Sapienza University of Rome

Master in Artificial Intelligence and Robotics
Master in Engineering in Computer Science

Machine Learning

A.Y. 2019/2020

Prof. L. Iocchi, F. Patrizi, V. Ntouskos

13. Complements of Neural Networks

L. Iocchi, F. Patrizi, V. Ntouskos

Overview

- Working with sequences
- Transfer Learning

References

Ian Goodfellow and Yoshua Bengio and Aaron Courville. Deep Learning - Chapter 10. http://www.deeplearningbook.org

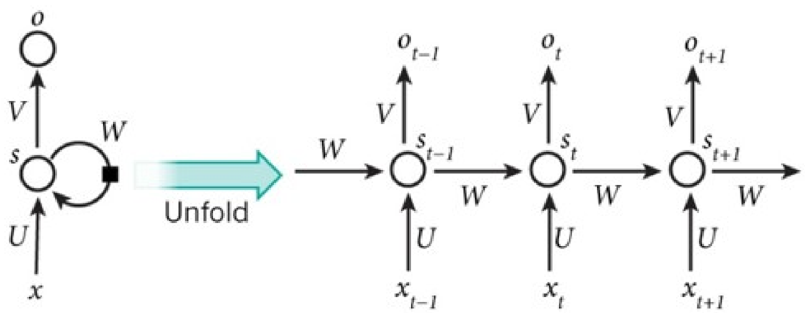
Working with sequences

Recurrent Networks for Sequences

Recurrent Networks and Long Short-Term Memories (LSTM) are models suitable for sequential data as

- language
- audio
- motion / dynamics
- videos

Trained using the back-propagation through time (BPTT) algorithm



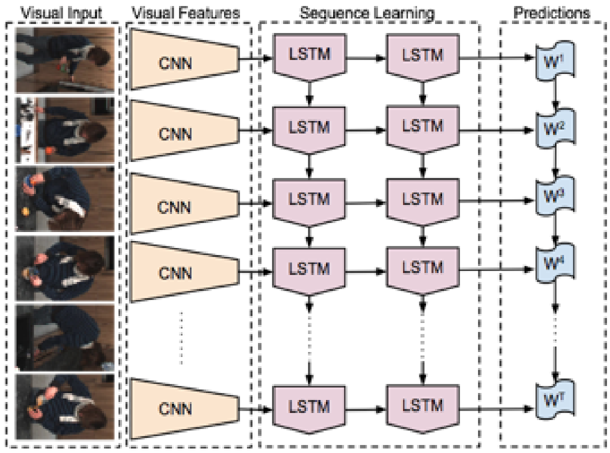
Working with sequences

Example

LRCN: Long-term Recurrent Convolutional Network

- activity recognition (sequence-in)
- image captioning (sequence-out)
- video captioning (sequence-to-sequence)

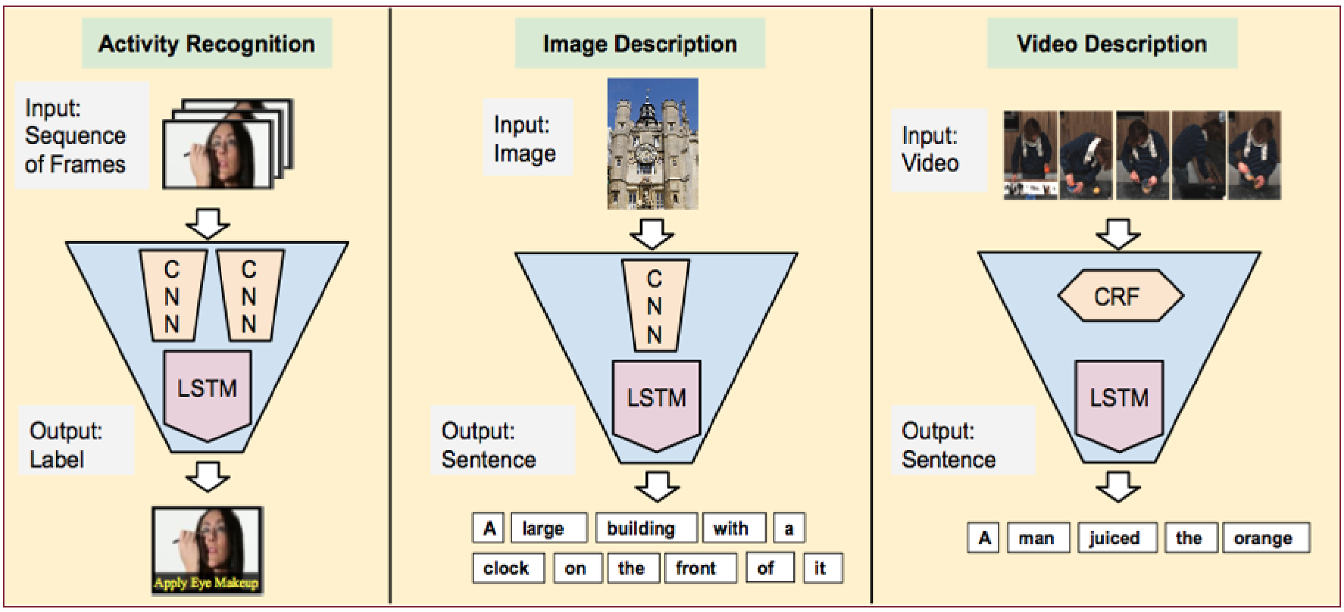
Recurrent + convolutional units for processing visual sequences



