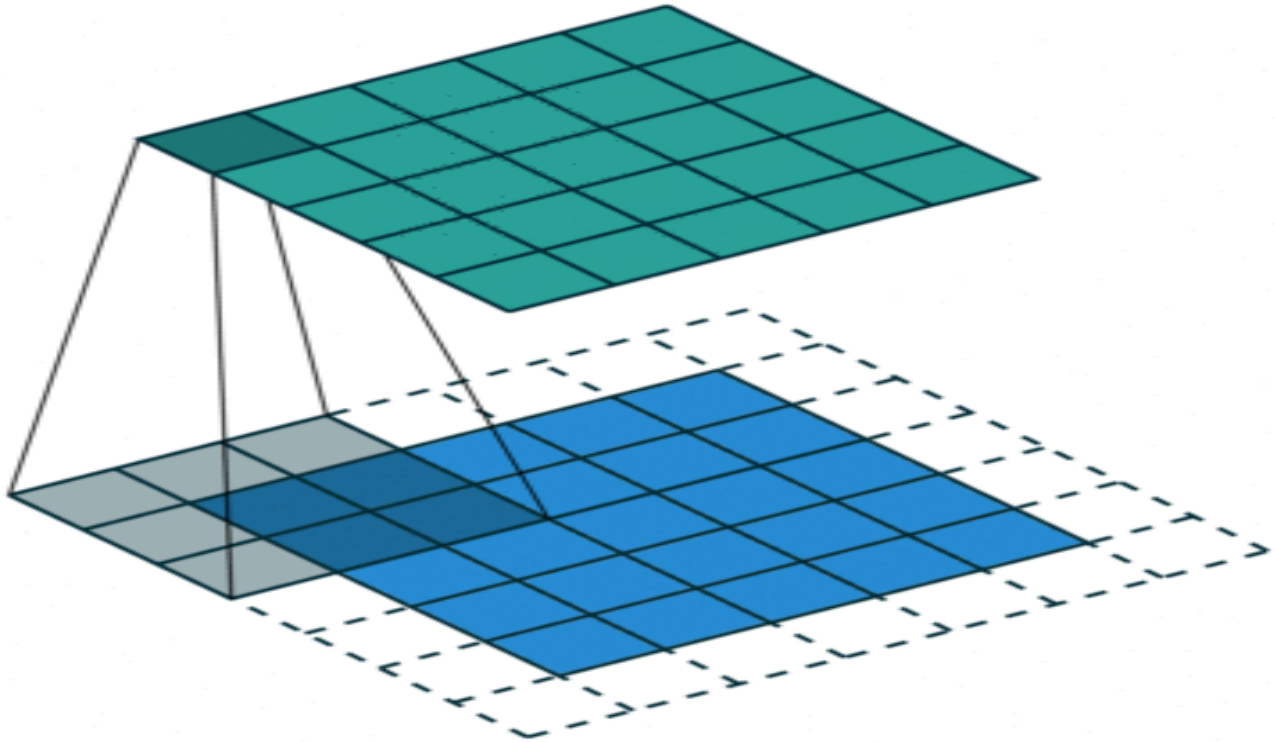
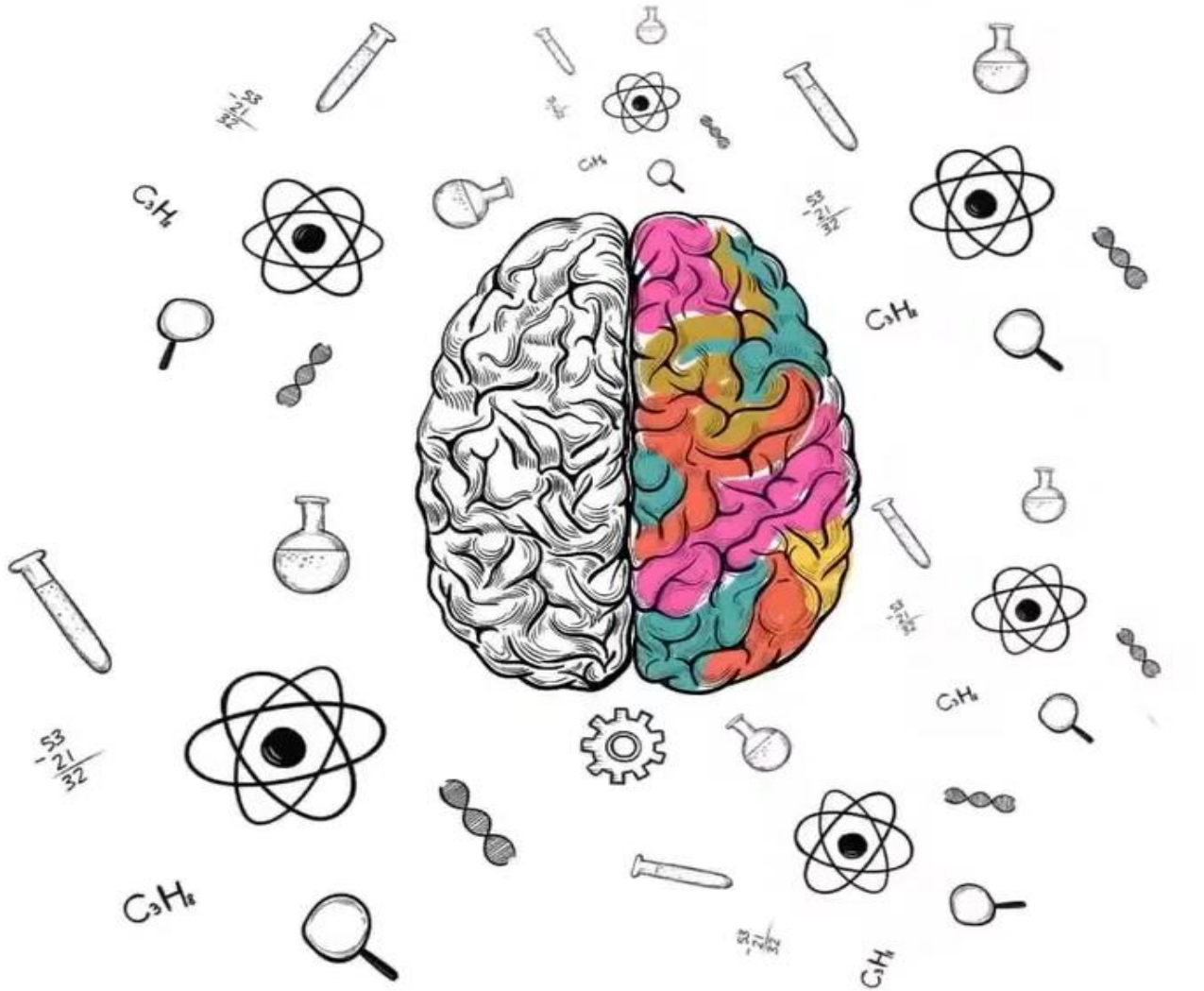
Convolutional Neural Network

By
Prof(Dr) Premanand P Ghadekar



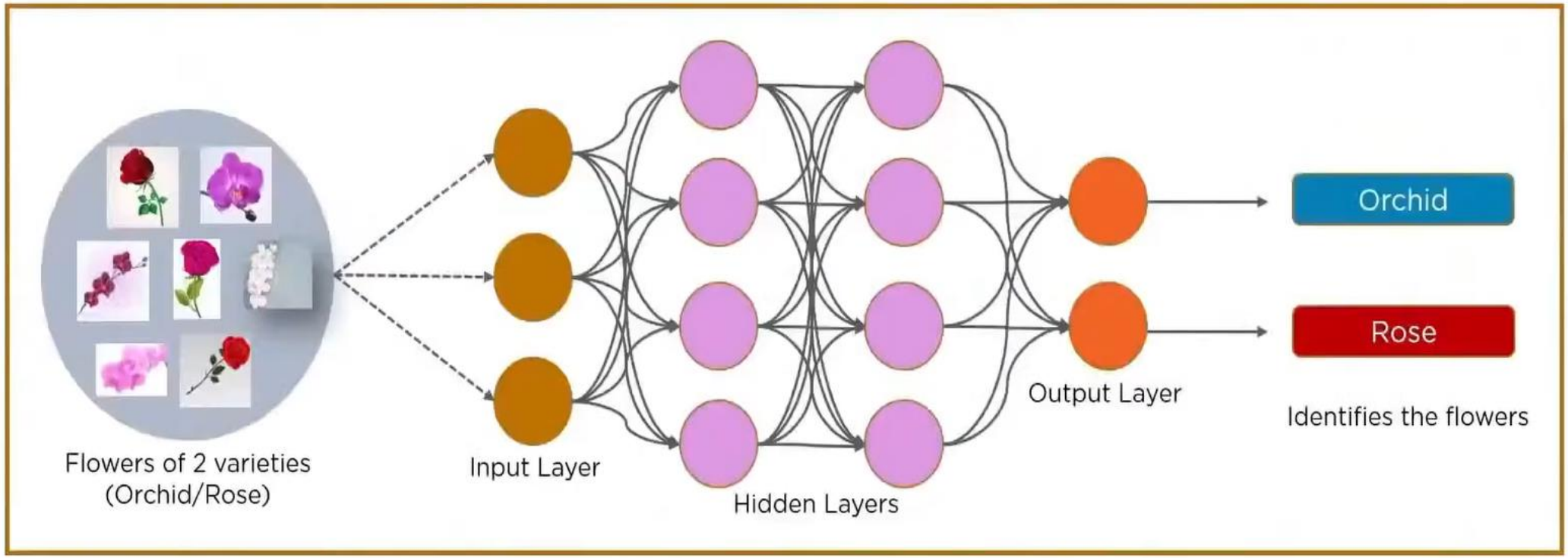
Outline

- Introduction
- Basic Concepts
- Artificial Intelligence
- Deep Learning
- Neural Network



Convolution Neural Network – Understanding

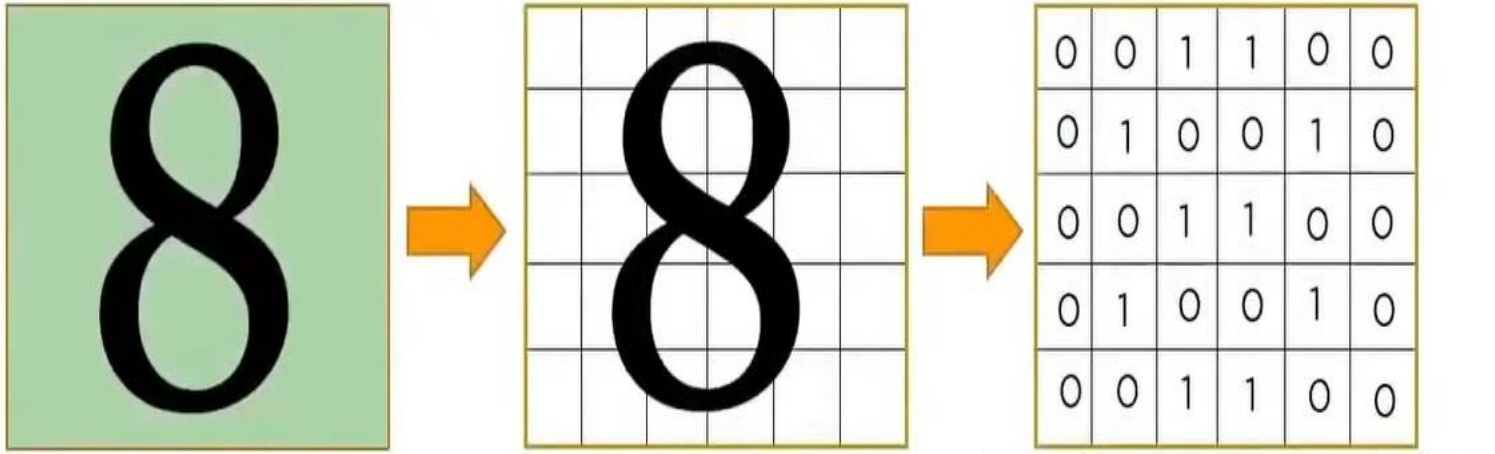
CNN is a feed forward neural network that is generally used to analyze visual images by processing data with grid like topology. A CNN is also known as a "*ConvNet*"



Convolution Neural Network – Understanding

Convolution operation forms the basis of any Convolution Neural Network

In CNN, every image is represented in the form of arrays of pixel values



Real Image of the digit 8

Represented in the form of an array

Digit 8 represented in the form of pixels of 0's and 1's

What is Convolution?

- Mathematically, **convolution is the summation of the element-wise product of 2 matrices.(Dot Product)**
- Let us consider an image ‘X’ & a filter ‘Y’ Both of them, i.e. X & Y, are matrices (image X is being expressed in the state of pixels).
- When we convolve the image ‘X’ using filter ‘Y’, we produce the output in a matrix, say’ Z’.

1	2	3
2	0	0
7	9	1

*

3	2	0
3	0	1
0	5	2

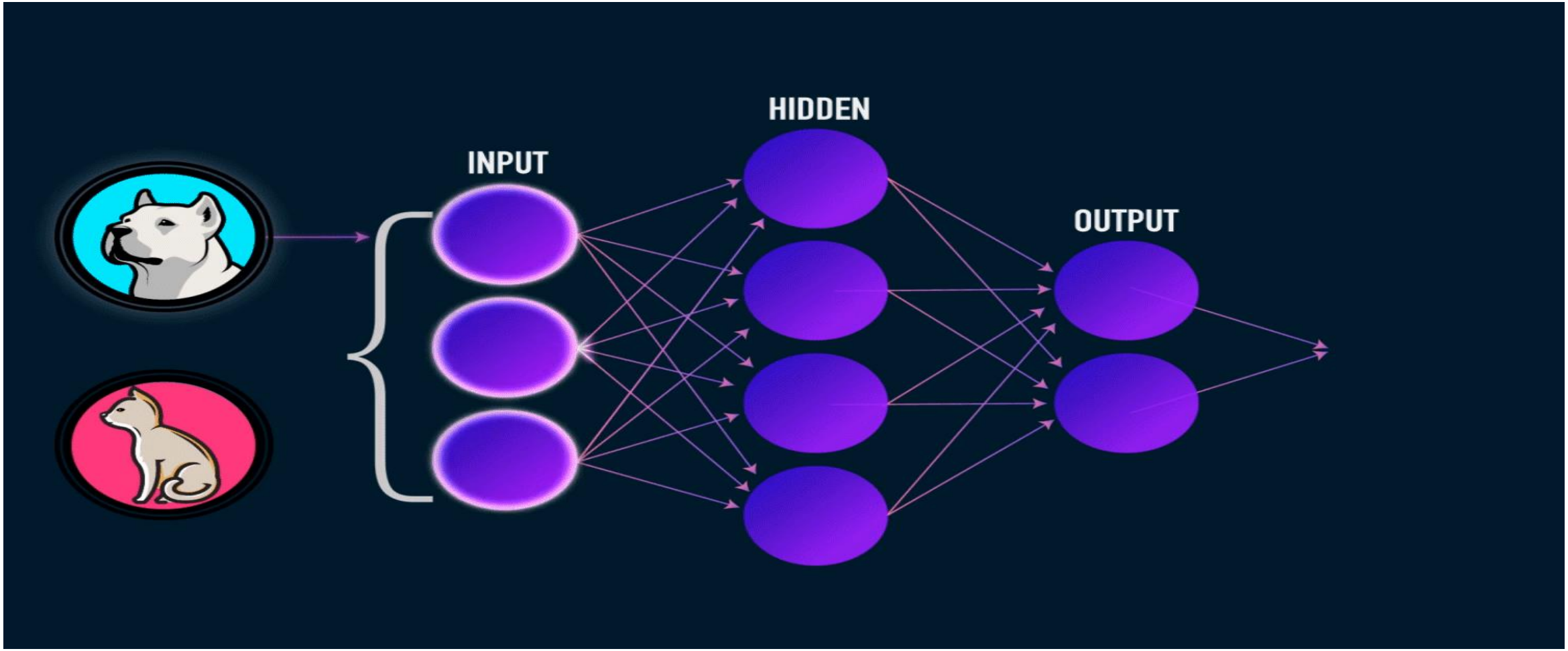
=

1*3 = 3	2*2 = 4	3*0 = 0
2*3 = 6	0*0 = 0	0*1 = 0
7*0 = 0	9*5 = 45	1*2 = 2

Finally, we compute the sum of all the elements in ‘Z’ to get a scalar number, i.e. 3+4+0+6+0+0+0+45+2 = 60

