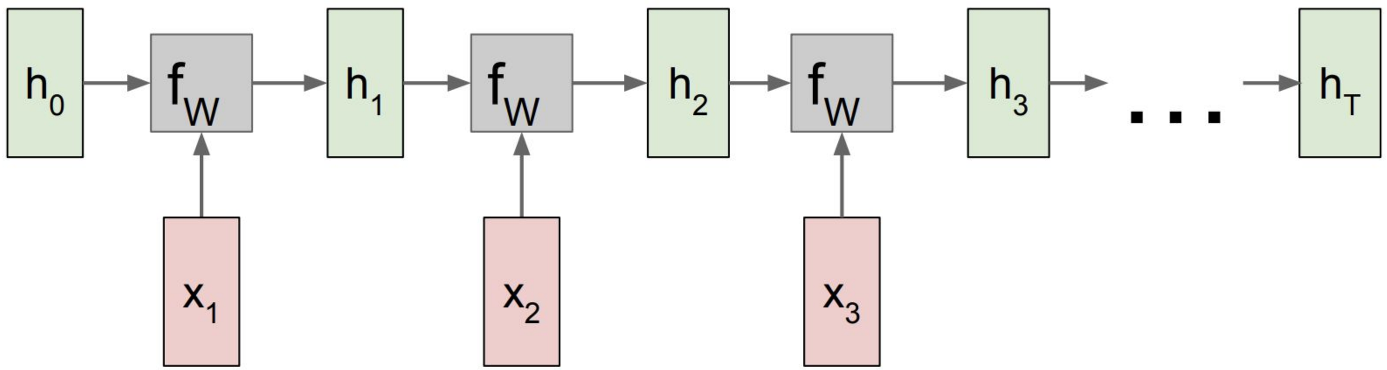
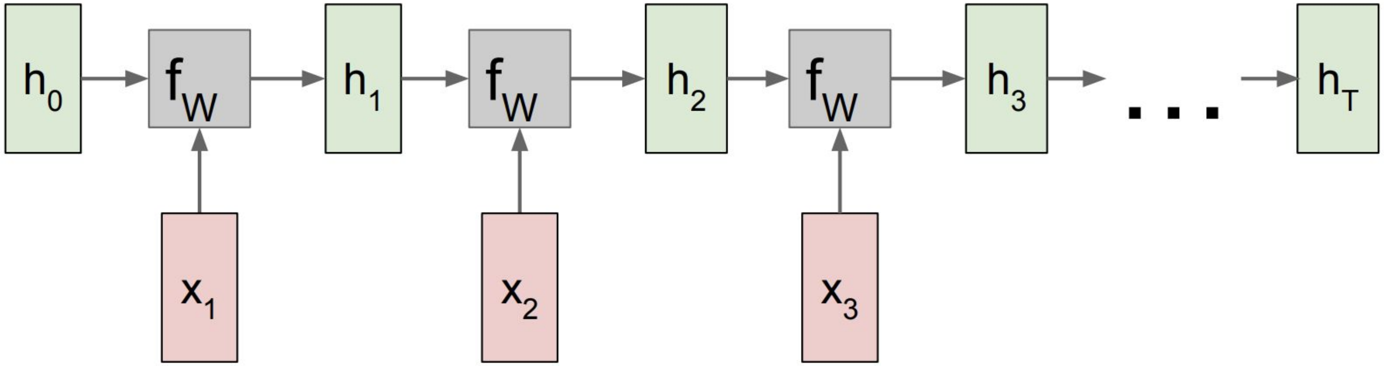
Recurrent Neural Networks

Recurrent Neural Network (Unrolled) Computational Graph



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

Recurrent Neural Network Computational Graph

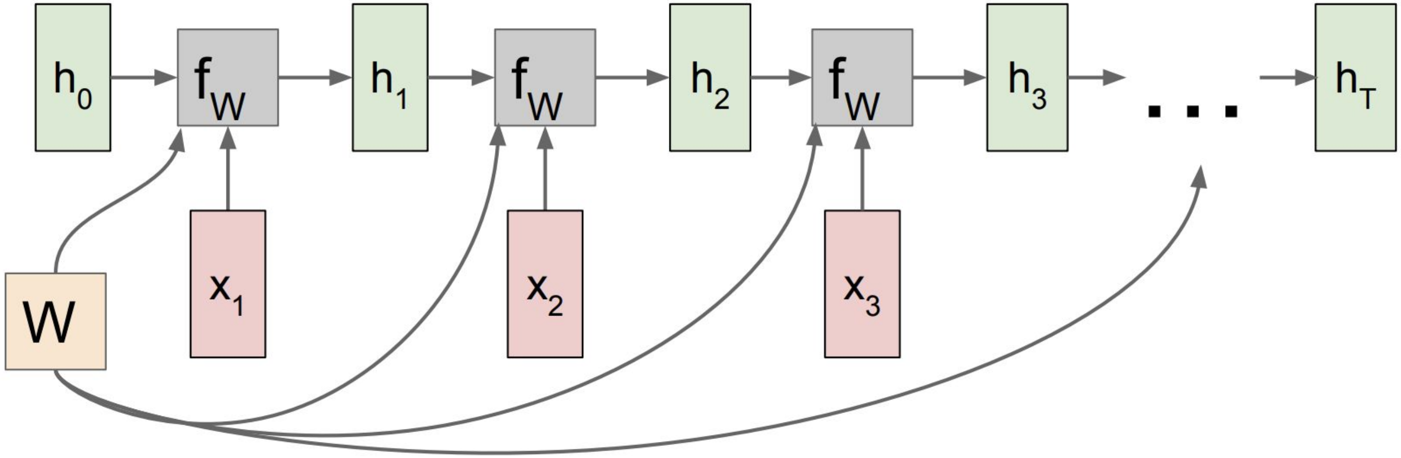


$$h_t = f_W(h_{t-1}, x_t) \longrightarrow \boxed{h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)}$$