LRCN for video captioning

Working with sequences

Visual Sequence Tasks



Jeff Donahue et al. CVPR'15

RNNs

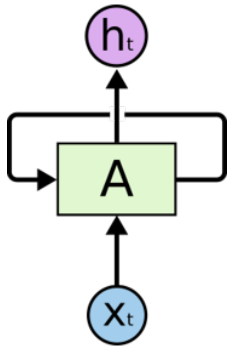
Dealing with sequences

- data points are related
- order is important!

(C)NNs disregard this information

Recurrent Neural Networks (RNNs) address this problem by introducing cells with loops

- allow information to persist
- the value of output h_t depends both on x_t and the previous output h_{t-1}

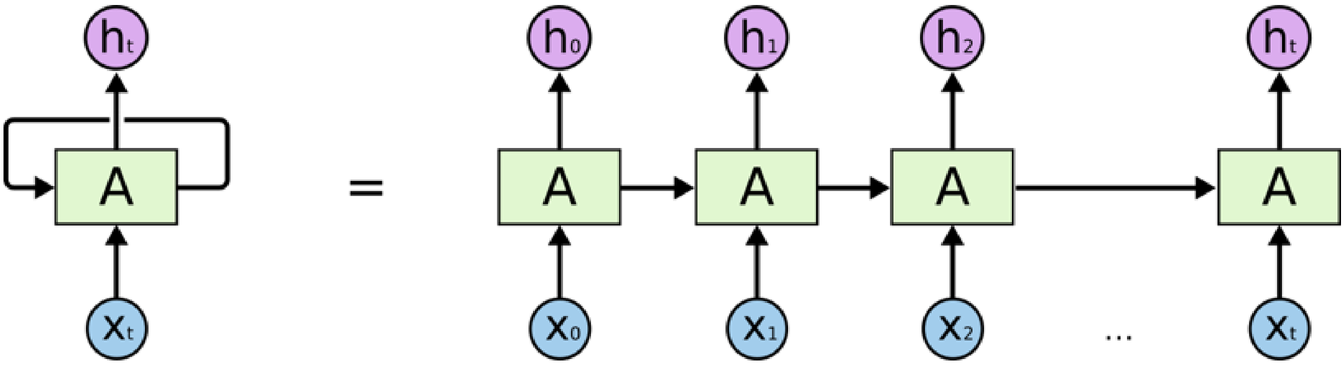


Sapienza University of Rome, Italy - Machine Learning (2019/2020)

