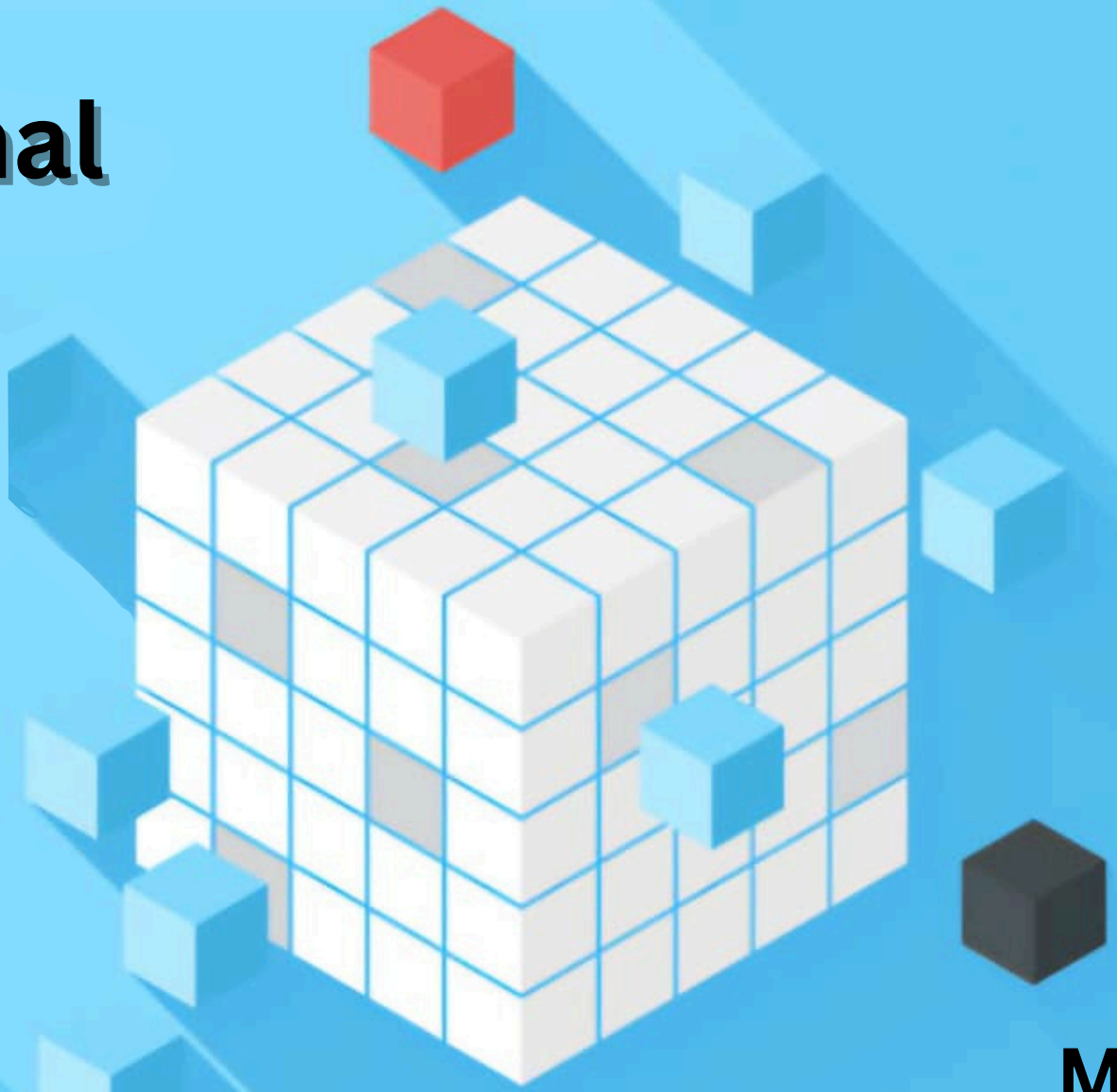
Convolutional Neural Networks (CNN)



Members

Rishav Singh - 1032233012

Kunal Tailor - 1032233258

Anuj Kudu - 1032233203

Yash Pathrikar - 1032233341

ML-C2-10

Neural Network

Here x1 and x2 are normalized attribute value of data.

y is the output of the neuron , i.e the class label.

x1 and x2 values multiplied by weight values w1 and w2 are input to the neuron x.

Value of x1 is multiplied by a weight w1 and values of x2 is multiplied by a weight w2.

Given that

- w1 = 0.5 and w2 = 0.5
- Say value of x1 is 0.3 and value of x2 is 0.8,

- So, weighted sum is :
- sum= w1 x x1 + w2 x x2 = 0.5 x 0.3 + 0.5 x 0.8 = 0.55

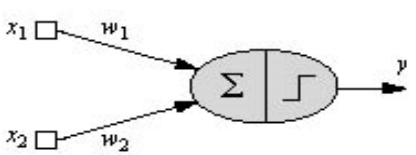
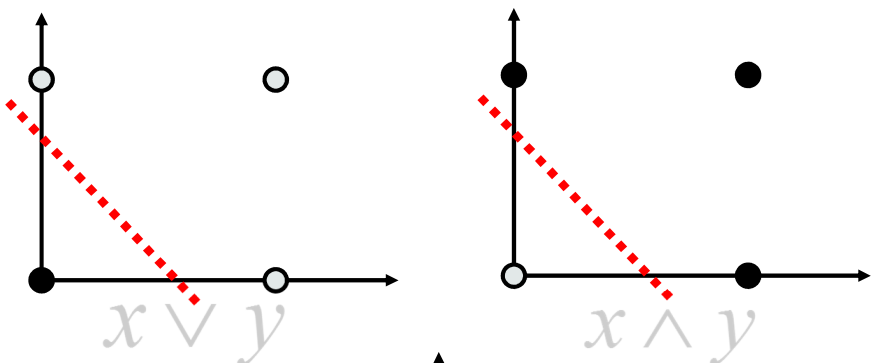


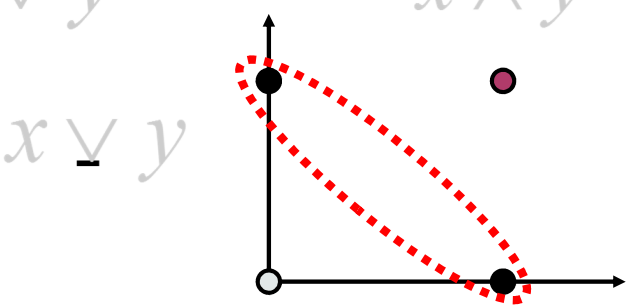
Fig1: an artificial neuron

Why We Need Multi Layer ?

Linear Separable:

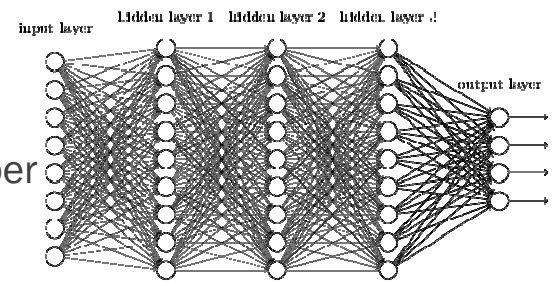


Linear inseparable:

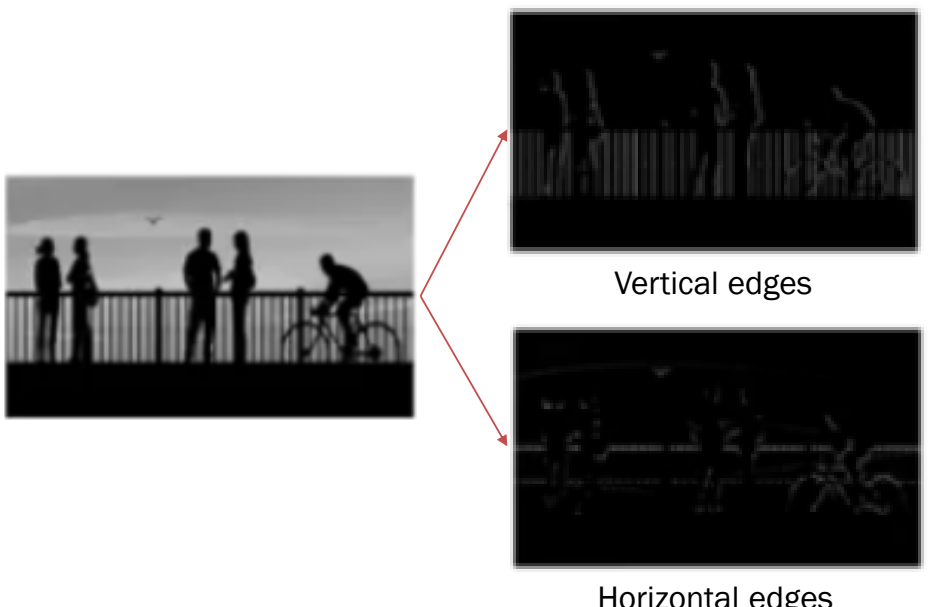


Neural Network?

- Suppose an image is of the size 68 X 68 X 3
  - Input feature dimension then becomes 12,288
- If Image size is of 720 X 720 X 3
  - Input feature dimension becomes 1,555,200
- Number of parameters will swell up to a HUGE number
- Result in more computational and memory requirements



Edge Detection

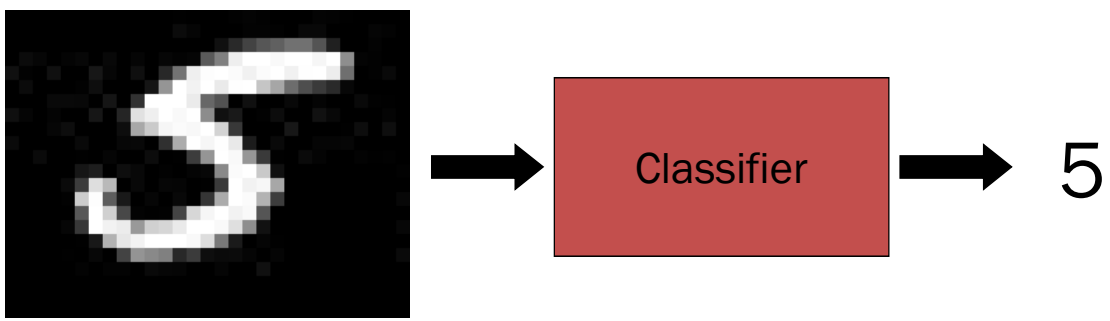


How do we detect these edges

Image Source: deeplearning.ai

Another Application

Digit Recognition



$X_1,\ldots,X_n \in \{0,1\}$ (Black vs. White pixels)

$Y \in \{5,6\}$ (predict whether a digit is a 5 or a 6)

The Bayes Classifier

In class, we saw that a good strategy is to predict:

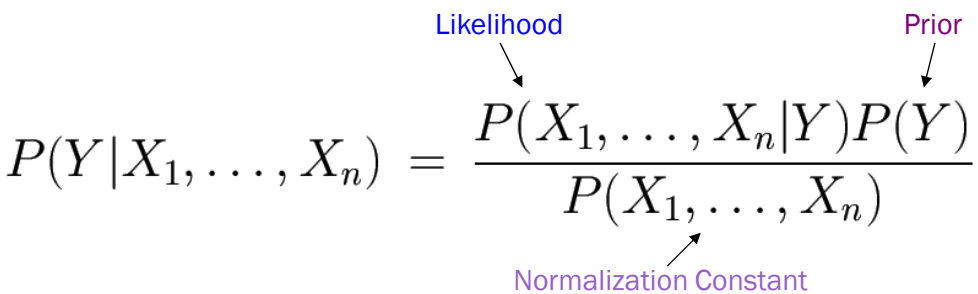
$$\arg\max_Y P(Y|X_1,\ldots,X_n)$$

- (for example: what is the probability that the image represents a 5 given its pixels?)

