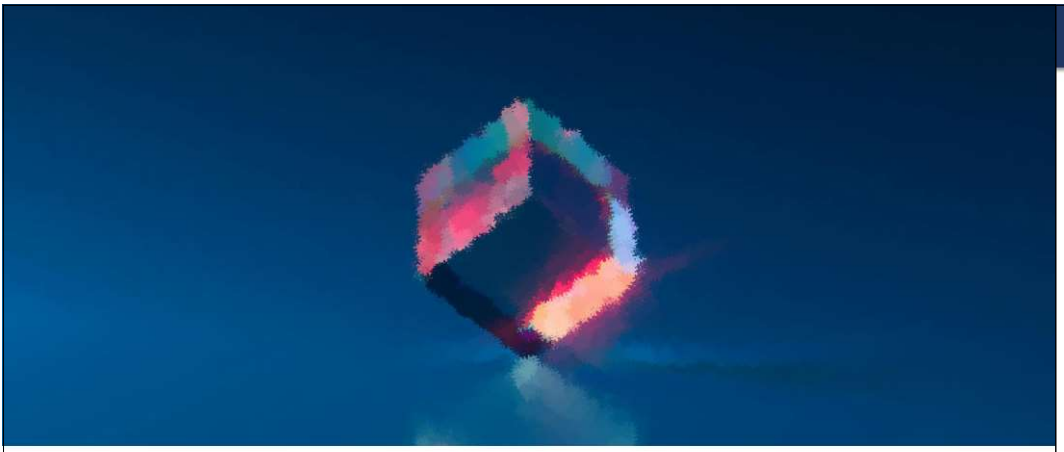


Introduction to Deep Learning

Course Instructor: Dr. M. Umair
Topic: Convolutional Neural Networks

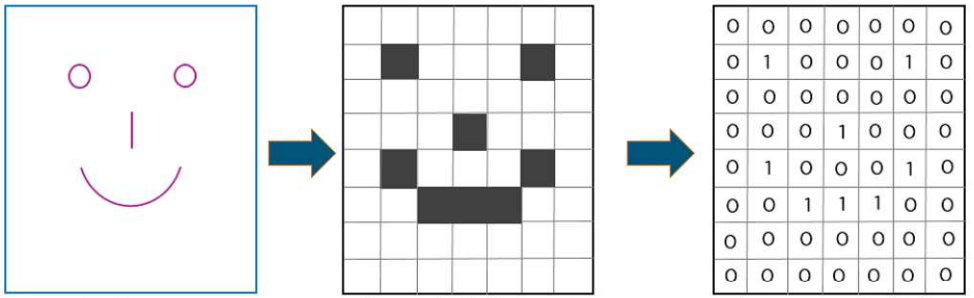
Agenda

1. Preliminaries
2. Convolutional Neural Networks
3. CNN's Architecture
4. Code Example

Preliminaries

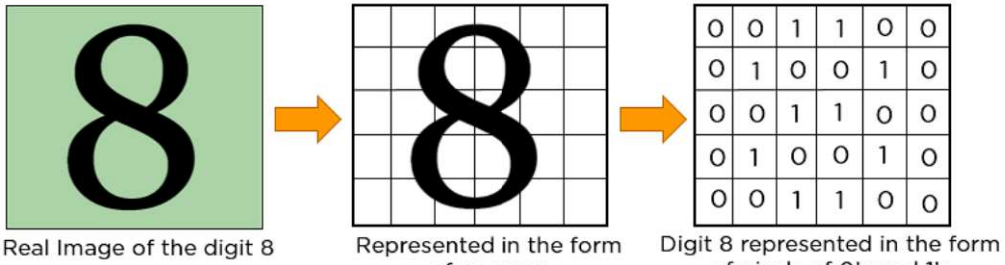
Preliminaries

- **Image Representation**
  - *Image* is represented in the form of an *array of pixel values.*



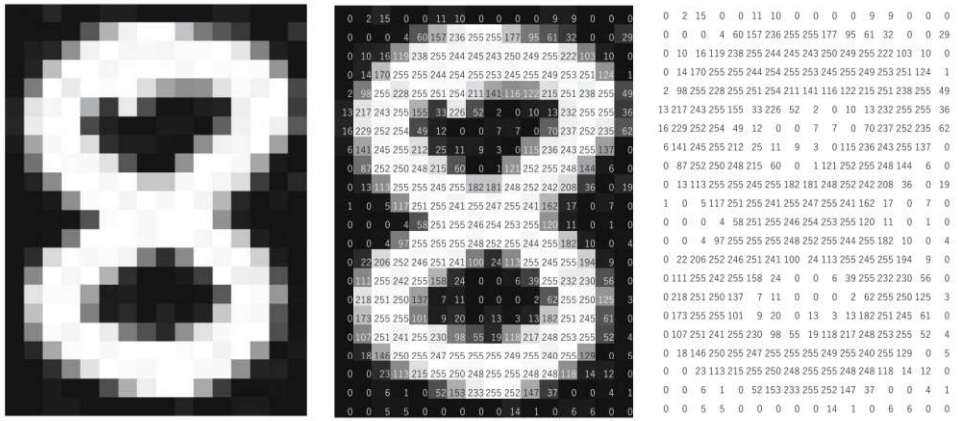
Preliminaries

- **Image Representation**
  - *Image* is represented in the form of an *array of pixel values.*



