Deep Learning for Natural Language Processing

Stephen Clark et al.
University of Cambridge and DeepMind

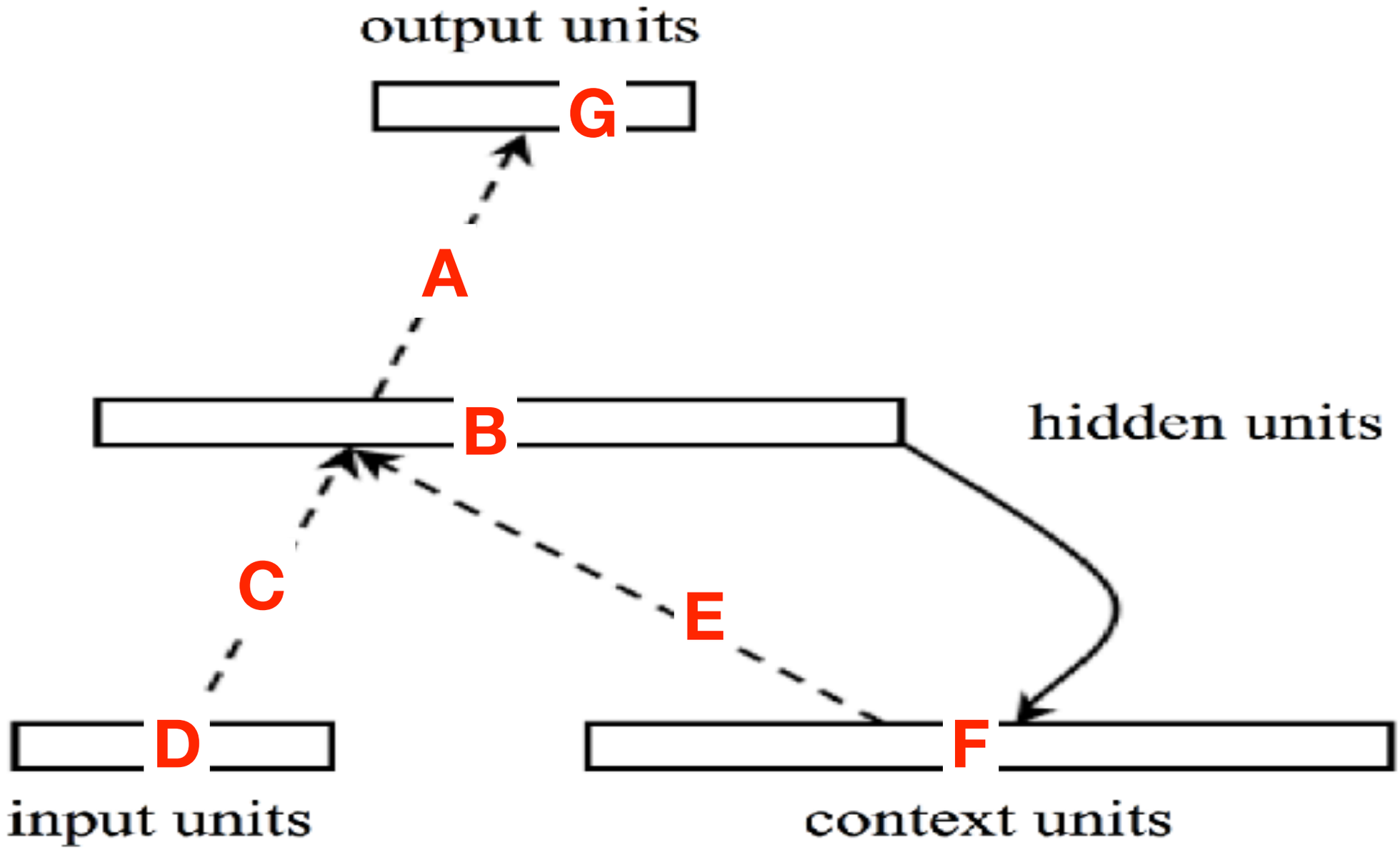
8. Long Short Term Memory

Felix Hill
DeepMind

RNNs: A recap

$$h_t = tanh(Uh_{t-1} + Wx_t)$$

$$y_t = Vh_t$$



What is the forward computation like?

$$h_t = tanh(Uh_{t-1} + Vx_t)$$

What is the forward computation like?

$$h_t = tanh(Uh_{t-1} + Vx_t)$$

$$h_{t+1} = tanh(Uh_t + Vx_{t+1})$$

What is the forward computation like?

