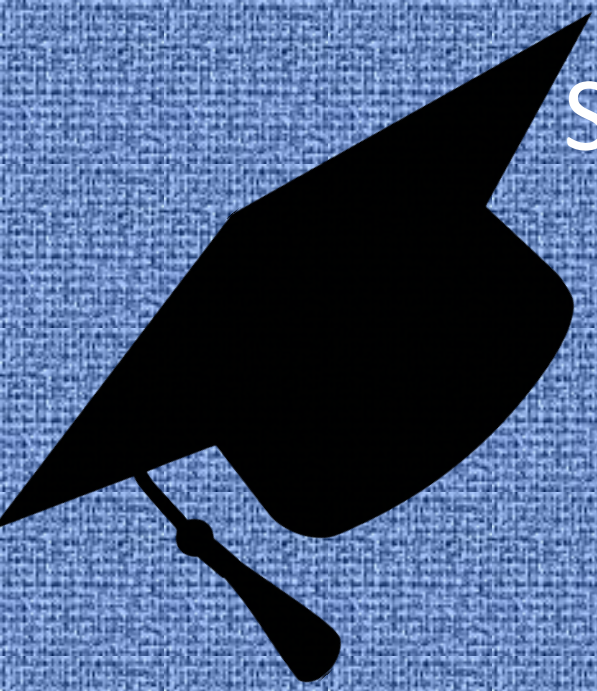


SELF DRIVING CAR
سيارة ذاتية القيادة

تقديم الطلاب :

حسن زينو – عمر الشيوخ

طالب العلي – سليمان الرستم

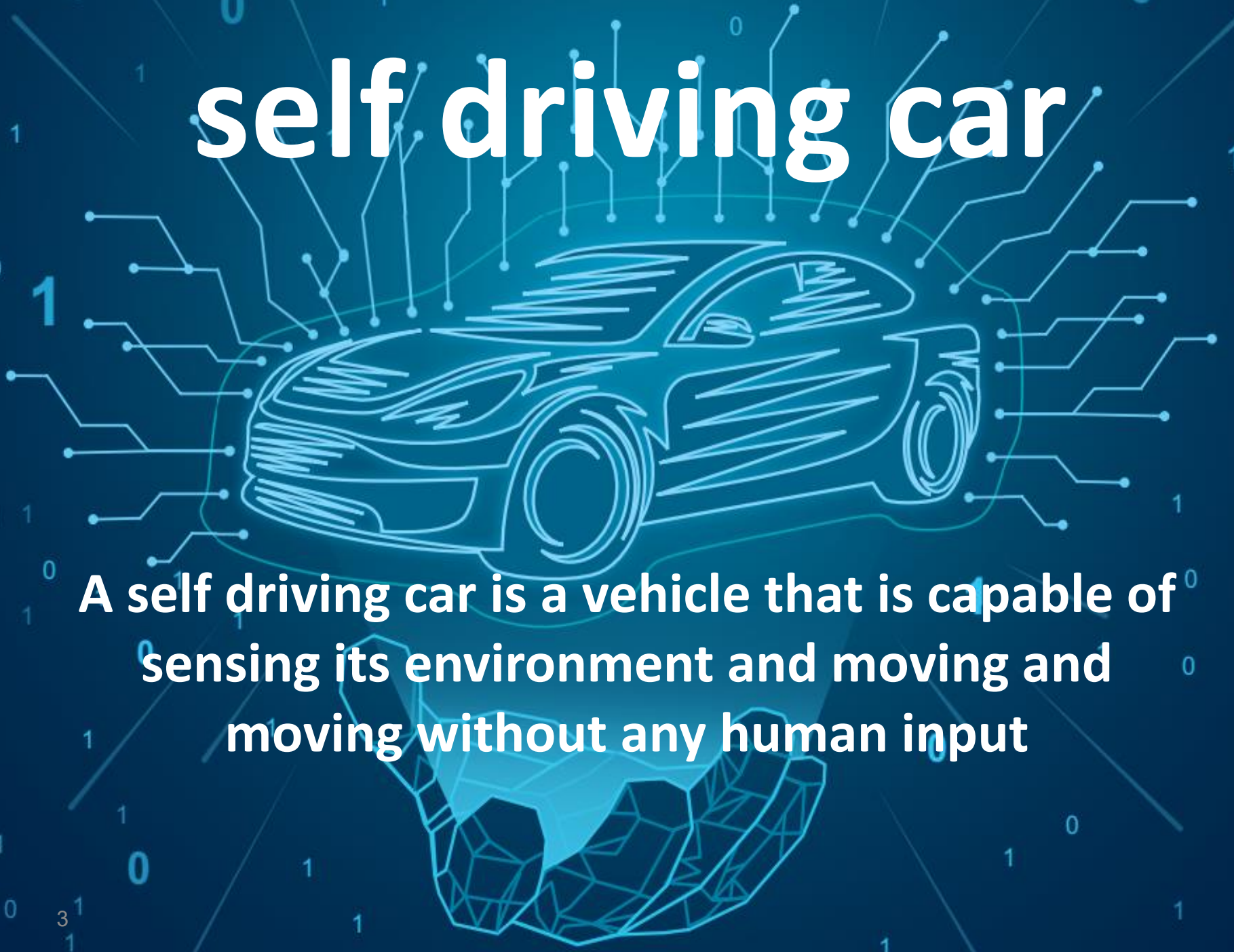
اشراف :

الدكتورة : يسر سليمان الاتاسي

Artificial intelligence (AI)

an area of computer science that emphasizes the creation of intelligent machines that work and react like human

self driving car



A self driving car is a vehicle that is capable of sensing its environment and moving and moving without any human input


