



Neural Networks

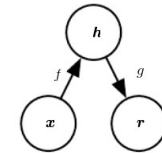
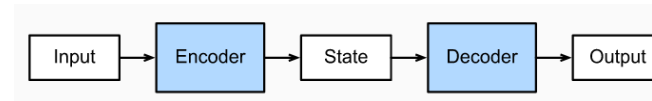
Lecture 10

Autoencoders and gated recurrent models

Igor Farkaš

2021

Autoencoders



- Encoder-decoder architecture = NN that is trained to attempt to copy its input to its output
- We focus on simpler case – a spatial mapping (no time involved)
- Encoder $h = f(x)$, decoder $r = g(h) = g(f(x))$ yields reconstruction
- $\dim(x) = \dim(r) > \dim(h) = >$ bottleneck
- imperfect reconstruction crucial (due to bottleneck)
- Can also be stochastic: $p_{\text{encoder}}(h | x)$ and $p_{\text{decoder}}(x | h)$, leading to generative models

2

Purpose

- Autoencoder (AE) – used for dim. reduction, since 1980s (LeCun, 1987; Bourlard & Kamp, 1988)
- **undercomplete** AE, i.e. If $\dim(h) < \dim(x) \rightarrow$ **bottleneck**
 - captures the most salient features of the training data
- Self-supervised training to minimize loss function $L(x, g(f(x)))$
- if linear and $L = \text{MSE}$, then \rightarrow PCA,
- nonlinear AE is a more powerful generalization
- **overcomplete** AE, i.e. $\dim(h) > \dim(x)$ interesting only...
- ... if regularized, in order to learn data distribution (in **latent space**)
- Interesting properties at hidden layer: sparsity, small derivatives of the representation, robustness

3

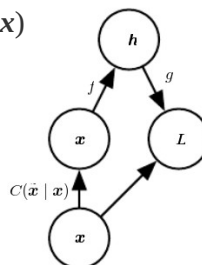
Sparse autoencoders

- Trained to minimize $L(x, g(f(x))) + P(h)$, ($P =$ **sparsity penalty**)
- typically used to learn features for another task (such as classification)
- e.g. $P(h) = \lambda \sum_i |h_i|$
- using ReLU activation function also enforces sparsity
- Probabilistic interpretation: learn generative model $p_{\text{model}}(x | h)$ that best explains observed data (by latent variables)
- Alternative: $L(x, g(f(x))) + P(h, x)$, where
- $P(h) = \lambda \sum_i \|\nabla_x h_i\|^2 \rightarrow$ **contractive autoencoder**

4

Denoising autoencoders

- Based on changing the reconstruction error term of the cost function (rather than adding penalty term)
- Minimizes $L(x, g(f(x')))$, where x' is noisy version of input x
- implicitly forced to learn the structure of data $p_{\text{data}}(x)$
- Introduces corruption process $C(x' | x)$
- DAE learns reconstruction distrib. $p_{\text{reconstruct}}(x | x')$
- ... from training pairs $\{x', x\}$
- can be trained by SGD as any feedforward NN

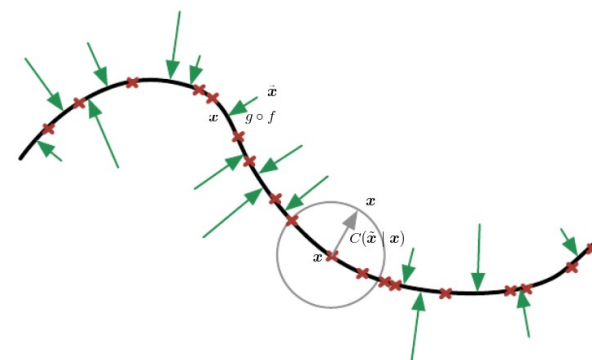


Original	noisy	reconstruction
7 2 1 0 4	7 2 1 0 4	7 2 1 0 4
1 4 9 5 9	1 4 9 5 9	1 4 9 5 9
0 6 9 0 1	0 6 9 0 1	0 6 9 0 1
5 9 7 8 4	5 9 7 8 4	5 9 7 8 4
7 6 6 5 4	7 6 6 5 4	7 6 6 5 4

