MLAI 504
NEURAL NETWORKS & DEEP LEARNING

Dr. Zein Al Abidin IBRAHIM

zein.ibrahim@ul.edu.lb

CONVOLUTIONAL NEURAL NETWORKS

DEEP LEARNING

DEEP NEURAL NETWORKS PROBLEMATICS

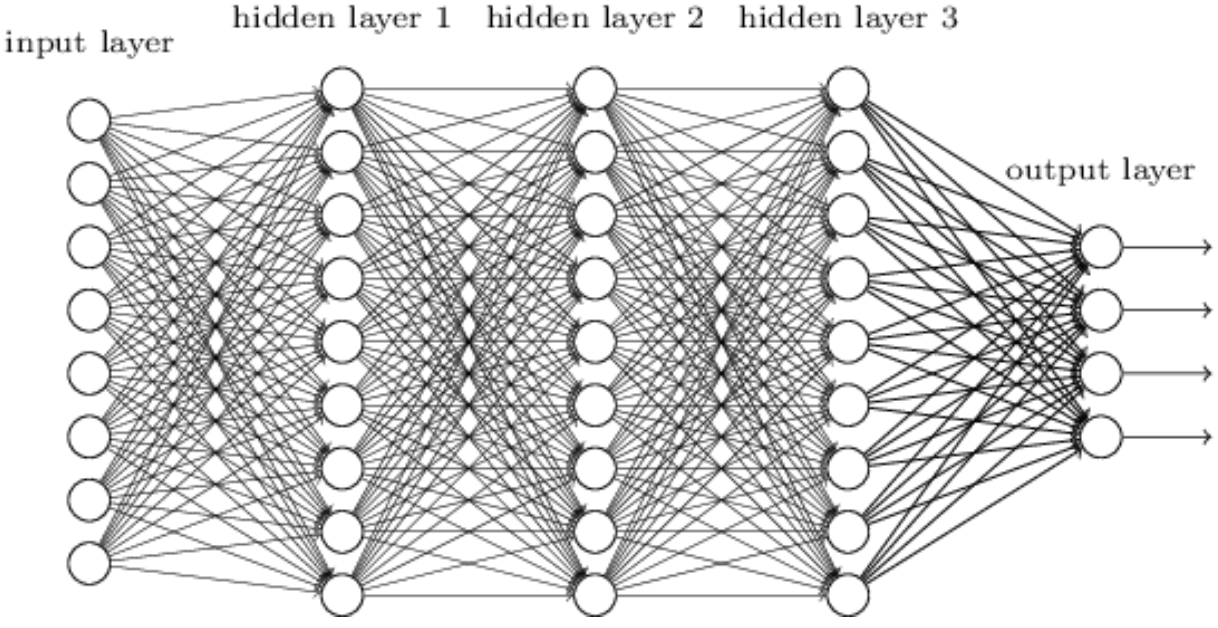
- Image size = 500x500x3
- Input dimension = 500*500*3=750000
- Number of neurons in the first hidden layer = 1000

->Number of weights for the first hidden layer = 750000*1000 = 750 million !!!

3

UTILITY OF FULLY CONNECTED LAYERS

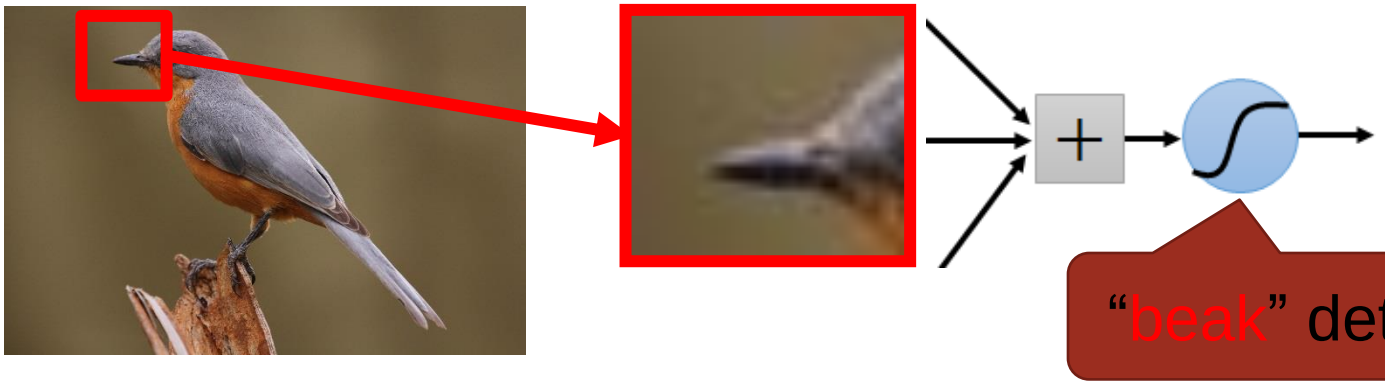
- We know it is good to learn a small model.
- From this fully connected model, do we really need all the edges?
- Can some of these be shared?



CONSIDER LEARNING AN IMAGE

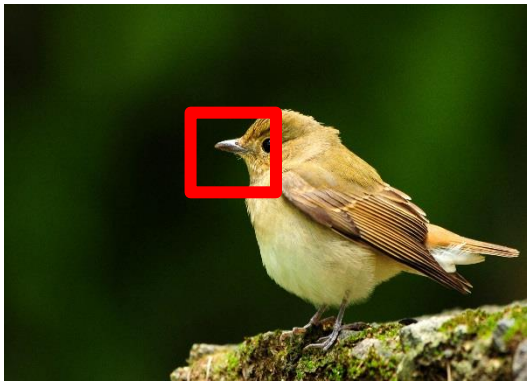
- Some patterns are much smaller than the whole image

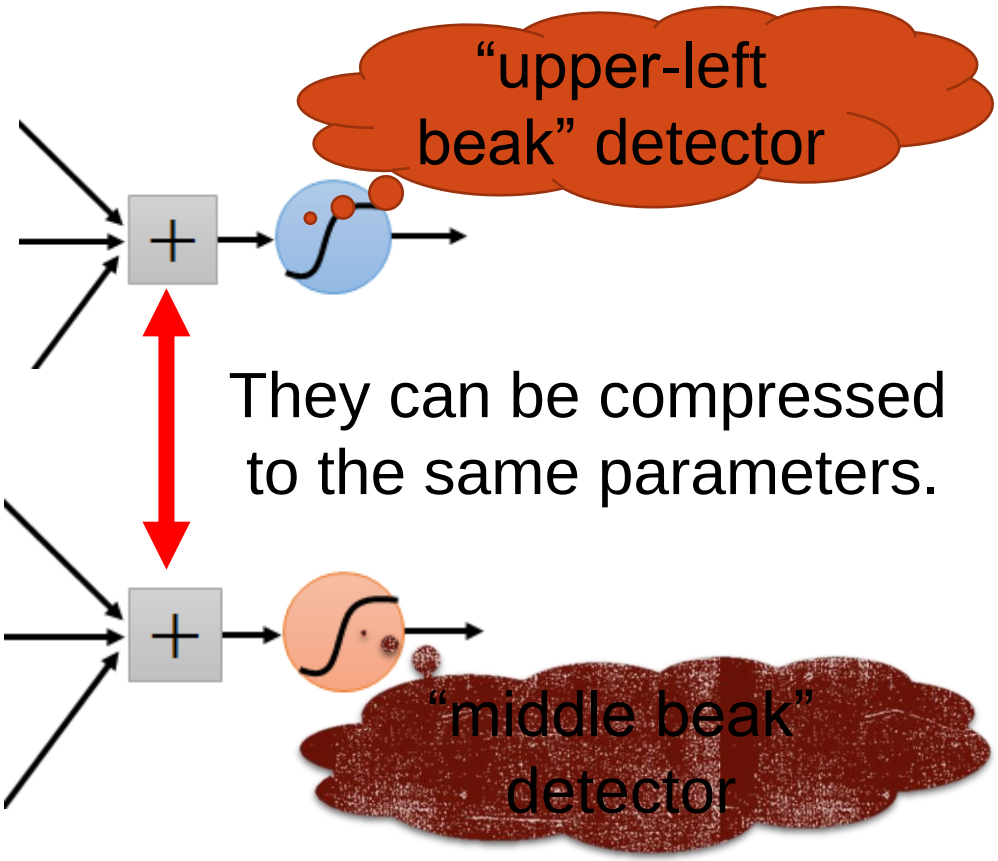
Can represent a small region with fewer parameters



SAME PATTERN APPEARS IN DIFFERENT PLACES: THEY CAN BE COMPRESSED!

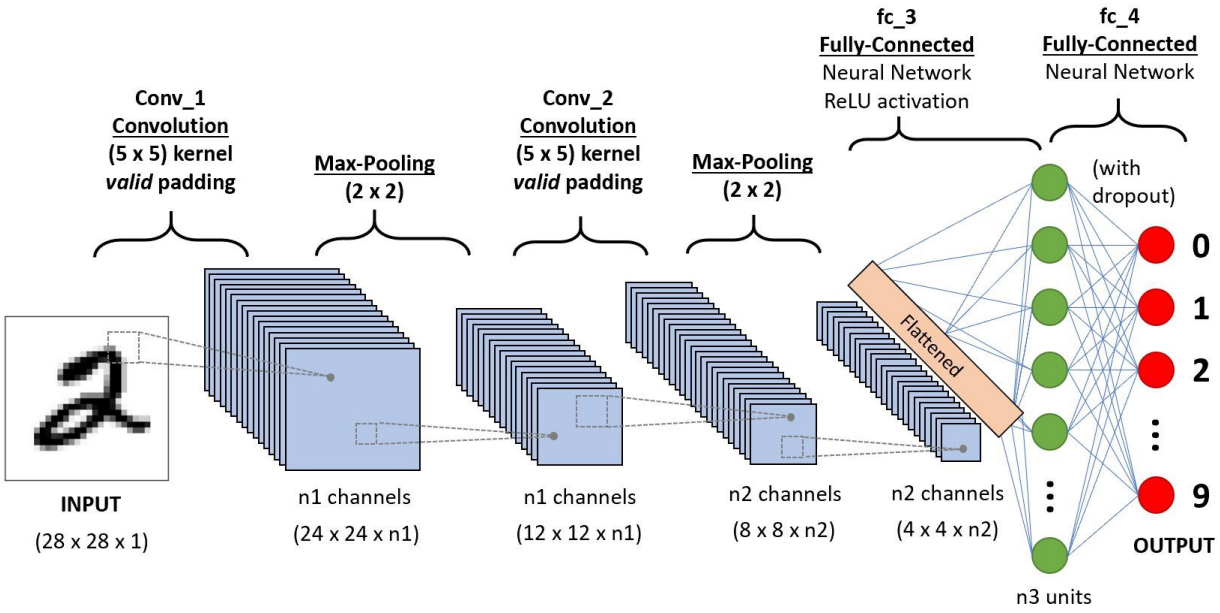
- What about training a lot of such "small" detectors and each detector must "move around".





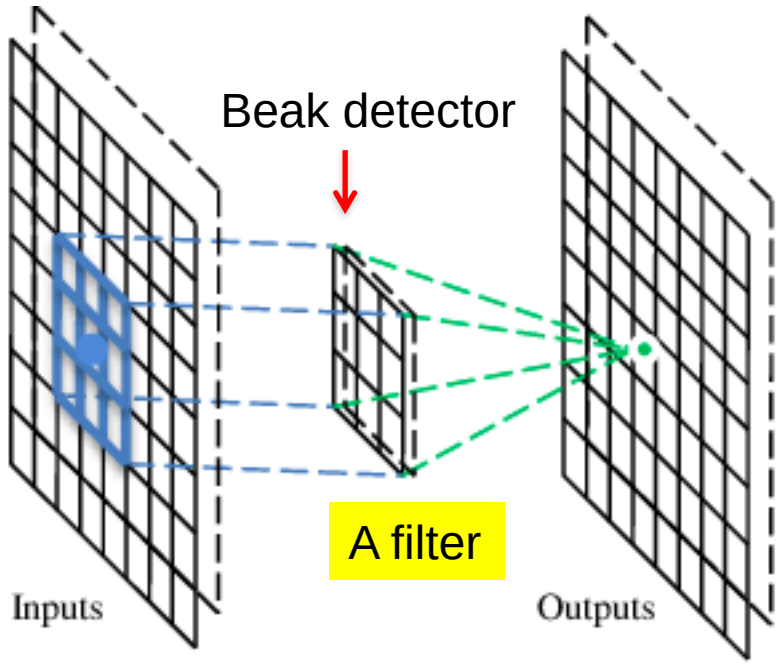
CONVOLUTION NEURAL NETWORKS

- A convolution neural network (CNN) is an MLP designed to recognize two-dimensional shapes with a high degree of invariance to translation, scaling and skewing and other sorts of distortion.
- They are mostly used in computer vision applications.



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).

