Ingénierie des systèmes décisionnels (NFE212)
Deep Learning

Nicolas Thome
Conservatoire Nationnal des Arts et Métiers (Cnam)
Laboratoire CEDRIC

le cnam

Cédric

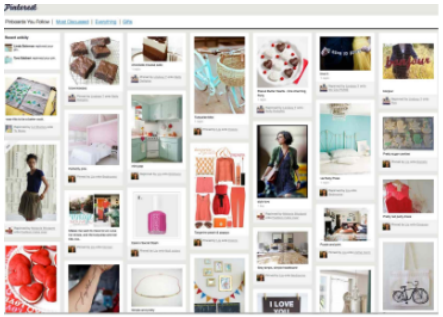
Outline

Context

Big Data

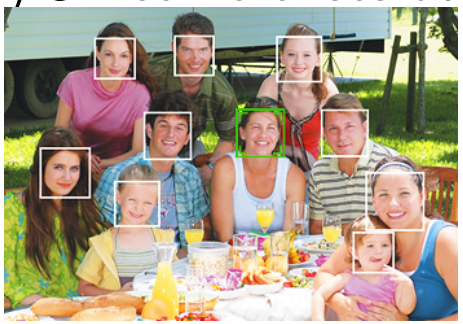
- Superabundance of data: images, videos, audio, text, use traces, *etc*

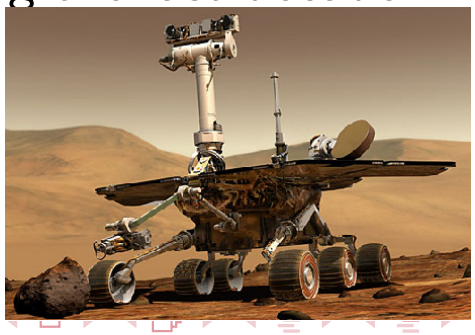
BBC: 2.4M videos


Facebook: 350B images
1B each day


100M monitoring cameras

- Obvious need to access, search, or classify these data: **Recognition**
- Huge number of applications: mobile visual search, robotics, autonomous driving, augmented reality, medical imaging *etc*
- Leading track in major ML/CV conferences during the last decade



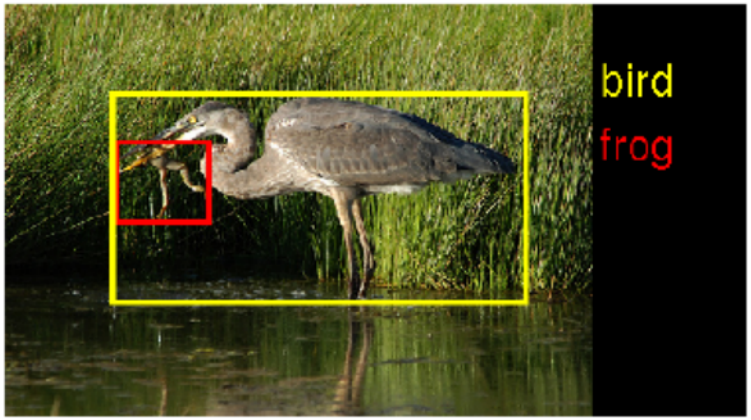


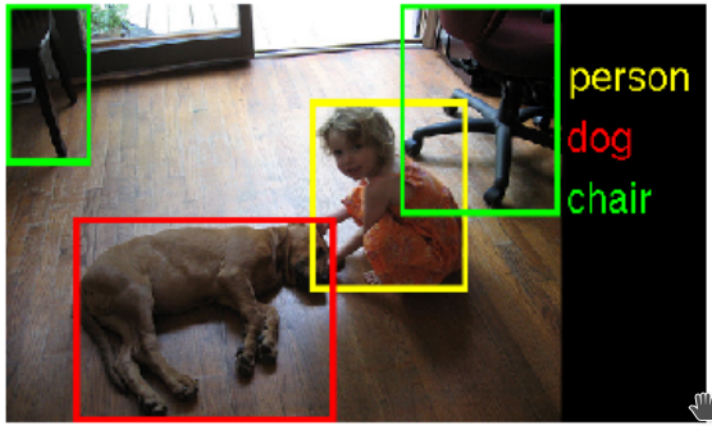
Introduction to Deep Learning

Recognition and classification

- Classification: data → set of pre-defined classes
- Recognition much more general than classification, *e.g.*
  - Object Localization in images
  - Sequence prediction for text, speech, audio, *etc*
- Many tasks can be cast as classification problems

⇒ **Importance of classification**

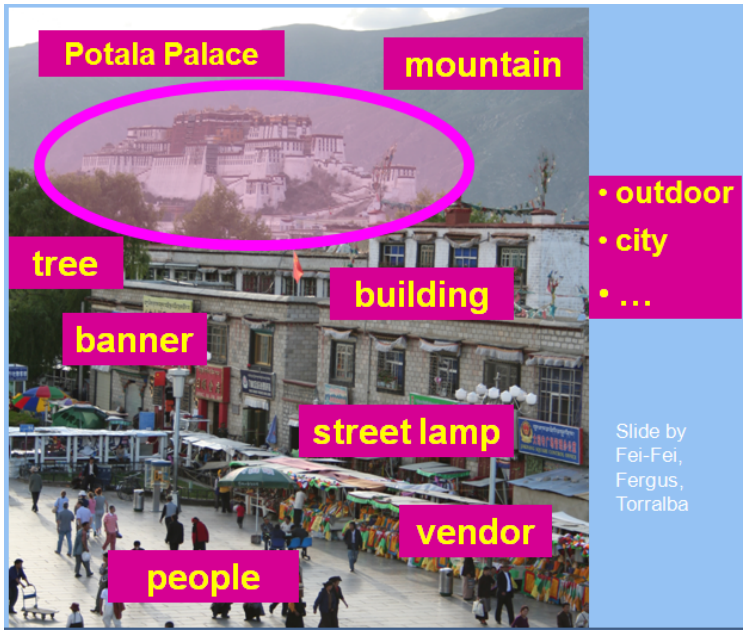




Focus on Visual Recognition: Perceiving Visual World

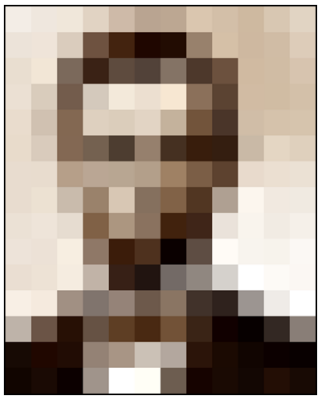
- Visual Recognition: archetype of low-level signal understanding
- Supposed to be a master class problem in the early 80's
- Certainly the most impacted topic by deep learning

- Scene categorization
- Object localization
- Context & Attribute recognition
- Rough 3D layout, depth ordering
- Rich description of scene, *e.g.* sentences



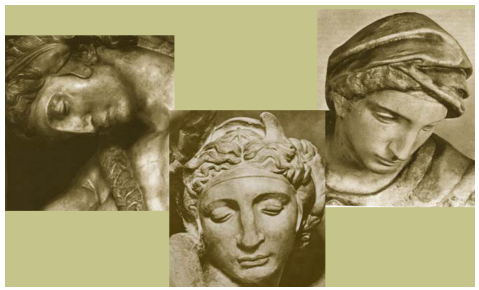
Recognition of low-level signals

Challenge: filling the semantic gap



What we perceive *vs*
What a computer sees

243	239	240	225	206	185	188	218	211	206	216	225
242	239	218	110	67	31	34	152	213	206	208	221
243	242	123	58	94	82	132	77	108	208	208	215
235	217	115	212	243	236	247	139	91	209	208	211
233	208	131	222	219	226	196	114	74	208	213	214
232	217	131	116	77	150	69	56	52	201	228	223
232	232	182	186	184	179	159	123	93	232	235	235
232	236	201	154	216	133	129	81	175	252	241	240
235	238	230	128	172	138	65	63	234	249	241	245
237	236	247	143	59	78	10	94	255	248	247	251
234	237	245	193	55	33	115	144	213	255	253	251
248	245	161	128	149	109	138	65	47	156	239	255
190	107	39	102	94	73	114	58	17	7	51	137
23	32	33	148	168	203	179	43	27	17	12	8
17	26	12	160	255	255	109	22	26	19	35	24

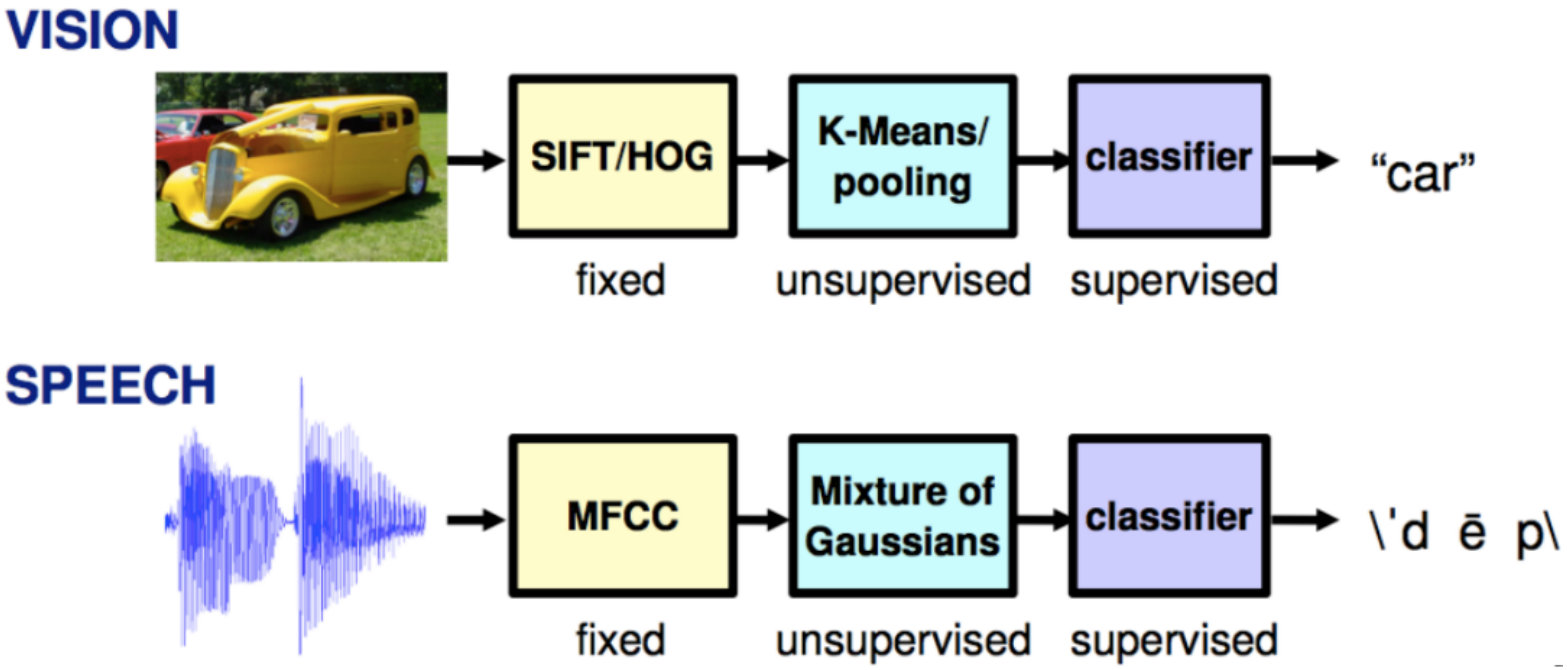


- Illumination variations
- View-point variations
- Deformable objects
- intra-class variance
- *etc*

⇒ **How to design "good" intermediate representation ?**

Deep Learning (DL) & Recognition of low-level signals

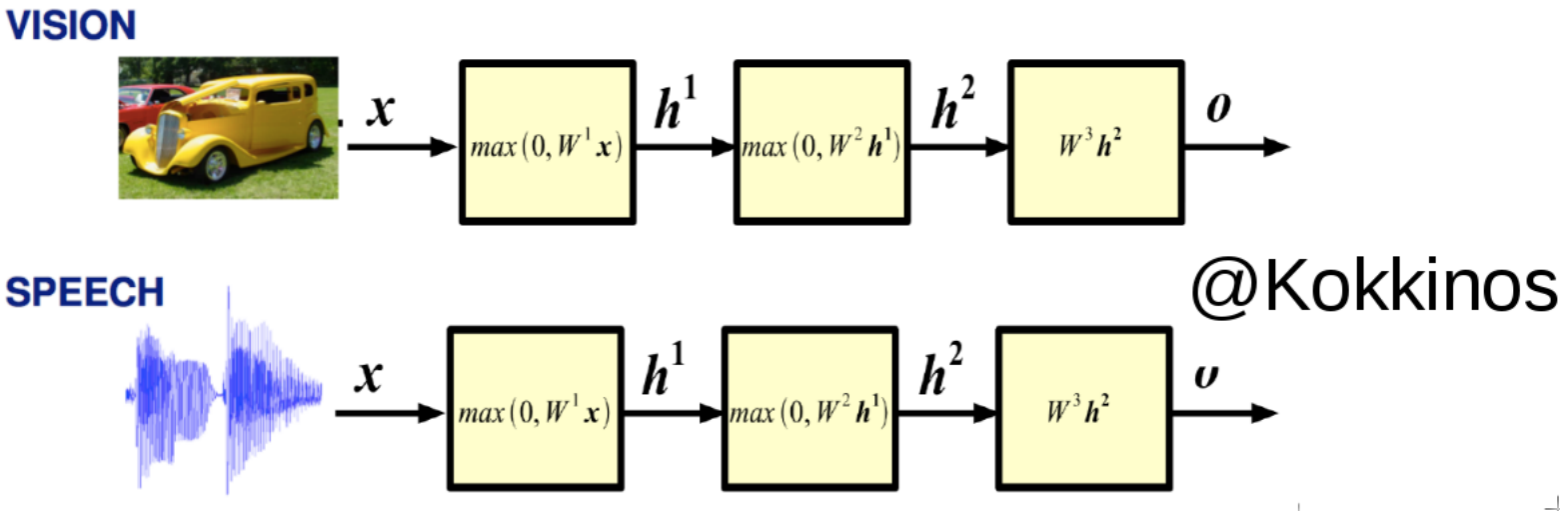
- DL: breakthrough for the recognition of low-level signal data
- Before DL: handcrafted intermediate representations for each task
  - ⊖ Needs expertise (PhD level) in each field
  - ⊖ Weak level of semantics in the representation



@Kokkinos

Deep Learning (DL) & Recognition of low-level signals

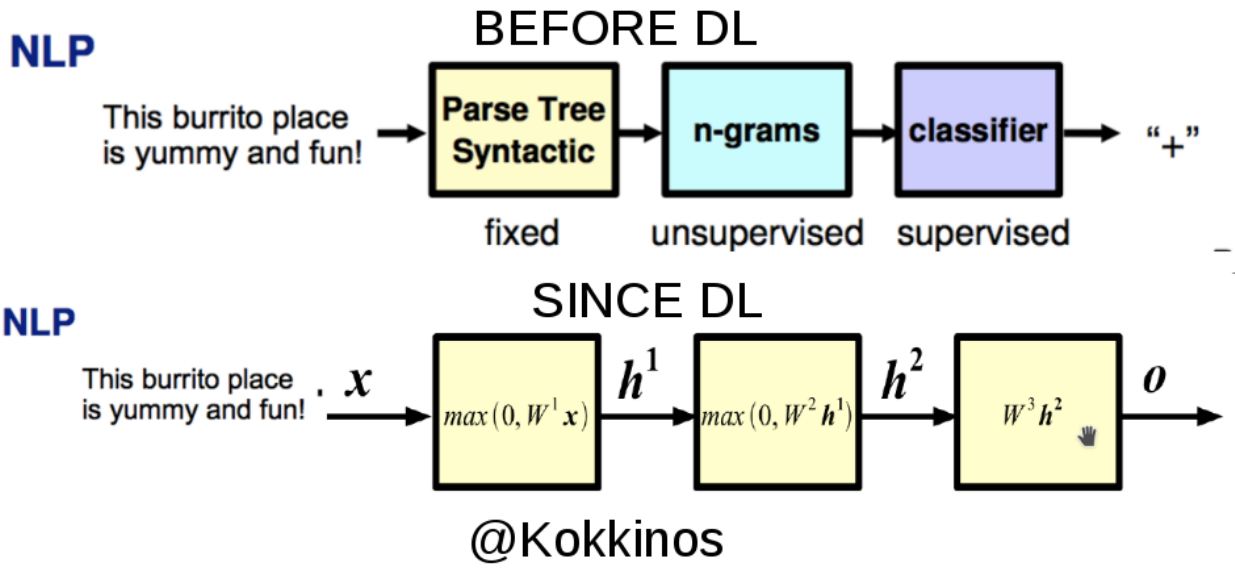
- DL: breakthrough for the recognition of low-level signal data
- Since DL: automatically **learning intermediate representations**
  - ⊕ Outstanding experimental performances >> handcrafted features
  - ⊕ Able to learn high level intermediate representations
  - ⊕ Common learning methodology ⇒ field independent, no expertise

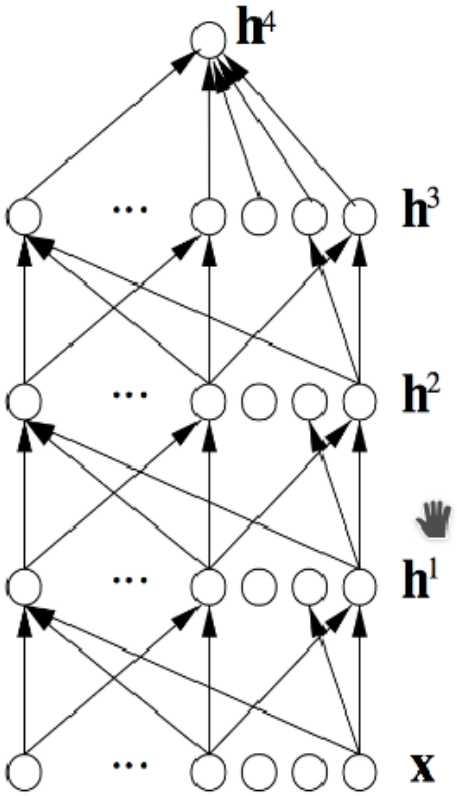


Deep Learning (DL) & Representation Learning

- DL: breakthrough for representation learning
  - Automatically learning intermediate levels of representation

- Ex: Natural language Processing (NLP)





Outline

The formal neuron [MP43]

x_i: inputs

w_i, b: weights

f: activation function

y: output of the neuron

$$y = f(w^\top x + b)$$

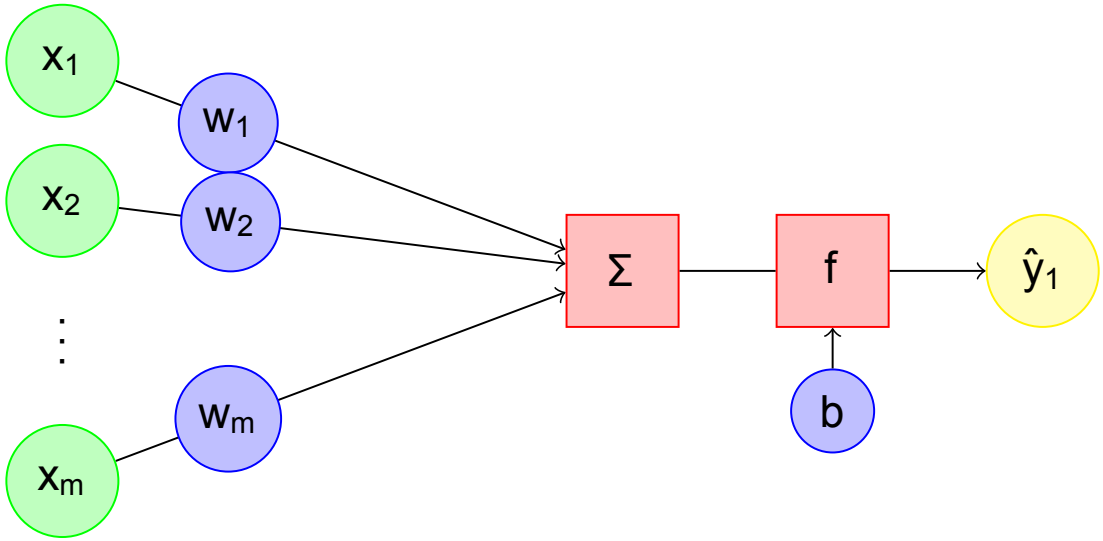
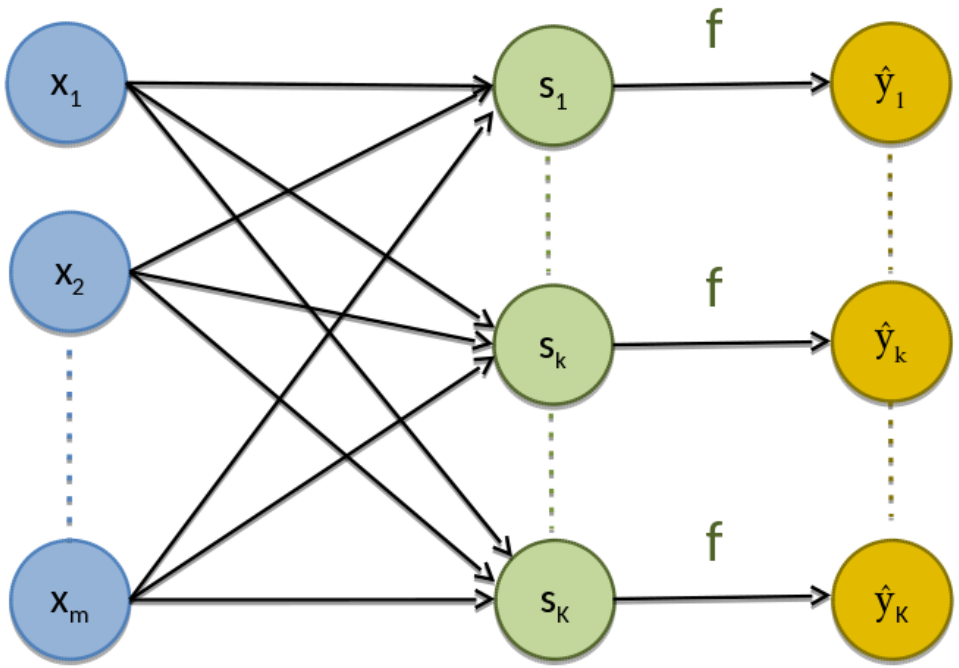


