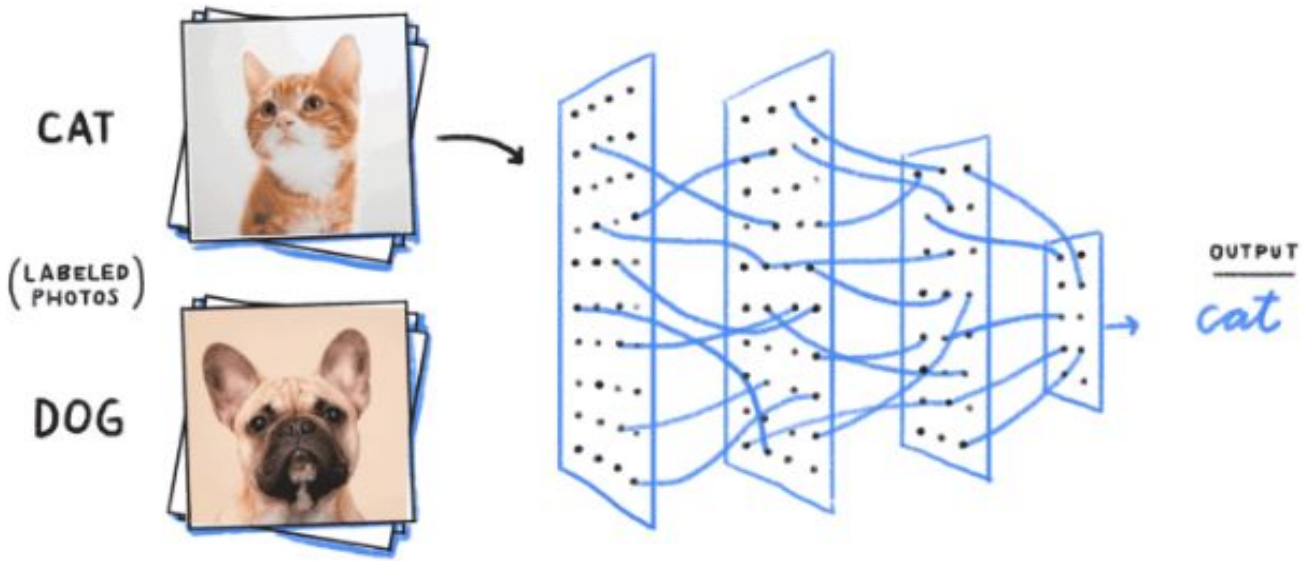


Introduction to Convolutional Neural Networks (CNNs)

Signal Processing Laboratory 5, EPFL
Presenter: Dr. Behzad Bozorgtabar
Email: behzad.bozorgtabar@epfl.ch

High Level Feature Detection

Nose
Eyes
Mouth

Wheels
License Plate
Headlights

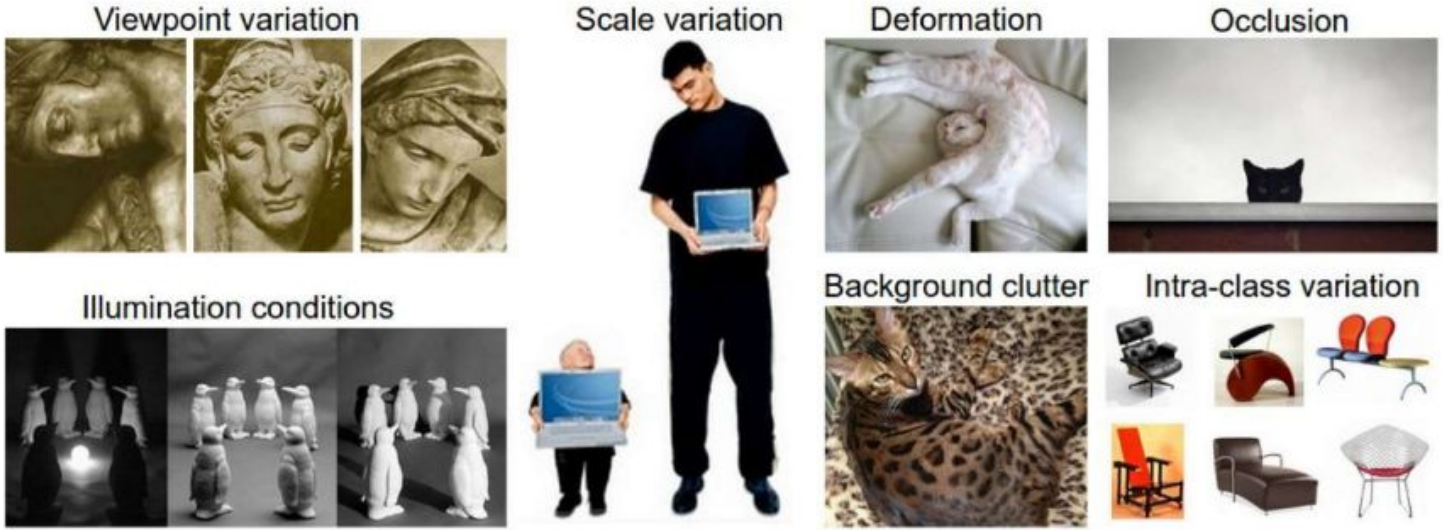
Door
Windows
Steps

Manual Feature Detection

Problems?

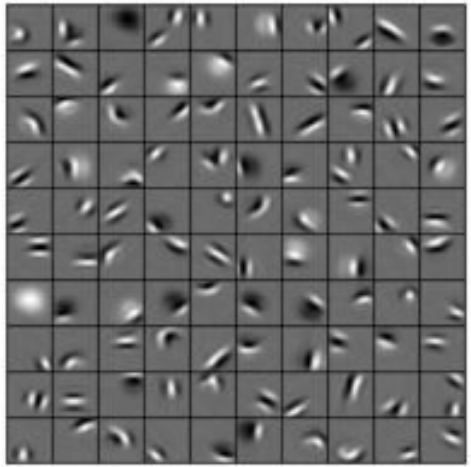
Manual Feature Detection

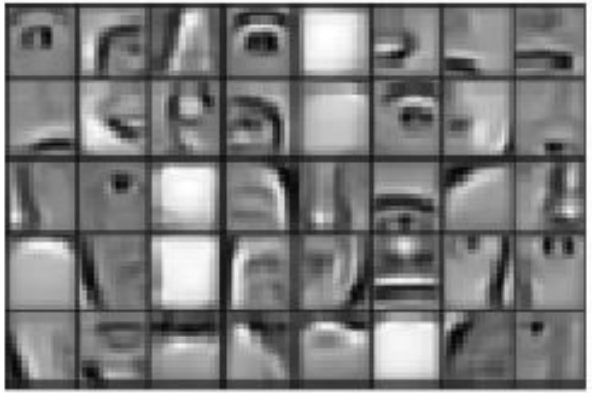
Domain knowledge → Define features → Detect features to classify

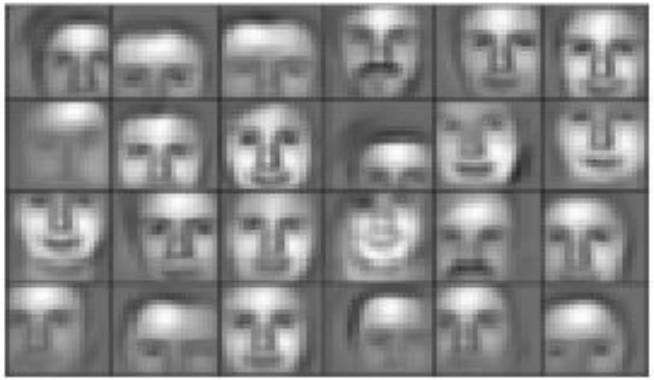


Learning Feature Representation

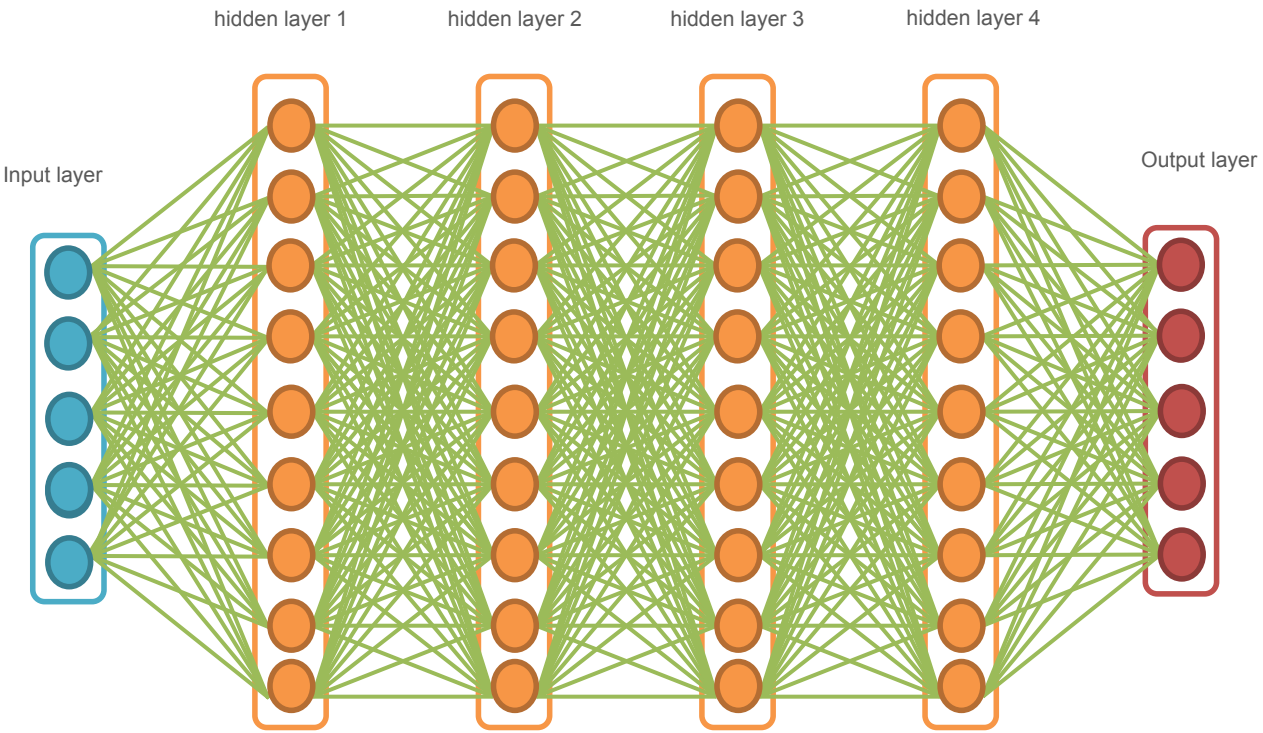
Can we learn a hierarchy of features directly from the data instead of hand designed features?







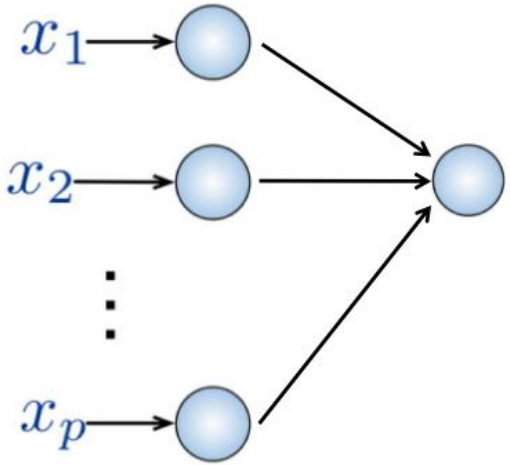
Fully Connected Neural Networks



Fully Connected Neural Networks

Input:

- 2D image
- Vector of pixel values



Fully Connected:

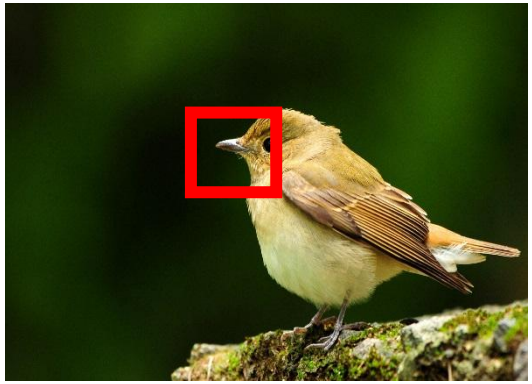
- Connect neuron in hidden layer to all neurons in input layer
- No spatial information!
- Too many parameters!

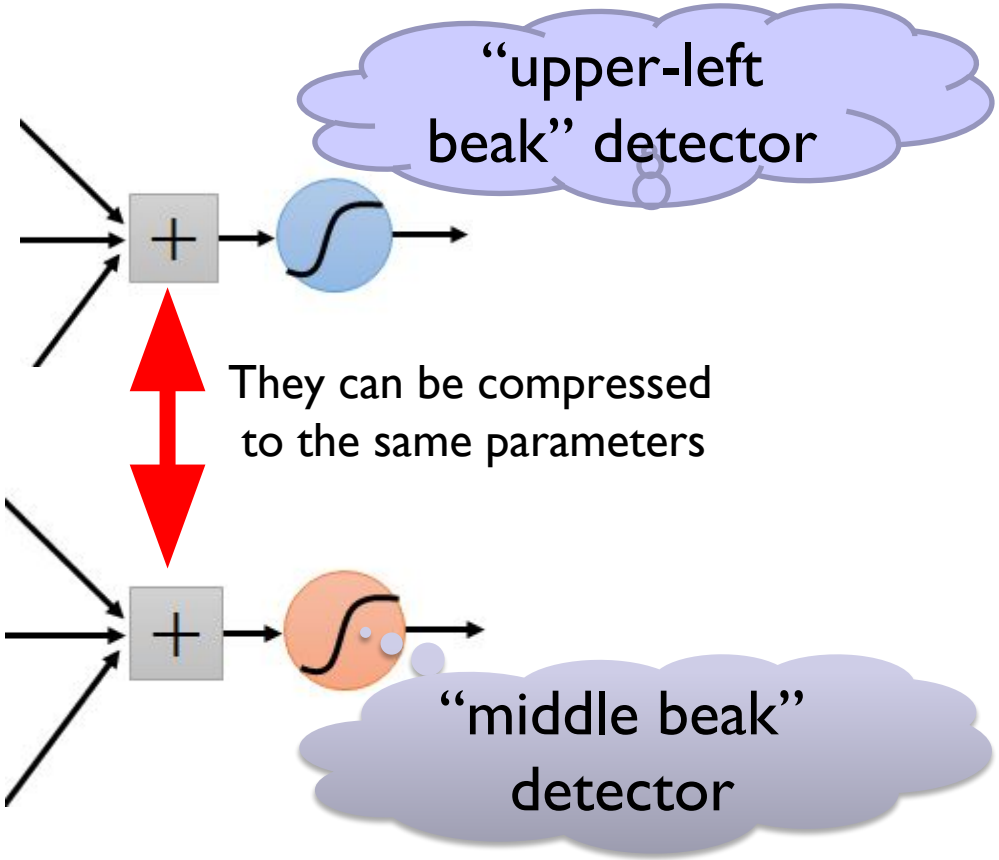
Can we find a better solution?

