Traffic Signs Recognition

Mingyang Cui

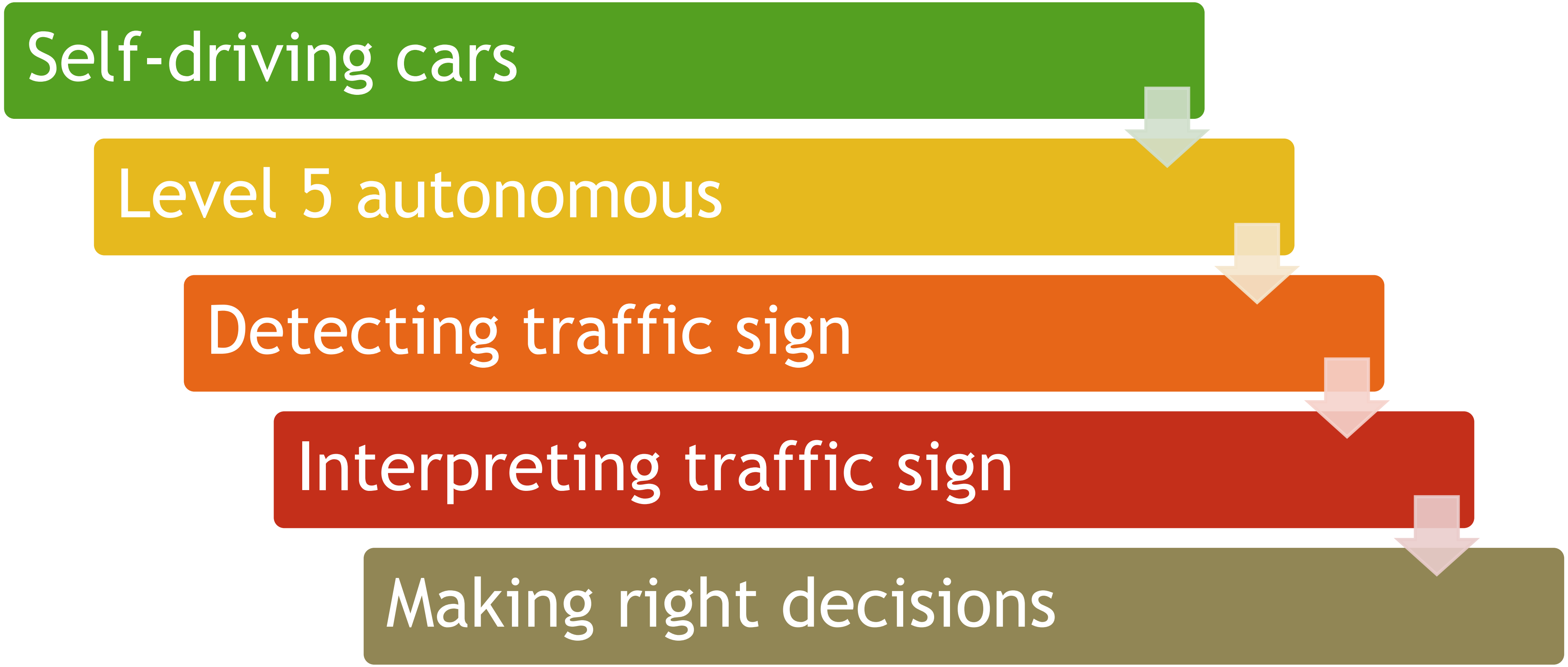
STAT 6289, George Washington University

Background

What is traffic sign recognition?

Why is it important in 2020?

