Volume 7, Issue 9, September – 2022 International Journal of Innovative Science and Research Technology
ISSN No:-2456-2165

Image Classification for Traffic Sign Recognition

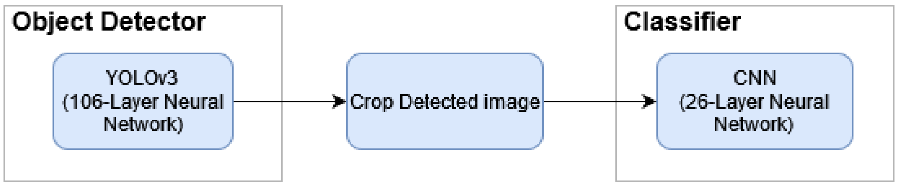
Vedant Mahangade, Atharva Kulkarni, Siddhant Lodha, Atharva Awale,
Department of Information Technology,
Sinhgad College of Engineering,
Pune, India

Abstract:- Traffic signs are a crucial part of our road environment. They provide crucial information, sometimes compelling recommendations, to ensure that driving behaviors are adjusted and that any currently enforced traffic regulations are observed. With majority of modern automobiles equipped with an automated driving assistance systems a robust and efficient traffic sign classifier would be considered a must. We propose a Traffic Sign Recognition system which follows a neural network-based approach that uses YOLOv3 (You Only Look Once Version 3) as object detector rather than a classifier followed by a CNN (Convolutional Neural Network) to classify traffic signs. This approach of dividing the modules to compute single task turns out to improve the system's performance even with limited training thus providing a better platform for development of models to solve similar tasks.

Keywords:- *YOLOv3, CNN, Traffic Sign, Image, Classification, Detection, Recognition.*

I. INTRODUCTION

Most traffic accidents nowadays are caused by drivers' unintentional disregard for traffic signs. With an automated driving assistance system, we will be able to dramatically reduce the number of traffic accidents and even prevent fatal accidents. The most prevalent of traffic sign recognizers today, are image classification systems based on one of two methods: one which contains a single algorithm like YOLO or R-CNN (Region-based Convolutional Neural Networks) for detection as well as classification and others which use traditional ML (Machine Learning) algorithms like HOG (Histogram of Oriented Gradients) for object detection along with classification algorithms like CNN. Traditional ML algorithms are not suitable for operations on multidimensional inputs like images, whereas algorithms like YOLO are specifically designed for tasks involving images. YOLO in its own can detect as well as classify objects in an image, however it is quite taxing in terms of training requirements. In order to obtain a comparatively better result and performance than other classifiers, a YOLO model has to be trained on thousands of images per class. Our proposed system aims to overcome this drawback by dividing the system into detector and classifier.


Object Detector
YOLOv3 (106-Layer Neural Network)
Crop Detected image
Classifier
CNN (26-Layer Neural Network)
Fig. 1:- Functional Block Diagram

We use a YOLOv3 module only for detection of traffic signs, instead of training the model to classify traffic signs across 43 different classes, we trained the model to detect traffic signs in images across 4 types based on color and shape. The detected sign is cropped and passed on to a separate 26 layered CNN model which classifies the traffic sign across 43 different classes. With this approach we can ensure that accuracy and efficiency of detection as well as classification is maintained even with limited training data.

II. LITERATURE REVIEW

A. *'Convolutional Neural Networks for image classification'* by Nadia Jmour, Sehla Zayen, Afef Abdelkrim

The proposed approach involves special case of transfer learning on CNN variation called AlexNet applied on the large-scale datasets from ImageNet, by transferring its learned image representations and reuse them to the classification task with limited training data. The main idea is based on designing a method which reuses a part of training layers of AlexNet. But this approach comes with limitations such as the model depends very much on the accuracy, fitting and availability of pre-trained model.

B. '*Traffic Sign Detection and Recognition using a CNN Ensemble*' by Aashrith Vennelakanti, Smriti Shreya, Resmi Rajendran, Debasis Sarkar, Deepak Muddegowda, Phanish Hanagal

The System is divided into two phases: detection and recognition. Detection is done based on shape and color of the traffic sign, followed by the sign validation. Once the sign is detected, the portion of the image is cropped and is feed to a CNN model for recognition. The use of CNN ensembles increases the accuracy of recognition as each CNN is trained separately over multiple epochs. However, since the detection is done on bases of shape and color of traffic sign, the model would provide skewed results if image provided is distorted or partial.

