# Artifical Neural Networks

STAT 27725/CMSC 25400: Machine Learning

Shubhendu Trivedi

University of Chicago

November 2015

- Things we will look at today
  - Biological Neural Networks as inspiration for Artificial Neural Networks
  - Model of a neuron (Perceptron)
  - Multi-layer Perceptrons
  - Training feed forward networks: The Backpropagation algorithm
  - Deep Learning: Convolutional Neural Networks
  - Visualization of learned features

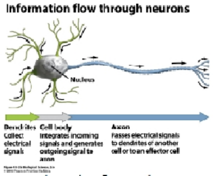
## Neural Networks

- The human brain has an estimated $10^{11}$ neurons, each connected on average, to $10^4$ others
- Inputs come from the dendrites, are aggregated in the soma. If the neuron starts firing, impulses are propagated to other neurons via axons
- Neuron activity is typically excited or inhibited through connections to other neurons
- The fastest neuron switching times are known to be on the order of $10^{-3}$ seconds - quite slow compared to computer switching speeds of $10^{-10}$ seconds
- Yet, humans are surprisingly quick in making complex decisions: Example - takes roughly $10^{-1}$ seconds to visually recognize your mother

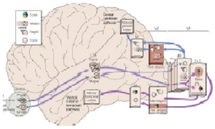
## Neural Networks

- Note that the sequence of neurons firings that can take place during this interval cannot be possibly more than a few hundred steps (given the switching speed of the neurons)
- Thus, depth of the network can't be great (clear layer by layer organization in the visual system)
- This observation has led many to speculate that the information-processing abilities of biological neural systems must follow from highly parallel processes, operating on representations that are distributed over many neurons.

## Neural Networks


Information flow through neurons
Nucleus
Dendrites Collect electrical signals
Cell body Integrates incoming signals and generates outgoing signal to axon
Axon Passes electrical signals to dendrites of another cell or to an effector cell

*simple functions*



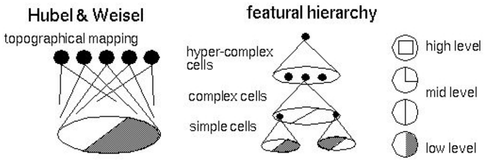
(Van Essen&Gallant, 1994)

*Multi-layered*

- Neurons are simple. But their arrangement in multi-layered networks is very powerful
- They self organize. Learning effectively is change in organization (or connection strengths).
- Humans are very good at recognizing patterns. How does the brain do it?

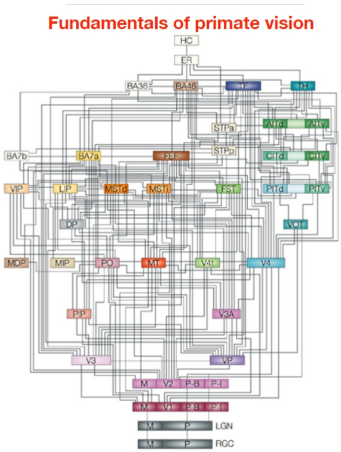
# Neural Networks

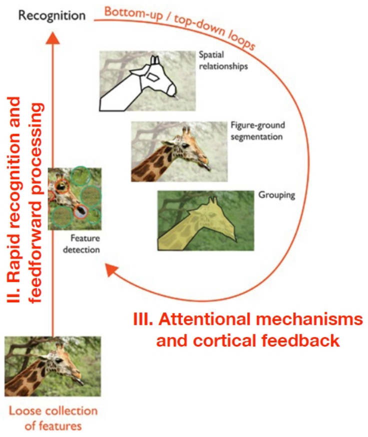
- In the perceptual system, neurons represent features of the sensory input
- The brain learns to extract many layers of features. Features in one layer represent more complex combinations of features in the layer below. (e.g. Hubel Weisel (Vision), 1959, 1962)



- How can we imitate such a process on a computer?

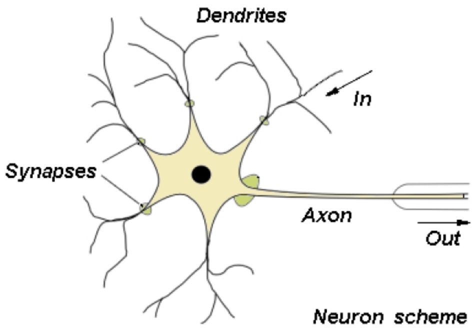
# Neural Networks

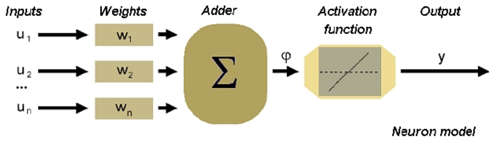



Recognition
Bottom-up / top-down loops
Spatial relationships
Figure-ground segmentation
Grouping
II. Rapid recognition and feedforward processing
Feature detection
III. Attentional mechanisms and cortical feedback
Loose collection of features

[Slide credit: Thomas Serre]

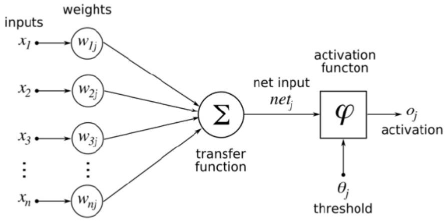
# First Generation Neural Networks: McCullogh Pitts (1943)





