VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering J-component

Report on

TRAFFIC SIGN DETECTION SYSTEM

Big Data Frameworks (CSE 3120)

Konki Mohit -20MIA1108

konki.mohit2020@vitstudent.ac.in

Abstract
Self driving cars in the automobile industry have evolved through the years. In this paper we propose a system with real-time detection and classification of traffic signs using CNN, a computer vision method. Convolutional neural networks are used in this system to extract features from images and then classify them into different types of traffic sign classifications. The system is capable of detecting and identifying a variety of traffic signs.
The project involves preprocessing the images, training a CNN model using Keras, and testing the accuracy of the model on a separate test dataset. The results of this project show that the developed system can accurately recognize traffic signs with an accuracy of over 98%.

Key Words: Traffic sign detection, convolutional neural network (CNN), Keras, Deep learning, Accuracy

1. Introduction:

Traffic Signs are an important part of driving. There are various signs like warning signs of curve or hair pin bend. Which will play a vital role in avoiding accidents and warning the person driving the car of what's ahead. Traffic signs indicate the speed limit and when to stop, have other functions. Temporary signs warn that the route might change or that the driver should be alert for workers, like the signs in construction zones. Signs for winding roads and ice on bridges are crucial as well because often the drivers wouldn't notice these things until they are directly above them. The most obvious advantage of traffic signs is driver safety. The roadways would be chaotic and unsafe without them. Drivers wouldn't know when to stop, whether to yield, or how fast to go. They wouldn't be informed of impending difficulties. Additionally, the absence of signage would allow careless drivers carte blanche to act whatever they pleased, endangering both themselves and other motorists. Rules are made plain by traffic signs, which also keep drivers safe.

As technology is increasing, many self-driving cars came into the picture. Many companies like Google, Tesla, Uber, Ford, Audi, Toyota, Mercedes-Benz are working on automating vehicles. For a human driving cars, seeing and understanding the signs is easy but for an AI it should be trained to read these signs and interpret them. This field of study is called computer vision. Computer vision helps the AI to derive meaningful information from digital images, videos and other visual inputs and take actions or make recommendations based on that information. Computer vision algorithms can be trained to recognize specific patterns and shapes that are commonly found in traffic signs, such as octagons for stop signs, circles for speed limit signs, and triangles for warning signs. These algorithms can also be trained to detect the specific colors and symbols used in different types of signs. It can analyze from cameras mounted on vehicles or on the side of the road, and quickly identify any traffic signs in the field of view. This information can be used to trigger automated responses, such as adjusting the speed of a self-driving car.

2. Background Study:

1) Road traffic sign detection and classification by A. de la Escalera; L.E. Moreno; M.A. Salichs; J.M. Armingol, The paper proposed a system for road traffic sign detection and classification using convolutional neural network (CNN). In the image preprocessing module, the authors applied histogram equalization and contrast stretching to enhance the contrast of the images. In the traffic sign detection module, they used a combination of color and shape features to detect candidate regions of interest (ROIs) that potentially contain traffic signs. Finally, in the traffic sign classification module, they used a convolutional neural network (CNN) to classify the ROIs into the corresponding traffic sign class.. They evaluated their system on the German Traffic Sign Recognition Benchmark (GTSRB) dataset and reported a recognition rate of 98.4% on the test set. The proposed system outperformed other state-of-the-art systems in terms of recognition rate. This system has the potential to improve road safety by providing accurate and reliable traffic sign recognition.

2) "Real-Time Traffic Sign Detection and Recognition Using Convolutional Neural Networks" by B. G. Kim et al. The method involves first detecting candidate regions using a region proposal network, and then classifying these regions using a CNN trained on traffic sign images.They used a combination of the GTSRB and KITTI datasets for training and testing their system. They evaluated their system using accuracy and speed metrics and reported a recognition rate of 99.5%. Their system used a CNN for traffic sign detection and recognition.

3) "Traffic Sign Recognition with Multi-Scale Convolutional Neural Networks" by S. Zhang et al. The method involves using a CNN with multiple scales to extract features from the input image, and then classifying these features using a softmax classifier. The authors also use data augmentation techniques to improve the robustness of the model. They used the GTSRB dataset for training and testing their system. They evaluated their system using accuracy, recall, and F1-score metrics and reported an F1-score of 0.968. Their system used a multi-scale CNN for traffic sign detection and recognition.

