औद्योगिक प्रशिक्षण के लिए राष्ट्रीय संस्थान

National Institute for Industrial Training

One Premier Organization with Non Profit Status | Registered Under Govt. of WB
Empanelled Under Planning Commission Govt. of India
Inspired By: National Task Force on IT & SD Government of India

National Institute for Industrial Training- One Premier Organization with Non Profit Status Registered Under Govt.of West Bengal,Empanelled Under Planning Commission Govt.of India , Empanelled Under Central Social Welfare Board Govt. of India , Registered with National Career Services , Registered with National Employment Services.

SUBJECT – TRAFFIC SIGN RECOGNITION SYSTEM

SUBMITTED BY – DEBANJAN SAHA

EMAIL – sahadebanjan9433@gmail.com

LinkedIn - www.linkedin.com/in/debanjan-saha-08b9391a4

Github - https://github.com/Debanjan-123

SUBMITTED TO – Mr. SOUMOTANU MAZUMDAR

STUDENT PROFILE

NAME – DEBANJAN SAHA

COLLEGE – UNIVERSITY OF ENGINEERING & MANAGEMENT KOLKATA

COURSE – B.TECH (CSE)

DEPARTMENT – COMPUTER SCIENCE & ENGINEERING

YEAR – 3RD YEAR (5TH SEMESTER)

EMAIL – sahadebanjan9433@gmail.com

PHONE NO – +91 9475951336

YEAR OF PASSING – 2023

PROJECT TOPIC – TRAFFIC SIGN RECOGNITION SYSTEM

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them. I am highly indebted to **NATIONAL INSTITUTE FOR INDUSTRIAL TRAINING** for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. I would like to express my gratitude towards **Mr. SOUMOTANU MAZUMDAR** for his support, co-operation and encouragement which helped me for completion of this project.

CONTENT

ABSTRACT

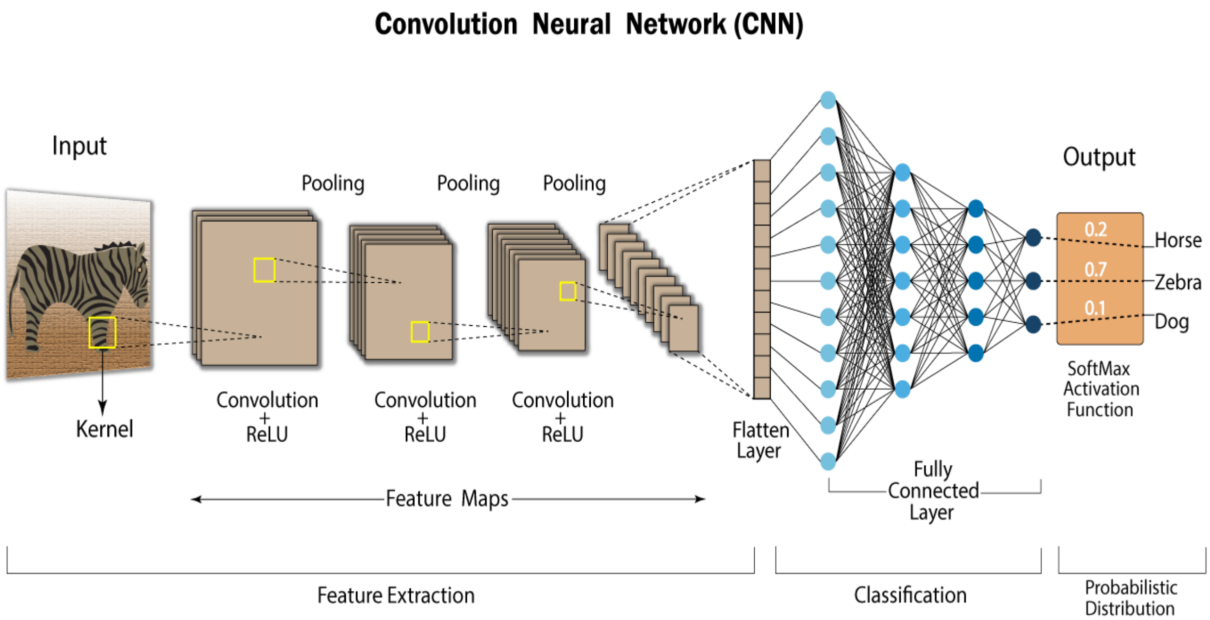
This project uses computer vision and machine learning along with deep learning techniques to predict the Traffic Sign Classification Model using CNN on the **German Traffic Sign Recognition Benchmark** (GTSRB) Dataset and then detecting the images of Indian Traffic Signs using the same Dataset which will be used as testing dataset while building classification model. Finally, any relevant sign is highlighted and output to the screen.

GTSRB Dataset

INTRODUCTION

In case of self driving cars in India there are multiple roads where traffic signs do not survive and are mostly broken are covered in trees or any hindrance. Traffic signs are mostly damaged and distorted by the civilians itself which causes more traffic sign violations which results in such conditions which will hardly be able to recognize the traffic sign and may cause serious problems for all vehicles surrounding it. Road safety is always a problem everywhere, especially in developed countries like the US, India etc. Thus it is necessary to develop a system which will help to recognize traffic signs without being affected by these anomalies. Here the same model is Implemented using Convolution Neural Networks with filter size 32, kernel size (5, 5) with Activation Function 'Rectified Linear Unit (ReLu)' along with Max Pooling of Pool Size (2, 2), Drop Out Rate 0.25 and finally the model is trained as vectors and converted into flatten layer after that again Activation Function 'ReLu' is called for the Activation Purpose of the Neural Network and finally the output is predicted.

In machine learning, Convolution Neural Networks (CNN) is complicated feed forward neural networks. CNNs are used for image classification and recognition due to its high accuracy. CNN follows a hierarchal model that specializes in processing data that has a grid like topology, such as an image. A digital image is a binary representation of visual data. It contains a series of pixels arranged in a grid like fashion that contains pixel values to denote the colour of each pixel. We present experiments on source and accuracy trade-offs and exhibits live achievements in contrast to different appreciated models on ImageNet classification.



MOTIVATION

Though driverless AI is advancing rapidly with the support of American giants like Google and Tesla; the technology is not yet mainstream. In the Indian scenario, however, technical problems persist. This exclusive Indian data provide much-needed impetus to the autonomous industry not only in India but across the world. Thus it is necessary to develop a system which will help recognize traffic signs without being affected by these anomalies.

OBJECTIVE

In this paper our aim is to implement Image Classification with Deep learning and Convolution Neural Network using Tensorflow. Different techniques have been applied for this purpose and a comparative study has been made between different accuracies shown by different models.

1. **Data Mining** - Data mining is the process of finding anomalies, patterns and correlations within large data sets to predict outcomes.
2. **Data Cleaning** - Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset.
3. **Data Preprocessing** - Data preprocessing can refer to manipulation or dropping of data before it is used in order to ensure or enhance performance, and is an important step in the data mining process.
4. **Exploratory Data Analysis** - It refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.
5. **Dividing into Testing & Training Set** - Data splitting is the act of partitioning available data into two portions; usually for cross-valedictory purposes. One portion of the data is used to develop a predictive model and the other to evaluate the model's performance.
6. **Applying Various Classification Models** - Various classification models were used to predict the probability of a particular problem like stroke in patients, after taking into consideration various other factors.

DATASET – The Dataset used here is GTSRB is a deep learning model with almost **50K images data of all classes**. The German Traffic Sign

Recognition Benchmark (GTSRB) contains **43 classes of traffic signs, split into 31,367 training images and 7842 test images**. The images have varying light conditions and rich backgrounds.

The CNN architecture model has all the following layers

Convolution Layers, Filter – 32, Kernel size - (5, 5), Activation – ReLu

Convolution Layers, Filter – 32, Kernel size – (5, 5), Activation – ReLu

Max Pool Layers, Pool Size – 2*2

Dropout Rate – 0.25

Convolution Layers, Filter – 64, Kernel size - (3, 3), Activation – ReLu

Convolution Layers, Filter – 64, Kernel size – (3, 3), Activation – ReLu

Max Pool Layers, Pool Size – 2*2

Dropout Rate – 0.25

Flatten Layer ()

Dense – 256, Activation – ReLu

Dropout Rate – 0.5

Dense – 43, Activation – Softmax

In all the Layers ReLu Activation Function used except the last layer which is Softmax Activation Function.

LITERATURE SURVEY

Human vision is a complex process which is still not understood completely. Computer vision is a technological implementation of human vision that enables computers to achieve human vision capabilities.

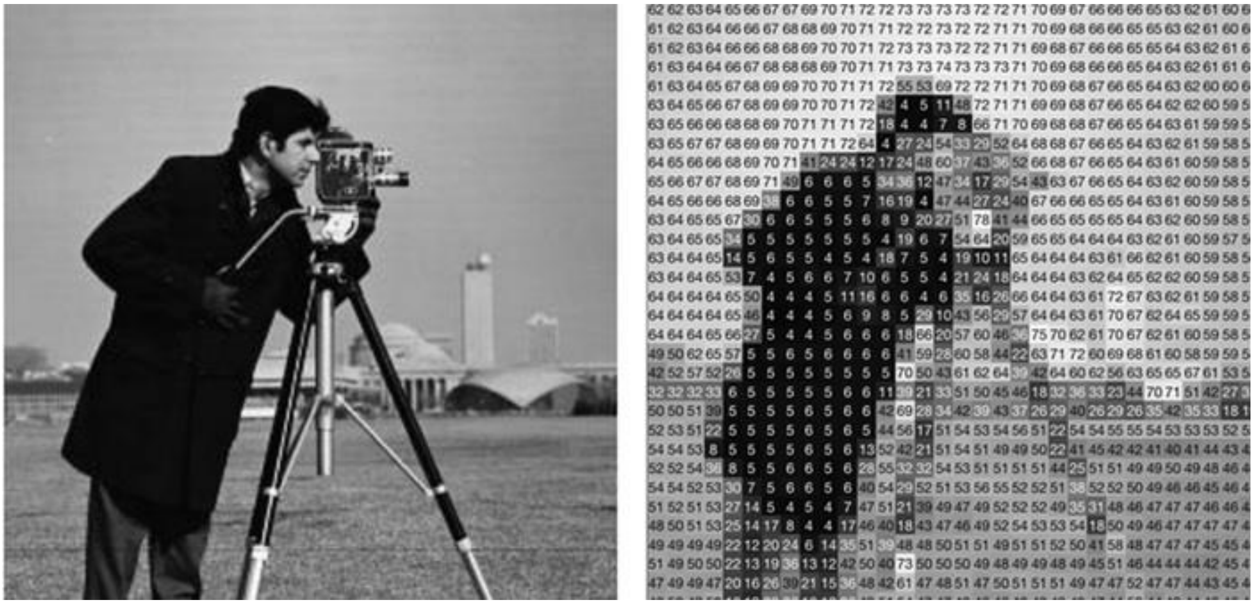


Fig: Human Vision Vs Computer Vision

Human Vision

The Human eye is a sense organ, part of sensory nervous system, which is capable of receiving images which are basically relayed to the brain.
First, light bounces off the image and enters the eyes through the cornea. Then, the cornea directs light to the pupils and iris, which work together to control the amount of light entering the eye. Once the light passes through the cornea, it enters the retina; the retina has special sensors called cones and rods, which are involved in colour vision.

