Project Report

On

A YOLO Based Approach for Traffic Sign Detection

Submitted by

15IT217 M M Vikram

Under the Guidance of

Prof Ananthanarayana V S

Dept. of Information Technology,

NITK, Surathkal

Date of Submission: April 2, 2018

Dept. of Information Technology

National Institute of Technology Karnataka, Surathkal.

2017-2018

Abstract

Autonomous driving is one of the interesting research areas of modern times and traffic sign detection is a very important and crucial problem in this research. In this project, we propose to explore the YOLO Architecture and its compatibility to solve this problem. The objective is locating and classification of traffic signs in natural street scenes.The key challenge to be addressed in this problem is recognition of minute targets in an extended and complex image background. Other object detection models like Fast R-CNN and Faster R-CNN[1] have been used for this problem. The main drawbacks with such methods is their speed - they fail to be real time. Thus, the motivation behind exploring YOLO for this task is the speed - it is about 6$\times$ faster than faster R-CNN. In this work, we also propose a novel modified loss function for the YOLO model to perform better for traffic sign detection.

Contents

List of Figures

List of Tables

1 Introduction

Over the last few years, autonomous driving cars have become popular among the research community. An autonomous car is a capable of sensing its environment and navigating without any human intervention. One of the most important things for an autonomous car is 'vision', which includes the capability of recognizing and detecting the traffic signs on the road. Traffic sign detection is a challenging task due to hindrances such as occlusion, changing lighting conditions, camera perspective and other factors which arise in natural scenes. Multiple signs in a single view is another challenge to address.

Up till recent years,the conventional approaches for traffic sign detection were based on certain traditional algorithms. The techniques for traffic sign detection generally use manually selected features to obtain the region proposals, and classifiers are then trained to filter out the negatives. Deep convolutional networks are applied to image recognition and object detection as they are giving the desired performance in terms of speed and accuracy. Convolutional neural networks do not require any hand crafted features as they inherently learn the generalized features. CNNs which have gained immense popularity in various classification tasks are now being extended to more rigorous tasks such as object detection which is a composite of both classification and localization - [2] has produced state-of-the-art benchmark on the German Traffic Sign Recognition Dataset.

One of the most important factors in real time traffic sign detection is the test time latency. CNNs were not considered feasible for real time traffic sign detection due to their complex computation. But the evolution of GPUs(Graphics Processing Unit) have paved the way to use CNNs for this purpose, owing to their high computing performance. We require a model which can detect and classify the signs at real time. In this work, we explore the YOLO architecture[3], which exhibits a real time object detection and classification at around 45 frames per second. This would be highly suitable for our problem, which requires high speed and accuracy. We have developed a YOLO based architecture for the traffic sign detection of Belgium Traffic Sign Dataset(BTSD)[4].

The further organization of the report is as follows - Section 2 covers the related works while the Section 3 describes the methodology and Section 4 compiles the results and analysis. Finally, the conclusion and future plan of action for the project is described in Section 5.

2 Literature Review

2.1 Related Work

Traffic sign detection methods can be broadly classified into two categories. This includes the class of techniques which use classical object detection models which rely on the characteristics of the traffic signs and the second class of techniques which include the recent deep learning based approaches which learn the general features instead of using the manually selected features as in the first approach.

The classical detection methods have mainly relied on feature extraction algorithms. Features like color and shape are used for both classification and detection. Images are generally transformed into HSV to overcome RGB color space limitations for various light condition.[5]. [6] presents a color probability model that computes maps based on Ohta space. The general shapes of the traffic signs include circle, triangle, rectangle or any other polygon. Contours extracted from edges along with the features of color and shape are used. In traditional detection methods, histogram of oriented (HOG) feature with SVM classifier are state of art techniques. [7] employs the HOG features along with an SVM classifier which gave a good performance on the German Traffic Sign Detection Benchmark (GTSDB)competition hosted by IJCNN in 2013.

Although manually selected features have achieved higher precision for traffic signs, traditional detection approaches are very specific and lack robustness towards changing scenes and its associated complexities. Thus deep learning(DL) approaches have become popular for object detection problems in recent years.The deep learning approaches can be further classified into region based techniques and regression based techniques.

Region-based convolution neural networks(R-CNN) give a superior performance in terms of localization and classification accuracies. They generate around 2000 regions of interest(ROI) by using the selective search algorithm and CNNs extract the features from these ROIs separately. At the last stage, an SVM classifier is used to predict the classes of objects. Linear regression for fine tuning the positions and sizes of the bounding boxes, further improves the performance of R-CNN [8]

Although the R-CNNs are able to give the desired results, the concerns are about its high cost in terms of memory and latencies. Employing neural network instead of Selective search in Fast-RCNN and Faster R-CNN not only improves the speed but also helps them achieve higher accuracies[1].

The other approach is an end-to-end learning model based on regression. Redmon J.

et al's algorithm[3] and YOLOv2 [9] integrate the tasks of localization and classification into a single convolutional network, thus increasing the speed at the cost of precision. The recently popular SSD(single shot detection) introduces default boxes and multi-scale feature mapping layers[10].

Thus we can conclude that the two-stage methods do better in terms of the localization and classification accuracies, however at increased cost of resources and latencies. The end-to-end one stage methods are useful in producing faster results at a reduced precision. Here, the latter would be more appropriate as the traffic sign detection must be real time.

