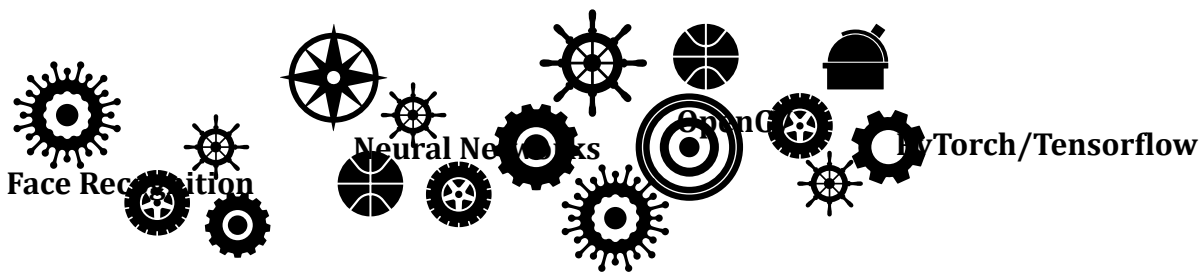
MPSSDEGB
कौशल से रोजगार, समृद्धि अपार

FOUNDATION FOR INNOVATION AND TECHNOLOGY TRANSFER
भारतीय प्रौद्योगिकी संस्थान दिल्ली
Indian Institute of Technology Delhi

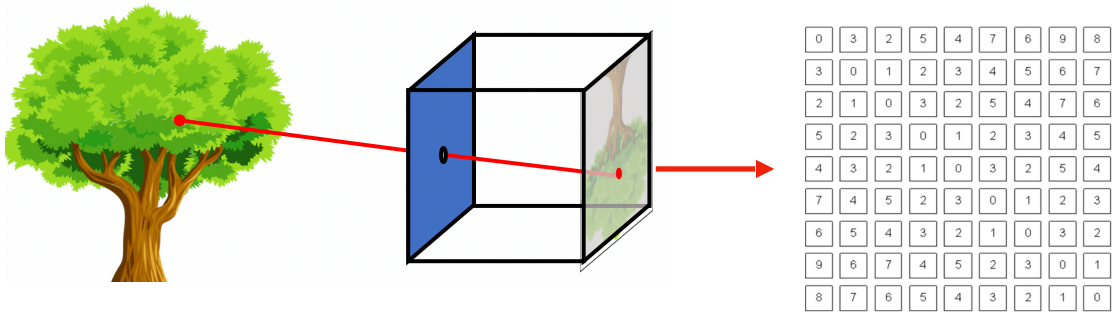
Computer Vision



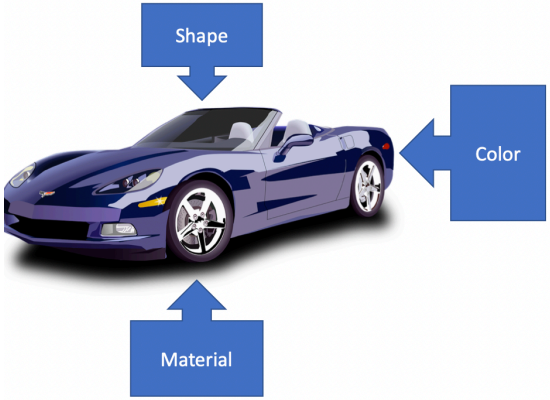
Deep Learning
Segmentation
Object Detection
Image Classification

The content is added from different sources

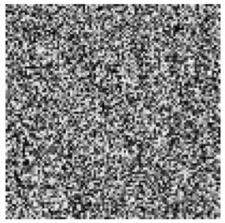
Computer Vision



- Computer vision process 2D matrix of picture element (pixel)
- In contrast, real world objects does not deal with pixels but with physical attributes such as shape, colour and material.



Not all 2D arrays are images



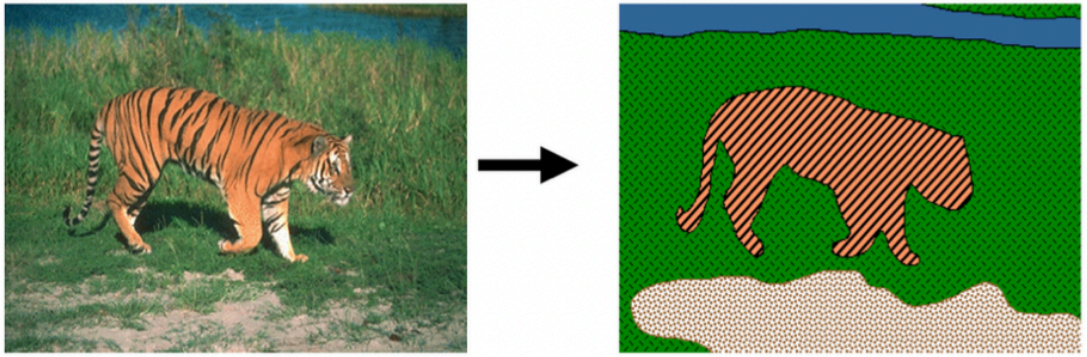
https://www.cs.cornell.edu/courses/cs4670/2018sp/lec02-imageformn.pdf

The content is added from different sources

Computer Vision

- Natural images are *not* arbitrary 2D arrays
- They have properties resulting from physics / math of image formation

- Convert from “pixels” to “objects”: which groups of pixels correspond to objects?



https://www.cs.cornell.edu/courses/cs4670/2018sp/lec02-imageformn.pdf

Computer Vision

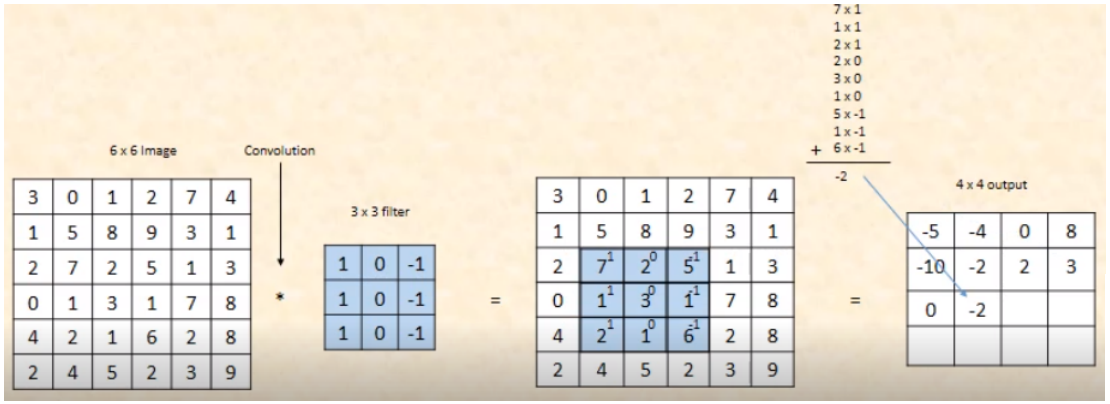
https://www.researchgate.net/figure/a-Input-binary-image-b-Measured-results-of-horizontal-edge-detection-using-Prewitt_fig10_313372308

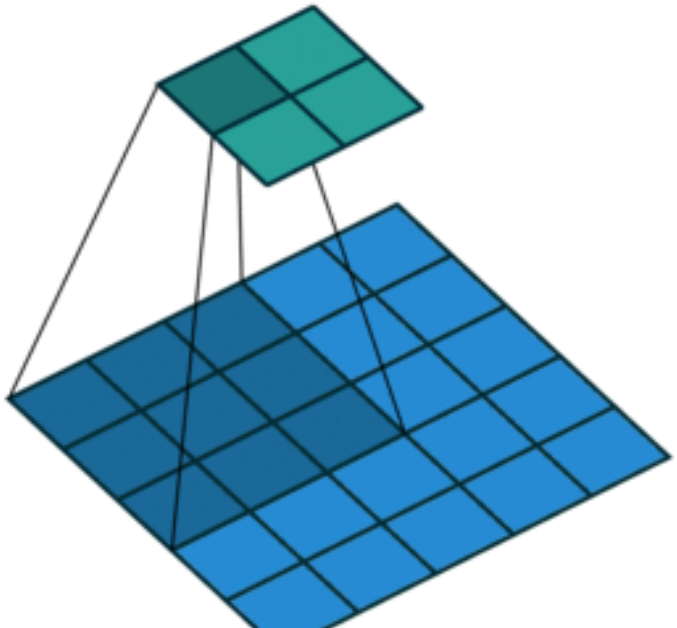
Mathematics

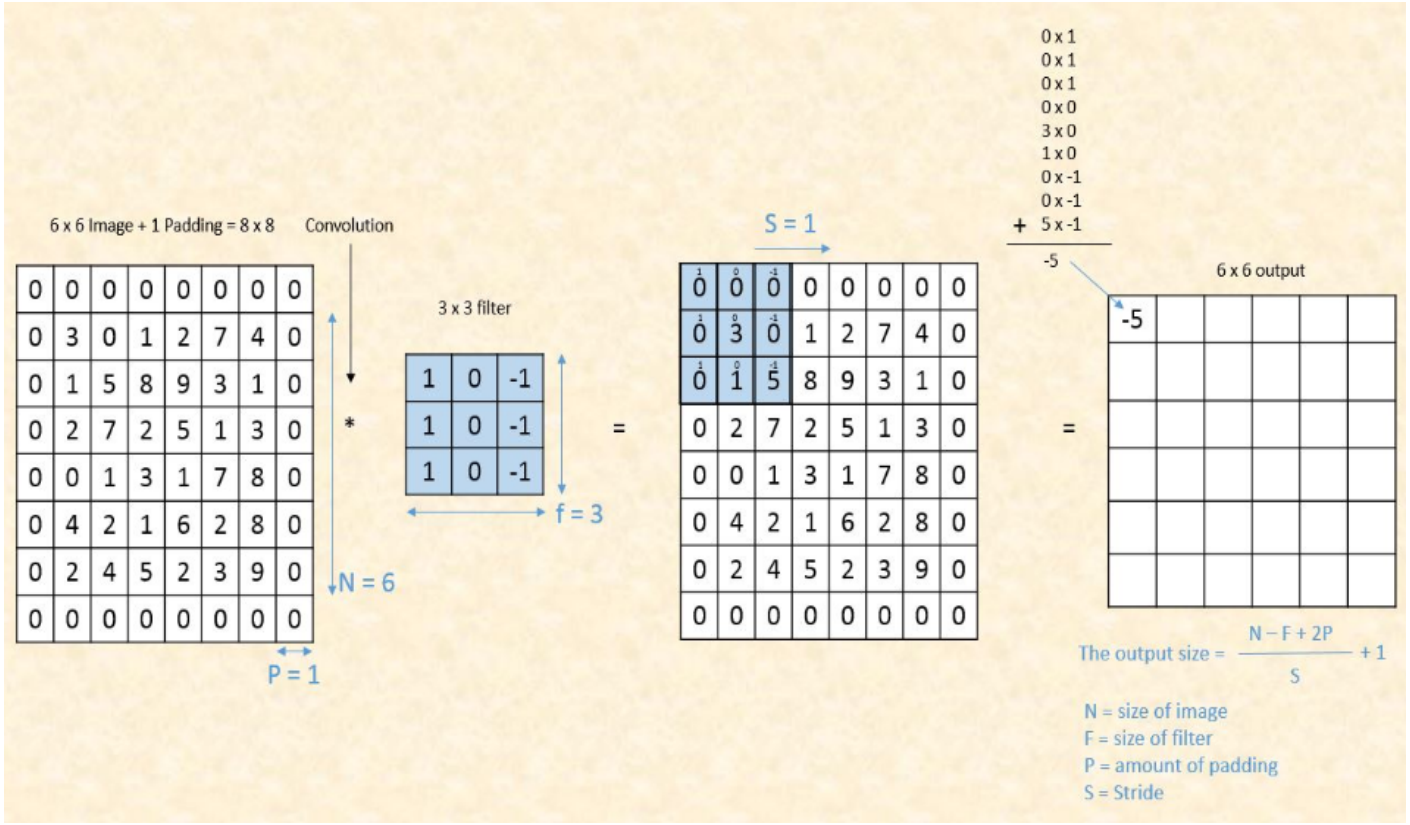
Convolution operation -

- convolving a 6 x 6 grayscale image with a 3 x 3 matrix called filter or kernel to produce a 4 x 4 matrix.

- dot product between the filter and the first 9 elements of the image matrix and fill the output matrix.






6 x 6 Image + 1 Padding = 8 x 8
Convolution
3 x 3 filter
S = 1
N = 6
P = 1
f = 3
0 x 1
0 x 1
0 x 1
0 x 0
3 x 0
1 x 0
0 x -1
0 x -1
+ 5 x -1
-5
6 x 6 output
The output size = (N − F + 2P) / S + 1
N = size of image
F = size of filter
P = amount of padding
S = Stride

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).