So … how do we compute that?

The Bayes Classifier

Use Bayes Rule!



Why did this help? Well, we think that we might be able to specify how features are "generated" by the class label

The Bayes Classifier

Let's expand this for our digit recognition task:

$$P(Y=5|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y=5)P(Y=5)}{P(X_1,\ldots,X_n|Y=5)P(Y=5)+P(X_1,\ldots,X_n|Y=6)P(Y=6)}$$

$$P(Y=6|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n|Y=6)P(Y=6)}{P(X_1,\ldots,X_n|Y=5)P(Y=5)+P(X_1,\ldots,X_n|Y=6)P(Y=6)}$$

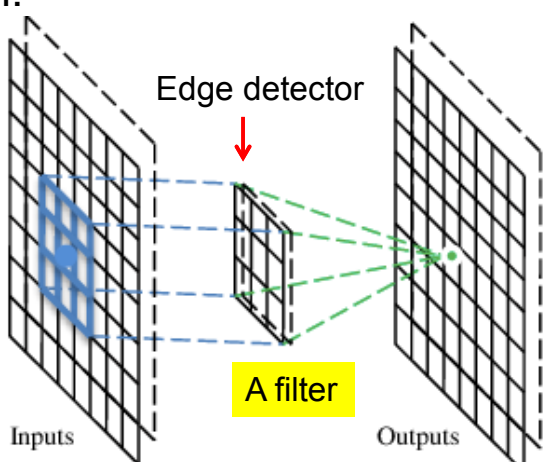
To classify, we'll simply compute these two probabilities and predict based on which one is greater

Drive into CNN

In a convolutional network (ConvNet), there are basically three types of layers:
1.Convolution layer
2.Pooling layer
3.Fully connected layer

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 does convolutional operation.



Convolution

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

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

Filter 2

⋮ ⋮

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

Image Source: internet

Convolution

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

Filter 1

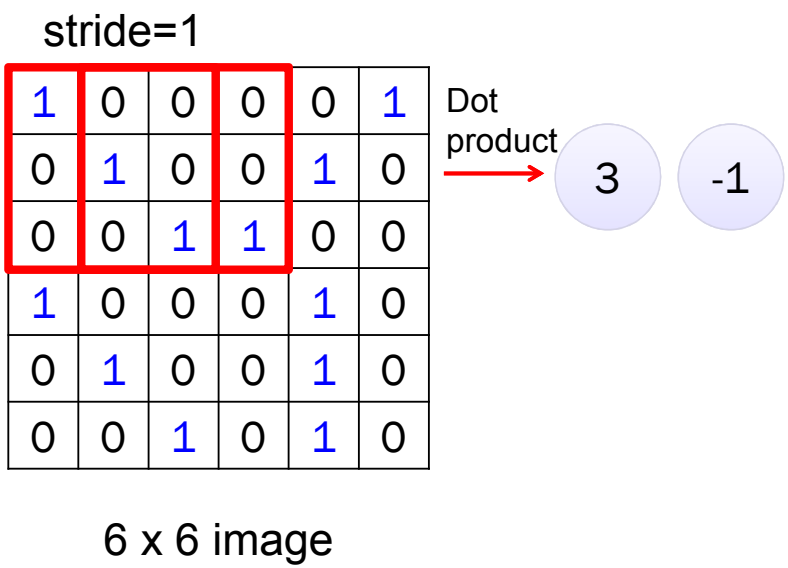
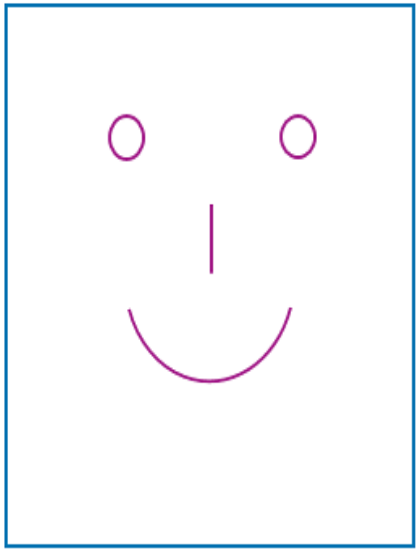
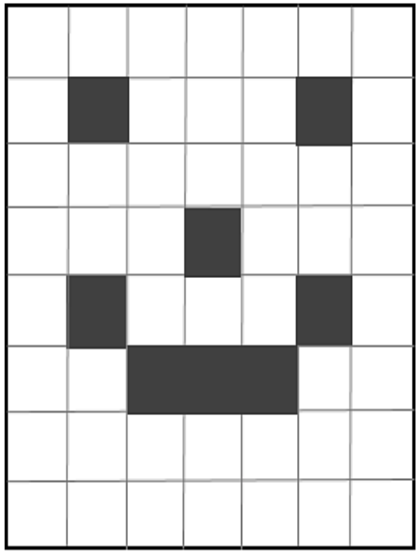


Image Source: internet

CONTD..



Real Image

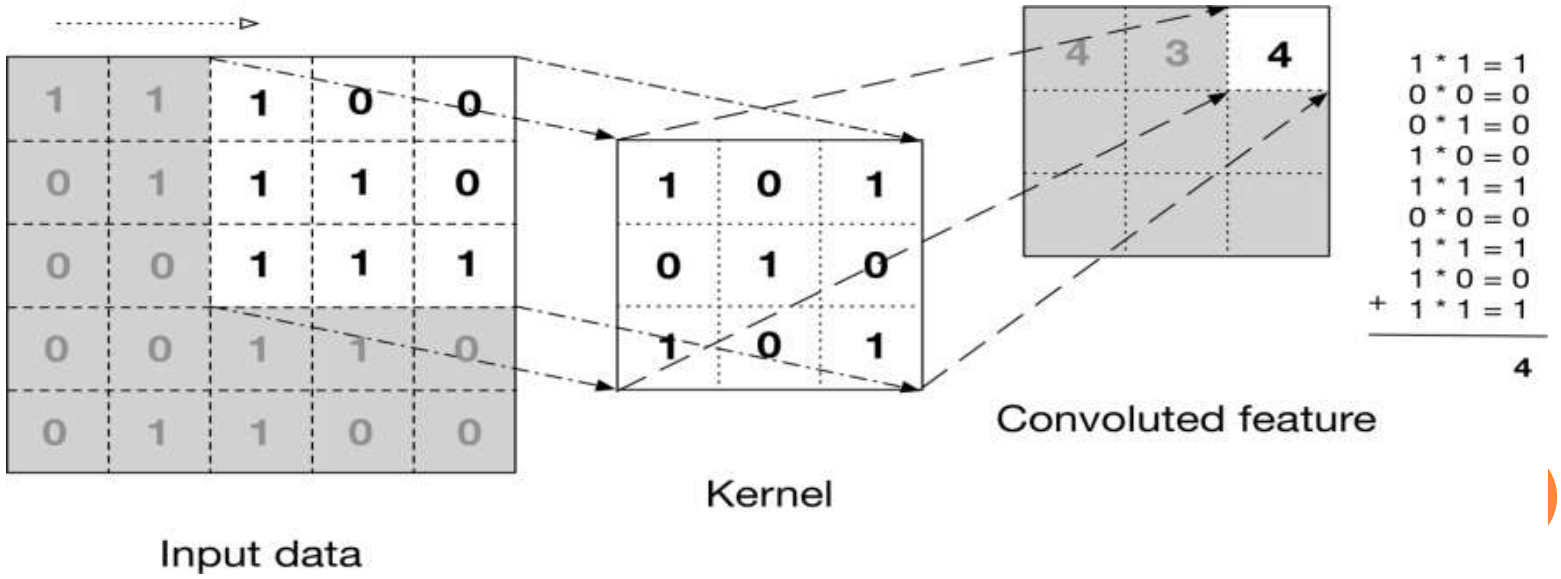


Represented in the form of black and white pixels

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

Image represented in the form of a matrix of numbers

- A feature detector, also known as a kernel or a filter, will traverse over the image's receptive fields, checking for the presence of the feature.
- For simplicity, let's stick with grayscale images as we try to **understand how CNNs work.**



CONTD..

- This implies the **input will have three dimensions**:
  - **height, width, and depth**, which match the **RGB color space of a picture.** Here we try to decompose RGB to a multidimensional layer and apply a filter to each layer.
- The dimension of the image matrix **is h×w×d.**
- The dimension of the filter is **fh×fw×d.**

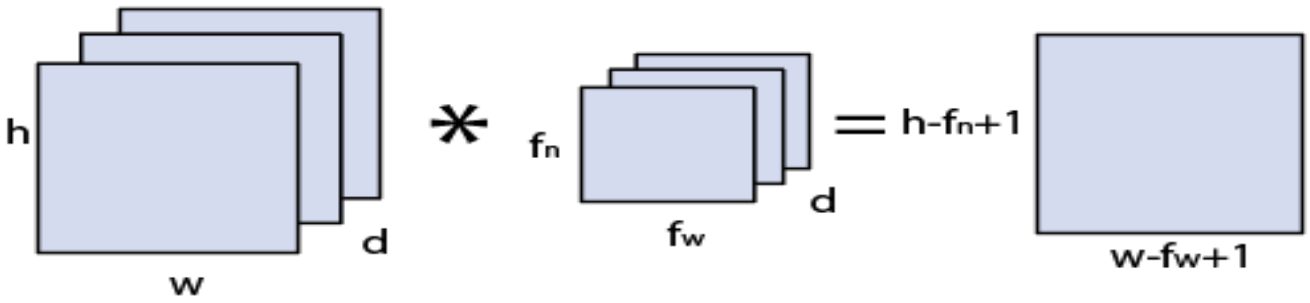
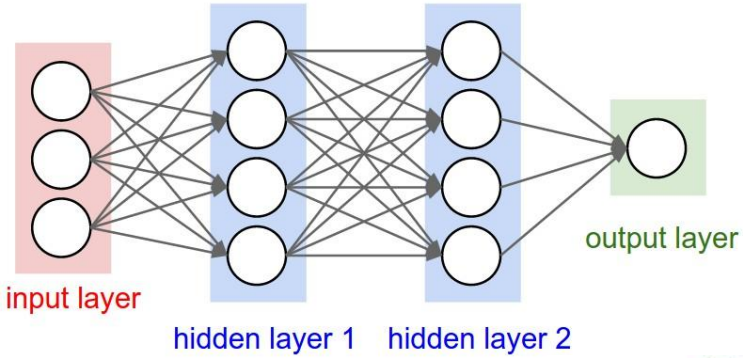


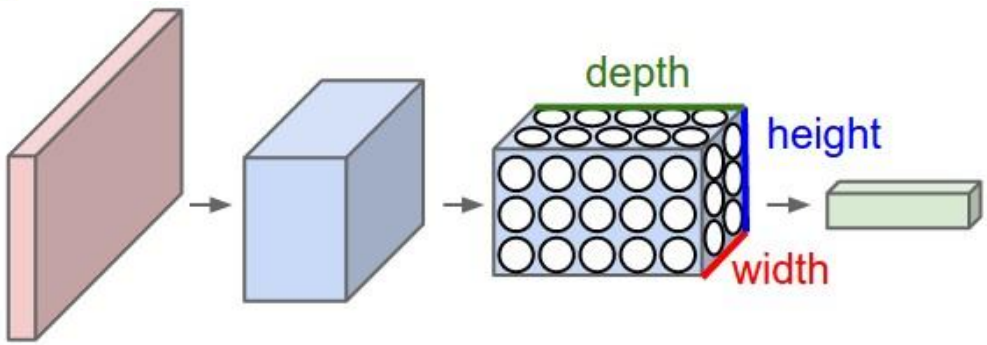
Image matrix multiplies kernl or filter matrix

Convnets



Layers used to build ConvNets:

- a stacked sequence of layers. 3 main types
- Convolutional Layer, Pooling Layer, and Fully-Connected Layer

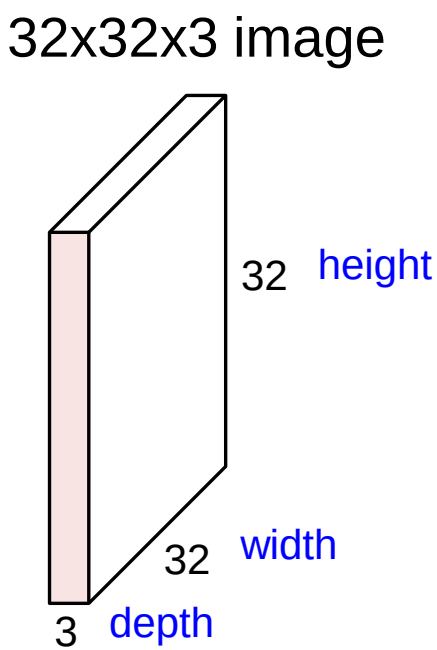


- every layer of a ConvNet transforms one volume of activations to another through a differentiable function.

