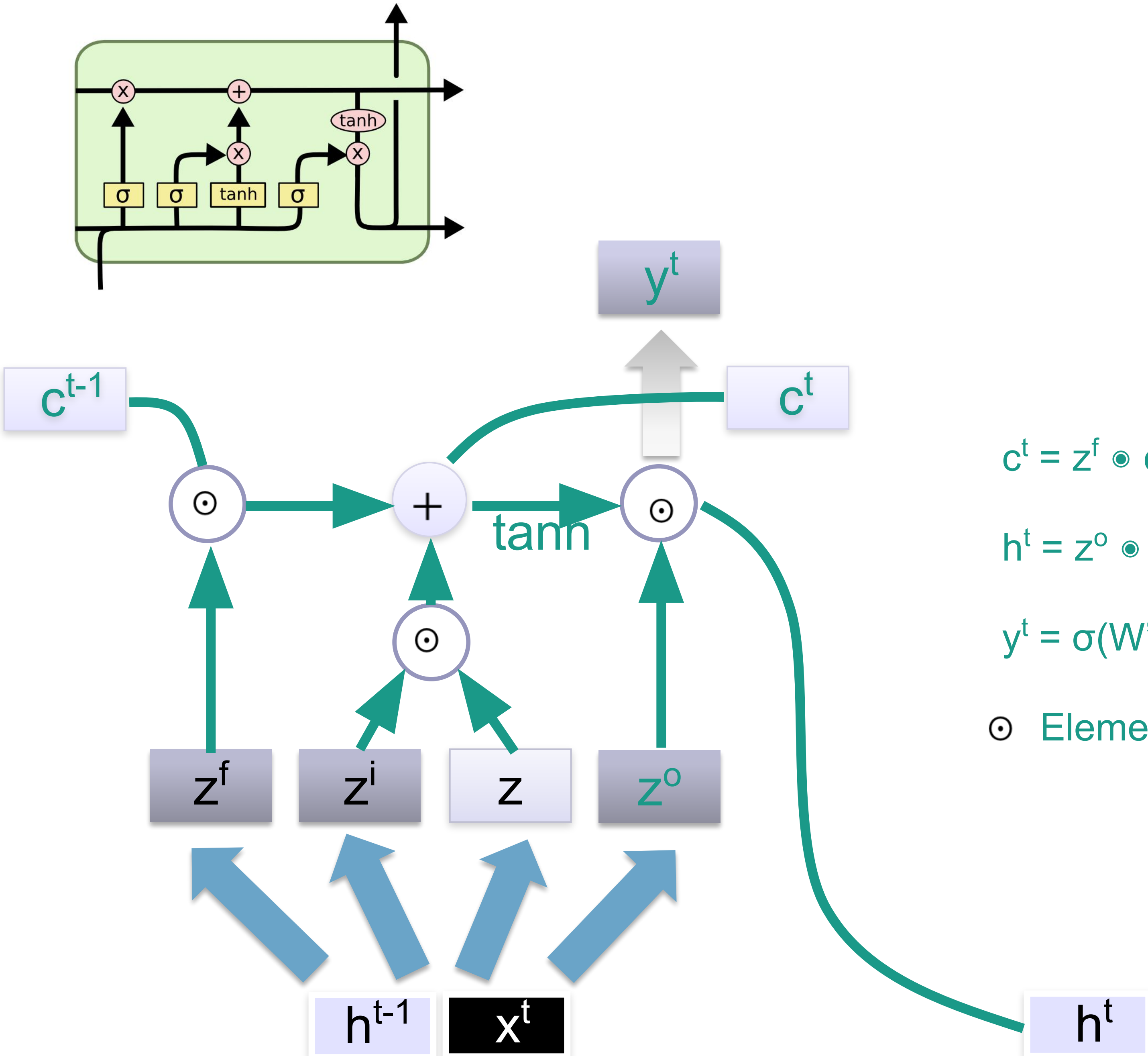
if 1 > sigmoid > 0, remember C^{t-1}

if sigmoid = 0, forget C^{t-1}

Input Gate

if 1 > sigmoid > 0, remember $\dot{C}_t$

if sigmoid = 0, forget $\dot{C}_t$

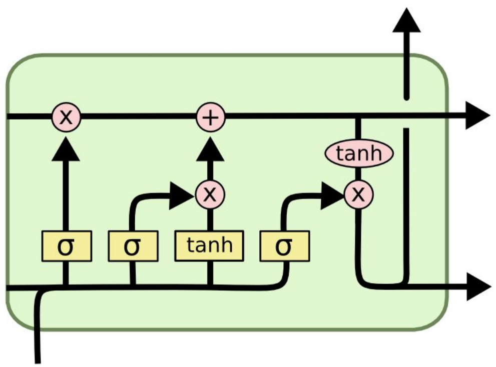


$c^t = z^f \odot c^{t-1} + z^i \odot z$

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

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

$\odot$ Element-wise multiply



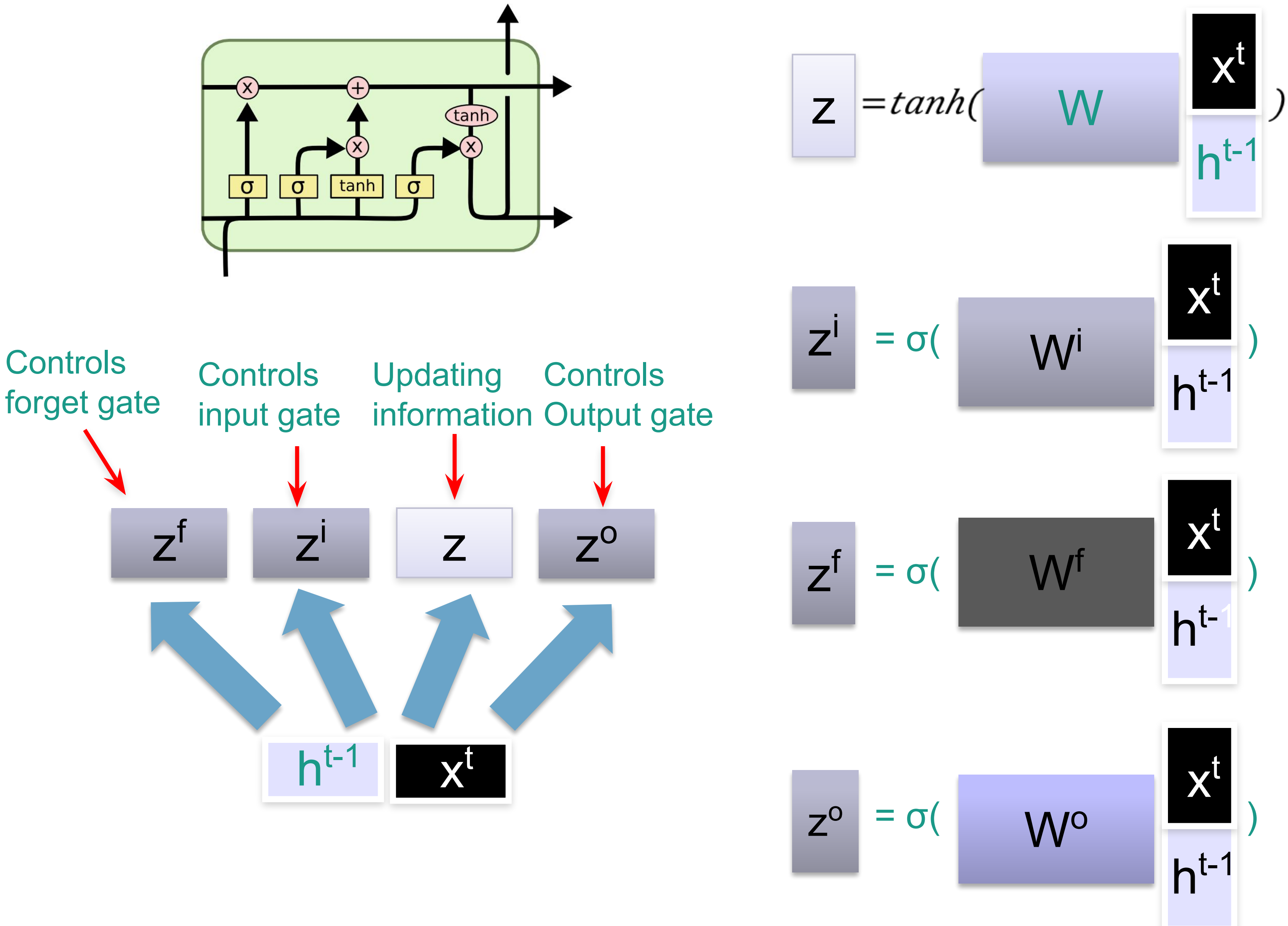
$$z = \left(\text{Transform} \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix} \right)$$

