

SRI SIDDHARTHA ACADEMY OF HIGHER EDUCATION

(Declared as Deemed to be University Under Section 3 of the UGC Act, 1956 Approved by AICTE,

Accredited by NBA, NAAC 'A' Grade)

AGALKOTE, TUMAKURU – 572107 KARNATAKA



Mini Project Report On

Road Sign Recognition System

Submitted by

Aditya P Shetty (19CS002)

Ansul(19CS013)

Under the guidance of

Dr.M.SIDDAPPA B.E,M.Tech,Ph.D

Professor and Head , Dept of CSE

SSIT, TUMAKURU-572105

BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE AND ENGINEERING



SRI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(A Constituent College of SRI SIDDHARTHA ACADEMY OF HIGHER EDUCATION)

MARALUR, TUMAKURU -572105,

2022-23

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING
SSIT, TUMKUR**

**SRI SIDDARTHA INSTITUTE OF TECHNOLOGY
MARALUR, TUMKUR -572105**
(A Constituent College of Sri Siddhartha Academy of Higher Education)



CERTIFICATE

This is to certify that the project work entitled “**Road Sign Recognition System**” is carried out by **Aditya P Shetty (19CS002)** and

Ansul(19CS013) in partial fulfillment of VI Semester “MINI PROJECT

(18CS6MP01)” Project work during the academic year 2022-23. It is certified that all corrections/suggestions indicating internal assessment have been incorporated in the report deposited in the department library. The project has been approved as it satisfies the academic requirements in respect of project work prescribe for VI Semester

GUIDED BY

Dr.M.SIDDAPPA B.E,M.Tech,Ph.D
Professor and Head, Dept of CSE

.....
Signature of the Guide

.....
Signature of the H.O.D

Dr.M.SIDDAPPA B.E,M.Tech,Ph.D
Professor and Head , Dept of CSE

ACKNOWLEDGMENT

This Mini-Project will be incomplete without thanking the personalities responsible for this venture, which otherwise would not have become a reality.

We express our profound gratitude to **DR. M. S. RAVIPRAKASHA, Principal, SSIT, Tumkur** for his moral support in completing our Mini-Project work.

We would like to thank the **DR. M. SIDDAPPA Head, Department of CSE SSIT, Tumkur** for providing all the facilities and also acting as our guide, helping and sharing technical expertise and timely advice.

We would like to thank our sincere gratitude to all teaching and non-teaching faculty of the department of CSE for guiding us in the completing MiniProject by giving valuable suggestions and encouragement.

By

ADITYA P SHETTY(19CS002)

ANSUL (19CS013)

ABSTRACT

Area of work

This Road sign recognition system is built in the field of Machine Learning(ML) which is the branch of Artificial Intelligence(AI) that focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Proposed work

To build a program that is able to train a neural network using our dataset and create an image classification system that will be able to identify and differentiate between various different types of road signs using a convolutional neural network(CNN) which is built on an algorithm that works similarly to that of the part of the human brain that can identify images.

Platform

For this project, we are using python as it provides the various libraries that have become a requirement for machine learning programs, these libraries include tensor-flow, Keras, OpenCV, and a large number of projects that are produced in this field already use Python as a platform, and the other platform that we are using to build this project include Google Collab and Jupyter notebooks to handle the python notebooks, Microsoft excel to create and edit the required CSV files and Kaggle for the required dataset.

Expected Results

In Machine learning, we feed in DATA(Input) + Output, run it on the machine during training and the machine creates its program(logic), which can be evaluated while testing.

CONTENTS	Page no
1. Introduction	6
1.1 Machine learning	6
1.2 Project Overview	7
1.3 Problem Statement	7
1.4 Objective	7
2. Software Requirement Specification	8
2.1 Functional Requirements	8
2.2 Non-functional requirement	8
3. Software Development Requirements	9
3.1 Development software	9
3.1.1 Jupyter Notebook	9
3.1.2 Google colab	9
3.1.3 Libraries	10
3.2 Hardware specification	11
4. Design	12
4.1 Architecture Diagram	12
4.1.1 System architecture	12
4.1.2 The different layers of the Neural Network	13
4.2 Module description	15
4.2.1 Training	15
4.2.2 Execution	16
4.3 Functionality Description	16
4.3.1 Image Classification	16
4.3.2 The model used: CNN	16

5. Implementation and Results	18
5.1 Training	18
5.2 Execution	18
5.3 Results	18
6. Screenshots	20
7. Conclusion and Future Enhancements	27

1.INTRODUCTION

1.1 Machine learning

In this project, we are working on Machine learning under the branch of AI which has some fundamental differences from traditional programming

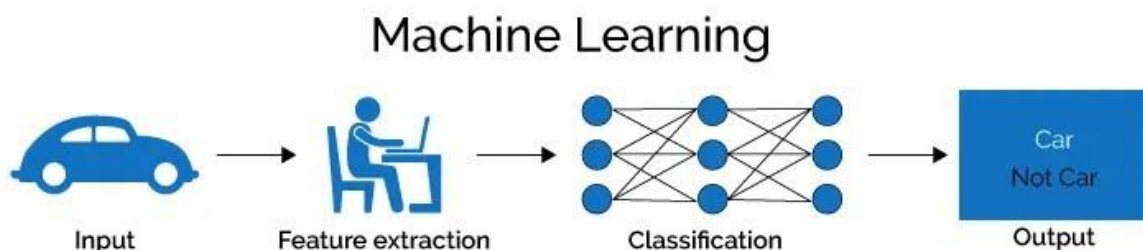
- **Traditional Programming:** We feed in DATA (Input) + PROGRAM (logic), run it on the machine, and get the output.
- **Machine Learning:** We feed in DATA(Input) + Output, run it on the machine during training and the machine creates its program(logic), which can be evaluated while testing.

Under the Machine learning bracket, we are working under a specific area called Image classification.

Machine learning is an application of AI that enables systems to learn and improve from experience without being explicitly programmed. Machine learning focuses on developing computer programs that can access data and use it to learn for themselves.

As a discipline, machine learning explores the analysis and construction of algorithms that can learn from and make predictions on data.

ML has proven valuable because it can solve problems at a speed and scale that cannot be duplicated by the human mind alone. With massive amounts of computational ability behind a single task or multiple specific tasks, machines can be trained to identify patterns in and relationships between input data and automate routine processes.



In the above example, machine learning has been used to classify whether a vehicle is a car or not.

INPUT refers to the data that is provided to the machine learning model

FEATURE EXTRACTION refers to the extraction of features from the input here in this example we can consider height, width, side profile,,, etc as the features

These features are then fed into the neural network and CLASSIFICATION is performed and the car is put into either car or not car class as OUTPUT.

1.2 Project Overview

This project is divided into two parts

i. Training: In this part of the project we will train a machine learning model using machine learning algorithms, optimizers and other functionalities using Tensor-flow and Keras.

ii. Execution: In this part, we will use the trained model to predict the road signals using a camera as a visual medium a bulk of this is achieved using the cv2 library.

1.3 Problem Statement

The road sign provides significant information that can help drive in a manner that is safe for the driver and other road users, but such vital information is disregarded as most road accidents are attributed to either reduced attention of drivers or that they simply choose to ignore the road signs.

Traffic signs are an important form of communication between the driver and the road but this important information may sometimes slip the eyes of the driver which may lead to disastrous consequences.

In traffic environments, Traffic Sign Recognition System is used to regulate the traffic signs, warn drivers, and command or prohibit certain actions. Fast Real-Time and robust traffic sign detection and recognition can support and disburden the driver and thus significantly increase driving safety and comfort. An automated road sign recognition system may play an important role in alerting the drivers of road conditions making driving safer.