CONVOLUTION

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

Filter 1

stride=1

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

Dot product

3 -1

6 x 6 image

CONVOLUTION

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

Filter 1

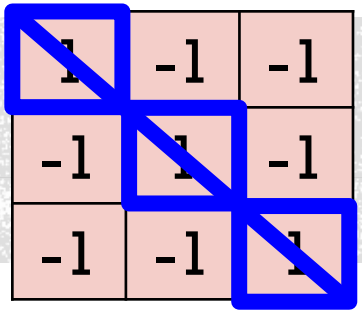
If stride=2

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

6 x 6 image

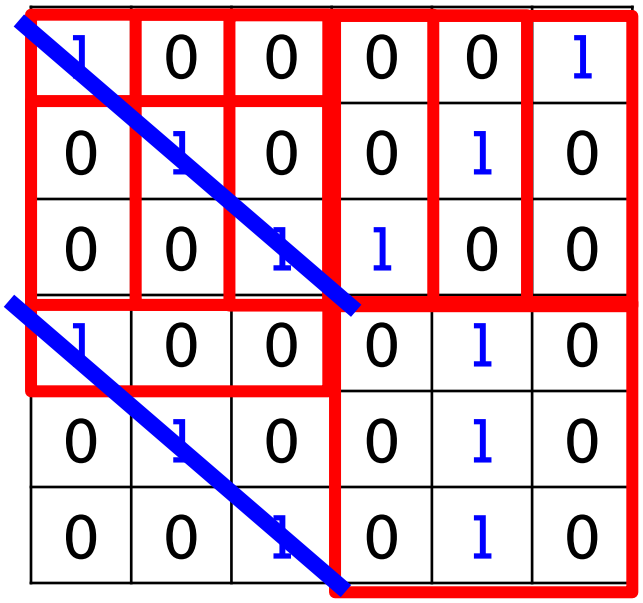
3 -3

CONVOLUTION

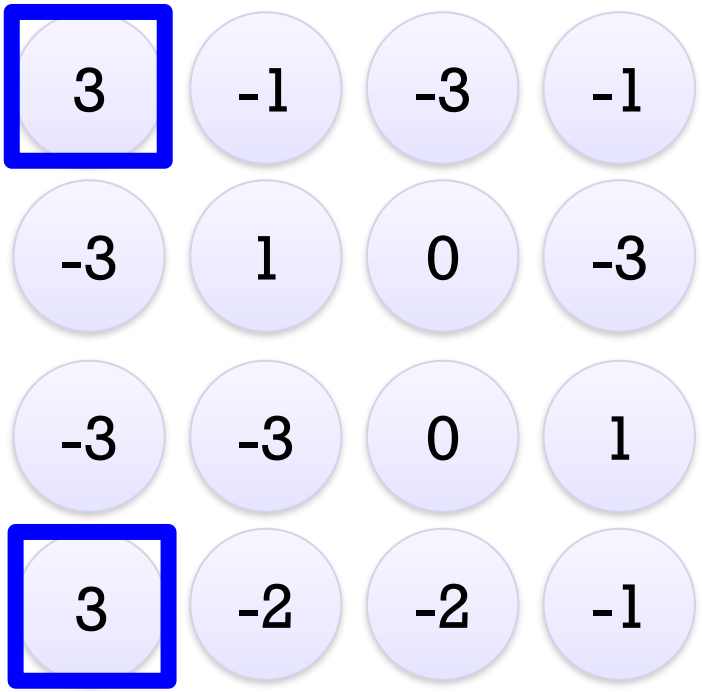


Filter 1

stride=1



6 x 6 image



CONVOLUTION

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

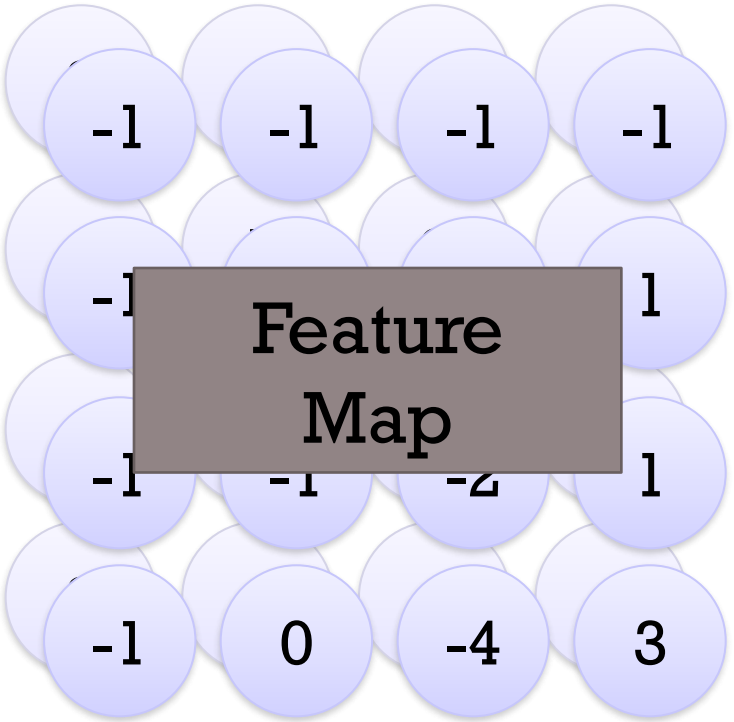
Filter 2

stride=1

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

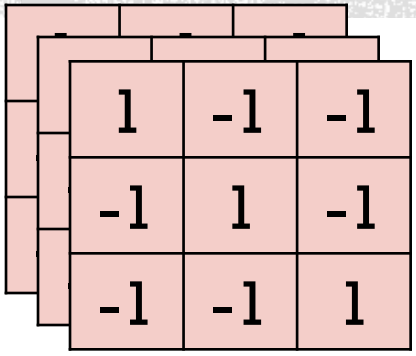
6 x 6 image

Repeat this for each filter

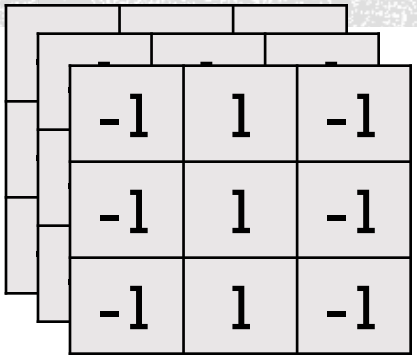


Two 4 x 4 images
Forming 2 x 4 x 4 matrix

COLOR IMAGE: RGB 3 CHANNELS

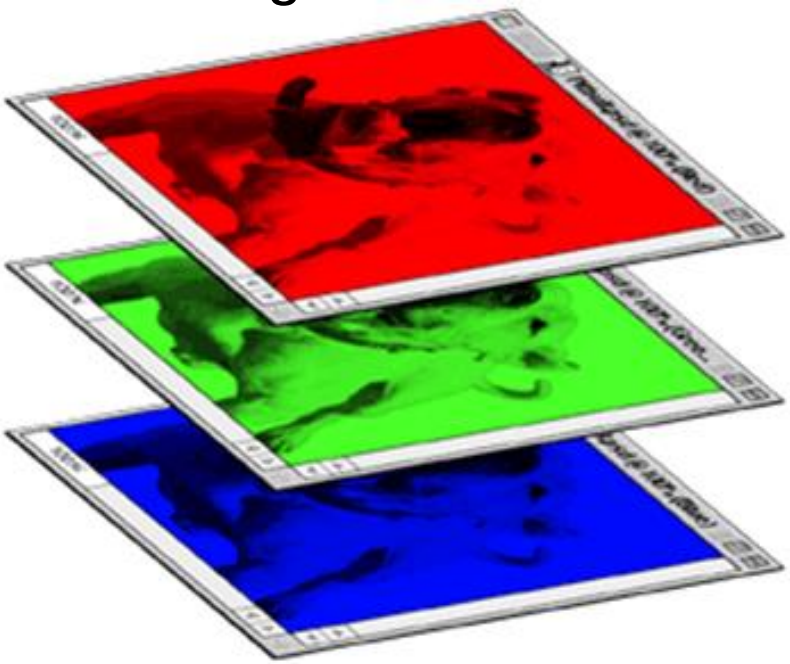


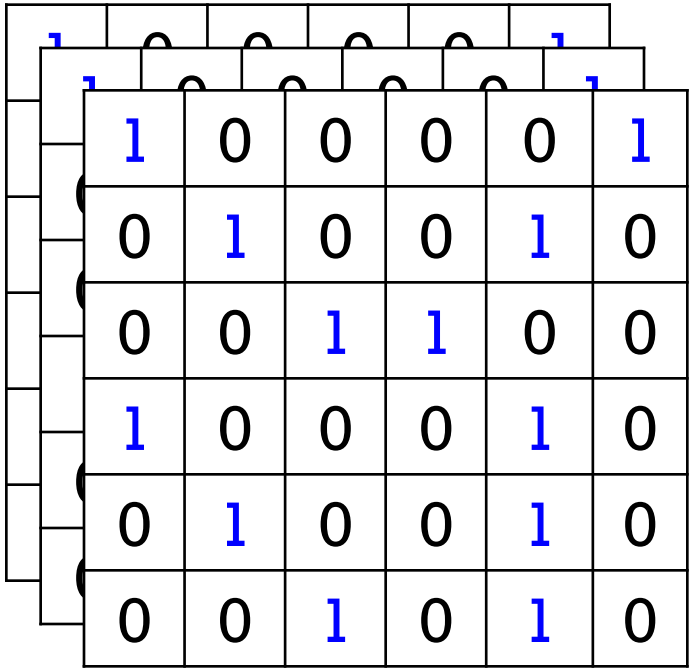
Filter 1



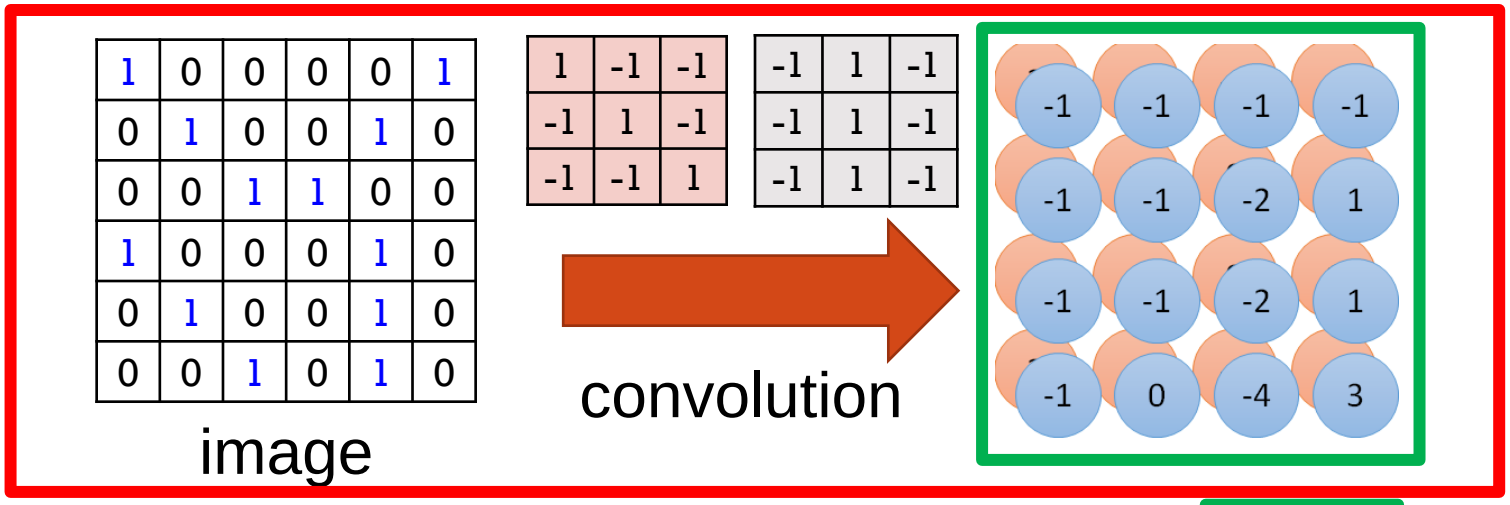
Filter 2

Color image

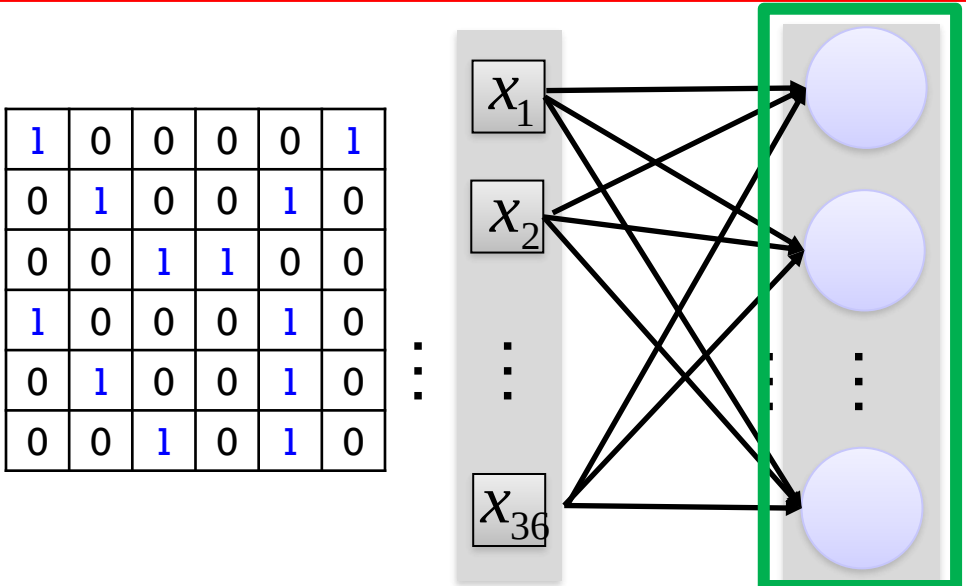


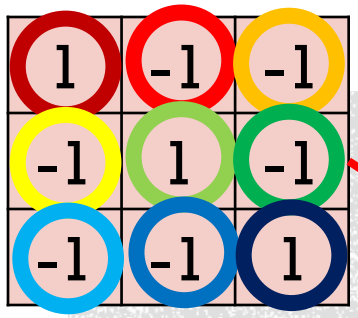


CONVOLUTION VS FULLY CONNECTED



Fully-connected

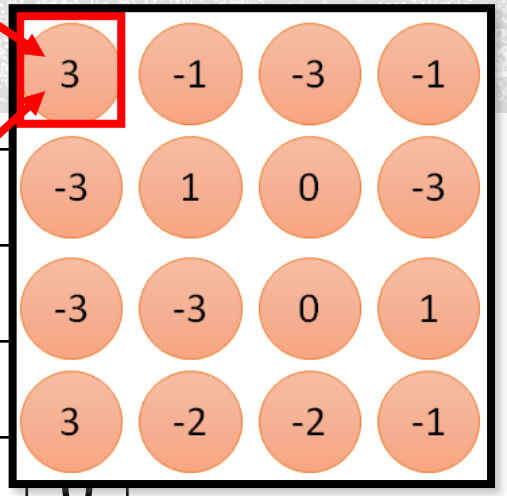


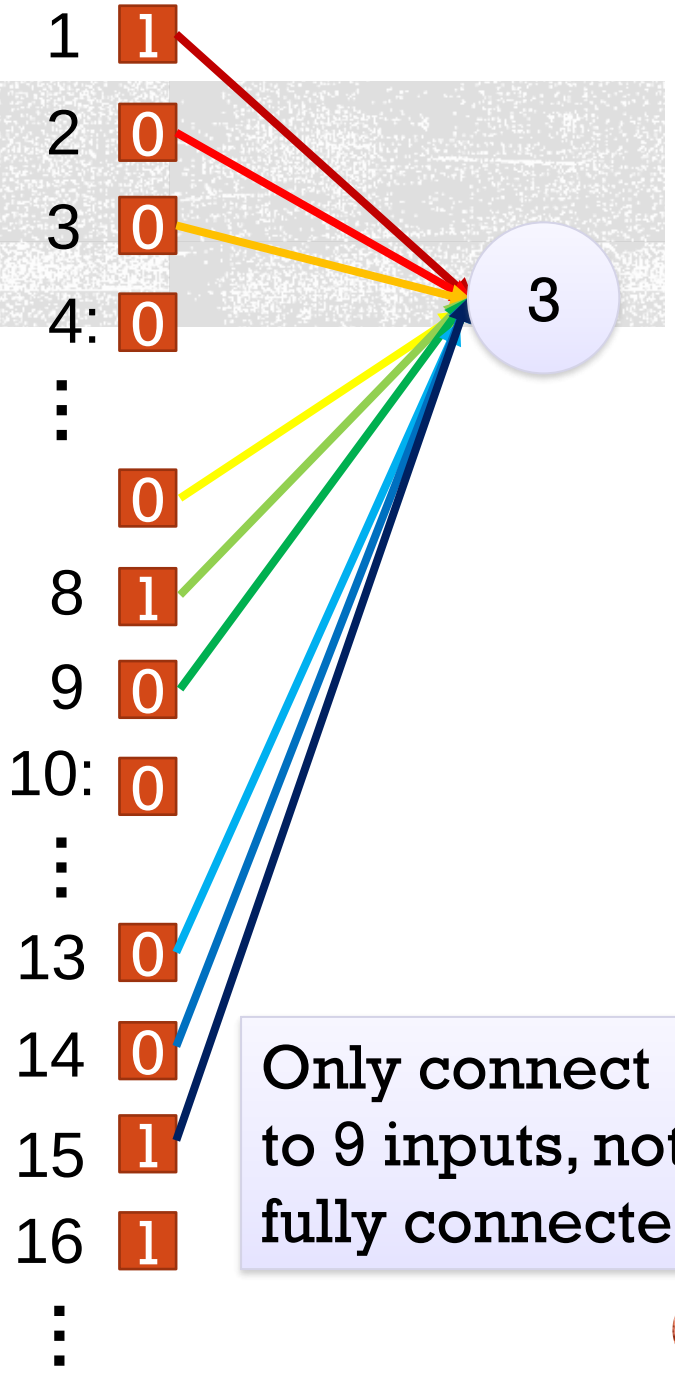


Filter 1

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

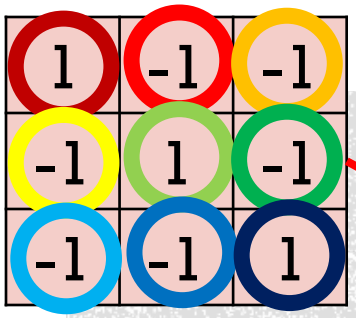
6 x 6 image



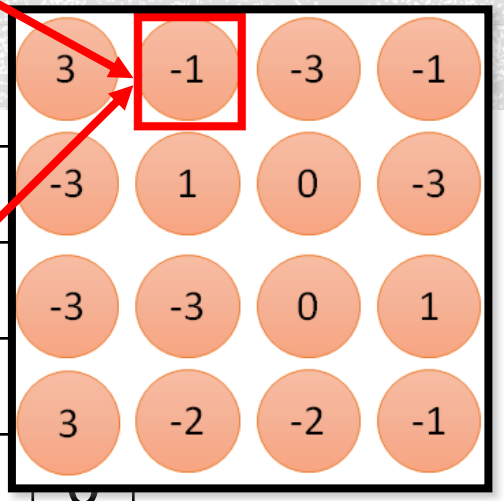


fewer parameters!