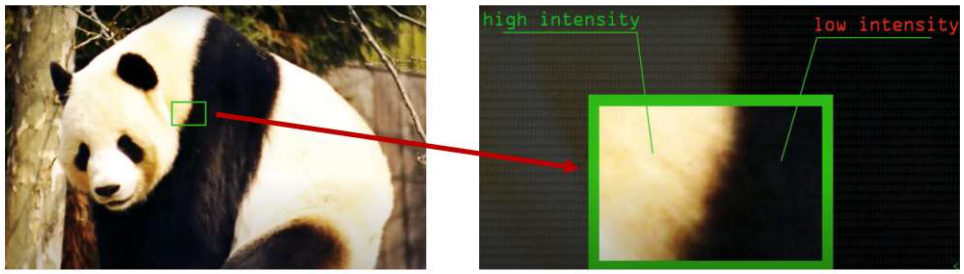
Preliminaries

- **Image Representation** – MNIST example



Preliminaries

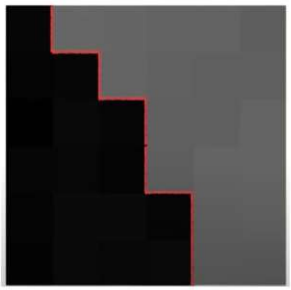
- **Edge Detection**
  - *Sharp change* in intensity or color.
  - *High value* indicates *steep change.*
  - *Low value* indicates *shallow change.*



Preliminaries

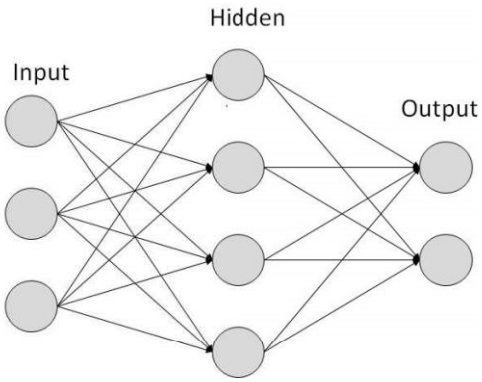
- **Edge Detection**
  - It's a techniques which *identifies significant transitions* in *pixel intensity or color*, indicating the boundaries of objects within an image.

12	90	89	86	87	82
10	12	88	85	83	84
9	15	12	84	84	88
12	14	10	82	88	89
11	17	16	12	88	90
10	16	15	17	89	88



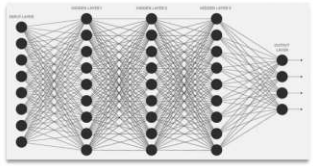
Preliminaries

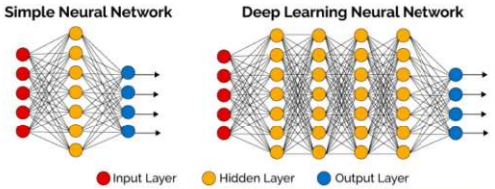
- **Basic Neural Network**

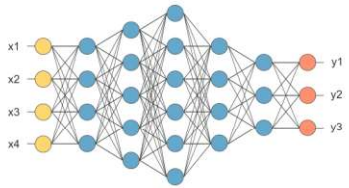


Preliminaries

- **Deep Neural Network**
  - *Deep* neural *network* consists of *more hidden layers.*
  - Each *input* is *connected* to the *hidden layer* and the neural *network* will *decide* the *connections.*



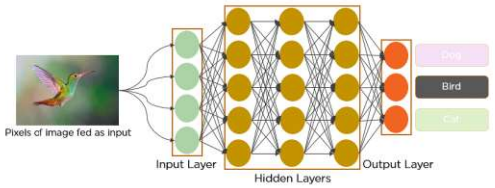




Convolutional Neural Networks

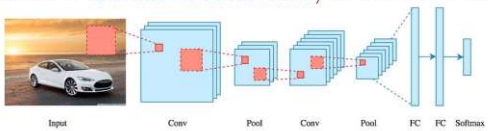
Convolutional Neural Networks

- **Convolutional Neural Network**
  - A Convolutional Neural Network (ConvNet or CNN) is a *type of deep neural network architecture.*
  - It is widely used for
    i. Image classification
    ii. Object detection
    iii. Semantic segmentation
    iv. Natural language processing etc.

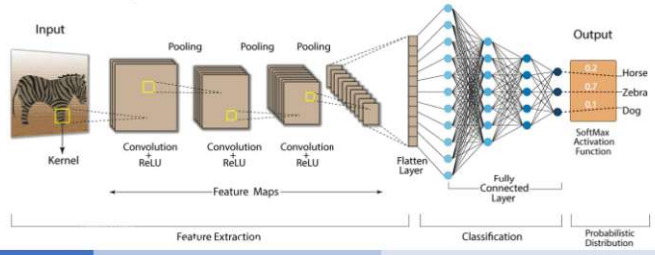


Convolutional Neural Networks

- All *CNN* models follow a *similar architecture*, as shown in the figure below.

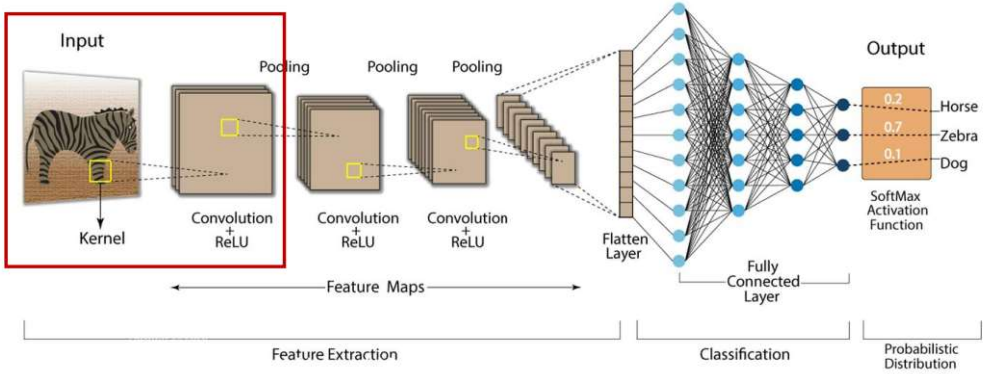


- More specifically, CNNs typically consist of *convolutional*, *pooling*, *flatten*, and *fully connected* layers.



