# A Shallow Introduction into the Deep Machine Learning

Jan Čech

# What is the "Deep Learning" ?

- Deep learning
  = both the classifiers and the features are learned automatically

image $\xrightarrow{\text{classifier}}$ label

- Typically not feasible, due to high dimensionality

image $\xrightarrow{\text{hand-engineering}}$ features $\xrightarrow{\text{classifier}}$ label

(e.g. SIFT, SURF, HOG, or MFCC in audio)

- Suboptimal, requires expert knowledge, works in specific domain only

image $\xrightarrow{\text{learning}}$ features $\xrightarrow{\text{classifier}}$ label

(feature hierarchies)

Deep neural network

# Deep learning successes



- Deep learning methods have been extremely successful recently
  - Consistently beating state-of-the-art results in many fields, winning many challenges by a significant margin

Computer vision:

- Hand writing recognition, Action/activity recognition, Face recognition
- Large-scale image category recognition (ILSVRC' 2012 challenge)

| | |
|---|---|
| INRIA/Xerox | 33%, |
| Uni Amsterdam | 30%, |
| Uni Oxford | 27%, |
| Uni Tokyo | 26%, |
| **Uni Toronto** | **16% (deep neural network)** [Krizhevsky-NIPS-2012] |

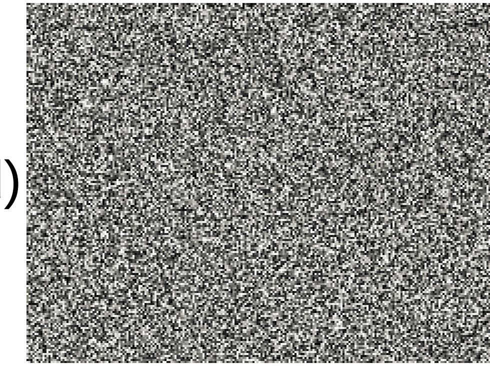
Automatic speech recognition:

- TIMIT Phoneme recognition, speaker recognition

Natural Language Processing, Text Analysis:

- IBM Watson

# Learning the representation – Sparse coding



- Natural image statistics
  - Luckily, there is a redundancy in natural images
  - Pixel intensities are not i.i.d. (but highly correlated)

- Sparse coding [Olshausen-1996, Ng-NIPS-2006]

Input images: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}; (x^{(i)} \in R^{n\times n})$

Learn dictionary of basis functions $\phi_1, \phi_2, \ldots, \phi_k; (\phi_j \in R^{n\times n})$ that

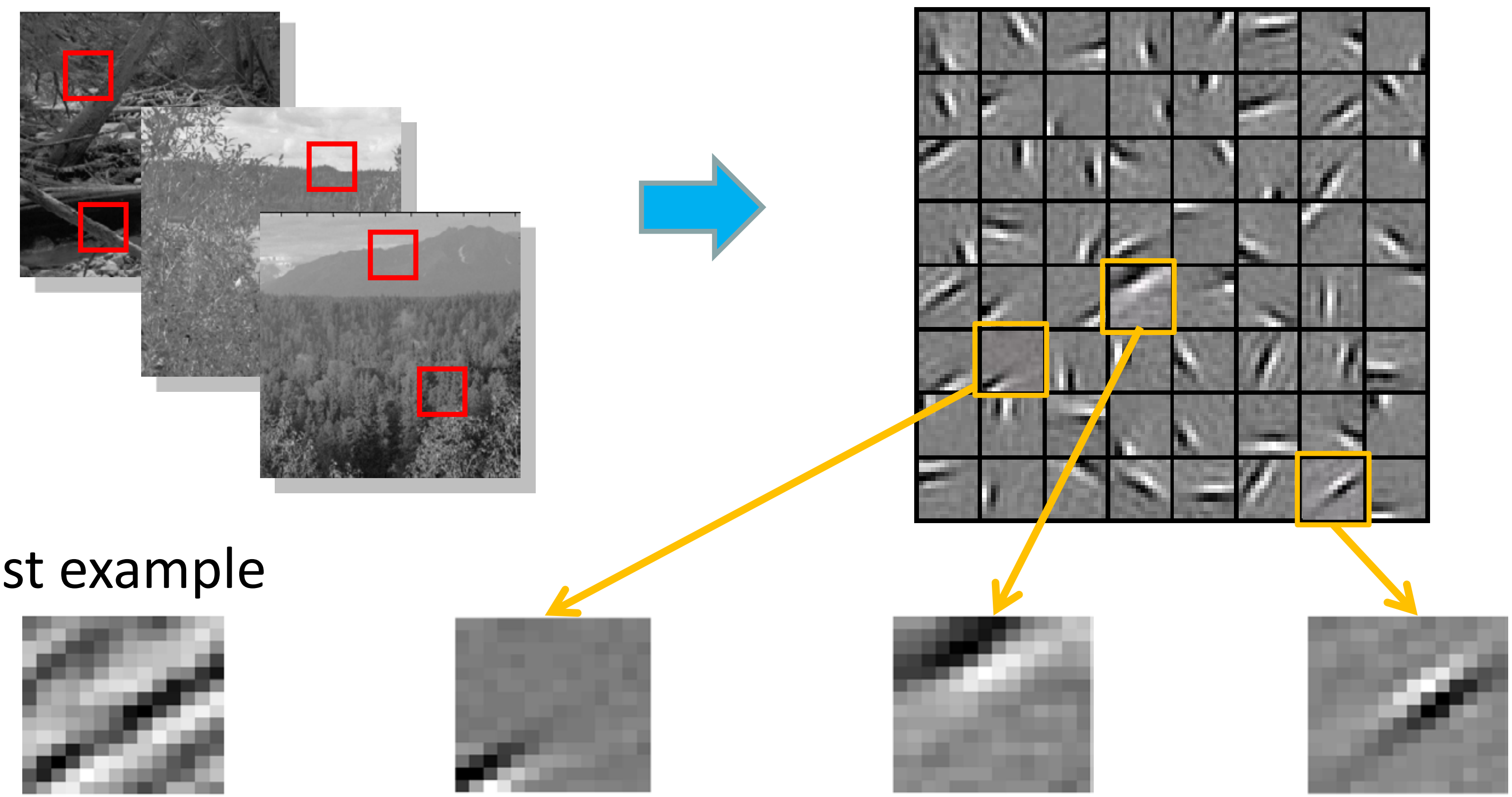
$$x \approx \sum_{j=1}^{k} a_j \phi_j \; ; \quad \text{s.t. } a_j \text{ are mostly zero, “sparse”}$$

$$\min_{a,\phi} \sum_{i=1}^{m} \left( \left\| x^{(i)} - \sum_{j=1}^{k} a_j^{(i)} \phi_j \right\|^2 + \lambda \sum_{j=1}^{k} |a_j^{(i)}| \right)$$

# Sparse coding

Natural Images

Learned bases ($\phi_1, \ldots, \phi_{64}$): "Edges"




$$x \approx 0.8 * \phi_{36} + 0.3 * \phi_{42} + 0.5 * \phi_{63}$$

[0, 0, …, 0, **0.8**, 0, …, 0, **0.3**, 0, …, 0, **0.5**, …]

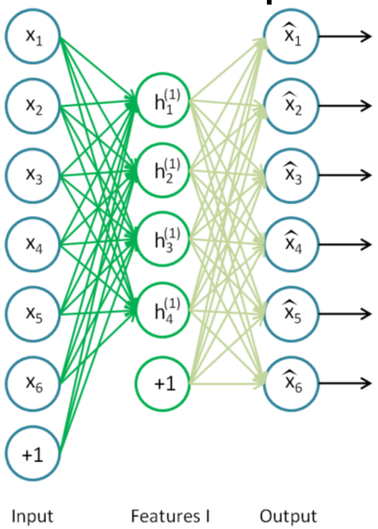
= [$a_1$, …, $a_{64}$] (feature representation)