Only connect to 9 inputs, not fully connected



Filter 1

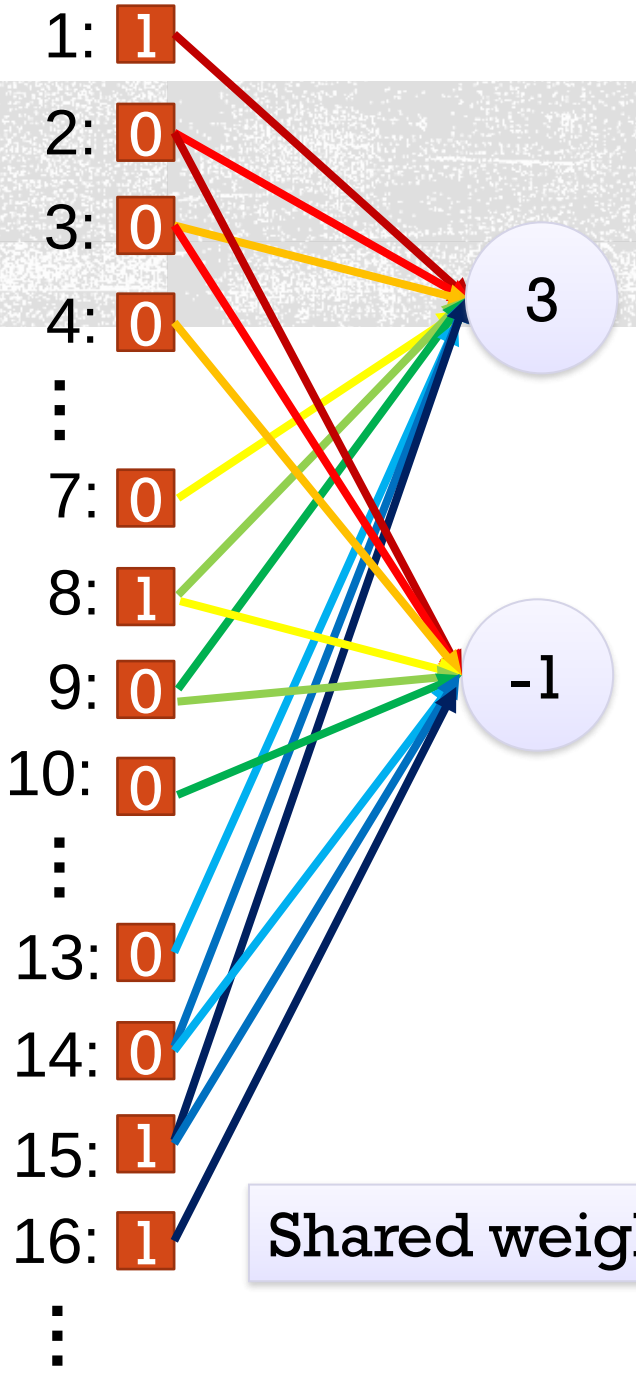


1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

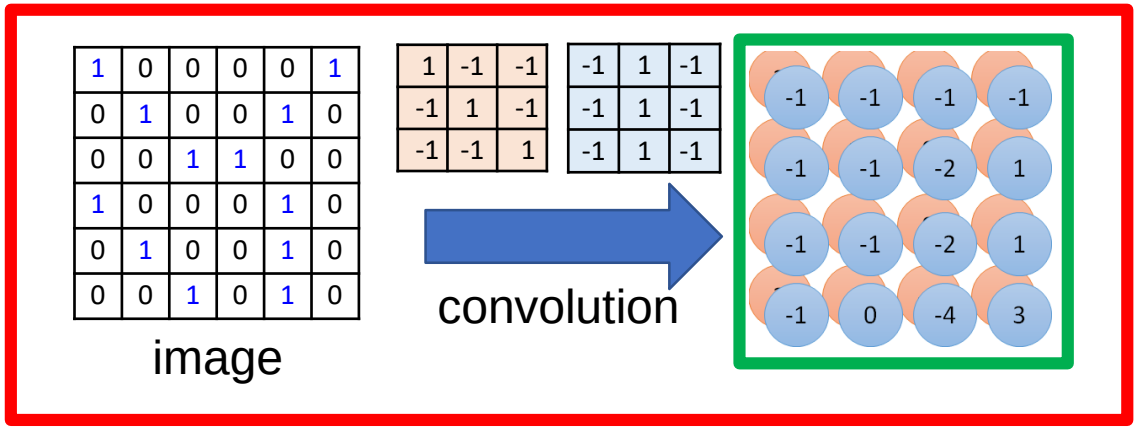
Fewer parameters

Even fewer parameters

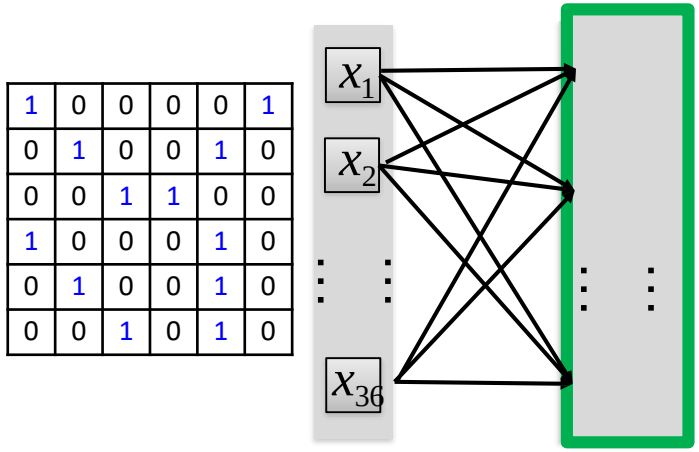


Shared weights

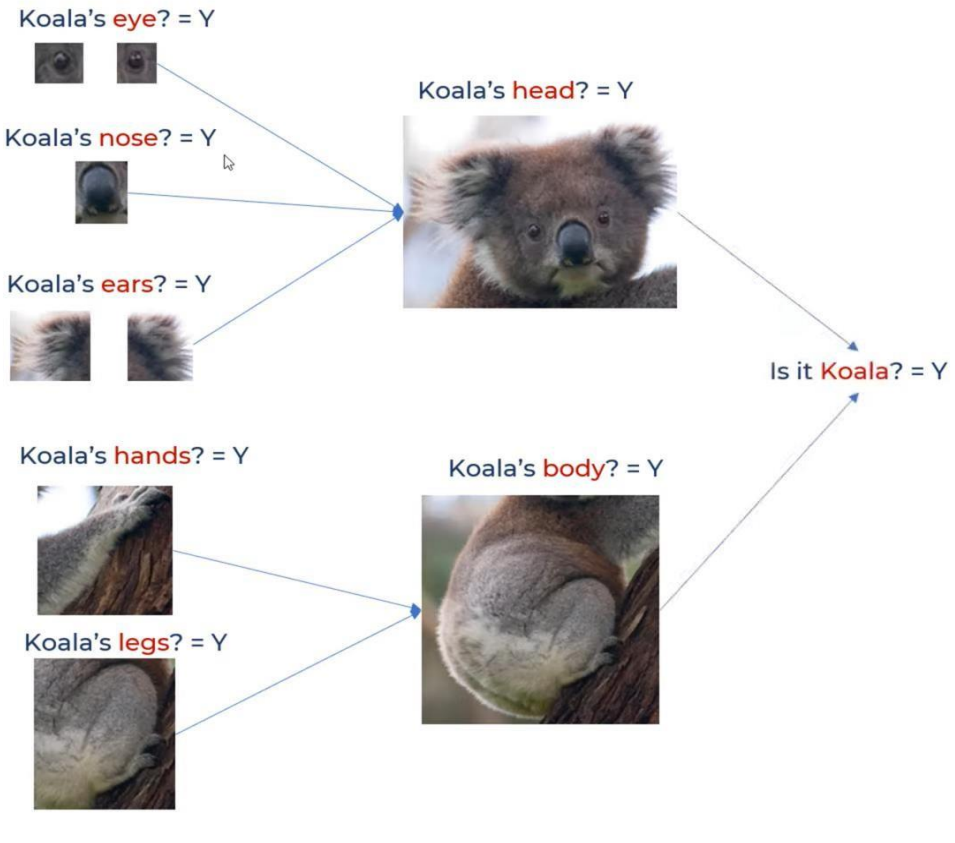
CONVOLUTION V.S. FULLY CONNECTED



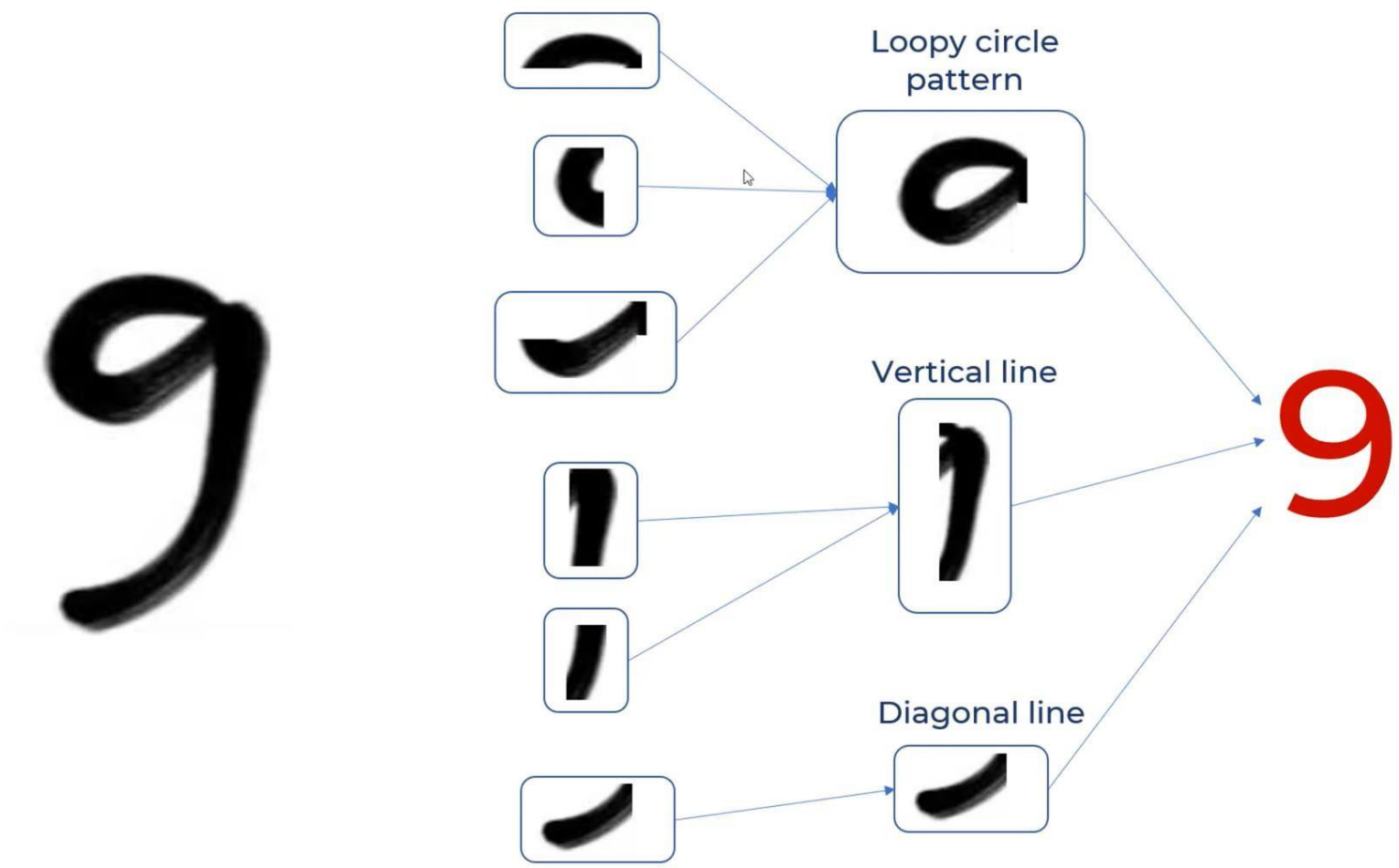
Fully-connected

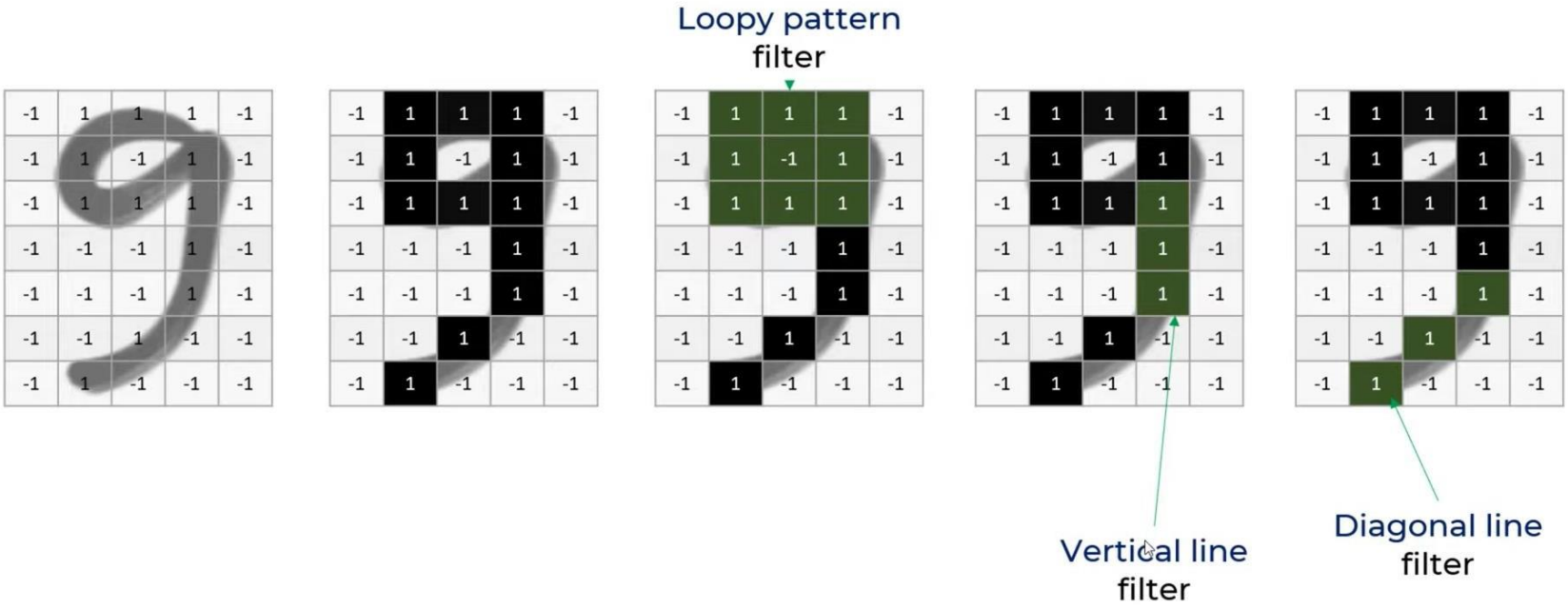


EXAMPLE


Koala's eye? = Y
Koala's nose? = Y
Koala's ears? = Y
Koala's head? = Y
Is it Koala? = Y
Koala's hands? = Y
Koala's legs? = Y
Koala's body? = Y

EXAMPLE




Loopy pattern filter
Vertical line filter
Diagonal line filter

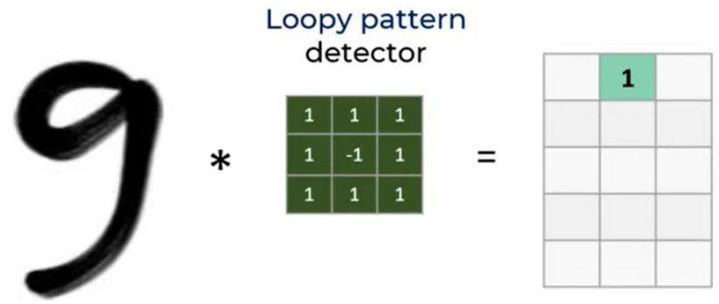
-1	**1**	**1**	**1**	-1
-1	**1**	-1	**1**	-1
-1	**1**	**1**	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	-1	**1**	-1
-1	-1	**1**	-1	-1
-1	**1**	-1	-1	-1

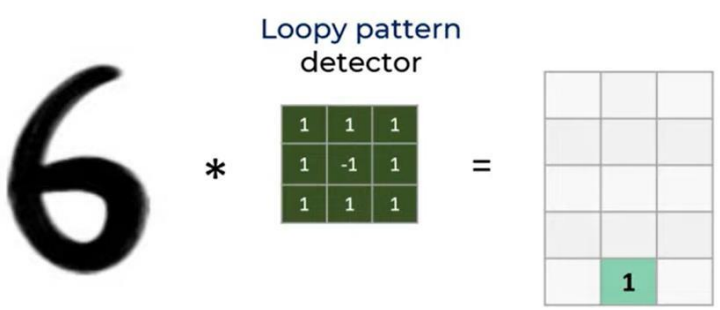
*

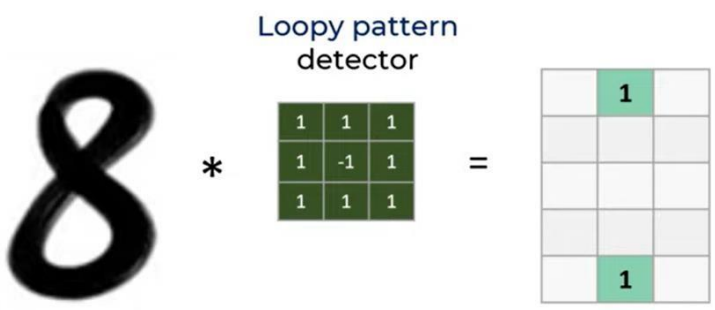
1	1	1
1	-1	1
1	1	1

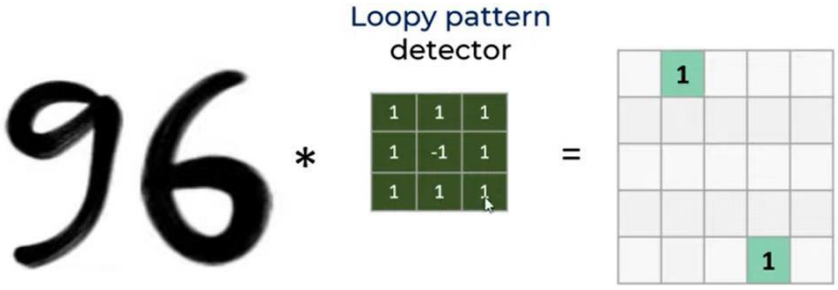
-0.11	**1**	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33

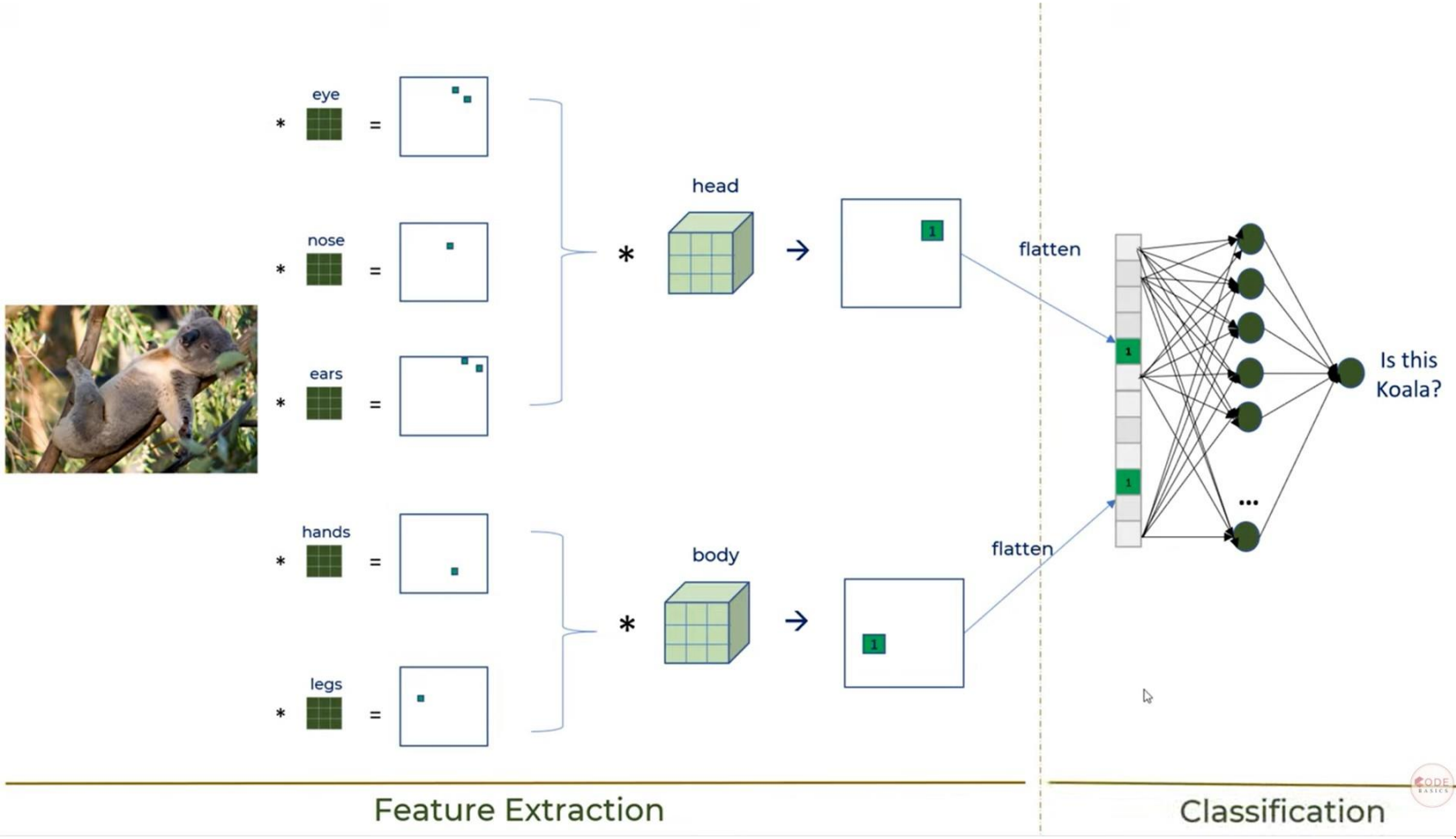
Feature Map


Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1


Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1


Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
1


Loopy pattern detector
1 1 1
1 -1 1
1 1 1
1
1

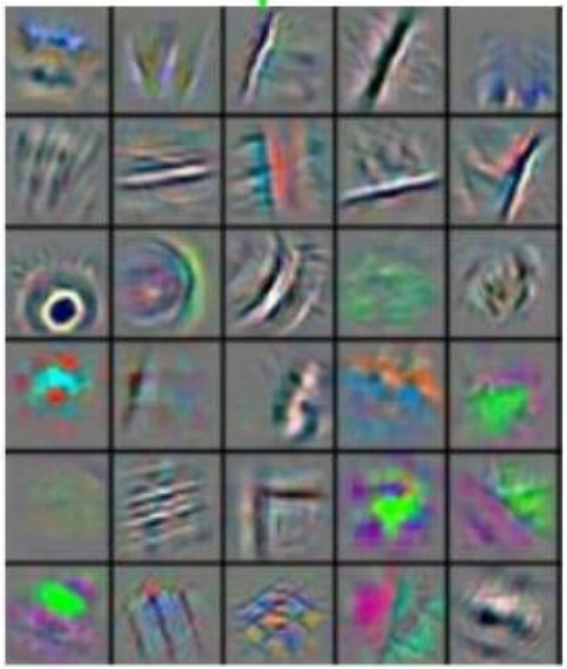

eye
nose
ears
hands
legs
head
body
flatten
flatten
Is this Koala?
Feature Extraction
Classification

Low-Level Feature

Mid-Level Feature

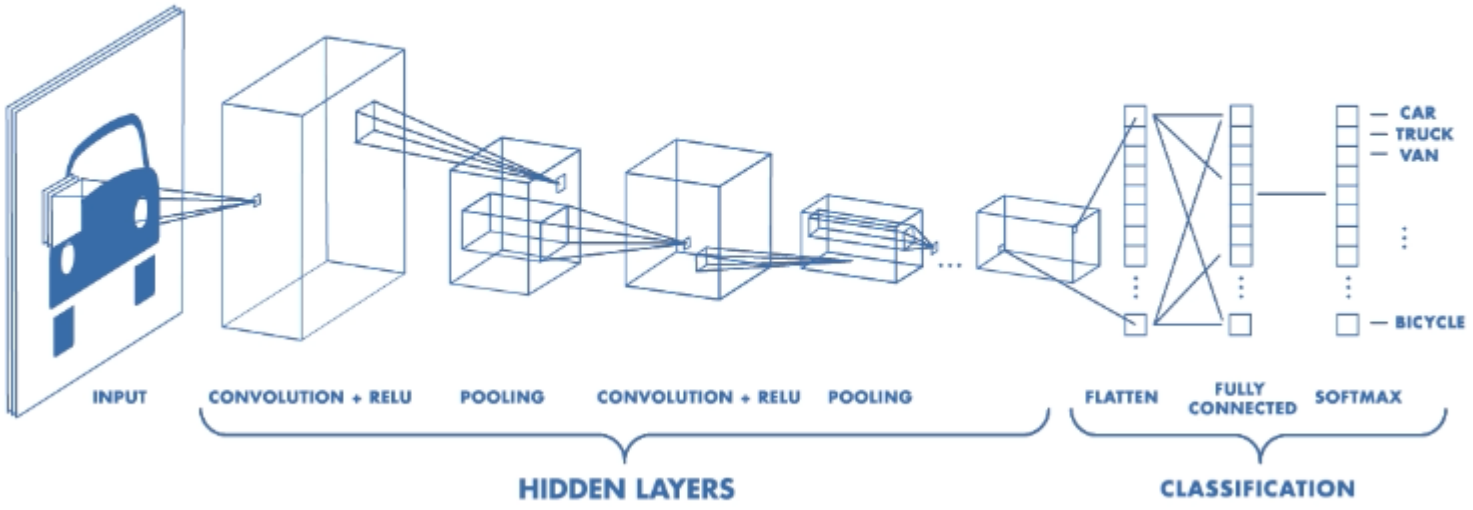
High-Level Feature

Trainable Classifier



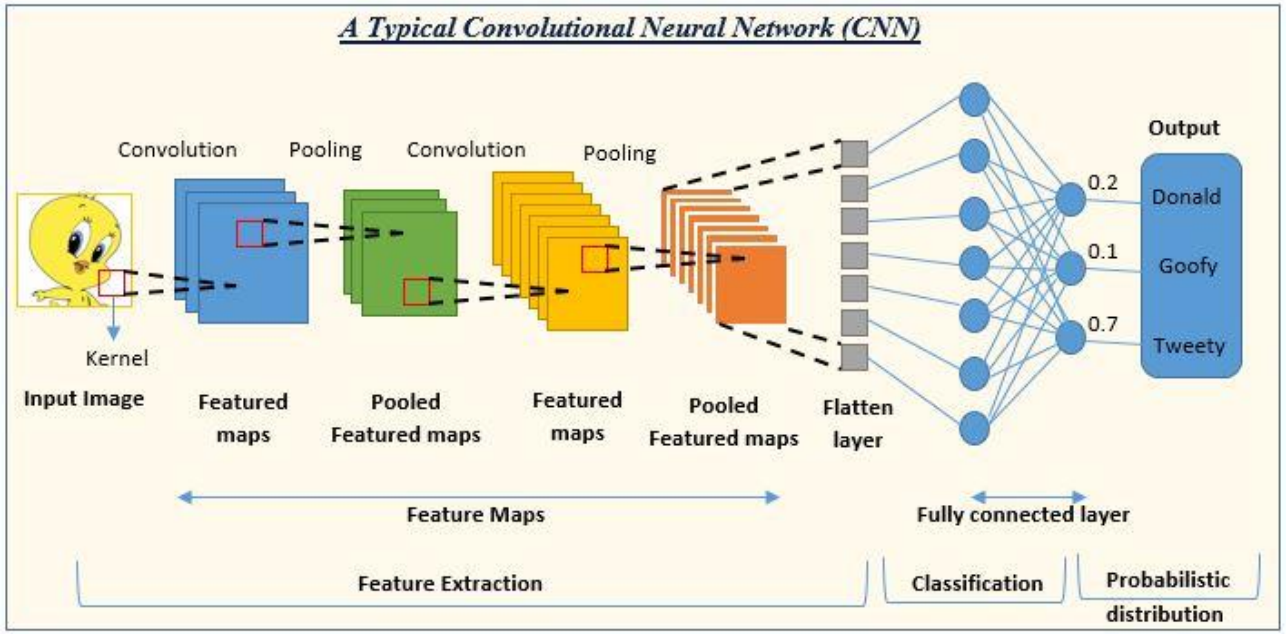
CONVOLUTIONAL NEURAL NETWORK ELABORATION

- CNN is mainly used for image recognition and classification.
- There are four operations in CNN
  - Convolution
  - Non-Linearity (ReLU) activation function
  - Subsampling or Pooling
  - Classification (fully connected layer)



