Traffic Signs Recognition

Deep learning CNN Project

Submitted by:

Sai Kumar M

ACKNOWLEDGMENT

For Traffic Sign Recognition project data is provided by the organizer . Referred to kagal and GitHub for better understanding and to get advanced techniques. Referred Youtube for CNN understanding and took help from these Official websites like w3schools and others for base knowledge.

INTRODUCTION

- ## Business Problem Framing

  You must have heard about self-driving cars in which the passenger can fully depend on the car for traveling. But to achieve level 5 autonomous, it is necessary for vehicles to understand and follow all traffic rules.

- ## Conceptual Background of the Domain Problem

  In the world of Artificial Intelligence and advancement in technologies, many researchers and big companies like Tesla, Uber, Google, Mercedes-Benz, Toyota, Ford, Audi, etc are working on autonomous vehicles and self-driving cars. So, for achieving accuracy in this technology, the vehicles should be able to interpret traffic signs and make decisions accordingly.

- ## Review of Literature

  This is a comprehensive summary of the research done on the Traffic sign recognition. There are several different types of traffic signs like speed limits, no entry, traffic signals, turn left or right, children crossing, no passing of heavy vehicles, etc. Traffic signs classification is the process of identifying which class a traffic sign belongs to. In this project, you have to build a Deep Neural Network model that can classify traffic signs present in the image into different categories. With this model, we should be able to read and understand traffic signs which are a very important task for all autonomous vehicles.

- ## Motivation for the Problem Undertaken

  This model will then be used by big companies like Tesla, Uber, Google,Mercedes-Benz, Toyota, Ford, Audi, etc for their advancement in self-driving cars.

Analytical Problem Framing

- Data Sources and their formats

  Data was provided by the organization, it was a zip file which contains train, test and csv of train and test. The dataset contains more than 50,000 images of different traffic signs. It is further classified into 43 different classes. The dataset is quite varying, some of the classes have many images while some classes have few images. The size of the dataset is around 300 MB. The dataset has a train folder which contains images inside each class and a test folder which you will use for testing your model. The 'train' folder contains 43 folders each representing a different class. The range of the folder is from 0 to 42.

- Data Inputs- Logic- Output Relationships

  CNN model is best for image configuration, Here for this model we will provide images as input and we prepare a model to recognize traffic signs on roads.

- Hardware and Software Requirements and Tools Used

  Hardware

  1. RAM − 128 GB DDR4 2133 MHz
  2. 2 TB Hard Disk (7200 RPM) + 512 GB SSD
  3. GPU − NVidia TitanX Pascal (12 GB VRAM)
  4. Intel Heatsink to keep temperature under control

  Software

  1. Tensorflow
  2. Keras

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

  Algorithms used for the training and testing

  model.fit(X_train, y_train, batch_size=32, epochs=15, validation_data=(X_test, y_test))

- Run and Evaluate selected models

  Snapshots of entire code as shown below

Traffic Sign Recognition

What is Traffic Signs Recognition?

There are several different types of traffic signs like speed limits, no entry, traffic signals, turn left or right, children crossing, no passing of heavy vehicles, etc. Traffic signs classification is the process of identifying which class a traffic sign belongs to.

Aim of the project

We have to build a Deep Neural Network model that can classify traffic signs present in the image into different categories. With this model, we should be able to read and understand traffic signs which are a very important task for all autonomous vehicles.

Importing all necessary libraries

Importing data and preprocessing

Converting image and label lists into numpy arrays

Splitting data and labels into training and testing dataset

CNN Model Building

Compilation of the model

fitting of the model

Importing all necessary libraries

```
from tensorflow.keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout
from tensorflow.keras.models import Sequential, load_model
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
import matplotlib.pyplot as plt
import tensorflow as tf
from PIL import Image
import pandas as pd
import numpy as np
import cv2
import os
```

Importing data and preprocessing

Info on data

The dataset contains more than 50,000 images of different traffic signs.

It is further classified into 43 different classes.