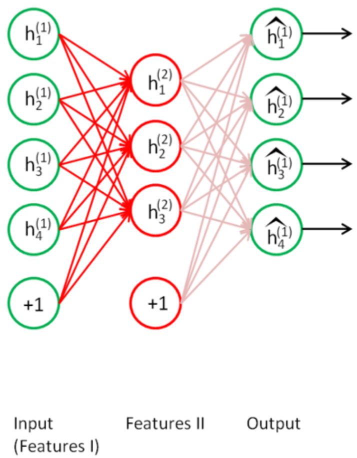
Compact & easily interpretable

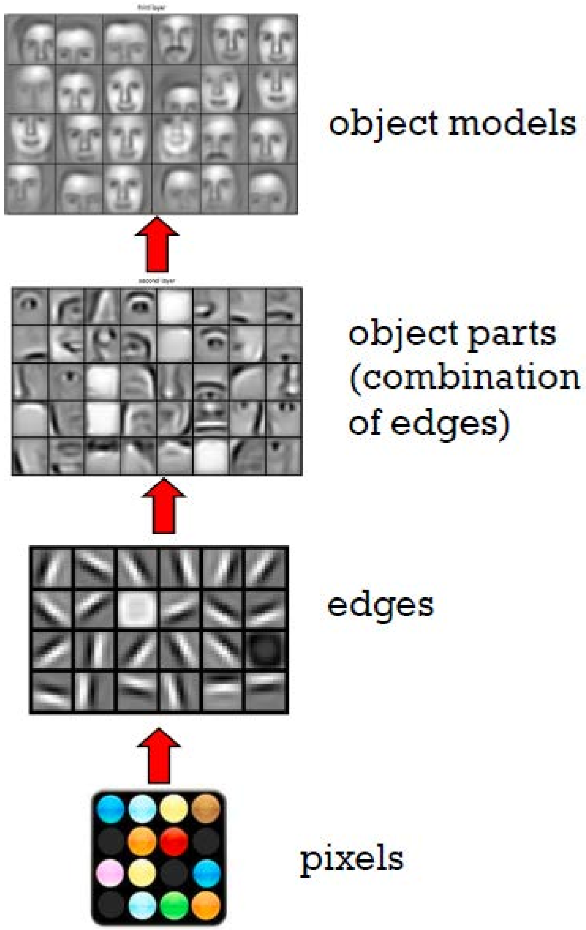
# Unsupervised Learning Hierarchies of features



- Many approaches to unsupervised learning of feature hierarchies
  - Sparse Auto-encoders [Bengio-2007]
  - Restricted Boltzmann Machines [Hinton-2006]
- These model can be stacked: lower hidden layer is used as the input for subsequent layers



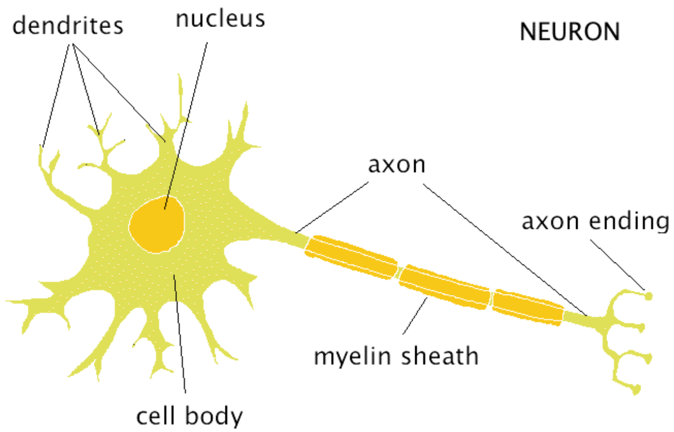

- **The hidden layers are trained to capture higher-order data correlations.**
- Learning the hierarchies and classification can be implemented by a (Deep) Neural Network

[Lee-ICML-2009]

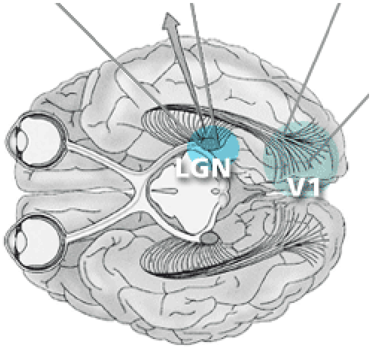
# Resemblance to sensory processing in the brain



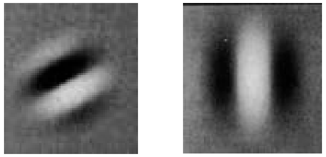
- Needless to say that the brain is a neural network




~ 2e-11 neurons
~ 1e-14 synapses




- Primary visual cortex V1
  - Neurophysiological evidences that primary visual cells are sensitive to the orientation and frequency (Gabor filter like impulse responses)
  - [Hubel-Wiesel-1959] (Nobel Price winners)
    - Experiments on cats with electrodes in the brain

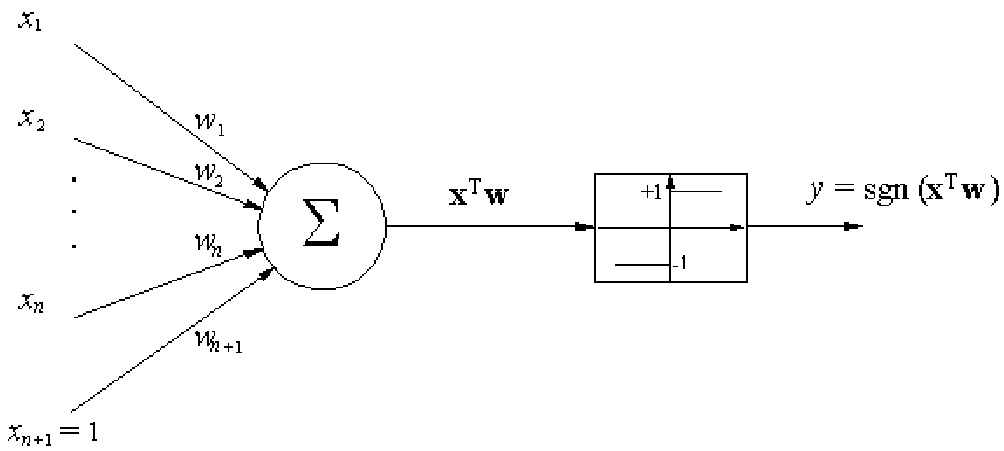


- A single learning algorithm hypothesis ?
  - “Rewiring” the brain experiment [Sharma-Nature-2000]
    - Connecting optical nerve into A1 cortex (a subject was able to solve visual tasks by using the processing in A1)

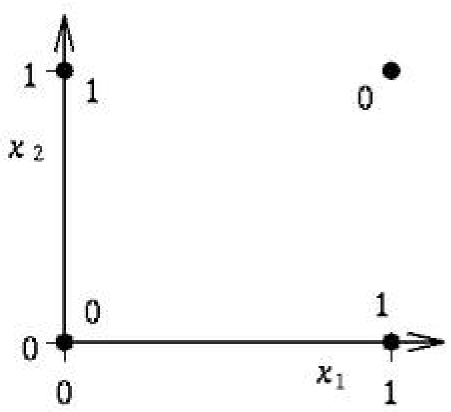
# (Artificial) Neural Networks



- Neural networks are here for more than 50 years
  - Rosenblatt-1956 (perceptron)



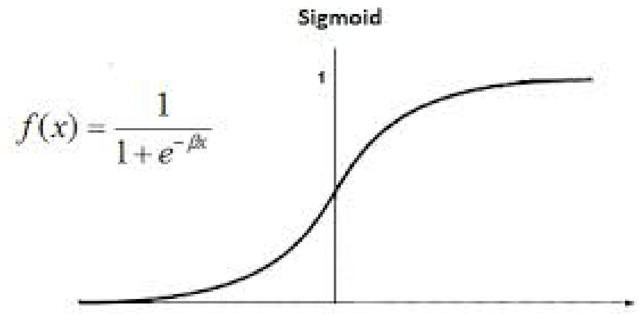

  - Minsky-1969 (xor issue, => skepticism)



# Neural Networks

Rumelhart and McClelland – 1986:

- Multi-layer perceptron,
- Back-propagation (supervised training)
  - Differentiable activation function
  - Stochastic gradient descent