Convolution

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

Filter 1

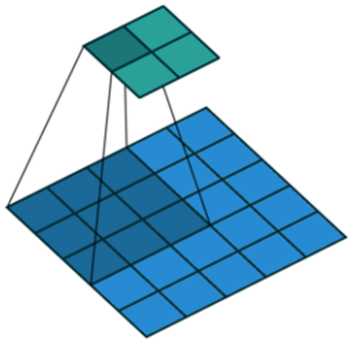
stride=1

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

Dot product → 3, -1

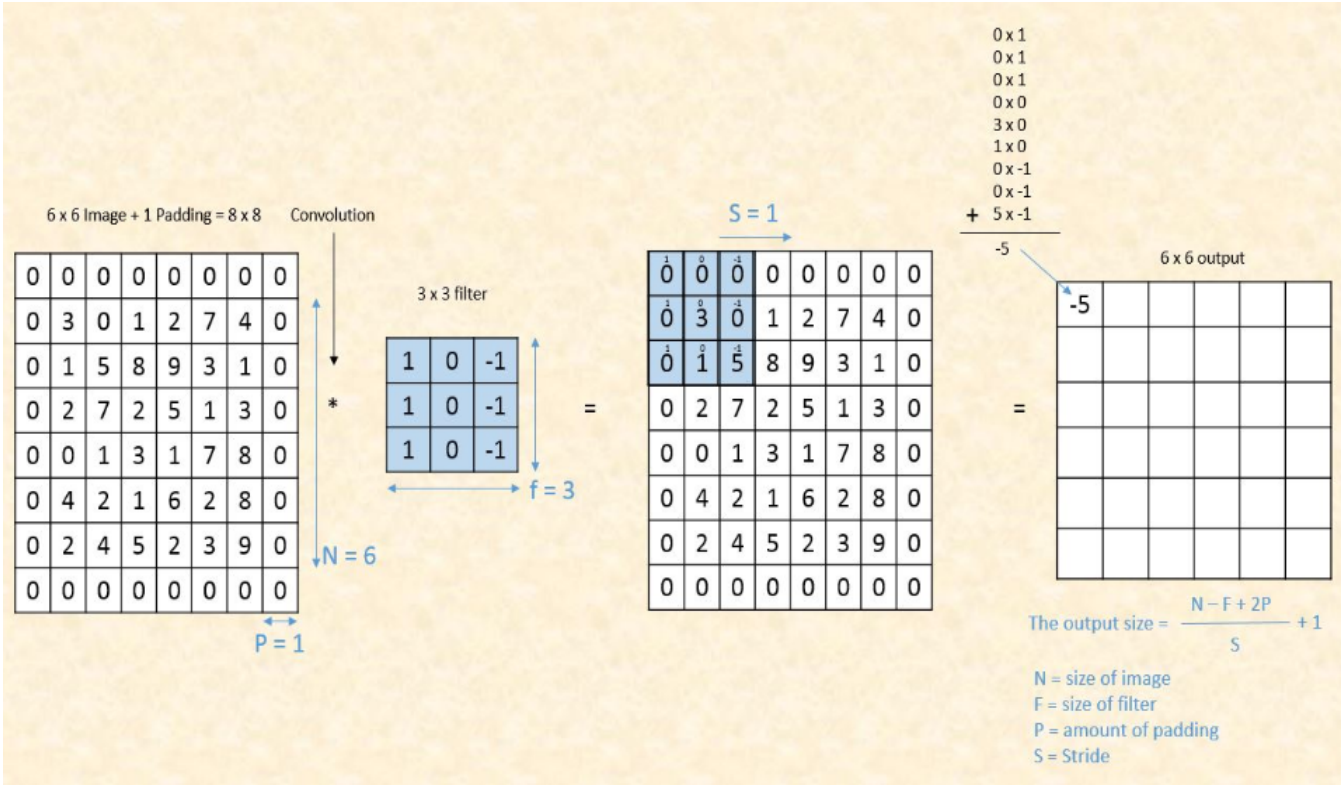
1*1 + 0*(-1) + 0*(-1) +
0*(-1) + 1*1 + 0*(-1) +
0*(-1) + 0*(-1) + 1*1



Challenges with convolution operation:

- Output may shrink.
- Data loss from image corners.

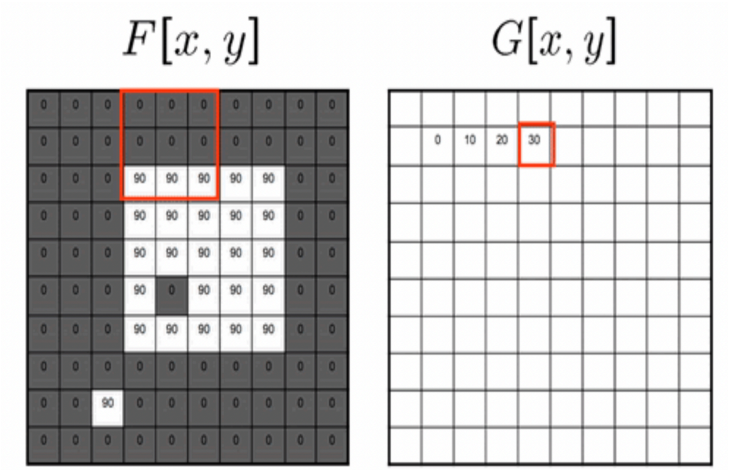
Solution - Padding (size of the output image is equal to input image size.)


6 x 6 Image + 1 Padding = 8 x 8
Convolution
3 x 3 filter
S = 1
N = 6
P = 1
f = 3
0 x 1
0 x 1
0 x 1
0 x 0
3 x 0
1 x 0
0 x -1
0 x -1
+ 5 x -1
-5
6 x 6 output
The output size = (N − F + 2P) / S + 1
N = size of image
F = size of filter
P = amount of padding
S = Stride

Computer Vision

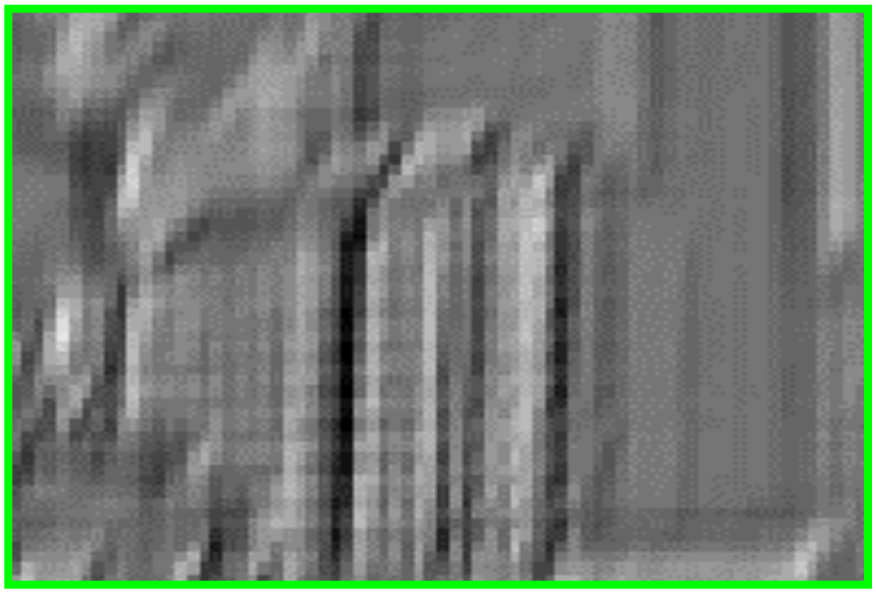
1. Filtering and Convolution

Filtering operations such as blurring, sharpening and noise reduction are applied to images using convolution. Convolution involves sliding a filter or kernel over an image and performing mathematical operations on each pixel. This process enables various improvements such as anti-aliasing, edge detection and texture removal.



https://medium.com/@naveenpandey2706/lesson-2-understanding-image-processing-basics-and-techniques-in-computer-vision-c1d7c9763966

https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html



- Weighted moving sum generating features

slide credit: S. Lazebnik

Biological Vision

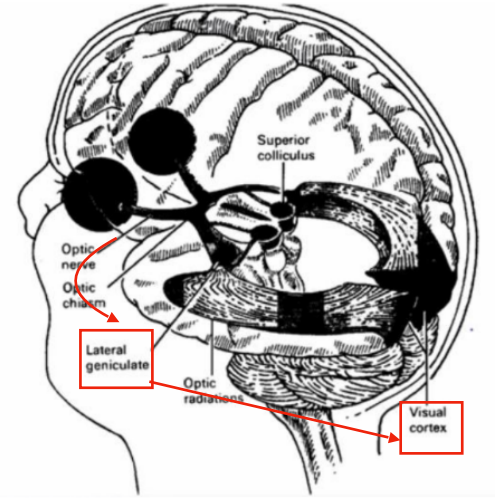
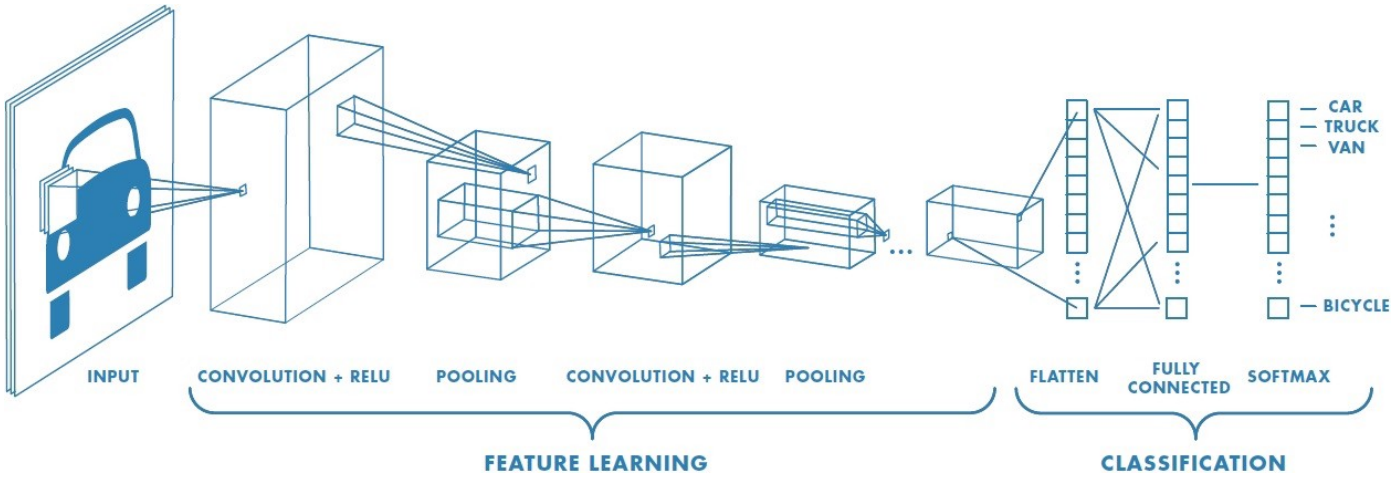


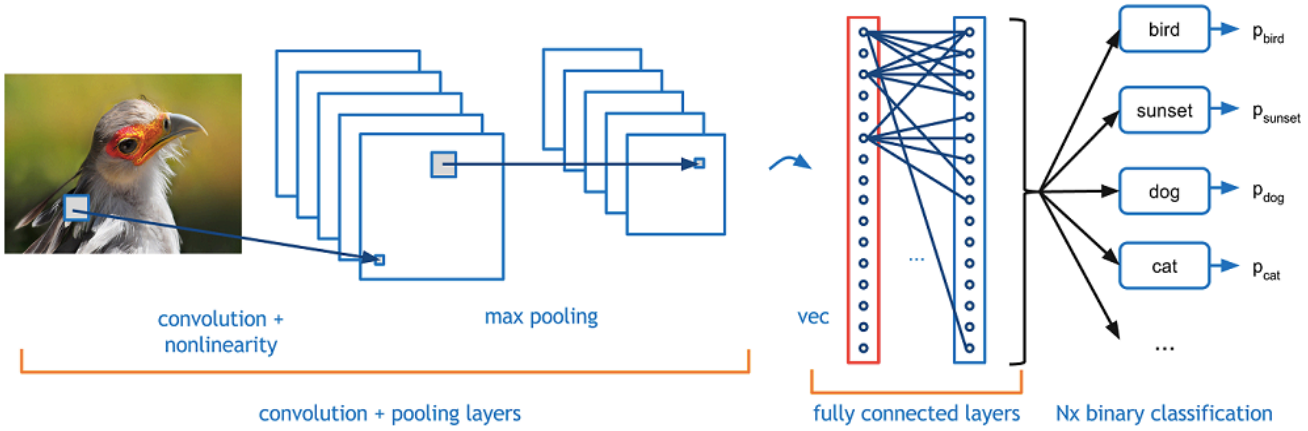
Image Credit:The Three R's of Computer Vision, Jitendra Malik
UC Berkeley

simple visual forms, edges, corners

Pathway of Information Processing

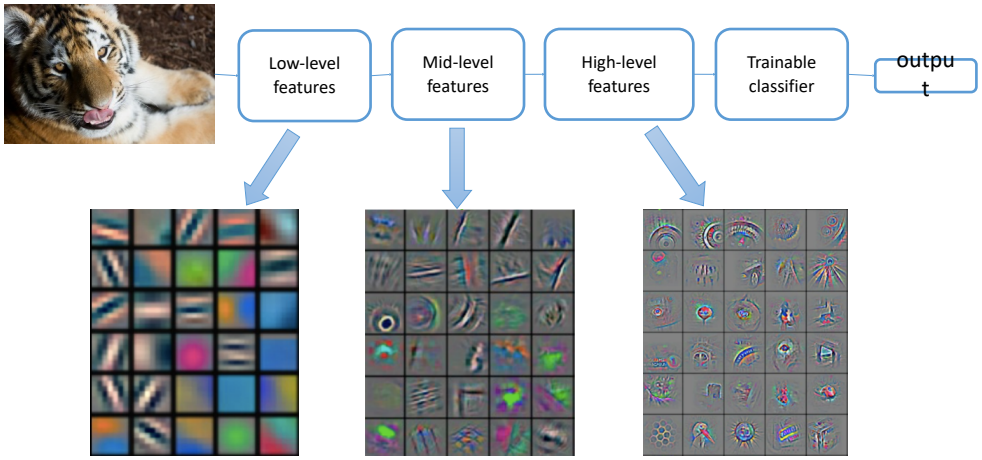
high-level object descriptions, faces, objects


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION


bird
sunset
dog
cat
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
convolution + nonlinearity
max pooling
vec
convolution + pooling layers
fully connected layers
Nx binary classification

DEEP LEARNING

- Deep learning (a.k.a. representation learning) seeks to learn rich hierarchical representations (i.e. features) automatically through multiple stage of feature learning process.