Transform = matrix

z^o z^f z^i obtained by the same way

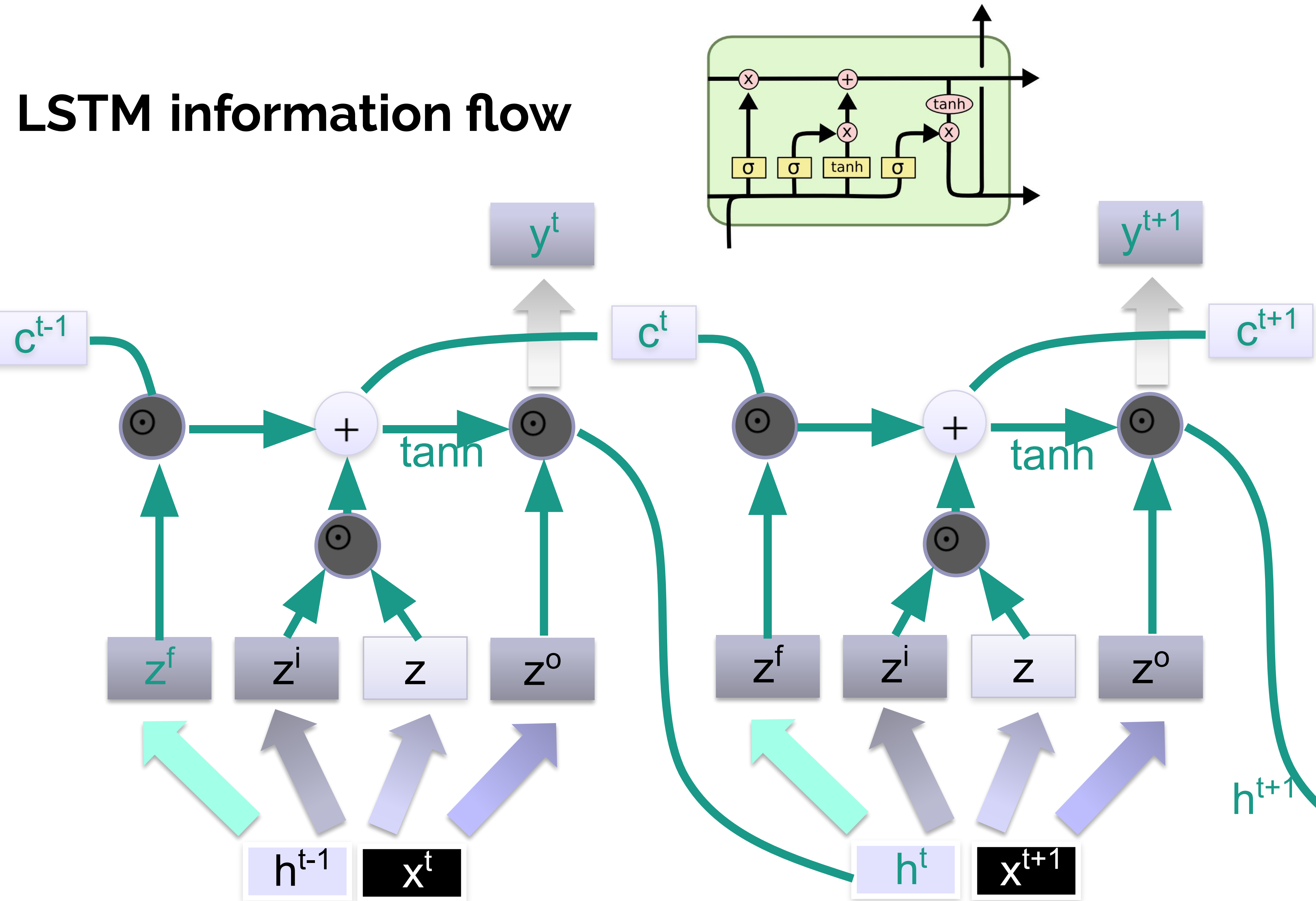
$\boxed{z^f \; z^i \; z \; z^o}$ ← Vector

h^{t-1} x^t



tanh
σ
σ
tanh
σ
Controls forget gate
Controls input gate
Updating information
Controls Output gate
z^f
z^i
z
z^o
h^{t-1}
x^t
z =tanh(W x^t h^{t-1})
z^i = σ(W^i x^t h^{t-1})
z^f = σ(W^f x^t h^{t-1})
z^o = σ(W^o x^t h^{t-1})

LSTM information flow



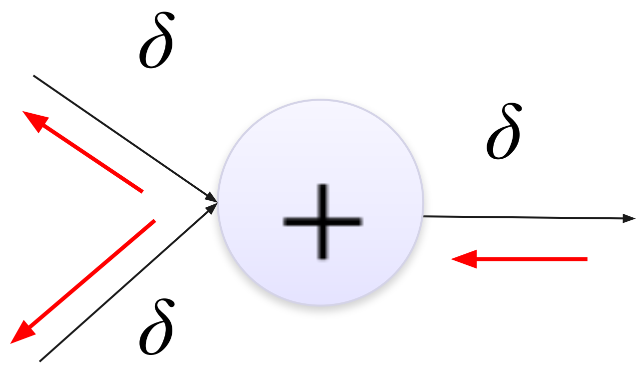
Constant Error Carousels

Gradient Vanishing / Exploding in RNN

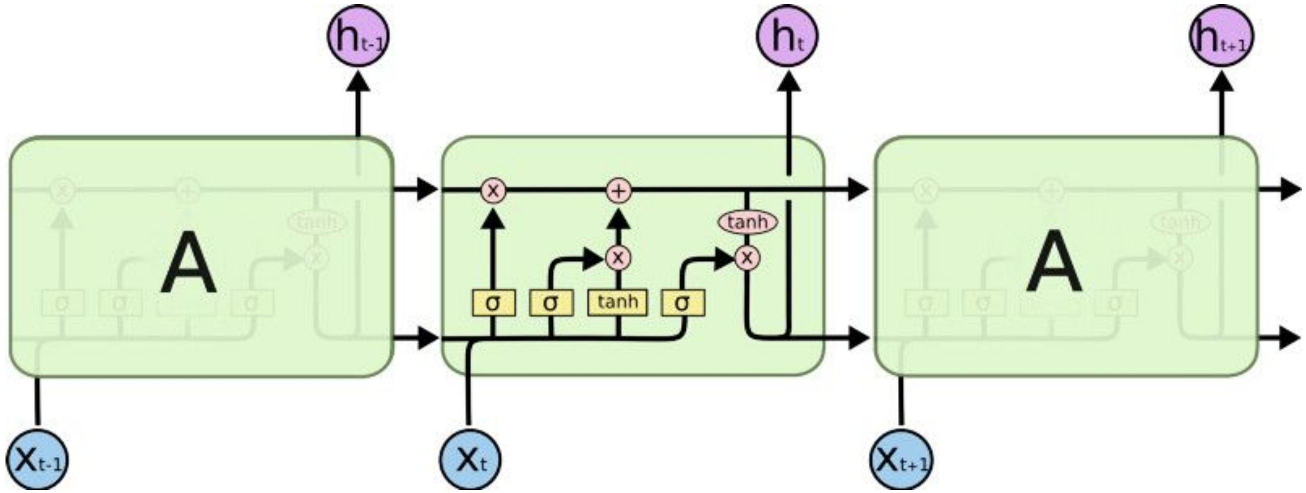
the repeated application of the recurrent weight matrix