RNNs

Another way to see RNNs is to unfold the loop

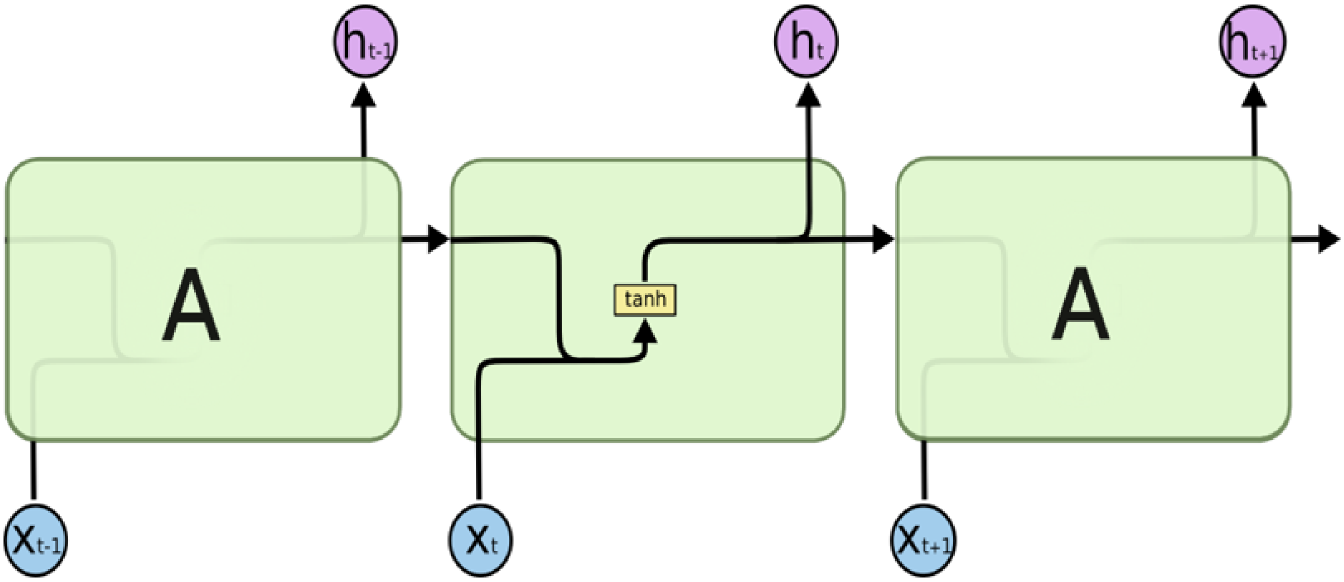
- each cell dedicated to a different data sample
- output h_t computed based also on h_{t-1}



Graphics taken from C. Olah's blog

RNNs

RNN structure



Graphics taken from C. Olah's blog

RNNs

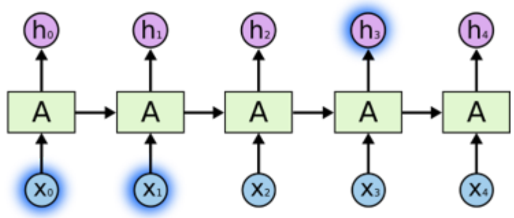
Problem

RNNs cannot capture long-term dependencies

- vanishing/exploding gradients problem

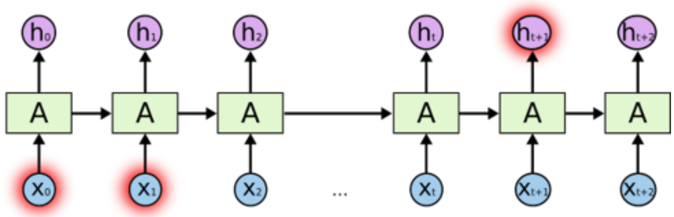
Example:

a) "The clouds are in the _."



Missing word directly after relative information

b) "I grew up in France... I speak fluently _."



