Why do we like LSTMs?

Tiago Pimentel

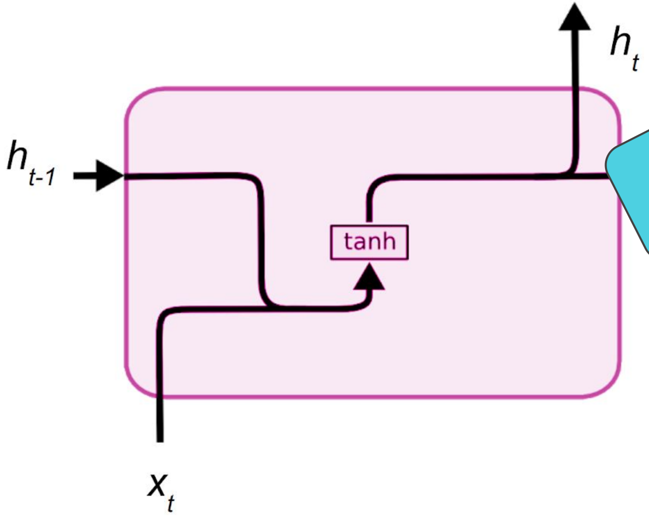
UNIVERSITY OF CAMBRIDGE

ETH zürich

③ Recurrent Neural Networks

(Vanilla) Recurrent Neural Networks

- There are many ways of framing an RNN, but at its core it is just a non-linear combination of the recurrent state and the inputs.



Recurrent hidden state

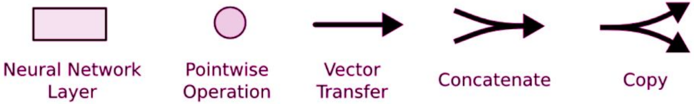
Previous hidden state

Input at time t

$$\mathbf{h}_t = g(\Theta[\mathbf{h}_{t-1}; \mathbf{x}_t])$$

where $\Theta \in \mathbb{R}^{d \times d}$ is a recurrence matrix,
g is a non-linearity (e.g., tanh)

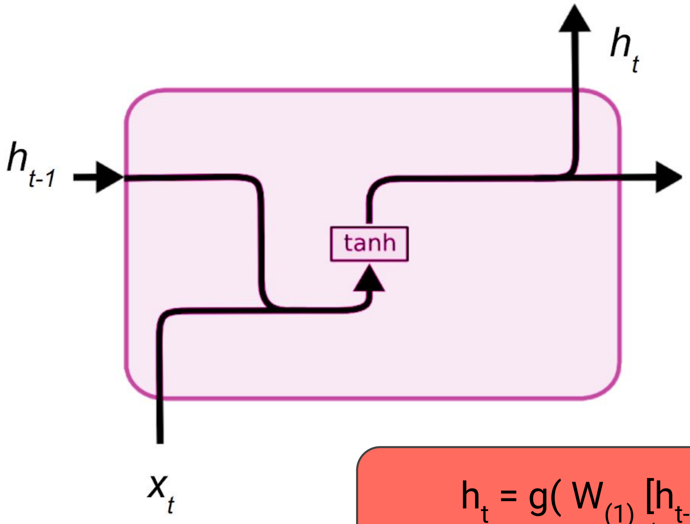
from Christopher Olah's blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/



③ Recurrent Neural Networks

(Vanilla) Recurrent Neural Networks

- There are many ways of framing an RNN, but at its core it is just a non-linear combination of the recurrent state and the inputs.



Elman recurrence (Elman, 1990):

$$\mathbf{h}_t = g(\Theta\, \mathbf{h}_{t-1} + \mathbf{x}_t)$$

Variant:

$$\mathbf{h}_t = g(\Theta[\mathbf{h}_{t-1}; \mathbf{x}_t])$$

where $\Theta \in \mathbb{R}^{d \times d}$ is a recurrence matrix,
g is a non-linearity (e.g., tanh)

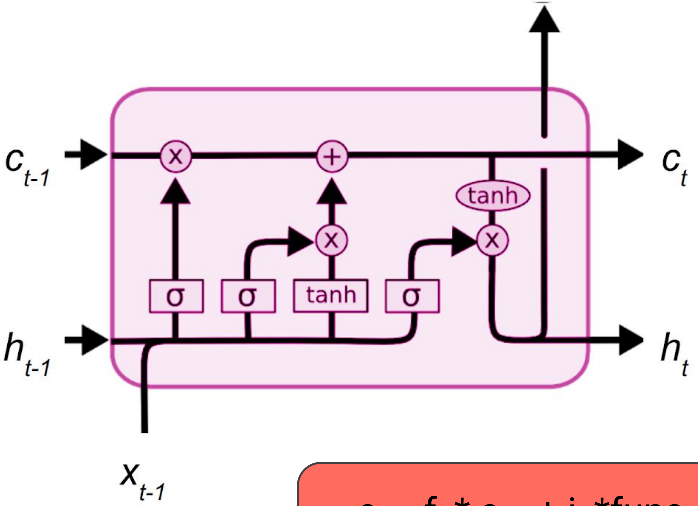
$h_t = g(W_{(1)} [h_{t-1}; x_t])$
$o_t = \mathrm{softmax}(W_{(2)} * h_t)$

from Christopher Olah's blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

③ Recurrent Neural Networks

Long short-term memory (LSTM)

LSTMs (Hochreiter and Schmidhuber, 1997) have the form



Previous hidden state

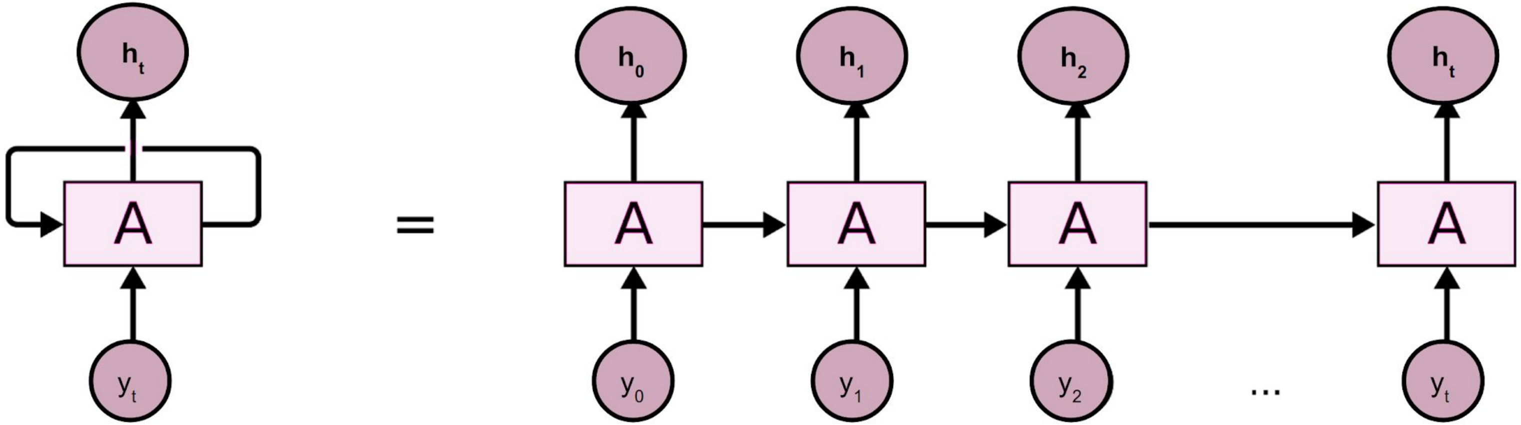
Modified input at time t

$\mathbf{f}_t$ …		…rget gate
… $[\mathbf{h}_{t-1}; \mathbf{x}_t] +$ …		…t gate
… $\tanh(\Theta_c[\mathbf{h}_{t-1}; \mathbf{x}_t]$ … $\mathbf{b}_c)$		…date candidate
$\mathbf{c}_t$	$= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$	update cell state
$\mathbf{o}_t$	$= \sigma(\Theta_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o)$	output gate
$\mathbf{h}_t$	$= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$	update hidden state

$c_t = f_t * c_{t-1} + i_t * func_1(c_{t-1}, x_t)$
$o_t = softmax(W_{(2)} * func_2(c_t, x_t))$

from Christopher Olah's blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

③ Backpropagation Through Time is Just Backpropagation



- Here *A* is whatever RNN cell we want to use (e.g., LSTM, RNN, etc.). Each timestep yields (i) an output and (ii) a recurrent connection.
- Backpropagating RNNs is the same as backpropagating through any neural network, except the parameters are shared across timesteps.
- This same idea can be used to, e.g., tie embeddings or reuse a filter in a CNN

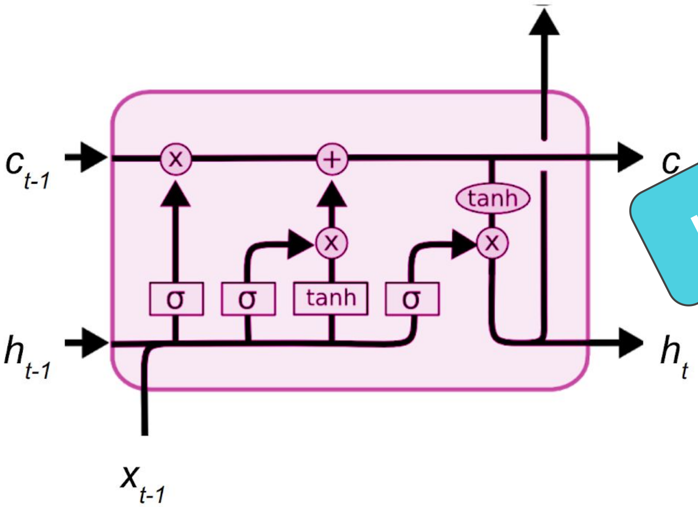
from Christopher Olah's blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