Convolution Layer

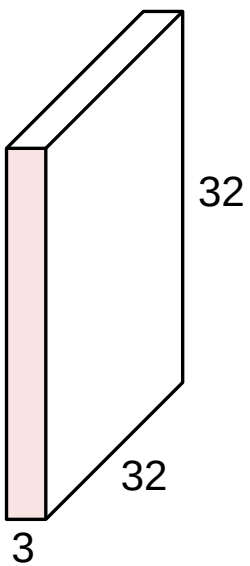
- The Conv layer is the core building block of a CNN
- The parameters consist of a set of learnable filters.
- Every filter is small spatially (width and height), but extends through the full depth of the input volume, eg, 5x5x3
- During the forward pass, we slide (convolve) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position.
- produce a 2-dimensional activation map that gives the responses of that filter at every spatial position.
- Intuitively, the network will learn filters that activate when they see some type of visual feature
- A set of filters in each CONV layer
  - each of them will produce a separate 2-dimensional activation map
  - We will stack these activation maps along the depth dimension and produce the output volume.

Convolutions: More detail

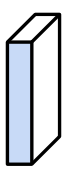


Convolutions: More detail

32x32x3 image



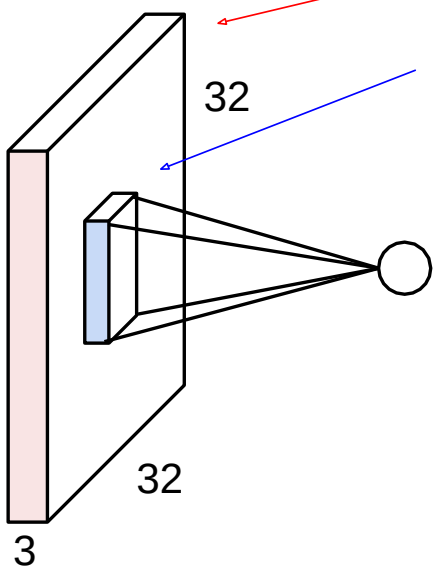
5x5x3 filter



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

Convolutions: More detail

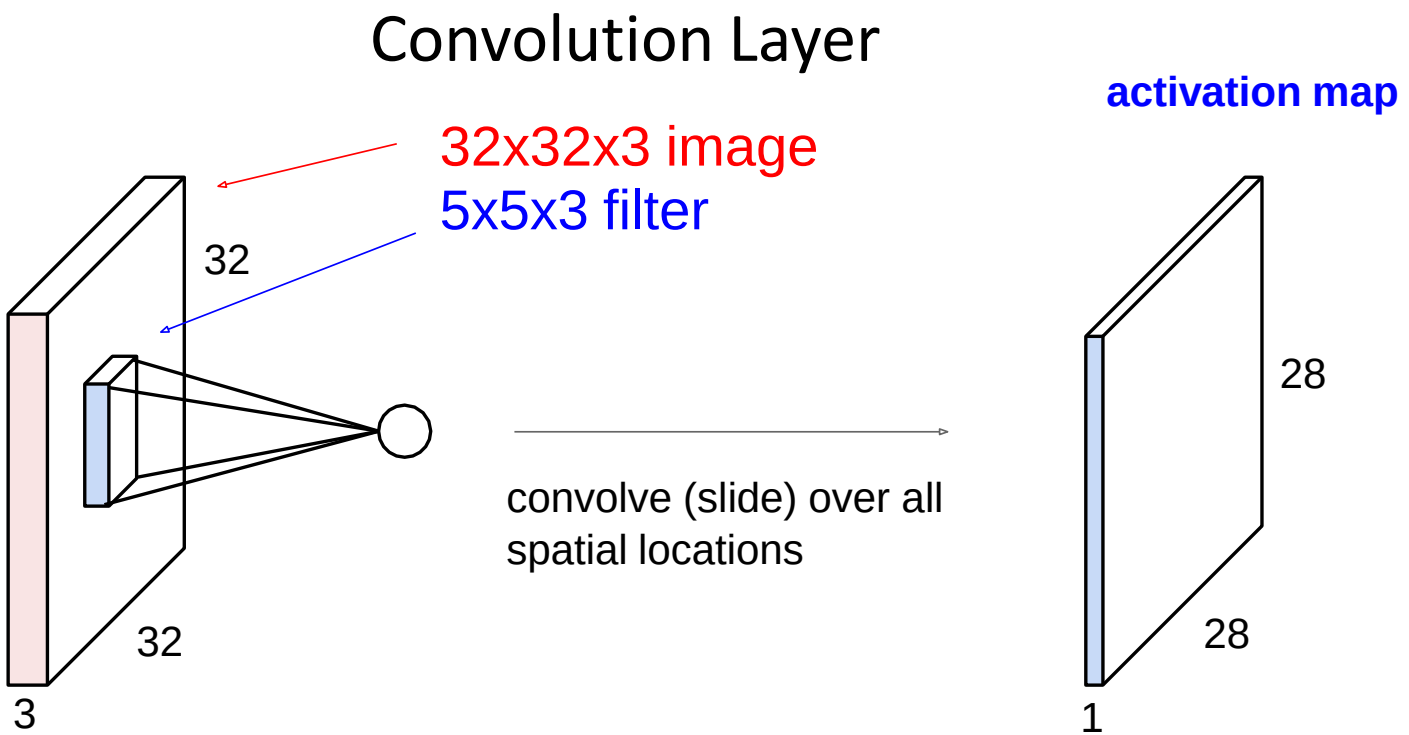
Convolution Layer



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Convolutions: More detail



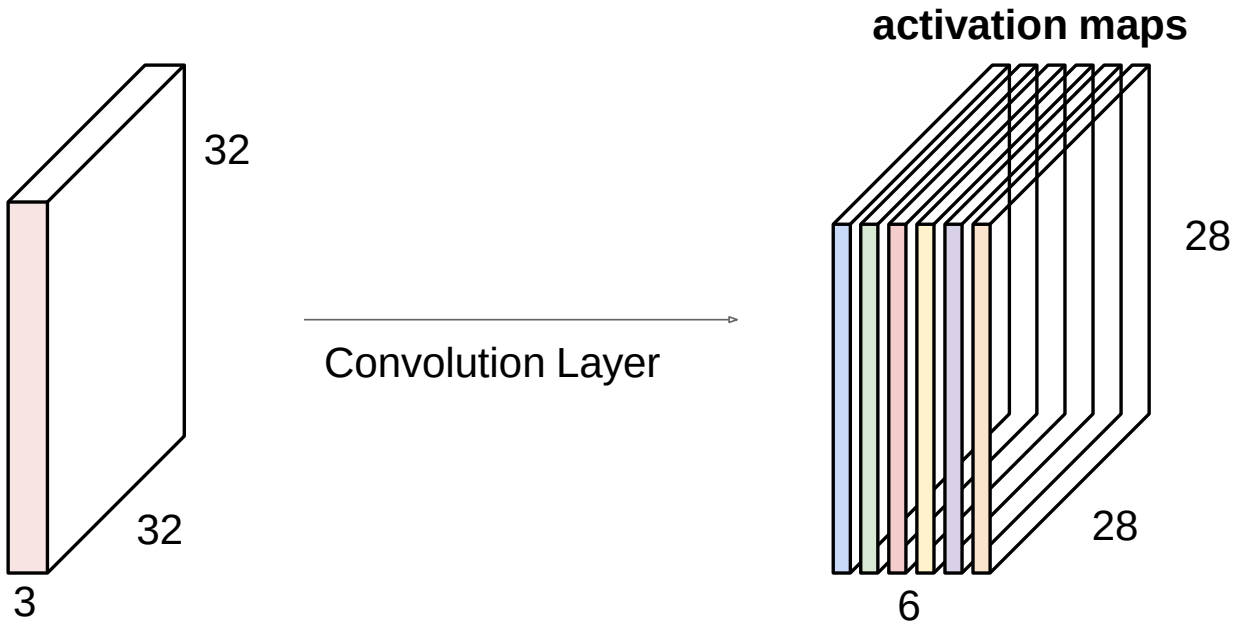
Convolutions: More detail

Convolution Layer

consider a second, green filter

Convolutions: More detail

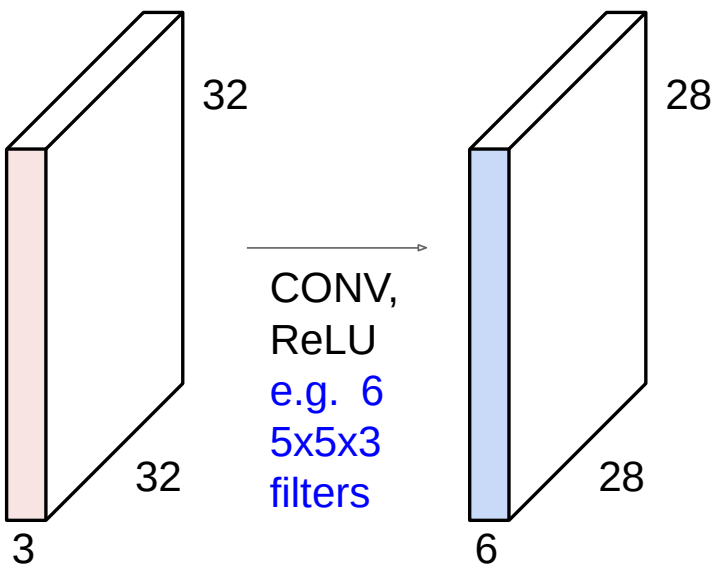
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

