ITMO UNIVERSITY

Deep nets architectures

Natalia Khanzhina

Senior Researcher

Machine Learning Lab

nehanzhina@corp.ifmo.ru

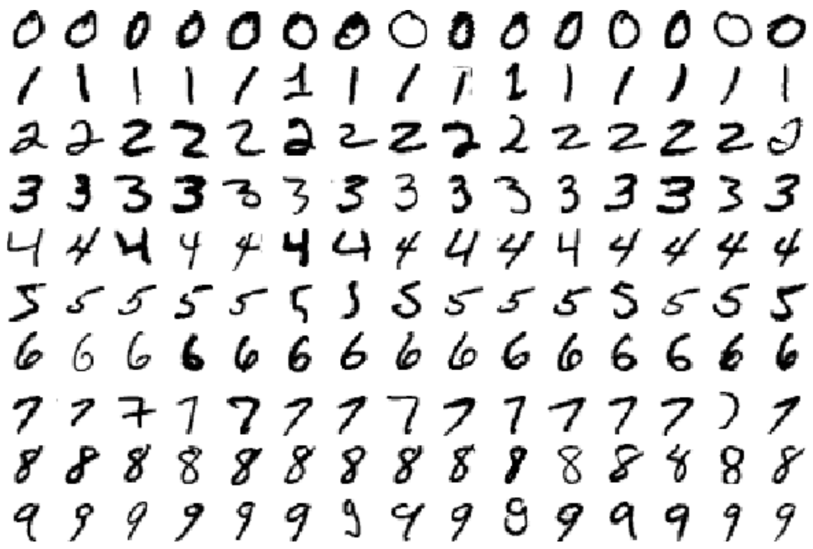
St. Petersburg, Russia

Outline

- MNIST dataset
- CNN
- Autoencoders
- Recurrent Network
- Long-Short Term Memory
- Hopfield network
- Hamming network

MNIST dataset

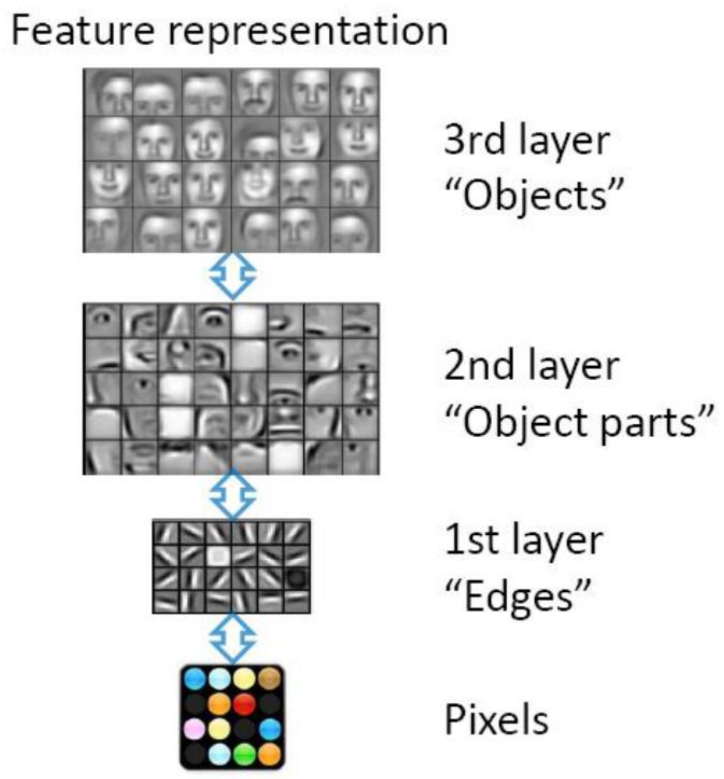
- The training set contains 60000 examples the test set 10000
- Grayscale images of size 28x28
- In past specific features extracted
- examples.http://yann.lecun.com/exdb/mnist/



Levels of abstraction

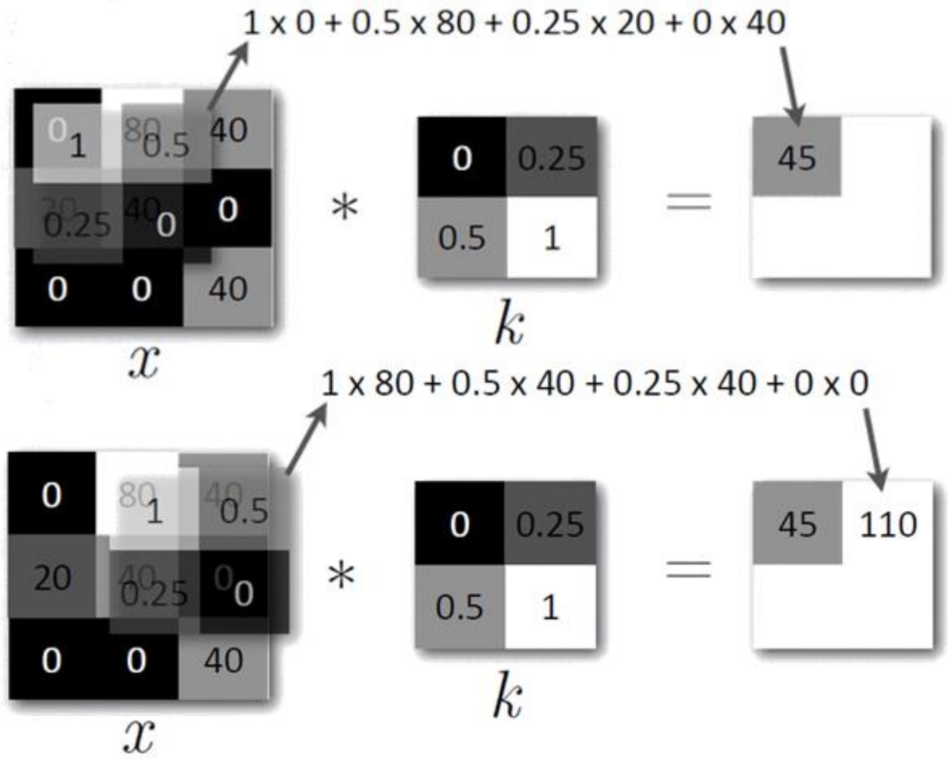
Hierarchical Learning:

- Natural progression from low level to high level structure as seen in natural complexity
- Easier to monitor what is being learnt and to guide the machine to better subspaces
- A good lower level representation can be used for many distinct tasks



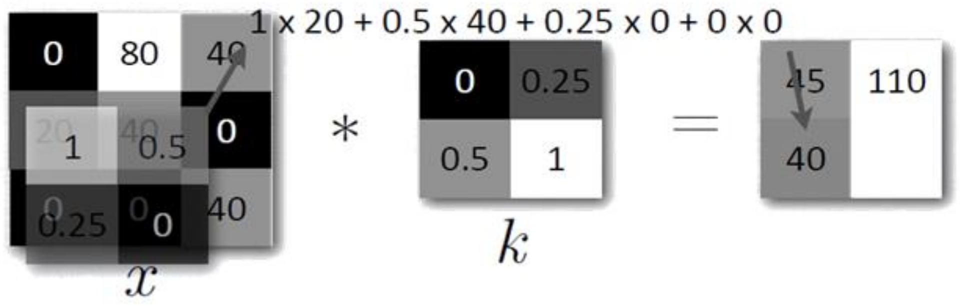
Discrete convolution

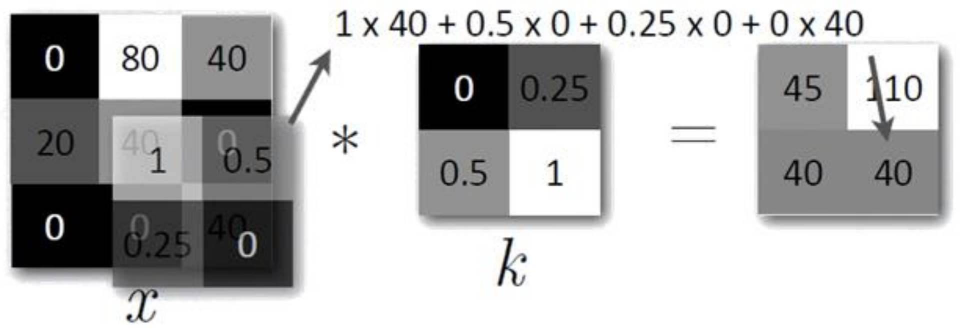
$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q}\, k_{r-p,r-q}$$



