



#DLBCN

AA2 Deep Learning: Lecture

Convolutional Neural Networks (CNN)



Xavier Giro-i-Nieto

xavier.giro@upc.edu

Associate Professor

Universitat Politècnica de Catalunya
Technical University of Catalonia



[GCED] [TelecomBCN DLAI]

Today's objective

- Distinguish between a fully connected vs a convolutional neural network.
- Value the translation invariance properties of convolutional filters

Slides by



Míriam Bellver

miriam.bellver@bsc.edu

PhD Candidate

Barcelona Supercomputing Center



Outline

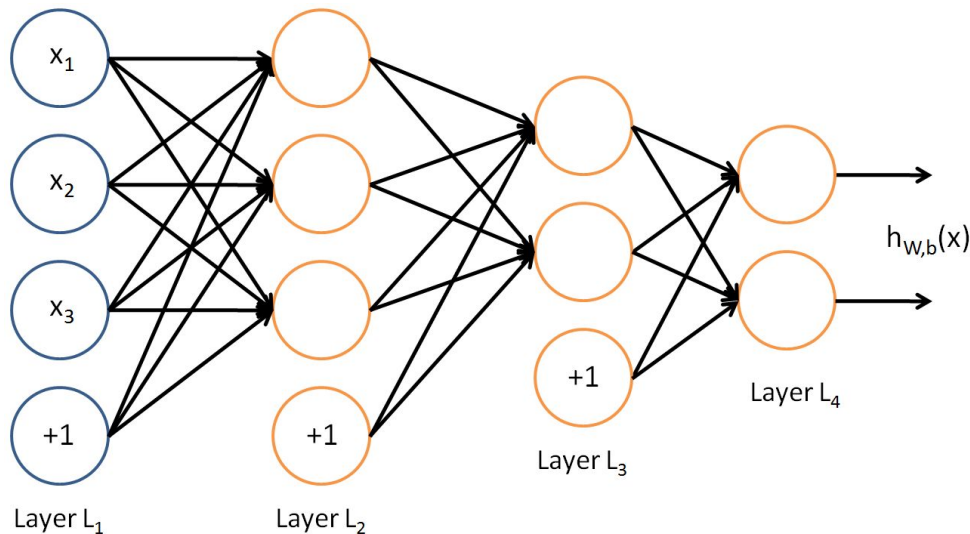
1. Limitations of Multi-Layer Perceptrons
2. Convolutional Layers
3. Pooling Layers
4. ConvNets Architectures for Computer Vision
5. Summary

Outline

1. **Limitations of Multi-Layer Perceptrons**
2. Convolutional Layers
3. Pooling Layers
4. ConvNets Architectures for Computer Vision
5. Summary

Limitations of MLPs

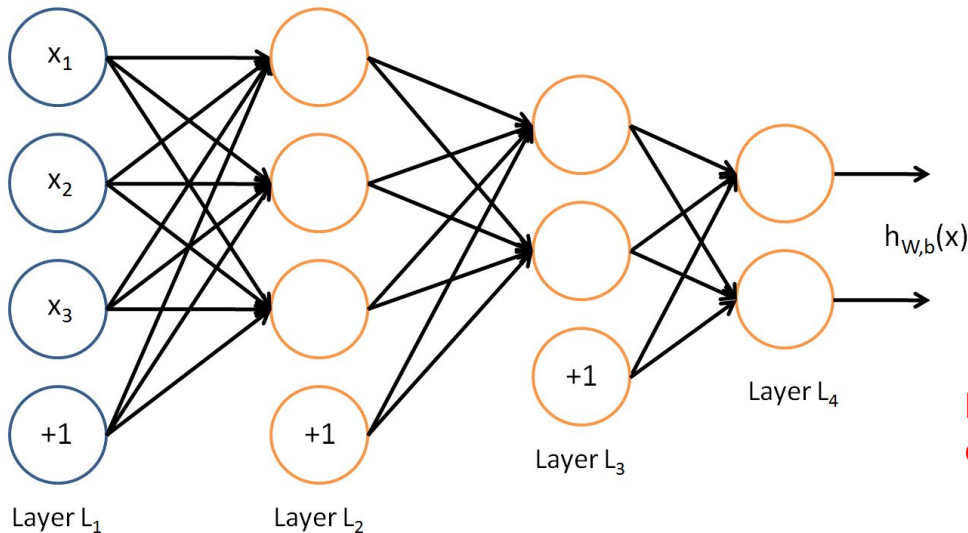
Multi-Layer Perceptrons (MLPs) are deep learning networks with several fully connected layers.



Network with two hidden layers and two outputs

Limitations of MLPs

Multi-Layer Perceptrons are deep learning networks with several fully connected layers.

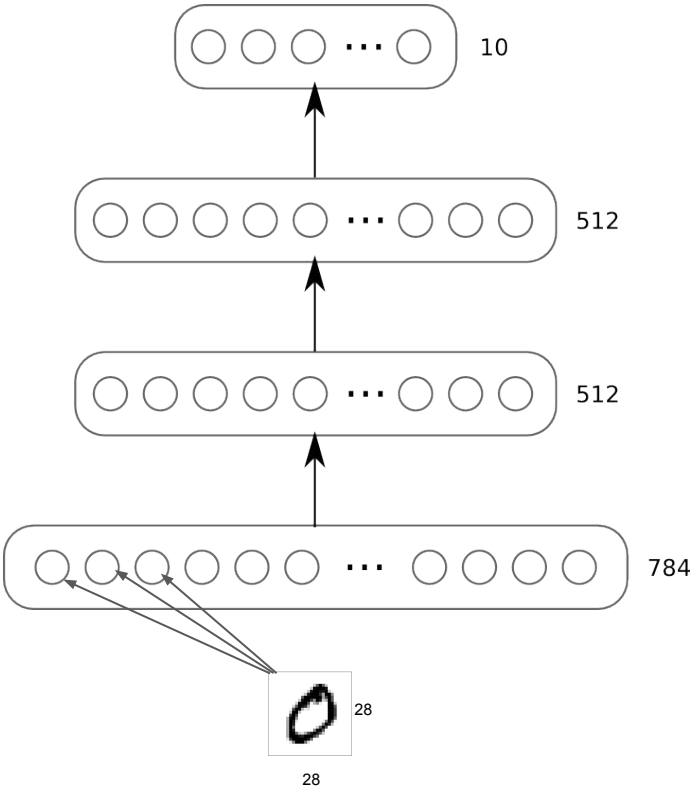


Network with two hidden layers and two outputs

Each layer is connected to all its inputs

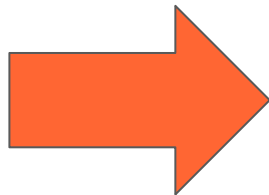
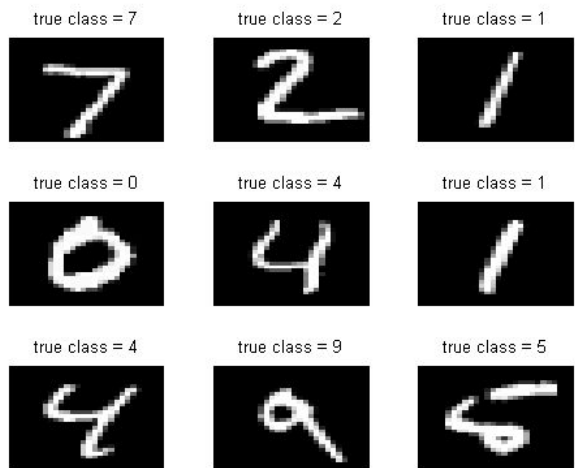
Feasible when dealing with certain data, even small image, but ...

Limitations of MLPs



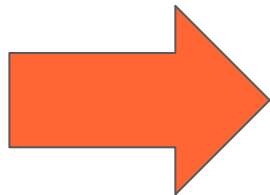
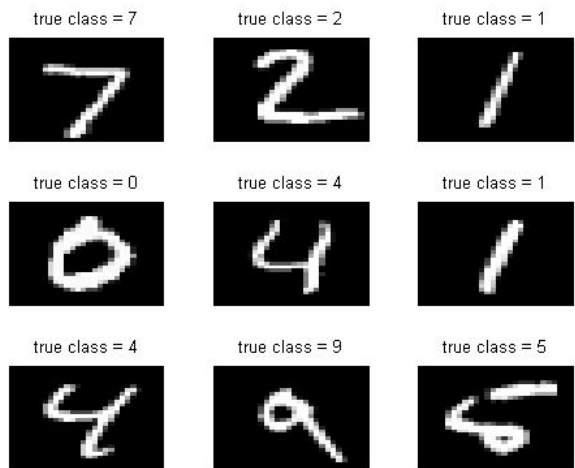
Limitations of MLPs

Can you recognize the digits on the left ? And their transformation with a fixed permutation on the right ? Why ?



Limitations of MLPs

What would happen if we trained an MLP on a version of MNIST dataset in which all pixels moved with a fixed permutation across all images ?