C. ‘*Traffic Signs Recognition in a mobile-based application using TensorFlow and Transfer Learning technics*’ by Annamária R. Várkonyi-Kóczy, Abdallah Benhamida, Miklos Kozlovszky

The proposed system uses a Single Shot MultiBox Detector (SSD) which is based on one single deep neural network to train the model on multiple objects per image. The system uses TensorFlow with transfer learning technique that makes it training process easier because of a pre-trained CNN model. The input to the network is images from the data set containing images with resolution of 300×300 pixels with multiple objects to provide faster training time and faster detection results compared to other types of neural networks. However, since the system was more focused on faster detection with least latency, the results were less accurate and was more prone to false outputs.

III. PROPOSED SYSTEM

We built a Traffic Sign Recognition System which can be split up into three functional modules, an object detector, a preprocessor and a classifier. However, the entire system contains more than just these three modules and the block wise representation can be seen in the given block diagram:

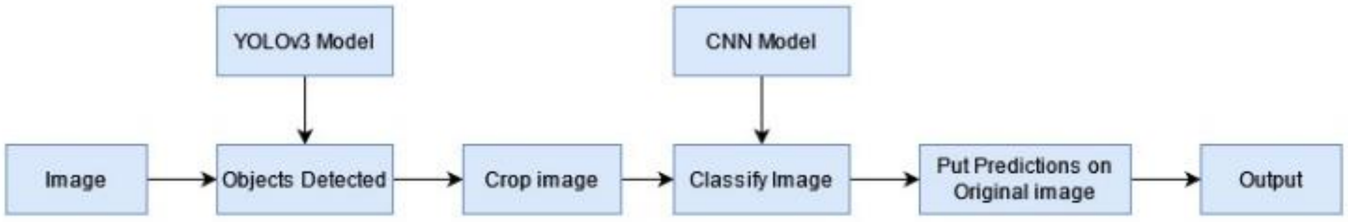


Fig. 2. Block Diagram

A. Object Detection

The object detector is basically a YOLOv3 model which is 106 Layer Convolutional Neural Network, based on the darknet YOLOv3 model trained on COCO dataset for image classification for 80 classes.

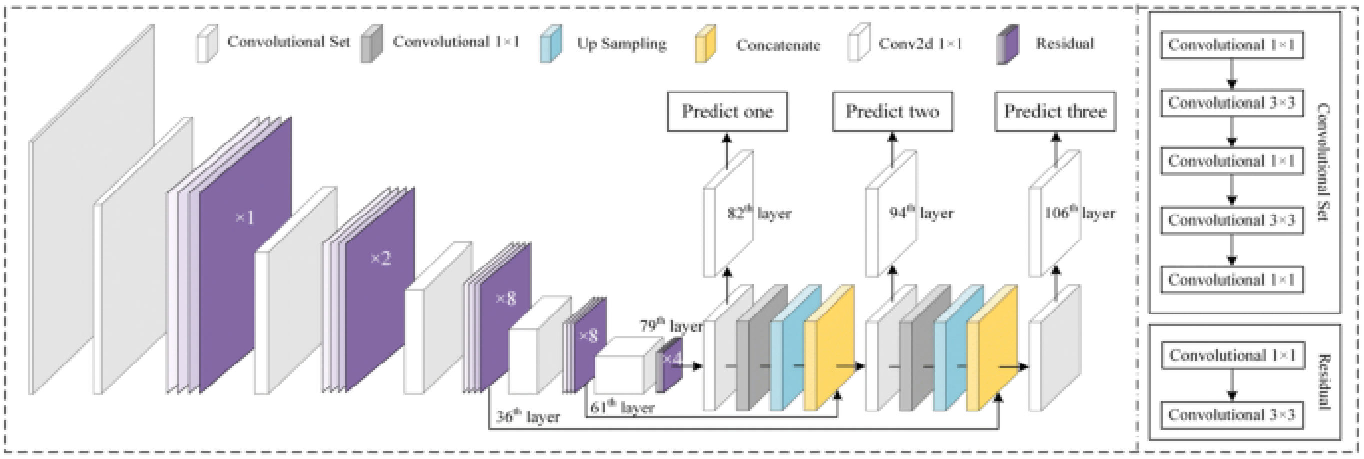


Fig. 3:- YOLOv3 Architecture

The model used in our system is trained on GTSDB (German Traffic Sign Detection Benchmark) dataset to detect traffic signs in an image. It contains 3 output layer and on each of these 3 layers the output contains bounding box co-ordinates and confidence on the scale of 0-1 for 4 following categories: Prohibited, Danger, Mandatory and Others

The model might detect irrelevant objects from the image, these objects are filtered out by setting the minimum confidence as 0.5.