Background



Self-driving cars

Level 5 autonomous

Detecting traffic sign

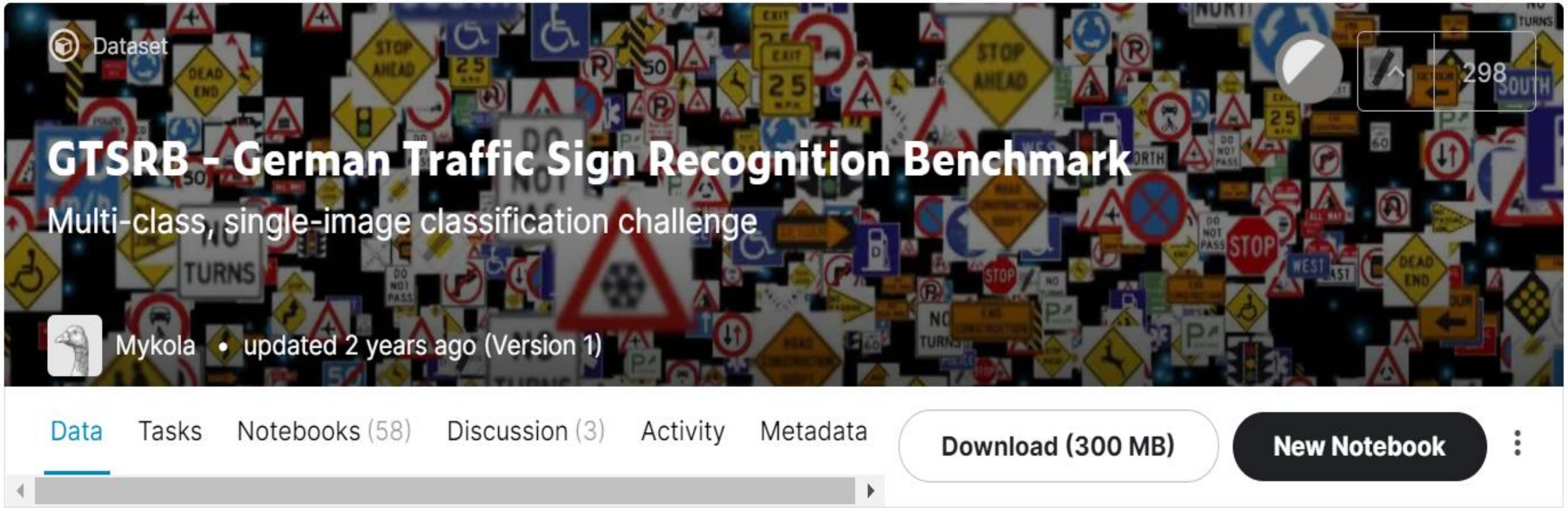
Interpreting traffic sign

Making right decisions

Background

- Build a deep neural net work model
- Classify traffic signs
- Read the traffic signs

Dataset



GTSRB - German Traffic Sign Recognition Benchmark

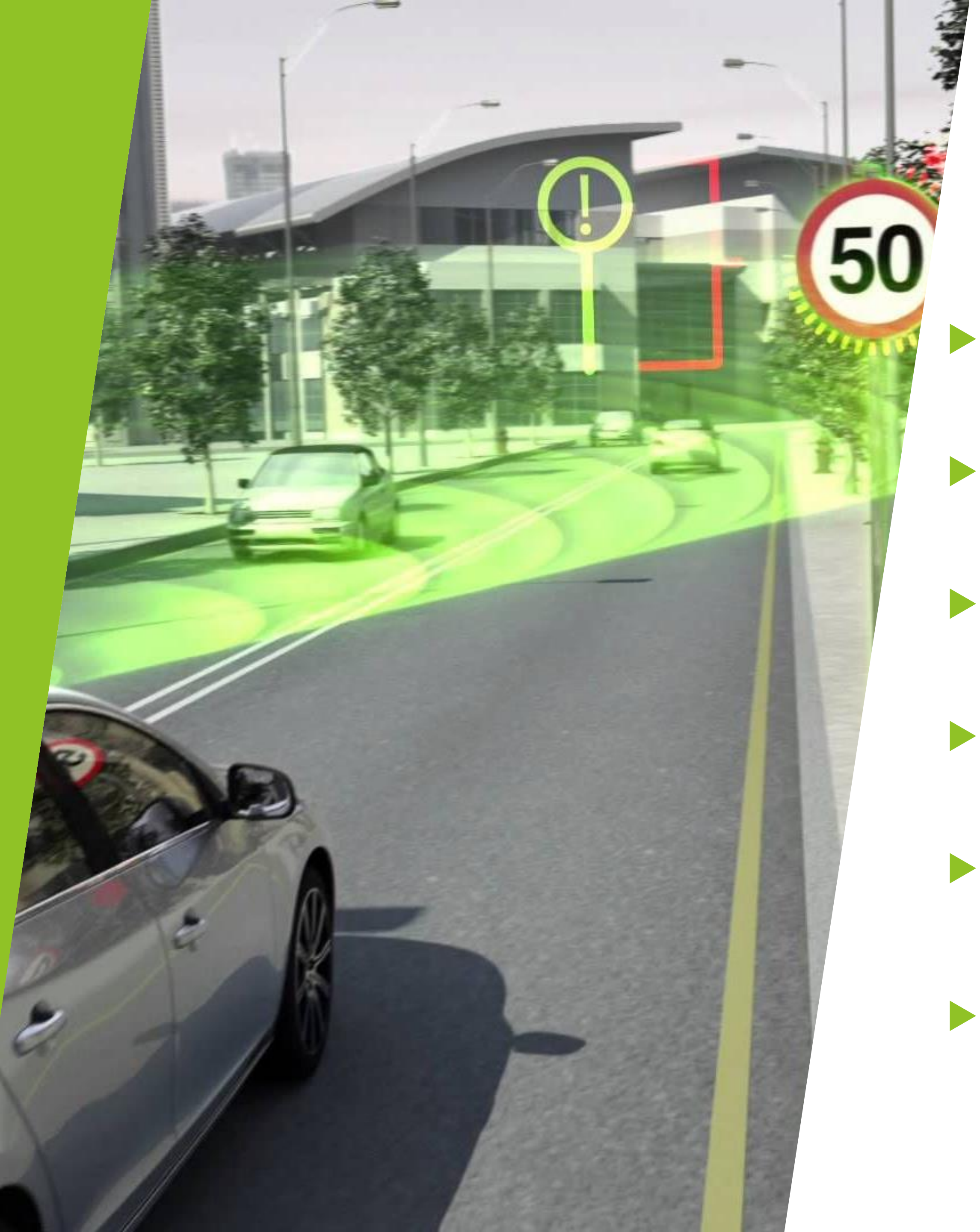
Multi-class, single-image classification challenge

Mykola • updated 2 years ago (Version 1)

- Public dataset available at Kaggle:
- https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign

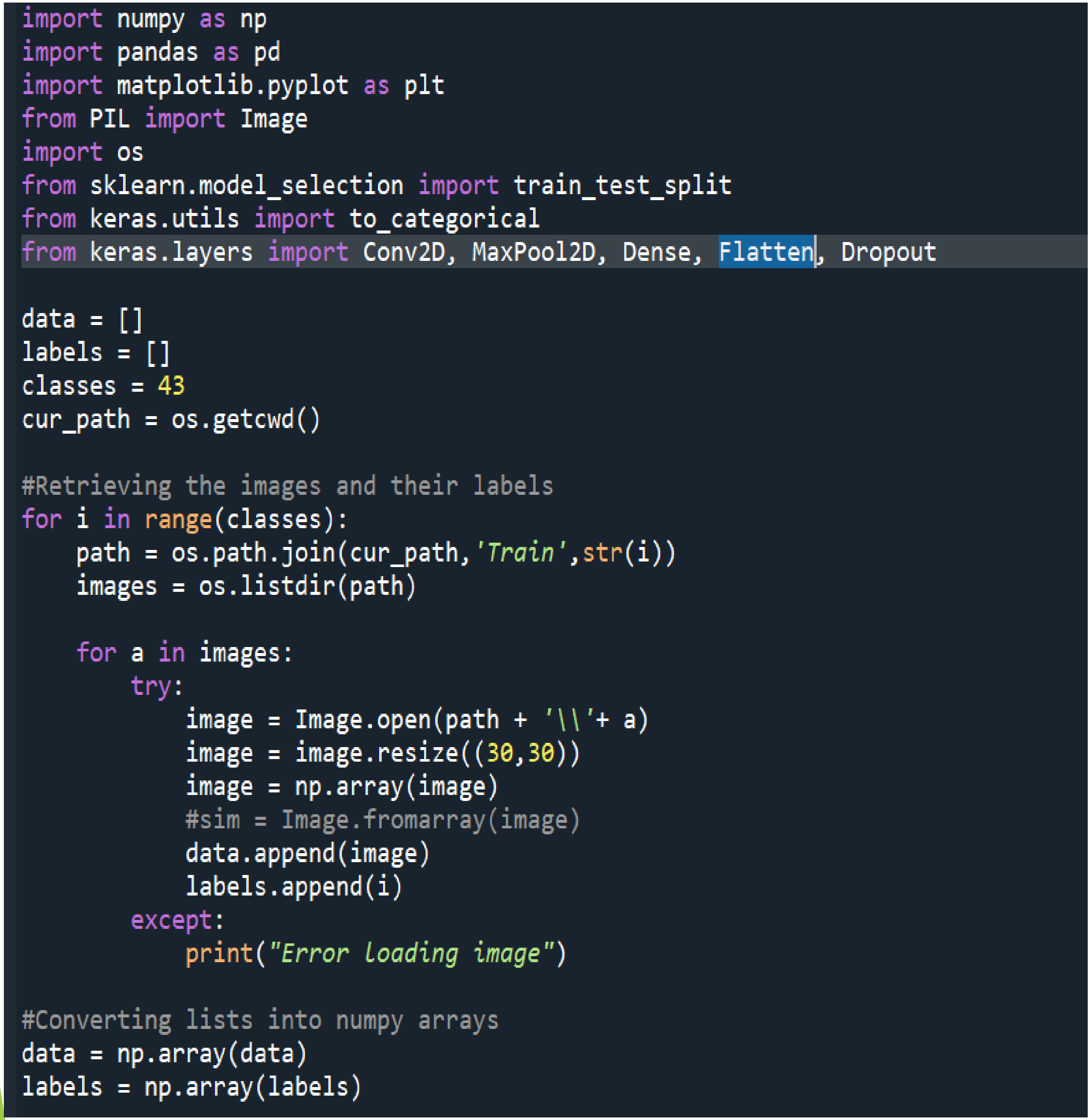
Dataset

- Large, lifelike database
- More than 50,000 images
- 43 different classes
- Single-image, multi-class classification problem
- Held at the International Joint Conference on Neural Networks (IJCNN) 2011



Approach

- Explore the dataset
- Build a CNN model
- Train and validate the model
- Plot the accuracy
- Test the model with test dataset
- Traffic Signs Classifier GUI.py

