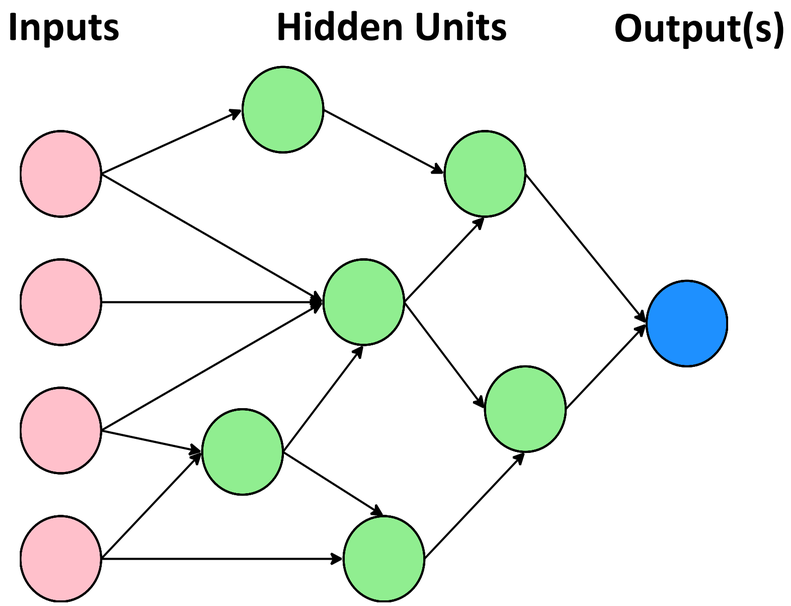
Neural Networks 2
Convolutional Networks

(Some slide credits to CMU 16-720 Fall 2014)

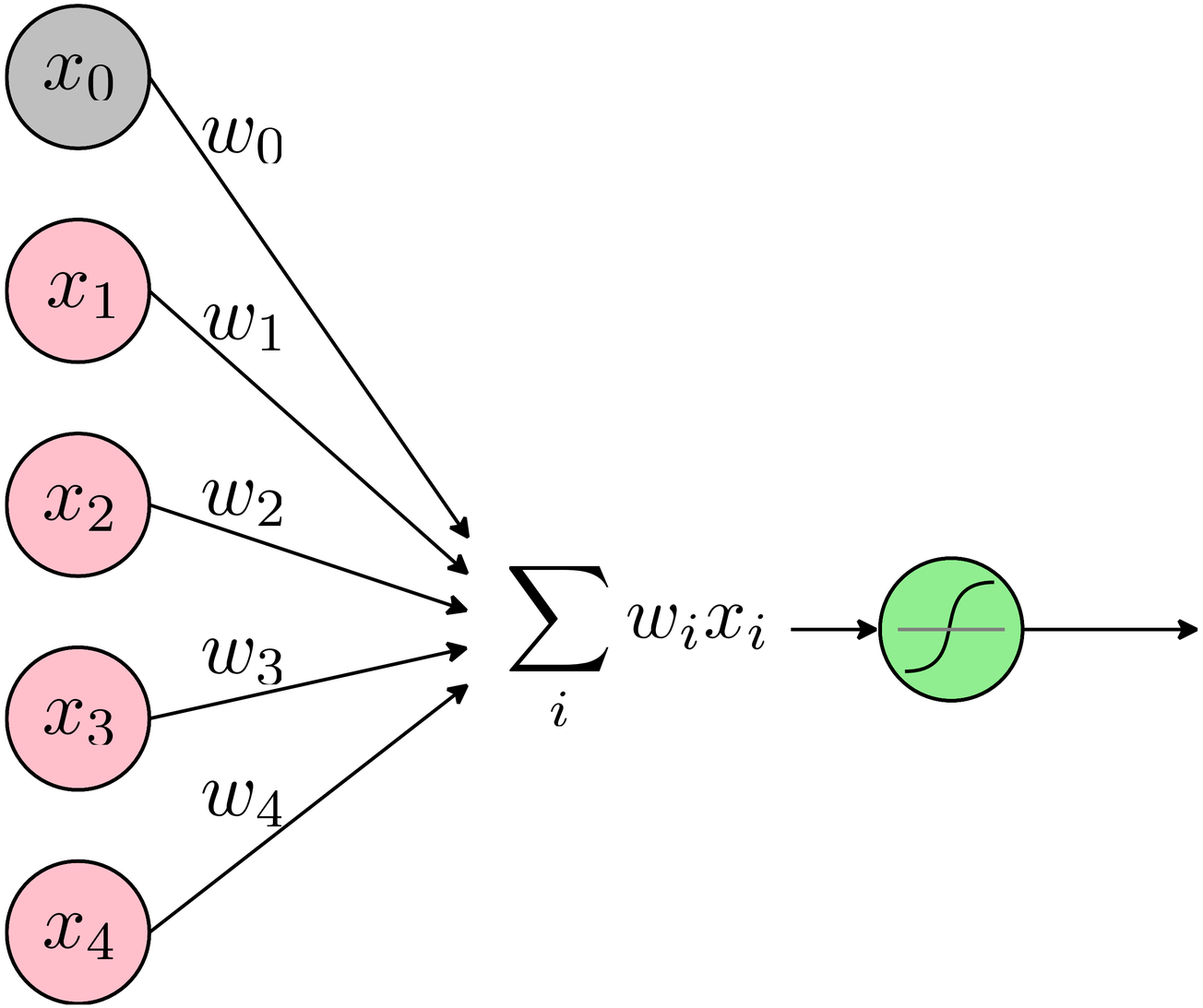
Gary Overett

Neural Network Questions

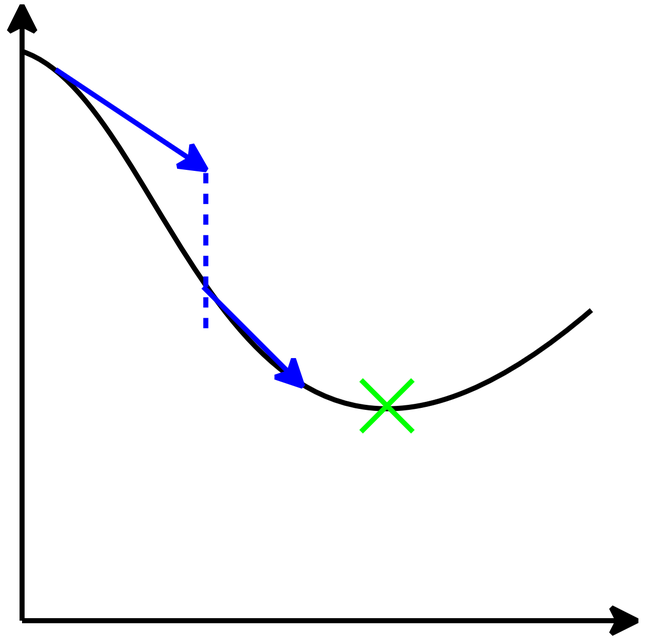
- What structure to use?
- What non-linearity's to adopt?
- Size/Depth of the network
- HOW TO DETERMINE (LEARN) THE WEIGHTS???
- How to use and arrange data?

