

# A Shallow Introduction into the Deep Machine Learning



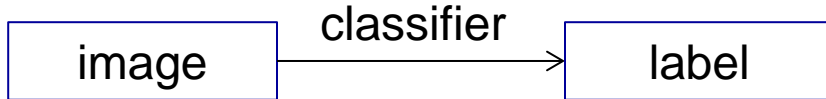
Jan Čech

# What is the “Deep Learning” ?

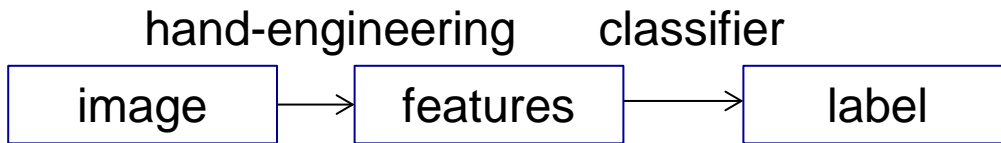


- Deep learning

= both the classifiers and the features are learned automatically

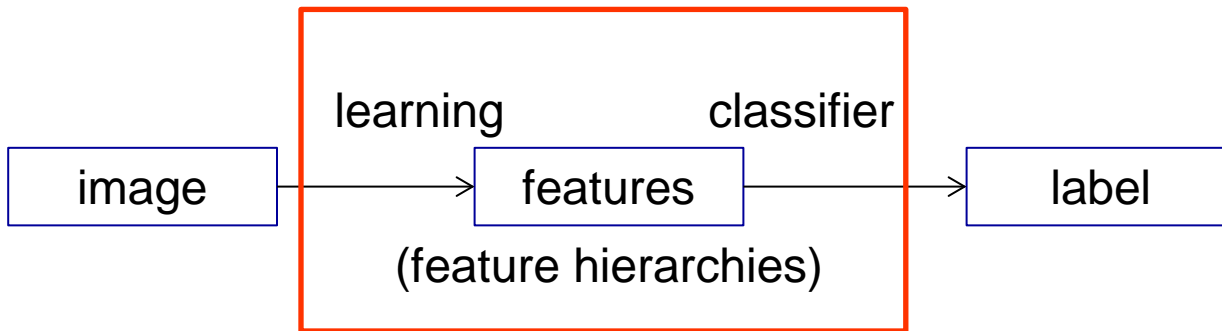


- Typically not feasible, due to high dimensionality



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

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



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%,
<b>Uni Toronto</b>	<b>16% (deep neural network) [Krizhevsky-NIPS-2012]</b>

## Automatic speech recognition:

- TIMIT Phoneme recognition, speaker recognition

## Natural Language Processing, Text Analysis:

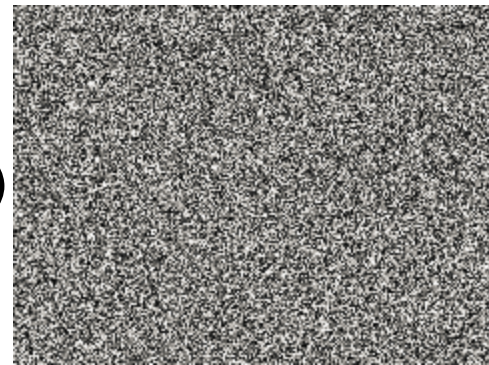
- IBM Watson, Google translate

# Learning the representation – Sparse coding



4

- 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:  $x^{(1)}, x^{(2)}, \dots, x^{(m)}; (x^{(i)} \in R^{n \times n})$

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

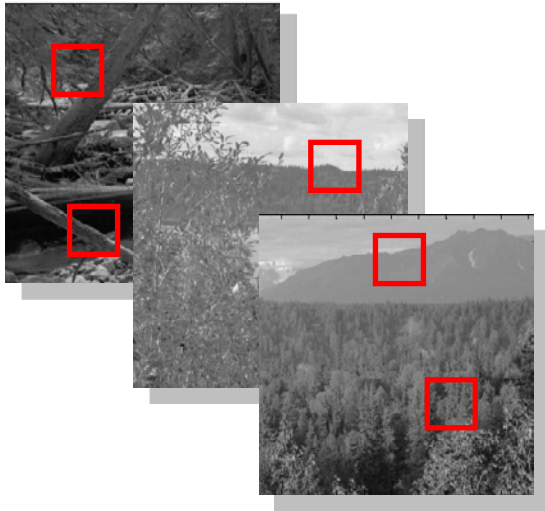
$$x \approx \sum_{j=1}^k a_j \phi_j ; \text{ s.t. } a_j \text{ are mostly zero, "sparse"}$$

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

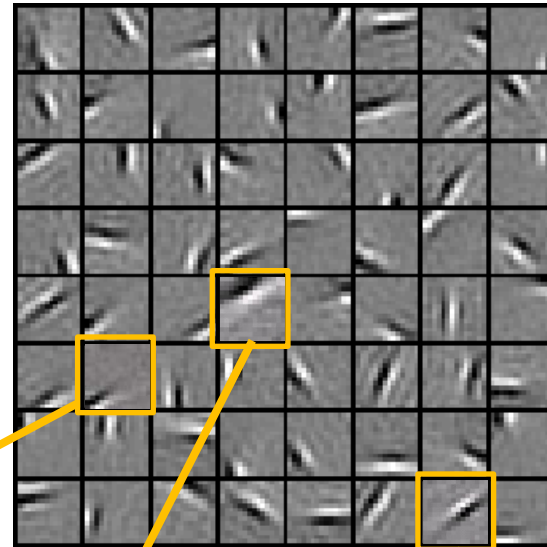
# Sparse coding



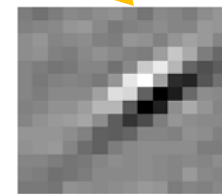
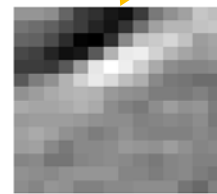
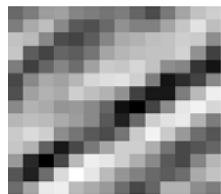
Natural Images