ANGLE
-0.77°
0.70
MPH
esc
MENU

Tools used

KERAS

TENSORFLOW

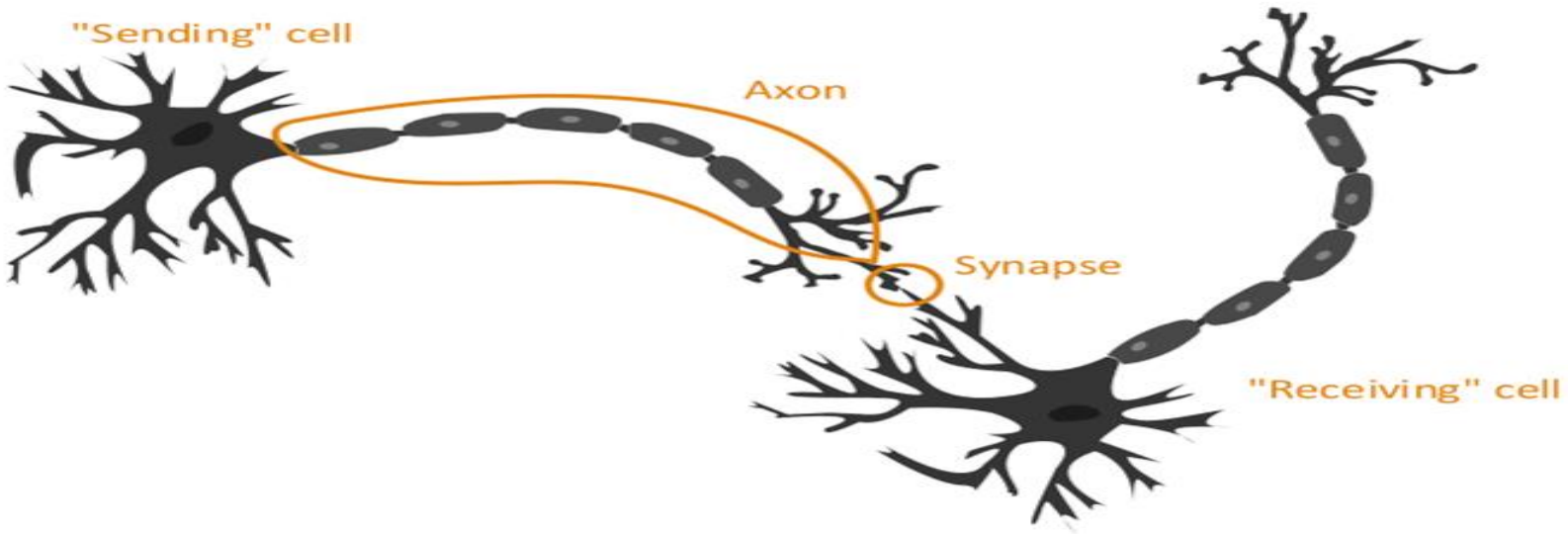
Classification

In machine learning and statistics, classification is a supervised learning approach in which the computer program learns from the data input given to it and then uses this learning to classify new observation.

Its types: Linear Classifiers, Logistic Regression, Naive Bayes Classifier.

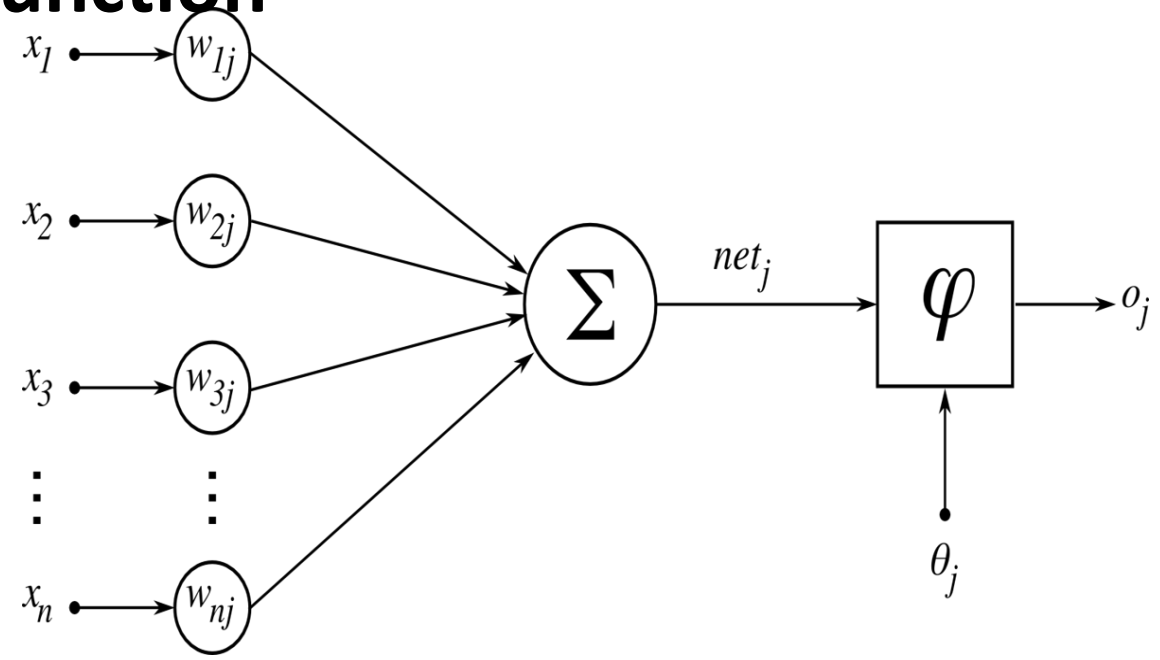
Artificial Neural Networks

We can define neural networks as a mathematical experiment to simulate the way the human brain works.