The weather conditions like rain and sometimes heavy fog and dew, especially during the early morning and late evening, also have been reported as some of the causes of many accident cases. This is because of the reduced visibility during such conditions, in these conditions cameras may be more capable of gathering information than a human eye which will lead to better traffic safety.

It will also play an important role in self-driving cars which are being worked on extensively and may be part of our everyday life in the near future.

1.4 Objective

- To create a more secure driving environment and reduce the rate of accidents that are caused due to the ignorance shown towards traffic signs.
- To create a technology for the evolving world that will be useful in the field of automobiles as self-driving vehicles become more prevalent.
- A system that can be used under lower visibility where the rate of human error is higher, one specific example we can consider is truck driving during the night.

2.SOFTWARE REQUIREMENT SPECIFICATION

2.1 Functional Requirements

- Show the images that have been processed to get an idea of how the grayscale and equalized images look.
- The training program must produce a model that is able to make predictions about the road signs.
- Access to access the laptop camera.
- The program must be able to predict the road sign presented in front of the camera.
- The program should open a different window with the prediction class, prediction class name, and probable accuracy along with the image that is predicted.
- You should be able to stop execution using the hotkey (here we have used 'q').

2.2 Non-functional Requirements

- The model must make accurate predictions at least 9 out of 10 times.
- The system must make said predictions in 2secs.
- The system must be reliable and the must not be any frequent crashes.

3. SOFTWARE DEVELOPMENT REQUIREMENTS

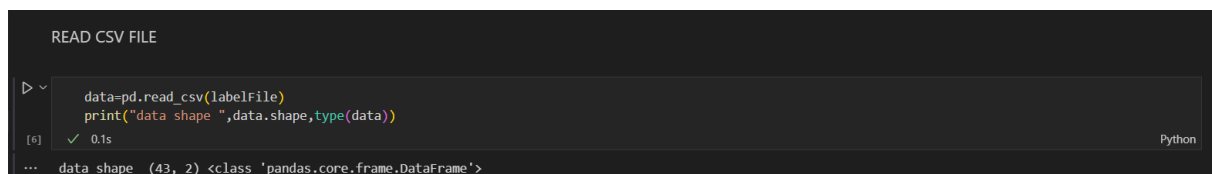
3.1 Development software

3.1.1 Jupyter Notebook (using Visual Studio Code)

The **Jupyter Notebook** is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

Visual studio code has been used as we prefer and are more familiar with the UI but there is no functionality of visual studio itself all the functionality comes from the jupyter notebook.

Jupyter notebook acts as an enhanced python IDE it provides cells that can be used to execute code in chunks this helps in understanding code in smaller bytes and also show the immediate output for the chunk of code.



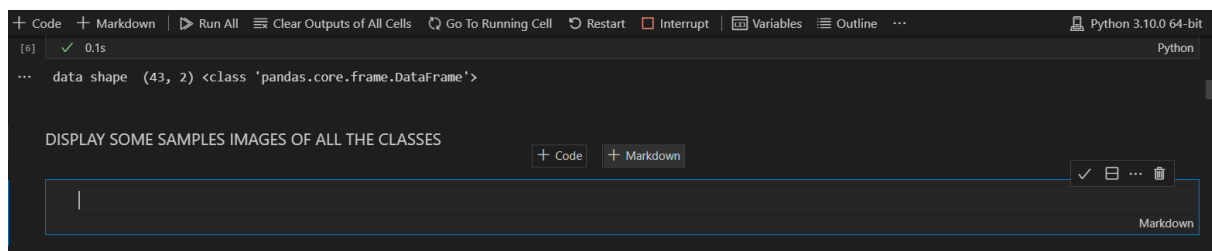
```
READ CSV FILE
```

```
data=pd.read_csv(labelFile)
print("data shape ",data.shape,type(data))
```

[6] ✓ 0.1s Python

... data shape (43, 2) <class 'pandas.core.frame.DataFrame'>

In the above example we can observe that the output is shown immediately after the code in the cell this simplifies the processes and helps us better understand the execution of the program.



Python 3.10.0 64-bit

```
[6] ✓ 0.1s Python
```

```
... data shape (43, 2) <class 'pandas.core.frame.DataFrame'>
```

DISPLAY SOME SAMPLES IMAGES OF ALL THE CLASSES

+ Code + Markdown

✓ ☰ ... 🗑️

Markdown

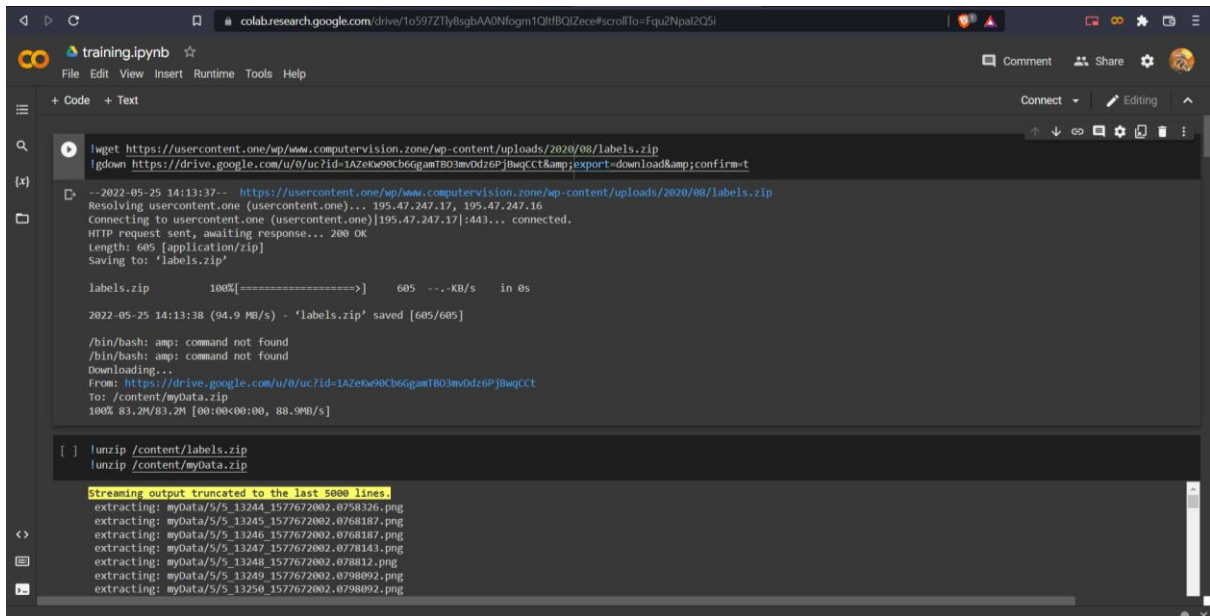
Jupyter also provides markdown cells which provide the same functionality of comments but are more visually appealing and more presentable.

