

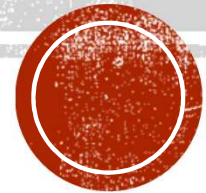
CONVOLUTION NEURAL NETWORK

By Eng: Khaled Mohammed Abdelgaber

Tele: +201150882331

E-mail: Khaled.edu.engineer@gmail.com

References: Deep Learning for Vision Systems by Mohamed Elgendy



WHAT ARE FEATURES?

- Features are parts or patterns of an object in an image that help to identify it.
- For example — a square has 4 corners and 4 edges, they can be called features of the square, and they help us humans identify it's a square.
- Features include properties like corners, edges, regions of interest points, ridges, etc.
- As shown in the image the yellow points show the features detected using a technique called Harris Detection.



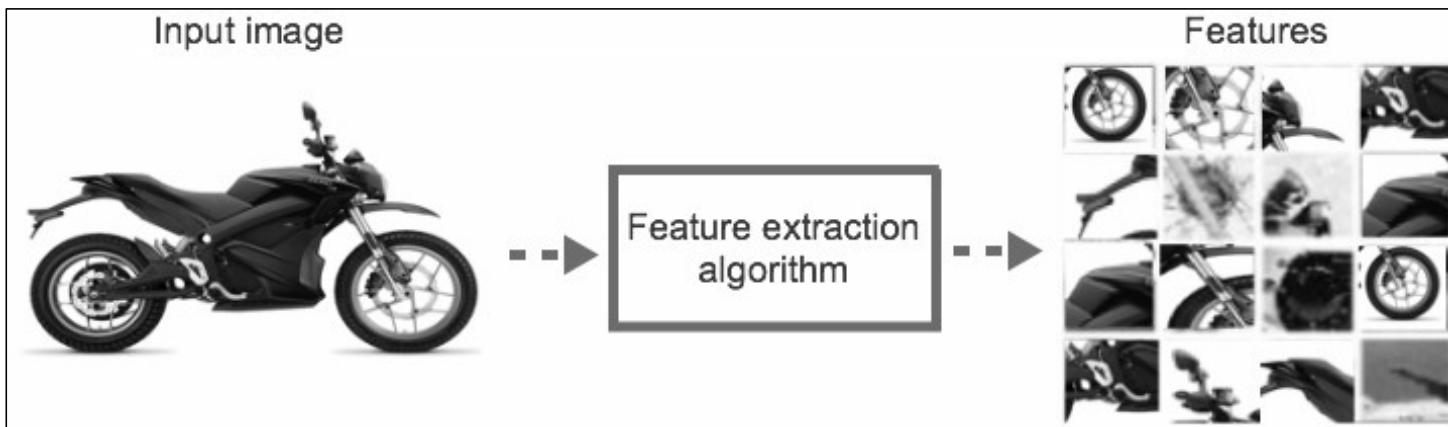
WHAT IS A FEATURE IN COMPUTER VISION?

- In computer vision, a feature is a measurable piece of data in your image that is unique to this specific object.
- It maybe a distinct color in an image or a specific shape such as a line, edge, or an image segment.
- A good feature is used to distinguish objects from one another.
- **For example**, if I give you a feature like a wheel, and ask you to guess whether the object is a motorcycle or a dog. What would your guess be? A motorcycle. Correct!
- In this case, the wheel is a strong feature that clearly distinguishes between motorcycles and dogs.
- **However**, if I give you the same feature (a wheel) and ask you to guess whether the object is a bicycle or a motorcycle.
- In this case, this feature is not strong enough to distinguish between both objects.
- Then we need to look for more features like a mirror, license plate, maybe a pedal that collectively describe an object.



CONT.

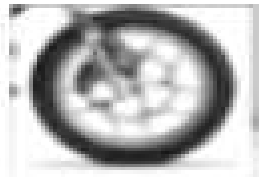
- In machine learning projects, we want to transform the raw data (image) into a *features vector* to show to our learning algorithm to learn the characteristics of the object.
- we need to know that the extraction algorithm produces a **vector** that contains a list of features. This is called **features vector** that is a 1D array that makes a robust representation of the object.



NOTES

- It is important to call out that the image above reflects features extracted from just one motorcycle. A very important characteristic of a feature is repeatability. As in the feature should be able to detect the motorcycles in general not just this specific one. So, in real world problems, the feature will not be an exact copy of the piece in the input image
- What makes a good (useful) feature ?
 - Machine learning models are only as good as the features you provide. That means coming up with good features is an important job in building ML models.

Feature after looking
at one image

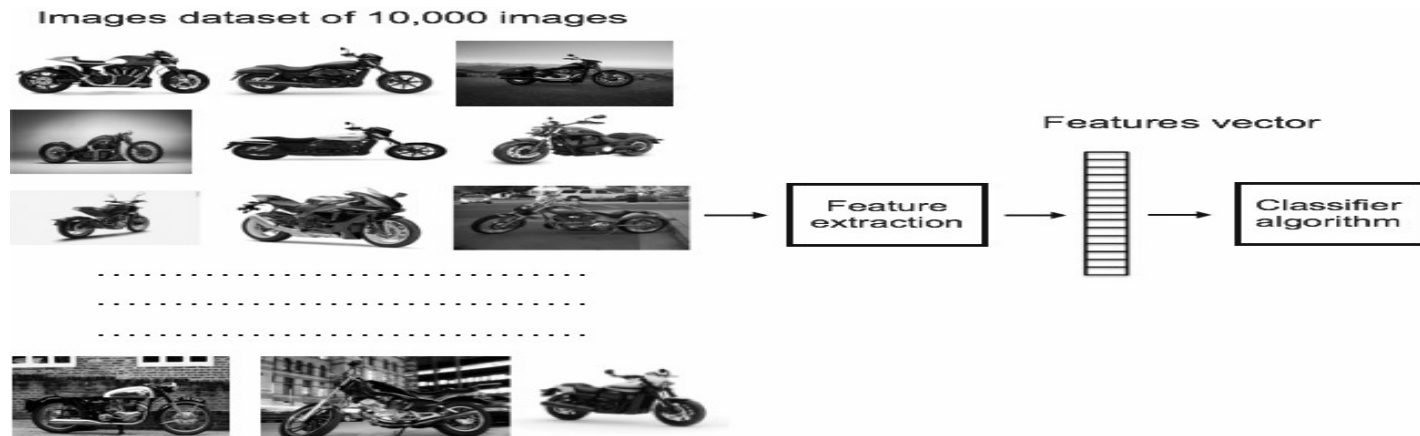


Feature after looking
at 1000s of images



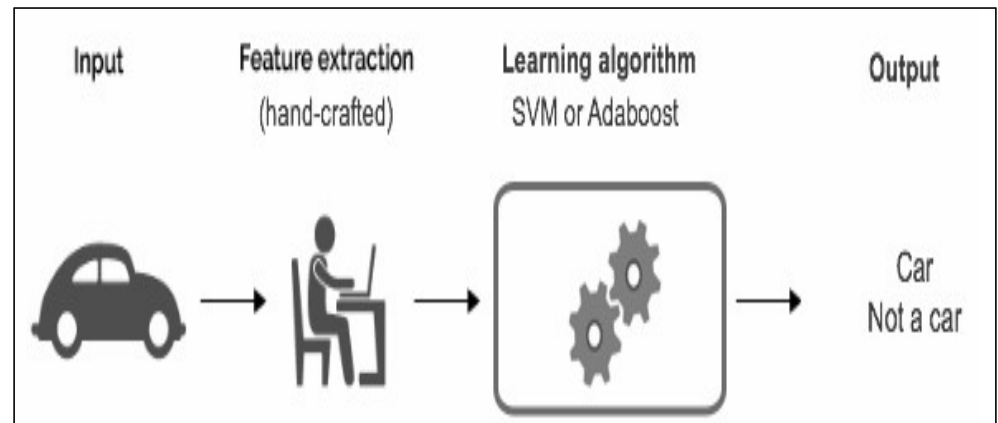
WHY USE FEATURES?

- The input image has too much extra information that is not necessary for classification.
- Therefore, the first step after preprocessing the image is to simplify the image by extracting the important information contained in the image and throwing away non-essential information.
- By extracting important colors or image segments, we can transform complex and large image data into smaller sets of features.
- This makes the task of classifying images based on their features done simpler and faster.
- Suppose we are given a dataset of 10,000 images of motorcycles each of 1,000 width x 1,000 height. Some images have solid backgrounds and others have busy backgrounds of unnecessary data.
- When these thousands of images are fed to the feature extraction algorithms, we lose all the unnecessary data that are not important to identify motorcycles and we only keep a consolidated list of useful features that can then be fed directly to the classifier.
- This process is a lot simpler than having the classifier look at a dataset of 10,000 images to learn the properties of motorcycles.