③ Recurrent Neural Networks

Original

Long short-term memory (LSTM)

LSTMs (Hochreiter and Schmidhuber, 1997) have the form



Introduced later in 1999

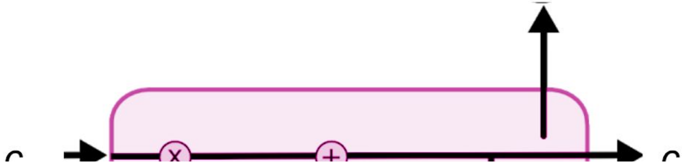
$\cancel{\ldots] + \mathbf{b}_f)}$	forget gate
$\ldots(\Theta_i[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_i)$	input gate
$\mathbf{c}_t = \ldots\mathrm{nh}(\Theta_c[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_c)$	update candidate
$\mathbf{c}_t = \cancel{\mathbf{f}_t \odot}\, \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$	update cell state
$\mathbf{o}_t = \sigma(\Theta_o[\mathbf{h}_{t-1},\mathbf{x}_t] + \mathbf{b}_o)$	output gate
$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$	update hidden state

from Christopher Olah's blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

3 Recurrent Neural Networks

Long short-term memory (LSTM)

LSTMs (Hochreiter and Schmidhuber, 1997) have the form



ced later in 1999

Learning to Forget: Continual Prediction with LSTM

Technical Report IDSIA-01-99

January, 1999

Felix A. Gers felix@idsia.ch

Jürgen Schmidhuber juergen@idsia.ch

Fred Cummins fred@idsia.ch

IDSIA, Corso Elvezia 36

6900 Lugano, Switzerland

www.idsia.ch

forget gate

input gate

update candidate

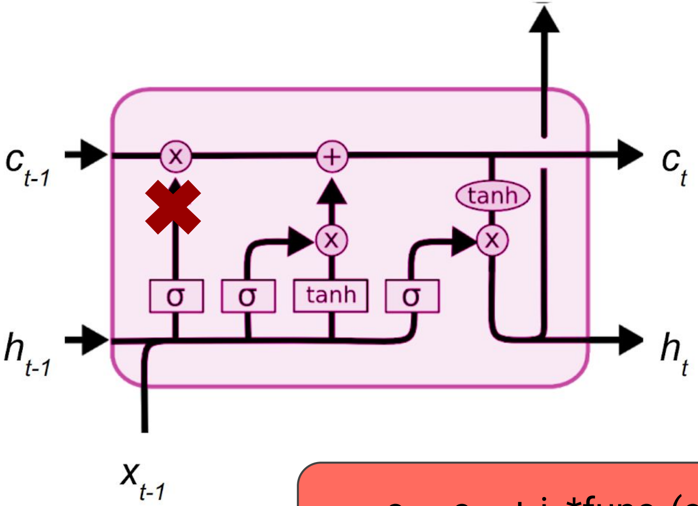
update cell state

output gate

update hidden state

Long short-term memory (LSTM)

LSTMs (Hochreiter and Schmidhuber, 1997) have the form



$\cancel{\mathbf{f}_t = \sigma(\Theta_f[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_f)}$	**forget gate**
$\mathbf{i}_t = \sigma(\Theta_i[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_i)$	**input gate**
$\tilde{\mathbf{c}}_t = \tanh(\Theta_c[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_c)$	**update candidate**
$\mathbf{c}_t = \cancel{\mathbf{f}_t \odot}\, \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$	**update cell state**
$\mathbf{o}_t = \sigma(\Theta_o[\mathbf{h}_{t-1},\mathbf{x}_t] + \mathbf{b}_o)$	**output gate**
$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$	**update hidden state**

$c_t = c_{t-1} + i_t * func_1(c_{t-1}, x_t)$

$o_t = softmax(W_{(2)} * func_2(c_t, x_t))$

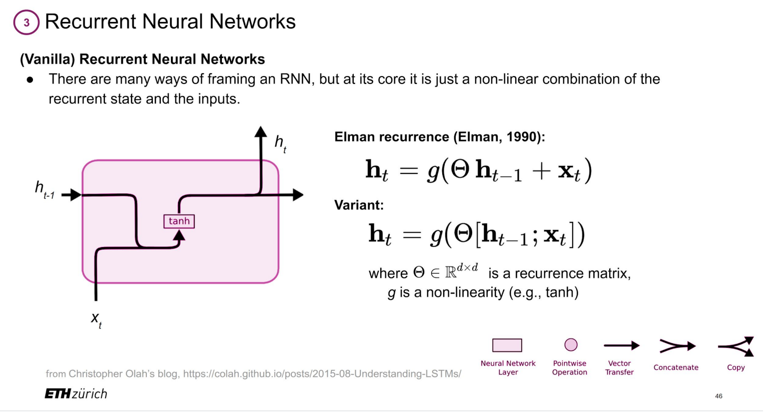
from Christopher Olah's blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Let’s go back to the Elman RNN

Which functions can Elman’s RNNs represent?

In theory, everything. They are turing complete!

-> “On the computational power of neural nets”, Siegelmann & Sontag (1995)



Let’s go back to the Elman RNN

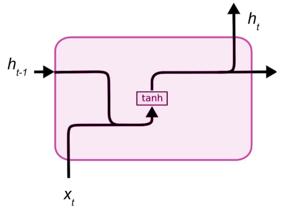
So why would we need something else?

Because some functions might not be **learnable** in practice.

③ Recurrent Neural Networks

(Vanilla) Recurrent Neural Networks

- There are many ways of framing an RNN, but at its core it is just a non-linear combination of the recurrent state and the inputs.



Elman recurrence (Elman, 1990):

$$\mathbf{h}_t = g(\Theta\, \mathbf{h}_{t-1} + \mathbf{x}_t)$$

Variant:

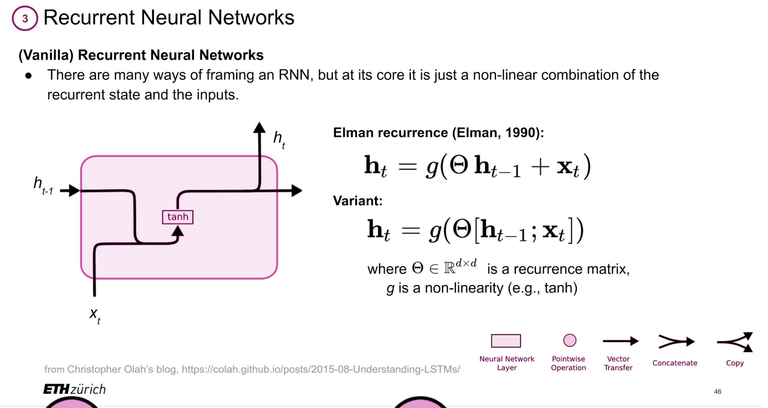
$$\mathbf{h}_t = g(\Theta[\mathbf{h}_{t-1}; \mathbf{x}_t])$$

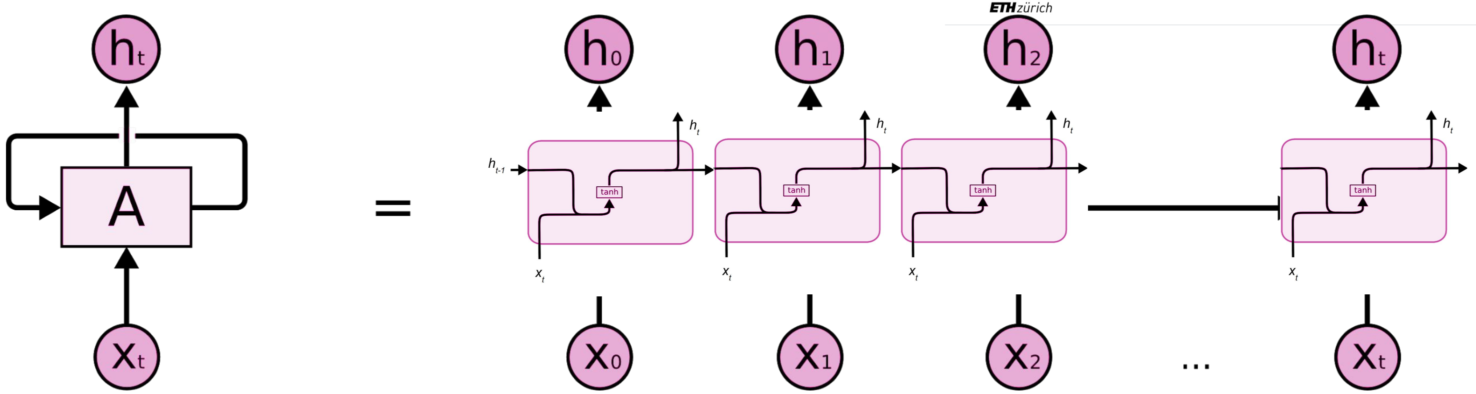
where $\Theta \in \mathbb{R}^{d \times d}$ is a recurrence matrix, g is a non-linearity (e.g., tanh)

from Christopher Olah’s blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Let’s go back to the Elman RNN

