CS6421: Deep Neural Networks

Gregory Provan

Spring 2020

Lecture xx: Convolution Neural Networks

Based on notes from John Canny, Ismini Lourentzou

Overview

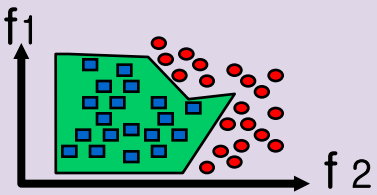
- Introduction
- Applications of CNNs
- CNN Operations
  - Convolution
  - Pooling
  - Classification

Introduction to CNNs

- A CNN is a feed-forward network that can extract topological properties from an image.
- Like almost every other neural network, they are trained with a version of the back-propagation algorithm.
- Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal preprocessing.
- They can recognize patterns with extreme variability (such as handwritten characters).

Classification

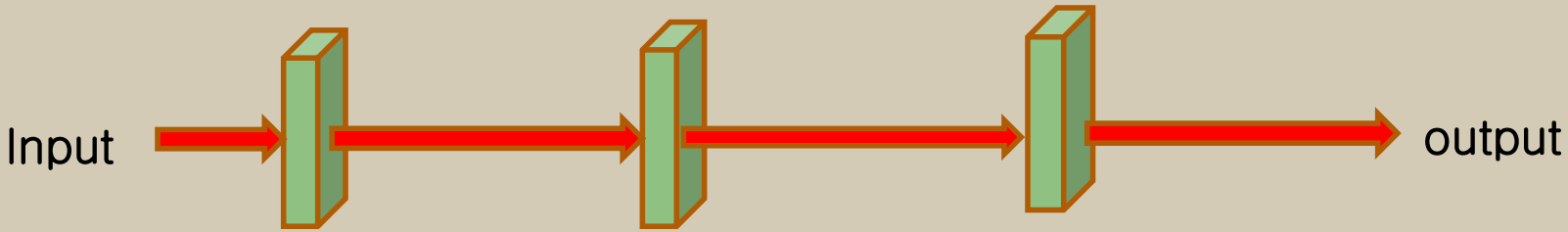
Traditional Classification



Input → **Pre-processing for feature extraction** → f1 ... fn → Classification → output

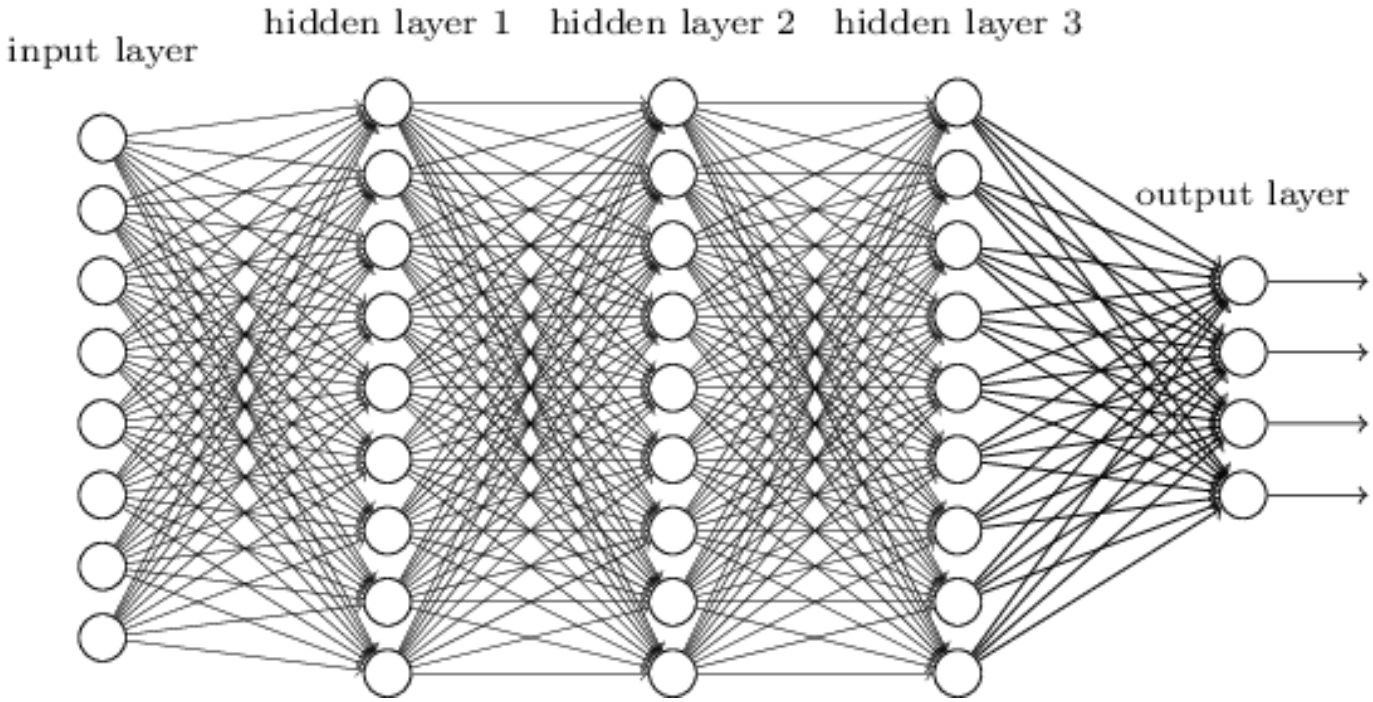
Input → **Feature extraction** → **Shift and distortion invariance** → **classification** → output

Convolutional neural network

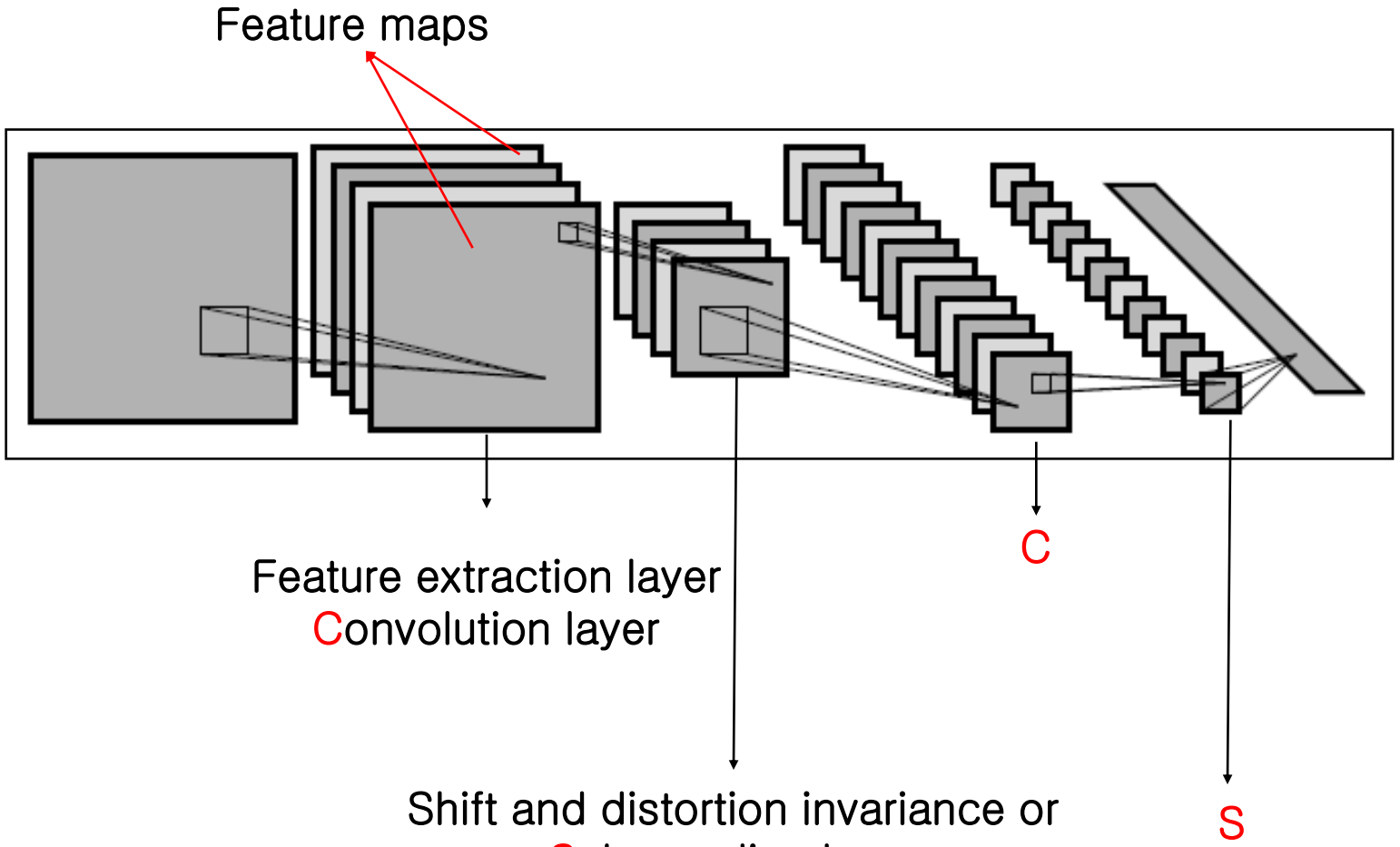


Architecture: Fully-Connected vs CNN

- We know it is good to learn a small model.
- From this fully connected model, do we really need all the edges?
- Can some of these be shared?



CNN Topology



Overview

- Introduction
- Applications of CNNs
- CNN Operations
  - Convolution
  - Pooling
  - Classification

Applications of CNNs

- Game playing
- Speech
- Text classification

AlphaGo: Playing the Game Go

19 x 19 matrix

Black: 1

white: -1

none: 0

Neural Network

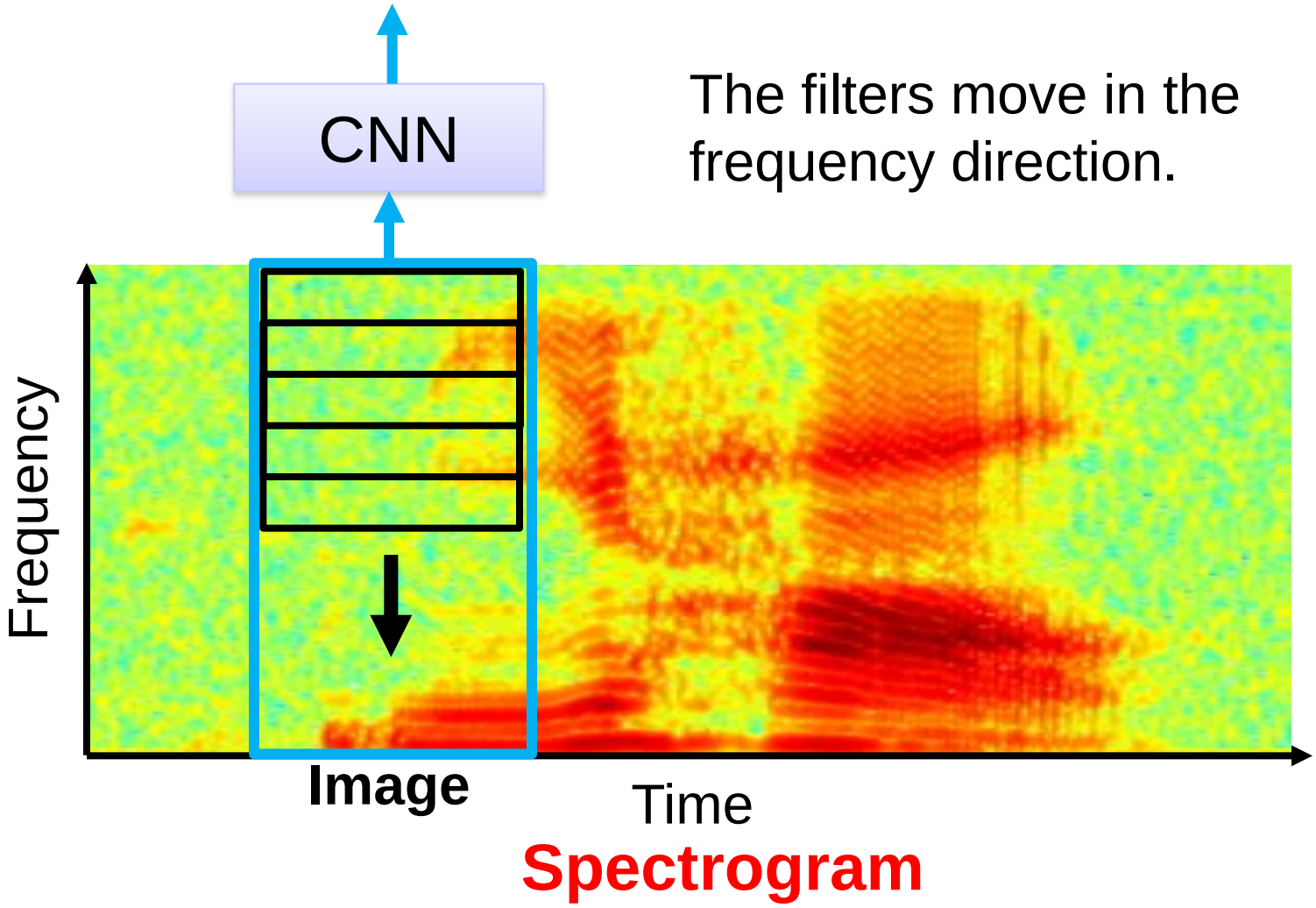
Next move

(19 x 19 positions)

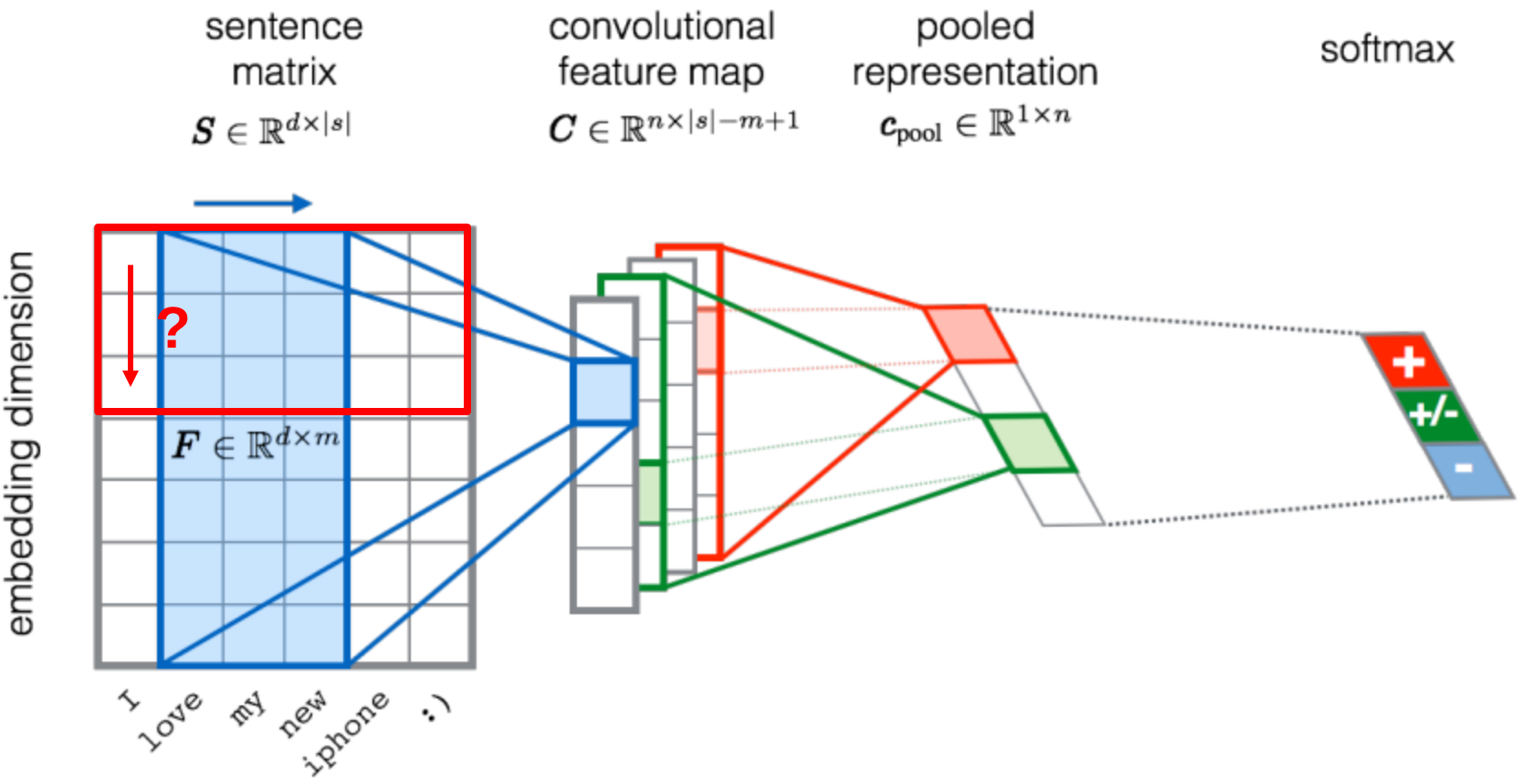
Fully-connected feedforward network can be used

But CNN performs much better

CNN in speech recognition



CNN in text classification

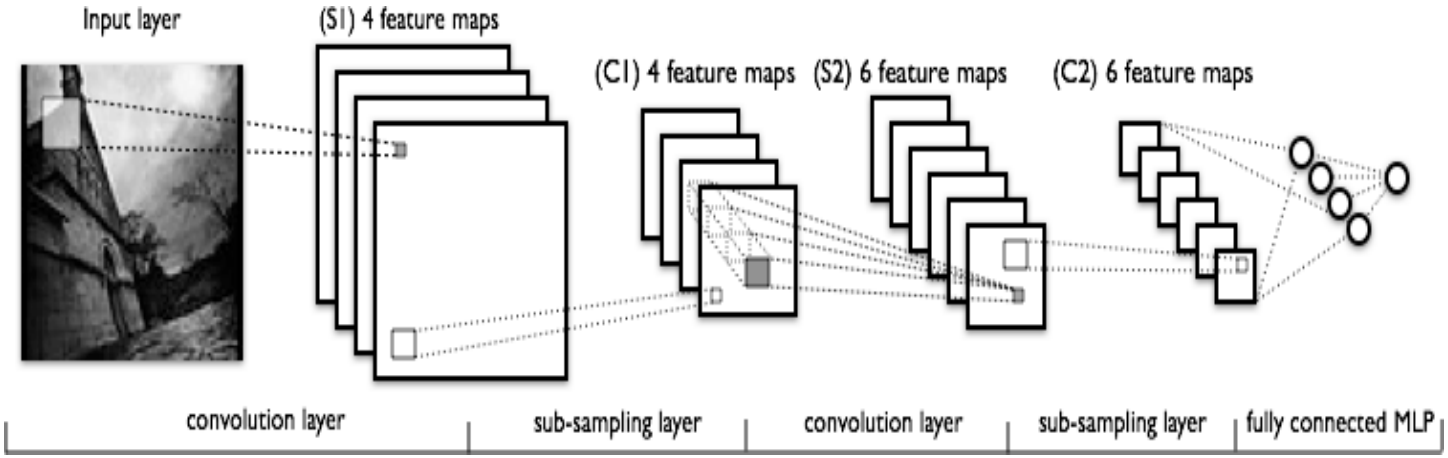


Source of image:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.703.6858&rep=rep1&type=pdf

Overview

- Introduction
- Applications of CNNs
- CNN Operations
  - Convolution
  - Pooling
  - Classification