$$h_t = tanh(Uh_{t-1} + Vx_t)$$

$$h_{t+1} = tanh(Uh_t + Vx_{t+1})$$

$$= tanh(Utanh(Uh_{t-1} + Vx_t) + Vx_{t+1})$$

$$= tanh(Utanh(Utanh(Utah.....)$$

DeepMind

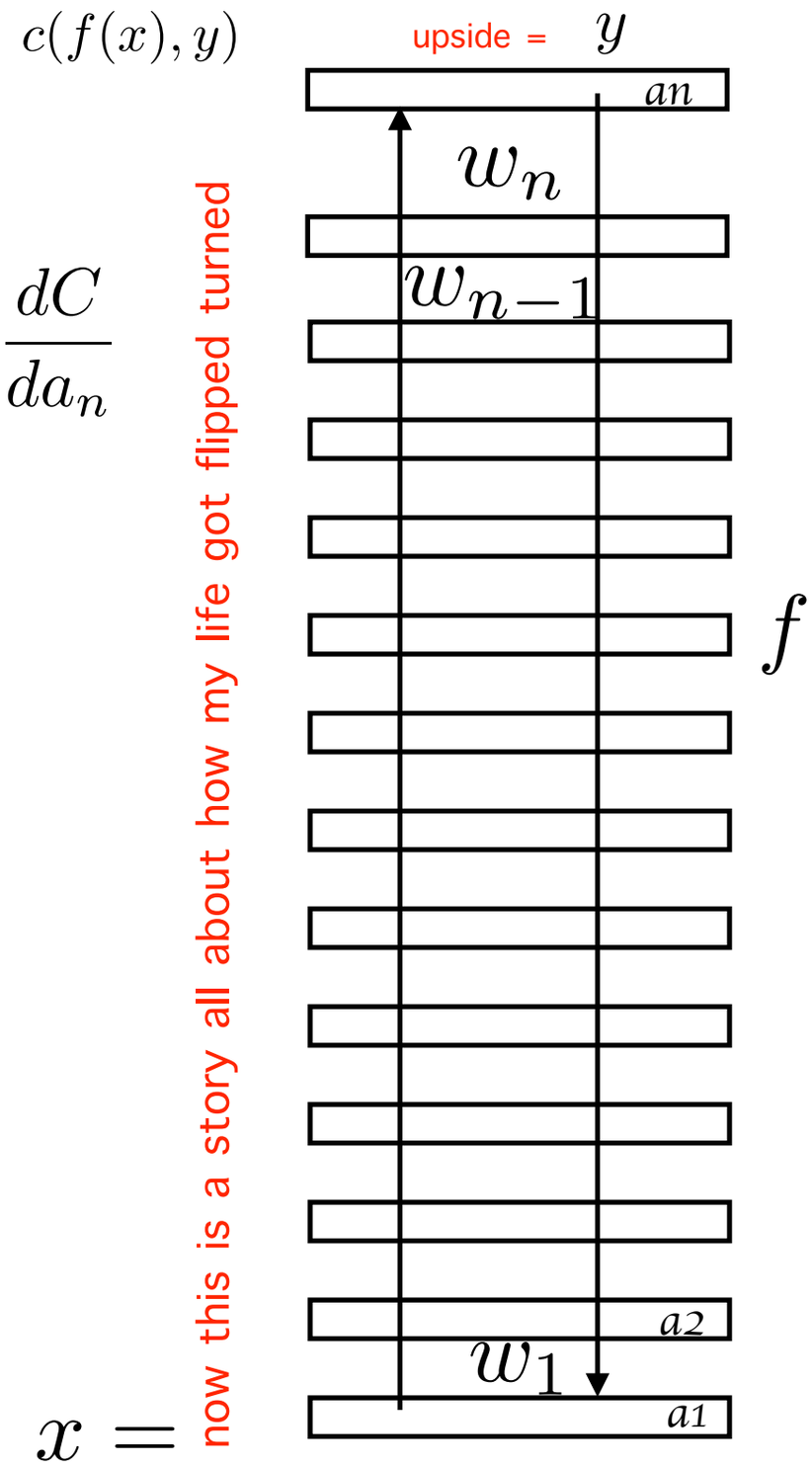
UNIVERSITY OF CAMBRIDGE

“Vanishing” gradients

$$\frac{dC}{dw_1} \propto \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \cdots \times w_n \times \sigma'(z_n) \times \frac{dC}{da_n}$$

where

$$a_i = \sigma(z_i)$$



“Vanishing” gradients

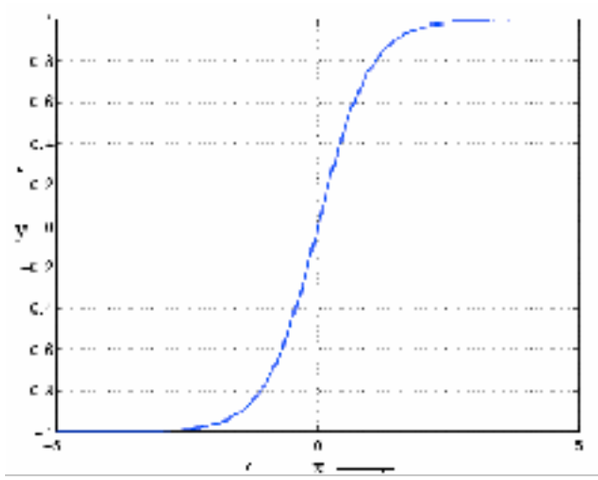
(or exploding)

in an RNN

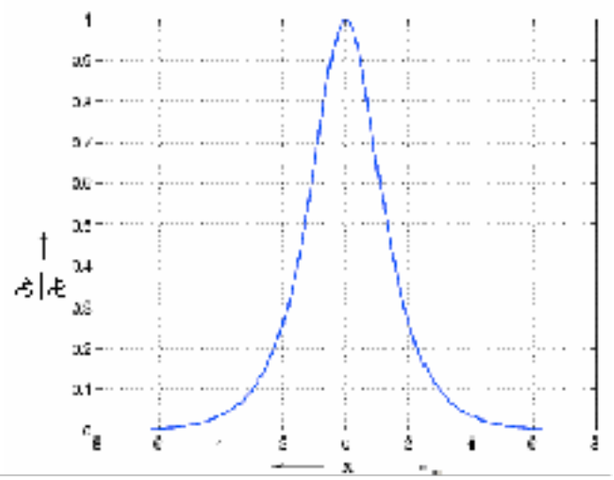
$$\frac{dC}{dw_1} \propto w^n \times \sigma'(z_1) \times \cdots \times \sigma'(z_n) \times \frac{dC}{da_n}$$