Figure: The formal neuron – Credits: R. Herault

Neural Networks

- Stacking several formal neurons ⇒ **Perceptron**



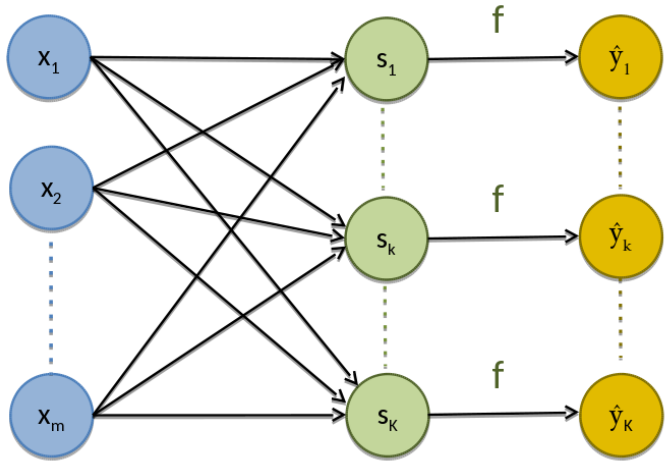
Perceptron and Multi-Class Classification

- **Soft-max Activation**:

$$\hat{y}_k = f(s_k) = \frac{e^{s_k}}{\sum_{k'=1}^{K} e^{s_{k'}}}$$

- **Probabilistic interpretation for multi-class classification**:
  - Each output neuron $\Leftrightarrow$ class
  - $\hat{y}_k \sim P(k/\mathrm{x}, \mathrm{w})$

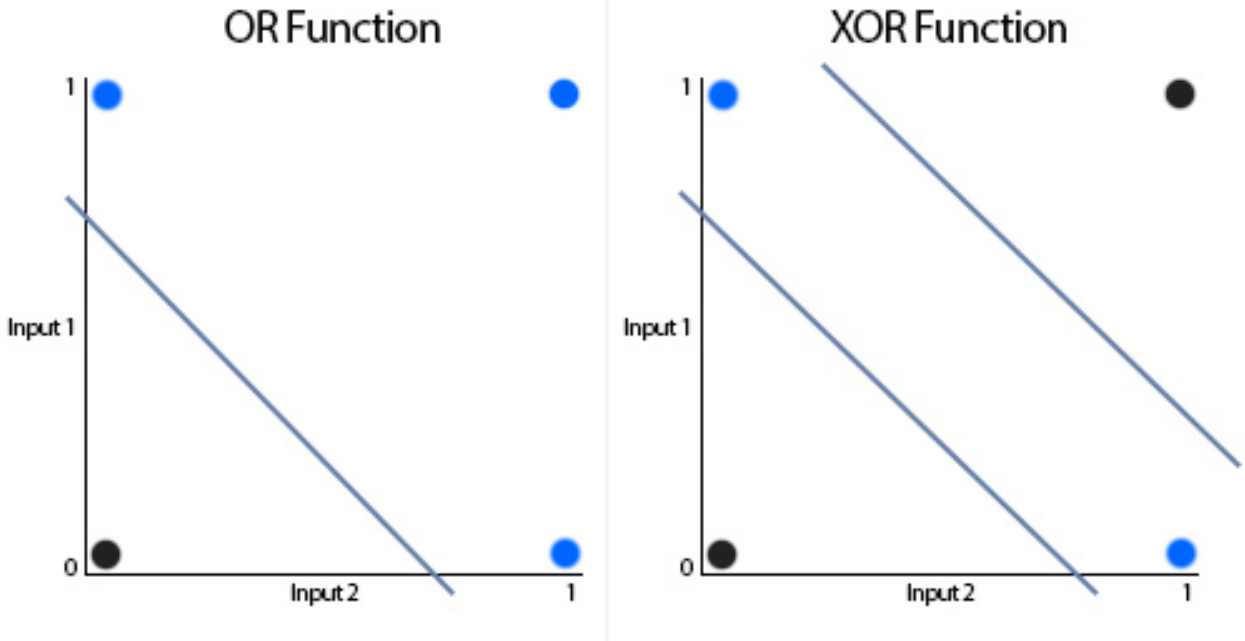
$\Rightarrow$ **Logistic Regression (LR) Model !**



Beyond Linear Prediction

X-OR Problem

- Logistic Regression (LR): NN with 1 input layer & 1 output layer
- LR: limited to linear decision boundaries
- **X-OR:** NOT 1 and 2 OR NOT 2 AND 1
  - **X-OR: Non linear decision function**



The Multi-Layer Perceptron (MLP)

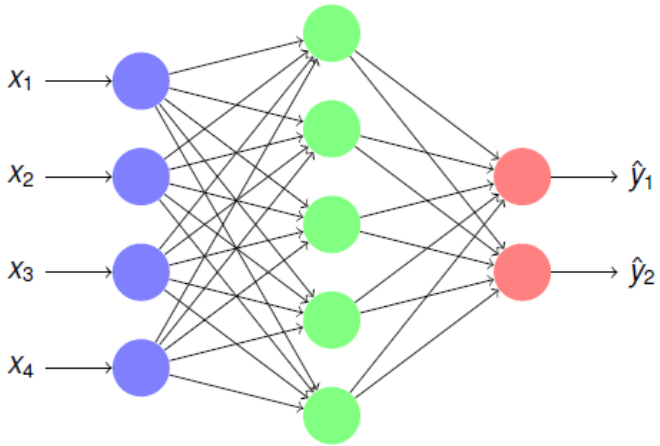


Figure: Perceptron with 1 hidden layer – Credits: R. Herault

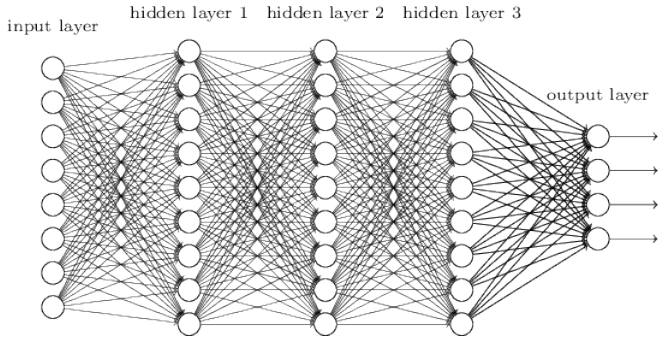
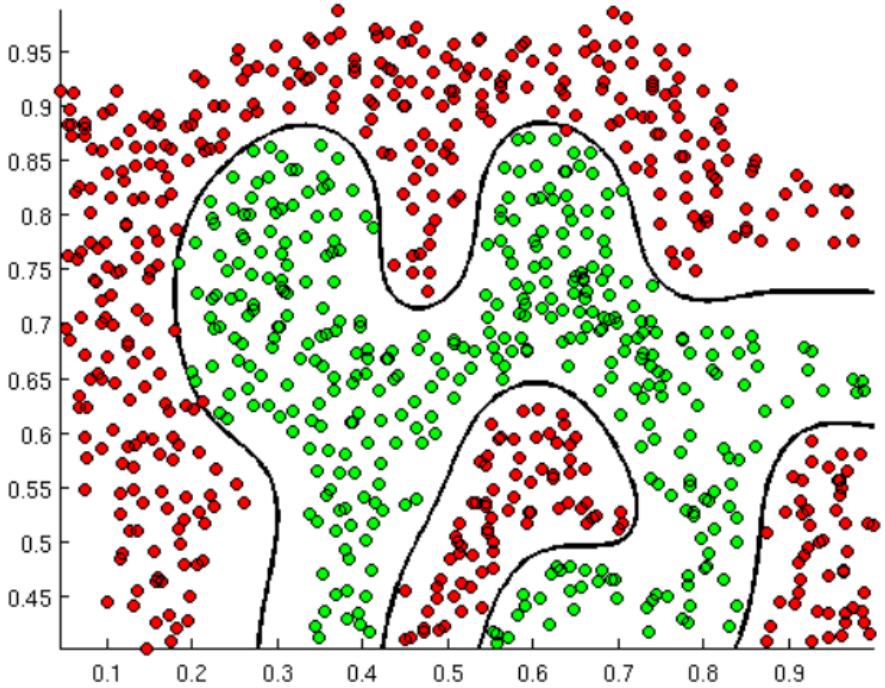


Figure: Stacking more layers, toward "deep learning" – Credits: M. Nielsen

- Basis of the "deep learning" field
- Principle: Stacking layers of neural networks to allow more complex and rich functions
- With a hidden layer, **can approximate any function** given enough hidden units [Cyb89]
- Can be seen as **different levels of abstraction** from low-level features to the high-level ones

The Multi-Layer Perceptron (MLP)

- **Neural network with one single hidden layer ⇒ universal approximator [Cyb89]**
  - Ex for classification: any decision boundaries can be expressed

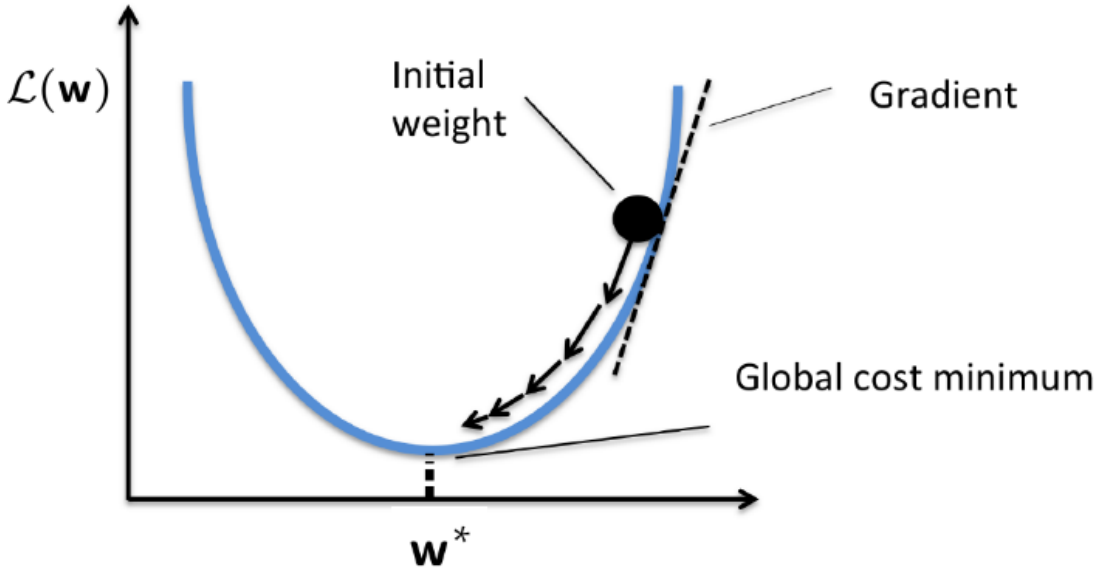


Training Multi-Layer Perceptron (MLP)

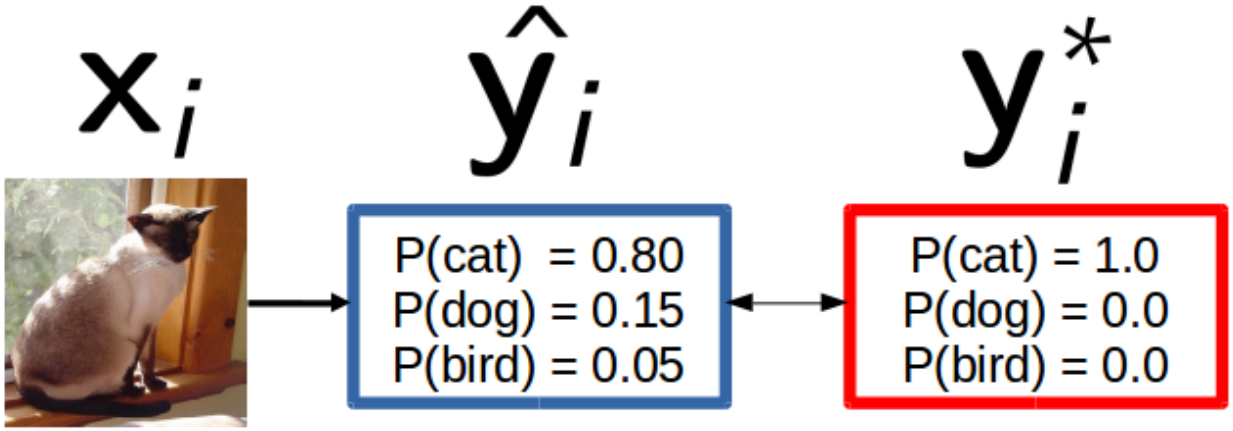
- Input x, output y
- A parametrized model $x \Rightarrow y$: $f_w(x_i) = \hat{y}_i$
- Supervised context: training set $\mathcal{A} = \{(x_i, y_i^*)\}_{i \in \{1,2,\ldots,N\}}$
  - A loss function $\ell(\hat{y}_i, y_i^*)$ for each annotated pair (x_i, y_i^*)
- Assumptions: parameters $w \in \mathbb{R}^d$ continuous, $\mathcal{L}$ differentiable
- Gradient $\nabla_w = \frac{\partial \mathcal{L}}{\partial w}$: steepest direction to decrease loss $\mathcal{L}$

MLP Training

- Gradient descent algorithm:
  - Initialyze parameters w
  - Update: $\mathrm{w}^{(t+1)} = \mathrm{w}^{(t)} - \eta \frac{\partial \mathcal{L}}{\partial \mathrm{w}}$
  - Until convergence, *e.g.* $\|\nabla_{\mathrm{w}}\|^2 \approx 0$



MLP Training: loss function



- **Input x_i, ground truth output supervision y_i^***
- One hot-encoding for y_i^*:

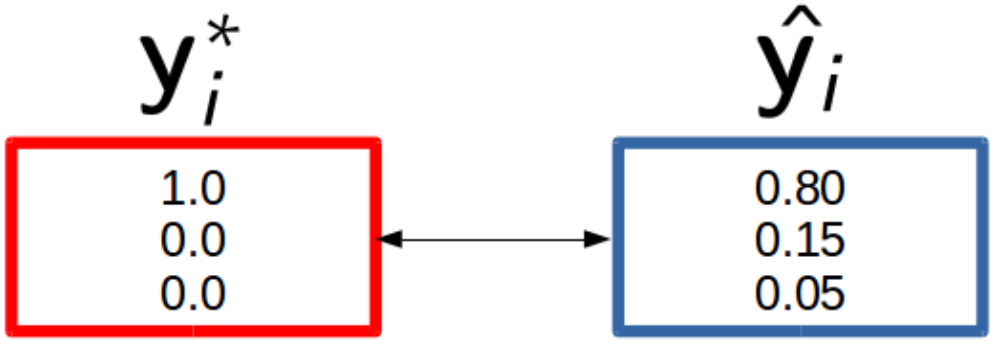
$$y^*_{c,i} = \begin{cases} 1 & \text{if c is the groud truth class for } x_i \\ 0 & \text{otherwise} \end{cases}$$

MLP Training

- Loss function: multi-class Cross-Entropy (CE) ℓ_{CE}
- ℓ_{CE}: Kullback-Leiber divergence between y_i^* and $\hat{y}_i$

$$\ell_{CE}(\hat{y}_i, y_i^*) = KL(y_i^*, \hat{y}_i) = -\sum_{c=1}^{K} y_{c,i}^* log(\hat{y}_{c,i}) = -log(\hat{y}_{c^*,i})$$

- KL asymmetric: $KL(\hat{y}_i, y_i^*) \neq KL(y_i^*, \hat{y}_i)$



$$KL(y_i^*, \hat{y}_i) = -log(\hat{y}_{c^*,i}) = -log(0.8) \approx 0.22$$

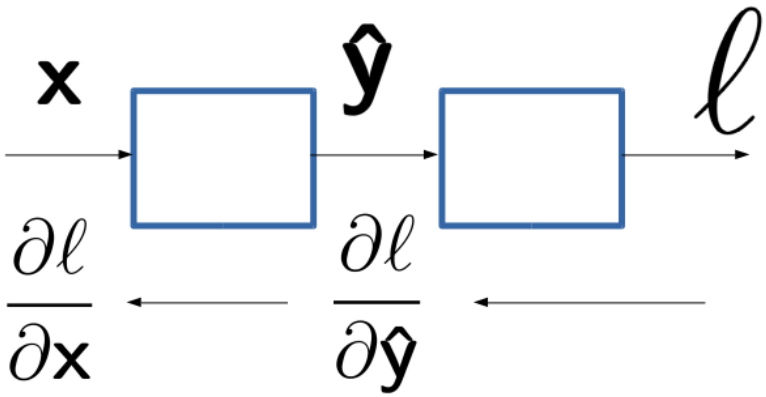
MLP Training: Backpropagation

- $\mathcal{L}_{CE}(\mathrm{W}, \mathrm{b}) = \frac{1}{N} \sum_{i=1}^{N} \ell_{CE}(\hat{\mathrm{y}}_i, \mathrm{y}_i^*) = -\frac{1}{N} \sum_{i=1}^{N} log(\hat{y}_{c^*,i})$
- ℓ_{CE} smooth convex upper bound of $\ell_{0/1}$
  $\Rightarrow$ **gradient descent optimization**
- Gradient descent: $\mathrm{W}^{(t+1)} = \mathrm{W}^{(t)} - \eta \frac{\partial \mathcal{L}_{CE}}{\partial \mathrm{W}}$
  $(\mathrm{b}^{(t+1)} = \mathrm{b}^{(t)} - \eta \frac{\partial \mathcal{L}_{CE}}{\partial \mathrm{b}})$
- **MAIN CHALLENGE: computing** $\frac{\partial \mathcal{L}_{CE}}{\partial \mathrm{W}} = \frac{1}{N} \sum_{i=1}^{N} \frac{\partial \ell_{CE}}{\partial \mathrm{W}}$?

