HOCHSCHULE
RAVENSBURG-WEINGARTEN
UNIVERSITY
OF APPLIED SCIENCES

A project report on

Traffic Sign Detection and Recognition Project

Project Group:-

1. **Omkar Ambar Chinchkar (32912)**
2. **Prajakta Jambuvant Ghodake (32916)**
3. **Gaurav Vijay Kakade (32962)**

Guided By:- Dr. rer. nat. Professor Stefan Elser

Date:- 10/07/2020

Abstract

Traffic sign Detection and Recogition is a crucial task in the Autonomous driving and ADAS Applications. Lot of work is being done in this field nowadays. There are various Datasets available for this task like the German Traffic Sign Recognition Benchmark (GTSRB), the German Traffic Sign Detection Benchmark (GTSDB), LISA Traffic Sign Dataset which contains the US Traffic Signs, KUL BelgiumTS Dataset and the Swedish Traffic Sign dataset (STS). For this Project, we used the GTSRB and GTSDB Datasets. The purpose of this project was to evaluate the Recognition and Detection models. The Evaluations were done seperately for Detection and Recognition, as well as both combined.

Contents

List of Figures

Acronyms

AP Average Precision.

CNN Convolutional Neural Network.

FN False Negative.

FP False positive.

IOU Intersection Over Union.

mAP Mean Average Precision.

R-FCN Region based Fully Convolutional Network.

TN True Negative.

TP True Positive.

1 Introduction

1.1 The German Traffic Sign Detection and Recognition Benchmark

The German Traffic Sign Detection Benchmark (GTSDB) was developed for research in the field of computer vision, pattern recognition and driving assistance. The dataset features single-image detection problem. It has 900 total images, which are further divided into 600 training images and 300 testing images. A single Image in this dataset may contain zero to six traffic signs. The Images in this dataset are in PPM format. The traffic sign sizes in the images differ in the range 16x16 to 128x128. Traffic signs in this Dataset can be visible in all viewpoints and under all lighting conditions [HSS^{+}13].

The German Traffic Sign Recognition Benchmark (GTSRB) features a Single-image, multi-class classification problem. It has more than 40 classes and more than 50,000 images in total. Each Image in this dataset contains one traffic sign. The Images in this dataset are in the PPM format. The size of the Images differs from 15x15 to 250x250 pixels. The Images within this dataset have a border of 10 percent around the actual traffic sign to permit edge-based approaches. Not all Images in this dataset are square and also not all Traffic signs within the Images at the center of the image [SSSI11].

1.2 Pre-trained R-FCN network for traffic sign detection

Objective of task 1 was to detect the traffic signs present in an image. This was done using the pretrained rfcn_resnet101 network. The author of paper "Evaluation of deep neural networks for traffic sign detection systems" Álvaro Arcos-García provided various evaluation results as well as pretrained networks. There was a github link provided in this paper, which was used for reference. The objective behind selecting the R-FCN network over other networks was that this network showed better results in terms of accuracy and time. [AGAGSM18]

R-FCN network is regional based fully convolutional network. This network is used for object detection. Performance of this network is better than the state of the art Faster R-CNN network. As mentioned in the paper by Kaiming He, the results of R-FCN compared with Faster R-CNN showed that testing was quick compare to later one. Even the mAP results obtained were better compared to Faster R-CNN. [DLHS16]

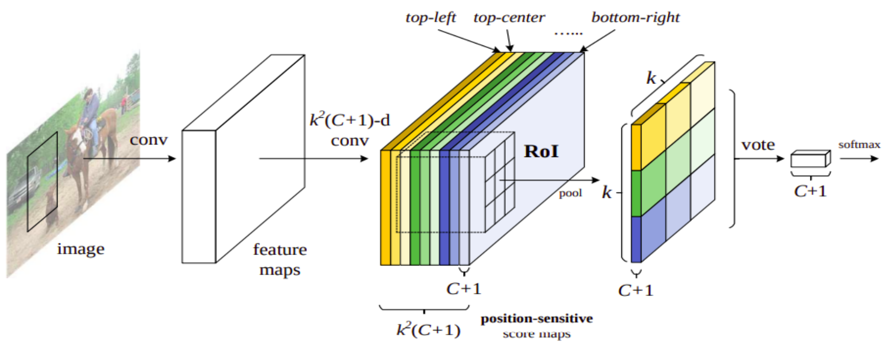

Figure 1.1: Overall architecture of R-FCN for object detection [DLHS16]

The figure 1.1, shows the main idea behind the R-FCN for object detection. Image is divided into 3*3 regions. These RoI are calculated over the image with the class scores. Then this is passed to position sensitive score maps. Based on the score of RoI, the object of interest is predicted.