Computer Vision

This subpart of Artificial Intelligence enables computer and systems to derive meaningful information from digital images. **Teaching a computer to see like a human is difficult** because we still do not really understand how human vision works. This means that computers can make inferences about images without human assistance. This seems simple because humans can effortlessly see the world around them.

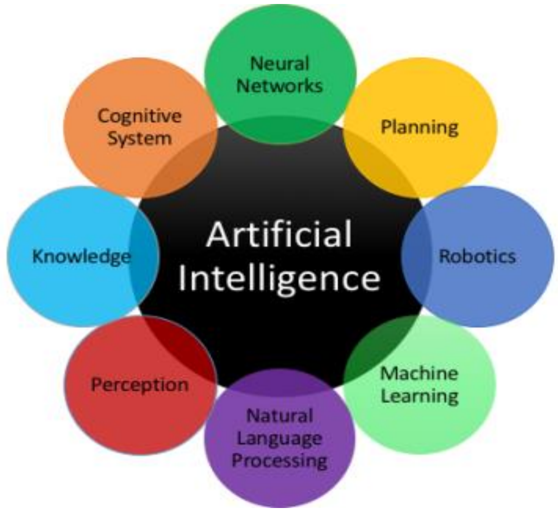
RESEARCH

After doing various surveys on various existing literature on fine-tuned image classification, opencv, tensorflow and keras and other machine learning models we first use the prexisting dataset GTSRB. Then we highlight the advanced methods and their comparative study in this paper. We will learn about the structure of the model used which will help us to better understand the underlying principal under each model.

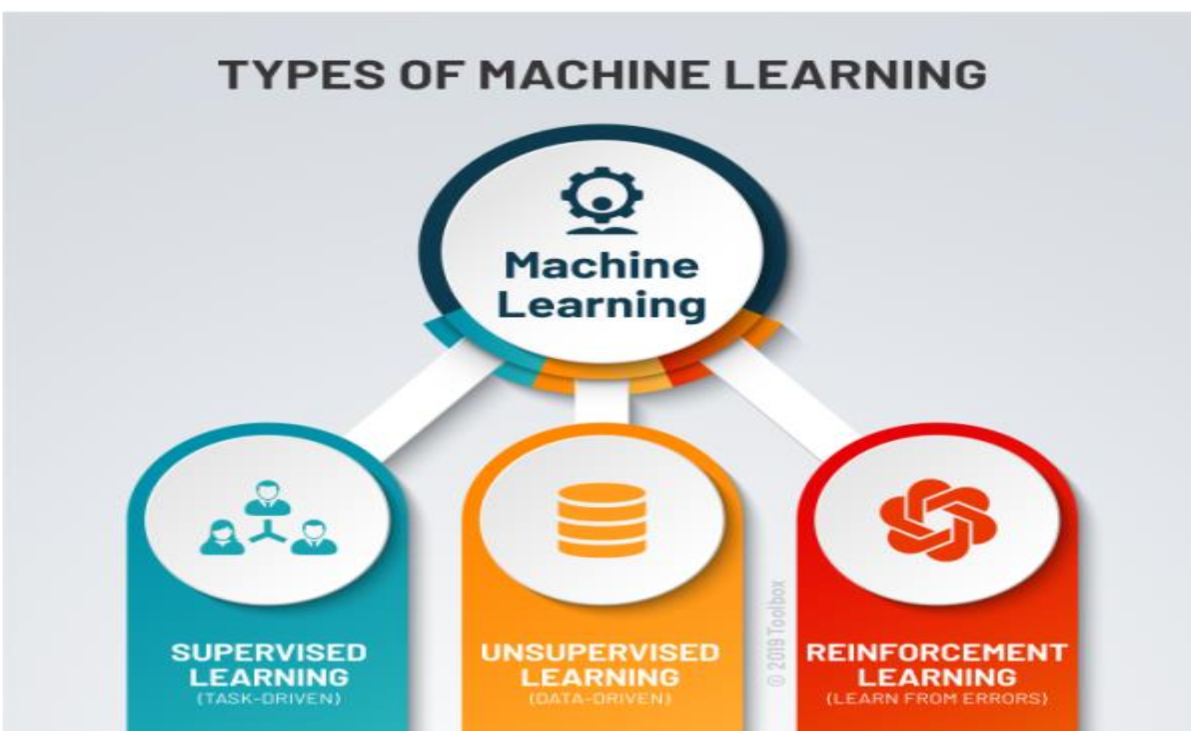
ABOUT PROJECT

The Technologies used in the Projects are Python, Machine Learning, Deep Learning, Computer Vision, Some Important Libraries Keras, and Opencv for Image Classification, Numpy, Pandas and Tensorflow.

Artificial Intelligence - The ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning, generalize, or learn from past experience.

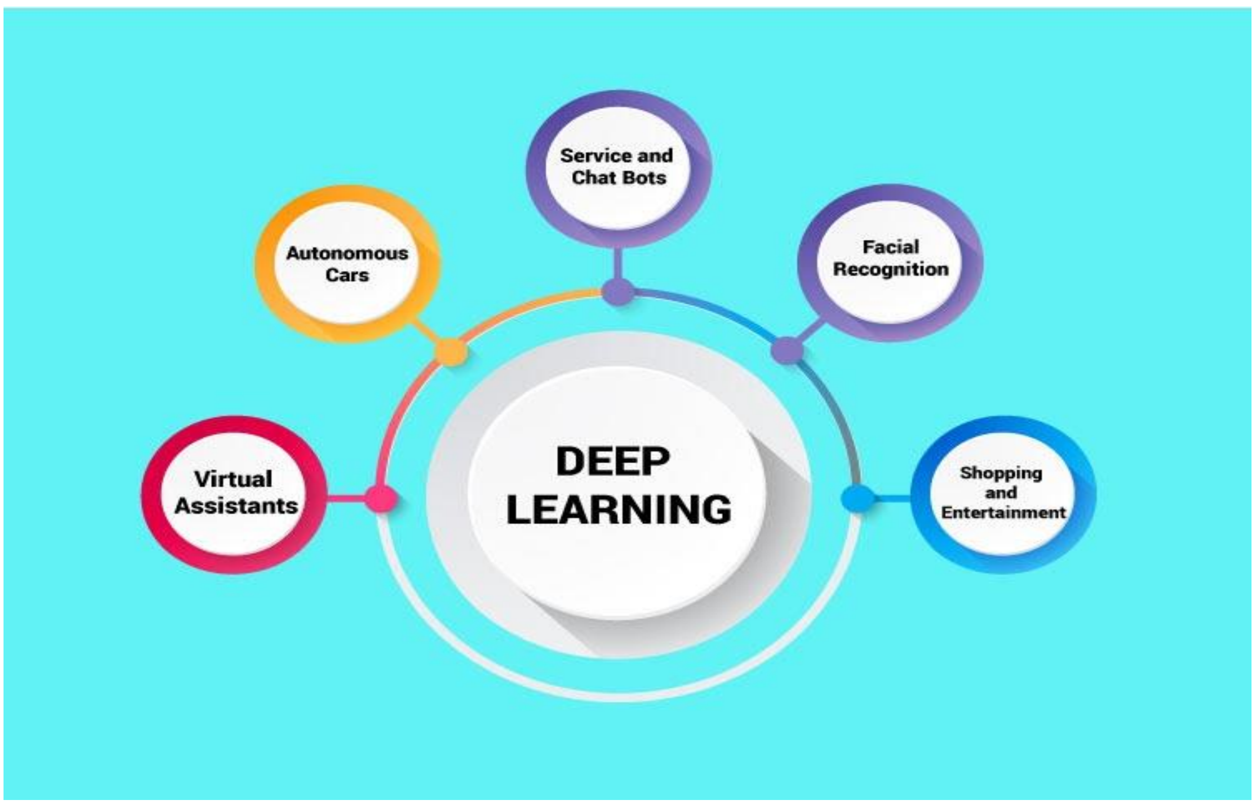


Machine Learning - Machine learning (ML) is the study of computer algorithms that can improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as training data, in order to make predictions or decisions without being explicitly programmed to do so.



Deep Learning - **Deep learning** (also known as **deep structured learning**) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Deep-learning architectures such as deep neural networks, deep belief networks, deep reinforcement learning, recurrent neural networks and convolution have been applied to fields including computer vision, speech recognition, natural language processing, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.



PROGRAMMING LANGUAGE PYTHON

Python is **an interpreted high-level general-purpose programming language**. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

APPLICATION OF PYTHON

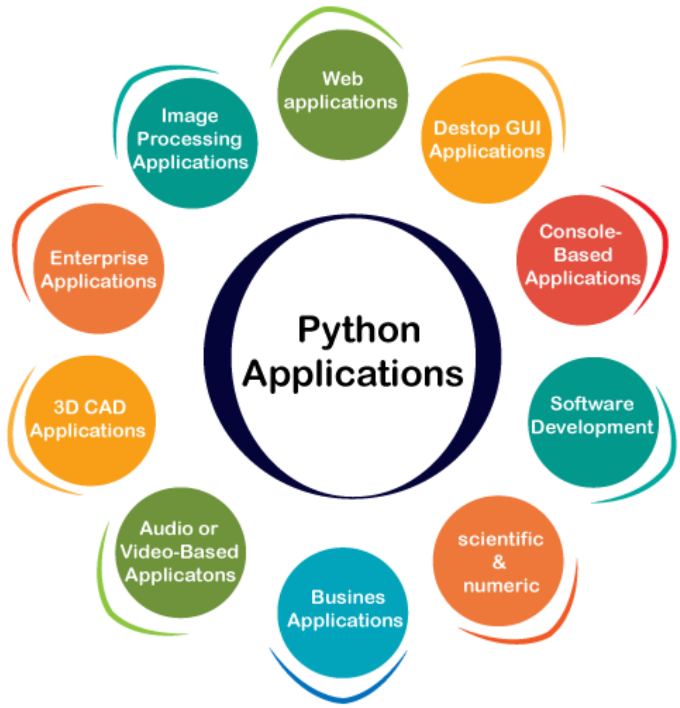
Applications of Python are:-

1. **Web Applications** - We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request etc. One of Python web-framework named Django is used on **Instagram.**

2. **Desktop GUI Applications** - The GUI stands for the Graphical User Interface, which provides a smooth interaction to any application. Python provides a Tk GUI library to develop a user interface.

