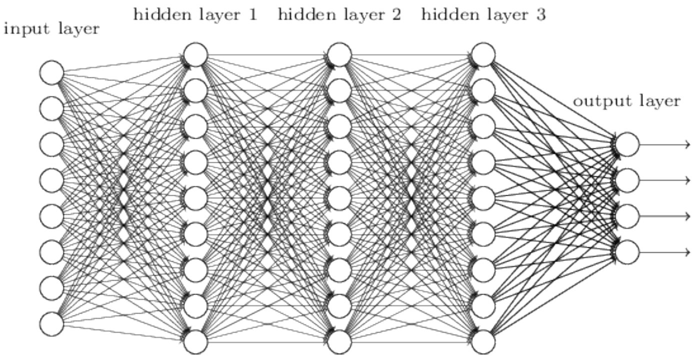
Convolutional Networks

Hakan Bilen

Machine Learning Practical — MLP Lecture 7
30 October / 6 November 2018

Recap: Fully-connected network for MNIST

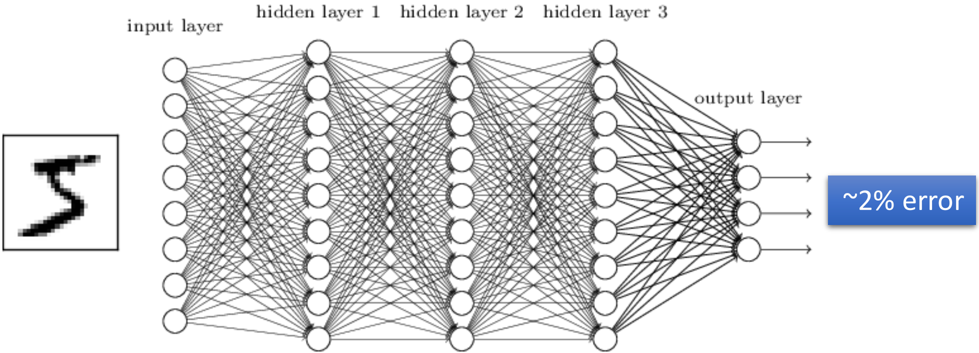


(image from: Michael Nielsen, *Neural Networks and Deep Learning*, http://neuralnetworksanddeeplearning.com/chap6.html)

Slide credits: S Renals' MLP 2017-18

Recap: Fully-connected network for MNIST

On MNIST, we can get about 2% error (or even better) using these kind of networks



(image from: Michael Nielsen, *Neural Networks and Deep Learning*, http://neuralnetworksanddeeplearning.com/chap6.html)

How about more complex image recognition tasks?

- Large variations in position, appearance, shape and size within same object category
- Small variations in appearance between different object categories
- Background clutter and occlusions
- Typical input image size is 227×227

Fully-connected network in high dimension

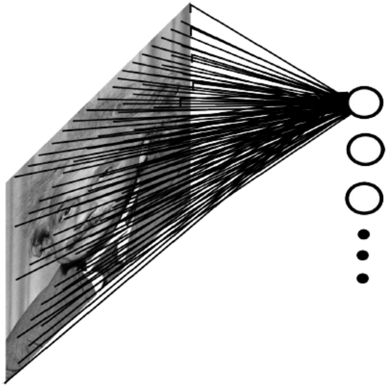


image credit: Lecun & Ranzato

Fully-connected network in high dimension

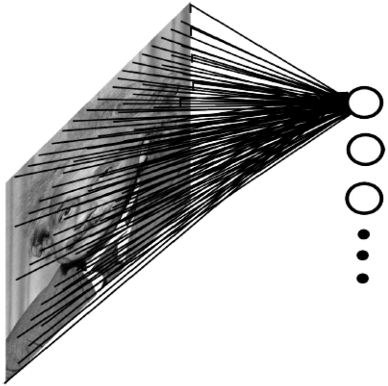


image credit: Lecun & Ranzato

For a 200×200 image and 1000 hidden units

- # input units is 40,000
- # hidden units is 1000
- # connections is 40,000,000
- # parameters is 40,000,000

Fully-connected network in high dimension

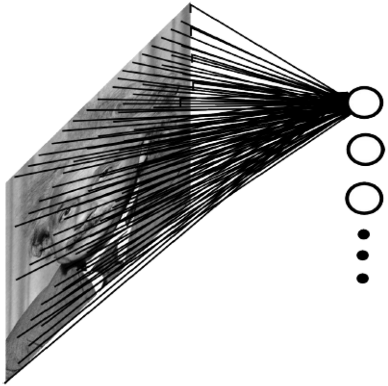


image credit: Lecun & Ranzato

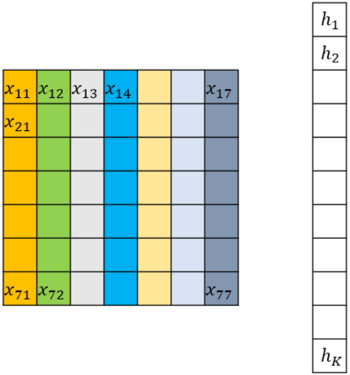
For a 200×200 image and 1000 hidden units

- # input units is 40,000
- # hidden units is 1000
- # connections is 40,000,000
- # parameters is 40,000,000

Observations:

- Too many parameters to learn!
- Spatial (2-D) structure of input image is ignored
- Neighbour pixels are treated separately

