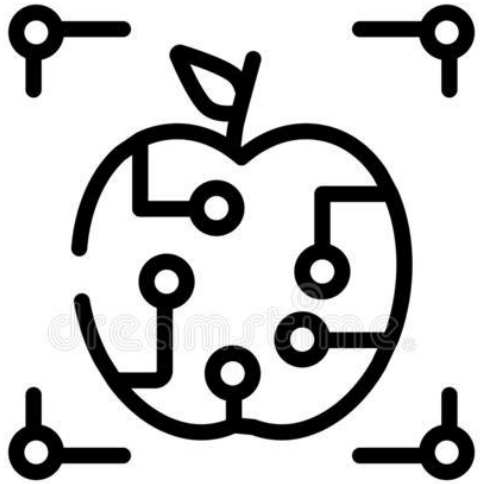
RECOGNITION OF OBJECTS with Convolutional Neural Network

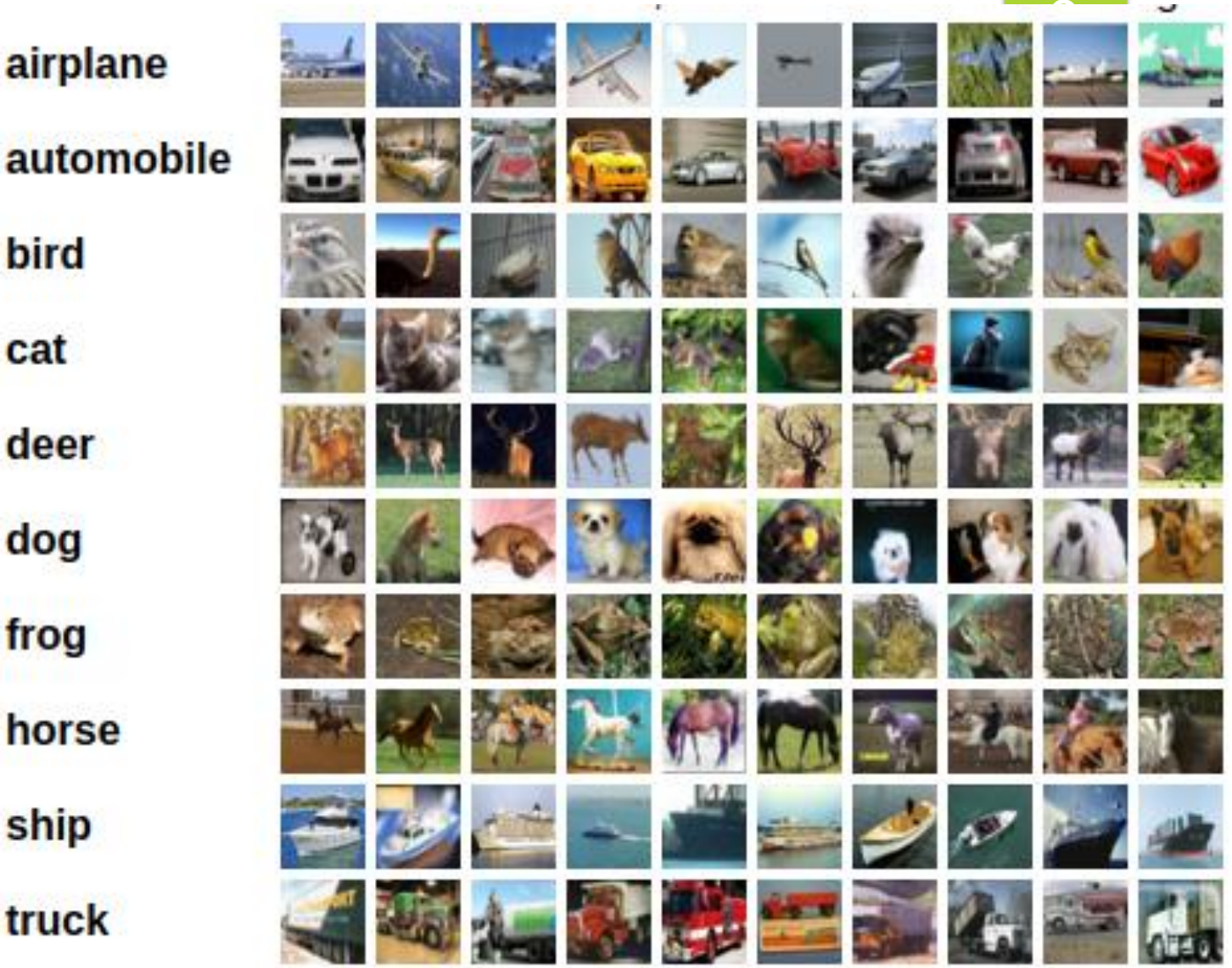
OBJECTIVE

The aim of this project is to build a deep learning model so as to recognize the object by using CIFAR-10 data set (Canadian Institute For Advanced Research)



CIFAR DATASET

- It is a collection of images that are commonly used to train machine learning and computer vision algorithms
- The **CIFAR-10** dataset consists of 60000 32x32 colour **images** in **10** classes, with 6000 **images** per class. There are 50000 training **images** and 10000 test **images**
- Each image is an RGB image of **size** 32x32.



CONCEPTS INVOLVED

- TENSORFLOW
- KERAS
- MATPLOTLIB
- SUPERVISED LEARNING
- COMPUTER VISION
- DEEP LEARNING
- CONVOLUTIONAL NEURAL NETWORK

TENSORFLOW

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.
we use that often to load data, complex mathematical computations

