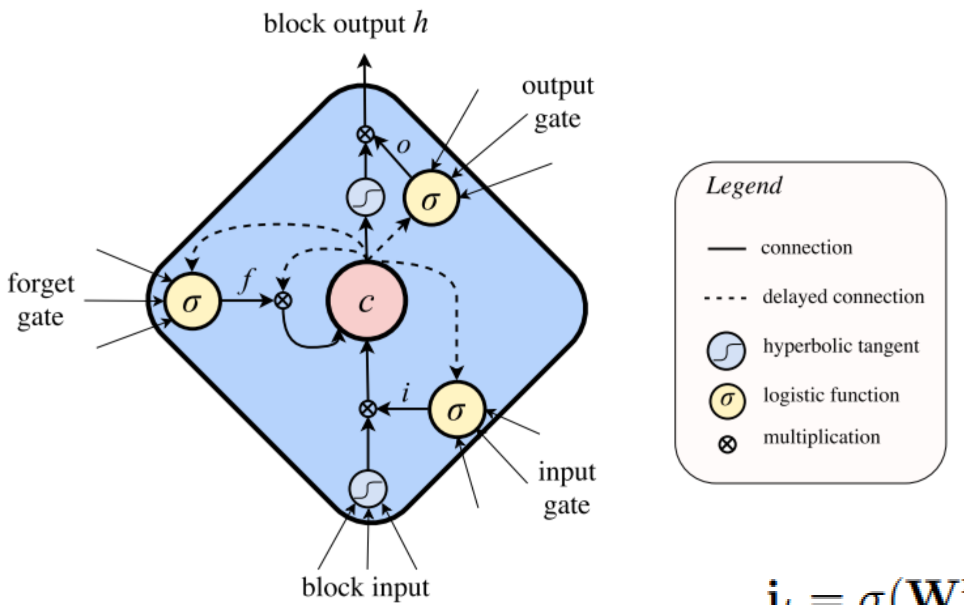
INTRODUCTION TO LSTM

LSTM

LSTM

- Most popular recurrent node type is *Long Short Term Memory* (LSTM)
- LSTM includes also *gates*, which can turn on/off the history and a few additional inputs.

LSTM block



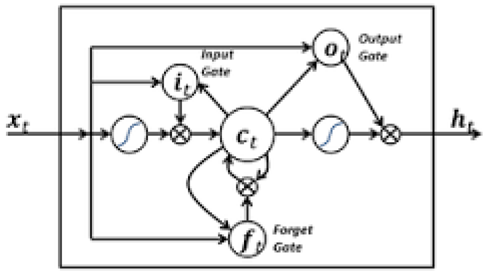
$$\mathbf{i}_t = \sigma(\mathbf{W}^{\mathrm{xi}}\mathbf{x}_t + \mathbf{W}^{\mathrm{hi}}\mathbf{h}_{t-1} + \mathbf{W}^{\mathrm{ci}}\mathbf{c}_{t-1} + \mathbf{b}^{\mathrm{i}})$$

$$\mathbf{f}_t = \sigma(\mathbf{W}^{\mathrm{xf}}\mathbf{x}_t + \mathbf{W}^{\mathrm{hf}}\mathbf{h}_{t-1} + \mathbf{W}^{\mathrm{cf}}\mathbf{c}_{t-1} + \mathbf{b}^{\mathrm{f}})$$

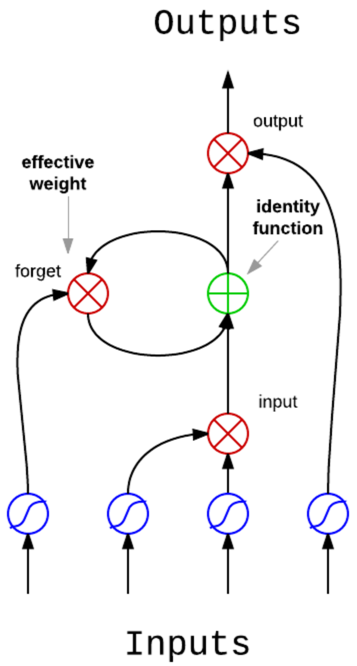
$$\mathbf{c}_t = \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \tanh(\mathbf{W}^{\mathrm{xc}}\mathbf{x}_t + \mathbf{W}^{\mathrm{hc}}\mathbf{h}_{t-1} + \mathbf{b}^{\mathrm{c}})$$

$$\mathbf{o}_t = \sigma(\mathbf{W}^{\mathrm{xo}}\mathbf{x}_t + \mathbf{W}^{\mathrm{ho}}\mathbf{h}_{t-1} + \mathbf{W}^{\mathrm{co}}\mathbf{c}_t + \mathbf{b}^{\mathrm{o}})$$

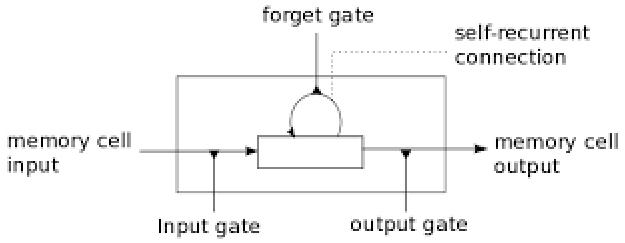
$$\mathbf{h}_t = \mathbf{o}_t \tanh(\mathbf{c}_t)$$



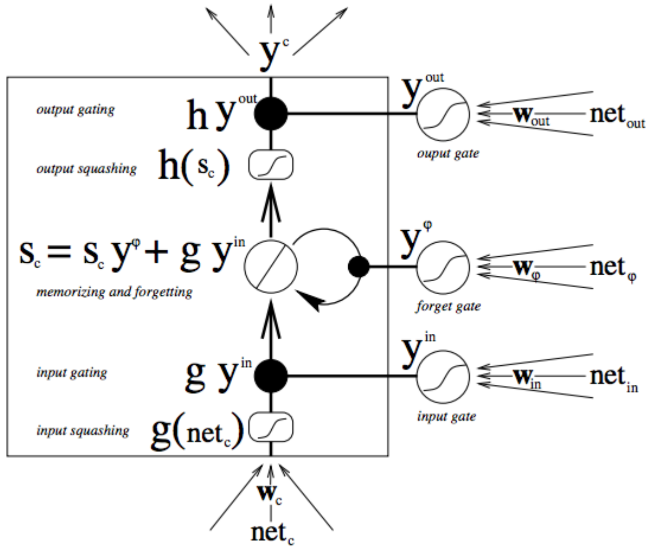
x_t
Input Gate
i_t
c_t
o_t
Output Gate
h_t
f_t
Forget Gate



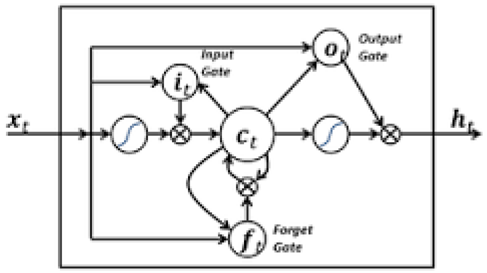
Outputs
output
effective weight
identity function
forget
input
Inputs



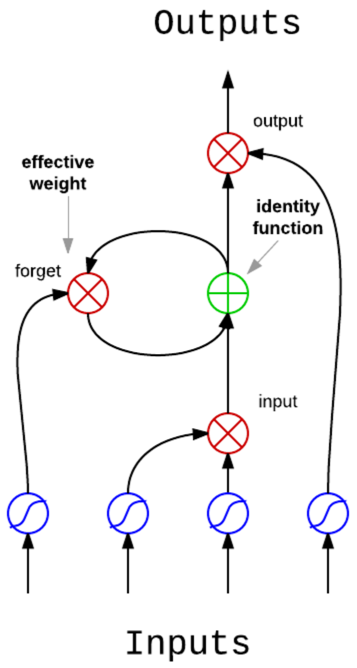
forget gate
self-recurrent connection
memory cell input
memory cell output
Input gate
output gate



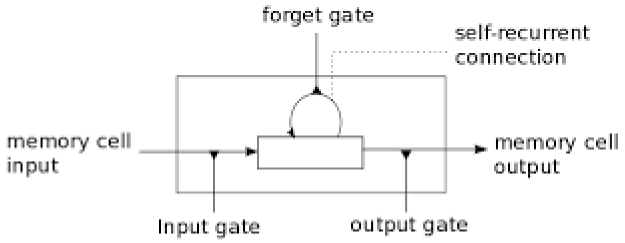
y^c
output gating
$h\,y^{out}$
y^{out}
w_{out}
net_{out}
ouput gate
output squashing
$h(s_c)$
$s_c = s_c\,y^{\varphi} + g\,y^{in}$
y^{φ}
w_{φ}
net_{φ}
memorizing and forgetting
forget gate
y^{in}
w_{in}
net_{in}
input gating
$g\,y^{in}$
input gate
input squashing
$g(net_c)$
w_c
net_c