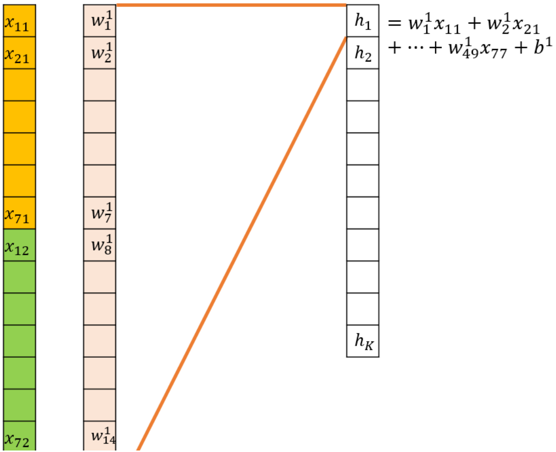
A closer look at fully connected nets



Assume that we have

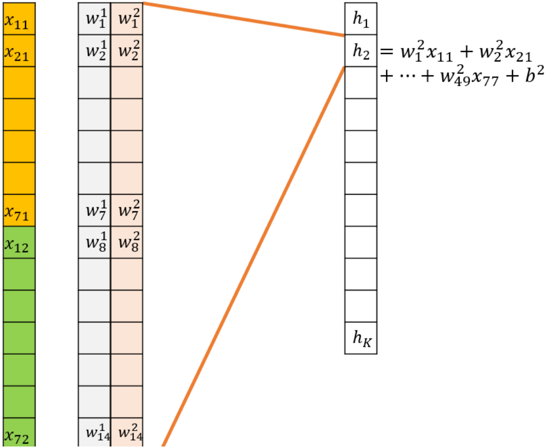
- $7{\times}7$ image X
- K hidden units

A closer look at fully connected nets



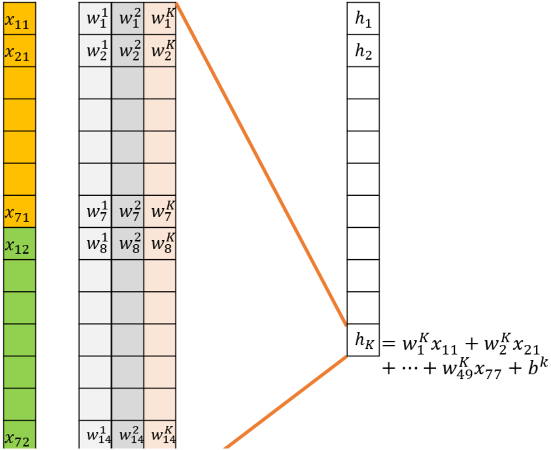
- Unroll the input (7×7) into 49-D
- Affine parameters $W \in \mathcal{R}^{49 \times K}$ and $b \in \mathcal{R}^K$

A closer look at fully connected nets



- Unroll the input (7×7) into 49-D
- Affine parameters $W \in \mathcal{R}^{49 \times K}$ and $b \in \mathcal{R}^K$

A closer look at fully connected nets



- Connections are dense
- Hidden unit h_k is connected to all input units x_{ij} through w_{ij}^k
- It does not know that x_{11} is adjacent to x_{12}