2.2 Outcome of Literature Review

As explained above, different models for traffic sign detection were explored and their pros and cons were understood. It was concluded that the YOLO model might be the most suitable for the application, because of its speed and lowest test time latency compared to other models. Even though the accuracy of the YOLO model is less compared to other models as discussed above, the ability of YOLO to detect and classify objects at real time outweighs this disadvantage. The accuracy of R-CNNs for object detection and classification is very good but would not be suitable for this application owing to the high latencies.

2.3 Problem Statement

To develop a YOLO based deep CNN model for Traffic Sign Detection and Classification on the Belgium Traffic Sign Dataset. The objective is to obtain a model which can detect and classify traffic signs simultaneously whose performance is comparable to other contemporary approaches such as R-CNNs and fast R-CNNs etc but in real time.

2.4 Research Objectives

The general objective is to obtain a YOLO based deep CNN model for Traffic Sign Detection and Classification on Belgium Traffic Sign Dataset(BTSD) which can detect and classify traffic signs simultaneously in real time with performance comparable to that of R-CNNs and fast R-CNNs. The specific objectives in order to obtain above objective are :

- To analyse the existing YOLO framework on PASCAL VOC dataset

- To train and test YOLO based deep CNN model on BTSD
- To estimate modifications in the YOLO framework that might suit better for BTSD
- To develop a new YOLO model with estimated modifications and train it on BTSD
- To compare the performance of our model with that of the existing works.

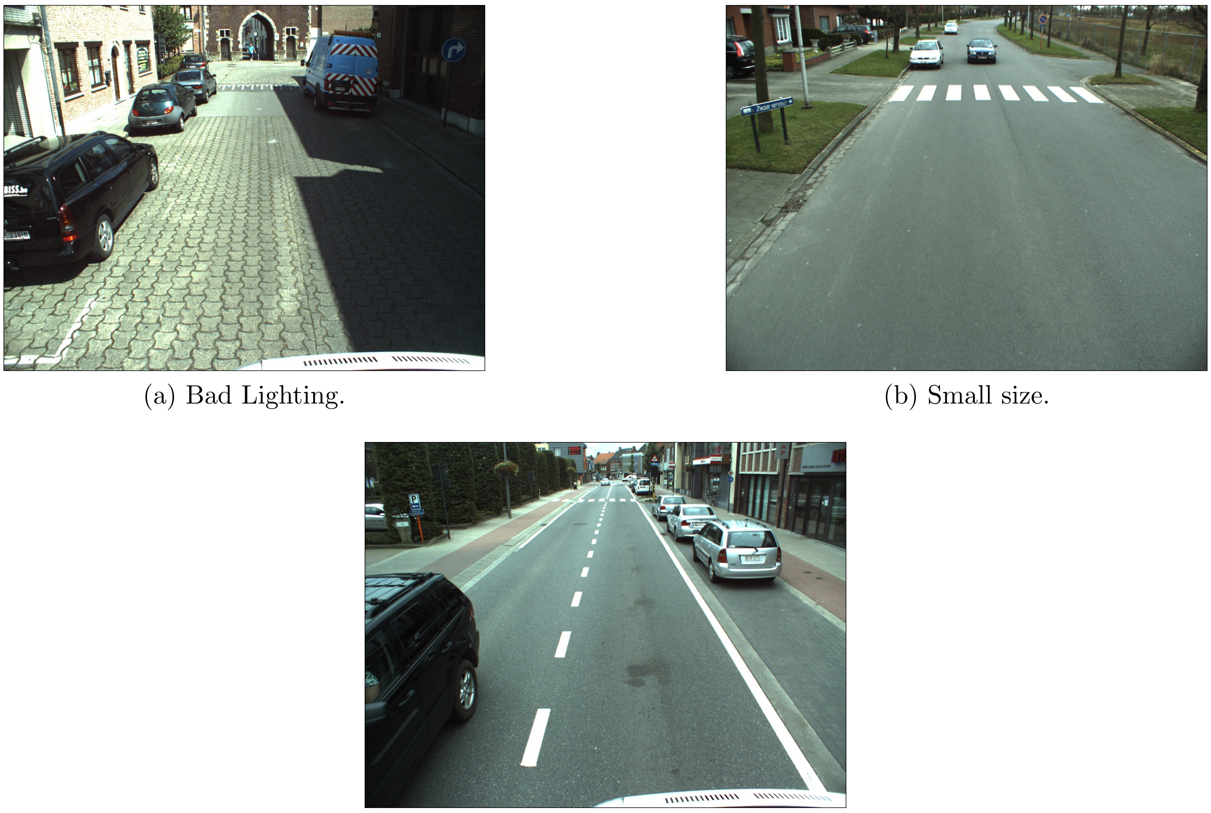
3 Methodology and Framework

Works such as [1] have given remarkable results on traffic sign detection tasks. However, they are not real time which is a key factor to be considered while evaluating the performance. Our model is built with YOLO Architecture as base focusing on fine-tuning the existing model based on the traffic signs.

