

Convolutional Neural Networks



Dr. Elli Angelopoulou

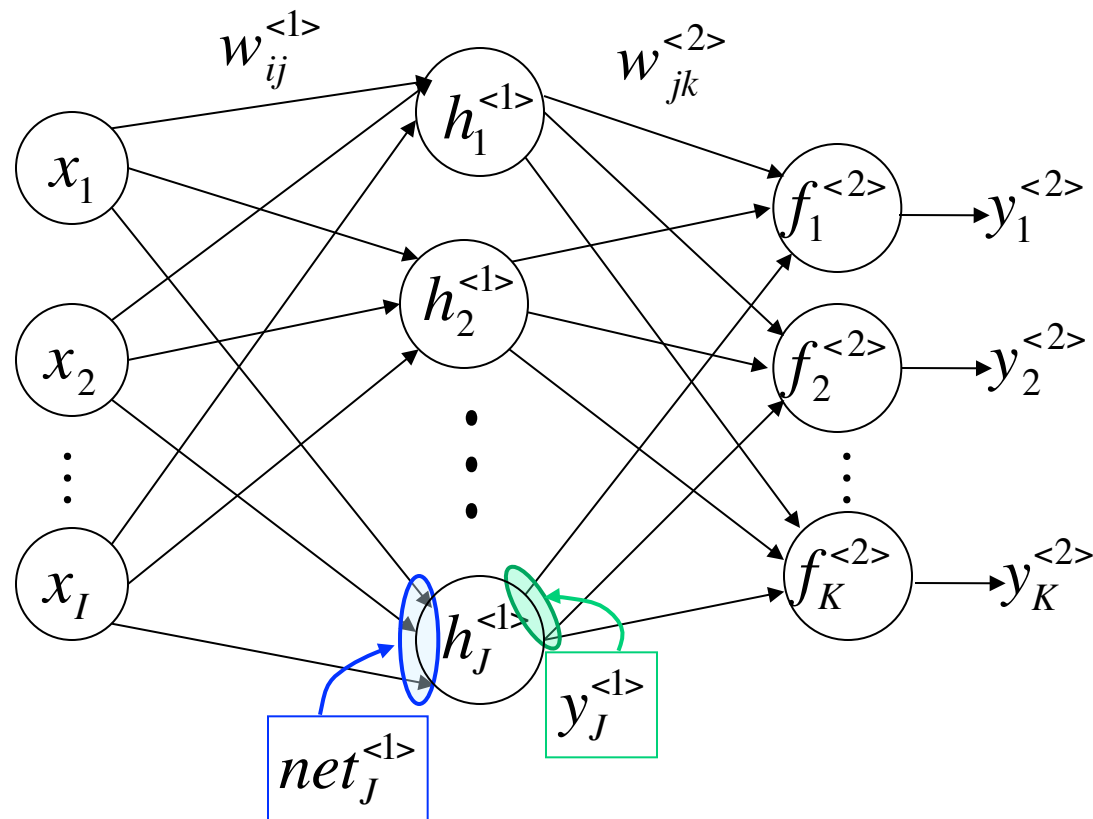
Lehrstuhl für Mustererkennung (Informatik 5)

Friedrich-Alexander-Universität Erlangen-Nürnberg



A Simple MLP

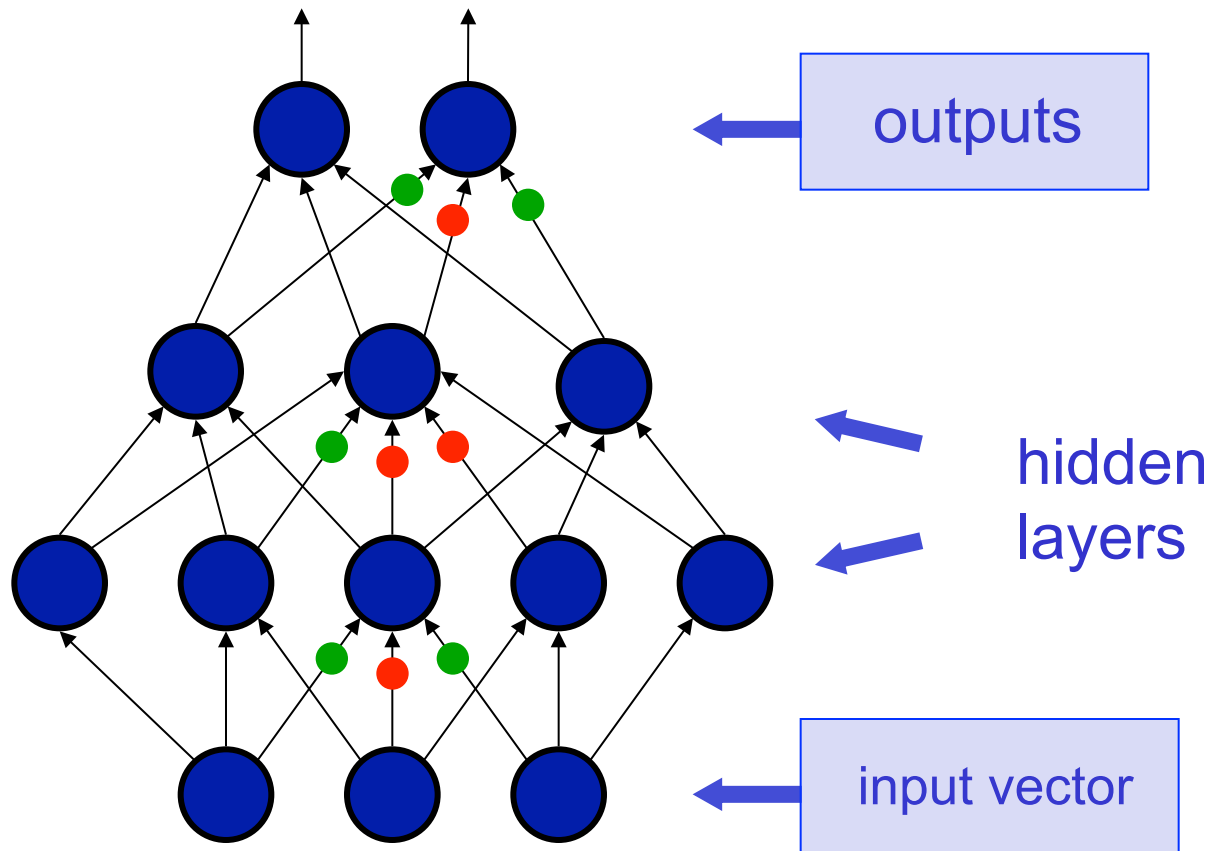
- Multilayer Perceptrons were popular in the 1980s.
- Could solve non-linear problems.
- Were trained via back-propagation.





Multilayer MLPs

- The use of multiple hidden layers is feasible.
- They can be trained via back-propagation.
- Learning time does not scale well with multiple hidden layers.





What Next?

- How do we overcome the training limitations of single MLP?
- Two approaches:
 - Convolutional Neural Networks (1998)
Influenced by additional observations that on certain types of neurons in biological systems, the topological layout of the stimuli is also exploited.
 - Deep Belief Networks (2000)
Based on the observations of Judea Pearl that even if two hidden causes are independent, then can become dependent when we observe an effect that they can both influence.



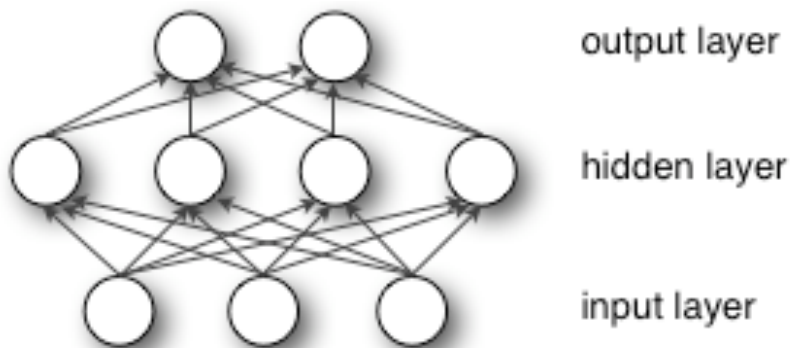
What Next?

- How do we overcome the training limitations of single MLP?
- Two approaches:
 - Convolutional Neural Networks
 - Influenced by additional observations that on certain types of neurons in biological systems, the topological layout of the stimuli is also exploited.
 - Deep Belief Networks
 - Based on the observations of Judea Pearl that even if two hidden causes are independent, then can become dependent when we observe an effect that they can both influence.



Convolutional Neural Networks

- Convolutional Neural Networks (CNN) are considered a variant of Multilayer Perceptrons.
- The biggest difference lies in the connectivity between layers.
- A hidden node in layer m , is only connected to a *local* subset of spatially contiguous nodes in layer $m-1$.

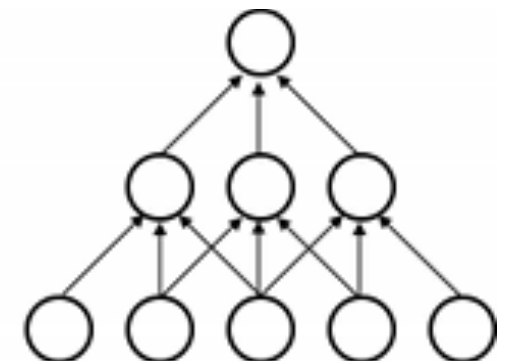


MLP

layer $m+1$

layer m

layer $m-1$



CNN



Regional Response

- It has been known since 1968 that particularly in the visual system (visual cortex) some cells are sensitive to small sub-regions of the input space.
- These small sub-regions (receptive fields) are tiled in such a way that they cover the entire visual field.
- The cells that respond to these local receptive fields act as local filters.
- This local filter operation is well-suited for exploiting:
 - Topological characteristics of the input signal
 - Spatially local correlations that are present in natural images



Connectivity of CNNs and Receptive Fields

- For an intuitive example, let layer $m-1$ be the retina, the light sensitive tissue lining the inner surface of the eye.
- The nodes in layer m have receptive fields (operate on local regions) of width 3.
- They nodes in layer m are connected to only 3 adjacent neurons in layer $m-1$.