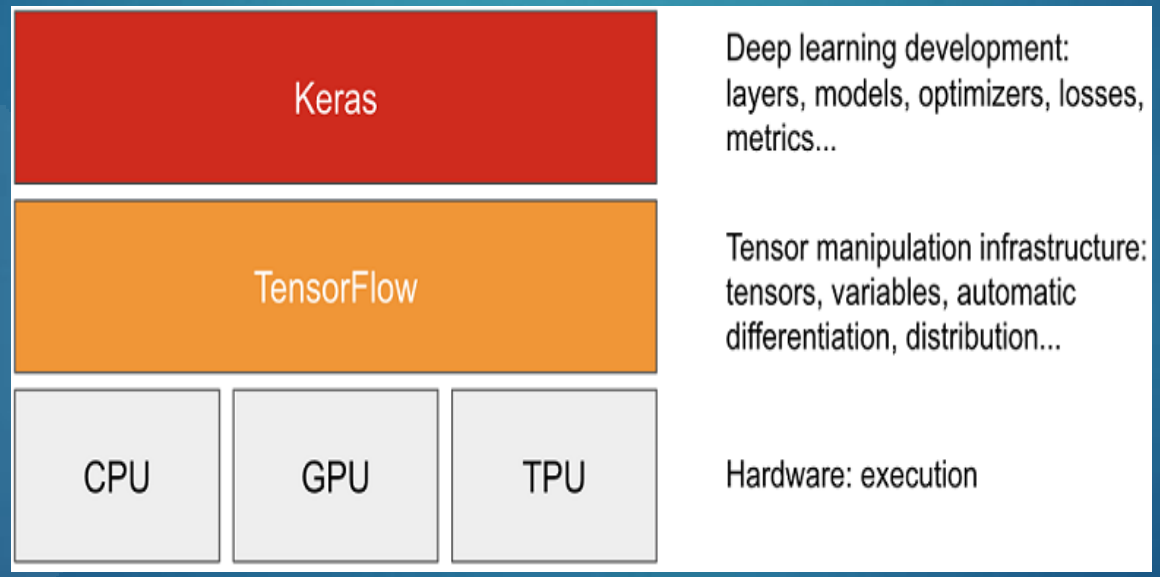
KERAS

- Keras is a high-level library that's built on top of Theano or TensorFlow. It provides a scikit-learn type API (written in Python) for building Neural Networks.
- Developers can use Keras to quickly build neural networks without worrying about the mathematical aspects of tensor algebra, numerical techniques, and optimization methods.
- **The key idea behind the development of Keras is to facilitate experimentations by fast prototyping.**

KERAS : high level

TENSORFLOW : both high level and low level

- Keras focuses on being modular, user-friendly, and extensible. It doesn't handle low-level computations; instead, it hands them off to another library called the Backend.
- Keras was adopted and integrated into TensorFlow in mid-2017. Users can access it via the tf.keras module. However, the Keras library can still operate separately and independently.



KERAS MODELS

SEQUENTIAL MODEL :

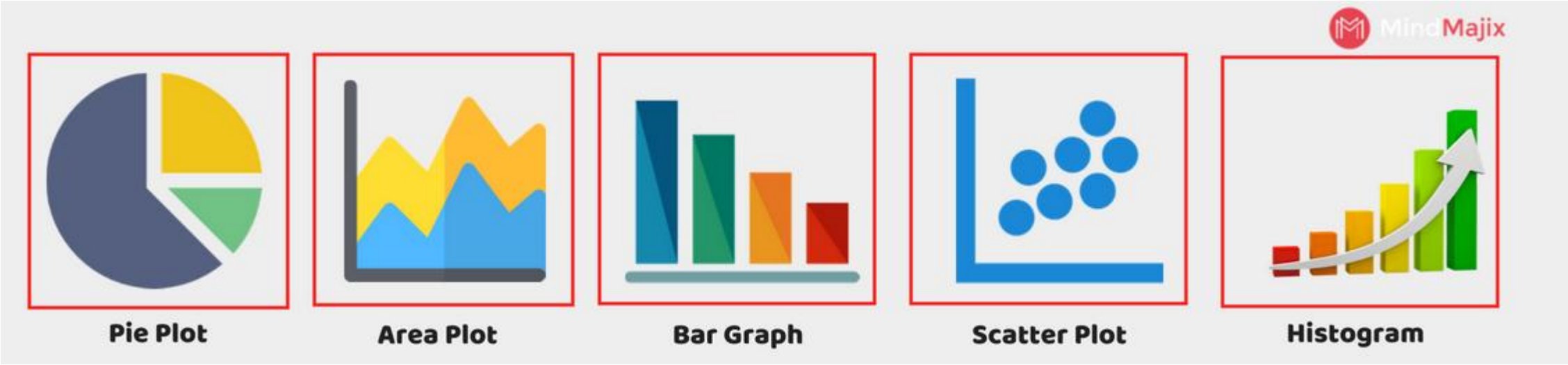
A **Sequential model** is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.
Sequential is the easiest way to build a **model** in Keras. It allows you to build a **model** layer by layer. Each layer has weights that correspond to the layer the follows it. We use the 'add()' function to add layers to our **model**.

FUNCTIONAL MODEL :

It is a more flexible API and is more advanced in nature. It is an alternative model against the Sequential Model that enables you to create models in a more complex manner. This model helps you to define multiple numbers of input and output.

Matplotlib

Matplotlib is a Python 2D plotting library that produces high-quality charts and figures, which helps us visualize extensive data to understand better. Pandas is a handy and useful data-structure tool for analyzing large and complex data.And **pyplot** function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.



COMPUTER VISION

- Computer vision is the process of understanding digital images and videos using computers.
- It seeks to automate tasks that human vision can achieve.
- This involves methods of acquiring, processing, analyzing, and understanding digital images, and extraction of data from the real world to produce information.
- It also has sub-domains such as object recognition, video tracking, and motion estimation, thus having applications in medicine, navigation, and object modeling.

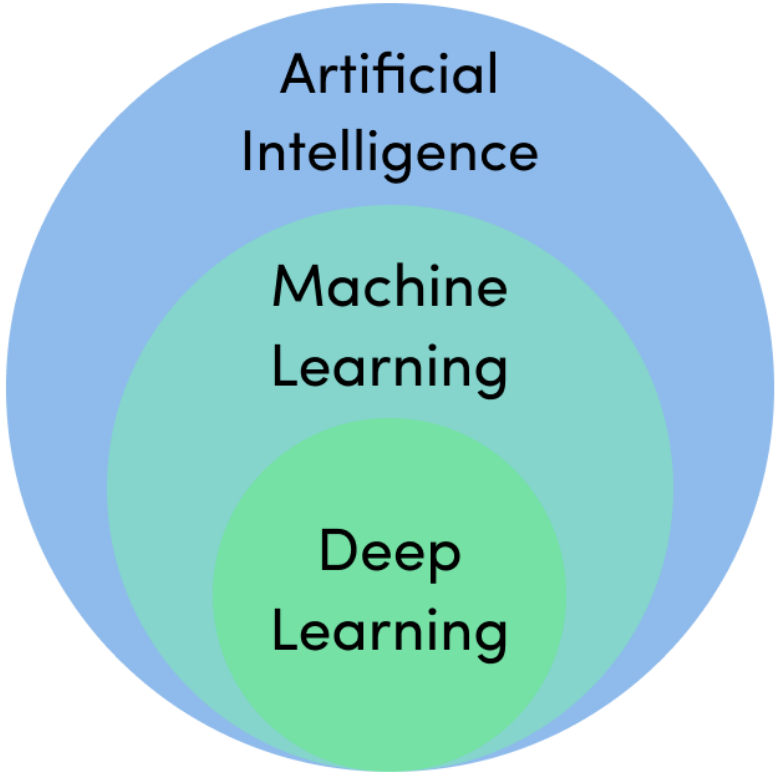
“The goal of computer vision is to understand the content of digital images and videos.”