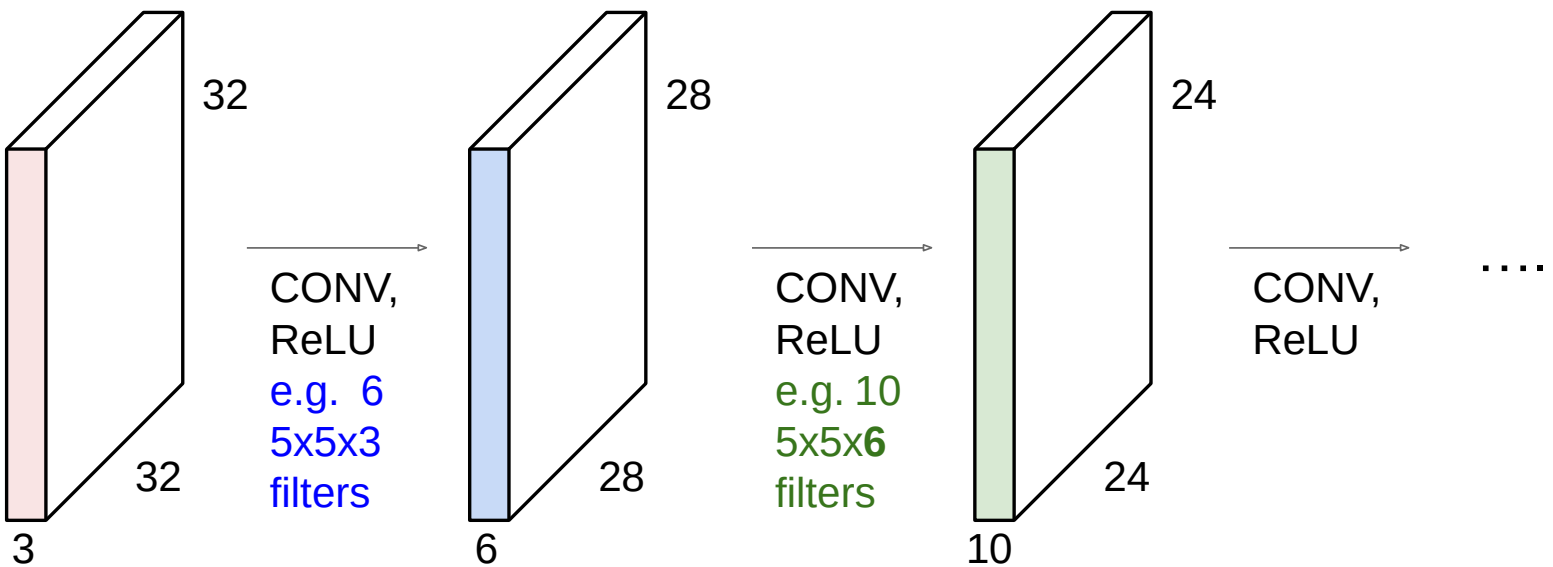
Convolutions: More detail

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



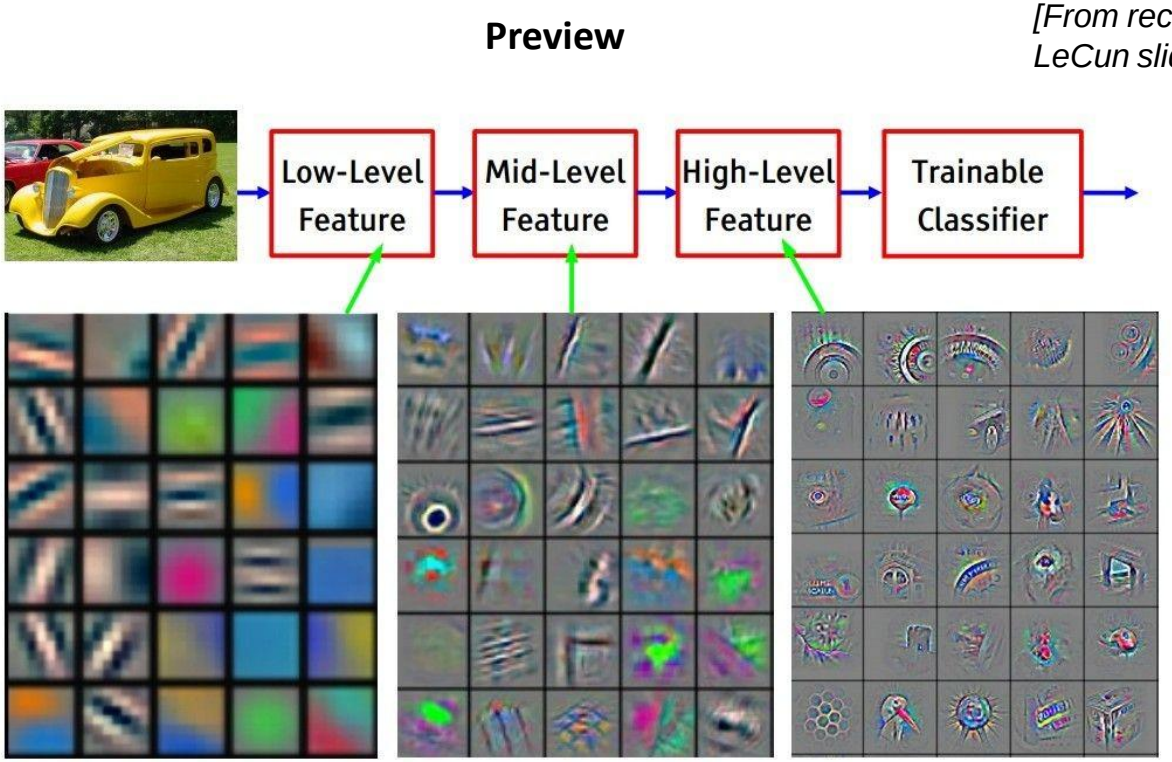
Convolutions: More detail

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



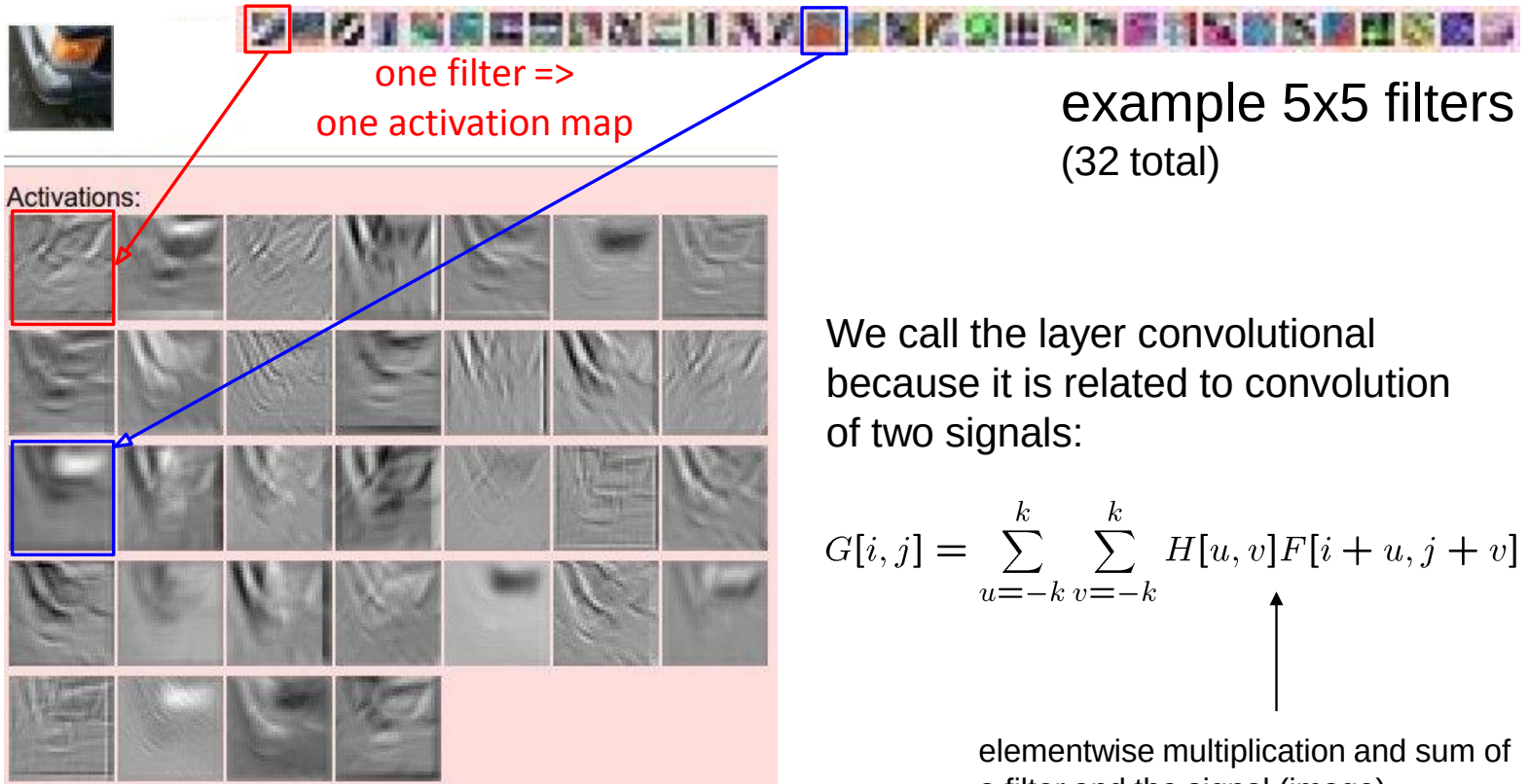
Convolutions: More detail

[From recent Yann LeCun slides]



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

Convolutions: More detail



example 5x5 filters
(32 total)

We call the layer convolutional because it is related to convolution of two signals:

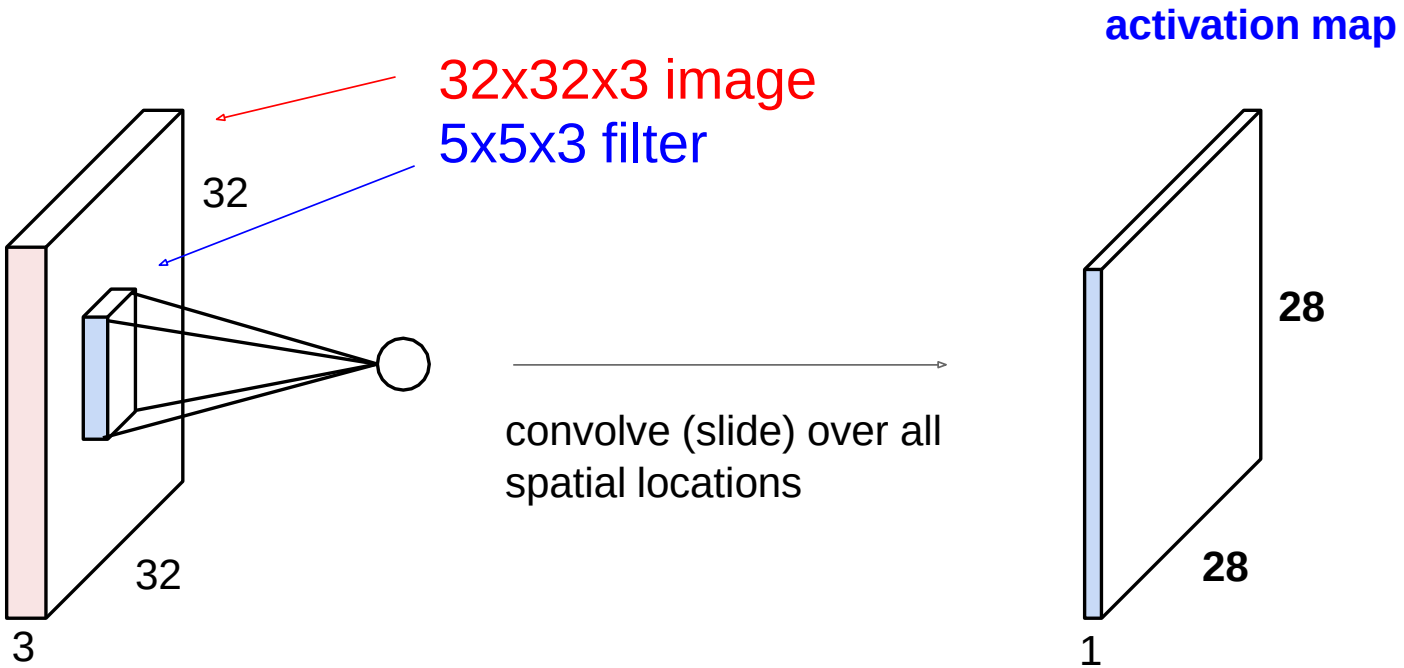
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

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

Adapted from Andrej Karpathy, Kristen Grauman

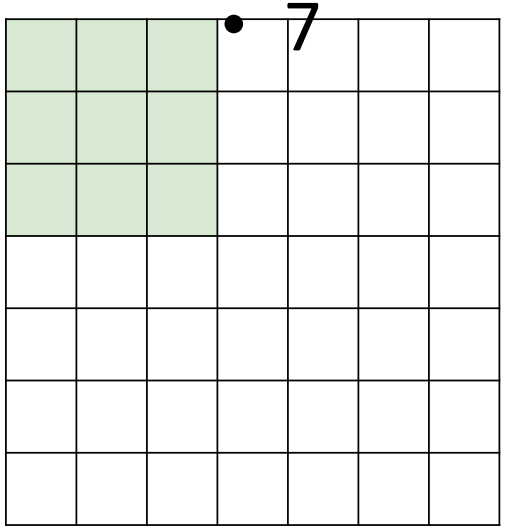
Convolutions: More detail

A closer look at spatial dimensions:



Convolutions: More detail

A closer look at spatial dimensions:

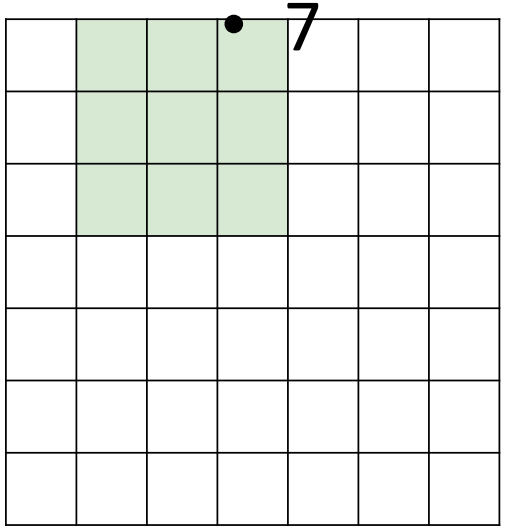


- 7
- 7

- 7x7 input (spatially) assume 3x3 filter

Convolutions: More detail

A closer look at spatial dimensions:

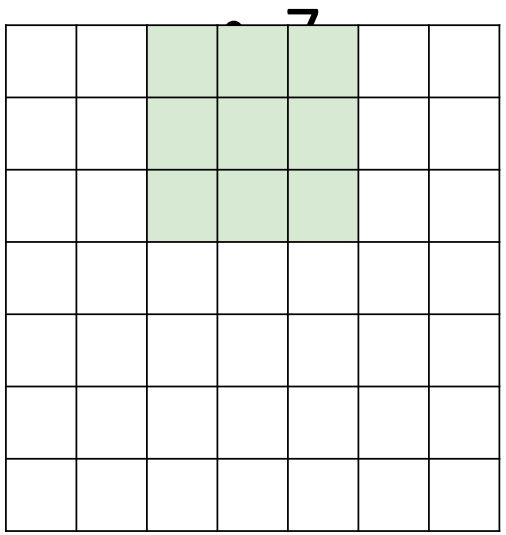


- 7
- 7

- 7x7 input (spatially) assume 3x3 filter

Convolutions: More detail

A closer look at spatial dimensions:

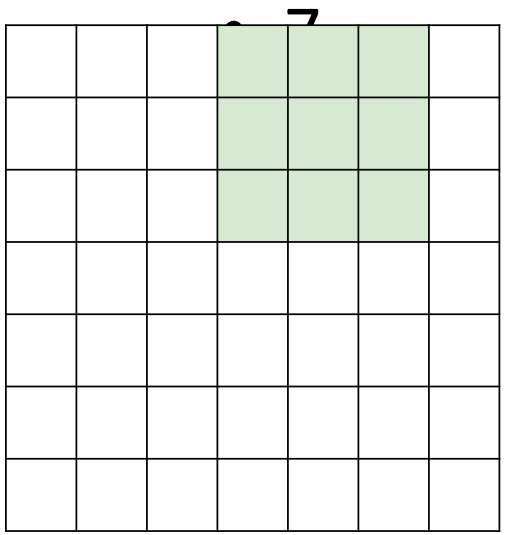


- 7x7 input (spatially) assume 3x3 filter

- 7

Convolutions: More detail

A closer look at spatial dimensions:

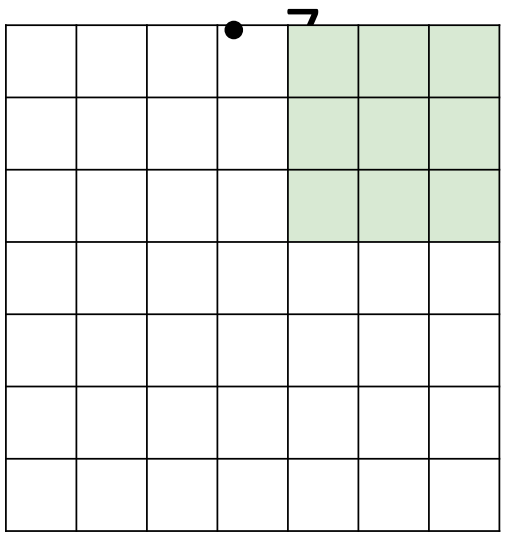


- 7x7 input (spatially) assume 3x3 filter

- 7

Convolutions: More detail

A closer look at spatial dimensions:

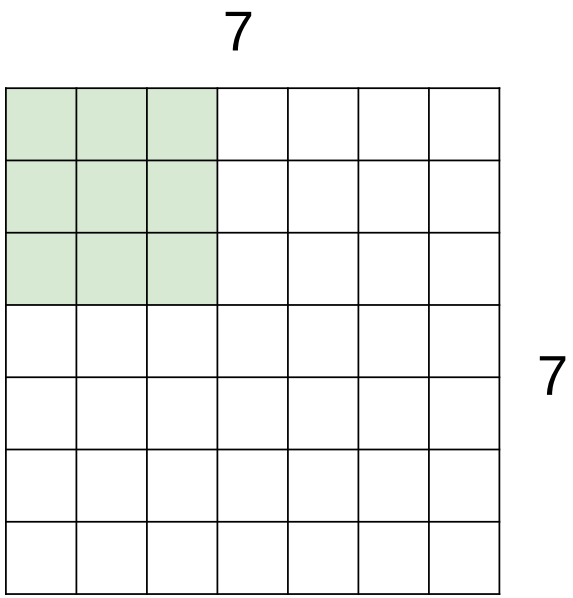


- 7x7 input (spatially) assume 3x3 filter

=> 5x5 output

Convolutions: More detail

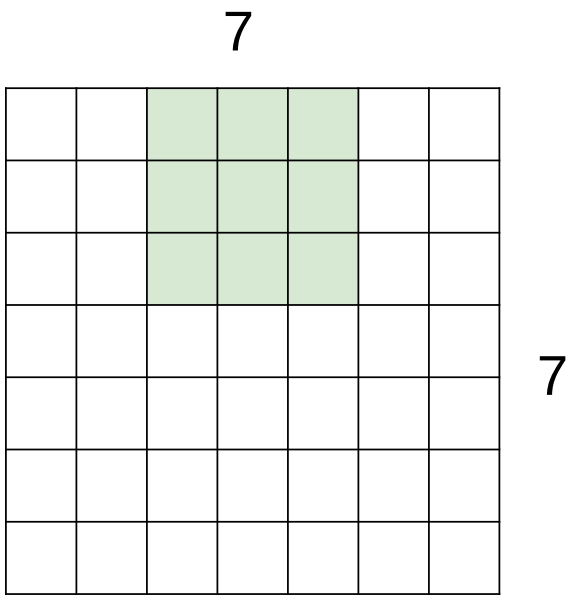
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Convolutions: More detail

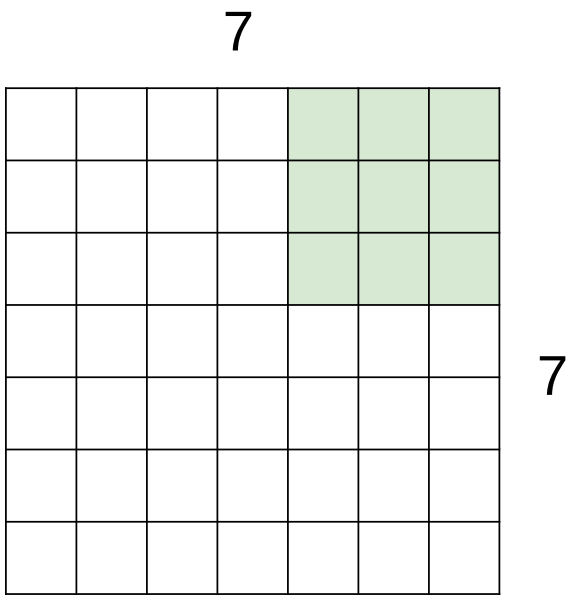
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Convolutions: More detail

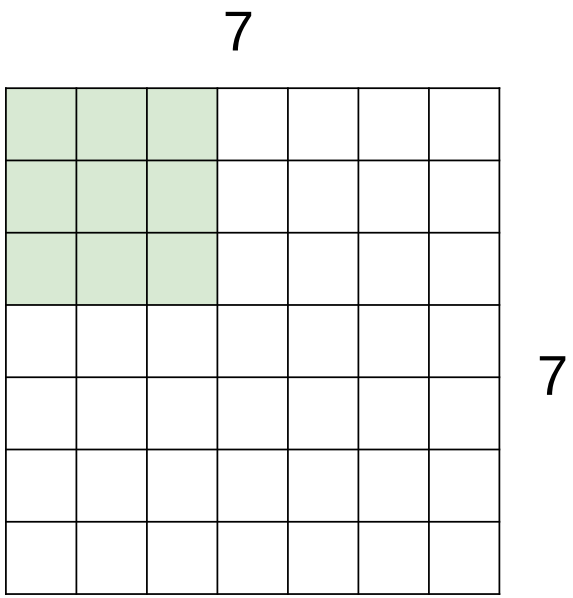
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Convolutions: More detail

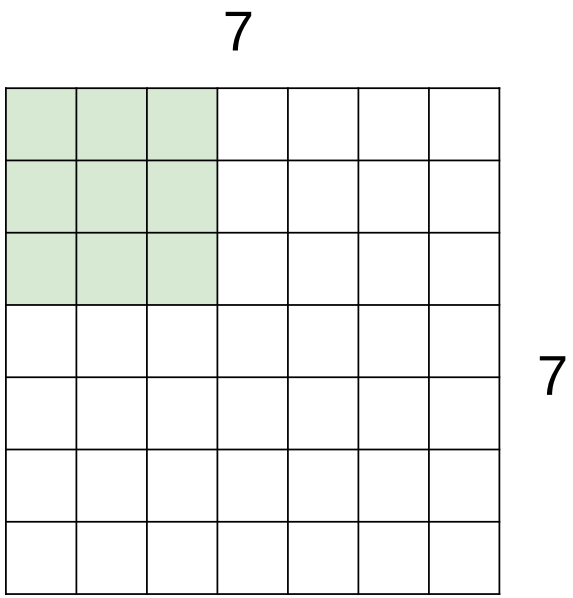
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

Convolutions: More detail

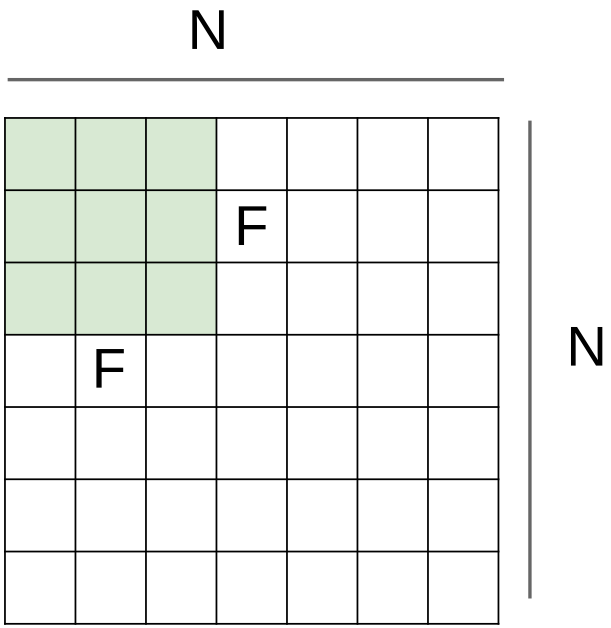
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

