

Recurrent Neural Networks

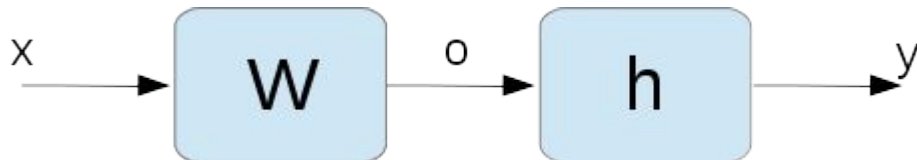
NN Reading Group

Marco Pedersoli & Thomas Lucas

Outline

- Motivation: why and when are RNN useful
- Formal definition of a RNN
- Backpropagation Through Time
- Vanishing Gradients
- Improved RNN
- Regularization for RNN
- Teacher Forcing Training

Introduction

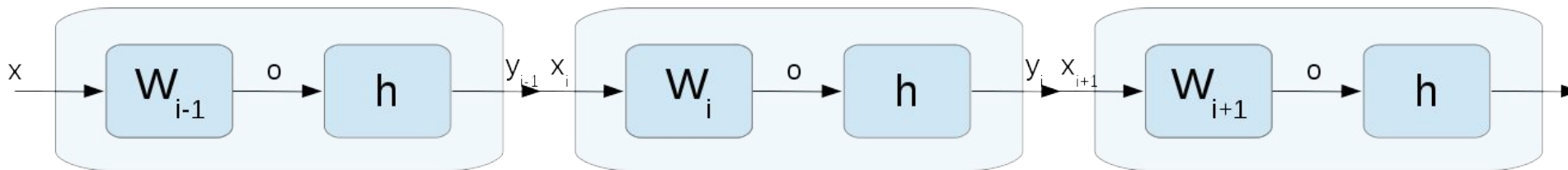


Building block: $y = h(Wx)$

W = linear (also convolutional, low rank, etc...)

h = element-wise non-linear operation (tanh, reLU, etc..)

Multilayer Network



Same input and output, but hidden representations: more powerful!

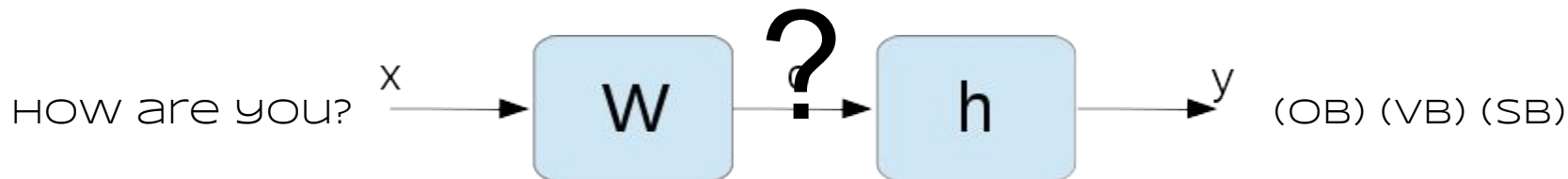
Composition of building blocks

Can still learn parameters with backpropagation

Sequential Input/Output

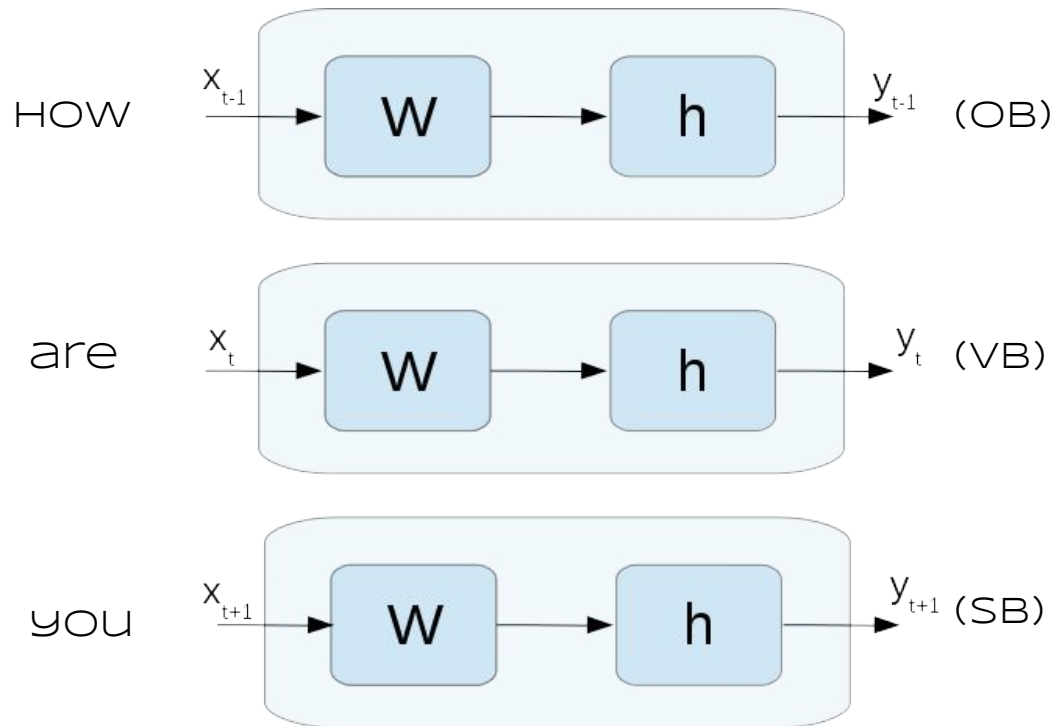
So far x, y fixed size vectors --> limiting factor!

We want input output with a complex structure eg. variable length sequence!



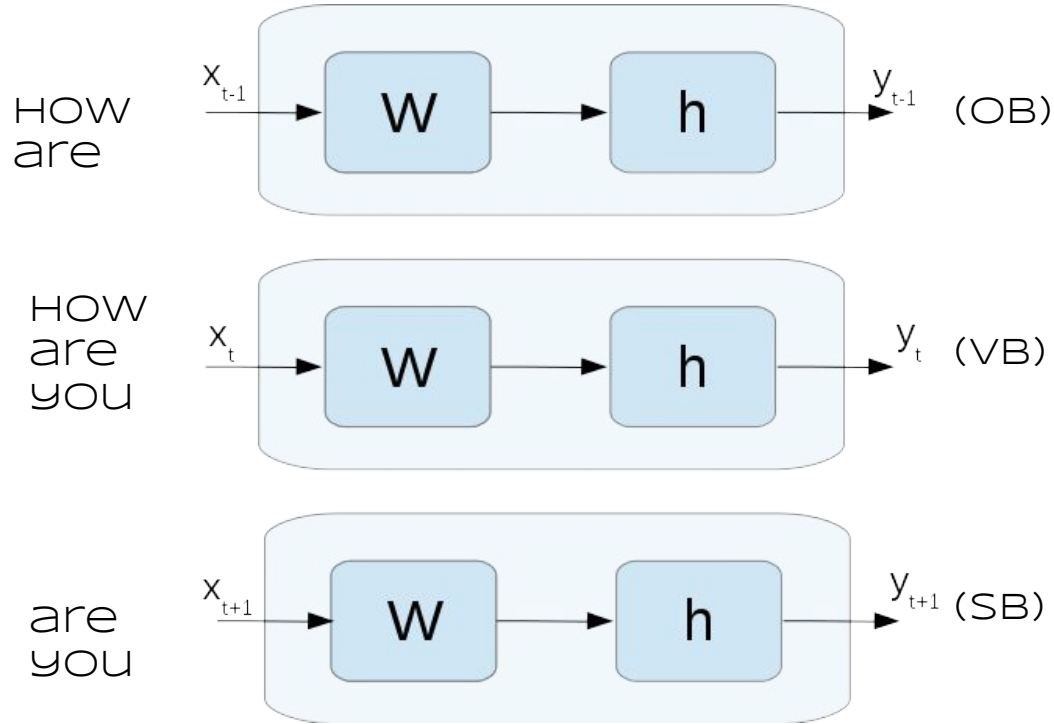
Can we reuse the same building block?

Repeated Network



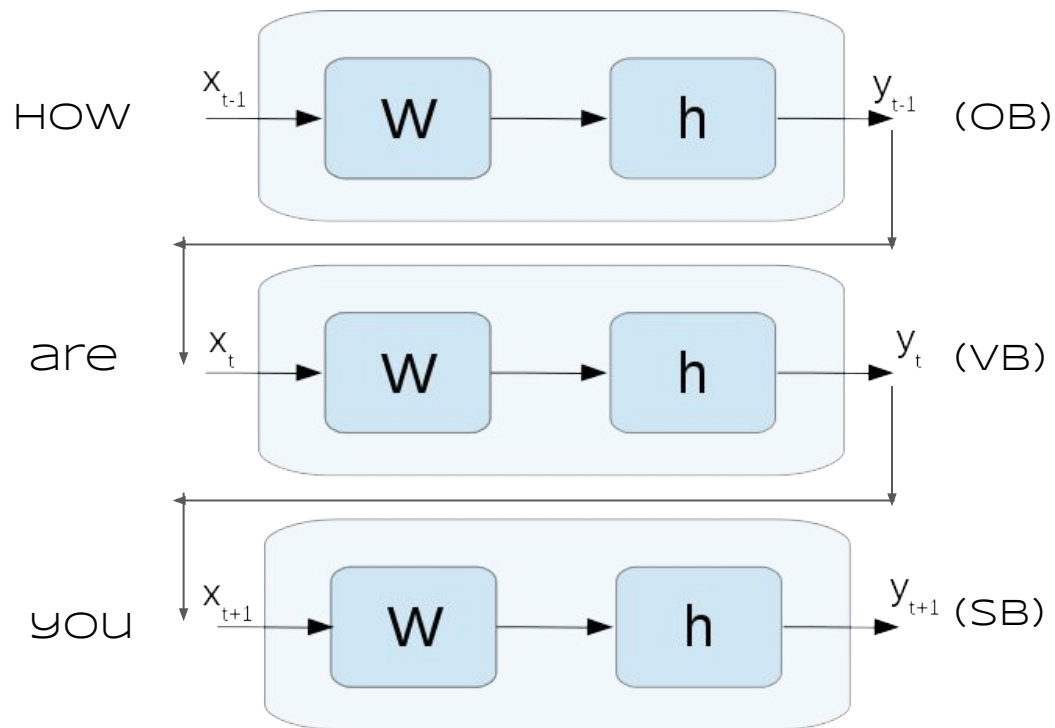
- + Works!
- + Scale well with data because reuses same parameters!
- No context, every decision is taken independently!

Convolutional Network



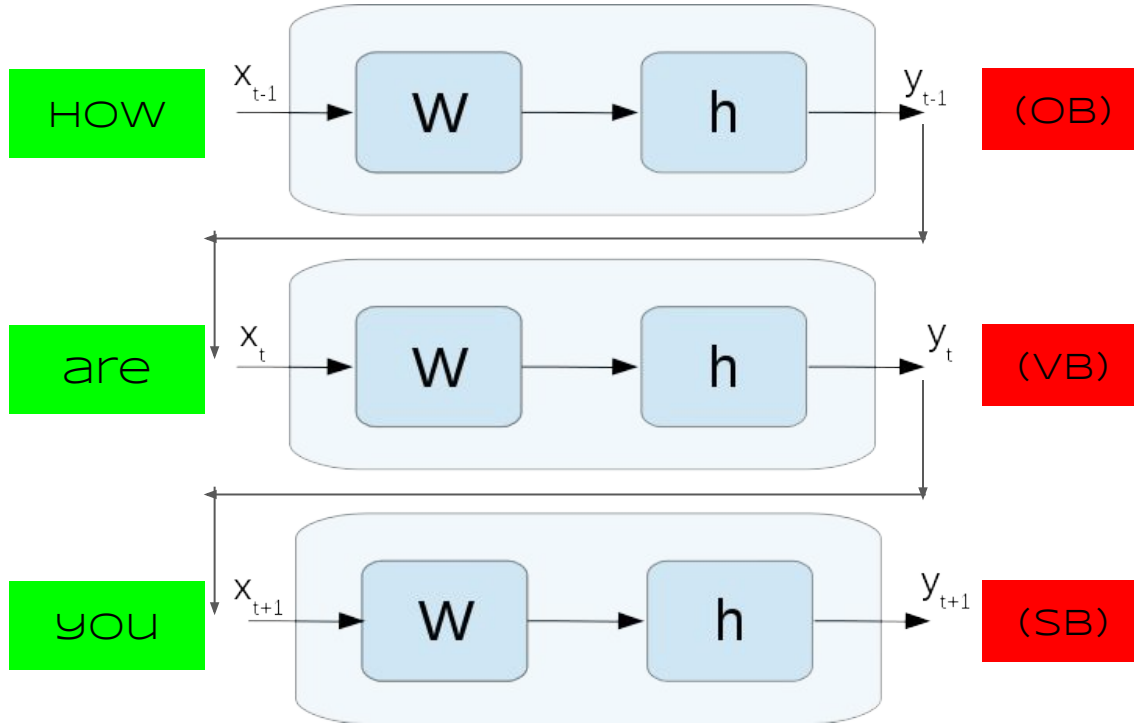
- + Works!
- + Scale well with data because same parameters reused!
- + Context, every decision depends also on the neighbours!
- Context has fixed, predefined structure!
- Does not scale if we want long range, sparse correlations!

Recurrent Network



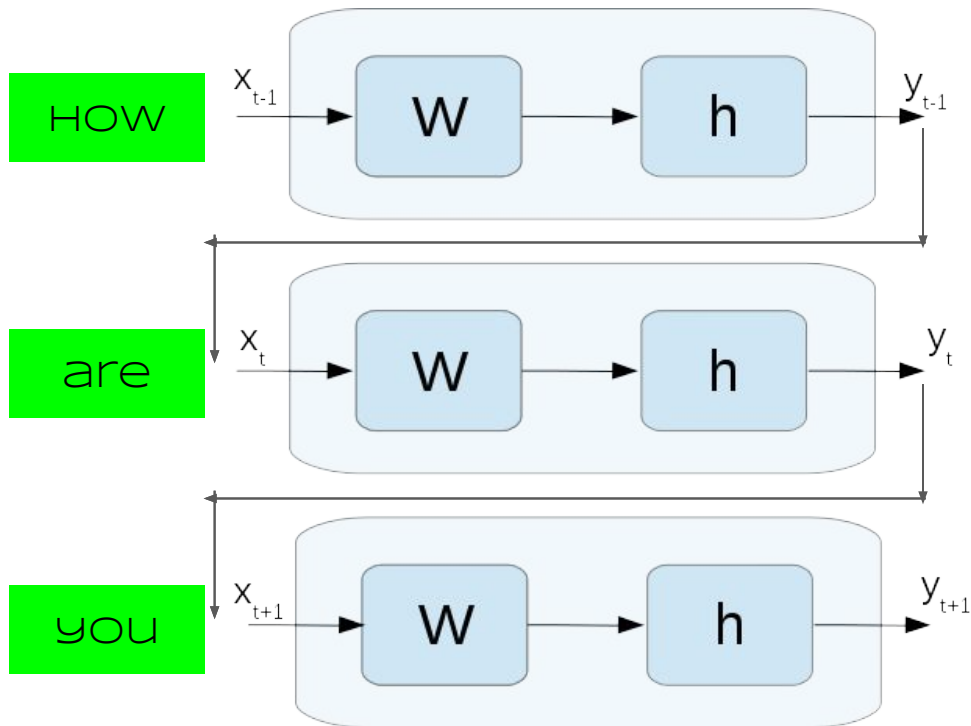
- + Works!
- + Scale well with data because same parameters reused!
- + State vector can summarize all the past, thus global context with long term dependencies!
- + Same optimization as before with Backpropagation

RNN topology: Many to Many (Coupled)



- + Works!
- + Scale well with data because same parameters reused!
- + State vector can summarize all the past, thus global context with long term dependencies!
- + Same optimization as before with Backpropagation

RNN topology: Many to one

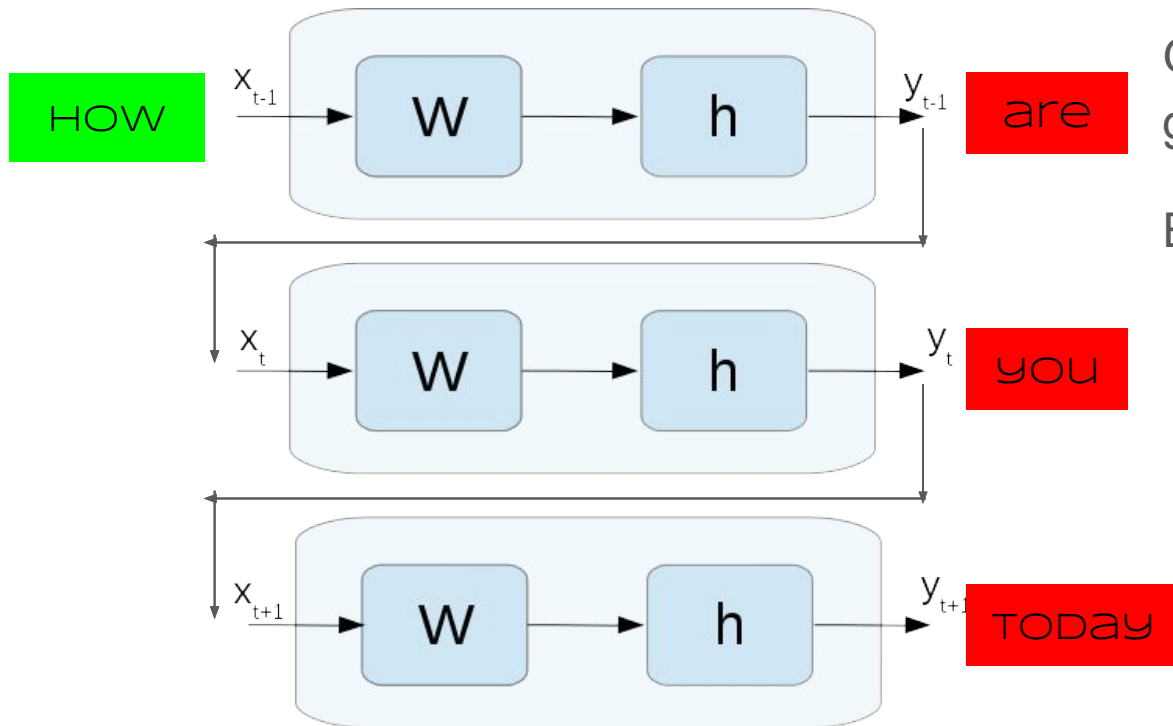


Classification problems with variable input size.

