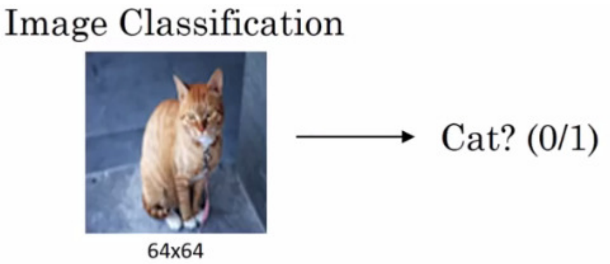
Convolutional Neural Network

(A Deep Neural Network)

ABHISHEK MUKHOPADHYAY

INSTRUCTOR: DR. PRADIPTA BISWAS

I3D LABORATORY, CPDM, IISC



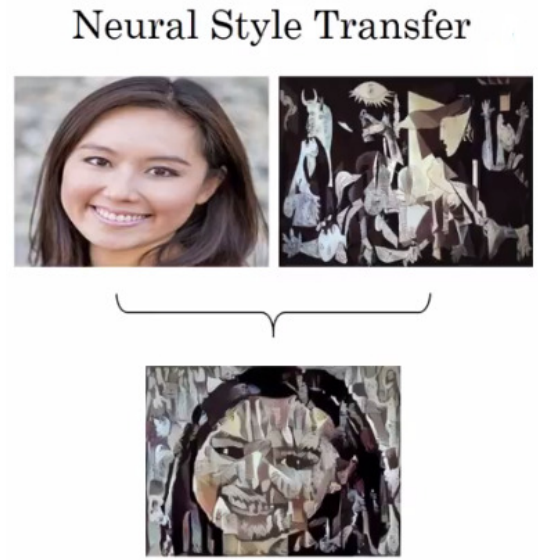


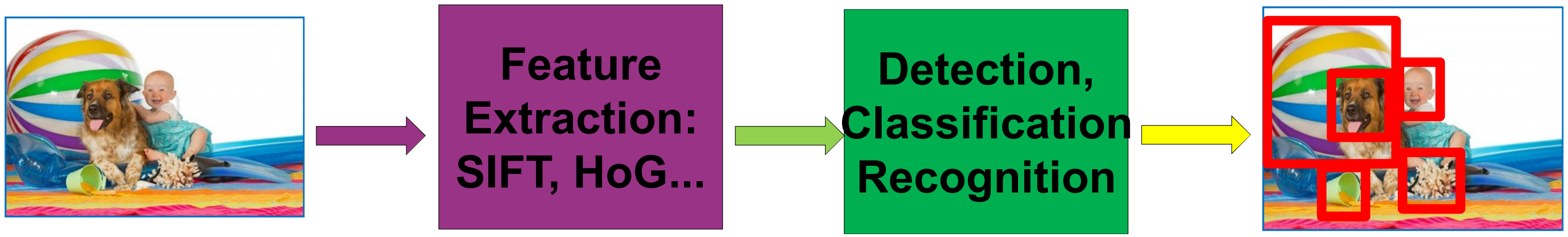
Image Source: deeplearning.ai

Computer Vision Problems

Classical Computer Vision Pipeline

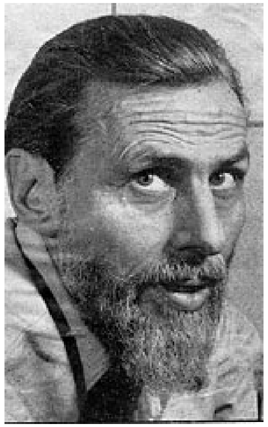
CV experts

1. Select / develop features: SURF, HoG, SIFT, RIFT, …
2. Add on top of this Machine Learning for multi-class recognition and train classifier

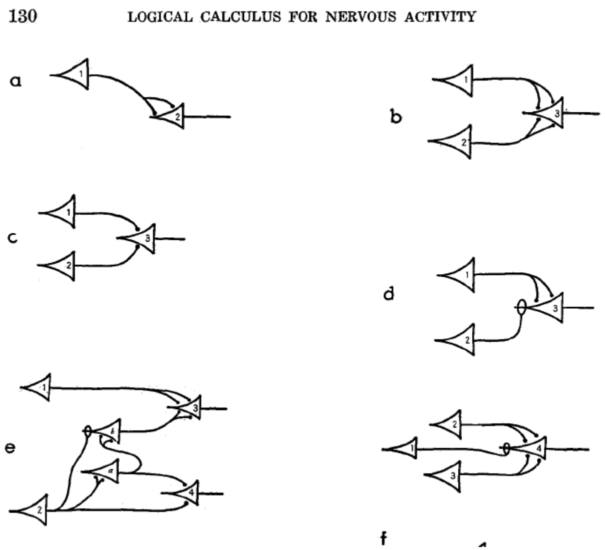


Classical CV feature definition is domain-specific and time-consuming

Neural Network



Warren McCulloch



Walter Pitts

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY, Bulletin of Mathematical Biophysics, Vol. 5, pp. 115-133 (1943).

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

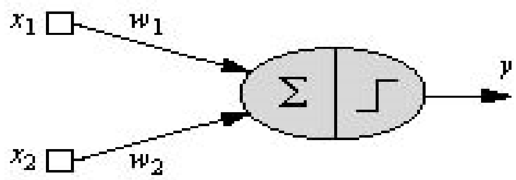
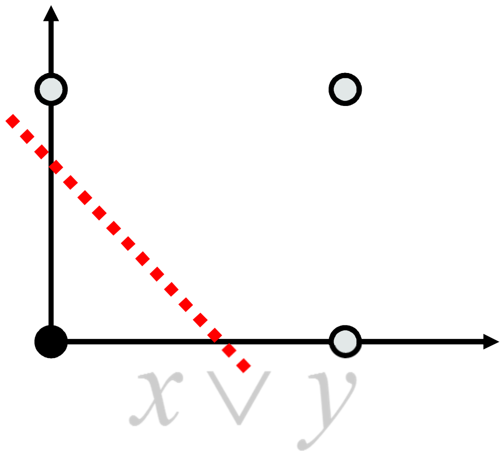
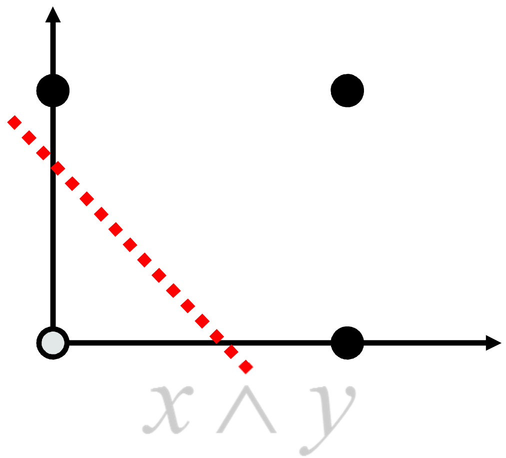


Fig1: an artificial neuron

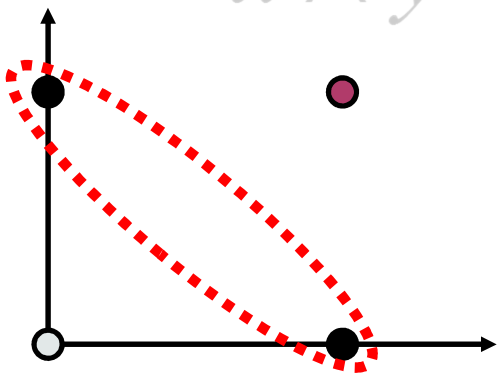
Why We Need Multi Layer ?

Linear Separable:





Linear inseparable: $x \veebar y$



Edge Detection

Vertical edges

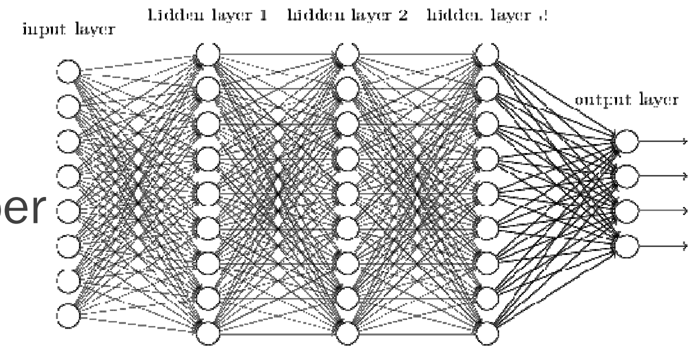
Horizontal edges

How do we detect these edges

Image Source: deeplearning.ai

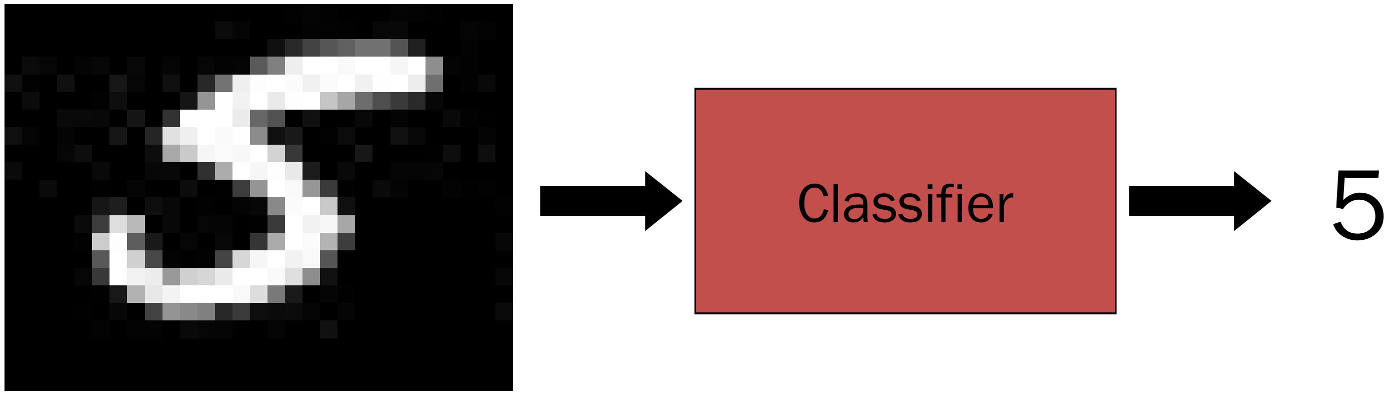
Neural Network?

- 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



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!

$$P(Y|X_1,\ldots,X_n) = \frac{\overbrace{P(X_1,\ldots,X_n|Y)}^{\text{Likelihood}}\overbrace{P(Y)}^{\text{Prior}}}{\underbrace{P(X_1,\ldots,X_n)}_{\text{Normalization Constant}}}$$

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:

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

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

Model Parameters

For the Bayes classifier, we need to "learn" two functions, the likelihood and the prior

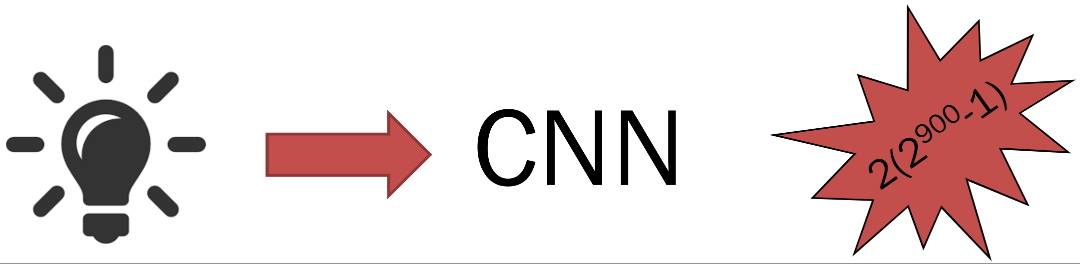
How many parameters are required to specify the prior for our digit recognition example?

Just 1

Model Parameters

How many parameters are required to specify the likelihood?