4) "Traffic Sign Detection and Recognition Using Deep Learning Based Object Detection Technique" by S. K. Patel et al. The paper proposed a YOLOv3 object detection algorithm. The method involves preprocessing the input images, training a deep neural network model using a dataset of traffic sign images, and using the trained model to detect and recognize traffic signs in real-time. The object detection technique used in this method is the Faster R-CNN (Region-based Convolutional Neural Network) algorithm, which allows the model to detect and classify multiple objects in an image simultaneously. It was implemented on the Belgian Traffic Sign Dataset for training and testing their system. They evaluated their system using precision, recall, and F1-score metrics and reported an F1-score of 0.949. Their system used the YOLOv3 object detection algorithm for traffic sign detection and recognition.

5) "Real-time traffic sign detection and recognition system using an enhanced deep convolutional neural network" by S. E. Zohdy, R. S. El-Desouky, and A. E. Hassanien
This paper proposed a real-time traffic sign detection and recognition system using an enhanced deep convolutional neural network (ED-CNN). The proposed CNN model uses a combination of convolutional, pooling, and fully connected layers to learn and classify traffic signs in real-time. The authors also use data augmentation techniques to improve the robustness of the model to different lighting conditions and occlusion. They used the German Traffic Sign Recognition Benchmark (GTSRB) dataset and achieved a recognition rate of 99.23% on the test set. They used a combination of color and shape features and an EDCNN for traffic sign detection and classification.

3. Dataset:

The German Traffic Sign Recognition Benchmark(GTSRB) dataset contains over 50,000 images of traffic signs, with 43 different classes of signs. Which are split into 39,209 training images and 12,630 test images. The images have varying light conditions and rich backgrounds. The images are in color and have a resolution of 32x32 pixels. The dataset includes images of traffic signs in different lighting conditions, weather conditions, and with occlusions. Each image in the dataset is labeled with the correct class of traffic sign.
The dataset was collected from real-world images of traffic signs in Germany, making it representative of real-world conditions. The GTSRB dataset has become a standard benchmark for evaluating the performance of traffic sign recognition systems.

4. Methodology:

The GTSRB dataset is preprocessed to ensure that all images are properly labeled and that there are no missing or duplicate images. Also, perform data augmentation techniques such as image rotation, scaling, and flipping to increase the size of the dataset and to provide more variety to the CNN during training. A sequential CNN model of 10 layers including

convolution layer, max pooling layer, and dense layer with ReLU activation function and softmax function in output layer. The Adam optimization algorithm and categorical cross entropy loss function is used. The CNN model takes images from the dataset as input, derives its features and uses them to predict. The CNN model is trained with the preprocessed train dataset. During training the accuracy is monitored on the validation set and the hyperparameters are adjusted accordingly. The model is then tested on the test dataset to evaluate its performance on traffic sign detection. Also, a quantitative analysis is performed to calculate the CNN's performance; metrics such as precision, recall, and F1-score are used.

5. Results & Discussion:

Assigning path for dataset

```
[ ]  data_dir = "/content/drive/MyDrive/Colab_Notebook/TSF/Traffic_sign_classification"
     train_path = 'Train'
     test_path = 'Test'

     # Resizing the images to 30x30x3
     IMG_HEIGHT = 30
     IMG_WIDTH = 30
     channels = 3
```

Finding total classes

```
NUM_CATEGORIES = len(os.listdir(train_path))
NUM_CATEGORIES
```