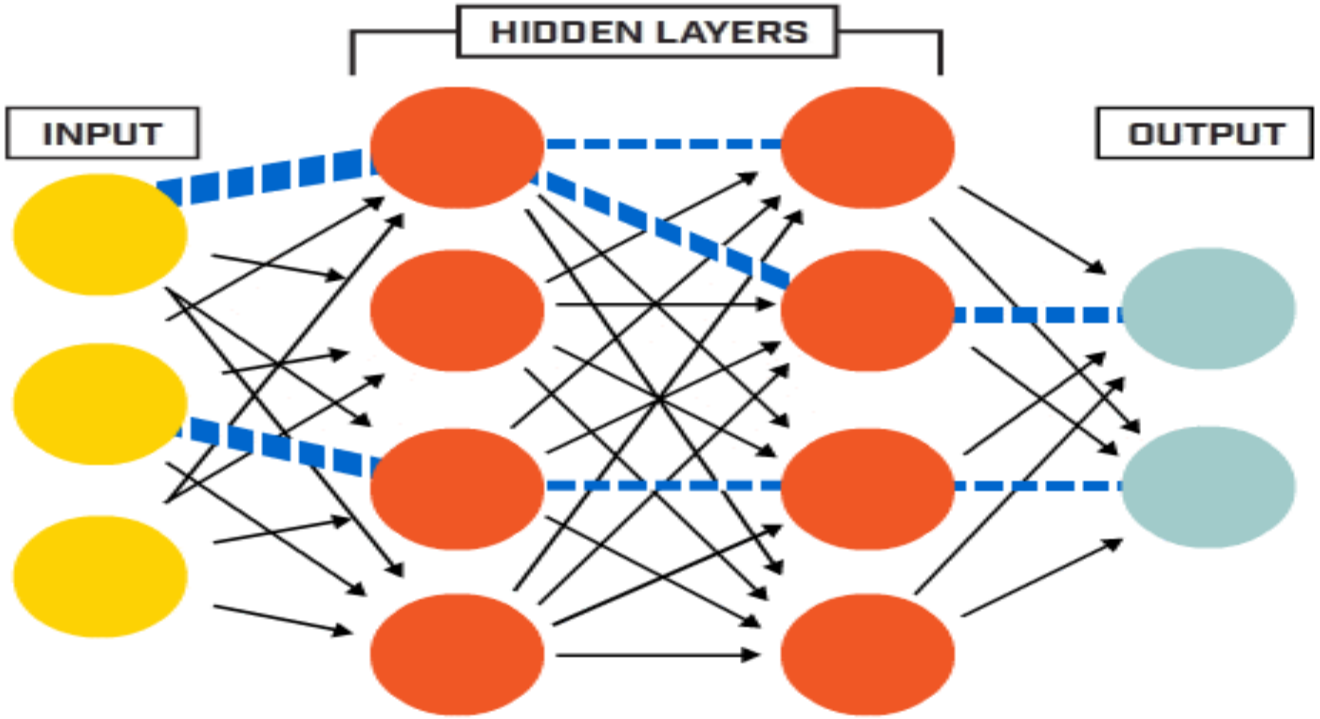


Components of an artificial neural network

- weights and biases
- Propagation function
- Learning rule

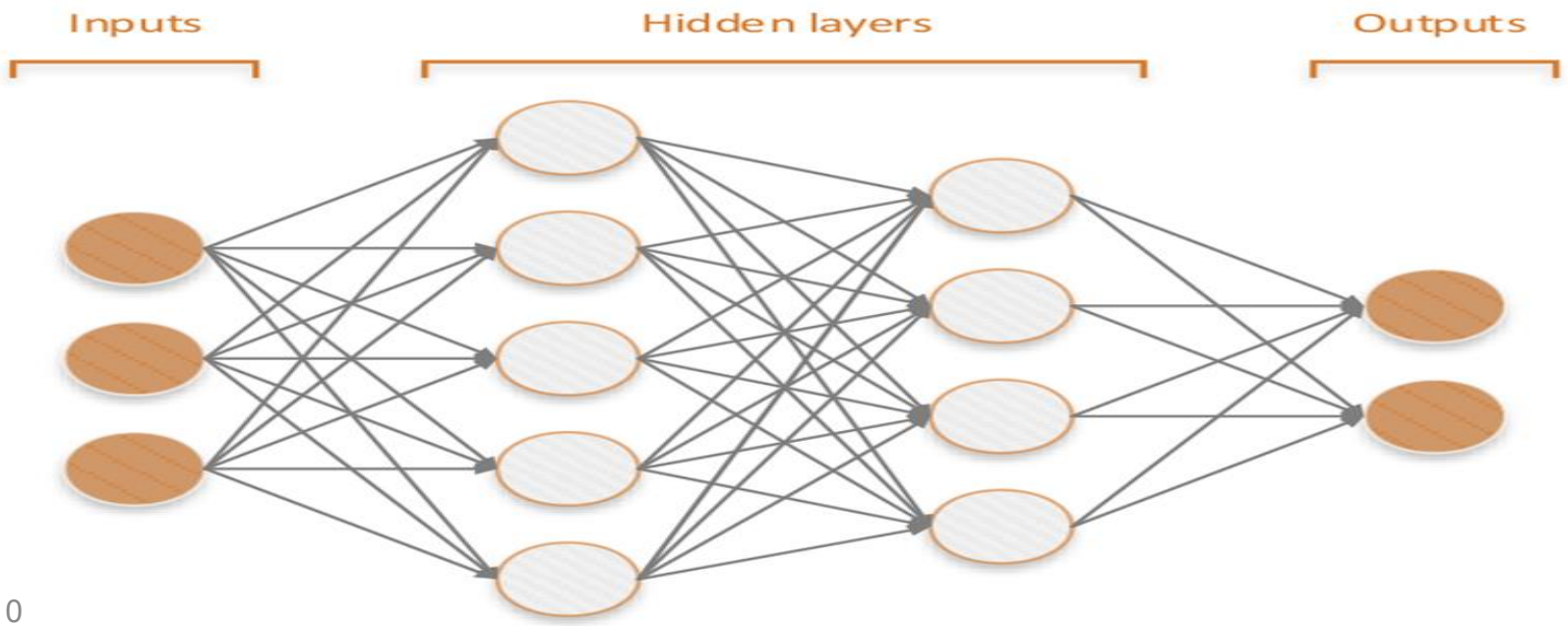


How does artificial neural networks work?