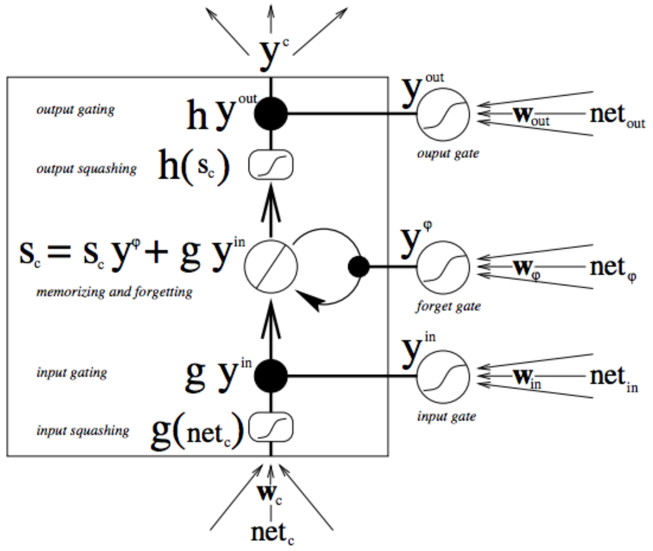
x_t
Input Gate
i_t
o_t
Output Gate
c_t
h_t
f_t
Forget Gate



Outputs
output
effective weight
identity function
forget
input
Inputs



forget gate
self-recurrent connection
memory cell input
memory cell output
Input gate
output gate

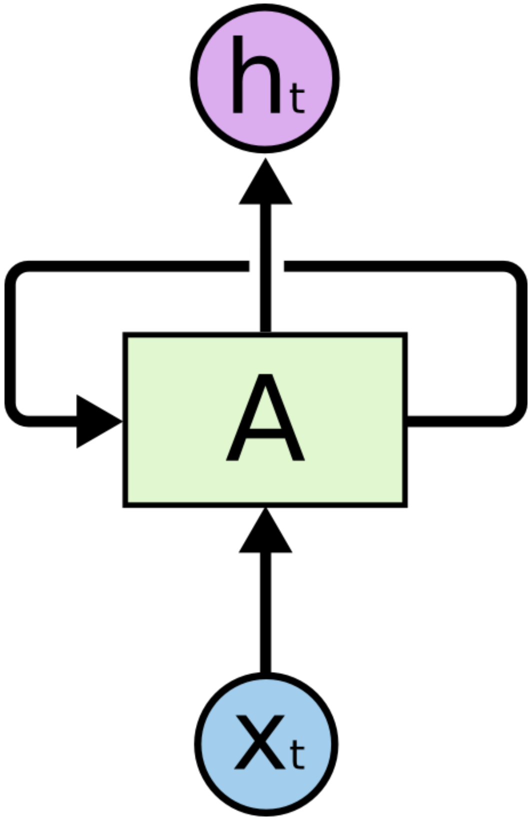


y^c
output gating
$h\,y^{out}$
y^{out}
w_{out}
net_{out}
ouput gate
output squashing
$h(s_c)$
$s_c = s_c\,y^{\varphi} + g\,y^{in}$
memorizing and forgetting
y^{φ}
w_{φ}
net_{φ}
forget gate
input gating
$g\,y^{in}$
y^{in}
w_{in}
net_{in}
input gate
input squashing
$g(net_c)$
w_c
net_c

?