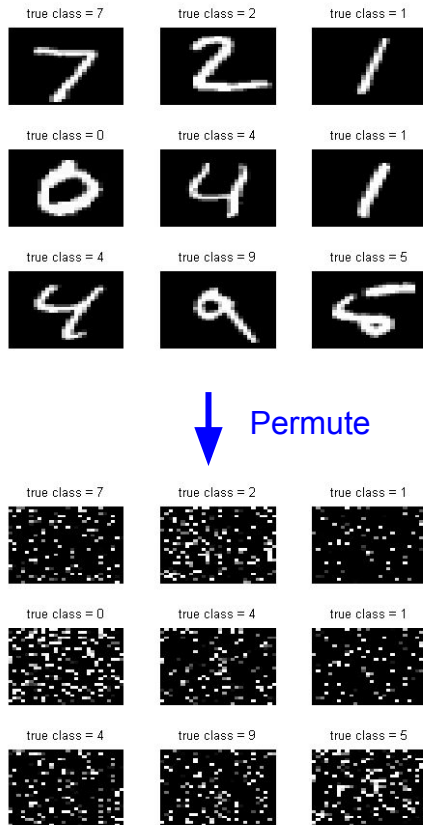
Limitations of MLPs

There is something interesting about solving MNIST classification with an MLP.

- It is possible to apply a fixed permutation to the pixels in the image (shuffle them around)
- This does **NOT** in any way affect the classification accuracy!
- Yet the resulting images are completely unintelligible to humans
- It is difficult to imagine that a human could learn to recognize permuted images of images

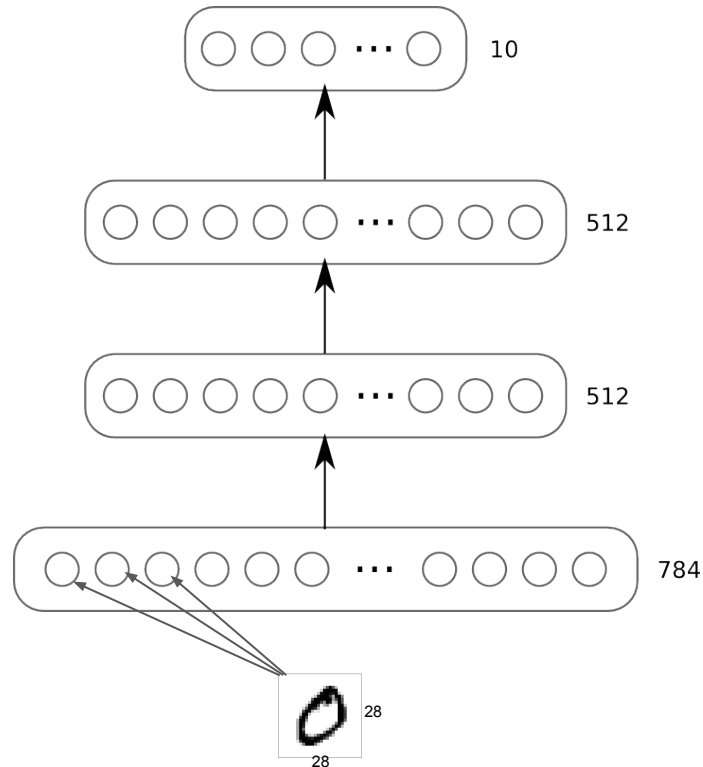
What's going on?

- Fully connected layers assume no spatial neighbourhood relationships
- Maybe we can do better if we somehow embed these structural relationships into the algorithm...



Limitations of MLPs

Layer	#Weights	#Biases	Total
1	784 x 512	512	401,920
2	512 x 512	512	262,656
3	512 x 10	10	5,130
			669,706



Limitations of MLPs

What challenge do you foresee if instead of classifying 28x28 images with an MLP, we try to classify full resolution RGB (3 channels) images ?



Limitations of MLPs

What challenge do you foresee if instead of classifying 28x28 images with an MLP, we try to classify full resolution RGB (3 channels) images ?



In this case learning features on the entire input is **computationally very expensive**. MLP networks are very slow when training them and also when doing inference in these cases.

Outline

1. Limitations of Multi-Layer Perceptrons
- 2. Convolutional Layers**
3. Pooling Layers
4. ConvNets Architectures for Computer Vision
5. Summary

Motivation for ConvNets

ConvNets are similar to ordinary Neural Networks, they are made up of **neurons** that have **learnable weights and biases**.

Each neuron receives some inputs, performs dot product and then there is a non-linearity. The main difference is that with ConvNet architectures there is the assumption that the **information of the input is local (such as in images)** so some properties this characteristic can be encoded in the architecture.

The result is that the **forward is more efficient** to implement, and the **number of parameters is greatly reduced**.

Convolutional neural networks (ConvNets, CNN)

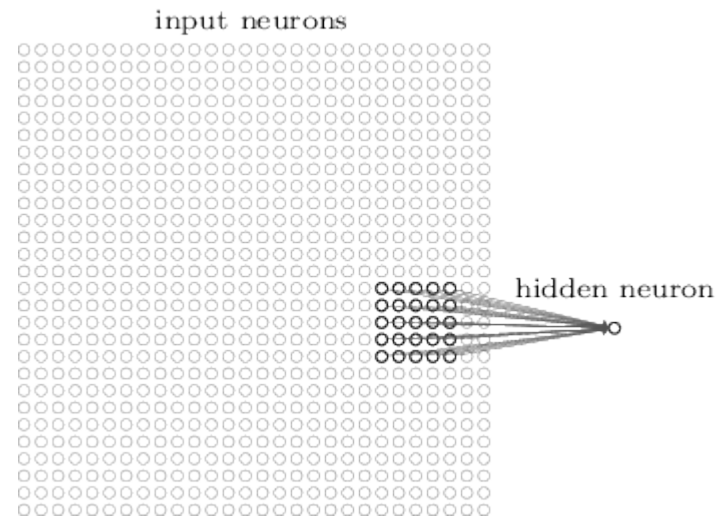
Key idea: good features to learn for images are:

- **Local:** only depend on a small part of the image, not the whole image
- **Translation invariant:** if a feature is good for one part of an image, it should be good for others too.

Instead of a big matrix multiplication on the whole image, apply a whole lot of little matrix multiplications against each image patch and store the local activations.

This is called **convolution**

Parameters are **shared** across these convolutional **kernels** (translation invariance)



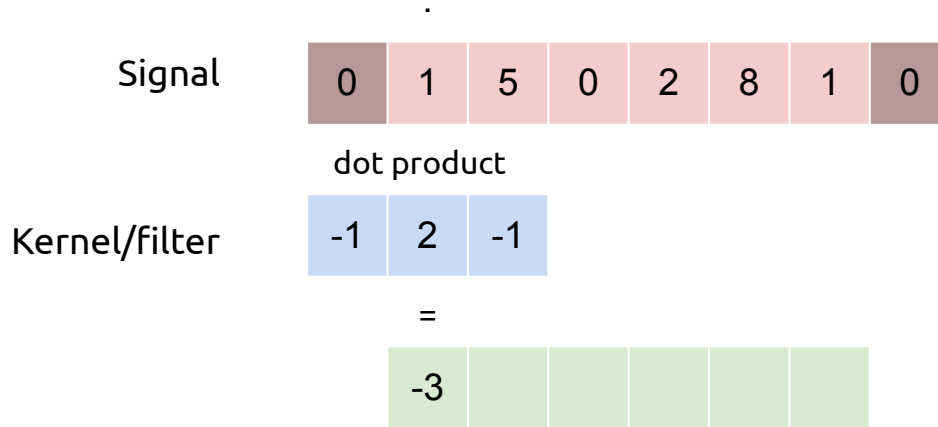
Convolutional neural networks

Is this exclusive to images?

NO!!

Convolutional Neural Networks have been proved to work well for other kinds of signals (**text, speech...**) as they are **computationally very efficient** and can learn **very useful representations!**

1D convolution

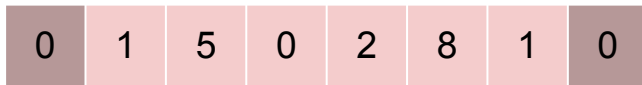


$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x - n_1, y - n_2]$$

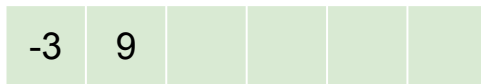
↑
elementwise multiplication and sum of
a filter and the signal (image)

1D convolution

Signal



Kernel/filter

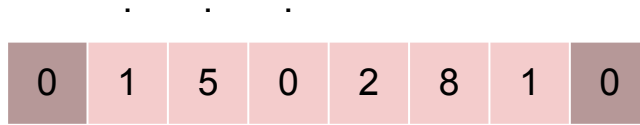


$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1,y-n_2]$$

↑
elementwise multiplication and sum of
a filter and the signal (image)

1D convolution

Signal



Kernel/filter



$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1,y-n_2]$$

↑
elementwise multiplication and sum of
a filter and the signal (image)