3.1 Architecture

YOLO makes use of a single neural network in predicting bounding boxes and probabilities of predicted class of objects. The images pass through the network only once. The image to be input is divided into $S \times S$ grids. Each grid cell makes k predictions of bounding boxes and confidence scores of these bounding boxes, with C conditional class probabilities. Each of the bounding boxes is characterized by a quintuple (x, y, w, h, cfd). The (x, y) coordinates are the center offset of the bounding box compared to the bounds of the grid cell. The values w and h are the width and height of the predicted object relative to the whole image. The confidence cfd is defined as $P_r(Object) * IOU_{pred}^{truth}$. The value of $P_r(Object)$ is 1 when a grid cell contains a part of a ground truth box, else its value is 0. IOU refers to to the intersection over union between the predicted box and its corresponding ground truth box, this metric varies between 0 and 1, where 0 means no overlap and 1 means that the predicted box is same as the ground truth. For each region, there is only one set of class scores C for all bounding boxes in that region. Thus, the YOLO network gives an output which is a vector of $S \times S \times (5B + C)$ numbers for each image. The cfd and class probabilities together determine the grids whose prediction will be selected.

At this point, the only difference in architecture between the original YOLO model and ours is in the last layer. The Pascal VOC dataset on which YOLO was trained had 20 classes where as the BTSD has only 13 classes. The number of units in last layer, in the case of original YOLO is $7 \times 7 \times (5 \times 2 + 20)$ and in our model it is $7 \times 7 \times (5 \times 2 + 13)$.


(a) Bad Lighting.

(b) Small size.

(c) Cluttered background.

Figure 1: Dataset represents the problems in real life traffic sign detection

3.2 Data Set

We have used the Belgium Traffic Sign Dataset. The traffic signs are classified into 13 classes based on their shape and color.This also includes an 'unlabeled' class, which does not fall under any category.

These classes include an undefined class, triangle signs, diamonds, red circle signs, vertical rectangles blue circle signs, forbidden signs, red and blue circle, reverse triangles ,horizontal rectangles,stop signs, squares and other traffic signs.

The dataset was generated by capturing images from multiple view points in urban street scenes. The amount of training, validation and testing images were 5905, 1301 and 1750 respectively. These images may have multiple traffic signs or even no traffic signs. It can be observed that these traffic signs occupy a small part of the image and are also prone to variations in lighting, orientation or occlusions. Before, the images are fed into the network, they are resized to the dimensions of 448×448.

The BTSD annotations provides labels and their corresponding bounding boxes. The bounding box labels consist of the following: x-coordinate, y-coordinate, width and height and the class of traffic sign,

The dataset well represents some difficulties associated with real traffic scenes due to illumination changes, small size and cluttered background as shown in Figure 1.

3.3 Loss Function

YOLO uses one single loss function for determining bounding boxes as well as classification of the objects. The loss function is given by :

$$Loss = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i{}^2) \tag{1}$$

$$+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \tag{2}$$

$$+\sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (C_i - \hat{C}_i)^2 \tag{3}$$

$$+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} (C_i - \hat{C}_i)^2 \tag{4}$$

$$+\sum_{i=0}^{S^2} 1_{i}^{obj} \sum_{c \epsilon classes} (p_i(c) - \hat{p}_i(c))^2 \tag{5}$$

Parts (1) and (2) are concerned with the loss of the bounding box predictions. The square root of the dimensions are used in this classification so that small deviations in large boxes incur less of a penalty when compared to such deviations for smaller bounding boxes. Part (3) and (4) are concerned with the loss of classification. 1_{ij}^{obj} computes the loss in situations when an object is present and 1_{ij}^{noobj} penalizes the confidence of object identification when there is no object. Part (5) deals with the confidence prediction loss. $p_i(c)$ gives the confidence of the ground truth box, which is nothing but the IOU of the ground truth box with the predicted bounding box. $\hat{p}_i(c)$ gives the predicted confidence of the neural network.

3.4 Evaluation Metric - Mean Average Precision (mAP)

Mean Average Precision is a metric used to evaluate object detection and classification models. To calculate the mAP for a set of detections, the interpolated average precision is calculated for each class, and a mean is calculated over it. For each class, the Average Precision is calculated using the area under the PR(Precision-Recall) curve for the predictions. The PR curve is constructed by associating each detection to its most overlapping ground truth object instance. Detections whose IOU with the ground truth above the threshold are considered as True Positives while the others are said to be False PositivesNext, two metrics called Precision and Recall are calculated as follows.

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{\text{No. of Ground Truth Boxes}}$$

The IOU or Intersection over Union of predicted bounding box B_p and ground truth bounding box B_{gt} is defined as follows.

$$IOU = \frac{\text{Area of overlap of } B_p \text{ and } B_{gt}}{\text{Area of union of } B_p \text{ and } B_{gt}}$$

The average precision(AP) is computed by averaging the precision values on the Precision Recall curve where recall is in the range [0, 0.1 ... 1].

$$AP = \frac{1}{11} \sum_{r \epsilon \{0,0.1,..,1\}} p_{\text{interp}}(r)$$

The precision at each recall level r is interpolated by determining the maximum precision measured for a method for which the corresponding recall value exceeds r.

$$p_{inter}(r) = max_{p:p \geq r} \text{ p}(\tilde{r})$$

where p($\tilde{r}$) is the measured precision at recall $\tilde{r}$.