- (Supposing that each image is 30x30 pixels)



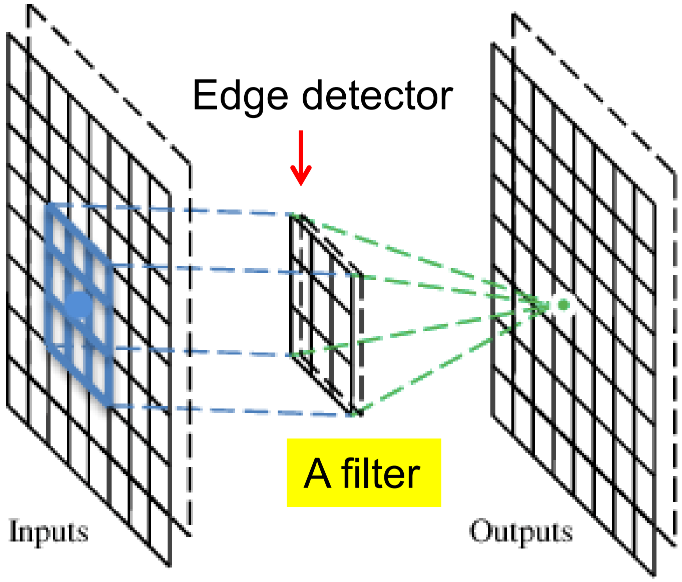
Drive into CNN

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

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

Image Source: internet

Convolution

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

3 -3

6 x 6 image

Image Source: internet

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

6 x 6 image

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

Image Source: internet

Convolution

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

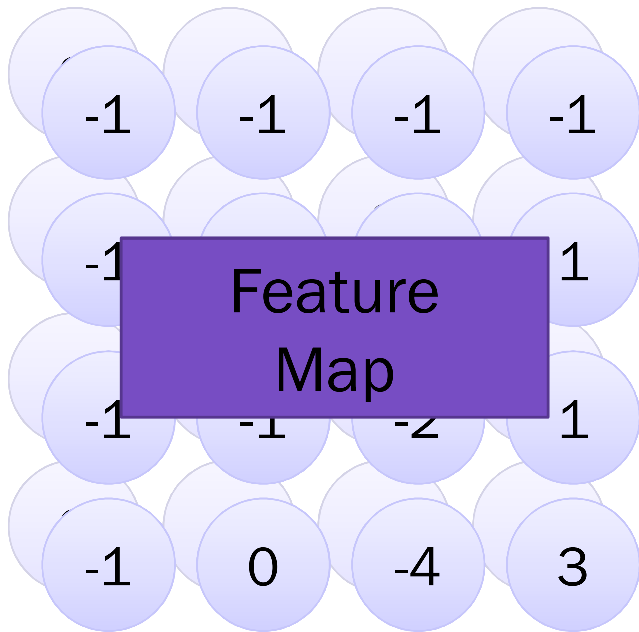
Filter 2

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

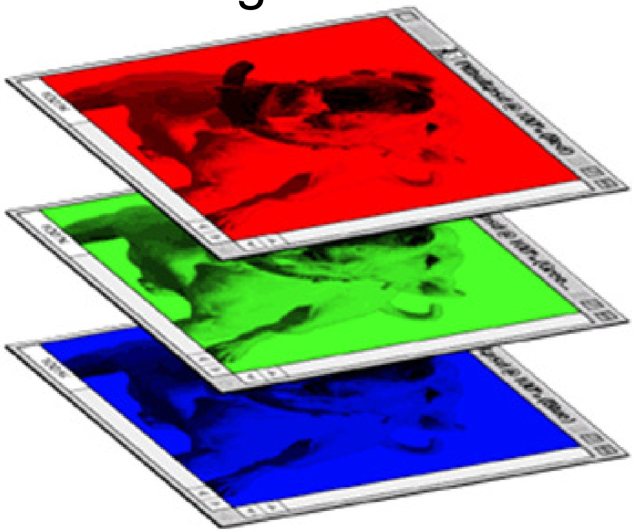


Two 4 x 4 images
Forming 4 x 4 x 2 matrix

Image Source: internet

Convolution over Volume

Color image



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

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