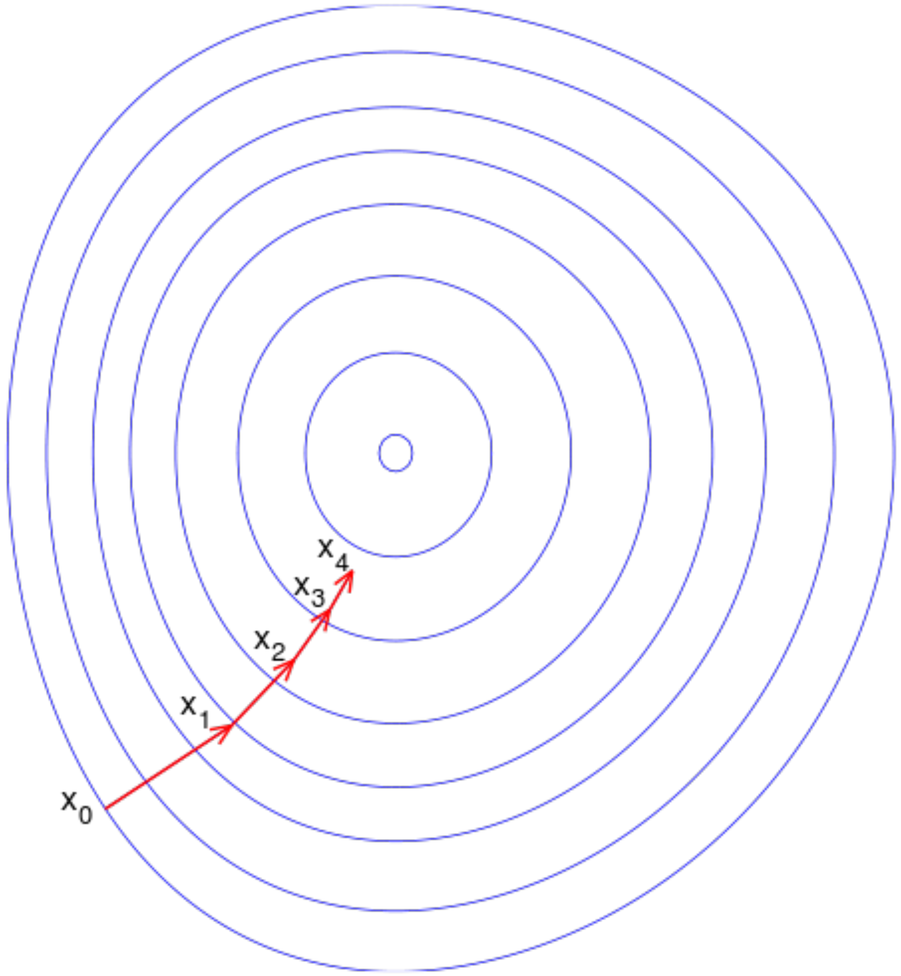


A Neuron

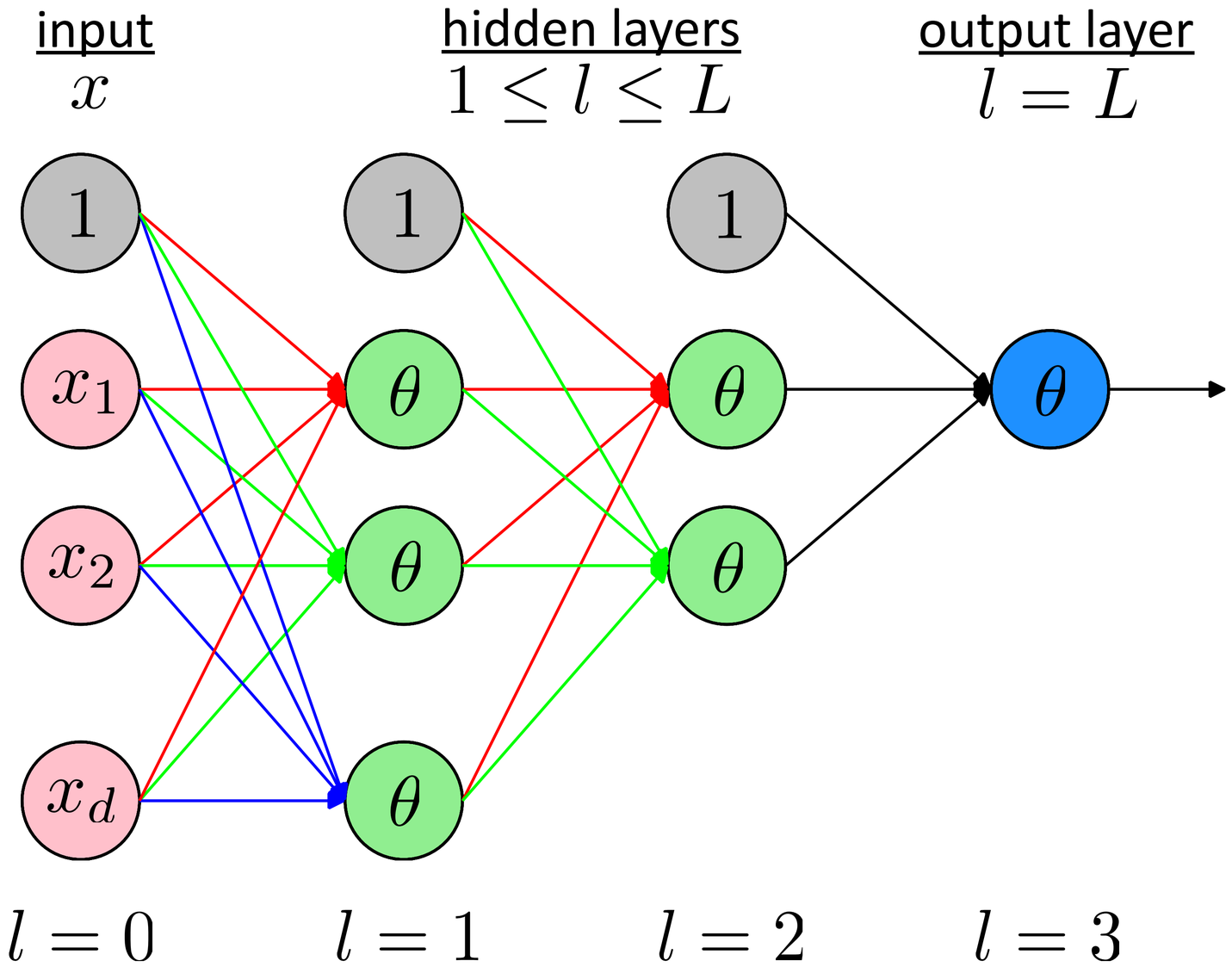


Gradient Descent

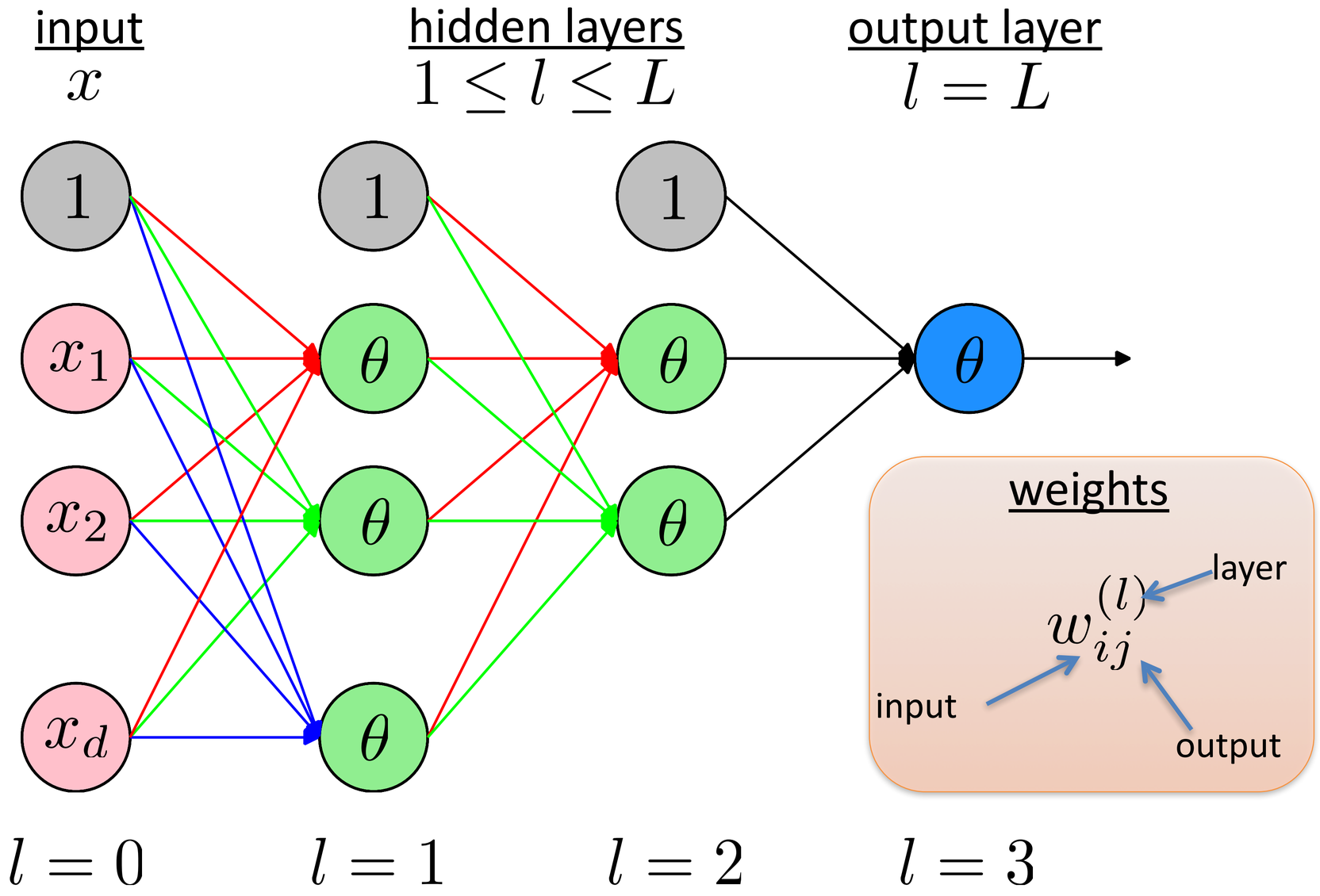




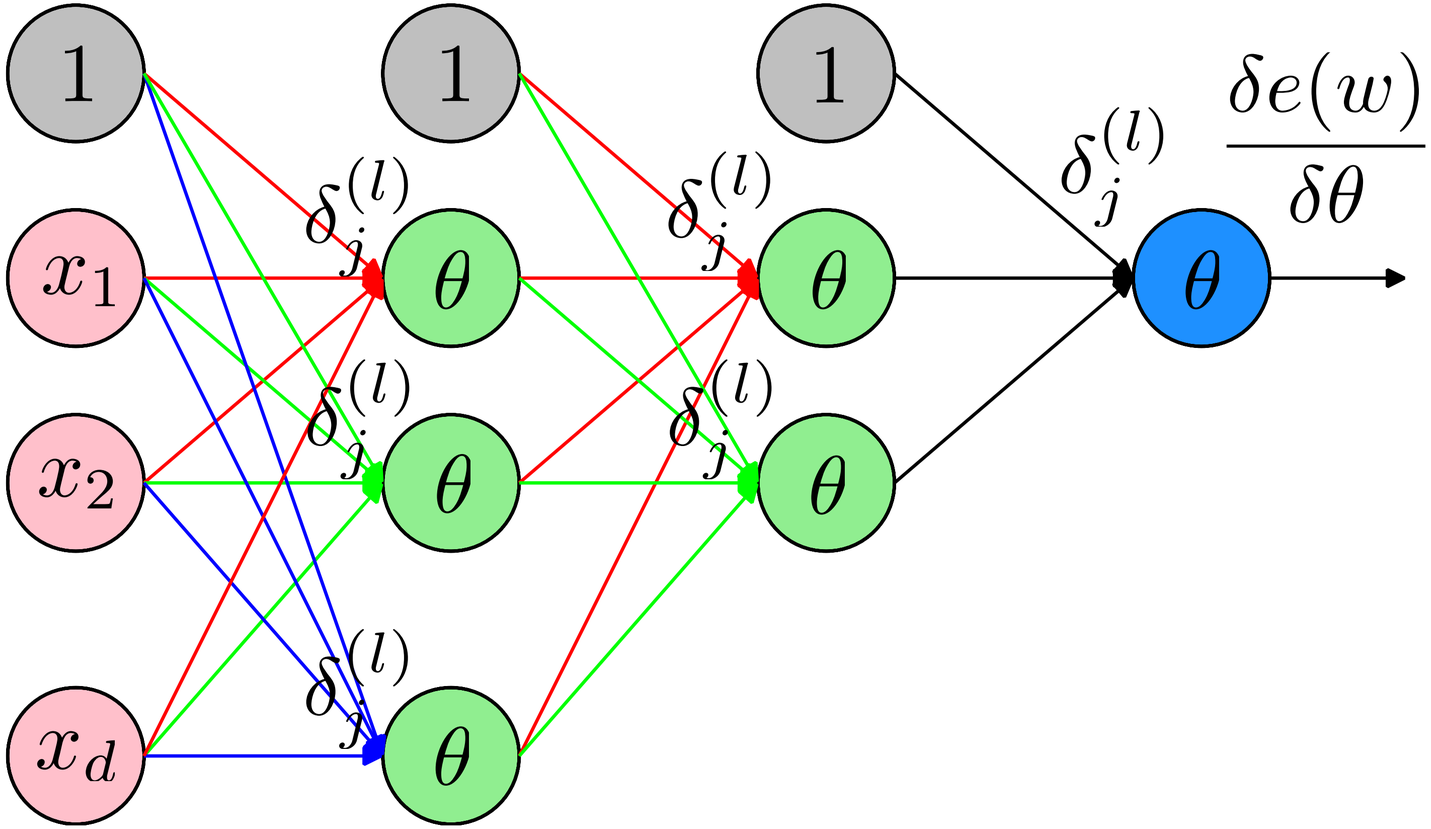
Neural Network Notation



Neural Network Notation

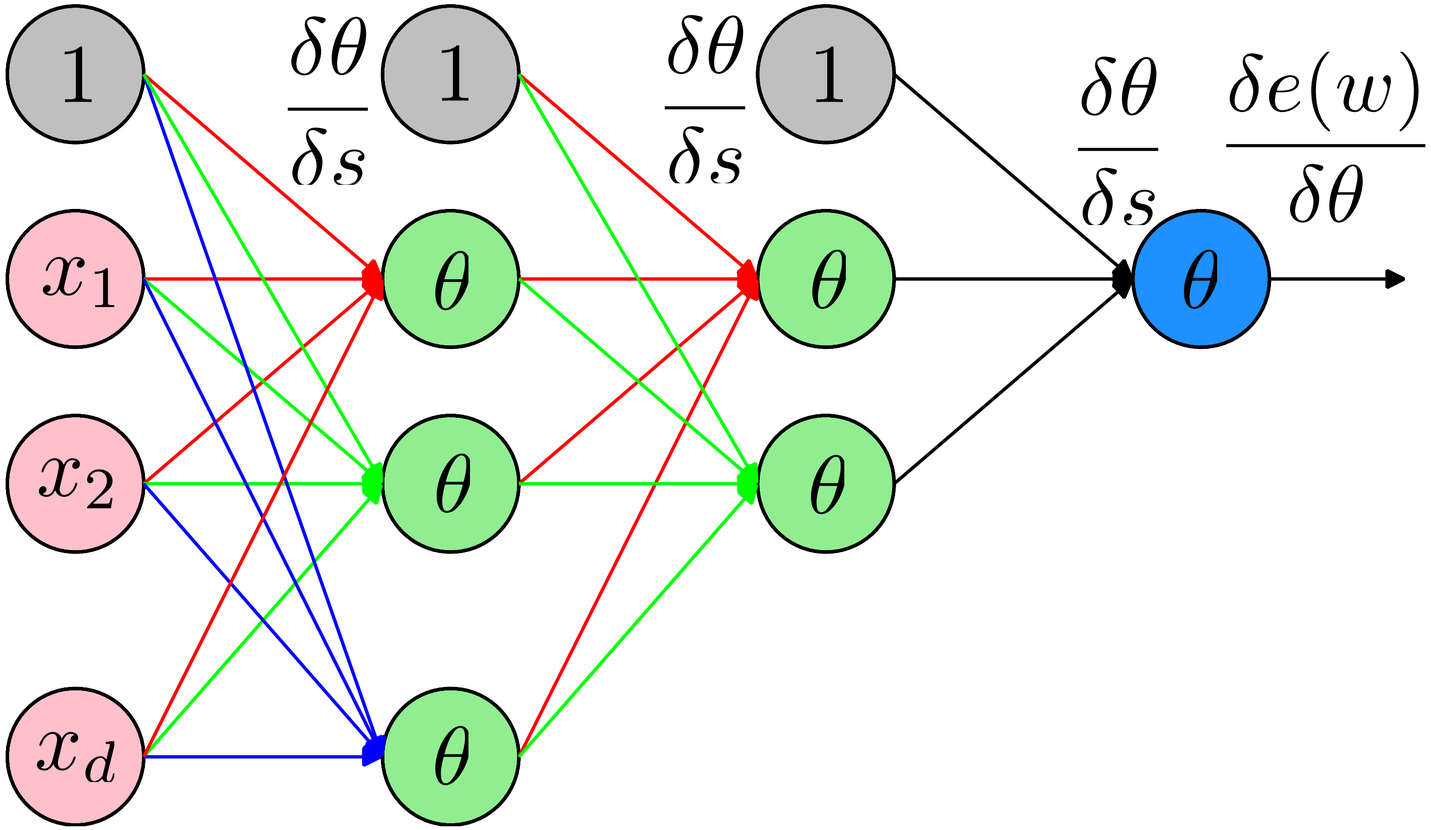


Computing the “Delta’s” Recursively

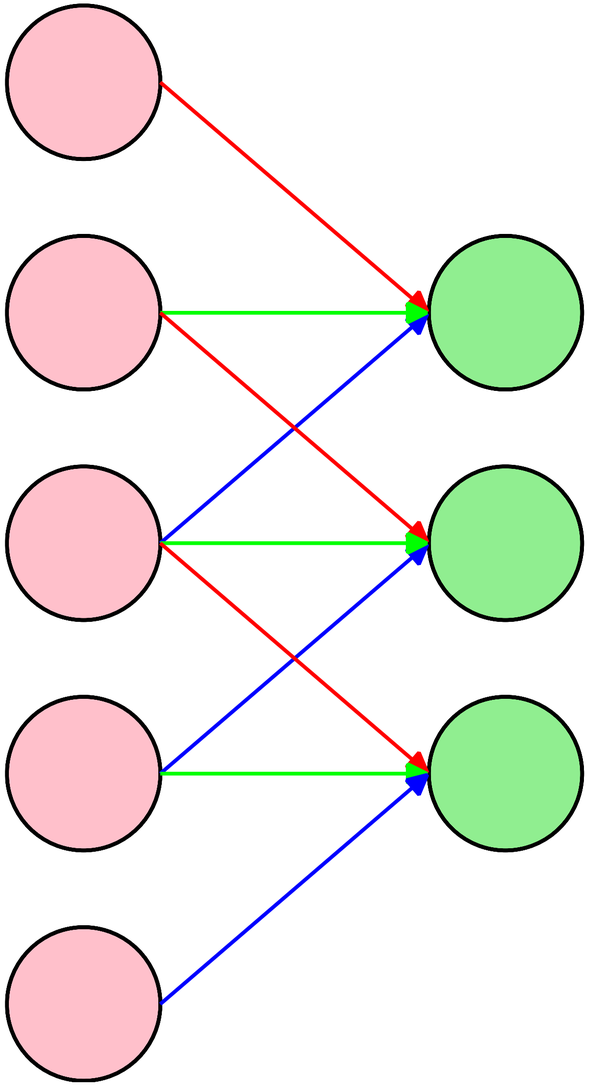


Chain Rule Magic

$$\frac{\delta e(w)}{\delta w_{ij}^{(1)}} = \frac{\delta e(w)}{\delta \theta}\frac{\delta \theta}{\delta s}\frac{\delta s}{\delta \theta}\frac{\delta \theta}{\delta s}\frac{\delta s}{\delta \theta}\frac{\delta \theta}{\delta s}\frac{\delta s}{\delta w}$$

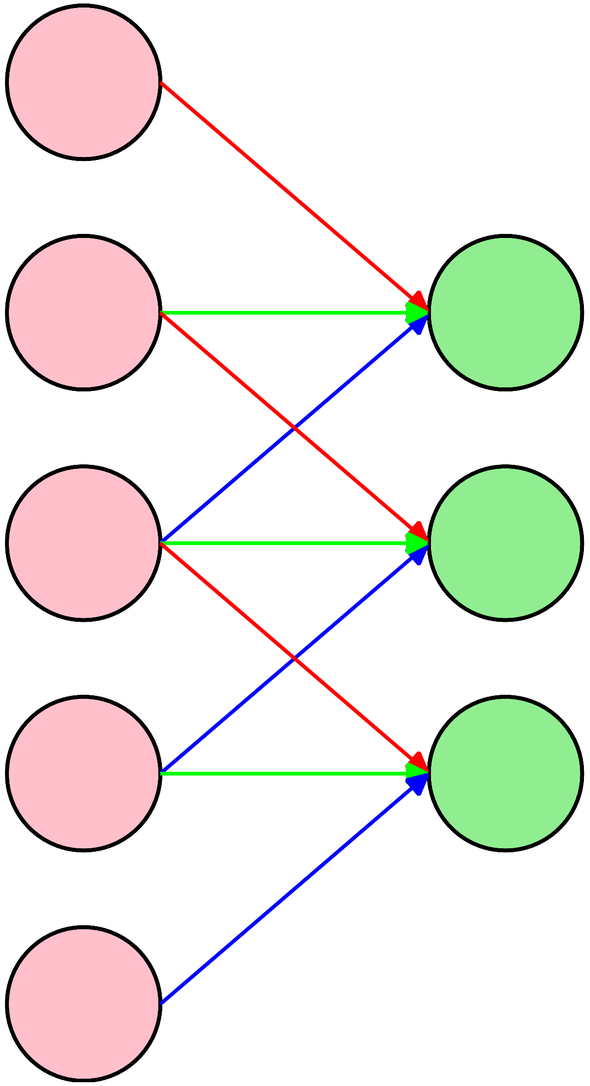


Convolutional Neural Networks



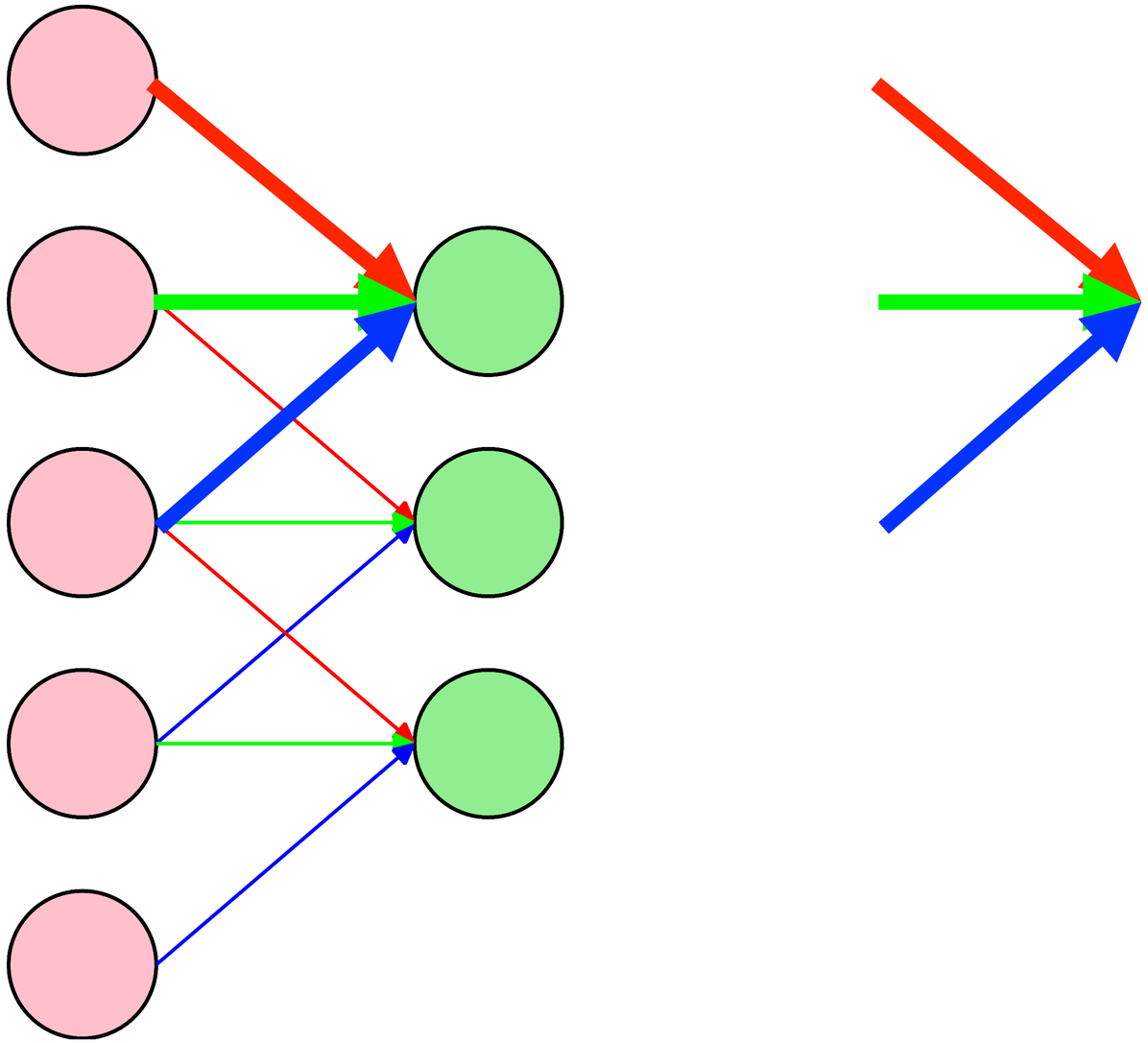
- Shared weights used across network
- May mimic a behavior of particular importance to the visual cortex of the brain