# A Model Adaptive Neuron

$$o = \phi(\theta + \sum_p w_p x_p),$$



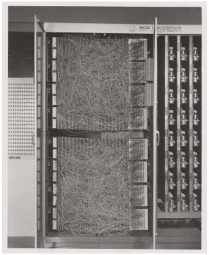
- This is just a Perceptron (seen earlier in class)
- Assumes data are linearly separable. Simple stochastic algorithm for learning the linear classifier
- Theorem (Novikoff, 1962) Let $\mathbf{w}$, $w_0$ be a linear separator with $\|w\| = 1$, and margin $\gamma$. Then Perceptron will converge after

$$O\Big(\frac{(\max_i \|x_i\|)^2}{\gamma^2}\Big)$$

## Perceptron as a model of the brain?

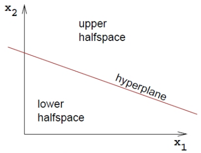
- Perceptron developed in the 1950s
- Key publication: *The perceptron: a probabilistic model for information storage and organization in the brain*, Frank Rosenblatt, Psychological Review, 1958
- Goal: Pattern classification
- From "Mechanization of Thought Process" (1959): "The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence. Later perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech and writing in another language, it was predicted."
- Another ancient milestone: Hebbian learning rule (Donald Hebb, 1949)

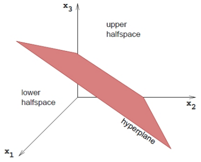
## Perceptron as a model of the brain?



- The Mark I perceptron machine was the first implementation of the perceptron algorithm.
- The machine was connected to a camera that used 2020 cadmium sulfide photocells to produce a 400-pixel image. The main visible feature is a patchboard that allowed experimentation with different combinations of input features.
- To the right of that are arrays of potentiometers that implemented the adaptive weights

## Adaptive Neuron: Perceptron

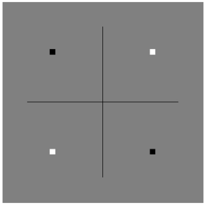




- A perceptron represents a decision surface in a $d$ dimensional space as a hyperplane
- Works only for those sets of examples that are *linearly separable*
- Many boolean functions can be represented by a perceptron: AND, OR, NAND,NOR

## Problems?

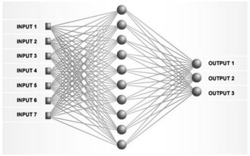
- If features are complex enough, anything can be classified
- Thus features are really hand coded. But it comes with a clever algorithm for weight updates
- If features are restricted, then some interesting tasks cannot be learned and thus perceptrons are fundamentally limited in what they can do. Famous examples: XOR, Group Invariance Theorems (Minsky, Papert, 1969)



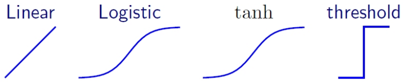
## Coda

- Single neurons are not able to solve complex tasks (linear decision boundaries). Limited in the input-output mappings they can learn to model.
- More layers of linear units are not enough (still linear). A fixed non-linearity at the output is not good enough either
- We could have multiple layers of adaptive, non-linear hidden units. These are called Multi-layer perceptrons
- Were considered a solution to represent nonlinearly separable function in the 70s
- Many local minima: Perceptron convergence theorem does not apply.
- Intuitive conjecture in the 60s: There is no learning algorithm for multilayer perceptrons

## Multi-layer Perceptrons

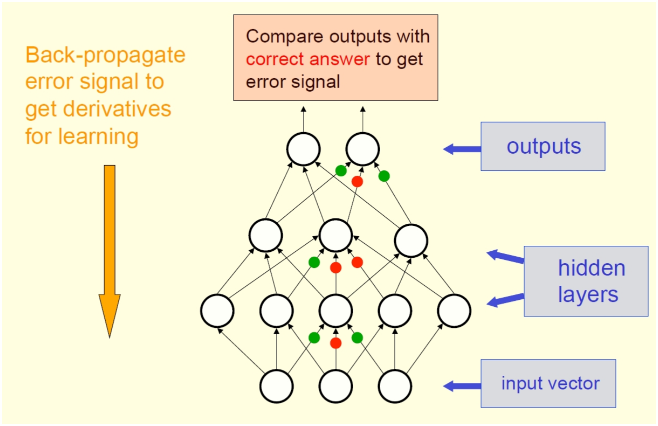


- Digression: Kernel Methods
- We have looked at how each neuron will look like
- But did not mention activation functions. Some common choices:



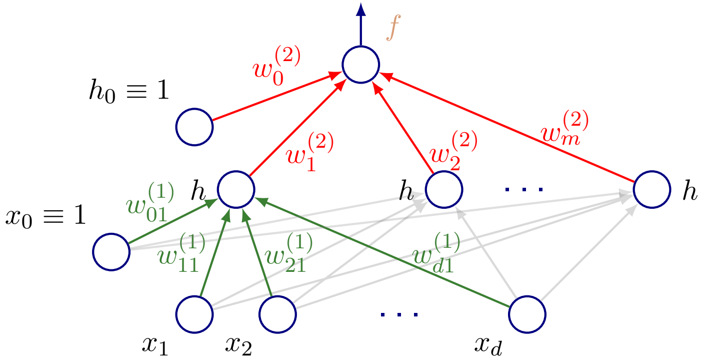
- How can we learn the weights?

