Recurrent neural networks

Ekaterina Lobacheva

Senior lecturer and PhD student at HSE

Research intern at Kaspersky Lab

Deep|Bayes

Motivation

Sequence input:

- Sentiment analysis

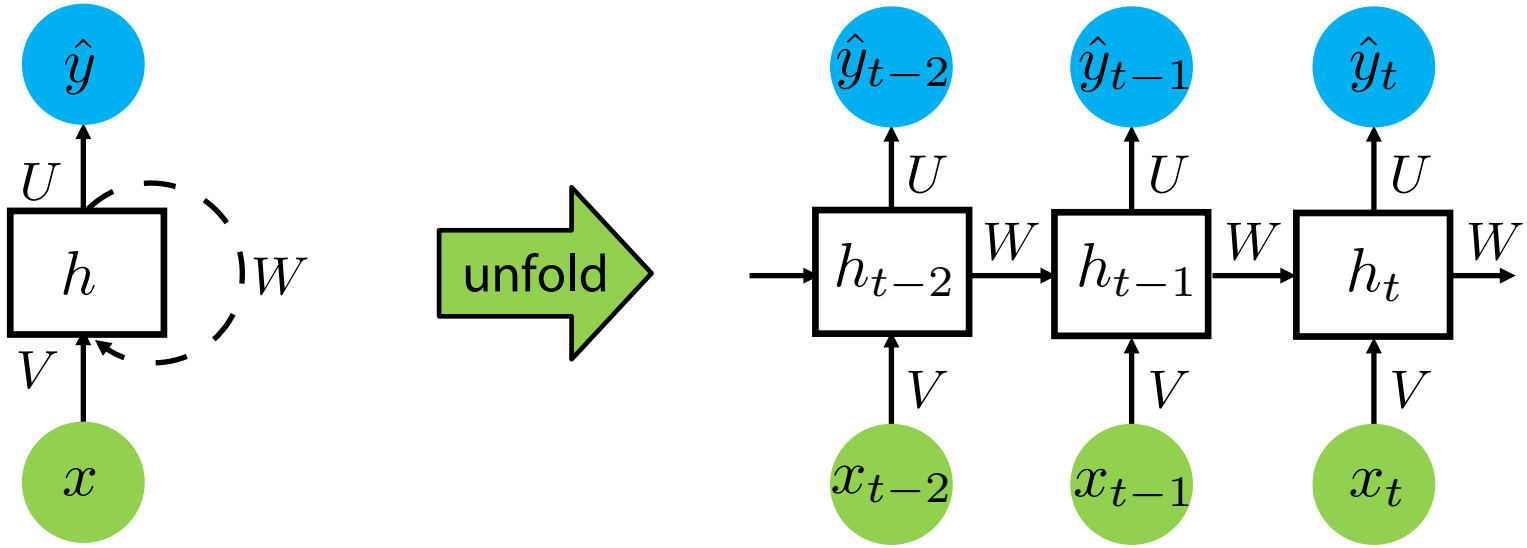
Sequence output:

- Image captioning

Sequence input and output:

- POS tagging
- Language model
- Handwriting generation
- Speech to text / text to speech
- Machine Translation

Recurrent neural network



x - input

$\hat{y}$ - output (prediction)

h - hidden state

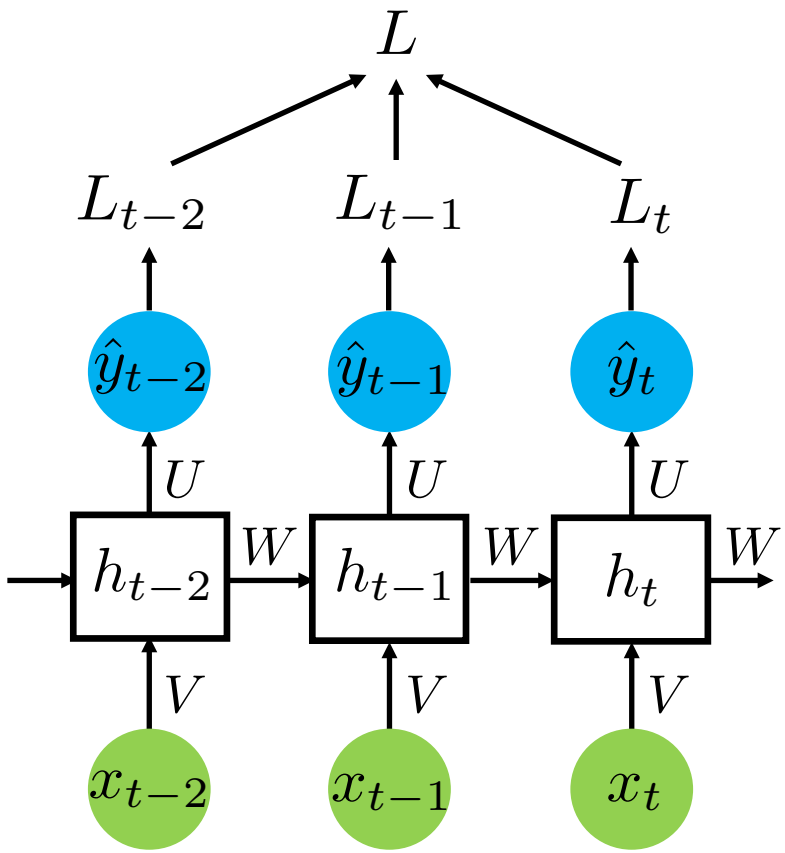
$$h_t = f_h(Vx_t + Wh_{t-1} + b_h)$$

$$\hat{y}_t = f_y(Uh_t + b_y)$$

Backpropagation through time

At each time step:

- y_t - true label
- $\hat{y}_t$ - prediction
- $L_t(y_t, \hat{y}_t)$ - some loss function



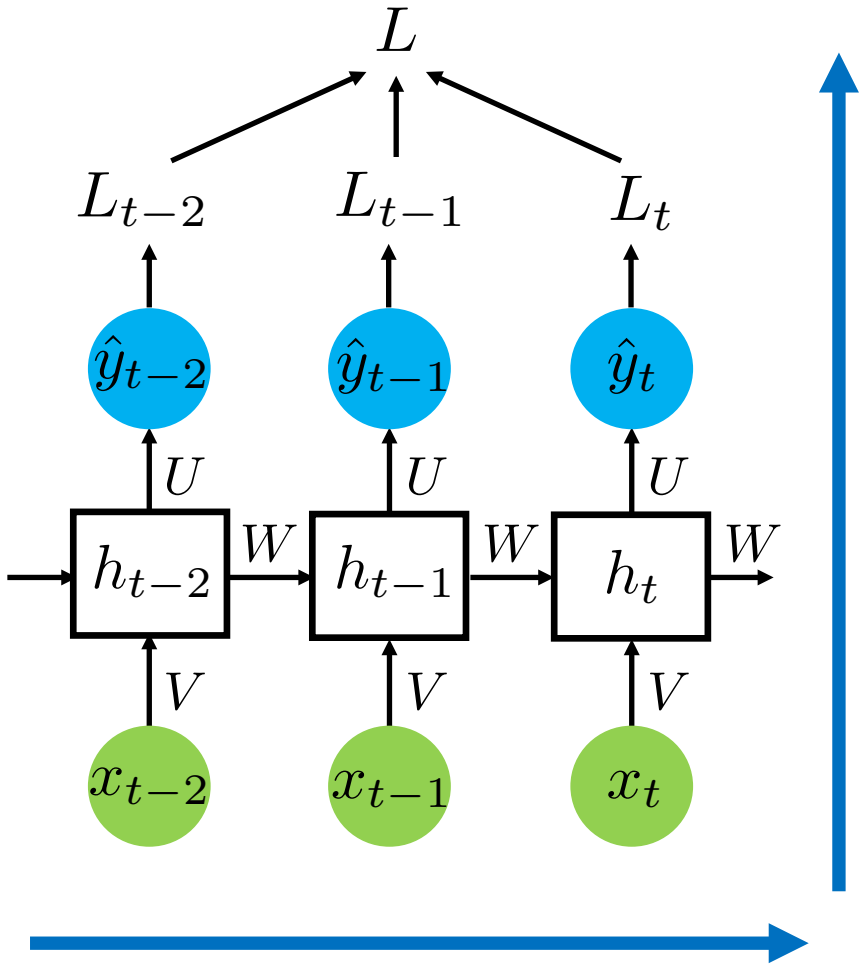
Loss:

$$L = \sum_i L_i(y_i, \hat{y}_i)$$

Backpropagation through time

Forward pass:

$$h_t,\ \hat{y}_t,\ L_t,\ L$$



Backpropagation through time

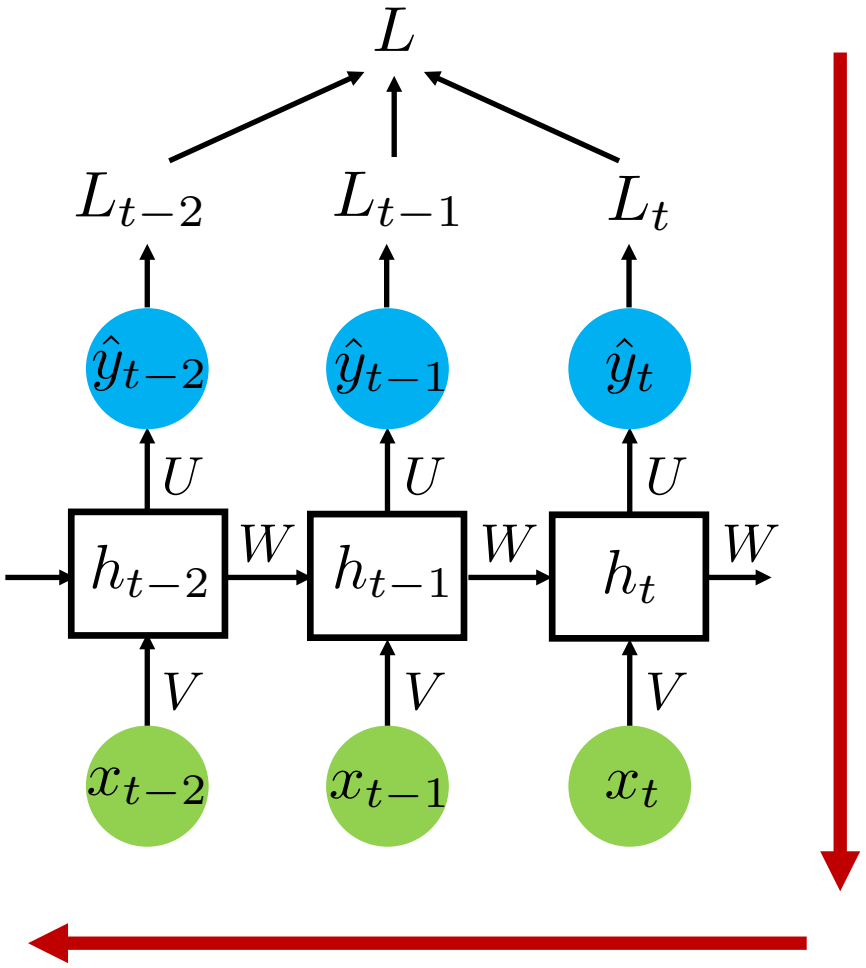
Forward pass:

$$h_t,\ \hat{y}_t,\ L_t,\ L$$

Backward pass:

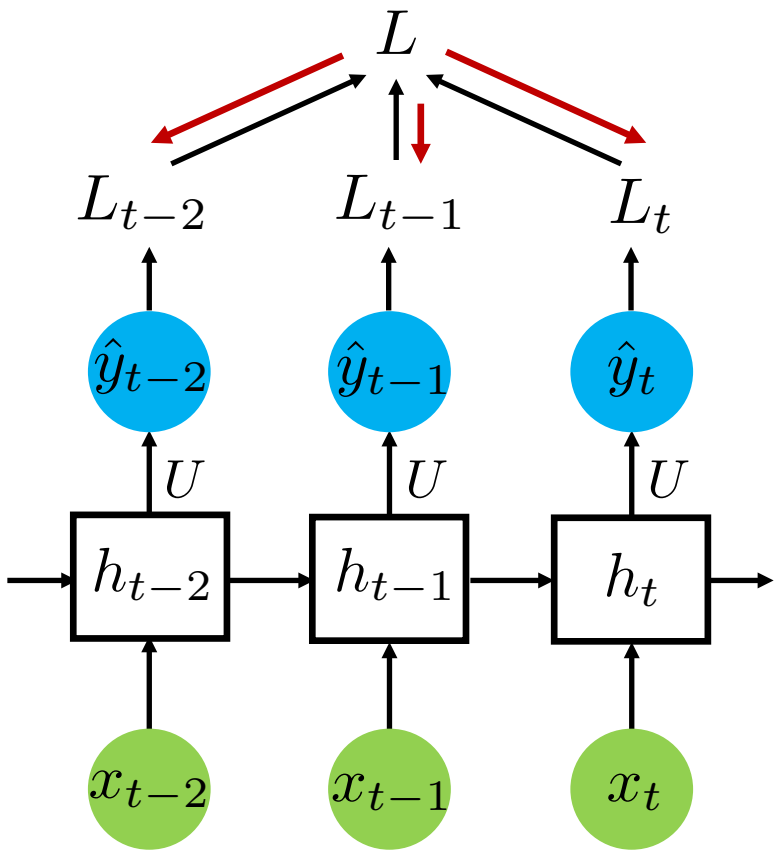
$$\frac{\partial L}{\partial U},\ \frac{\partial L}{\partial V},\ \frac{\partial L}{\partial W},$$
$$\frac{\partial L}{\partial b_x},\ \frac{\partial L}{\partial b_h}$$

We backpropagate through layers and time



Backpropagation through time

$$\frac{\partial L}{\partial U} = \sum_{i=0}^{T} \frac{\partial L_i}{\partial U}$$



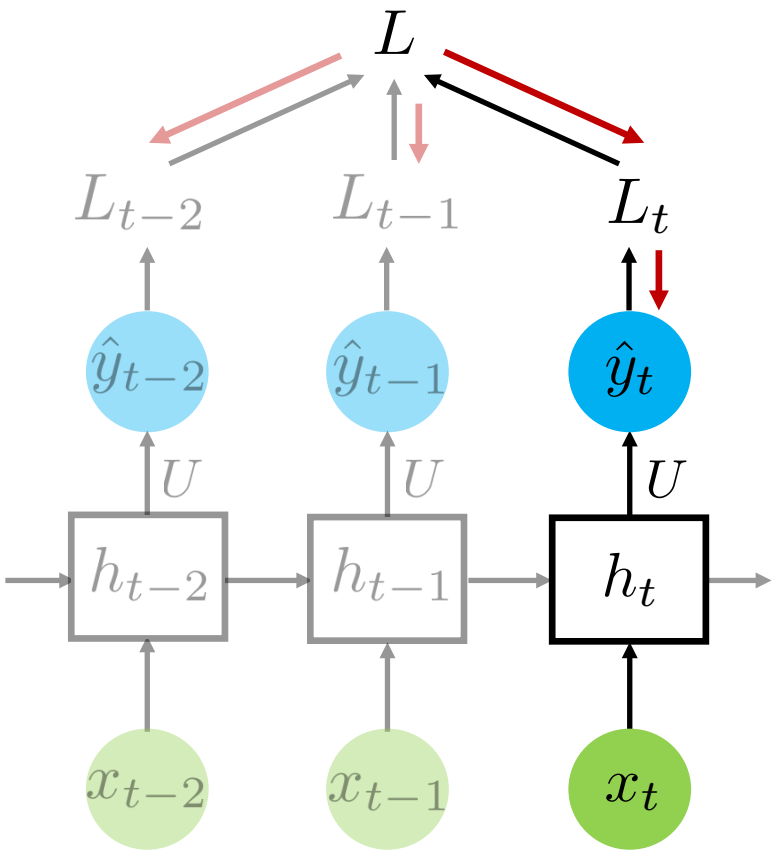
Backpropagation through time

$$\frac{\partial L}{\partial U} = \sum_{i=0}^{T} \frac{\partial L_i}{\partial U}$$

$$\frac{\partial L_t}{\partial U} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial U}$$

$$\hat{y}_t = f_y(Uh_t + b_y)$$

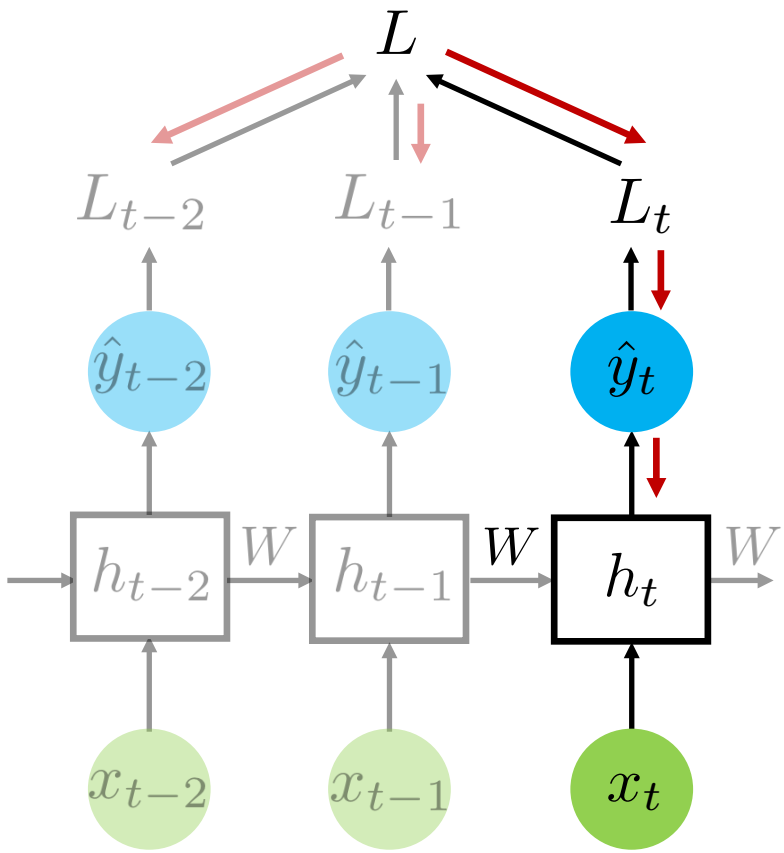
this is the only dependence



Backpropagation through time

$$\frac{\partial L}{\partial W} = \sum_{i=0}^{T} \frac{\partial L_i}{\partial W}$$

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W}$$



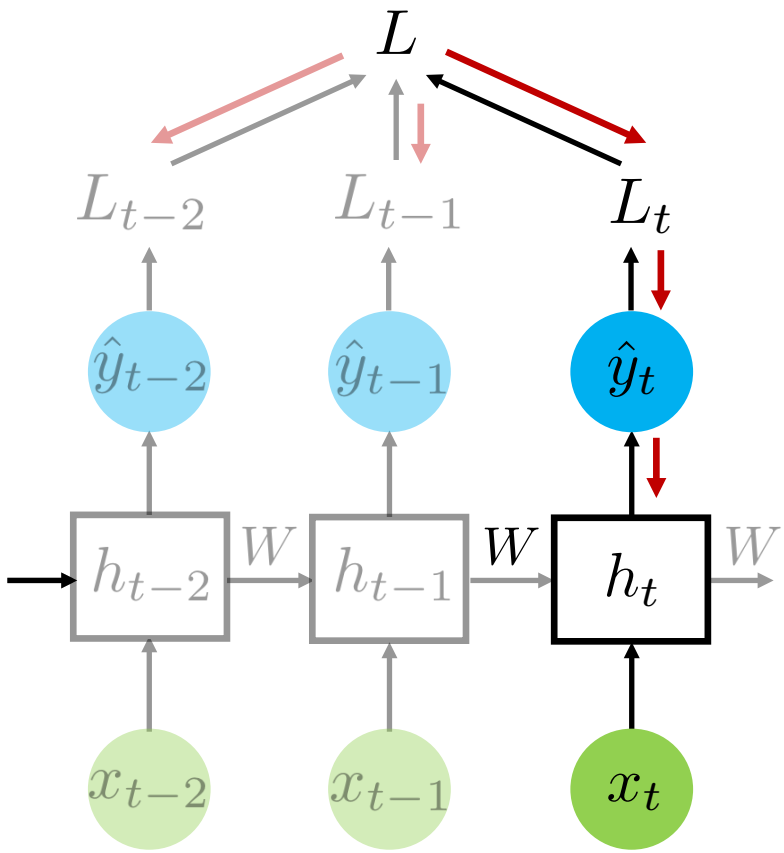
Backpropagation through time

$$\frac{\partial L}{\partial W} = \sum_{i=0}^{T} \frac{\partial L_i}{\partial W}$$

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W}$$

$$h_t = f_h(V x_t + W h_{t-1} + b_h)$$

This is NOT the only dependence!