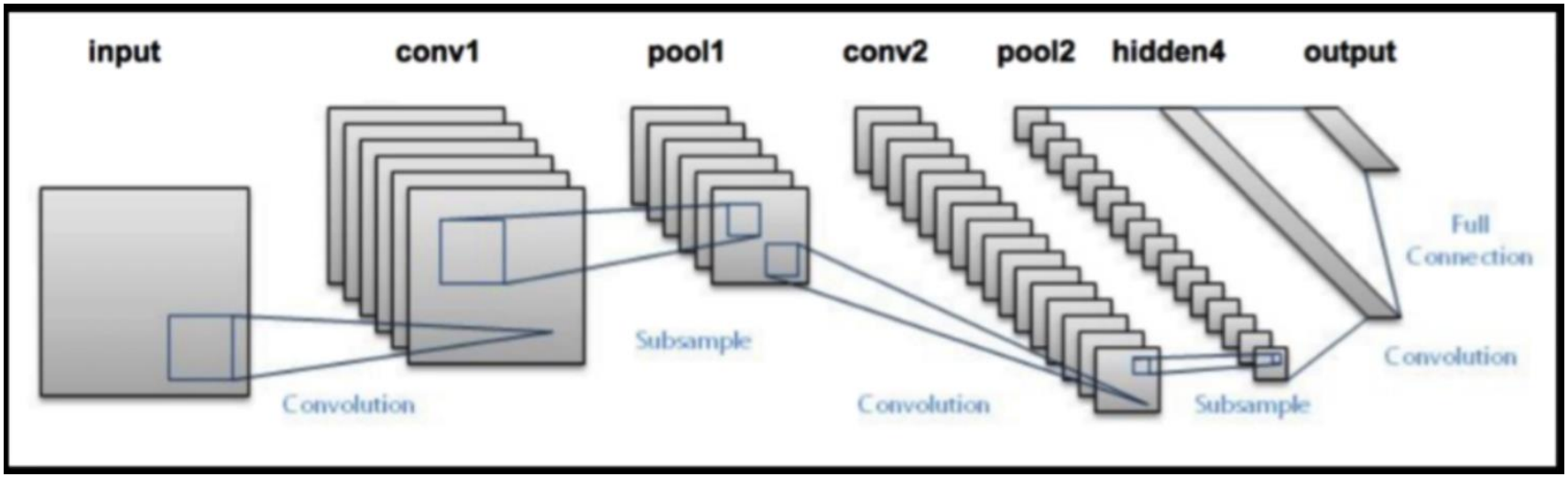
Feed-forward Neural Networks

Feed-Forward networks because there are no cycles, the input signals go through the network from the first (input) layer to the last (output) one directly. If there are other layers between the input and output ones, they are called hidden layers.



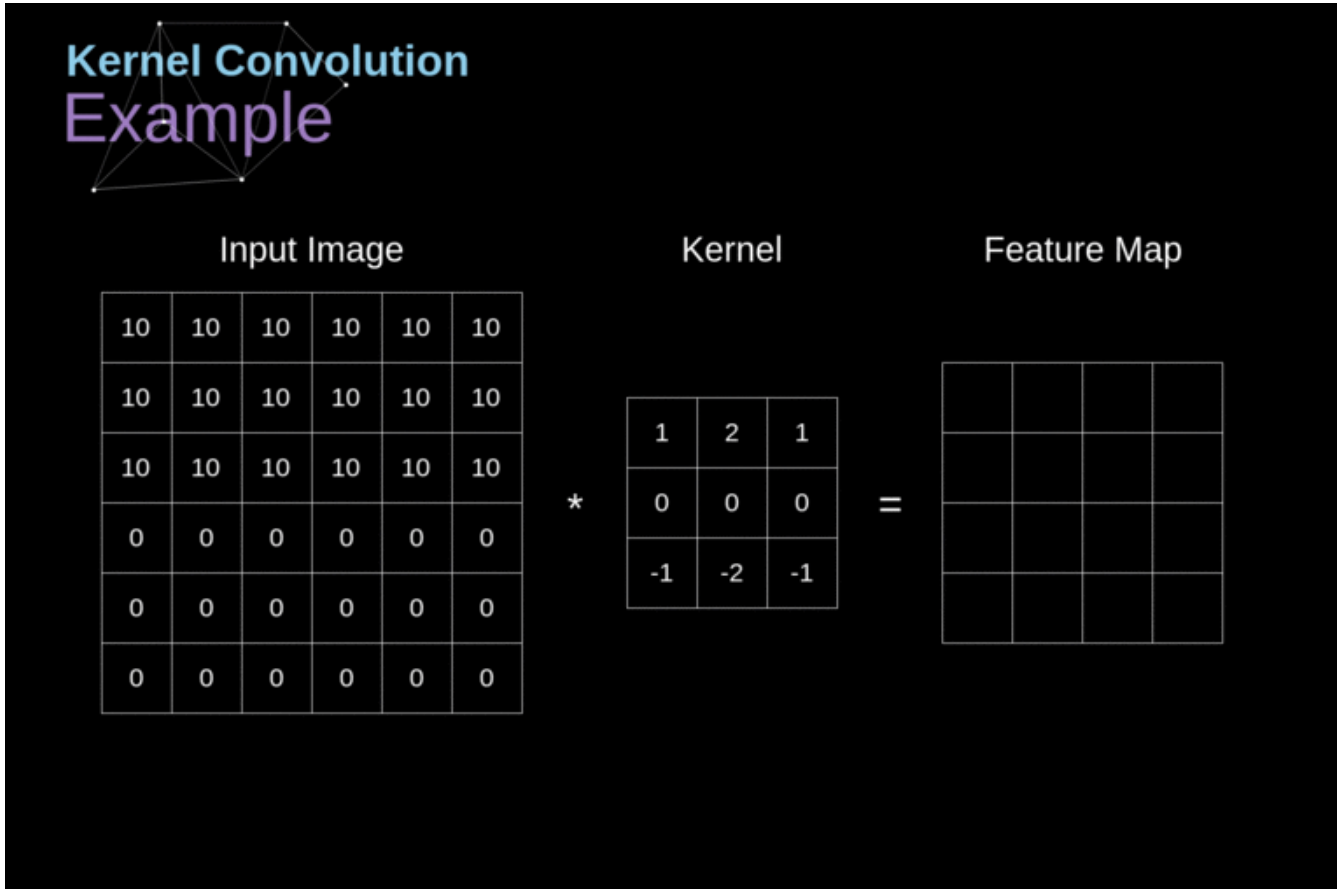
CNN

In deep learning, a **convolutional neural network** (**CNN**, or **ConvNet**) is a class of deep neural networks, most commonly applied to analyzing visual imagery



