CS396: Selected CS2
(Deep Learning for visual recognition)

Spring 2022

Dr. Wessam EL-Behaidy
Associate Professor, Computer Science Department,
Faculty of Computers and Artificial Intelligence,
Helwan University.

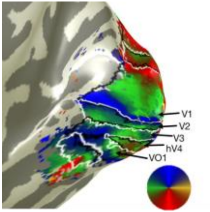
Lecture 3: Convolution Neural Network (CNN or ConvNets)

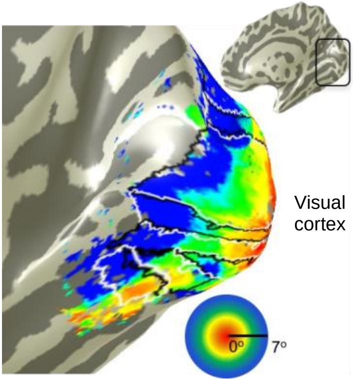
Convolutional Neural Networks

Convolutional neural networks (**ConvNets** or **CNNs**) is one of the main categories to do images recognition, images classifications. Objects detections, recognition faces etc., are some of the areas where CNNs are widely used.

Human brain

Topographical mapping in the cortex: nearby cells in cortex represent nearby regions in the visual field





Visual cortex

Retinotopy images courtesy of Jesse Gomez in the Stanford Vision & Perception Neuroscience Lab.

11

Hierarchical organization

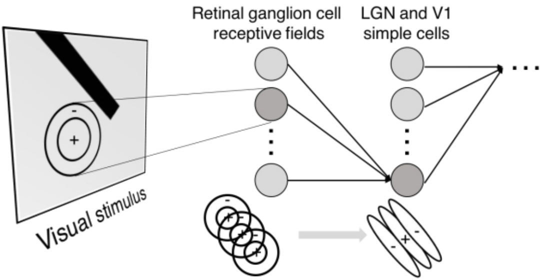


Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

Simple cells:
Response to light orientation

Complex cells:
Response to light orientation and movement

Hypercomplex cells:
response to movement with an end point

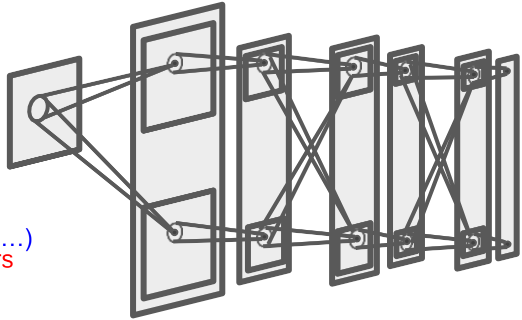
No response

Response (end point)

A bit of history:

Neocognitron

[Fukushima 1980]

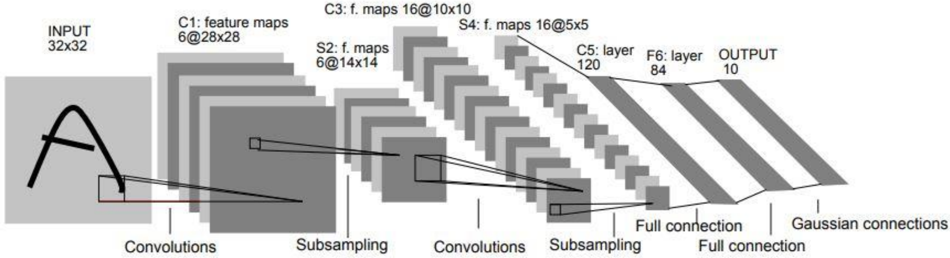


"sandwich" architecture (SCSCSC…)
simple cells: modifiable parameters
complex cells: perform pooling