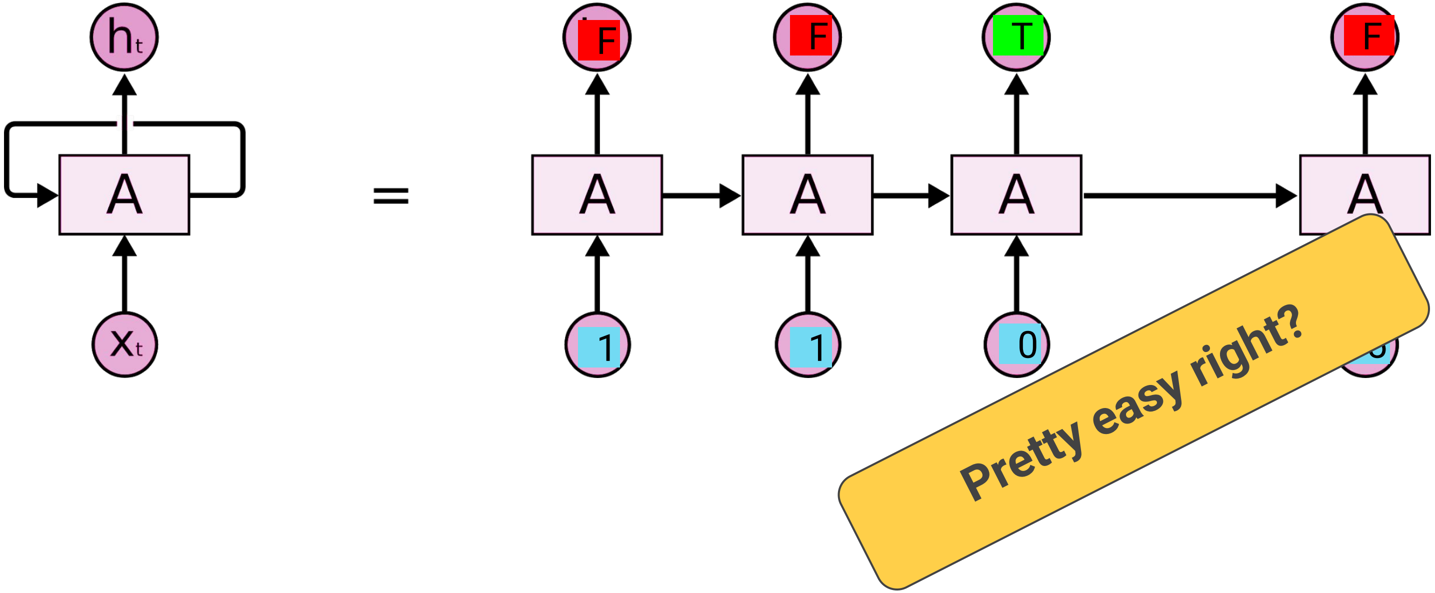
Let’s unfold this RNN on time





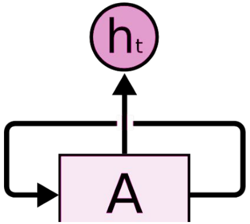
A simple example

Our task is to identify the number 0 in the input

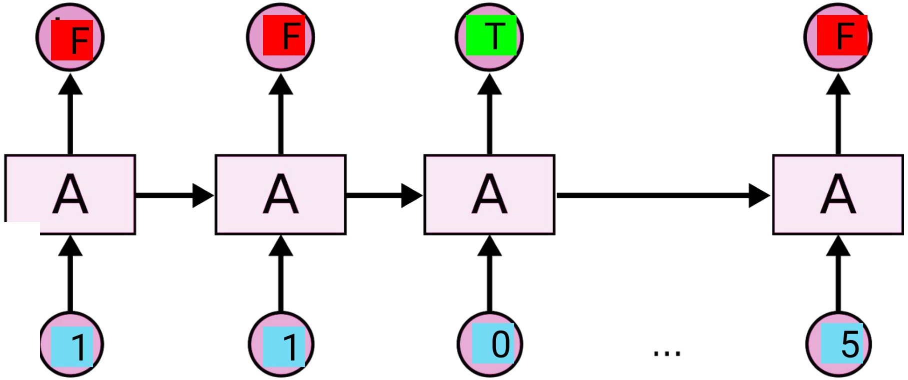


A simple example

Our task is to identify the number 0 in the input



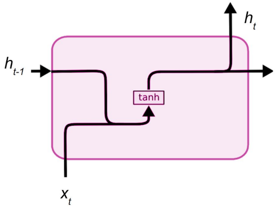
=



3 Recurrent Neural Networks

(Vanilla) Recurrent Neural Networks

- There are many ways of framing an RNN, but at its core it is just a non-linear combination of the recurrent state and the inputs.



Elman recurrence (Elman, 1990):

$$\mathbf{h}_t = g(\Theta\,\mathbf{h}_{t-1} + \mathbf{x}_t)$$

Variant:

$$\mathbf{h}_t = g(\Theta[\mathbf{h}_{t-1}; \mathbf{x}_t])$$

where $\Theta \in \mathbb{R}^{d \times d}$ is a recurrence matrix, g is a non-linearity (e.g., tanh)

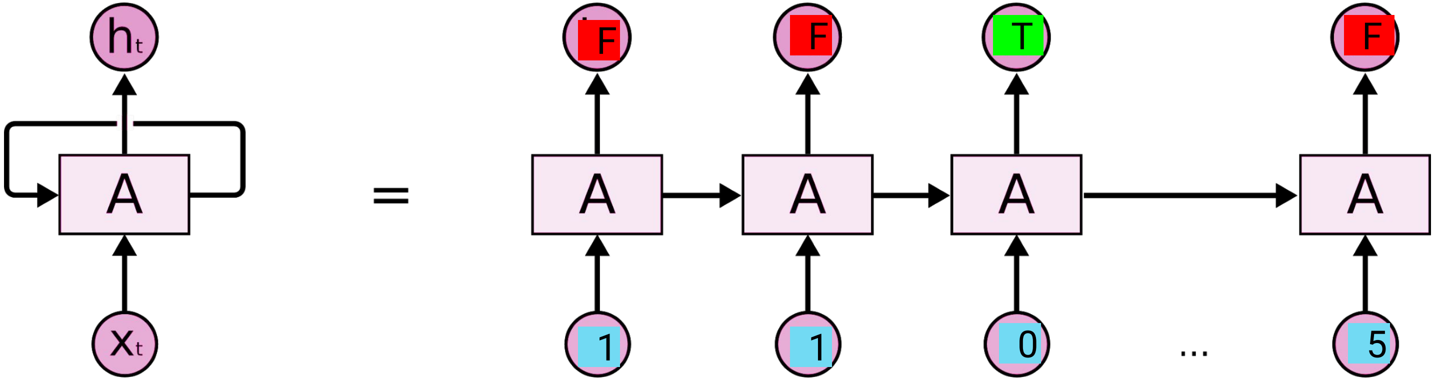
from Christopher Olah's blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

ETH zürich

$$h_t = g(\, W_{(1)}\, [h_{t-1}; x_t])$$
$$o_t = \mathrm{softmax}(W_{(2)} * h_t)$$

A simple example

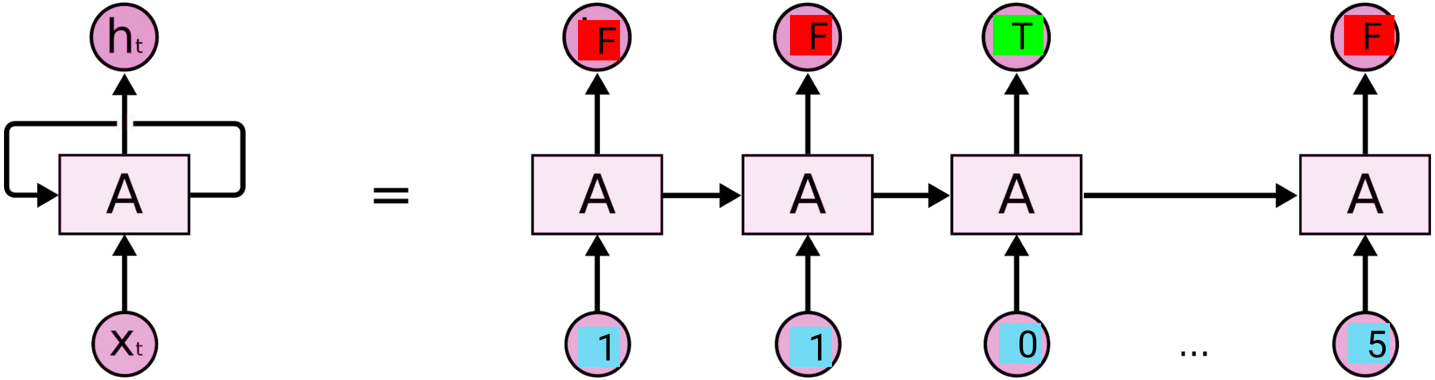
Our task is to identify the number 0 in the input