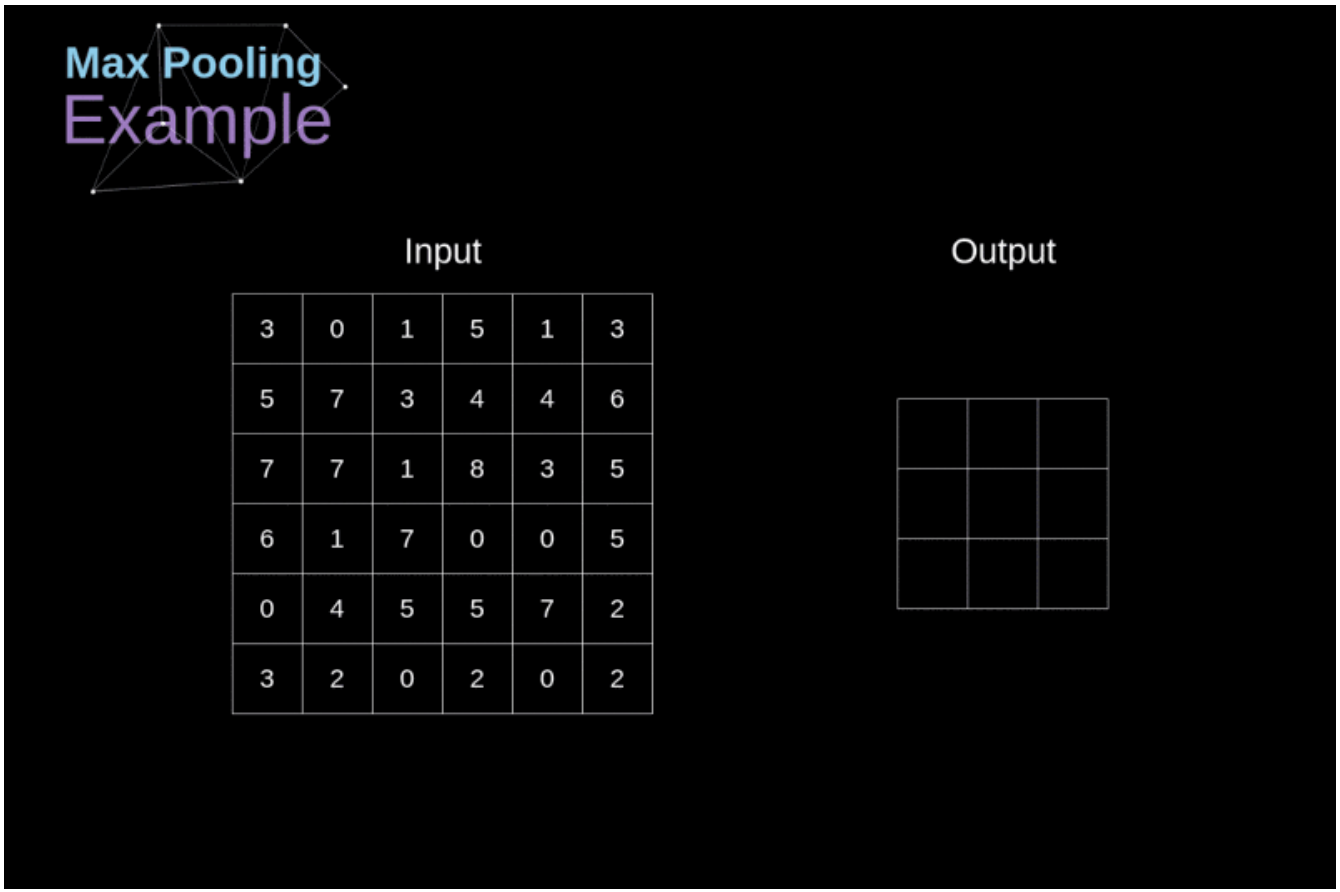
Design CNN

Convolutional layer

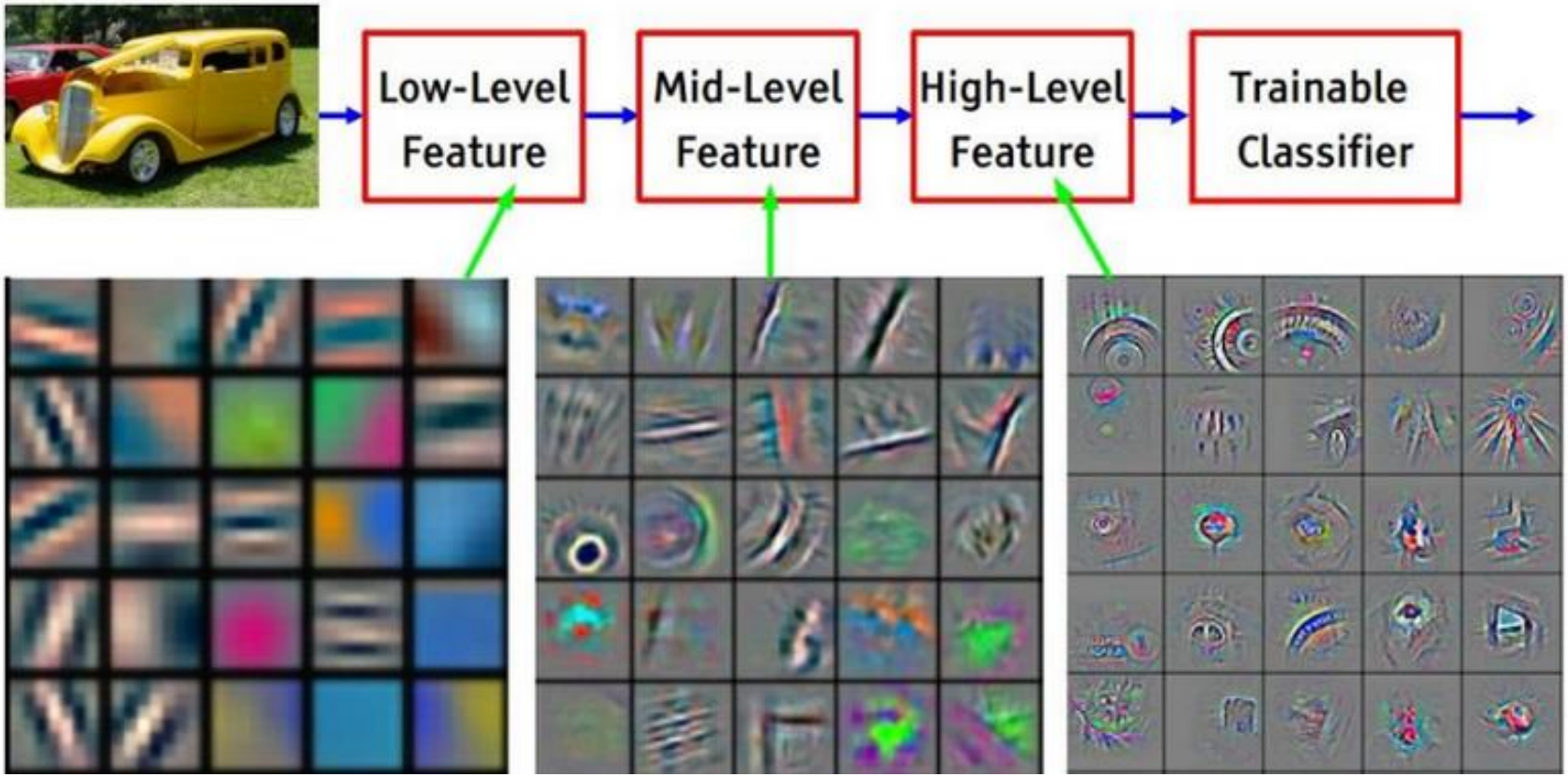


Design CNN

Pooling



Feature levels



Design CNN

Fully connected

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

Changing values of filter matrix

Traffic sign classification

```
!git clone https://bitbucket.org/jadslim/german-traffic-signs
```

```
[ ]  !ls german-traffic-signs
```

```
signnames.csv  test.p  train.p  valid.p
```

Then we order them according to id's

	ClassId	SignName
0	0	Speed limit (20km/h)
1	1	Speed limit (30km/h)
2	2	Speed limit (50km/h)
3	3	Speed limit (60km/h)
4	4	Speed limit (70km/h)
5	5	Speed limit (80km/h)
6	6	End of speed limit (80km/h)
7	7	Speed limit (100km/h)
8	8	Speed limit (120km/h)
9	9	No passing
10	10	No passing for vechiles over 3.5 metric tons
11	11	Right-of-way at the next intersection
12	12	Priority road
13	13	Yield
14	14	Stop
15	15	No vechiles
16	16	Vechiles over 3.5 metric tons prohibited
17	17	No entry
18	18	General caution
19	19	Dangerous curve to the left
20	20	Dangerous curve to the right