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



Test example



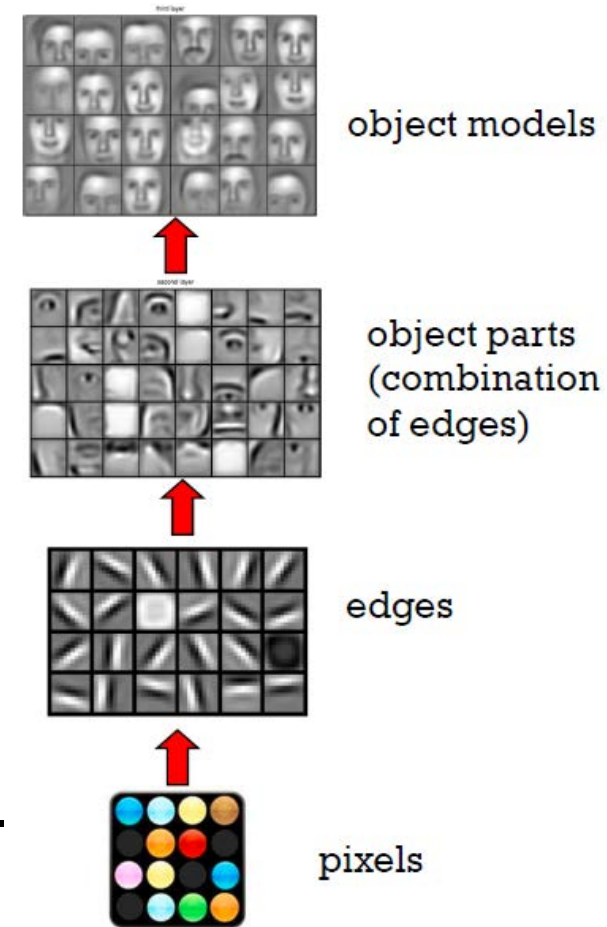
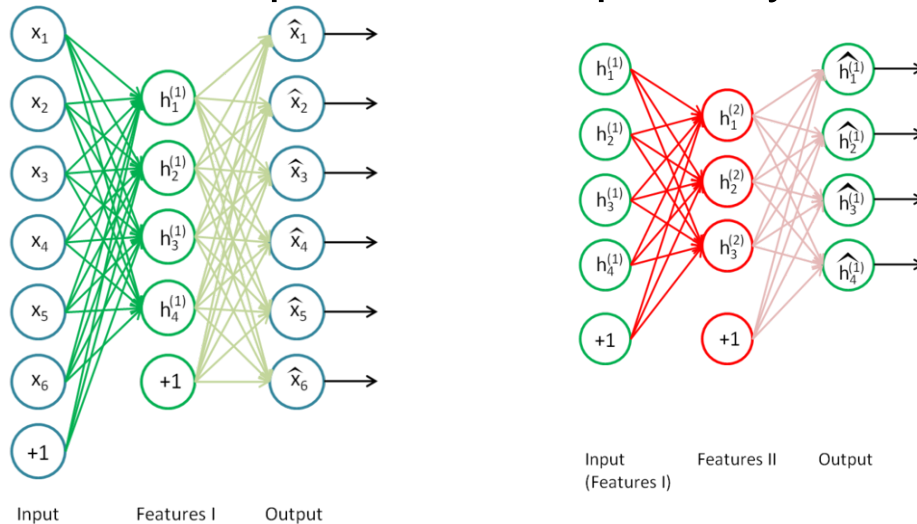
$$x \approx 0.8 * \phi_{36} + 0.3 * \phi_{42} + 0.5 * \phi_{63}$$

$[0, 0, \dots, 0, \mathbf{0.8}, 0, \dots, 0, \mathbf{0.3}, 0, \dots, 0, \mathbf{0.5}, \dots]$   
 $= [a_1, \dots, a_{64}]$  (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



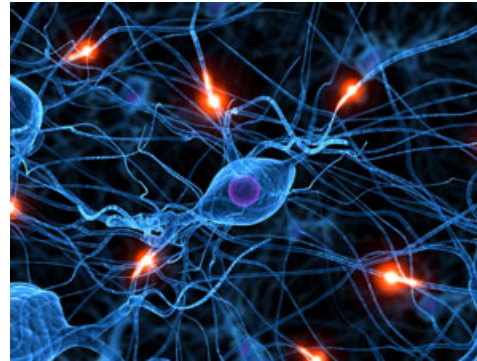
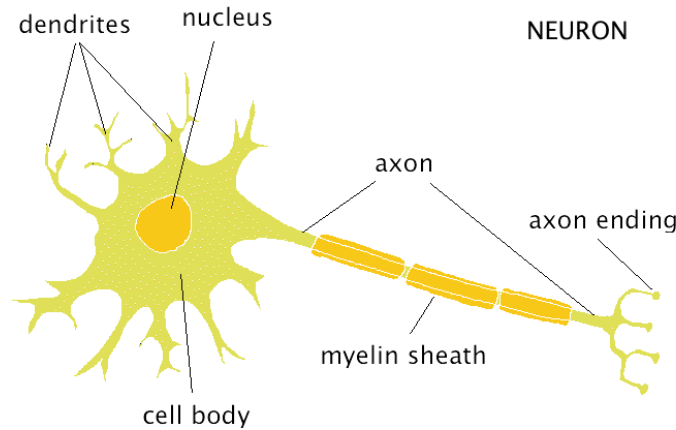
- The hidden layers are trained to capture higher-order data correlations.**
- Learning the hierarchies and classification can be implemented by a (Deep) Neural Network

[Lee-ICML-2009]

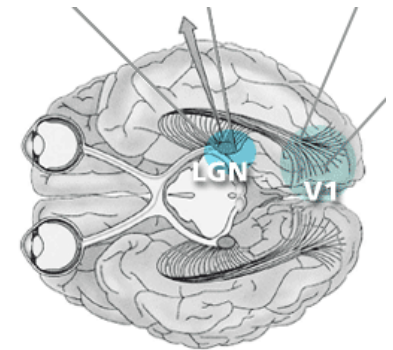
# Resemblance to sensory processing in the brain



- Needless to say that the brain is a neural network

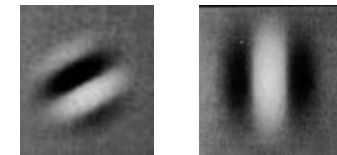


~ 2e-11 neurons  
~ 1e-14 synapses



- 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



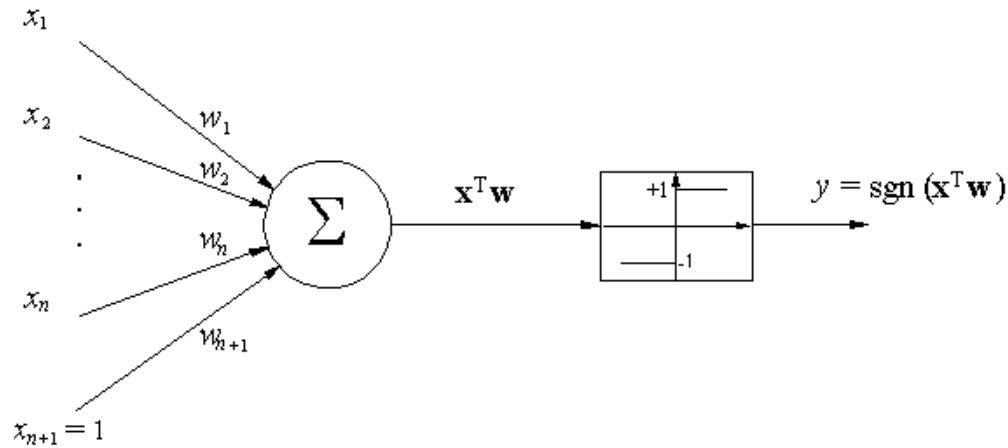
- A single learning algorithm hypothesis ?

- “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)

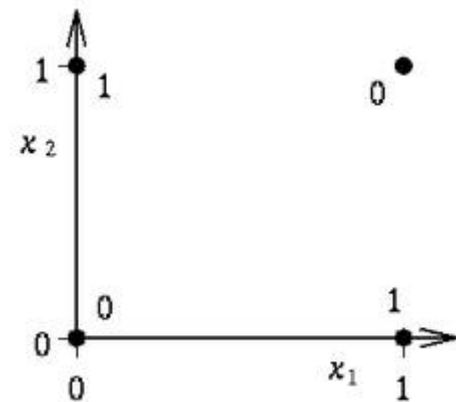
# (Artificial) Neural Networks



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



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



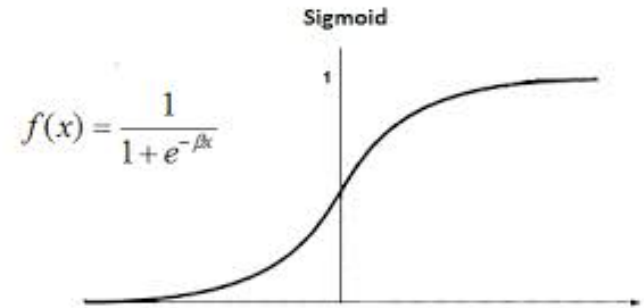


# Neural Networks



Rumelhart and McClelland – 1986:

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



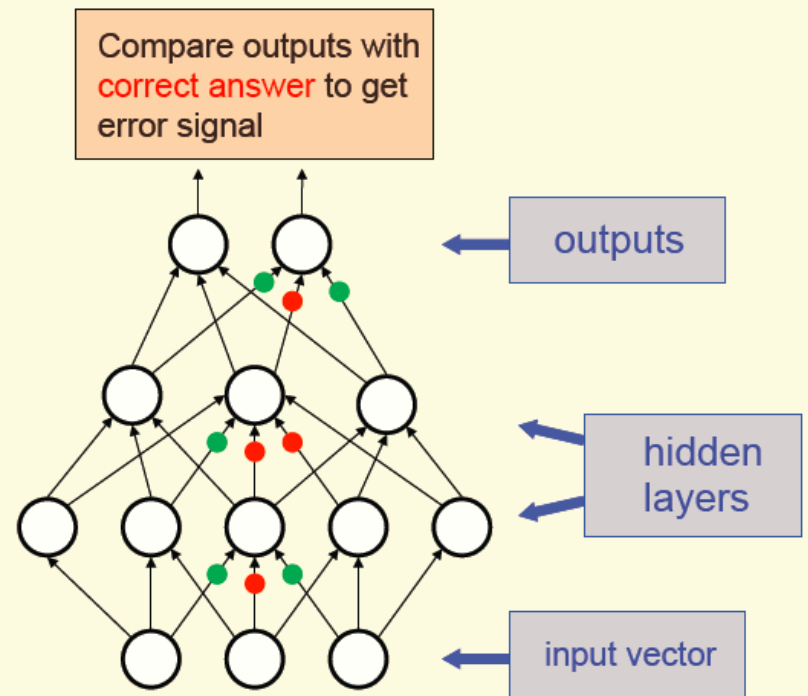
Empirical risk

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

Update weights:

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

Back-propagate error signal to get derivatives for learning



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

# What was wrong with back propagation?



10

- 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

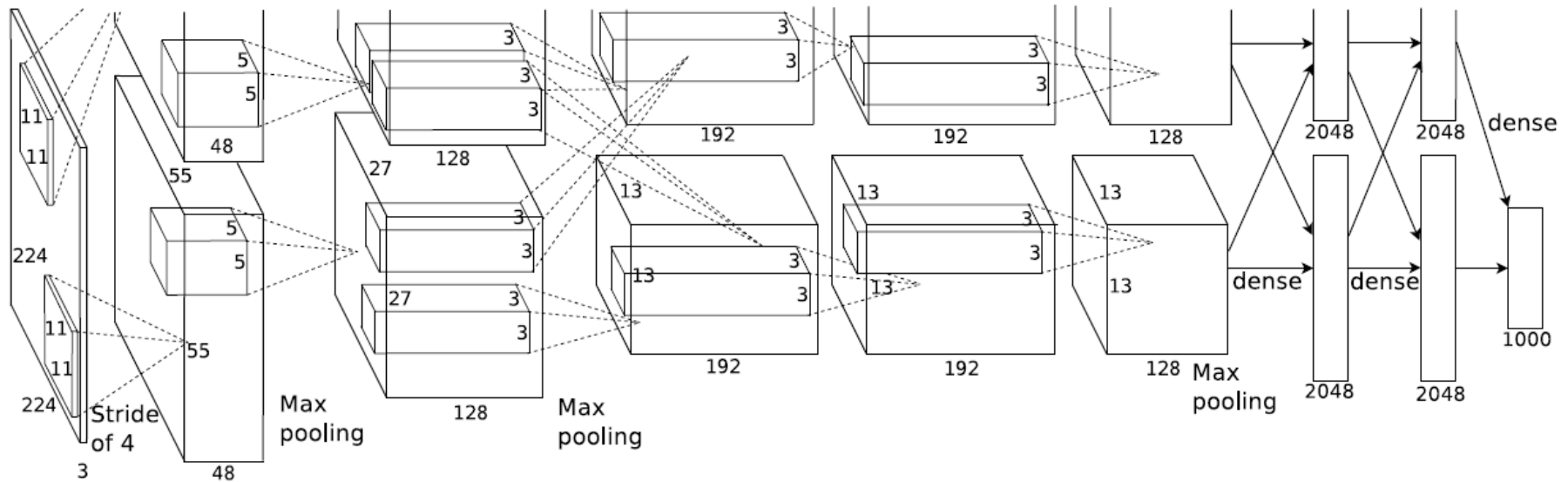
- However nowadays:
  - Weights can be initialized better (Use of unlabeled data, Restricted Boltzmann Machines)
  - Large collections of labeled data available
    - ImageNet (14M images, 21k classes, hand-labeled)
  - Reducing the number of parameters by weight sharing
    - Convolutional layers –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



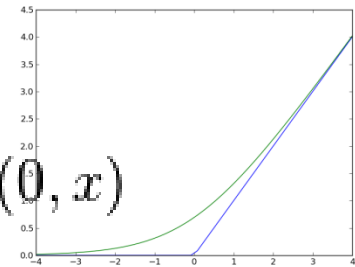
- 5 convolutional layers, 3 fully connected layers
- 60M parameters, trained on 1.2M images (~1000 examples for each category)

# Deep convolutional neural networks



12

- Additional tricks: “Devil is in the details”
  - Rectified linear units instead of standard sigmoid
    - => Mitigate vanishing gradient problem  $f(x) = \max(0, x)$
  - 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



## Novel tricks

- Network initialization
  - Mishkin, Matas. *All you need is a good init*. ICLR 2016
  - Weights initialization: zero mean, unit variance, orthogonality
  
- Batch normalization
  - Iosse, Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. NIPS 2015
  - Zero mean and unit variance weights are “supported” during training to avoid vanishing gradient

⇒ Small sensitivity to learning rate setting (can be higher, faster training – 10 times fewer epochs needed)

⇒ Regularizer (dropout can be excluded/smaller) (better optimum found)

**Input:** Values of  $x$  over a mini-batch:  $\mathcal{B} = \{x_{1\dots m}\}$ ;  
 Parameters to be learned:  $\gamma, \beta$

**Output:**  $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

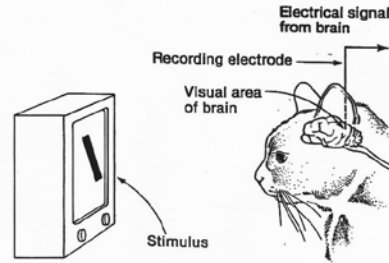
**Algorithm 1:** Batch Normalizing Transform, applied to activation  $x$  over a mini-batch.

# Deep convolutional neural networks



14

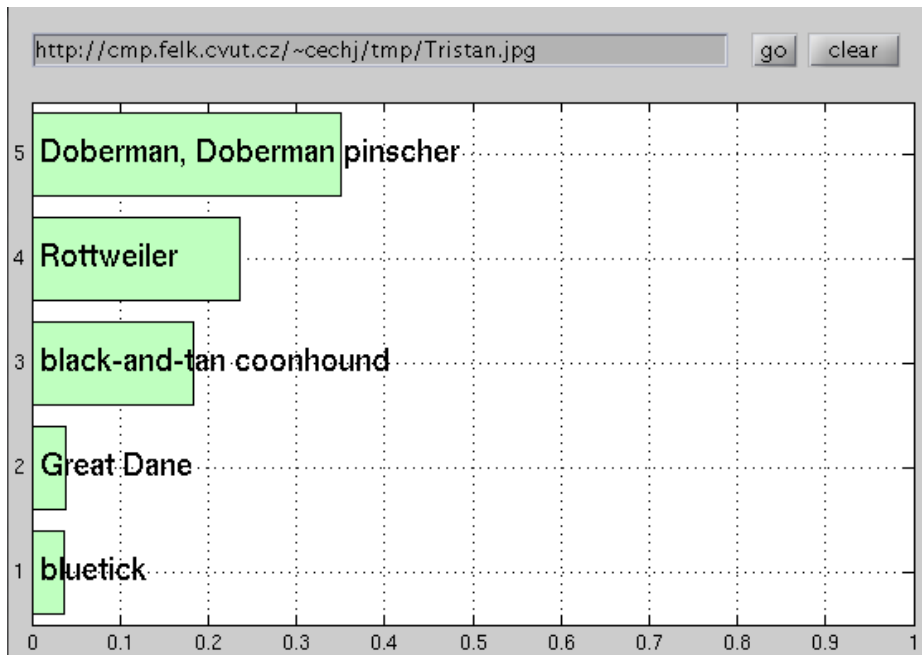
- No unsupervised pre-initialization!
  - The training is supervised by standard back-propagation
  - enough labeled data: 1.2M labeled training images for 1k categories
  - Learned filters in the first layer
    - Resemble cells in primary visual cortex