CONVOLUTIONAL NEURAL NETWORK ELABORATION

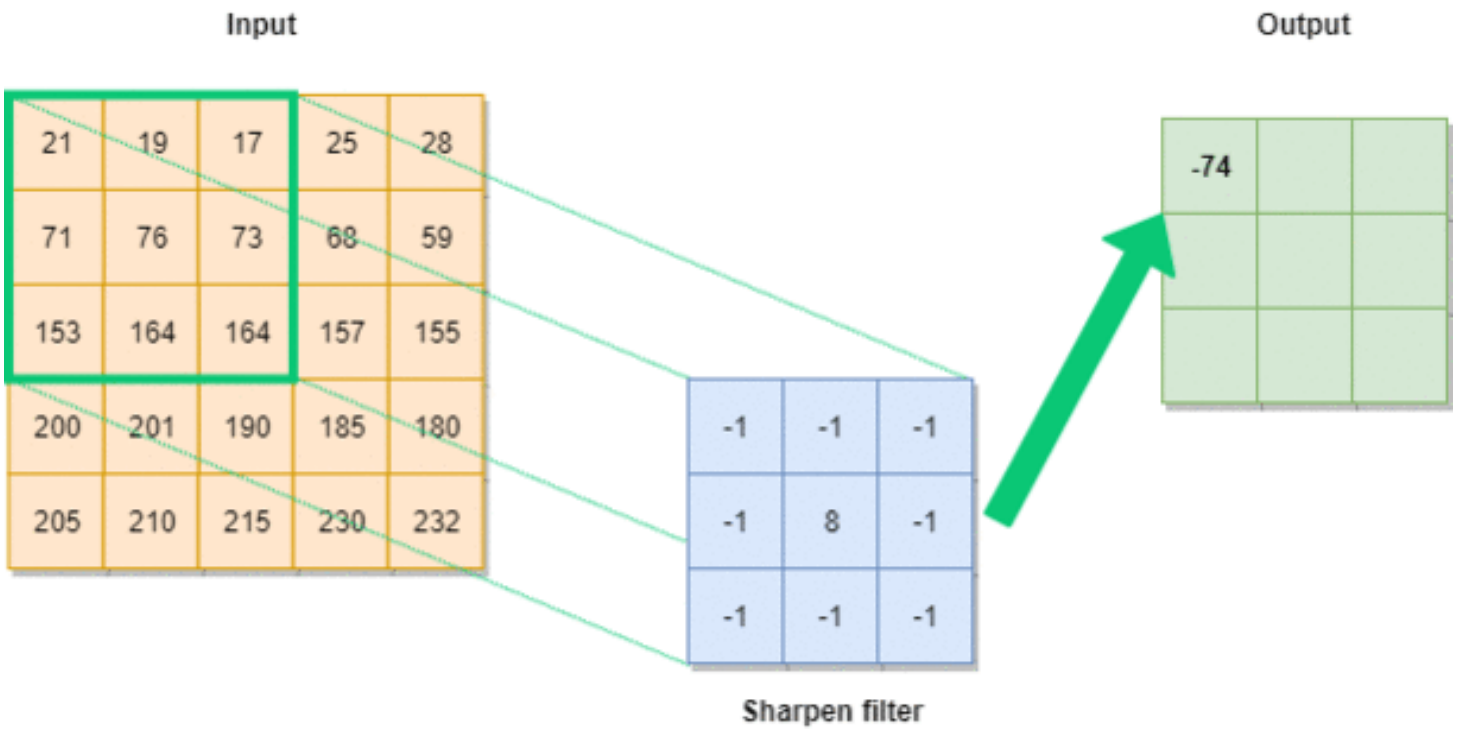
- CNN is mainly used for image recognition and classification.
- There are four operations in CNN
  - Convolution
  - Non-Linearity (ReLU) activation function
  - Subsampling or Pooling
  - Classification (fully connected layer)

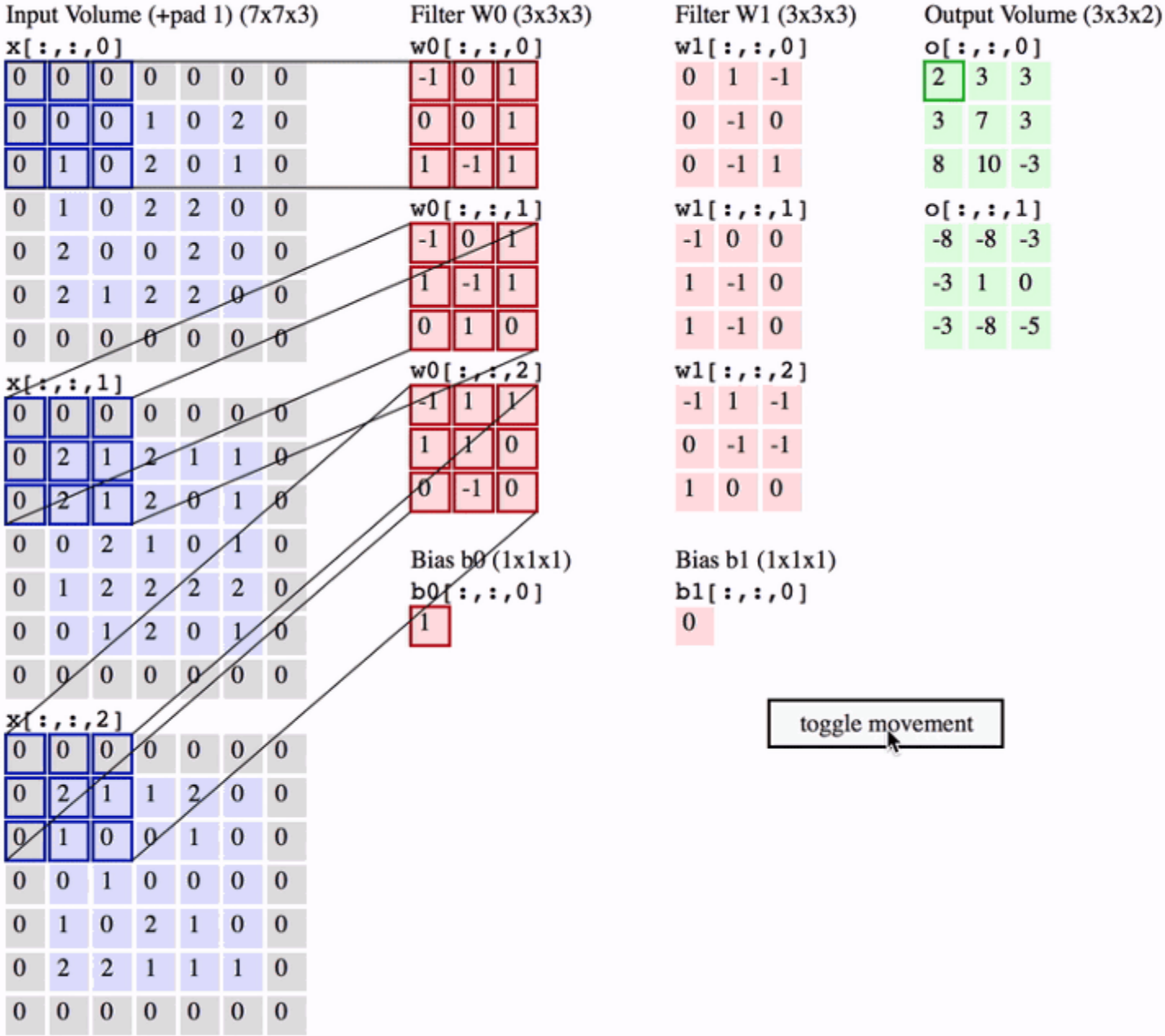


CONVOLUTIONAL NEURAL NETWORK ELABORATION

- CNN mainly used for image recognition and classification.
- There are four operations in CNN
  - Convolution
  - Non-Linearity (ReLU) activation function
  - Subsampling or Pooling
  - Classification (fully connected layer)

CONVOLUTION OPERATION




Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 0 1 0 2 0
0 1 0 2 0 1 0
0 1 0 2 2 0 0
0 2 0 0 2 0 0
0 2 1 2 2 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 2 1 2 1 1 0
0 2 1 2 0 1 0
0 0 2 1 0 1 0
0 1 2 2 2 2 0
0 0 1 2 0 1 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 2 1 1 2 0 0
0 1 0 0 1 0 0
0 0 1 0 0 0 0
0 1 0 2 1 0 0
0 2 2 1 1 1 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
-1 0 1
0 0 1
1 -1 1
w0[:,:,1]
-1 0 1
1 -1 1
0 1 0
w0[:,:,2]
-1 1 1
1 1 0
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
0 1 -1
0 -1 0
0 -1 1
w1[:,:,1]
-1 0 0
1 -1 0
1 -1 0
w1[:,:,2]
-1 1 -1
0 -1 -1
1 0 0
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
2 3 3
3 7 3
8 10 -3
o[:,:,1]
-8 -8 -3
-3 1 0
-3 -8 -5
toggle movement

THREE-CHANNELS IMAGE → AVERAGING

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

$308 + (-498) + 164 + 1 = -25$

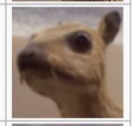
Bias = 1

Output

-25				...
				...
				...
				...
...	...	...	...	...

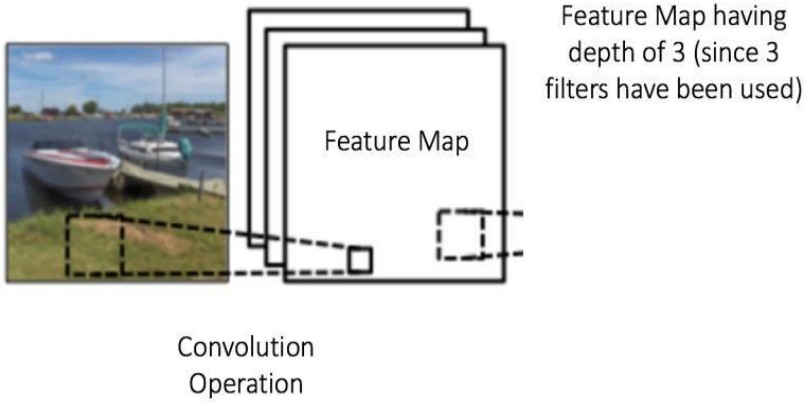
CONVOLUTIONAL NEURAL NETWORK ELABORATION

- Convolution
  - The 3x3 matrix is called a filter or kernel or feature detector which is applied by sliding (striding).
    - The matrix formed is called the feature map.
  - Many filters exits such as
    - Edge detection
    - Blur
    - Sharpen
  - In a CNN, each filter detect different feature of the input image.
  - In practice, a CNN *learns* the values of these filters on its own during the training process
    - (although we still need to specify parameters such as number of filters, filter size, architecture of the network etc. before the training process).
    - The more number of filters we have, the more image features get extracted and the better our network becomes at recognizing patterns in unseen images.

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

CONVOLUTIONAL NEURAL NETWORK ELABORATION

- The size of the Feature Map (Convolved Feature) is controlled by three parameters that we need to specify before the convolution step is performed:
  - **Depth:** Depth corresponds to the number of filters we use for the convolution operation in the network.



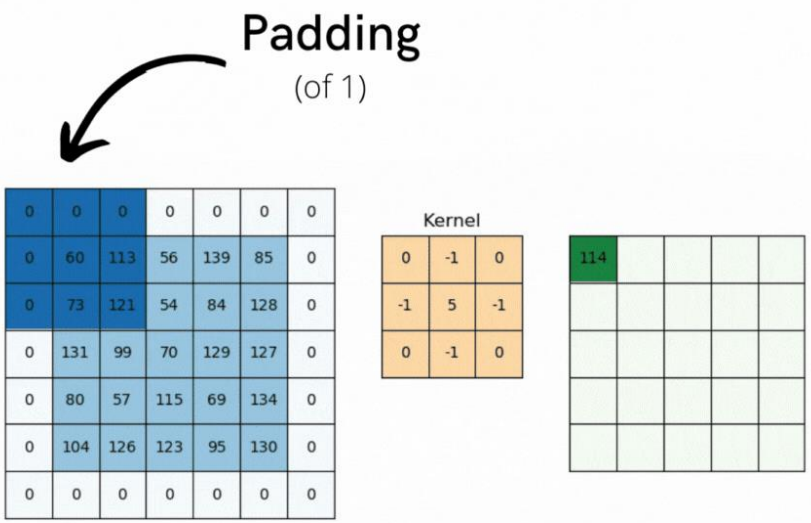
In the above figure , we are performing convolution of the original boat image using three distinct filters, thus producing three different feature maps as shown.

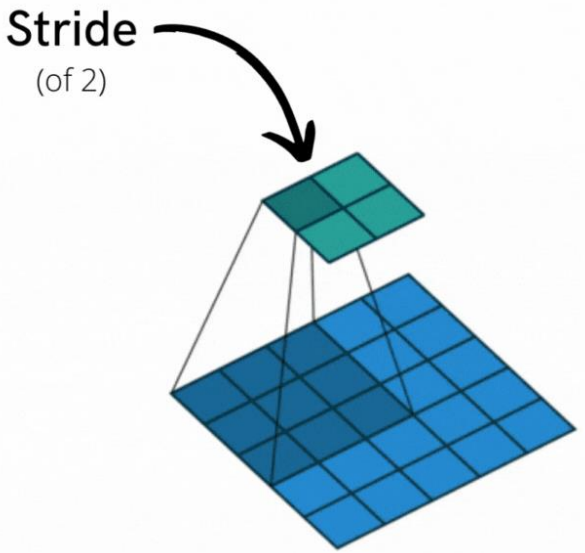
- You can think of these three feature maps as stacked 2d matrices, so, the 'depth' of the feature map would be three.

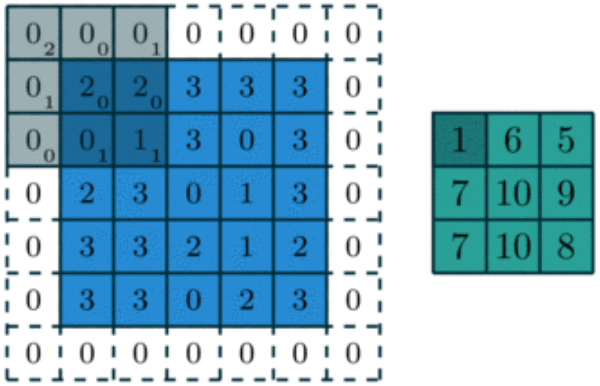
CONVOLUTIONAL NEURAL NETWORK ELABORATION

- **Stride** is the number of pixels by which we slide our filter matrix over the input matrix.
  - When the stride is 1 then we move the filters one pixel at a time.
  - When the stride is 2, then the filters jump 2 pixels at a time as we slide them around.
    - Having a larger stride will produce smaller feature maps.
- **Zero-padding:** Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix.

CONVOLUTIONAL NEURAL NETWORK ELABORATION

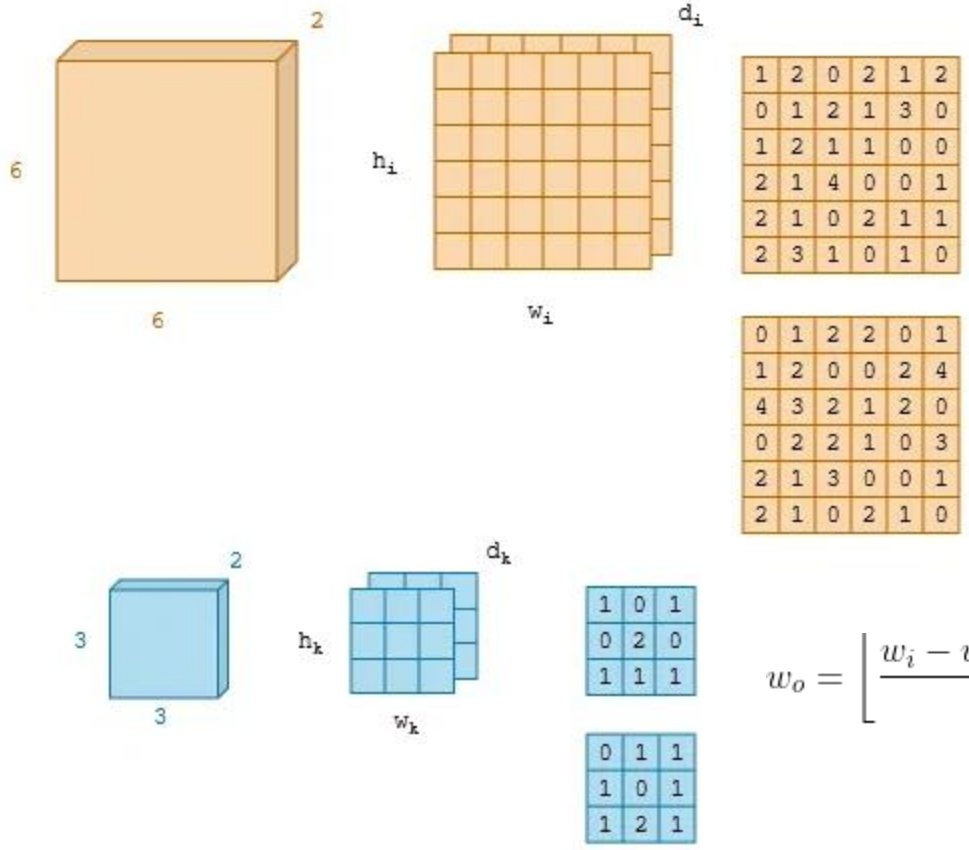






1	6	5
7	10	9
7	10	8

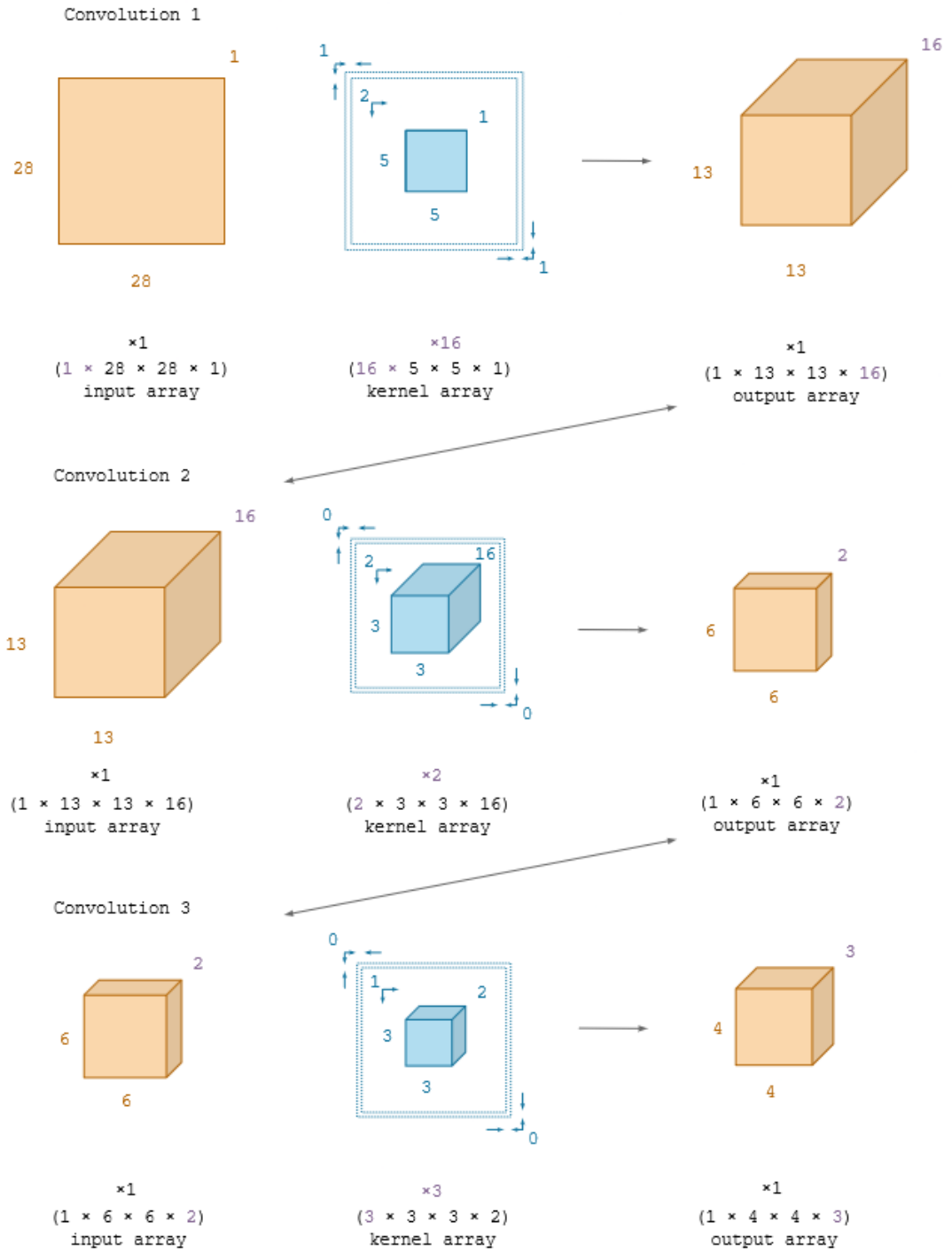
SIZE OF OUTPUT IN TERMS OF INPUTS



$$w_o = \left\lfloor \frac{w_i - w_k + 2 \cdot p_w}{s_w} \right\rfloor + 1 \; ; \; h_o = \left\lfloor \frac{h_i - h_k + 2 \cdot p_h}{s_h} \right\rfloor + 1$$

$$o = \left\lfloor \frac{6 - 3 + 2 \cdot 0}{2} \right\rfloor + 1 = \left\lfloor \frac{3}{2} \right\rfloor + 1 = 2$$