1.3 Traffic sign Recognition using Tensorflow Keras

Autonomous driving is no longer confined to the works of science fiction. There are already vehicles on the road today with Advanced Driver Assistance Systems (ADAS) that maintain speed, brake, and plan with limited or no driver engagement. Building on these advancements, there was a necessity to detect and correctly recognise traffic signs on the road. There has been a lot of work done in this particular subject. In the previous work, a real-time traffic sign recognition algorithm robust to identify all traffic sign categories was built. It was a two-level detection framework which consists of the region proposal module responsible for locating the objects and the classification module which aimed to classify the located objects [WLC$^+$20].

In this part of the project, similar protocol was followed but for recognition of the traffic signs a classifier was build and trained rather than using a pre-trained model. German Traffic Sign Recognition Benchmark (GTSRB) was used as a training and testing dataset.

In this work, Tensorflow 2.0 and Keras deep learning framework was used to implement a Convolutional Neural Network (CNN) for traffic sign recognition. Keras and TensorFlow 2.0 provides three methods to implement a neural network architecture like Sequential API, Functional API and Model subclassing. In this work, the Keras sequential API method was used to build a CNN model [Ros19]. The reason behind choosing this method was that it is the simplest of all three. But it also have some pitfalls like the created model cannot share layers or cannot have branches. The model was further trained on the GTSRB training dataset.

2 Traffic Sign Detection (Task 1)

2.1 Implementing the model on sample Image

Detecting specific objects in an image is one of the task in the field of computer vision and deep neural networks. The object which is required to be detected in an image is considered while training the neural network. Once trained neural network is fully functioning, testing of neural network becomes major task. For this purpose testing images from dataset are considered. In this project, R-FCN pre trained network was selected for traffic sign detection. This network was pre-trained using GTSDB training images. The scope of this project was to test this pre-trained network on testing images from GTSDB dataset. If one image from this testing dataset is passsed to network it would predict the presence of traffic sign in that image and draw bounding box around it. The figures 2.1 and 2.2, should help in understanding the functioning of neural network on new testing image. The figure 2.1 is full scene image, in which traffic sign was predicted by network. The figure 2.2 is enlarged image to understand the bounding box features in prediction image. The green colored bounding box represents the predicted box by R-FCN network while purple bounding box represents ground truth provided in dataset label file.

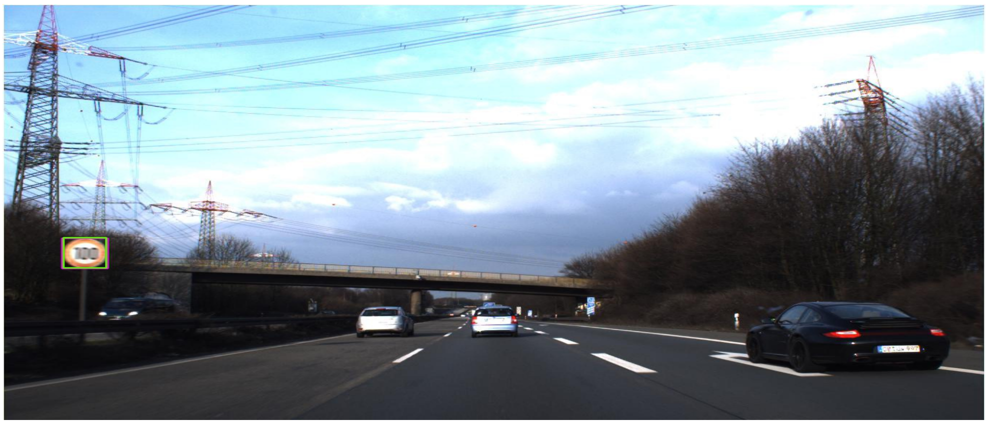
Figure 2.1: Detected traffic sign using R-FCN network