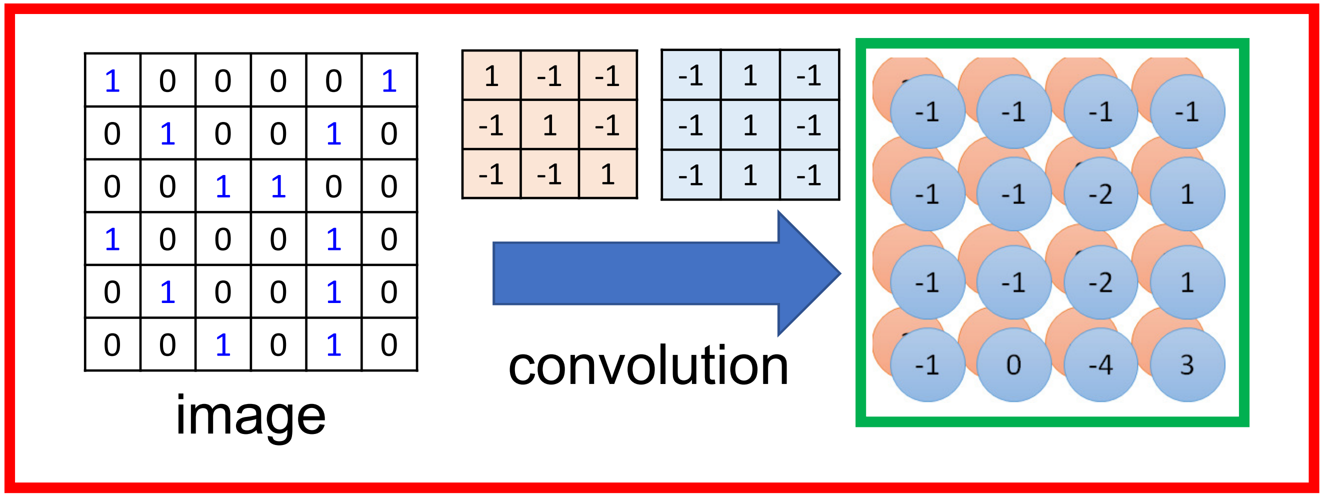
Filter 1

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

Filter 2

Image Source: internet

Convolution v.s. Fully Connected



Fully-connected

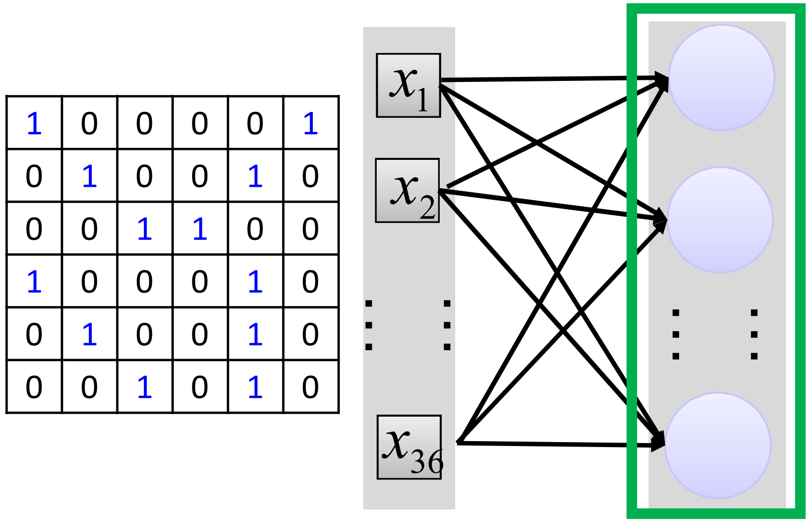


Image Source: internet

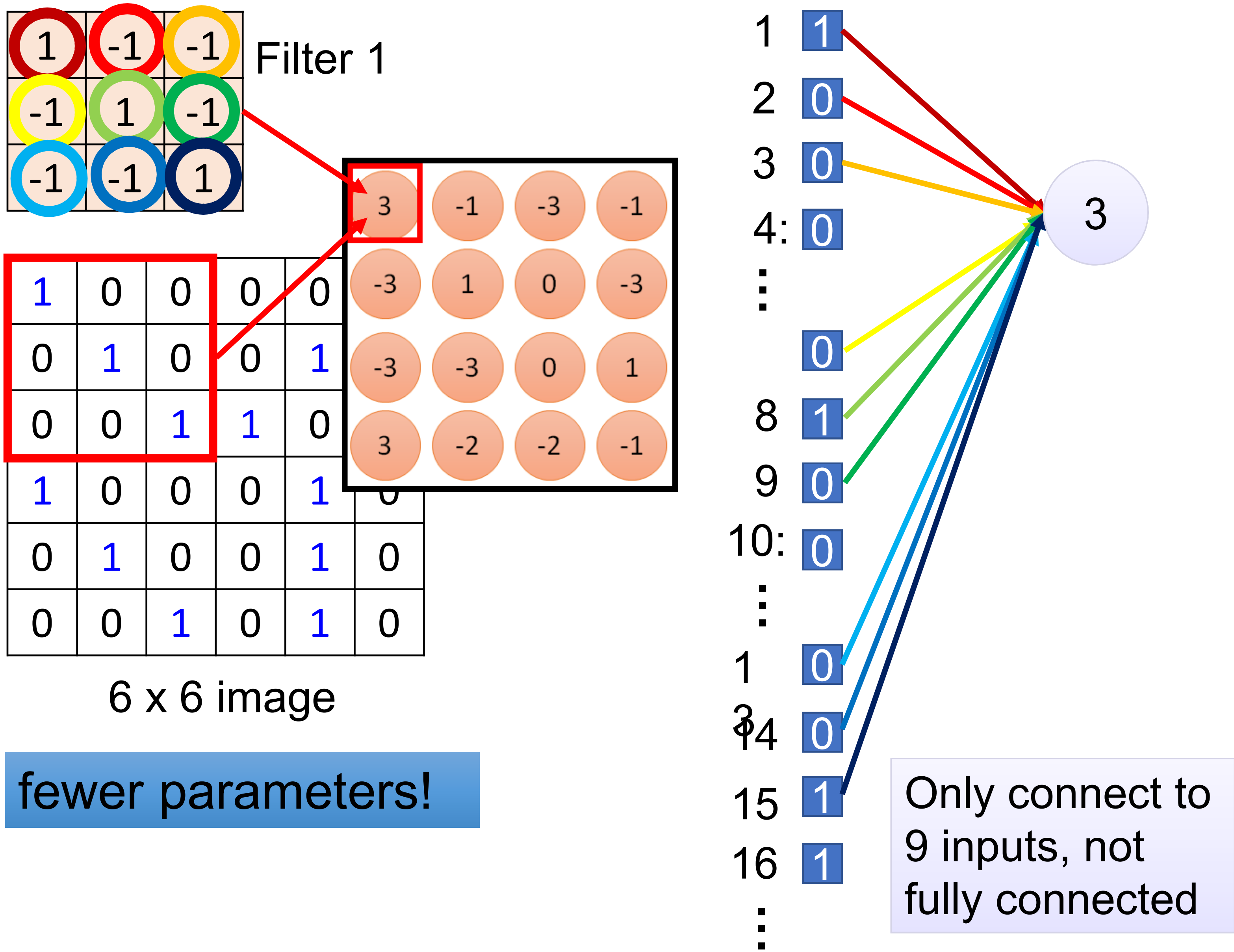


Image Source: internet

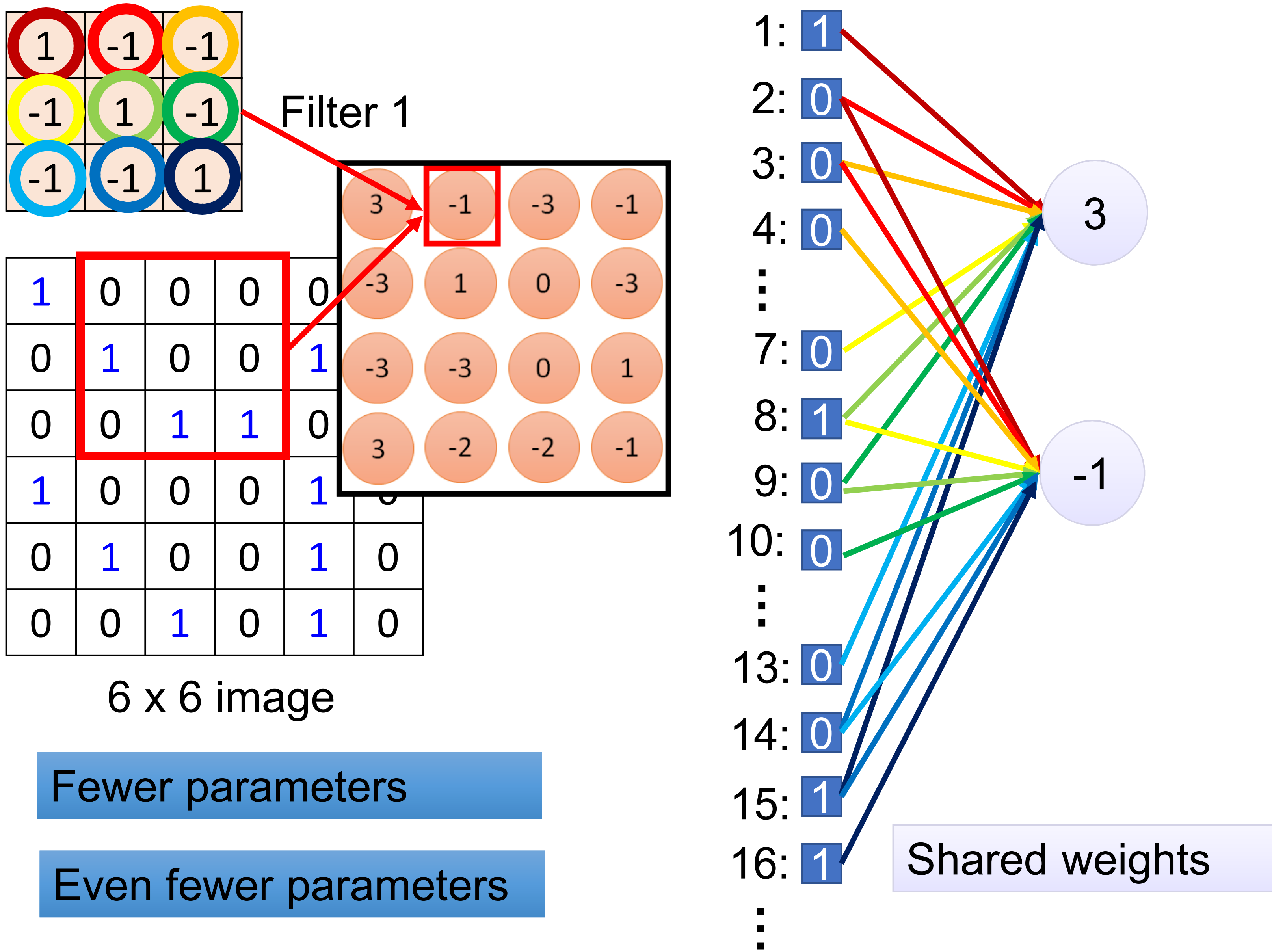


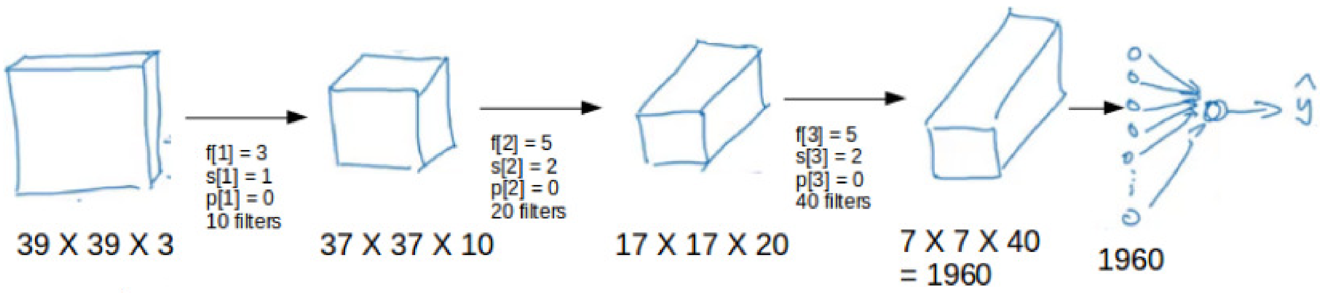
Image Source: internet

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

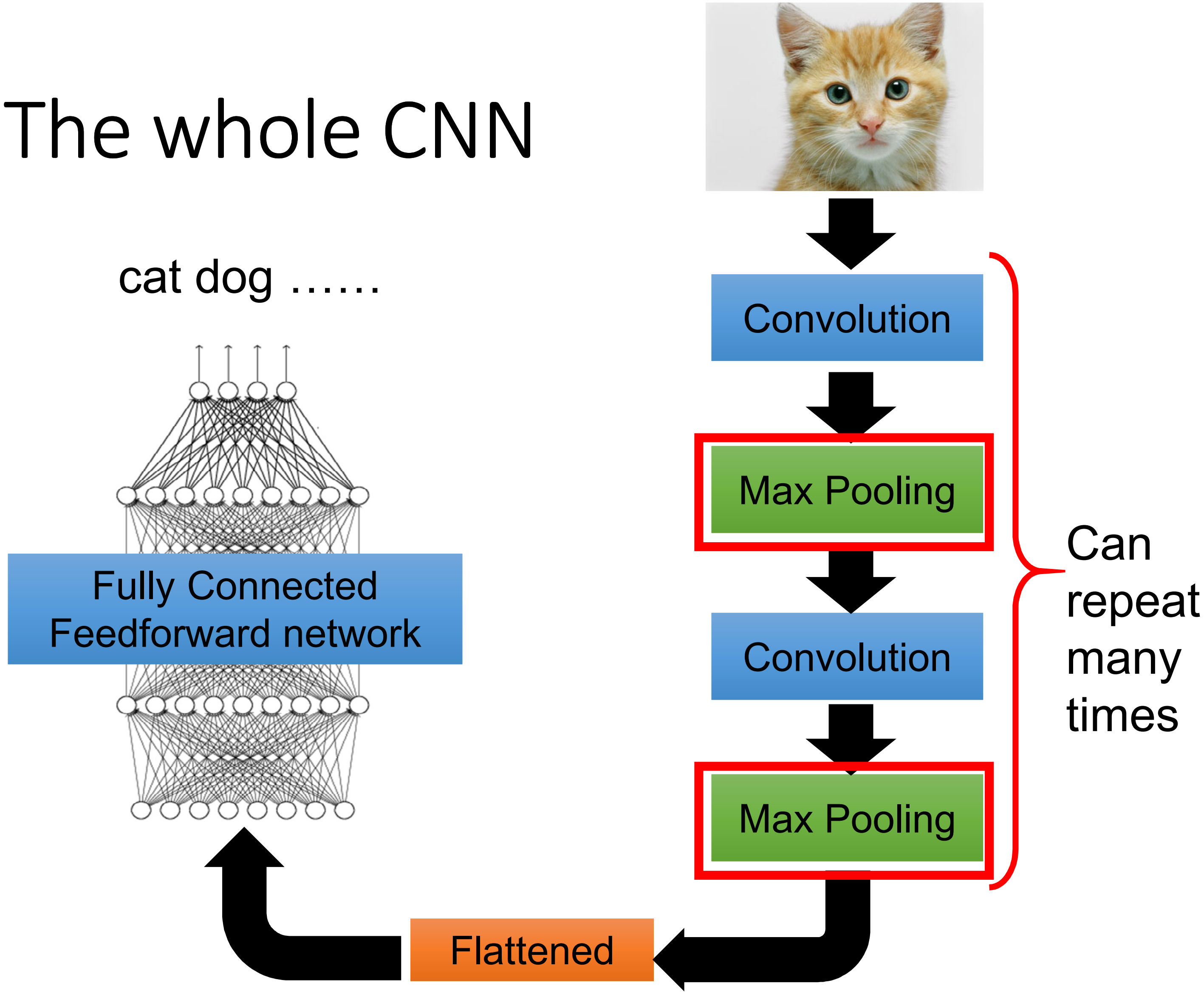


Image Source: internet

Max Pooling

Filter 1

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

Filter 2

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

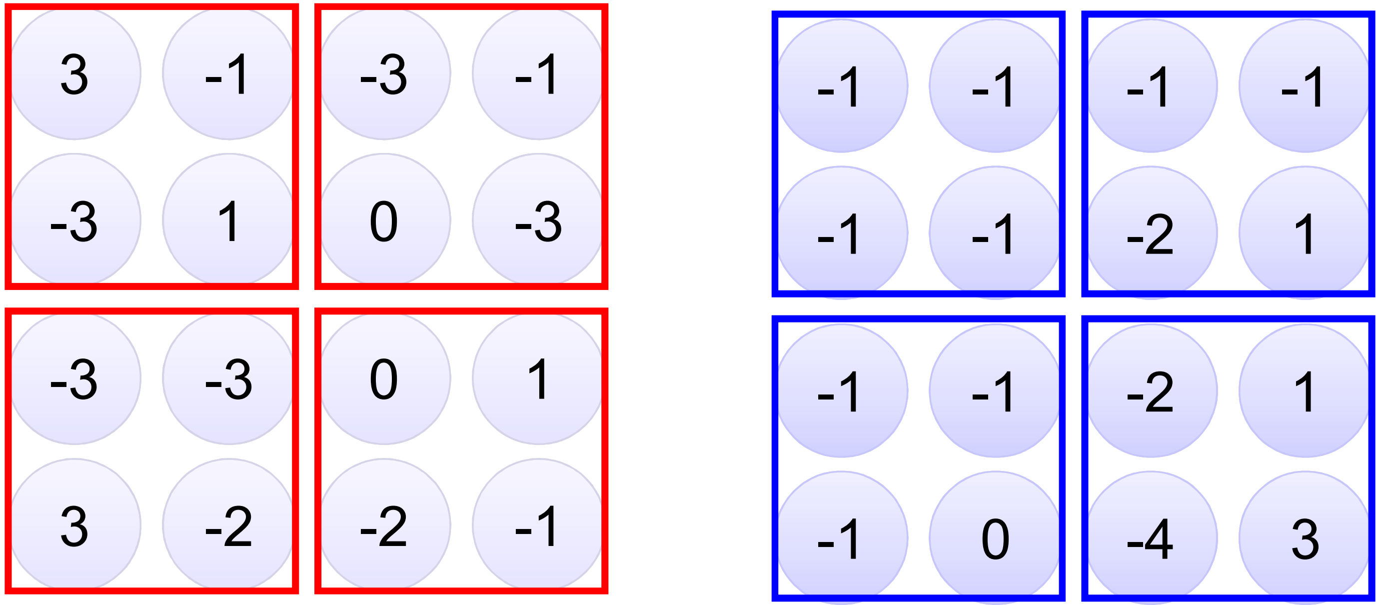


Image Source: internet

Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

We can subsample the pixels to make image smaller

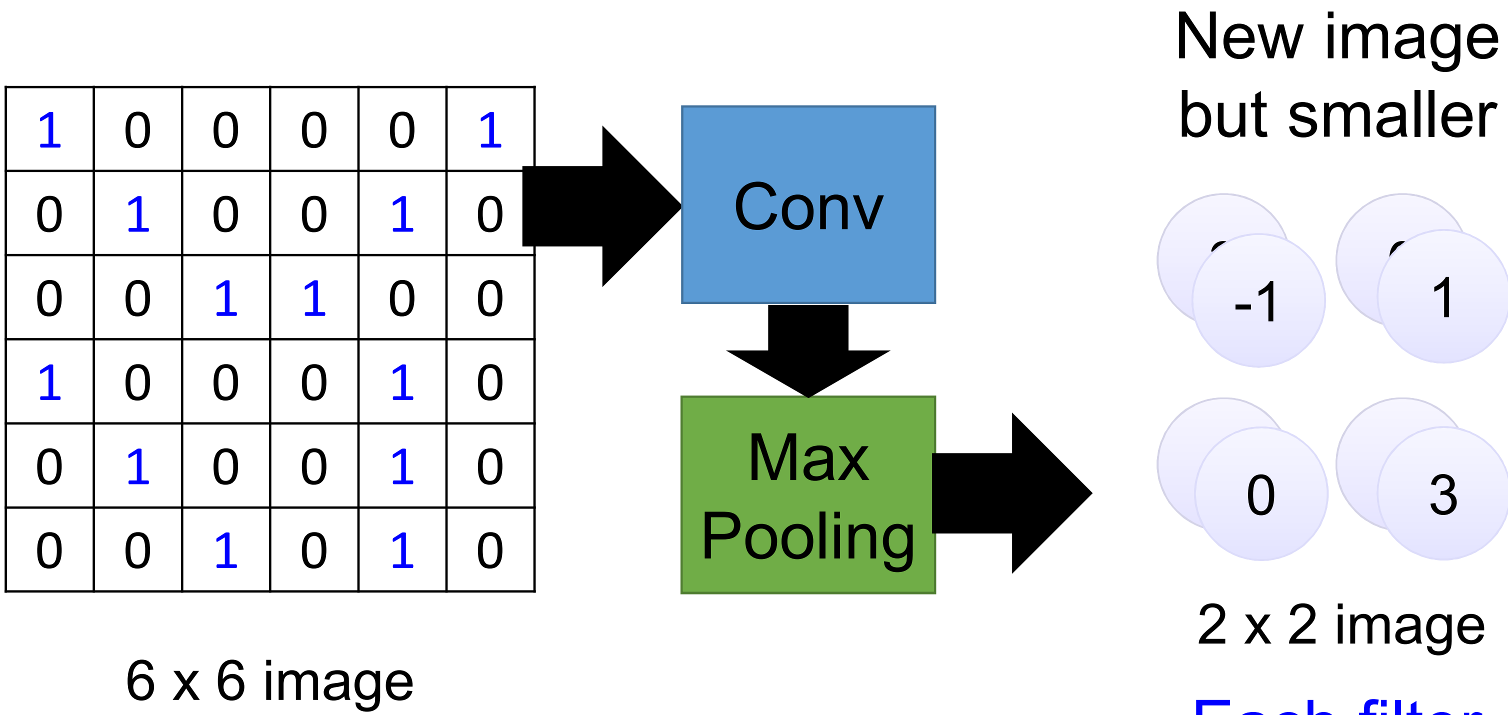
fewer parameters to characterize the image