  $\Rightarrow$ Key Property: chain rule $\frac{\partial \mathrm{x}}{\partial z} = \frac{\partial \mathrm{x}}{\partial \mathrm{y}} \frac{\partial \mathrm{y}}{\partial z}$

  $\Rightarrow$ **Backpropagation of gradient error!**

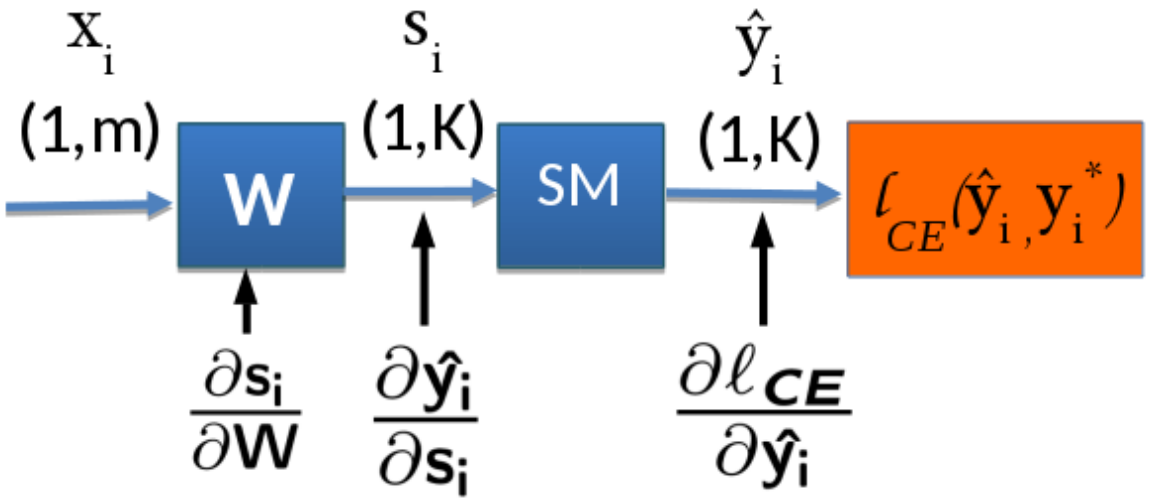
Chain Rule



$$\frac{\partial \ell}{\partial x} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial x}$$

- Logistic regression:

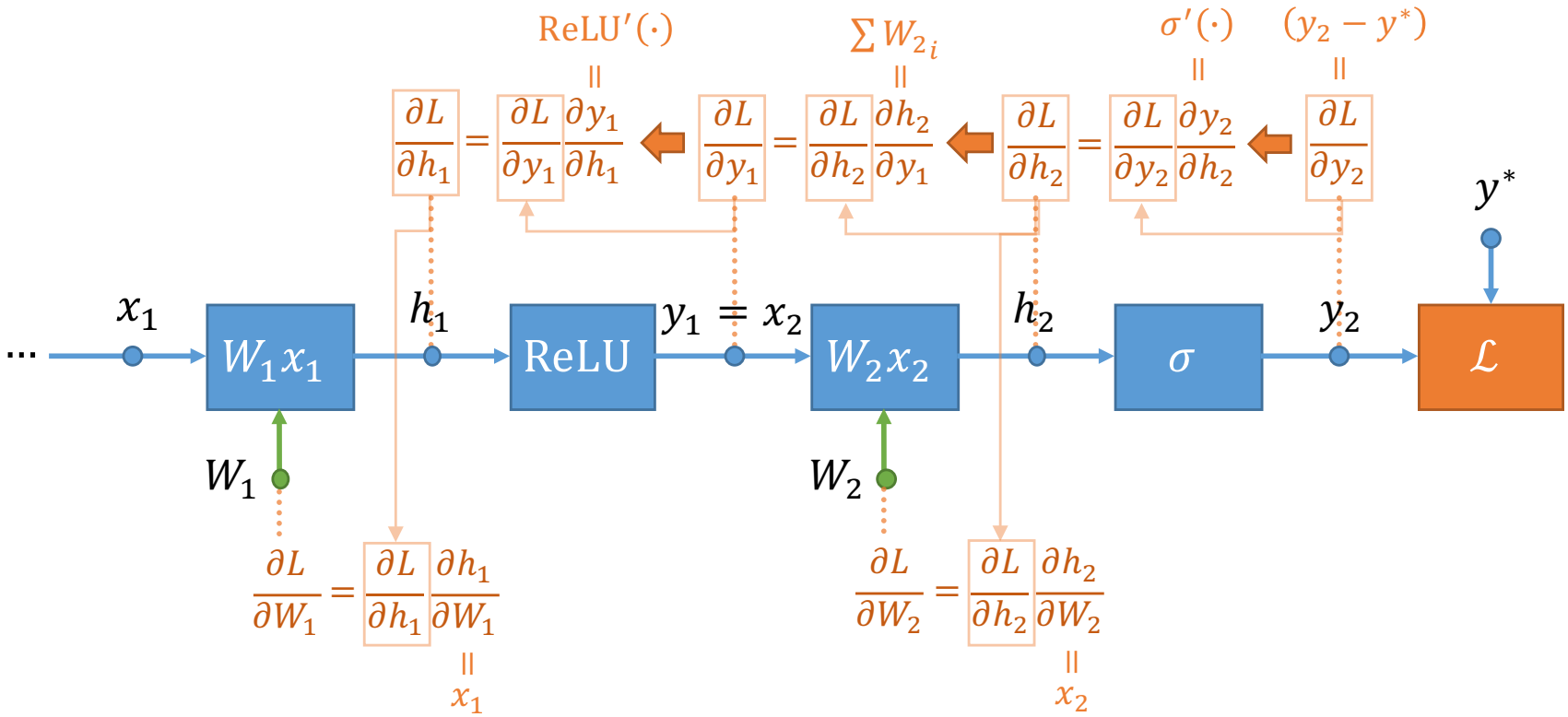
$$\frac{\partial \ell_{CE}}{\partial W} = \frac{\partial \ell_{CE}}{\partial \hat{y}_i} \frac{\partial \hat{y}_i}{\partial s_i} \frac{\partial s_i}{\partial W}$$



Backpropagation

- Extension to deeper networks: same principle

$$W_i^{(t+1)} = W_i^{(t)} - \lambda \boxed{\frac{\partial L}{\partial W_i}}$$



Training issues

Optimization

- Training with gradient descent: too slow even for moderate dimensionality dataset size
- Using Stochatic Gradient Descent (SGD)
- Issues with DEEP fully connected MLP: exploding and vanishing gradient
  $\Rightarrow$ difficult to train such models

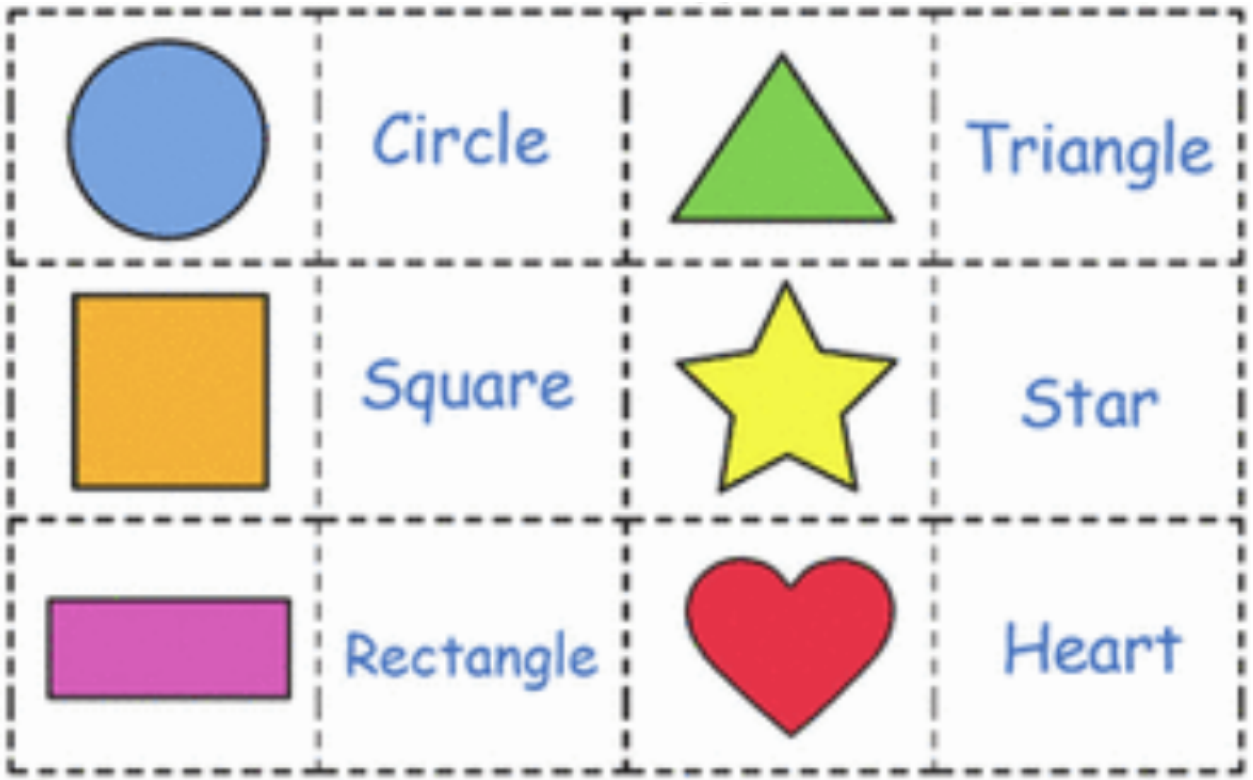
Controlling Overfitting

- Early stopping in validation set
- Weight decay: $\sim \ell_2$ regularization

Outline

Fully Connected Networks: Limitations

- Fully connected networks: no assumption on data structure
  - Structure can be learned but need lots of annoatated data
  - Prior knowlege on data structure ⇒ useful
- Example: MLP training for shape recognition from color images

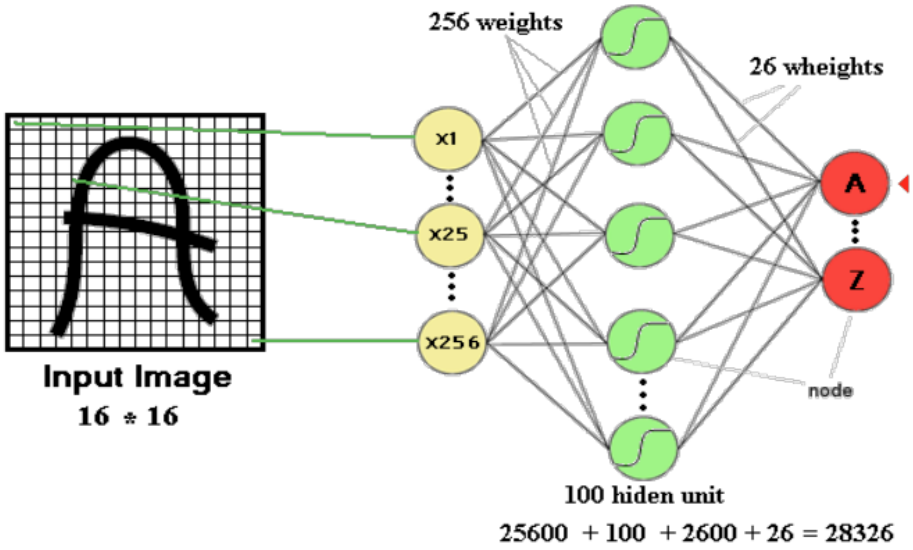


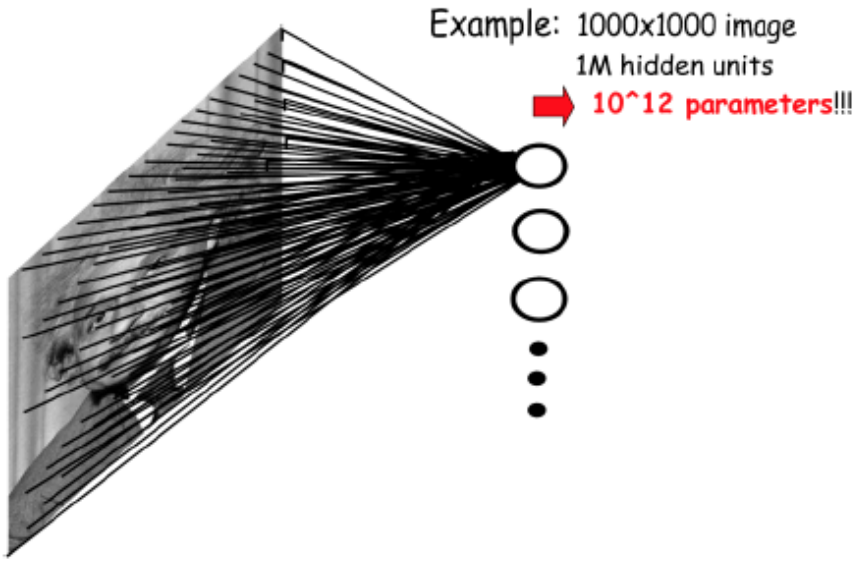
Input image encoding:

- Color (RBG) ?
- Grayscale $L = \frac{R+B+G}{3}$?

Fully Connected Networks: Limitations

- Scalability issue with Fully Connected Networks (MLP)

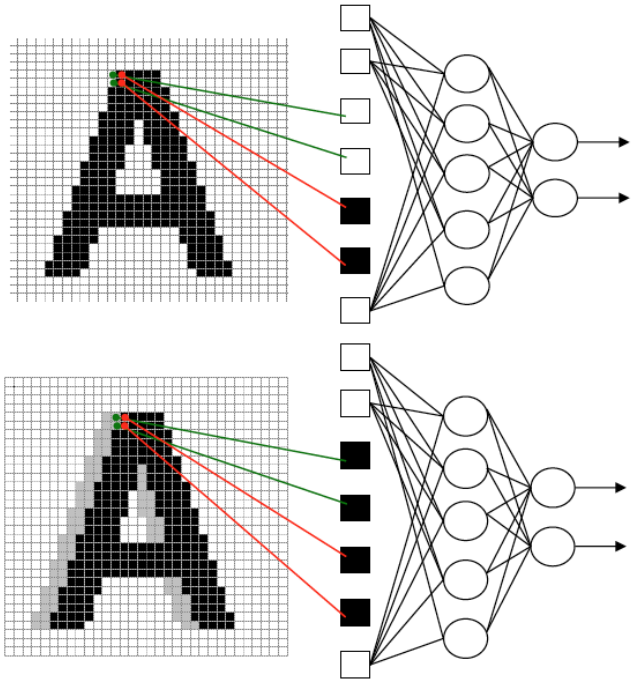


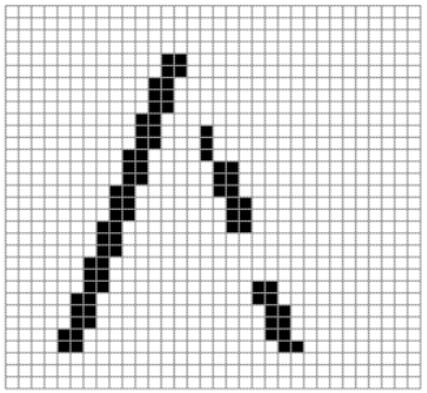


$\Rightarrow$ # Parameter explosion even for a single hidden layer !

Fully Connected Networks: Limitations

- Invariance and robustness to deformation (stability)
- What we expect:
  - Small deformation in input space ⇒ similar representations
  - Large transfo in input space ⇒ very dissimilar representations
- Example (image): impact of a 2 pixel shift





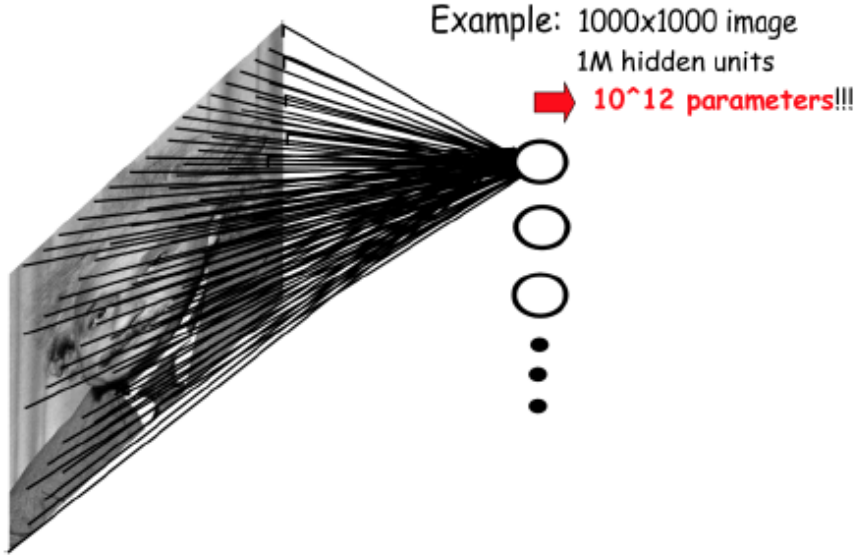
154 input change
from 2 shift left
77 : black to white
77 : white to black

@LeCun

Fully Connected Networks: Limitations

Conclusion of MLP on raw data

- Brute force connection of images as input of MLP NOT a good idea
  - No Invariance/Robustness of the representation because topology of the input data completely ignored
    $\Rightarrow$ *e.g.* indifferent to permutations of input pixel
  - Nb of weights grows largely with the size of the input image

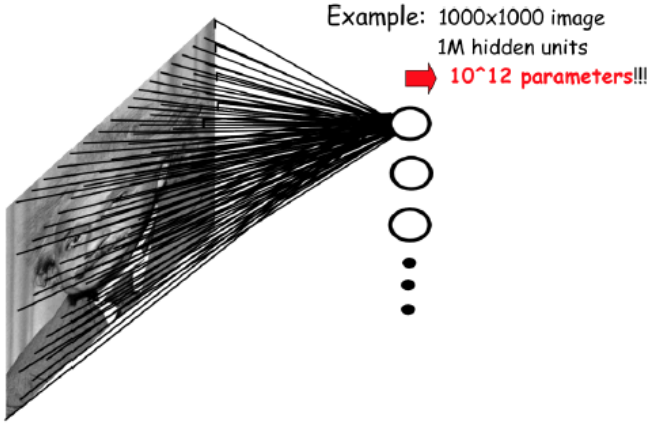


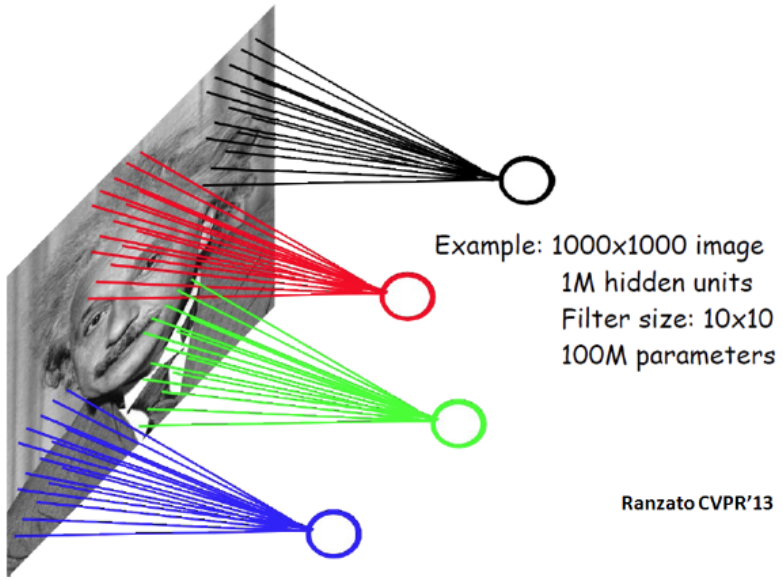
$\Rightarrow$ **How keep spatial topology?**
$\Rightarrow$ **How to limit the number of parameters?**

Taking advantage of structure: Convolution

How to limit the number of parameters?

1. Sparse connectivity: hidden unit only connected to a local patch
   - Weights connected to the patch: **filter** or **kernel**
   - Inspired by biological systems: cell only sensitive to a small sub-region of the input space (receptive field). Many cells tiled to cover the entire visual field

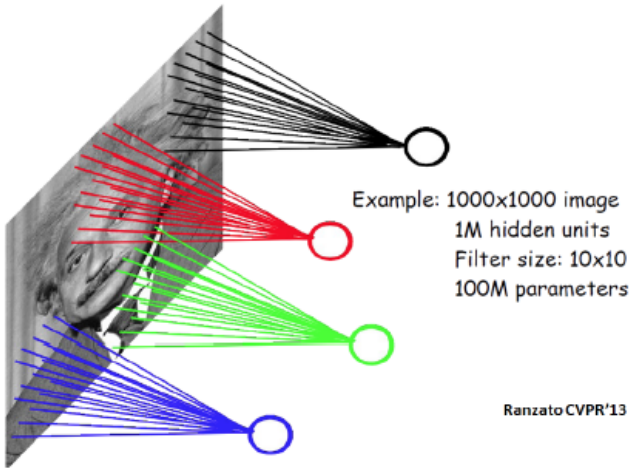


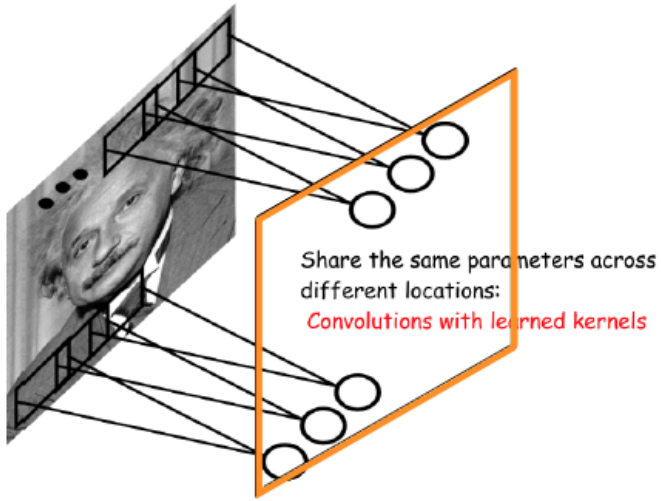


Taking advantage of structure: Convolution

How to limit the number of parameters?

