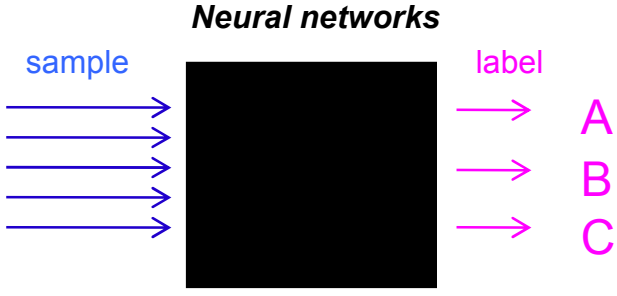
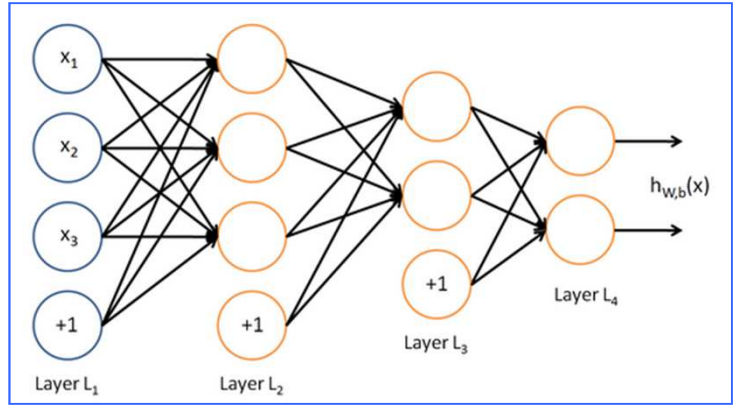
Intelligent Image and Graphics Processing

Convolutional Neural Networks

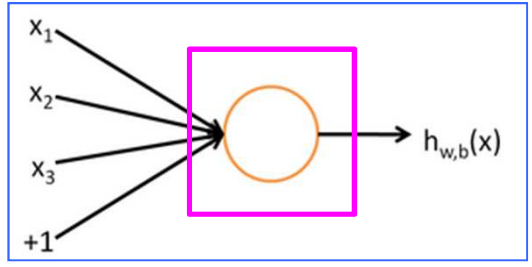
布树辉
bushuhui@nwpu.edu.cn
http://www.adv-ci.com

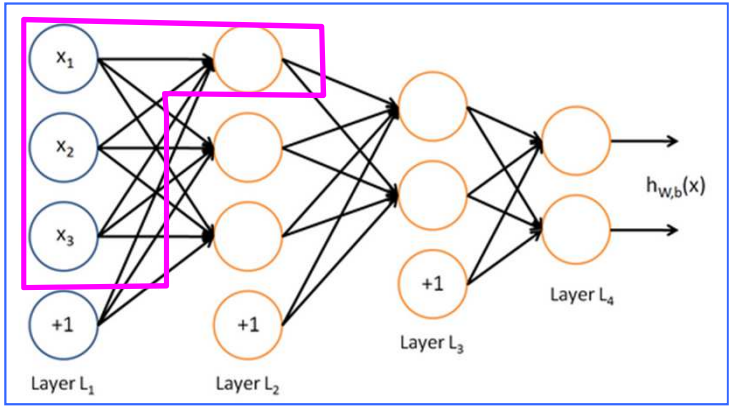
Neural networks

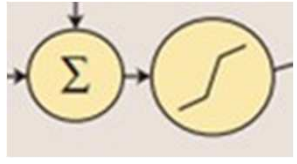




Neural networks - feedforward networks







$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)} \sum_{i=1}^{3} W_i x_i + b)$$

$$a^{(l+1)} = f(z^{(l+1)})$$

Neural networks - Network training

Cost function of Single training example

$$J(W,b;x,y) = \frac{1}{2}\left\|h_{W,b}(x) - y\right\|^2$$

统计学习常用的损失函数有以下几种：

（1）0-1 损失函数（0-1 loss function）

$$L(Y,f(X)) = \begin{cases} 1, & Y \neq f(X) \\ 0, & Y = f(X) \end{cases} \quad (1.5)$$

（2）平方损失函数（quadratic loss function）

$$L(Y,f(X)) = (Y - f(X))^2 \quad (1.6)$$

（3）绝对损失函数（absolute loss function）

$$L(Y,f(X)) = |Y - f(X)| \quad (1.7)$$

（4）对数损失函数（logarithmic loss function）或对数似然损失函数（log-likelihood loss function）

$$L(Y,P(Y|X)) = -\log P(Y|X) \quad (1.8)$$

Cost function of all examples

$$\begin{aligned} J(W,b) &= \left[\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \\ &= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x^{(i)}) - y^{(i)}\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \end{aligned}$$

Gradient Descent

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha\frac{\partial}{\partial W_{ij}^{(l)}}J(W,b)$$

$$b_i^{(l)} = b_i^{(l)} - \alpha\frac{\partial}{\partial b_i^{(l)}}J(W,b)$$

Neural networks - Gradient Descent

$J(W,b) =$

1. 对于所有 l，令 $\Delta W^{(l)} := 0$，$\Delta b^{(l)} := 0$ （设置为全零矩阵或全零向量），n_l 的激活值。
2. 对于 $i = 1$ 到 m，
   a. 使用反向传播算法计算 $\nabla_{W^{(l)}} J(W,b;x,y)$ 和 $\nabla_{b^{(l)}} J(W,b;x,y)$。
   b. 计算 $\Delta W^{(l)} := \Delta W^{(l)} + \nabla_{W^{(l)}} J(W,b;x,y)$。
   c. 计算 $\Delta b^{(l)} := \Delta b^{(l)} + \nabla_{b^{(l)}} J(W,b;x,y)$。
3. 更新权重参数：

$=$

$\frac{\partial}{\partial W_{ij}^{(l)}} J$

$\frac{\partial}{\partial b_i^{(l)}} J$

残差计算方法如下：

$$W^{(l)} = W^{(l)} - \alpha \left[\left(\frac{1}{m} \Delta W^{(l)} \right) + \lambda W^{(l)} \right]$$

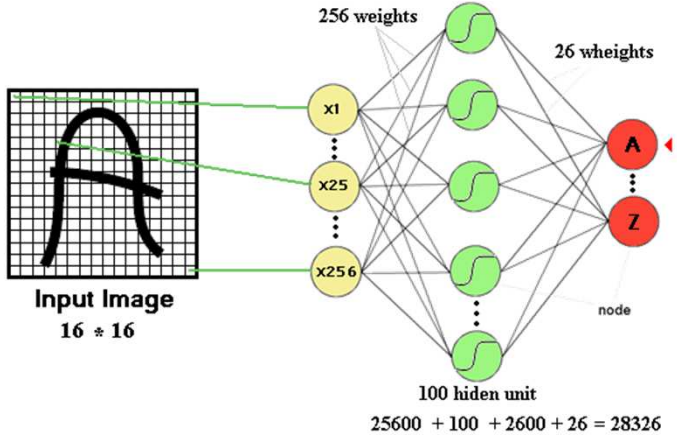
$$b^{(l)} = b^{(l)} - \alpha \left[\frac{1}{m} \Delta b^{(l)} \right]$$

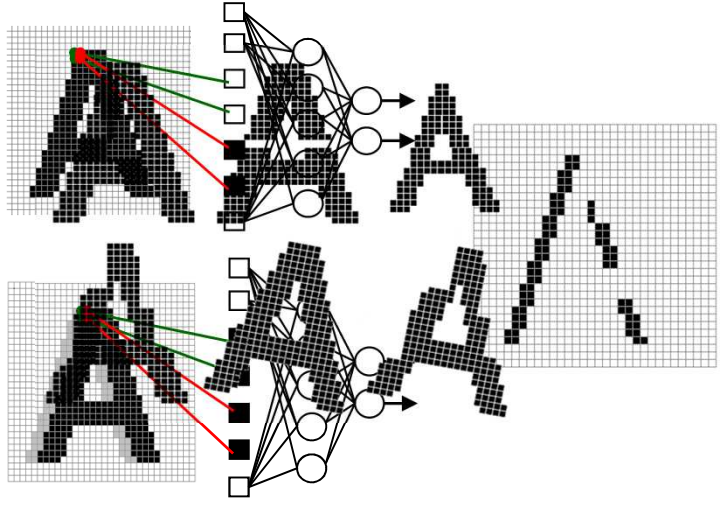
$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W,b;x,y) = \delta_i^{(l+1)}.$$

UFDL: http://ufldl.stanford.edu/wiki/index.php/Backpropagation_Algorithm

Neural networks - Disadvantages

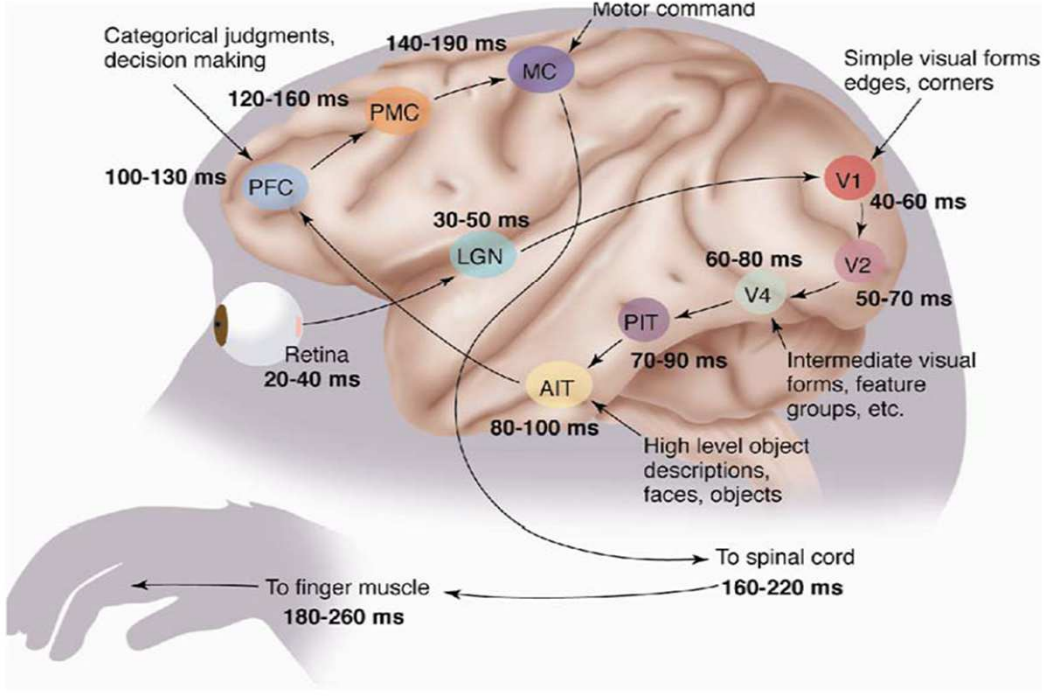
- The number of trainable parameters becomes extremely large
- Little or no invariance to shifting, scaling, and other forms of distortion





Mammalian visual pathway is hierarchical

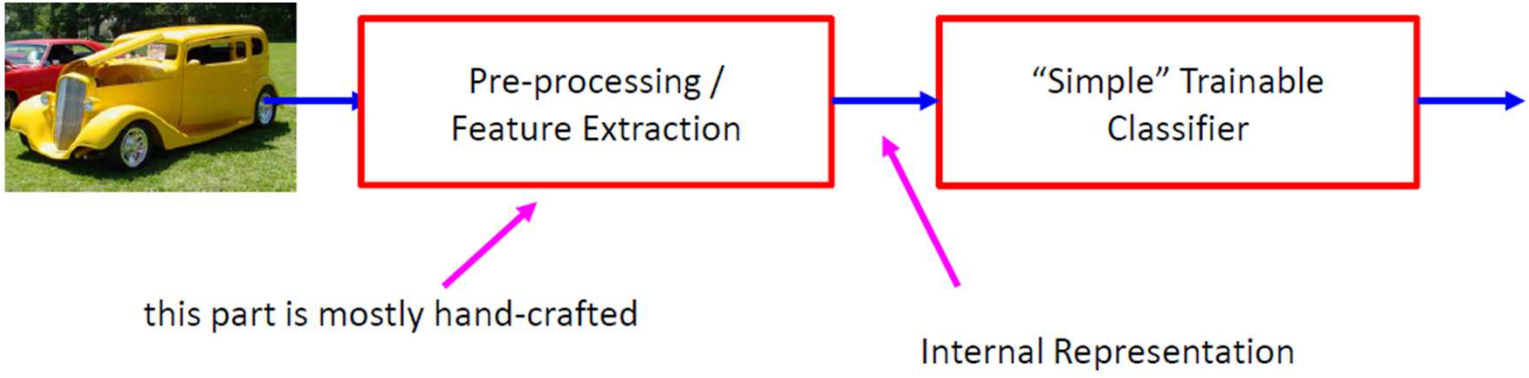
- The ventral (recognition) pathway in the visual cortex
  - Retina → LGN → V1 → V2 → V4 → PIT → AIT (80-100ms)



