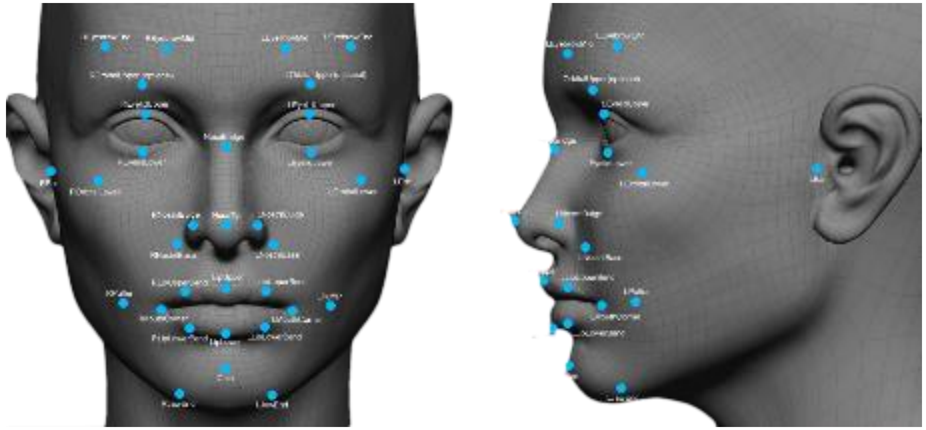


Convolutional neural networks. Part 2

Lecture # 9 of MIPT course: Introduction to machine learning

Speaker: Alexey O. Seleznev
PhD in Computational Chemistry

5VISION TEAM

Moscow
2017

OUTLINE

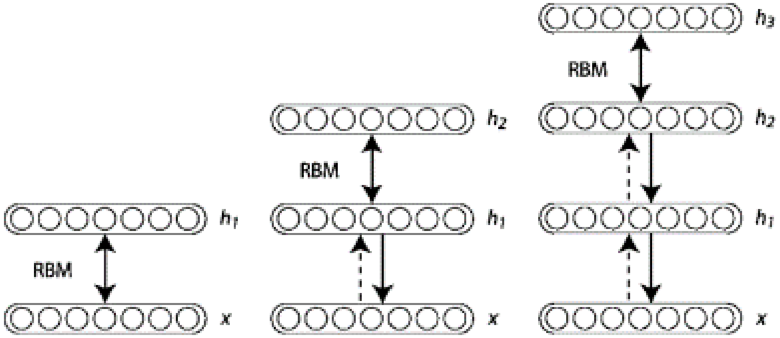
- Pre-training
  - Pre-training for deep neural networks
  - Transfer learning
- Deconvolutional layers
- Convolutional filter visualization
- Convolutional networks for time series analysis
- Discussion on Assignment # 2
- References

Pre-training

Pre-training for Deep Neural Networks

❖ **The idea**:

- a way of weight initialization in deep networks by subsequent training of stacked unsupervised layers (denoising autoencoders, restricted Bolzmann machines)



- The weights are then fine-tuned using the standard backpropagation procedure

❖ Historically, pre-training was a technique that tremendously accelerated the development of deep learning due to its ability to alleviate the vanishing/exploding gradient problem

Pre-training for Deep Neural Networks

- After such innovations in purely supervised training as ReLu activation function and Glorot initialization, it has become possible to train deep networks without unsupervised pre-training
- Nonetheless, it does not mean that pre-training is senseless:
  - pre-training is a powerful regularization technique (underfitting -> worse results; overfitting -> better results)
  - pre-training let us start from the better basin of attraction, the advantage of great importance for non-convex optimization problems
  - variance reduction technique
- Pre-training resolves the chicken-and-egg dilemma: the lower layers of a supervised deep architecture need the upper layers to define what they should extract, and vice versa

Erhan et al (2010)

Transfer Learning

- It is extremely rare when unsupervised pre-training techniques are applied to convolutional networks:
  - the necessity to build a convolutional decoder for autoencoder or (if we use RBM) non-obvious way to build RBM
  - existence of a simpler technique: transfer learning
- The **idea** behind Transfer Learning is to initialize weights of convolutional layers with weights of convolutional layers of other deep networks that have been already trained.
- Three variants of Transfer Learning
  - Fixed feature extractor mode (copy convnet weights and fix them, train only the classifier on the top of the model)
  - Fine-tuning mode (copy convent weights, train the whole model)
  - Fully pre-trained mode (copy weights for the whole model)

Transfer Learning

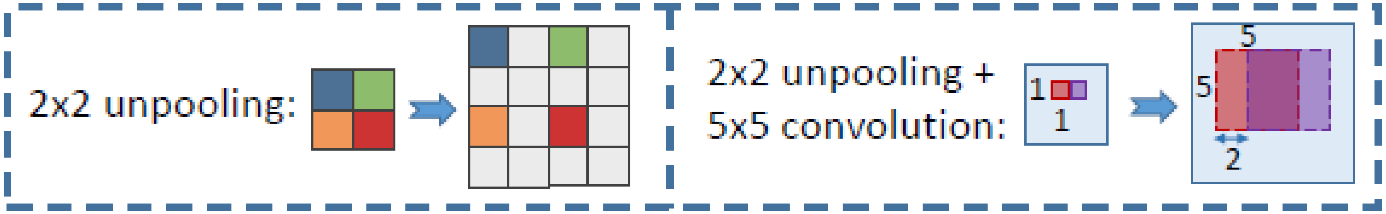
- Which mode to use?
  - Two factors: the size of the new dataset and how similar the new one is to the old one
  - 4 situations:
    - New dataset is small and similar to the original one. Fixed feature extractor mode
    - New dataset is large and similar to the original dataset. Fine-tuning mode or fully pre-trained model
    - New dataset is small but very different from the original one. Take weights of first few convolutional layers
    - New dataset is large and very different from the original one. Either train from scratch or fine-tuning mode (if we have enough data)