Convolutional Neural Networks



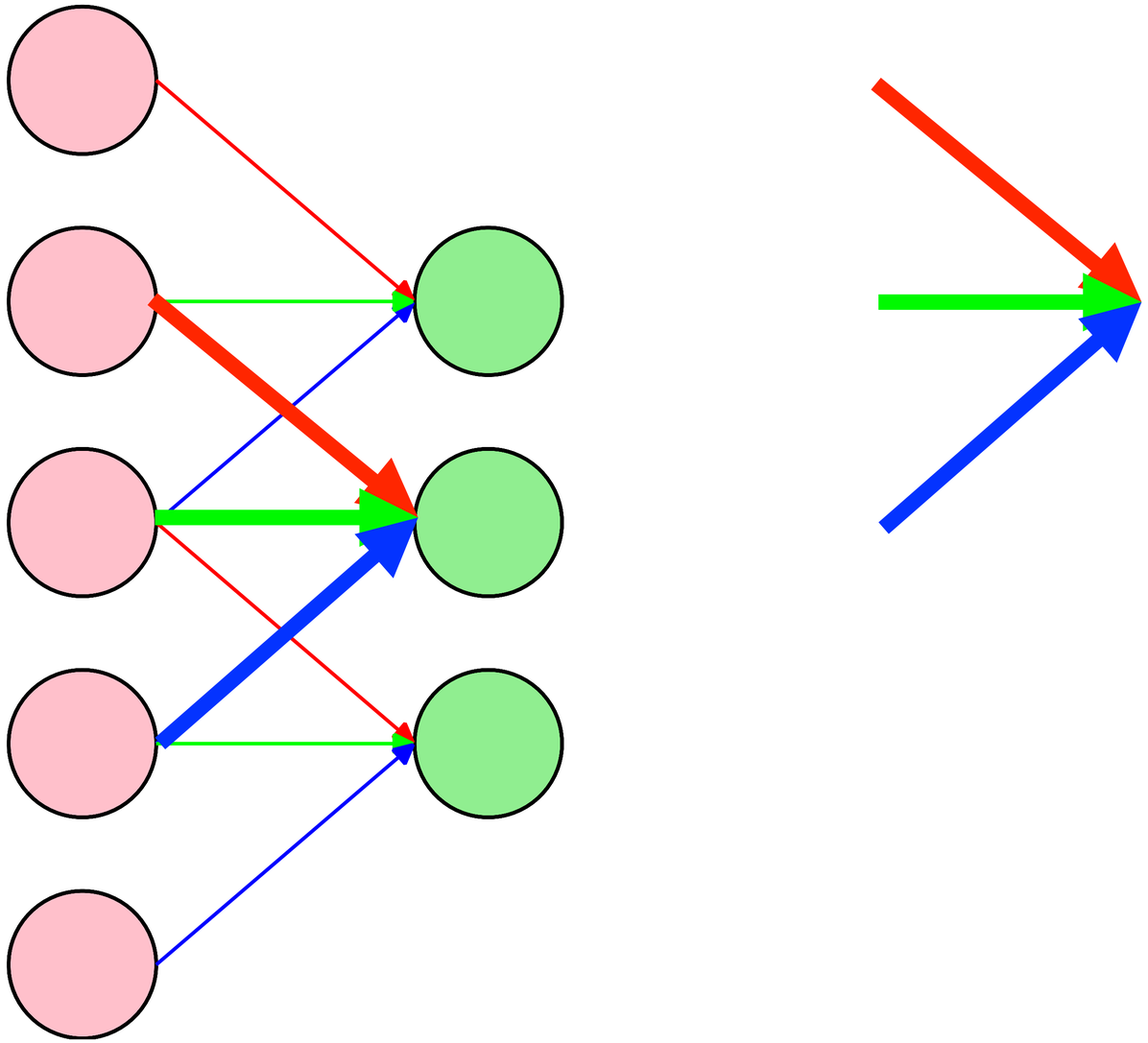
- Shared weights used across network
- May mimic a behavior of particular importance to the visual cortex of the brain
- Note that we have lost a degree of freedom. All those weights of the **same** color must now have the same weight
- We also reduce memory, computation and optimization problem size :)

Convolutional Neural Networks



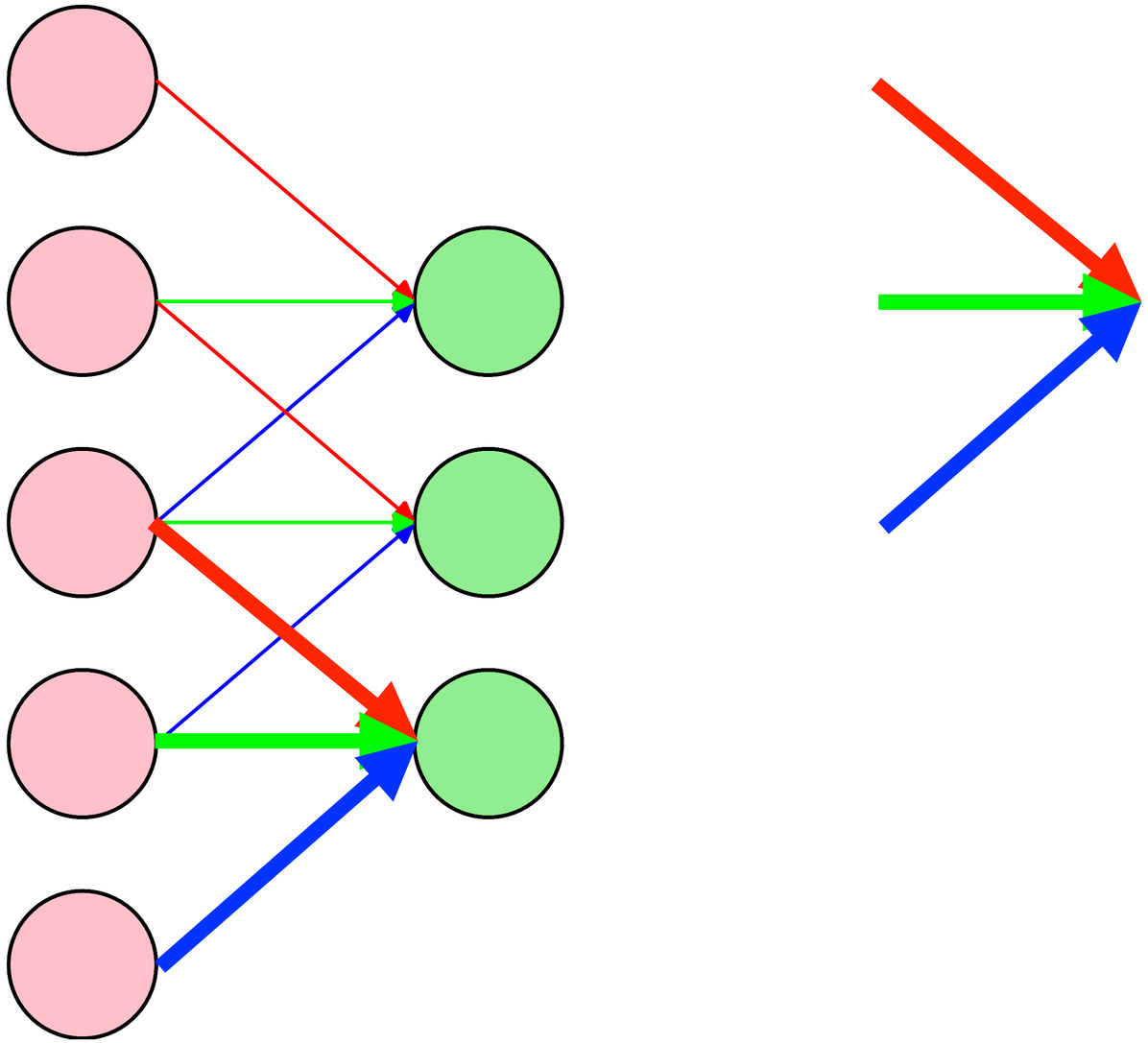
- Think of these 3 weights as a **kernel** being applied by **convolution** over the input **spatially**

Convolutional Neural Networks



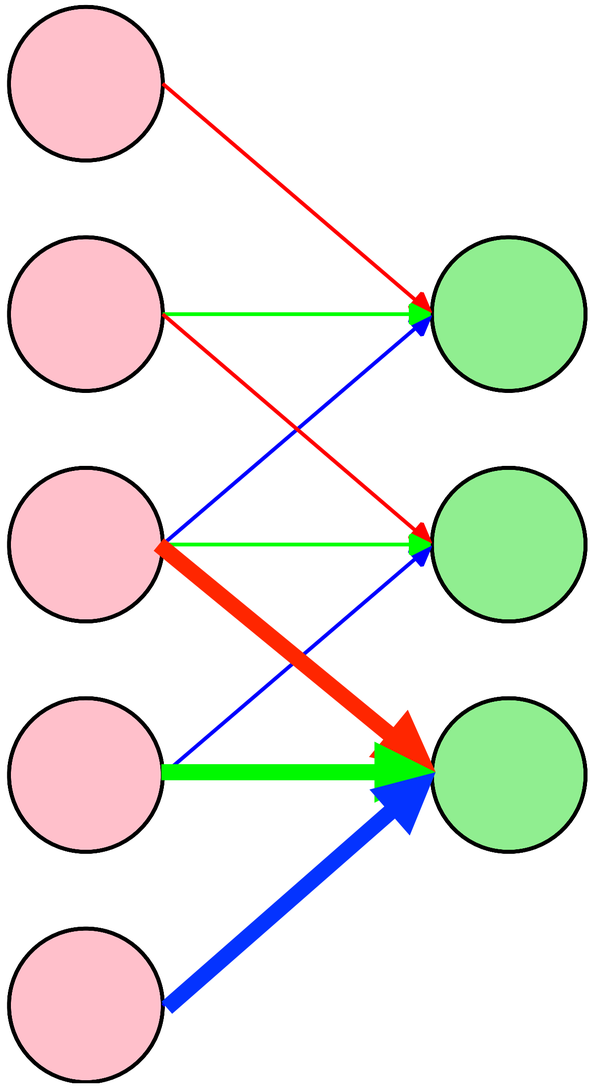
- Think of these 3 weights as a **kernel** being applied by **convolution** over the input **spatially**

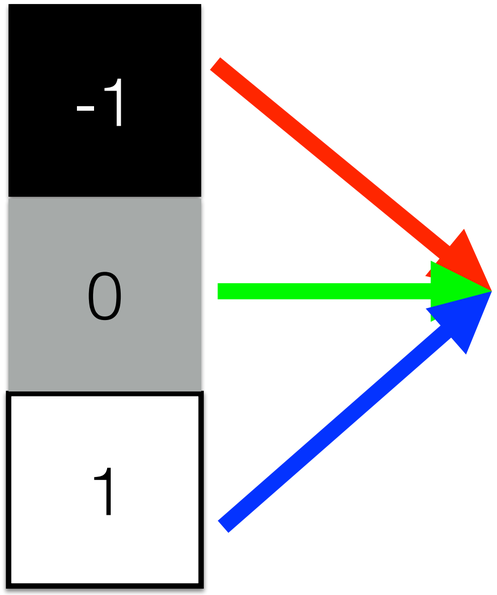
Convolutional Neural Networks



- Think of these 3 weights as a **kernel** being applied by **convolution** over the input **spatially**

CNN Kernel's as "Features"





Example Edge Detection Kernel

- Each CNN Kernel can be thought of as a convolution filter or **feature**
- Only, we do not design it! Gradient descent will determine what is 'learnt'.
- Low level features tend to resemble popular features such as; gradients, edges, corners, centre-surround features and so on…

What is the purpose of image features?

- Encapsulate Prior Knowledge : e.g. symmetry, texture
- Statistical Invariance : e.g. illumination invariance, scale invariance etc.
- Simplification : e.g. Haar Wavelets
- Computational Optimization : e.g. integral images, LiteHOG, etc.
- Dimensionality reduction : raw pixels suffer the curse of dimensionality : e.g. SVM & HOG pedestrian detection (Dalal & Triggs)

Impressive Results

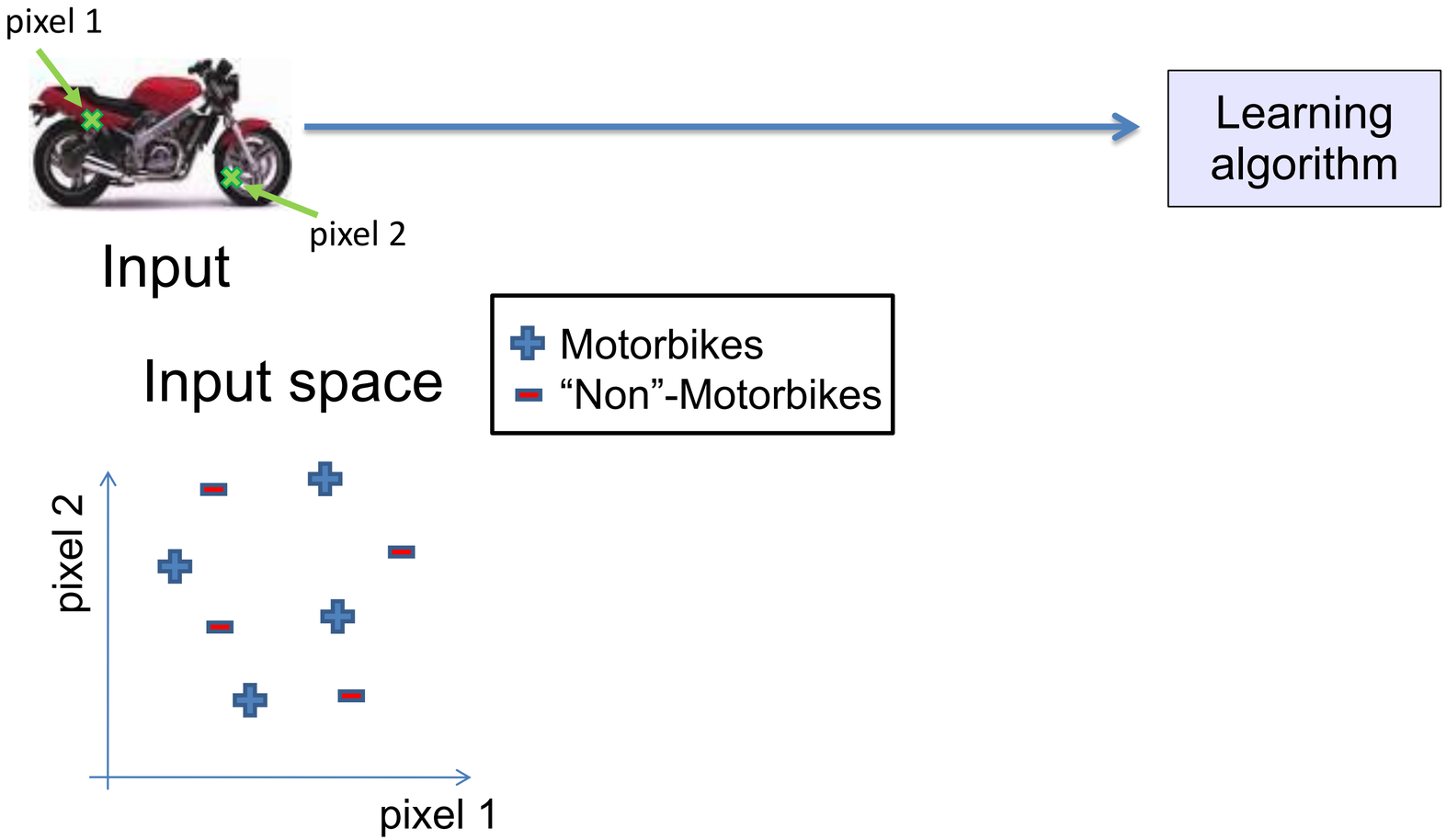
- Classifying handwritten digits (MNIST)
  - Better than humans
- Chinese character recognition
- Traffic Sign Recognition 99% accurate
  - Better than humans
- Winning entries for ImageNet Challenge