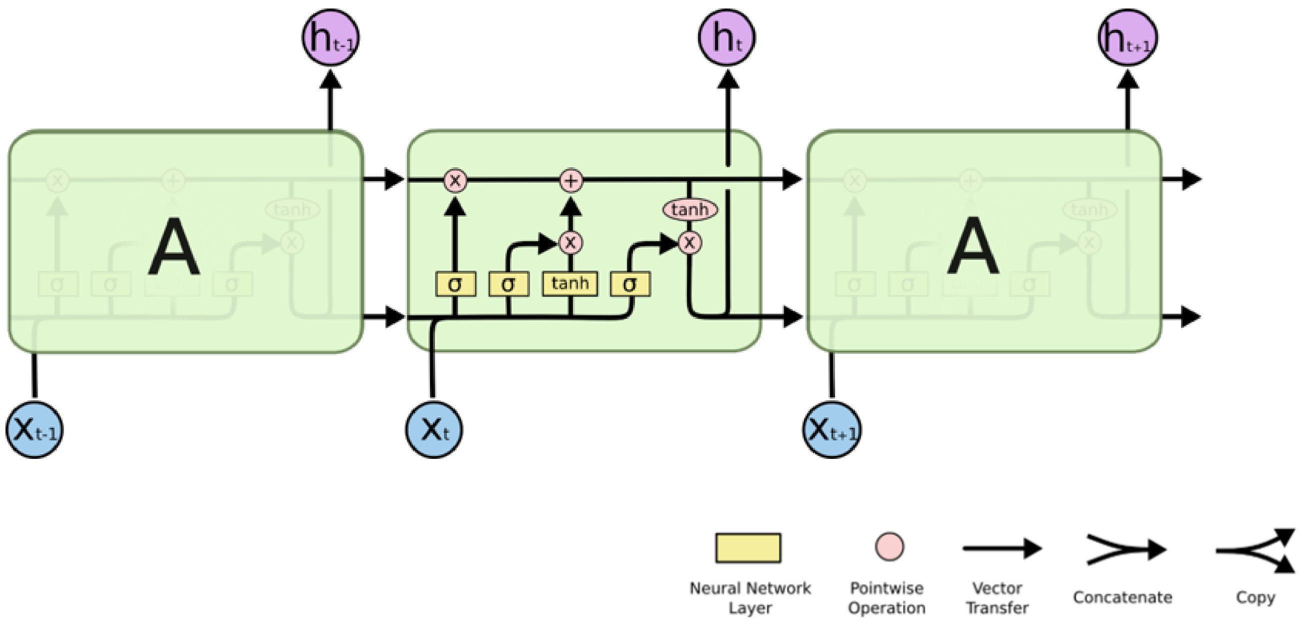
Missing word long after relative information

Graphics taken from C. Olah's blog

LSTMs

Long Short Term Memories (LSTMs)