Now the mAP is calculated as follows.

$$mAP = \frac{\sum_{\text{i } \epsilon \text{ classes}} AP_i}{\text{Total no. of classes}}$$

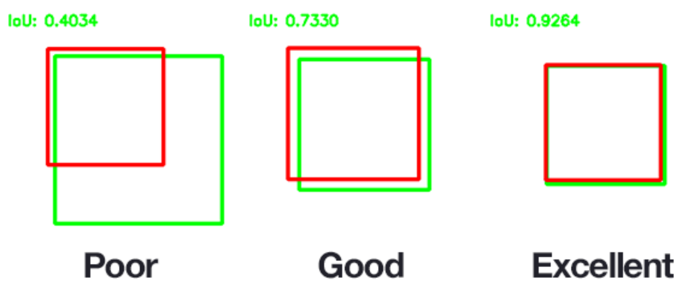


Figure 2: Comparison of different IOUs

4 Work Done

This section describes the work done by us during the course of the project. We first built the existing YOLO architecture and used the pre-trained weights. This was evaluated using the above explained mAP metric for a sanity check to ensure that the implementation does not have bugs. We have also completed the code for loading BTSD, defining the model and architecture for the BTSD dataset and the code for training. We trained this model on BTSD and have shown the training graphs obtained in the further subsections. We have also trained the model using the modified loss function and the results have been encouraging.

4.1 Experimental Framework

We are using the TensorFlow deep learning framework for the implementation. It is an open source software library originally developed by the Google for numerical computation using data-flow graphs. A graph is a data structure that describes the complete computation to be performed. In this framework, mathematical operations are represented by nodes, and edges represent data that is transferred between nodes. Data in TensorFlow are represented as tensors. Tensors are nothing but multidimensional arrays. Even though this framework is useful in various fields owing to its fast numerical computation, it is primarily being used in deep learning research, in both industry and academia.

The experiments were conducted on a system with 56 Intel Xeon CPU E52680 cores and Nvidia Tesla M40 GPU with 3072 Nvidia CUDA cores and 24 GB of GDDR5 video memory.

4.2 Modifying the Loss Function

It is observed that in the existing YOLO architecture, there is equal preference to both smaller and bigger objects. But it was observed that in BTSD, the traffic signs are usually small objects in images. Our objective being detection of traffic signs, the model must give more importance to detecting smaller objects accurately. So, we propose a modification to the existing loss function, which penalizes the model more if it does not detect a small image accurately, compared to larger objects. Intuitively, the model learns to detect smaller objects in images, thus complying with our objective.

Then 1 and 2 can be thus modified as:

$$\begin{aligned} Loss = (\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (x_i - \hat{x_i})^2 + (y_i - \hat{y_i}^2) \\ + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\hat{h_i}})^2) \times \frac{1}{\sqrt{h_i \times w_i}} \end{aligned} \tag{6}$$

When an object is not present in a particular grid, a factor of 10^5 is multiplied instead of the factor $\frac{1}{\sqrt{h_i \times w_i}}$.

4.3 Results and Analysis

This section describes the results we have got up till now.

4.3.1 Pascal VOC Dataset for a sanity check

As mentioned earlier, we first recreated the existing work on Pascal VOC 2007 dataset. The average precision obtained (area under the 11 point precision recall curve) for each class is shown in Table 1. The mean of all these values ie. mAP obtained was 54.7%

4.3.2 Training on BTSD

While training on the BTSD dataset, we visualized how the following losses are varying. Figure 3 shows the variation of class loss during training. The variation of coordinate loss is shown in Figure 4. The object and no object losses are represented in Figure 5 and Figure 6. The total loss is shown in Figure 7. It can be observed that the loss using the modified loss function is higher than the original loss. But this is due to the additional

Table 1: Result obtained on PASCAL VOC 2007

S.No	Class	Average Precision
1	bottle	0.134
2	horse	0.773
3	motorbike	0.616
4	sheep	0.473
5	sofa	0.463
6	dining table	0.415
7	person	0.503
8	dog	0.799
9	aeroplane	0.789
10	train	0.826
11	bicycle	0.579
12	chair	0.252
13	cow	0.558
14	boat	0.484
15	potted plant	0.254
16	car	0.518
17	cat	0.804
18	bird	0.534
19	tvmonitor	0.560
20	bus	0.607
	mAP	0.547

factor of $\frac{1}{\sqrt{h \times w}}$ that has been multiplied to the original loss. The important point to be noted here is that the modified loss is dipping sharply during training, which shows that this model is learning at a very fast rate.