1D convolution

Signal

0	1	5	0	2	8	1	0
---	---	---	---	---	---	---	---

Kernel/filter

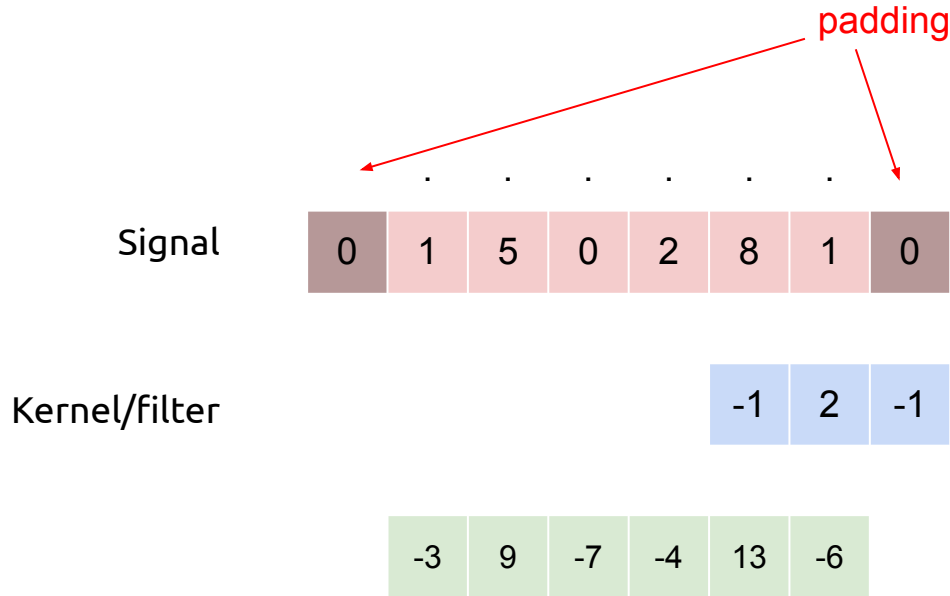
-1	2	-1
----	---	----

-3	9	-7	-4	13	-6
----	---	----	----	----	----

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1,y-n_2]$$

↑
elementwise multiplication and sum of
a filter and the signal (image)

1D convolution: Zero padding



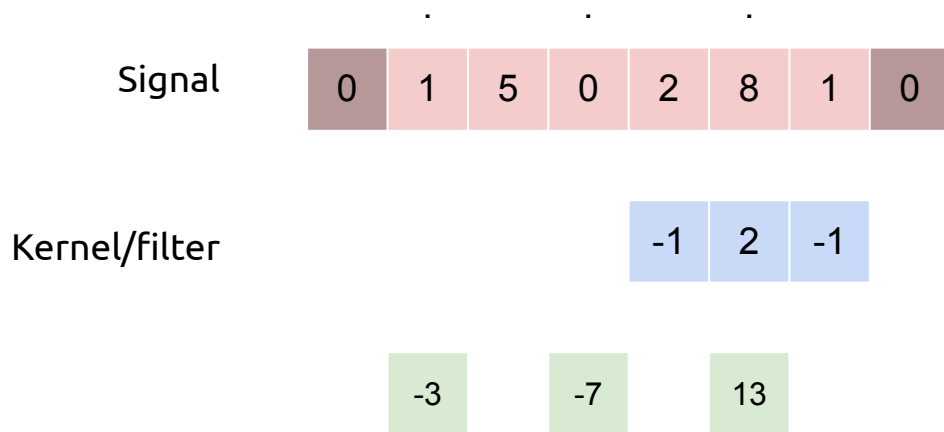
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1,y-n_2]$$

↑
elementwise multiplication and sum of
a filter and the signal (image)

Zero padding=1

Convolved signal has same
dimension as the input signal

1D convolution: Stride



$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1,y-n_2]$$

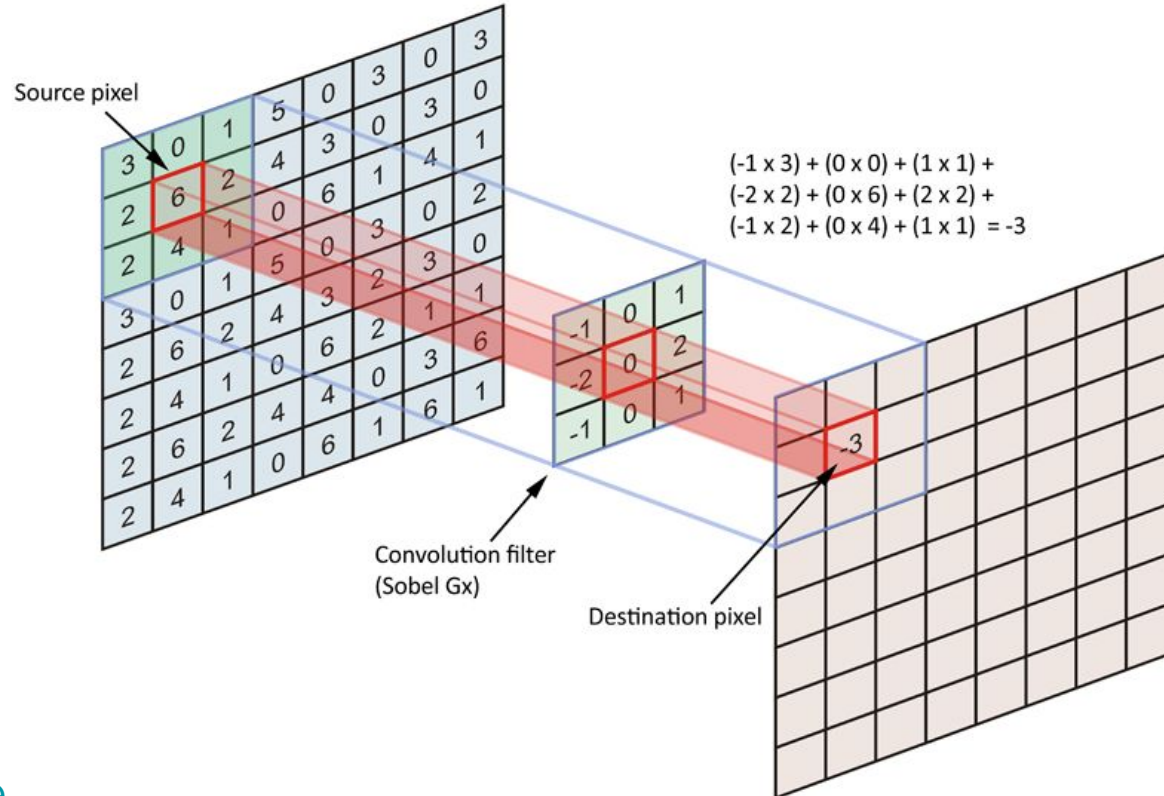
↑
elementwise multiplication and sum of
a filter and the signal (image)

Hyperparameters

Zero padding=1 + Stride=2

Convolved signal has lower
dimension (half) then the input
signal

2D convolution



2D convolution

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

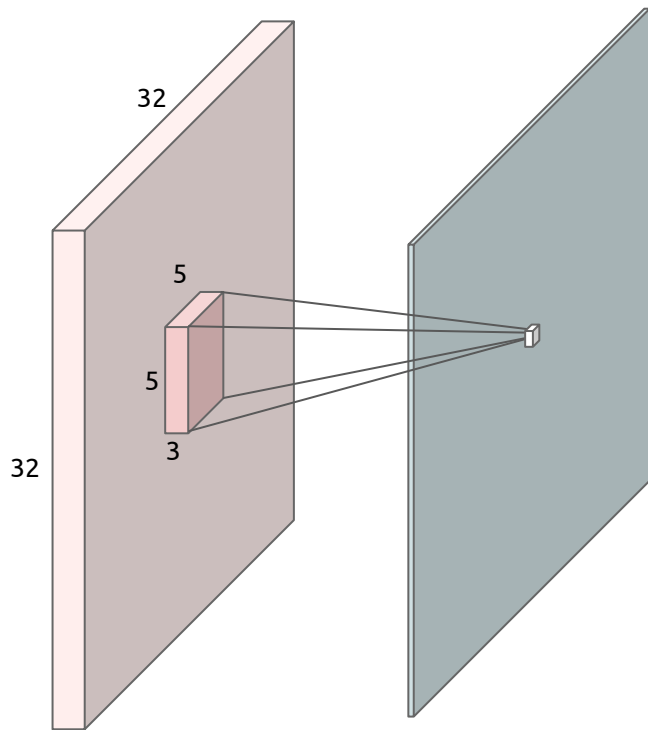
Convolution in Images

A 5x5 convolution on a volume of depth 3 (e.g. an image) needs a filter (kernel) with 5x5x3 elements (weights) + a bias.

Kernels move along 2 dimensions, that's why these are 2D convolutions as well.

Andrej Karpathy's demo:

<http://cs231n.github.io/convolutional-networks/#conv>

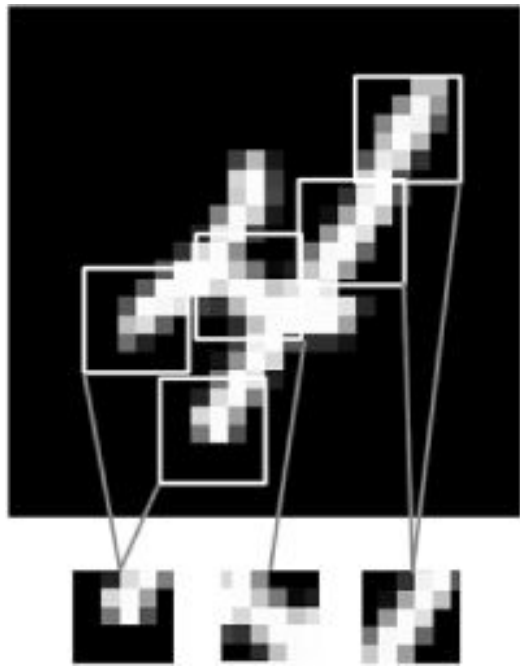


What are these filters learning?