3.1.2 Google Colab

Colaboratory or Colab for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

In short Google colab is a cloud service provided by Google where we can access Google's powerful computers in order to execute our programs faster although colab itself is

sufficient in most cases here we use it only while building and testing our code as it is faster than our pc but in our case, as we need access to the webcam in this project we will not be able to execute our prediction program in colab as it is just easier to just execute the final programs on our pc.



```
!wget https://usercontent.one/wp-content/uploads/2020/08/labels.zip
!gdown https://drive.google.com/u/0/uc?id=1AZetw90Cb6ggamT803mw0dz6Pj8wqCt&export=download&confirm=t

--2022-05-25 14:13:37-- https://usercontent.one/wp-content/uploads/2020/08/labels.zip
Resolving usercontent.one (usercontent.one)... 195.47.247.17, 195.47.247.16
Connecting to usercontent.one (usercontent.one)|195.47.247.17|:443... connected.
HTTP request sent, awaiting response... 200 OK
length: 605 [application/zip]
Saving to: 'labels.zip'

labels.zip      100%[=====] 605  --.-KB/s  in 0s

2022-05-25 14:13:38 (94.9 MB/s) - 'labels.zip' saved [605/605]

/bin/bash: amp: command not found
/bin/bash: amp: command not found
Downloading...
From: https://drive.google.com/u/0/uc?id=1AZetw90Cb6ggamT803mw0dz6Pj8wqCt
To: /content/myData.zip
100% 83.2M/83.2M [00:00<00:00, 88.9MB/s]

[ ] !unzip /content/labels.zip
[ ] !unzip /content/myData.zip

Streaming output truncated to the last 5000 lines
extracting: myData/5/5_1324_1577672002_0798126.png
extracting: myData/5/5_13245_1577672002_0768187.png
extracting: myData/5/5_13246_1577672002_0768187.png
extracting: myData/5/5_13247_1577672002_0778143.png
extracting: myData/5/5_13248_1577672002_078812.png
extracting: myData/5/5_13249_1577672002_0798092.png
extracting: myData/5/5_13250_1577672002_0798092.png
```

3.1.3 Libraries

In this project, we use many libraries the important ones are listed below:

NumPy: NumPy brings the computational power of languages like C and Fortran to Python, a language much easier to learn and use. With this power comes simplicity: a solution in NumPy is often clear and elegant.

Matplotlib: Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.

Pandas: pandas is a fast, powerful, flexible and easy-to-use open-source data analysis and manipulation tool, built on top of the Python programming language.

Tensorflow: The core open-source library to develop and train ML models. 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.

Keras: Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result as fast as possible is key to doing good research.*

Cv2: OpenCV (Open-source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception.

and many more.

```
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.optimizers import Adam
from keras.utils.np_utils import to_categorical
from keras.layers import Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
import cv2
from sklearn.model_selection import train_test_split
import pickle
import os
import pandas as pd
import random
from keras.preprocessing.image import ImageDataGenerator
```

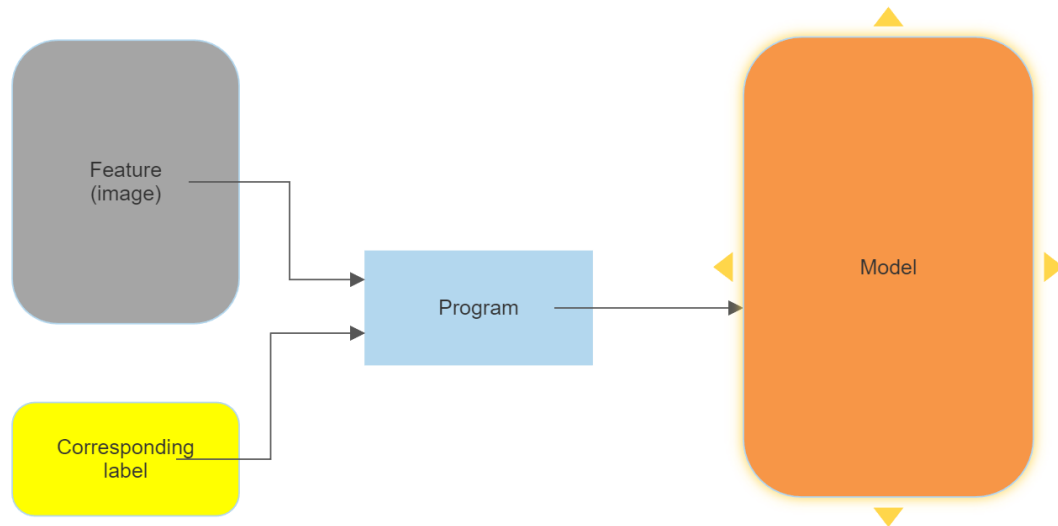
3.2 Hardware specification and software requirements

- Operating system: Windows 10-11
- Language: Python 10
- CPU: Equivalent to a Ryzen 5-5000
- Ram: 8 GB
- GPU: equivalent to vega 8(integrated), dedicated GPU will lead to faster processing.

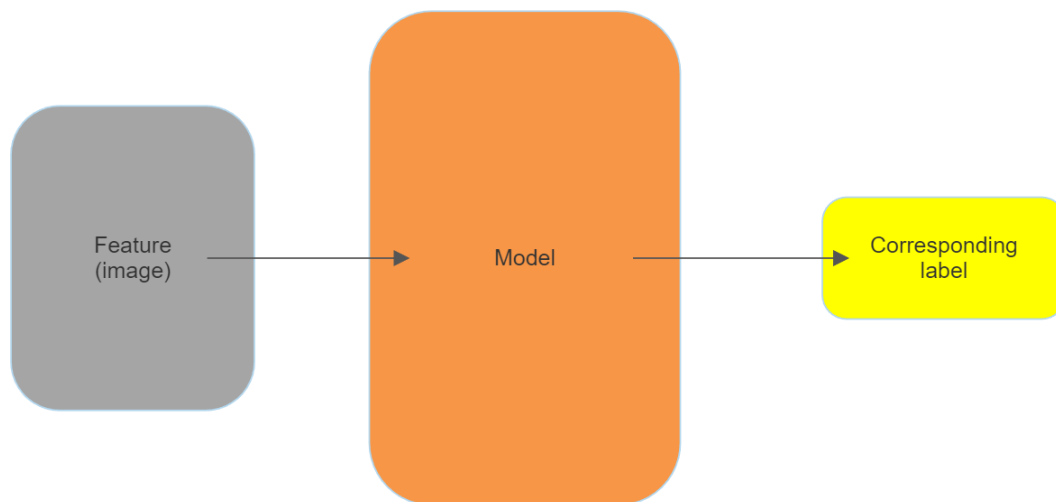
4. DESIGN

4.1 Architecture Diagram

4.1.1 System architecture

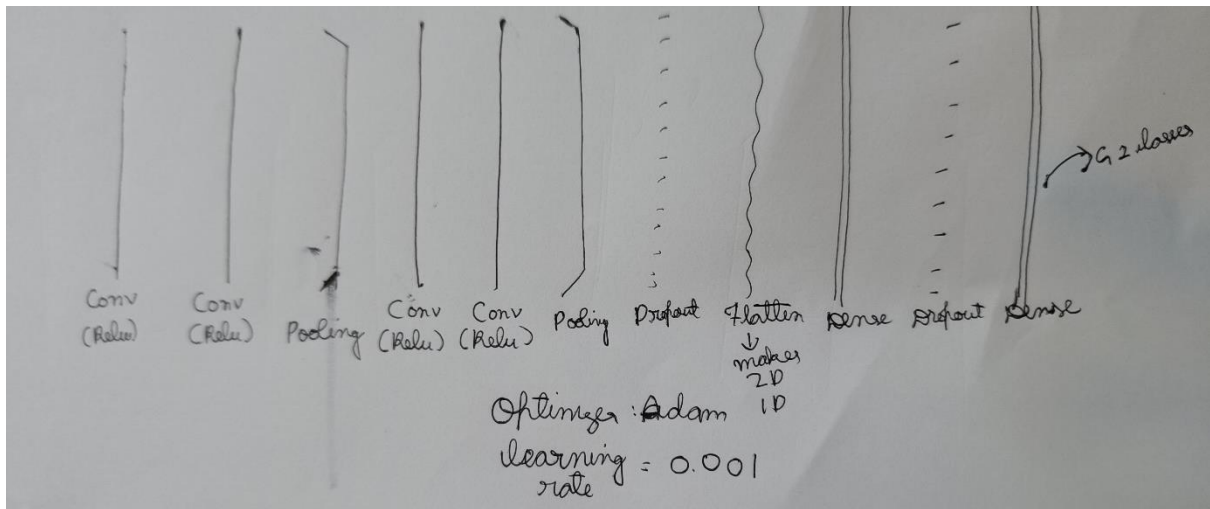


The above diagram shows us how the training program uses the features or images of the road signs with their corresponding label that consists of the class number and the class name of the road sign in the feature, to train and produce a prediction.