- Training time:
  - 5 days on NVIDIA GTX 580, 3GB memory (Krizhevsky, today faster)
  - 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

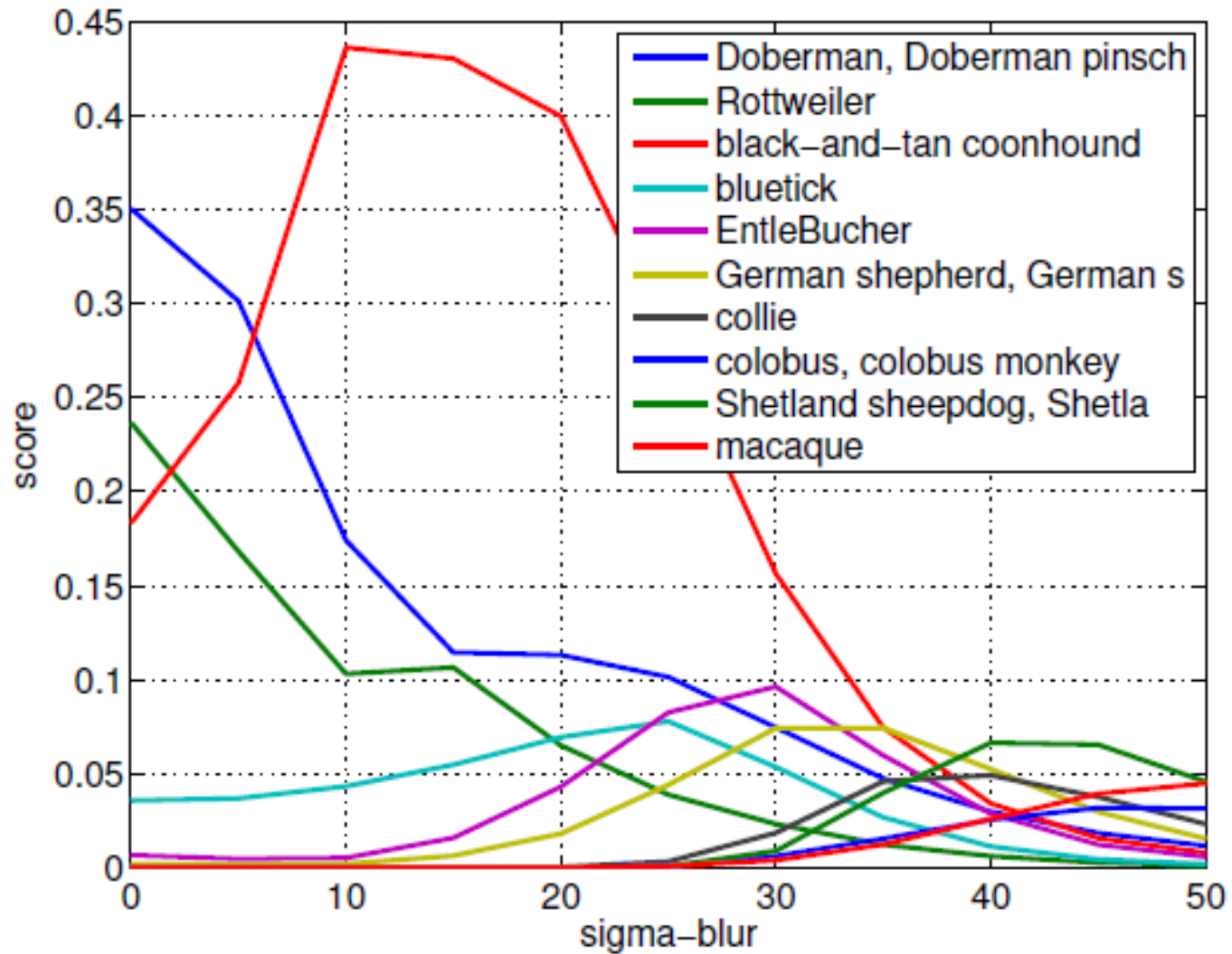
- Implementation by Yangqing Jia, <http://caffe.berkeleyvision.org/>
  - network pre-trained for 1000 categories provided
- Which categories are pre-trained?
  - 1000 “most popular” (probably mostly populated)
  - Typically very fine categories (dog breeds, plants, vehicles...)
  - Category “person”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



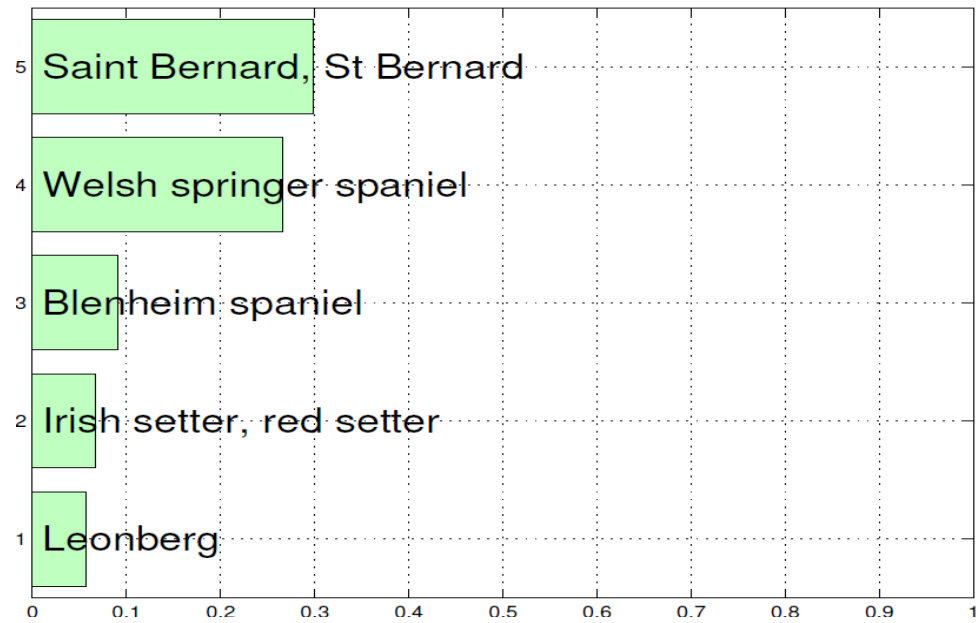
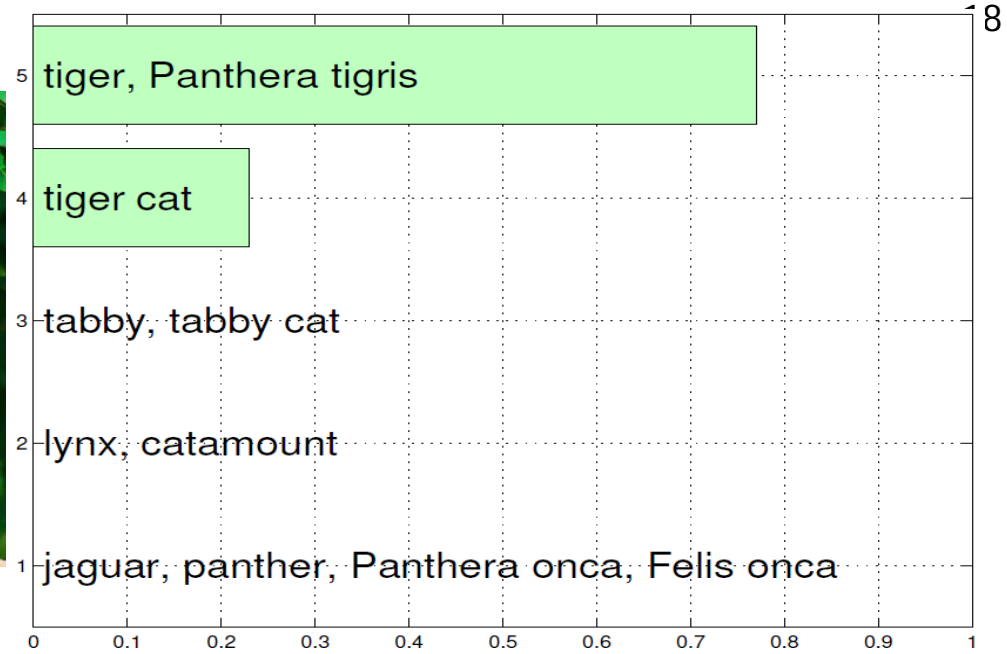