DEEP LEARNING

WHAT IS NEURAL NETWORK ?

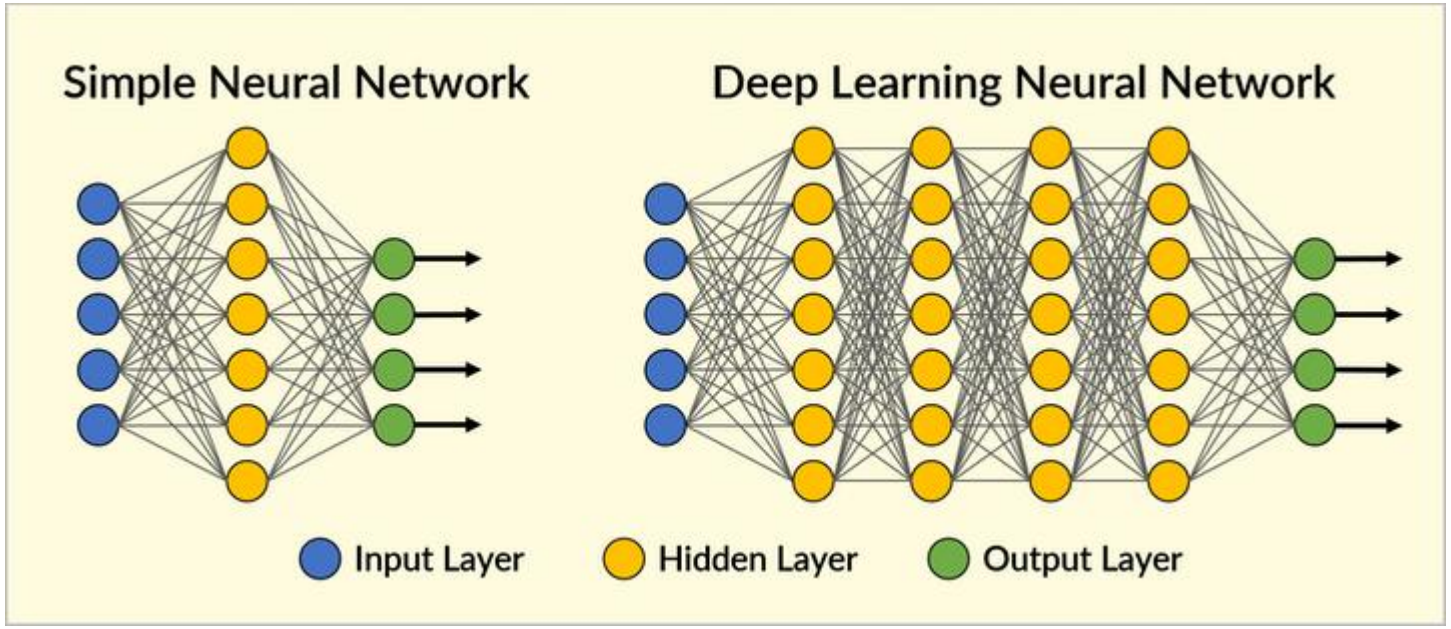
A **neural network** is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, **neural networks** refer to systems of neurons, either organic or artificial in nature.

MACHINE LEARNING AND DEEP LEARNING WILL BE THE SUB-DOMAINS OF AI



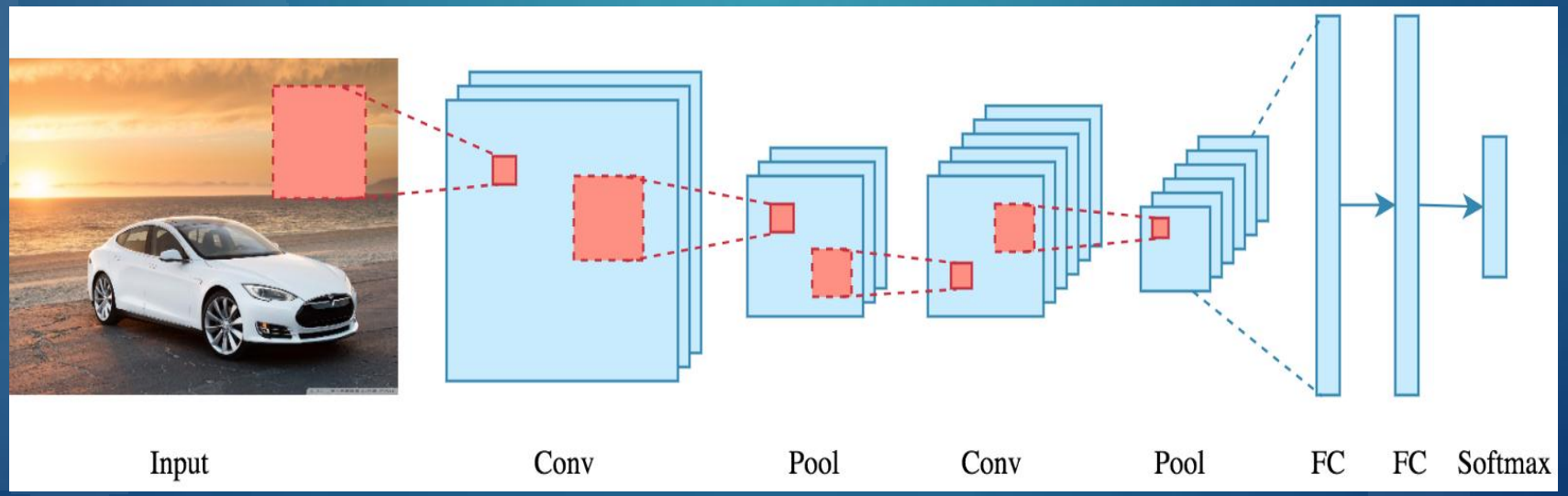
What is deep learning ?

Deep learning is an artificial intelligence (AI) function that imitates the workings of the human brain in processing data and creating patterns for use in decision making.. Also known as **deep** neural **learning** or **deep neural network**. Simply deep indicates a deep evaluation or examination with more hidden layers used for complex models .