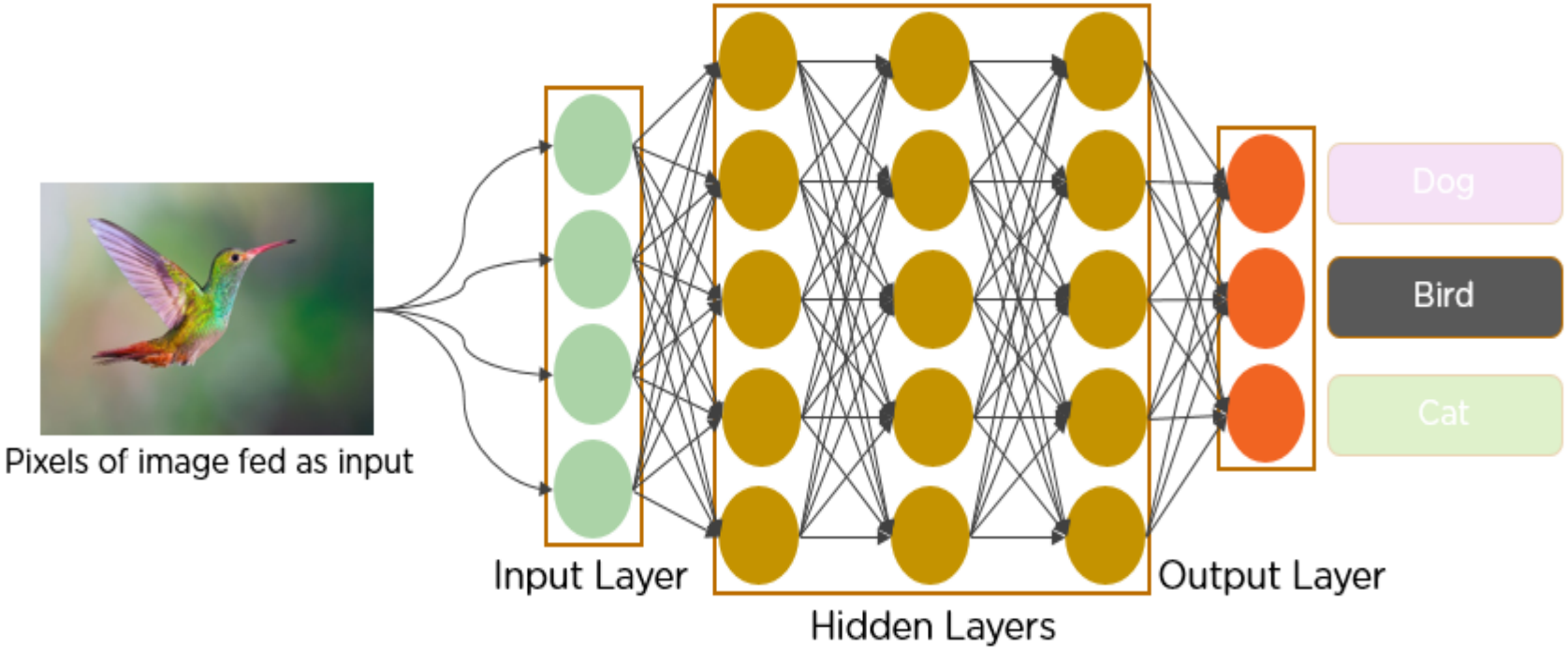
Convolution Neural Network – Understanding

- **Convolutional Neural Network** is a specialized neural network **designed for visual data, such as images & videos.** But CNNs also work well for non-image data **(especially in NLP & text classification).**
- Mathematically, convolution is the summation of the element-wise product of 2 matrices.



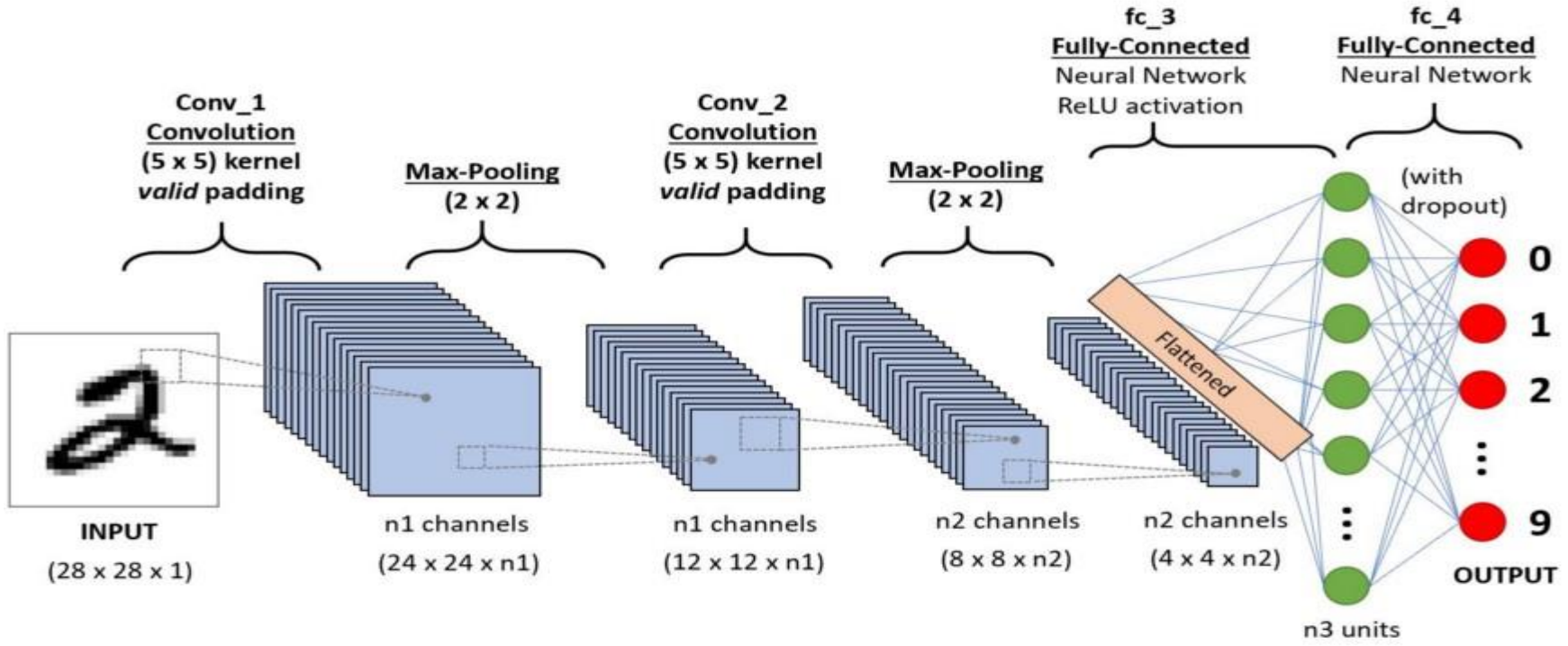
Introduction

- One of the most popular deep neural networks. is **Convolutional Neural Networks** The AI system, which became known as **AlexNet** (named after its main creator, Alex Krizhevsky).
- At the heart of AlexNet was Convolutional Neural Networks a special type of neural network that roughly imitates human vision. Over the years CNNs have become a very important part of many Computer Vision applications



Background of CNN

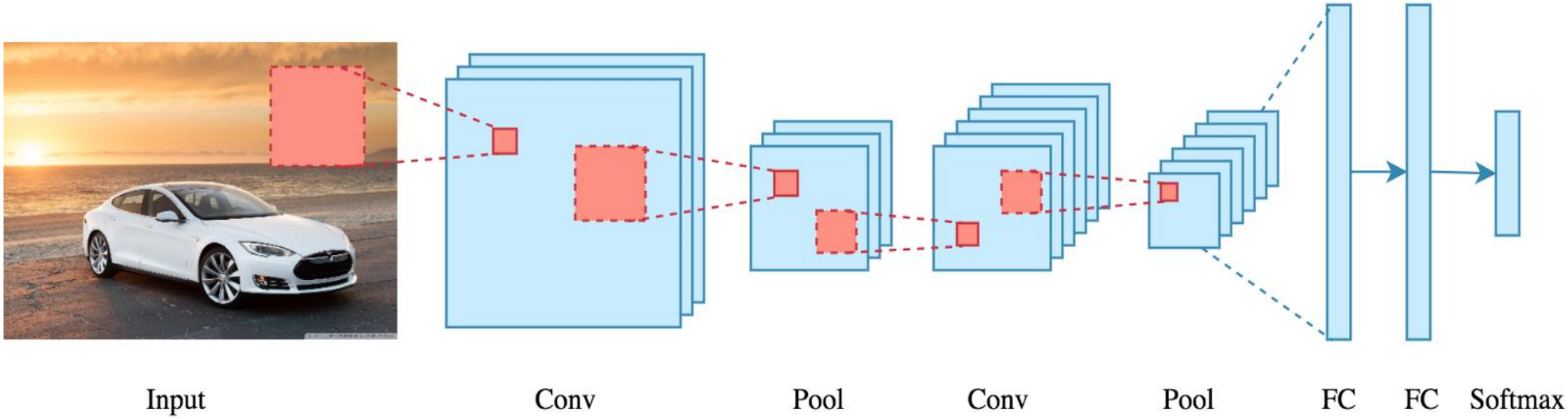
CNN's were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits.
It was mostly used in the postal sectors to read zip codes, pin codes, etc.
Drawback-it requires a large amount of data to train and also requires a lot of computing resources.



In 2012 Alex Krizhevsky realized that it was time to bring back the branch of deep learning that uses multi-layered neural networks.

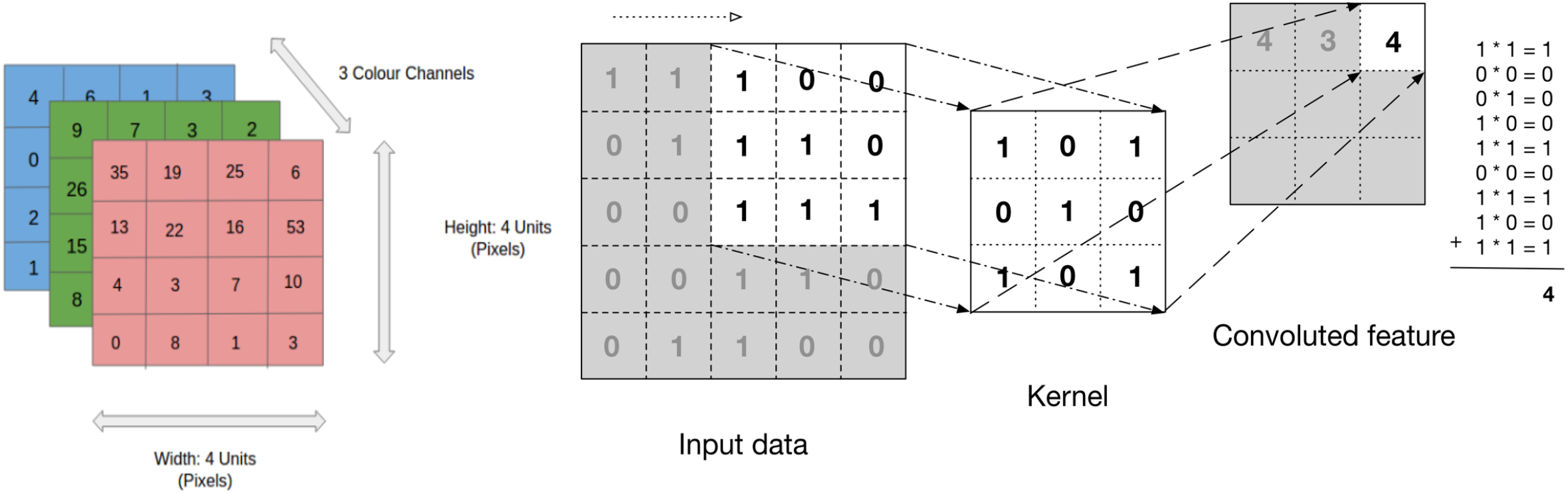
What exactly is a CNN?

- In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery.
- Now when we think of a neural network, we think about matrix multiplications but that is not the case with ConvNet.
- **It uses a special technique called Convolution.**
- Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.



The role of the ConvNet is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction.

How does it work?



The above image shows what a convolution is.

Feature size = ((Image size − Kernel size) / Stride) + 1

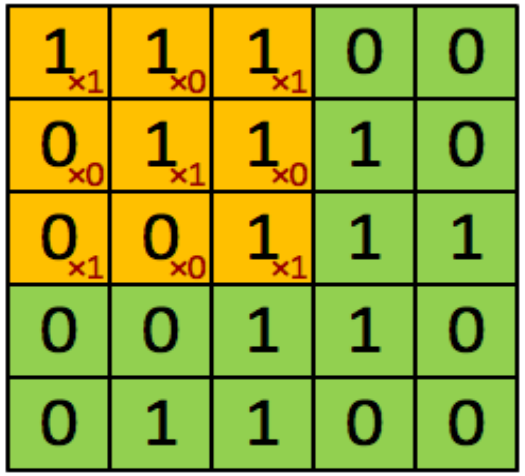
What if you want the feature to be of the same size as the input image?

Soln-Padding

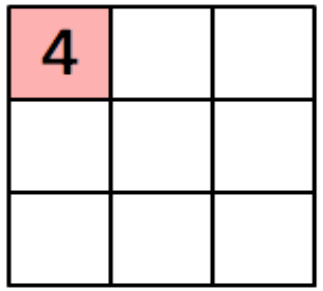
Feature size = ((Image size + 2 * Padding size − Kernel size) / Stride)+1

How does it work?

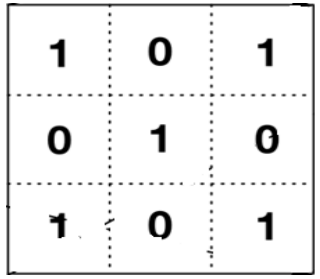
We take a filter/kernel(3×3 matrix) and apply it to the input image to get the convolved feature. This convolved feature is passed on to the next layer.



Image



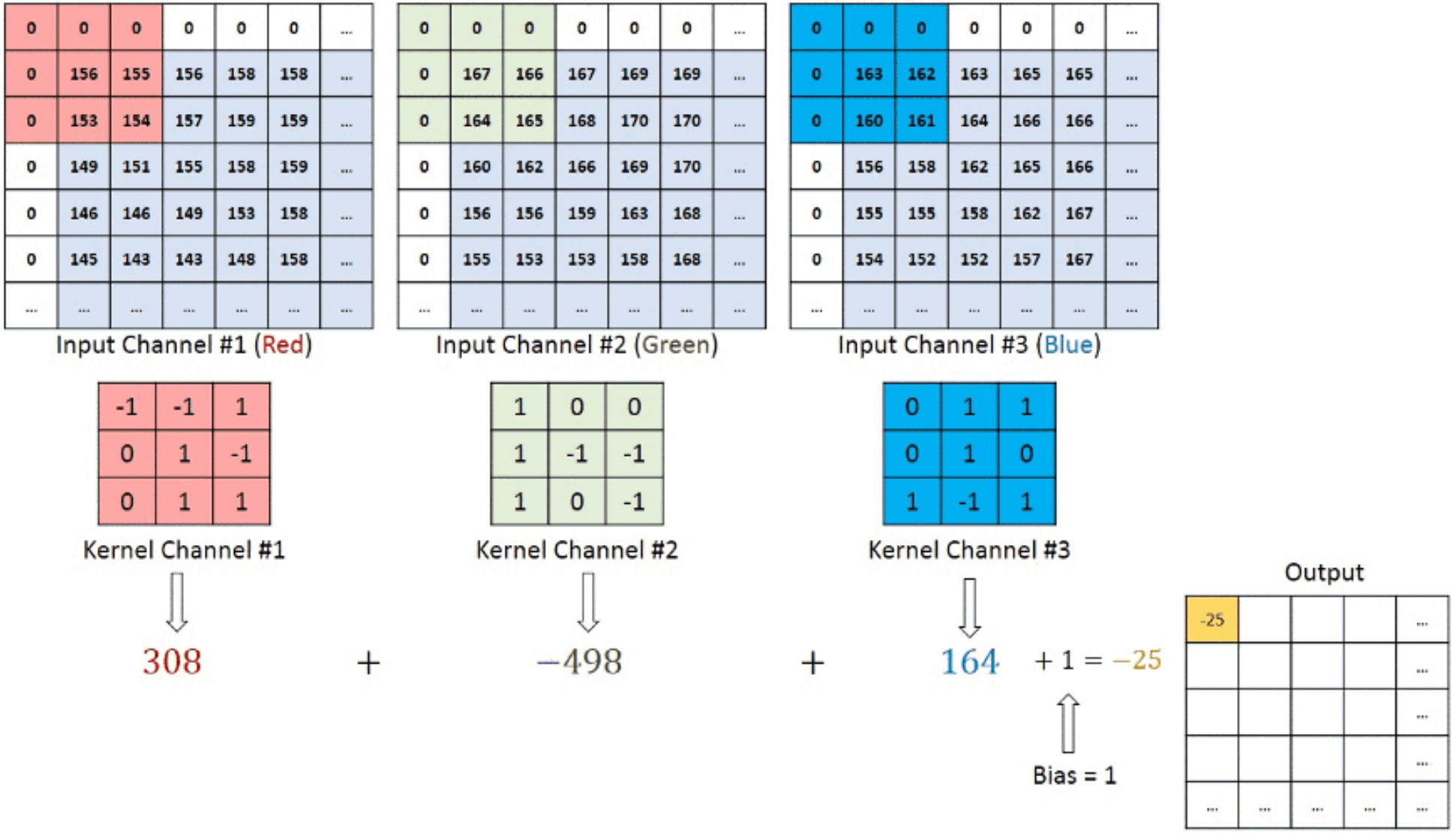
Convolved Feature



Kernel

How does it work?