For more information, you can see:
https://www.youtube.com/watch?v=Qil4kmvm2Sw

LeNet-5 – A Classic CNN Architecture

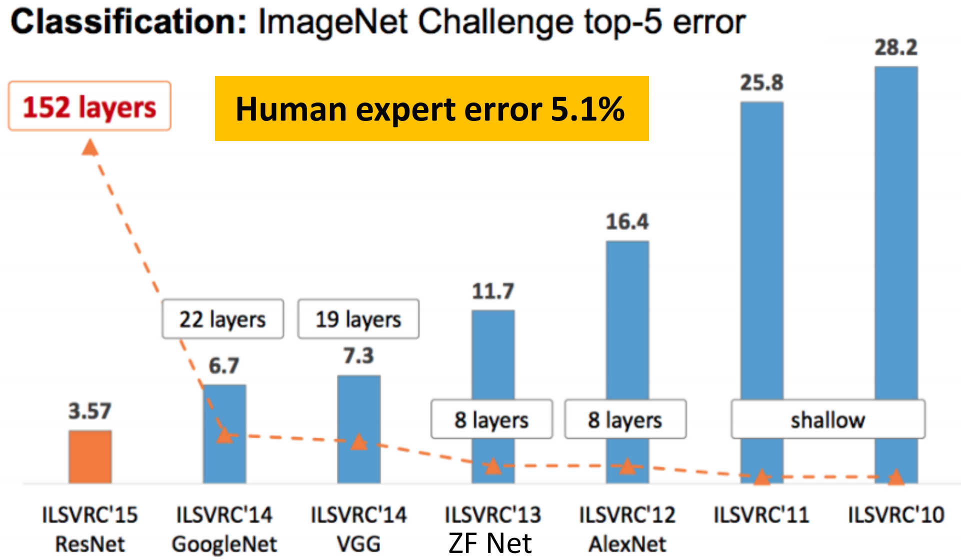
Yann LeCun et al., proposed a neural network architecture for handwritten and machine-printed character recognition in 1990's which they called **LeNet-5**. The architecture is straightforward and simple to understand that's why it is mostly used as a first step for teaching Convolutional Neural Network.



ImageNet & ILSVRC

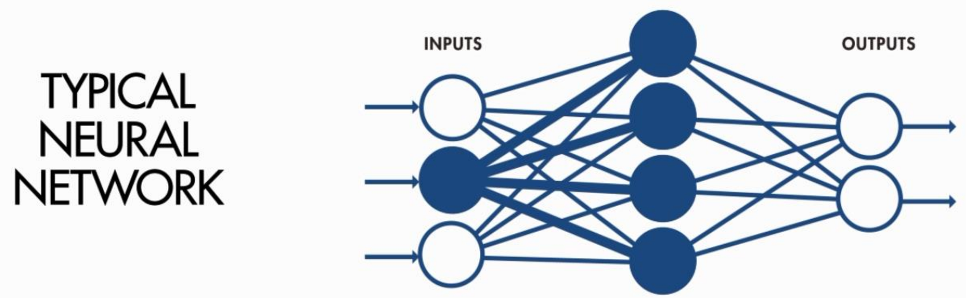
- ~14 million labeled images, 20k classes
- Images gathered from Internet
- Human labels via Amazon MTurk
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC): 1.2 million training images, 1000 classes

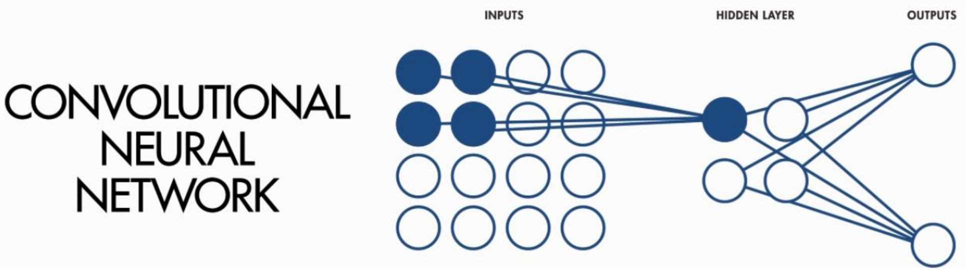
CNN architectures won in ILSVRC



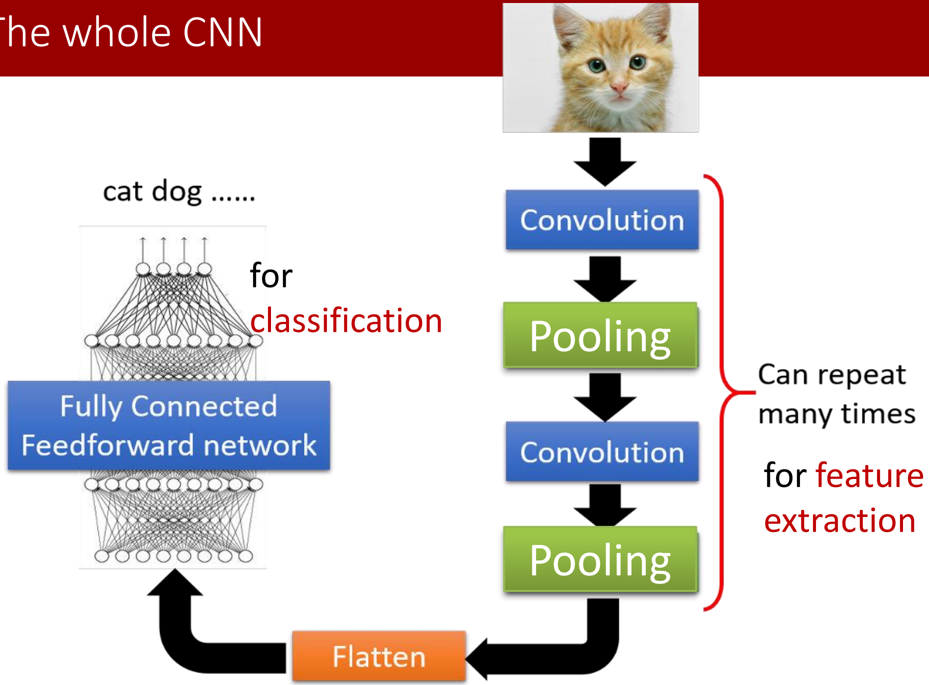
Refer to [1] for more details (The-9-Deep-Learning-Papers-You-Need-To-Know-About)