Vanilla Recurrent Neural Network

Recurrent Neural Network Computational Graph

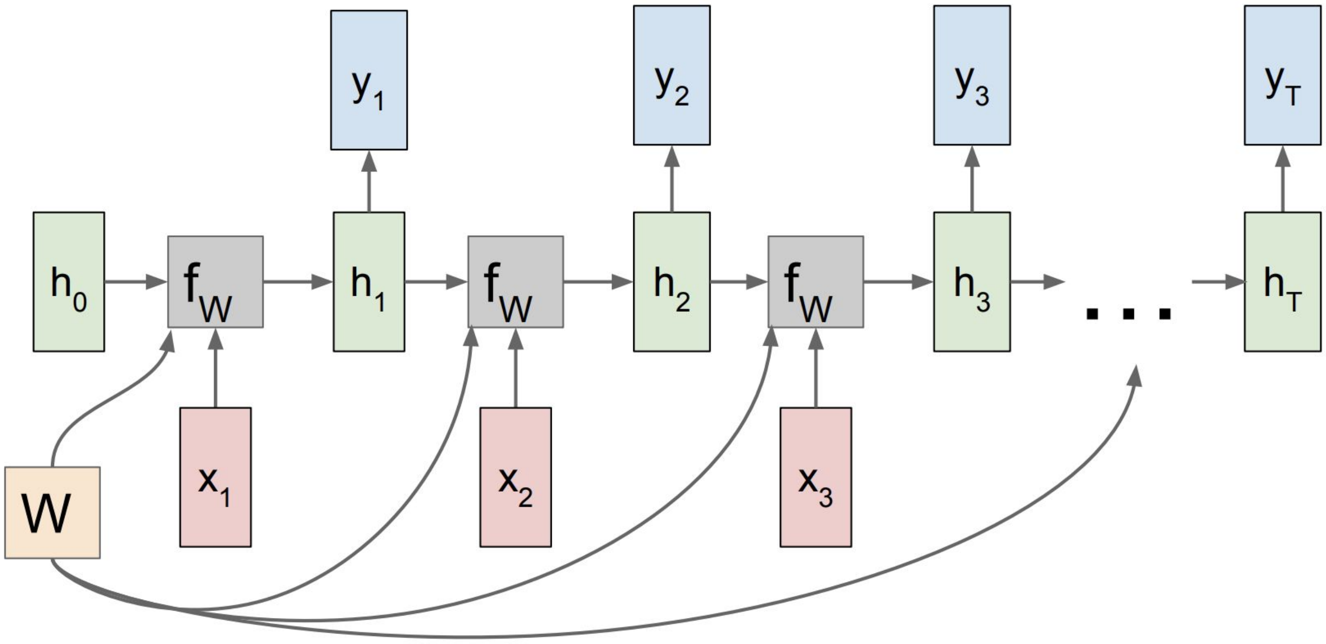
Re-use the same weight matrix at every time-step



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

Vanilla Recurrent Neural Network

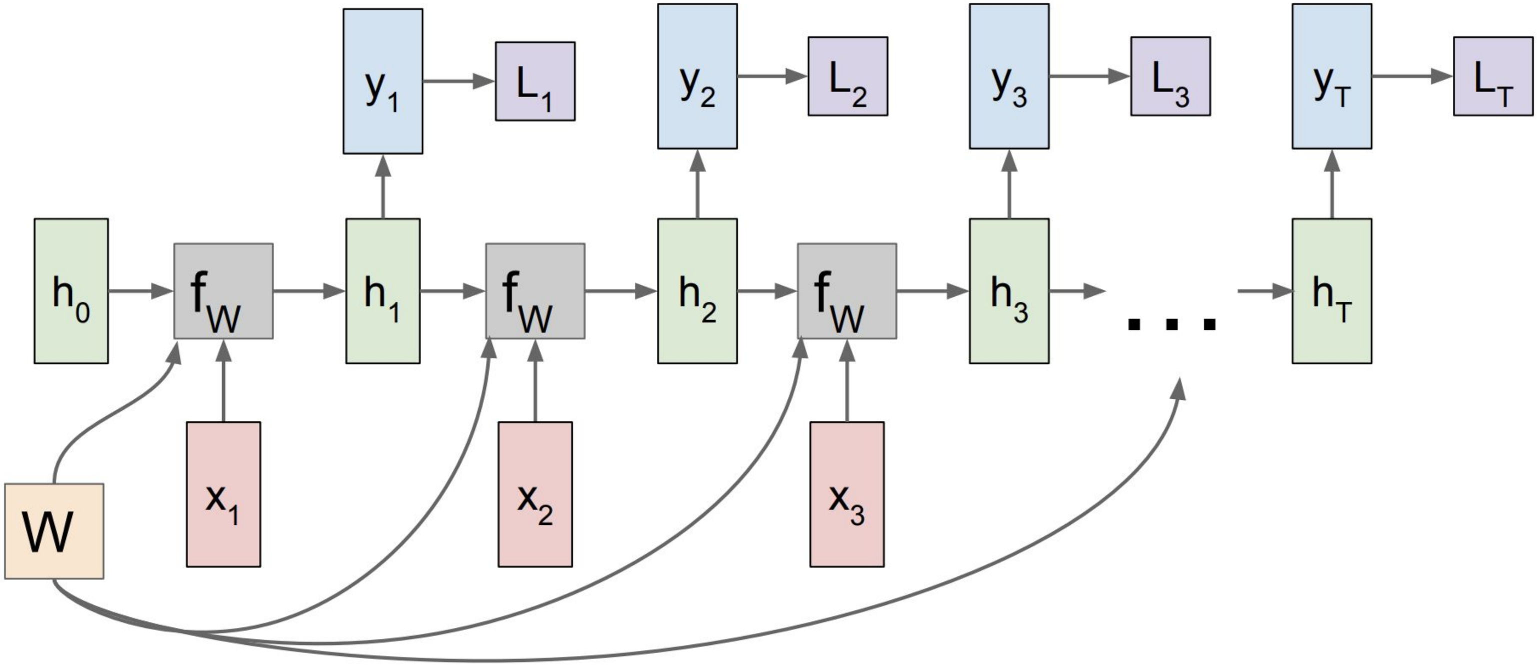
Recurrent Neural Network Computational Graph



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

$$\boxed{y_t = W_{hy}h_t}$$

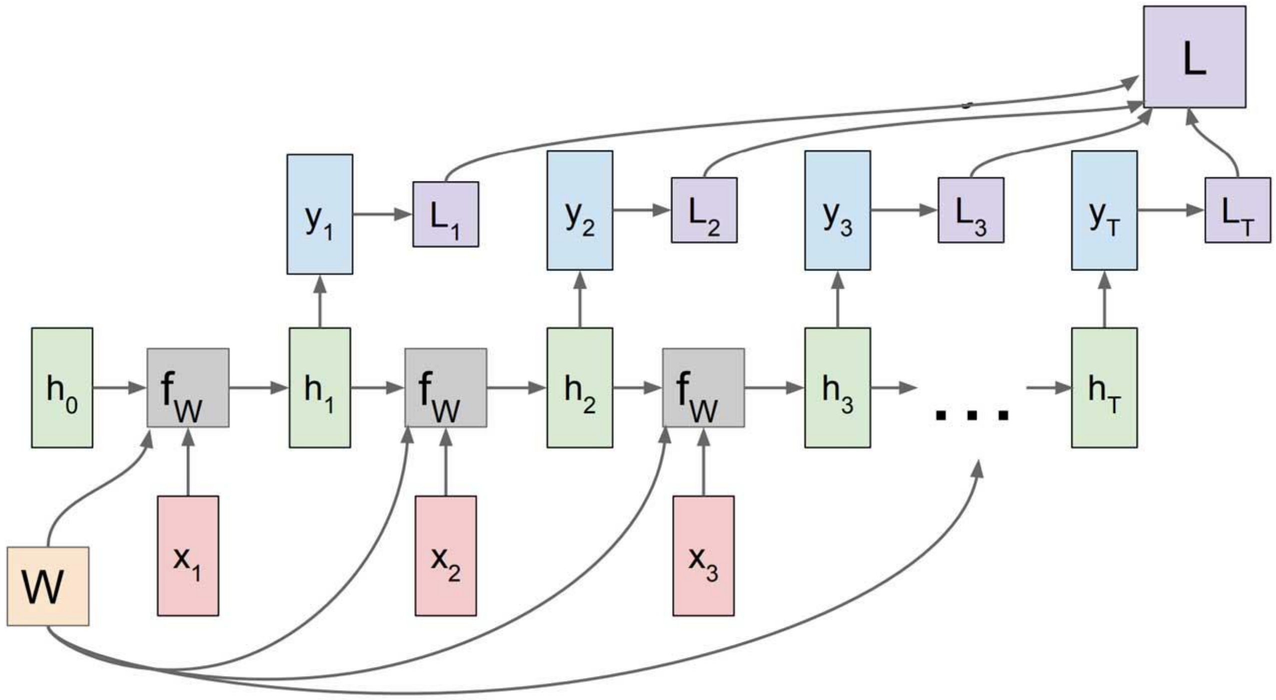
Recurrent Neural Network Computational Graph