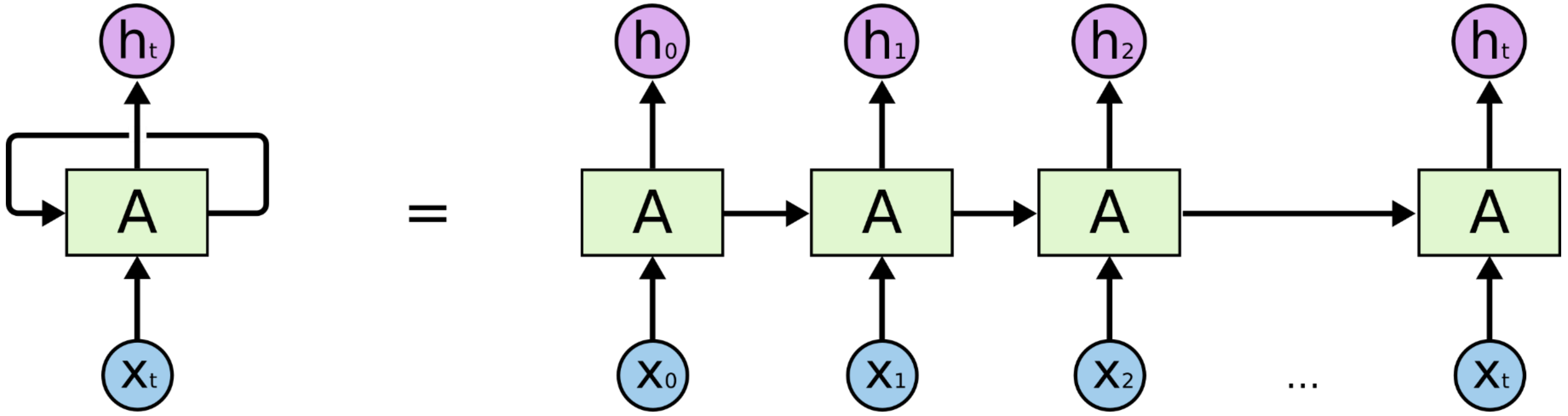
RNN

RNN

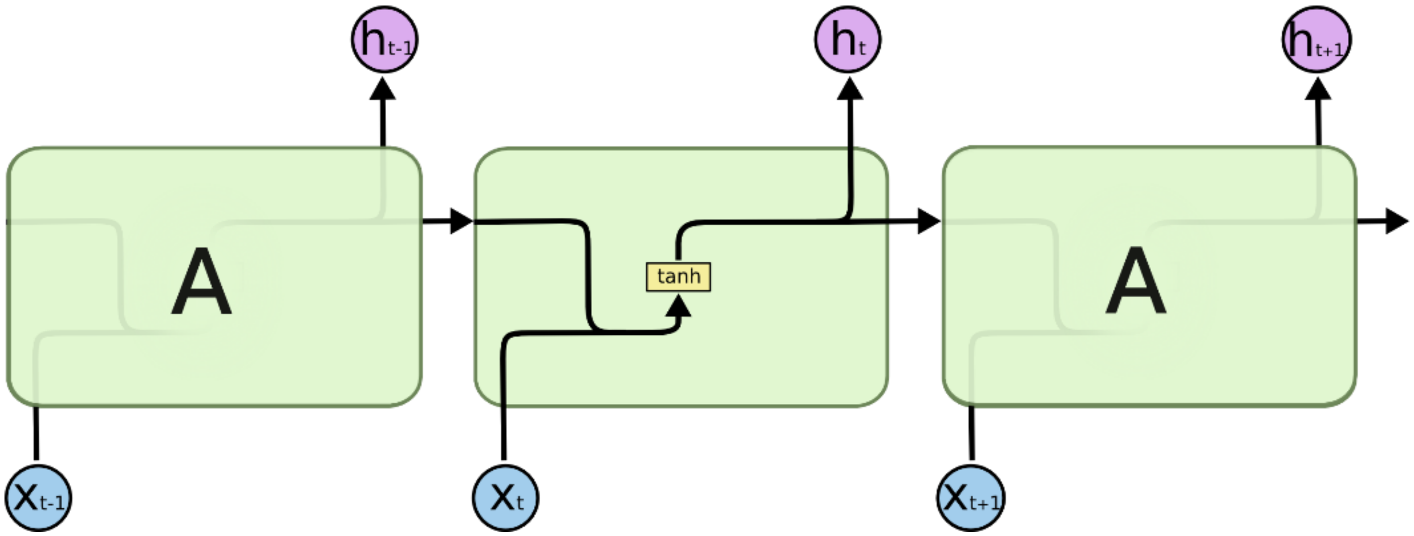


colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN

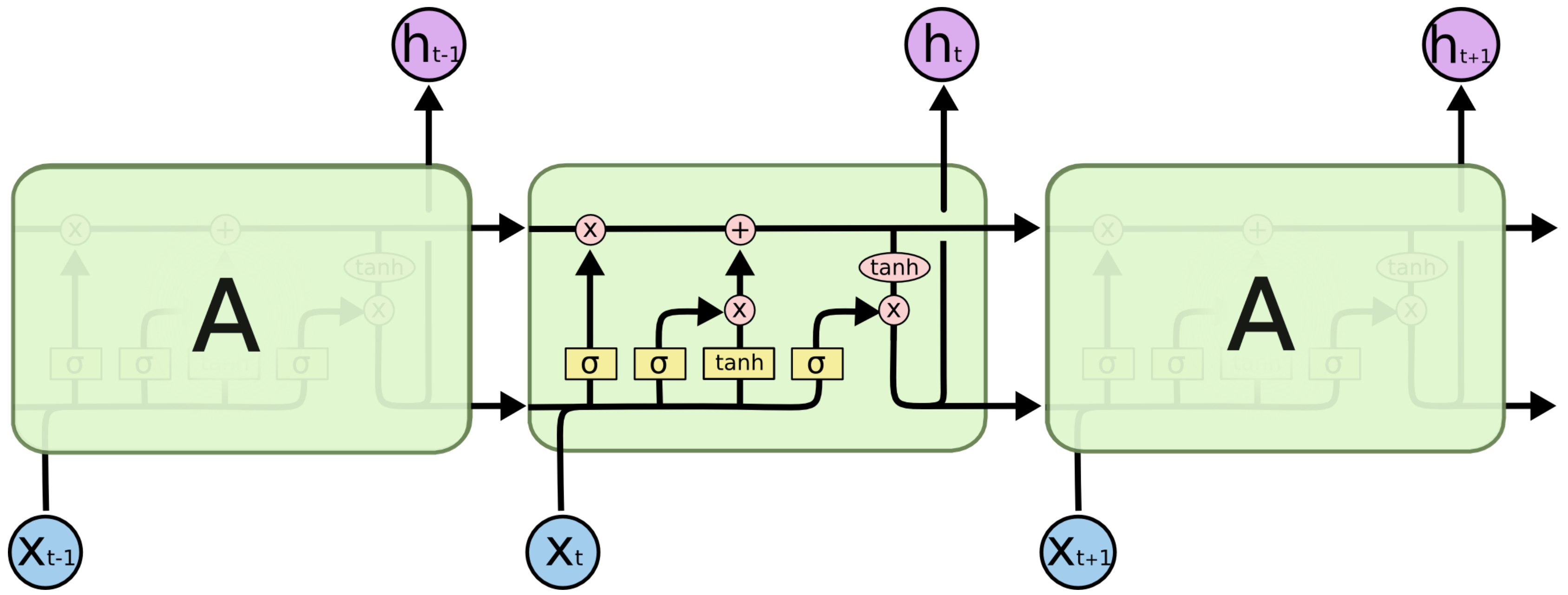


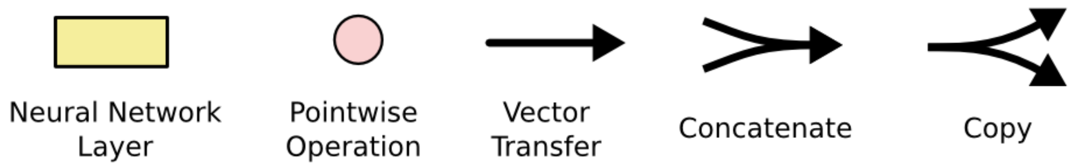
Repeating module in RNN



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

Repeating module in LSTM





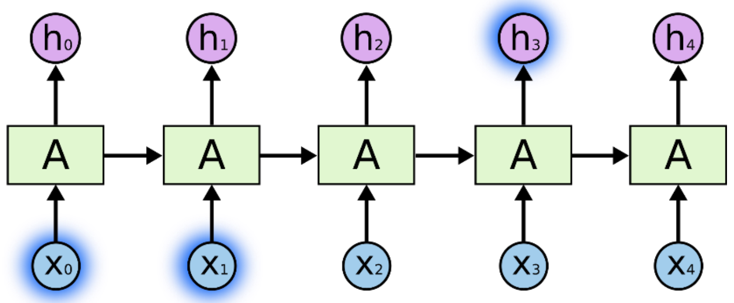
ⓘ colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM MOTIVATIONS

Problems of Long-term dependencies

- One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task

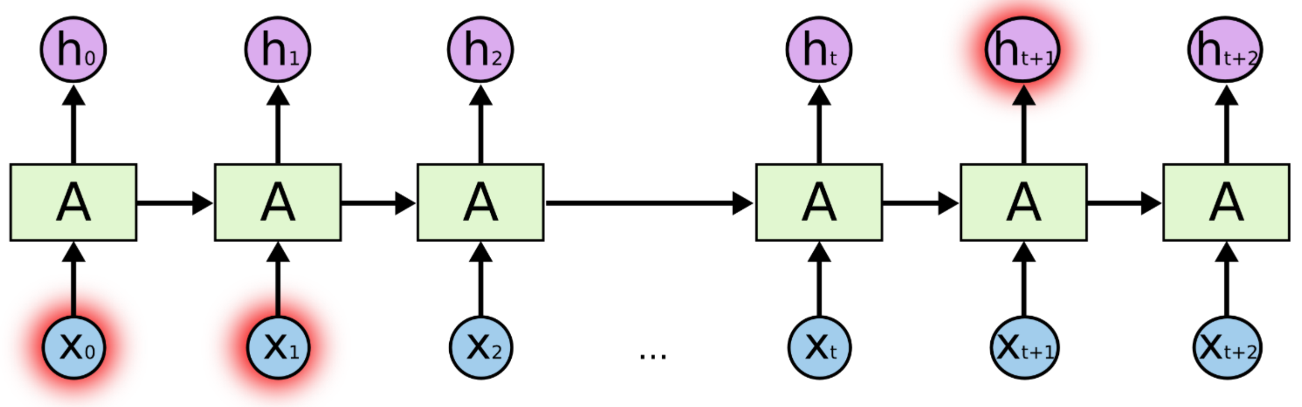
- Example: the prediction of the next word based on the previous ones
  - "the clouds are in the ***sky***,"



ⓘ colah.github.io/posts/2015-08-Understanding-LSTMs/

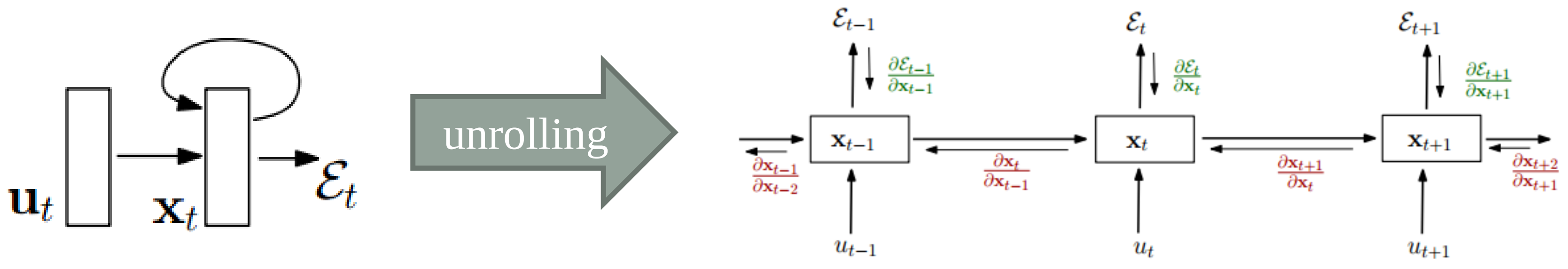
Problems of Long-term dependencies

- Large gap between the relevant information and the place that it's needed
  - "I grew up in France… I speak fluent ***French***."