Convolutional Neural Networks

- Each layer in a CNN applies a different set of filters, typically hundreds or thousands of them, and combines the results, feeding the output into the next layer in the network.

- During training, a CNN automatically learns the values for these filters.

- In the context of image classification, our CNN may learn to:

✓ Detect edges from raw pixel data in the first layer.

✓ Use these edges to detect shapes (i.e., “blobs”) in the second layer.

✓ Use these shapes to detect higher-level features such as facial structures, parts of a car, etc. in the highest layers of the network.

- The last layer in a CNN uses these higher-level features to make predictions regarding the contents of the image.

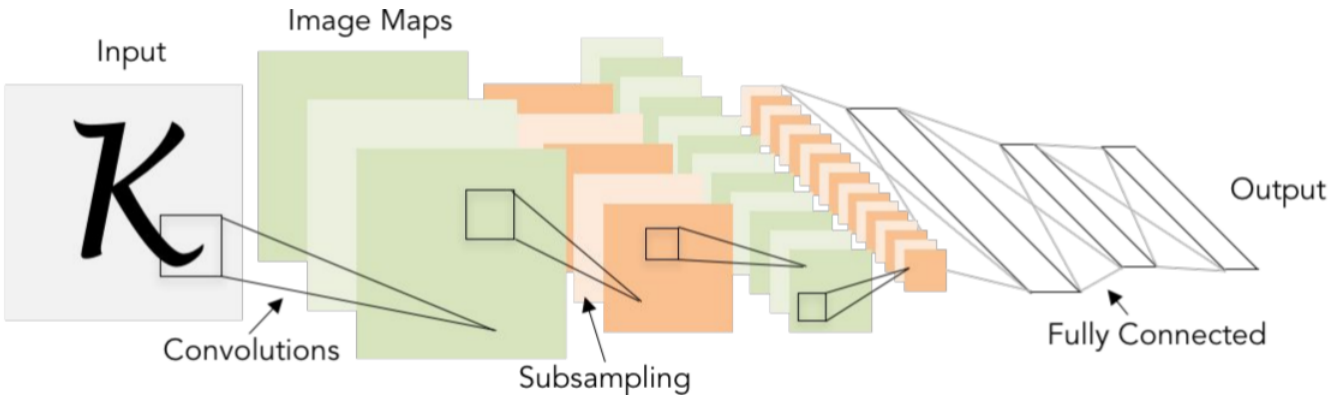
Convolutional Neural Networks

CNN consist of three layers -

1. Convolution Layer,
2. Pooling Layer, and
3. Fully Connected Layer.

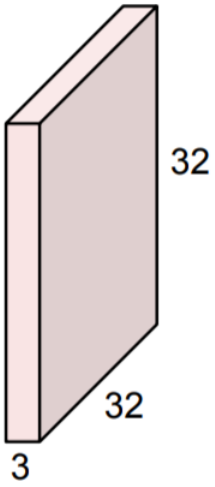
ConvNet is architecture is formed stacking these layers.

- The output can be a softmax layer indicating whether there is a cat or something else. You can also have a sigmoid layer to give you a probability of the image being a cat.



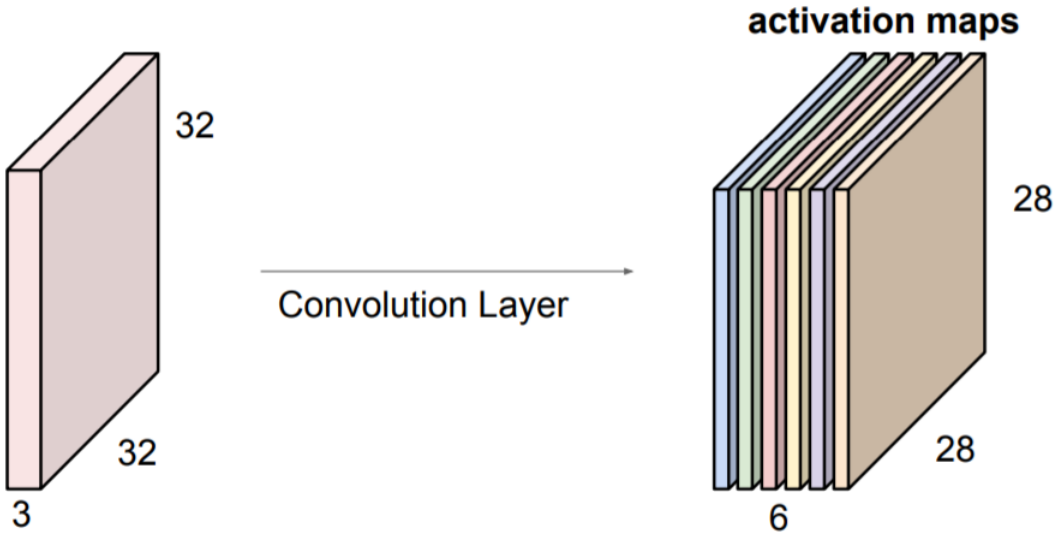
Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

32x32x3 image

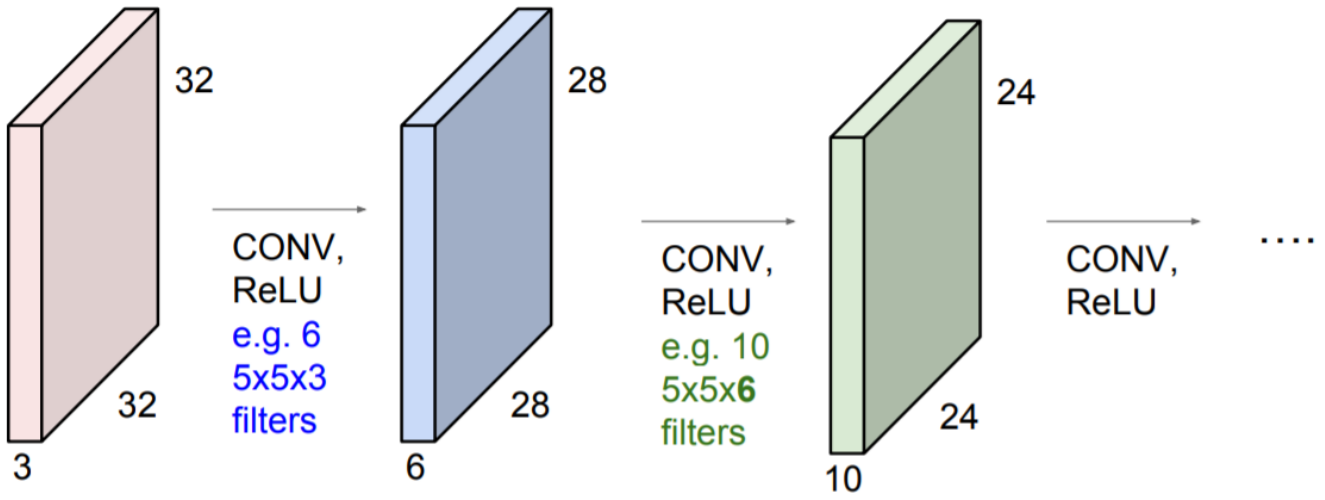


5x5x3 filter

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



Ex: we have 6 - > 5x5 filters, we'll get 6 separate activation maps. "new image" of size 28x28x6!


32
32
3
CONV,
ReLU
e.g. 6
5x5x3
filters
28
28
6
CONV,
ReLU
e.g. 10
5x5x6
filters
24
24
10
CONV,
ReLU
....

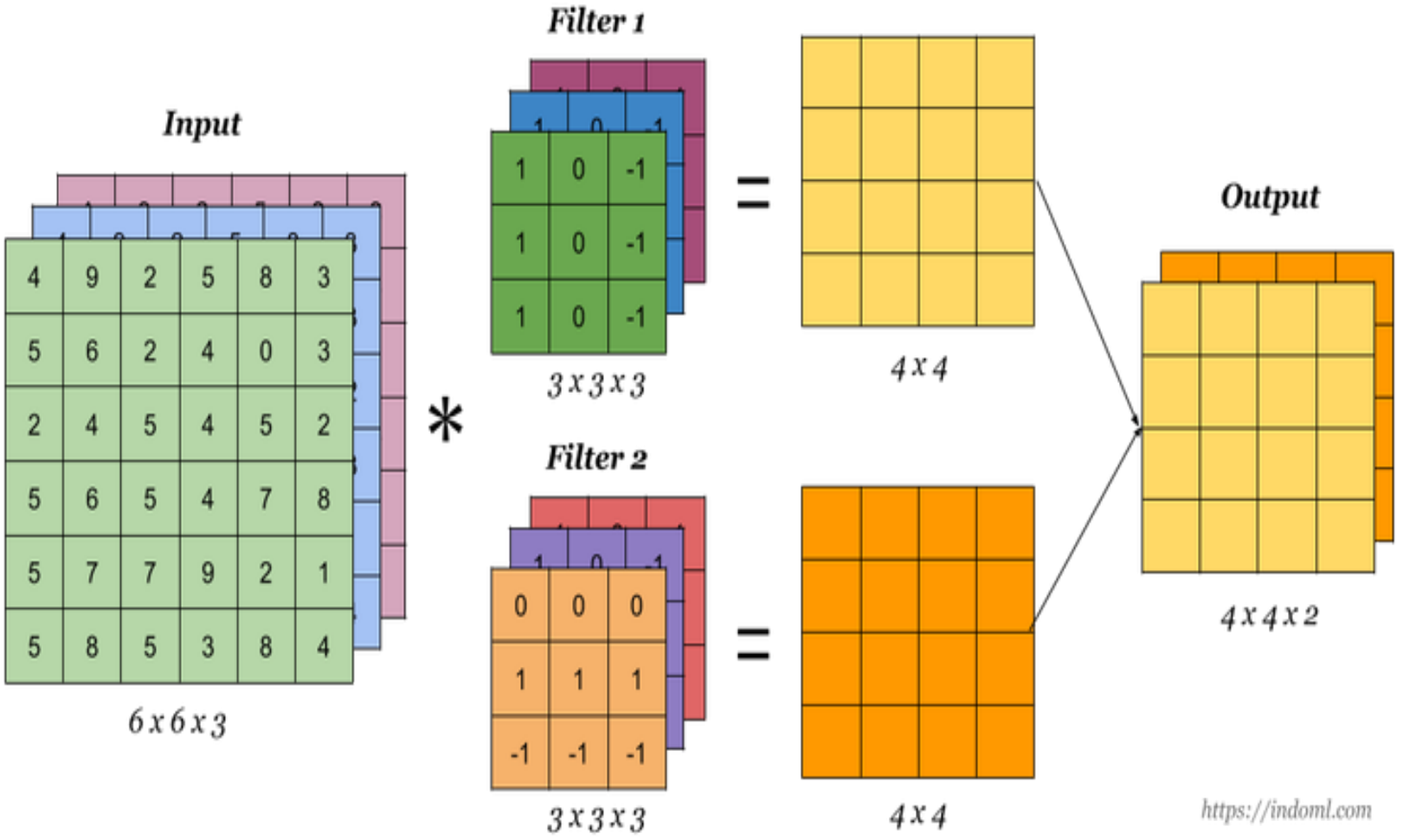
Convolution over volume

depth of the filter = depth of image (necessary condition)

- For every filter one feature map is created
- Depth of the output image represents the number of filters

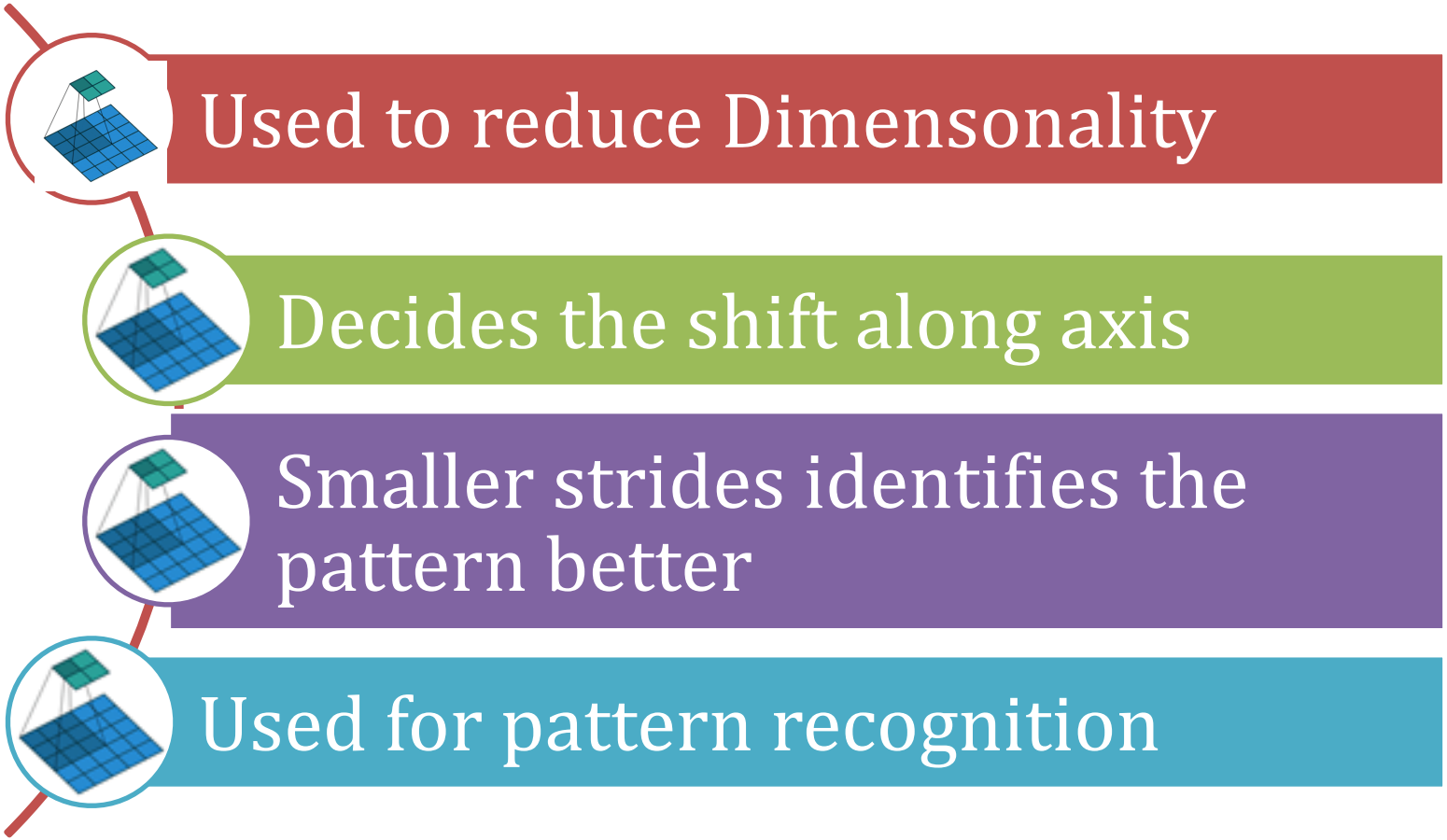
With every increasing layer size of image decreases and number of filter increases