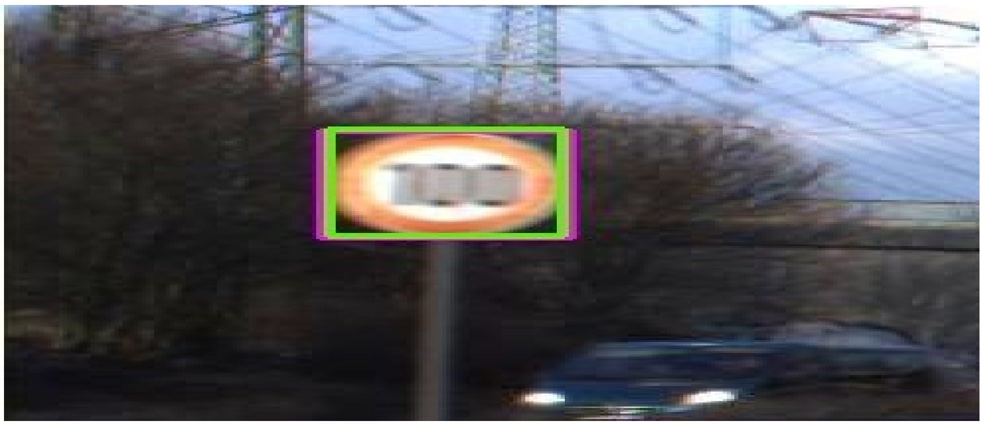

Figure 2.2: Groundtruth and Predicted bounding boxes

In figures 2.1 and 2.2, it could be seen that the ground truth and predicted bounding box were matching quite well. This was very rare scenario in object detection where predicted bounding box was matching exactly with ground truth bounding box. Sometimes the network showed predicted traffic sign bounding box at different position than one provided by ground truth. Sometimes there was no match at all in between predicted bounding box and ground truth bounding box. To understand this behaviour of network for different images and to validate the full functioning of network for future testing images. It was crucial to evaluate the network using some kind of metrics. In this project Intersection over Union metric was selected as evaluation metric.

To evaluate any network, you need predicted data for whole testing dataset. This data could be obtained by processing the testing image on pre-trained network. In this project, testing data from GTSDB dataset was processed using pre-trained model. The results of each image was stored in a text file with details of bounding boxes and image id. This was done to decrease the computational time of evaluation. Sample image data **"601.jpg;84;449;143;507"** shows, how it was stored in prediction text file. Predicted bounding box data was stored in string format in text file. Each parameter of this string was separated by string separator semi-colon (;). First part of string was image id, second to fourth part of string was Xmin, Ymin, Xmax, Ymax co-ordinates of predicted box respectively. After processing each image from testing dataset results were appended in text file. Completed text file was then passed on to evaluation code.

2.2 Evaluation Metric

As discussed earlier, evaluation metric used for evaluating traffic sign detection network was Intersection over Union. It is also known as Jaccard Index. Intersection over Union is one of the method of calculating the true predictions and false predictions. In this method, area of each predicted box and ground truth box were considered. Then the intersection area between predicted and ground truth bounding box was calculated. After calculating the intersection area of these two bounding box, ratio of this calculated area was taken over Union of area between predicted bounding box and ground truth bounding box. This whole value was nothing but intersection over union. Following formula should help in understanding the concept of intersection over union in detail.

$$IoU = \frac{Intersection\ area\ of\ Overlap}{Union\ area\ of\ bounding\ boxes}$$

Generally if calculated IoU value is greater than 0.9, then it is considered to be excellent prediction. If this value is in the range 0.7-0.9 it is considered to be good prediction. If this value is less than 0.5 it is considered to be poor prediction. Depending upon value of IoU, predicted box was considered either true positive or false positive. In following section, implementation of this metric is described.

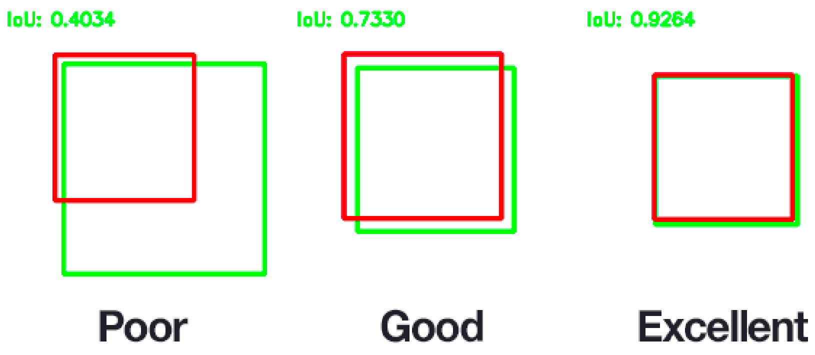

Figure 2.3: Prediction Classification depending on IOU values [Ros16]