CONVOLUTIONAL NEURAL NETWORK

- Convolutional Neural Networks or CNN is a type of deep neural networks that are efficient at extracting meaningful information from visual imagery.
- The role of the CNN is to reduce the images into a form that is easier to process, without losing features that are critical for getting a good prediction.



Parameters in CNN

- **filters:** Integer, the dimensionality of the output space (i.e. the number of output filters in the convolution).
- **kernel_size**: An integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. Can be a single integer to specify the same value for all spatial dimensions.
- **strides:** The amount by which the filter shifts is the stride
- **padding:** one of "valid" or "same" (case-insensitive). "valid" means no padding. "same" results in padding with zeros evenly to the left/right or up/down of the input such that output has the same height/width dimension as the input. Padding is simply a process of adding layers of zeros to our input images so as to avoid the problems
- **activation**: Activation function to use. If you don't specify anything, no activation is applied

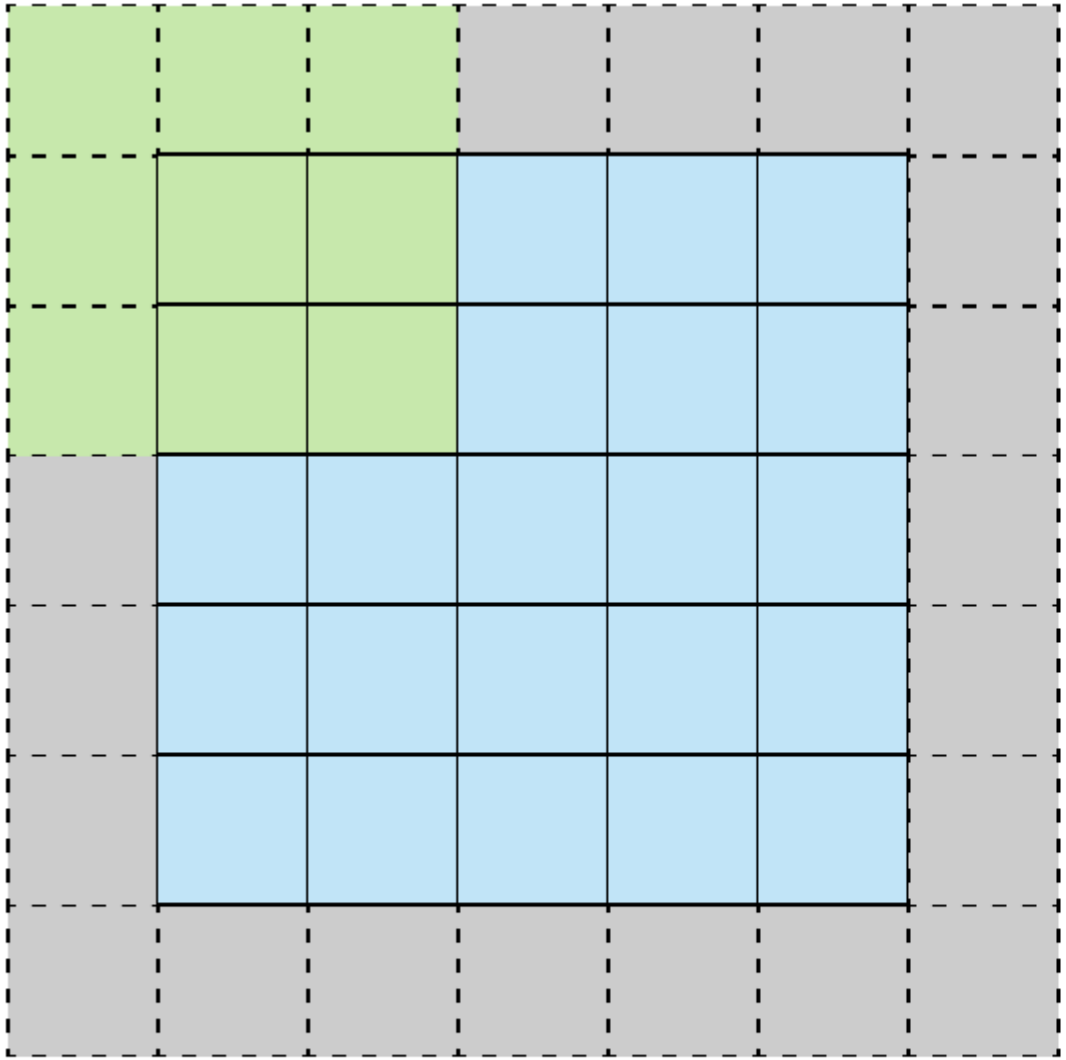
CNN layer working

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

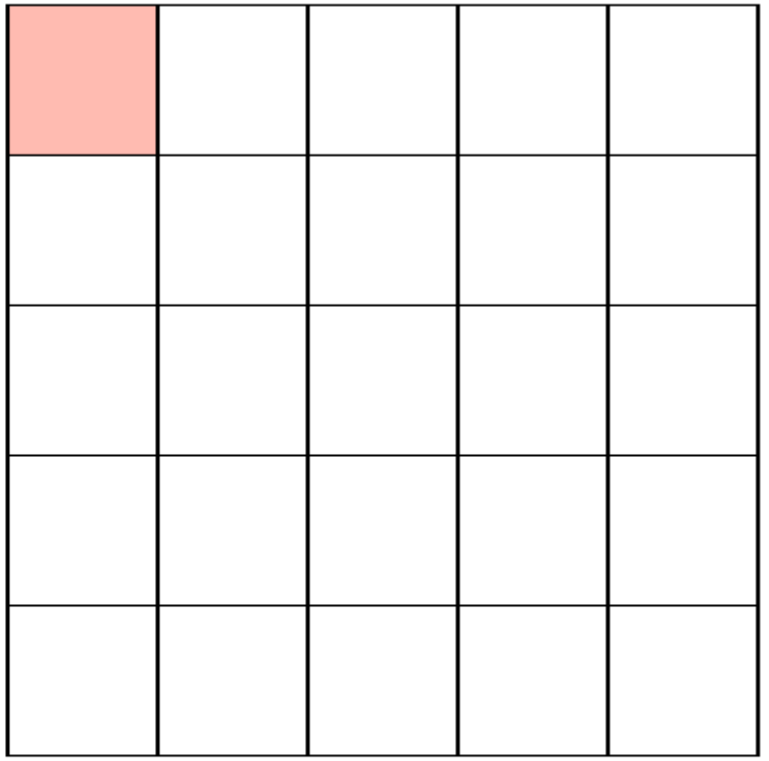
Input x Filter

4		

Feature Map



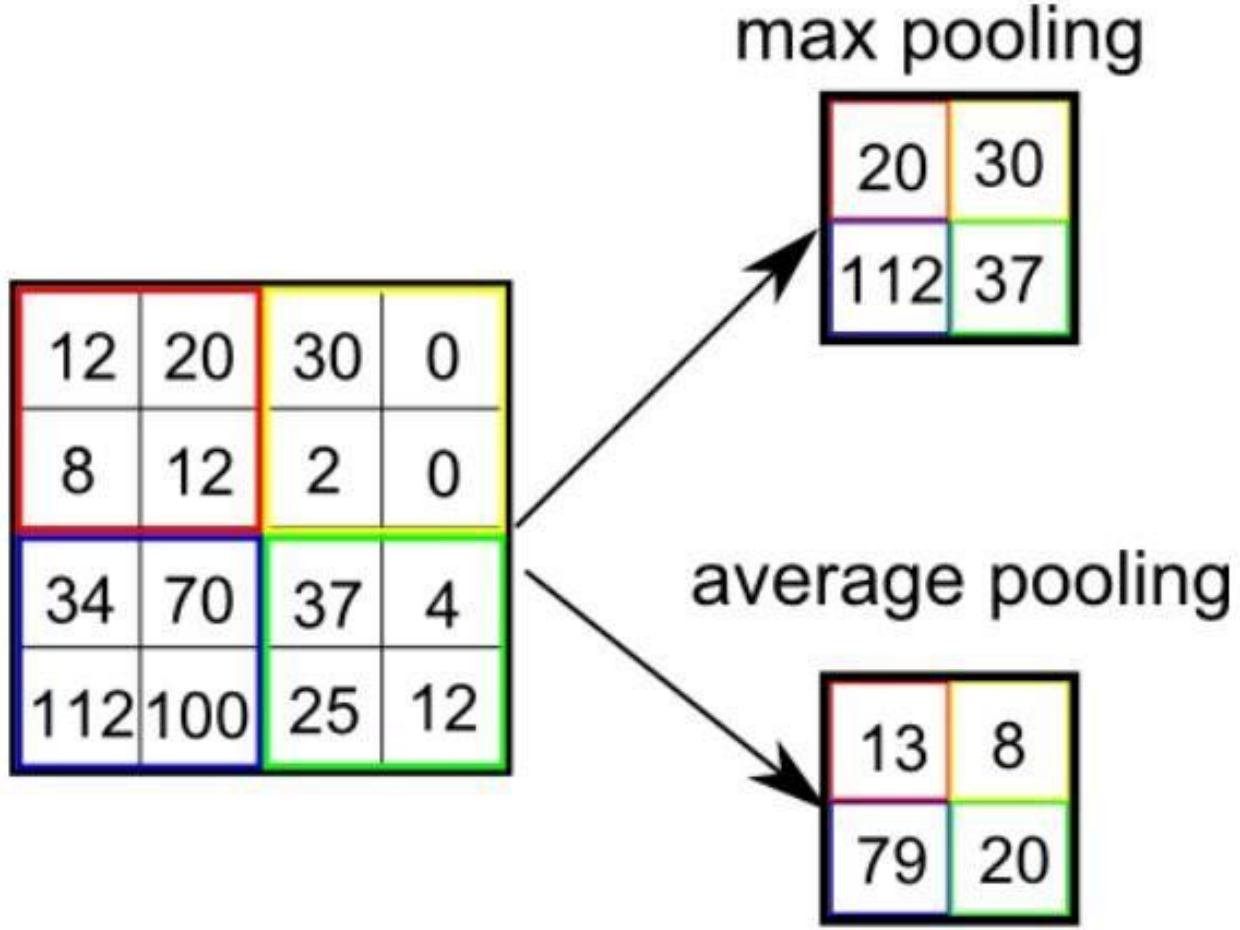
Stride 1 with Padding



Feature Map

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.



DENSE LAYER

The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most commonly used layer in the models.

Keras Dense Layer Parameters

Let us see main parameters of dense layer function of Keras below –

1. Units