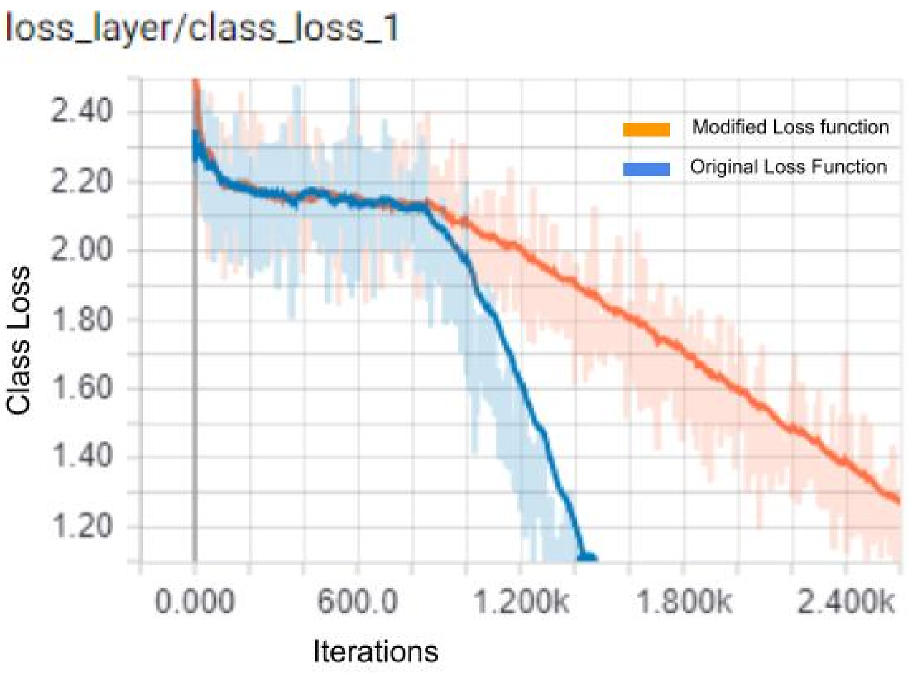


Figure 3: BTSD Class Loss

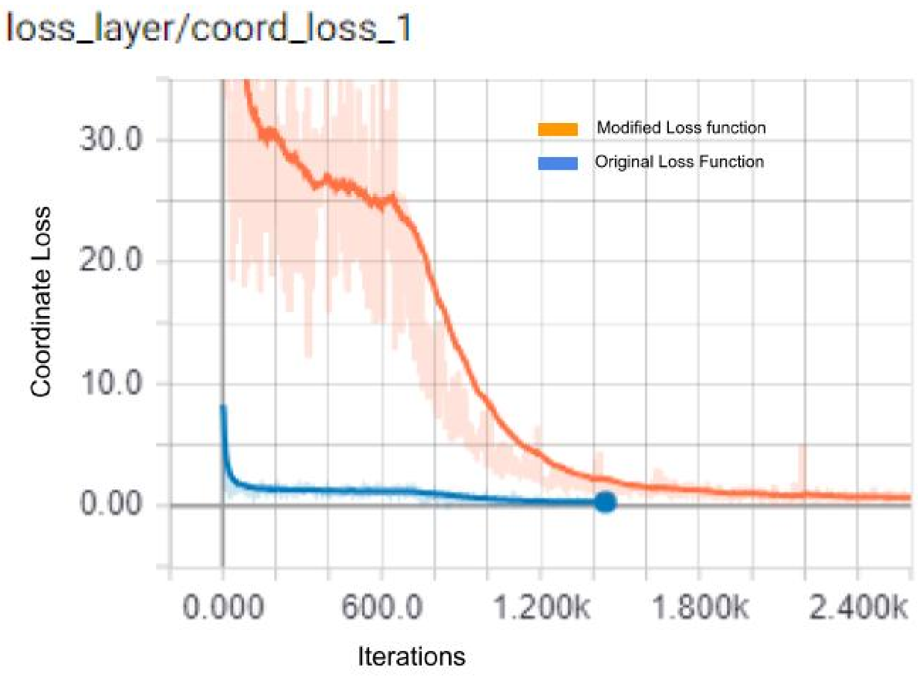


Figure 4: BTSD Coordinates Loss

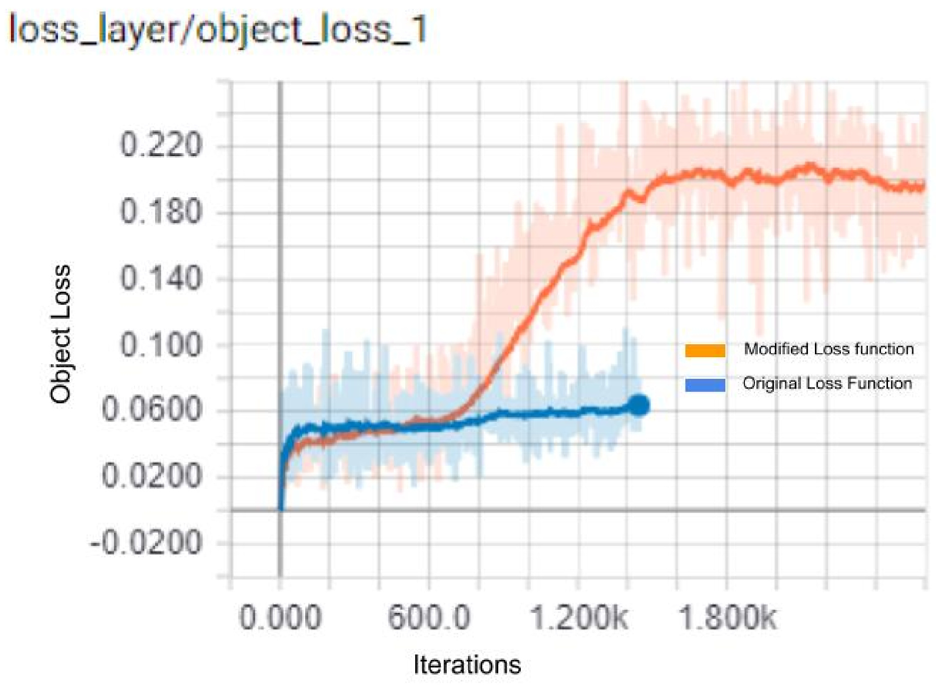


Figure 5: BTSD Object Confidence Loss

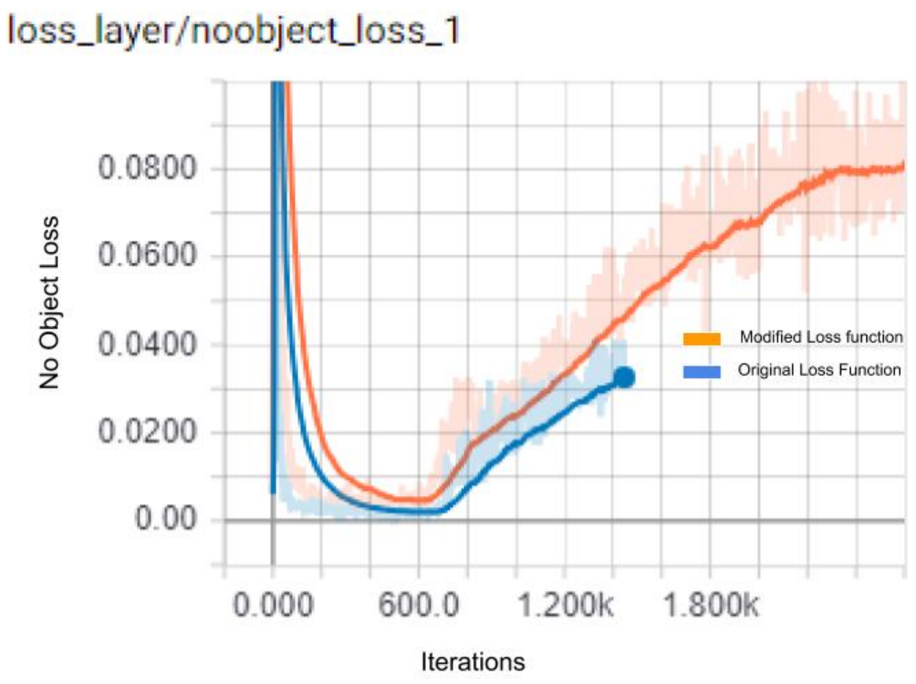


Figure 6: BTSD No object confidence loss

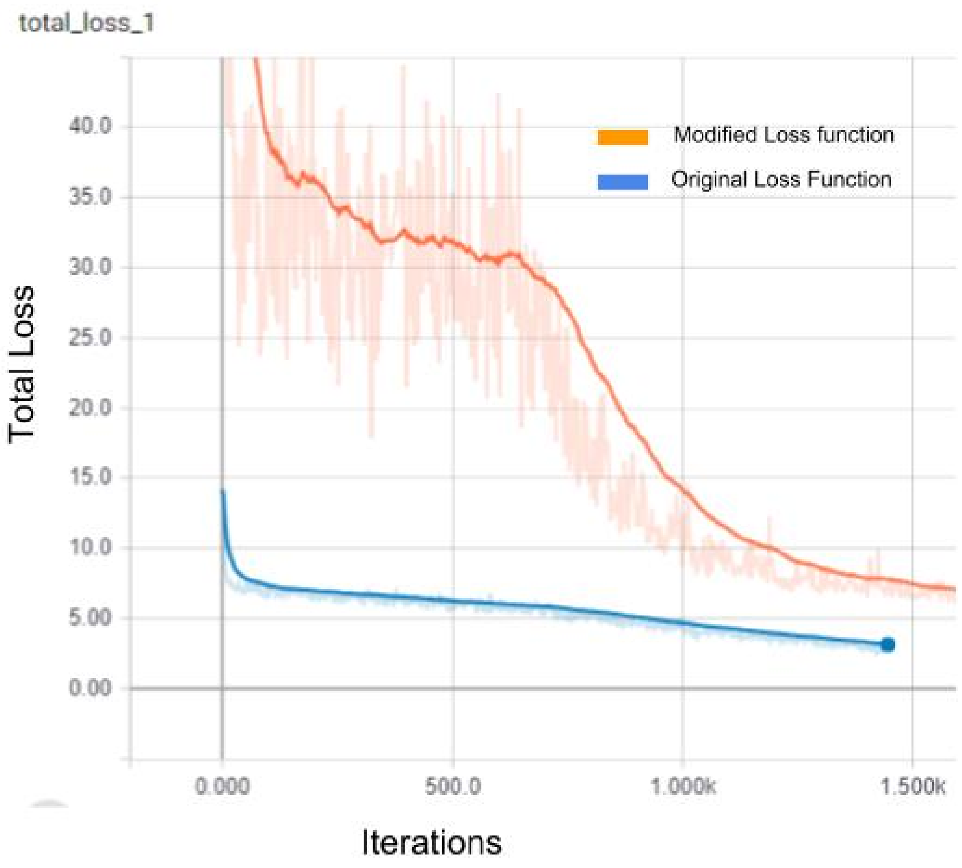


Figure 7: BTSD Total loss

1. Class Loss - This is the loss which accounts for the classification errors. It is given by equation 3 + equation 4
2. Coordinates Loss - This is the loss which accounts for the errors in detection. It is given by equation 2 + equation 3
3. Object confidence, No object confidence - These are the errors in the confidence when the object is present and not present respectively. They are given by equation 5. Ideally object confidence must equal the IOU with ground truth and no object confidence must be 0. Loss is calculated accordingly from these ideal values.
4. Total Loss - The sum of the above losses.