opendeep.org

5

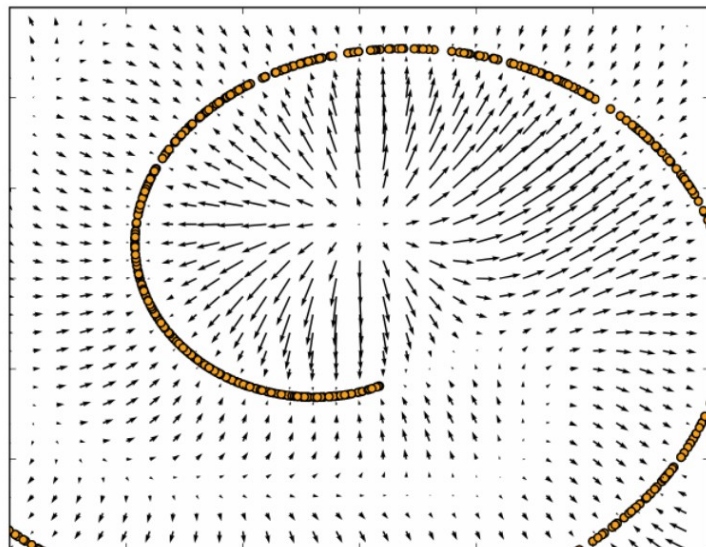
Graphical interpretation of DAE learning



- data x assumed to lie on a low-dim. manifold \mathbf{M} (black curve)
- x' represent departures from \mathbf{M}
- DAE learns a vector field (green arrows): $g(f(x)) - x$
- projections onto the manifold

6

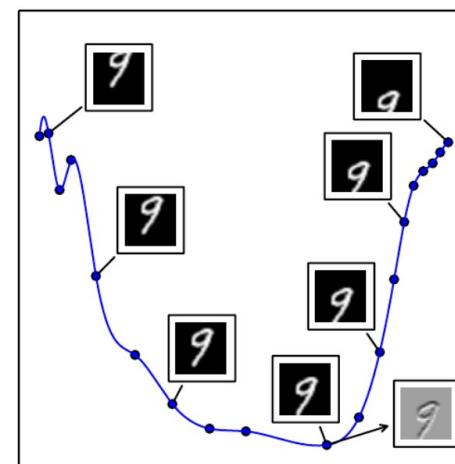
Example: 2D \rightarrow 1D



7

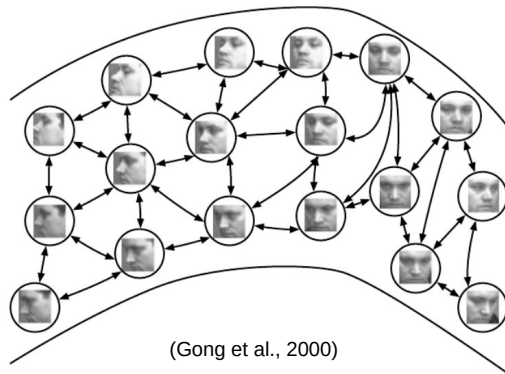
Manifold learning with autoencoder

- 1D example in 784-dim. space
- vertically translated images \rightarrow a coordinate along \mathbf{M}
- \mathbf{M} projected in 2D (via PCA)
- Each node is associated with a tangent plane that spans the directions of variations associated with difference vectors between the example and its neighbors
- shown (bottom right) in example