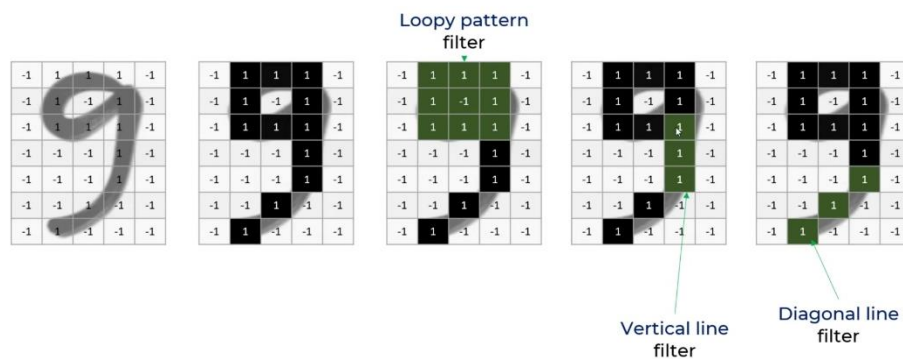
The prediction model that was produced will receive input as the features or images and the model will predict the images class number and the class name (corresponding label).

4.1.2 The different layers of the Neural Network

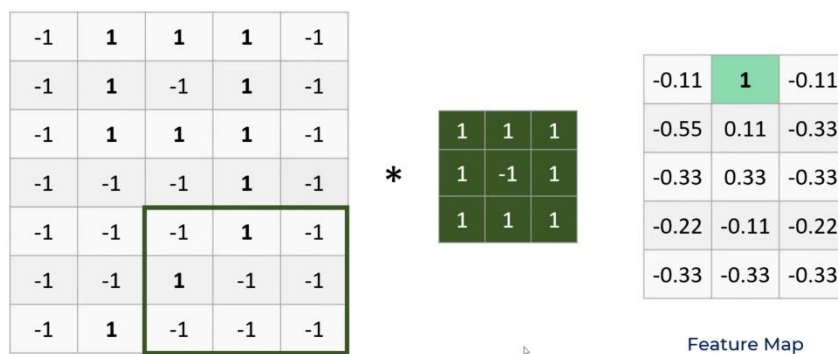


The image above shows the various layers used in the neural network

Conv: CNN has their “neurons” arranged more like those of the frontal lobe, the area responsible for processing visual stimuli in humans and other animals. The layers of neurons are arranged in such a way as to cover the entire visual field avoiding the piecemeal image processing problem of traditional neural networks.



CNN uses feature extraction, CNN identifies common features in the image and creates a filter for each such feature it uses this filter to create a feature map this feature map is used to create neurons for the next layer.



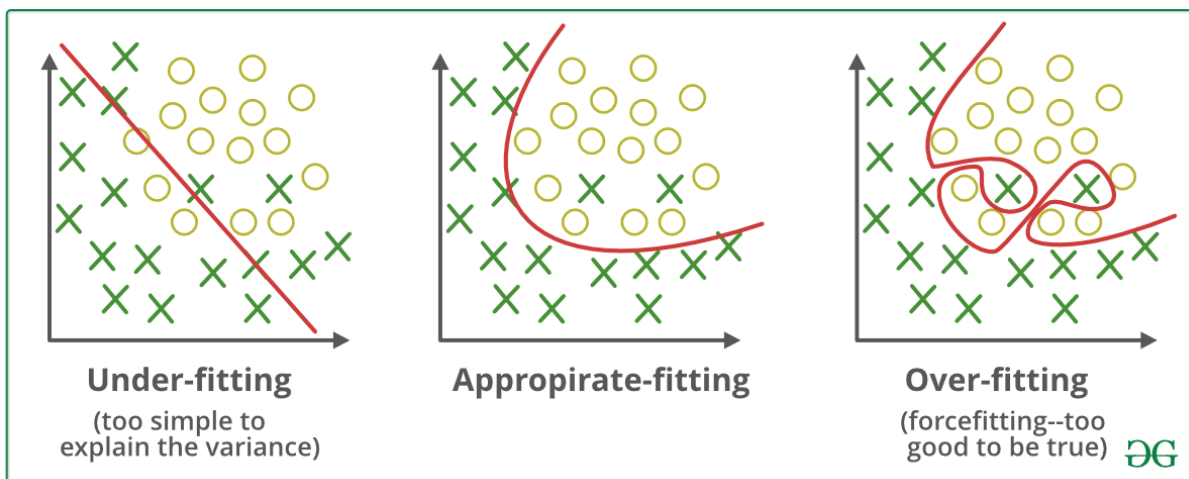
Pooling(Max Pooling): Here we use pooling to reduce the size of the feature map and hence reduce computation, it does this by choosing the maximum value in a particular window that is smaller than the size of the feature map.

5	1	3	4
8	2	9	2
1	3	0	1
2	2	2	0

8	9
3	2

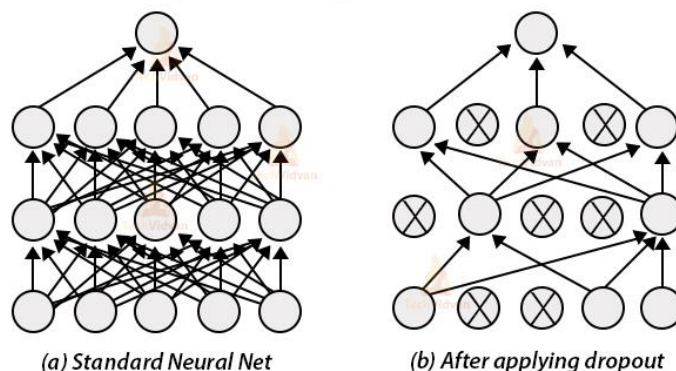
2 by 2 filter with stride = 2

Dropout: When you have a large number of layers we can have the problem of overfitting.

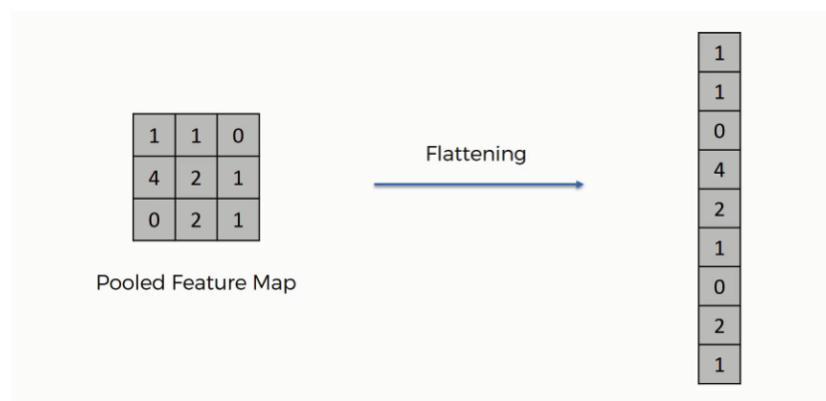


Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. The term "dropout" refers to dropping out units (both hidden and visible) in a neural network.