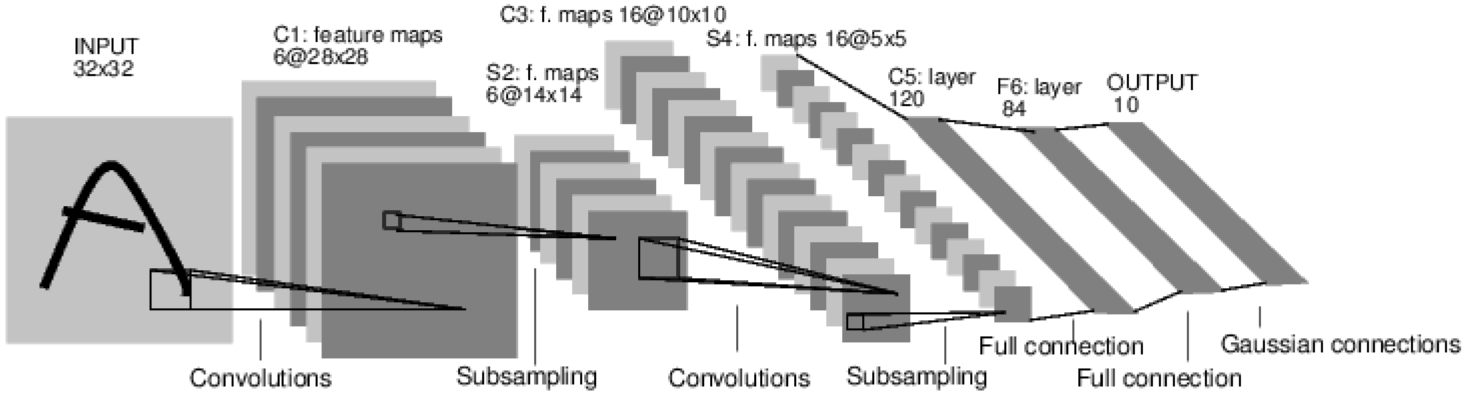
Discrete convolution

$$(x * k)_{ij} = \sum_{pq} x_{i+p,j+q} \, k_{r\text{-}p,r\text{-}q}$$





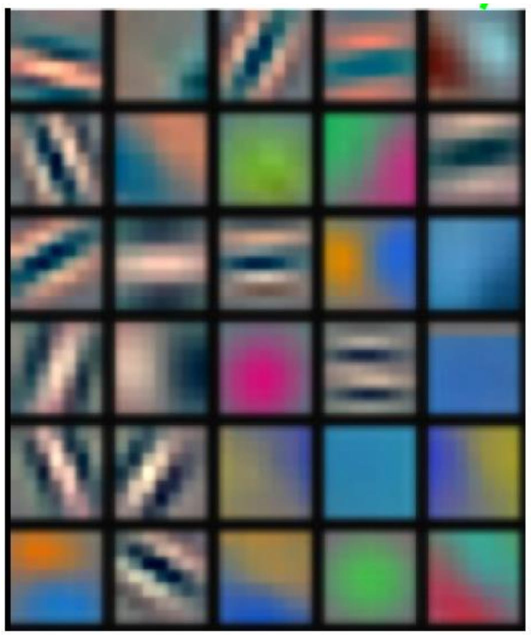
LeNet



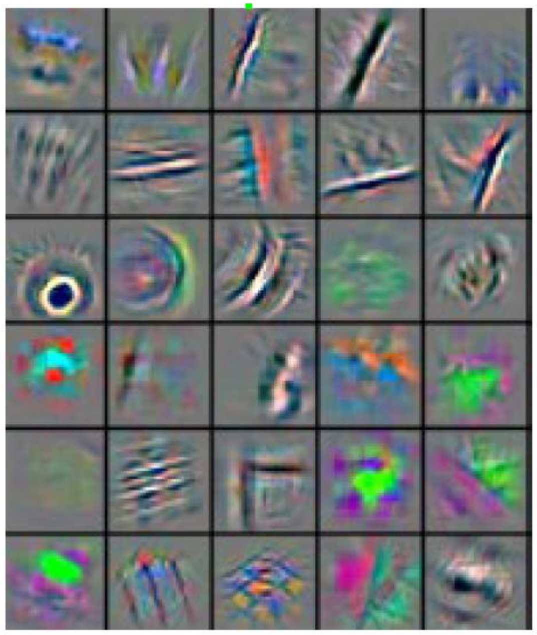
LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

What do trained kernels look like?

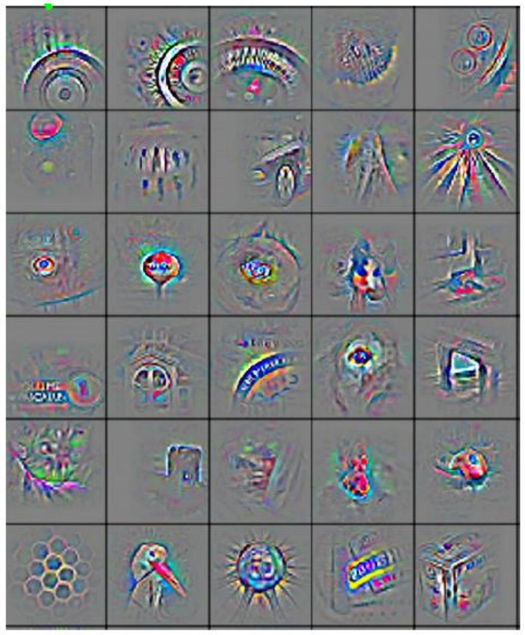
low feature



medium feature



high feature

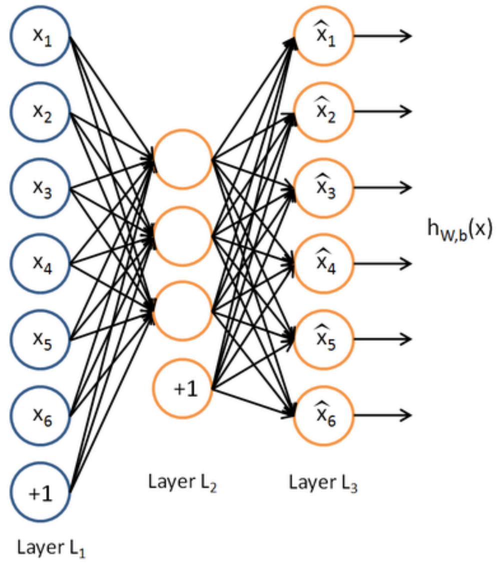


- Each kernel composes a local patch of lower-level features into high level representation