Some challenges for machine learning in vision

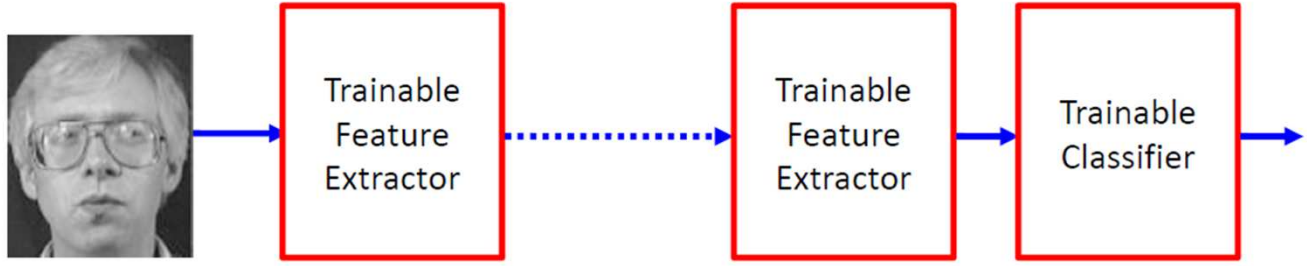
- Efficient Learning Algorithms for multi-stage “Deep Architectures”
  - Multi-stage learning is difficult
- Learning good internal representations of the world (features)
  - Hierarchy of features that capture the relevant information and eliminates irrelevant variabilities
- Learning with unlabeled samples (unsupervised learning)
  - Animals can learn vision by just looking at the world
  - No supervision is required
- Deep learning algorithms that can be applied to other modalities
  - If we can learn feature hierarchies for vision, we can learn feature hierarchies for audition, natural language, ...

The traditional “shallow” architecture for recognition



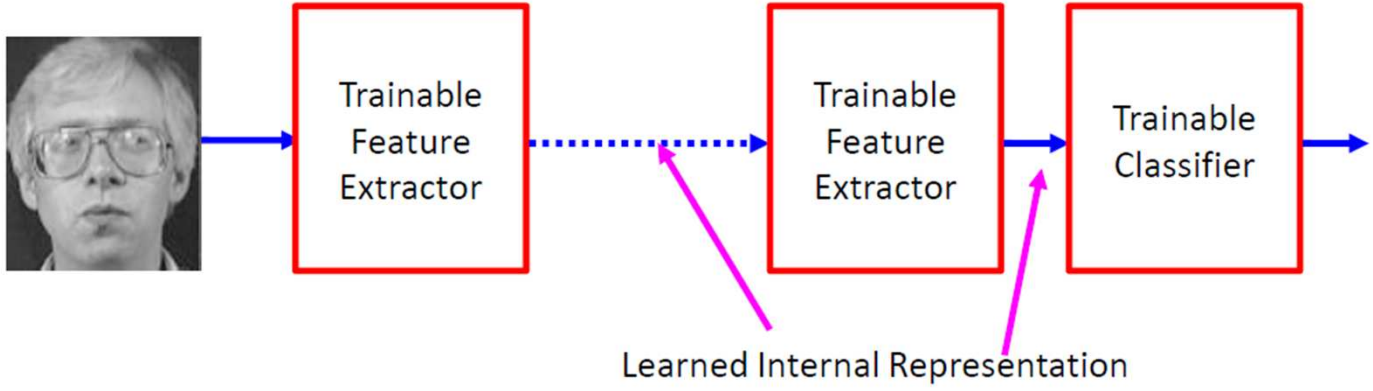
- The raw input is pre-processed through a hand-crafted feature extractor
- The features are not learned
- The trainable classifier is often generic (task independent), and “simple” (linear classifier, kernel machine, nearest neighbor,.....)
- The most common Machine Learning architecture: the Kernel Machine

Good representations are hierarchical



- In Language: hierarchy in syntax and semantics
  - Words->Parts of Speech->Sentences->Text
  - Objects,Actions,Attributes...-> Phrases -> Statements -> Stories
- In Vision: part-whole hierarchy
  - Pixels->Edges->Textons->Parts->Objects->Scenes

“Deep” learning: learning hierarchical representations

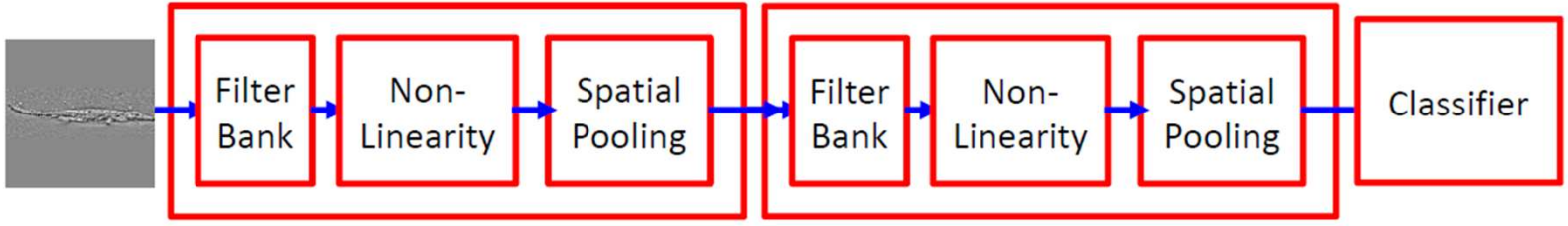


- Deep Learning: learning a hierarchy of internal representations
- From low-level features to mid-level invariant representations, to object identities
- Representations are increasingly invariant as we go up the layers
- using multiple stages gets around the specificity/invariance dilemma

Do we really need deep architecture?

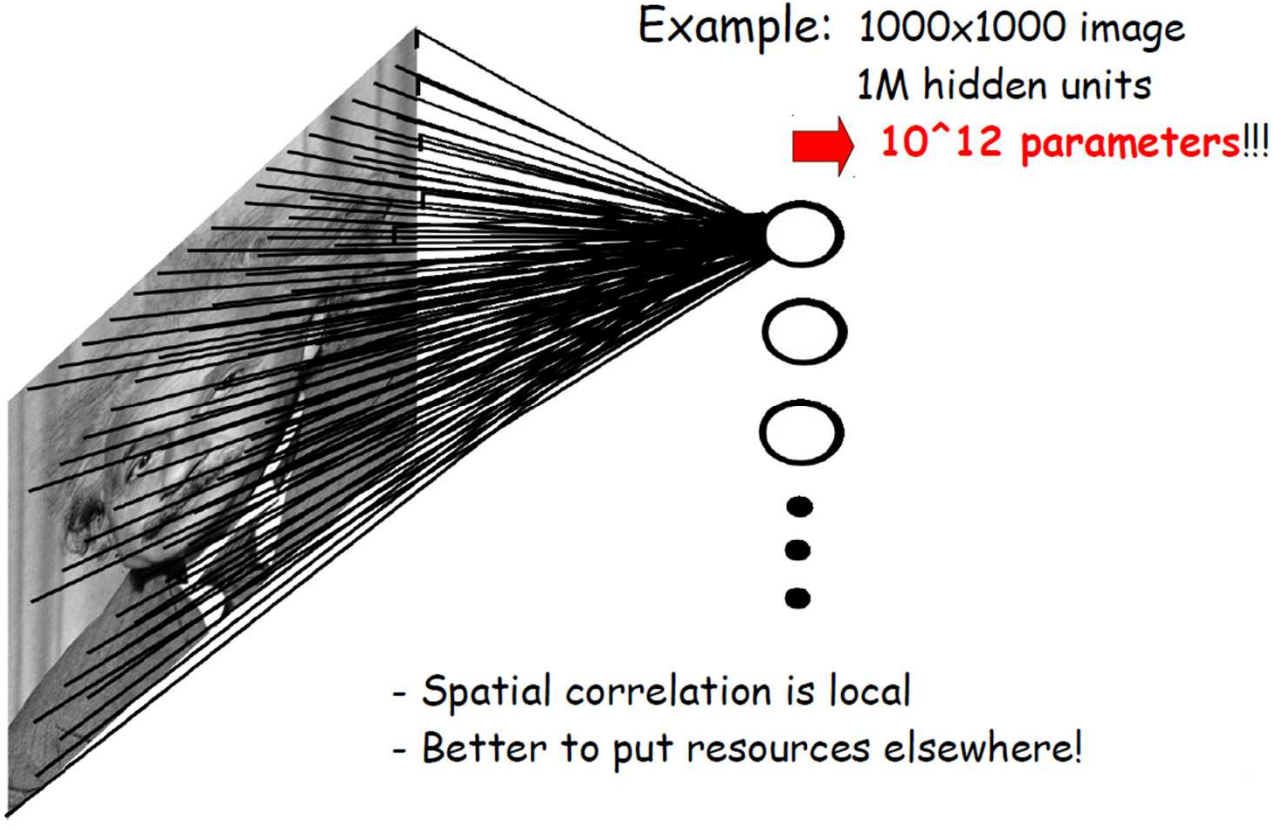
- We can approximate any function as close as we want with shallow architecture. Why would we need deep ones?
  - kernel machines and 2-layer neural net are “universal”.
- Deep learning machines
- Deep machines are more efficient for representing certain classes of functions, particularly those involved in visual recognition
  - they can represent more complex functions with less “hardware”
- We need an efficient parameterization of the class of functions that are useful for “AI” tasks.

Hierarchical / deep architectures for vision

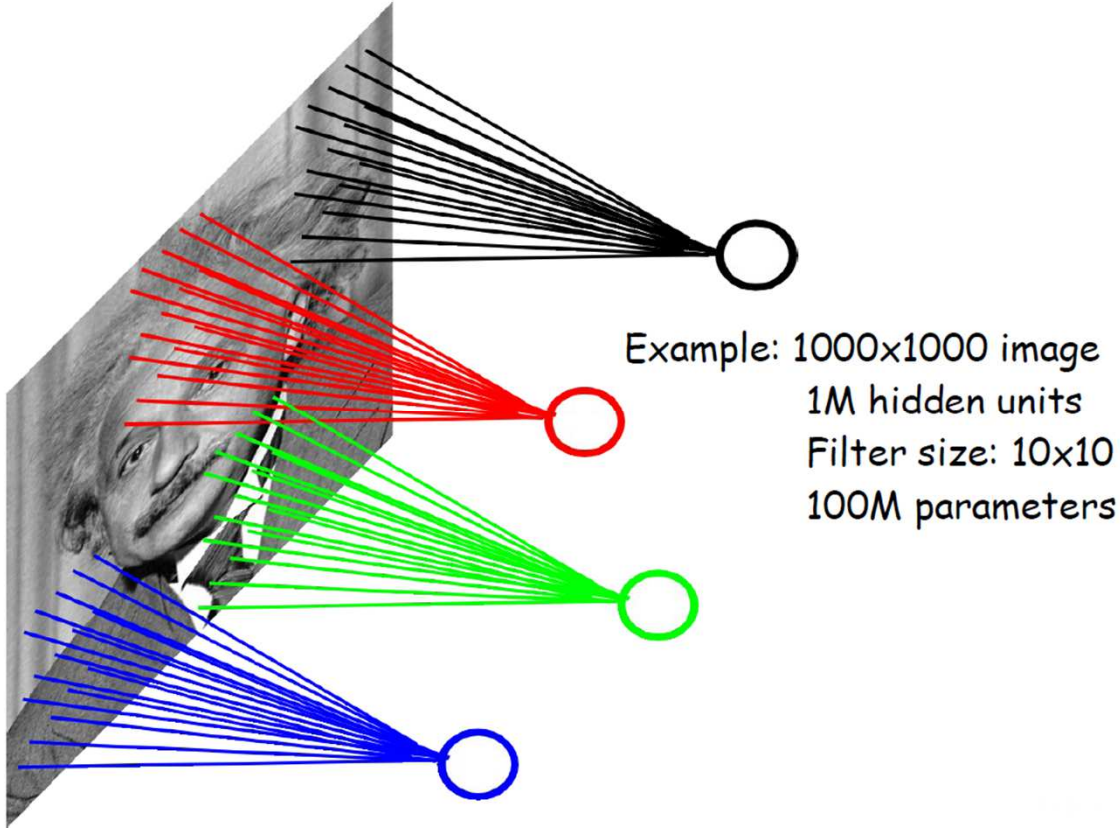


- Multiple Stages
- Each Stage is composed of
  - A bank of local filters (convolutions)
  - A non-linear layer (may include harsh non-linearities, such as rectification, contrast normalization, etc...).
  - A feature pooling layer
- Multiple stages can be stacked to produce high-level representations
  - Each stage makes the representation more global, and more invariant
- The systems can be trained with a combination of unsupervised and supervised methods

