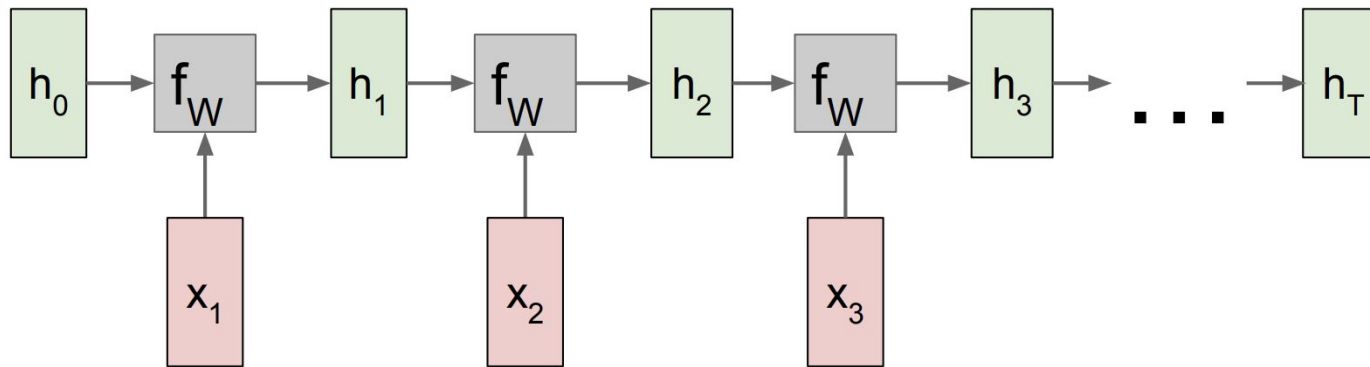


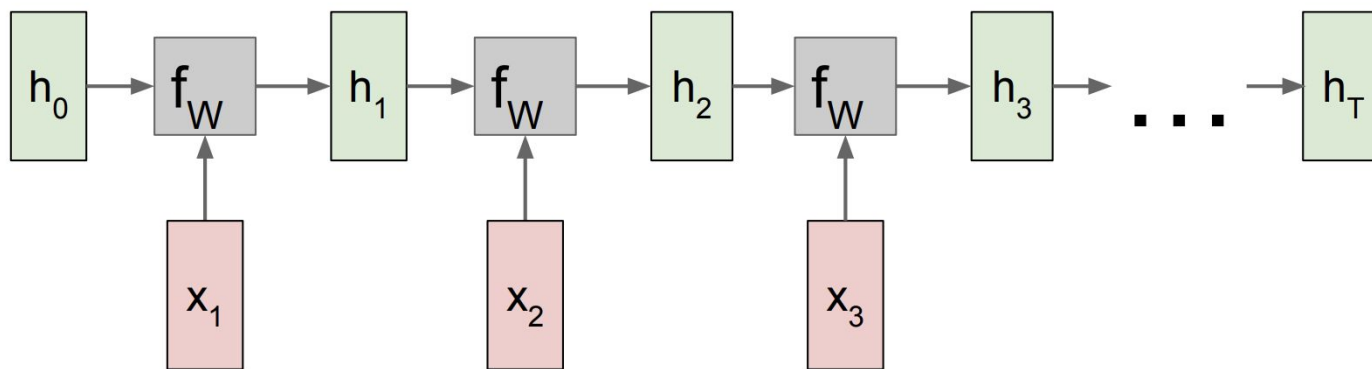
# Recurrent Neural Networks

# Recurrent Neural Network (Unrolled) Computational Graph



$$h_t = f_W(h_{t-1}, x_t)$$

# Recurrent Neural Network Computational Graph

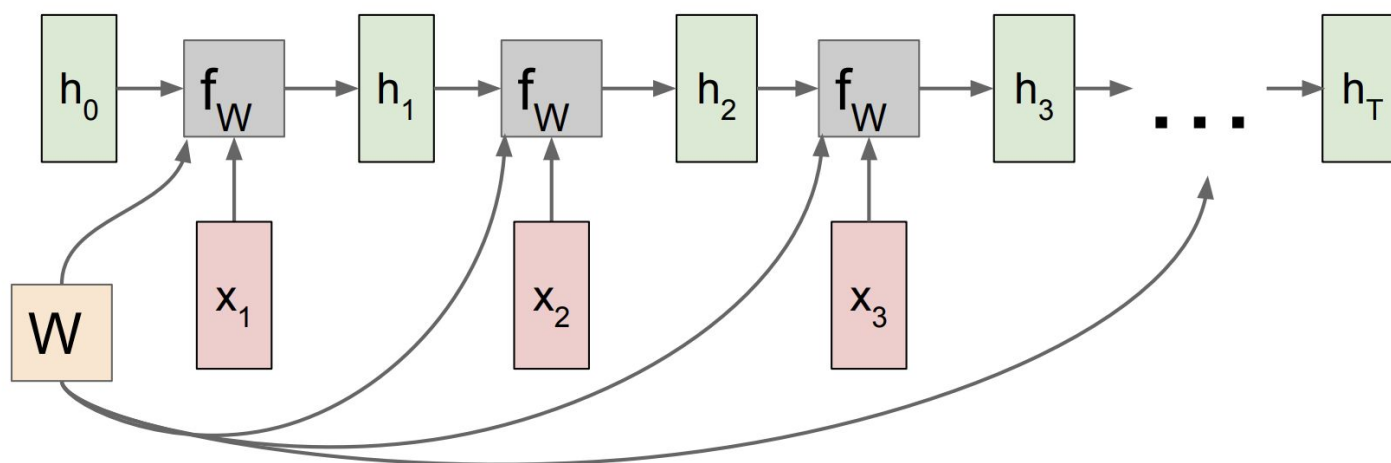


$$h_t = f_W(h_{t-1}, x_t) \longrightarrow h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