# Sensitivity to image blur

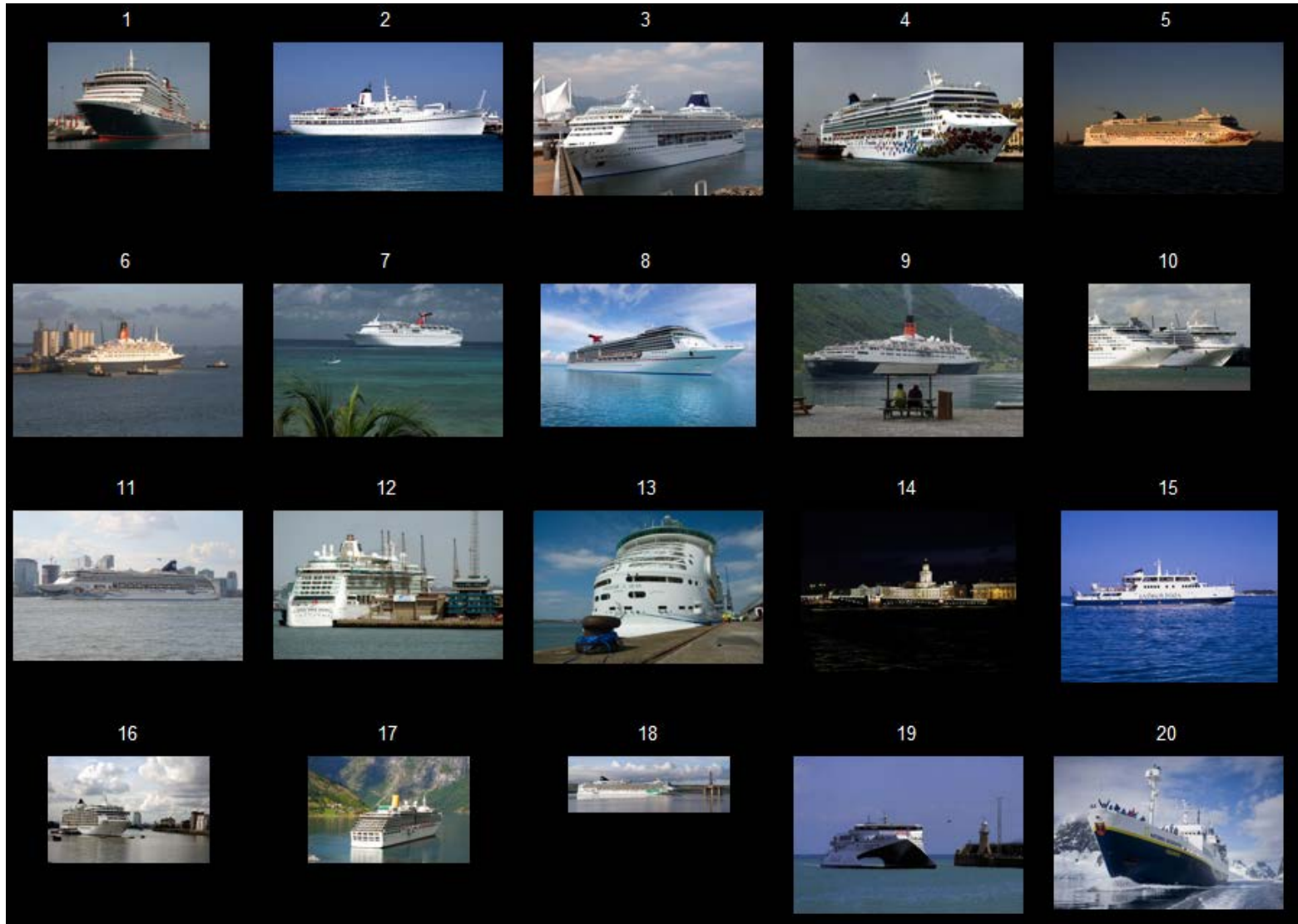


# It is not a texture only...



# 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



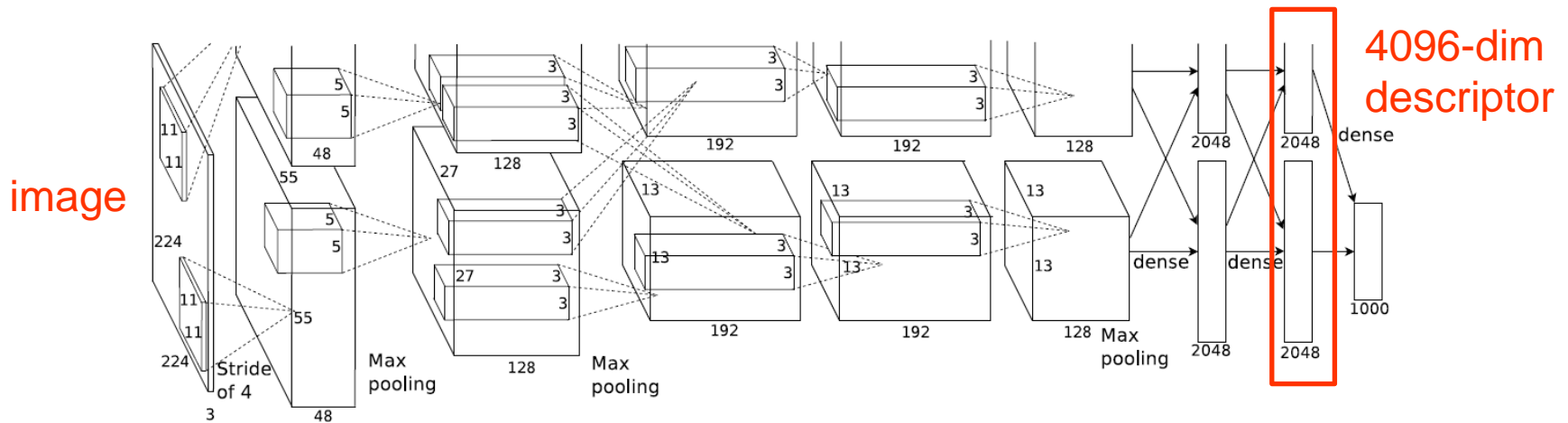
- Category: stethoscope (results out of 50k-random-Profiset)



# 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, CVPR, 2014
    - 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