Convolutional Neural Networks



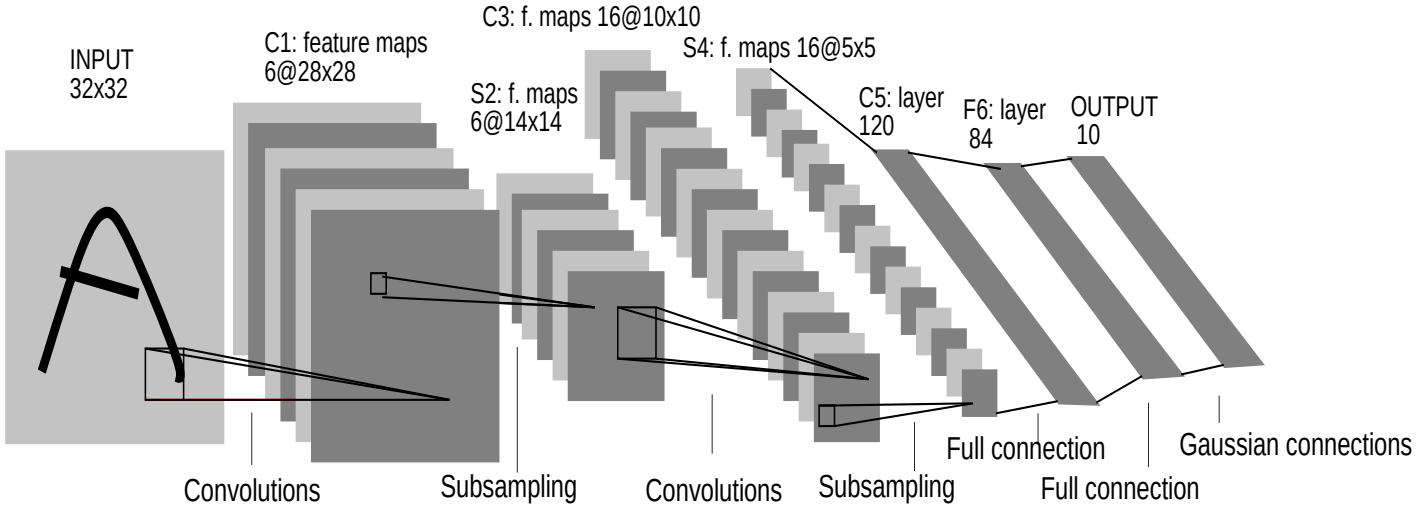
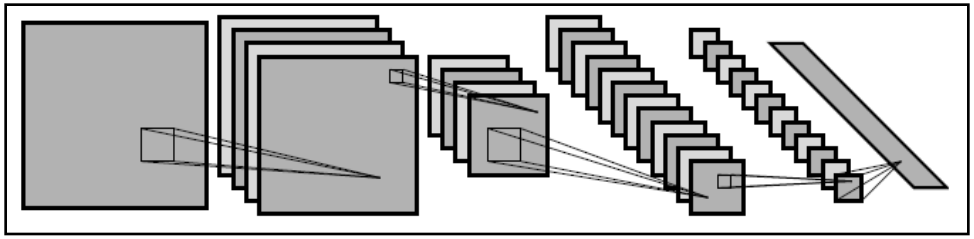
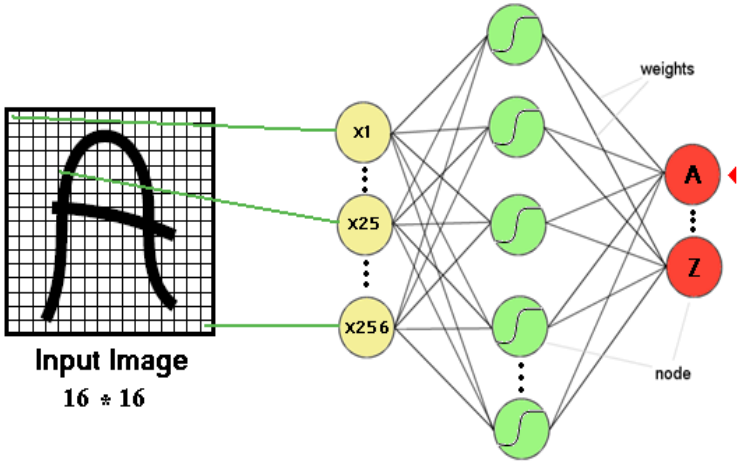


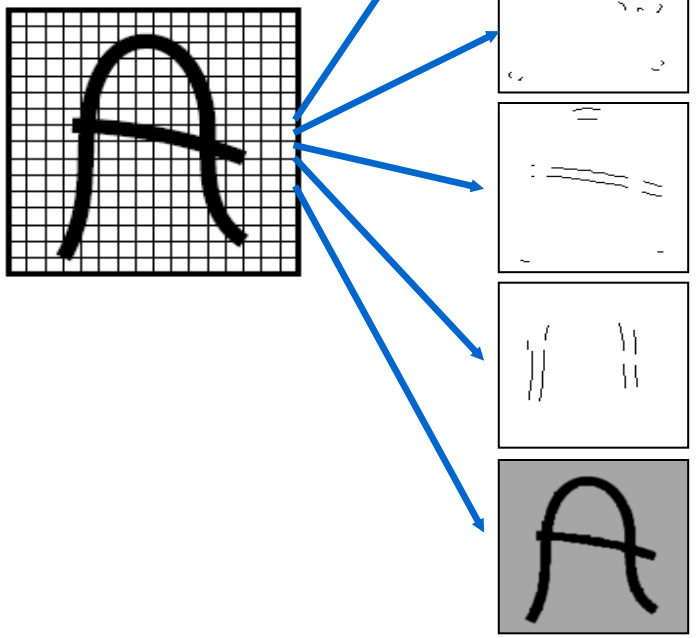
Image Credit: Yann LeCun, Kevin Murphy

Feature extraction layer or Convolution layer

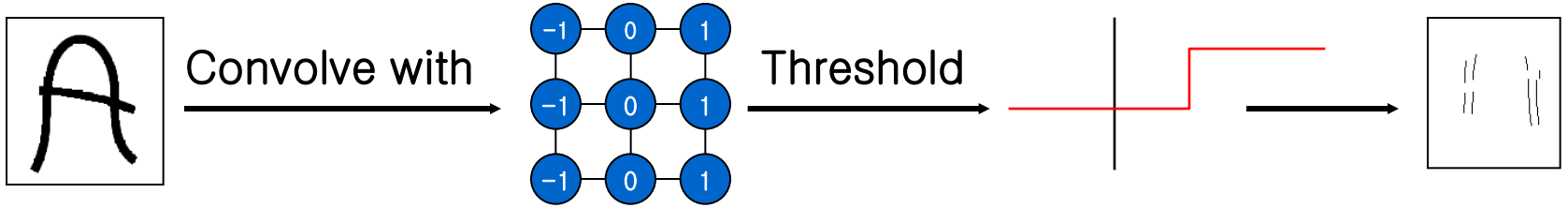
Detect the same feature at different positions in the input image.

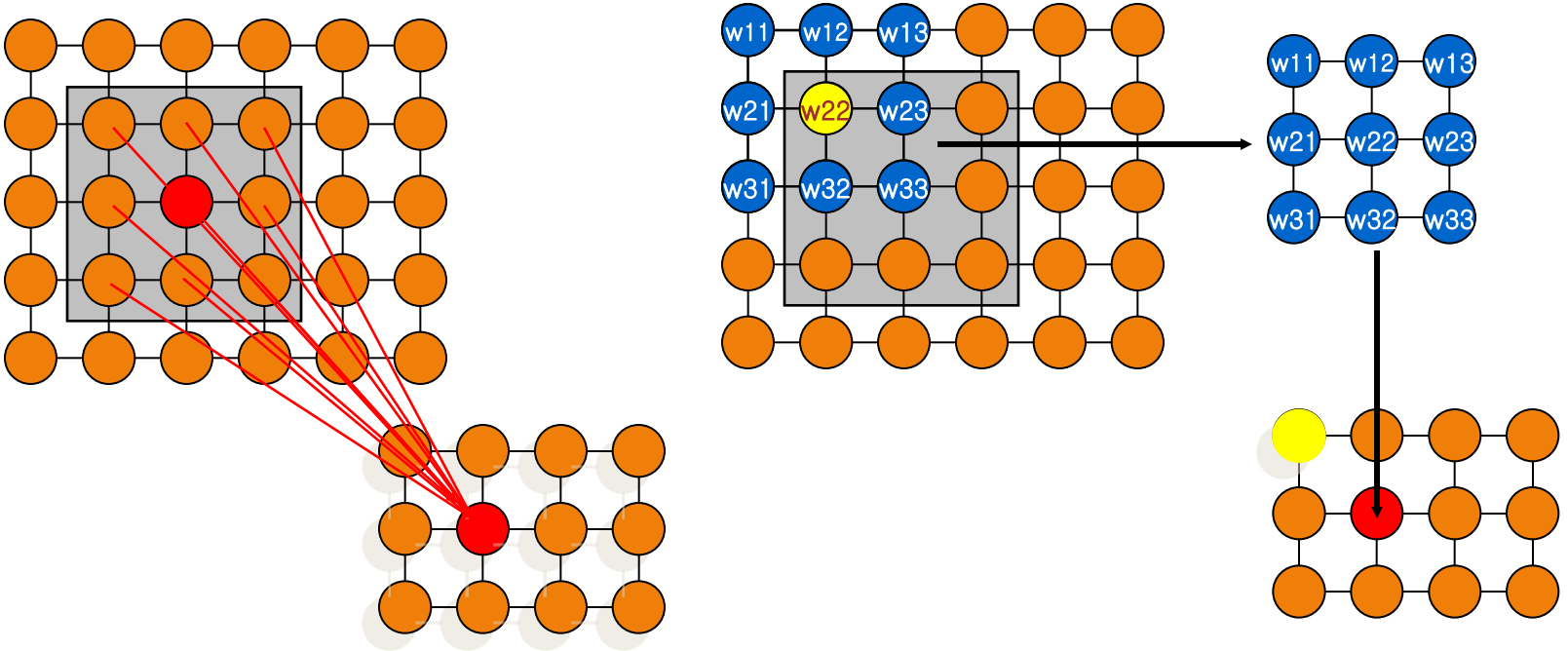






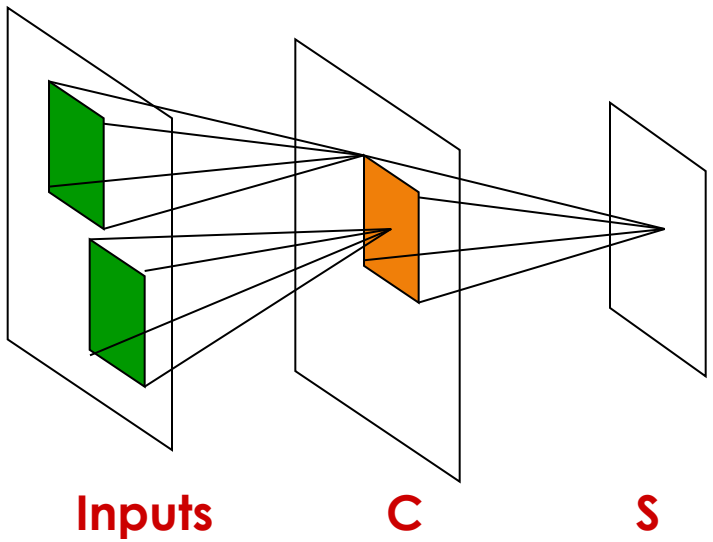
Feature extraction





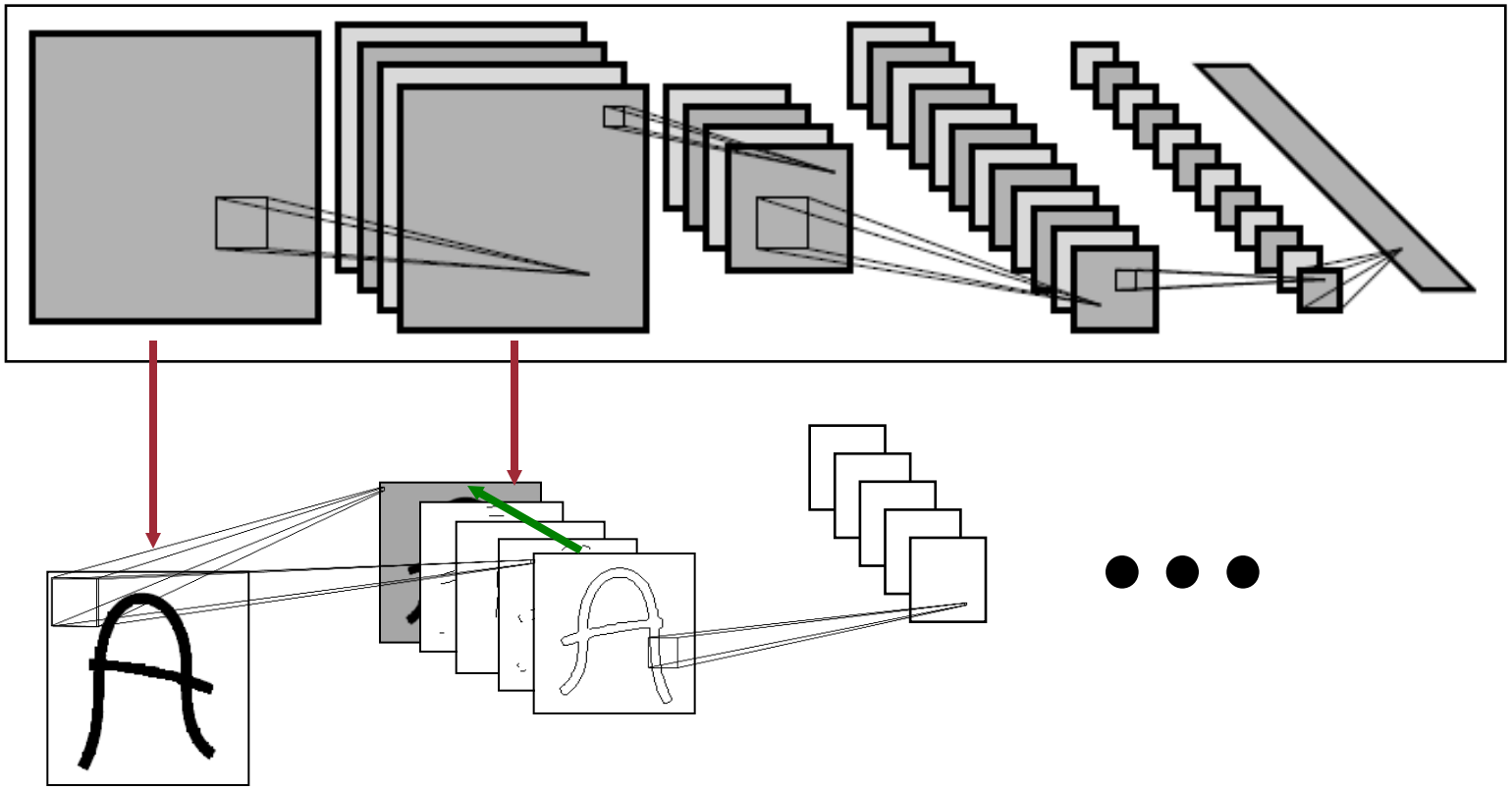
Feature extraction

- Shared weights: all neurons in a feature share the same weights (but not the biases).
- In this way all neurons detect the same feature at different positions in the input image.
- Reduce the number of free parameters.

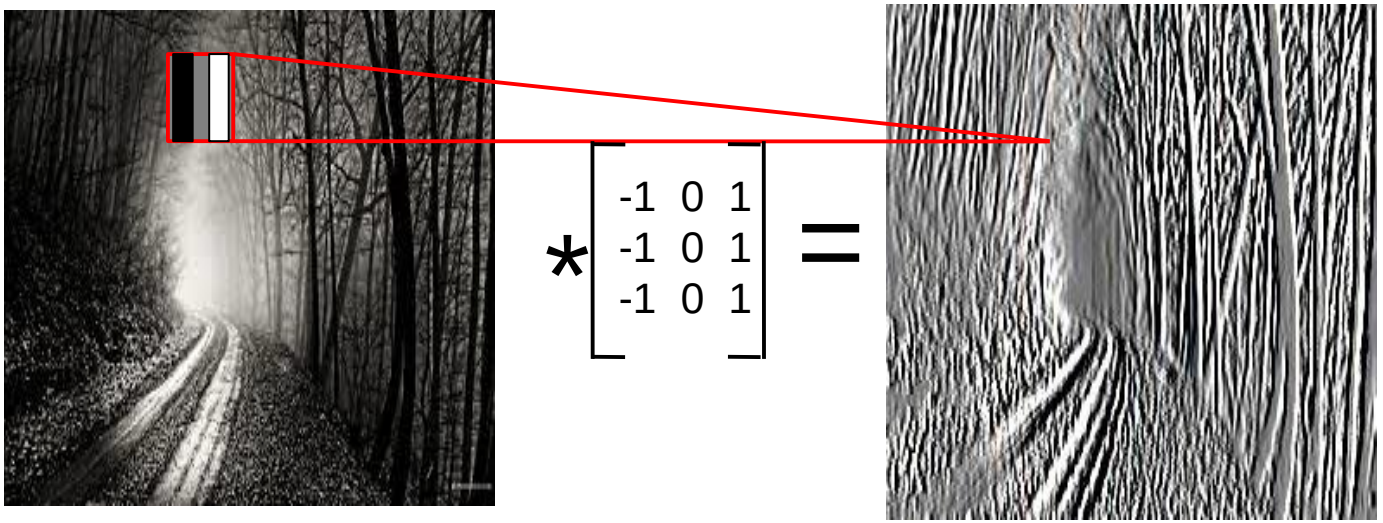


Feature extraction

- If a neuron in the feature map fires, this corresponds to a match with the template.

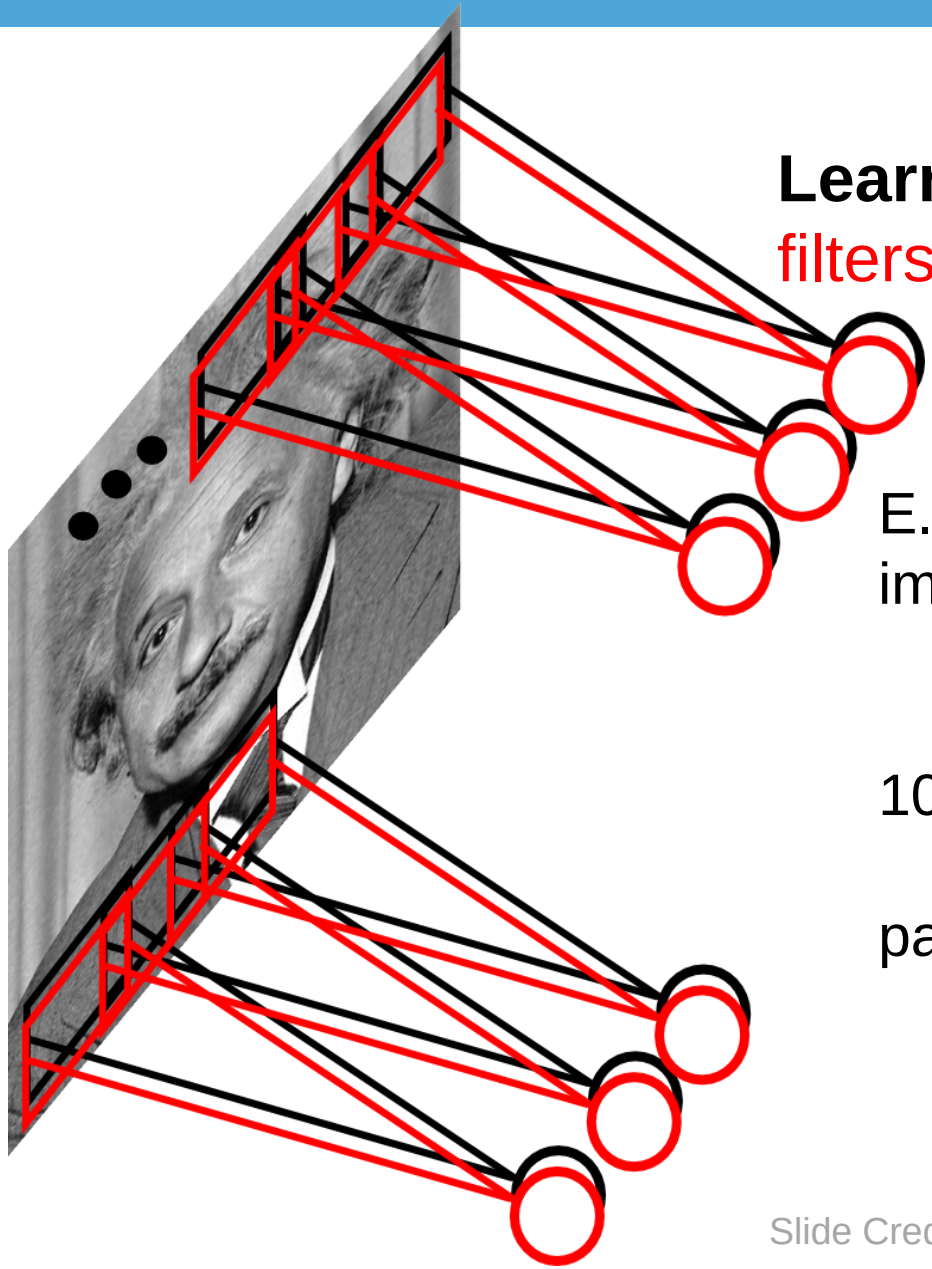


Convolutional Layer



Slide Credit: Marc'Aurelio Ranzato

Convolutional Layer



Learn multiple filters.

