SUMMER INTERNSHIP REPORT

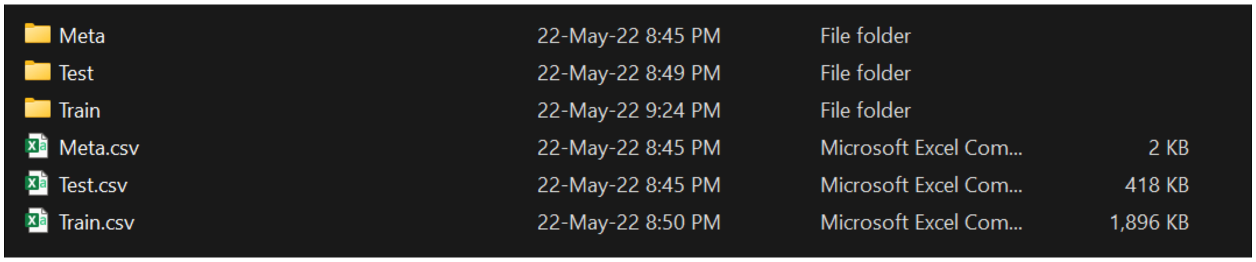
SIGN RECOGNITION USING GTSRB DATASET

Area of Online Internship:	Image Processing
Intern Name:	Vikash Kumar
Name Of Institution:	INDIAN INSTITUTE OF TECHNOLOGY, INDORE
Faculty Mentor Name:	Dr. Vimal Bhatia
Duration:	1 Month (20-05-22 to 24-06-22)
Date of submission:	27-06-22

CONTENTS:

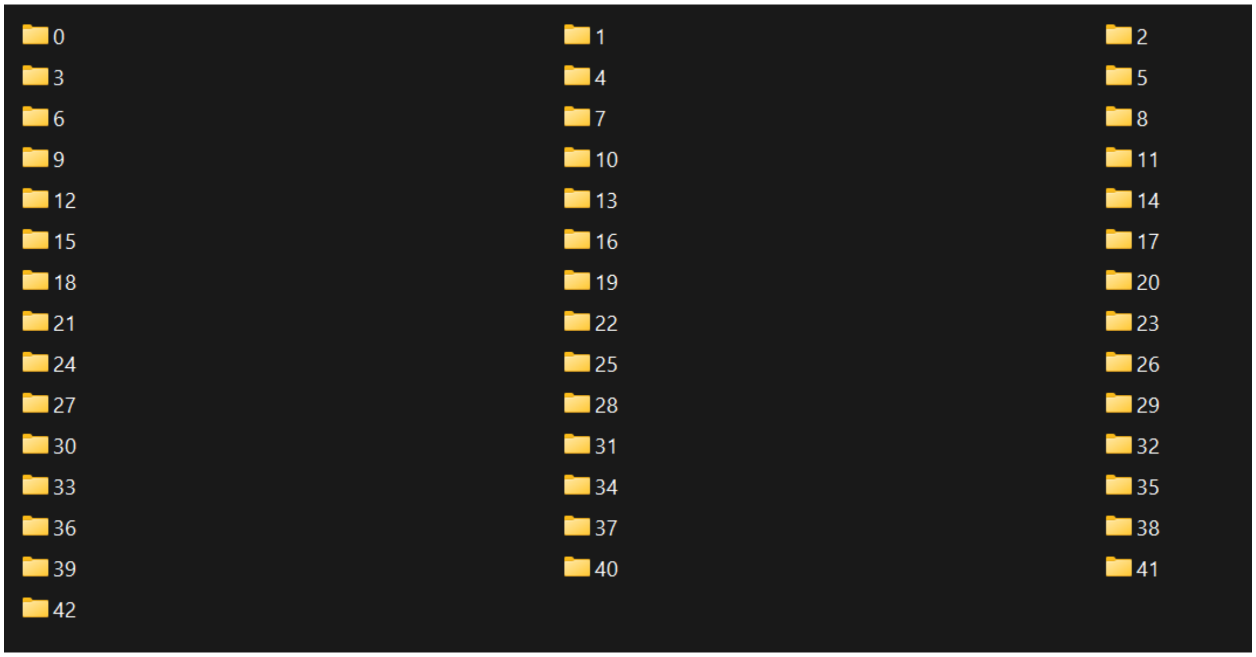
DATA SET: GTSRB – German Traffic Sign Recognition Benchmark

The Data set contains Meta, Training and Testing image directories, and csv file for each directory. The csv file contains the path and category description.



Meta	22-May-22 8:45 PM	File folder	
Test	22-May-22 8:49 PM	File folder	
Train	22-May-22 9:24 PM	File folder	
Meta.csv	22-May-22 8:45 PM	Microsoft Excel Com...	2 KB
Test.csv	22-May-22 8:45 PM	Microsoft Excel Com...	418 KB
Train.csv	22-May-22 8:50 PM	Microsoft Excel Com...	1,896 KB

Training directory contains images of 43 different classes which belong to 43 different types traffic signs.