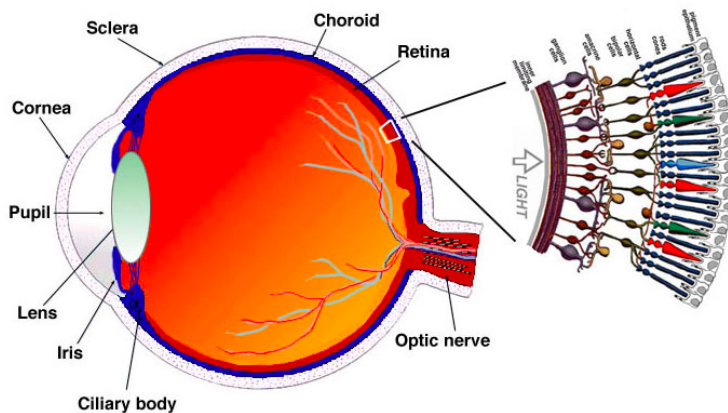
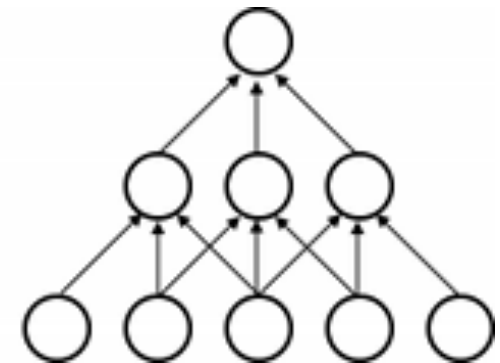


Fig. 1.1. A drawing of a section through the human eye with a schematic enlargement of the retina.

layer $m+1$

layer m

layer $m-1$

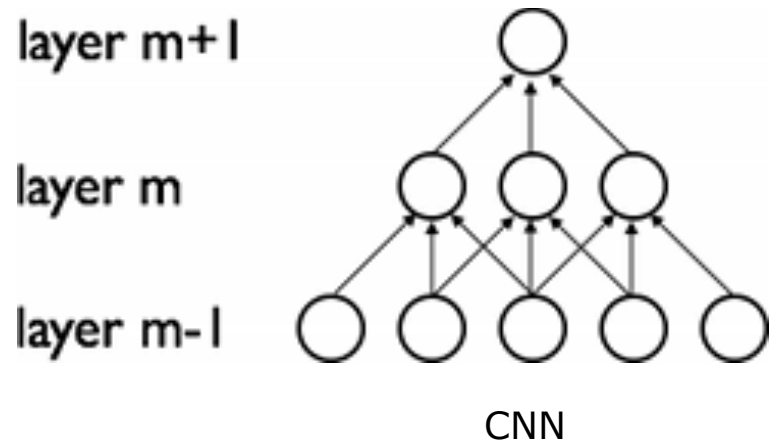


CNN

Connectivity of CNNs and Receptive Fields



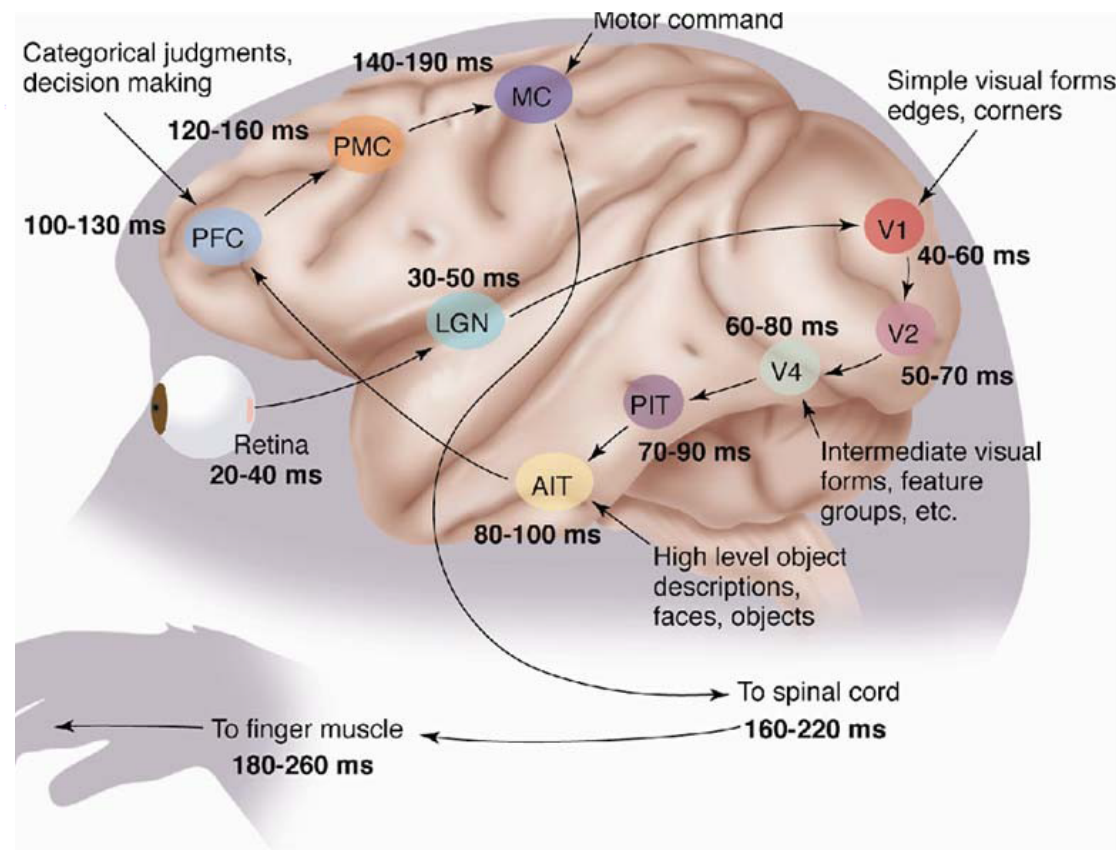
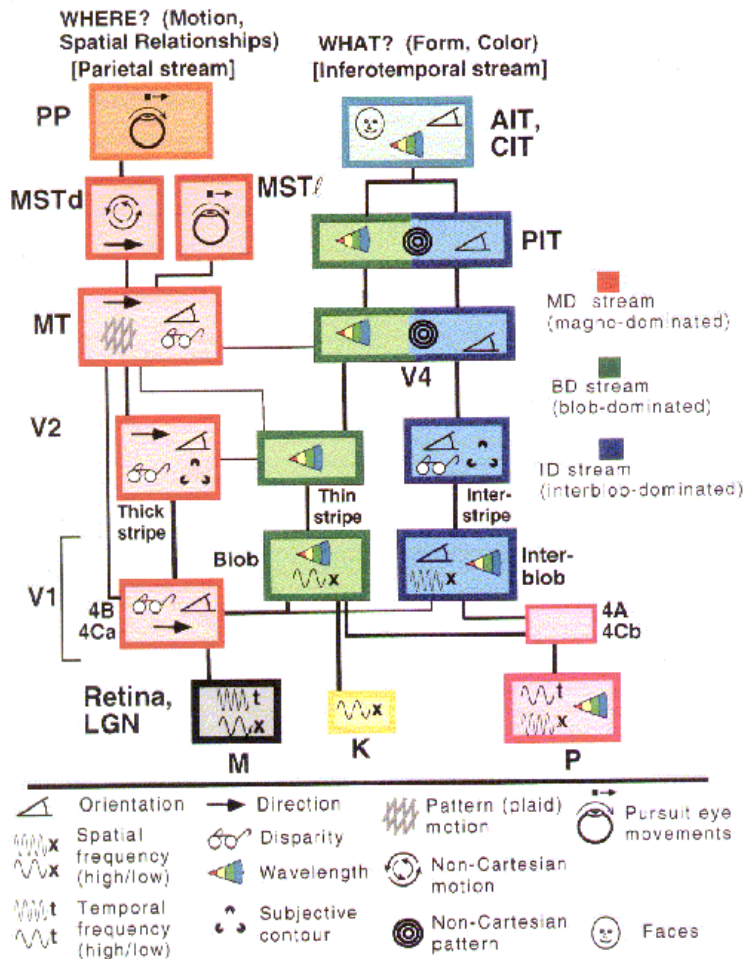
- The nodes in layer m are local filters that operate on 3-elt wide regions.
- During training, the CNN learns the best filters, the filters that produce the strongest response.
- Stacking such layers leads to increasingly larger filters.
- Layer $m+1$ encodes a non-linear filter (feature) which operates on receptive fields of width.





Second Motivation Behind CNN

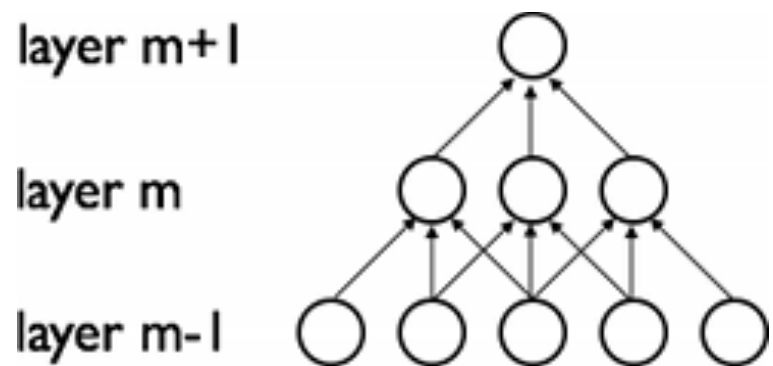
- The recognition pathway in the visual cortex has multiple stages.





Deep Convolutional Networks

- By stacking multiple hidden layers, where each node has only limited regional support, one creates a stack of filters each operating over a larger region.
- At layer $m+1$ information is shared on disjoint computations at layer m .
- Each successive layer computes a more complex function/feature
- Each successive layer introduces a higher level of abstraction.



