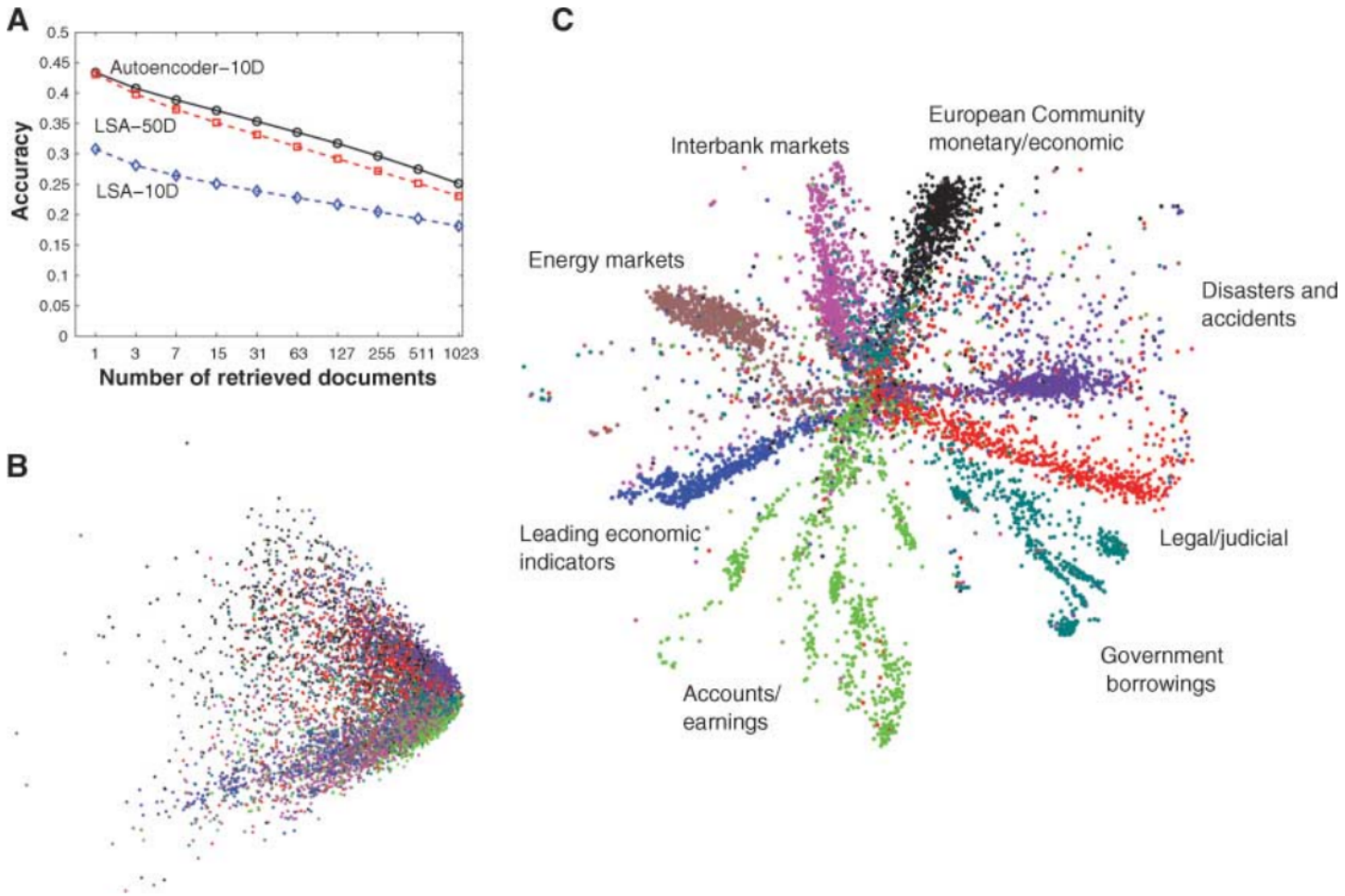
DEEP LEARNING

- Deep Learning has generated much excitement in the scientific community and in industry in the last few years thanks to many breakthrough results in
  - Speech recognition
  - Machine translation
  - Computer vision
    - Object recognition in images
    - Video understanding
  - Natural language understanding
  - Game playing

•Slides originated by www.deeplearningbook.org/
and by the PhD Course slides by Marco Lippi
http://lia.disi.unibo.it/Staff/MarcoLippi/teaching/ml_phd.html

SUCCESS STORIES OF DL

Document classification on the Reuter corpus

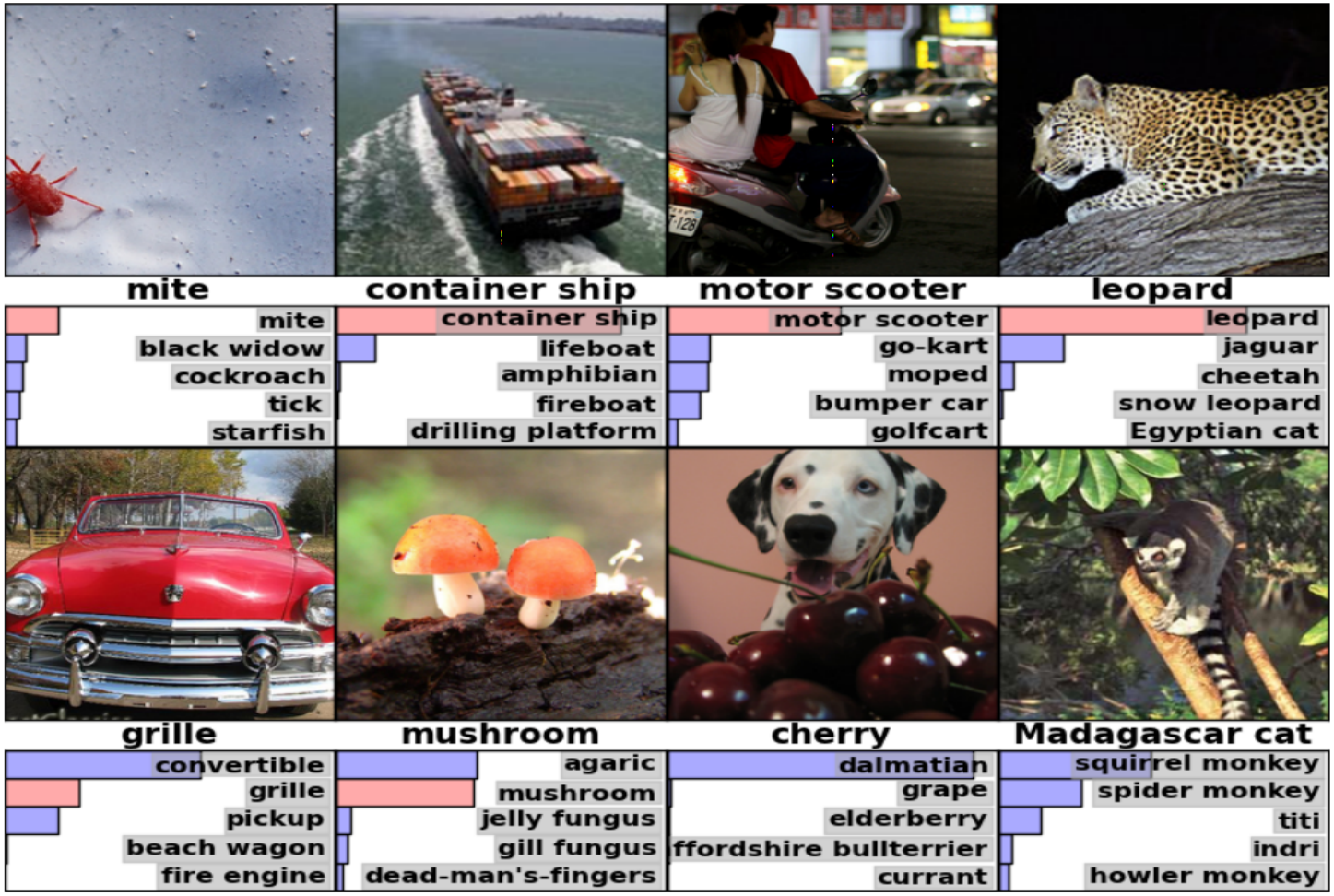


SUCCESS STORIES OF DL

- **ImageNet Challenge**: Universal image classifier
  - 14M images
  - 20k categories
  - Tagged via crowdsourcing
  - Labelled with the WordNet hierarchy

[Figure from vision.stanford.edu]

SUCCESS STORIES OF DL



[Figure from Krizhevsky et al., 2012]

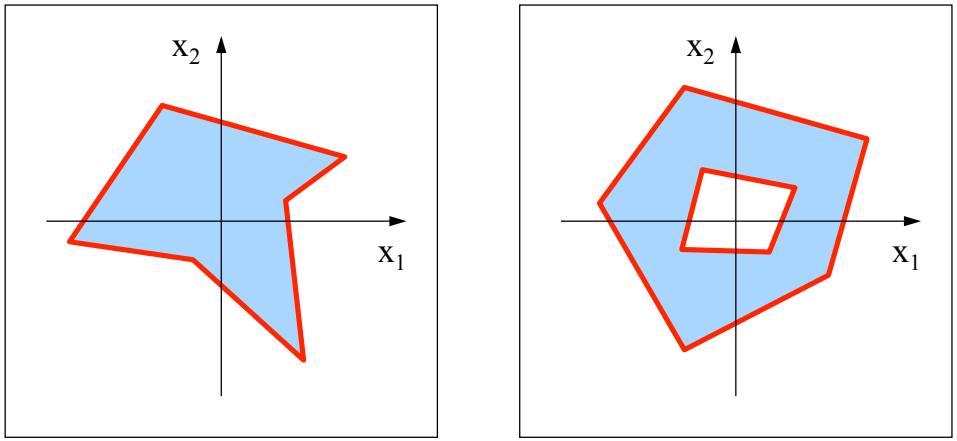
SUCCESS STORIES OF DL

DEEP LEARNING

- What has changed with respect to traditional neural networks.
- Let us revisit a slide: is it really true?

Four-layer networks

- The addition of other layers does not improve the classification ability.



DEEP LEARNING

- With traditional NN it is true.
- **Universal approximation properties and depth**

  Feed forward network with at least one hidden layer provide a universal approximation framework.

  – Regardless of what function we are trying to learn, we know that a large multi-layer NN is able to ***represent*** it

  – We are not guaranteed that the training algorithm will be able to ***learn*** that function for two reasons:

    - Not able to find appropriate values for weights
    - Overfitting

DEEP LEARNING

- Universal approximation theorem states that there exist a network **large enough** to achieve any degree of accuracy we desire, but the theorem does not say how large this network will be.
- **In the worst case, an exponential number of hidden units may be required.**
  - The hidden layer might be infeasibly large and may fail to learn to generalize correctly.

- So adding more layers might be useful under some circumstances that we detail in the following.

DEEP LEARNING

- So adding more layers might be useful but we have to change few things:
  - Activation function (from sigmoid to relu)
  - Semi-supervised learning
    - Unsupervised pre-training
    - Backpropagation for fine tuning
  - Exploit the structure of data (ex. Locality in CNN)
    - Many specialized networks have fewer connections reducing the number of parameters to learn: problem dependent.

DEEP LEARNING

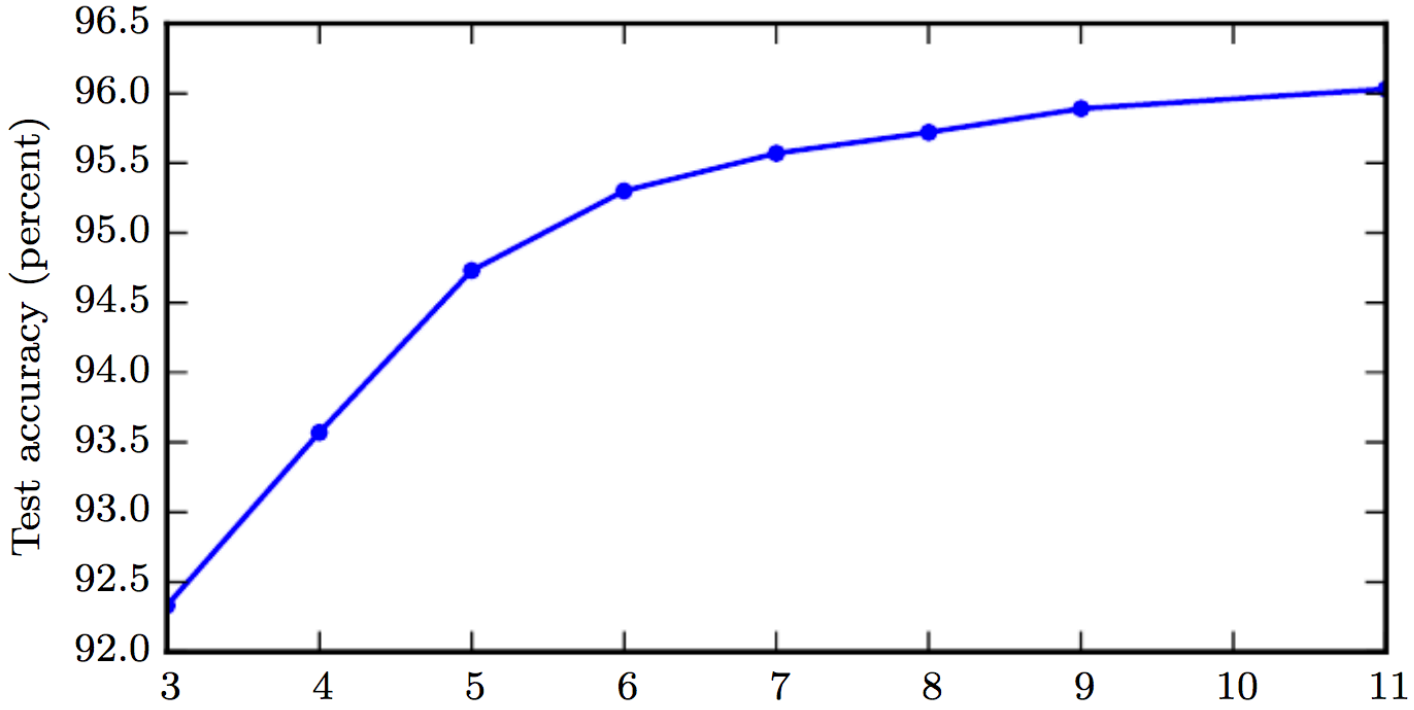
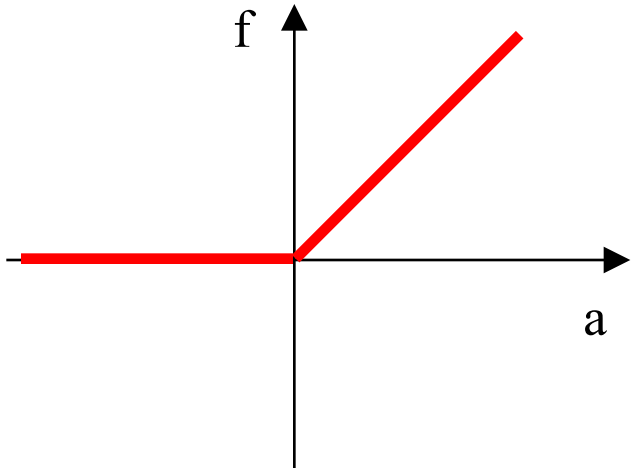


Figure 6.6: Empirical results showing that deeper networks generalize better when used to transcribe multi-digit numbers from photographs of addresses. Data from Goodfellow *et al.* (2014d). The test set accuracy consistently increases with increasing depth. See figure 6.7 for a control experiment demonstrating that other increases to the model size do not yield the same effect.