## Learning multiple layers of features


Back-propagate error signal to get derivatives for learning
Compare outputs with correct answer to get error signal
outputs
hidden layers
input vector

[Slide: G. E. Hinton]

## Review: neural networks



- Feedforward operation, from input $\mathbf{x}$ to output $\hat{y}$:

$$\hat{y}(\mathbf{x};\mathbf{w}) = f\left(\sum_{j=1}^{m} w_j^{(2)} h\left(\sum_{i=1}^{d} w_{ij}^{(1)} x_i + w_{0j}^{(1)}\right) + w_0^{(2)}\right)$$

*Slide adapted from TTIC 31020, Gregory Shakhnarovich*

## Training the network

- Error of the network on a training set:

$$L(X; \mathbf{w}) = \sum_{i=1}^{N} \frac{1}{2} \left(y_i - \hat{y}(\mathbf{x}_i; \mathbf{w})\right)^2$$

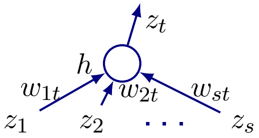
- Generally, no closed-form solution; resort to gradient descent
- Need to evaluate derivative of $L$ on a single example
- Let's start with a simple linear model $\hat{y} = \sum_j w_j x_{ij}$:

$$\frac{\partial L(\mathbf{x}_i)}{\partial w_j} = \underbrace{(\hat{y}_i - y_i)}_{\text{error}} x_{ij}.$$

# Backpropagation

- General unit activation in a multilayer network:

$$z_t = h\left(\sum_j w_{jt} z_j\right)$$



- Forward propagation: calculate for each unit $a_t = \sum_j w_{jt} z_j$
- The loss $L$ depends on $w_{jt}$ only through $a_t$:

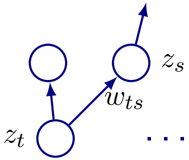
$$\frac{\partial L}{\partial w_{jt}} = \frac{\partial L}{\partial a_t}\frac{\partial a_t}{\partial w_{jt}} = \frac{\partial L}{\partial a_t} z_j$$

## Backpropagation

$$\frac{\partial L}{\partial w_{jt}} = \frac{\partial L}{\partial a_t} z_j \qquad \frac{\partial L}{\partial w_{jt}} = \underbrace{\frac{\partial L}{\partial a_t}}_{\delta_t} z_j$$

- Output unit with linear activation: $\delta_t = \hat{y} - y$
- Hidden unit $z_t = h(a_t)$ which sends inputs to units $S$:

$$\delta_t = \sum_{s \in S} \frac{\partial L}{\partial a_s} \frac{\partial a_s}{\partial a_t} = h'(a_t) \sum_{s \in S} w_{ts} \delta_s$$



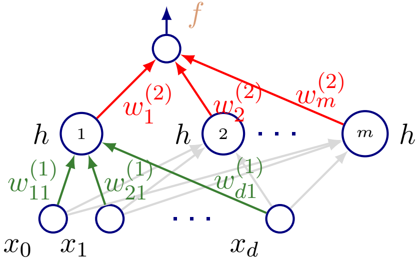
$$a_s = \sum_{j : j \to s} w_{js} h(a_j)$$

## Backpropagation: example

- Output: $f(a) = a$
- Hidden:

$$h(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}},$$

$$h'(a) = 1 - h(a)^2.$$



- Given example $\mathbf{x}$, feed-forward inputs:

$$\text{input to hidden: } a_j = \sum_{i=0}^{d} w_{ij}^{(1)} x_i,$$

$$\text{hidden output: } z_j = \tanh(a_j),$$

$$\text{net output: } \hat{y} = a = \sum_{j=0}^{m} w_j^{(2)} z_j.$$

# Backpropagation: example

$$a_j = \sum_{i=0}^{d} w_{ij}^{(1)} x_i, \quad z_j = \tanh(a_j), \quad \hat{y} = a = \sum_{j=0}^{m} w_j^{(2)} z_j.$$

- Error on example $\mathbf{x}$: $L = \frac{1}{2}(y - \hat{y})^2$.
- Output unit: $\delta = \frac{\partial L}{\partial a} = y - \hat{y}$.
- Next, compute $\delta$s for the hidden units:

$$\delta_j = (1 - z_j)^2 w_j^{(2)} \delta$$

- Derivatives w.r.t. weights:

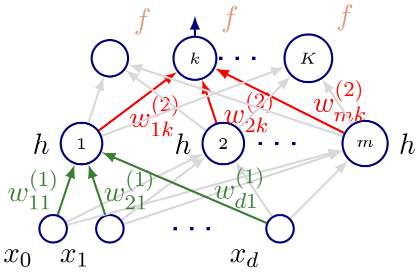
$$\frac{\partial L}{\partial w_{ij}^{(1)}} = \delta_j x_i, \qquad \frac{\partial L}{\partial w_j^{(2)}} = \delta z_j.$$

- Update weights: $w_j \leftarrow w_j - \eta \delta z_j$ and $w_{ij}^{(1)} \leftarrow w_{ij}^{(1)} - \eta \delta_j x_i$. $\eta$ is called the weight decay