EXAMPLE

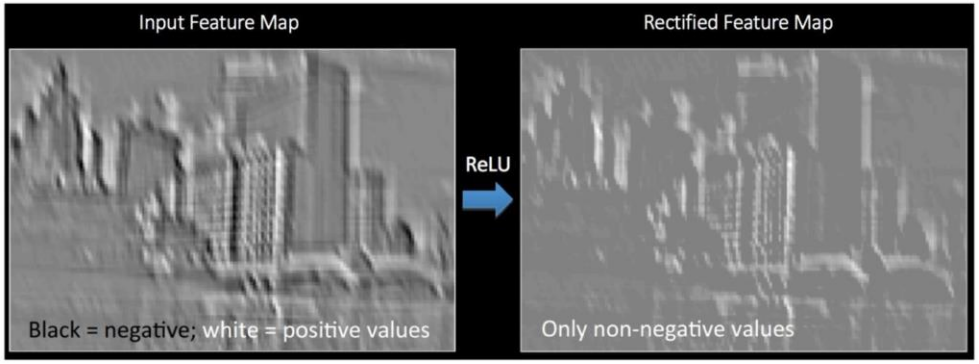

Convolution 1
1
28
28
×1
(1 × 28 × 28 × 1)
input array
1
2
1
5
5
1
×16
(16 × 5 × 5 × 1)
kernel array
16
13
13
×1
(1 × 13 × 13 × 16)
output array
Convolution 2
16
13
13
×1
(1 × 13 × 13 × 16)
input array
0
2
16
3
3
0
×2
(2 × 3 × 3 × 16)
kernel array
2
6
6
×1
(1 × 6 × 6 × 2)
output array
Convolution 3
2
6
6
×1
(1 × 6 × 6 × 2)
input array
0
1
2
3
3
0
×3
(3 × 3 × 3 × 2)
kernel array
3
4
4
×1
(1 × 4 × 4 × 3)
output array

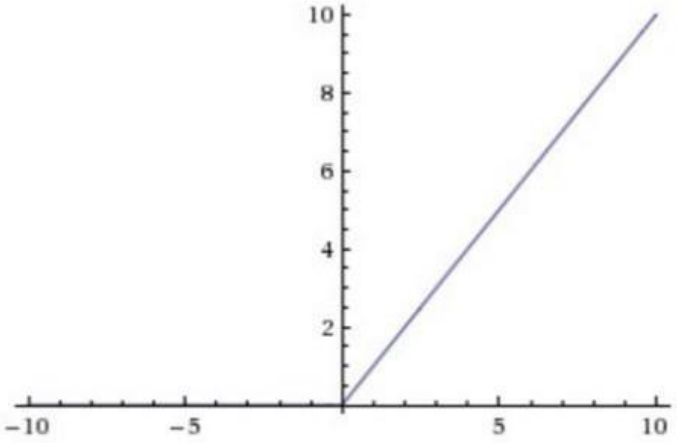
CONVOLUTIONAL NEURAL NETWORK ELABORATION

- CNN mainly used for image recognition and classification.
- There are four operations in CNN
  - Convolution
  - Non-Linearity (ReLU) activation function
  - Subsampling or Pooling
  - Classification (fully connected layer)

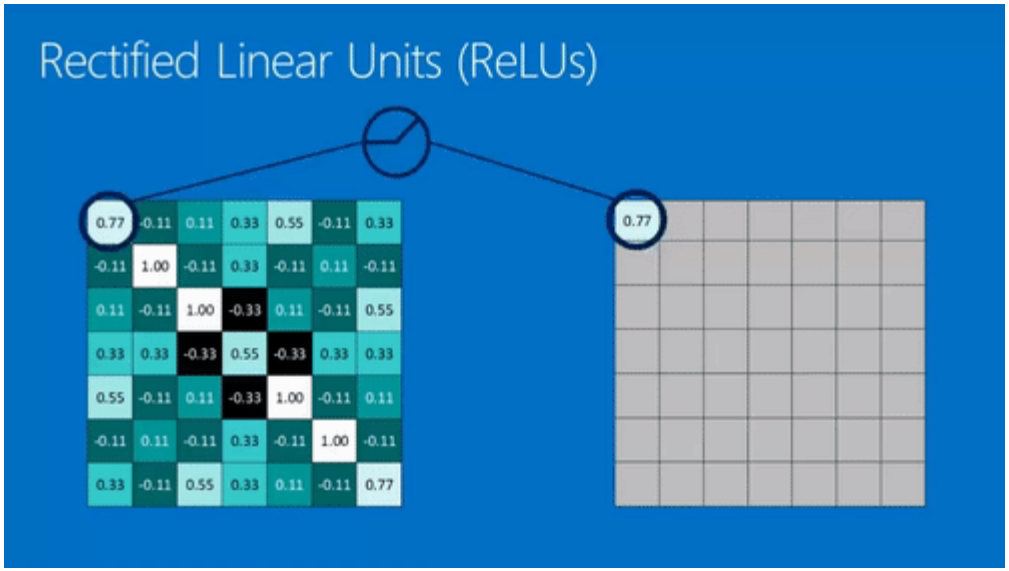
CONVOLUTIONAL NEURAL NETWORK ELABORATION

- Non-Linearity (ReLU)
  - An additional operation called ReLU has been used after every Convolution operation
  - ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity
  - There are non-linear functions such as **tanh** or **sigmoid** can also be used instead of ReLU, but ReLU has been found to perform better in most situations.





RELU

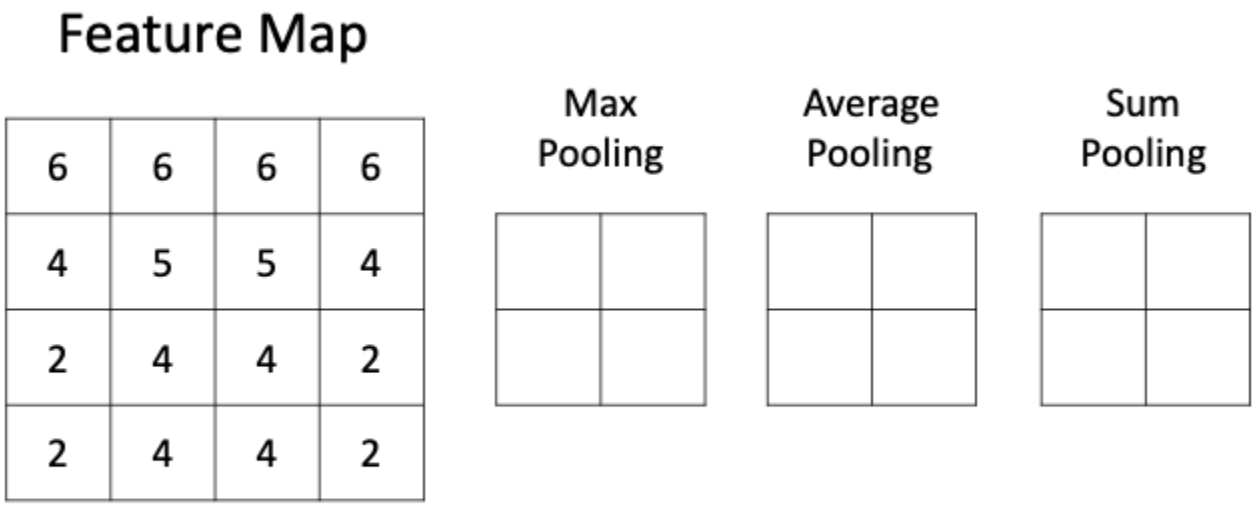

Rectified Linear Units (ReLUs)
0.77 -0.11 0.11 0.33 0.55 -0.11 0.33
-0.11 1.00 -0.11 0.33 -0.11 0.11 -0.11
0.11 -0.11 1.00 -0.33 0.11 -0.11 0.55
0.33 0.33 -0.33 0.55 -0.33 0.33 0.33
0.55 -0.11 0.11 -0.33 1.00 -0.11 0.11
-0.11 0.11 -0.11 0.33 -0.11 1.00 -0.11
0.33 -0.11 0.55 0.33 0.11 -0.11 0.77
0.77

CONVOLUTIONAL NEURAL NETWORK ELABORATION

- CNN mainly used for image recognition and classification.
- There are four operations in CNN
  - Convolution
  - Non-Linearity (ReLU) activation function
  - Subsampling or Pooling
  - Classification (fully connected layer)

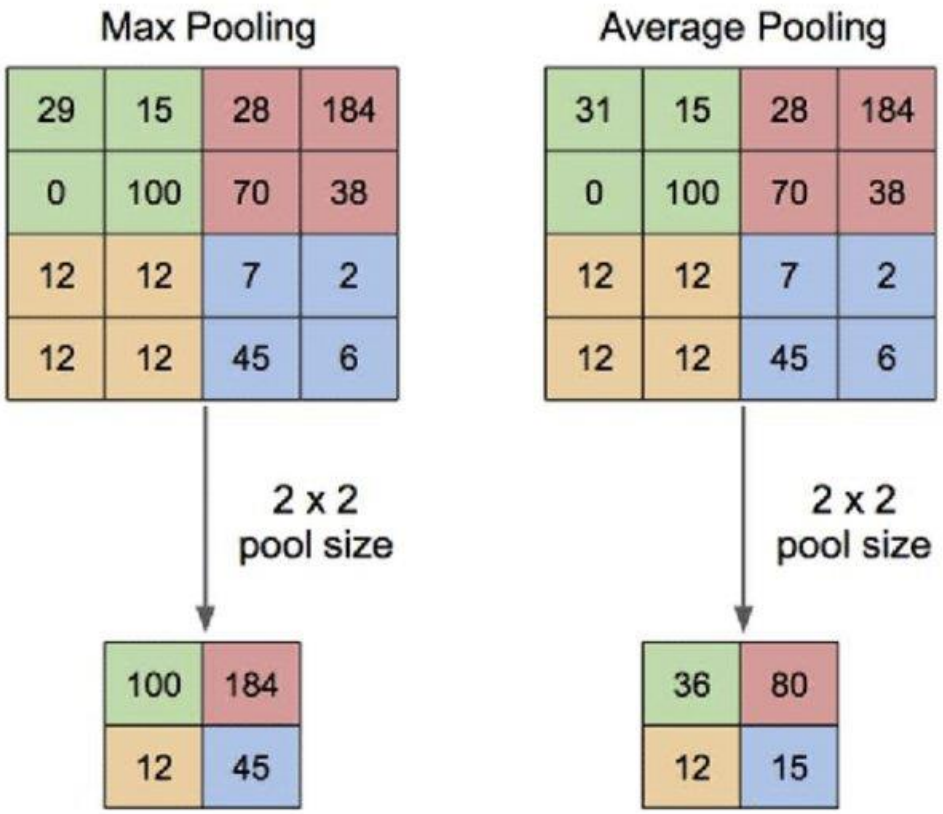
CONVOLUTIONAL NEURAL NETWORK ELABORATION

- Subsampling
  - Spatial Pooling (also called subsampling or down-sampling) reduces the dimensionality of each feature map but retains the most important information.
  - Spatial Pooling can be of different types: Max, Average, Sum etc.



CONVOLUTIONAL NEURAL NETWORK ELABORATION

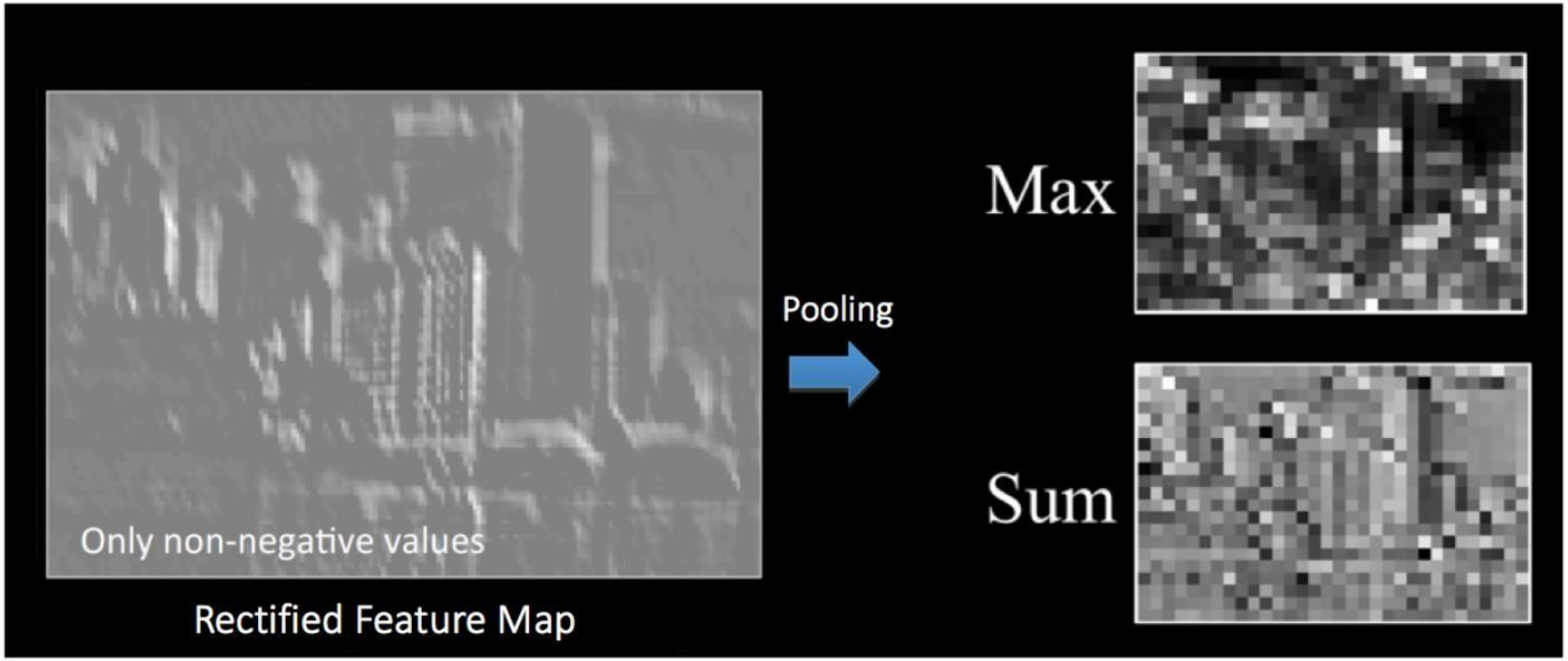
- The below figure shows an example of Max Pool and Average Pool on a rectified image (obtained after convolution + ReLU operation) by using a 2×2 window and stride = 2.



We slide our 2 x 2 window by 2 cells (also called 'stride') and take the maximum/average value in each region. As shown in **the figure** , this reduces the dimensionality of our feature map.

CONVOLUTIONAL NEURAL NETWORK ELABORATION

- The below figure shows the effect of Subsampling on the Rectified Feature Map we received after the ReLU operation



CONVOLUTIONAL NEURAL NETWORK ELABORATION

- The function of Subsampling is to progressively reduce the spatial size of the input representation. In particular, subsampling
  - makes the input representations (feature dimension) smaller and more manageable and reduces the number of parameters and computations in the network, therefore, controlling overfitting.
  - makes the network invariant to small transformations, distortions and translations in the input image (a small distortion in input will not change the output of Pooling – since we take the maximum / average value in a local neighbourhood).
  - helps us arrive at an almost scale invariant representation of our image (the exact term is “equivariant”). This is very powerful since we can detect objects in an image no matter where they are located.

Convolution

- Connections sparsity reduces overfitting
- Conv + Pooling gives location invariant feature detection
- Parameter sharing

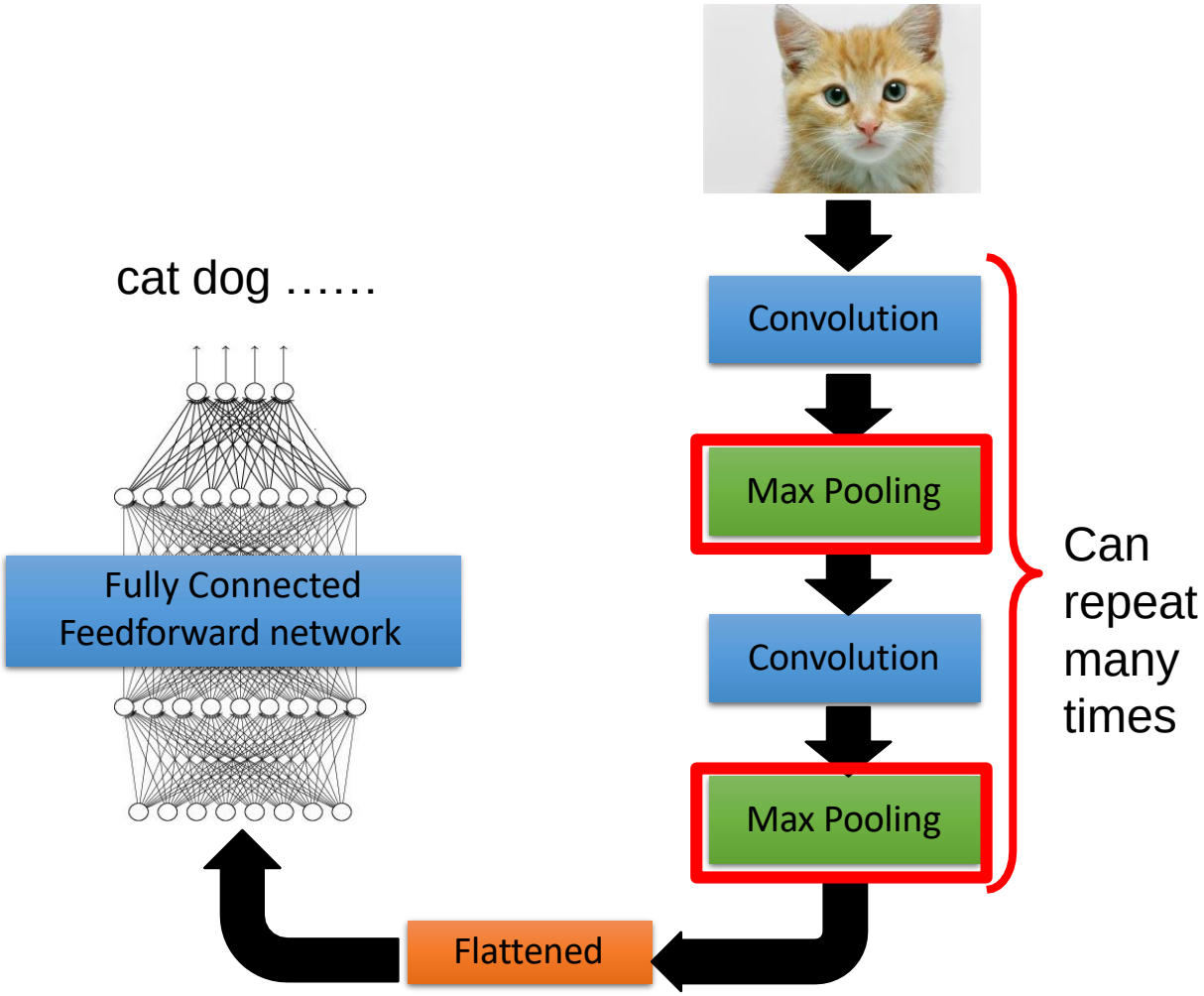
ReLU

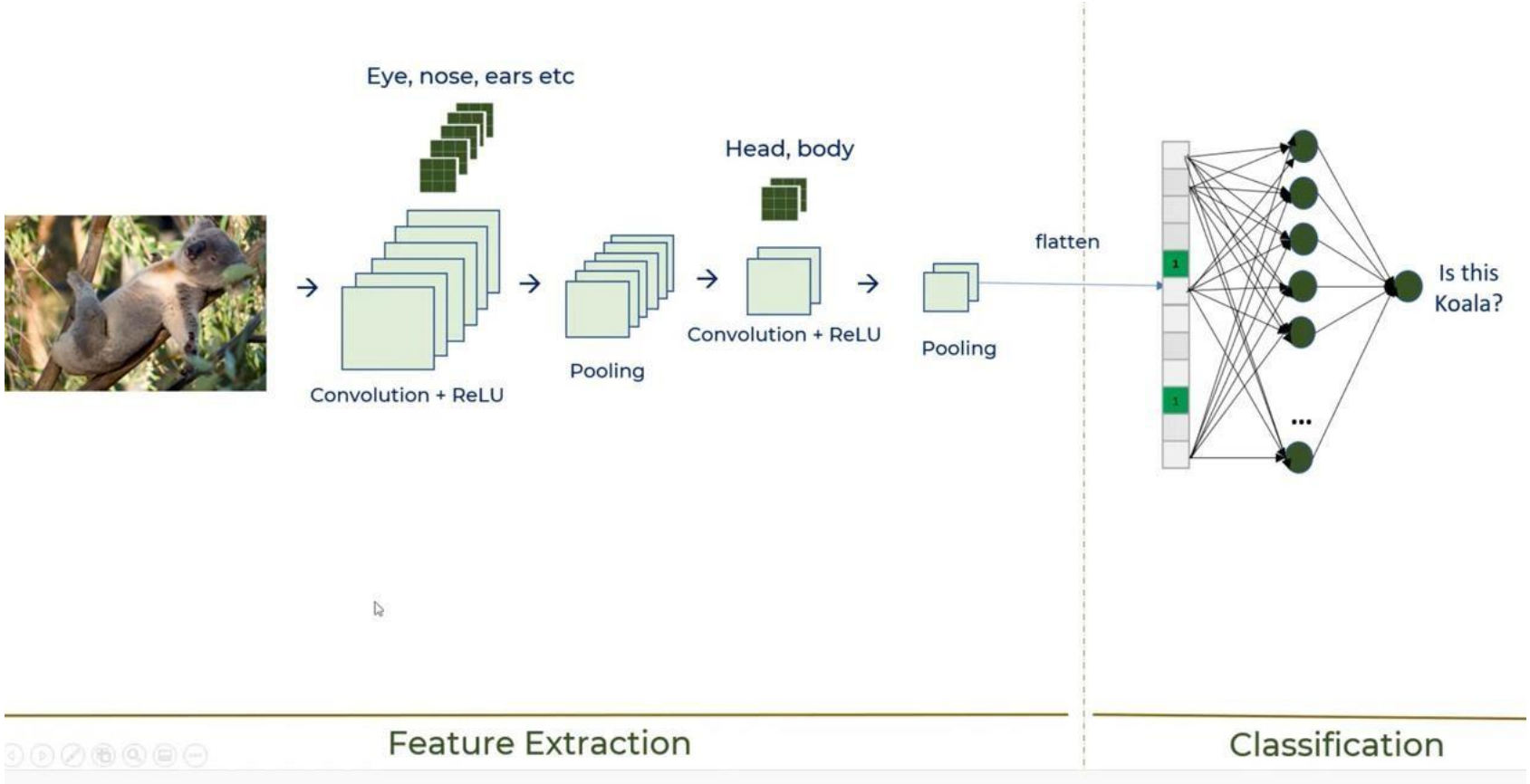
- Introduces nonlinearity
- Speeds up training, faster to compute