Convolutional networks

x_{11}	x_{12}	x_{13}				x_{17}
x_{21}						
x_{71}	x_{72}					x_{77}

Input X

w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}

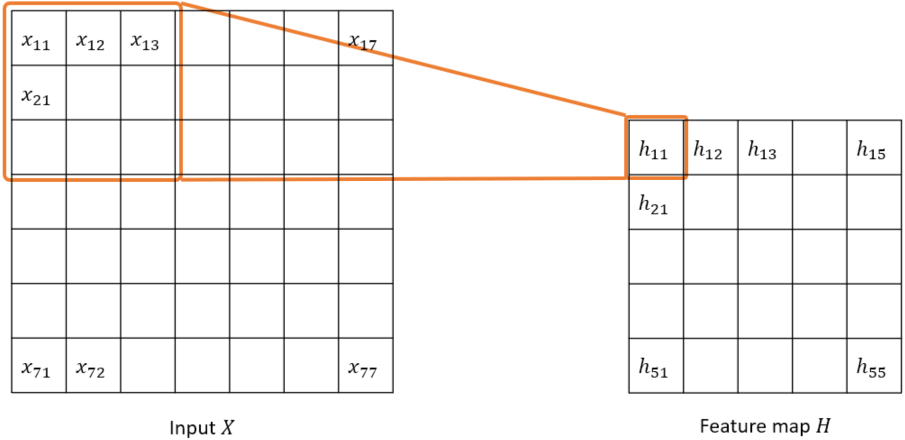
Convolution kernel W

h_{11}	h_{12}	h_{13}		h_{15}
h_{21}				
h_{51}				h_{55}

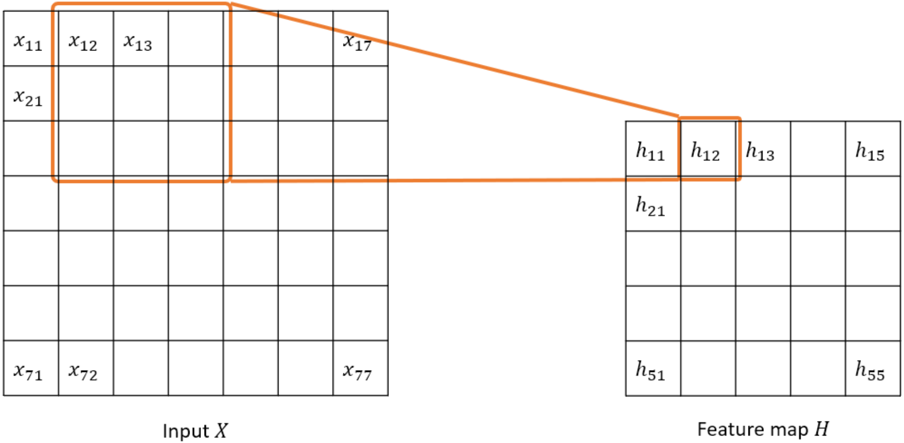
Feature map H

Convolutional networks

$$h_{11} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i,j} + b$$

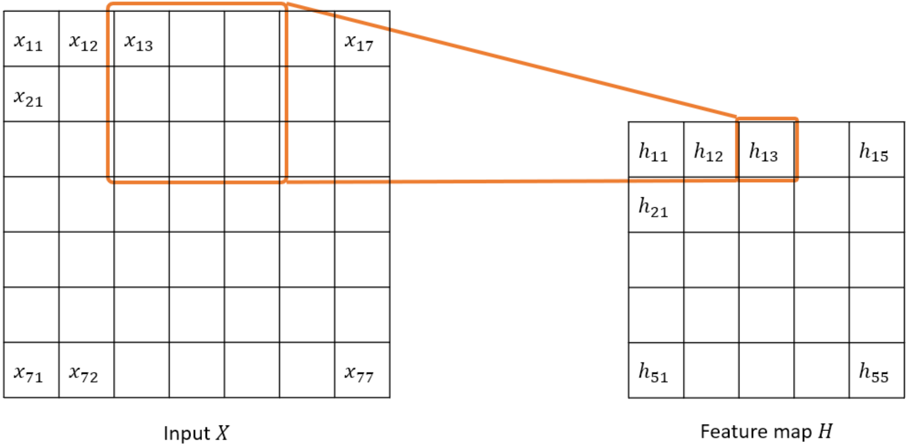


$$h_{12} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i,j+1} + b$$



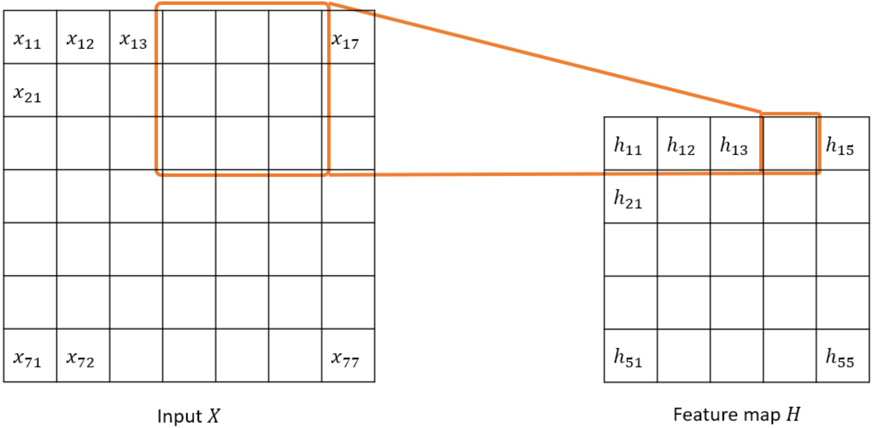
Convolutional networks

$$h_{13} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i,j+2} + b$$



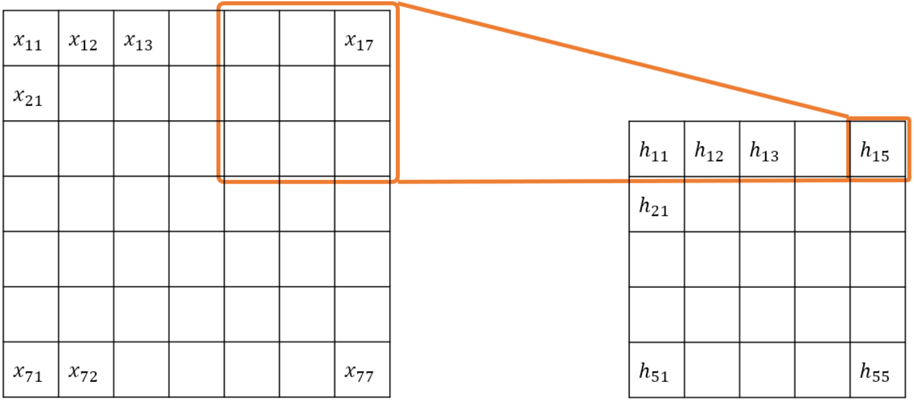
Convolutional networks

$$h_{14} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i,j+3} + b$$



Convolutional networks

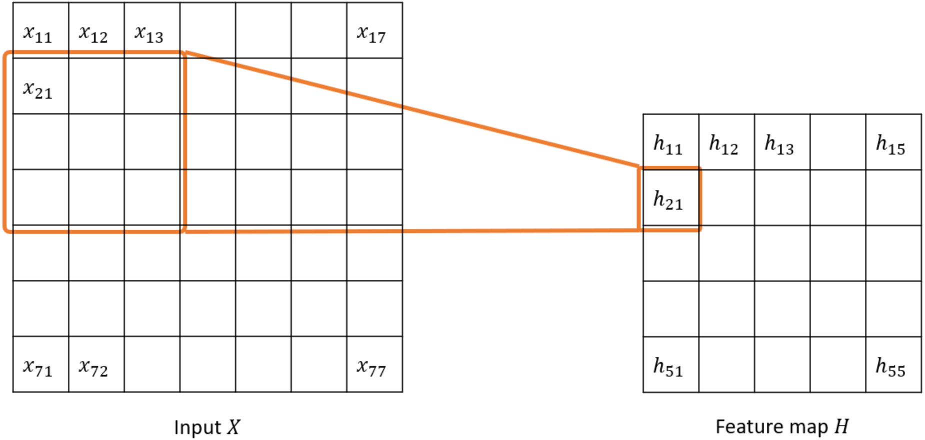
$$h_{15} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i,j+4} + b$$



Input X

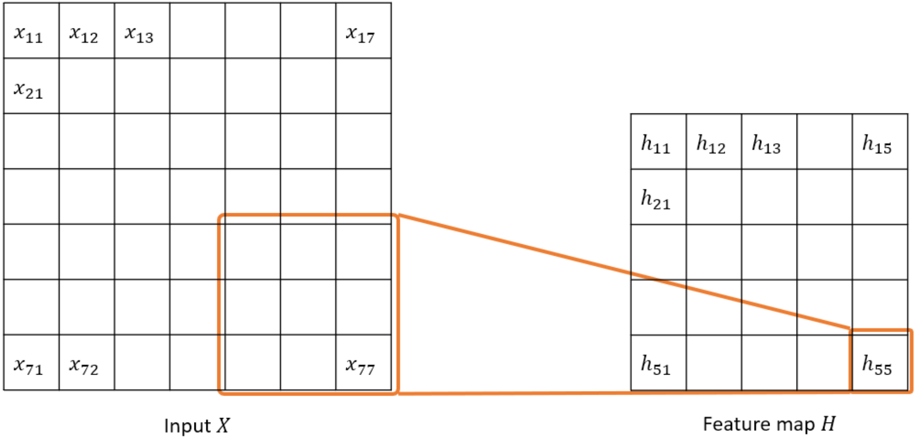
Feature map H

$$h_{21} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i+1,j} + b$$



Convolutional networks

$$h_{55} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i+4,j+4} + b$$



Convolutional networks

Number of ...

- parameters is $3 \times 3 + 1$ (9 for kernel + 1 for bias)
- hidden units is 5×5
- connections is $5 \times 5 \times 3 \times 3$

Convolutional networks

Number of ...