Vanilla Recurrent Neural Network

# Recurrent Neural Network Computational Graph

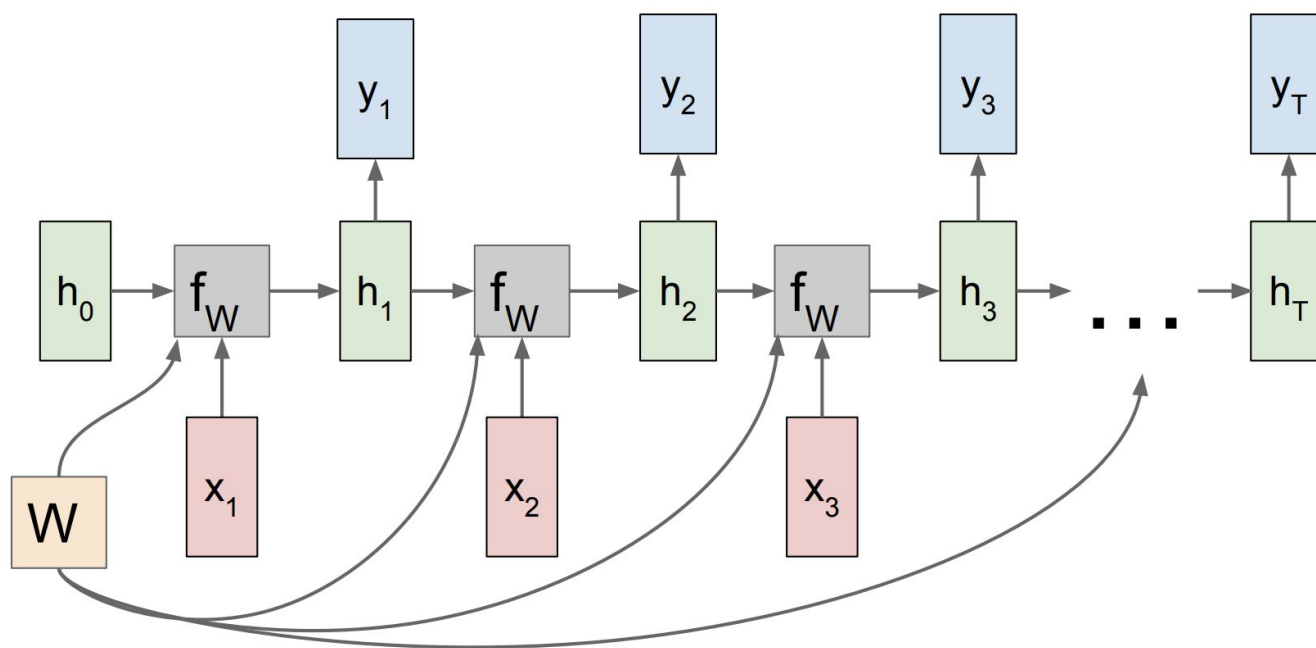
Re-use the same weight matrix at every time-step



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

Vanilla Recurrent Neural Network

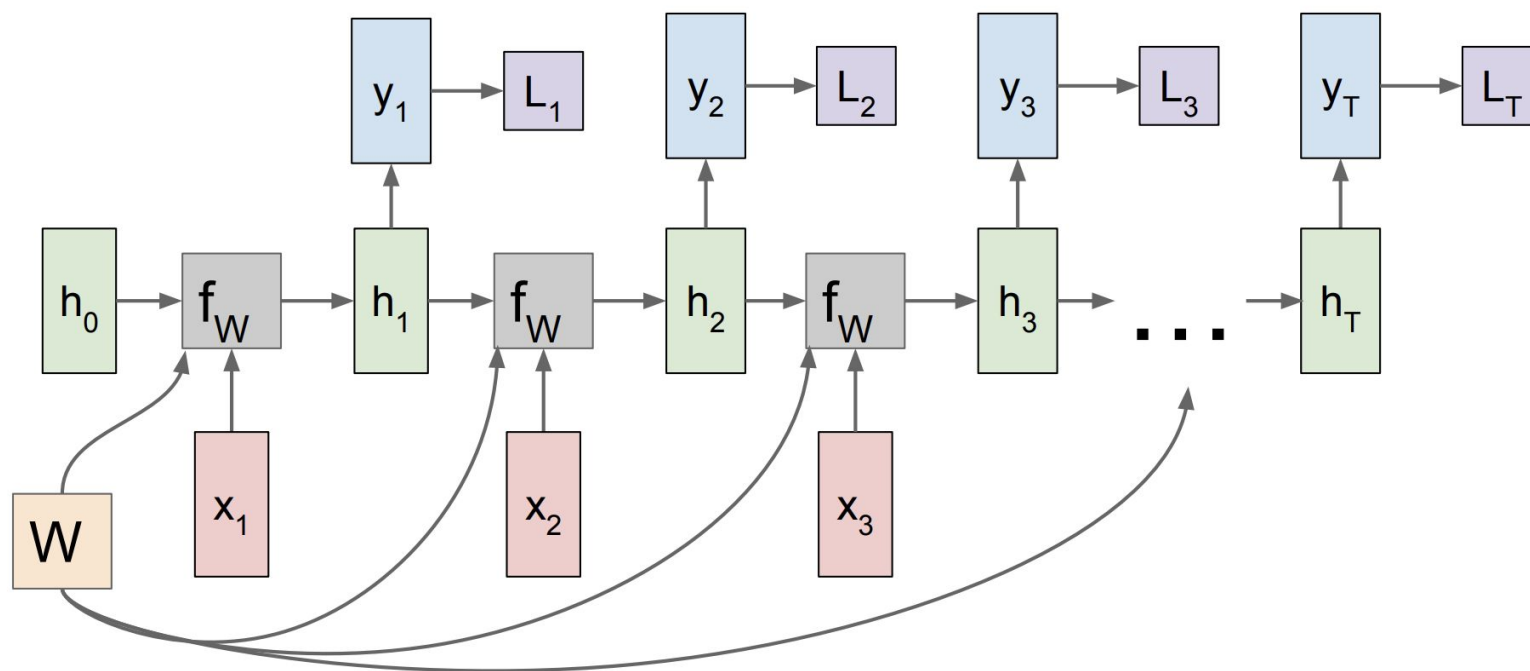
# Recurrent Neural Network Computational Graph



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

$$y_t = W_{hy}h_t$$

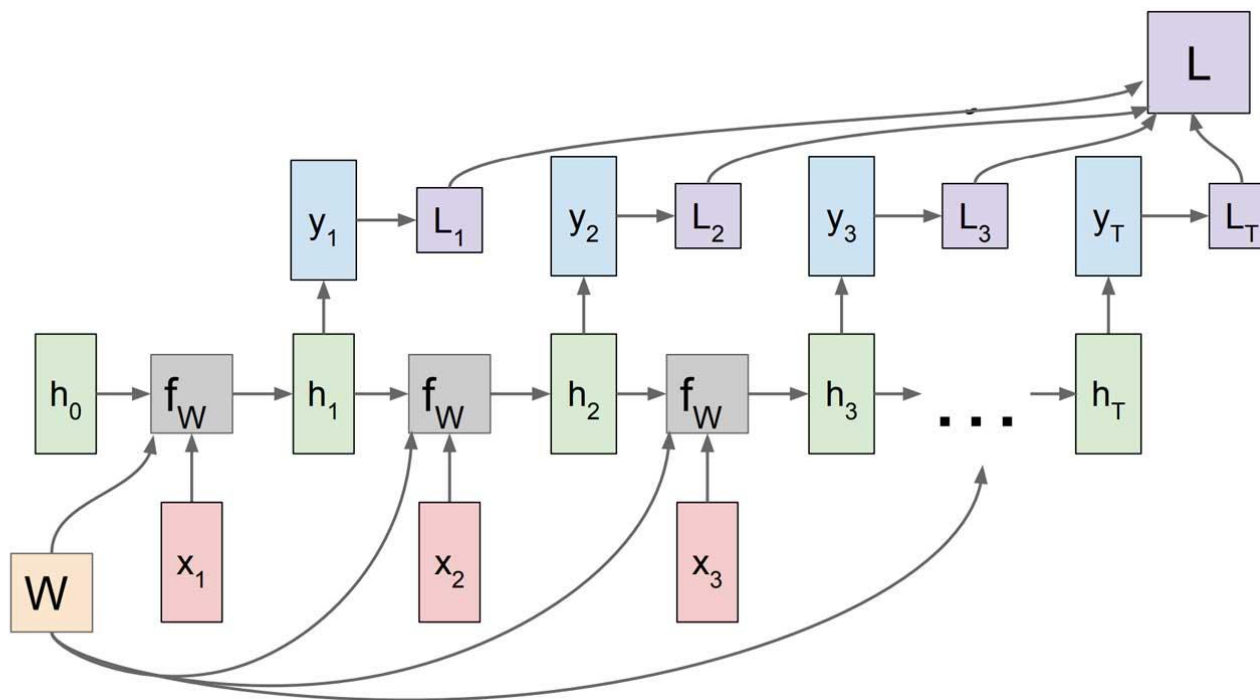
# Recurrent Neural Network Computational Graph