Learning Representations and Classifiers

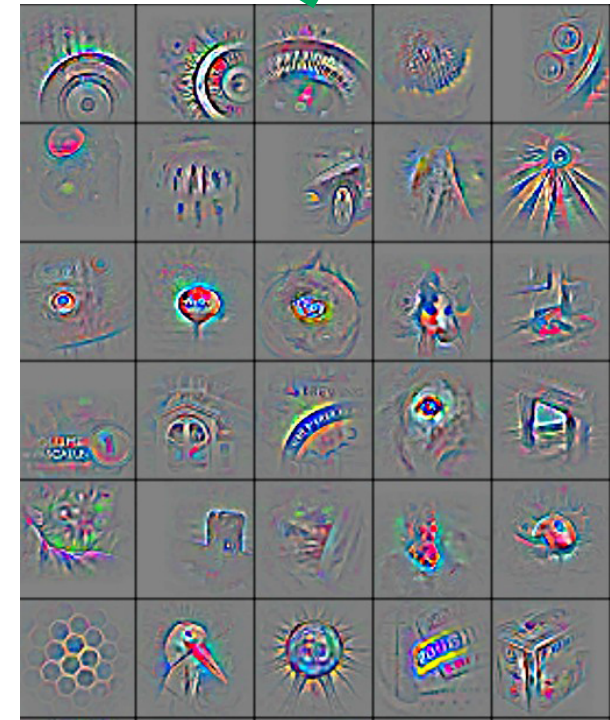
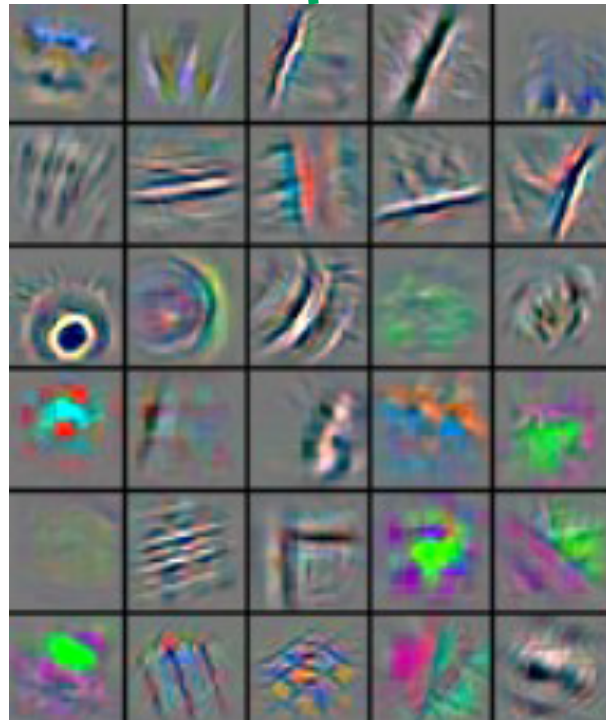
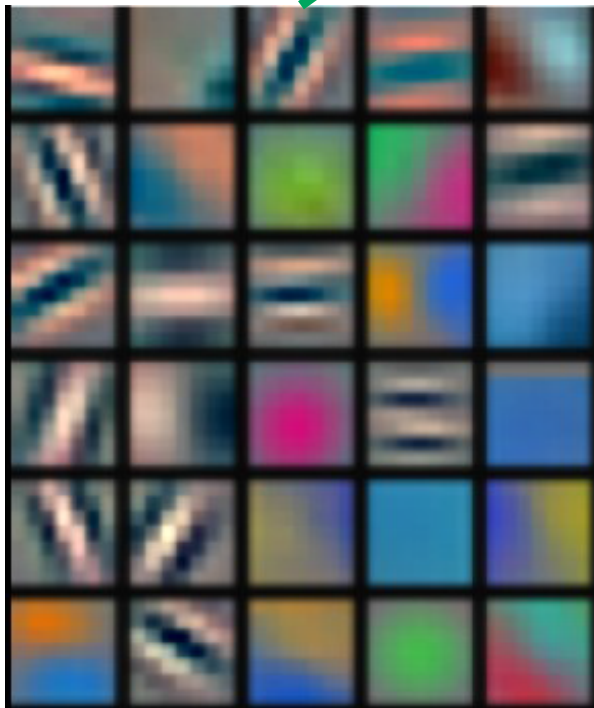


Low-Level
Features

Mid-level
Features

High-level
Features

Classification



Key Concepts of Convolutional Nets



- Local Connectivity
- Shared Weights
- Convolutional Layer
- Pooling Layer

Key Concepts of Convolutional Nets

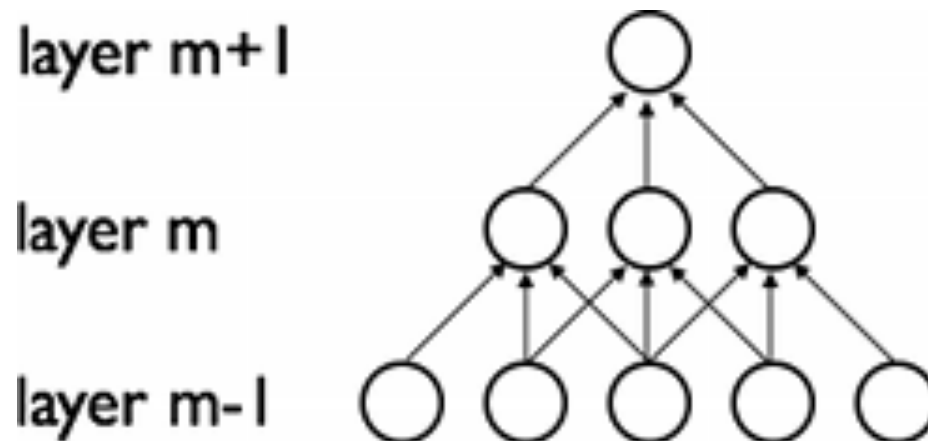


- Local Connectivity
- Shared Weights
- Convolutional Layer
- Pooling Layer

Local Connectivity



- Allows for the gradual introduction of levels of abstraction.
- Exploits topological cues in input.



Key Concepts of Convolutional Nets

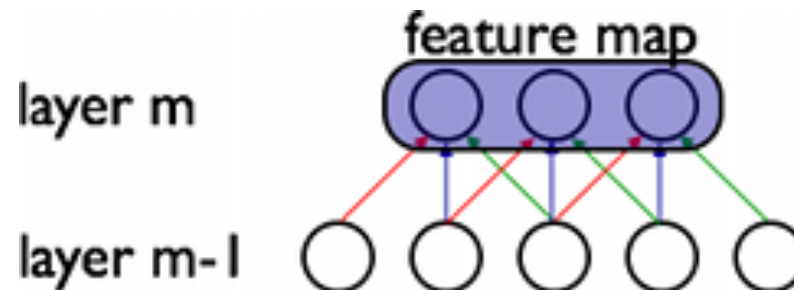


- Local Connectivity
- Shared Weights
- Convolutional Layer
- Pooling Layer



Shared Weights

- Each node (filter) h_i is replicated across the entire visual field.
- The replicated local nodes form a feature map.
- All nodes in a feature map have the same weight vector.
- Hence each node performs the same operation in different local regions.
- Easier to train via gradient descent.



Key Concepts of Convolutional Nets



- Local Connectivity
- Shared Weights
- Convolutional Layer
- Pooling Layer



Convolutional Layer

- Recall that we can apply LSI filters via convolution:

$$R(x, y) = I * H = \sum_{i=-k}^k \sum_{j=-k}^k I(x-i, y-j)H(i, j)$$

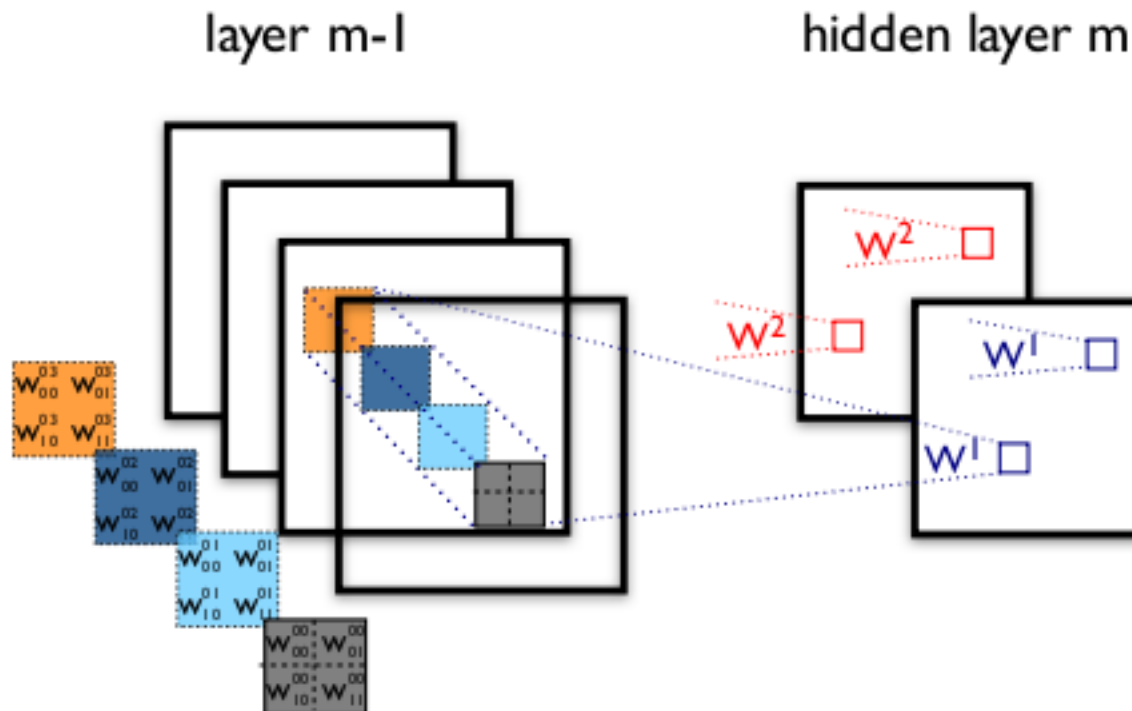
- One can think of the filter kernel H as a matrix of weights w_{ij} that are used to compute a weighted sum.
- A CNN node can then compute a feature map by locally convolving the input image with an LSI filter, adding a bias and then applying a non-linear activation function.

$$h_{ij}^m = \tanh\left((W^m * x)_{ij} + b_m\right)$$



Convolutional Layer Example

- The figure shows two convolutional layers. Layer $m-1$ contains 4 feature maps. Layer m contains 2 feature maps.