Convolution over volume

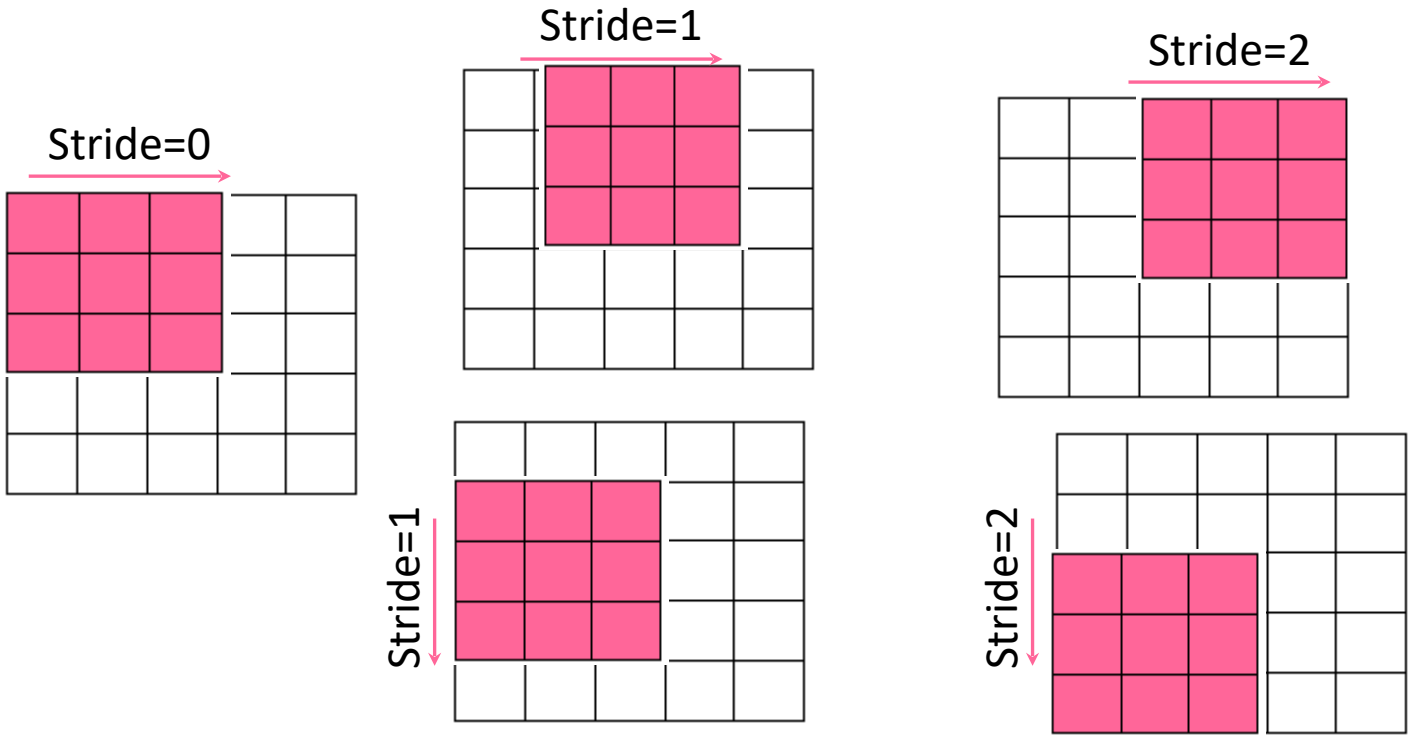


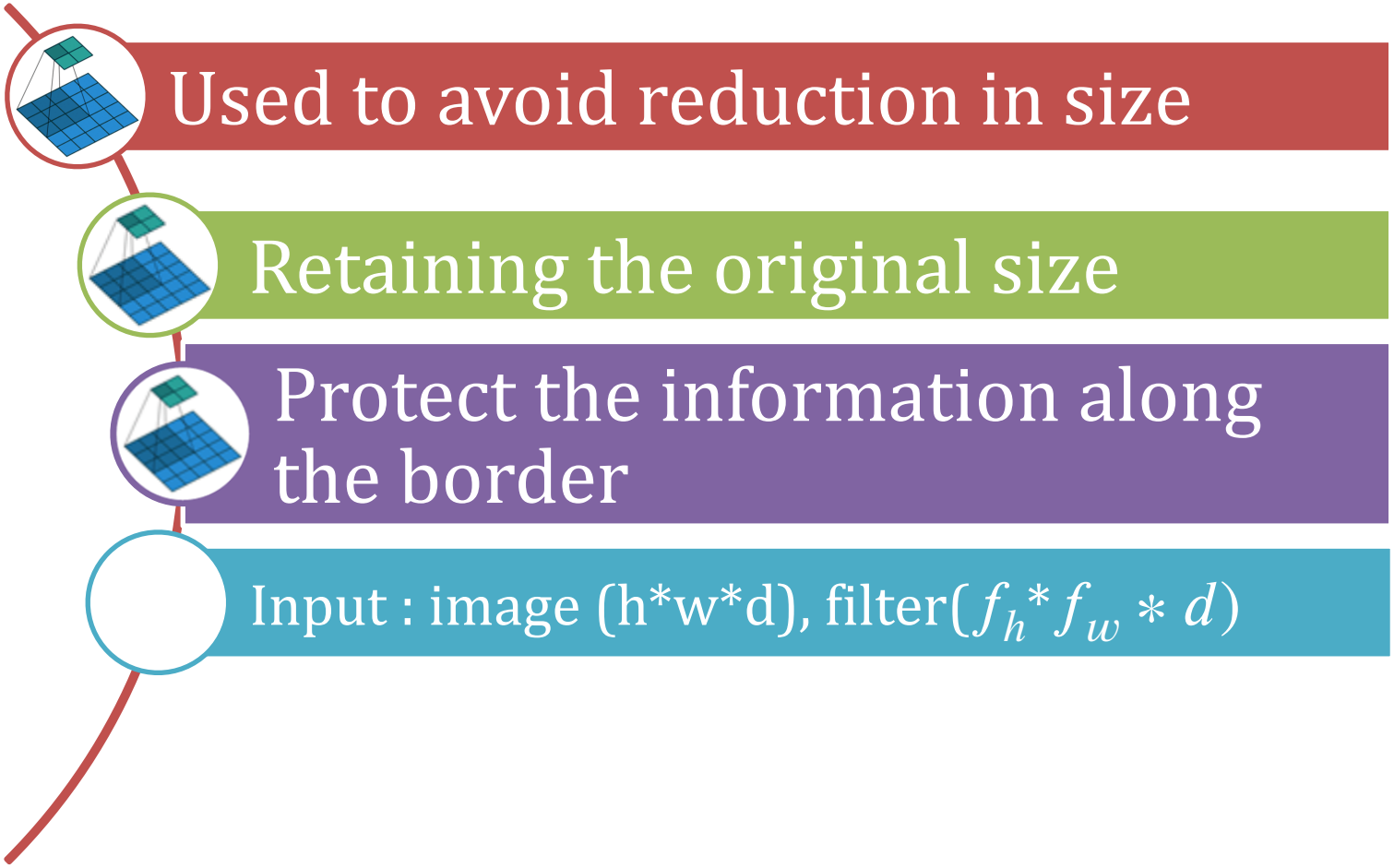
https://images.app.goo.gl/kGgsx37M3rcgtgsS9

Strides


Used to reduce Dimensonality
Decides the shift along axis
Smaller strides identifies the pattern better
Used for pattern recognition

Stride- Visualisation

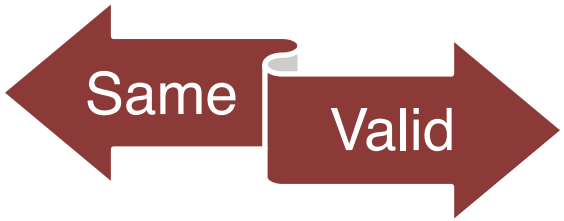



Used to avoid reduction in size
Retaining the original size
Protect the information along the border
Input : image (h*w*d), filter($f_h * f_w * d$)

Types of padding

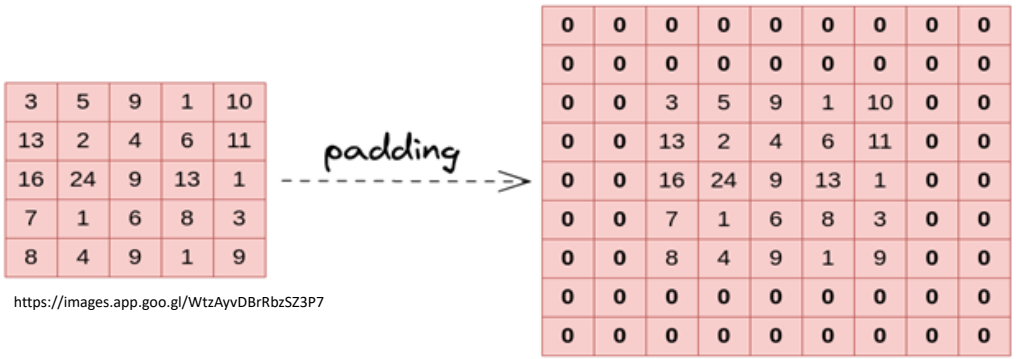
Same

after padding the size of image is uneffected.

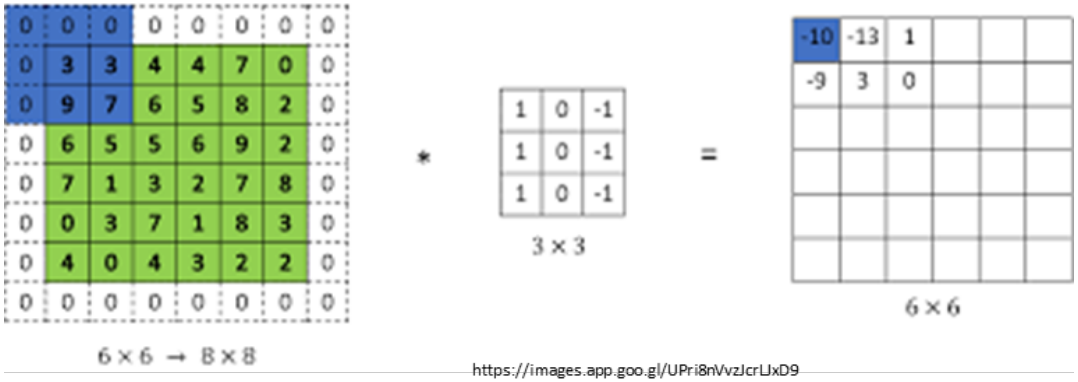


Valid

no padding done.



https://images.app.goo.gl/WtzAyvDBrRbzSZ3P7



https://images.app.goo.gl/UPri8nVvzJcrLJxD9

Padding Numerical

- If the image matrix is $6 * 6$
- If the size of the filter is $3 * 3 * 2$,
- If stride is 1 and padding is 1
- The output is calculated as

$$\left(\frac{h + (2 * p) - fh}{s} + 1\right) * \left(\frac{w + (2 * p) - fw}{s} + 1\right) * d$$

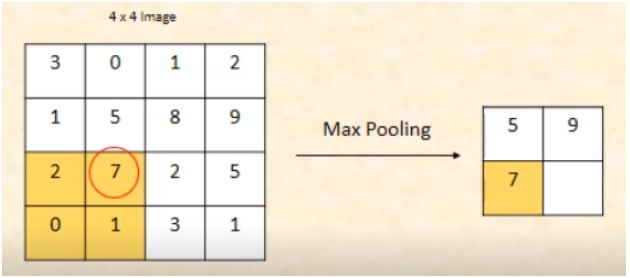
- So, output image is

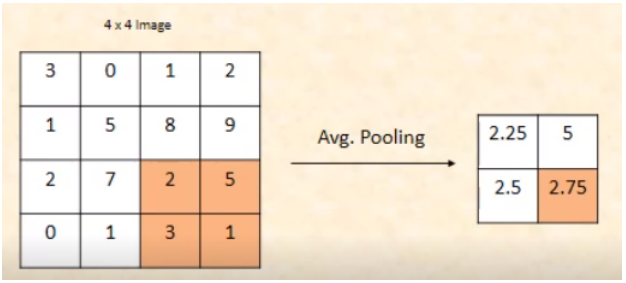
$$\left(\frac{6 + (2 * 1) - 3}{1} + 1\right) * \left(\frac{6 + (2 * 1) - 3}{1} + 1\right) * 2$$

=6*6

Pooling Layer:

- Reduce number of parameters.
- In pooling layer, we have two hyperparameters filter size and stride which are fixed only once.