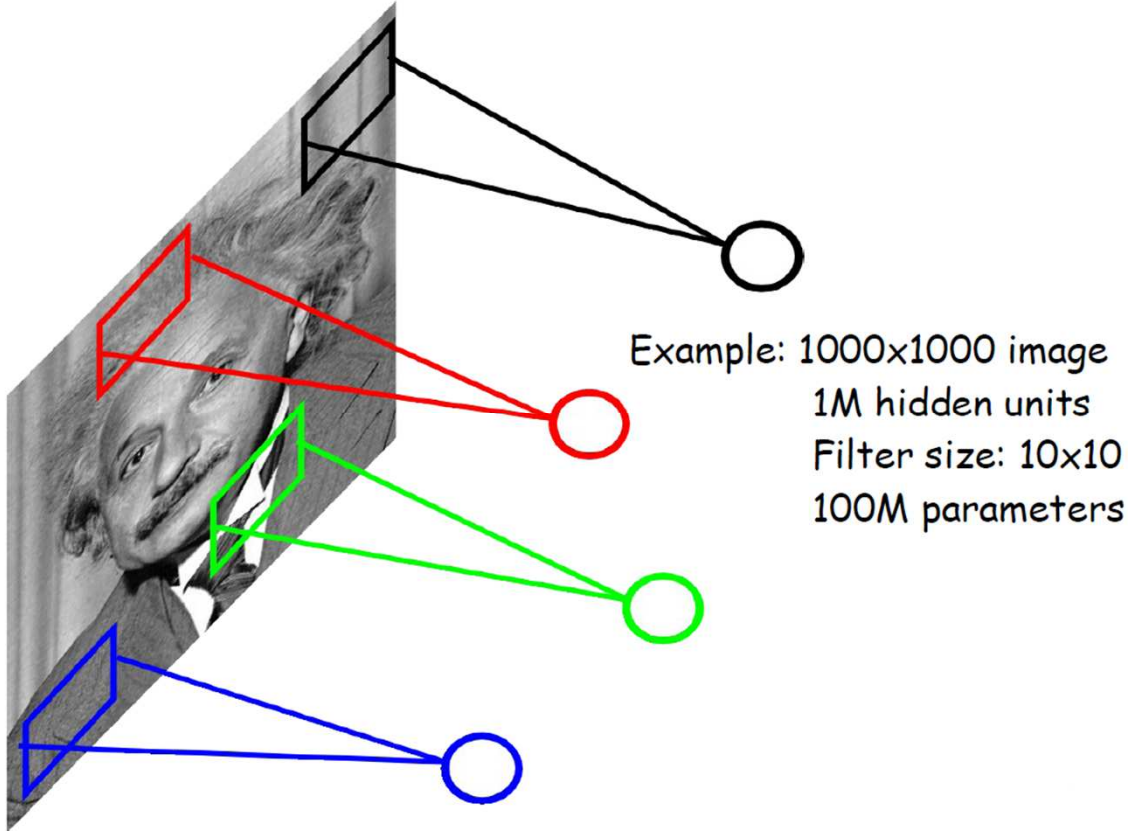
Fully connected neural networks



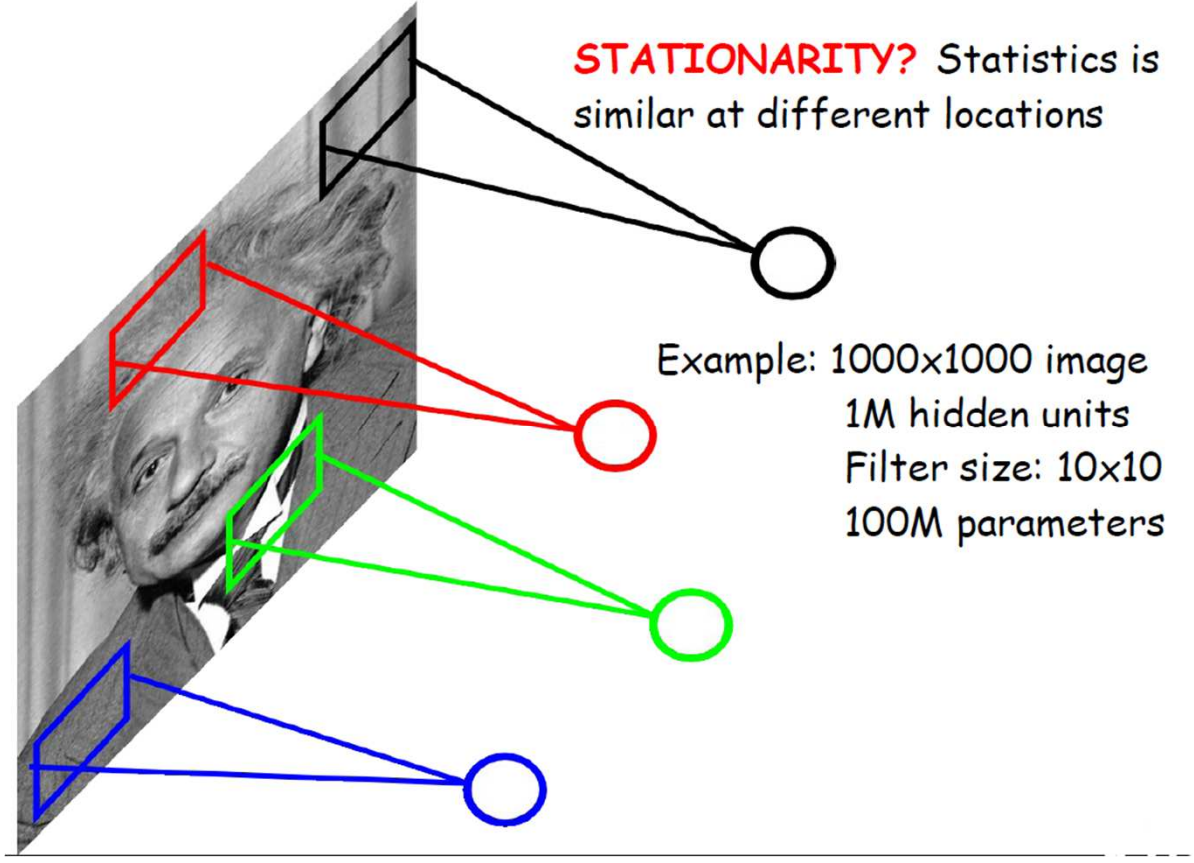
Locally connected neural networks



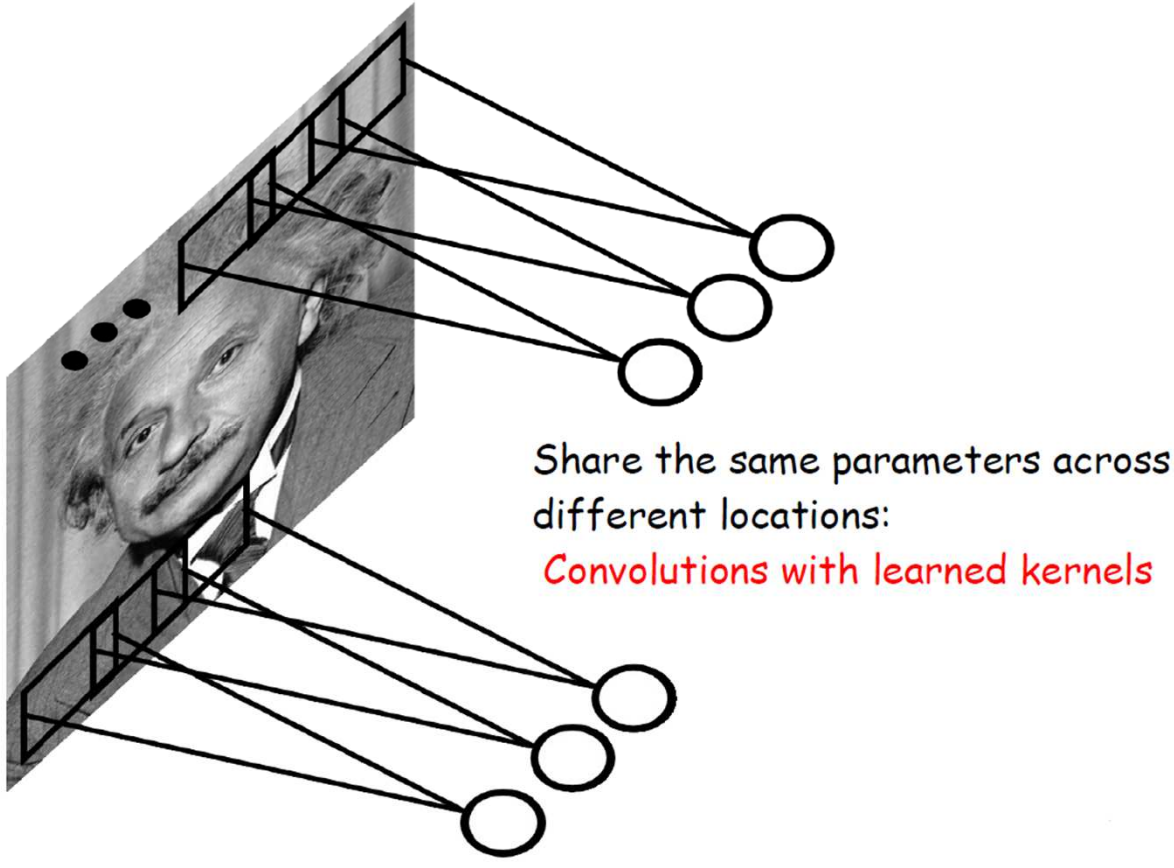
Locally connected neural networks



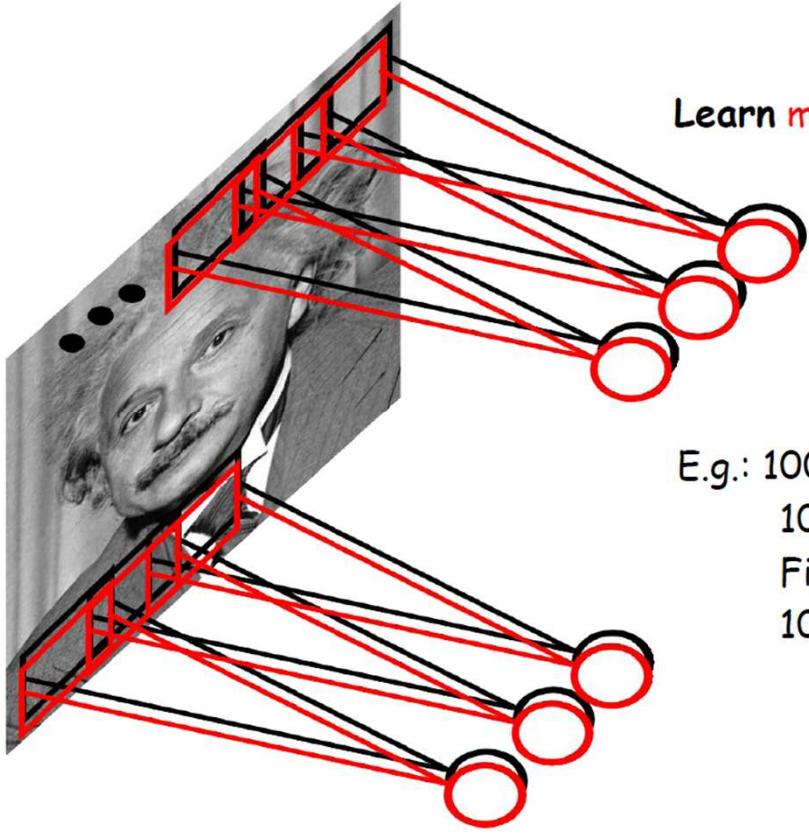
Locally connected neural networks



Convolutional Network



Convolutional Network



Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

Neural network for vision

A standard neural network applied to images:

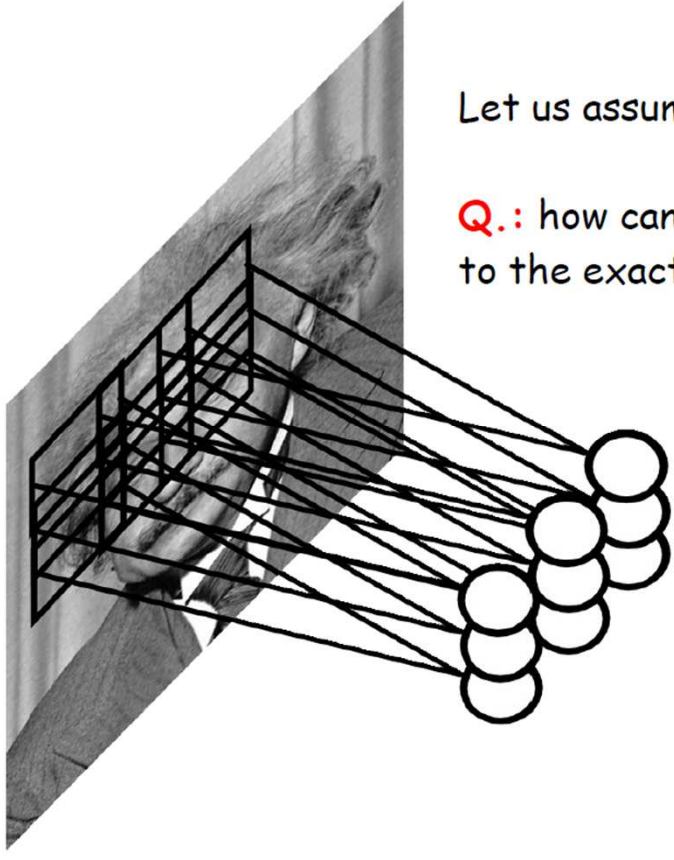
- Scales quadratically with the size of the input
- Does not leverage stationarity

Solution:

- Connect each hidden unit to a small patch of the input
- Share the weight across hidden units

This is called: convolutional network

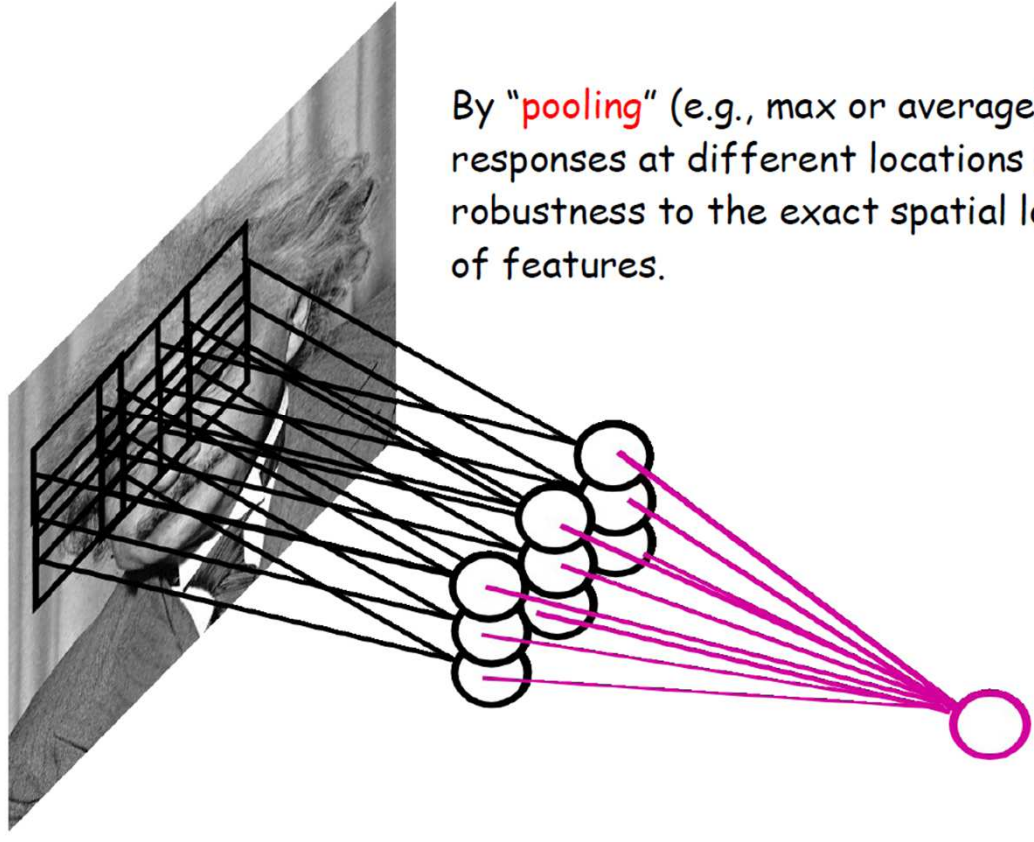
Convolutional network



Let us assume filter is an "eye" detector.

Q.: how can we make the detection robust to the exact location of the eye?

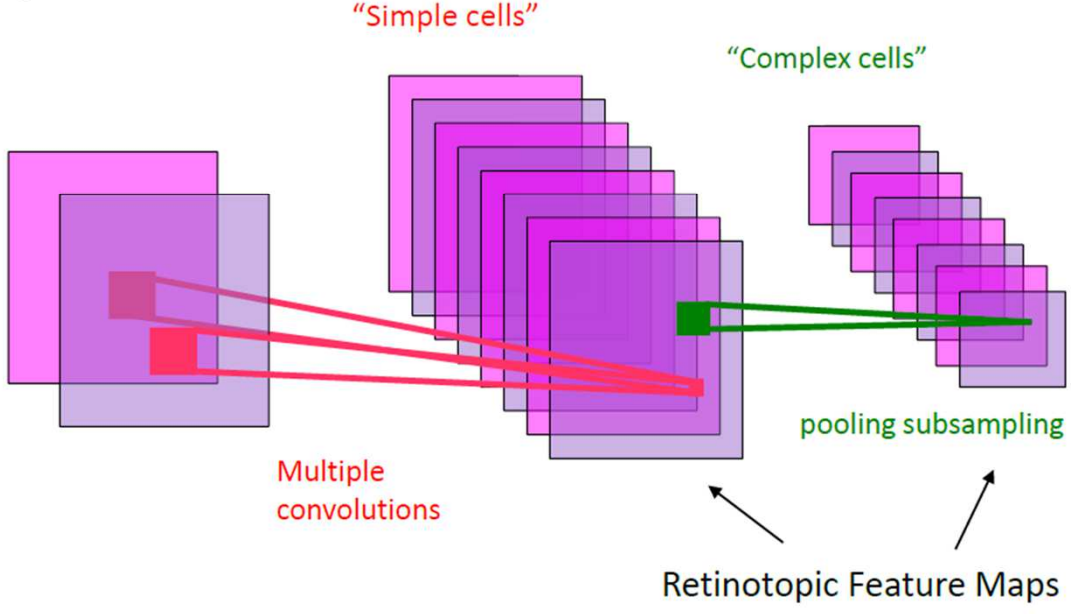
Convolutional network



By "pooling" (e.g., max or average) filter responses at different locations we gain robustness to the exact spatial location of features.

Filtering + Nonlinearity + Pooling = 1 stage of a Convolutional Network

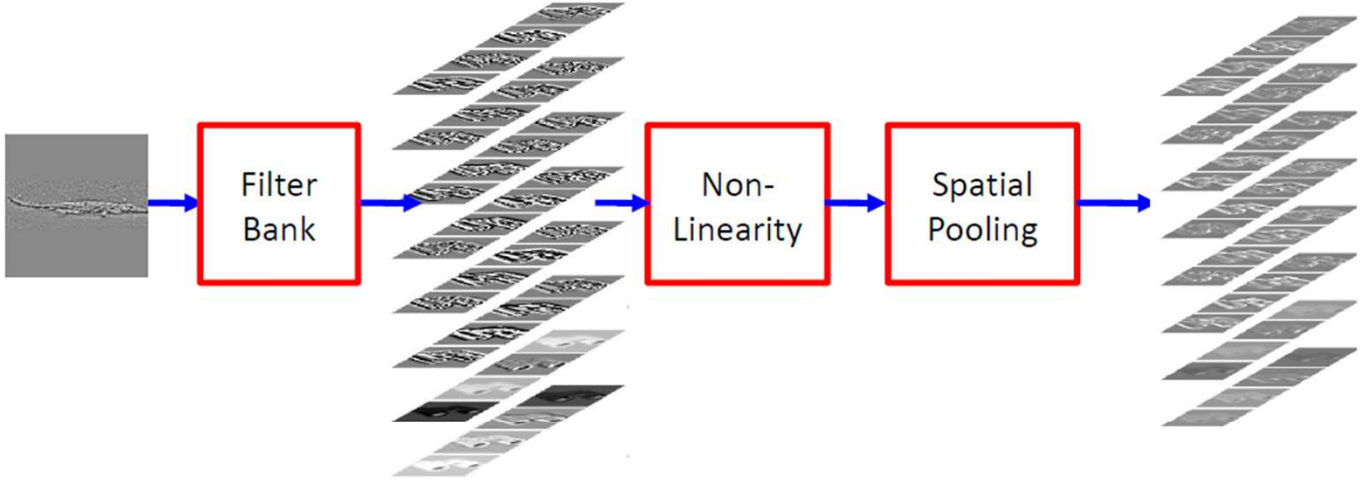
- [Hubel & Wiesel 1962]:
  - simple cells detect local features
  - complex cells "pool" the outputs of simple cells within a retinotopic neighborhood.



Convolutional Neural networks

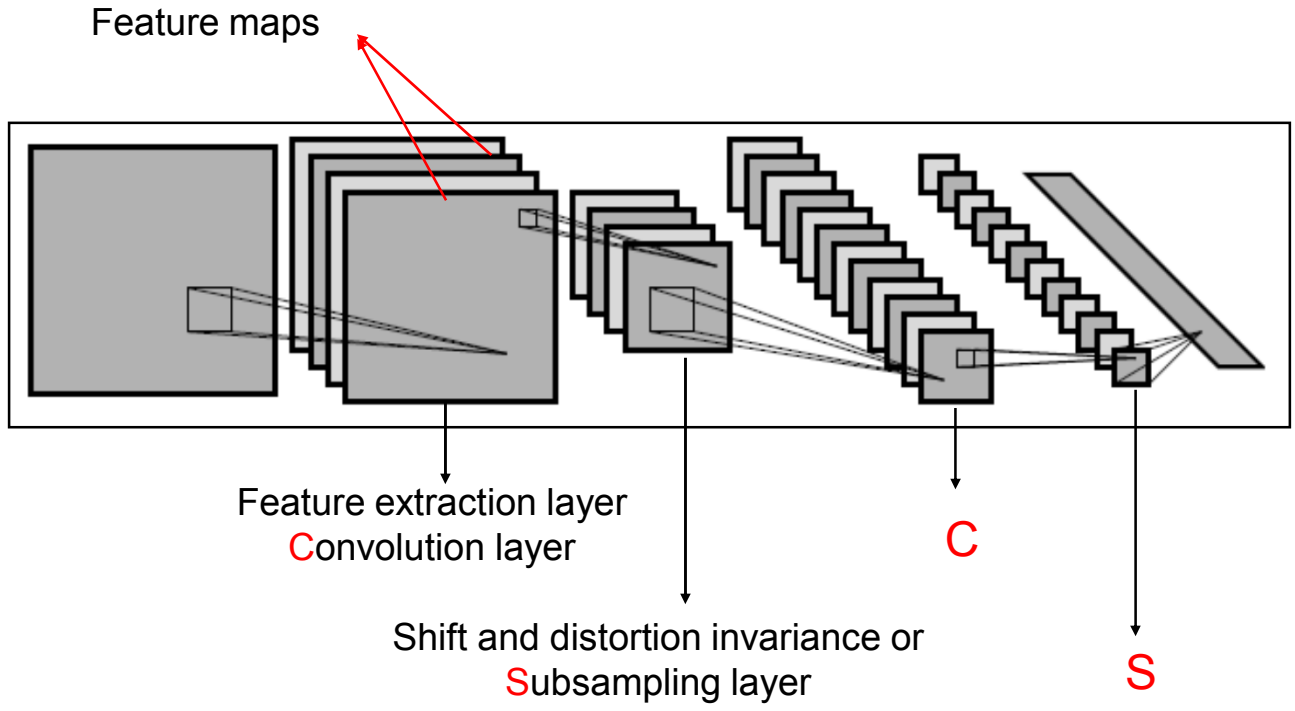
- CNN is a feed-forward network that can extract topological properties from an image.
- Like almost every other neural networks they are trained with a version of the back-propagation algorithm.
- They can recognize patterns with extreme variability (such as handwritten characters).