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

Training recurrent networks



$\mathbf{u}_t$
$\mathbf{x}_t$
$\mathcal{E}_t$
unrolling
$\mathcal{E}_{t-1}$
$\frac{\partial \mathcal{E}_{t-1}}{\partial \mathbf{x}_{t-1}}$
$\mathcal{E}_t$
$\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t}$
$\mathcal{E}_{t+1}$
$\frac{\partial \mathcal{E}_{t+1}}{\partial \mathbf{x}_{t+1}}$
$\frac{\partial \mathbf{x}_{t-1}}{\partial \mathbf{x}_{t-2}}$
$\mathbf{x}_{t-1}$
$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_{t-1}}$
$\mathbf{x}_t$
$\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{x}_t}$
$\mathbf{x}_{t+1}$
$\frac{\partial \mathbf{x}_{t+2}}{\partial \mathbf{x}_{t+1}}$
u_{t-1}
u_t
u_{t+1}

$$\mathbf{x}_t = F(\mathbf{x}_{t-1}, \mathbf{u}_t, \theta)$$

$$\mathbf{x}_t = \mathbf{W}_{rec}\sigma(\mathbf{x}_{t-1}) + \mathbf{W}_{in}\mathbf{u}_t + \mathbf{b}$$

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{1 \le t \le T} \frac{\partial \mathcal{E}_t}{\partial \theta}$$

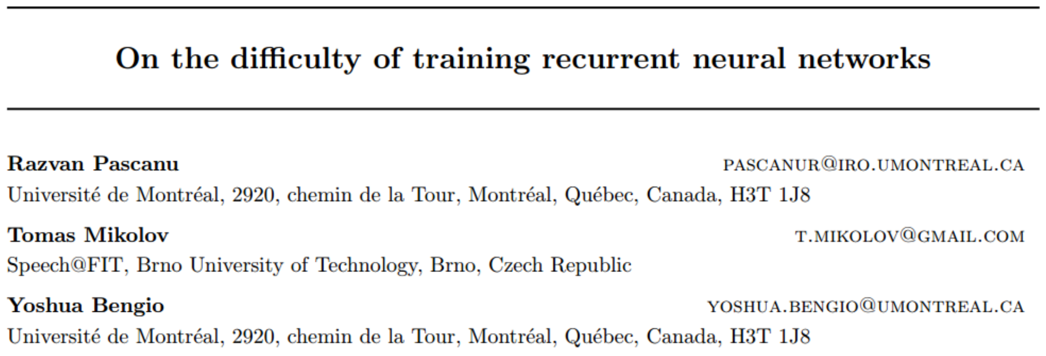
$$\frac{\partial \mathcal{E}_t}{\partial \theta} = \sum_{1 \le k \le t} \left(\frac{\partial \mathcal{E}_t}{\partial \mathbf{x}_t} \frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} \frac{\partial^+ \mathbf{x}_k}{\partial \theta} \right)$$

$$\frac{\partial \mathbf{x}_t}{\partial \mathbf{x}_k} = \prod_{t \ge i > k} \frac{\partial \mathbf{x}_i}{\partial \mathbf{x}_{i-1}} = \prod_{t \ge i > k} \mathbf{W}_{rec}^T diag(\sigma'(\mathbf{x}_{i-1}))$$

Exploding and vanishing gradients

- The exploding gradients problem
  - the large increase in the norm of the gradient during training.
- The vanishing gradients
  - long term components go exponentially fast to norm 0

On the difficulty of training RNN

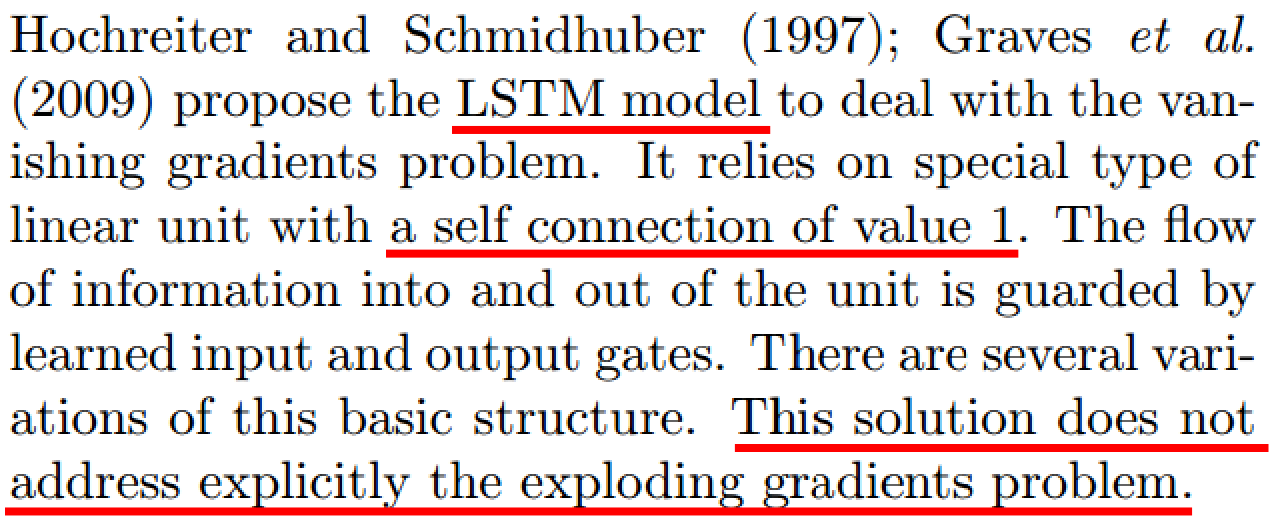


On the difficulty of training recurrent neural networks

Razvan Pascanu PASCANUR@IRO.UMONTREAL.CA
Université de Montréal, 2920, chemin de la Tour, Montréal, Québec, Canada, H3T 1J8

Tomas Mikolov T.MIKOLOV@GMAIL.COM
Speech@FIT, Brno University of Technology, Brno, Czech Republic

Yoshua Bengio YOSHUA.BENGIO@UMONTREAL.CA
Université de Montréal, 2920, chemin de la Tour, Montréal, Québec, Canada, H3T 1J8

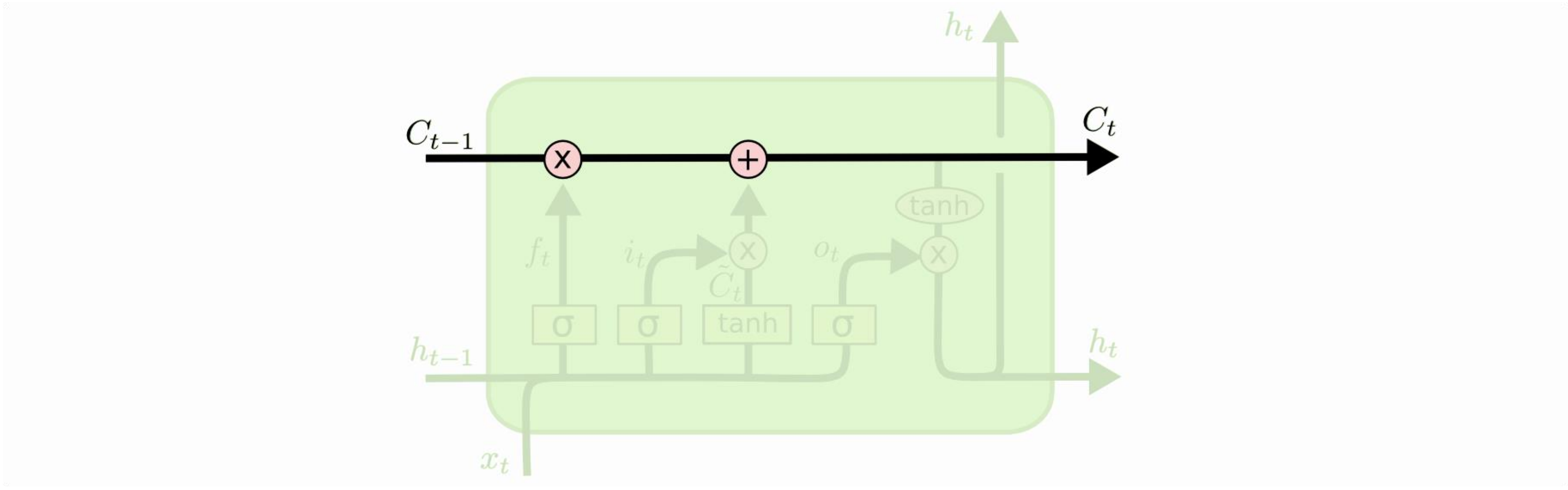


Hochreiter and Schmidhuber (1997); Graves *et al.* (2009) propose the LSTM model to deal with the vanishing gradients problem. It relies on special type of linear unit with a self connection of value 1. The flow of information into and out of the unit is guarded by learned input and output gates. There are several variations of this basic structure. This solution does not address explicitly the exploding gradients problem.