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		

The test folder contains 12,631 images which can be used to evaluate the trained model.

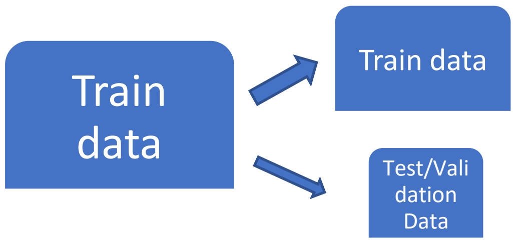
METHODOLOGY:

We will make a CNN model to classify the images. Basically, it is a multiclass classification problem.

We will solve the above problem using TensorFlow, pandas and NumPy.

First, we will extract data from the image data set that needs to be fed into the CNN model. After extracting the image data, we will do preprocessing on them and divide them into 2 set in a ratio of 8:2.

80% percent of images would be used for training and rest 20% would be used for validation.



Then we would make our model.

We would use multiple Convolutional and pooling layers in our model.

After compiling our model, we will start training our model with suitable number of epochs.

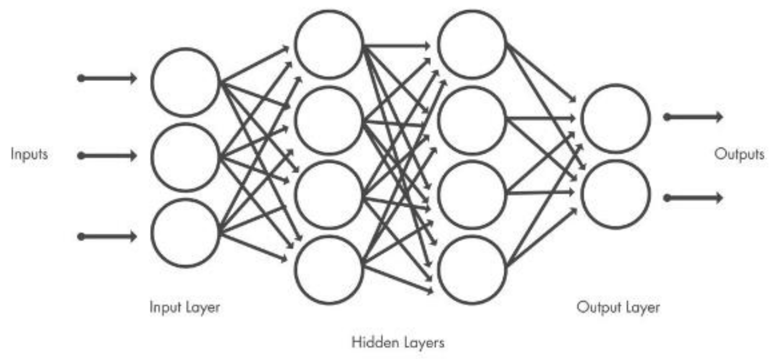
After training we will use the testing data on our trained model to predict results and ultimately find the accuracy of the model and save the model after getting satisfactory results.

CNN- CONVOLUTIONAL NEURAL NETWORK

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.

The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction.

Working:

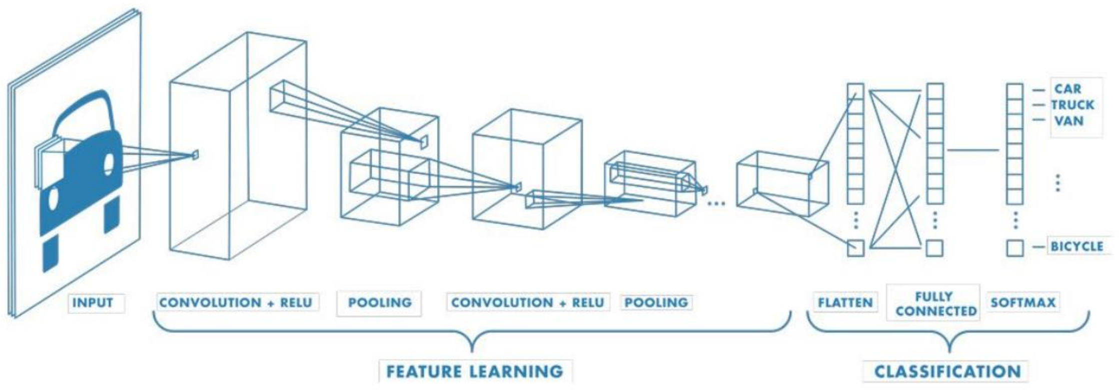


Convolutional Neural Network are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs.
They have 3 main types of layers:

- Convolutional layer
- Pooling Layer
- Fully- Connected (FC) Layer

The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colors and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.

Illustration of CNN in action:



Above diagram includes multiple Convolution and pooling layers, similar to one which we have adopted on the GTSRB dataset and also the above diagram illustrates multi-class classification problem similar to our problem.

MODEL DESCIPTION:

LAYER No.	DESCRIPTION
1.	2D- CONVOLUTIONAL (filters-32, kernel-{5,5}, af-“relu”)
2.	2D- CONVOLUTIONAL (filters-32, kernel-{5,5}, af-“relu”)
3.	MAX – POOLING (pool-{2,2})
4.	DROPOUT (rate-0.25)
5.	2D- CONVOLUTIONAL (filters-64, kernel-{3,3}, af-“relu”)
6.	2D- CONVOLUTIONAL (filters-64, kernel-{3,3}, af-“relu”)
7.	MAX – POOLING (pool-{2,2})
8.	DROPOUT (rate-0.25)
9.	FLATTEN
10.	DENSE (nodes-256, af-“relu”)
11.	DROPOUT (rate – 0.5)
12.	DENSE (nodes-43, af-“softmax”)