E.g.: 200x200 image

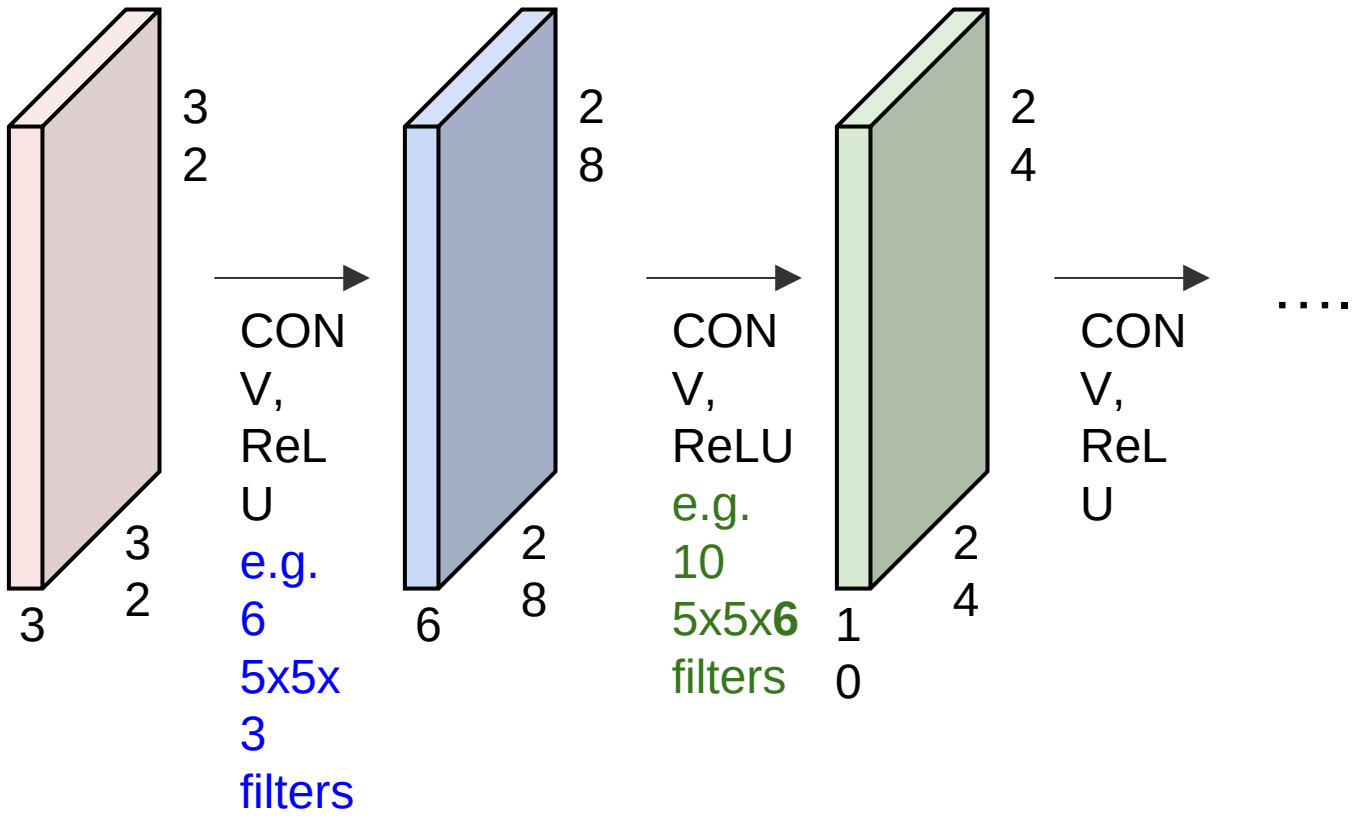
100 Filters

Filter size: 10x10

10K parameters

Slide Credit: Marc'Aurelio Ranzato

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

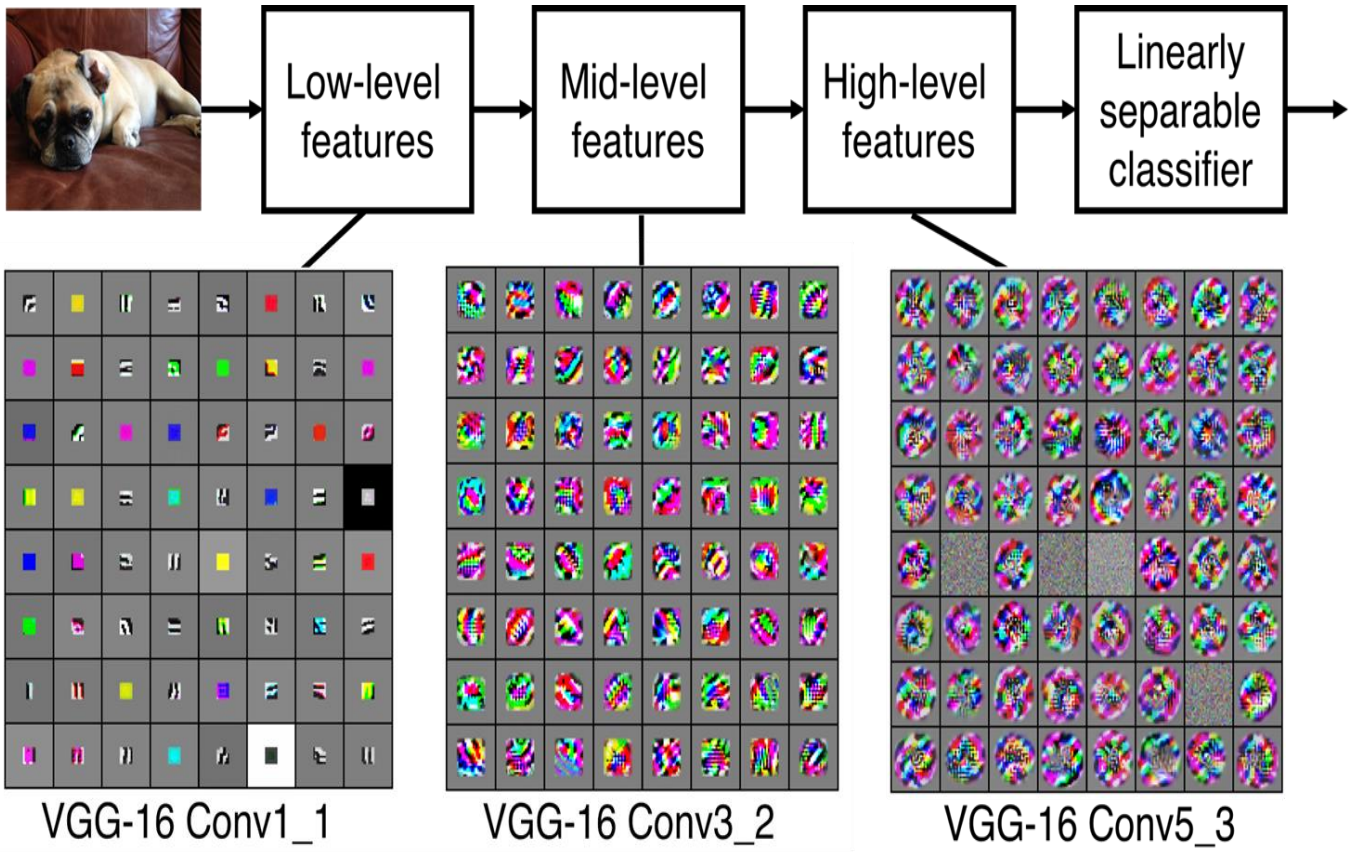


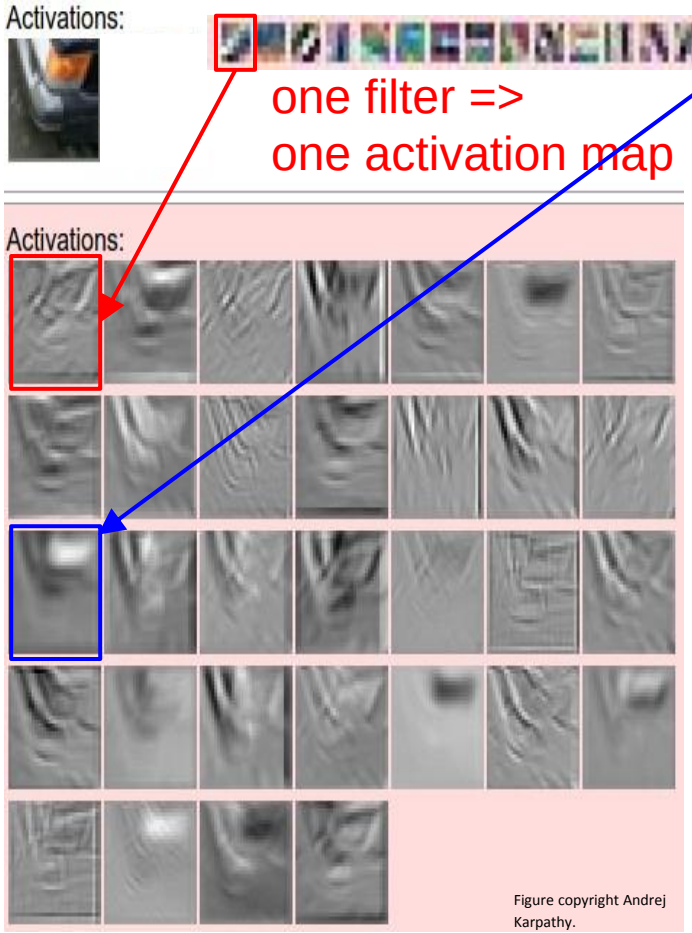
Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Extraction of Feature Maps at Multiple Levels

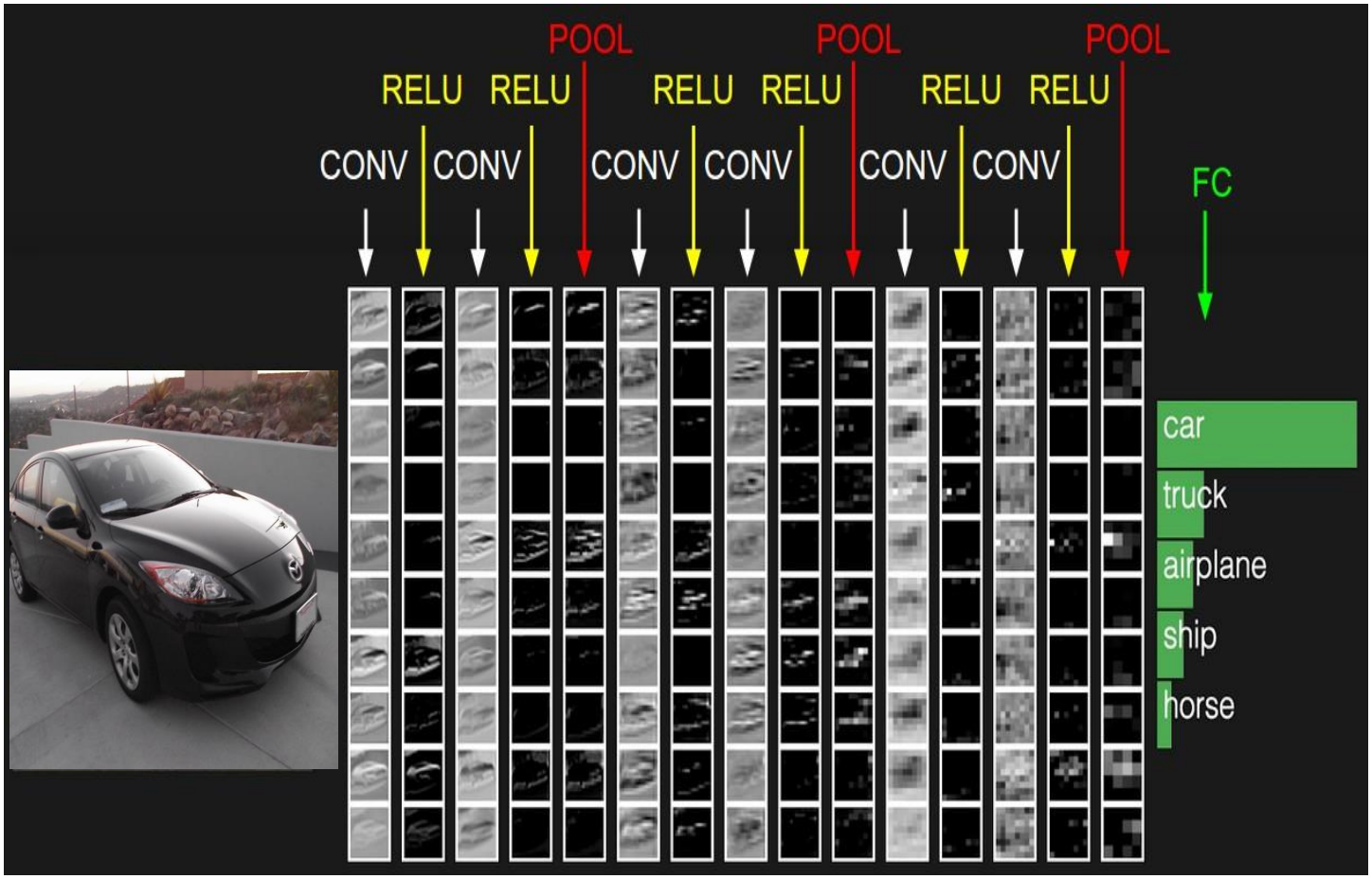
[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].





example 5x5 filters
(32 total)

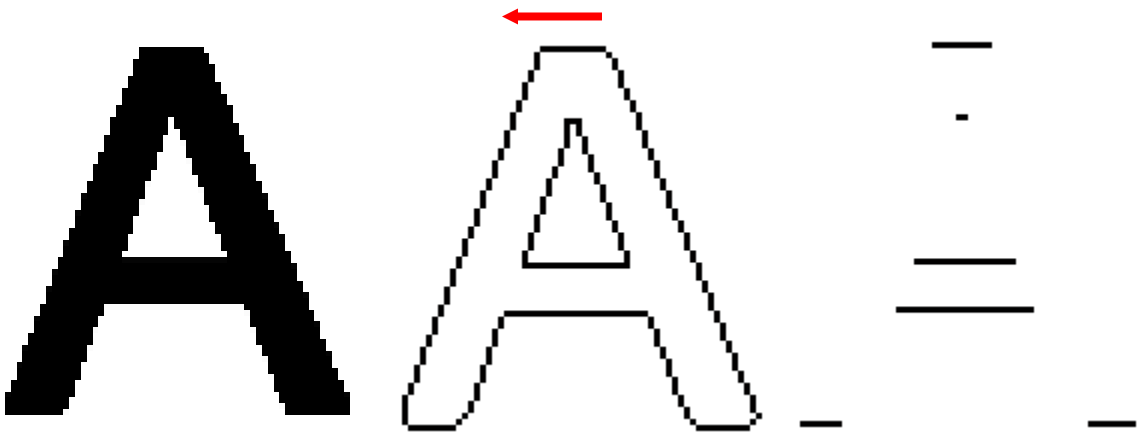


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

Subsampling layer

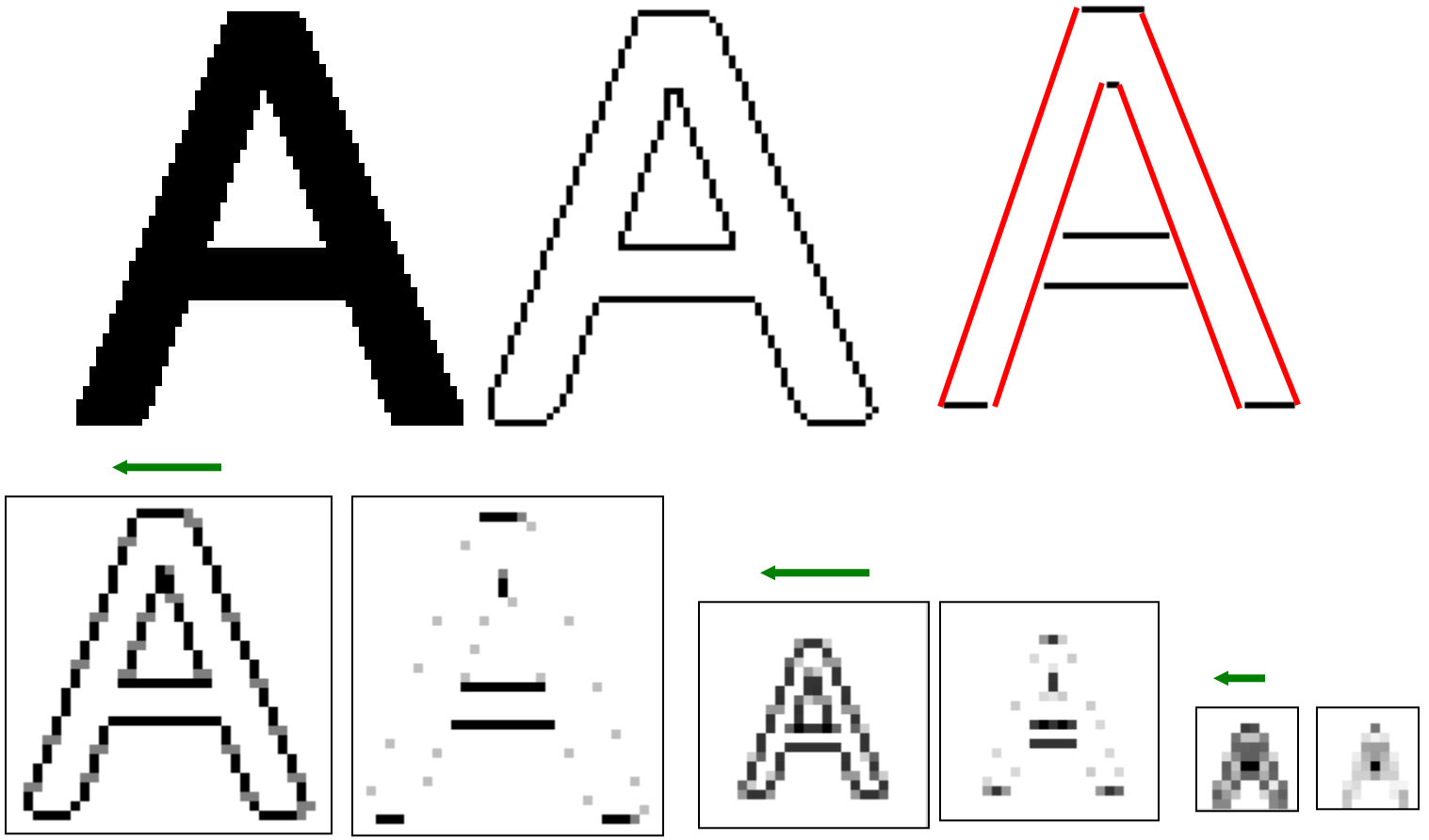
- the subsampling layers reduce the spatial resolution of each feature map

- By reducing the spatial resolution of the feature map, a certain degree of shift and distortion invariance is achieved.