The **most basic parameter** of all the parameters, it uses positive integer as it value and represents the **output size** of the layer.

It is the unit parameter itself that plays a major role in the **size of the weight matrix** along with the **bias vector**.

2. Activation

The activation parameter is helpful in applying the element-wise activation function in a dense layer. By default, Linear Activation is used but we can alter and switch to any one of many options that Keras provides for this.

ACTIVATION FUCNTIONS – RELU, SOFTMAX

RELU :

The **rectified linear activation function** or **ReLU** for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

SOFTMAX:

The softmax function squashes the outputs of each unit to be between 0 and 1, just like a sigmoid function. But it also divides each output such that the total sum of the outputs is equal to 1 (check it on the figure above).
The output of the softmax function is equivalent to a categorical probability distribution, it tells you the probability that any of the classes are true.

STEPS INVOLVED

- Import all the libraries ,packages
- Load the data from tensorflow or keras
- Split the data
- Pre-process the data into the appropriate form
  a) normalize
  b) one-hot encoding
- Start creating model
- Add all the layers sequentially
- Compile, train and validate
- Predict and print the predicted labels with respective objects

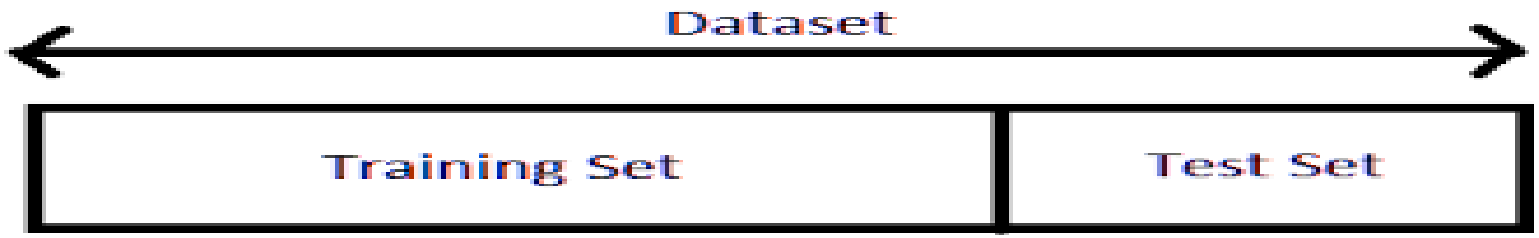
Importing dataset and splitting

LOADING DATASET

```
tf.keras.datasets.cifar10.load_data()
 Or
keras.datasets.cifar10.load_data()
```

SPLITTING OF DATASET

Further the data is splitted into training and testing set based on size of the dataset .