In the case of RGB color, channel take a look at this animation to understand its working



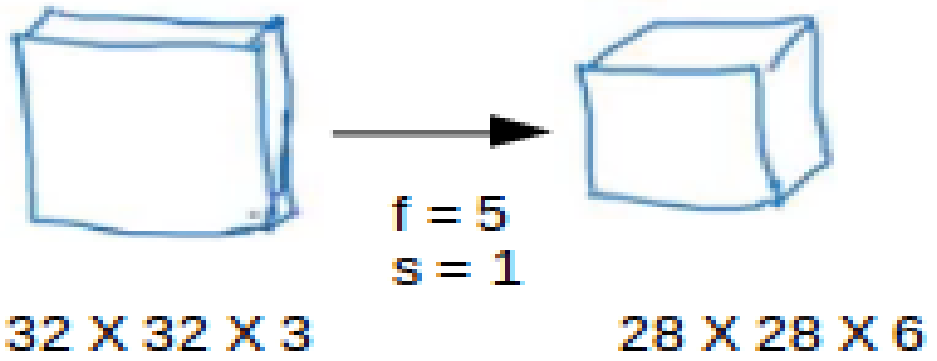
Why Convolutions?

There are primarily two major advantages of using convolutional layers over using just fully connected layers:

1.Parameter sharing

2.Sparsity of connections

Consider the below example:

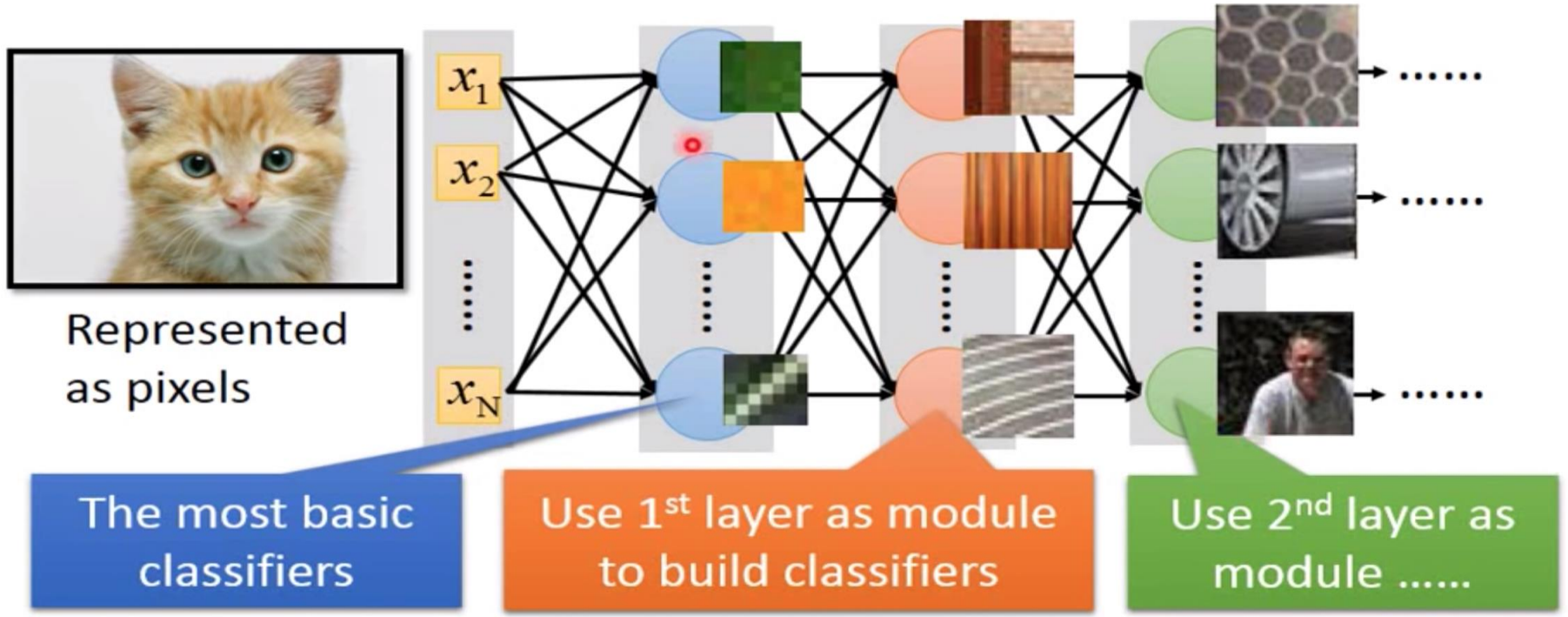


- **ANN**-If we would have used just the fully connected layer, the number of parameters would be = 32*32*3*28*28*6, which is nearly equal to 14 million! Makes no sense, right?
- **CNN**-If we see the number of parameters in case of a convolutional layer, it will be = (5*5 + 1) * 6 (if there are 6 filters), which is equal to 156. **Convolutional layers reduce the number of parameters and speed up the training of the model significantly.**
- The second advantage of convolution is the **sparsity of connections**. For each layer, **each output value depends on a small number of inputs, instead of taking into account all the inputs.**

Why CNN for Image

Why CNN for Image?

[Zeiler, M. D., *ECCV 2014*]



Can the network be simplified by considering the properties of images?

Why CNN for Image

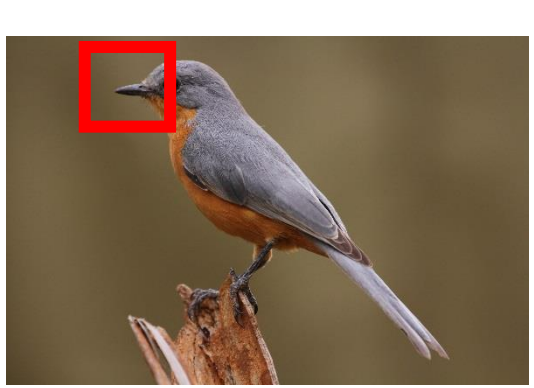
- Some patterns are much smaller than the whole image

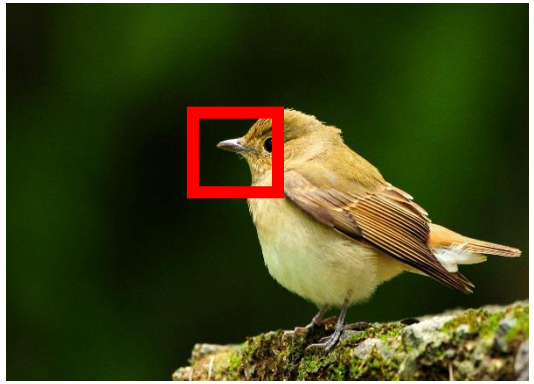
A neuron does not have to see the whole image to discover the pattern.

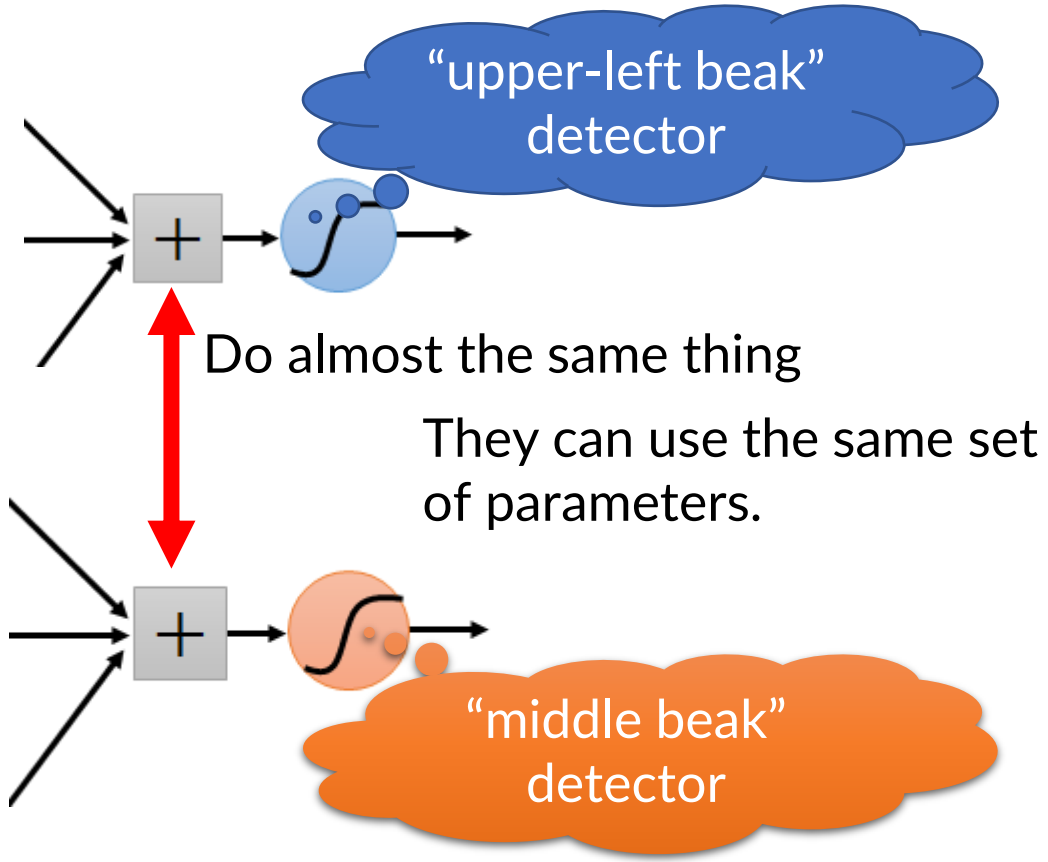
Connecting to small region with less parameters

Why CNN for Image

- The same patterns appear in different regions.







Why CNN for Image

- Subsampling the pixels will not change the object

bird

subsampling

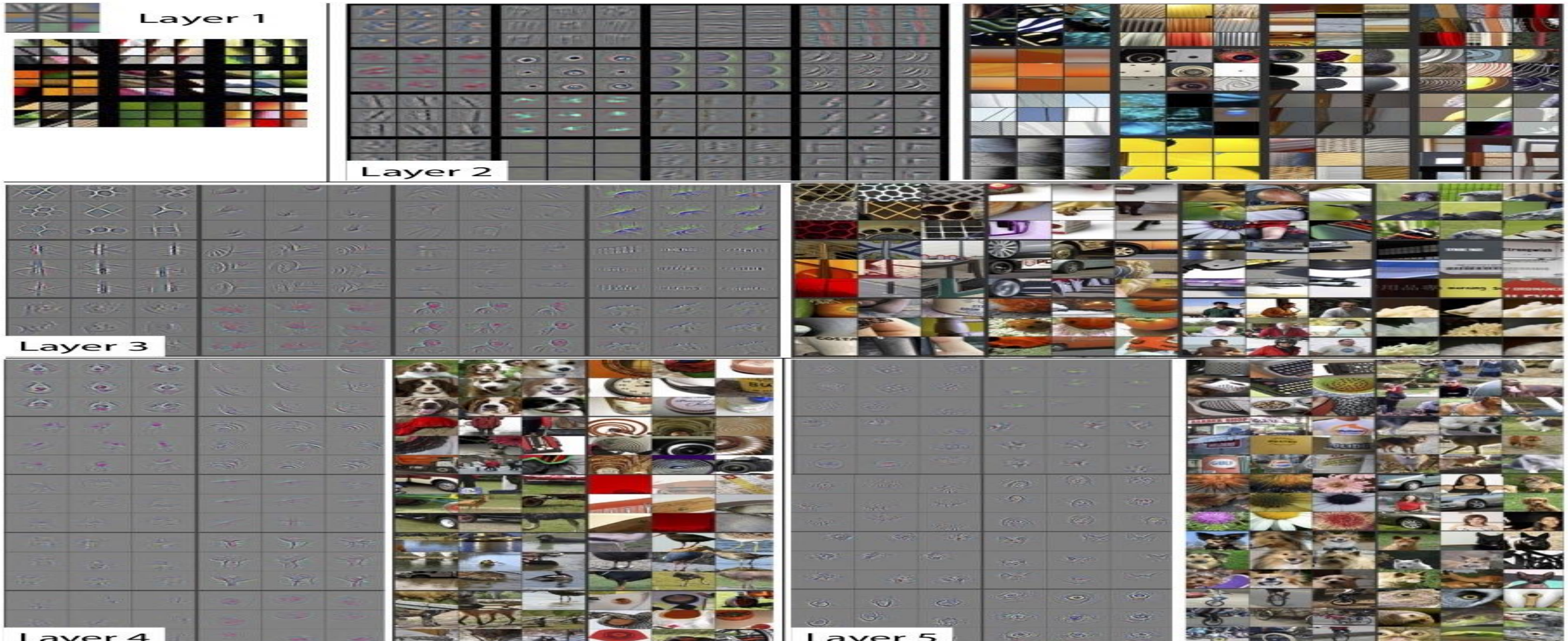
bird

We can subsample the pixels to make image smaller

→ Less parameters for the network to process the image

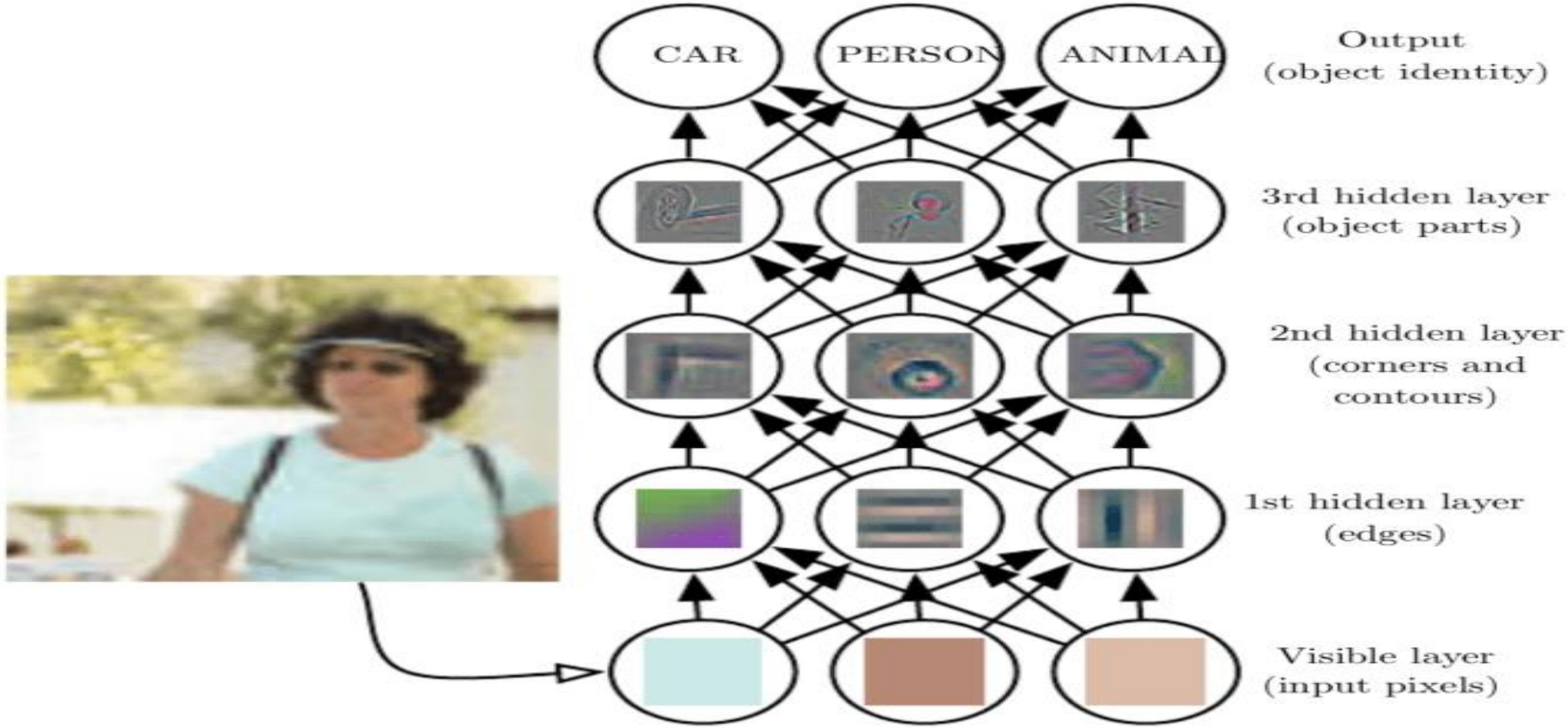
How does it work?