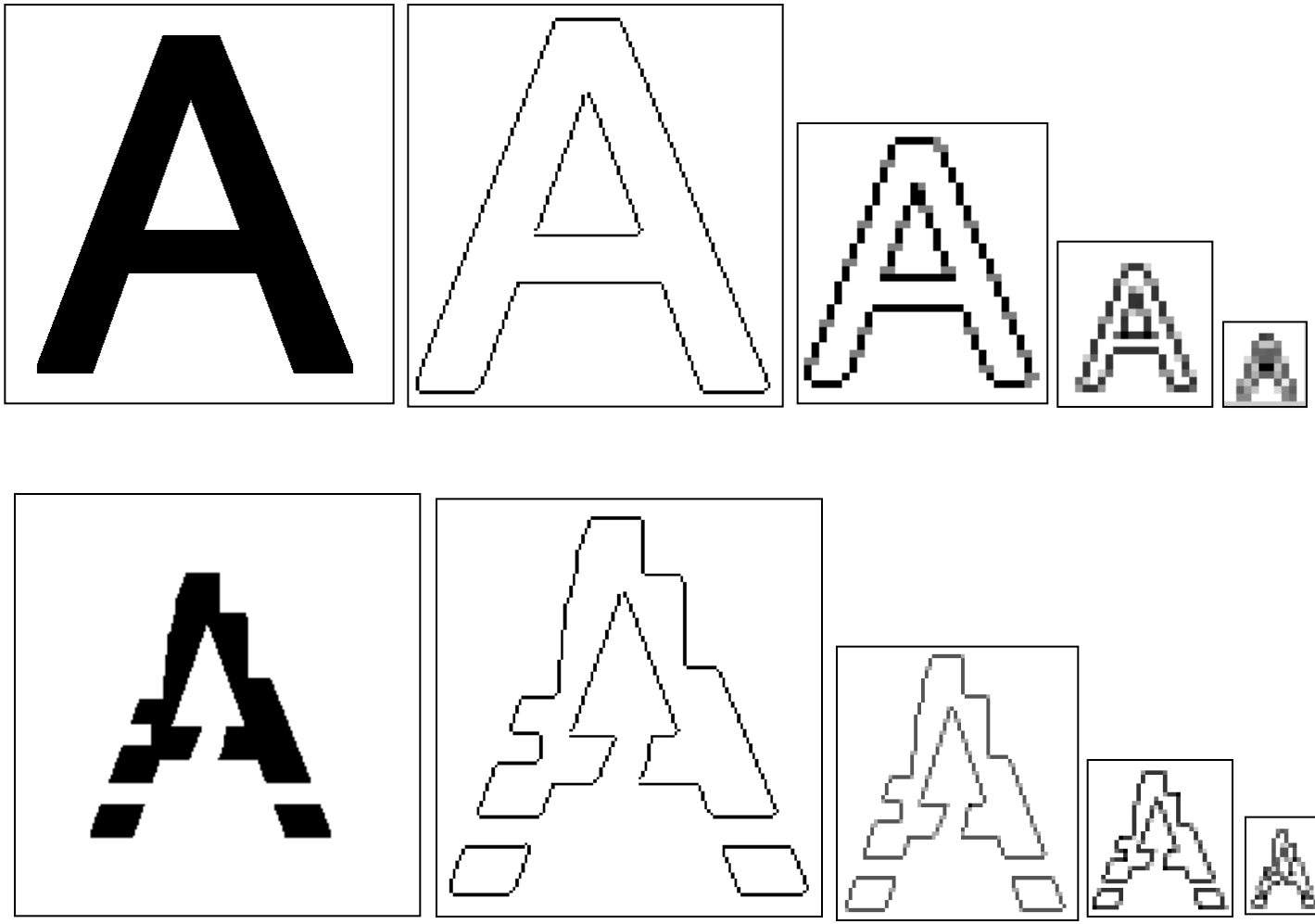


Subsampling layer

subsampling layers reduce the spatial resolution of each feature map

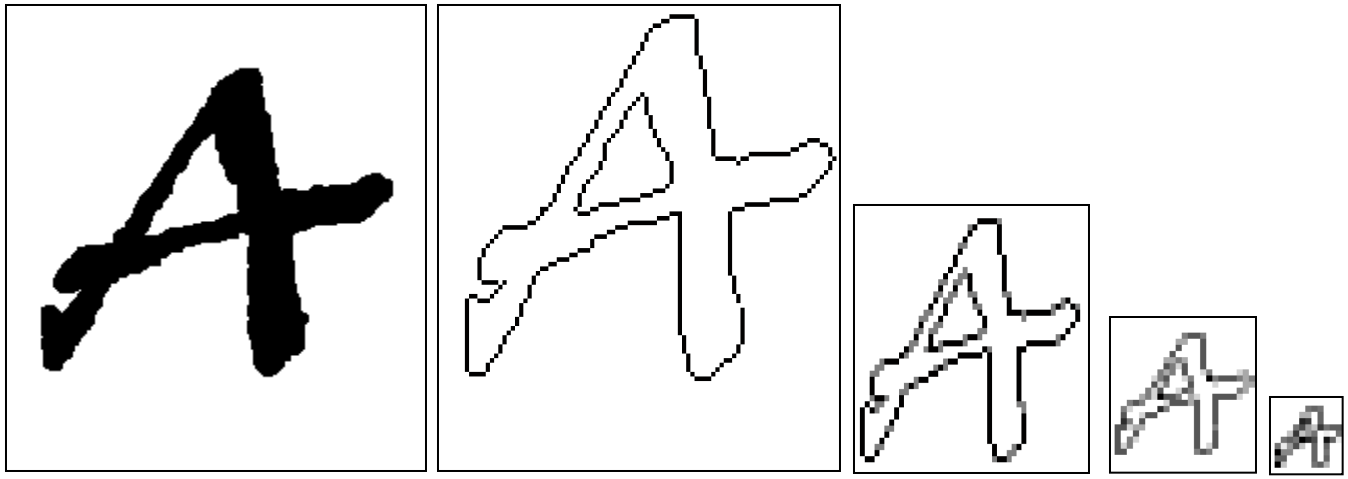


Subsampling layer



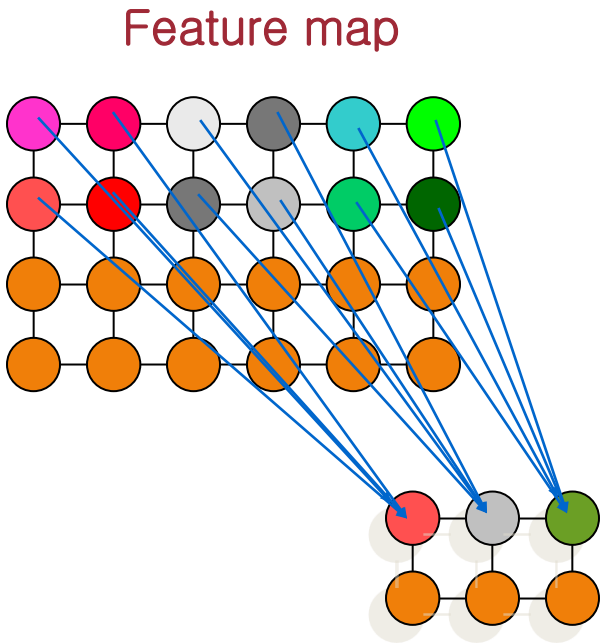
Subsampling layer

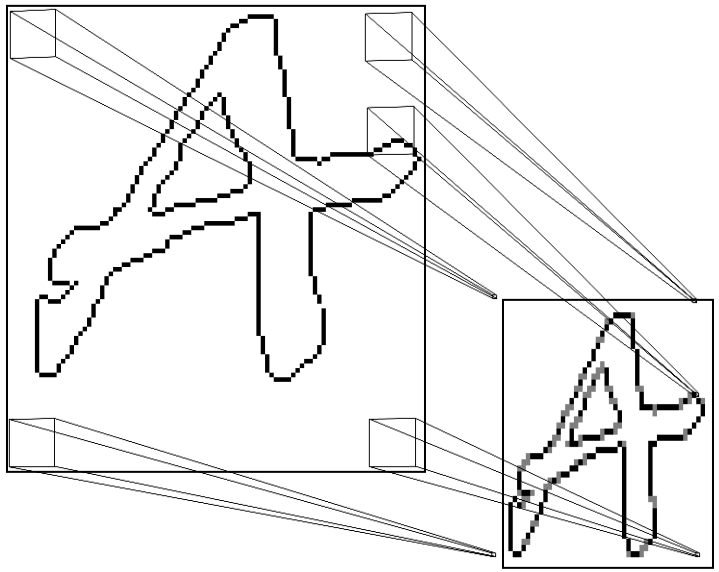
weight sharing is also applied in subsampling layers.



Subsampling layer

weight sharing reduces the effect of noise and shift or distortion





Process of CNN Inference

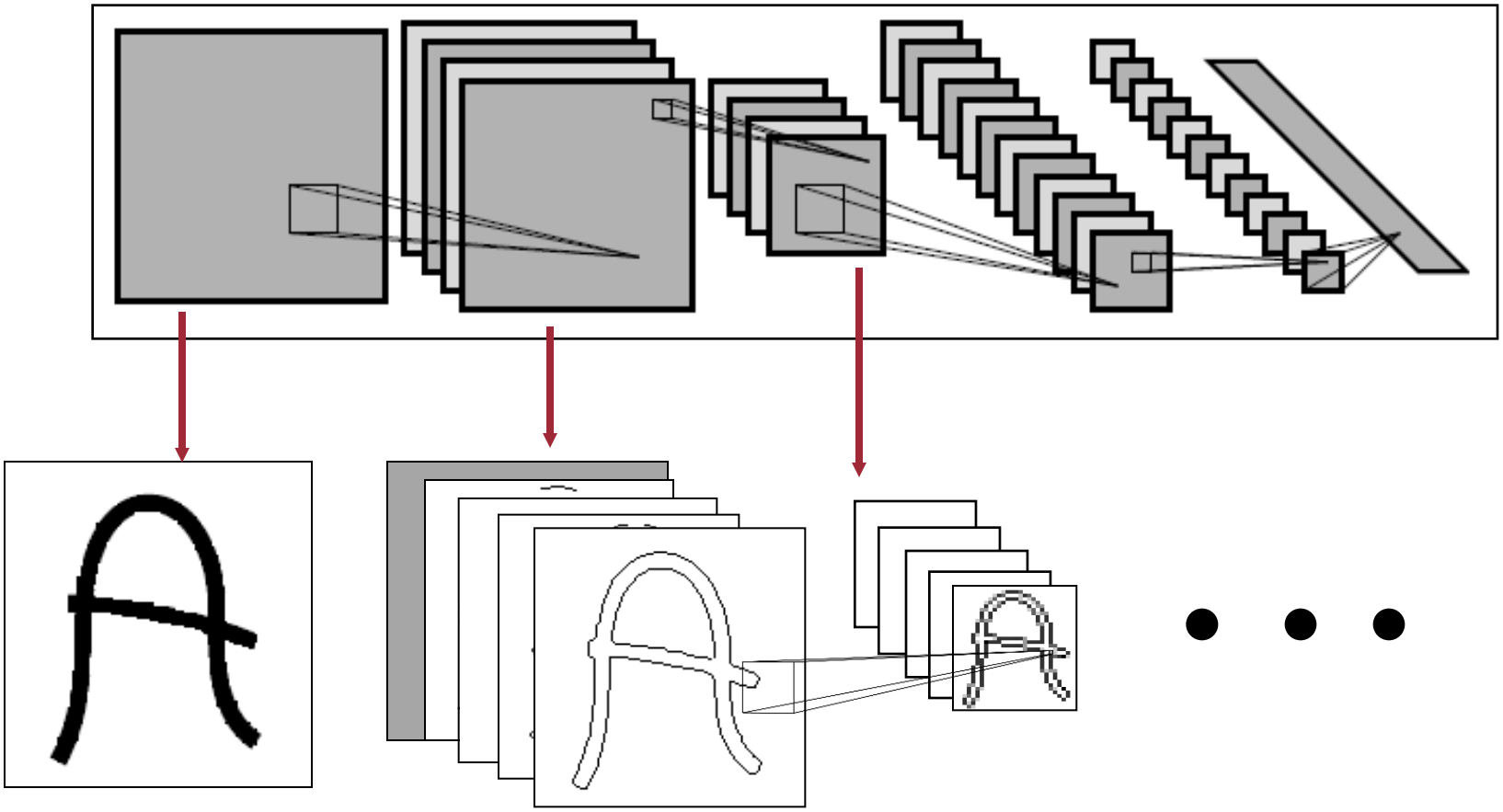


Image- and Region-Specific Filters

- CNNs use specific filters
- Detect specific image properties
  - Multiple levels
    - Low-level: edge
    - Higher-level: face

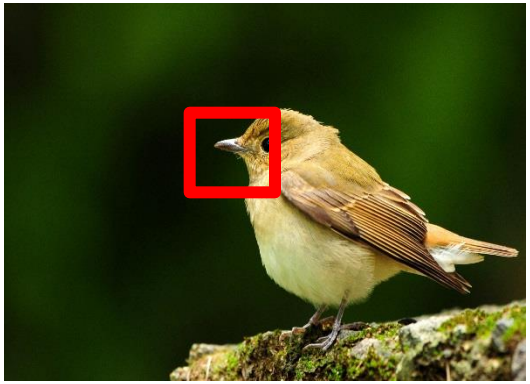
Consider learning an image:

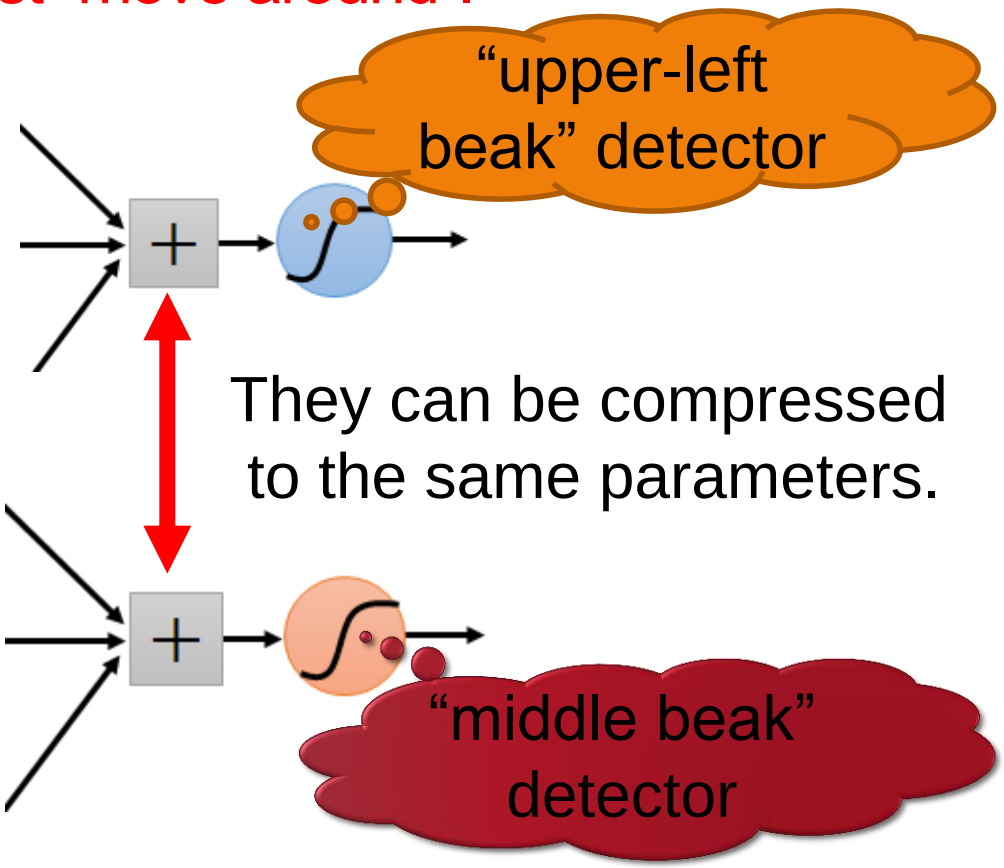
- Some patterns are much smaller than the whole image

Can represent a small region with fewer parameters

CNN for Image Analysis

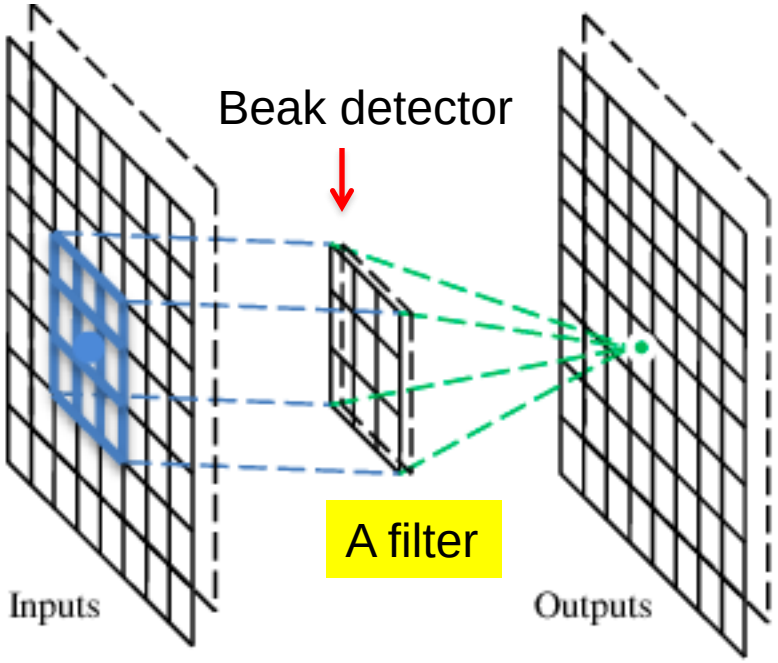
Same pattern appears in different places:
They can be compressed!
What about training a lot of such “small” detectors and each detector must “move around”.





A convolutional layer

- A CNN is a neural network with some convolutional layers (and some other layers).
- A convolutional layer has a number of filters that performs a convolution operation.

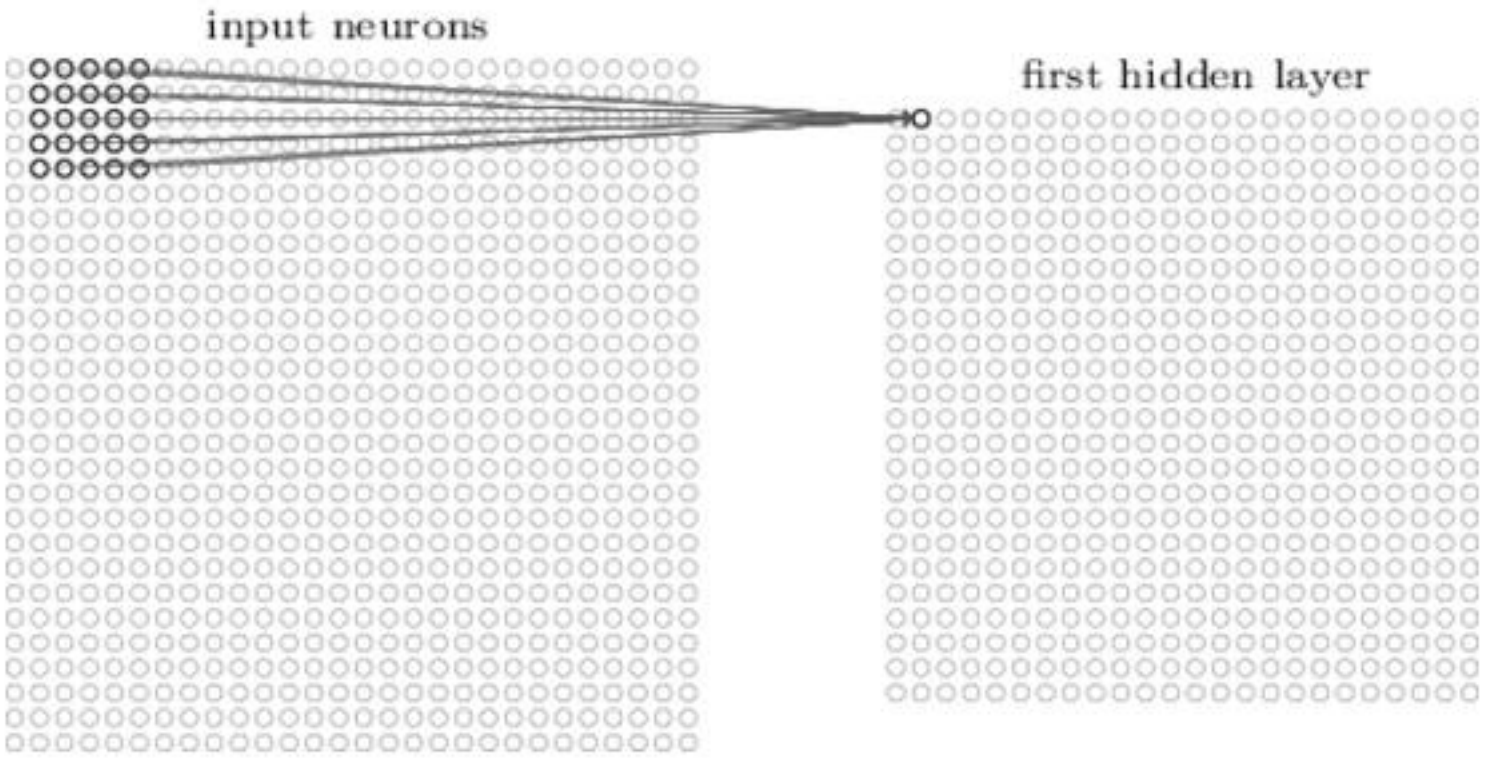


What's a convolution?

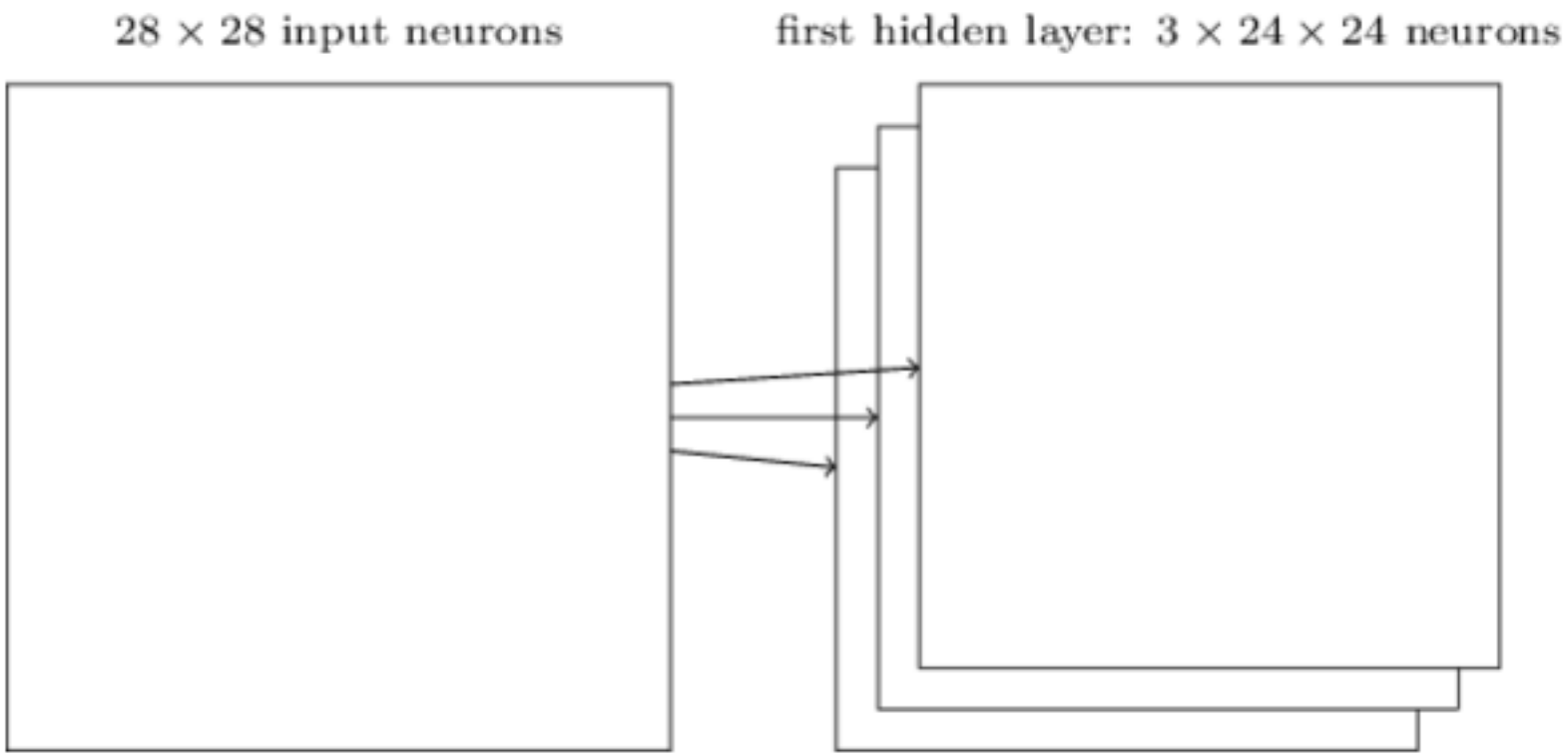
- Basic idea:
  - Pick a 3x3 matrix F of weights
  - Slide this over an image and compute the "inner product" (similarity) of F and the corresponding field of the image, and replace the pixel in the center of the field with the output of the inner product operation
- Key point:
  - Different convolutions extract different types of low-level "features" from an image
  - All that we need to vary to generate these different features is the weights of F