CORE IDEA BEHIND LSTMS

Cell state

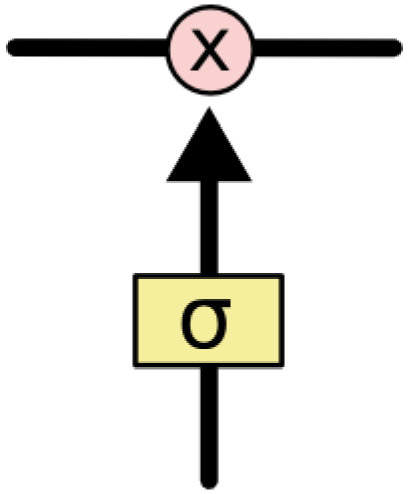
- Cell state
  - It runs straight down the entire chain, with linear interactions.
  - LSTM has the ability to remove or add information to the cell state, regulated by gates.



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

Gates

- Gates are a way to optionally let information through
  - a sigmoid neural net layer and a pointwise multiplication operation.
  - The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through.
    - A value of zero means "let nothing through,"
    - A value of one means "let everything through"

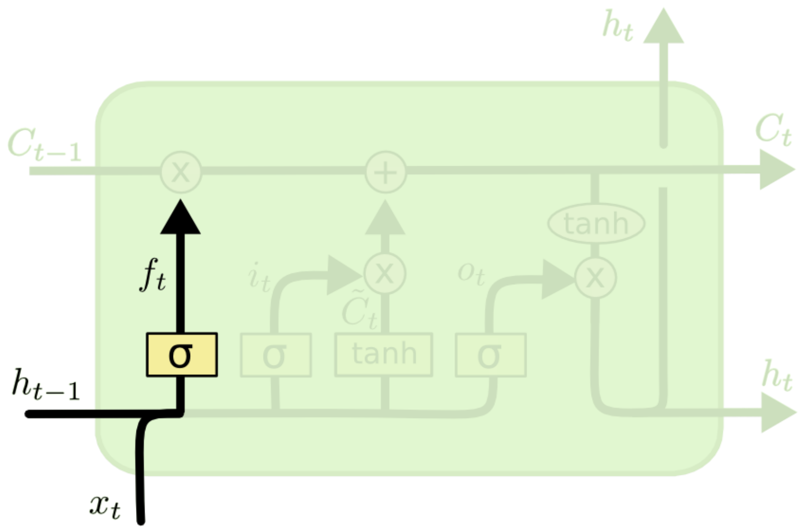


colah.github.io/posts/2015-08-Understanding-LSTMs/

STEP-BY-STEP LSTM WALK THROUGH

Forget gate layer

- Forget gate
  - to decide what information we're going to throw away from the cell state
  - this decision is made by a sigmoid layer called the "forget gate layer."



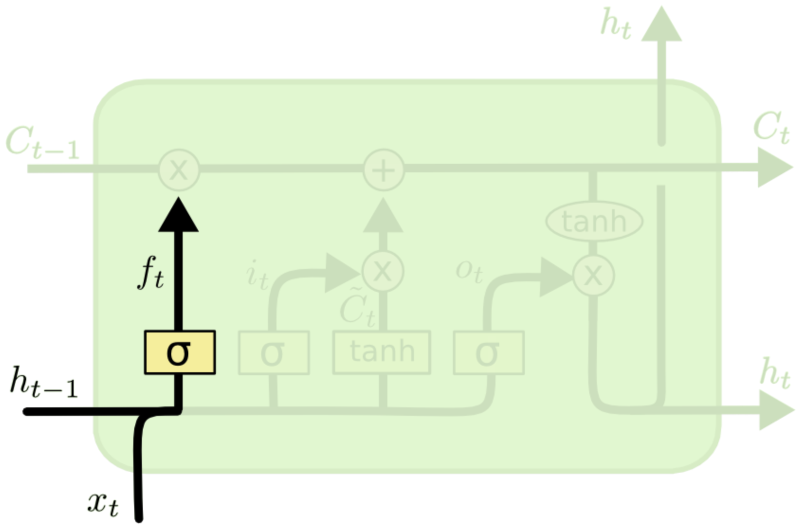
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

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

Forget gate layer

- $f_t = 1$: completely keep this information
- $f_t = 0$: completely get rid of this information

- Ex) Language model example
  - When we a new subject, we want to forget the gender information in C_{t-1}

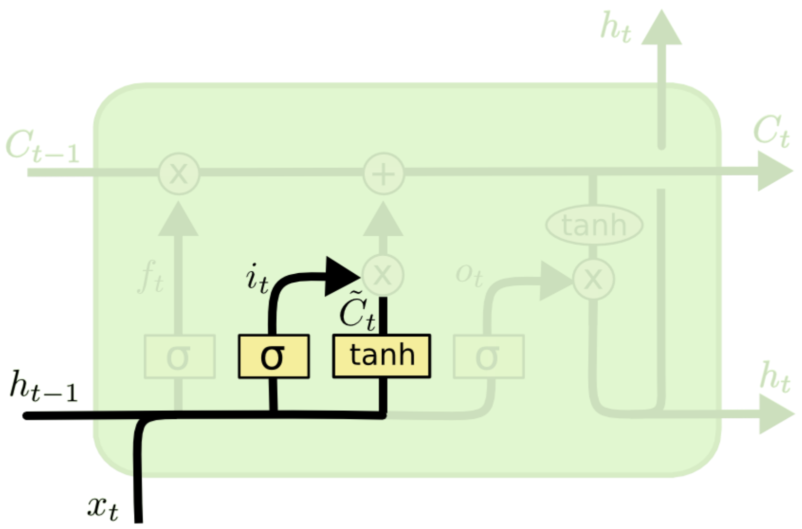


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

ⓘ colah.github.io/posts/2015-08-Understanding-LSTMs/

Input gate layer

- Input gate
  - to decide what new information we're going to store in the cell state
  - a sigmoid layer called the "input gate layer" decides which values we'll update



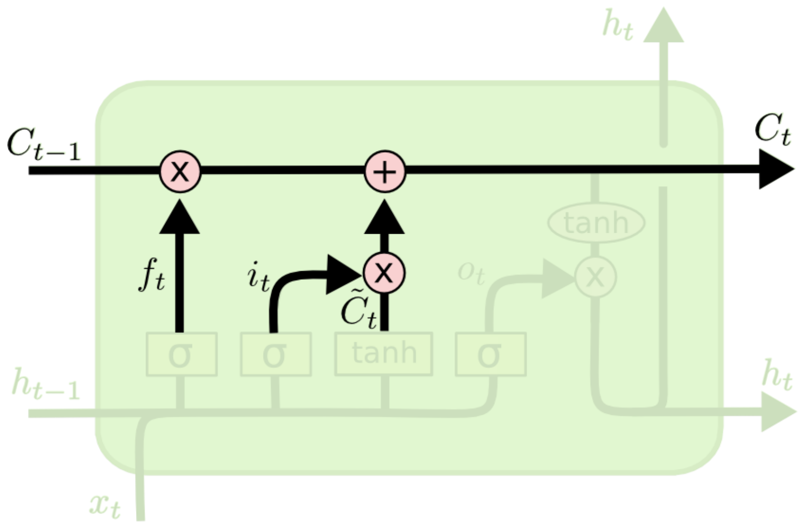
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

ⓘ colah.github.io/posts/2015-08-Understanding-LSTMs/

Cell sate update

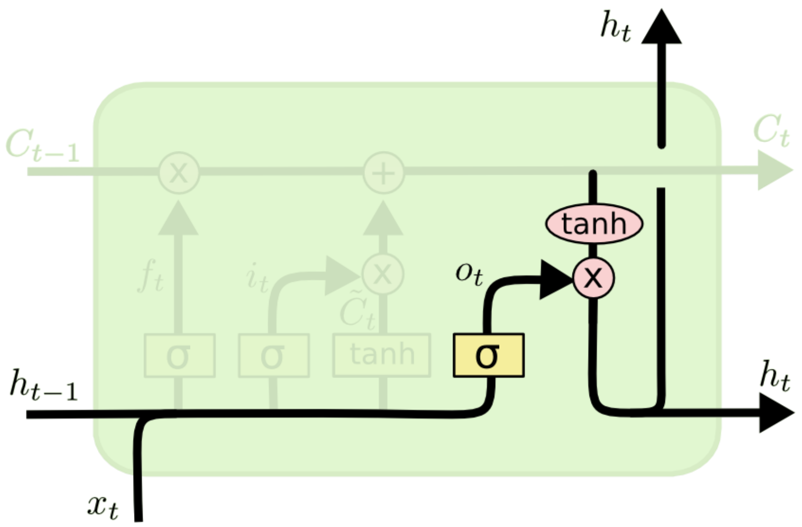
- We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add $i_t * \tilde{C}_t$.
- In the case of the language model, this is where we'd actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps.