http://cs231n.github.io/transfer-learning/

Deconvolutional layers

Deconvolutional Layers

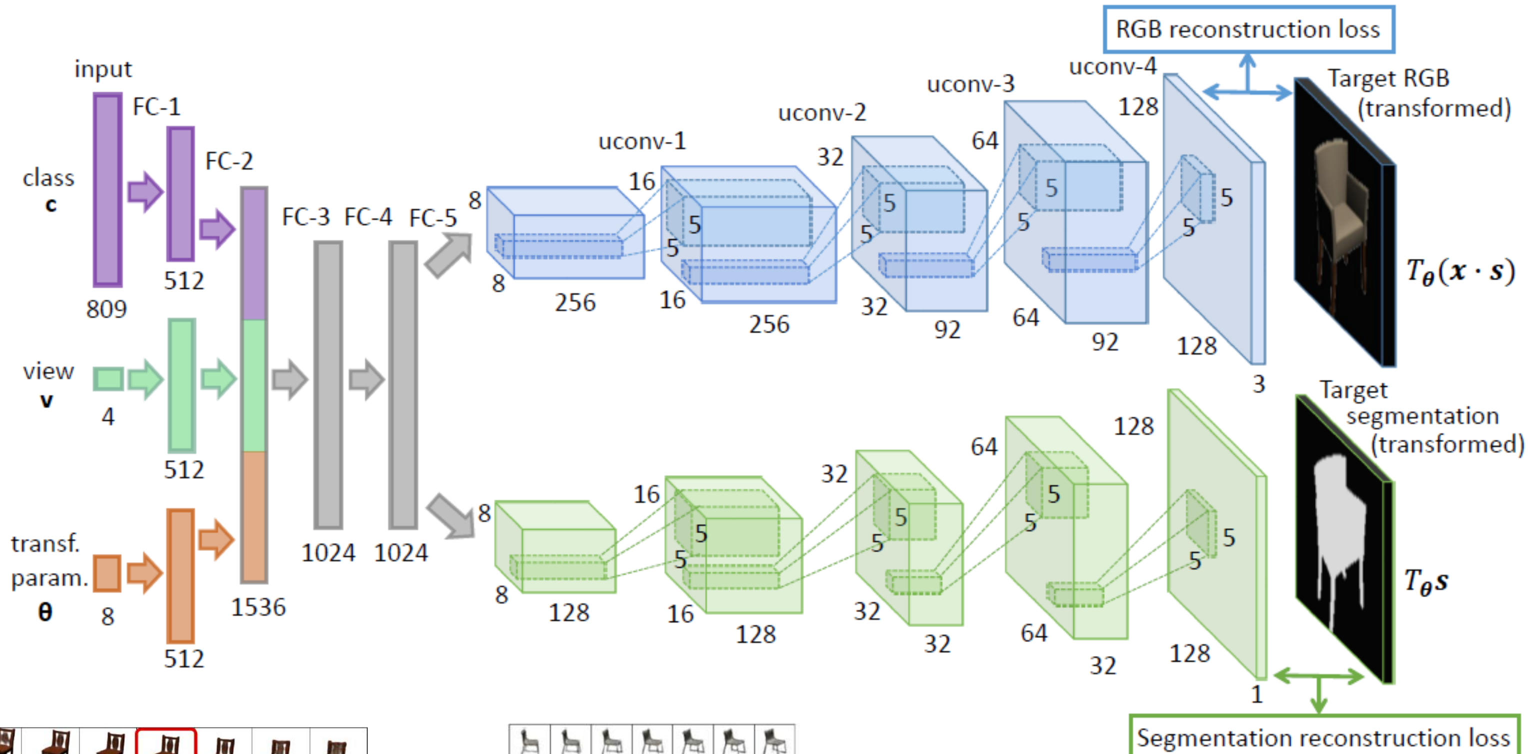
- Why do we need an inverse operation to convolution?
  - Build an autoencoder
  - Visualize filters
- How to create a deconvolutional network?
  - Unpooling + convolution layer
  - Unpooling + ‘untrainable’ deconvolution layer
  - Trainable deconvolutional layers
- The simplest variant is to build a mix of unpooling and convolutional layers.

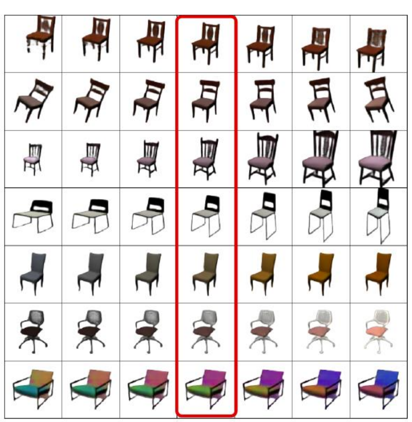


Dosovitskiy et al (2015)