ACTIVATION FUNCTION

- Rectified linear activation function: f(a) = max(0,a)



- Recommended with most feed forward NN
- The function remains very close to linear thus preserving many properties of linear models.

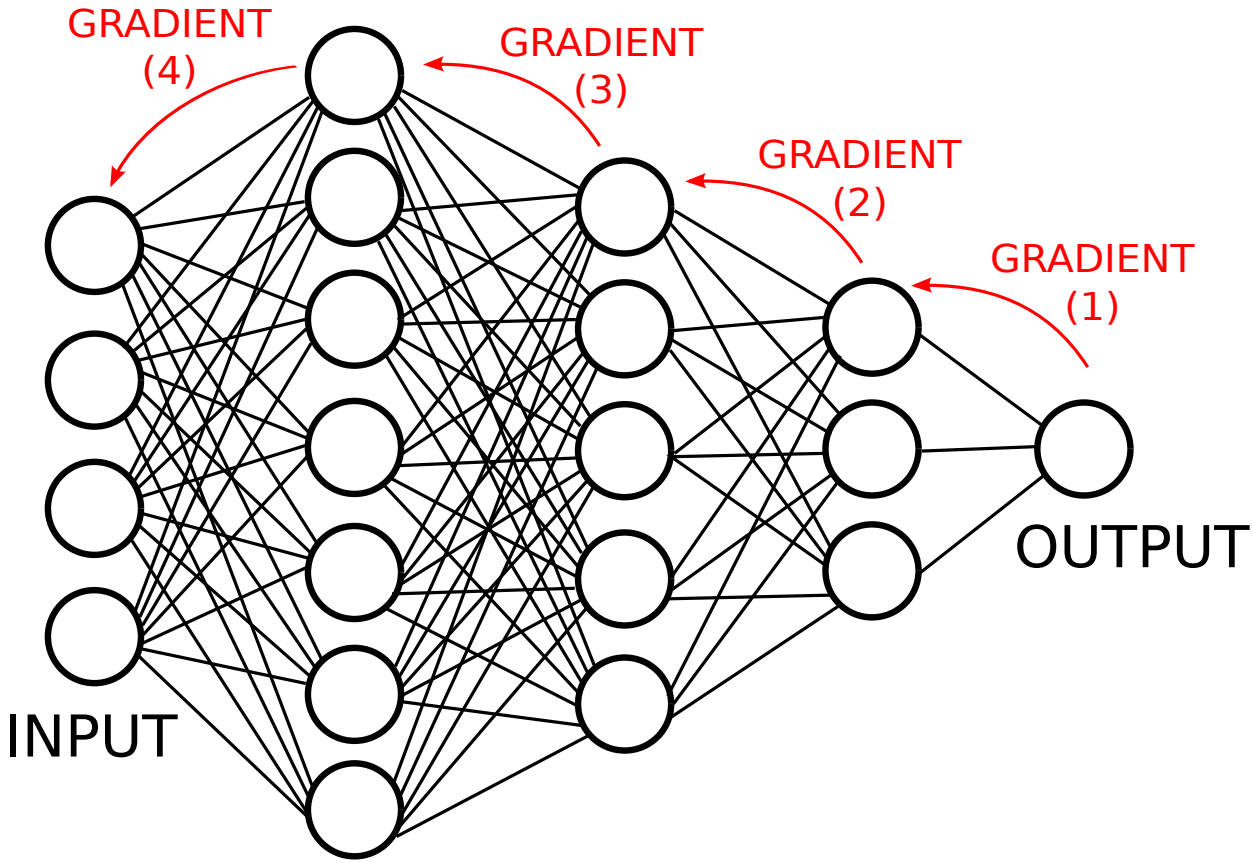
SEMI-SUPERVISED LEARNING

- Traditional NN are trained using supervised back propagation.
- They rely only on labeled data:
  - Labeled data are difficult to find
  - Unlabelled data are everywhere (and are cheap)

- Back-propagation has the problem of vanishing gradients
  - Difficult to use when training a NN from scratch
  - Good for fine tuning

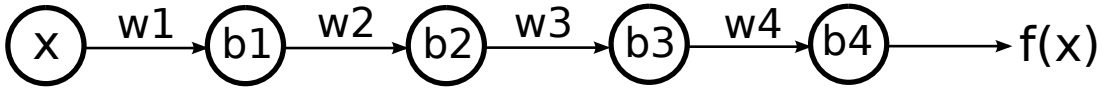
BACK-PROPAGATION PITTFALLS

- Vanishing gradient



BACK-PROPAGATION PITTFALLS

- Consider this example taken from Nielsen: four layer NN

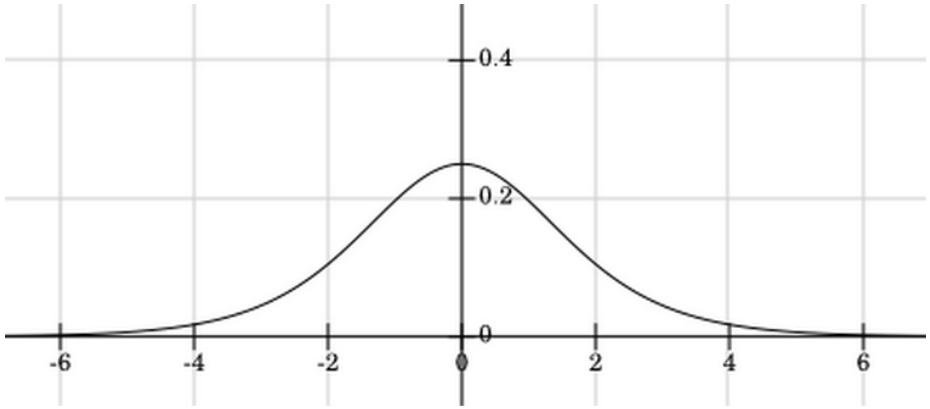


$$z_j = \sigma(w_j\, z_{j-1} + b_j)$$

- If we derive zj with respect to b1, we have to multiply the derivative of the sigmoid for the weight of the same level for all levels

BACK-PROPAGATION PITTFALLS

- Derivative of the sigmoid



- That means that

$$|w_j * \sigma'(z_j)| < ¼$$

- Lower layers tend to have either vanishing or exploding gradients

SEMI-SUPERVISED LEARNING

- Breakthrough in 2006 with Deep Belief Networks (DBN) developed at the University of Toronto (Hinton et. al. 2006)
  - Train one layer at a time
  - Use non-supervised learning
  - Back-propagation for fine tuning of the network

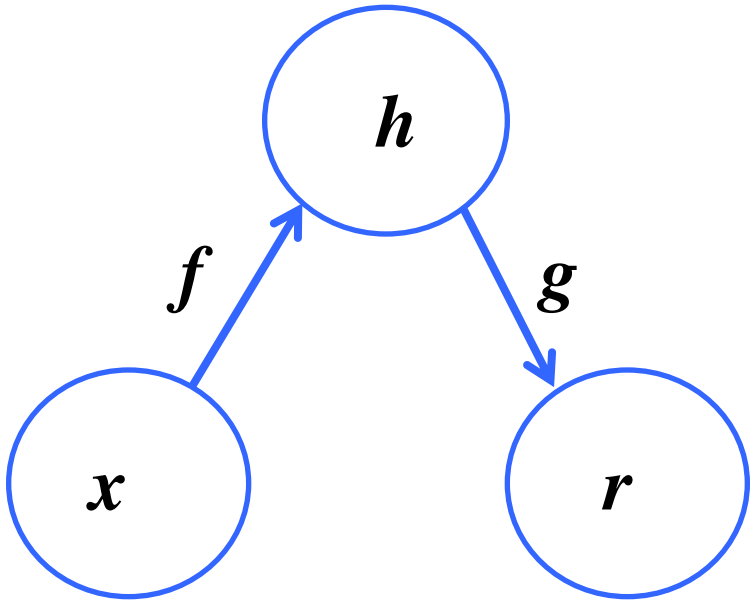
SEMI-SUPERVISED LEARNING

- Supervised learning
  - Data consist in observations X with a label Y
  - Given an observation x in X find a prediction y in Y
  - Learn a function f: X ➔Y from data

- Unsupervised learning
  - Data consist in observations X
  - Find patterns and regularities in data
  - **Find most important features representing data**

AUTOENCODERS

- Autoencoders are NN that learn a representation of the input and is able to reconstruct it



$h=f(x)$

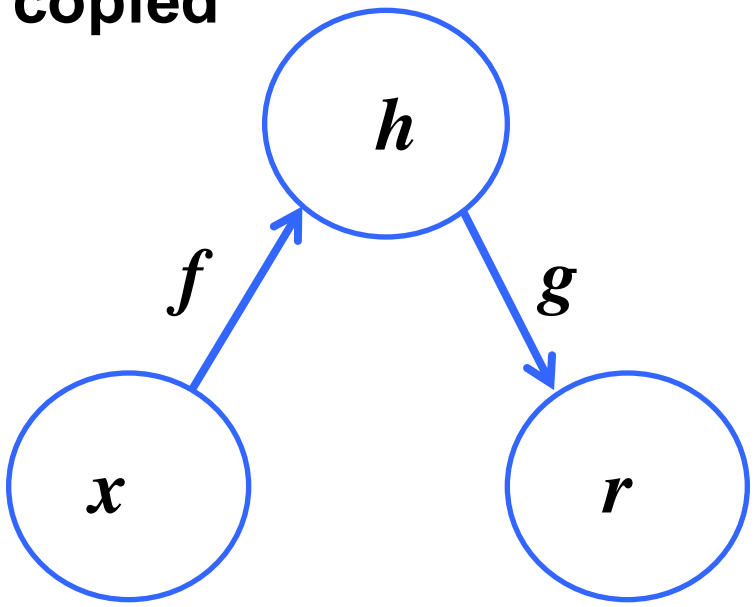
$r=g(h)$

Unsupervised learning

Minimize reconstruction error

AUTOENCODERS

- We do not want to have a perfect copy, but an **approximation that prioritizes which aspects should be copied**



$h=f(x)$

$r=g(h)$

f: encoder

g:decoder

Minimize a loss function: $L(x, g(f(x)))$

It penalizes g(f(x)) to be dissimilar to x

ex. MSE

UNDERCOMPLETE AUTOENCODERS

- h has a smaller dimension than x
  - Feature selection
  - Dimensionality reduction similar to PCA
  - Capture the most salient features of the training data
  - If the decoder is linear and L is the mean square error: Principal Component Analysis
  - Non linear f and g learn more powerful generalizations of PCA

OVERCOMPLETE AUTOENCODERS

- h has a larger dimension than x
  - Induce parameter **sparsity**
  - Discover non-linear relations that are hardly discovered via statistical methods

REGULARIZED AUTOENCODERS

- Regularization prevents the autoencoder to learn useless mapping, e.g. the identity function

- Regularized autoencoders use a loss function that encourages the model to have other properties besides the ability to copy its input to its output:
  - sparsity of the representation
  - small derivatives of the representation
  - robustness to noise and missing input

SPARSE AUTOENCODERS

- A sparse autoencoder is simply an autoencoder whose training criterion involves a sparsity penalty Ω(h) on the code layer h, in addition to the reconstruction error:

$$L(x, g(f(x))) + \Omega(h)$$

- Sparse autoencoders are typically used to learn features for another task such as classification

$$\Omega(h) = \lambda \sum_i |h_i|$$

PENALIZING DERIVATIVES

- This forces the model to learn a function that does not change much when x changes slightly:

$$L(x, g(f(x))) + \Omega(h, x)$$

- Because this penalty is applied only at training examples, it forces the autoencoder to learn features that capture information about the training distribution.

$$\Omega(h, x) = \lambda \sum_i \left\| \nabla_x h_i \right\|$$

CONTRACTIVE AUTOENCODER

DENOISING AUTOENCODERS

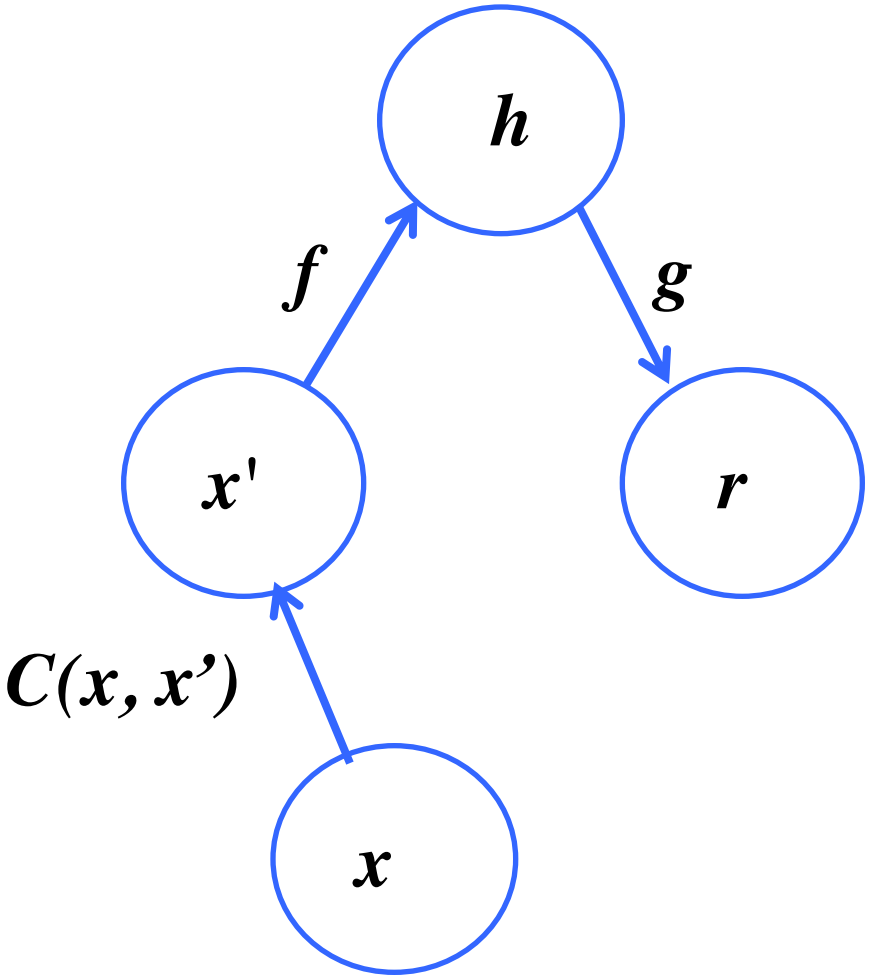
- Rather than adding a penalty a denoising autoencoder has a different loss function computed on a corrupted input vector ***x'***:

$$L(x, g(f(x')))$$

- The denoising autoencoder (DAE) receives a corrupted data point as input and is trained to predict the original, uncorrupted data point as its output