- parameters is $3 \times 3 + 1$ (9 for kernel + 1 for bias)
- hidden units is 5×5
- connections is $5 \times 5 \times 3 \times 3$

Properties

- Weights (conv kernel) are shared across all hidden units
- Spatial correspondence between pixels and hidden units ("2D matrix of hidden units" = "feature map")
- Translation invariance: extract same features irrespective of where an image patch is located in the input

Question

If $X \in \mathcal{R}^{7\times7}$ and $W \in \mathcal{R}^{3\times3}$, the feature map dimensionality $H \in \mathcal{R}^{5\times5}$.

Question

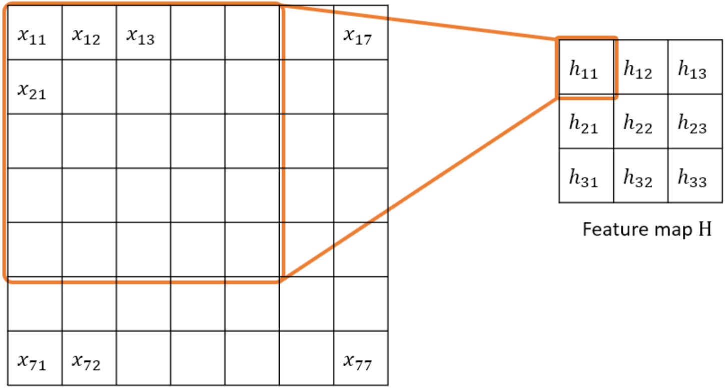
If $X \in \mathcal{R}^{7\times 7}$ and $W \in \mathcal{R}^{5\times 5}$, what will be the feature map dimensionality H?
(a) 3×3, (b) 5×5, (c) 7×7

Question

If $X \in \mathcal{R}^{7\times7}$ and $W \in \mathcal{R}^{5\times5}$, what will be the feature map dimensionality H?

(a) 3×3, (b) 5×5, (c) 7×7

$$h_{11} = \sum_{i=1}^{5} \sum_{j=1}^{5} w_{i,j} x_{i,j} + b$$

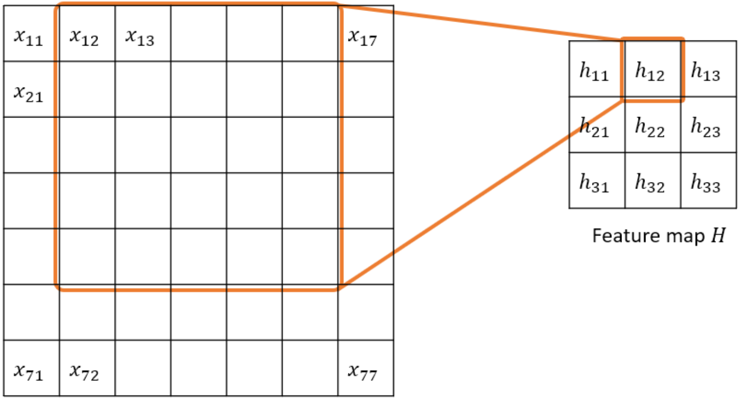


Question

If $X \in \mathcal{R}^{7\times7}$ and $W \in \mathcal{R}^{5\times5}$, what will be the feature map dimensionality H?

(a) 3×3, (b) 5×5, (c) 7×7

$$h_{12} = \sum_{i=1}^{5} \sum_{j=1}^{5} w_{i,j} x_{i,j+1} + b$$



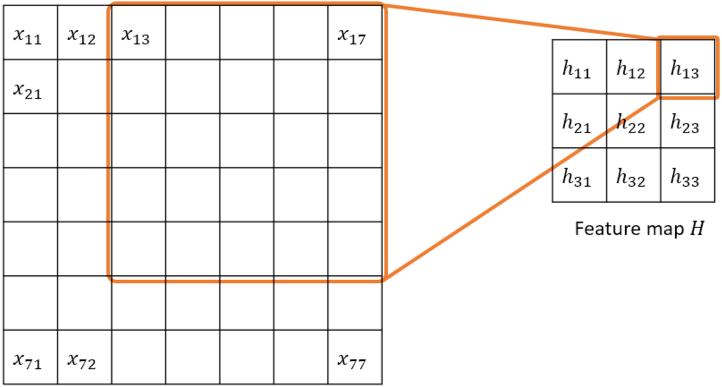
Feature map H

Question

If $X \in \mathcal{R}^{7\times 7}$ and $W \in \mathcal{R}^{5\times 5}$, what will be the feature map dimensionality H?

(a) 3×3, (b) 5×5, (c) 7×7

$$h_{13} = \sum_{i=1}^{5} \sum_{j=1}^{5} w_{i,j} x_{i,j+2} + b$$



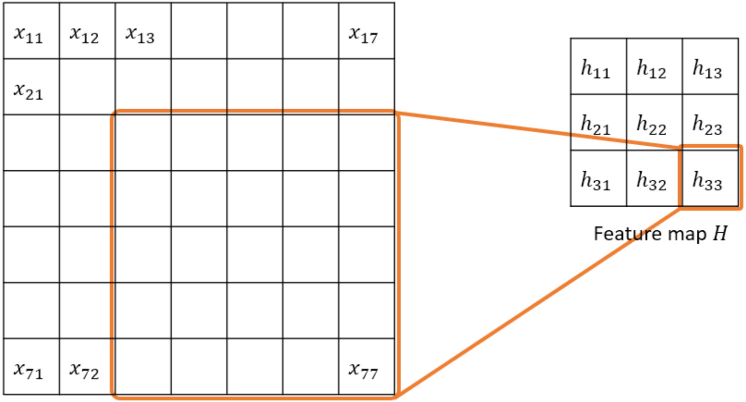
Feature map H

Question

If $X \in \mathcal{R}^{7\times7}$ and $W \in \mathcal{R}^{5\times5}$, what will be the feature map dimensionality H?

(a) 3×3, (b) 5×5, (c) 7×7

$$h_{33} = \sum_{i=1}^{5} \sum_{j=1}^{5} w_{i,j} x_{i+2,j+2} + b$$



Calculating the output size

Q1. If $X \in \mathcal{R}^{M \times N}$ and $W \in \mathcal{R}^{F \times F}$, what will the output dimensionality be?

Calculating the output size

Q1. If $X \in \mathcal{R}^{M \times N}$ and $W \in \mathcal{R}^{F \times F}$, what will the output dimensionality be?