Convolving an image with an ANN

Note that the parameters in the matrix defining the convolution are **tied** across all places that it is used

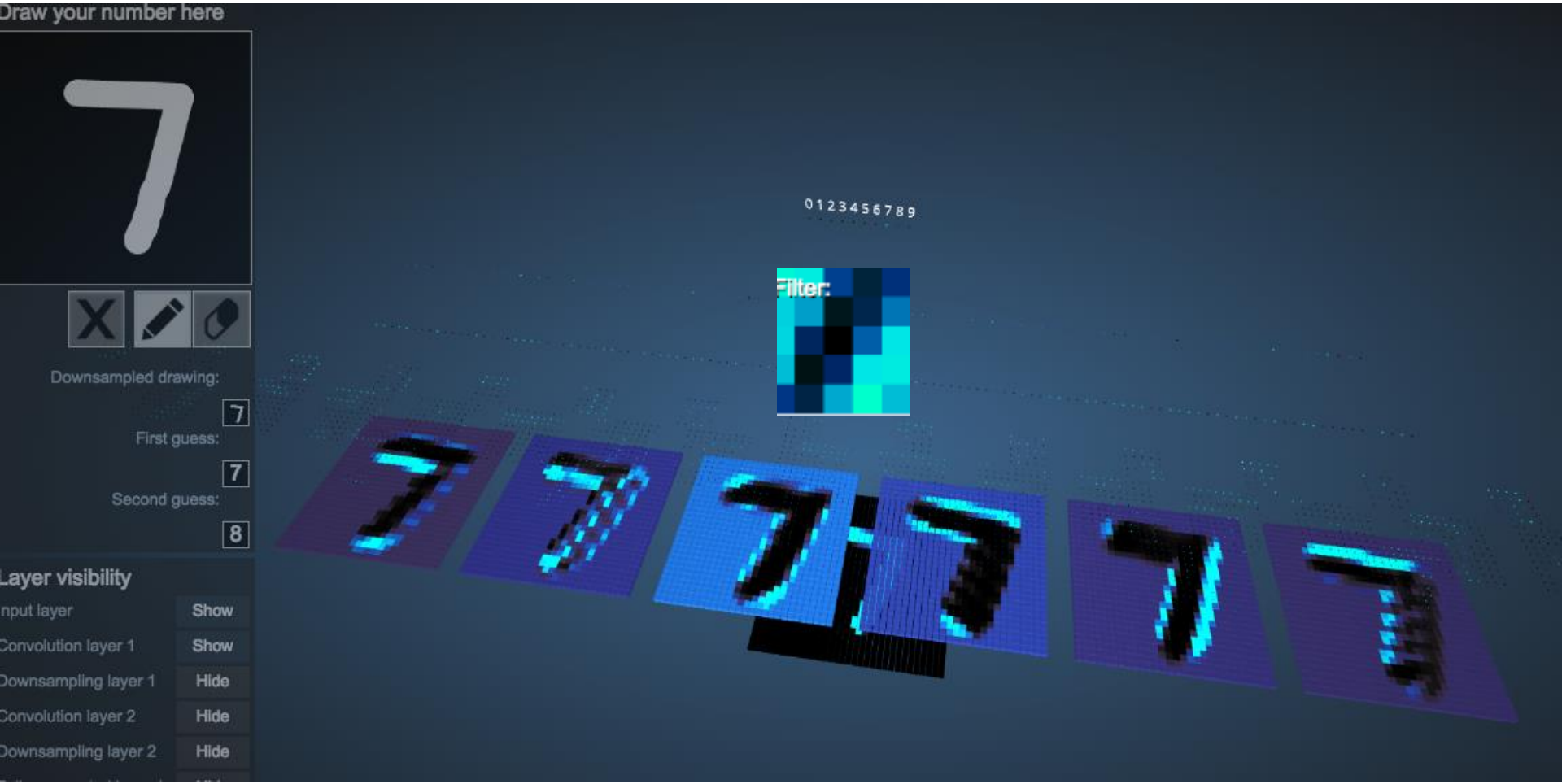


How do we do many convolutions of an image with an ANN?



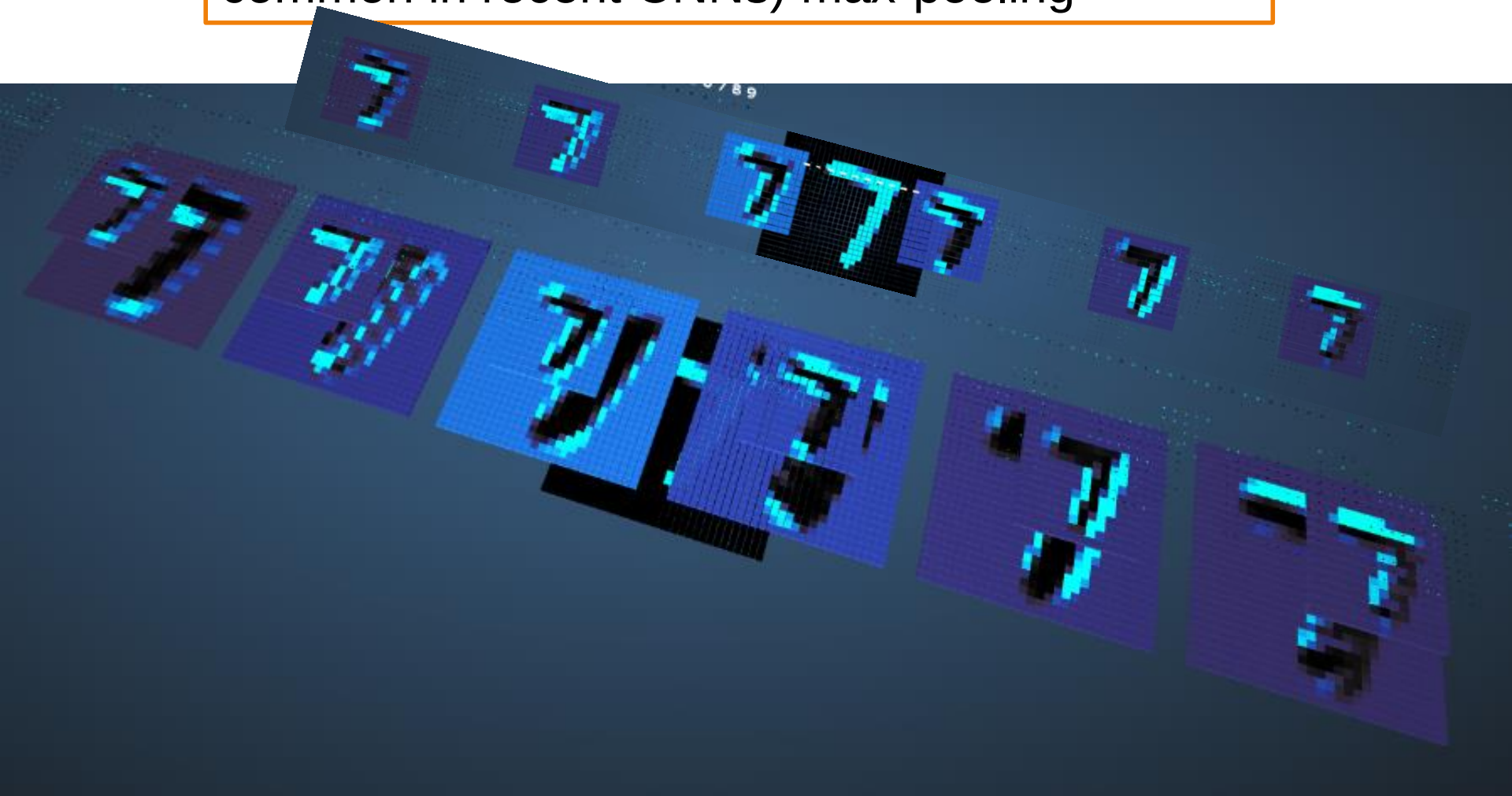
Example: 6 convolutions of a digit

http://scs.ryerson.ca/~aharley/vis/conv/



CNNs typically alternate convolutions, non-linearity, and then downsampling

Downsampling is usually averaging or (more common in recent CNNs) max-pooling



Why do max-pooling?

- Saves space
- Reduces overfitting?
- Because I'm going to add *more* convolutions after it!
  - Allows the short-range convolutions to extend over larger subfields of the images
    - So we can spot larger objects
    - Eg, a long horizontal line, or a corner, or …

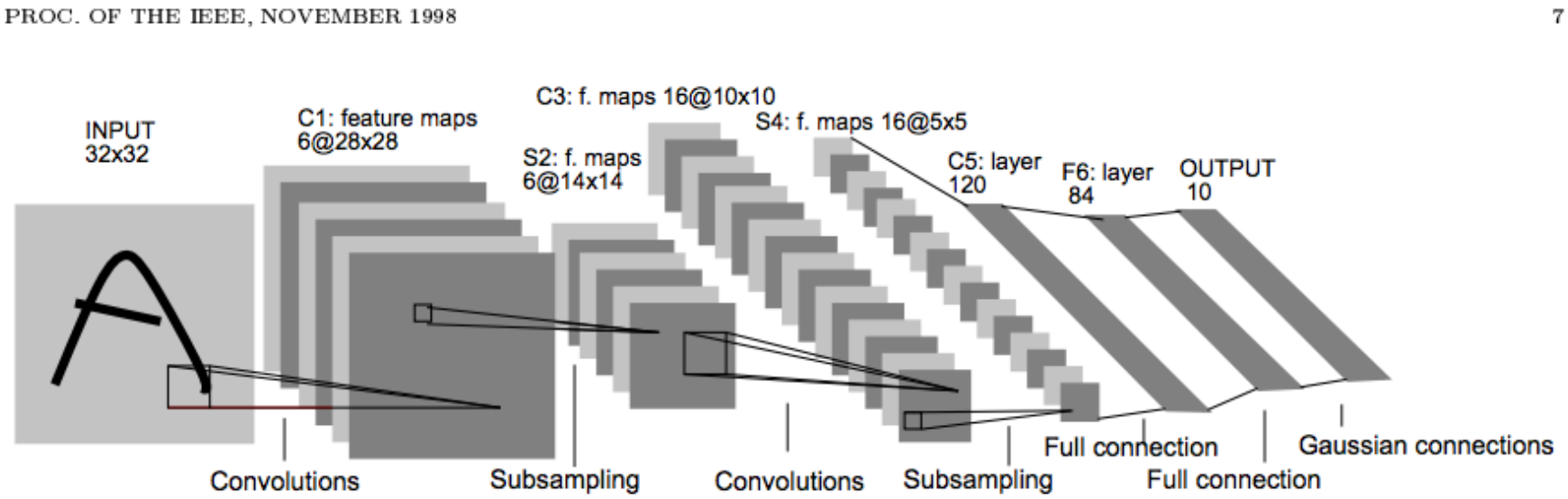


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Another CNN visualization

https://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html

Network Visualization

input (24x24x1)
max activation: 0.99607, min: 0

Activations:

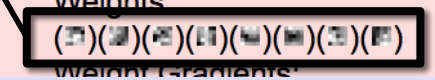
Weights:

filter size 5x5x1, stride 1
max activation: 2.96187, min: -5.43735
max gradient: 0.00068, min: -0.00102
parameters: 8x5x5x1+8 = 208

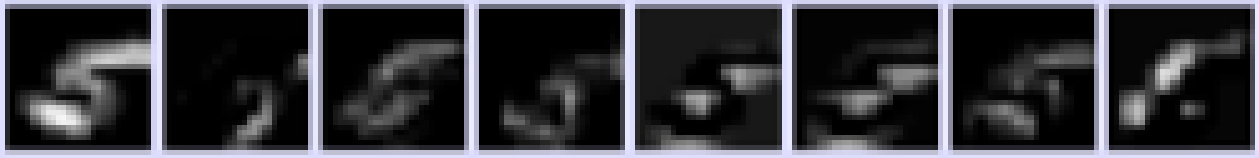
Activations:

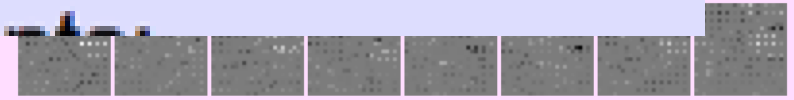
Activation Gradients:

Weights:



Activations:





pool (12x12x8)
pooling size 2x2, stride 2
max activation: 2.96187, min: 0
max gradient: 0.00106, min: -0.00102

Activations:

Activation Gradients:

Alternating Convolution and Sub-sampling

5 layers up

The subfield in a large dataset that gives the strongest output for a neuron

Convolution

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

Convolution filters

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

Filter 1

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

Filter 2

⋮ ⋮

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

Convolution

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

Dot product → 3 -1

6 x 6 image

Convolution

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

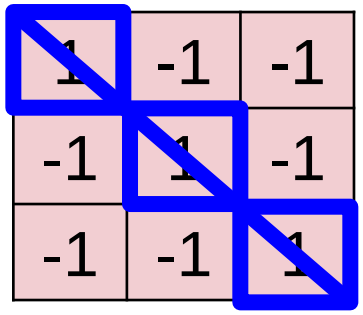
6 x 6 image

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

Filter 1

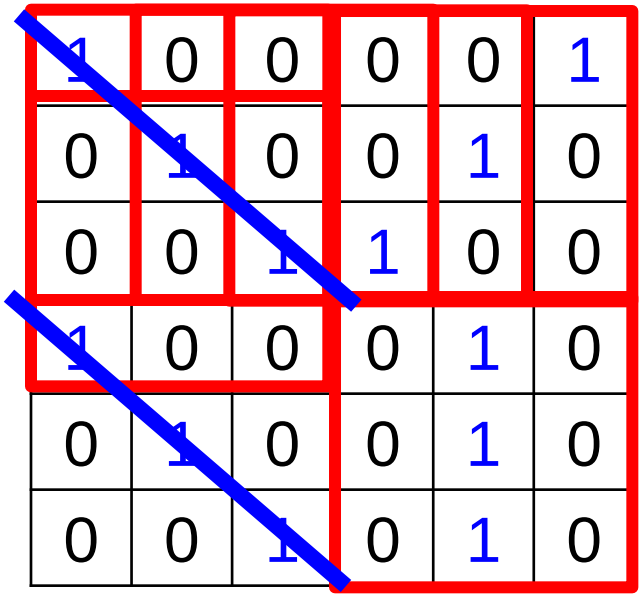
3 -3

Convolution

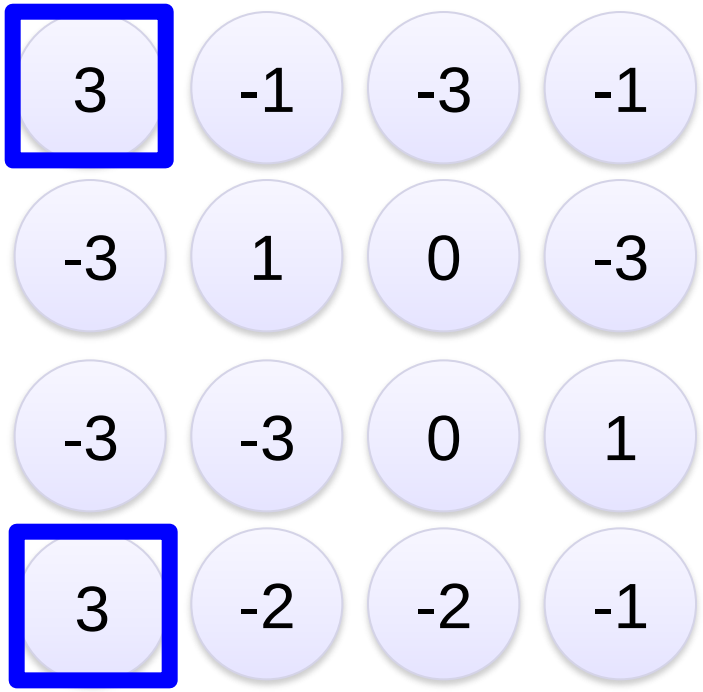


Filter 1

stride=1



6 x 6 image



Convolution

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

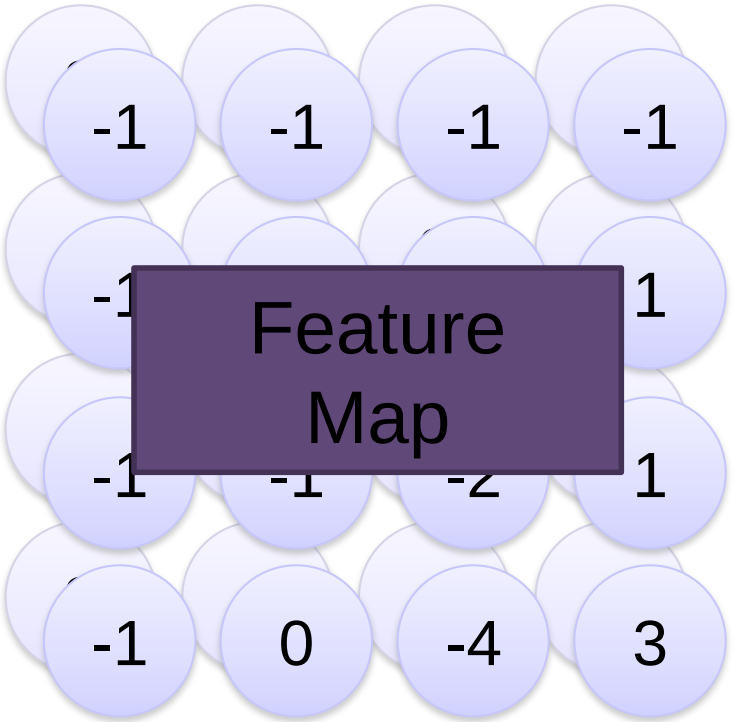
Filter 2

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

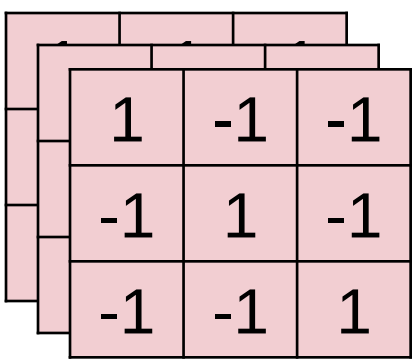
6 x 6 image

Repeat this for each filter

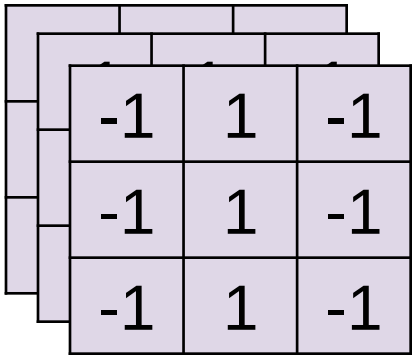


Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Color image: RGB 3 channels

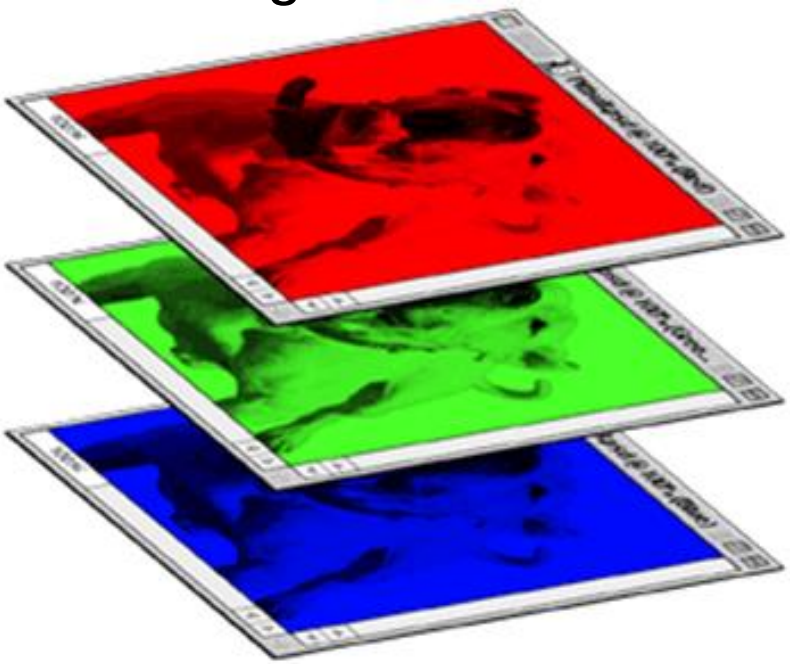


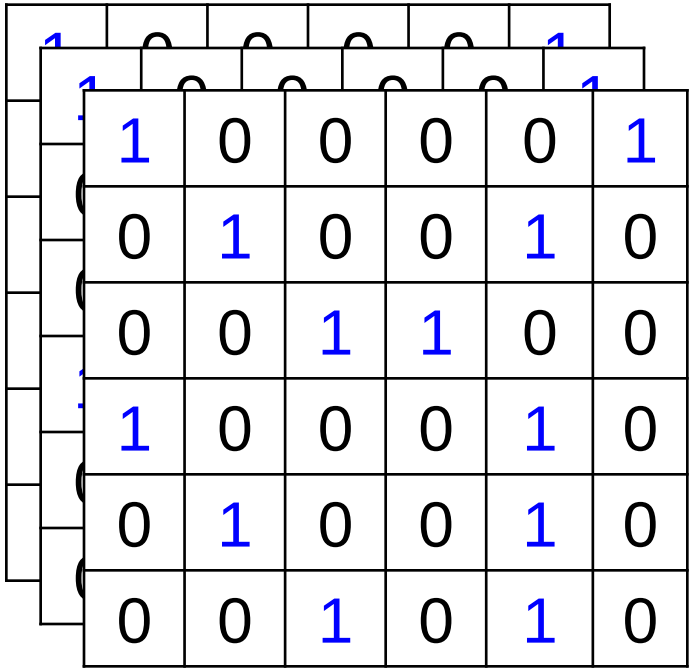
Filter 1



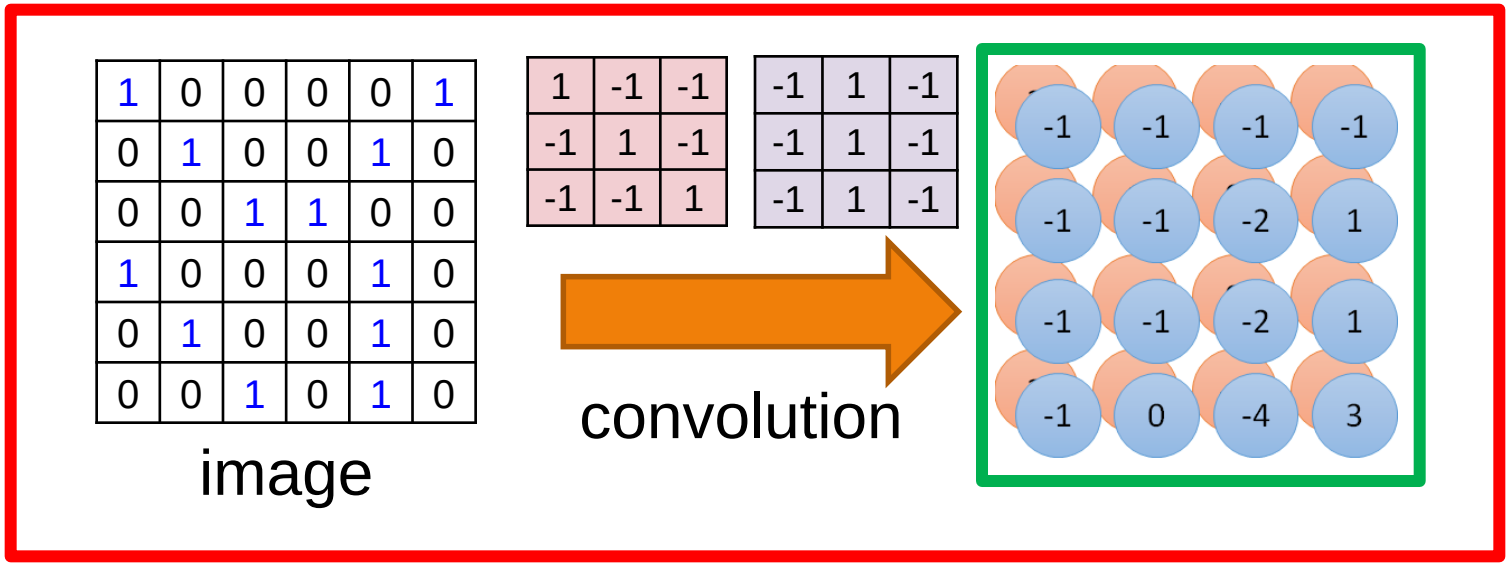
Filter 2

Color image

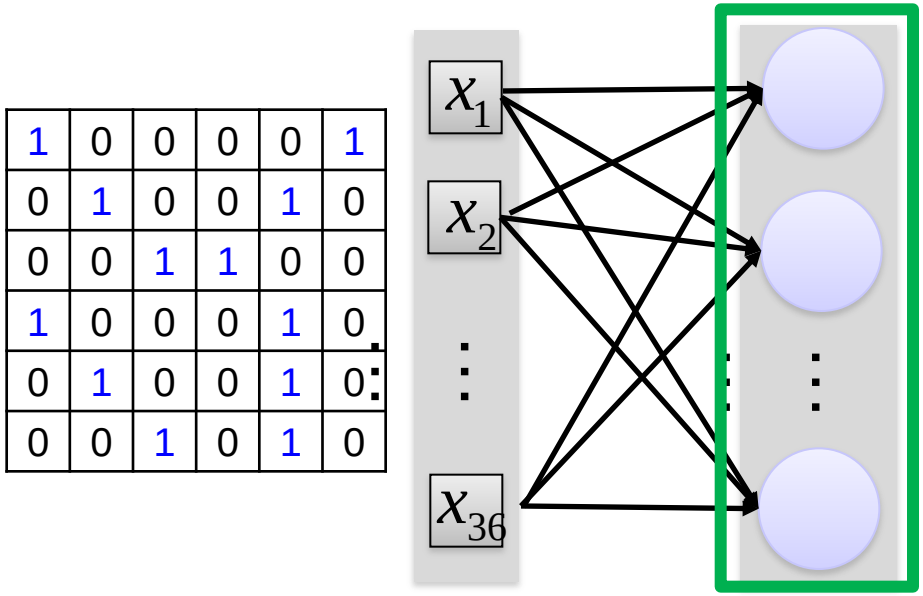


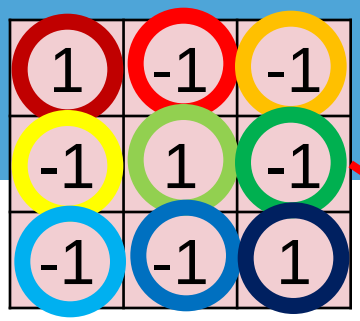


Convolution v.s. Fully Connected



Fully-connected

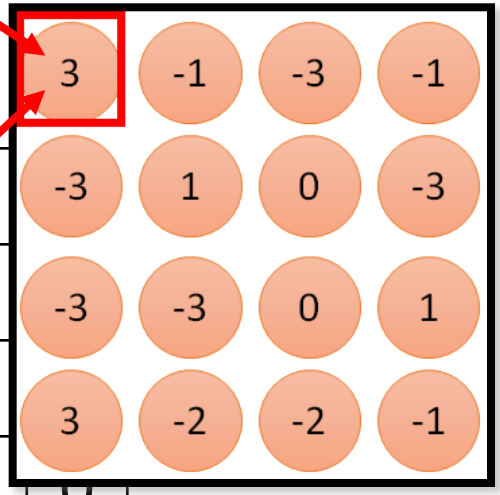




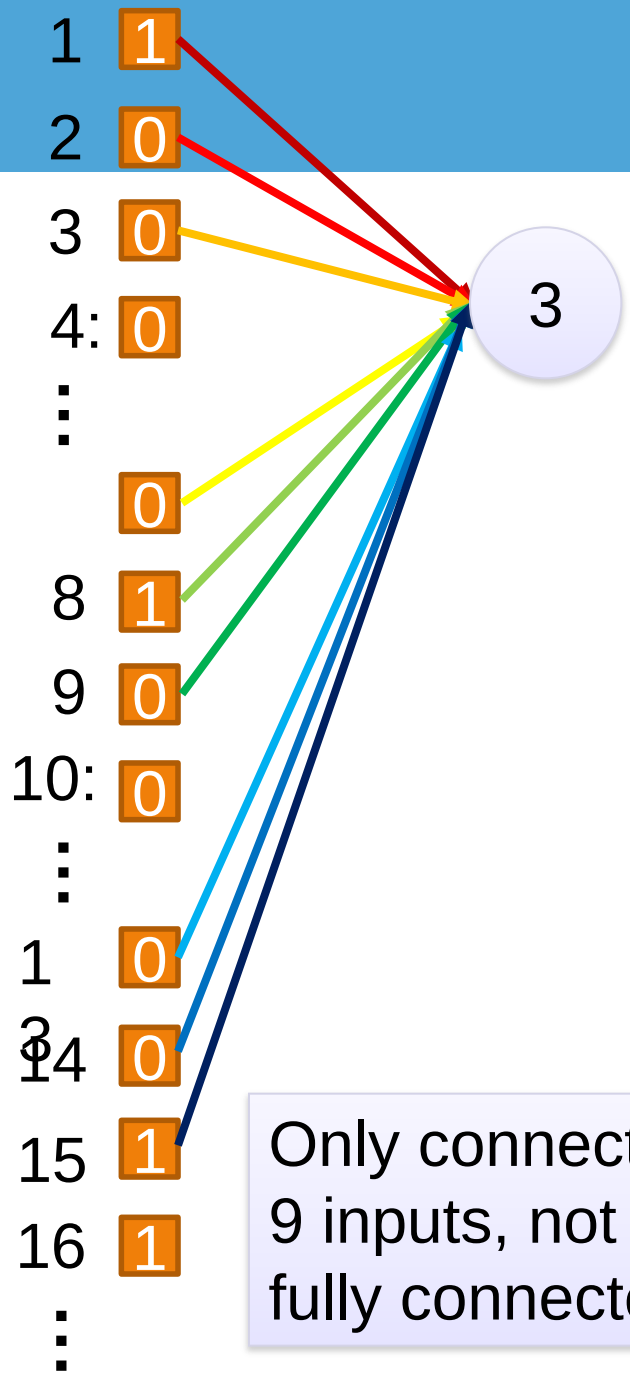
Filter 1

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

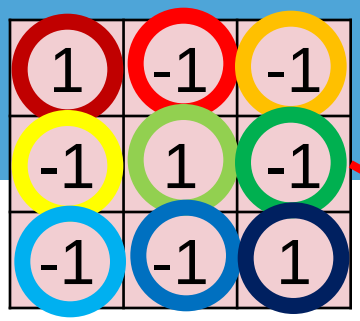
6 x 6 image



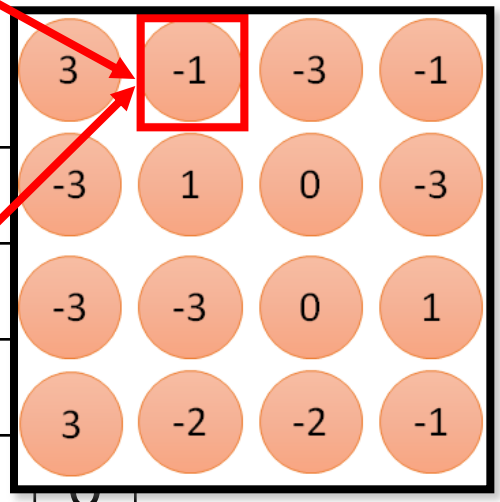
fewer parameters!



Only connect to 9 inputs, not fully connected



Filter 1

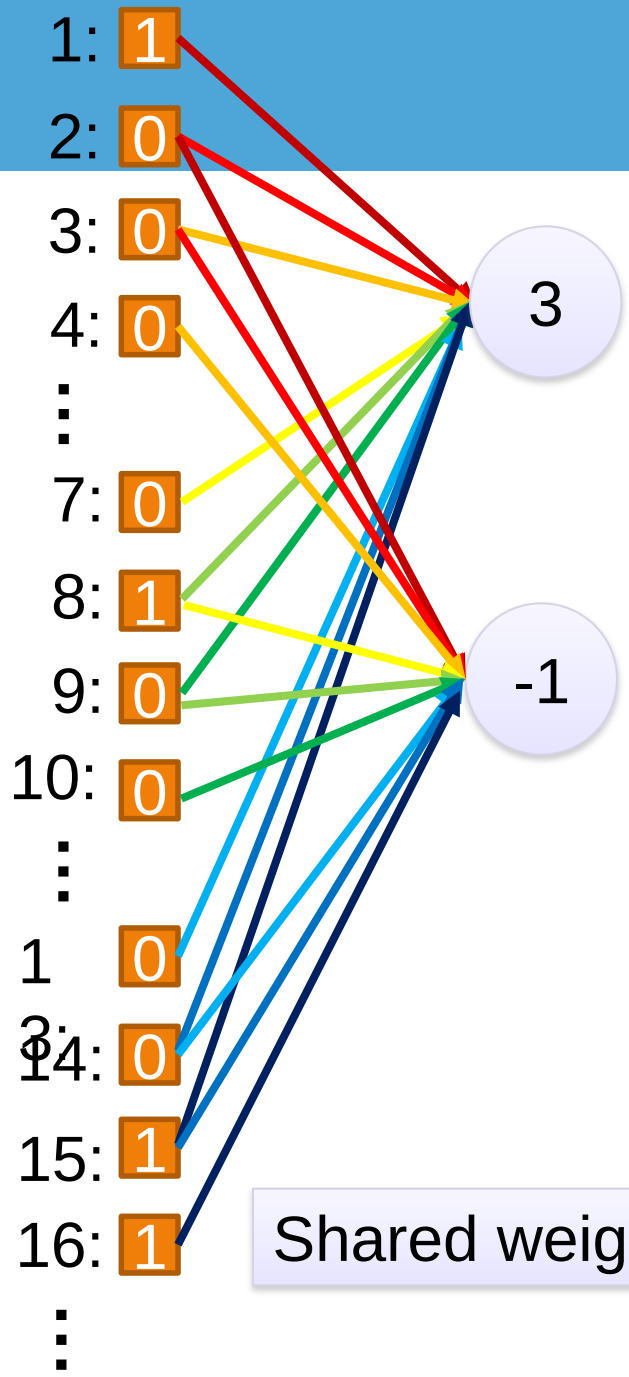


1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

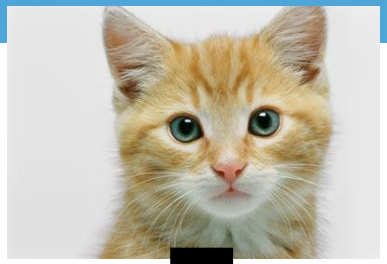
Fewer parameters

Even fewer parameters



Shared weights

The whole CNN



Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flattened

Fully Connected Feedforward network

cat dog ……

Max Pooling

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

Filter 1

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

Filter 2

3	-1
-3	1

-3	-1
0	-3

-3	-3
3	-2

0	1
-2	-1

-1	-1
-1	-1

-1	-1
-2	1

-1	-1
-1	0

-2	1
-4	3

Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

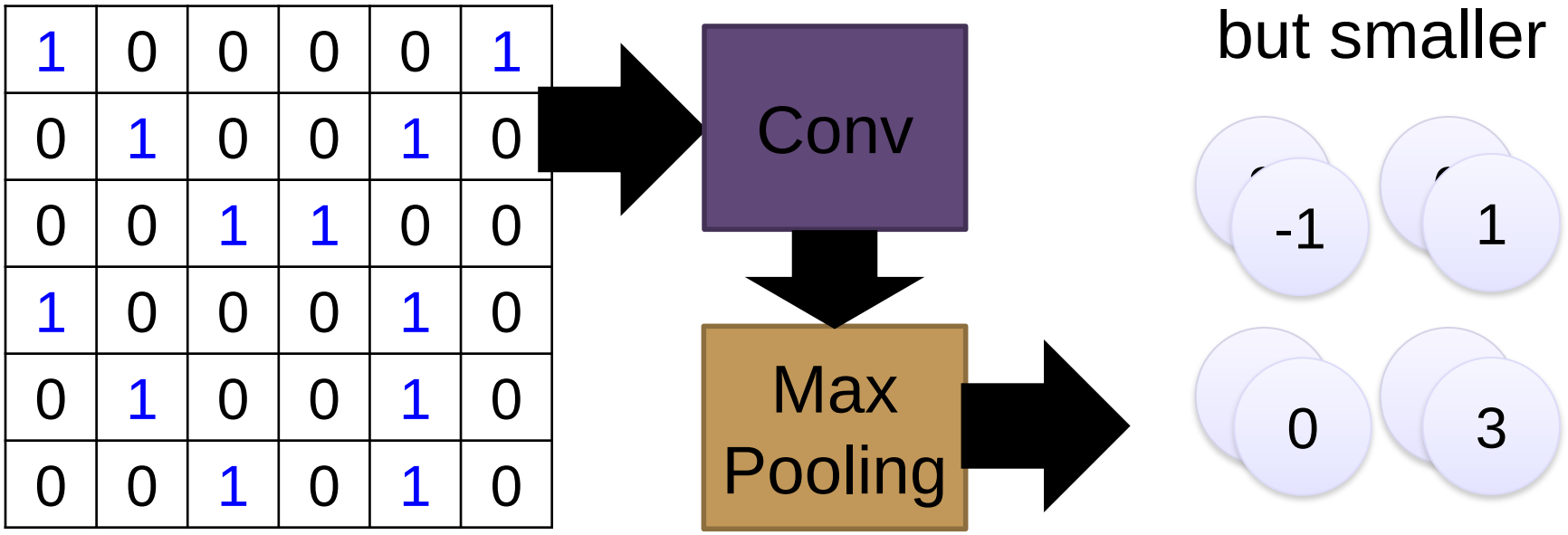
We can subsample the pixels to make image smaller

→ fewer parameters to characterize the image

CNN compression

- A CNN compresses a fully connected network in two ways
  - Reducing number of connections
  - Shared weights on the edges
  - Max pooling further reduces the complexity

Max Pooling


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
Conv
Max Pooling
New image but smaller
-1
1
0
3
6 x 6 image
2 x 2 image
Each filter is a channel

Why do max-pooling?

- Saves space
- Reduces overfitting?
- Because we add *more* convolutions after it!
  - Allows the short-range convolutions to extend over larger subfields of the images
    - So we can spot larger objects
    - Eg, a long horizontal line, or a corner, or …

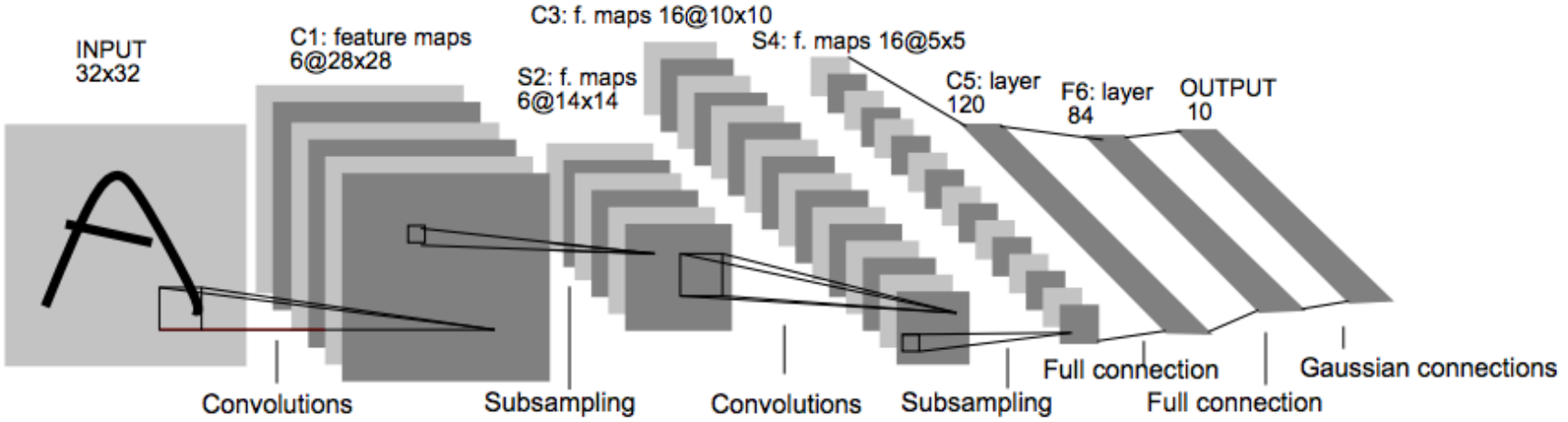


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Why do max-pooling (2)?