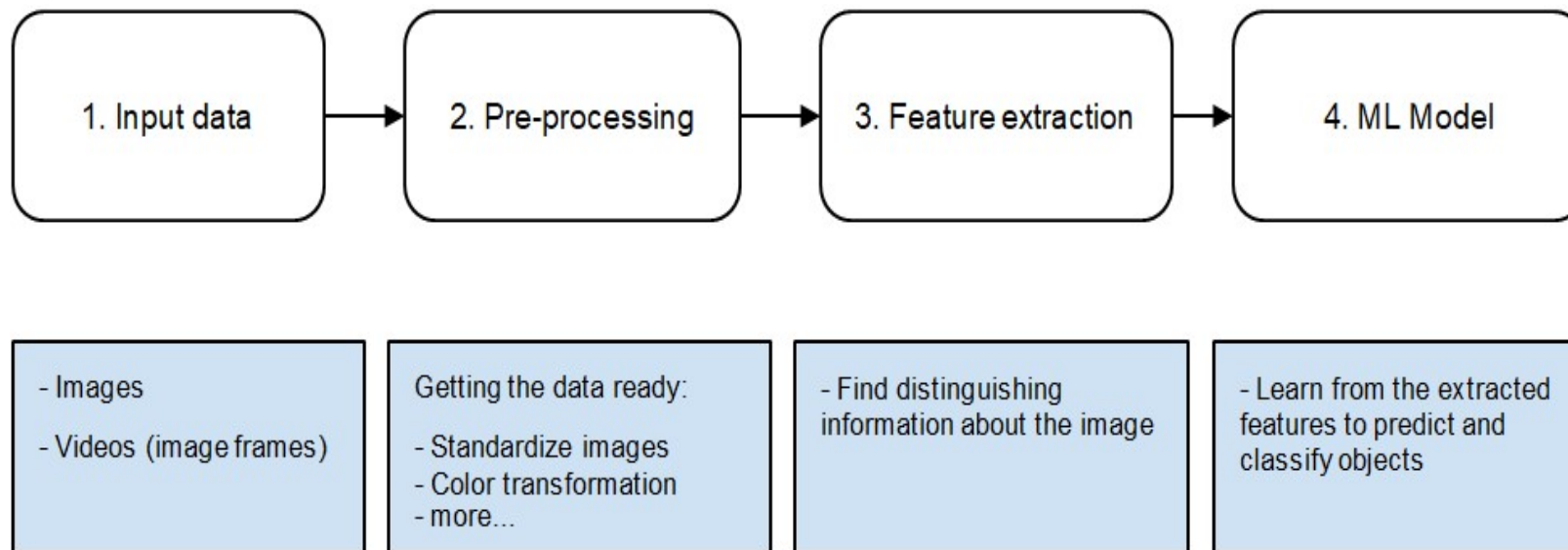


TRADITIONAL MACHINE LEARNING USES HAND-CRAFTED FEATURES

- In traditional machine learning problems, we spend a good amount of time in **manual** features selection and engineering.
- In this process we rely on our domain knowledge (or partnering with domain experts) to create features that make machine learning algorithms work better.
- We then feed the produced features to a classifier like Support Vector Machines (SVM) or Adaboost to predict the output.
- Some of the hand-crafted feature sets are:
 - Histogram of Oriented Gradients (HOG)
 - Haar Cascades
 - Scale-Invariant Feature Transform (SIFT)
 - Speeded Up Robust Feature (SURF)

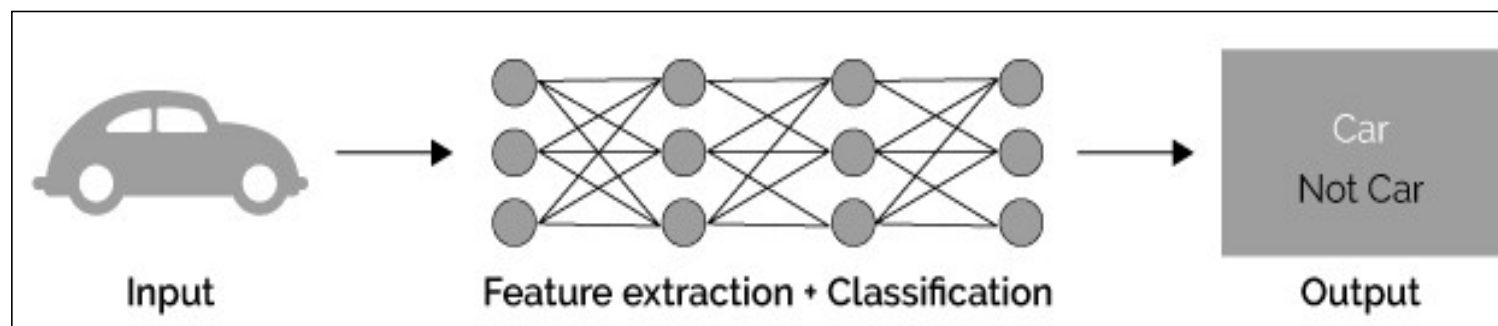


TRADITIONAL ML FOR IMAGE CLASSIFICATION

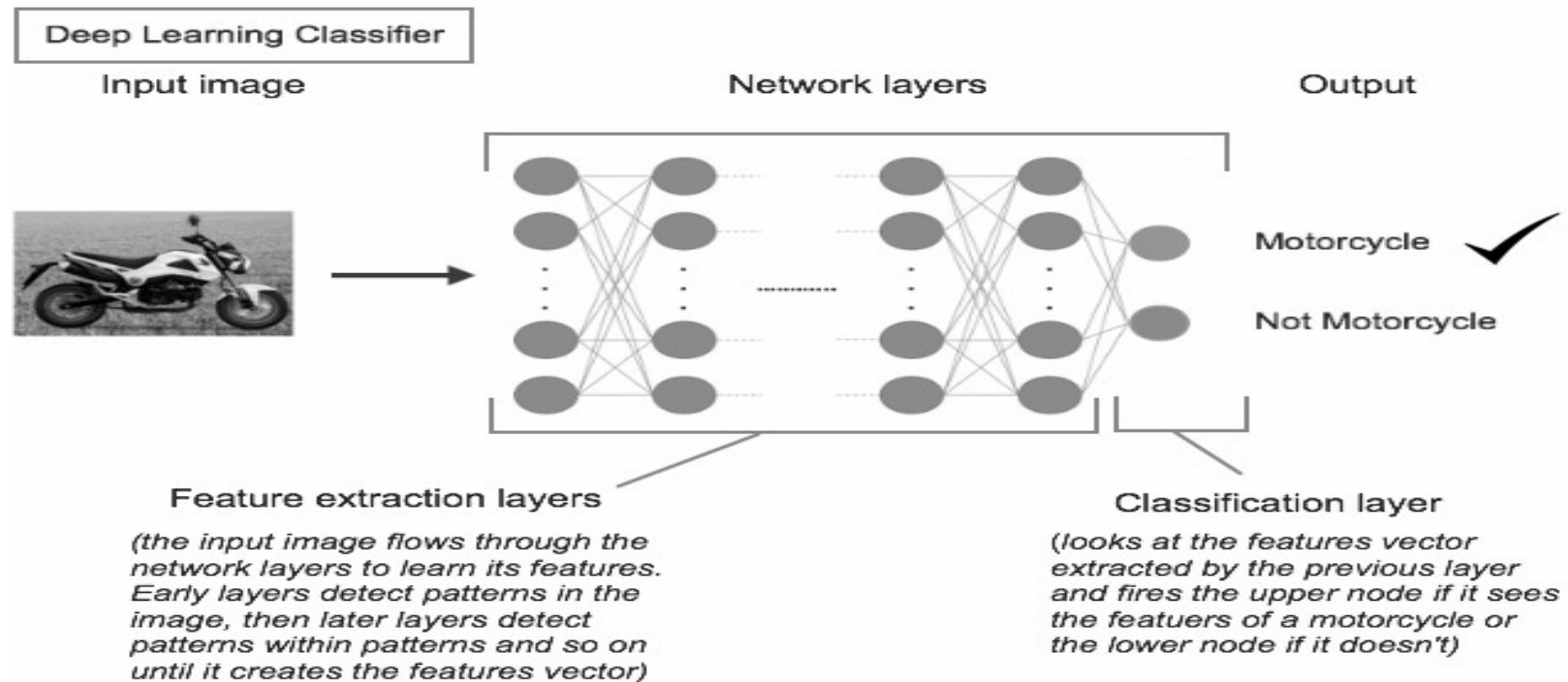


DEEP LEARNING AUTOMATICALLY EXTRACT FEATURES

- In deep learning however, we do not need to manually extract features from the image. The network extracts features **automatically** while and learns their importance on the output by applying weights to its connections.
- You just feed the raw image to the network and while it passes through the network layers it identifies patterns within the image to create features.
- Neural networks can be thought of as feature extractors + classifiers that are end-to-end trainable as opposed to traditional ML models that use hand-crafted features.