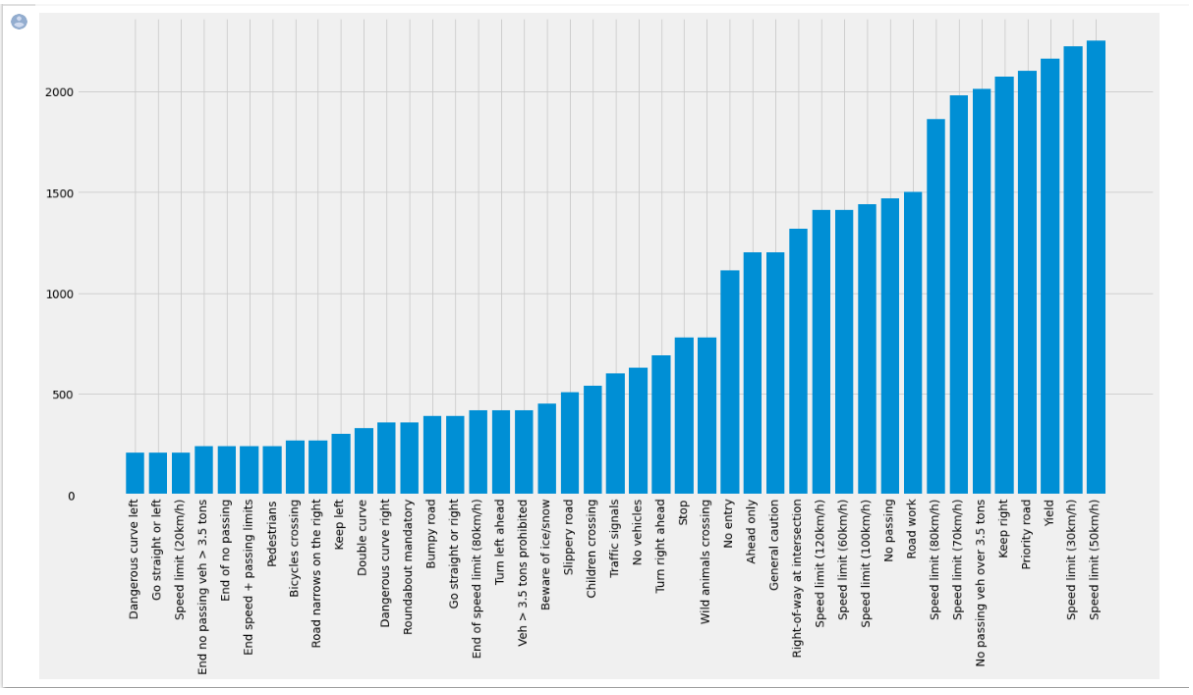
```
43
```

All labels are given 0-42

```
classes = { 0:'Speed limit (20km/h)',
            1:'Speed limit (30km/h)',
            2:'Speed limit (50km/h)',
            3:'Speed limit (60km/h)',
            4:'Speed limit (70km/h)',
            5:'Speed limit (80km/h)',
            6:'End of speed limit (80km/h)',
            7:'Speed limit (100km/h)',
            8:'Speed limit (120km/h)',
            9:'No passing',
            10:'No passing veh over 3.5 tons',
            11:'Right-of-way at intersection',
            12:'Priority road',
            13:'Yield',
            14:'Stop',
            15:'No vehicles',
            16:'Veh > 3.5 tons prohibited',
            17:'No entry',
            18:'General caution',
            19:'Dangerous curve left',
            20:'Dangerous curve right',
            21:'Double curve',
            22:'Bumpy road',
            23:'Slippery road',
            24:'Road narrows on the right',
            25:'Road work',
            26:'Traffic signals',
            27:'Pedestrians',
            28:'Children crossing',
            29:'Bicycles crossing',
            30:'Beware of ice/snow',
            31:'Wild animals crossing',
            32:'End speed + passing limits',
            33:'Turn right ahead',
            34:'Turn left ahead',
            35:'Ahead only',
            36:'Go straight or right',
            37:'Go straight or left',
            38:'Keep right',
            39:'Keep left',
            40:'Roundabout mandatory',
            41:'End of no passing',
            42:'End no passing veh > 3.5 tons' }
```