af- activation function

TRAINING CODE:

```
# IMPORTING PACKAGES
# Fundamental classes
import numpy as np
import pandas as pd
import tensorflow as tf
import os

# Image related
import cv2
from PIL import Image

# For ploting
import matplotlib.pyplot as plt

# For the model and it's training
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense,Flatten,Dropout


#LOADING IMAGE DATA
# Setting variables for later use
data = []
labels = []
classes = 43
cur_path = os.getcwd()

# Retrieving the images and their labels
for i in range(classes):
    path = os.path.join('../input/gtsrb-german-traffic-sign/','train',str(i))
    images = os.listdir(path)

    for a in images:
        try:
            image = Image.open(path + '/'+ a)
            image = image.resize((30,30))
            image = np.array(image)
            #sim = Image.fromarray(image)
            data.append(image)
            labels.append(i)
        except:
            print("Error loading image")

# Converting lists into numpy arrays
data = np.array(data)
labels = np.array(labels)

# Checking data shape
```

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

# SPLITTING AND PREPROCESSING

# Splitting training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(data, labels,
test_size=0.2, random_state=42)

# Displaying the shape after the split
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)

# Converting the labels into one hot encoding
y_train = to_categorical(y_train, 43)
y_test = to_categorical(y_test, 43)


# BUILDING THE 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))
model.add(Dense(43, activation='softmax'))

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

#Model display
model.summary()

# TRAINING THE MODEL
with tf.device('/GPU:0'):
    epochs = 10
    history = model.fit(X_train, y_train, batch_size=32, epochs=epochs,
validation_data=(X_test, y_test))


# PLOTTING GRAPH ACCURACY V/S EPOCHS AND
      LOSS V/S EPOCHS
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(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()
plt.show()
```

TESTING ACCURACY ON TEST DATASET

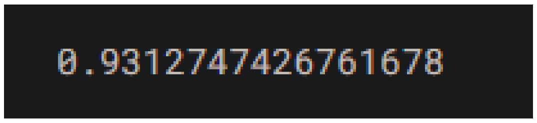
```
from sklearn.metrics import accuracy_score

# Importing the test dataset
y_test = pd.read_csv('../input/gtsrb-german-traffic-sign/Test.csv')

labels = y_test["ClassId"].values
imgs = y_test["Path"].values

data=[]

# Retreiving the images
with tf.device('/GPU:0'):
    for img in imgs:
        image = Image.open('../input/gtsrb-german-traffic-sign/'+img)
        image = image.resize([30, 30])
        data.append(np.array(image))

X_test=np.array(data)

with tf.device('/GPU:0'):
    pred = np.argmax(model.predict(X_test), axis=-1)

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

SAVING MODEL FOR FURTHER USE

```
model.save('traffic_classifier.h5')
```

ACCURACY:

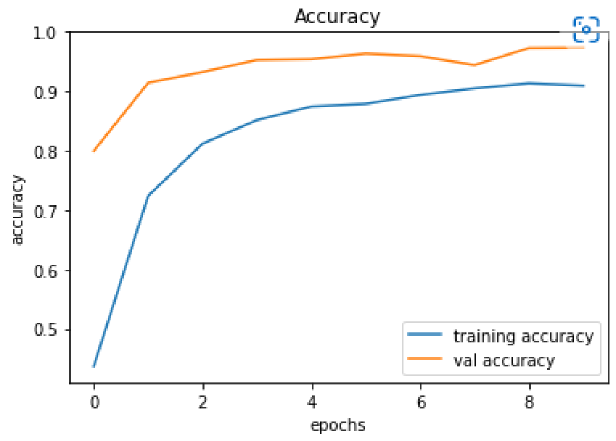
While predicting the test images an accuracy of 93.12% was achieved.



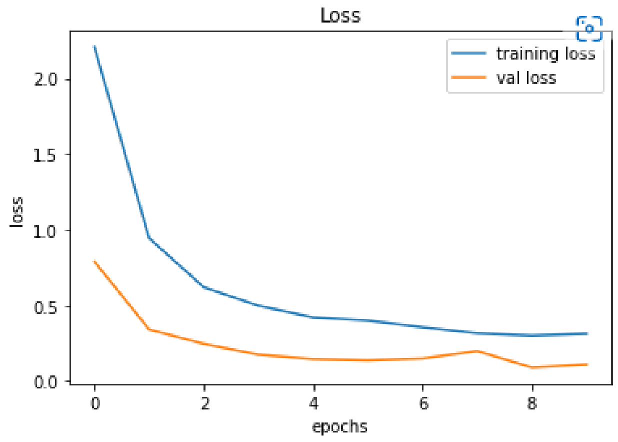
```
0.9312747426761678
```

TRAINING ANALYSIS:

## PLOT - ACCURACY V/S EPOCHS-



## PLOT - LOSS V/S EPOCHS-



SOURCES USED:

1. YouTube
2. KAGGLE DATASET(GTSRB)
3. What are Convolutional Neural Networks? | IBM
4. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way | by Sumit Saha | Towards Data Science
5. What is a Convolutional Neural Network? - MATLAB & Simulink (mathworks.com)