$$h_t = g(\, W_{(1)}\, [h_{t-1}; x_t])$$
$$o_t = \text{softmax}(W_{(2)} * h_t)$$

A simple example

Our task is to identify the number 0 in the input

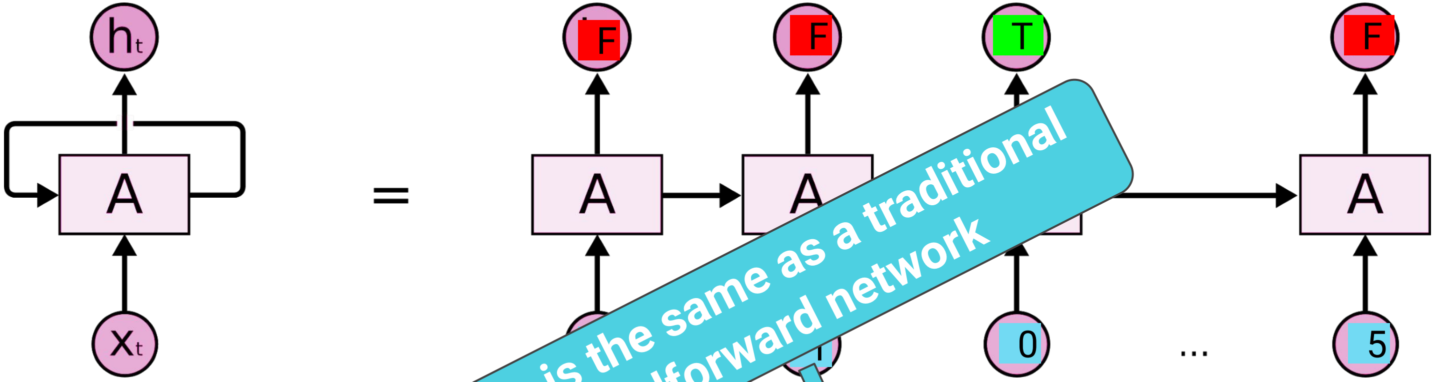


For this task: o_t only depends on x_t

$h_t = g(W_{(1)} [h_{t-1}; x_t])$
$o_t = \text{softmax}(W_{(2)} * h_t)$

A simple example

Our task is to identify the number 0 in the input



This is the same as a traditional feedforward network

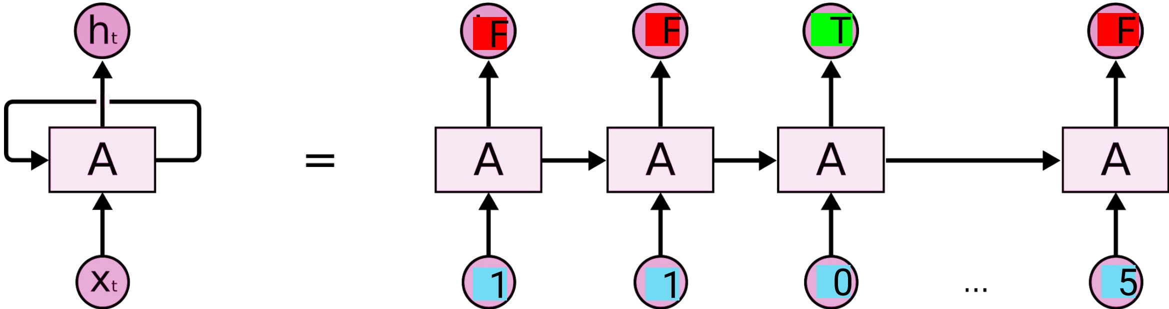
For this task: o_t only de… x_t
So let's "simplify" the a…alysis

$h_t = g(W_{(1)} [\cancel{h_{t-1}}; x_t])$
$o_t = \text{softmax}(W_{(2)} * h_t)$

A simple example

Let’s estimate the gradients

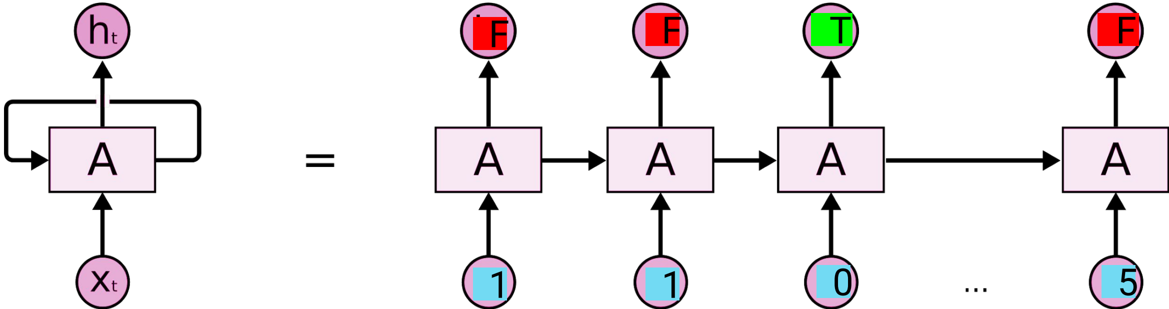
$$h_t = g(\,W_{(1)}\,[\cancel{h_{t-1}};\, x_t])$$
$$o_t = \mathrm{softmax}(W_{(2)} * h_t)$$



A simple example

Let’s estimate the gradients

$$h_t = g(W_{(1)}\, x_t)$$
$$o_t = \mathrm{softmax}(W_{(2)} * h_t)$$

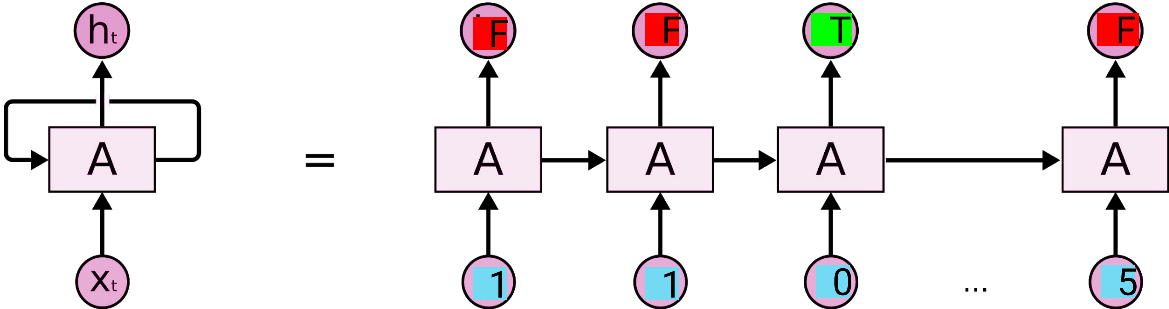


A simple example

Let's estimate the gradients

$$h_t = g(W_{(1)} x_t)$$
$$o_t = \text{softmax}(W_{(2)} * h_t)$$

$$do_t / dx_t = (do_t / dh_t) * (dh_t / dx_t)$$



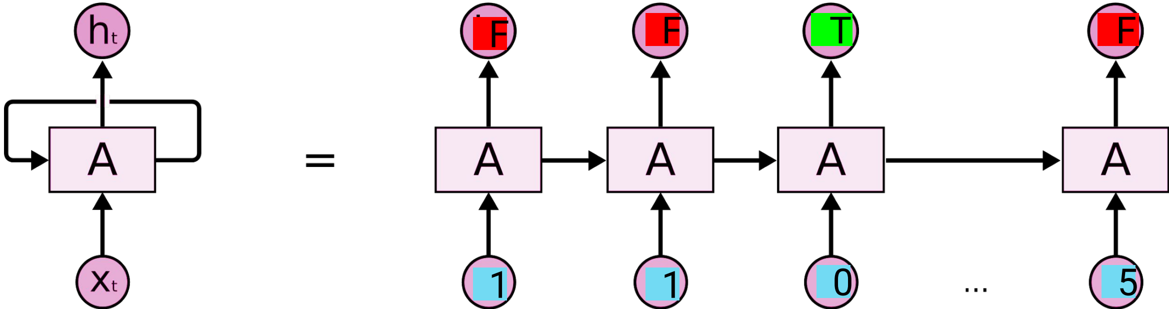
A simple example

$h_t = g(W_{(1)}\ x_t)$
$o_t = \text{softmax}(W_{(2)} * h_t)$

Let's estimate the gradients

$$do_t / dx_t = (do_t / dh_t) * (dh_t / dx_t)$$

$$do_t / dh_t = \text{softmax}'(W_{(2)} * h_t) * W_{(2)}$$



A simple example

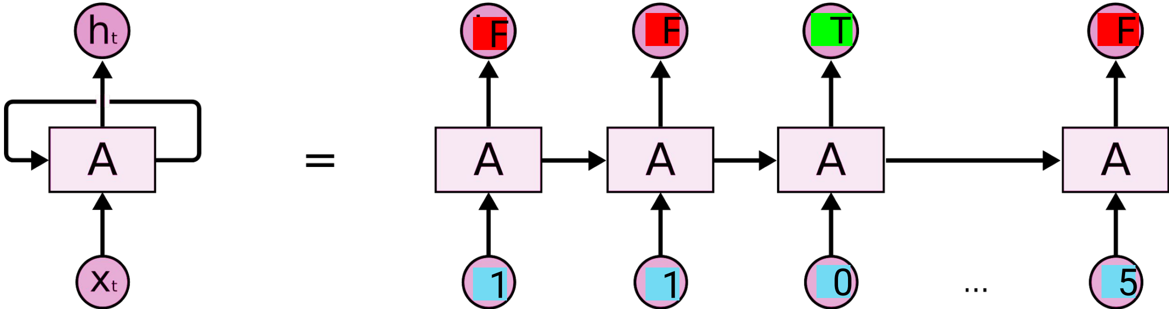
Let's estimate the gradients

$$h_t = g(W_{(1)}\, x_t)$$
$$o_t = \text{softmax}(W_{(2)} * h_t)$$

$$do_t / dx_t = (do_t / dh_t) * (dh_t / dx_t)$$

$$do_t / dh_t = \text{softmax}'(W_{(2)} * h_t) * W_{(2)}$$

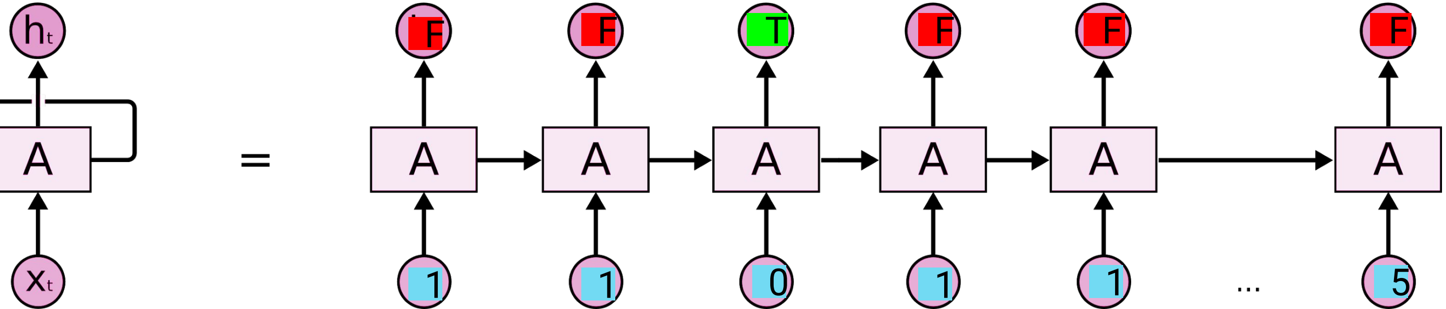
$$dh_t / dx_t = g'(W_{(1)} * x_t) * W_{(1)}$$



A less simple example

Let's change the task a little.