small change, big consequences

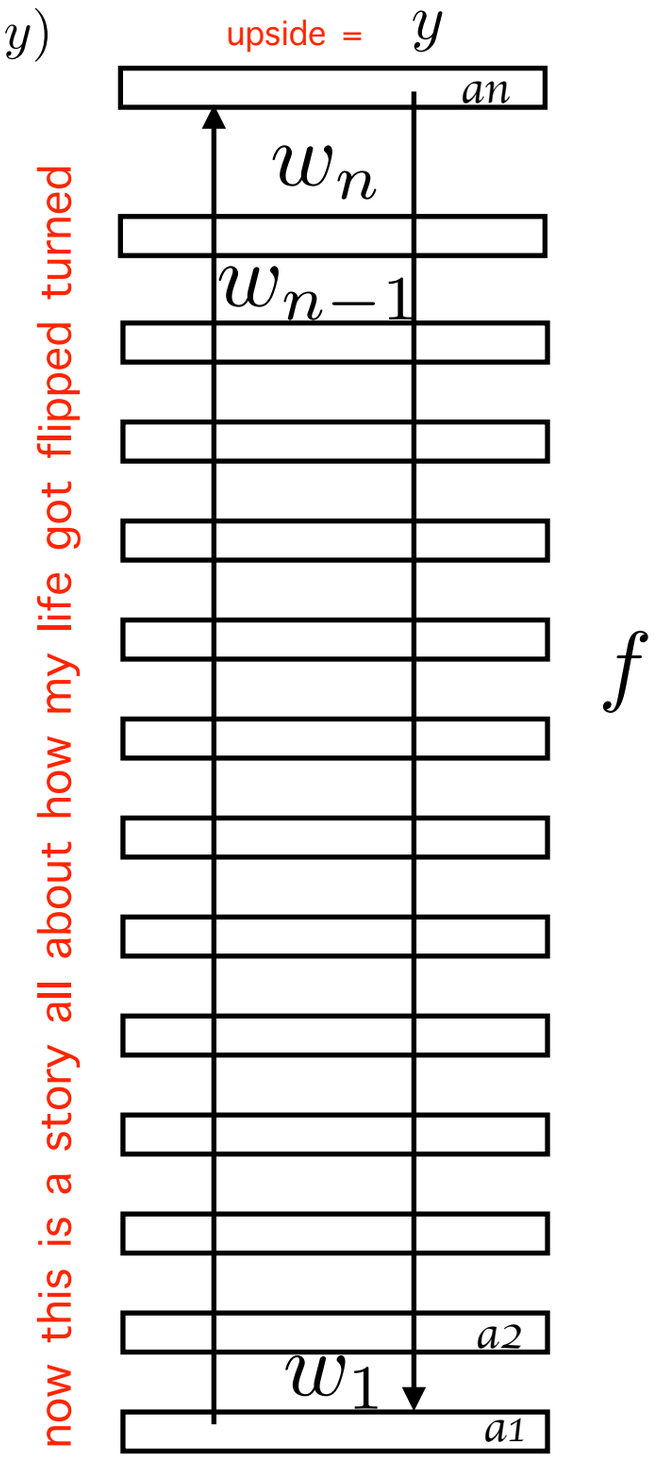
$\sigma(x)$



$\sigma'(x)$



$c(f(x), y)$

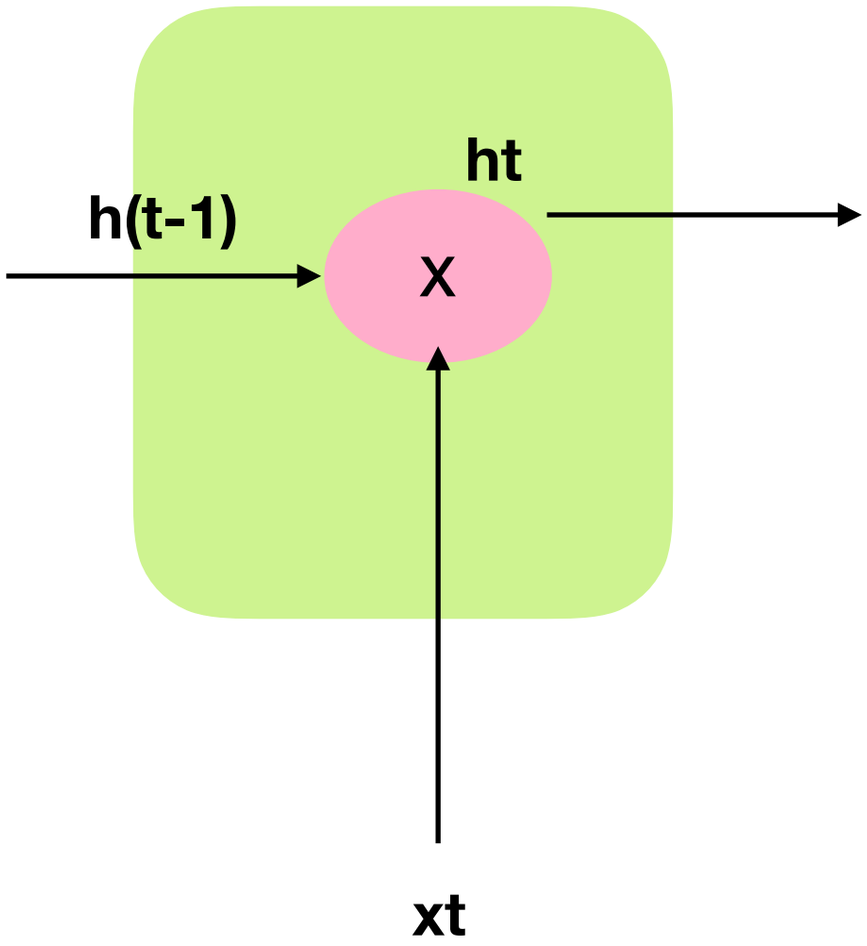


Long Short Term Memory (LSTM)

