In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
import tensorflow as tf
from PIL import Image
import os
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
```

In [ ]:
```
data = []
labels = []
classes = 43
cur_path = r"D:\Full Stack Data Science AI & ML\ClassNotes\Traffic_sign_Project\arc

#Retrieving the images and their labels
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))
            image = np.array(image)
            #sim = Image.fromarray(image)
            data.append(image)
            labels.append(i)
        except:
            print("Error loading image")
```

In [ ]:
```
#Converting lists into numpy arrays
data = np.array(data)
labels = np.array(labels)

print(data.shape, labels.shape)
#Splitting training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, ra

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)
```

In [ ]:
```
#Building the model
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_tr
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

epochs = 15
history = model.fit(X_train, y_train, batch_size=32, epochs=epochs, validation_data
#model.save("my_model.h5")
```

In [ ]:
```
model.save("my_model.h5")
```

In [ ]:
```
#testing accuracy on test dataset
from sklearn.metrics import accuracy_score

y_test = pd.read_csv(r'D:\Full Stack Data Science AI & ML\ClassNotes\Traffic_sign_P

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

data=[]
for image in imgs:
    print(image)
    try:
        image = Image.open(image)
        image = image.resize((30,30))
        image = np.array(image)
        #sim = Image.fromarray(image)
        data.append(np.array(image))
    except:
        print("Error loading image")
```

In [ ]:
```
try:
            image = Image.open(path + '\\'+ img)
            image = image.resize((30,30))
            image = np.array(image)
            #sim = Image.fromarray(image)
            data.append(np.array(image))
        except:
            print("Error loading image")
```

In [ ]:
```
X_test=np.array(data)

pred = model.predict_generator(X_test)


# Get the predicted probabilities for each class
pred_probabilities = model.predict(X_test)
```

```
# Get the class with the highest probability for each sample
pred = np.argmax(pred_probabilities, axis=1)
```

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