4.3.3 Results on Belgium Traffic Sign Dataset

We trained the modified YOLO model using both the standard loss function and our modified loss function. We evaluated the model using mAP as explained earlier. A sample of the original dataset was selected with equal number of annotations for each

Table 2: Average Precision for each class

Class Name	Standard loss function	Modified loss function
redbluecircle signs	0.393	0.673
redcircle	0.517	0.333
bluecircle signs	0.333	0.428
revtriangle signs	0.416	0.222
rectanglesdown signs	0.454	0.562
triangles	0.136	0.66
forbidden signs	0.449	0.25
diamond signs	0.126	0.666
rectanglesup signs	0.19	0.833
stop signs	0.095	0.25
other defined traffic signs	0.384	0.5
undefined	0.32	0.98
mAP	**0.332**	**0.525**

image. The Average Precision for both these methods are listed in Table 2. We obtained an mAP of 0.332 using the standard loss function and an mAP of 0.525 using our modified loss function. The same has been illustrated in Figure 8. The model trained using the modified loss function outperforms the standard model as evident from the evaluated mAPs of the models shown in Figure 9.

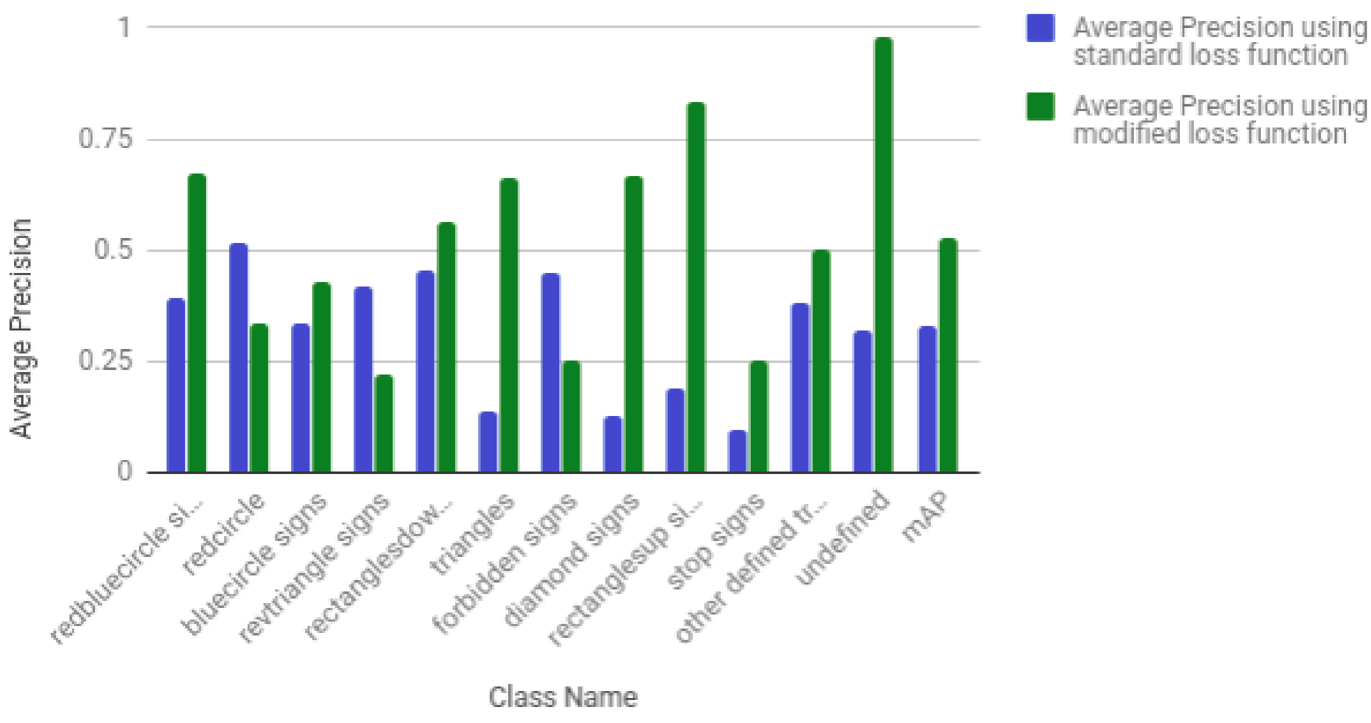


Figure 8: Class-wise comparison of average precision

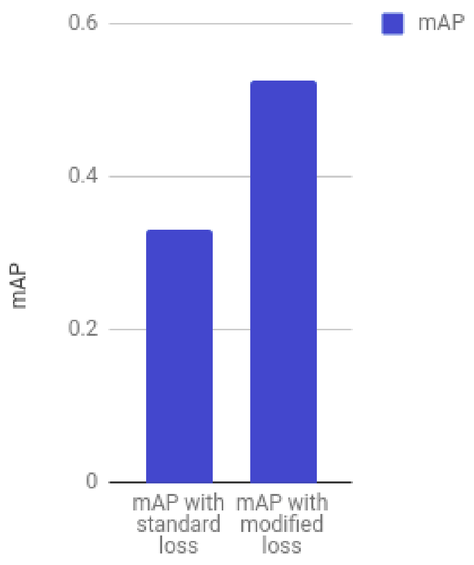


Figure 9: Comparison of mAP

4.3.4 Sample Detections

Figure 10 shows a few sample detections. The class predicted has been mentioned in the captions.