Convolutions: More detail

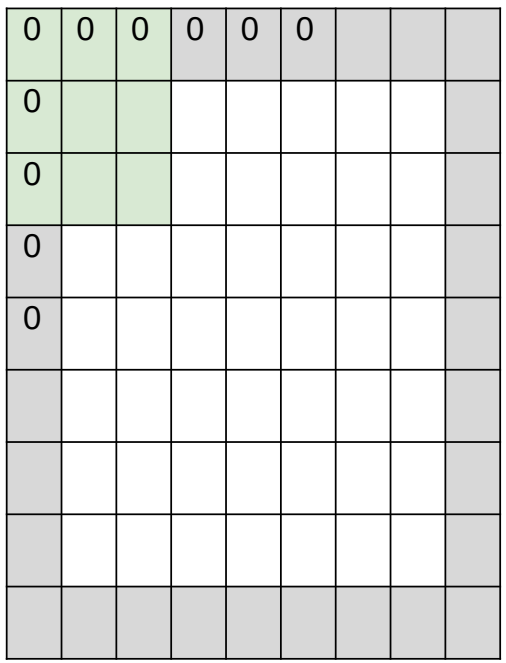


Output size:
(N - F) / stride + 1

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 :\

Convolutions: More detail

In practice: Common to zero pad the border

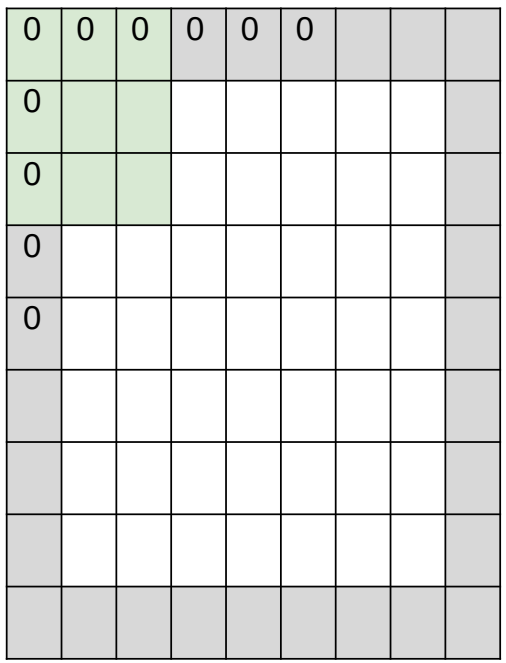


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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

Convolutions: More detail

In practice: Common to zero pad the border

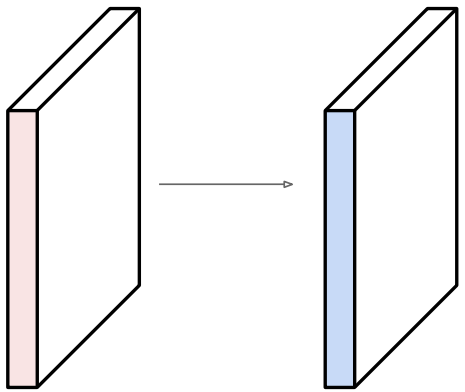


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

Convolutions: More detail

Examples time:

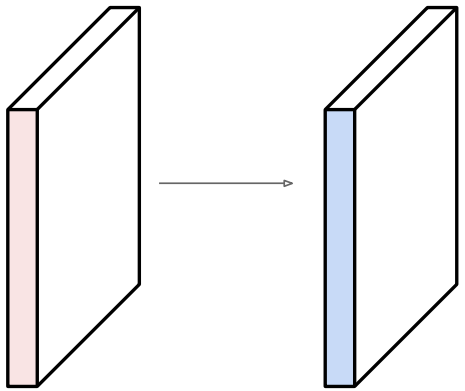


Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?

Convolutions: More detail

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**

Image Source: deeplearning.ai

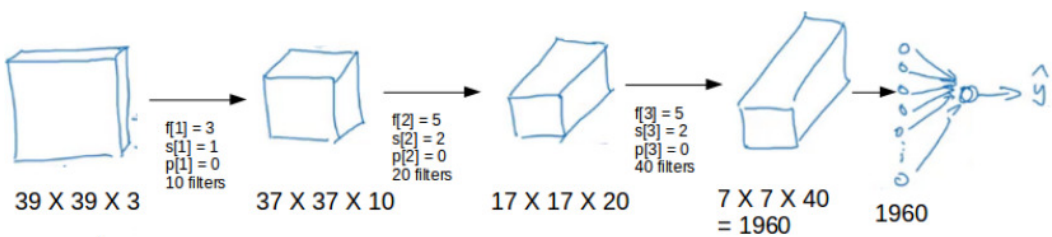
Computer Vision Problems

Suppose we have 10 filters applying on input (6 X 6 X 3), each of shape 3 X 3 X 3. What will be the number of parameters in that layer?

- Number of parameters for each filter = 3*3*3 = 27
- There will be a bias term for each filter, so total parameters per filter = 28
- As there are 10 filters, the total parameters for that layer = 28*10 = 280

Image Source: internet

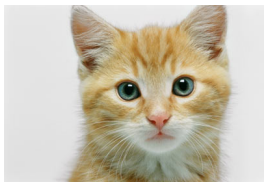
Simple Convolutional Neural Network



- **Size of feature vector : (n+2p-f)/s +1**
- **n : dimension of matrix**
- **p : size of padding**
- **f : size of filter**
- **s : size of stride**

Image Source: deeplearning.ai

The whole CNN



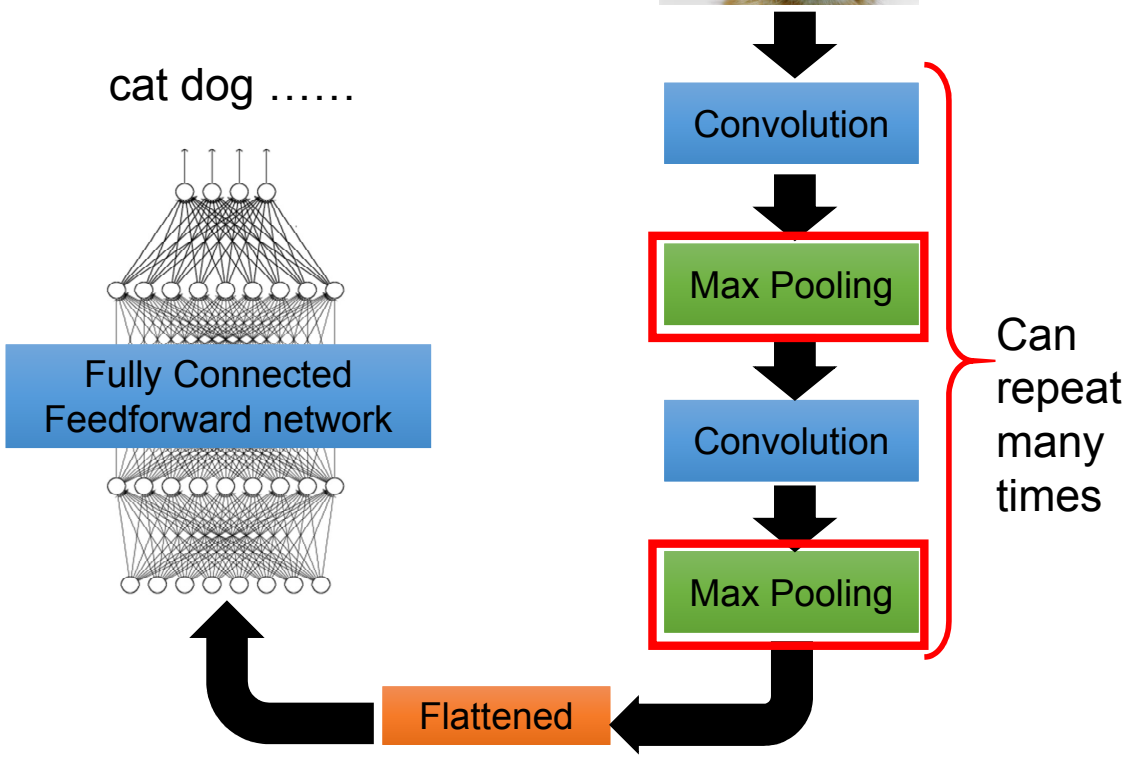


Image Source: internet

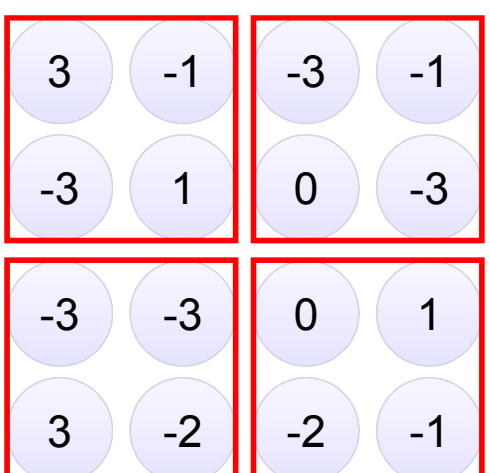
Max Pooling

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

Filter 1

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

Filter 2



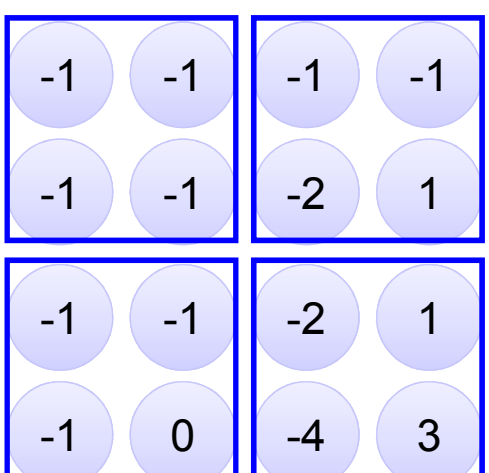


Image Source: internet

Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

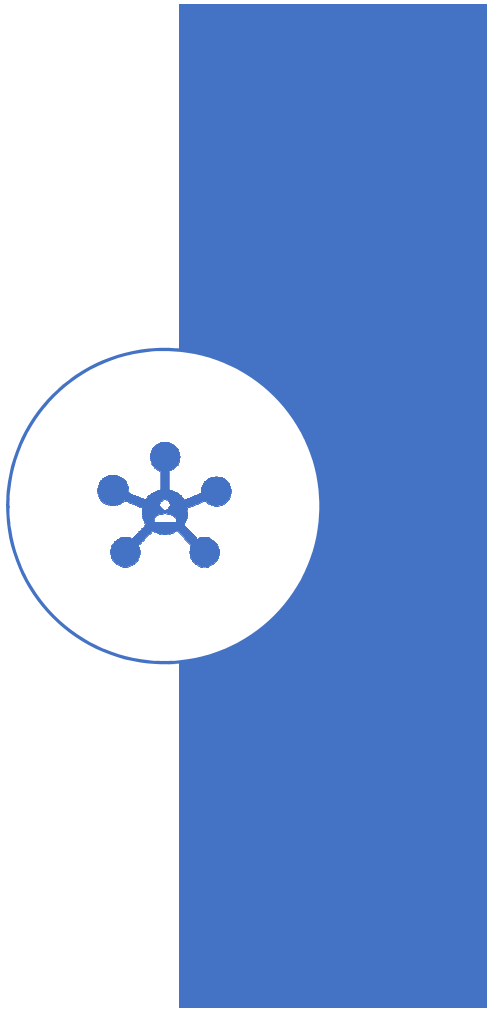
We can subsample the pixels to make image smaller