Gradients Vanish in LSTM

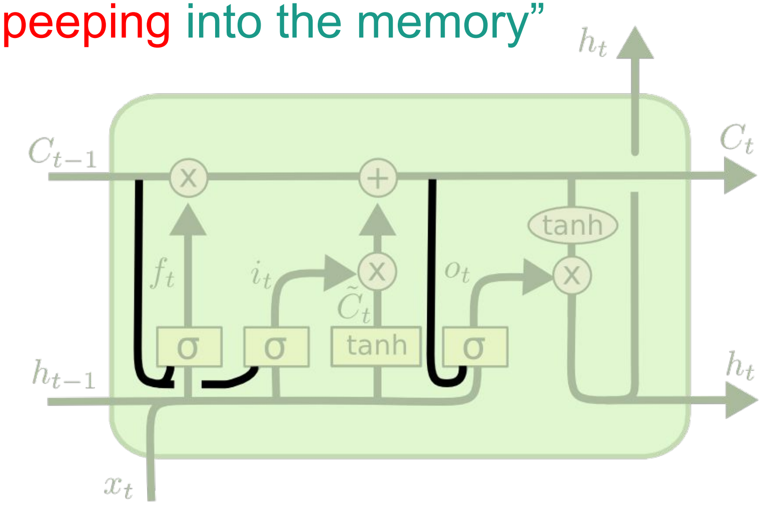
Only some of them, gradient information local in time will still be present.



Peephole LSTM



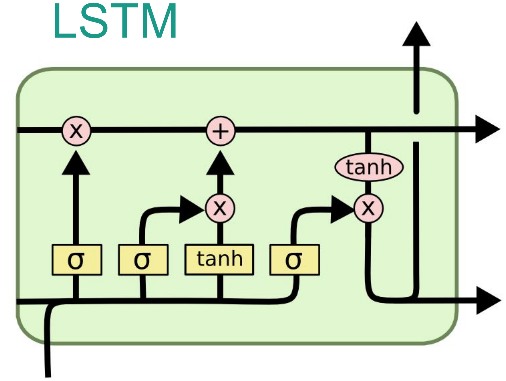
Allows “peeping into the memory”

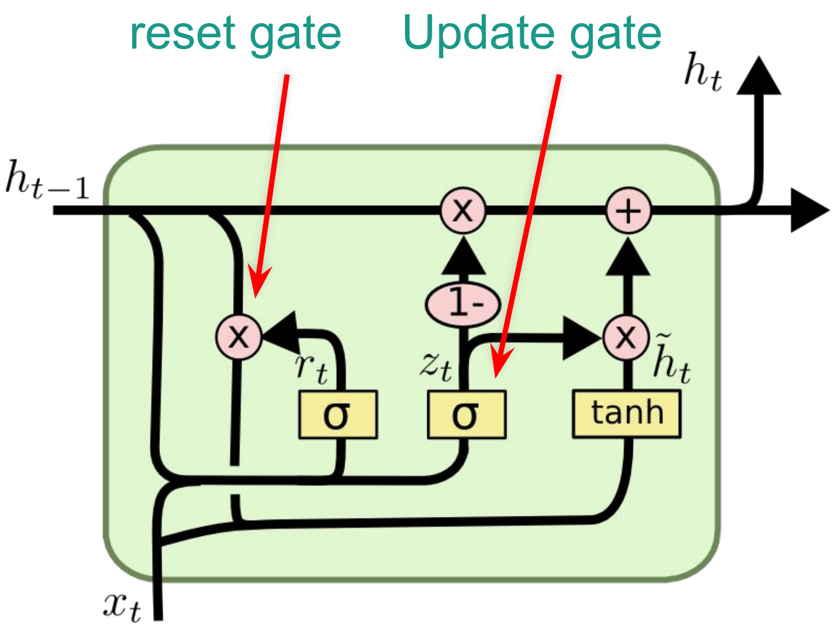


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

GRU – gated recurrent unit

(more compression)





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

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

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

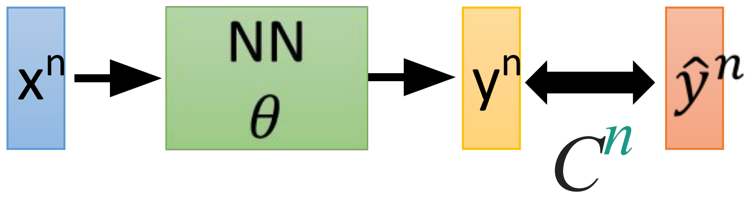
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

It combines the forget and input into a single update gate. It also merges the cell state and hidden state. This is simpler than LSTM. There are many other variants too.