## Multidimensional output

- Loss on example $(\mathbf{x}, \mathbf{y})$:

$$\frac{1}{2}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2$$



- Now, for each output unit $\delta_k = y_k - \hat{y}_k$;
- For hidden unit $j$,

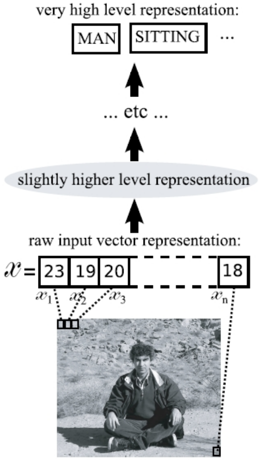
$$\delta_j = (1 - z_j)^2 \sum_{k=1}^{K} w_{jk}^{(2)} \delta_k.$$

## Multilayer Perceptrons

- Theoretical result [Cybenko, 1989]: 2-layer net with linear output can approximate any continuous function over compact domain to arbitrary accuracy (given enough hidden units!)
- The more number of hidden layers, the better...
- .. in theory. Large neural networks need a lot of labeled data, and optimization is hard
- Neural Networks went out of fashion due to this reason roughly between 1990-2005
- Since 2006, they have made a comeback. Mostly due to availability of large datasets, more computational resources at hand, and a number of tricks to make them work
- Return very competitive and state of the art performance in tasks with perceptual input such as Vision and Speech (better than human performance in some tasks), and dominating the landscape in Natural Language Processing

## Deep Learning: Convolutional Neural Networks

# Hierarchial Representations

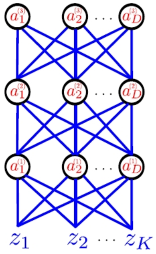

very high level representation:
MAN
SITTING
...
... etc ...
slightly higher level representation
raw input vector representation:
$x$ =
23
19
20
18
$x_1$
$x_2$
$x_3$
$x_n$

Let's elaborate a bit.

## Why use Deep Multi Layered Models?

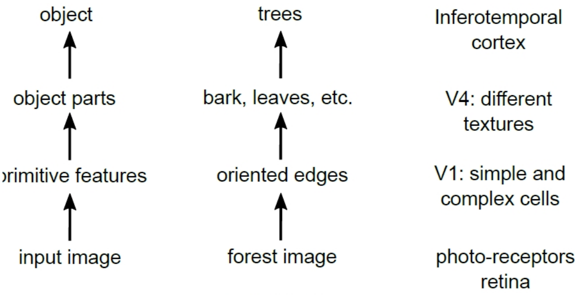
Argument 1: Visual scenes are hierarchially organized (so is language, Neural Nets for that next time)

$$\begin{tikzcd}
\text{object} & \text{trees} \\
\text{object parts} \arrow[u] & \text{bark, leaves, etc.} \arrow[u] \\
\text{primitive features} \arrow[u] & \text{oriented edges} \arrow[u] \\
\text{input image} \arrow[u] & \text{forest image} \arrow[u]
\end{tikzcd}$$


$a_1^{(3)}$ $a_2^{(3)}$ ... $a_D^{(3)}$
$a_1^{(2)}$ $a_2^{(2)}$ ... $a_D^{(2)}$
$a_1^{(1)}$ $a_2^{(1)}$ ... $a_D^{(1)}$
$z_1$ $z_2$ ... $z_K$

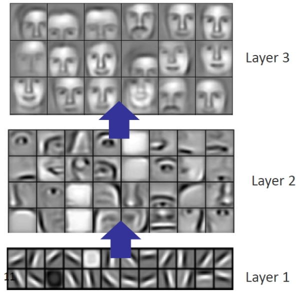
## Why use Deep Multi Layered Models?

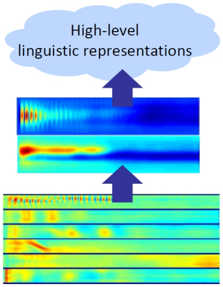
Argument 2: Biological vision is hierarchically organized, and we want to glean some ideas from there



Argument 3: Shallow representations are inefficient at representing highly varying functions

## Why use Deep Multi Layered Models?



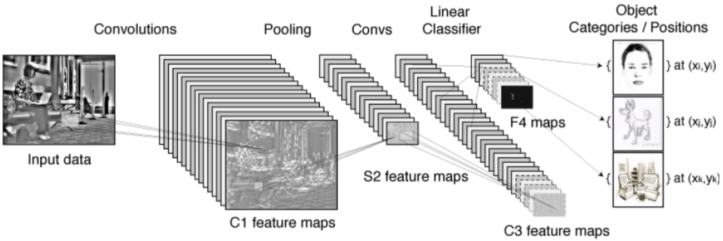


[Figure: Honglak Lee]

## Motivation: Vision

- How can we produce good internal representations of visual data to support recognition?
- What do we mean by good? The learning machine should be able to classify objects into classes, and not be affected by things such as pose, scale, position of the object in the image, lighting conditions, clutter, occlusion etc.
- One way to attempt to do this has resulted in a breed of feedforward neural networks with a very specific kind of architecture - Convolutional Neural Networks. This architecture tries to capture some of the above invariances.
- Originally introduced in 1989 (*Backpropagation applied to handwriting zip code recognition*, Y. LeCun, 1989; *Gradient-based learning applied to document recognition*, LeCun *et al*, 1998)