- At some point the feature maps start to get very sparse and blobby
  - indicators of some semantic property, not a recognizable transformation of the image
- Then just use them as features in a “normal” ANN

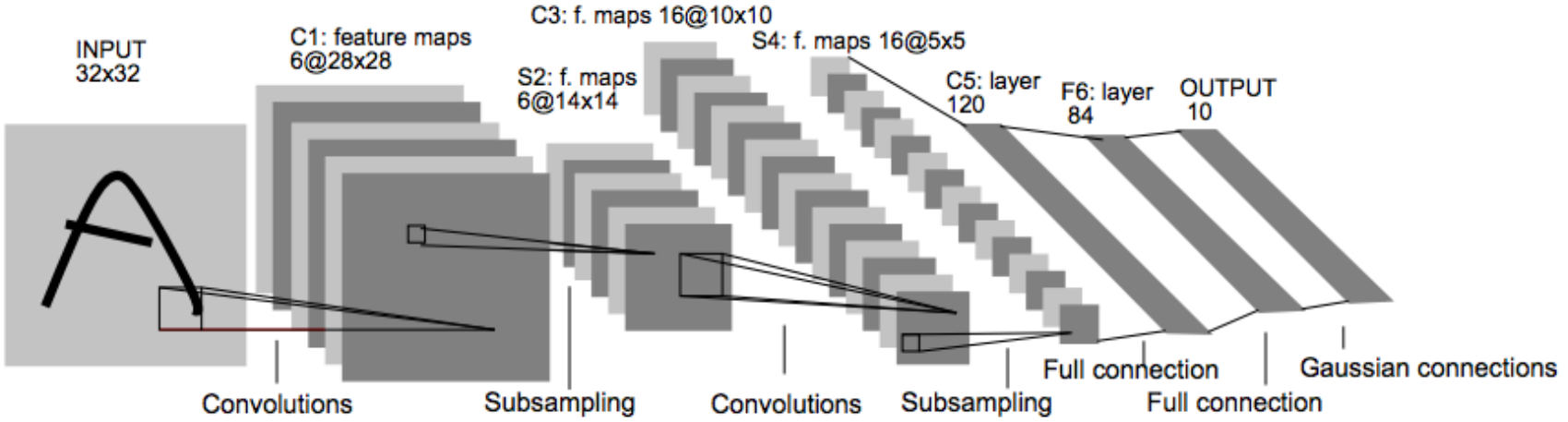


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

Network Visualization

input (24x24x1)
max activation: 0.99607, min: 0

Activations:

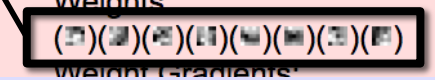
Weights:

filter size 5x5x1, stride 1
max activation: 2.96187, min: -5.43735
max gradient: 0.00068, min: -0.00102
parameters: 8x5x5x1+8 = 208

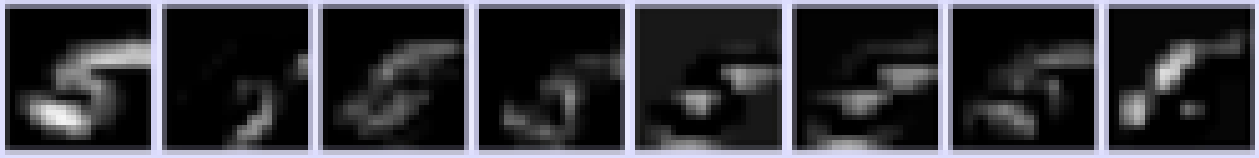
Activations:

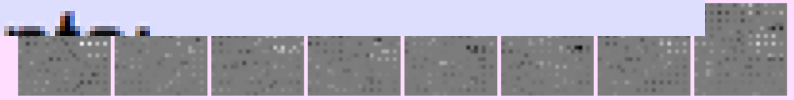
Activation Gradients:

Weights:



Activations:





pool (12x12x8)
pooling size 2x2, stride 2
max activation: 2.96187, min: 0
max gradient: 0.00106, min: -0.00102

Activations:

Activation Gradients:

conv (12x12x16)
filter size 5x5x8, stride 1
max activation: 5.58937, min: -11.45423
max gradient: 0.00053, min: -0.00106
parameters: 16x5x5x8+16 = 3216

Activations:

Activation Gradients:

Weights:

relu (12x12x16)
max activation: 5.58937, min: 0
max gradient: 0.0007, min: -0.0011

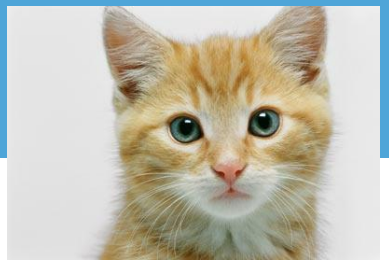
Activations:

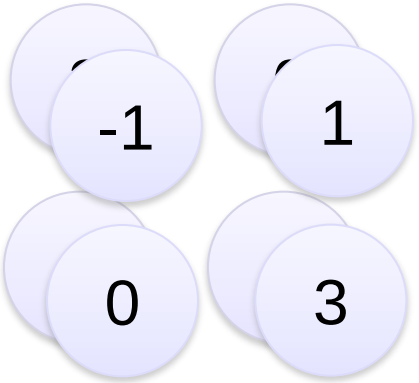
Activation Gradients:

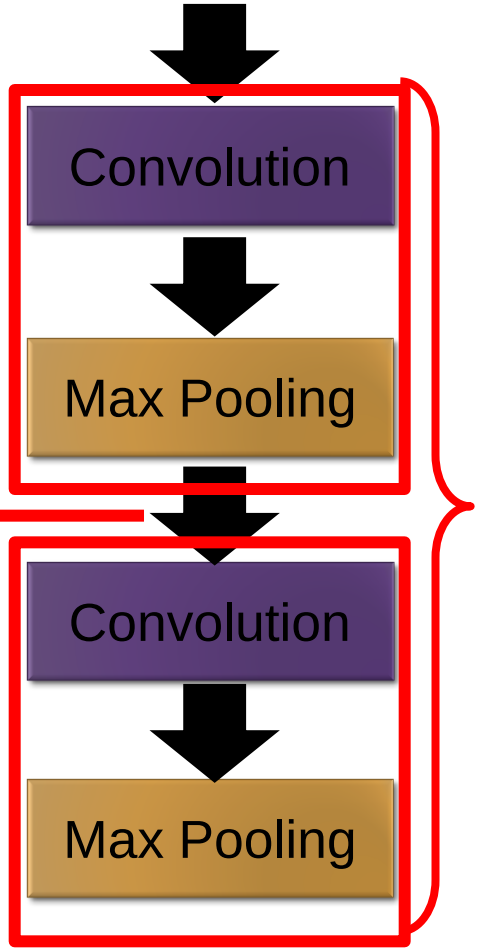
softmax (1x1x10)
max activation: 0.99864, min: 0
max gradient: 0, min: 0

Activations:

The whole CNN



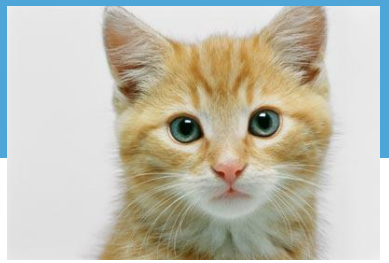




Smaller than the original image

The number of channels is the number of filters

The whole CNN



cat dog ……

Convolution

Max Pooling

A new image

Convolution

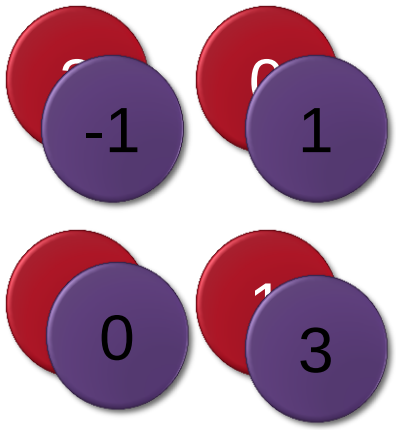
Max Pooling

A new image

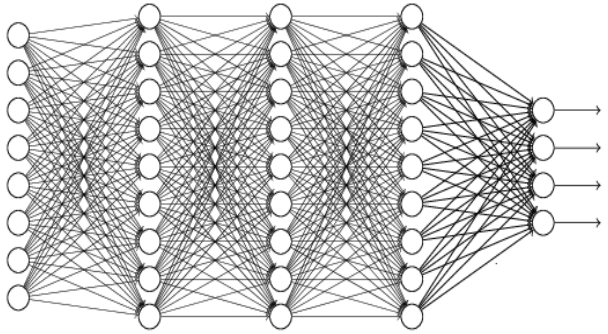
Flattened

Fully Connected Feedforward network

Flattening



Flattened

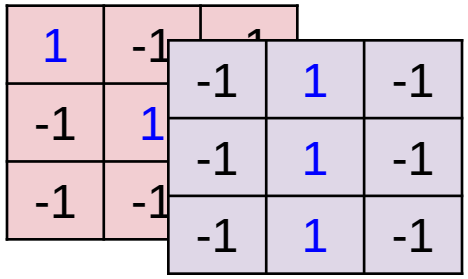


Fully Connected
Feedforward network

CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

```
model2.add( Convolution2D( 25,3,3,
                input_shape=(28,28,1)) )
```



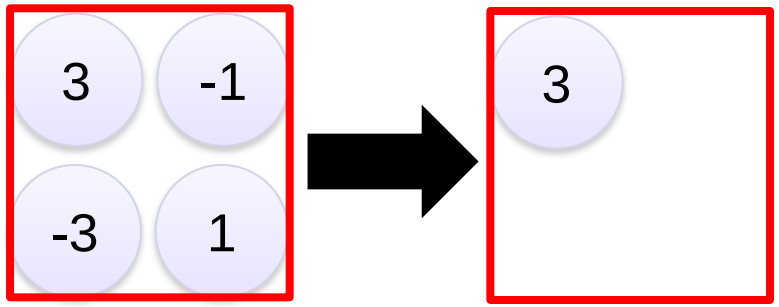
There are 25 3x3 filters.

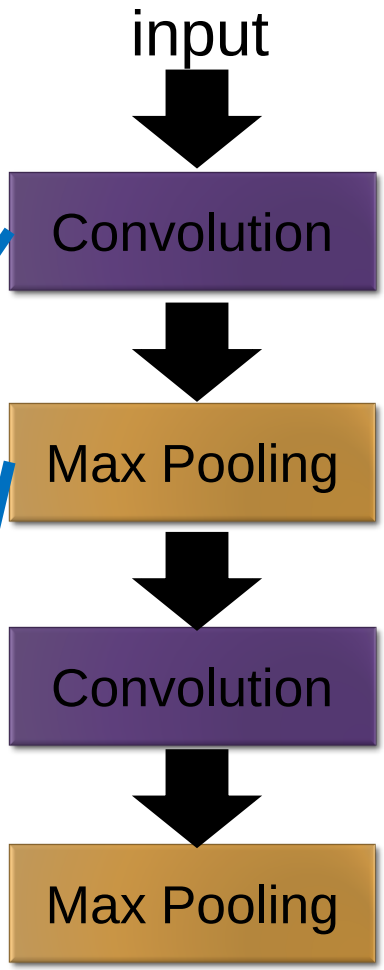
Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

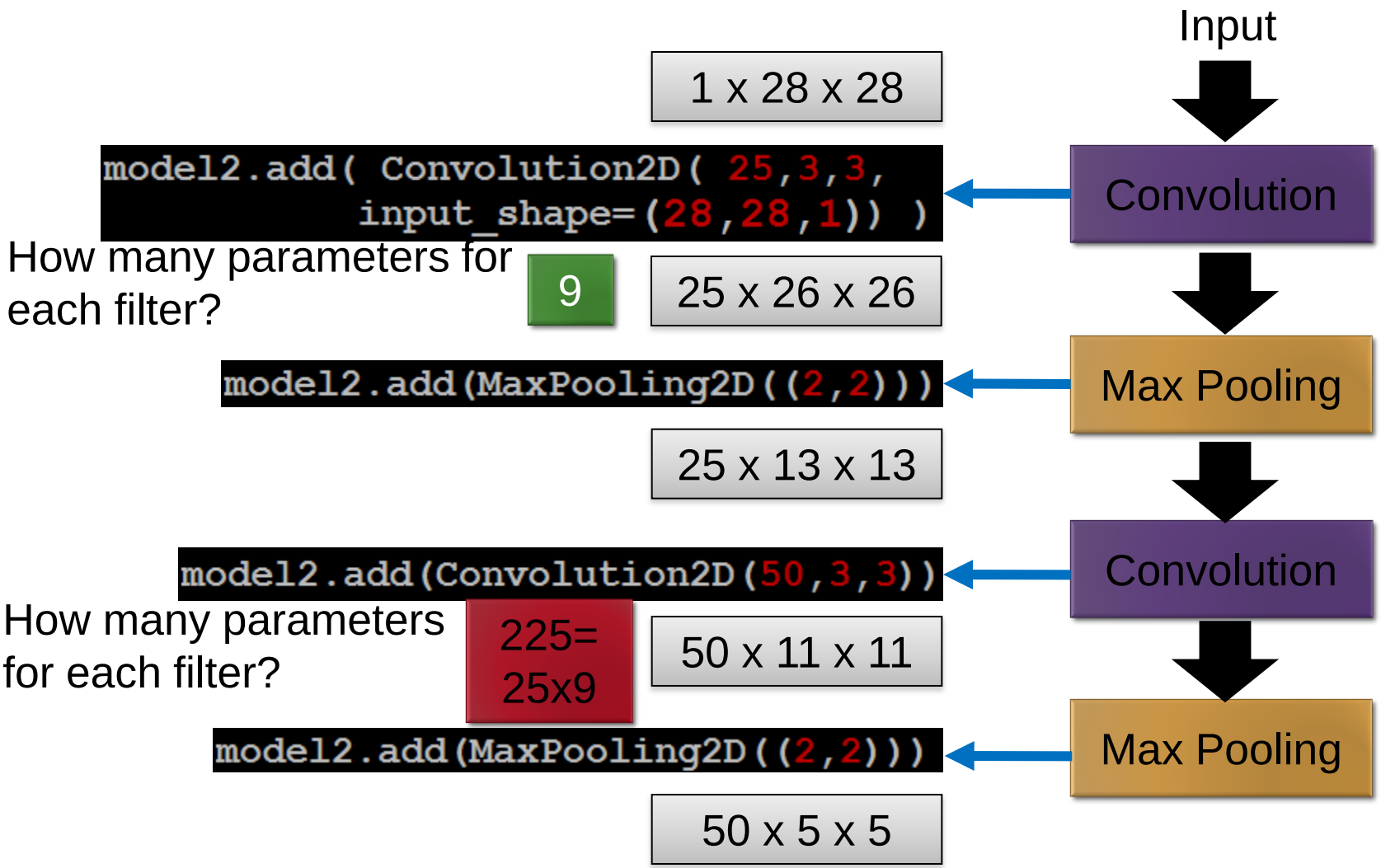
```
model2.add(MaxPooling2D((2,2)))
```





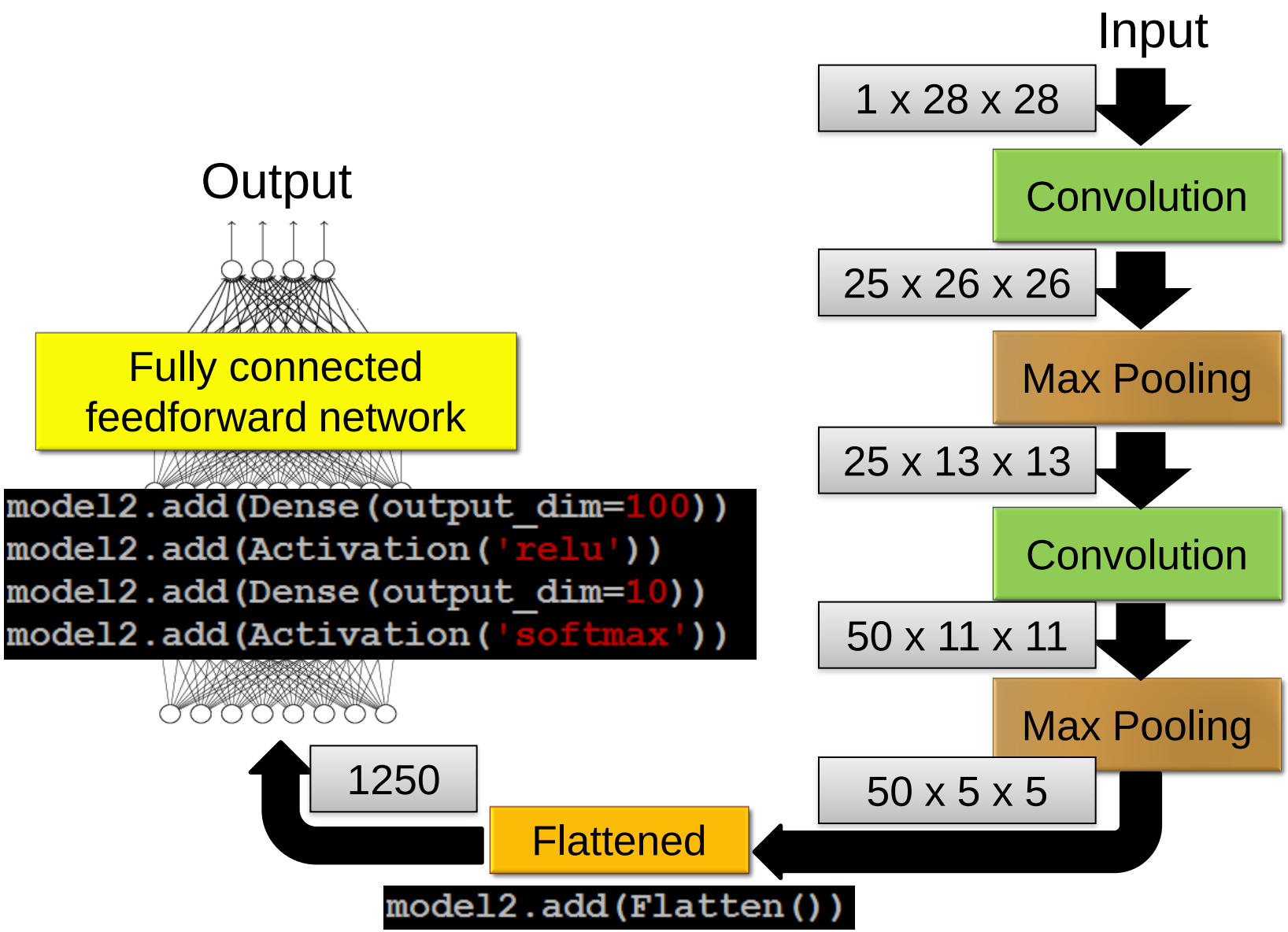
CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***



CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***



Contents

Convolutions: the basics

Convolutional networks

Interpreting convolutional networks

Common applications

Discrete convolutions

Focus on 1 & 2-D signals but signal dimensionality is arbitrary (usually 1,2,3,4-D)

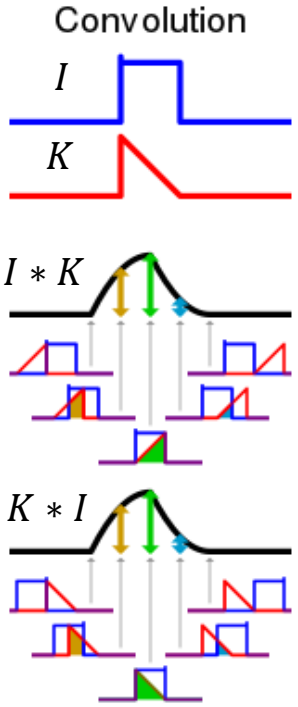


Image by Cmglee - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20206883

The 2-D discrete convolution of two signals I and K is defined as:

$$(I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n)$$

$$= \sum_m \sum_n I(i-m,j-n)K(m,n)$$

Where $-\infty \leq m,n \leq \infty$. Finite signals can be extended by adding zeros (more on this later)

Convolution vs Cross-correlation

$$(I * K)(i,j) = \\ = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

vs

$$(I \star K)(i,j) = \\ = \sum_m \sum_n I(m,n)K(i+m, j+n)$$

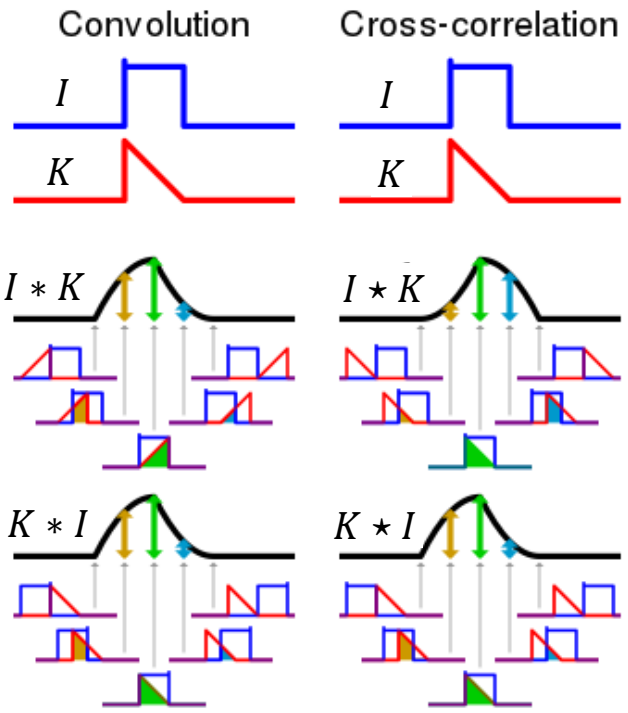


Image by Cmglee - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=20206883

Cross-correlation quantifies **presence of $I(i,j)$ in (shifted) $K(i,j)$**

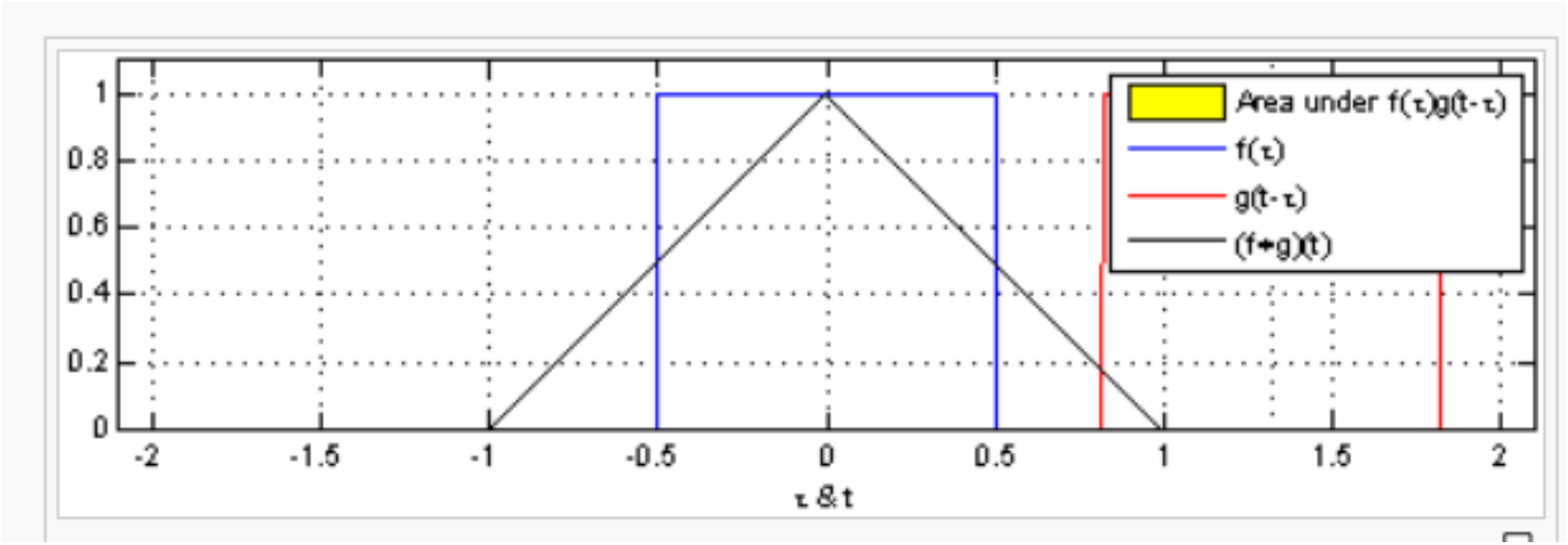
Most ML libraries implement convolutional layers as cross-correlation layers.