X,*: element-wise multiply

BPTT

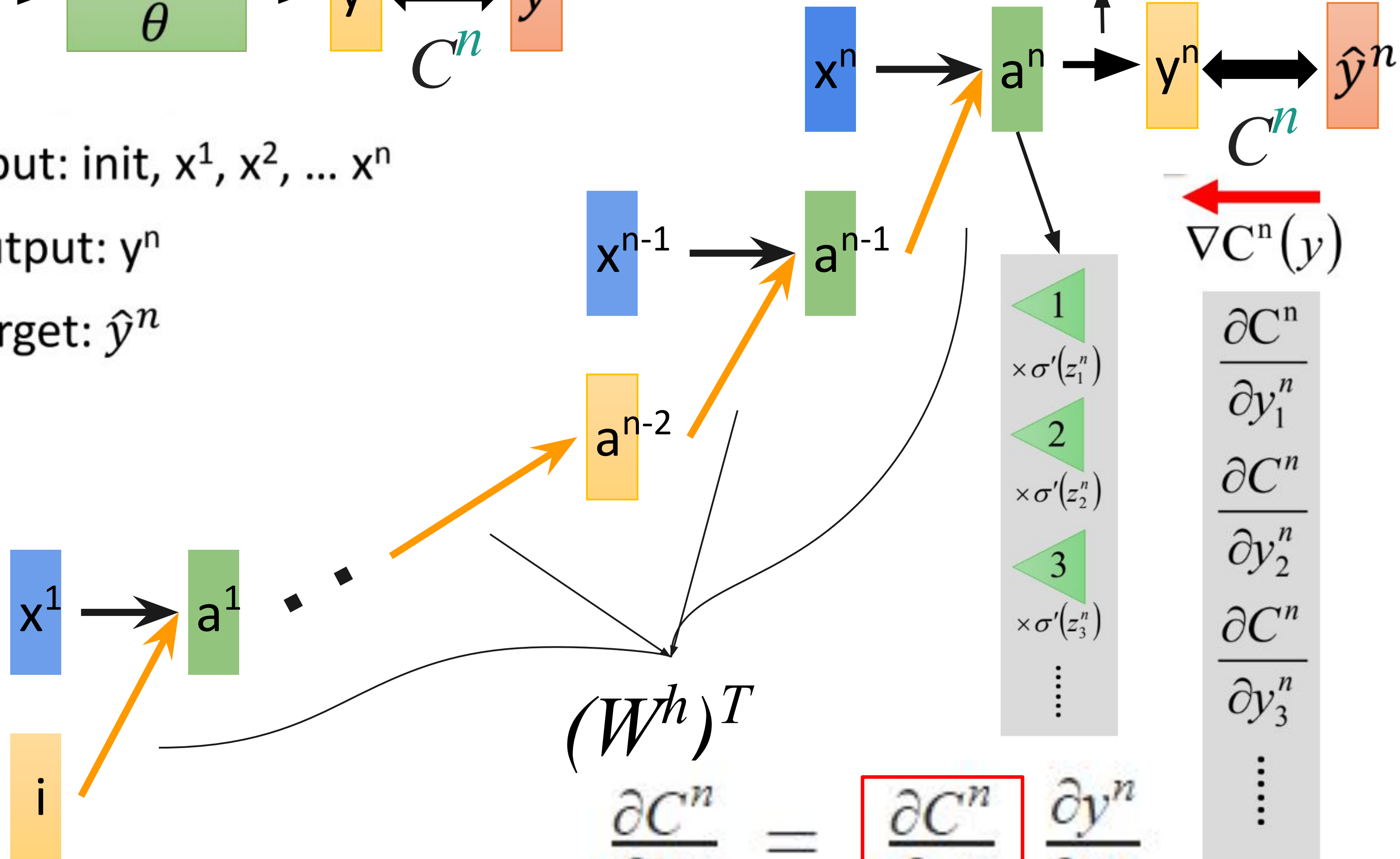
Backpropagation Through Time



Input: init, x^1, x^2, … x^n

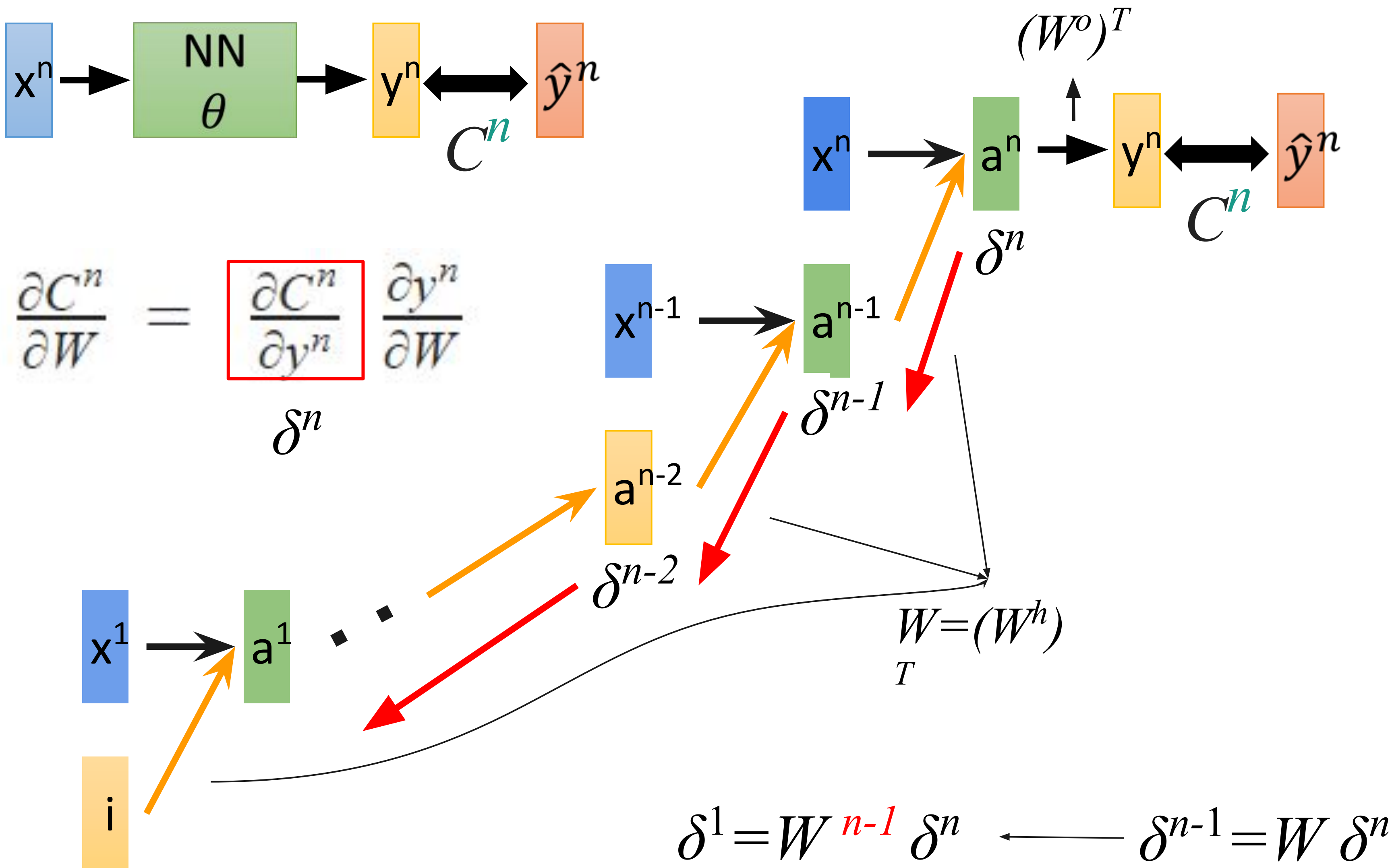
output: y^n

target: $\hat{y}^n$

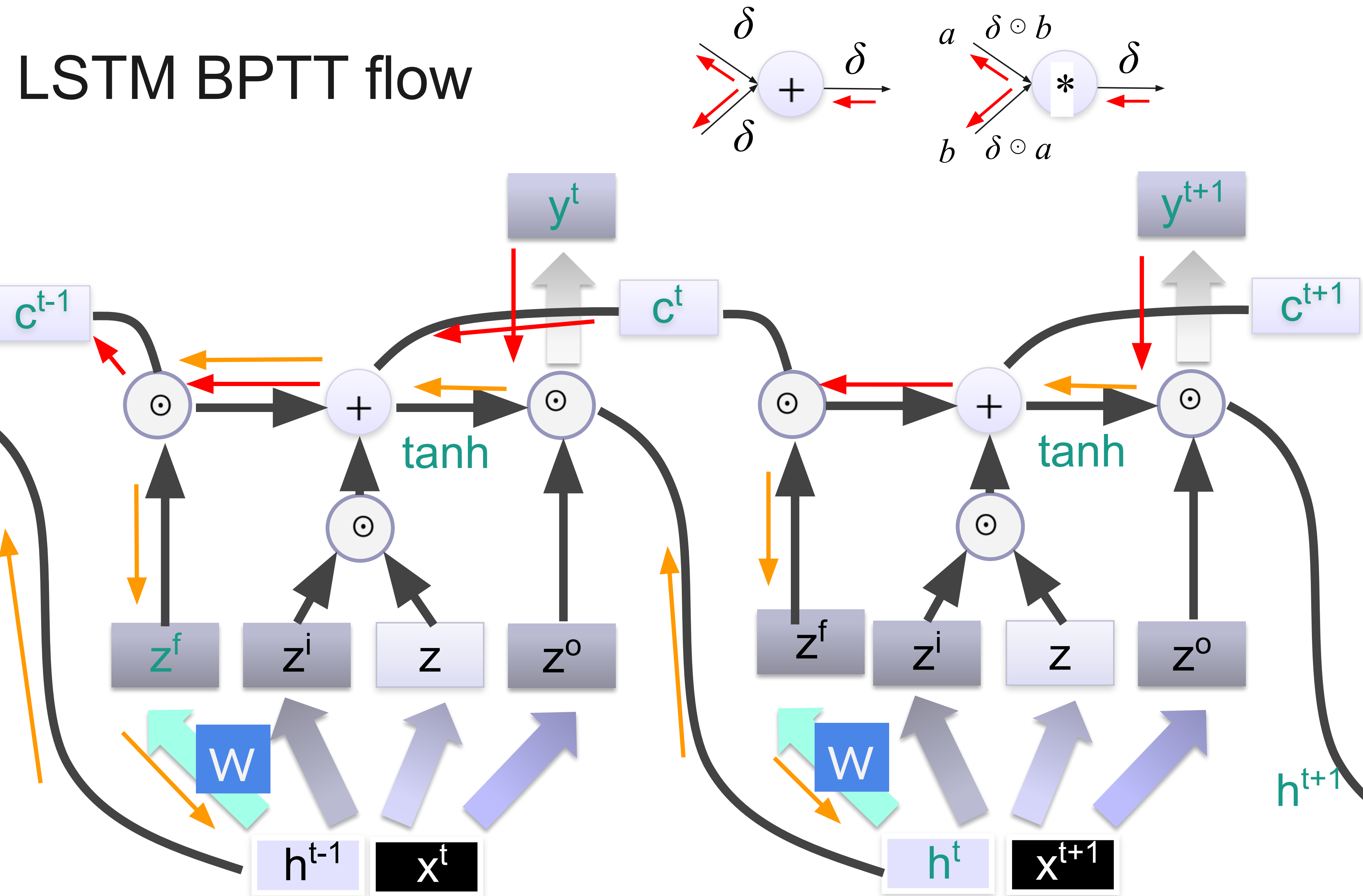


$$\frac{\partial C^n}{\partial W} = \boxed{\frac{\partial C^n}{\partial y^n}} \frac{\partial y^n}{\partial W}$$