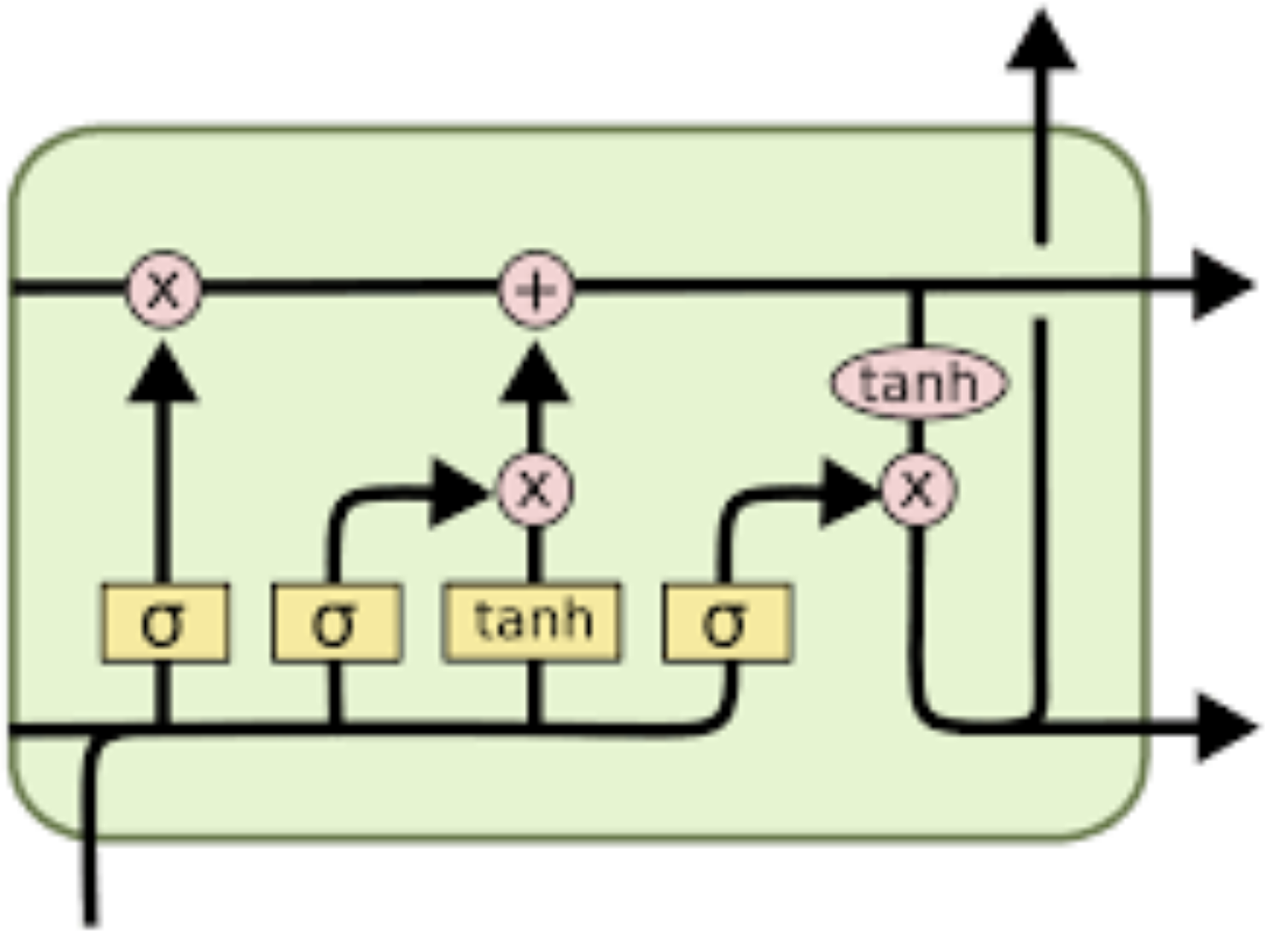
Hochreiter and Schmidhuber, 1997

RNN

Elman, 1990



LSTM

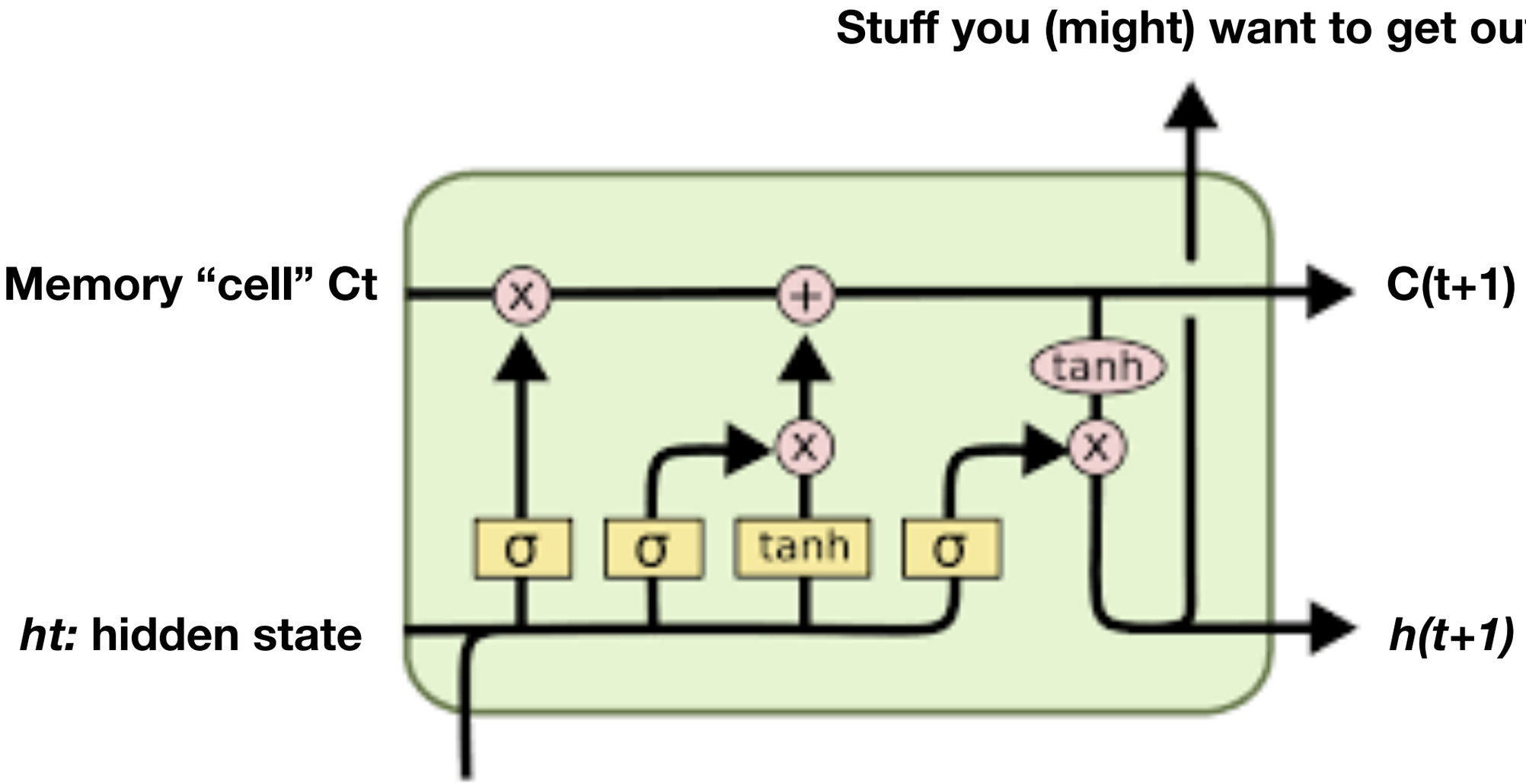


Thanks to
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

DeepMind

UNIVERSITY OF CAMBRIDGE

Be careful with the past!



Thanks to
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

DeepMind

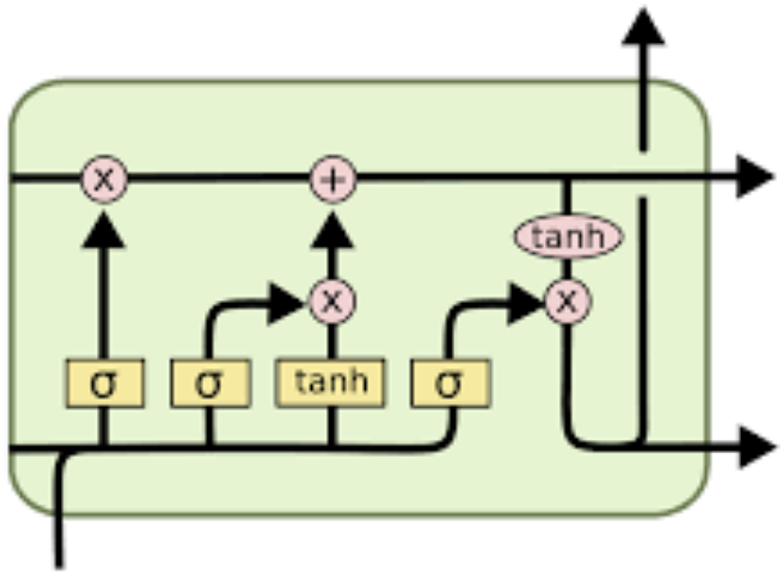
UNIVERSITY OF CAMBRIDGE

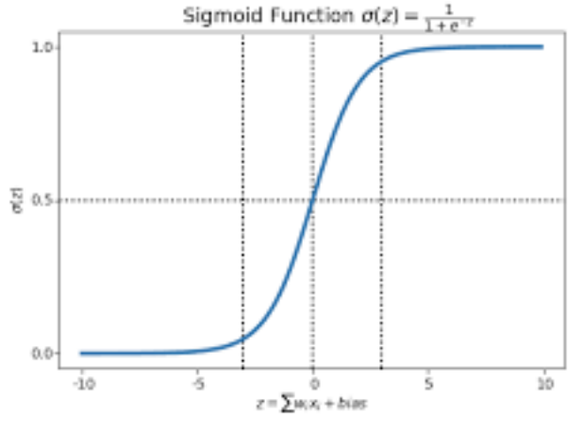
Sigmoid “gates”