2. Shared Weights
   - Hidden nodes at different locations share the same weights
     - Substantially reduces the number of parameters to learn
   - Keep spatial information in a 2D feature map (hidden layer map)





⇒ Computing responses at hidden nodes equivalent to convolving input image with a linear filter (learned)

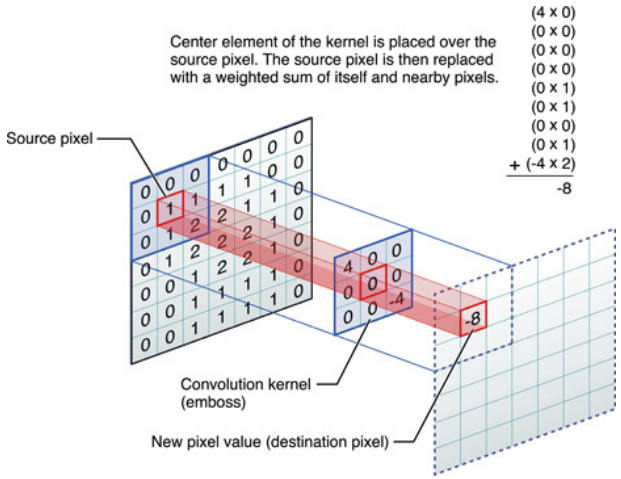
⇒ A learned filter as a feature detector

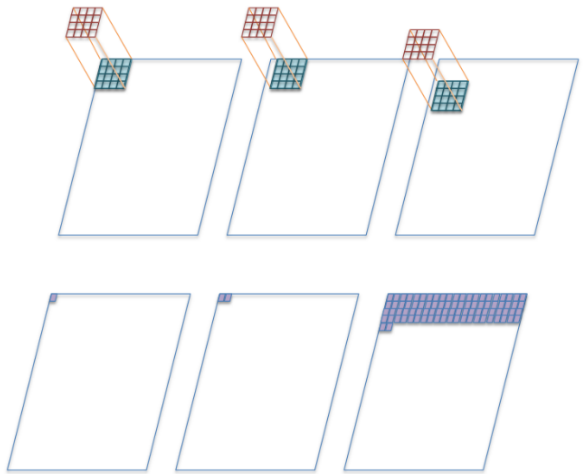
Convolution: Scalar Images

- 2D convolution with a Finite Impulse Response (FIR) h of size d (odd):

$$f'(i,j) = (f \star h)(i,j) = \sum_{n=-\frac{d-1}{2}}^{\frac{d-1}{2}} \sum_{m=-\frac{d-1}{2}}^{\frac{d-1}{2}} f(i-n, m-j)h(n,m)$$

- Simply centering filer h in pixel (x,y) $\Rightarrow$ weighted sum

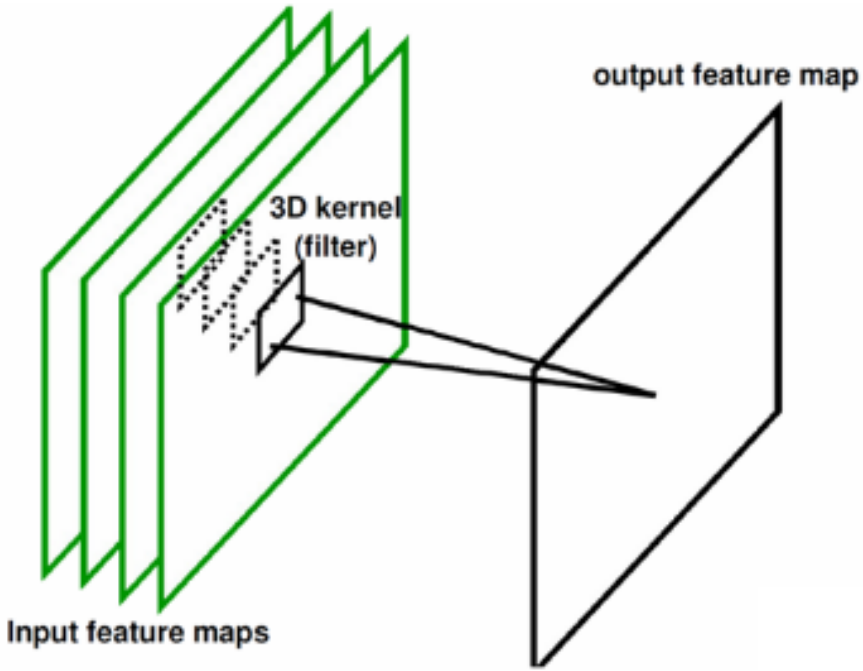


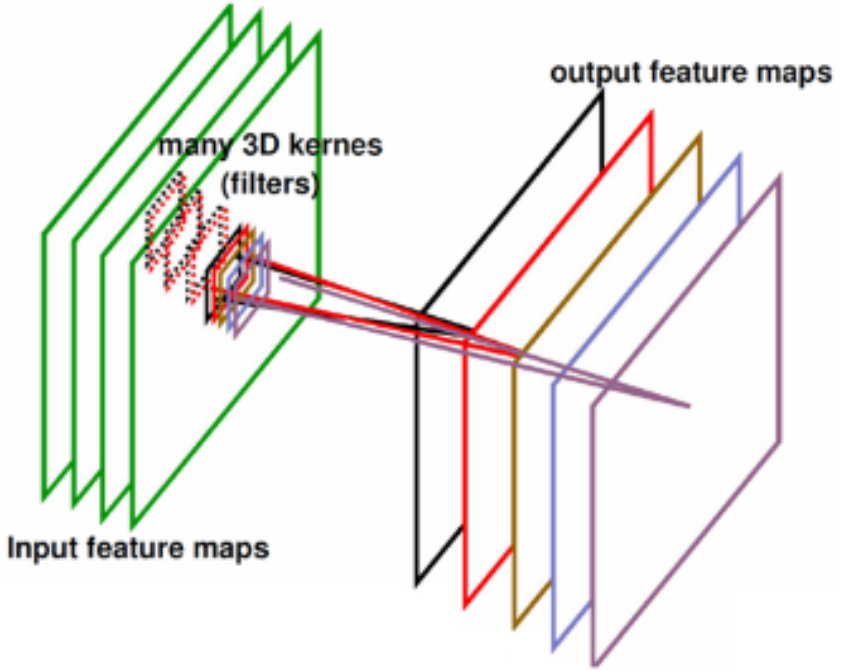


- Output for 1 filer (resp. K filters): 1 2D map (resp. K 2D maps)

Convolution: Vectorial Images (depth M)

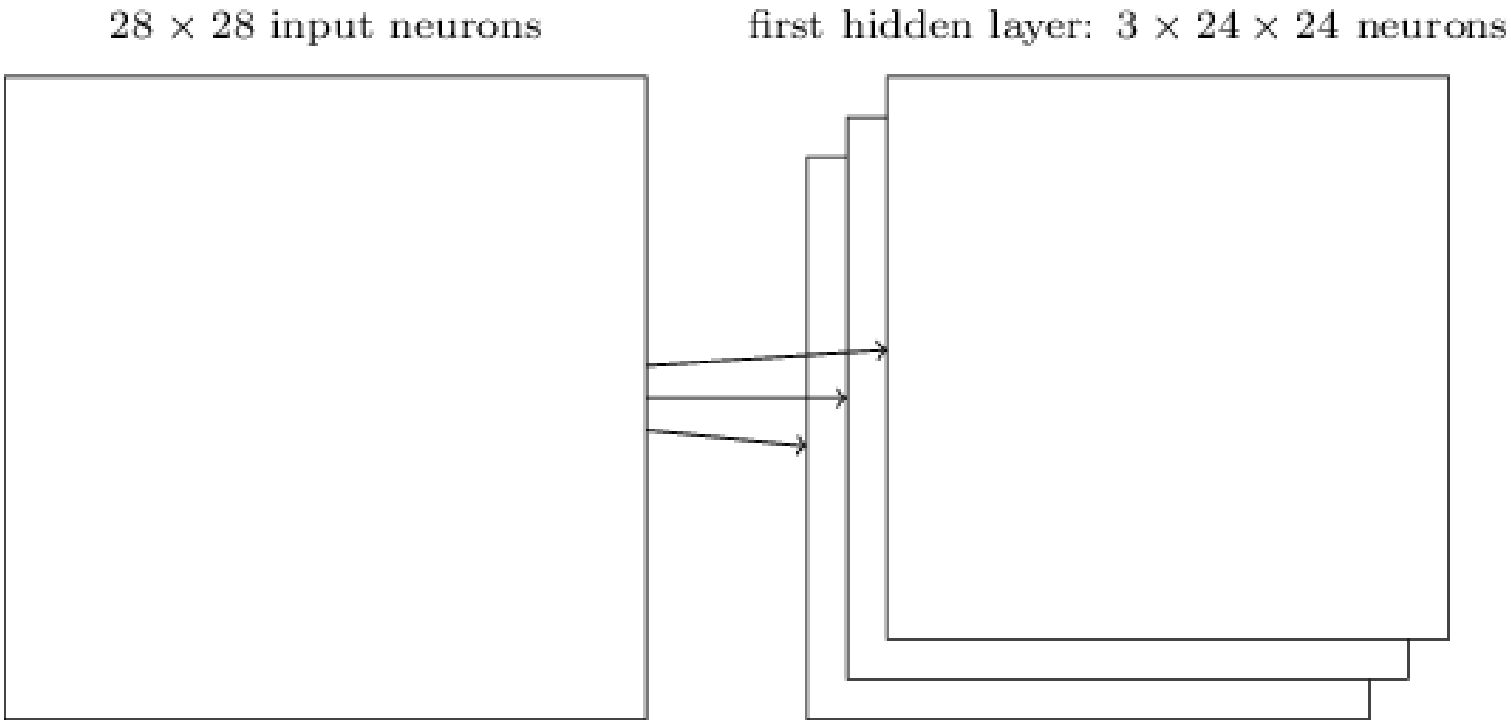
- Each filter has size *dxdxM*
- Example with $M = 3$, *e.g.* color images:





Convolutionnal Layers

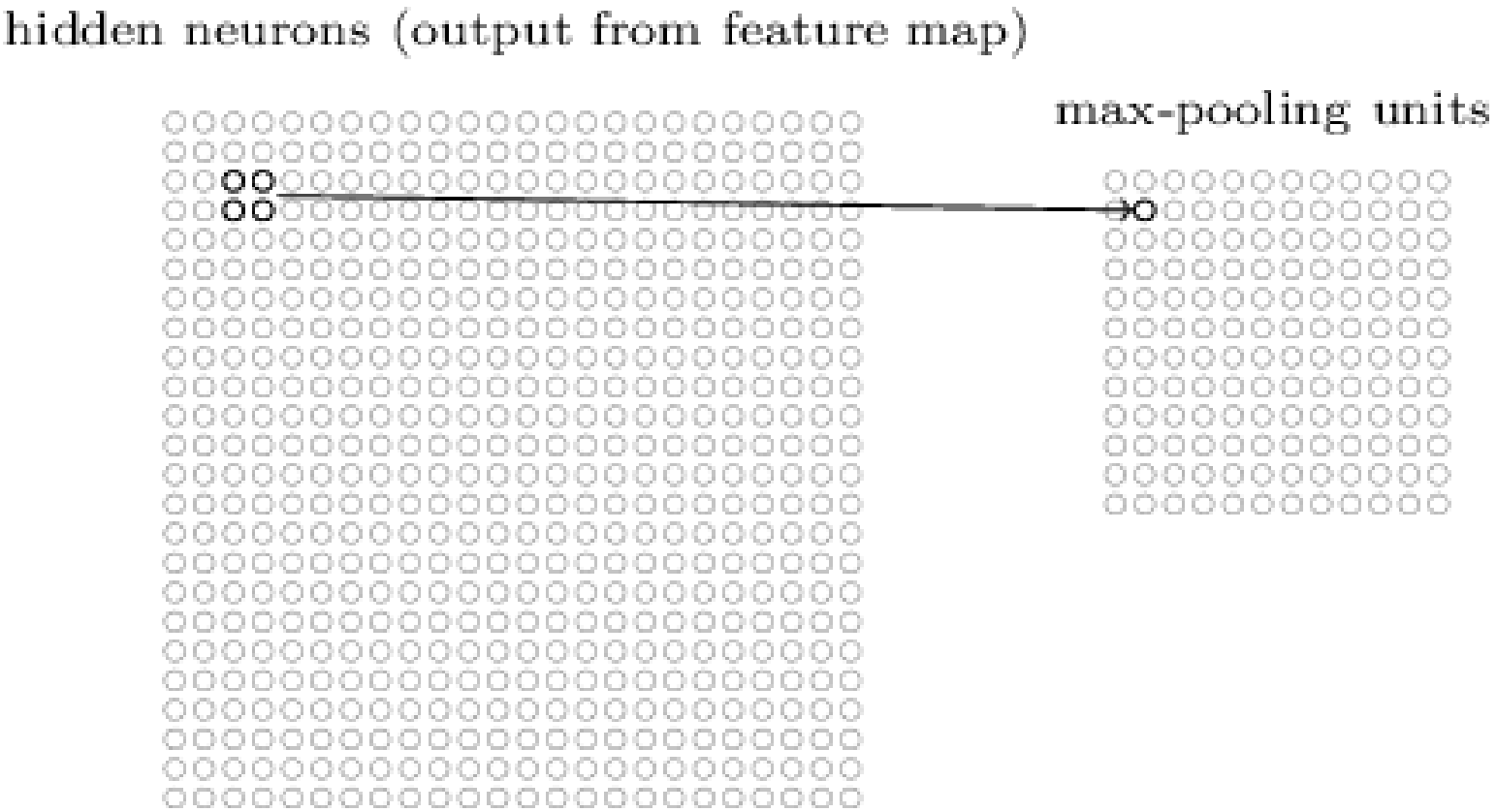
- Convolution layer ⇐ local feature from previous layers
- Feature maps are equivariant to translation
- Followed by non-linearity (activation function)



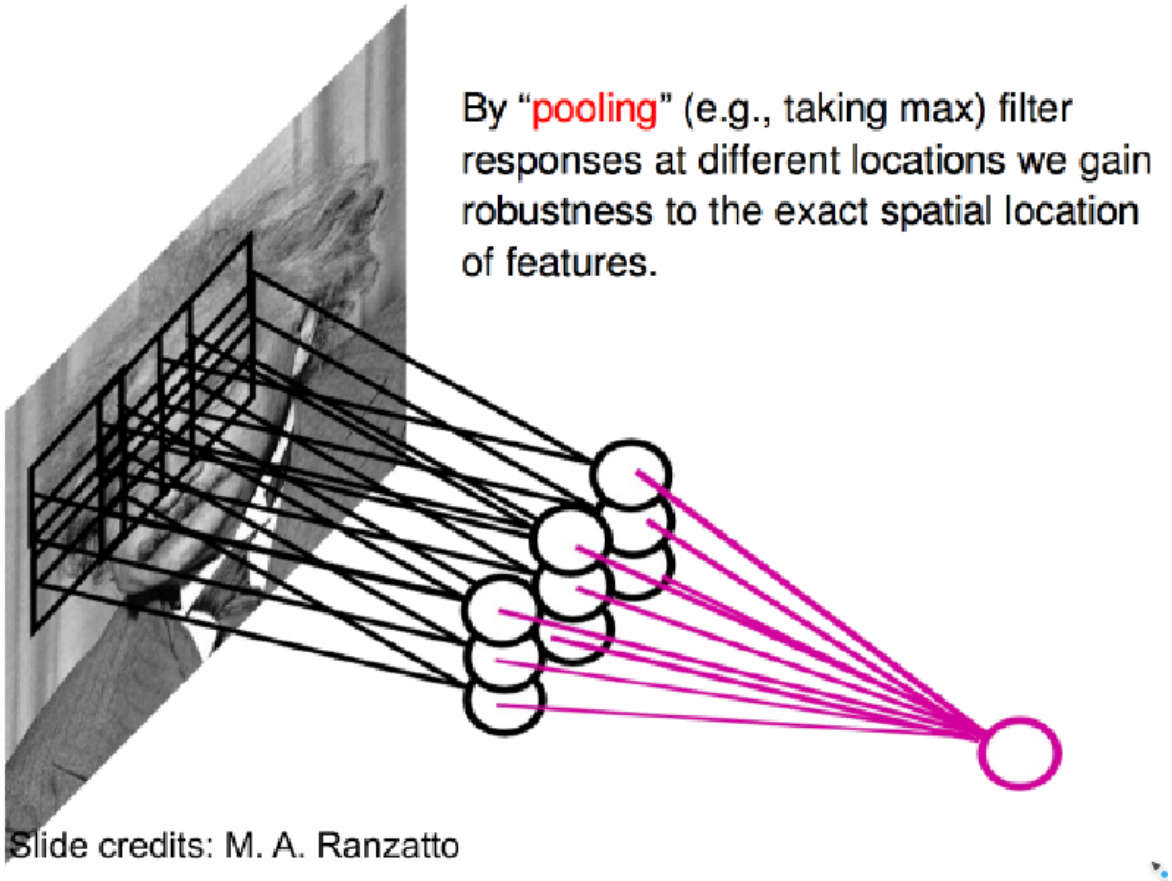
⇒ **How to gain (local) shift invariance ?**

Pooling Layers

- Spatial agregation for each layer
- If stride $s > 1$, sptial resolution decreases (subsampling) $\Rightarrow$ gaining invariace to local translations



Pooling Layers



Pooling Layers: Examples

Max-pooling:

$$h_j^n(x,y) = max_{\bar{x}\in N(x),\,\bar{y}\in N(y)} h_j^{n-1}(\bar{x},\bar{y})$$

Average-pooling:

$$h_j^n(x,y) = 1/K \sum_{\bar{x}\in N(x),\,\bar{y}\in N(y)} h_j^{n-1}(\bar{x},\bar{y})$$

L2-pooling:

$$h_j^n(x,y) = \sqrt{\sum_{\bar{x}\in N(x),\,\bar{y}\in N(y)} h_j^{n-1}(\bar{x},\bar{y})^2}$$

L2-pooling over features:

$$h_j^n(x,y) = \sqrt{\sum_{k\in N(j)} h_k^{n-1}(x,y)^2}$$

Slide credits: M. A. Ranzatto

Convolutional Neural Networks (ConvNets)

- An elementary block: Convolution + Non linearity + pooling
- Stack several blocks: Convolutional Neural Networks (ConvNets)

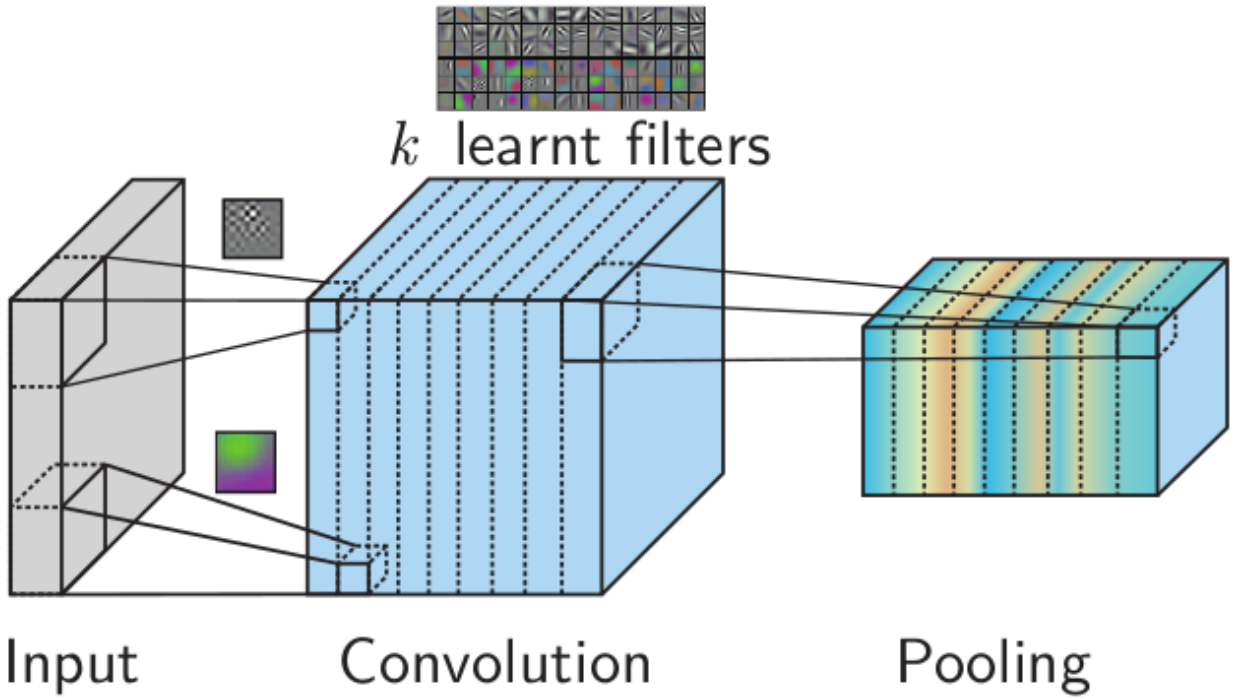


Figure: Important building blocks in CNN

Convolutional Neural Networks (ConvNets)