$$h_t = f_W(h_{t-1}, x_t) \longrightarrow h_t = \tanh(W_{hh}h_{t-1} + W_{hx}x_t)$$

$$y_t = W_{hy}h_t$$

# Recurrent Neural Network Computational Graph



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

$$y_t = W_{hy}h_t$$

# RNN - Forward Pass in Numpy!!

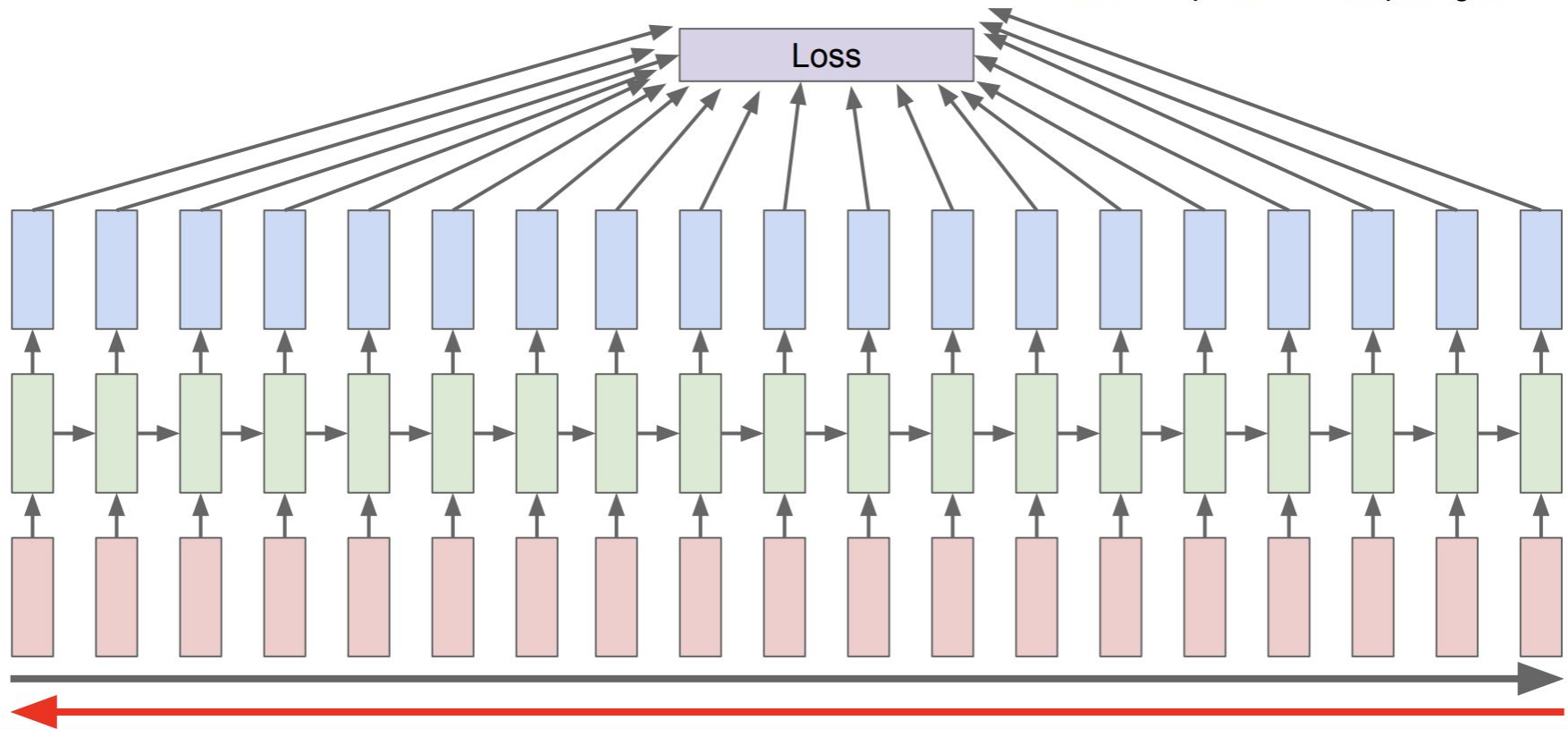
```
xs, hs, ys, ps = {}, {}, {}, {}  
hs[-1] = np.copy(hprev)  
loss = 0  
# forward pass  
for t in xrange(len(inputs)):  
    xs[t] = np.zeros((vocab_size,1)) # encode in 1-of-k representation  
    xs[t][inputs[t]] = 1  
    hs[t] = np.tanh(np.dot(Wxh, xs[t]) + np.dot(Whh, hs[t-1]) + bh) # hidden state  
    ys[t] = np.dot(Why, hs[t]) + by # unnormalized log probabilities for next chars  
    ps[t] = np.exp(ys[t]) / np.sum(np.exp(ys[t])) # probabilities for next chars  
    loss += -np.log(ps[t][targets[t],0]) # softmax (cross-entropy loss)
```