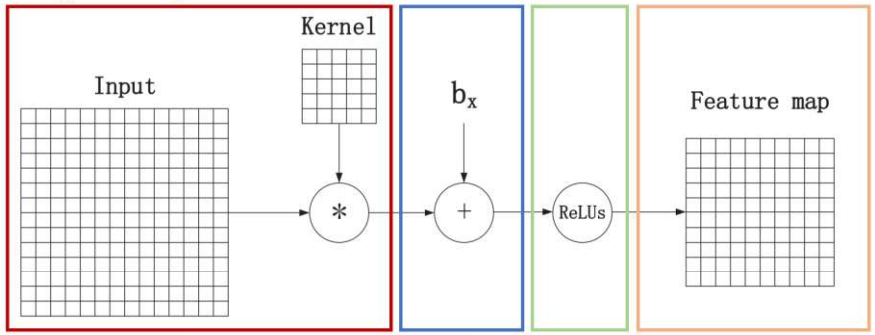
CNN Architecture

Convolution Layer



Convolution Layer

- **Overview**
  - Following is the *overall process* of *convolution layer* (in the simplified case), where b_x is the bias.



Convolution Layer

- **Convolution**
  - The *main idea* behind CNNs is to use *convolutional* layers.
  - Convolution is a specialized type of *linear operation* used for *feature extraction*, where a small array of numbers, called a *kernel*, is applied across the *input*, which is an array of numbers, called a *tensor*. The results is a *feature map*.

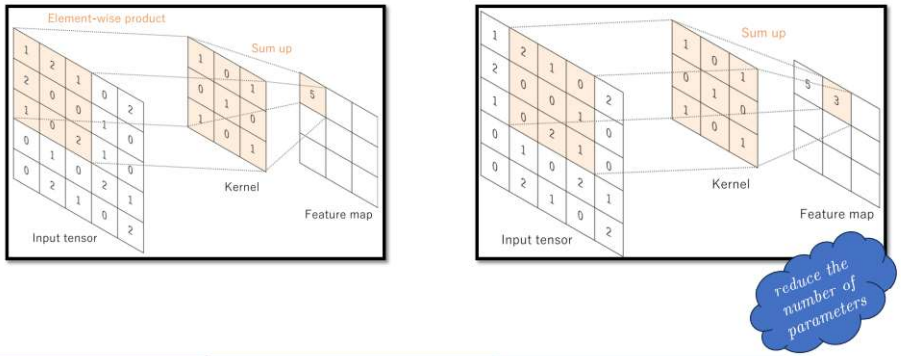
$$y_j = \sum k_{ij} * x_i$$

where $*$ denotes the convolution operation, y_j is the jth feature map of output, k_{ij} is the trainable convolutional kernel (also called filter), and x_i is the ith input.

Convolution Layer

- **Convolution - Example**

$$y_j = \sum k_{ij} * x_i$$

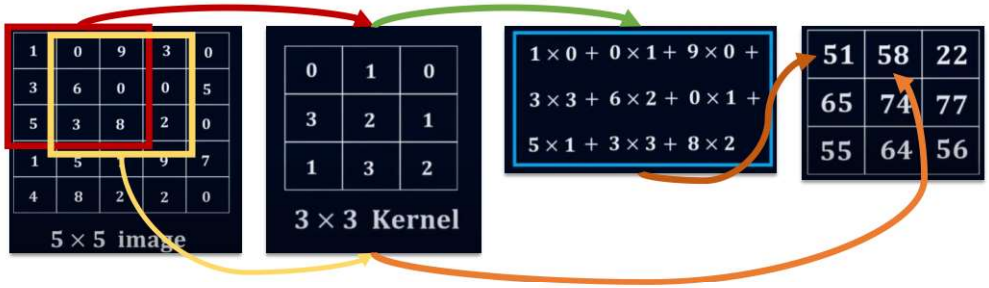


Convolution Layer

- **Kernels (filters) & Feature Maps**
  - The *convolutional layers* apply a *set of kernels (filters)* to the input data, which *results* in a *set of feature maps* that capture the features of the input data.
  - The *number* of the *feature map* is the *same* as the *depth* of the convolution *kernel*.
  - The *filters are learned during the training process*, and they are *used* to *detect* patterns and *features* in the input data.

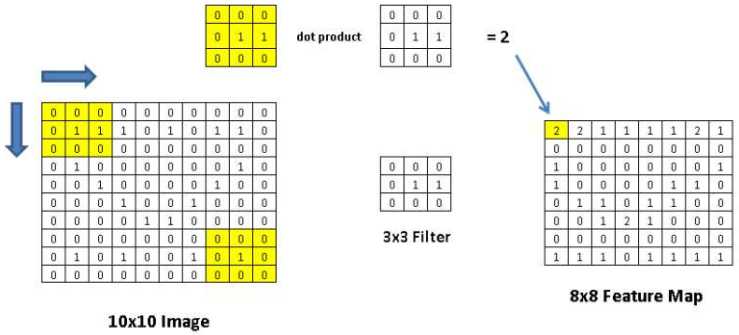
Convolution Layer

- **Kernels or Filters**
  - *Kernels (filters) reduces* the *dimension* and *extracts* the *features*.



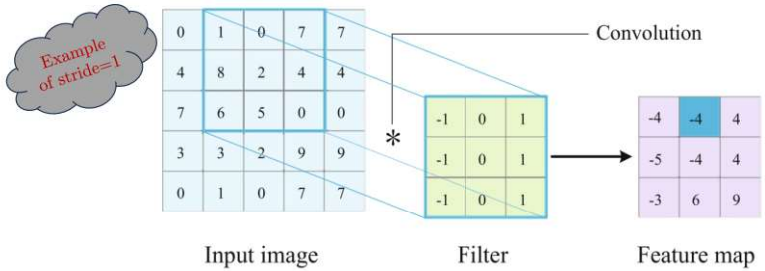
Convolution Layer

- **Kernels or Filters**
  - *Kernels (filters) reduces* the *dimension* and *extracts* the *features*.



