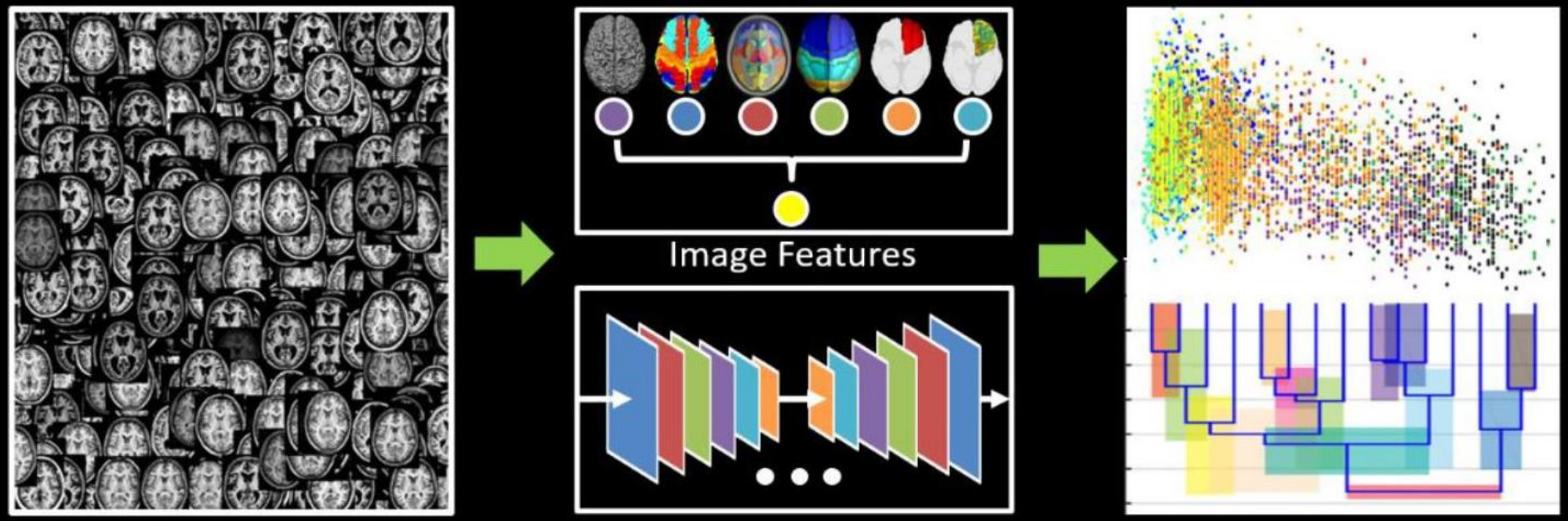
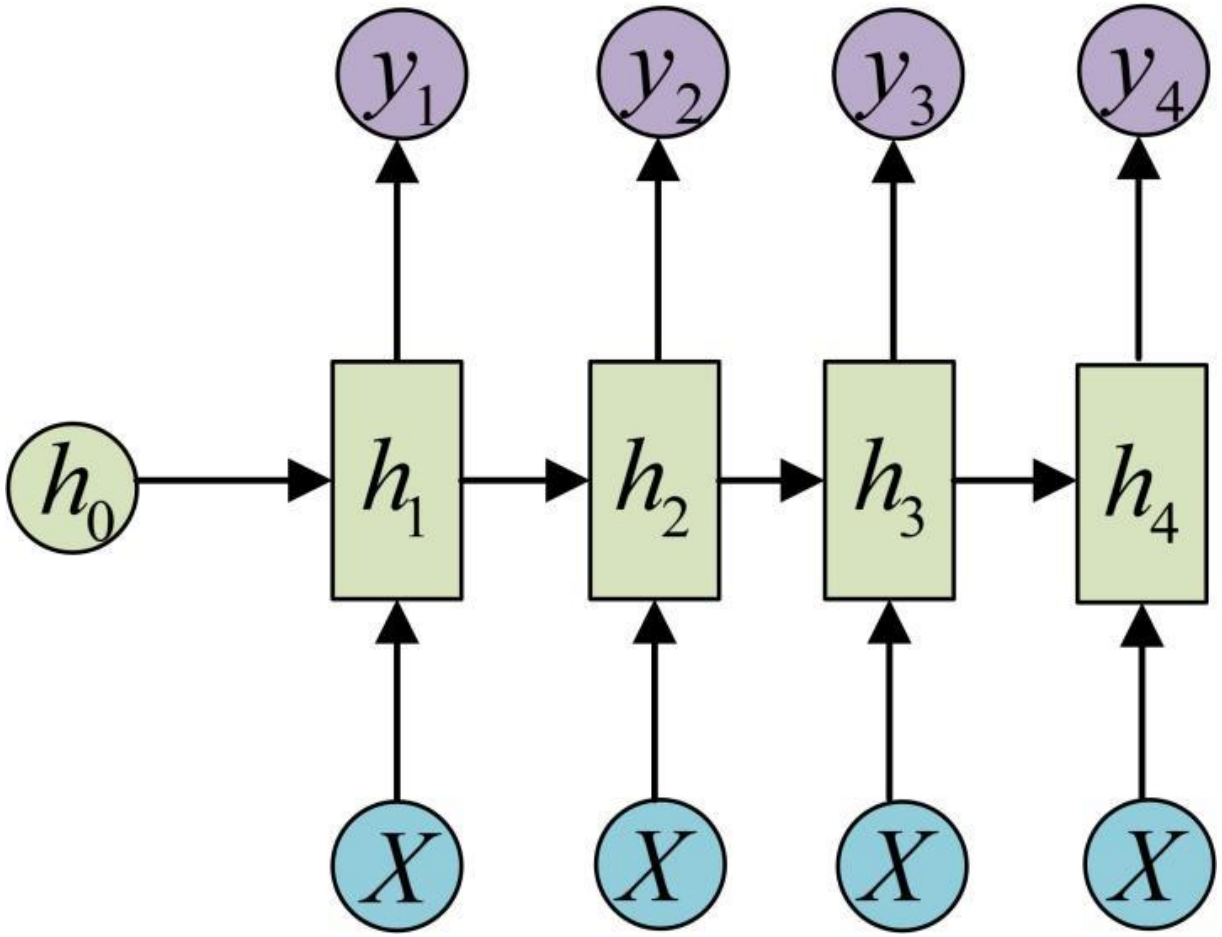


Spatial-temporal Model: RNN and LSTM

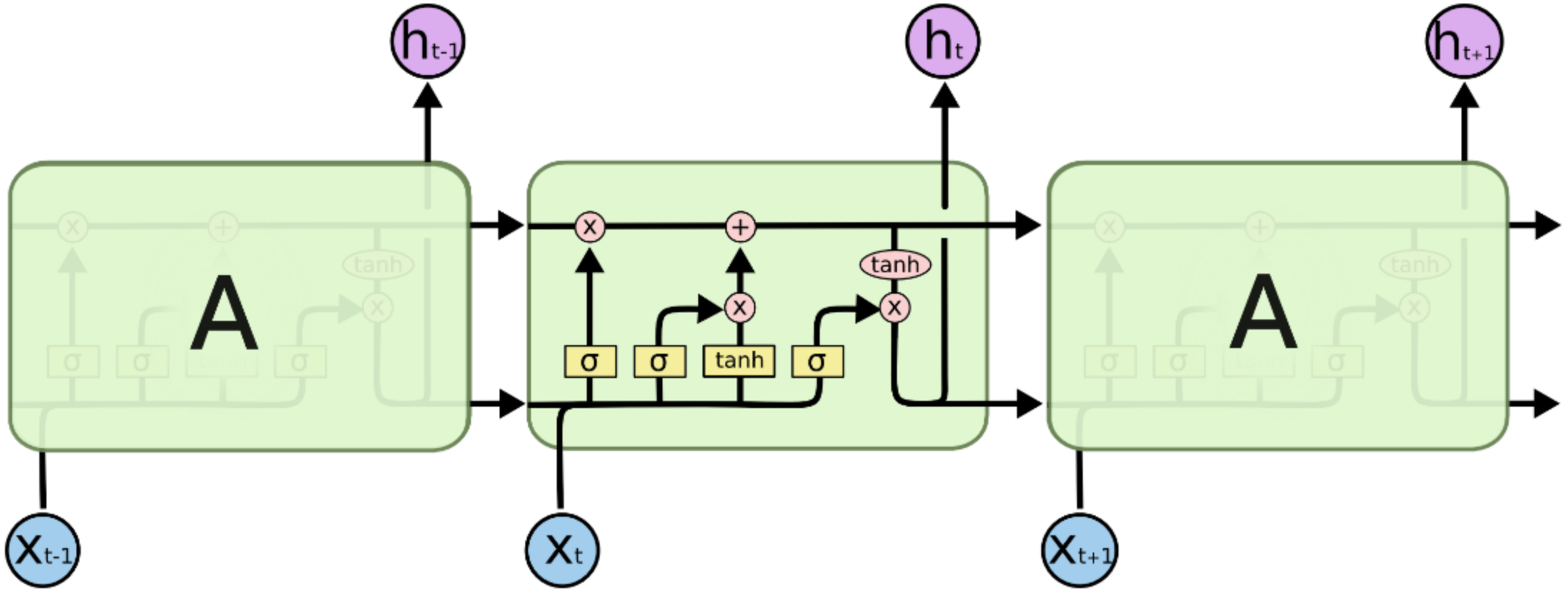
[Spring 2020 CS-8395 Deep Learning in Medical Image Computing]

Instructor: Yuankai Huo, Ph.D.
Department of Electrical Engineering and Computer Science
Vanderbilt University

N vs N



https://zhuanlan.zhihu.com/p/28054589

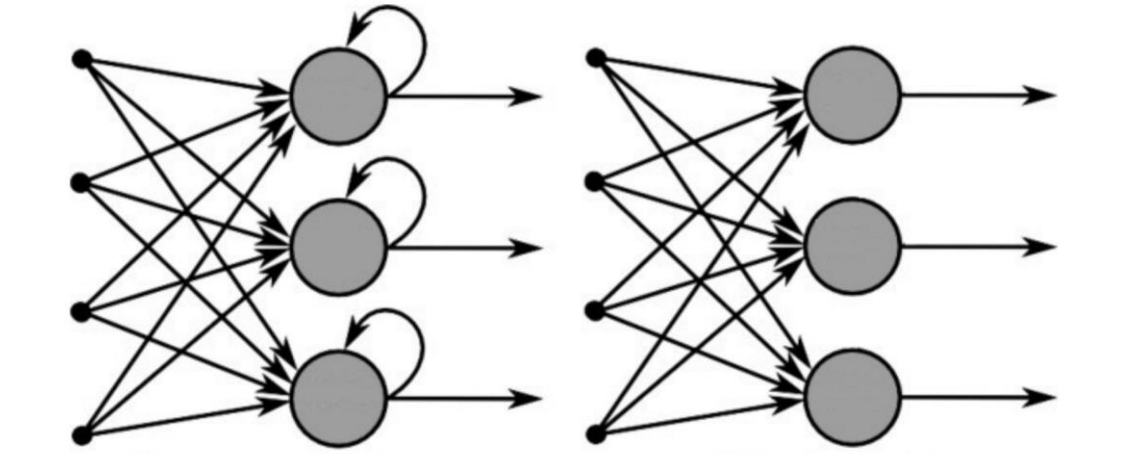


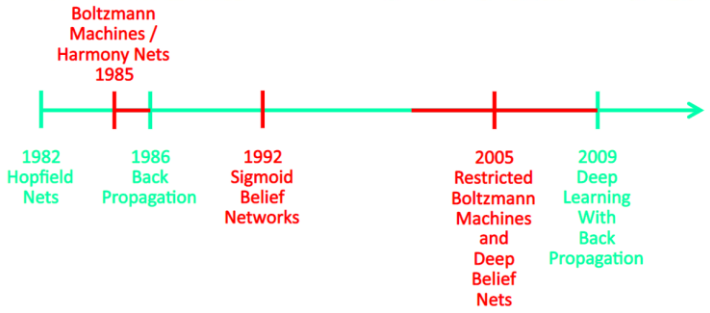
The repeating module in an LSTM contains four interacting layers.

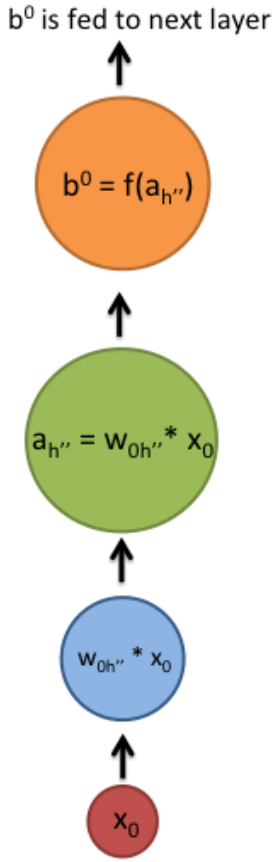
Topics

- RNN
- LSTM
- ConvLSTM

Feed-forward and Recurrent Network

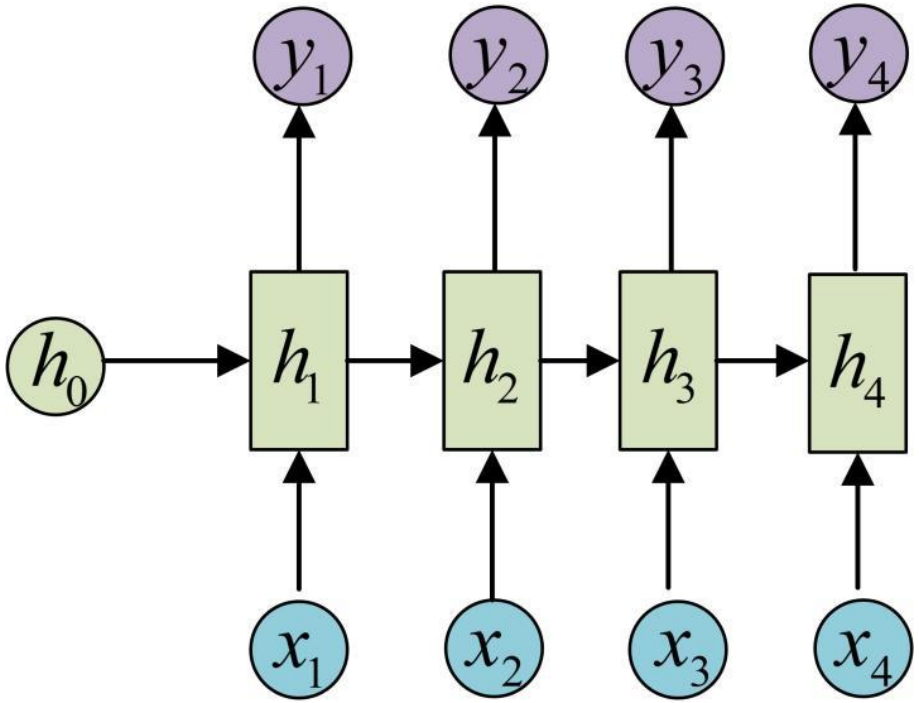






https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5 https://www.leiphone.com/news/201608/syAwLNx4bGPuFYI1.html https://slideplayer.com/slide/3383596/

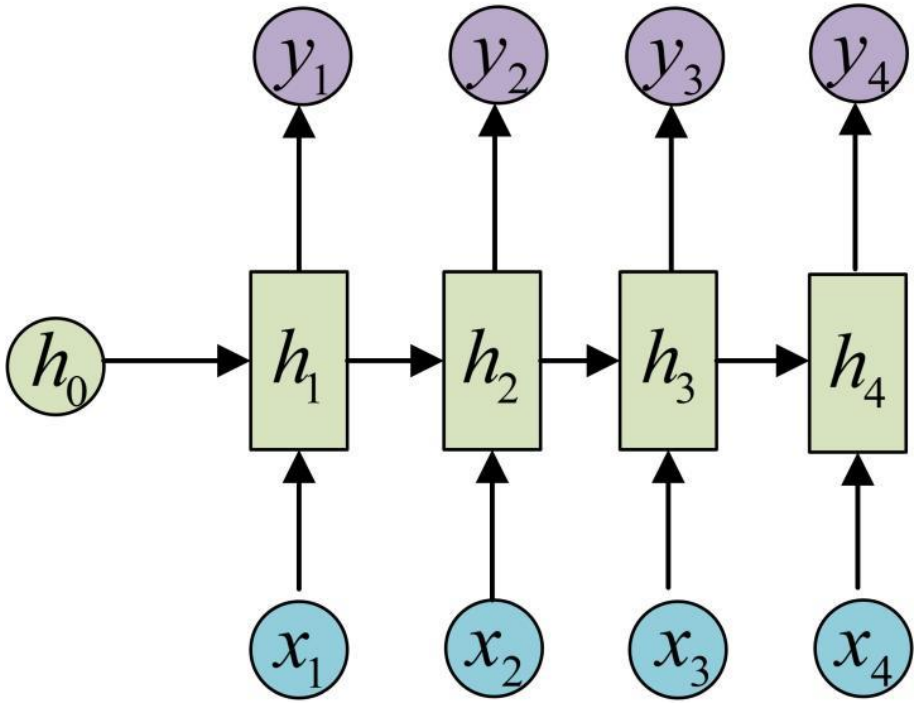
RNN



RNN

$$h_t = \tanh(w_{ih}x_t + b_{ih} + w_{hh}h_{(t-1)} + b_{hh})$$

RNN and LSTM



RNN

$$h_t = \tanh(w_{ih}x_t + b_{ih} + w_{hh}h_{(t-1)} + b_{hh})$$

LSTM

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$$

$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf})$$

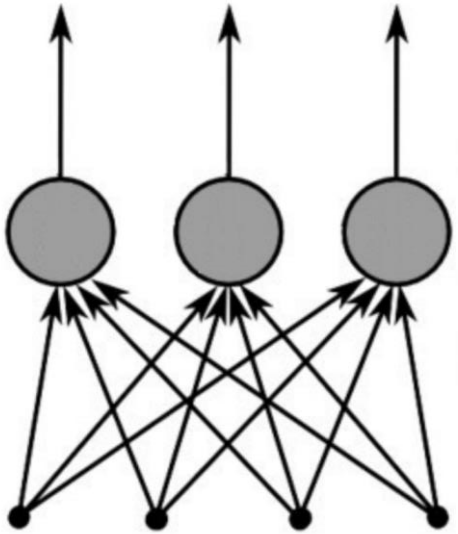
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg})$$

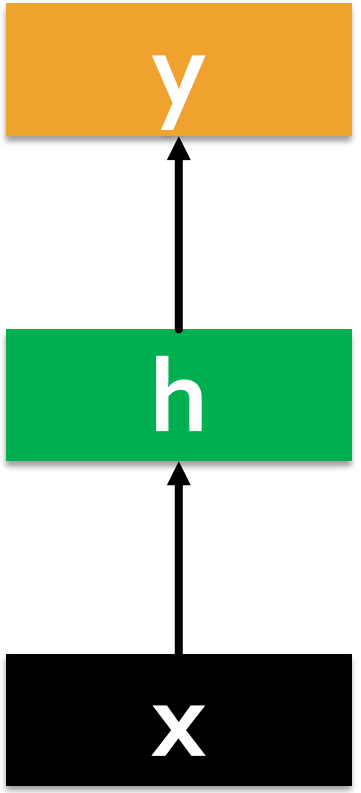
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho})$$

$$c_t = f_t c_{(t-1)} + i_t g_t$$

$$h_t = o_t \tanh(c_t)$$

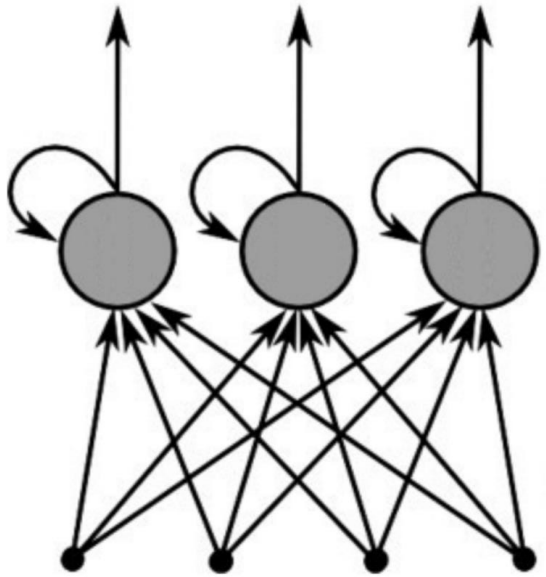
Deep Network

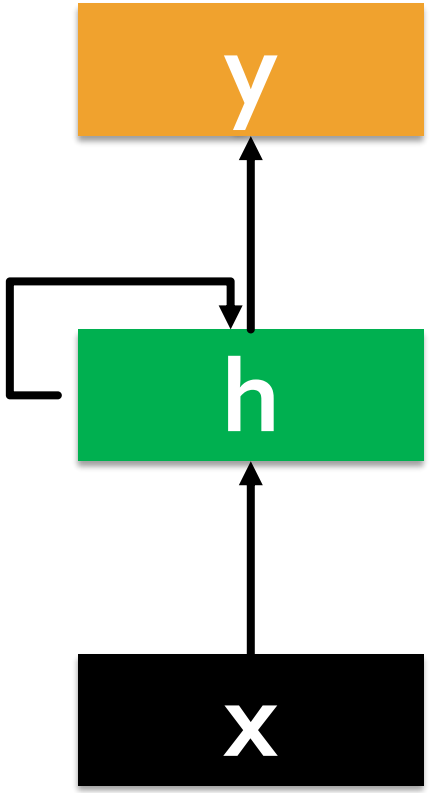




https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

RNN



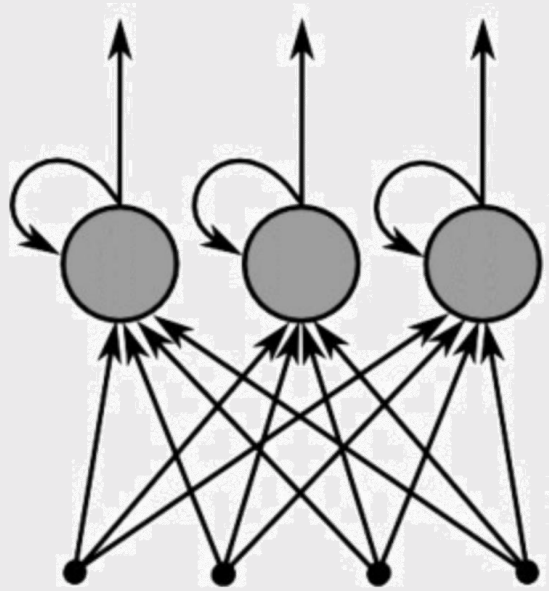


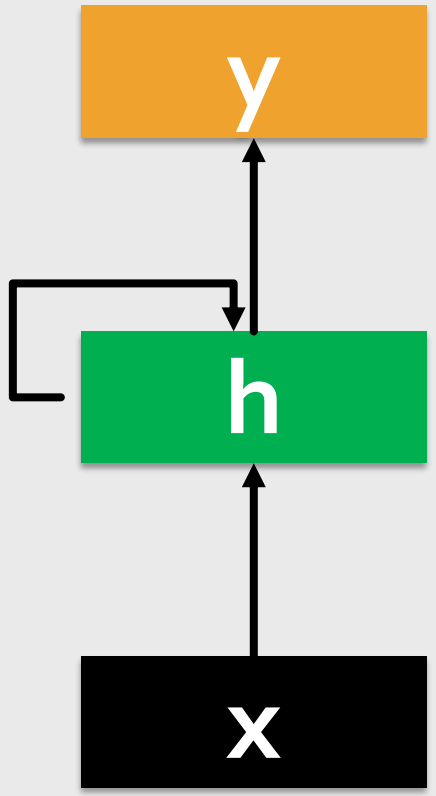
https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