$$f_t = \sigma(W_{f_h} h_{t-1} + W_{f_x} x_t + b_f)$$

$$i_t = \sigma(W_{i_h} h_{t-1} + W_{i_x} x_t + b_i)$$

$$o_t = \sigma(W_{o_h} h_{t-1} + W_{o_x} x_t + b_o)$$



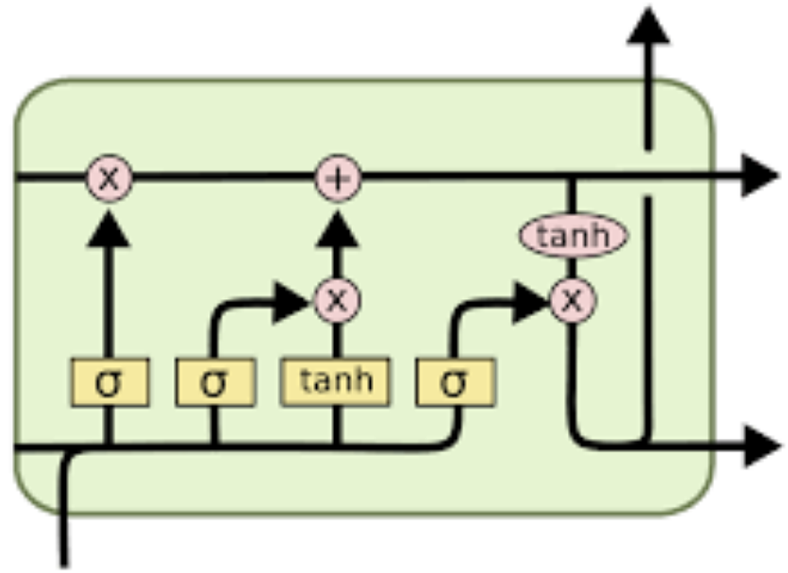


DeepMind

UNIVERSITY OF CAMBRIDGE

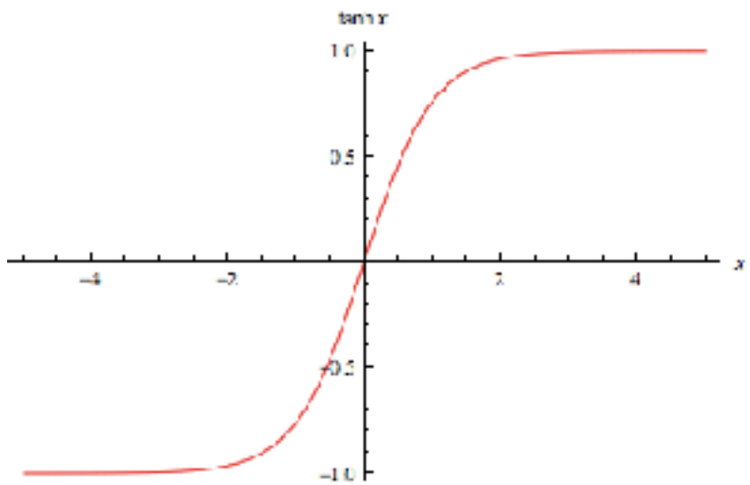
New stuff to consider

$$f_t = \sigma(W_{f_h} h_{t-1} + W_{f_x} x_t + b_f)$$
$$i_t = \sigma(W_{i_h} h_{t-1} + W_{i_x} x_t + b_i)$$
$$o_t = \sigma(W_{o_h} h_{t-1} + W_{o_x} x_t + b_o)$$



$$\tilde{C}_t = \tanh(W_{c_h} h_{t-1} + W_{c_x} x_t + b_c)$$

Look familiar?



DeepMind

UNIVERSITY OF CAMBRIDGE

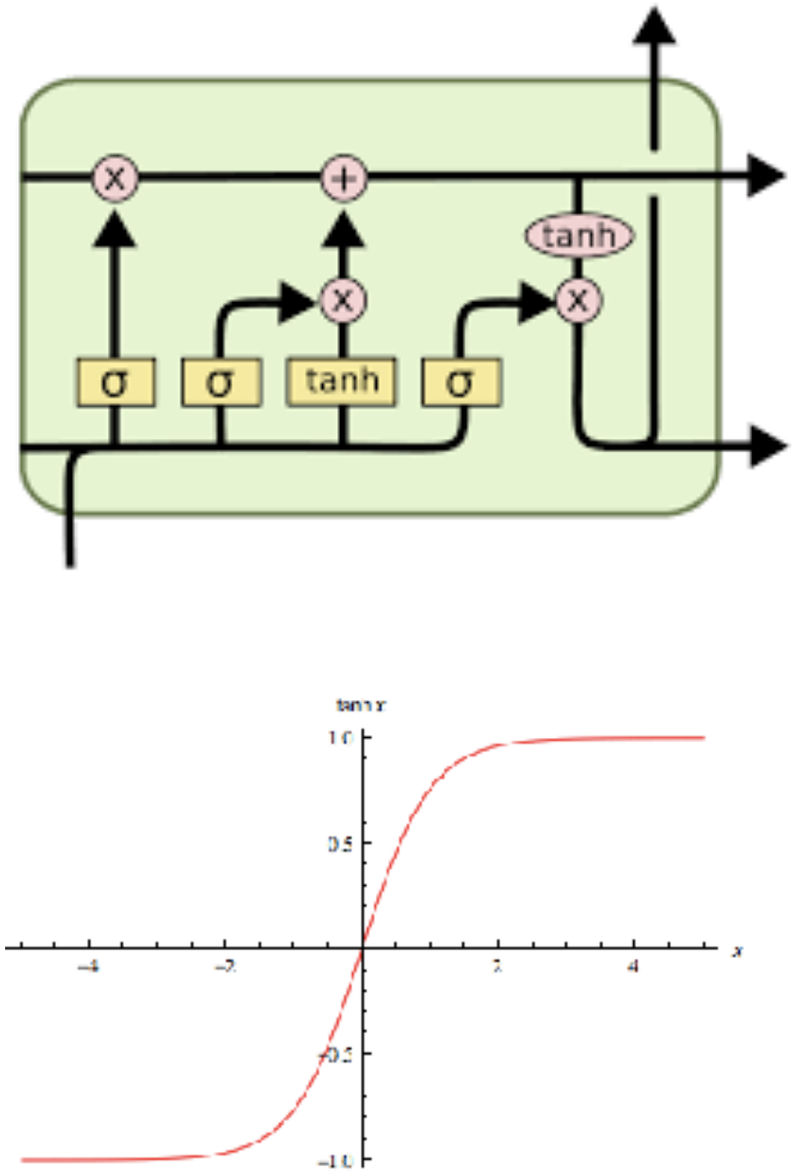
Carefully update your cell!!

$$f_t = \sigma(W_{f_h} h_{t-1} + W_{f_x} x_t + b_f)$$
$$i_t = \sigma(W_{i_h} h_{t-1} + W_{i_x} x_t + b_i)$$
$$o_t = \sigma(W_{o_h} h_{t-1} + W_{o_x} x_t + b_o)$$
$$\tilde{C}_t = \tanh(W_{c_h} h_{t-1} + W_{c_x} x_t + b_c)$$