Typical NN & Convolutional NN



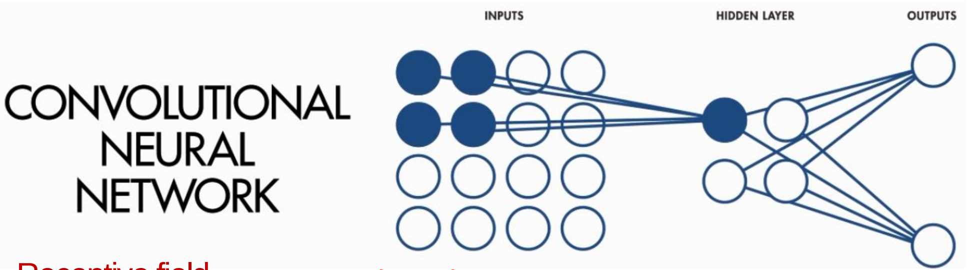


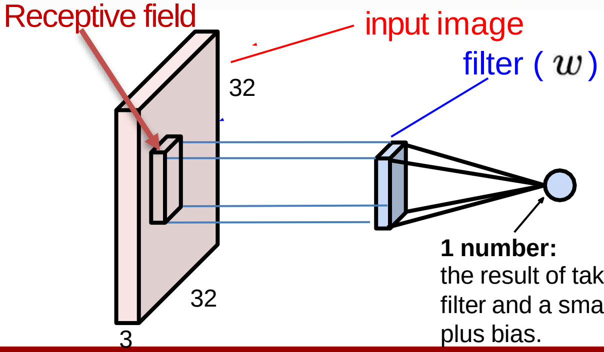
The whole CNN



Convolutional Layer
(Conv layer)

Convolution Layer: Terminology

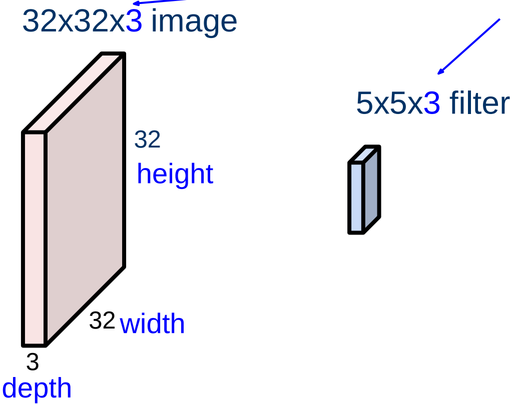




1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image plus bias.

Convolution Layer: Filter depth

Filters always extend the full depth of the input volume

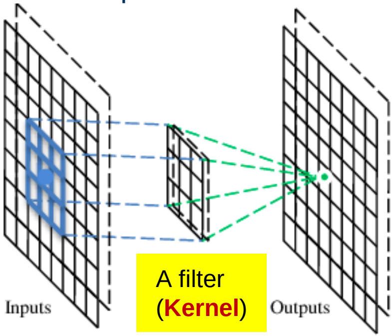


Convolutional Operation

A convolutional layer has a number of filters that does **convolutional operation** of two signals:

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

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



Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

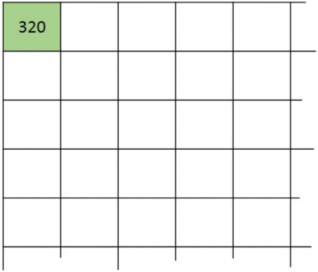
Convolution Operation: Example 1

0	0	0	0	0	0
0	105	102	100	97	96
0	103	99	103	101	102
0	101	98	104	102	100
0	99	101	106	104	99
0	104	104	104	100	98

Input Image

Filter(**Kernel**)