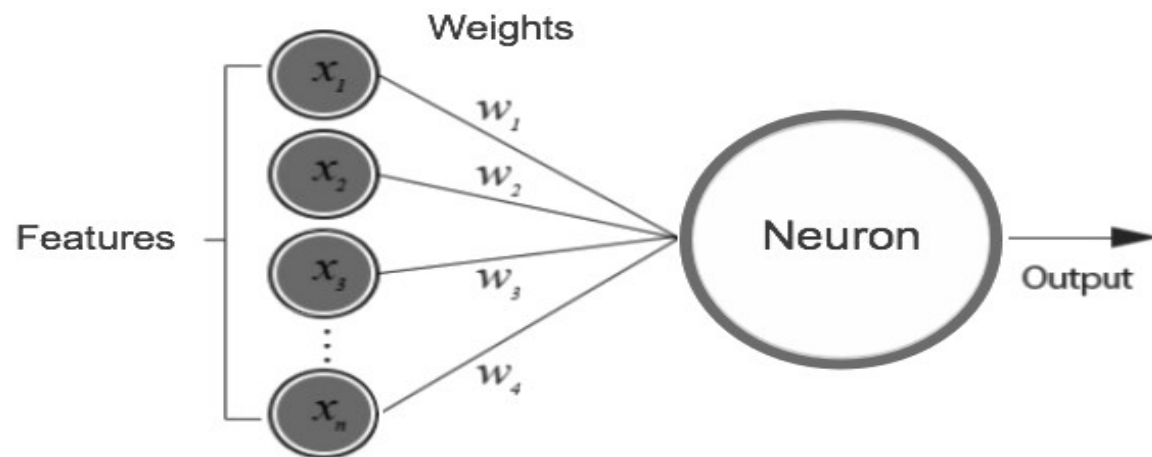


DEEPLARNING FOR CLASSIFICATION

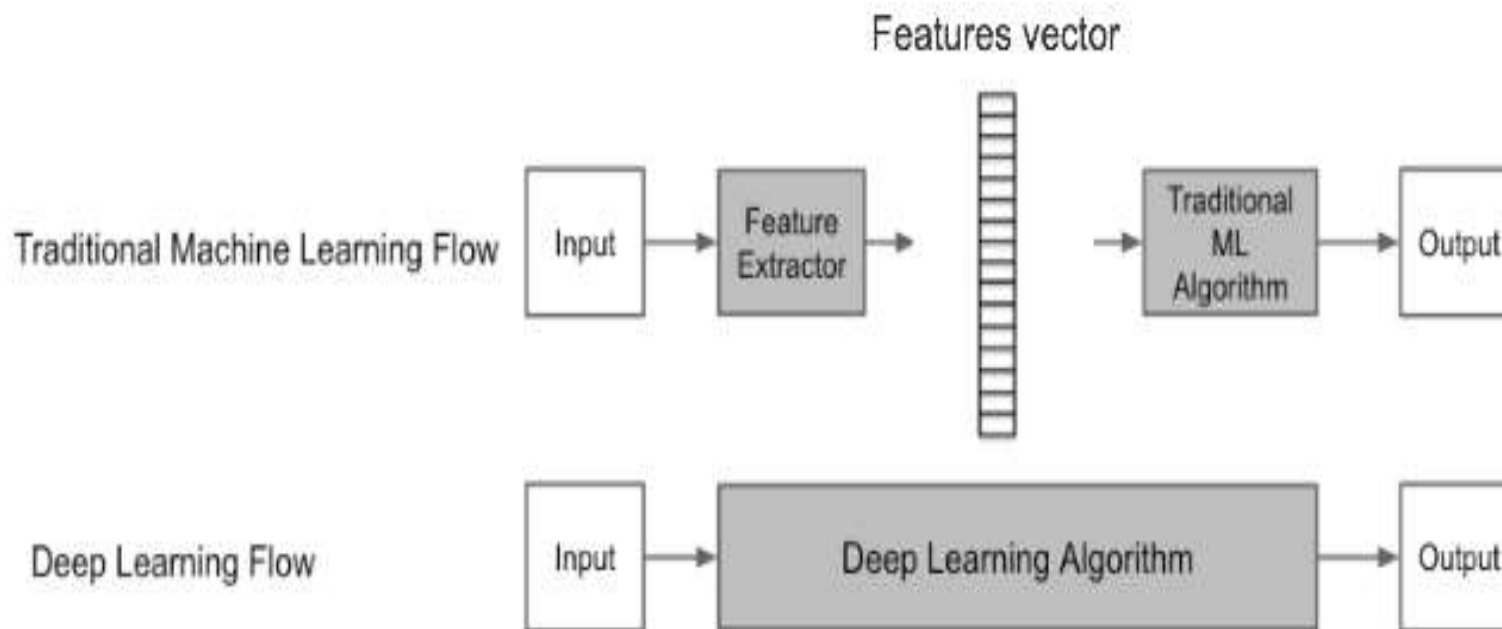


HOW DOES NEURAL NETWORKS DISTINGUISH USEFUL FEATURES FROM THE NON-USEFUL FEATURES?

- You might get the impression that neural networks only understands the useful features but that's not entirely true.
- Neural Networks scoop all the features available and give them random weights.
- During the training process it adjusts these weights to reflect their importance and how they should impact the output prediction.
- The patterns with the highest appearance frequency will have higher weights and in turn are considered more useful features.
- Whereas, features with lowest weights will have very low impact on the output. This is called the learning process.



ML VS DL



EXAMPLES OF TRADITIONAL FEATURE DETECTION TECHNIQUES

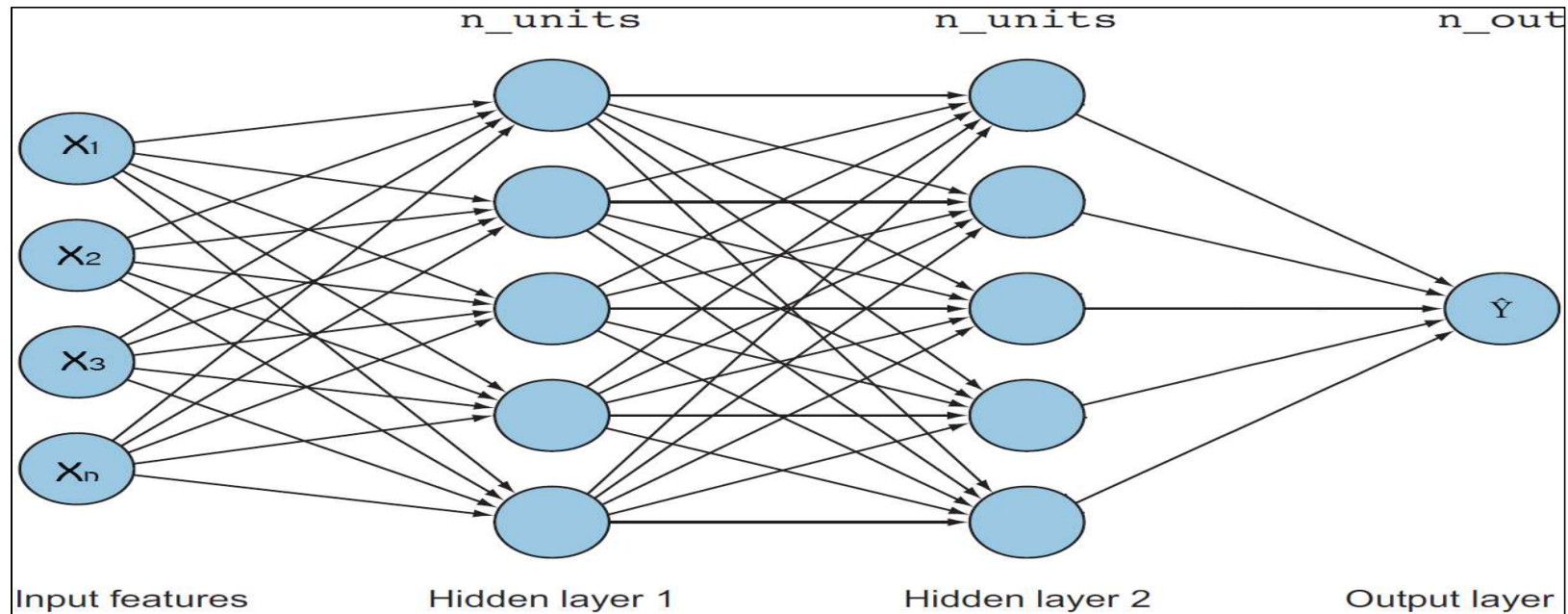
- **Harris Corner Detection** — Uses a Gaussian window function to detect corners. ([read more](#))
- **Shi-Tomasi Corner Detector** — The authors modified the scoring function used in Harris Corner Detection to achieve a better corner detection technique ([read more](#))
- **Scale-Invariant Feature Transform (SIFT)** — This technique is scale invariant unlike the previous two. ([read more](#))
- **Speeded-Up Robust Features (SURF)** — This is a faster version of SIFT as the name says. ([read more](#))
- **Features from Accelerated Segment Test (FAST)** — This is a much more faster corner detection technique compared to SURF. ([read more](#))
- **Binary Robust Independent Elementary Features (BRIEF)** — This is only a feature descriptor that can be used with any other feature detector. This technique reduces the memory usage by converting descriptors in floating point numbers to binary strings. ([read more](#))
- **Oriented FAST and Rotated BRIEF (ORB)** — SIFT and SURF are patented and this algorithm from OpenCV labs is a free alternative to them, that uses FAST keypoint detector and BRIEF descriptor. ([read more](#))



IMAGE CLASSIFICATION USING MLP

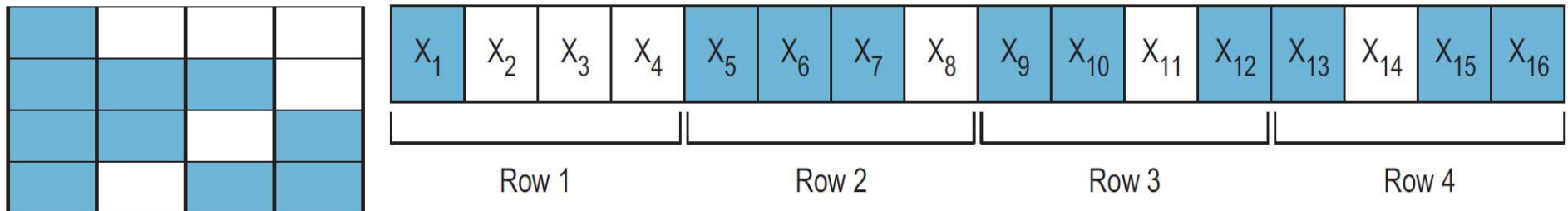


IMAGE CLASSIFICATION USING MLP



INPUT LAYER

- We had a 1D vector of features ($x_1, x_2, x_3, \dots, x_n$) that are fed to the input layer of the network. But to allow the network to understand 2D images, we need to pre-process the images first before feeding them to the network. [need to be flatten]
- Assume we have an image of 28 pixels wide x 28 pixels height. This image is seen by the computer as a 28x28 matrix with pixel values range from 0 to 255 (0 for black and 255 for white and the range in between for gray scale).
- Since MLPs only take the input as a 1D vector with dimensions (1, n), they cannot take the raw 2D image matrix with dimensions (x, y). To fit the image in the input layer, we first need to transform our image into one large vector with the dimensions (1, n) that contains all the pixels' values of the image. This process is called *image flattening*. In this example, the total number (n) of pixels in this image is $28 \times 28 = 784$.
- the input layer in this example will have a total of 784 nodes. $x_1, x_2, x_3, \dots, x_{784}$.



HIDDEN LAYERS AND OUTPUT LAYER

- the neural network can have one or more hidden layers (technically, as many as you want). Each layer has one or more neurons (also technically as many as you want). Your main job, as a neural network engineer, is to design these layers.
- And don't forget to add the activation function (for example ReLU) for each hidden layer.
- The output layer is pretty straight forward. In classification problems, the number of nodes in the output layer should be equal to the **number of classes** that you are trying to classify. In this problem, we are classifying 10 digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9). Then we will need to add one last Dense layer that contains 10 nodes.

```
# same as before, we start with importing keras library
from keras.models import Sequential
# here we import a layer called Flatten to convert the image matrix into a vector
from keras.layers import Flatten

# define the model
model = Sequential()
# add the flatten layer - also known as the input layer
model.add( Flatten(input_shape = (28,28) )
```

```
# import Dense layer
from keras.layers import Dense

# add two Dense layers with 512 nodes each
model.add(Dense(512, activation = 'relu'))
model.add(Dense(512, activation = 'relu'))

# add one output Dense layer with 10 nodes
model.add(Dense(10, activation = 'softmax'))
```



DRAWBACKS OF MLPs IN PROCESSING IMAGES

1. **Flattening the image to a 1D vector input leads to losing the spatial features of 2D images:**

- CNNs on the other hand, do not require flattening the image. We can feed the raw image matrix of pixels to our network. This will allow the CNN to understand that pixels that are close to each other are heavily related than the pixels that are far apart.