8

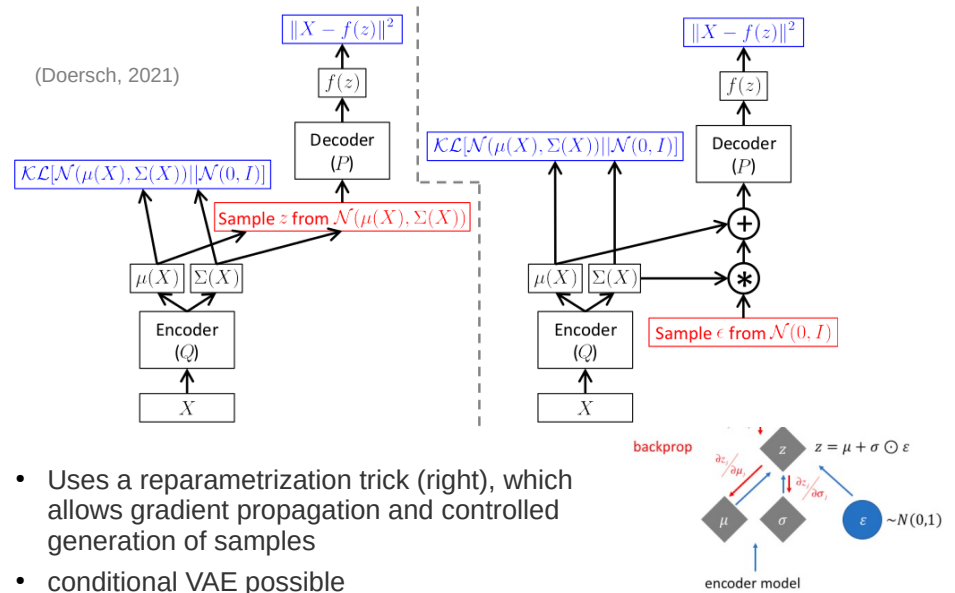
2D example with manifold of faces



- Unsupervised learning of manifold (**embedding**) based on a (nonparametric) nearest neighbors graph
- Generalization to new examples possible via interpolation for dense graphs

9

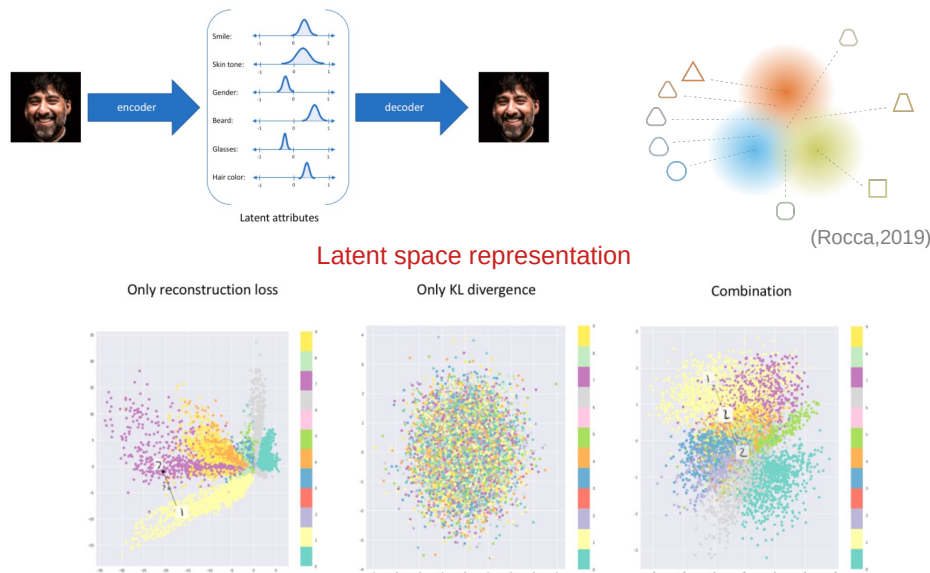
Variational autoencoder (VAE)



- Uses a reparametrization trick (right), which allows gradient propagation and controlled generation of samples
- conditional VAE possible