0	-1	0
-1	5	-1
0	-1	0



Activation map (Feature map)

$$\begin{aligned} &0*0+0*-1+0*0 \\ &+0*-1+105*5+102*-1 \\ &+0*0+103*-1+99*0 \quad =320 \end{aligned}$$

Convolution with horizontal and vertical strides = 1

Convolution Operation: Example 2

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

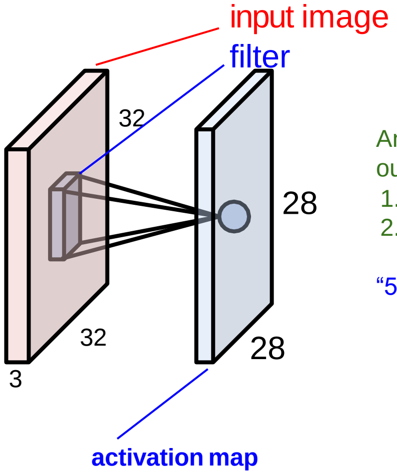
$$308 + (-498) + 164 + 1 = -25$$

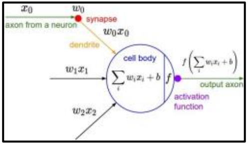
Bias = 1

Output

-25				...
				...
				...
				...
...	...	...	...	...

Convolution Operation: Activation Map



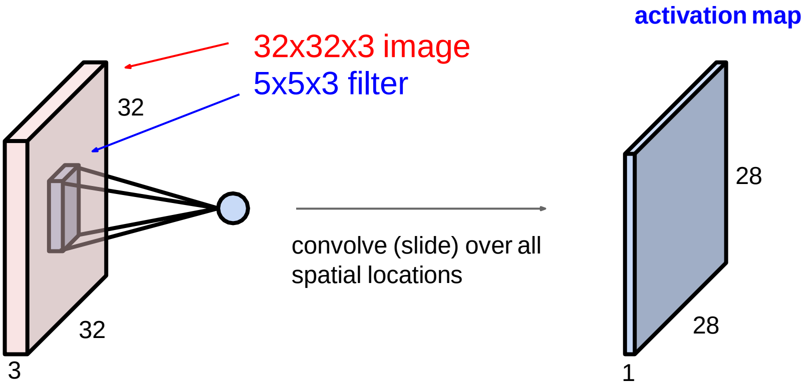


An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

Convolution Layer: Filters



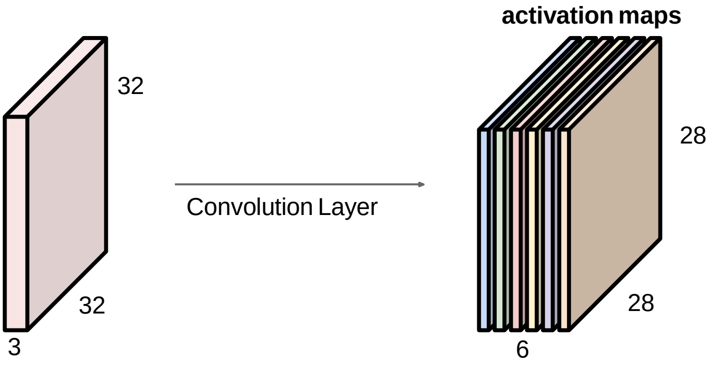
convolve (slide) over all spatial locations

Convolution Layer: Filters

consider a second, green filter

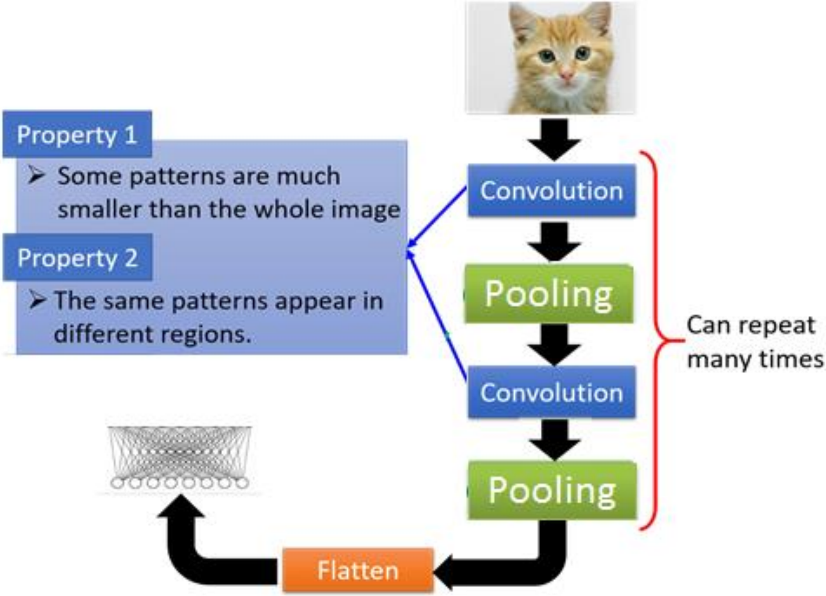
Convolution Layer: Filters

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

Convolution Layer Properties



Two Filters Example:
Property 1

These are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1
Matrix

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2
Matrix

⋮ ⋮

Property 1

Each filter detects a small pattern (3 x 3).