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

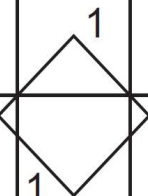
Input vector = [1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]



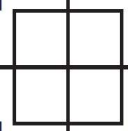
CONT. SPATIAL INFORMATION LOSS

- Input vector = [1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
- When the training is complete, the network will learn to identify a square *only* when the input nodes x_1 , x_2 , x_5 , and x_6 are fired. But what happens when we have new images with square shapes but located in different areas in the image?

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

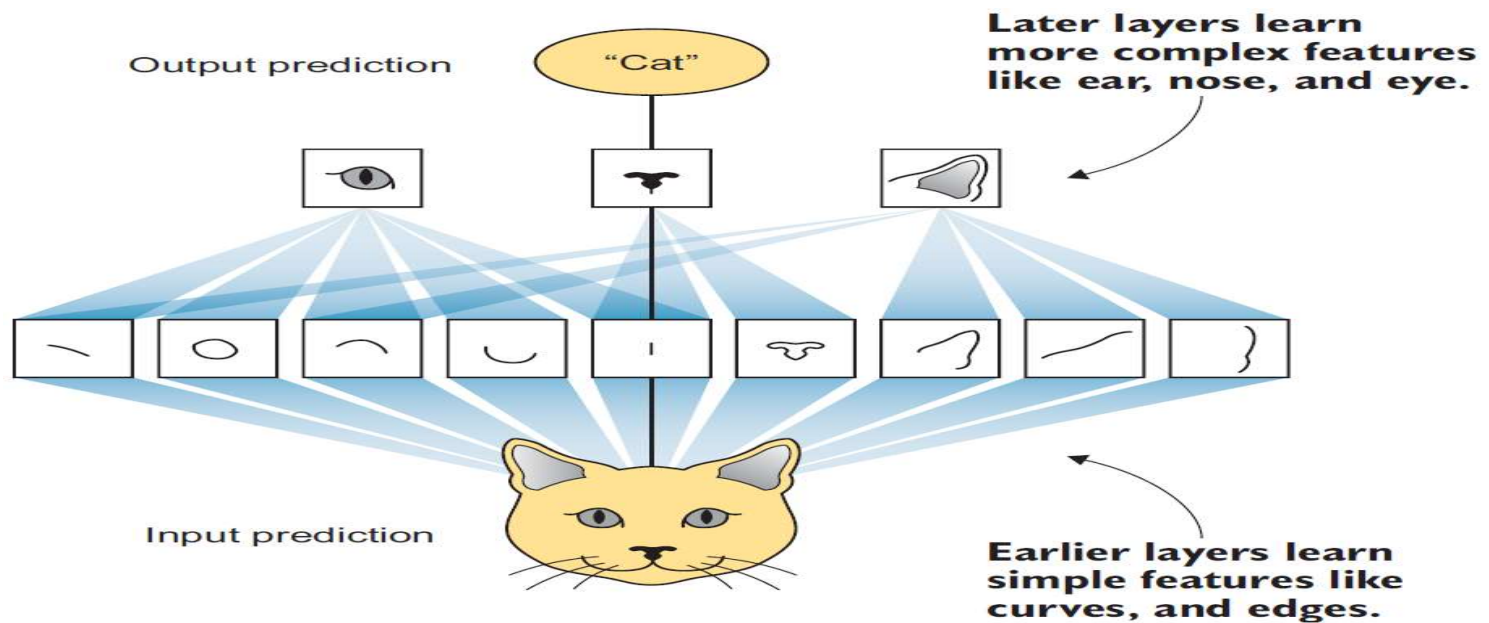


0	0	0	0
0	0	0	0
0	0	1	1
0	0	1	1



CONT. SPATIAL INFORMATION LOSS

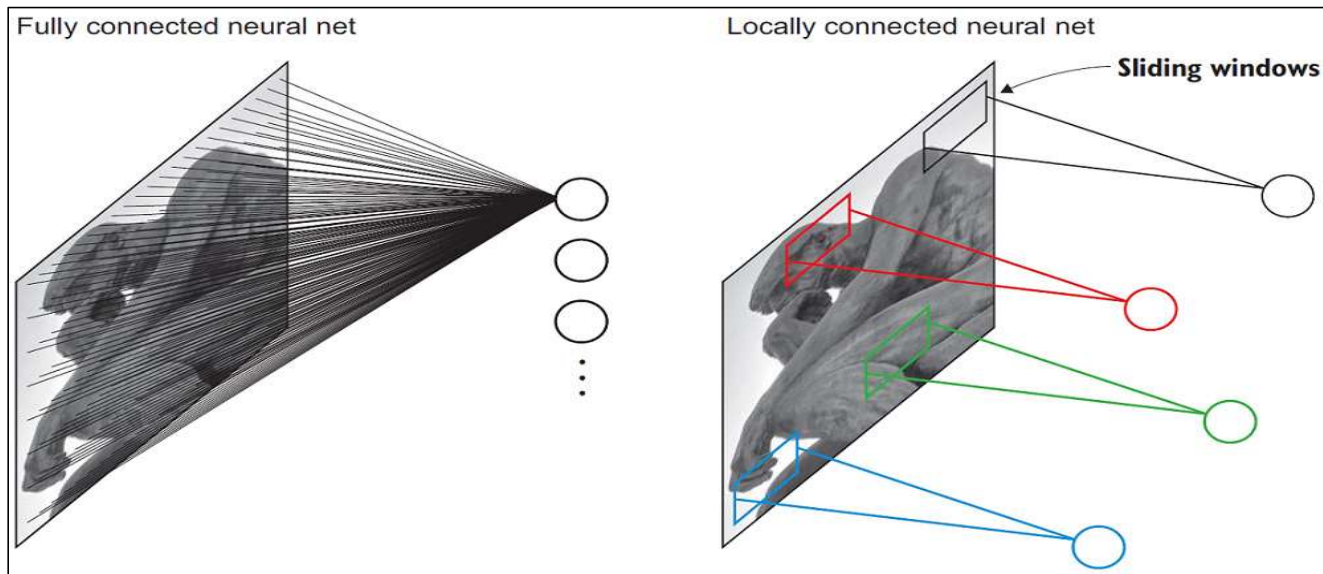
- Another example of feature learning: if we want to teach the network to recognize cats, ideally we want the neural network to learn the full shape of the cat features regardless to where they appear on the image (ears, nose, eyes, etc.).
- This only happens when the network looks at the image as a set of pixels that when close to each other they are heavily related.



SECOND DRAWBACK OF MLP

2. **Fully connected (dense) layers :**

- Increase the number of parameters
- CNNs on the other hand are *locally connected* layers. Where there nodes are connected to only a small subset of the previous layers' nodes. Locally connected layers use far fewer parameters than a densely connected layer as we will see when we discuss CNNs.