Local Feature Detector

- Some patterns are much smaller than the whole image
- Can represent a small region with fewer parameters?

Spatially Shared Parameters





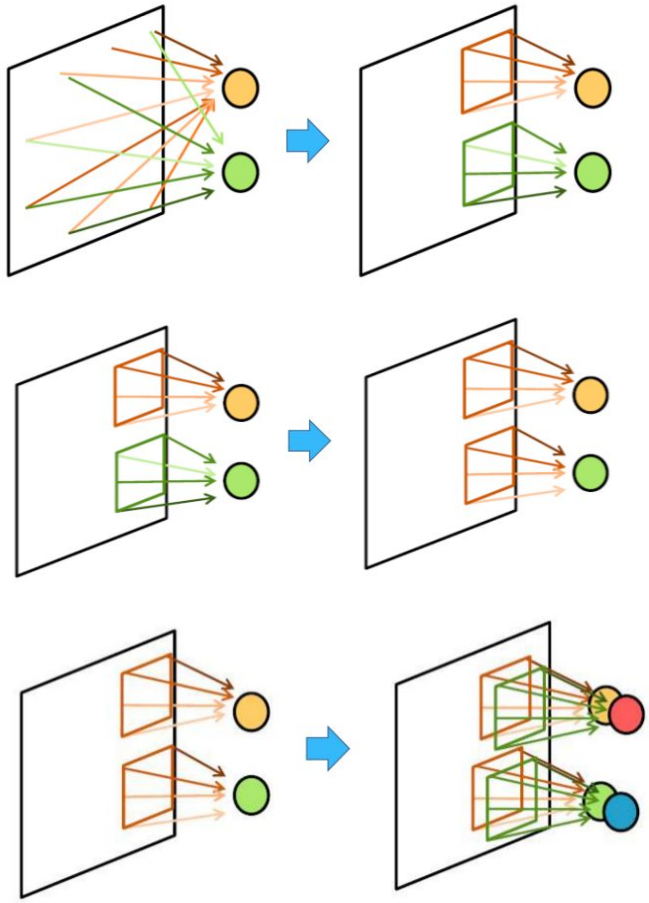
Applying Filters to Extract Features

1. Apply a set of filters to extract local features -> local connectivity

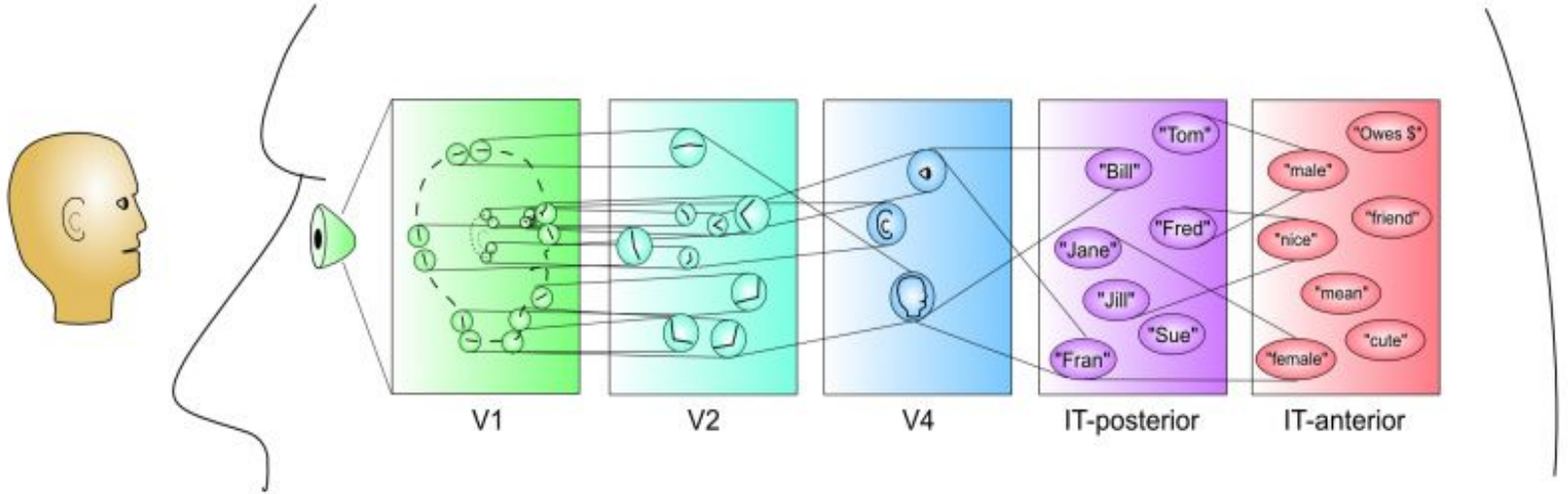
2. Spatially share parameters of each filter

3. Learn multiple filters

(still much fewer parameters than fully connect net)



What inspired Convolutional Networks?



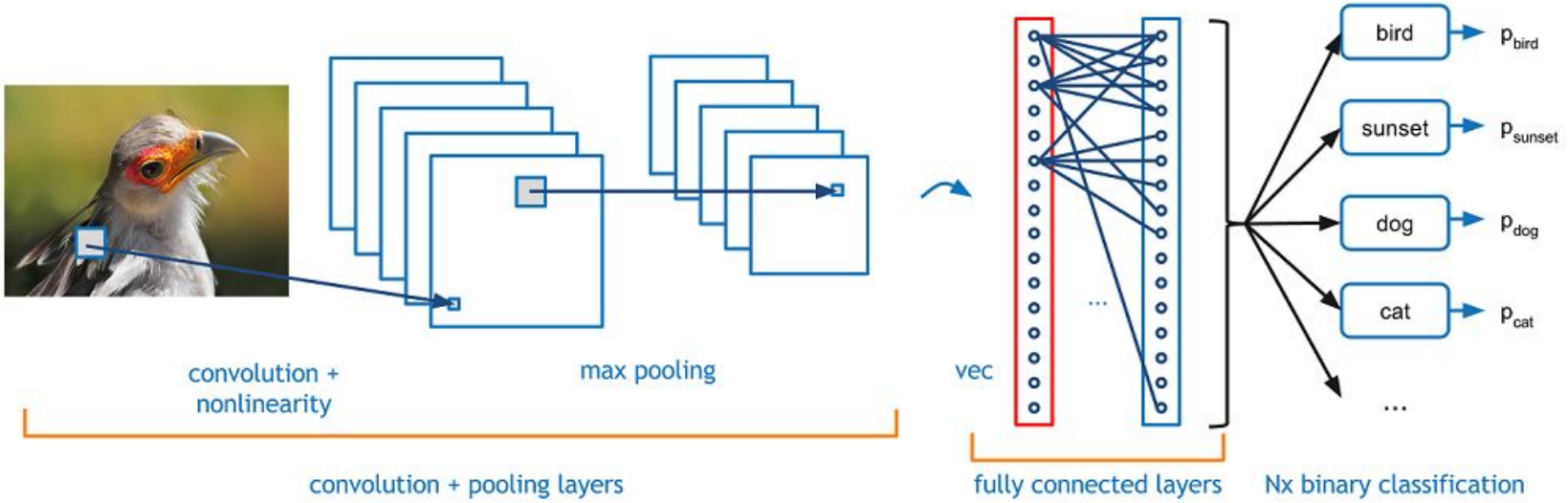
Inspired by **Hubel & Wiesel in 1962**

- ❏ Layered architecture of neurons in the brain
- ❏ Kunihiko Fukushima introduced CNN in 1980
- ❏ Yann LeCun used back-propagation to train CNN in 1989

Convolutional Neural Networks

(without the brain stuff)

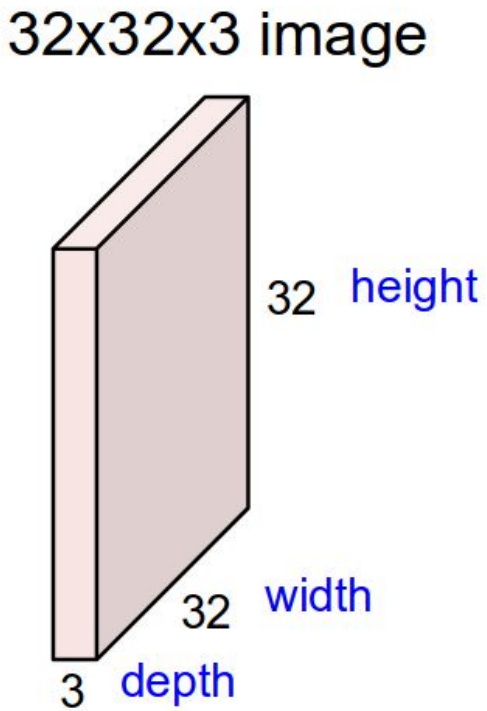
Convolutional Neural Networks (CNNs)



Convolution: Apply filters with learned weights to generate feature maps.
Non-linearity Activations: Often ReLU.
Pooling: Downsampling operation on each feature map.
Fully connected layers: Classification.

Convolution Layers

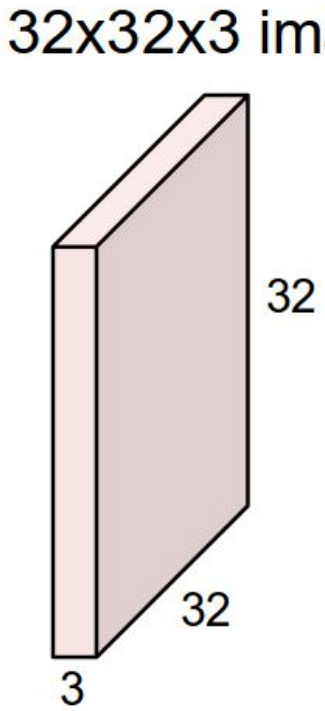
Convolution Layer



- Colored images introduces an additional ‘depth’ field
- The image constitutes a 3-dimensional structure called the **Input Volume**

Figure from : Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer

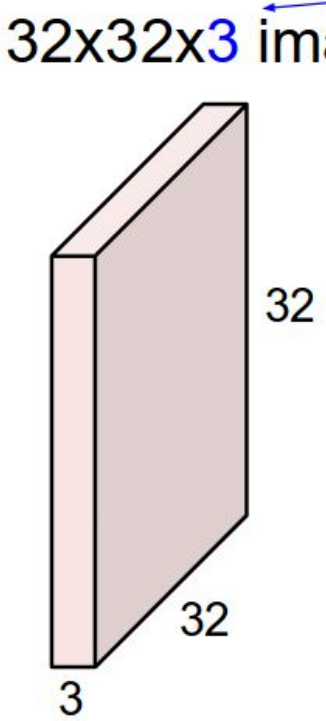


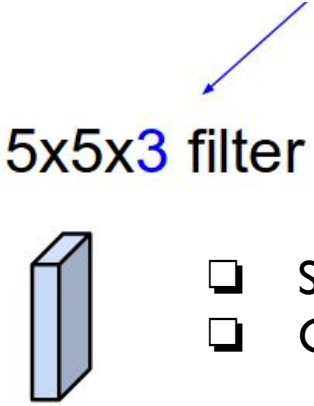
- Slide the filter over the image spatially
- Computing dot products

Figure from : Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer

Filters always extend the
full depth of the input volume

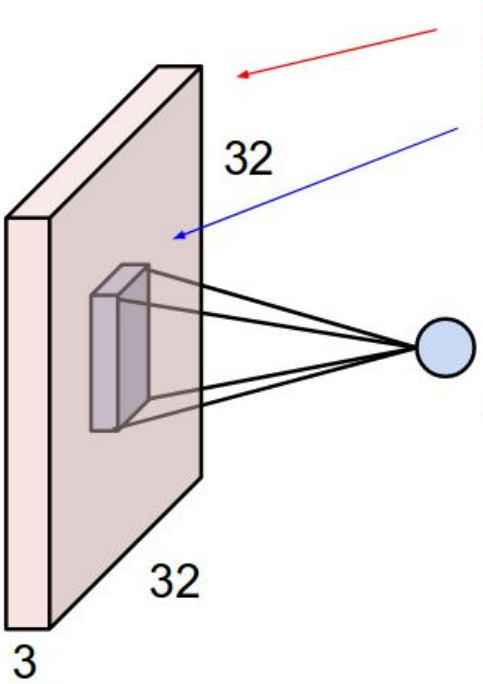




- Slide the filter over the image spatially
- Computing dot products

Figure from : Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer



One number
The result of taking dot product
(i.e. 5*5*3=75-dimensional dot product+bias)