2.3 Processing predicted and ground truth text files for evaluation

Following pseudo code explains how predicted and ground truth files are read and processed for further use in evaluation:

```
Initialize an empty list for predicted text file entries and ground truth text file entries
Open predicted text file and ground truth text file using python file
Use readline() method of python file to read each line from file. Store this result into empty lists created earlier
Access this list once above operation is completed. Each element of list is in String form
Convert this string into integer form for further processing, along with this remove ".jpg" extension from the image id
Once both predicted and ground truth lists are prepared. Check if it is in intended form. If yes then move on to next steps
```

2.4 Implementing Evaluation Metric

Following pseudo code shows how implementation of evaluation metric was conducted:

```
Create a intersection function. Input to this function will be co-ordinates of predicted and ground truth box

    Compare Xmin, Xmax, Ymin, Ymax co-ordinates of predicted box and ground truth box.
    Maximum value between the two is considered as ixmin, iymin, ixmax, iymax
    Calculate the height and width of intersection.
    Width can be obtained by subtracting ixmax and ixmin.
    Height can be obtained by subtracting iymin and iymax.
    Compare this subtraction results with 0 and take maximum of this two as height and width
    Compute the area of intersection area
    area of intersection = height*width
    Return intersection area to calling function

Create a union function. Input this function will be co-ordinates of predicted box, ground truth box and intersection area calculated in intersection function

    Calculate the area of predicted box and ground truth box
    Calculate union area
    Union area = area of predicted + area of ground truth - intersection area
    Return union area to calling function
```

```
Create compute intersection over union function. Input to this function will be ground truth list,
single entry from prediction list, result matrix

    Loop Ground truth list
    Compare image id from ground truth list with image id of prediction box
    If ground truth image id and predicted image id match is found.
        Call intersection function by passing required parameters
        Call union function by passing required parameters
        Compute intersection over union using results obtained from previous steps
        If the obtained iou result is more than 0.7
            Store results in result matrix. First entry in result matrix is image id.
            Second entry is 1 (TP). Third entry is 0 (FP)
        Else
            Store results in result matrix. First entry in result matrix is image id.
            Second entry is 0 (TP). Third entry is 1 (FP)

Call compute IoU function by passing required parameters. Here prediction list should be looped
to pass single entries from prediction list to this function
```

2.5 Calculating Precision, Recall and computing Average precision

2.5.1 Precision and Recall

Precision and Recall are two important terms in evaluation of any dataset on model. Precision is nothing but portion of total prediction which is accurate. Recall is portion of total positives over entire number of objects present in ground truth.

Following are formulae for Precision and Recall:

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

$$Recall = \frac{TP}{TP + FN}$$

where TP = True Positive, FP = False Positive, FN = False Negative

After obtaining result matrix with Image id, TP, FP values. Proceed to compute the cumulative values of TP and FP. Following pseudo code explains how precision, recall was calculated in this project:

```
Cumulative values of TP and FP can be calculated using the numpy.cumsum function.
Append the result of this to result matrix.
Compute precision and recall for each row of result matrix
Append the results in same matrix with addition of new columns.
Note: Use standard formula for calculating precision and recall values.
```