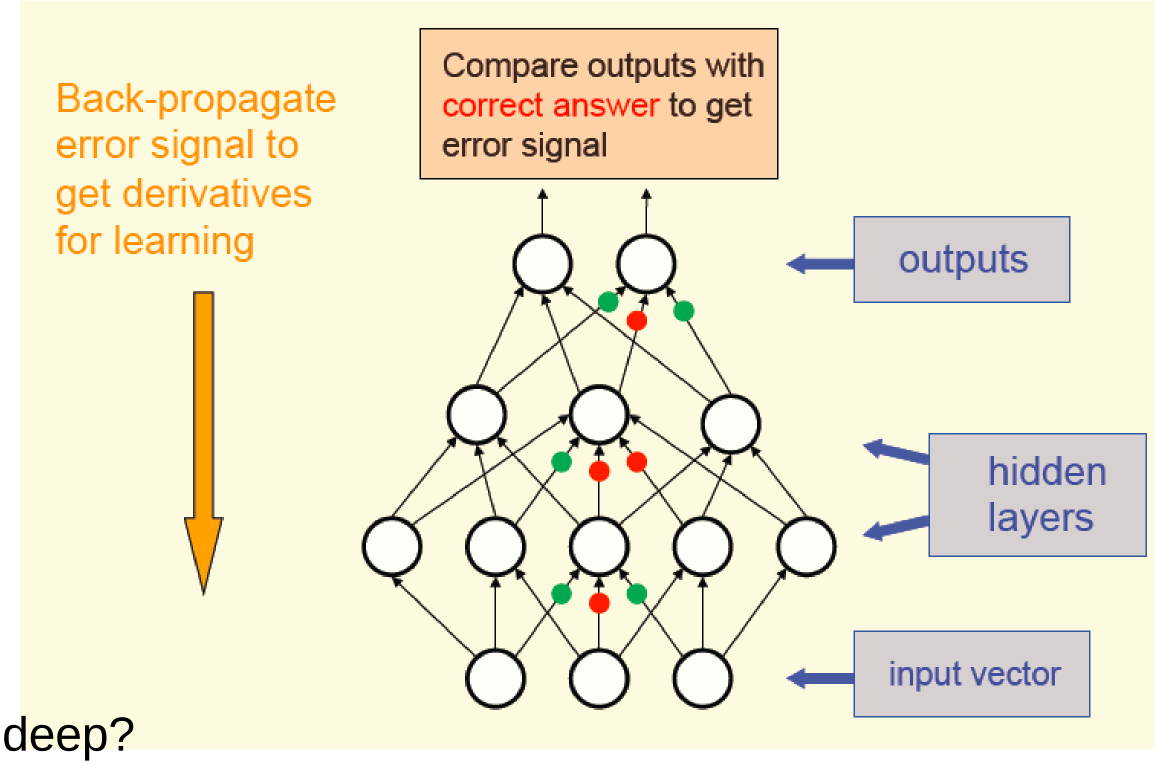



Empirical risk

$$Q(w) = \sum_{i=1}^{n} Q_i(w),$$

Update weights:

$$w := w - \alpha \nabla Q_i(w).$$




What happens if a network is deep?
(it has many layers)

# What was wrong with back propagation?

- Local optimization only (needs a good initialization, or re-initialization)
- Prone to over-fitting
  - too many parameters to estimate
  - too few labeled examples
- Computationally intensive

=> Skepticism: A deep network often performed worse than a shallow one

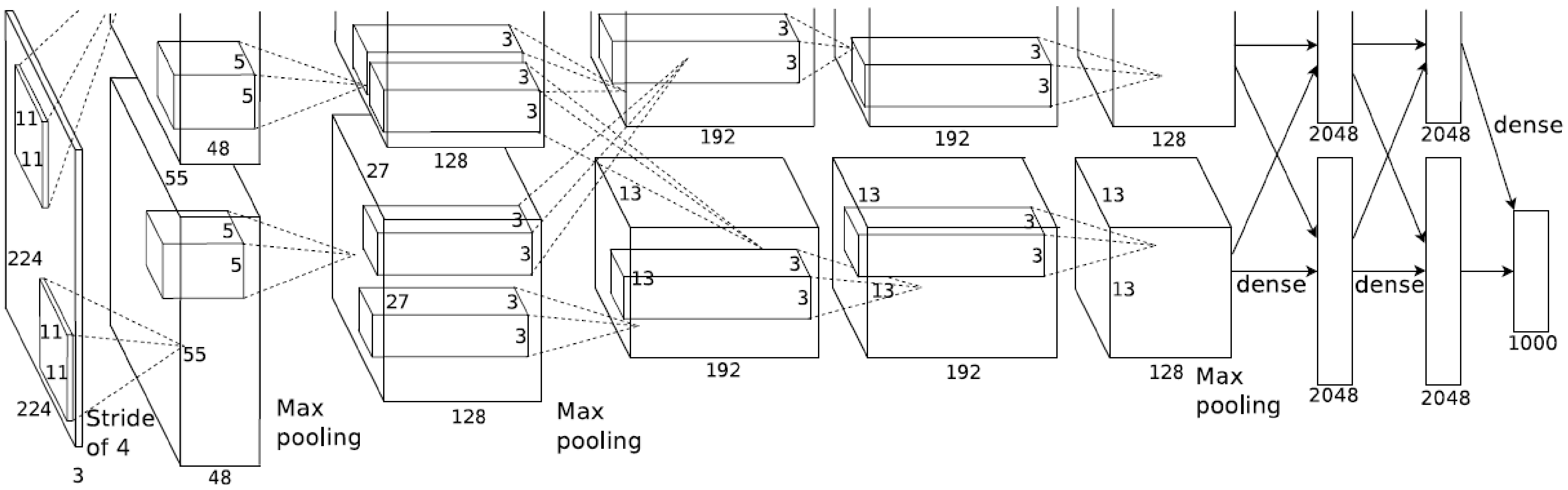
- However nowadays:
  - Weights can be initialized better (Use of unlabeled data, Restricted Boltzmann Machines)
  - Large collections of labeled data available
    - ImageNet (14M images, 21k classes, hand-labeled)
  - Reducing the number of parameters by weight sharing
    - Convolutional layers – [LeCun-1989]
  - Fast enough computers (parallel hardware, GPU)

=> Optimism: It works!

# Deep convolutional neural networks

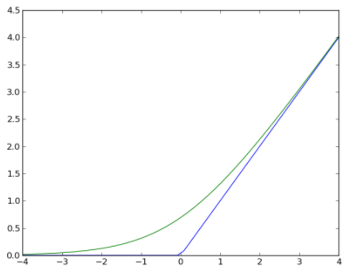


- An example for Large Scale Classification Problem:
  - Krizhevsky, Sutskever, Hinton: ImageNet classification with deep convolutional neural networks. NIPS, 2012.
    - Recognizes 1000 categories from ImageNet
    - Outperforms state-of-the-art by significant margin (ILSVRC 2012)




- 5 convolutional layers, 3 fully connected layers
- 60M parameters, trained on 1.2M images (~1000 examples for each category)

# Deep convolutional neural networks



- Additional tricks: "Devil is in the details"
  - Rectified linear units instead of standard sigmoid

    $$f(x) = \max(0, x)$$

  - Convolutional layers followed by max-pooling
    - Local maxima selection in overlapping windows (subsampling)

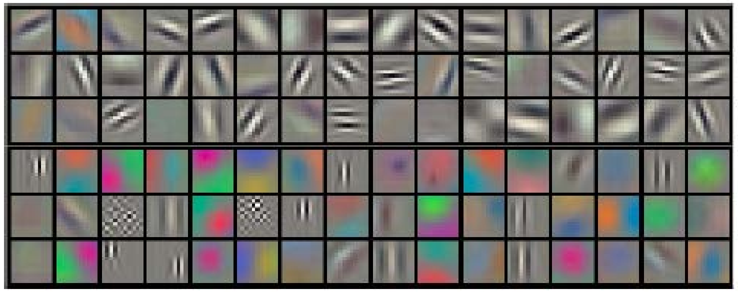
    => dimensionality reduction, shift insensitivity
  - Dropout
    - Averaging results of many independent models (similar idea as in Random forests)
    - 50% of hidden units are randomly omitted during the training, but weights are shared in testing time

    => Probably very significant to reduce overfitting
  - Data augmentation
    - Images are artificially shifted and mirrored (10 times more images)

    => transformation invariance, reduce overfitting

# Deep convolutional neural networks

- No unsupervised pre-initialization!
  - The training is supervised by standard back-propagation
  - enough labeled data: 1.2M labeled training images for 1k categories
  - Learned filters in the first layer
    - Resemble cells in primary visual cortex