Learning Feature Representations

- In computer vision Deep Learning is very much about learning good feature representations (and its very impressive at learning these!)
  - w/ some qualification
  - needs lots of data
  - longer training times (than say SVM + <your fav. feature>)
  - really needs GPU compute power

Slides shamelesslv stolen from CVPR2012 tutorial

Feature representations



"Deep Learning Methods for Vision", Honglak Lee, University of Michigan, CVPR2012

Slides shamelessly stolen from CVPR2012 tutorial

Feature representations

Input

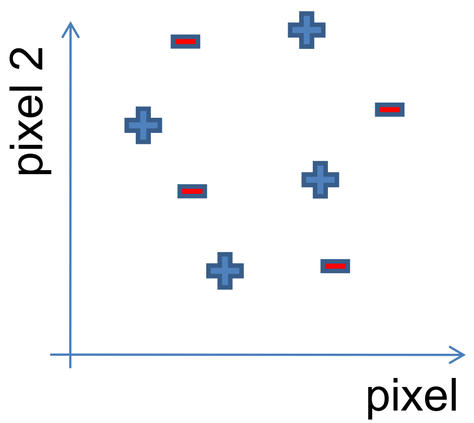
Feature representation

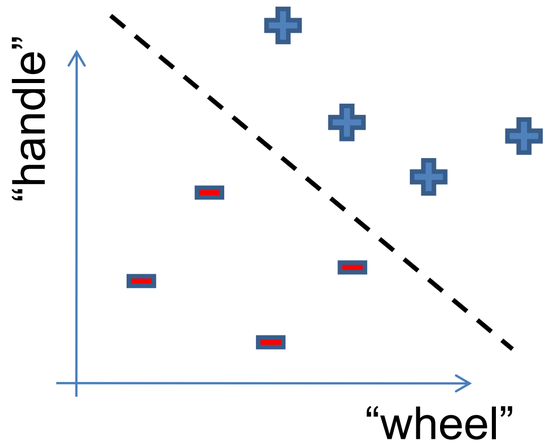
Learning algorithm

Input space

Motorbikes
"Non"-Motorbikes

Feature space



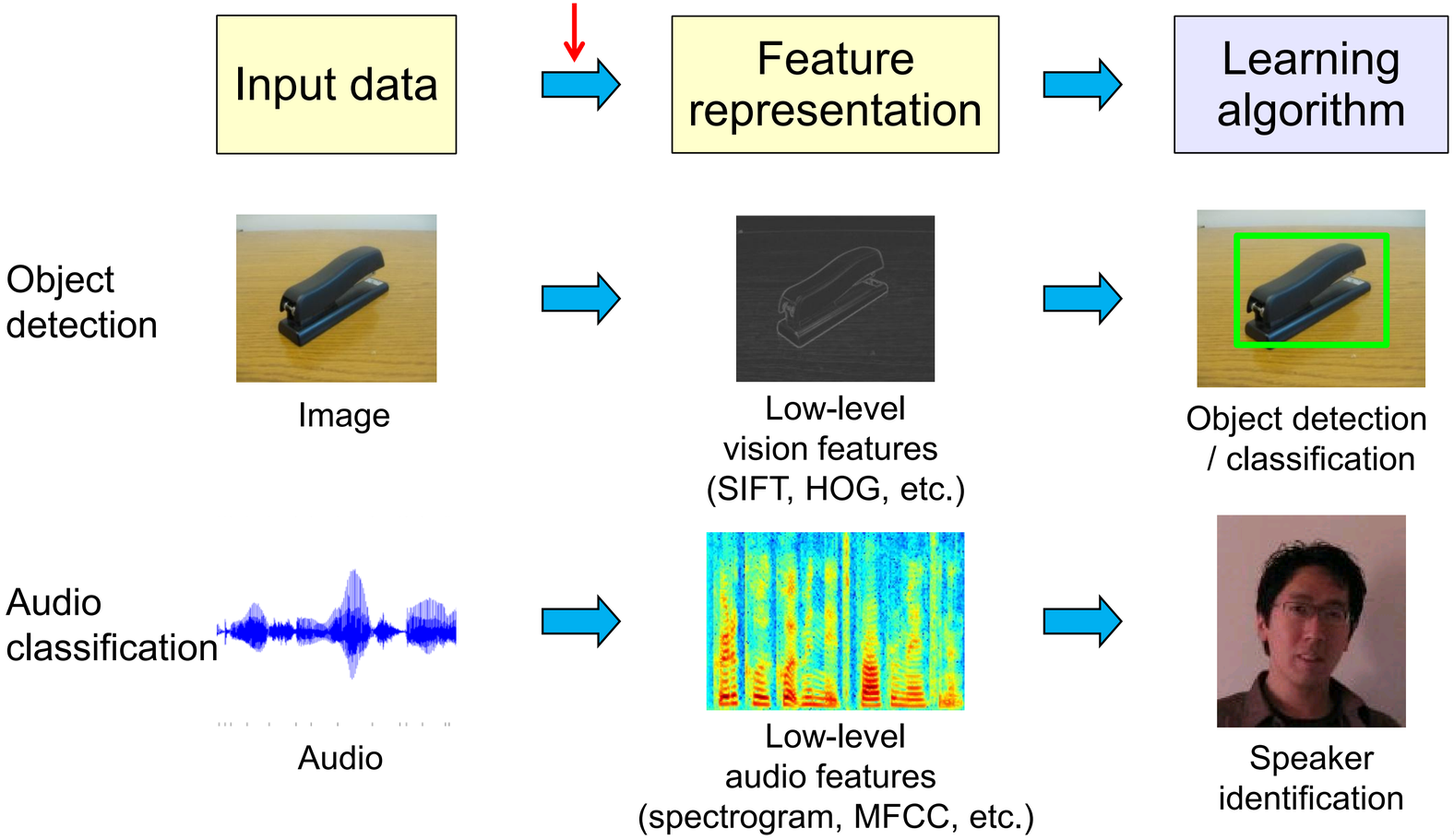


"Deep Learning Methods for Vision", Honglak Lee, University of Michigan, CVPR2012

Slides shamelessly stolen from CVPR2012 tutorial

How is computer perception done?

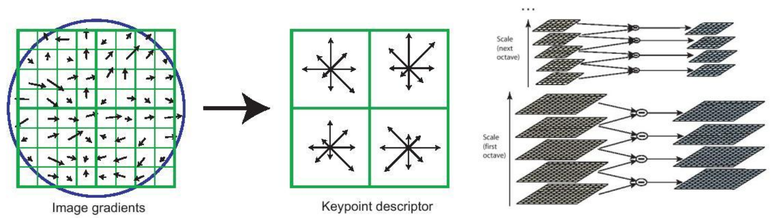
State-of-the-art:
"hand-crafting"



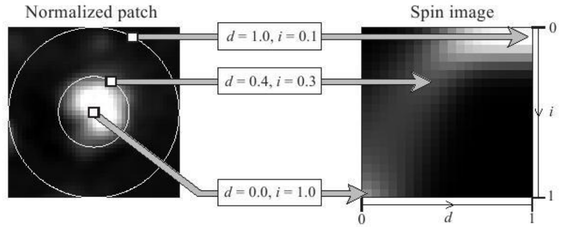
"Deep Learning Methods for Vision", Honglak Lee, University of Michigan, CVPR2012

Slides shamelessly stolen from CVPR2012 tutorial

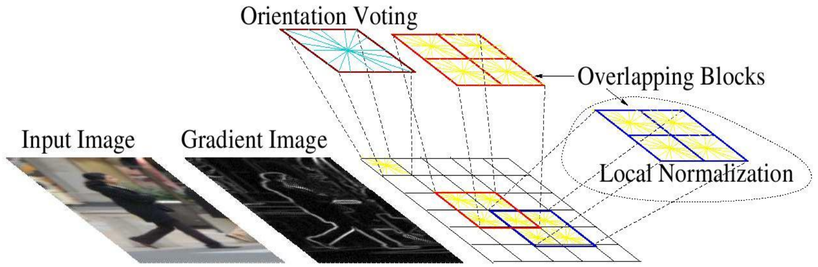
Computer vision features



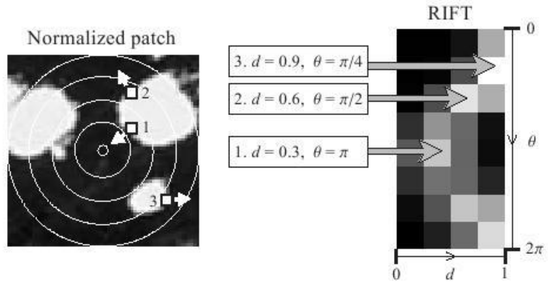
SIFT



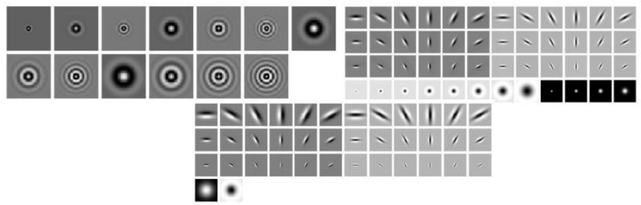
Spin image



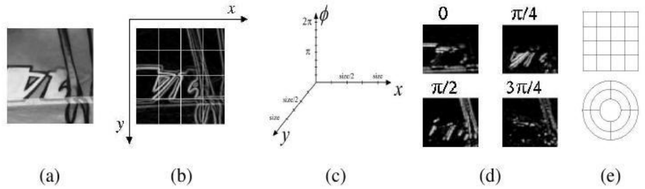
HoG



RIFT



Textons

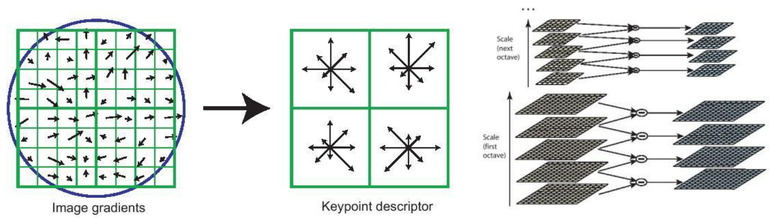


GLOH

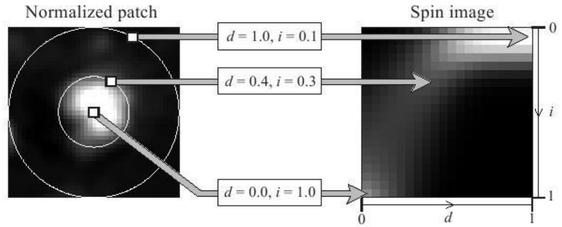
"Deep Learning Methods for Vision", Honglak Lee, University of Michigan, CVPR2012

Slides shamelessly stolen from CVPR2012 tutorial

Computer vision features



SIFT



Spin image

Hand-crafted features:

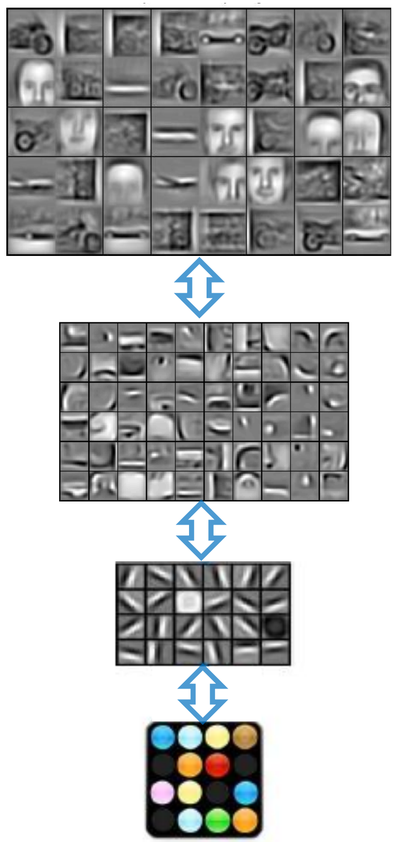
1. Needs expert knowledge
2. Requires time-consuming hand-tuning
3. (Arguably) one of the limiting factors of computer vision systems

"Deep Learning Methods for Vision", Honglak Lee, University of Michigan, CVPR2012

Slides shamelessly stolen from CVPR2012 tutorial

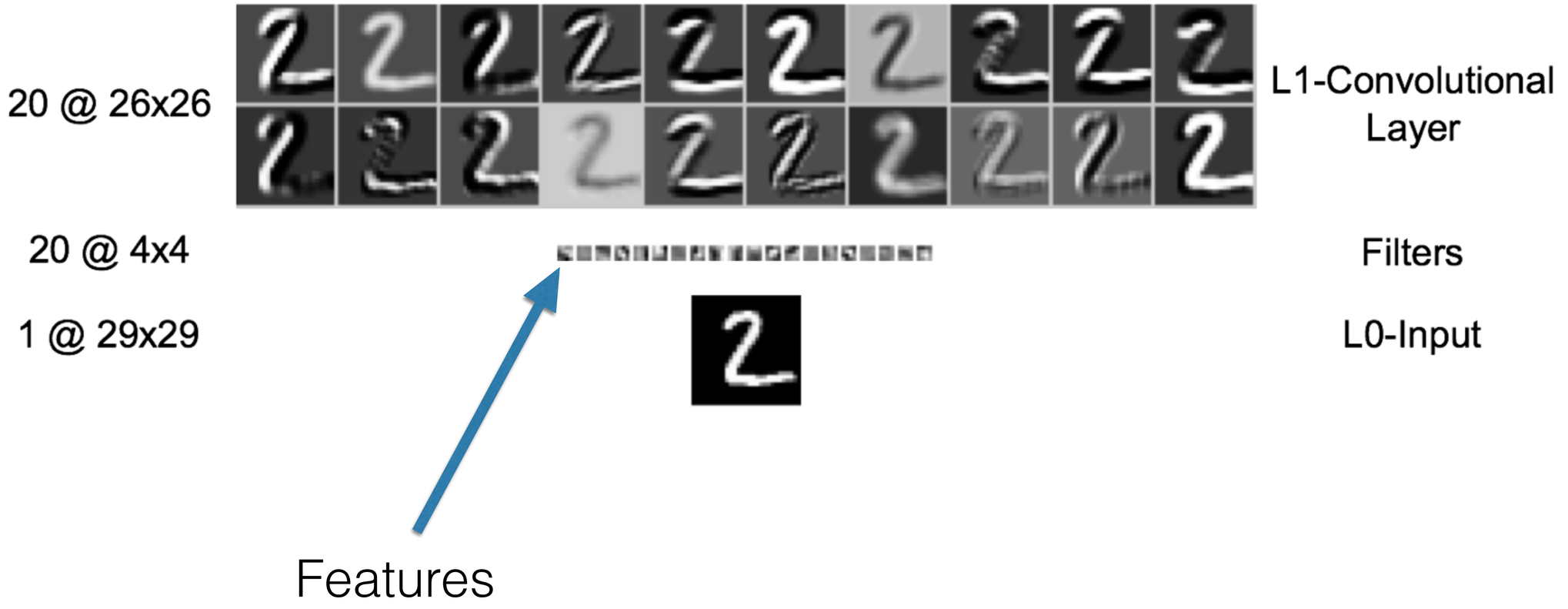
Learning Feature Hierarchy

- Deep Learning
  - Deep architectures can be representationally efficient.
  - Natural progression from low level to high level structures.
  - Can share the lower-level representations for multiple tasks.

