Graphic Era (Deemed to be University)

Dehradun

Mini Project Report

2021 - 2022

—

Project Topic

Smart Traffic Management System Using Artificial Intelligence

---

Submitted To:

Dr. Sachin Sharma
Assistant Professor
Dept. of Computer Science & Eng.
GEU, Dehradun.

Submitted By:

Ankit Kumar
Undergraduate Student
Dept. of Computer Science & Eng.
GEU, Dehradun
University Roll No: 2014567
Enrollment No: GE-192014567
B.Tech. (CSE), 5th Semester
Section: C, Class Roll No: 7

Content

1.Introduction

1.1. Artificial Intelligence (AI)

the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. The term is frequently applied to the project of developing systems endowed with the intellectual processes characteristic of humans, such as the ability to reason, discover meaning,

1.2. Reason For Traffic congestion

The main reason for traffic congestion is the fact that every single person wants to move at the same time every single day. This is because the school and economy system expect that people go to school and work at the same time. Traffic congestion and long travel times are undesirable because they discourage future economic growth, increase vehicular emissions, increase fuel expenses, increase operating costs for both private and freight vehicles, decrease economies of agglomeration, heighten the psychological and **these reasons call's us for a proper traffic management system**.

2. Traffic Management System

Traffic Management refers to the combination of measures that serve to preserve traffic capacity and improve the security, safety and reliability of the overall road transport system.

Some of the basic measures that we all have been using for so long are:

- The use of public transportation
- Carpooling
- Avoiding these peak hours
- Implementation of some of the existing traffic rules
- Following of some of the road signs

Even after all these management techniques and basic mesures we still can't conclude that we are free from Traffic congestion. Its because a human is commited to create mistakes, regardless of any traffic rules and road signs.

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.

3. Traffic Signs Recognition

Need : 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.

Advantage : Traffic Sign Recognition ensures that the current speed limit and other road signs are displayed to the driver on an ongoing basis. Automatic recognition functions through a link between images captured by a camera and the speed limit information stored in the navigation system.

4. About the Project

The main objective of our project is to design and construct a computer based system which can automatically detect the road signs so as to provide assistance to the user or the machine so that they can take appropriate actions. The proposed approach consists of building a deep neural network model that can classify traffic signs present in the image into different categories. With this model, we are able to read and understand traffic signs which are a very important task for all autonomous vehicles.

Our approach to building this traffic sign classification model is discussed in four steps:

1. Explore the dataset
2. Build a CNN model
3. Train and validate the model
4. Test the model with test dataset

4.1. Background and related work

- Many different techniques have been applied to detect traffic signs. Most of these techniques are based on using HOG and SIFT features.
- In our approach we use biologically inspired convolutional neural networks to build a model which can predict the type of traffic sign.

4.2. The Dataset of the Project

The dataset we have used for this project is the GTSRB (German traffic sign recognition benchmark) public dataset available at Kaggle.

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 that has traffic sign images in 43 different classes, a Test folder that has over 12,000 images for testing purposes. A test.csv file that contains the path of the test images along with their respective classes.

A sample of the dataset can be seen in **Figure 2** above — notice how the traffic signs have been *pre-cropped* for us, implying that the dataset annotators/creators have manually labeled the signs in the images *and* extracted the traffic sign Region of Interest (ROI) for us, thereby simplifying the project.

In the real-world, traffic sign recognition is a two-stage process:

- **Localization:** Detect and localize where in an input image/frame a traffic sign is.
- **Recognition:** Take the localized ROI and actually *recognize* and *classify* the traffic sign.

Deep learning object detectors can perform localization and recognition in a single forward-pass of the network

4.3. Explore the dataset

we iterate over all the classes and append images and their respective labels in the data and labels list.

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

With the sklearn package, we use the train_test_split() method to split training and testing data.