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

$$\boxed{y_t = W_{hy}h_t}$$

Recurrent Neural Network Computational Graph



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

$$\boxed{y_t = W_{hy}h_t}$$

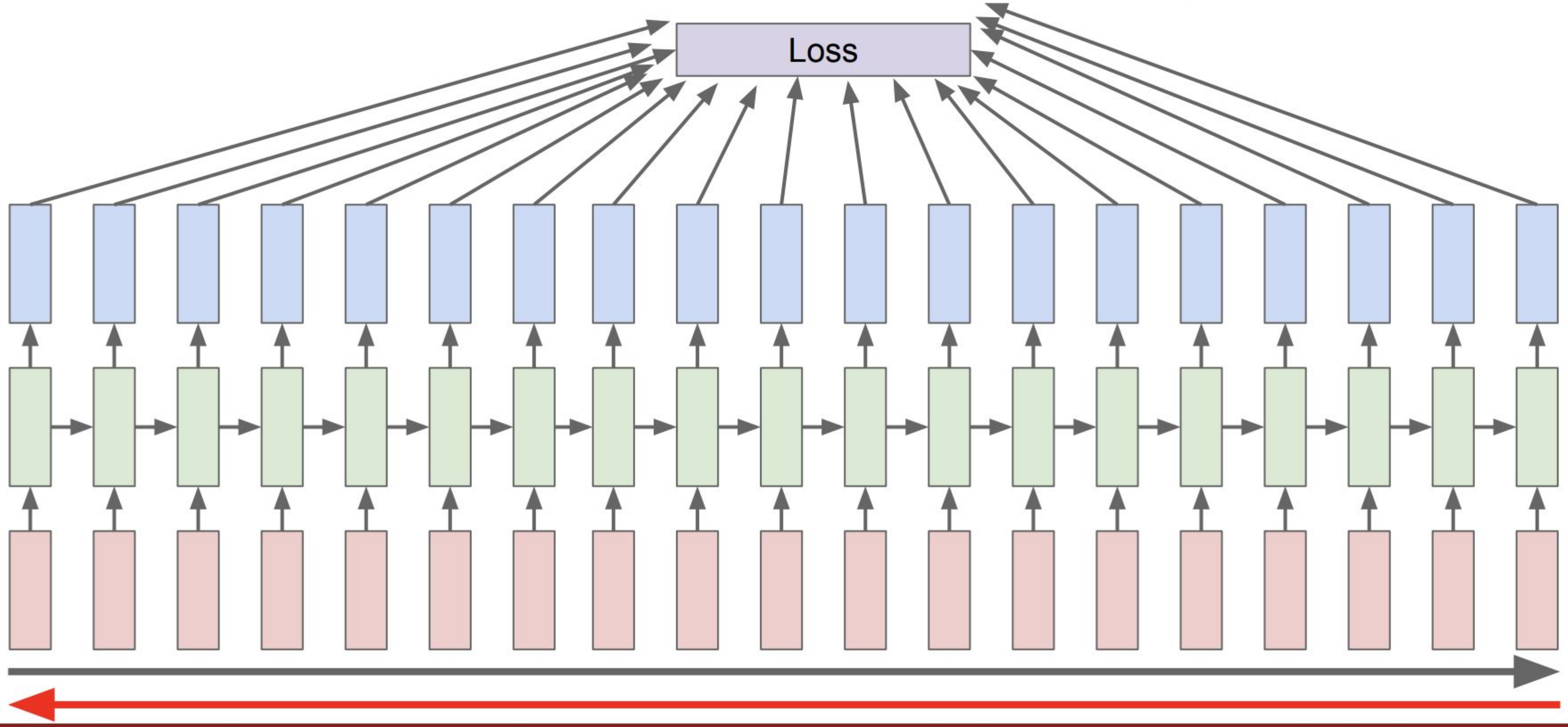
RNN - Forward Pass in Numpy!!

```
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)
```

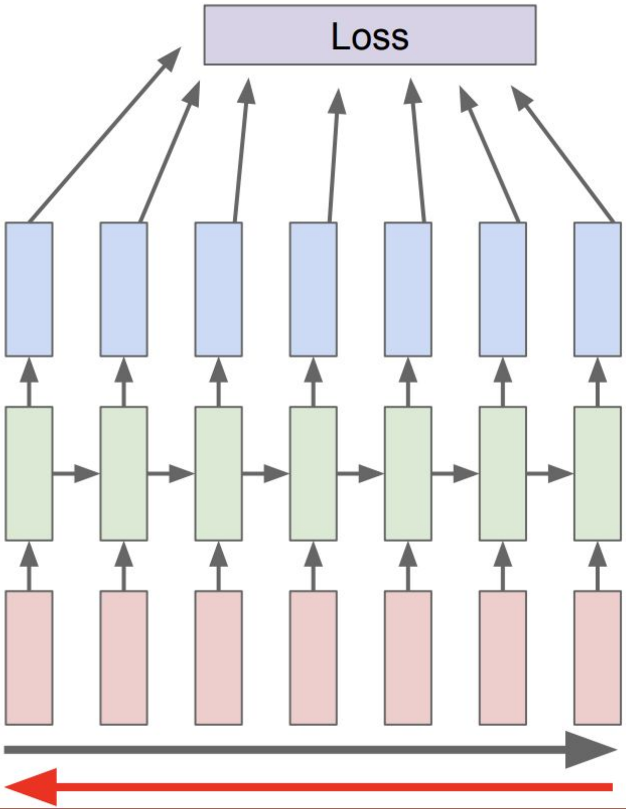
Source : https://gist.github.com/karpathy/d4dee566867f8291f086