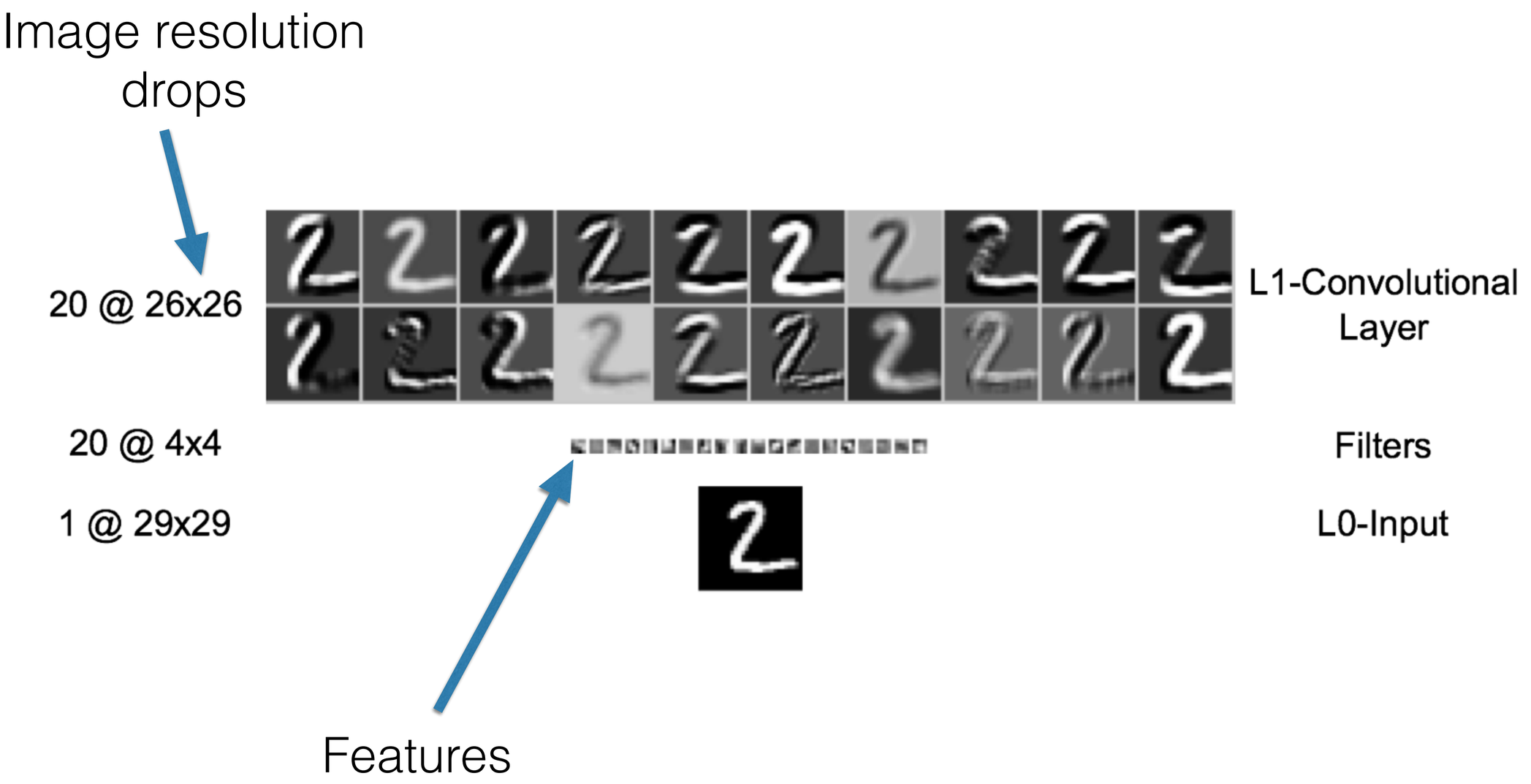


“Deep Learning Methods for Vision”, Honglak Lee, University of Michigan, CVPR2012

MNIST Example : Initial Layer



MNIST Example : Initial Layer



MNIST Example : Initial Layer

Image resolution drops

Exploding Data!

L1-Convolutional Layer

20 @ 26x26

But I have grown by a factor of 20
20 - “Feature Maps”

20 @ 4x4

Filters

1 @ 29x29

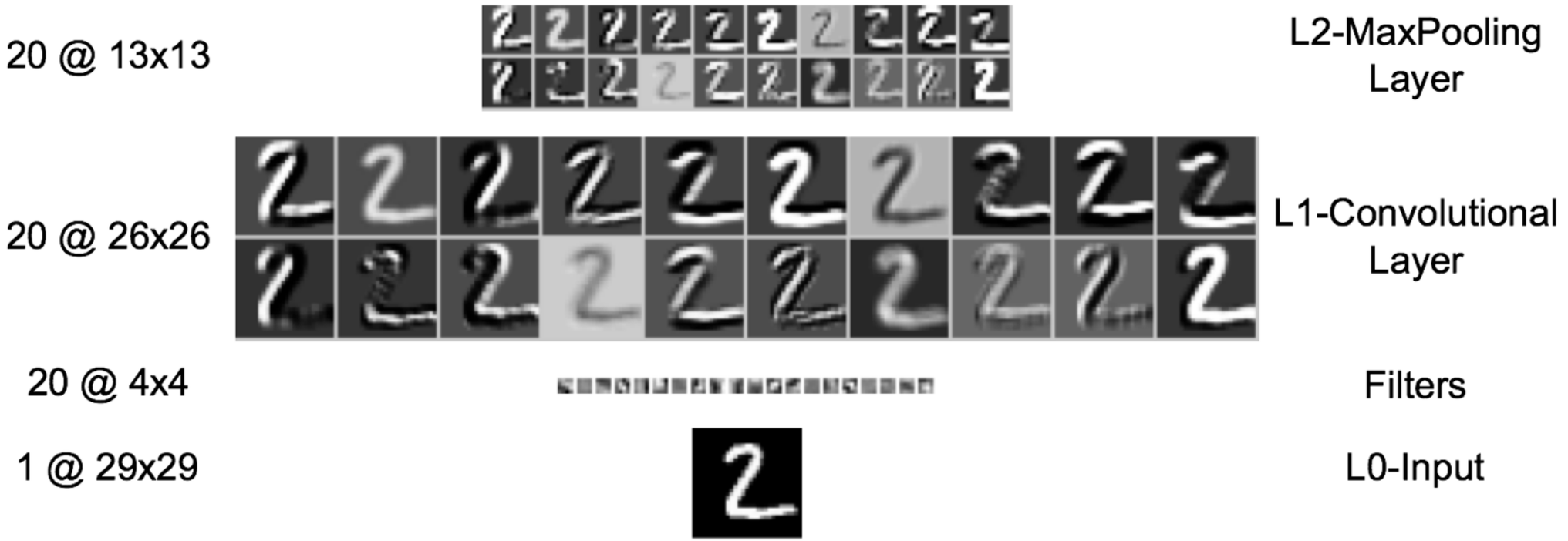
L0-Input

Features

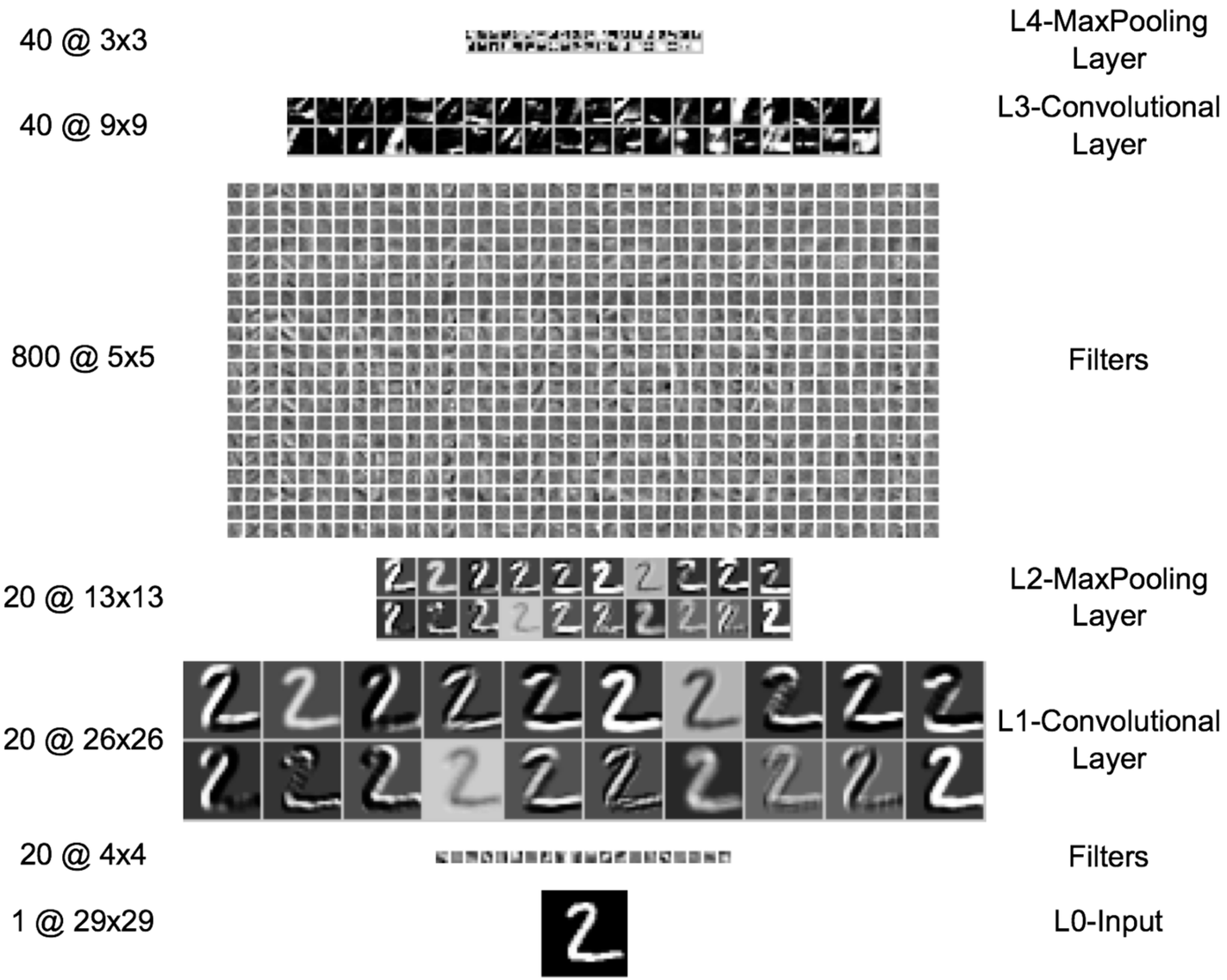
Pooling as Data Reduction

- The use of multiple filters leads to an explosive growth in the amount of data being pushed forward through the network.
- Therefore we use “Pooling” methods to reduce (or summarize) the data. There are many kinds of pooling:
  - Average Pooling - just scale the output e.g. 2x2 “pixel” region -> 1x1
  - Max Pooling - pass forward the strongest response e.g. strongest response in a 2x2 “pixel”

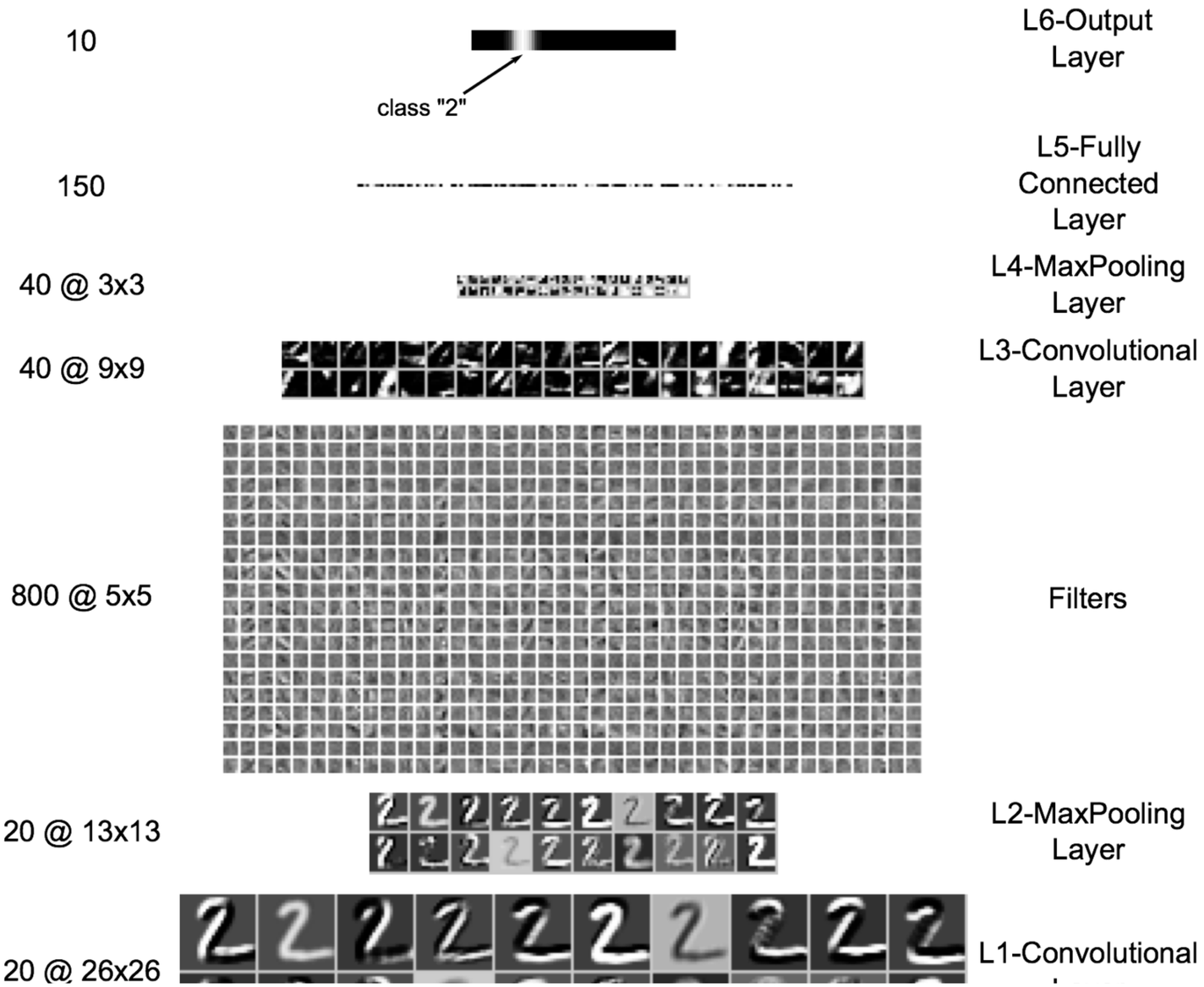
Pooling


20 @ 13x13
L2-MaxPooling Layer
20 @ 26x26
L1-Convolutional Layer
20 @ 4x4
Filters
1 @ 29x29
L0-Input

Alternating Feature & Pooling Layers



Fully Connected Final Layer

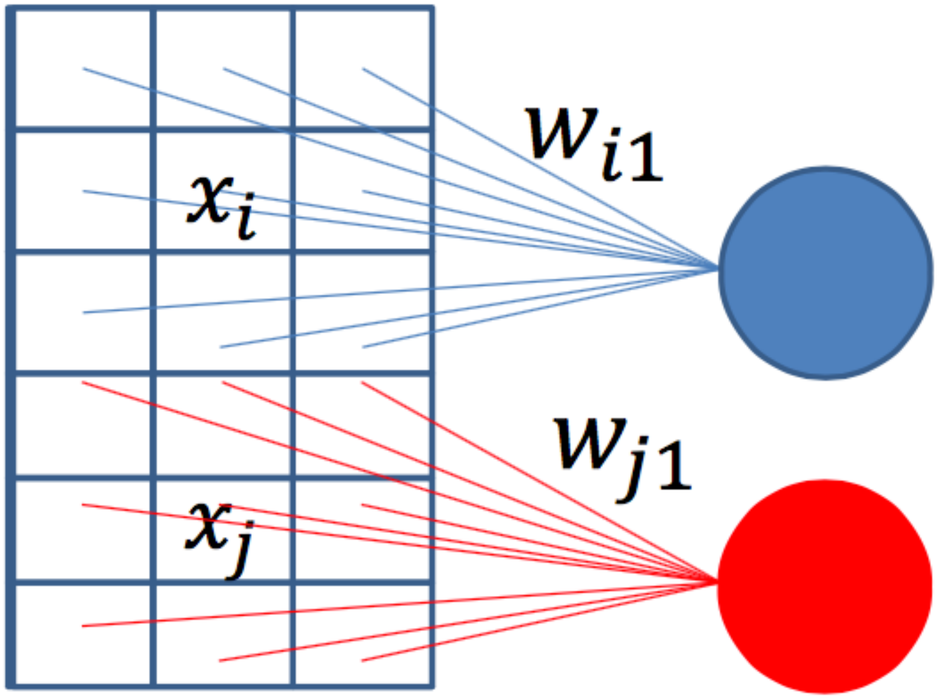


Different Types of Hidden Nodes

- Feature Convolutional Nodes
- Pooling Nodes
- Fully Connected Layers

Convolutional Nodes : 3x3 example

Same weights for i and j: $W_{ik} = W_{jk} = W_k$

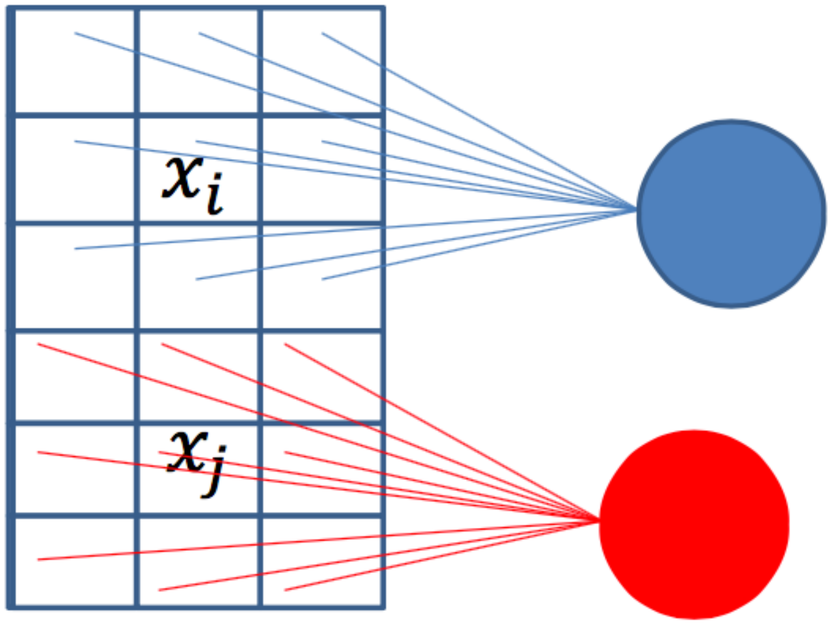


$$y_i = \phi\left(\sum_{k \in N(i)} w_k x_k\right)$$

$$y_j = \phi\left(\sum_{k \in N(j)} w_k x_k\right)$$

Pooling Nodes

- Combines the values of nodes from the previous layers without computing new features
- Example: Max pooling:
  - Take the max of the nodes from the previous layers in $K \times K$ neighborhood N . .
  - Reduce the number of nodes
- Others: L2 pooling, ...