## Convolutional Neural Networks



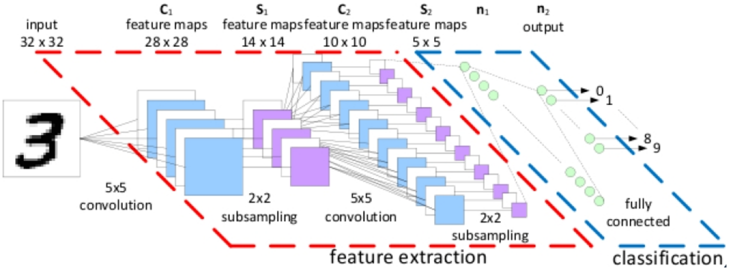
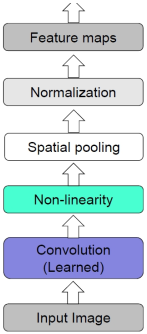


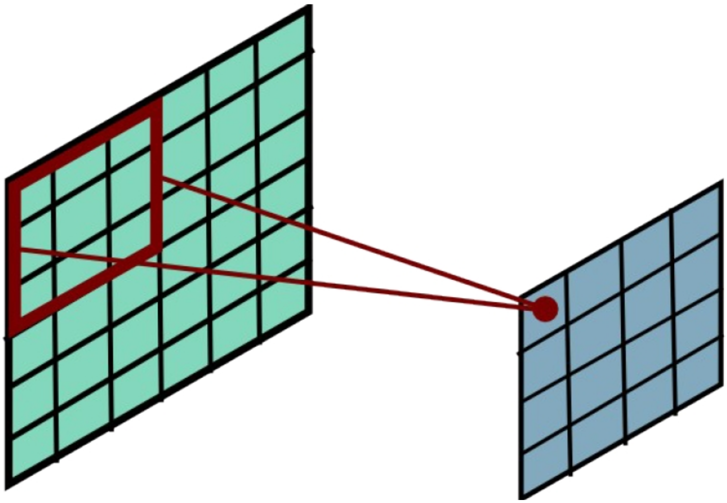
*Figure: Yann LeCun*

## Convolutional Neural Networks

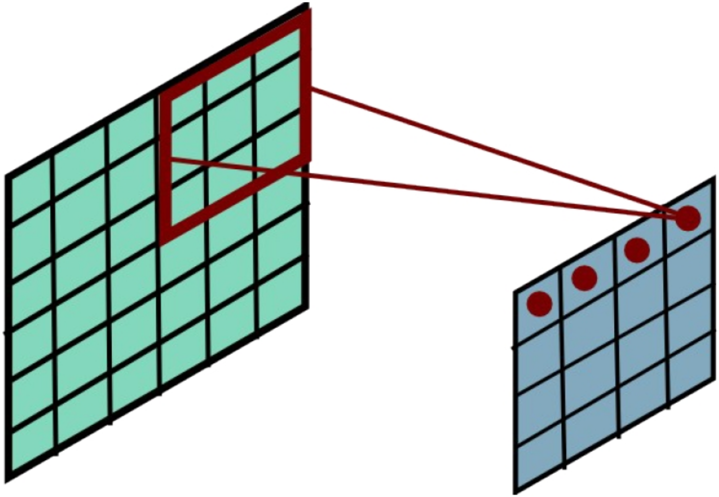


- Feedforward feature extraction: Convolve input with learned filters $\rightarrow$ Non-linearity $\rightarrow$ Spatial Pooling $\rightarrow$ Normalization.
- Training is done by backpropagating errors