10

VAE properties

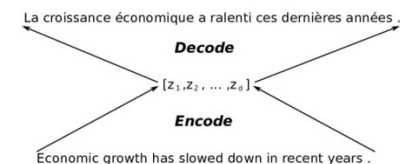


<https://www.jeremyjordan.me/variational-autoencoders/>

11

Applications of autoencoders

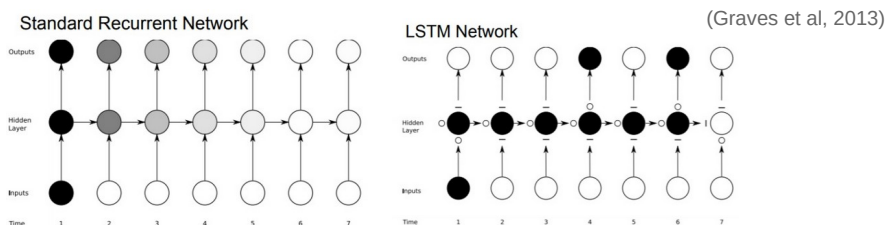
- Explicit dim. reduction for subsequent classification – reduces error (also less memory and runtime)
- can be applied recursively (hierarchically)
- **Information retrieval** – task of finding entries in a database that resemble (are relevant for) a query entry
 - entries mapped to binary low-dim. hash codes (fast search)
 - entries with the same or slightly different codes (a few bits flipped) retrieved → **semantic hashing**
 - sigmoid units used in encoded (forced to saturate)
 - technique used for text and images
- machine translation



12

Recurrent NN models with gated units

- Help preserve long-term dependencies (via gradient learning)
- Two models will be mentioned: GRU (Cho et al, 2014), LSTM (Hochreiter & Schmidhuber, 2007) – more complex
- New components:
 - memory cell (to capture long-term dep.)
 - skipping irrelevant inputs (in latent space)
 - resetting (internal state representation)



13

Gating the hidden state

minibatches (of size n)

$\mathbf{X}_t [n \times d]$ (examples \times dimension)

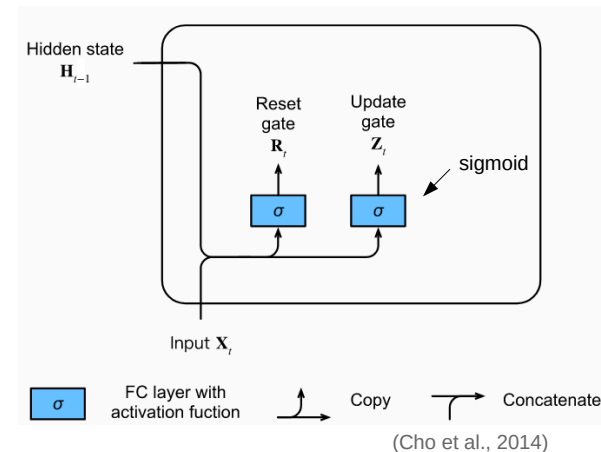
$\mathbf{H}_{t-1} [n \times h]$

$\mathbf{R}_t \mathbf{Z}_t [n \times h]$

$\mathbf{W}_{xr}, \mathbf{W}_{xz} [d \times h]$

$\mathbf{W}_{hr}, \mathbf{W}_{hz} [h \times h]$

$\mathbf{b}_t, \mathbf{b}_z [1 \times h]$



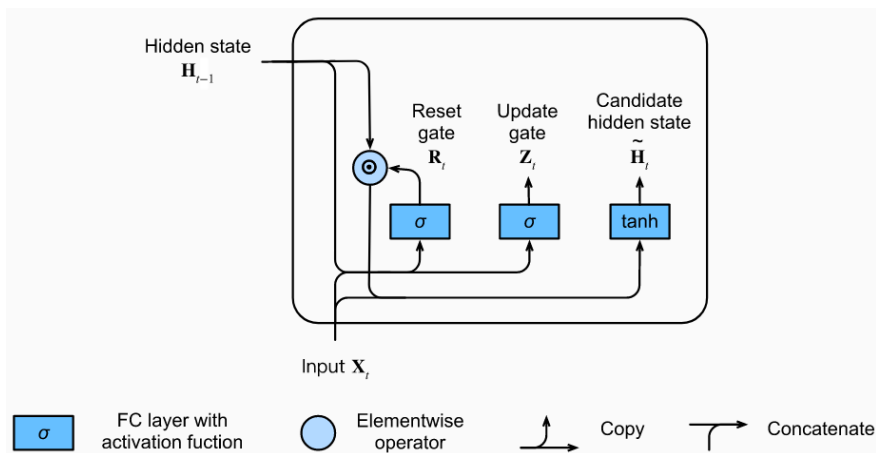
$$\mathbf{R}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xr} + \mathbf{H}_{t-1} \mathbf{W}_{hr} + \mathbf{b}_r)$$

$$\mathbf{Z}_t = \sigma(\mathbf{X}_t \mathbf{W}_{xz} + \mathbf{H}_{t-1} \mathbf{W}_{hz} + \mathbf{b}_z)$$