B. Preprocessing

Although we have filtered out the weaker detections there might be some ambiguity in the detected objects. We use non maximum suppression of the resultant bounding boxes to exclude some of bounding boxes if their confidences are lower than 0.3 or other bounding box is present for same region with better confidence. For each ROI (Region Of Interest) detected, we crop the image to apply few transformations before feeding it to the classifier.

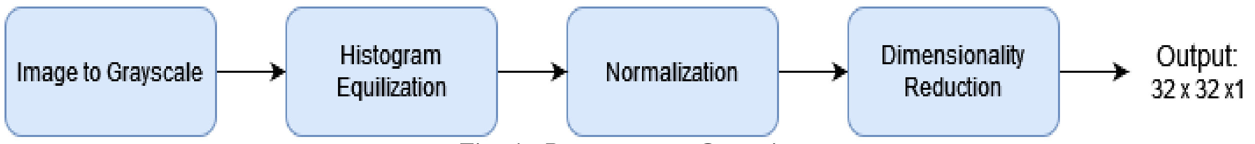


Fig. 4:- Pre-processor Operations

The cropped image is converted to grayscale so that single intensity value for each pixel is retrieved instead to 3. Then we apply Histogram equalization the grayscale image, it increases the global contrast of the image to help distinguish between a background and foreground. Now we normalize the image based on intensity which changes the range of pixel intensity values. Since the image is in grayscale, only one channel needs to be normalized. Then we apply dimensionality reduction by collapsing the image into single channel and the processed image has dimensions as 32×32×1. The image is then forwarded to CNN classifier.

C. Classifier

The CNN classifier used is a 26 layered architecture which includes convolutional, max pooling, batch normalization, dense, dropout and flatten layers. The filter size used is 3X3 and the number of filters in first layer is 16 which increases to 64 at the end of the feature extraction segment. To prevent the over-fitting of the model we add dropout layer. There are 4 dense layer which perform the classification task, the last of which uses softmax classifier function which is best suited for multi-class classification.

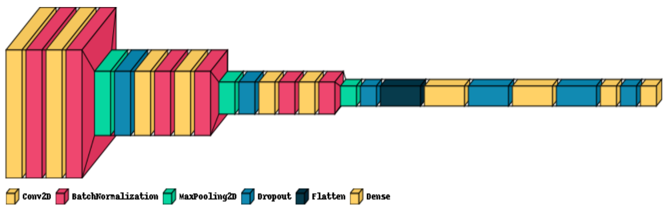


Fig. 5:- 26-Layer CNN Architecture

The input to the CNN is an image of 32X32X1 dimension. The filters move around this 2D matrix producing feature maps by performing dot product. These are then feed to the dense or the fully connected layers of the CNN to perform matrix vector multiplication for classification and the last layer implements softmax function on the layer inputs to provide list of classes and its probability. The one with highest probability is considered as the label for traffic sign.

IV. PHASE SPLIT UP

A. Dataset Generation

Since the system uses two different models for two different tasks, we trained each model on different datasets. The YOLOv3 model which is used as an object detector has been trained on GTSDB dataset and the CNN model has been trained on GTSRB (German Traffic Sign Recognition Benchmark) dataset.