$$w^T x + b$$

Figure from : Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer

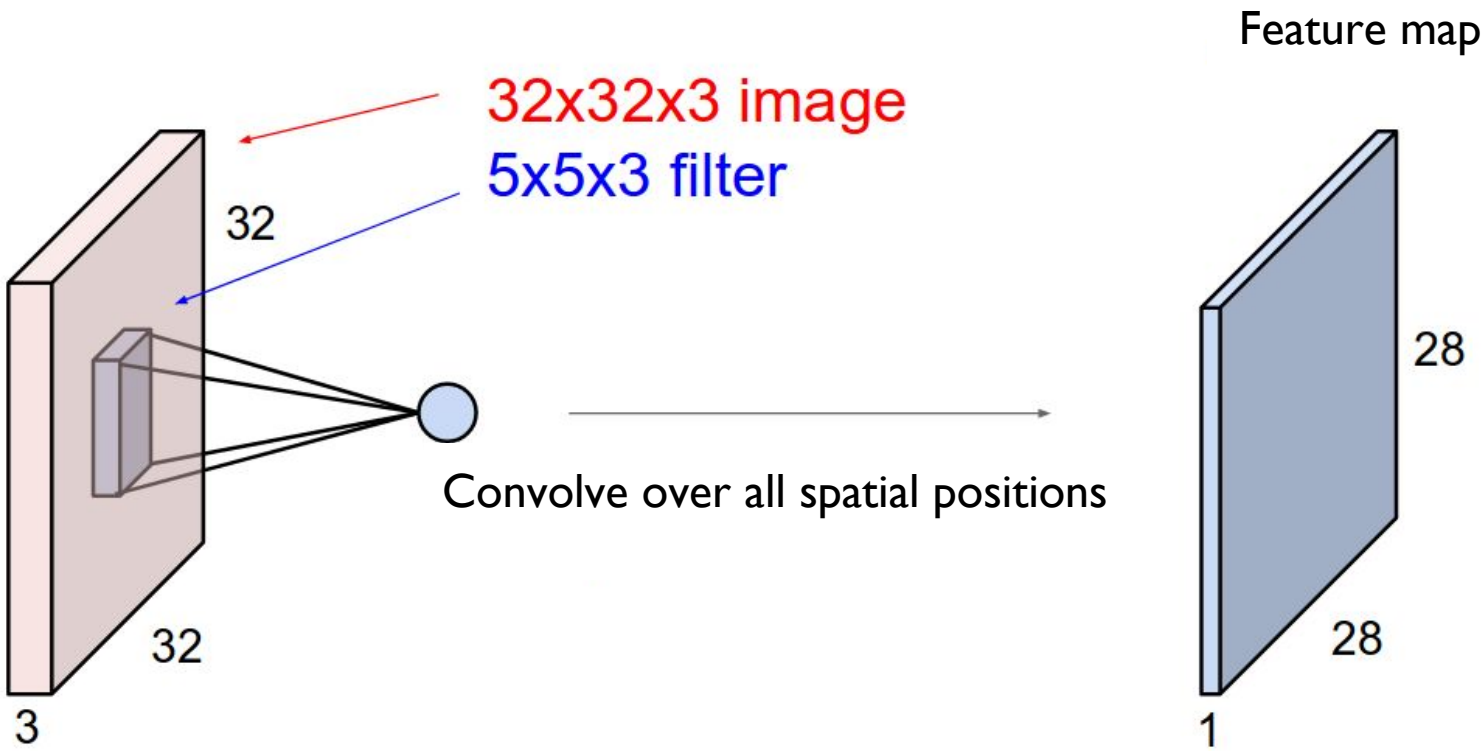


Figure from : Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer

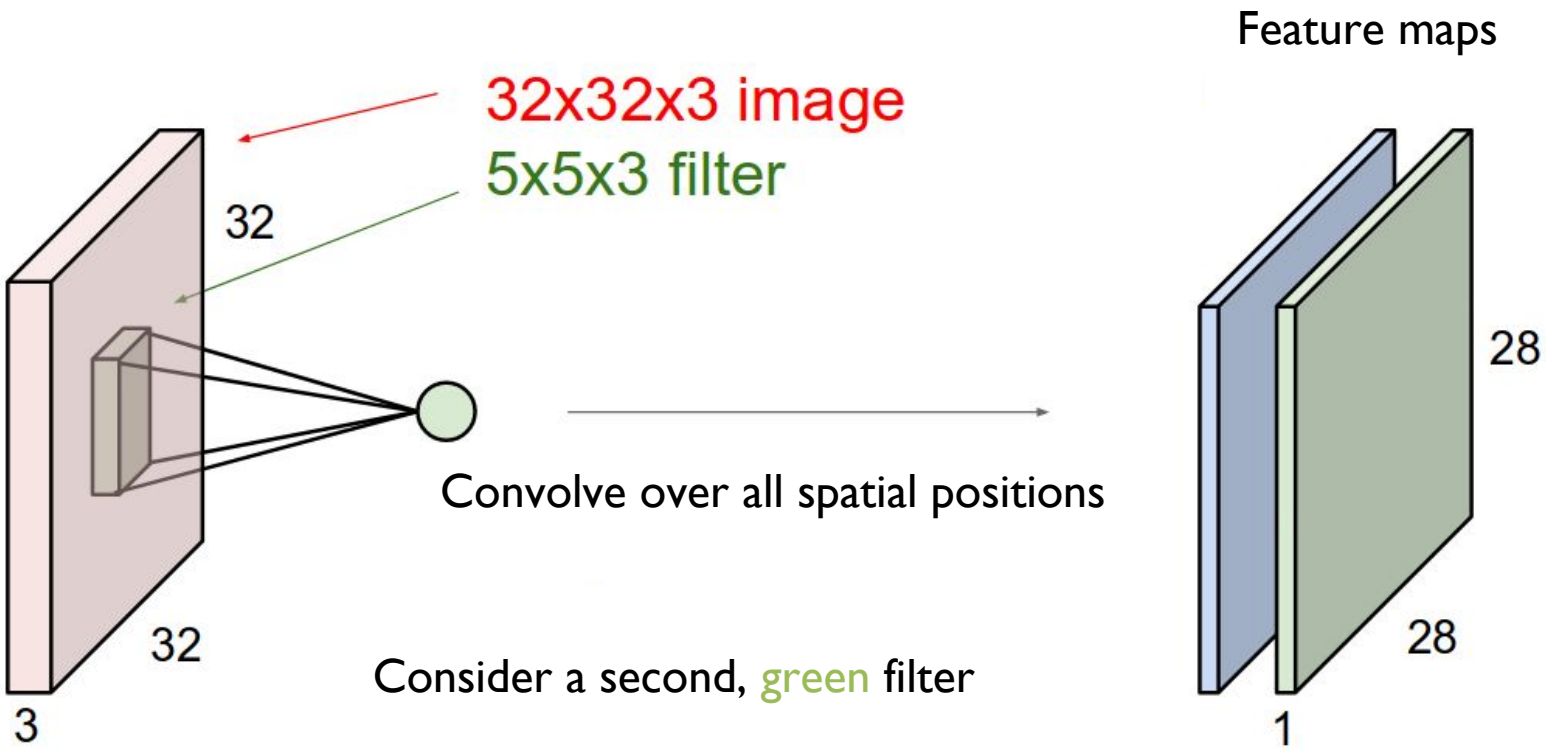
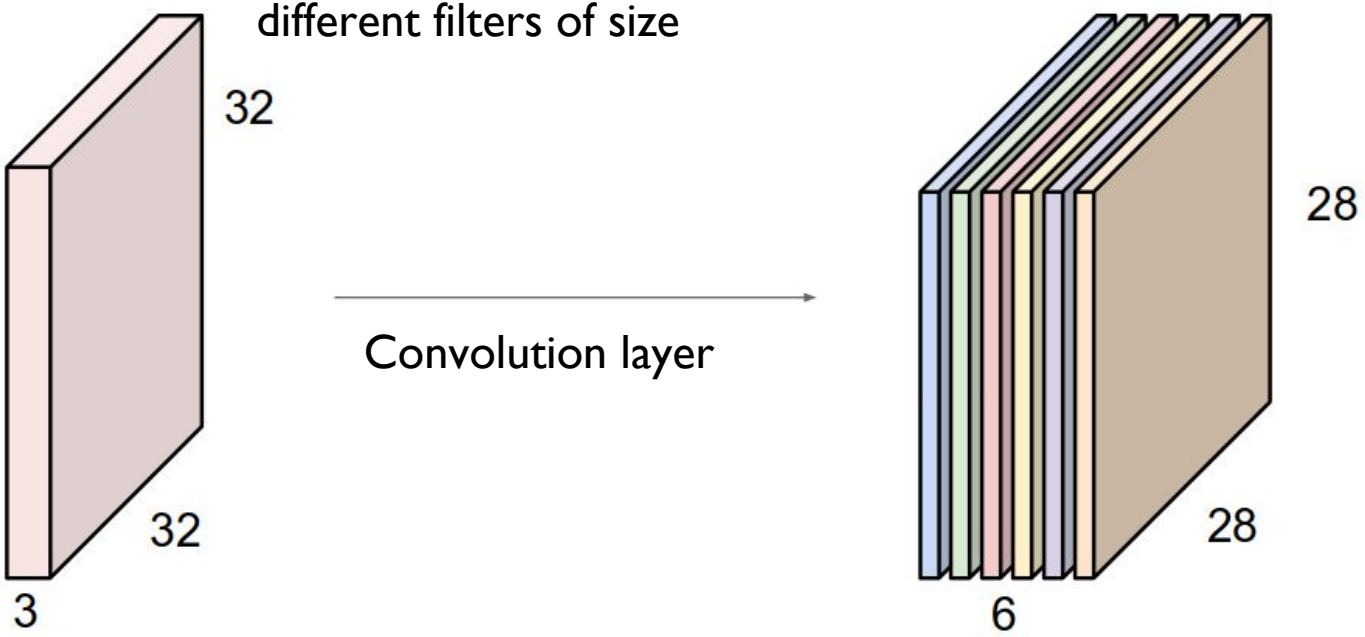


Figure from : Fei-Fei Li & Andrej Karpathy & Justin Johnson

Convolution Layer

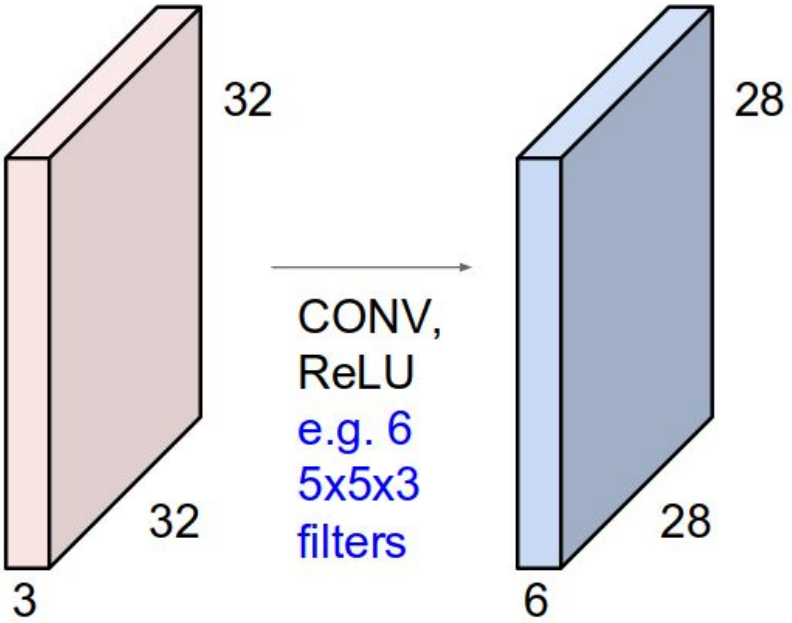
For example, if we have 6 different filters of size