DENOISING AUTOENCODERS

- $C(x, x')$ is a corruption process representing the conditional distribution over corrupted samples x' given x



DENOISING AUTOENCODERS

- The autoencoder then learns a reconstruction distribution $p_{reconstruct}(x \mid x')$ estimated from training pairs (x,x'), as follows:

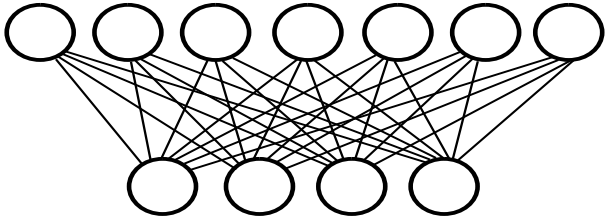
  - Sample a training example x from the training data
  - Sample a corrupted version x' from C(x | x')
  - Use (x,x') as a training example for estimating the autoencoder reconstruction distribution

    $p_{reconstruct}(x \mid x') = p_{decoder}(x \mid h)$ with $h=f(x')$

LAYER SIZE AND DEPTH

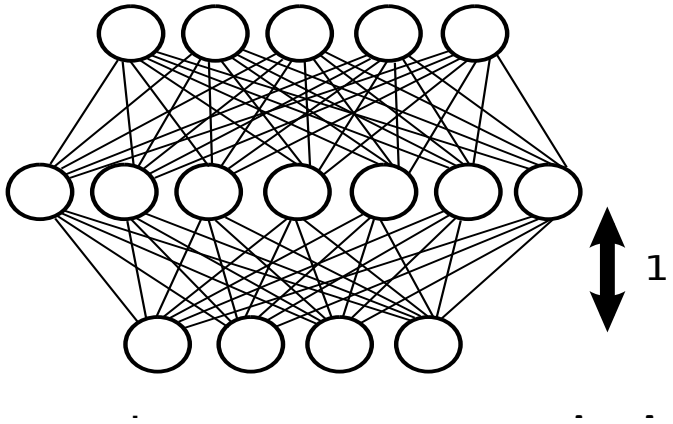
- Autoencoders are often trained with a single layer encoder/decoder
- Deep encoders have advantages:
  - depth can exponentially reduce the computational cost of representing some functions
  - depth can exponentially reduce the amount of training data needed
  - deep autoencoders yield a much better compression

DEEP NETS GENERAL SCHEME

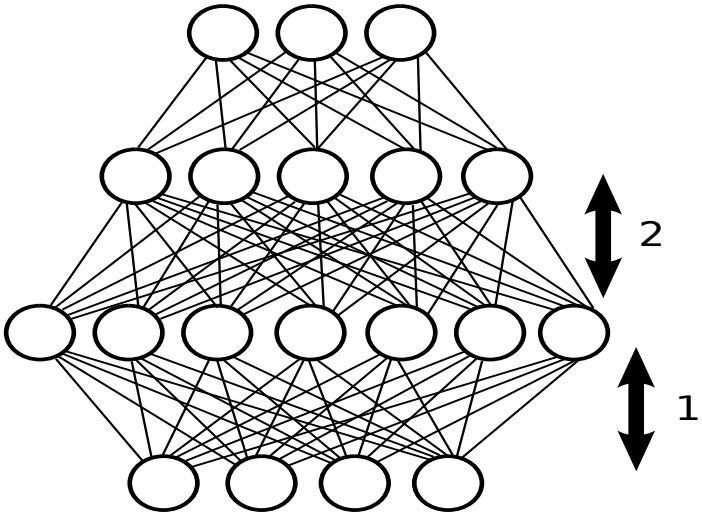


- Training a level at a time: **unsupervised learning**
- **Each level can be an autoencoder**

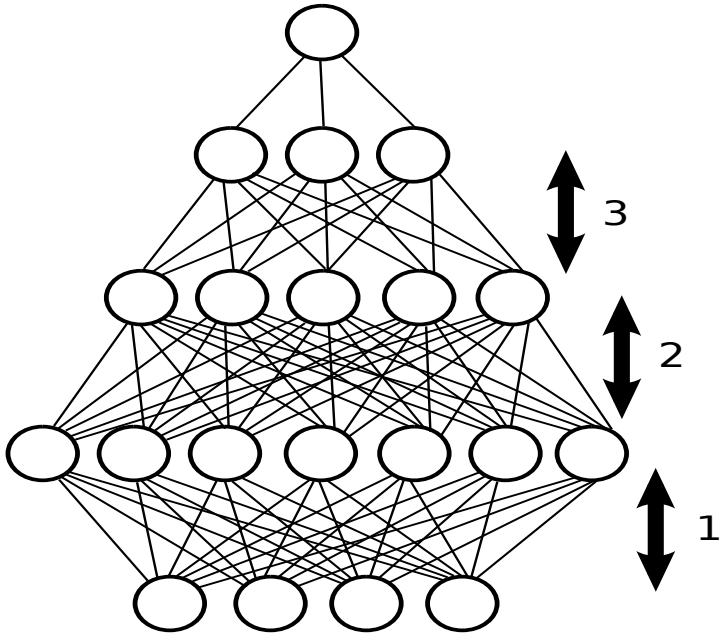
DEEP NETS GENERAL SCHEME



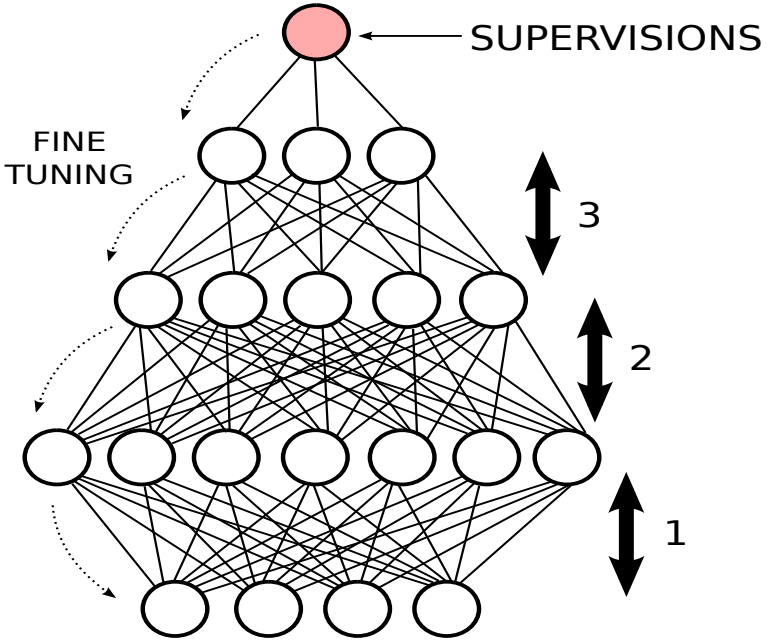
DEEP NETS GENERAL SCHEME



DEEP NETS GENERAL SCHEME



DEEP NETS GENERAL SCHEME



•Fine tuning with supervised learning and back propagation

LEARNING FEATURES

- Deep learning is essentially **representation learning**

  • Most machine learning approaches work well for specific tasks owing to **a great effort** in the design of **good features**

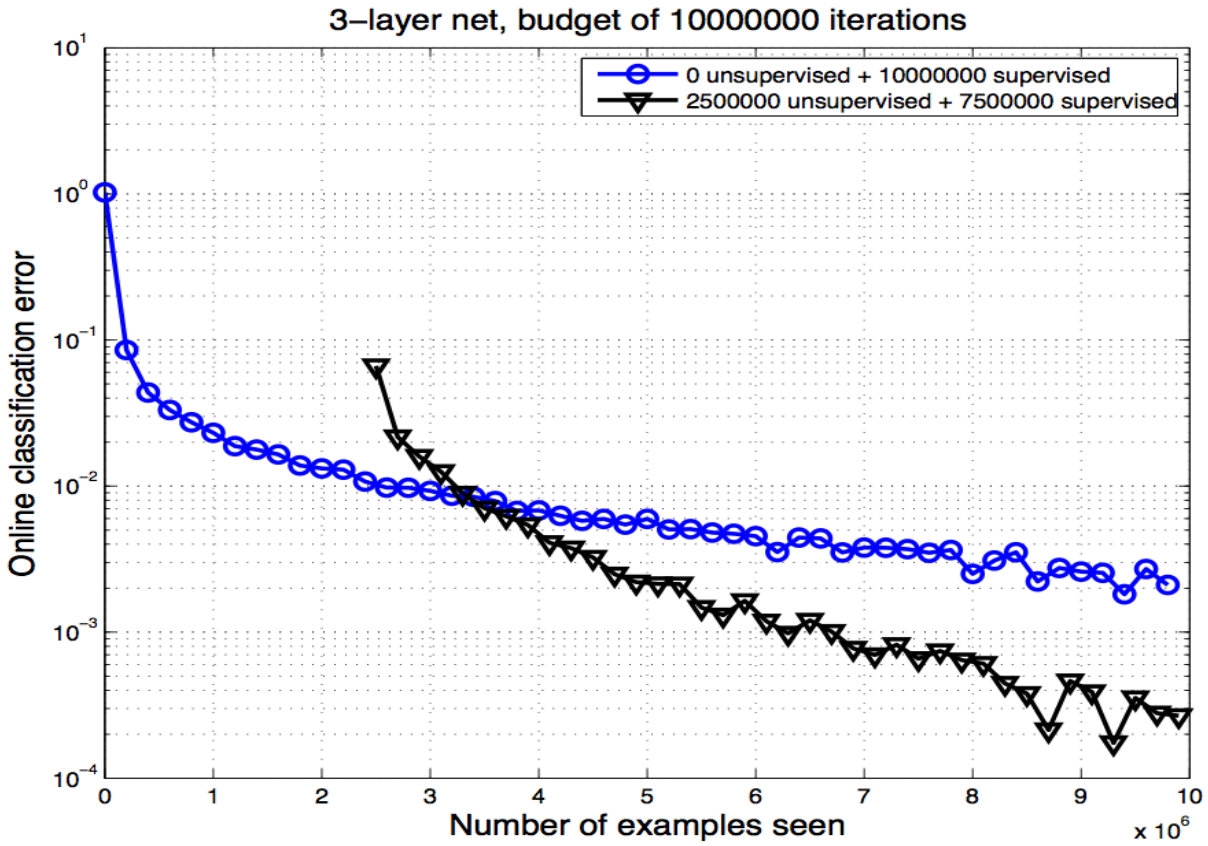
  • Finding appropriate features can be **a time-consuming** but also **subtle task** even for specialists

  • Representation learning tries to **automatically learn good features** from raw input data (even not knowing the task?)

  • Deep learning tries to learn a **hierarchy of multiple levels** of representation having **increasing complexity**

LEARNING FEATURES

•For the same training error (at different points during training), the test error is systematically lower with unsupervised pre-training. [Erhan et al., 2009]



LEARNING FEATURES

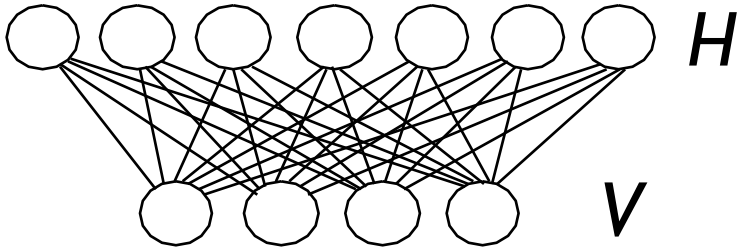
- As discussed in [Erhan et al., 2009], unsupervised pre-training can be seen as a form of regularizer (or prior)…

- Unsupervised pre-training amounts to **a constraint on the region in the parameter space where a solution is allowed**.

- Experiments show that the effect of unsupervised pre-training is most marked for the lower layers of a deep architecture.