- Generally, Feature maps stacked together at one point ⇒ fully connected layers

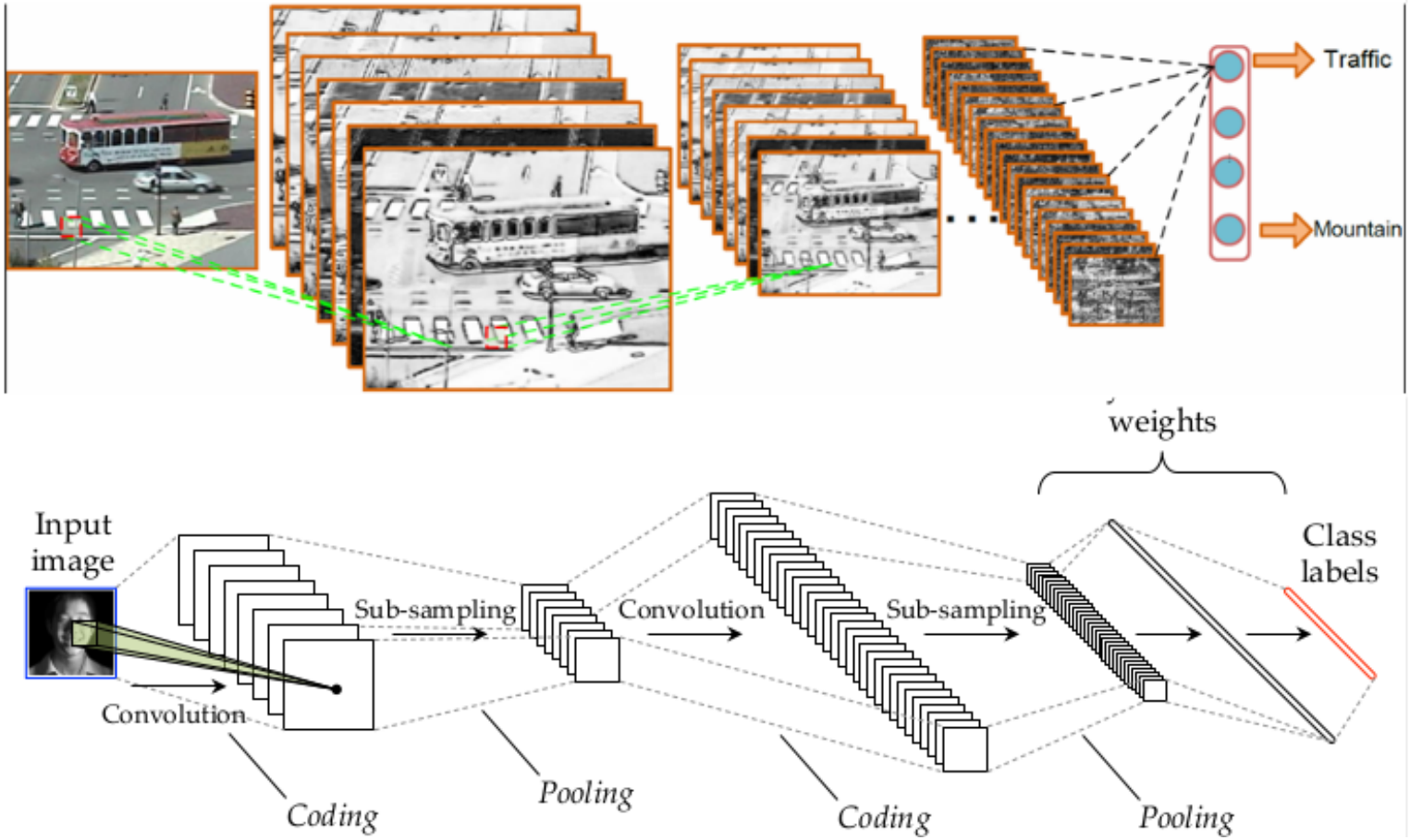
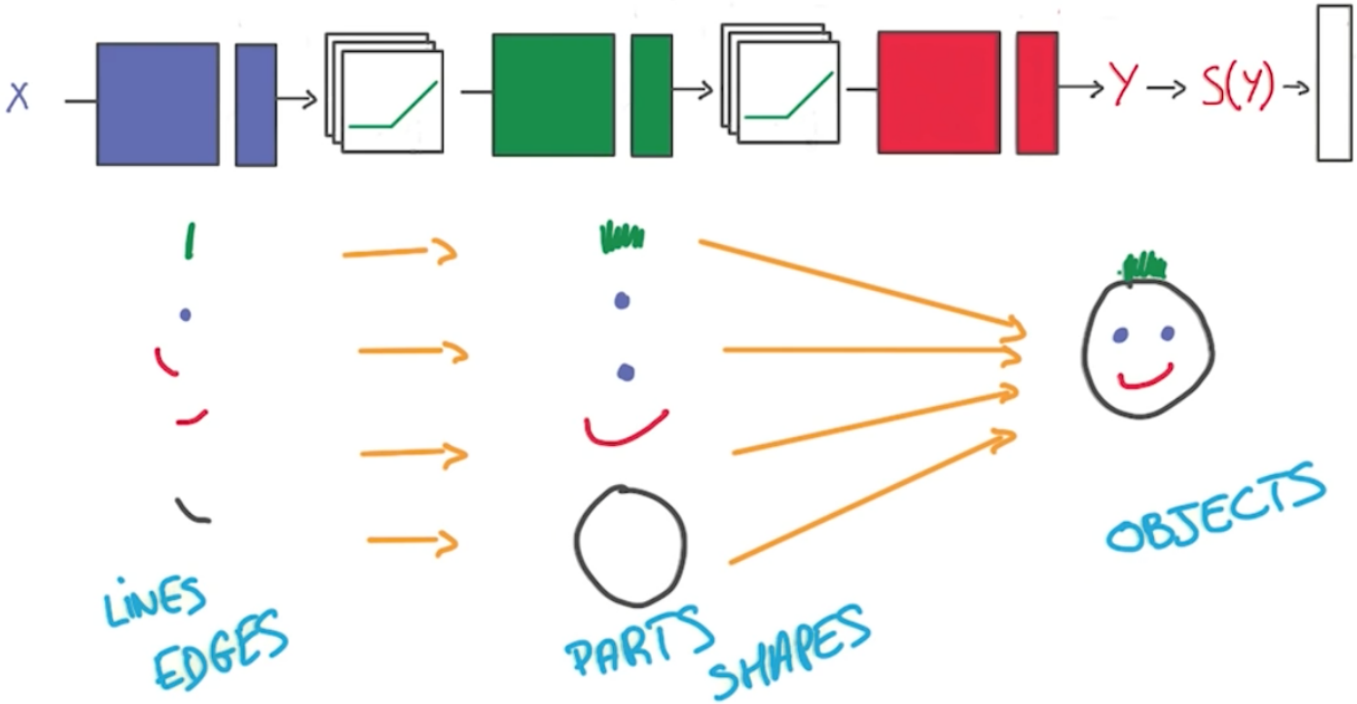


Figure: Important building blocks in CNN

ConvNets: Conclusion

- Crucial step for tacking advantage of structure ⇒ local processing
- Useful for many data types and applications:
  - Low-level signal, *e.g.* image, audio (speech, music)
  - More semantic data, *e.g.* modern text embedding (`word2vec`) or RNN
- Block [Convolution + Non linearity + pooling] intuitive for modeling hierarchical information extraction

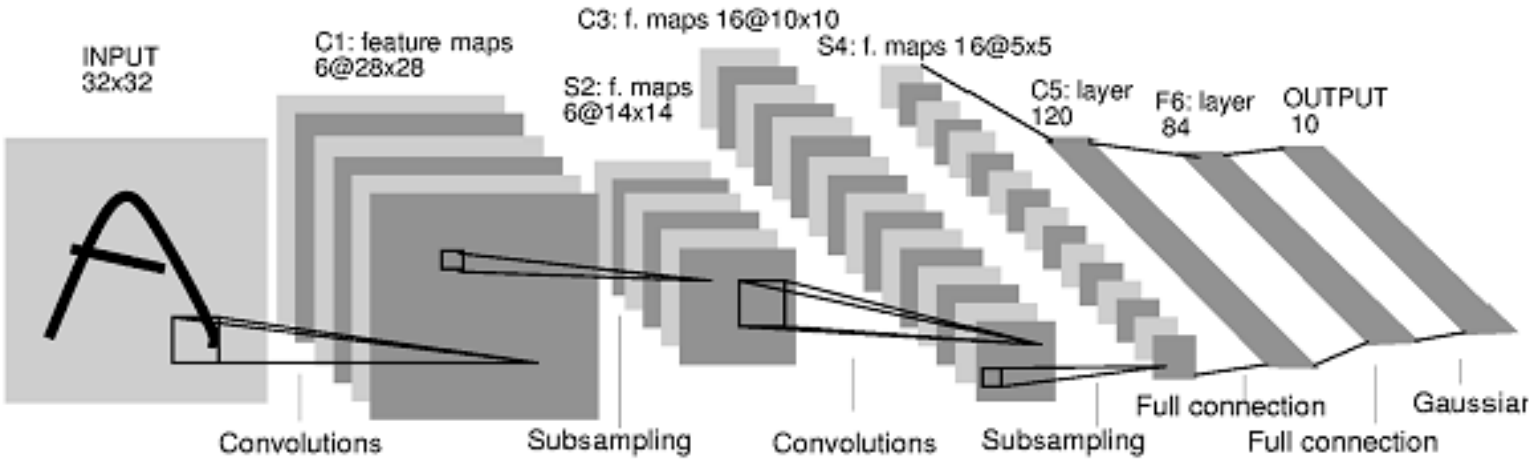


Outline

Deep Learning: Trends and methods in the last four decades

80's: 1st Convolutionnal Neural Networks

- LeNet 5 Model [LBD$^+$89], trained using back-prop

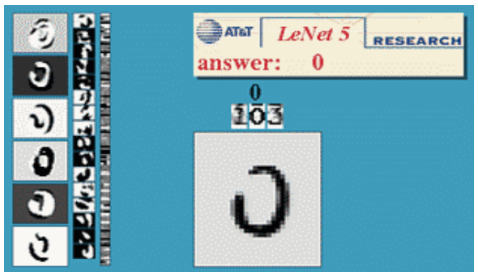


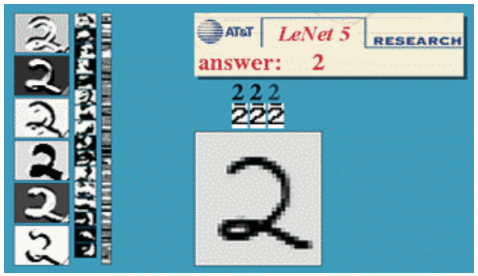
- Input: 32x32 pixel image. Largest character is 20x20
- 2 successive blocks [Convolution + Sigmoid + Pooling (+sigmoid)]
  Cx: Convolutional layer, Sx: Subsampling layer
- C5: convolution layer ~ fully connected
- 2 Fully connected layers Fx

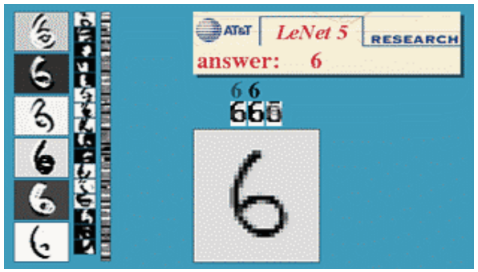
80's: LeNet 5 Model

- Evaluation on MNIST
- Total # parameters ~ 60000
  - 60,000 original datasets: test error: 0.95%
  - 540,000 artificial distortions + 60,000 original: Test error: 0.8%

- **Successful deployment for postal code reading in the US**



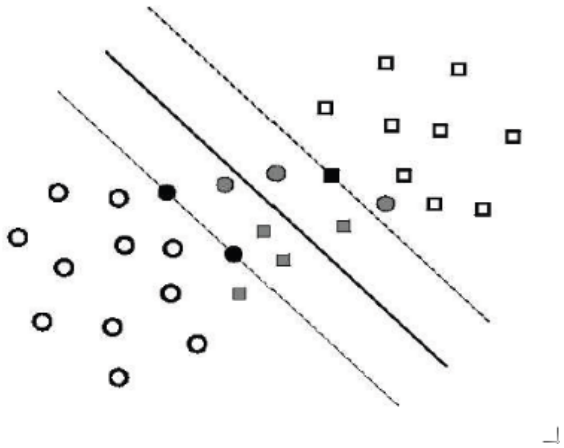


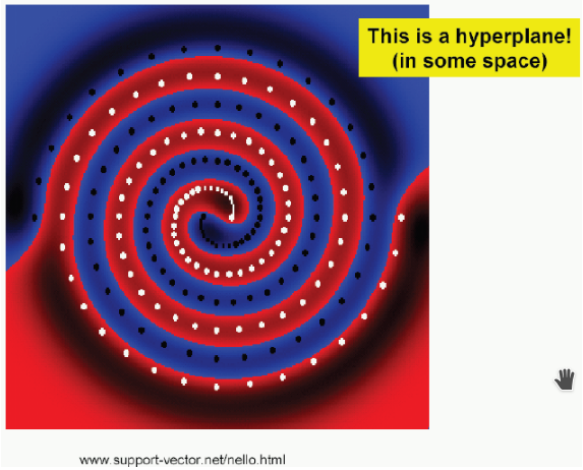


Deep Learning: Trends and methods in the last four decades

90's: start of winter for deep learning

- Deep neural nets =' black magic', black boxes
  - Lack of interpretability
  - Optimization issues for highly non-convex objective function
- **Golden age of kernel methods**
  - Generalization theory with Support Vector Machines
  - Extension to non-linear modes: kernel trick
    - Kernel encode prior knowledge (structure) on data
  - Convex optimization problem

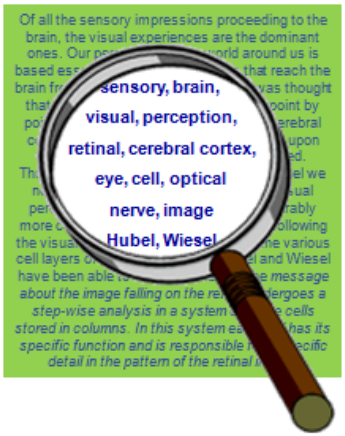


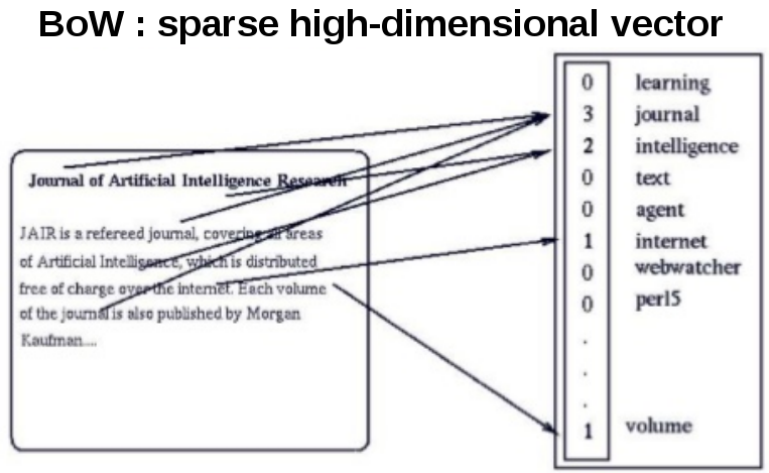


Deep Learning: Trends and methods in the last four decades

2000's: Bag of Words Model (BoW)

- Started from the Information Retrieval (IR) community
- Text classification : document as a histogram of word occurrences



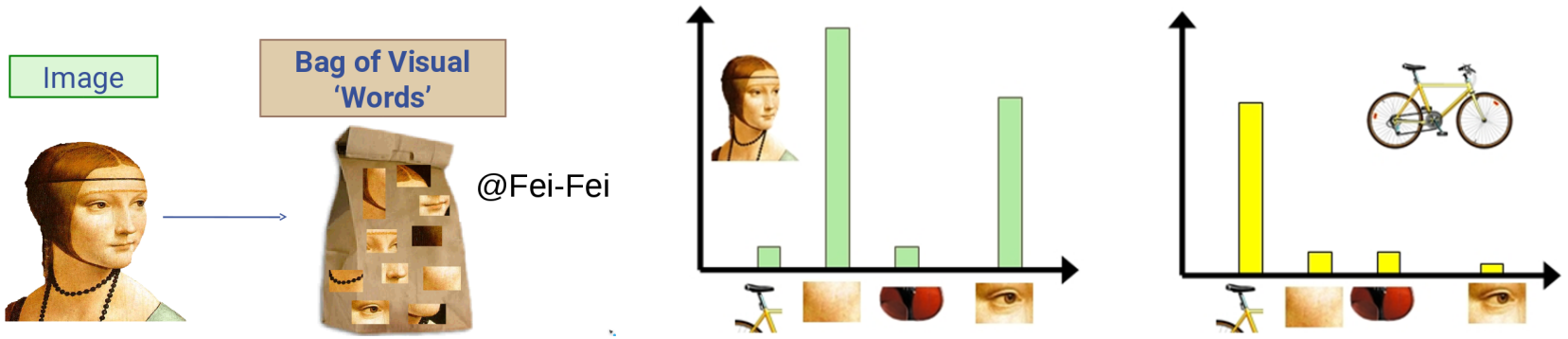


- Bow representation as input for powerful classifiers, *e.g.* SVM

Deep Learning: Trends and methods in the last four decades

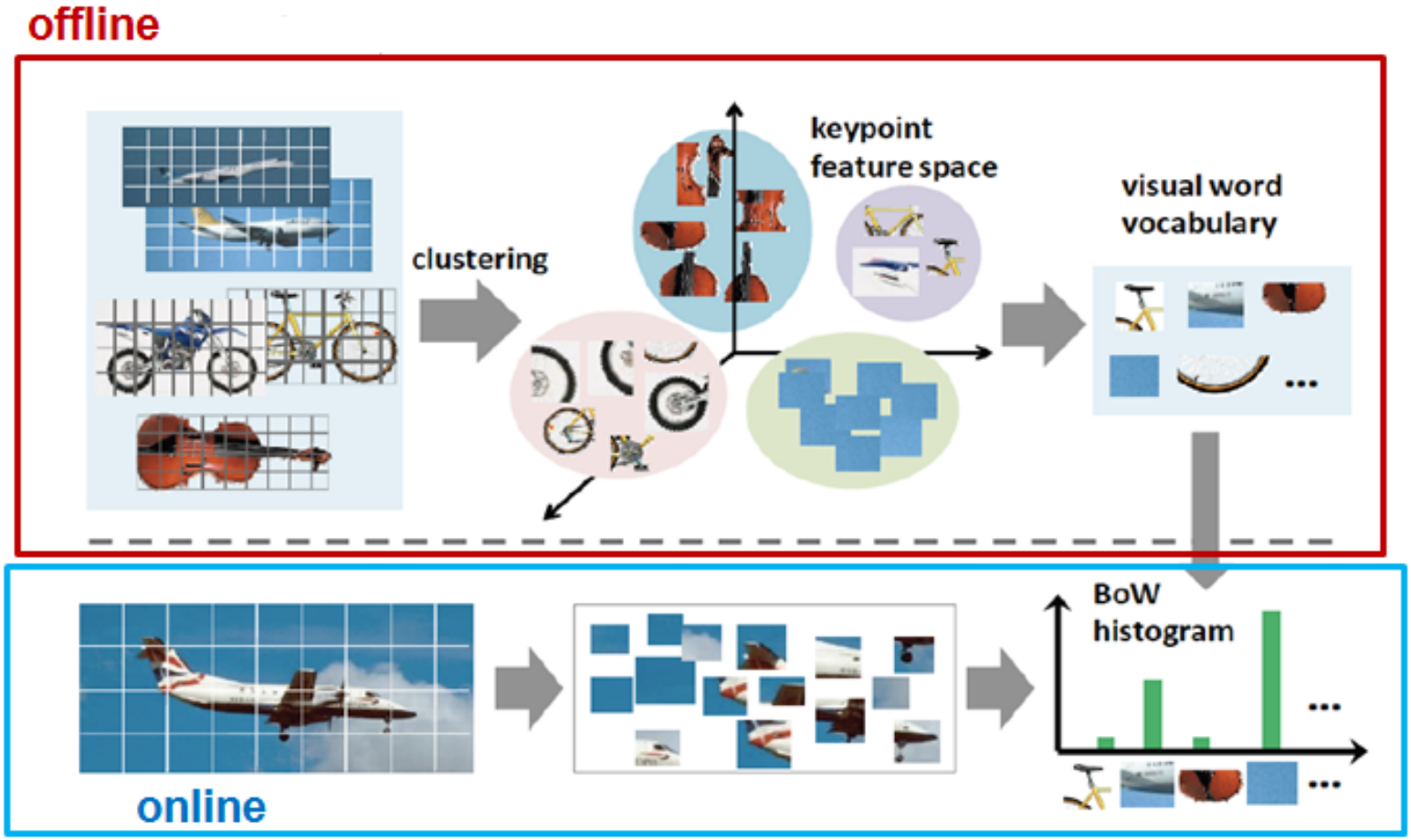
2000's: Bag of Words Model

- Adapting the BoW model for visual recognition ?
  ⇒ Bag of Visual Word (BoV)
- Main challenge: definition of visual words unclear!