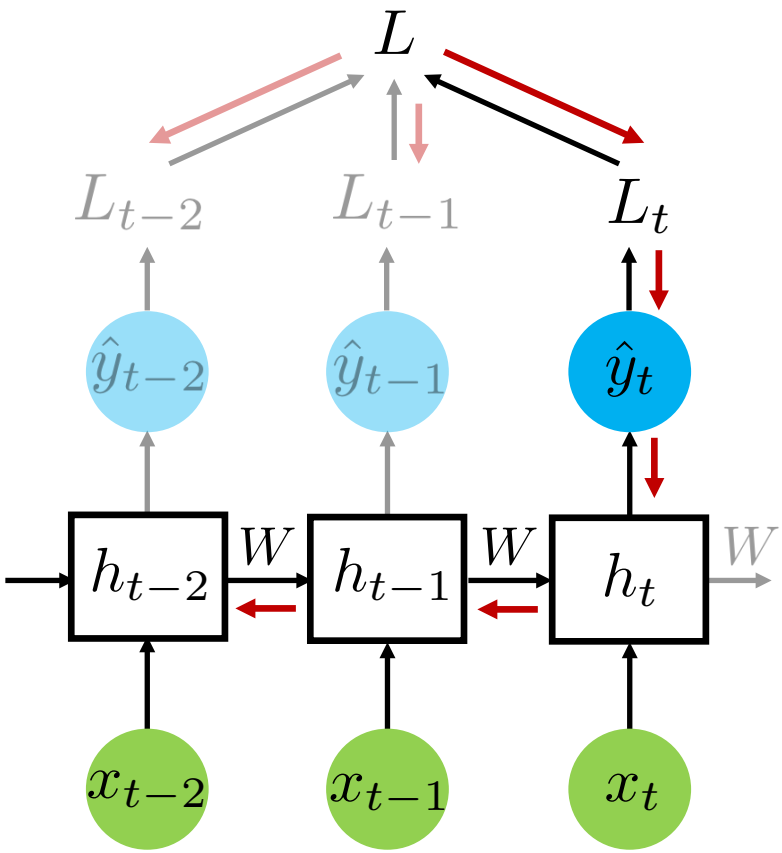
Backpropagation through time

$$\frac{\partial L}{\partial W} = \sum_{i=0}^{T} \frac{\partial L_i}{\partial W}$$

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W}$$

$$h_t = f_h(V x_t + W h_{t-1} + b_h)$$

This is NOT the only dependence!



$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \left(\frac{\partial h_t}{\partial W} + \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial W} + \cdots \right)$$

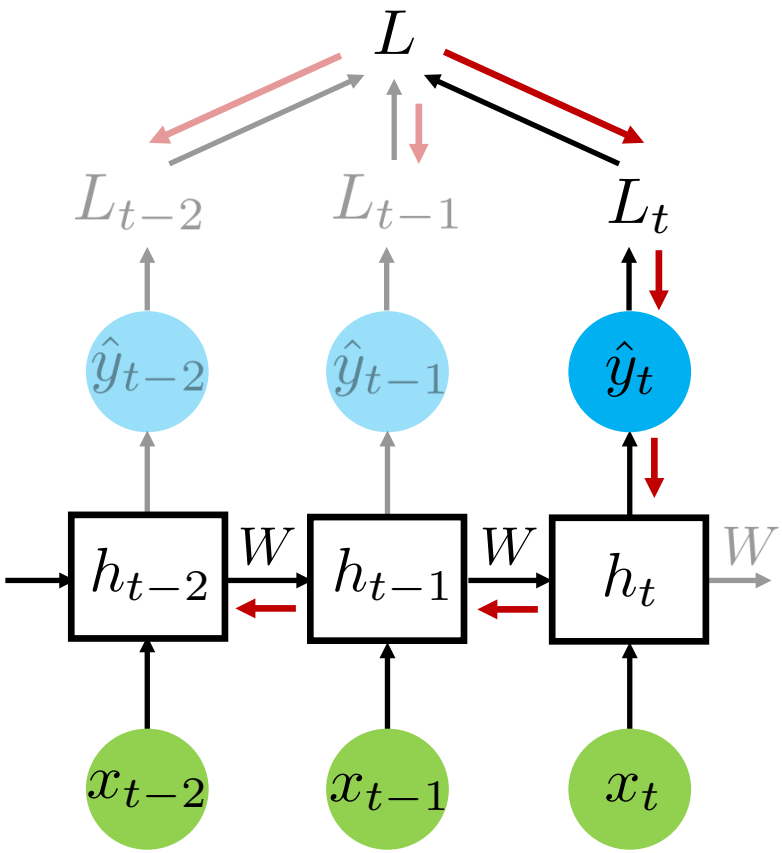
Backpropagation through time

$$\frac{\partial L}{\partial W} = \sum_{i=0}^{T} \frac{\partial L_i}{\partial W}$$

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \frac{\partial h_t}{\partial W}$$

$$h_t = f_h(V x_t + W h_{t-1} + b_h)$$

This is NOT the only dependence!



$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial h_t} \sum_{k=0}^{t} \left(\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}$$

Vanishing and exploding grads

$$\frac{\partial L_t}{\partial W} \propto \sum_{k=0}^{t} \left(\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}$$

$$\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 < 1$$  Vanishing gradients

$$\left\| \frac{\partial h_i}{\partial h_{i-1}} \right\|_2 > 1$$  Exploding gradients

Vanishing and exploding grads

$$h_t = f_h(Wx_t + Wh_{t-1} + b_h) = f_h(pr_t)$$

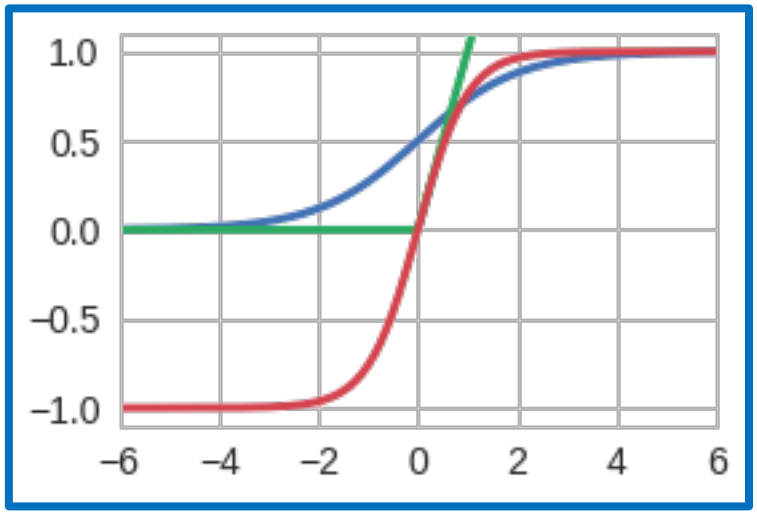
$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial h_t}{\partial inp_t} \frac{\partial inp_t}{\partial h_{t-1}} = diag(f_h'(pr_t)) \cdot W$$

Vanishing and exploding grads

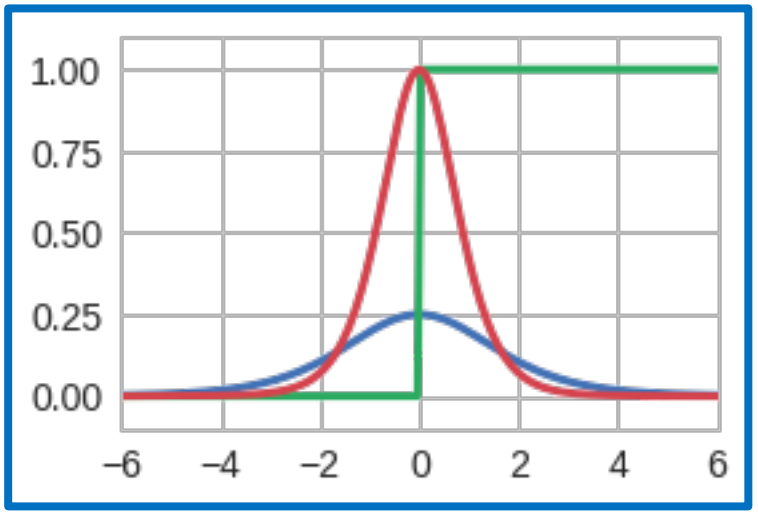
$$h_t = f_h(Wx_t + Wh_{t-1} + b_h) = f_h(pr_t)$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial h_t}{\partial inp_t}\frac{\partial inp_t}{\partial h_{t-1}} = \boxed{diag(f_h'(pr_t))} \cdot W$$

sigmoid, tanh, ReLU

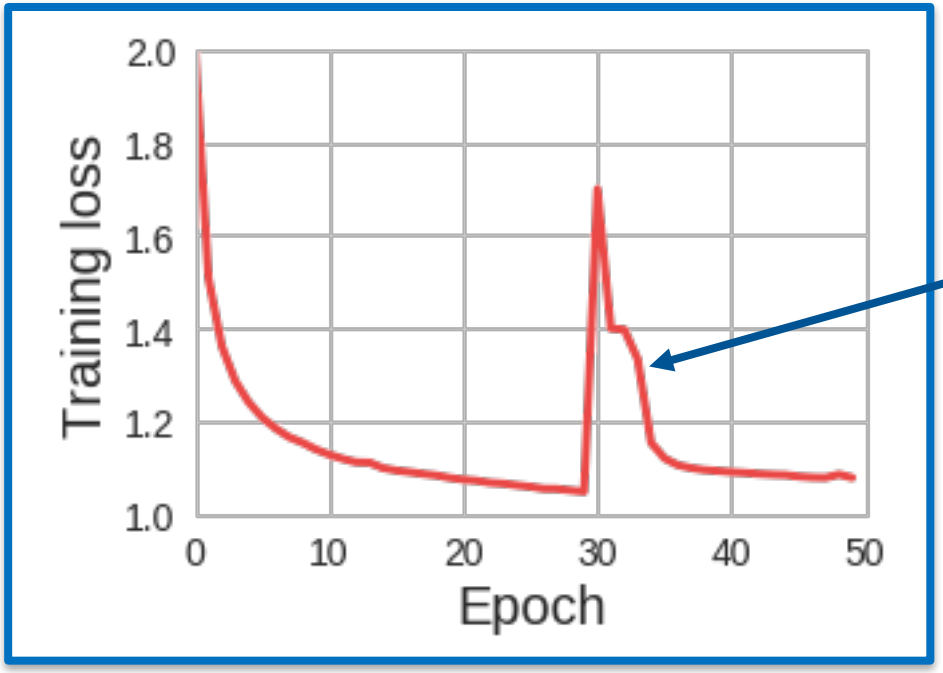


Derivatives



Exploding gradients: detection

Unstable learning curve



This is it!

If the gradients contain NaNs you end up with NaNs in the weights

Gradient clipping

Gradient $g = \frac{\partial L}{\partial \theta}$, θ - all the network parameters

If $||g||$> threshold:

$$g \leftarrow \frac{threshold}{\|g\|} g$$

Simple but still very effective!

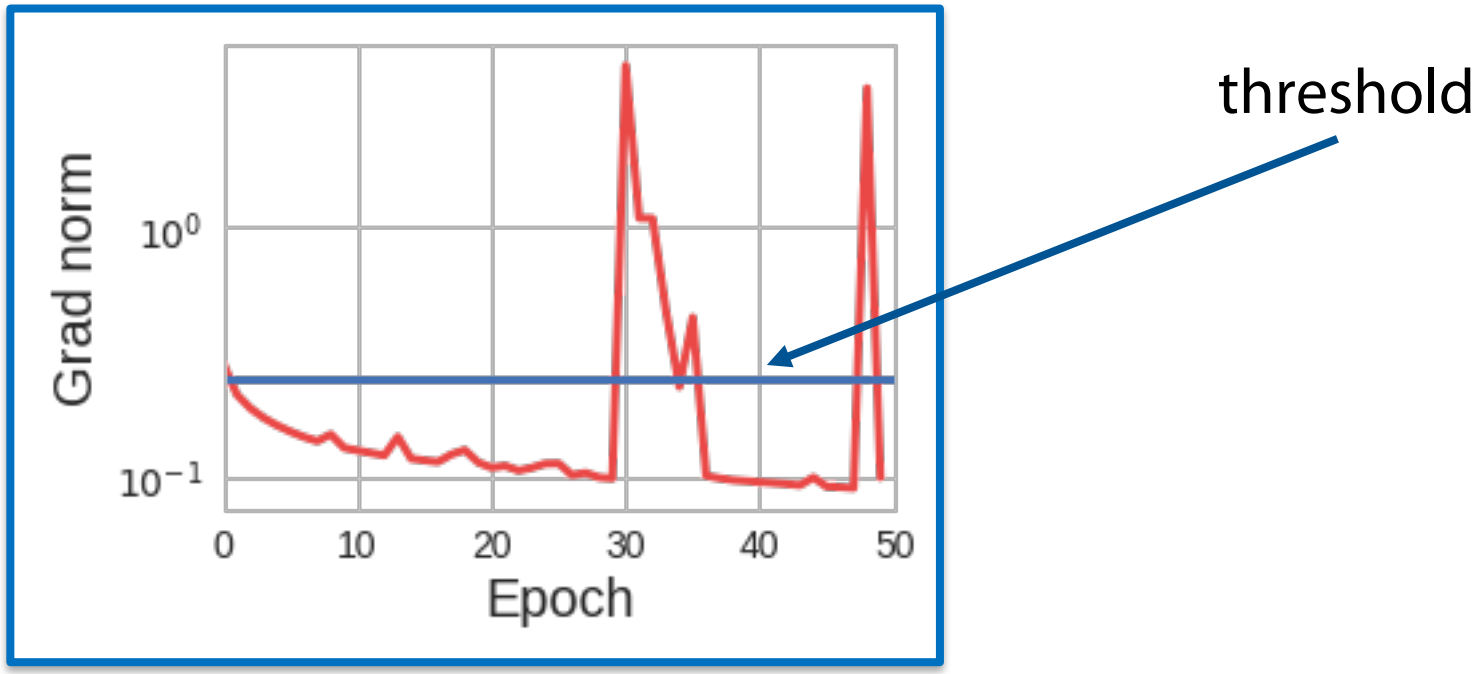
It is enough to clip $\frac{\partial h_t}{\partial h_{t-1}}$

[Pascanu et al., 2012]

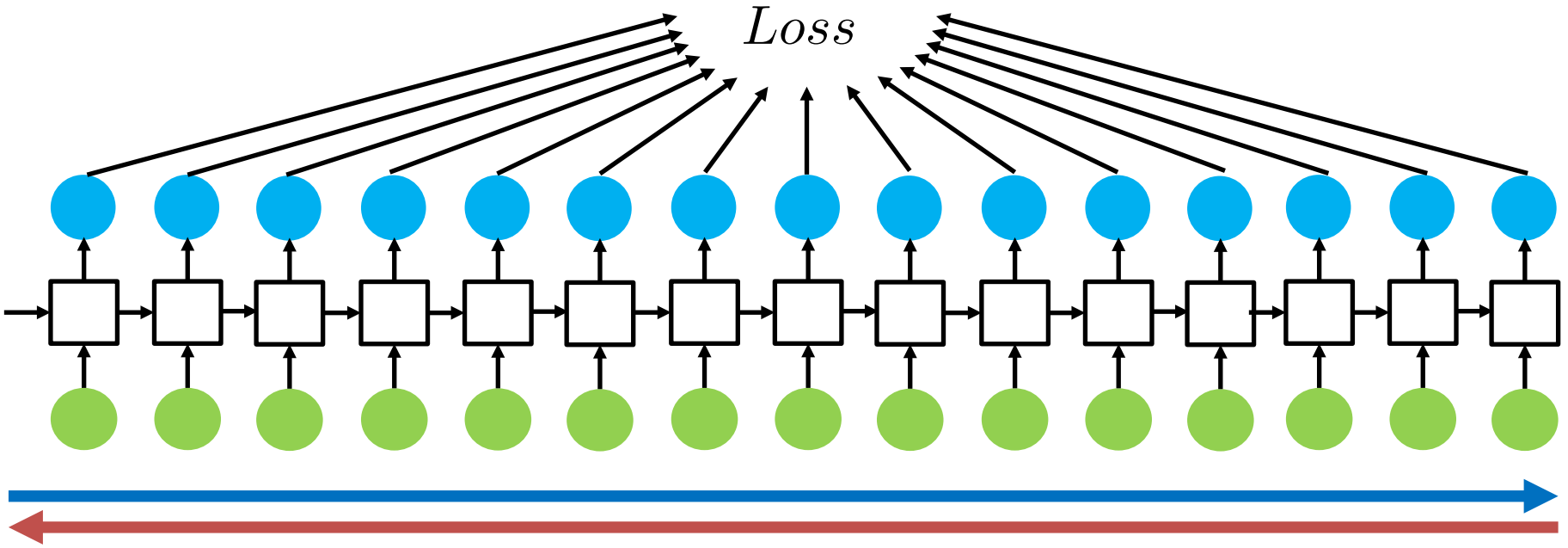
Gradient clipping

Choose the highest threshold which helps to overcome the exploding gradient problem