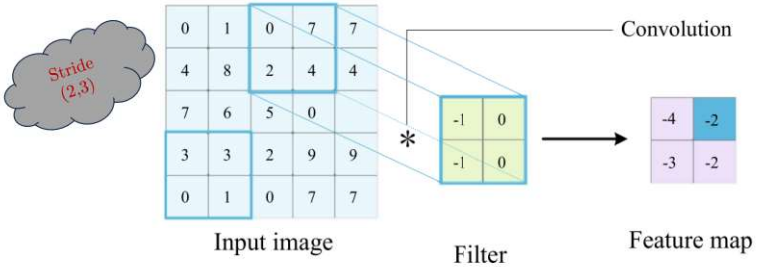
Convolution Layer

- **Strides**
  - **Stride** is the *number of rows and columns* that the convolution *kernel slides over* the *input matrix* in order from left to right and top to bottom, starting from the top left of the input matrix.



Convolution Layer

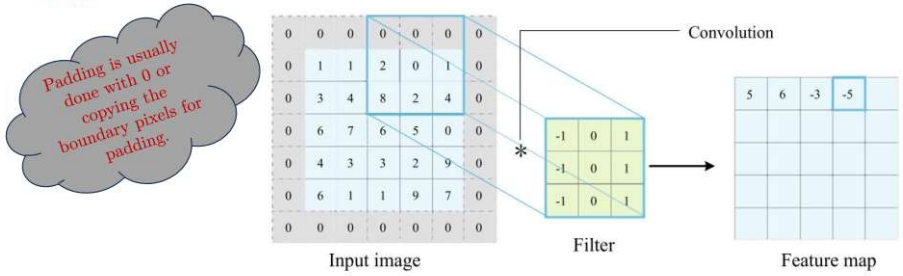
- **Strides**
  - Another *example* is a *stride of 2 in the horizontal direction and 3 in the vertical direction.*



The output data size,
computational complexity,
and feature extraction capability
can all be impacted by the stride.

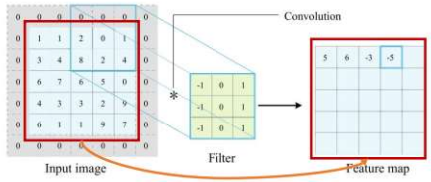
Convolution Layer

- **Padding**
  - **Padding** is the process of *adding* a certain number of *pixels* to the *edges* of the *input data* so that the *size* of the *output data* can *match* the *input data.*



Convolution Layer

- **Padding**
  - *Padding* makes it easier for the convolution kernel to *learn* the *information surrounding the input image.*
  - For instance, when the *5 × 5 × 1 image* is *reinforced* into a *7 × 7 × 1* and applied to the *3 × 3 × 1 kernel* over it, *the computed matrix* is shown to be of dimensions *5 × 5 × 1.*

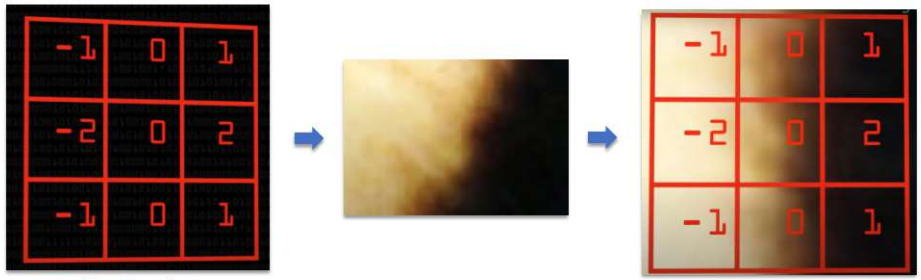


If the same convolution procedure is done without padding, the output might have a smaller-sized image.

Consequently, a 5 × 5 × 1 image will be converted to a 3 × 3 × 1 image.

Convolution Layer

- **Extracting Features**
  - Edge detection



Negative value indicates light to dark.

- **Extracting Features**
  - Extracting vertical edge

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

$*$

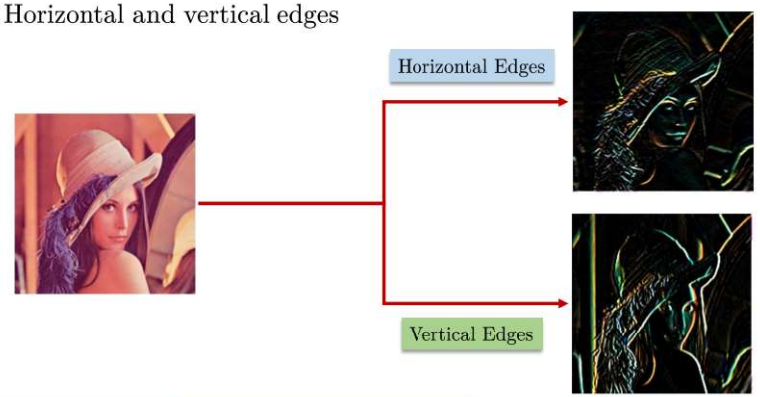
1	0	-1
1	0	-1
1	0	-1

$=$

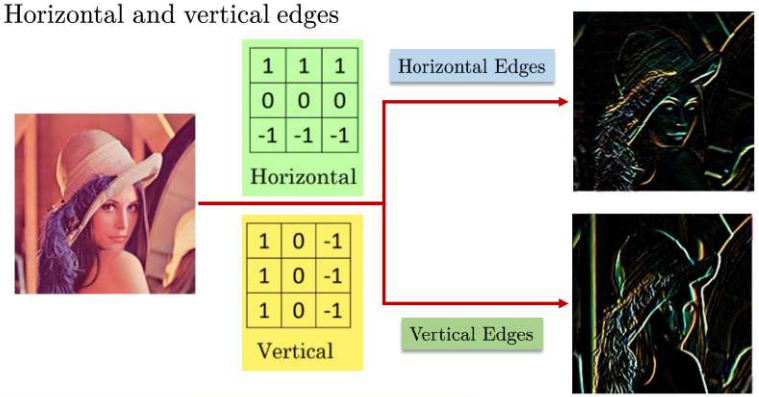
0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Positive value indicates light to dark.