Backpropagation through time

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

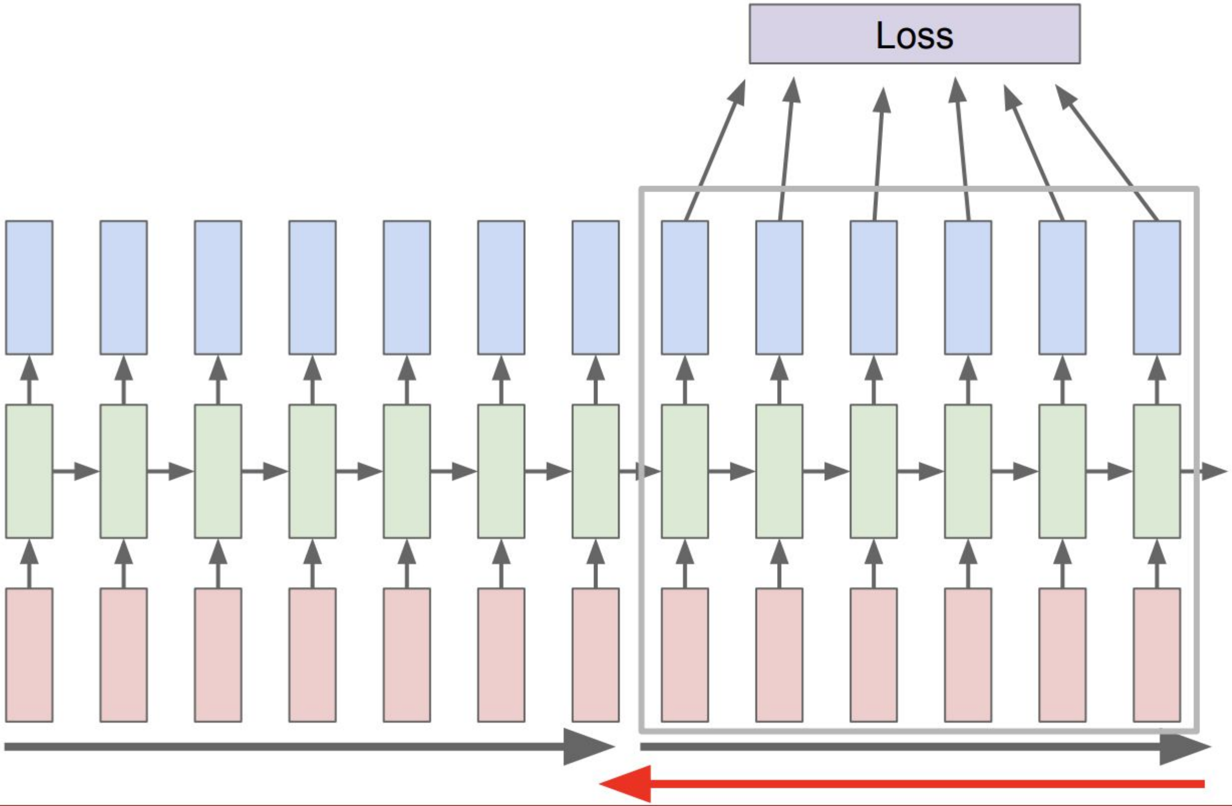


Truncated Backpropagation through time



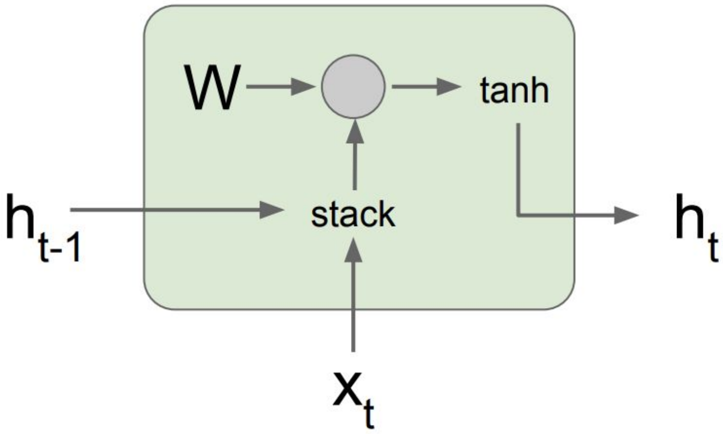
Run forward and backward through chunks of the sequence instead of whole sequence

Truncated Backpropagation through time



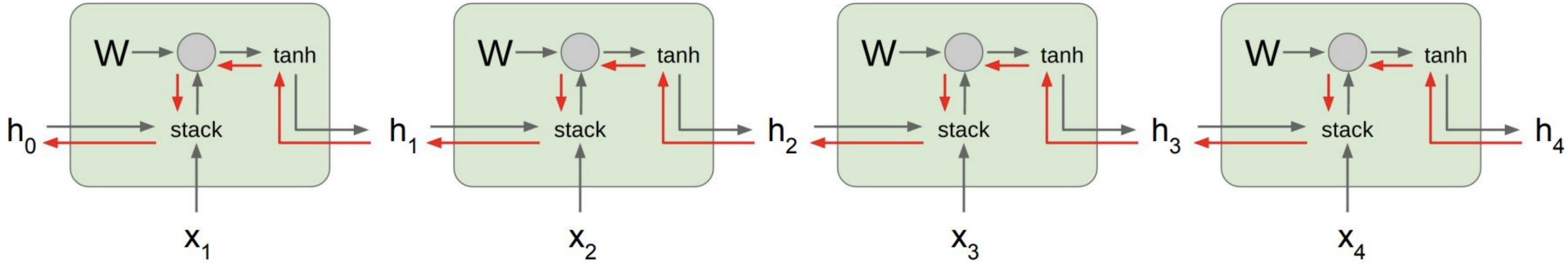
Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Moving Towards LSTMs