21	21	Double curve

Image preprocessing

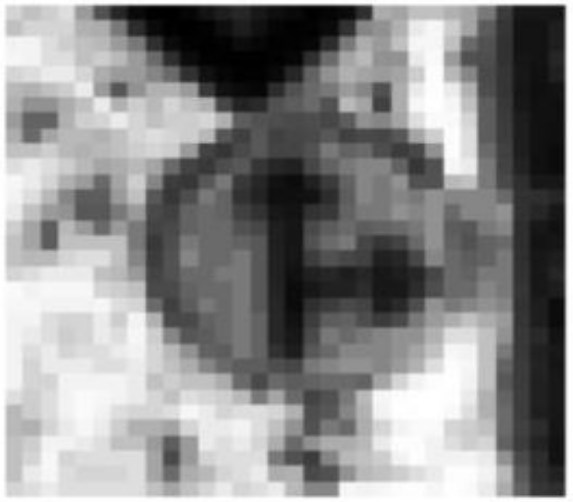
1-To GrayScale

```
def grayscale(img):
  img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
  return img

img = grayscale(X_train[1000])
plt.imshow(img)
plt.axis("off")
```

2-Equalize using histogram

```
[ ]  def equalize(img):
         img = cv2.equalizeHist(img)
         return img
     img = equalize(img)
     plt.imshow(img)
     plt.axis("off")
     print(img.shape)
```



Traffic Sign Classification

```
def leNet_model():
    model = Sequential()
    model.add(Conv2D(30, (5, 5), input_shape=(32, 32, 1), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Conv2D(15, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Flatten())
    model.add(Dense(500, activation = 'relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))
    #Compile model
    model.compile(Adam(lr = 0.01), loss = 'categorical_crossentropy', metrics = ['accuracy'])
    return model
```

```
#Test image
print("predicted sign: "+ str(model.predict_classes(img)))
```

```
predicted sign: [23]
```