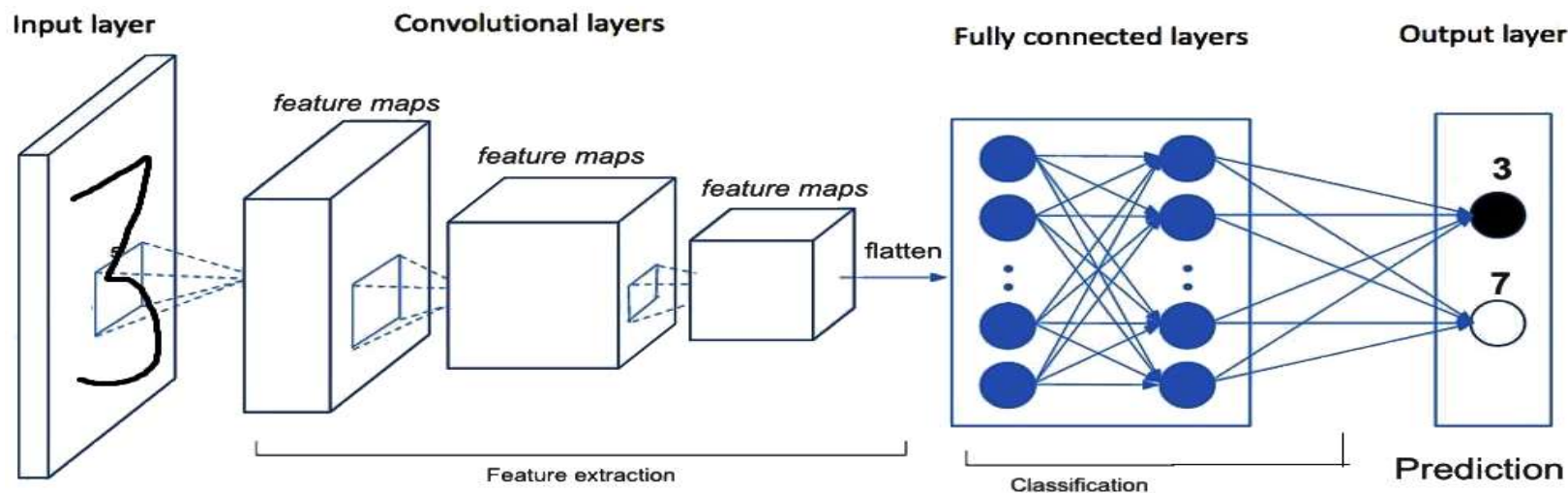


CNN



THE BIG PICTURE

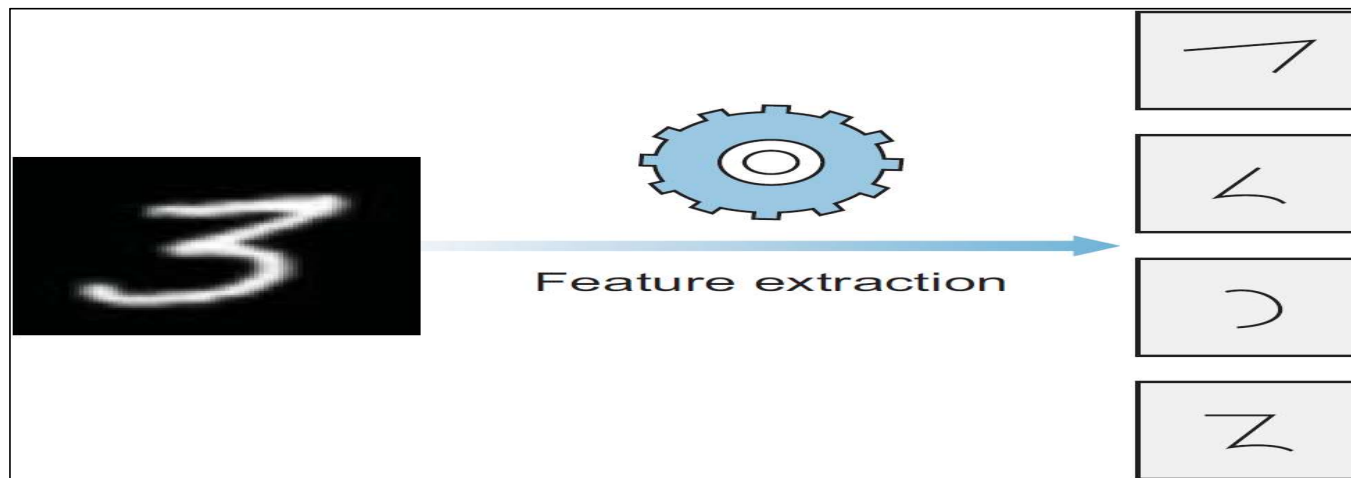
- Just like how regular neural networks contain multiple layer which allow each layer to find successively complex features, CNNs work the same way.
- All the same. The difference is that we will use convolutional layers instead of regular fully connected layers for the feature learning piece.



A CLOSER LOOK ON FEATURE EXTRACTION

- You can think of the feature extraction step as breaking large images into smaller pieces of features and stacking them into a vector.

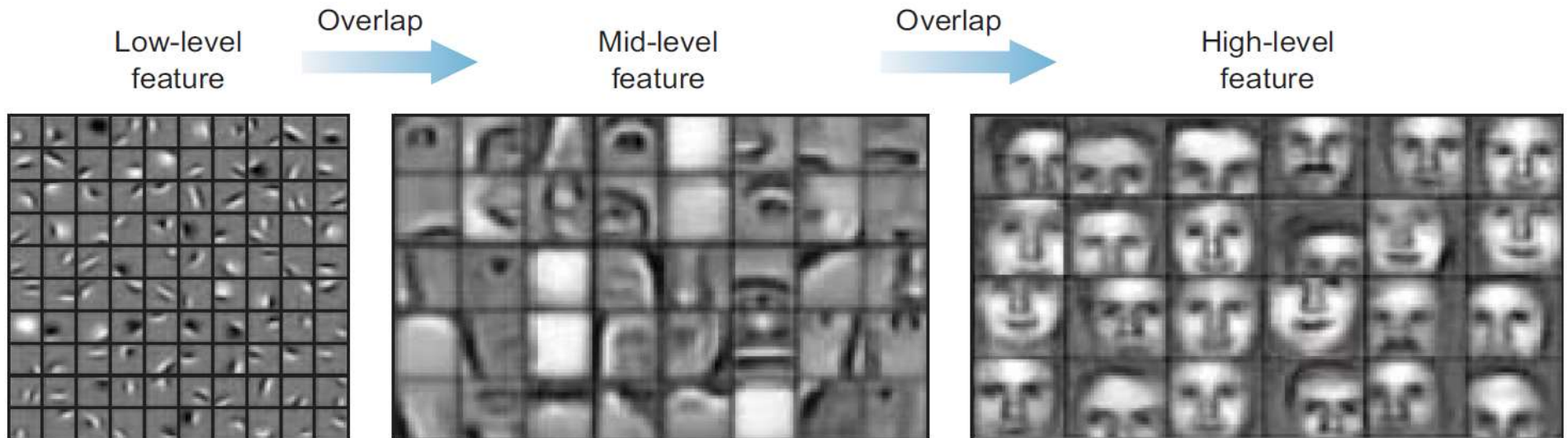
■



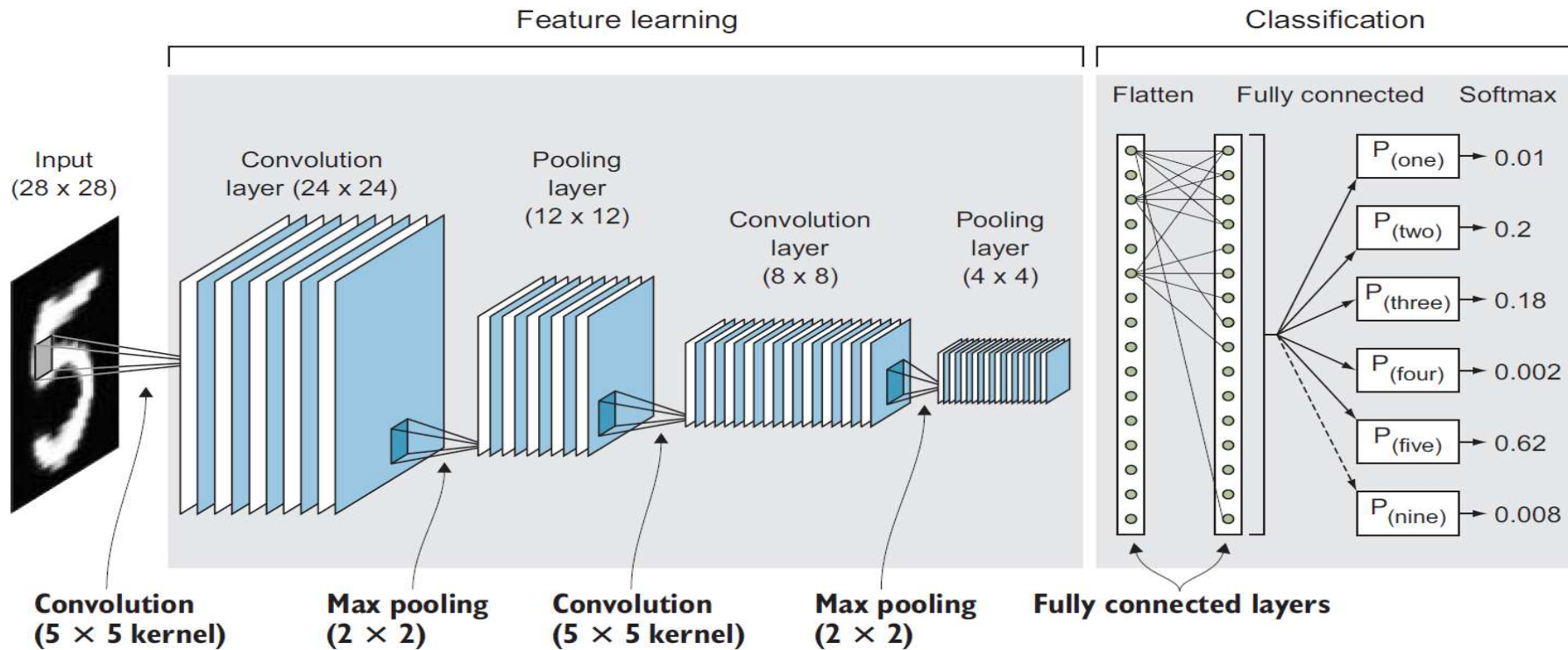
HOW CNN EXTRACT FEATURES

- How CNN extract these features?

- It is important to callout that the CNN doesn't go from the image input to the features vector directly in one layer.
- This usually happens in 10s and 100s of layers .

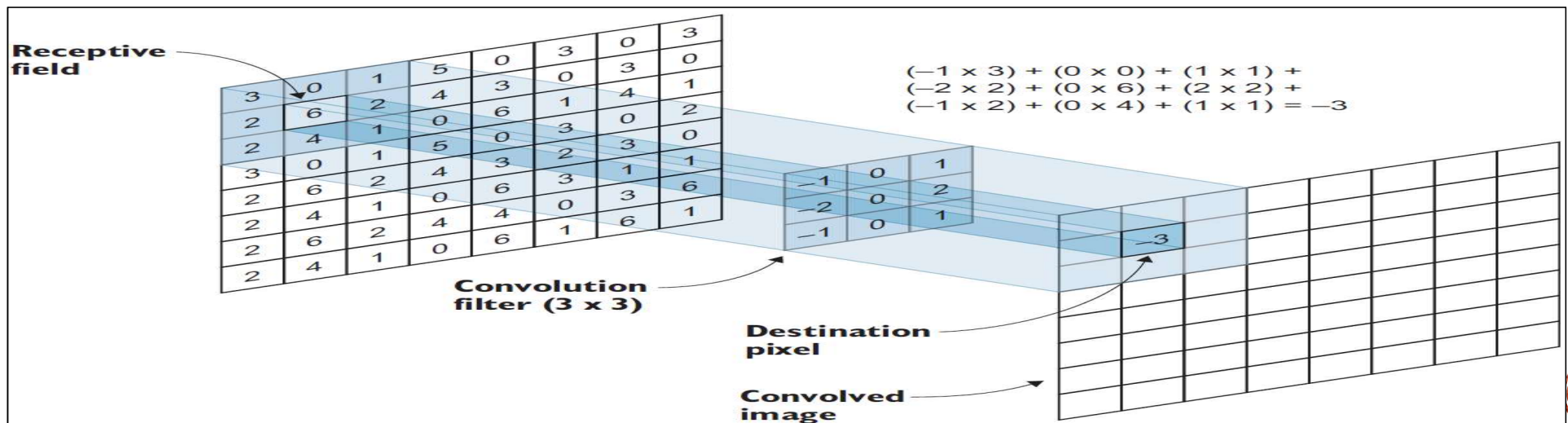


CNN ARCHITECTURE

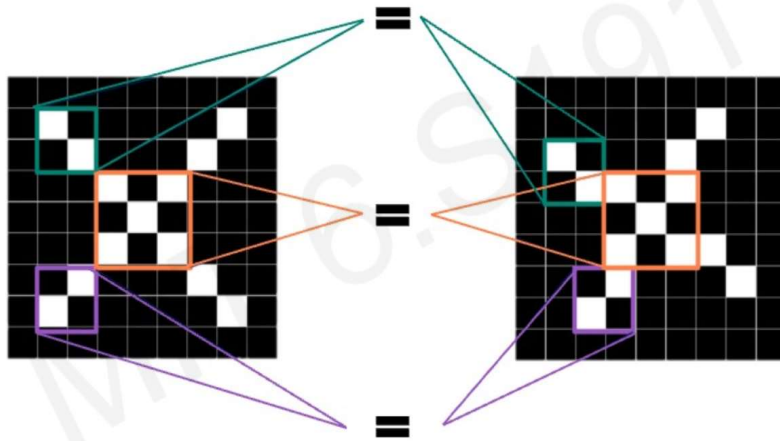


CONVOLUTIONAL LAYERS (CONV)

- Convolutional layer is the core building block of a Convolutional Neural Network. Convolution layers act like a feature finder window that slides over the image pixel-by-pixel to extract meaningful features that identify the objects in the image.
- What is convolution?**
 - In mathematics, convolution is the operation of two functions to produce a third modified function.
 - the first function is the input image and the second function is the CONV filter. We will perform some mathematical operations to produce a modified image with new pixel values



Features of X



X or X?