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

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

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

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

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

W : h x 2h

LSTM

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

Vanilla RNN

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

W : h x 2h

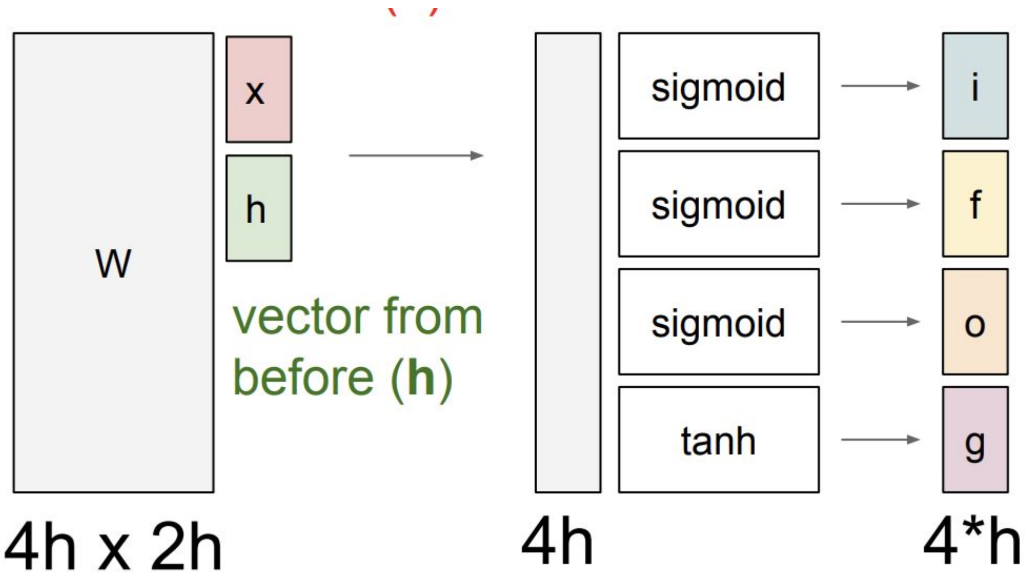
LSTM

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

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

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

W : 4h x 2h

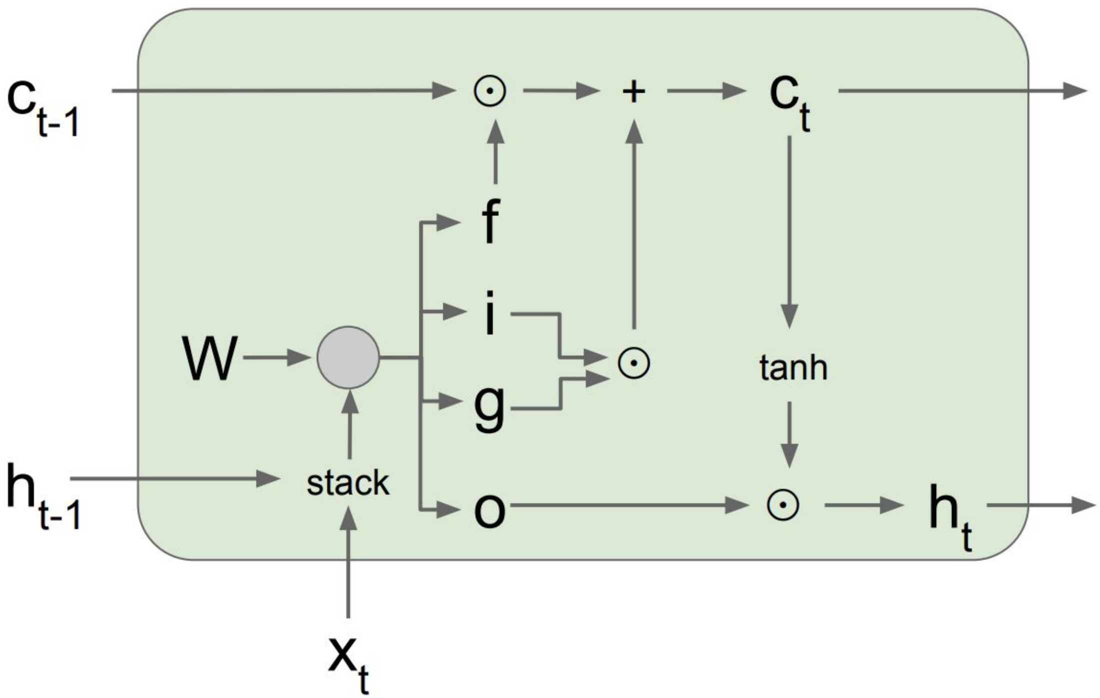


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

What's going on?

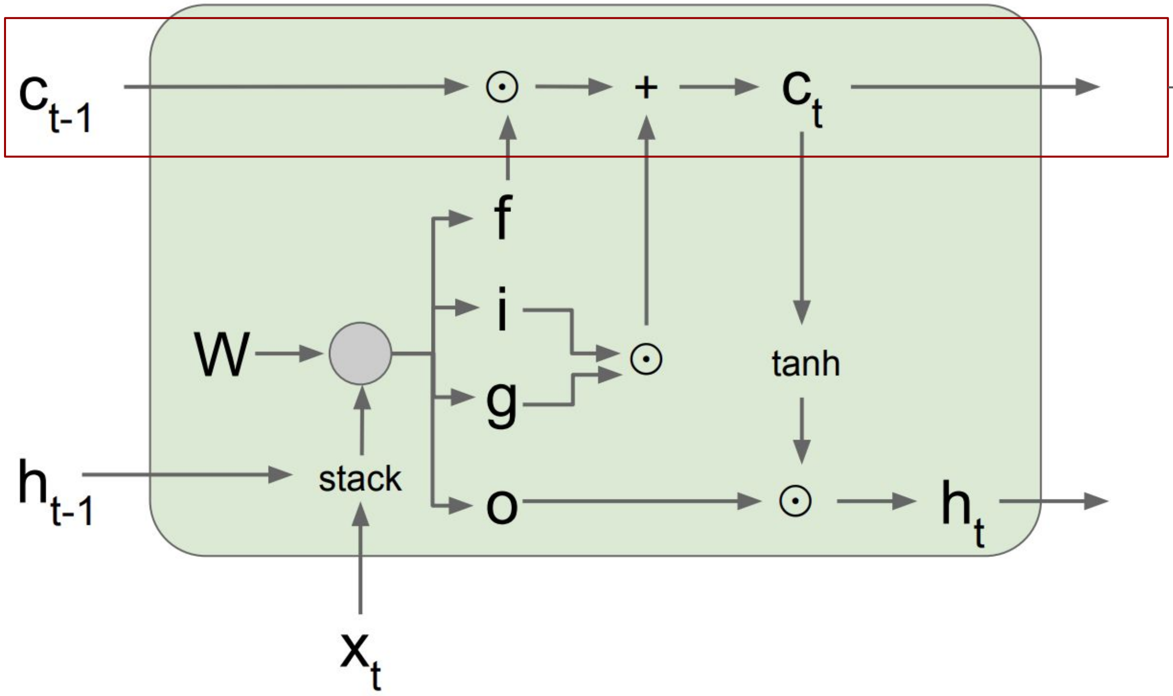


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

What's going on?