Deconvolutional Layers

❖ Example of the first type model



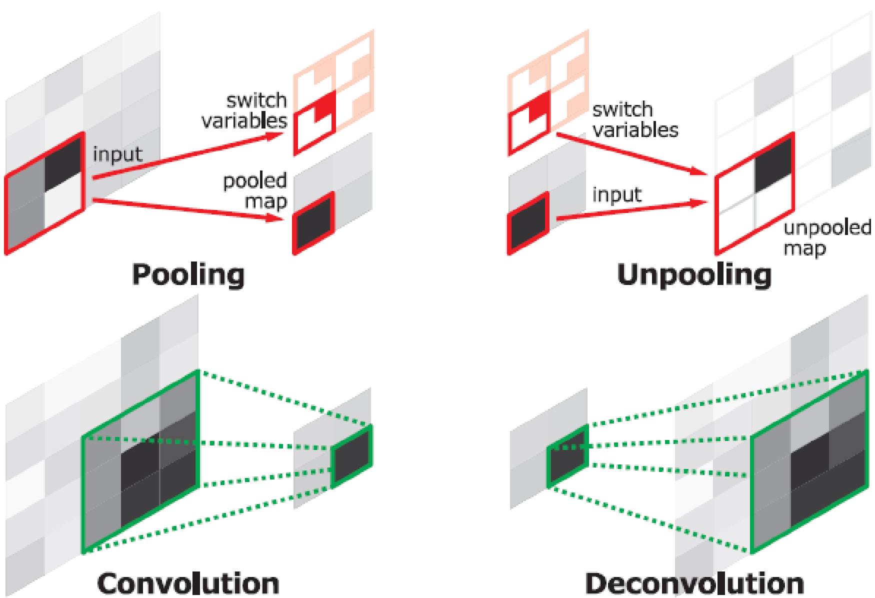


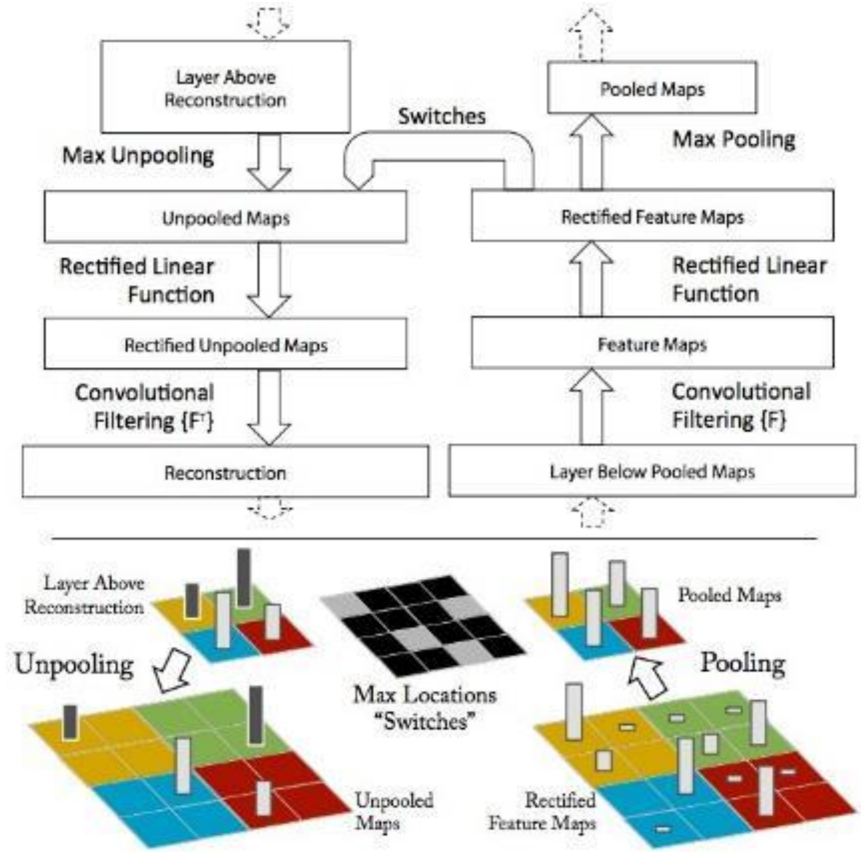
Dosovitskiy et al (2015)

Deconvolutional Layers

❖ Second variant is usually used for visualization of filter weights and contains three operations:

- Unpooling
- Rectified
- Deconvolution

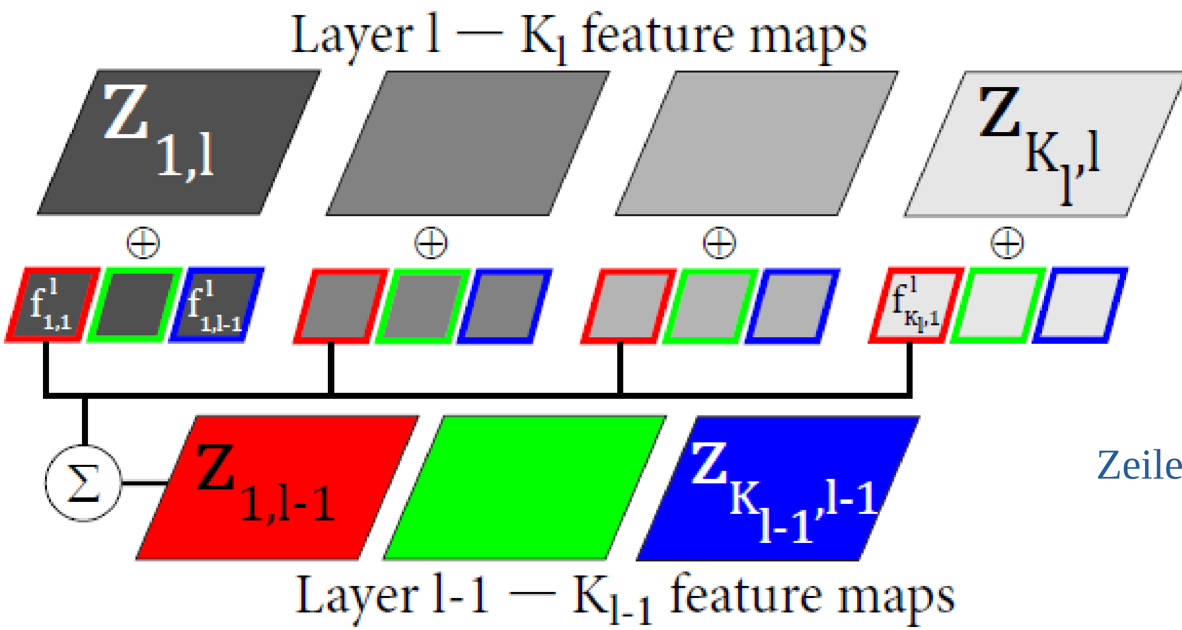




Zeiler et al (2013)