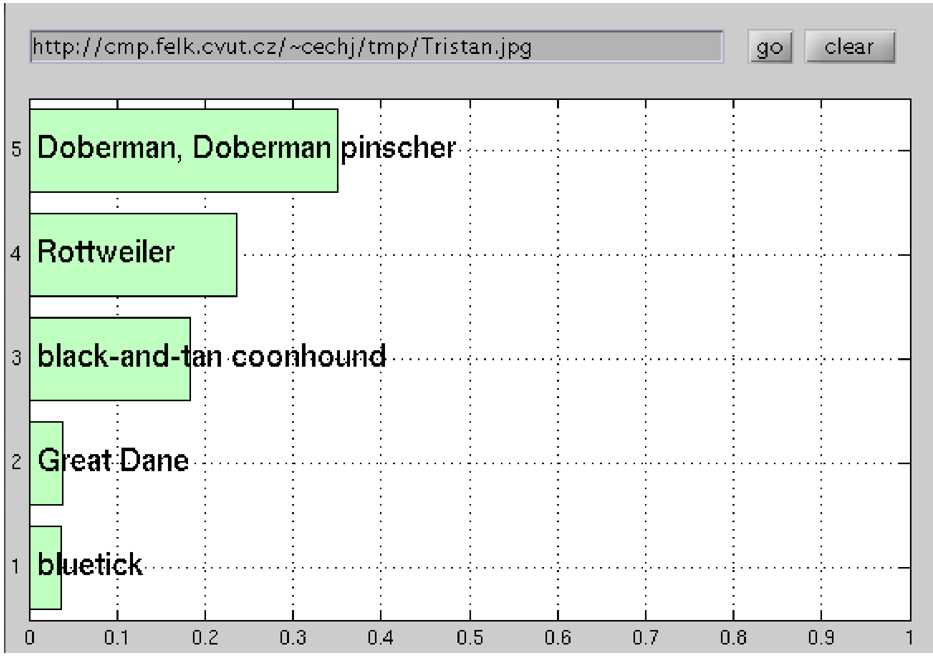


- Training time:
  - 5 days on NVIDIA GTX 580, 3GB memory
  - 90 cycles through the training set
- Test time (forward step) on GPU
  - Implementation by Yangqing Jia, http://caffe.berkeleyvision.org/
  - 5 ms/image in a batch mode
  - (my experience: 100 ms/image in Matlab, including image decompression and normalization)

# Preliminary experiments 1: Category recognition

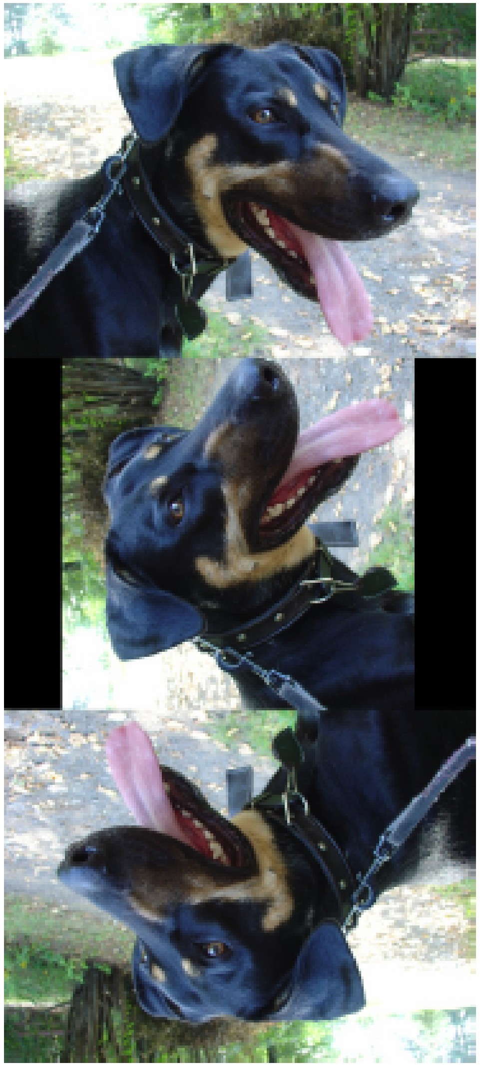


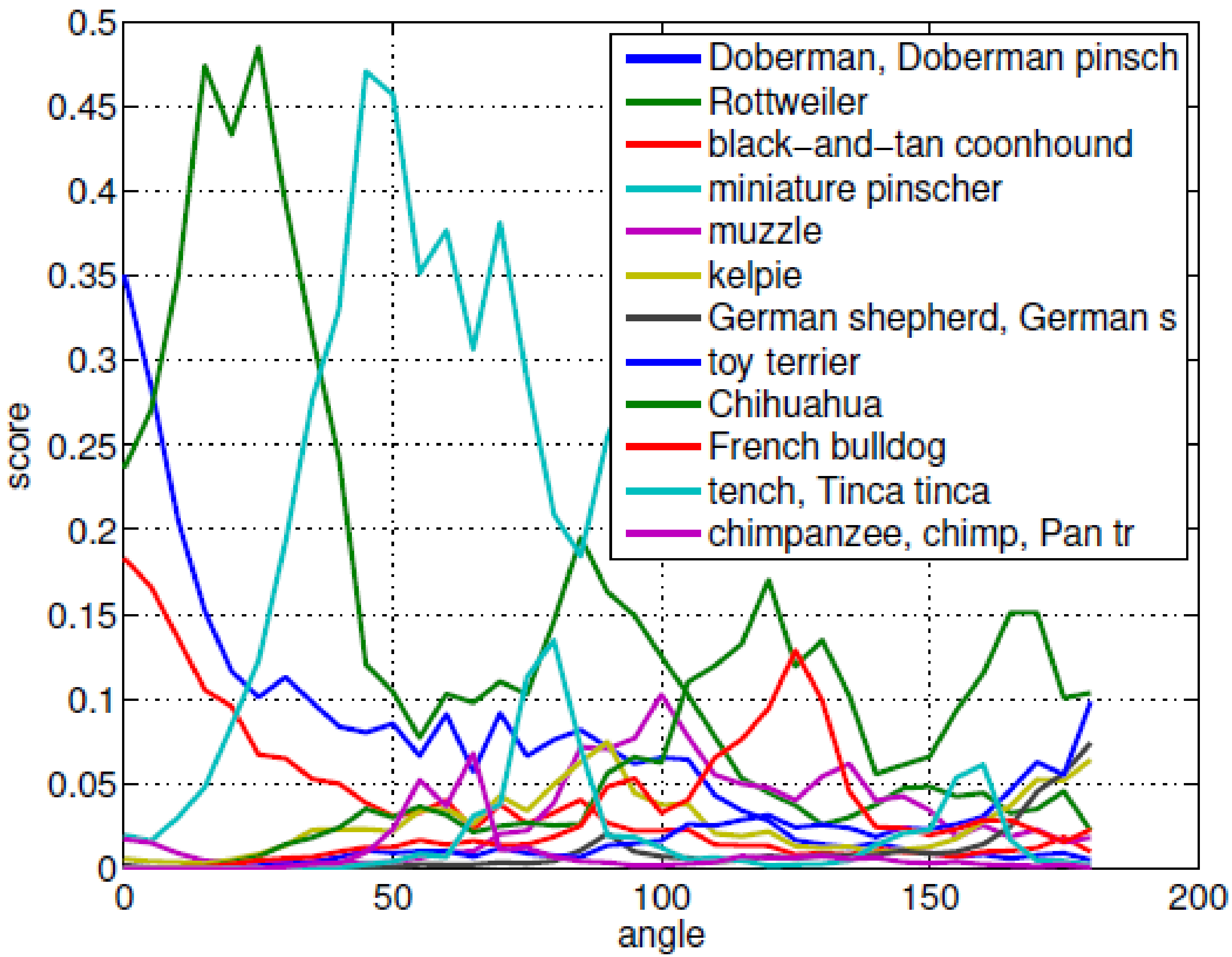
- Implementation by Yangqing Jia, http://caffe.berkeleyvision.org/
  - network pre-trained for 1000 categories provided
- Which categories are pre-trained?
  - 1000 "most popular" (probably mostly populated)
  - Typically very fine categories (dog breeds, plants, vehicles…)
  - Category "person" (or derived) is missing
  - Recognition subjectively surprisingly good…