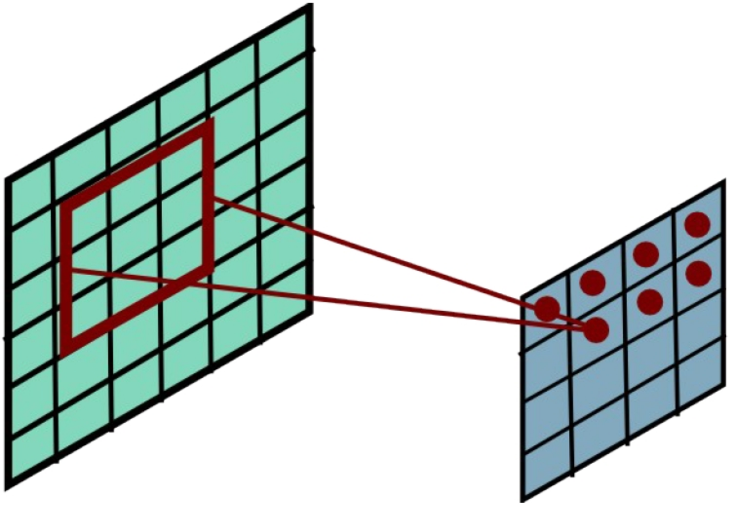
## Convolutional Layer



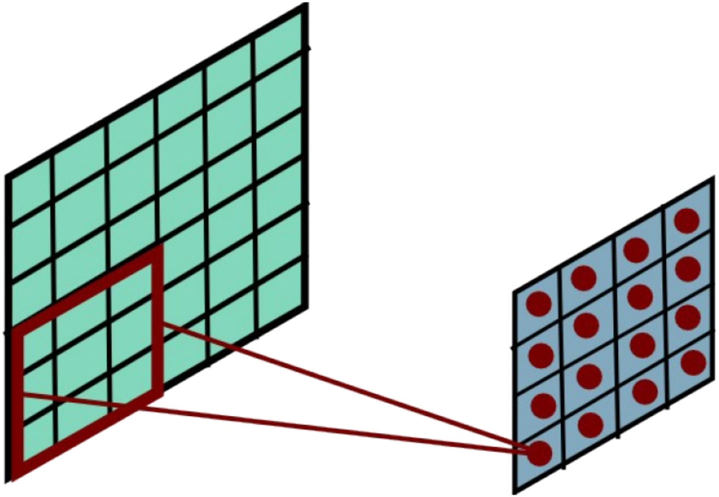
## Convolutional Layer



## Convolutional Layer



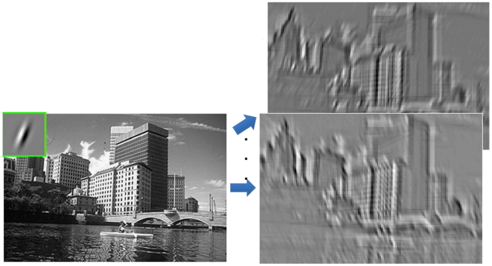
## Convolutional Layer



[Illustration by Ranzato. Mathieu *et al.* Fast Training of CNNs through FFTs]

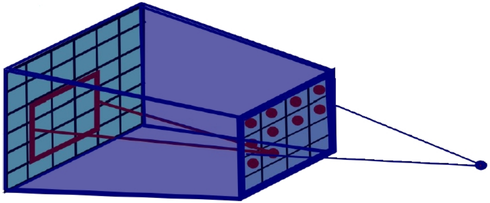
## Convolutional Layer

- Learn multiple such filters
- If 100 filters are used, we get 100 feature maps



## Subsampling

- Pass the each "pixel" of the feature map through a nonlinearity
- Subsample to reduce the size of feature map into half (need not be half)

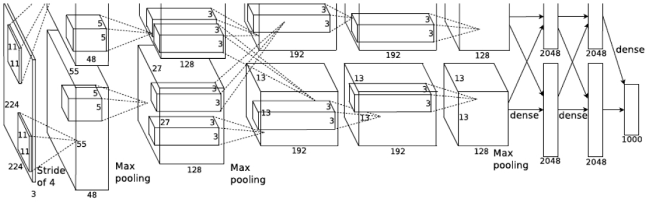


- repeat convolutions on these reduced images, followed by non-linearity and subsampling.
- Eventually we will have feature maps of size 1. These are fed to a classifier, such as SVM for the final classification

## Visualizing Features: ImageNet Challenge 2012

- 14 million labeled images with 20,000 classes
- Images gathered from the internet and labeled by humans via Amazon Turk
- Challenge: 1.2 million training images, 1000 classes.

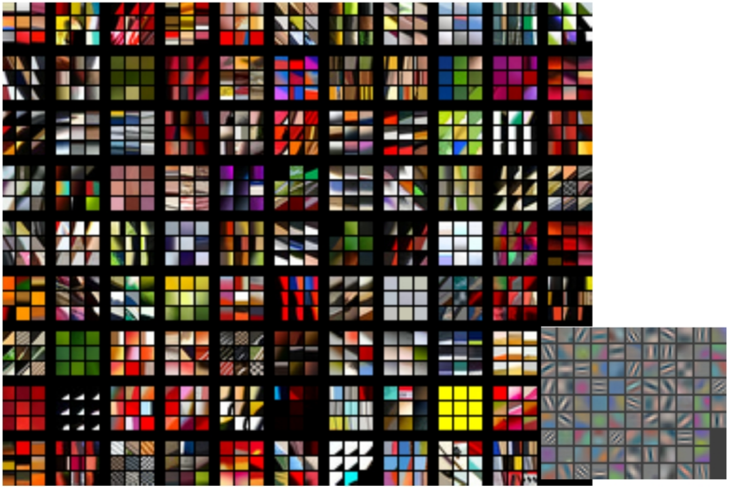
# Visualizing Features: ImageNet Challenge 2012



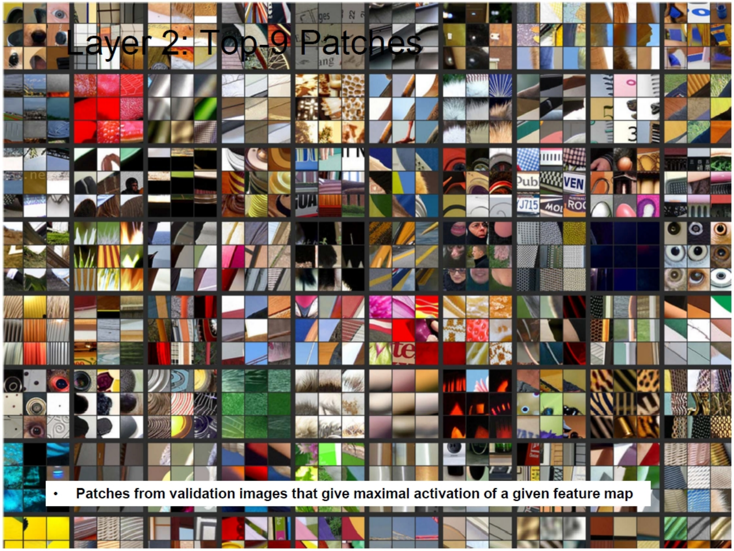
- Winning model ("AlexNet") was a convolutional network similar to Yann LeCun, 1998
- More data: 1.2 million versus a few thousand images
- Fast two GPU implementation trained for a week
- Better regularization (DropOut, next time?)