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

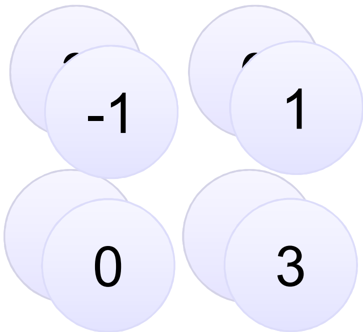
Max Pooling



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

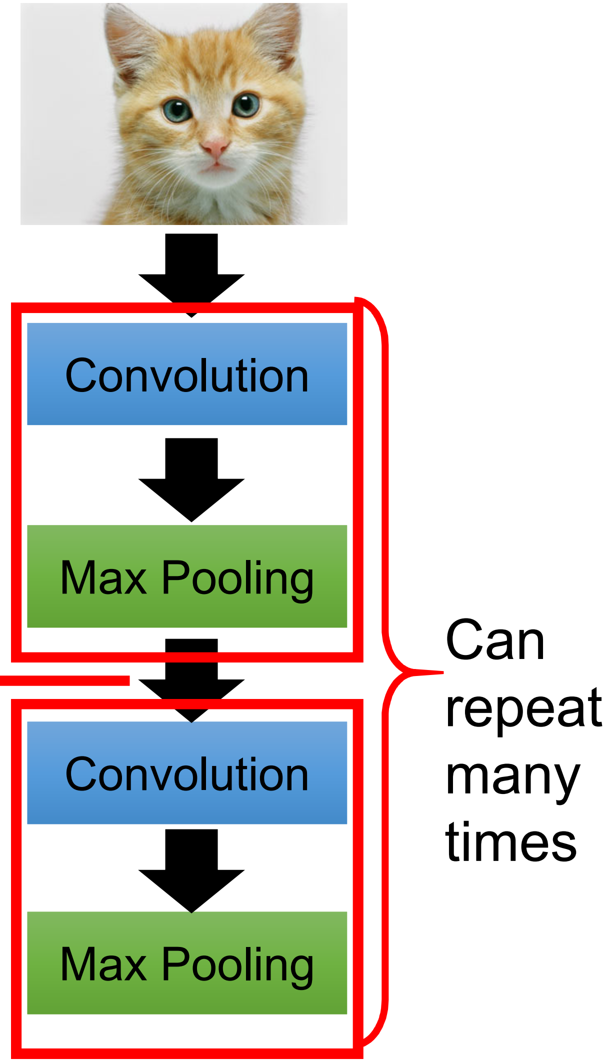


Image Source: internet

The whole CNN

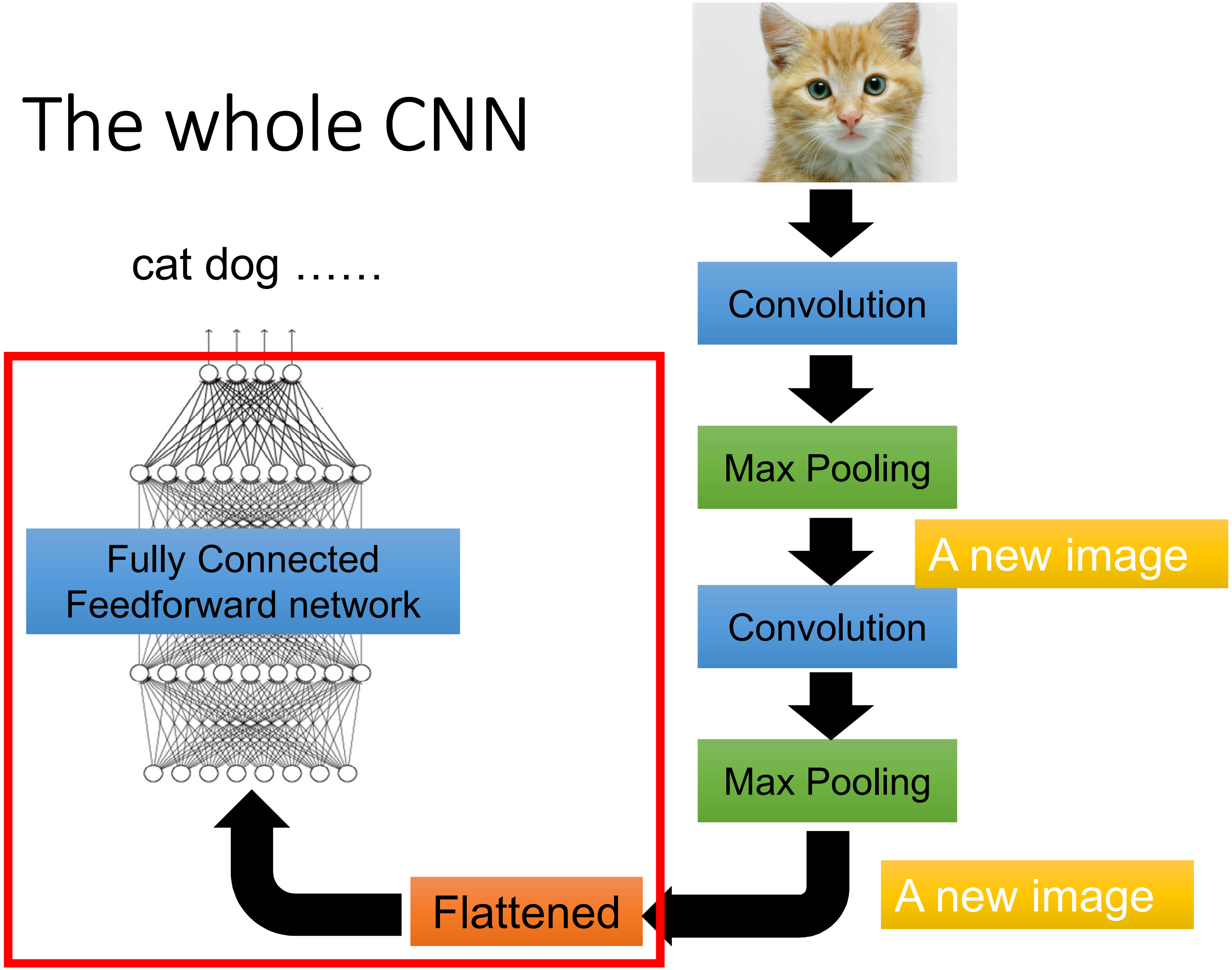


Image Source: internet

Flattening

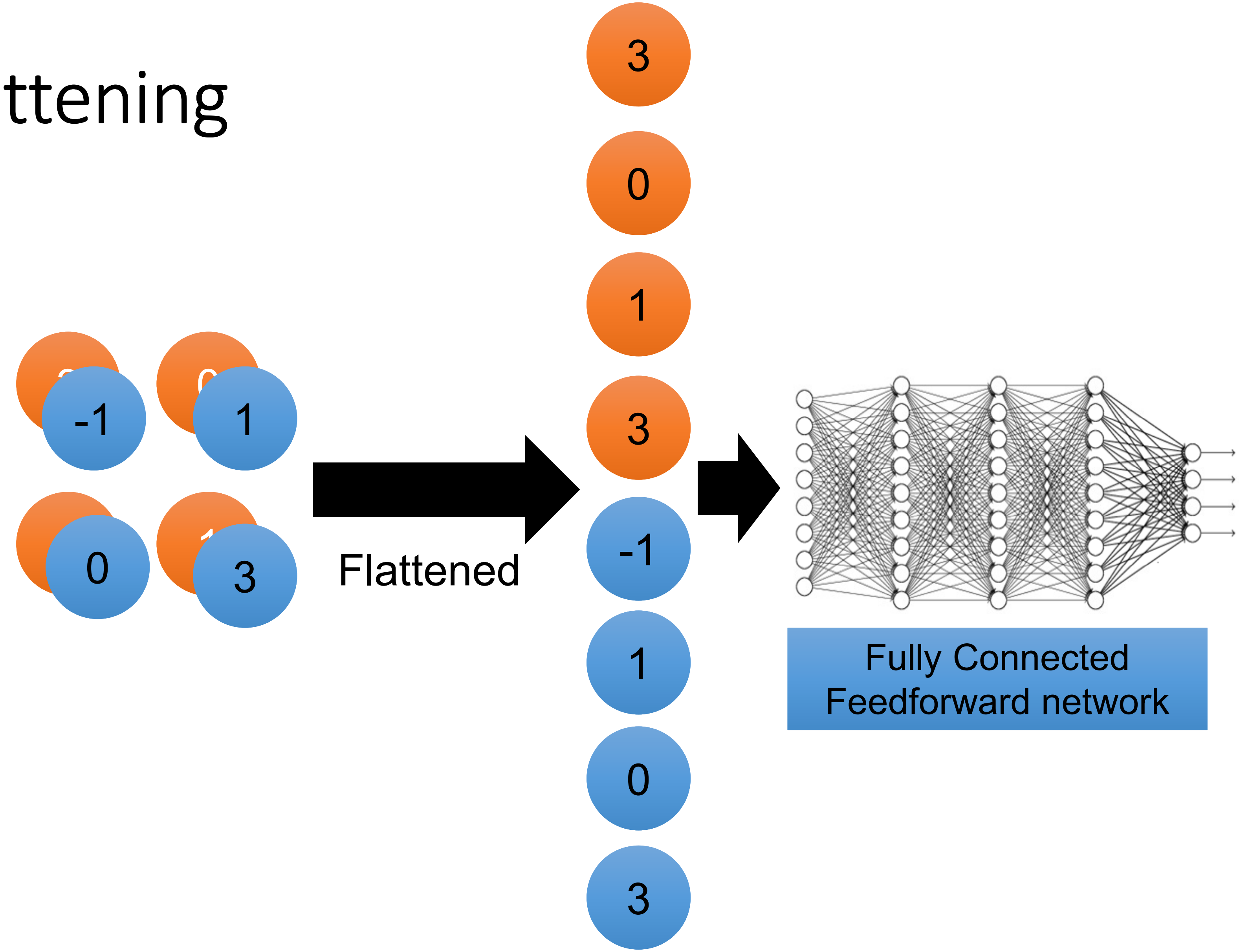
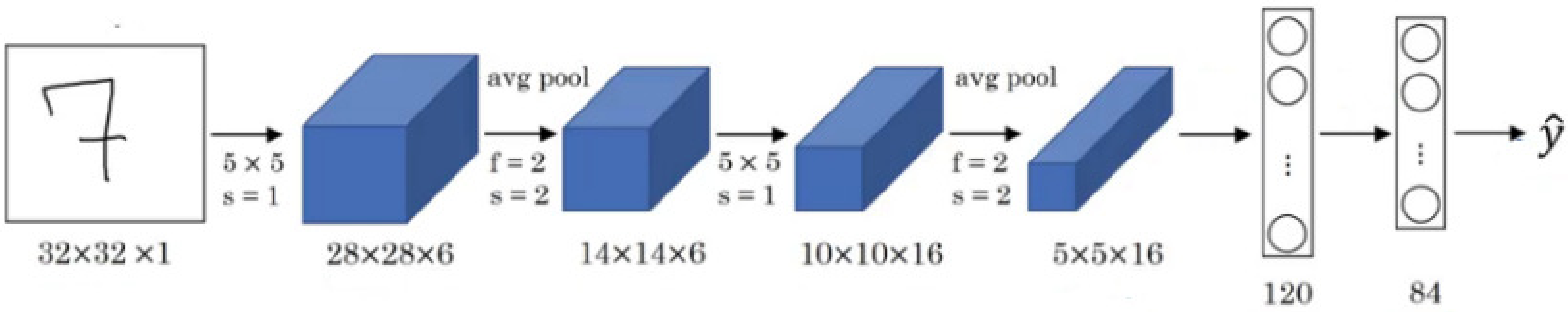


Image Source: internet

Classic Networks

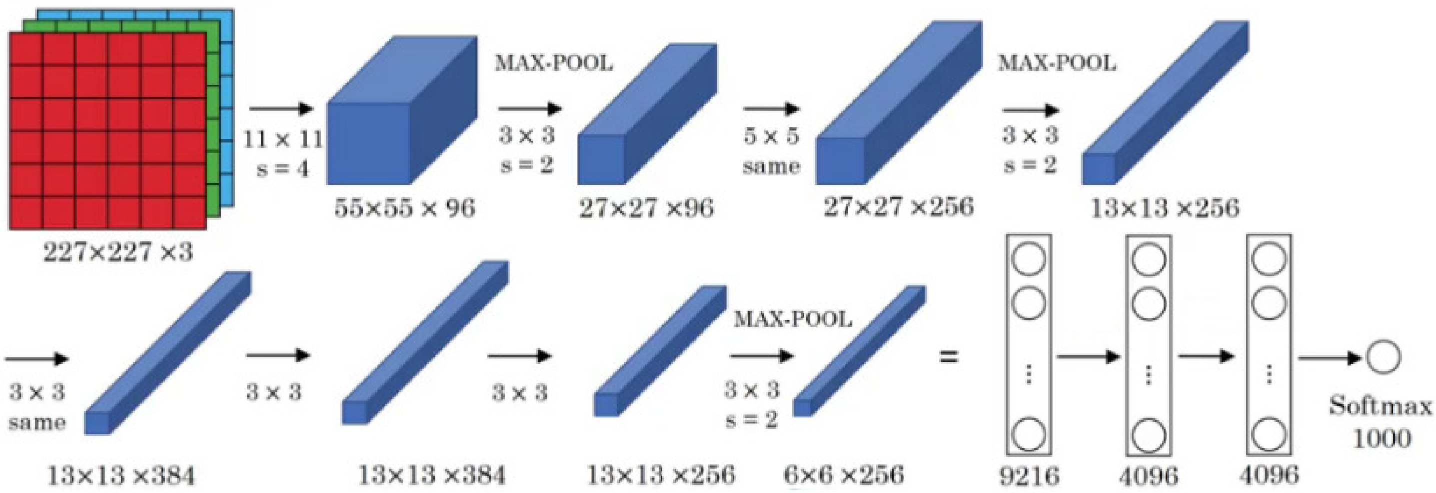
1. LeNet-5
2. AlexNet
3. VGG

LeNet-5



- **Parameters:** 60k
- **Layers flow:** Conv → Pool → Conv → Pool → FC → FC → Output
- **Activation functions:** Sigmoid/tanh and ReLU

AlexNet

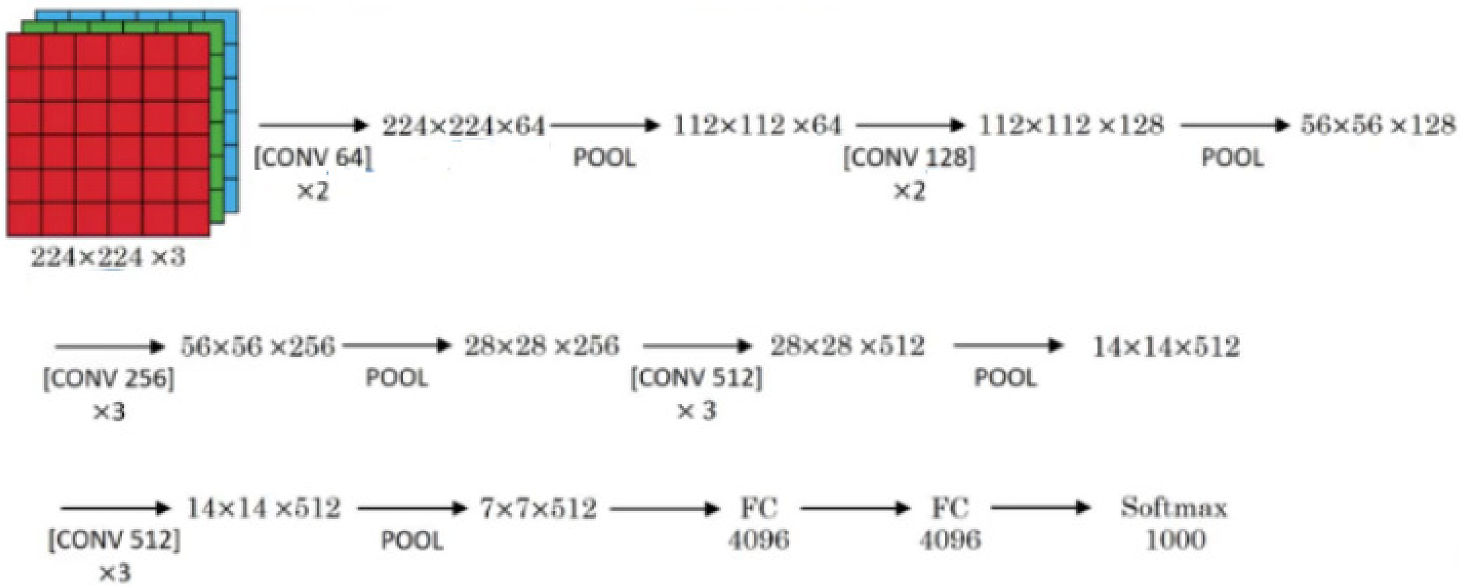


•**Parameters:** 60 million

•**Activation functions:** ReLU

Image Source: deeplearning.ai

VGG-16



- **Parameters:** 138 million