Source : <https://gist.github.com/karpathy/d4dee566867f8291f086>

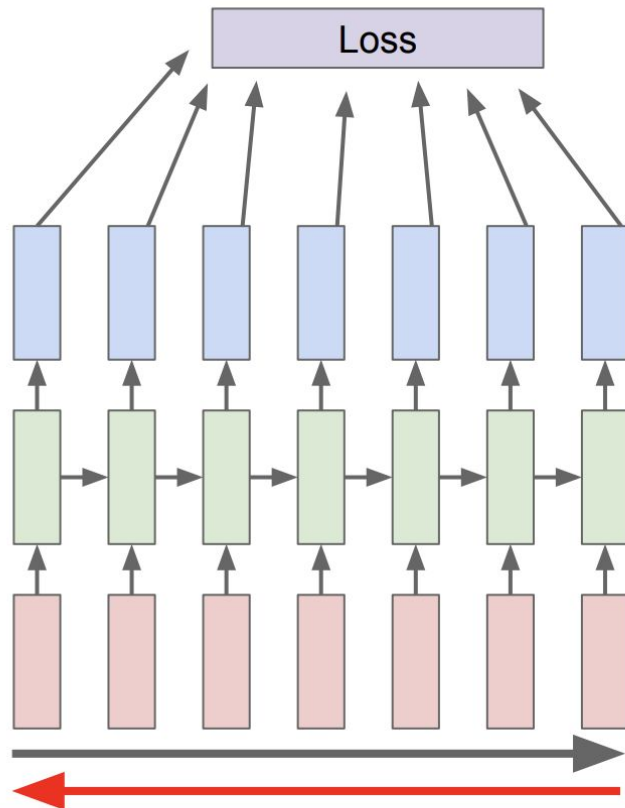


# Backpropagation through time

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

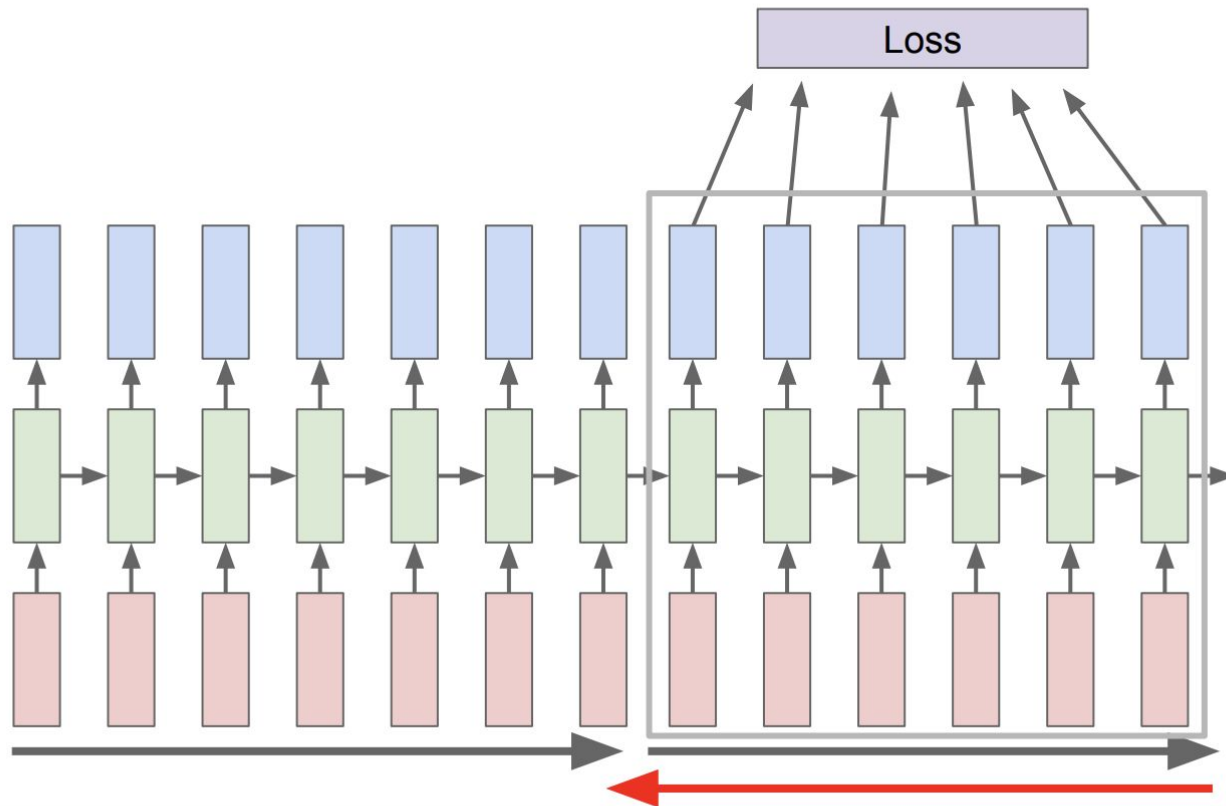


# Truncated Backpropagation through time



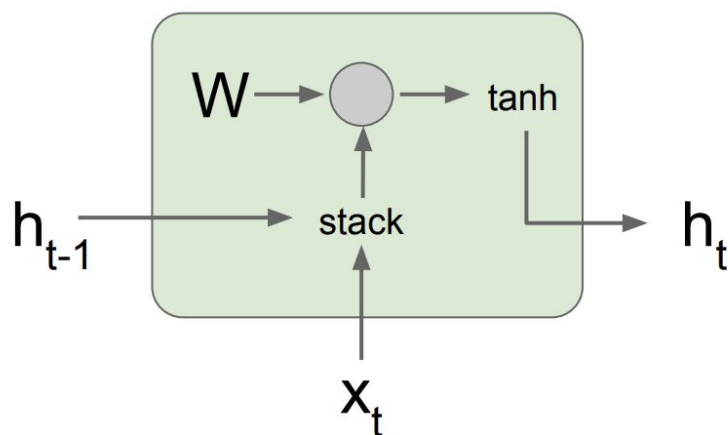
Run forward and backward through chunks of the sequence instead of whole sequence

# Truncated Backpropagation through time



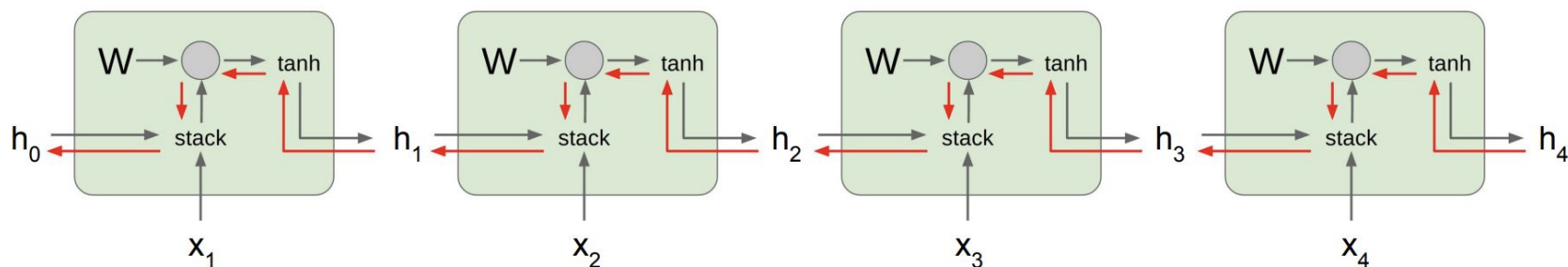
Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

# Moving Towards LSTMs



$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{hx}x_t) \\ &= \tanh\left((W_{hh} \quad W_{hx}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \\ &= \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right) \end{aligned}$$

# Vanishing/Exploding Gradients in Long Sequences



Computing gradient of  $h_0$  involves many factors of  $W$  (and repeated  $\tanh$ )