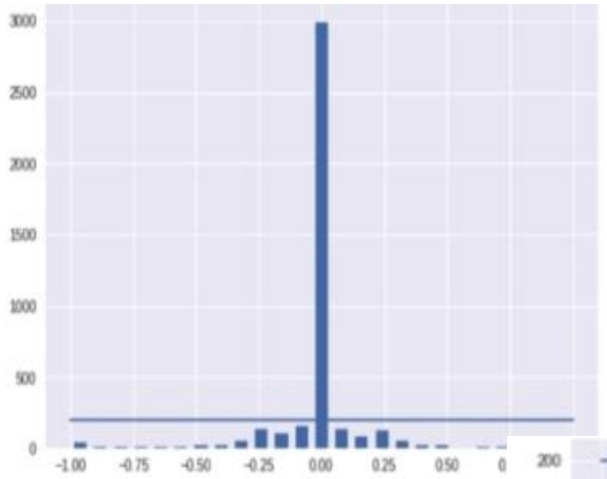
Now lets move to the project workflow

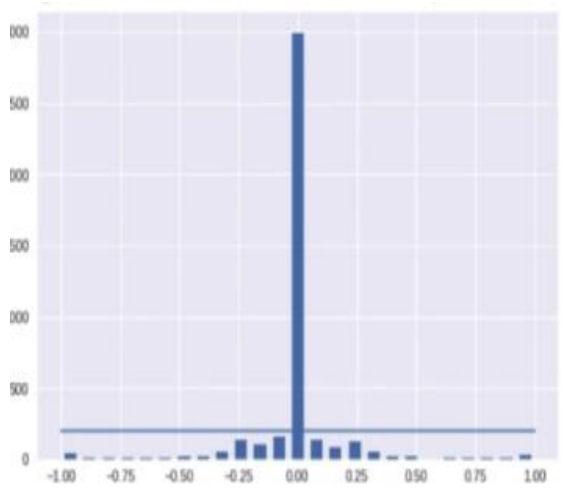
First step: is to collect our data :

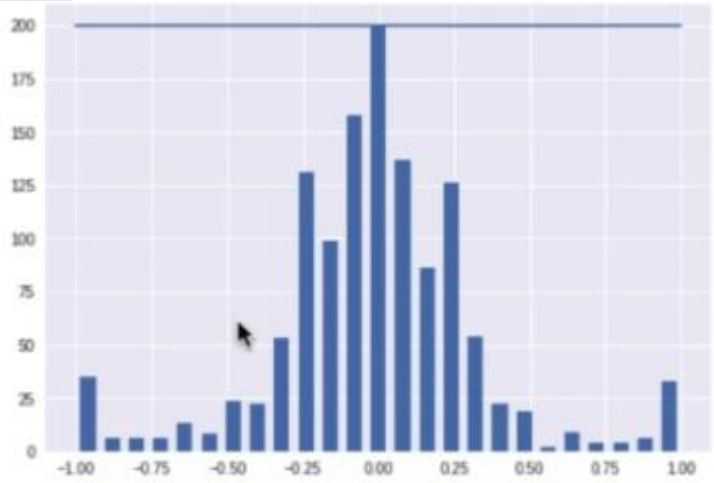
The Udacity simulator record the road and the angle and the speed of our car and image of the road on that information.

Second step is to Prepare the data :

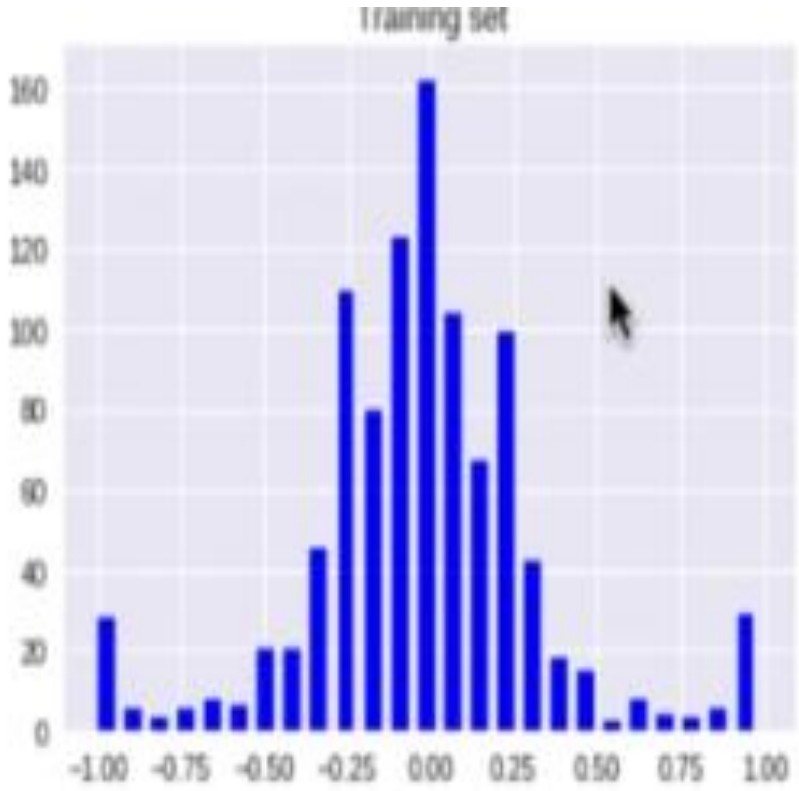
Now we need to upload the data to git hup to use it with coolab and then we need to make our data balance .