E.g.:

- Classify the category of a sentence
- Sentiment Classification on a audio track

RNN topology: One to many

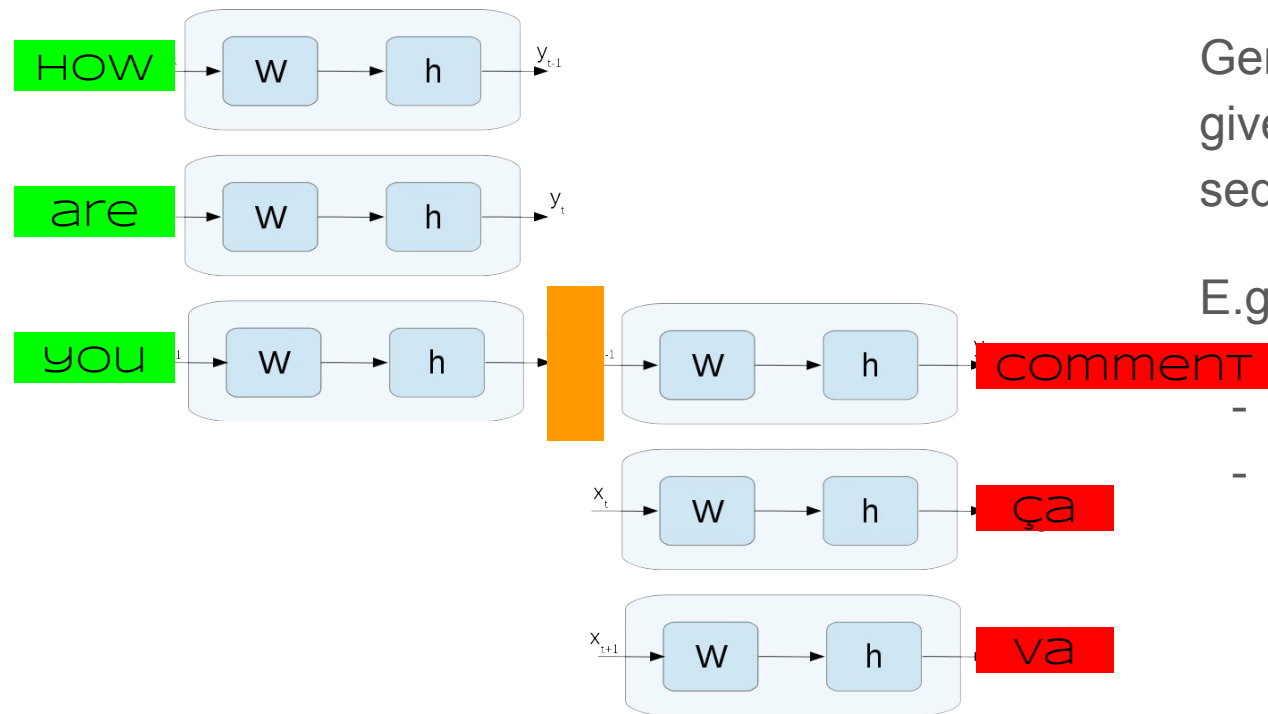


Generation of a sequence given an initial state.

E.g.:

- Generate a sentence describing an image
- Generate a sentence given the first word

RNN topology: Many to Many (Encoder/Decoder)

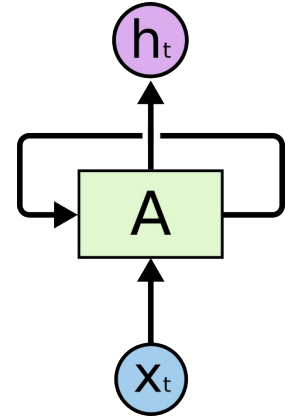
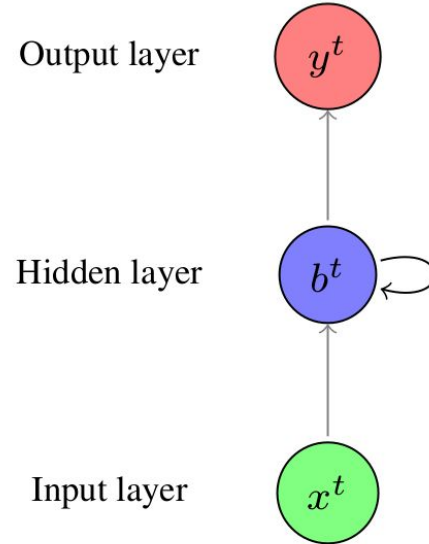
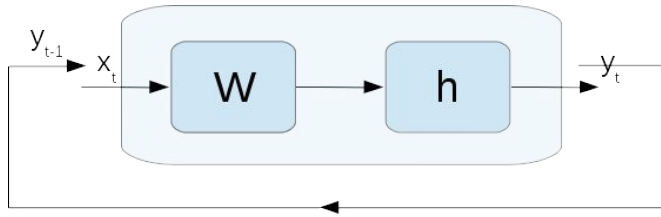


Generation of a sequence given a different length sequence.

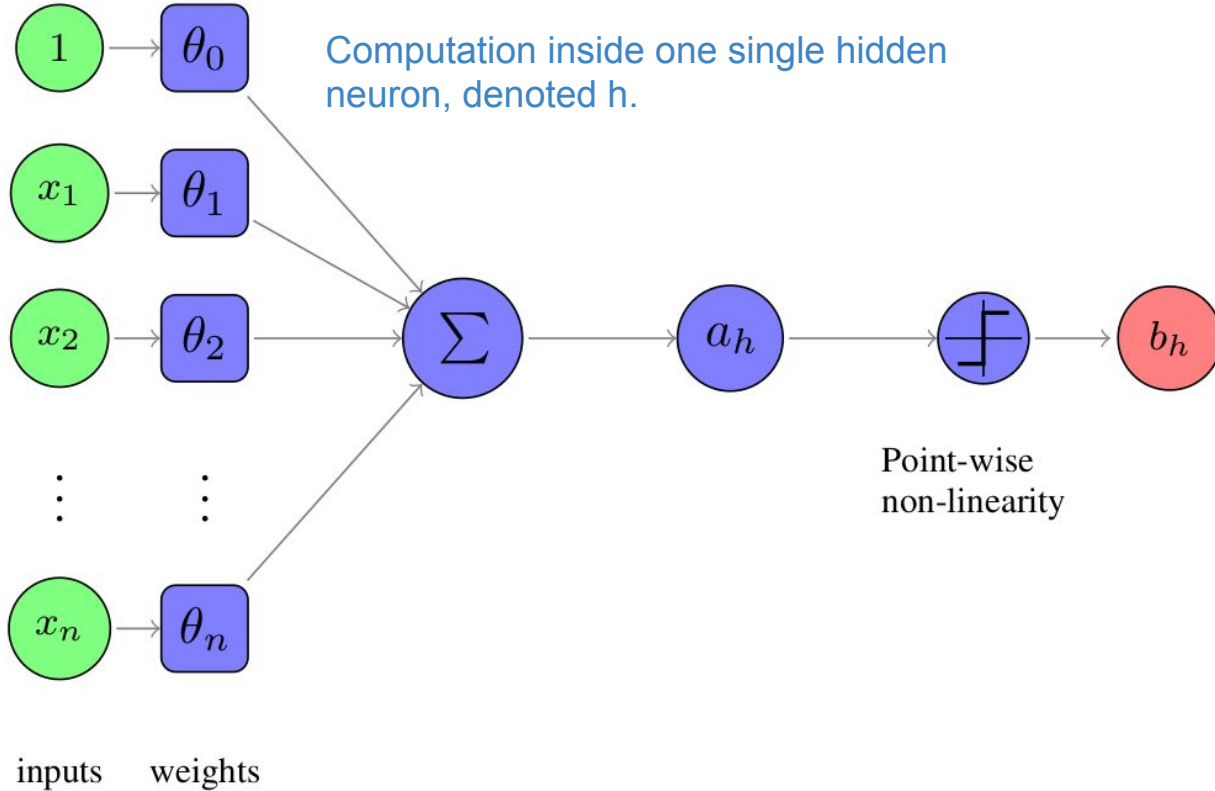
E.g.:

- Translate a sentence
- Describe a movie

RNN graphical representation



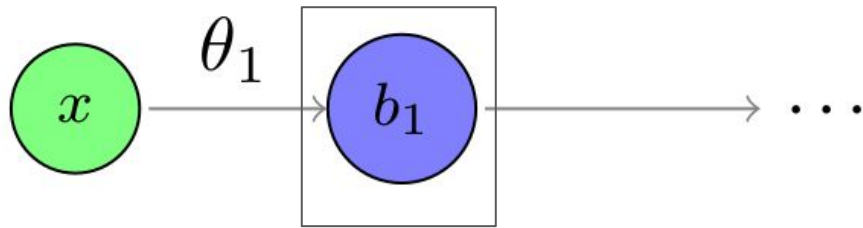
Forward propagation in a single neuron



$$a_h^t = \sum_{i=1}^I \theta_{ih} x_i^t$$

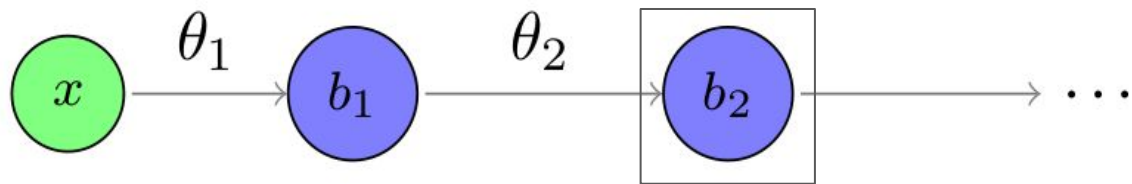
$$b_h^t = \sigma_h(a_h^t)$$

A network with one neuron per layer



$$b_1 = \sigma(\theta_1 x)$$

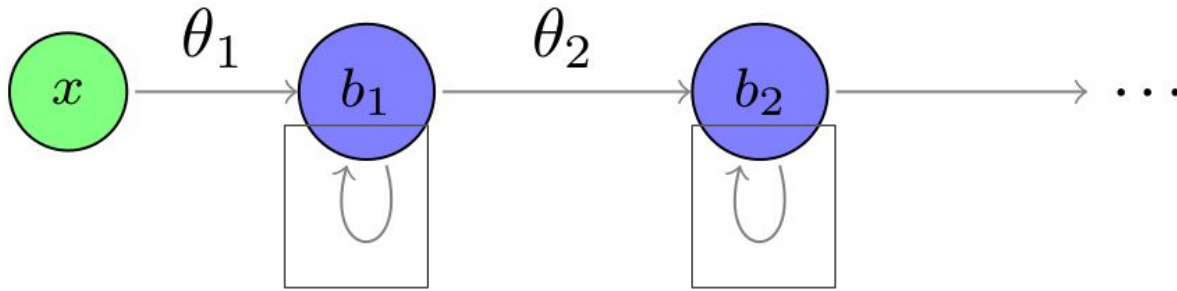
A network with one neuron per layer



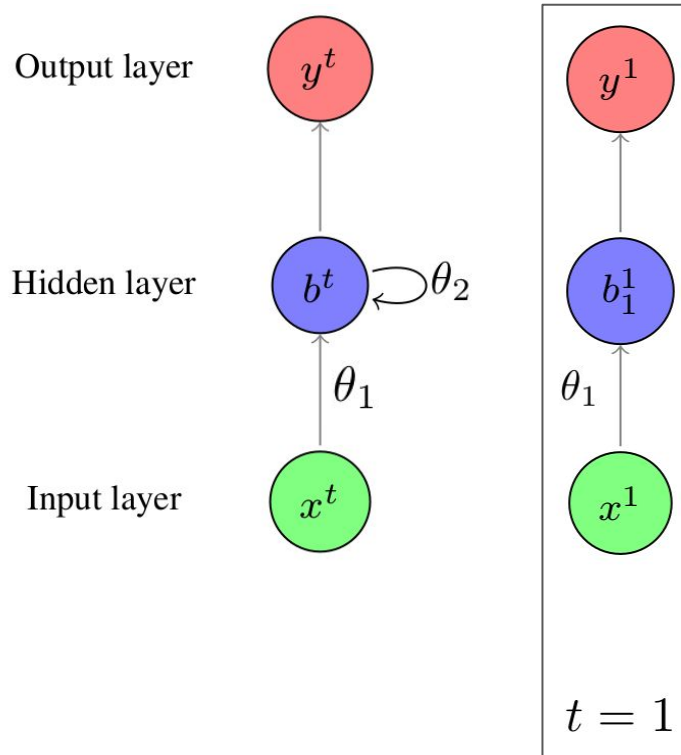
$$b_1 = \sigma(\theta_1 x)$$

$$\begin{aligned} b_2 &= \sigma(\theta_2 b_1) \\ &= \sigma(\theta_2 \sigma(\theta_1 x)) \end{aligned}$$