A. $(M - F + 1) \times (N - F + 1)$

Calculating the output size

Q1. If $X \in \mathcal{R}^{M \times N}$ and $W \in \mathcal{R}^{F \times F}$, what will the output dimensionality be?

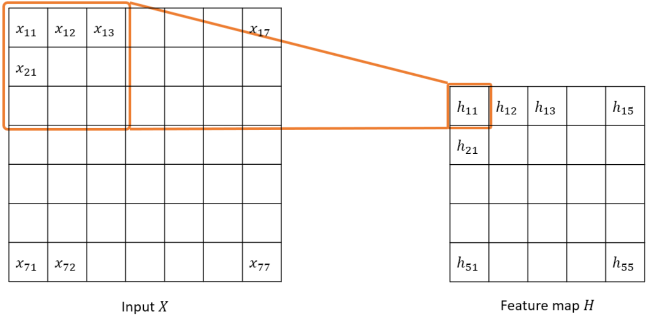
A. $(M - F + 1) \times (N - F + 1)$

Q2. Feature map formula?

A. $h_{ij} = \sum_{k=1}^{F} \sum_{l=1}^{F} w_{k,l} x_{k+i-1, l+j-1} + b$

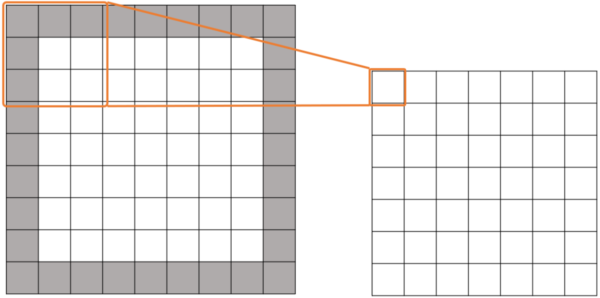
Input size = feature map size

Q. What can we do to get 7×7 feature map size?



Padding

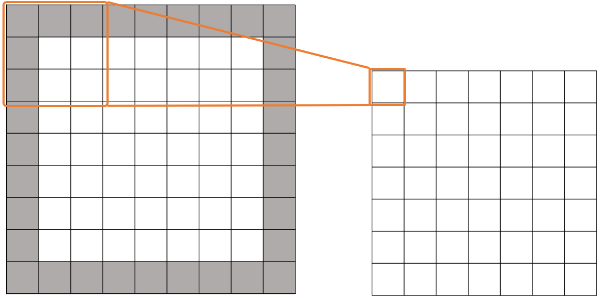
Q. What can we do to get 7×7 feature map size?



Padded input X Feature map H

Padding

Q. What can we do to get 7×7 feature map size?



Padded input X Feature map H

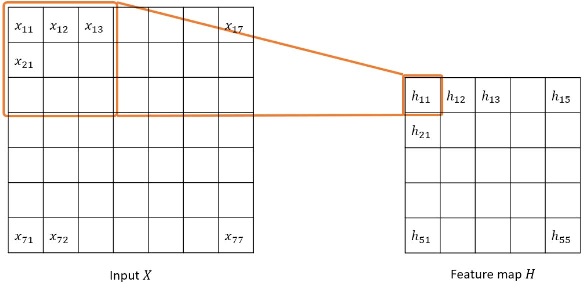
Take-home

What is feature map size when $X \in \mathcal{R}^{M \times N}$, $W \in \mathcal{R}^{F \times F}$ and padding P?

Stride

Q. What if stride (s) is 2?

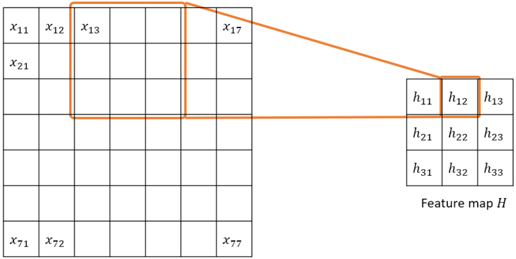
$$h_{11} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i,j} + b$$



Stride

Q. What if stride (s) is 2?

$$h_{12} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i,j+2} + b$$

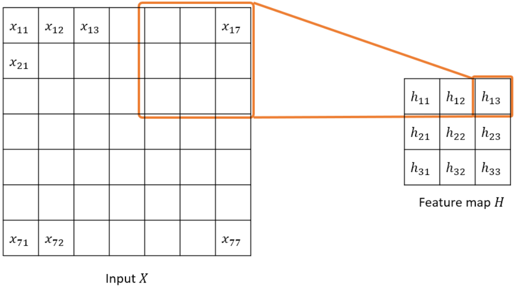


Input X

Feature map H

Q. What if stride (s) is 2?

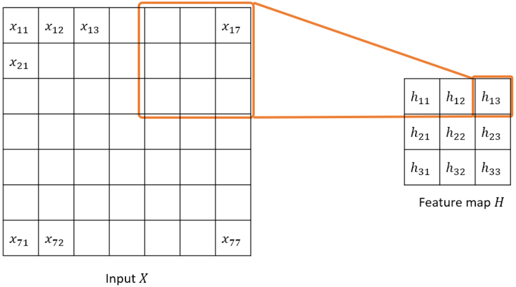
$$h_{13} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i,j+4} + b$$



Stride

Q. What if stride (s) is 2?

$$h_{13} = \sum_{i=1}^{3} \sum_{j=1}^{3} w_{i,j} x_{i,j+4} + b$$

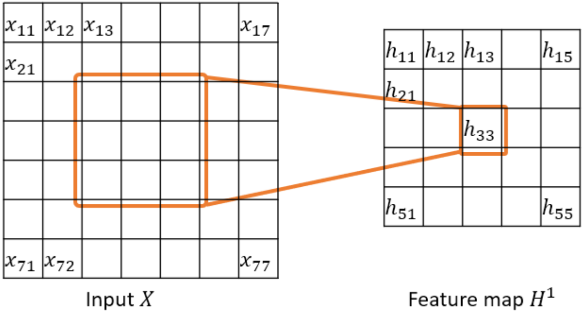


Take-home

What is feature map size when $X \in \mathcal{R}^{M \times N}$, $W \in \mathcal{R}^{F \times F}$, padding P and stride S?