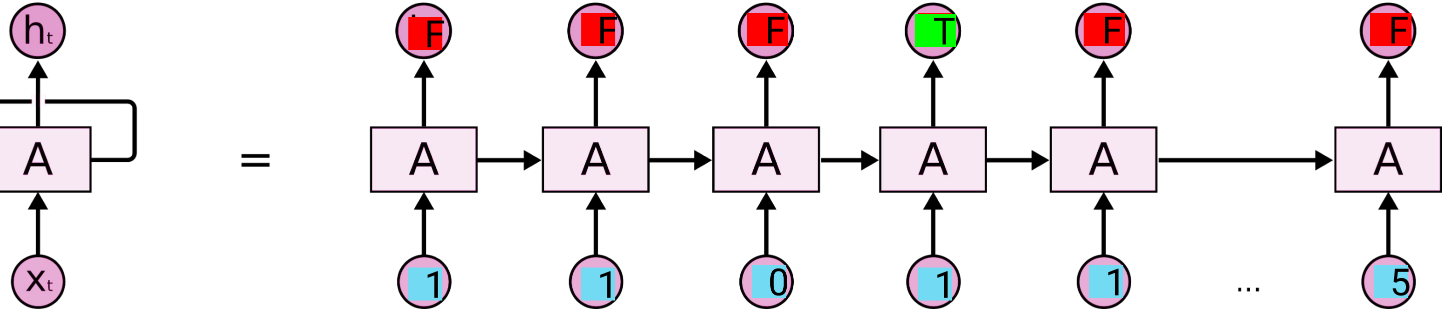
We will add some time lag in the response



A less simple example

Let's change the task a little.

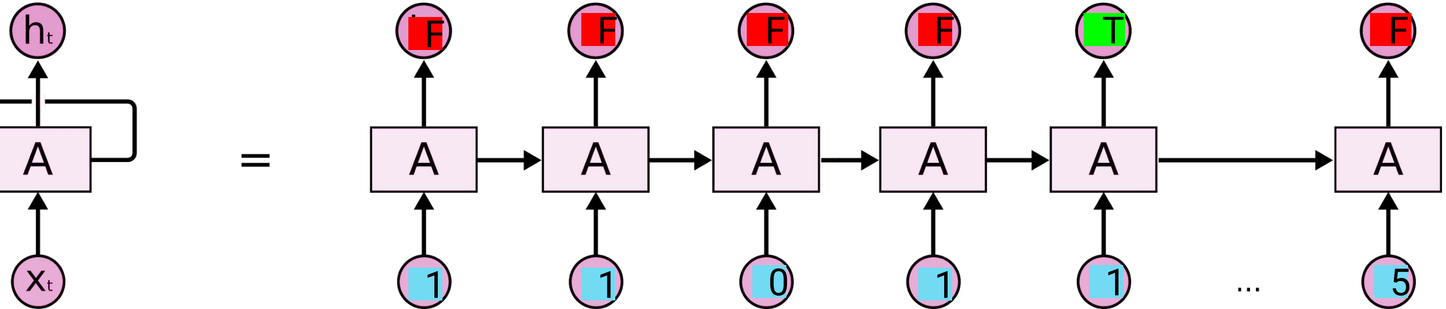
We will add some time lag in the response



A less simple example

Let's change the task a little.

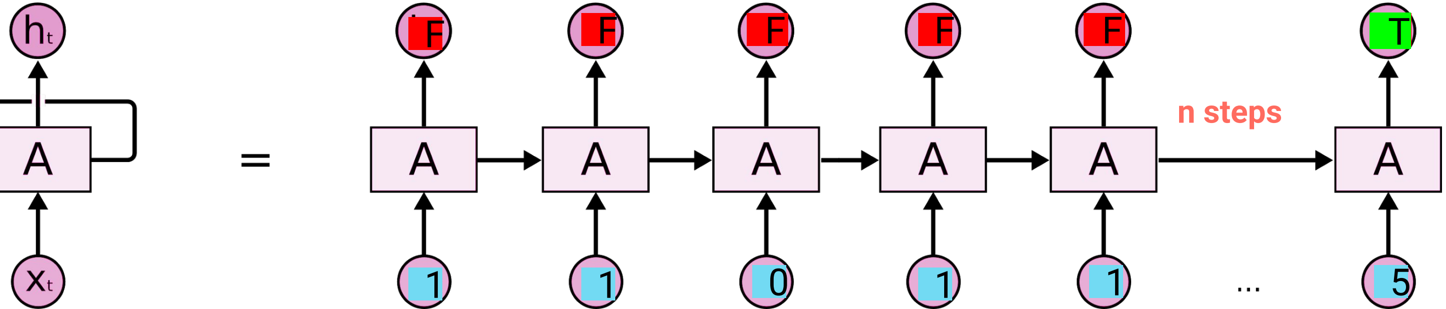
We will add some time lag in the response



A less simple example

Let’s change the task a little.

We will add some time lag in the response

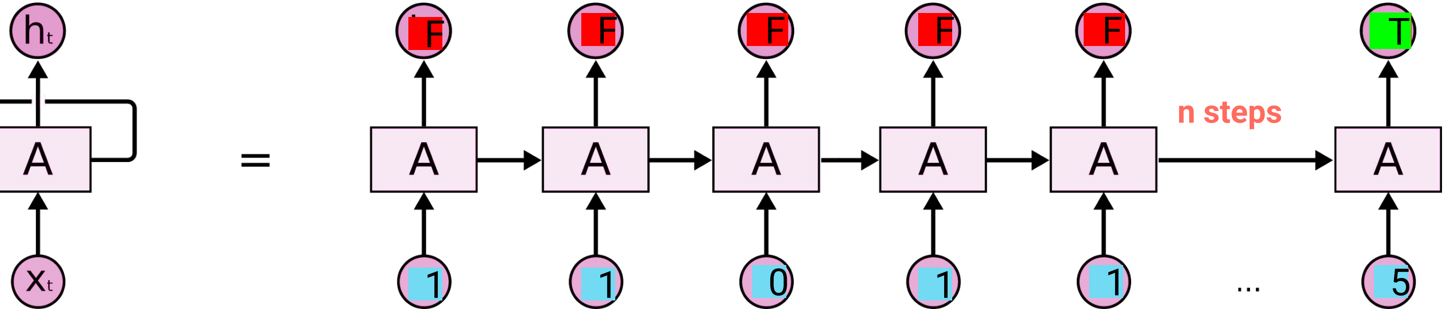


A less simple example

How will the RNN learn this?

We will ignore this term now

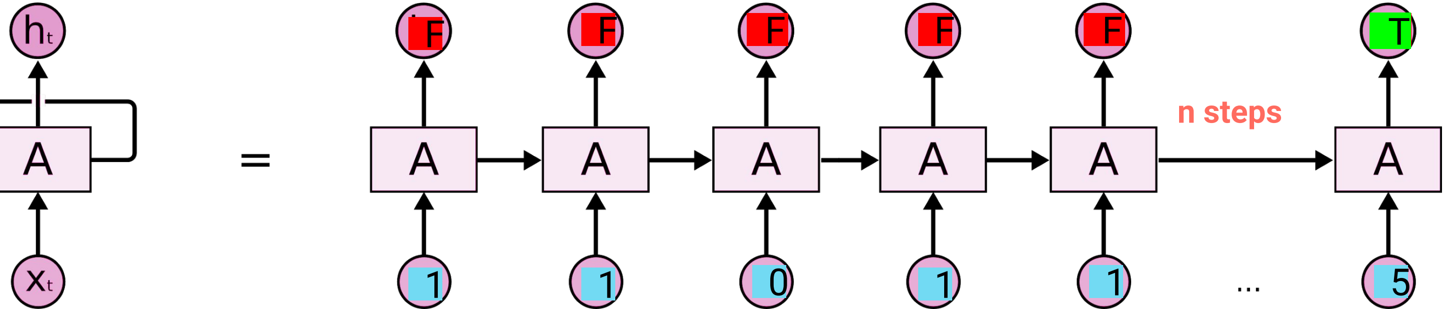
$$h_t = g(W_{(1)} [h_{t-1}])$$
$$o_t = \text{softmax}(W_{(2)} * h_t)$$



A less simple example

How will the RNN learn this?