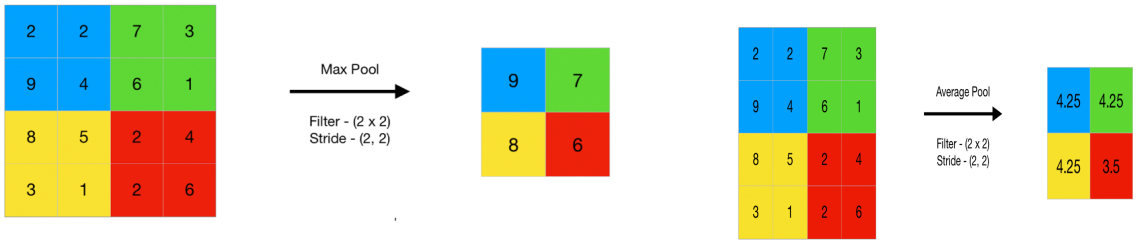




- Pooling layers are used to reduce the dimensions of the feature maps. Reduces the number of parameters to learn and the amount of computation performed in the network.

- The pooling layer summarises the features present in a region of the feature map generated by a convolution layer. So, further operations are performed on summarised features instead of precisely positioned features generated by the convolution layer. This makes the model more robust to variations in the position of the features in the input image.

- Average pooling computes the average of the elements present in the region of feature map covered by the filter. Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.



Why Pooling

- Subsampling pixels will not change the object

bird

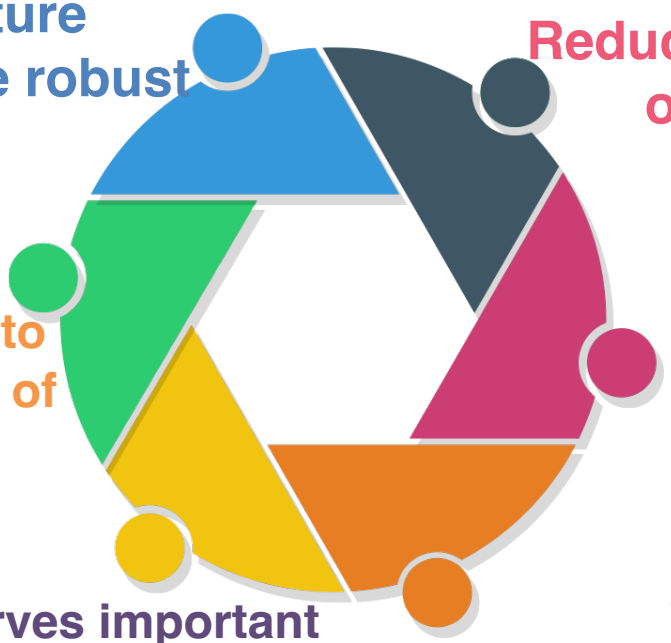
Subsampling

bird

We can subsample the pixels to make image smaller

fewer parameters to characterize the image

Importance of pooling



Makes feature detection more robust

Reduces the size of input

Achives invariance to change in position, illumination, robustness to clutter, and compactness of representation

Increases the speed of computation

Preserves important features discarding unwanted

Channel independent and reduces complexity

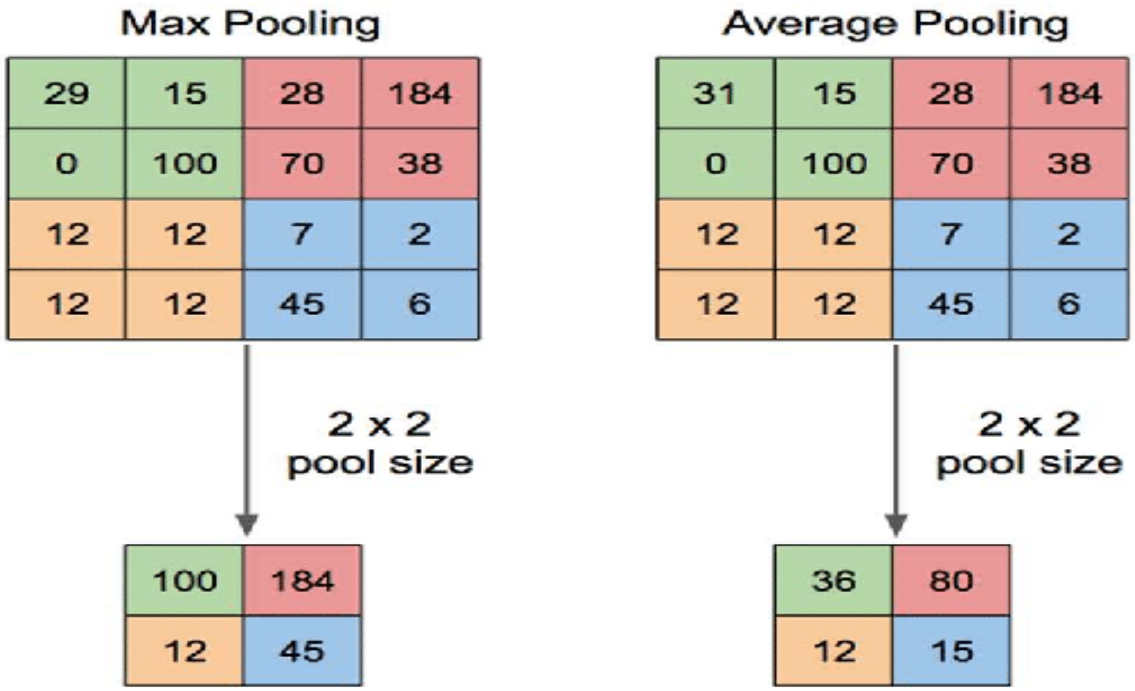
Types of Pooling

Max

Take the max value in each block

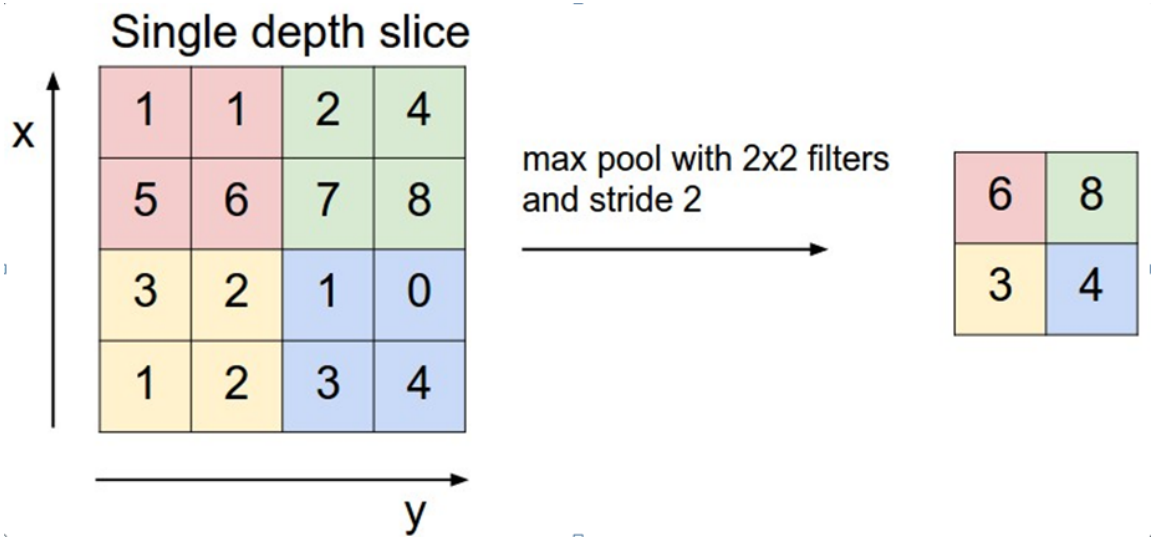
Average

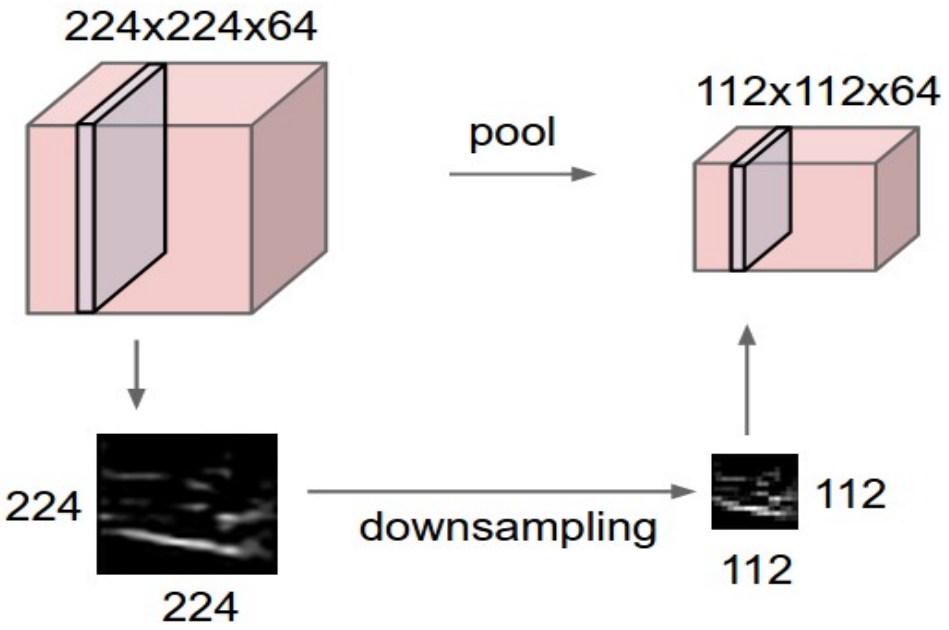
average all values in each block



https://images.app.goo.gl/MWnbBi743TrW6EtYA

Visualisation of Pooling

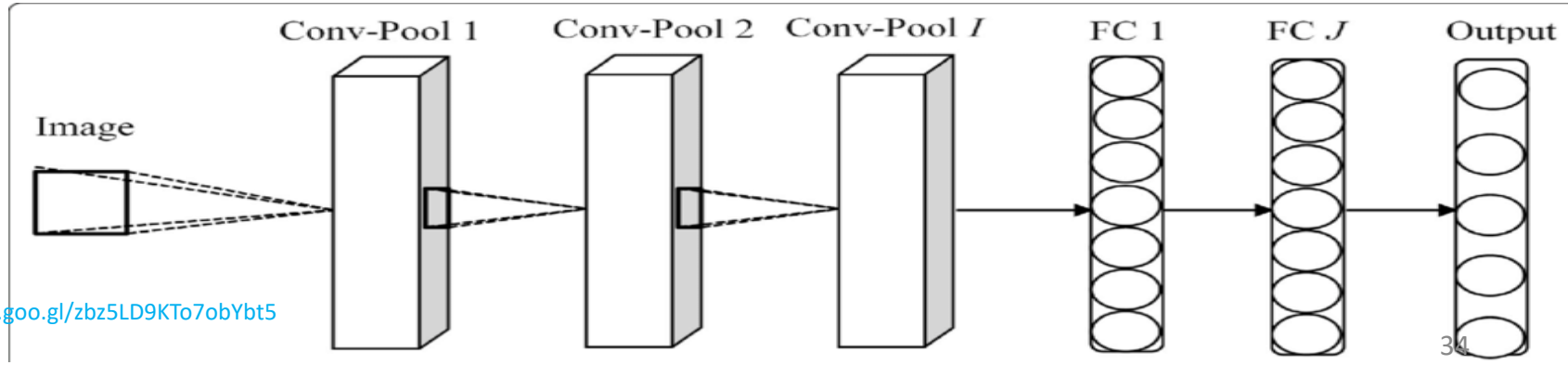




Flattening or FC layer

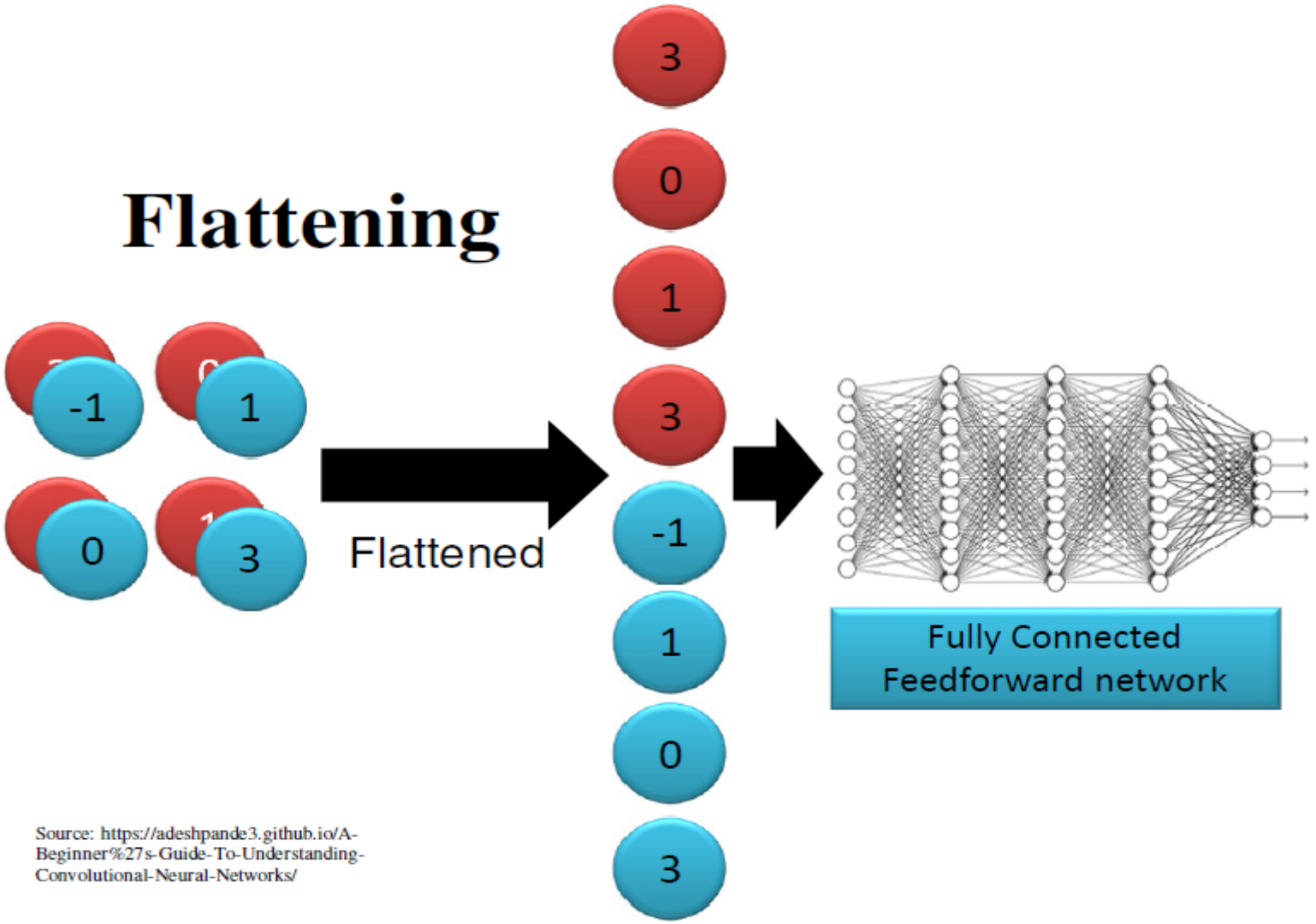
Hidden layers are called classification layer