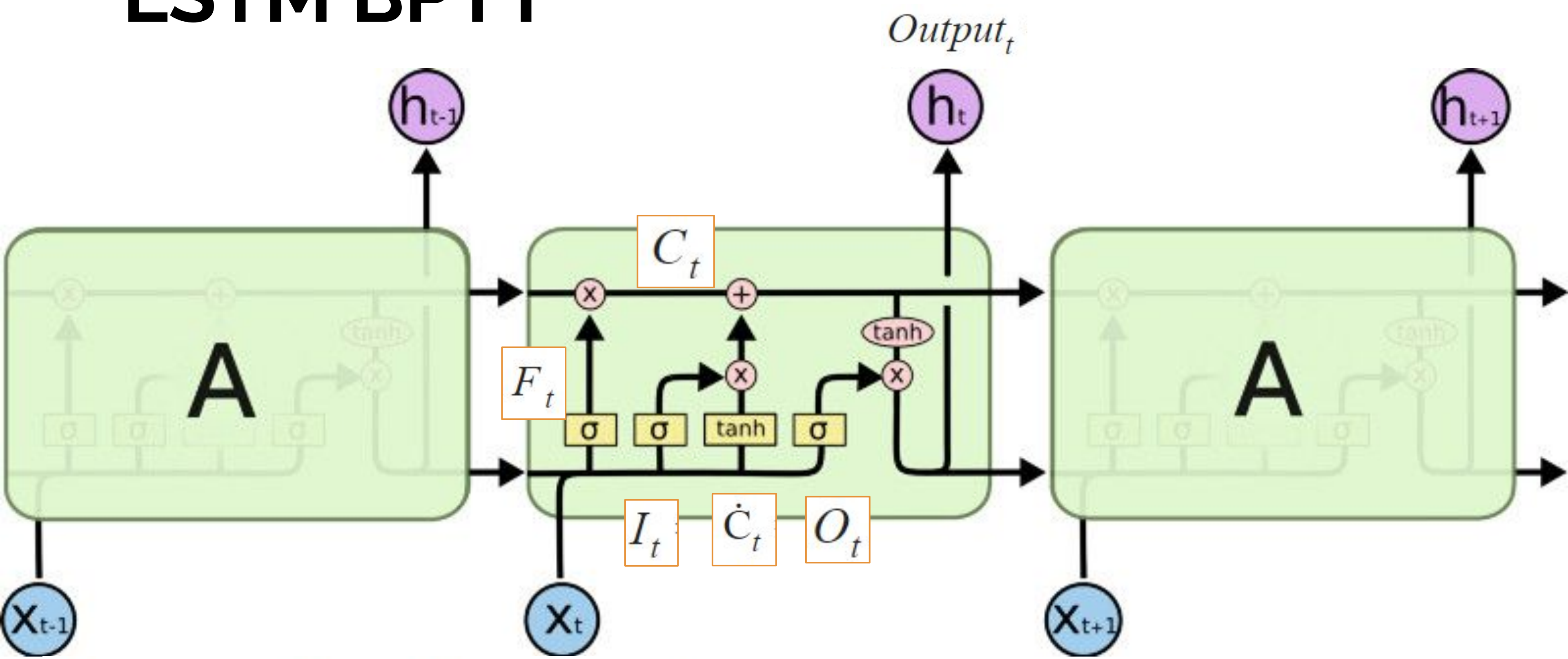
Gradient Exploding / Vanishing



LSTM BPTT flow



LSTM BPTT



$$F_t = \sigma(W_f X_t + W_{recf} h_{t-1} + b_f)$$

$$I_t = \sigma(W_i X_t + W_{reci} h_{t-1} + b_i)$$

$$O_t = \sigma(W_o X_t + W_{reco} h_{t-1} + b_o)$$

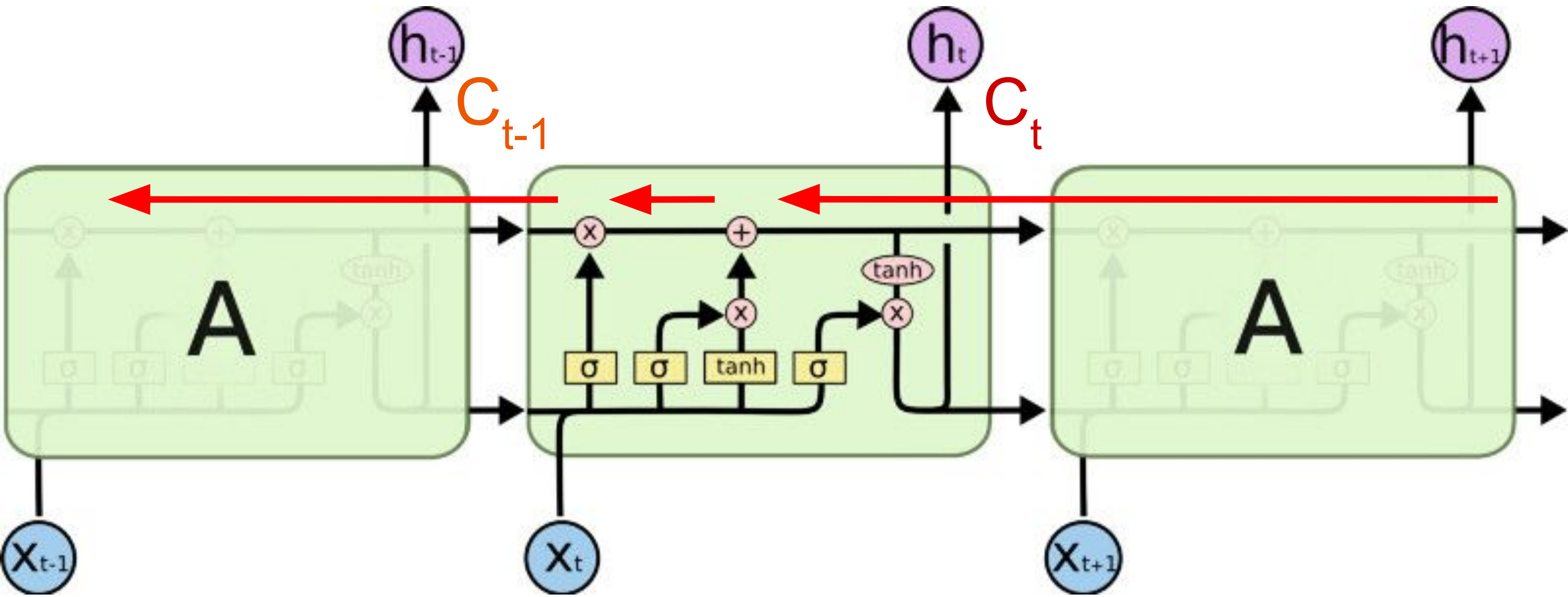
$$\dot{C}_t = tanh(W_{\dot{C}} X_t + W_{rec\dot{C}} h_{t-1} + b_{\dot{C}})$$