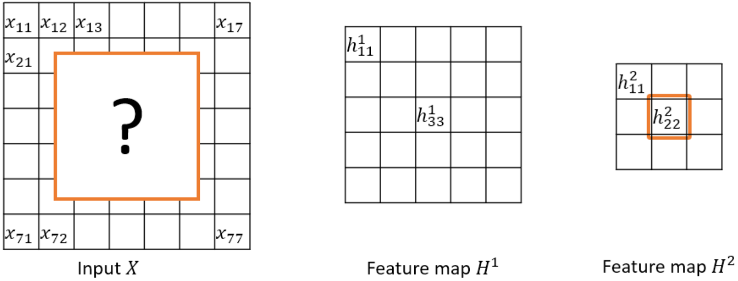
Receptive field

- Biology: The **receptive field** of an individual sensory neuron is the particular region of the sensory space,
- Convolutional networks: The region in the input space that a hidden unit is looking at.



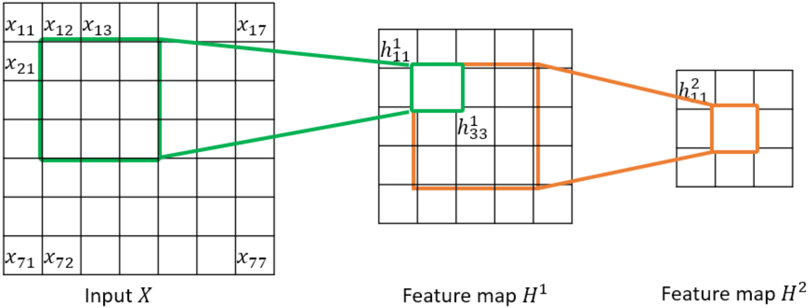
Receptive field

Q. Assume that $X \in \mathcal{R}^{7\times7}$, $W^1 \in \mathcal{R}^{3\times3}$, $W^2 \in \mathcal{R}^{3\times3}$.
Receptive field of a hidden unit in second convolutional layer?
3, 5, 6, 7?



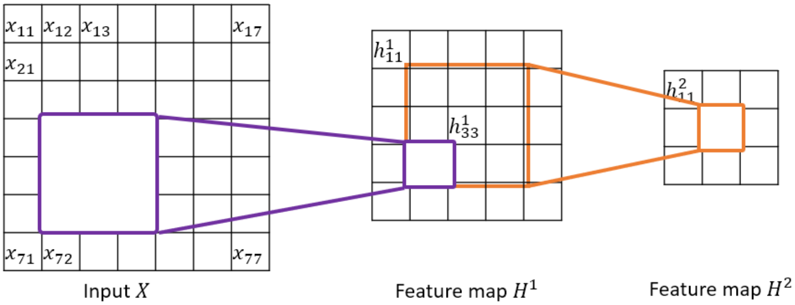
Receptive field

Receptive field of a hidden unit in second convolutional layer?



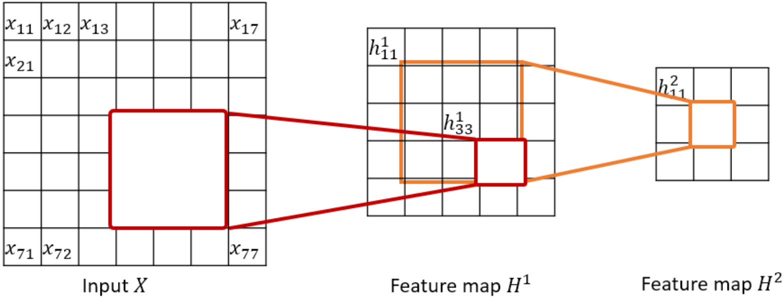
Receptive field

Receptive field of a hidden unit in second convolutional layer?

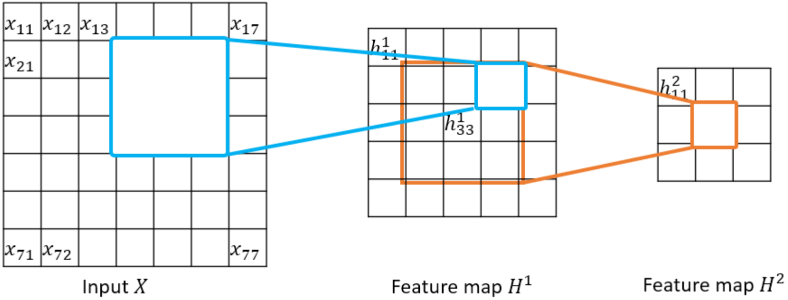


Receptive field

Receptive field of a hidden unit in second convolutional layer?

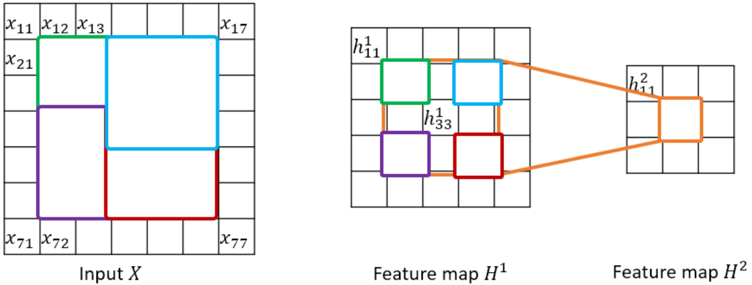


Receptive field of a hidden unit in second convolutional layer?



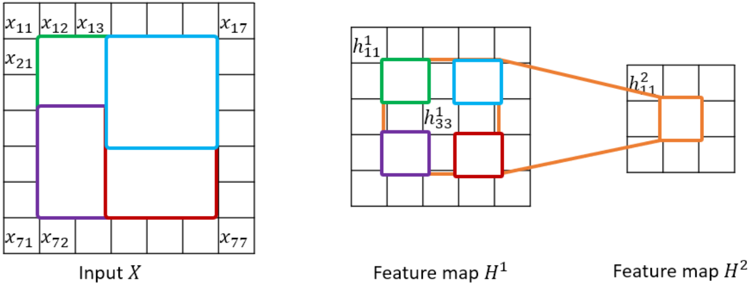
Receptive field

Receptive field of a hidden unit in second convolutional layer?



