Detection and Classification of Traffic Signs for Driverless Cars

Palash P. Thakur
Department of Eelectronics Engg.
Shri Ramdeobaba College of Engineering and Management
Nagpur, India
thakurpp@rknec.edu

Sharayu R. Choudhari
Department of Electronics Engg.
Shri Ramdeobaba College of Engineering and Management
Nagpur, India
choudharisr@rknec.edu

Vikas R. Gupta
Department of Electronics Engg.
Shri Ramdeobaba College of Engineering and Management
Nagpur, India
guptavr1@rknec.edu

***Abstract*—Higher levels of autonomy have the potential to reduce risky and dangerous driver behaviors. Driverless cars require robust and error free architecture in order to ensure the maximum safety of a person. Hence it is important for every driverless system to be able to identify traffic signs with maximum accuracy. In order to solve this problem, we have proposed a robust solution to detect and classify traffic signs for self-driving driverless cars. It is two stage architecture with a detection algorithm used to detect traffic signs followed by a classification algorithm which classifies detected traffic sign into 43 categories. Canny edge detector is used to detect the edges of the localized traffic sign and to draw shape-based boundary around traffic sign. The detection algorithm used here is YOLOv3 which is state of the art detection algorithm and it is used for real time detection of objects. The classifier architecture is build using CNN (Convolutional Neural Networks) and it is trained on 86,989 images, each of size 32x32 with 3 color channels. The original dataset is the German Traffic Sign Detection Benchmark (GTSDB) with 50,000 images and imbalanced classes, and with the help of equalization of images and some pre-processing it is converted into balanced dataset. The training and testing of detection model is performed on the German Traffic Sign Recognition Benchmark (GTSRB).**

Keywords—Traffic sign detection, detection and classification, Convolution Neural Network, YOLO, Canny edge detector.

I. INTRODUCTION

Advanced driver assistance system (ADAS) requires robust and accurate architecture. Any driverless car must be able to identify all the major objects that comes under its surrounding. One of the most important task in designing Advanced Driver Assistance System(ADAS) is the accurate detection of traffic sign. Since there are large number of traffic signs available, it becomes difficult task for one to accurately detect and classify traffic signs with the same accuracy of driver enabled cars. If any ADAS is deployed without the traffic sign detection architecture, it may result in fatal accidents.

There are many traffic sign detection architectures available that deals with the individual set of problems. In paper [2], CNN aggregate network is used with limited set of data and the model is only able to detect circular and triangular traffic signs. In paper [3], the traffic signs are classified into 5 classes, but individual classification is not performed, SVM (Support Vector Machine) is used for detection and CNN is used for classification. In paper [4], YOLOv2 is used for detection task, CNN architecture is used for classification of only 16 traffic signs. In paper [5], German traffic sign dataset is used with no preprocessing as the dataset contains class imbalance problem and model is trained with imbalance class.

The most challenging task in traffic sign recognition is that it occupies very little space in a frame and it becomes difficult task to detect such small object in a given frame. Currently there are many state of the art detection algorithms is available that can detect any object it is trained on. The challenge comes when the detection in done in real time. Even though most of the detection architecture are state of the art, but they fail to deliver fast and real time detection of objects. If we consider Faster R-CNN [6], it has complex architecture inspired from VGG16 and it cannot be used for real time applications. Whereas YOLOv3 [7] has simpler architecture and can be used for real time detection of objects.

YOLOv3 [7] is single stage detector and it compromises with the accuracy while detecting and classifying at the same time. It is good in detecting small objects like traffic sign in our case but fails to classify them due to its architectural design. Tasks where detection and classification both are involved; multistage methods are proven to be more effective as compared to single stage. Training any model requires a good quality of sufficiently large dataset. GTSRB and GTSDB is the German traffic sign benchmark dataset for both detection and classification and it is used in most of the problems which deals with traffic signs.

In this paper our ultimate goal is to overcome the flaws that previous papers have and create a combined architecture that detects, classify and draw bounding box around the traffic sign based on its shape. We have used YOLOv3 [7] to detect location of traffic signs on a given frame of video and designed a shallow multilayer perceptron architecture with convolutional layer to classify traffic signs into 43 categories and Canny edge detector is used to detect boundaries of traffic signs. Dataset used to perform detection and classification is German traffic sign dataset but with some preprocessing to overcome class imbalance problem.