$$C_t = C_{t-1} F_t + I_t \dot{C}_t$$

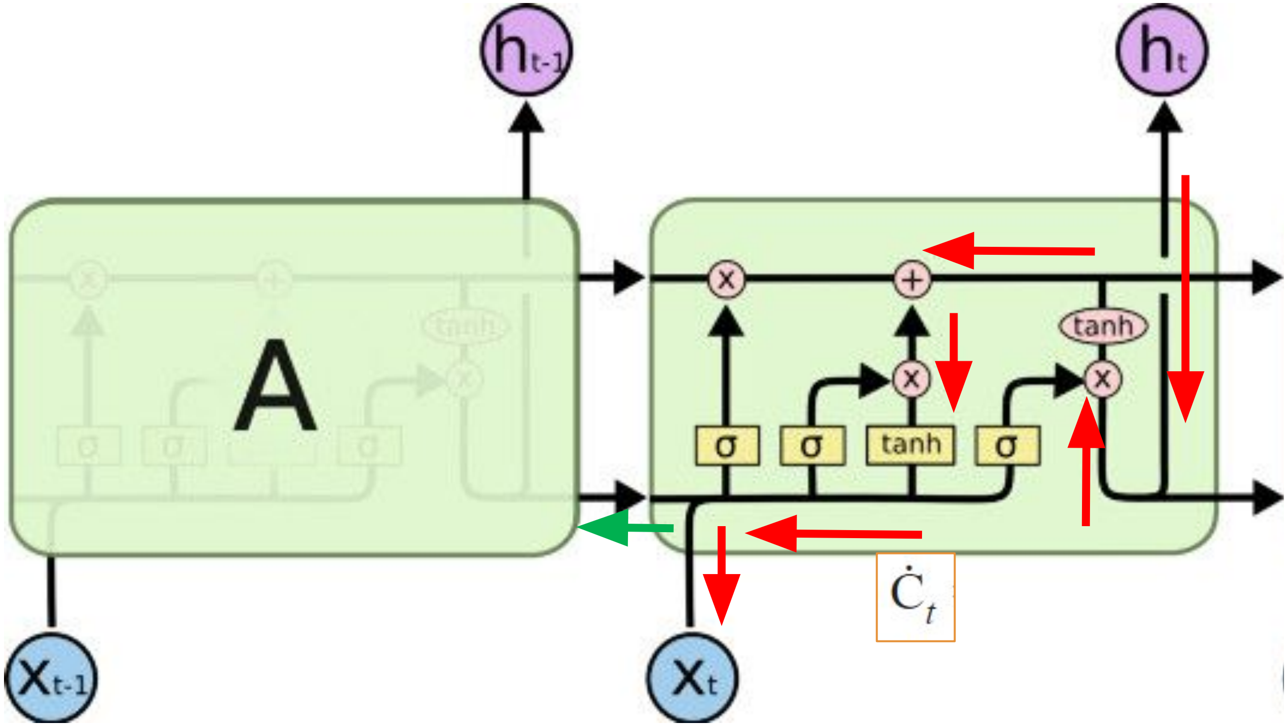
$$Output_t = O_t tanh(C_t)$$

$$Cost_t = \frac{1}{t} (Output_t - \widehat{y}_t)^2$$

Cell memory$_{(t=2)}$



Memory Backpropagation (t = 2)



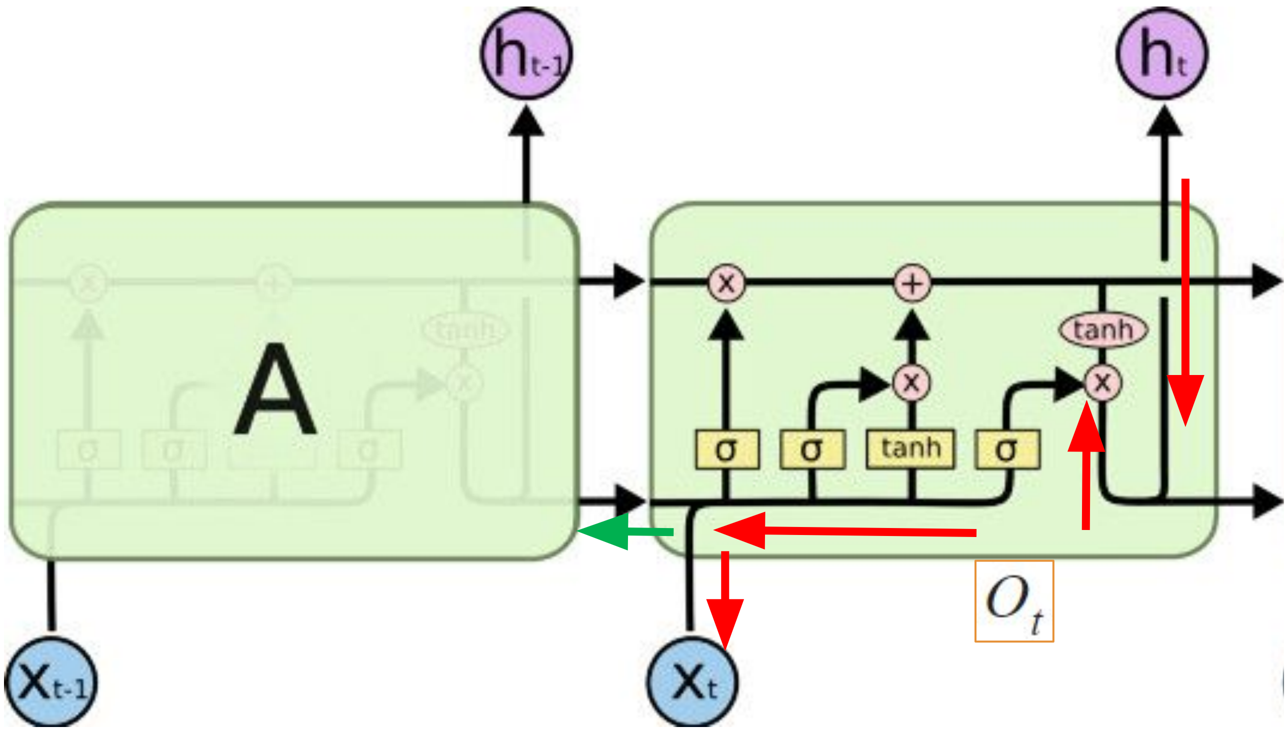
$$\frac{dCost_2}{dW_{\dot{C}}} = \frac{dCost_2}{dOutput_2}\ \frac{dOutput_2}{dC_2(out)}\ \frac{dC_2(out)}{dC_2(in)}\ \frac{dC_2(in)}{d\dot{C}_2(out)}\ \frac{d\dot{C}_2(out)}{d\dot{C}_2(in)}\ \frac{d\dot{C}_2(in)}{dW_{\dot{C}}}$$

$$= (Output_2 - \hat{y}_2)\ O_2\ (1 - tanh(C_2)^2)\ I_2(1 - tanh()^2)\ X_2$$

$$\frac{dCost_2}{dW_{rec\dot{C}}} = \frac{dCost_2}{dOutput_2}\ \frac{dOutput_2}{dC_2(out)}\ \frac{dC_2(out)}{dC_2(in)}\ \frac{dC_2(in)}{d\dot{C}_2(out)}\ \frac{d\dot{C}_2(out)}{d\dot{C}_2(in)}\ \frac{d\dot{C}_2(in)}{dW_{rec\dot{C}}}$$

$$= (Output_2 - \hat{y}_2)\ O_2\ (1 - tanh(C_2)^2)\ I_2(1 - tanh()^2)\ Output_1$$

Output Gate Backpropagation (t = 2)



