TRAFFIC SIGN RECOGNITION

GROUP 7:
DHIRAJ AGARWAL
DAYANAND JAJODIA
ARNAB SAHA
HAIDER PARVEZ

Machine Learning

Machine learning (ML) is a type of artificial intelligence (AI) that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values.

Machine learning is important because it gives enterprises a view of trends in customer behavior and business operational patterns, as well as supports the development of new products. Many of today's leading companies, such as Facebook, Google and Uber, make machine learning a central part of their operations. Machine learning has become a significant competitive differentiator for many companies.

INTRODUCTION

Traffic signs are devices placed along, besides, or above a highway, roadway, pathway, or
other route to guide, warn, and regulate the flow of traffic, including motor vehicles, bicycles, pedestrians, equestrians, and other travelers. Since safety becomes more important for customers, Traffic Sign Recognition (TSR) becomes one of today's research subjects aiming to improve safety of driving. While, they are presently developed just to warn drivers about some important traffic signs, in the future, these systems may take control of the vehicle under some circumstances. The input is mainly consisting of visual data during a drive. According to those inputs, the action of driving is performed by the driver. Although, today's technology cannot process all visual inputs as a human being, by a system focusing on some specified portion of this process, the workload of drivers can be decreased.

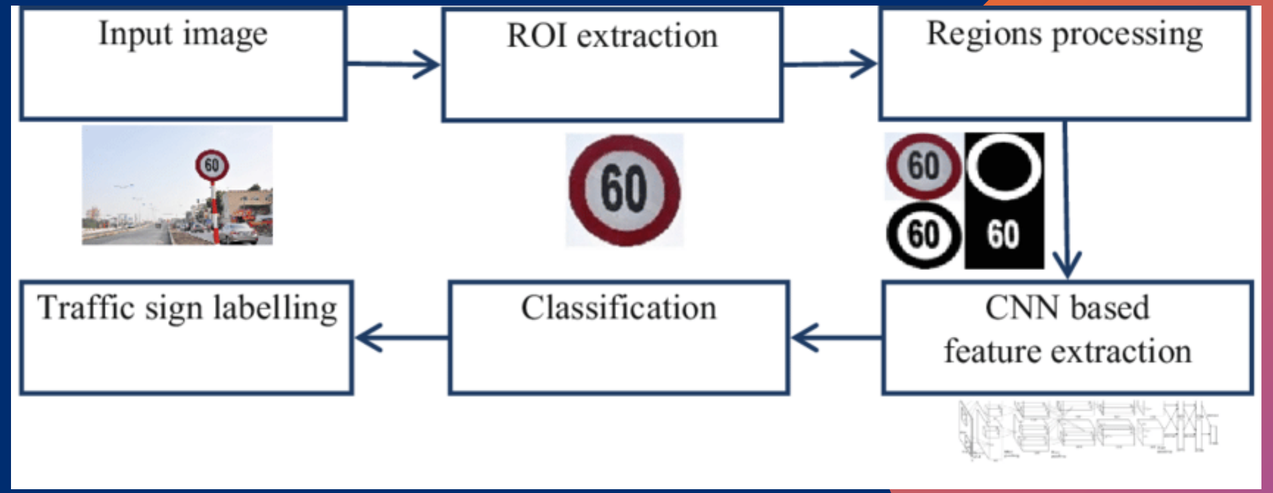
PROBLEM STATEMENT

In a professional context it often happens that private or corporate clients coder a publication to be made and presented with the actual content still not being ready. Think of a news blog that's filled with content hourly on the day of going live. However, reviewers tend to be distracted by comprehensible content, say, a random text copied from a newspaper or the internet. The are likely to focus on the text, disregarding the layout and its elements. factors, such as tiredness, sleeplessness or so. As a result, the safety of driving is improved. For this
purpose, TSR systems are developed, mainly to decrease the probability of missing some important traffic signs on the road. The problem may seem to be easy to handle, at the first glance. However, since the process on a visual data is performed by human brain, based on all of his experiences, it cannot be easy for a computer to perform the same process. Instead of those experiences, just some knowledge about distinctive features of traffic signs can be used. These features mainly consist of color and shape information. Although, this knowledge is not enough to separate traffic signs from other objects, the segmentation can be improved by the help of some intelligence while using the knowledge. For example, since the illumination is changing from time to time or from scene to scene, the color identification shall be performed accordingly. Additionally, since some traffic signs may be partially observed because of
some conditions, the shape information shall be extracted despite of those conditions.