Key Concepts of Convolutional Nets



- Local Connectivity
- Shared Weights
- Convolutional Layer
- Pooling Layer



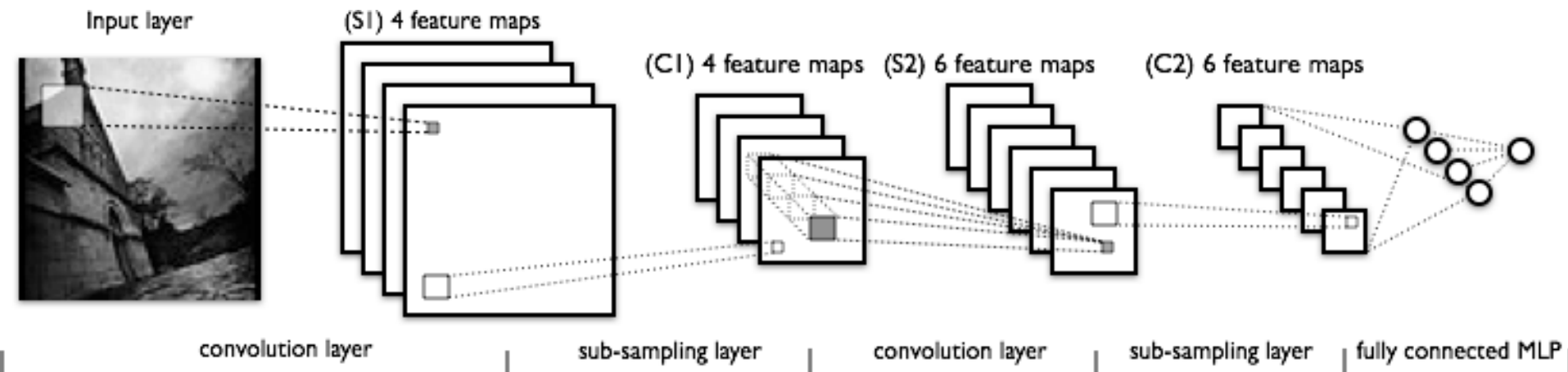
Pooling Layer

- Consecutive convolutional layers, each multiple feature maps can result in a very computationally expensive network.
- Employ a 2nd type of layer, a *pooling layer*, which performs non-linear down-sampling.
- Most widely used is the max-pooling method.
- Max-pooling partitions the input image into a set of non-overlapping rectangles and outputs the maximum value of the rectangle.
- Interleave pooling layers and convolution layers.



Full Model of CNN

- Typically a Deep Convolutional Neural Network will then have the following form:

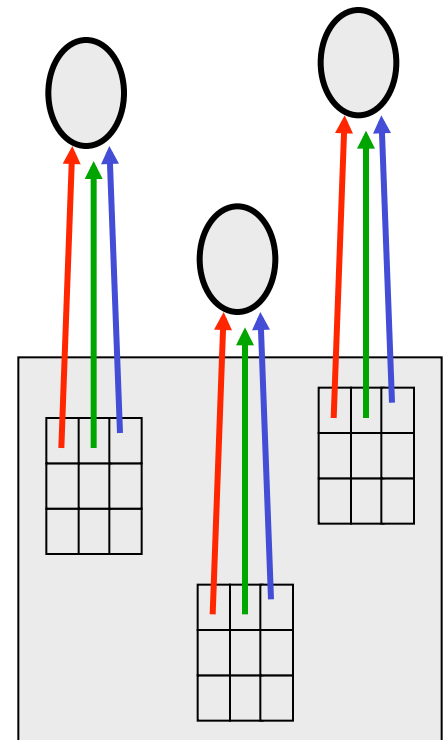




Replicating Features

- Use many different copies of the same feature detector with different positions.
 - Could also replicate across scale and orientation (tricky and expensive)
 - Replication **greatly reduces the number of free parameters** to be learned.
- Use several different feature types, each with its own map of replicated detectors.
 - Allows each patch of image to be represented in several ways.

Connections of the same color have the same weight.





Back-propagation with Weight Constraints

- The back-propagation algorithm must be adapted in order to incorporate linear constraints between the weights.
- We compute the gradients as usual, and then modify the gradients so that they satisfy the constraints.
 - So if the weights started off satisfying the constraints, they will continue to satisfy them.

To constrain: $w_1 = w_2$
 we need: $\Delta w_1 = \Delta w_2$

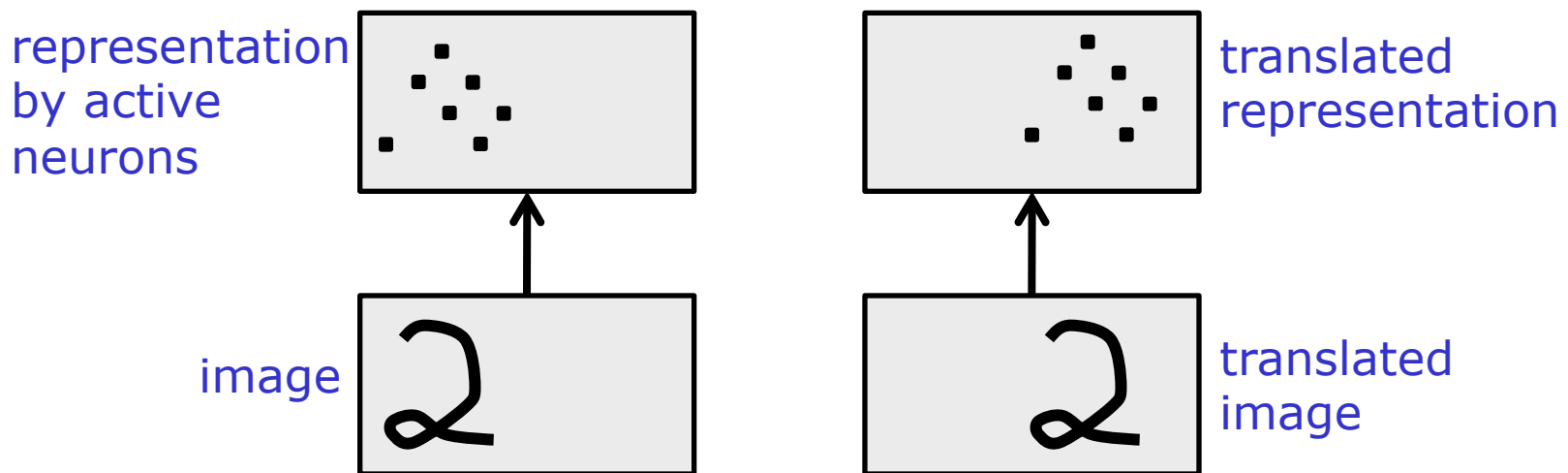
compute: $\frac{\partial E}{\partial w_1}$ and $\frac{\partial E}{\partial w_2}$

use $\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial w_2}$ for w_1 and w_2



Why Feature Replication

- **Equivariant activities:** Replicated features do **not** make the neural activities invariant to translation. The activities are equivariant.



- **Invariant knowledge:** If a feature is useful in some locations during training, detectors for that feature will be available in all locations during testing.



Why Pooling

- Get a small amount of translational invariance at each level by averaging four neighboring replicated detectors to give a single output to the next level.
 - This reduces the number of inputs to the next layer of feature extraction, thus allowing us to have many more different feature maps.
 - Taking the maximum of the four works slightly better.
- **Problem:** After several levels of pooling, we have lost information about the precise positions of things.
 - This makes it impossible to use the precise spatial relationships between high-level parts for recognition.

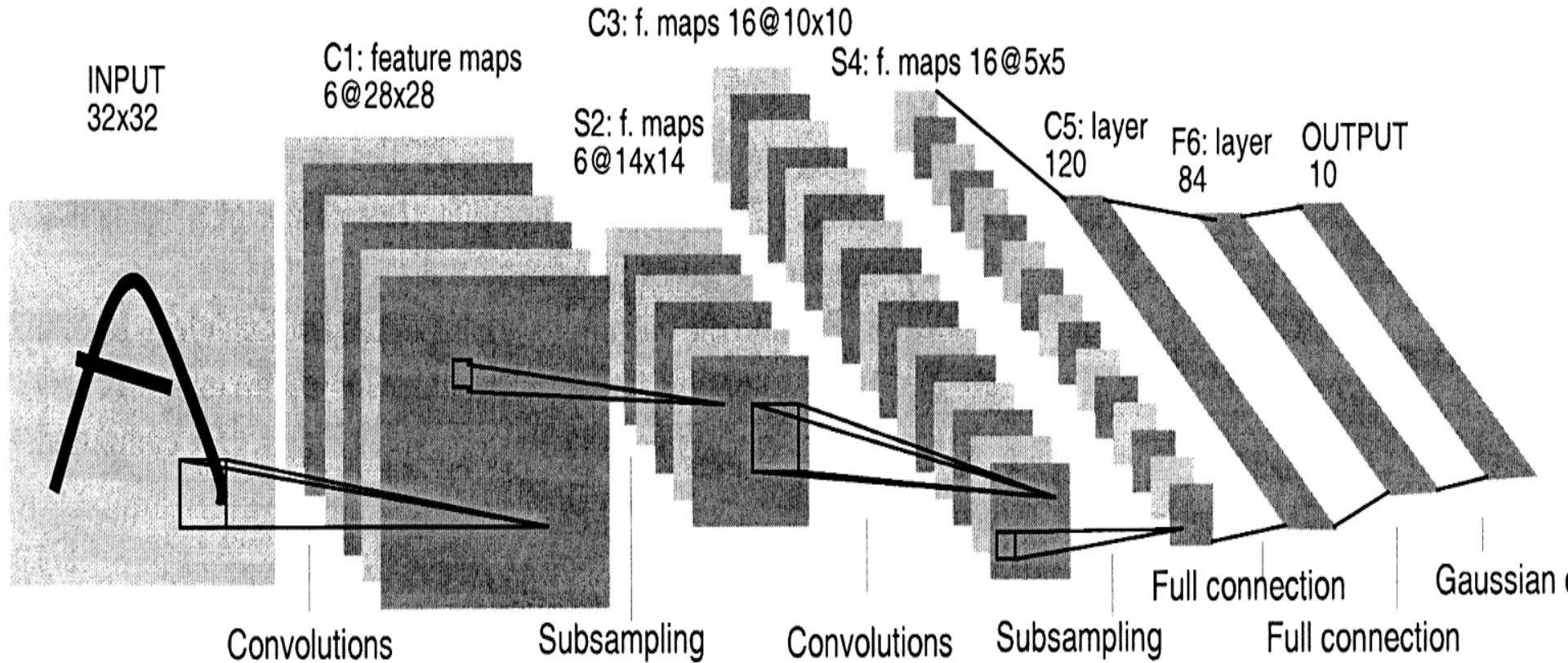
Le Net



- Yann LeCun and his collaborators developed a really good recognizer for handwritten digits by using back-propagation in a feed-forward net with:
 - Many hidden layers
 - Many maps of replicated units in each layer.
 - Pooling of the outputs of nearby replicated units.
 - A wide net that can cope with several characters at once even if they overlap.
 - A clever way of training a complete system, not just a recognizer.
- This net was used for reading $\sim 10\%$ of the checks in North America.
- Demos of LENET can be found at <http://yann.lecun.com>



The Architecture of LeNet5





The 82 Errors of LeNet5



Notice that most of the errors are cases that people find quite easy.