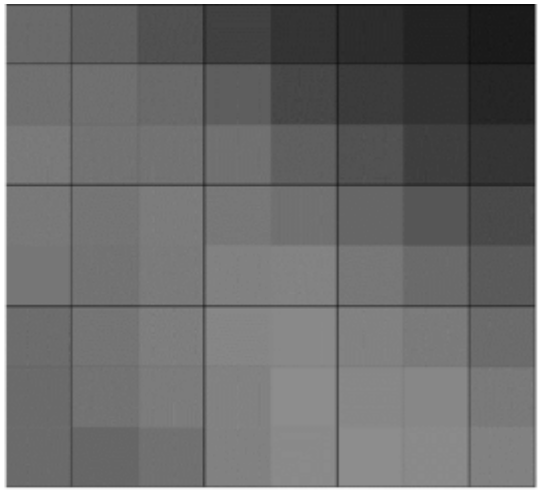


PRE-PROCESSING -NORMALIZATION

- In image processing, **normalization** is a process that changes the range of pixel intensity values.
- **Normalization** is an important step which ensures that each input parameter (pixel, in this case) has a similar data distribution. This makes convergence faster while training the network
- Most probably the images are converted to gray scale as the pixels range will be 0-255 only

107	98	82	66	53	46	36	28
113	113	107	95	72	61	52	39
123	116	116	114	97	83	64	54
122	118	125	121	114	103	87	75
119	116	124	130	132	121	108	92
110	118	127	135	138	131	124	114
108	116	125	131	141	137	136	123
108	104	114	130	139	141	139	130

a

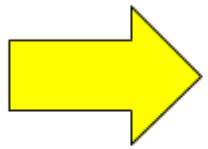


b

ONE hot encoding

One-hot is a group of bits among which the legal combinations of values are only those with a single high bit and all the others low.