3. **Software Development** - Python is useful for the software development process. It works as a support language and can be used to build control and management, testing, etc.

4. **Audio or Video-based Applications** - Python is flexible to perform multiple tasks and can be used to create multimedia applications. Some multimedia applications which are made by using Python are Tim Player, cplay, etc.



FUTURE SCOPE

- The future scope of python is bright as it also helps to build different trending technologies projects such as Machine Learning, Artificial Intelligence etc. The popular Python libraries for the data visualization are MATPLOTLIB and SEABORN.
- In the field of Artificial Intelligence, Python is used as an engineering tool. The scope of Artificial Intelligence with python is pretty wide and being open source people will contribute to it and keep it going.
- Python has numerous of frameworks, libraries like Sk-learn, Numpy, Pandas, Seaborn, Matplotlib, and many more which has made python so popular.

- The future scope of Python deals with analyzing a large number of data sets across computer clusters through its high-performance toolkits and libraries.

TOP COMPETITORS OF PYTHON

The future scope of python programming language also depends on its competitors in the IT market. But, due to the fact that it has become a core language for future technologies such as artificial intelligence, big data, etc., it is surely going to rise further and will be able to beat its competitors. The top competitors of Python are listed as Java, C, C++.

PROJECT REQUIREMENTS

The libraries that have been imported for this project is listed below:

1. **Numpy:** Numpy stands for numeric python which is a python package for the computation and processing of the multidimensional and single dimensional array elements. It can be used to perform a wide variety of mathematical operation on array, it has also function to work on linear algebra, Fourier transform and also on matrix etc.

2. **Pandas:** Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. It is fast, powerful, flexible and easy to use open source

data analysis and manipulation tool, build on the top of Python programming language.

The key features

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of datasets.
- Label-based slicing, indexing and subsetting of large datasets.

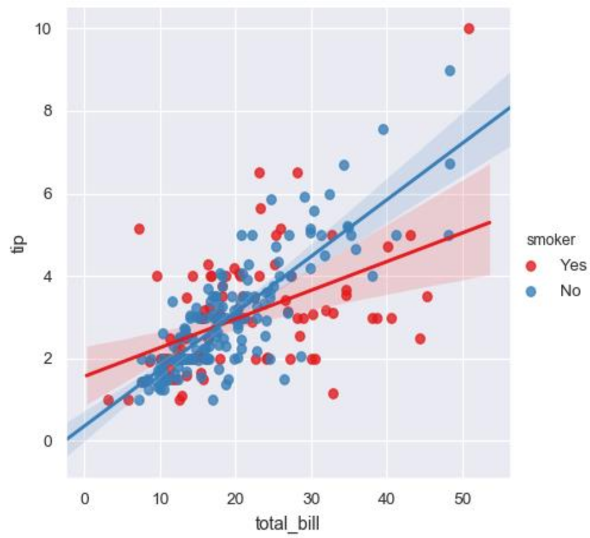
3. **Matplotlib:** It is a plotting library for python programming language. Matplotlib is one of the most popular Python packages used for data visualization. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It is a cross-platform library for making 2D plots from data in arrays.

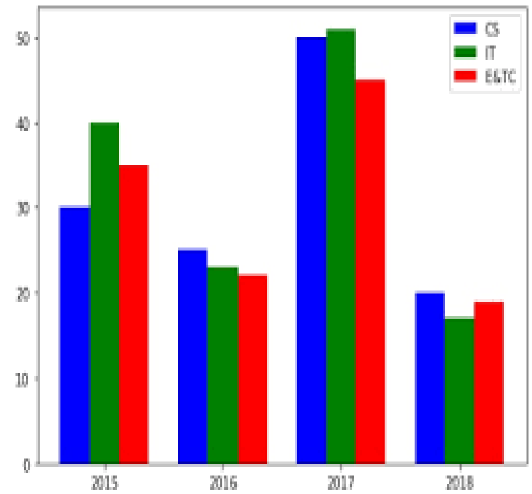
4. **Seaborn:** It is basically used to plot statistical graphics. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. Seaborn can be installed using pip on Windows. It is used to provide graphical representation of random distribution. Visualization is the central part in seaborn library for data exploration and data visualization.

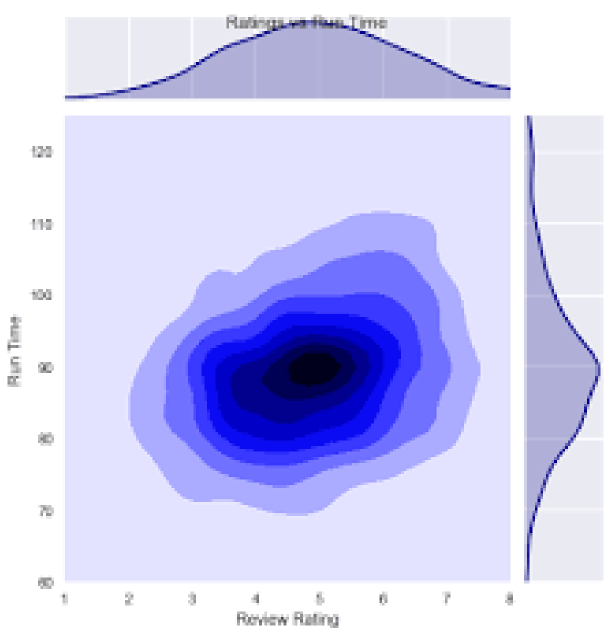
5. **Pygame:** This library is used mainly to create games but here we are using it to ring an alarm.

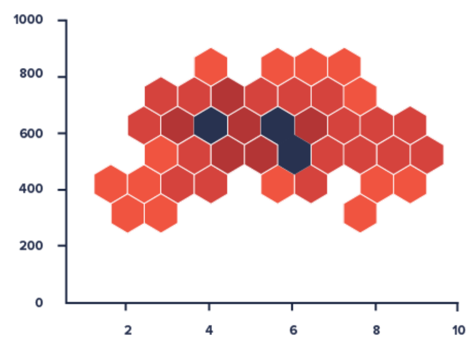
6. **Tensorflow:** TensorFlow is an open source library for fast numerical computing. It was created and is maintained by Google and released under the Apache 2.0 open source license. TensorFlow provides a collection of workflows to develop and train models using python or javascript and east to deploy in cloud or in any browser.

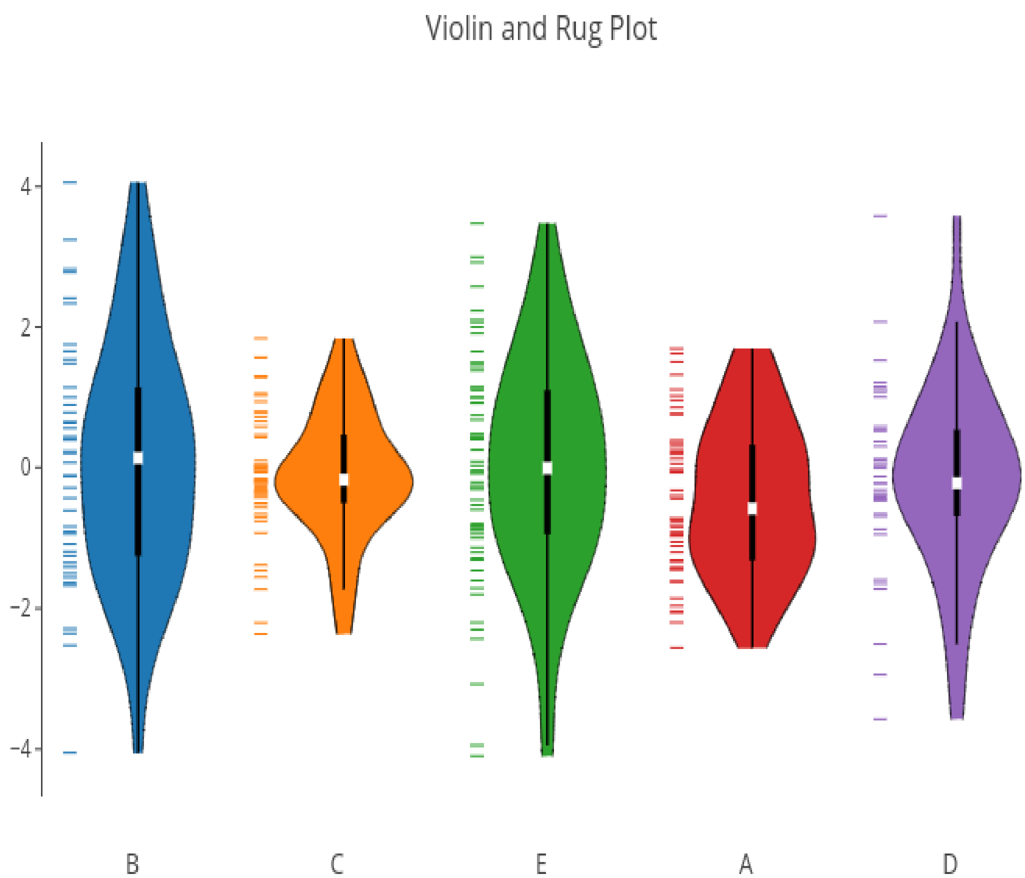
Some Plot in Seaborn Machine Learning Library