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

forget some stuff

think some new stuff

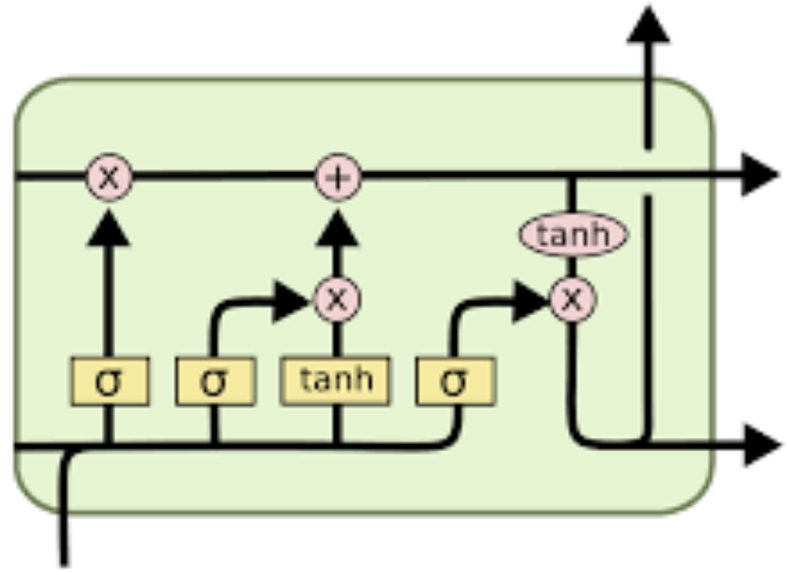


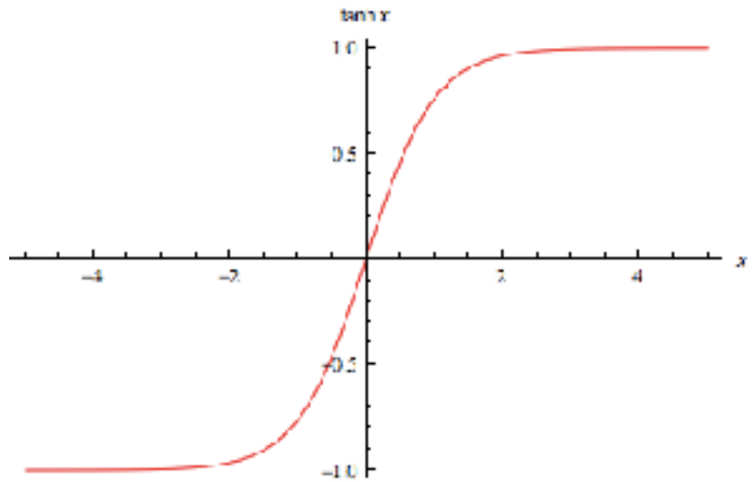
DeepMind

UNIVERSITY OF CAMBRIDGE

Use your cell (if you want!!)

$$f_t = \sigma(W_{f_h} h_{t-1} + W_{f_x} x_t + b_f)$$
$$i_t = \sigma(W_{i_h} h_{t-1} + W_{i_x} x_t + b_i)$$
$$o_t = \sigma(W_{o_h} h_{t-1} + W_{o_x} x_t + b_o)$$
$$\tilde{C}_t = \tanh(W_{c_h} h_{t-1} + W_{c_x} x_t + b_c)$$
$$C_t = C_{t-1} * f_t + \tilde{C}_t * i_t$$
$$h_t = o_t * \tanh(C_t)$$





DeepMind

UNIVERSITY OF CAMBRIDGE

Why does it work?

plays the "role" of weights
but always in (0,1)
(can't explode!)

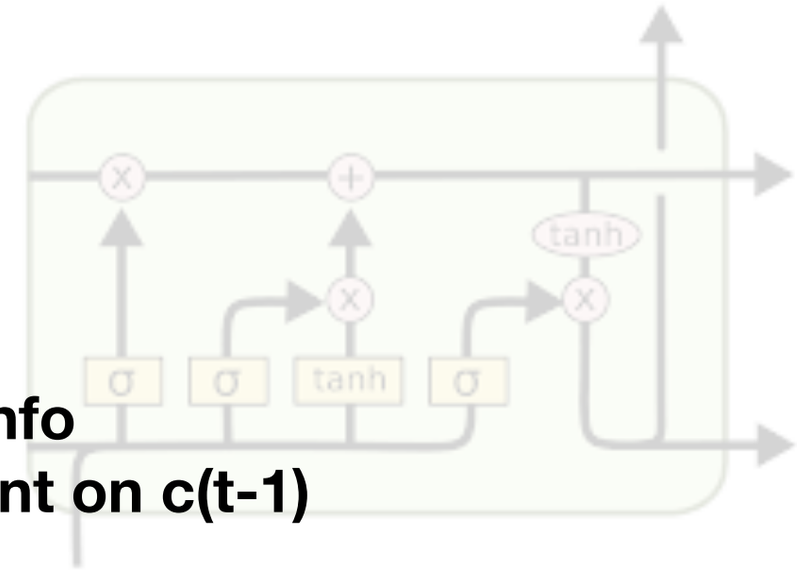
no tanh.
activation = id

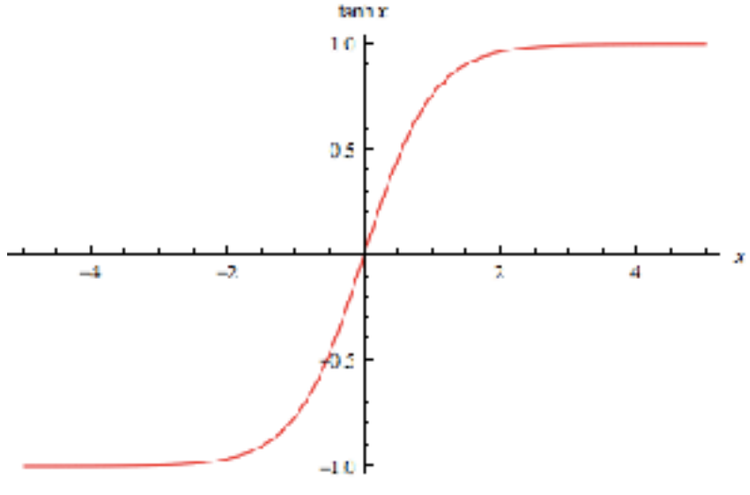
new info
- not dependent on c(t-1)

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

vs.

$$h_t = tanh(Uh_{t-1} + Wx_t)$$





DeepMind

UNIVERSITY OF CAMBRIDGE

How many weights in an LSTM?