- RNNs with special structure
- designed to capture long-term dependencies

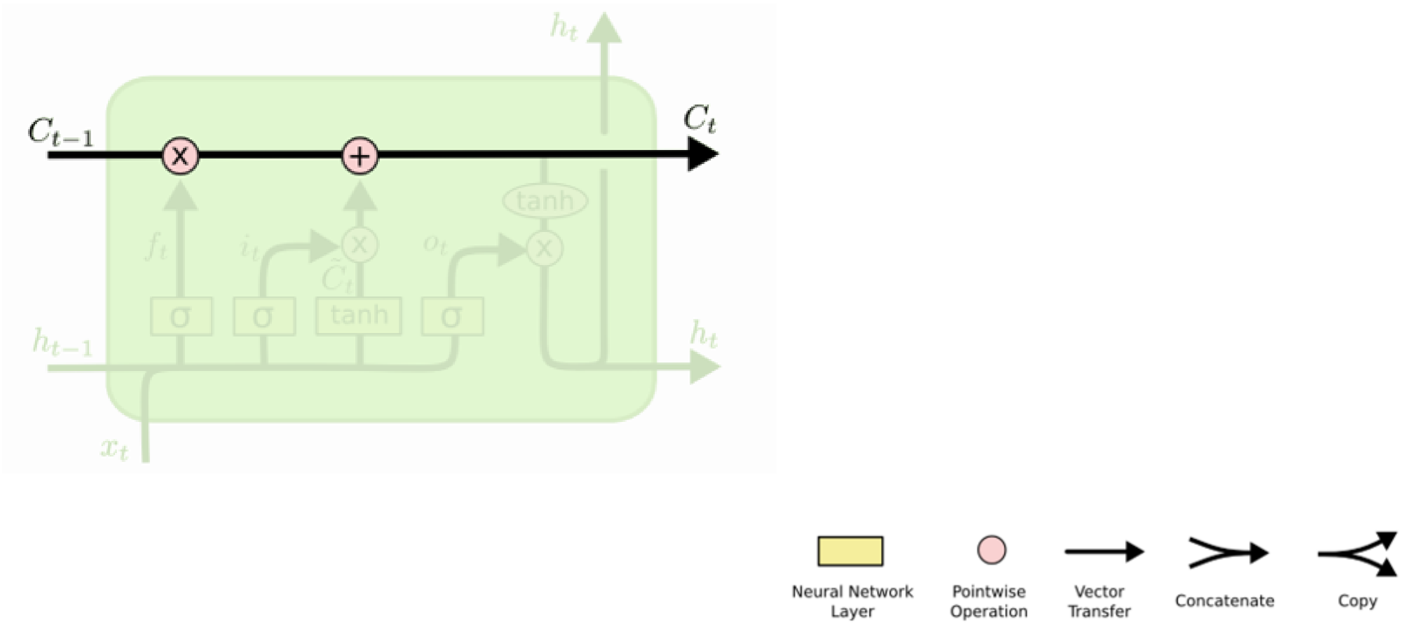


Graphics taken from C. Olah's blog

LSTMs

Main idea: Separate cell output h_t and cell state C_t

- state is modified through a series of structures called gates

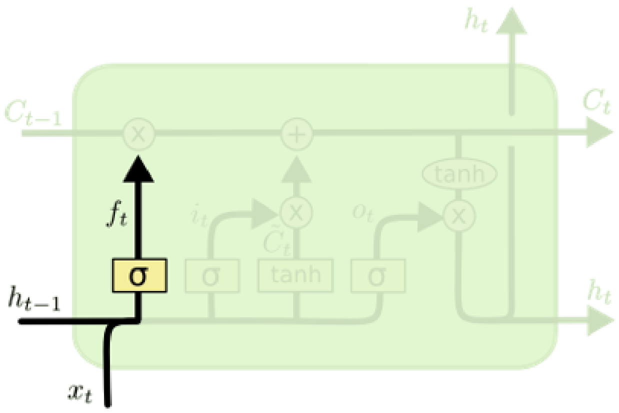


Graphics taken from C. Olah's blog

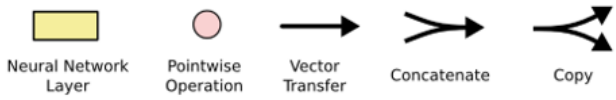
LSTMs

1st gate: *forget* mechanism

- drop elements of C_{t-1} based on values of h_{t-1} and input x_t



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

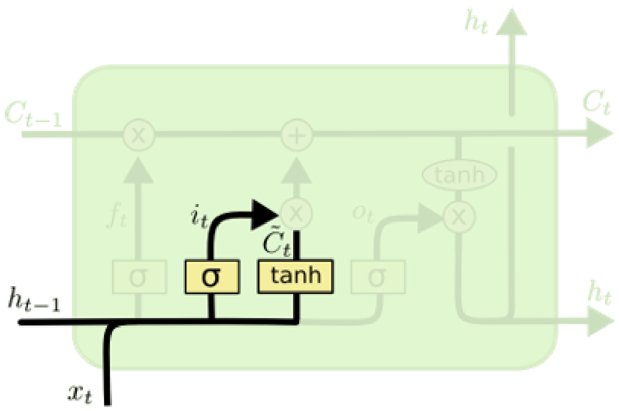


Graphics taken from C. Olah's blog

LSTMs

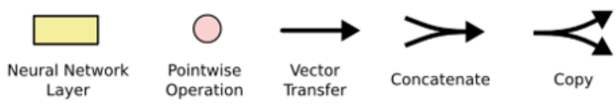
2nd and 3rd gate: *update* mechanism

1. decide which elements of C_{t-1} to update
2. compute the update vector $\tilde{C}_t$