[A. Krizhevsky, I. Sutskever, G. E. Hinton: ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012]

## Layer 1 filters

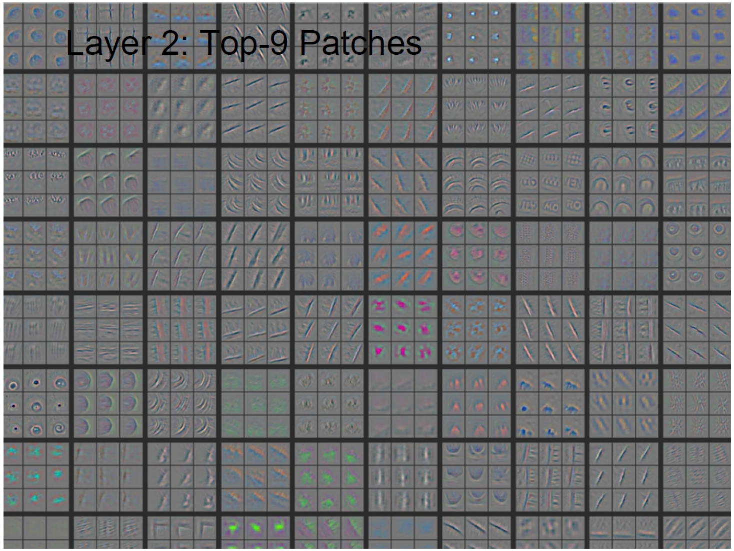


## Layer 2 Patches

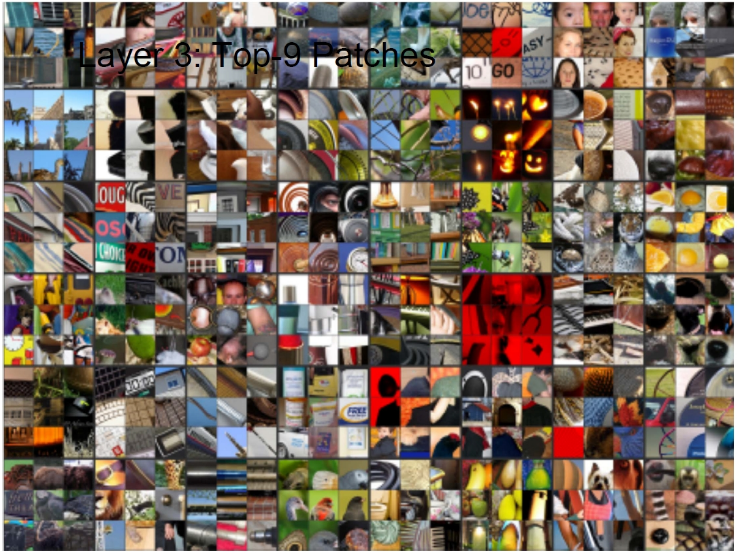

Layer 2: Top-9 Patches

- Patches from validation images that give maximal activation of a given feature map