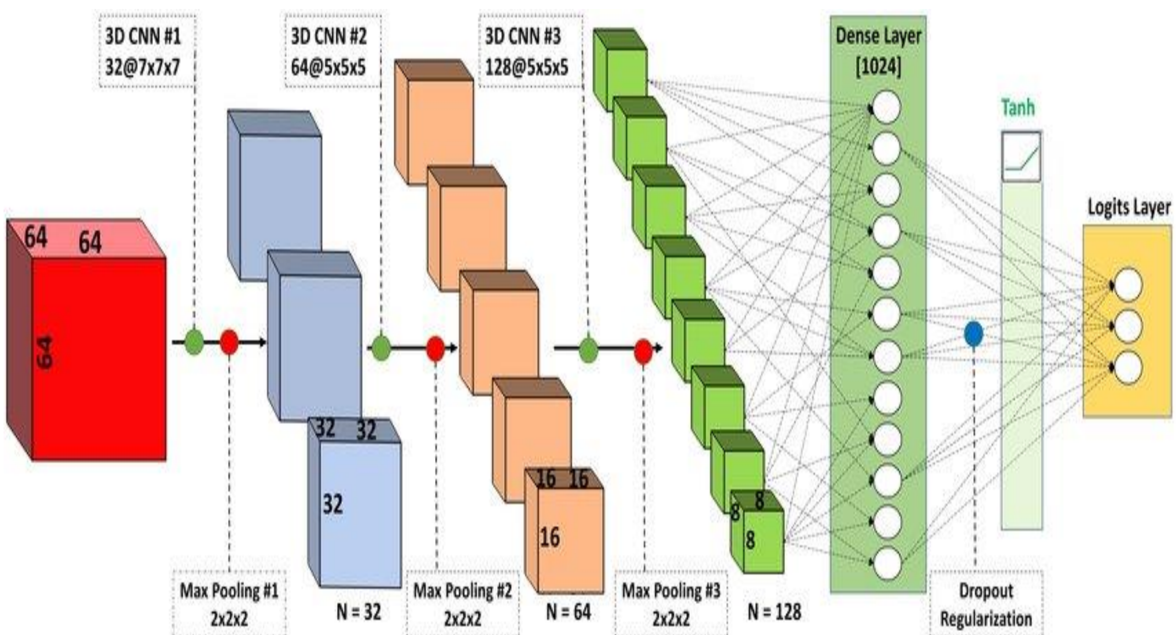
7. **Input Layer:** In Keras, the input layer itself is not a layer, but a tensor. It's theI starting tensor you send to the first hidden layer. This tensor must have the same shape as your training data. They consist of artificial neurons.

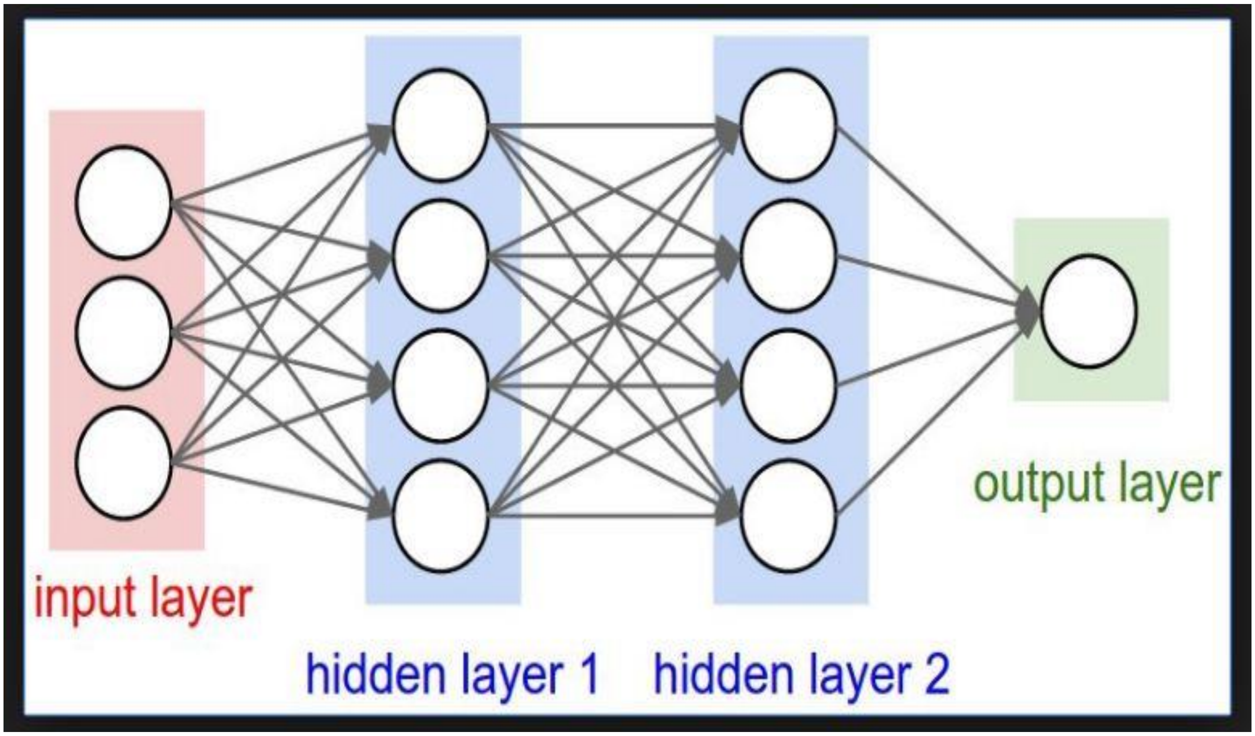
8. **Hidden Layer:** A hidden layer is an artificial neural network that is in between input layers and output layers, where the artificial neurons take in a set of weighted inputs and produce an output through activation.

9. **Output Layer:** The output layer in an artificial neural network is the last layer of neurons that produces given outputs for the program.

10. **Keras:** Keras is an open source deep learning framework for python. It has been developed by an artificial intelligence researcher at Google named Francois Chollet.

11. **Sklearn:** Sklearn is most common library that contains a lot of efficient tools for machine learning and statistical models including different regression, classification, clustering etc. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality 23 reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.





HARDWARE & SOFTWARE REQUIREMENT

SOFTWARE REQUIREMENTS

Operating system: Windows

Front End: Python 3.7

Platform: Anaconda Navigator

HARDWARE REQUIREMENTS

Machine: DELL LAPTOP i7 7TH GENERATION

Speed: 1.60 GHz & above

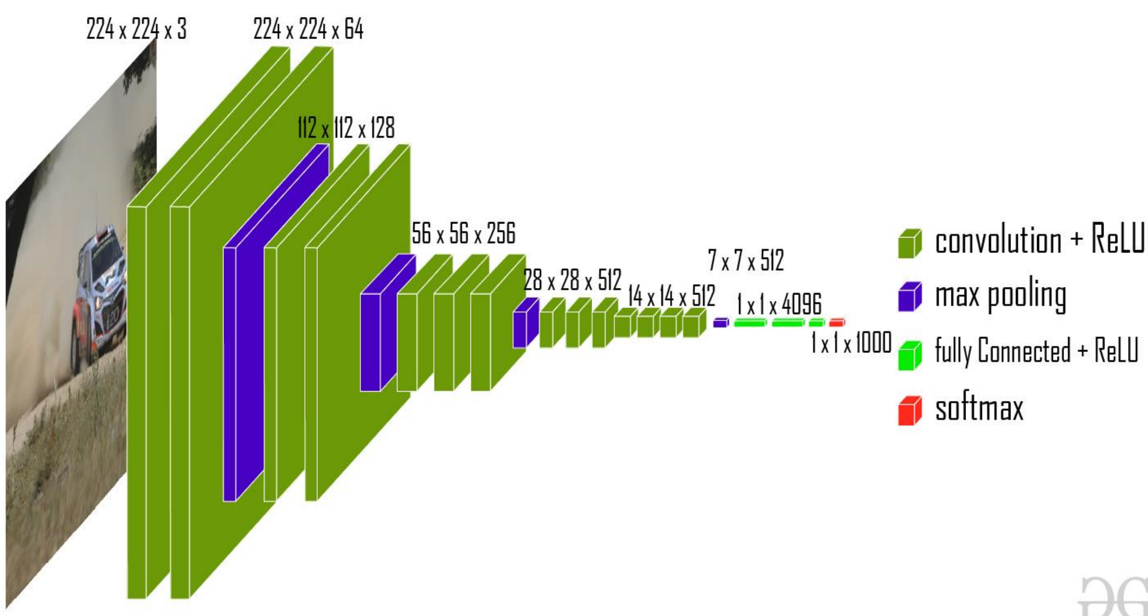
RAM: 8 GB

Hard disk: 1 TB WITH 128 GB SSD

DESCRIPTION OF THE TRAINING MODEL

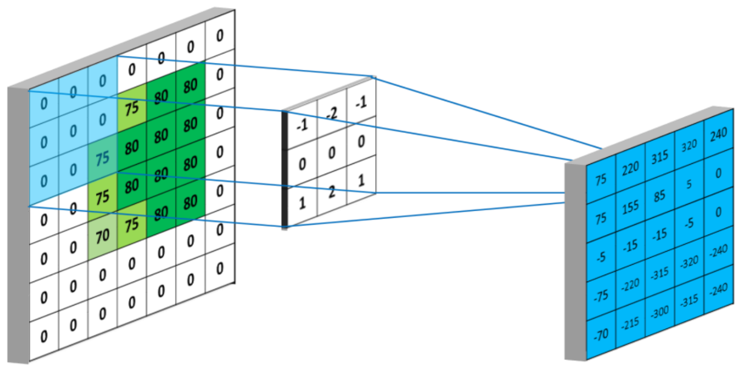
In neural networks, Convolutional neural network (ConvNets or CNNs) is one of the main categories to do images recognition, images classifications. Objects detections, recognition face etc., are some of the areas where CNNs are widely used.

CNN is a type of deep learning model for processing data that has a grid pattern, such as images, which is inspired by the organization of animal visual cortex [13, 14] and designed to automatically and adaptively learn spatial hierarchies of features, from low- to high-level patterns.