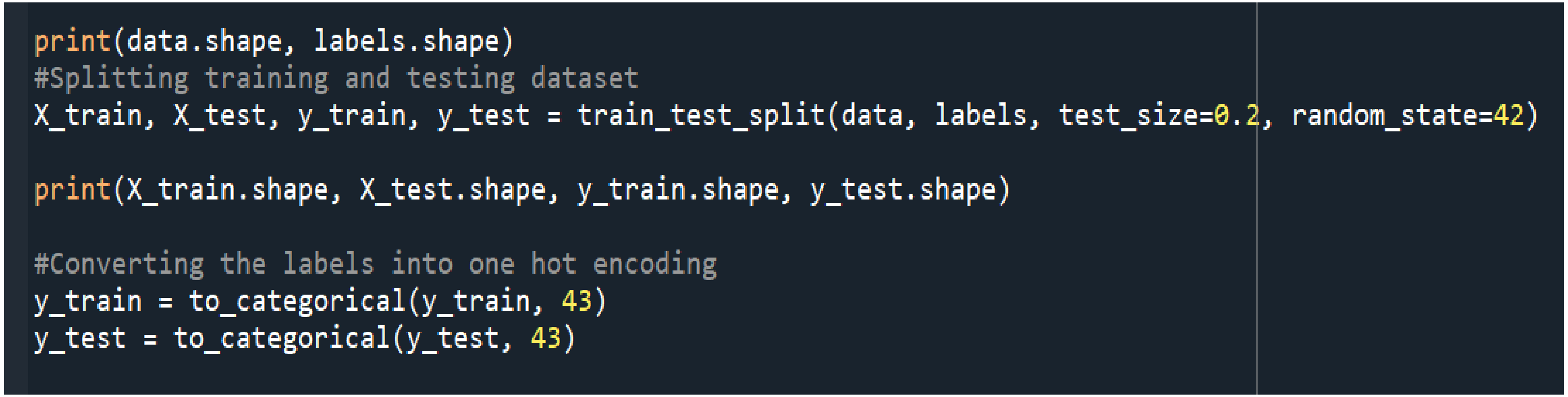


```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
import os
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout

data = []
labels = []
classes = 43
cur_path = os.getcwd()

#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")

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

Explore the Dataset

- Train folders contain 43 folders represents 43 classes
- OS module helps iterate over all the classes
- Append images and their respective labels
- The PIL library helps open image content into an array.

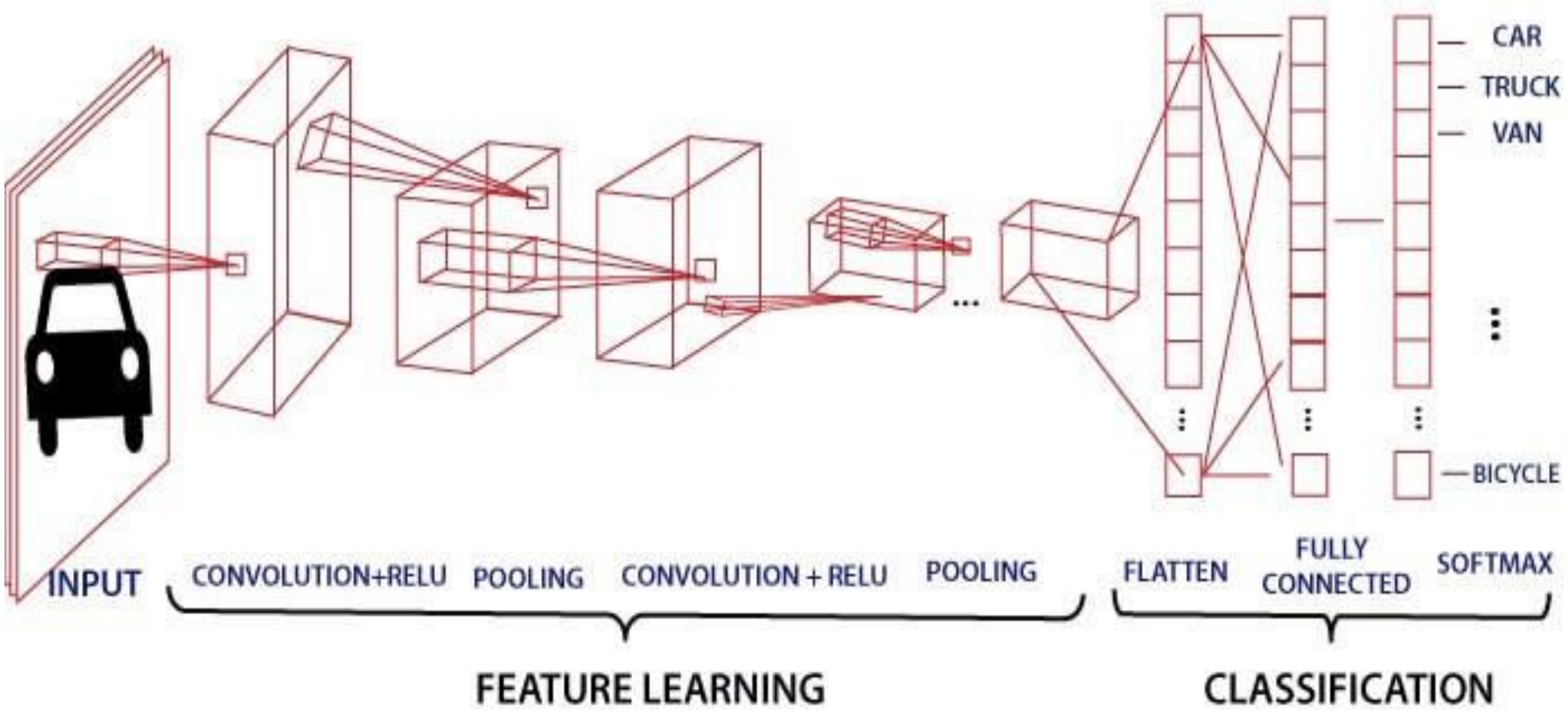
Explore the dataset



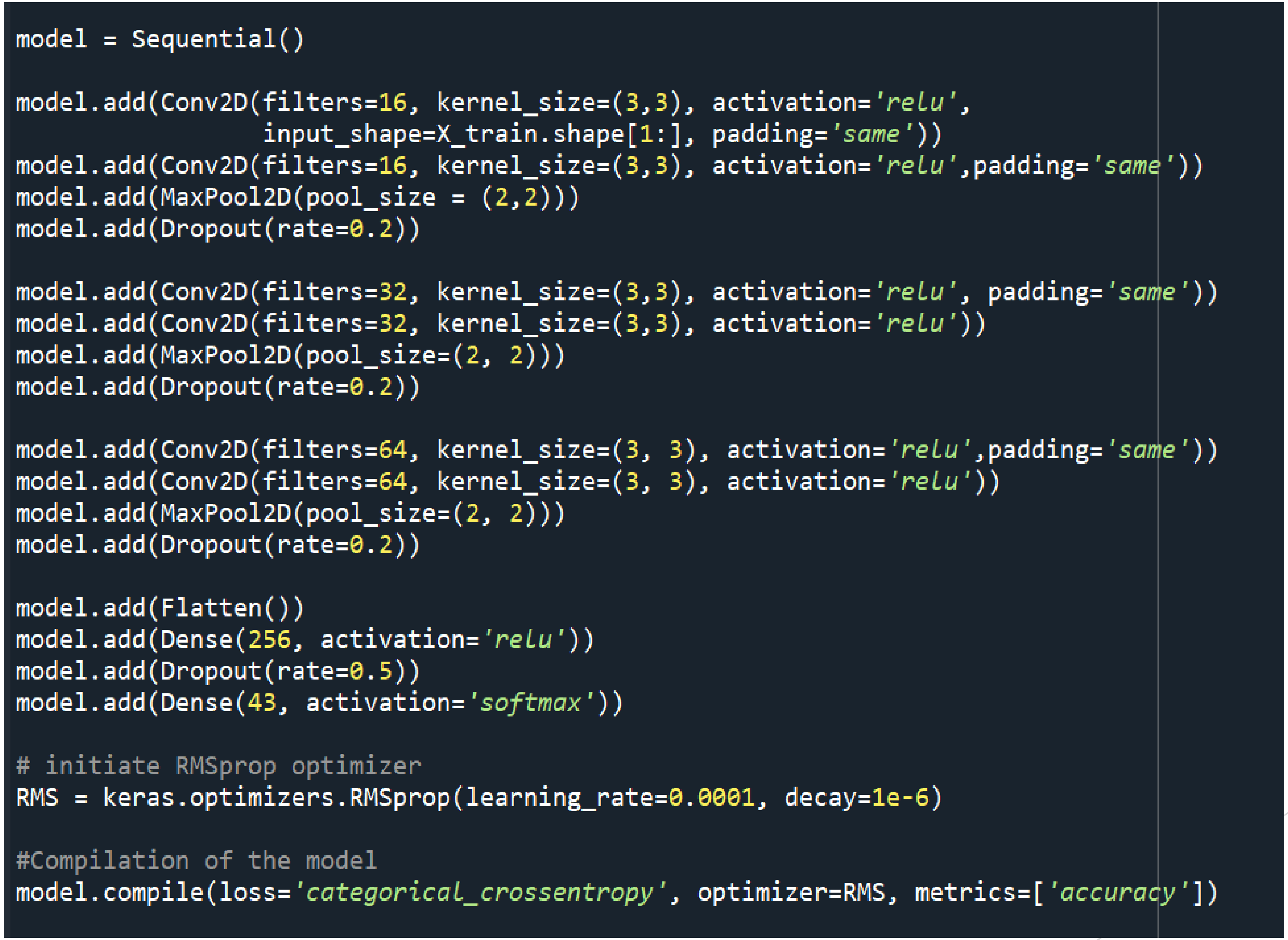
```
print(data.shape, labels.shape)
#Splitting training and testing dataset
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)