- Solution: compute a dictionary on local image regions (clustering)
  - Local regions represented by handcrafted descriptors, *e.g.* SIFT

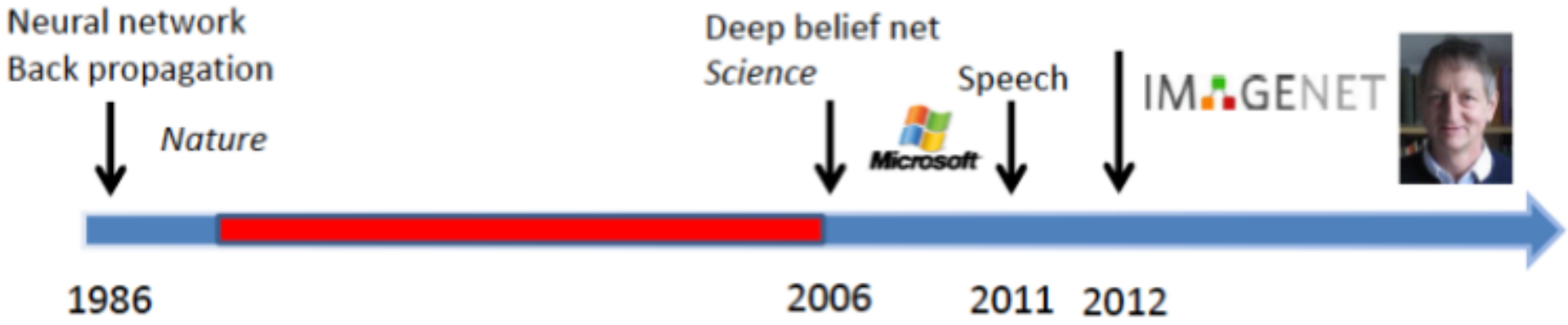
2000's: Bag of Visual Words Model



- 2000's: BoW + SVM state-of-the-art
- Many works on kernel on BoW, coding & pooling → 2012

Deep Learning: Trends and methods in the last four decades

Deep Learning renewal since 2006

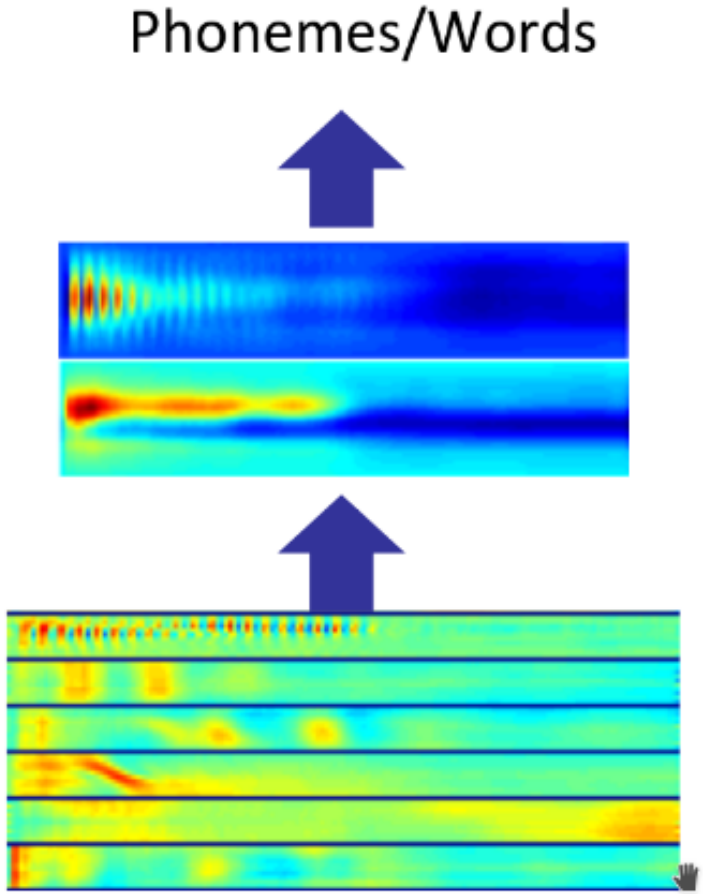


- 2006: new unsupervised learning for Deep Belief Nets (DBN) [HOT06]
- Theoretical results for improving model quality with depth
- Unsupervised training used as init for supervised learning with back-prop

Deep Learning and ConvNet for Speech Recognition

- First DL breakthrough on large datasets: speech recognition
- Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition, Dahl et al. (2010)

Acoustic model	Recog \ WER	RT03S FSH	Hub5 SWB
Traditional features	1-pass –adapt	**27.4**	**23.6**
Deep Learning	1-pass –adapt	**18.5** (–33%)	**16.1** (–32%)



@Socher

Deep Learning and ConvNet for Image Classification

- ImageNet ILSVRC Challenge (Stanford):
  - $1,200,000$ training images, $1,000$ classes, mono-label
  - Based on WordNet hierarchy (ontology)
  - Evaluation: top-5 error
- Up to 2012, leading approaches: BoW + SVM
- ILSVRC'12: the deep revolution $\Rightarrow$ outstanding success of ConvNets [KSH12]

Rank	Name	Error rate	Description
1	**U. Toronto**	0.15315	Deep learning
2	U. Tokyo	0.26172	Hand-crafted features and learning models. Bottleneck.
3	U. Oxford	0.26979	
4	Xerox/INRIA	0.27058	

2012: the deep revolution

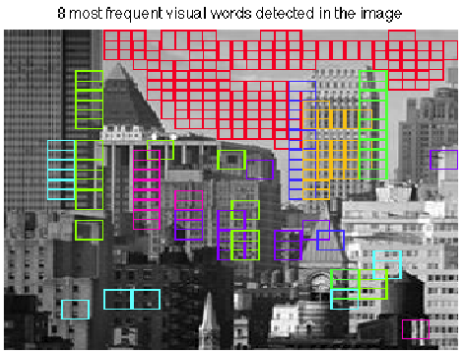
Deep ConvNet success at ILSVRC'12

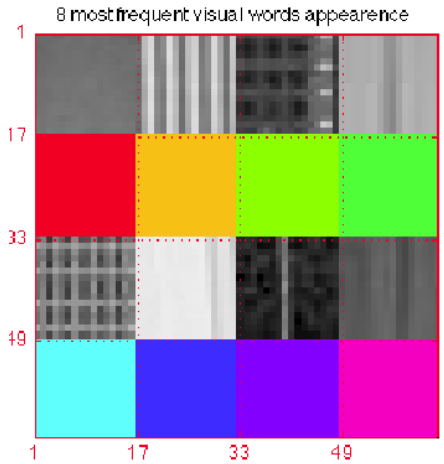
Two main practical reasons:

1. Huge number of labeled images (10^6 images)
   - Possible to train very large models without over-fitting
   - Larger models enables to learn rich (semantic) features hierarchies
2. GPU implementation for training
   - Relatively cheap and fast GPU
   - Training time reduced to 1-2 weeks (up to 50x speed up)

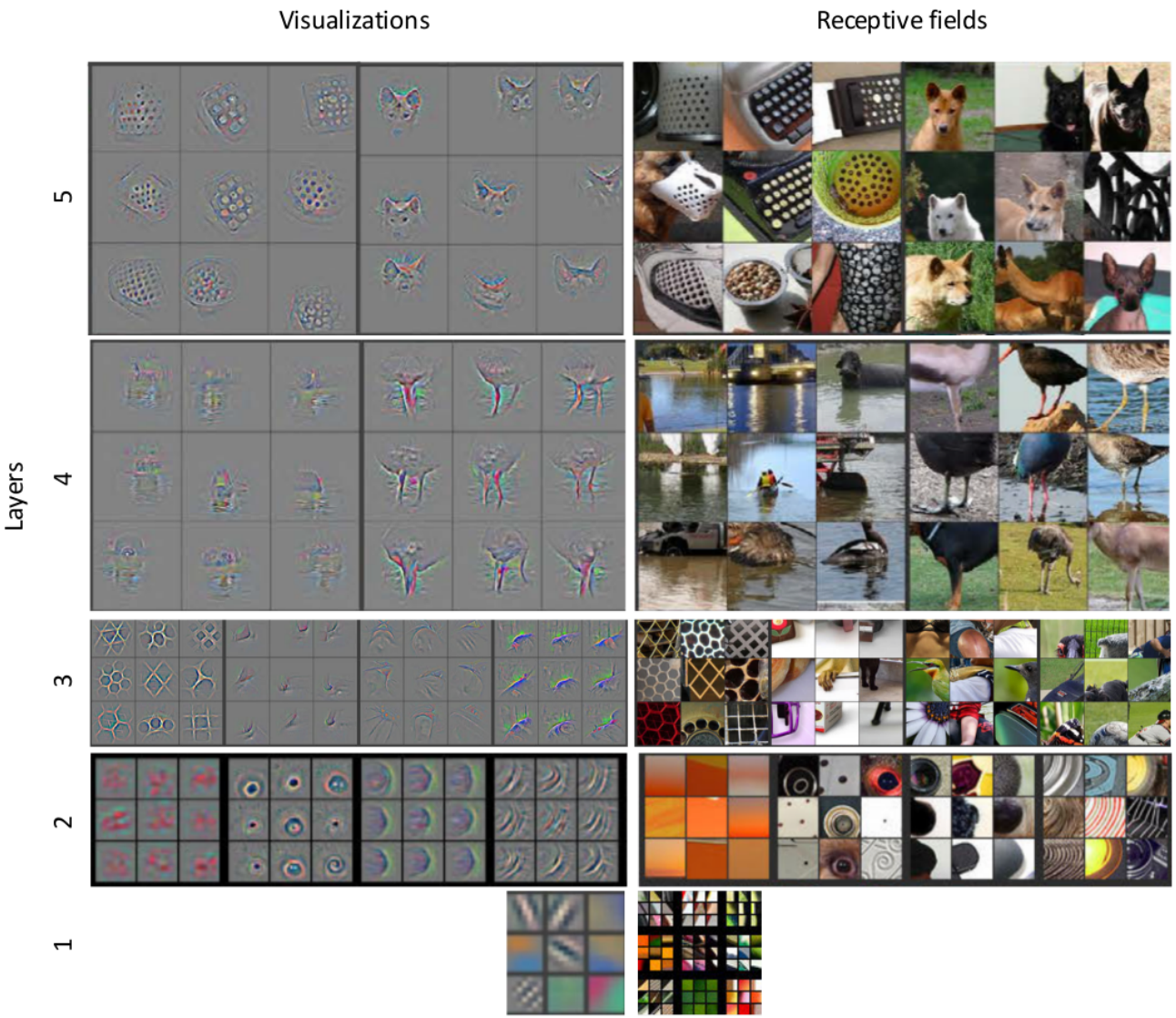
IMAGENET

Deep Learning in 2012: Representation Learning



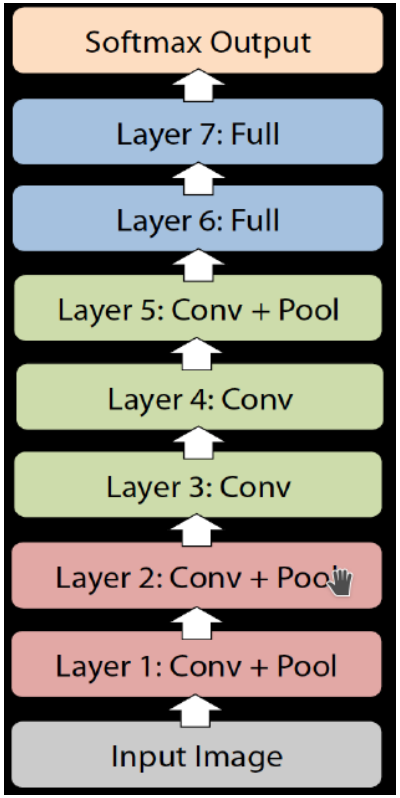


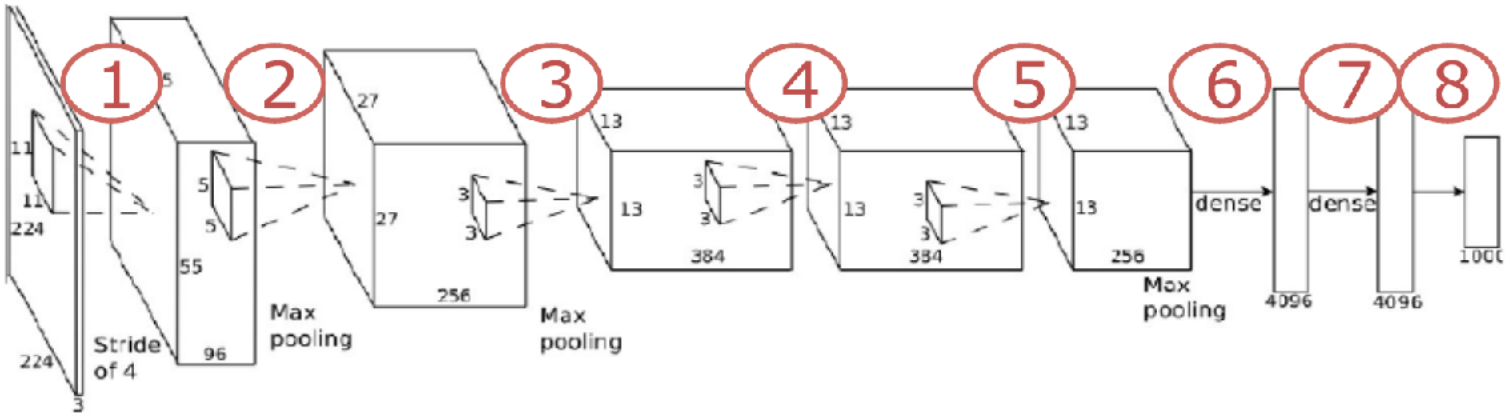
Deep: more semantic features



AlexNet [KSH12] in ILSVRC'12

- 60,000,000 parameters
- 650,000 neurons - 630,000,000 connections
- 5 convolutional layers, 3 Fully Connected (FC)
  - Convolution layer: Convolution + non linearity (ReLU) + pooling
  - Full= FC + non linearity - Final FC: 4096-dim
- Trained on 2 GPUs for a week





AlexNet [KSH12] in ILSVRC'12

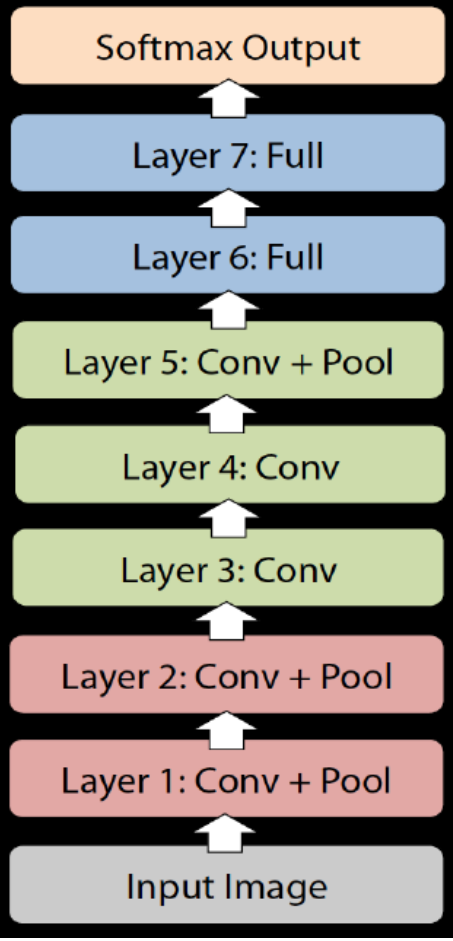
First Convolutionnal Layer

- Input: Images: 227x227x3
- Filter (receptive field) size F: 11, S (stride) = 4
- 96 filters ⇒ output size 55*55*96 = 290,400 neurons
- Each Filter has 11*11*3 = 363 weights + 1 bias = 364 params
  - N.B.: Convolution in whole feature map depth (*cf* LeNet 5 discussion)
- # parms: 96 * 364 = 34, 944

AlexNet [KSH12] in ILSVRC'12

Architecture of Krizhevsky et al.

- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR'09]
- 18.2% top-5 error
- Our reimplementation:
  18.1% top-5 error

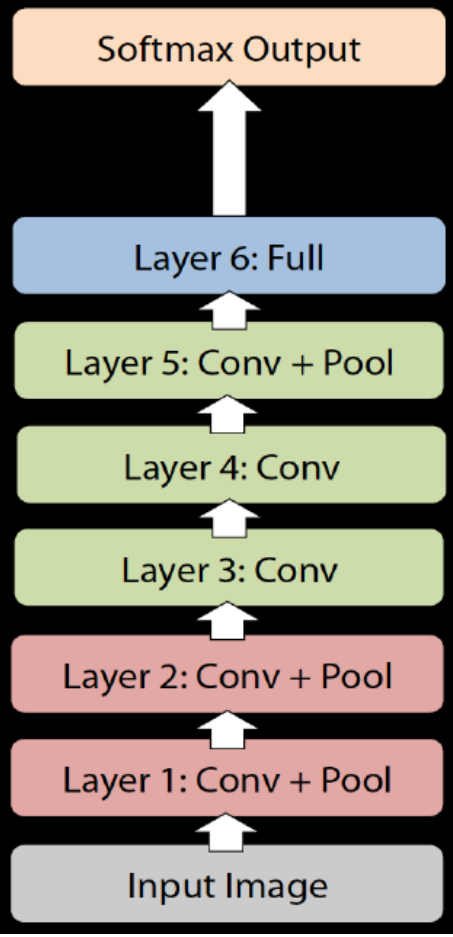


Credit: R. Fergus

AlexNet [KSH12] in ILSVRC'12

Architecture of Krizhevsky et al.

- Remove top fully connected layer
  – Layer 7
- Drop 16 million parameters
- Only 1.1% drop in performance!

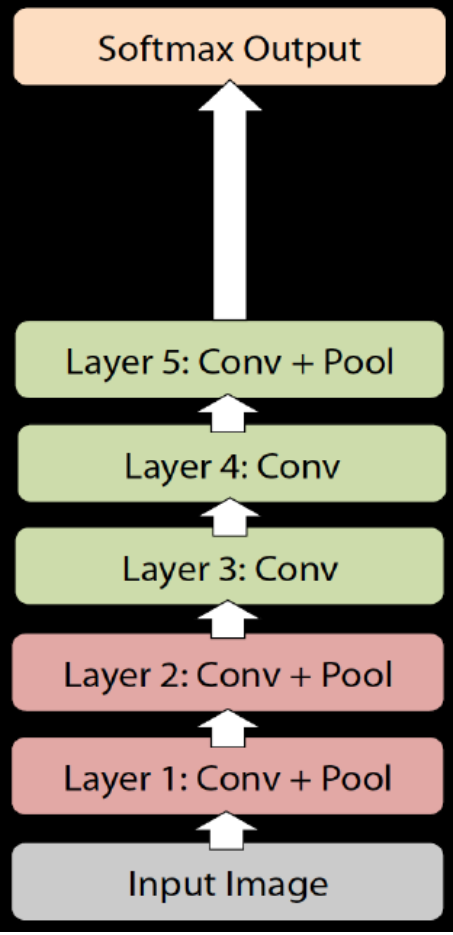


Credit: R. Fergus

AlexNet [KSH12] in ILSVRC'12

Architecture of Krizhevsky et al.

- Remove both fully connected layers
  - Layer 6 & 7
- Drop ~50 million parameters
- 5.7% drop in performance



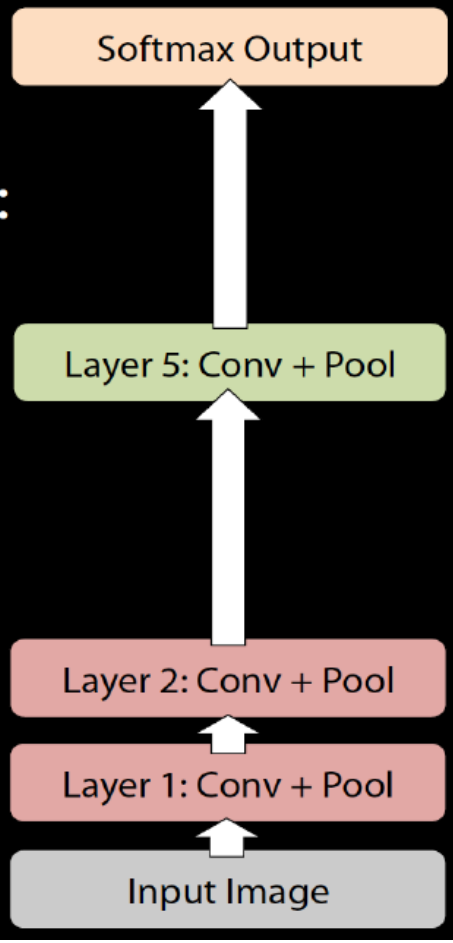
Credit: R. Fergus

AlexNet [KSH12] in ILSVRC'12

Architecture of Krizhevsky et al.