1: 0	2: 2131.81	3: 2383.91	4: 2609.95	5: 2624.66
6: 2676.67	7: 2713.51	8: 2771.19	9: 2775.77	10: 2790.22
11: 2853.21	12: 2887.67	13: 2909.2	14: 2938.18	15: 3054.01
16: 3065.6	17: 3079.88	18: 3086.55	19: 3104.52	20: 3111.01



# Preliminary experiments 3: Similarity search

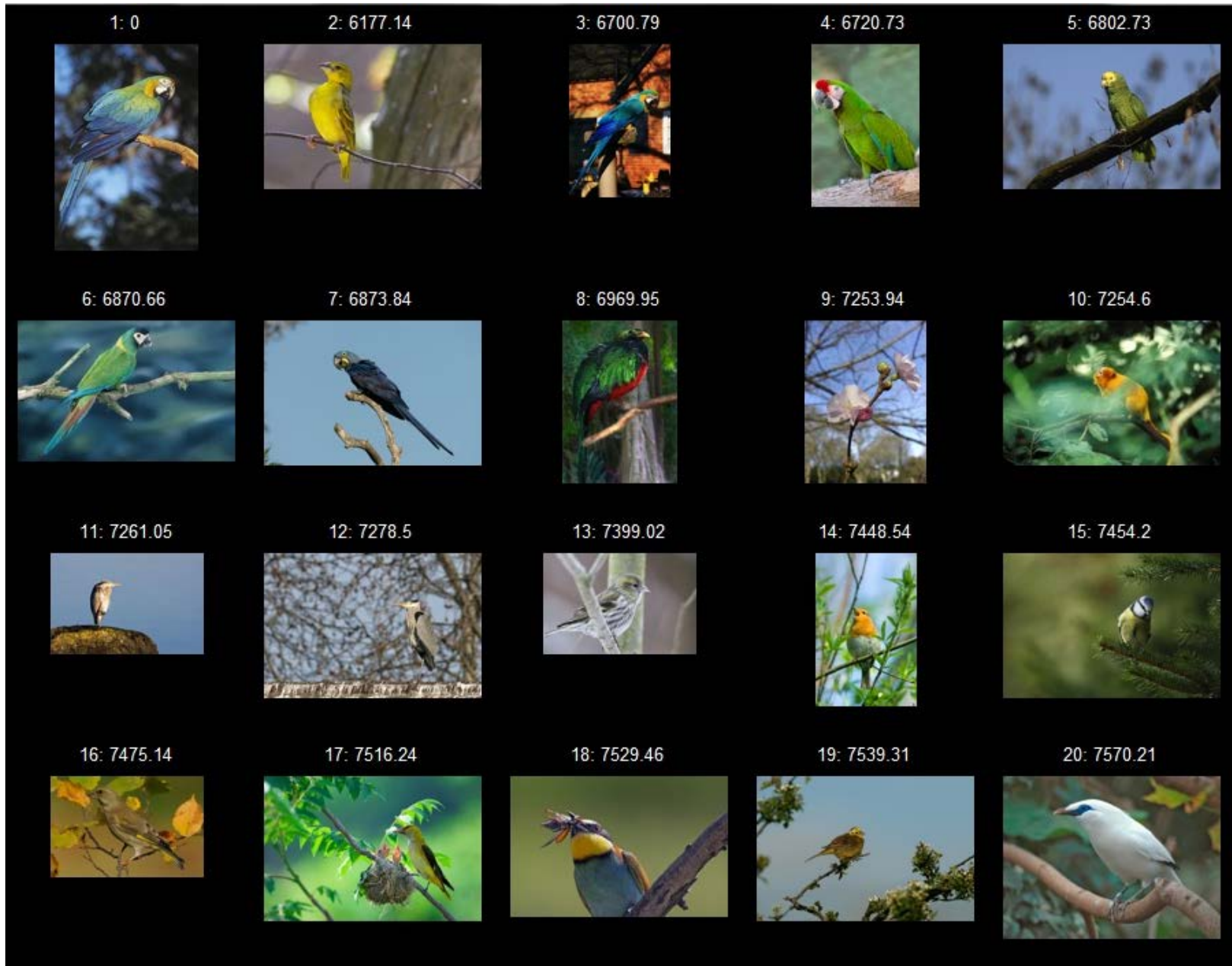
## MUFIN results



# Preliminary experiments 3: Similarity search



Caffe NN results



# Preliminary experiments 3: Similarity search

## MUFIN results



# Preliminary experiments 3: Similarity search



Caffe NN results

1: 0	2: 2184.64	3: 2542.78	4: 2621.33	5: 2671.85
6: 2768.1	7: 2861.94	8: 2880.53	9: 3021.58	10: 3071.51
11: 3075.16	12: 3083.7	13: 3137.8	14: 3191.61	15: 3208.09
16: 3211.58	17: 3212.02	18: 3216.26	19: 3217.13	20: 3221.93

# Preliminary experiments 3: Similarity search

## MUFIN results



# Preliminary experiments 3: Similarity search



Caffe NN results

1: 0	2: 2812.02	3: 2968.18	4: 3189.3	5: 3284.86
6: 3286.28	7: 3304.93	8: 3402.86	9: 3433.69	10: 3473.81
11: 3495.67	12: 3528.47	13: 3549.56	14: 3559.5	15: 3562.74
16: 3574.01	17: 3576.81	18: 3597.88	19: 3599.39	20: 3662.85



# Preliminary experiments 3: Similarity search

## MUFIN results



# Preliminary experiments 3: Similarity search



Caffe NN results

1: 0 	2: 3356.56 	3: 3368.62 	4: 3386.68 	5: 3398.03 
6: 3477.87 	7: 3561.41 	8: 3712.54 	9: 3742.91 	10: 3747.17 
11: 3749.72 	12: 3774.72 	13: 3786.68 	14: 3800.45 	15: 3826.56 
16: 3845.7 	17: 3849.7 	18: 3918.38 	19: 3952.4 	20: 3979.94 

# Preliminary experiments 3: Similarity search

## MUFIN results



# Preliminary experiments 3: Similarity search

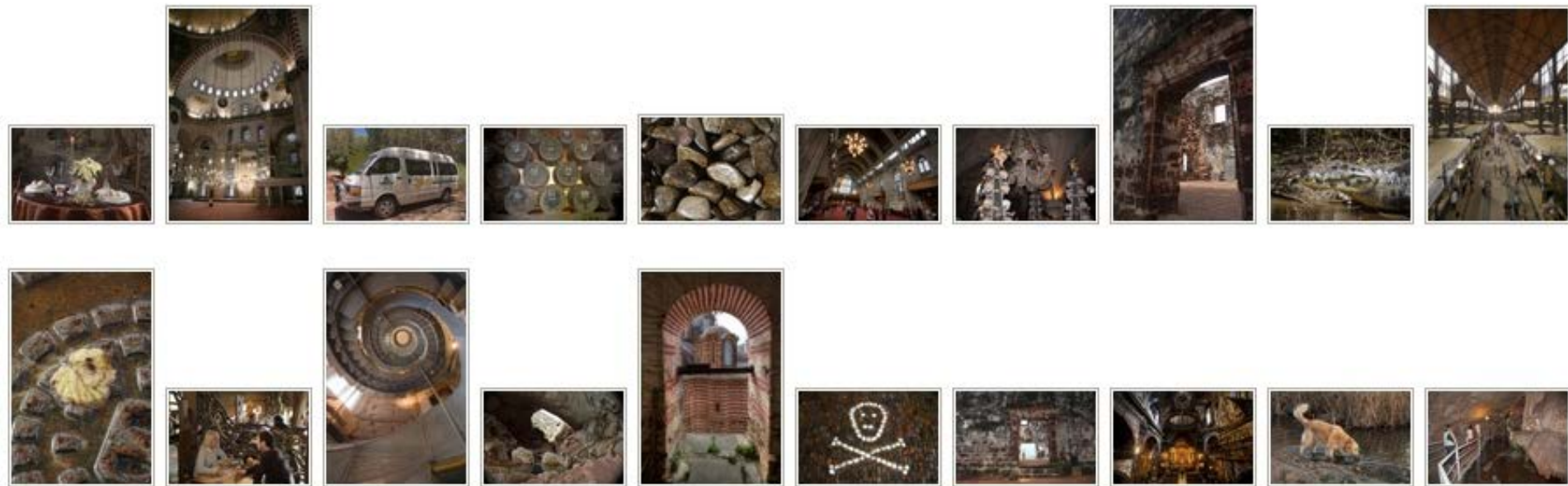


Caffe NN results

1: 0 	2: 2363.57 	3: 2446.99 	4: 2474.93 	5: 2487.54 
6: 2548.59 	7: 2584.15 	8: 2605.26 	9: 2613.48 	10: 2614.88 
11: 2678.7 	12: 2682.7 	13: 2708.36 	14: 2716.57 	15: 2775.47 
16: 2791.85 	17: 2827.6 	18: 2860.28 	19: 2889.72 	20: 2912.05 