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

?

==

$$\begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & 1 & -1 & -1 & -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 & -1 & 1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 & -1 & -1 & 1 & -1 \\ -1 & -1 & 1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

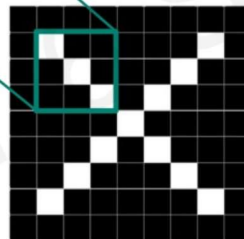
Image is represented as matrix of pixel values... and computers are literal!
We want to be able to classify an X as an X even if it's shifted, shrunk, rotated, deformed.

Filters to Detect X Features

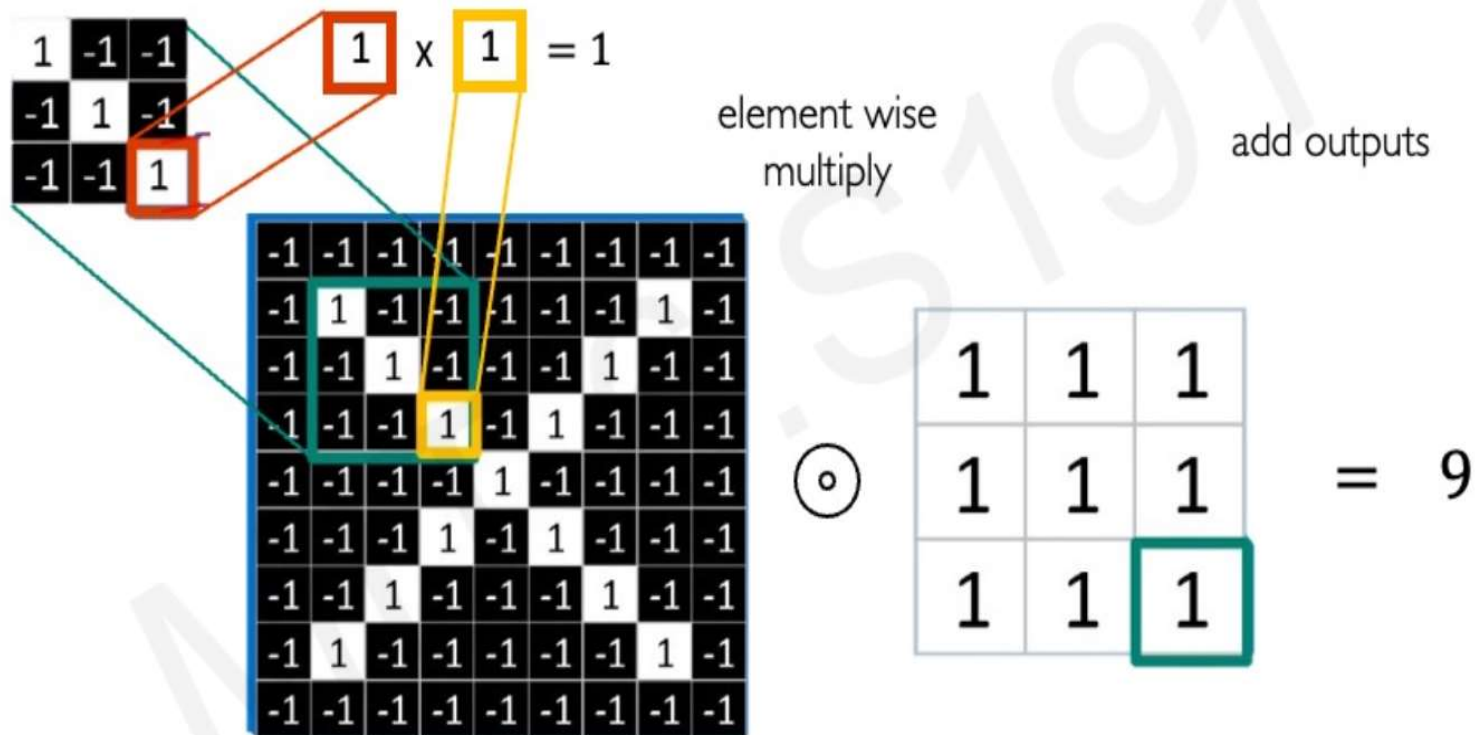
filters

$$\begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}$$

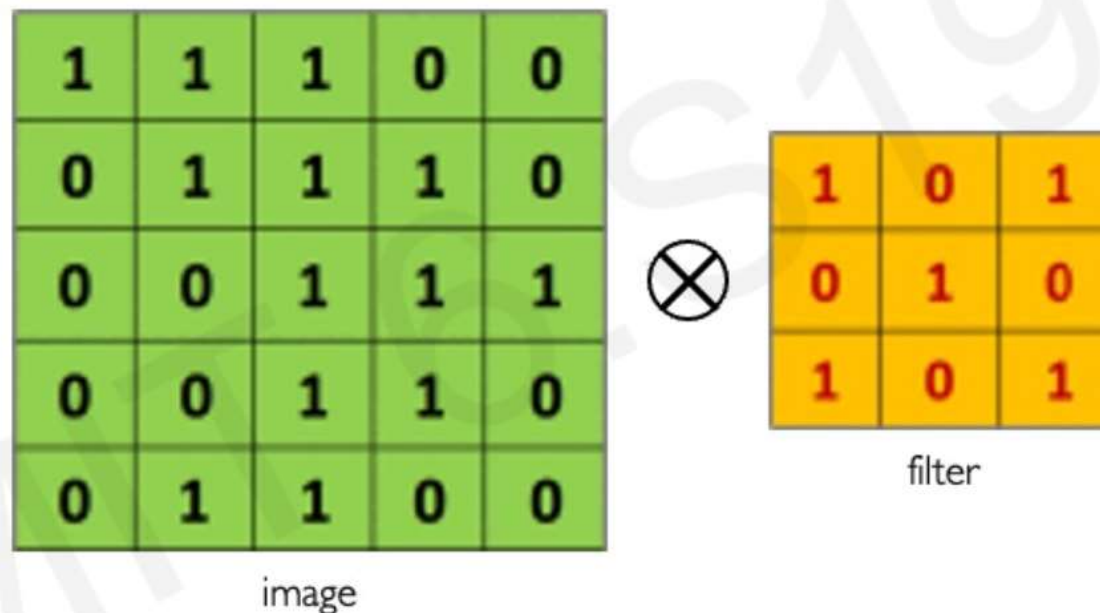
$$\begin{bmatrix} -1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & -1 \end{bmatrix}$$


The Convolution Operation



The Convolution Operation

Suppose we want to compute the convolution of a 5x5 image and a 3x3 filter:

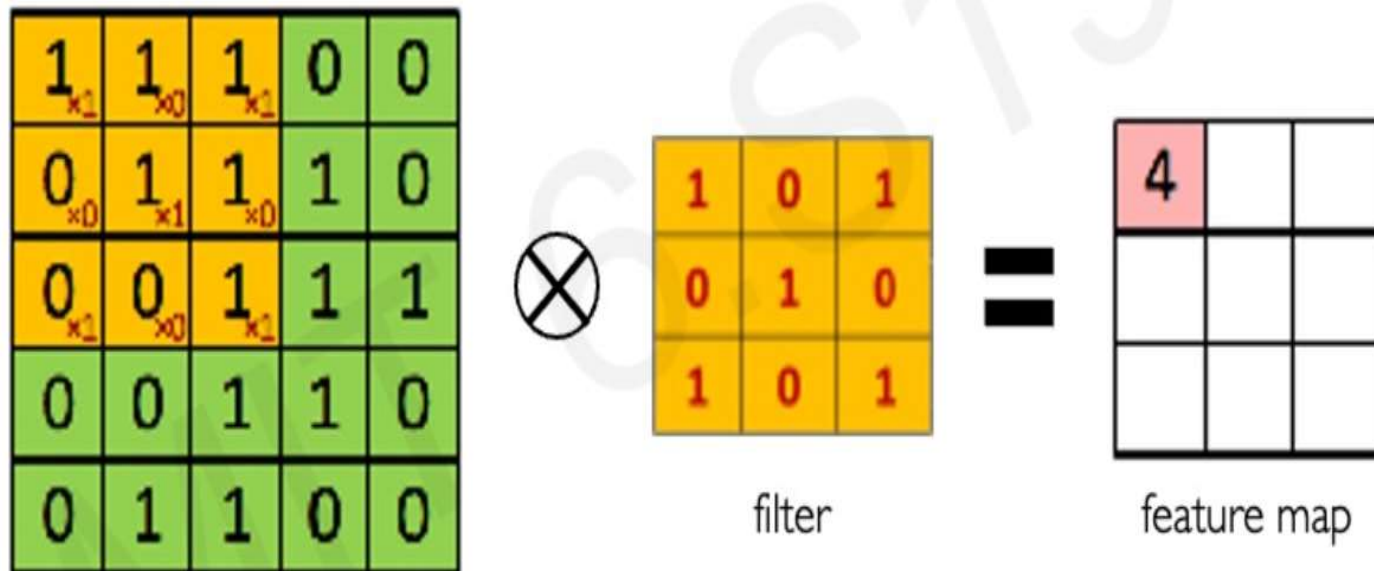


We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs...