- Now try removing upper feature extractor layers & fully connected:
  - Layers 3, 4, 6 ,7
- Now only 4 layers
- 33.5% drop in performance

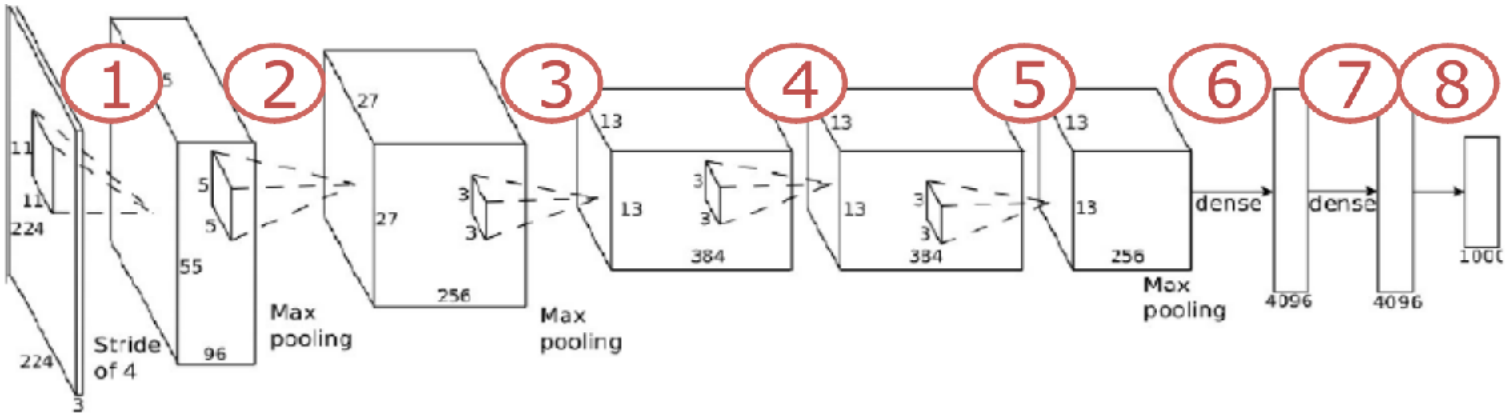
→Depth of network is key



Credit: R. Fergus

AlexNet [KSH12] in ILSVRC'12

- Same global architecture as older nets, *e.g.* LeNet
  - Trained with back-prop and stochastic gradient descent
- But bigger (deeper and wider): 60 10^6 parameters *vs* 60 10^3
  - Needs more data (10^6 *vs* 10^4)
  - GPU implementation for fast training
- Also some architectural and optim improvements (see next course):
  - Non-linearity: ReLU *vs* sigmoid
  - Overlapping pooling (Local Response Normalisation, LRN)
  - Regularization: data augmentation, dropout

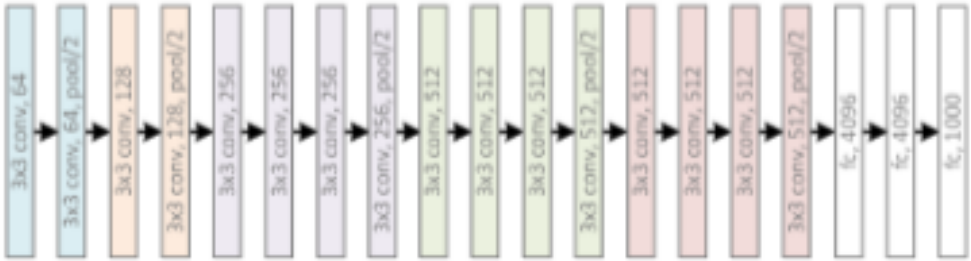


Outline

Deep Learning since 2012

More & more data (Facebook 10^9 images / day), larger & larger networks

VGG, 16/19 layers, 2014



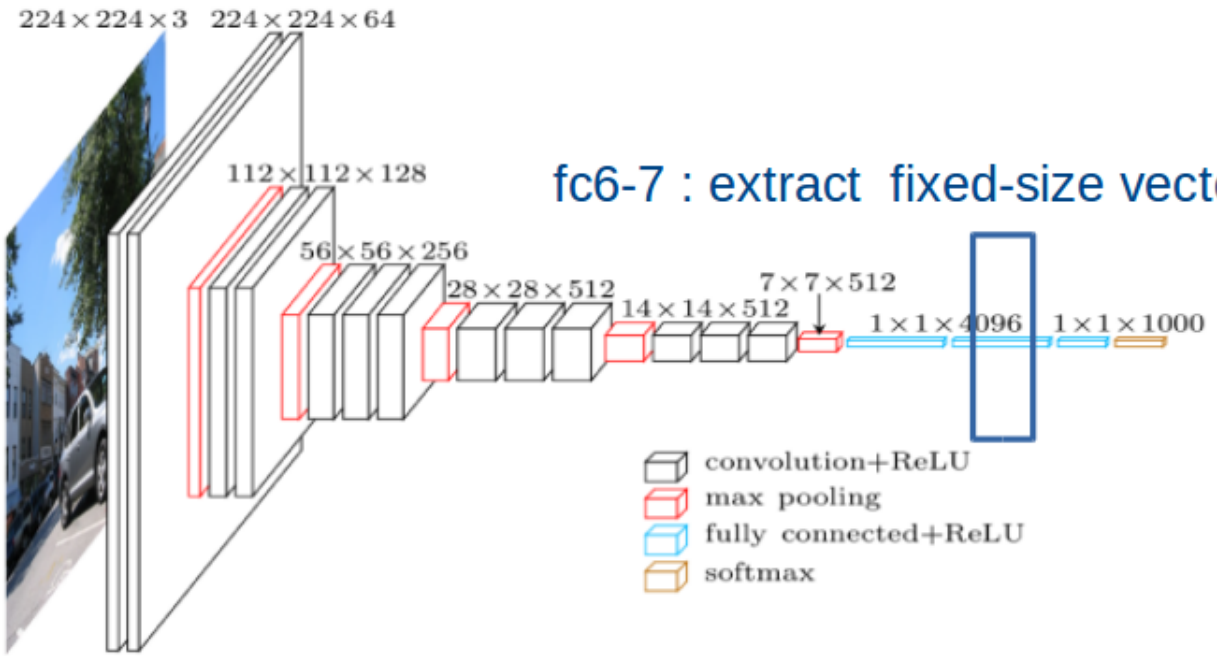
GoogleNet, 22 layers, 2014

ResNet, 152 layers, 2015

Deep Learning since 2012

Transferring Representations learned from ImageNet

- Deep ConvNets require large-scale annotated datasets
  - Huge # parameters ⇒ difficult to train from scratch on "small datasets"
- **BUT:** Extract layer ⇒ fixed-size vector: **"Deep Features" (DF)**



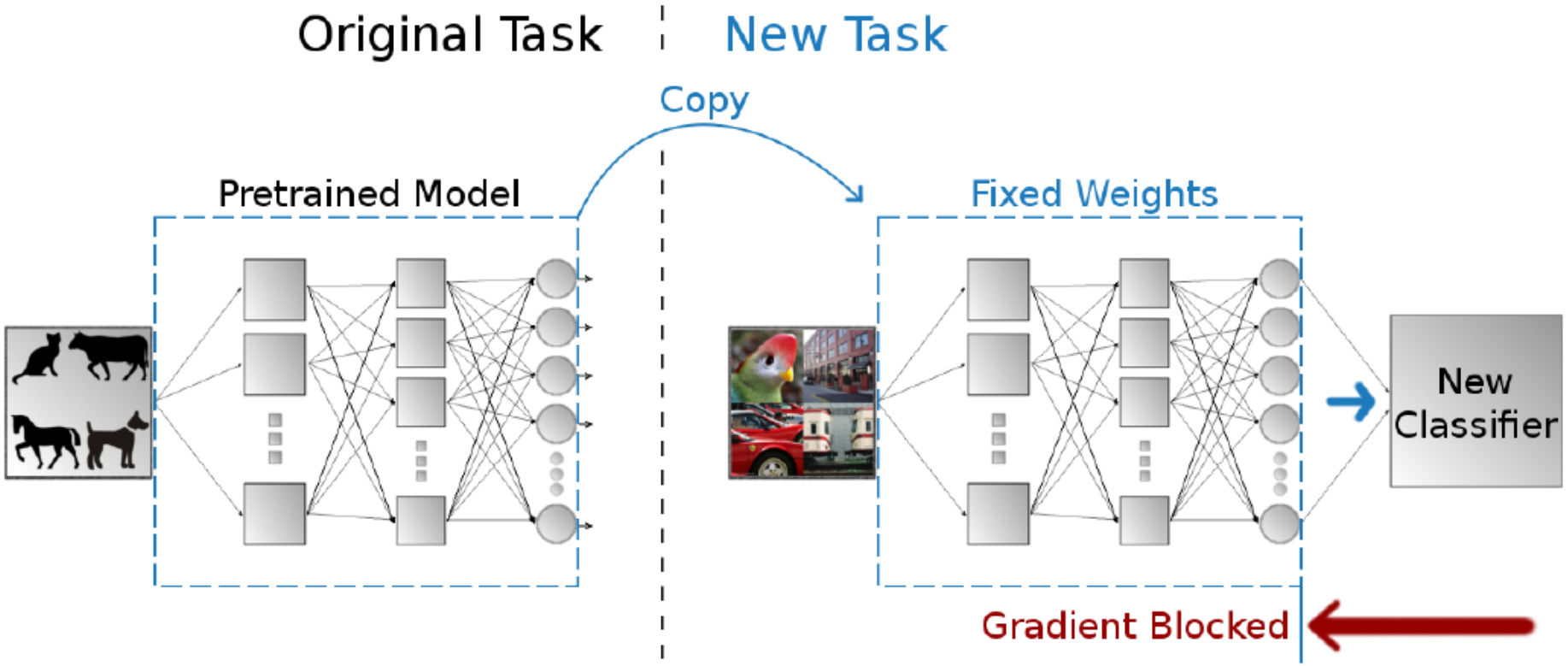
- **Now state-of-the-art for any visual recognition task**

Deep Features (DF) and Domain Adaptation

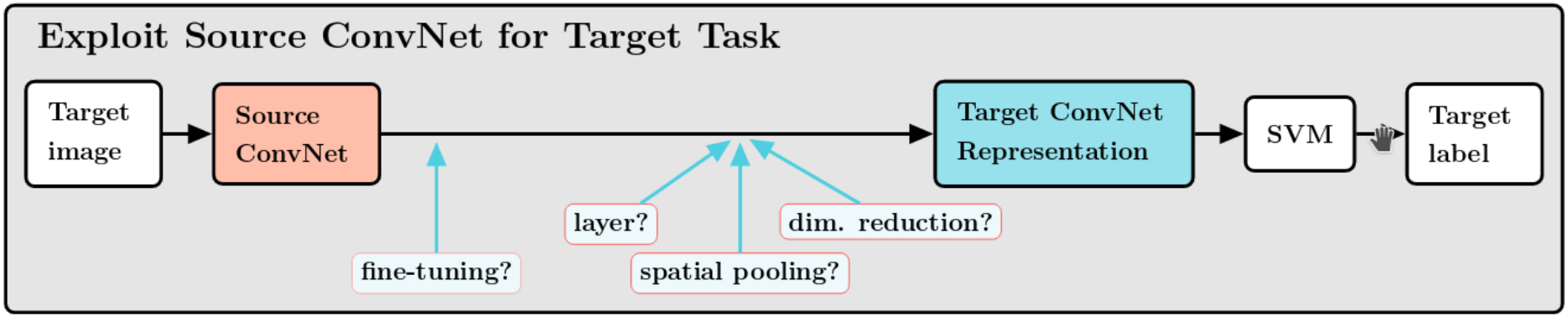
- DF $\Leftarrow$ Deep ConvNet trained on a large-scale dataset (ImageNet, Places, *etc*)
- DF: off-the-shelf features for any visual recognition task
- DF: Generic features, very robust to :
  - Dataset change $\Rightarrow$ apply DF for classification with different images, different classes
  - Domain change $\Rightarrow$ apply DF for other visual tasks, *e.g.* localization, segmentation, pose estimation, retrieval *etc*
- Since 2012: all performance re-benchmarked with DF
- Need to adapt the final layer to match the target domain goal

Deep Features (DF) and Domain Adaptation

DF: off-the-shelf descriptors



Deep Features (DF) and Domain Adaptation

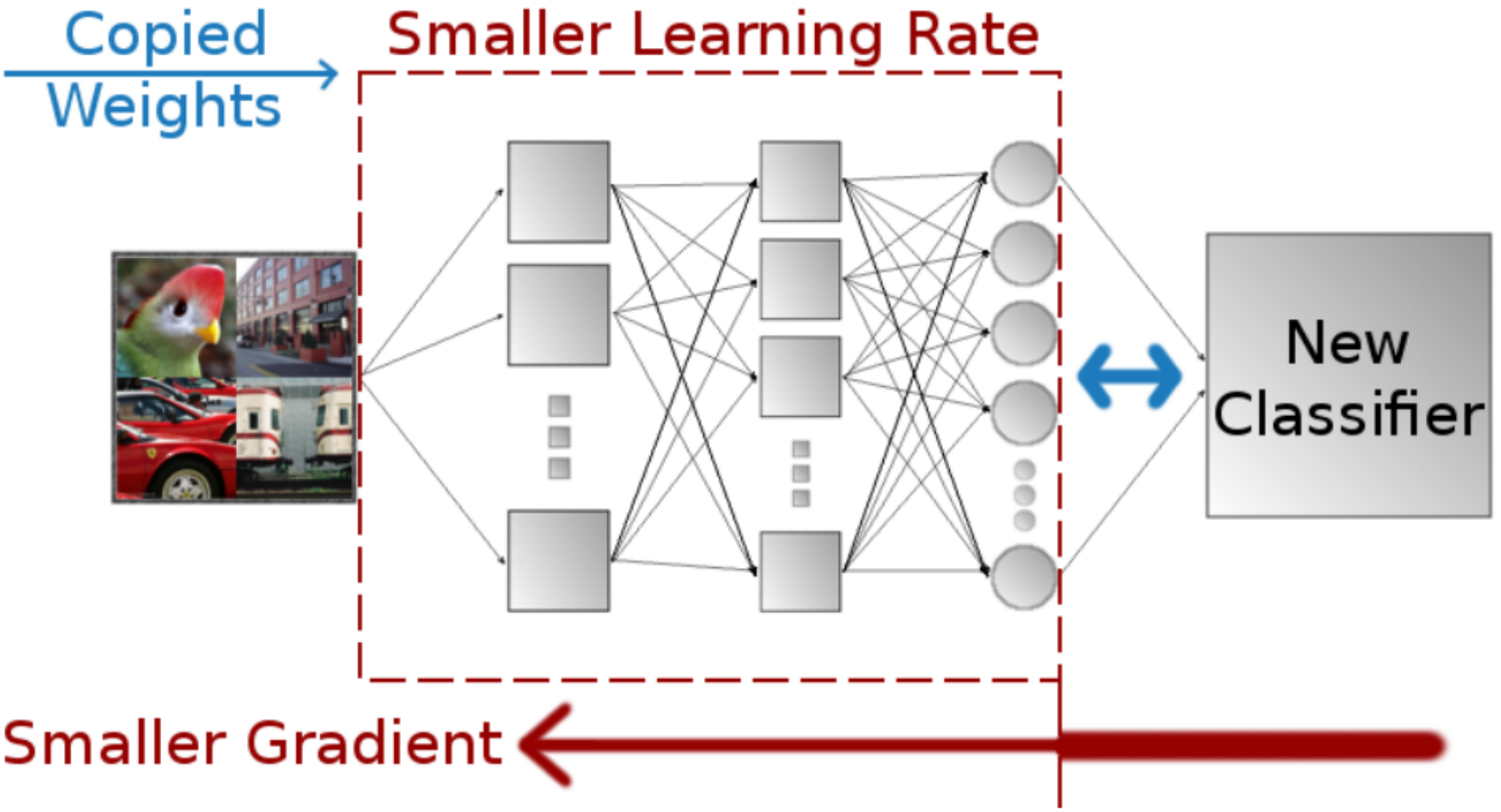


Credit: Razavian *et. al.* [ARS+16]

- Dataset change: fine-tuning ConvNet parameters on the target domain
  - Different learning for fined-tuned & from scratch parameters
- Domain (task) change: adapt archi to the target task (*e.g.* pooling)

Deep Features (DF) and Domain Adaptation

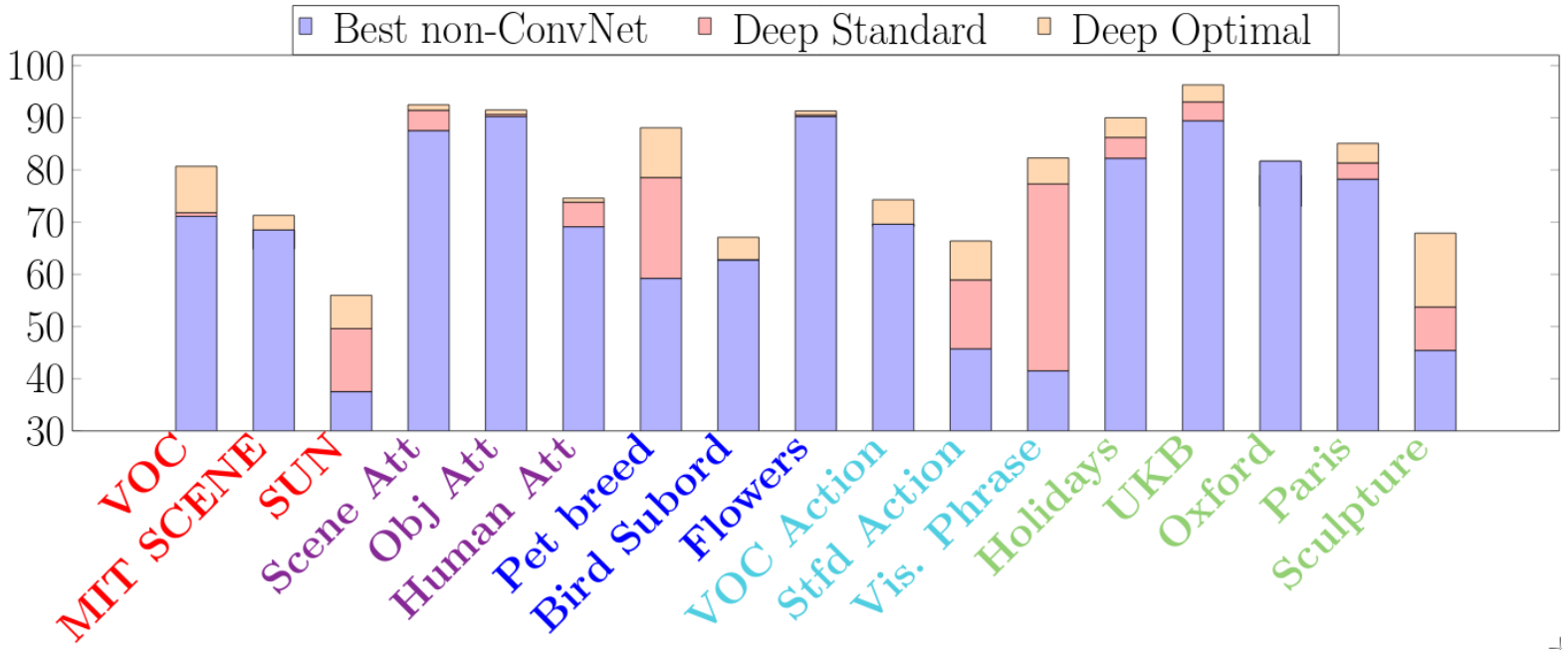
DF: fine-tuning



Deep Features (DF) and Domain Adaptation

Increasing distance from ImageNet →

Image Classification	Attribute Detection	Fine-grained Recognition	Compositional	Instance Retrieval
PASCAL VOC Object [9]	H3D human attributes [6]	Cat&Dog breeds [29]	VOC Human Action [9]	Holiday scenes [17]
MIT 67 Indoor Scenes [33]	Object attributes [10]	Bird subordinate [43]	Stanford 40 Actions [46]	Paris buildings [31]
SUN 397 Scene [45]	SUN scene attributes [30]	102 Flowers [27]	Visual Phrases [34]	Sculptures [4]

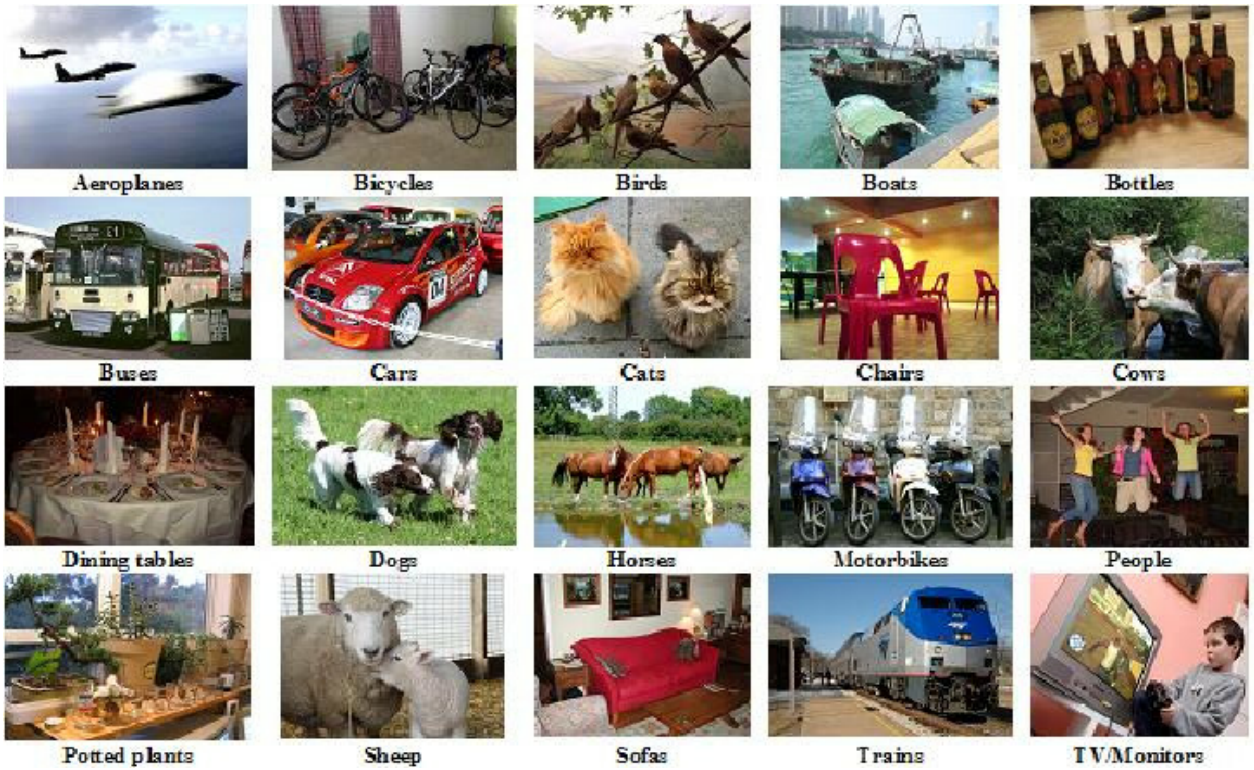


Credit: Razavian *et. al.* [ARS+16]

Deep Features (DF) and Domain Adaptation

DF for Image classification on other datasets

- **Small size datasets**: $\sim 10^3 - 10^4$ ex, *e.g.* VOC'07 (20 classes, 5000 ex)



Model type	Test mAP
From Scratch	39.79
BoW [ATC+13]	61.6
Transfer	83.22
Fine Tuning	**85.70**

Table: Mean Average Precision.