The human error rate is probably 20 to 30 errors but nobody has had the patience to measure it.



Priors and Prejudice

- We can put our prior knowledge about the task into the network by designing appropriate:
 - Connectivity.
 - Weight constraints.
 - Neuron activation functions
- This is less intrusive than hand-designing the features.
 - But it still prejudices the network towards the particular way of solving the problem that we had in mind.
- Alternatively, we can use our prior knowledge to create a whole lot more training data.
 - This may require a lot of work (Hofman&Tresp, 1993)
 - It may make learning take much longer.
- It allows optimization to discover clever ways of using the multi-layer network that we did not think of.
 - And we may never fully understand how it does it.

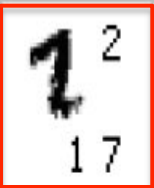
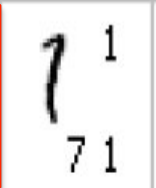


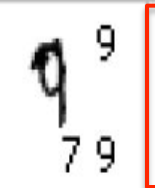

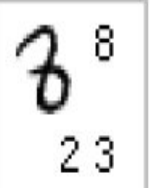
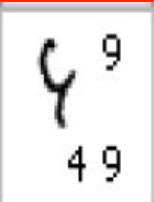



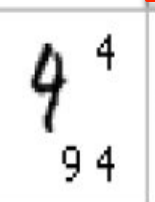

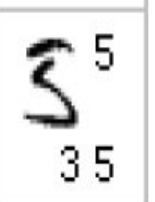





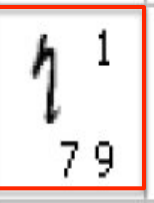
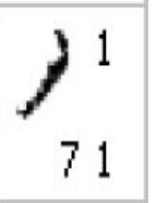


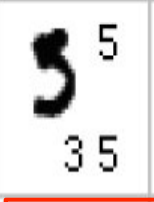

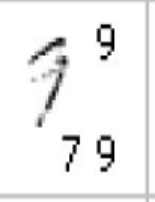
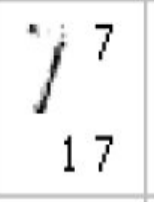
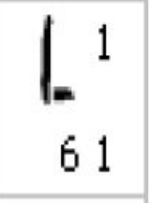
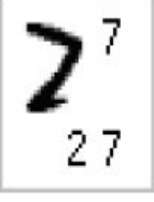


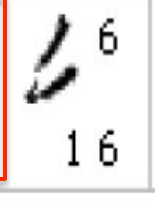

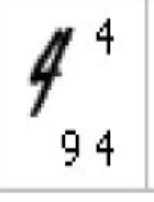
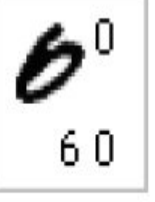


The Brute Force Approach

- LeNet uses knowledge about the invariances to **design**:
 - the local connectivity
 - the weight-sharing
 - the pooling.
- This achieves about 80 errors.
 - This can be reduced to about 40 errors by using many different transformations of the input and other tricks (Ranzato 2008)
- Ciresan et al. (2010) inject knowledge of invariances by creating a huge amount of carefully designed extra training data:
 - For each training image, they produce many new training examples by applying many different transformations.
 - They can then train a large, deep, dumb net on a GPU without much overfitting.
- They get about 35 errors.



The Errors by the Ciresan et al. Net

 2 17	 1 71	 8 98	 9 59	 9 79	 5 35	 8 23
 9 49	 5 35	 4 97	 9 49	 4 94	 2 02	 5 35
 6 16	 4 94	 0 60	 6 06	 6 86	 1 79	 1 71
 9 49	 0 50	 5 35	 8 98	 9 79	 7 17	 1 61
 7 27	 8 58	 2 78	 6 16	 5 65	 4 94	 0 60

The top printed digit is the right answer. The bottom two printed digits are the network's best two guesses.

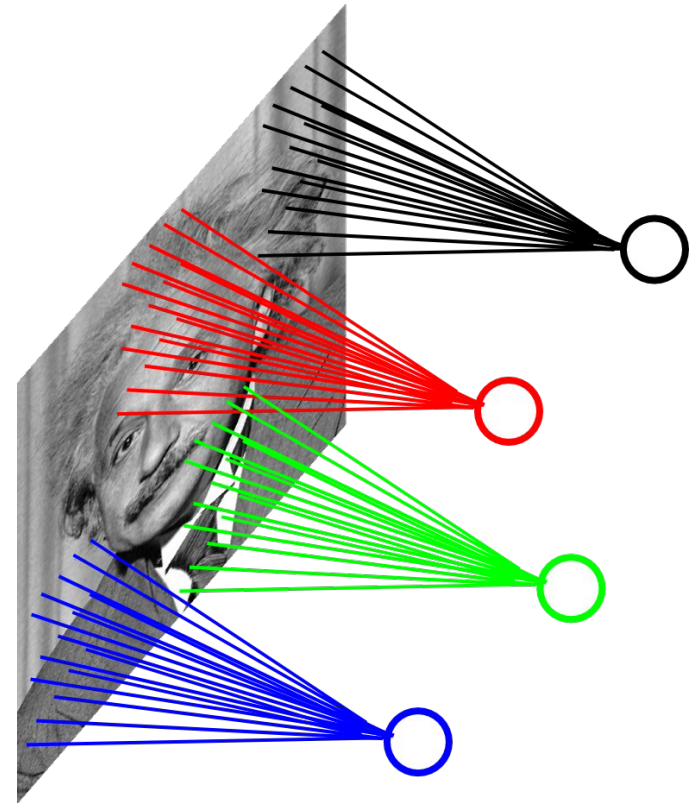
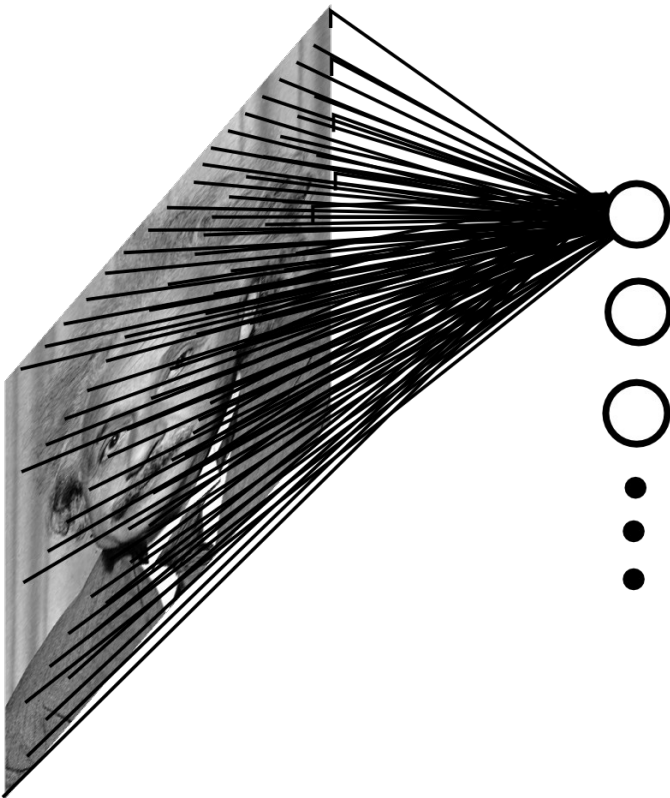
The right answer is **almost** always in the top 2 guesses.

With model averaging they can now get about 25 errors.



Traditional MLPs vs. CNNs

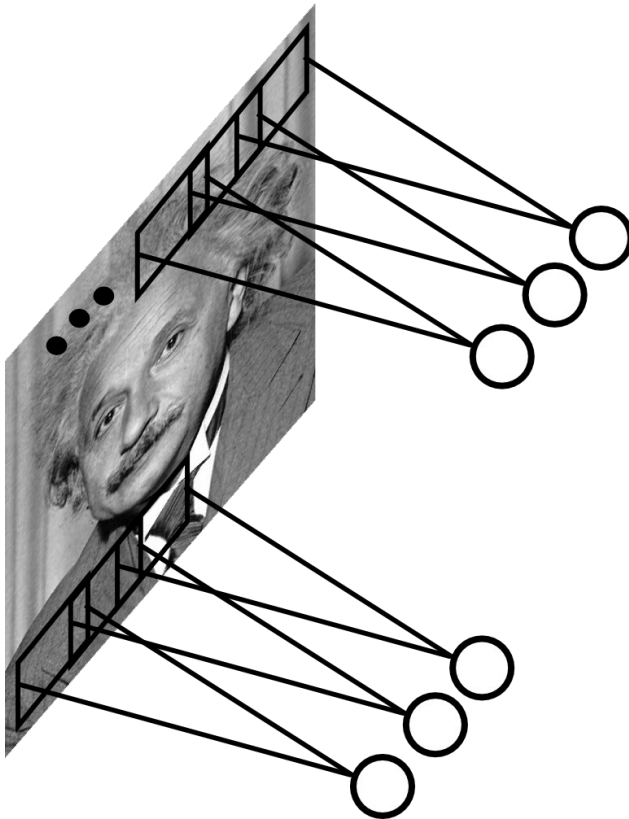
- Consider a 200x200 image
 - A fully connected MLP with 400K hidden nodes has 16 billion parameters
 - A locally connected CNN with 400K hidden nodes each operating on 10x10 receptive fields has 40 million parameters.