The 'train' folder contains 43 folders each representing a different class. The range of the folder is from 0 to 42.

```
data = []
labels = []
categories = 43
height = 30
width = 30
curr_path = os.getcwd()                # The method os. getcwd() in Python returns the current working directory of a process.

for i in range(categories):
    path = os.path.join(curr_path,'Dataset/train',str(i))       #joins one or more path components
    images = os.listdir(path)                                   #lists files and directories in the given path
    for a in images:
        try:
            image = Image.open(path + '//'+ a)                  #opening other image file
            image = image.resize((height,width))                #resizing the image to maintain uniform
            image = np.array(image)                             #getting array of images using numpy
            data.append(image)                                  #appending images data to data list
            labels.append(i)                                    #appending the labels list
        except:
            print("Error loading image")
```

Converting image and label lists into numpy arrays

We need to convert the data and label list into numpy arrays for feeding to the model.

```
data = np.array(data)                       #data list into numpy array
labels = np.array(labels)                   #label list into numpy array
print(data.shape, labels.shape)             #shape of data and label
```

```
(39209, 30, 30, 3) (39209,)
```

The shape of data is (39209, 30, 30, 3)

which means that there are 39209 images

Splitting data and labels into training and testing dataset

```
X_train, X_test, y_train, y_test = train_test_split(data,          #train_test_split() method to split training and testing data
                                                    labels,
                                                    test_size=0.2,
                                                    random_state=30)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)    # shape of training and testing
```

```
(31367, 30, 30, 3) (7842, 30, 30, 3) (31367,) (7842,)
```

test size is 0.2 i,e 20 % data

For training

The shape of data is (31367, 30, 30, 3)

which means that there are 31367 images

image size 30×30 pixels

The last 3 defines the data contains colored images i,e (RGB value).

for testing

The shape of data is (7842, 30, 30, 3)

which means that there are 7842 images

image size 30×30 pixels

The last 3 defines the data contains colored images i,e (RGB value).

```
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=32, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.3))
model.add(Flatten())
model.add(Dense(190, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))
```

Compilation of the model

Compile defines the loss function, the optimizer and the metrics

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

Fitting of a model

Trains the model for a fixed number of epochs

```
model.fit(X_train, y_train, batch_size=32,
          epochs=15,
          validation_data=(X_test, y_test))
```

Trains the model for a fixed number of epochs

```
model.fit(X_train, y_train, batch_size=32,
          epochs=15,
          validation_data=(X_test, y_test))
```

```
Epoch 1/15
981/981 [==============================] - 94s 95ms/step - loss: 3.7343 - accuracy: 0.2080 - val_loss: 1.4752 - val_accuracy:
0.5925
Epoch 2/15
981/981 [==============================] - 89s 91ms/step - loss: 1.5402 - accuracy: 0.5169 - val_loss: 0.7089 - val_accuracy:
0.8045
Epoch 3/15
981/981 [==============================] - 90s 92ms/step - loss: 0.9317 - accuracy: 0.7135 - val_loss: 0.2731 - val_accuracy:
0.9227
Epoch 4/15
981/981 [==============================] - 90s 92ms/step - loss: 0.6147 - accuracy: 0.8157 - val_loss: 0.2000 - val_accuracy:
0.9454
Epoch 5/15
981/981 [==============================] - 91s 92ms/step - loss: 0.4929 - accuracy: 0.8503 - val_loss: 0.1549 - val_accuracy:
0.9601
Epoch 6/15
981/981 [==============================] - 95s 97ms/step - loss: 0.4246 - accuracy: 0.8739 - val_loss: 0.1807 - val_accuracy:
0.9526
Epoch 7/15
981/981 [==============================] - 91s 92ms/step - loss: 0.4085 - accuracy: 0.8793 - val_loss: 0.1050 - val_accuracy:
0.9730
Epoch 8/15
981/981 [==============================] - 92s 93ms/step - loss: 0.3586 - accuracy: 0.8931 - val_loss: 0.1322 - val_accuracy:
0.9637
```

- Key Metrics for success in solving problem under consideration

  Accuracy metrics used in the compilation of the model.

Compilation of the model

Compile defines the loss function, the optimizer and the metrics

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

CONCLUSION

- Key Findings and Conclusions of the Study

  With the provided dataset we were able to build a model without any hurdles. Data provided by organizations is pretty huge. In Data Science the weightage of the data plays a huge role in the building model.

  Finally we are able to gain an accuracy of 93% in this traffic sign recognition CNN model.

- Learning Outcomes of the Study in respect of Data Science

  Learning Outcomes of the Study in respect of Data Science are we got practical experience in deep learning convolutional Neural Network. We were able to use new functions like getcwd(), PTL library and to_category() function which helps us in simplifying our project and saves our time , and able to solve minor problems with new techniques which came in the way of modelling.

- Limitations of this work and Scope for Future Work

  What are the limitations of this solution provided, the future scope?

  Preprocessing data was a bit of a complex task in this project. Training the model was very time consuming.

  We can extend this project and use it in a new direction such as detecting fuel stations, stay, restaurants, service stations and so on.