- Fine Tuning > Transfer >> Handcrafted (BoW)
- From scratch does not work (not enough training data)

Deep Features (DF) and Domain Adaptation

DF for Image classification on other datasets

- **Medium size datasets**, *e.g.* Food (LIP6) : 101 classes, 10^5 ex

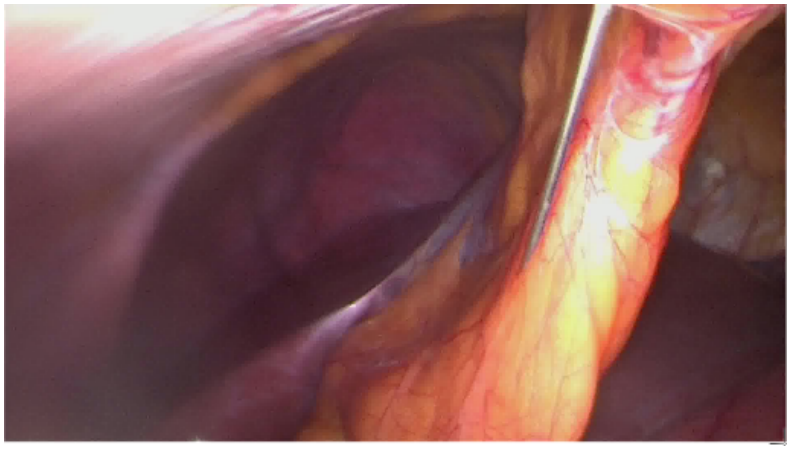
Modèle	Test top 1 (%)
(a) Bag of visual Words	23.96
Overfeat & Extraction	33.91
Overfeat & From Scratch	47.46
Overfeat & Fine Tuning	57.98
(b) Vgg16 & Extraction	40.21
Vgg16 & From Scratch	53.62
Vgg16 & Fine Tuning	**65.71**

- From scratch DOES work (well !)
- Fine Tuning >> From scratch >> Transfer >> Handcrafted (BoW)

Deep Features (DF) and Domain Adaptation

DF for Image classification on other datasets

- Another ex: M2CAI'16 challenge - large domain shift (medical images)
  - Medium-size: 22 videos, $\sim 60 \cdot 10^4$ images, 8 classes

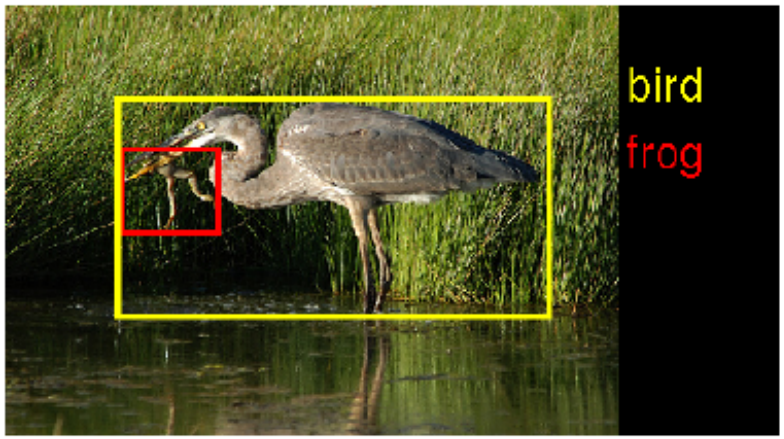


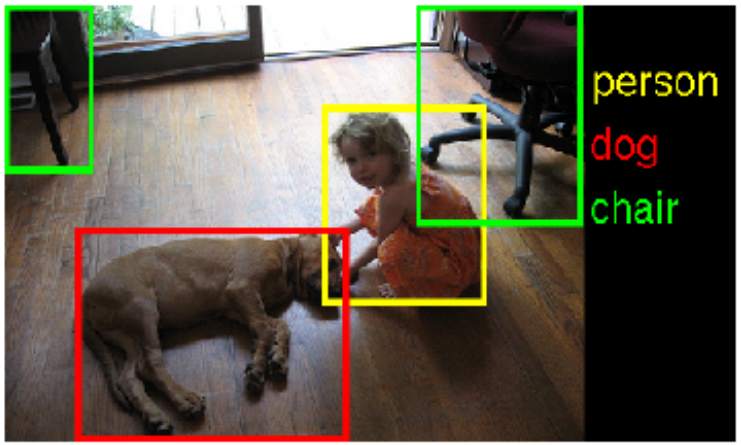
Model	Acc Top1(%)
Transfer	59.27
From Scratch	69.13
Fine Tuning	79.06

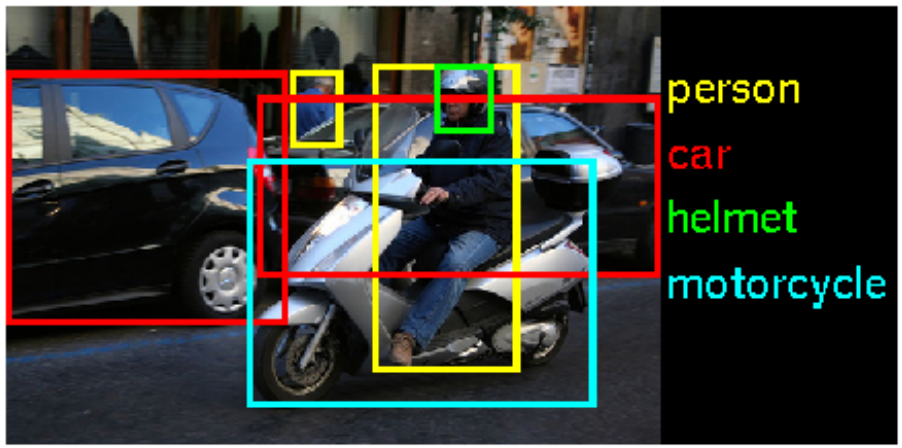
- Fine Tuning >> From scratch >> Transfer
- Transfer already good baseline despite big visual content shift

Deep Features (DF) and Domain Adaptation

Task Adaptation: Localization



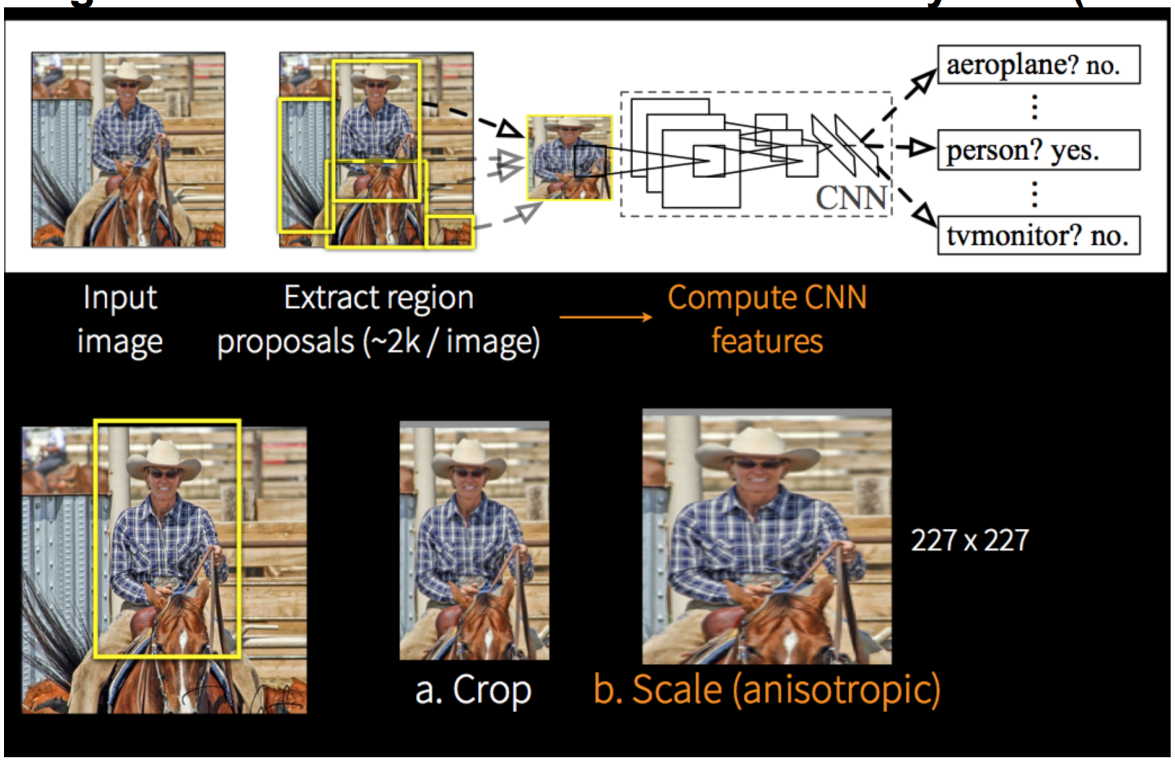




Deep Features (DF) and Domain Adaptation

Task Adaptation: Localization

Regions with Convolutional Neural Net.s system (RCNN)



R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR* 14

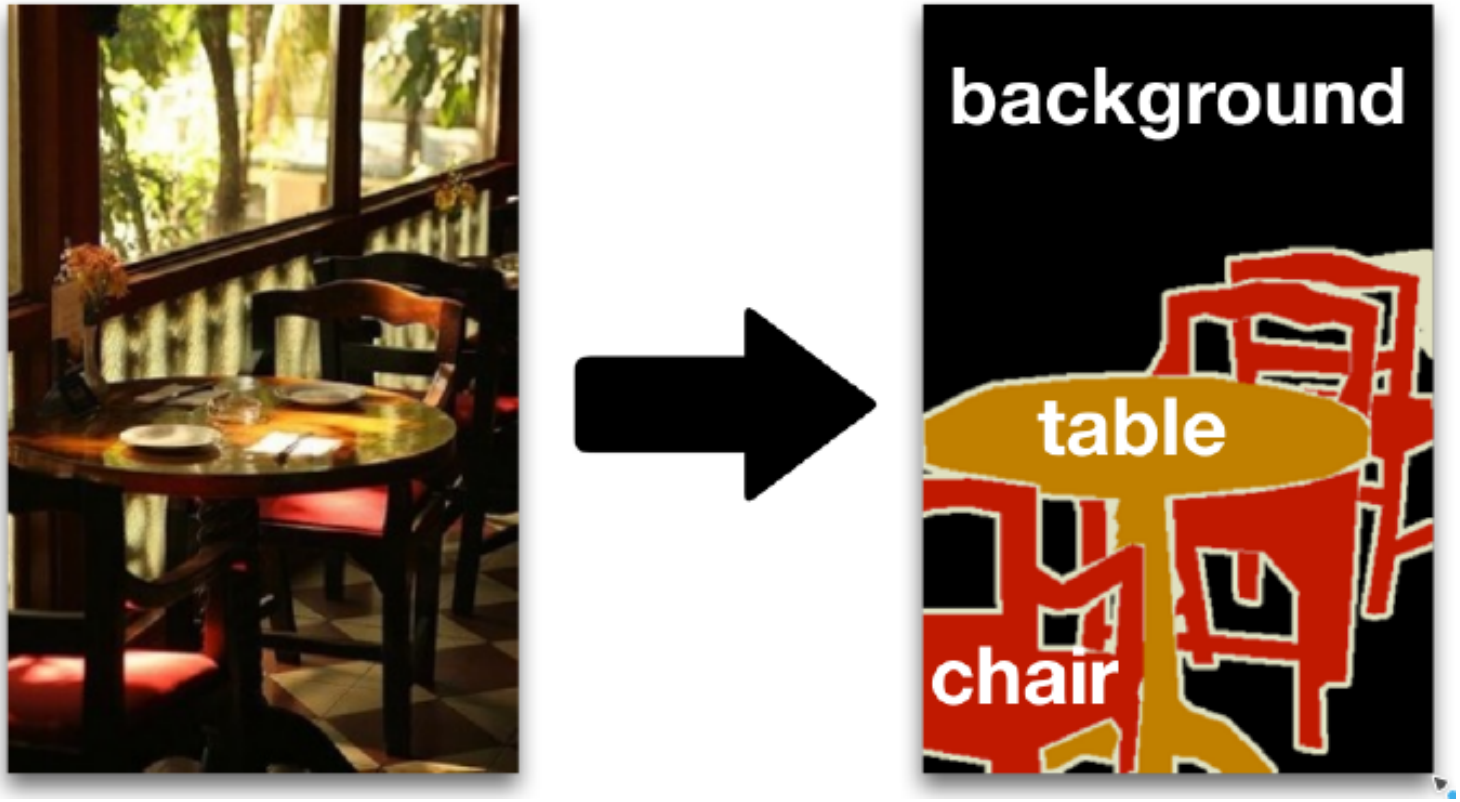
Eval on VOC'07:

R-CNN	**58.5**
DPM HoG	34.3

- R-CNN ⇒ region proposals ⇒ Deep Features ⇒ classify
- Significantly outperformed previous models (DPM on HoG features)

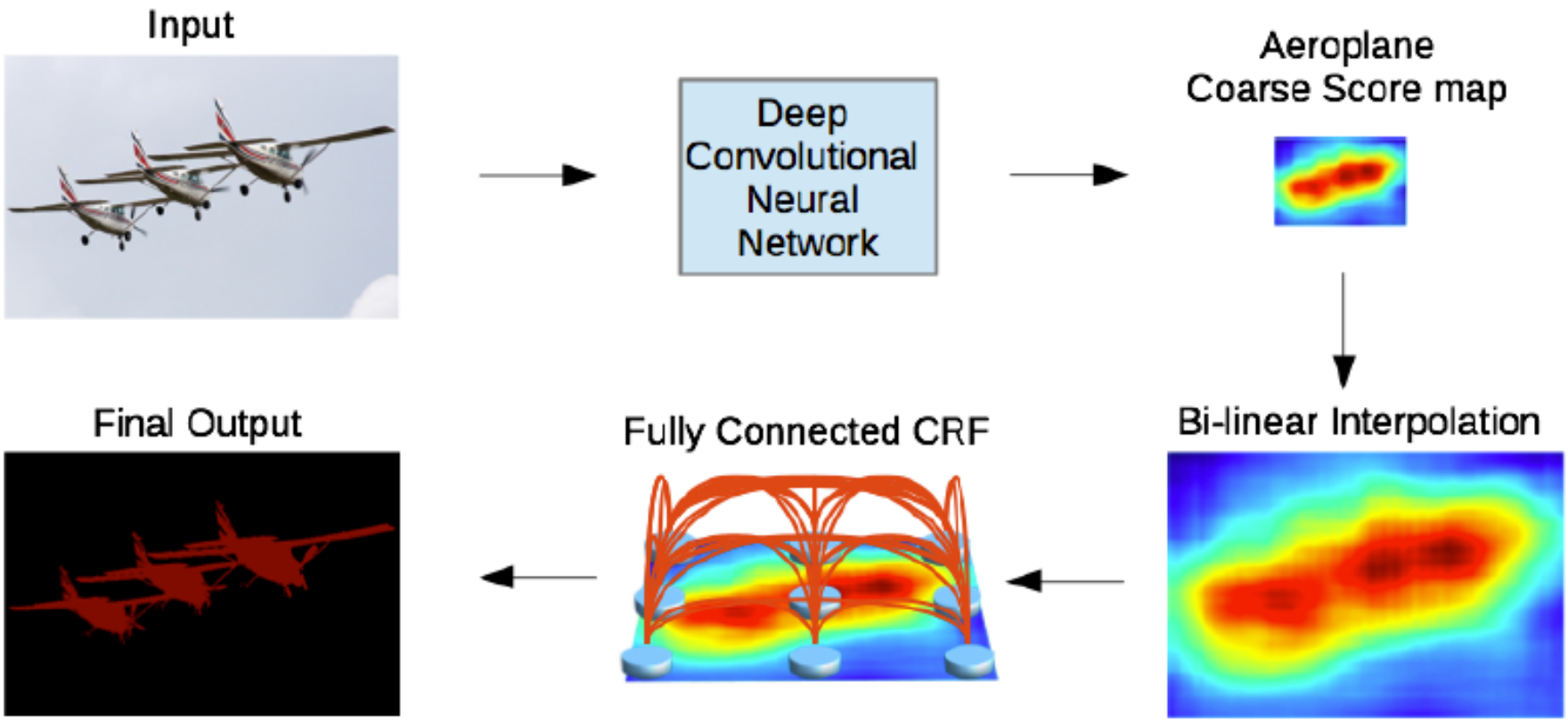
Deep Features (DF) and Domain Adaptation

Task Adaptation: Semantic Segmentation



Deep Features (DF) and Domain Adaptation

Task Adaptation: Semantic Segmentation

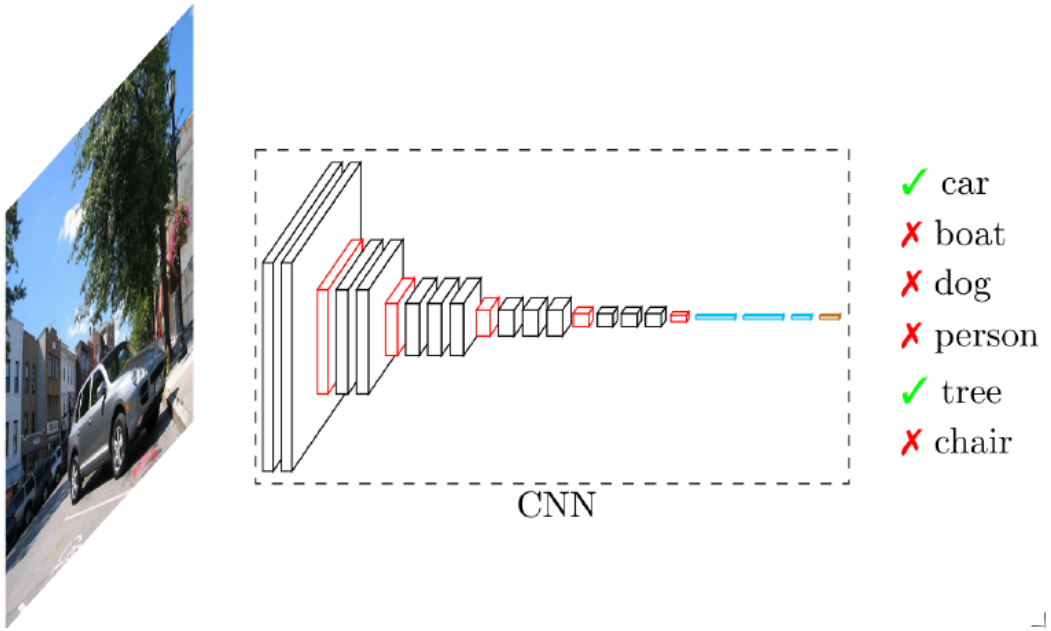


Chen *et.al.* ICLR'15

Deep Features (DF) and Domain Adaptation

Conclusion

- Deep Feature in transfer mode
  - Very good baseline
  - Ofter > descriptors based on expert knowledge
  - The solution for small databases
- From medium-size: from scratch possible and very competitive
- Fine-tuning always improves performances (small → large datasets)



References I


Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson, *Factors of transferability for a generic convnet representation*, IEEE Trans. Pattern Anal. Mach. Intell. **38** (2016), no. 9, 1790–1802.


George Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems **2** (1989), no. 4, 303–314.


Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh, *A fast learning algorithm for deep belief nets*, Neural Comput. **18** (2006), no. 7, 1527–1554.


Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 2012, pp. 1097–1105.


Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel, *Backpropagation applied to handwritten zip code recognition*, Neural computation **1** (1989), no. 4, 541–551.


Warren S McCulloch and Walter Pitts, *A logical calculus of the ideas immanent in nervous activity*, The bulletin of mathematical biophysics **5** (1943), no. 4, 115–133.