4.4 Individual Contributions

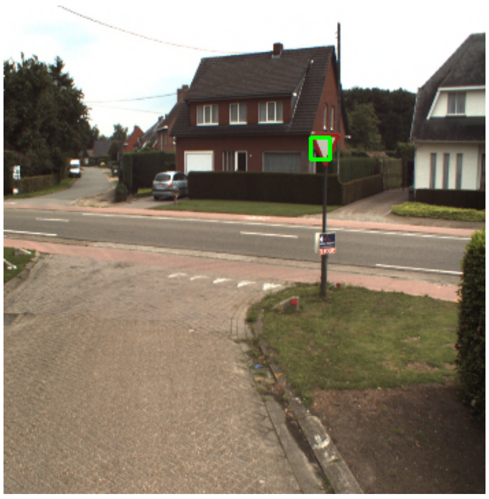
5 Conclusion and Future Work

In this work, we recreated the existing YOLO work on the PASCAL VOC and evaluated it. Later we worked on BTSD and trained a deep convolutional YOLO model to detect and classify the 13 classes of BTSD as shown in previous sections. We modified the existing standard loss function of YOLO to suit our objectives as elaborated previously. We evaluated both the models and obtained encouraging results for the same. We concluded that the model trained using modified loss function performed better than the standard YOLO model. Exploiting the natural constraints of our problem statement that the traffic signs are usually small in the images, we have thus defined a custom loss function to achieve the objective.

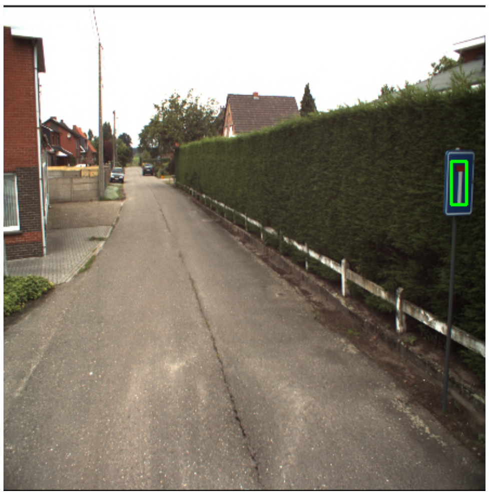
Going ahead, we would like to explore a few modifications to the existing work to improve the performance as explained in the next section.

(a) Detecting the stop sign

(b) Detecting triangle sign.



(c) Detecting the inverted triangle sign



(d) Detecting rectanglesup sign

Figure 10: Detection on sample images

5.1 Passing through autoencoder before YOLO

We could use autoencoders as attention networks before passing the input images through the YOLO model. Autoencoders are unsupervised neural networks, whose objective is to learn a set of weights, with target value as the input itself. It can be used to learn structures in inputs. This property of autoencoders can be useful to us as we can train this network to learn and recognize just the traffic signs. The autoencoder network can be trained with the cropped images of traffic signs. The entire image with the background can be passed through this trained autoencoder and hope that it would give a differential response towards the traffic signs when compared to other objects in the background. This would help in recognizing the required objects and leave out the others (eg. cars in the background etc.). This output of the autoencoder could be given as the input to the YOLO network, instead of just the image.

References

[1] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool, "Traffic sign recognitionhow far are we from the solution?" in *Neural Networks (IJCNN), The 2013 International Joint Conference on.* IEEE, 2013, pp. 1–8.

[2] J. Zhang, Q. Huang, H. Wu, and Y. Liu, "A shallow network with combined pooling for fast traffic sign recognition," *Information*, vol. 8, no. 2, p. 45, 2017.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once - unified, real-time object detection," in *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA*, 2016, pp. 779–788.

[4] R. Timofte, K. Zimmermann, and L. Van Gool, "Multi-view traffic sign detection, recognition, and 3d localisation," *Machine vision and applications*, vol. 25, no. 3, pp. 633–647, 2014.

[5] G. Wang, G. Ren, and T. Quan, "A traffic sign detection method with high accuracy and efficiency," in *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE), Hangzhou, China*, 2013, pp. 22–23.

[6] G. Wang, G. Ren, and T. F. Quan, "A trafc sign detection method with high accuracy and efciency," in *In Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE), Hangzhou, China*, 2013, pp. 1426–1429.

[7] G. Wang, G. Ren, Z. Wu, Y. Zhao, and L. Jiang, “A robust, coarse-to-ne trafc sign detection method,” in *In Proceedings of the 2013 International Joint Conference on Neural Networks, Dallas, TX, USA*, 2013, pp. 754–758.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA,*, 2014, pp. 580–587.

[9] J. Redmon and A. Farhadi, “YOLO9000: better, faster, stronger,” *CoRR*, vol. abs/1612.08242, 2016.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Y. Fu, and A. C. Berg, “SSD: single shot multibox detector,” *CoRR*, vol. abs/1512.02325, 2015.