Pooling

- Reduces dimensions and computation
- Reduces overfitting
- Makes the model tolerant towards small distortion and variations

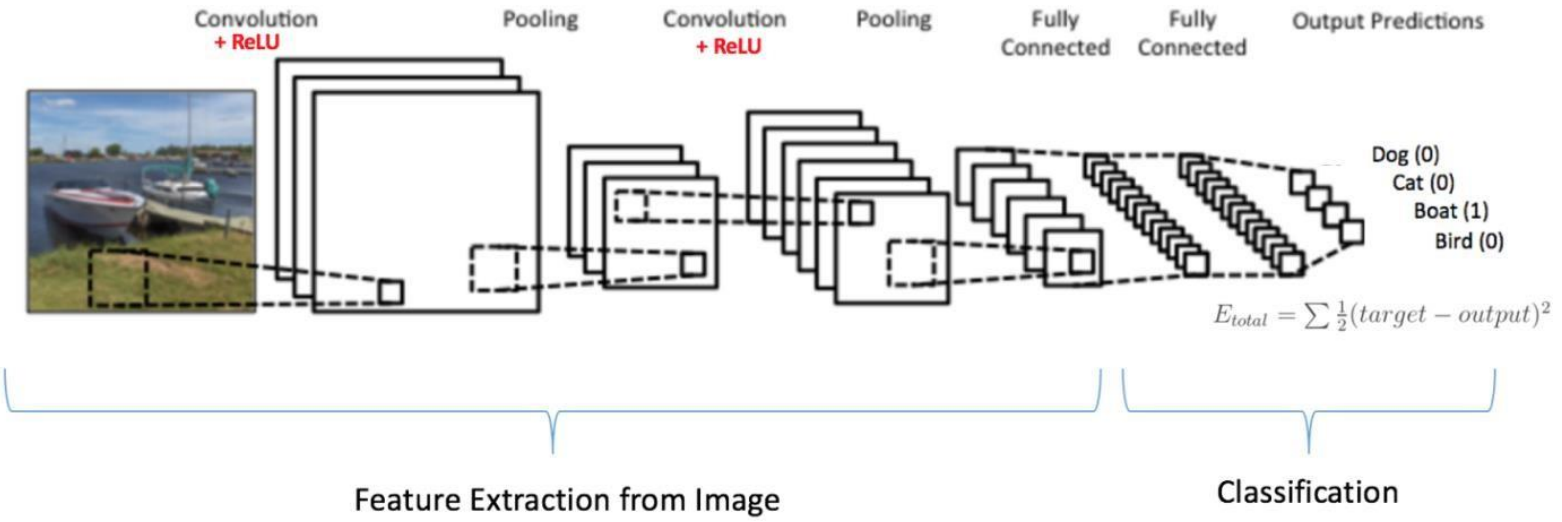
THE WHOLE CNN




Eye, nose, ears etc
Head, body
Convolution + ReLU
Pooling
Convolution + ReLU
Pooling
flatten
Is this Koala?
Feature Extraction
Classification

CONVOLUTIONAL NEURAL NETWORK ELABORATION

- Putting it all together

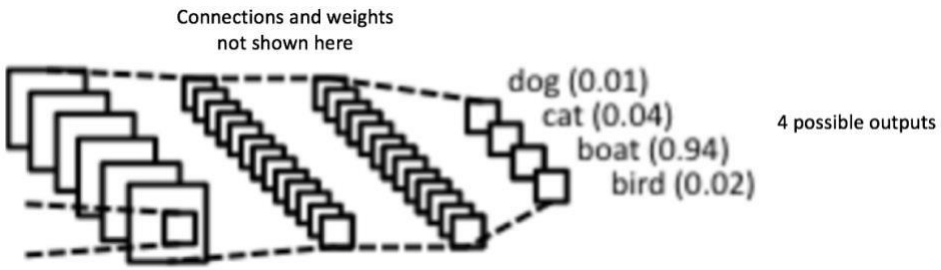


CONVOLUTIONAL NEURAL NETWORK ELABORATION

- CNN is mainly used for image recognition and classification.
- There are four operations in CNN
  - Convolution
  - Non-Linearity (ReLU) activation function
  - Subsampling or Pooling
  - Classification (fully connected layer)

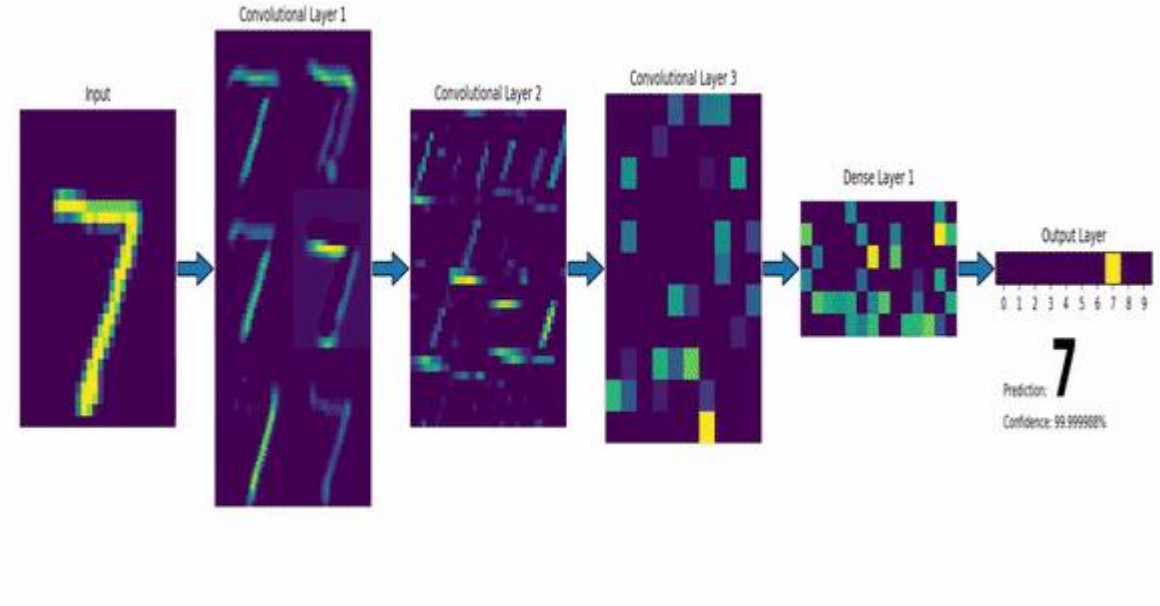
CONVOLUTIONAL NEURAL NETWORK ELABORATION

- The Fully Connected layer is a traditional MLP using SoftMax activation function
  - The SoftMax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one.
- The output from the convolutional and pooling layers represent high-level features of the input image.
- The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.



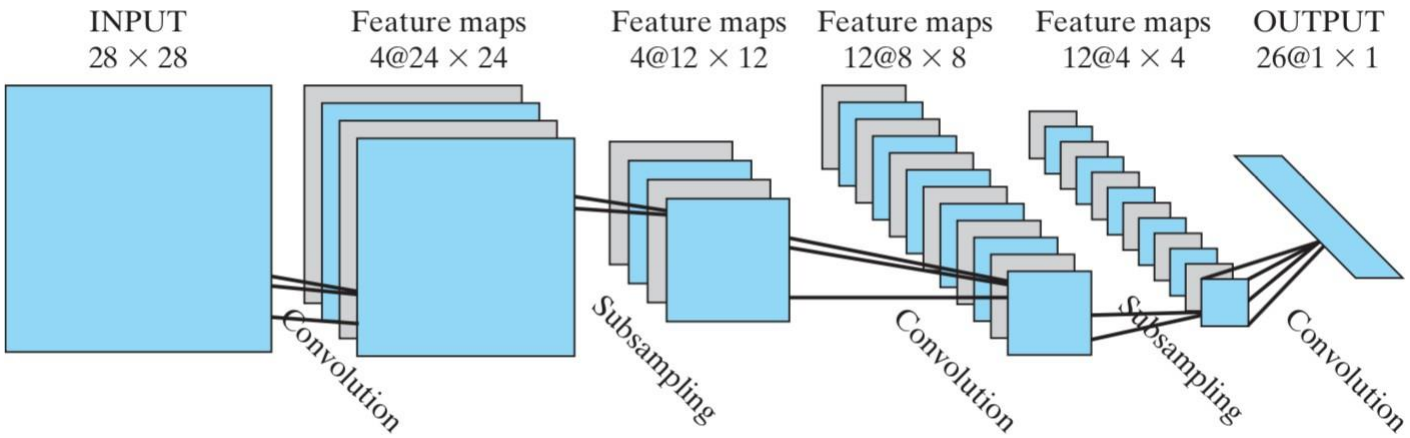
CONVOLUTION NETWORKS

- **Feature extraction:**
  - Each neuron takes its synaptic inputs from a local *receptive field* in the previous layer, thereby forcing it to extract local features.
  - Once a feature has been extracted, its exact location becomes less important, as long as its position relative to other features is approximately preserved.
- **Feature mapping**
  - Each computational layer of the network is composed of multiple *feature maps*
    - *Shift invariance*, forced into the operation of a feature map through the use of *convolution* with a kernel of small size, followed by a nonlinear function (sigmoid, RELU, …);
    - *Reduction in the number of parameters*, accomplished through the use of
      - *weight sharing.*


Input
Convolutional Layer 1
Convolutional Layer 2
Convolutional Layer 3
Dense Layer 1
Output Layer
0 1 2 3 4 5 6 7 8 9
Prediction:
7
Confidence: 99.999988%

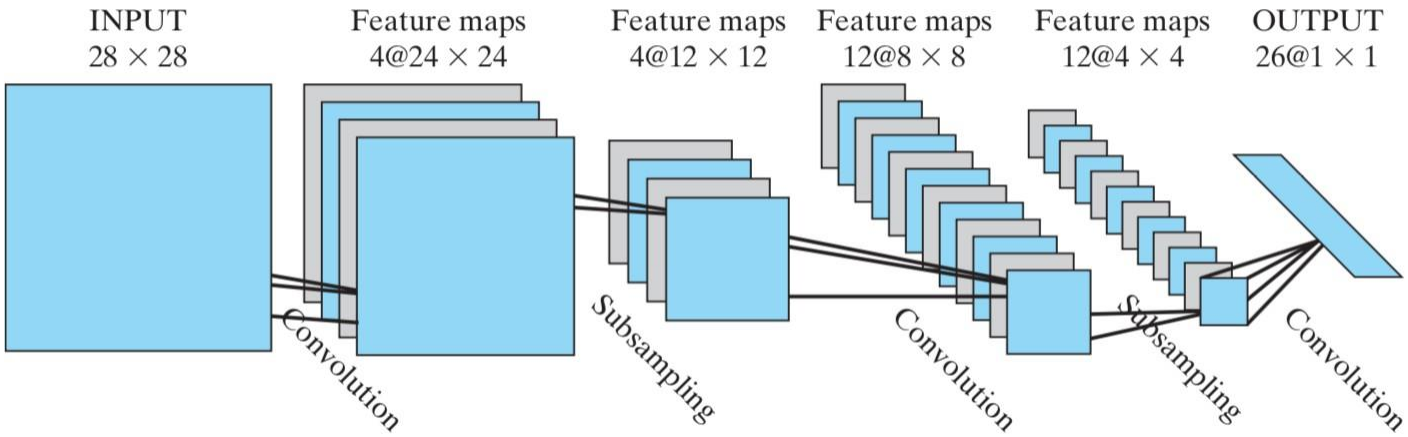
CNN EXAMPLE

- The below convolution network is made up of four hidden layers and an output layer.
- The input layer is 28 x 28 sensory nodes receiving images of different characters that have been approximately centred and normalized



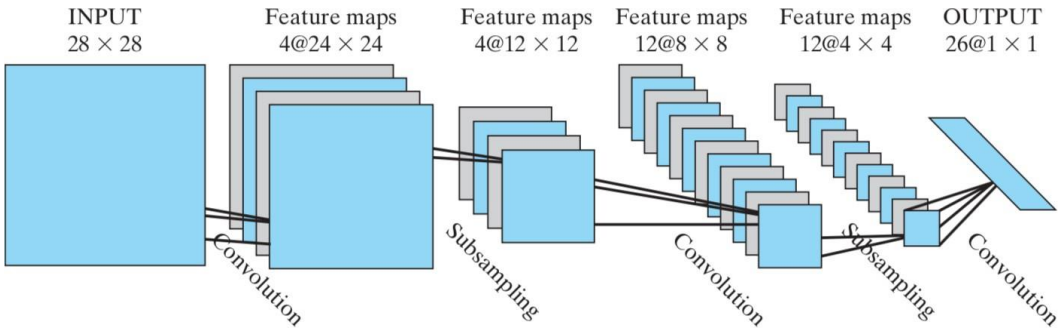
CNN EXAMPLE

- The first hidden layer performs convolution. It consists of four feature maps, with each feature map consisting of 24 x24 neurons.
  - Each neuron is assigned a receptive field of size 5 x 5.
- The second hidden layer performs subsampling and local averaging. It also consists of four feature maps, but each feature map is now made up of 12 x 12 neurons.



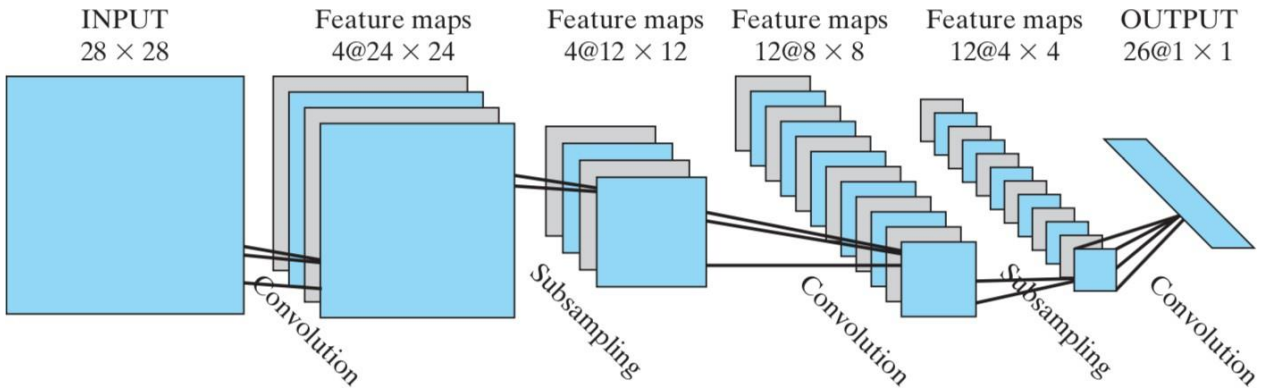
CNN EXAMPLE

- The third hidden layer performs a second convolution.
  - It consists of 12 feature maps, with each feature map consisting of 8 x 8 neurons.
  - Each neuron in this hidden layer may have synaptic connections from several feature maps in the previous hidden layer.
  - Otherwise, it operates in a manner similar to the first convolutional layer.
- The fourth hidden layer performs a second subsampling and local averaging.
  - It consists of 12 feature maps, but with each feature map consisting of 4 x 4 neurons.
  - Otherwise, it operates in a manner similar to the first subsampling layer.
- The output layer performs one final stage of convolution.
  - It consists of 26 neurons, with each neuron assigned to one of 26 possible characters.
  - As before, each neuron is assigned a receptive field of size 4 x 4.



CNN EXAMPLE

- The MLP shown below contains approximately 100,000 synaptic connection, but only 2600 synaptic weight
  - This is done by weight sharing
  - The capacity of the learning machine is thereby reduced, which in turn improves the machine's ability to generalize.
  - The adjustment of the parameters are done using stochastic mode of the BP algorithm.

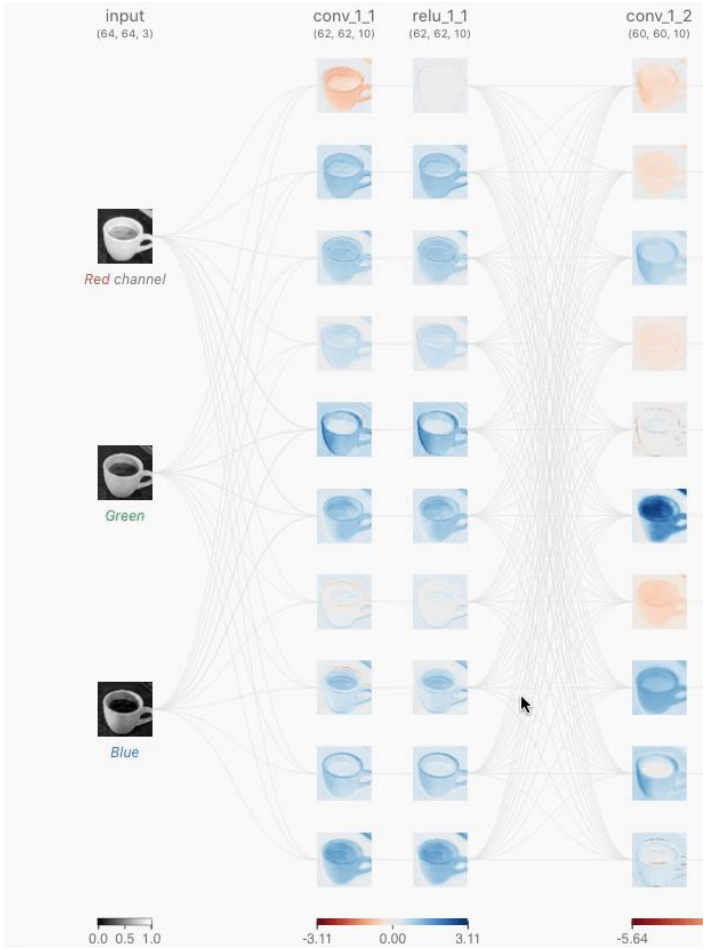


CONVOLUTIONAL NEURAL NETWORK TRAINING

- **Step1:** We initialize all filters and parameters / weights with random values
- **Step2:** The network takes a training image as input, goes through the forward propagation step (convolution, ReLU and subsampling operations along with forward propagation in the Fully Connected layer) and finds the output probabilities for each class.
  - Let's say the output probabilities are [0.2, 0.4, 0.1, 0.3]
  - Since weights are randomly assigned for the first training example, output probabilities are also random.
- **Step3:** Calculate the total error at the output layer (summation over all 4 classes)
  - **Total Error = $\sum$ ½ (target probability – output probability) ^{2}**

CONVOLUTIONAL NEURAL NETWORK TRAINING

- **Step4:** Use Backpropagation to calculate the *gradients* of the error with respect to all weights in the network and use *gradient descent* to update all filter values / weights and parameter values to minimize the output error.
  - The weights are adjusted in proportion to their contribution to the total error.
  - When the same image is input again, output probabilities might now be [0.1, 0.1, 0.7, 0.1], which is closer to the target vector [0, 0, 1, 0].
  - This means that the network has learnt to classify this particular image correctly by adjusting its weights / filters such that the output error is reduced.
  - Parameters like number of filters, filter sizes, architecture of the network etc. have all been fixed before Step 1 and do not change during training process – only the values of the filter matrix and connection weights get updated.
- **Step5:** Repeat steps 2-4 with all images in the training set.