RNN

$$\mathbf{h}_t = \boldsymbol{\theta}\boldsymbol{\phi}(\mathbf{h}_{t-1}) + \boldsymbol{\theta}_x \mathbf{x}_t$$

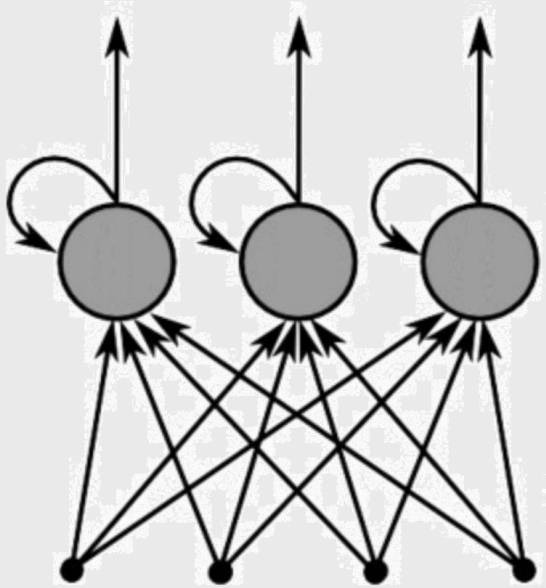
$$\mathbf{y}_t = \boldsymbol{\theta}_y \boldsymbol{\phi}(\mathbf{h}_t)$$

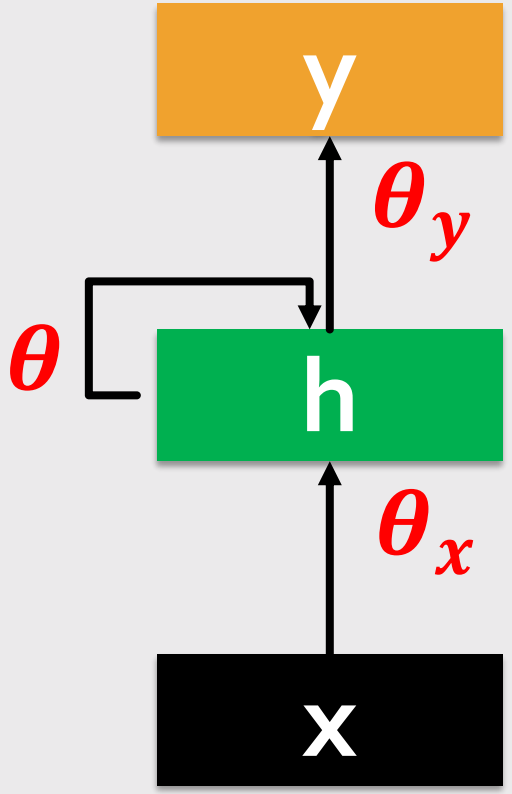




https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

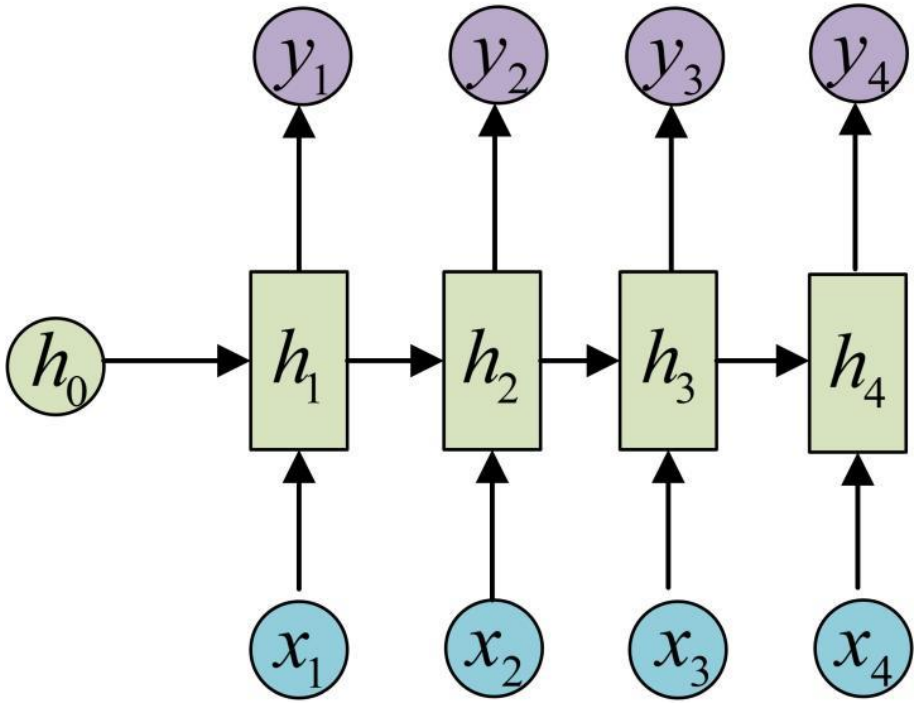
CONV and RNN





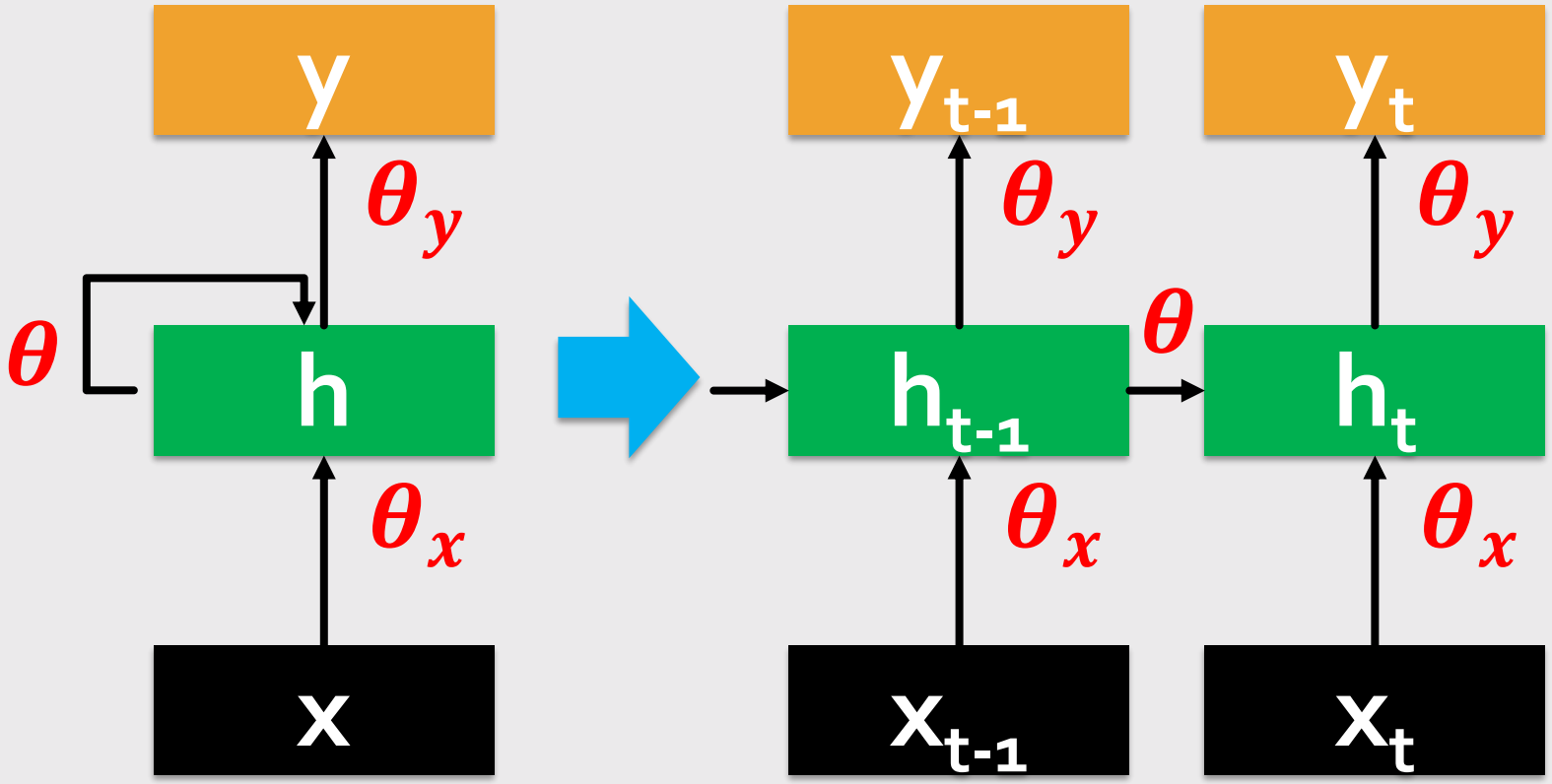
https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

What we see



Same thing

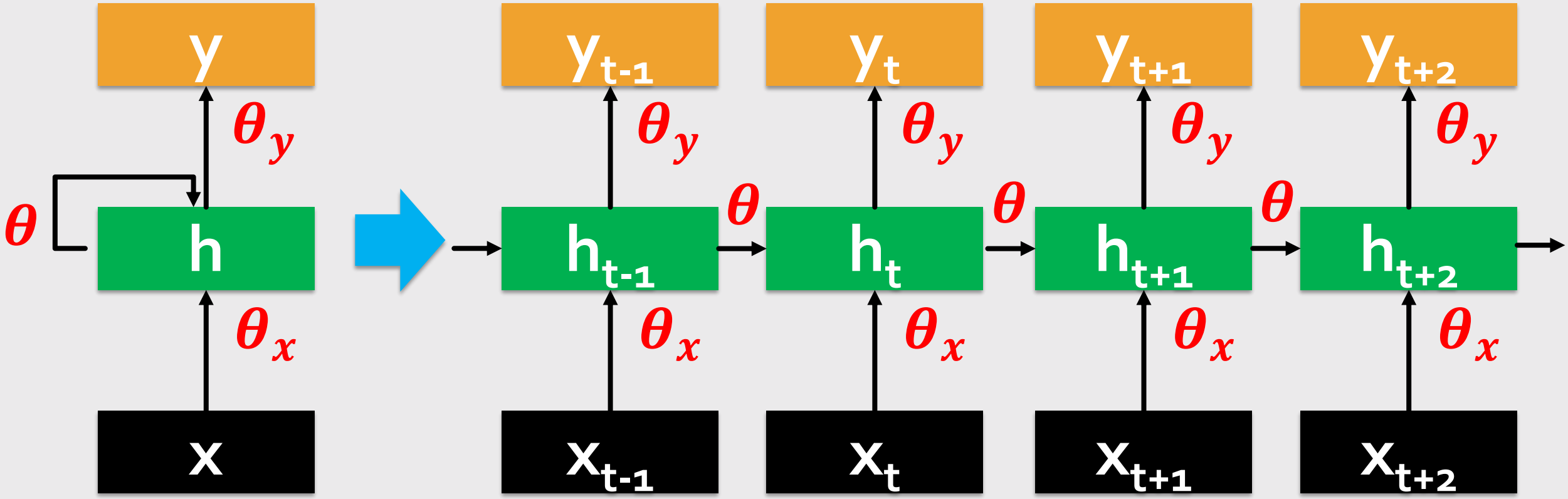
$$\mathbf{h}_t = \theta\phi(\mathbf{h}_{t-1}) + \theta_x \mathbf{x}_t$$
$$\mathbf{y}_t = \theta_y \phi(\mathbf{h}_t)$$



https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

You can extend

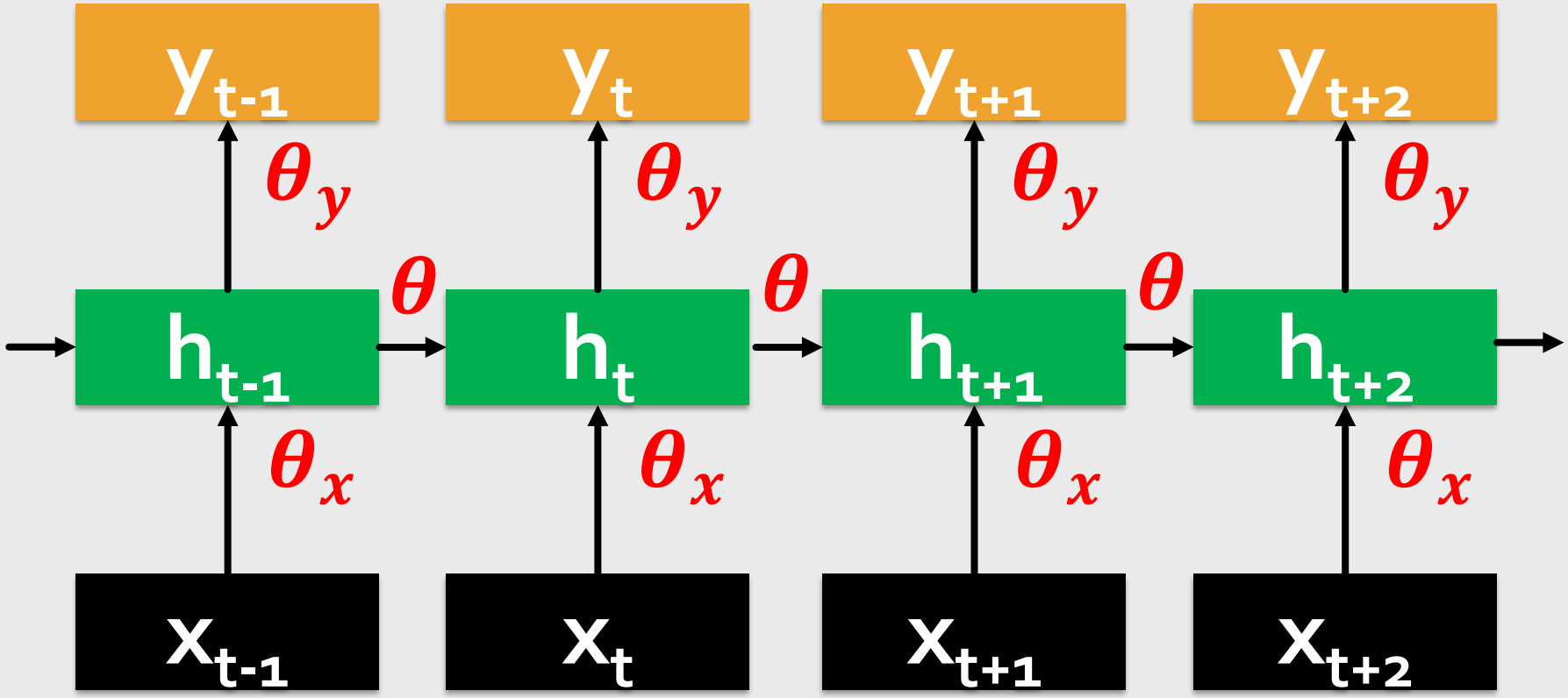
$$\mathbf{h}_t = \theta\phi(\mathbf{h}_{t-1}) + \theta_x \mathbf{x}_t$$
$$\mathbf{y}_t = \theta_y \phi(\mathbf{h}_t)$$



https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

Calculate Loss

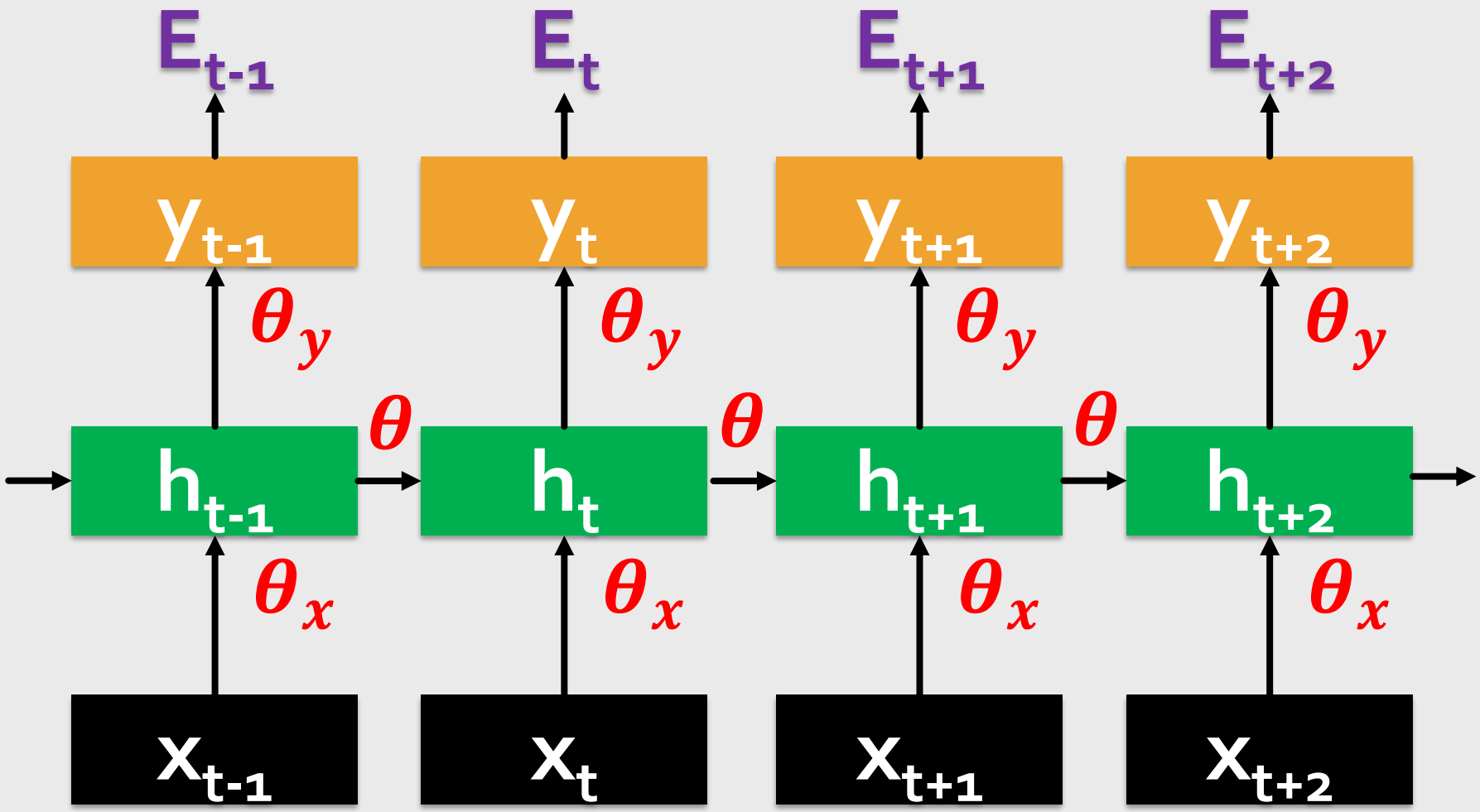
$$\mathbf{h}_t = \theta\phi(\mathbf{h}_{t-1}) + \theta_x \mathbf{x}_t$$
$$\mathbf{y}_t = \theta_y \phi(\mathbf{h}_t)$$



https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

Calculate Loss

$$\mathbf{h}_t = \theta\phi(\mathbf{h}_{t-1}) + \theta_x \mathbf{x}_t$$
$$\mathbf{y}_t = \theta_y \phi(\mathbf{h}_t)$$

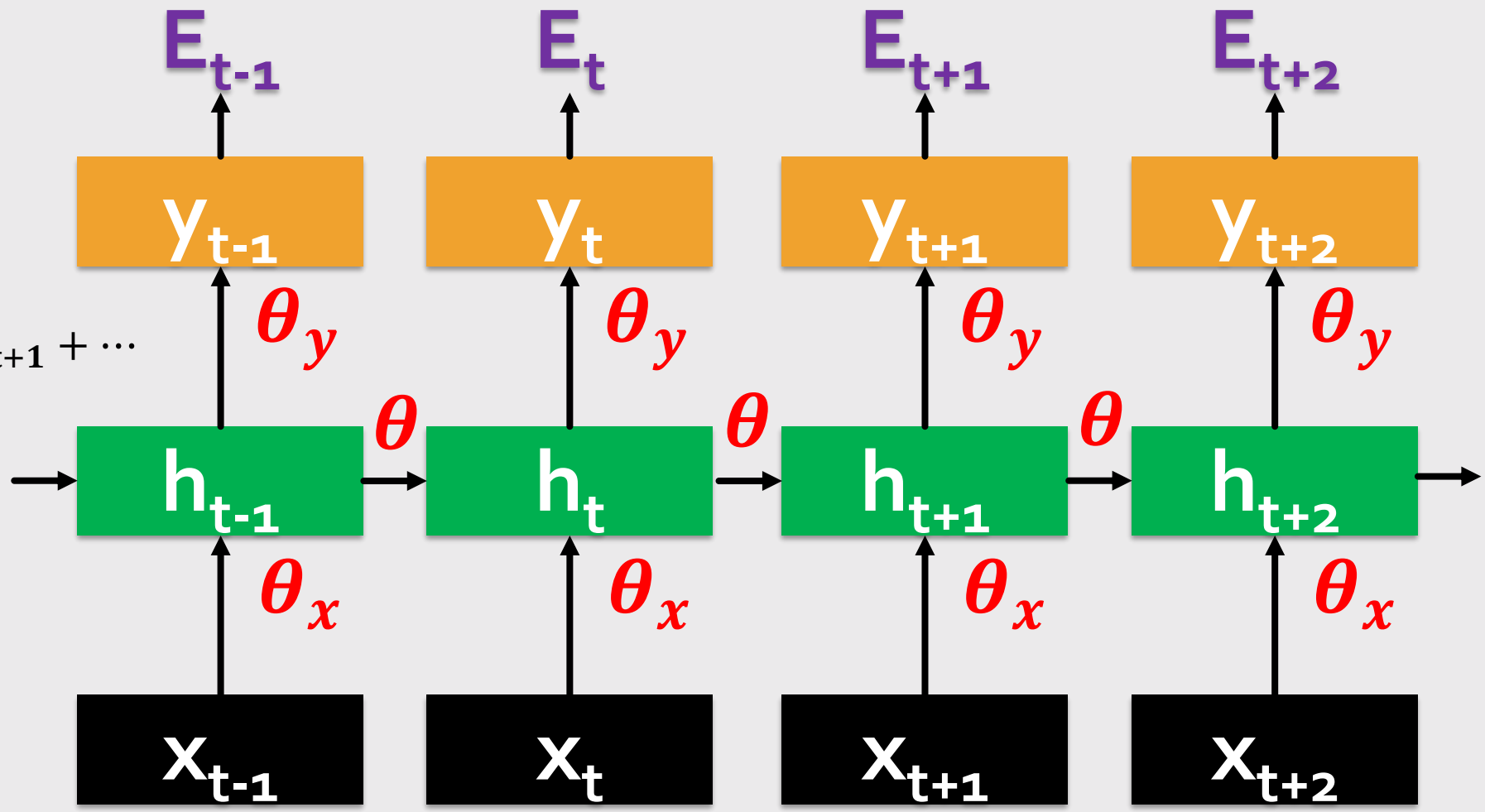


https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

Calculate Loss

$$\mathbf{h}_t = \theta\phi(\mathbf{h}_{t-1}) + \theta_x \mathbf{x}_t$$
$$\mathbf{y}_t = \theta_y \phi(\mathbf{h}_t)$$