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)
```

- **Data.shape = (39209,30,30,3)**
- **There are 39209 images**
- **Size 30 x 30 pixels**
- **Data contain colored images (RGB value)**
- **Split training and testing data.**
- **Convert the labels into one-hot encoding.**

Build CNN model basic Idea



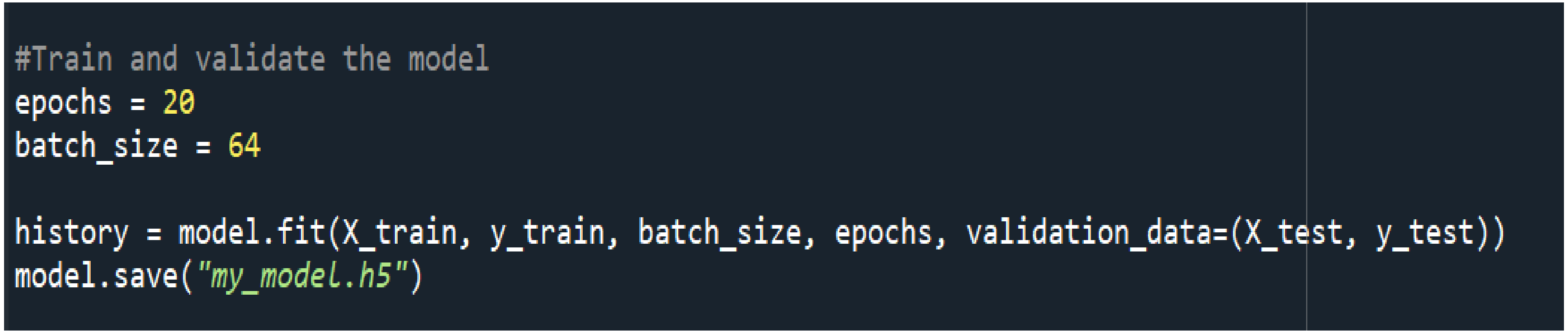
Build CNN model