fewer parameters to characterize the image

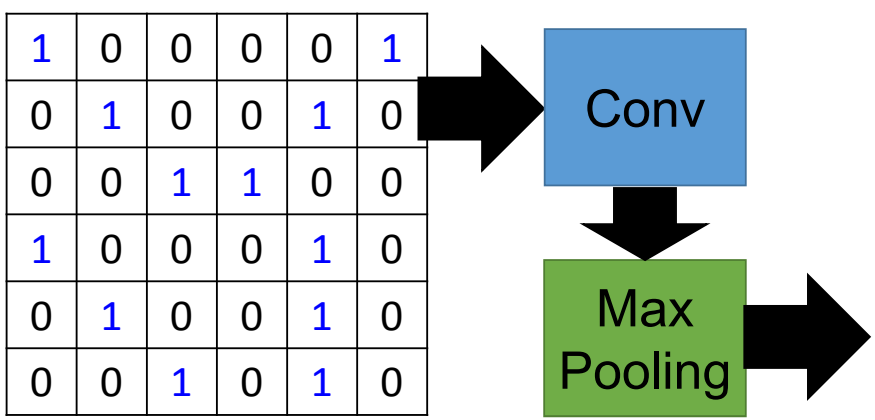
Image Source: internet

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

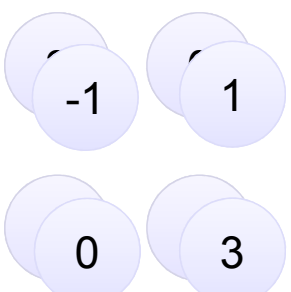


Max Pooling



6 x 6 image

New image but smaller

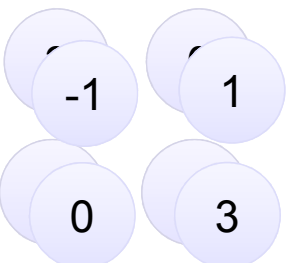


2 x 2 image

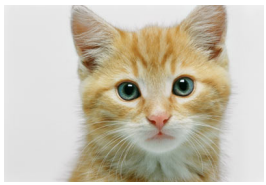
Each filter is a channel

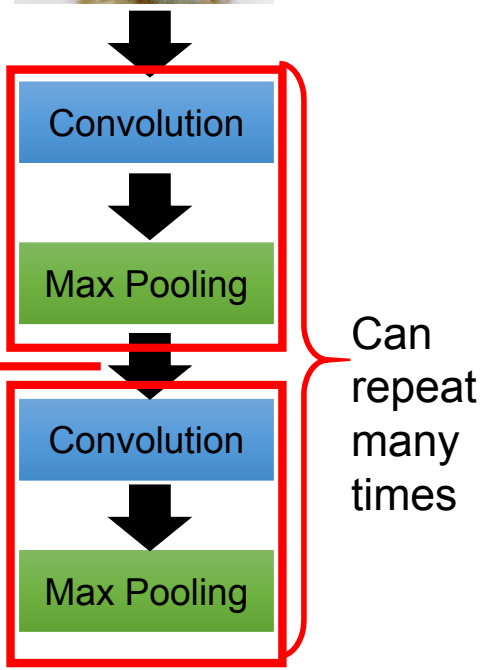
Image Source: internet

The whole CNN



A new image



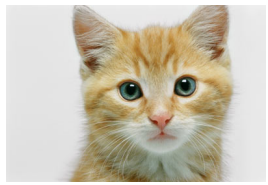


Smaller than the original image

The number of channels is the number of filters

Image Source: internet

The whole CNN



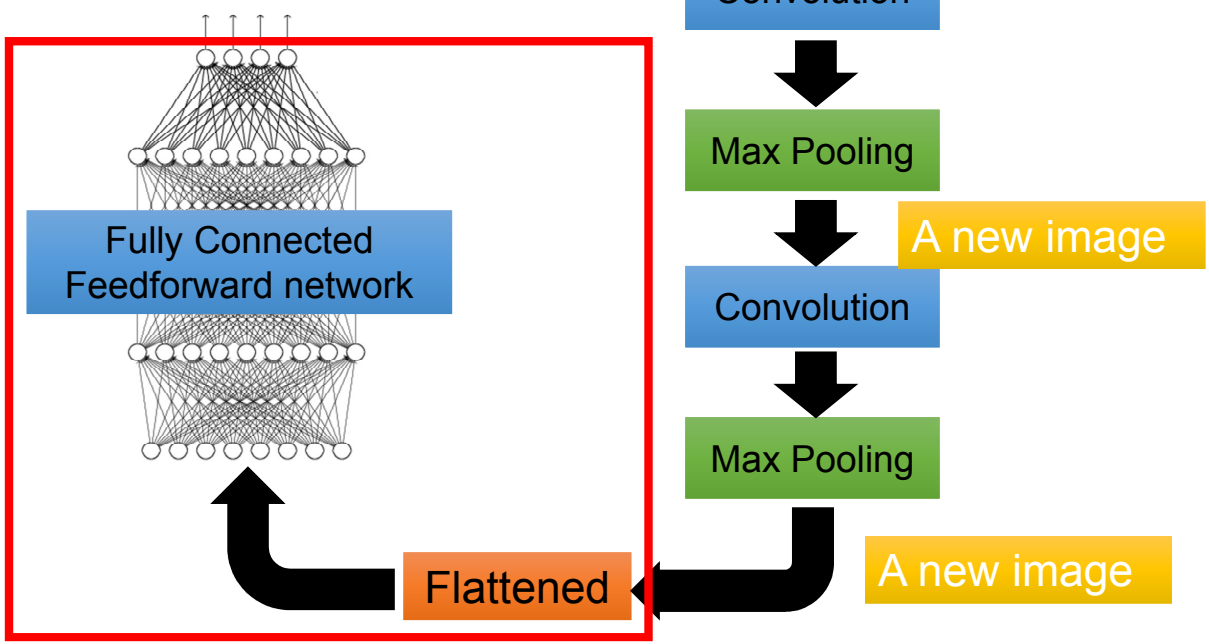


Image Source: internet

Flattening

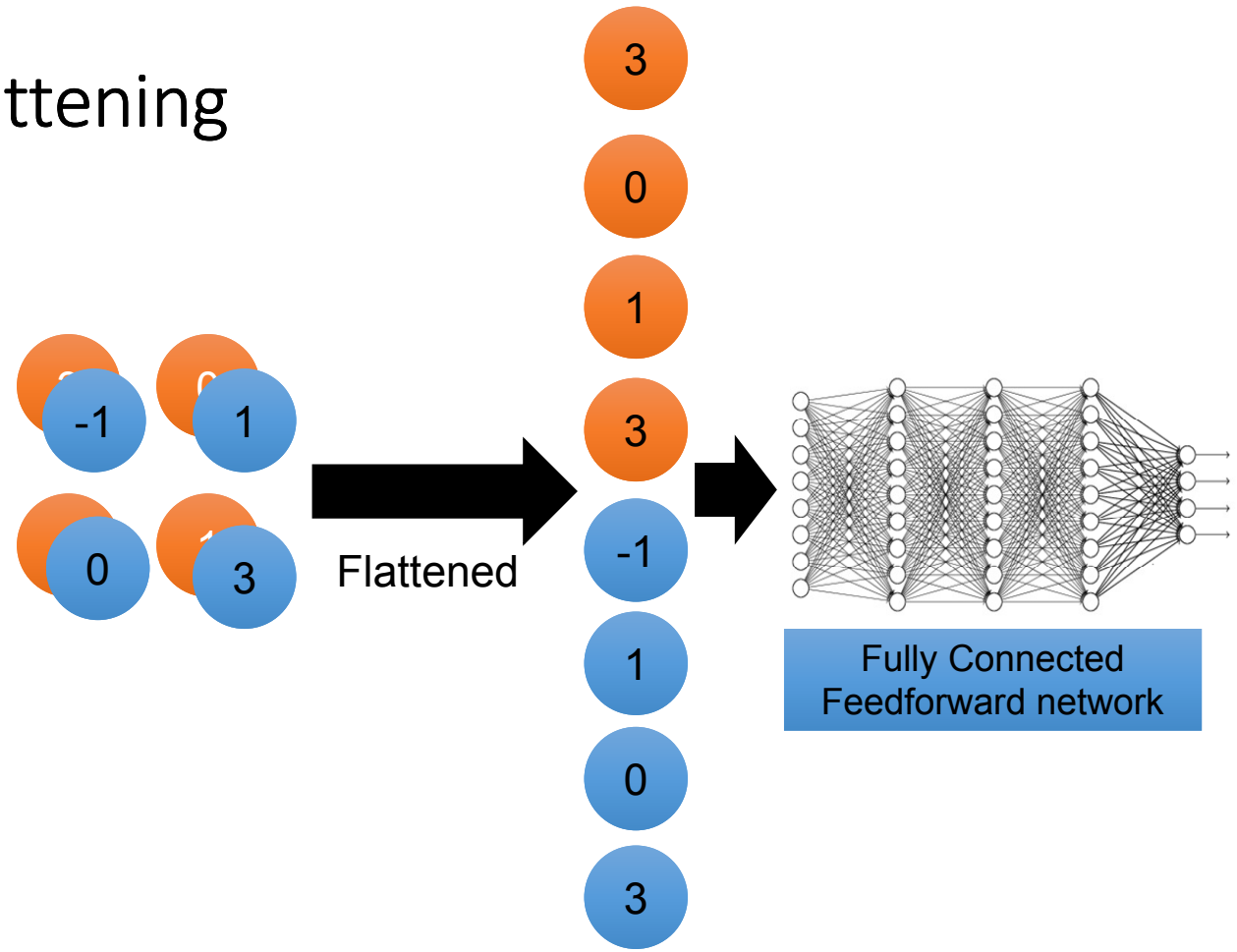


Image Source: internet

Classic Networks

1. LeNet-5
2. AlexNet
3. VGG

Object Detection using CNN

Classification

Classification + Localization = Detection

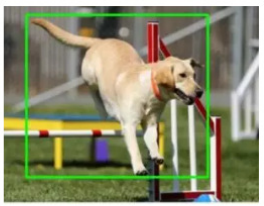
Object Detection is modeled as a classification problem

- We take windows of fixed sizes
- Run over input image at all the possible locations
- Feed these patches to an image classifier.
- It predicts the class of the object in the window(or background if none is present)

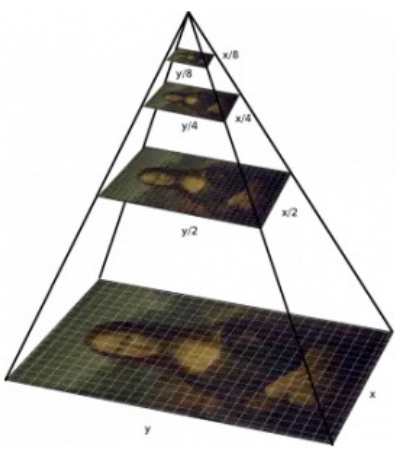
Problem ➡ Solution

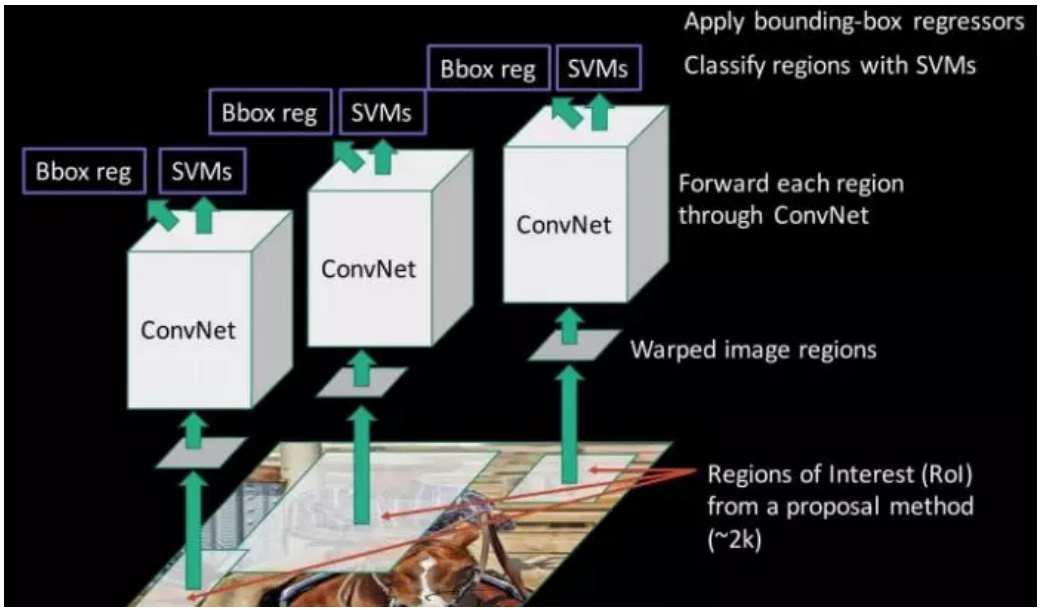
- **Resize the image at multiple scales**
- **Most commonly, the image is downsampled(size is reduced)**
- **On each of these images, a fixed size window detector is run.**
- **Now, all these windows are fed to a classifier to detect the object of interest**

Small sized object



Big sized object. What size do you choose for your sliding window detector?