Dense / fully-connected layers would be connected to all the input, and learn global patterns.

Convolutions, on the other hand, learn local patterns, which is very convenient in images.

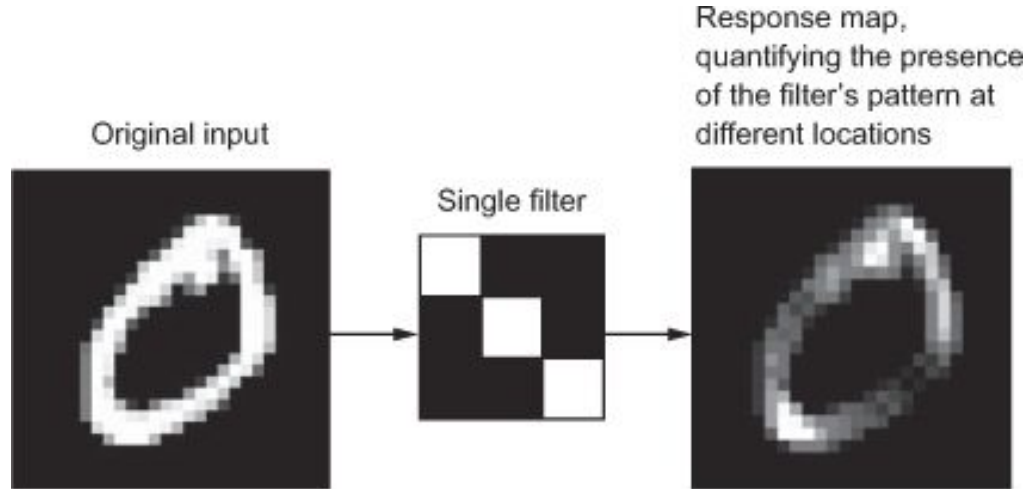
Each of these filters is a **feature detector**, it may learn to detect a horizontal/vertical pattern, a certain texture, or more complex features such as objects parts.



[Figure Credit](#)

What are these filters learning?

For instance, this filter detects diagonal patterns. This concrete filter is a 3x3 filter, typical kernels are 3x3 or 5x5.



What are these filters learning?

Guess what this filter is detecting..

	0	1	0	
	1	-4	1	
	0	1	0	

CLUE!! What happens if the input is the following?

1	1	1
1	1	1
1	1	1

What are these filters learning?

Taking the difference between a pixel and its neighbors detects edges:



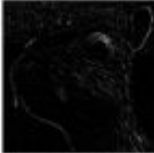

	0	1	0	
	1	-4	1	
	0	1	0	



[Image Credit](#)

What are these filters learning?

More examples

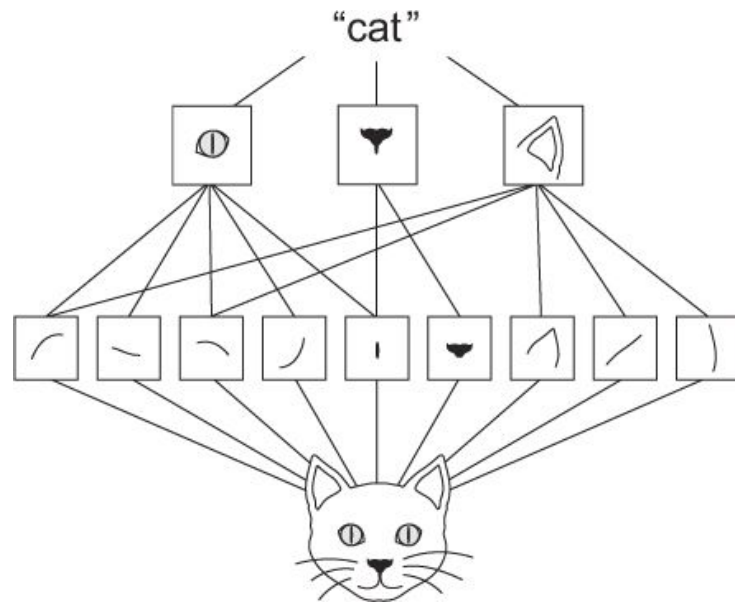
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

[Figure Credit](#)

Why using CNNs for images?

- The patterns they learn are **translation invariant**:
 - Each filter will learn to detect a certain pattern, that can be recognized in any location of the image. Then these filters are translation invariant. This is data efficient, and with less training samples CNNs can learn representations that can generalize.
- They can learn **hierarchies of patterns**.
 - A first convolutional layer will learn small local patterns, and the following convolutional layers will learn larger patterns made of features of the first layers, and so on. Then convnets can learn complex and abstract visual concepts.



Example reduction parameters

Fully connected network:

1000x1000 image

1M hidden units

10^{12} parameters!

- Spatial correlation is local
- Better to put resources elsewhere!

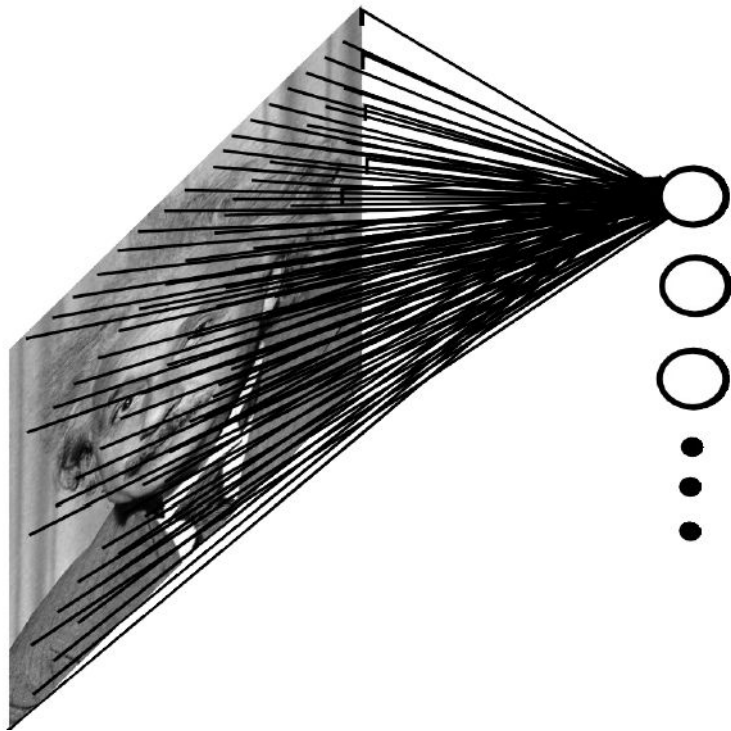


Figure Credit: Ranzatto

Example reduction parameters

The idea is to learn **multiple filters**. Each of these learned filters will detect a different ‘feature’.

1000x1000 image

100 Filters

Filter size: 10x10

How many convolutional weights must be learned?

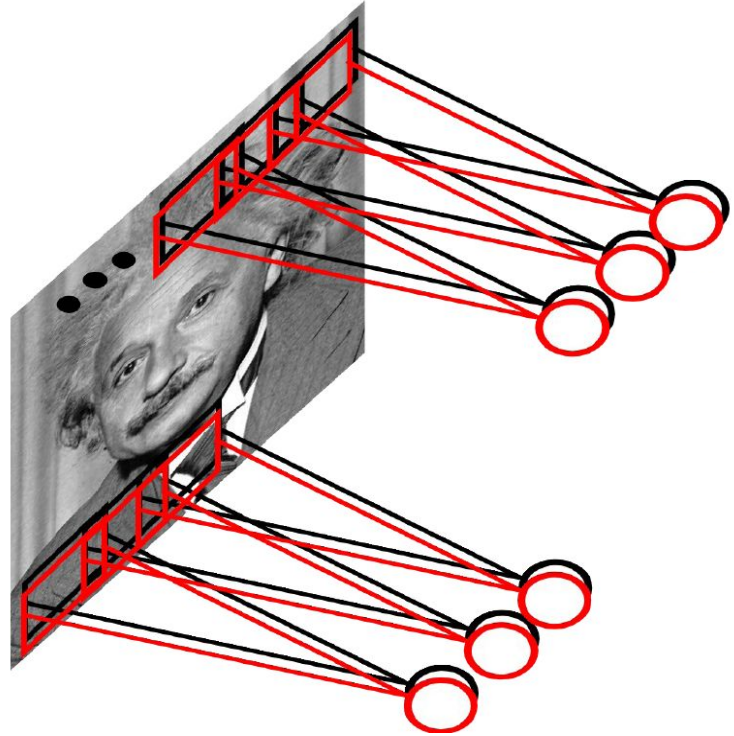


Figure Credit: Ranzatto

Example reduction parameters

The idea is to learn **multiple filters**. Each of these learned filters will detect a different ‘feature’.

1000x1000 image

100 Filters

Filter size: 10x10

10K parameters

How does the size of the input image affect the amount of parameters ?