$$\mathbf{E} = \ldots + \mathbf{E_{t-1}} + \mathbf{E_t} + \mathbf{E_{t+1}} + \mathbf{E_{t+1}} + \cdots$$



https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

Calculate Loss

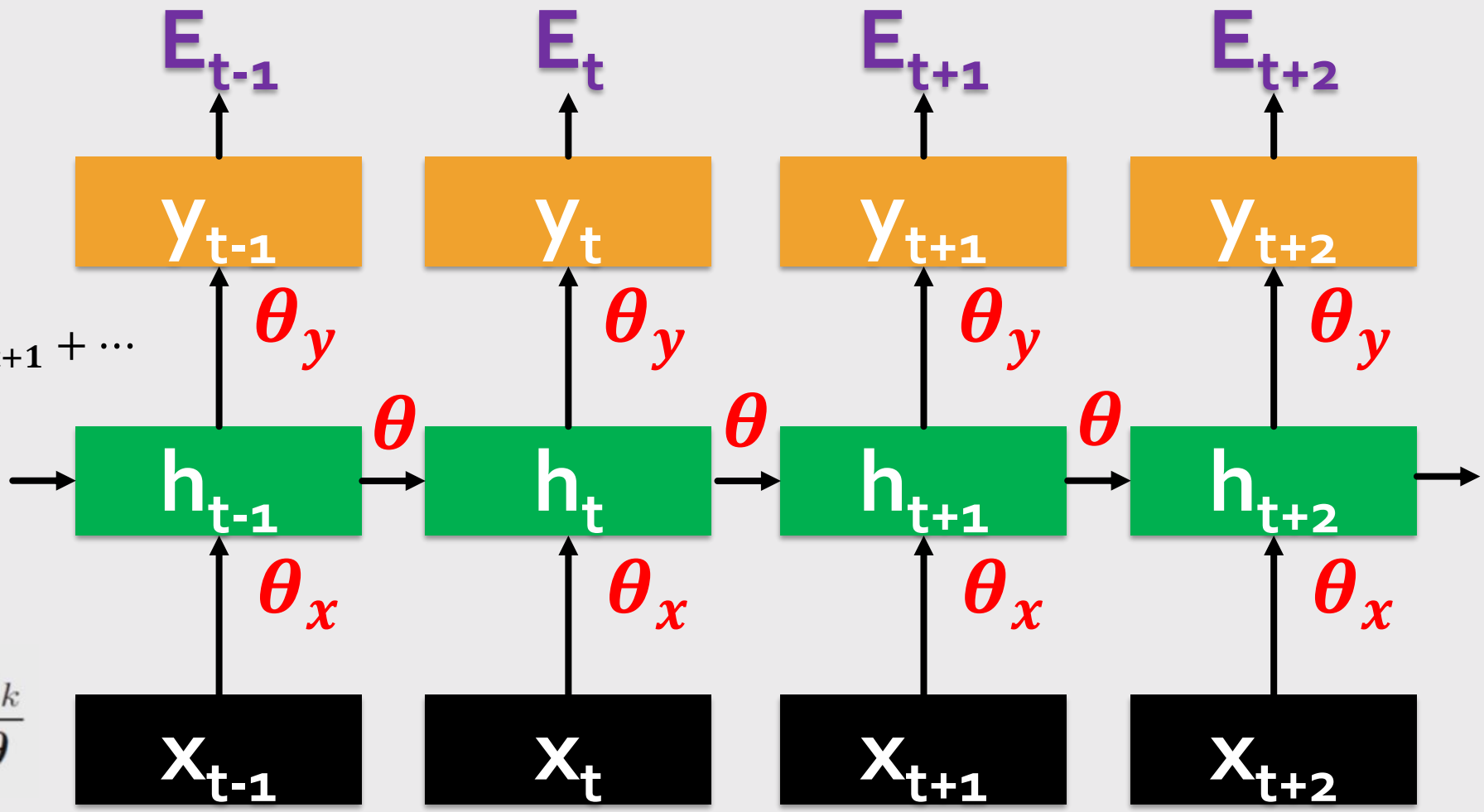
$$\mathbf{h}_t = \theta\phi(\mathbf{h}_{t-1}) + \theta_x \mathbf{x}_t$$

$$\mathbf{y}_t = \theta_y \phi(\mathbf{h}_t)$$

$$\mathbf{E} = \ldots + \mathbf{E_{t-1}} + \mathbf{E_t} + \mathbf{E_{t+1}} + \mathbf{E_{t+1}} + \cdots$$

$$\frac{\partial E}{\partial \theta} = \sum_{t=1}^{S} \frac{\partial E_t}{\partial \theta}$$

$$\frac{\partial E_t}{\partial \theta} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \theta}$$



https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

Calculate Loss

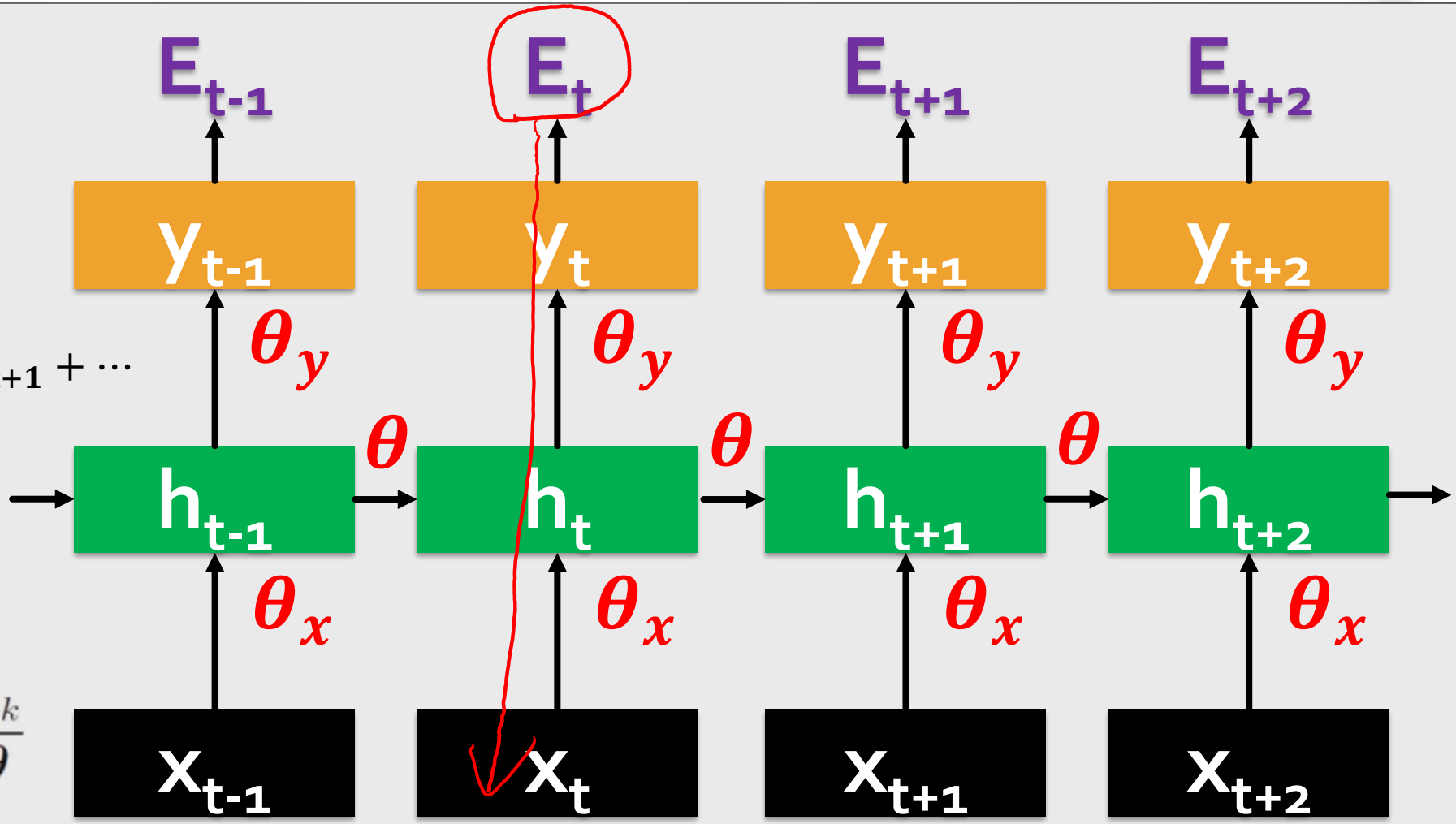
$$\mathbf{h}_t = \theta\phi(\mathbf{h}_{t-1}) + \theta_x \mathbf{x}_t$$

$$\mathbf{y}_t = \theta_y \phi(\mathbf{h}_t)$$

$$\mathbf{E} = \ldots + \mathbf{E_{t-1}} + \mathbf{E_t} + \mathbf{E_{t+1}} + \mathbf{E_{t+1}} + \cdots$$

$$\frac{\partial E}{\partial \theta} = \sum_{t=1}^{S} \frac{\partial E_t}{\partial \theta}$$

$$\frac{\partial E_t}{\partial \theta} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \theta}$$



https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

Calculate Loss

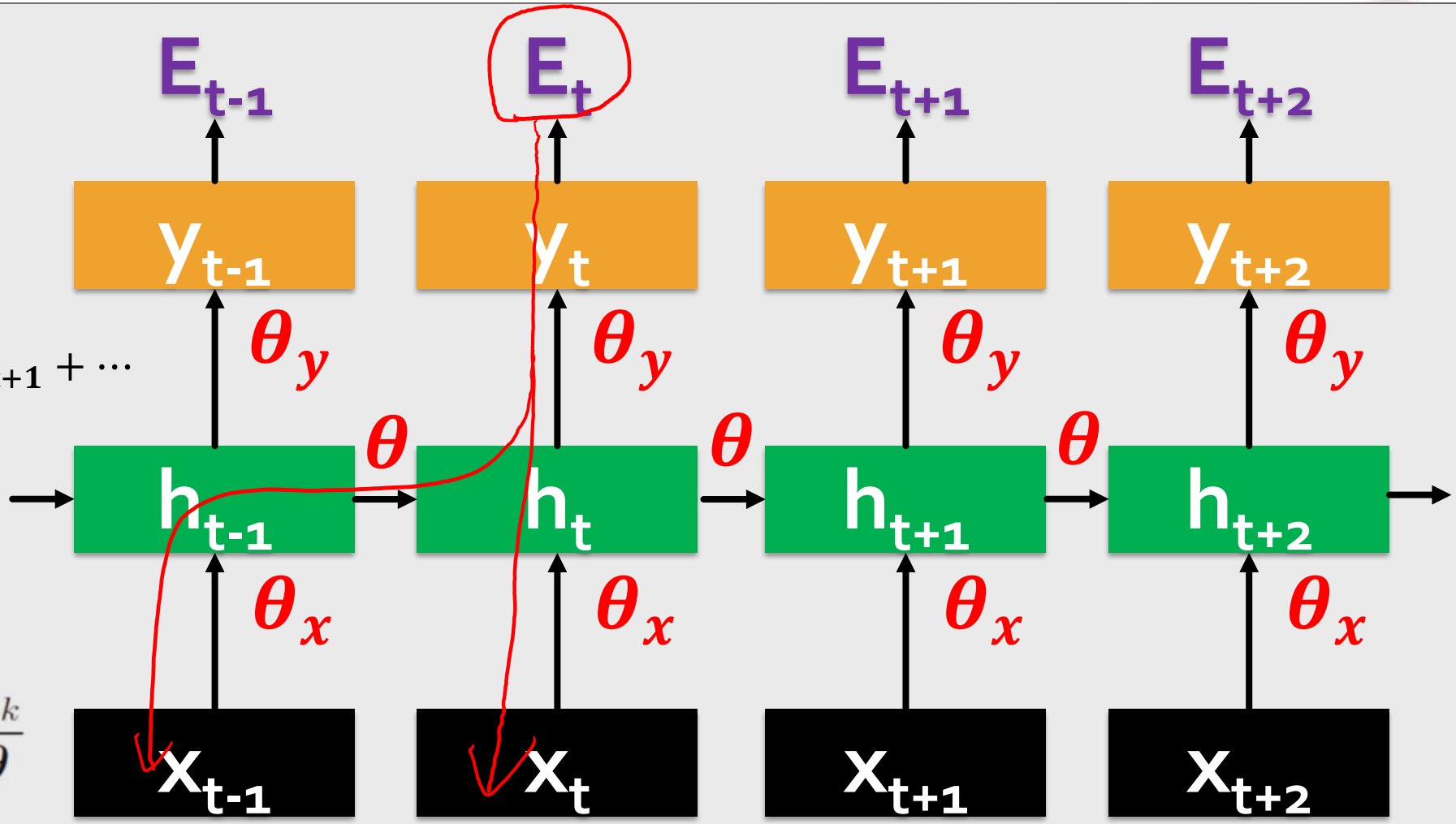
$$\mathbf{h}_t = \theta\phi(\mathbf{h}_{t-1}) + \theta_x \mathbf{x}_t$$

$$\mathbf{y}_t = \theta_y \phi(\mathbf{h}_t)$$

$$\mathbf{E} = \ldots + \mathbf{E_{t-1}} + \mathbf{E_t} + \mathbf{E_{t+1}} + \mathbf{E_{t+1}} + \cdots$$

$$\frac{\partial E}{\partial \theta} = \sum_{t=1}^{S} \frac{\partial E_t}{\partial \theta}$$

$$\frac{\partial E_t}{\partial \theta} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \theta}$$



https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

Calculate Loss

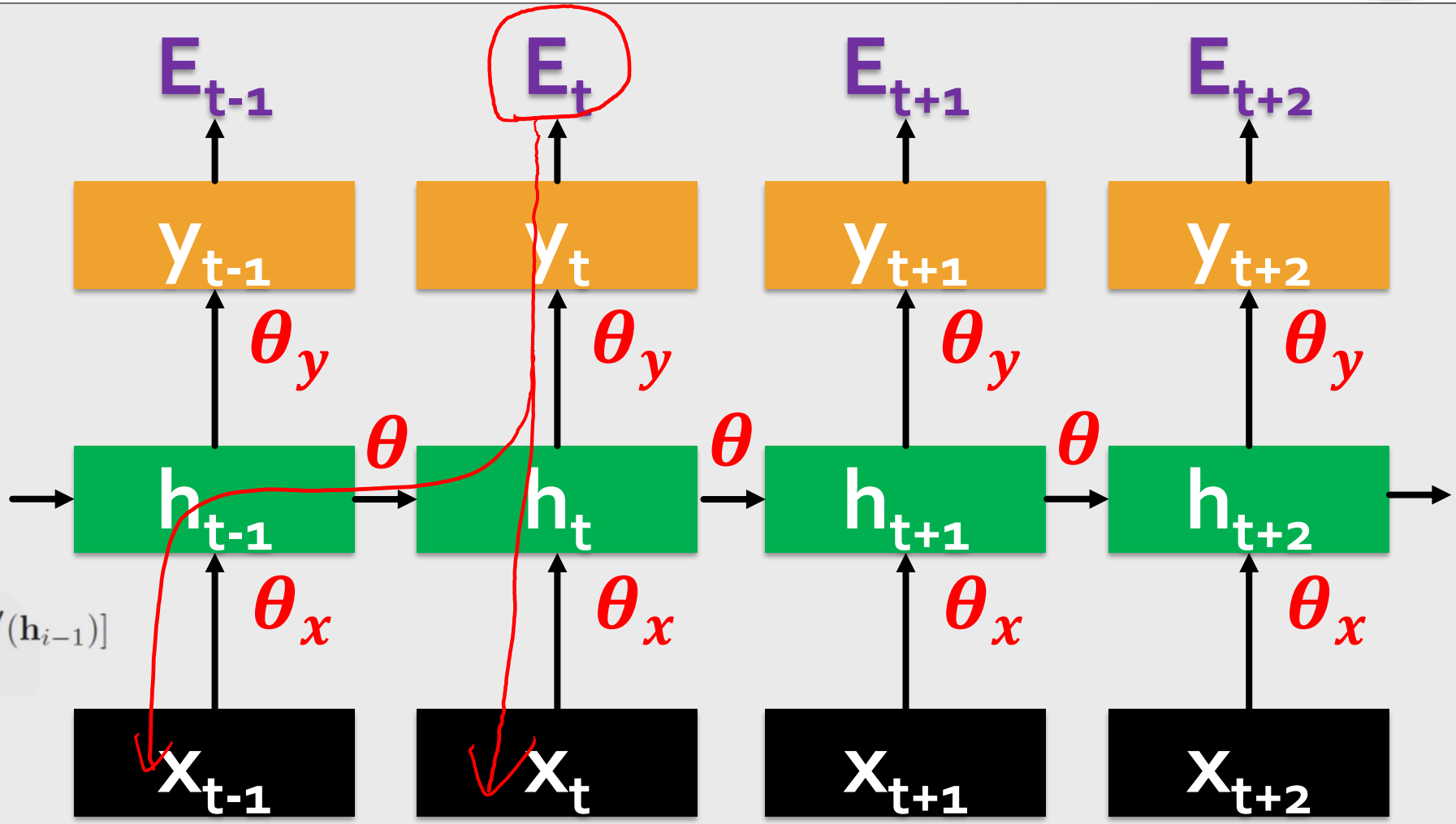
$$\mathbf{h}_t = \boldsymbol{\theta}\phi(\mathbf{h}_{t-1}) + \boldsymbol{\theta}_x \mathbf{x}_t$$

$$\mathbf{y}_t = \boldsymbol{\theta}_y \phi(\mathbf{h}_t)$$

$$\frac{\partial E}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{S} \frac{\partial E_t}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial E_t}{\partial \boldsymbol{\theta}} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \prod_{i=k+1}^{t} \boldsymbol{\theta}^T \operatorname{diag}[\phi'(\mathbf{h}_{i-1})]$$



https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

$$\mathbf{h}_t = \boldsymbol{\theta}\boldsymbol{\phi}(\mathbf{h}_{t-1}) + \boldsymbol{\theta}_x \mathbf{x}_t$$

$$\mathbf{y}_t = \boldsymbol{\theta}_y \boldsymbol{\phi}(\mathbf{h}_t)$$

$$\frac{\partial E}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{S} \frac{\partial E_t}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial E_t}{\partial \boldsymbol{\theta}} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \prod_{i=k+1}^{t} \boldsymbol{\theta}^T \operatorname{diag}[\boldsymbol{\phi}'(\mathbf{h}_{i-1})]$$

$$\left\| \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right\| \leq \|\boldsymbol{\theta}^T\| \|\operatorname{diag}[\boldsymbol{\phi}'(\mathbf{h}_{i-1})]\| \leq \gamma_\theta \gamma_\phi$$

$$\left\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \right\| \leq (\gamma_\theta \gamma_\phi)^{t-k}$$

https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

$$\mathbf{h}_t = \boldsymbol{\theta}\boldsymbol{\phi}(\mathbf{h}_{t-1}) + \boldsymbol{\theta}_x \mathbf{x}_t$$

$$\mathbf{y}_t = \boldsymbol{\theta}_y \boldsymbol{\phi}(\mathbf{h}_t)$$

$$\frac{\partial E}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{S} \frac{\partial E_t}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial E_t}{\partial \boldsymbol{\theta}} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{h}_k}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} = \prod_{i=k+1}^{t} \boldsymbol{\theta}^T \mathrm{diag}[\boldsymbol{\phi}'(\mathbf{h}_{i-1})]$$