Feature extraction by filtering and pooling

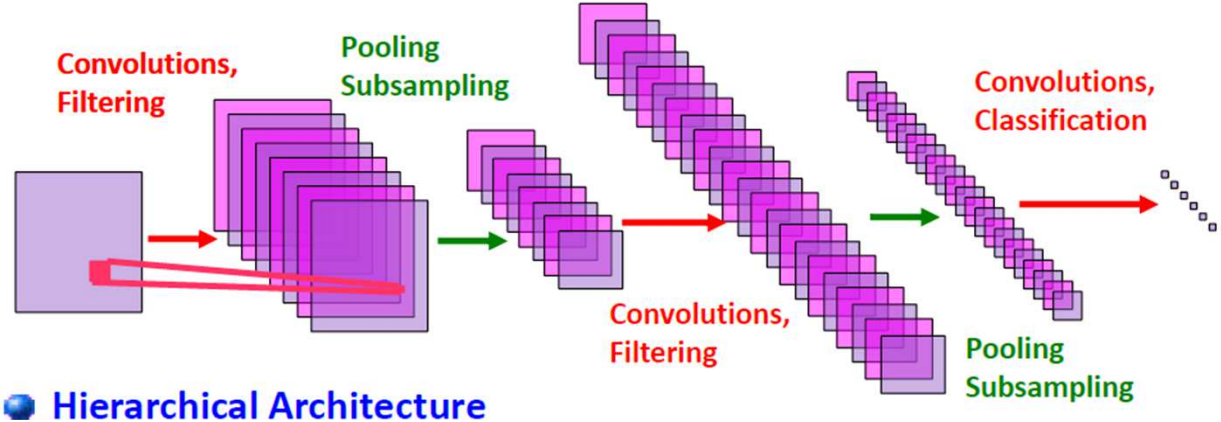


- Biologically-inspired models of low-level feature extraction
  - Inspired by [Hubel and Wiesel 1962]
- Only 3 types of operations needed for traditional ConvNets:
  - Convolution/Filtering
  - Pointwise Non-Linearity
  - Pooling/Subsampling

Convolutional Neural networks

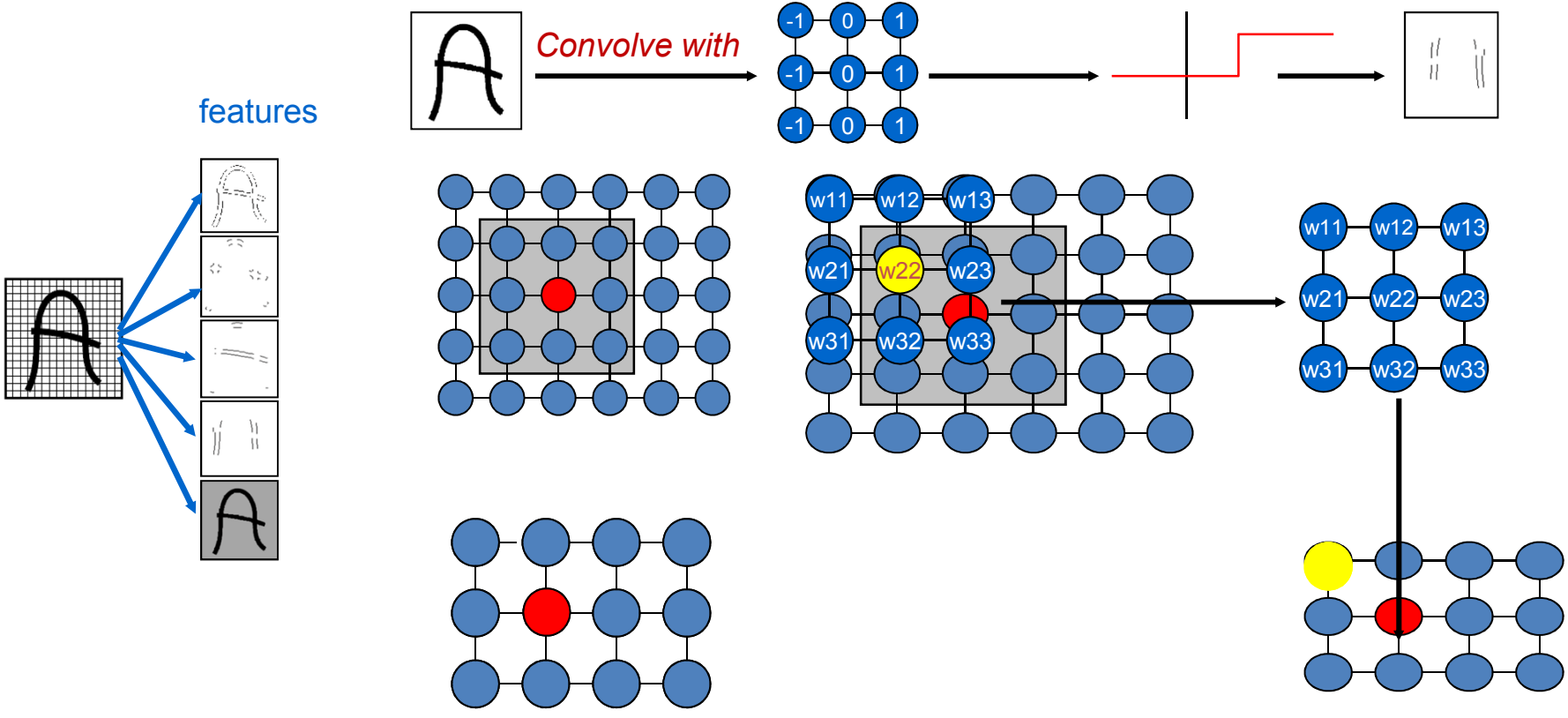


Convolutional Network: multi-stage trainable architecture

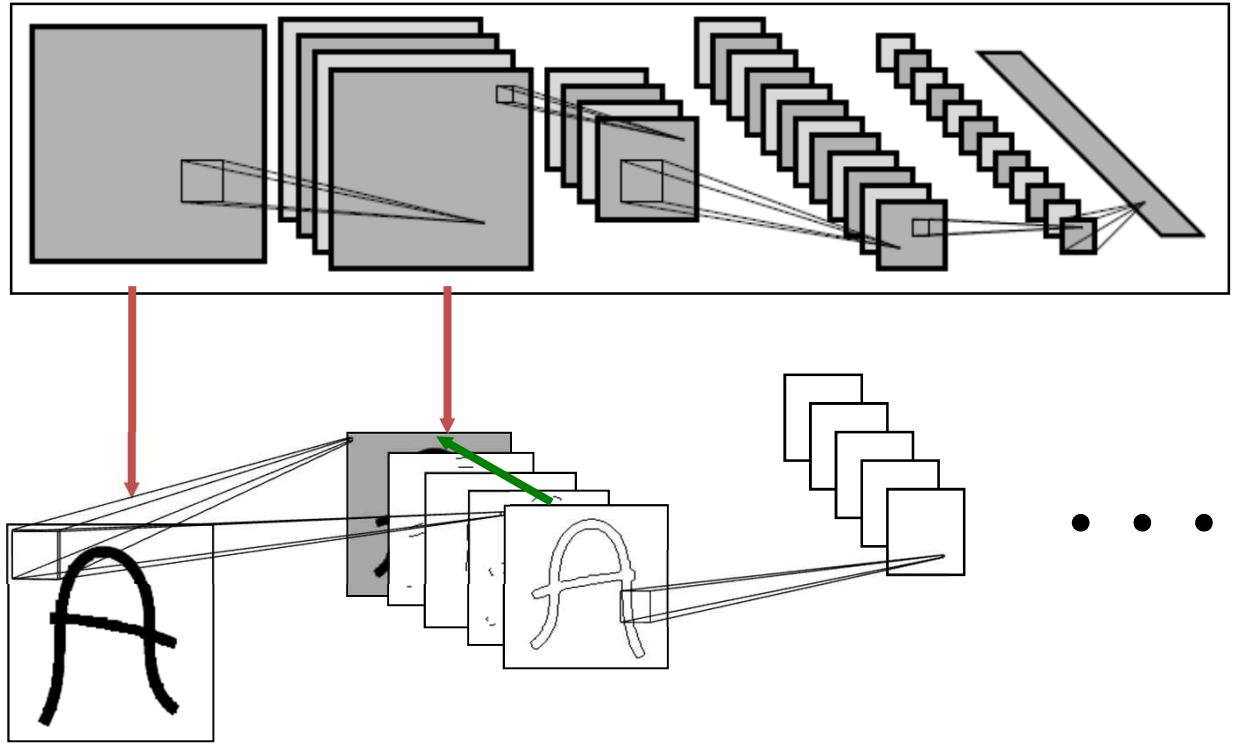


- **Hierarchical Architecture**
  - Representations are more global, more invariant, and more abstract as we go up the layers
- **Alternated Layers of Filtering and Spatial Pooling**
  - Filtering detects conjunctions of features
  - Pooling computes local disjunctions of features
- **Fully Trainable**
  - All the layers are trainable

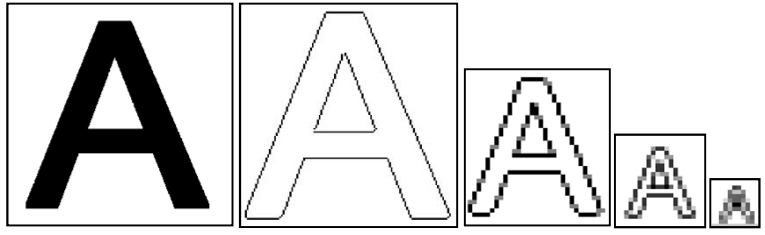
Convolutional layer or Feature extraction layer

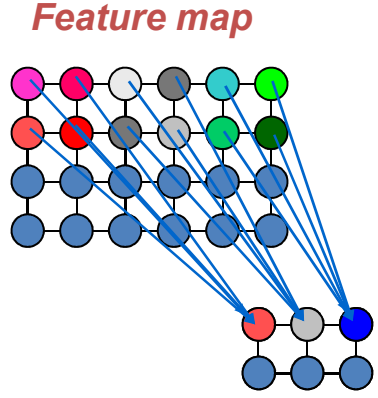


Convolutional layer or Feature extraction layer



Subsampling layer

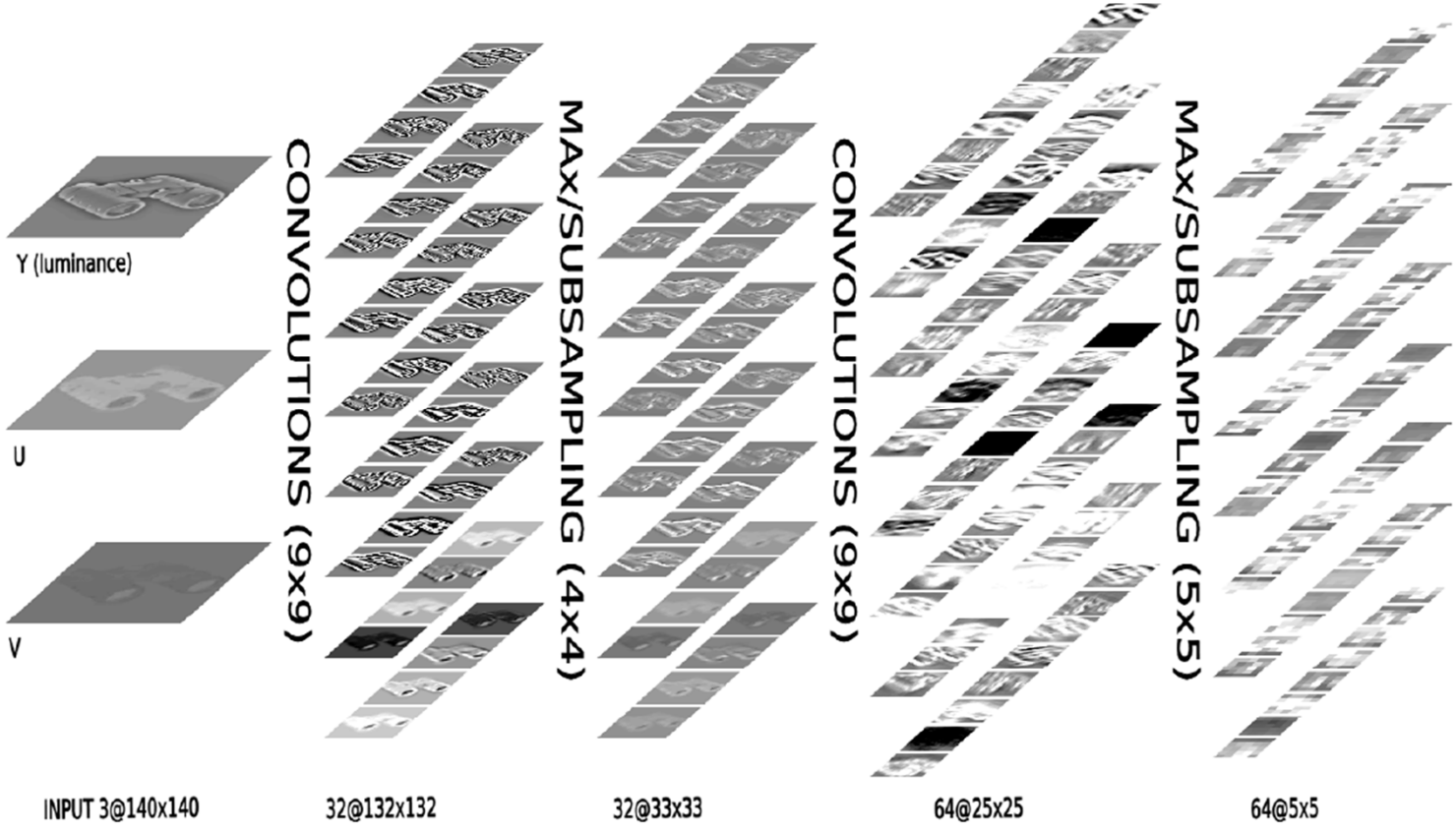




☯By reducing the spatial resolution of the feature map, reduce the effect of noises and shift or distortion.