# Preliminary experiments 3: Similarity search

## MUFIN results



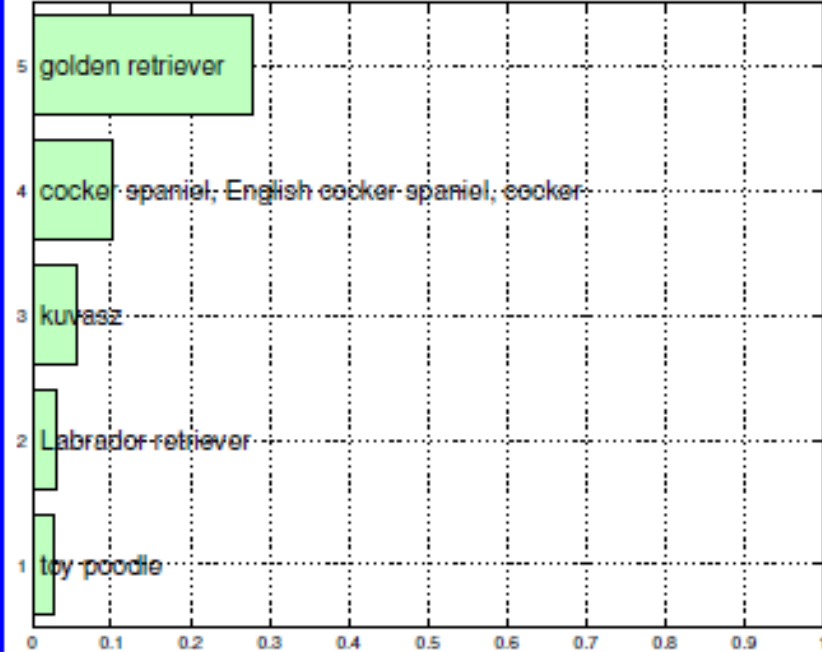
# Preliminary experiments 3: Similarity search



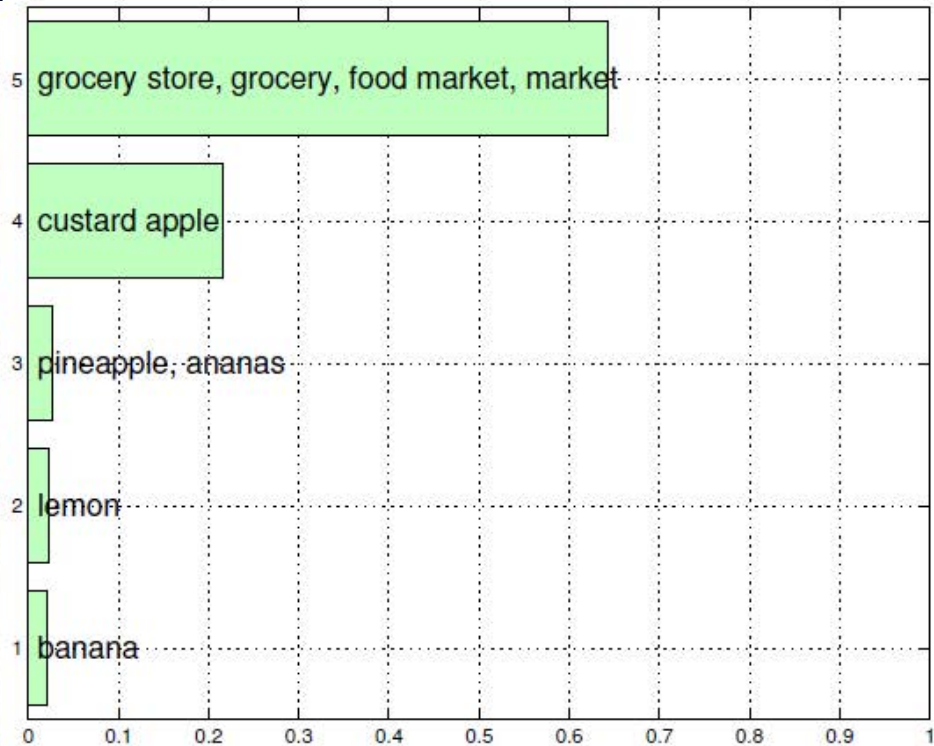
Caffe NN results

1: 0	2: 3362.93	3: 3432.31	4: 3584.88	5: 3878.5
6: 3951.13	7: 3959.28	8: 3962.82	9: 3966.25	10: 3979.92
11: 3998.15	12: 4012.28	13: 4026.56	14: 4026.59	15: 4037.17
16: 4039.89	17: 4056.42	18: 4060.53	19: 4073.56	20: 4074.8

# Multiple object classes



# Multiple object classes





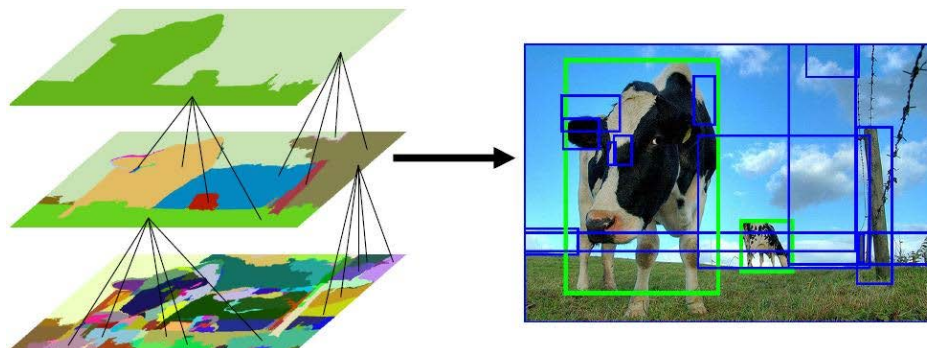
# Object detection: Deep Nets and Sliding Windows

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

=> Category independent region proposals:

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

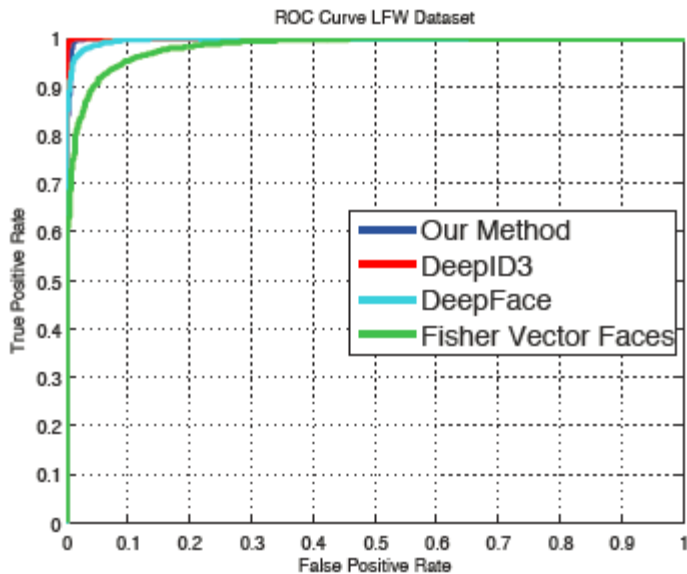
More recent approaches, see next lecture