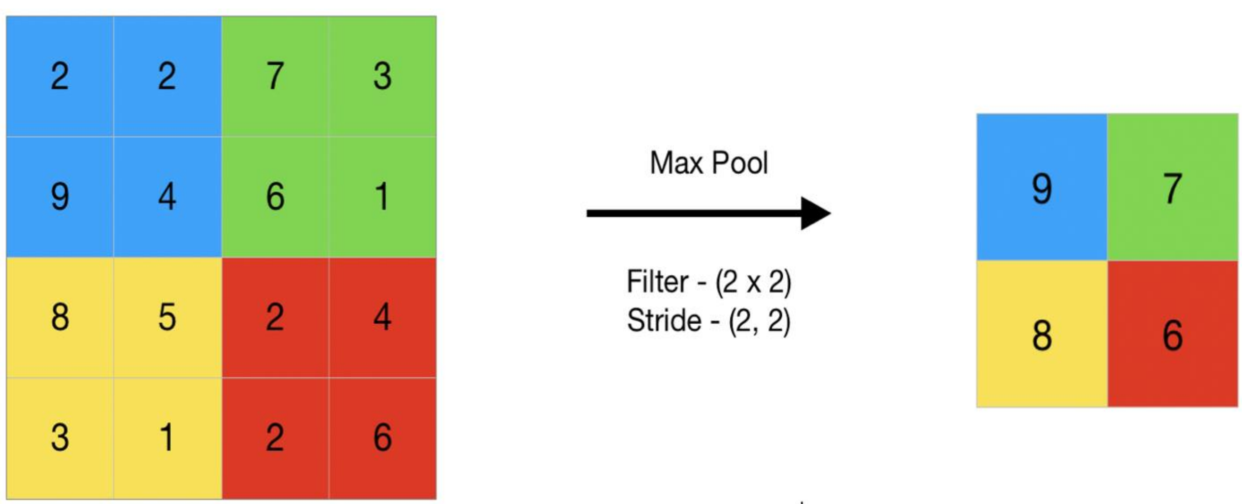
Kernel

In Convolutional neural network, the kernel is nothing but a filter that is used to extract the features from the images. The kernel is a matrix that moves over the input data, performs the dot product with the sub-region of input data, and gets the output as the matrix of dot products.



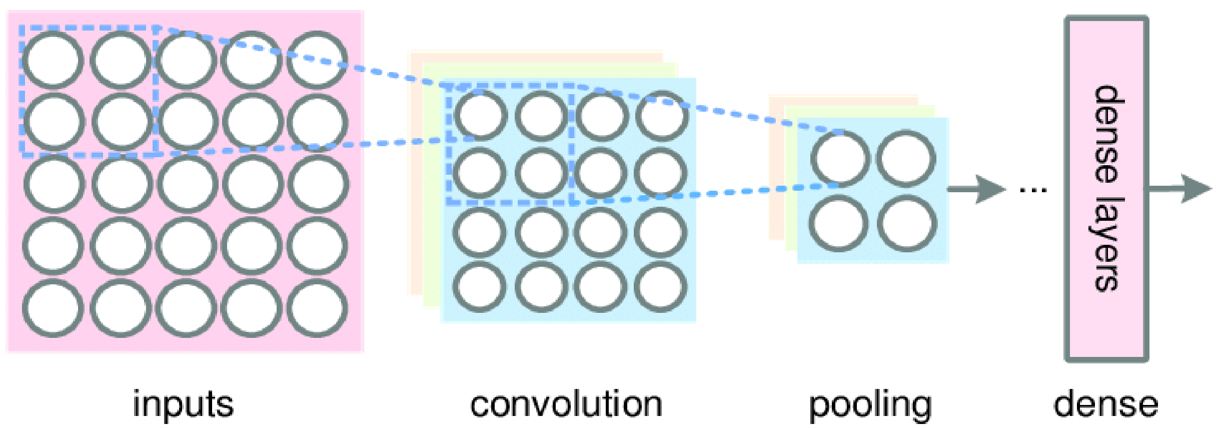
Pooling Layer

A pooling layer is another building block of a CNN. Pooling. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling layer operates on each feature map independently.



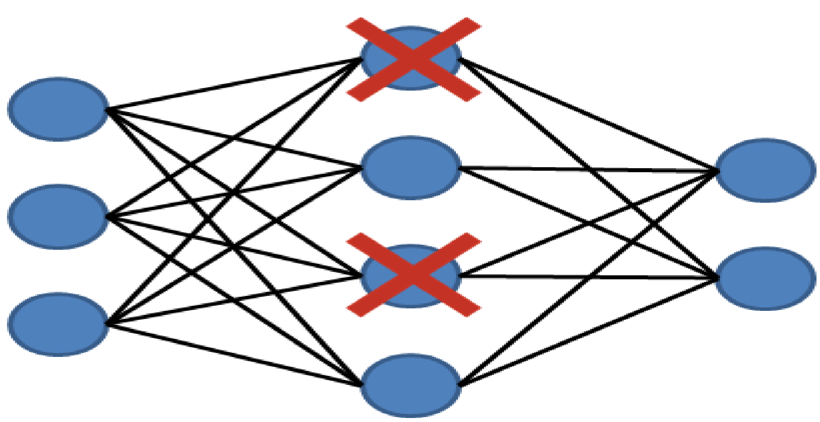
Dense Layer

Dense Layer is simple layer of neurons in which each neuron receives input from all the neurons of previous layer, thus called as dense. Dense Layer is used to classify image based on output from convolutional layers.



Dropout Layer

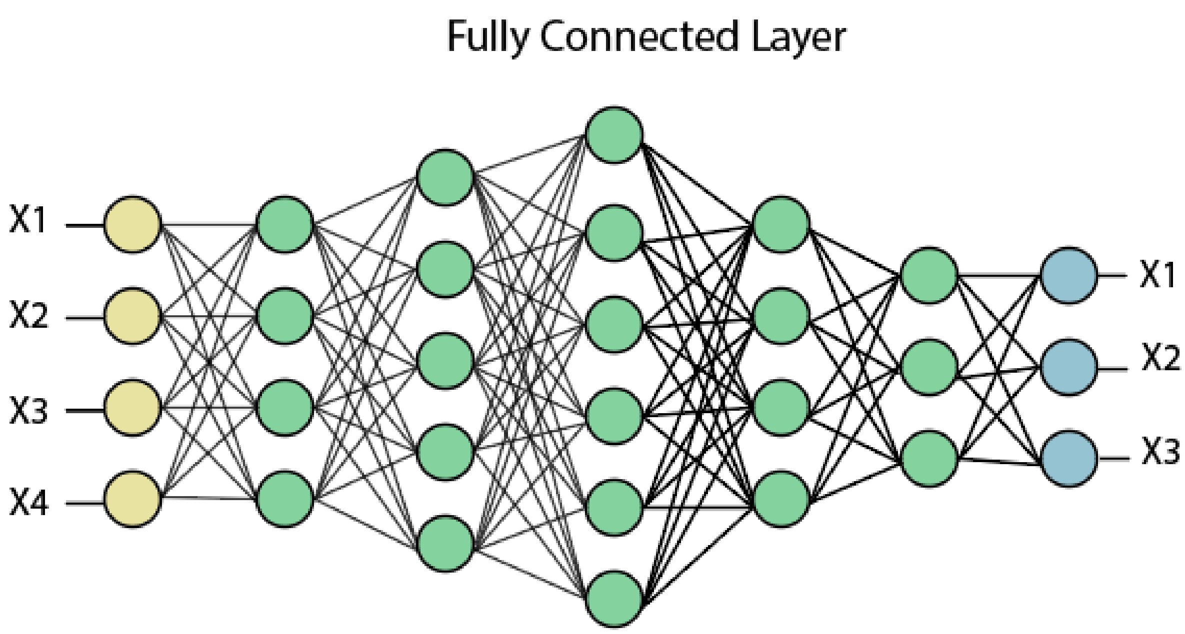
Dropout is a technique used to prevent a model from over fitting. Dropout works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase.



Fully Connected Layer

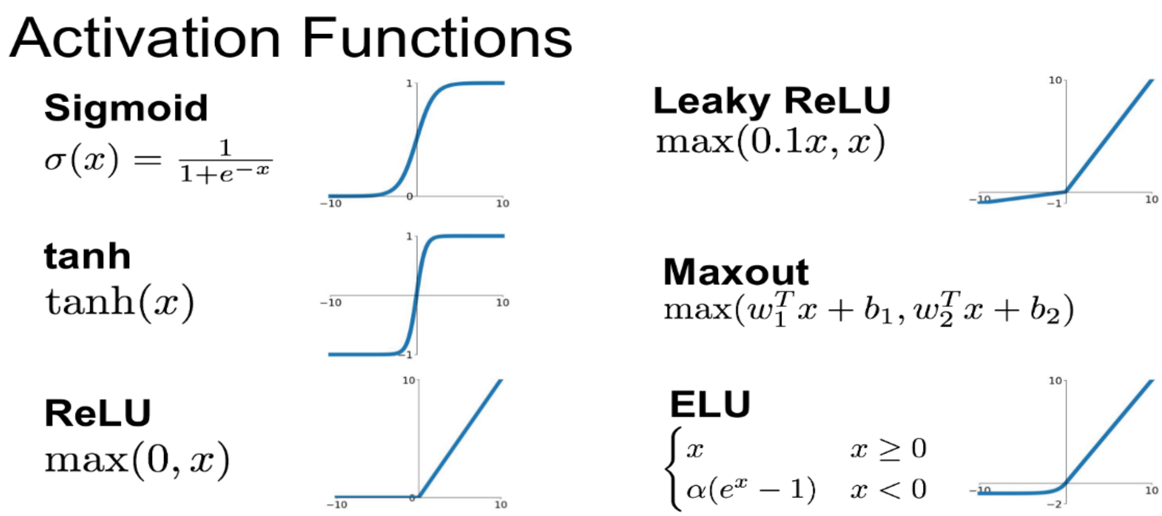
Fully Connected Layer is simply, feed forward neural networks. Fully Connected Layers form the last few layers in the network. The input to the fully connected layer is the output from the final Pooling or Convolutional Layer, which is flattened and then fed into the fully connected layer.

The **input** to the fully connected layer is the output from the *final* Pooling or Convolutional Layer, which is *flattened* and then fed into the fully connected layer.



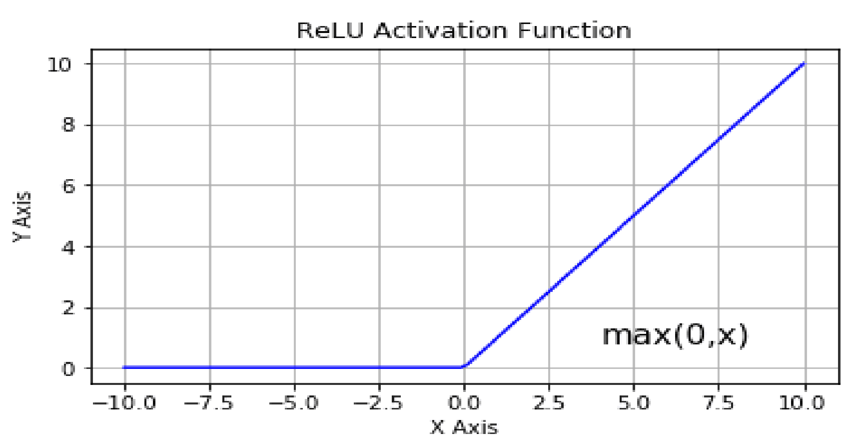
Activation Function

The activation function is **a node that is put at the end of or in between Neural Networks**. This makes it very computational efficient as few neurons are activated per time. It does not saturate at the positive region. In practice, ReLU converges six times faster than tanh and sigmoid activation functions.