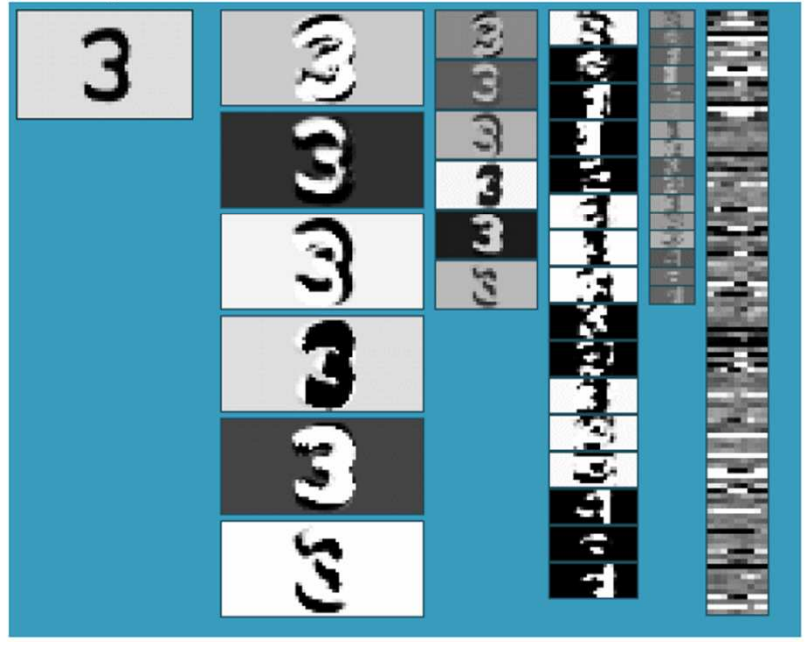
☯the weight sharing is also applied in subsampling layers

Convolutional network architecture

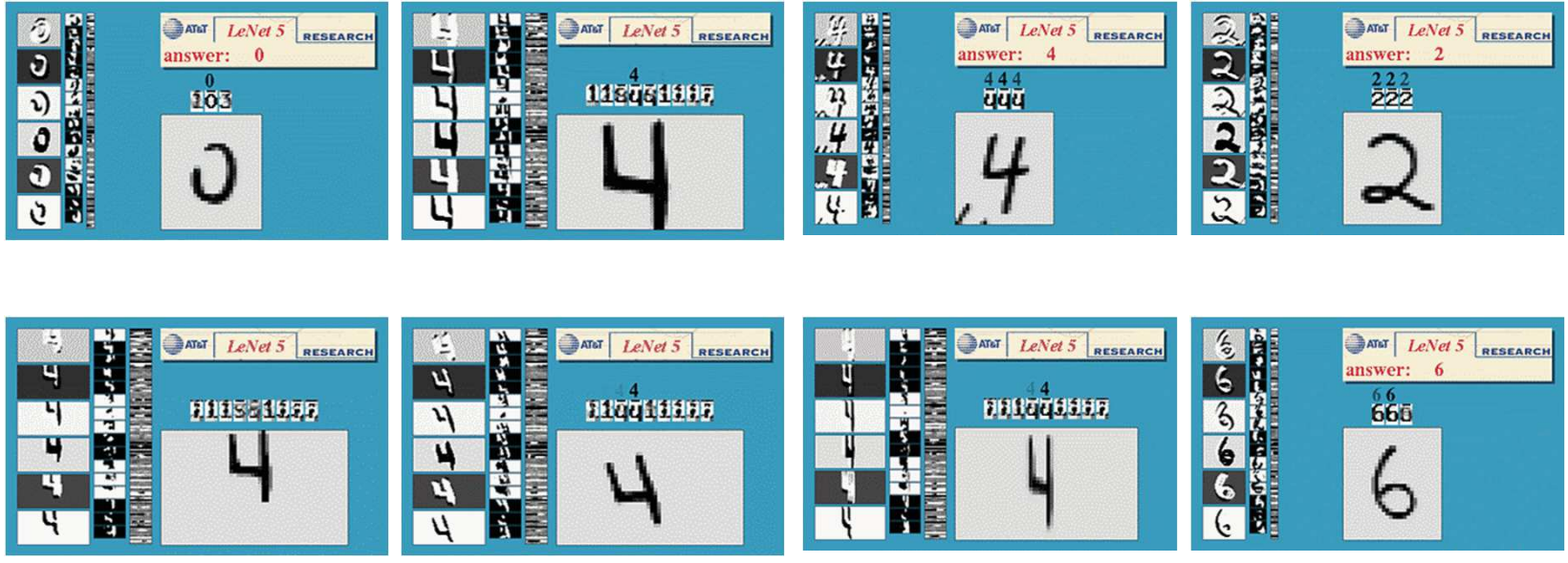


Convolutional Nets and other multistage Hubel-Wiesel architecture

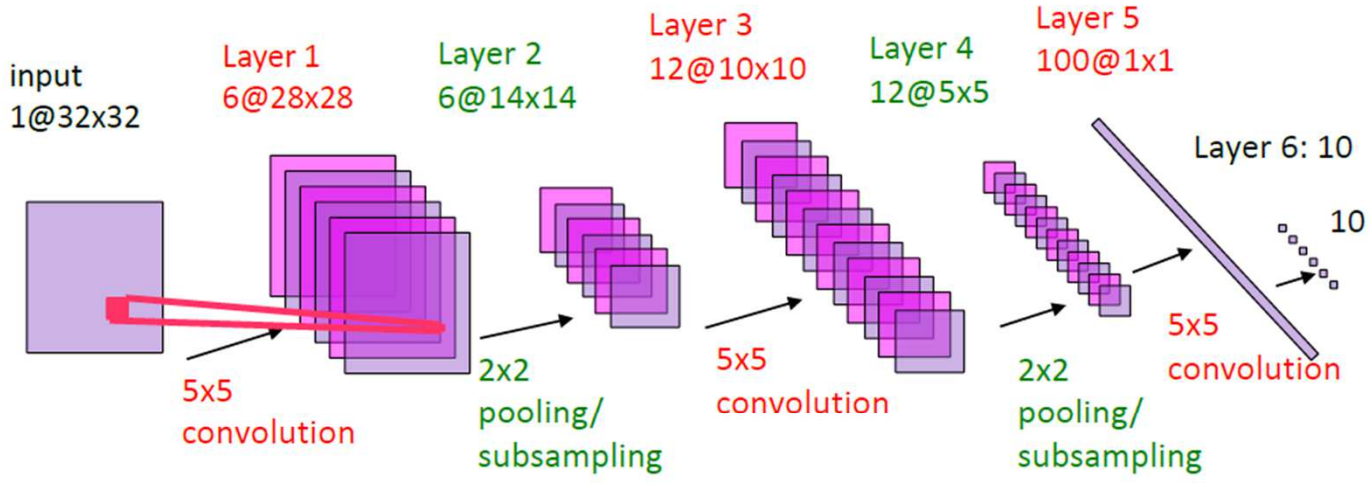
- Building a complete artificial vision system:
  - Stack multiple stages of simple cells / complex cells layers
  - Higher stages compute more global, more invariant features
  - Stick a classification layer on top
  - [Fukushima 1971-1982]
    - neocognitron
  - [LeCun 1988-2007]
    - convolutional net
  - [Poggio 2002-2006]
    - HMAX
  - [Ullman 2002-2006]
    - fragment hierarchy
  - [Lowe 2006]
    - HMAX
- **QUESTION: How do we find (or learn) the filters?**



Convolutional networks architecture for hand-writing recognition

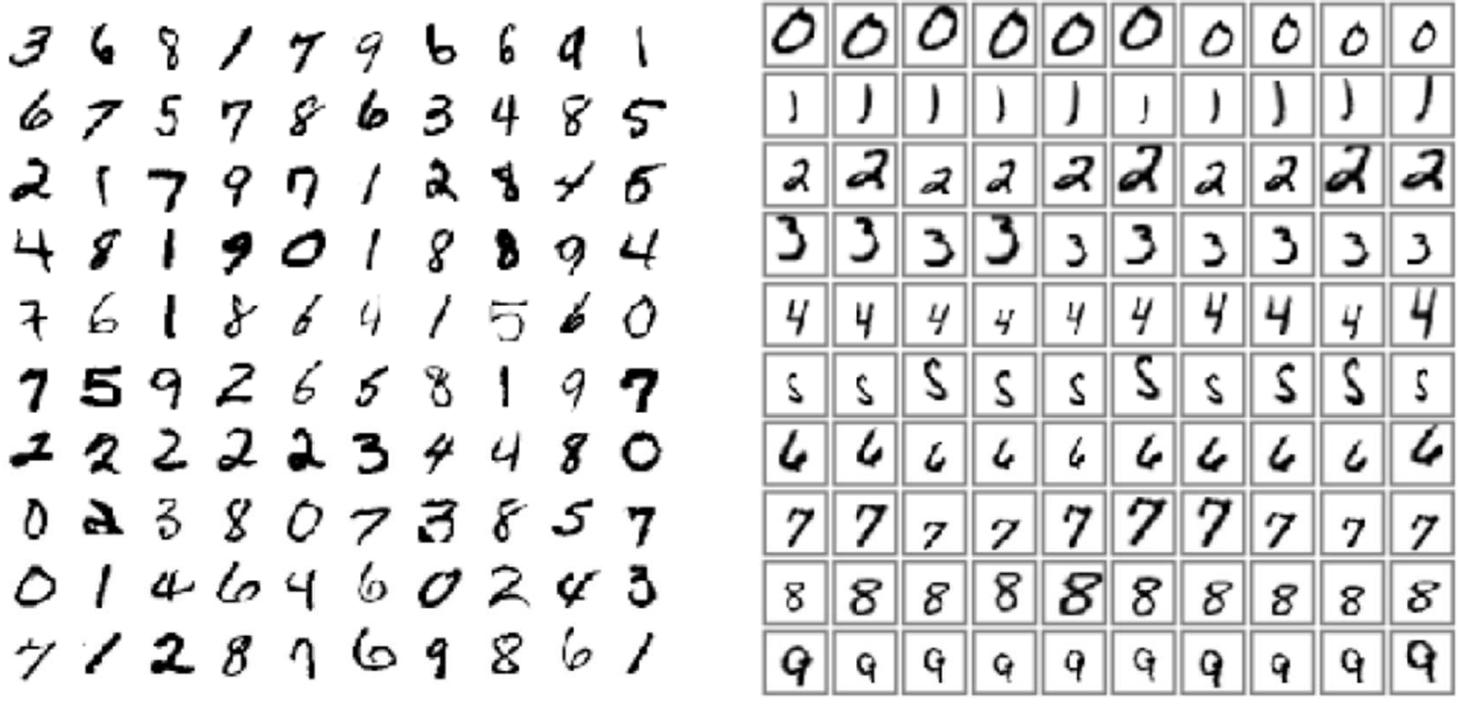


Convolutional networks architecture for hand-writing recognition



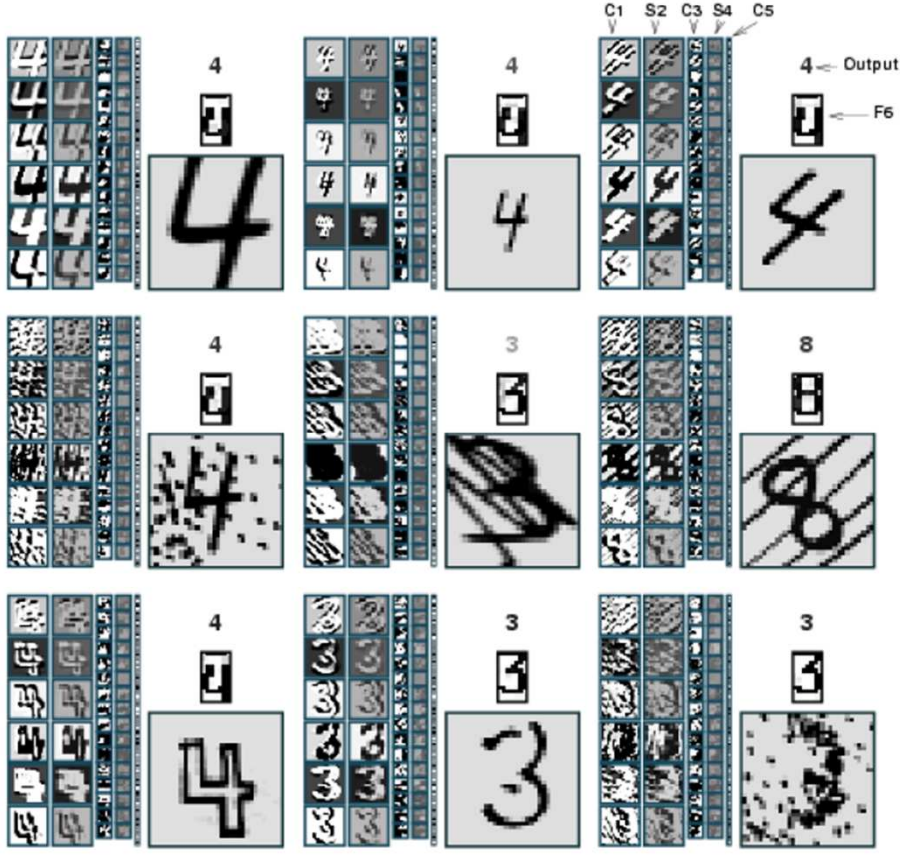
- **Convolutional net for handwriting recognition** (400,000 synapses)
- Convolutional layers (simple cells): all units in a feature plane share the same weights
- Pooling/subsampling layers (complex cells): for invariance to small distortions.
- **Supervised gradient-descent learning using back-propagation**
- **The entire network is trained end-to-end. All the layers are trained simultaneously.**
- [LeCun et al. Proc IEEE, 1998]