Deconvolutional Layers

- Third variant is purely unsupervised and deals with convolution filter generation and visualization



Zeiler et al (2010)

$$C_l(y) = \frac{\lambda}{2}\sum_{i=1}^{I}\sum_{c=1}^{K_{l-1}} \|\sum_{k=1}^{K_l} g^l_{k,c}(z^i_{k,l} \oplus f^l_{k,c}) - z^i_{c,l-1}\|_2^2 + \sum_{i=1}^{I}\sum_{k=1}^{K_l} |z^i_{k,l}|^p$$

Z – feature map
l – layer
f – filters
c – channel
g – connectivity matrix

Break 5 min

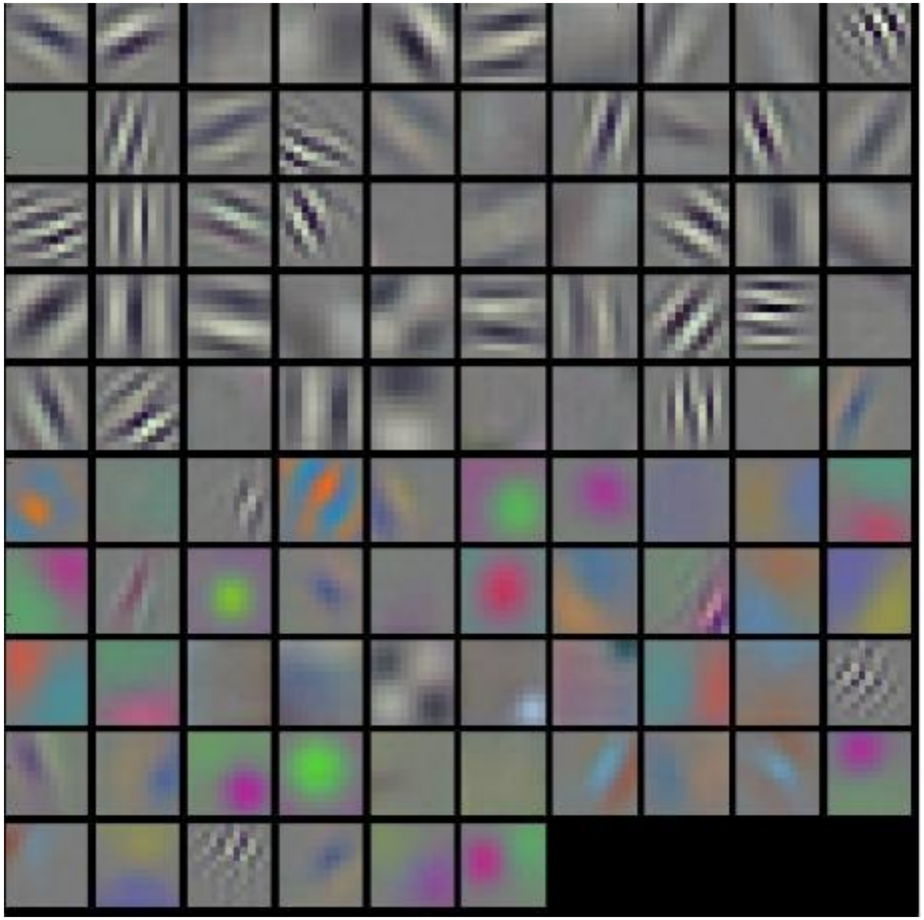
Convolutional Filter Visualization

Convolutional Filter Visualization

- Motivation:
  - Get insights into the learning process
  - Image processing
  - Pre-training
- The simplest approach to visualize filters
  - Fix weights for the filters under study and weights of all layers between them and the image
  - Present image as a noised background with variable values
  - Construct loss-function as a mean value of the filter under study
  - Optimize the image to minimize loss function
- The method suggested only displays weights, but do establish connections between the input image and the filter (no feedback)
- Deconvolutional networks solve this problem
- Moreover, unsupervised deconvolutional networks can generate filters suitable for pre-training

Convolutional Filter Visualization

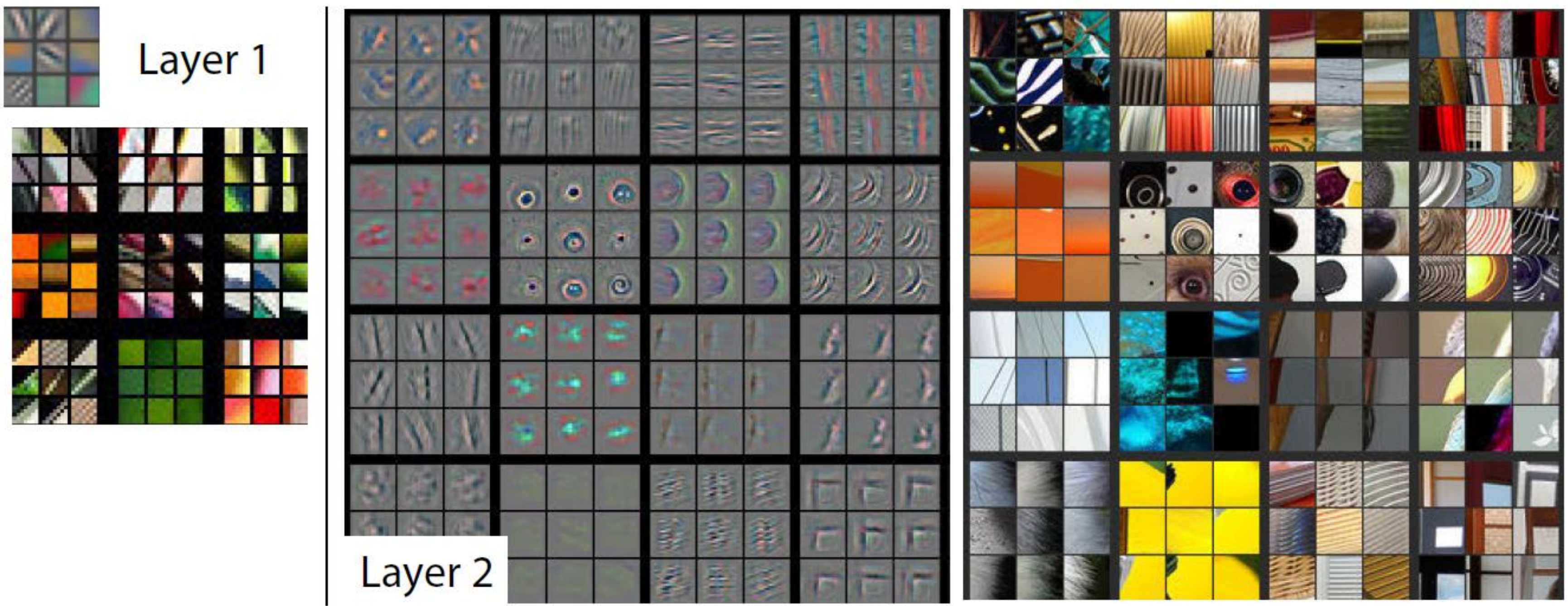
❖ Examples of the first layer filters from AlexNet:



http://cs231n.github.io/understanding-cnn/

Convolutional Filter Visualization

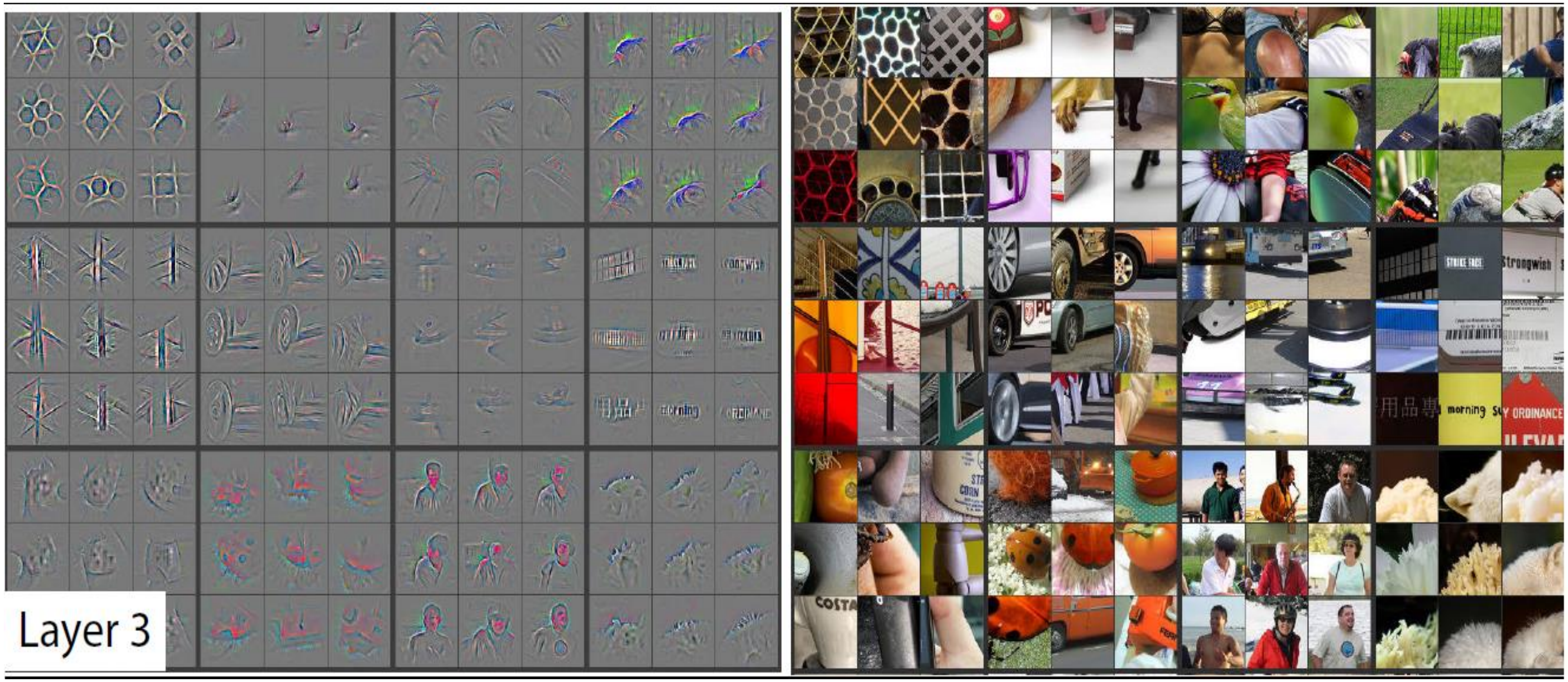
❖ Weights visualization by supervised deconvolutional networks (increase in the level of abstraction):



Zeiler et al (2013)