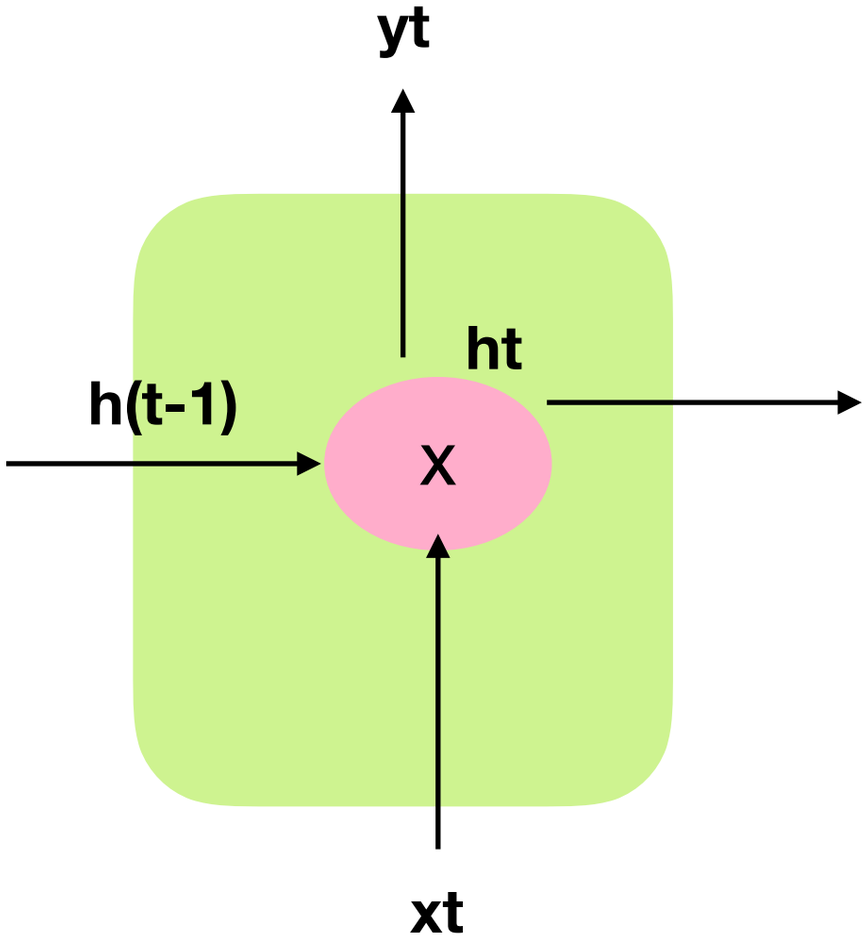
1: 100,000 words in my vocab

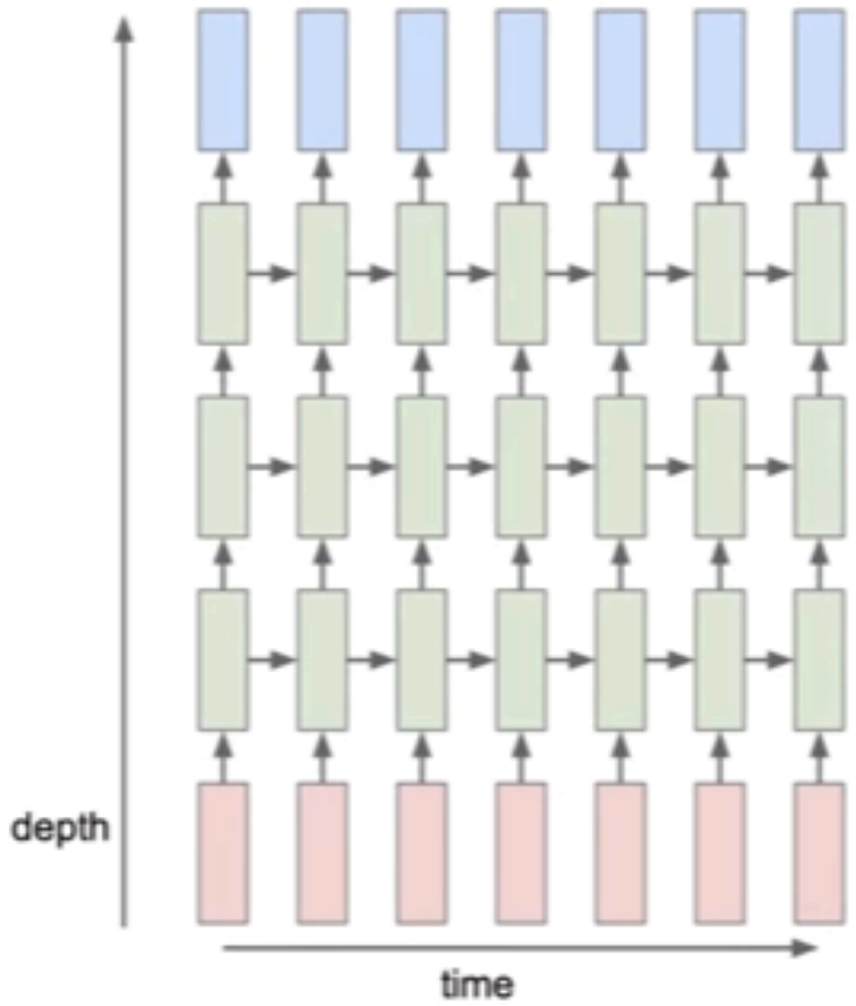
2: layers C_t have 500 units

DeepMind

UNIVERSITY OF CAMBRIDGE

Can we go deeper?



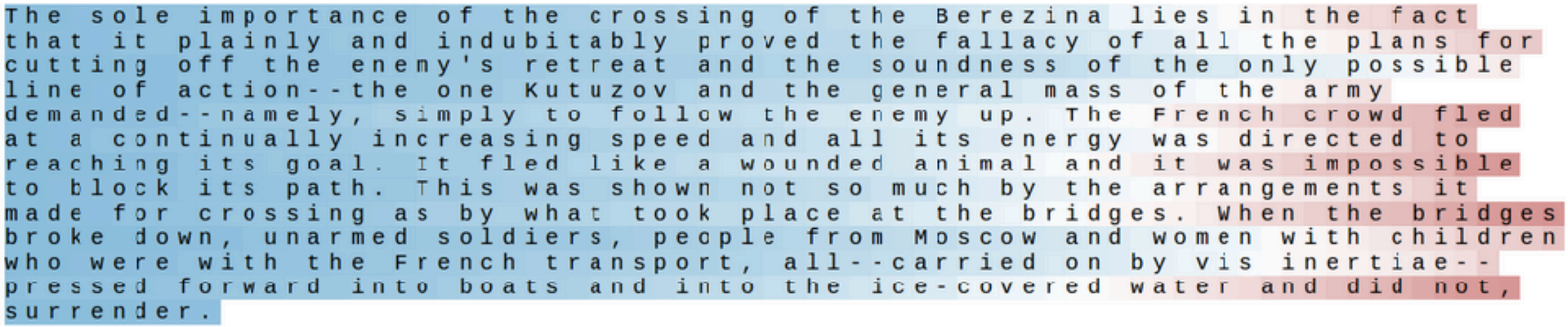


DeepMind

UNIVERSITY OF CAMBRIDGE

Understanding recurrent nets….