MNIST handwritten digit dataset

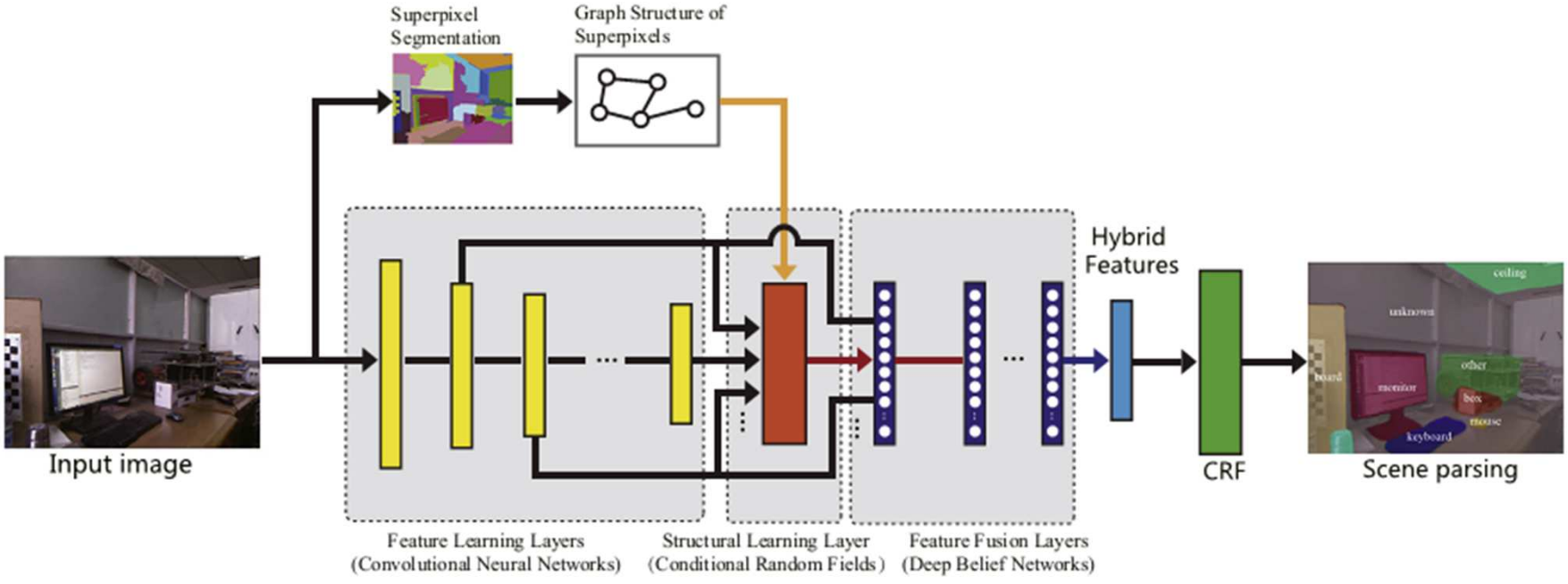


- Handwritten Digit Dataset MNIST: 60,000 training samples, 10,000 test samples

Invariance and robustness to noise

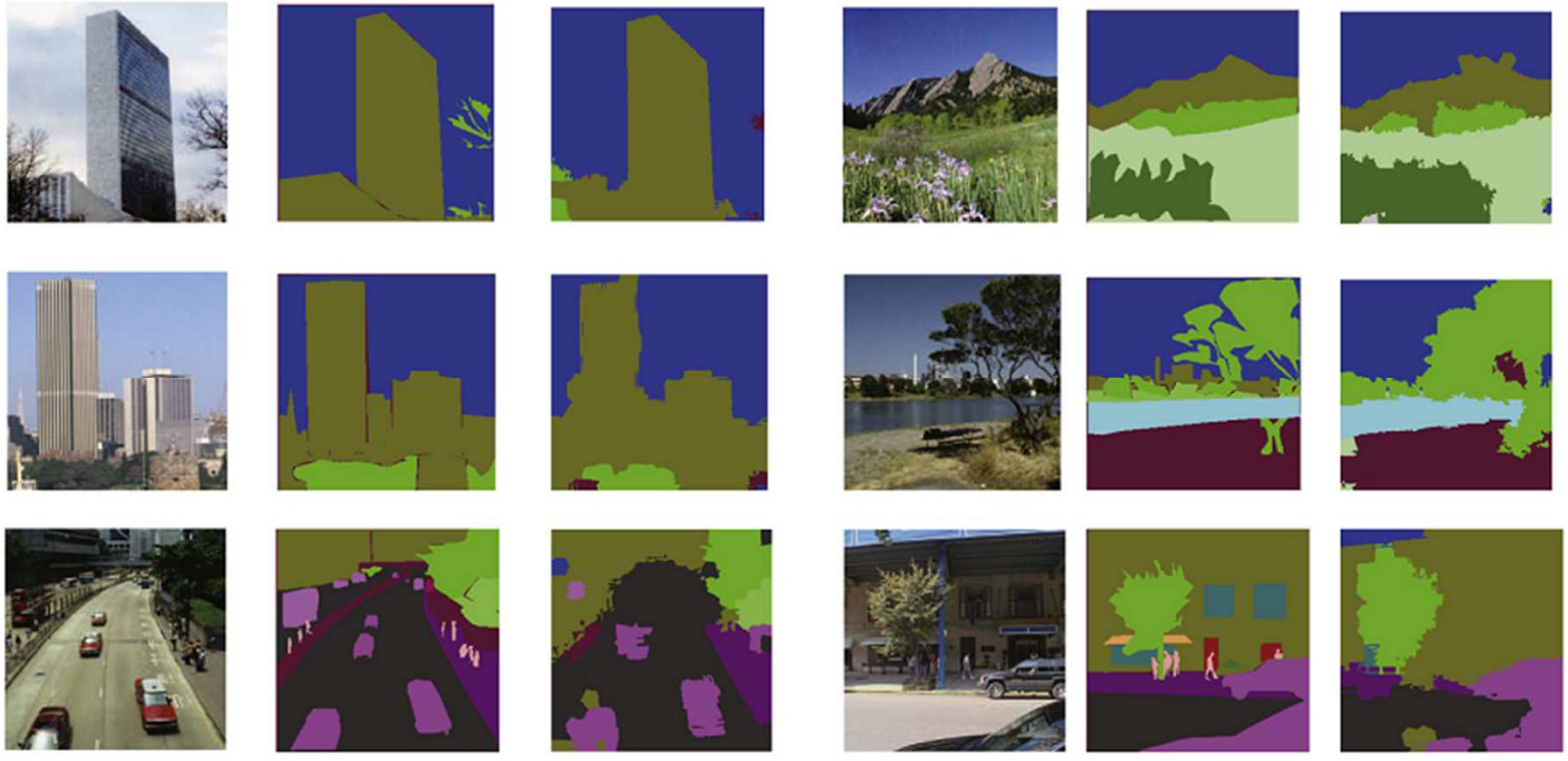


Scene Parsing



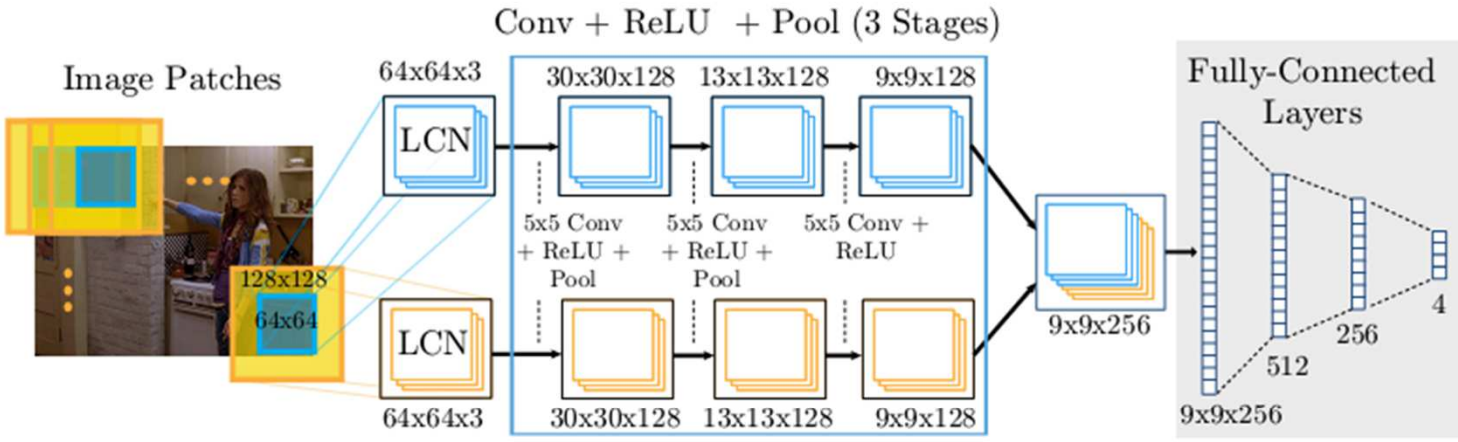
Scene parsing using inference Embedded Deep Networks

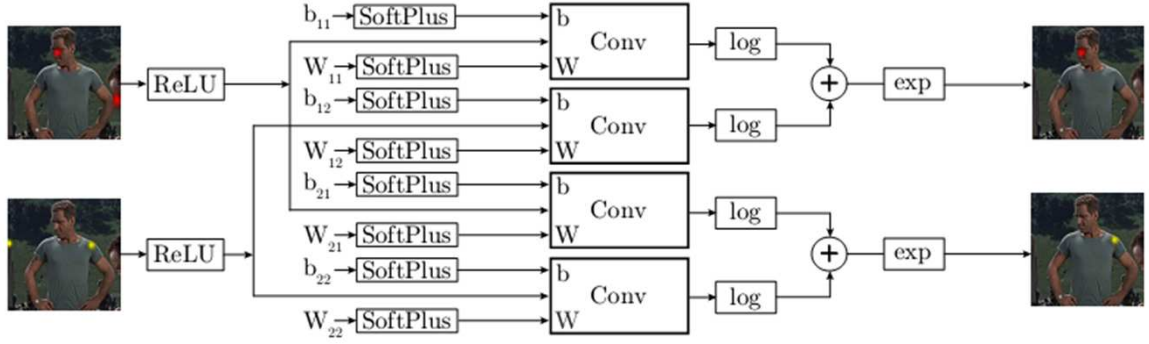
Scene Parsing



Scene parsing using inference Embedded Deep Networks

Human Pose Estimation





Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation

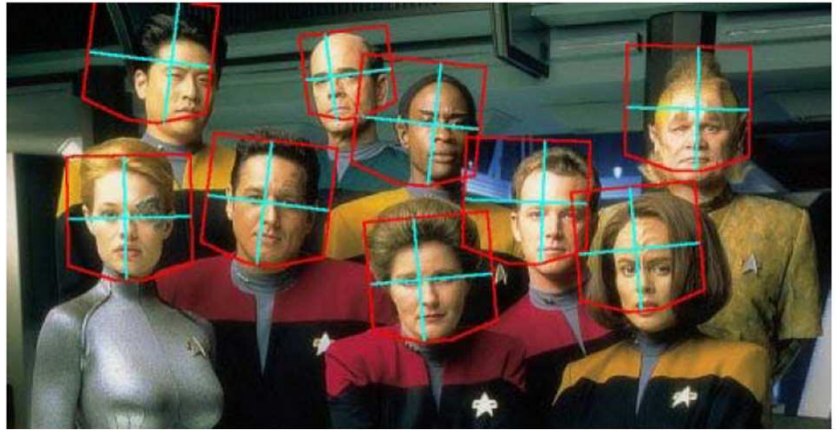
Human Pose Estimation

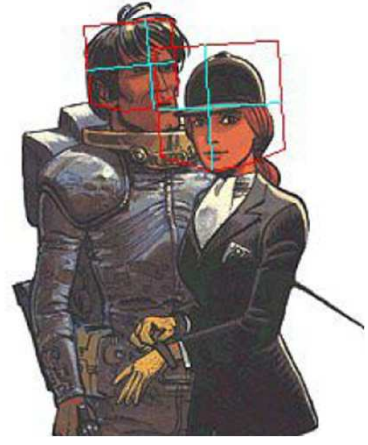
Joint Training of a Convolutional Network and a Graphical Model for Human Pose Estimation