We stack these up to get new volume of size

Figure from : Fei-Fei Li & Andrej Karpathy & Justin Johnson

ConvNet: Stack of Convolution Layers



ConvNet: Stack of Convolution Layers

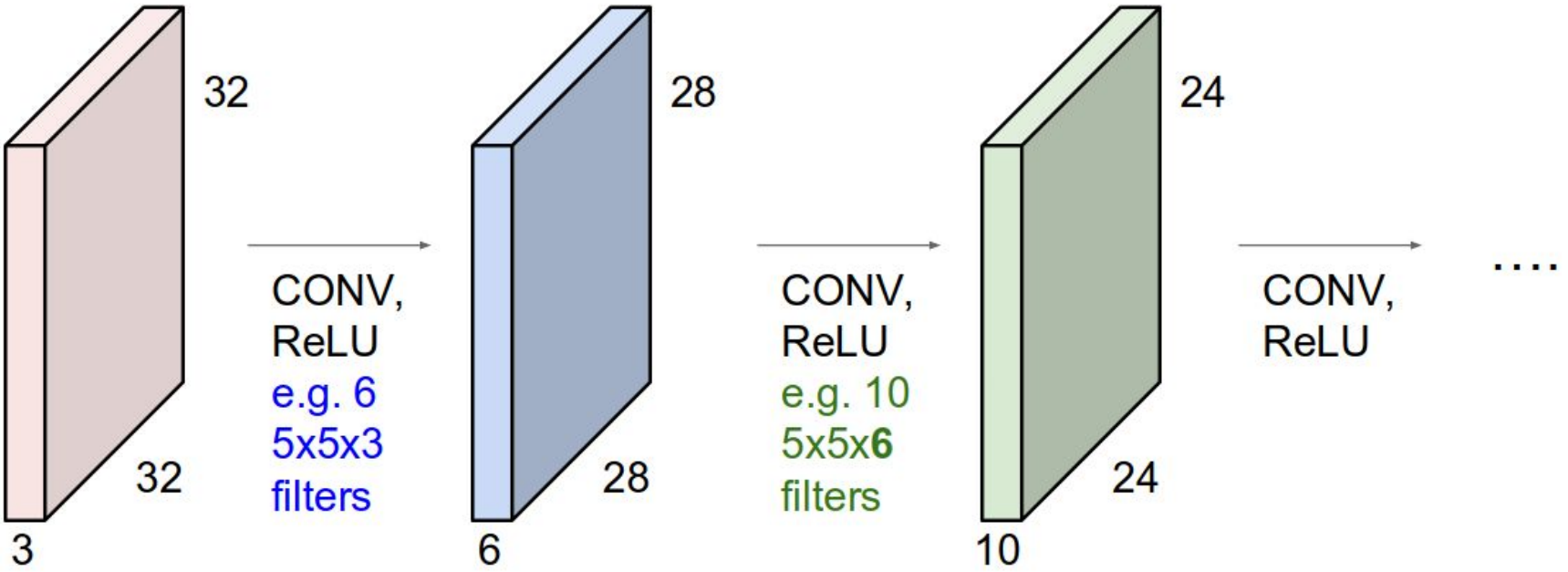


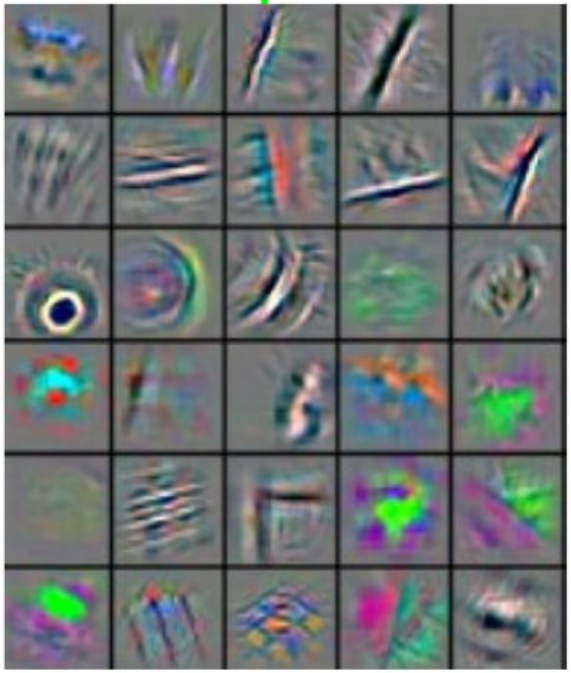
Figure from : Yann LeCun

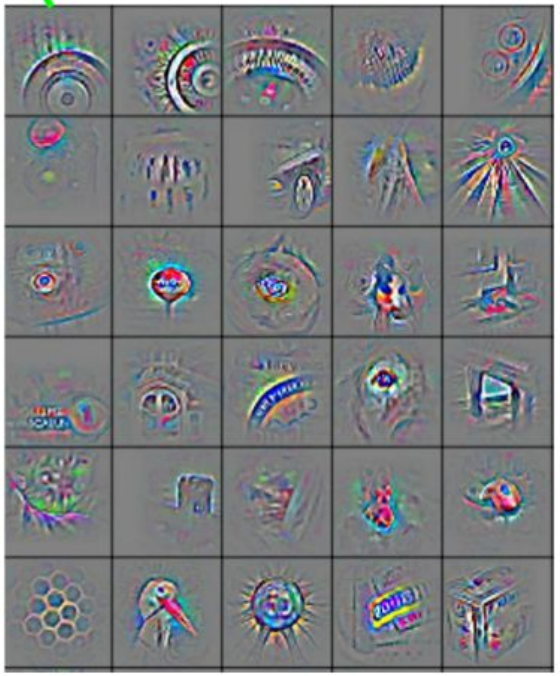
Low-Level Feature

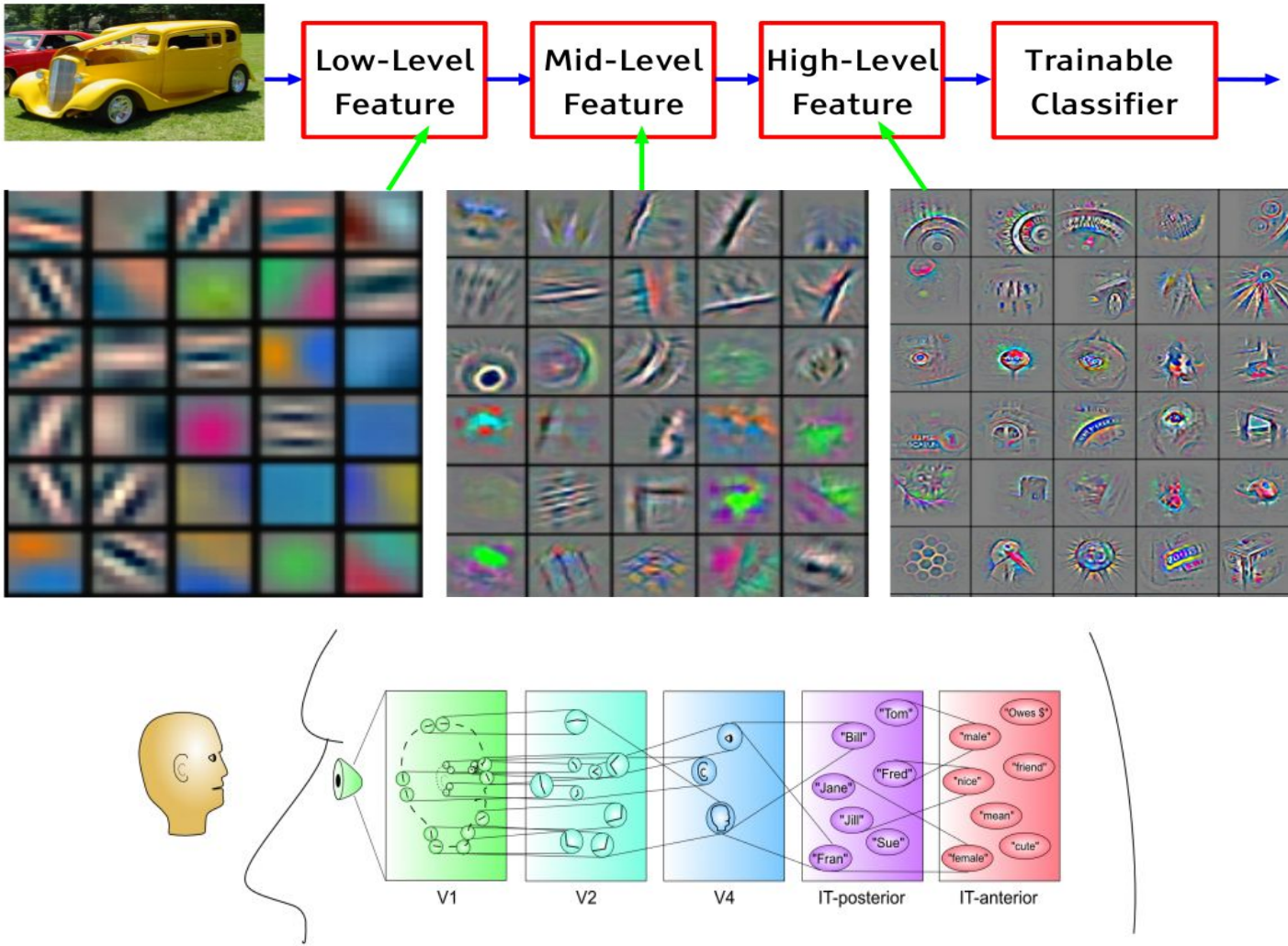
Mid-Level Feature

High-Level Feature

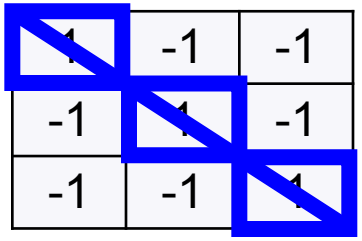
Trainable Classifier





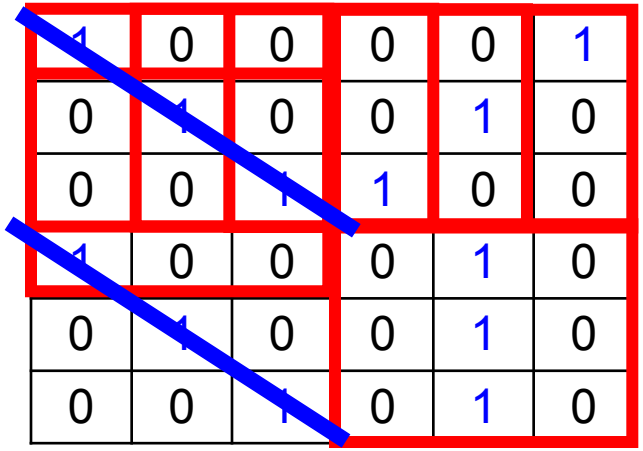

Low-Level Feature
Mid-Level Feature
High-Level Feature
Trainable Classifier
"Tom"
"Bill"
"Fred"
"Jane"
"Jill"
"Sue"
"Fran"
"Owes $"
"male"
"friend"
"nice"
"mean"
"cute"
"female"
V1
V2
V4
IT-posterior
IT-anterior

A closer look at convolution

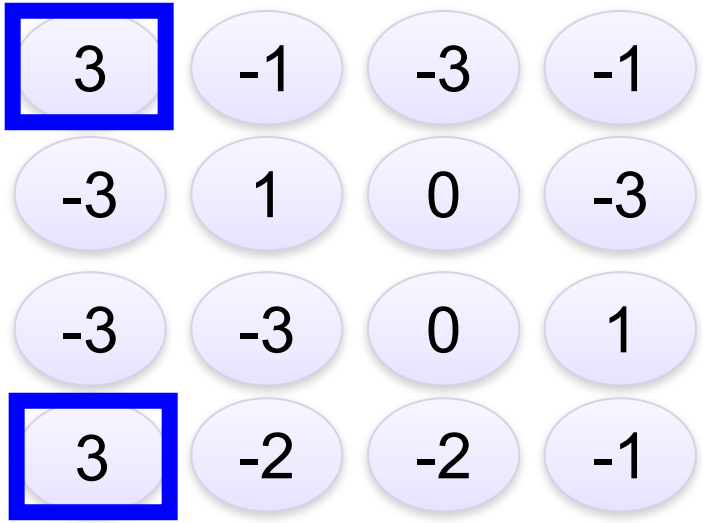


Filter 1

stride=1



6 x 6 image



A closer look at convolution

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

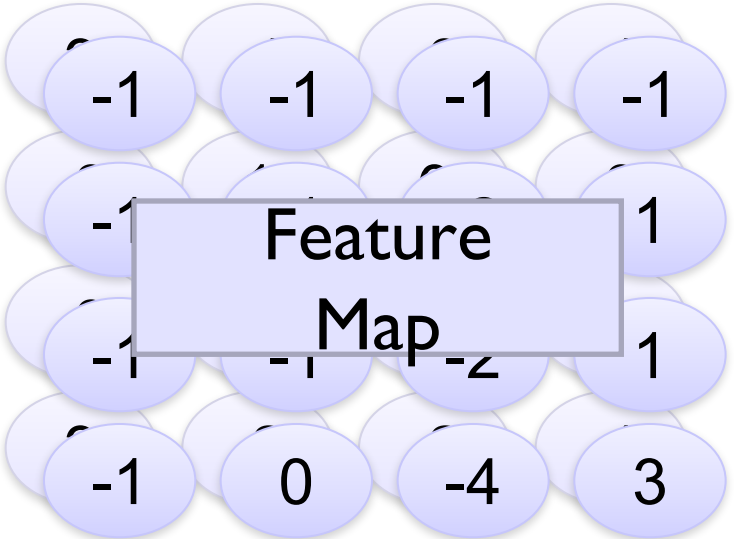
Filter 2

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

A closer look at convolution

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

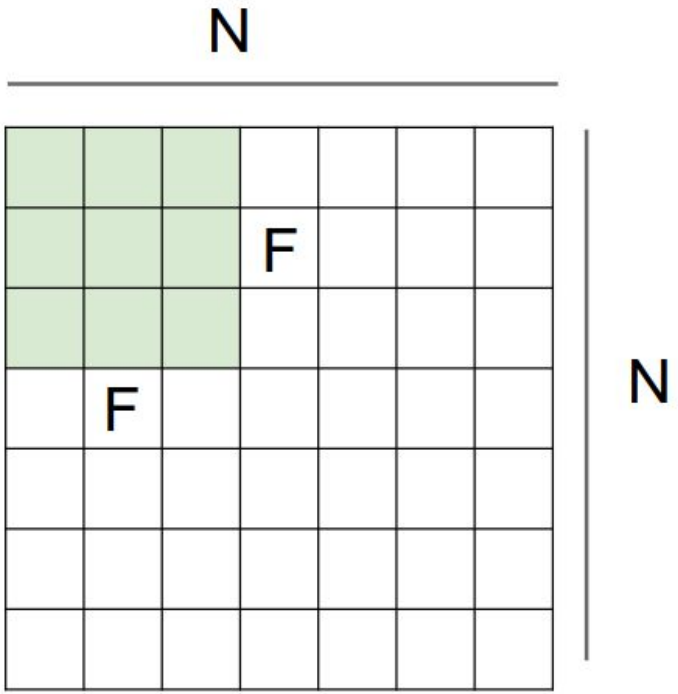
Filter 1

If stride=3

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

7 x 7 image

3 -1 ?



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 :\

Padding

If stride=3

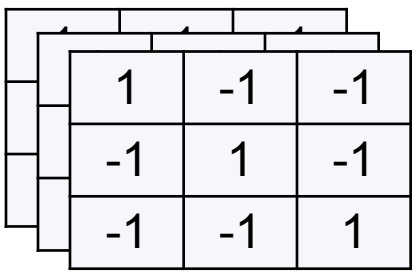
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

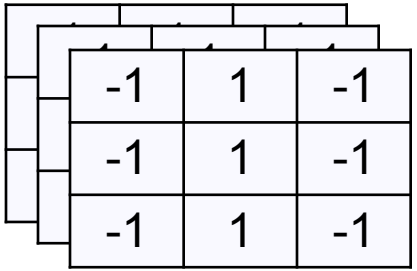
0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	0
0	0	1	0	0	1	0	0	0
0	0	0	1	1	0	0	0	0
0	1	0	0	0	1	0	1	0
0	0	1	0	0	1	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	1	0	1	0	1	0
0	0	0	0	0	0	0	0	0