DEEP BELIEF NETWORKS

- Same schema seen before:
  - Each level is a Restricted Boltzman Machine



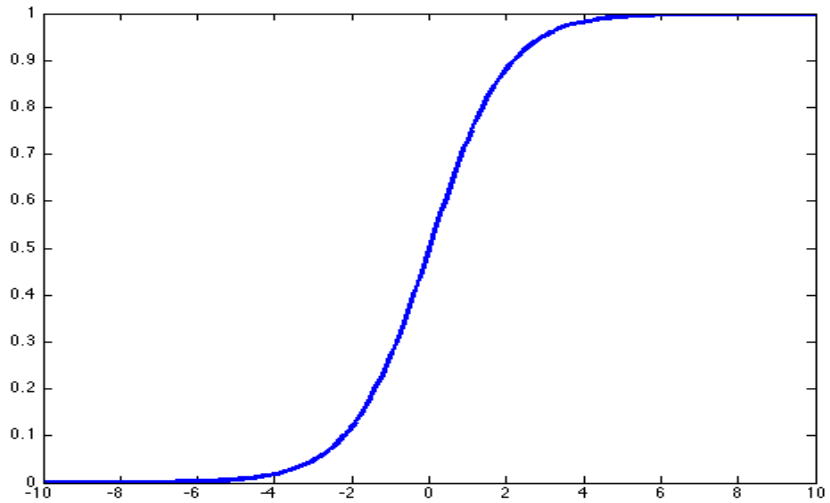
- Modeling two sets of random variables V and H:
  - $V = V_1, \ldots, V_m$ (**visible**)
  - $H = H_1, \ldots, H_n$ (**hidden**)

  all H_i are independent when conditioning on V

  all V_j are independent when conditioning on H

DEEP BELIEF NETWORKS

- Same schema seen before:
  - Each level is a Restricted Boltzman Machine
  - Induces a probability factorization.



$$Prob(neuron = active) = \frac{1}{1 + e^{-input}}$$

Restricted Boltzmann Machines (RBMs)

•Since there are no connections between units of the same layer, the structure of the RBM induces a probability factorization, so that:

$$P(H = h|V = v) = \prod_i P(h_i|v)$$

$$P(V = v|H = h) = \prod_i P(v_i|h)$$

•The probability distribution over such units is computed as:

$$P(h_j = 1|v) = \sigma(c_j + \sum_i v_i W_{ij})$$

$$P(v_i = 1|h) = \sigma(b_j + \sum_i h_j W_{ij})$$

Restricted Boltzmann Machines (RBMs)

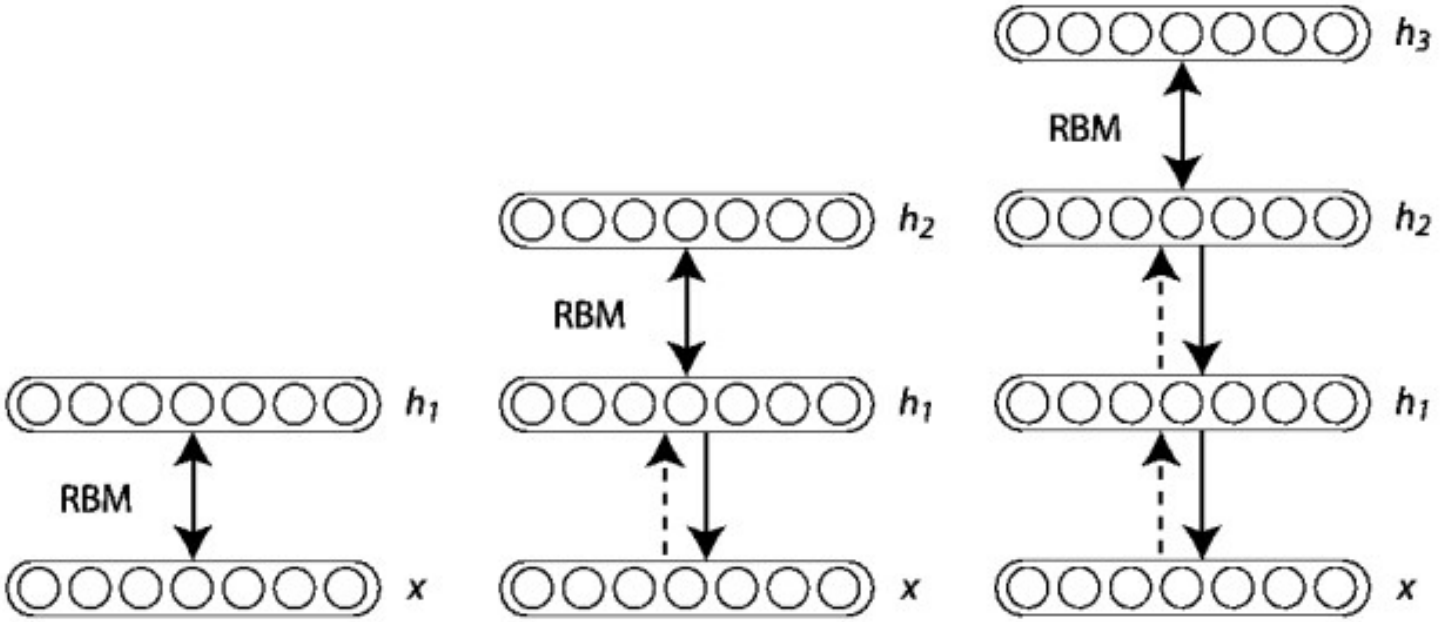
- Classic learning method for RBMs: **Maximum Likelihood**
- The likelihood of the visible data *v* can be written as a **sum over all possible configurations of hidden states**

$$L = P_{model}(v) = \sum_{h} p(v, h) = 1/Z \exp\left(\sum_{ij} v_i w_{ij} h_j \right)$$

- Maximize likelihood **of given data** w.r.t. weights w_{ij}

Deep Belief Networks [Hinton et al., 2006]

- A DBN consists in a stack of RBMs



- Layer-wise pre-training of all layers
- Fine tuning with backpropagation

Deep Belief Networks [Hinton et al., 2006]

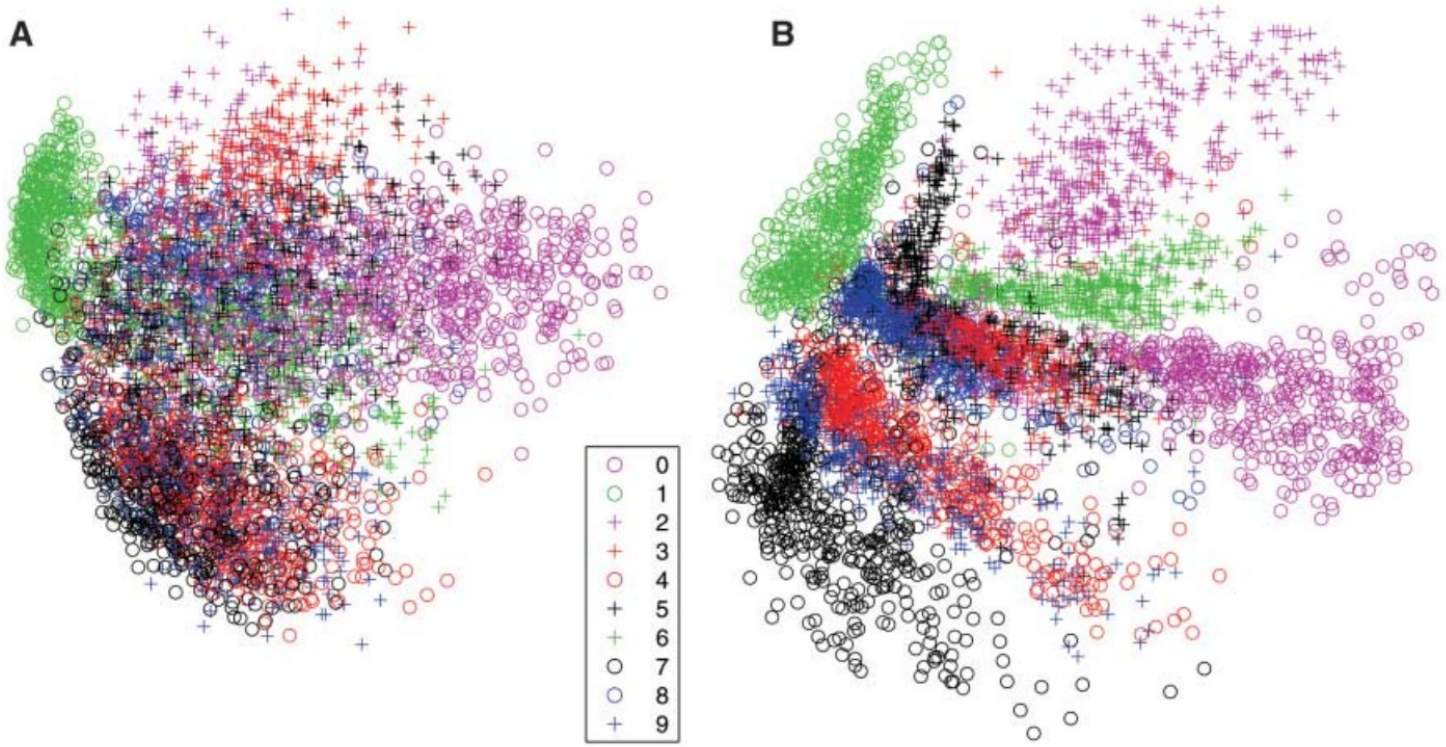
- Experiments performed on:
  - Image classification (Digits, Faces)
  - Document classification (Reuters corpus)

- In both cases, a DBN was trained with 4 layers, with the final feature encoder consisting in only 2 dimensions:

  - 784 – 1000 – 500 – 250 – 2 (images)
  - 2000 – 500 – 250 – 125 – 2 (text)

Deep Belief Networks [Hinton et al., 2006]

- Image classification on the MNIST digit data set

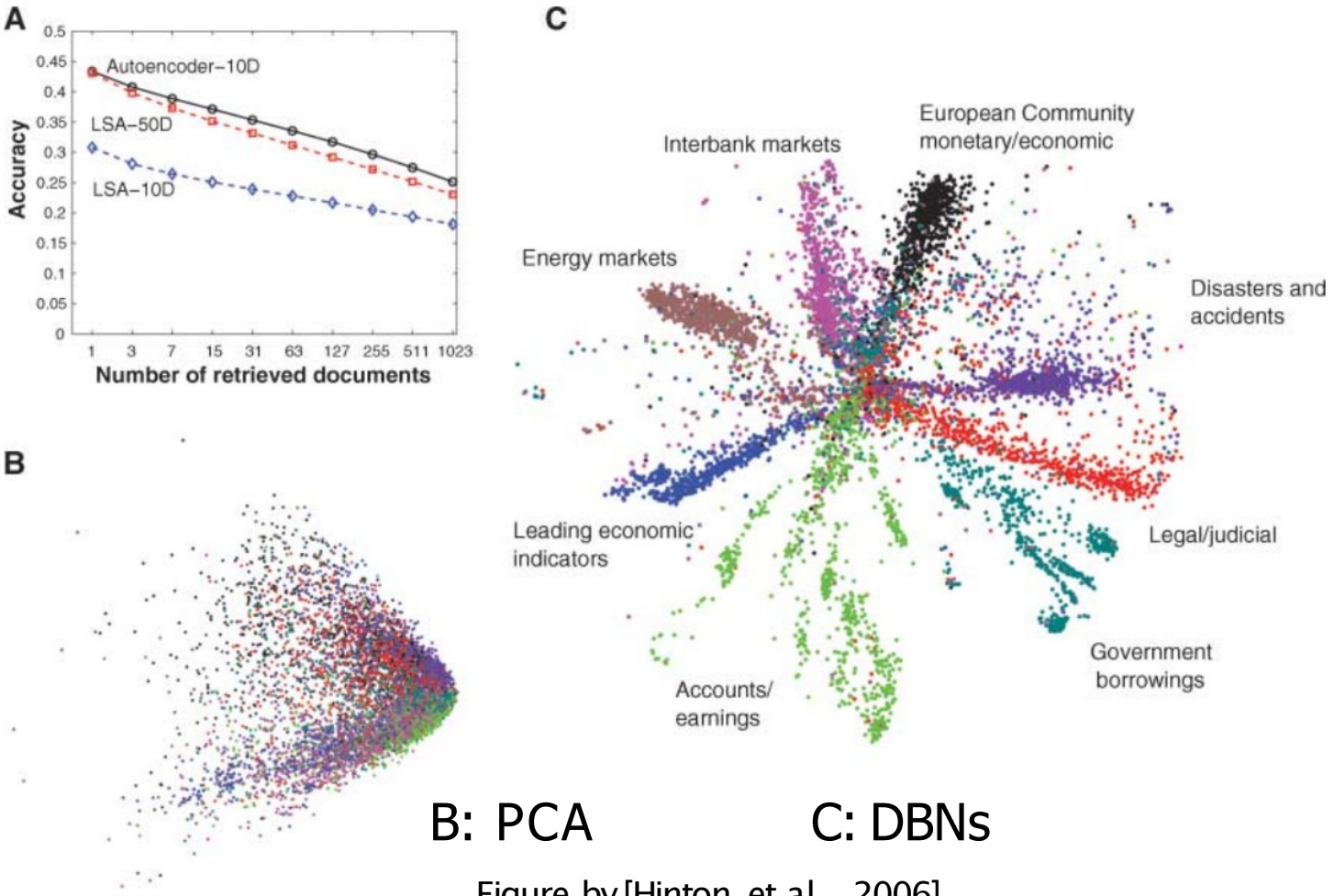


A: PCA B: DBNs

- Figure by [Hinton et al., 2006]]

Deep Belief Networks [Hinton et al., 2006]

- Document classification on the Reuter corpus



B: PCA C: DBNs

Figure by [Hinton et al., 2006]

Deep Belief Networks [Hinton et al., 2006]

- Digit reconstruction

- Figure by [Hinton et al., 2006]]

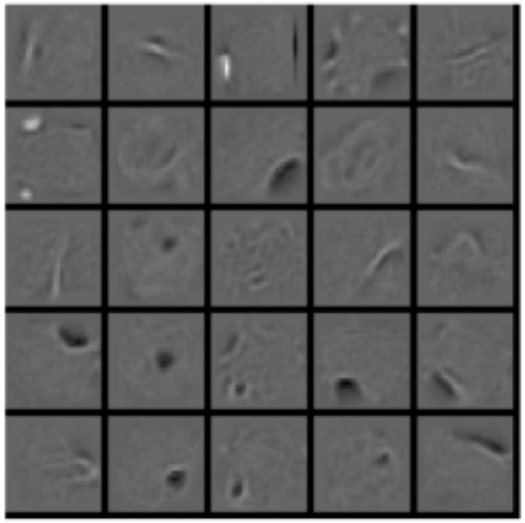
1: original images, 2: DBN (top layer 30), 3-4: PCA

Deep Belief Networks [Hinton et al., 2006]

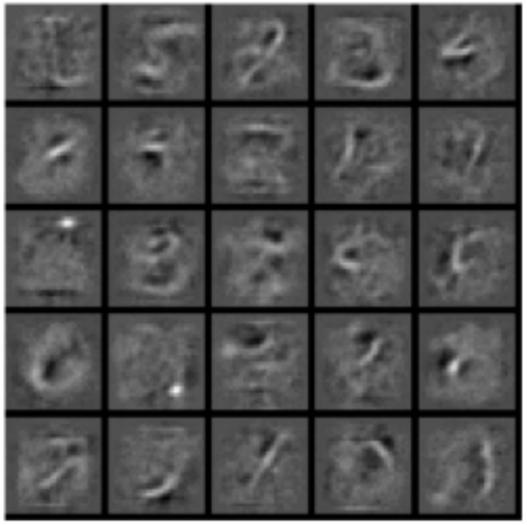
- Faces reconstruction

Deep Belief Networks [Hinton et al., 2006]