Cell sensitive to position in line:



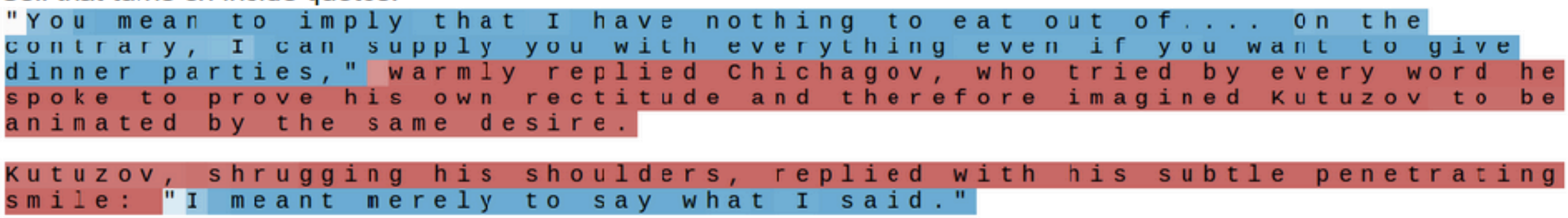
Thanks to
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

DeepMind

UNIVERSITY OF CAMBRIDGE

Understanding recurrent nets….

Cell that turns on inside quotes:



Thanks to
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

DeepMind

UNIVERSITY OF CAMBRIDGE

Understanding recurrent nets….

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
        siginfo_t *info)
{
  int sig = next_signal(pending, mask);
  if (sig) {
    if (current->notifier) {
      if (sigismember(current->notifier_mask, sig)) {
        if (!(current->notifier)(current->notifier_data)) {
          clear_thread_flag(TIF_SIGPENDING);
          return 0;
        }
      }
    }
    collect_signal(sig, pending, info);
  }
  return sig;
}
```

Thanks to
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

DeepMind

UNIVERSITY OF CAMBRIDGE

Understanding recurrent nets….

A large portion of cells are not easily interpretable. Here is a typical example:

```
/*  Unpack  a  filter  field's  string  representation  from  user-space
 *  buffer.  */
char  *audit_unpack_string(void  **bufp,  size_t  *remain,  size_t  len)
{
  char  *str;
  if  (!*bufp  ||  (len  ==  0)  ||  (len  >  *remain))
    return  ERR_PTR(-EINVAL);
  /*  Of  the  currently  implemented  string  fields,  PATH_MAX
   *  defines  the  longest  valid  length.
   */
```

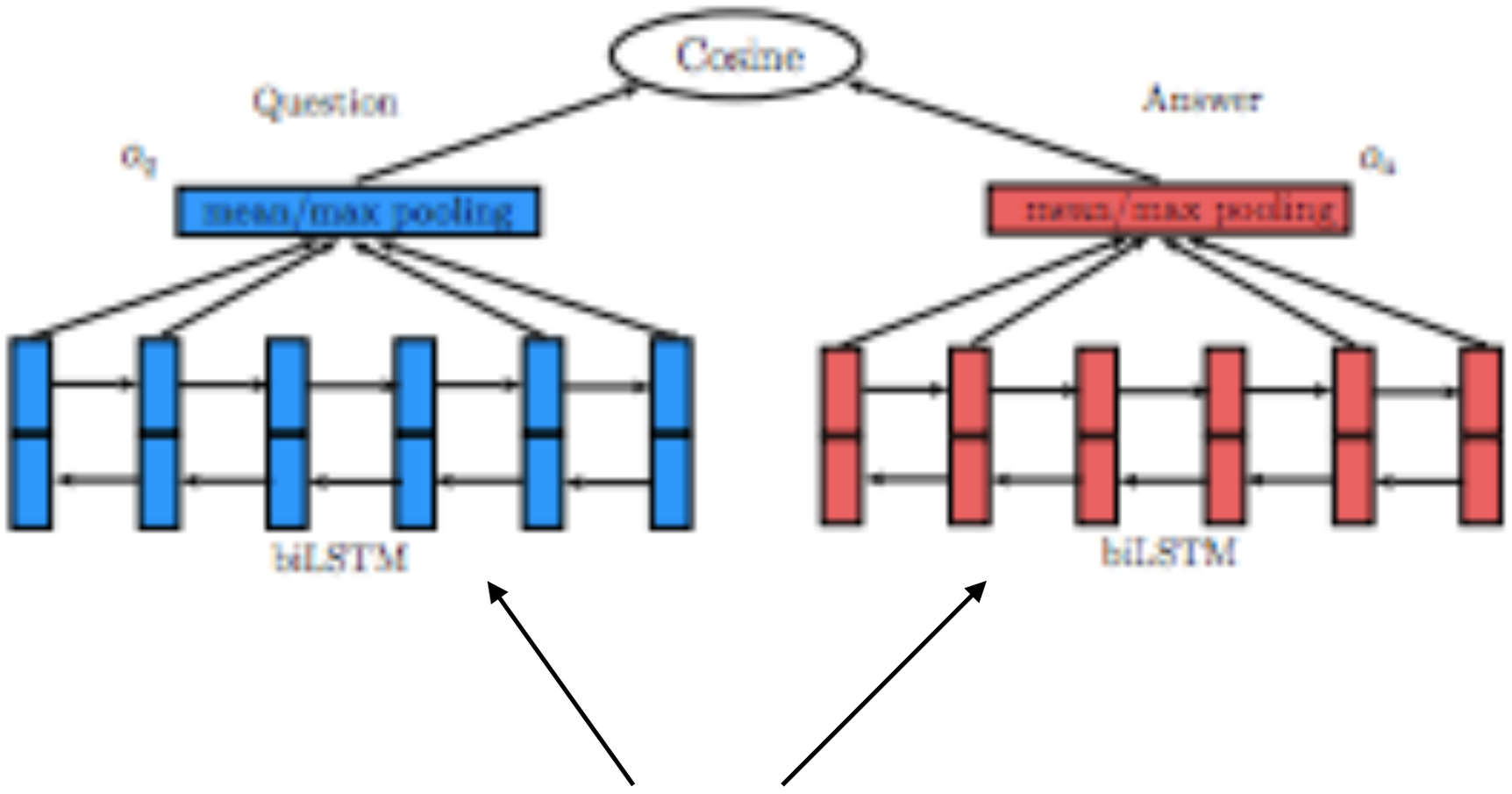
Thanks to
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

DeepMind

UNIVERSITY OF CAMBRIDGE

Why is it called Long Short Term Memory?

NLP for the LSTM generation



“Read” the sentence both ways and concatenate h_t at each time step

“LSTM-based deep learning models for non-factoid answer selection.” Ming Tan et al. 2015

References

A problem

“Learning long-term dependencies with gradient descent is difficult". Bengio et al. 1994

A solution

“Long short-term memory”. Hochreiter and Schmidhuber, 1997

A(nother) cottage industry

“LSTM-based deep learning models for non-factoid answer selection.” Ming Tan et al. 2015

A very clear explanation of LSTMs

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