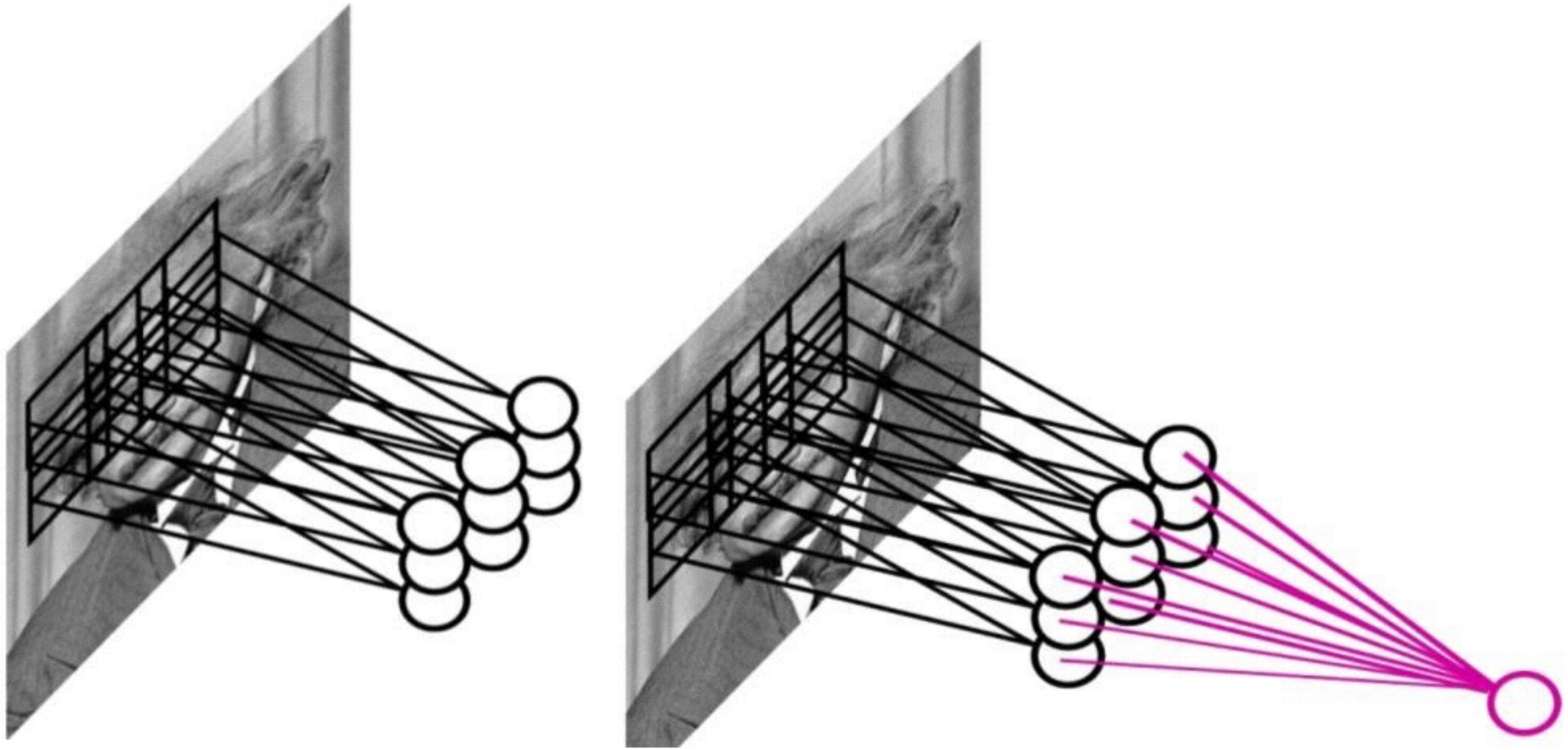
$$y_i = \max_{k \in N(i)} x_k$$

$$y_j = \max_{k \in N(j)} x_k$$

Example: R Fergus

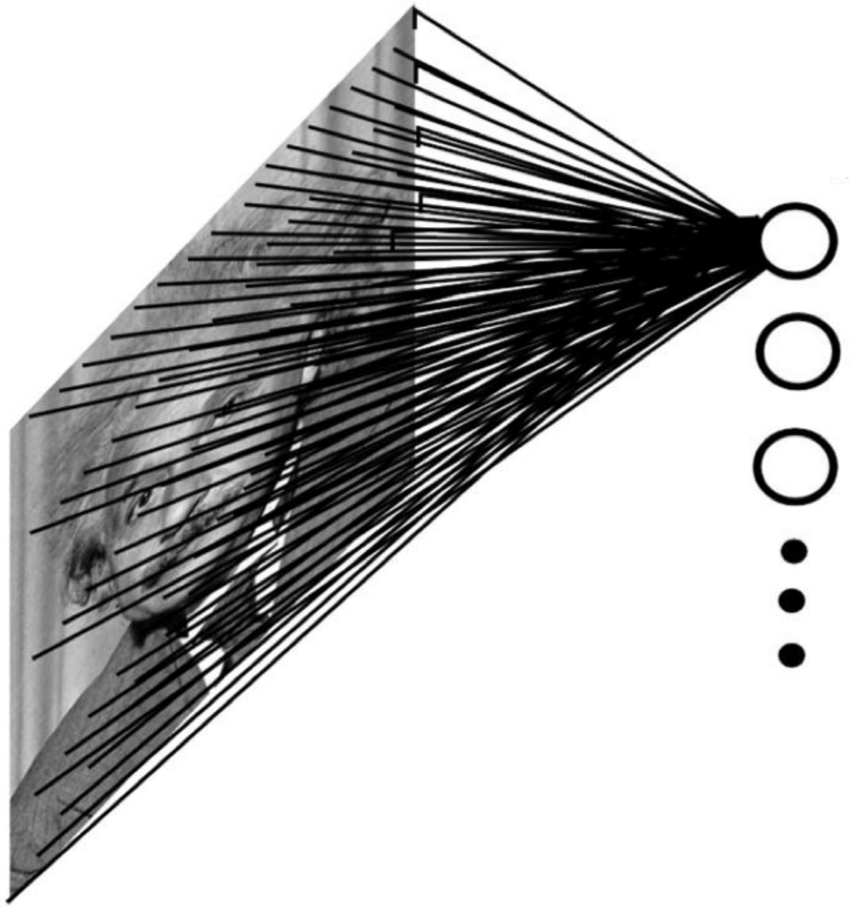
Convolution

Pooling



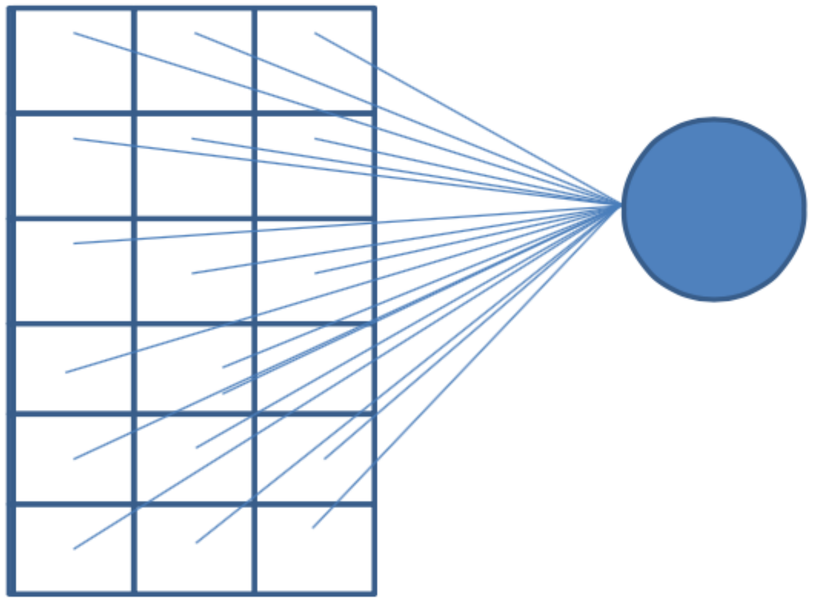
Example: R Fergus

Fully Connected



Fully Connected Node

- All the nodes from the previous layer are connected to the hidden units
- All with different weights
- Very large number of connections (N x M): This is used only in the last levels when the “image” has been reduced sufficiently



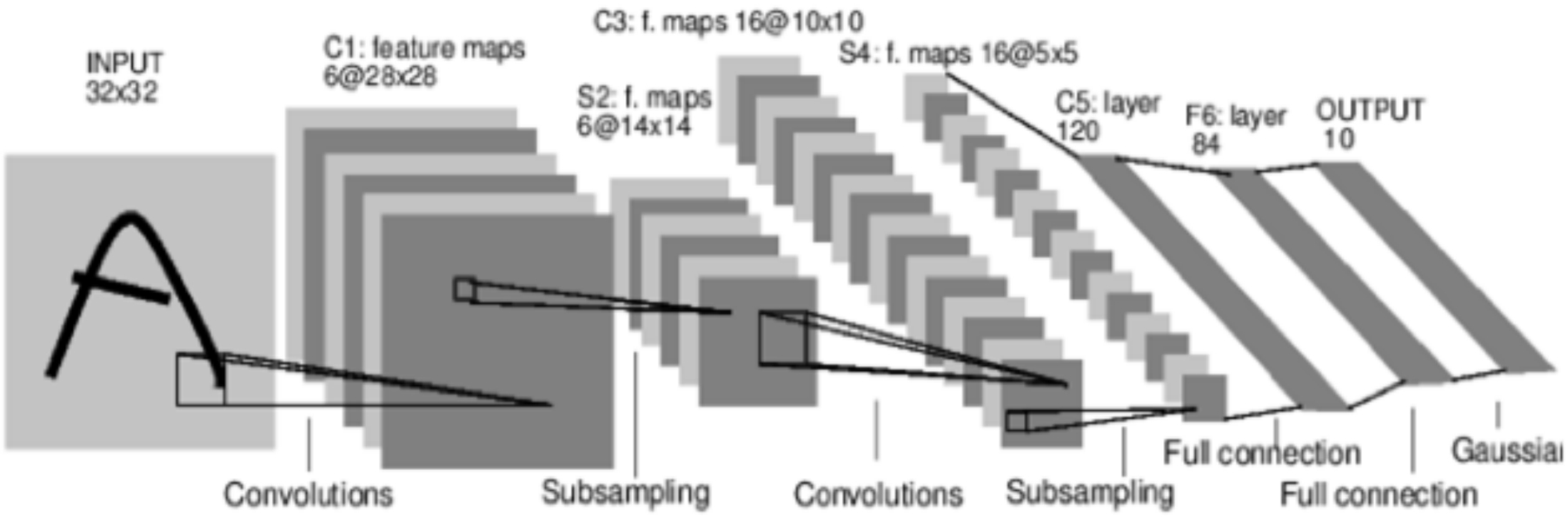
$$y_i = \phi\left(\sum_k w_k x_k\right)$$

Training

- Learn weights by minimizing error (or maximizing similarity) on training sample
- Output z is function of input x and W = vector of weights
- Minimize loss between output z and true value of training sample $L(z(x, W), \hat{z})$
- Backpropagation:
  - Estimate change on weights ΔW from first order derivative
    $\Delta L(z(x, W), \hat{z}) \approx \cdots \frac{\partial z(x,W)}{\partial W} \Delta W$
  - (Complicated) derivative computed with repeated chain rule
- Stochastic gradient:
  - We should use the sum over all the samples:
    $L(W) = \sum_x L(z(x, W), \hat{z}) \qquad W \leftarrow W - \alpha \nabla L(W)$
  - Intractable because of number of samples
  - Instead update on samples chosen randomly:

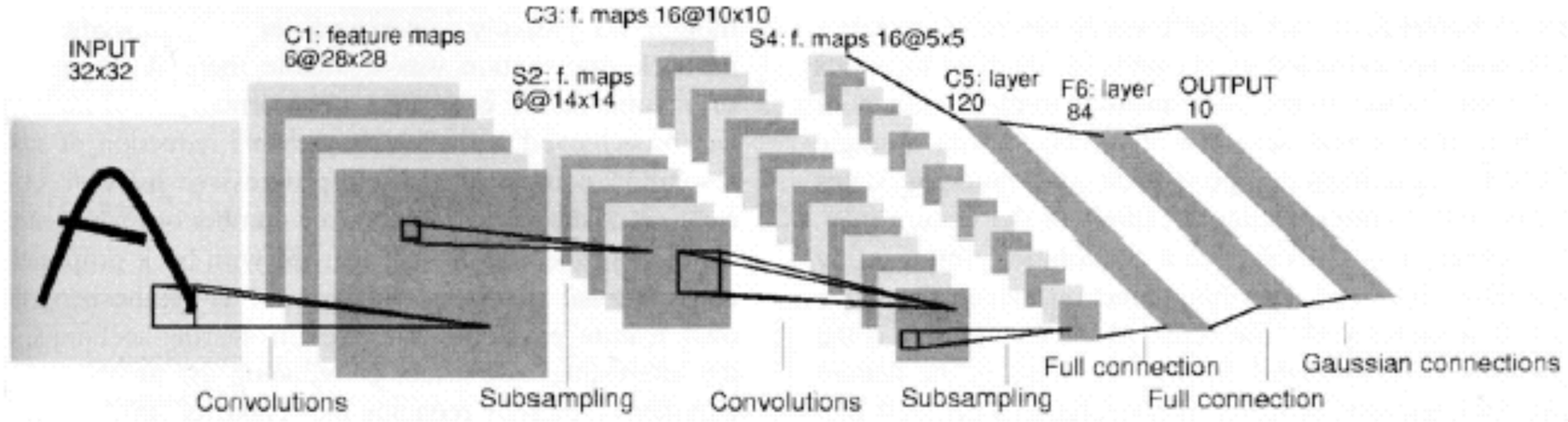
$$W \leftarrow W - \alpha \nabla L(z(x, W), \hat{z})$$