Example CNN Setup

- Consider a 200x200 image
 - Set up 10 feature maps, each of size 200x200.
 - Each hidden node is applied on a 10x10 region.
 - 10 different filters applied simultaneously on the image.
 - 10 filters, each 10x10 => 1000 parameters

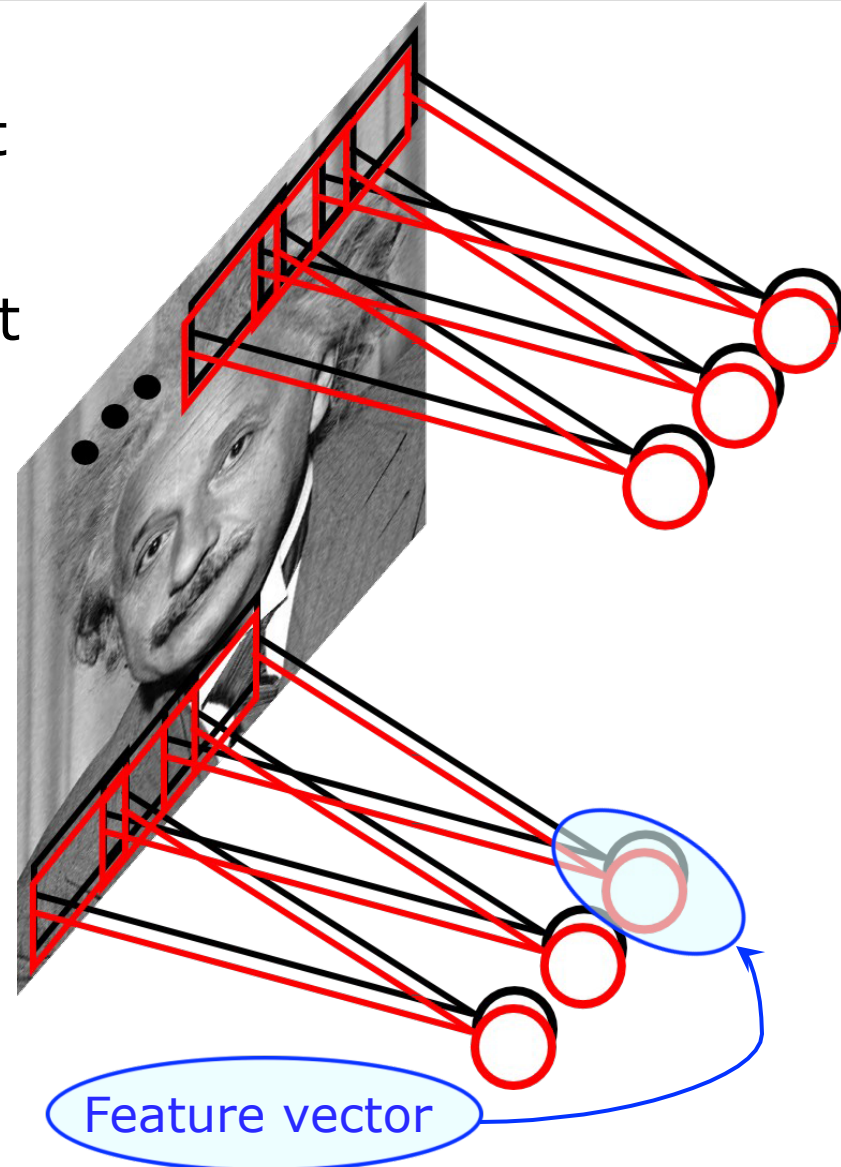
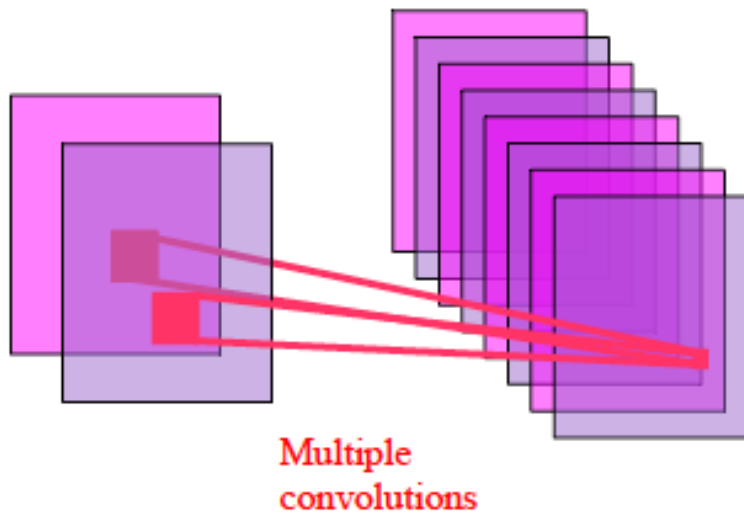


Schematic representation
of a single feature map.
This is repeated 10 times.

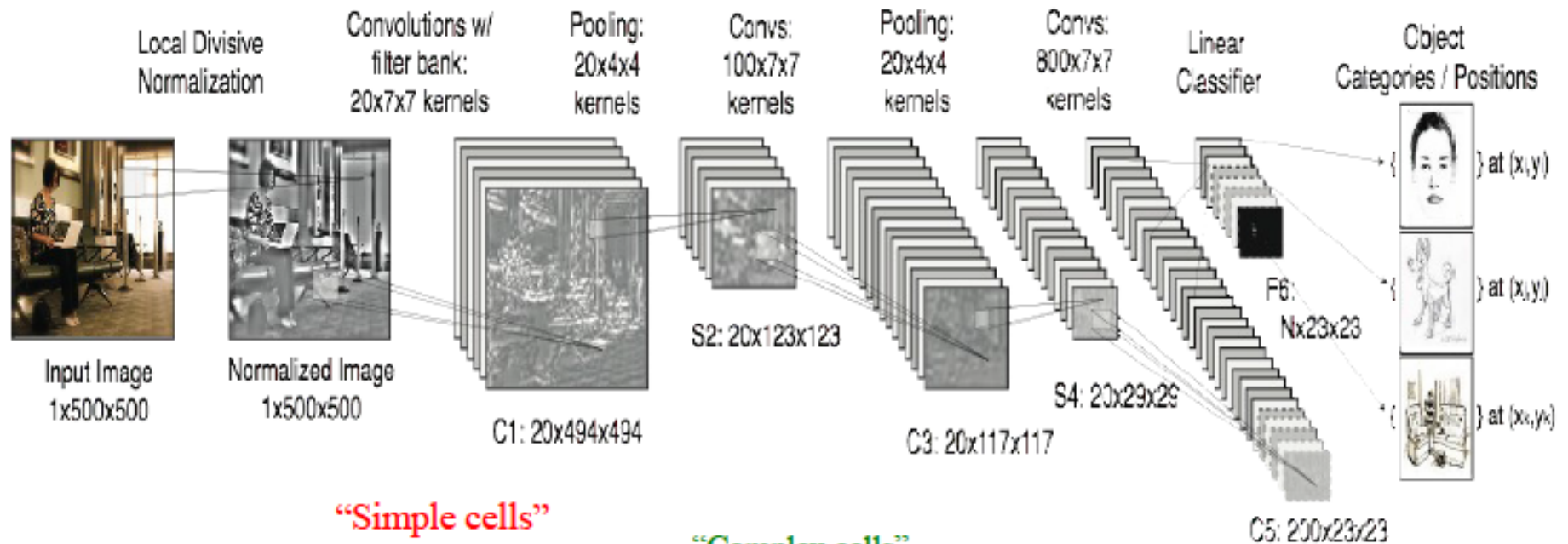
Multiple Convolutions with Different Kernels



- By using many different filters, one can detect multiple motifs at each location.
- The collection of nodes looking at the same local patch is comparable to a feature vector for that patch.

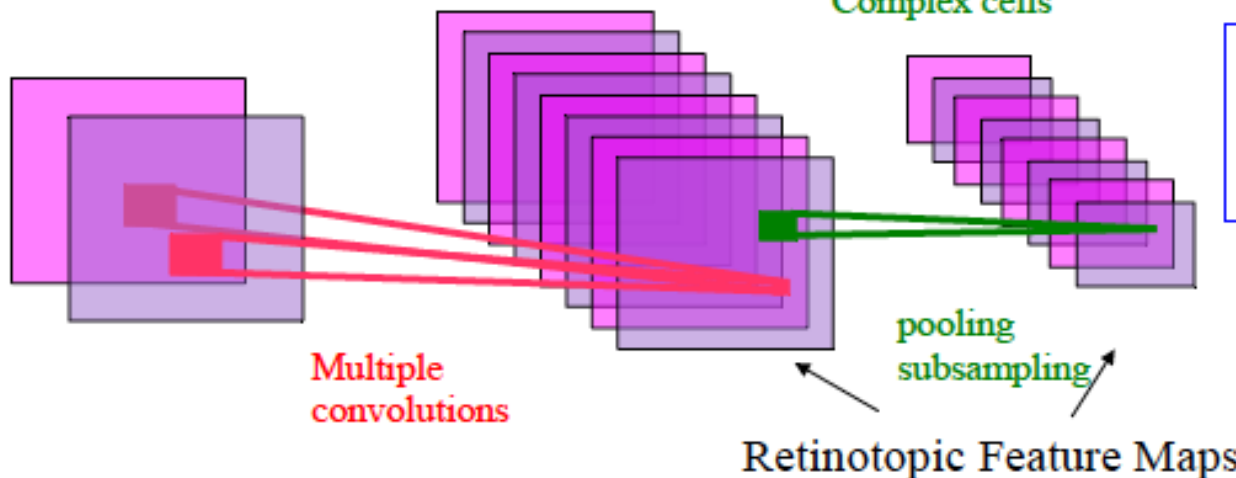


Example: Multistage Hubel-Wiesel System



“Simple cells”

“Complex cells”



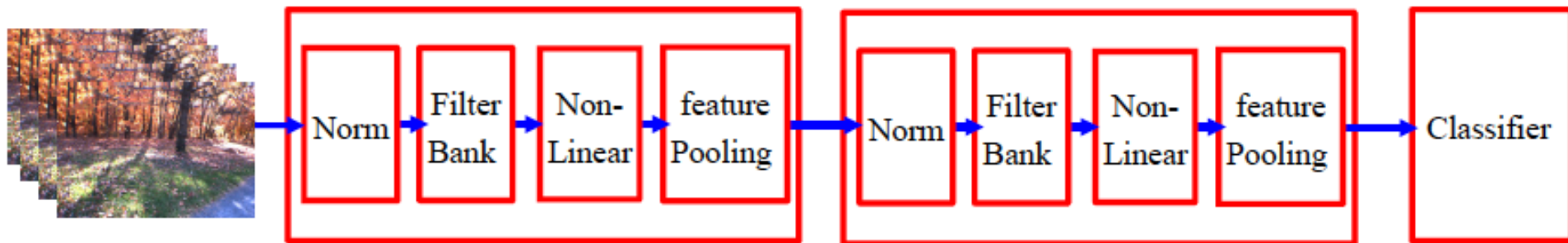
- Training is supervised
- With stochastic gradient descent