➢ *Detection Dataset*

The GTSDB dataset used as detection dataset contains 900 images with traffic scenes spread over 43 classes which significantly low for training a YOLOv3 model. The images per class were very few and the distribution was overall uneven as the distribution graph below shows:

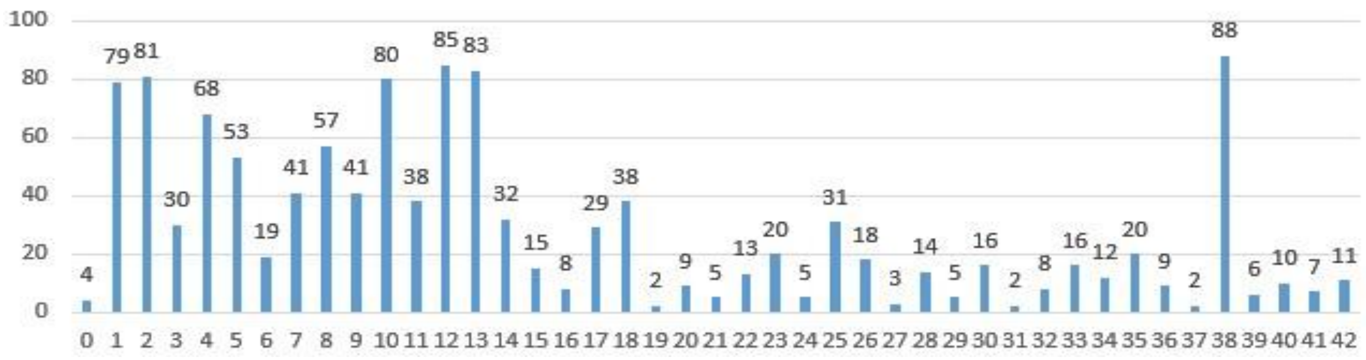


Fig. 6:- Detection Dataset Original Distribution

As mentioned, in our approach we used the YOLOv3 as a detector rather than a classifier, we divided the dataset across 4 categories instead of considering 43 classes. The 4 categories were based on following characteristics of shape, background color and border color:

- Prohibitory – Circular, White background, Red border
- Danger – Triangular, White background, Red border
- Mandatory – Circular, Blue background
- Others – Remaining classes

After the updated categorization of dataset, the distribution was comparatively balanced:

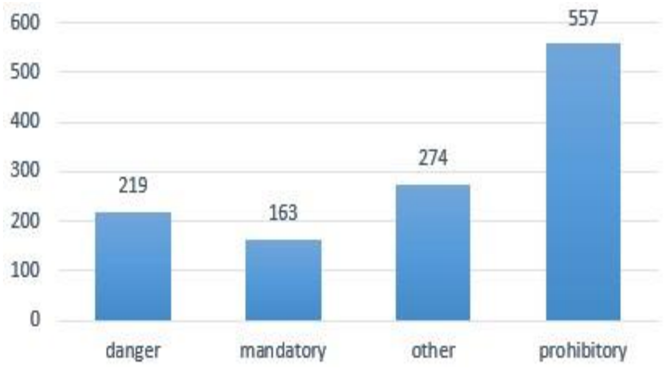


Fig. 7:- Detection Dataset Categorized Distribution

The training dataset, by default had a ground truth file in '.txt' format containing image file names and their objects' coordinates and the images were '.ppm' files. We created a python script to convert the entire dataset into YOLO specific format, which is, images with '.jpg' extension and a text file for each image containing their object coordinates. The dataset was then divided to form separate training and validation datasets with 5:1 train to validation ratio.

➢ *Classification Dataset*

The CNN model used was trained on GTSRB dataset which contains 50000 cropped traffic sign images. The images per class although unevenly distributed, were significantly enough for classification of traffic signs. The distribution of number of images by class was as below:

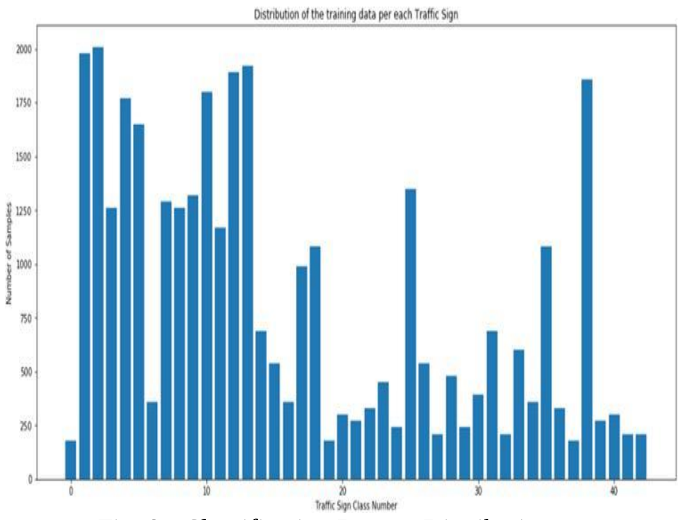


Fig. 8:- Classification Dataset Distribution

65% of the images where used for training, 25 for validation and 10% for testing. These images where in pickled format with size of 32×32 pixels and were utilized for training and validation as it is.