Color
Red
Red
Yellow
Green
Yellow



Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

IMPORTING LAYERS and ADDING LAYERS

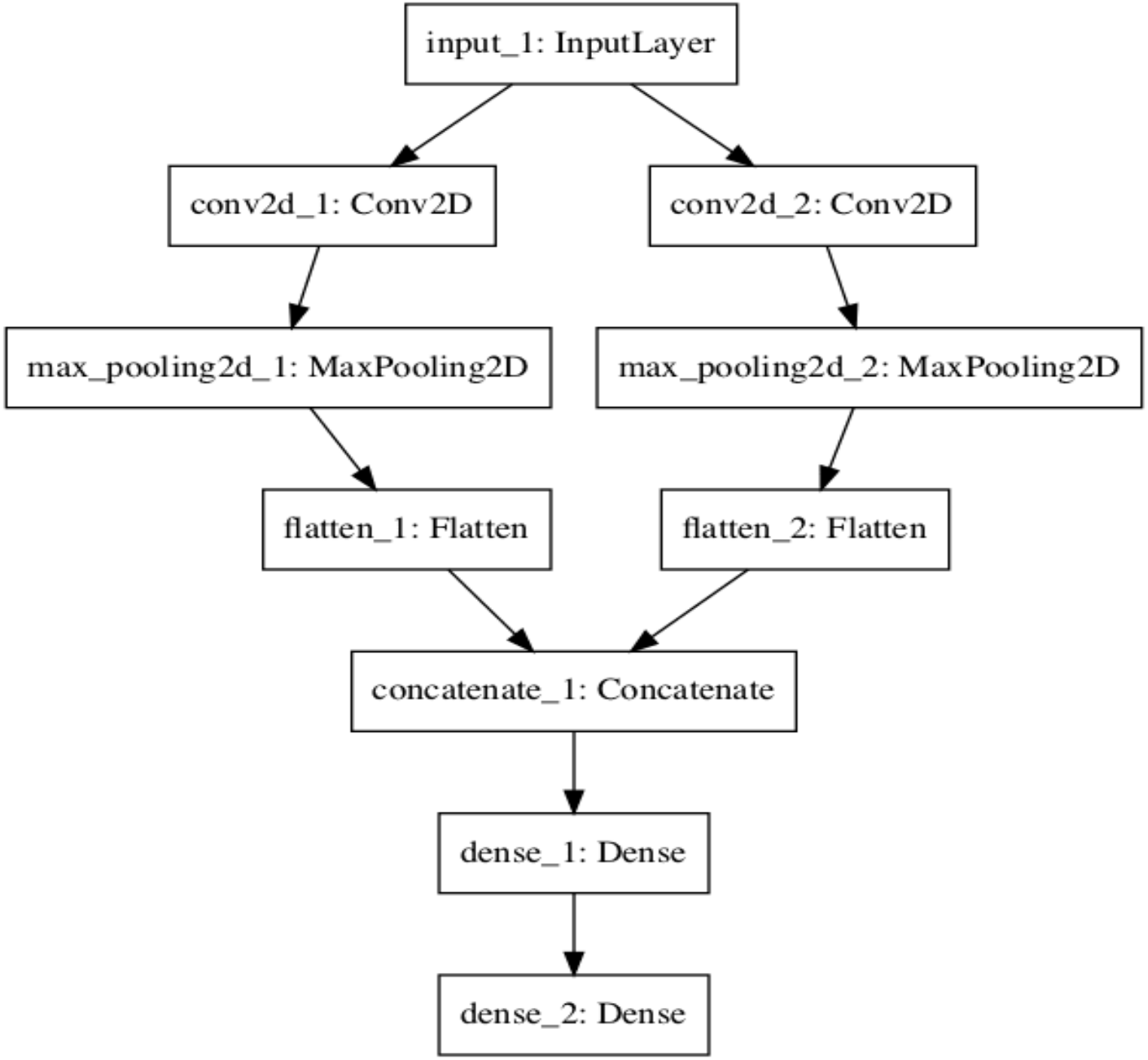
FROM KERAS

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Conv2D, MaxPool2D, Flatten
```

From tensorflow

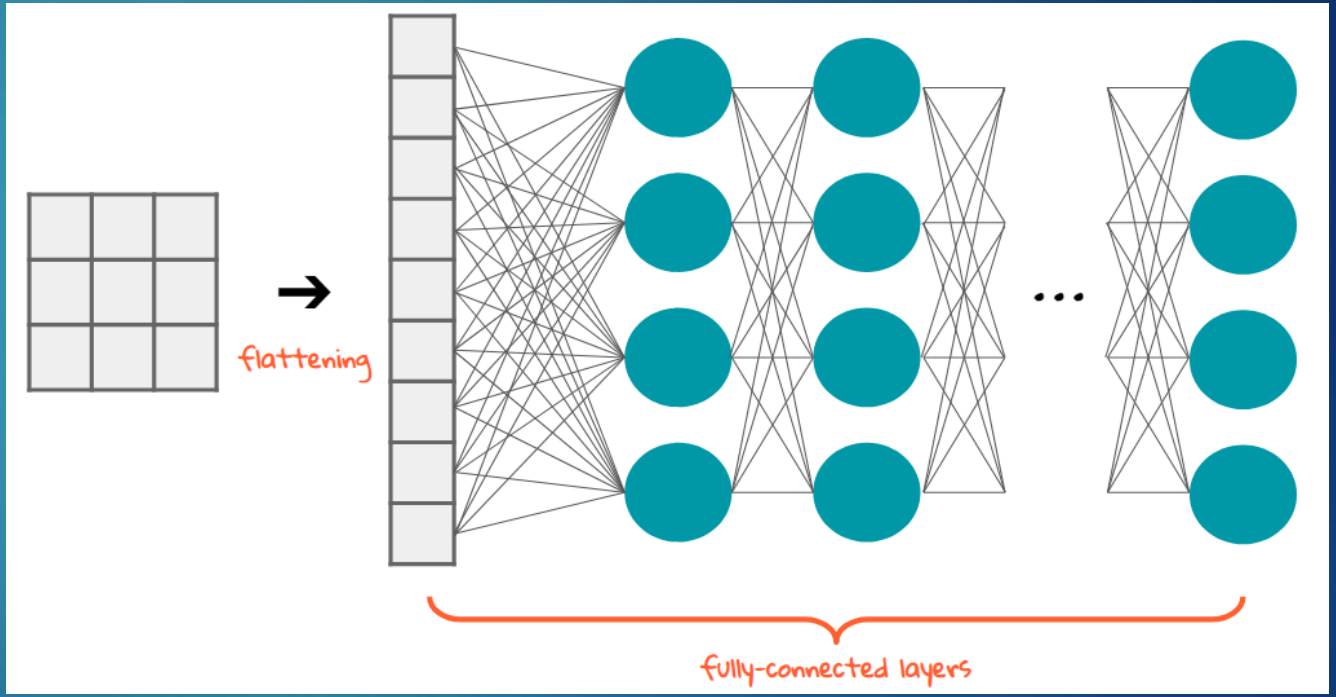
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, Dropout, Flatten,
 MaxPooling2D
```

We add layer my using add () function in sequential order as we chose sequential model


input_1: InputLayer
conv2d_1: Conv2D
conv2d_2: Conv2D
max_pooling2d_1: MaxPooling2D
max_pooling2d_2: MaxPooling2D
flatten_1: Flatten
flatten_2: Flatten
concatenate_1: Concatenate
dense_1: Dense
dense_2: Dense

FLATTEN LAYER

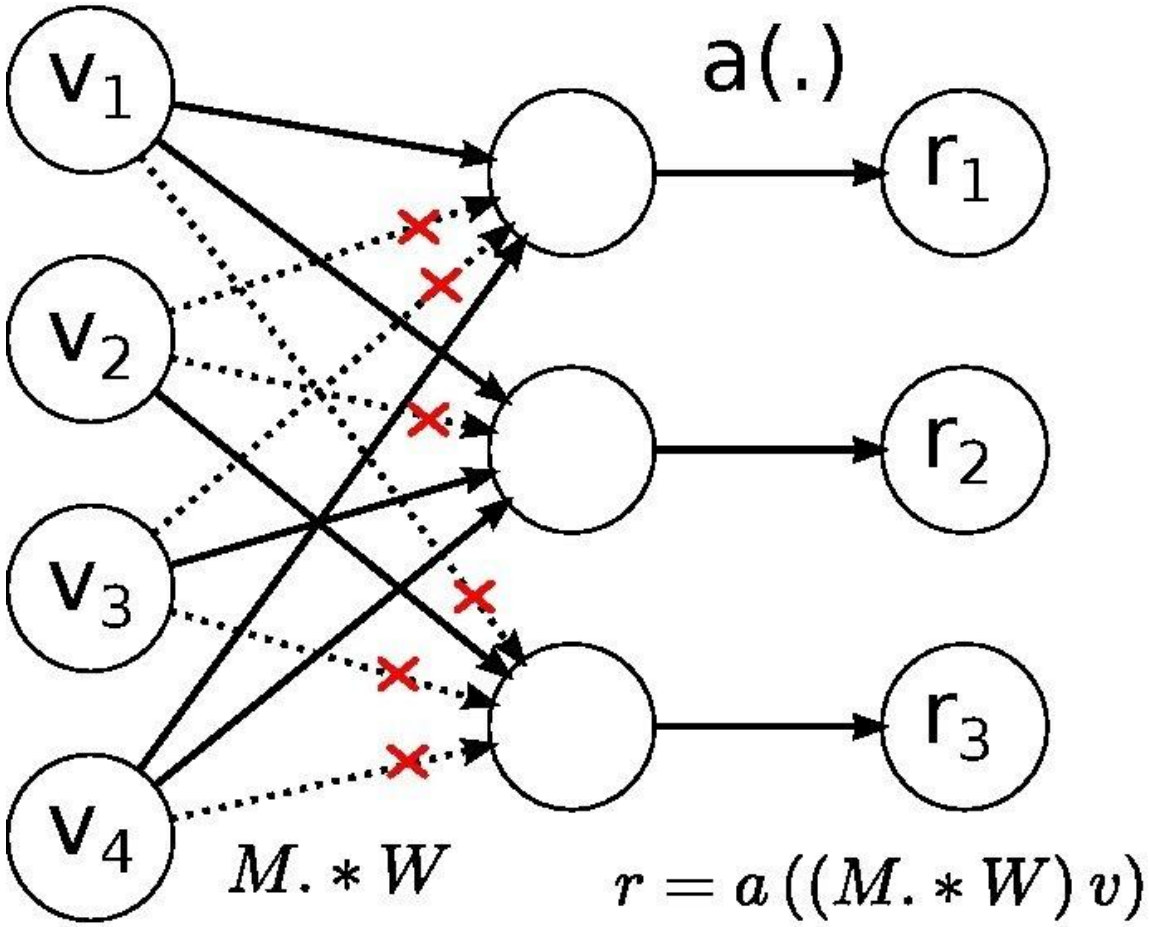
- **Flattening** is converting the data into a 1-dimensional array for inputting it to the next **layer**. We **flatten** the output of the convolutional **layers** to create a single long feature vector.
- And it is connected to the final classification model, which is called a fully-connected **layer**



Dropout

Dropout is a way to regularize the neural network. During training, it may happen that neurons of a particular layer may always become influenced only by the output of a particular neuron in the previous layer. In that case, the neural network would overfit.