Curve of the gradient norm



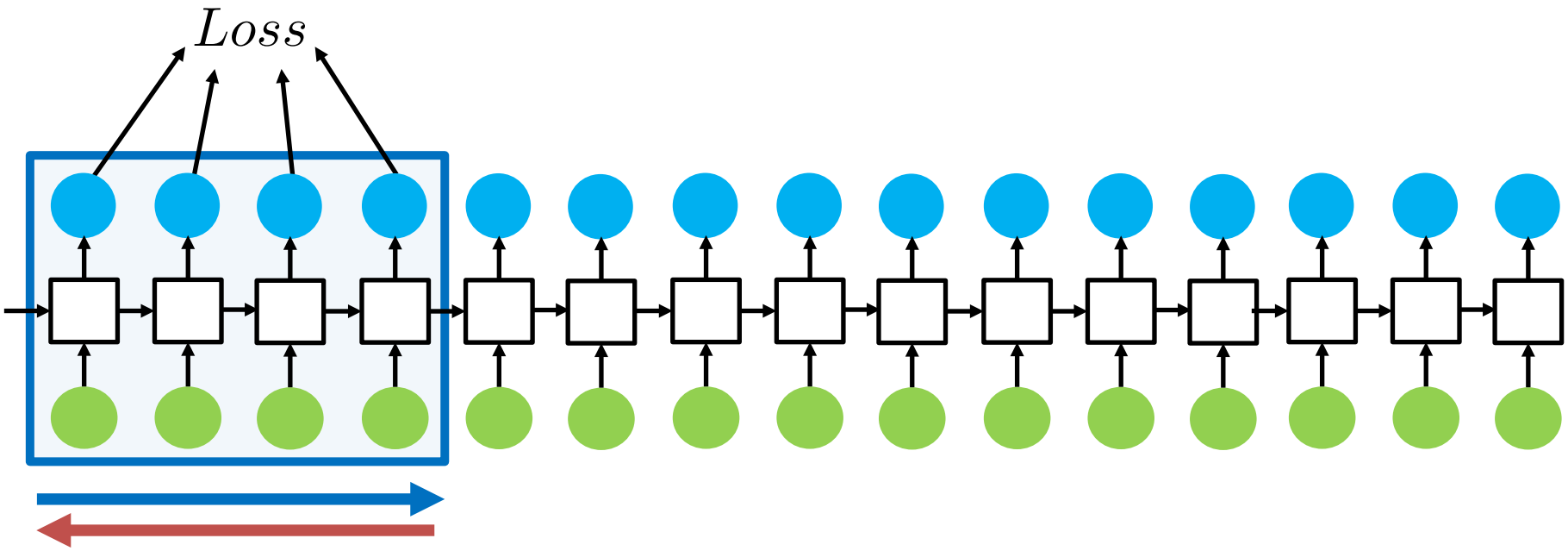
Truncated BPTT



Forward pass through the entire sequence to compute the loss

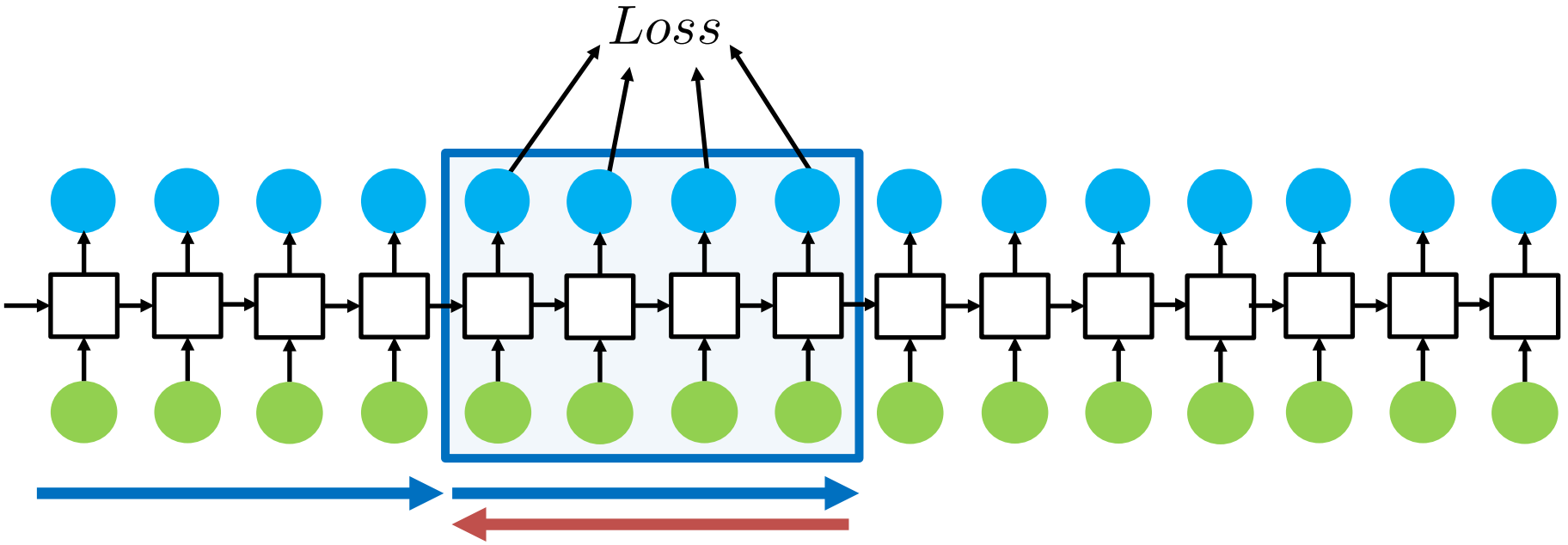
Backward pass through the entire sequence to compute the gradient

Truncated BPTT



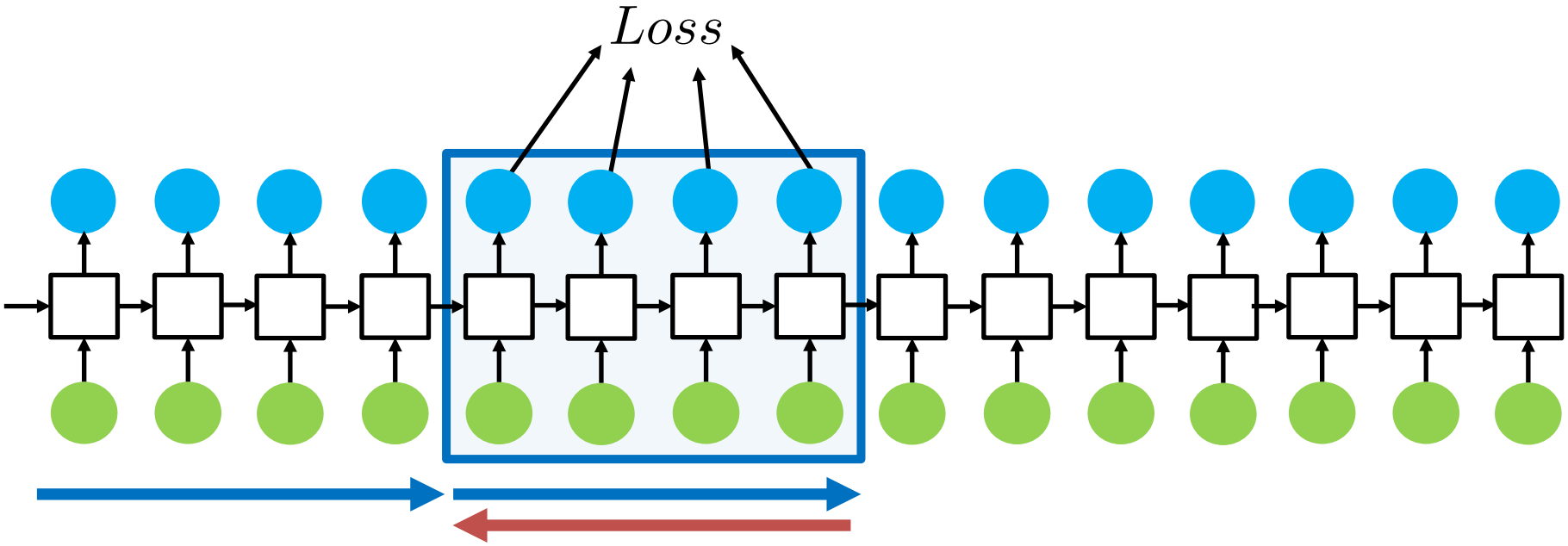
Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps.

Truncated BPTT



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps.

Truncated BPTT



Truncated BPTT is much faster but it doesn't come without a price! Dependencies longer than the chunk size don't affect the training but at least they still work at forward pass.

Vanishing gradients

- ReLU activation function
- Careful initialization
- Skip connections

- Gated models:
  LSTM (Hochreiter and Schmidhuber, 1997; Gers et al., 1999)
  GRU (Cho et al., 2014)
  …

- Orthogonal and unitary matrices in RNN (Saxe et al., 2014; Le et al., 2015; Arjovsky and Shah and Bengio, 2016; …)
- Regularization (Pascanu et al., 2012; Vorontsov et al., 2017)

- …

Careful Initialization

$$\frac{\partial L_t}{\partial W} \propto \sum_{k=0}^{t} \left(\prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial h_k}{\partial W}$$

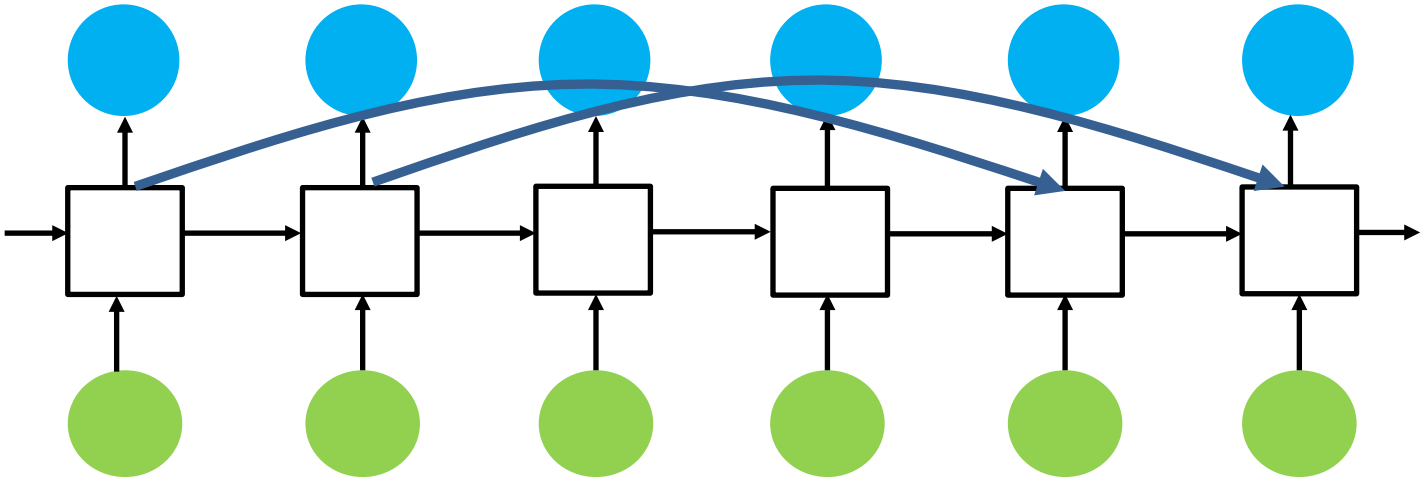
Q is orthogonal if $Q^T = Q^{-1}$ =>

$\prod_i Q_i$ doesn't explode or vanish

- Initialize W with an orthogonal matrix
- Initialize W with an identity matrix
- Use orthogonal W through the whole training

[Le et al., 2015]

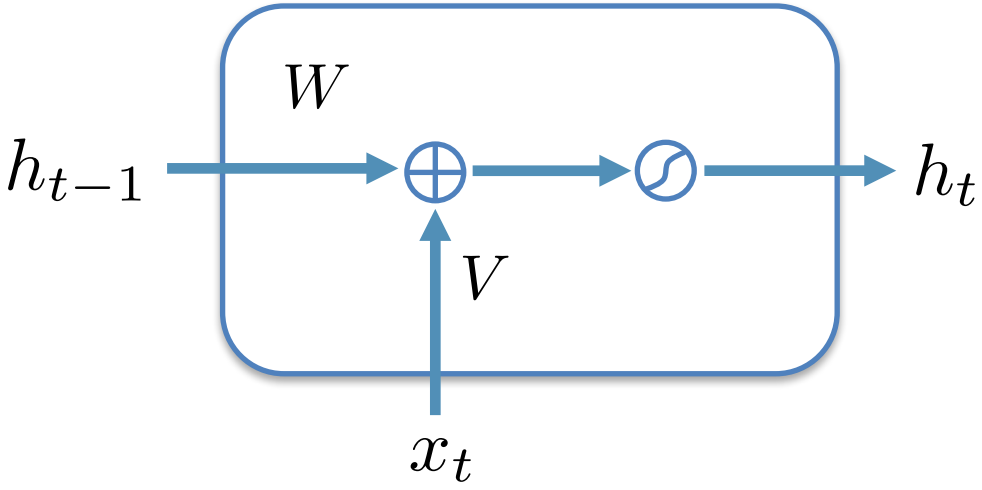
Skip connections



Add shortcuts => shorter ways for the gradients => learn longer dependencies

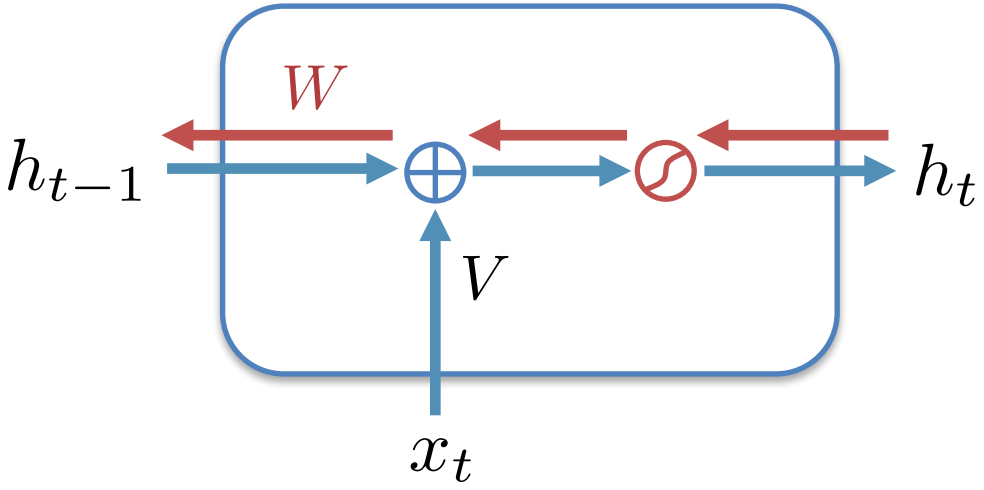
The idea is similar to the residual connections in the ResNet

Simple RNN



$$h_t = \tilde{f}(Vx_t + Wh_{t-1} + b_h)$$

Simple RNN



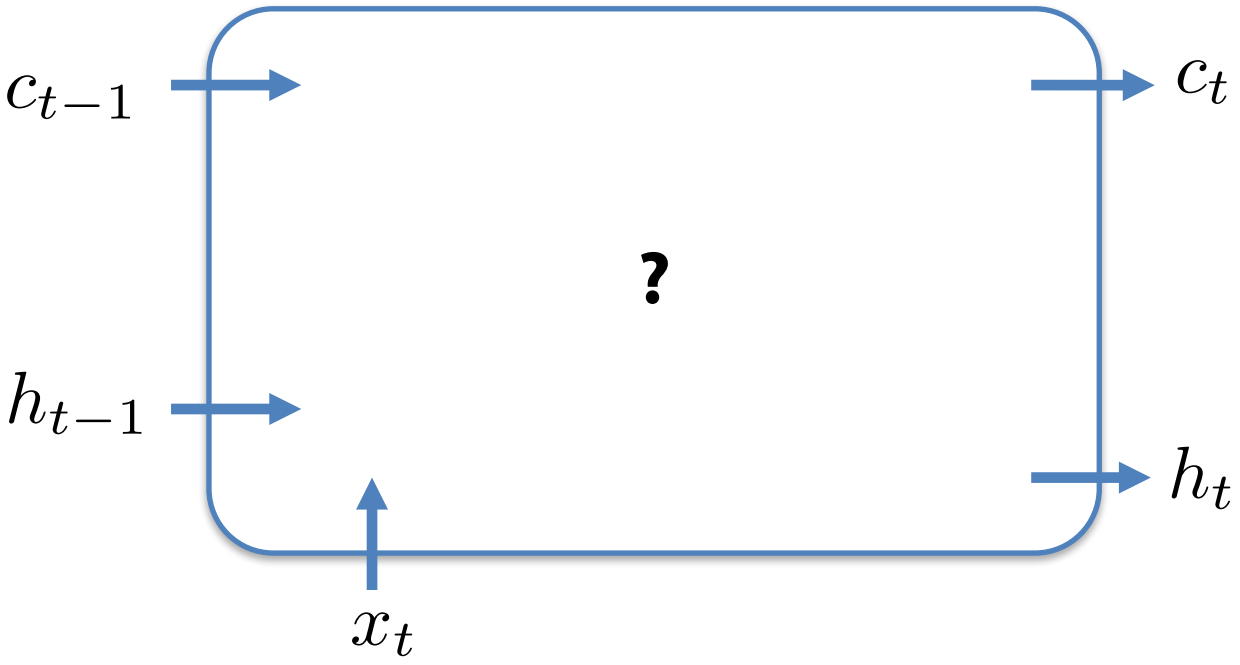
$$h_t = \tilde{f}(Vx_t + Wh_{t-1} + b_h)$$

Backward pass

W and nolinearity ➡ vanishing gradients

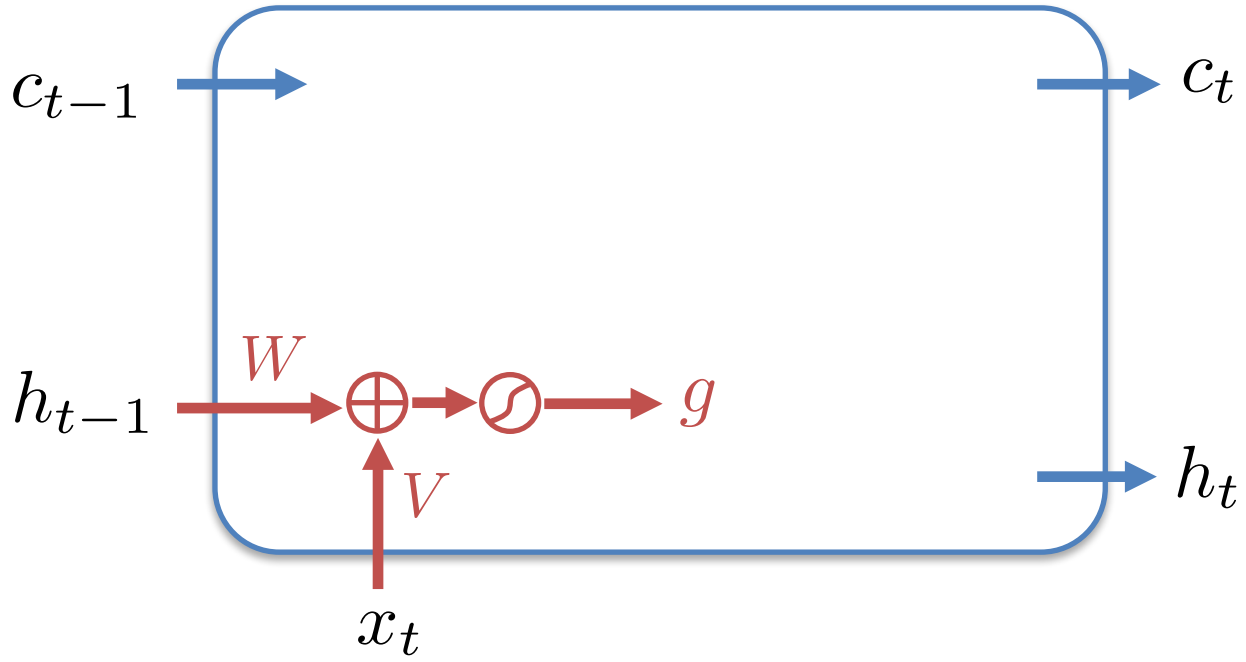
We need a short way for the gradients!