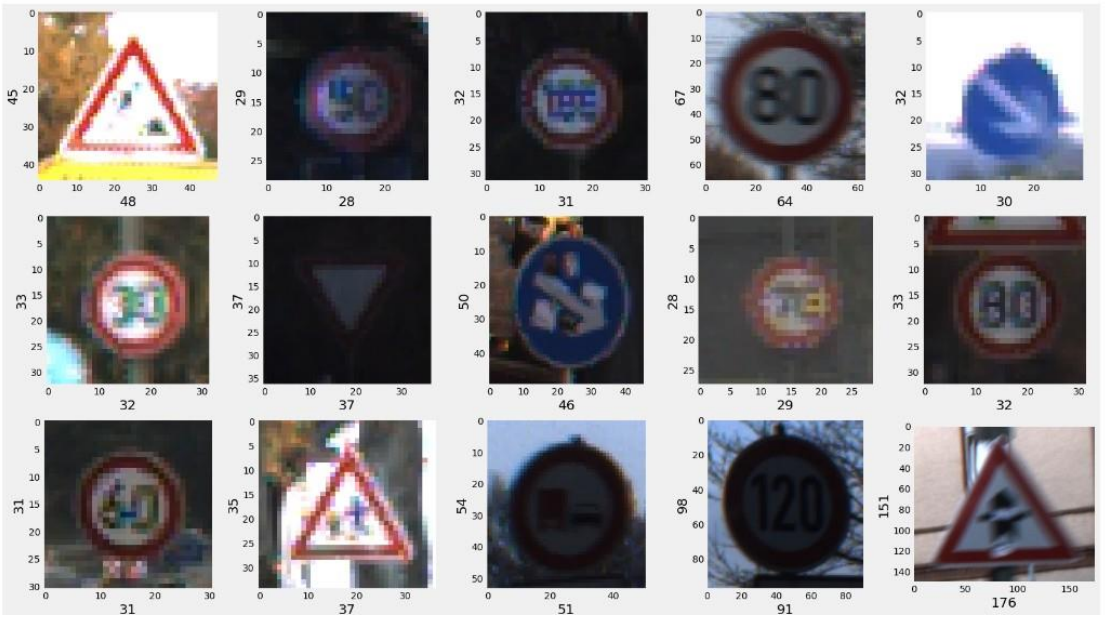
Visualizing The Dataset

```
folders = os.listdir(train_path)

train_number = []
class_num = []

for folder in folders:
    train_files = os.listdir(train_path + '/' + folder)
    train_number.append(len(train_files))
    class_num.append(classes[int(folder)])

# Sorting the dataset on the basis of number of images in each class
zipped_lists = zip(train_number, class_num)
sorted_pairs = sorted(zipped_lists)
tuples = zip(*sorted_pairs)
train_number, class_num = [ list(tuple) for tuple in  tuples]

# Plotting the number of images in each class
plt.figure(figsize=(21,10))
plt.bar(class_num, train_number)
plt.xticks(class_num, rotation='vertical')
plt.show()
```



Visualizing 25 random images from test data

```
[ ] # Visualizing 25 random images from test data
    import random
    from matplotlib.image import imread

    test = pd.read_csv(data_dir + '/Test.csv')
    imgs = test["Path"].values

    plt.figure(figsize=(25,25))

    for i in range(1,26):
        plt.subplot(5,5,i)
        random_img_path = data_dir + '/' + random.choice(imgs)
        rand_img = imread(random_img_path)
        plt.imshow(rand_img)
        plt.grid(b=None)
        plt.xlabel(rand_img.shape[1], fontsize = 20)#width of image
        plt.ylabel(rand_img.shape[0], fontsize = 20)#height of image
```



Collecting the training data

```
image_data = []
image_labels = []

for i in range(NUM_CATEGORIES):
    path = data_dir + '/Train/' + str(i)
    images = os.listdir(path)

    for img in images:
        try:
            image = cv2.imread(path + '/' + img)
            image_fromarray = Image.fromarray(image, 'RGB')
            resize_image = image_fromarray.resize((IMG_HEIGHT, IMG_WIDTH))
            image_data.append(np.array(resize_image))
            image_labels.append(i)
        except:
            print("Error in " + img)

# Changing the list to numpy array
image_data = np.array(image_data)
image_labels = np.array(image_labels)

print(image_data.shape, image_labels.shape)
```

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

Shuffling the training data

```
[ ]  shuffle_indexes = np.arange(image_data.shape[0])
     np.random.shuffle(shuffle_indexes)
     image_data = image_data[shuffle_indexes]
     image_labels = image_labels[shuffle_indexes]
```