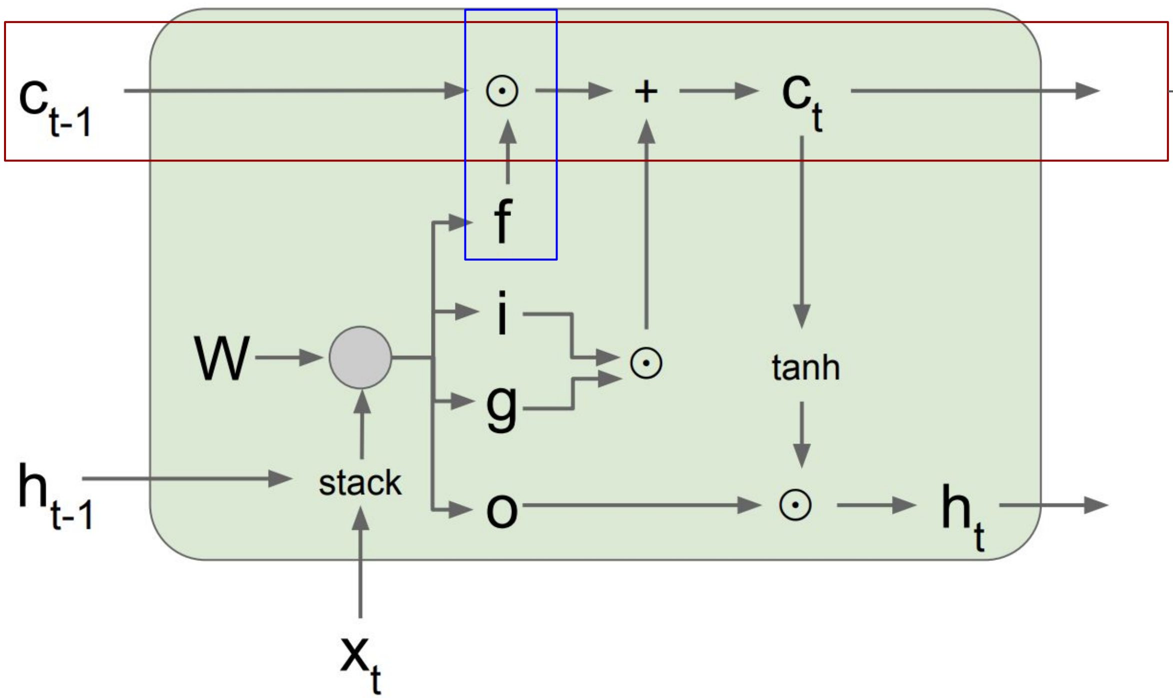
Conveyor Belt of past information
(C is a vector)

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

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

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

What's going on?



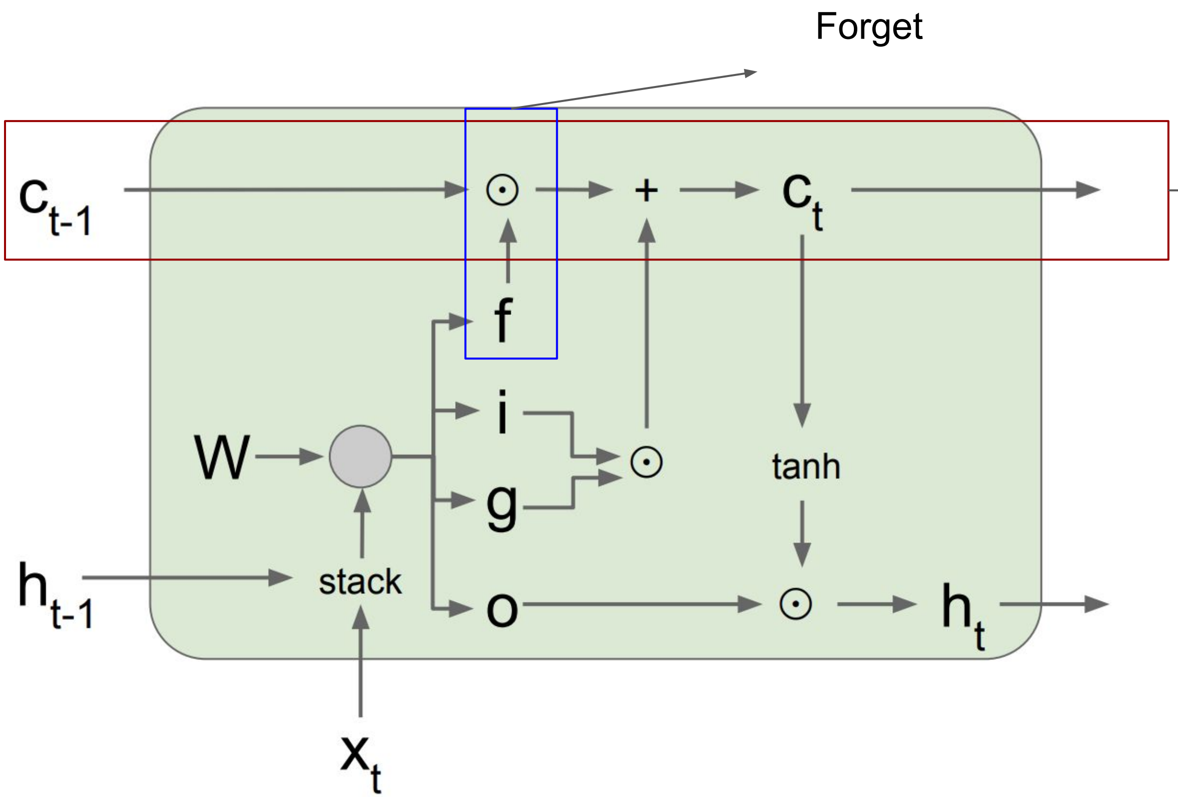
Conveyor Belt of past information
(C is a vector)

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

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

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

What’s going on?



Conveyor Belt of past information
(C is a vector)

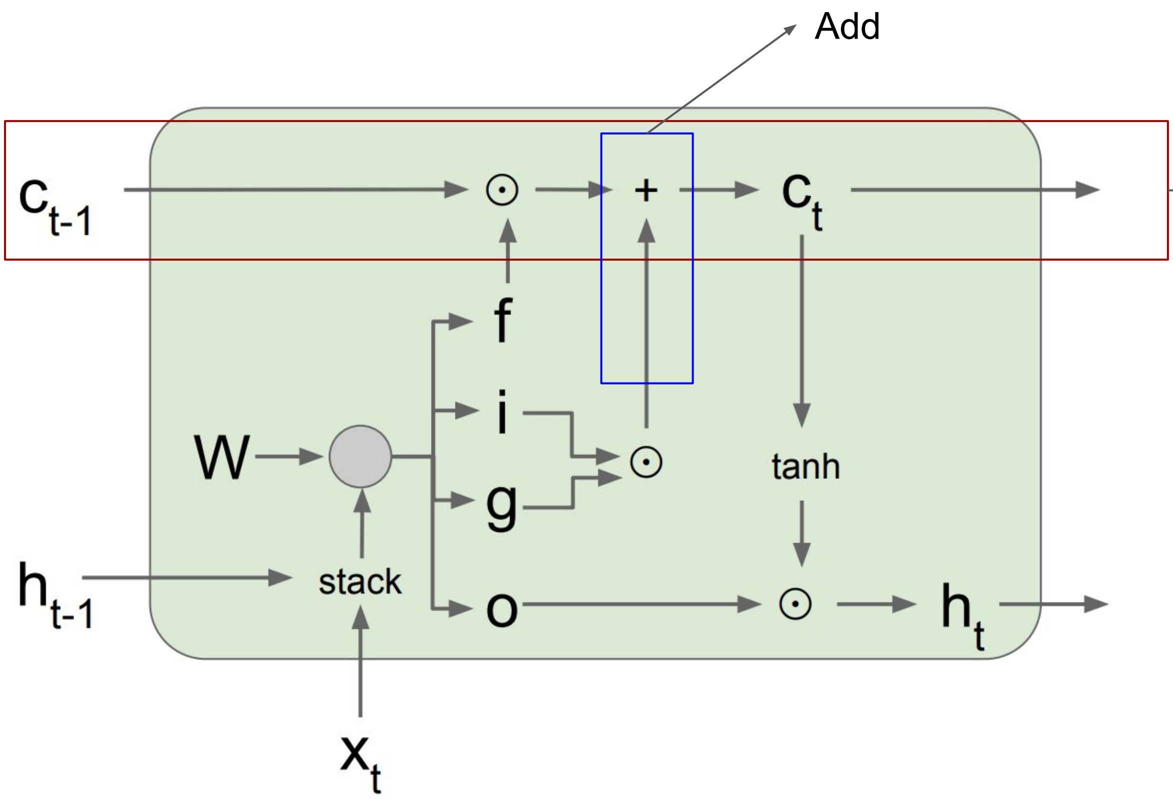
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