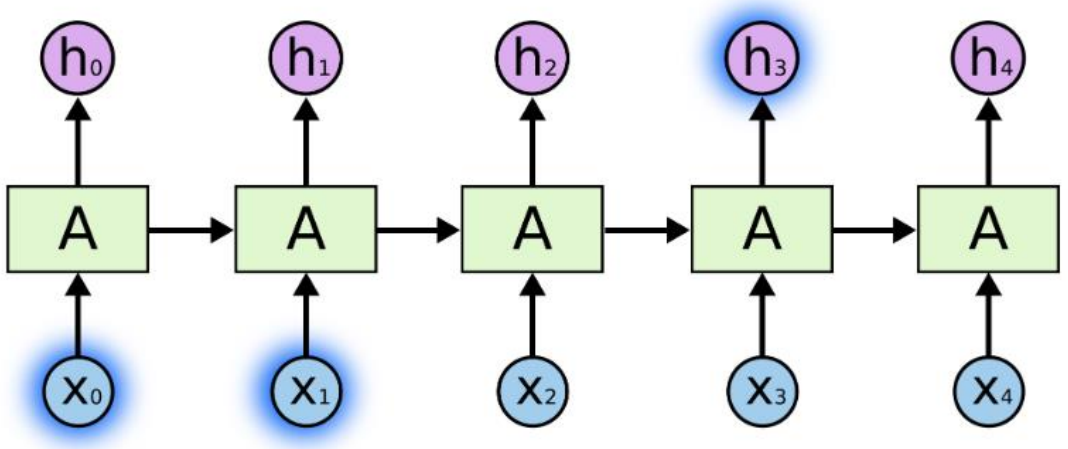
$$\left\| \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \right\| \leq \|\boldsymbol{\theta}^T\| \|\mathrm{diag}[\boldsymbol{\phi}'(\mathbf{h}_{i-1})]\| \leq \gamma_\theta \gamma_\phi$$

$$\left\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \right\| \leq (\gamma_\theta \gamma_\phi)^{t-k}$$

https://www.cs.ox.ac.uk/people/nando.defreitas/machinelearning/

Gradient Vanishing

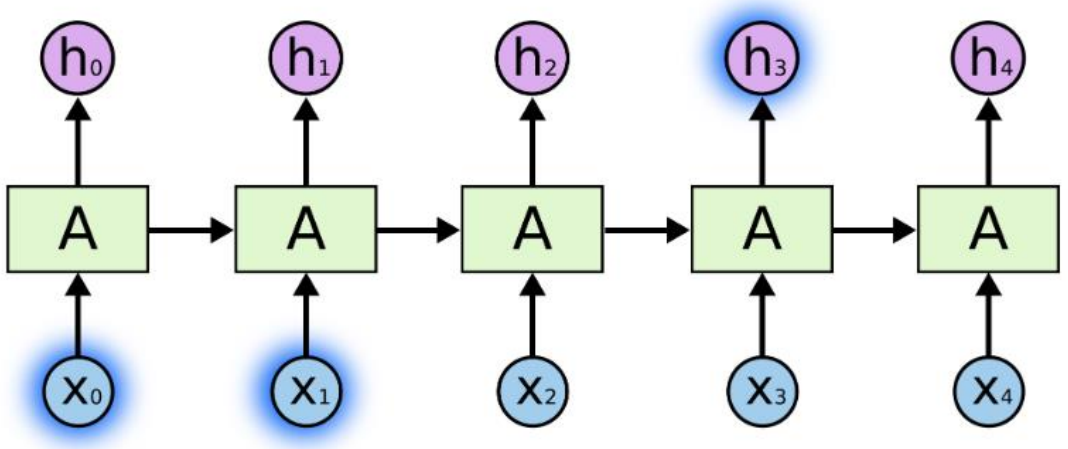
the clouds are in the ___



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

Gradient Vanishing

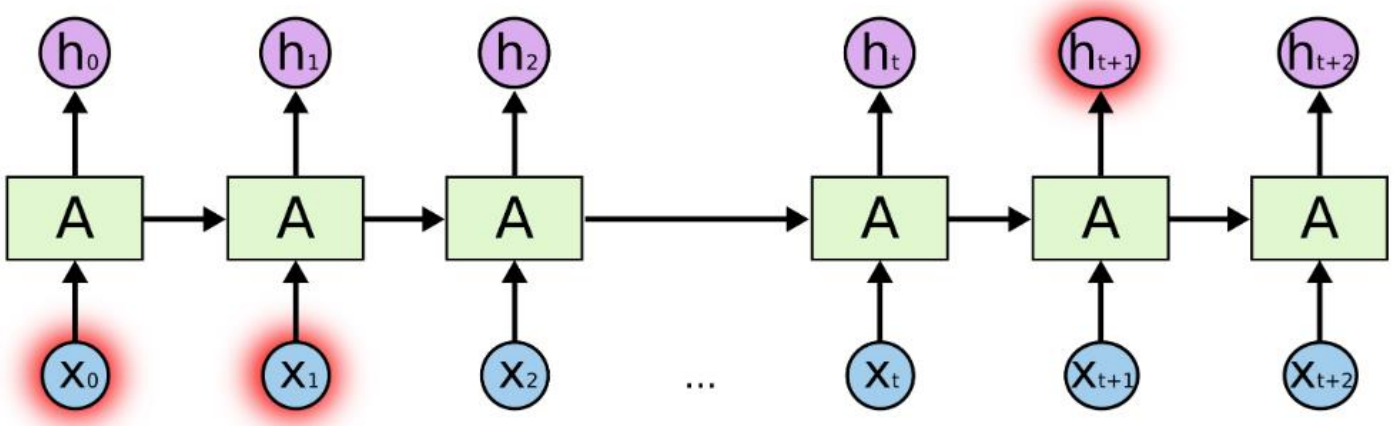
the clouds are in the sky



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

Gradient Vanishing

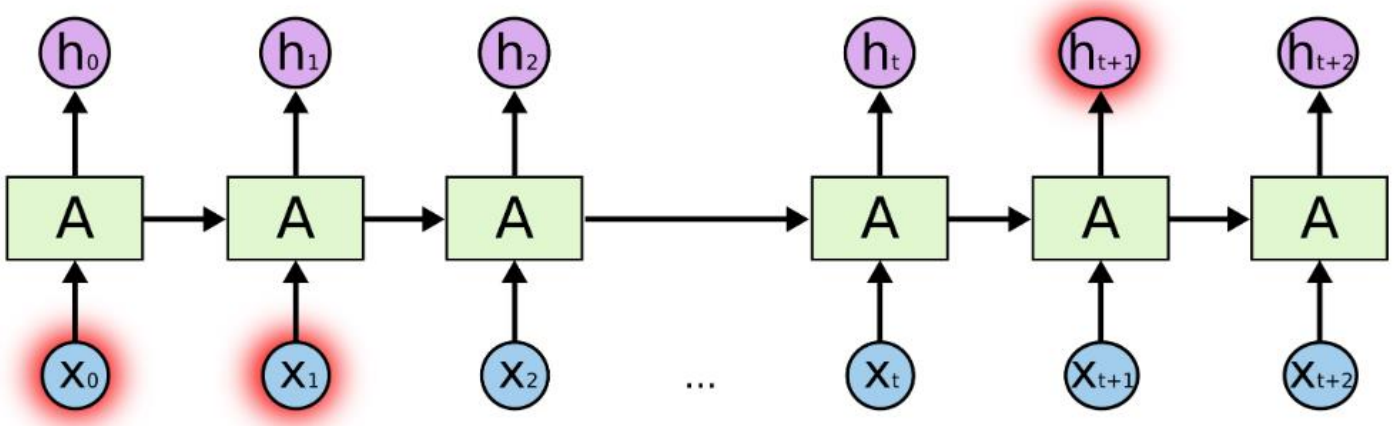
I grew up in France… I speak fluent _____.



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

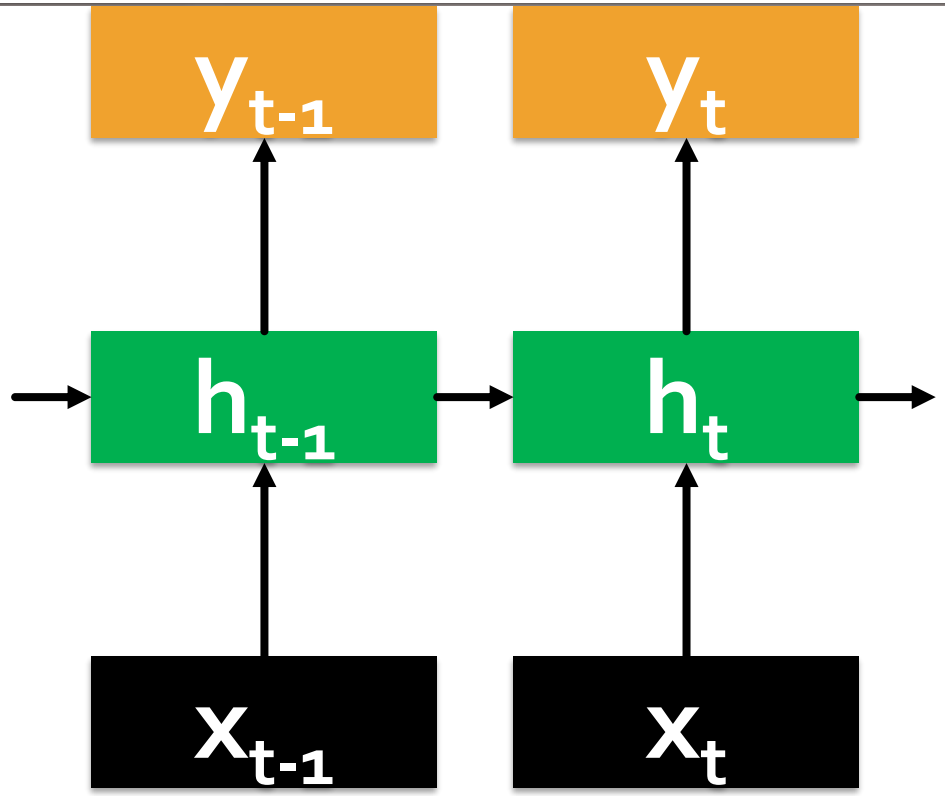
Gradient Vanishing

I grew up in France… I speak fluent French.



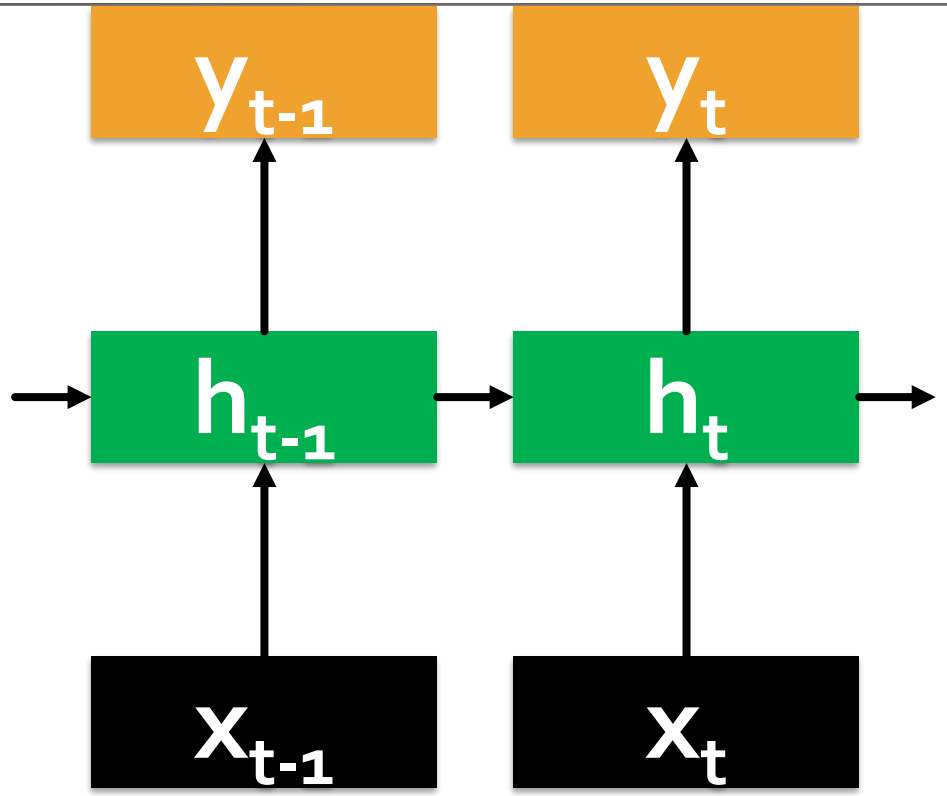
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

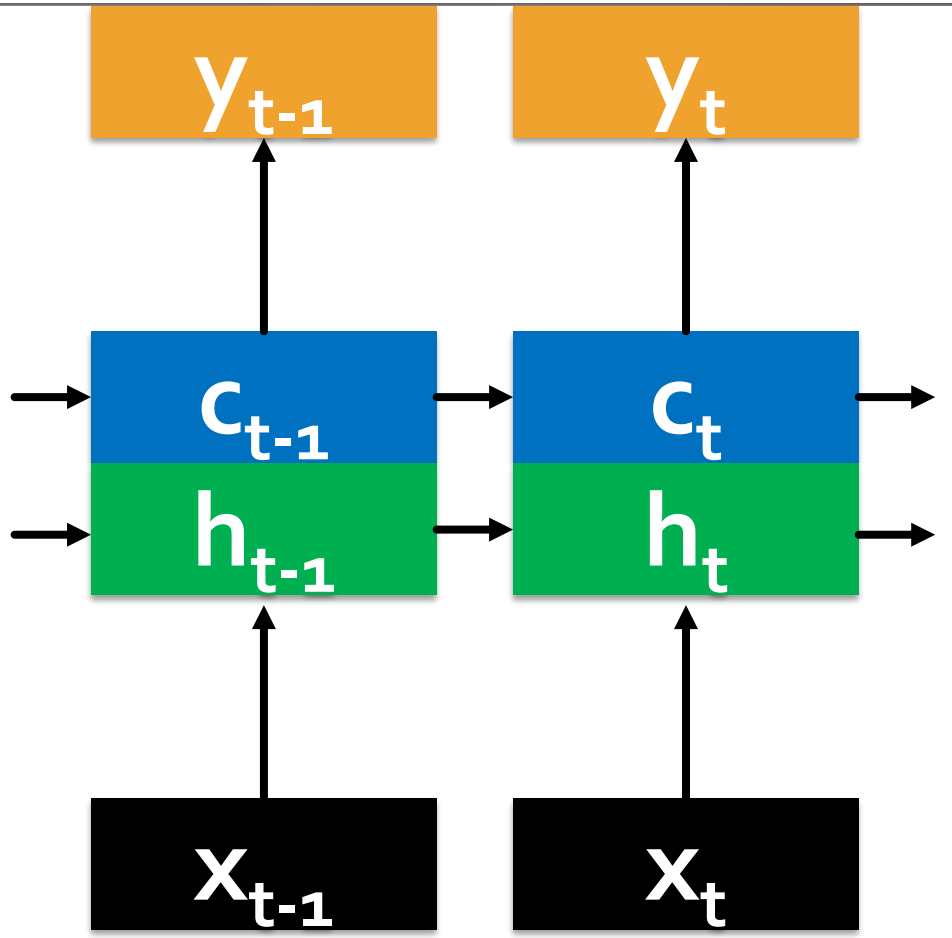
RNN



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

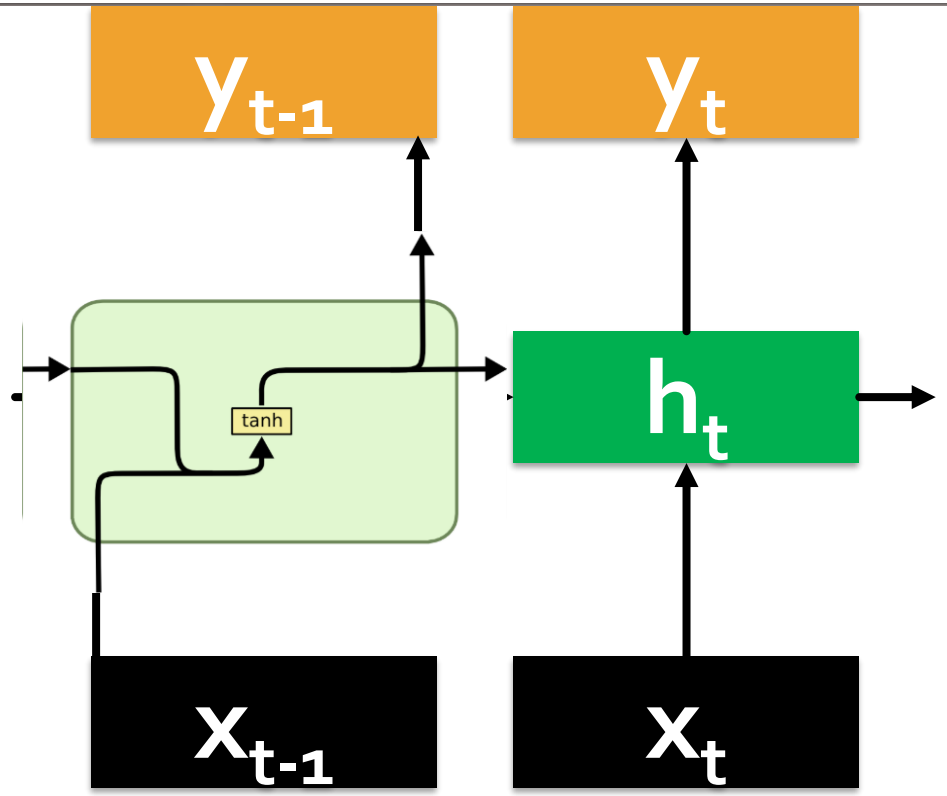
LSTM

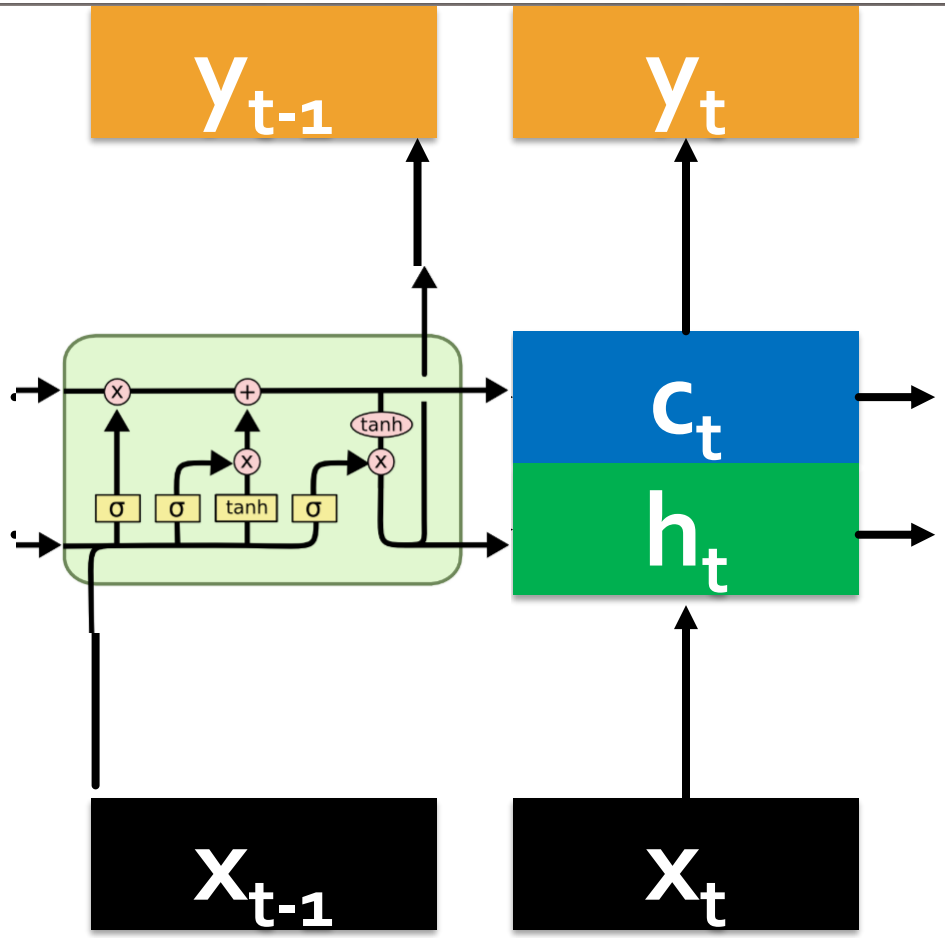




http://colah.github.io/posts/2015-08-Understanding-LSTMs/

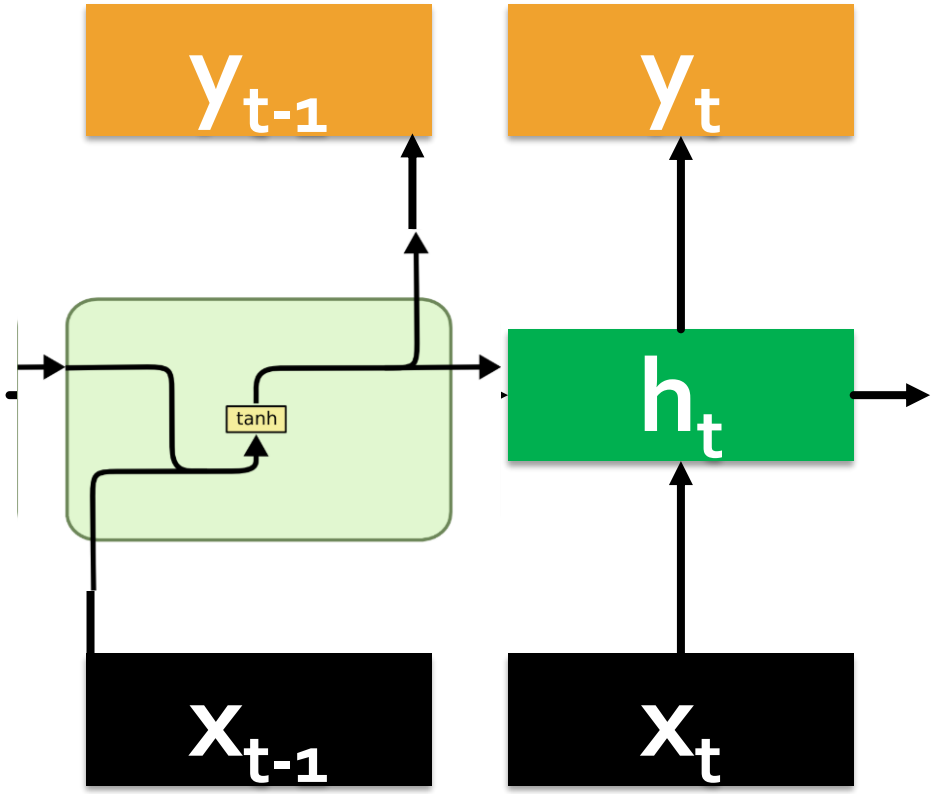
LSTM

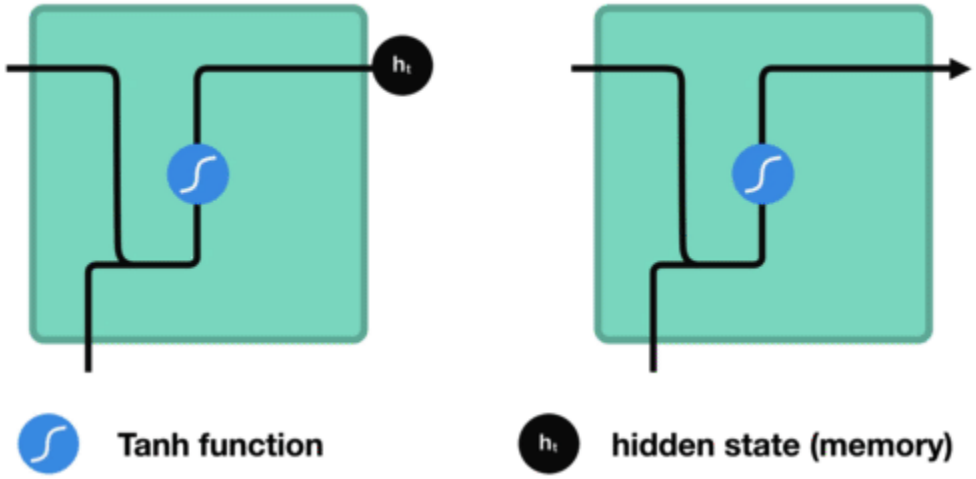




http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN



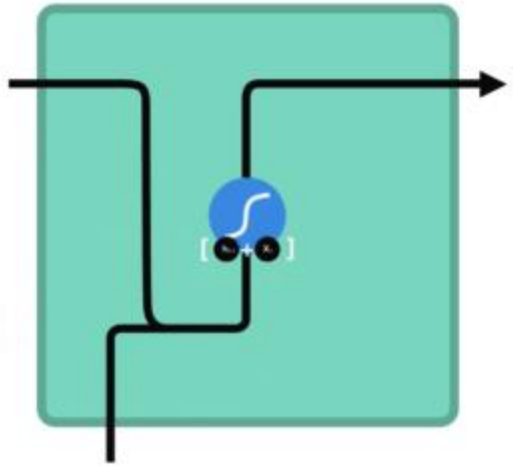


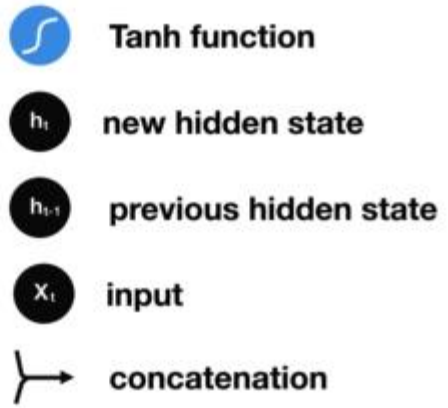
https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