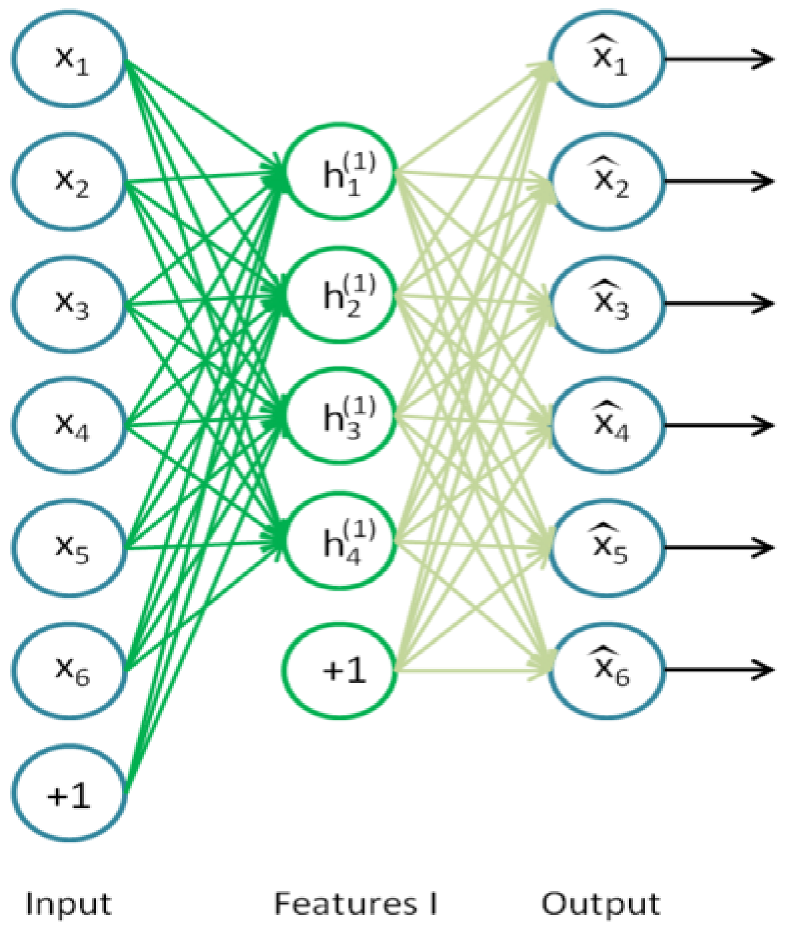
Autoencoders

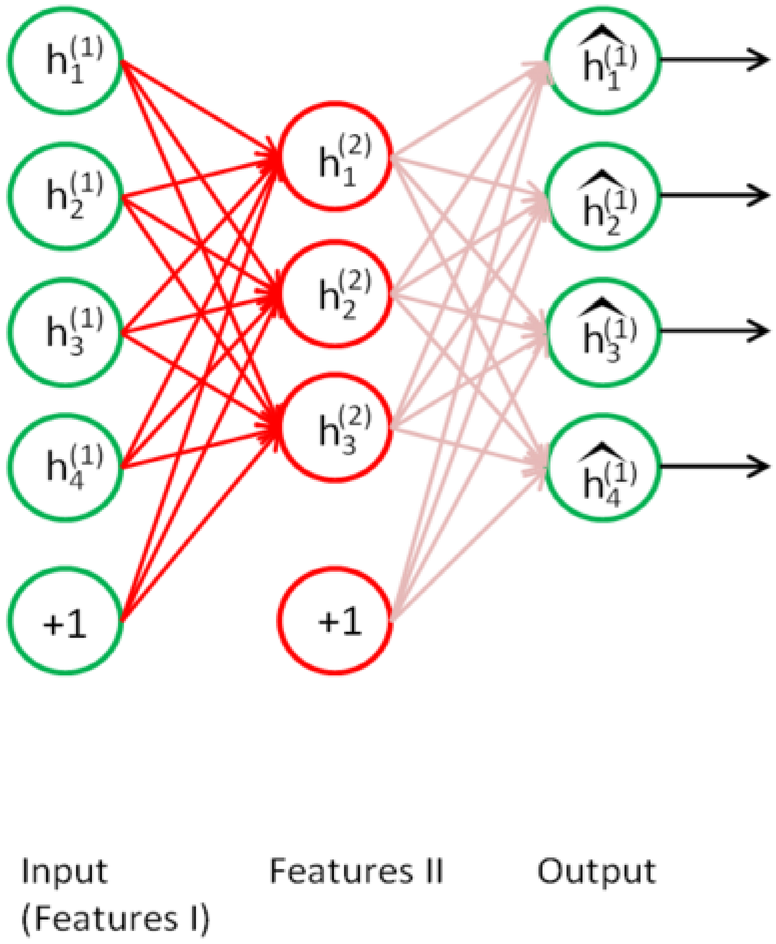
Autoencoder: a feed-forward neural network trained to reproduce its input at the output layer

- Make non-linear dimensionality reduction
- Train via backpropagation
- 1-layer autoencoder is similar with PCA