2.5.2 Average precision

Average precision shows the ability of network to accurately predict the objects present in an image. This is generally calculated over the results of whole testing dataset. Precision and Recall values obtained in previous calculations are used to calculate average precision. Graph of precision and recall is plotted. Area under this curve is nothing but the average precision. There are two ways of calculating average precision. First one is by using 11 point interpolation method to compute average precision. Second one is all point interpolation method for computing average precision. Following pseudo code explains how average precision was calculated in this project:

```
Define a function for calculating 11 point interpolation. Input to function is precision, recall list
and interpolation point.
        Check the greater value between interpolation point and recall value.
        Iterate recall values for comparison.
        Pass this value as greater recall to next step.
        Return precision value corresponding to greater recall which was obtained in previous step.
        Create list of returned precision values for each interpolation point $(precision_interpolated_list)$.
        Plot graph of precision and recall.
        Plot graph of interpolation points and precision_interpolated_list
        $CalculateAP\_11\_point_interpolation$
        $AP11 = (sumofprecision\_interpolated\_list)/11$
        Return $AP11$

  Define a function for all point interpolation. Input to this function is precision, recall lists.
        Create a list of interpolation points using each element of recall list
        Create a list of precision interpolated list accessing each element precision list
        Iterate through precision interpolated list to check each element with its next element.
        Plot graph of precision and recall
        Plot graph of interpol_points and precision_interpolated_lists
        Calculate average precision by iterating through the precision interpolated list.
        If subtraction between interpol_point at I and i-1 is not zero
           AP += (interpol_points[i] - interpol_points[i - 1]) * precision_interpolated_list[i]
        Return final AP value to calling function
```

3 Traffic Sign Recognition (Task 2)

In this work, the main aim was recognizing traffic signs in an image, for this purpose tensor flow deep learning framework was used. The recognition task was implemented using training and testing dataset of GTSRB. There are 43 classes in the training dataset which contains 39209 samples. The testing dataset contains 12628 testing samples.

3.1 Implementation

Task 1: Implementing a Convolutional Neural Network using keras sequential API library.

- The main parameters for building a network were image dimensions and depth of the image. To have more features a 5 x 5 kernel was used that helped distinguish between different traffic sign shapes and colors in the dataset. The Network consists of two fully connected layers and a Softmax classifier. Also, to regularize or to prevent overfitting dropout was used.

Task 2: Training the Classifier on GTSRB dataset.

- In this task, the GTSRB dataset was split into training and testing datasets. The training dataset contains 39209 samples and testing dataset contains 12628 samples. Before training the model with the images of the training dataset these images were pre-processed that includes basic filtering, mathematical morphology, regions properties using skimage library. The model was trained and the ground truth information was extracted that includes the image dimensions (height, width and co-ordinates) with image-id (signifies the class of the image).

Task 3: Testing and Evaluation

- The classifier was tested and evaluated on the testing dataset using F1 Score as an evaluation metric.

The figure 3.1 shows the traffic sign is recognized and classified in the 'Keep right' and 'Speed limit (30Km/h)' classes respectively. It is seen that the classifier model has correctly classified and recognized these images giving a True Positive for the particular above classes.

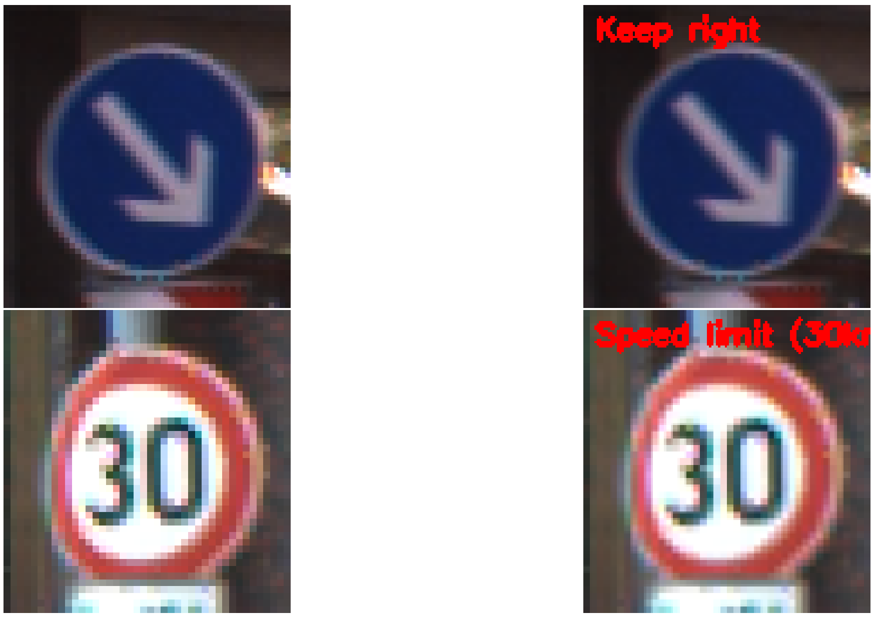

Figure 3.1: Recognition and Classification of Traffic Signs

3.2 Evaluation Metric

3.2.1 F1-Score

In the recognition task, F1 Score is used to measure the accuracy of the model. It is different from the accuracy metric as it gives the harmonic mean of the precision and recall. The following is the formula to calculate F1 Score:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The value of F1 Score is 1 when both precision and recall is 1 and vice-versa [fsc19].