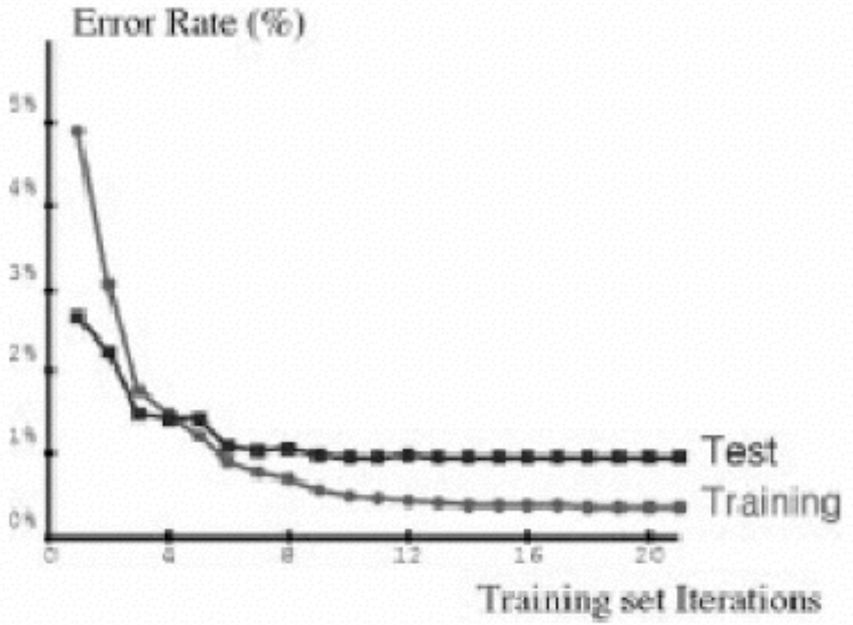
Example : Lecun Digit Recognition



~350,000 connections total 60,000 parameters

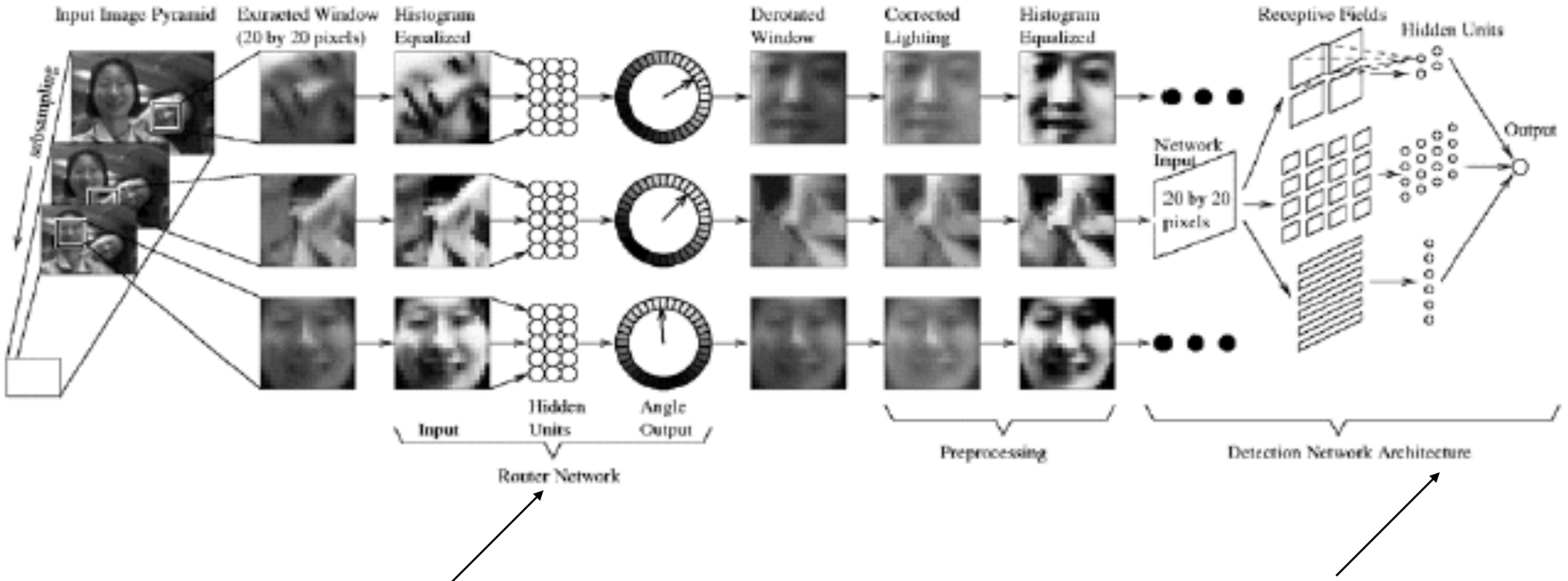
Example : Lecun Digit Recognition





Convolutional Networks: Lecun et al.
http://yann.lecun.com/exdb/lenet/

(Classic) Example: Face detection



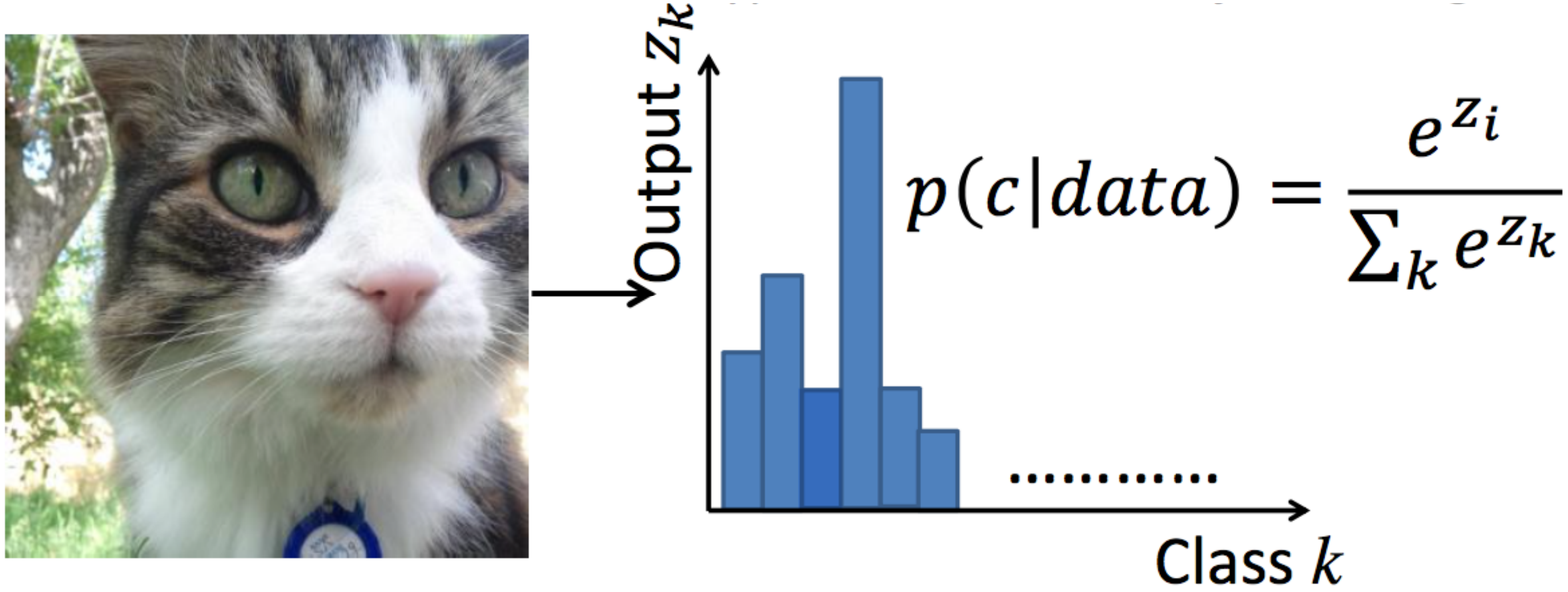
f(x) = orientation of template sampled at 10° intervals

f(x) = face detection posterior

Example from Rowley et al.

ImageNet Classification

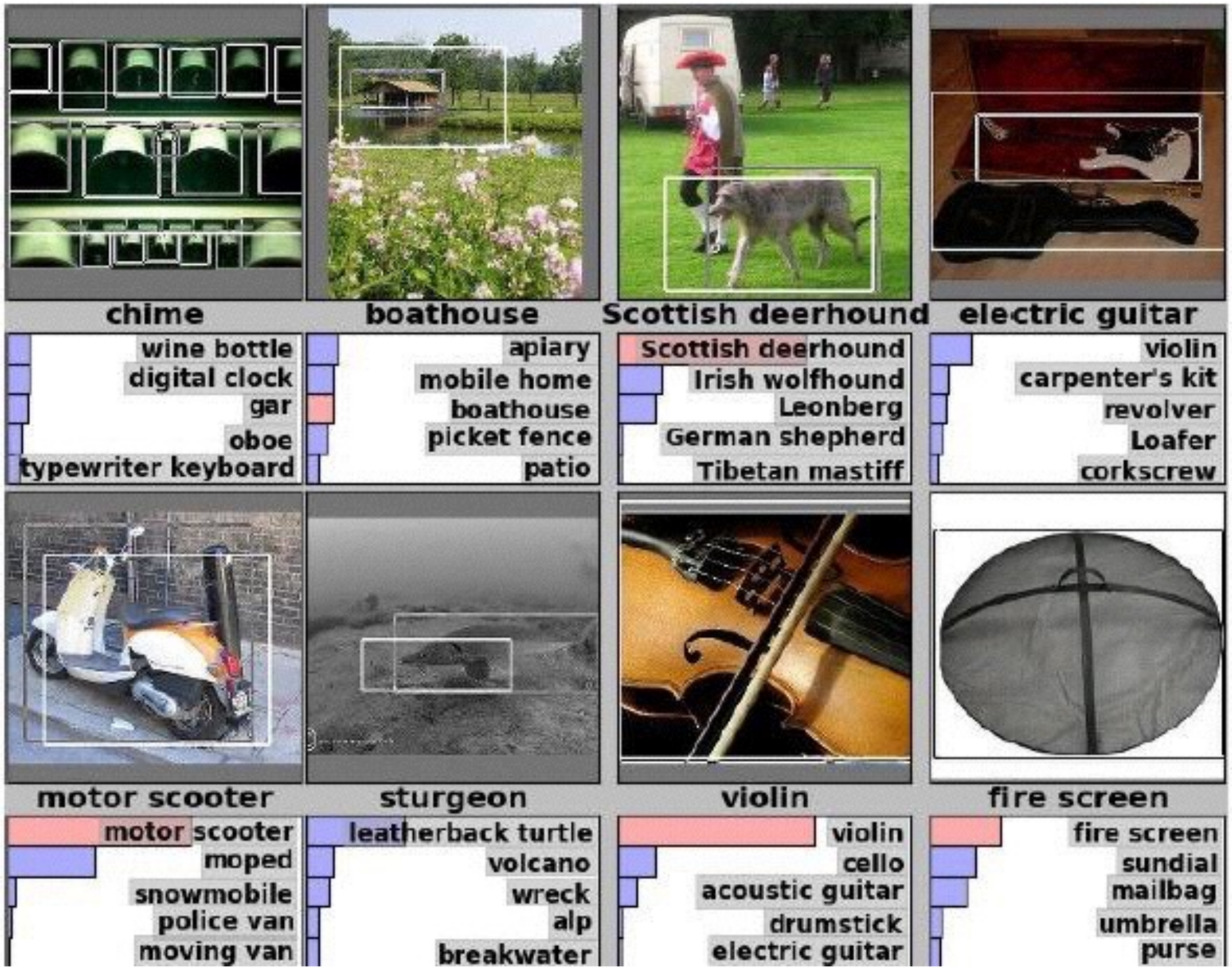
- Imagenet dataset (1000 classes!)
- Input: Image, output: Probability of object type in image (logistic model)
- Classification: Classify image type
- Localization/detection: Find type and location of object in image



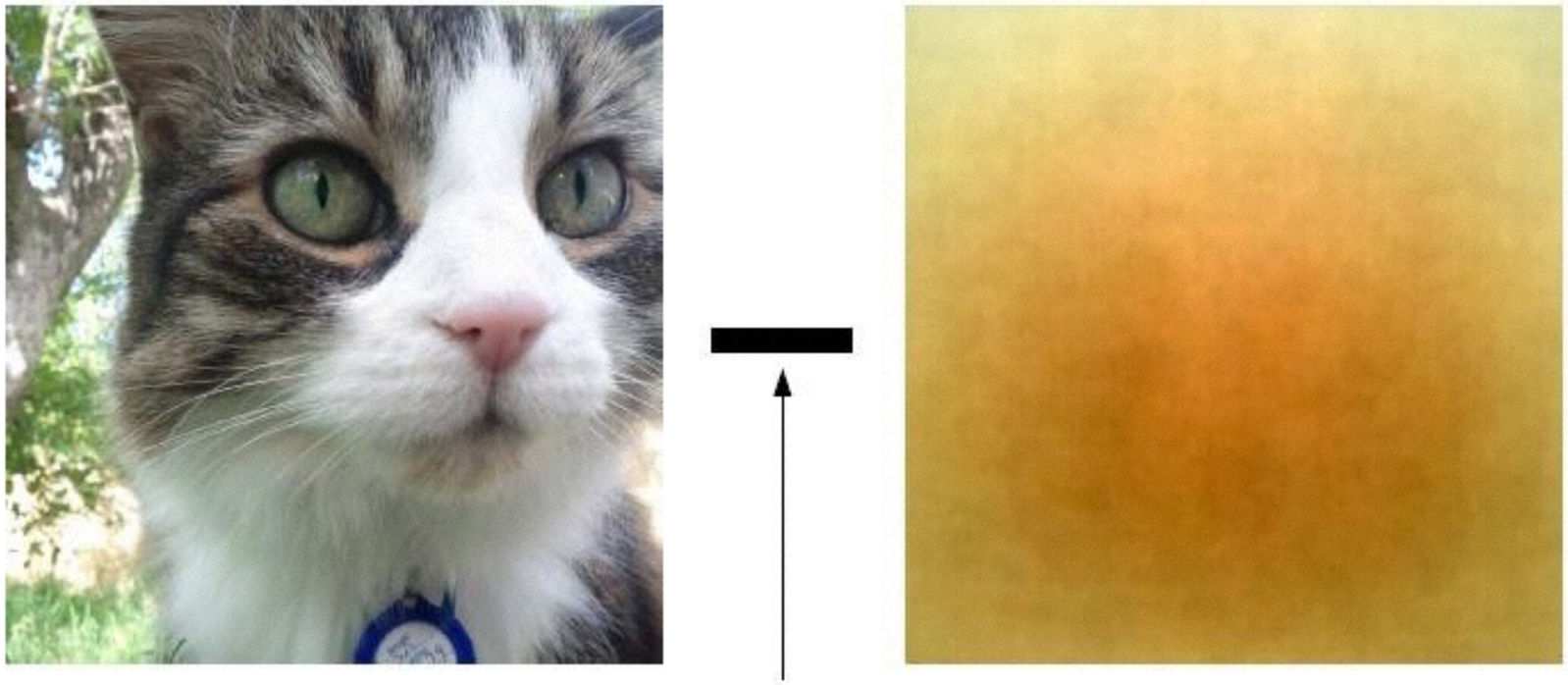
Krizhevsky, A., Sutskever, I. and Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012.