SOLUTION APPROACH

Most traffic recognition algorithms divide the problem into three stages:

i. It does a rough segmentation to determine the location of the signs.
ii. It does a category determination.
iii. It performs a classification in which it identifies the exact content of the extracted traffic signs using various machine learning techniques.



LIBRARIES USED

01

import numpy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices.

02

import matplotlib

Matplotlib is a python library used to create 2D graphs and plots by using python scripts. It has a module named pyplot which makes things easy for plotting by providing feature to control line styles, font properties, formatting axes etc.

03

import pandas

Pandas is a game changer, and it is one of the most popular and commonly used tools in data analytics. Pandas is that it takes data from a CSV or TSV file or a SQL database and generates a Python object with rows and columns called a data frame, which looks remarkably similar to a table in statistics tools like Excel. Working with this is far more convenient than dealing with lists and/or dictionaries.

04

import PIL

The Python Imaging Library adds image processing capabilities to your Python interpreter.
This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

05

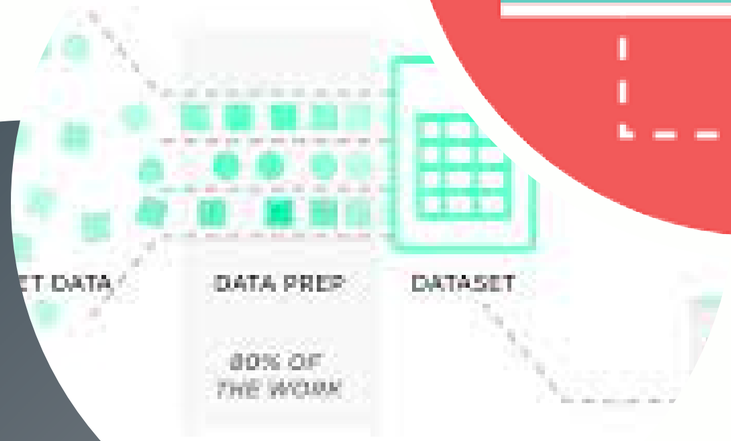
import tkinter

Tkinter is the de facto way in Python to create Graphical User interfaces (GUIs) and is included in all standard Python Distributions. In fact, it's the only framework built into the Python standard library. This Python framework provides an interface to the Tk toolkit and works as a thin object-oriented layer on top of Tk. The Tk toolkit is a cross-platform collection of 'graphical control elements', aka widgets, for building application interfaces.

06

from sklearn.model_selection import train_test_split

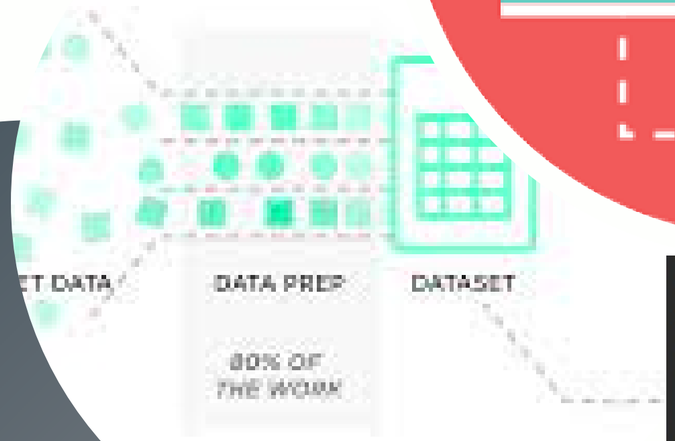
Split arrays or matrices into random train and test subsets. Quick utility that wraps input validation, next(ShuffleSplit().split(X, y)), and application to input data into a single call for splitting (and optionally subsampling) data into a one-liner.