Splitting the data into train and validation set

```
[ ] X_train, X_val, y_train, y_val = train_test_split(image_data, image_labels, test_size=0.3, random_state=42, shuffle=True)

    X_train = X_train/255
    X_val = X_val/255

    print("X_train.shape", X_train.shape)
    print("X_valid.shape", X_val.shape)
    print("y_train.shape", y_train.shape)
    print("y_valid.shape", y_val.shape)
```

```
X_train.shape (27446, 30, 30, 3)
X_valid.shape (11763, 30, 30, 3)
y_train.shape (27446,)
y_valid.shape (11763,)
```

One hot encoding the labels

```
y_train = keras.utils.to_categorical(y_train, NUM_CATEGORIES)
y_val = keras.utils.to_categorical(y_val, NUM_CATEGORIES)

print(y_train.shape)
print(y_val.shape)
```

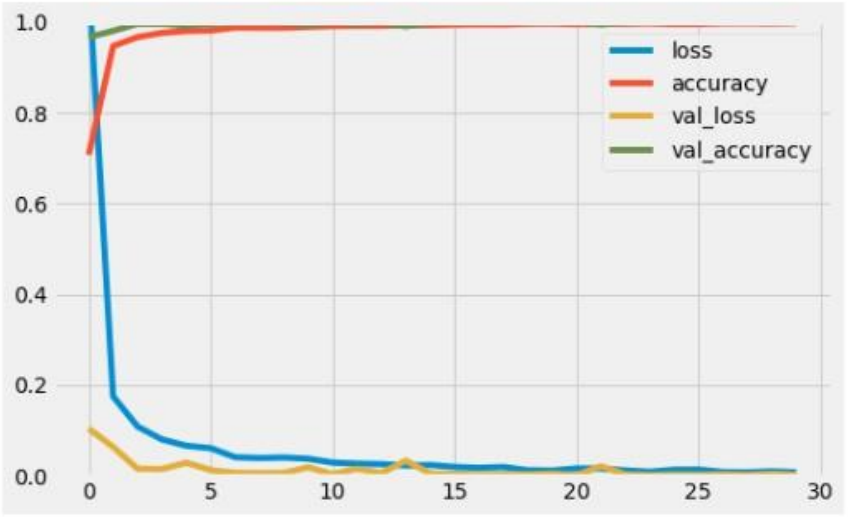
```
(27446, 43)
(11763, 43)
```

Making the model

```
model = keras.models.Sequential([
    keras.layers.Conv2D(filters=16, kernel_size=(3,3), activation='relu', input_shape=(IMG_HEIGHT,IMG_WIDTH,channels)),
    keras.layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2, 2)),
    keras.layers.BatchNormalization(axis=-1),

    keras.layers.Conv2D(filters=64, kernel_size=(3,3), activation='relu'),
    keras.layers.Conv2D(filters=128, kernel_size=(3,3), activation='relu'),
    keras.layers.MaxPool2D(pool_size=(2, 2)),
    keras.layers.BatchNormalization(axis=-1),

    keras.layers.Flatten(),
    keras.layers.Dense(512, activation='relu'),
    keras.layers.BatchNormalization(),
    keras.layers.Dropout(rate=0.5),

    keras.layers.Dense(43, activation='softmax')
])
```

Evaluating the model

```
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
plt.gca().set_ylim(0, 1)
plt.show()
```