- **Pool:** MAX with stride 2
- **CONV layer**: stride 1

Image Source: deeplearning.ai

CNN in Keras

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

Input

1 x 28 x 28

Convolution

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

How many parameters for each filter? 9

25 x 26 x 26

Max Pooling

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

25 x 13 x 13

Convolution

```
model2.add(Convolution2D(50,3,3))
```

How many parameters for each filter? 225= 25x9

50 x 11 x 11

Max Pooling

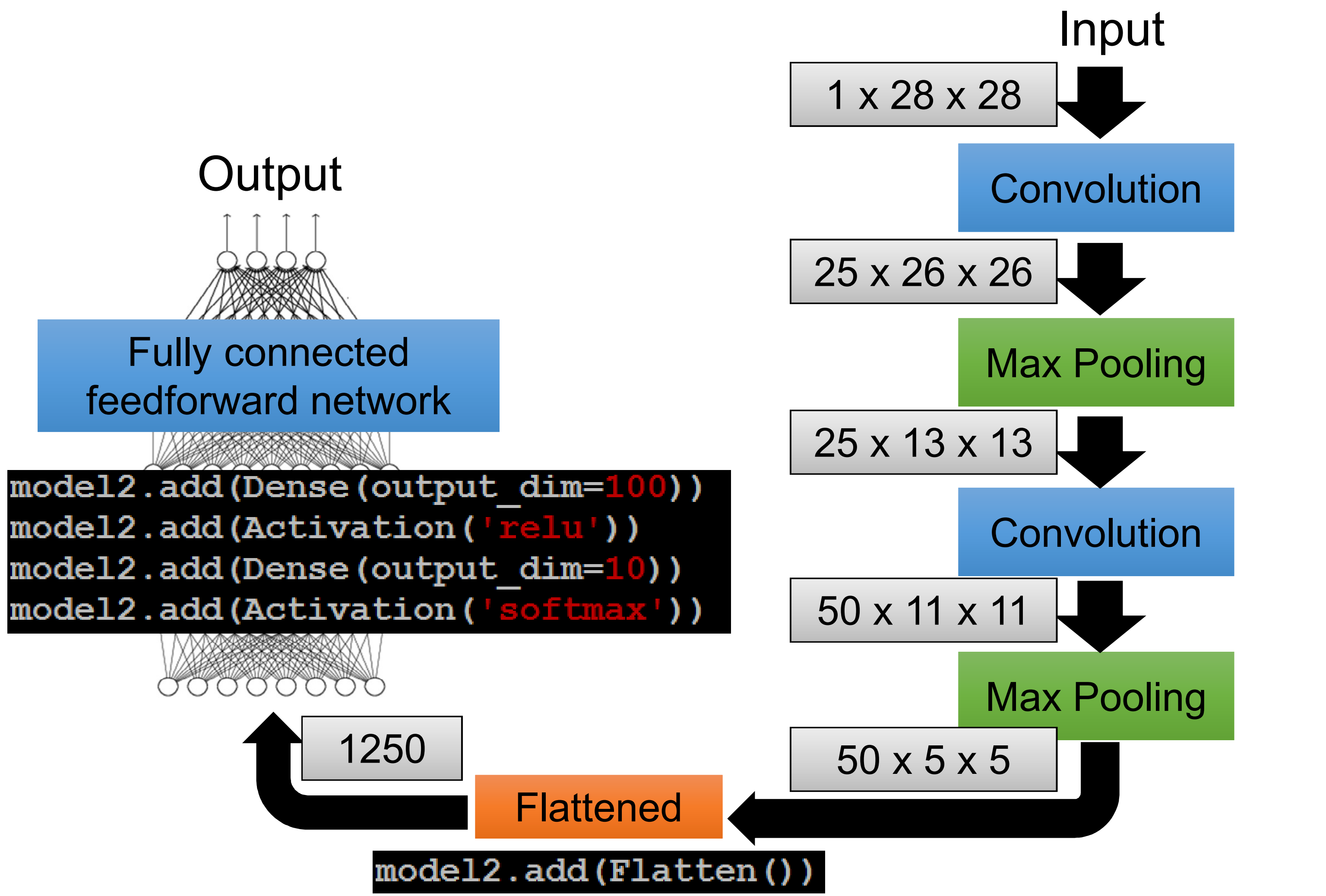
```
model2.add(MaxPooling2D((2,2)))
```

50 x 5 x 5

Image Source: internet

CNN in Keras

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




```
model2.add(Dense(output_dim=100))
model2.add(Activation('relu'))
model2.add(Dense(output_dim=10))
model2.add(Activation('softmax'))
```

```
model2.add(Flatten())
```