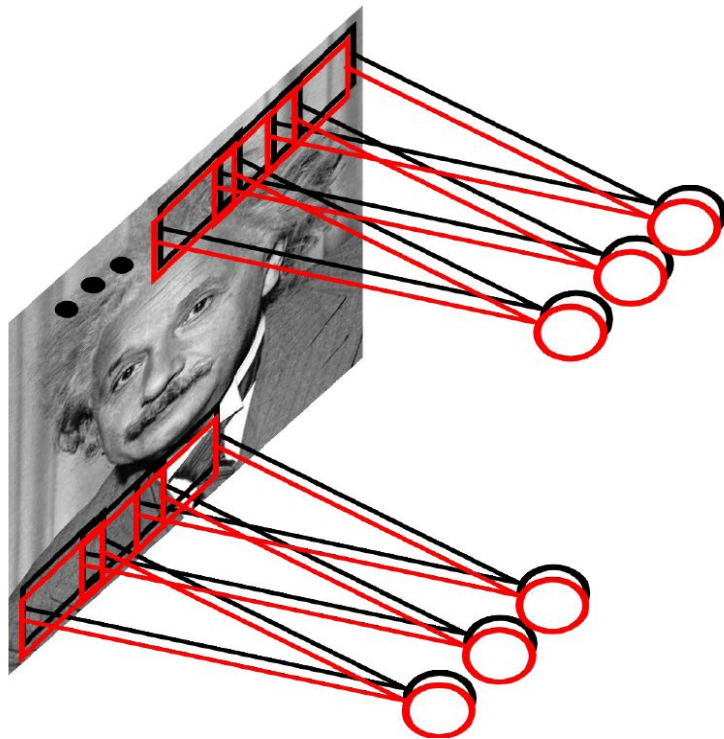


Figure Credit: Ranzatto

Example reduction parameters

The idea is to learn **multiple filters**. Each of these learned filters will detect a different ‘feature’.

1000x1000 image

100 Filters

Filter size: 10x10

10K (weights) + 100 (biases) parameters

parameters size does not depend on input
image size!

Finally, 10.1K vs 10^{12} parameters

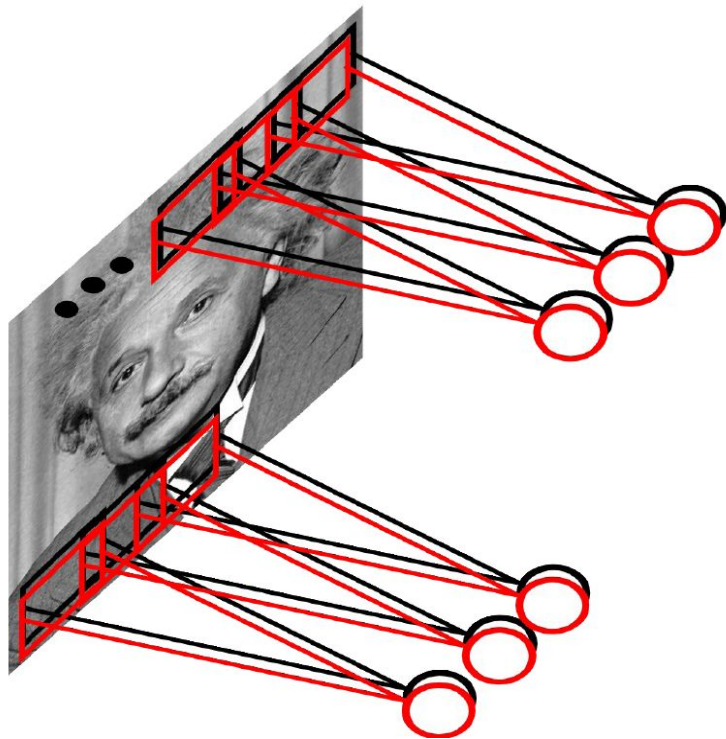


Figure Credit: Ranzatto

Feature Maps

We have said that convnets consists on **learning filters that are feature detectors**. The key is to learn several of these features. The responses at different locations from these filters create what's called **feature maps**.

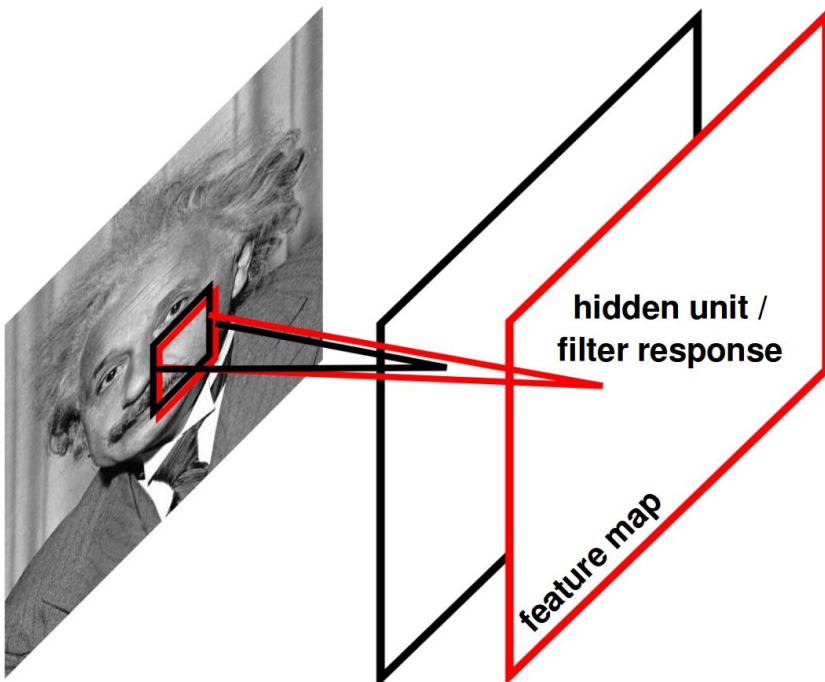


Figure Credit: Ranzatto

Convolutional Layer

Finally you can think of a convolutional layer as a module that transforms some feature maps to other feature maps, which learn higher-abstract concepts

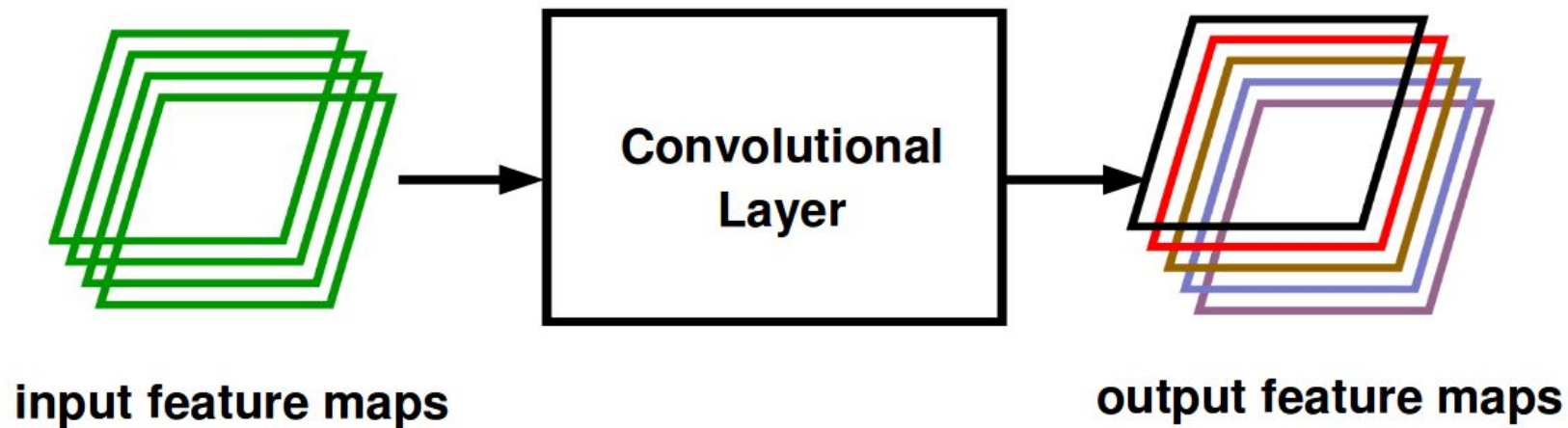
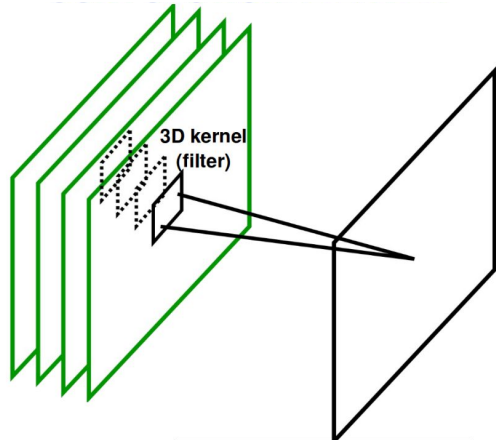


Figure Credit: Ranzatto

Feature Maps

And these are feature maps when the input is 3D



Many feature maps
→

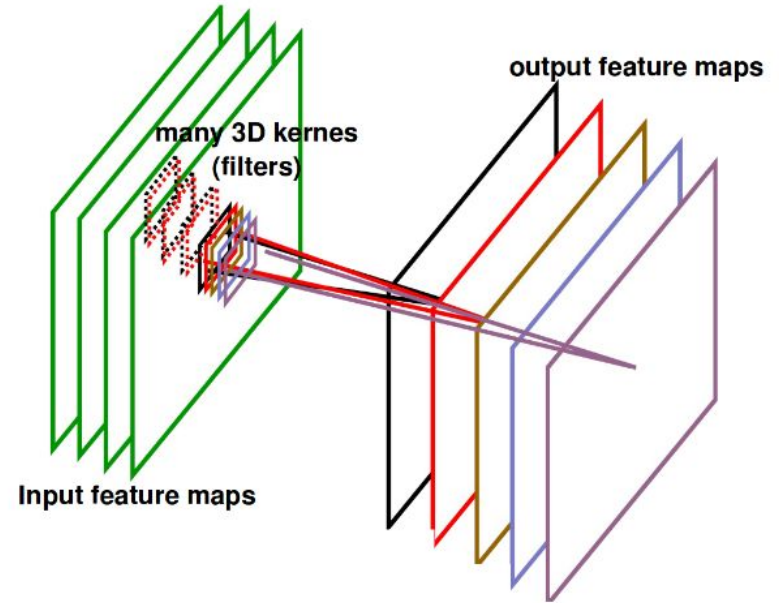


Figure Credit: Ranzatto

Example of ConvNet on an Image

Convolutional Layer



[Figures Credit, NYU course](#)

Outline

1. Limitations of Multi-Layer Perceptrons
2. Convolutional Layers
- 3. Pooling Layers**
4. ConvNets Architectures for Computer Vision
5. Summary

Pooling Layer

So a pooling layer is a module that reduces spatially the dimensions of some input feature maps.