2 1 0

Color images

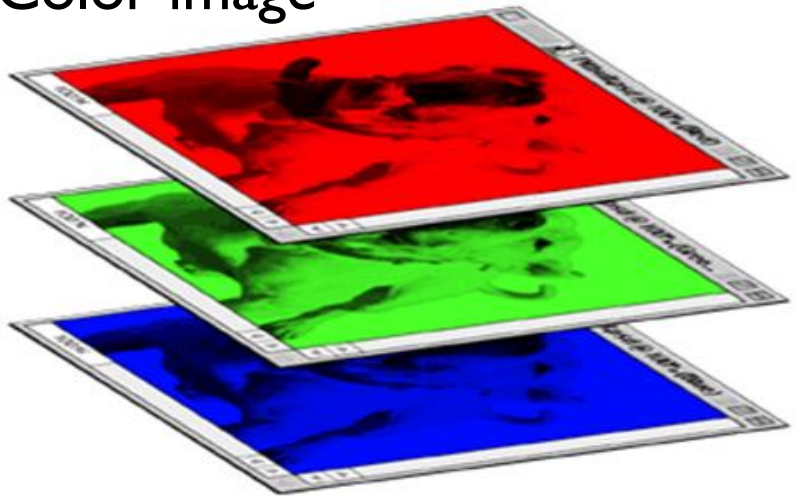


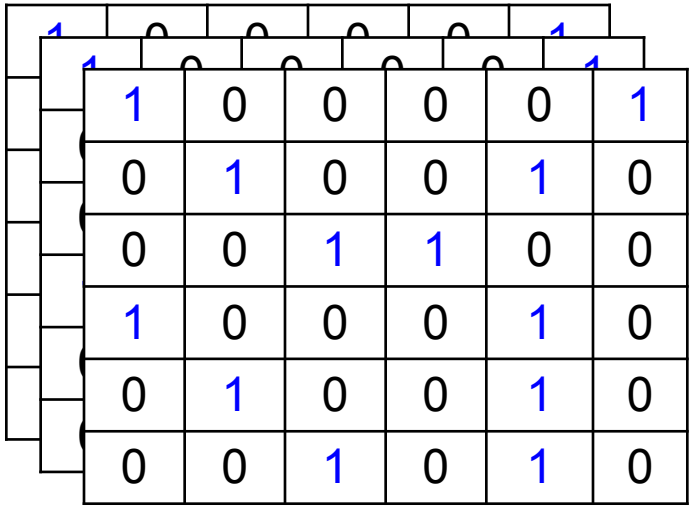
Filter 1



Filter 2

Color image

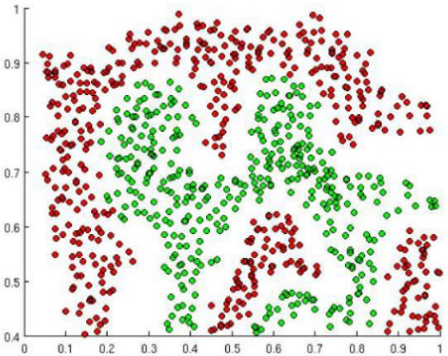




Activation Functions

Non-Linear Activation Functions

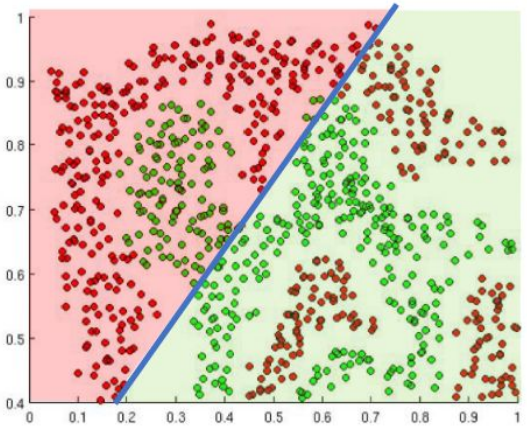
The purpose of activation functions is to introduce non-linearities into the network



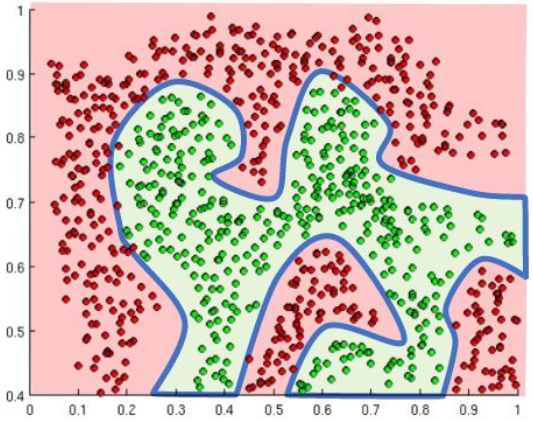
What if we wanted to build a Neural Network to distinguish green vs red points?

Non-Linear Activation Functions

The purpose of activation functions is to introduce non-linearities into the network



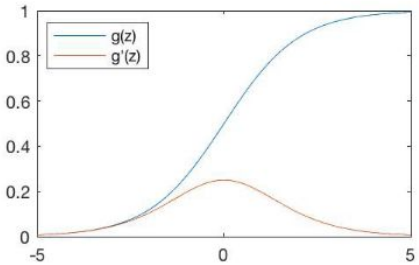
Linear Activation functions produce linear decisions no matter the network size



Non-linearities allow us to approximate arbitrarily complex functions

Common Activation Functions

Sigmoid Function

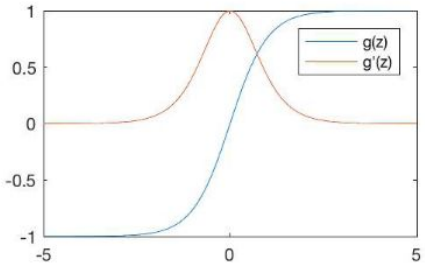


$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

`tf.nn.sigmoid(z)`

Hyperbolic Tangent

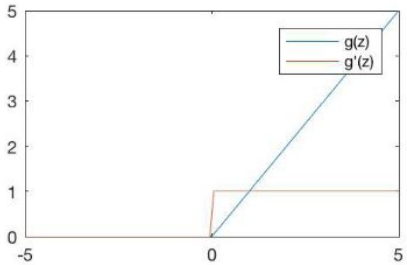


$$g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

`tf.nn.tanh(z)`

Rectified Linear Unit (ReLU)

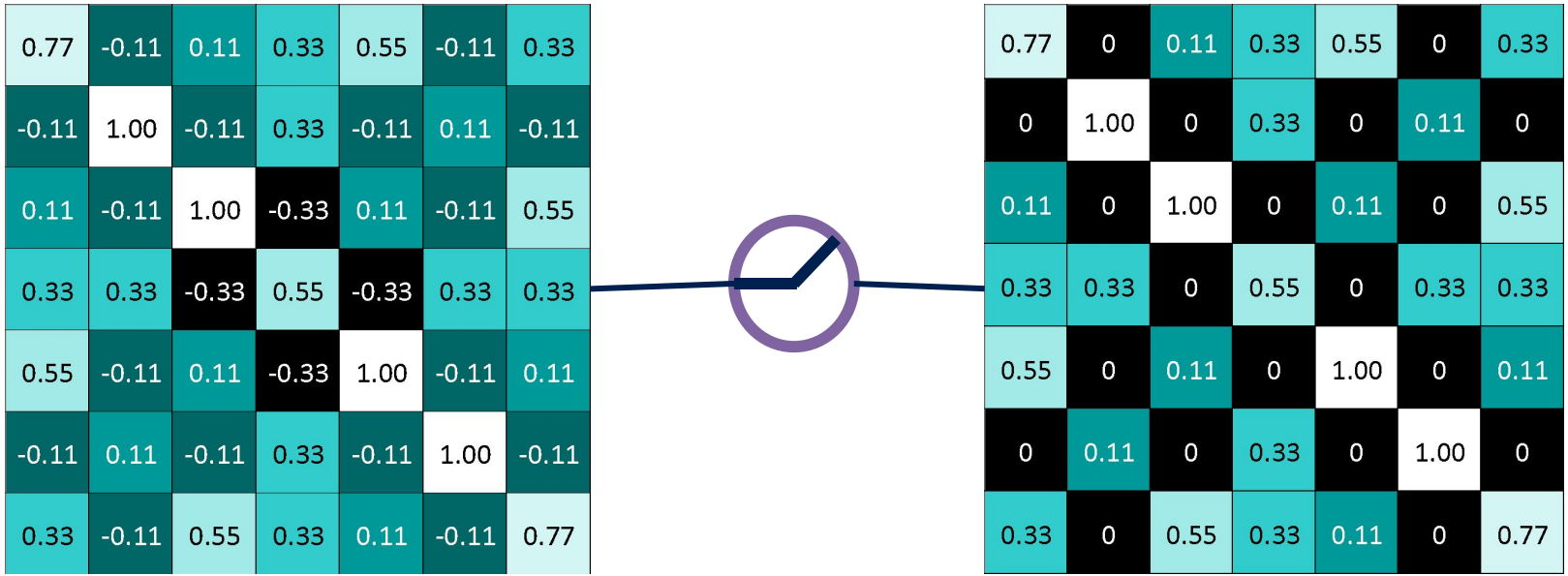


$$g(z) = \max(0, z)$$

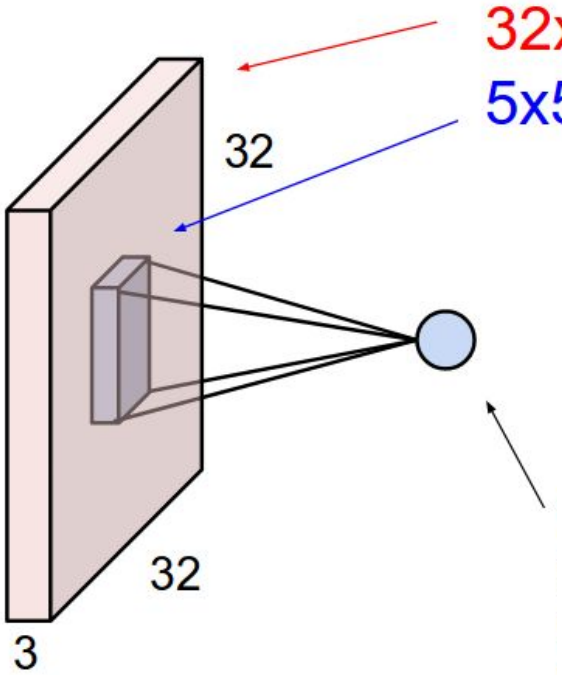
$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

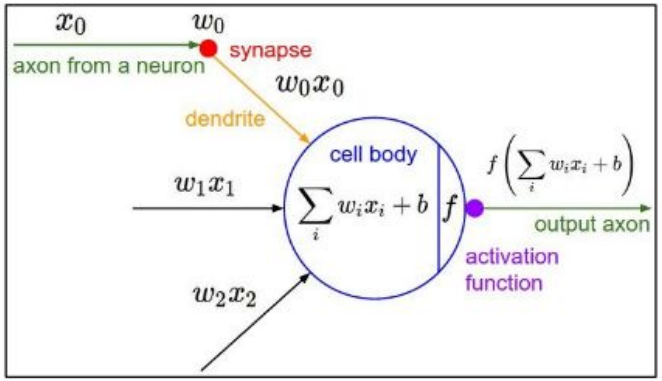
`tf.nn.relu(z)`

Rectified Linear Units (ReLUs)



The neuron view of Convolution Layer

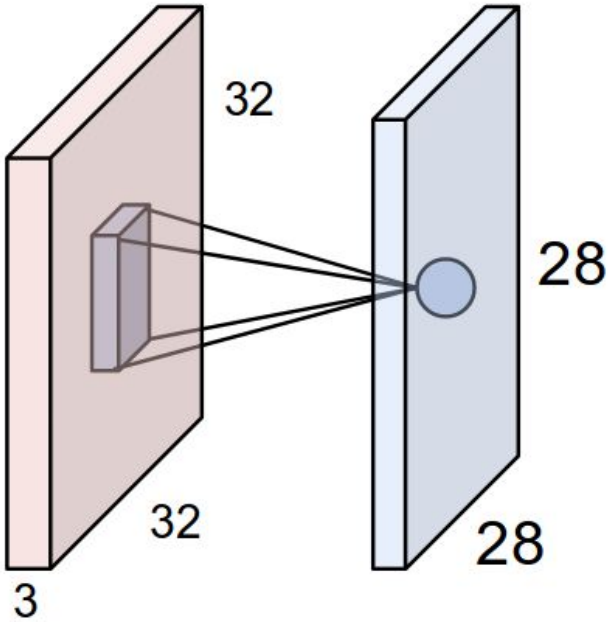




For a neuron:

- Take input from small patch (local connectivity)
- Compute dot product and apply bias
- Apply non-linear activation function

The neuron view of Convolution Layer

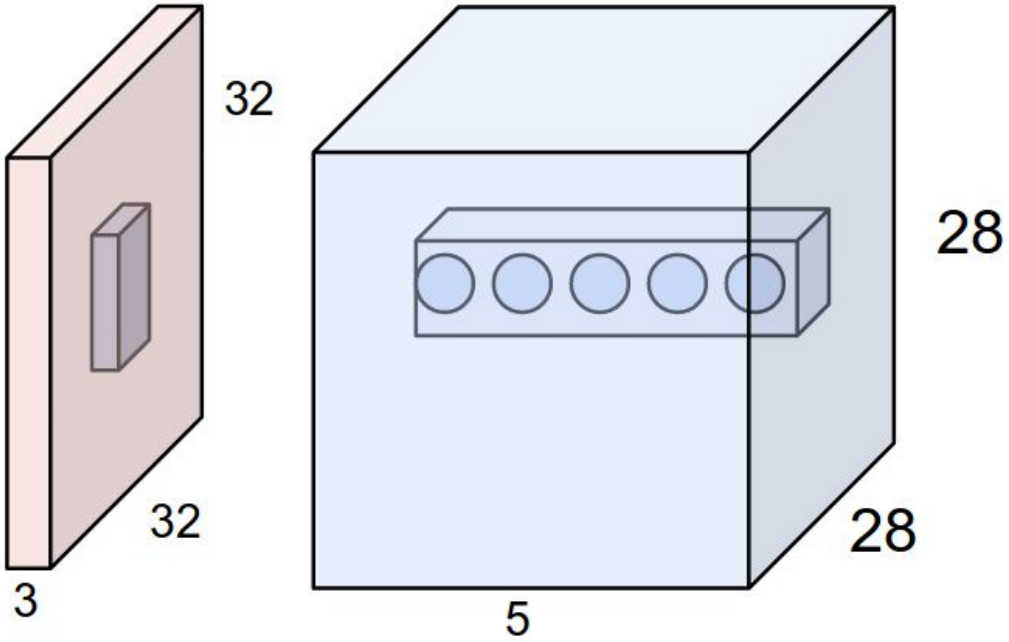


A feature map is a sheet of neurons:

- Each is connected to a small region in the input
- All of them share parameters
- Compute dot product and apply bias

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

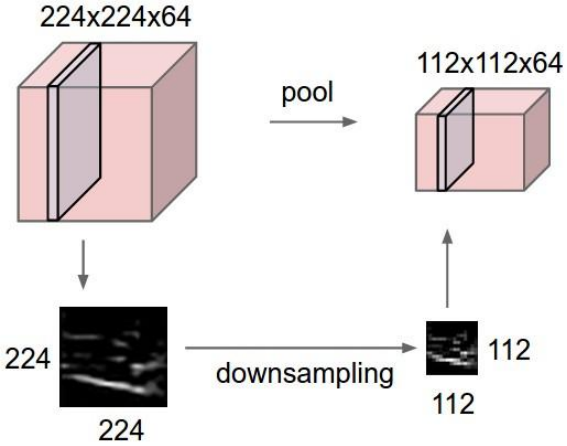
The neuron view of Convolution Layer



There will be 5 different neurons all looking at the same region in the input volume

Pooling Layers

The pooling Layer



Pooling layer:

- reduces computational complexity
- helps the network achieve spatial invariance

Max Pooling

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

Filter 1

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

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

Filter 2

-1	-1	-1	-1
-1	-1	-2	1
-1	-1	-2	1
-1	0	-4	3

Shift Invariance

1. **Convolution Layer** - the shared weights of the filter across all patches of the image - so the weights learnt will be invariant to position.