```
model = Sequential()

model.add(Conv2D(filters=16, kernel_size=(3,3), activation='relu',
                 input_shape=X_train.shape[1:], padding='same'))
model.add(Conv2D(filters=16, kernel_size=(3,3), activation='relu',padding='same'))
model.add(MaxPool2D(pool_size = (2,2)))
model.add(Dropout(rate=0.2))

model.add(Conv2D(filters=32, kernel_size=(3,3), activation='relu', padding='same'))
model.add(Conv2D(filters=32, kernel_size=(3,3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.2))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu',padding='same'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.2))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))

# initiate RMSprop optimizer
RMS = keras.optimizers.RMSprop(learning_rate=0.0001, decay=1e-6)

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

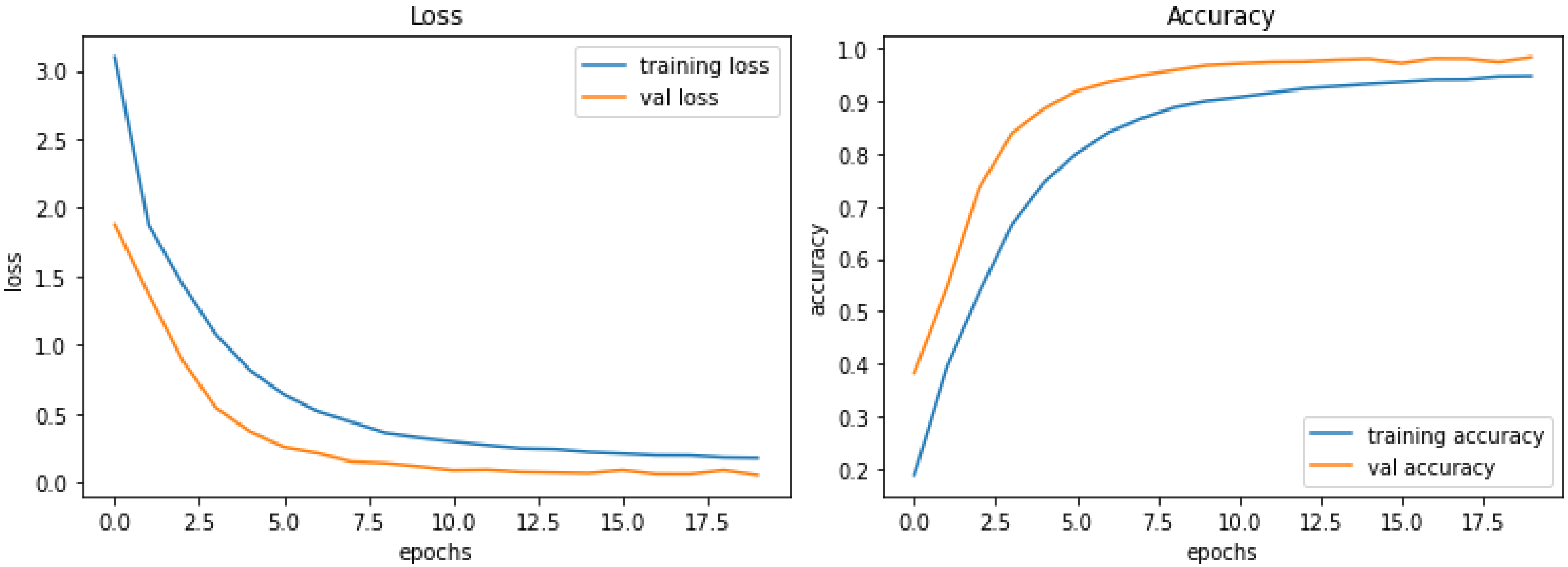
Train and validate the model

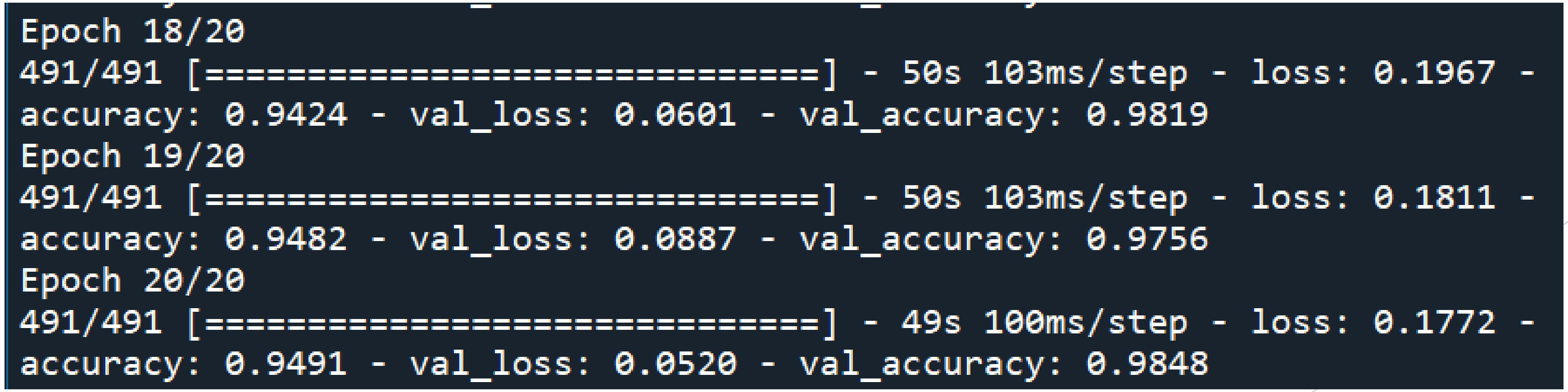
- Train the model using model.fit()
- Set epochs = 20
- Try batch size 32 and 64



```
#Train and validate the model
epochs = 20
batch_size = 64

history = model.fit(X_train, y_train, batch_size, epochs, validation_data=(X_test, y_test))
model.save("my_model.h5")
```