Loading the test data and running the predictions

```
test = pd.read_csv(data_dir + '/Test.csv')

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

data =[]

for img in imgs:
    try:
        image = cv2.imread(data_dir + '/' +img)
        image_fromarray = Image.fromarray(image, 'RGB')
        resize_image = image_fromarray.resize((IMG_HEIGHT, IMG_WIDTH))
        data.append(np.array(resize_image))
    except:
        print("Error in " + img)
X_test = np.array(data)
X_test = X_test/255

pred = model.predict_classes(X_test)

#Accuracy with the test data
print('Test Data accuracy: ',accuracy_score(labels, pred)*100)
```

```
from sklearn.metrics import classification_report

print(classification_report(labels, pred))
```

```
              precision    recall  f1-score   support

           0       0.79      1.00      0.88        60
           1       1.00      1.00      1.00       720
           2       0.99      1.00      0.99       750
           3       0.96      0.98      0.97       450
           4       1.00      1.00      1.00       660
           5       0.98      0.99      0.98       630
           6       1.00      0.97      0.99       150
           7       1.00      1.00      1.00       450
           8       1.00      0.97      0.98       450
           9       1.00      1.00      1.00       480
          10       1.00      1.00      1.00       660
          11       0.96      1.00      0.98       420
          12       1.00      0.98      0.99       690
          13       1.00      1.00      1.00       720
          14       1.00      1.00      1.00       270
          15       0.99      1.00      0.99       210
          16       1.00      1.00      1.00       150
          17       1.00      1.00      1.00       360
          18       0.99      0.93      0.96       390
          19       0.97      1.00      0.98        60
          20       0.98      1.00      0.99        90
          21       0.83      1.00      0.91        90
          22       0.99      0.84      0.91       120
          23       0.99      0.99      0.99       150
          24       0.97      0.98      0.97        90
          25       1.00      0.97      0.98       480
          26       0.83      1.00      0.91       180
          27       0.86      0.50      0.63        60
          28       0.99      0.99      0.99       150
          29       0.85      1.00      0.92        90
          30       0.99      0.83      0.90       150
          31       1.00      1.00      1.00       270
          32       1.00      1.00      1.00        60
          33       0.99      1.00      1.00       210
          34       1.00      1.00      1.00       120
          35       0.99      0.98      0.99       390
          36       0.98      1.00      0.99       120
          37       1.00      0.98      0.99        60
          38       1.00      1.00      1.00       690
          39       0.99      0.98      0.98        90
          40       0.96      0.98      0.97        90
          41       1.00      1.00      1.00        60
          42       0.99      1.00      0.99        90

    accuracy                           0.98     12630
   macro avg       0.97      0.97      0.97     12630
weighted avg       0.99      0.98      0.98     12630
```

```
plt.figure(figsize = (25, 25))

start_index = 0
for i in range(25):
    plt.subplot(5, 5, i + 1)
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])
    prediction = pred[start_index + i]
    actual = labels[start_index + i]
    col = 'g'
    if prediction != actual:
        col = 'r'
    plt.xlabel('Actual={} || Pred={}'.format(actual, prediction), color = col)
    plt.imshow(X_test[start_index + i])
plt.show()
```

6. Conclusion:

A traffic sign detection system based on deep learning techniques has the potential to improve road safety by accurately and efficiently identifying traffic signs on roads. Deep

learning algorithms, particularly convolutional neural networks (CNNs), can be trained to recognize a wide range of traffic signs with high accuracy.
However, the performance of the system depends on several factors, including the quality and quantity of the training data, the algorithm used, and the hardware resources available. To ensure the system's success, it is essential to use diverse and high-quality datasets, test the system on a separate validation set, optimize the algorithm's performance, and ensure adequate hardware resources.
Future work in this field could focus on improving the system's performance in adverse weather and lighting conditions, optimizing the algorithm's speed for real-time performance, developing multi-task learning algorithms that incorporate related tasks such as vehicle detection and pedestrian detection, and enabling large-scale deployment of the system.
A traffic sign detection system based on deep learning has the potential to significantly improve road safety, but it requires careful attention to data quality, algorithm performance, and hardware resources to ensure its success.

References:

https://ieeexplore.ieee.org/abstract/document/6033589
https://www.caeaccess.org/research/volume3/number2/billah-2015-cae-651877.pdf
https://paperswithcode.com/paper/evaluation-of-deep-neural-networks-for
https://www.sciencedirect.com/science/article/abs/pii/S092523121830924X
https://www.sciencedirect.com/science/article/pii/S2405844022030808
https://www.analyticsvidhya.com/blog/2021/12/traffic-signs-recognition-using-cnn-and-keras-in-python/
https://paperswithcode.com/paper/novel-deep-learning-model-for-traffic-sign
https://ieeexplore.ieee.org/document/649946