B. *Training*

Training AI models for image-based operations is a heavily resource intensive task and require high processing power to efficient training. Both the models were trained on Google Colab, a cloud based Jupyter Notebook environment which allows to run python scripts on hosted runtime. These models where trained using NVIDIA Tesla K80 GPU hosted on google colab platform.

➢ *YOLOv3 Training*

The model used for detection was a YOLOv3 network from darknet which was pretrained on COCO image dataset for 80 classes. The model training parameters were changed for training the model on the traffic sign dataset. We trained the model for 8000 epochs using the darknet framework, which provides efficient use of resources while training.

For training the model the parameters set were as following:

Parameters	Formulas and Values
Batch	32 for training and 1 for testing
Subdivisions	16 for training and 1 for testing
Max batches	(no of classes * 2000) = 4*2000 = 8000
Steps	80% of max_batches, 90% of max_batches = 6400, 7200
Number of filters	(no of classes + coordinates + 1)*masks = (4+4+1)*3 = 27
Input Size	416, 416
Learning Rate	0.001
Learning Decay Rate	0.0005

Table 1:- YOLO model Training Parameters

➢ *CNN Training*

Two CNN models were taken into consideration, a 10-layer neural network and one 26-layer network. Each trained several times over different hyper parameter to find the best suited model with better performance. Both the models where trained for different number of epochs ranging from 10 to 30, with batch size varying from 32 to 128 and on different learning rates as 0.001, 0.0008, 0.0005 and 0.0001.

The models were evaluated on basis of Test accuracy and test loss. For both the architectures, the models with highest test accuracy and considerably low test loss was considered for comparison. The table below shows the hyper-parameters and performance metrics of best models obtained from two architectures:

Model	**Hyper-parameters**				**Metrics**	
	Learning Rate	***Batch Size***	***Steps per epoch***	***Epochs***	***Test Loss***	***Test Accuracy***
Model 1	0.0005	64	544	20	0.193	0.926
Model 2	0.0001	32	1088	30	0.189	0.959

Table 2:- CNN model Training Parameters

The model 2 loss is less than model 1, and the loss curve over validation set is also comparatively smooth. For accuracy the model 1 performs better with slightly high accuracy than model 2 but the accuracy curve is not as smooth as that for model 2, which means that the model 1 is slightly overfit. Considering these factors, we decided to use model 2 as our final model for classification.

C. *UI development*

For development of UI we used PyQT5 which is a python GUI toolkit. PyQT5 offers cache functionality therefore the app launches quicker. The UI components and the system modules are kept separate and are linked together with a loader file. User can run the loader file to use the classifier, upload the image, click on Classify button and the result image with bounding boxes as well the output on output-panel is displayed. A copy of result image is stored in the results folder with the text file containing the list of objects.

V. RESULTS

A. *Object Detector Evaluation*

The YOLO model was evaluated based on mAP (Mean Average Precision) with IoU (Intersection over Union) of 50% or in short mAP@50. The final model used for the system has a mAP@50 value of 99.135%. For every 1000 epochs while training the mAP was calculated and the model was saved. The model with highest mAP value was taken into consideration. The variation of mAP over 8000 epochs can be seen in the graph below.

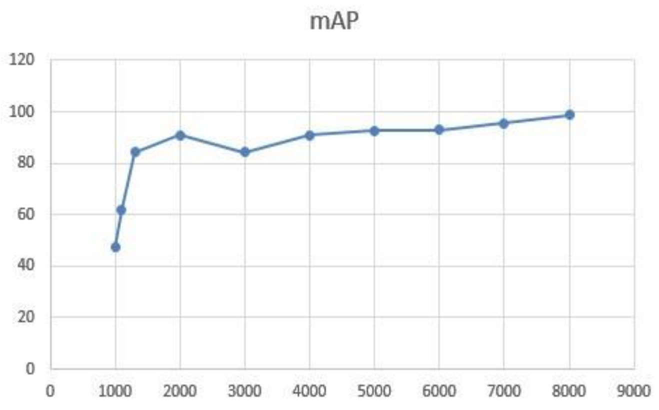


Fig. 9:- mAP over Epochs Curve

The mAP is calculated by taking mean of average precision of the recognition of 4 classes which are given below:

- Prohibitory = 100.00%
- Mandatory = 96.98%
- Danger = 100.00%
- Other = 99.56%

These values are obtained by testing the model on 111 images of the test set. On average the model takes 0.88 seconds to detect signs and can detect up to 5 signs and sometimes more depending upon the quality of image.