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

What's going on?



Conveyor Belt of past information
(C is a vector)

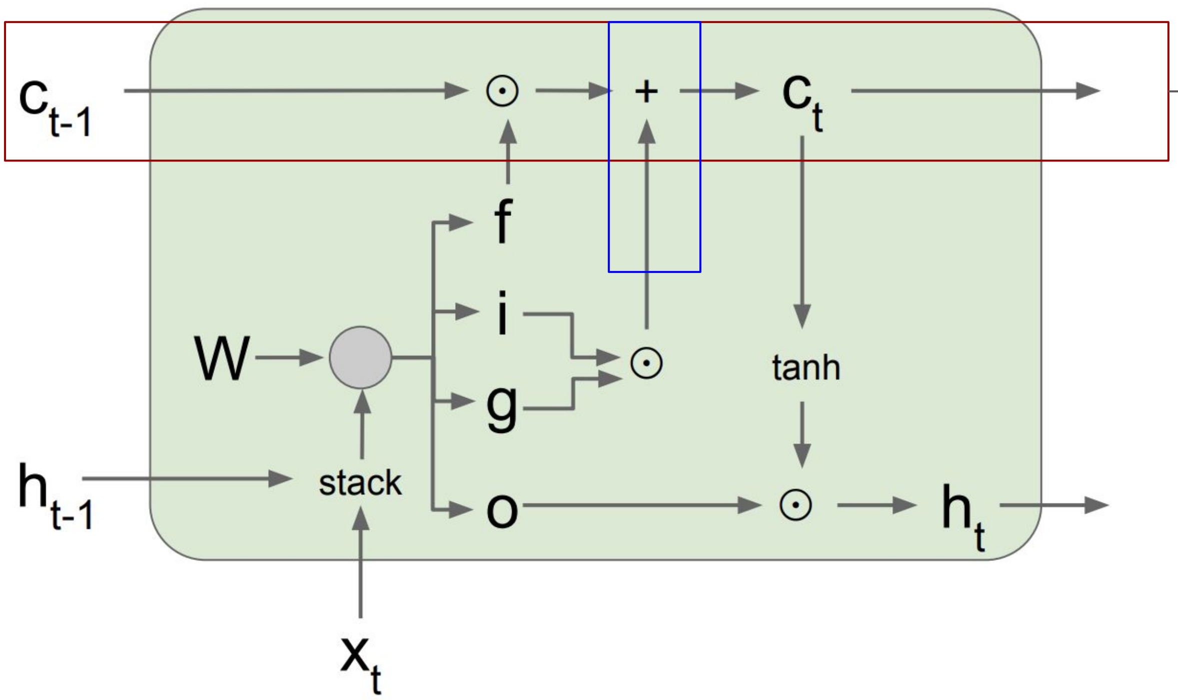
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

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

What's going on?



Conveyor Belt of past information
(C is a vector)

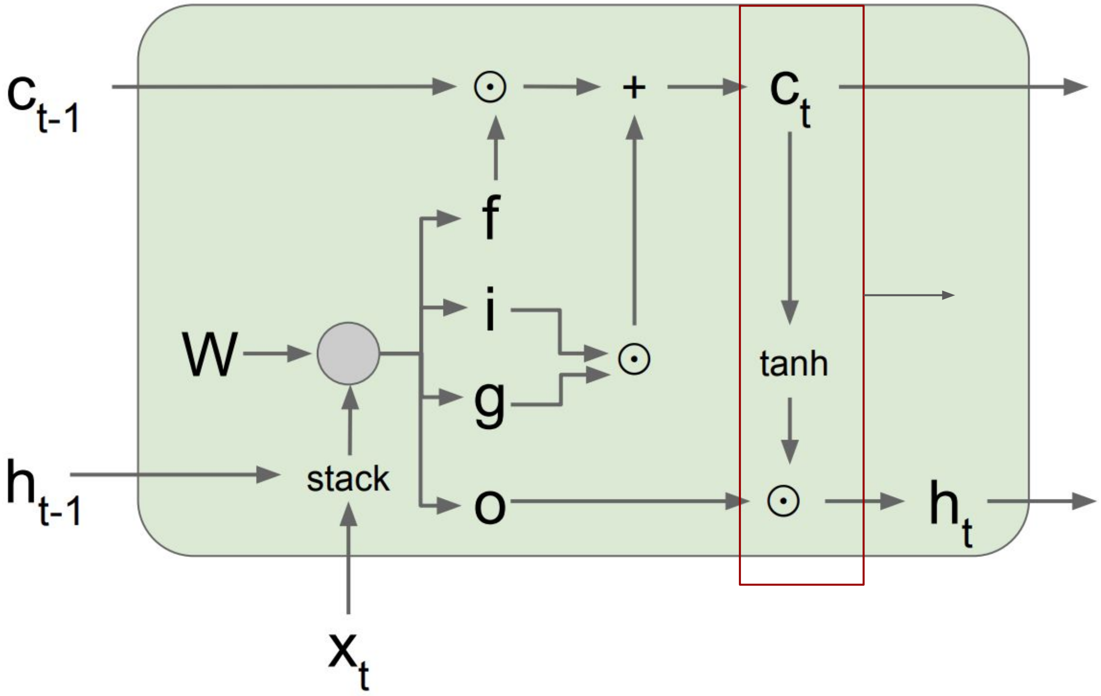
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

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

What's going on?