Largest singular value  $> 1$ :  
**Exploding gradients**

Largest singular value  $< 1$ :  
**Vanishing gradients**

## Vanilla RNN

$$h_t = \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

## LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

## Vanilla RNN

$$h_t = \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

$W : h \times 2h$

## LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$
$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$

## Vanilla RNN

$$h_t = \tanh \left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)$$

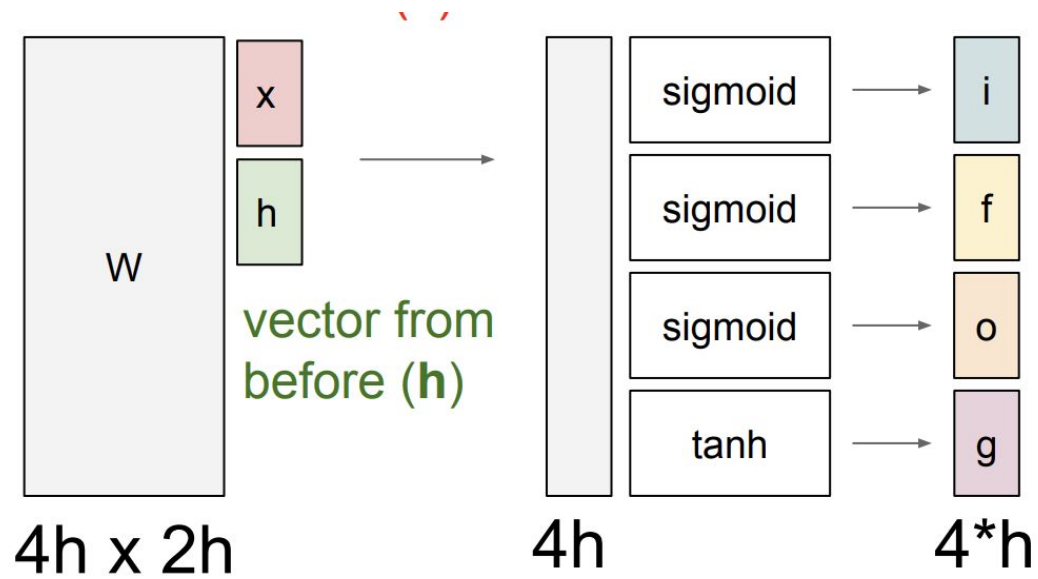
$W : h \times 2h$

## LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$
$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$

$W : 4h \times 2h$



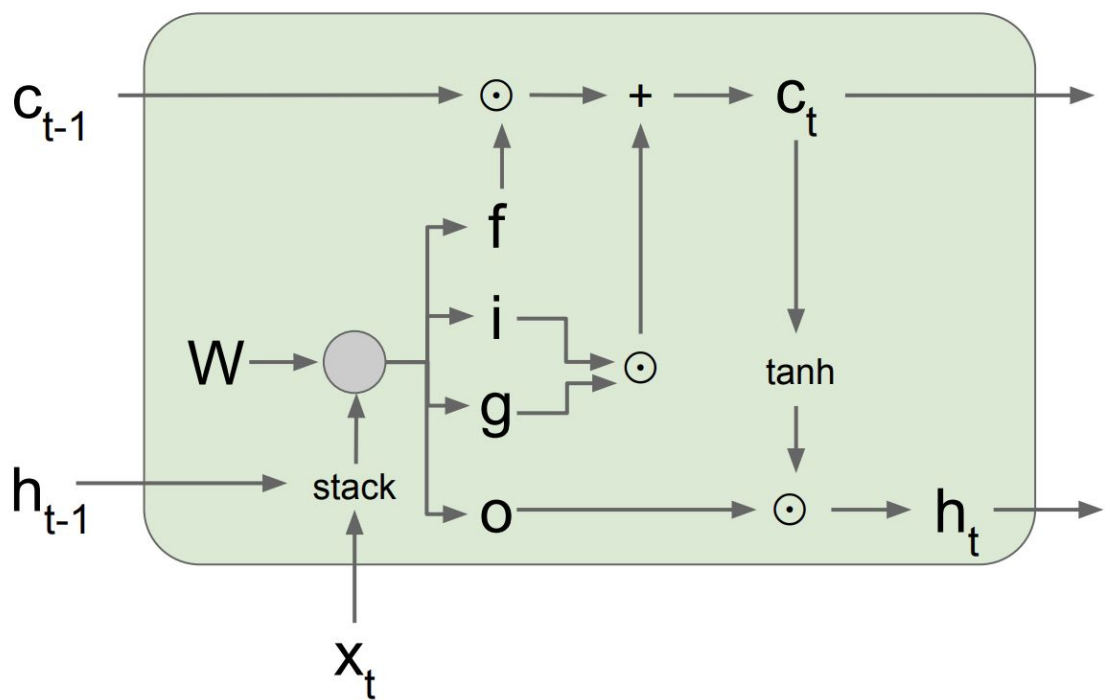


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

# What's going on?

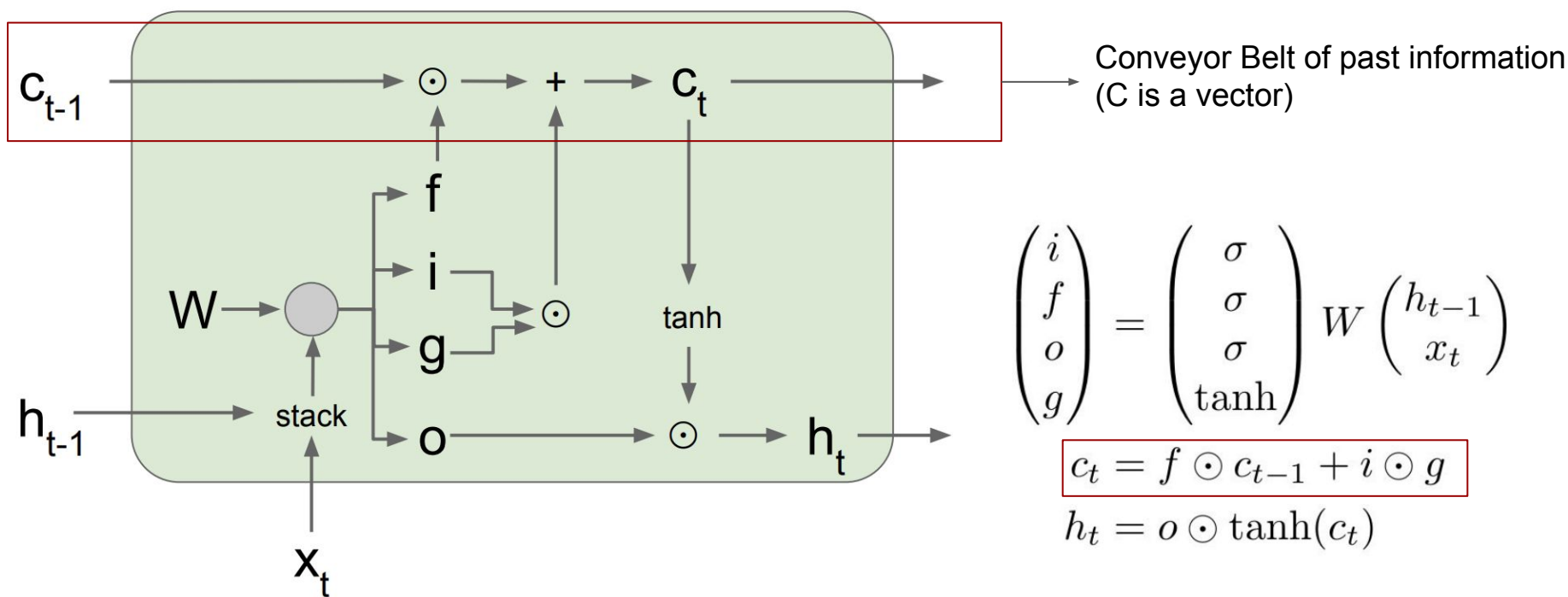


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

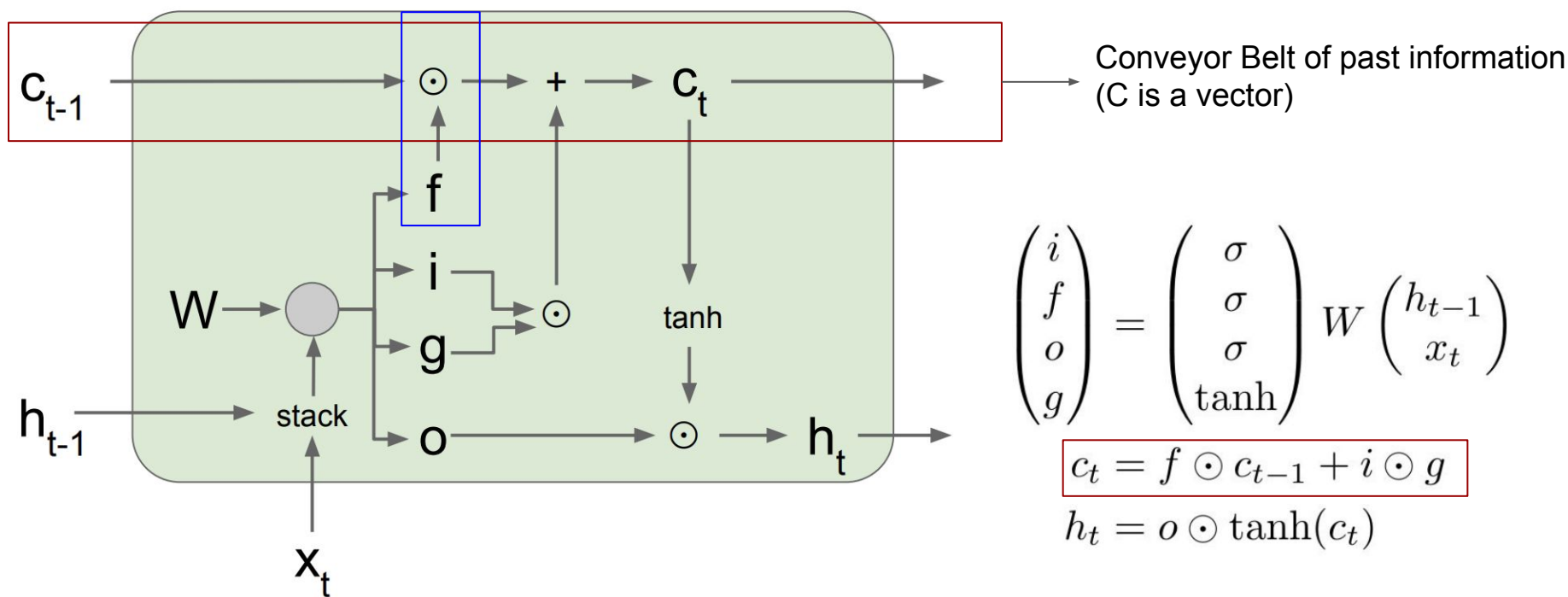
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

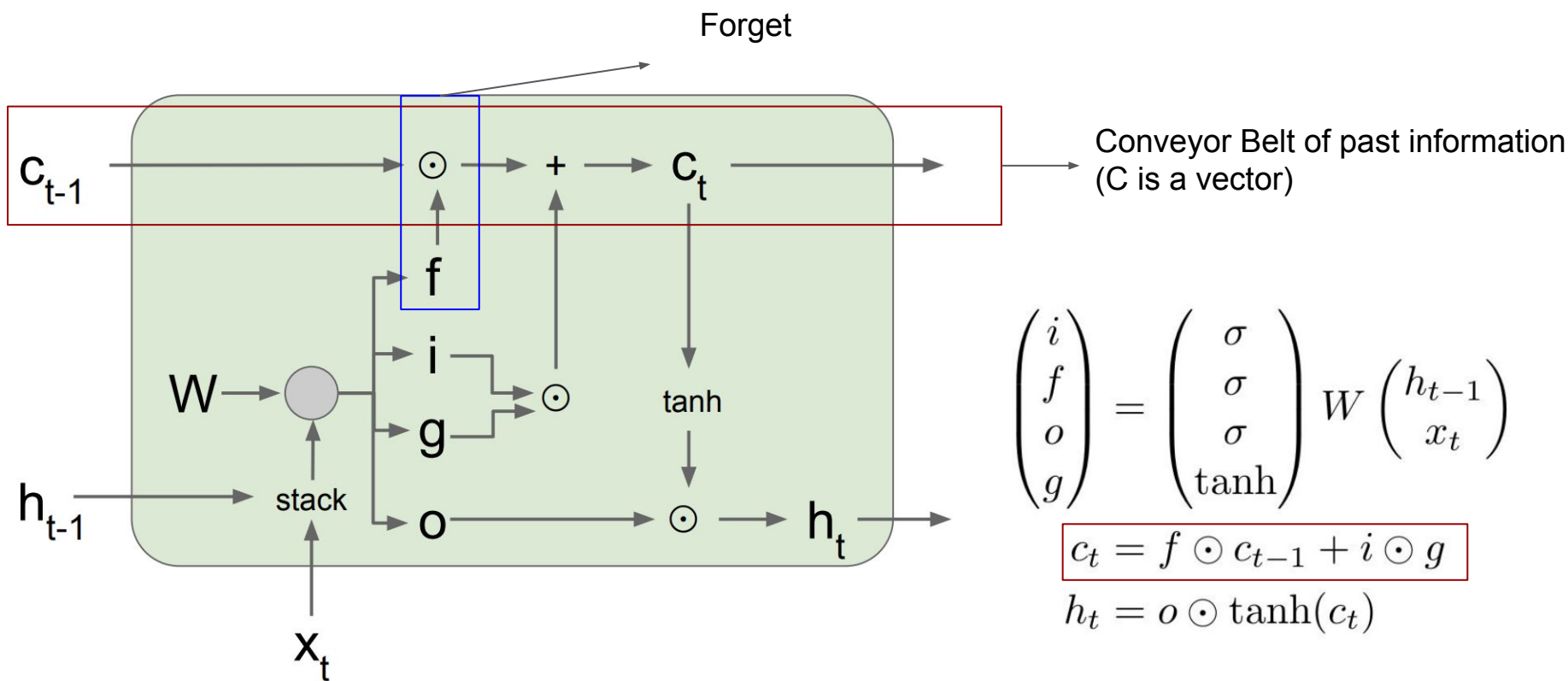
# What's going on?