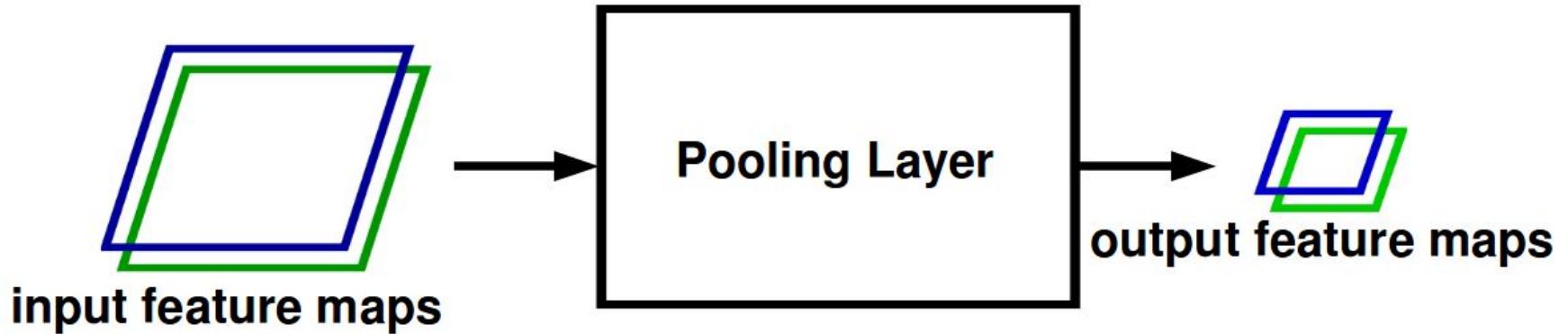
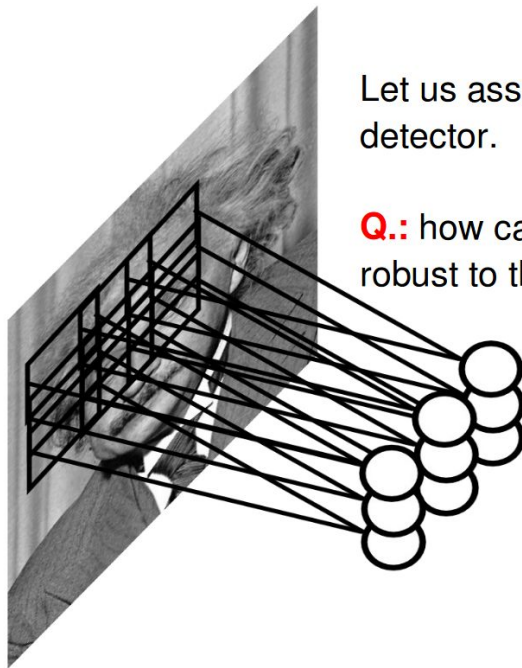


Figure Credit: Ranzatto

Pooling Layer

Pooling is a downsample operation along the spatial dimensions (width, height)

- It **reduces** progressively the **spatial size of the representation**, so it reduces the computation greatly.
- Provides **invariance to small local changes**



Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?

Figure Credit: Ranzatto

Pooling Layer

Pooling is a downsample operation along the spatial dimensions (width, height)

- It **reduces** progressively the **spatial size of the representation**, so it reduces the computation greatly.
- Provides **invariance to small local changes**

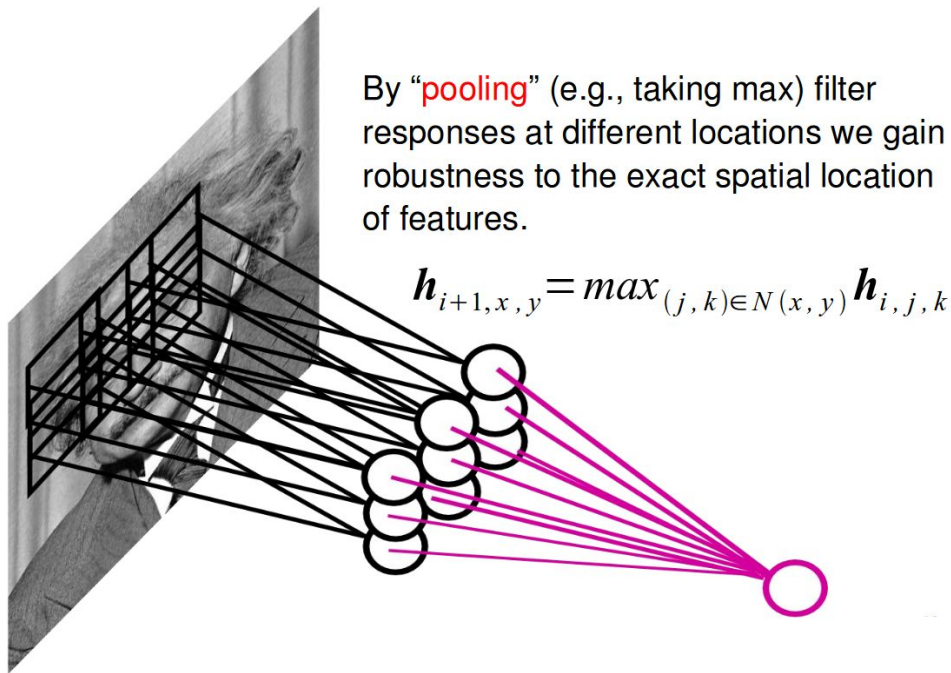
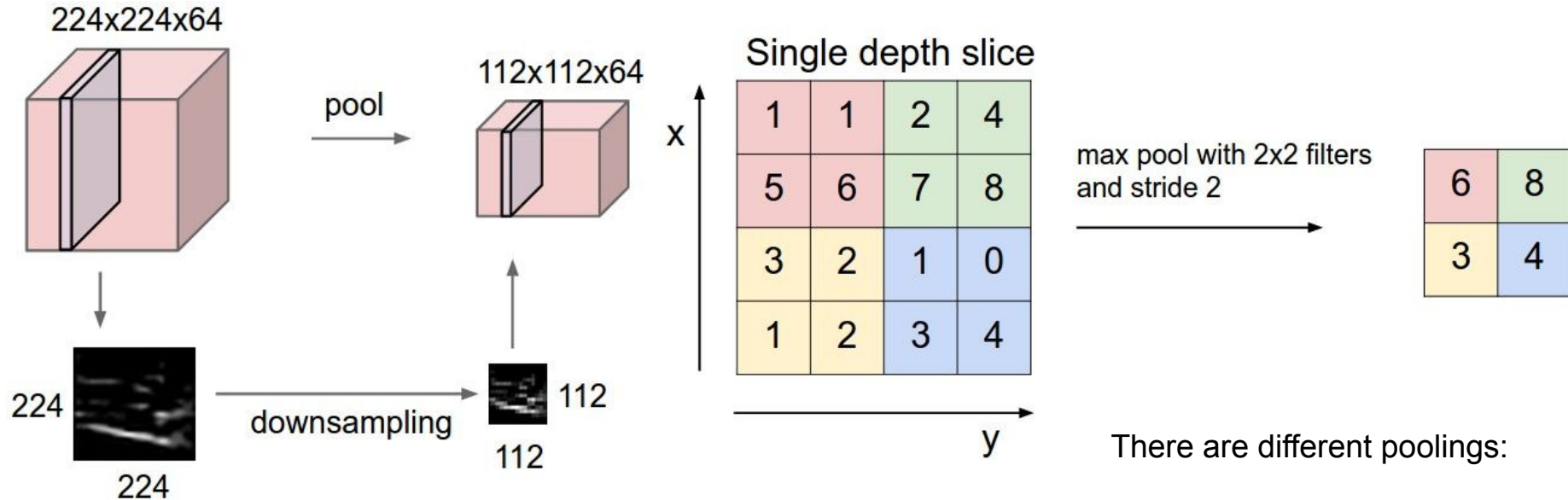