# Sensitivity to image rotation

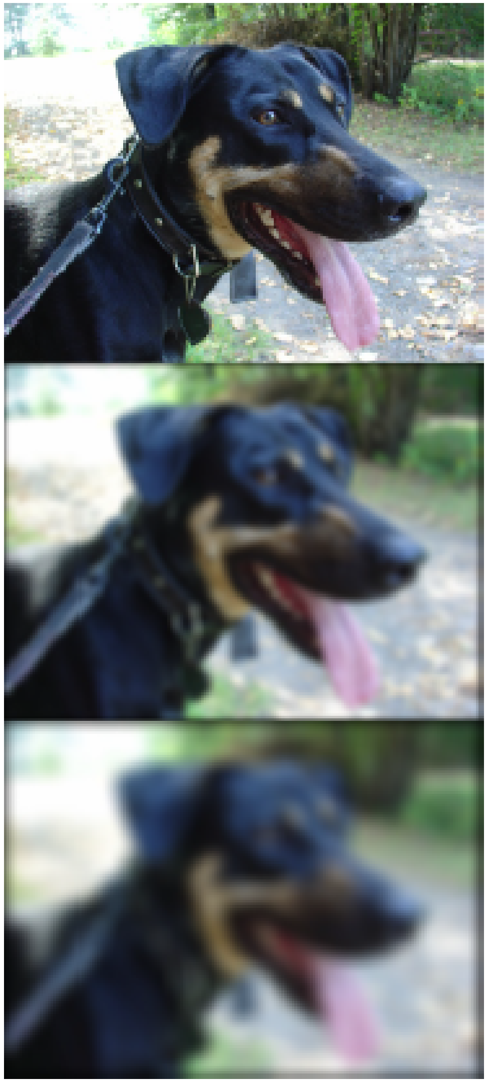


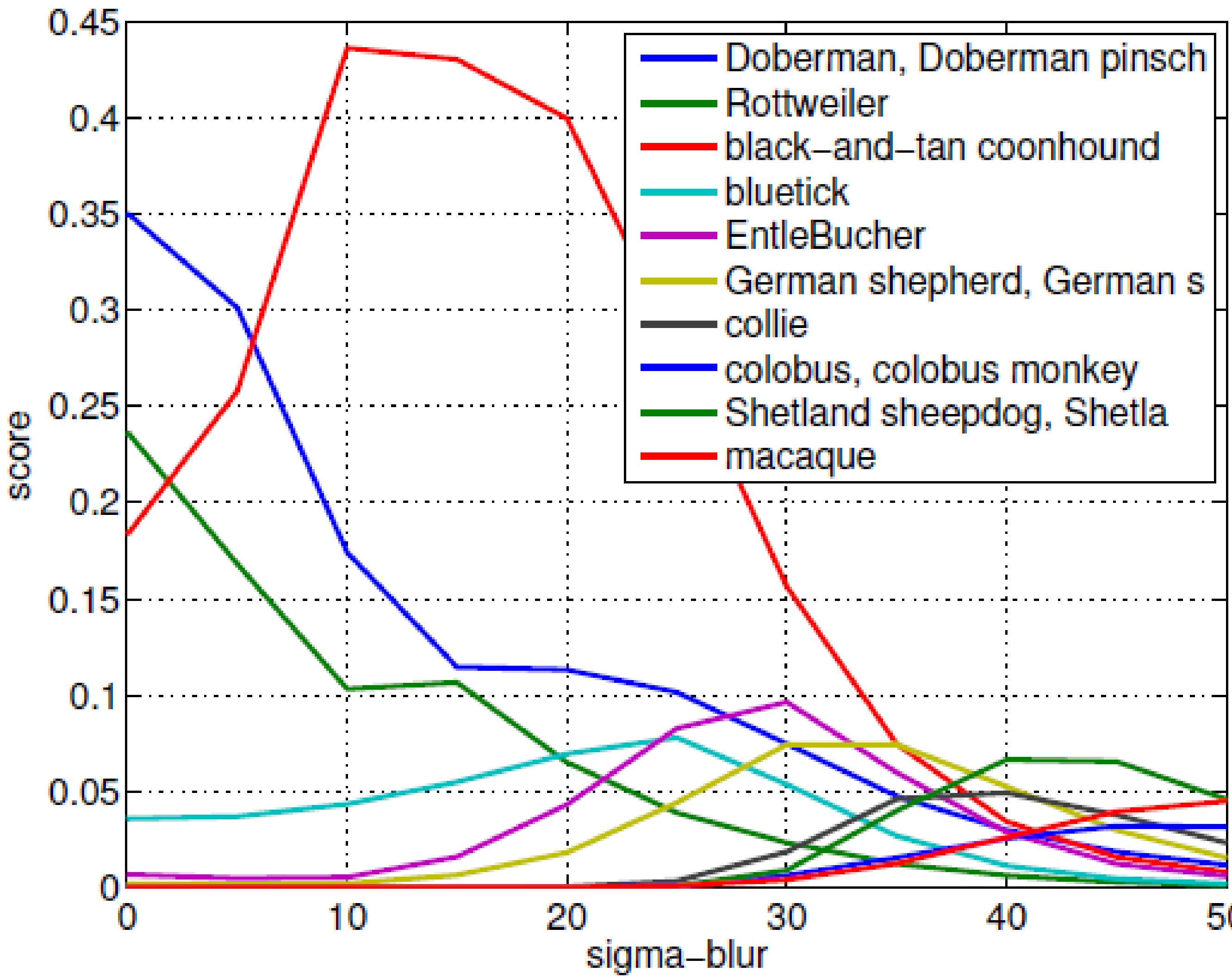




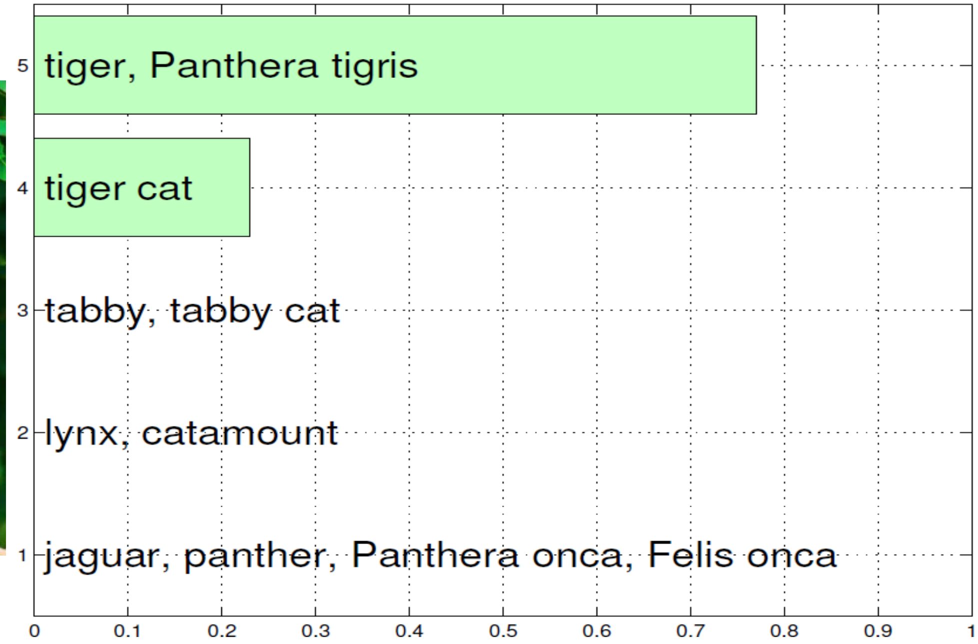
# Sensitivity to image blur



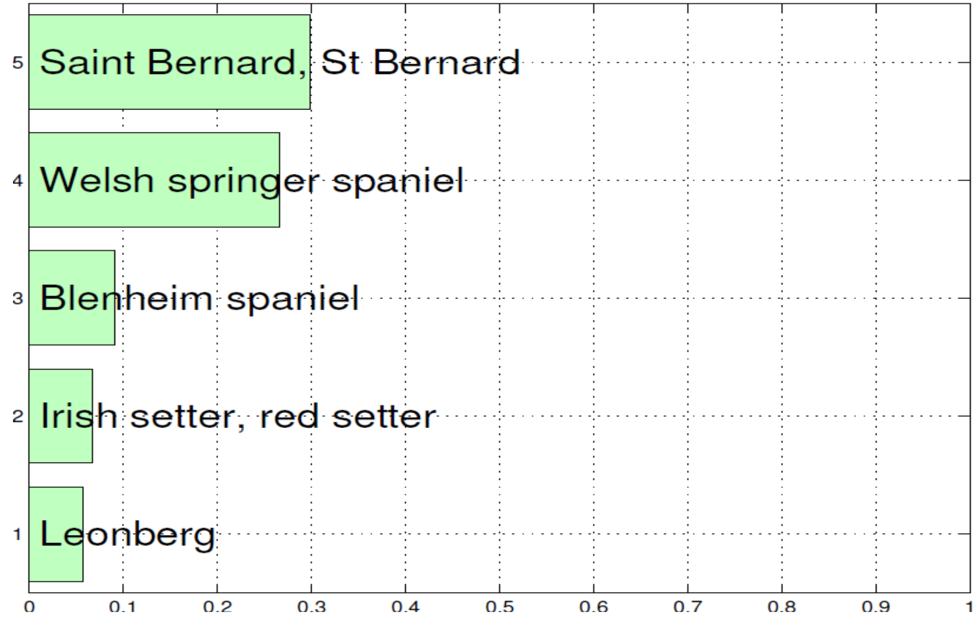




# It is not a texture only...

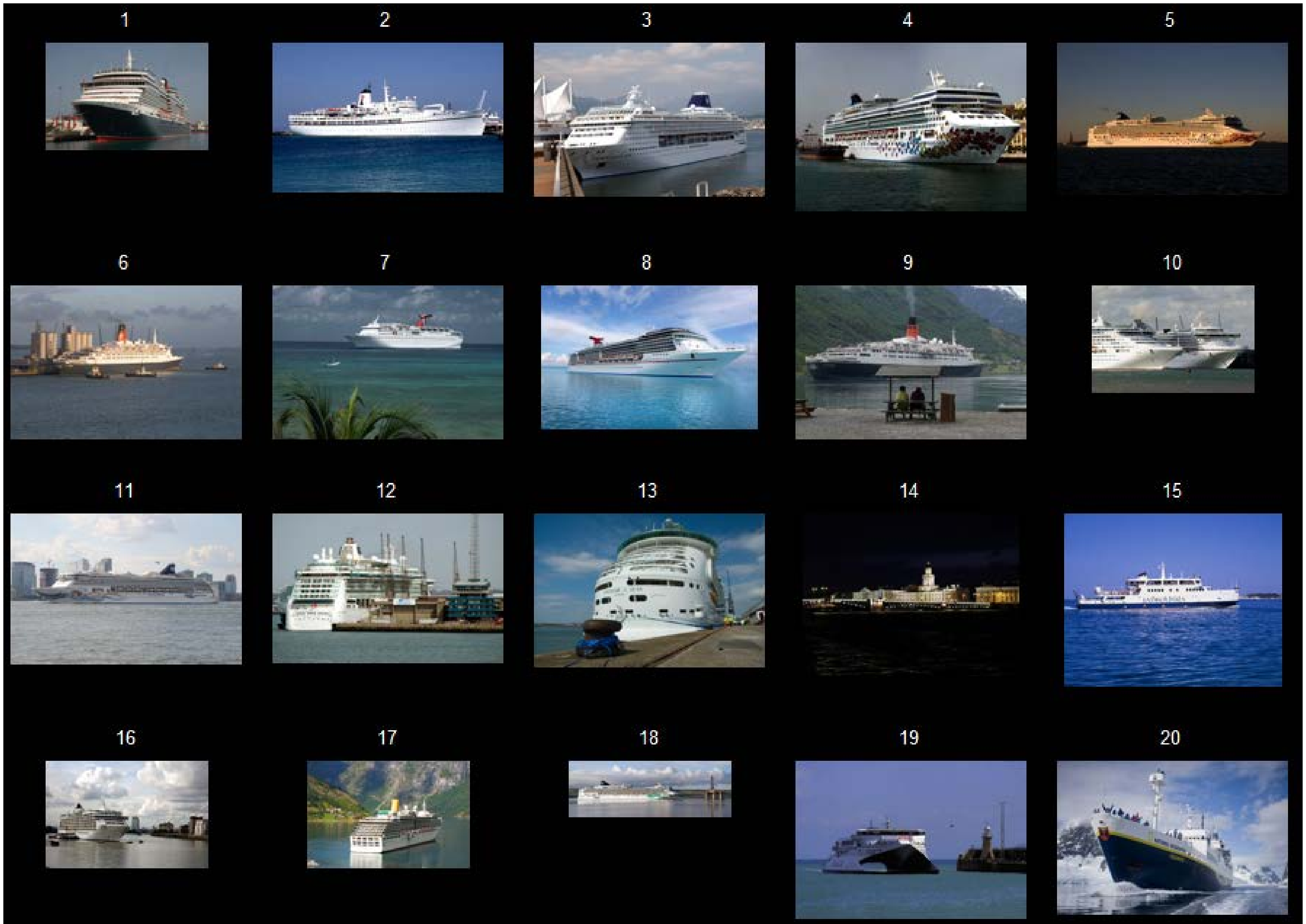






# Preliminary experiments 2: Category retrieval

- 50k randomly selected images from Profimedia dataset
- Category: Ocean liner



# Preliminary experiments 2: Category retrieval

- Category: Restaurant (results out of 50k-random-Profiset)

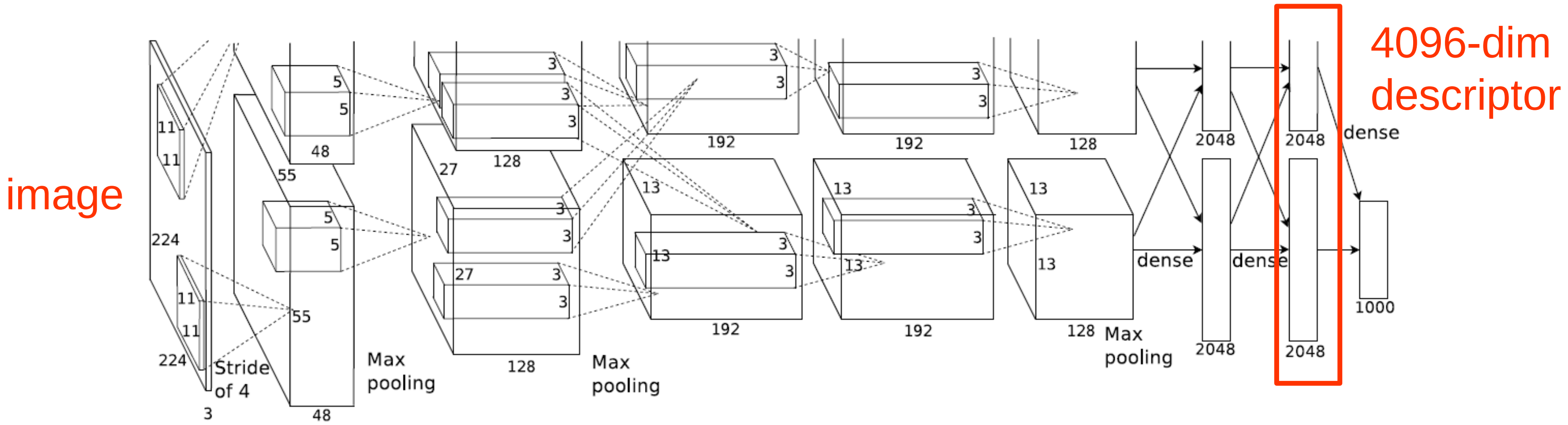
# Preliminary experiments 2: Category retrieval

- Category: stethoscope (results out of 50k-random-Profiset)

# Preliminary experiments 3: Similarity search



- Indications in the literature that the last hidden layer carry semantics
  - Last hidden layer (4096-dim vector), final layer category responses (1000-dim vector)
  - New (unseen) categories can be learned by training (a linear) classifier on top of the last hidden layer
    - Oquab, Bottou, Laptev, Sivic, TR-INRIA, 2013
    - Girshick, Dphanue, Darell, Malik, CVPR, 2014
  - **Responses of the last hidden layer can be used as a compact global image descriptor**
    - Semantically similar images should have small Euclidean distance



# Preliminary experiments 3: Similarity search



- Qualitative comparison: (20 most similar images to a query image)
  1. MUFIN annotation (web demo), http://mufin.fi.muni.cz/annotation/, [Zezula et al., *Similarity Search: The Metric Space Approach.* 2005.]
     - Nearest neighbour search in **20M** images of Profimedia
     - Standard global image statistics (e.g. color histograms, gradient histograms, etc.)
  2. Caffe  NN (last hidden layer response + Euclidean distance),
     - Nearest neighbour search in **50k** images of Profimedia