DATASET

The data we have used in this project was in csv.file. It was taken from traffic.com . The dataset consists of 720 rows and 8 columns with no null values. Column data consist of independent Features. The independent features contain both categorical and numeric values.

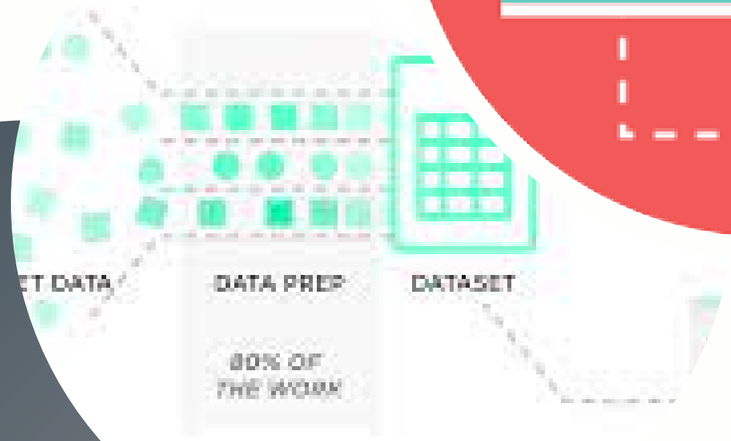
Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId	Path
53	54	6	5	48	49	16	Test/00000.png
42	45	5	5	36	40	1	Test/00001.png
48	52	6	6	43	47	38	Test/00002.png
27	29	5	5	22	24	33	Test/00003.png
60	57	5	5	55	52	11	Test/00004.png
52	56	5	5	47	51	38	Test/00005.png
147	130	12	12	135	119	18	Test/00006.png
32	33	5	5	26	28	12	Test/00007.png
45	50	6	5	40	45	25	Test/00008.png
81	86	7	7	74	79	35	Test/00009.png
38	37	6	5	33	32	12	Test/00010.png
45	44	6	5	40	39	7	Test/00011.png
79	73	7	7	72	67	23	Test/00012.png
36	37	5	6	31	32	7	Test/00013.png
43	41	5	5	37	36	4	Test/00014.png
27	27	6	6	22	22	9	Test/00015.png
37	38	5	6	31	32	21	Test/00016.png
32	33	5	5	27	28	20	Test/00017.png
35	35	5	6	30	29	27	Test/00018.png
34	40	6	6	29	35	38	Test/00019.png
32	33	5	6	27	28	4	Test/00020.png



DATA READING

It is the data that we need to load for starting any of the ML project. With respect to data, the most common format of data for ML projects is CSV (comma-separated values). Basically, CSV is a simple file format which is used to store tabular data (number and text) such as a spreadsheet in plain text.

```
C:\Users\dayan\Downloads\Python-Project-Traffic-Sign-Classification\Traffic sign classification\archive (1)\train\7
C:\Users\dayan\Downloads\Python-Project-Traffic-Sign-Classification\Traffic sign classification\archive (1)\train\8
C:\Users\dayan\Downloads\Python-Project-Traffic-Sign-Classification\Traffic sign classification\archive (1)\train\9
C:\Users\dayan\Downloads\Python-Project-Traffic-Sign-Classification\Traffic sign classification\archive (1)\train\10
C:\Users\dayan\Downloads\Python-Project-Traffic-Sign-Classification\Traffic sign classification\archive (1)\train\11
```



DATA MANIPULATION

Data manipulation is the process of changing data to make it easier to read or be easier to process.

```
[11]: 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))
      model.add(Dense(43, activation='softmax'))

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