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

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

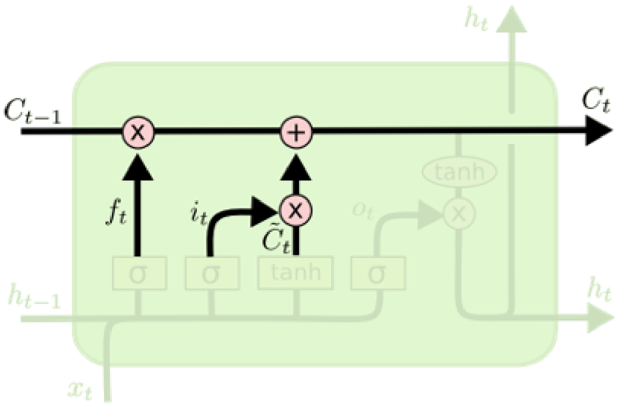


Graphics taken from C. Olah's blog

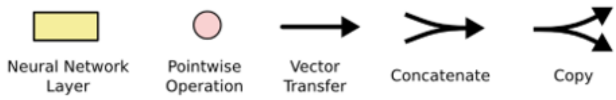
LSTMs

2nd and 3rd gate: *update* mechanism

3. update elements selected via i_t with values computed in $\tilde{C}_t$



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



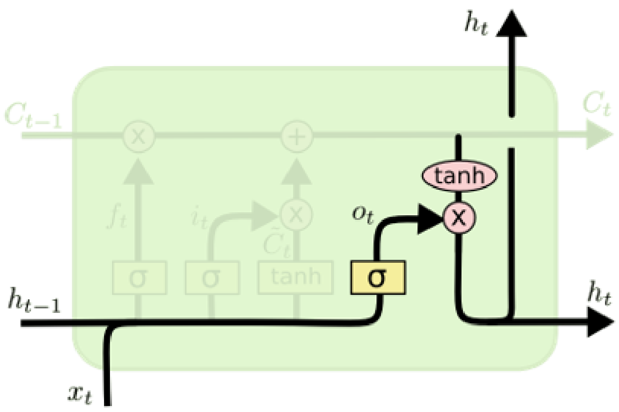
Graphics taken from C. Olah's blog

Sapienza University of Rome, Italy - Machine Learning (2019/2020)

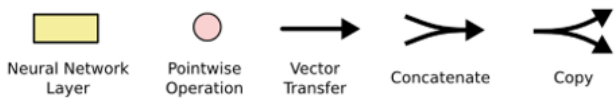
LSTMs

Last step: compute *output*

- Based on C_t, h_{t-1}, x_t



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



Graphics taken from C. Olah's blog

LSTMs - typical use cases

Visual Sequence Tasks

- One-to-one: classical NNs (no RNN)
- One-to-many: sequence output (e.g. image captioning)
- Many-to-one: sequence input (e.g. video classification)
- Many-to-many: sequence to sequence (e.g. machine translation)
- Many-to-many synced: e.g. frame per frame video analysis

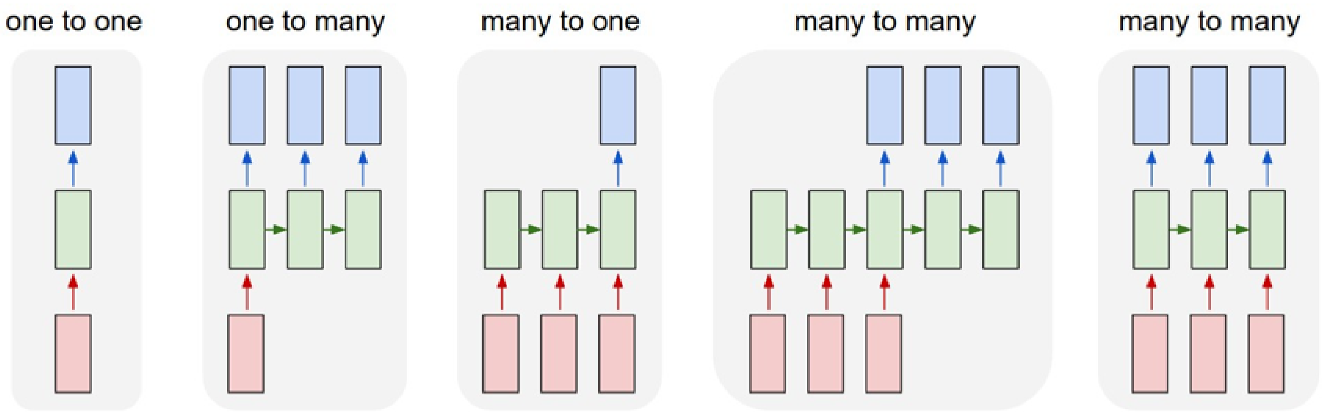


Image credit: A. Karpathy

Transfer Learning

Definitions

- $\mathcal{D}$ is a *domain* defined by data points $X \in \mathcal{X}$ distributed according to $P(X)$
- $\mathcal{T}$ is a *learning task* defined by labels $Y \in \mathcal{Y}$ and a target function $f : \mathcal{X} \mapsto \mathcal{Y}$.