LSTM: version 0



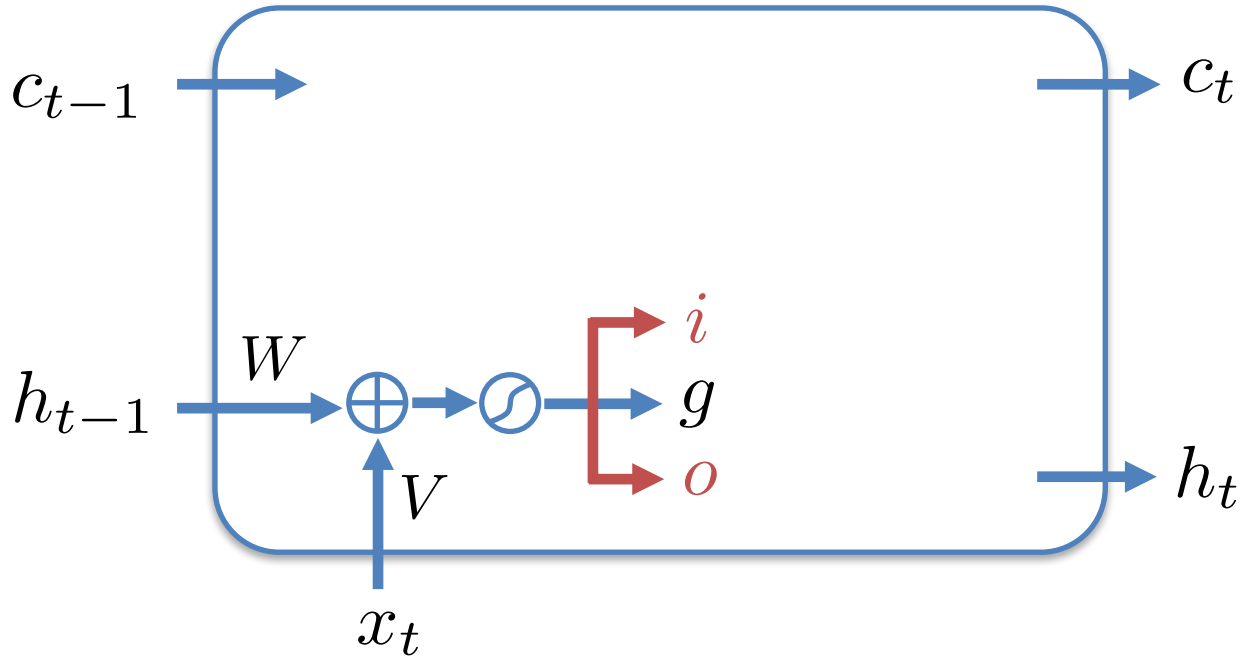
[Hochreiter, Schmidhuber, 1997]

LSTM: version 0



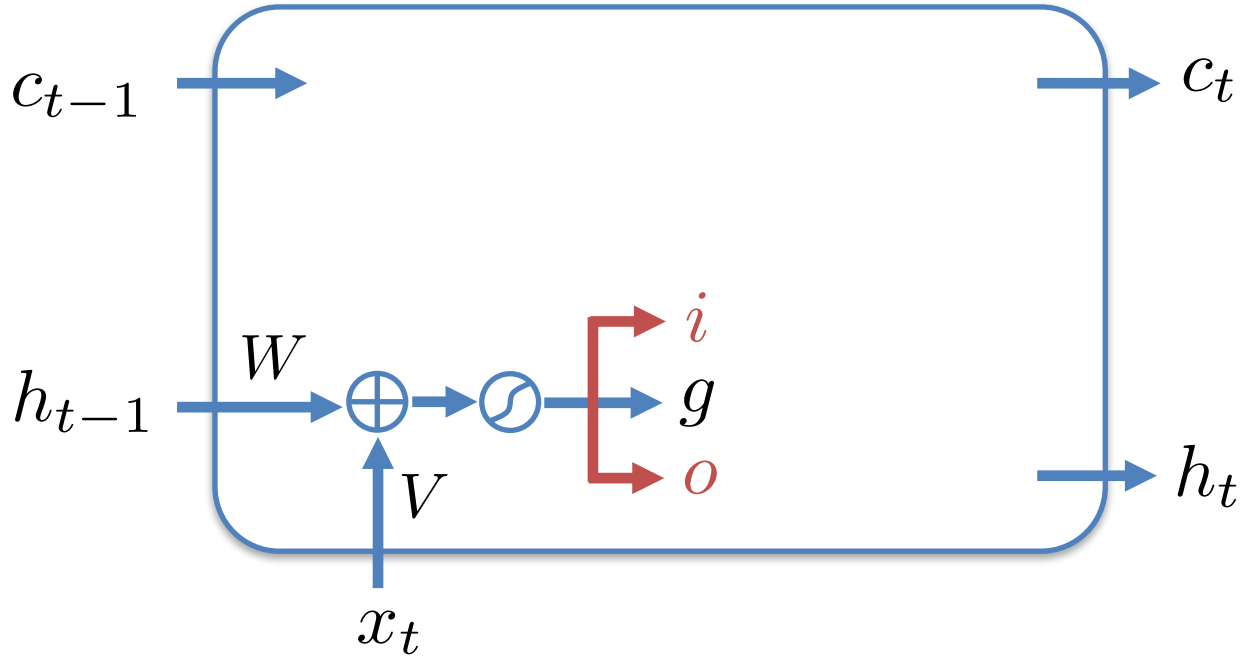
$$g_t = \tilde{f}(V_g x_t + W_g h_{t-1} + b_g)$$

LSTM: version 0



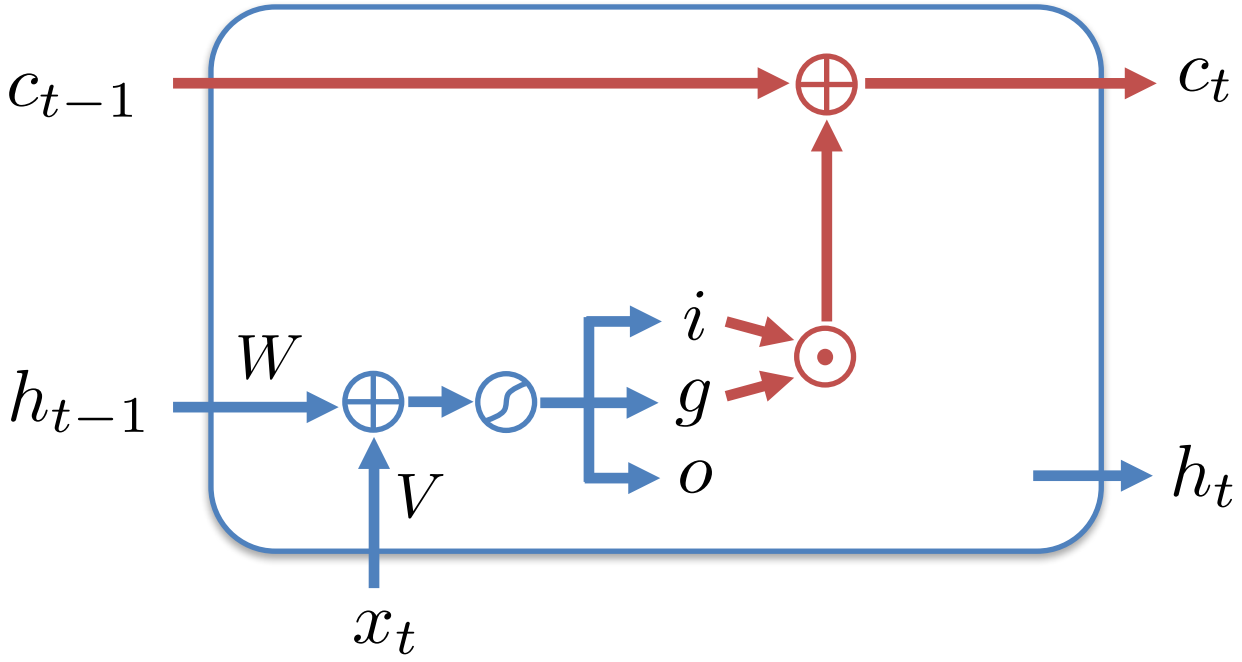
$$g_t = \tilde{f}(V_g x_t + W_g h_{t-1} + b_g)$$
$$i_t = \sigma(V_i x_t + W_i h_{t-1} + b_i)$$
$$o_t = \sigma(V_o x_t + W_o h_{t-1} + b_o)$$

LSTM: version 0



$$\begin{pmatrix} g_t \\ i_t \\ o_t \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \sigma \\ \sigma \end{pmatrix} (Vx_t + Wh_{t-1} + b)$$

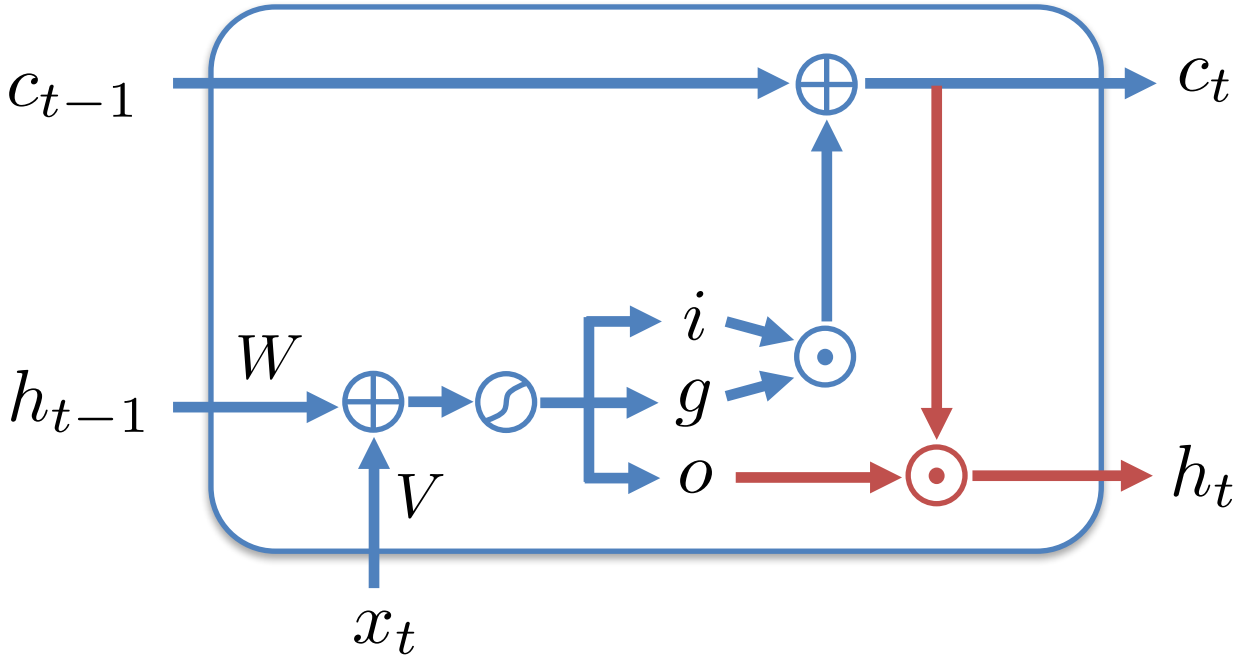
LSTM: version 0



$$\begin{pmatrix} g_t \\ i_t \\ o_t \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \sigma \\ \sigma \end{pmatrix} (Vx_t + Wh_{t-1} + b)$$

$$c_t = c_{t-1} + i_t \cdot g_t$$

LSTM: version 0

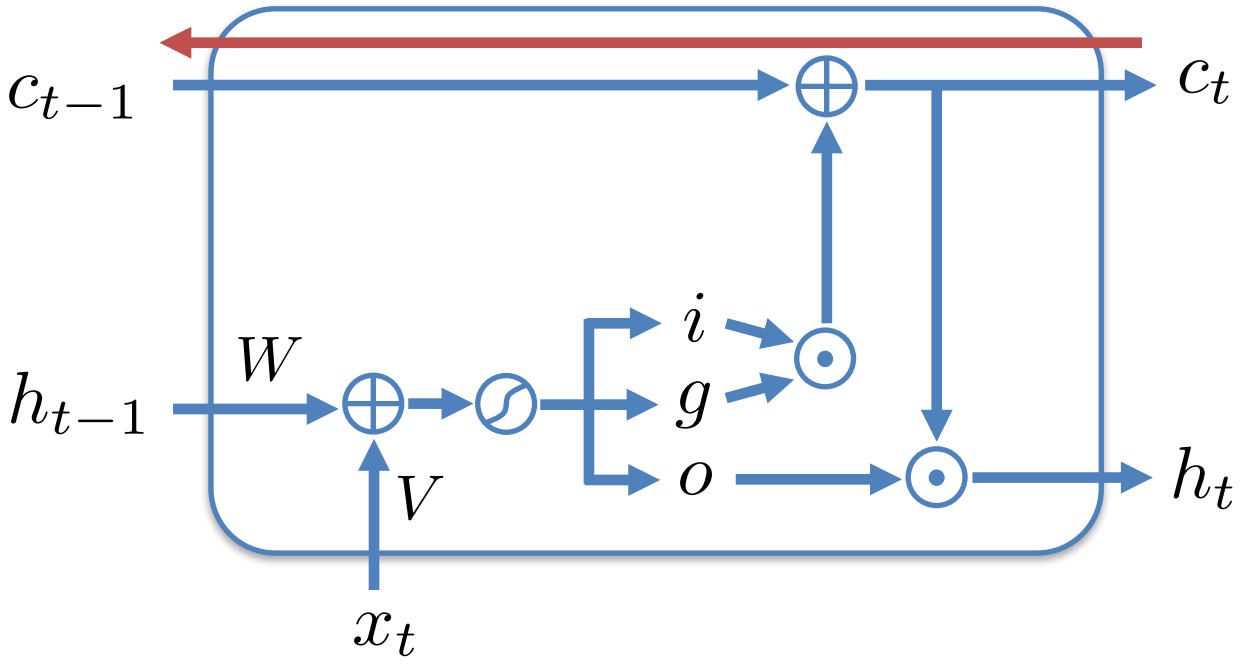


$$\begin{pmatrix} g_t \\ i_t \\ o_t \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \sigma \\ \sigma \end{pmatrix} (Vx_t + Wh_{t-1} + b)$$

$$c_t = c_{t-1} + i_t \cdot g_t$$

$$h_t = o_t \cdot \tilde{f}(c_t)$$

LSTM: version 0

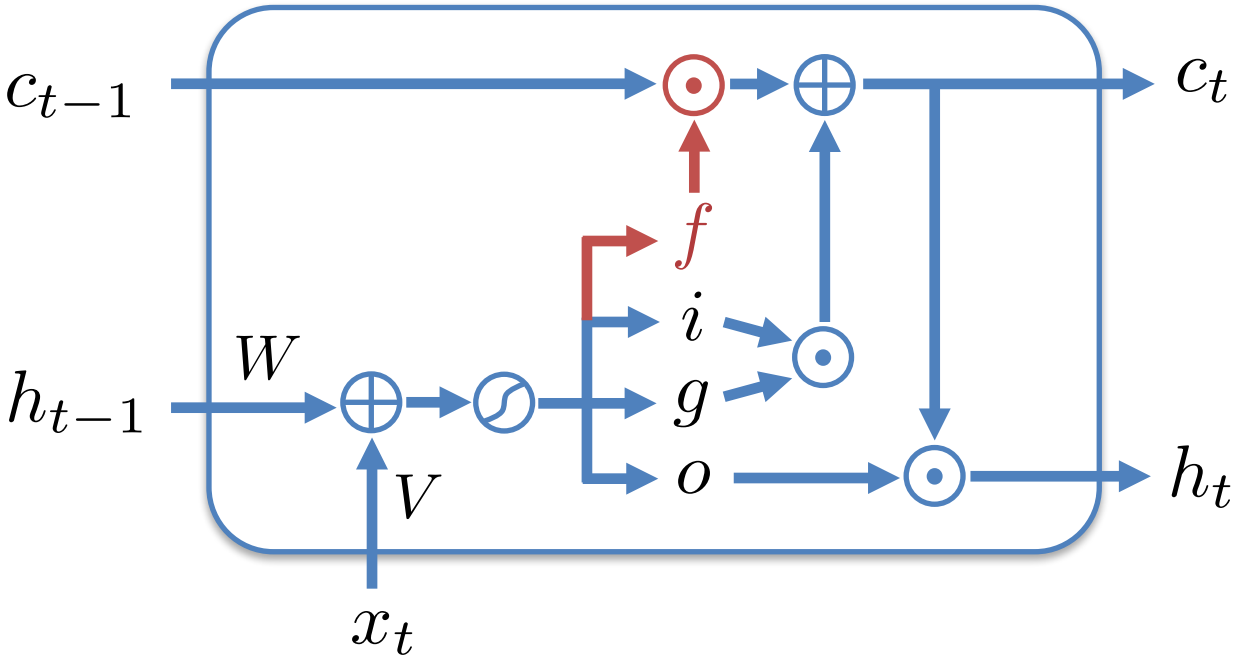


$$c_t = c_{t-1} + i_t \cdot g_t$$

$$\frac{\partial h_t}{\partial h_{t-1}} \Rightarrow \frac{\partial c_t}{\partial c_{t-1}} = 1$$

Gradients do not vanish!

LSTM: forget sometimes

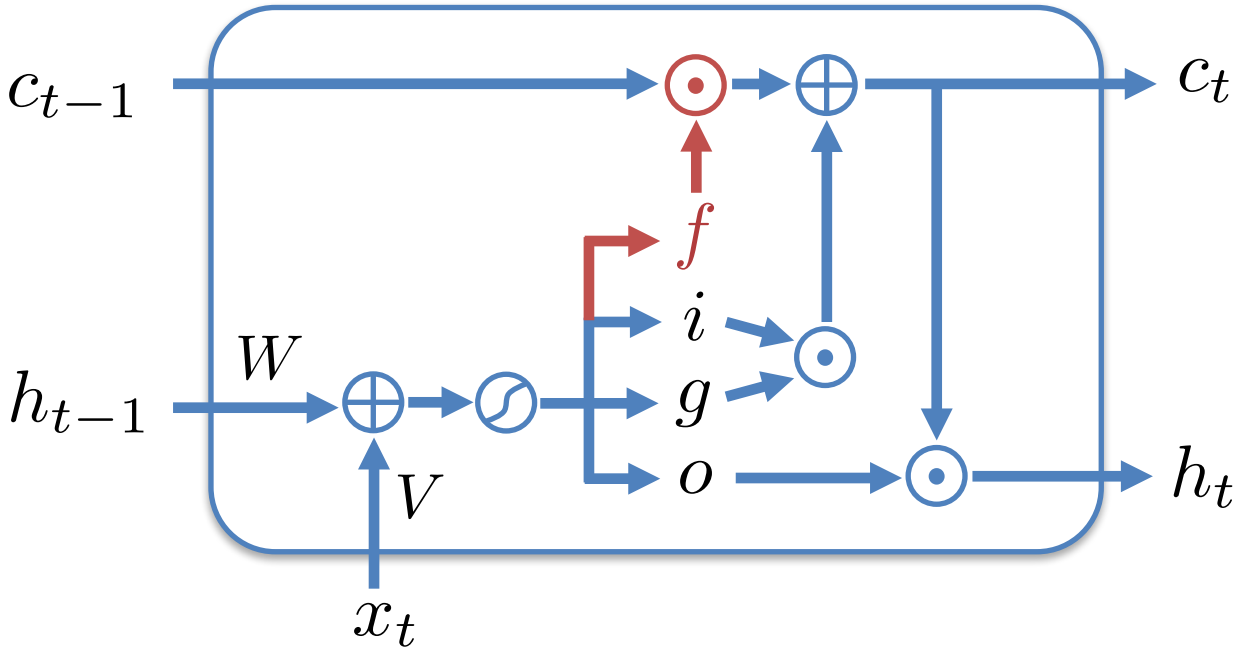


$$\begin{pmatrix} g_t \\ i_t \\ o_t \\ f_t \end{pmatrix} = \begin{pmatrix} \tilde{f} \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} (Vx_t + Wh_{t-1} + b)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t$$
$$h_t = o_t \cdot \tilde{f}(c_t)$$

[Gers et al., 1999]

LSTM: forget sometimes



$$f_t = \sigma(V_f x_t + W_f h_{t-1} + b_f)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t$$