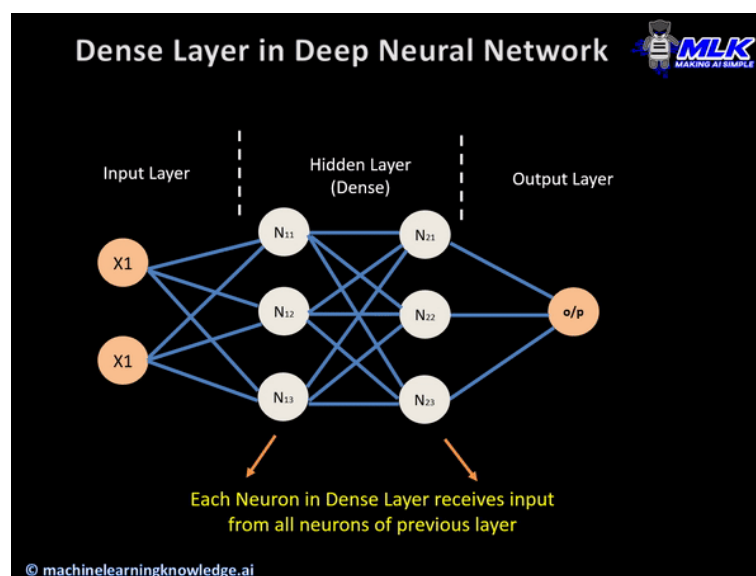
Dropout Layer in Keras



Flatten: Flatten is used to convert the 2D data into simple 1D data.



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



4.2 Module description

4.2.1 Training

First, we import the necessary libraries, such as TensorFlow, Keras, pandas, NumPy, OpenCV, etc. Next, we count the number of classes and import the images. Finally, we divide the data into three sets, training, testing, and validation, and see if the number of images matches the number of labels for each set. Finally, we read the CSV file containing the information about which images belong to which class and display a bar graph regarding this. Pre-process the images to make them simpler for the computer to process and to reduce the amount of work required. Additionally, augment the photos to provide more general and varied training data for the model. Create the model using the `model.add()` function to add layers to the network, and train the model, Plot a graph to examine how accuracy and loss change over time. Store the model in h5 format, and that's it.

4.2.2 Execution

Import the necessary libraries first and adjust the resolution of the camera. Determine the probability threshold then pick the video camera and set it up.

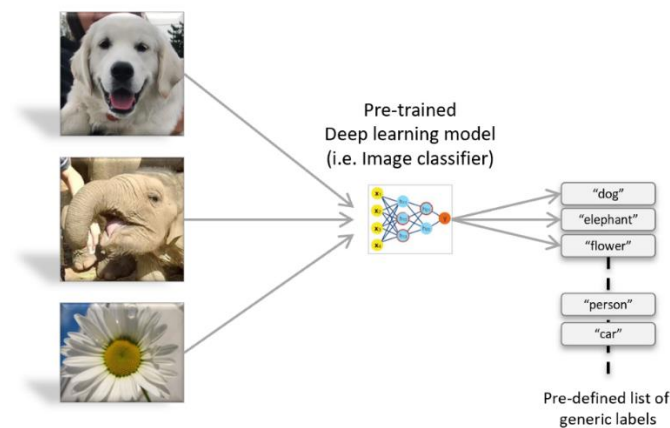
Add a while(True) loop in which, pre-process the incoming image similar to how you would during training, and read the image, make a prediction based on the image using the trained model, if probability > threshold, the image with its class number, class name, and probability should be displayed.

When the hotkey (here, "q") is pushed, the execution is stopped.

4.3 Functionality Description

4.3.1 Image Classification

Image classification involves teaching an Artificial Intelligence (AI) how to detect objects in an image based on their unique properties.



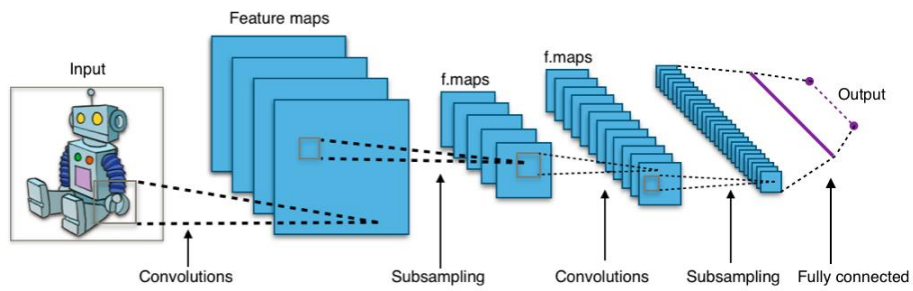
An image classification among dogs, elephants, flowers, humans, and a car

It is one of the areas where Deep learning models are very successfully applied to practical applications. It is an active area of research. Many approaches have been proposed and many more are popping up.

4.3.2 The model used: CNN

A convolutional neural network (CNN) is a form of artificial neural network that is specifically intended to process pixel input and is used in image recognition and processing.

A neural network is a hardware and/or software system that mimics the behaviour of neurons in the brain. Traditional neural networks aren't designed for image processing and must be fed images in smaller chunks. CNN's "neurons" are structured more like those in the frontal lobe, the area in humans and other animals responsible for processing visual inputs.



The above image shows the working of the CNN which is similar to the car classification example shared before, that is a feature is recognized feature maps are created for the said feature which for images is a 2D matrix with numbers ranging from 0 to 255.

5.IMPLEMENTATION AND RESULTS

5.1 Training

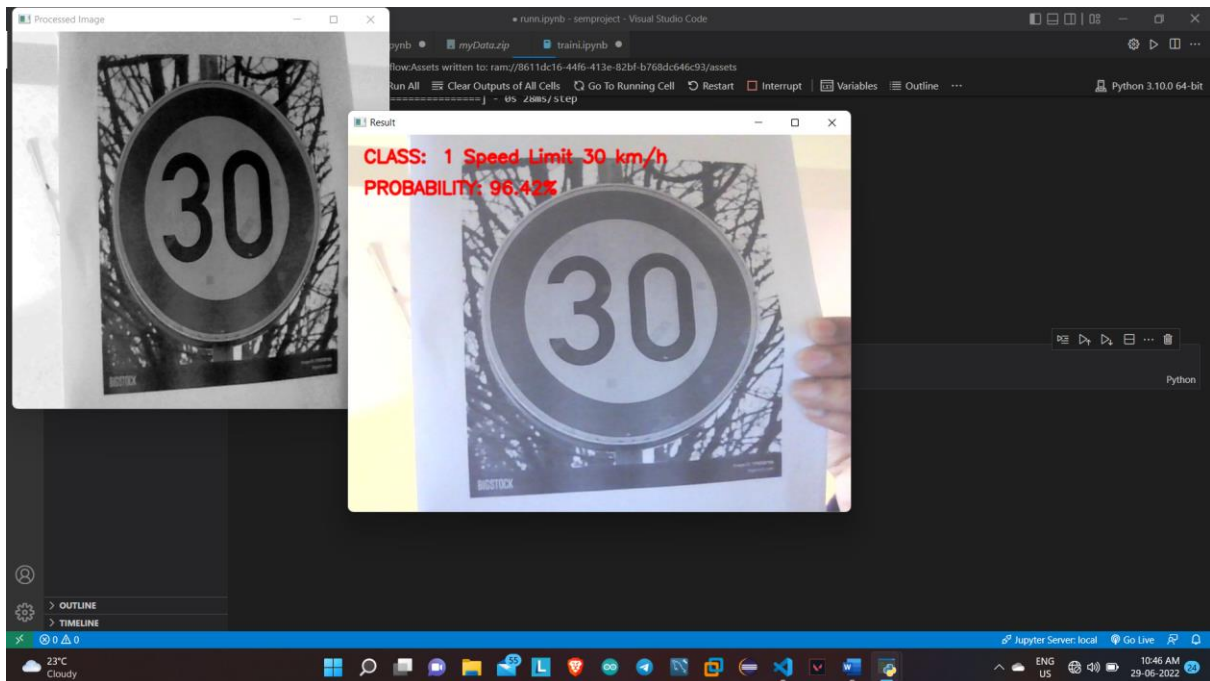
1. Import the required libraries this contains many libraries like TensorFlow, Keras, pandas, NumPy, OpenCV, etc.
2. Count the number of classes and import the images.
3. Split the data into three sets that are training set, testing set, and validation set.
4. Check if the number of images matches the number of labels for each of the three sets.
5. Read the CSV file containing the info about which images belong to which class.
6. Display some sample images for each class
7. Display a bar graph showing the number of samples of each class.
8. Pre-process the images so that they are easier to process for the machine and also cost less computation.
9. Augment the images so that we can have a more generic and more examples to train the model.
10. Create the model using the model.add() function to add layers to the network.
11. Train the model.
12. Plot a graph to see the changes in loss with each epoch.
13. Plot a graph to see the changes in accuracy with each epoch.
14. Store the model in h5 format.