2. **Max Pooling Layer -** by taking the max value of the pixel, subsequent layers of the CNN don't care about the specific position in the patch that the max value was in.

Cat

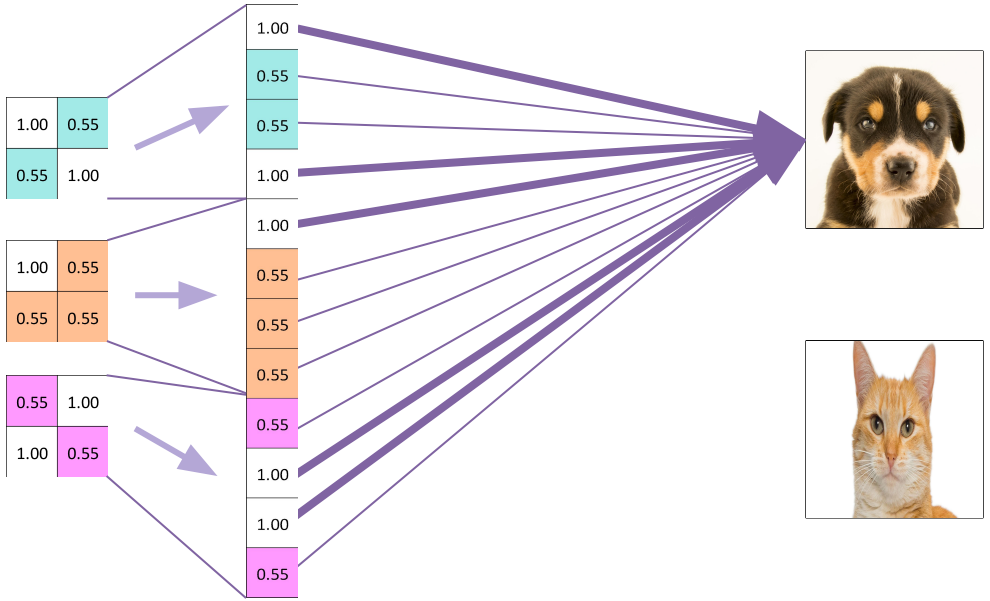
Cat

Fully Connected Layers

Fully Connected Layer

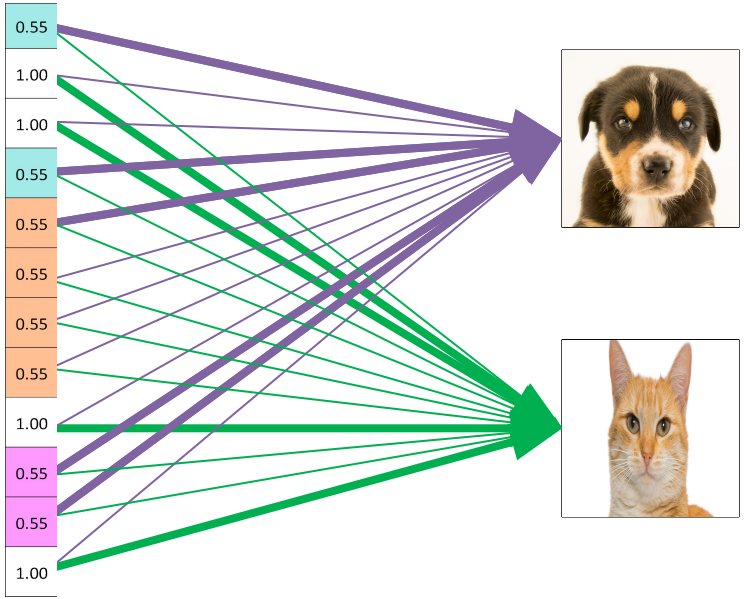
Every feature value gets a vote
Express output as probability of image belonging to a particular class

$$softmax\left(y_i\right) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}$$



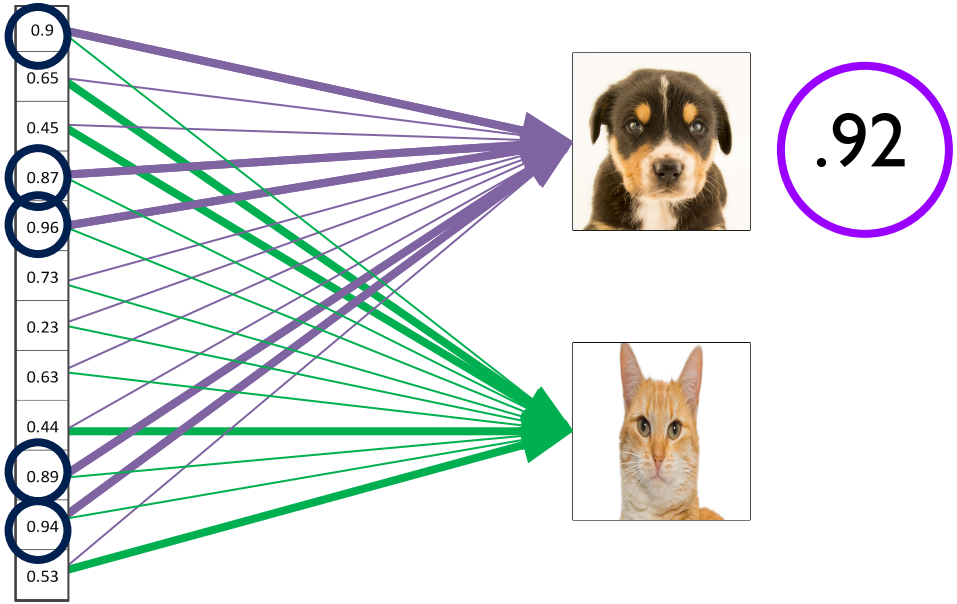
Fully Connected Layer

$$softmax\left(y_i\right) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}$$



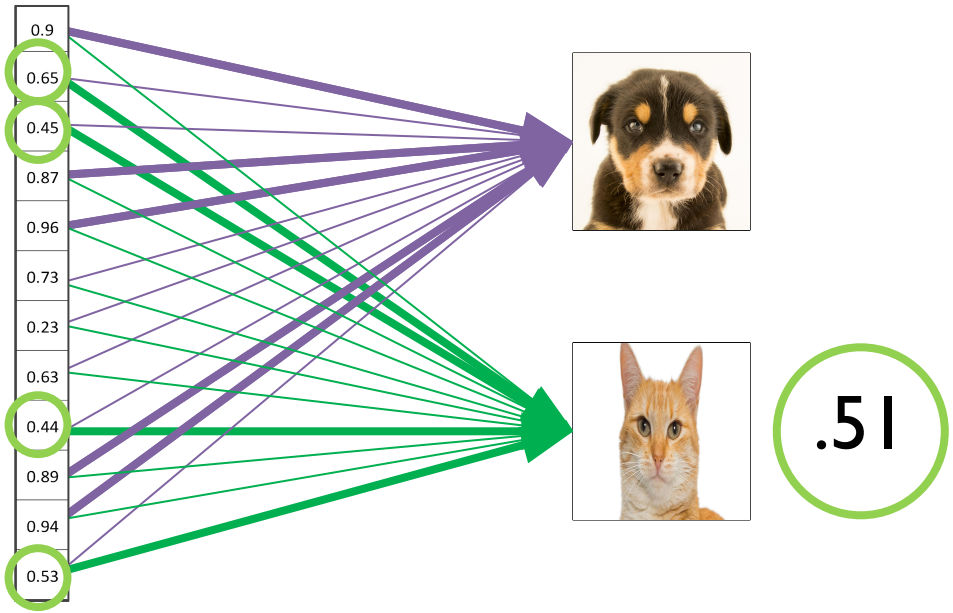
Fully Connected Layer

$$softmax\left(y_i\right) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}$$



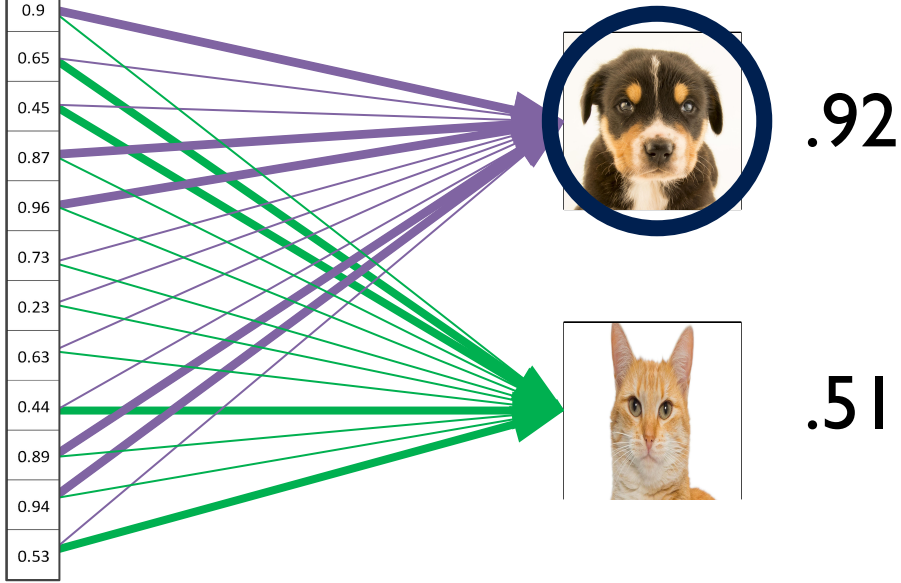
Fully Connected Layer

$$softmax\left(y_i\right) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}$$



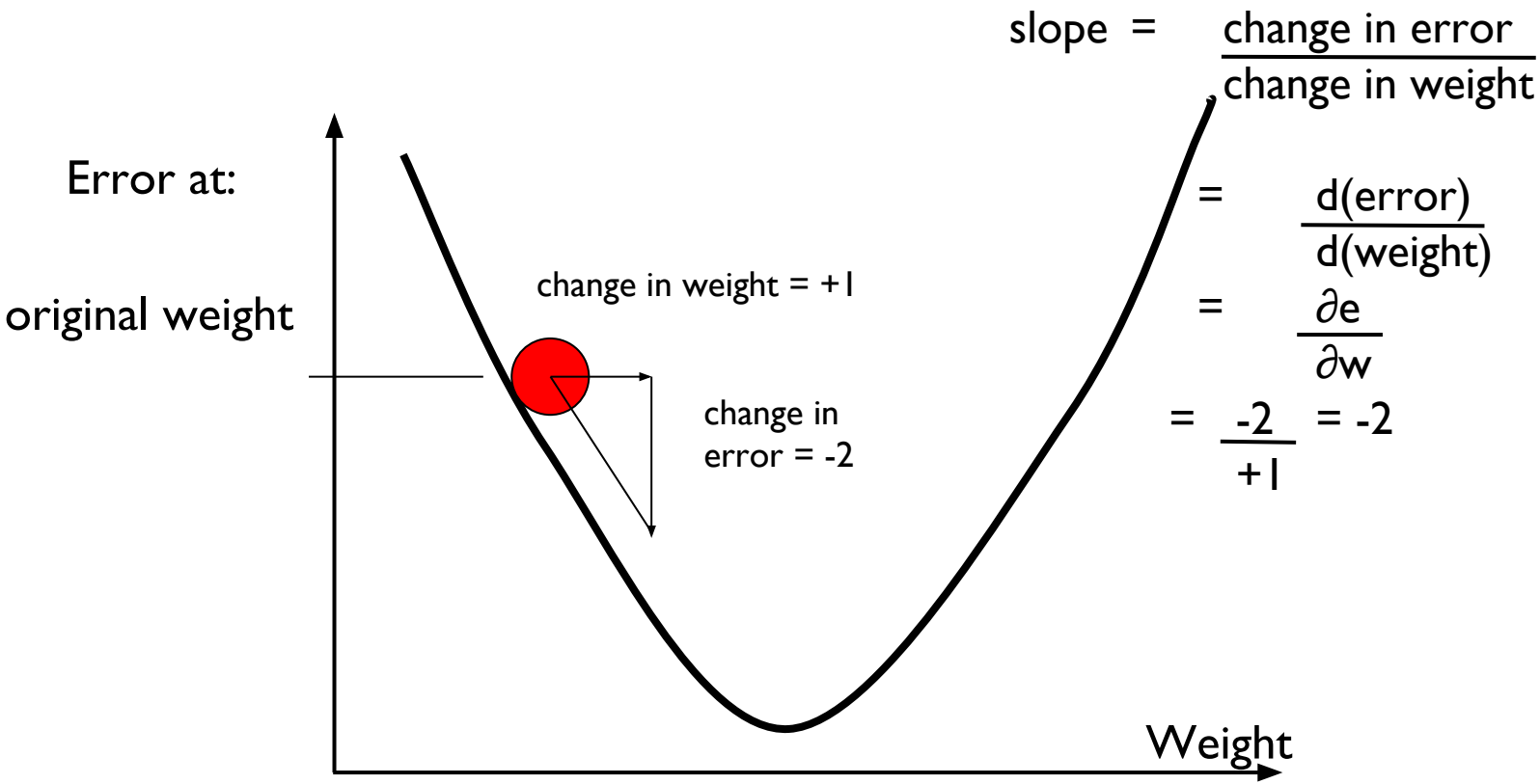
Fully Connected Layer

$$softmax\left(y_i\right) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}$$



Backpropagation

How backpropagation works?