The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:

1	1 _{x1}	1 _{x0}	0 _{x1}	0
0	1 _{x0}	1 _{x1}	1 _{x0}	0
0	0 _{x1}	1 _{x0}	1 _{x1}	1
0	0	1	1	0
0	1	1	0	0



1	0	1
0	1	0
1	0	1

filter



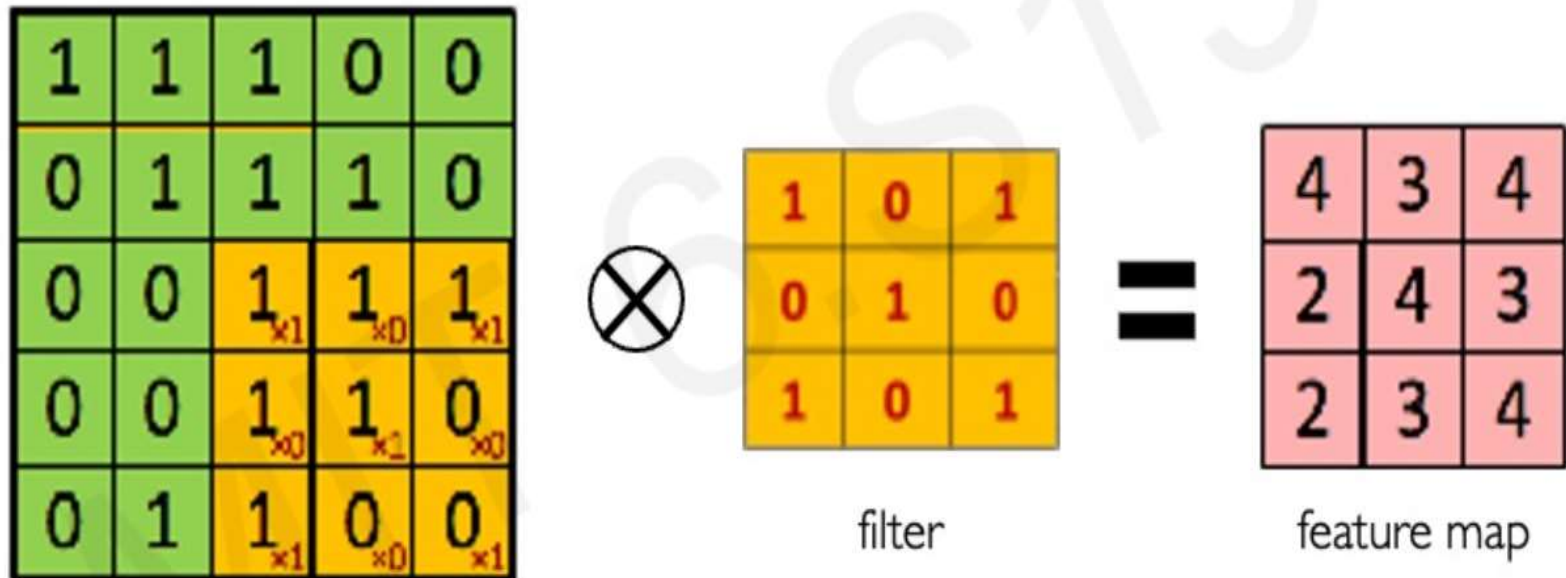
4	3	

feature map

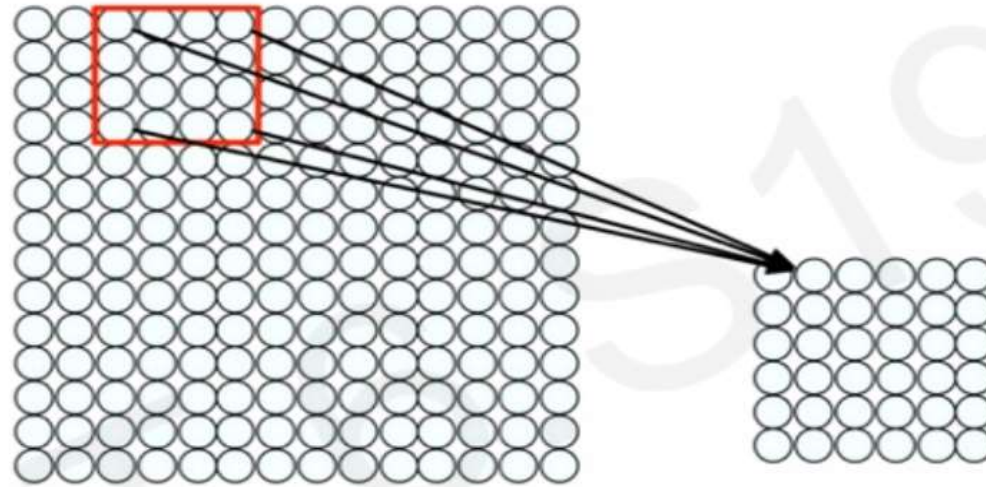


The Convolution Operation

We slide the 3x3 filter over the input image, element-wise multiply, and add the outputs:



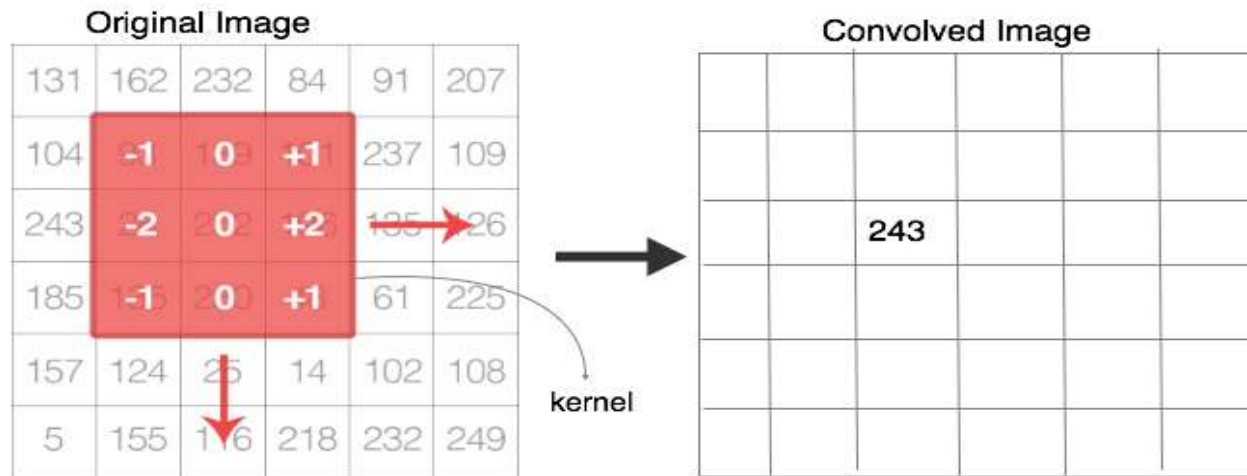
Feature Extraction with Convolution



- 1) Apply a set of weights – a filter – to extract **local features**
- 2) Use **multiple filters** to extract different features
- 3) **Spatially share** parameters of each filter



CONVOLUTION DETAILS.



$$(93 \times -1) + (139 \times 0) + (101 \times 1) + (26 \times -2) + (252 \times 0) + (196 \times 2) + (135 \times -1) + (240 \times 0) + (48 \times 1) = 243$$



CONVOLUTION DETAILS

- The small 3x3 matrix in the middle *is* the convolution filter. They are also called *Kernels*.
- The kernel slides over the original image pixel-by-pixel and does some math calculations to get the values of the new “convolved” image at the next layer. The area of the image that the filter convolves is called the receptive field.
- **What are the Kernel values?**
 - Remember the connection weights in the MLP, In CNNs, this convolution matrix *is* the weights.
 - This means that they are also *randomly initialized* and the values are *learned* by the network (so you will not have to worry about assigning its values).
 - The math goes like this: same as in MLP, remember we used to multiply the input by the weights and sum them all together to get the weighted sum?
 - $weighted\ sum = x_1 \cdot w_1 + x_2 \cdot w_2 + x_3 \cdot w_3 + \dots + x_n \cdot w_n + b$
 - We do the same thing, except that in CNNs the neurons and the weights are structured in a matrix shape. So, we multiply each pixel in the receptive field by the corresponding pixel in the convolution filter and sum them all together to get the value of the center pixel in the new image.



KERNELS [FILTERS]

- These kernels are often called weights because they determine how important a pixel is in forming a new output image.
- Similar to what we discussed before in MLP about weights, the weights represent the importance of the feature on the output.
- **In images, the input features are the pixel values.**
- Other filters can be applied to detected different types of features.
- For example:
 - some filters detect horizontal edges, others detect vertical edges,
 - some other filters detect more complex shapes like corners and so on.
- The point is, these filters when applied in the convolutional layers, yield the feature learning behaviour that we discussed earlier: first they learn simple features like edges and straight lines then later layers learn more complex features.