Figure Credit: Ranzatto

Pooling Layer

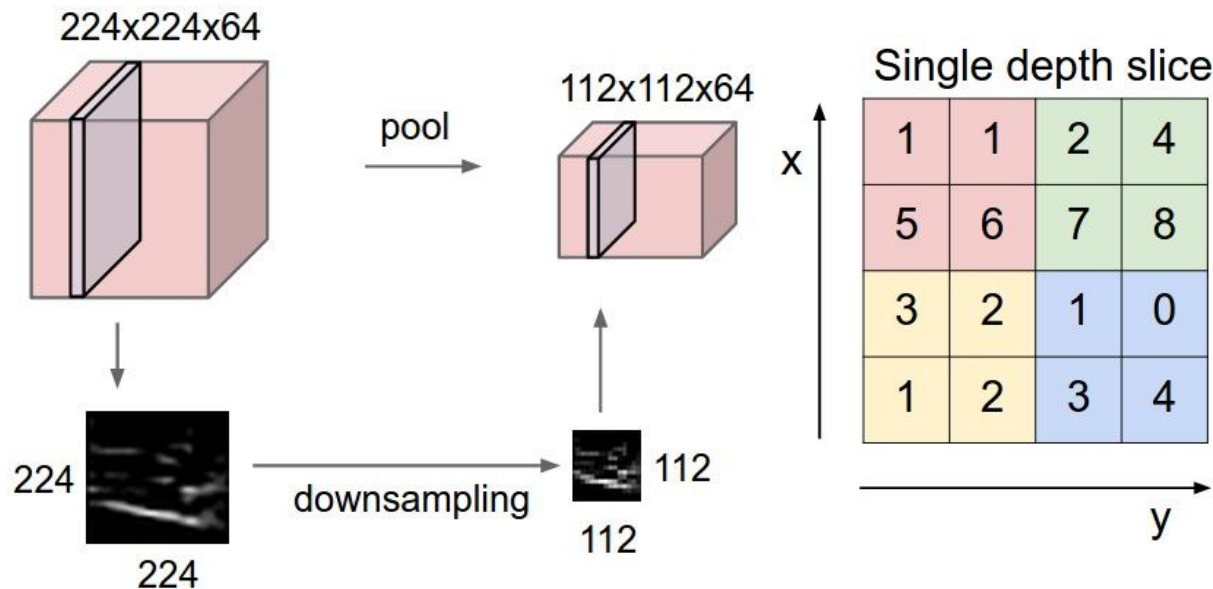


[Figure Credit, CS231n Course](#)

There are different poolings:

- **Max Pooling**
- Average Pooling
- Stochastic Pooling

Pooling Layer



now, a filter of 2x2 will see a bigger patch of the input image! Receptive field will be bigger, and learn more abstract concepts!

max pool with 2x2 filters and stride 2

6	8
3	4

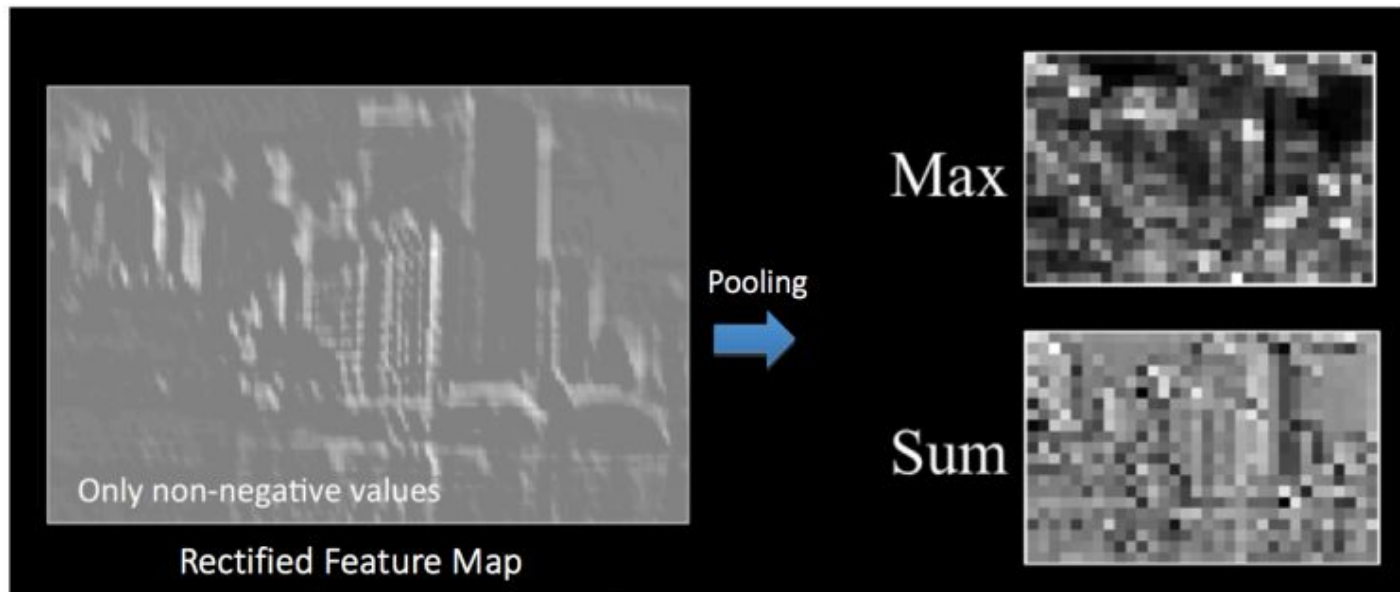
There are different poolings:

- **Max Pooling**
- Average Pooling
- Stochastic Pooling

[Figure Credit, CS231n Course](#)

Example of ConvNet on an Image

Pooling Layer

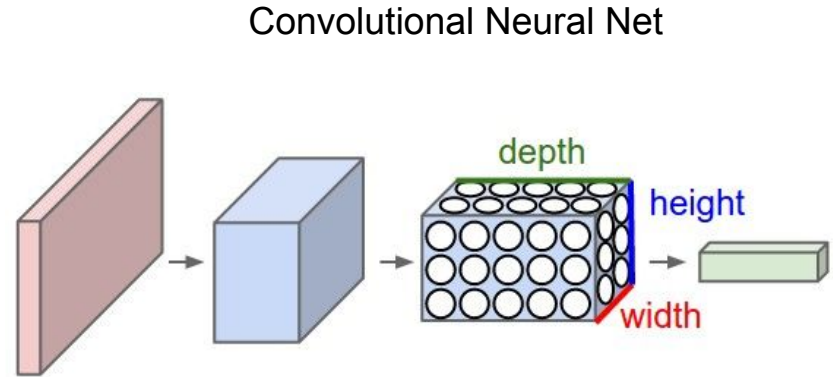
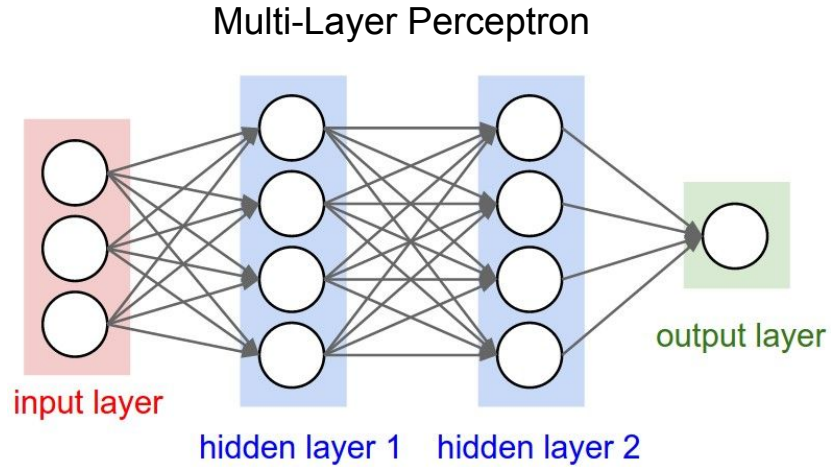


[Figures Credit, NYU course](#)

Outline

1. Limitations of Multi-Layer Perceptrons
2. Convolutional Layers
3. Pooling Layers
- 4. ConvNets Architectures for Computer Vision**
5. Summary