Making the network recurrent

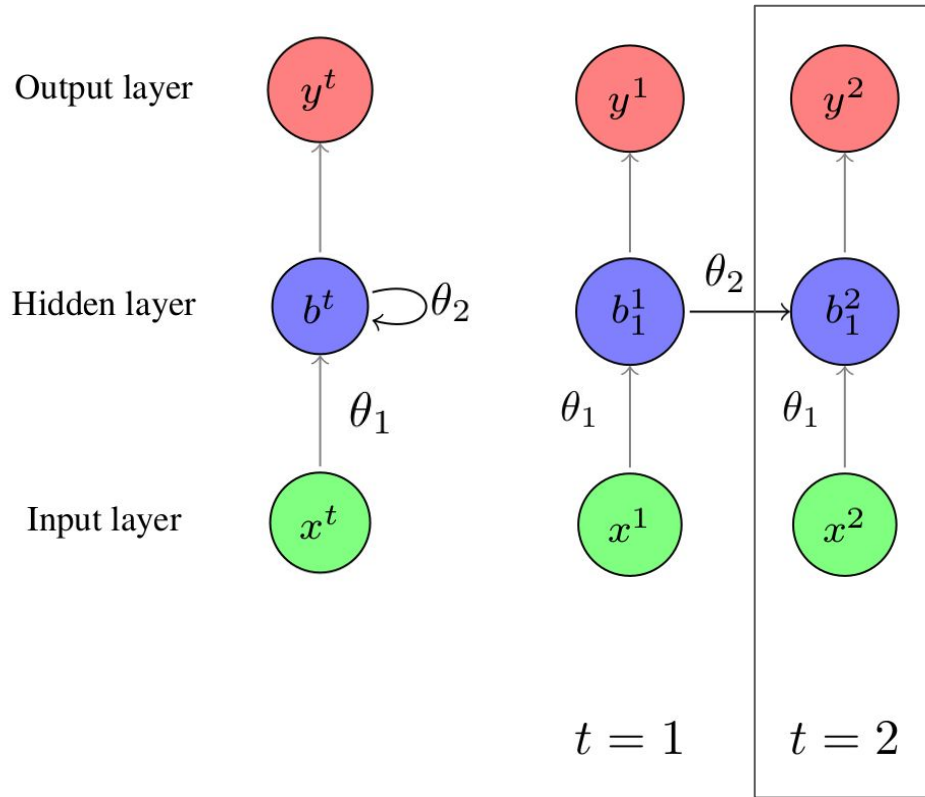


Unfolding the graph in time



$$b_1^1 = \sigma(\theta_1 x^1 + 0)$$

Unfolding the graph in time



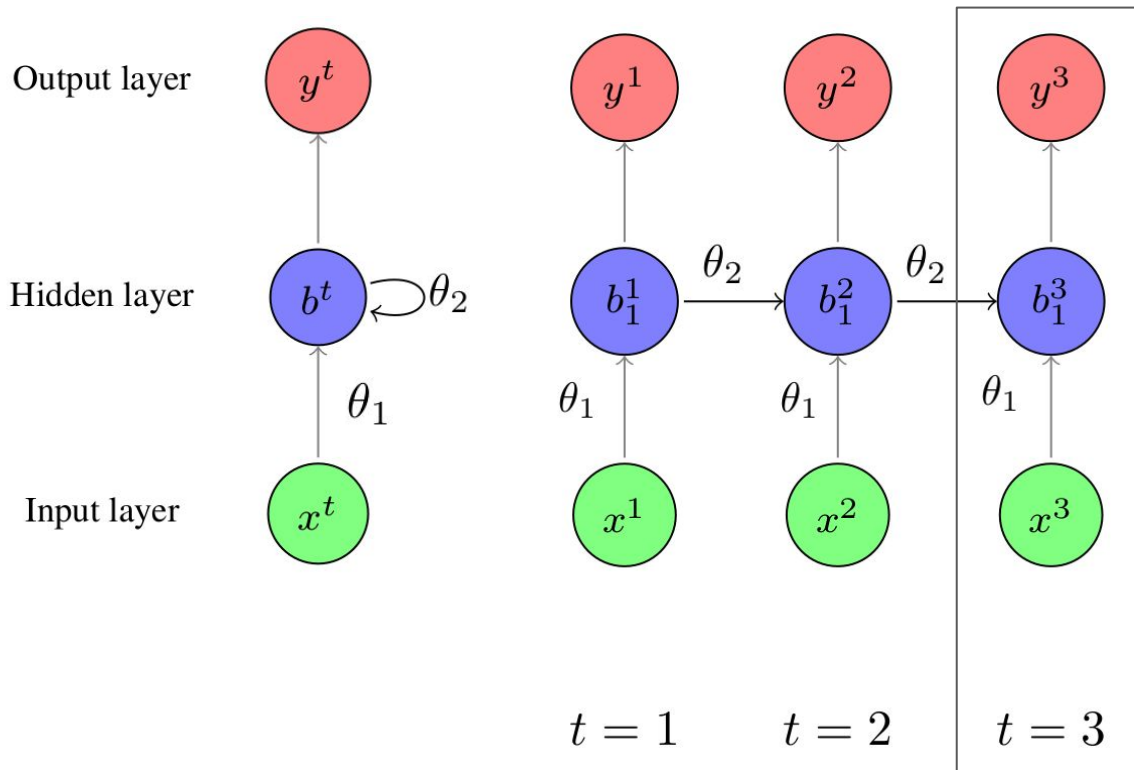
$$b_1^1 = \sigma(\theta_1 x^1 + 0)$$

$$a_1^2 = \theta_1 x^2 + \theta_2 b_1^1$$

$$b_1^2 = \sigma(a_1^2)$$

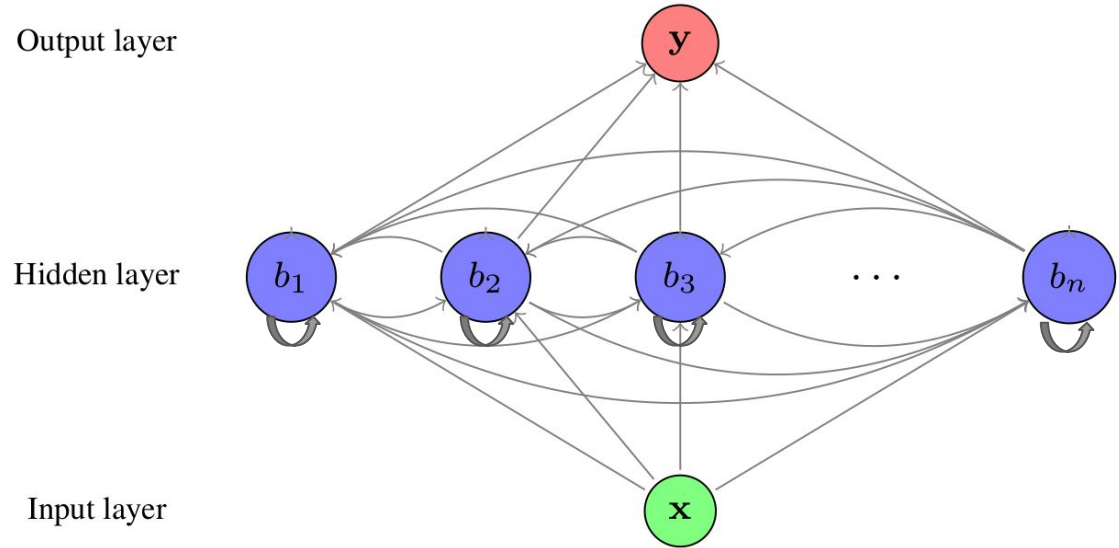
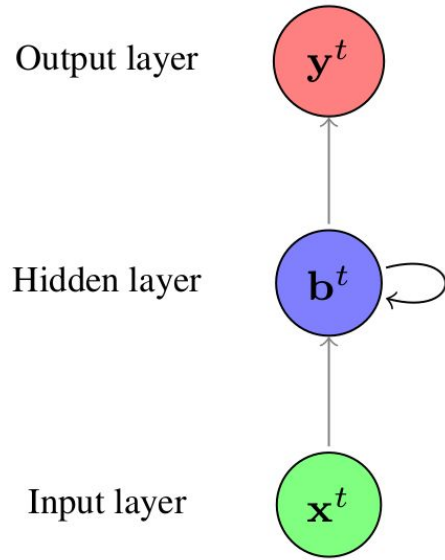
$$= \sigma(\theta_1 x^2 + \theta_2 b_1^1)$$

Unfolding the graph in time

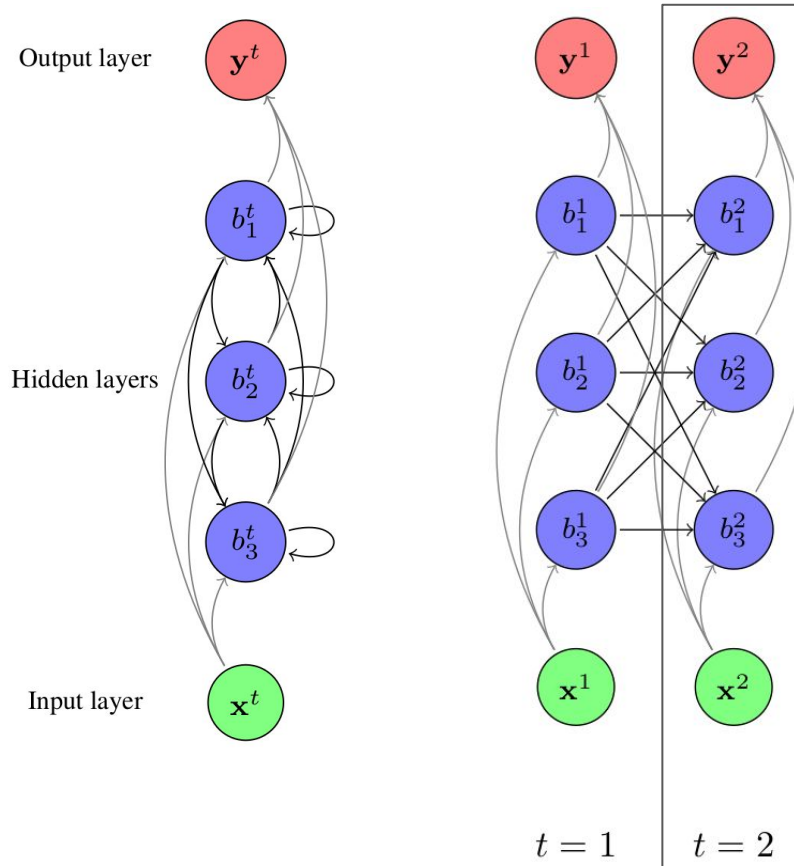


$$\begin{aligned} a_1^3 &= \theta_1 x^3 + \theta_2 b_1^2 \\ b_1^3 &= \sigma(a_1^3) \\ &= \sigma(\theta_1 x^3 + \theta_2 \sigma(\theta_1 x^2 + \theta_2 b_1^1)) \end{aligned}$$

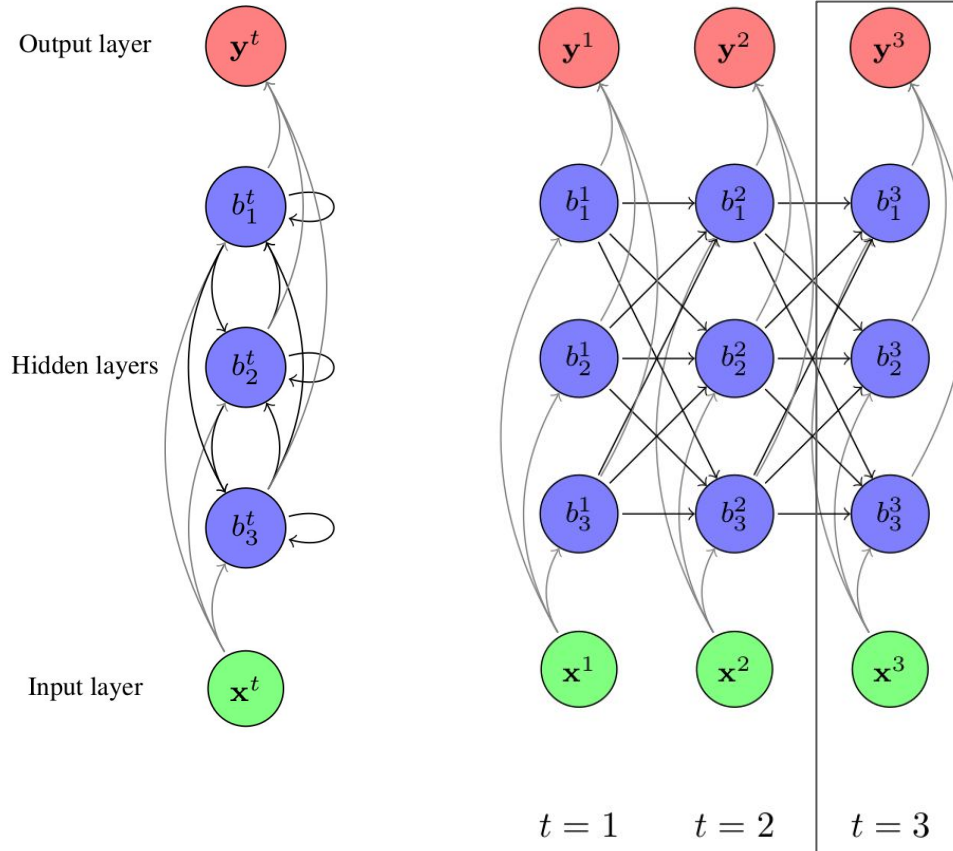
A recurrent network with one hidden layer



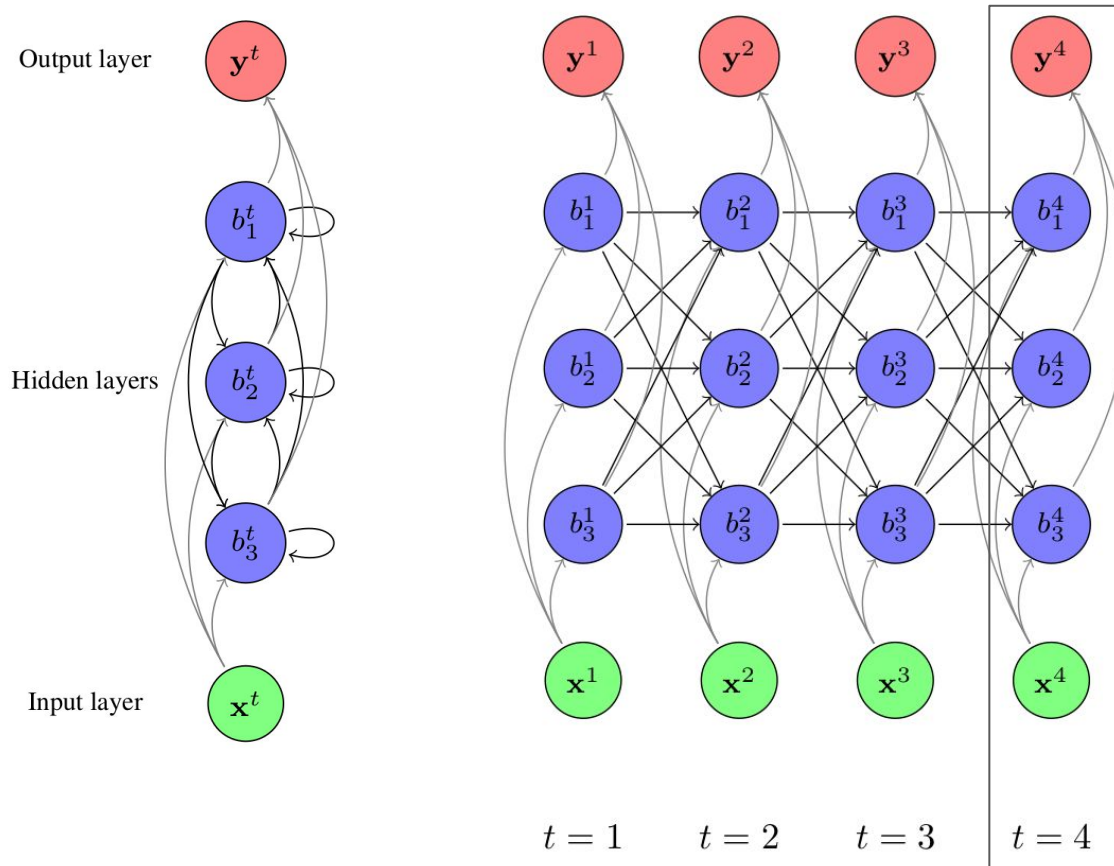
Unfolding the graph



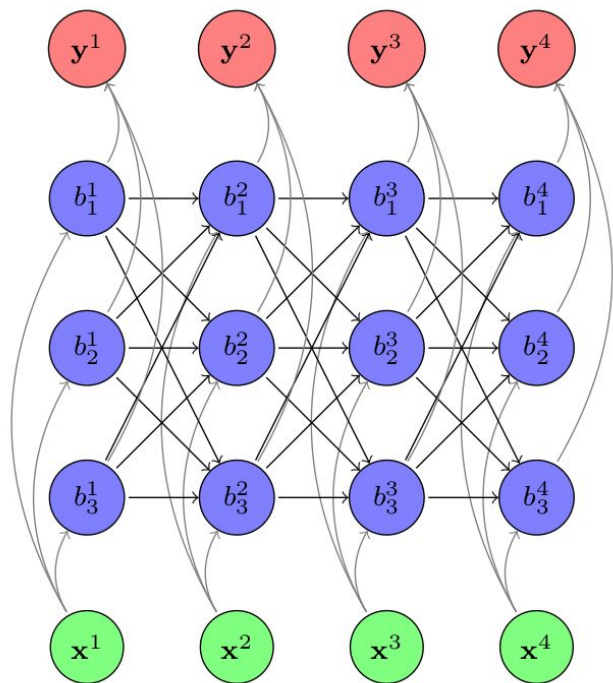
Unfolding the graph



Unfolding the graph



Equations of the recurrence



$t = 1$ $t = 2$ $t = 3$ $t = 4$

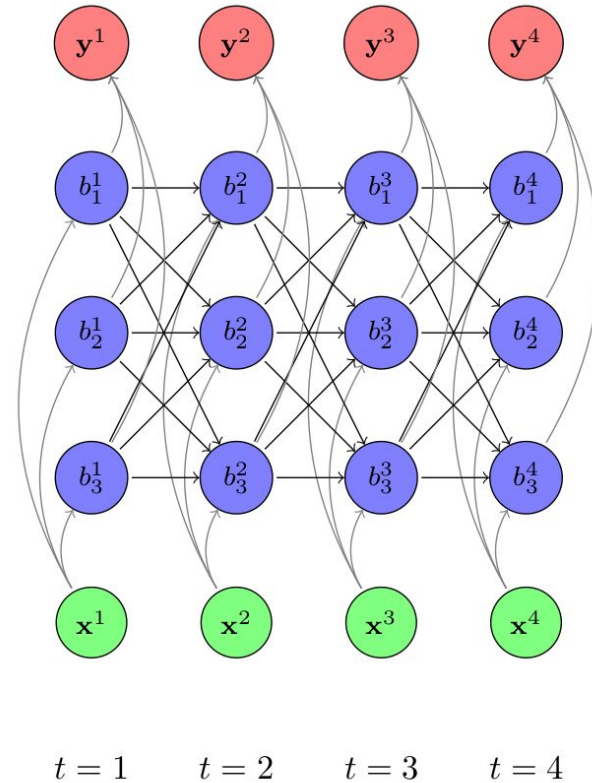
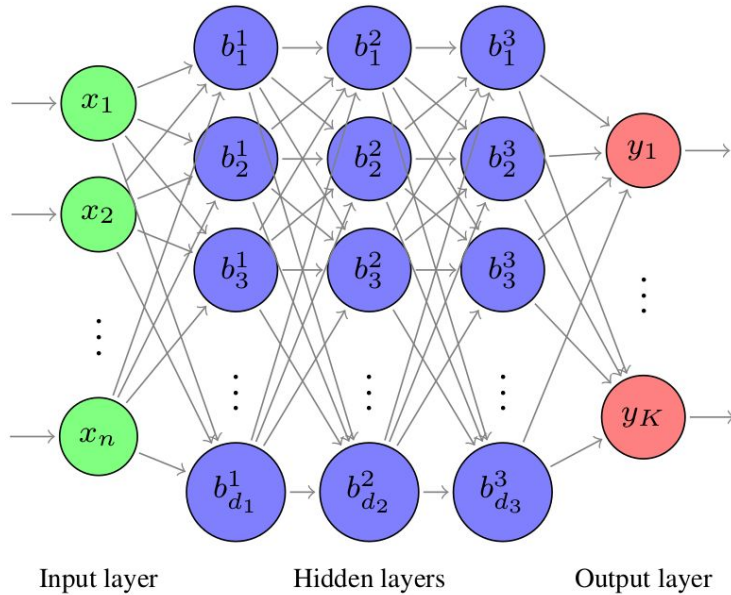
$$a_h^t = \sum_{i=1}^I \theta_{ih} x_i^t + \sum_{h'=1}^H \theta_{h'h} b_{h'}^{t-1}$$