RNN Cell

RNN

$$h_t = \tanh(w_{ih} x_t + b_{ih} + w_{hh} h_{(t-1)} + b_{hh})$$

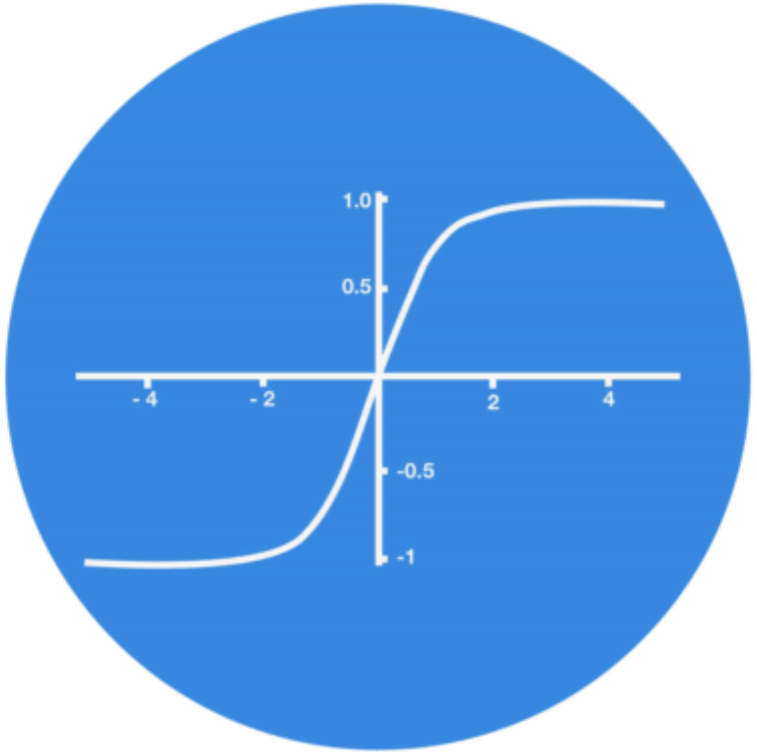




https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

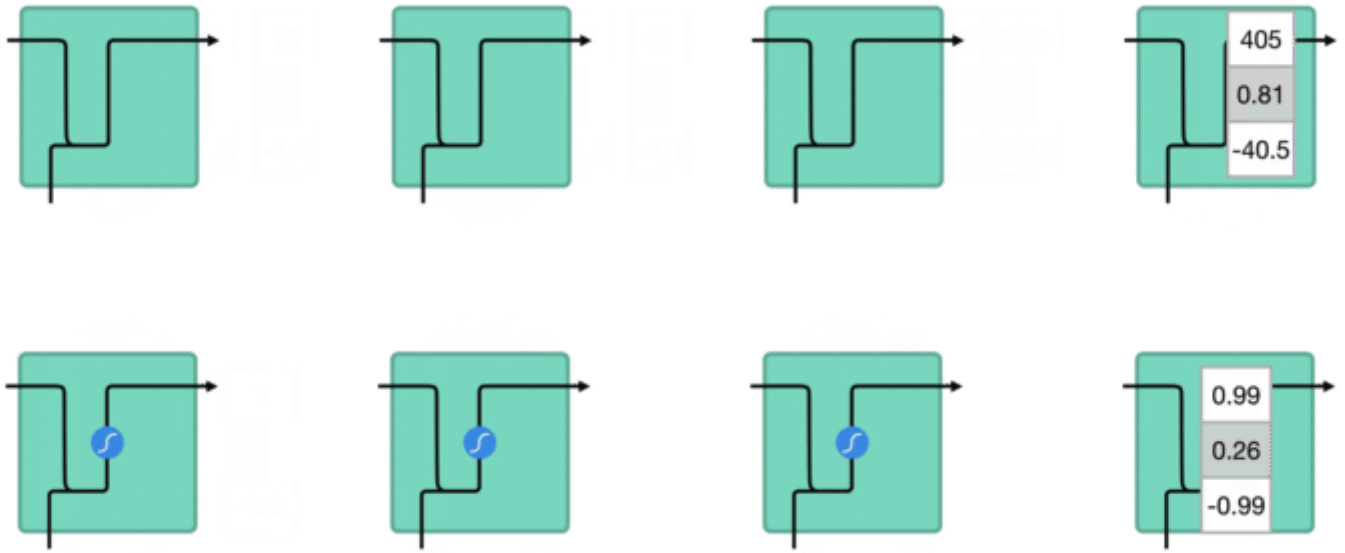
tanh

5
0.1
-0.5



https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Why



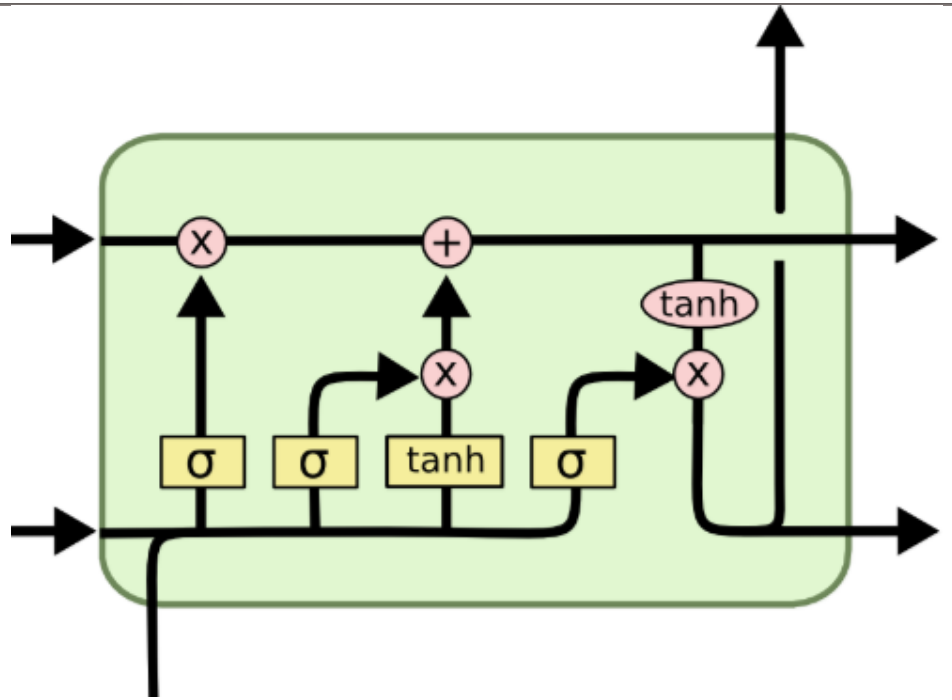
https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

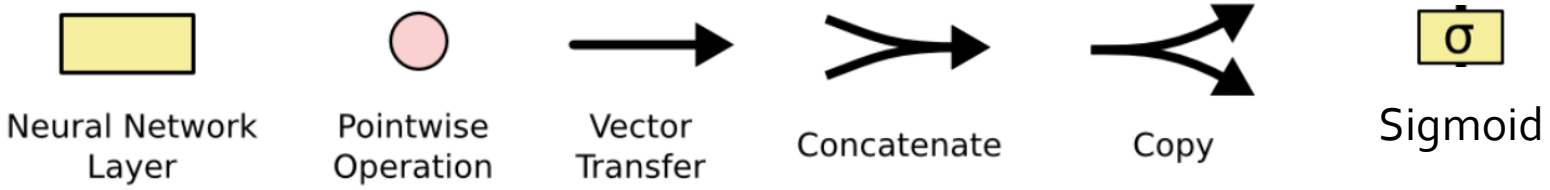
Why Tanh

Recent research on deep feedforward networks has also produced some impressive results [19, 3] and there is now a consensus that for deep networks, rectified linear units (ReLUs) are easier to train than the logistic or tanh units that were used for many years [27, 40]. At first sight, ReLUs seem inappropriate for RNNs because they can have very large outputs so they might be expected to be far more likely to explode than units that have bounded values. A second aim of this paper is to explore whether ReLUs can be made to work well in RNNs and whether the ease of optimizing them in feedforward nets transfers to RNNs.

arxiv：[1504.00941] A Simple Way to Initialize Recurrent Networks of Rectified Linear Units)

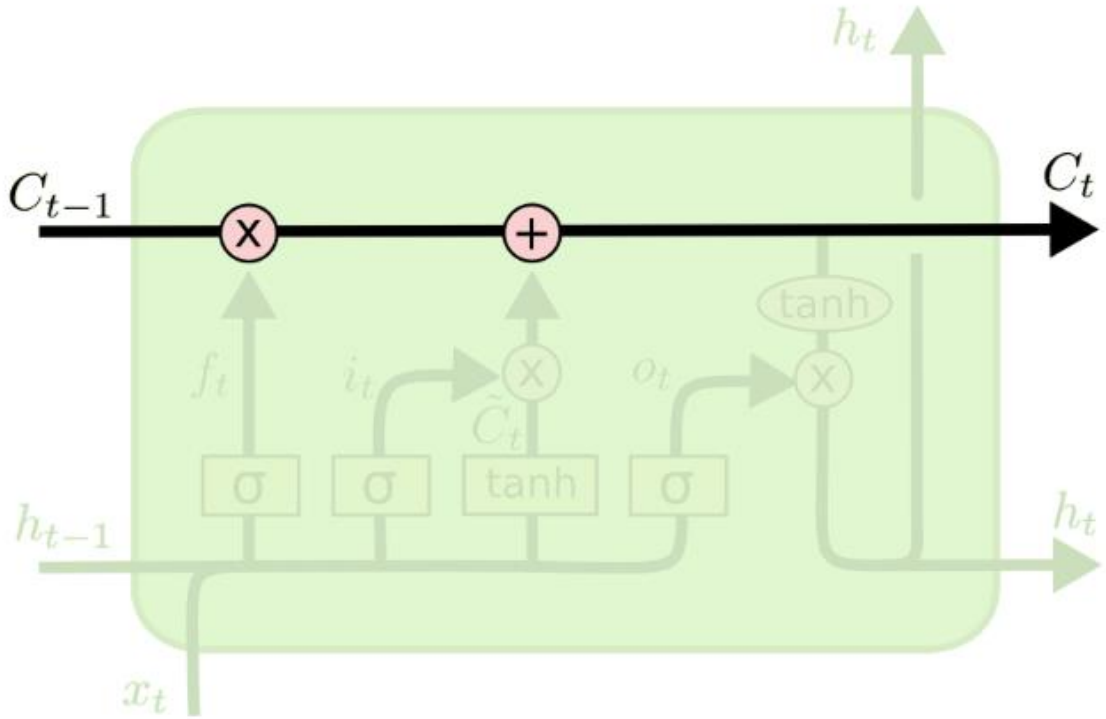
LSTM

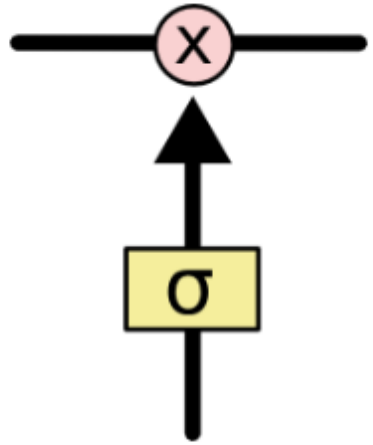




http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Without Changing Memory

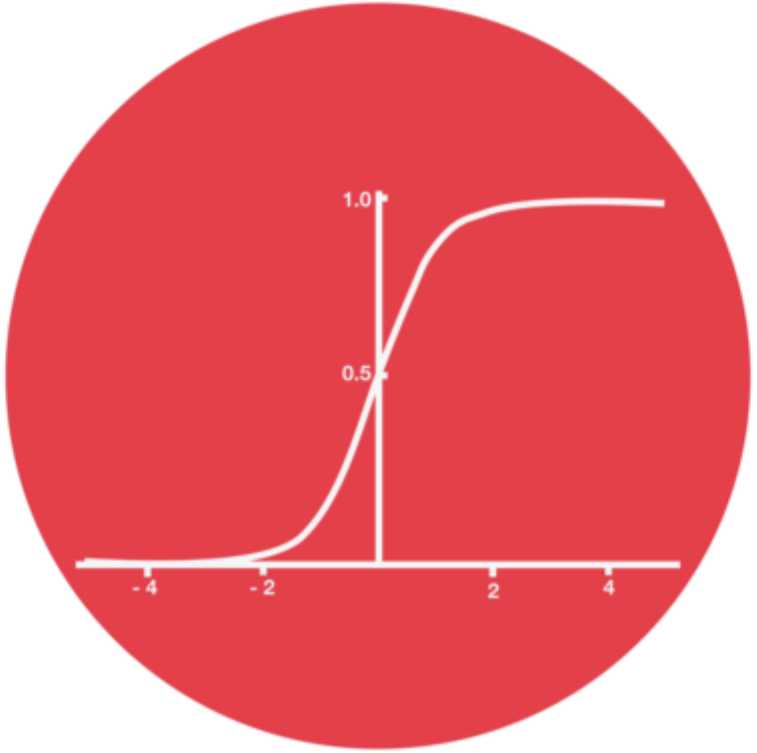


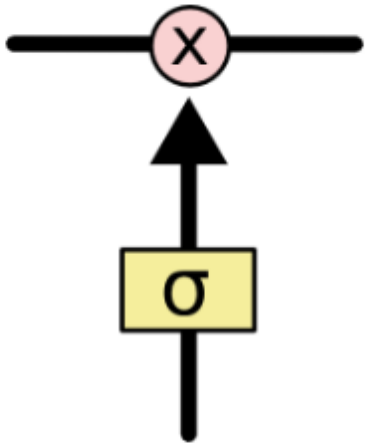


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gate

5
0.1
-0.5



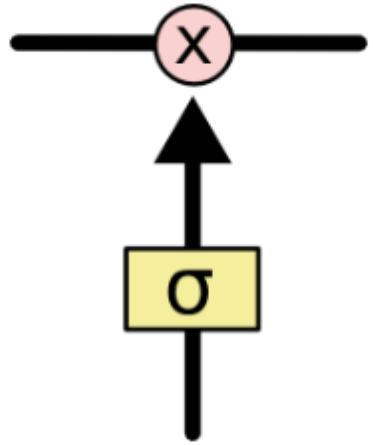


https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Hadamard Product

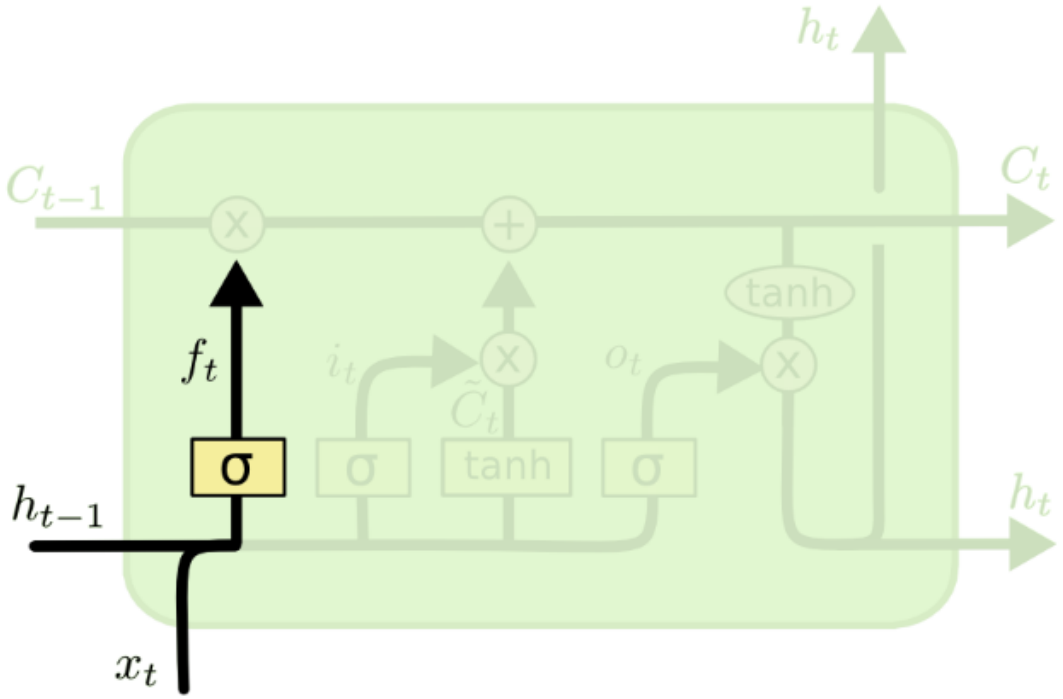
$$A \circ B = C$$

$$\begin{bmatrix} a_{11} & \cdots & a_{1k} \\ a_{21} & \cdots & a_{2k} \\ \vdots & \ddots & \vdots \\ a_{j1} & \cdots & a_{jk} \end{bmatrix} \circ \begin{bmatrix} b_{11} & \cdots & b_{1k} \\ b_{21} & \cdots & b_{2k} \\ \vdots & \ddots & \vdots \\ b_{j1} & \cdots & b_{jk} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} & \cdots & a_{1k}b_{1k} \\ a_{21}b_{21} & \cdots & a_{2k}b_{2k} \\ \vdots & \ddots & \vdots \\ a_{j1}b_{j1} & \cdots & a_{jk}b_{jk} \end{bmatrix}$$



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

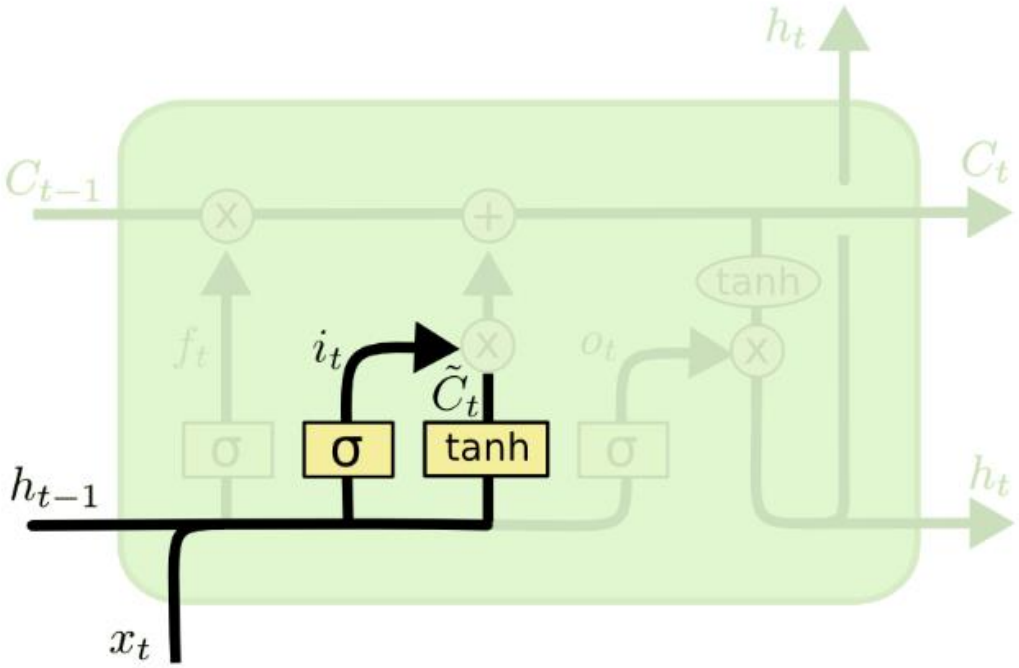
Forget Gate Layer



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

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

Input Gate Layer

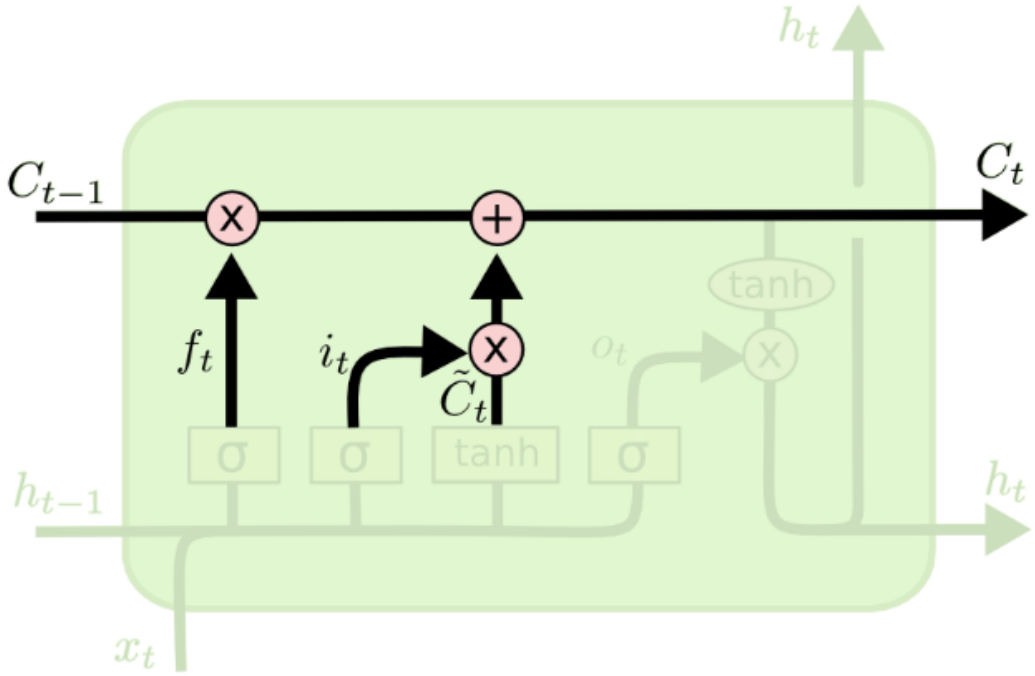


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

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

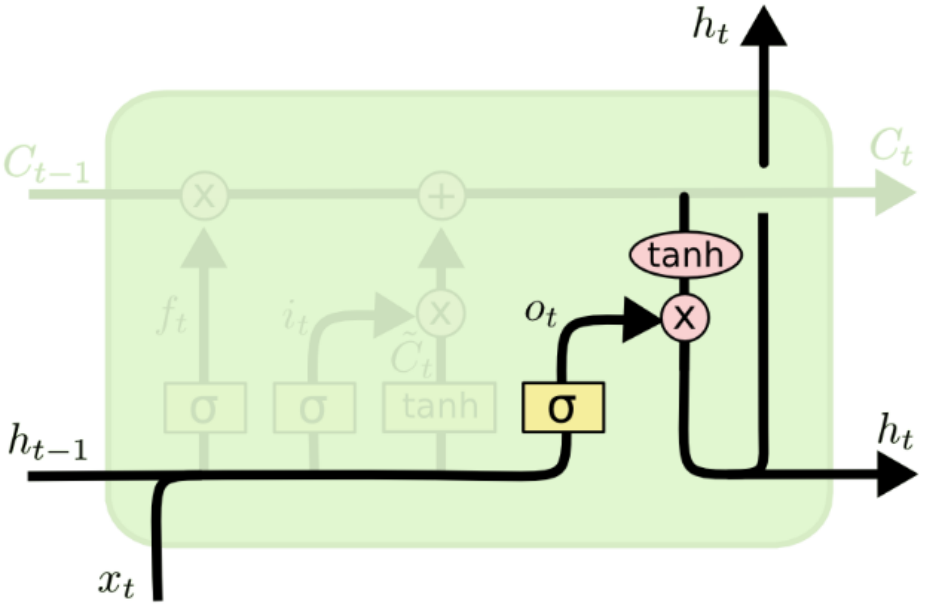
Update Cell State



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

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

Update Output



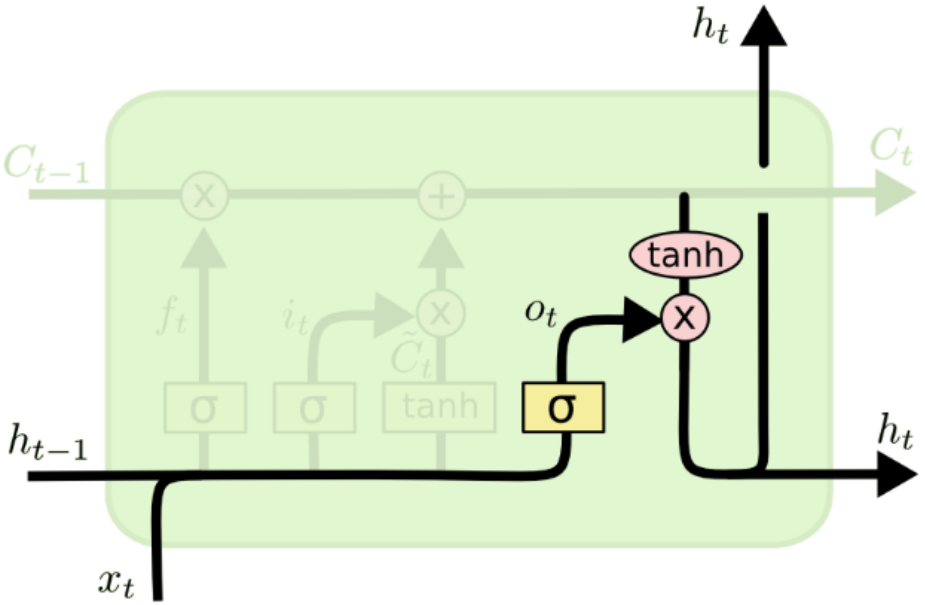
$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

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

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

Update Output

I grew up in France… I speak fluent French.