ReLu Activation Function

ReLU stands for rectified linear unit, and is a type of activation function. Mathematically, it is defined as **y = max (0, x)**. ReLU is the most commonly used activation function in neural networks, especially in CNNs. If you are unsure what activation function to use in your network, ReLU is usually a good first choice.



Softmax Activation Function

The softmax function, also known as softargmax or normalized exponential function is a generalization of the logistic function to multiple dimensions.

Our output for the Softmax function is the ratio of the exponential of the parameter and the sum of exponential parameter. θ, on a high level is the sum of the score of each occurring element in the vector. In a generalized form we say that θ is the transpose of the weights matrix w, multiplied by the feature matrix x.

$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

SOURCE CODE SNIPPETS

Import Necessary Deep Learning and Machine Learning Library

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import tensorflow as tf
```

```
from PIL import Image

import cv2

from sklearn.model_selection import train_test_split

from keras.utils import to_categorical

from keras.models import Sequential, load_model

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

Importing Dataset ''Traffic Sign Recognition''

```
import os

os.chdir('D:\Traffic Signal')
```

Process images for different 43 classes of Traffic Signals

```
data =[]

labels = []

classes =43

cur_path = os.getcwd()

for i in range(classes):

    path = os.path.join(cur_path,'train',str(i))

    images = os.listdir(path)

    for a in images:

        try:

            image = Image.open(path +'\\'+ a)
```

```
        image = image.resize((30,30))

        # Resizing all images into 30*30

        image =np.array(image)

        data.append(image)

        labels.append(i)

    except Exception as e:

        print(e)
```

Convert image data list into Numpy Array

```
data = np.array(data)

labels = np.array(labels)

print(data.shape, labels.shape)
```

Save the Model

```
np.save('./training/data',data)

np.save('./training/target',labels)
```

Load the Model

```
data=np.load('./training/data.npy')

labels=np.load('./training/target.npy')
```

Split the data into training and testing

```
X_train, X_test, y_train, y_test =train_test_split(data, labels,
test_size=0.2, random_state=0)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

Convert labels into one hot encoding
y_train = to_categorical(y_train,43)

y_test = to_categorical(y_test,43)

Build CNN Model

```
model =Sequential()

model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu',
input_shape=X_train.shape[1:]))

model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Dropout(rate=0.25))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu'))

model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu'))

model.add(MaxPool2D(pool_size=(2,2)))

model.add(Dropout(rate=0.25))

model.add(Flatten())

model.add(Dense(256, activation='relu'))

model.add(Dropout(rate=0.5))

# We have 43 classes that's why we have defined 43in the dense
model.add(Dense(43, activation='softmax'))

model.add(Dense(43, activation='softmax'))
```

Training Model for 15 Epochs

```
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

epochs = 15

history = model.fit(X_train, y_train, batch_size=32, epochs=epochs,
validation_data=(X_test, y_test))
```

Save the Model

```
model.save("./training/TSR.h5")
```

 # Classes of trafic signs

```
classes = { 0:'Speed limit (20km/h)',

        1:'Speed limit (30km/h)',

        2:'Speed limit (50km/h)',

        3:'Speed limit (60km/h)',

        4:'Speed limit (70km/h)',

        5:'Speed limit (80km/h)',

        6:'End of speed limit (80km/h)',

        7:'Speed limit (100km/h)',

        8:'Speed limit (120km/h)',

        9:'No passing',

        10:'No passing veh over 3.5 tons',
```

11:'Right-of-way at intersection',

12:'Priority road',

13:'Yield',

14:'Stop',

15:'No vehicles',

16:'Vehicle > 3.5 tons prohibited',

17:'No entry',

18:'General caution',

19:'Dangerous curve left',

20:'Dangerous curve right',

21:'Double curve',

22:'Bumpy road',

23:'Slippery road',

24:'Road narrows on the right',

25:'Road work',

26:'Traffic signals',

27:'Pedestrians',

28:'Children crossing',

29:'Bicycles crossing',

30:'Beware of ice/snow',

```
31:'Wild animals crossing',
32:'End speed + passing limits',
33:'Turn right ahead',
34:'Turn left ahead',
35:'Ahead only',
36:'Go straight or right',
37:'Go straight or left',
38:'Keep right',
39:'Keep left',
40:'Roundabout mandatory',
41:'End of no passing',
42:'End no passing vehicle > 3.5 tons' }
```

Importing and Loading the Saved Model For Prediction

```
import os
os.chdir(r'D:\Traffic Signal')
from keras.models import load_model
model = load_model('./training/TSR.h5')
```

Predicting on the Test Data

```
from PIL import Image
```

```
import numpy as np

import matplotlib.pyplot as plt

def test_on_img(img):

    data=[]

    image = Image.open(img)

    image = image.resize((30,30))

    data.append(np.array(image))

    X_test=np.array(data)

    Y_pred = model.predict_classes(X_test)

    return image,Y_pred

plot,prediction = test_on_img(r'D:\Traffic Signal\Test\00124.png')

s = [str(i) for i in prediction]

a = int("".join(s))

print("Predicted traffic sign is: ", classes[a])

plt.imshow(plot)

plt.show()

plot,prediction = test_on_img(r'D:\Traffic Signal\Test\00209.png')

s = [str(i) for i in prediction]

a = int("".join(s))

print("Predicted traffic sign is: ", classes[a])
```

```
plt.imshow(plot)

plt.show()

plot,prediction = test_on_img(r'D:\Traffic Signal\Test\12041.png')

s = [str(i) for i in prediction]

a = int("".join(s))

print("Predicted traffic sign is: ", classes[a])

plt.imshow(plot)

plt.show()

plot,prediction = test_on_img(r'D:\Traffic Signal\Test\11765.png')

s = [str(i) for i in prediction]

a = int("".join(s))

print("Predicted traffic sign is: ", classes[a])

plt.imshow(plot)

plt.show()

plot,prediction = test_on_img(r'D:\Traffic Signal\Test\12059.png')

s = [str(i) for i in prediction]

a = int("".join(s))

print("Predicted traffic sign is: ", classes[a])

plt.imshow(plot)

plt.show()
```

```
plot,prediction = test_on_img(r'D:\Traffic Signal\Test\12118.png')

s = [str(i) for i in prediction]

a = int("".join(s))

print("Predicted traffic sign is: ", classes[a])

plt.imshow(plot)

plt.show()

plot,prediction = test_on_img(r'D:\Traffic Signal\Test\11454.png')

s = [str(i) for i in prediction]

a = int("".join(s))

print("Predicted traffic sign is: ", classes[a])

plt.imshow(plot)

plt.show()

plot,prediction = test_on_img(r'D:\Traffic Signal\Test\11311.png')

s = [str(i) for i in prediction]

a = int("".join(s))

print("Predicted traffic sign is: ", classes[a])

plt.imshow(plot)

plt.show()

plot,prediction = test_on_img(r'D:\Traffic Signal\Test\12258.png')

s = [str(i) for i in prediction]
```

```
a = int("".join(s))

print("Predicted traffic sign is: ", classes[a])

plt.imshow(plot)

plt.show()

plot,prediction = test_on_img(r'D:\Traffic Signal\Test\11866.png')

s = [str(i) for i in prediction]

a = int("".join(s))

print("Predicted traffic sign is: ", classes[a])

plt.imshow(plot)

plt.show()
```

Plotting Graphs for Accuracy and Loss for the Model

```
plt.figure(0)

plt.plot(history.history['accuracy'],label='training accuracy')

plt.plot(history.history['val_accuracy'],label='val accuracy')

plt.title('Accuracy')

plt.xlabel('Epochs')

plt.ylabel('accuracy')

plt.legend()

plt.show()

plt.figure(0)
```

```
plt.plot(history.history['loss'],label='training loss')

plt.plot(history.history['val_loss'],label='val loss')

plt.title('Loss')

plt.xlabel('Epochs')

plt.ylabel('loss')

plt.legend()

plt.show()
```

Calculating Accuracy and Error for the Model

```
def testing(testcsv):

    y_test = pd.read_csv(testcsv)

    label = y_test['ClassId'].values

    imgs = y_test['Path'].values

    data = []

    for img  in imgs:

        image = Image.open(img)

        image = image.resize((30,30))

        data.append(np.array(image))

    X_test = np.array(data)

    return X_test,label

X_test, label = testing('Test.csv')
```

```
Y_pred = model.predict_classes(X_test)

print(Y_pred)

from sklearn.metrics import accuracy_score

from sklearn import metrics

print("Accuracy Score is: ",accuracy_score(label,Y_pred))

print("Error is : ",np.sqrt(metrics.mean_squared_error(label,Y_pred)))
```

SNAPSHOT

Traffic Signal Classification Using CNN

Import Necessary Deep Learning and Machine Learning Library

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from PIL import Image
import cv2
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential, load_model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
```

```
C:\Users\Debanjan Saha\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing (type, 1)
or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)typ
e'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\Debanjan Saha\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing (type, 1)
or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)typ
e'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\Debanjan Saha\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing (type, 1)
or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)typ
e'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\Debanjan Saha\anaconda3\lib\site-packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing (type, 1)
or '1type' as a synonym of type is deprecated; in a future version of numpy, it will be understood as (type, (1,)) / '(1,)typ
e'.
```

Importing Dataset "Traffic Signal Recognition"

```
import os
os.chdir('D:\Traffic Signal')
```

Process images for different 43 classes of Traffic Signals

```
data =[]
labels = []
classes =43
cur_path = os.getcwd()
for i in range(classes):
    path = os.path.join(cur_path,'train',str(i))
    images = os.listdir(path)
    for a in images:
        try:
            image = Image.open(path +'\\'+ a)
            image = image.resize((30,30))
            # Resizing all images into 30*30
            image =np.array(image)
            data.append(image)
            labels.append(i)
        except Exception as e:
            print(e)
```

Convert image data list into Numpy Array

```
data = np.array(data)
labels = np.array(labels)
print(data.shape, labels.shape)
```

```
(39209, 30, 30, 3) (39209,)
```

Save the Model

```
np.save('./training/data',data)
np.save('./training/target',labels)
```