The first layer usually extracts basic features such as horizontal or diagonal edges.
This output is passed on to the next layer which detects more complex features such as corners or combinational edges.
As we move deeper into the network it can identify even more complex features such as objects, faces, etc.



How does it work?

Based on **the activation map of the final convolution layer**, the **classification layer outputs a set of confidence scores (values between 0 and 1)** that specify how likely the image is to belong to a "class." For instance, if you have a ConvNet that detects cats, dogs, and horses, the output of the final layer is the possibility that the input image contains any of those animals.



CNN has the following five basic components:

- **Convolution** : to detect features in an image while preserving the spacial features of the image.
- **ReLU** : to make the image smooth and make boundaries distinct. It brings a sharp or drastic change when there is a different feature.
- **Pooling** : to help fix distorted images.
- **Flattening** : to turn the image into a suitable representation.

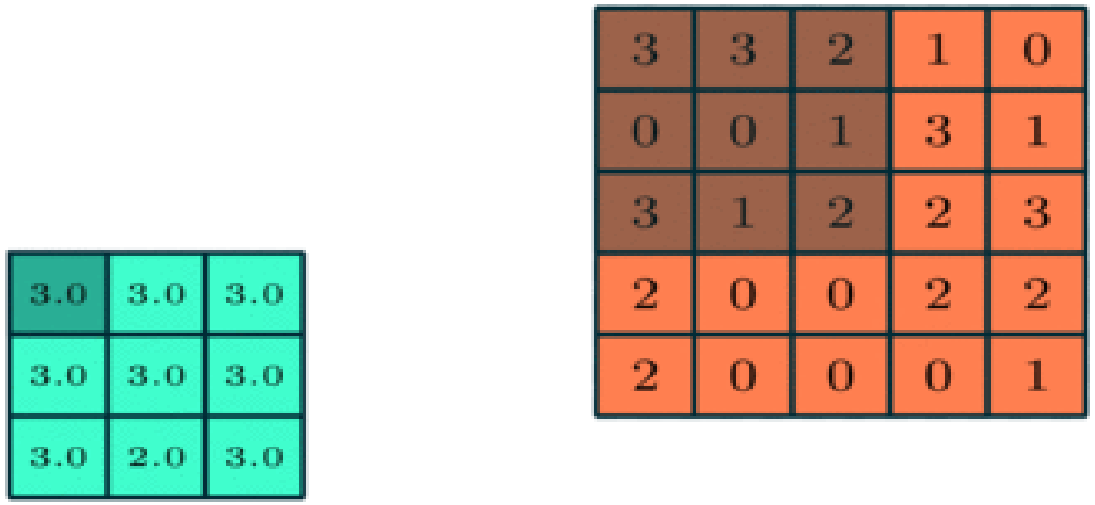
  In the flattening procedure, we basically take the **elements in a pooled feature map** and **put them in a vector form. This becomes the input layer for the upcoming ANN.**
- **Full connection** : to process the data in a neural network.

  Once the image is convolved, max pooled and flattened, the result is a vector. This vector acts as the input layer for an ANN which then works normally to detect the image.

What's a pooling layer?

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature.

This is to decrease the computational power required to process the data by reducing the dimensions. There are two types of pooling average pooling and max pooling.



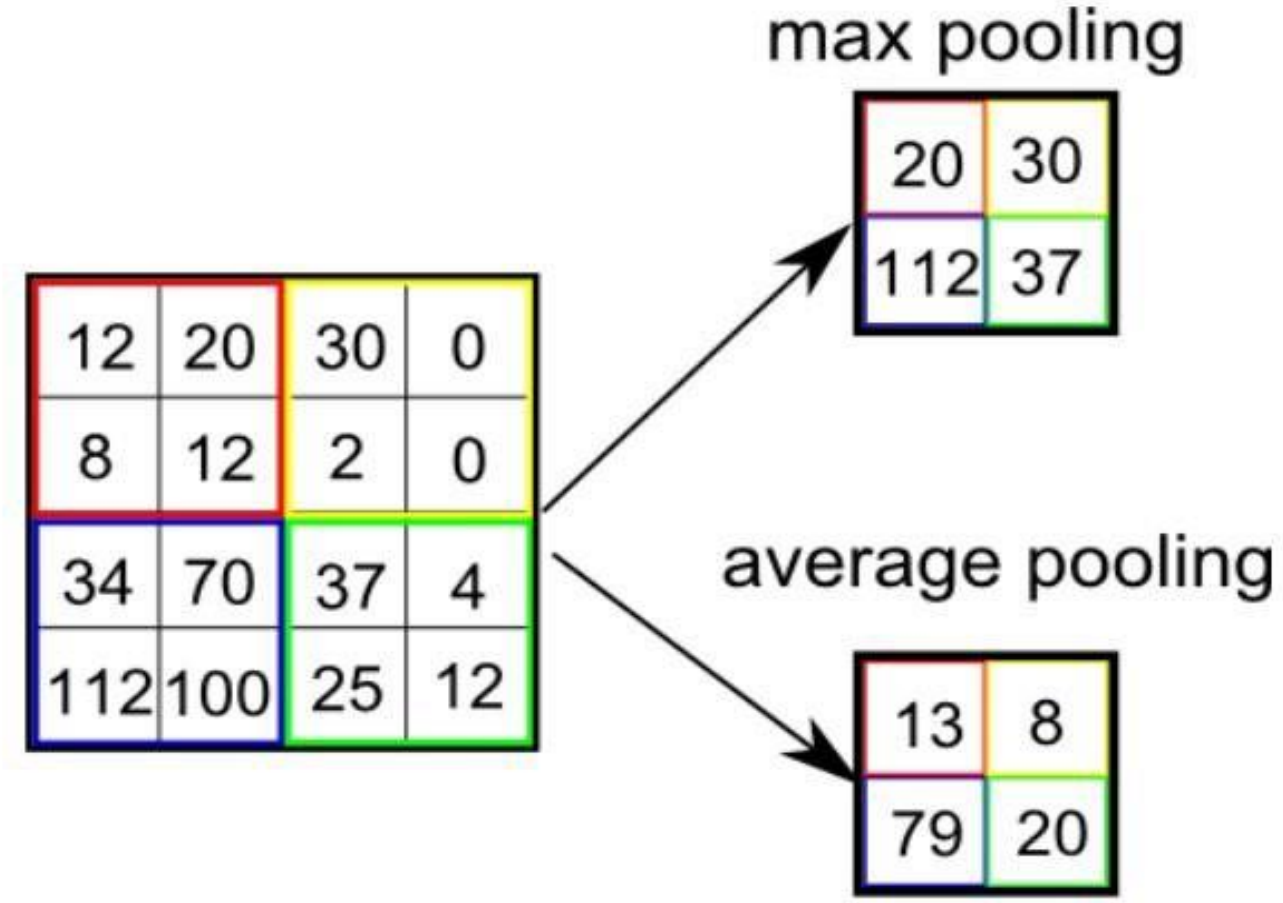
3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

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

Max Pooling also performs as a **Noise Suppressant**. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction.

Average Pooling Vs Max Pooling

Average Pooling returns the average of all the values from the portion of the image covered by the Kernel. Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.



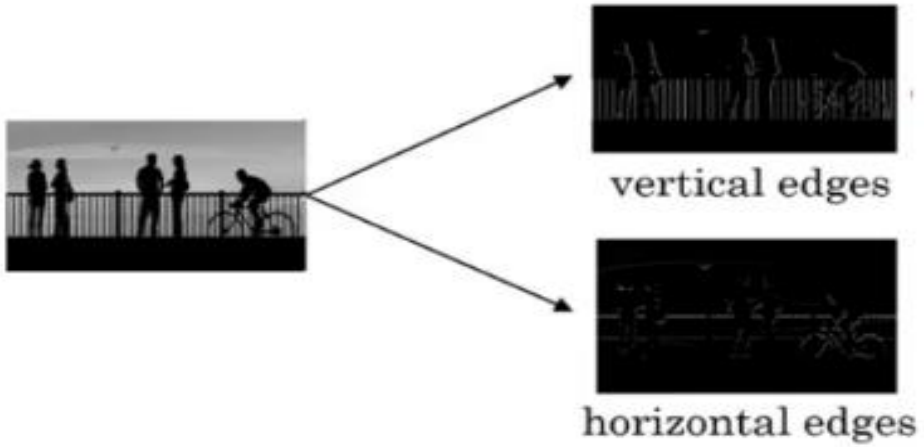
What are Filters/Kernels?

- A **filter provides** a measure for **how close a patch or a region of the input resembles a feature**. A feature may be any prominent aspect – a vertical edge, a horizontal edge, an arch, a diagonal, etc.
- A filter acts as a single template or pattern, which, when convolved across the input, **finds similarities between the stored template & different locations/regions in the input image.**

Edge Detection Example

The **early layers of a neural network detect edges** from an image. **Deeper layers might be able to detect the cause of the objects** and even **more deeper layers might detect the cause of complete objects (like a person's face).** Suppose we are given the below image:

As you can see, **there are many vertical and horizontal edges in the image**. The first thing to do is to detect these edges:



Edge Detection Example

But, how do we detect these edges? To illustrate this, let's take a 6 X 6 grayscale image (i.e. only one channel):

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Next, we convolve this 6 X 6 matrix with a 3 X 3 filter:

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	**1**	3
0	**1**	3	1	**7**	8
4	2	1	6	2	8
2	4	5	2	3	9

6 X 6 image

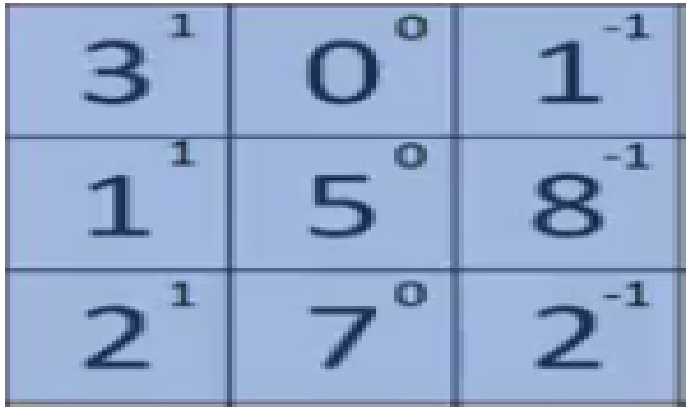
*

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

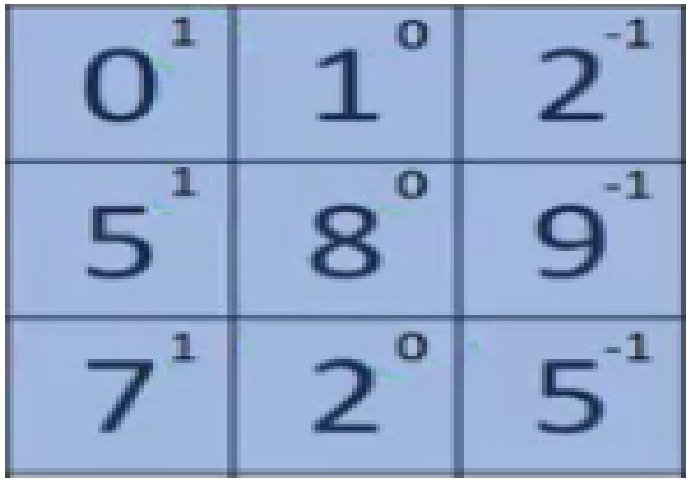
3 X 3 filter

Edge Detection Example

After the convolution, we will get a 4 X 4 image. The first element of the 4 X 4 matrix will be calculated as:



So, we take the first 3 X 3 matrix from the 6 X 6 image and multiply it with the filter. Now, the first element of the 4 X 4 output will be the sum of the element-wise product of these values, i.e. 3*1 + 0 + 1*-1 + 1*1 + 5*0 + 8*-1 + 2*1 + 7*0 + 2*-1 = -5. To calculate the second element of the 4 X 4 output, we will shift our filter one step towards the right and again get the sum of the element-wise product:



Edge Detection Example

Similarly, we will convolve over the entire image and get a 4 X 4 output:

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

So, convolving a 6 X 6 input with a 3 X 3 filter gave us an output of 4 X 4. Consider one more example:

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

*

Limitations

- Several studies have shown that **CNNs trained on ImageNet and other popular datasets fail to detect objects when they see them under different lighting conditions and from new angles.**
- CNN's were **widely used to moderate content on social media**. But despite the vast resources of images and videos that they were trained on it **still isn't able to completely block and remove inappropriate content.**

The whole CNN

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten

Fully Connected Feedforward network

cat dog ……

The whole CNN

Property 1

- Some patterns are much smaller than the whole image

Property 2

- The same patterns appear in different regions.

Property 3

- Subsampling the pixels will not change the object

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten

The whole CNN

Convolution

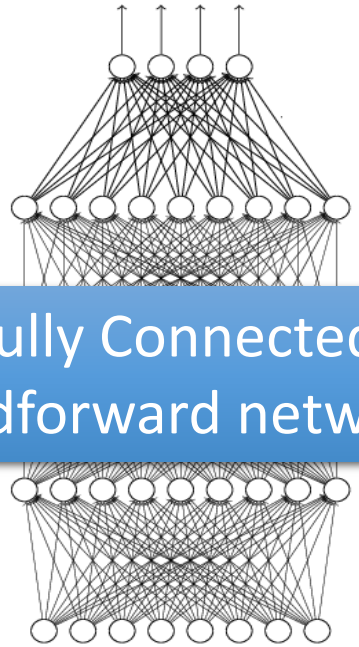
Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten



Fully Connected Feedforward network

cat dog ……

CNN – Convolution

Those 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

Matrix

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

Filter 2

Matrix

⋮

Property 1

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

CNN – Convolution

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

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

Filter 1

3 -1

CNN – Convolution

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

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

Filter 1

3 -3

We set stride=1 below

CNN – Convolution

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

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

Property 2

CNN – Convolution

stride=1

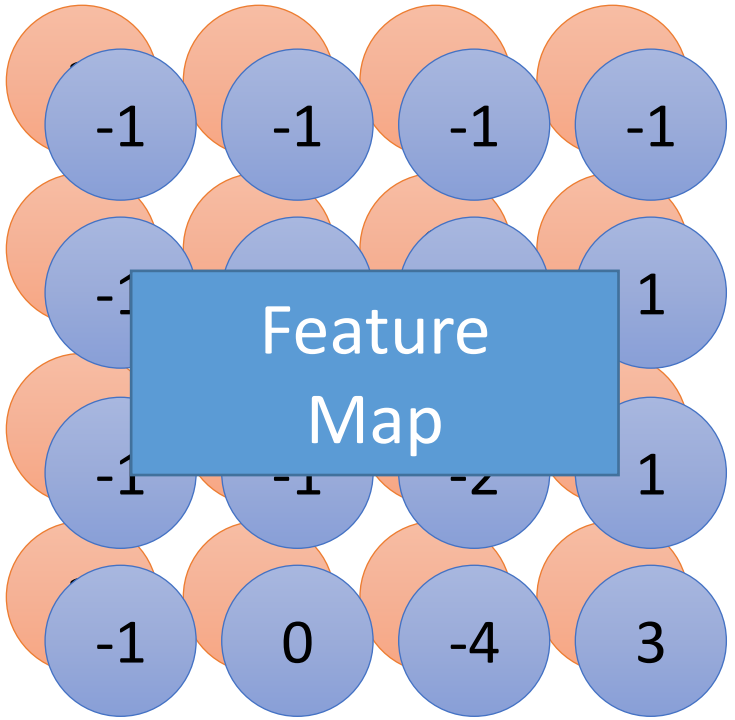
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