(Zhang et al, 2020)

14

Reset gates in action

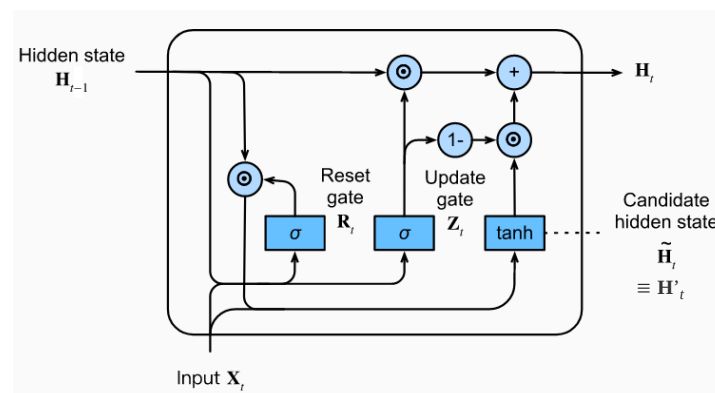


$$\mathbf{H}_t = \tanh(\mathbf{X}_t \mathbf{W}_{xh} + \mathbf{R}_t \odot \mathbf{H}_{t-1} \mathbf{W}_{hh} + \mathbf{b}_h)$$

- help capture short-term dependencies in time series

15

Update gates in action



$$\mathbf{H}_t = \mathbf{Z}_t \odot \mathbf{H}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{H}}_t$$

- help capture long-term dependencies in time series

16

LSTM's gated memory cells

- inspired by logic gates of a computer
- 3 gates controls the behavior of the memory cell (**latent state**)
- output gate** – controls when to read from the cell
- input gate** – controls when to read data into the cell
- forget gate** – controls when to reset the contents of the cell
- In addition, LSTM introduces a **memory cell (C)**
 - having the same shape as latent state (**H**)
 - providing additional information
- GRU is simpler: has a single mechanism for input and forgetting

17

LSTM's three gates

minibatches (of size n)

$\mathbf{X}_t [n \times d]$ (examples \times dimension)

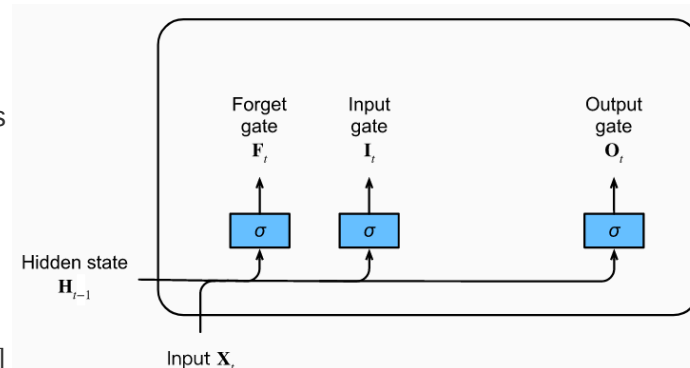
$\mathbf{H}_{t-1} [n \times h]$

$\mathbf{R}_t, \mathbf{Z}_t [n \times h]$

$\mathbf{W}_{xi}, \mathbf{W}_{xf}, \mathbf{W}_{xo} [d \times h]$

$\mathbf{W}_{hi}, \mathbf{W}_{hf}, \mathbf{W}_{ho} [h \times h]$

$\mathbf{b}_i, \mathbf{b}_f, \mathbf{b}_o [1 \times h]$



$$\begin{aligned} \mathbf{I}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xi} + \mathbf{H}_{t-1} \mathbf{W}_{hi} + \mathbf{b}_i) \\ \mathbf{F}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xf} + \mathbf{H}_{t-1} \mathbf{W}_{hf} + \mathbf{b}_f) \\ \mathbf{O}_t &= \sigma(\mathbf{X}_t \mathbf{W}_{xo} + \mathbf{H}_{t-1} \mathbf{W}_{ho} + \mathbf{b}_o) \end{aligned}$$