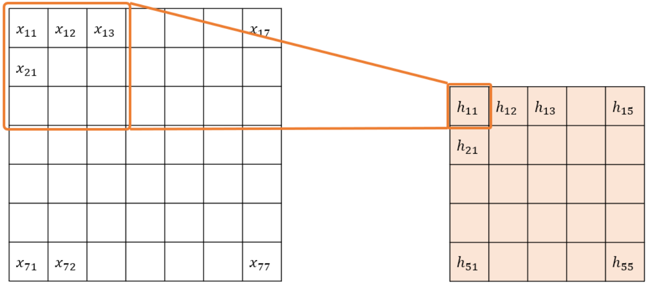
Receptive field

Receptive field of a hidden unit in second convolutional layer?



Q. What would be the receptive field for a hidden unit in an one-layer fully-connected network?

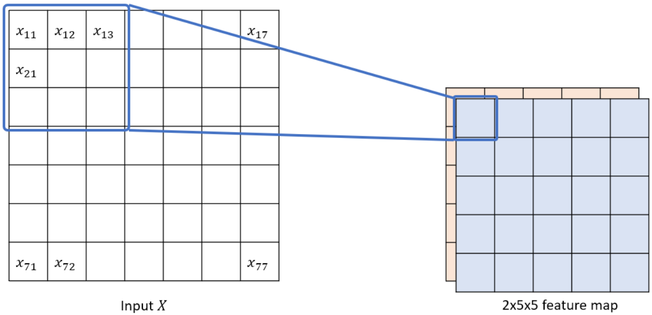
Multiple output feature maps



Input X

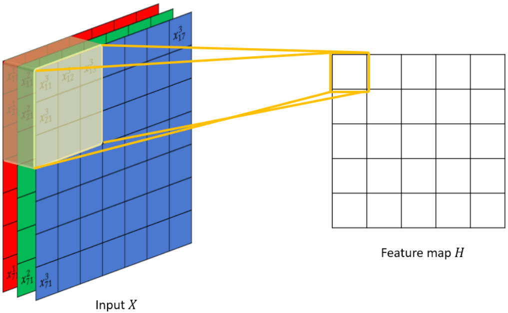
5x5 feature map

Multiple output feature maps



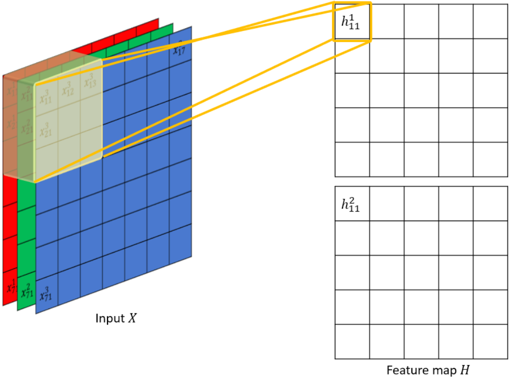
- \# feature maps is $F_{\text{out}} = 2$
- \# hidden units is $F_{\text{out}} \times (5 \times 5)$
- \# of parameters is $F_{\text{out}} \times (3 \times 3 + 1)$
- \# of connections is $F_{\text{out}} \times (5 \times 5 \times 3 \times 3)$

Multiple input feature maps (or input images)



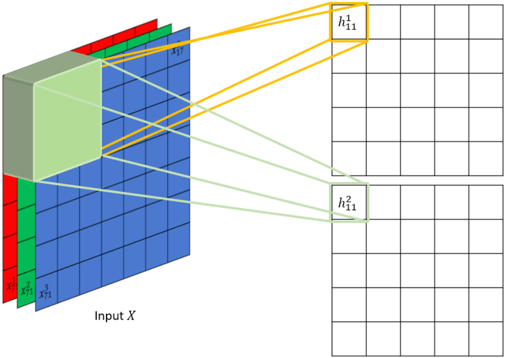
- $\#$ input image is $F_{\text{in}} = 3$
- $\#$ input units is $F_{\text{in}} \times 7 \times 7$
- $\#$ hidden units is 5×5
- $\#$ parameters is $F_{\text{in}} \times 3 \times 3 + 1$ for bias ($F_{\text{in}} \times 3 \times 3 + 1$)
- $\#$ connections is $F_{\text{in}} \times 5 \times 5 \times 3 \times 3$
- Typically we do not tie weights across feature maps
- Local receptive fields across multiple input images

Multiple input and output feature maps



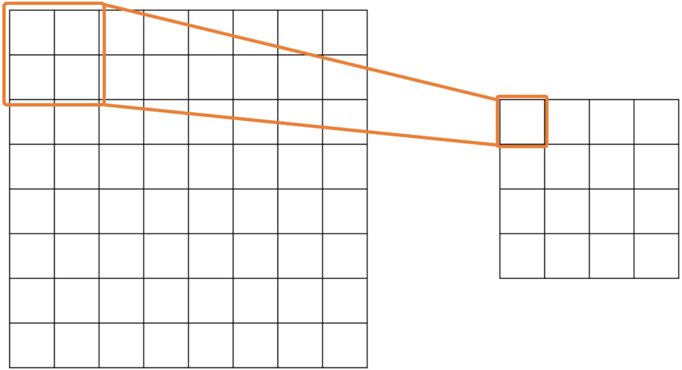
- $F_{\text{in}} = 3$ and $F_{\text{out}} = 2$

Multiple input and output feature maps

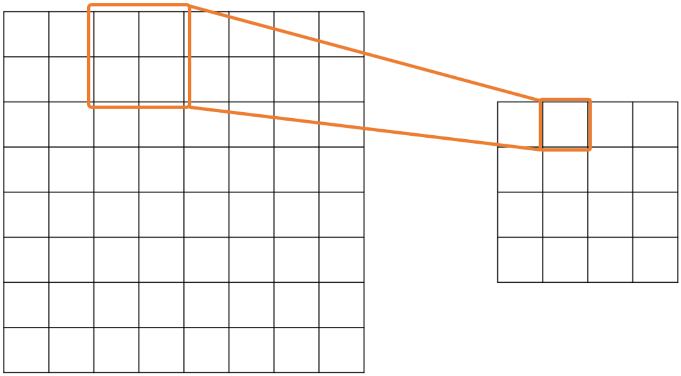


- $F_{\text{in}} = 3$ and $F_{\text{out}} = 2$
- # input units is $F_{\text{in}} \times 7 \times 7$
- # hidden units is $F_{\text{out}} \times 5 \times 5$
- # parameters is $F_{\text{in}} \times F_{\text{out}} \times 3 \times 3 + F_{\text{out}}$ for bias