[LeCun et al. 89]

[LeCun et al. 98]



Typical CNN Layout

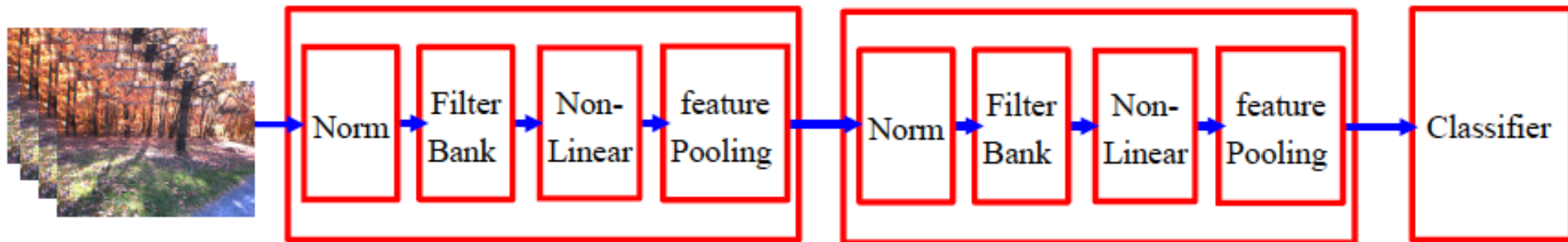


- Stacking multiple stages of [Normalization -> Filter Bank -> Non-Linearity -> Pooling].
- Normalization: variations on “whitening” like high-pass filtering, subtraction of average, local contrast normalization, variance normalization.
- Filter Bank: dimension expansion, projection on new basis
- Non-Linearity: sparsification, sigmoid, lateral inhibition, winner-takes-all
- Pooling: aggregation over space or feature type

$$L_p: \sqrt[p]{X_i^p}; \text{ PROB: } \frac{1}{b} \log \left(\sum_i e^{bX_i} \right)$$



Typical CNN Layout - continued



- Filter Bank -> Non-Linearity allows for non-linear embedding in high dimensions.
- Feature Pooling allows for dimensionality reduction, smoothing, contraction.
- Filter Banks are learned at every stage
- Hierarchy of features:
 - Basic elements inspired by the visual (or auditory cortex).
 - This hierarchy follows the Simple Cell + Complex Cell model of Hubel and Wiesel, 1962.
 - Many “traditional” feature extraction methods like SIFT, HoG, SURF, GIST use such hierarchies.
- Very active field since the mid-2000s.

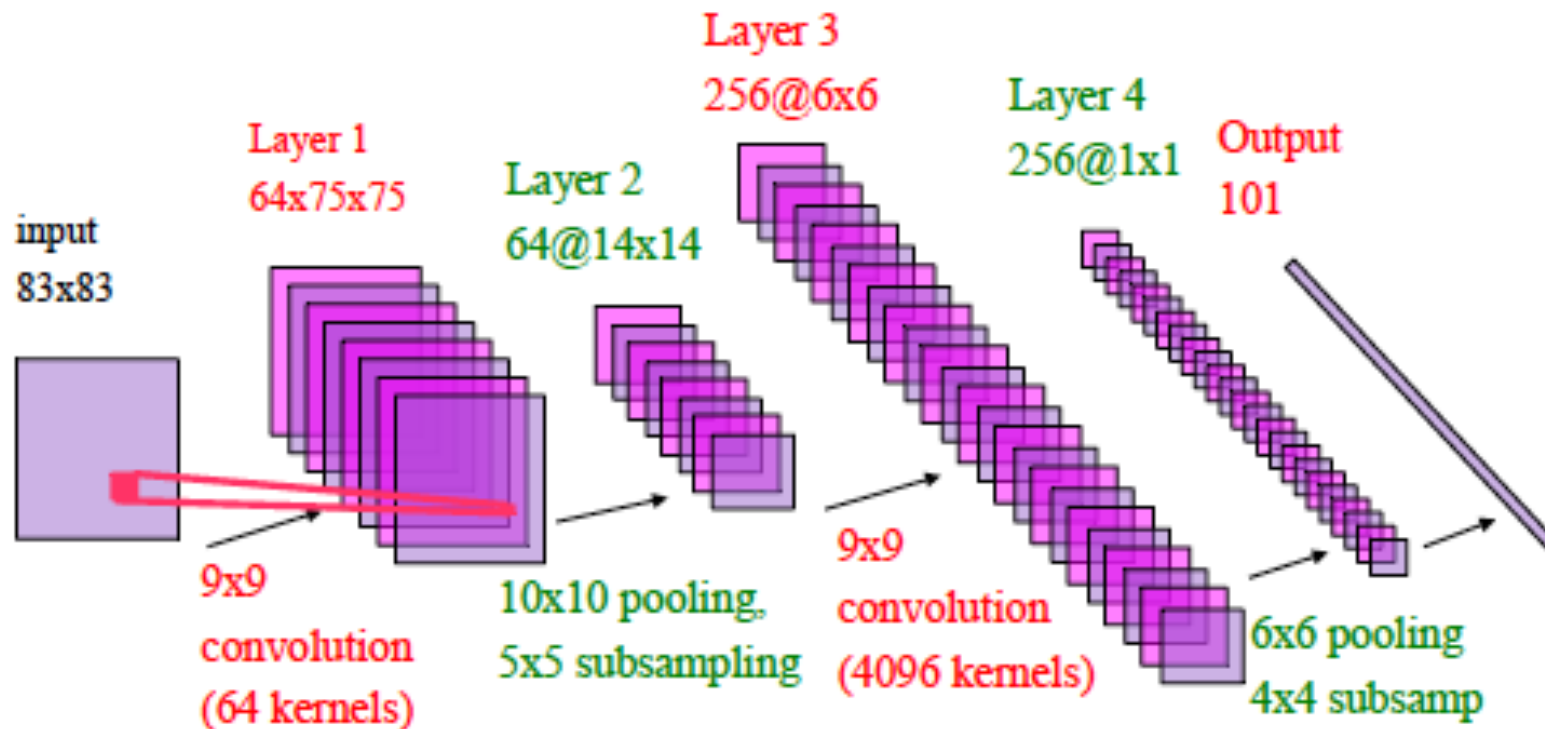
Ideas from Neuroscience and Psychophysics



- Combination of simple cells and complex cells
- Local receptive fields
- Self-similar receptive fields over the visual fields (convolutions)
- Complex cells (pooling)
- Rectified linear units (non-linearity)
- Band-pass filtering and contrast normalization of input.
- Lateral inhibition (divisive contrast normalization)
- Sub-sampling in V1-V2-V4



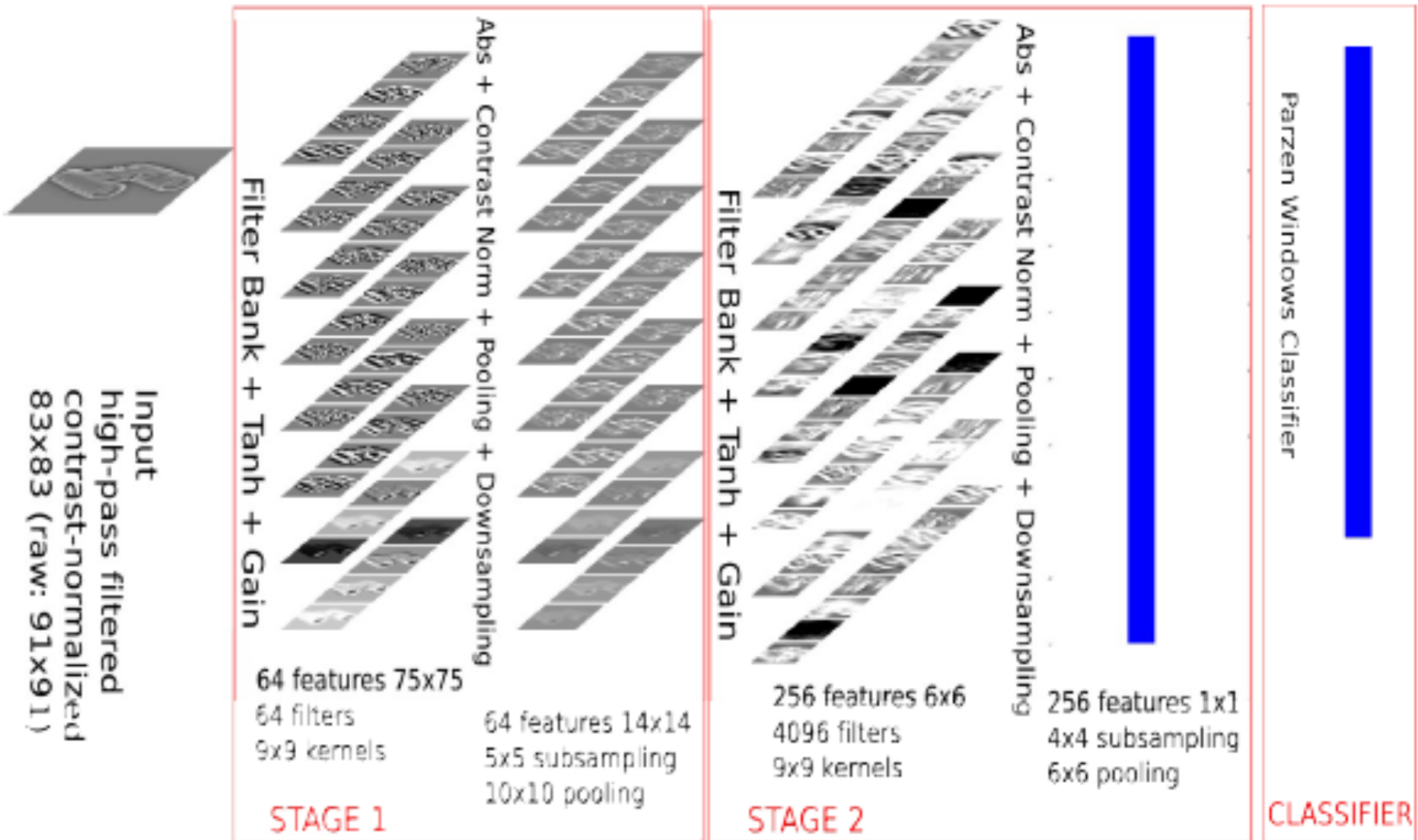
ConvNet (circa 1990)