II. PROPOSED APPROACH

The intuition behind developing the model is to create a solution that solves multiple problems in a single pipeline, and consume least hardware resource and is fast and accurate too. We have partitioned entire model into two parts where each part performs

the task assigned to it. The two phases are detection phase and classification phase respectively. The detection phase detects the images from given frame and return actual location of image from given frame and
classification model classifies the detected part of image into respective category of traffic sign and returns the name of predicted class along with the probability. Instead of drawing rectangular bounding boxes we have performed shape detection for which we have used Canny edge detector [1].

A. Detection Phase

There are large number of state of the art detection algorithms available. They detect and classify large variety of objects and are trained on very huge dataset. But while dealing with the tasks related to driverless cars, real time and fast detection plays a very important role. Many state of the art algorithms are proven to be ineffective in real time detection due to their complex architecture. While all other algorithms fail, YOLO [7] is promising in real time and fast detection of objects. For the detection task we have used darknet framework in order to train our detection model. The YOLO weights are trained specifically for the detection of traffic signs.

We are using YOLOv3 [7] as a detector model. The traffic signs are divided into 4 random categories so as to perform detection task on them. Since as per the architecture of YOLO we cannot keep only one class in detection. The formula for determining batch size is:

Batch size = Number of Classes * 2000

At the same time batch size should not be less than 2000, hence we cannot keep only one class in detection as well, so to avoid this error and to increase batch size we divided traffic signs into 4 parts prohibitory, danger, Mandatory and other. Yolov3 has 53 layers stacked with more 53 layers making total of 106 layers. It has ability to detect small objects and retain low level features, as in our case traffic signs occupy very little space in any frame. There are various types of Yolo architectures are available, we used 416 x 416 model from darknet.

Input to the network= (n, 416,416,3)

Where n= Number of image

We are require to re calculate number of filters after every convolution layers. The formula to calculate filters is

$$filters = (3 * (5 + 4)) = 27$$

In Yolo detection is done at three stages (layers):82,94,106.

At each stage image is down sampled by 32. Every stage is responsible to draw 3 bounding box around that object. With the help of non-maximum suppression and the IOU (Intersection over union) the correct bounding box is predicted. *fig. 1* shows the architecture of YOLOv3[7].

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 1: Yolov3 actual architecture

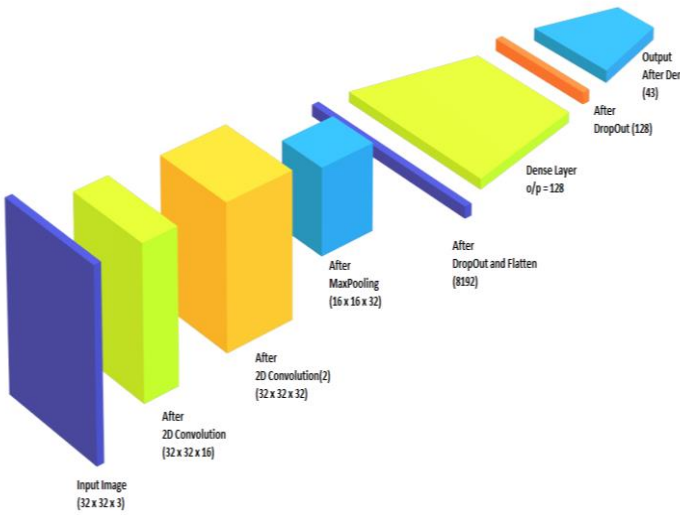

Figure 2: CNN architecture

In our model we are taking the coordinates of traffic sign from given image so as to crop the part of image containing traffic sign and then passing that part to our classification model.

B. Classification phase

In the detection module, the output we get is the part of the image that contains the traffic sign. But this traffic sign is unclassified. For classification of the Traffic signs into its

classes, we use a Neural Network that can classify the given image of traffic sign into 43 unique classes.

The Neural Network Comprises of several of the following layers:

1. Convolutional Neural Network
2. Max Pooling
3. Flatten
4. Dropout
5. Dense

These Neural Networks perform various major tasks in classifying the input image in the 43 classes.

The Output of the Detection Phase i.e. the YOLO network is first preprocessed, resized into an input image of 32 x 32(x 3 channels) and given as an input to our classification Neural Network. The Neural Network architecture is shown in *fig. 2*.