# What's going on?

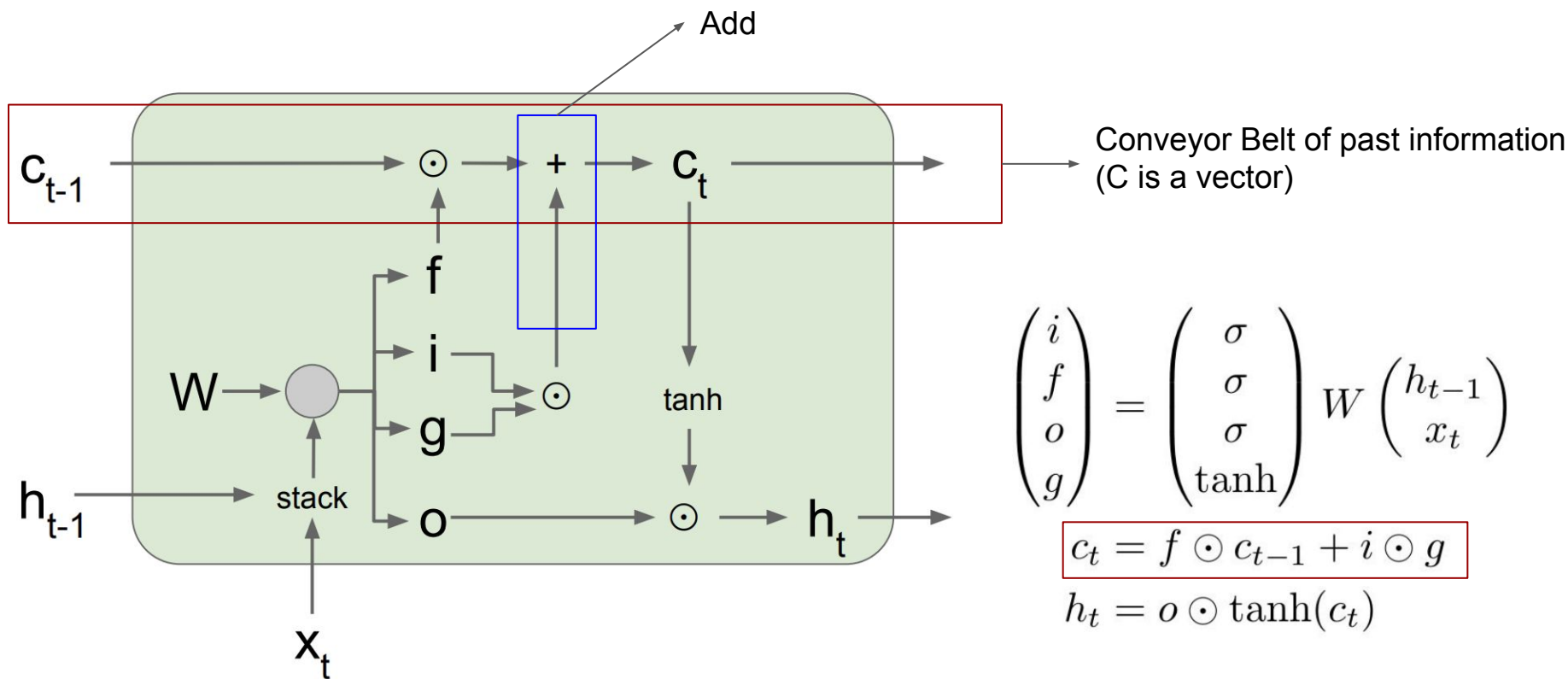


# What's going on?



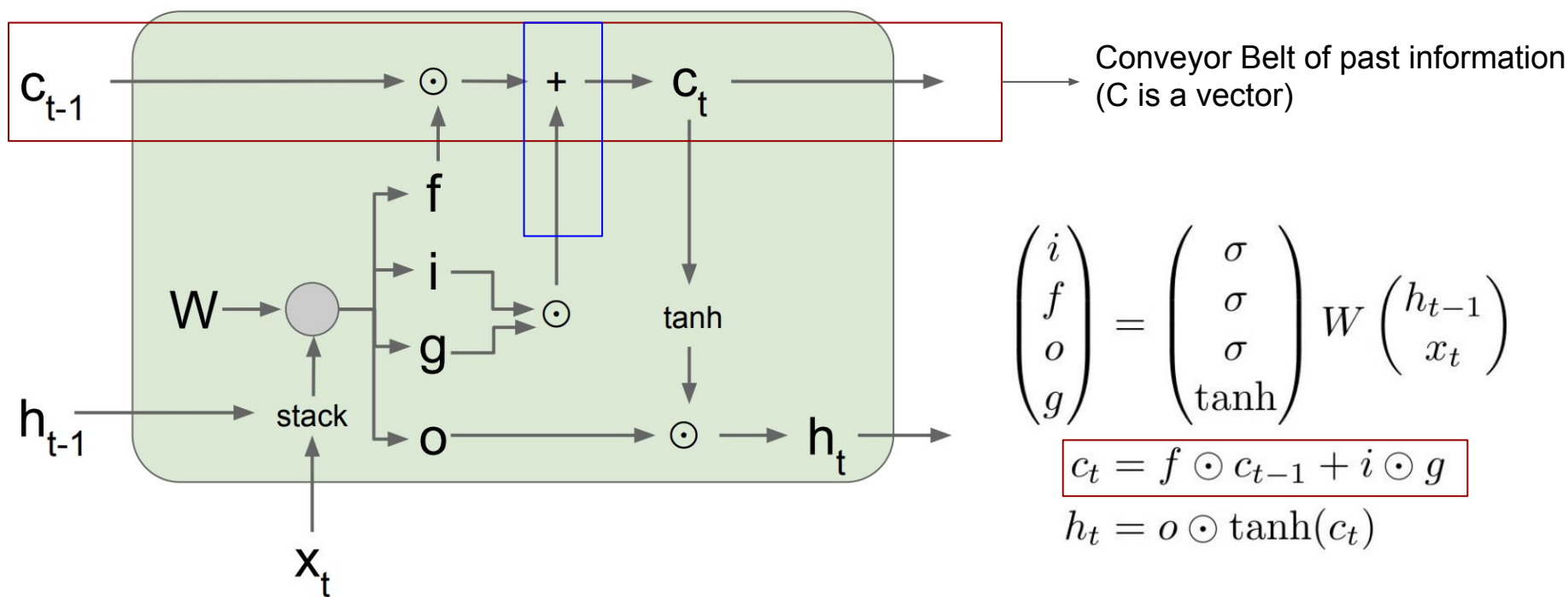
At each time-step, we can remove (forget  $f$ ) or add information to the conveyor belt.

# What's going on?



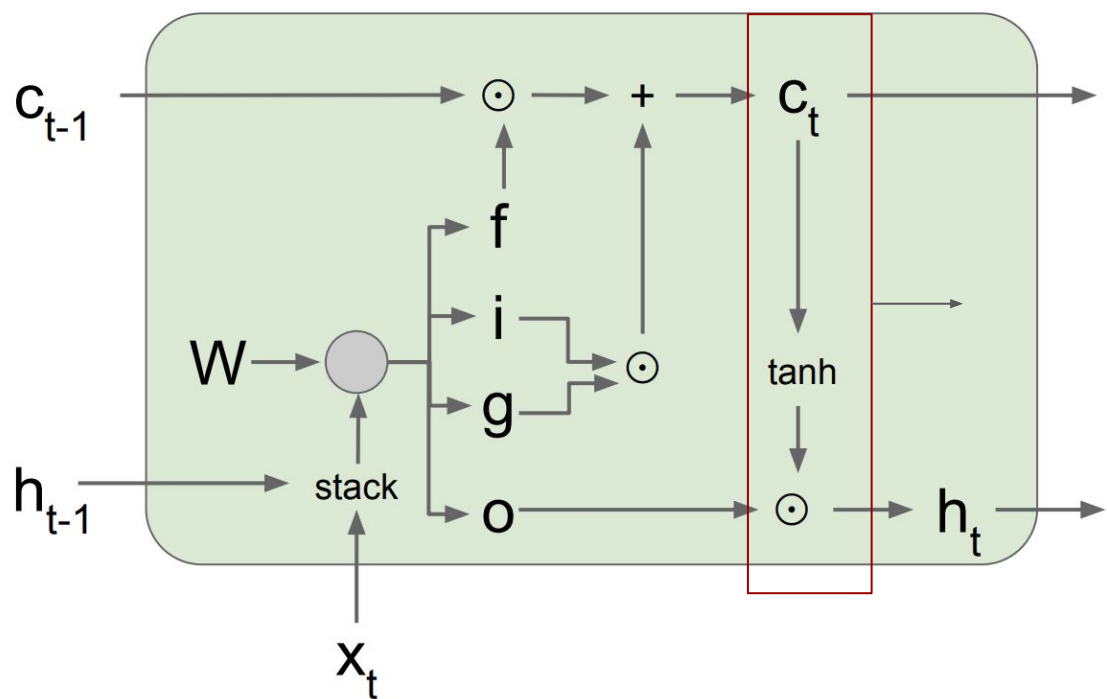
At each time-step, we can remove (forget  $f$ ) or add information to the conveyor belt.