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

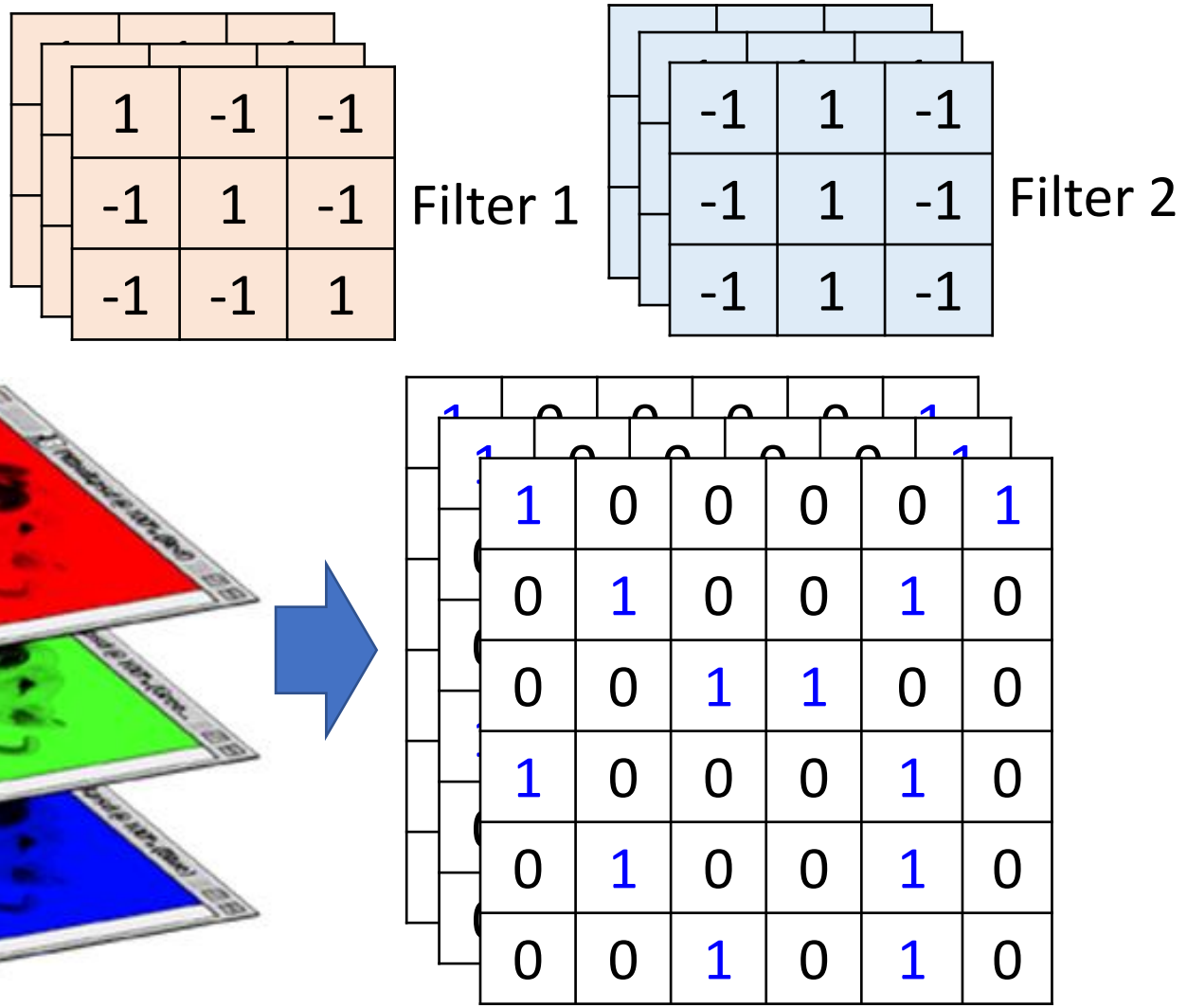
Filter 2

Do the same process for every filter

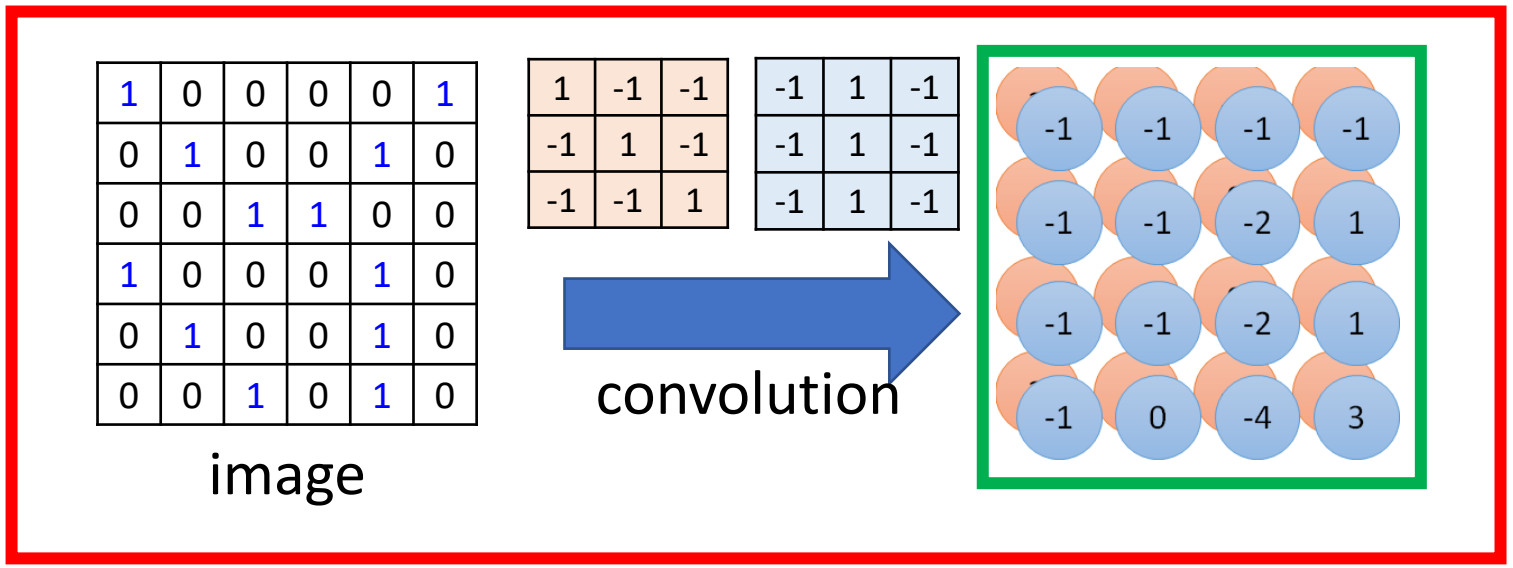


4 x 4 image

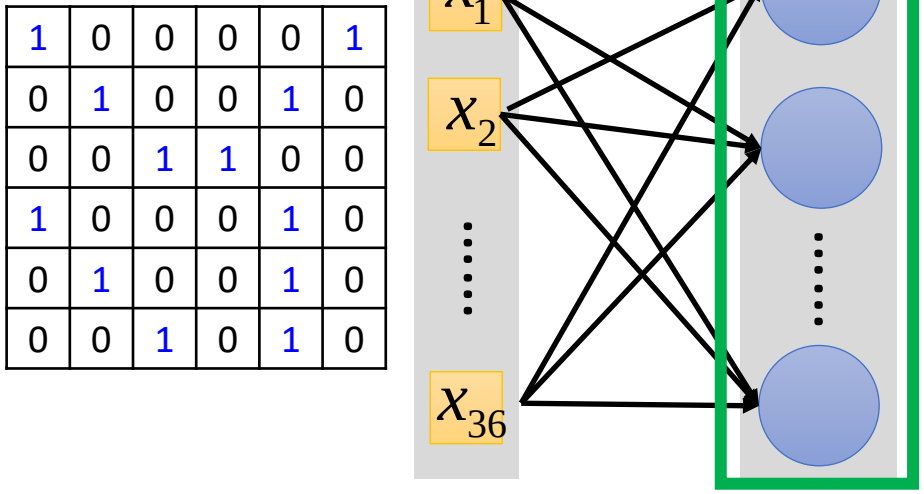
CNN – Colorful image



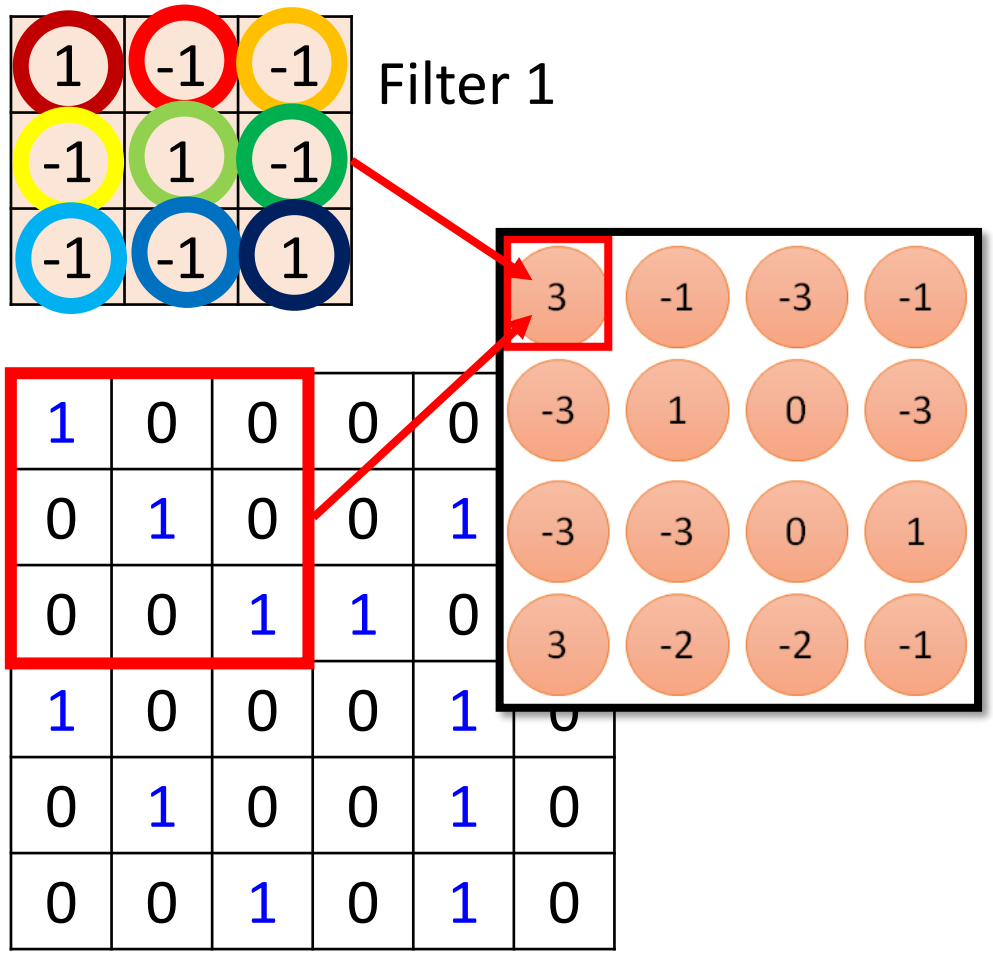
Convolution v.s. Fully Connected



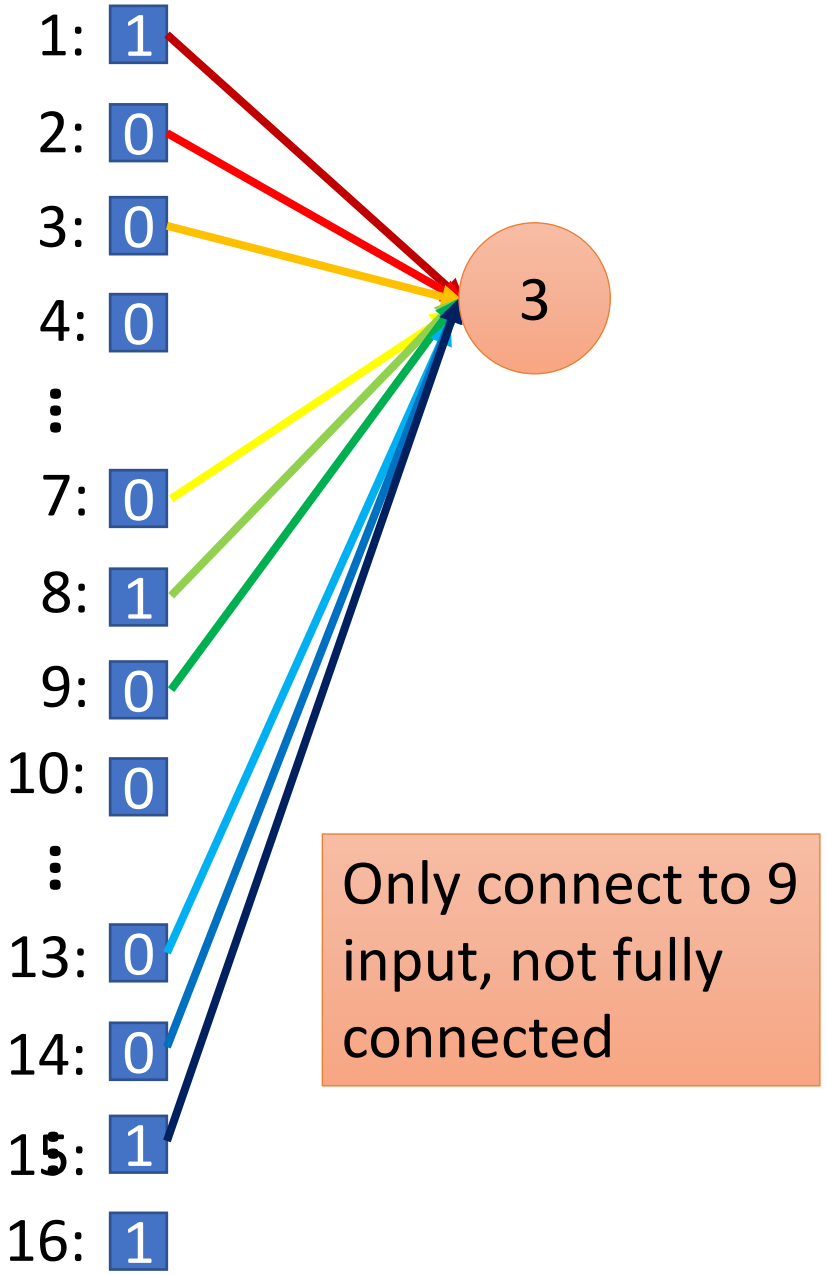
Fully-connected



Convolution v.s. Fully Connected

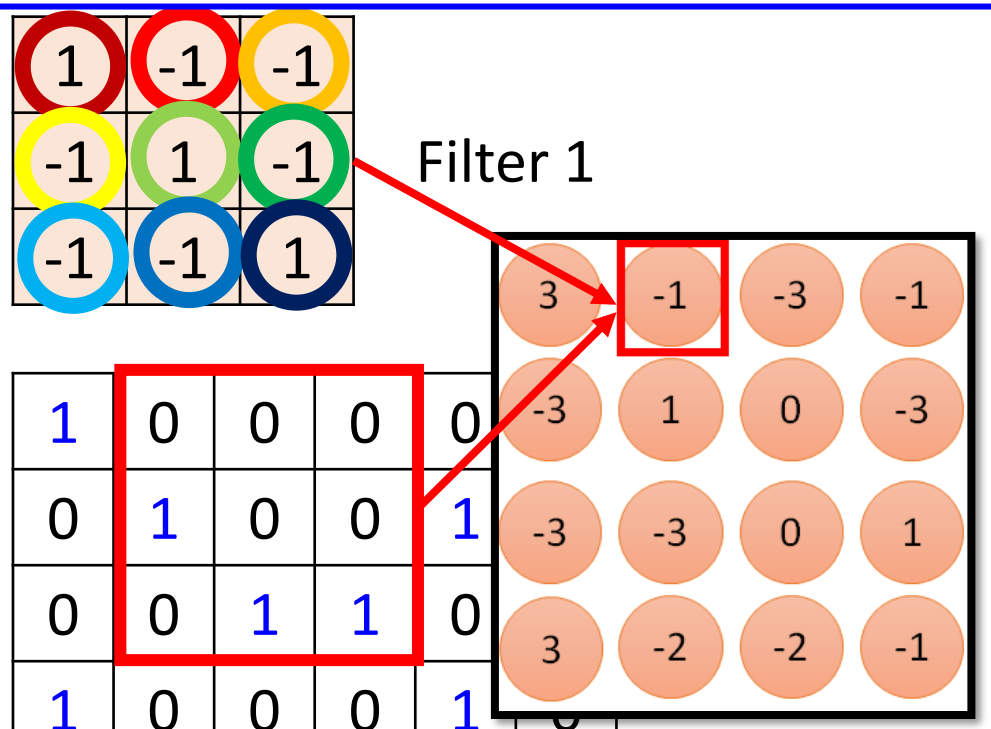


Less parameters!