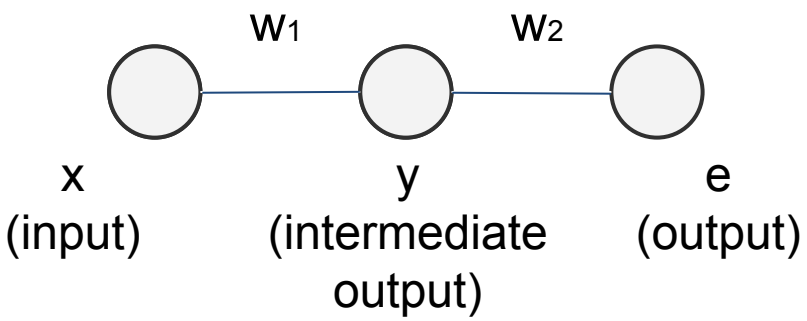
Chaining

$$y = x * w_1$$

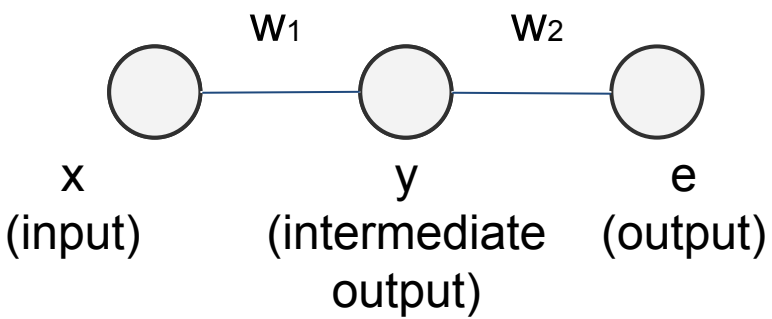
$$\frac{\partial y}{\partial w_1} = x$$

$$e = y * w_2$$

$$\frac{\partial e}{\partial y} = w_2$$



Chaining



$$y = x * w_1$$

$$\frac{\partial y}{\partial w_1} = x$$

$$e = y * w_2$$

$$\frac{\partial e}{\partial y} = w_2$$

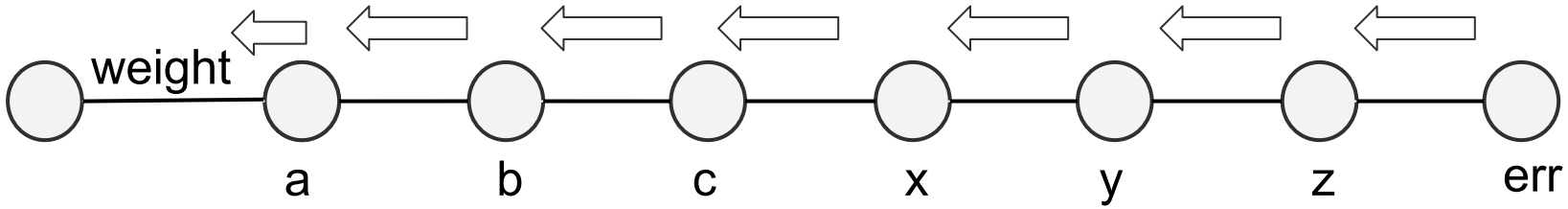
$$e = x * w_1 * w_2$$

$$\frac{\partial e}{\partial w_1} = x * w_2$$

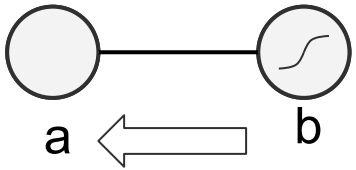
$$\frac{\partial e}{\partial w_1} = \frac{\partial y}{\partial w_1} * \frac{\partial e}{\partial y}$$

Chaining

$$\frac{\partial err}{\partial weight} = \frac{\partial a}{\partial weight} * \frac{\partial b}{\partial a} * \frac{\partial c}{\partial b} * \frac{\partial d}{\partial c} * \ldots * \frac{\partial y}{\partial x} * \frac{\partial z}{\partial y} * \frac{\partial err}{\partial z}$$

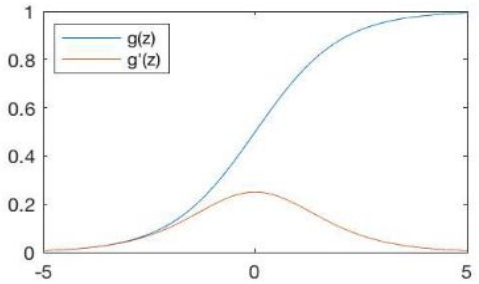


Backpropagation Challenge



$$\frac{\partial err}{\partial a} = \frac{\partial b}{\partial a} * \frac{\partial err}{\partial b}$$

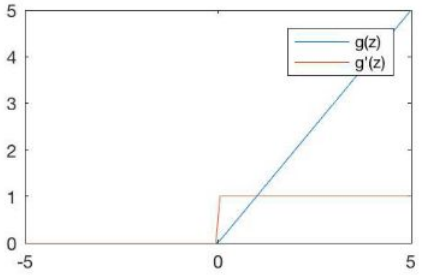
Sigmoid Function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

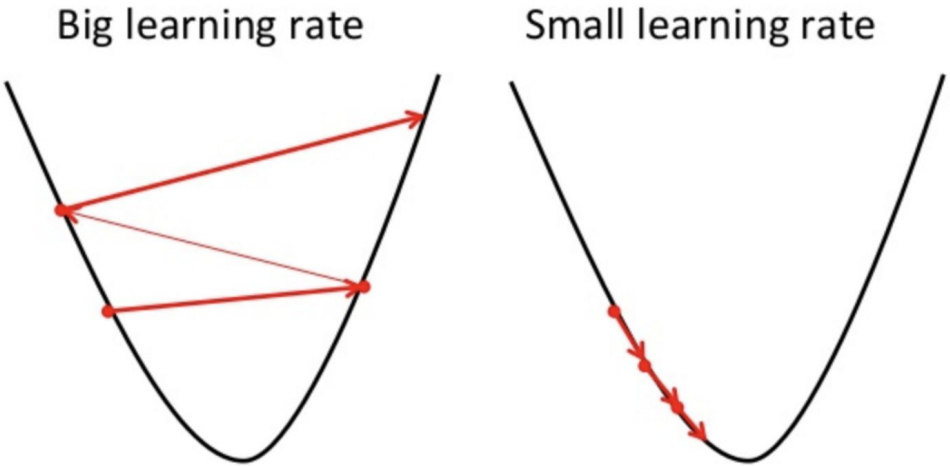
$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

Learning Rate

Learning rate controls how much to adjust the weights with respect to the loss gradient

- High learning rate overshoot the minimum
- Small learning rate can be slow to converge

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} E(w_i)$$

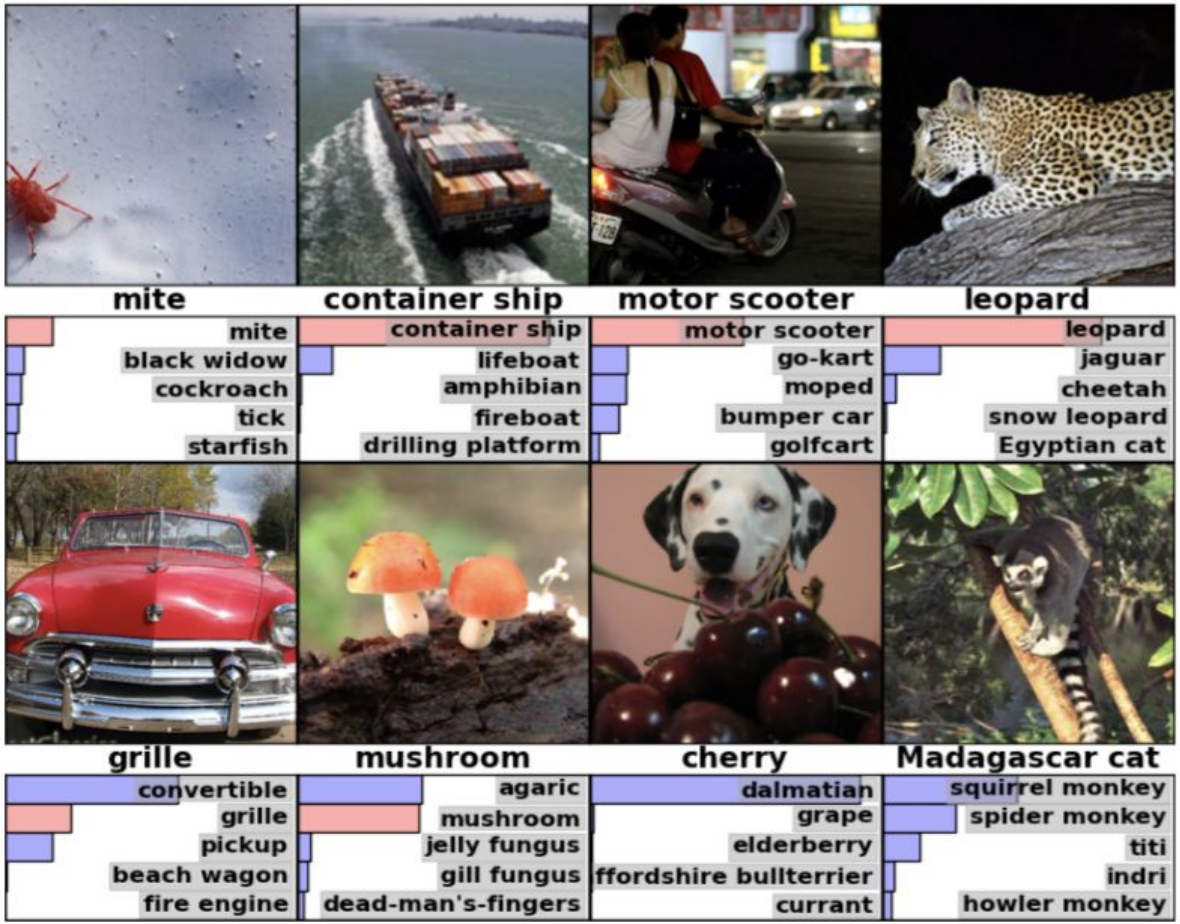


CNN for Classification

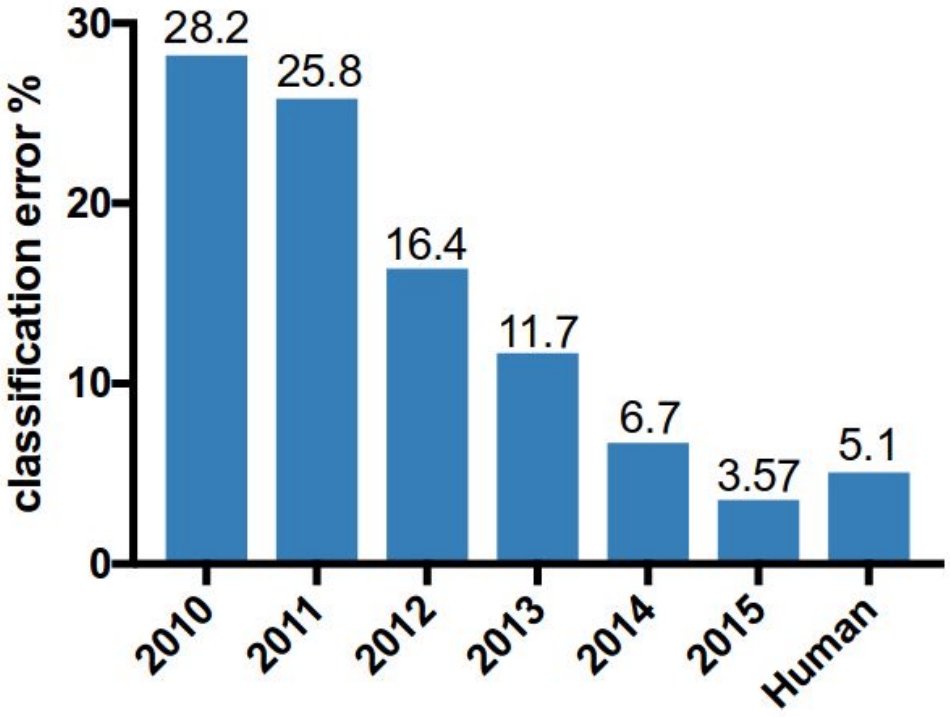
ImageNet

1000 object categories
Images:

- 1.2 M Train
- 100 K Test



ImageNet Challenge: Classification Task



2012: AlexNet. First CNN to win.
- 8 layers, 61 million parameters

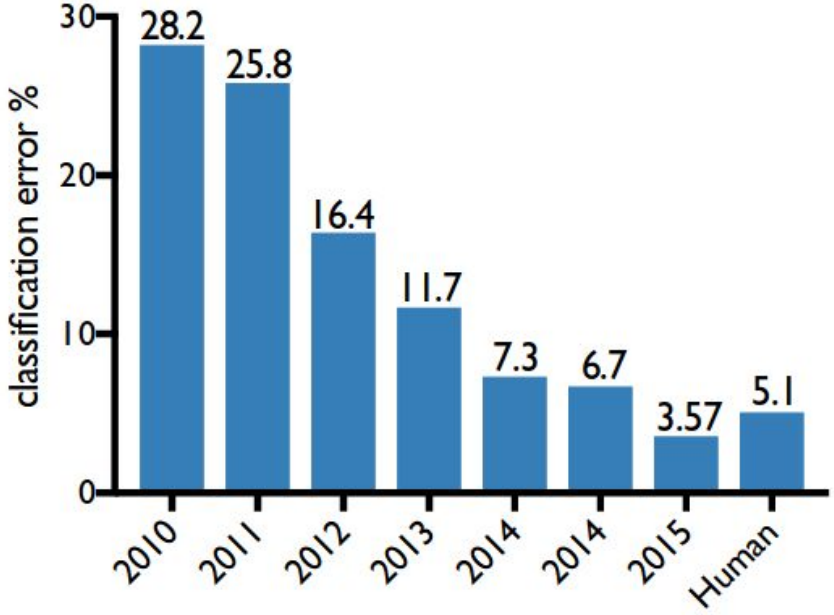
2013: ZFNet
- 8 layers, more filters

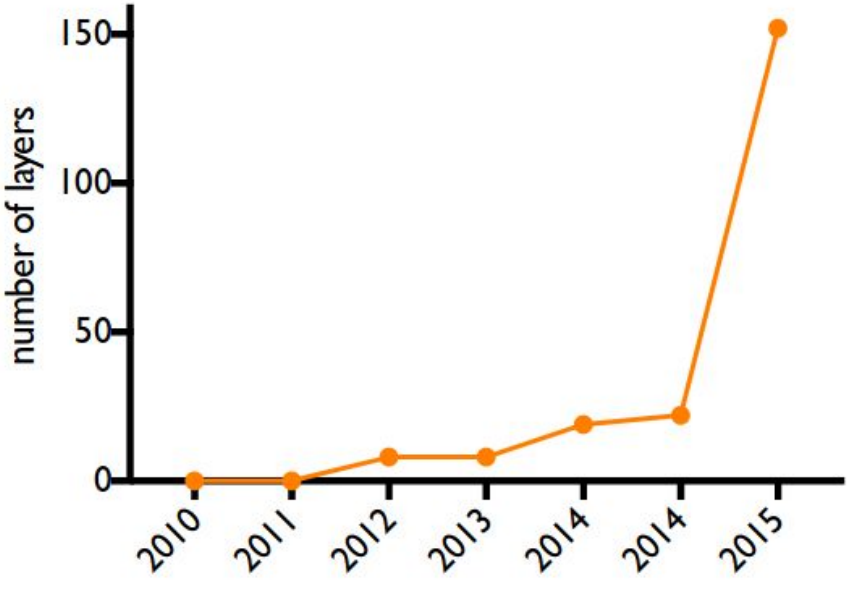
2014: VGG
- 19 layers

2014: GoogLeNet
- "Inception" modules
- 22 layers, 5million parameters

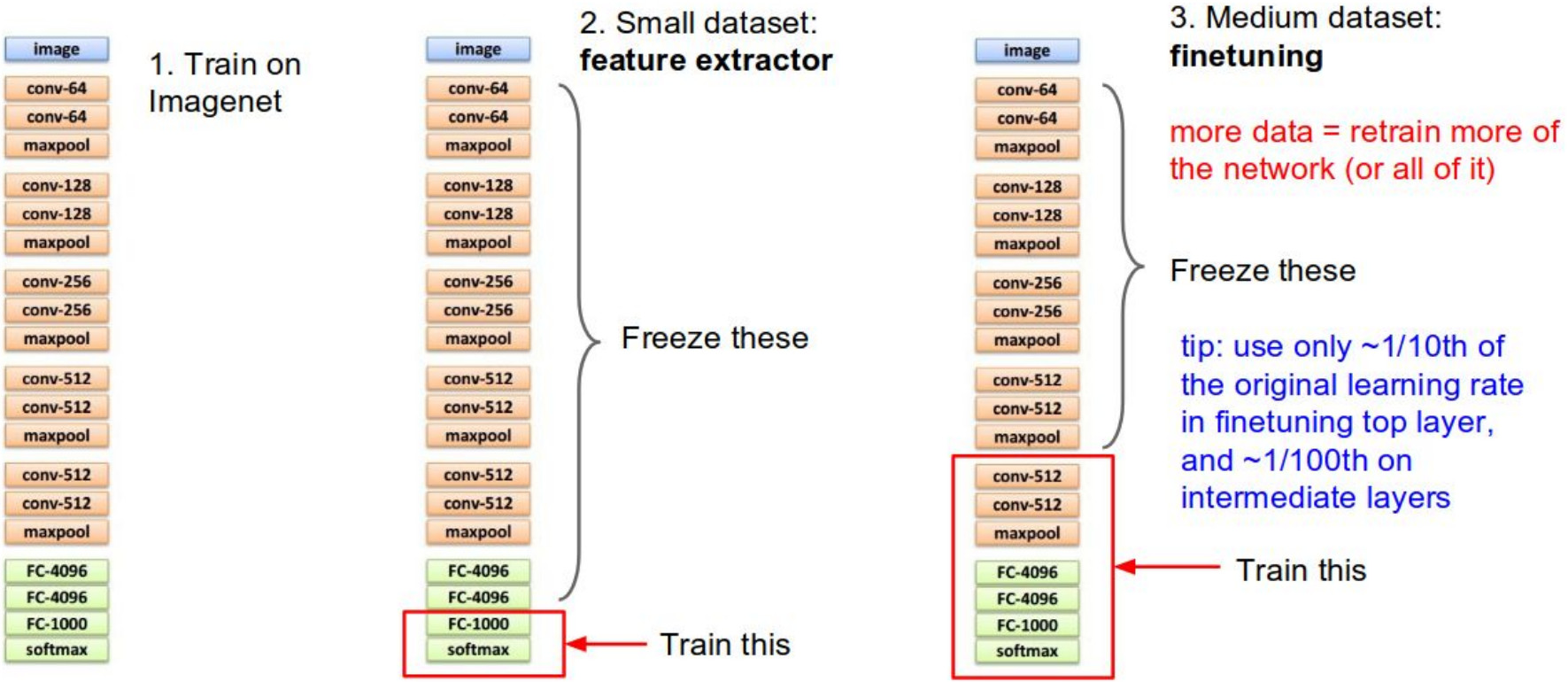
2015: ResNet
- 152 layers

ImageNet Challenge: Classification Task





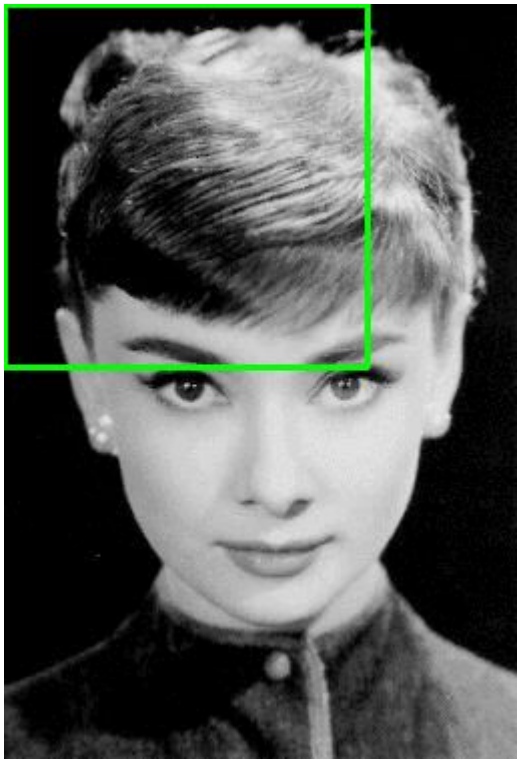
Transfer Learning with CNNs for Classification



Slide from : Fei-Fei Li & Andrej Karpathy & Justin Johnson

Use Cases

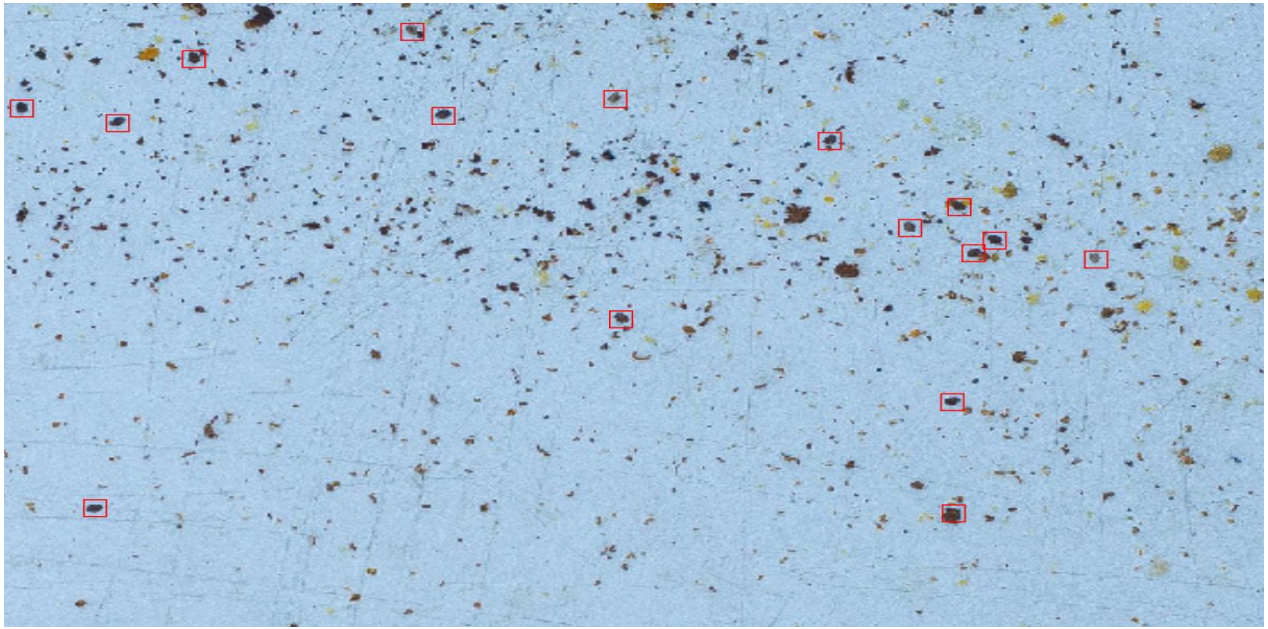
Object Detection Using Sliding Window Detector



Take a set of image patches

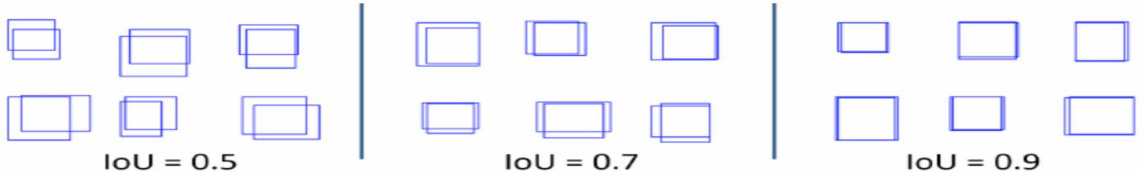
- For each patch, classify what object the patch contains, using an object classifier trained with (hand-crafted features, neural net or CNN)

Ways to Improve Recall



- Do not know the location of the object of interest
  - sliding window at small strides
- Do not know the size/ aspect ratio of the object of interest
  - windows of different sizes

Non-Maximum Suppression (to Improve Precision)

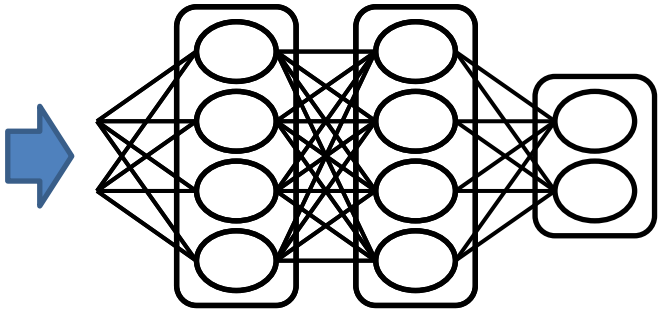


Sample heuristics:

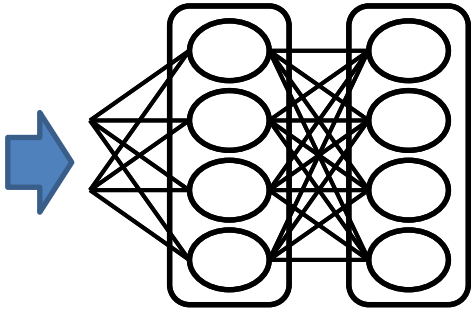
- Remove all boxes, if confidence < T1
- For each object class, pick box with highest confidence, remove all boxes with IoU > T2, stop when all boxes are considered.

$$IoU(A, B) = \frac{A \cap B}{A \cup B}$$

Beyond Classification



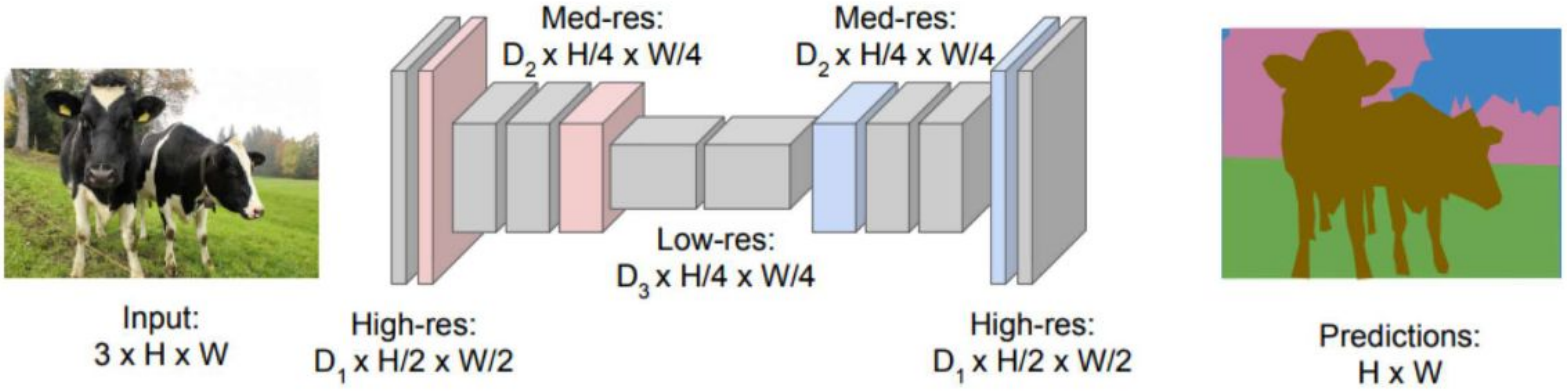
Beagle



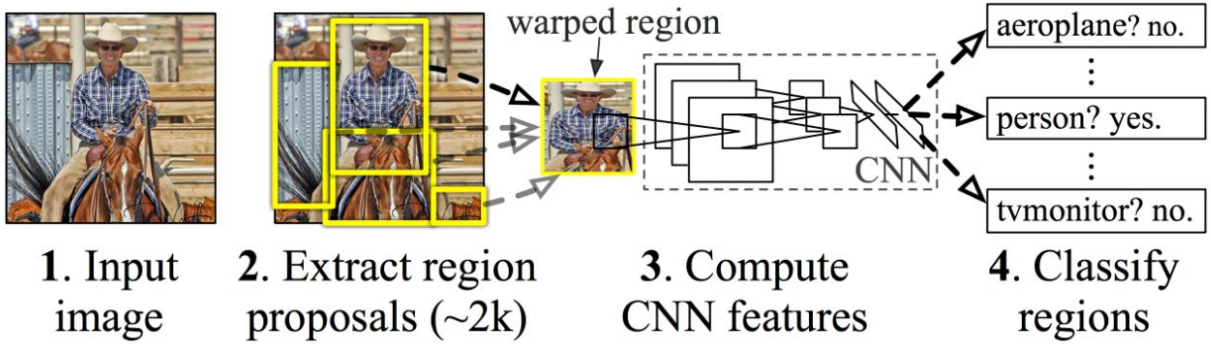
- Image Retrieval
- Detection (RCNN)
- Segmentation (FCN)
- Depth Estimation
- …

Semantic Segmentation: FCNs

FCN: Fully Convolutional Networks
with downsampling and upsampling layers



Object Detection



R-CNN
(Girschick et. al. 2013)

- Region proposals reduce brute force search as in sliding window
- Each proposed region is warped to match input size expected by CNN
- CNN gives more discriminatory features, compared to hand-crafted

Thank You