B. *Classifier Evaluation*

The 26-layered CNN used has been tested on test dataset containing 12630 images with loss and accuracy as metrics. The best model was chosen from two architectures 1st with 7 training parameter variations and the second with 13 therefore the model used was best among 20, whose parameters were empirically determined. The model's test loss score is 0.1897 and the accuracy score is 0.9597. The two images below depict the Loss and Accuracy curve of the 26-layered CNN model used in the system.

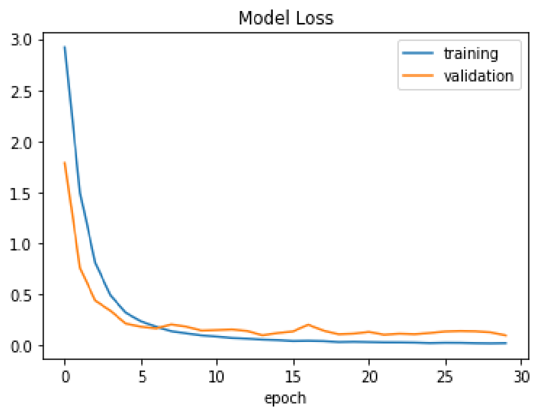


Fig. 10:- Loss Curve

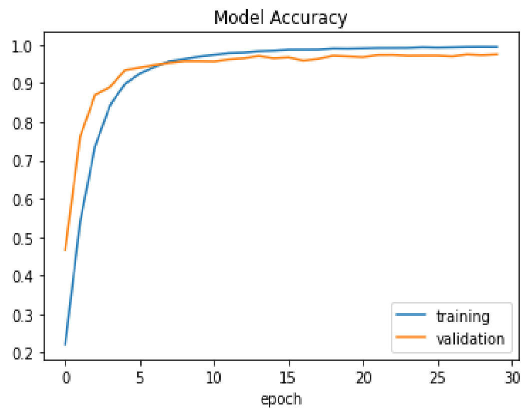


Fig. 11:- Accuracy Curve

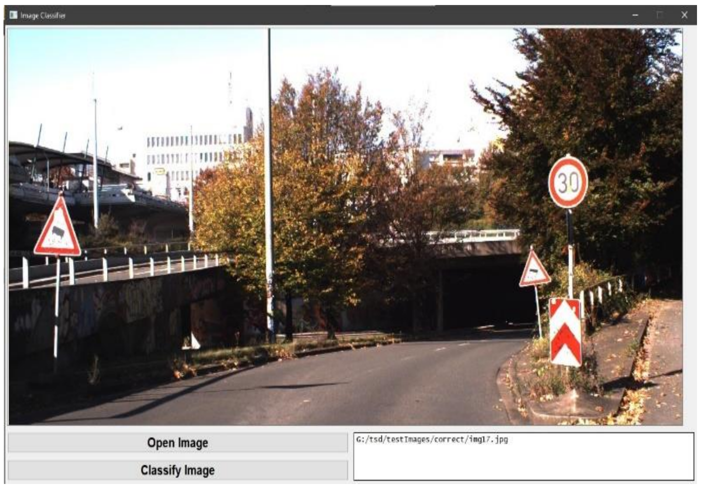
For 1st classification the model takes 0.524 seconds on average and for rest of the classification around 0.096 seconds. For the 1st classification the model takes comparatively more time than that for others.

C. *Results*

Since two different models with different nature and objectives are used for detection and classification, a combined metric to evaluate the those two at the same time cannot be established. Hence time was the only common factor which can be used to test the entire system's performance.