Two Filters Example:

Stride=1

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Dot product

3

-1

Two Filters Example:
Stride=2

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3

-3

Two Filters Example:

Property 2

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Property 2

Two Filters Example: Feature map

Feature map=activation map

-1	1	-1
-1	1	-1
-1	1	-1

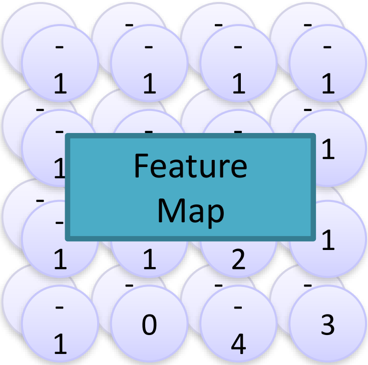
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

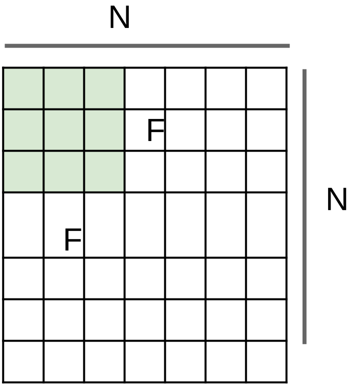
6 x 6 image

Repeat this for each filter



Feature map= 4 x 4 x 2

Feature Map Output Size



Feature map output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 **:**

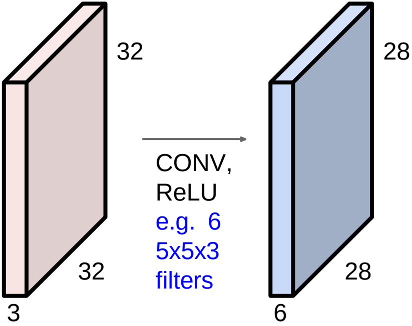
doesn't fit!

cannot apply 3x3 filter on 7x7

input with stride 3.

Remember back to…

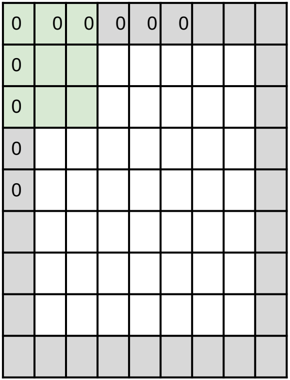
E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially! (32 -> 28 ...). Shrinking too fast is not good, doesn't work well.



(recall:)

(N - F) / stride + 1

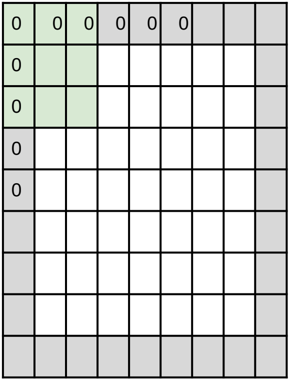
In practice: Common to zero padding the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

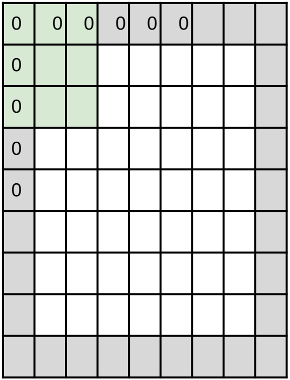
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

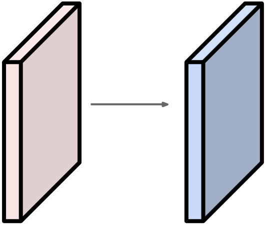
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

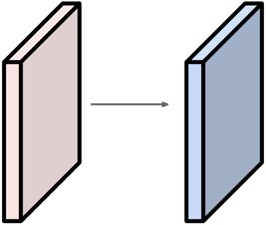
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

Examples time:



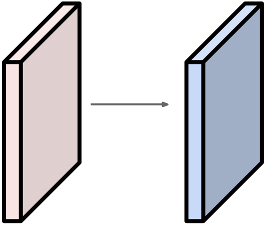
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so