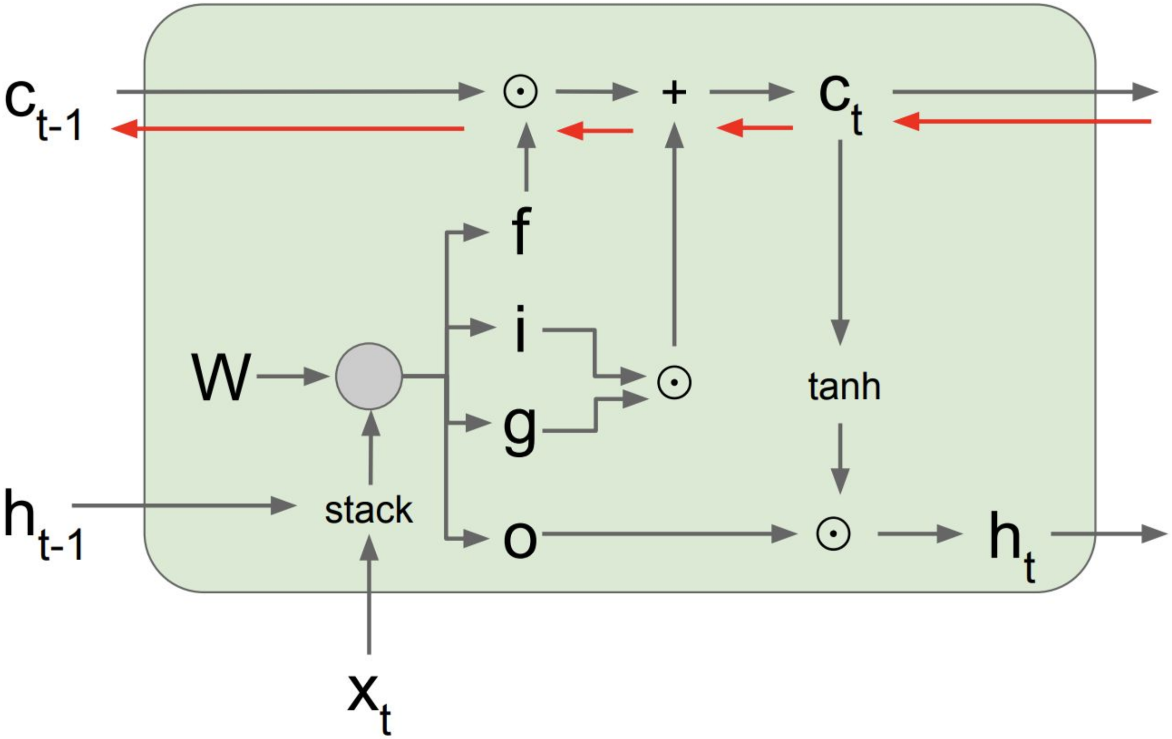
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

$$\boxed{h_t = o \odot \tanh(c_t)}$$

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

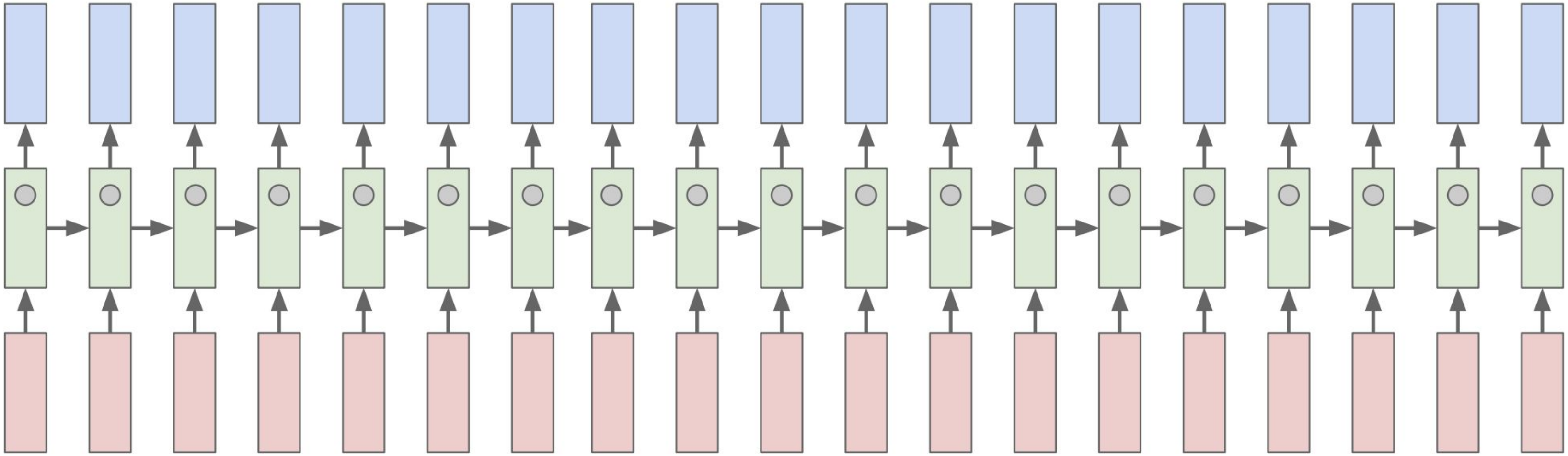
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

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

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

RNN - Visualisation

Searching for interpretable cells



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

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.

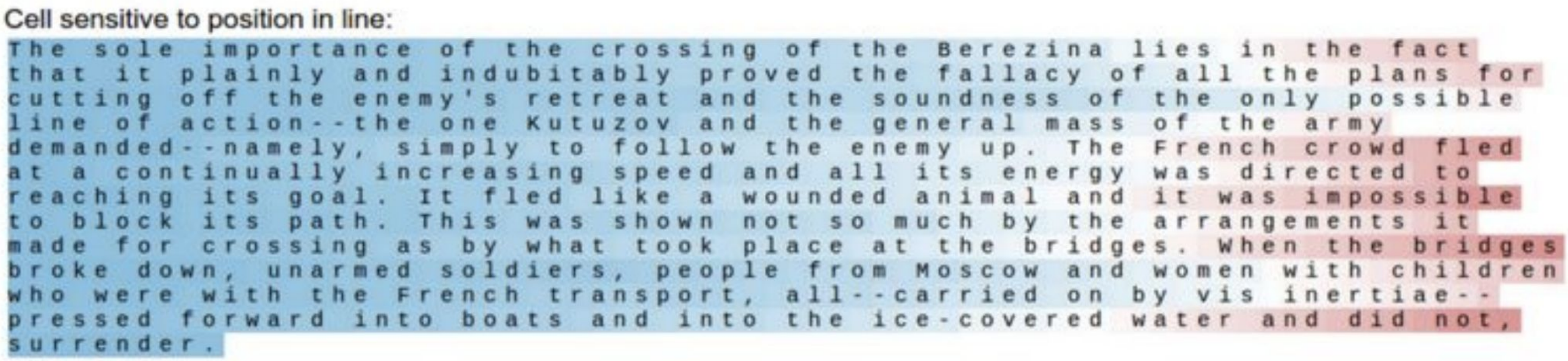
Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

quote detection cell

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

RNN - Visualisation

Searching for interpretable cells



line length tracking cell

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

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/