Image Source: internet

Object Detection using CNN

Classification

CAT

Classification + Localization = Detection

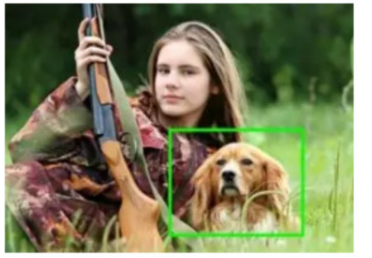
CAT

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)

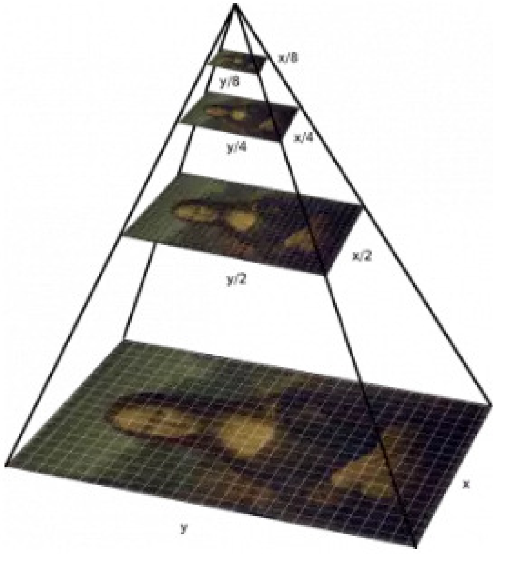
Problem ➡ Solution

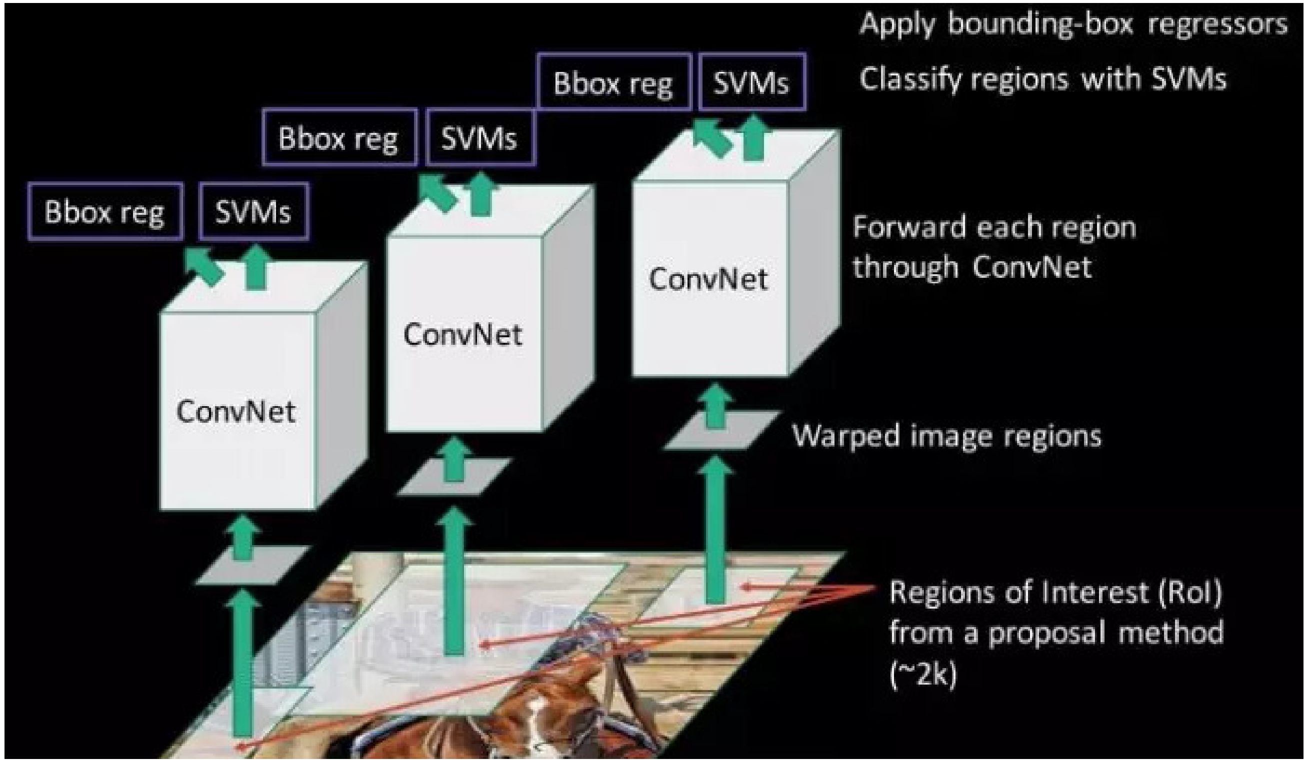
- Resize the image at multiple scales
- Most commonly, the image is downsampled(size is reduced)
- On each of these images, a fixed size window detector is run.
- Now, all these windows are fed to a classifier to detect the object of interest



Small sized object

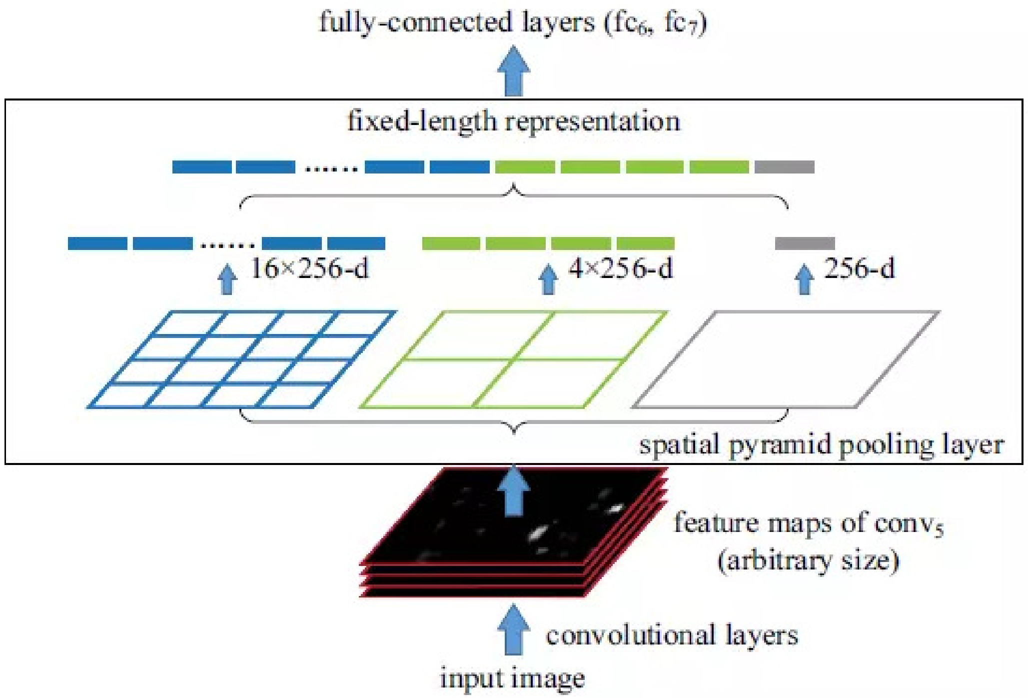
Big sized object. What size do you choose for your sliding window detector?





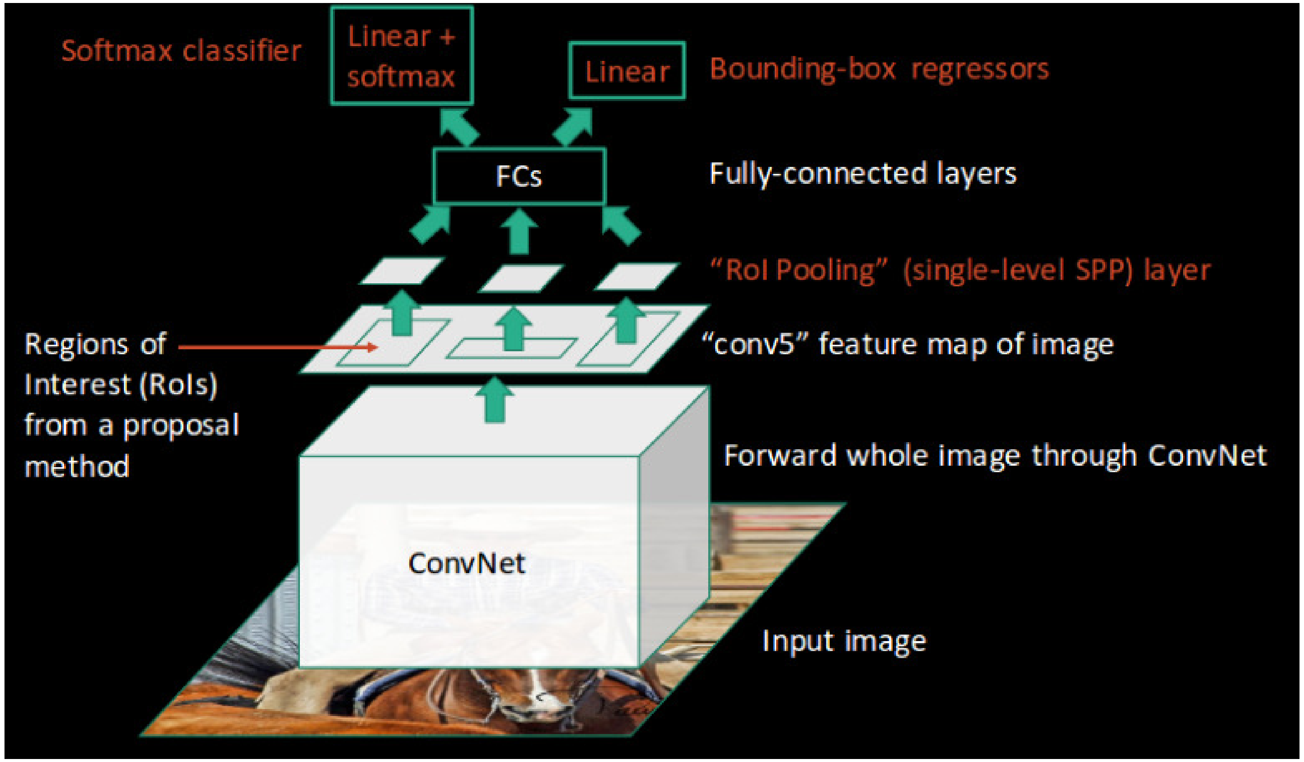
Region-based Convolutional Neural Networks(R-CNN)

- 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.