Load the Model

```
data=np.load('./training/data.npy')
labels=np.load('./training/target.npy')
```

Split the data into training and testing

```
X_train, X_test, y_train, y_test =train_test_split(data, labels, test_size=0.2, random_state=0)
```

```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(31367, 30, 30, 3) (7842, 30, 30, 3) (31367,) (7842,)
```

Convert labels into one hot encoding

```
y_train = to_categorical(y_train,43)
y_test = to_categorical(y_test,43)
```

Build CNN Model

```
model =Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu'))
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(rate=0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
# We have 43 classes that's why we have defined 43in the dense model.add(Dense(43, activation='softmax'))
model.add(Dense(43, activation='softmax'))
```

```
WARNING:tensorflow:From C:\Users\Debanjan Saha\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:4070: The name t
f.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

Training Model for 15 Epochs

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
epochs = 15
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data=(X_test, y_test))
```

```
WARNING:tensorflow:From C:\Users\Debanjan Saha\anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:422: The name t
f.global_variables is deprecated. Please use tf.compat.v1.global_variables instead.

Train on 31367 samples, validate on 7842 samples
Epoch 1/15
31367/31367 [==============================] - 70s 2ms/step - loss: 2.1116 - accuracy: 0.4612 - val_loss: 0.7737 - val_accurac
y: 0.8057
Epoch 2/15
31367/31367 [==============================] - 70s 2ms/step - loss: 0.9358 - accuracy: 0.7231 - val_loss: 0.3665 - val_accurac
y: 0.9096
Epoch 3/15
31367/31367 [==============================] - 70s 2ms/step - loss: 0.6163 - accuracy: 0.8134 - val_loss: 0.1840 - val_accurac
y: 0.9531
Epoch 4/15
31367/31367 [==============================] - 71s 2ms/step - loss: 0.4643 - accuracy: 0.8599 - val_loss: 0.2068 - val_accurac
y: 0.9438
Epoch 5/15
31367/31367 [==============================] - 73s 2ms/step - loss: 0.3601 - accuracy: 0.8939 - val_loss: 0.1034 - val_accurac
y: 0.9694
Epoch 6/15
31367/31367 [==============================] - 72s 2ms/step - loss: 0.3170 - accuracy: 0.9084 - val_loss: 0.1637 - val_accurac
y: 0.9481
Epoch 7/15
31367/31367 [==============================] - 71s 2ms/step - loss: 0.2842 - accuracy: 0.9167 - val_loss: 0.0741 - val_accurac
y: 0.9796
Epoch 8/15
31367/31367 [==============================] - 70s 2ms/step - loss: 0.2690 - accuracy: 0.9239 - val_loss: 0.0699 - val_accurac
y: 0.9813
```

Save the Model

```
model.save("./training/TSR.h5")
```

```python
# Classes of trafic signs
classes = { 0:'Speed limit (20km/h)',
            1:'Speed limit (30km/h)',
            2:'Speed limit (50km/h)',
            3:'Speed limit (60km/h)',
            4:'Speed limit (70km/h)',
            5:'Speed limit (80km/h)',
            6:'End of speed limit (80km/h)',
            7:'Speed limit (100km/h)',
            8:'Speed limit (120km/h)',
            9:'No passing',
            10:'No passing veh over 3.5 tons',
            11:'Right-of-way at intersection',
            12:'Priority road',
            13:'Yield',
            14:'Stop',
            15:'No vehicles',
            16:'Vehicle > 3.5 tons prohibited',
            17:'No entry',
            18:'General caution',
            19:'Dangerous curve left',
            20:'Dangerous curve right',
            21:'Double curve',
            22:'Bumpy road',
            23:'Slippery road',
            24:'Road narrows on the right',
            25:'Road work',
            26:'Traffic signals',
            27:'Pedestrians',
            28:'Children crossing',
            29:'Bicycles crossing',
```

Importing and Loading the Saved Model For Prediction

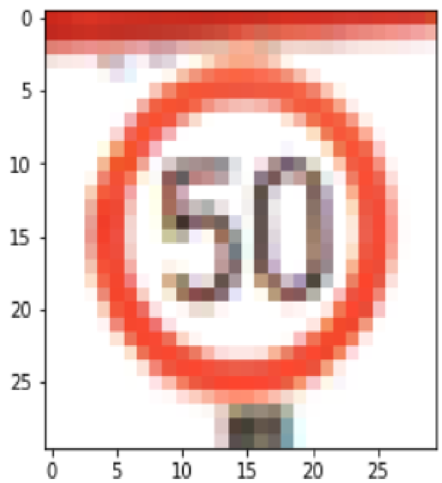
```python
import os
os.chdir(r'D:\Traffic Signal')
from keras.models import load_model
model = load_model('./training/TSR.h5')
```

Predicting on the Test Data

```python
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
def test_on_img(img):
    data=[]
    image = Image.open(img)
    image = image.resize((30,30))
    data.append(np.array(image))
    X_test=np.array(data)
    Y_pred = model.predict_classes(X_test)
    return image,Y_pred
```

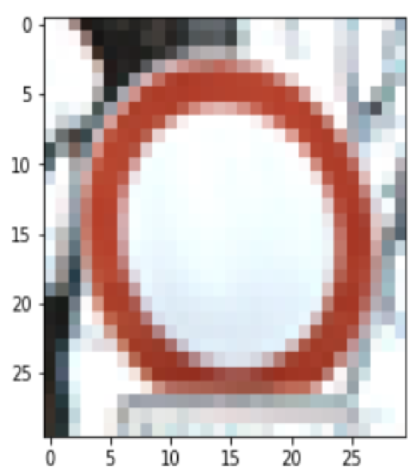
```python
plot,prediction = test_on_img(r'D:\Traffic Signal\Test\00124.png')
s = [str(i) for i in prediction]
a = int("".join(s))
print("Predicted traffic sign is: ", classes[a])
plt.imshow(plot)
plt.show()
```

```
Predicted traffic sign is:  Speed limit (50km/h)
```



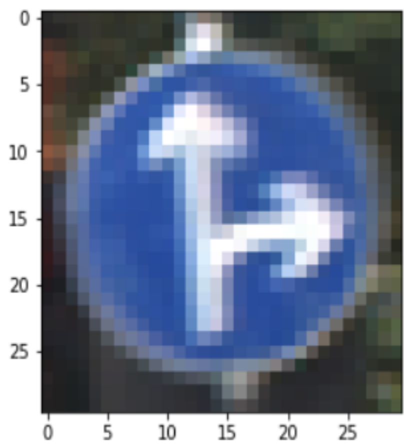

```python
plot,prediction = test_on_img(r'D:\Traffic Signal\Test\00209.png')
s = [str(i) for i in prediction]
a = int("".join(s))
print("Predicted traffic sign is: ", classes[a])
plt.imshow(plot)
plt.show()
```

```
Predicted traffic sign is:  No vehicles
```



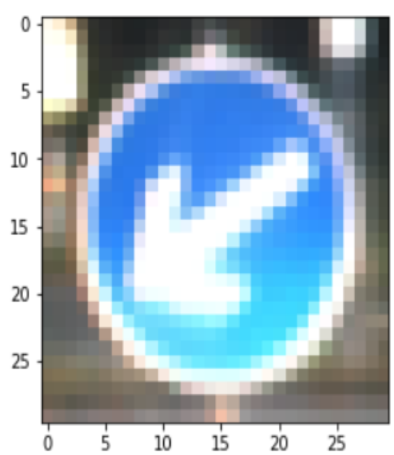
```
plot,prediction = test_on_img(r'D:\Traffic Signal\Test\12041.png')
s = [str(i) for i in prediction]
a = int("".join(s))
print("Predicted traffic sign is: ", classes[a])
plt.imshow(plot)
plt.show()
```

```
Predicted traffic sign is:  Go straight or right
```



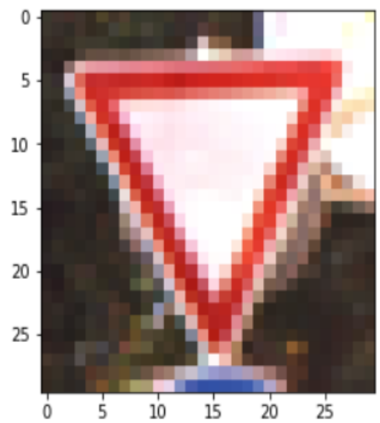
```
plot,prediction = test_on_img(r'D:\Traffic Signal\Test\11765.png')
s = [str(i) for i in prediction]
a = int("".join(s))
print("Predicted traffic sign is: ", classes[a])
plt.imshow(plot)
plt.show()
```

```
Predicted traffic sign is:  Keep left
```



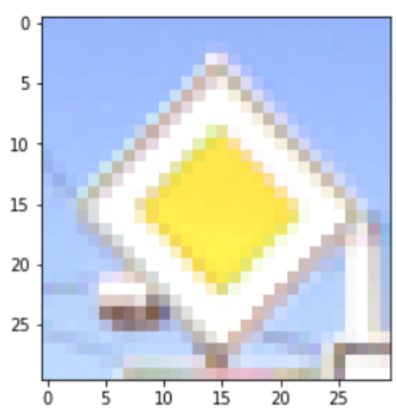
```python
plot,prediction = test_on_img(r'D:\Traffic Signal\Test\12059.png')
s = [str(i) for i in prediction]
a = int("".join(s))
print("Predicted traffic sign is: ", classes[a])
plt.imshow(plot)
plt.show()
```

```
Predicted traffic sign is:  Yield
```



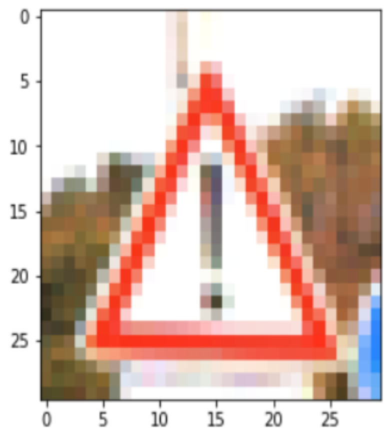

```python
plot,prediction = test_on_img(r'D:\Traffic Signal\Test\12118.png')
s = [str(i) for i in prediction]
a = int("".join(s))
print("Predicted traffic sign is: ", classes[a])
plt.imshow(plot)
plt.show()
```

```
Predicted traffic sign is:  Priority road
```