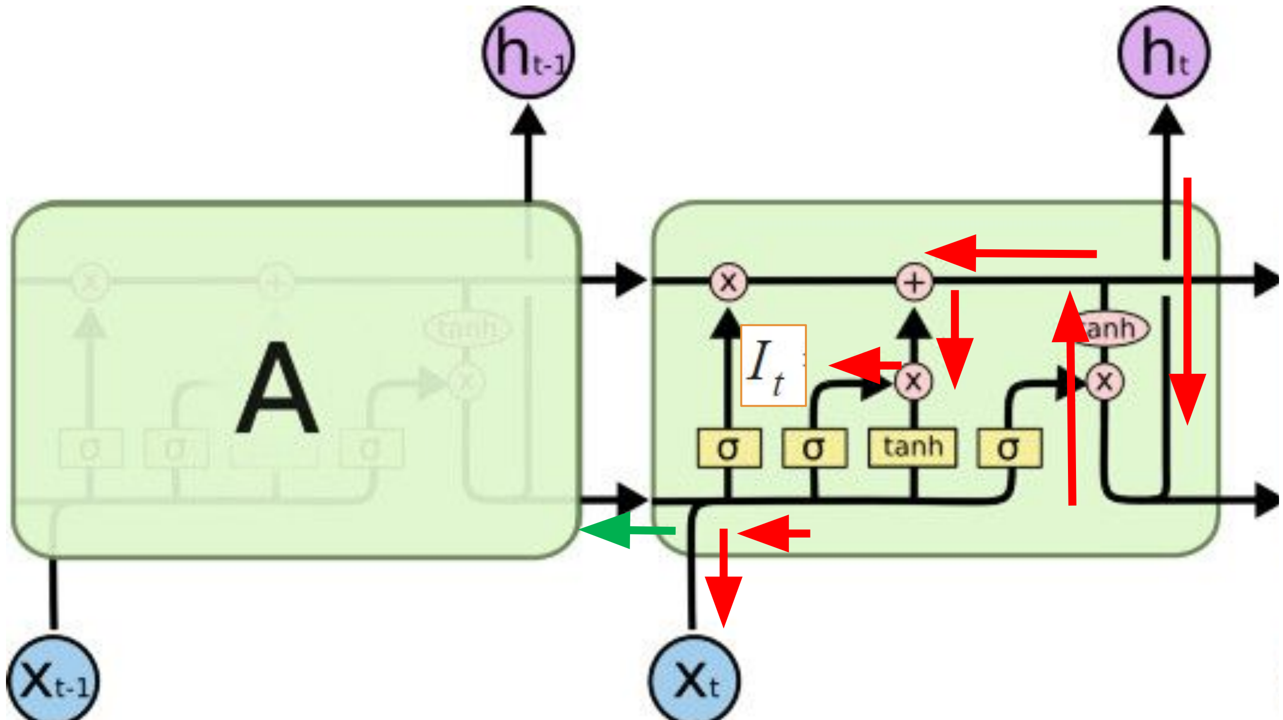
$$\frac{dCost_2}{dW_o} = \frac{dCost_2}{dOutput_2}\ \frac{dOutput_2}{dO_2(out)}\ \frac{dO_2(out)}{dO_2(in)}\ \frac{dO_2(in)}{dW_o}$$

$$= (Output_2 - \hat{y}_2)\ tanh(C_2)\ (\sigma()\ (1-\sigma()))\ X_2$$

$$\frac{dCost_2}{dW_{reco}} = \frac{dCost_2}{dOutput_2}\ \frac{dOutput_2}{dO_2(out)}\ \frac{dO_2(out)}{dO_2(in)}\ \frac{dO_2(in)}{dW_{reco}}$$

$$= (Output_2 - \hat{y}_2)\ tanh(C_2)\ (\sigma()\ (1-\sigma()))\ Output_1$$

Input Gate Backpropagation (t = 2)



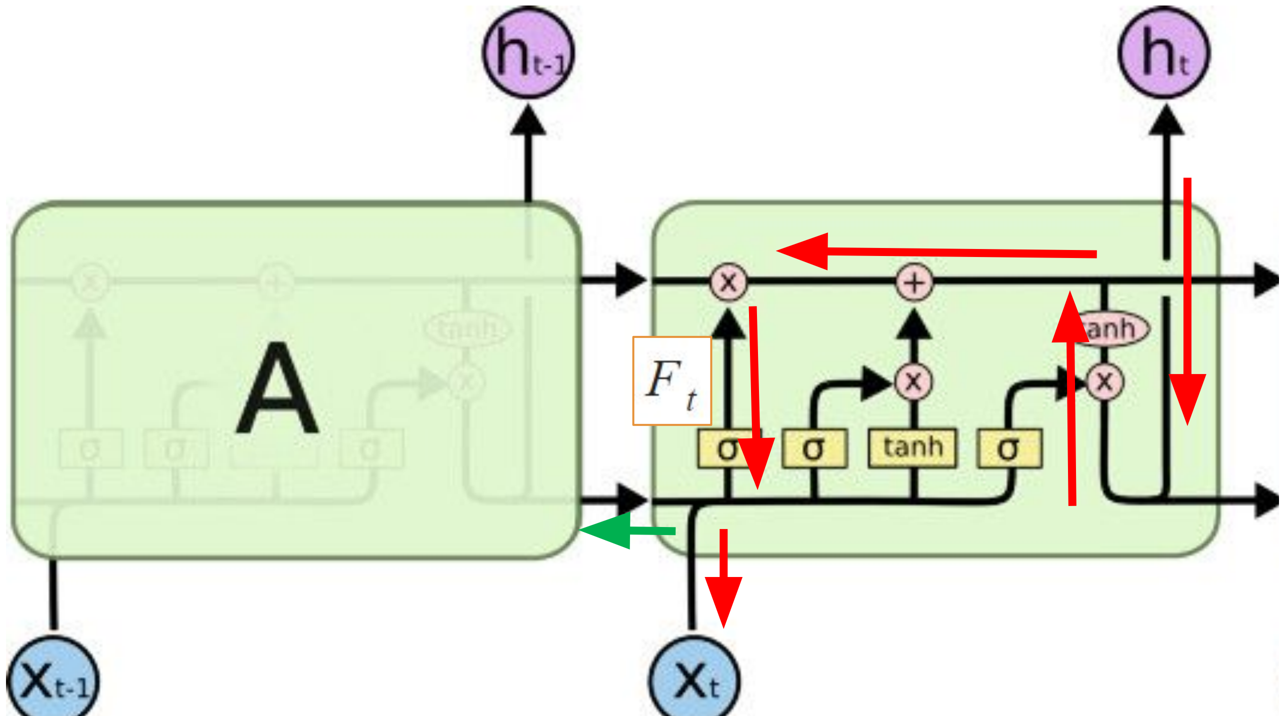
$$\frac{dCost_2}{dW_I} = \frac{dCost_2}{dOutput_2}\frac{dOutput_2}{dC_2(out)}\frac{dC_2(out)}{dC_2(in)}\frac{dC_2(in)}{dI_2(out)}\frac{dI_2(out)}{dI_2(in)}\frac{dI_2(in)}{dW_I}$$

$$= (Output_2 - \hat{y}_2)\ O_2\ (1 - tanh(C_2)^2)\ \dot{C}_2(\sigma()\ (1 - \sigma()))\ X_2$$

$$\frac{dCost_2}{dW_{recI}} = \frac{dCost_2}{dOutput_2}\frac{dOutput_2}{dC_2(out)}\frac{dC_2(out)}{dC_2(in)}\frac{dC_2(in)}{dI_2(out)}\frac{dI_2(out)}{dI_2(in)}\frac{dI_2(in)}{dW_{recI}}$$

$$= (Output_2 - \hat{y}_2)\ O_2\ (1 - tanh(C_2)^2)\ \dot{C}_2(\sigma()\ (1 - \sigma()))Output_1$$

Forget Gate Backpropagation (t = 2)