- Edgeboxes [Zitnick-ECCV-2014]



- Face recognition, face verification
  - Architecture similar to AlexNet - very deep CNN (softmax at the last layer)
- [Taigman-ECVV-2014] DeepFace: Closing the Gap to Human-Level Performance in Face Verification (authors from Facebook)
- [Parkhi-BMVC-2015] Deep Face recognition (authors from Oxford Uni)
  - 2.6M images of 2.6k celebrities, trained net available

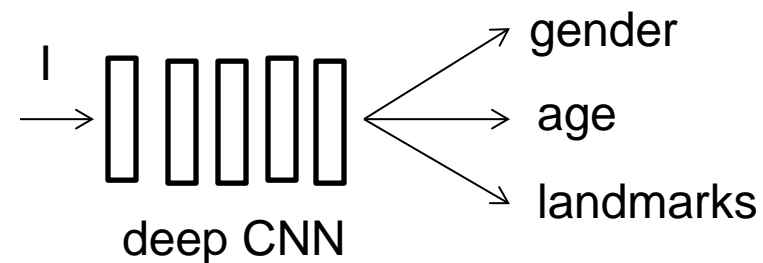


No.	Method	# Training Images	# Networks	Accuracy
1	Fisher Vector Faces	-	-	93.10
2	DeepFace (Facebook)	4 M	3	97.35
3	DeepFace Fusion (Facebook)	500 M	5	98.37
4	DeepID-2,3	Full	200	99.47
5	FaceNet (Google)	200 M	1	98.87
6	FaceNet+ Alignment (Google)	200 M	1	99.63
7	(VGG Face)	2.6 M	1	98.78

- Face represented by penultimate layer response, similarity search, large scale indexing

# Face interpretation tasks

- Facial landmarks, Age / Gender estimation
  - Multitask network
    - Shared representation
    - Combination of both classification and regression problems



# Age estimation – How good the network is?

- Our survey  
~20 human subjects , ~100 images of 2 datasets

## MORPH dataset

True: 22, MAE: 18.8



True: 36, MAE: 17.8



True: 33, MAE: 16.3



True: 22, MAE: 16.1



True: 25, MAE: 16.0



## IMDB dataset

True: 25, MAE: 0.5



True: 66, MAE: 1.0



True: 29, MAE: 1.0



True: 19, MAE: 1.0



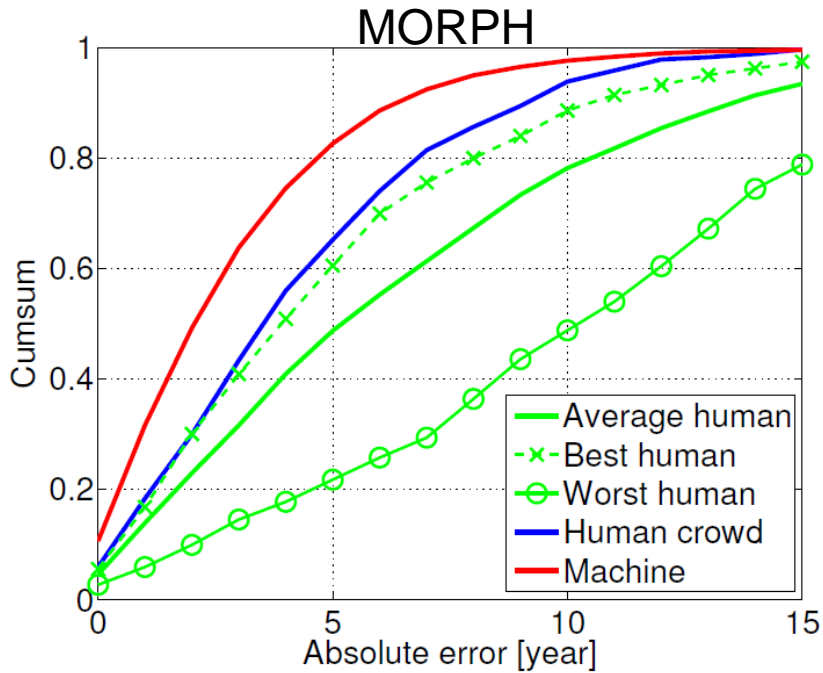
True: 43, MAE: 1.0



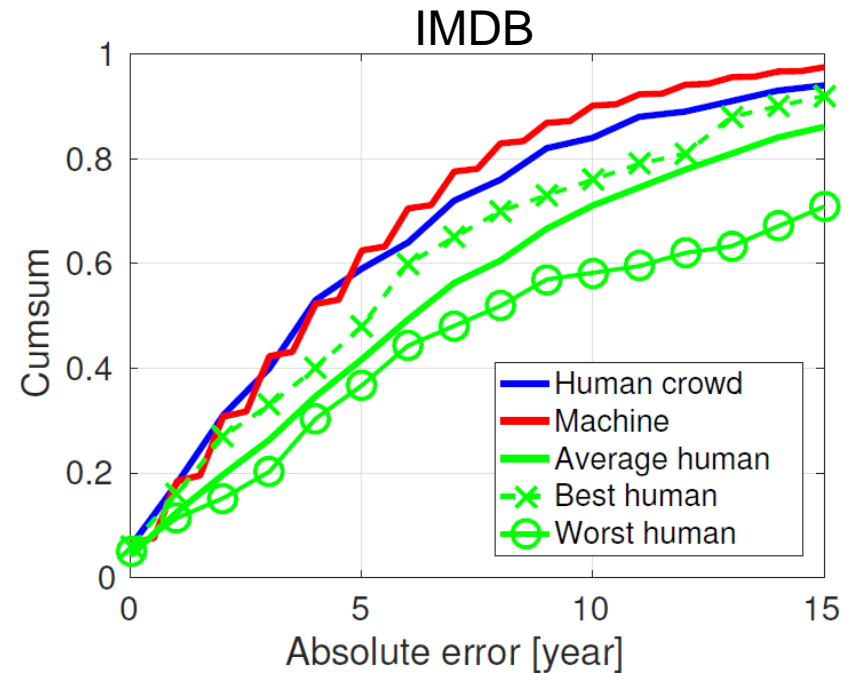
# Age estimation – How good the network is?



- Better than average human...



	MAE	CS5	MaxAE
Average human :	6.8	48.6	24.1
Human crowd :	4.7	65.1	19.0
Machine :	3.2	82.6	26.0



	MAE	CS5	MaxAE
Average human :	8.2	41.7	31.5
Human crowd :	5.7	59.0	21.0
Machine :	5.1	62.5	42.7

- [Franc-Cech-BwildW-2017]
- Network runs real-time on CPU

# General recipe to use deep neural networks



46

- 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 (sigmoid)
  - 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... 😊
- Novel:
  - Use Batch Normalization [Ioffe-Szegedy-NIPS-2015]
- Experience:
  - Data matters (the more data the better), data augmentation helps
  - Network architecture (if deep enough) is surprisingly not critical

# Conclusions



47

- It efficiently learns the abstract representation (shared among classes)
  - The network miraculously captures semantics...
- Low computational demands for running, Training needs GPU
- Many “deep” toolboxes: Caffe (Berkeley), MatconvNet (Oxford), TensorFlow (Google), Theano (Montreal), Torch, ...
- 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

- Do we understand enough what is going on?

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



Human\_Abducted\_by\_UFO.mp4