```
(39209, 30, 30, 3) (39209,)
(31367, 30, 30, 3) (7842, 30, 30, 3) (31367,) (7842,)
Epoch 1/15
981/981 [==============================] - 110s 109ms/step - loss: 2.1574 - accuracy: 0.4442 - val_loss: 0.7532 - val_accuracy: 0.8020
Epoch 2/15
981/981 [==============================] - 307s 313ms/step - loss: 1.0230 - accuracy: 0.6990 - val_loss: 0.4191 - val_accuracy: 0.8791
Epoch 3/15
981/981 [==============================] - 89s 91ms/step - loss: 0.7958 - accuracy: 0.7563 - val_loss: 0.3086 - val_accuracy: 0.9097
Epoch 4/15
981/981 [==============================] - 108s 110ms/step - loss: 0.6883 - accuracy: 0.7902 - val_loss: 0.2912 - val_accuracy: 0.9144
Epoch 5/15
981/981 [==============================] - 105s 107ms/step - loss: 0.6036 - accuracy: 0.8146 - val_loss: 0.2020 - val_accuracy: 0.9403
```

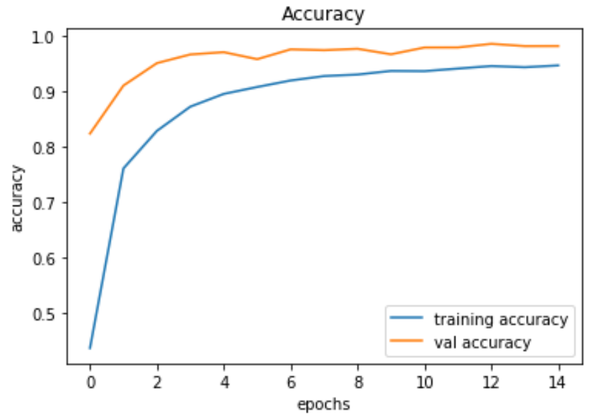
ACCURACY AND LOSS

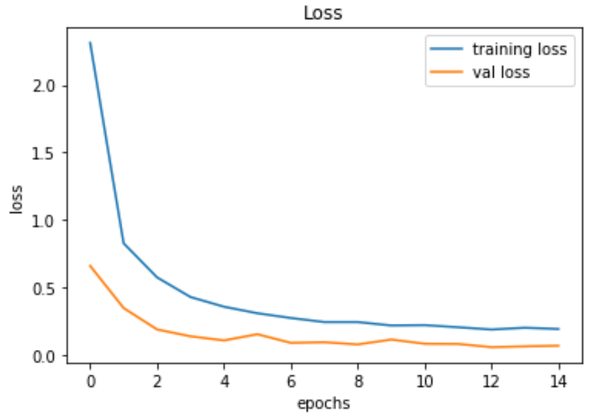
```
[13]: plt.figure(0)
      plt.plot(history.history['accuracy'], label='training accuracy')
      plt.plot(history.history['val_accuracy'], label='val accuracy')
      plt.title('Accuracy')
      plt.xlabel('epochs')
      plt.ylabel('accuracy')
      plt.legend()

      plt.figure(1)
      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()
```

```
[13]: <matplotlib.legend.Legend at 0x24eece89e48>
```

```
[13]: <matplotlib.legend.Legend at 0x24eece89e48>
```





TEST WITH DATASET

Our dataset contains a test folder and in a test.csv file, we have the details related to the image path and their respective class labels. We extract the image path and labels using pandas. Then to predict the model, we have to resize our images to 30×30 pixels and make a numpy array containing all image data. From the sklearn.metrics, we imported the accuracy_score and observed how our model predicted the actual labels. We achieved a 95% accuracy in this model.

[14]:
```
from sklearn.metrics import accuracy_score
import pandas as pd
y_test = pd.read_csv('Test.csv')

labels = 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)

pred = model.predict_classes(X_test)

#Accuracy with the test data
from sklearn.metrics import accuracy_score
accuracy_score(labels, pred)
```

[14]:
```
0.9532066508313539
```

OBSERVATION

RESULTS

PROJECT NAME:- TRAFFIC SIGN RECOGNITION

CONTRIBUTION:-

1. **DHIRAJ AGARWAL - CODING AND DOCUMENTATION**
2. **DAYANAND JAJODIA - CODING AND DOCUMENTATION**
3. **ARNAB SAHA - CODE AND DATA COLLECTION**
4. **HAIDER PARVEZ - CODE AND DATASETS**

CONCLUSION

In this python project, we have successfully classified the traffic sign classifier with 90% accuracy and loss changes with time, which is pretty good from a simple CNN module.

STUDIO SHODWE

Thank You