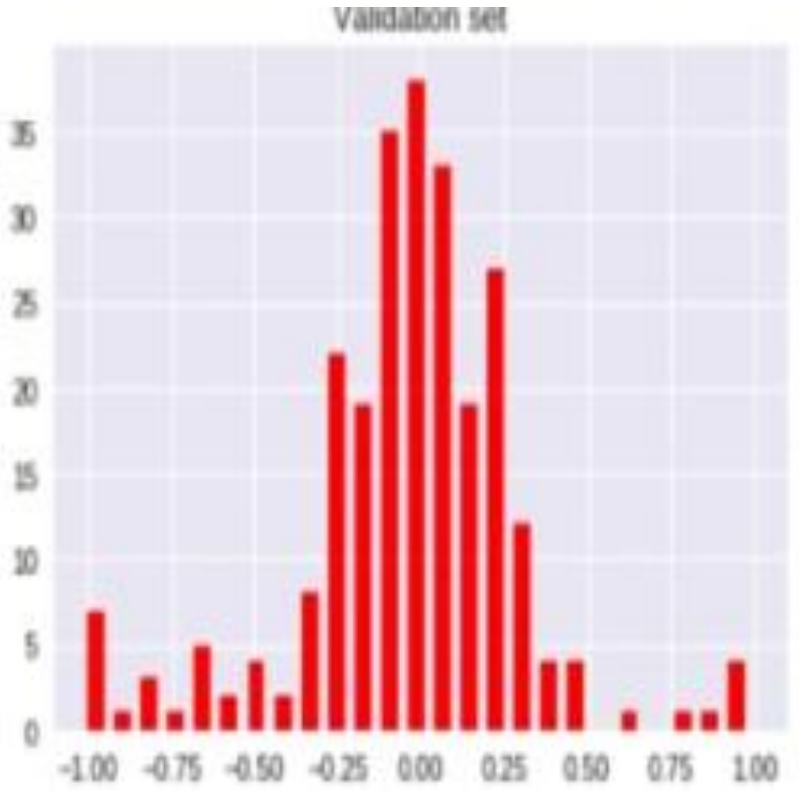






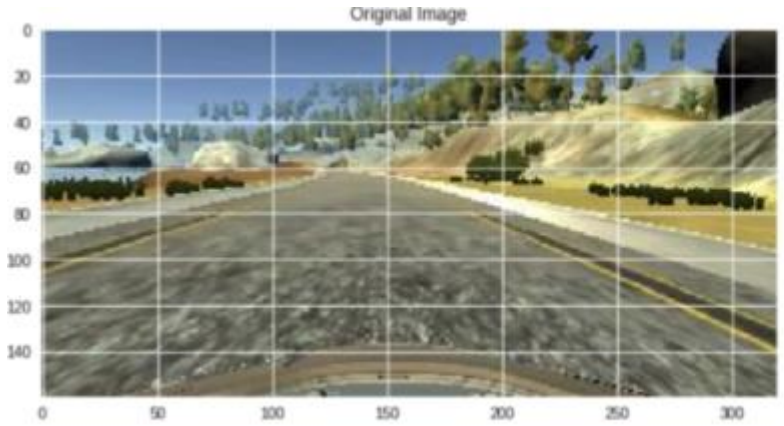
Third step is split the data into training and validation

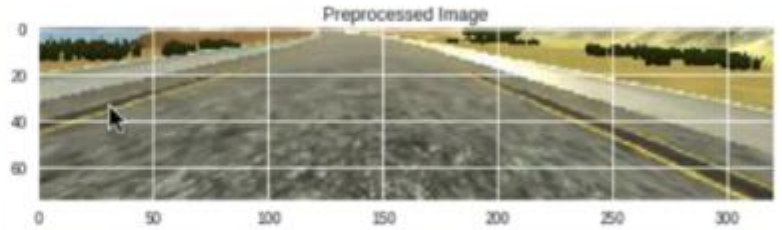


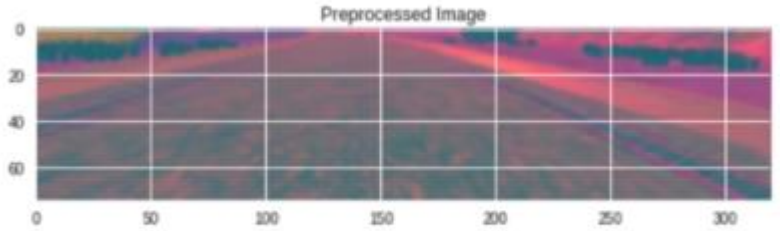


Forth step is image preprocessing

Now we need to make the image in the training sit good for our model .

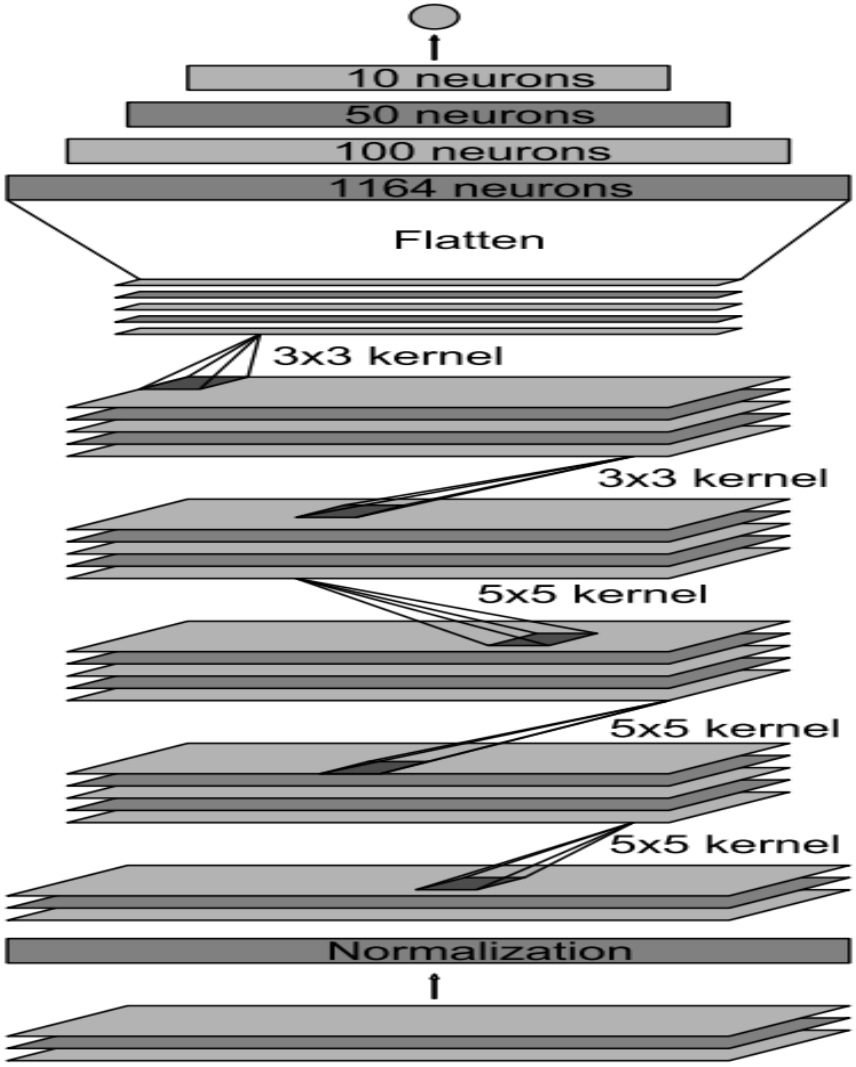






The fifth step is to define the model

The model we will use is NVidia model



شكرا لحضوركم

THANKS FOR COMING