Given

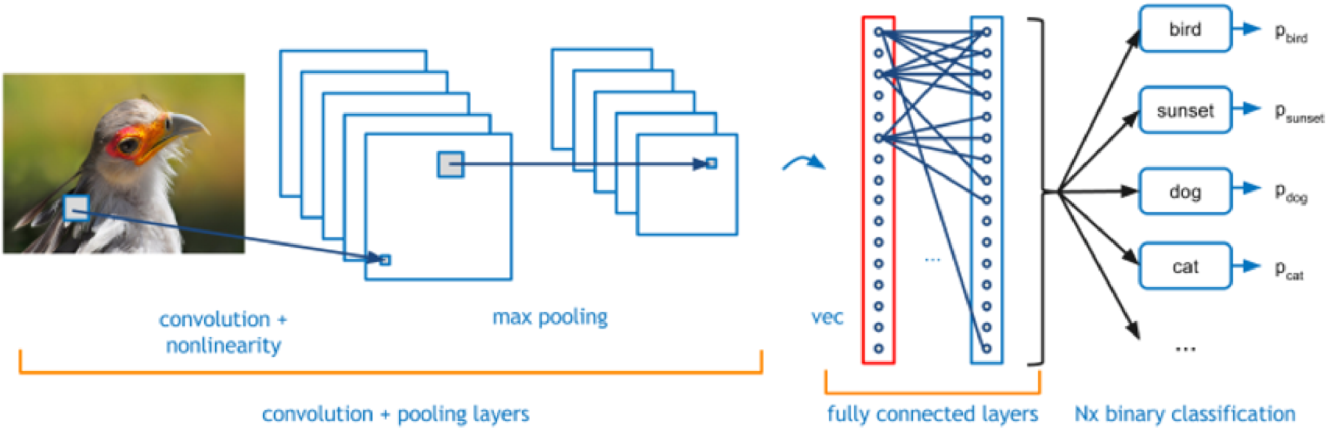
- $\mathcal{D}_S$ and $\mathcal{T}_S$ a source domain and learning task
- $\mathcal{D}_T$ and $\mathcal{T}_T$ a target domain and learning task

Goal

improve learning of $f_T : \mathcal{X}_T \mapsto \mathcal{Y}_T$ using knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$

Transfer Learning - Example

Image classification using a CNN



CNN pre-trained on Imagenet

- millions of images (source domain)
- classification of 1000 classes (source learning task)

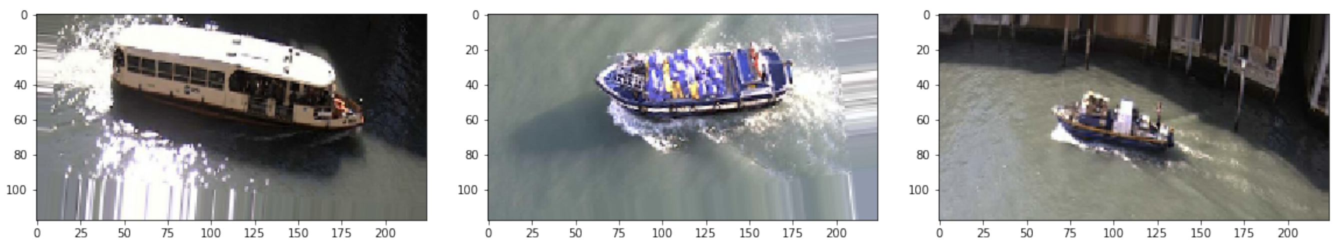
Slide from from Caffe framework tutorial @CVPR2015

Transfer Learning - Example

Image classification using a CNN

Use a pre-trained model for a different domain and/or learning task
E.g. Boat recognition in ARGOS dataset:

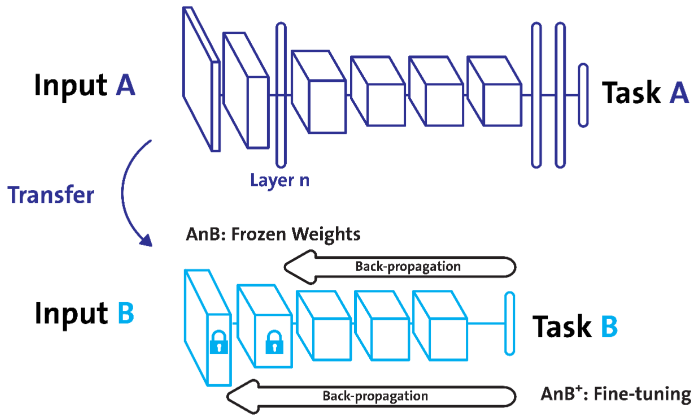
- thousands of boat images (target domain)
- classification of 20 boat classes (target learning task)



Transfer Learning - Example

1st solution - Fine-tuning

- use same network architecture with pre-trained model
- network parameters ‘copied’ from the pre-trained model
- no random initialization



Strategies

- training of all network parameters
- ‘freeze’ parameters of some layers (usually the first ones)

Pro Full advantage of the CNN!
Con ‘Heavy’ training

Image from P. Gudikandula’s blog

Transfer Learning - Example

2nd solution - CNN as feature extractor

1. extract features at a specific layer of CNN, usually:
   - last convolutional layer (flattened)
   - dense layers
2. collect extracted features $\mathbf{x}'$ of training/validation split and associate corresponding labels t in a new dataset $D' = \{(\mathbf{x}'_1, t_1), \dots, (\mathbf{x}'_N, t_n)\}$
3. train a new classifier C' using dataset D', e.g.
   - ANN (extreme case of fine-tuning)
   - SVM
   - linear classifier
   - ...
4. classify extracted features of test set using the classifier C'

Pro No need to train the CNN!
Con Cannot modify features, source and target domains should be as ‘compatible’ as possible

Transfer Learning - Example

2nd solution - CNN as feature extractor

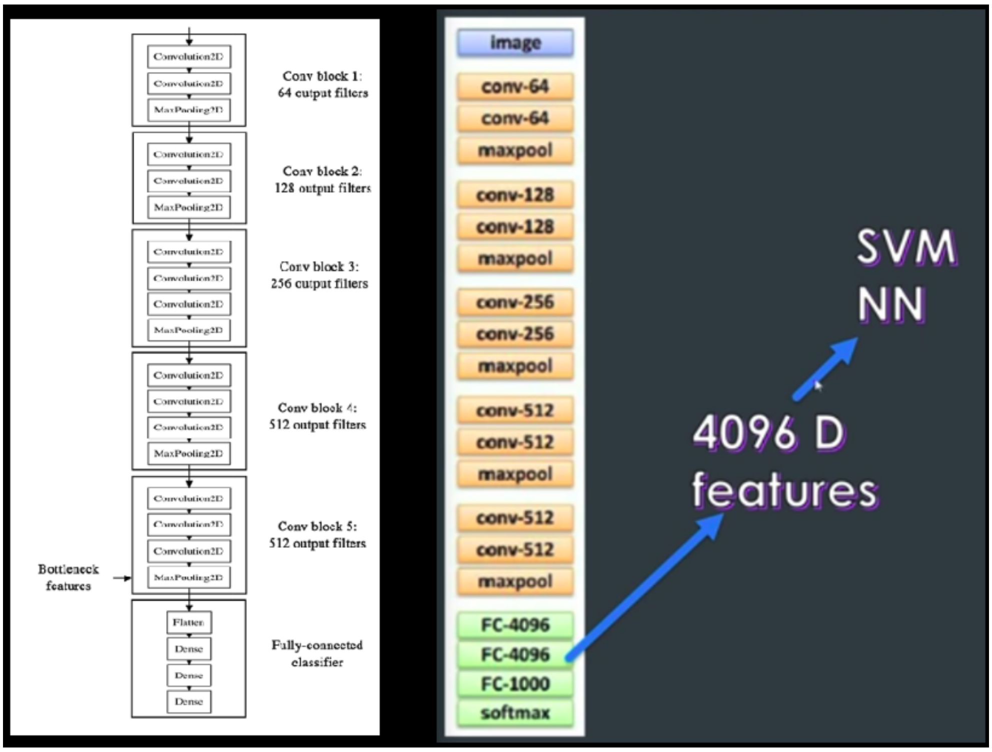


Image from P. Gudikandula's blog