MUFIN results

# Preliminary experiments 3: Similarity search



Caffe NN results

# Preliminary experiments 3: Similarity search



MUFIN results

# Preliminary experiments 3: Similarity search

Caffe NN results

# Preliminary experiments 3: Similarity search

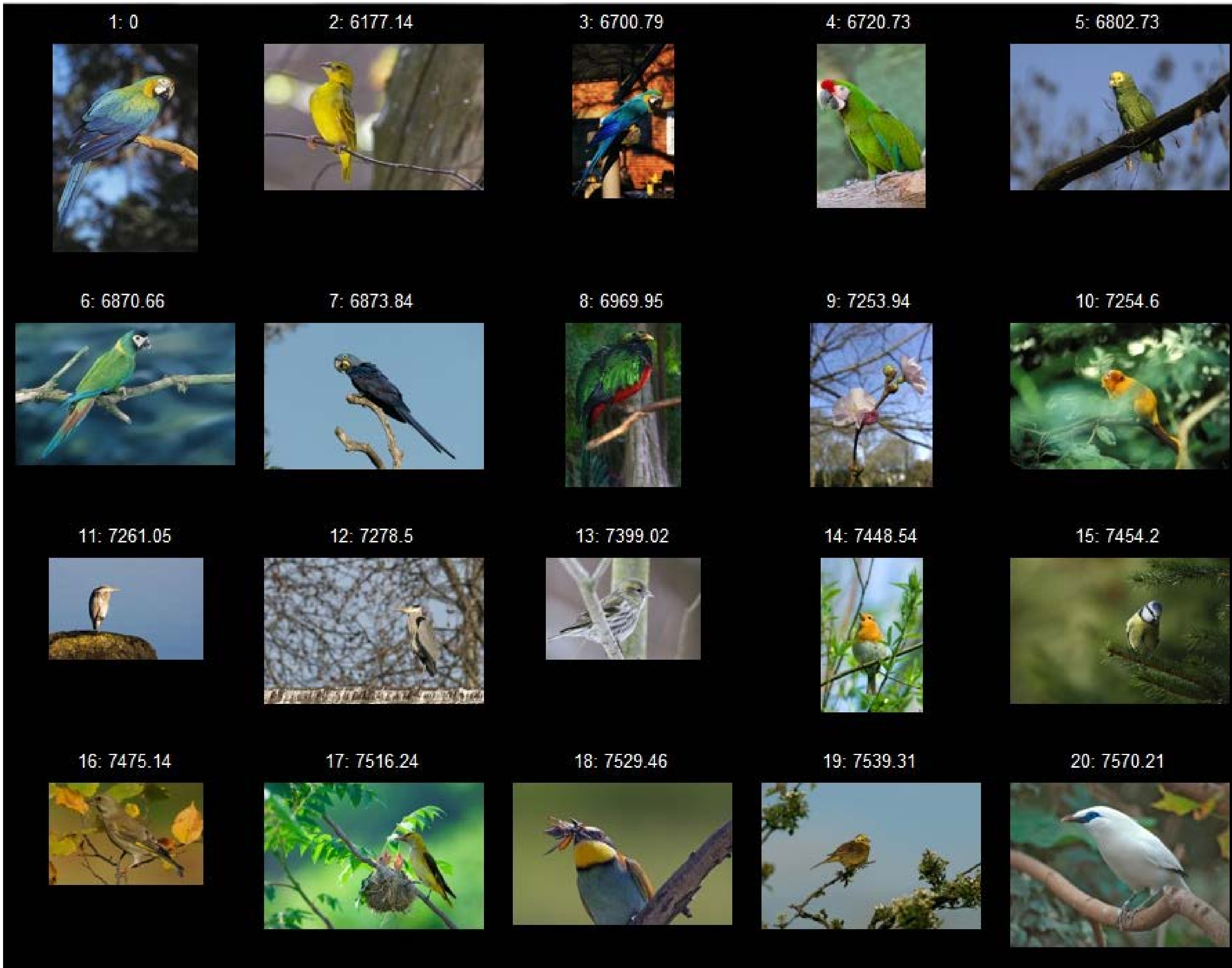


MUFIN results

# Preliminary experiments 3: Similarity search



Caffe NN results



# Preliminary experiments 3: Similarity search



MUFIN results

# Preliminary experiments 3: Similarity search

Caffe NN results

# Preliminary experiments 3: Similarity search



MUFIN results

# Preliminary experiments 3: Similarity search



Caffe NN results

# Preliminary experiments 3: Similarity search



MUFIN results

# Preliminary experiments 3: Similarity search



Caffe NN results

# Preliminary experiments 3: Similarity search

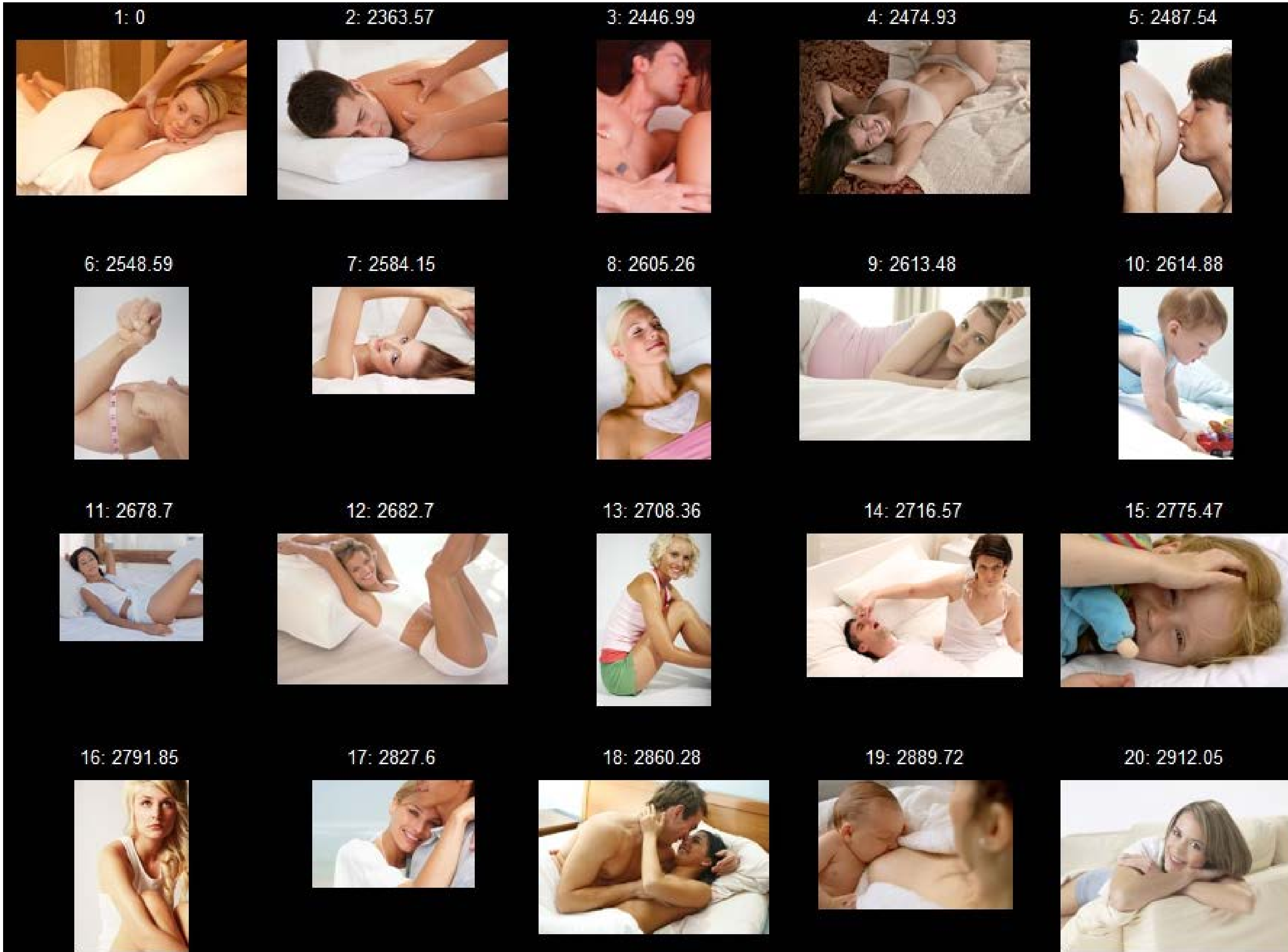


MUFIN results

# Preliminary experiments 3: Similarity search

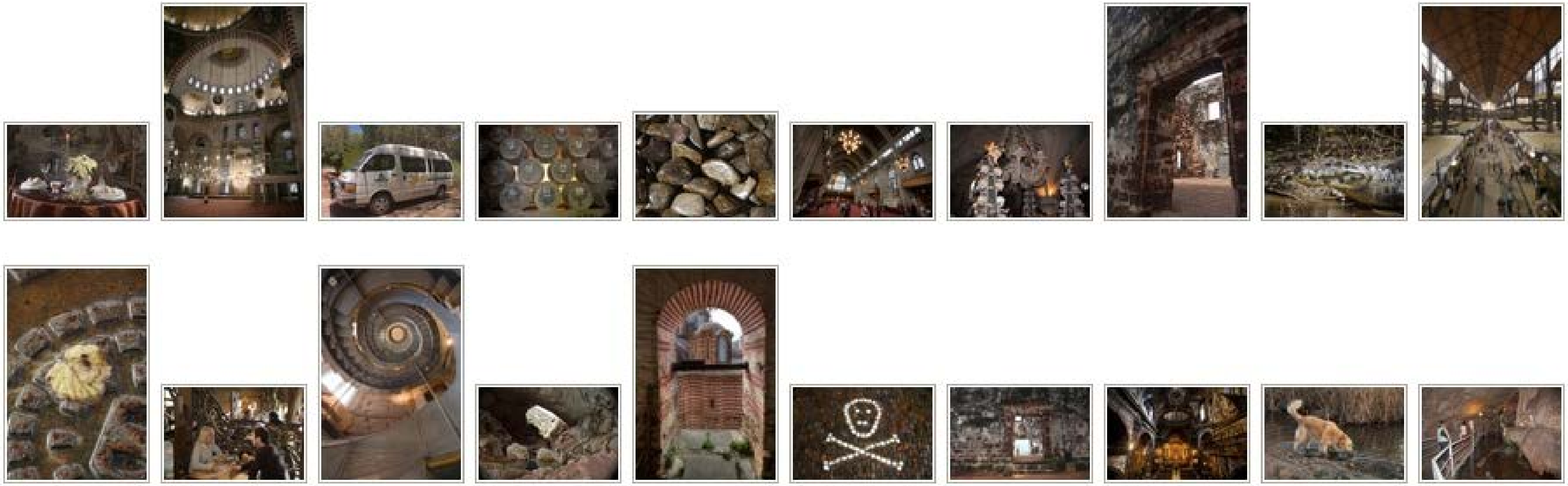


Caffe NN results



# Preliminary experiments 3: Similarity search

MUFIN results

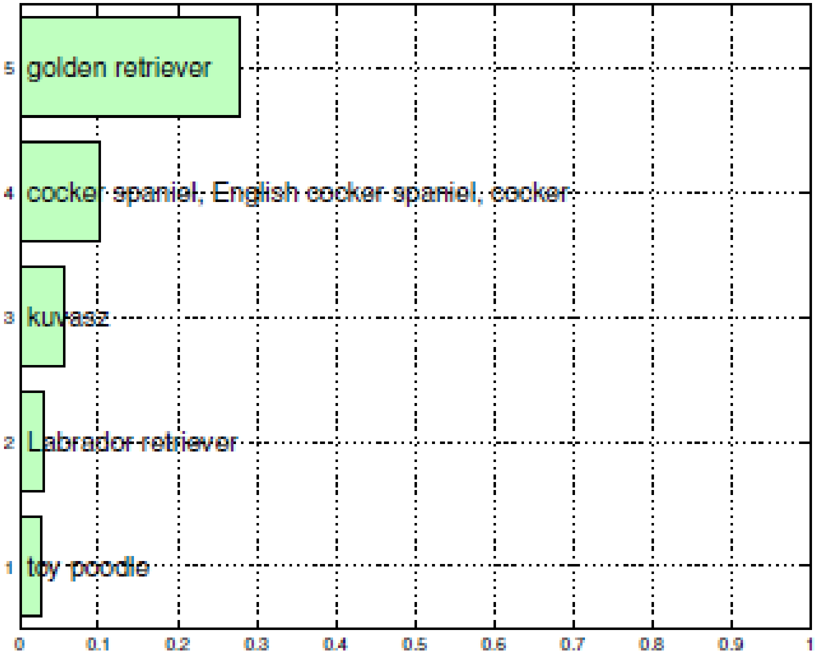


# Preliminary experiments 3: Similarity search

Caffe NN results

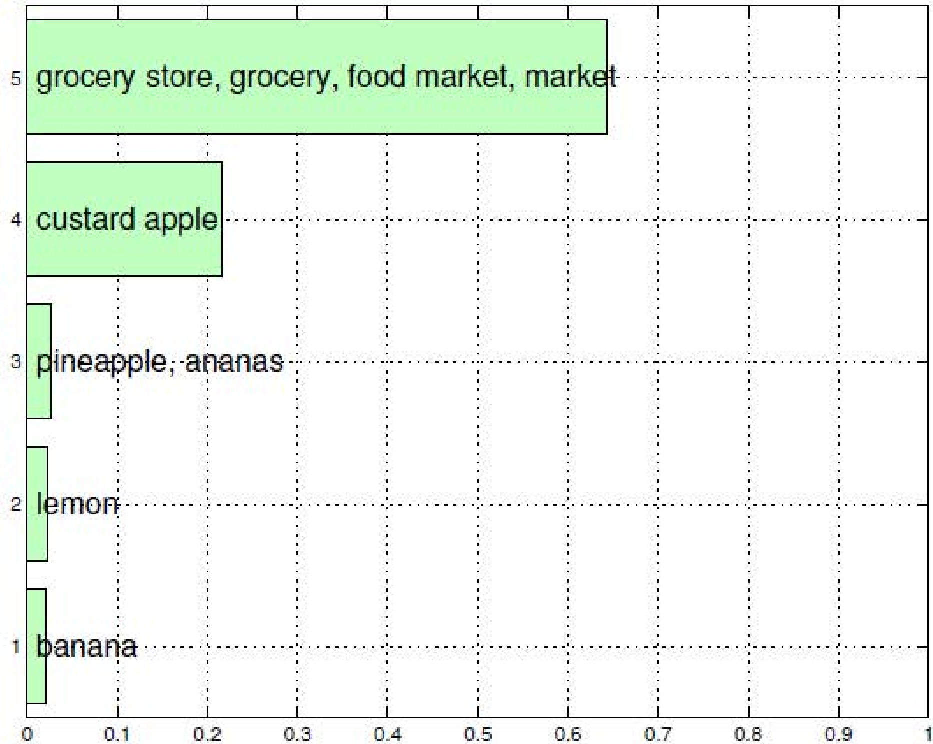
# Multiple object classes





# Multiple object classes





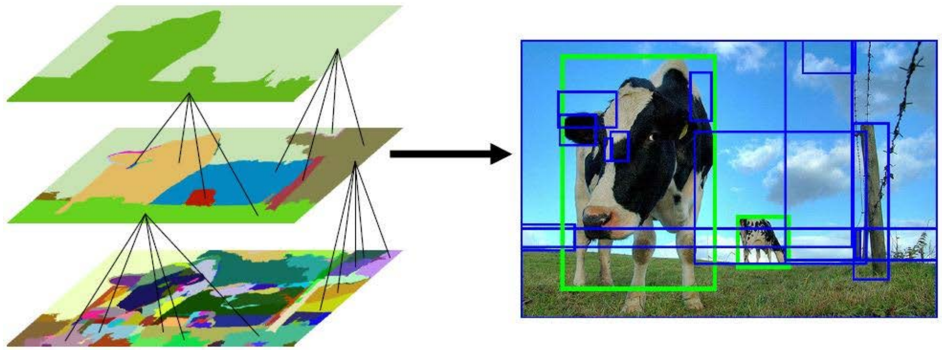
# Object detection: Deep Nets and Sliding Windows



- An image of a scene contains multiple objects
- Exhaustive sliding window detector prohibitively slow