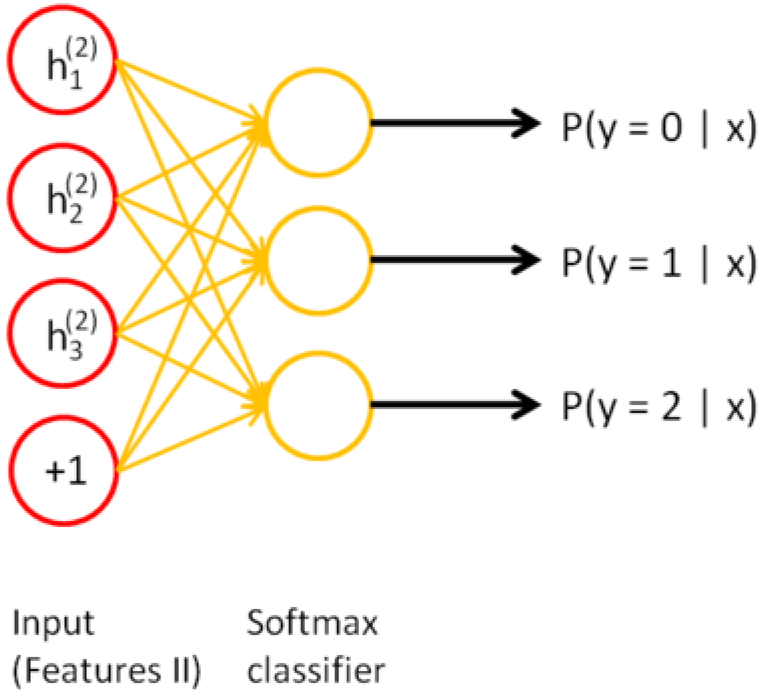


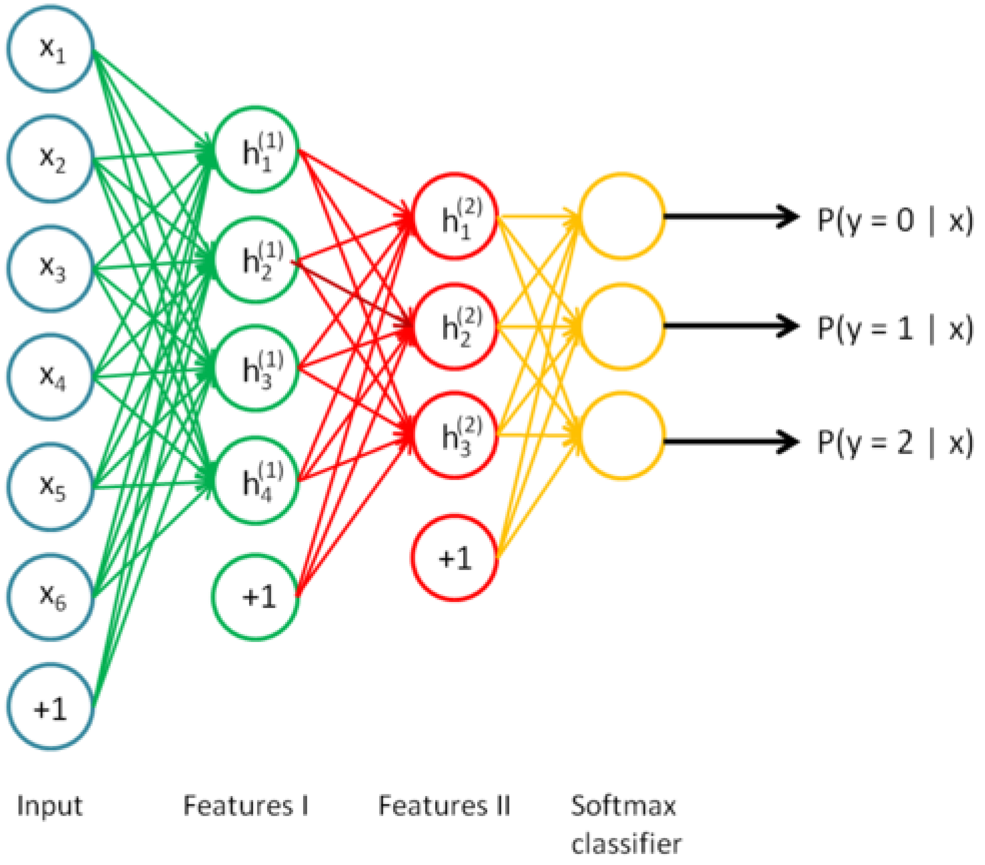
Autoencoders





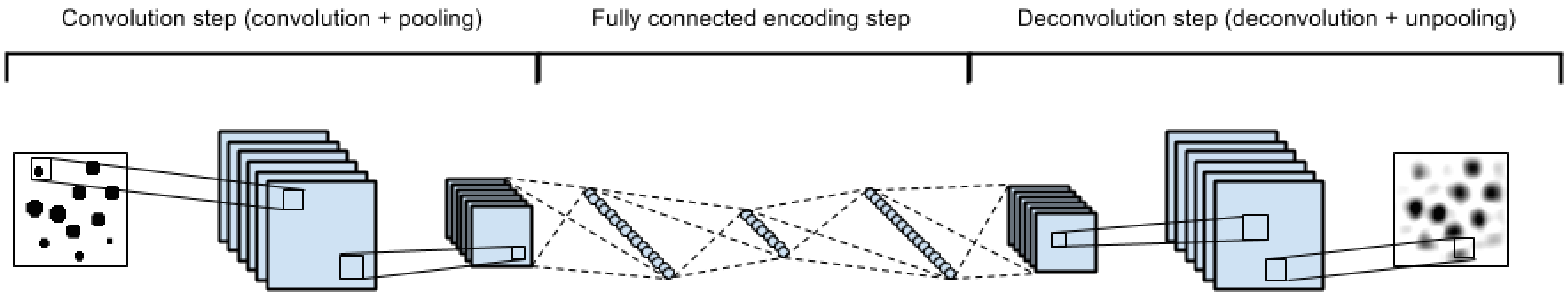
Autoencoders



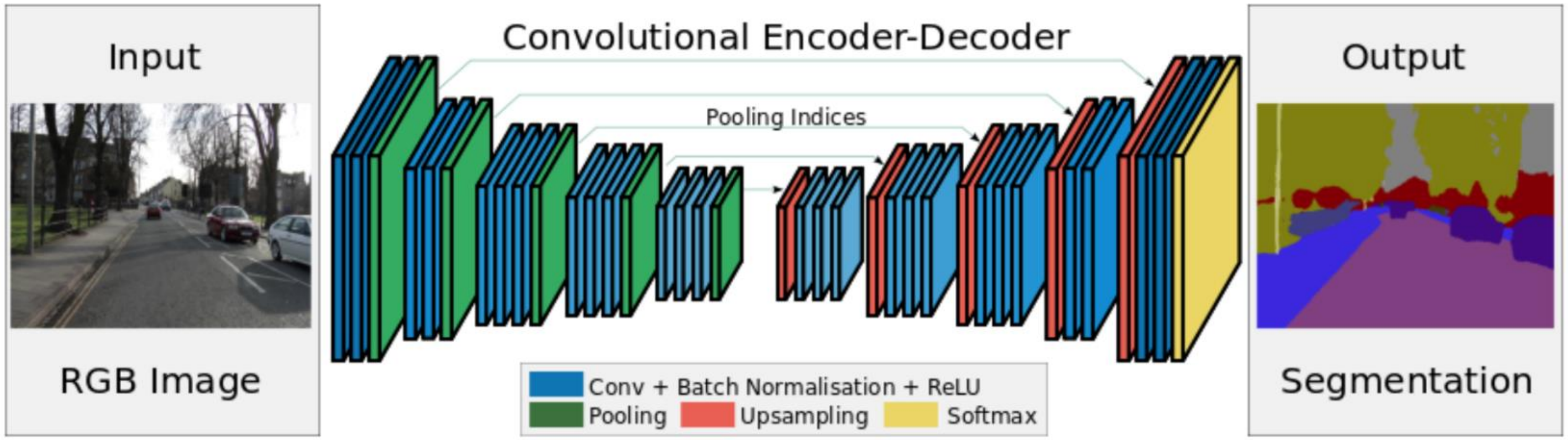


ITMO UNIVERSITY

C-Autoencoders


Convolution step (convolution + pooling)
Fully connected encoding step
Deconvolution step (deconvolution + unpooling)

Autoencoders for semantic segmentation

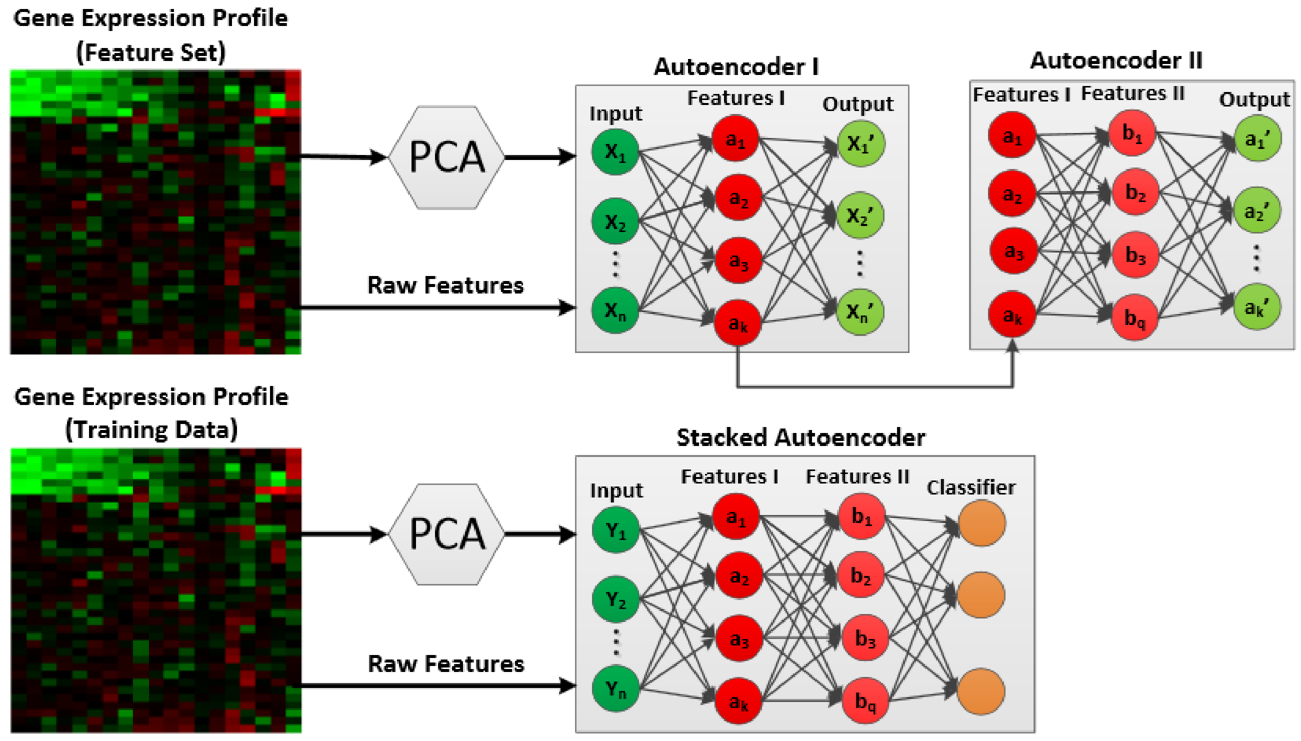


Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation." IEEE transactions on pattern analysis and machine intelligence 39.12 (2017): 1481-1495.