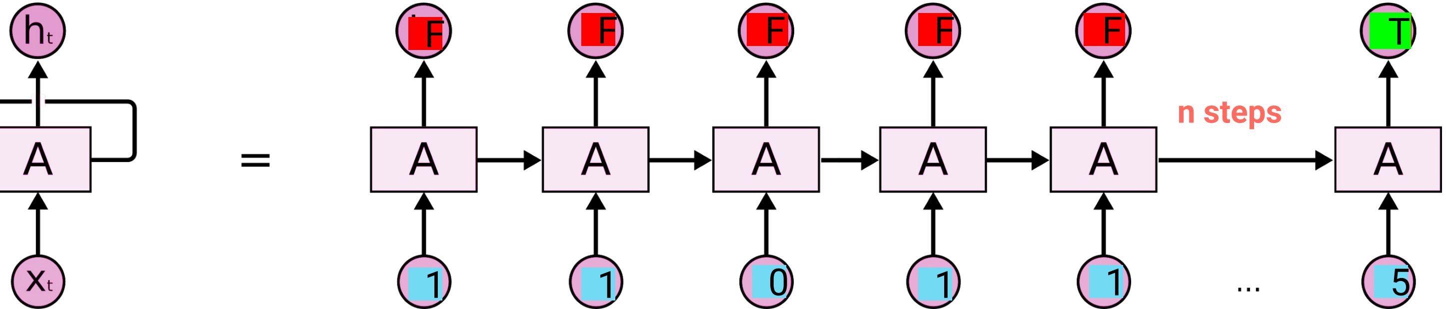
$$o_t = \mathrm{softmax}(W_{(2)} * h_t)$$
$$h_t = g(\, W_{(1)} \, [h_{t-1}; x_t])$$



A less simple example

How will the RNN learn this?

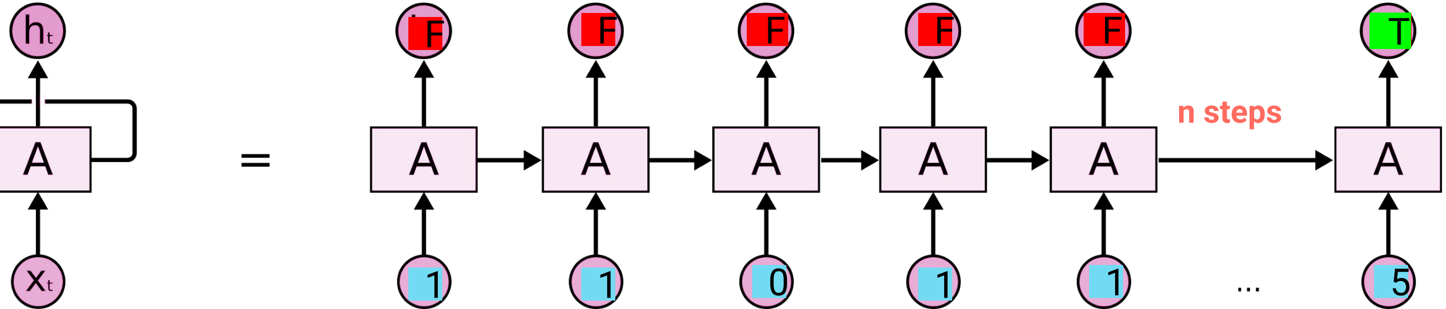
$$o_t = \mathrm{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1} \cancel{x_t}])$$



A less simple example

How will the RNN learn this?

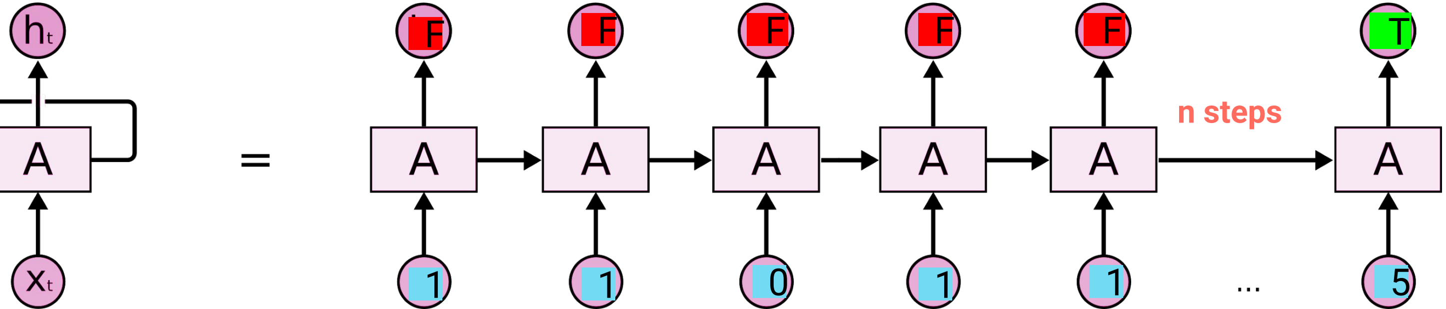
$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(\,W_{(1)}\,[h_{t-1}\,\cancel{x_t}])$$
$$h_{t-1} = g(\,W_{(1)}\,[h_{t-2}\,\cancel{x_t}])$$



A less simple example

How will the RNN learn this?

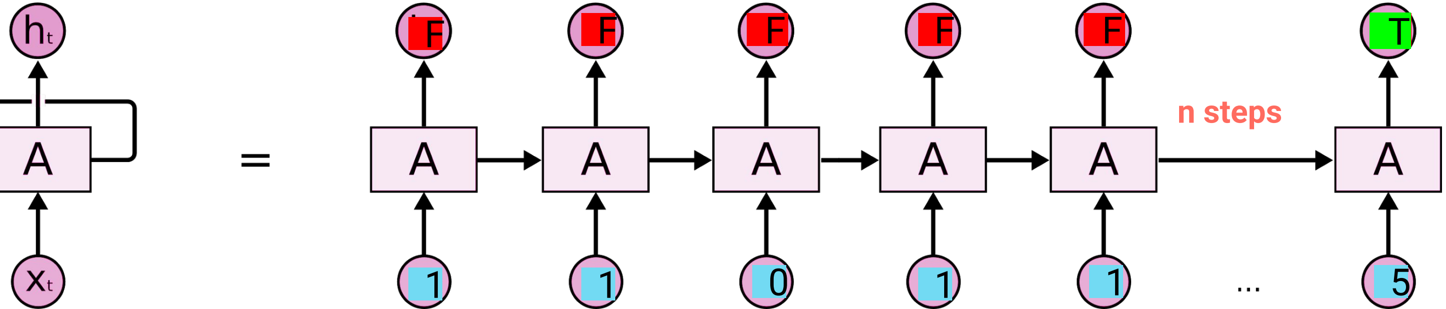
$$o_t = \mathrm{softmax}(W_{(2)} * h_t)$$
$$h_t = g(\, W_{(1)}\, [h_{t-1}, \cancel{x_t}])$$
$$h_{t-1} = g(\, W_{(1)}\, [h_{t-2}, \cancel{x_t}])$$
$$h_{t-2} = g(\, W_{(1)}\, [h_{t-3}, \cancel{x_t}])$$



A less simple example

How will the RNN learn this?

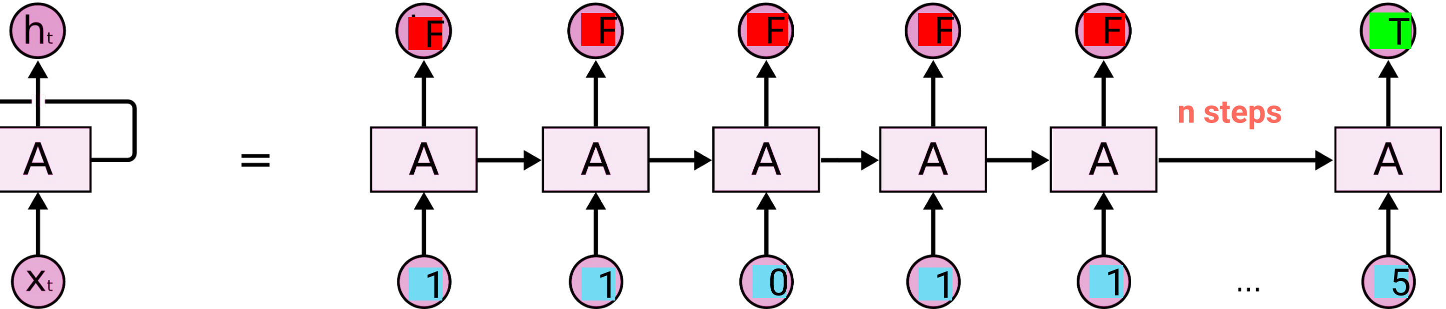
$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1} \cancel{x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2} \cancel{x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3} \cancel{x_t}])$$
$$\ldots$$



A less simple example

How will the RNN learn this?

$$
\begin{aligned}
o_t &= \text{softmax}(W_{(2)} * h_t) \\
h_t &= g(\, W_{(1)}\, [h_{t-1} \cancel{; x_t}]) \\
h_{t-1} &= g(\, W_{(1)}\, [h_{t-2} \cancel{; x_t}]) \\
h_{t-2} &= g(\, W_{(1)}\, [h_{t-3} \cancel{; x_t}]) \\
&\dots \\
h_{t-n+1} &= g(\, W_{(1)}\, [\cancel{h_{t-n}}; x_t])
\end{aligned}
$$



A less simple example

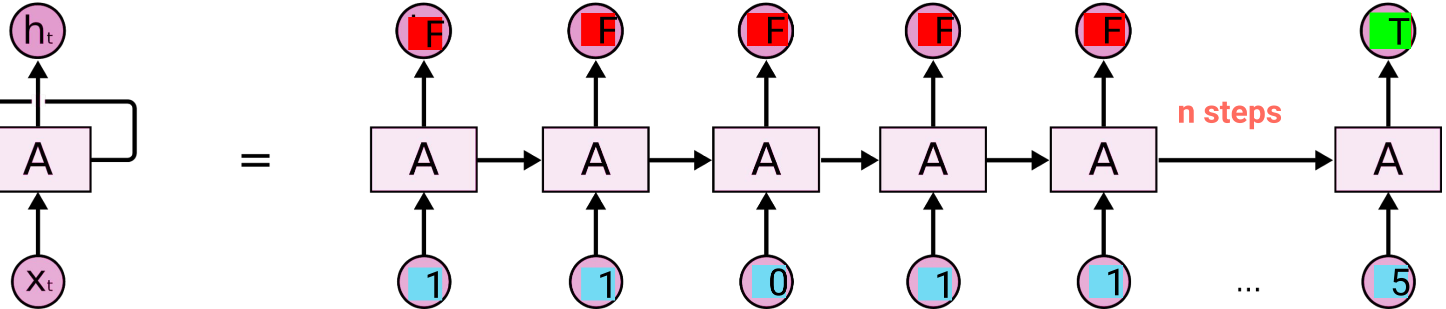
Let's estimate the gradients again!

$$do_t / dx_t = (do_t / dh_t) * (dh_t / dh_{t-1}) * (dh_{t-1} / dx_{t-n})$$

$$do_t / dh_t = softmax'(W_{(2)} * h_t) * W_{(2)}$$

$$dh_t / dh_{t-1} = g'(W_{(1)} * h_{t-1}) * W_{(1)}$$

$$o_t = softmax(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1}; \cancel{x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2}; \cancel{x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3}; \cancel{x_t}])$$
$$\dots$$
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$



A less simple example

Let's estimate the gradients again!