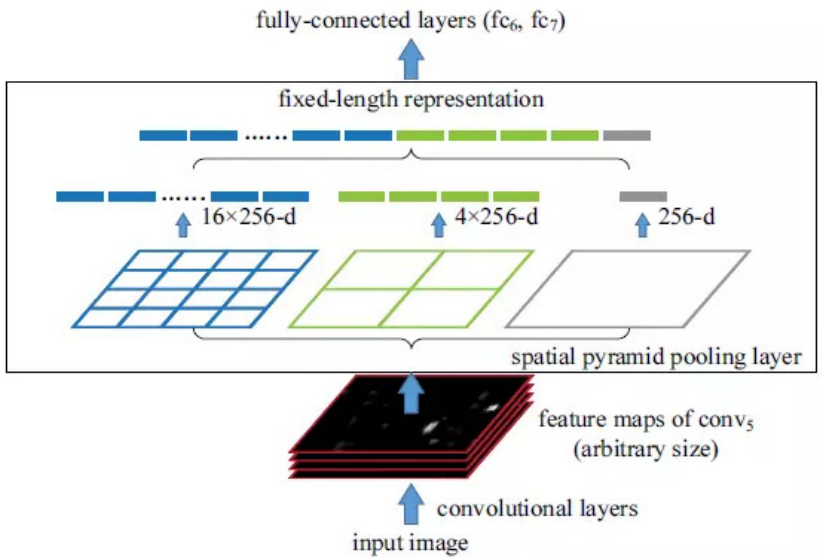




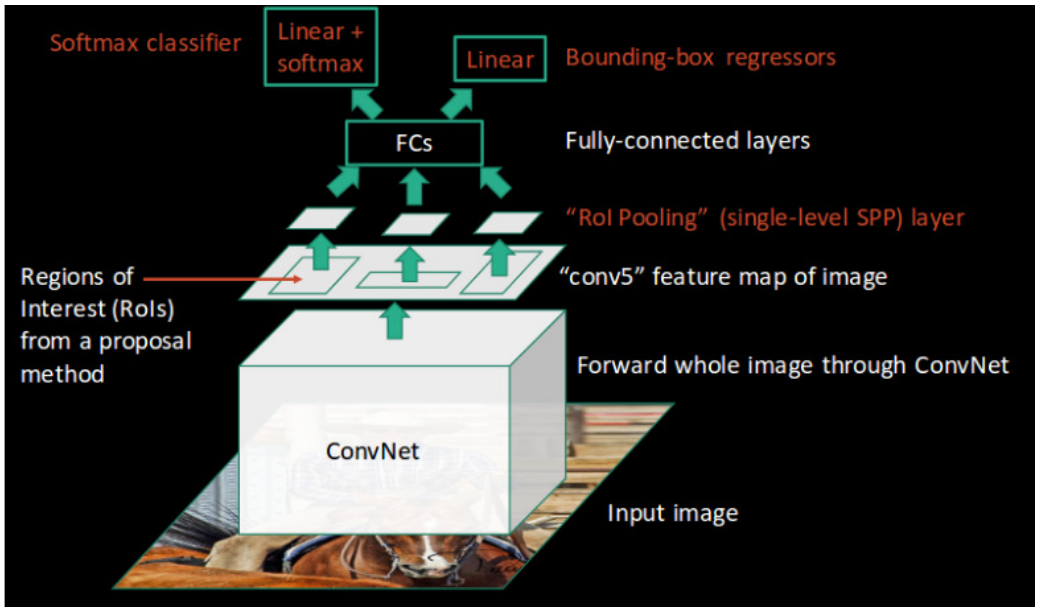
Region-based Convolutional Neural Networks(R-CNN)

- Run **Selective Search** to generate probable objects (~2k regions)
- Feed these patches to CNN, followed by SVM to predict the class of each patch.
- Optimize patches by training bounding box regression separately.

Spatial Pyramid Pooling(SPP-net)



- Calculate the CNN representation for entire image only once
- **It uses spatial pooling after the last convolutional layer**
- SPP layer divides a region of any arbitrary size into a constant number of bins and max pool is performed on each of the bins
- Since the number of bins remains the same, a constant size vector is produced

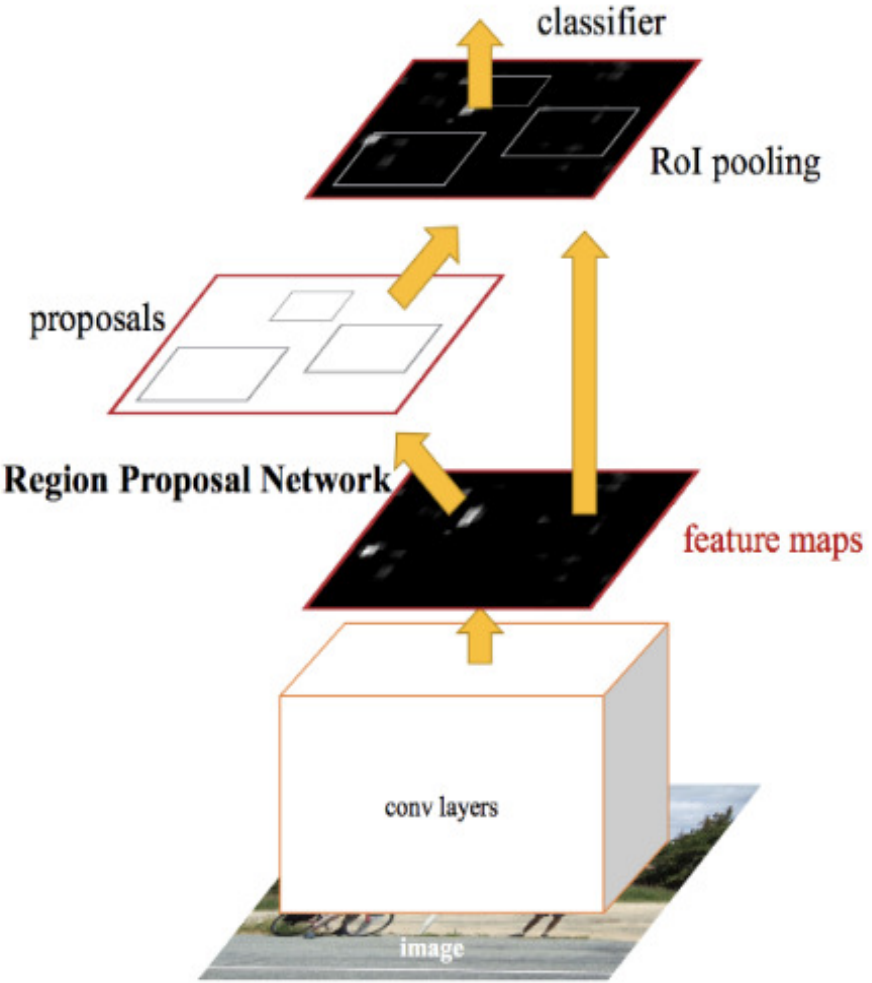


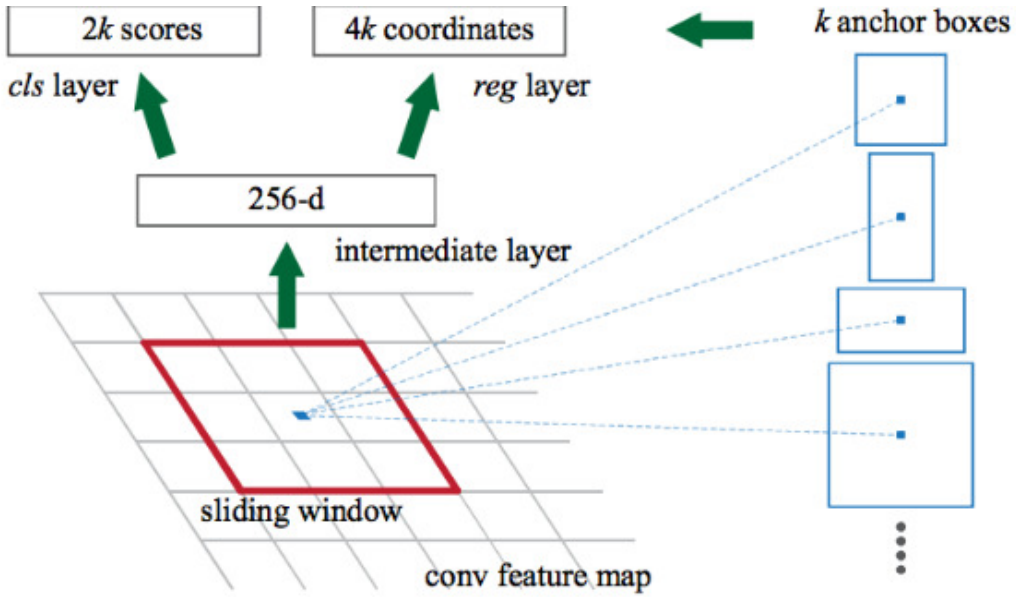
Fast R-CNN

- Fast RCNN uses the ideas from SPP-net and RCNN
- Apply the RoI pooling layer on the extracted regions of interest to make sure all the regions are of the same size
- These regions are passed on to a fully connected network which classifies them, as well as returns the bounding boxes using softmax and linear regression layers simultaneously

Faster R-CNN

- We take an image as input and pass it to the ConvNet which returns the feature map for that image
- **Region Proposal Network** (lightweight CNN) is applied on these feature maps. This returns the object proposals along with their objectness score
- A RoI pooling layer is applied on these proposals to bring down all the proposals to the same size
- Finally, the proposals are passed to a fully connected layer which has a softmax layer and a linear regression layer at its top, to classify and output the bounding boxes for objects.





Region Proposal Network (RPN)

- RPN uses a sliding window over the feature maps
- At each window, it generates *k* Anchor boxes of different shapes and sizes
- For each anchor, RPN predicts two things:
  - first is the probability that an anchor is an object
  - Second is the bounding box regressor for adjusting the anchors to better fit the object

Summary of the object detection models

Algorithm	Features	Prediction time / image	Limitations
CNN	Divides the image into multiple regions and then classify each region into various classes.	–	Needs a lot of regions to predict accurately and hence high computation time.
RCNN	Uses selective search to generate regions. Extracts around 2000 regions from each image.	40-50 seconds	High computation time as each region is passed to the CNN separately also it uses three different model for making predictions.
Fast RCNN	Each image is passed only once to the CNN and feature maps are extracted. Selective search is used on these maps to generate predictions. Combines all the three models used in RCNN together.	2 seconds	Selective search is slow and hence computation time is still high.
Faster RCNN	Replaces the selective search method with region proposal network which made the algorithm much faster.	0.2 seconds	Object proposal takes time and as there are different systems working one after the other, the performance of systems depends on how the previous system has performed.

Thank You