1^{st}-layer features



2^{nd}-layer features

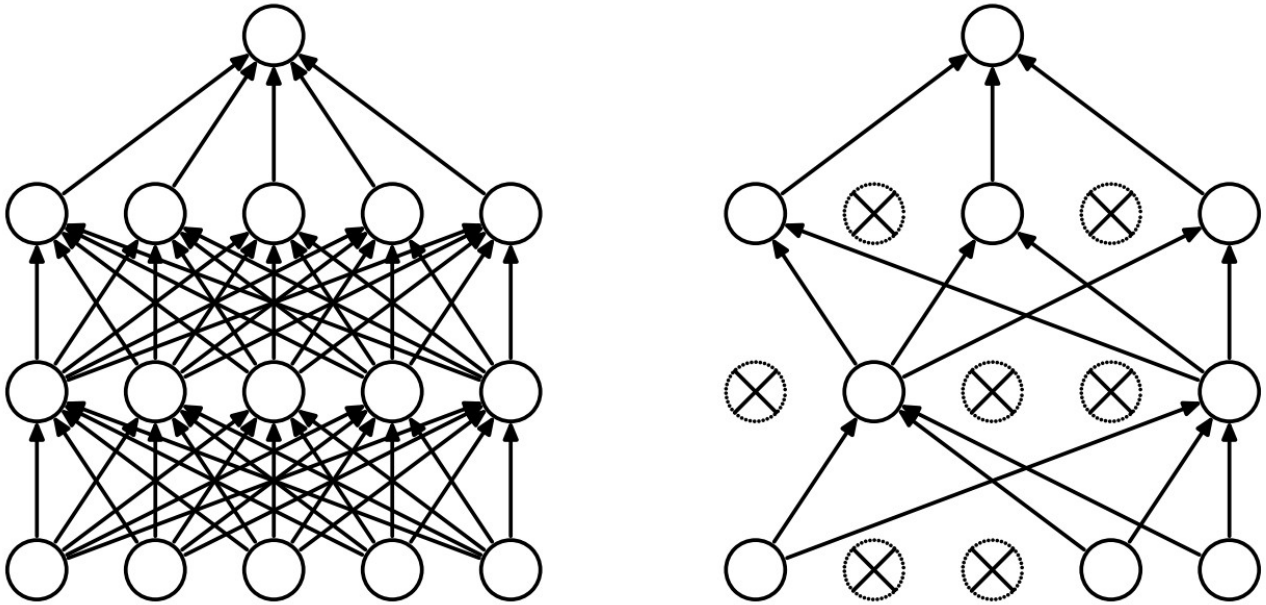


DROPOUT

- The mechanism transforms a deep network in a **thinned** one.
- A neural network with n units can be seen as the **combination** of 2^n possible thinned neural networks
- Combining classifiers almost **always** improves performance
- Training different architectures with different parameters and/or inputs would be **too expensive**!

DROPOUT

•The key idea of dropout is to **randomly drop units and their connections** during training.

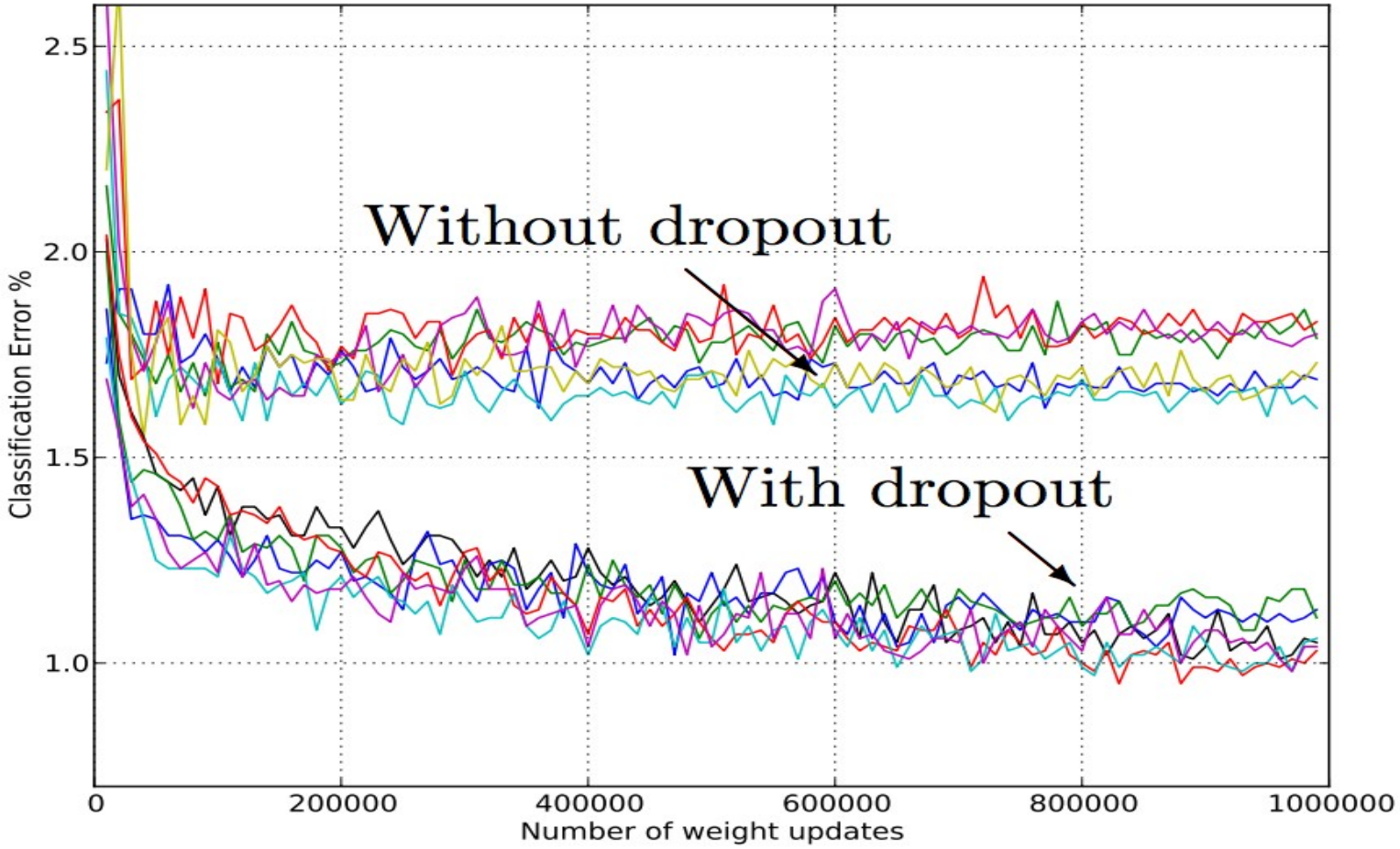


(a) Standard Neural Net

(b) After applying dropout.

[Figure by Srivastava et al., 2014]

DROPOUT



- [Figure by Srivastava et al., 2014]

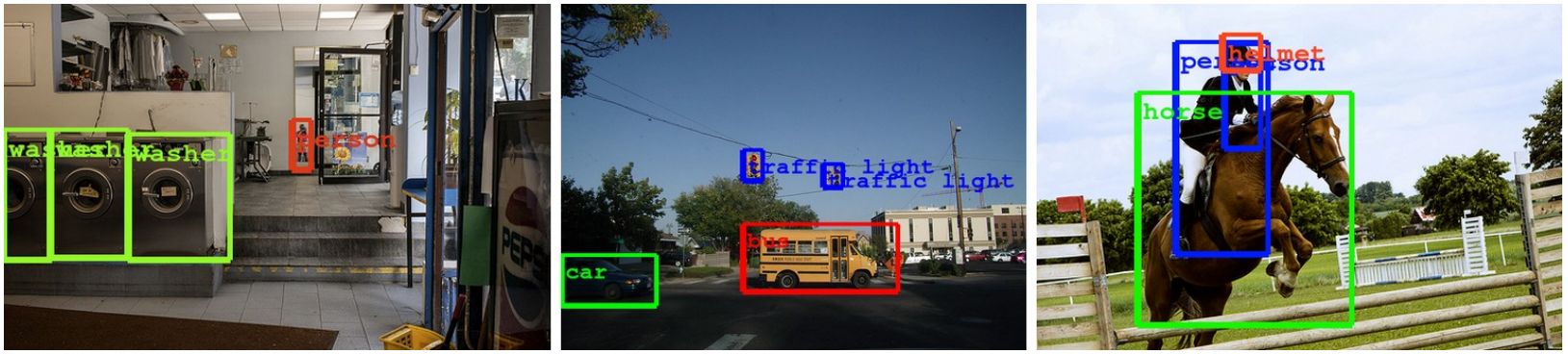
CONVOLUTIONAL NEURAL NETWORKS

- CNN are a specialised kind of NN for processing data that have a known grid-like topology:
  - Time series data (1D grid)
  - Image data (2D grid)

- CNN employ a mathematical operation called **convolution**.
- CNN are NN that use convolution in place of general matrix multiplication in at least one of their layers.

CONVOLUTIONAL NEURAL NETWORKS

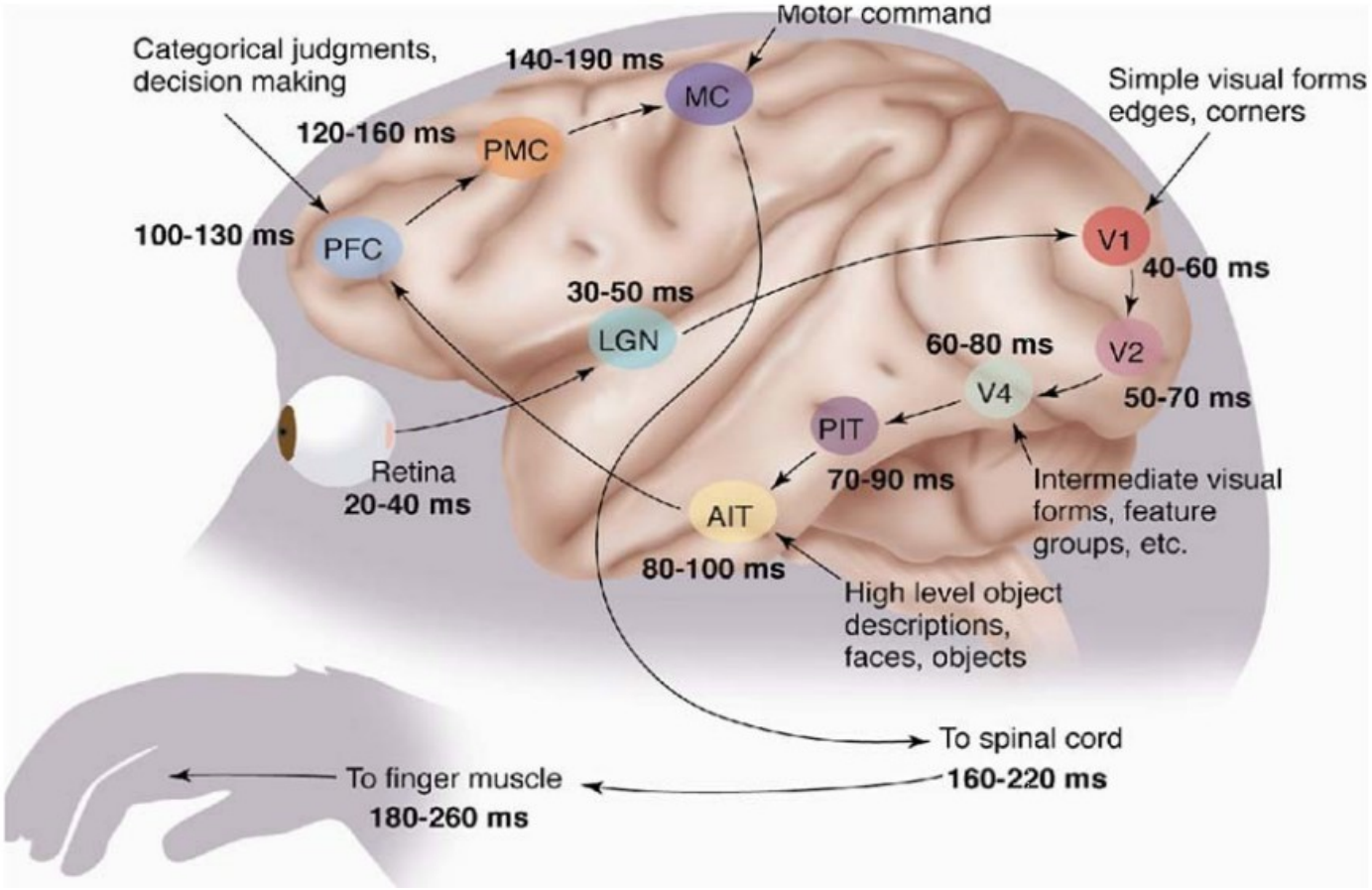
- Architecture inspired by biological processes, focused on **vision**: neurons respond to overlapping regions in a **visual field**
- extremely **fast** computation (especially now with GPUs) pioneering models **back from the 80s-90s** !



[Figure from Google Research]

Quite a long history. . .

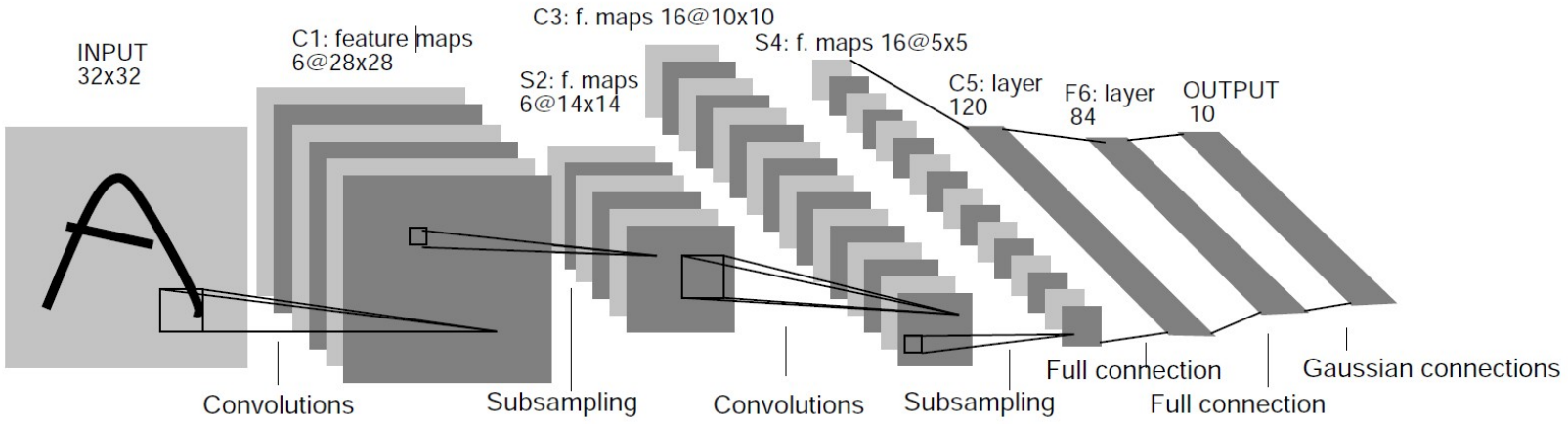
- The human visual cortex **is** hierarchical



- [Figure from nyu.edu, Simon Thorpe]

Quite a long history. . .