$$b_h^t = \sigma_h(a_h^t)$$

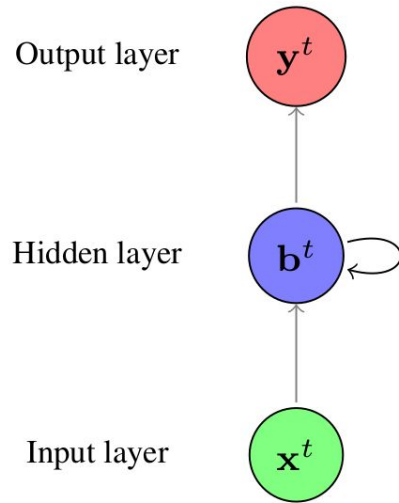
$$\mathbf{a}^t = \theta_{in}^T \mathbf{x} + \theta_{rec}^T \mathbf{a}^{t-1}$$

$$\mathbf{b}^t = \sigma(\mathbf{a}^t)$$

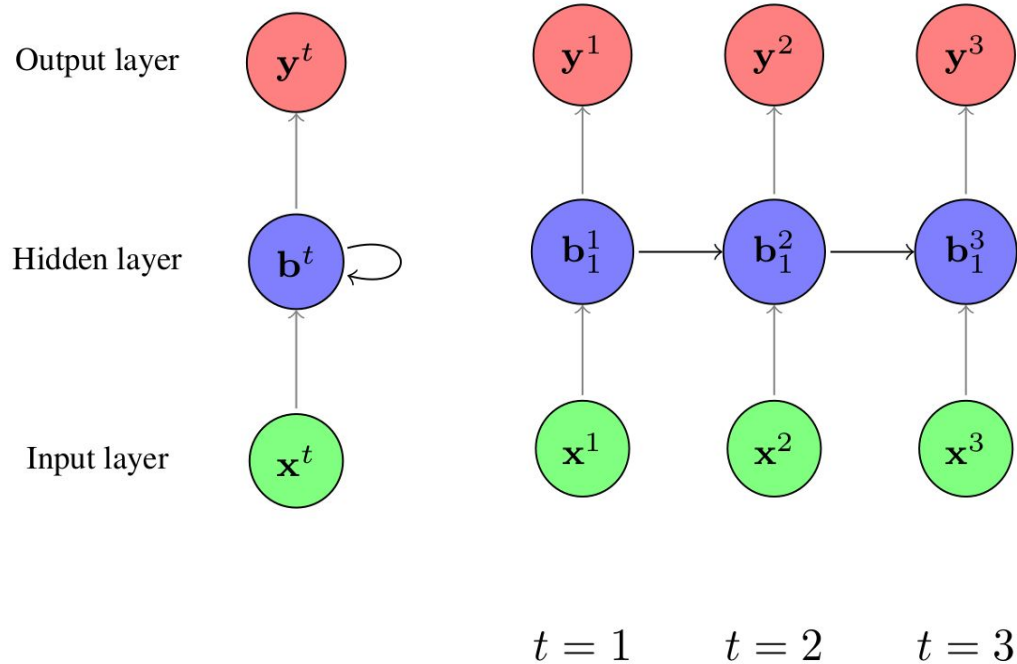
The unfolded graph is a feed-forward graph



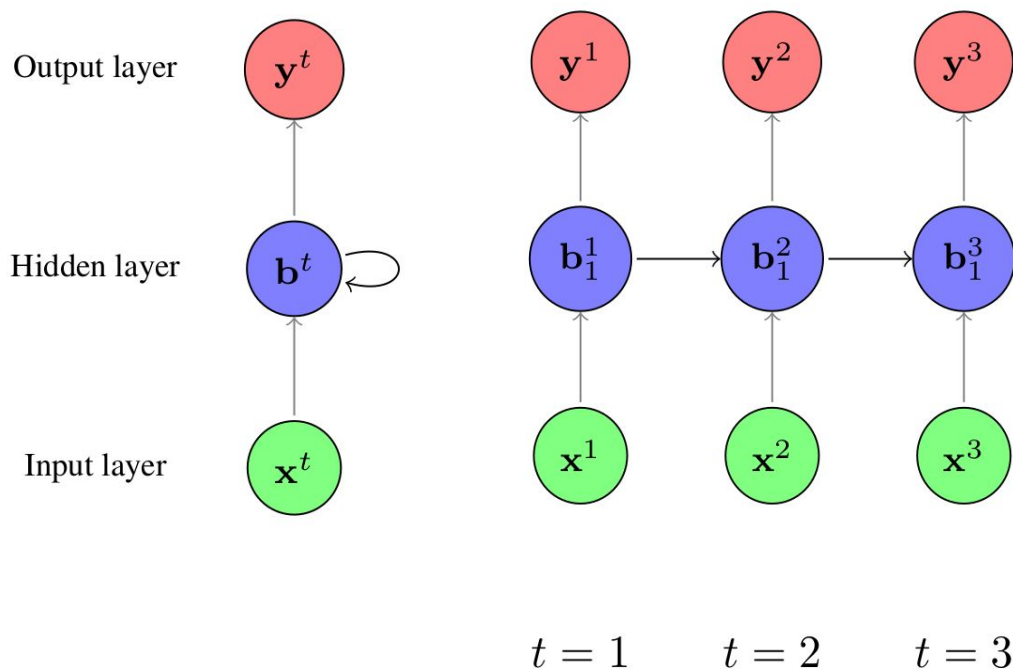
Recap



Recap



Unfolding the graph in time



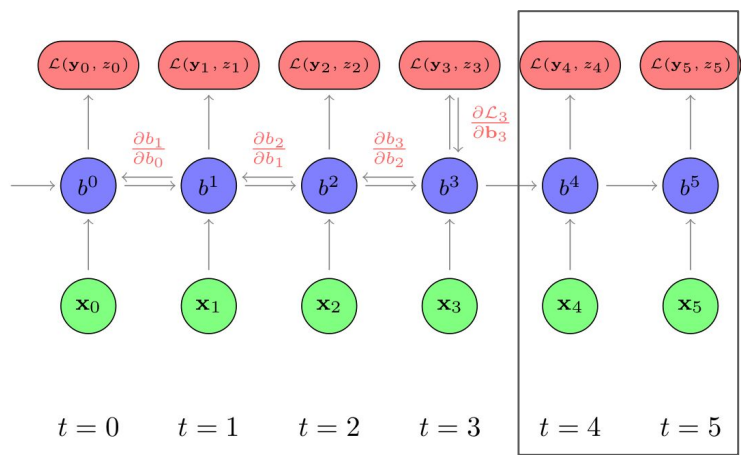
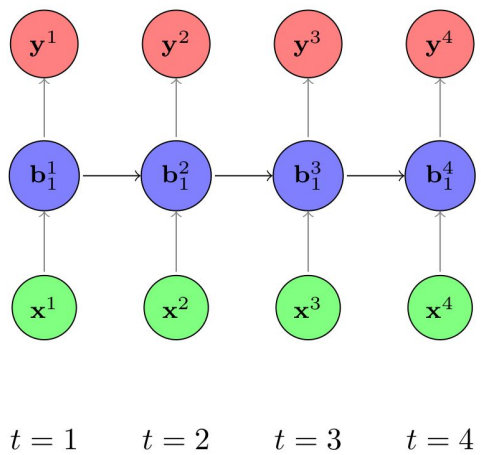
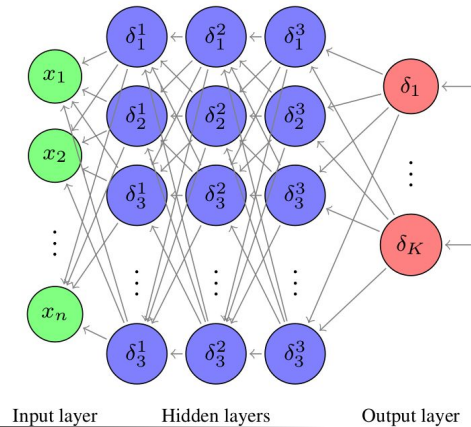
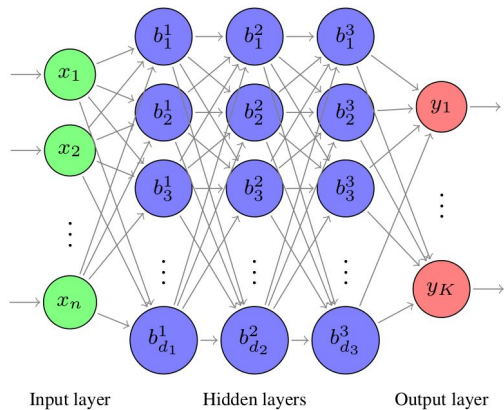
$$a_h^t = \sum_{i=1}^I \theta_{ih} x_i^t + \sum_{h'=1}^H \theta_{h'h} b_{h'}^{t-1}$$

$$b_h^t = \sigma_h(a_h^t)$$

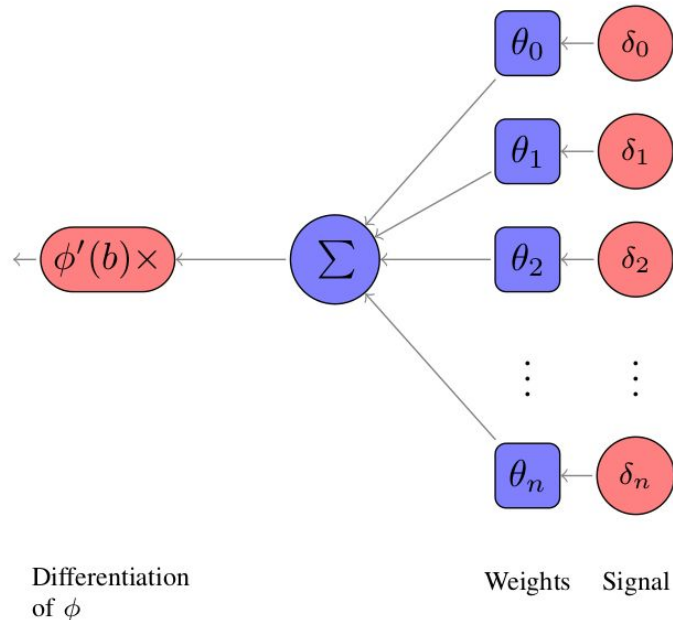
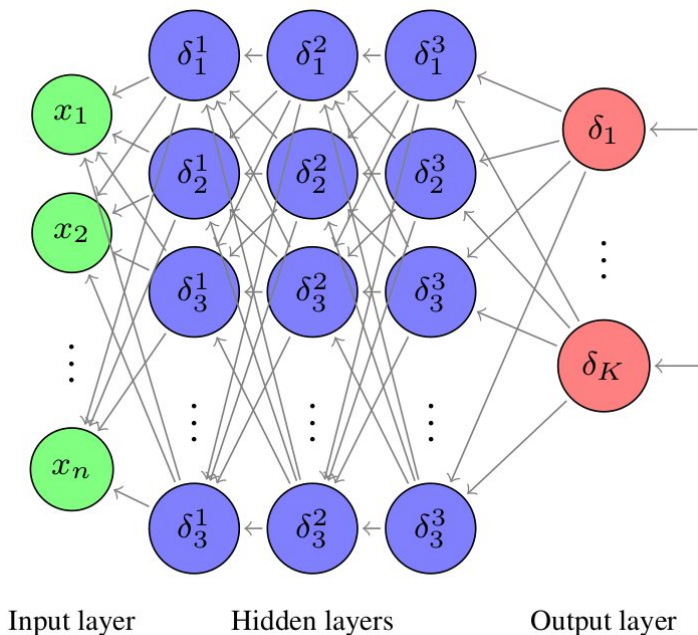
$$\mathbf{a}^t = \theta_{in}^T \mathbf{x} + \theta_{rec}^T \mathbf{a}^{t-1}$$

$$\mathbf{b}^t = \sigma(\mathbf{a}^t)$$

Back-propagation in a recurrent network



Equations of the back-propagation



$$\delta_j = \frac{\partial \mathcal{L}}{\partial a_j}$$

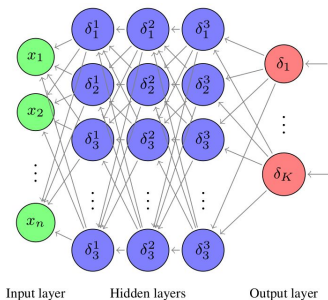
$$\delta_h = \sigma'(a_h) \sum_{k=1}^K \delta_k \theta_{hk}$$

Last hidden layer

$$\delta_h = \sigma'(a_h) \sum_{h' \in H^{l+1}} \delta_{h'} \theta_{hh'}$$

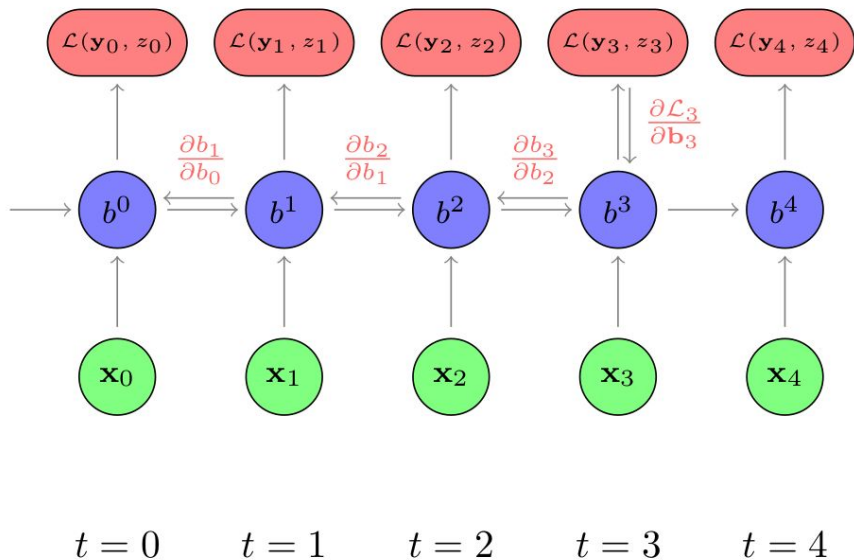
All other hidden layers

Equations of the back-propagation



$$\delta_h = \sigma'(a_h) \sum_{k=1}^K \delta_k \theta_{hk}$$

$$\delta_h = \sigma'(a_h) \sum_{h' \in H^{l+1}} \delta_{h'} \theta_{hh'}$$



$$\delta_j^t = \frac{\partial \mathcal{L}}{\partial a_j^t}$$

$$\delta_h^t = \sigma'(a_h^t) \left(\sum_{k=1}^K \delta_k^t \theta_{hk} + \sum_{h'=1}^H \delta_{h'}^{t+1} \theta_{hh'} \right)$$

$$\frac{\partial \mathcal{L}}{\partial \theta_{ij}} = \sum_{t=1}^T \frac{\partial \mathcal{L}}{\partial a_j^t} \frac{\partial a_j^t}{\partial \theta_{ij}} = \sum_{t=1}^T \delta_j^t b_i^t$$

The vanishing gradient problem

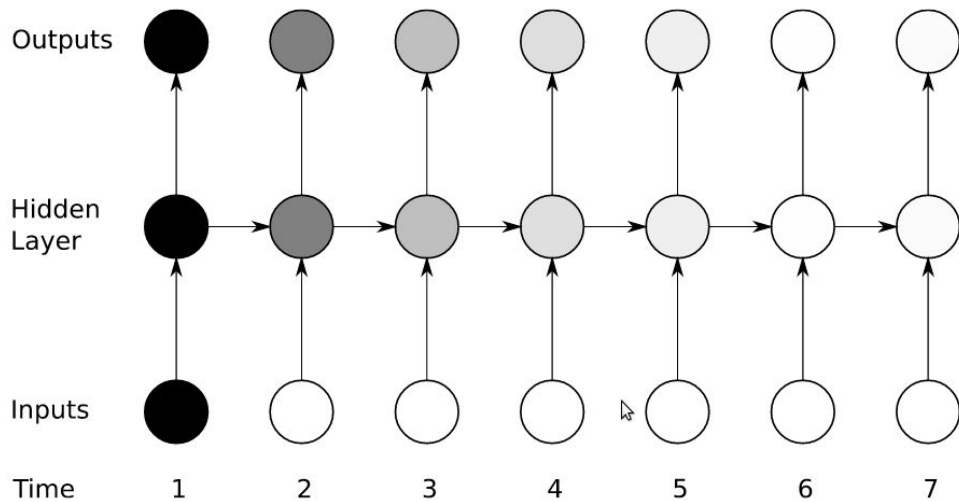
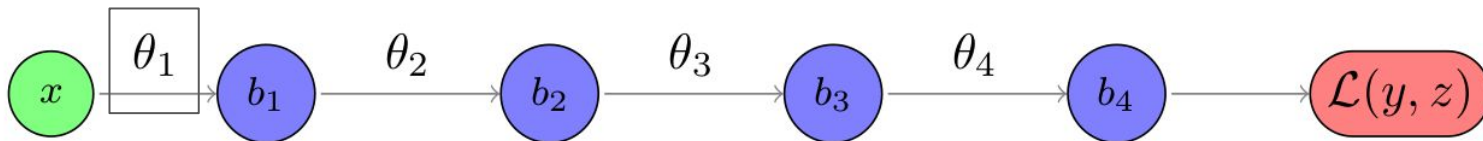