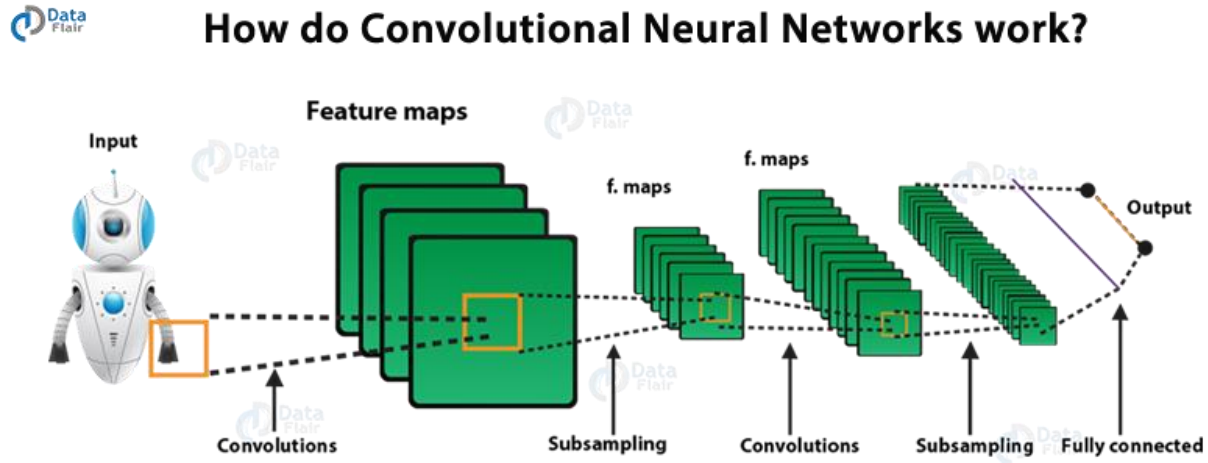
From the keras.utils package, we use to_categorical method to convert the labels present in y_train and t_test into one-hot encoding.

```
[10]: print(data.shape, labels.shape)
      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)

      y_train = to_categorical(y_train, 43)
      y_test = to_categorical(y_test, 43)
```

```
(39209, 30, 30, 3) (39209,)
(31367, 30, 30, 3) (7842, 30, 30, 3) (31367,) (7842,)
```

4.4. Build a CNN model

To classify the images into their respective categories, we will build a CNN model CNN is best for image classification purposes therefore, are mostly in the field of computer vision where they are used for classifying images, segmenting them and also performing localization on the images.



We compile the model with Adam optimizer which performs well and loss is “categorical_crossentropy” because we have multiple classes to categorise.

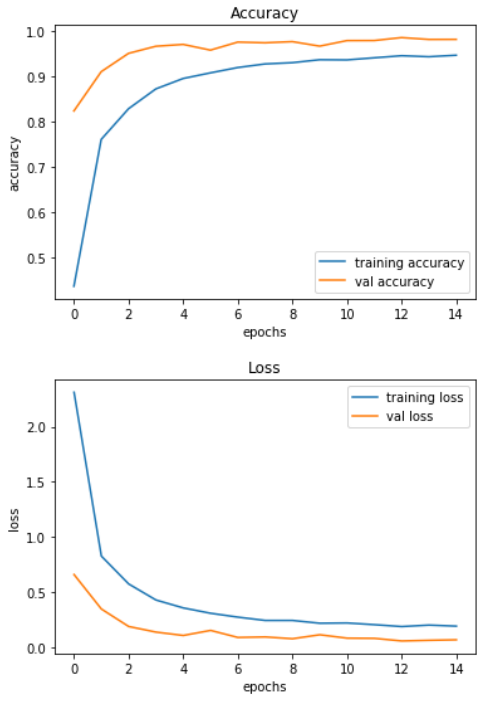
```
[11]: model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu', input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))
model.add(Conv2D(filters=64, kernel_size=(3, 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'])
```

4.5. Train and validate the model

After building the model architecture, we then train the model using model.fit(). After trying with batch size 32 and 64. Our model performed better with 64 batch size. And after 15 epochs the accuracy was stable.Therefore limited the trainig to 15 epochs only.

Our model got a 95% accuracy on the training dataset. With matplotlib, we plot the graph for accuracy and the loss.

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



4.6. Testing model with test 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
```

5. Traffic Signs Classifier GUI

Tkinter is a GUI toolkit in the standard python library. Make a new file in the project folder and copy the below code. Save it as gui.py and you can run the code by typing python gui.py in the command line.

In this file, we have first loaded the trained model 'traffic_classifier.h5' using Keras. And then we build the GUI for uploading the image and a button is used to classify which calls the classify() function. The classify() function is converting the image into the dimension of shape (1, 30, 30, 3). This is because to predict the traffic sign we have to provide the same dimension we have used when building the model. Then we predict the class, the model.predict_classes(image) returns us a number between (0-42)

6. Conclusion

In this Python project with source code, we have successfully classified the traffic signs classifier with 95% accuracy and also visualized how our accuracy and loss changes with time, which is pretty good from a simple CNN model

To create our traffic sign classifier, we:

- Utilized the popular German Traffic Sign Recognition Benchmark **(GTSRB)** as our dataset.
- Implemented a Convolutional Neural Network called TrafficSignNet using the Keras deep learning library.
- Trained TrafficSignNet on the GTSRB dataset, obtaining **95% accuracy**.
- Created a Python script GUI that loads our trained TrafficSignNet model and then classifies new input images.