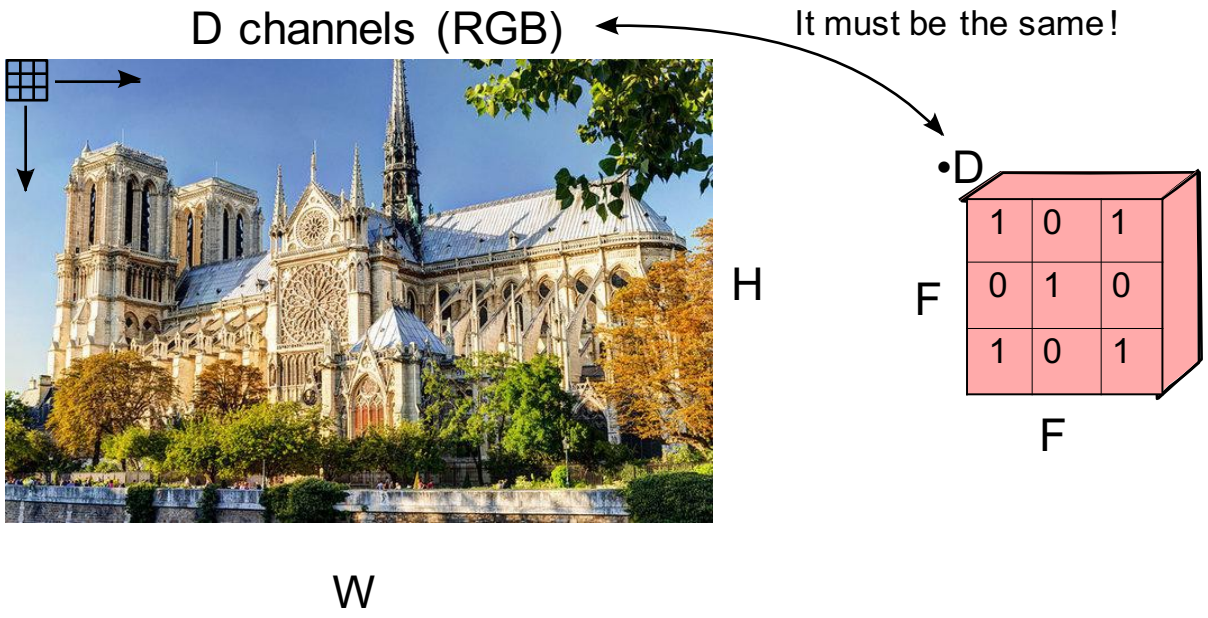
- **LeNet5** (LeCun et al., 1989)



- [Figure from LeCun et al., 1989]

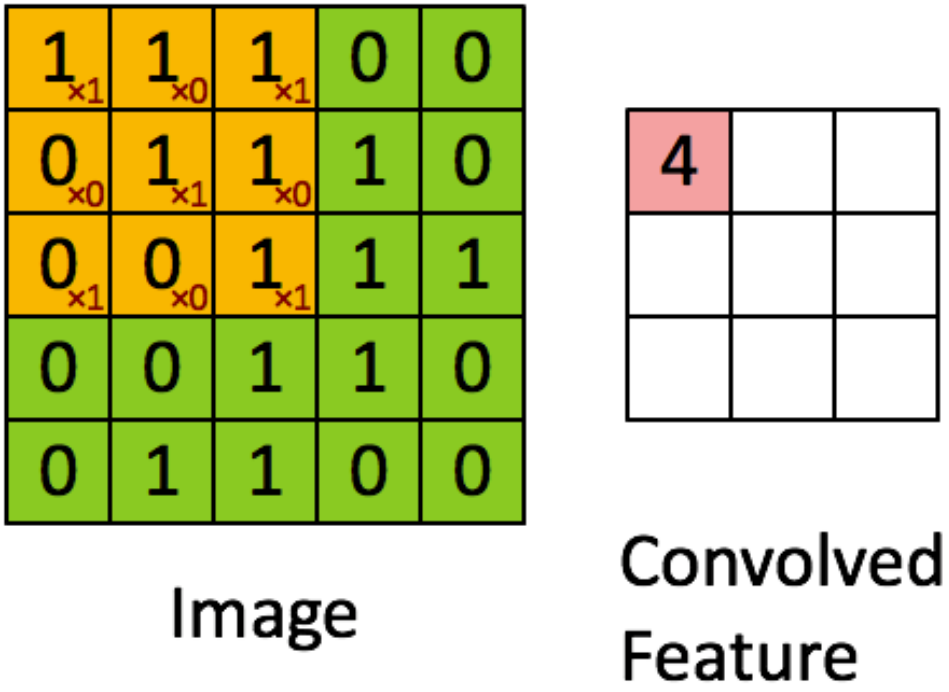
RECEPTIVE FIELDS AND CONV FILTERS

• **Convolutional filter**: a matrix (tensor) of weights to be applied on the image to perform **convolutions**



Receptive fields and convolutional filters

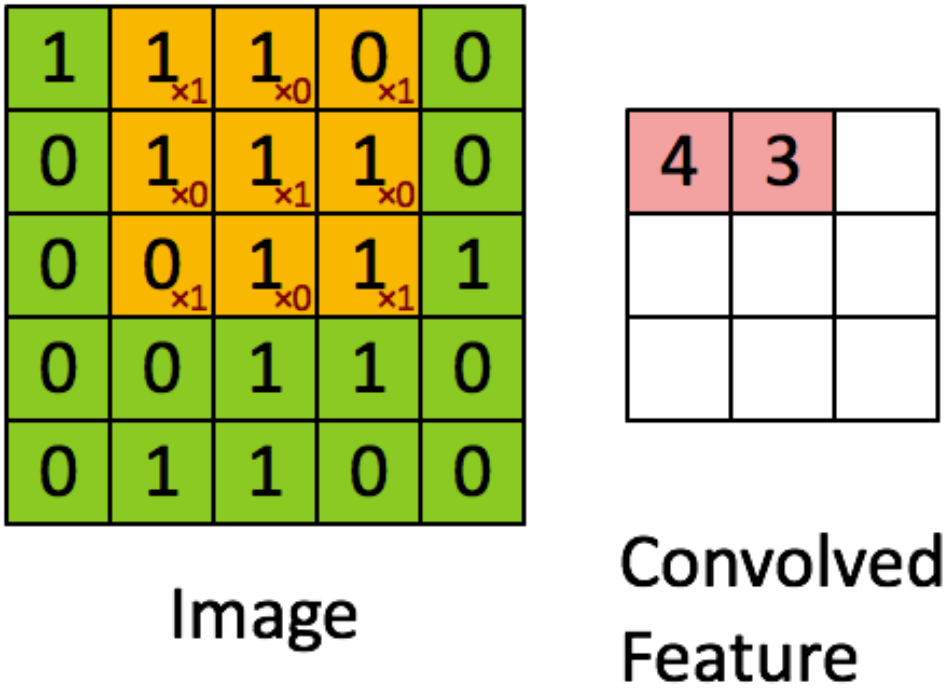
- Convolution between image patch and filter



•[Figure from http://deeplearning.stanford.edu/]

Receptive fields and convolutional filters

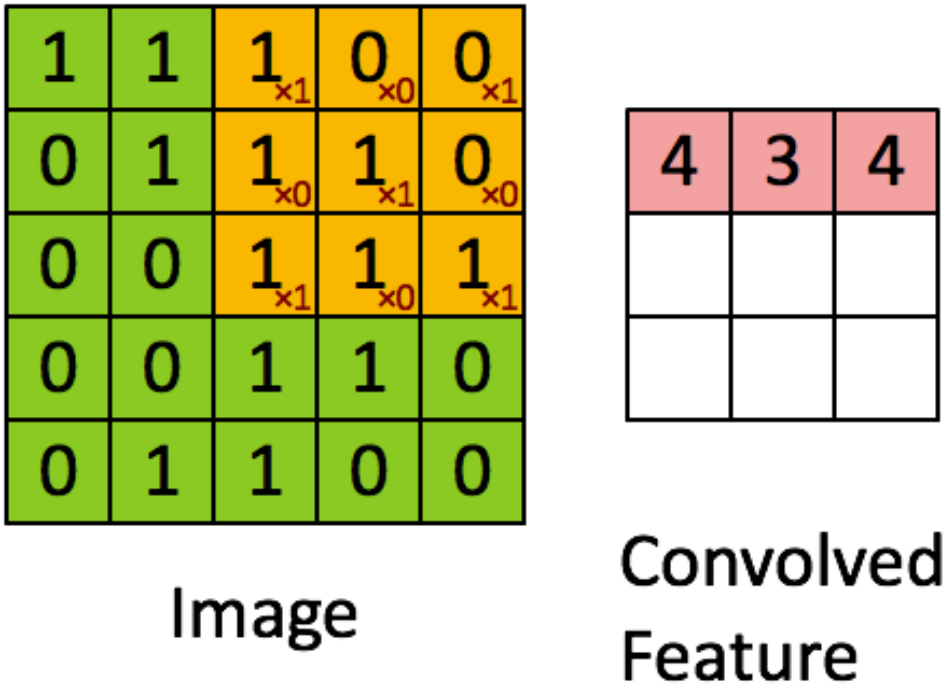
- Convolution between image patch and filter



•[Figure from http://deeplearning.stanford.edu/]

Receptive fields and convolutional filters

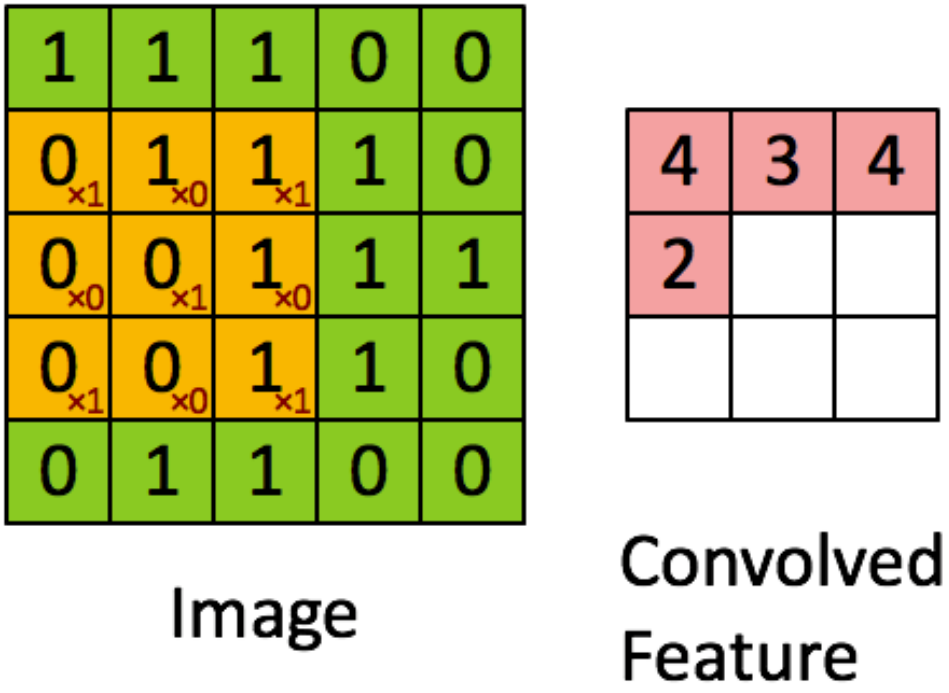
- Convolution between image patch and filter



•[Figure from http://deeplearning.stanford.edu/]

Receptive fields and convolutional filters

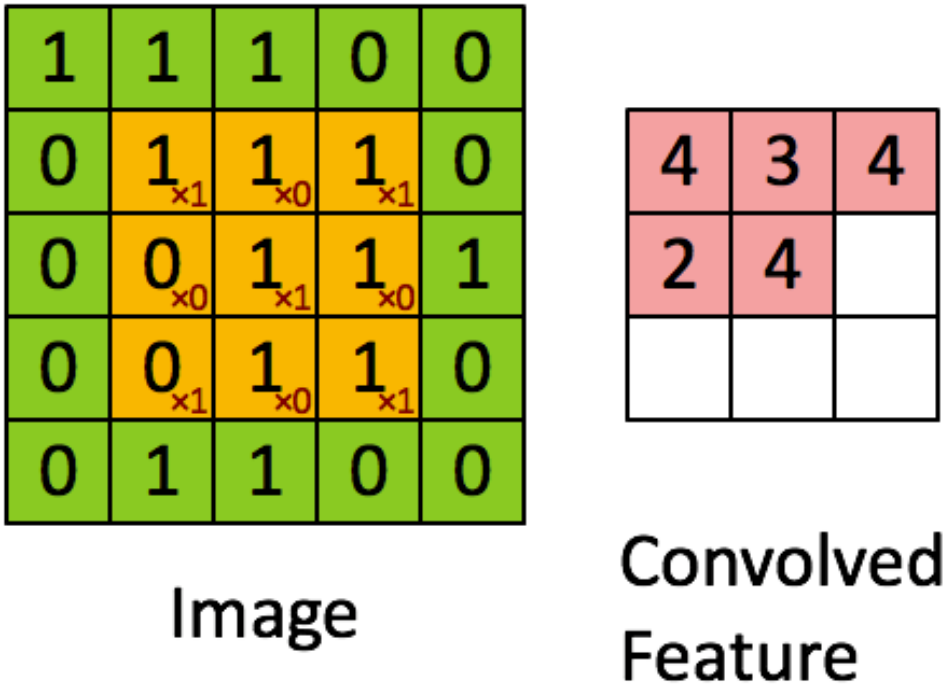
- Convolution between image patch and filter



•[Figure from http://deeplearning.stanford.edu/]