The Convolution Layers retains the properties of the image that is been given input. The series of 2 Convolutional Layers were given an input of (32 x 32 x 3) and Output as (32 x 32 x 32) . In this case we haven't reduced the size of the image using 2D Convolution since we have used a MaxPooling layer for reducing the image. The MaxPooling Layer Reduces the size of 2 axes of the input to half in this network, without retaining the major properties. The input shape is (32 x 32 x 32) and Output (16 x 16 x 32). The DropOut layer ignores a small percentage of training data so as to avoid overfitting of the model on the training data. Overfitting associates with high accuracy on training data but a significantly lesser lower accuracy on testing data. The DropOut layer is followed by the Flatten Layer which converts the 3D array into a 1D Array i.e. (16x16x32) to (8192). This converts the 3D array to an input for the Dense Layers. Dense Layers consist of many layers which convert Input into a desired output shape having many trainable weights. Here we converted (8192) shaped array into (128) shaped array. This is again followed by a DropOut layer for avoiding overfitting. Adding a Dense Layer again, output shape is (43). These 43 Output are the probabilities of the 43 classes where we take the highest class probability and classify the image. The Output layers are Categorical classes, so the Activation Function for the Output Layers have SoftMax activation function

We train this Neural Network by using a Dataset which is shown later in this paper. While training the Network, the time required to train and the accuracy of the model depends on the following factors,

1. Optimizer used.
2. Number of Epochs.
3. Correct Loss function

We use Adam Optimizer in this case. Adam optimizer is a combination of Gradient decent with momentum and RMSprop optimizers. It combines the benefits of both the optimizers into one. The v_t part of the formula refers to the Gradient Decent part and the s_t part refers to the RMSprop part.

The number of epochs for which the Network is trained determines the accuracy. The more it is trained it is the better is the accuracy. But after some particular number of epochs, the accuracy saturates and increases very slowly. This is the ideal point.The *fig.3* shows accuracy vs epochs plot for training (blue line) and testing (orange line) dataset.

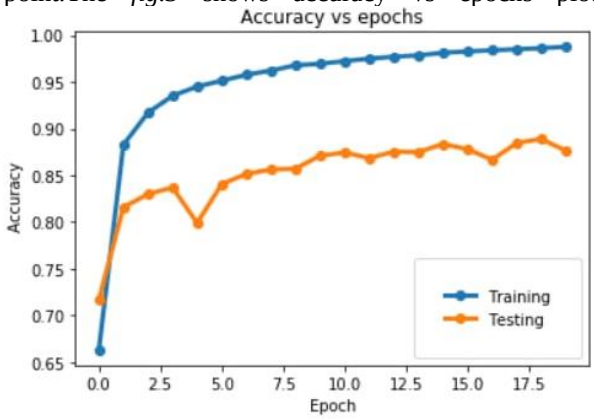

Figure 3: Accuracy vs Epochs

Figure 4: Different classes of traffic signs

The output consists probability distribution in between 43 different classes, so use of relevant classification loss function i.e. CategoricalCrossEntropy is used. This gives the system the correct Loss function according to the data.

C. Edge Detection

As discussed in the approach, we are drawing the edges or the approximate boundaries of the traffic signs with the help of their original positional coordinates that are returned by Yolov3. By default, we can draw rectangular bounding box around the detected traffic sign but it gives no information about the edges, shape and orientation of traffic sign. There are large number of edge detection algorithms are available. Common edge detection algorithms include Sobel, Canny, Prewitt, Roberts, and fuzzy logic methods. In this model we used Canny edge detector [1].

III. DATASET

Any deep learning problem requires a huge amount of good dataset that can be used to train, validate and test our algorithm for its speed, accuracy and error. There are various types of datasets available when dealing with traffic sign. In paper [5], authors have used Japanese dataset for training their model. Same way Chinese data too is available and it is open source. In our paper, we are using German traffic sign dataset [German paper] which is benchmark for traffic sign detection and classification. The dataset is German traffic sign detection benchmark (GTSDB) and German traffic sign recognition benchmark (GTSRB). The GTSDB dataset is used to train Yolov3 for the task of detection of traffic signs. The dataset comprises of total 900 images out of which we are using 800 for training and 100 for testing. The images are labeled and provided with the coordinates of bounding boxes on that image. The sample images are as shown in *fig. 5*:

Figure 5: Representation of training dataset for detection task with bounding boxes around traffic signs

The images for training has resolution of 1360 x 800. Since Yolov3 down samples the image into 416 x 416 we do not need to be worried about the resolution of images. The detection model is trained on this dataset.