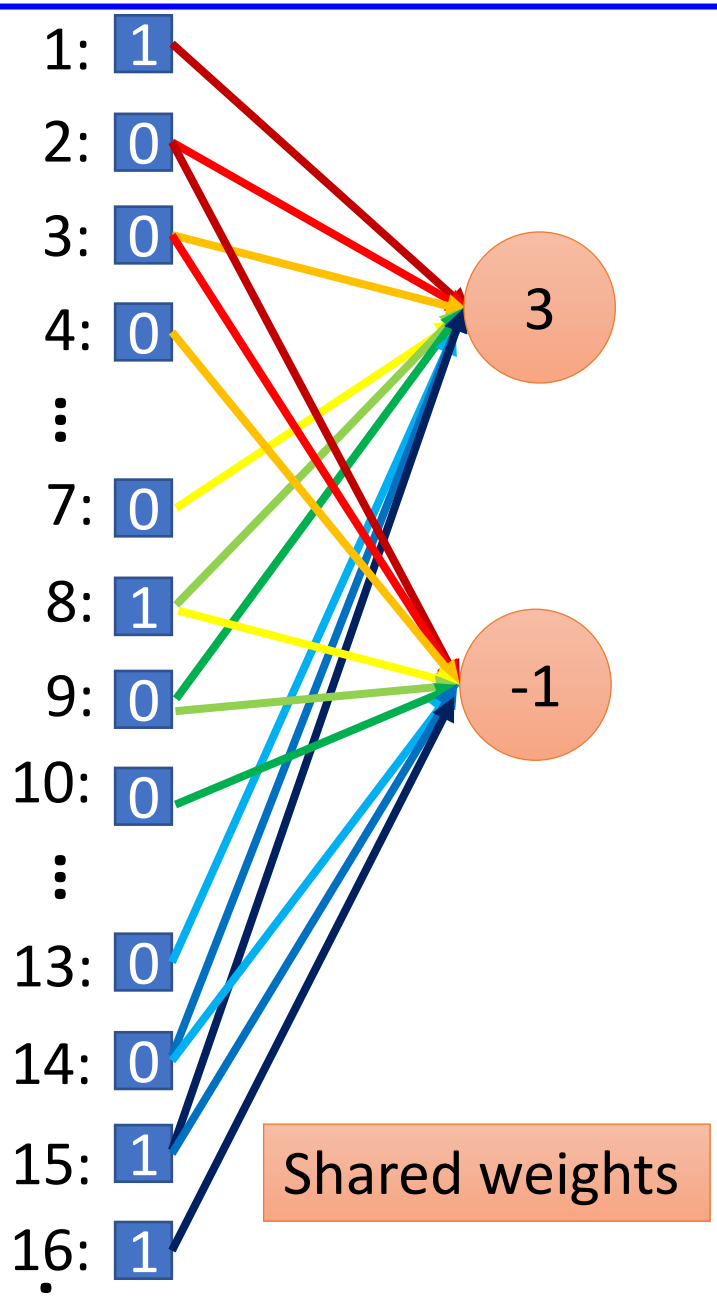
Only connect to 9 input, not fully connected

Convolution v.s. Fully Connected



Less parameters!

Even less parameters!



Shared weights

The whole CNN

Convolution

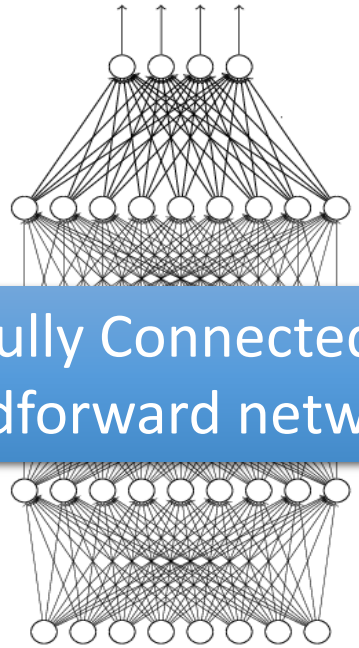
Max Pooling

Convolution

Max Pooling

Can repeat many times

Flatten



Fully Connected Feedforward network

cat dog ……

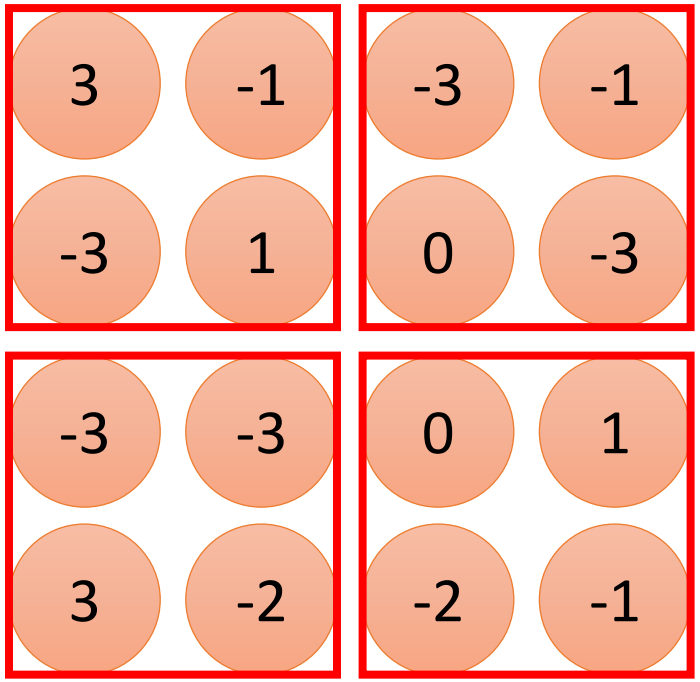
CNN – Max Pooling

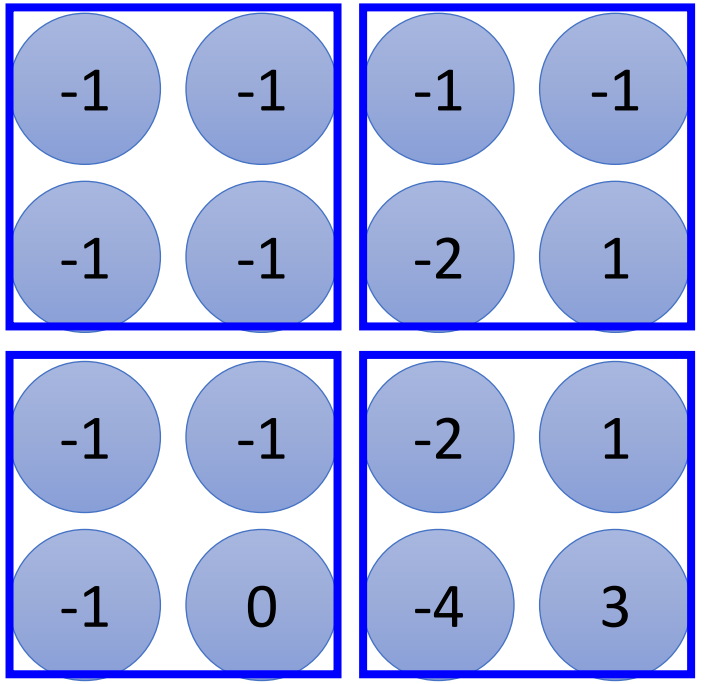
1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

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

Filter 2

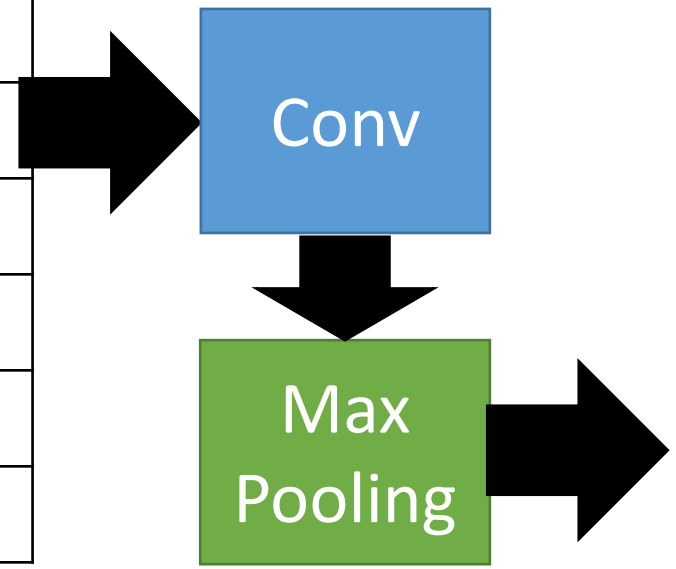




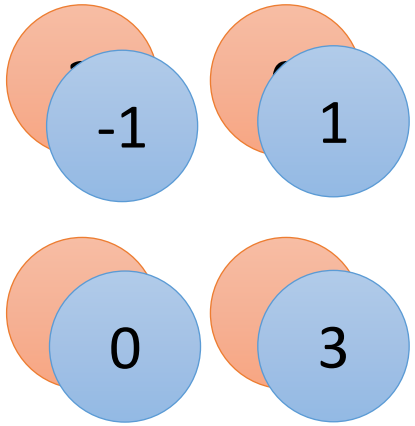
CNN – Max Pooling

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



New image but smaller



2 x 2 image

Each filter is a channel

The whole CNN

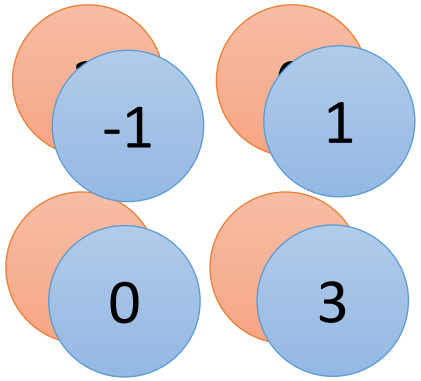
Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times



A new image

Smaller than the original image

The number of the channel is the number of filters

The whole CNN

Convolution

Max Pooling

A new image

Convolution

Max Pooling

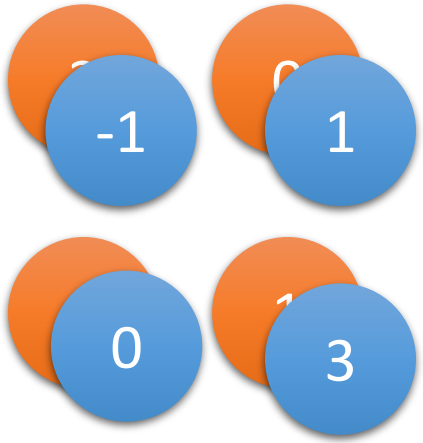
A new image

Flatten

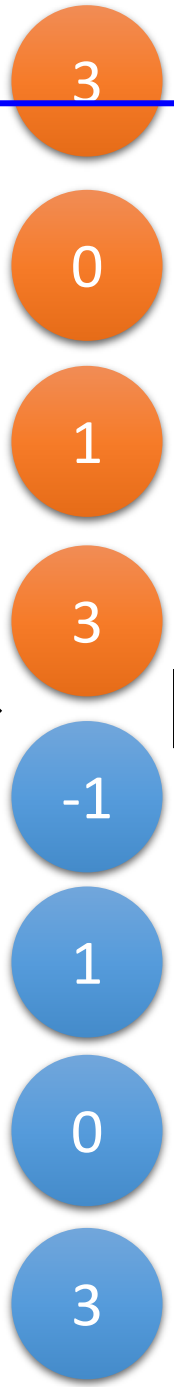
Fully Connected Feedforward network

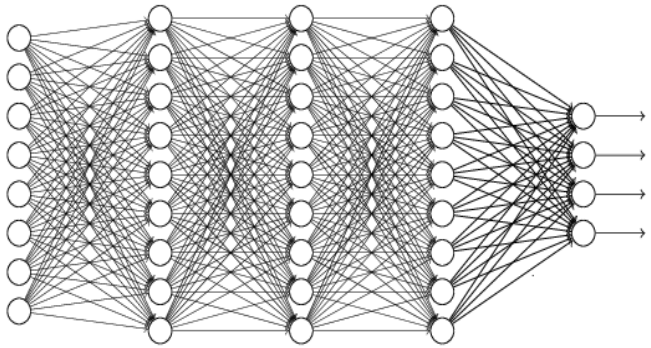
cat dog ……

Flatten



Flatten

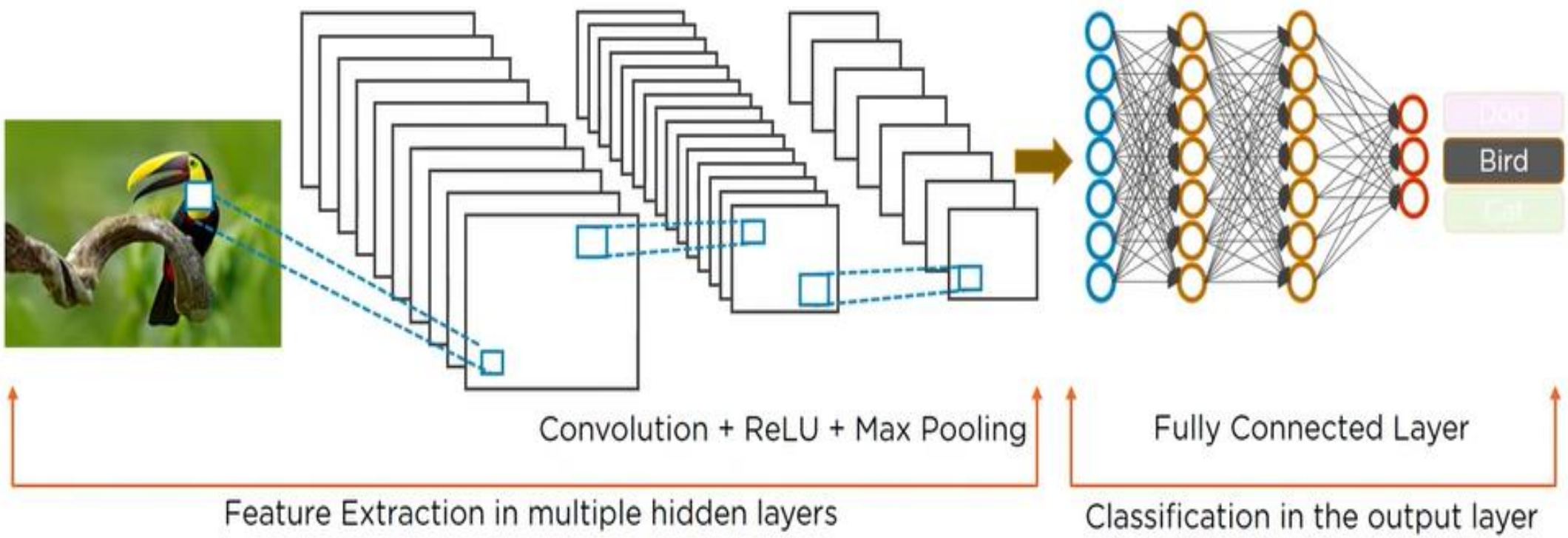




Fully Connected Feedforward network

CNN-Example to detect Bird

Lets see the entire process how CNN recognizes a bird



CNN-Example to calculate parameters

SL.No		Activation Shape	Activation Size	# Parameters
1.	Input Layer:	(32, 32, 3)	3072	0
2.	CONV1 (f=5, s=1)	(28, 28, 8)	6272	608
3.	POOL1	(14, 14, 8)	1568	0
4.	CONV2 (f=5, s=1)	(10, 10, 16)	1600	3216
5.	POOL2	(5, 5, 16)	400	0
6.	FC3	(120, 1)	120	48120
7.	FC4	(84, 1)	84	10164
8.	Softmax	(10, 1)	10	850

Input layer: Input layer has nothing to learn, at its core, what it does is just provide the input image's shape. So, no learnable parameters here. Thus, number of parameters = 0.

CONV layer: ((m * n * d)+1)* k) where-width m, height n, previous layer's filters d, and k in the current layer. - (((5*5*3)+1)*8) = 608.

POOL layer: This has got no learnable parameters because all it does is calculate a specific number, no backprop learning involved! Thus, number of **parameters = 0**.

CNN-Example to calculate parameters

SL.No		Activation Shape	Activation Size	# Parameters
1.	Input Layer:	(32, 32, 3)	3072	0
2.	CONV1 (f=5, s=1)	(28, 28, 8)	6272	608
3.	POOL1	(14, 14, 8)	1568	0
4.	CONV2 (f=5, s=1)	(10, 10, 16)	1600	3216
5.	POOL2	(5, 5, 16)	400	0
6.	FC3	(120, 1)	120	48120
7.	FC4	(84, 1)	84	10164
8.	Softmax	(10, 1)	10	850

Parameters in the fourth **CONV2(filter shape =5*5, stride=1) layer is: ((shape of width of filter * shape of height filter * number of filters in the previous layer+1) * number of filters) = (((5*5*8)+1)*16) = 3216.**

The fifth **POOL2 layer** has no parameters.