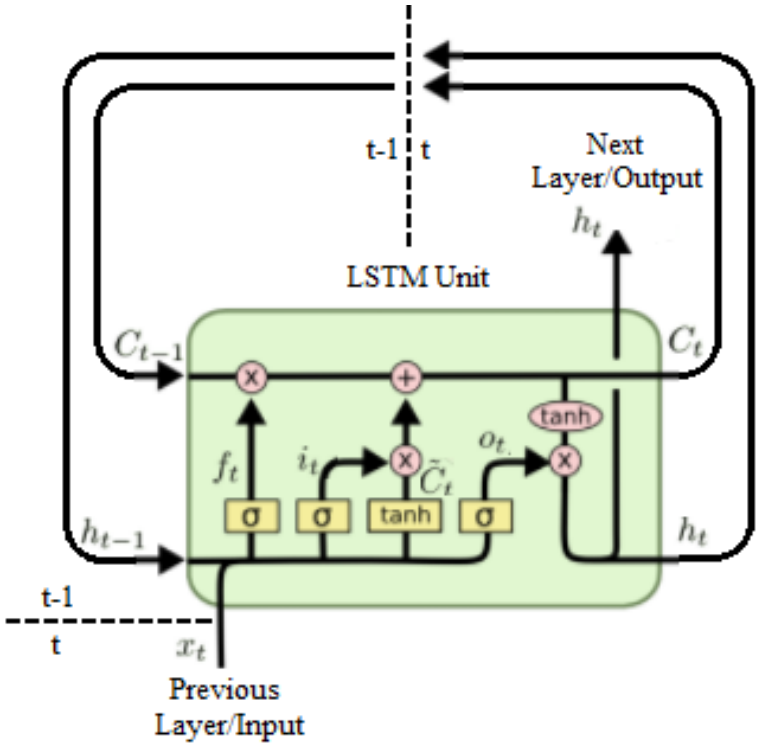


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

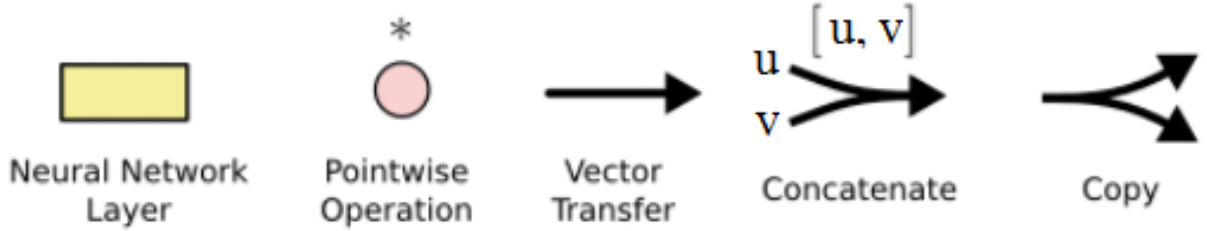
$$h_t = o_t * \tanh\left(C_t\right)$$

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

LSTM

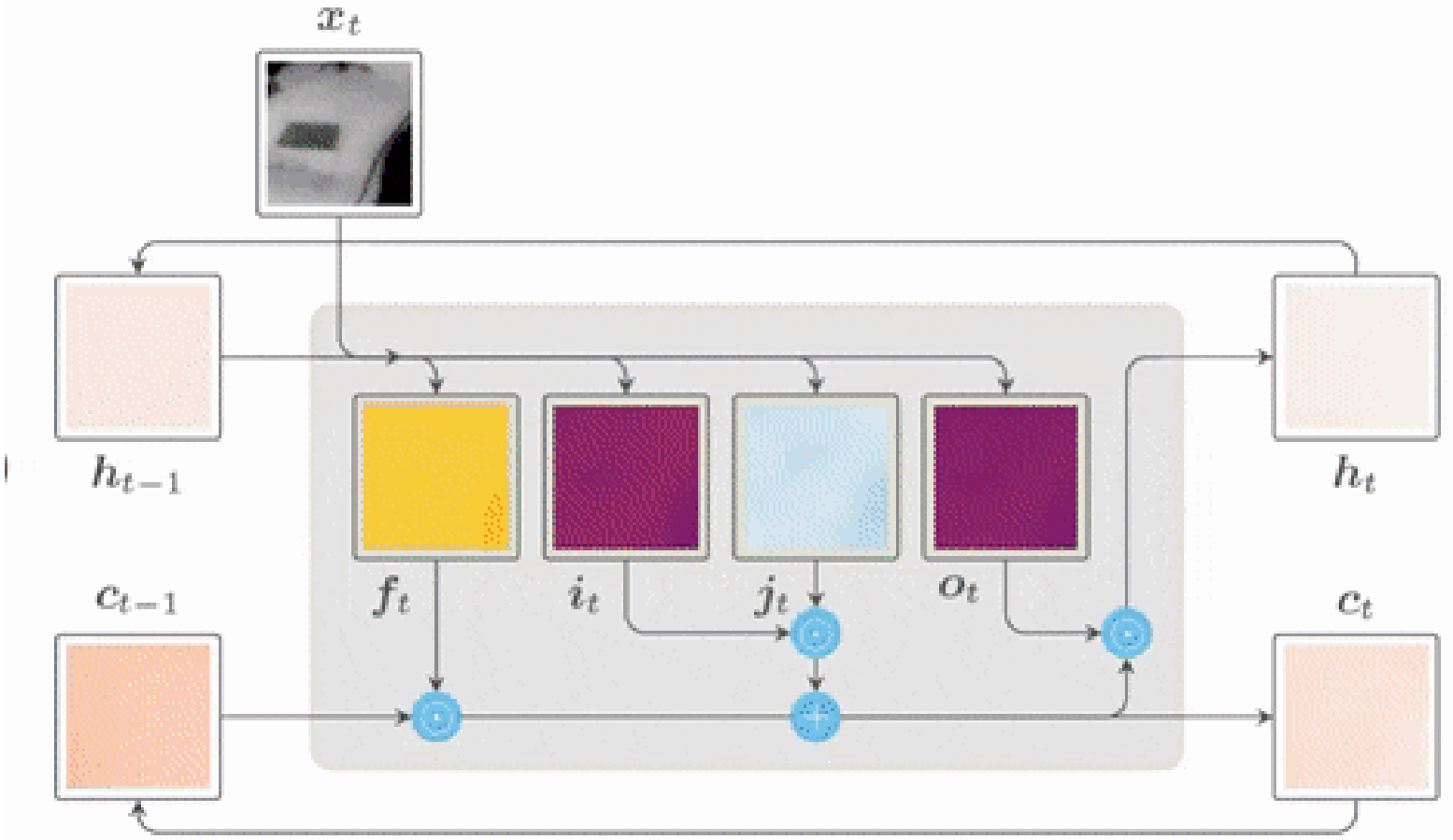


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$



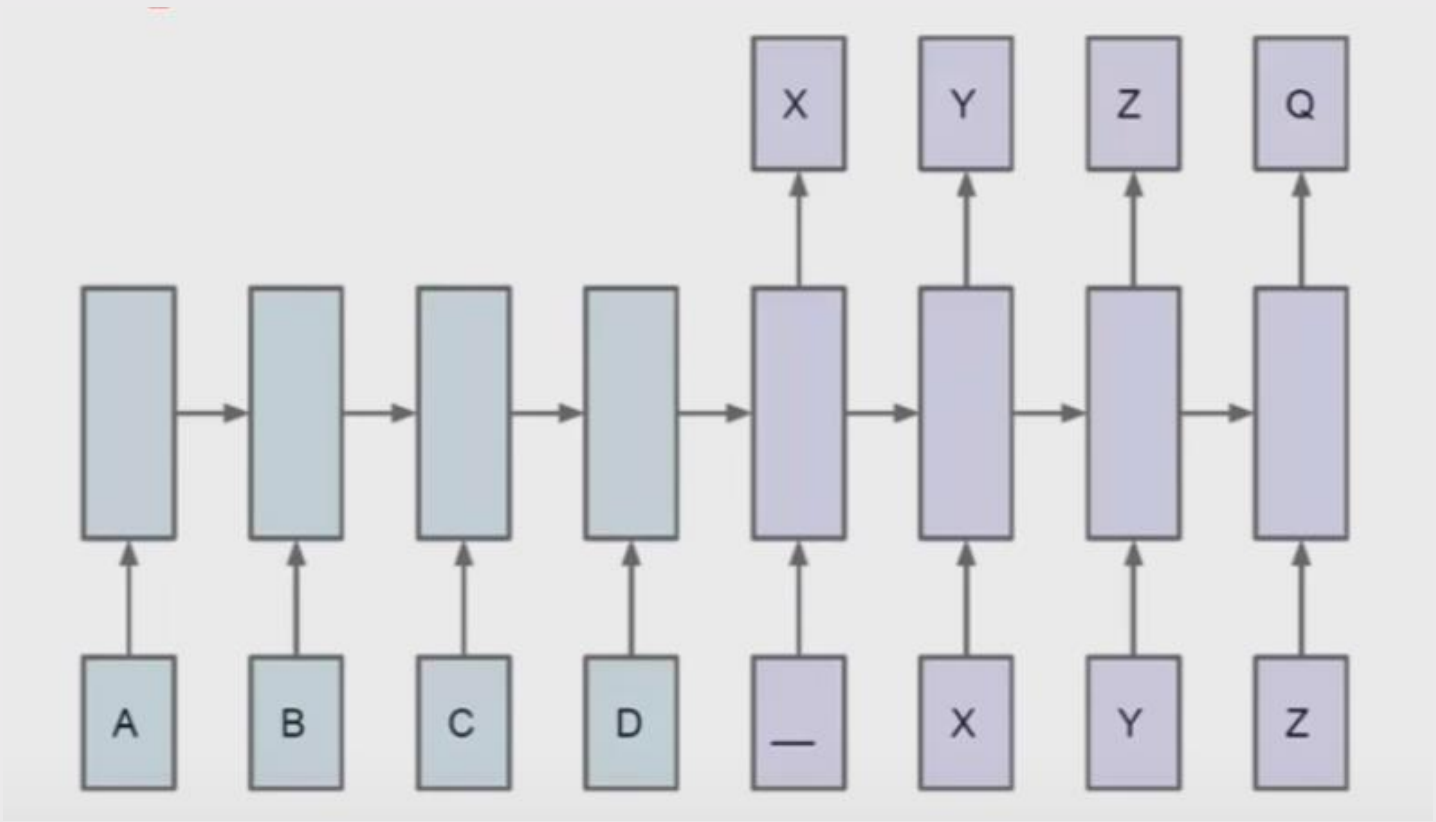
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

What's going on

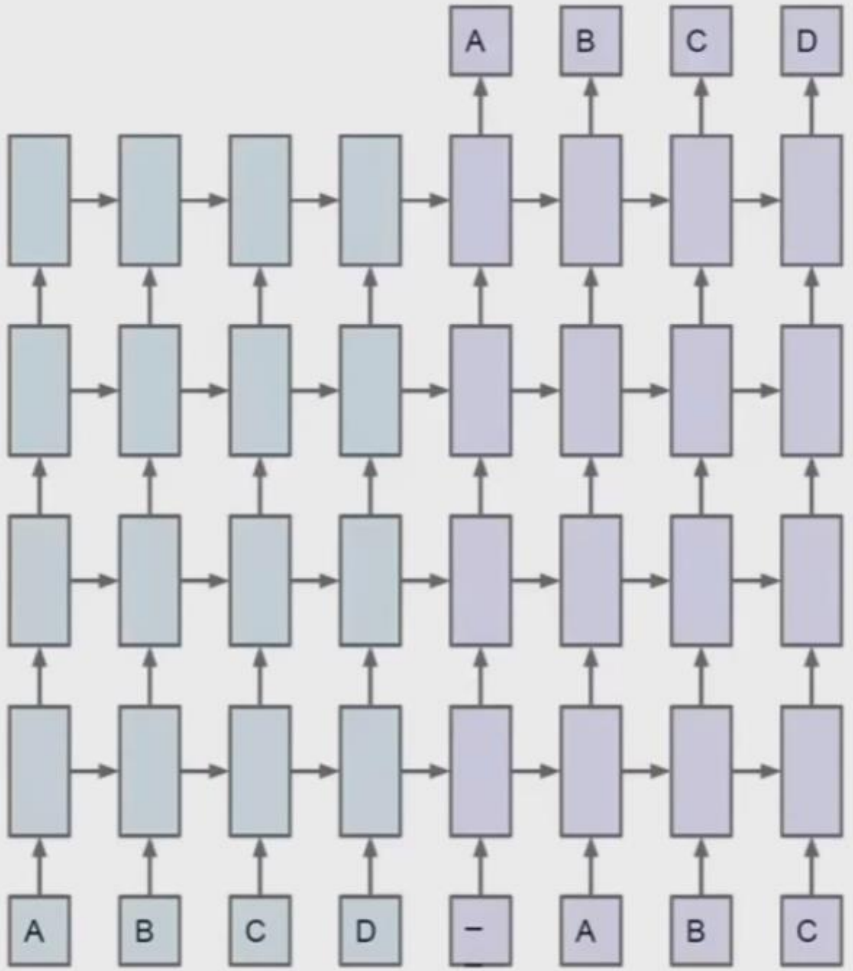


https://gfycat.com/gifs/detail/poorincrediblecrab

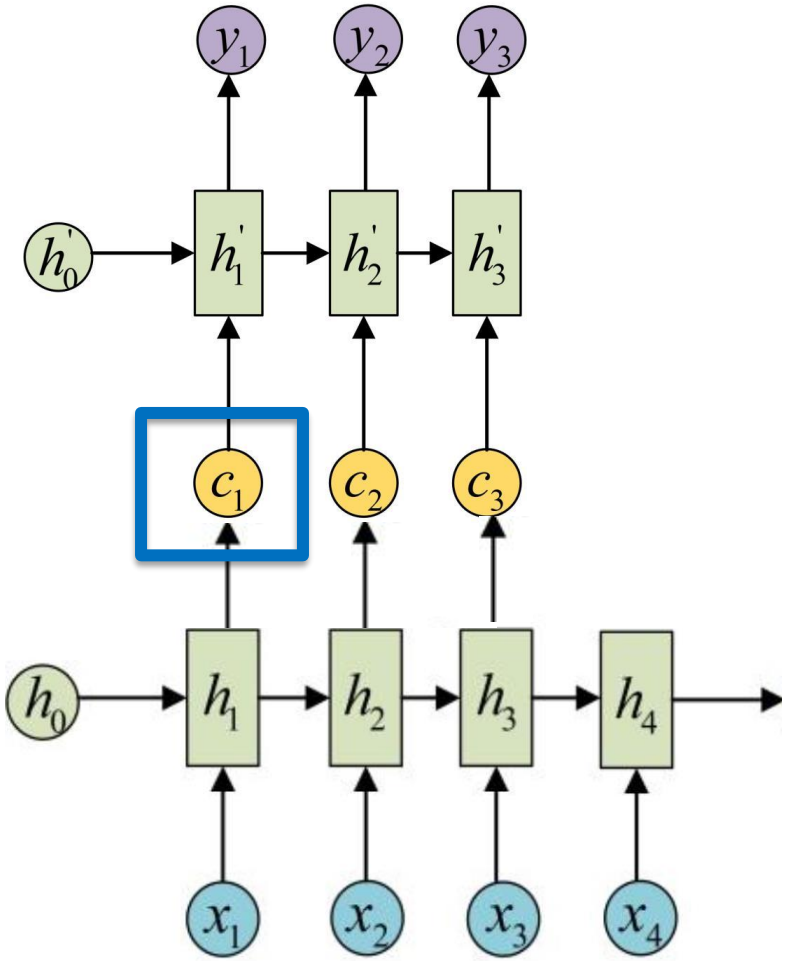
One Layer

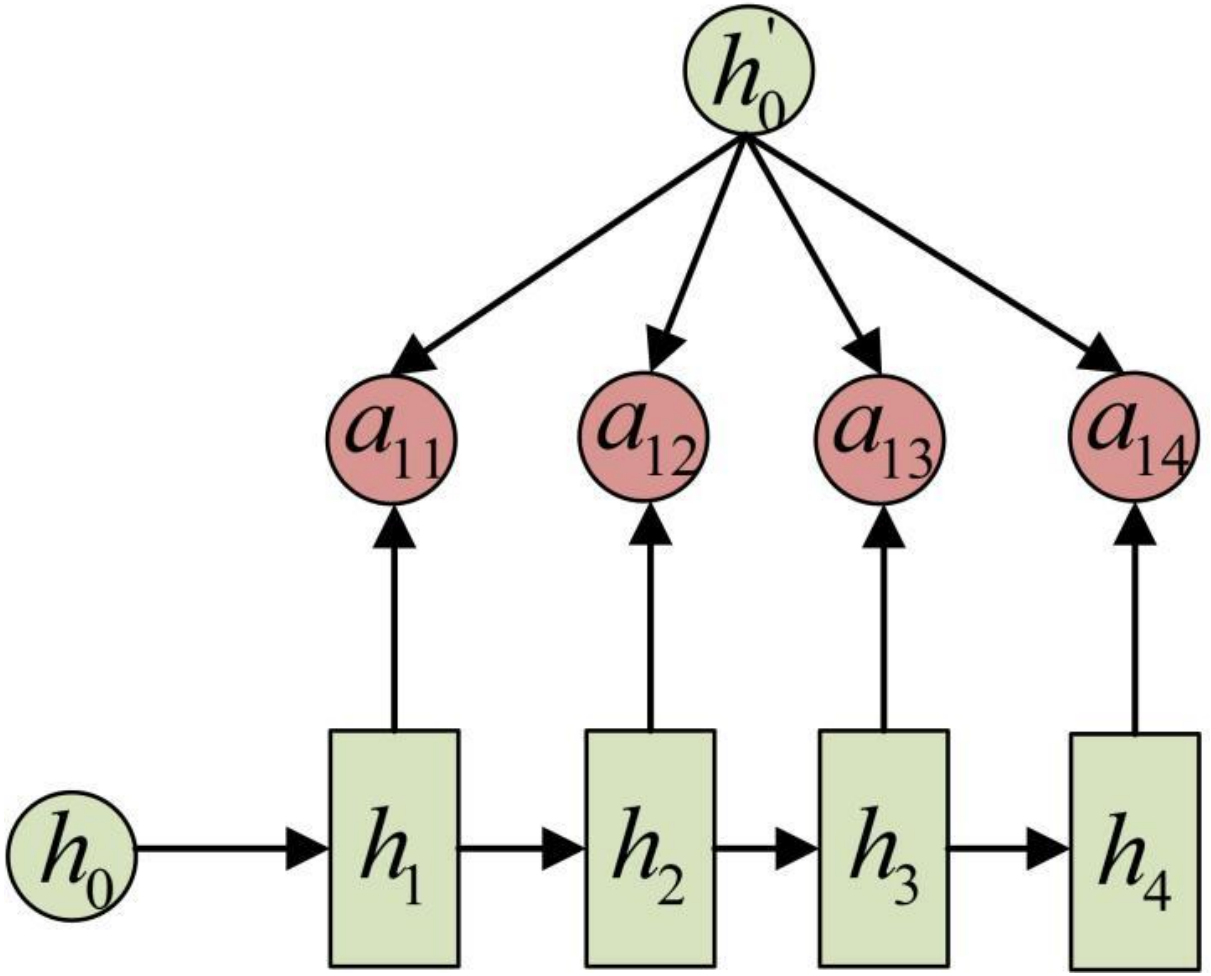


Deep Learning

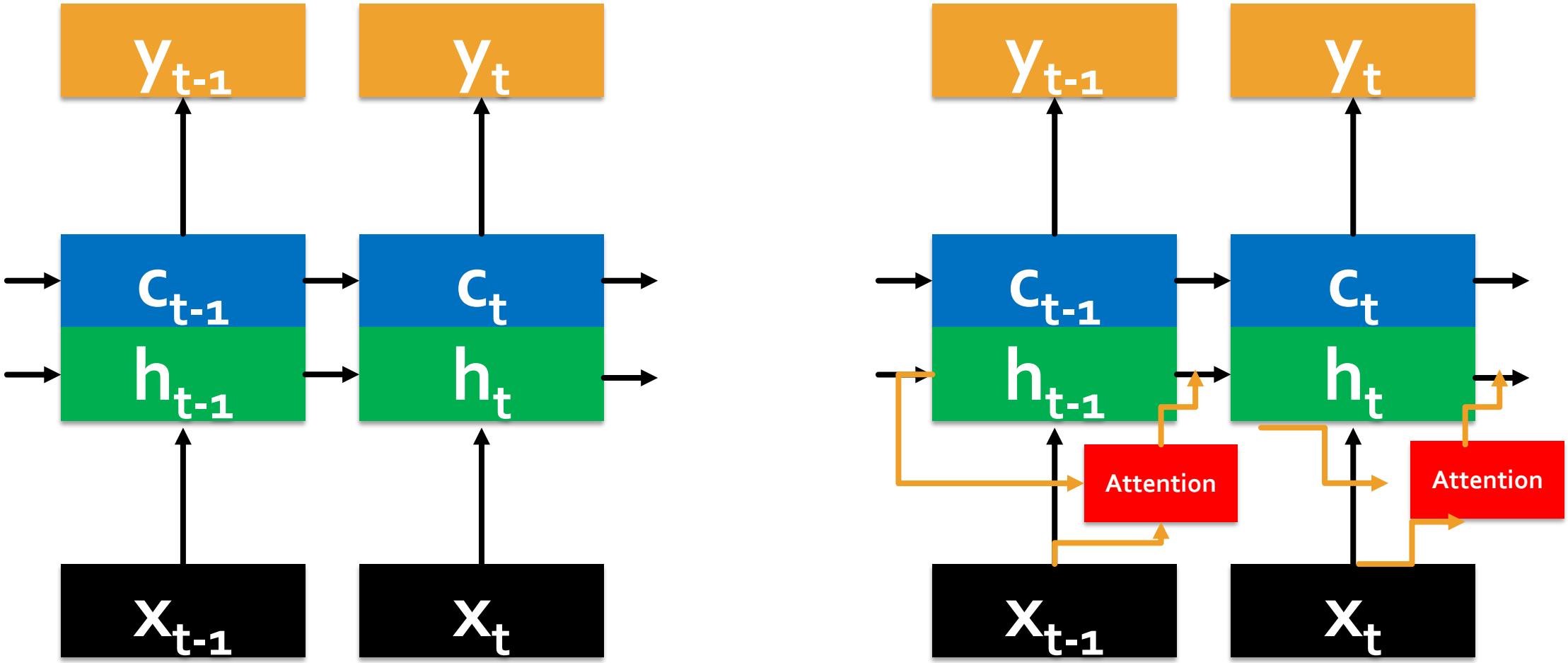


Attention Model

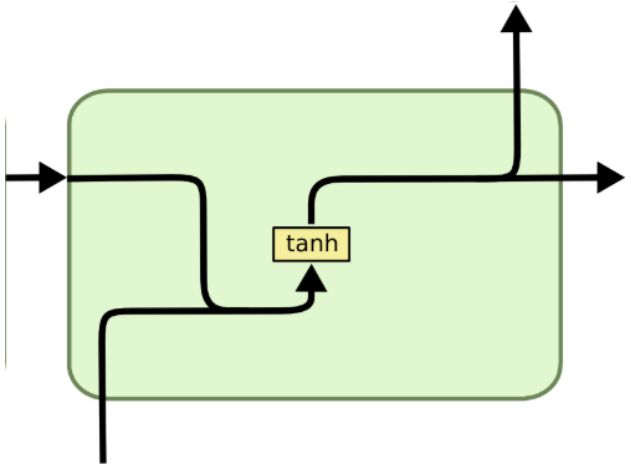




Attention Model with LSTM



Pytorch



class **torch.nn.RNN**(**args*, ***kwargs*) [source]

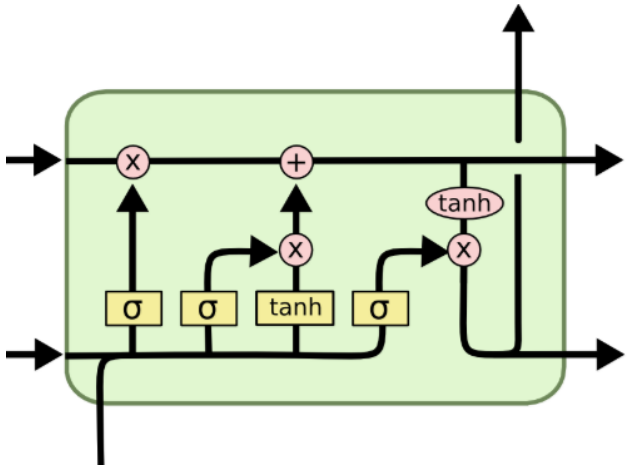
Applies a multi-layer Elman RNN with *tanh* or *ReLU* non-linearity to an input sequence.