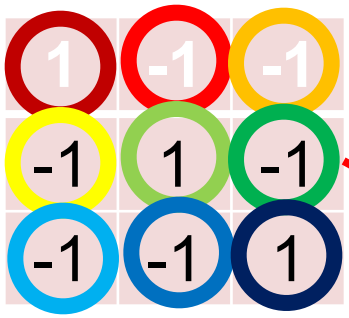
Number of neurons in the output layer is equal to the number of classes.



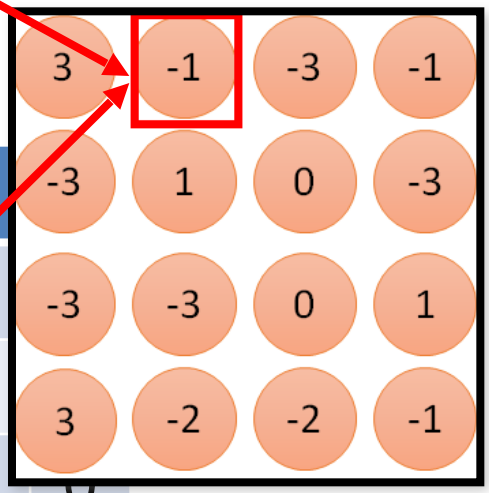
https://images.app.goo.gl/zbz5LD9KTo7obYbt5

Flattening or FC layer





Filter 1

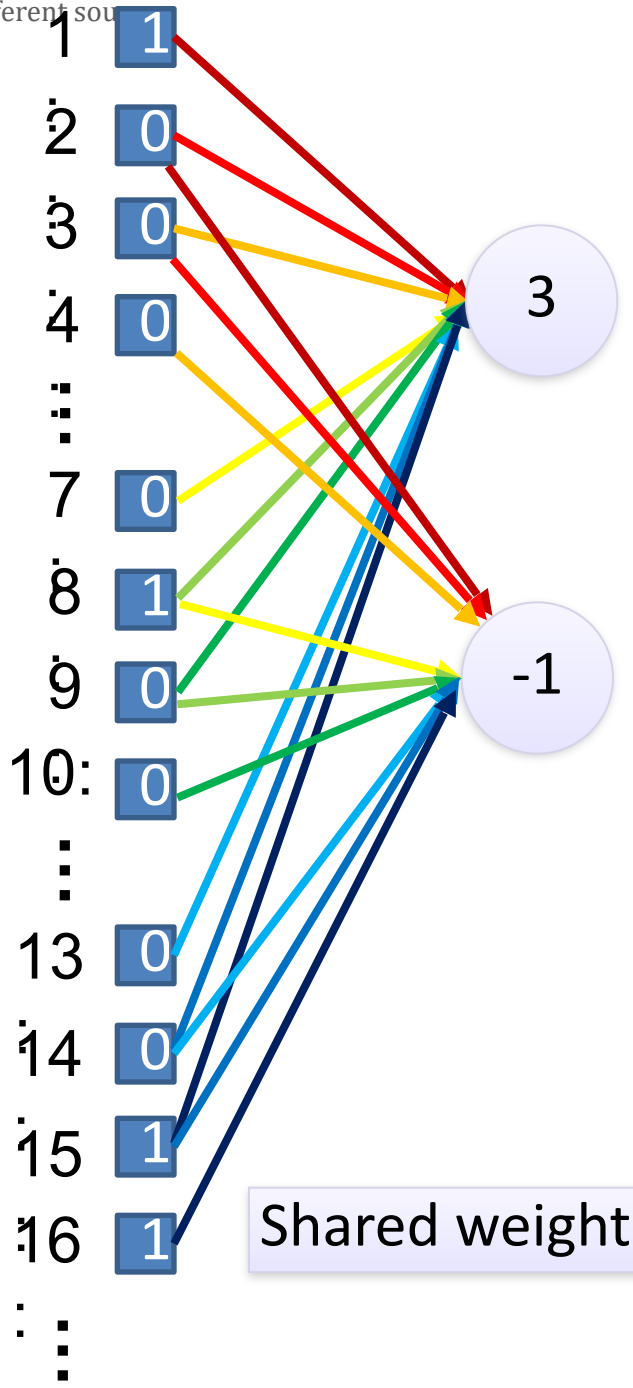


1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Fewer parameters

Even fewer parameters

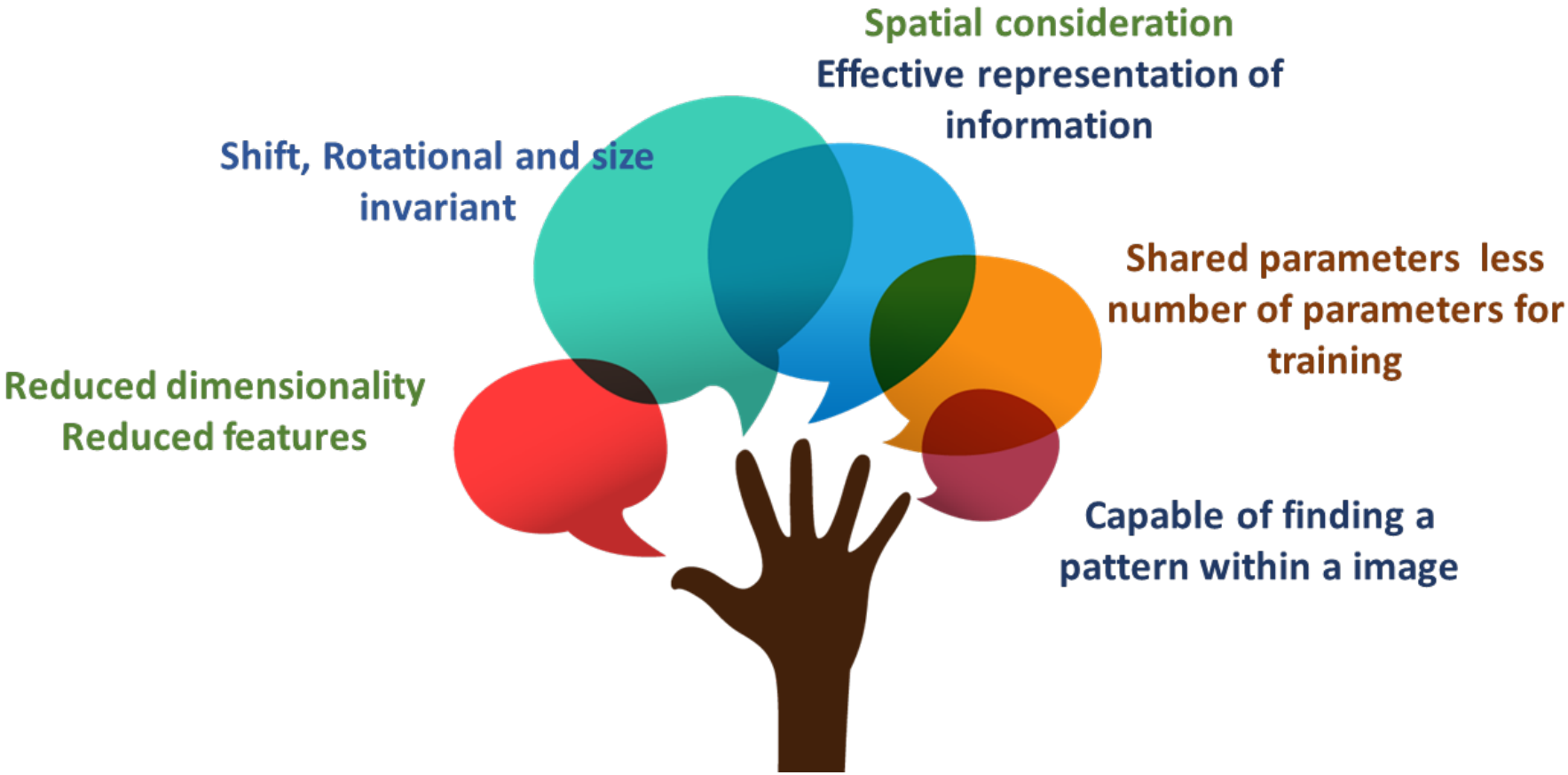


Shared weights

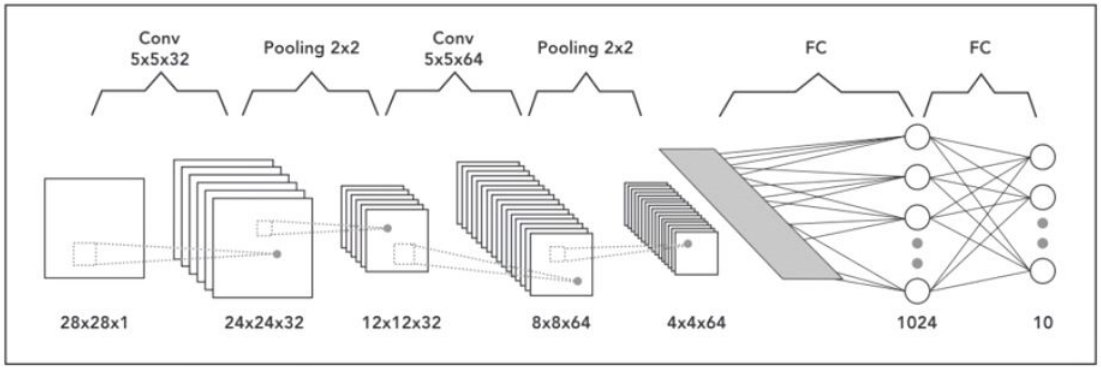
CNN Overview

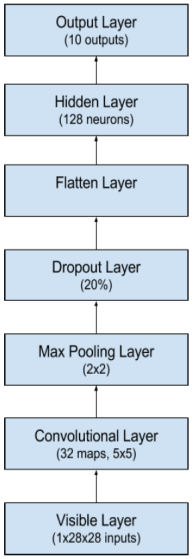
- Better than fully connected network as it considers the spatial aspect .
- Use shared weights to reduce the number of parameters and hence number of trainable parameters are reasonable
- Represent a small region with fewer parameters hence reducing dimensionality
- The kernels (shared weights) are employed for learning the patterns, giving importance to only those features that are important
- The patterns in the input data will be further refined
- Little or no invariance to shifting, scaling, and other forms of distortion
- Network is capable of finding a object/ feature in any part of the image.