Parameters in the Sixth **FC3 layer is((current layer c*previous layer p)+1*c) = 120*400+1*120= 48120.**

Parameters in the Seventh **FC4 layer is: ((current layer c*previous layer p)+1*c) = 84*120+1* 84 = 10164.**

The Eighth **Softmax layer has ((current layer c*previous layer p)+1*c) parameters = 10*84+1*10 = 850.**

CNN Applications

- *Convolutional Neural Networks (CNNs) are a type of Neural Network that has excelled in a number of contests involving Computer Vision and Image Processing.*
- *Image Classification and Segmentation, Object Detection, Video Processing, Natural Language Processing, and Speech Recognition are just a few of CNN's fascinating application areas.*

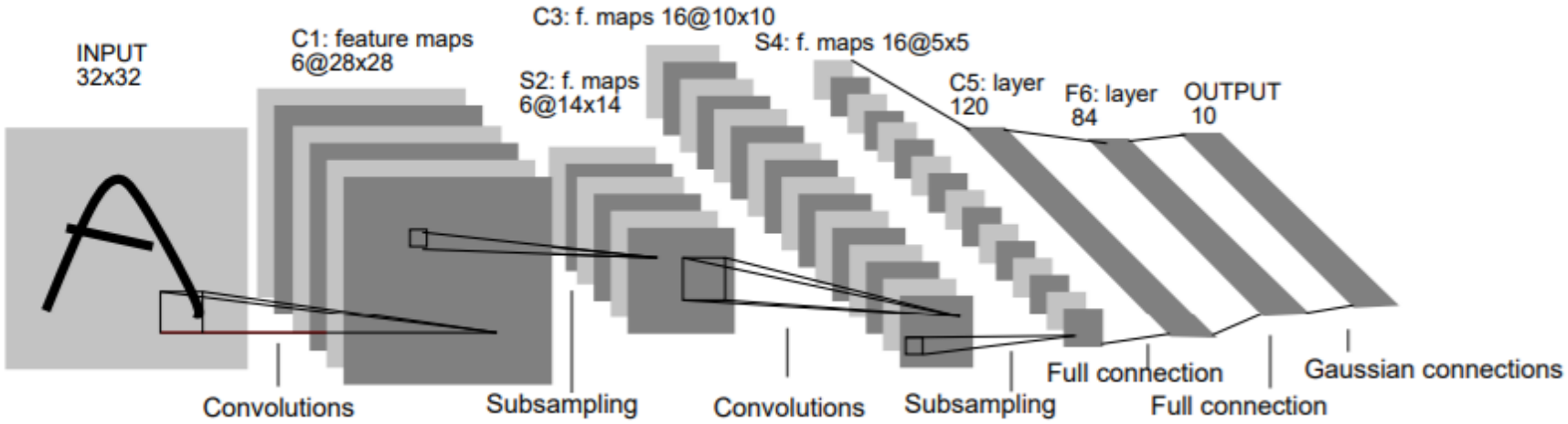
Classic Networks

The following are the popular CNN networks:

1. LeNet-5
2. AlexNet
3. VGG-16
4. ResNet
5. GoogleNet/Inception
6. MobileNetV etc.

LeNet-5 Networks

Let's start with LeNet-5- It is a 7-level convolutional network.



- **LeNet is the most popular CNN architecture** it is also the **first CNN model** which came in the year 1998. LeNet was originally developed to categorize handwritten digits from 0–9 of the MNIST Dataset.
- It is **made up of seven layers**, each with its own set of trainable parameters.
- It **accepts a 32 × 32 pixel picture**, which is rather huge in comparison to the images in the data sets used to train the network.
- **RELU is the activation function** that has been used.

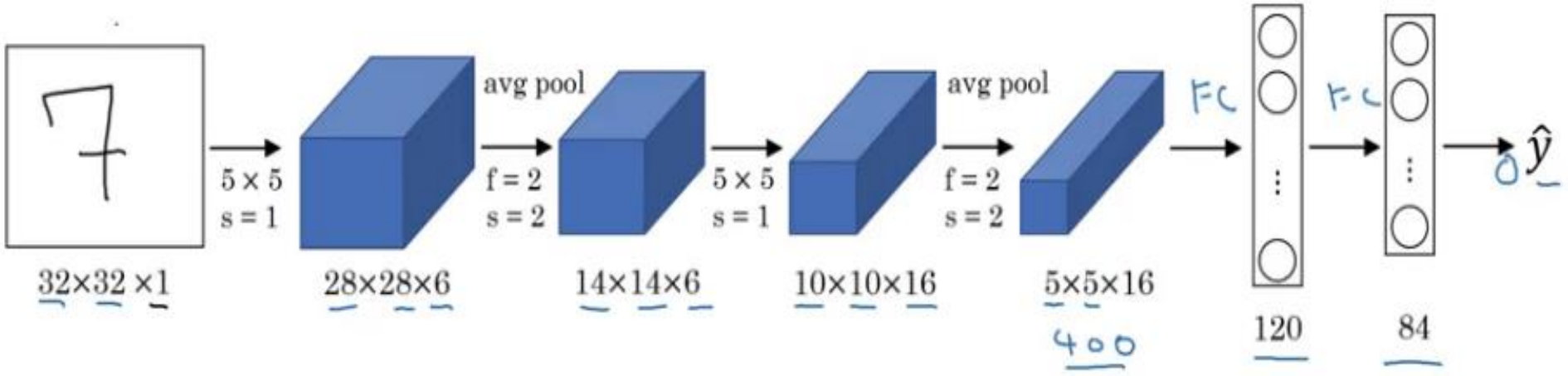
LeNet-5 Networks

The layers are laid out in the following order:

- The **First Convolutional Layer is made up of 6, 5 X 5 filters with a stride of 1.**
- The Second Layer is a 2 X 2 average-pooling or "sub-sampling" layer with a stride of 2.
- The Third Layer is similarly a Convolutional layer, with 16 ,5 X 5 filters and a stride of 1.
- The Fourth Layer is another 2 X 2 average-pooling layer with a stride of 2.
- The fifth layer is basically connecting the output of the fourth layer with a fully connected layer consisting of 120 nodes.
- The Sixth Layer is a similarly fully-connected layer with 84 nodes that derives from the outputs of the Fifth Layer's 120 nodes.
- The seventh layer consists of categorising the output of the previous layer into ten classifications based on the 10-digits it was trained to identify.
- It was one of the first effective digit-recognition algorithms for classifying handwritten digits.

LeNet-5 Networks

Let's start with LeNet-5- It is a7-level convolutional network.



It takes a grayscale image as input. Once we pass it through a combination of convolution and pooling layers, the output will be passed through fully connected layers and classified into corresponding classes. The total number of parameters in LeNet-5 are:

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

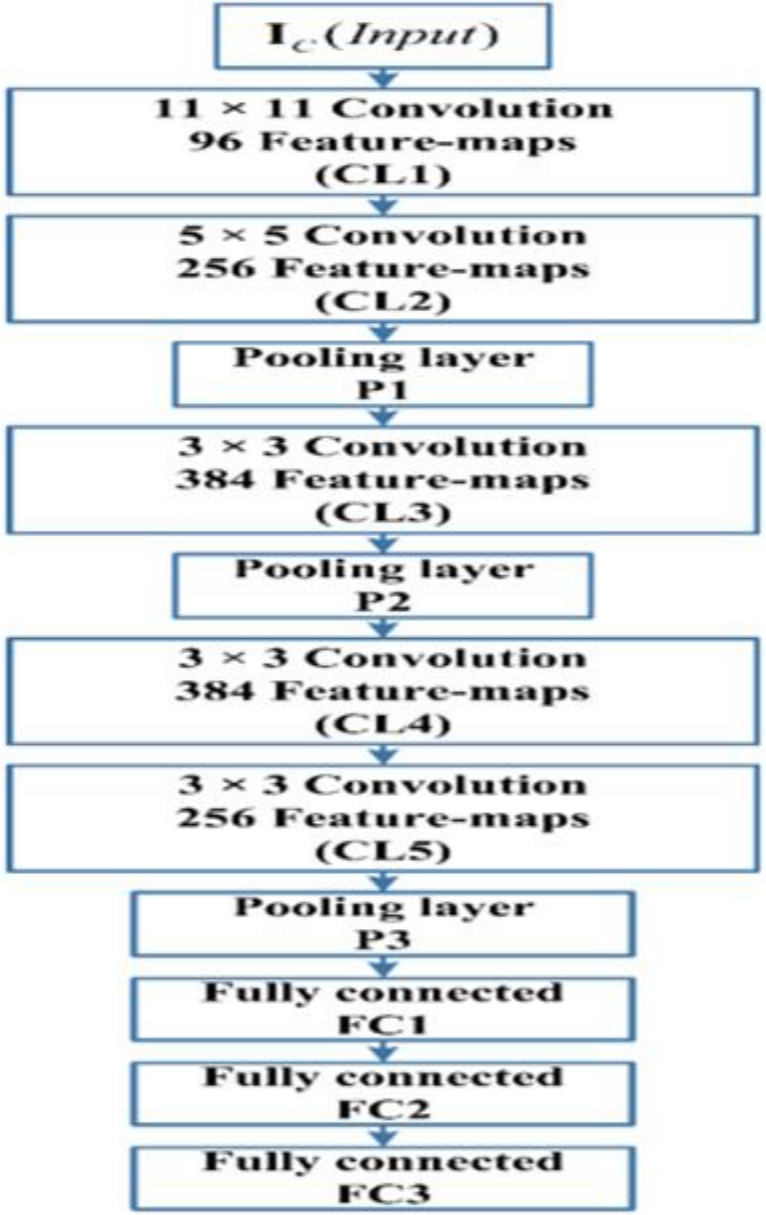
LeNet-5 Networks-Drawbacks

This network was **ineffective in terms of computing cost and accuracy when it came to processing huge images and categorizing among a large number of object classes.**

AlexNet Networks

- Starting with **an 11x11 kernel, Alexnet is built up of 5 conv layers.**
- For the three massive linear layers, it was the **first design to use max-pooling layers, ReLu activation functions, and dropout.**
- The network was **used to classify images into 1000 different categories.**
- The network is **similar to the LeNet Architecture, but it includes a lot more filters than the original LeNet, allowing it to categorise a lot more objects.**
- Furthermore, **it deals with overfitting by using "dropout" rather than regularization**
- **Regularization is a set of techniques that can prevent overfitting in neural networks and thus improve the accuracy of a Deep Learning model when facing completely new data from the problem domain.**

AlexNet Networks-Basic layout of AlexNet architecture



AlexNet Networks-Basic layout of AlexNet architecture

- **First, a Convolution Layer (CL) with 96 11 X 11 filters and a stride of 4.**
- After that, a Max-Pooling Layer (M-PL) with a filter size of 3 X 3 and a stride of 2 is applied.
- **A CL of 256 filters of size 5 X 5 and stride = 4 is used once more.**
- Then an M-PL with a filter size of 3 X 3 and a stride of 2 was used.
- **A CL of 384 filters of size 3 X 3 and stride = 4 was used.....2**
- Again, a CL of 384 filters of size 3 X 3 and stride = 4.
- **A CL of 256 filters of size 3 X 3 and stride = 4 was used again.**
- Then an M-PL with a filter size of 3 X 3 and a stride of 2 was used.
- When the output of the last layer is transformed to an input layer, as in the Fully Linked Block**, it has 9261 nodes**, all of which are **fully connected to a hidden layer with 4096 nodes.**
- The first hidden layer is once again fully linked to a 4096-node hidden layer.

VGG-16 Networks

- **VGG is a convolutional neural network design** that has been around for a long time.
- It was based on a study on how to make such networks more dense.
- **Small 3 x 3 filters are used** in the network.
- The network is otherwise defined by its simplicity, with simply pooling layers and a fully linked layer as additional components.
- In comparison to AlexNet, **VGG was created with 19 layers deep to replicate the relationship between depth and network representational capability.**

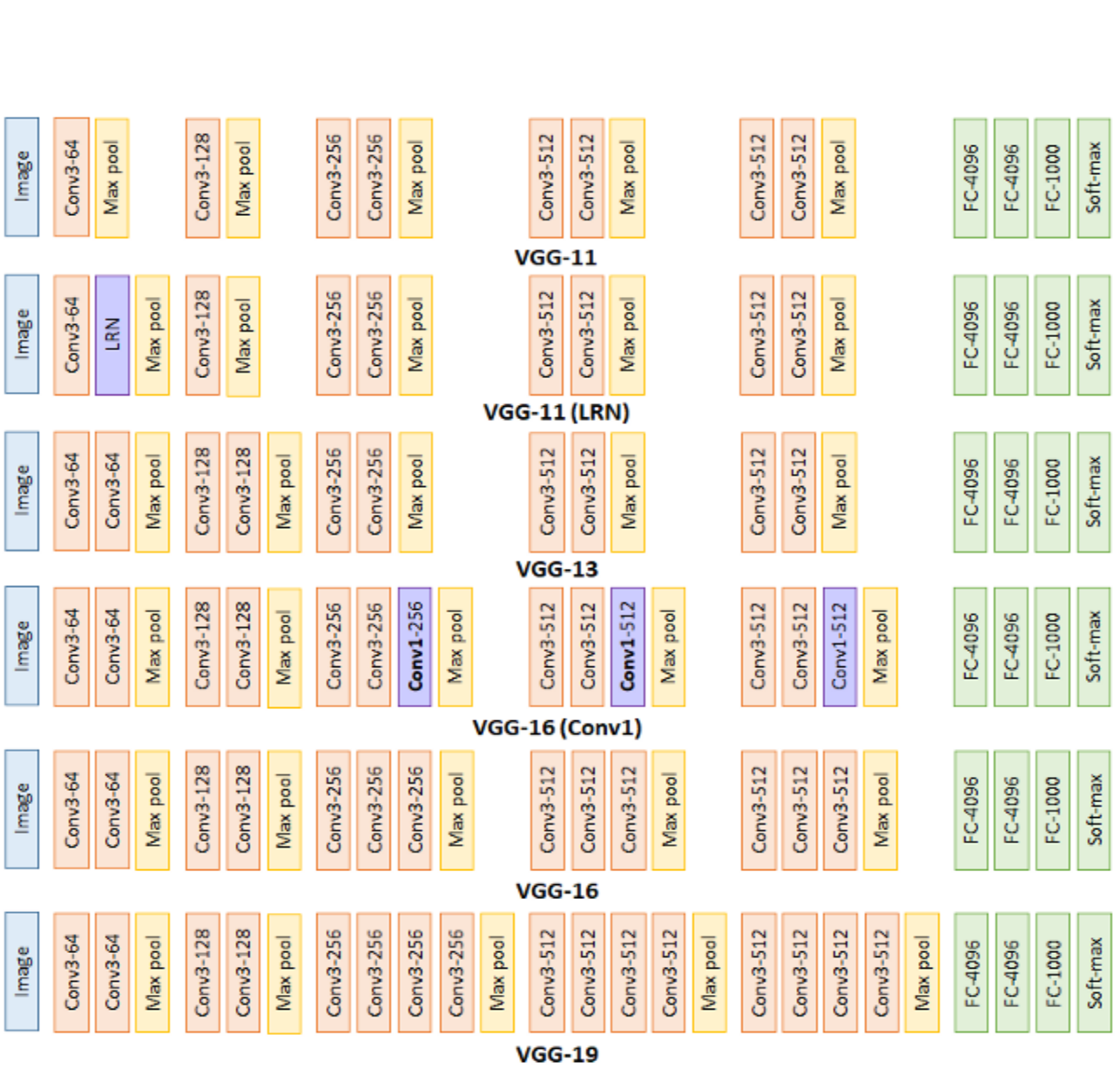
VGG-16 Networks

Based on these observations, **VGG replaced the 11x11 and 5x5 filters with a stack of 3x3 filters, demonstrating that the simultaneous placement of small size (3x3) filters may provide the effect of a big size filter (5x5 and 7x7).**

By lowering the number of parameters, the usage of tiny size filters gives an additional benefit of **low computing complexity.**

These discoveries helped in a new research trend at CNN, which is to work with lower size filters.

VGG-16 Networks



	Number of Parameters (millions)	Top-5 Error Rate (%)
VGG-11	133	10.4
VGG-11 (LRN)	133	10.5
VGG-13	133	9.9
VGG-16 (Conv1)	134	9.4
VGG-16	138	8.8
VGG-19	144	9.0

VGG-16 Networks

The underlying idea behind VGG-16 was to use a much simpler network where the focus is on having convolution layers that have 3 X 3 filters with a stride of 1 (and always using the same padding).

The max pool layer is used after each convolution layer with a filter size of 2 and a stride of 2. Let's look at the architecture of VGG-16:

As it is a bigger network, the number of parameters are also more.

- **Parameters:** 138 million

These are three classic architectures.

ResNet Networks

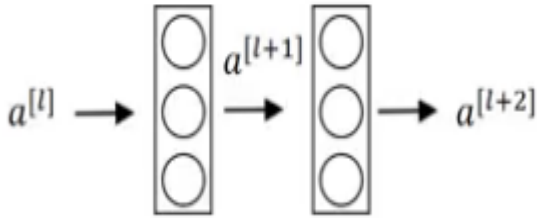
Training deep networks can lead to problems like **vanishing and exploding gradients**.

How do we deal with these issues? We can use **skip connections where we take activations from one layer and feed it to another layer that is even more deeper in the network.**

There are residual blocks in ResNet which help in training deeper networks.