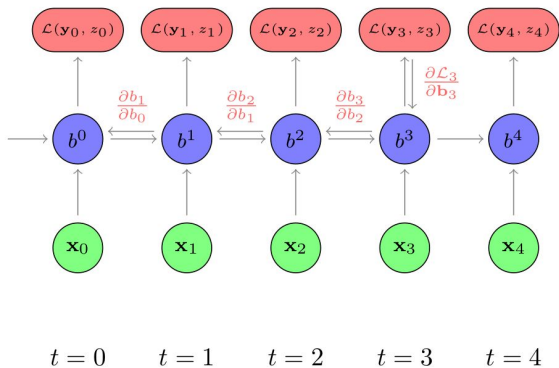
Figure credit: Alex Graves

A simple case of vanishing gradient



$$\frac{\partial \mathcal{L}}{\partial \theta_1} = \frac{\partial \mathcal{L}}{\partial h_4} \times \theta_4 \times \sigma'(h_3) \times \theta_3 \times \sigma'(h_2) \times \theta_2 \times \sigma'(h_1) \times x$$

A sufficient condition for gradients to vanish




$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{t=1}^T \frac{\partial \mathcal{L}_t}{\partial \theta}$$

$$\frac{\partial \mathcal{L}_t}{\partial \theta} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \theta}$$

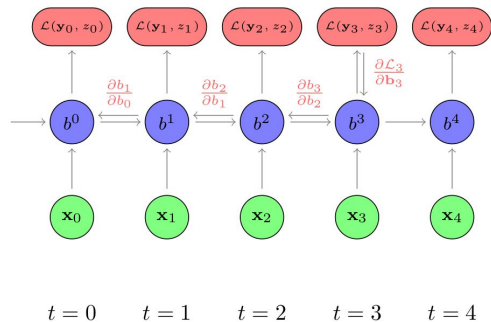
$$\frac{\partial \mathcal{L}_t}{\partial \theta} = \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{b}_t} \frac{\partial \mathbf{b}_t}{\partial \theta}$$

$$= \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{b}_t} \left(\sum_{k=1}^{t-1} \frac{\partial \mathbf{b}_t}{\partial \mathbf{b}_k} \frac{\partial \mathbf{b}_k}{\partial \theta} \right)$$

$$= \sum_{k=1}^{t-1} \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{b}_t} \frac{\partial \mathbf{b}_t}{\partial \mathbf{b}_k} \frac{\partial \mathbf{b}_k}{\partial \theta}$$


$$\frac{\partial \mathbf{b}_t}{\partial \theta} = \left(\sum_{k=1}^{t-1} \frac{\partial \mathbf{b}_t}{\partial \mathbf{b}_k} \frac{\partial \mathbf{b}_k}{\partial \theta} \right)$$

A sufficient condition for gradients to vanish



$$\begin{aligned}
 \frac{\partial \mathcal{L}_t}{\partial \theta} &= \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \theta} \\
 \frac{\partial \mathcal{L}_t}{\partial \theta} &= \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{b}_t} \frac{\partial \mathbf{b}_t}{\partial \theta} \\
 &= \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{b}_t} \left(\sum_{k=1}^{t-1} \frac{\partial \mathbf{b}_t}{\partial \mathbf{b}_k} \frac{\partial \mathbf{b}_k}{\partial \theta} \right) \\
 &= \sum_{k=1}^{t-1} \frac{\partial \mathcal{L}_t}{\partial \mathbf{y}_t} \frac{\partial \mathbf{y}_t}{\partial \mathbf{b}_t} \boxed{\frac{\partial \mathbf{b}_t}{\partial \mathbf{b}_k}} \frac{\partial \mathbf{b}_k}{\partial \theta} \\
 \boxed{\frac{\partial \mathbf{b}_t}{\partial \mathbf{b}_k}} &= \prod_{i=k+1}^t \frac{\partial \mathbf{b}_i}{\partial \mathbf{b}_{i-1}}
 \end{aligned}$$

We have the following sufficient conditions: If there exists $\beta < 1$ such that for all i , $\frac{\partial \mathbf{b}_i}{\partial \mathbf{b}_{i-1}} \leq \beta$ then $\frac{\partial \mathbf{b}_t}{\partial \mathbf{b}_k}$ vanishes exponentially with the number of time steps. If there exists $\gamma > 1$ such that for all i , $\frac{\partial \mathbf{b}_i}{\partial \mathbf{b}_{i-1}} \geq \gamma$ then $\frac{\partial \mathbf{b}_t}{\partial \mathbf{b}_k}$ explodes exponentially with the number of time steps.

A sufficient condition for gradients to vanish

$$\boxed{\frac{\partial \mathbf{b}_t}{\partial \mathbf{b}_k}}$$

$$= \prod_{i=k+1}^t \frac{\partial \mathbf{b}_i}{\partial \mathbf{b}_{i-1}}$$

$$\mathbf{b}_i = \phi(\theta_{hid} \mathbf{b}_{i-1} + \theta_{in} x_i)$$

$$\frac{\partial \mathbf{b}_i}{\partial \mathbf{b}_{i-1}} = \theta_{hid}^\top \text{diag}(\phi'(\mathbf{b}_{i-1}))$$

$$\left\| \frac{\partial \mathbf{b}_i}{\partial \mathbf{h}_{i-1}} \right\|_{L_2} \leq \|\theta_{hid}^\top\|_{L_2} \times \|\text{diag}(\phi'(\mathbf{h}_k))\|_{L_2}$$

Therefore there exists $\beta < 1$ such that:

$$\left\| \frac{\partial \mathbf{b}_t}{\partial \mathbf{b}_k} \right\|_{L_2} < \beta^{t-k}$$

A sufficient condition for gradients to vanish

Conclusion: ϕ' is upper bounded by α and $\|\theta_{hid}\|_{L_2} < \frac{1}{\alpha}$ is sufficient for $\left\| \frac{\partial \mathbf{b}_i}{\partial \mathbf{h}_{i-1}} \right\|_{L_2}$ to vanish exponentially fast.

The contraposition of this property is: if ϕ' is upper bounded by α and gradient explosion is observed then $\|\theta_{hid}\|_{L_2}$ is greater than $\frac{1}{\alpha}$

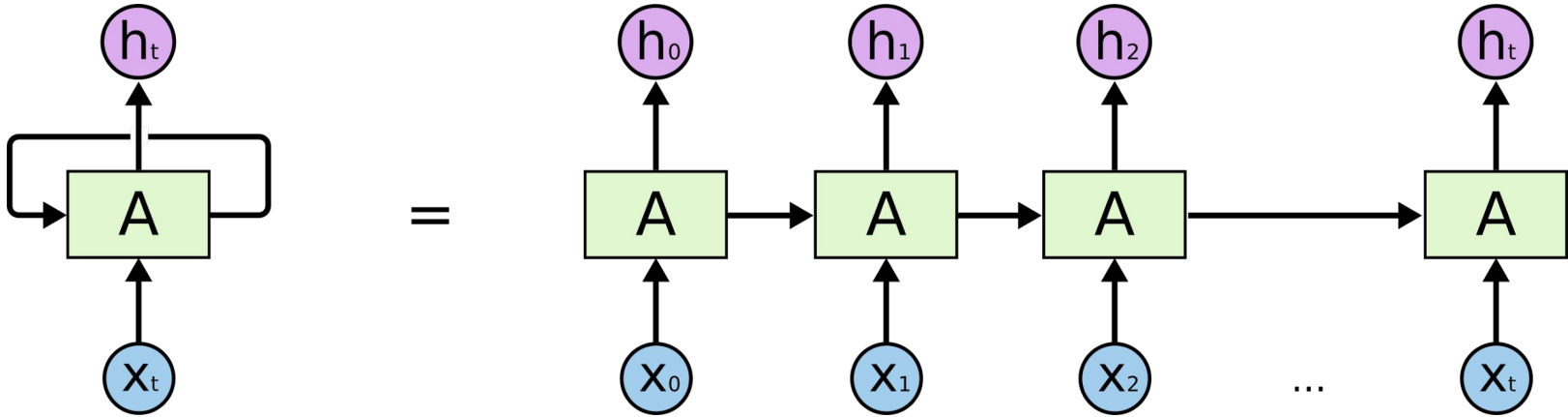
We cannot proceed similarly as with equations (1) to get a simple sufficient condition on θ_{hid} for $\left\| \frac{\partial \mathbf{b}_i}{\partial \mathbf{h}_{i-1}} \right\|_{L_2}$ to explode: although $\|\theta_{hid}\|_{L_2}$ is lower-bounded by $\lambda_{min}(\theta_{hid})$, $\phi'(x) \rightarrow 0$ when $x \rightarrow \infty$ and so the product of operator norms can not be lower-bounded without further hypothesis. Therefore we simply keep the result that $\left\| \frac{\partial \mathbf{b}_i}{\partial \mathbf{h}_{i-1}} \right\|_{L_2} > 1$ is sufficient for the gradient to explode. In practice this can happen, as the previous intuitive example shows.

Improved RNNs

- Gated Units
 - Long-Short Term Memory RNN
 - Gated Recurrent Unit
- Skip connections
 - Clockwork RNN
 - Hierarchical Multi-resolution RNN

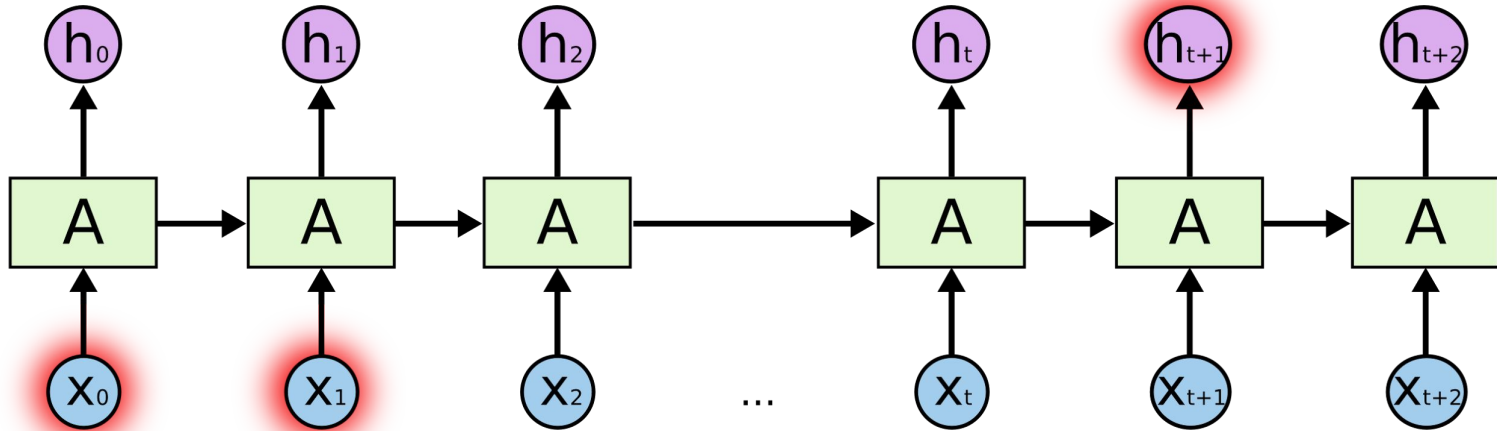
LSTM recurrent networks

[based on Christopher Olah blog]

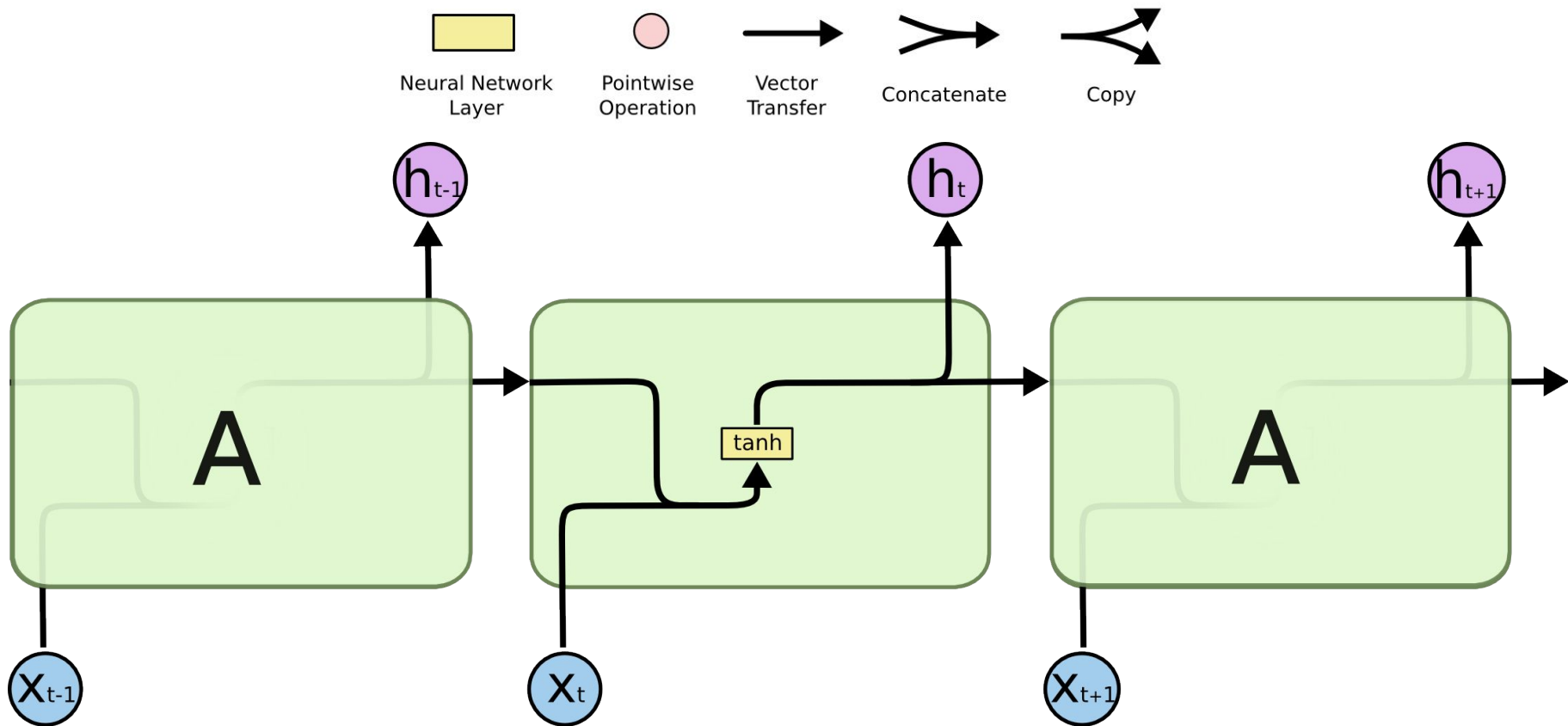


LSTM recurrent networks

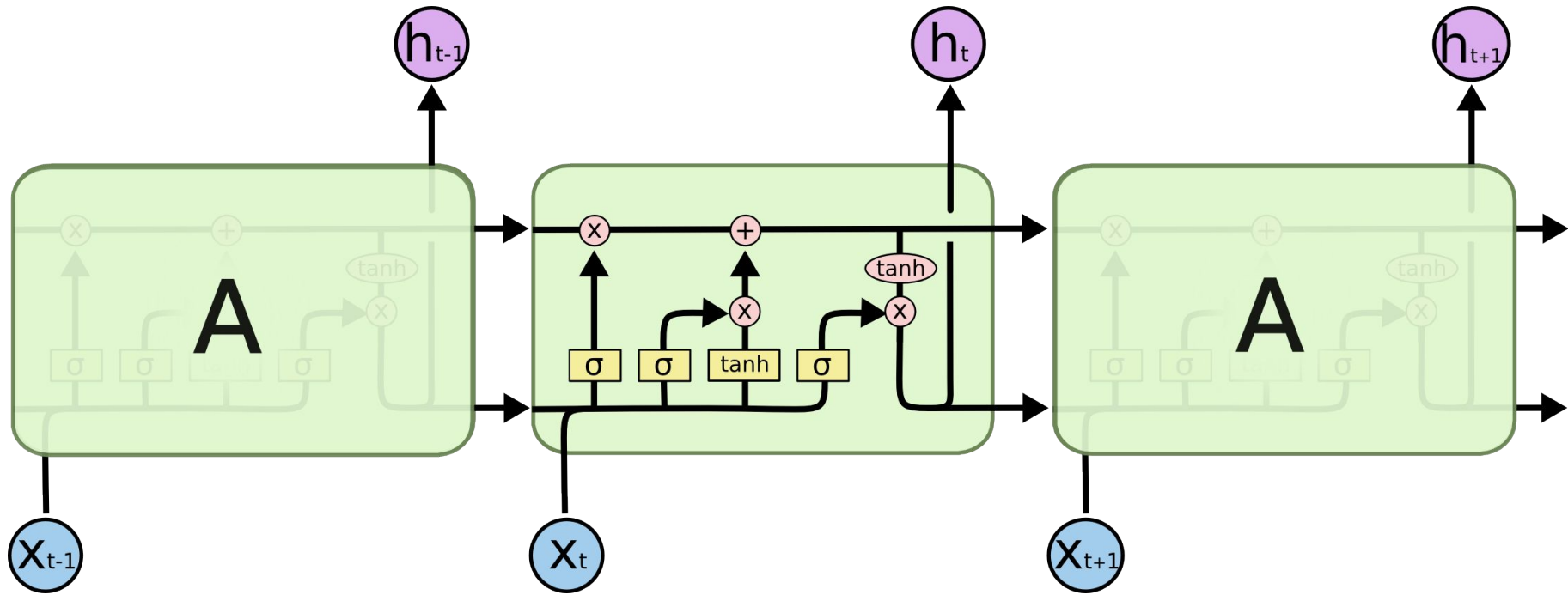
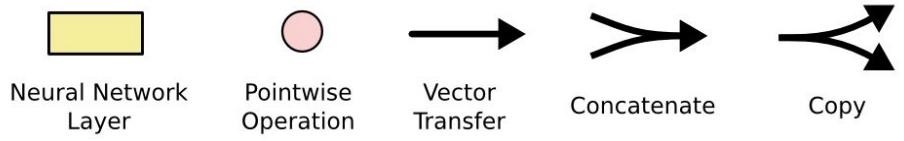
Long term dependencies problem