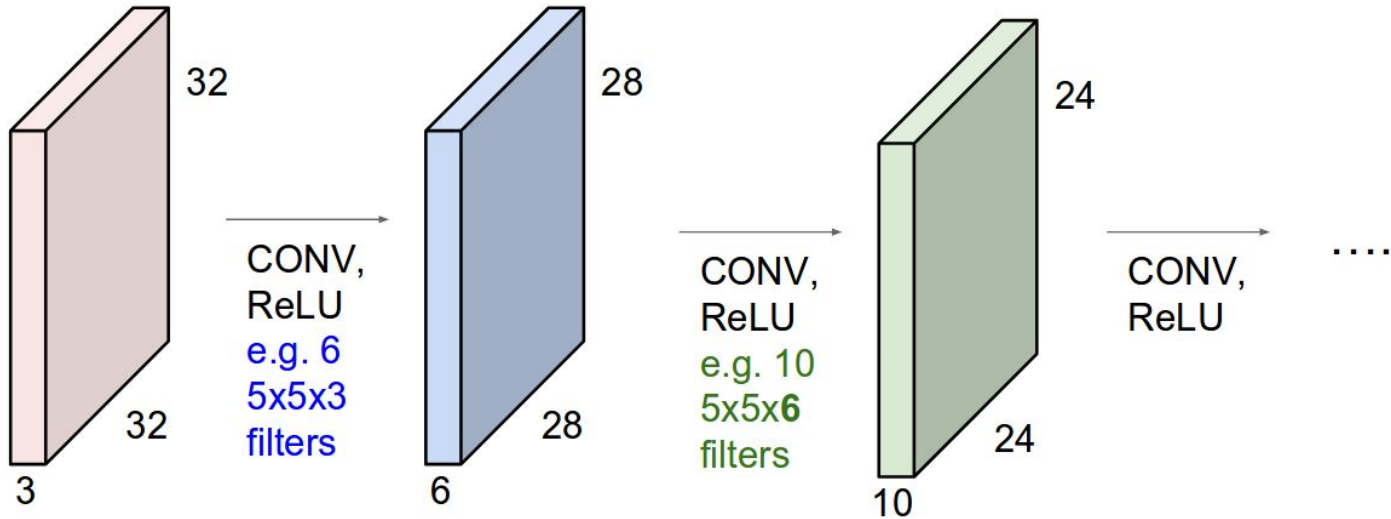
ConvNets Architecture



[Figures Credit, CS231 Course](#)

ConvNets Architecture

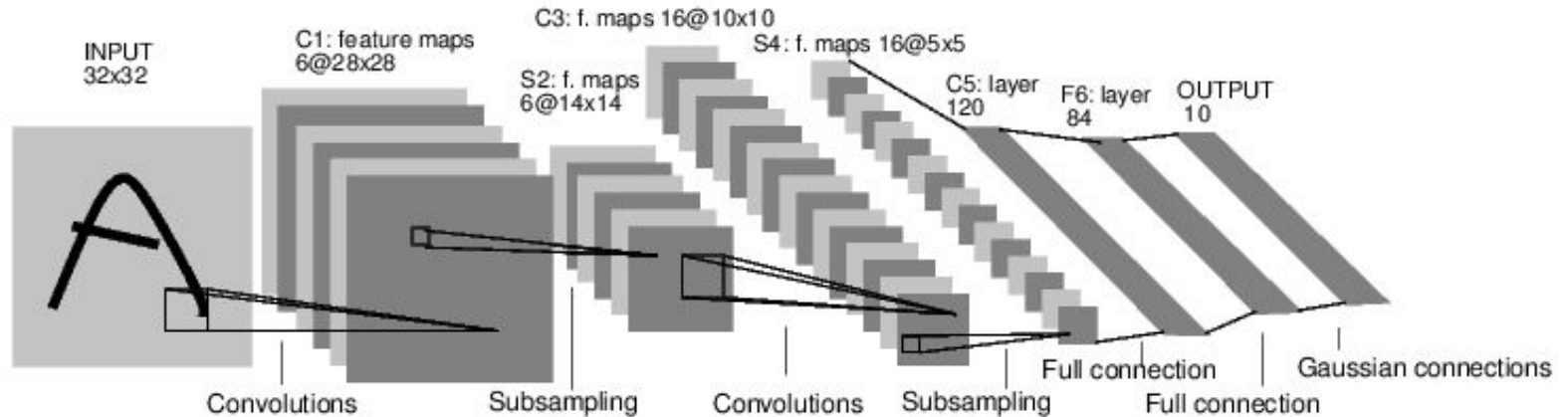
A ConvNet is a sequence of Convolution Layers, interspersed with activation functions



[Figure Credit, CS231n Course](#)

ConvNets Architecture

LeNet-5: The most typical architecture consists on several convolutional layers, interspersed with pooling layers, and followed by a small number of fully connected layers



LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). [Gradient-based learning applied to document recognition](#). *Proceedings of the IEEE*, 86(11), 2278-2324.

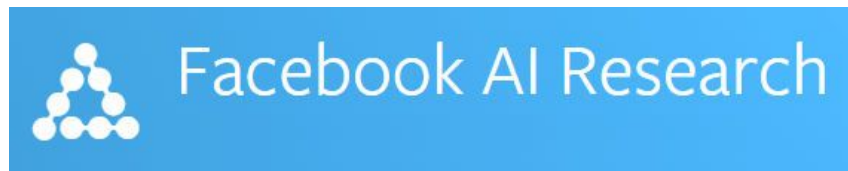
Convolutional Neural Network (CNN)



Yann LeCun
t'ha retuitat el tuit.

Mostra-ho

Yann LeCun



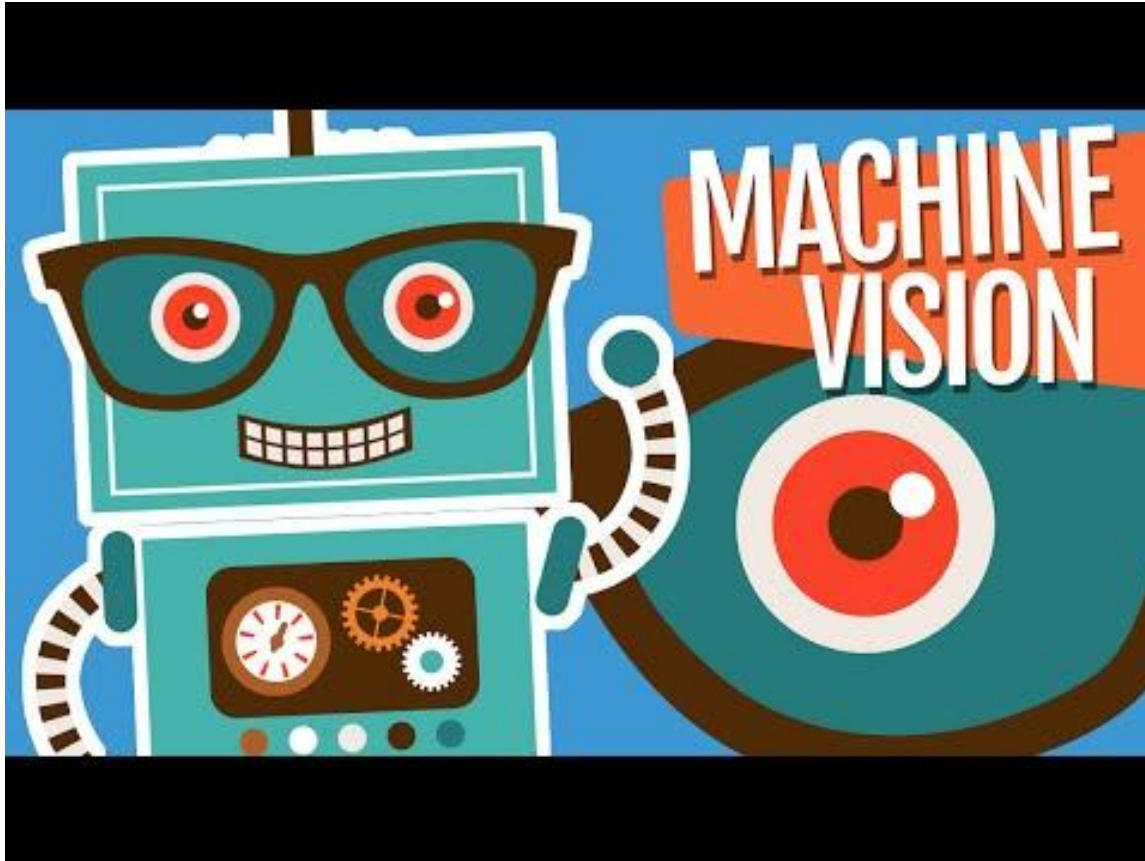
Xavier Giró-i-Nieto @DocXavi

Thank you very much (moltes gràcies) to
[@ylecun](#) for his kindness with our team.
Looking forward to seeing you in
Barcelona soon ! [#nips2016](#)



You can also
check [@boredyannlecun](#)
on Twitter...

Convolutional Neural Network (CNN)



Outline

1. Limitations of Multi-Layer Perceptrons
2. Convolutional Layers
3. Pooling Layers
4. ConvNets Architectures for Computer Vision
- 5. Summary**

Homework

A Convolutional layer of a CNN has an RGB input of 64×64 pixels and it's made of 8 filters of size 5×5 , a stride of 2 and zero padding.

- a) What is the number of parameters of this layer ?
- b) What are the dimensions of the resulting feature maps ?

Summary

- Significantly less parameters to learn:
 - Small local kernels
 - Shared parameters
- Faster training
 - Weight sharing means gradients are averaged for every location of the kernel
- Local features
 - Can be used to detect object location for images
- Hierarchy of Features
 - By combining several levels of abstraction convnets can learn very complex representations
- Interpretability
 - It is possible to visualize the learned representations

Undergradese

What undergrads ask vs. what they're REALLY asking

"Is it going to be an open book exam?"

Translation: "I don't have to actually memorize anything, do I?"

"Hmm, what do you mean by that?"

Translation: "What's the answer so we can all go home."

"Are you going to have office hours today?"

Translation: "Can I do my homework in your office?"

"Can i get an extension?"

Translation: "Can you re-arrange your life around mine?"

"Is this going to be on the test?"

Translation: "Tell us what's going to be on the test."

"Is grading going to be curved?"

Translation: "Can I do a mediocre job and still get an A?"

JORGE CHAM © 2008



WWW.PHDCOMICS.COM