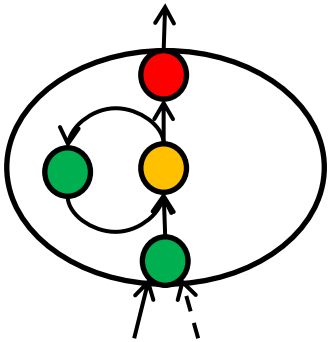
$$\frac{\partial c_t}{\partial c_{t-1}} = f_t$$

High initial b_f

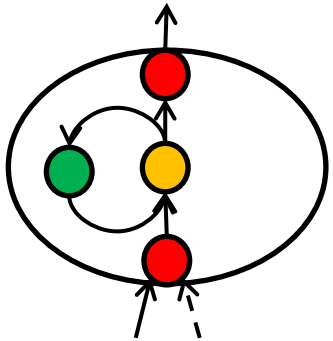
Long short term memory:

Examples

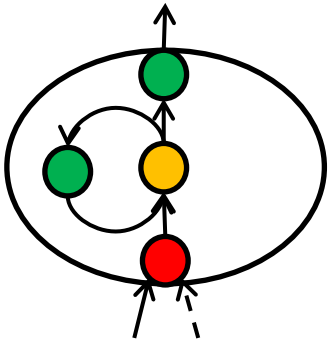
Captures info



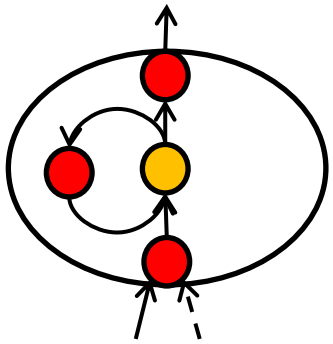
Keeps info



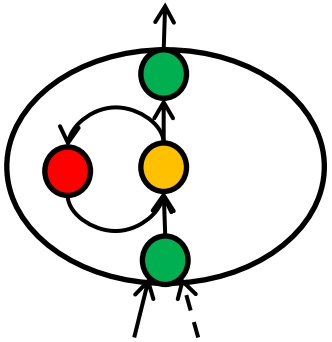
Releases info



Erases info



= RNN



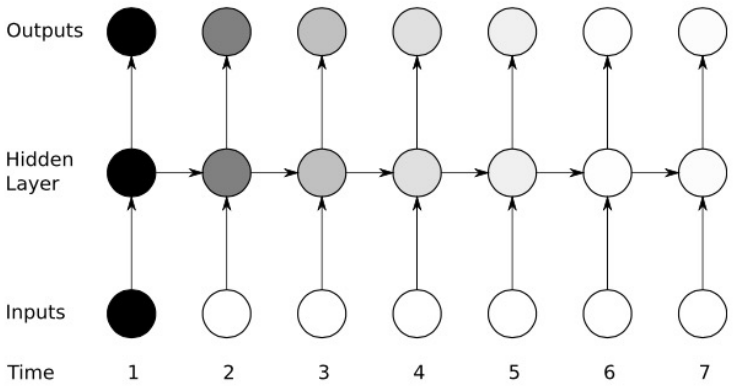
- gate is close

- gate is open

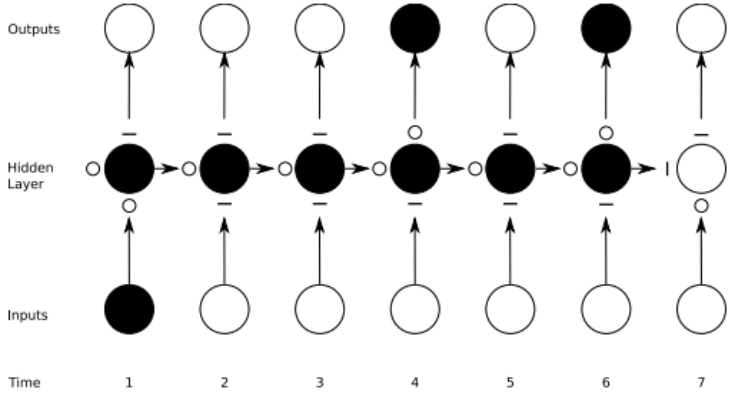
Long short term memory:

Examples

RNN



LSTM



– - gate is close

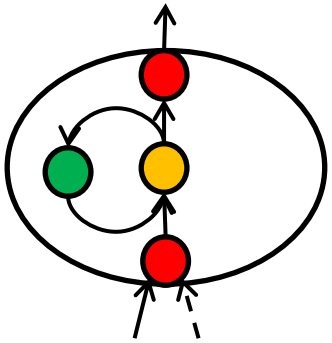
○ - gate is open

[Graves, 2012]

Long short term memory:

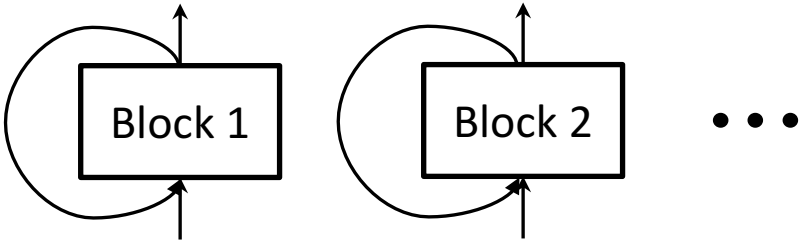
Initialization

LSTM cell:



$$b_i = -a < 0$$
$$b_f = a > 0$$
$$b_o = -a < 0$$

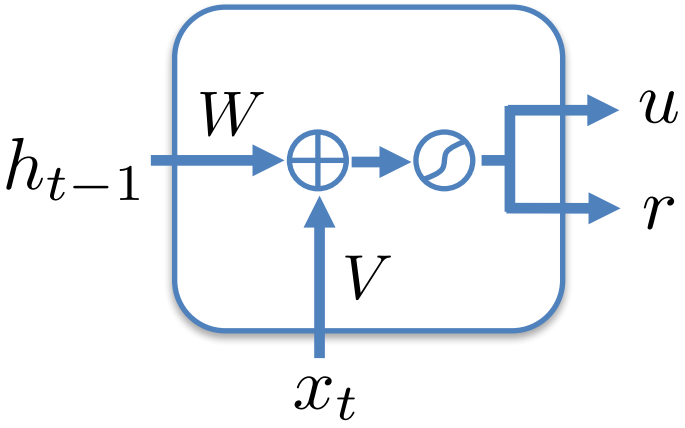
LSTM layer:

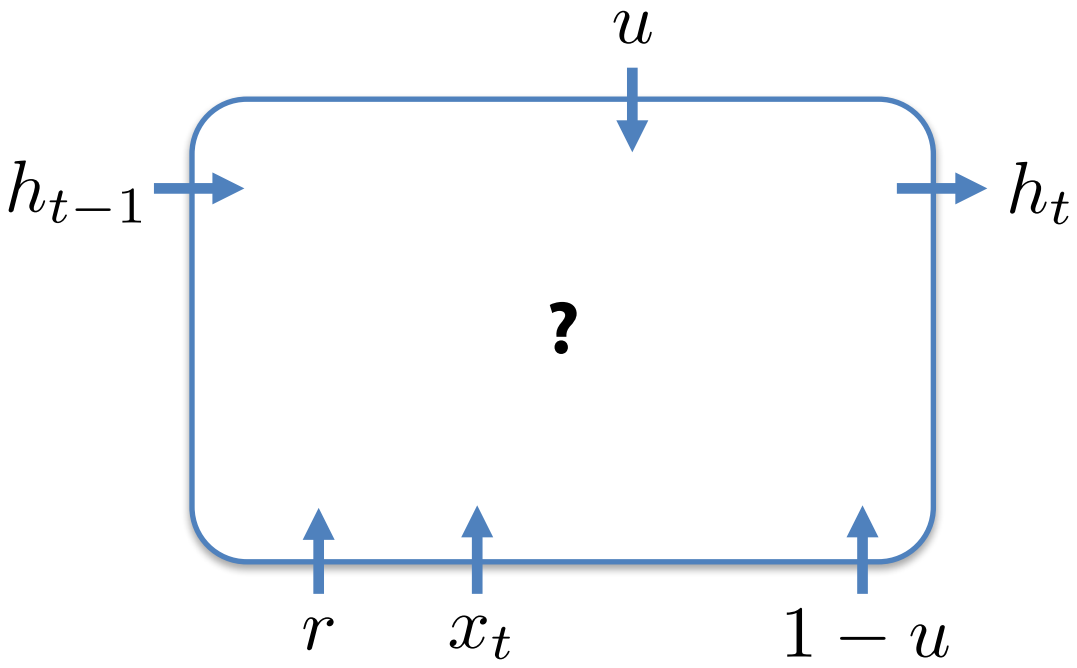


$a = 0.5$ $a = 1$

- Blocks start work one by one
- Blocks capture dependencies with different ranges
- We learn to forget only if it necessary

GRU

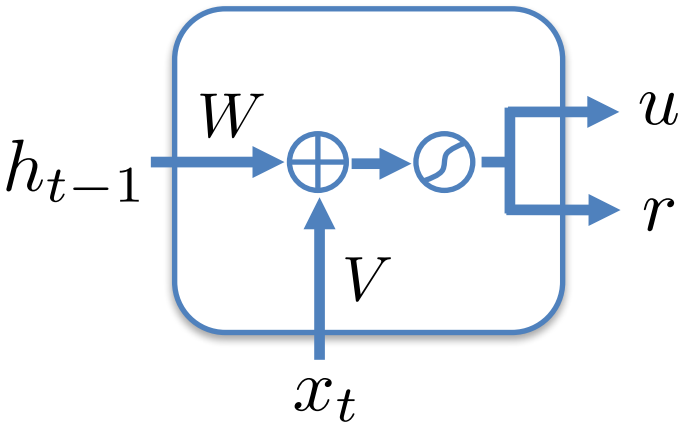


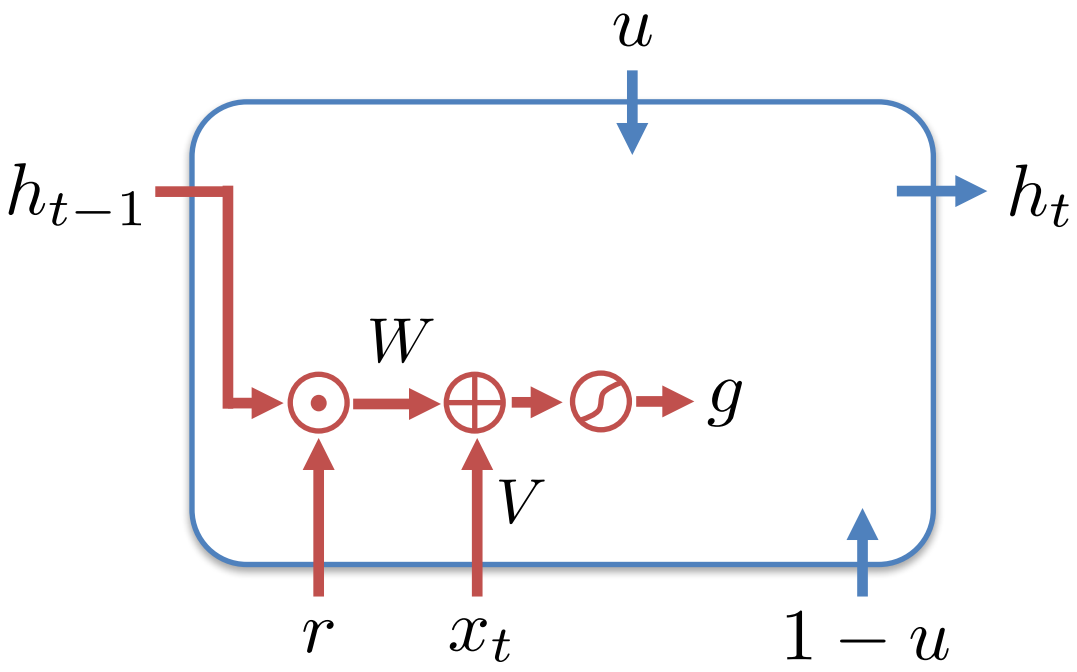


$$\begin{pmatrix} r_t \\ u_t \end{pmatrix} = \sigma(Vx_t + Wh_{t-1} + b)$$

[Cho et al., 2014]

GRU

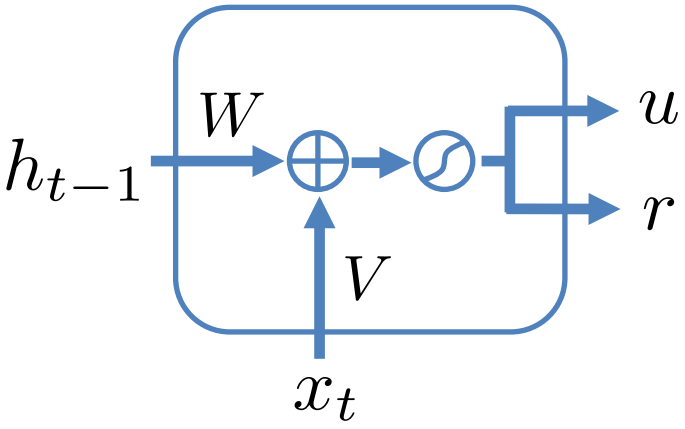


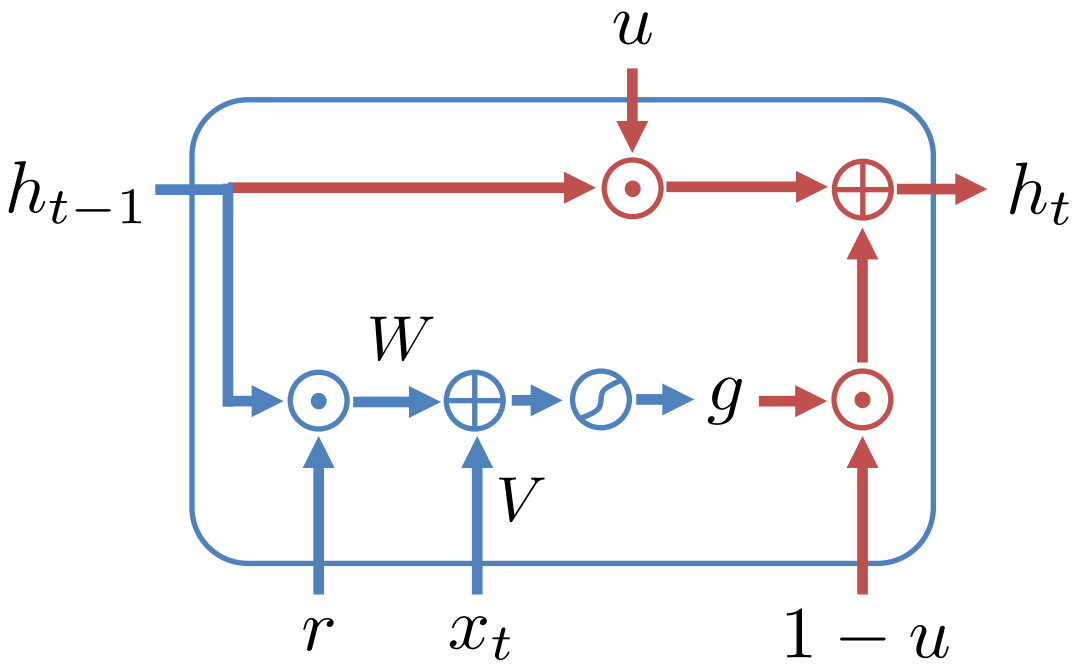


$$\begin{pmatrix} r_t \\ u_t \end{pmatrix} = \sigma(Vx_t + Wh_{t-1} + b)$$

$$g_t = \tilde{f}\left(V_g x_t + W_g(h_{t-1} \cdot r_t) + b_g\right)$$

GRU



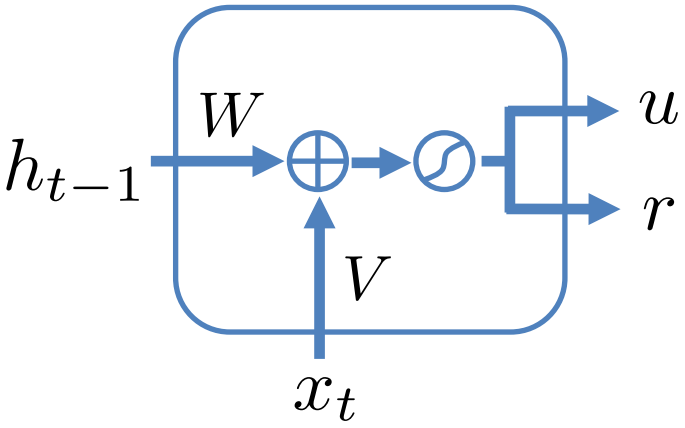


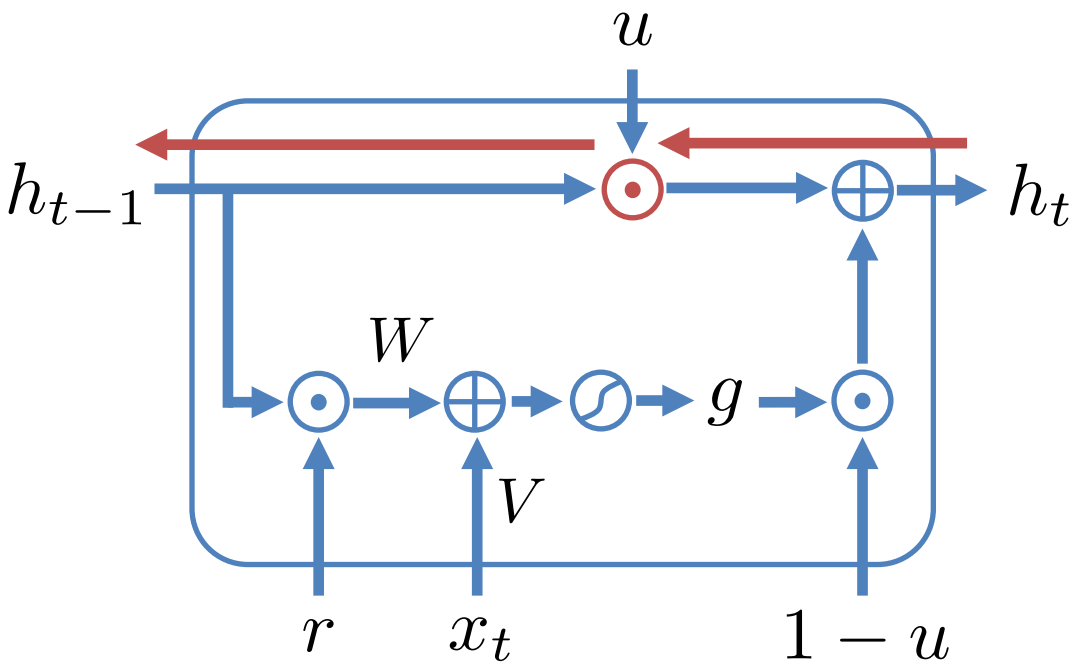
$$\begin{pmatrix} r_t \\ u_t \end{pmatrix} = \sigma(Vx_t + Wh_{t-1} + b)$$

$$g_t = \tilde{f}\left(V_g x_t + W_g(h_{t-1} \cdot r_t) + b_g\right)$$

$$h_t = (1 - u_t) \cdot g_t + u_t \cdot h_{t-1}$$

GRU





$$u_t = \sigma(V_u x_t + W_u h_{t-1} + b_u) \qquad h_t = (1 - u_t) \cdot g_t + u_t \cdot h_{t-1}$$

$$\frac{\partial h_t}{\partial h_{t-1}} = u_h + (1 - u_h) \cdot \frac{\partial g_h}{\partial h_{h-1}}$$

High initial b_u

Orthogonal and unitary matrices

$$\frac{\partial L_t}{\partial W} = \frac{\partial L_t}{\partial \hat{y}_t}\frac{\partial \hat{y}_t}{\partial h_t}\sum_{k=0}^{t}\left(\prod_{i=k+1}^{t}\frac{\partial h_i}{\partial h_{i-1}}\right)\frac{\partial h_k}{\partial W}$$

$$\left\|\frac{\partial h_k}{\partial W}\prod_{k+1}^{t} diag(f_h'(pr_t))\cdot W\right\| \leq \left\|\frac{\partial h_k}{\partial W}\right\|\prod_{k+1}^{t}\|D\cdot W\| =$$

W ⟹ Orthogonal or unitary ⟹ $= \left\|\frac{\partial h_k}{\partial W}\right\|\prod_{k+1}^{t}\|D\| =$

D ⟹ ReLU ⟹ $= \left\|\frac{\partial h_k}{\partial W}\right\|$

uRNN

$$W = D_3 R_2 F^{-1} D_2 \Pi R_1 F D_1$$

- $\mathbf{D}$, a diagonal matrix with $\mathbf{D}_{j,j} = e^{iw_j}$, with parameters $w_j \in \mathbb{R}$,
- $\mathbf{R} = \mathbf{I} - 2\frac{vv^*}{\|v\|^2}$, a reflection matrix in the complex vector $v \in \mathbb{C}^n$,
- $\mathbf{\Pi}$, a fixed random index permutation matrix, and
- $\mathcal{F}$ and $\mathcal{F}^{-1}$, the Fourier and inverse Fourier transforms.