- **Extracting Features**
  - Horizontal and vertical edges



- **Extracting Features**
  - Horizontal and vertical edges



- **Extracting Features**
  - Some commonly used kernels for edge detection.

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

Sobel Filter

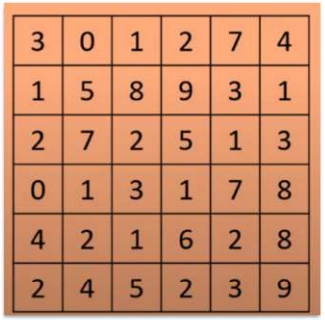
$$G_x = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix} * I$$

$$G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix} * I$$

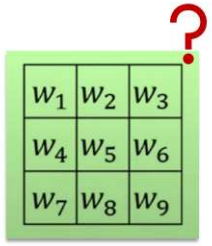
Schaar Filter

Convolution Layer

- **Kernel values, CNN perspective**
  - Which kernel values to choose?



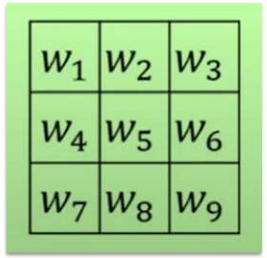
A complex image



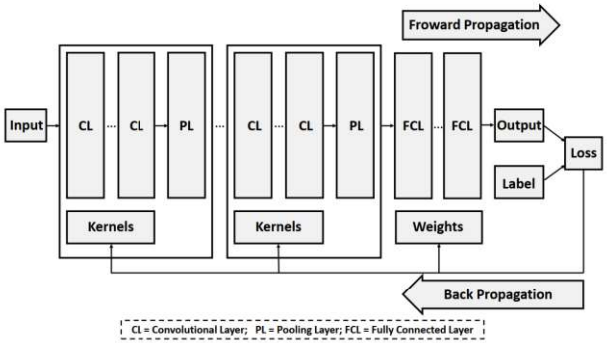
Kernel

Convolution Layer

- **Kernel values, CNN perspective**
  - Which kernel values to choose?

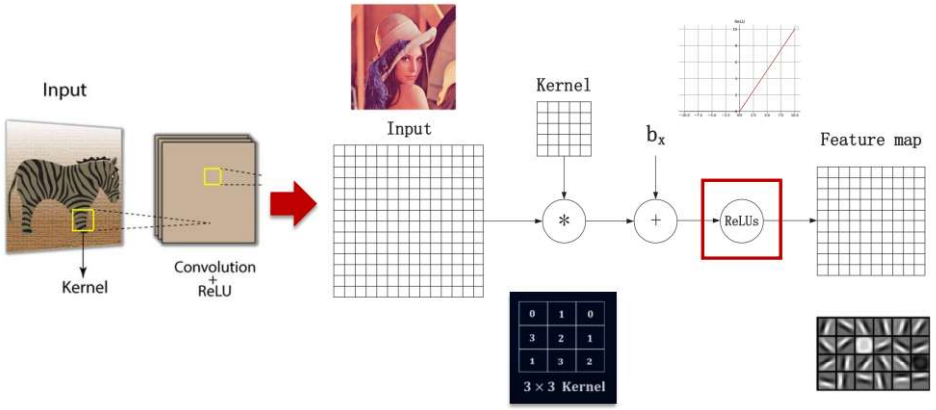


Learning these values makes CNN more robust as compared to computer vision



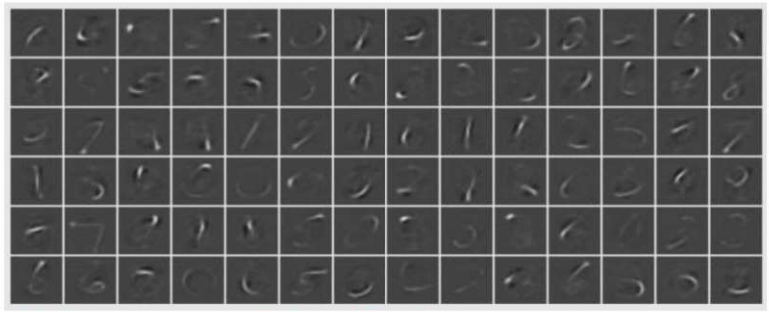
Convolution Layer

- **ReLU Activation**



Convolution Layer

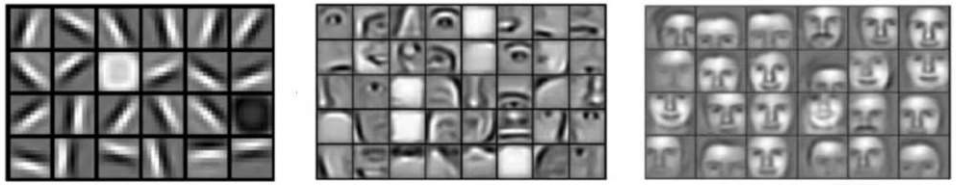
- **First CNN layer output**



Activations taken from the *first convolutional layer* of a simplistic deep CNN, after training on the *MNIST dataset* of handwritten digits. If you look carefully, you can see that the *network has successfully picked up on characteristics* unique to specific numeric digits.