- **Non-Linearity:** half-wave rectification, shrinkage function, sigmoid
- **Pooling:** average, L1, L2, max
- **Training:** Supervised (1988-2006), Unsupervised+Supervised (2006-now)

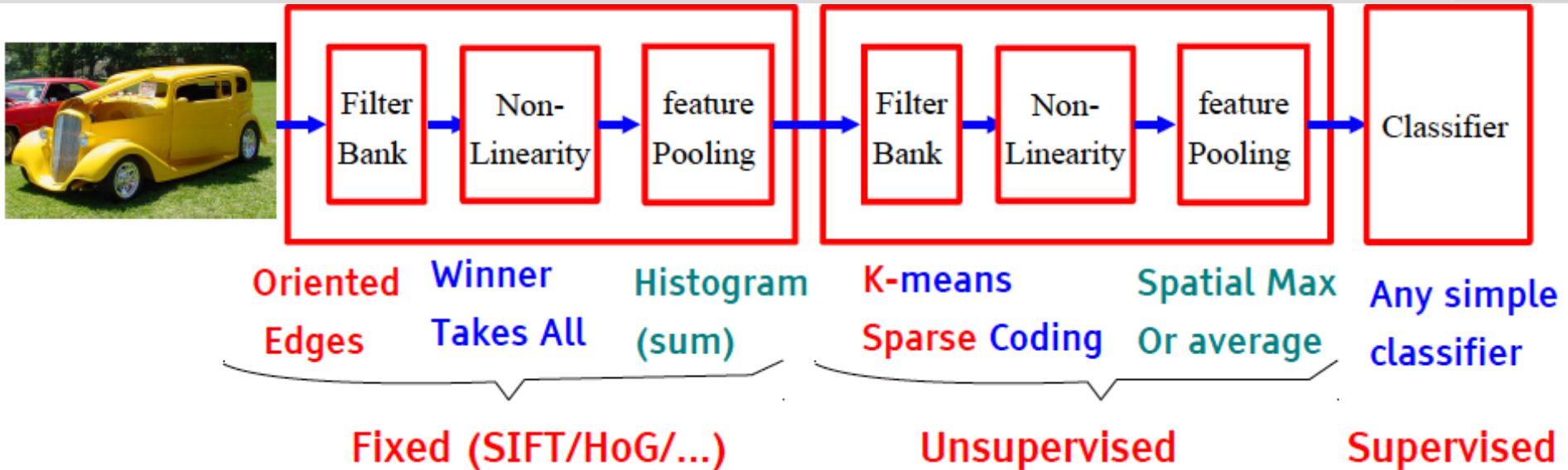


ConvNet - continued





Typical Recognition Pipeline 2006-2012



- **Fixed Features + unsupervised mid-level features + simple classifier**
 - ▶ SIFT + Vector Quantization + Pyramid pooling + SVM
 - [Lazebnik et al. CVPR 2006]
 - ▶ SIFT + Local Sparse Coding Macrofeatures + Pyramid pooling + SVM
 - [Boureau et al. ICCV 2011]
 - ▶ SIFT + Fisher Vectors + Deformable Parts Pooling + SVM
 - [Perronin et al. 2012]



Deep CNNs Very Successful (Best) At

- Handwriting recognition
- OCR in the wild (2011)
- Traffic sign recognition (2011)
- Pedestrian detection (2013)
- Volumetric brain image segmentation (2009)
- Human action recognition (2011)
- Object recognition (2012)
- Scene parsing (2012)
- Scene parsing from depth images (2013)
- Speech recognition (2012)
- Breast cancer cell mitosis detection (2011)



Examples

■ Traffic Sign Recognition (GTSRB)

- ▶ German Traffic Sign Reco Bench
- ▶ 99.2% accuracy



■ House Number Recognition (Google)

- ▶ Street View House Numbers
- ▶ 94.3 % accuracy

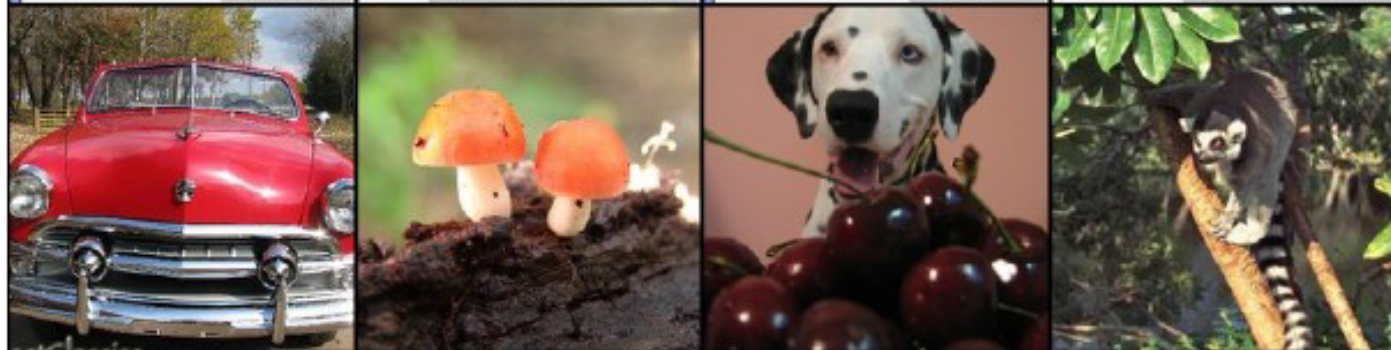


Object Recognition (Krizhevsky et al. 2012)



mite container ship motor scooter leopard

--	--	--	--



grille mushroom cherry Madagascar cat

--	--	--	--

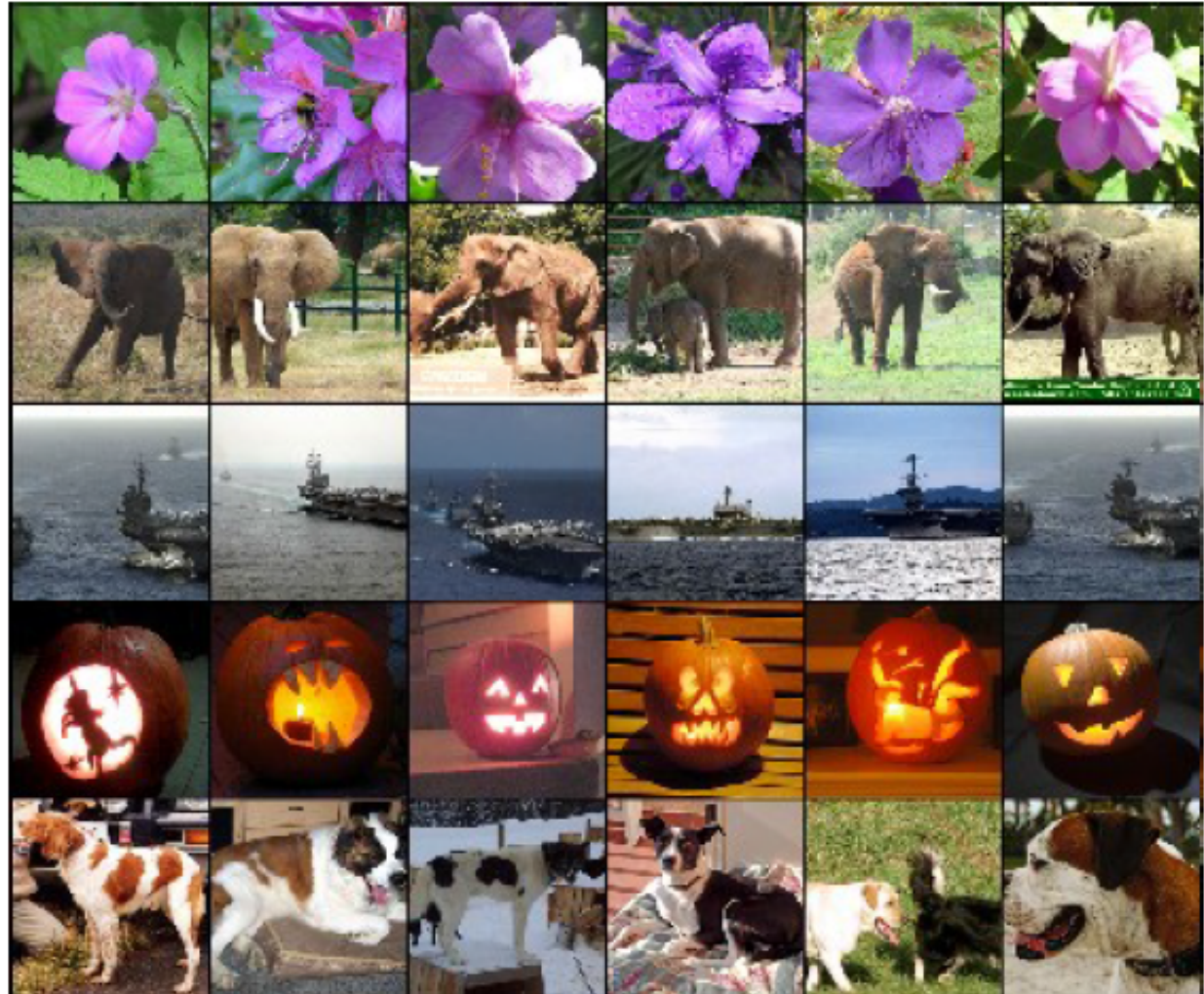
Object Recognition (Krizhevsky et al. 2012)



**TEST
IMAGE**



RETRIEVED IMAGES



References



A large portion of the material on these slides is taken from:

1. Geoffrey Hinton, "CSC2535:Advanced Machine Learning, Lecture 6a, Convolutional Neural Networks for Hand-Written Digit Recognition," <http://www.cs.toronto.edu/~hinton/csc2535/notes/lec6a.ppt>
2. Yann LeCun and Marc'Aurelio Ranzato, "Deep Learning Tutorial", *ICML* June 1013. <http://www.cs.nyu.edu/~yann/talks/lecun-ranzato-icml2013.pdf>
3. Deep Learning Tutorials <http://deeplearning.net/tutorial/>