5.2 Execution

1. Import the required libraries.
2. Set the camera resolution.
3. Set the probability threshold.
4. Choose the camera and setup the video camera.
5. Enter a while(True) loop.
6. Pre-process the incoming image in a similar way to that of training.
7. Read the image.
8. Predict the image using the model created by training.
9. If probability>threshold then display the image with its class number, class name, and probability.
10. Stop the execution when the hotkey is pressed (here 'q').

5.3 Result

The result is the class number, class name, and probability along with the image for which this is determined



6.SCREENSHOTS

```
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.optimizers import Adam
from keras.utils.np_utils import to_categorical
from keras.layers import Dropout, Flatten
from keras.layers.convolutional import Conv2D, MaxPooling2D
import cv2
from sklearn.model_selection import train_test_split
import pickle
import os
import pandas as pd
import random
from keras.preprocessing.image import ImageDataGenerator
```

✓ 8.8s

Python

Parameters

```
path = "myData" # folder with all the class folders
labelFile = 'labels.csv' # file with all names of classes
batch_size_val=50 # how many to process together
steps_per_epoch_val=435
epochs_val=10
imageDimensions = (32,32,3)
testRatio = 0.2 # if 1000 images split will 200 for testing
validationRatio = 0.2 # if 1000 images 20% of remaining 800 will be 160 for validation
```

✓ 0.7s

Python

Importing of the Images

+ Code

+ Markdown

```
count = 0
images = []
classNo = []
myList = os.listdir(path)
print("Total Classes Detected:",len(myList))
noOfClasses=len(myList)
print("Importing Classes.....")
for x in range (0,len(myList)):
    myPicList = os.listdir(path+"/"+str(count))
    for y in myPicList:
        curImg = cv2.imread(path+"/"+str(count)+"/"+y)
        images.append(curImg)
        classNo.append(count)
    print(count, end = " ")
    count +=1
print(" ")
images = np.array(images)
classNo = np.array(classNo)
```

✓ 3m 50.2s

Python

Total Classes Detected: 43

Importing Classes.....

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42

Split Data

```
X_train, X_test, y_train, y_test = train_test_split(images, classNo, test_size=testRatio)
X_train, X_validation, y_train, y_validation = train_test_split(X_train, y_train, test_size=validationRatio)
```

✓ 0.1s Python

TO CHECK IF NUMBER OF IMAGES MATCHES TO NUMBER OF LABELS FOR EACH DATA SET

```
print("Data Shapes")
print("Train",end = "");print(X_train.shape,y_train.shape)
print("Validation",end = "");print(X_validation.shape,y_validation.shape)
print("Test",end = "");print(X_test.shape,y_test.shape)
assert(X_train.shape[0]==y_train.shape[0]), "The number of images in not equal to the number of lables in training set"
assert(X_validation.shape[0]==y_validation.shape[0]), "The number of images in not equal to the number of lables in validation set"
assert(X_test.shape[0]==y_test.shape[0]), "The number of images in not equal to the number of lables in test set"
assert(X_train.shape[1]==(imageDimesions)), "The dimesions of the Training images are wrong "
assert(X_validation.shape[1]==(imageDimesions)), "The dimesionas of the Validation images are wrong "
assert(X_test.shape[1]==(imageDimesions)), "The dimesionas of the Test images are wrong"
```

✓ 0.7s Python

Data Shapes

Train(22271, 32, 32, 3) (22271,)

Validation(5568, 32, 32, 3) (5568,)

Test(6960, 32, 32, 3) (6960,)

READ CSV FILE

```
data=pd.read_csv(labelFile)
print("data shape ",data.shape,type(data))
```

✓ 0.1s Python

data shape (43, 2) <class 'pandas.core.frame.DataFrame'>

DISPLAY SOME SAMPLES IMAGES OF ALL THE CLASSES

```
num_of_samples = []
cols = 5
num_classes = noofClasses
fig, axs = plt.subplots(nrows=num_classes, ncols=cols, figsize=(5, 300))
fig.tight_layout()
for i in range(cols):
    for j,row in data.iterrows():
        x_selected = X_train[y_train == j]
        axs[j][i].imshow(x_selected[random.randint(0, len(x_selected)- 1), :, :], cmap=plt.get_cmap("gray"))
        axs[j][i].axis("off")
        if i == 2:
            axs[j][i].set_title(str(j)+ " -> row["Name"])
            num_of_samples.append(len(x_selected))
```

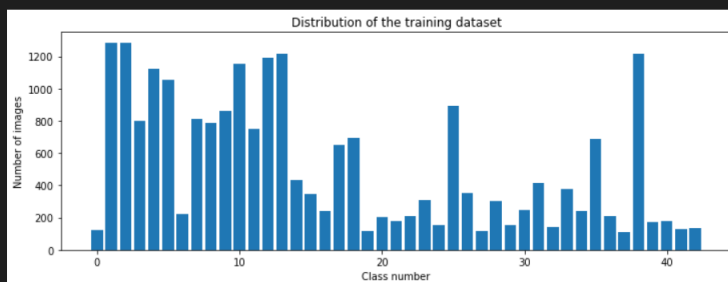
✓ 16.5s Python

DISPLAY A BAR CHART SHOWING NO OF SAMPLES FOR EACH CATEGORY

```
print(num_of_samples)
plt.figure(figsize=(12, 4))
plt.bar(range(0, num_classes), num_of_samples)
plt.title("Distribution of the training dataset")
plt.xlabel("Class number")
plt.ylabel("Number of images")
plt.show()
```

✓ 0.4s Python

[124, 1284, 1286, 802, 1122, 1052, 221, 814, 789, 862, 1151, 748, 1194, 1215, 431, 346, 239, 650, 697, 114, 202, 178, 209, 307, 154, 890, 351, 115, 306, 154, 245, 413, 143, 376, 241, 690, 209, 111, 1218, 170, 180, 131, 137]



PREPROCESSING THE IMAGES

```
def grayscale(img):
    img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    return img
def equalize(img):
    img = cv2.equalizeHist(img)
    return img
def preprocessing(img):
    img = grayscale(img)      # CONVERT TO GRAYSCALE
    img = equalize(img)      # STANDARDIZE THE LIGHTING IN AN IMAGE
    img = img/255            # TO NORMALIZE VALUES BETWEEN 0 AND 1 INSTEAD OF 0 TO 255
    return img

X_train=np.array(list(map(preprocessing,X_train))) # TO ITERATE AND PREPROCESS ALL IMAGES
X_validation=np.array(list(map(preprocessing,X_validation)))
X_test=np.array(list(map(preprocessing,X_test)))
```