$do_t / dx_t = (do_t / dh_t) * (dh_t / dh_{t-1}) * (dh_{t-1} / dx_{t-n})$

$do_t / dh_t = \text{softmax}'(W_{(2)} * h_t) * W_{(2)}$

$dh_t / dh_{t-1} = g'(W_{(1)} * h_{t-1}) * W_{(1)}$

$dh_{t-1} / dx_{t-n} = ???$

$o_t = \text{softmax}(W_{(2)} * h_t)$

$h_t = g(W_{(1)} [h_{t-1}\cancel{; x_t}])$

$h_{t-1} = g(W_{(1)} [h_{t-2}\cancel{; x_t}])$

$h_{t-2} = g(W_{(1)} [h_{t-3}\cancel{; x_t}])$

...

$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$

A less simple example

Let’s estimate the gradients again!

$$do_t / dx_t = (do_t / dh_t) * (dh_t / dh_{t-1}) * (dh_{t-1} / dx_{t-n})$$

$$do_t / dh_t = \text{softmax}'(W_{(2)} * h_t) * W_{(2)}$$

$$dh_t / dh_{t-1} = g'(W_{(1)} * h_{t-1}) * W_{(1)}$$

$$dh_{t-1} / dx_{t-n} = (dh_{t-1} / dh_{t-2}) * (dh_{t-2} / dx_{t-n})$$

$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1}\cancel{; x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2}\cancel{; x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3}\cancel{; x_t}])$$
$$\ldots$$
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n};} x_t])$$

A less simple example

Let's estimate the gradients again!

$do_t / dx_t = (do_t / dh_t) * (dh_t / dh_{t-1}) * (dh_{t-1} / dx_{t-n})$

$do_t / dh_t = \text{softmax}'(W_{(2)} * h_t) * W_{(2)}$

$dh_t / dh_{t-1} = g'(W_{(1)} * h_{t-1}) * W_{(1)}$

$dh_{t-1} / dx_{t-n} = (dh_{t-1} / dh_{t-2}) * (dh_{t-2} / dx_{t-n})$

$dh_{t-2} / dx_{t-n} = (dh_{t-2} / dh_{t-3}) * (dh_{t-3} / dx_{t-n})$

$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1}; \cancel{x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2}; \cancel{x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3}; \cancel{x_t}])$$
$$\ldots$$
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less simple example

Let's estimate the gradients again!

$do_t / dx_t = (do_t / dh_t) * (dh_t / dh_{t-1}) * (dh_{t-1} / dx_{t-n})$

$do_t / dh_t = \text{softmax}'(W_{(2)} * h_t) * W_{(2)}$

$dh_t / dh_{t-1} = g'(W_{(1)} * h_{t-1}) * W_{(1)}$

$dh_{t-1} / dx_{t-n} = (dh_{t-1} / dh_{t-2}) * (dh_{t-2} / dx_{t-n})$

$dh_{t-2} / dx_{t-n} = (dh_{t-2} / dh_{t-3}) * (dh_{t-3} / dx_{t-n})$

...

$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1}; \cancel{x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2}; \cancel{x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3}; \cancel{x_t}])$$
$$\ldots$$
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less simple example

Let's estimate the gradients again!

$$do_t / dx_t = (do_t / dh_t) * (dh_t / dx_{t-n})$$

$$do_t / dh_t = \text{softmax}'(W_{(2)} * h_t) * W_{(2)}$$

$$dh_t / dx_{t-n} = \prod_{i=0}^{n} [g'(W_{(1)} * h_{t-i}) * W_{(1)}] * (dh_{t-n} / dx_{t-n})$$

$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1} \cancel{; x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2} \cancel{; x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3} \cancel{; x_t}])$$
...
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less simple example

Let’s estimate the gradients again!

$$do_t / dx_t = (do_t / dh_t) * (dh_t / dh_{t-1}) * (dh_{t-1} / dx_{t-n})$$

$$do_t / dh_t = \text{softmax}'(W_{(2)} * h_t) * W_{(2)}$$

$$dh_t / dh_{t-1} = g'(W_{(1)} * h_{t-1}) * W_{(1)}$$

$$dh_{t-1} / dx_{t-n} = (dh_{t-1} / dh_{t-2}) * (dh_{t-2} / dx_{t-n})$$

$$dh_{t-2} / dx_{t-n} = (dh_{t-2} / dh_{t-3}) * (dh_{t-3} / dx_{t-n})$$

...

$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1} \cancel{; x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2} \cancel{; x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3} \cancel{; x_t}])$$
$$\dots$$
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less simple example

Let's estimate the gradients again!

$$do_t / dx_t = (do_t / dh_t) * (dh_t / dx_{t-n})$$

$$do_t / dh_t = \text{softmax}'(W_{(2)} * h_t) * W_{(2)}$$

$$dh_t / dx_{t-n} = \prod_{i=0}^{n} [g'(W_{(1)} * h_{t-i}) * W_{(1)}] * (dh_{t-n} / dx_{t-n})$$

$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1}; \cancel{x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2}; \cancel{x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3}; \cancel{x_t}])$$
$$\dots$$
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less simple example

Let's estimate the gradients again!

$$do_t / dx_t = (do_t / dh_t) * (dh_t / dx_{t-n})$$

$$do_t / dh_t = \text{softmax}'(W_{(2)} * h_t) * W_{(2)}$$

$$dh_t / dx_{t-n} = \prod_{i=0}^{n} [g'(W_{(1)} * h_{t-i}) * W_{(1)}] * (dh_{t-n} / dx_{t-n})$$

What will happen if **n** is large?

$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1} \cancel{; x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2} \cancel{; x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3} \cancel{; x_t}])$$
...
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less sim example

dh_t / dh_{t-1}

$dh_t / dx_{t-n} = \prod_{i=0}^{n} [g'(W_{(1)} * h_{t-i}) * W_{(1)}] * (dh_{t-n} / dx_{t-n})$

What will happen if **n** is large?

$$
\begin{aligned}
o_t &= \text{softmax}(W_{(2)} * h_t) \\
h_t &= g(\, W_{(1)} \, [h_{t-1}; \cancel{x_t}]) \\
h_{t-1} &= g(\, W_{(1)} \, [h_{t-2}; \cancel{x_t}]) \\
h_{t-2} &= g(\, W_{(1)} \, [h_{t-3}; \cancel{x_t}]) \\
&\dots \\
h_{t-n+1} &= g(\, W_{(1)} \, [\cancel{h_{t-n}}; x_t])
\end{aligned}
$$

A less simple example

$dh_t / dx_{t-n} = \prod_{i=0}^{n} [g'(W_{(1)} * h_{t-i}) * W_{(1)}] * (dh_{t-n} / dx_{t-n})$

What will happen if **n** is large? It depends

$$o_t = \mathrm{softmax}(W_{(2)} * h_t)$$
$$h_t = g(\, W_{(1)} [h_{t-1} \cancel{; x_t}])$$
$$h_{t-1} = g(\, W_{(1)} [h_{t-2} \cancel{; x_t}])$$
$$h_{t-2} = g(\, W_{(1)} [h_{t-3} \cancel{; x_t}])$$
$$\ldots$$
$$h_{t-n+1} = g(\, W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less simple example

$dh_t / dx_{t-n} = \prod_{i=0}^{n} [g'(W_{(1)} * h_{t-i}) * W_{(1)}] * (dh_{t-n} / dx_{t-n})$

What will happen if **n** is large? It depends

- $|g'(W_{(1)} * h_{t-i}) * W_{(1)}| > 1$

$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1}; \cancel{x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2}; \cancel{x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3}; \cancel{x_t}])$$
$$\dots$$
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less simple example

$$dh_t / dx_{t-n} = \prod_{i=0}^{n} [g'(W_{(1)} * h_{t-i}) * W_{(1)}]* (dh_{t-n} / dx_{t-n})$$

What will happen if **n** is large? It depends

- $|g'(W_{(1)} * h_{t-i}) * W_{(1)} | > 1$
  - Gradients will exponentially grow and “explode”

$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1}\cancel{; x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2}\cancel{; x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3}\cancel{; x_t}])$$
$$\ldots$$
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less simple example

$dh_t / dx_{t-n} = \prod_{i=0}^{n} [g'(W_{(1)} * h_{t-i}) * W_{(1)}]* (dh_{t-n} / dx_{t-n})$