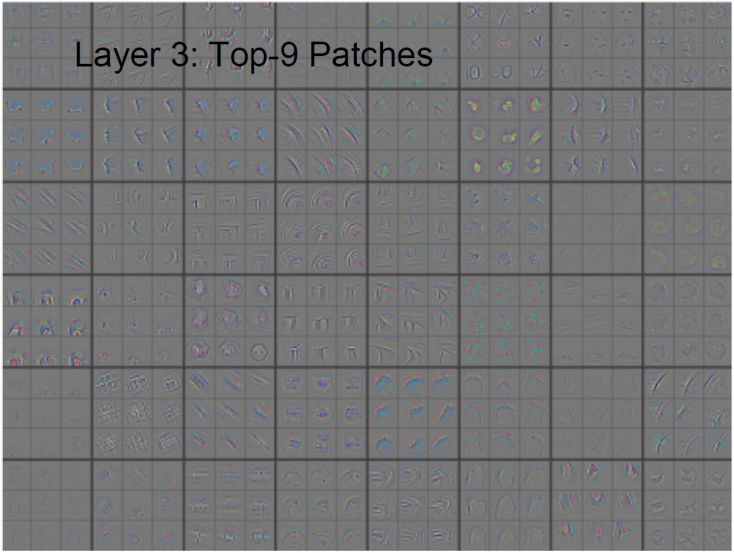
## Layer 2 Patches



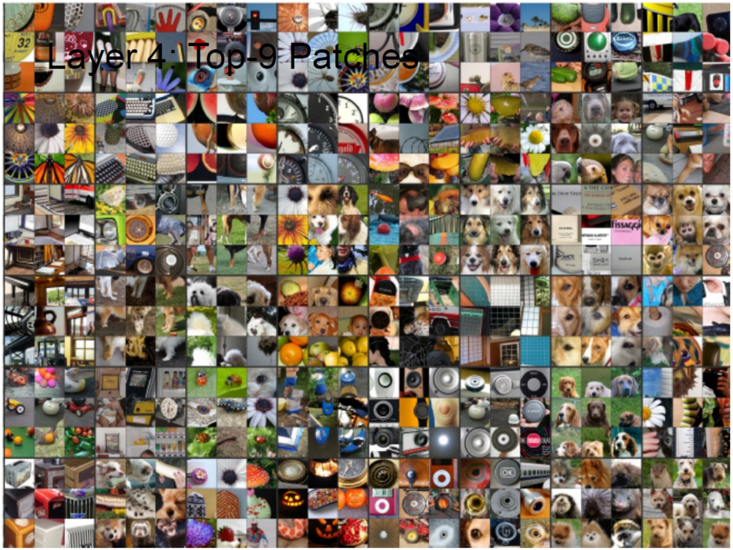
## Layer 3 Patches



## Layer 3 Patches



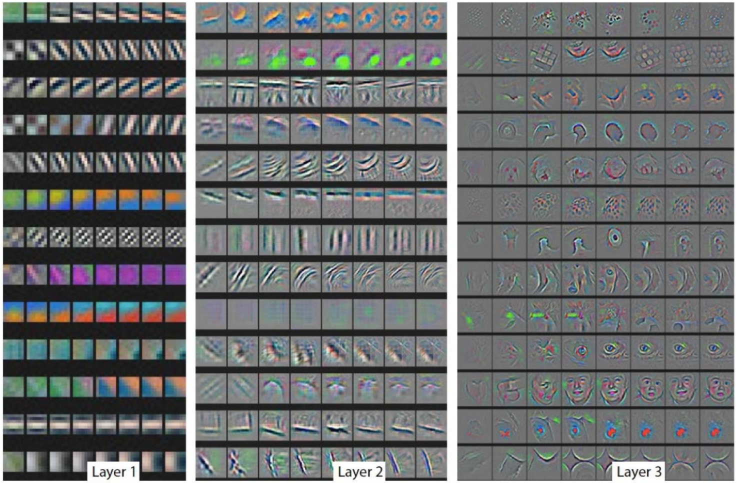
## Layer 4 Patches



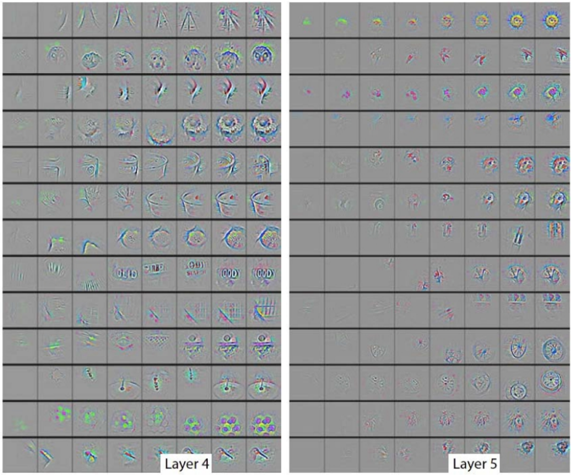
## Layer 4 Patches

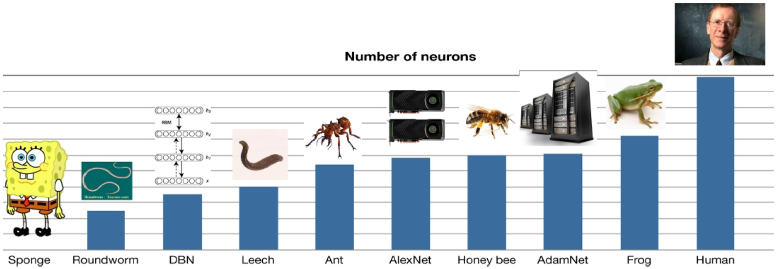
Layer 4: Top-9 Patches

## Evolution of Filters



## Evolution of Filters

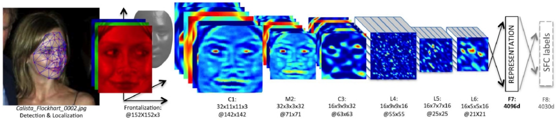


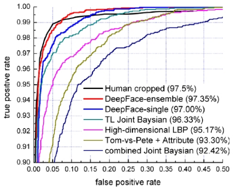


(Goodfellow, 2013)

## Conv. Net Successes

- Very active area of research. Best accuracies on Google Street view, MNIST, traffic sign recognition, object detection, face recognition (DeepFace is used on FB) and detection, semantic segmentation are by using convolutional networks
- Current networks are deeper (Google LeNet has 22 layers, "Highway Networks" (Schmidhuber *et al*) can be deeper still)





## Next time

- Regularization in training feedforward networks
- Basic ideas of neural generative models (Restricted Boltzmann Machines, Deep Belief Nets), Autoencoders. Connections to Manifold Learning
- Recurrent Neural Networks (modeling sequences)
- Time permitting: Recursive Neural Networks and Neural word embeddings