- 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

Spatial Pyramid Pooling(SPP-net)

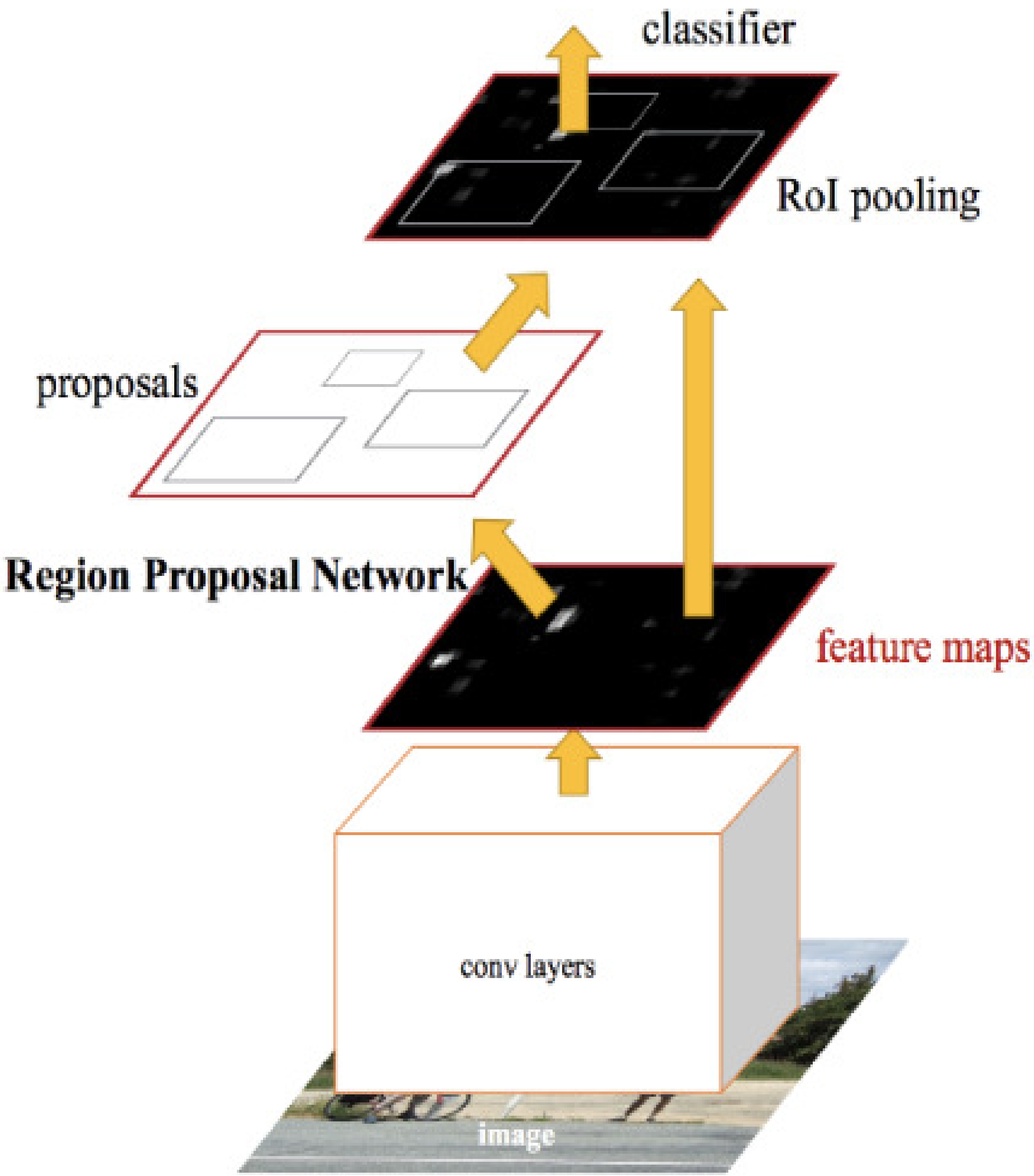


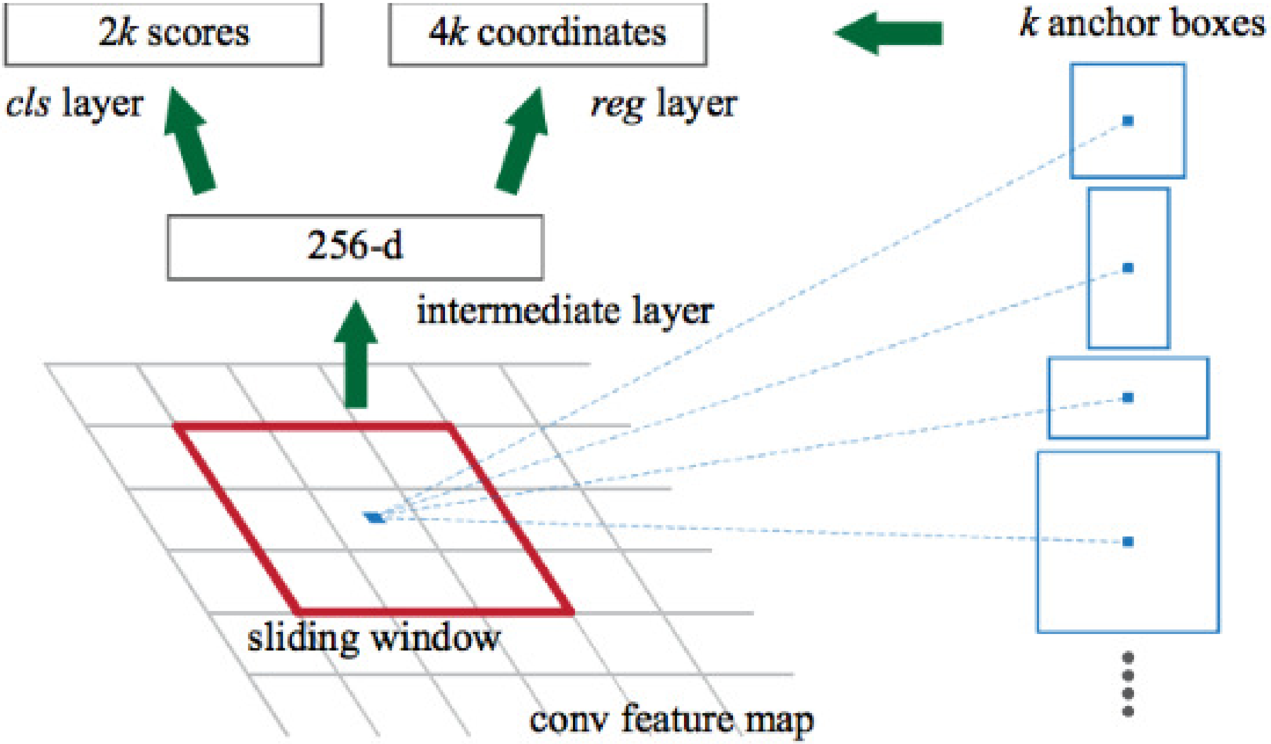
Fast R-CNN

- 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.

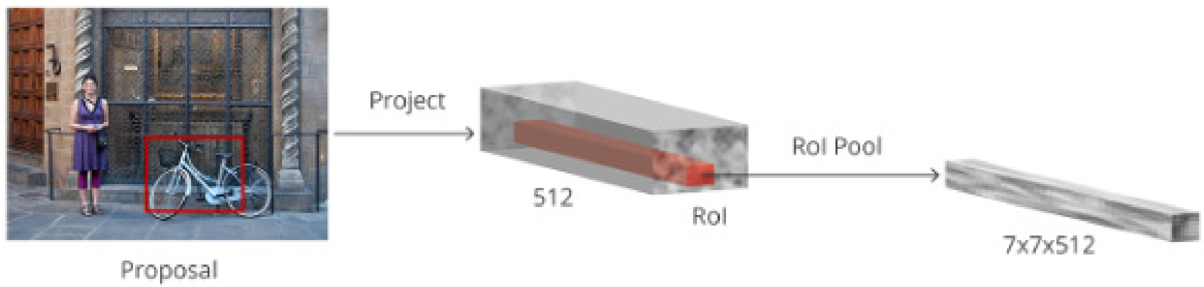




Region Proposal Network (RPN)

- 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

Region Proposal Network (RPN)



- We now have bounding boxes of different shapes and sizes which are passed on to the RoI pooling layer
- It extracts fixed sized feature maps for each anchor
- These feature maps are passed to a fully connected layer
- It has a softmax and a linear regression layer
    - Classifies the object
    - predicts the bounding boxes for the identified objects

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.

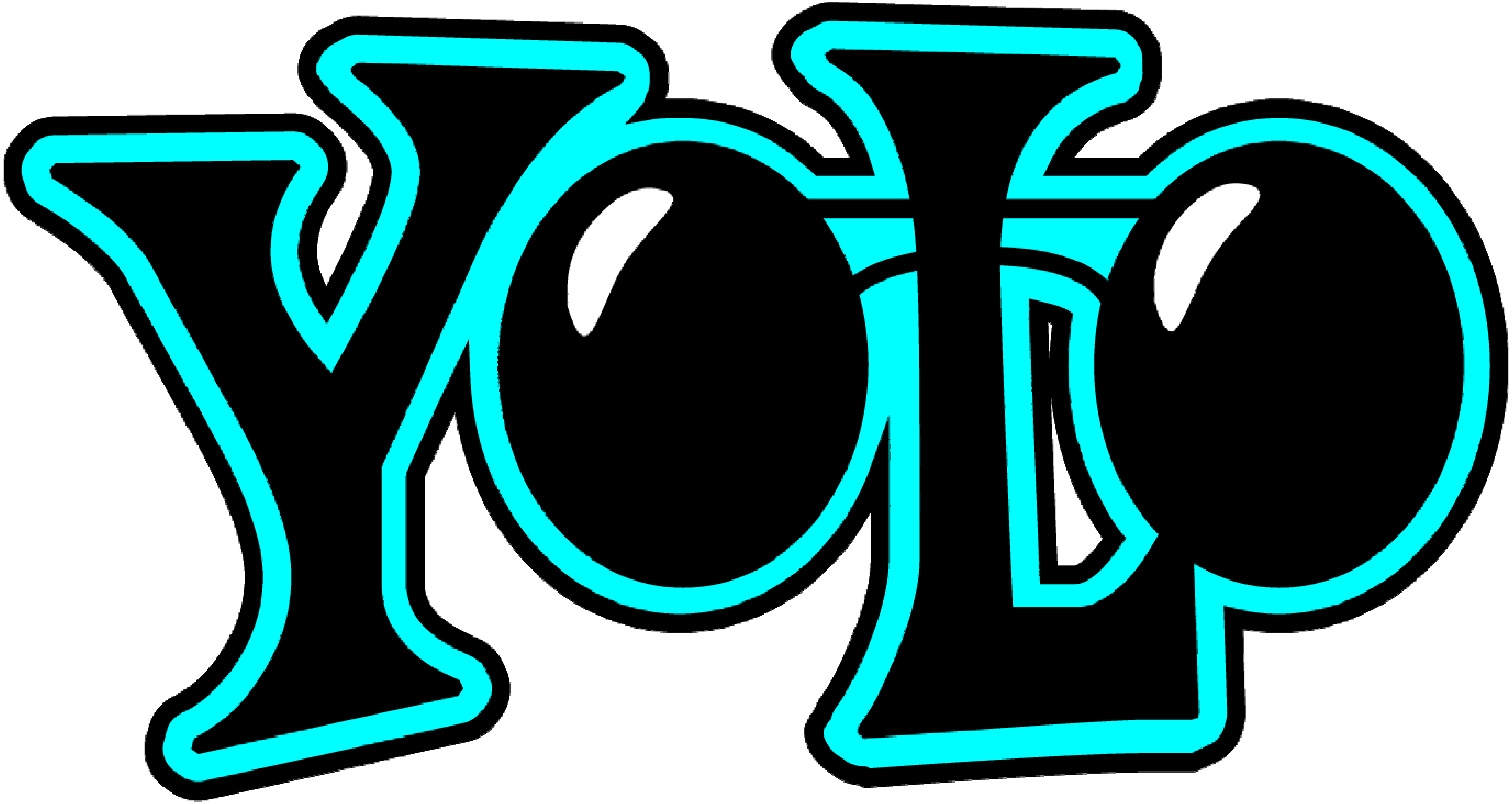
Two stages and Single stage Object Detectors