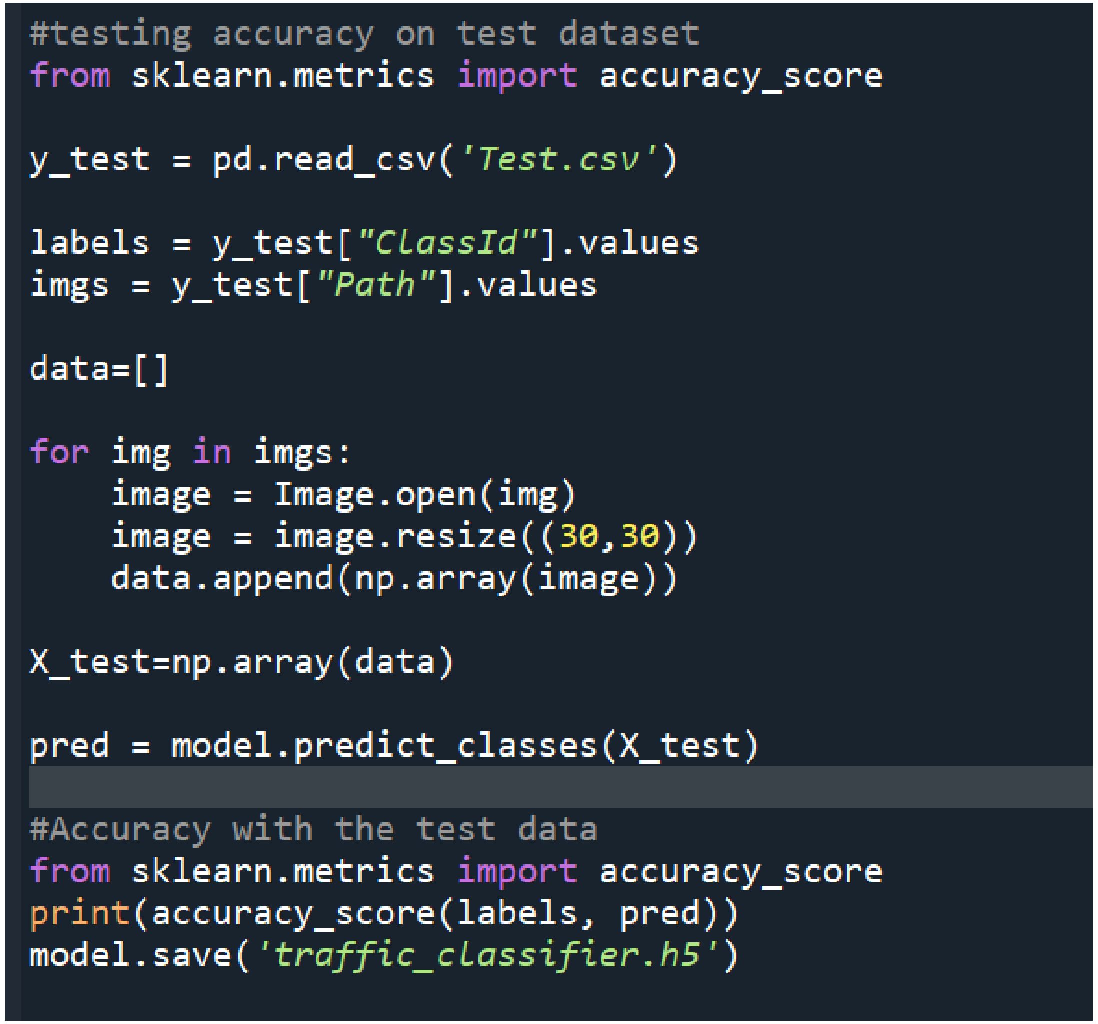
Plot the accuracy



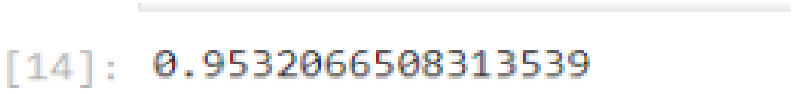


```
Epoch 18/20
491/491 [==============================] - 50s 103ms/step - loss: 0.1967 -
accuracy: 0.9424 - val_loss: 0.0601 - val_accuracy: 0.9819
Epoch 19/20
491/491 [==============================] - 50s 103ms/step - loss: 0.1811 -
accuracy: 0.9482 - val_loss: 0.0887 - val_accuracy: 0.9756
Epoch 20/20
491/491 [==============================] - 49s 100ms/step - loss: 0.1772 -
accuracy: 0.9491 - val_loss: 0.0520 - val_accuracy: 0.9848
```