For each element in the input sequence, each layer computes the following function:

$$h_t = \tanh(w_{ih}x_t + b_{ih} + w_{hh}h_{(t-1)} + b_{hh})$$

where h_t is the hidden state at time *t*, x_t is the input at time *t*, and $h_{(t-1)}$ is the hidden state of the previous layer at time *t-1* or the initial hidden state at time *0*. If `nonlinearity` is *'relu'*, then *ReLU* is used instead of *tanh*.

Pytorch



class **torch.nn.LSTM**(**args, **kwargs*) [source]

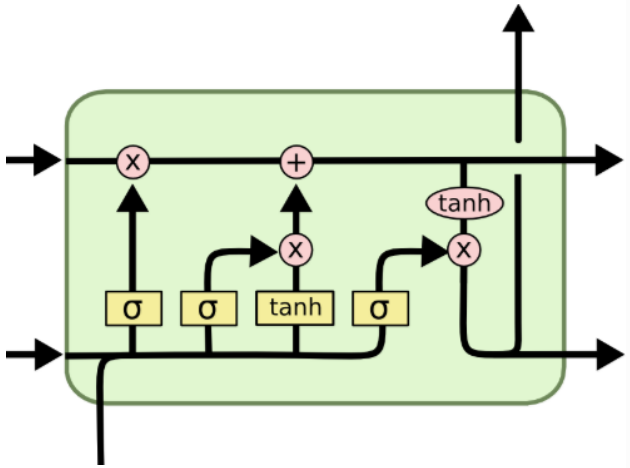
Applies a multi-layer long short-term memory (LSTM) RNN to an input sequence.

For each element in the input sequence, each layer computes the following function:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t c_{(t-1)} + i_t g_t \\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$

where h_t is the hidden state at time *t*, c_t is the cell state at time *t*, x_t is the input at time *t*, $h_{(t-1)}$ is the hidden state of the previous layer at time *t-1* or the initial hidden state at time *0*, and i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, respectively. σ is the sigmoid function.

Pytorch



Parameters:

- **input_size** – The number of expected features in the input *x*
- **hidden_size** – The number of features in the hidden state *h*
- **num_layers** – Number of recurrent layers. E.g., setting `num_layers=2` would mean stacking two LSTMs together to form a *stacked LSTM*, with the second LSTM taking in outputs of the first LSTM and computing the final results. Default: 1
- **bias** – If `False`, then the layer does not use bias weights *b_ih* and *b_hh*. Default: `True`
- **batch_first** – If `True`, then the input and output tensors are provided as (batch, seq, feature). Default: `False`
- **dropout** – If non-zero, introduces a *Dropout* layer on the outputs of each LSTM layer except the last layer, with dropout probability equal to `dropout`. Default: 0
- **bidirectional** – If `True`, becomes a bidirectional LSTM. Default: `False`

Pseudo Code

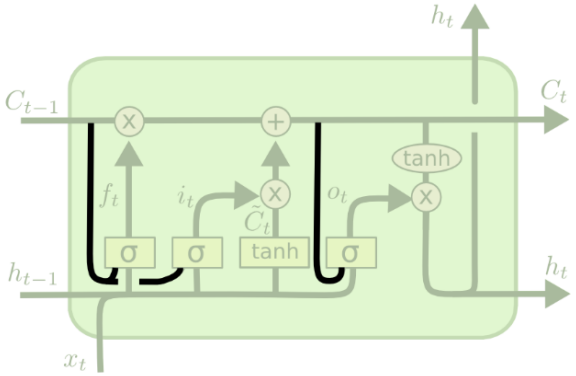
$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t c_{(t-1)} + i_t g_t \\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$

```
def LSTMCELL(prev_ct, prev_ht, input):
    combine = prev_ht + input
    ft = forget_layer(combine)
    candidate = candidate_layer(combine)
    it = input_layer(combine)
    Ct = prev_ct * ft + candidate * it
    ot = output_layer(combine)
    ht = ot * tanh(Ct)
    return ht, Ct


ct = [0, 0, 0]
ht = [0, 0, 0]

for input in inputs:
    ct, ht = LSTMCELL(ct, ht, input)
```

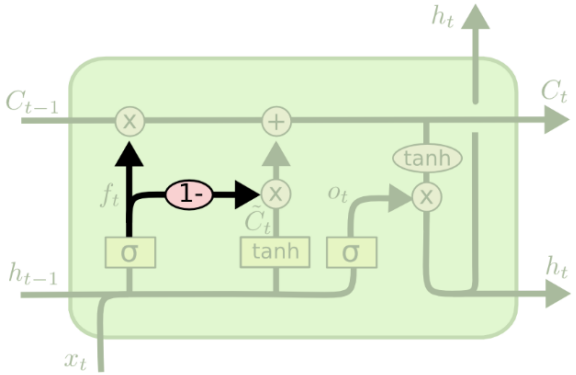
https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

https://pytorch.org/tutorials/beginner/nlp/sequence_models_tutorial.html

Variations

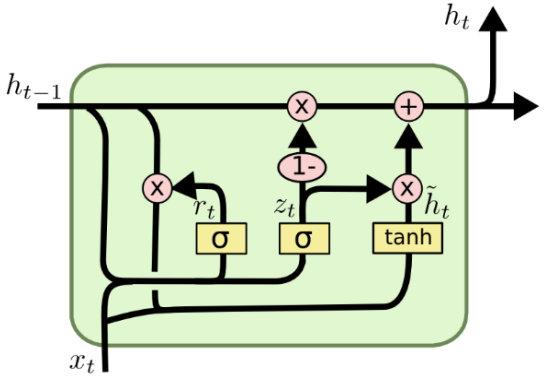


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$



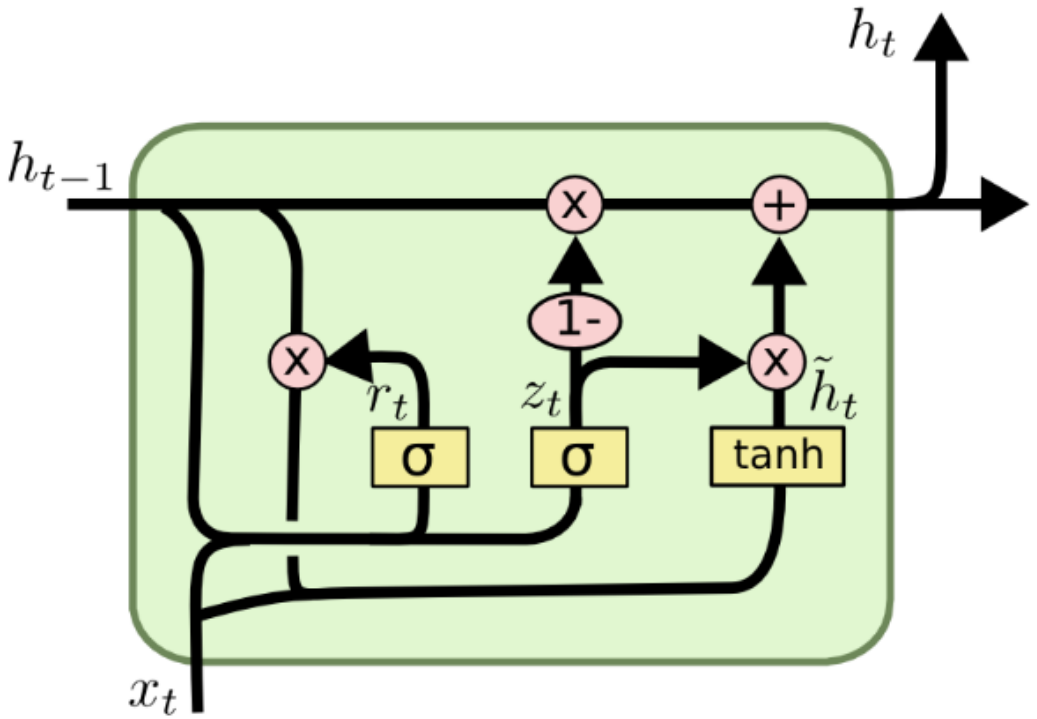
$$C_t = f_t * C_{t-1} + (\mathbf{1} - \boldsymbol{f_t}) * \tilde{C}_t$$

GRU



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

GRU



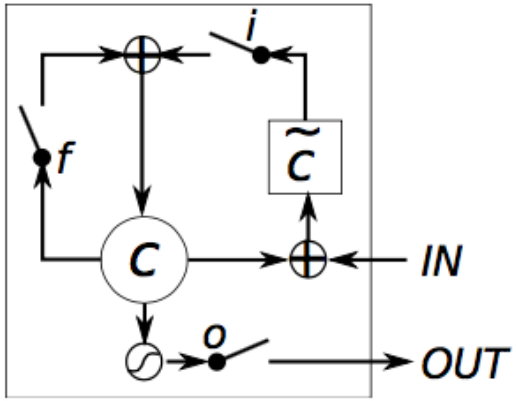
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

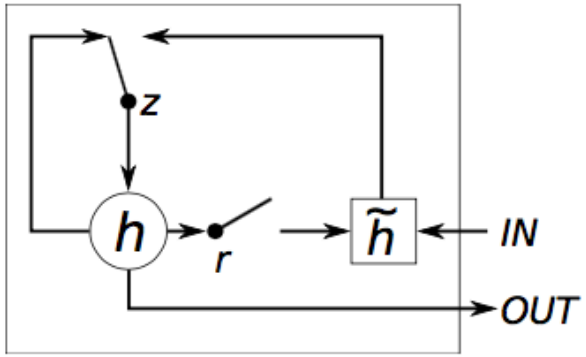
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

LSTM vs GRU



(a) Long Short-Term Memory



(b) Gated Recurrent Unit

Figure 1: Illustration of (a) LSTM and (b) gated recurrent units. (a) i, f and o are the input, forge and output gates, respectively. c and $\tilde{c}$ denote the memory cell and the new memory cell content. (b r and z are the reset and update gates, and h and $\tilde{h}$ are the activation and the candidate activation.

			tanh	GRU	LSTM
Music Datasets	Nottingham	train	3.22	2.79	3.08
		test	**3.13**	3.23	3.20
	JSB Chorales	train	8.82	6.94	8.15
		test	9.10	**8.54**	8.67
	MuseData	train	5.64	5.06	5.18
		test	6.23	**5.99**	6.23
	Piano-midi	train	5.64	4.93	6.49
		test	9.03	**8.82**	9.03
Ubisoft Datasets	Ubisoft dataset A	train	6.29	2.31	1.44
		test	6.44	3.59	**2.70**
	Ubisoft dataset B	train	7.61	0.38	0.80
		test	7.62	**0.88**	1.26

Table 2: The average negative log-probabilities of the training and test sets.

Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling

Conv LSTM

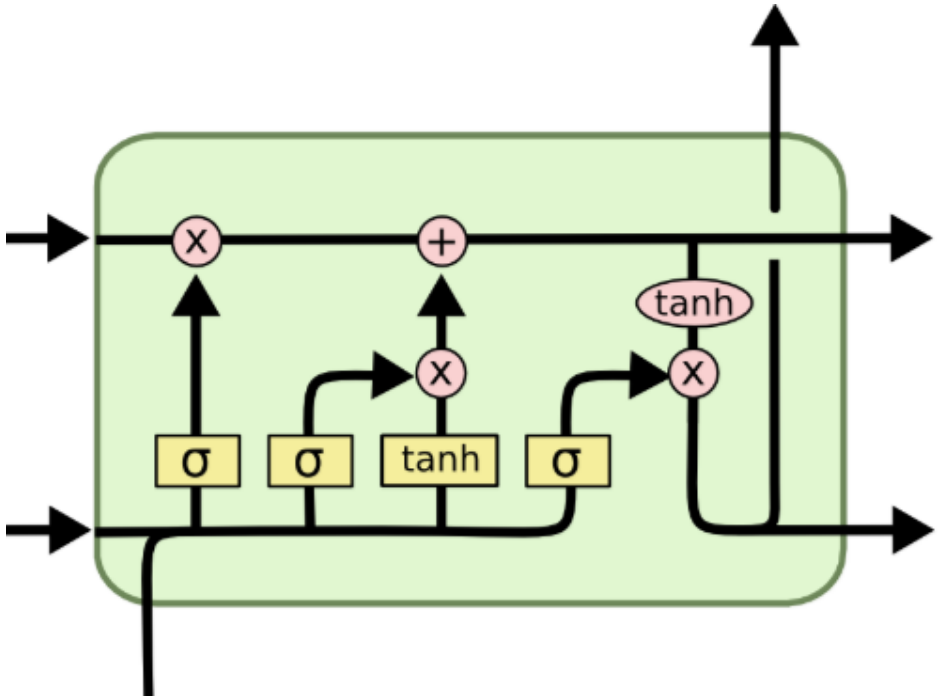
Convolutional LSTM Network: A Machine Learning Approach for ...

https://arxiv.org › cs ▼ 翻译此页

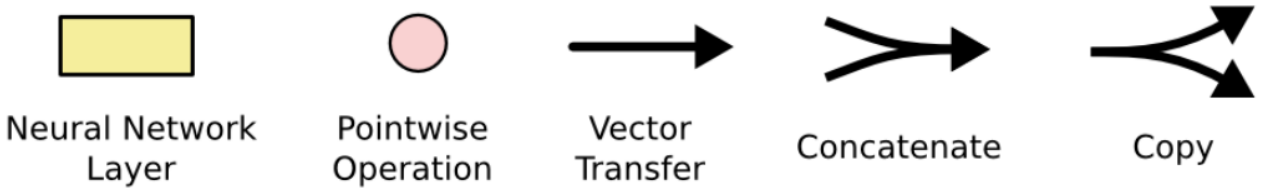
作者：X Shi - 2015 - 被引用次数：472 - 相关文章

2015年6月13日 - Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. The goal of precipitation nowcasting is to predict the future rainfall intensity in a local region over a relatively short period of time.

LSTM

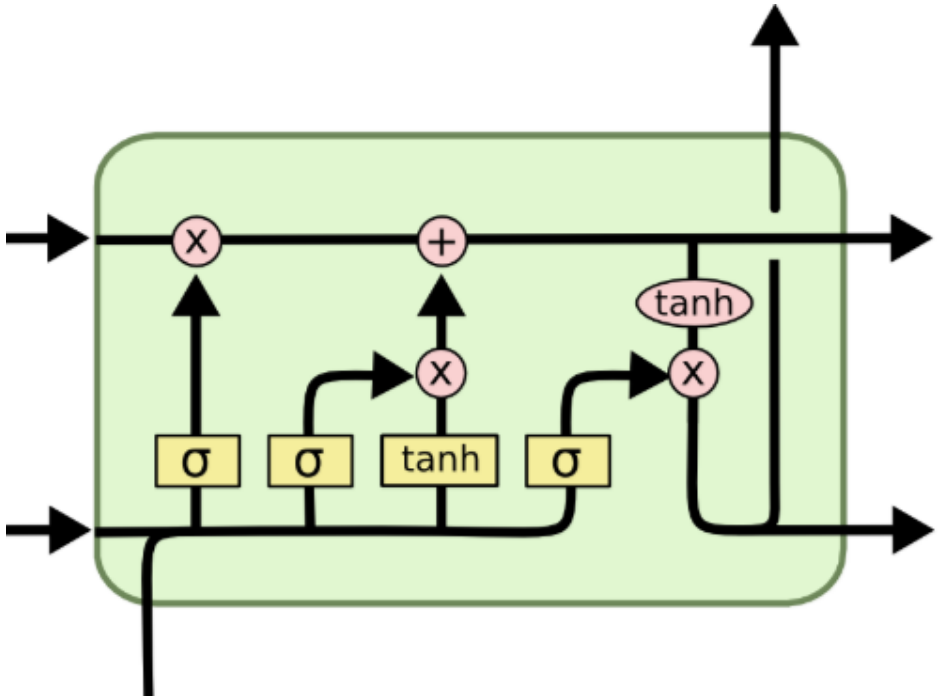


$$
\begin{aligned}
f_t &= \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right) \\
i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right) \\
h_t &= o_t * \tanh\left(C_t\right)
\end{aligned}
$$



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

ConvLSTM



$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o) \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f) \\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o) \\
\mathcal{H}_t &= o_t \circ \tanh(\mathcal{C}_t)
\end{aligned}
$$

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

ConvLSTM

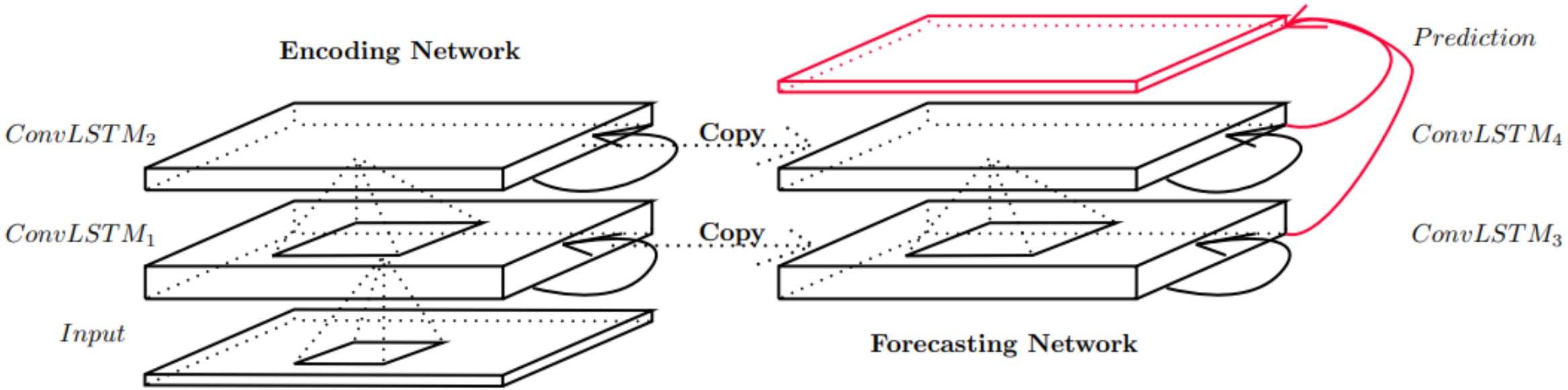


Figure 3: Encoding-forecasting ConvLSTM network for precipitation nowcasting

https://arxiv.org/pdf/1506.04214.pdf

Examples

Deep Chronnectome Learning via Full Bidirectional Long Short-Term Memory Networks for MCI Diagnosis

Weizheng Yan[1,2,3], Han Zhang[3], Jing Sui[1,2], and Dinggang Shen[3(✉)]

[1] Brainnetome Center and National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
Department of Radiology and BRIC, University of North Carolina of Chapel Hill, Chapel Hill, NC, USA
dgshen@med.unc.edu

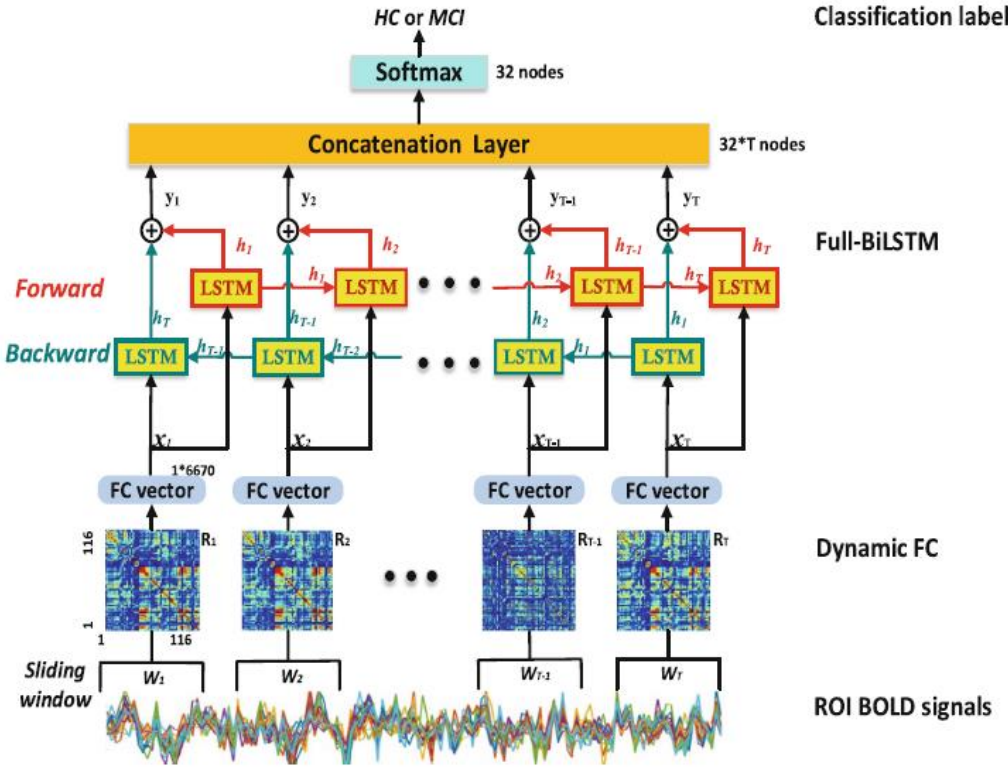


Fig. 1. Overview of the Full-BiLSTM for MCI classification.