Convolution Layer



Architecture:





CNN Architecture for MNIST

Summary of CNN

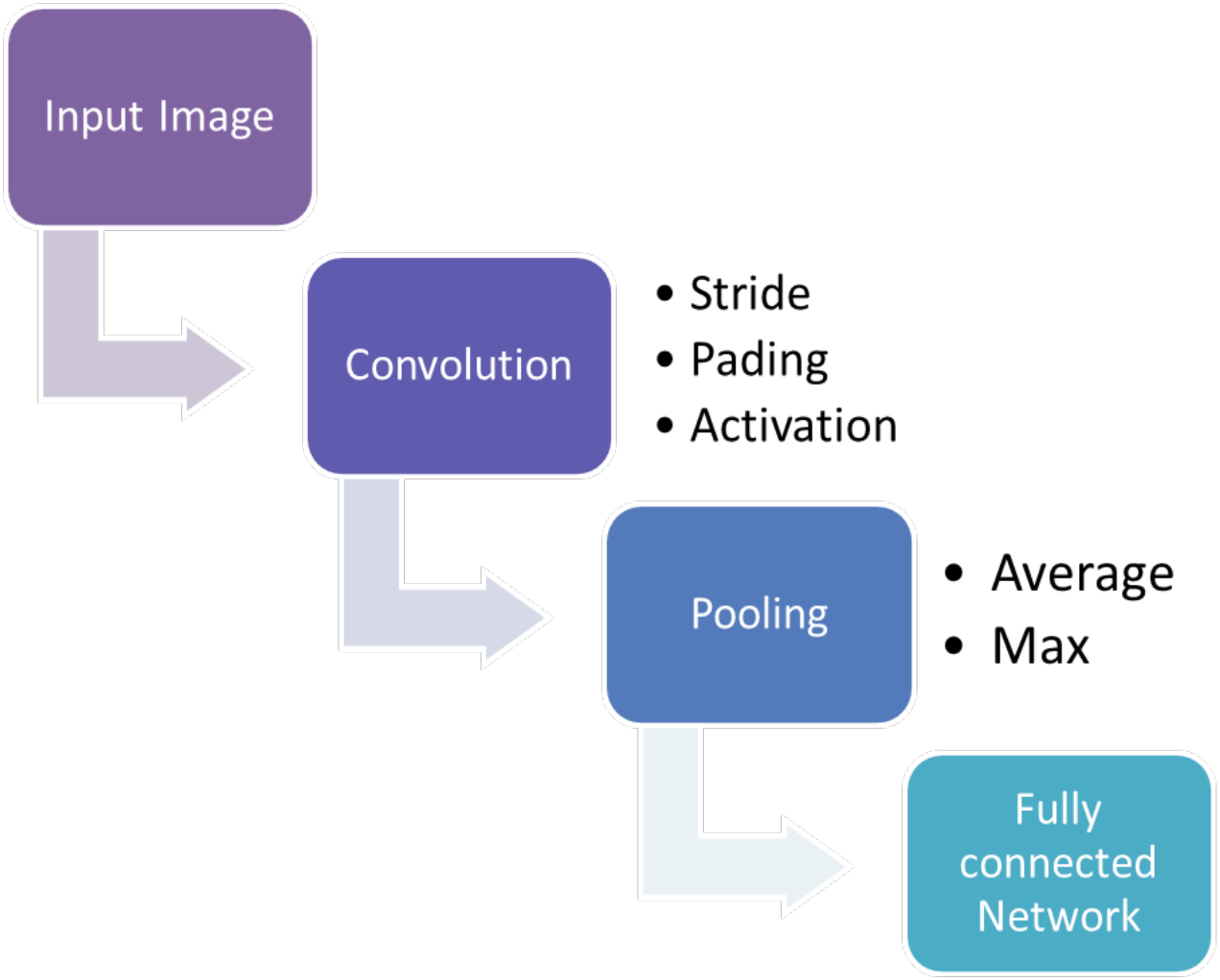
- CNN helps in analyzing the imaginary.
- Q -> What human see vs what system see?
- Size of image reducing with increase in stride value.
- Padding the input image with zero - retains depth.
- Kernal/filter that extracts useful info (edges, color, ...)
- Pooling helps in reducing the spatial size of the image.
- Output volume is controled by 3 parameters (no. of filters, stride, zero padding) - ([W-F+2P]/S) + 1.
- Output layer in CNN is a FC layer.

CNN: INPUT => CONV => RELU => FC => SOFTMAX

CNN that accepts an input, applies a convolution layer, then an activation layer, then a fully-connected layer, and, finally, a softmax classifier to obtain the output classification probabilities.

Architecture of CNN

- A typical CNN has 4 layers



Quiz

Ex:

Q1. Consider input image of 32X32X3 and we have 10 -> 5x5 filters with stride 1, pad 2.

What is output volume?

Q2 . Number of parameters in this layer?

Key

- (32+2*2-5)/1+1 = 32 spatially, so 32x32x10.

- each filter has 5*5*3 + 1 = 76 params (+1 for bias) => 76*10 = 760

Datasets to work with CNN

MNIST:

- classify the handwritten digits 0–9.
- 60,000 training images and 10,000 testing images.
- 28×28 grayscale.

Kaggle Dogs vs. Cats:

- 3-class animals dataset consisting of 1,000 images per dog, cat, and panda class respectively for a total of 3,000 images.

CIFAR-10:

- CIFAR-10 consists of 60,000 32×32×3 (RGB) images resulting in a feature vector dimensionality of 3072.
- standard benchmark dataset for image classification
- airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks.

SMILES:

13,165 grayscale images in the dataset, with each image having a size of 64×64.

CALTECH-101:

- dataset of 8,677 images includes 101 categories.
- popular benchmark dataset for object detection.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC):

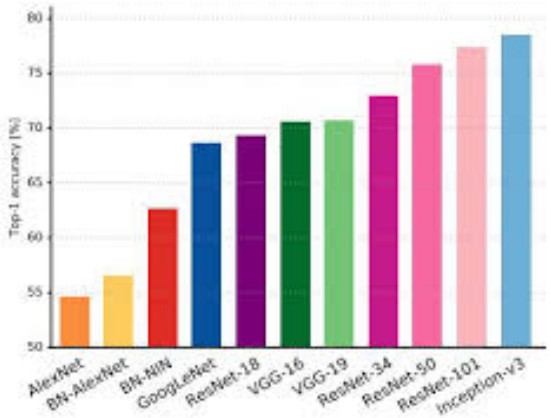
- 1,000 separate categories using approximately 1.2 million images for training, 50,000 for validation, and 100,000 for testing

pascal voc 2012 dataset:'

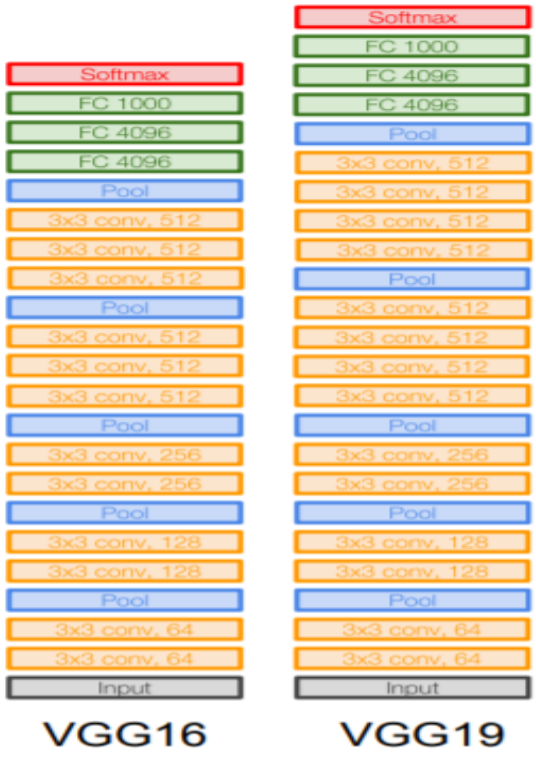
- popular dataset for building and evaluating algorithms for image classification, object detection, and segmentation.

Backbone Architectures

- VGG16
- VGG19
- Lenet
- Inception net
- Resnet
- Alexnet
- Googlenet



VGG Architecture



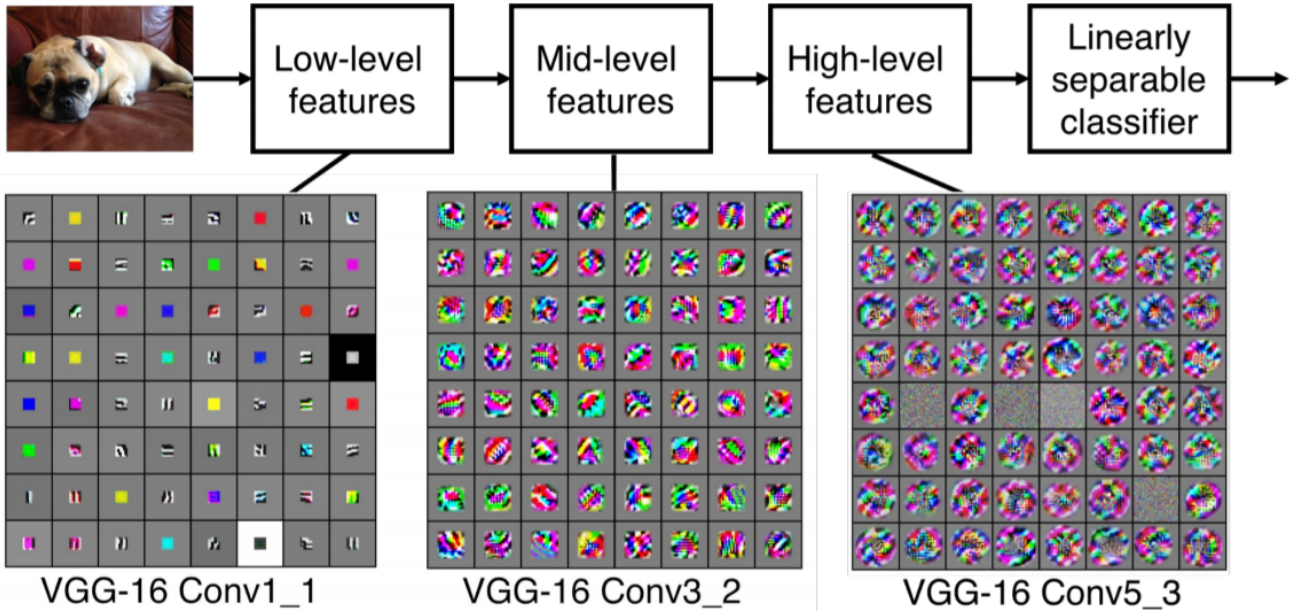
Achieved excellent results on the ILSVRC-2014 (ImageNet competition).

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14



Visualization

During training, the input to our ConvNets is a fixed-size 224 × 224 RGB image. The only pre-processing we do is subtracting the mean RGB value, computed on the training set, from each pixel. The image is passed through a stack of convolutional (conv.) layers, where we use filters with a very small receptive field: 3 × 3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations we also utilise 1 × 1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for 3 × 3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2 × 2 pixel window, with stride 2.

A stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks.

All hidden layers are equipped with the rectification (ReLU (Krizhevsky et al., 2012)) non-linearity. We note that none of our networks (except for one) contain Local Response Normalisation (LRN) normalisation (Krizhevsky et al., 2012): as will be shown in Sect. 4, such normalisation does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time. Where applicable, the parameters for the LRN layer are those of (Krizhevsky et al., 2012).

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 224, 224, 64)	1792
conv2d_2 (Conv2D)	(None, 224, 224, 64)	36928
max_pooling2d_1 (MaxPooling2	(None, 112, 112, 64)	0
conv2d_3 (Conv2D)	(None, 112, 112, 128)	73856
conv2d_4 (Conv2D)	(None, 112, 112, 128)	147584
max_pooling2d_2 (MaxPooling2	(None, 56, 56, 128)	0
conv2d_5 (Conv2D)	(None, 56, 56, 256)	295168
conv2d_6 (Conv2D)	(None, 56, 56, 256)	590080
conv2d_7 (Conv2D)	(None, 56, 56, 256)	590080
max_pooling2d_3 (MaxPooling2	(None, 28, 28, 256)	0
conv2d_8 (Conv2D)	(None, 28, 28, 512)	1180160
conv2d_9 (Conv2D)	(None, 28, 28, 512)	2359808
conv2d_10 (Conv2D)	(None, 28, 28, 512)	2359808
max_pooling2d_4 (MaxPooling2	(None, 14, 14, 512)	0
conv2d_11 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_12 (Conv2D)	(None, 14, 14, 512)	2359808
conv2d_13 (Conv2D)	(None, 14, 14, 512)	2359808
max_pooling2d_5 (MaxPooling2	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_1 (Dense)	(None, 4096)	102764544
dense_2 (Dense)	(None, 4096)	16781312
dense_3 (Dense)	(None, 1000)	4097000

Total params: 138,357,544
Trainable params: 138,357,544
Non-trainable params: 0

- **input size: 224 x 224;**
- **the receptive field size is 3 x 3;**
- **the convolution stride is 1 pixel;**
- **the padding is 1 (for receptive field of 3 x 3) so we keep the same spatial resolution;**
- **the max pooling is 2 x 2 with stride of 2 pixels;**
- **there are two fully connected layers with 4096 units each;**
- **the last layer is a softmax classification layer with 1000 units (representing the 1000 ImageNet classes);**
- **the activation function is the ReLU**
- **We can now calculate the number of learnable parameters**

See Network behaviour

- first convolutional layer, the network has to learn 64 filters with size 3x3 along the input depth (3). Plus, each one of the 64 filters has a bias, so the total number of parameters is 64*3*3*3 + 64 = 1792.

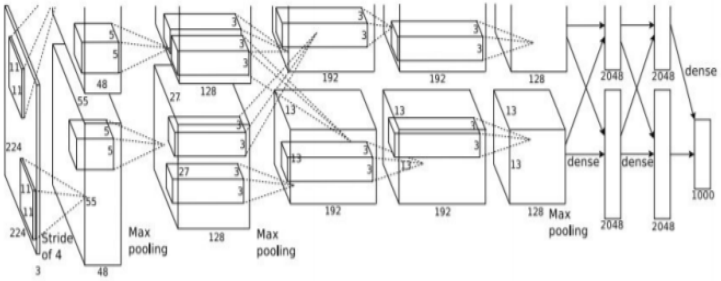
- output of the first convolutional layer will be 224 x 224 x 64.

- In pooling layer, we have to consider the size of the window and the stride.

- To calculate the number of parameters in the fully-connected layers, we have to multiply the number of units in the previous layer by the number of units in the current layer.

- Number of units in the last convolutional layer will be 7x7x512. So, the total number of parameters in the first fully-connected layer will be 7x7x512x4096 + 4096 = 102764544

Alexnet



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Input: 227x227x3 images
First layer (CONV1): 96 11x11 filters applied at stride 4

Q: what is the output volume size?