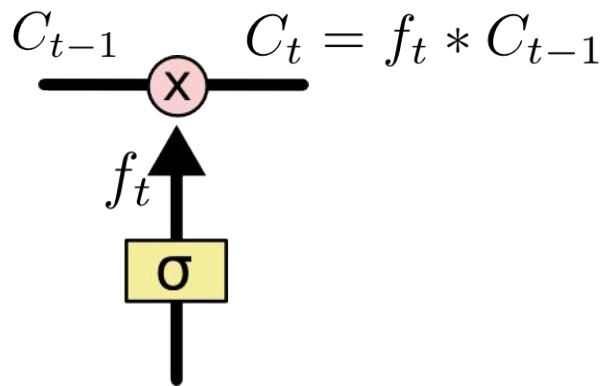
Inside a standard RNN



Inside a LSTM



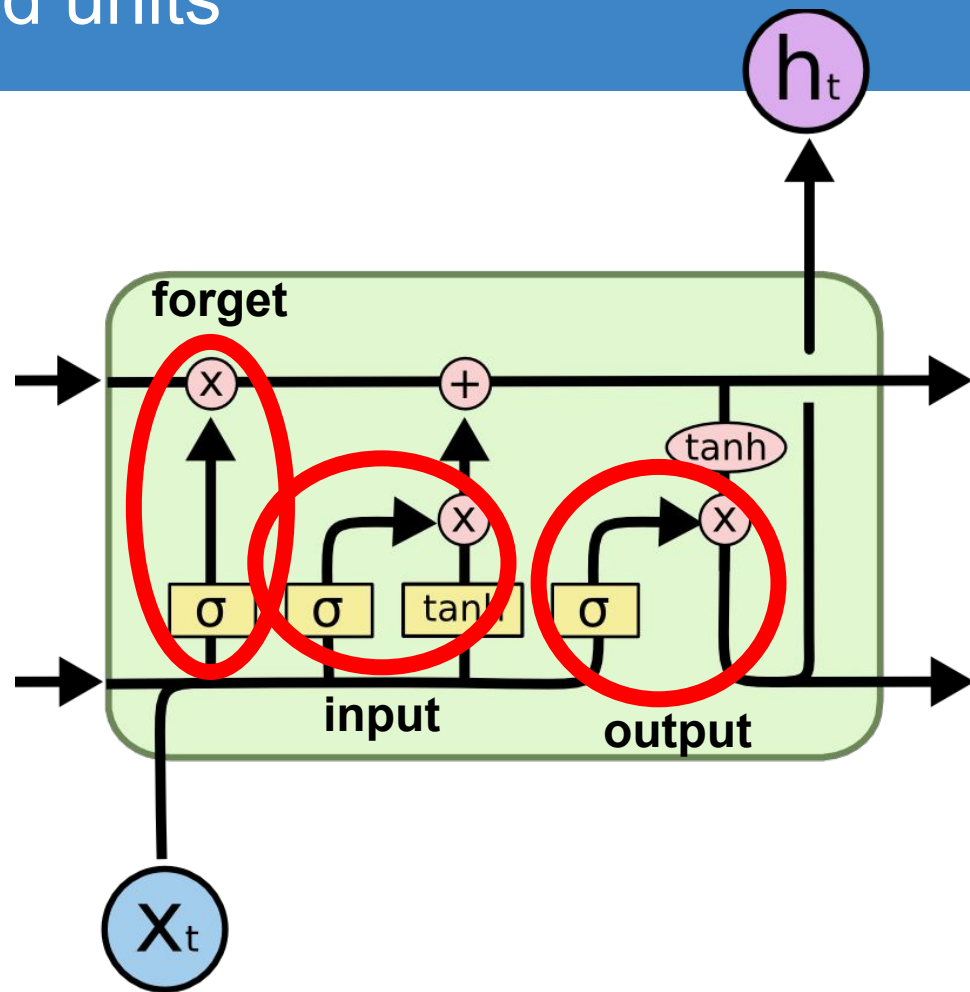
Gated units



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

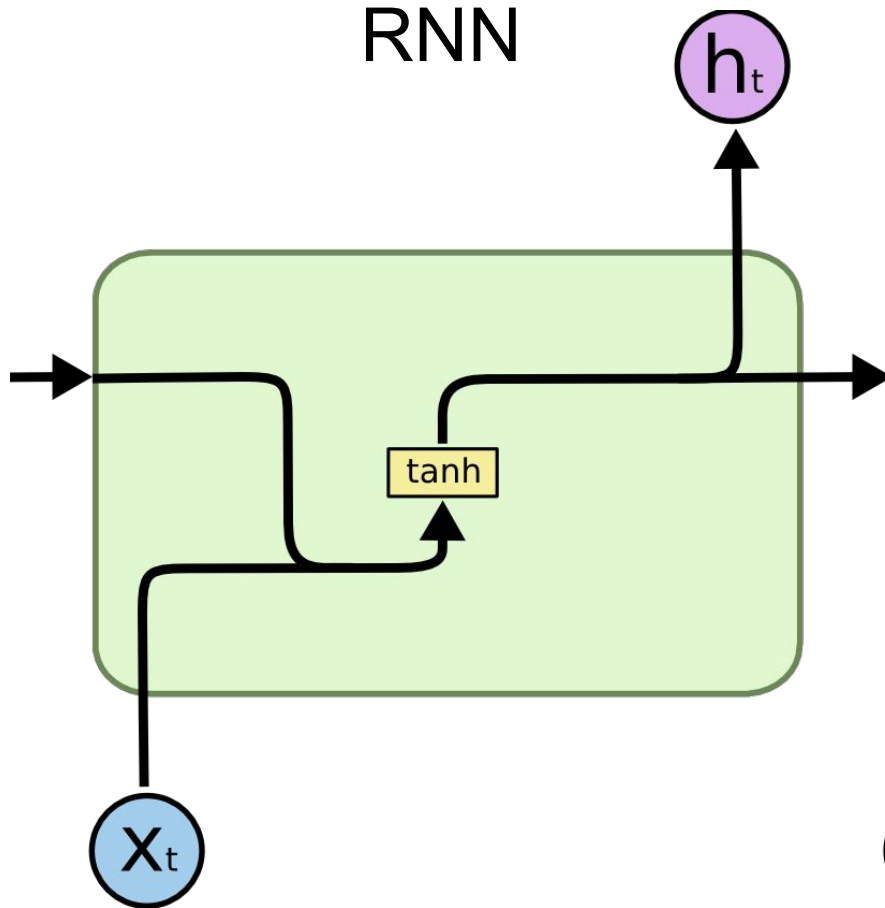
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