What will happen if **n** is large? It depends

- $|g'(W_{(1)} * h_{t-i}) * W_{(1)} | > 1$
  - Gradients will exponentially grow and "explode"
- $|g'(W_{(1)} * h_{t-i}) * W_{(1)} | < 1$

$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1}; \cancel{x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2}; \cancel{x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3}; \cancel{x_t}])$$
$$\ldots$$
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less simple example

$$dh_t / dx_{t-n} = \prod_{i=0}^{n} [g'(W_{(1)} * h_{t-i}) * W_{(1)}] * (dh_{t-n} / dx_{t-n})$$

What will happen if **n** is large? It depends

- $|g'(W_{(1)} * h_{t-i}) * W_{(1)} | > 1$
  - Gradients will exponentially grow and "explode"
- $|g'(W_{(1)} * h_{t-i}) * W_{(1)} | < 1$
  - Gradients will exponentially shrink and "vanish"

$$o_t = \text{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1}; \cancel{x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2}; \cancel{x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3}; \cancel{x_t}])$$
$$\ldots$$
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less simple example

$dh_t / dx_{t-n} = \prod_{i=0}^{n} [g'(W_{(1)} * h_{t-i}) * W_{(1)}] * (dh_{t-n} / dx_{t-n})$

What will happen if **n** is large? It depends

- $|g'(W_{(1)} * h_{t-i}) * W_{(1)} | > 1$
  - Gradients will exponentially grow and “explode”
- $|g'(W_{(1)} * h_{t-i}) * W_{(1)} | < 1$
  - Gradients will exponentially shrink and “vanish”
- $|g'(W_{(1)} * h_{t-i}) * W_{(1)} | = 1$

$$o_t = \mathrm{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1}; \cancel{x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2}; \cancel{x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3}; \cancel{x_t}])$$
$$\dots$$
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less simple example

$$dh_t / dx_{t-n} = \Pi_{i=0}^{n} [g'(W_{(1)} * h_{t-i}) * W_{(1)}] * (dh_{t-n} / dx_{t-n})$$

What will happen if **n** is large? It depends

- $|g'(W_{(1)} * h_{t-i}) * W_{(1)}| > 1$
  - Gradients will exponentially grow and "explode"
- $|g'(W_{(1)} * h_{t-i}) * W_{(1)}| < 1$
  - Gradients will exponentially shrink and "vanish"
- $|g'(W_{(1)} * h_{t-i}) * W_{(1)}| = 1$
  - Gradients can flow through "infinite" time steps

$$o_t = \mathrm{softmax}(W_{(2)} * h_t)$$
$$h_t = g(W_{(1)} [h_{t-1} \cancel{; x_t}])$$
$$h_{t-1} = g(W_{(1)} [h_{t-2} \cancel{; x_t}])$$
$$h_{t-2} = g(W_{(1)} [h_{t-3} \cancel{; x_t}])$$
$$\ldots$$
$$h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])$$

A less sim example

dh_t / dh_{t-1}

$$dh_t / dx_{t-n} = \prod_{i=0}^{n} [g'(W_{(1)} * h_{t-i}) * W_{(1)}]* (dh_{t-n} / dx_{t-n})$$

What will happen if **n** is large? It depends

- $|g'(W_{(1)} * h_{t-i}) * W_{(1)} | > 1$
  - Gradients will exponentially grow and “explode”
- $|g'(W_{(1)} * h_{t-i}) * W_{(1)} | < 1$
  - Gradients will exponentially shrink and “vanish”
- $|g'(W_{(1)} * h_{t-i}) * W_{(1)} | = 1$
  - Gradients can flow through “infinite” time steps

So why LSTMs?

$$\begin{gathered}
o_t = \mathrm{softmax}(W_{(2)} * h_t) \\
h_t = g(W_{(1)} [h_{t-1}; \cancel{x_t}]) \\
h_{t-1} = g(W_{(1)} [h_{t-2}; \cancel{x_t}]) \\
h_{t-2} = g(W_{(1)} [h_{t-3}; \cancel{x_t}]) \\
\ldots \\
h_{t-n+1} = g(W_{(1)} [\cancel{h_{t-n}}; x_t])
\end{gathered}$$

Why LSTMs?

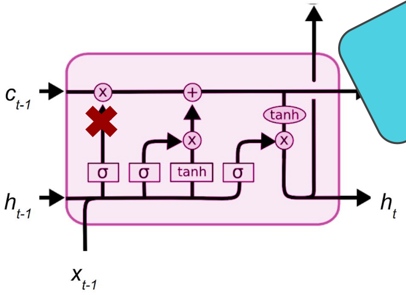
We want gradients to flow!

i.e. $dh_t \,/\, dh_{t-1} = 1$

③ Recurrent Neural Networks

Long short-term memory (LSTM)

LSTMs (Hochreiter and Schmidhuber, 1997) have the form



Direct connection

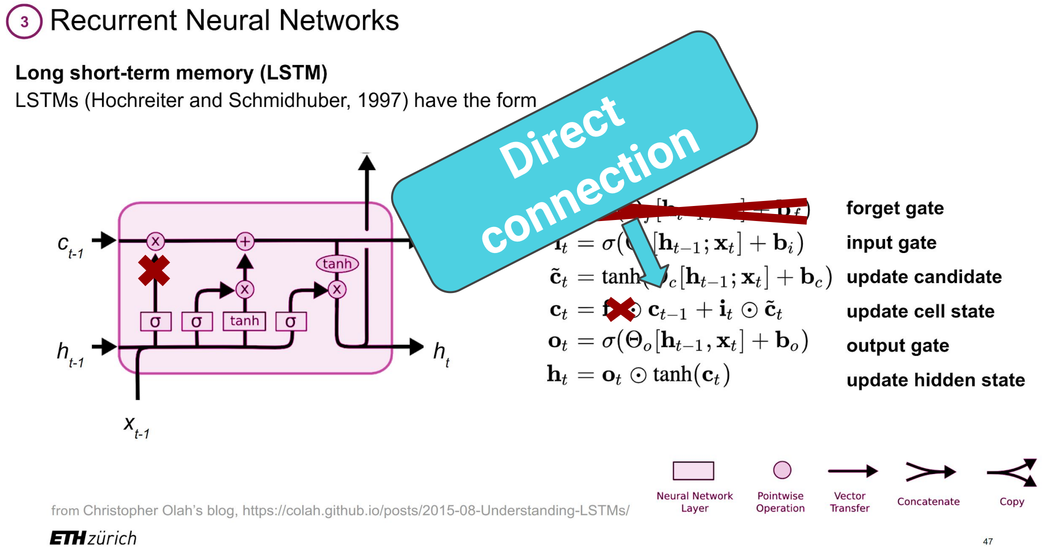
$$
\begin{aligned}
&\cancel{\mathbf{f}_t = \sigma(\Theta_f[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_f)} && \text{forget gate}\\
\mathbf{i}_t &= \sigma(\Theta_i[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_i) && \text{input gate}\\
\tilde{\mathbf{c}}_t &= \tanh(\Theta_c[\mathbf{h}_{t-1};\mathbf{x}_t] + \mathbf{b}_c) && \text{update candidate}\\
\mathbf{c}_t &= \cancel{\mathbf{f}_t} \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t && \text{update cell state}\\
\mathbf{o}_t &= \sigma(\Theta_o[\mathbf{h}_{t-1},\mathbf{x}_t] + \mathbf{b}_o) && \text{output gate}\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) && \text{update hidden state}
\end{aligned}
$$

from Christopher Olah's blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Why LSTMs?

We want gradients to flow!

i.e. $dc_t / dc_{t-1} = 1$



Why LSTMs?

We want gradients to f

i.e. $dc_t / dc_{t-1} = 1$

$c_t = c_{t-1} + i_t * func_1(c_{t-1}, x_t)$
$o_t = softmax(W_{(2)} * func_2(c_t, x_t))$

Deep Residual Learning for Image Recognition

Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun
Microsoft Research
{kahe, v-xiangz, v-shren, jiansun}@microsoft.com

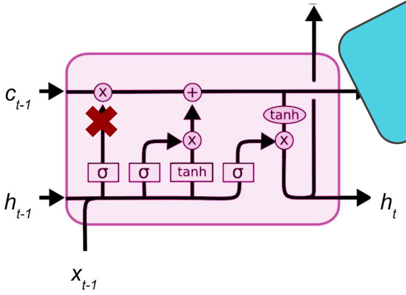
Same trick as ResNet, but ~20 years before

Does this remind you of something else?

③ Recurrent Neural Networks

Long short-term memory (LSTM)
LSTMs (Hochreiter and Schmidhuber, 1997) have the form

Direct connection



$$
\begin{aligned}
\mathbf{f}_t &= \cancel{\sigma(\Theta_f[\mathbf{h}_{t-1};\mathbf{x}_t]+\mathbf{b}_f)} && \text{forget gate}\\
\mathbf{i}_t &= \sigma(\Theta_i[\mathbf{h}_{t-1};\mathbf{x}_t]+\mathbf{b}_i) && \text{input gate}\\
\tilde{\mathbf{c}}_t &= \tanh(\Theta_c[\mathbf{h}_{t-1};\mathbf{x}_t]+\mathbf{b}_c) && \text{update candidate}\\
\mathbf{c}_t &= \cancel{\mathbf{f}_t \odot} \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t && \text{update cell state}\\
\mathbf{o}_t &= \sigma(\Theta_o[\mathbf{h}_{t-1},\mathbf{x}_t]+\mathbf{b}_o) && \text{output gate}\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) && \text{update hidden state}
\end{aligned}
$$

from Christopher Olah's blog, https://colah.github.io/posts/2015-08-Understanding-LSTMs/