Convolution Layer

- Layer by layer output of a CNN



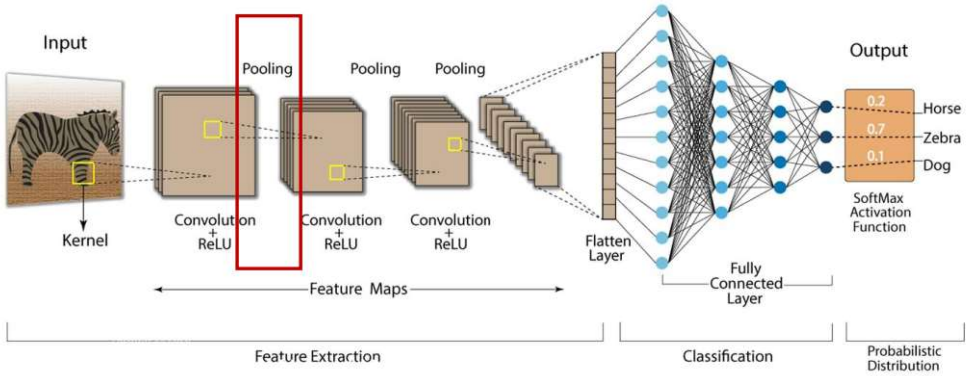
Earlier Layers
Detect edges

Later Layers
Detects part of object

Even Later Layers
Detects complete object

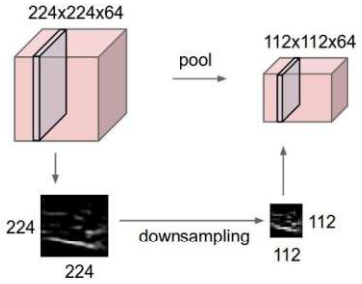
CNN Architecture

Pooling Layer



Pooling Layer

- Introduction
  - The POOLING LAYERS are used to *down-sample* the feature maps, which helps to *reduce the number of parameters* in the model and *prevent overfitting.*

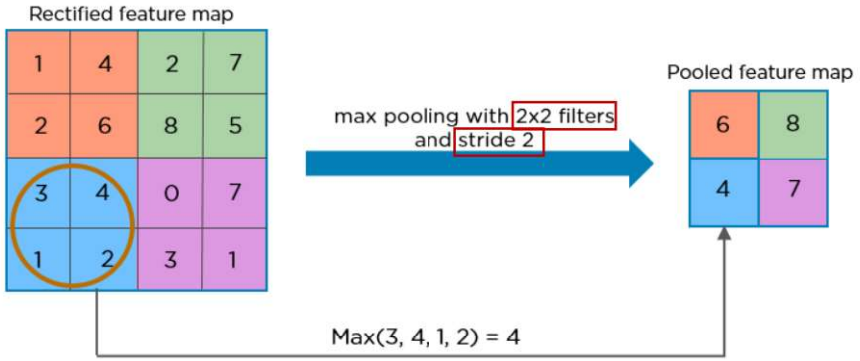


Pooling Layer

- **Introduction**
  - The *pooling procedure*, like the convolution process, can be thought of as a pooling *function without weights*, in which the *input feature* mapping group is divided into many *regions* and each area is pooled to *yield* a *value as a generalization* of this region.
  - There are *several types of pooling layers*, including *max* pooling, *min* pooling, *average* pooling, and *sum* pooling etc.