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

Output

- We run a sigmoid layer which decides what parts of the cell state we're going to output.
- Then, we put the cell state through tanh (to push the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

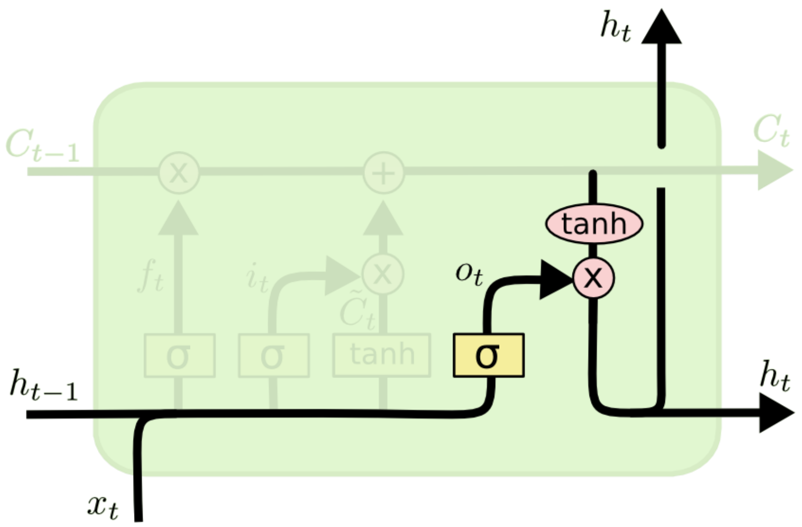


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

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

Output

- For the language model example, since it just saw a subject, it might want to output information relevant to a verb.
- For example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into.



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

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

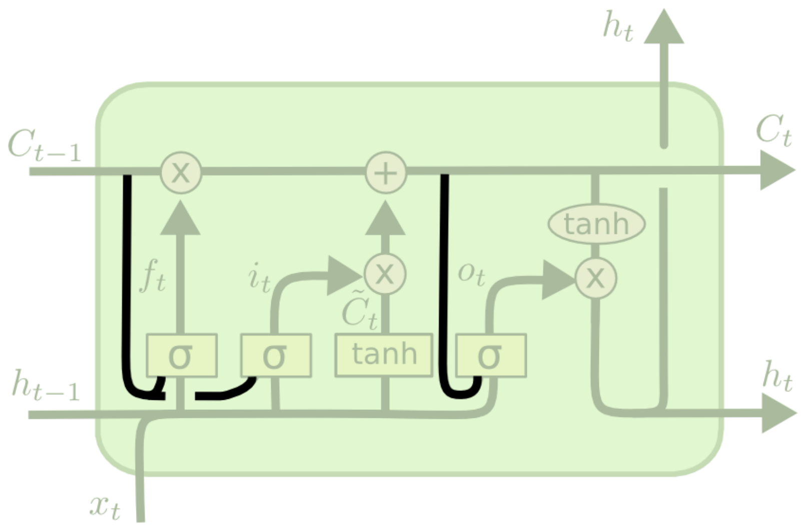
colah.github.io/posts/2015-08-Understanding-LSTMs/

VARIANTS OF LONG SHORT TERM MEMORY

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

Peephole connection

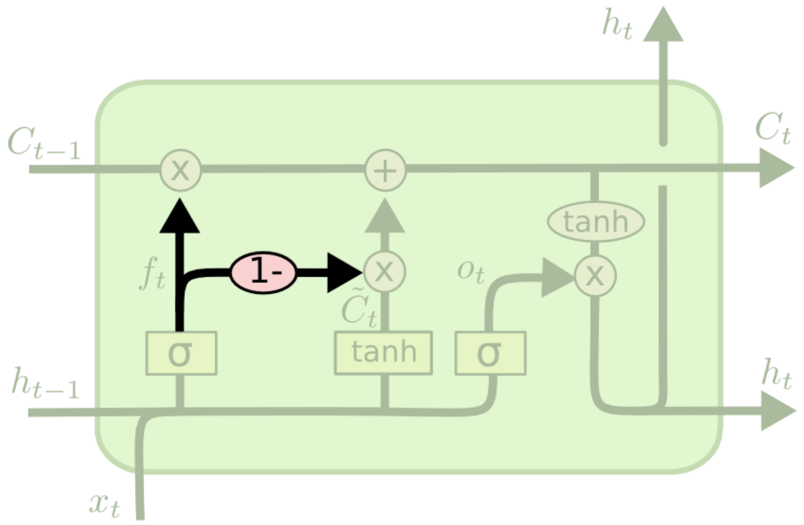
- Let the gate layers look at the cell state.



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

Coupled forget and input gates

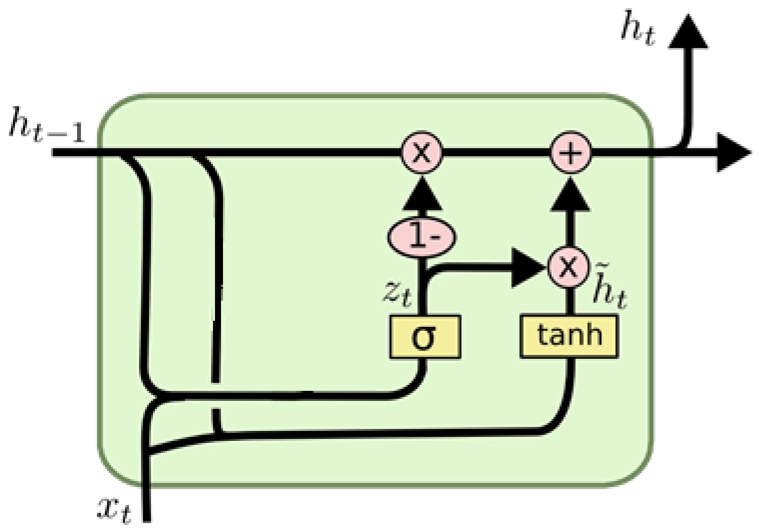
- We only forget when we're going to input something in its place. We only input new values to the state when we forget something older.



$$C_t = f_t * C_{t-1} + (\mathbf{1} - \boldsymbol{f_t}) * \tilde{C}_t$$

Gated Recurrent Unit

- It combines the forget and input gates into a single "update gate." It also merges the cell state and hidden state, and makes some other changes.



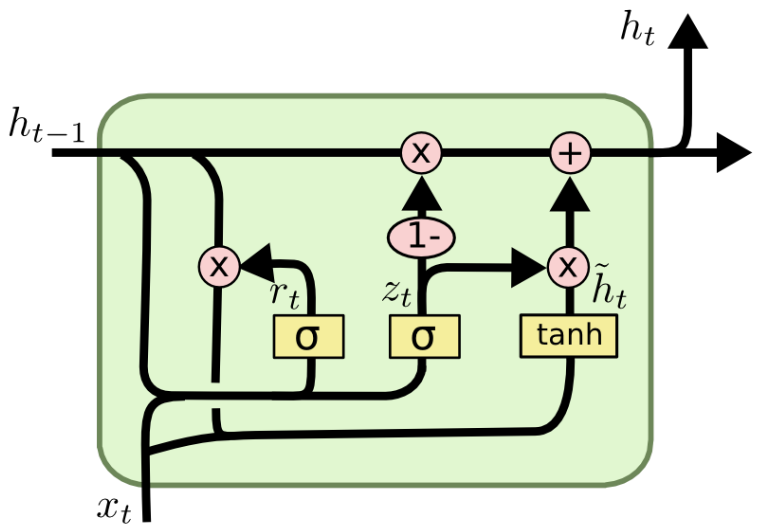
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

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

Gated Recurrent Unit

- It combines the forget and input gates into a single "update gate." It also merges the cell state and hidden state, and makes some other changes.