Receptive fields and convolutional filters

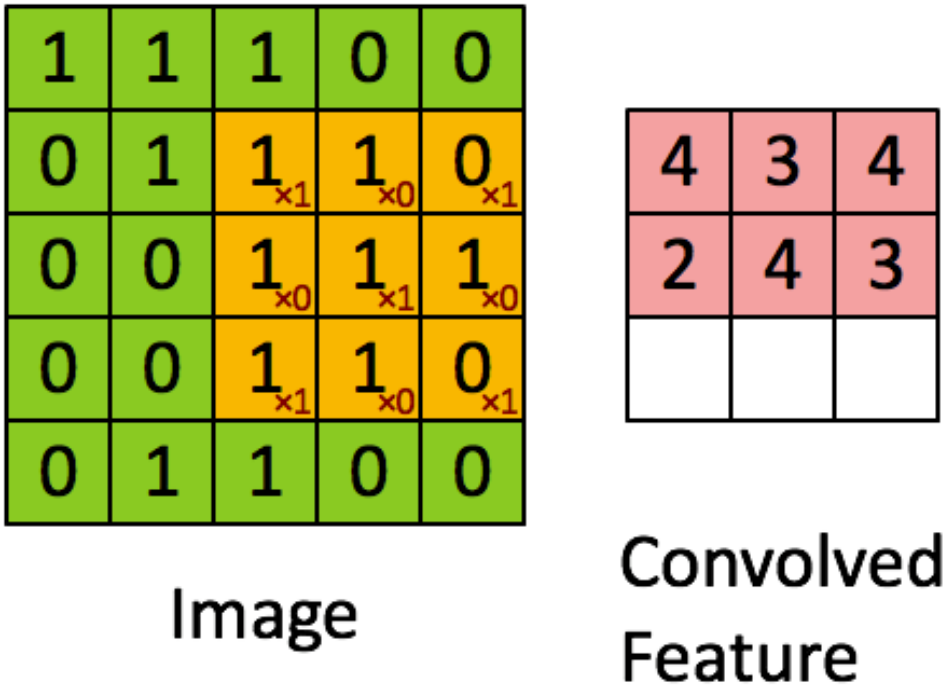
- Convolution between image patch and filter



•[Figure from http://deeplearning.stanford.edu/]

Receptive fields and convolutional filters

- Convolution between image patch and filter

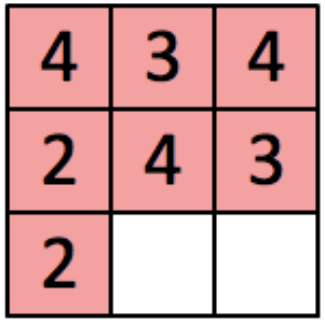


•[Figure from http://deeplearning.stanford.edu/]

Receptive fields and convolutional filters

- Convolution between image patch and filter

Image

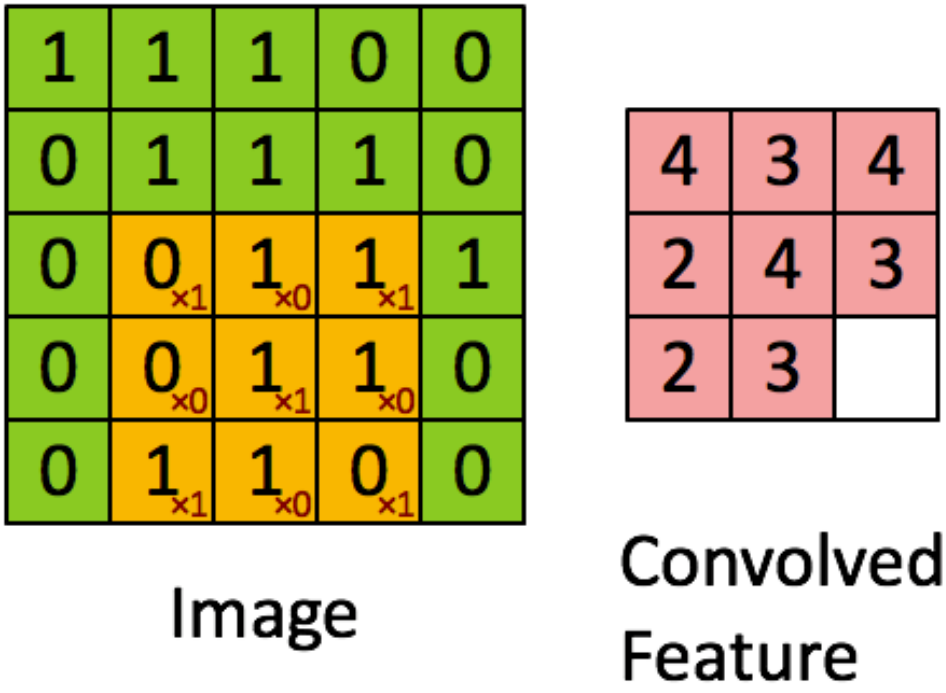


Convolved Feature

- [Figure from http://deeplearning.stanford.edu/]

Receptive fields and convolutional filters

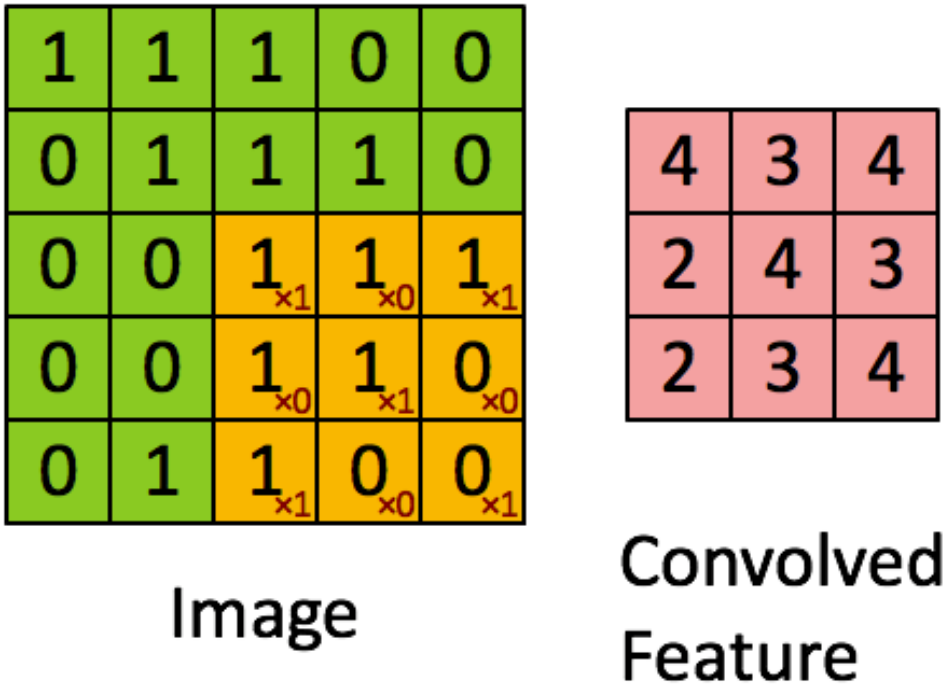
- Convolution between image patch and filter



•[Figure from http://deeplearning.stanford.edu/]

Receptive fields and convolutional filters

- Convolution between image patch and filter



•[Figure from http://deeplearning.stanford.edu/]

Receptive fields and convolutional filters

Stride

- Hyper-parameter S indicating the "step" to be used when moving the filter on the image
  - given a $W \times H$ image
    given a $F \times F$ filter
  - $(W - F)/S$ and $(H - F)/S$ must be integers

Zero padding

- Adding zeros along the border to allow convolutions on all pixels
  - if $S = 1 \rightarrow$ zero padding with $(F - 1)/2$

CONVOLUTIONAL NEURAL NETWORKS

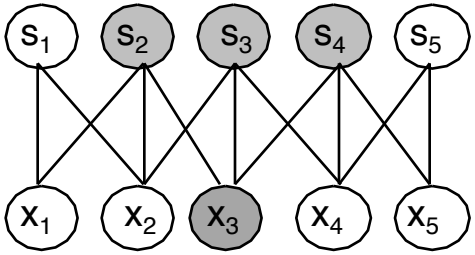
- Convolutional networks leverage three important ideas that improve a machine learning system:
  - sparse interactions
  - parameter sharing
  - equivariant representation

SPARSE INTERACTION

- Traditional networks connect each neuron of a layer with ALL neurons of adjacent layers
    - each output is influenced by all inputs
- In CNN instead there isa significantly lower number of connections
    - Ex. In an image with thousend or million of pixels we might detect meaningful features with kernels that occupy tens or hundreds of pixels
    - Less parameters
    - Reduced memory requirement
    - improves efficiency

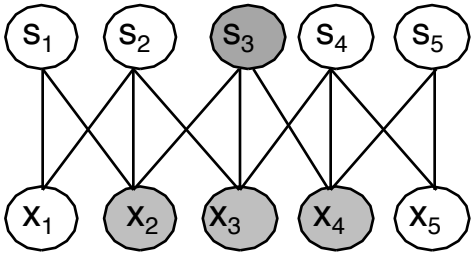
SPARSE CONNECTIVITY

Sparse connectivity view from below

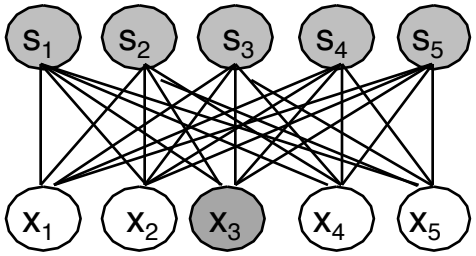


When s is formed by convolutions with a kernel of width 3, only three outputs are affected by x_3

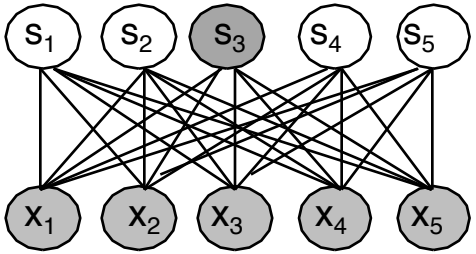
Sparse connectivity view from above



When s is formed by convolutions with a kernel of width 3, only three inputs affect s_3: **receptive field of s_3**



When s is formed by matrix multiplications, all of the outputs are affected by x_3



When s is formed by matrix multiplications, all of the inputs affect s_3

PARAMETER SHARING

- Parameter sharing refers to use the same parameter for more than one function in a model.
  - In traditional networks each element of the weight matrix is used exactly once when computing the output of a layer: it is multiplied by the input and then never revisited.
  - In a CNN each member of the kernel is used at every position of the input (except for the boundaries).
  - Rather then learning a separate set of parameters for every location, we learn one set.

PARAMETER SHARING

- Forward propagation runtime, O(kxn)
- Storage requirement: k parameters
- k several orders of magnitude smaller than m and n

Convolution is thus dramatically more efficient than dense matrix multiplication in terms of memory requirements and statistical efficiency

PARAMETER SHARING

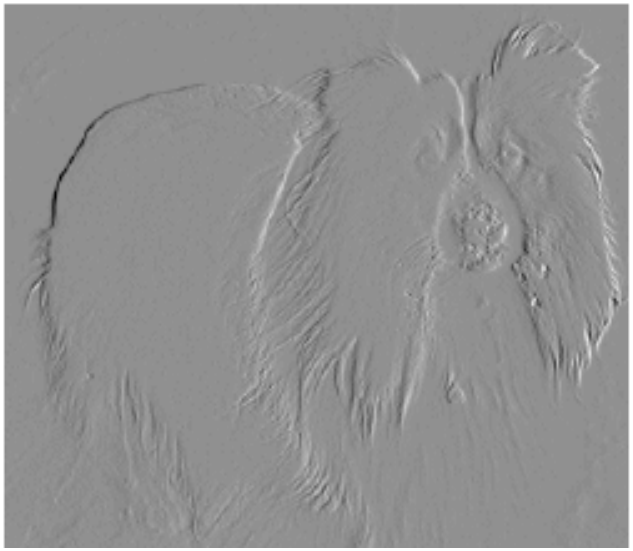
- Forward propagation runtime, O(kxn)
- Storage requirement: k parameters
- k several orders of magnitude smaller than m and n

Convolution is thus dramatically more efficient than dense matrix multiplication in terms of memory requirements and statistical efficiency

PARAMETER SHARING

- Edge detection: second image obtained from the first by taking each pixel and subtracting the value of its neighboring pixel on the left
- the transformation can be described by a convolution kernel containing two elements
- Original image 320x280 Output image 319x280

•[Figure from http://deeplearningbook.org/]



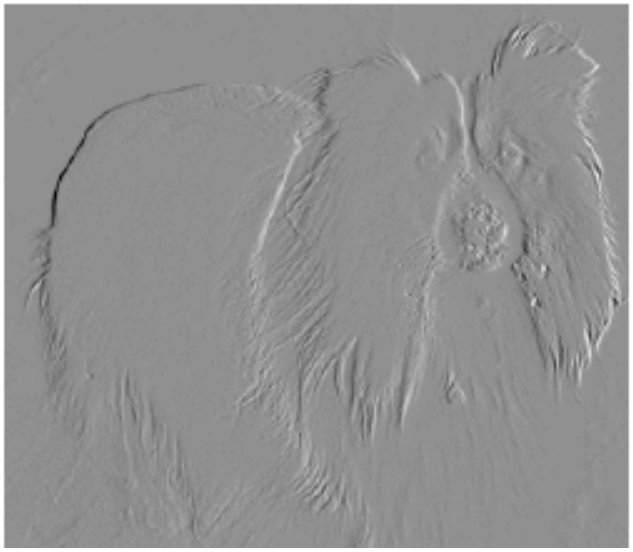
PARAMETER SHARING

- Floating point operations with convolution

  319 x 280 x 3 = 267960

Floating point operations with matrix multiplication

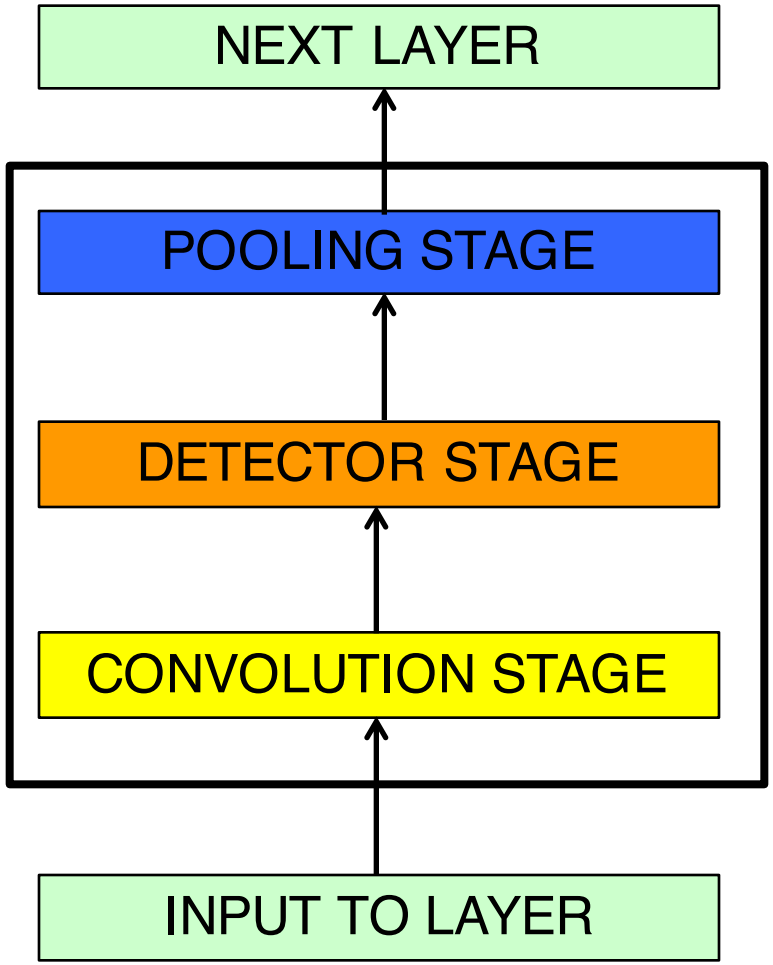
  320 x 280 x 319 x 280 > 8 billions



EQUIVARIANCE

- In case of convolution, the particular form of parameter sharing causes the layer to have a property called **equivariance** to translation
- f(x) is equivariant to a function g if f(g(x)) = g(f(x))
  - if the input changes, the output changes the same way
- Convolution is not equivariant to rotations, changes in scale etc.