Autoencoders for semantic segmentation

- The task is to classify every pixel
- State-of-the-art results in this field
- The most cool nets are:
- U-net
- SegNet
- MaskNet

Autoencoders in bioinformatics

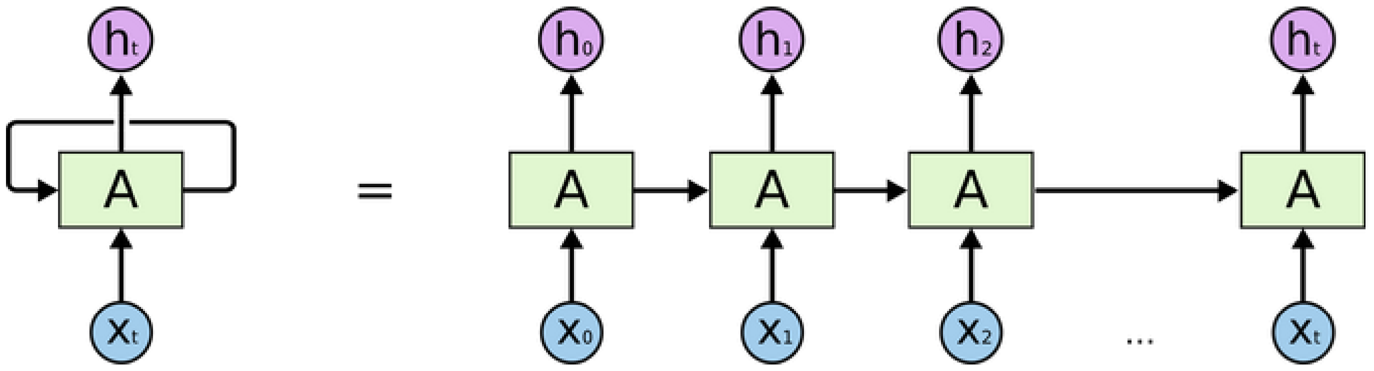


Fakoor, Rasool, et al. "Using deep learning to enhance cancer diagnosis and classification." *Proceedings of the International Conference on Machine Learning*. 2013.

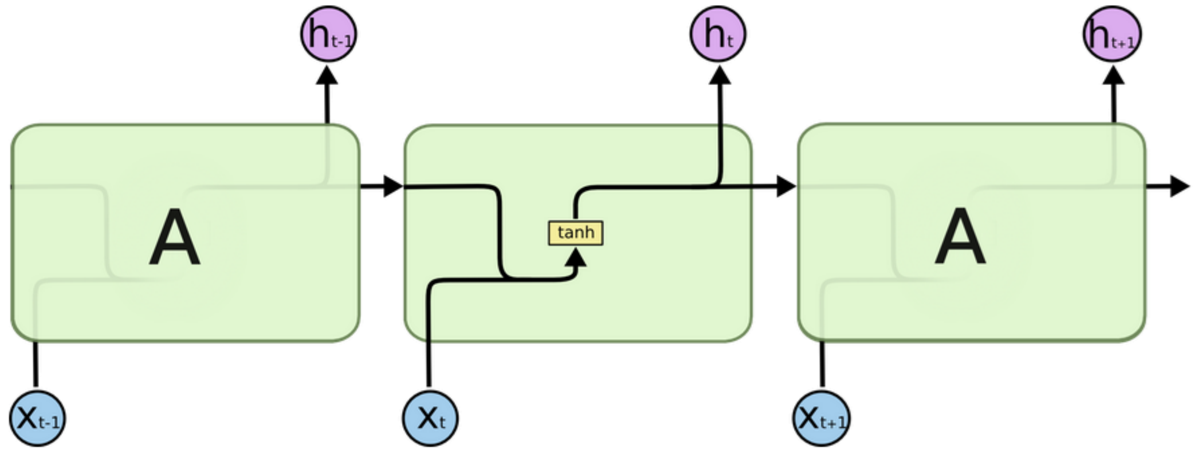
Recurrent Neural Network (RNN)

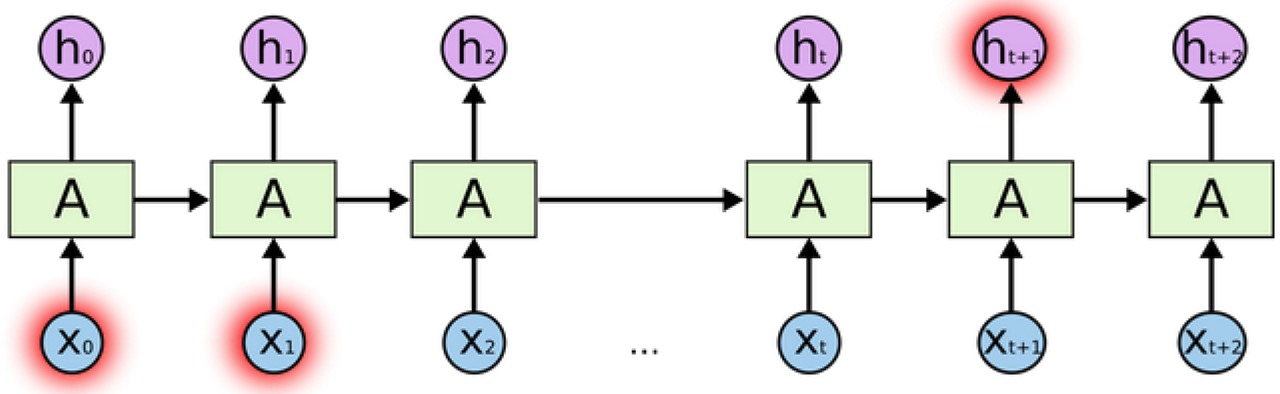
RNN: a neural network with recurrent connections

- ✓ Suitable for sequence data: time series, text, audio



Recurrent Neural Network (RNN)