- Commutativity not important for most ConvNets
- Can avoid flipping one of the signals ⇒ **easier to implement**

Continuous convolution

https://en.wikipedia.org/wiki/Convolution

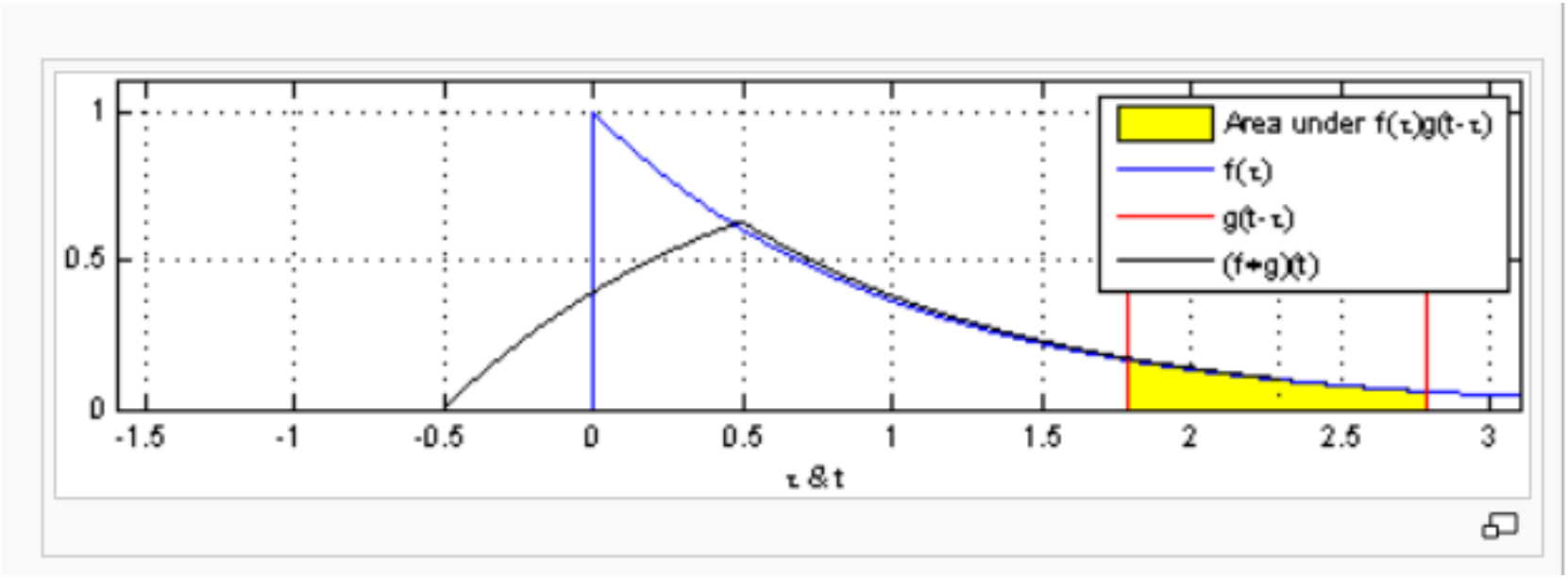
1-D $$(f * g)(t) \ \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$
$$= \int_{-\infty}^{\infty} f(t-\tau)\, g(\tau)\, d\tau.$$



Continuous convolution

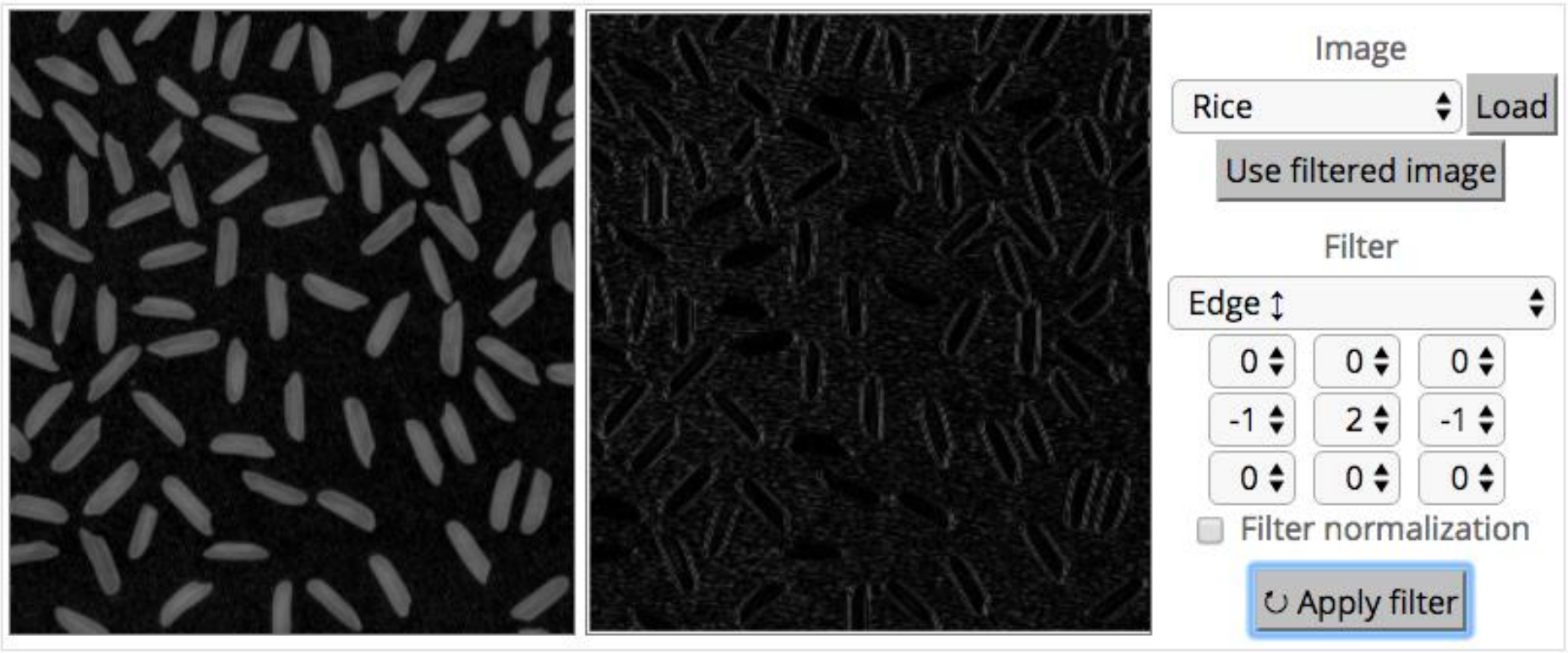
https://en.wikipedia.org/wiki/Convolution

1-D $$(f * g)(t) \ \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$
$$= \int_{-\infty}^{\infty} f(t-\tau)\, g(\tau)\, d\tau.$$



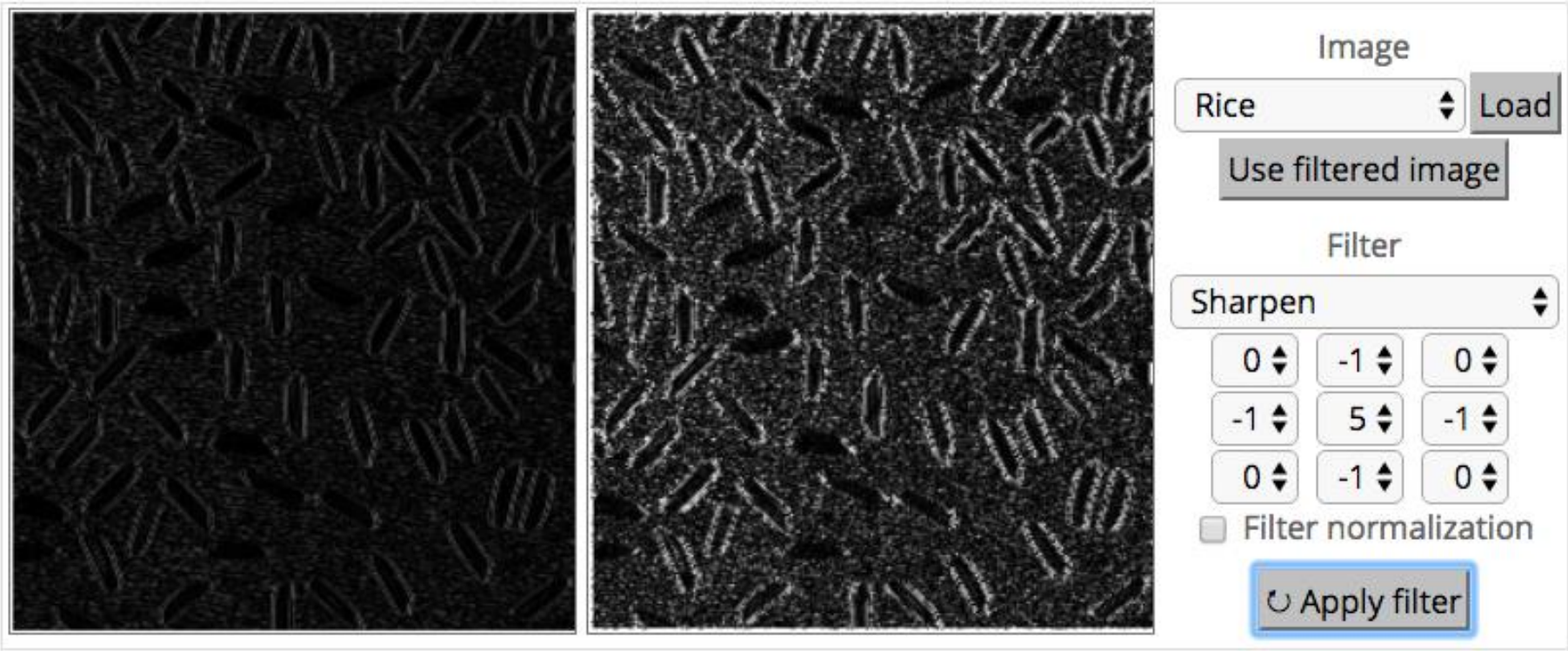
What's a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



What's a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo

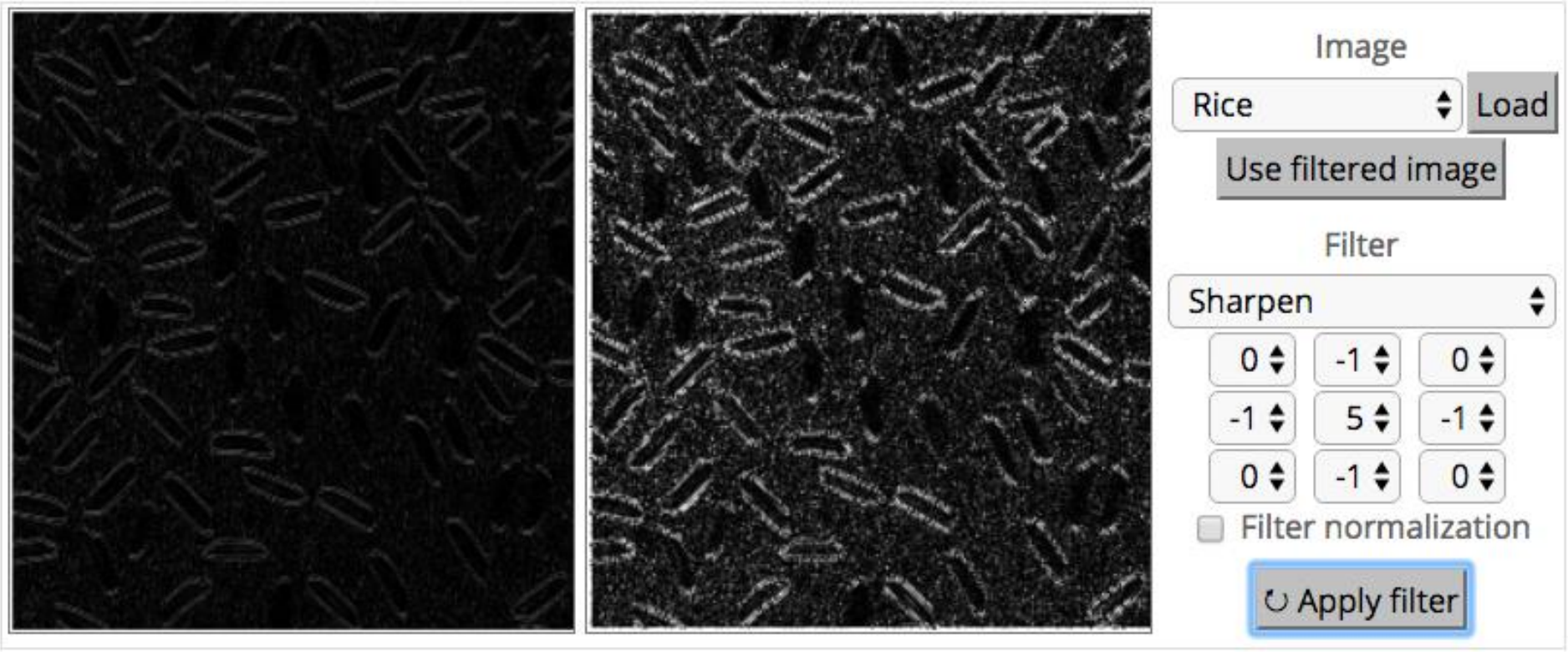


What's a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo

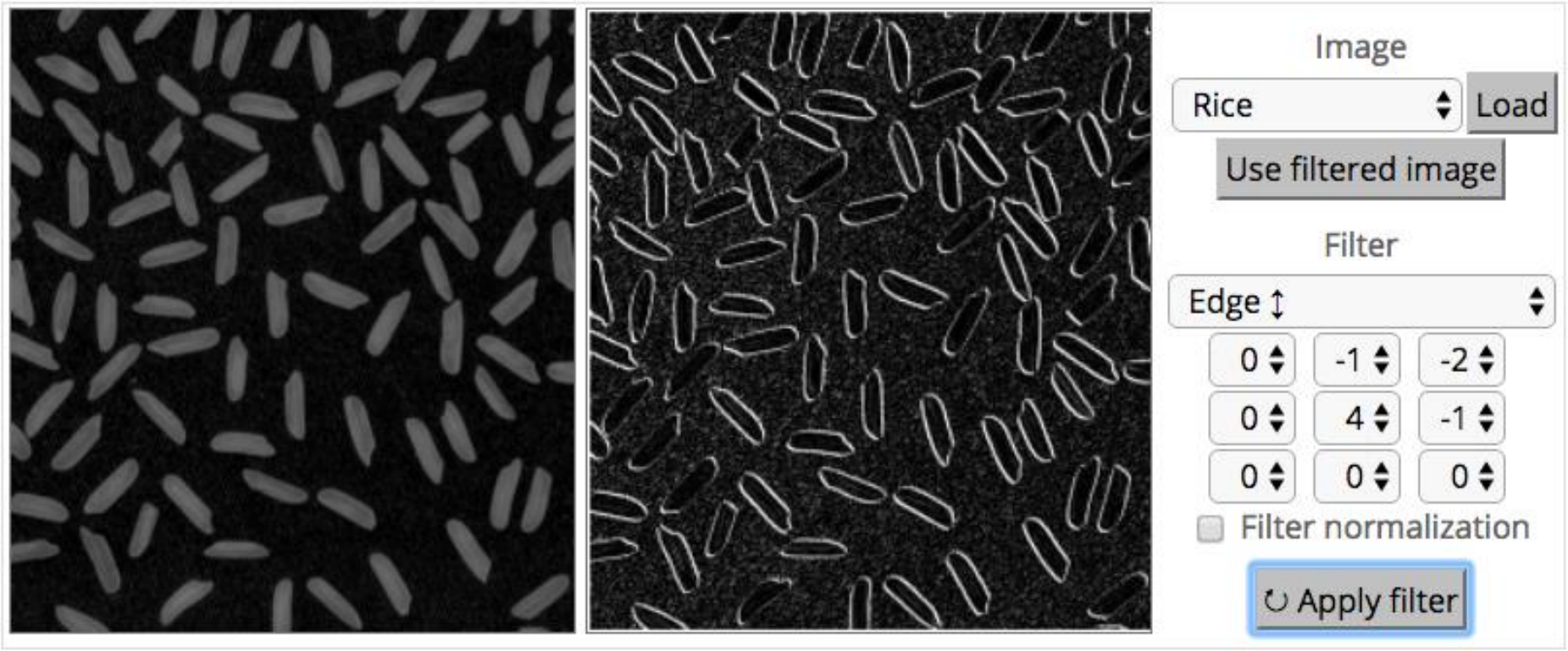
What's a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



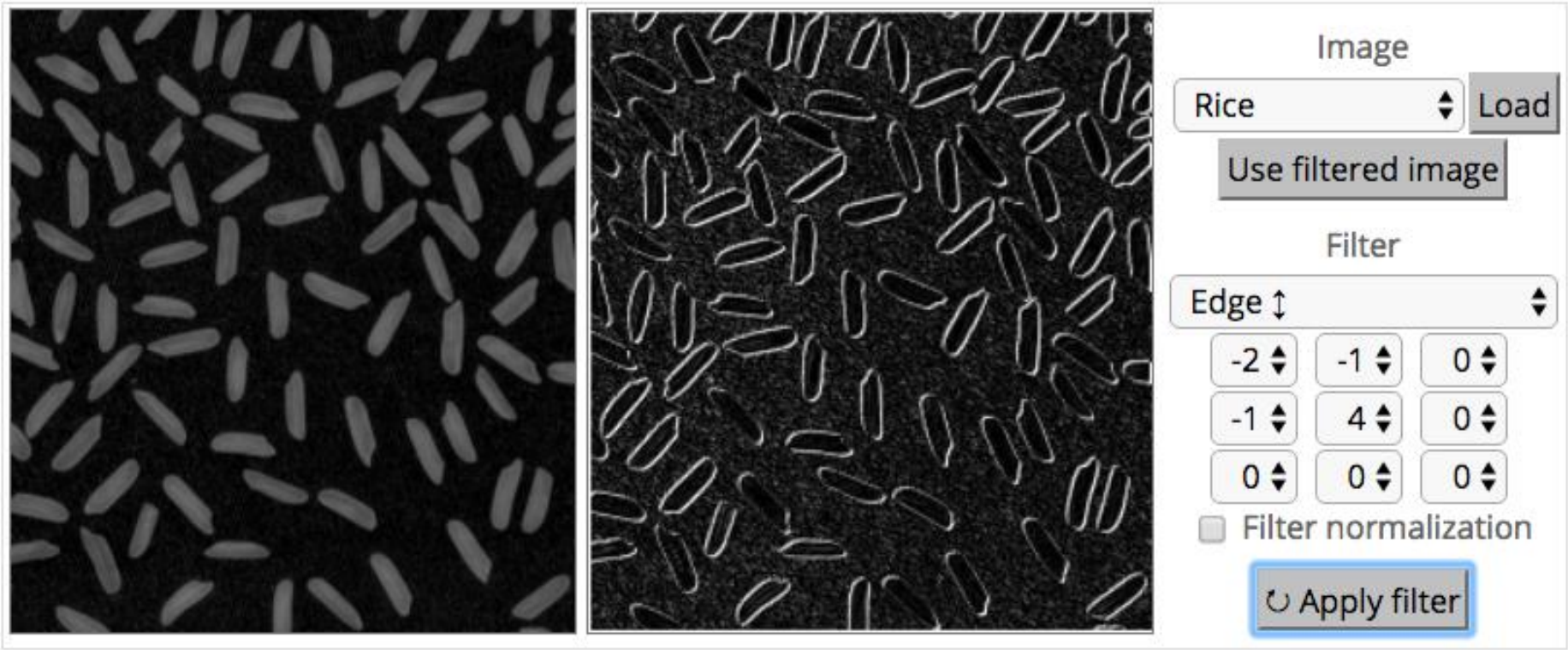
What's a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



What's a convolution?

http://matlabtricks.com/post-5/3x3-convolution-kernels-with-online-demo



Summary