Two stage Detectors

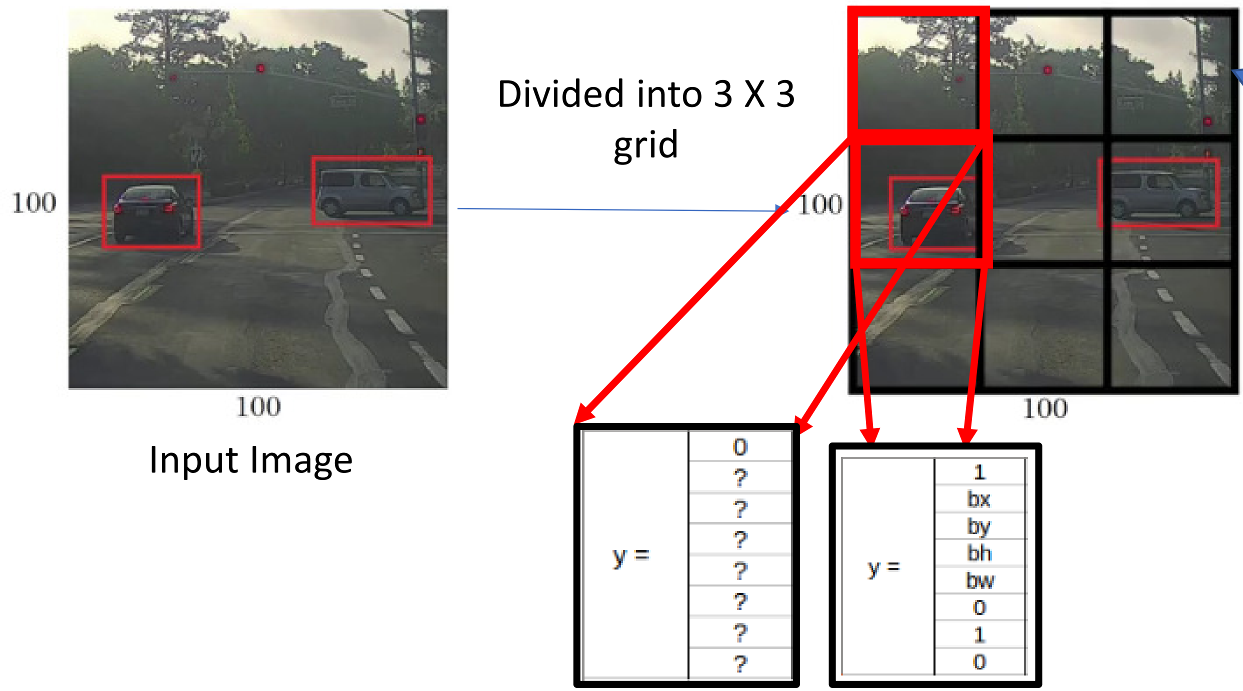
- first generates so-called region proposals — areas of the image that potentially contain an object
- Then it makes a separate prediction for each of these regions
- Examples : R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN

One stage Detectors

- These models skip the explicit region proposal stage but apply the detection directly on dense sampled areas
- Examples: Single Shot Detector (SSD), YOLO family


YOLO

How does YOLO Framework Function



- Image classification and localization are applied on each grid
- Suppose We have 3 classes. Let's say the classes are Pedestrian, Car, and Motorcycle, respectively. So, for each grid cell, the label y will be an eight-dimensional vector

y =	pc
	bx
	by
	bh
	bw
	c1
	c2
	c3

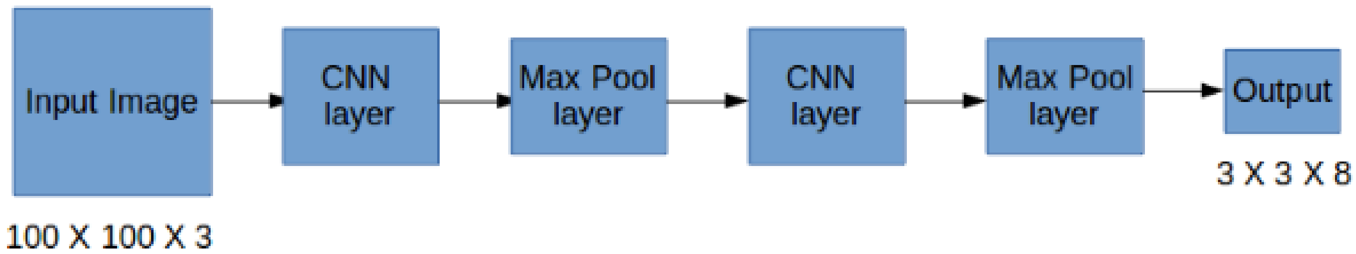


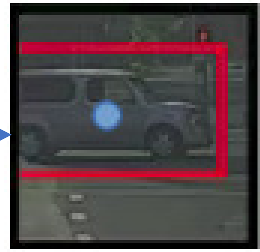
Image Source: deeplearning.ai

Bounding box in details

Grid contain bounding box

YOLO assign coordinates to all the grids

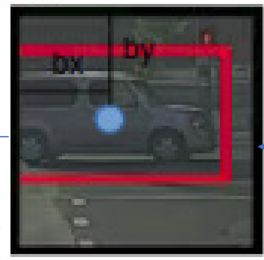
(0,0)



(1,1)

b_x, b_y are the x and y coordinates of the midpoint of the object with respect to this grid

(0,0)



(1,1)

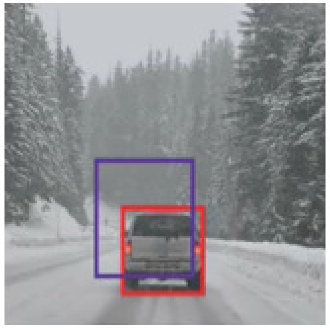
b_h :height of the bounding box / height of the grid
b_w : width of the bounding box / width of the grid

y =	1
	0.4
	0.3
	0.9
	0.5
	0
	1
	0

Image Source: deeplearning.ai

Intersection over Union and Non-Max Suppression

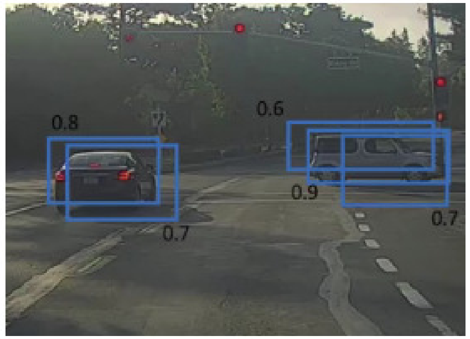
How can we decide whether the predicted bounding box is giving us a good outcome ?



IoU = Area of the intersection / Area of the union

If IoU>0.5, we accept predicted bounding box

Rather than detecting an object just once, they might detect it multiple times



Non-Max Suppression

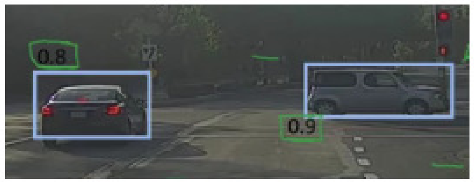
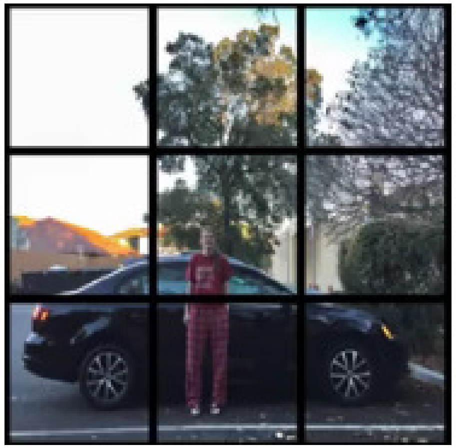


Image Source: deeplearning.ai

Anchor Box

what if there are multiple objects in a single grid?



midpoint of both the objects
lies in the same grid

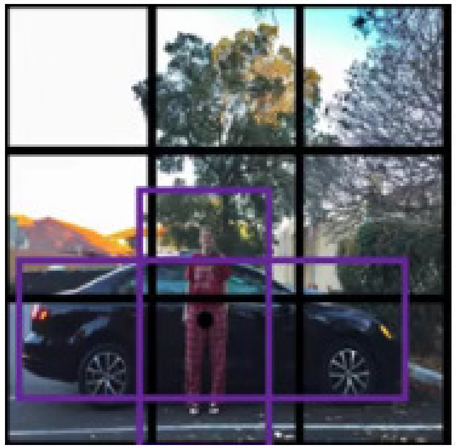
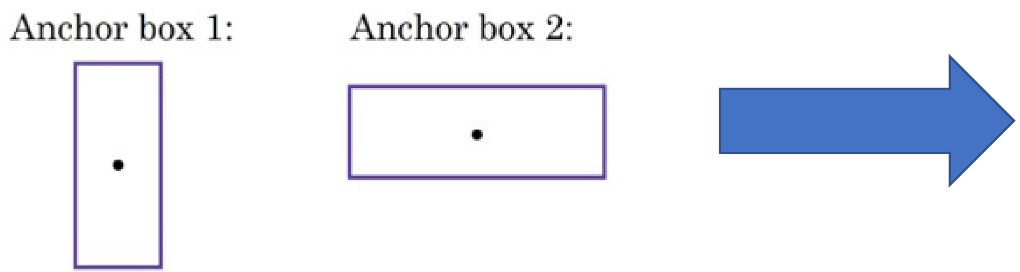
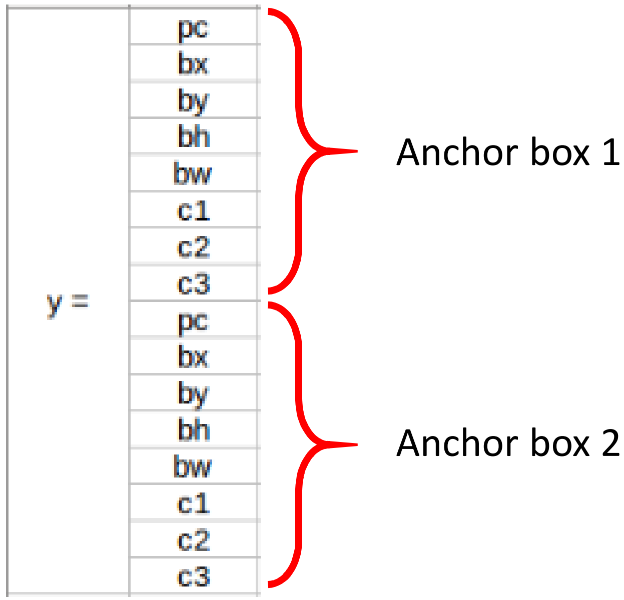


Image Source: deeplearning.ai

Anchor Box

what if there are multiple objects in a single grid?





- Since the shape of anchor box 1 is similar to the bounding box for the person, the latter will be assigned to anchor box 1 and the car will be assigned to anchor box 2
- The output in this case, instead of 3 X 3 X 8 (using a 3 X 3 grid and 3 classes), will be 3 X 3 X 16 (since we are using 2 anchors)

Image Source: deeplearning.ai

You Only Look Once

- Training
  - 3 X 3 grid with two anchors per grid
  - 3 different object classes
  - **y labels will have a shape of 3 X 3 X 16**
  - suppose if we use 5 anchor boxes per grid
  - number of classes has been increased to 5
  - target will be 3 X 3 X 10 X 5 = 3 X 3 X 50

- An input image of shape (608, 608, 3)
- Output volume of (19, 19, 425)
- 5 is the number of anchor boxes per grid
- **How many classes are there?**

Answer : 80 classes