Residual Blocks

The general flow to calculate activations from different layers can be given as:

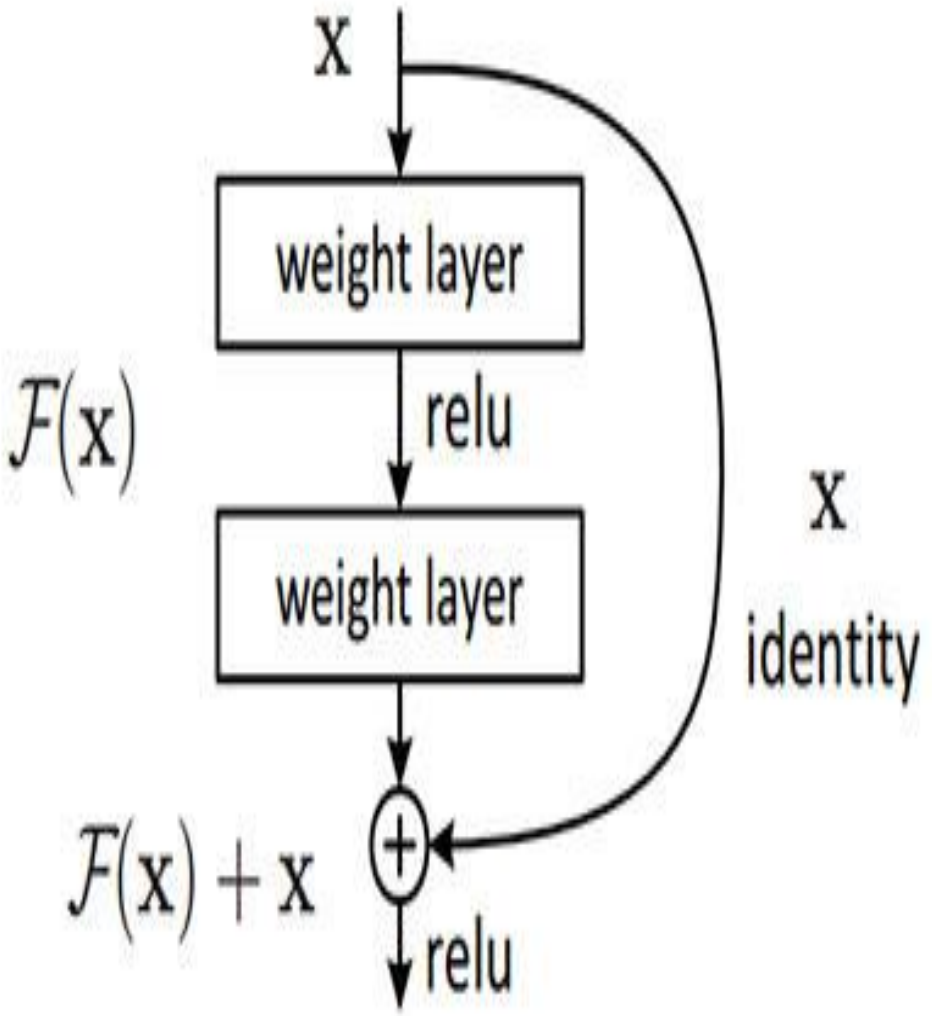


This is how we calculate the activations $a^{[l+2]}$ using the activations $a^{[l]}$ and then $a^{[l+1]}$. $a^{[l]}$ needs to go through all these steps to generate $a^{[l+2]}$:

ResNet Networks

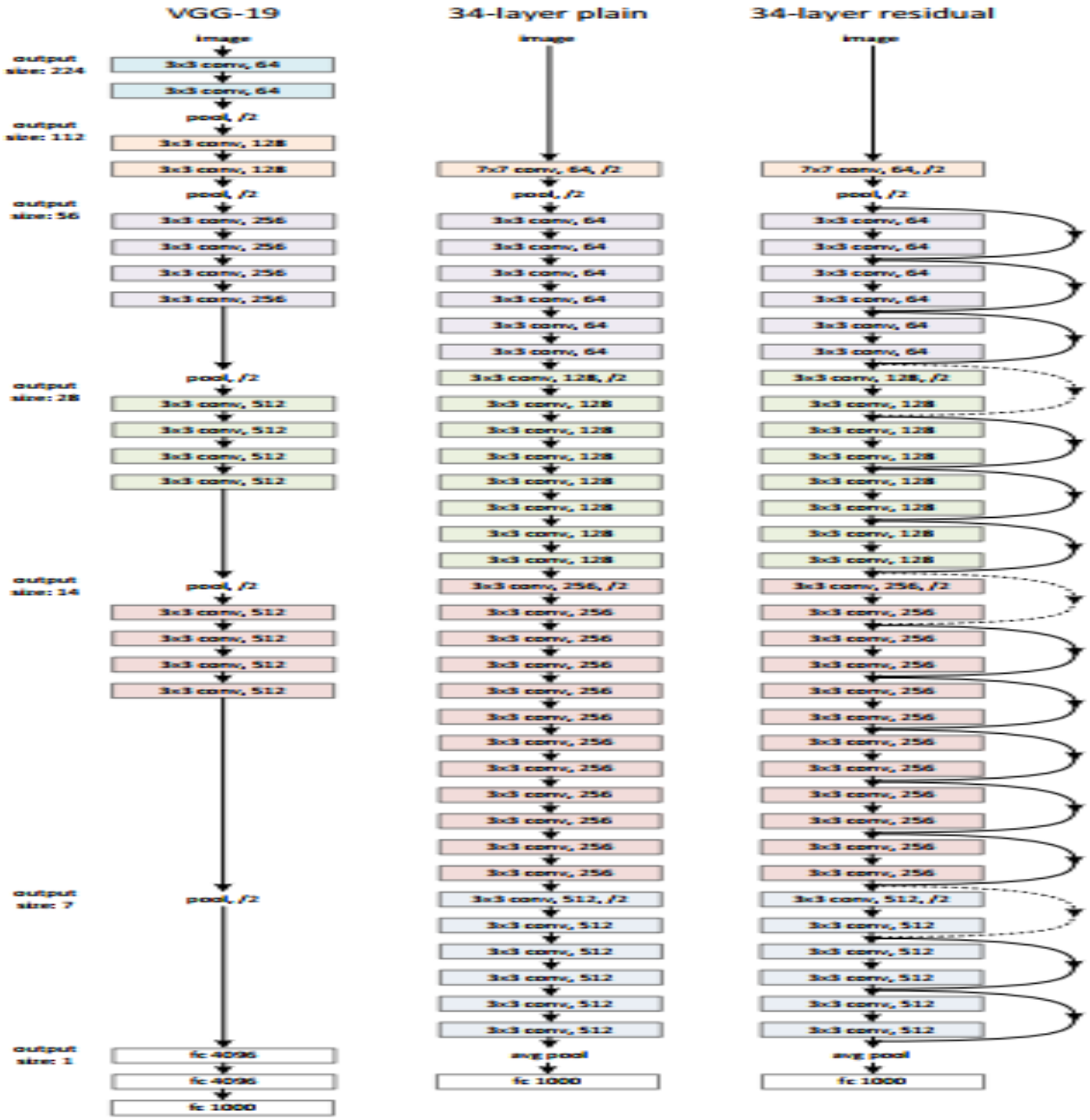
- *ResNet is a well-known deep learning model that was first introduced in a paper by Shaoqing Ren, Kaiming He, Jian Sun, and Xiangyu Zhang.*
- *In 2015, a study titled "Deep Residual Learning for Image Recognition" was published.*
- *ResNet is one of the most widely used and effective deep learning models to date.*
- **ResNets are made up of what's known as a residual block.**
- This is **built on the concept of "skip-connections"** and **uses a lot of batch-normalization** to let it train **hundreds of layers successfully without sacrificing speed over time.**
- **Batch normalization is a technique used in deep learning to normalize the inputs of each layer in a neural network.**

ResNet Networks



Skip connections or *shortcuts* are used to jump over some layers

ResNet Networks-Complete Architecture



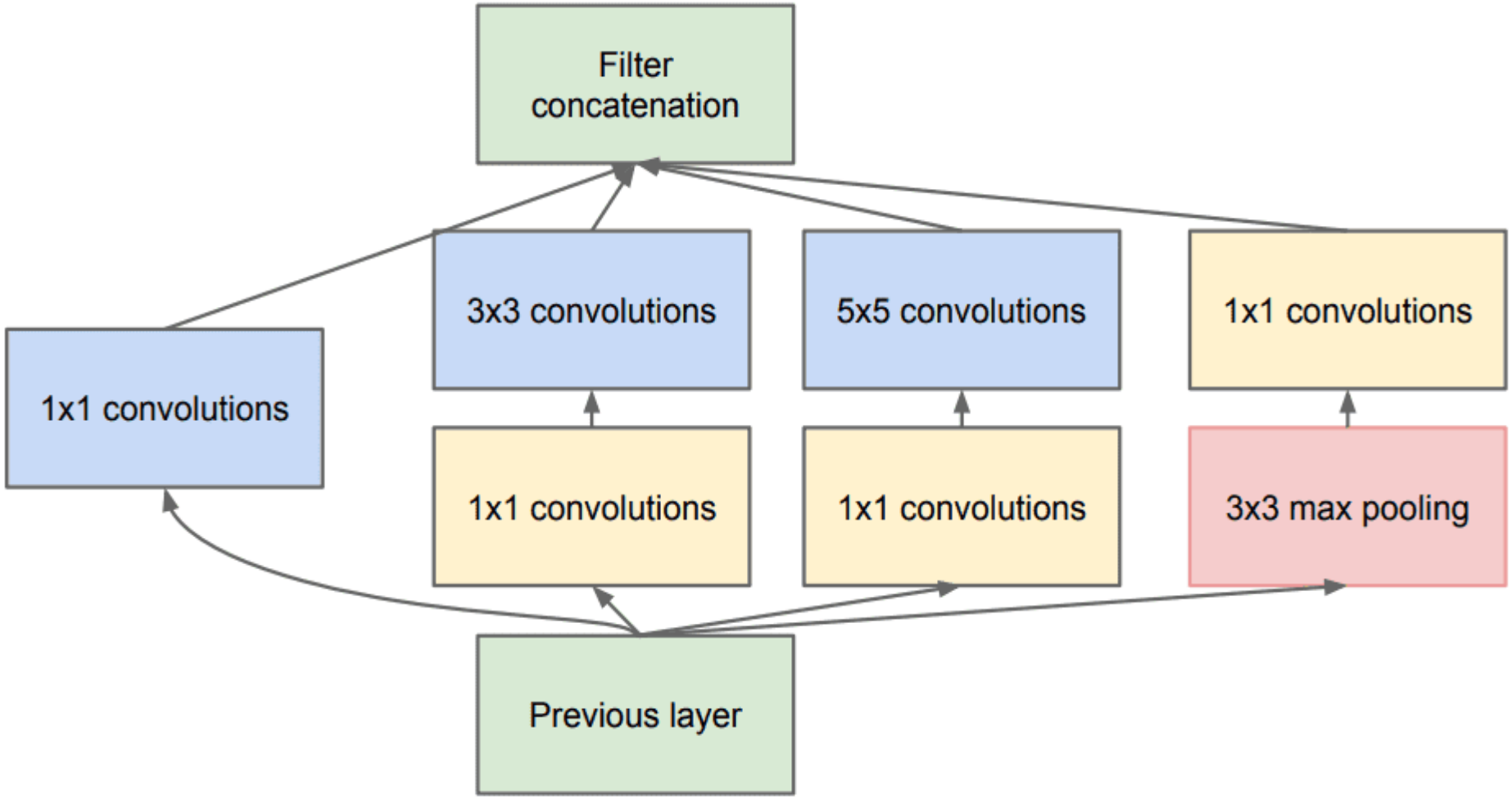
GoogleNet / Inception

- The **ILSVRC 2014 competition was won by the GoogleNet or Inception Network**, which had **a top-5 error rate of 6.67 percent, which was virtually human level performance.**
- Google created the model, which incorporates an improved implementation of the original LeNet design. This is based on the inception module concept.
- GoogLeNet is a variation of the Inception Network, which is a 22-layer deep convolutional neural network.
- GoogLeNet is now utilised for a variety of computer vision applications, including face detection and identification, adversarial training, and so on.

ILSVRC-ImageNet Large Scale Visual Recognition Challenge 2014

GoogleNet / Inception

The inception Module looks like this:



GoogleNet / Inception

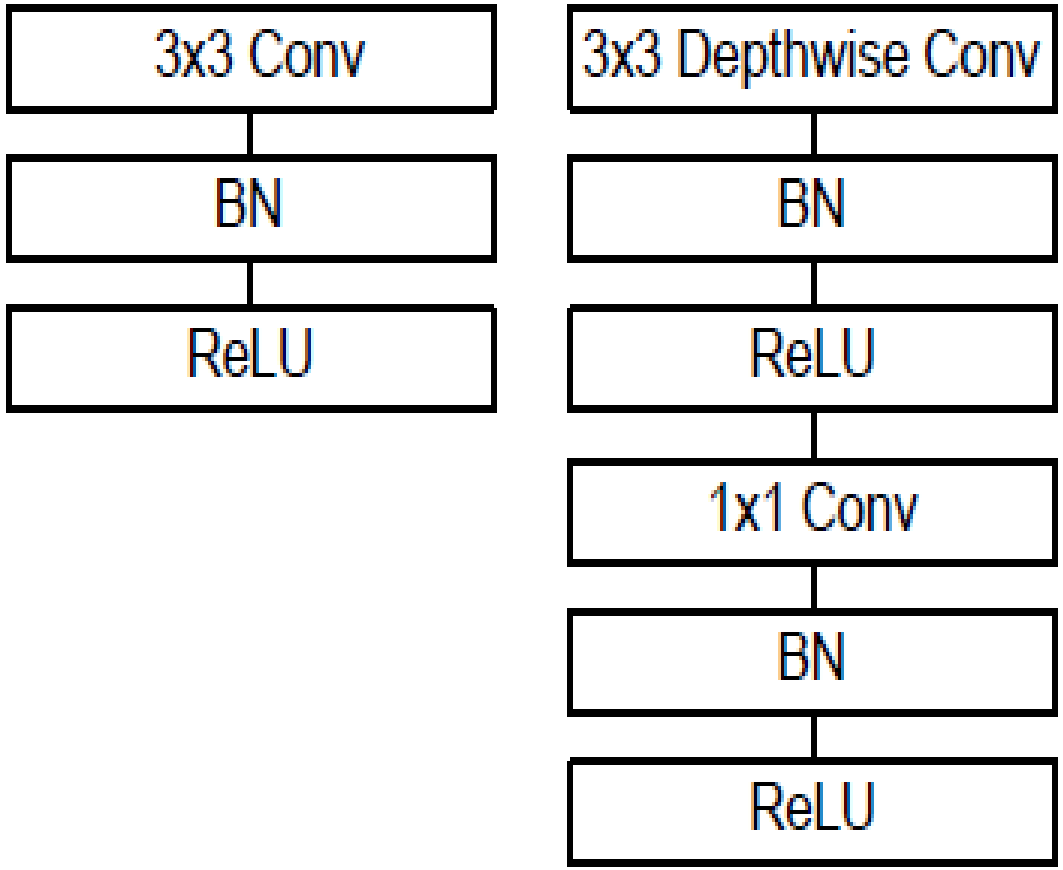
- The InceptionNet/GoogleLeNet design is made up of nine inception modules stacked on top of each other, with max-pooling layers between them (to halve the spatial dimensions).
- It is made up of 22 layers (27 with the pooling layers).
- After the last inception module, it employs global average pooling.

MobileNetV1

- Nine inception modules are placed on top of each other in the InceptionNet/GoogleLeNet architecture, with max-pooling layers between them (to halve the spatial dimensions).
- **There are 22 layers in all (27 with the pooling layers).**
- It uses **global average pooling after the final inception module.**
- The **MobileNet model is built on depthwise separable convolutions, which are a type of factorised convolution that divides a regular convolution into a depthwise convolution and a pointwise convolution.**
- The **depthwise convolution used by MobileNets applies a single filter to each input channel.**
- The **depthwise convolution's outputs are then combined using an 11 convolution by the pointwise convolution.**
- In one step, a conventional convolution filters and mixes inputs to create a new set of outputs.
- The depthwise separable convolution divides this into two layers: one for filtering and the other for combining.

MobileNetV1-Architecture

- Except for the first layer, which is a complete convolution, the MobileNet structure is based on depthwise separable convolutions, as discussed in the preceding section.
- With the exception of the last fully connected layer, which has no nonlinearity and feeds into a softmax layer for classification, all layers are followed by a batchnorm and ReLU nonlinearity



Comparison

Model	Layer	Parameter [M]	Network size[MB]
Alexnet	8	61	227
GoogLeNet	22	7	27
Inception-v3	48	23.9	89
VGG-16	16	138	515
ResNet18	18	25.6	96
ResNet50	50	44.6	167


ThankYou
Merci
Salamat
Kop
Tack
Hvala
Gra
Xie
kun
Dankie
Kruthagnathalu
Arigatou
Dhonnobaad
Asante
Dhan
daa
Hain
ederim
Shukriya
Khopjai
Dhanyavad
Gomapsupnida
Kun
Danke
dank
Terima
Enkosi
Euxaristo
Fyrir
Blagodariya
Nandree
Gracias
Grazzi
raibh
krap
Dakujem
Daw
Waad
Dhanyavadaalu
Takk
Dank
Gamsahapnida
Te ekkür
Dekuju/Dekujeme
Diolch
Kiitos
Sheun
umesc
Shnorhakalutiun
Kasih
Mamnoon
Todah
Shokriya
Ngiyabonga
Cam
Dziękuję
Shokrun
Spaas
Mul
Ačı
or
al
Dhanyavaad
Go
Grazie
Falemiinderit
Dr