(Hochreiter & Schmidhuber, 1997)

18

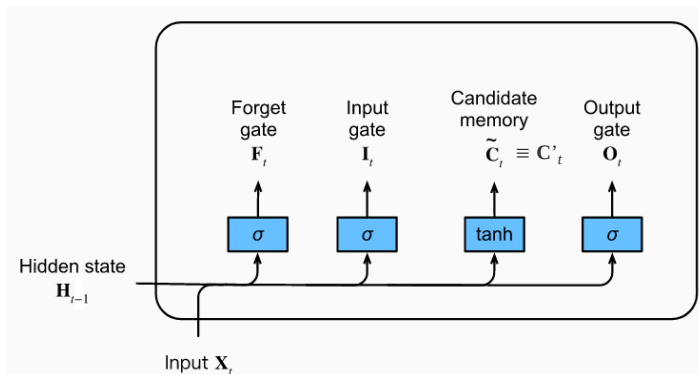
Candidate memory cell

$\mathbf{W}_{xc} [d \times h]$

$\mathbf{W}_{hc} [h \times h]$

$\mathbf{B}_c [1 \times h]$

$\mathbf{C}'_t [n \times h]$

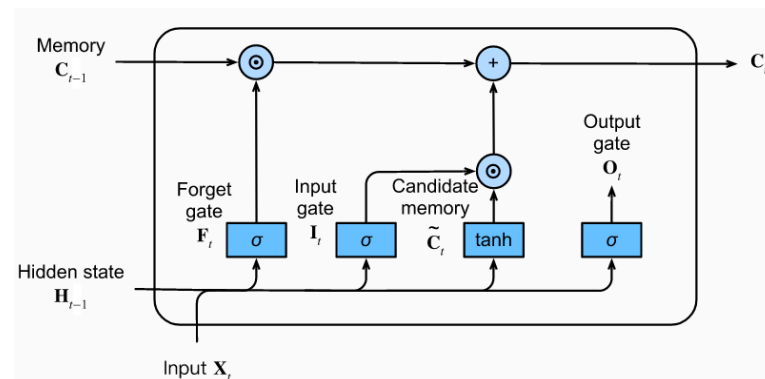


$$\mathbf{C}'_t = \tanh(\mathbf{X}_t \mathbf{W}_{xc} + \mathbf{H}_{t-1} \mathbf{W}_{hc} + \mathbf{b}_c)$$

computation similar to the 3 gates described above, but using a tanh function

19

Memory cell

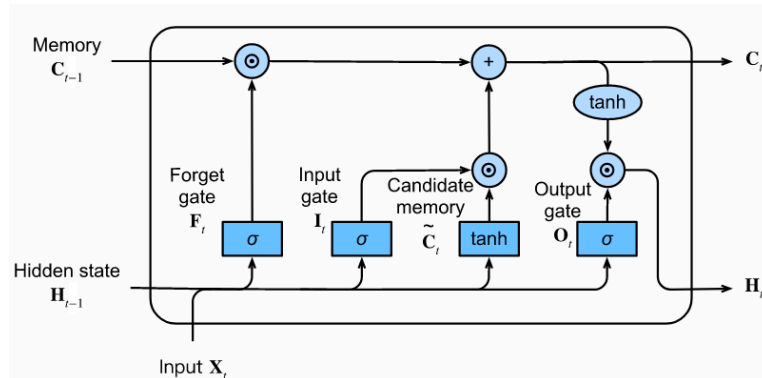


LSTM has 2 parameters: \mathbf{I}_t governs how much of new data we take via \mathbf{C}'_t and \mathbf{F}_t determines how much of the old memory content \mathbf{C}_{t-1} we retain.

$$\mathbf{C}_t = \mathbf{F}_t \odot \mathbf{C}_{t-1} + \mathbf{I}_t \odot \mathbf{C}'_t$$