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

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

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

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

VARIANTS OF LSTM – BLOCK DIAGRAMS

ⓘ colah.github.io/posts/2015-08-Understanding-LSTMs/

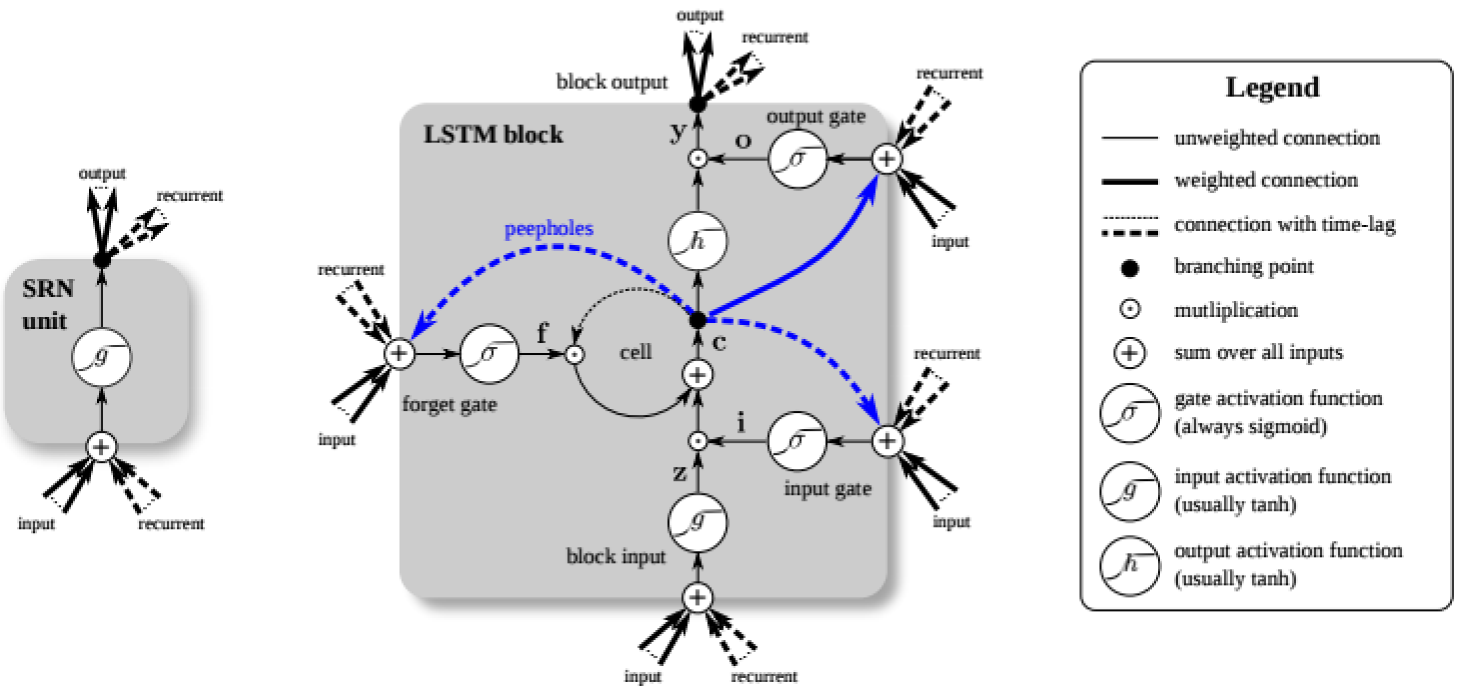
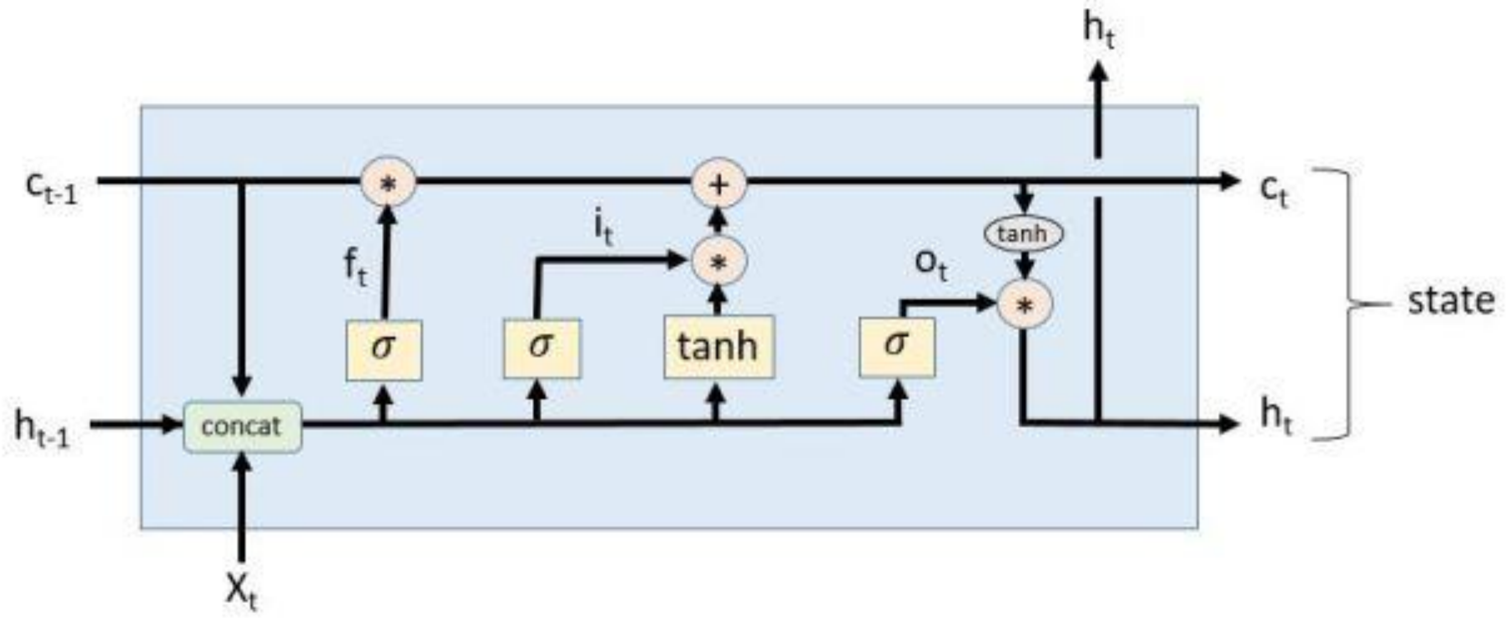
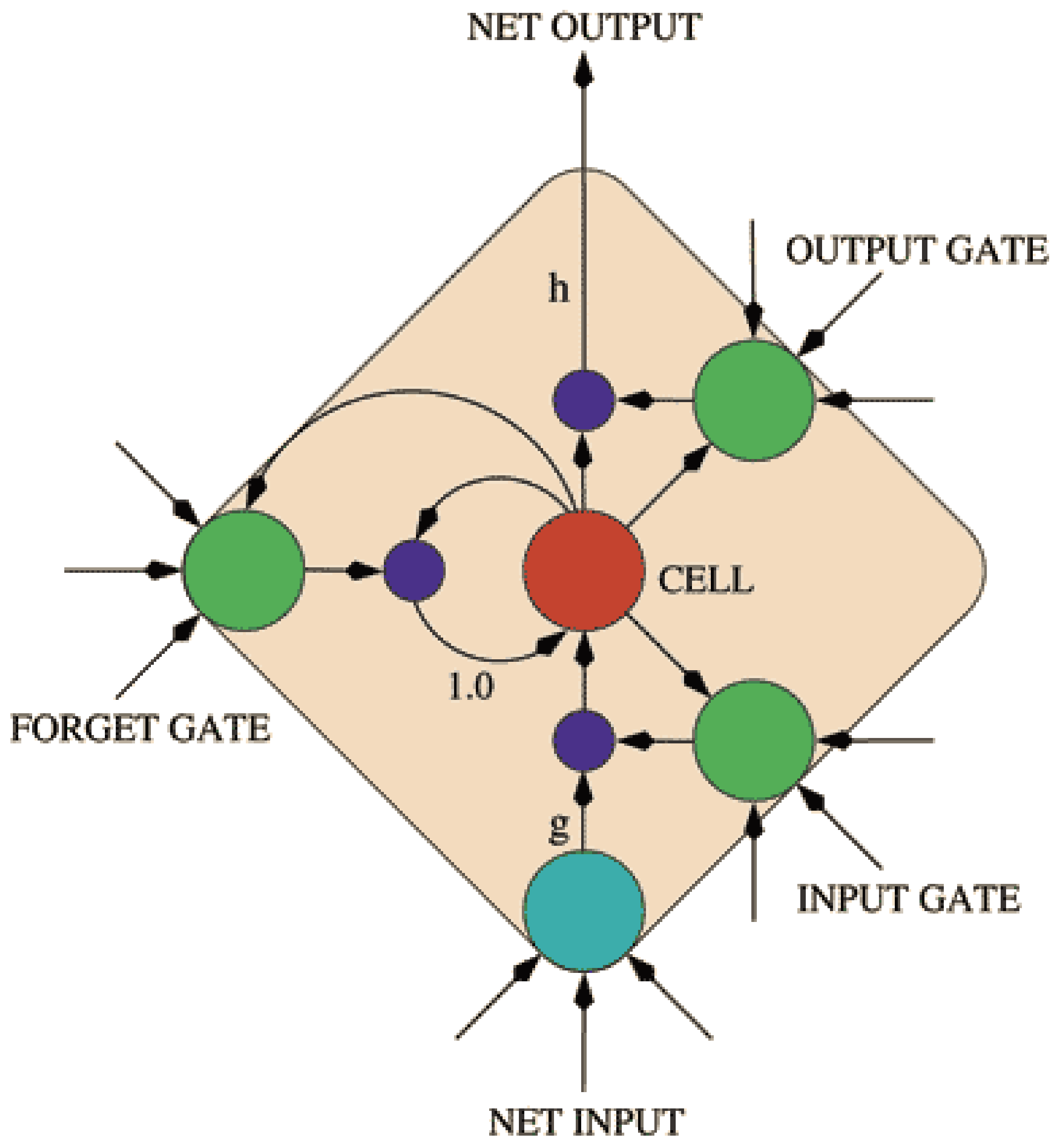
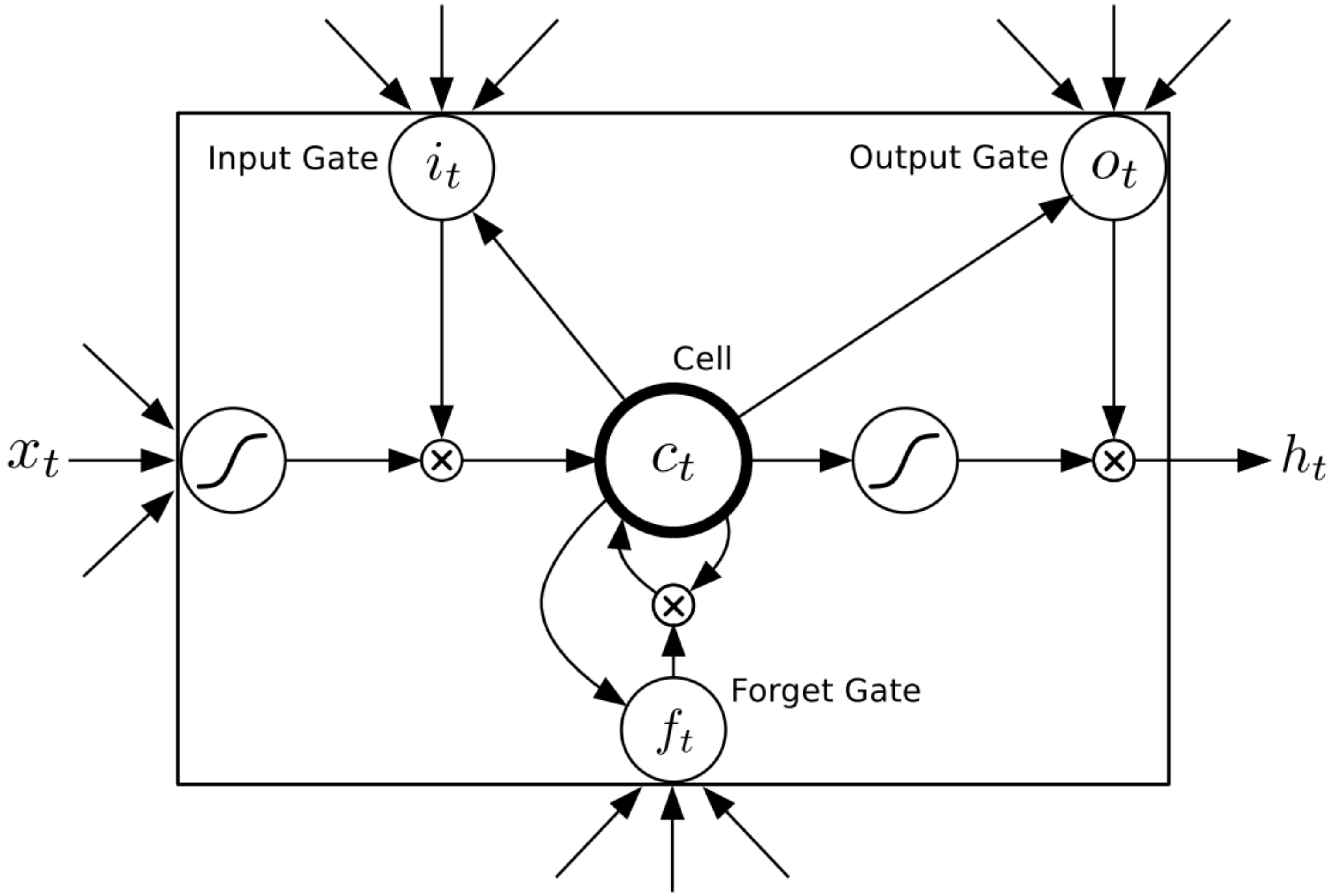