Face detection and pose estimation: results

Face detection: results

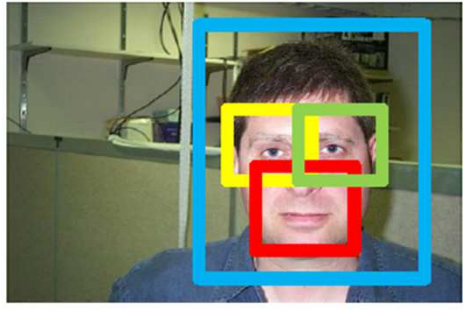
Data Set->	TILTED		PROFILE		MIT+CMU	
False positives per image->	4.42	26.9	0.47	3.36	0.5	1.28
Our Detector	90%	97%	67%	83%	83%	88%
Jones & Viola (tilted)	90%	95%	x		x	
Jones & Viola (profile)	x		70%	83%	x	

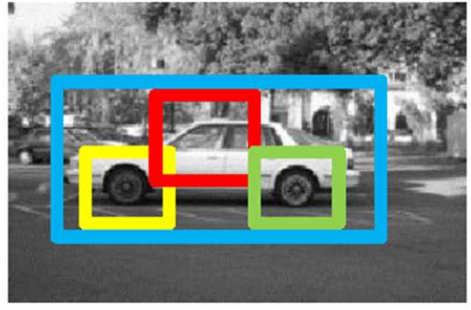




Learning object representations

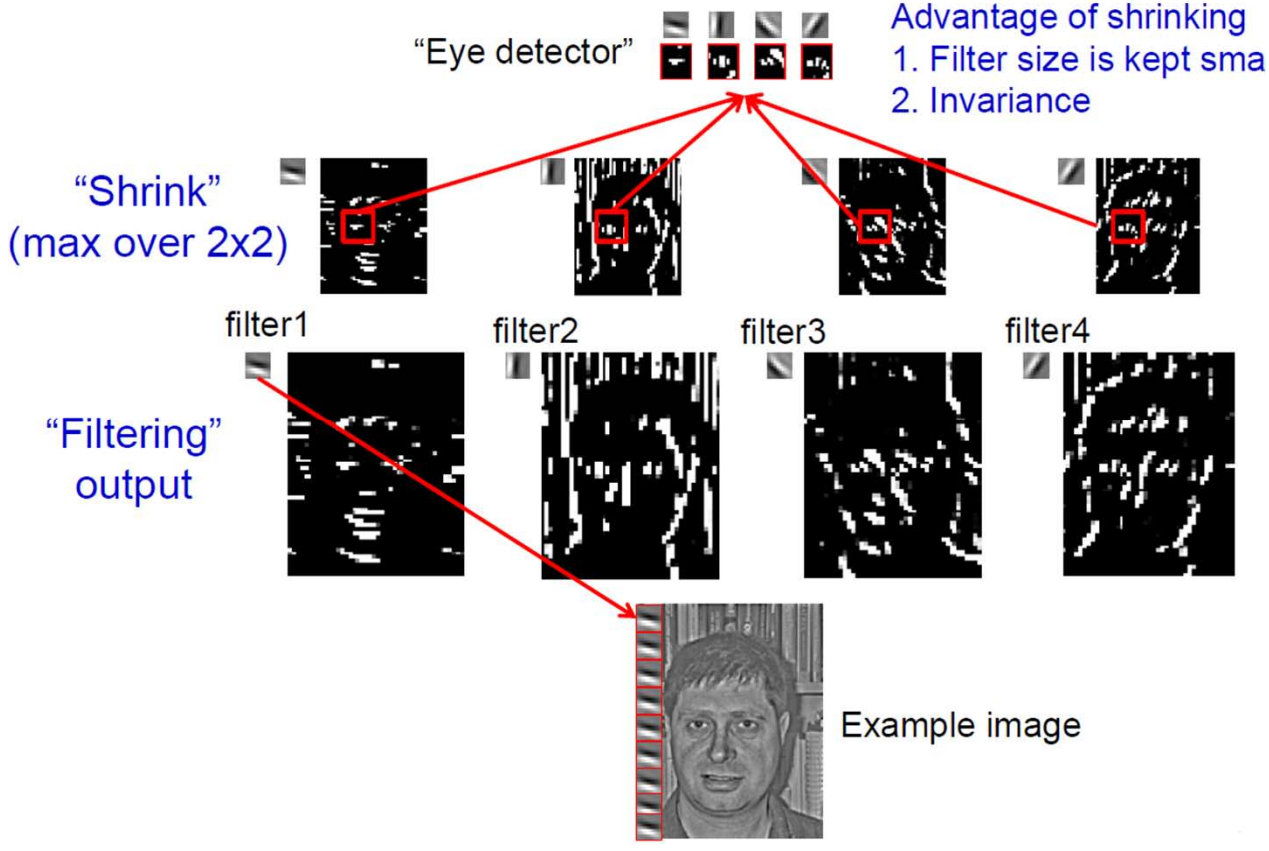
- Learning objects and parts in images





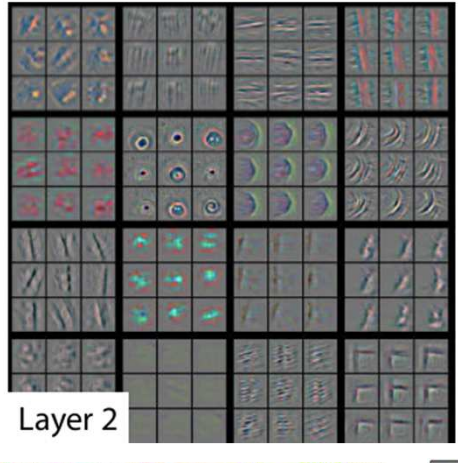
- Large image patches contain interesting higher-level structures.
  - E.g., object parts and full objects

Illustration: learning an “eye” detector

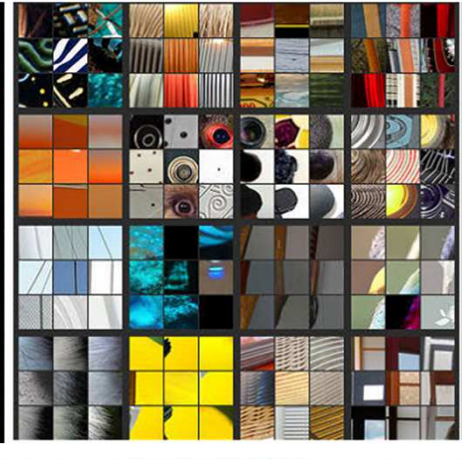


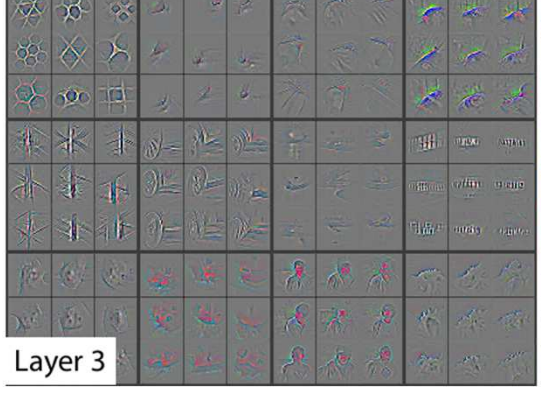
Learning object representations

Layer 1

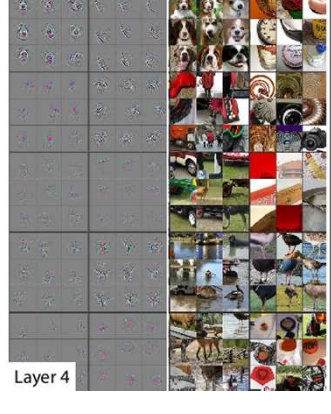


Layer 2





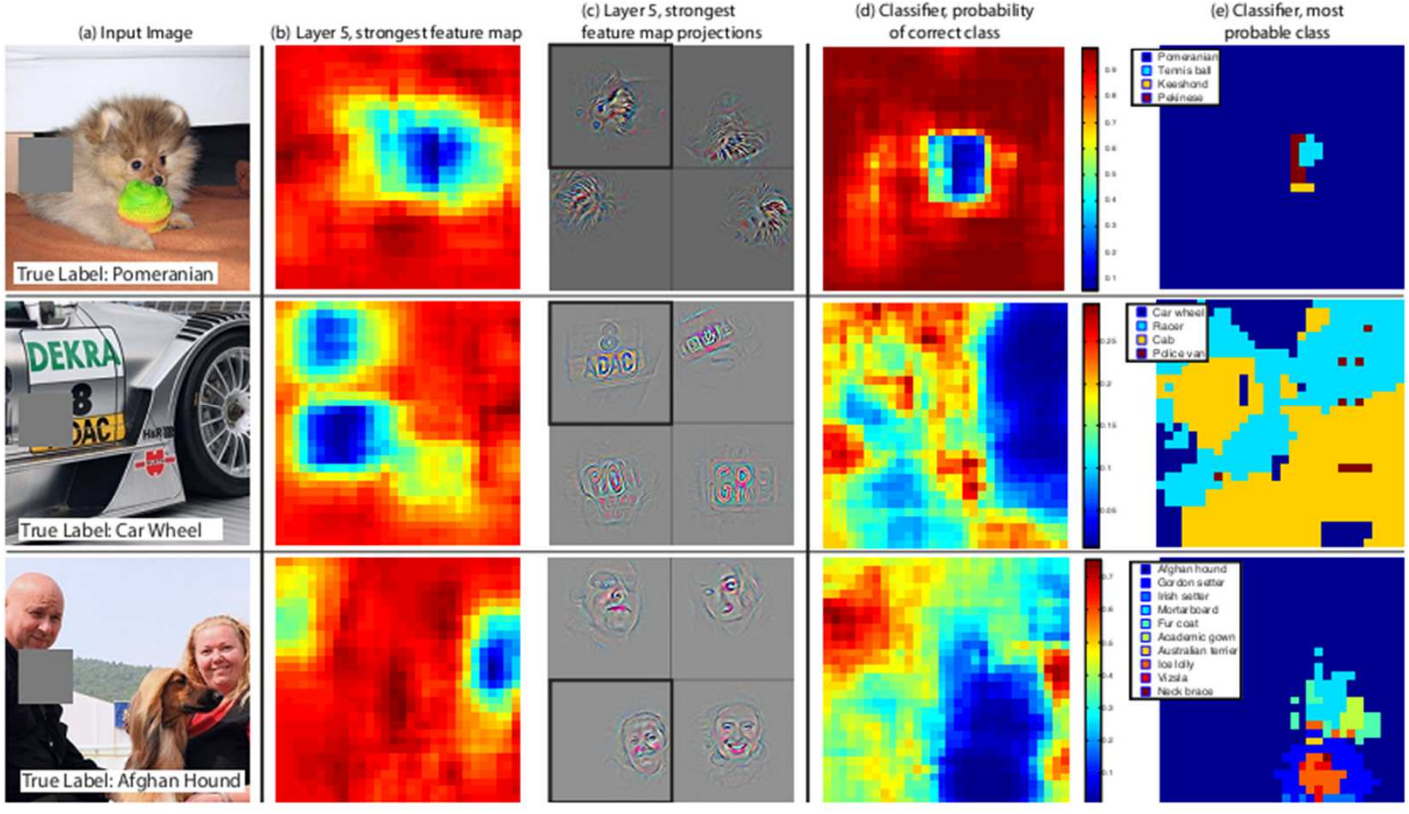
Layer 3



Layer 4

Layer 5

Learning object representations



Disadvantages

- At runtime the convolution operations are computationally expensive and take up about 67% of the time.
- Learned model can not adapt to other domain.
- Need large amount training samples.
- Without memory capability
- Need explicit structure inference