Classification model requires very large amount of images of traffic signs. Since we are aiming to categories 43 classes, dataset with very low number of images are not going to give us the good result. The GTSRB dataset is used for classification without extra preprocessing. In paper [5] authors have used the same dataset from GTSRB without any preprocessing. The problem with the original dataset is that the classes are highly imbalanced. Imbalance in class will result in unequal distribution of weights to the traffic signs and result is false accuracy. We have trained the classification model with the original dataset and the training and testing accuracy is 99.9 %, the dataset has 50,000 images. However, we cannot give more priority to accuracy when we know that the classes are highly imbalance and we are training on imbalance classes. Before start training, we have to make sure that the classes are equally distributed. For this we applied rotation and brightening to all the images to create new samples. The sign of the dataset has increased to almost 87,000 after applying this technique of data augmentation. Histogram of 43 classes for training dataset with their number of examples for traffic signs classification before and after equalization by adding transformed images (brightness and rotation) from original dataset is shown in *fig. 6.*

Figure 6: Histogram of dataset before and after preprocessing

Sample of dataset containing different classes is shown in *fig. 3.* After training the classification model, the training accuracy is 98% and testing accuracy is 90%.

IV. RESULT

The network when given an input of an image or video, gives out the image with localised Traffic sign and its class. The model works efficiently in most lighting and weather conditions and is tested on such conditions. For a better visualization of the traffic sign, the edge detection model [1] is also applied for localization. The detection model has mAP (mean Average Precision) of 93 %. The classification model has testing accuracy of 90%. We can see the sample output images in *fig.7.* The darknet framework (YOLOv3) [7] can compute using cpu as well as gpu power. In this case we tested it on both. CPU took

approximately 1 second to process 1 frame (Intel i5 9th gen) and GPU took approx. 0.28 second to process 1 frame (GPU by Google Colab) The GPU setup takes 0.28 seconds to process 1 frame. So, the processing speed is approx. 3 fps (Frames per second) on GPU. On CPU it is 1 fps.

Figure 7: Detected traffic signs

V. CONCLUSION

In this paper, we have detected and classified traffic signs out of raw images. These signs are classified into 43 classes and then are localized using edge detection technique [1]. Here, two stage model is used. One for classification and one for detection. The model is able to detect images in both bright and low light condition. This approach led us to compromise with speed but it is delivering better accuracy. The use of two stage model is done because of lack of good quality of sufficient dataset and computing power, which is required to train a detection model. The speed of model can be improved significantly by using only one model. The model sometimes fails to detect traffic signs from very high resolution videos. Training it on a larger dataset with a better and more diverse resolutions will be done in future. Improving the classification dataset by increasing the variety of resolutions and the number of classes of the training dataset to improve the classification accuracy. The maximum processing performance obtained is 3 fps whereas the real-time input usually works on 25-30 fps. Optimising the performance using known techniques will be done. A new architecture similar to this using shape detection specially designed for detection of traffic signs can be implemented to decrease the noise that will be the input of the Architecture, hence increasing accuracy.

REFRENCES

[1] J. Canny, "A computational approach to edge detection", in IEEE Trans, on Pattern Analysis and Machine Intelligence, 1986.

[2] Aashrith Vennelakanti ,Smriti Shreya , Resmi Rajendran, Debasis Sarkar , Deepak Muddegowda , Phanish Hanagal,"Traffic sign detection and recognition using CNN ensemble", in IEEE International Conference on Consumer Electronics (ICCE), 2019.

[3] Yi Yang, Hengliang Luo, Huarong Xu, and Fuchao WuK. Elissa, "Towards real time traffic sign detection and classification",in IEEE transactions on intelligent transportation systems,2016.

[4] Ryo Hasegawa, Yutaro Iwamoto, Yen-Wei Chen,"Robust detection and classification of Japenese traffic signs in complex scenes based on the Deep learning", in IEEE Consumer Electronics (GCCE). 8th Global Conference, 2019.

[5] Shehan P Rajendran ; Linu Shine ; R Pradeep ; Sajith Vijayaraghavan,"Real time traffic sign recognition using Yolo3 based detector", in Computing, Communication and Networking Technologies (ICCCNT), 10th International Conference. 2019.

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, arXiv:1506.01497v3 [cs.CV] , 2016.

[7] Joseph Redmon, Ali Farhadi, YOLOv3: An increamental improvement, arXiv:1804.02767 [cs.CV], 2018.