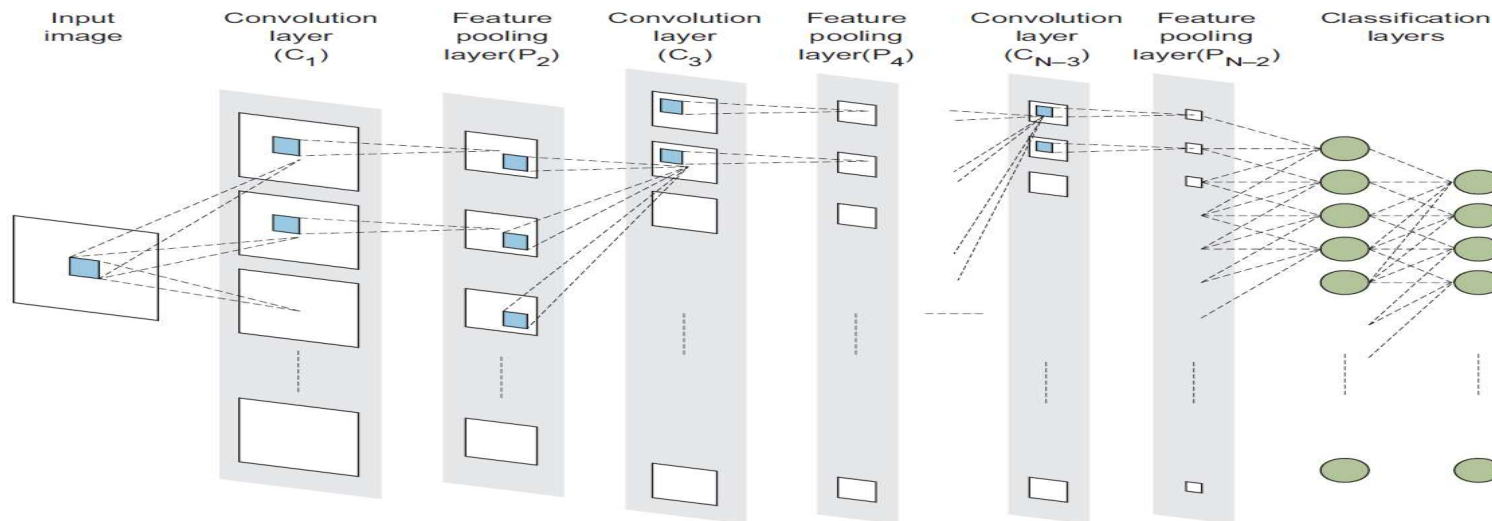
Edge detection
kernel

0	-1	0
-1	4	-1
0	-1	0



NUMBER OF FILTERS IN THE CONV LAYER

- the CONV layers are the hidden layers. And to increase the number of neurons in hidden layers, we increase the number of kernels in CONV layers. Each Kernel unit is considered a neuron.
- So, by increasing the number of kernels in a CONV layer, we are increasing the number of hidden units which make our network more complex to detect more complex patterns. Similarly when we added more neurons (hidden units) to the hidden layers in the MLP.



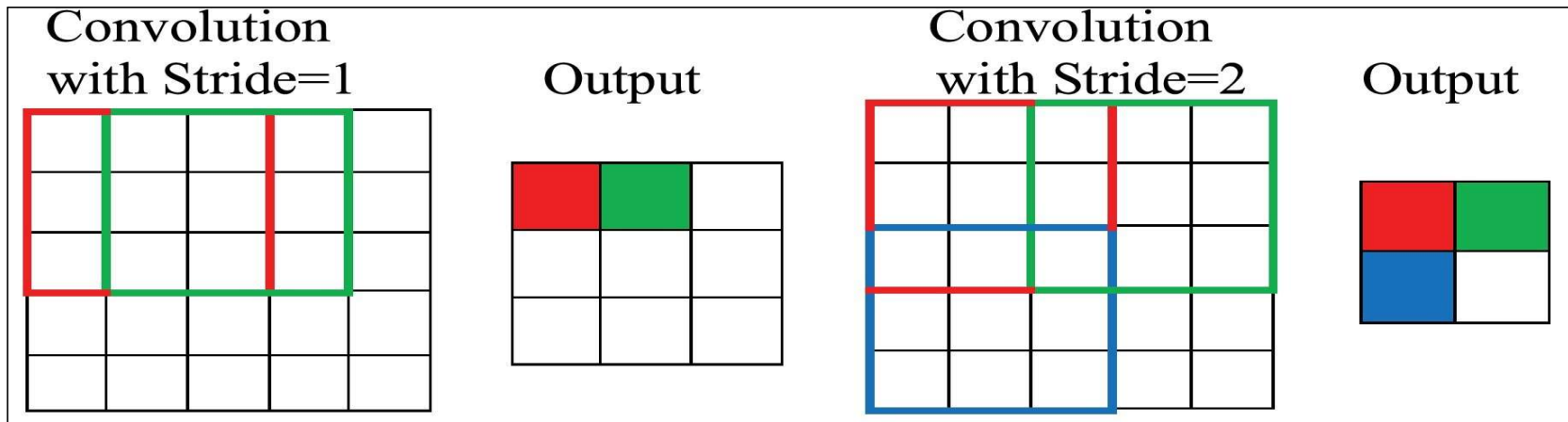
KERNEL SIZE

- ***Kernel_size*** is one of the **hyperparameters** that you will be setting when building a convolutional layer.
- Like most neural networks hyperparameters, there is no one good answer that fits all problems.
- **The intuition is:** smaller filters will capture very fine details of the image and bigger filters will leave out minute details in the image.



STRIDE AND PADDING

- These two hyperparameters you will usually think of them both together because they both control the shape of the output of this CONV layer.
- **Stride:**
 - is the amount by which the filter slides over the image.
 - For example, to slide the CONV filter one pixel at a time, then the stride = 1.
 - If we want to jump two pixels at a time, then stride = 2.
 - Jumping pixels will produce smaller output volumes spatially.
 - [Strides of 3 or more are uncommon and rare in practice.](#)



CONT.

■ **Padding:**

- often called zero-padding because we add zeros around the border of an image.
- Padding is most commonly used to allow us to preserve the spatial size of the input volume so the input and output width and height are the same.
- This padding adds some extra space to cover the image which helps the kernel to improve performance. [This is more helpful when used to detect the borders of an image.](#)
- This way we can use CONV layers without necessarily shrinking the height and width of the volumes.
- This is important for building deeper networks. since otherwise the height/width would shrink as you go to deeper layers.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	123	94	2	4	0	0
0	0	11	3	22	192	0	0
0	0	12	4	23	34	0	0
0	0	194	83	12	94	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Padding = 2



POOLING LAYERS OR SUBSAMPLING (POOL)

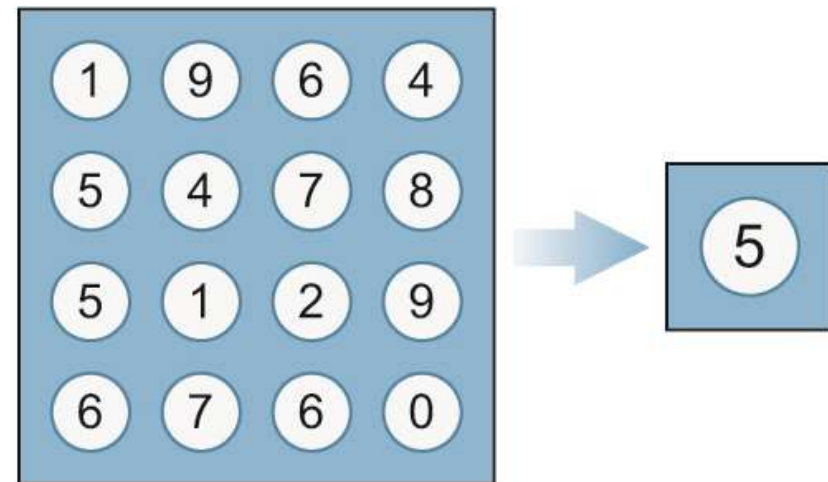
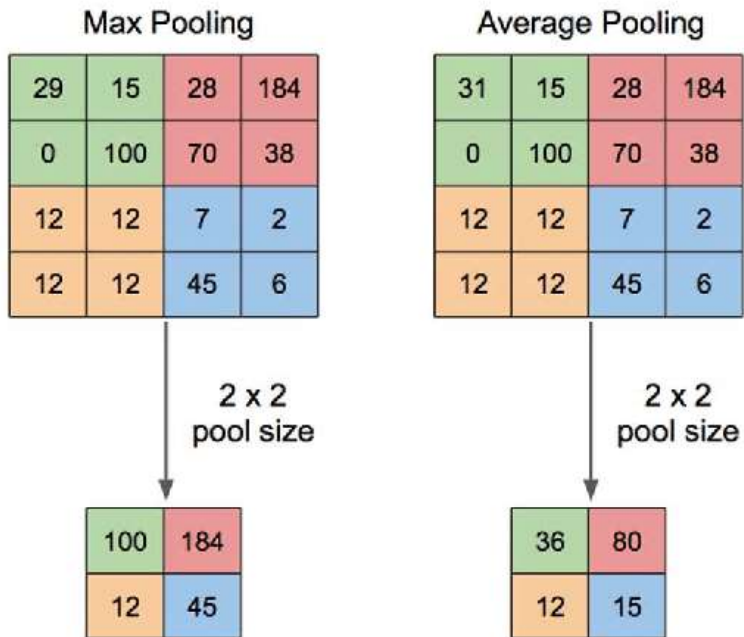
- Why do we need padding?
 - Adding more CONV layers increases the depth of the output layer which leads to the increase of the number of parameters that the network needs to optimize (learn).
 - You can see that after adding several CONV layers (usually tens or even hundreds) this will produce a huge number of parameters (weights).
 - This increase in the network dimensionality will increase the time and space complexity of the mathematical operations that take place in the learning process.
 - **This is when the pooling layers come in.** *Subsampling or pooling helps in reducing the size of the network by reducing the number of parameters passed to the next layer.*
- *What is the pooling layer ?*
 - The pooling operation resizes its input by applying a summary statistic function, such as a maximum or average, to reduce the overall number of parameters to be passed on to the next layer.
- **The goal of the pooling layer is to downsample the feature maps produced by the CONV layer into smaller number of parameters to reduce the computational complexity.**



CONT.

▪ Pooling types:

1. Max pooling
2. Average pooling
3. Global average pooling



Global average pooling



FULLY CONNECTED LAYERS (FC)

- *WHY USE FC LAYERS?*
 - the reason we used CONV layers earlier is that MLP lose a lot of valuable information when trying to extract features from the image because we have to flatten the image before feeding it to the network, whereas CONV layers can process raw images.
 - Now that we have the features extracted, we can use regular MLPs to classify them.
- **Input flattened vector:** Look at the diagram below, to feed the features tube to the MLP for classification, we flatten it to a vector with the dimensions $(1, n)$. For example, if the features tube has the dimensions of $5 \times 5 \times 40$, the flattened vector will be $= (1, 1000)$.
- **Hidden layer (FC):** we add one or more FC layers and each layer has 1 or more neurons (similarly to what we did when we built regular MLPs)
- **Output layer:** it is recommended to use “softmax activation function” for classification problems when we are classifying more than 2 classes.