Test the model



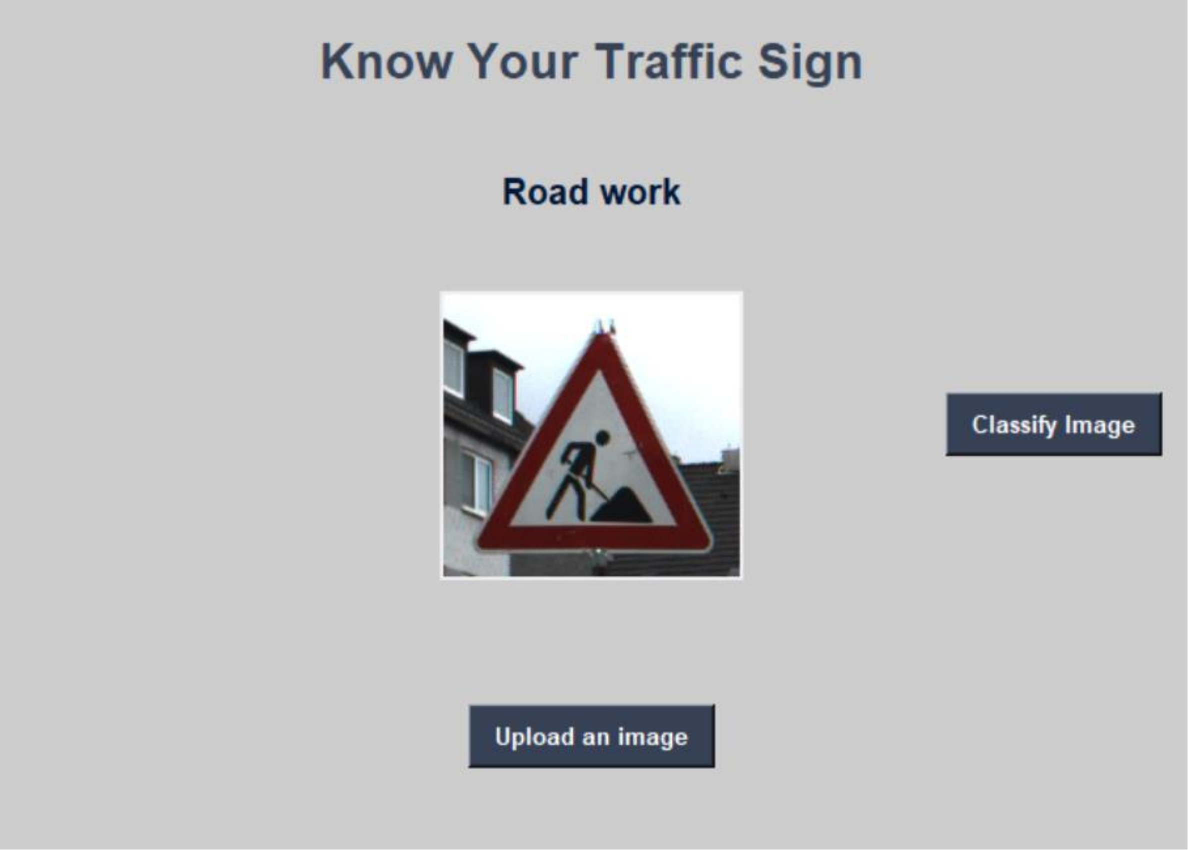
```
#testing accuracy on test dataset
from sklearn.metrics import accuracy_score

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
print(accuracy_score(labels, pred))
model.save('traffic_classifier.h5')
```



```
[14]:  0.9532066508313539
```

- Predict the model
- Resize the image to 30x30
- Make a np.array
- Acchive 95% test accuracy

Traffic Signs Classifier GUI.py



Know Your Traffic Sign

Road work

- A graphical user interface for our traffic signs classifier with Tkinter
- Tkinter is a GUI toolkit in the standard python library
- Call classify()
- load_model('traffic_classifier.h5')
- Image shape (1,30,30,3)

Summary

- In this project, we have successfully classified the traffic signs with 95% accuracy and make plots to visualize how our accuracy and loss changes with time.

Attribution resource:

- https://www.kaggle.com/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign German Traffic Sign Recognition Benchmark, by International Joint Conference on Neural Networks (2011)
- https://towardsdatascience.com/recognizing-traffic-signs-with-over-98-accuracy-using-deep-learning-86737aedc2ab Recognizing Traffic Signs With 98% Accuracy Using Deep Learning, by Eddie Forson (2017)
- https://towardsdatascience.com/traffic-sign-detection-using-convolutional-neural-network-660fb32fe90e Traffic Sign Detection using Convolutional Neural Network, by Sanket Doshi (2019)
- https://data-flair.training/blogs/python-project-traffic-signs-recognition/ Python Project on Traffic Signs Recognition with 95% Accuracy using CNN & Keras, by DataFlair Team (2019)

Thank you!