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

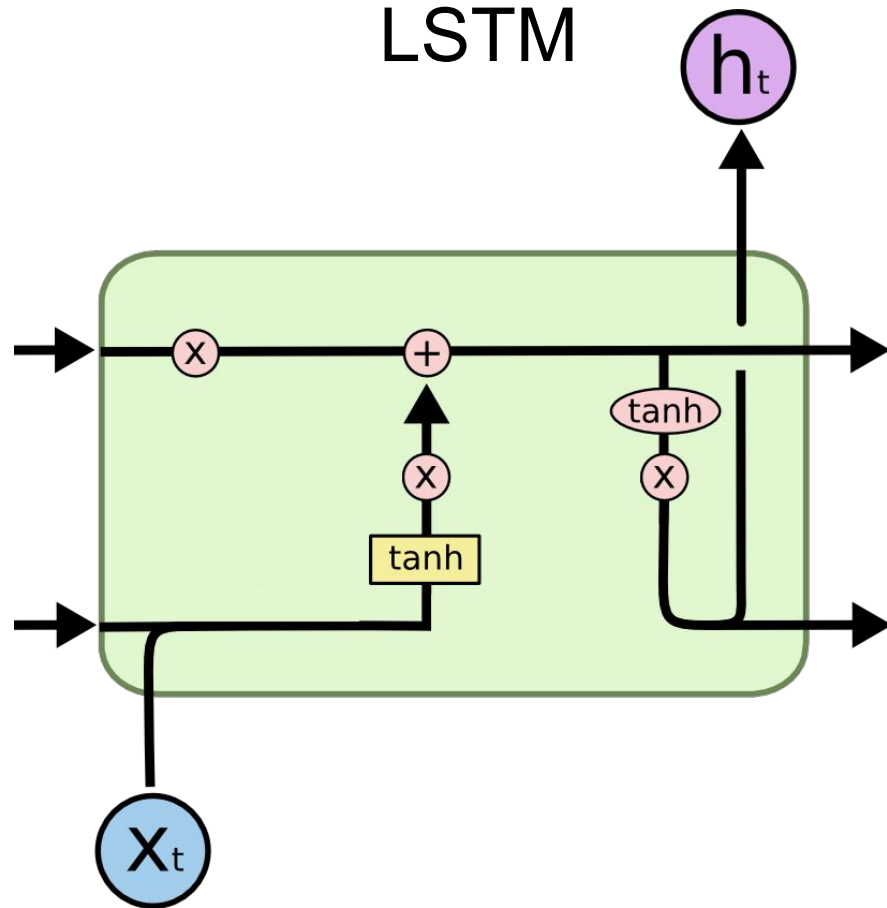


Gated units

RNN



LSTM



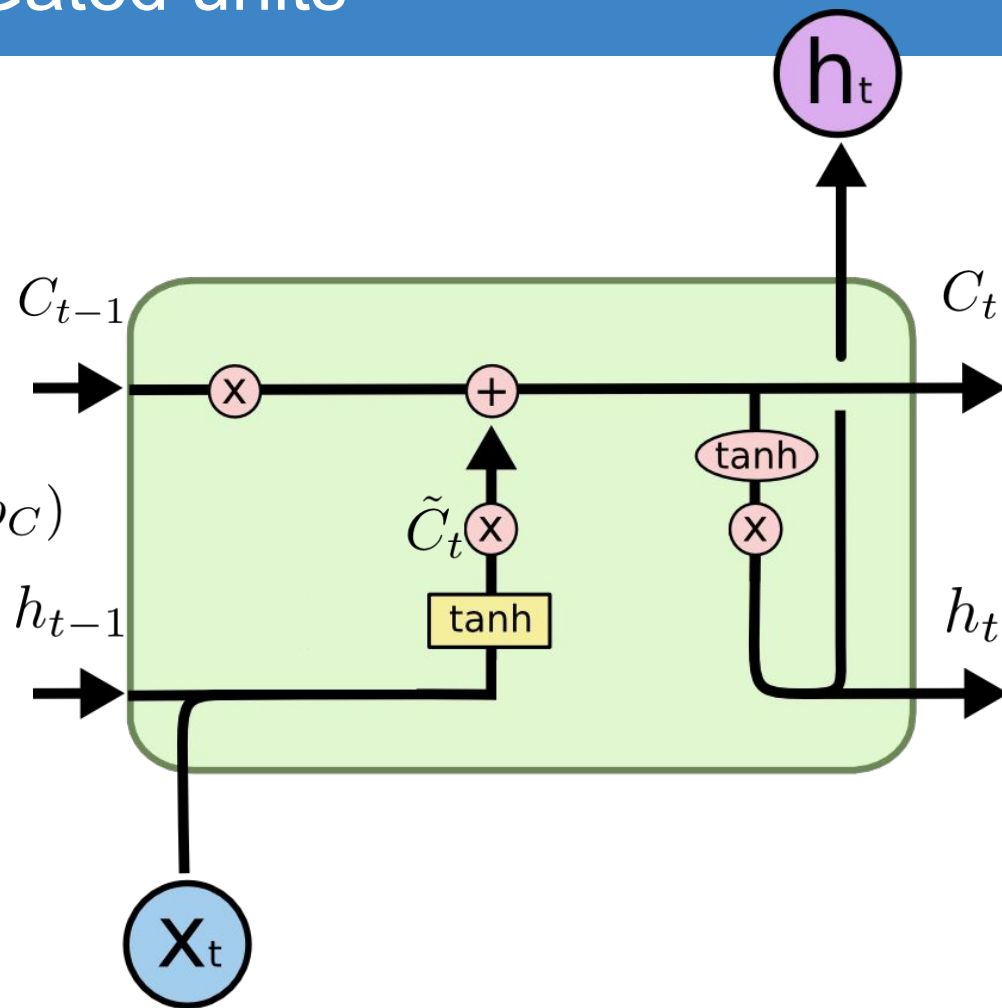
Gated units

LSTM

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

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

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

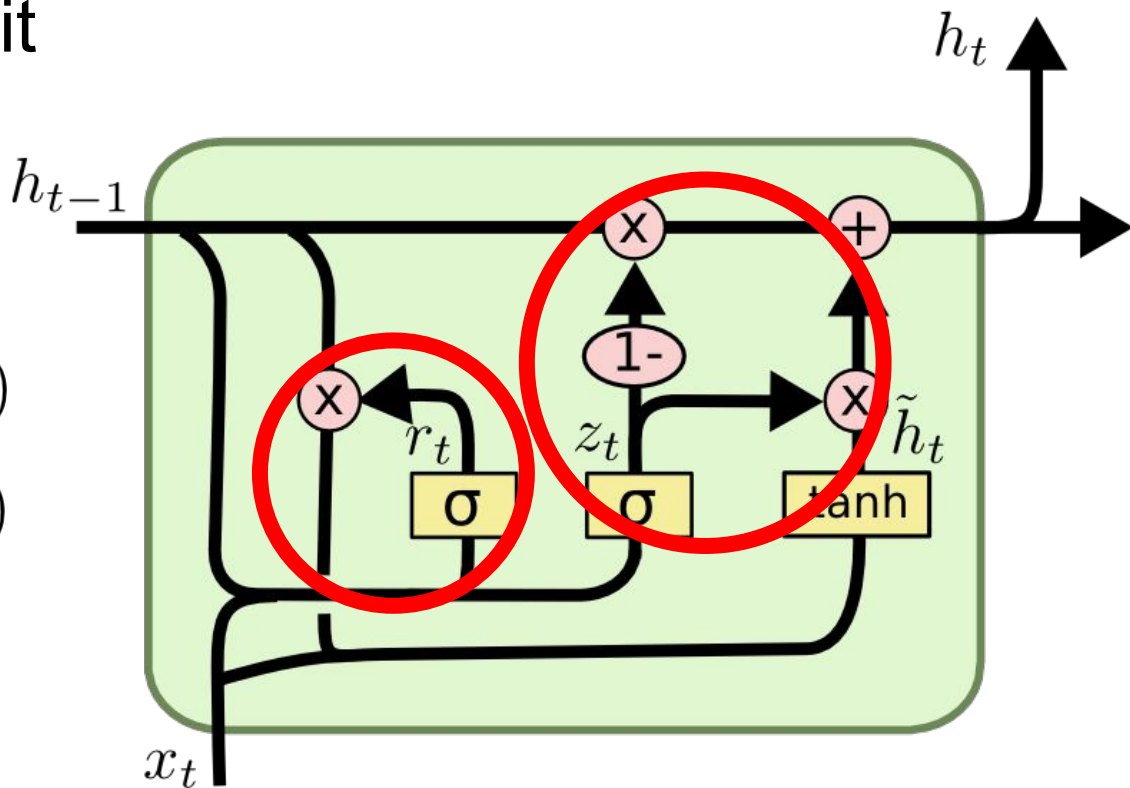


Gated units

Gated Recurrent Unit

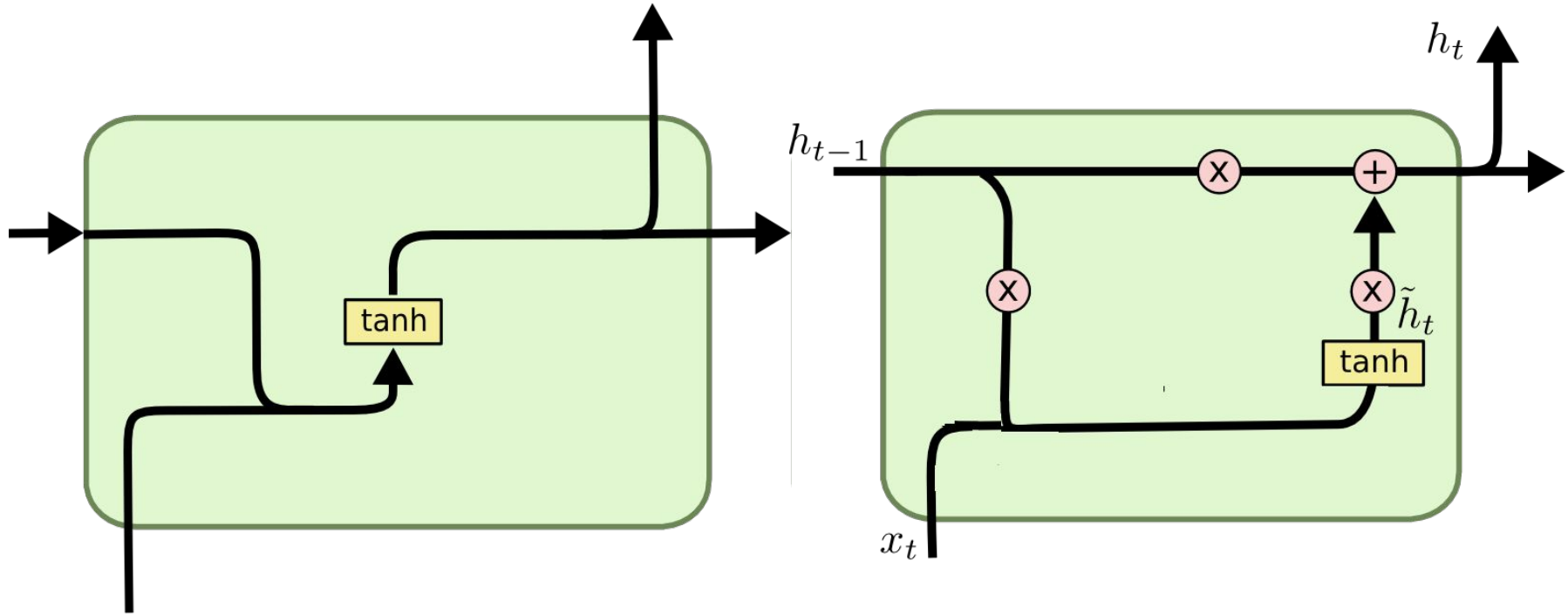
$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

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



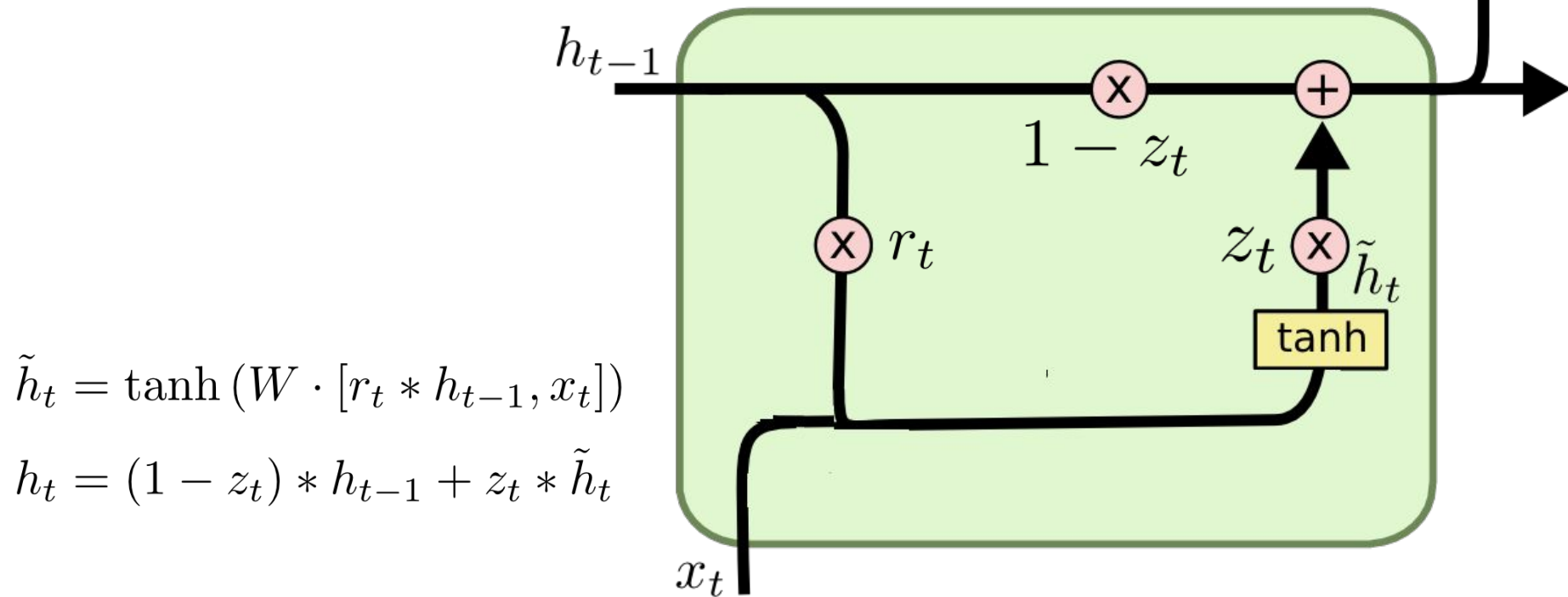
Gated units

Gated Recurrent Unit



Gated units

Gated Recurrent Unit



LSTM equations

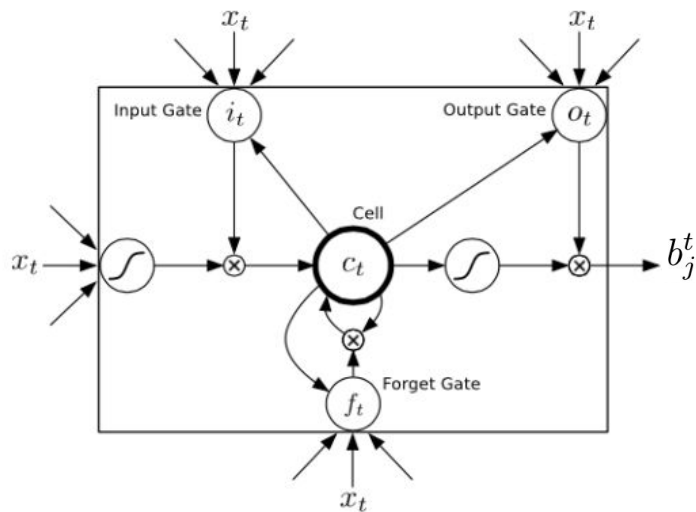


Figure credit: Alex Graves

$$i_j^t = \phi\left(\sum_k \theta_{kj}^{(in)} x_k^t + \sum_l \theta_{lj}^{(hid)} b_l^{t-1} + \sum_m \theta_{mj}^{(cell)} c_m^{t-1}\right)$$

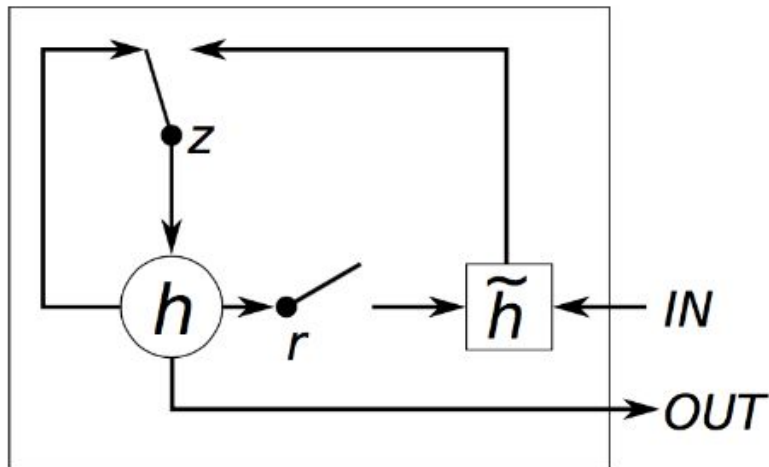
$$f_j^t = \phi\left(\sum_k \theta_{kj}^{(in)} x_k^t + \sum_l \theta_{lj}^{(hid)} b_l^{t-1} + \sum_m \theta_{mj}^{(cell)} c_m^{t-1}\right)$$

$$c_j^t = f_j^t c_j^{t-1} + i_j^t \phi\left(\sum_k \theta_{kj}^{(in)} x_k^t + \sum_l \theta_{lj}^{(hid)} b_l^{t-1}\right)$$

$$o_j^t = \phi\left(\sum_k \theta_{kj}^{(in)} x_k^t + \sum_l \theta_{lj}^{(hid)} b_l^{t-1} + \sum_m \theta_{mj}^{(cell)} c_m^t\right)$$

$$b_j^t = o_j^t \phi(c_j^t)$$

GRU Equations



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

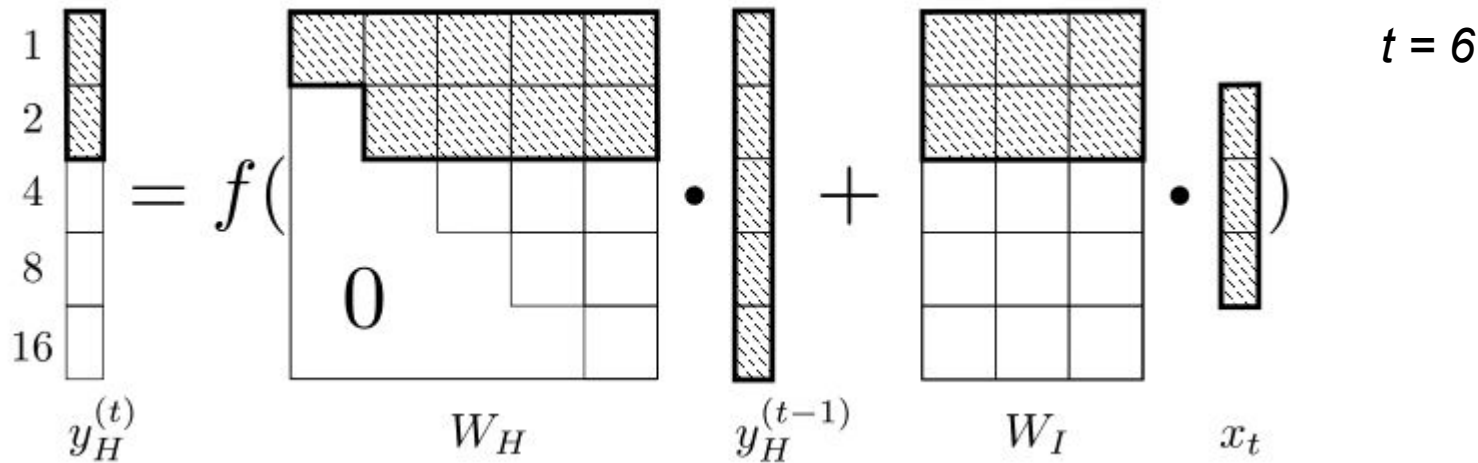
$$z_t^j = \phi(W_z x_t + U_x h_{t-1})^j$$

$$\tilde{h}_t^j = \phi(W x_t + U(r_t \odot h_{t-1}))^j$$

$$r_t^j = \phi(W_r x_t + U_r h_{t-1})^j.$$

Clockwork RNN

[Koutnik et al. 2014]

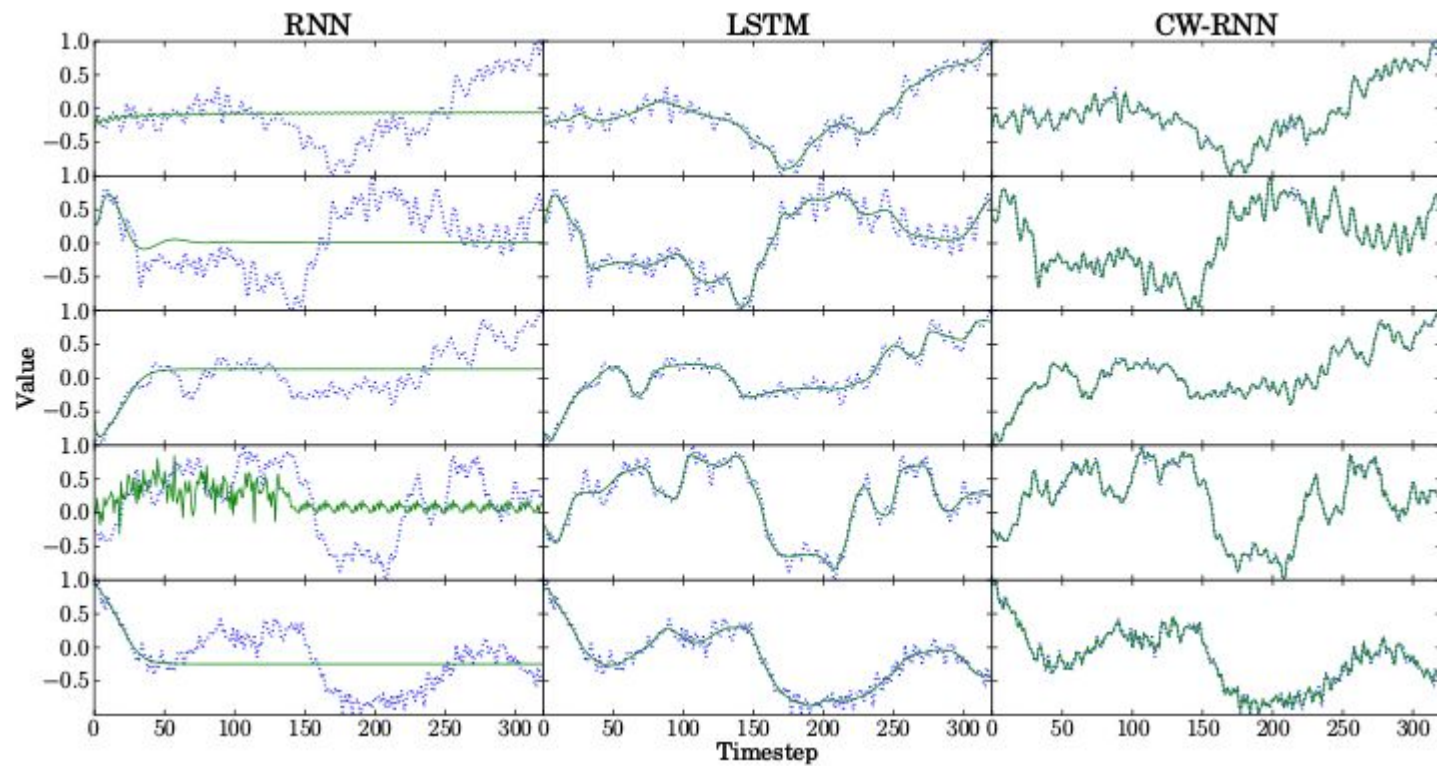


$$\mathbf{y}_H^{(t)} = f_H(\mathbf{W}_H \cdot \mathbf{y}^{(t-1)} + \mathbf{W}_I \cdot \mathbf{x}^{(t)}),$$

$$\mathbf{y}_O^{(t)} = f_O(\mathbf{W}_O \cdot \mathbf{y}_H^{(t)}),$$

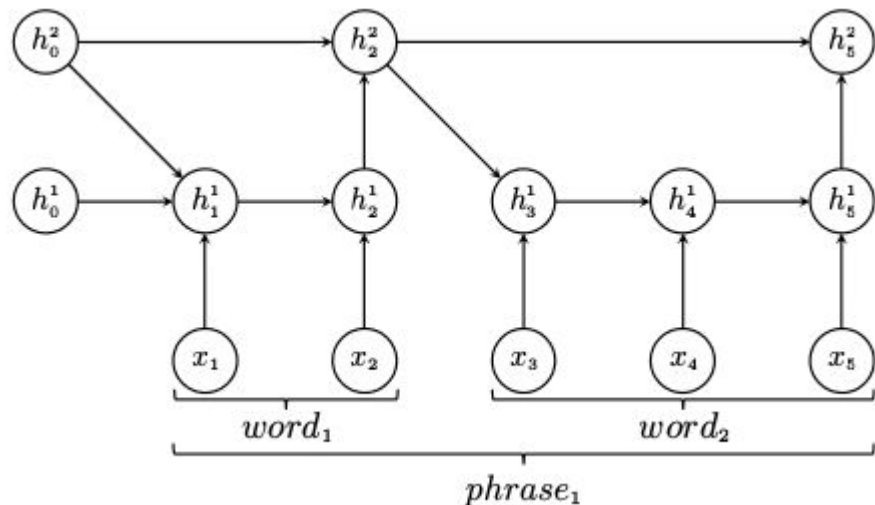
- faster than standard RNN
- handles better long term dependencies