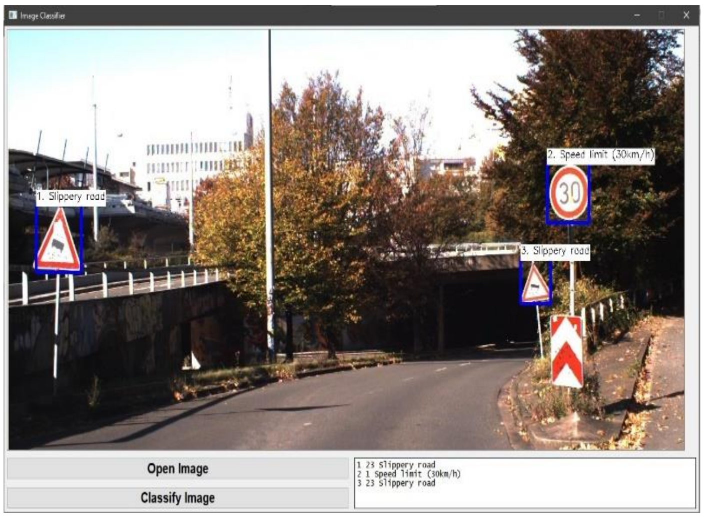
The average computational time calculated for 111 images for entire process of loading the image, detecting objects, preprocessing the cropped images, classifying the traffic signs, putting labels, saving the image and displaying it comes out to be around 2.579 seconds.

The two images below are the snapshots of one of test-cases. The input image provided to the system containing more than 1 traffic sign and the output displayed by system.



```
G:/tsd/testImages/correct/img17.jpg
```

Fig. 12:- Input Image



```
1 23 Slippery road
2 1 Speed limit (30km/h)
3 23 Slippery road
```

Fig. 13:- Output Image with Classificaiton Label

The system took exactly 1.96382 seconds to detect and classify 3 traffic signs from the image correctly. The maximum number of traffic sign detected in an image were 11 and were correctly classified.

VI. CONCLUSION

The traffic sign classifier gives comparatively accurate detection and classification in comparatively short time. The result shows that the system is robust and efficient. The system has been successfully implemented with the desired results obtained given that the modules were trained on limited resources.

The system performance is well above the benchmark set by studying the previous methodology. There are various ways new models can be created using this proposed system, for example training the yolo model with Indian traffic scenes with categorized classes and training the CNN with cropped images of expanded classes will help create a system which can be implemented for detecting Indian road traffic signs. Even a video can be given as input by writing a code which divides the video into frames and feeds to the system, the same video can be replaced with a real time feed from camera to provide real-time traffic sign recognition.

REFERENCES

[1]. A. Avramović, D. Tabernik, D. Skočaj, “Real-time Large-Scale Traffic Sign Detection”, Symposium on Neural Networks and Applications, Serbia,2018.5. S. Indolia, A. K. Goswami, S. P. Mishra, P. Asopa, “Conceptual Understanding of CNN a Deep Learning Approach.” International Conference on Computational Intelligence and Data Science,2018.

[2]. Aashrith Vennelakanti, Smriti Shreya, Resmi Rajendran, Debasis Sarkar, Deepak Muddegowda, Phanish Hanagal, “Traffic Sign Detection and Recognition using a CNN Ensemble”. International Conference on Computational Intelligence and Data Science, 2018.

[3]. Abdallah Benhamida, Annamária R. Várkonyi-Kóczy, Miklos Kozlovszky, “Traffic Signs Recognition in a mobile-based application using TensorFlow and Transfer Learning technics”, SoSE 2020 • IEEE 15th International Conference of System of Systems Engineering • June 2-4, 2020 Budapest, Hungary

[4]. Branislav Novak, Velibor Ilić, Bogdan Pavković, “YOLOv3 Algorithm with additional convolutional neural network trained for traffic sign recognition”. 2020 Zooming Innovation in Consumer Technologies Conference (ZINC)

[5]. N Jmour, S. Zayen, A. Abdelkrim, “Convolutional Neural Networks for image classification.” International Conference on Robots & Intelligent System, 2018.

[6]. R. C. Gonzalez, “Deep Convolutional Neural Networks” IEEE Signal Processing Magazine (79-87),2018 15. Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. (2014). “Scalable object detection using deep neural networks,” in Computer Vision and Pattern Recognition Frontiers in Robotics and AI www.frontiersin.org November 2015

[7]. Wang Canyong, “Research and Application of Traffic Sign Detection and Recognition Based Deep Learning.” International Conference on Robots & Intelligent System,2018.