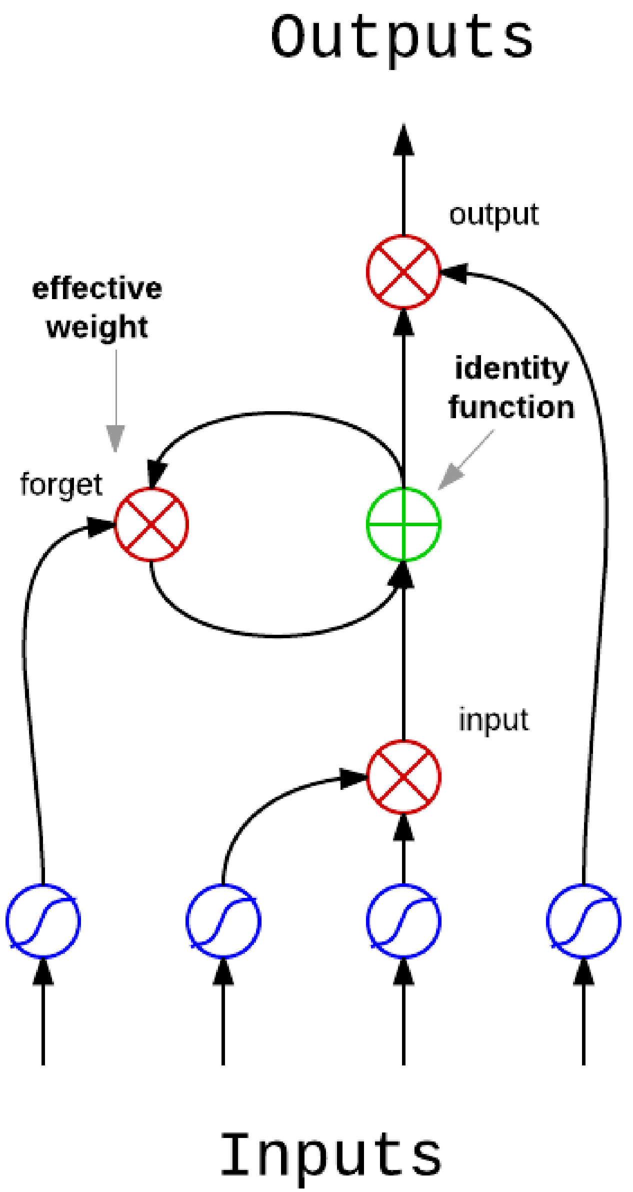
Figure 1. Detailed schematic of the Simple Recurrent Network (SRN) unit (left) and a Long Short-Term Memory block (right) as used in the hidden layers of a recurrent neural network.



h_t
c_{t-1}
c_t
i_t
f_t
o_t
tanh
σ
σ
tanh
σ
state
h_{t-1}
concat
h_t
X_t


NET OUTPUT
OUTPUT GATE
h
CELL
FORGET GATE
1.0
INPUT GATE
g
NET INPUT


Input Gate
i_t
Output Gate
o_t
Cell
x_t
c_t
h_t
Forget Gate
f_t


Outputs
output
effective weight
identity function
forget
input
Inputs

Example

- w/o peephole connection
- w/   peephole connection

Summary

- RNNs allow a lot of flexibility in architecture design
- Common to use LSTM or GRU: their additive interactions improve gradient flow
- Backward flow of gradients in RNN can explode or vanish.
- Exploding is controlled with gradient clipping. Vanishing is controlled with additive interactions (LSTM)