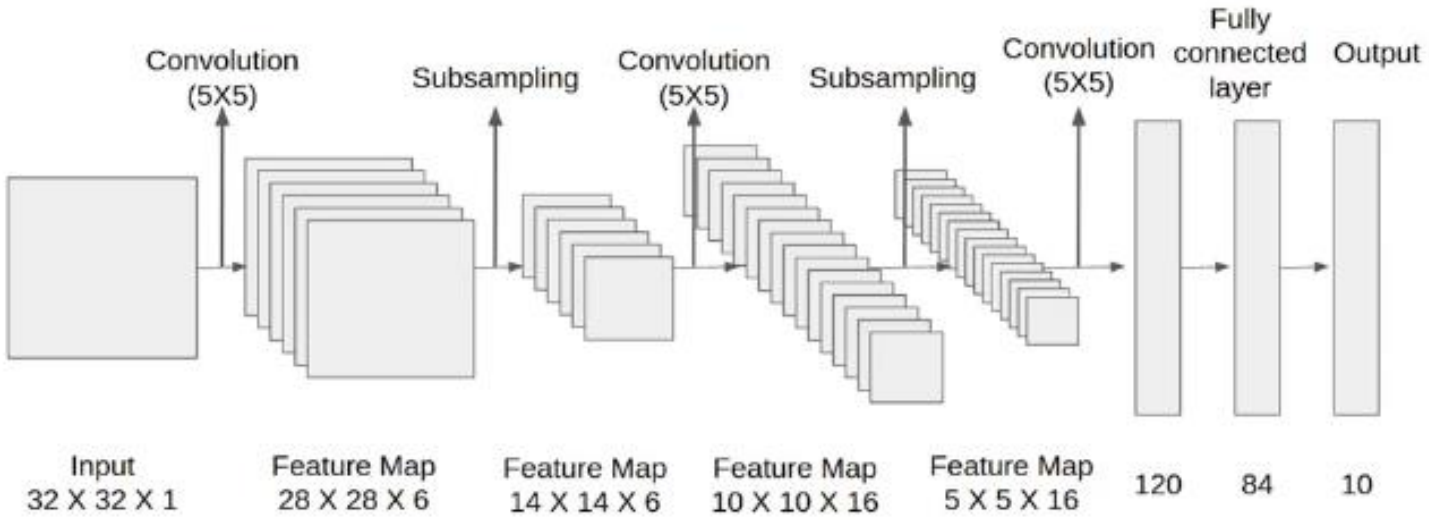

input
(64, 64, 3)
conv_1_1
(62, 62, 10)
relu_1_1
(62, 62, 10)
conv_1_2
(60, 60, 10)
Red channel
Green
Blue
0.0 0.5 1.0
-3.11 0.00 3.11
-5.64

WELL-KNOWN CNN ARCHITECTURES (1)

- **Among the well-known CNN architectures used today:**
- **LeNet**
- **AlexNet**
- **VGG**
- **GoogLeNet**
- **ResNet**
- **DenseNet**
- **MobileNet**

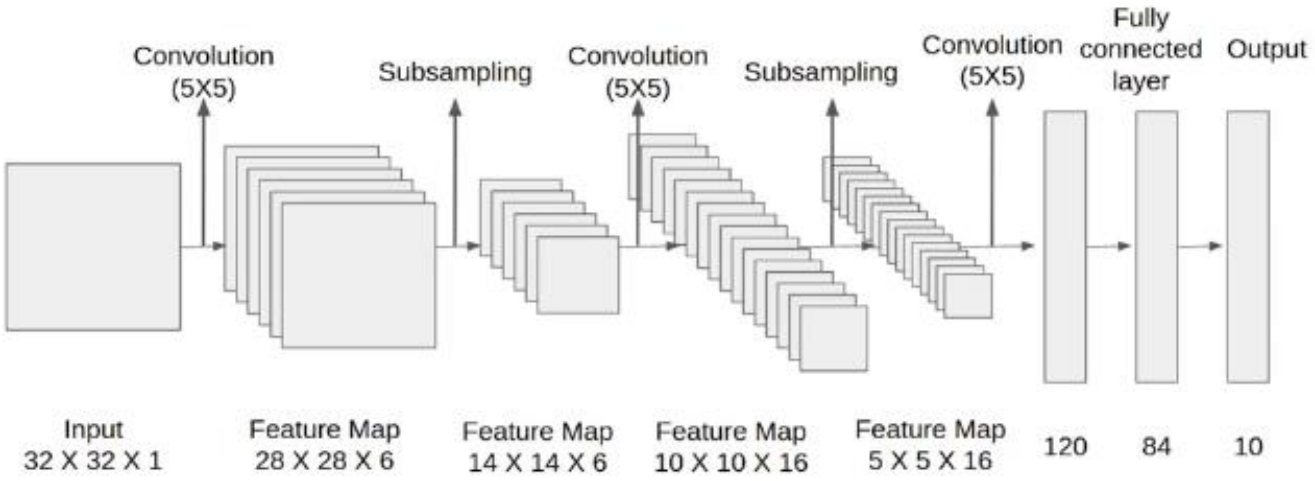
LENET

- LeNet → one of the earliest CNN architectures,
- Developed in the 1998 by Yann LeCun et al.
- Several versions → LeNet-1, LeNet-3, LeNet-4, and LeNet-5
- Designed to recognize handwritten digits and was used for automated check reading.



LENET5

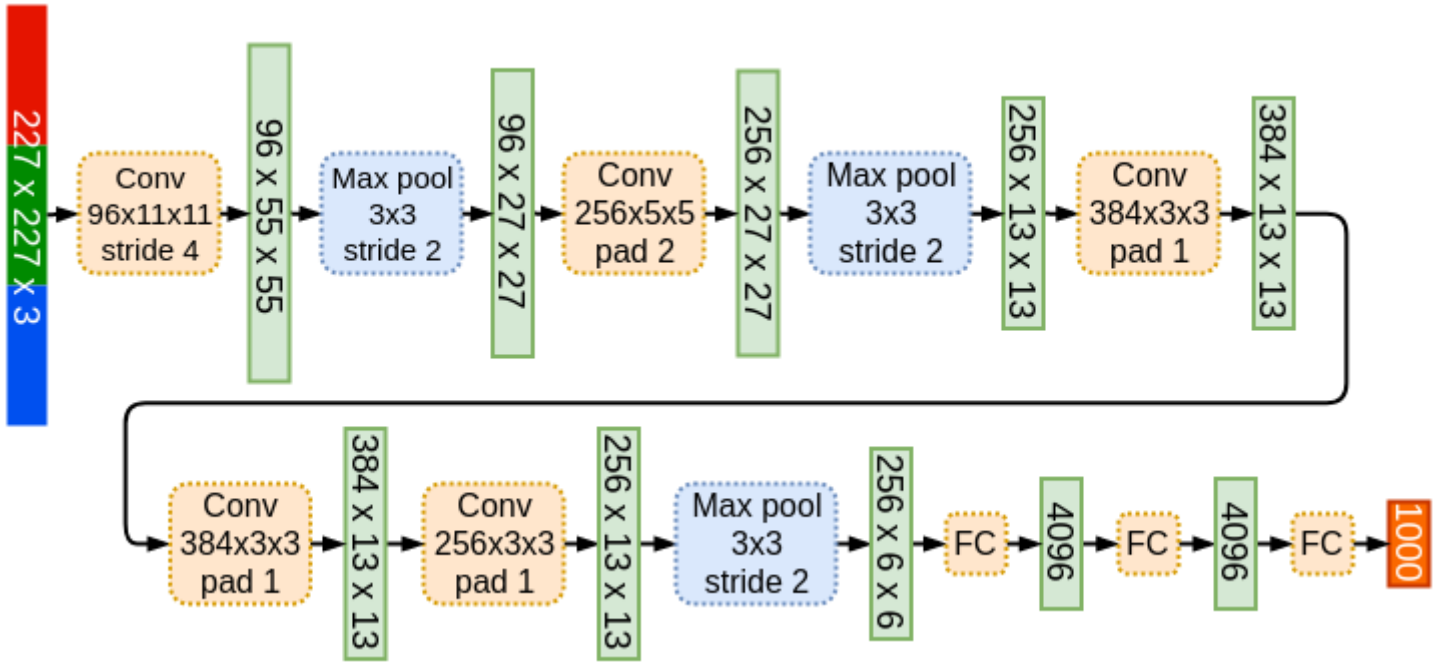
- LeNet5



Layer	# filters / neurons	Filter size	Stride	Size of feature map	Activation function
Input	-	-	-	32 X 32 X 1	
Conv 1	6	5 * 5	1	28 X 28 X 6	tanh
Avg. pooling 1		2 * 2	2	14 X 14 X 6	
Conv 2	16	5 * 5	1	10 X 10 X 16	tanh
Avg. pooling 2		2 * 2	2	5 X 5 X 16	
Conv 3	120	5 * 5	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax

ALEXNET

- AlexNet was the breakthrough architecture that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012.
- It had a deeper architecture than previous CNNs and used ReLU activation functions to speed up training.



ALEXNET

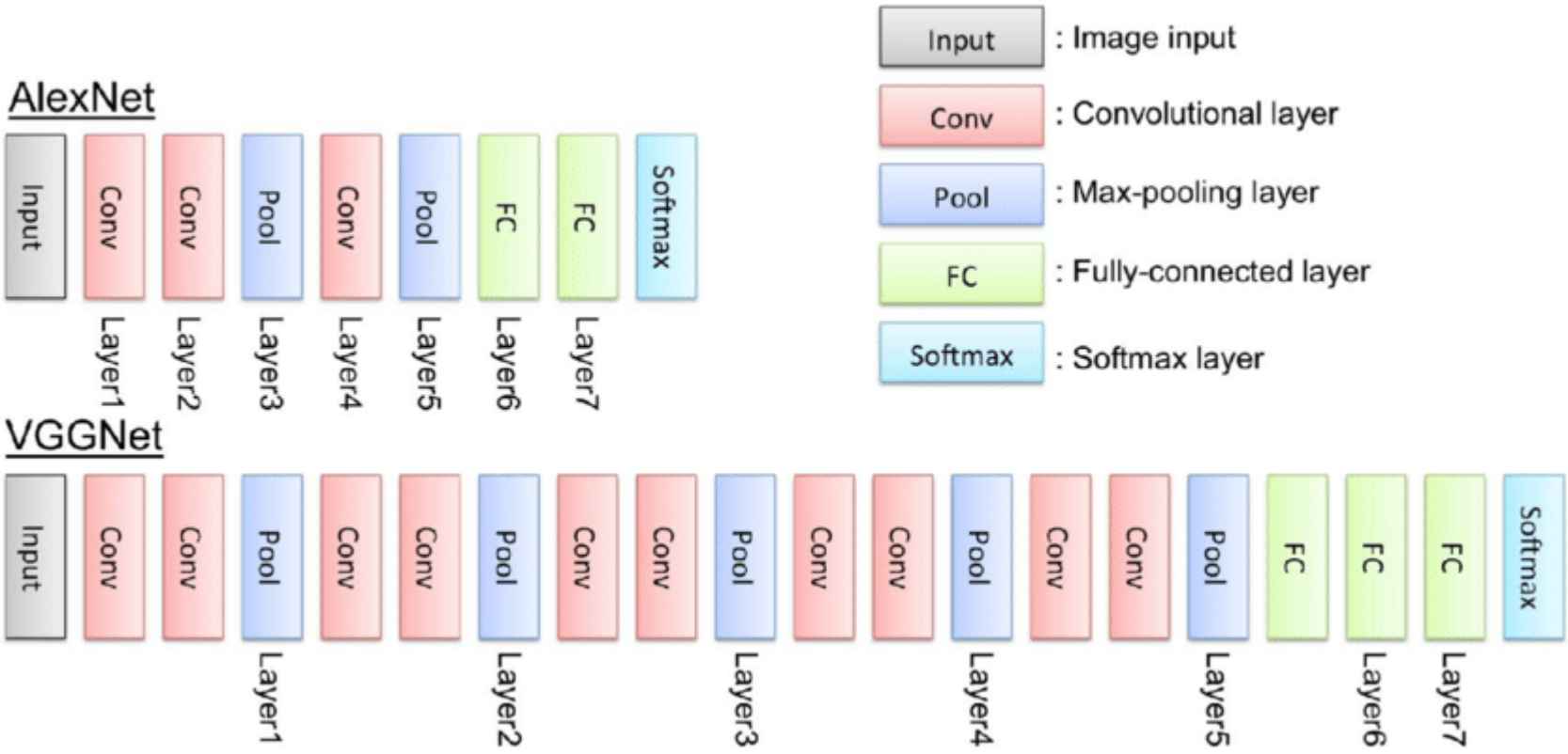
Architecture: AlexNet

Layer	# filters / neurons	Filter size	Stride	Padding	Size of feature map	Activation function
Input	-	-	-	-	227 x 227 x 3	-
Conv 1	96	11 x 11	4	-	55 x 55 x 96	ReLU
Max Pool 1	-	3 x 3	2	-	27 x 27 x 96	-
Conv 2	256	5 x 5	1	2	27 x 27 x 256	ReLU
Max Pool 2	-	3 x 3	2	-	13 x 13 x 256	-
Conv 3	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 4	384	3 x 3	1	1	13 x 13 x 384	ReLU
Conv 5	256	3 x 3	1	1	13 x 13 x 256	ReLU

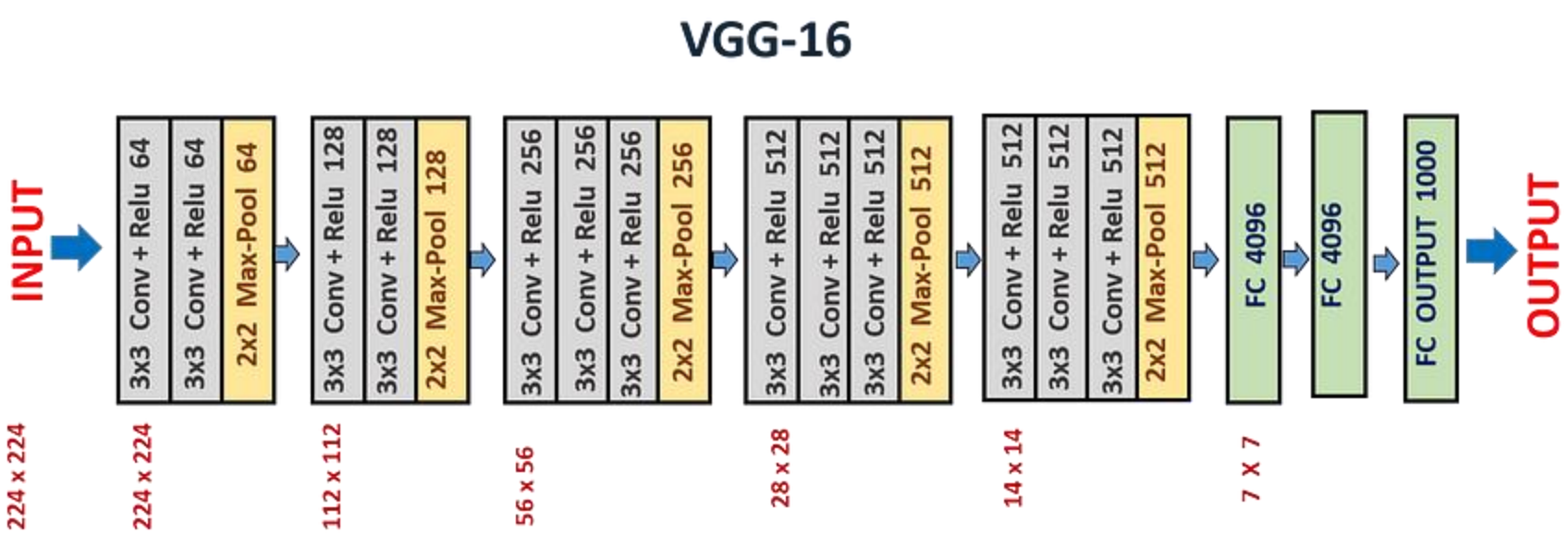
VGG

- VGG (Visual Geometry Group) was a group of CNN architectures developed by researchers at the University of Oxford.
- Simple and uses 3x3 convolutional filters throughout the network.
- Several architectures → VGG16 (16 layers), VGG19 (19 layers)

VGG VS ALEXNET



VGG16


VGG-16
INPUT
224 x 224
224 x 224
3x3 Conv + Relu 64
3x3 Conv + Relu 64
2x2 Max-Pool 64
112 x 112
3x3 Conv + Relu 128
3x3 Conv + Relu 128
2x2 Max-Pool 128
56 x 56
3x3 Conv + Relu 256
3x3 Conv + Relu 256
3x3 Conv + Relu 256
2x2 Max-Pool 256
28 x 28
3x3 Conv + Relu 512
3x3 Conv + Relu 512
3x3 Conv + Relu 512
2x2 Max-Pool 512
14 x 14
3x3 Conv + Relu 512
3x3 Conv + Relu 512
3x3 Conv + Relu 512
2x2 Max-Pool 512
7 X 7
FC 4096
FC 4096
FC OUTPUT 1000
OUTPUT

VGG16

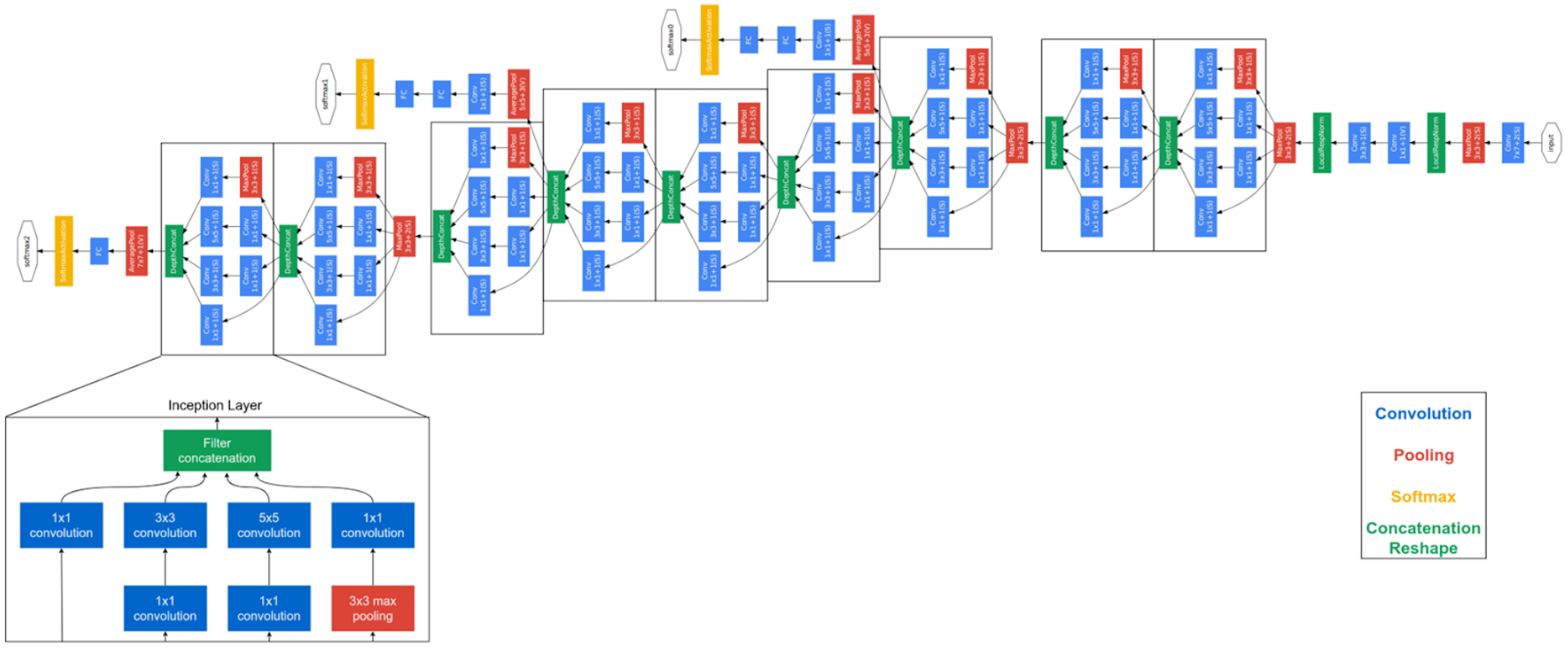
- VGG (Visual Geometry Group) was a group of CNN architectures developed by researchers at the University of Oxford.
- Simple and uses 3x3 convolutional filters throughout the network.