Classification Examples

Detection/Localization Examples

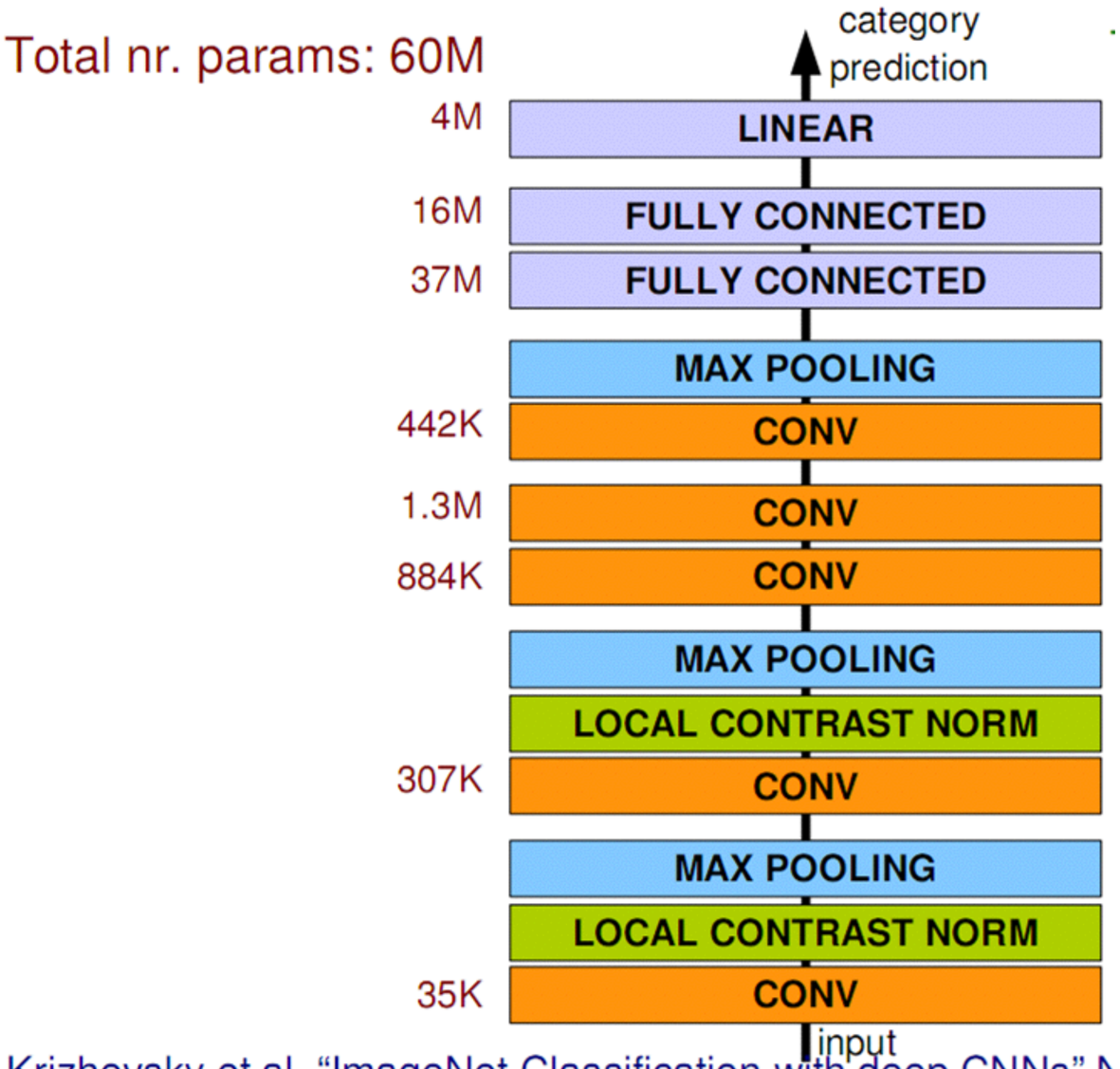


Input Preparation



Normalize Inputs by subtracting the mean of all training images

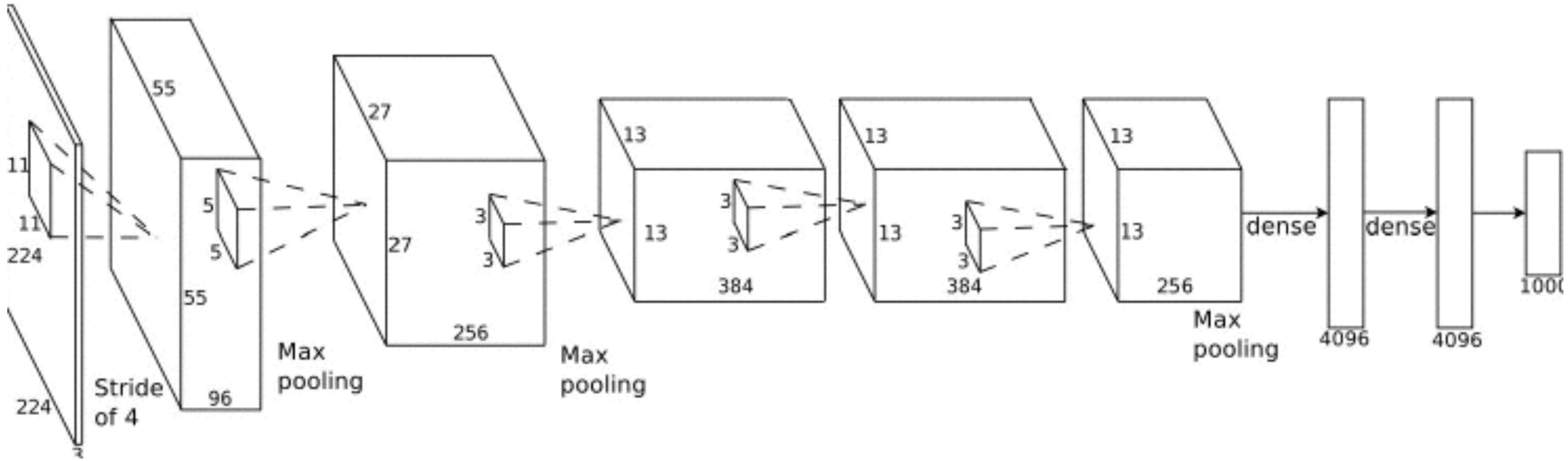
Full Architecture



Krizhevsky et al. “ImageNet Classification with deep CNNs” N

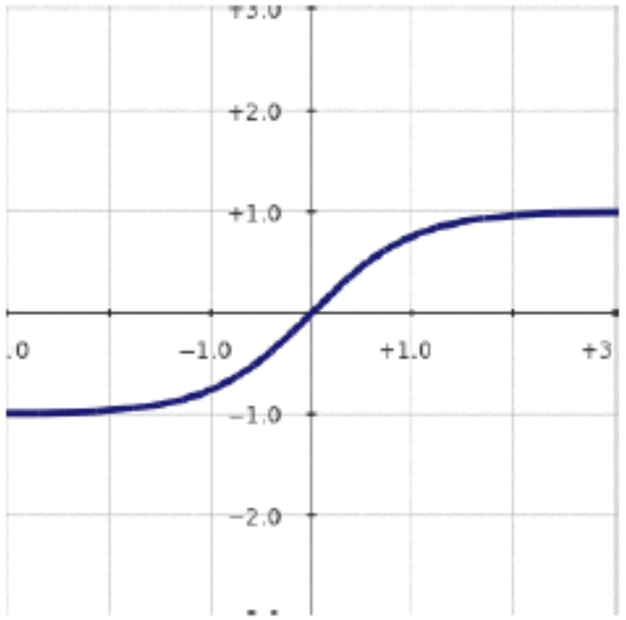
Architecture “AlexNet”

- 5 convolution layer
- 3 max pooling
- 2 fully connected
- Total number of nodes= 253440+186624+64896+64896+43264+4096+4096+10 00=650K
- 630 million connections
- 60 million parameters (!!)

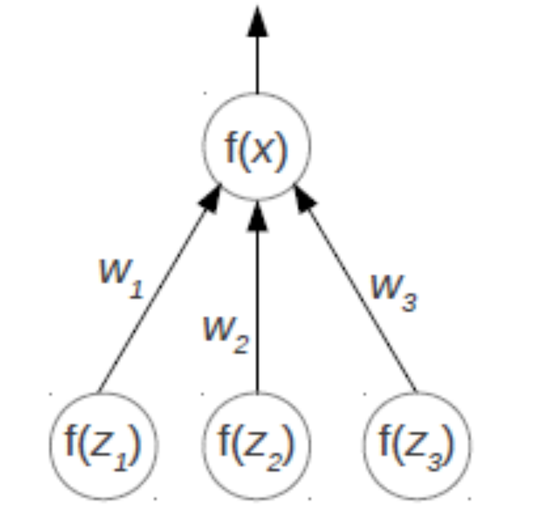


Choice of Non-Linearity

$f(x) = \tanh(x)$



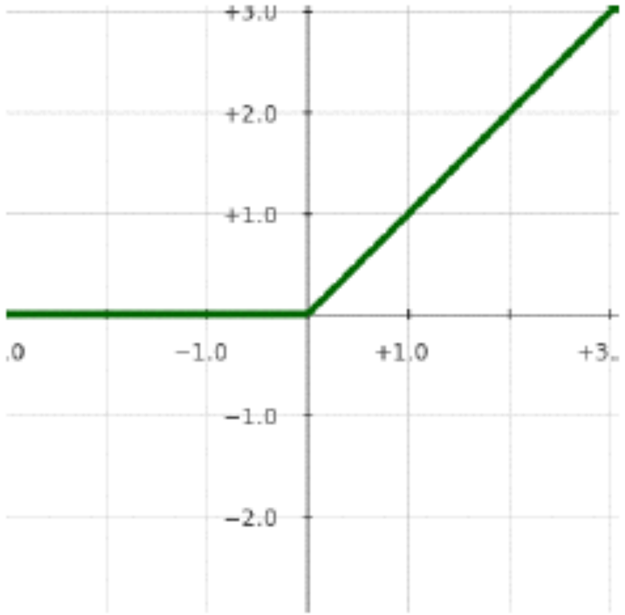
Hard to Train



$x = w_1 f(z_1) + w_2 f(z_2) + w_3 f(z_3)$

x is called the total input to the neuron, and $f(x)$ is its output

$f(x) = \max(0, x)$

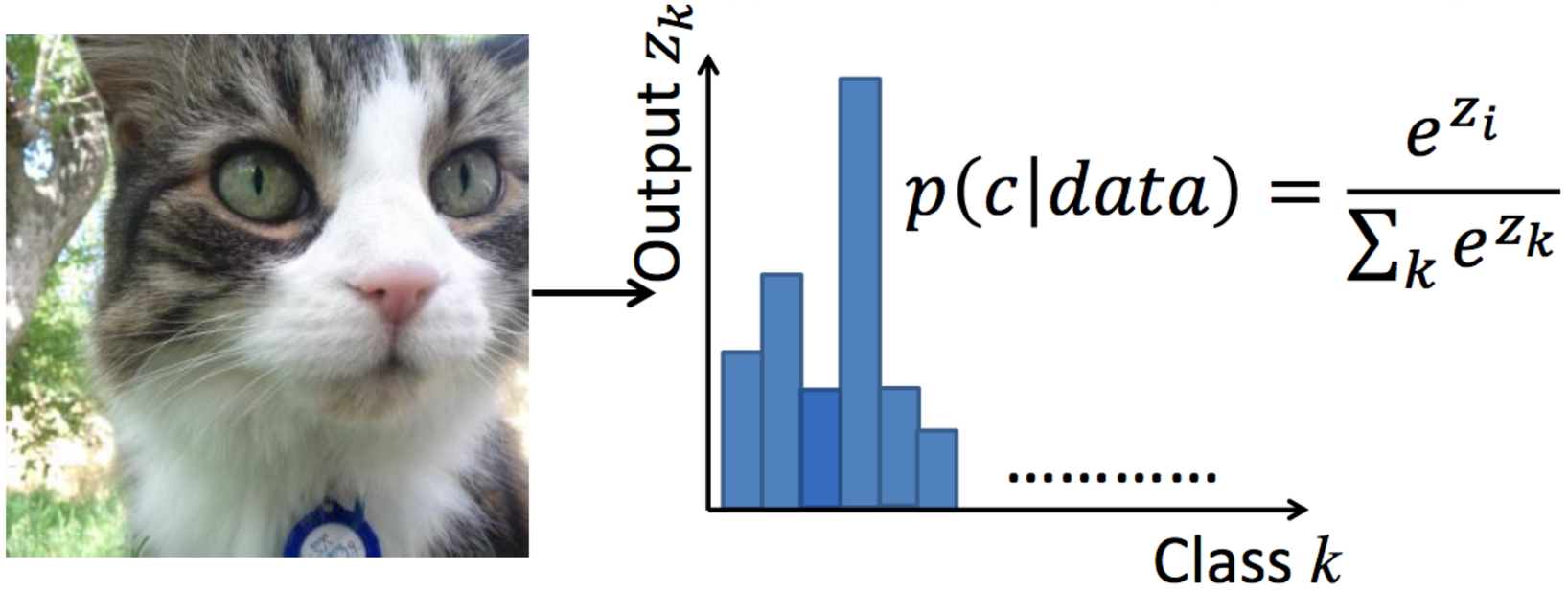


Easy to Train

Empirical “Fact” no real justification!

Training

- Initialization: Random (Gaussian noise) values for the parameters
- Backpropagation to maximize **log probability** of true class
- Stochastic gradient descent



Overfitting Issues

- We need to learn 60 million parameters!!
- Limited training data
- Will lead to massive overfitting
- Two tricks (essential):
  - Data augmentation
  - Dropout

Data Augmentation

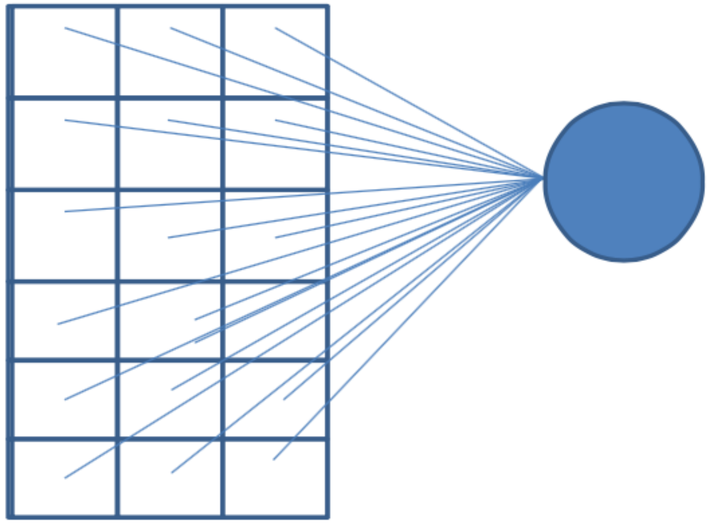
- In all systems: Essential to create additional training data to avoid overfitting to limited samples

- Example:
  – Train on 224x224 patches extracted randomly from 256x256

- images
  – Horizontal reflections

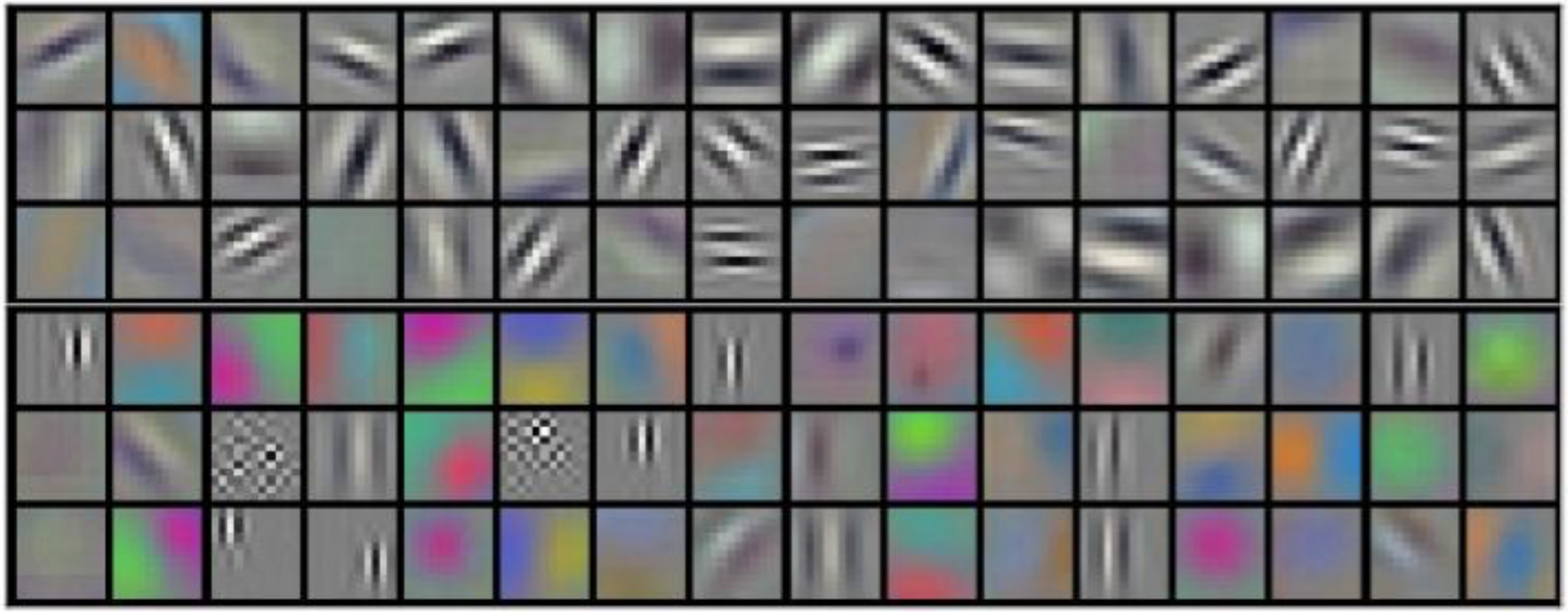
Dropout

- Essential: Randomly drop connections to prevent the network to overfit to a specific configuration of connections
- Example:
  – Independently set each hidden unit activity to zero with 0.5
- probability
  – Done in the two fully-connected hidden layers



$$y_i = \begin{cases} \phi\left(\sum_k w_k x_k\right) \\ 0 \text{ with probability } 0.5 \end{cases}$$

96 1st Layer Convolutional Filters



Retrieval Example