✓ 1.9s

Python

ADD A DEPTH OF 1

```
X_train=X_train.reshape(X_train.shape[0],X_train.shape[1],X_train.shape[2],1)
X_validation=X_validation.reshape(X_validation.shape[0],X_validation.shape[1],X_validation.shape[2],1)
X_test=X_test.reshape(X_test.shape[0],X_test.shape[1],X_test.shape[2],1)
```

✓ 0.1s

Python

AUGMENTATION OF IMAGES: TO MAKE IT MORE GENERIC

```
dataGen= ImageDataGenerator(width_shift_range=0.1, # 0.1 = 10% IF MORE THAN 1 E.G 10 THEN IT REFERS TO NO. OF PIXELS EG 10 PIXELS
                             height_shift_range=0.1,
                             zoom_range=0.2, # 0.2 MEANS CAN GO FROM 0.8 TO 1.2
                             shear_range=0.1, # MAGNITUDE OF SHEAR ANGLE
                             rotation_range=10) # DEGREES
dataGen.fit(X_train)
batches= dataGen.flow(X_train,y_train,batch_size=20) # REQUESTING DATA GENERATOR TO GENERATE IMAGES BATCH SIZE = NO. OF IMAGES CREATED EACH TIME ITS CALLED
X_batch,y_batch = next(batches)
```

✓ 0.2s

Python

TO SHOW AGMENTED IMAGE SAMPLES

```
fig,axs=plt.subplots(1,15,figsize=(20,5))
fig.tight_layout()

for i in range(15):
    axs[i].imshow(X_batch[i].reshape(imageDimensions[0],imageDimensions[1]))
    axs[i].axis('off')
plt.show()

y_train = to_categorical(y_train,noOfClasses)
y_validation = to_categorical(y_validation,noOfClasses)
y_test = to_categorical(y_test,noOfClasses)
```

✓ 1.7s

Python



CONVOLUTION NEURAL NETWORK MODEL

```
def myModel():
    no_of_filters=60
    size_of_filter=(5,5) # THIS IS THE KERNEL THAT MOVE AROUND THE IMAGE TO GET THE FEATURES.
    # THIS WOULD REMOVE 2 PIXELS FROM EACH BORDER WHEN USING 32 32 IMAGE
    size_of_filter2=(3,3)
    size_of_pool=(2,2) # SCALE DOWN ALL FEATURE MAP TO GENERALIZE MORE, TO REDUCE OVERFITTING
    no_of_nodes = 500 # NO. OF NODES IN HIDDEN LAYERS
    model= Sequential()
    model.add(Conv2D(no_of_filters,size_of_filter,input_shape=(imageDimensions[0],imageDimensions[1],1),activation='relu')) # ADDING MORE CONVOLUTION
    model.add(Conv2D(no_of_filters, size_of_filter, activation='relu'))
    model.add(MaxPooling2D(pool_size=size_of_pool)) # DOES NOT EFFECT THE DEPTH/NO OF FILTERS

    model.add(Conv2D(no_of_filters//2, size_of_filter2,activation='relu'))
    model.add(Conv2D(no_of_filters // 2, size_of_filter2, activation='relu'))
    model.add(MaxPooling2D(pool_size=size_of_pool))
    model.add(Dropout(0.5))

    model.add(Flatten())
    model.add(Dense(no_of_nodes,activation='relu'))
    model.add(Dropout(0.5)) # INPUTS NODES TO DROP WITH EACH UPDATE 1 ALL 0 NONE
    model.add(Dense(noOfClasses,activation='softmax')) # OUTPUT LAYER
    # COMPILER MODEL
    model.compile(Adam(lr=0.001),loss='categorical_crossentropy',metrics=['accuracy'])
    return model
```

✓ 0.1s

Python

TRAIN

```
model = myModel()
print(model.summary())
history=model.fit_generator(dataGen.flow(X_train,y_train,batch_size=batch_size_val),steps_per_epoch=steps_per_epoch_val,epochs=epochs_val,validation_data=(X_val,y_val))
```

✓ 15m 13.8s

Python

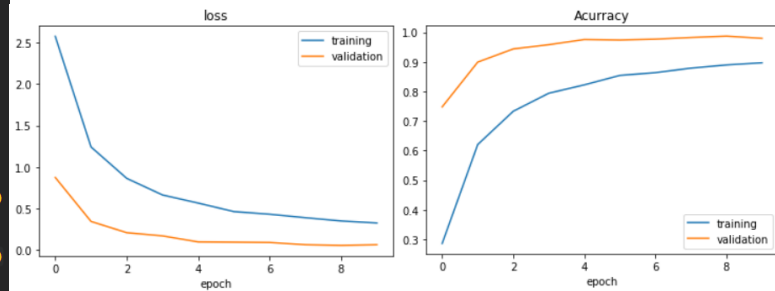
Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 28, 28, 60)	1560
conv2d_1 (Conv2D)	(None, 24, 24, 60)	90060
max_pooling2d (MaxPooling2D)	(None, 12, 12, 60)	0
conv2d_2 (Conv2D)	(None, 10, 10, 30)	16230
conv2d_3 (Conv2D)	(None, 8, 8, 30)	8130
max_pooling2d_1 (MaxPooling2D)	(None, 4, 4, 30)	0
dropout (Dropout)	(None, 4, 4, 30)	0
flatten (Flatten)	(None, 480)	0
dense (Dense)	(None, 500)	240500
dropout_1 (Dropout)	(None, 500)	0
...		
Total params: 378,023		
Trainable params: 378,023		
Non-trainable params: 0		


```

plt.figure(1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training', 'validation'])
plt.title('loss')
plt.xlabel('epoch')
plt.figure(2)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'])
plt.title('Accuracy')
plt.xlabel('epoch')
plt.show()
score = model.evaluate(X_test, y_test, verbose=0)
print('Test Score:', score[0])
print('Test Accuracy:', score[1])
model.save("D:\programs\semproject\model2.h5")

```

✓ 8.3s



STORE THE MODEL

```

model.save("D:\programs\semproject\model2.h5")
cv2.waitKey(0)

```

✓ 2.2s

```

import numpy as np
import cv2
import pickle
from tensorflow import keras
import os

#####

frameWidth= 640          # CAMERA RESOLUTION
frameHeight = 480
brightness = 180
threshold = 0.75         # PROBABILITY THRESHOLD
font = cv2.FONT_HERSHEY_SIMPLEX
#####

# SETUP THE VIDEO CAMERA
cap = cv2.VideoCapture(0)
cap.set(3, frameWidth)
cap.set(4, frameHeight)
cap.set(10, brightness)
# IMPORT THE TRAINED MODEL
path=".\\model2.h5"
model = keras.models.load_model(path)

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

def equalize(img):
    img =cv2.equalizeHist(img)
    return img

def preprocessing(img):
    img = grayscale(img)
    img = equalize(img)
    img = img/255
    return img

def getCalssName(classNo):