TYPICAL CONVOLUTIONAL LAYER



The pooling function replaces the output of the net at a certain location with a summary statistics of nearby outputs

Each activation is run through a non linear activation function - ReLu

It performs several convolutions in parallel to produce a set of linear activations

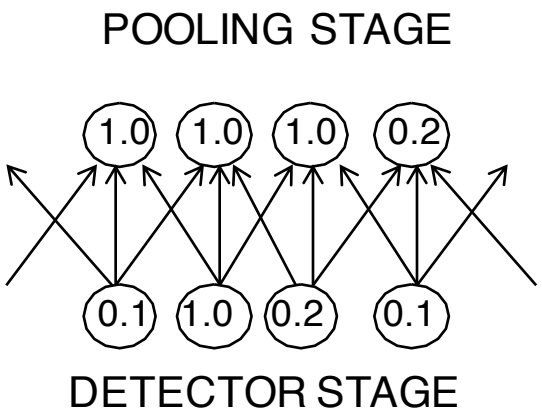
POOLING

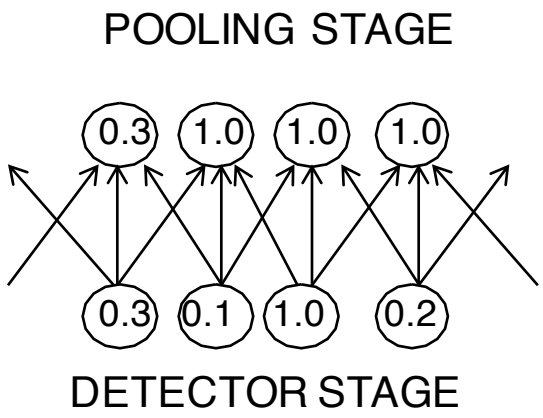
- Pooling aggregates the output at a certain location with a summary statistic of the nearby output.
- Common pooling functions:
  - **max pooling** operation reports the max output within a rectangular neighborhood
  - **average** of a rectangular neighborhood
  - **L^2 norm** of a rectangular neighborhood
  - **weighted average** based on the distance from the central pixel.

- Pooling helps to make a representation become approx invariant to small translations of the input.

POOLING

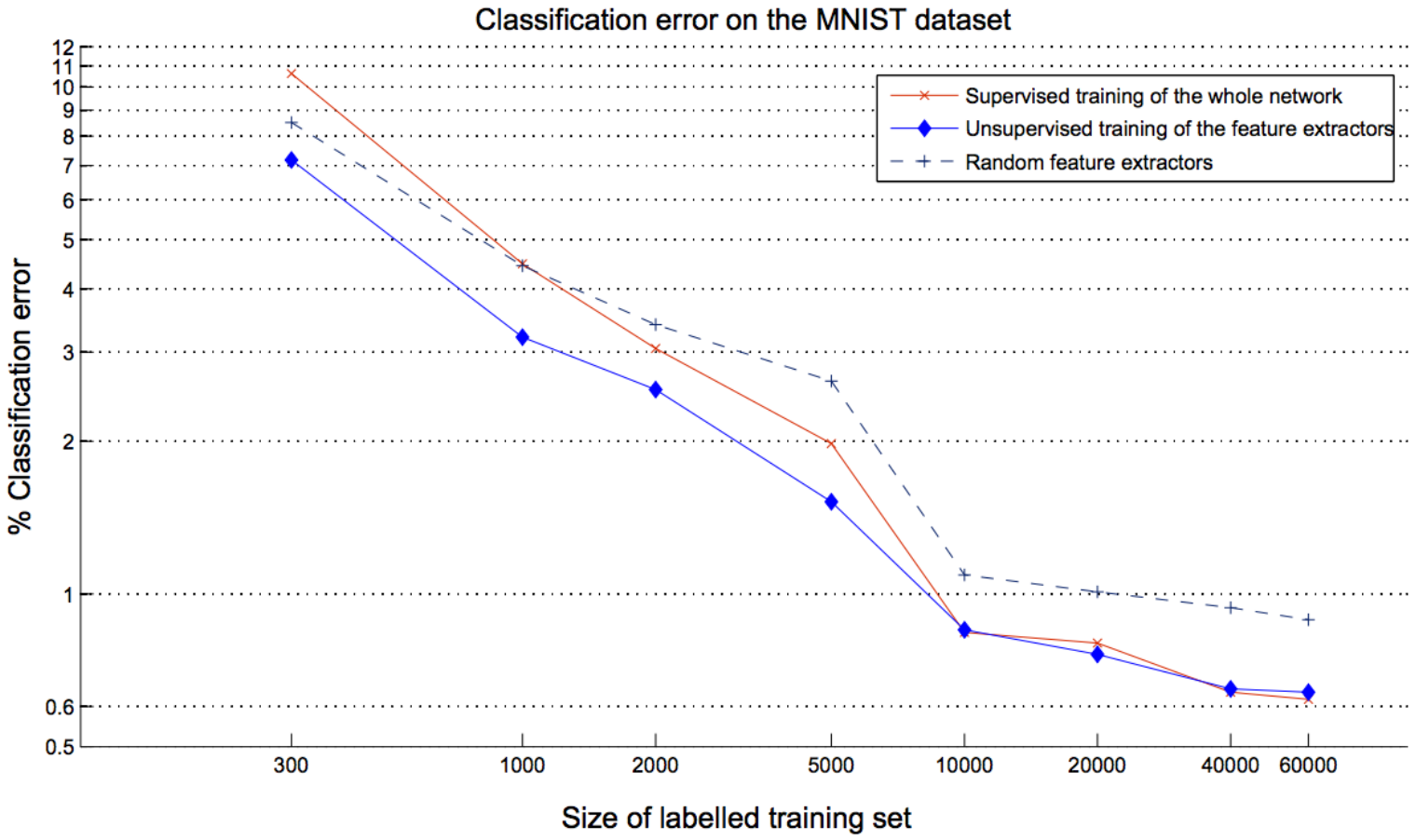
- Pooling helps to make a representation become approx invariant to small translations of the input.
  - if we translate the input by a small amount the values of most of the pooled outputs do not change





Convolutional neural networks

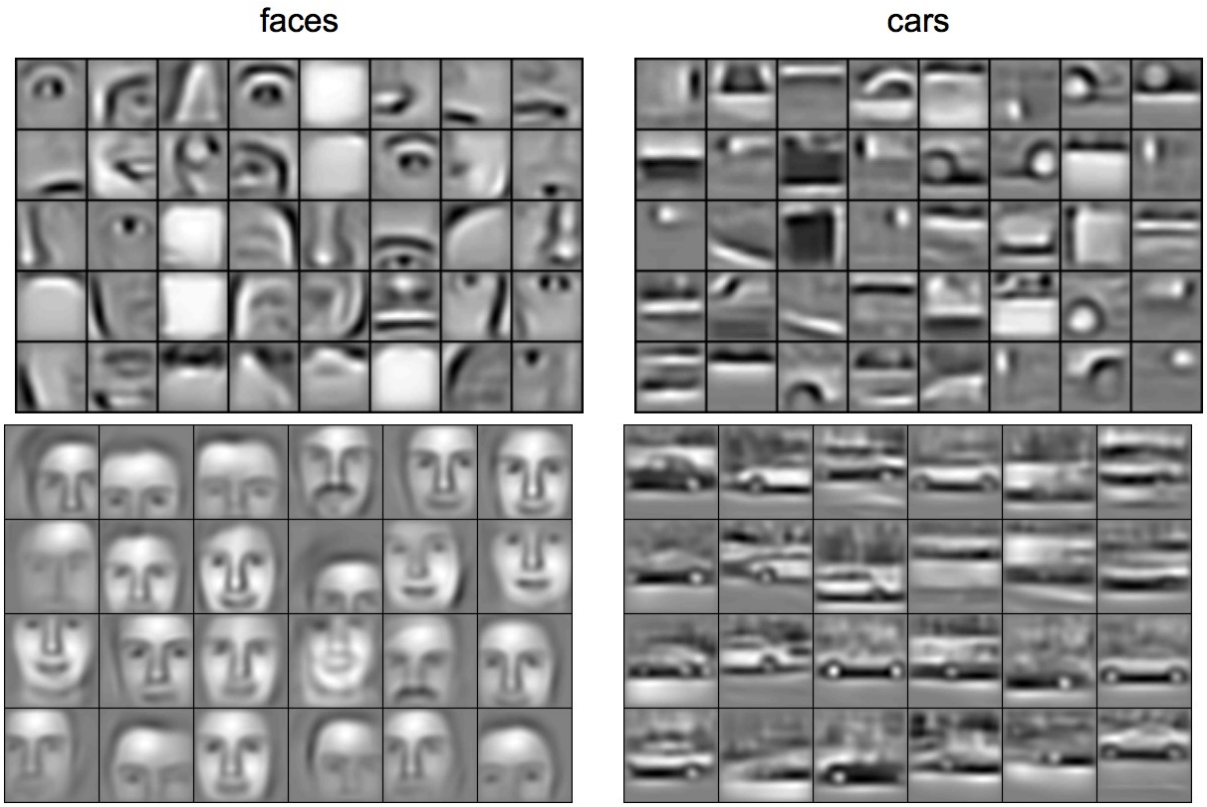
- A crucial advantage must be **in the structure** of a CNN !



- [Figure from Ranzato et al., 2007]

FEATURE EXTRACTION

- Features obtained with various natural images



[Figures by Lee et al., 2009]

ImageNet

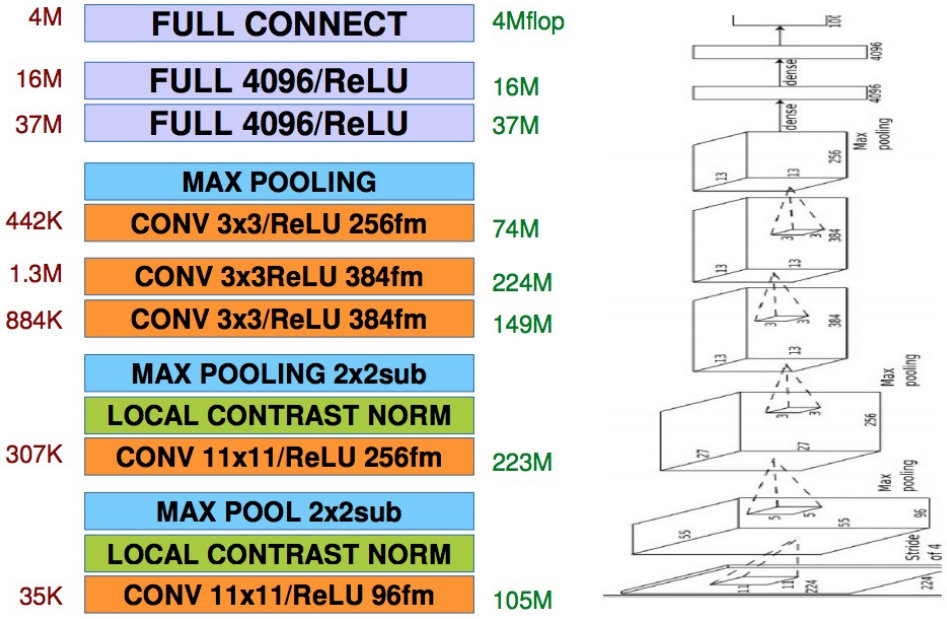
- The dream: build a computer vision system capable of **recognizing thousands of object categories**
  - over 14 millions images
  - over 20 thousands categories tagged via **crowdsourcing** !
  - organized according to the **WordNet** noun hierarchy

[Figure from vision.stanford.edu]

The ImageNet challenge

- **Annual competition**:
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)

- Breakthrough in 2012 by AlexNet [Krizevsky et al., 2012]*



The ImageNet challenge

- Breakthrough in 2012 by AlexNet [Krizevsky et al., 2012]
  - ~60,000,000 parameters
  - ~650,000 neurons
  - trained for **5-6 days** with 2 GPUs in parallel
  - achieved a top-5 error rate of 18.2 % (second best 26.2 %) reduced to 15.4 % with multiple models and pre-training
  - achieved a top-1 error rate 40.7 %
  - reduced to 36.7 % with multiple models and pre-training

The ImageNet challenge

- Predictions

- [Figure from Krizhevsky et al., 2012]

The ImageNet challenge

- Features of the 2014 edition:
  - 1.2 million images for training
    50k images for validation 100k
    images for test
  - 1,000 semantic categories
  - Winner: **GoogLeNet** (22 layers !!!) → 6.67 % Top-5 error

- In 2015 **ResNet** (Microsoft Research) → 3.6 % top-5 error !!!
  - They employed a 152-layer net !!!
  - They won all the tasks (localization, detection, segmentation)