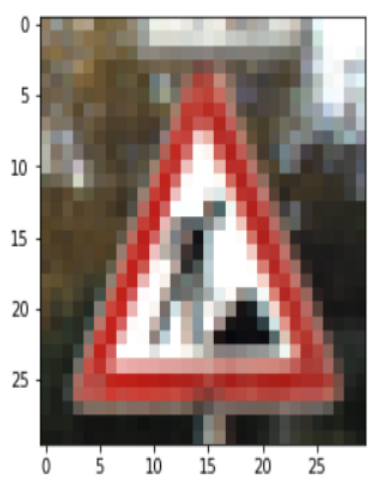
```
plot,prediction = test_on_img(r'D:\Traffic Signal\Test\11454.png')
s = [str(i) for i in prediction]
a = int("".join(s))
print("Predicted traffic sign is: ", classes[a])
plt.imshow(plot)
plt.show()
```

```
Predicted traffic sign is:  General caution
```



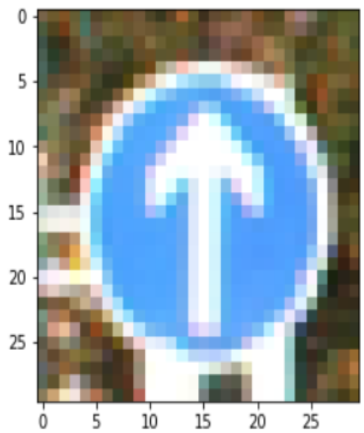

```
plot,prediction = test_on_img(r'D:\Traffic Signal\Test\11311.png')
s = [str(i) for i in prediction]
a = int("".join(s))
print("Predicted traffic sign is: ", classes[a])
plt.imshow(plot)
plt.show()
```

```
Predicted traffic sign is:  Road work
```



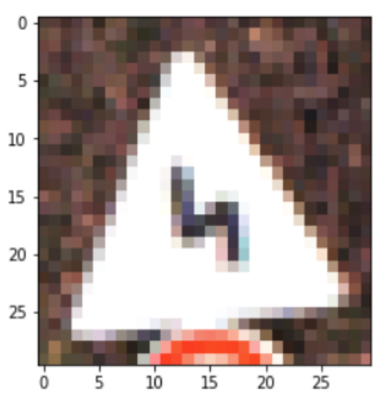
```
plot,prediction = test_on_img(r'D:\Traffic Signal\Test\12258.png')
s = [str(i) for i in prediction]
a = int("".join(s))
print("Predicted traffic sign is: ", classes[a])
plt.imshow(plot)
plt.show()
```

Predicted traffic sign is: Ahead only



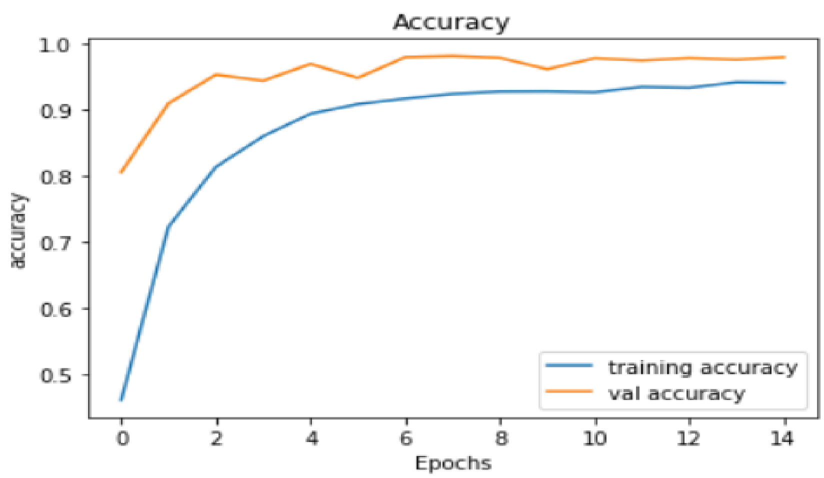
```
plot,prediction = test_on_img(r'D:\Traffic Signal\Test\11866.png')
s = [str(i) for i in prediction]
a = int("".join(s))
print("Predicted traffic sign is: ", classes[a])
plt.imshow(plot)
plt.show()
```

Predicted traffic sign is: Double curve

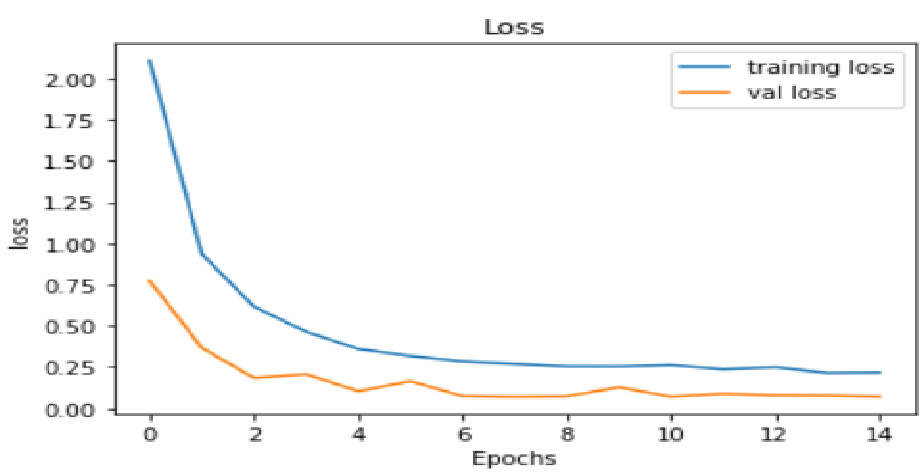


Plotting Graphs for Accuracy and Loss for the Model

```
plt.figure(0)
plt.plot(history.history['accuracy'],label='training accuracy')
plt.plot(history.history['val_accuracy'],label='val accuracy')
plt.title('Accuracy')
plt.xlabel('Epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()
```




```
plt.figure(0)
plt.plot(history.history['loss'],label='training loss')
plt.plot(history.history['val_loss'],label='val loss')
plt.title('Loss')
plt.xlabel('Epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



Calculating Accuracy and Error for the Model

```
def testing(testcsv):
    y_test = pd.read_csv(testcsv)
    label = y_test['ClassId'].values
    imgs = y_test['Path'].values
    data = []
    for img  in imgs:
        image = Image.open(img)
        image = image.resize((30,30))
        data.append(np.array(image))
    X_test = np.array(data)
    return X_test,label
```

```
X_test, label = testing('Test.csv')
```

```
Y_pred = model.predict_classes(X_test)
```

```
print(Y_pred)
```

```
[16  1 38 ...  3  7 10]
```

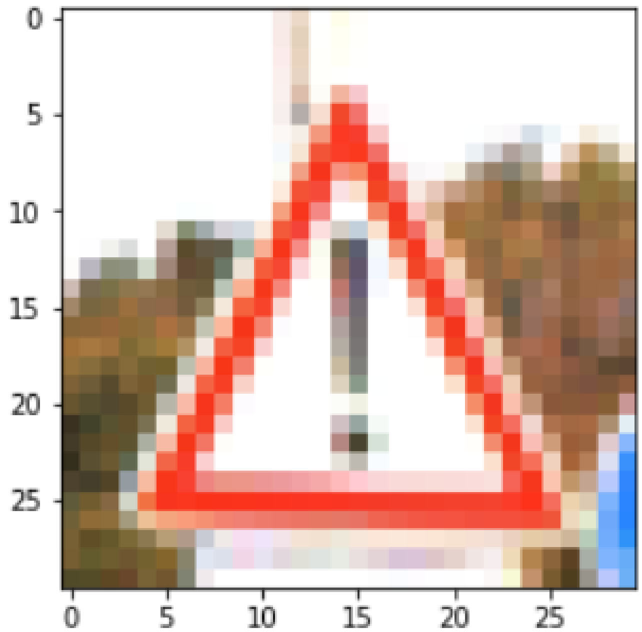
```
from sklearn.metrics import accuracy_score
from sklearn import metrics
print("Accuracy Score is: ",accuracy_score(label,Y_pred))
print("Error is : ",np.sqrt(metrics.mean_squared_error(label,Y_pred)))
```

```
Accuracy Score is:  0.9414093428345209
Error is :  3.4657354556593627
```

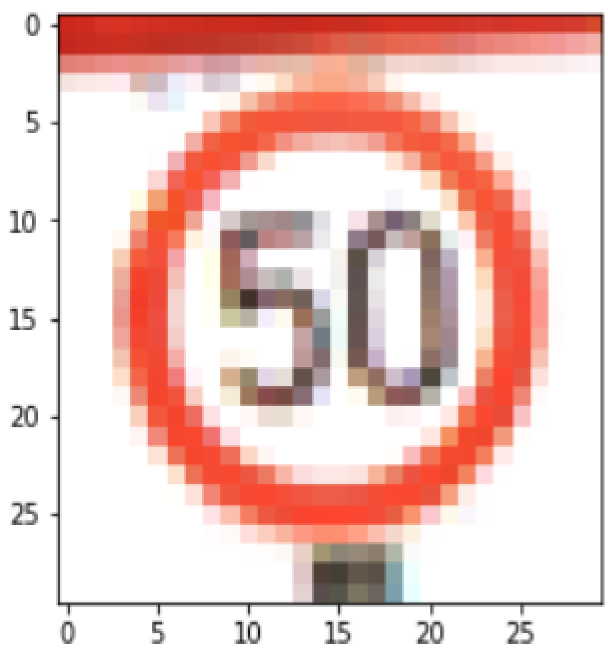
RESULT

Tried my Level Best with an Accuracy of **94.14%** while predicting Different Traffic Sign of Different Classes of the Test Data for the Model of **''Traffic Sign Classification'' using the Dataset "German Traffic Sign Recognition Benchmark"**.

Predicted traffic sign is: General caution



Predicted traffic sign is: Speed limit (50km/h)



CONCLUSION

```
from sklearn.metrics import accuracy_score
from sklearn import metrics
print("Accuracy Score is: ",accuracy_score(label,Y_pred))
print("Error is : ",np.sqrt(metrics.mean_squared_error(label,Y_pred)))
```

```
Accuracy Score is:  0.9414093428345209
Error is :  3.4657354556593627
```

Using CNN with **GTSRB Dataset** we got **94.14%** accuracy. For more accuracy we must increase the number of Epochs to get a good Training at the Training Phase for Prediction also we can Change the Activation Function to improve the Model Prediction.

REFERENCES

The contents have been gathered from the following:

- Information: GOOGLE
- Images: GOOGLE IMAGES
- Snapshots: Self-performed