```

```

def getCalssName(classNo):
    if classNo == 0: return 'Speed Limit 20 km/h'
    elif classNo == 1: return 'Speed Limit 30 km/h'
    elif classNo == 2: return 'Speed Limit 50 km/h'
    elif classNo == 3: return 'Speed Limit 60 km/h'
    elif classNo == 4: return 'Speed Limit 70 km/h'
    elif classNo == 5: return 'Speed Limit 80 km/h'
    elif classNo == 6: return 'End of Speed Limit 80 km/h'
    elif classNo == 7: return 'Speed Limit 100 km/h'
    elif classNo == 8: return 'Speed Limit 120 km/h'
    elif classNo == 9: return 'No passing'
    elif classNo == 10: return 'No passing for vechiles over 3.5 metric tons'
    elif classNo == 11: return 'Right-of-way at the next intersection'
    elif classNo == 12: return 'Priority road'
    elif classNo == 13: return 'Yield'
    elif classNo == 14: return 'Stop'
    elif classNo == 15: return 'No vechiles'
    elif classNo == 16: return 'Vechiles over 3.5 metric tons prohibited'
    elif classNo == 17: return 'No entry'
    elif classNo == 18: return 'General caution'
    elif classNo == 19: return 'Dangerous curve to the left'
    elif classNo == 20: return 'Dangerous curve to the right'
    elif classNo == 21: return 'Double curve'
    elif classNo == 22: return 'Bumpy road'
    elif classNo == 23: return 'Slippery road'
    elif classNo == 24: return 'Road narrows on the right'
    elif classNo == 25: return 'Road work'
    elif classNo == 26: return 'Traffic signals'
    elif classNo == 27: return 'Pedestrians'
    elif classNo == 28: return 'Children crossing'
    elif classNo == 29: return 'Bicycles crossing'
    elif classNo == 30: return 'Beware of ice/snow'
    elif classNo == 31: return 'Wild animals crossing'
    elif classNo == 32: return 'End of all speed and passing limits'
    elif classNo == 33: return 'Turn right ahead'
    elif classNo == 34: return 'Turn left ahead'
    elif classNo == 35: return 'Ahead only'
    elif classNo == 36: return 'Go straight or right'
    elif classNo == 37: return 'Go straight or left'
    elif classNo == 38: return 'Keep right'
    elif classNo == 39: return 'Keep left'
    elif classNo == 40: return 'Roundabout mandatory'
    elif classNo == 41: return 'End of no passing'
    elif classNo == 42: return 'End of no passing by vechiles over 3.5 metric tons'
    else: return 'Unsure'

```

```

while True:

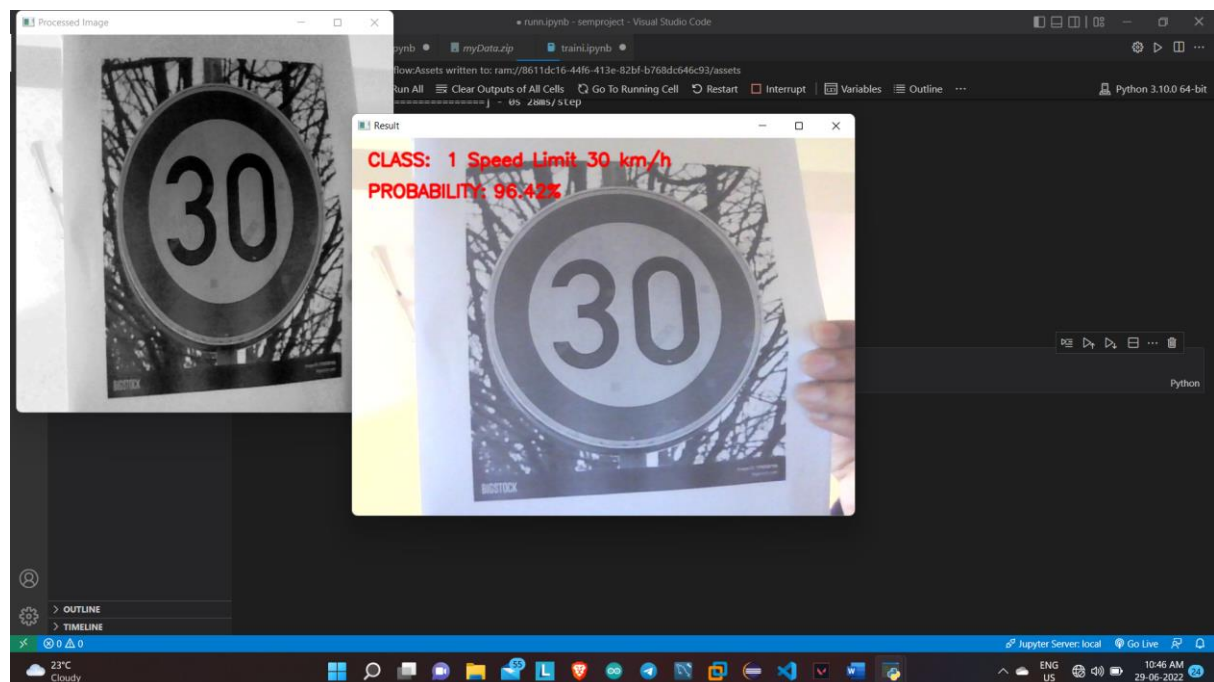
    # READ IMAGE
    success, imgOriginal = cap.read()

    # PROCESS IMAGE
    img = np.asarray(imgOriginal)
    img = cv2.resize(img, (480, 480))
    img = preprocessing(img)
    cv2.imshow("Processed Image", img)
    img = cv2.resize(img, (32,32))
    img = img.reshape(1, 32, 32, 1)
    cv2.putText(imgOriginal, "CLASS: ", (20, 35), font, 0.75, (0, 0, 255), 2, cv2.LINE_AA)
    cv2.putText(imgOriginal, "PROBABILITY: ", (20, 75), font, 0.75, (0, 0, 255), 2, cv2.LINE_AA)

    # PREDICT IMAGE
    predictions = model.predict(img)
    predict_x=model.predict(img)
    classIndex=np.argmax(predict_x)
    probabilityValue =np.amax(predictions)
    if probabilityValue > threshold:
        print(getCalssName(classIndex))
        cv2.putText(imgOriginal,str(classIndex)+" "+str(getCalssName(classIndex)), (120, 35), font, 0.75, (0, 0, 255), 2, cv2.LINE_AA)
        cv2.putText(imgOriginal, str(round(probabilityValue*100,2) )+"%", (180, 75), font, 0.75, (0, 0, 255), 2, cv2.LINE_AA)
        cv2.imshow("Result", imgOriginal)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

```



7.CONCLUSION AND FUTURE ENHANCEMENTS

Traffic Sign Recognition is a primary goal of almost all road environment understanding systems. The road sign provides significant information that can help drive in a manner that is safe for the driver and other road users. Most road accidents are attributed to either reduced attention of drivers or that they simply choose to ignore the road signs. The weather conditions like rain and sometimes heavy fog and dew, especially during the early morning and late evening, also have been reported as some of the causes of many accident cases. Therefore, the recognition of road signs would be of great help in order to reduce the number of traffic accidents and deaths. The development of road sign detection and recognition systems using image processing technology will ensure that each driver is aware of the rules and hazards on the road and will hopefully reduce the number of accidents and deaths. The Road Sign Recognition project is a field of applied computer vision research concerned with the automatic detection and classification of traffic signs. The application of the system will be helpful in improving road safety. The RSR can be a subsystem of the Driver Support System (DSS). The main aim is to provide DSS with the ability to understand its neighborhood environment and so permit advanced driver support such as collision prediction and avoidance. This application can also prove to be important to a robotic vehicle that automatically drives on the road.

The current accuracy in ideal conditions is around **97%**, but in practice, it is a bit lower hence the most important future work for this present topic would be to work on a collection of a large number of road sign images, and expand the database, retrain the neuron network. The system can be enhanced by using cost-effective techniques, which would assist the driver in notifying the distance between the road sign and the current position of the car. Even the system could be expanded to detect and differentiate between inanimate and living objects for example people crossing the roads.