=> Category independent region proposals:

- – Objectness [Alexe-TPAMI-2012]
- – Selective search [Uijlings-IJCV-2013]



- – Edgeboxes [Zitnick-ECCV-2014]

# General recipe to use deep neural networks



- Recipe to use deep neural network to “solve any problem” (by G. Hinton)
  - Have a deep net
  - If you do not have enough labeled data, pre-train it by unlabeled data; otherwise do not bother with pre-initialization
  - Use rectified linear units instead of standard neurons
  - Use dropout to regularize it (you can have many more parameters than training data)
  - If there is a spatial structure in your data, use convolutional layers
  - Have fun… ☺

# Conclusions

- It efficiently learns the abstract representation (shared among classes)
  - The network captures semantics…
- Preliminary experiments with Berkley’s toolbox confirm outstanding performance of the Deep Convolutional Neural Network (recognition, similarity search)
- Low computational demands (100 ms / image) on GPU including loading, image normalization, propagation.
- NNs are (again) in the “Golden Age” (or witnessing a bubble), as many practical problems seem solvable in near future
- Explosion of interest of DNN in literature, graduates get incredible offers, start-ups appear all the time

Human_Abducted_by_UFO.mp4

- Do we understand enough what is going on?

http://www.youtube.com/watch?v=ybgjXfFMah8

Acknowledgement: I borrowed some images from slides of G. Hinton, A. Ng, Y. Le Cun.