Complex:

- hidden units,
- in-to-hidden
- hidden-to-hidden

$$\Rightarrow \quad o_t = f\left(U \begin{pmatrix} Re(h_t) \\ Im(h_t) \end{pmatrix} + b_o\right)$$

$$modReLU(z) = \begin{cases} (|z| + b)\frac{z}{|z|} & if\ |z| + b \geq 0 \\ 0 & if\ |z| + b < 0 \end{cases}$$

[Arjovsky et al., 2016]

uRNN

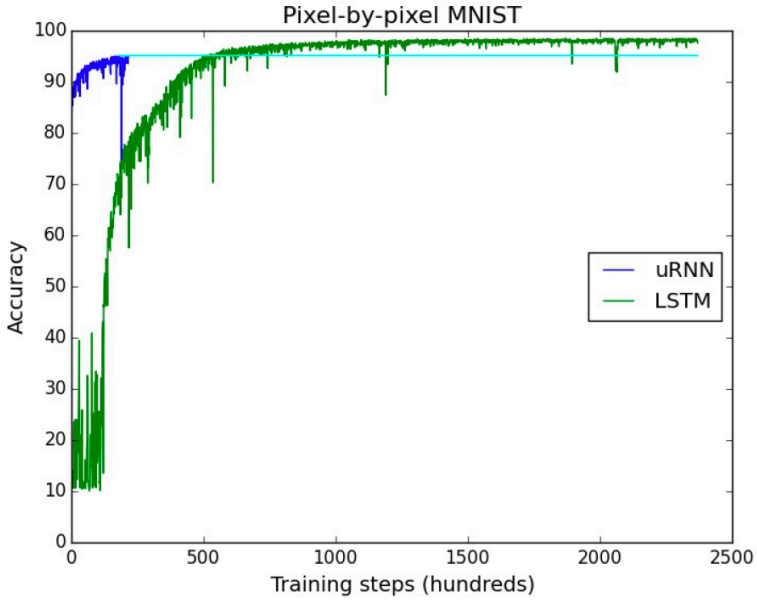
Pros:

- No (almost) vanishing or exploding gradients
- Memory: $O(n)$, time: $O(n\ logn)$
- Good parametrization: $O(n)$ parameters $\rightarrow$ more hidden units
- Very long dependencies

Cons:

- LSTM has stronger local dependencies

uRNN



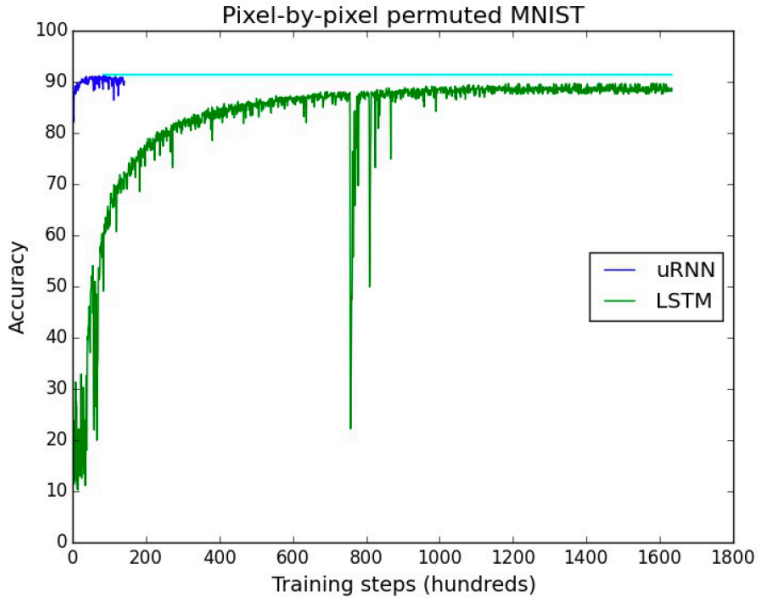


Figure 3. Results on pixel by pixel MNIST classification tasks. The uRNN is able to converge in a fraction of the iterations that the LSTM requires. The LSTM performs better on MNIST classification, but the uRNN outperforms on the more complicated task of permuted pixels.

Orthogonal and unitary matrices

Regularization 1:

$$\Omega = \|W^T W - I\|^2$$

[Vorontsov et al., 2017]

Regularization 2:

$$\Omega = \sum_t \Omega_t = \sum_t \left(\frac{\left\| \frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \right\|}{\left\| \frac{\partial L}{\partial h_t} \right\|} - 1 \right)^2$$

[Pascanu et al., 2012]

Gated unitary models

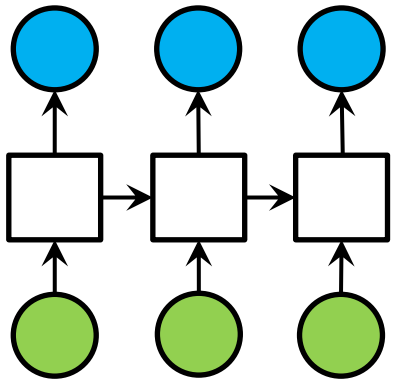
[Jing et al., 2017]

Examples

Classification of sequence elements

Synced sequence input and output:

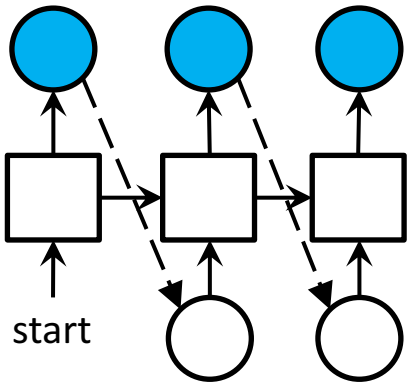
- POS tagging
- Video frames classification



Text generation

Next symbol/word

Current symbol/word



Text generation

PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.

Andrej Karpathy blog

Text generation

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma **??**. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{array}{ccc} \mathcal{S} & \longrightarrow & \\ \downarrow & & \\ \xi & \longrightarrow & \mathcal{O}_{X'} \\ & & \uparrow \\ \mathrm{gor}_s & & \end{array} \qquad \begin{array}{ccc} & = \alpha' & \longrightarrow \\ & \updownarrow & \\ = \alpha' & \longrightarrow & \alpha \end{array} \qquad \begin{array}{c} X \\ \downarrow \\ \mathrm{d}(\mathcal{O}_{\mathcal{X}_{X/k}}, \mathcal{G}) \end{array}$$

$$\mathrm{Spec}(K_\psi) \qquad \mathrm{Mor}_{Sets}$$

is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas **??**.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\overline{x}} \quad \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_t}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition **??** and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma **??**. This is a sequence of $\mathcal{F}$ is a similar morphism.

[Andrej Karpathy blog]

Text generation

Cell sensitive to position in line:

```
The sole importance of the crossing of the Berezina lies in the fact
that it plainly and indubitably proved the fallacy of all the plans for
cutting off the enemy's retreat and the soundness of the only possible
line of action--the one Kutuzov and the general mass of the army
demanded--namely, simply to follow the enemy up. The French crowd fled
at a continually increasing speed and all its energy was directed to
reaching its goal. It fled like a wounded animal and it was impossible
to block its path. This was shown not so much by the arrangements it
made for crossing as by what took place at the bridges. When the bridges
broke down, unarmed soldiers, people from Moscow and women with children
who were with the French transport, all--carried on by vis inertiae--
pressed forward into boats and into the ice-covered water and did not,
surrender.
```

Cell that turns on inside quotes:

```
"You mean to imply that I have nothing to eat out of.... On the
contrary, I can supply you with everything even if you want to give
dinner parties," warmly replied Chichagov, who tried by every word he
spoke to prove his own rectitude and therefore imagined Kutuzov to be
animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating
smile: "I meant merely to say what I said."
```

Cell that robustly activates inside if statements:

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

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

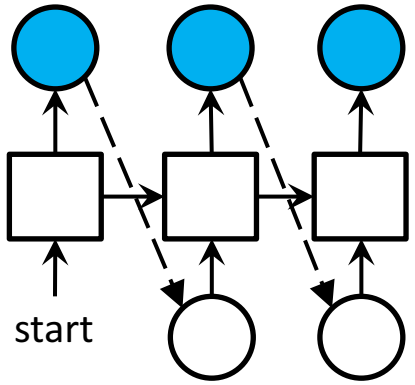
```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
  char *str;
  if (!*bufp || (len == 0) || (len > *remain))
    return ERR_PTR(-EINVAL);
  /* Of the currently implemented string fields, PATH_MAX
 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

[Andrej Karpathy blog]

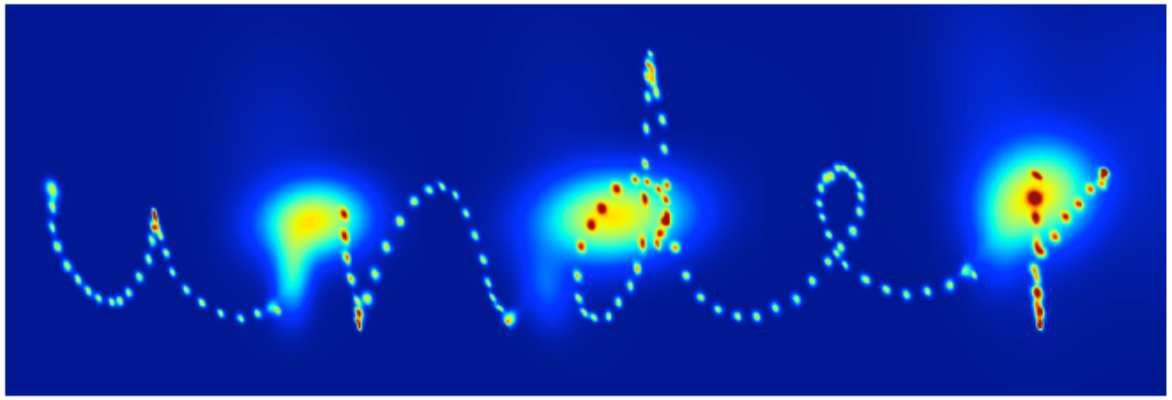
Handwriting generation:

handwriting -> handwriting

Next pen position (we predict parameters):
x1, x2 - mixture of bivariate Gaussians
x3 - Bernoulli distribution



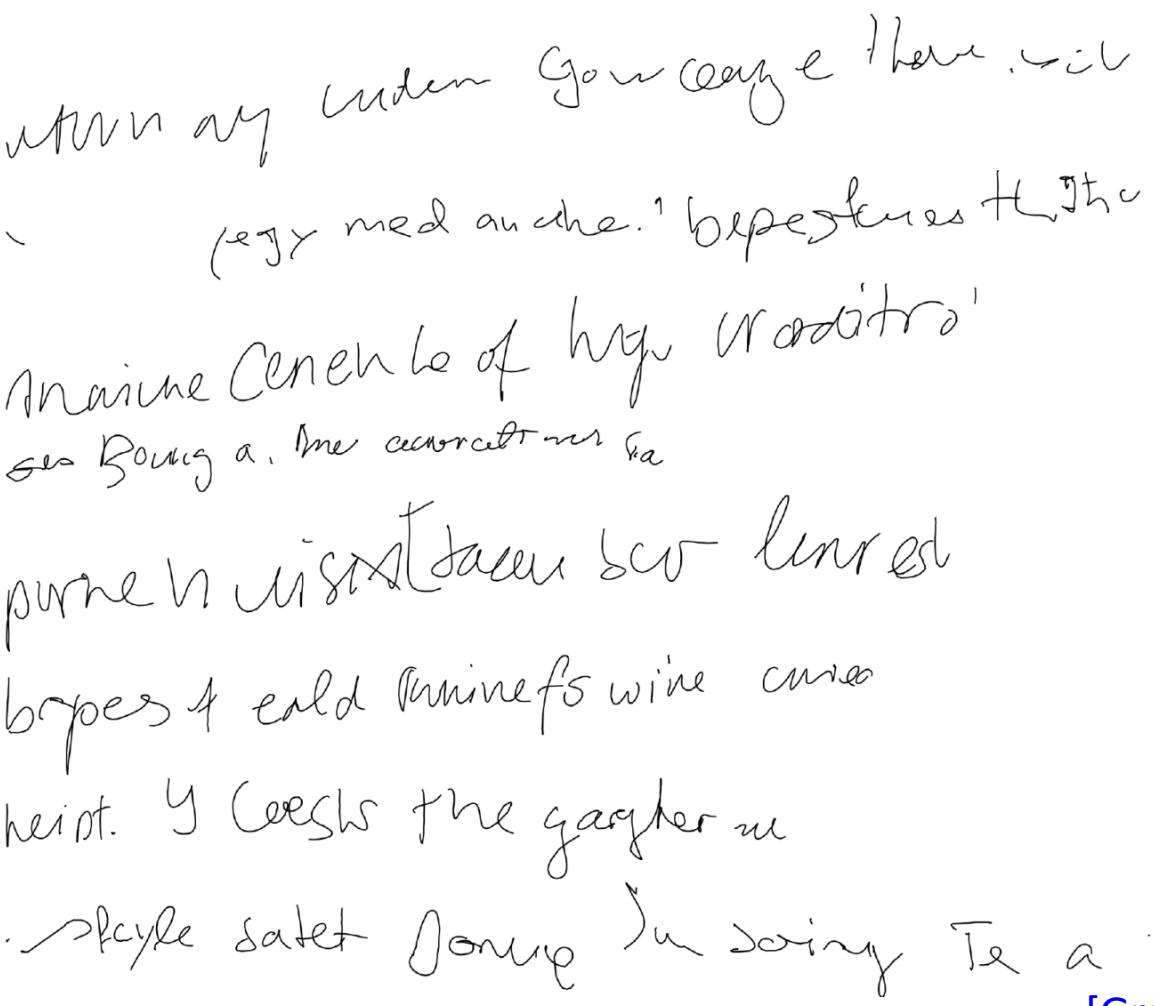
Current pen position:
x1, x2 – pen offset
x3 – is it end of the stroke



[Graves,2014]

Handwriting generation:

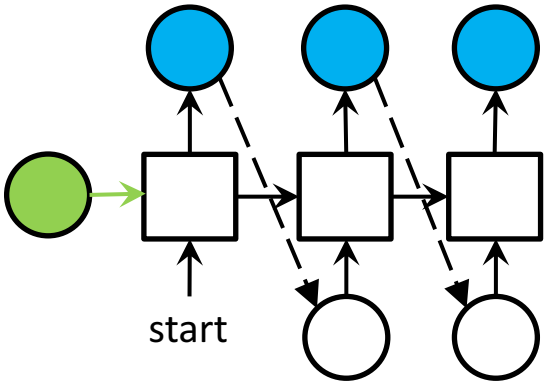
example

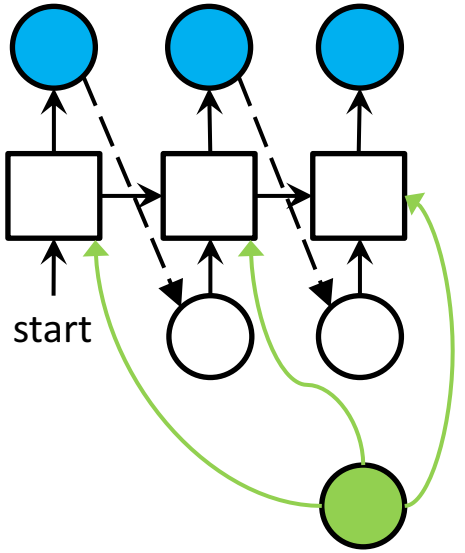


[Graves,2014]

Sequence generation

- Handwriting synthesis
- Image captioning





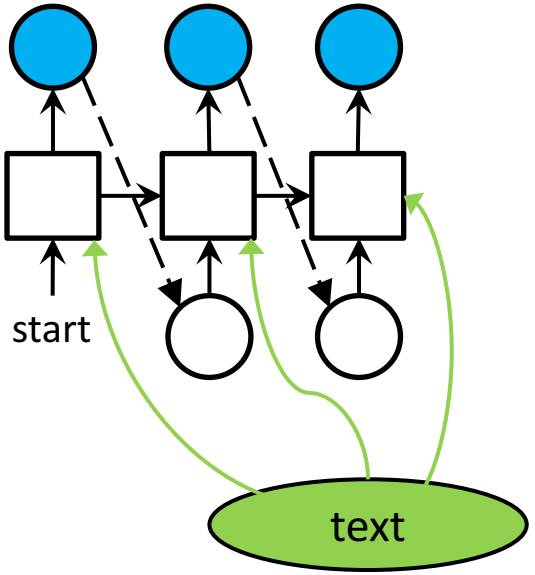
Handwriting synthesis:

text -> handwriting

Next pen position

Current pen position

Which letter we write now



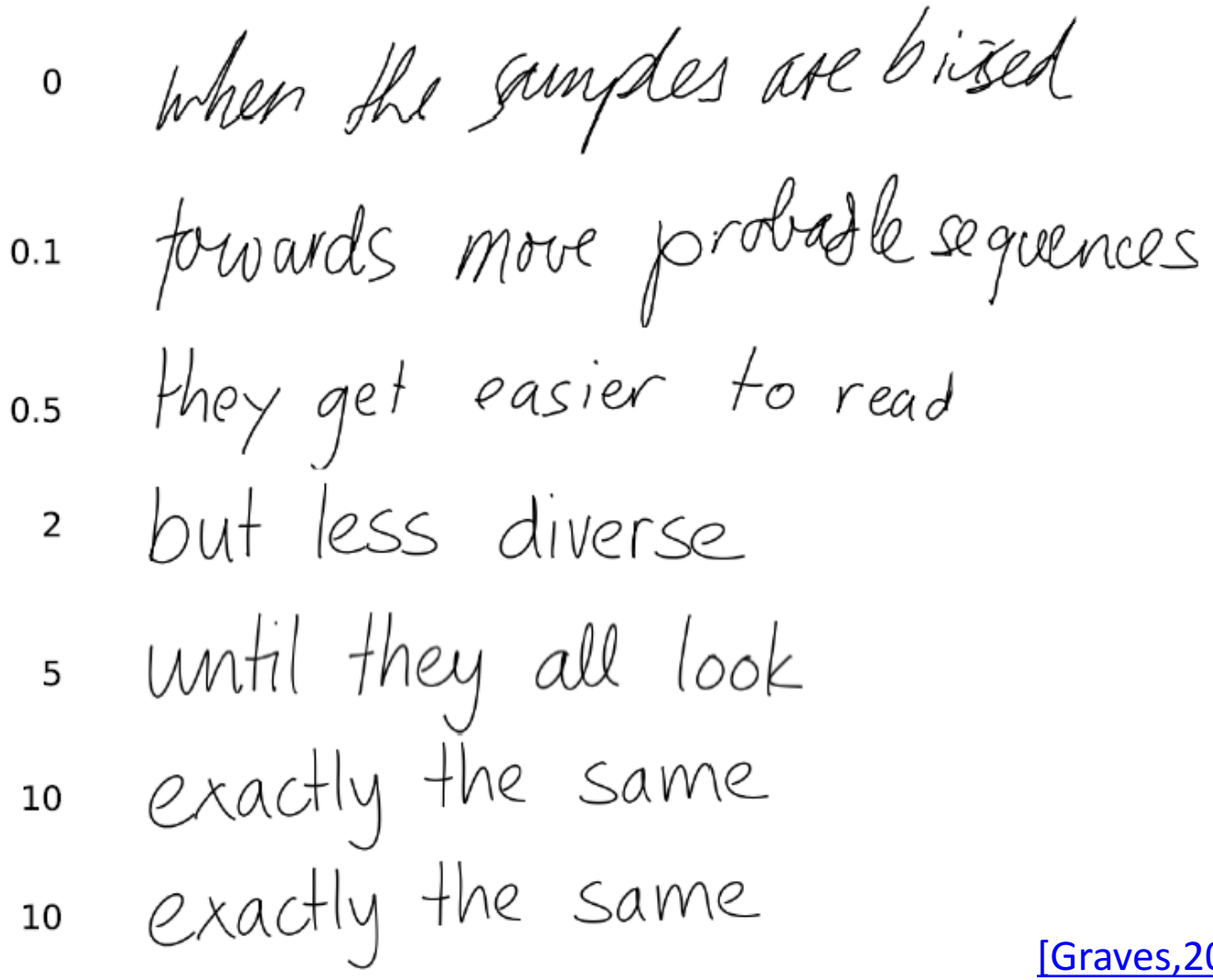
Demo

[Graves,2014]

Handwriting synthesis:

biased sampling

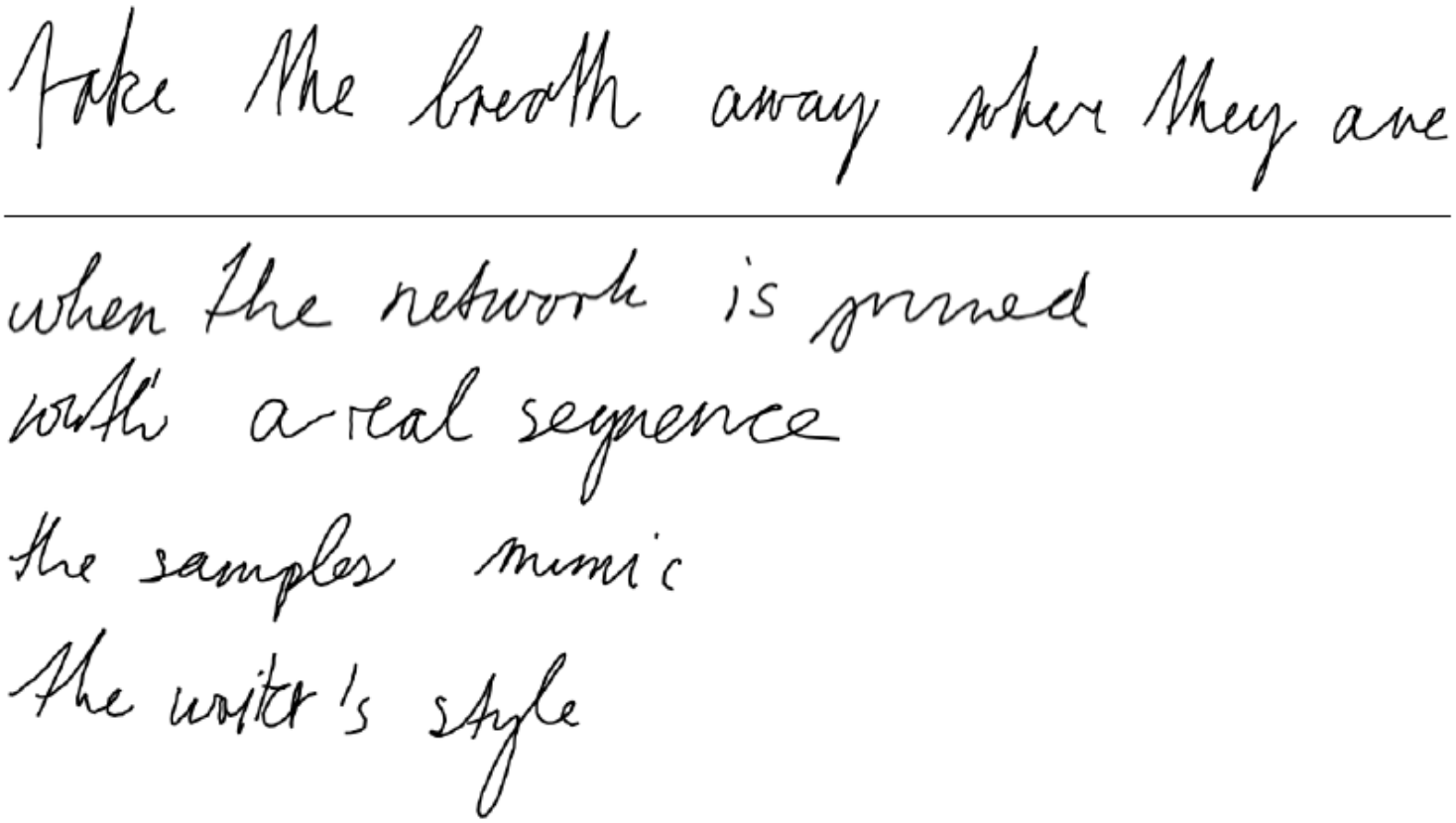
bias



[Graves,2014]

Handwriting synthesis:

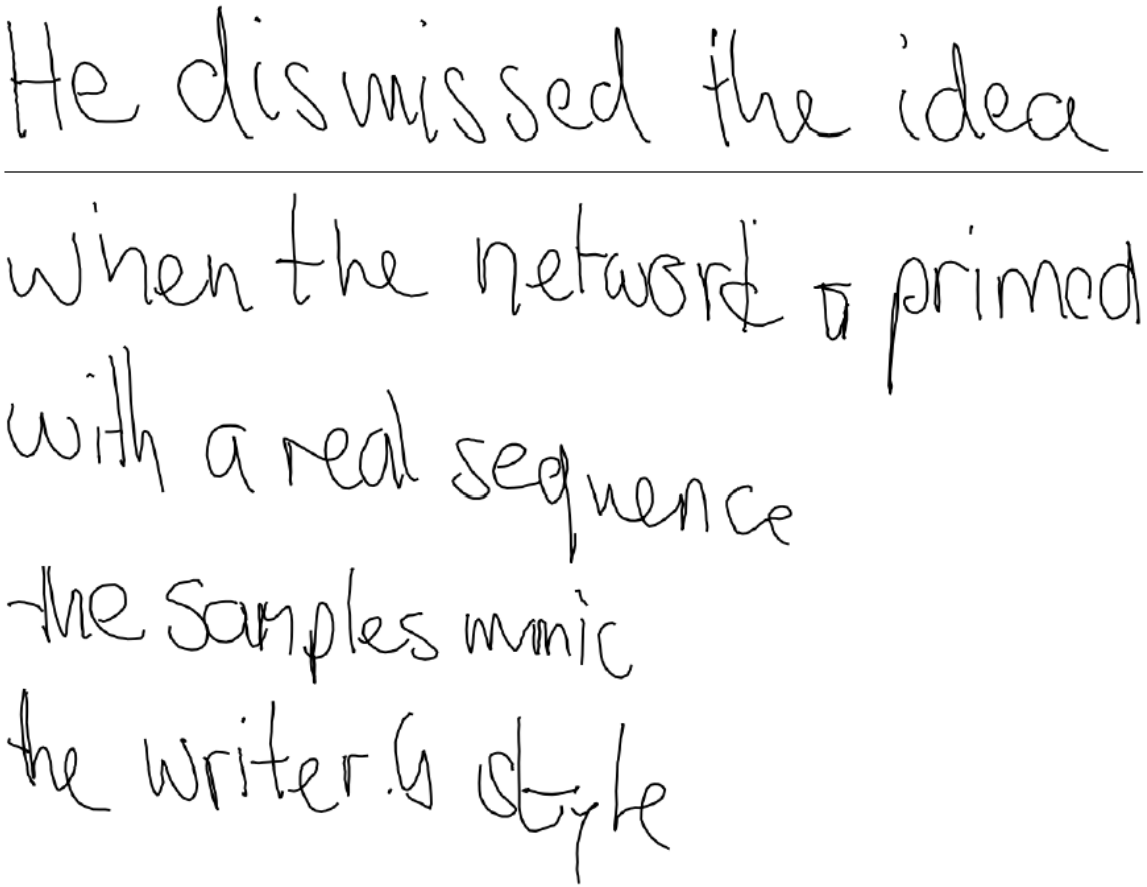
primed sampling



[Graves,2014]

Handwriting synthesis:

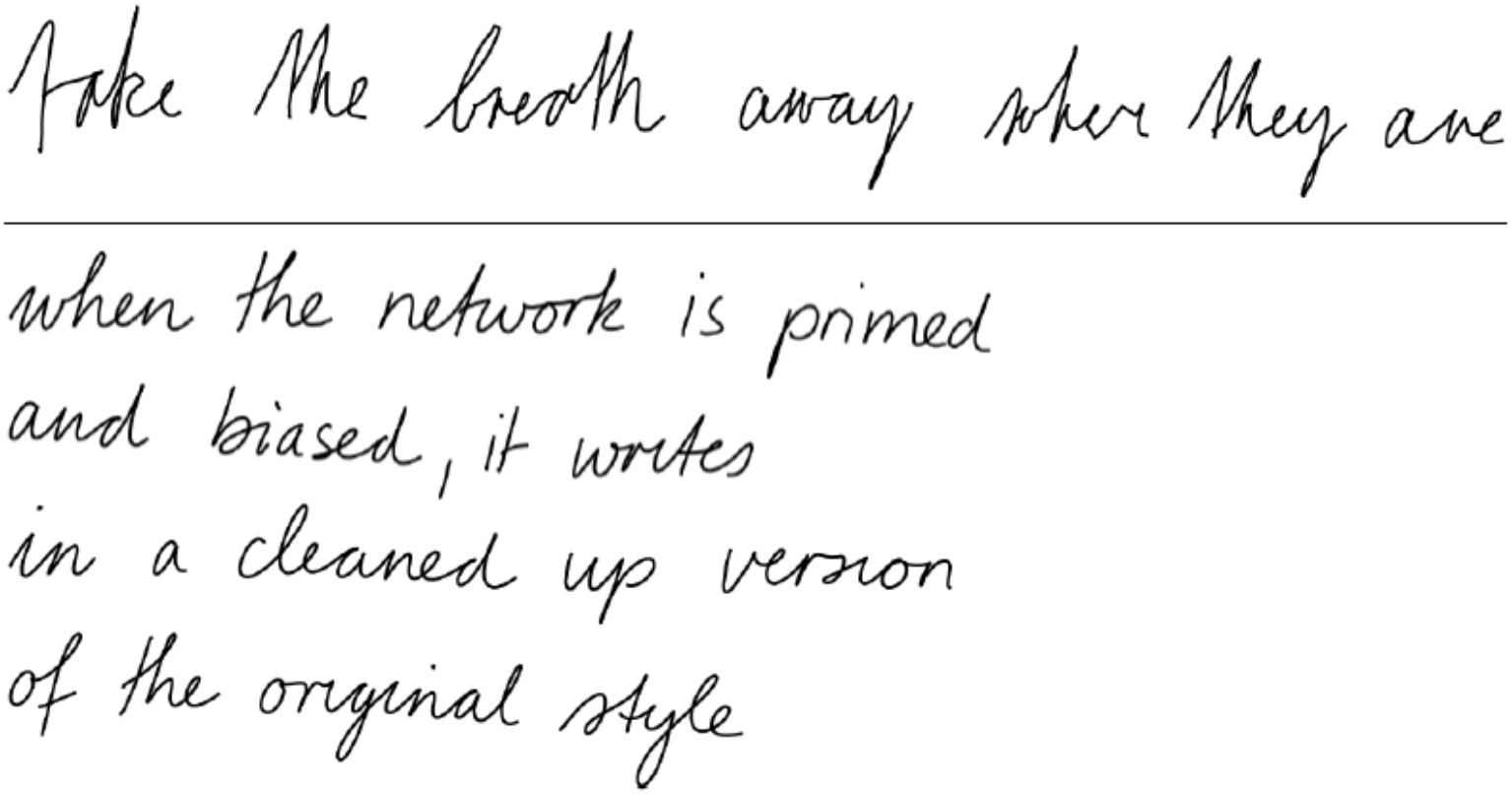
primed sampling



[Graves,2014]

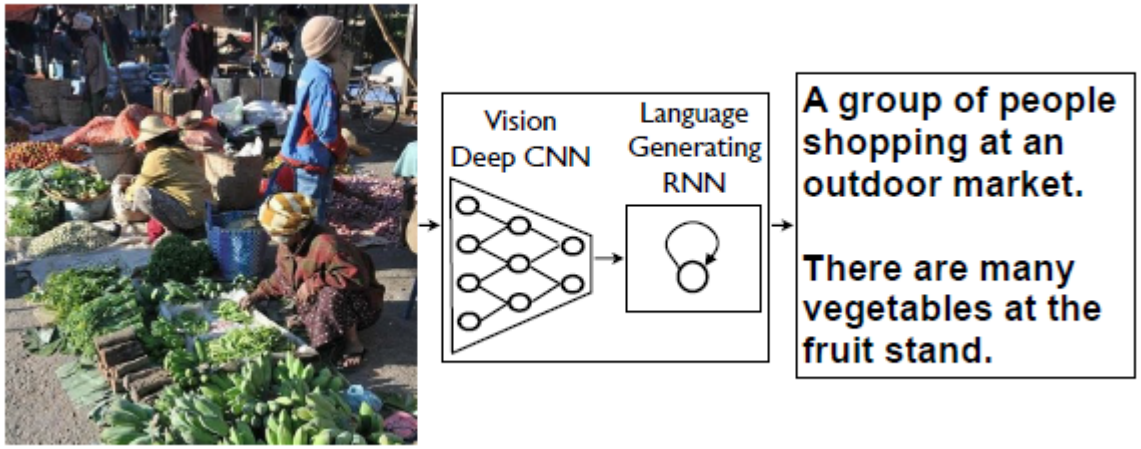
Handwriting synthesis:

primed and biased sampling



[Graves,2014]

Image Caption Generation



Demo (images)

Demo (top images for test texts)

Demo (more sophisticated model)

[Vinyals et al., 2015]

Image Caption Generation

a vase with flowers in it on a table
`logprob: -7.62`

a bed with a white comforter and a white blanket
`logprob: -12.01`

a group of people sitting around a table with food
`logprob: -7.32`

I am not really confident, but I think it's a pile of fruit sitting on a bed.

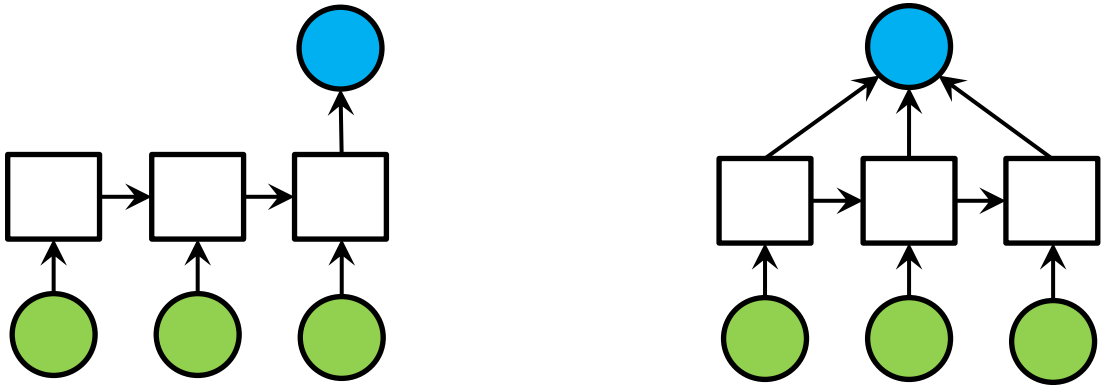
I am not really confident, but I think it's a man riding a snowboard down a snow covered slope.

I think it's a star filled sky.

Sequence input

Sequence classification:

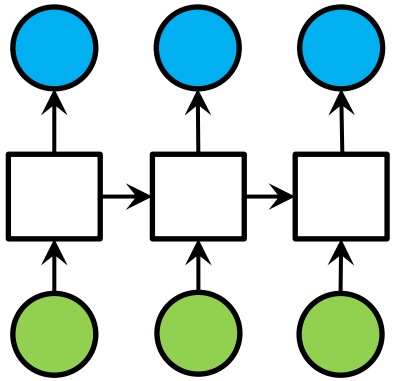
- Sentiment analysis



Sequence to sequence

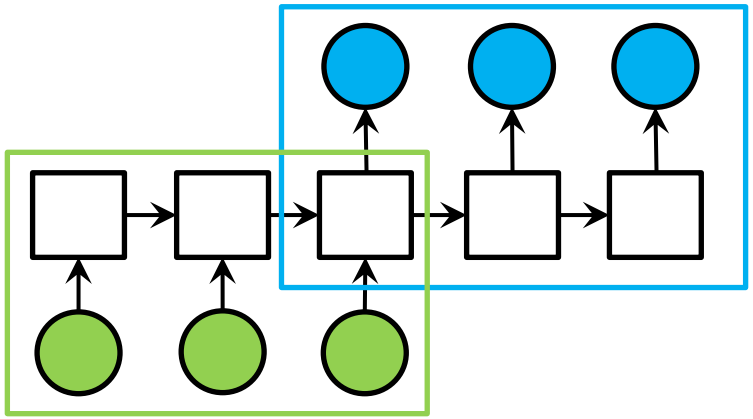
- Handwriting to text / text to handwriting
- Speech to text / text to speech

Input and output have different length!



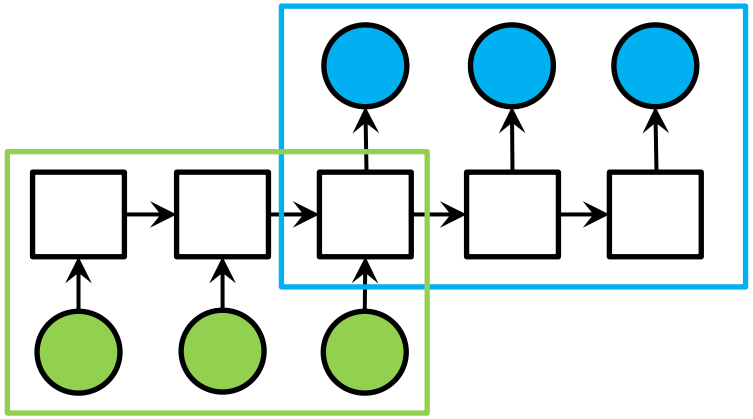
[Graves et al., 2006]

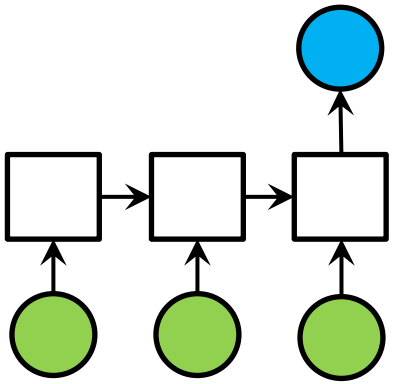
- Machine Translation

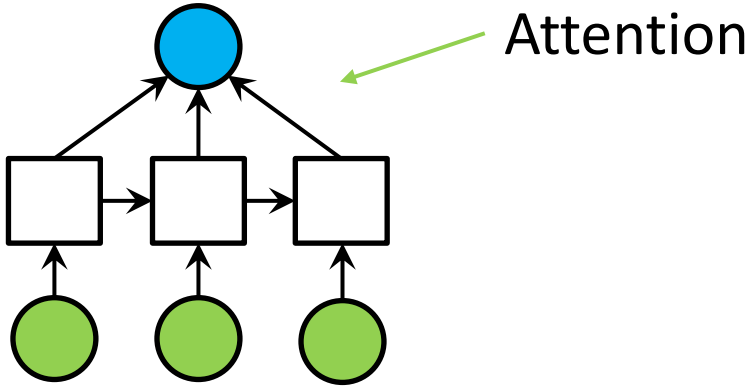


[Sutskever et al., 2014]

Sequence input



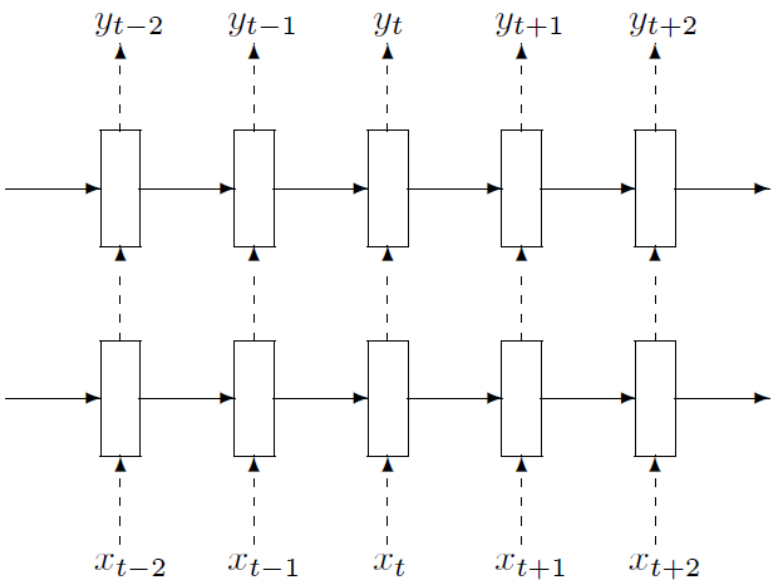


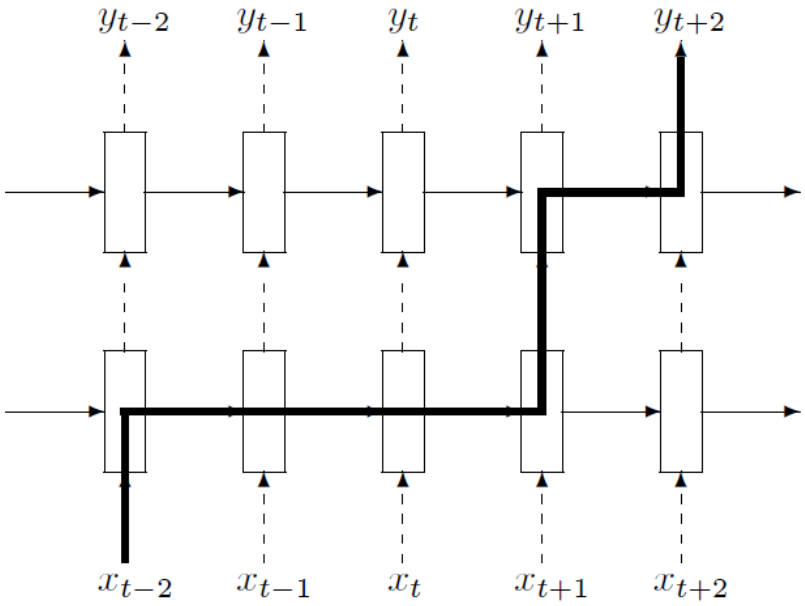


Tips and tricks

Naive Dropout for RNN

Only to non-recurrent connections!

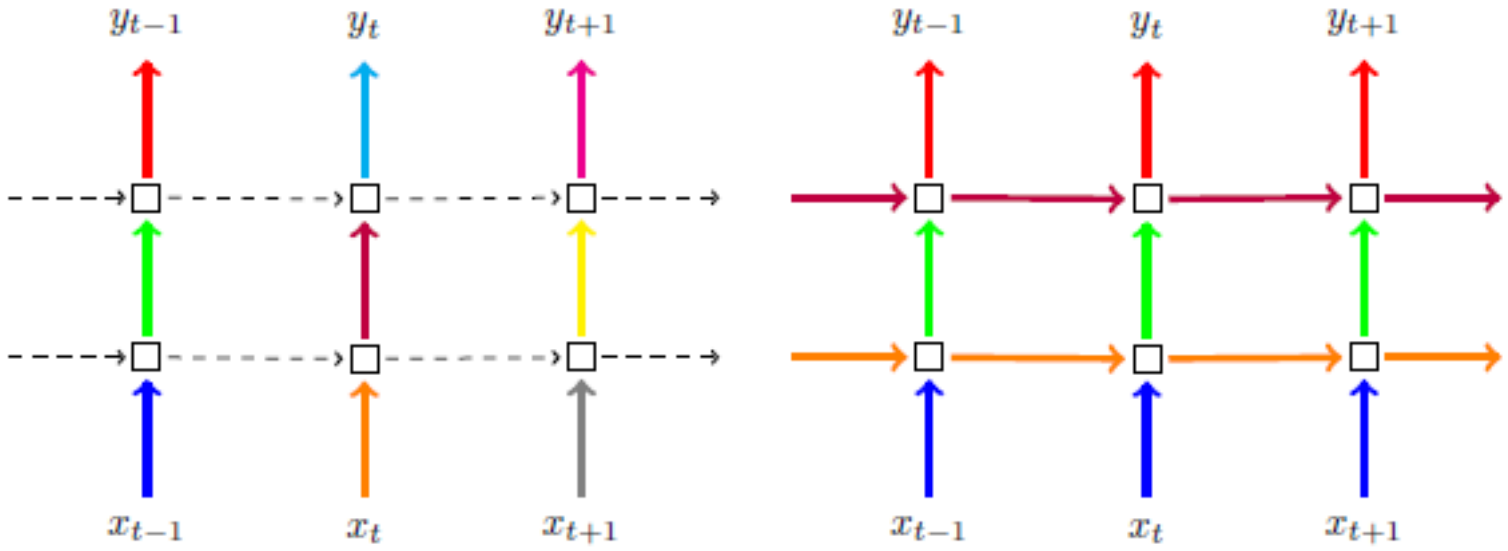




[Zaremba et al., 2015]

[Laurent et al., 2016]

Bayesian dropout for RNN

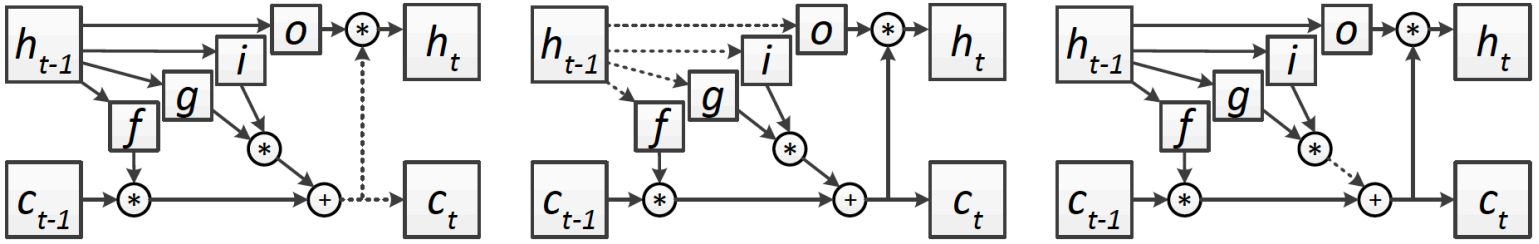


(a) Naive dropout RNN

(b) Variational RNN

[Gal, Ghahramani, 2016]

Dropout for LSTM/GRU



(a) Moon et al., 2015 (b) Gal, 2015 (c) Ours

Figure 1: Illustration of the three types of dropout in recurrent connections of LSTM networks. Dashed arrows refer to dropped connections. Input connections are omitted for clarity.

LSTM

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

GRU

$$h_t = (1 - u_t) \cdot d(g_t) + u_t \cdot h_{t-1}$$

[Semeniuta et al., 2016]

Zoneout

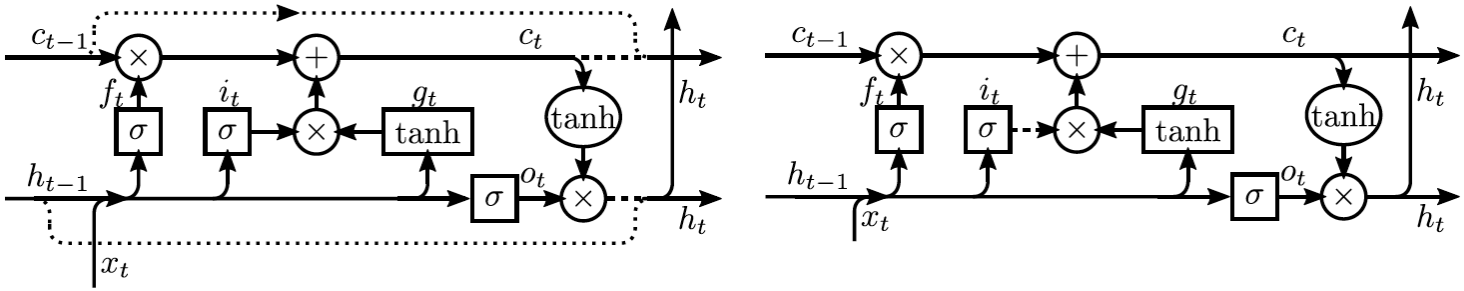


Figure 2: (a) Zoneout, vs (b) the recurrent dropout strategy of (Semeniuta et al., 2016) in an LSTM.

$$c_t = d_t^c \odot c_{t-1} + (1 - d_t^c) \odot \left(f_t \odot c_{t-1} + i_t \odot g_t \right)$$
$$h_t = d_t^h \odot h_{t-1} + (1 - d_t^h) \odot \left(o_t \odot \tanh \left(f_t \odot c_{t-1} + i_t \odot g_t \right) \right)$$

[Krueger et al., 2017]

Layer Normalization

Changes in the output of one layer will tend to cause highly correlated changes in the summed inputs to the next layer

Compute the layer normalization statistics over all the hidden units in the same layer!

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2} \qquad \bar{a}_i^l = \frac{g_i^l}{\sigma_i^l}\left(a_i^l - \mu_i^l\right)$$

All the hidden units in a layer share the same normalization terms, but different training cases have different normalization terms.

[Lei Ba et al., 2016]

Layer Normalization for RNN

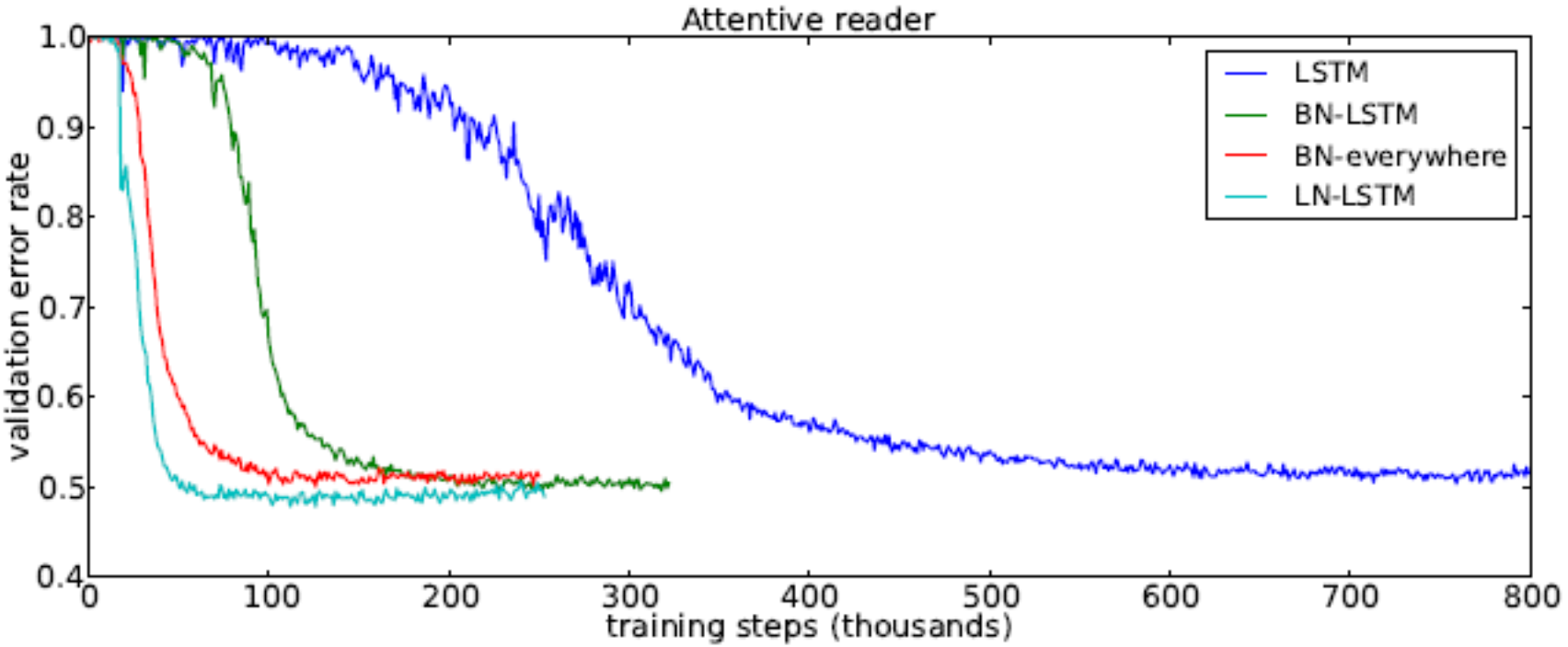
$$\mathbf{a}^t = W_{hh} h^{t-1} + W_{xh} \mathbf{x}^t$$

$$\mathbf{h}^t = f\left[\frac{\mathbf{g}}{\sigma^t} \odot \left(\mathbf{a}^t - \mu^t\right) + \mathbf{b}\right] \qquad \mu^t = \frac{1}{H}\sum_{i=1}^{H} a_i^t \qquad \sigma^t = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^t - \mu^t\right)^2}$$