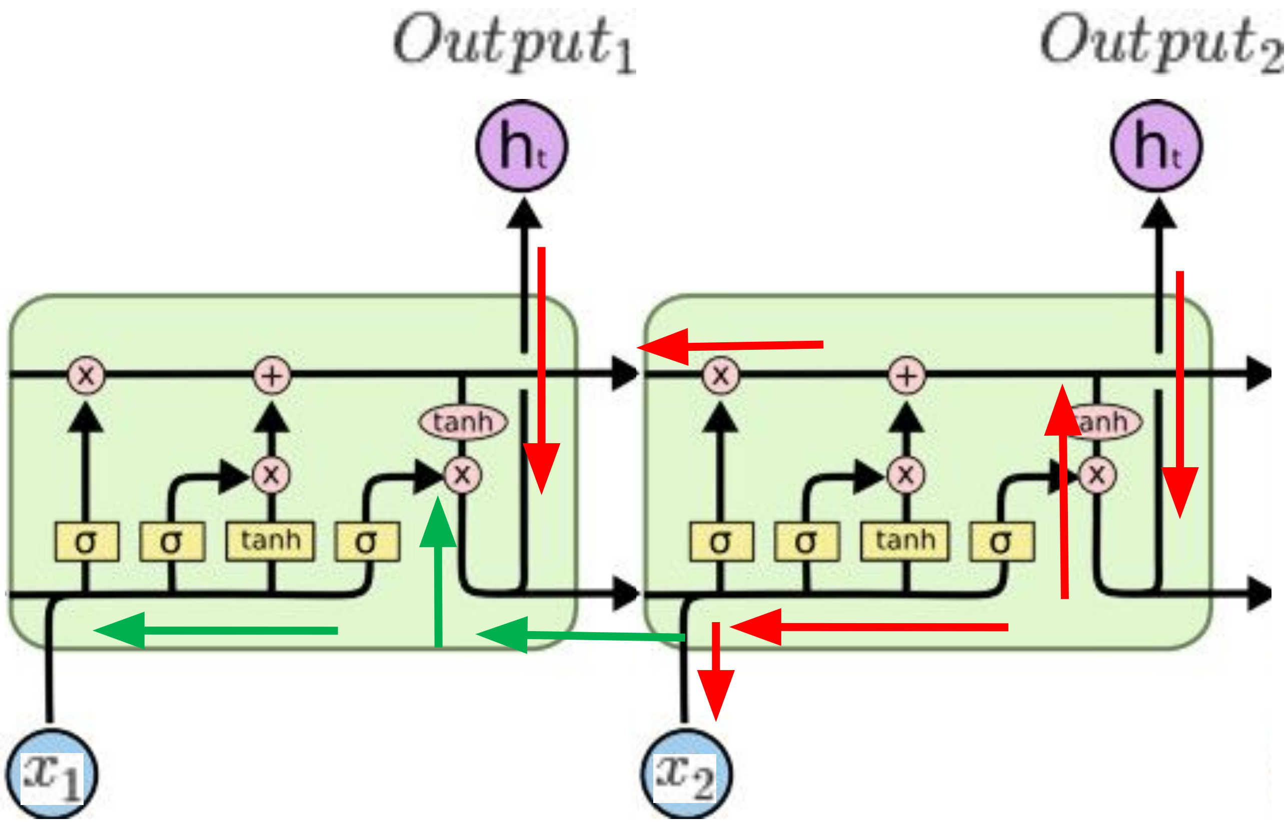
$$\frac{dCost_2}{dW_F} = \frac{dCost_2}{dOutput_2}\frac{dOutput_2}{dC_2(out)}\frac{dC_2(out)}{dC_2(in)}\frac{dC_2(in)}{dF_2(out)}\frac{dF(out)}{dF_2(in)}\frac{dF_2(in)}{dW_F}$$

$$= (Output_2 - \hat{y}_2)\ O_2\ (1 - tanh(C_2)^2)\ C_1(\sigma()\ (1 - \sigma()))\ X_2$$

$$\frac{dCost_2}{dW_{recF}} = \frac{dCost_2}{dOutput_2}\frac{dOutput_2}{dC_2(out)}\frac{dC_2(out)}{dC_2(in)}\frac{dC_2(in)}{dF_2(out)}\frac{dF(out)}{dF_2(in)}\frac{dF_2(in)}{dW_{recF}}$$

$$= (Output_2 - \hat{y}_2)\ O_2\ (1 - tanh(C_2)^2)\ C_1(\sigma()\ (1 - \sigma()))\ Output_1$$

Backpropagation to Cell1 (t = 1)



Backpropagation to Cell1 Output Gate$_{(t=1)}$

$$\frac{dCost_{total}}{dW_o} = \frac{dCost_1}{dW_o} + \boxed{\frac{dCost_2}{dW_o}}$$

$$\boxed{\begin{aligned} &= \frac{dCost_2}{dOutput_1}\frac{dOutput_1}{dO_1(out)}\frac{dO_1(out)}{dO_1(in)}\frac{dO_1(in)}{dW_o} + \\ &\frac{dCost_2}{dOutput_2}\frac{dOutput_2}{dO_2(out)}\frac{dO_2(out)}{dO_2(in)}\frac{dO_2(in)}{dOutput_1}\frac{dOutput_1}{dO_1(out)}\frac{dO_1(out)}{dO_1(in)}\frac{dO_1(in)}{dW_o} + \\ &\frac{dCost_2}{dOutput_2}\frac{dOutput_2}{dC_2(out)}\frac{dC_2(out)}{dC_2(in)}\frac{dC_2(in)}{dI_2(out)}\frac{dI_2(out)}{dI_2(in)}\frac{dI_2(in)}{dOutput_1}\frac{dOutput_1}{dO_1(out)}\frac{dO_1(out)}{dO_1(in)}\frac{dO_1(in)}{dW_o} + \\ &\frac{dCost_2}{dOutput_2}\frac{dOutput_2}{dC_2(out)}\frac{dC_2(out)}{dC_2(in)}\frac{dC_2(in)}{d\dot{C}_2(out)}\frac{d\dot{C}_2(out)}{d\dot{C}_2(in)}\frac{d\dot{C}_2(in)}{dOutput_1}\frac{dOutput_1}{dO_1(out)}\frac{dO_1(out)}{dO_1(in)}\frac{dO_1(in)}{dW_o} + \\ &\frac{dCost_2}{dOutput_2}\frac{dOutput_2}{dC_2(out)}\frac{dC_2(out)}{dC_2(in)}\frac{dC_2(in)}{dF_2(out)}\frac{dF(out)}{dF_2(in)}\frac{dF_2(in)}{dOutput_1}\frac{dOutput_1}{dO_1(out)}\frac{dO_1(out)}{dO_1(in)}\frac{dO_1(in)}{dW_o} \end{aligned}}$$

Simplify Backpropagation(t = 1)

$$\frac{dCost_{total}}{dW_o} = \frac{dCost_1}{dW_o} + \boxed{\frac{dCost_2}{dW_o}}$$

$$= (Output_2 - \widehat{y}_1)\ tanh(C_1)\ (\sigma()\ (1 - \sigma()))\ X_1 +$$

$$\boxed{\begin{aligned}
&(Output_2 - \widehat{y}_2)\ tanh(C_2)\ (\sigma()\ (1 - \sigma()))\ \boxed{W_{recO}}\ tanh(C_1)(\sigma()\ (1 - \sigma()))\ X_1 + \\
&(Output_2 - \widehat{y}_2)\ O_2\ (1 - tanh(C_2)^2)\ I_2(1 - tanh()^2)\ \boxed{W_{rec\dot{C}}}\ tanh(C_1)(\sigma()\ (1 - \sigma()))\ X_1 + \\
&(Output_2 - \widehat{y}_2)\ O_2\ (1 - tanh(C_2)^2)\ \dot{C}_2(\sigma()\ (1 - \sigma()))\ \boxed{W_{recI}}\ tanh(C_1)(\sigma()\ (1 - \sigma()))\ X_1 + \\
&(Output_2 - \widehat{y}_2)\ O_2\ (1 - tanh(C_2)^2)\ C_1(\sigma()\ (1 - \sigma()))\ \boxed{W_{recF}}\ tanh(C_1)(\sigma()\ (1 - \sigma()))\ X_1
\end{aligned}}$$

Thank you