# What's going on?



$g$  decides what new information to add and  $i$  indicates its importance (between 0 to 1)

# What's going on?



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

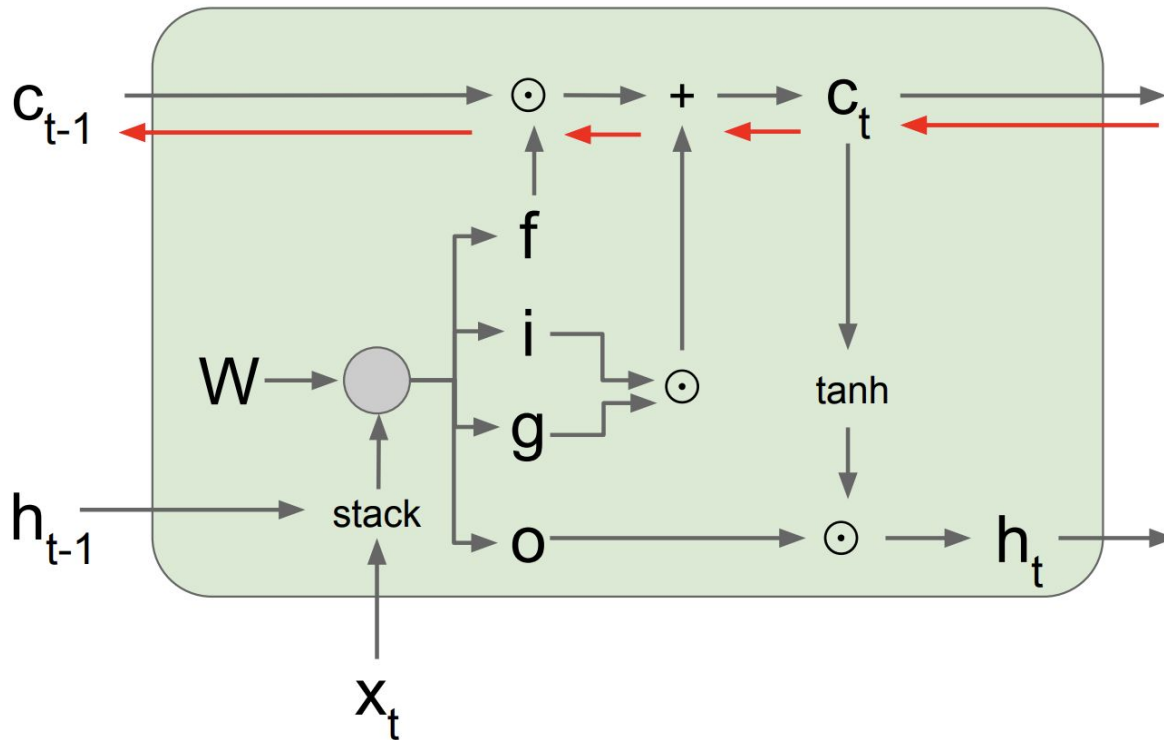
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

Next : We decide what information of the cell state ( $c_t$ ) to keep by the sigmoid  $o$  and we scale that information between  $(-1, 1)$  using  $\tanh$  to get  $h_t$



# What's going on?



Backpropagation from  $c_t$  to  $c_{t-1}$  only elementwise multiplication by  $f$ , no matrix multiply by  $W$

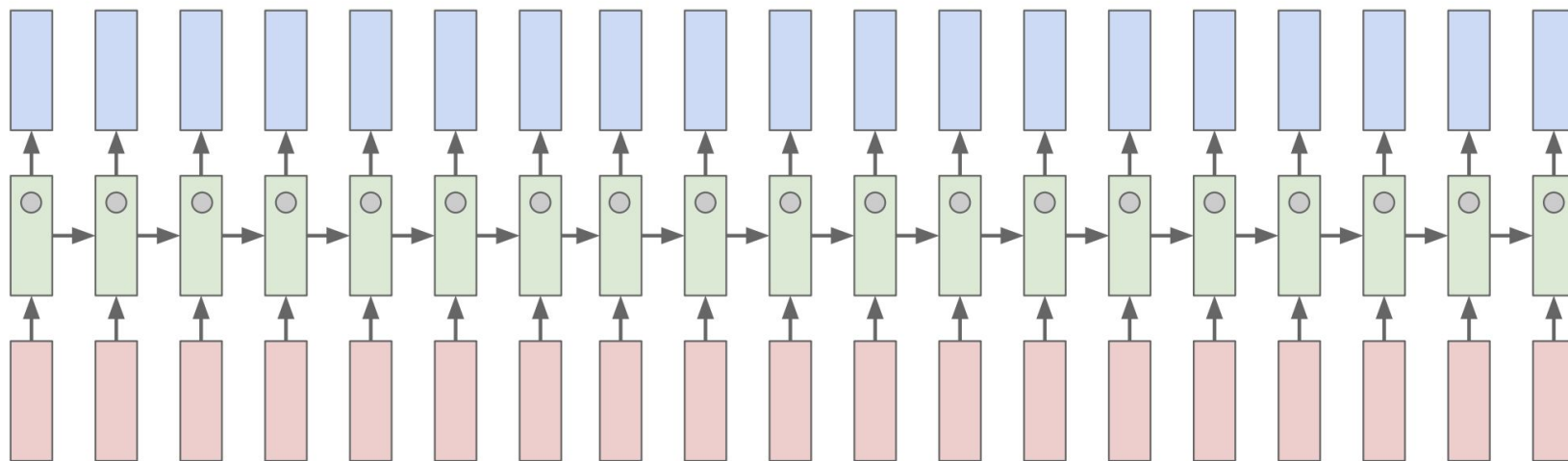
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

# RNN - Visualisation

## Searching for interpretable cells



Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 10 - 56 May 4, 2017

# RNN - Visualisation

## Searching for interpretable cells

"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

quote detection cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 10 - 58 May 4, 2017

# RNN - Visualisation

## Searching for interpretable cells

Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

line length tracking cell

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Figures copyright Karpathy, Johnson, and Fei-Fei, 2015; reproduced with permission

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 10 - 59 May 4, 2017

# References

- CS231n : Figures and some slides heavily borrowed from CS231n:
  - <http://cs231n.stanford.edu/syllabus.html>
- Christopher Colah's Blog on LSTM :
  - <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>