Clockwork RNN



Clockwork RNN

[Chung et al. 2016]



Use prior knowledge about data to define a hierarchy of representations

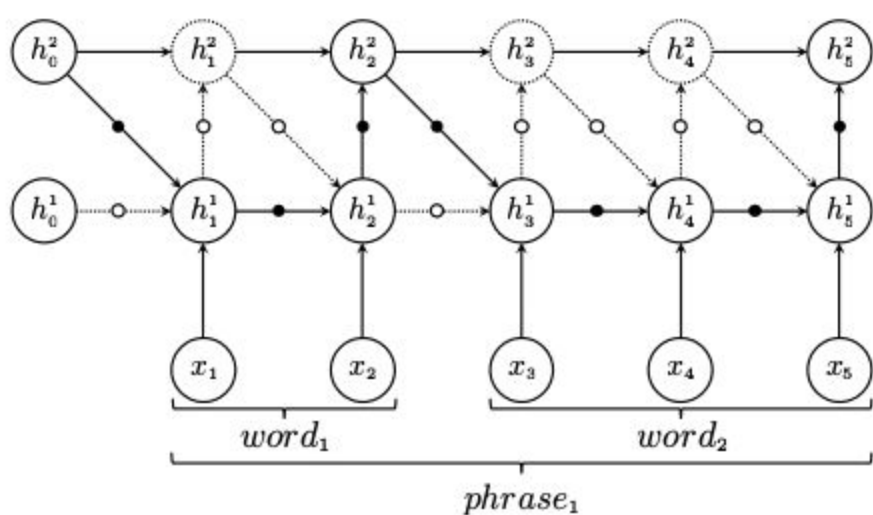
- h_1 represents char
- h_2 represents words
- h_3 represents sentences

UPDATE: standard RNN step

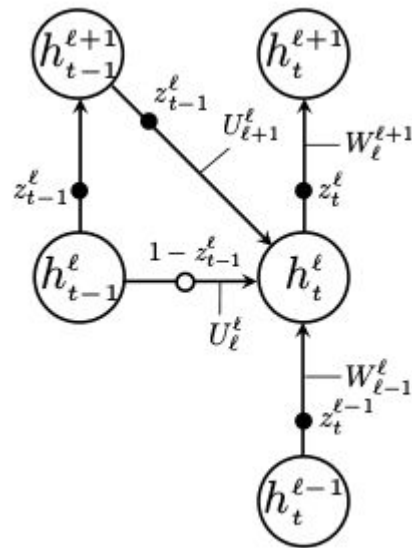
COPY: h is copied to the next timestep

FLUSH: state passed to the upper layer

Hierarchical Multiscale RNN



Learn when to flush and update from data



if $z_{t-1}^\ell = 0$ and $z_t^{\ell-1} = 1$ (UPDATE)

if $z_{t-1}^\ell = 0$ and $z_t^{\ell-1} = 0$ (COPY)

if $z_{t-1}^\ell = 1$ (FLUSH).

Hierarchical Multiscale RNN

Optimization problem:

z is discrete!

- Sampling: REINFORCE
- Soft: softmax on z
- Step Fn:
straight-through estim.-
 - fwd pass step
 - bkd pass hard sigmoid
- Step Fn & Annealing:
 - slope from 1 to 5

Penn Treebank			
	Model		BPC
	Norm-stabilized RNN	(Krueger and Memisevic, 2015)	1.48
	Clockwork RNN	(Koutník <i>et al.</i> , 2014)	1.46
	HF-MRNN	(Mikolov <i>et al.</i> , 2012)	1.41
	MI-RNN	(Wu <i>et al.</i> , 2016)	1.39
	ME n -gram	(Mikolov <i>et al.</i> , 2012)	1.37
	Batch-normalized LSTM	(Cooijmans <i>et al.</i> , 2016)	1.32
	Zoneout RNN	(Krueger <i>et al.</i> , 2016)	1.30
	HM-LSTM	Sampling	1.30
	HM-LSTM	Soft	1.29
	HM-LSTM	Step Fn.	1.28
	HM-LSTM	Step Fn. & Slope Annealing	1.27

Table 1: Bits-per-character on the Penn Treebank test set.

Hierarchical Multiscale RNN

Penn Treebank Line 1



Penn Treebank Line 2



Penn Treebank Line 3



IAM-OnDB dataset:
handwriting data

He believes in low prices. He is

A visualization of the handwriting data from the IAM-OnDB dataset. The text "He believes in low prices. He is" is shown in a cursive font. The letters are colored in a gradient from blue to red. The background is white.

Ground truth of pen-tip location

predict (x,y) coordinates
of strokes

He believes in low prices. He is

A visualization of the predicted pen-tip location for the handwriting data. The text "He believes in low prices. He is" is shown in a cursive font. The letters are colored in a gradient from blue to red. A small square marker is placed at the start of the first stroke of the letter 'H'.

The states of z^1

Regularization in RNNs

Standard Dropout in recurrent layers does not work well because it causes loss of long-term memory!

- Dropout in input-to-hidden and hidden-to-output [*Zaremba et al. 2014*]
- Apply dropout at sequence level (same zeroed units for the entire sequence) [*Gal 2016*]
- Dropout only at the cell update (for LSTM and GRU units) [*Semeniuta et al. 2016*]
- Enforcing norm of the hidden state to be similar along time [*Krueger & Memisevic 2016*]
- Zoneout some hidden units (copy their state to the next timestep) [*Krueger et al. 2016*]

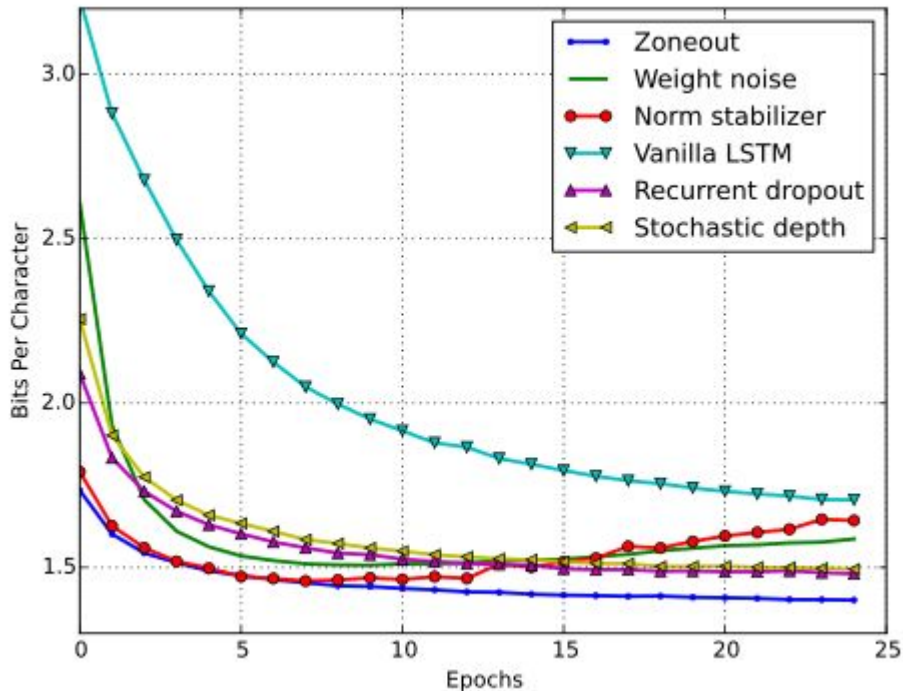
RNN Regularization

Similar to dropout:

instead of dropping out hidden units,
here it zones out hidden units!

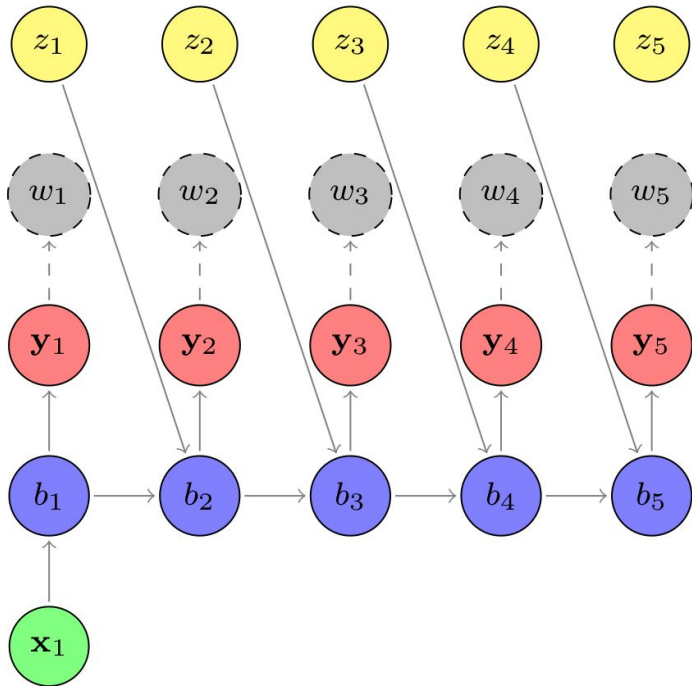
character-level Penn Treebank

Model	Valid	Test
LSTM	1.664	1.398
Recurrent dropout	1.481	1.382
BatchNorm	—	1.32
Zoneout	1.362	1.297

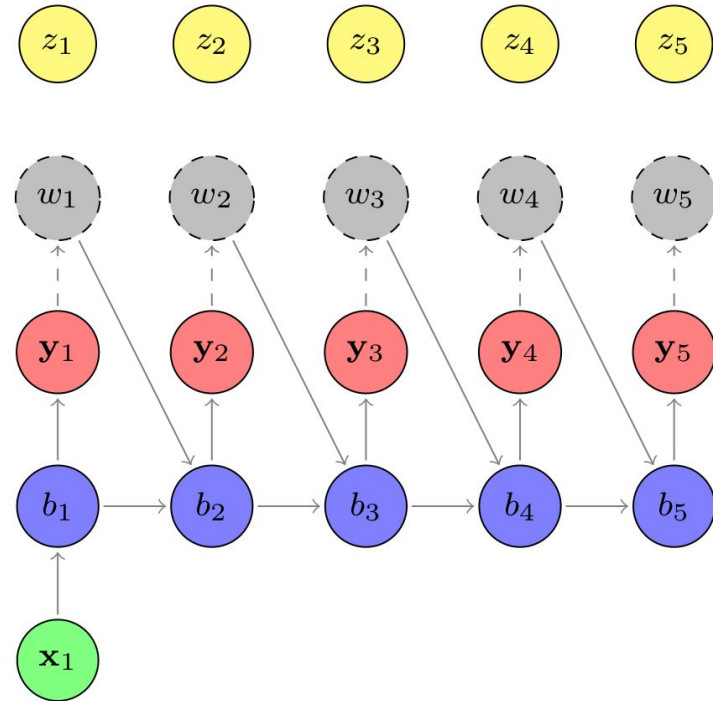


Teacher Forcing for sequence prediction

Training: Teacher forcing



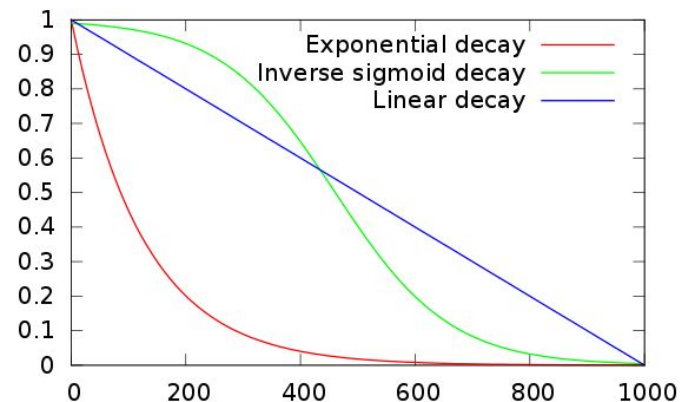
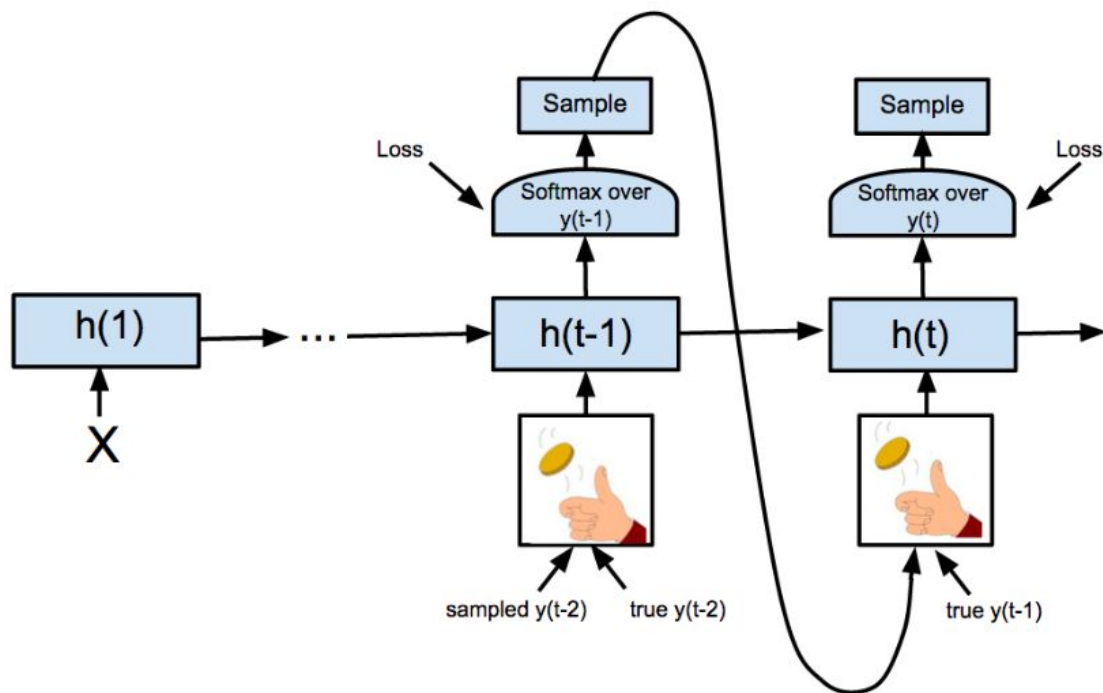
Test: Sampling from Y



Scheduled Sampling

[Bengio et al. 2015]

Slowly move from *Teacher Forcing* to *Sampling*



Probability of sampling from the ground truth

Scheduled Sampling

Microsoft COCO development set

Approach vs Metric	BLEU-4	METEOR	CIDER
Baseline	28.8	24.2	89.5
Baseline with Dropout	28.1	23.9	87.0
Always Sampling	11.2	15.7	49.7
Scheduled Sampling	30.6	24.3	92.1
Uniform Scheduled Sampling	29.2	24.2	90.9
Baseline ensemble of 10	30.7	25.1	95.7
Scheduled Sampling ensemble of 5	32.3	25.4	98.7

Baseline: based on google NIC

Baseline with dropout: see part about regularization of RNN

Always Sampling: using sampling from the beginning of training

Scheduled Sampling: the proposed approach with inverse Sigmoid decay

Uniform scheduled Sampling: using scheduled sampling but with uniform Y sampling

Sequence Level Training

[Ranzato et al. 2016]

During training objective is different than at test time

- Training: generate next word given the previous
- Test: generate the entire sequence given an initial state

Optimize directly evaluation metric (e.g. BLUE score for sentence generation)

Set the problem as a Reinforcement Learning:

- RNN is an *Agent*
- *Policy* defined by the learned parameters
- *Action* is the selection of the next word based on the policy
- *Reward* is the evaluation metric

Comparison of different training approaches

- XENT = *Cross Entropy loss*: standard approach, but suffers of exposure bias
- DAD = *Scheduled Sampling*: the loss is always based on fixed target labels, that may not be aligned with the so far generated sentence
- E2E = *End-to-end backprop*: select top k output and renormalize. Uses a schedule between target labels and top k output
- MIXER = *Sequence level training*: optimize at sequence level. Start from model trained with Cross Entropy and use reinforce for the last K steps of the sequence.

<i>TASK</i>	XENT	DAD	E2E	MIXER
<i>summarization</i>	13.01	12.18	12.78	16.22
<i>translation</i>	17.74	20.12	17.77	20.73
<i>image captioning</i>	27.8	28.16	26.42	29.16

Further reading

- Other topologies
 - Bidirectional RNN
 - Recursive RNN
- Applications
 - RNN for text generation
 - RNN for image captioning
 - RNN language translation
 - pixel RNN for image generation
- RNN + Attention models
 - Draw for image generation
 - Encode + Review + Decode

Outline:

- Motivation: why we need RNNs / intuition of how they work. (for the moment I will use old illustrations, then I will try to use yours...)
- Equations and notations (+derivation of backprop?)
- RNN topologies
- Intuitions of why the gradients vanish.
- Sufficient condition for vanishing gradient.
- Solutions to the vanishing gradient problem.
 - Gated units (Lstm/Gru). (some equations)
 - Clockwork RNN.
 - Hierarchical Multiscale RNN (<http://arxiv.org/pdf/1609.01704v2.pdf>).
- Teacher forcing
 - scheduled sampling
 - optimization on BLUE scores
- RNNs with bells and whistles
 - Regularisation of RNNs (zone-out)