Pooling (subsampling)



Pooling (subsampling)



- Similar to convolution, slides over input pixels but no learnable parameters
- Has local receptive field too
- Typical stride $S > 1$

Pooling

- Pooling or subsampling takes a feature map and reduces it in size – e.g. by transforming a set of 2x2 regions to a single unit
- Reduces computation time and memory use
- Pooling functions
  - Max-pooling – takes the maximum value of the units in the region
  - L_p-pooling – take the L_p norm of the units in the region:

$$h' = \left(\sum_{i \in \text{region}} h_i^p \right)^{1/p}$$

  - Average- / Sum-pooling – takes the average / sum value of the pool
- Information reduction – pooling removes precise location information for a feature
- Apply pooling to each feature map separately

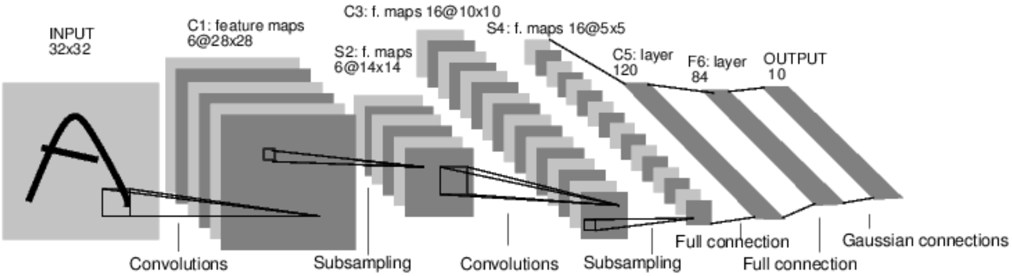
Learning image kernels

Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	

https://en.wikipedia.org/wiki/Kernel_(image_processing)

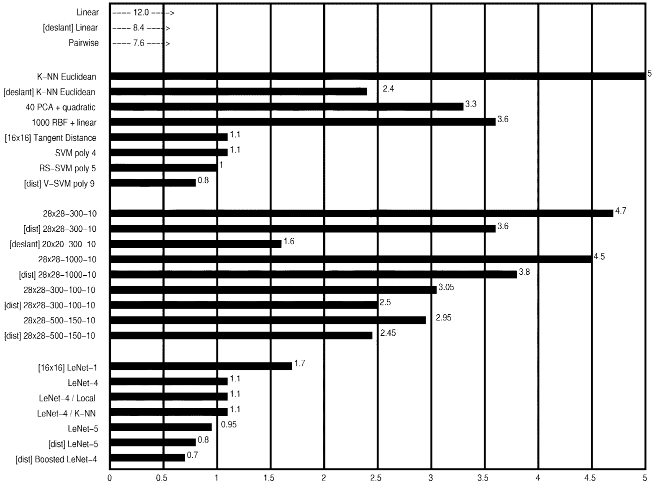
- Image kernels have been **hand designed** and used for feature extraction in image processing (e.g. edge detection)
- Pros: No need for data and training
- Cons 1: Learning filters can be more optimal (minimising network cost function)
- Cons 2: Difficult to design filters for complex tasks (e.g. recognising a cat)
- Automating feature engineering

Example: LeNet5 (LeCun et al, 1997)



- Convolutional layer (convolution + non-linearity)
- Subsampling (max pooling)
- Final fully connected hidden layer (no weight sharing)
- Softmax output layer

MNIST Results (1997)



ImageNet Classification ("AlexNet")

Krizhevsky, Sutskever and Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", NIPS'12.

http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

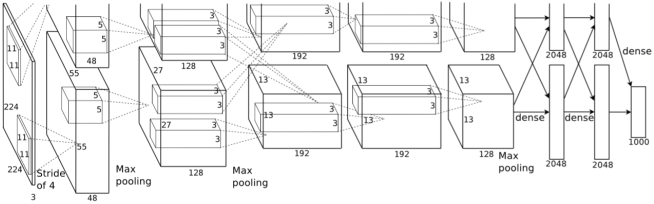
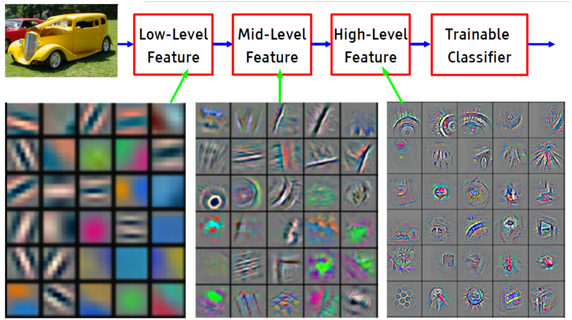


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Hierarchical Representations

Pixel $\rightarrow$ edge $\rightarrow$ texton $\rightarrow$ motif $\rightarrow$ part $\rightarrow$ object



Zeiler & Fergus, "Visualizing and Understanding Convolutional Networks", ECCV'14.

https://cs.nyu.edu/~fergus/papers/zeilerECCV2014.pdf

Slide credits: Lecun & Ranzato

Training Convolutional Networks

- Train convolutional networks with a straightforward but careful application of backprop / SGD
- Exercise: prior to the next lecture, write down the gradients for the weights and biases of the feature maps in a convolutional network. Remember to take account of weight sharing.
- Next lecture: implementing convolutional networks: how to deal with local receptive fields and tied weights, computing the required gradients...

Summary

- Convolutional networks include local receptive fields, weight sharing, and pooling leading to:
  - Modelling the spatial structure
  - Translation invariance
  - Local feature detection
- Reading:

  Michael Nielsen, *Neural Networks and Deep Learning* (ch 6)

  http://neuralnetworksanddeeplearning.com/chap6.html

  Yann LeCun et al, "Gradient-Based Learning Applied to Document Recognition", *Proc IEEE*, 1998.

  http://dx.doi.org/10.1109/5.726791

  Ian Goodfellow, Yoshua Bengio & Aaron Courville,
  Deep Learning (ch 9)

  http://www.deeplearningbook.org/contents/convnets.html