Convolutional Filter Visualization

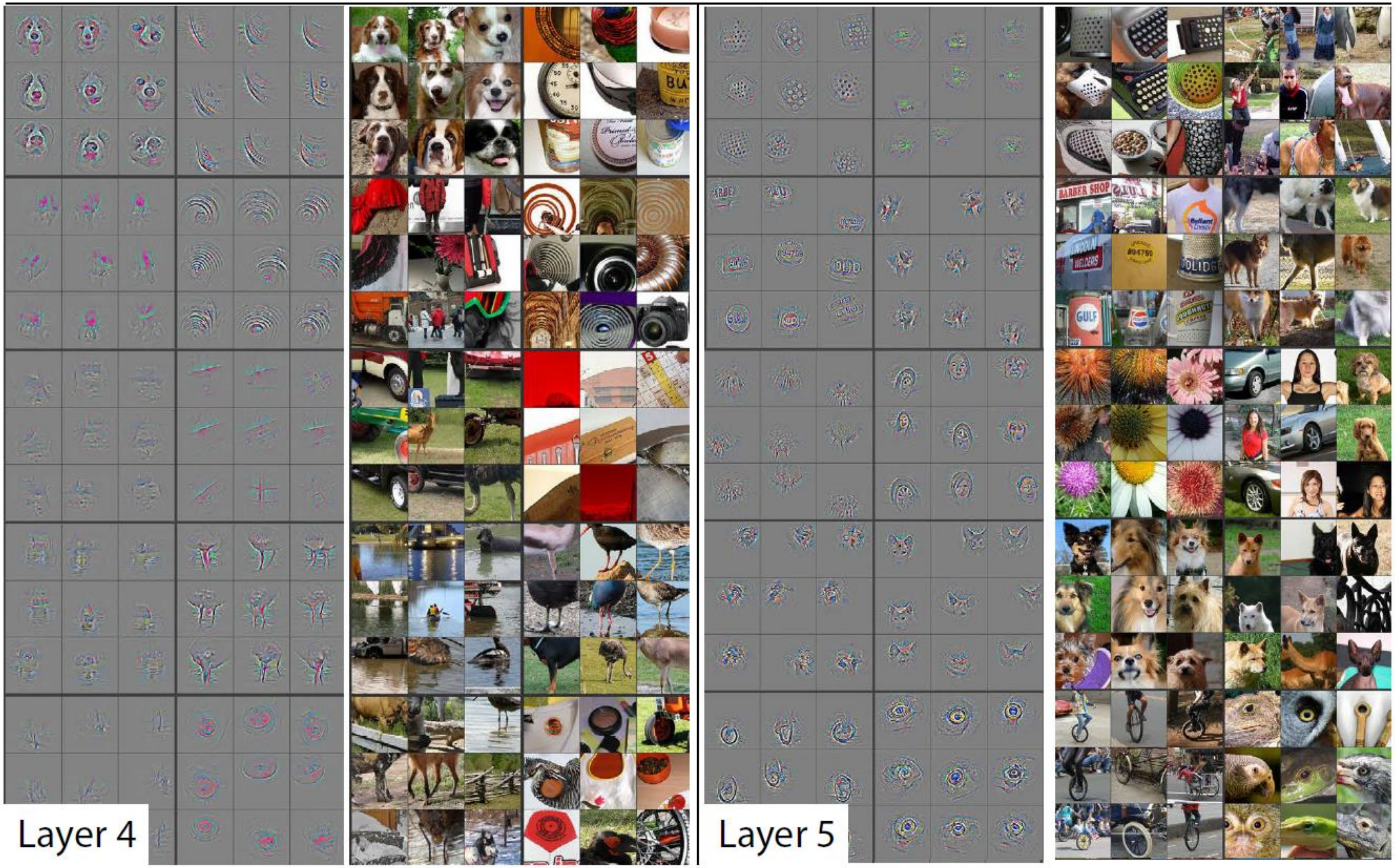
(continue)



Zeiler et al (2013)

Convolutional Filter Visualization

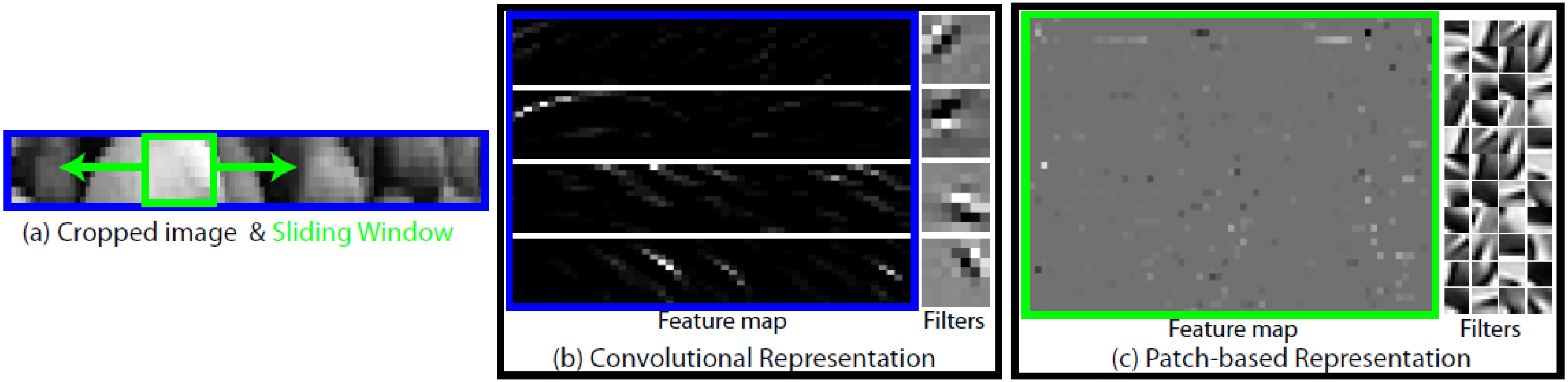
(continue)

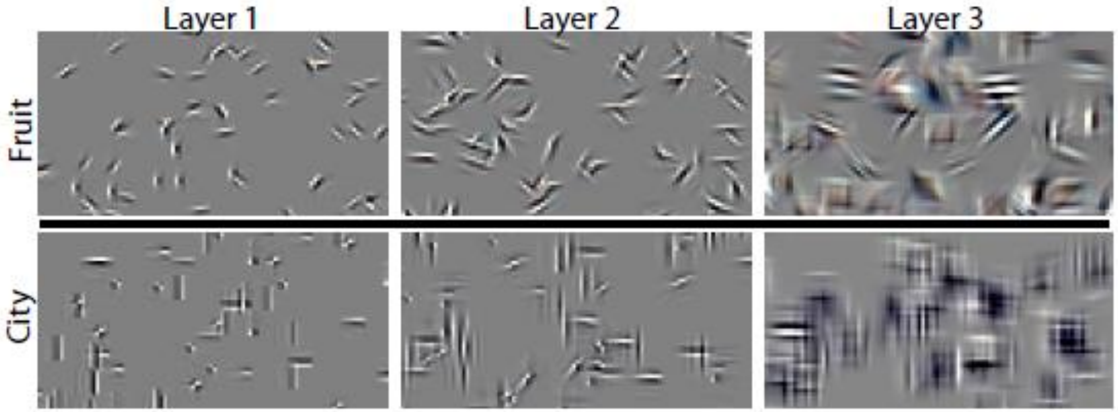


Zeiler et al (2013)