20

Hidden states



Hidden state is O_t -gated version of the tanh of the memory cell:

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t)$$

21

Complete LSTM dynamics

Input gates

$$\mathbf{g}(t) = \sigma(\mathbf{U}^{\text{inp}} \mathbf{x}(t) + \mathbf{W}^{\text{inp}} \mathbf{h}(t-1) + \mathbf{b}^{\text{inp}})$$

Forget gates

$$\mathbf{f}(t) = \sigma(\mathbf{U}^{\text{fgt}} \mathbf{x}(t) + \mathbf{W}^{\text{fgt}} \mathbf{h}(t-1) + \mathbf{b}^{\text{fgt}})$$

Cell state

$$\mathbf{s}(t) = \mathbf{f}(t) \odot \mathbf{s}(t-1) + \mathbf{g}(t) \odot \sigma(\mathbf{U}^{\text{fgt}} \mathbf{x}(t) + \mathbf{W}^{\text{fgt}} \mathbf{h}(t-1) + \mathbf{b})$$

Output gates

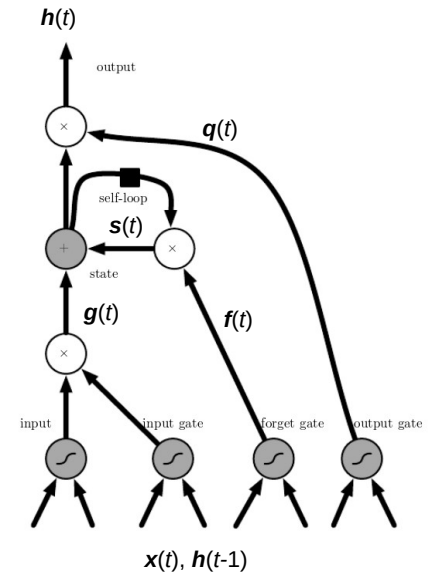
$$\mathbf{q}(t) = \sigma(\mathbf{U}^{\text{out}} \mathbf{x}(t) + \mathbf{W}^{\text{out}} \mathbf{h}(t-1) + \mathbf{b}^{\text{out}})$$

LSTM state output

$$\mathbf{h}(t) = \tanh(\mathbf{s}(t)) \odot \mathbf{q}(t)$$

\odot = element-wise multiplication

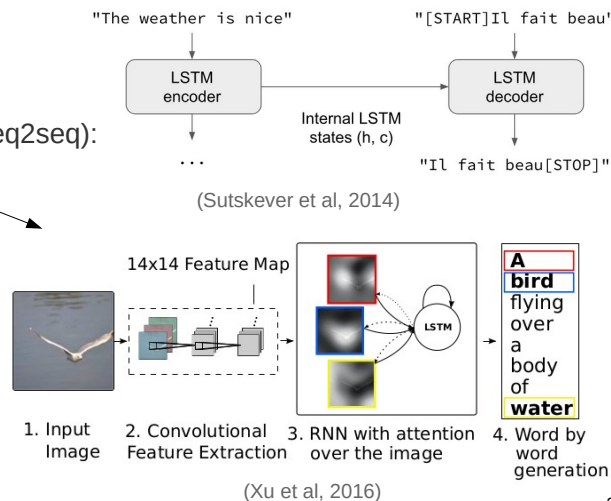
(Goodfellow et al, 2015) - use different symbols



22

Applications of LSTM

- unconstrained handwriting recognition
- speech recognition
- music generation
- parsing (PoS tagging)
- machine translation (seq2seq):
- image captioning
- ...
- new:
 - attention mech.
 - bidirect. models



(Xu et al, 2016)

23

LSTM summary

- Using trained gates, it introduces self-loops to produce paths where the gradient can flow for long (neither exploding nor vanishing)
- the time scale of integration can be changed dynamically
- the cell state is the core of the LSTM, controlled by the gates
- Trainable with various methods, e.g. SGD, 2nd order methods, Nesterov gradient, ...
- Various variants found useful, clipping the gradient, e.g. element-wise (Mikolov, 2012); or by L2 norm (Pascanu et al, 2013).
- Can be combined with autoencoders

24