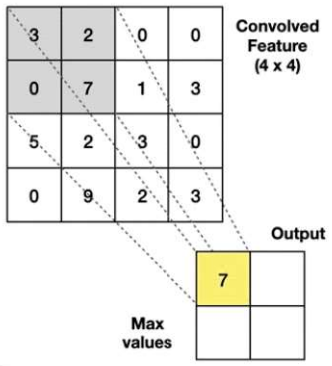
Pooling Layer

- **How it Works?**



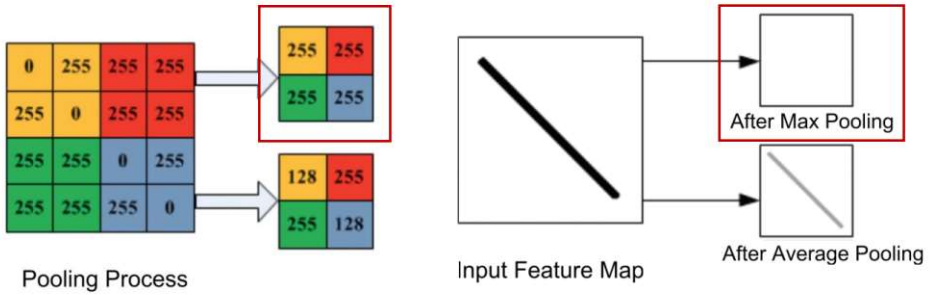
Pooling Layer

- **Max Pooling - Example**



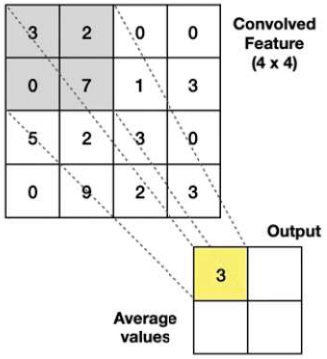
Pooling Layer

- **Max Pooling Result**



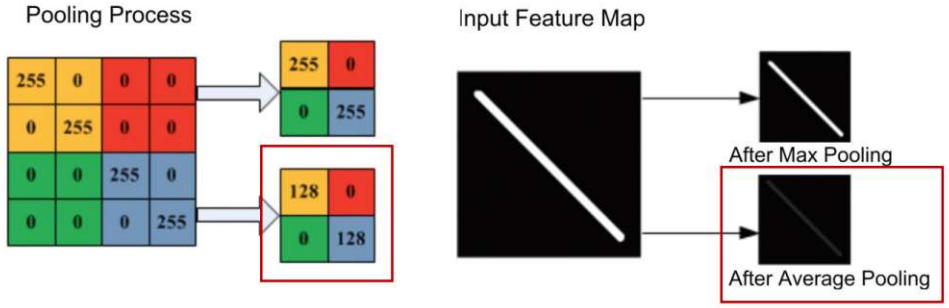
Pooling Layer

- Average Pooling - Example



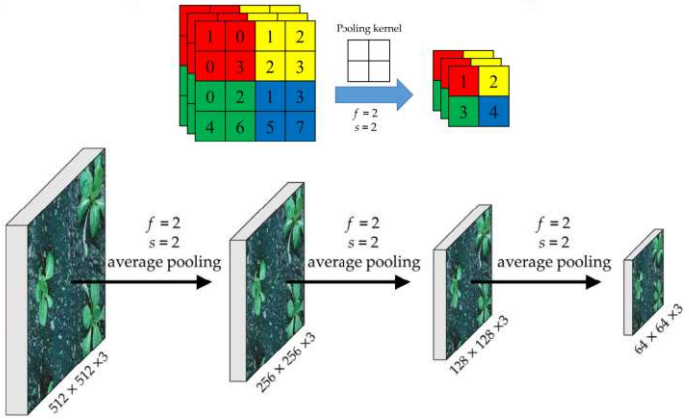
Pooling Layer

- Average Pooling Results



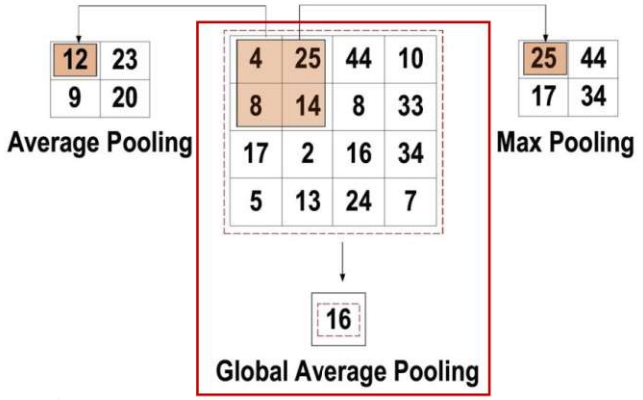
Pooling Layer

- Average Pooling – Order 3 Tensor Example



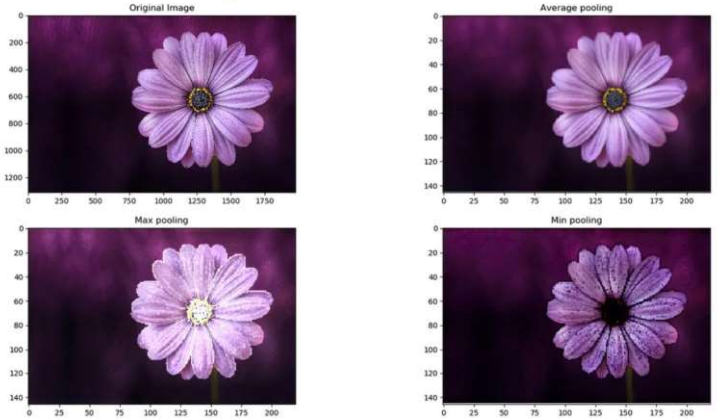
Pooling Layer

- Global Average Pooling - Example



Pooling Layer

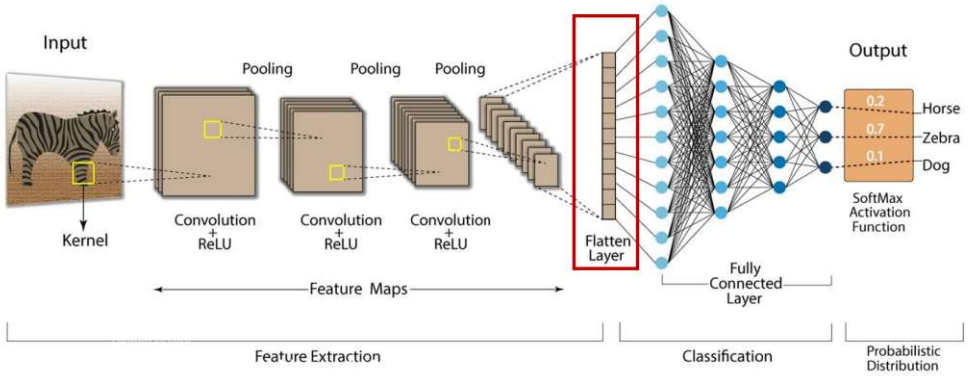
- Some Further Examples



Pooling layer not only effectively compresses the amount of data and parameters, reduces the feature map dimension, minimizes overfitting and reduces the sensitivity of the network to minor variations in input structure

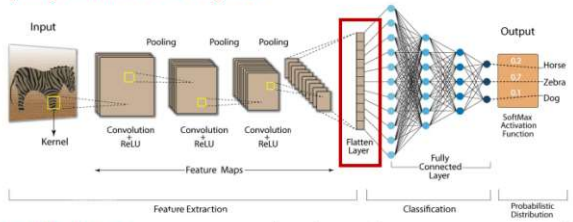
CNN Architecture

Flattening Layer



Flattening Layer

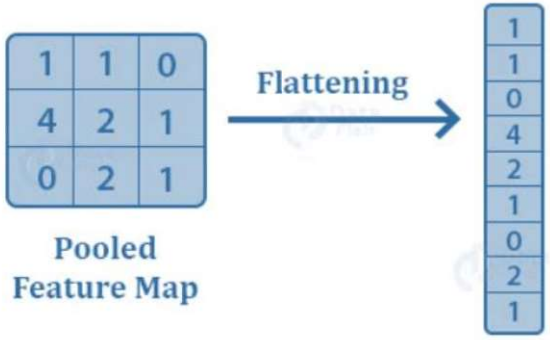
- **The Purpose**
  - The **FLATTEN LAYER** serves the purpose of *reshaping* the *output* of the preceding layer *into a one-dimensional vector*, which can then be *fed into* subsequent *fully connected layers*.