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

GOOGLENET

- GoogLeNet, also known as Inception, was developed by Google researchers in 2014.
- Researchers believed that increasing the depth of neural networks would lead to better performance.
  - However, this approach posed several challenges, such as vanishing gradients and increased computational cost.
- GoogLeNet team proposed solution to these problems → "inception modules,"
  - The inception module is designed to capture multi-scale features of the input image by performing convolutional operations with filters of different sizes (1x1, 3x3, 5x5) in parallel.
  - The idea behind the inception module is to allow the network to learn both local and global features by performing convolutions with filters of different sizes.
  - This architecture allows the network to learn both fine-grained and coarse-grained features.

GOOGLENET


Inception Layer
Filter concatenation
1x1 convolution
3x3 convolution
5x5 convolution
1x1 convolution
1x1 convolution
1x1 convolution
3x3 max pooling
Convolution
Pooling
Softmax
Concatenation
Reshape

GOOGLENET

- GoogLeNet, also known as Inception, was developed by Google researchers in 2014.
- It introduced the idea of using multiple filter sizes in parallel to capture features at different scales.

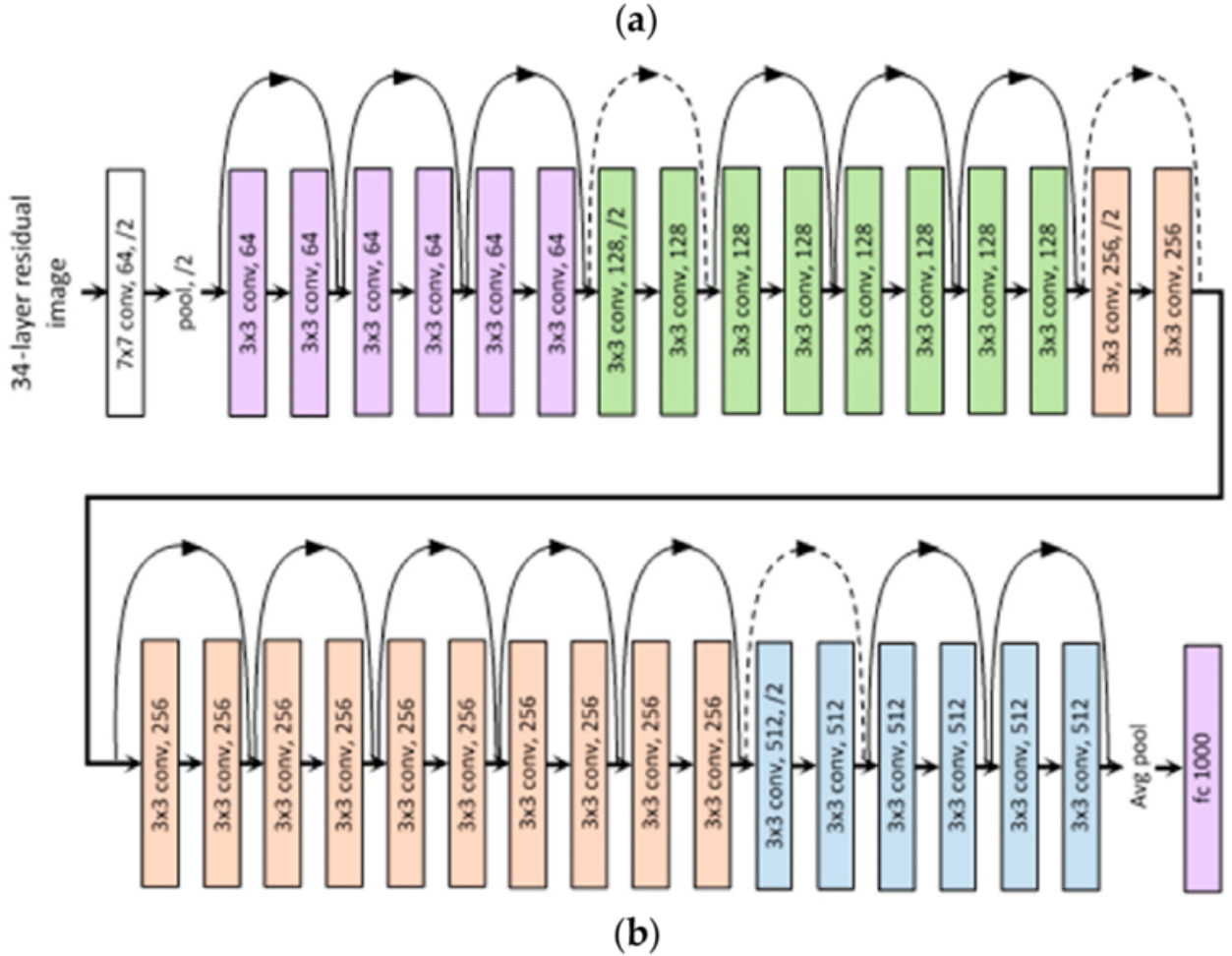
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

RESNET

- ResNet (Residual Network) developed by researchers at Microsoft in 2015.
- It used residual connections to help train very deep networks, allowing for more accurate classification.
  - a residual refers to the difference between the output of a layer and its input.
  - ResNet solves the problem of vanishing gradient by introducing skip connections that allow the input to be directly passed to a later layer in the network. This is achieved by adding the input to the output of a layer, creating a residual block.

RESNET

- ResNet (Residual Network) developed by researchers at Microsoft in 2015.



DENSENET

- DenseNet was developed in 2016 and uses densely connected blocks of layers.
- In a dense block in DenseNet, each layer receives inputs not only from the previous layer but also directly from all preceding layers. The output feature maps of each layer are concatenated and passed as input to the next layer.
- This dense connectivity pattern reduces the number of parameters while preserving or even enhancing the flow of information through the network.

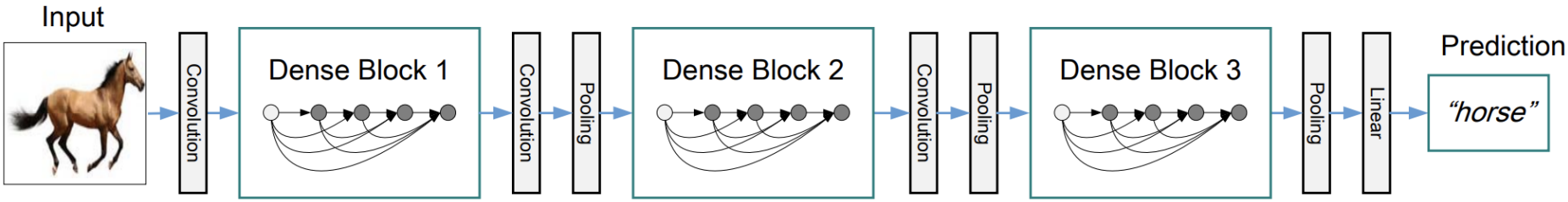
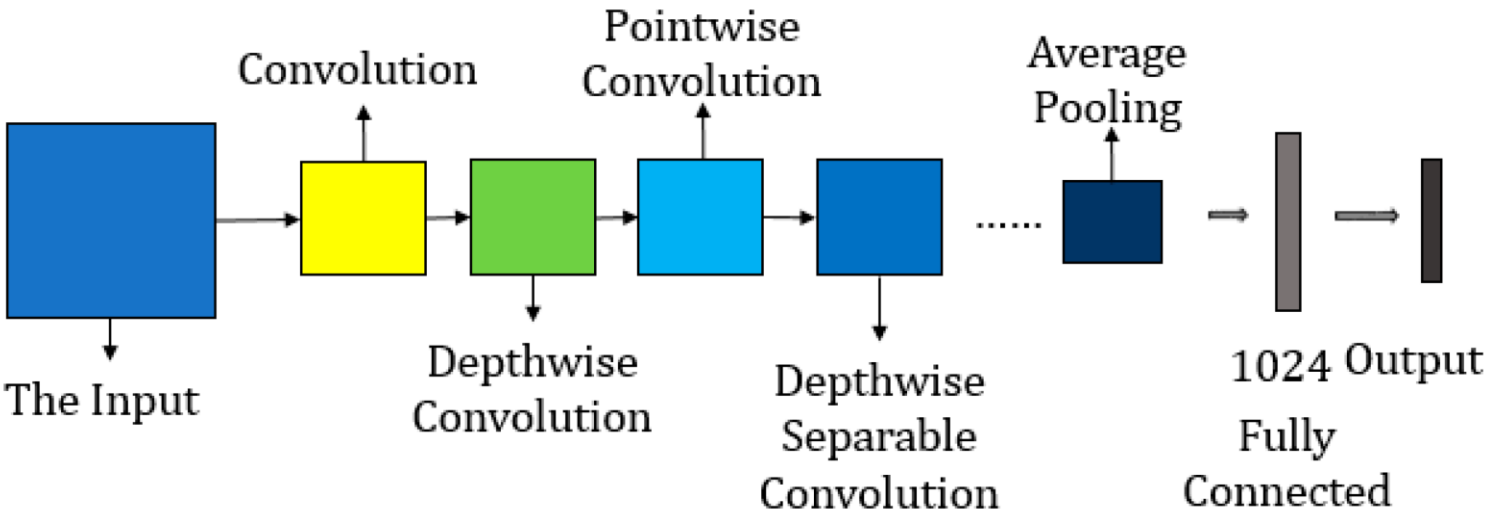


Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

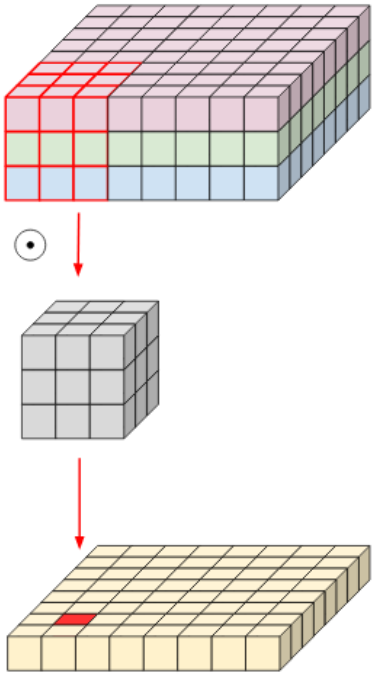
MOBILENET

- MobileNet was designed for mobile and embedded devices, with a focus on reducing the number of parameters and computational complexity while maintaining accuracy.
  - Depth-wise convolution
  - Point-wise convolution

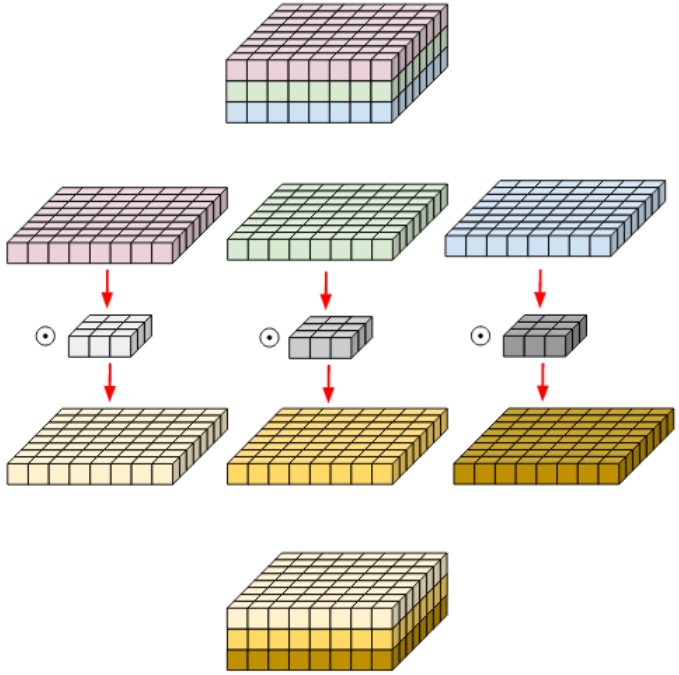


DEPTH-WISE AND POINT-WISE CONVOLUTION

- Standard 3d convolution
- Depth-wise → each filter on each channel separately



Standard convolution

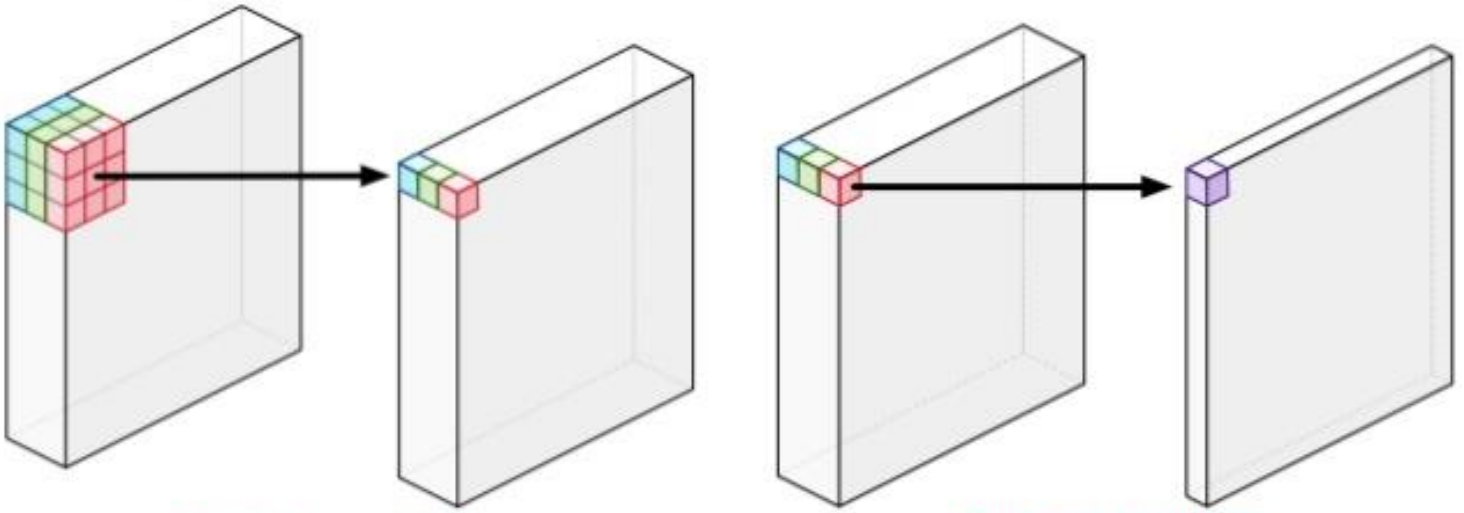


Depth-wise convolution

DEPTH-WISE AND POINT-WISE CONVOLUTION

Depthwise Separable Convolution

- Depthwise Convolution + Pointwise Convolution(1x1 convolution)



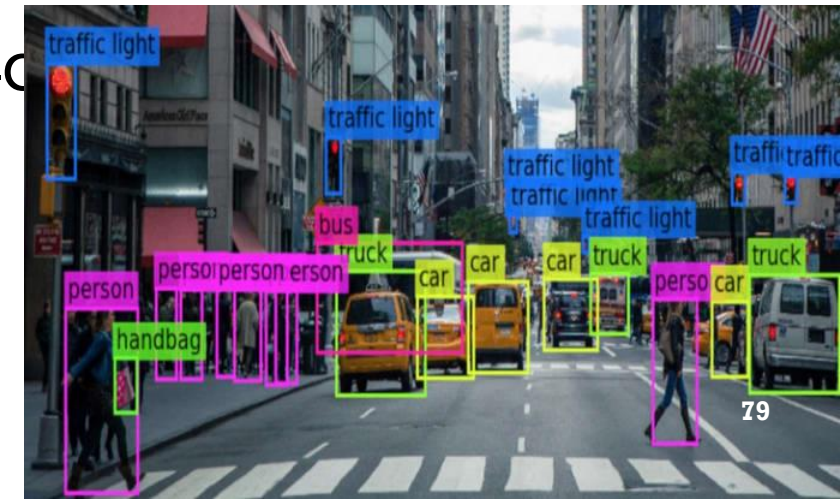
Depthwise convolution

Pointwise convolution

Figures from http://machinethink.net/blog/googles-mobile-net-architecture-on-iphone/

ARCHITECTURES VS APPLICATIONS

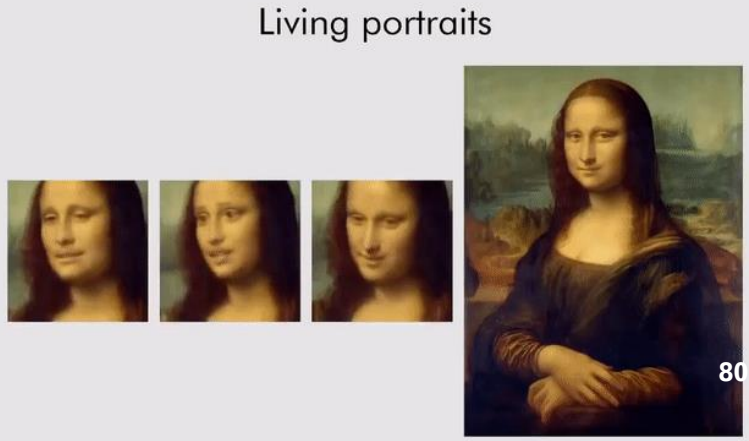
- **Image classification**
  - ✓AlexNet
  - ✓VGG (e.g., VGG16, VGG19)
  - ✓GoogLeNet (also known as Inception)
  - ✓ResNet (e.g., ResNet50, ResNet101)
  - ✓DenseNet
  - ✓MobileNet
- **Object detection**
  - ✓R-CNN (e.g., Fast R-CNN, Faster R-C
  - ✓YOLO (You Only Look Once)
  - ✓SSD (Single Shot Detector)
  - ✓RetinaNet



ARCHITECTURES VS APPLICATIONS

- **Semantic segmentation**
  - ✓FCN (Fully Convolutional Network)
  - ✓U-Net
  - ✓SegNet
  - ✓DeepLab

- **Generative modeling**
  - ✓GAN (Generative Adversarial Network)
  - ✓VAE (Variational Autoencoder)
  - ✓PixelCNN



SOME GAN-BASED APPLICATIONS

- ✓ Here are some examples of the top 10 GAN-based applications:
- ✓ Image generation: GANs can generate high-quality images of various types, such as human faces, landscapes, and animals. This has applications in art, design, and advertising.
- ✓ Image-to-image translation: GANs can be used to translate an image from one domain to another, such as converting a daytime image to a nighttime image, or converting a sketch to a photorealistic image.
- ✓ Super-resolution: GANs can be used to generate high-resolution images from low-resolution images, which has applications in medical imaging, satellite imaging, and surveillance.
- ✓ Video synthesis: GANs can generate realistic videos by combining generated frames with real frames, which has applications in entertainment and virtual reality.
- ✓ 3D object generation: GANs can generate 3D objects from 2D images, which has applications in manufacturing, prototyping, and product design.
- ✓ Text-to-image synthesis: GANs can generate images from textual descriptions, which has applications in e-commerce, advertising, and visual storytelling.

SOME GAN-BASED APPLICATIONS

- ✓ Style transfer: GANs can be used to transfer the style of one image to another, which has applications in art and design.
- ✓ Data augmentation: GANs can be used to generate synthetic data to augment real data for training machine learning models, which can improve model accuracy.
- ✓ Image inpainting: GANs can fill in missing parts of an image based on the surrounding context, which has applications in photo editing and restoration.
- ✓ Image manipulation: GANs can be used to manipulate images in various ways, such as changing the pose or expression of a person in an image, or changing the background. This has applications in entertainment, advertising, and social media.
- ✓ Image colorization: GANs can be used to colorize grayscale images, which has applications in photography and art.
- ✓ Face swapping: GANs can be used to swap faces in images or videos, which has applications in entertainment and social media.
- ✓ ……