Layer Normalization doesn't work well for CNN.

[Lei Ba et al., 2016]

BN vs LN for LSTM



[Lei Ba et al., 2016]

Reference: theory

Hochreiter, Sepp, and Jurgen Schmidhuber. Long short-term memory // Neural computation 9.8: 1735-1780. 1997.

F. A. Gers, J. Schmidhuber, F. Cummins. Learning to Forget: Continual Prediction with LSTM // Tech. Rep. No. IDSIA-01-99, 1999.

F. A. Gers. Long Short-Term Memory in Recurrent Neural Networks // PhD thesis, Department of Computer Science, Swiss Federal Institute of Technology, Lausanne, EPFL, Switzerland, 2001.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, Jürgen Schmidhuber. LSTM: A Search Space Odyssey.

Kyunghyun Cho et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation// EMNLP, 2014.

Razvan Pascanu, Tomas Mikolov, Yoshua Bengio. On the difficulty of training Recurrent Neural Networks // ICML, 2013.

Tomas Mikolov et al. Learning Longer Memory in Recurrent Neural Networks // ICLR, 2015.

Quoc V. Le, Navdeep Jaitly, Geoffrey E. Hinton. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units // arXiv, 2015.

Martin Arjovsky, Amar Shah, Yoshua Bengio. Unitary Evolution Recurrent Neural Networks // ICML, 2016.

Eugene Vorontsov et al. On orthogonality and learning recurrent networks with long term dependencies // ICML, 2017.

Zakaria Mhammedi et al. Efficient Orthogonal Parametrisation of Recurrent Neural Networks Using Householder Reflections // ICML, 2017.

Li Jing et al. Gated Orthogonal Recurrent Units: On Learning to Forget // arXiv, 2017.

Daniel Neil et al. Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences // NIPS, 2016.

Reference: examples

Sequence generation

- **Character-wise text generation with Multiplicative RNN**

  Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating Text with Recurrent Neural Networks // ICML 2011.

  demo, slides

- **Word-wise text generation with RNN (RNN vs n-grams)**

  Mikolov Tomá, Karafiát Martin, Burget Luká, Èernocký Jan, Khudanpur Sanjeev. Recurrent neural network based language model. // Proceedings of the 11th Annual Conference of the International Speech Communication Association (INTERSPEECH 2010).

  Mikolov Tomá. Statistical Language Models based on Neural Networks // PhD thesis, Brno University of Technology, 2012.

  lib+demo

- **Both character and word-wise text generation + handwritten generation + handwritten synthesis (all with LSTM)**

  A. Graves. Generating Sequences With Recurrent Neural Networks.

  slides, handwritten synthesis demo

Hermann et al. Teaching Machines to Read and Comprehend // NIPS 2015

A. Graves et al. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks // ICML 2006

Reference: examples

Sequence translation

Ilya Sutskever, Oriol Vinyals, Quoc Le. Sequence to Sequence Learning with Neural Networks // NIPS 2014

K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation // EMNLP 2014.

Dzmitry Bahdanau, Kyunghyun Cho, Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate // ICLR, 2015.

demo

Image Caption Generation

O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator // *CVPR*, 2015.

Andrej Karpathy, Li Fei-Fei. Deep Visual-Semantic Alignments for Generating Image Descriptions // CVPR, 2015.

demo (images), demo (top images for test texts)

Ryan Kiros, Ruslan Salakhutdinov, Richard Zemel. Unifying Visual-Semantic Embeddings with Multimodal Neural Language Models // TACL, 2015

Kenneth Tran et al. Rich Image Captioning in the Wild // CVPR Workshops, 2016.

demo

Reference: tips and tricks

Nitish Srivastava et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting // JMLR, 2014.

Wojciech Zaremba, Ilya Sutskever, Oriol Vinyals. Recurrent Neural Network Regularization // arXiv, 2014.

Yarin Gal, Zoubin Ghahramani. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks // NIPS, 2016.

Stanislau Semeniuta et al. Recurrent Dropout without Memory Loss // COLING, 2016.

David Krueger et al. Zoneout: Regularizing RNNs by Randomly Preserving Hidden Activations // ICLR, 2017.

Sergey Ioffe, Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift // ICML, 2015.

César Laurent et al. Batch Normalized Recurrent Neural Networks // ICASSP, 2016.

Tim Cooijmans et al. Recurrent Batch Normalization // arXiv, 2016.

Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton. Layer Normalization // arXiv, 2016.

A list of resources dedicated to RNNs: Awesome Recurrent Neural Networks

Andrej Karpathy . The Unreasonable Effectiveness of Recurrent Neural Networks // blogpost.

Andrej Karpathy, Justin Johnson, Li Fei-Fei. Visualizing and Understanding Recurrent Networks // ICLR, 2016.