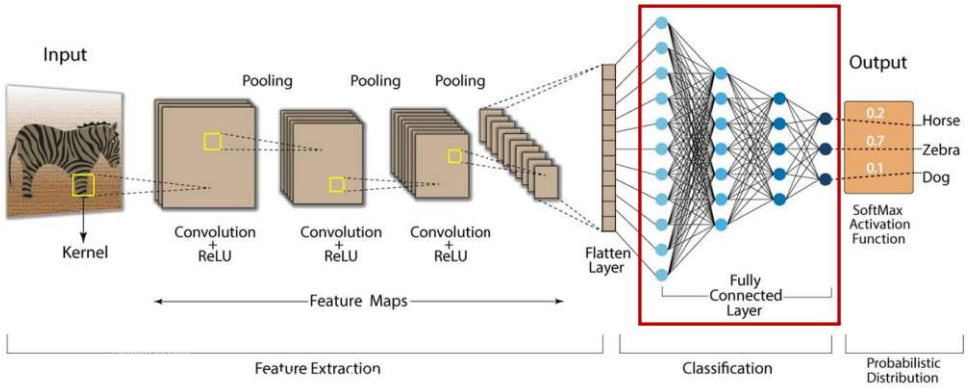
  - This is *typically done* to prepare the data for input into a fully connected layer or another type of *layer that requires a one-dimensional input.*

Flattening Layer

- **How it Works?**



CNN Architecture
Fully Connected Layer

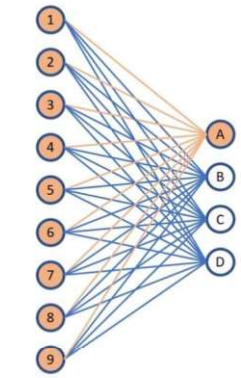


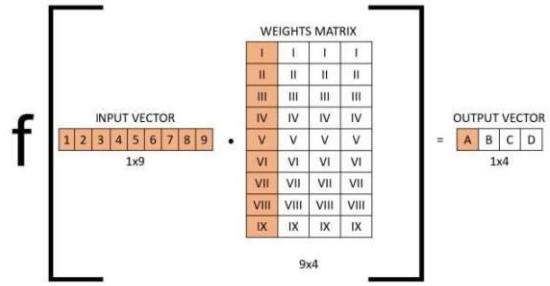
Fully Connected Layer

- **Introduction**
  - The **FULLY CONNECTED** layers are *used to classify* the input data *based* on the *features extracted* by the convolutional and pooling layers.
  - They are *typically the last layers* in the CNN architecture, and they are used to *produce the final output of the model.*
  - The distilled *knowledge* of the last layer is *passed* through *fully-connected layers* with different hidden layers, followed by a *SoftMax activation* function.

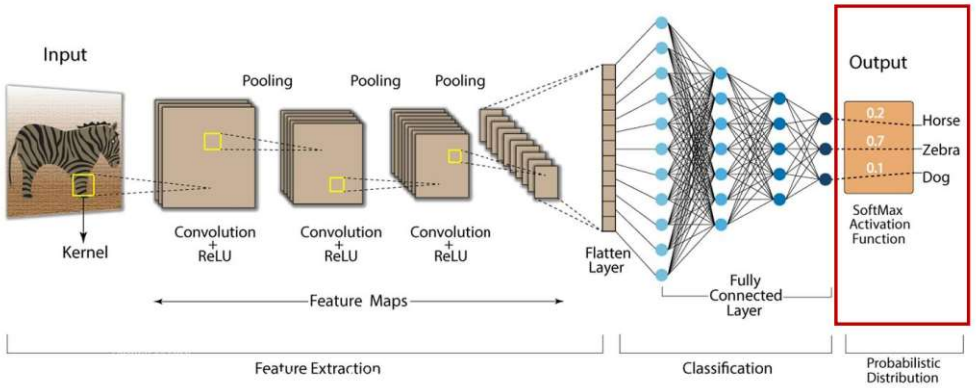
Fully Connected Layer

- **How it Works?**





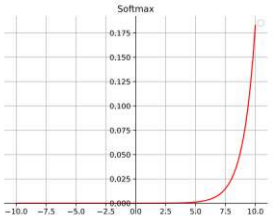
CNN Architecture
Softmax Activation



Softmax Activation

- **Introduction**
  - **Softmax** is an *activation function* for *multi-classification problems*.

$$Soft\max(x) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$



  - For any *real vector* of *length K*, *Softmax activation* can compress it into a *real vector* of *length K*, with values in the *range (0, 1)* and *vector elements summing to 1.*

Code Example

Code Example

- **Convolutional Neural Network (CNN)**
  - https://www.tensorflow.org/tutorials/images/cnn

References

REFERENCES

1. http://yann.lecun.com/exdb/mnist
2. Introduction to Convolutional Neural Networks, Jianxin Wu
3. https://cs231n.github.io/convolutional-networks/
4. Convolutional neural networks: an overview and application in radiology, Rikiya Yamashita et al., 2018
5. https://towardsdatascience.com/a-practical-guide-on-convolutional-neural-networks-cnns-with-keras-21421172005e
6. Injurious or Noninjurious Defect Identification From MFL Images in Pipeline Inspection Using Convolutional Neural Network, Jian Feng et al., 2017
7. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions, Laith Alzubaidi et al., 2021
8. An Introduction to Convolutional Neural Networks, Keiron O'Shea et al., 2015
9. A review of convolutional neural networks in computer vision, Xia Zhao et al., 2024
10. https://builtin.com/machine-learning/fully-connected-layer
11. https://pub.towardsai.net/introduction-to-pooling-layers-in-cnn-dafe61eabe34
12. RunPool: A Dynamic Pooling Layer for Convolution Neural Network, Putra Wanda et al., 2020
13. An Improved U-Net Model Based on Multi-Scale Input and Attention Mechanism: Application for Recognition of Chinese Cabbage and Weed, Zhongyang Ma et al., 2023