Examples

An unsupervised long short-term memory neural network for event detection in cell videos

Ha Tran Hong Phan[a], Ashnil Kumar[a], David Feng[a,d], Michael Fulham[b,c], Jinman Kim[a]

[a] School of Information Technologies, The University of Sydney

[b] Sydney Medical School, The University of Sydney

[c] Department of Molecular Imaging, Royal Prince Alfred Hospital

[d] Med-X Research Institute, Shanghai Jiao Tong University

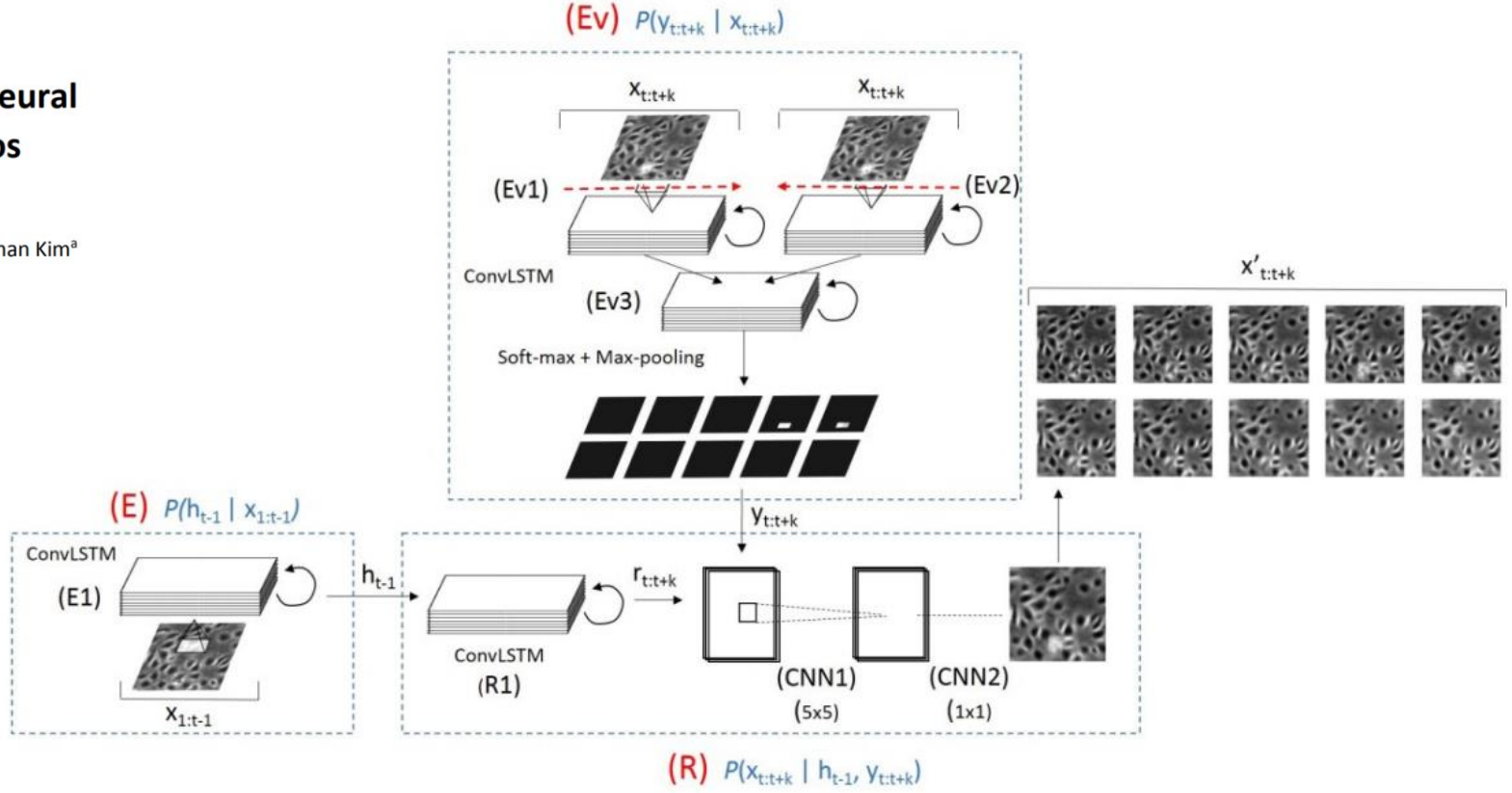


Figure 2: Outline of our ConvLSTM model.

Example

Deep Image-to-Image Recurrent Network with Shape Basis Learning for Automatic Vertebra Labeling in Large-Scale 3D CT Volumes

Dong Yang[1], Tao Xiong[2], Daguang Xu[3(✉)], S. Kevin Zhou[3], Zhoubing Xu[3], Mingqing Chen[3], JinHyeong Park[3], Sasa Grbic[3], Trac D. Tran[2], Sang Peter Chin[2], Dimitris Metaxas[1], and Dorin Comaniciu[3]

[1] Department of Computer Science, Rutgers University, Piscataway, NJ 08854, USA
[2] Department of Electrical and Computer Engineering, The Johns Hopkins University, Baltimore, MD 21218, USA
[3] Medical Imaging Technologies, Siemens Healthcare Technology Center, Princeton, NJ 08540, USA
{daguang.xu,shaohua.zhou,sasa.grbic, dorin.comaniciu}@siemens-healthineers.com

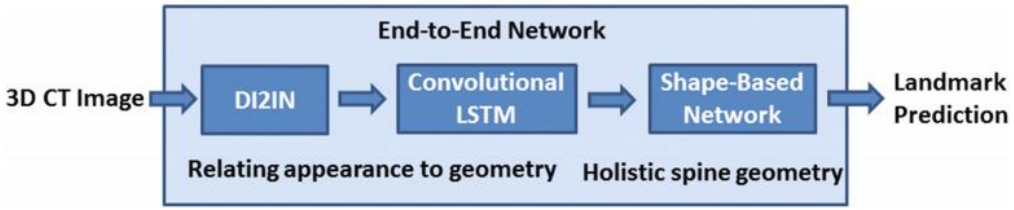


Fig. 1. Proposed method consisting of three major components: DI2IN, ConvLSTM and shape-based Network.

Improving Deep Pancreas Segmentation in CT and MRI Images via Recurrent Neural Contextual Learning and Direct Loss Function

Jinzheng Cai[1], Le Lu[3], Yuanpu Xie[1], Fuyong Xing[2], Lin Yang[1,2]

[1] Department of Biomedical Engineering,
[2] Department of Electrical and Computer Engineering,
University of Florida, Gainesville, FL 32611, USA
[3] Department of Radiology and Imaging Sciences,
National Institutes of Health Clinical Center, Bethesda, MD 20892, USA

Abstract. Deep neural networks have demonstrated very promising performance on accurate segmentation of challenging organs (*e.g.,* pancreas) in abdominal CT and MRI scans. The current deep learning approaches conduct pancreas segmentation by processing sequences of 2D image slices independently through deep, dense per-pixel masking for each image, without explicitly enforcing spatial consistency constraint on segmentation of successive slices. We propose a new convolutional/recurrent neural network architecture to address the contextual learning and segmentation consistency problem. A deep convolutional sub-network is first designed and pre-trained from scratch. The output layer of this network module is then connected to recurrent layers and can be fine-tuned for contextual learning, in an end-to-end manner. Our recurrent sub-network is a type of Long short-term memory (LSTM) network that performs segmentation on an image by integrating its neighboring slice segmentation predictions, in the form of a dependent sequence processing. Additionally, a novel segmentation-direct loss function (named Jaccard Loss) is proposed and deep networks are trained to optimize Jaccard Index (JI) directly. Extensive experiments are conducted to validate our proposed deep models, on quantitative pancreas segmentation using both CT and MRI scans. Our method outperforms the state-of-the-art work on CT [11] and MRI pancreas segmentation [1], respectively.

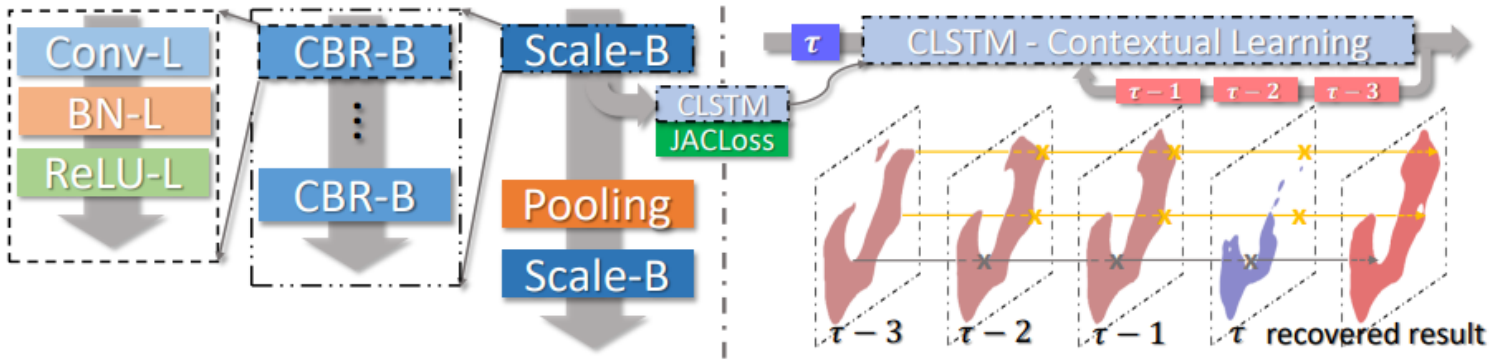


Fig. 1: **Network architecture:** Left is the CBR block (CBR-B) that contains convolutional layer (Conv-L), batch normalization layer (BN-L), and ReLU layer (ReLU-L). While, each scale block (Scale-B) has several CBR blocks and followed with a pooling layer. Right is the CLSTM for contextual learning. Segmented outcome at slice τ would be regularized by the results of slice $\tau - 3$, $\tau - 2$, and $\tau - 1$. For example, contextual learning is activated in regions with $\times$ markers, where sudden losses of pancreas areas occurs in slice τ comparing to consecutive slices.

Example

Recurrent Neural Networks for Semantic Instance Segmentation

Amaia Salvador[1], Míriam Bellver[2], Víctor Campos[2], Manel Baradad[1]
Ferran Marques[1] Jordi Torres[2] and Xavier Giro-i-Nieto[1]

[1]Universitat Politècnica de Catalunya [2]Barcelona Supercomputing Center

Abstract. We present a recurrent model for semantic instance segmentation that sequentially generates binary masks and their associated class probabilities for every object in an image. Our proposed system is trainable end-to-end from an input image to a sequence of labeled masks and, compared to methods relying on object proposals, does not require post-processing steps on its output. We study the suitability of our recurrent model on three different instance segmentation benchmarks, namely Pascal VOC 2012, CVPPP Plant Leaf Segmentation and Cityscapes. Further, we analyze the object sorting patterns generated by our model and observe that it learns to follow a consistent pattern, which correlates with the activations learned in the encoder part of our network.

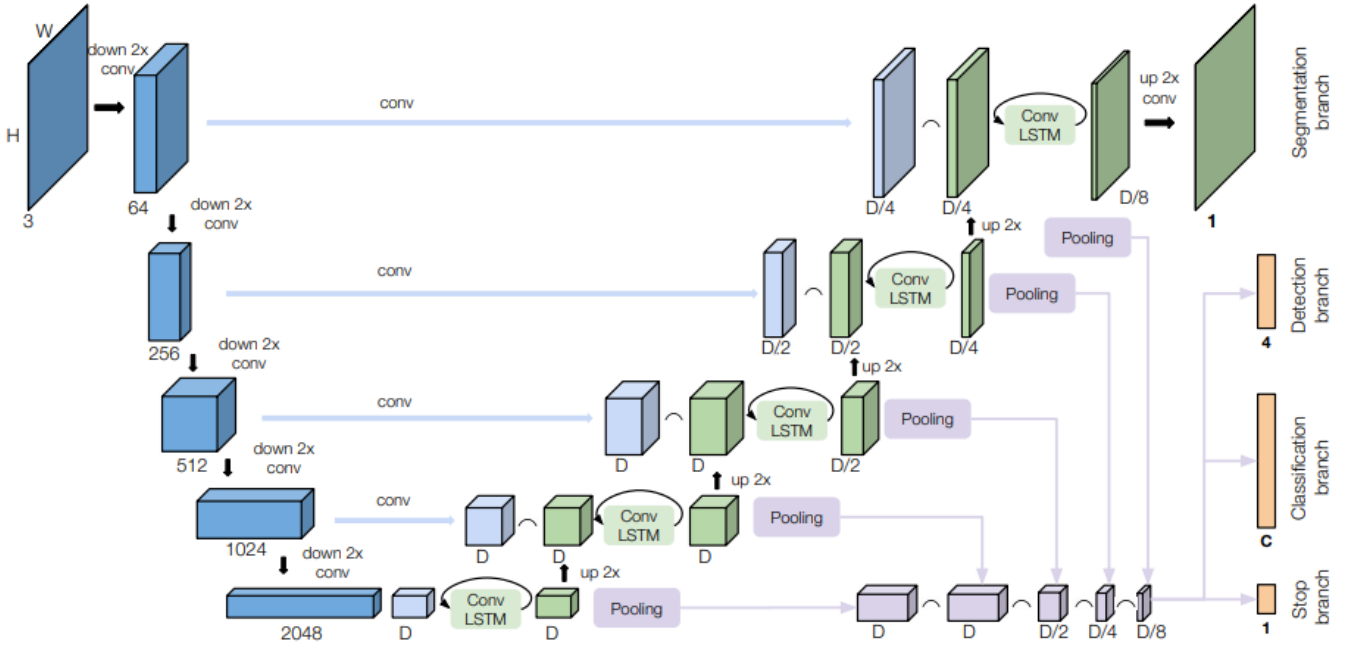


Fig. 1: Our proposed recurrent architecture for semantic instance segmentation.

Modeling 4D fMRI Data via Spatio-Temporal Convolutional Neural Networks (ST-CNN)

Yu Zhao[1], Xiang Li[2], Wei Zhang[1], Shijie Zhao[3], Milad Makkie[1], Mo Zhang[4], Quanzheng Li[2,4,5(✉)], and Tianming Liu[1(✉)]

[1] The University of Georgia, Athens, GA 30605, USA
tianming.liu@gmail.com
[2] MGH/BWH Center for Clinical Data Science, Boston, MA 02115, USA
li.quanzheng@mgh.harvard.edu
[3] Northwestern Polytechnical University, Xi'an 710072, Sha'anxi, China
[4] Peking University, Beijing 100080, China
[5] Laboratory for Biomedical Image Analysis, Beijing Institute of Big Data Research, Beijing 100871, China

Abstract. Simultaneous modeling of the spatio-temporal variation patterns of brain functional network from 4D fMRI data has been an important yet challenging problem for the field of cognitive neuroscience and medical image analysis. Inspired by the recent success in applying deep learning for functional brain decoding and encoding, in this work we propose a spatio-temporal convolutional neural network (ST-CNN) to jointly learn the spatial and temporal patterns of targeted network from the training data and perform automatic, pinpointing functional network identification. The proposed ST-CNN is evaluated by the task of identifying the Default Mode Network (DMN) from fMRI data. Results show that while the framework is only trained on one fMRI dataset, it has the sufficient generalizability to identify the DMN from different populations of data as well as different cognitive tasks. Further investigation into the results show that the superior performance of ST-CNN is driven by the jointly-learning scheme, which capture the intrinsic relationship between the spatial and temporal characteristic of DMN and ensures the accurate identification.

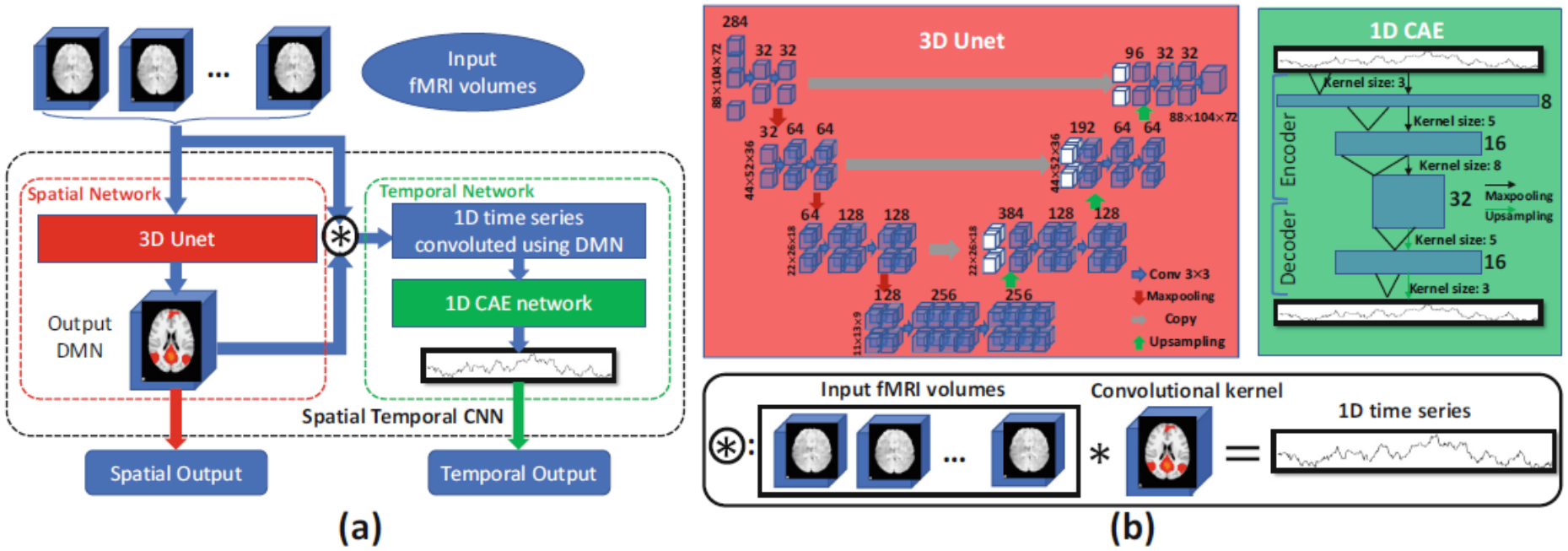


Fig. 1. (a). Algorithmic pipeline of ST-CNN; (b). Spatial network structure, temporal network structure, and the combination of the spatial and temporal domain.

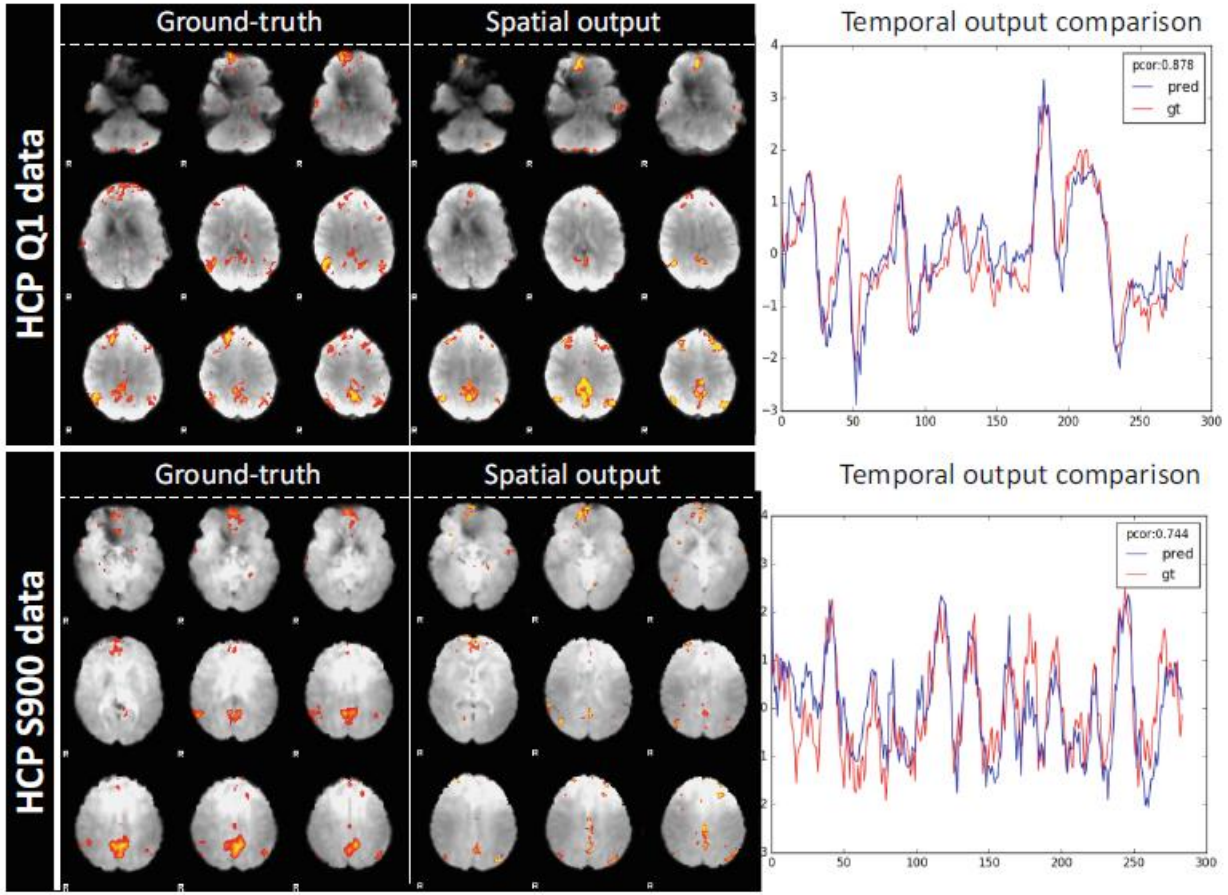


Fig. 3. Examples of comparisons between ST-CNN outputs and ground-truth from sparse representation. Here we showed 2 subjects' comparison results from two different datasets (1 HCP Q1 subjects and 1 HCP S900 subjects). Spatial maps are very similar and time series have Pearson correlation coefficient values 0.878 in HCP Q1 data, and 0.744 in HCP S900 data. Red curves are ground-truth. Blue curves are ST-CNN temporal outputs.