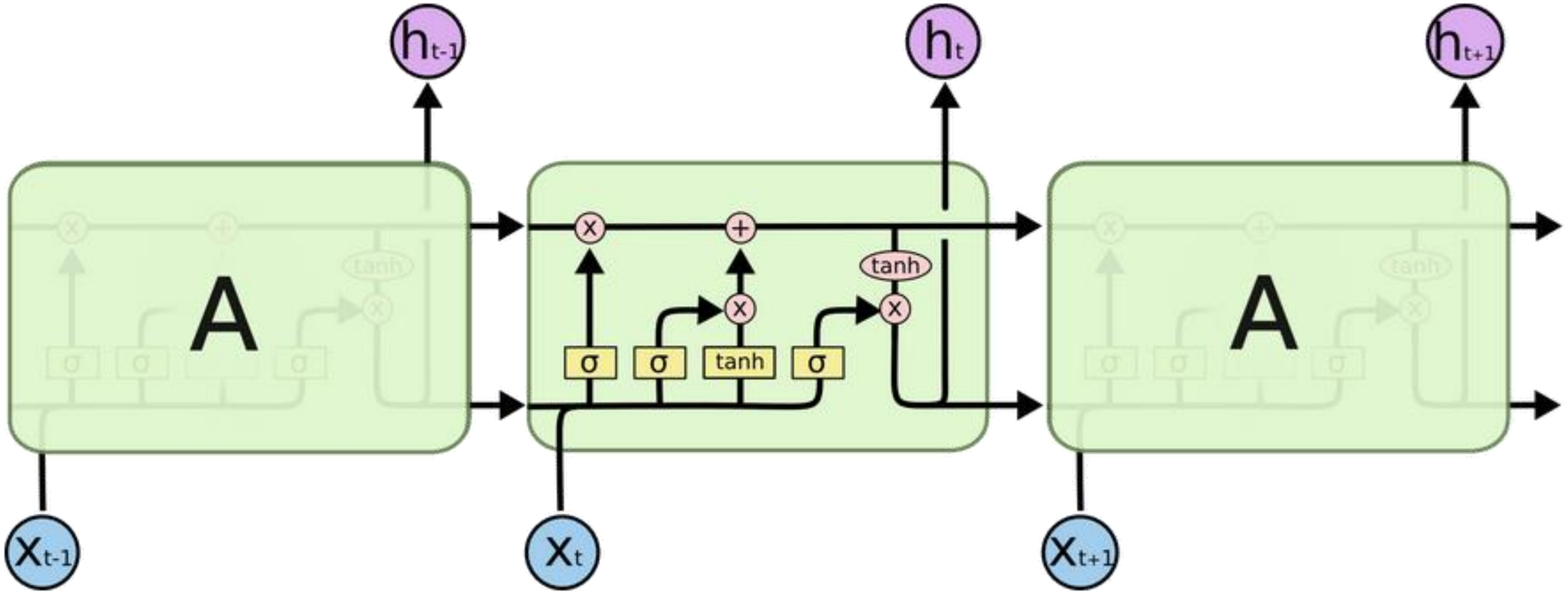




Long Short-Term Memory (LSTM)

LSTM: a special case of RNN able to learn long-term dependencies

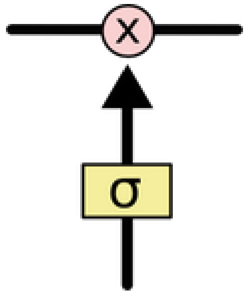
- There are four neural network layers in repeating module



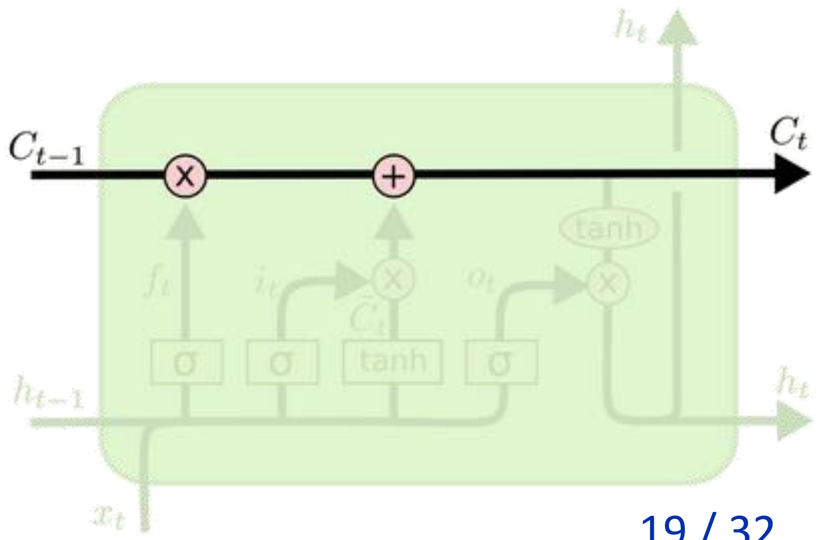
Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1732-1780.

LSTM: Cell state

Cell state: runs straight down the entire chain, with only some minor linear interactions

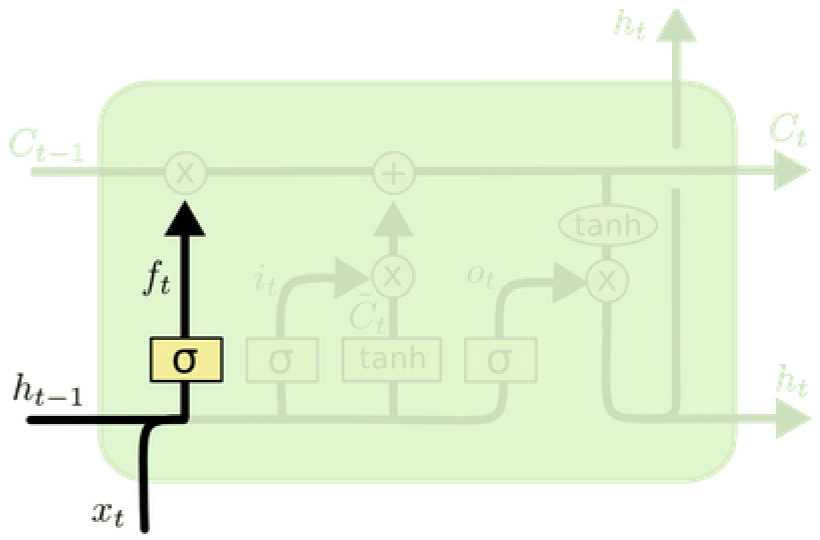


- LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to sample the input information.
- The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. LSTM has 3 gates



LSTM: Forget gate layer

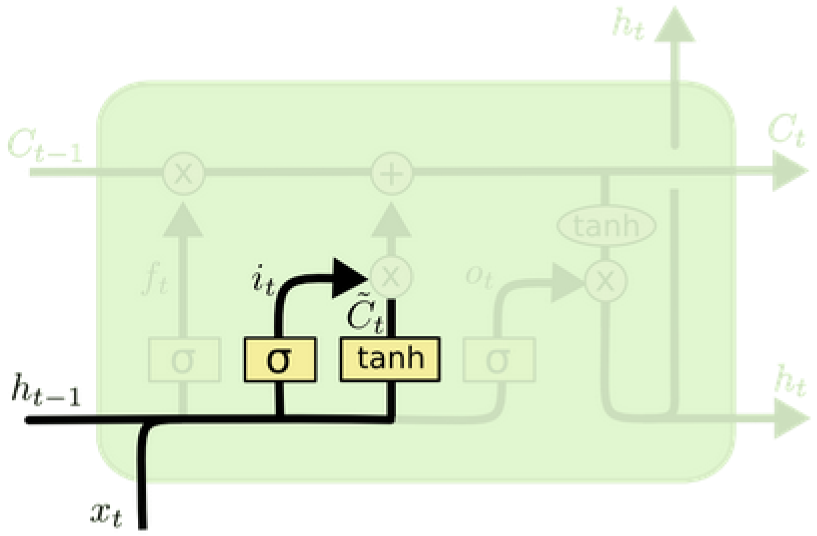
- It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents “completely keep this” while a 0 represents “completely get rid of this”



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

LSTM: Input gate layer

- The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, C_t, that could be added to the state. In the next step, we'll combine these two to create an update to the state



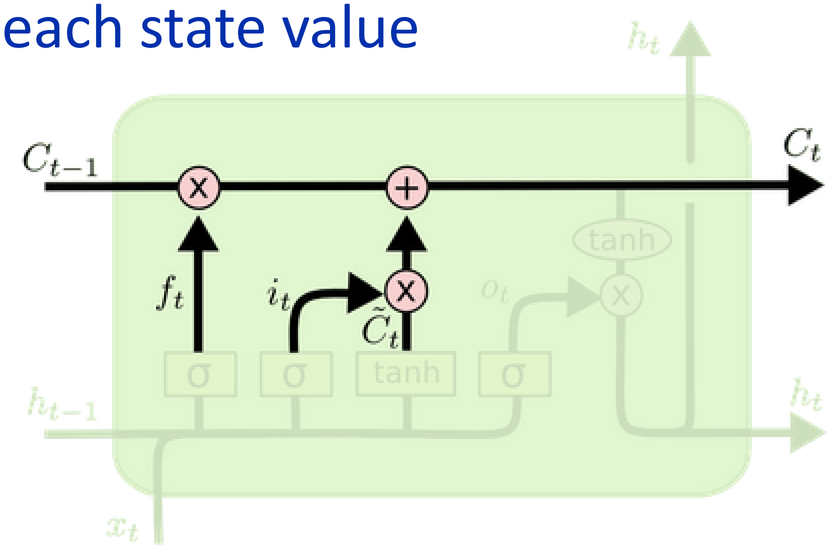
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