Hint: (227-11)/4+1 = 55

Paramaters?
Input: 227x227x3 images
First layer (CONV1): 96 11x11 filters applied at stride 4

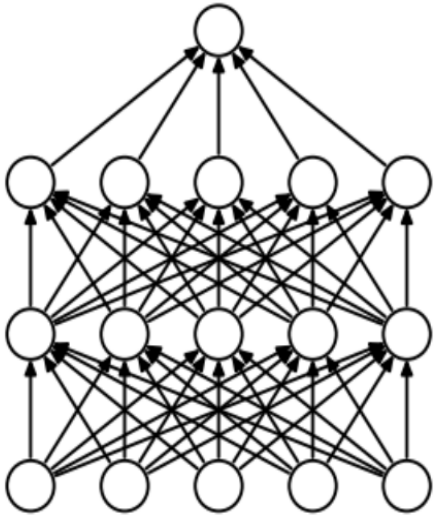
Output volume [55x55x96]
Parameters: (11*11*3)*96 = 34974

Dropout layer

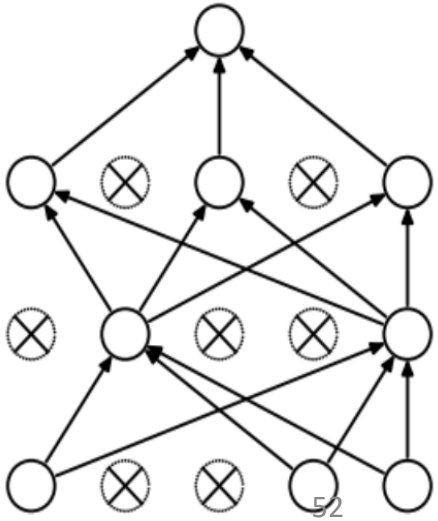
Used during training to reduce overfitting

- Drops a random set of activations in a layer by setting them to zero.
- For different training samples, different nodes are dropped

Dropout for different layers can be different



(a) Standard Neural Net

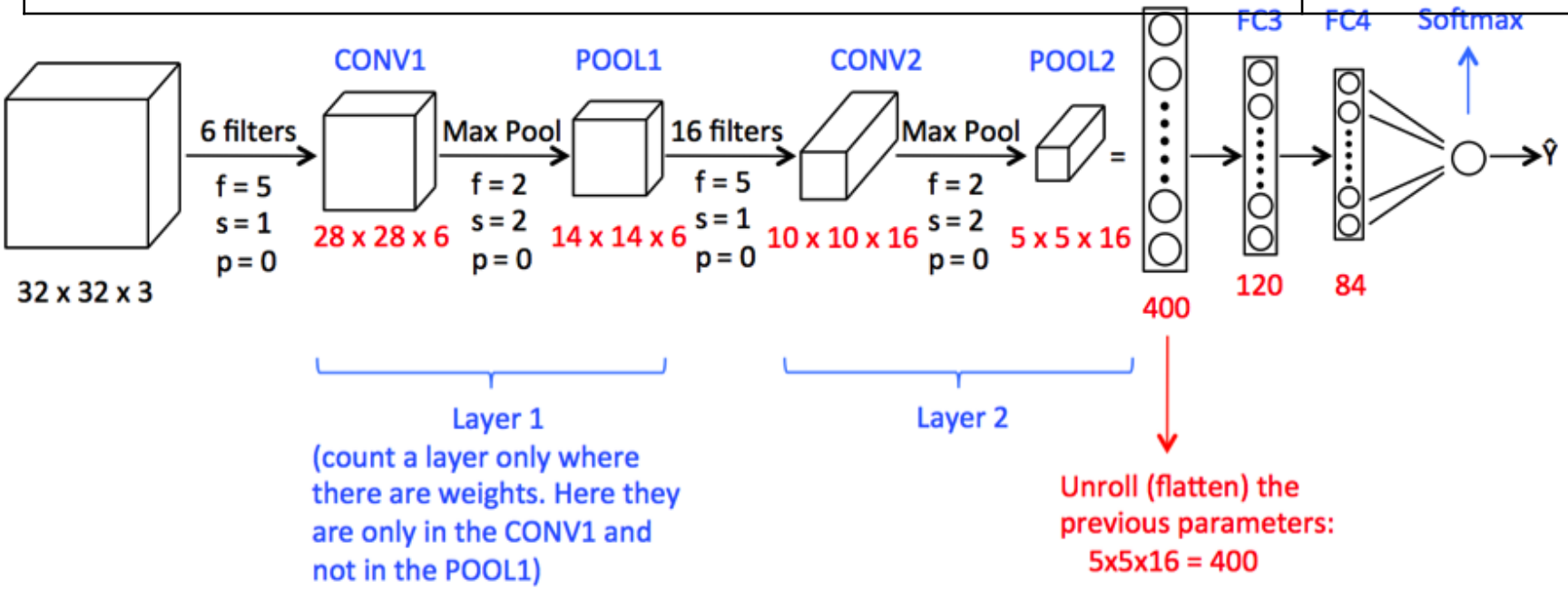


(b) After applying dropout.

https://images.app.goo.gl/LhYZ4VxGAiUPN9258

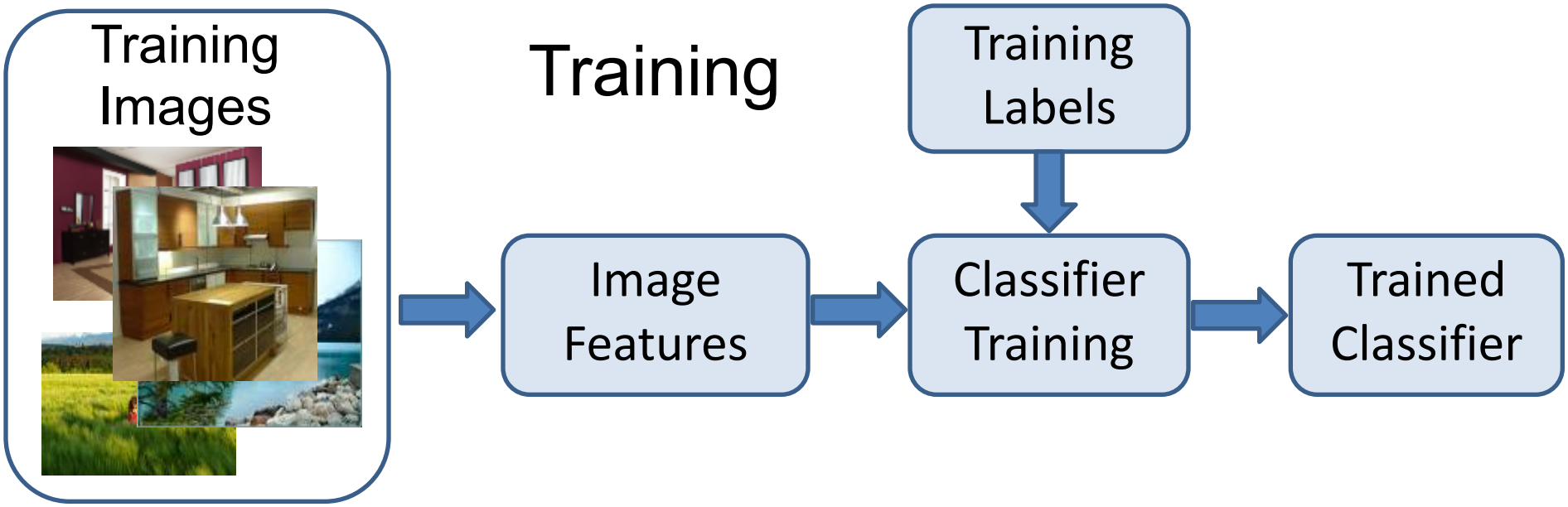
Simple convolution n/w example

layers	Activation shape	Activation size	#parameters
input	(32,32,3)		
Conv1 (f=5,s=1,p=0)	(28,28,6)		
Pool1 (f=2,s=2)	(14,14,6)		
Conv1 (f=5,s=1,p=0)	(10,10,16)		
Pool2 (f=2,s=2)	(5,5,16)		
FC3	(120,1)		
FC4	(84,1)		
softmax	(10,1)		
Total trainable parameters			62006



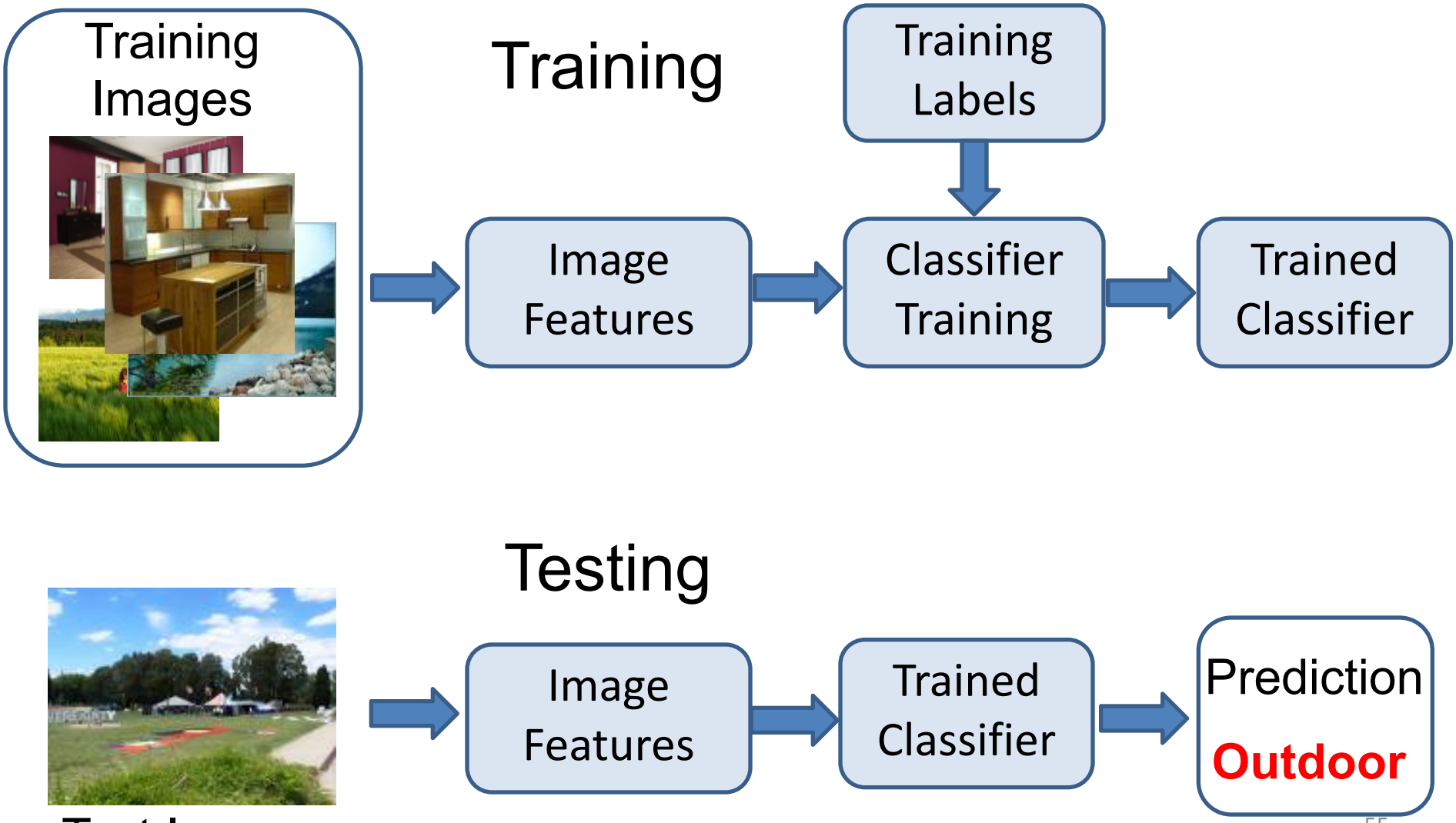
https://images.app.goo.gl/Su24Zbekb9P4pphu5

Traditional Image Categorization: Training phase



Slide credit: Jia-Bin Huang

Traditional Image Categorization: Testing phase

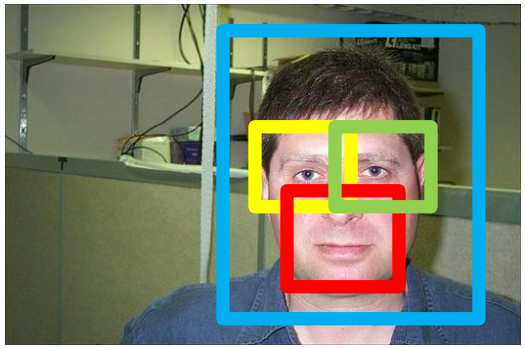


Slide credit: Jia-Bin Huang

Learning Feature Hierarchy

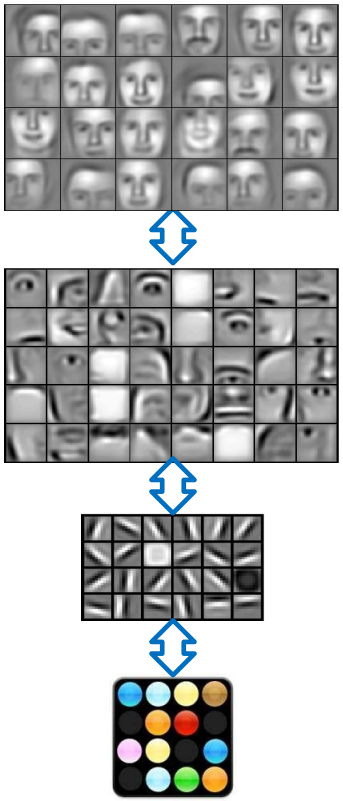
Goal: Learn **useful higher-level features** from images

Input data



Lee et al., ICML 2009;
CACM 2011

Feature representation



3rd layer
"Objects"

2nd layer
"Object parts"

1st layer
"Edges"

Pixels

Slide: Rob Fergus