32x32x10

Volume Size:
(N + 2P – F) / stride + 1

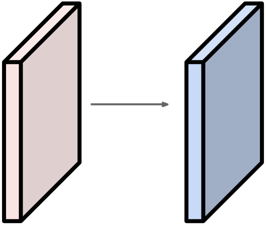
Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params (+1 for bias)
=> 76*10 = **760**

Convolution Layer Summary

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

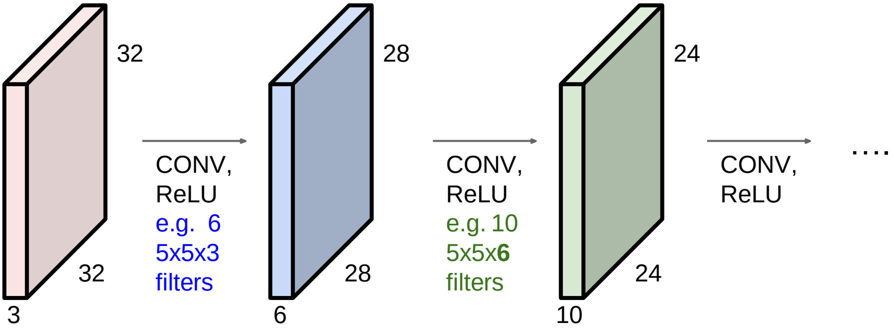
Common settings:

K = (powers of 2, e.g. 32, 64, 128, 512)

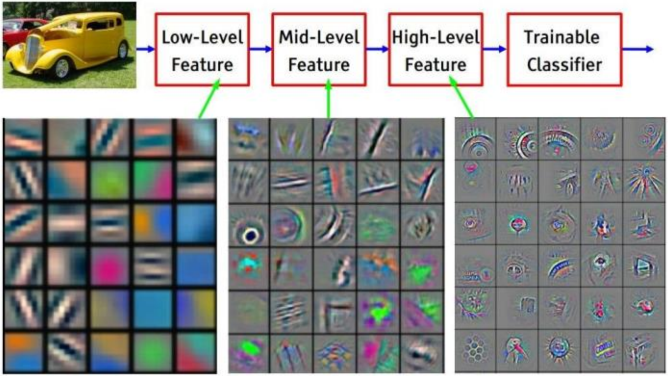
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 1, S = 1, P = 0

Sequence of Convolution Layers

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



Preview



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Pooling Layer
(Pool layer)

Pooling Layer Property

Property 1

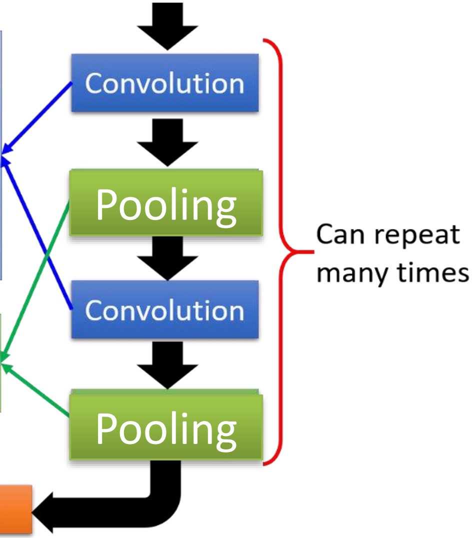
- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object



Flatten

Why Pooling?

Subsampling pixels will not change the object

bird

Subsampling

bird

We can subsample the pixels to make image smaller

fewer parameters to characterize the image

Max Pooling

- operates over each activation map independently
- max pool with 2x2 filters and stride 2

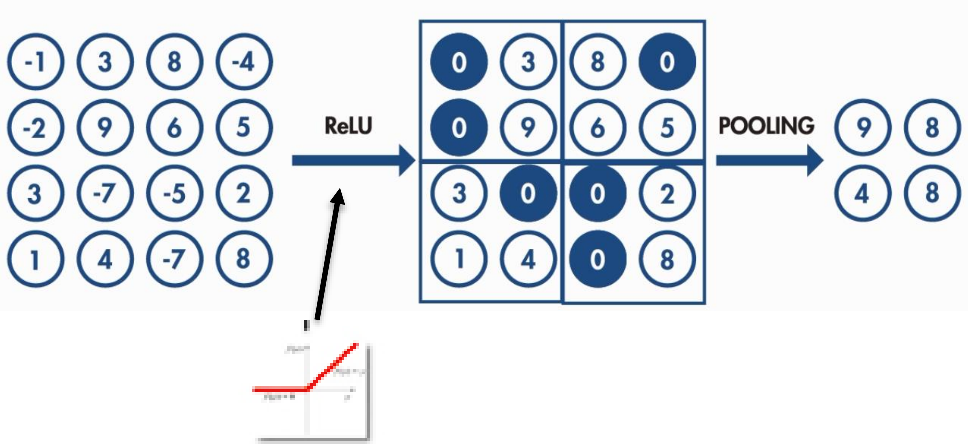
Ex: Feature map 4x4x2

3	0	0	0
0	1	0	0
0	0	0	4
3	0	0	0

0	0	0	0
0	0	0	1
0	0	0	1
0	0	0	3

Note:

Feature Map



Pooling Layer Summary

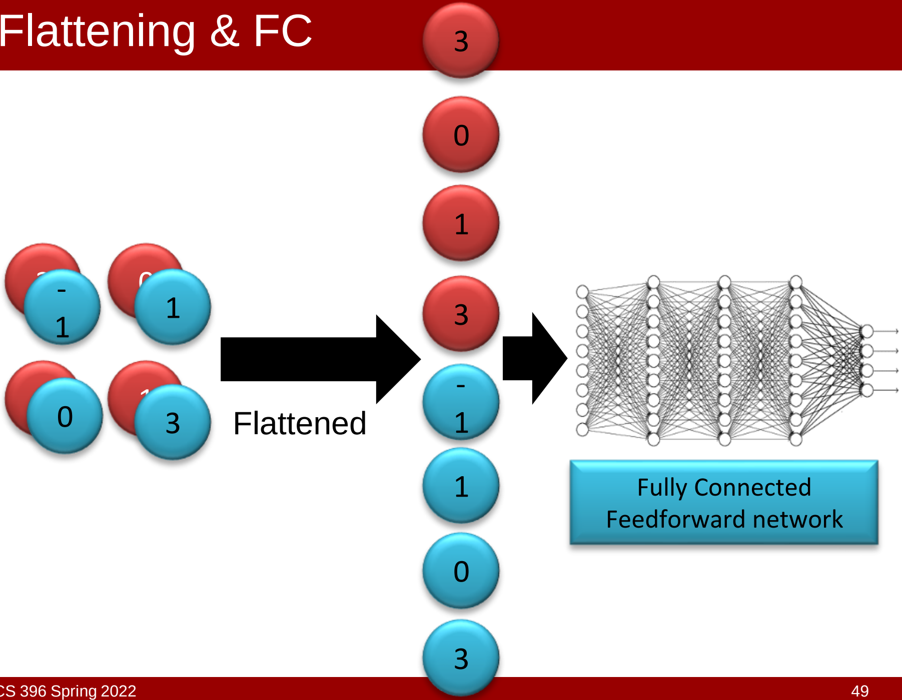
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires tvvo hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Common settings:

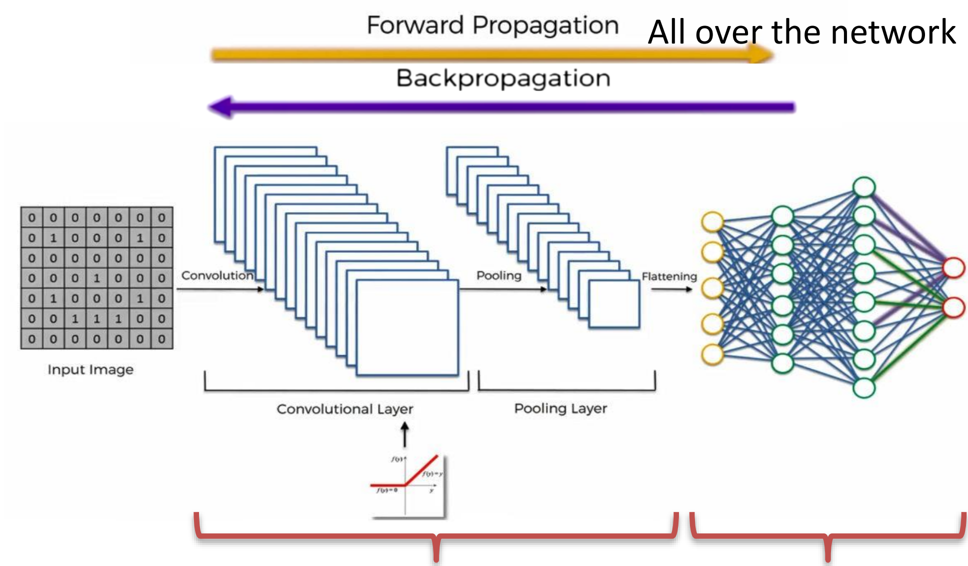
F = 2, S = 2
F = 3, S = 2

Flattening Layer
&
Fully connected network
(FC)