ITMO UNIVERSITY

LSTM: Input gate layer

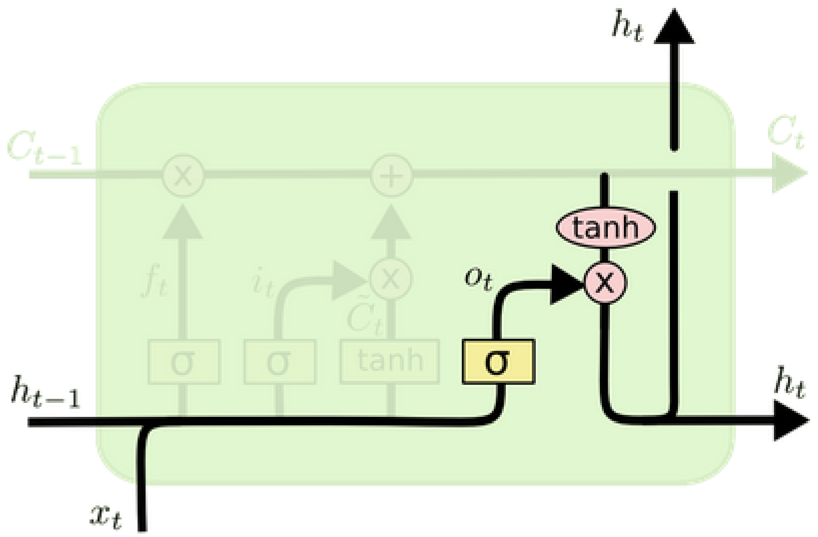
- It's now time to update the old cell state, C_{t-1}, into the new cell state C_t. The previous steps already decided what to do, we just need to actually do it
- We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add $i_t * C_t$. This is the new candidate values, scaled by how much we decided to update each state value



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

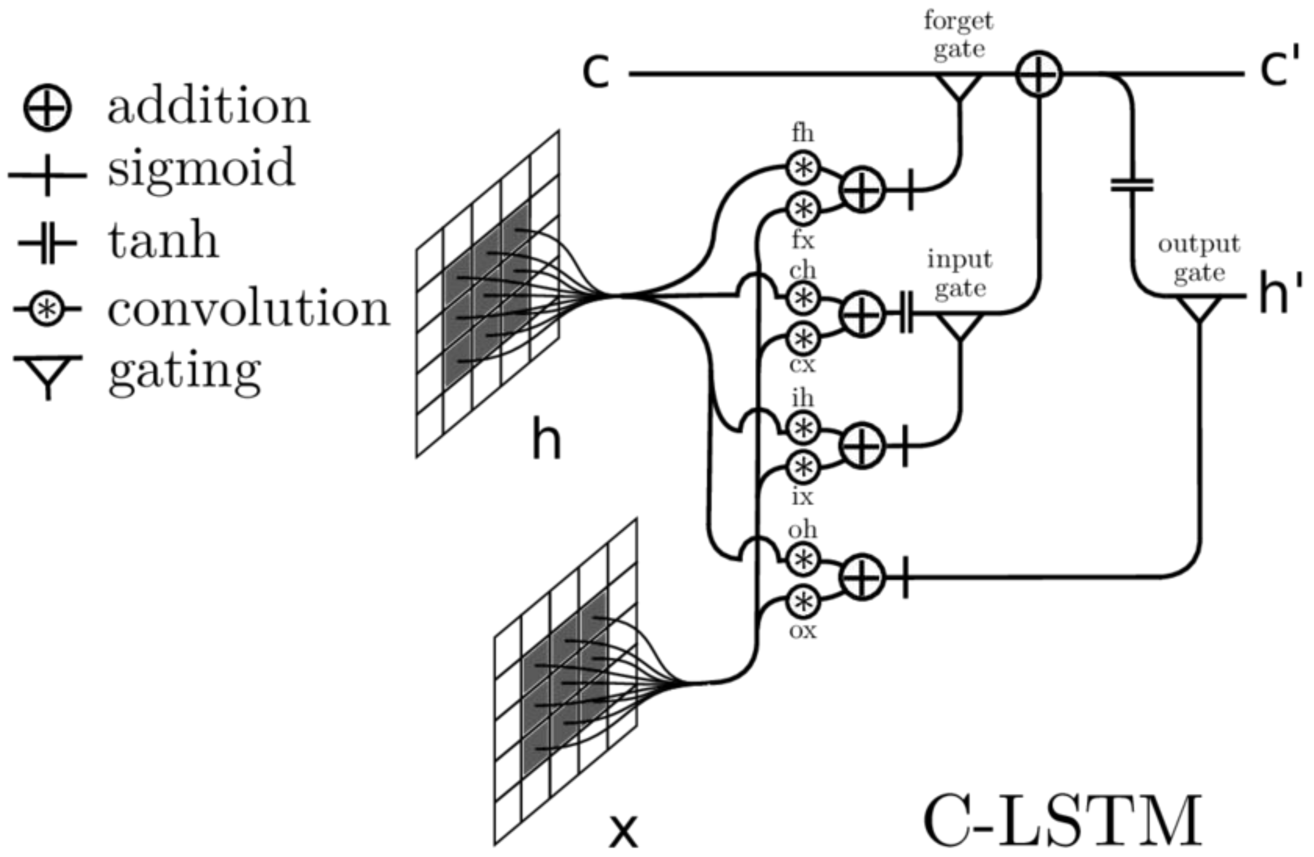
LSTM: Output gate layer

- The output is based on cell state, but filtered. First, we run a sigmoid layer which decides what parts of the cell state we’re going to output. Then, we pass the cell state through tanh (to make the values to be between −1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