Dropout prevents overfitting and regularizes by randomly cutting the connections (also known as dropping the connection) between neurons in successive layers during training



Compiling model

Compile defines the loss function, the optimizer and the metrics. It has nothing to **do** with the weights and you can **compile** a **model** as many times as you want without causing any problem to pretrained weights. You need a **compiled model** to train (because training uses the loss function and the optimizer

```
compile(
   optimizer,
   loss = None,
   metrics = None,
   loss_weights = None,
   sample_weight_mode = None,
   weighted_metrics = None,
   target_tensors = None
)
```

The important arguments are as follows −loss function,Optimizer,metrics

LOSS :

The error for the current state of the model must be estimated repeatedly. This requires the choice of an error function, conventionally called a **loss function**, that can be used to estimate the loss of the model so that the weights can be updated to reduce the loss on the next evaluation.

METRICS :

Metrics is used to evaluate the performance of your model. It is similar to loss function, but not used in training process. Keras provides quite a few metrics as a module, **metrics** and they are as follows

- accuracy
- binary_accuracy
- categorical_accuracy
- sparse_categorical_accuracy

OPTIMIZER :

Optimization is an important process which optimize the input weights by comparing the prediction and the loss function. Keras provides quite a few optimizer as a module, *optimizers* **.** Adam is the **best optimizers**. If one wants to train the **neural network** in less time and more efficiently than Adam is the **optimizer**.

TRAINING MODEL

- `model.fit(X_train, Y_train, batch_size=128, epochs=20, validation_data=(X_test, Y_test))`

- the **batch size** is a number of samples processed before the **model** is updated.
- The number of epochs is the number of complete passes through the training dataset.
- The **size** of a **batch** must be more than or equal to one and less than or equal to the number of samples in the training dataset.

PREDICTING- BATCH , VERBOSE

BATCHES : Keras can **predict** multiple values at the same time, like if you input a vector of 100 elements, **Keras** can compute one **prediction** for each element, giving 100 outputs. This computation can also be done in **batches**

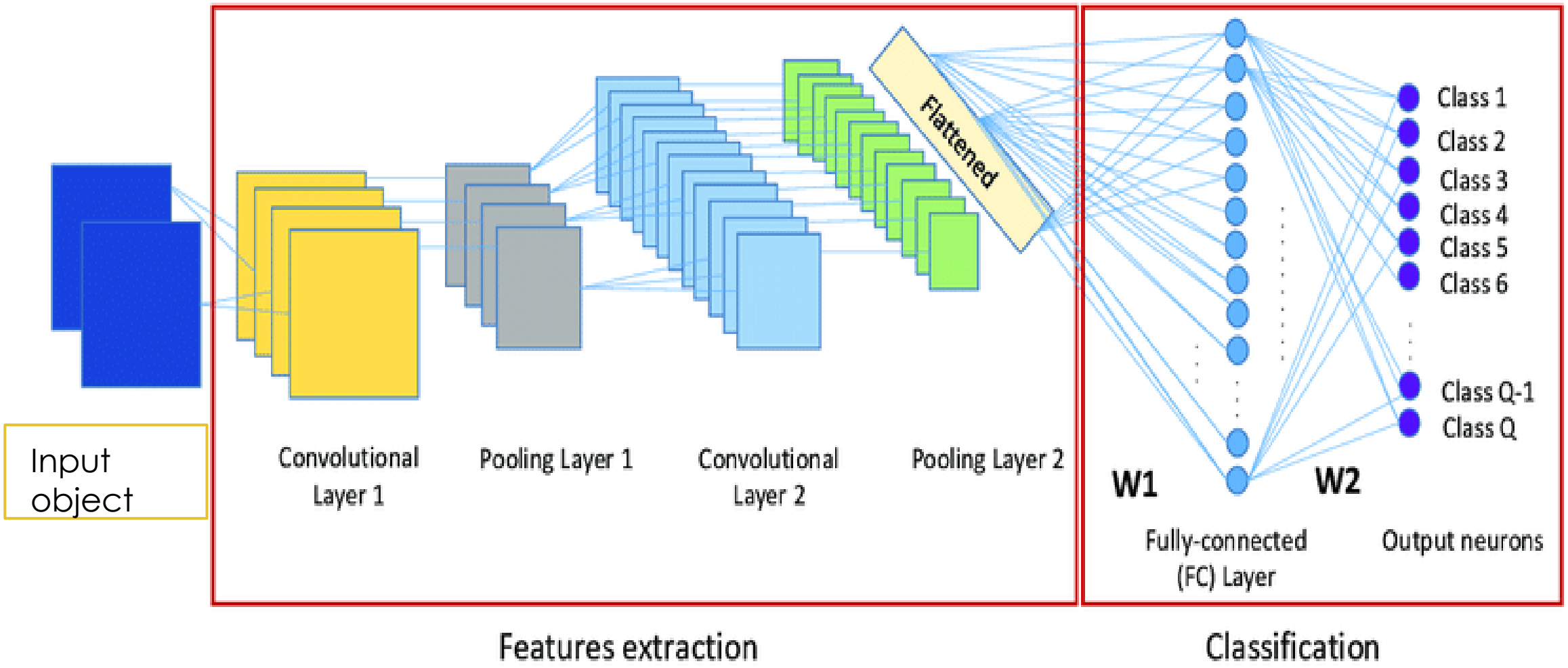
VERBOSE :

By setting verbose 0, 1 or 2 you just say how do you want to 'see' the training progress for each epoch.

verbose=0 will show you nothing (silent)

verbose=1 will show you an animated progress bar like this: [==========]

Verbose=2 will just mention the number of epoch like this: Epoch 1/10


Input object
Convolutional Layer 1
Pooling Layer 1
Convolutional Layer 2
Pooling Layer 2
Flattened
Class 1
Class 2
Class 3
Class 4
Class 5
Class 6
Class Q-1
Class Q
W1
W2
Fully-connected (FC) Layer
Output neurons
Features extraction
Classification