Flattening & FC



CNN: Extraction & Classification



Feature extraction

Classification

Example

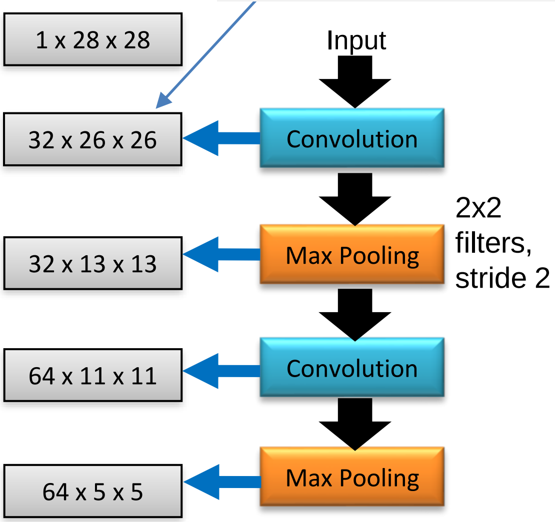
Recall:) Volume Size
(N + 2P – F) / stride + 1

How many parameters for each filter, if we use 32 3x3 filter, with stride 1, pad 0 ?

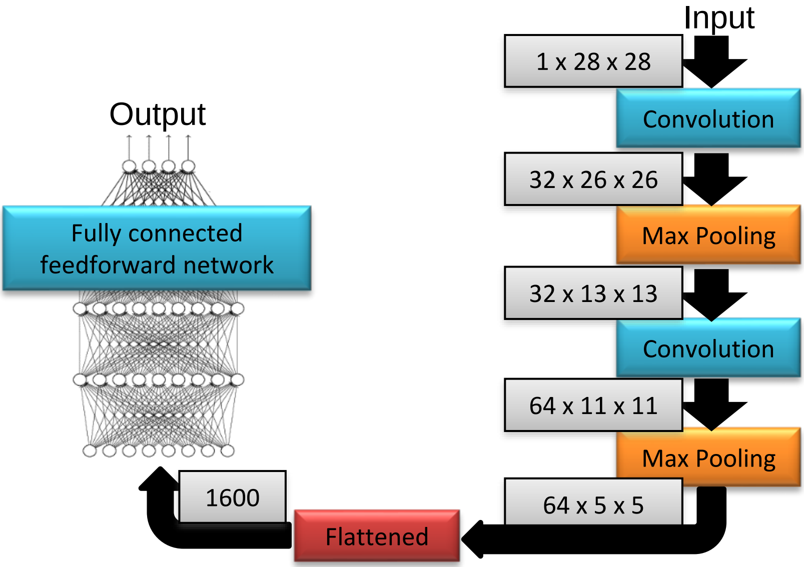
3x3x1 +1=10

How many parameters for each filter, if we use 64 3x3 filter, with stride 1, pad 0 ?

3 x 3 x 32 +1= 289

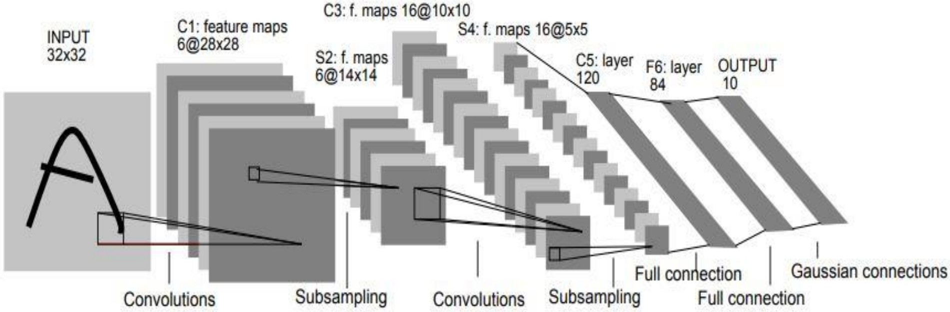


Example


Input
1 x 28 x 28
Convolution
32 x 26 x 26
Max Pooling
32 x 13 x 13
Convolution
64 x 11 x 11
Max Pooling
64 x 5 x 5
Flattened
1600
Fully connected
feedforward network
Output

Case Study: LeNet-5

[LeCun et al., 1998]



Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

This lecture references

[1] https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

- CS231 Stanford: https://www.youtube.com/watch?v=LxfUGhug-iQ
- Dr. Ghada‘s Slides of Pattern recognition course Spring 2018 http://www.fcih.net/ghada/pattern-recognition/
- https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html
- http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Lecture/CNN (v2).pdf
- https://ai.stackexchange.com/questions/8701/what-is-the-difference-between-a-receptive-field-and-a-feature-map