Convolutional Filter Visualization

❖ Unsupervised weights:

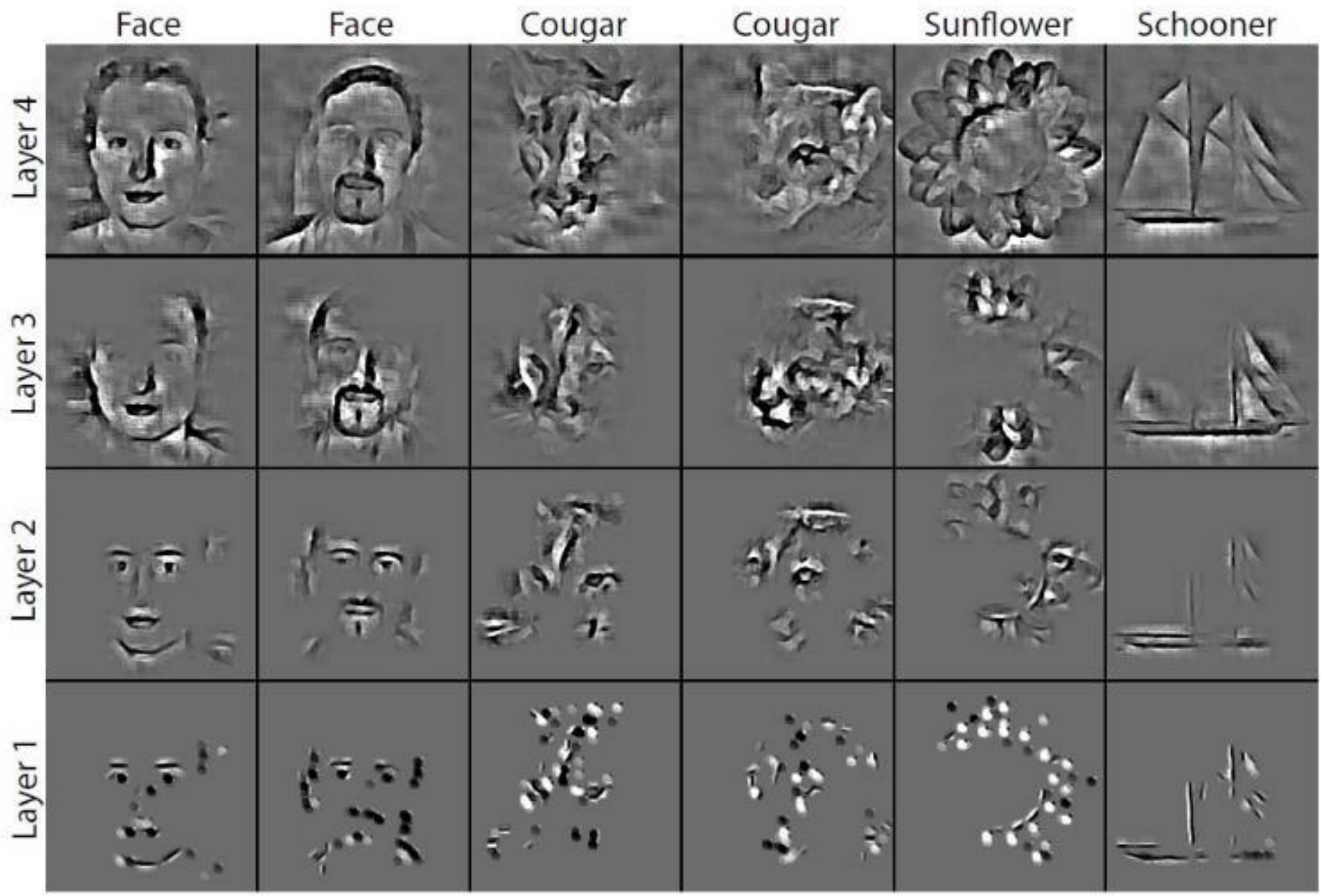




Zeiler et al (2010)

Convolutional Filter Visualization

❖ Visualization as a way of image processing:

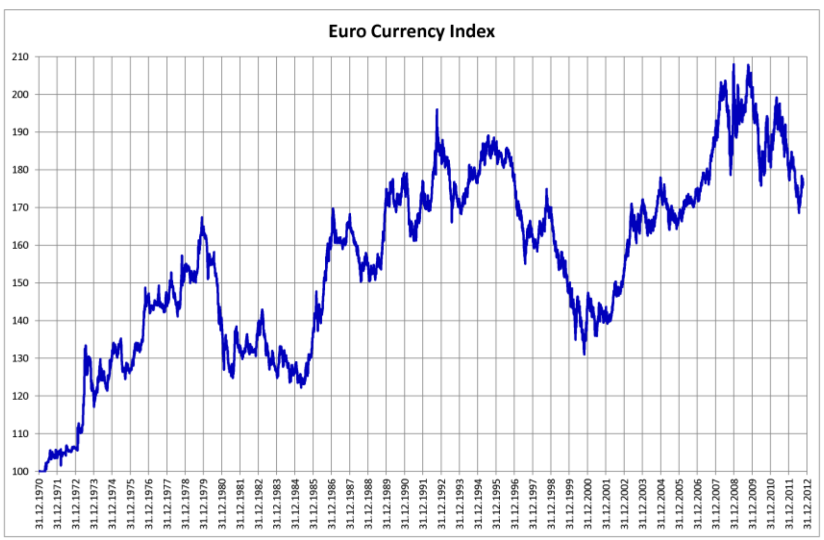


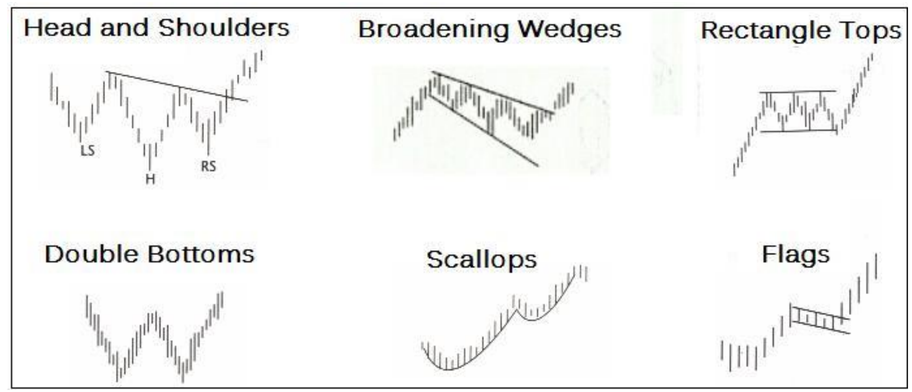
Zeiler et al (2013)

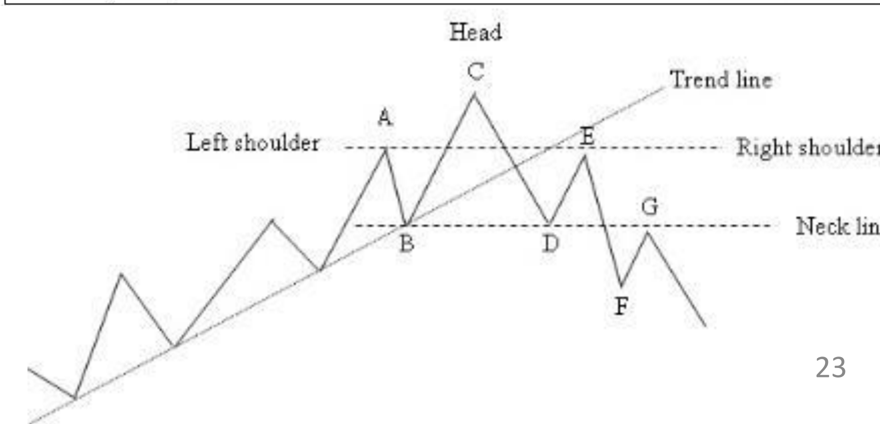
Convolutional Networks for Time Series Analysis

Convolutional Networks for Time Series Analysis

- CNN can be applied to analyzing time series. It is possible because in time series we are interested in finding patterns invariant w.r.t. such symmetry transformations as translations in time and resizing
- Examples of applications:
    - Technical analysis for financial markets







Convolutional Networks for Time Series Analysis

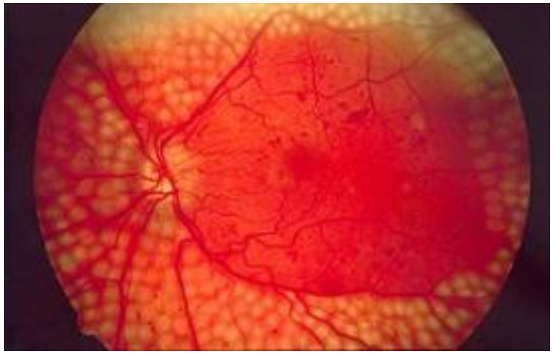
- Analysis of medical data sequences

- Analysis of music

Discussion on Assignment # 2

Discussion on Assignment # 2

- Objective:
  - Build a ConvNet capable of distinguishing diabetic retinopathy stages based on color fundus photos
  - Your evaluation metric is Quadratic Weighted Kappa



- How to approach the task:
  - Analyze the baseline solution. Any questions about it?
  - Check solutions suggested by Kaggle Diabetic Retinopathy Detection Competition participants
  - Simple ideas:
    - Tune architecture and hyperparameters

Discussion on Assignment # 2

- Try to augment training data by modified images (rotations, crops, resizing, …)
- Try learning rate schedule
- Take into account correlation between left and right eyes
- Try different activation functions, e.g. maxout
- Use ensembles of different models
- Try unsupervised pre-training
- Test fractional-pooling https://arxiv.org/pdf/1412.6071.pdf

References

- D. Erhan, Y. Bengio, A. Courville, P. Manzagol, P. Vincent. *Why does Unsupervised Pre-training Help Deep Learning?* Journal of Machine Learning Research 11 (2010) 625-660
- A. Dosovitskiy, J. T. Springenberg, M. Tatarchenko, T. Brox. *Learning to Generate Chairs, Tables and Cars with Convolutional Networks*. ArXiv 1411.5928 (2015)
- M. Zeiler, R. Fergus. *Visualizing and Understanding Convolutional Networks* ArXiv 1311.2901 (2013)
- M. Zeiler, D. Krishnan, G. Taylor and R. Fergus. *Deconvolutional Networks*. Conference on Computer Vision and Pattern Recognition (2010)

General Information

- Next Lecture:
  - Title: ‘Recurrent neural networks: Part 1’
  - Main topics:
    - Motivation for recurrent networks (RNN)
    - Biological basis for RNN
    - Simple RNNs
    - Gated RNNs
    - RNN applications
  - Schedule: 10 May, Wednesday 18:30
- Assignments:
  - Assignment #3 due is 24 May 2017
  - Submissions for assignments #2 and #3 should be made through Kaggle