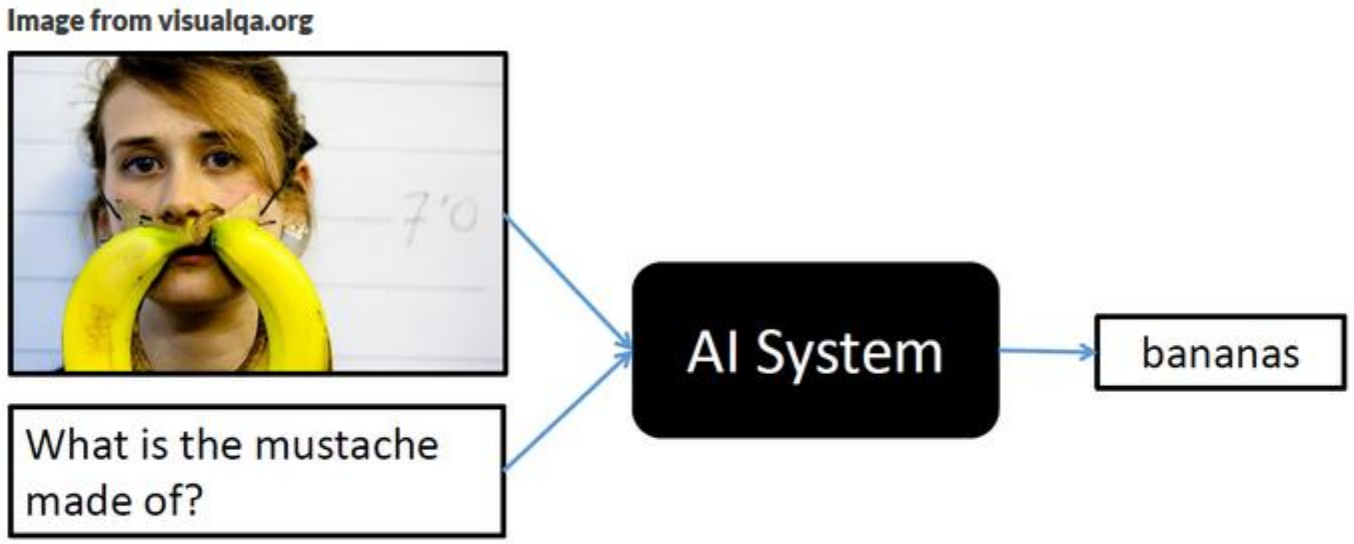


$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

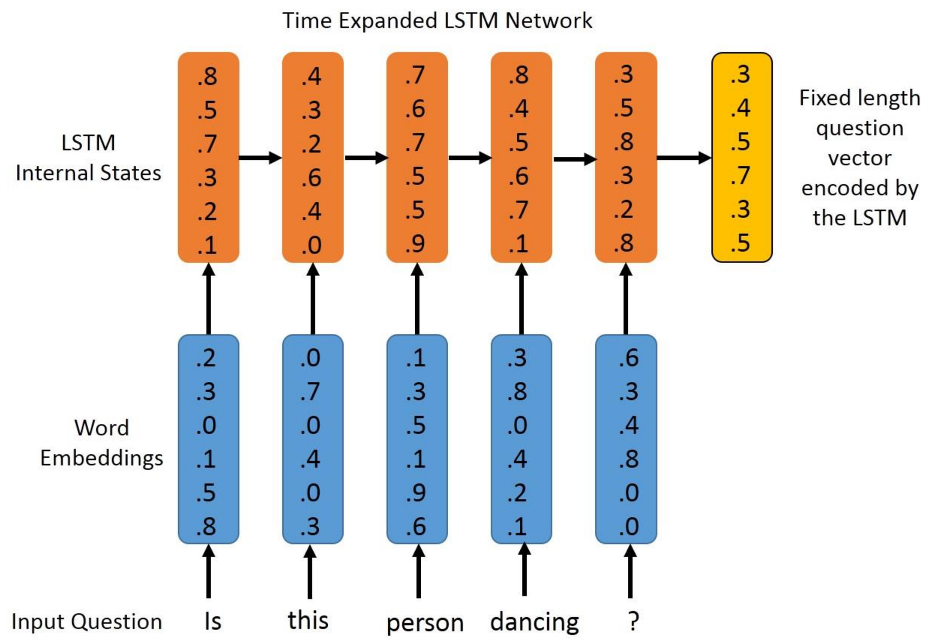
C-LSTM


addition
sigmoid
tanh
convolution
gating
c
forget gate
c'
fh
fx
ch
cx
ih
ix
oh
ox
input gate
output gate
h'
h
x
C-LSTM

Visual question answering



Visual question answering: LSTM



Model	Accuracy
BOW+CNN	44.30%
LSTM-Language only	42.51%
LSTM+CNN	47.80%

Hopfield neural network

- ✓ Recurrent neural network
- ✓ John Hopfield, 1982
- ✓ Has "associative" memory
- ✓ Guaranteed to converge to a local minimum
- ✓ 1 layer
- ✓ Fast training

Hopfield neural network

- Initial weights:

$$w_{ij} = \begin{cases} \sum_{k=0}^{m-1} x_i^k x_j^k, i \neq j \\ 0, i = j \end{cases}$$

- Predict:

1. An unknown signal is sent to the inputs of the network. Setting axon values:

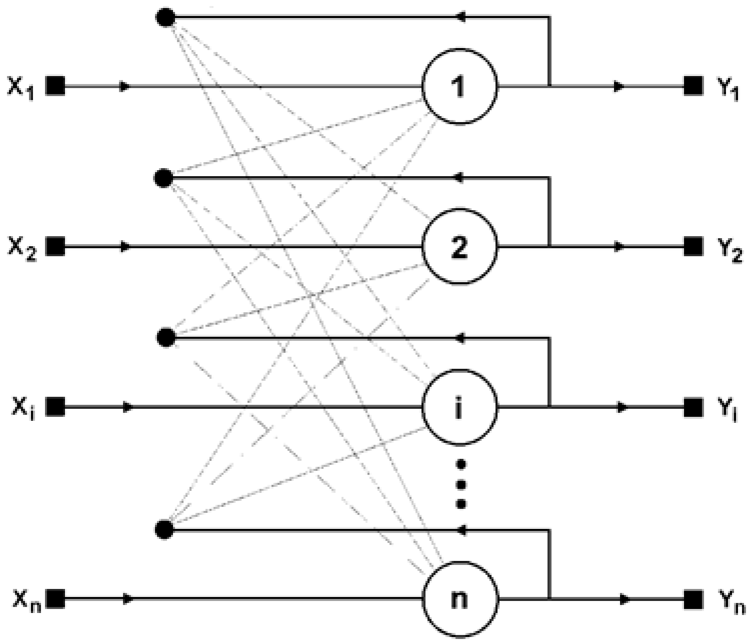
   $y_i(0) = x_i,\ i = 0 \ldots n\text{-}1,$

2. The new state of neurons and axons is calculated:

$$s_j(p+1) = \sum_{i=0}^{n-1} w_{ij} y_i(p)$$

$$y_j(p+1) = f\left[s_j(p+1)\right]$$

3. Check if the output values of the axons have changed during the last iteration. If yes - go to point 2, otherwise the end



ITMO UNIVERSITY

Hamming neural network

- Hamming distance from the tested image to all samples
- 2 layers
- Initial weights:

$$w_{ik} = \frac{x_i^k}{2},\ i=0...n\text{-}1,\ k=0...m\text{-}1$$

$$T_k = n/2,\ k = 0...m\text{-}1$$

- Predict:

1. An unknown signal is sent to the inputs of the network. Setting axon values:

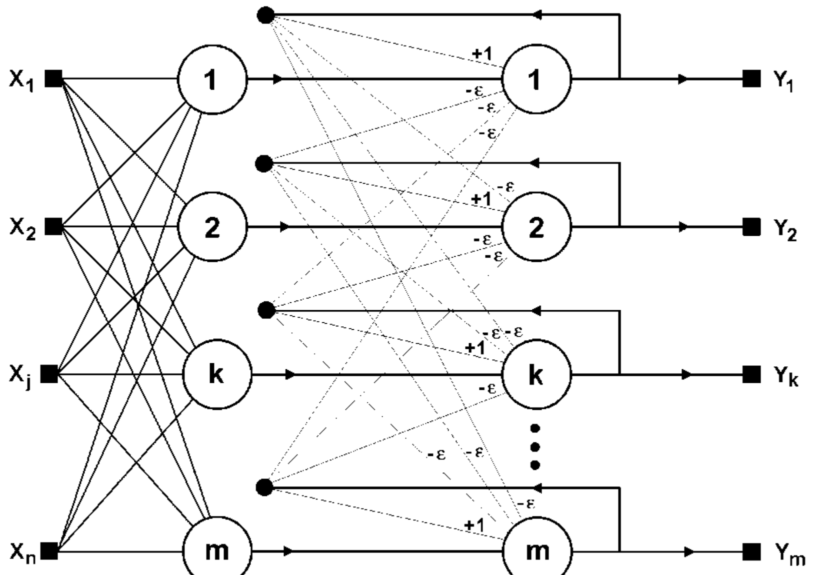
$$y_j^{(1)} = s_j^{(1)} = \sum_{i=0}^{n-1} w_{ij} x_i + T_j \qquad y_j^{(2)} = y_j^{(1)},\ j = 0...m\text{-}1$$

2. The new state of neurons and axons is calculated:

$$s_j^{(2)}(p+1) = y_j(p) - e\sum_{k=0}^{m-1} y_k^{(2)}(p), k{}^1 j,\ j = 0...m-1$$

$$y_j^{(2)}(p+1) = f\left[s_j^{(2)}(p+1)\right],\ j = 0...m-1$$

3. Check if the output values of the axons have changed during the last iteration. If yes - go to point 2, otherwise the end



Materials

Presentation was prepared using:

1. http://avisingh599.github.io/deeplearning/visual-qa/
2. http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Task

1. Implement Hopfield or Hamming network on Python
2. Train it on MNIST dataset (~0.15n samples)

Thanks for attention!

Questions?