3.2.2 Precision and Recall

Precision and Recall is calculated with total number of TP, FP and FN. Precision refers to the percentage of the results that are relevant and recall refers to the percentage of all relevant results correctly detected by the algorithm. Precision and Recall formulae are same as mentioned in section 2.5.1

3.3 Implementing Evaluation Metric

Pseudocode for prediction and evaluation of traffic sign recognition model:

```
Import packages and modules
Parse command line arguments
Print "loading model"
load the model
open, read and split csv files
Print "Start Predicting"
loop over i_path
    define the path for images and re-iterate
FOR every image in a path
    read the image
    resize the image
    preprocess the image
    prediction using CNN
    Create a list
load the image, resize and draw the label
save the image to the disk
Print "Evaluation of the network"
For image in list
    If predicted image have same class id as groundtruth images
        Counted as a True Positive
    Else
        Counted as a False Positive
Generate the classification report
Print "Classification report"
```

4 Processing Detection Images on Recognition model (Task 3)

4.1 Processing GTSDB Dataset Images using ROI

The Traffic signs detected in the Detection task were cropped and further used on the Recognition model. The Images in the Dataset were in the PPM format. These Images were converted into JPG format to enable their cropping. Following is the Pseudocode for converting the PPM Images to JPG Images:

```
Create a For loop with range equal to the number of Images.
        Create a variable which stores the Image number.
        Create a string which stores the name of extension .ppm.
        Create a string which stores the file path until the parent directory of the Images.
        Use the OS module to join the path.
        Open the Image with the Image module of PIL library and the joined path.
        Create a string which stores the name of extension .jpg.
        Use the OS module to join the file path to save the converted Images.
        Save the Image with the Image module of PIL library to the joined path.
```

The Bounding box co-ordinates obtained from the Detection task were used to crop the Image. These Bounding box co-ordinates were obtained from the Detection task in the form of a Text file. Following is the Pseudocode for cropping the Images from the dataset according to the bounding box co-ordinates:

```
Create an Empty list named List1.
Open the Text file and assign it to a variable.
Create a While loop which runs until the conditions inside are True.

        Create another variable.
        Store the data of single line from Text file in the variable.
        Store and append the data from the variable in the List1 created before.

Create an Empty list named List2.
Create a For loop of range equal to length of List1.

        Create a variable.
        Split the data from the List1 according to semi-colons ';' in it and store it in this variable.
        Store and append the data from this variable into the list2.

Create a For loop of range equal to the length of list2.

        Open the Images one by one using the Image module of PIL library.
        Create variable named Ymin and assign the value of first element of list2.
        Create variable named Ymax and assign the value of second element of list2.
        Create variable named Xmin and assign the value of third element of list2.
        Create variable named Xmax and assign the value of fourth element of list2.
        Crop the Image using the Image module of PIL Library according to dimensions.
        Save the cropped Image with .png extension by providing path using OS module of Python.
```

The cropped Images were then sent to the Recognition model. As the Recognition task was on a another machine, the cropped images were transferred with the help of a online data transfer site 'WeTransfer'.

4.2 Using the cropped Images on Recognition model

In this task, the same classifier model used in task 2 of traffic sign recognition was tested and evaluated using a testing dataset. The testing dataset had the detected images of traffic sign detection task which were cropped in order to have a better classification result using the classifier used in recognition task. In this part of the project, similar steps as task 2 were followed. The model was evaluated using F1 Score and Accuracy

Figure 4.1: Recognition and Classification of cropped Traffic Signs

The figure 4.1 shows the recognition and classification of cropped image that was detected in the task 1.

5 Results and Discussions

5.1 Results of Detection task:

- **Results when IoU>0.5**
  - Average Precision using 11 point interpolation was 0.67 or 67 percent.(Fig.5.1)
  - Average Precision using all point interpolation was 0.70 or 70 percent.(Fig.5.2)
- **Results when IoU>0.7**
  - Average Precision using 11 point interpolation was 0.65 or 65 percent.(Fig.5.3)
  - Average Precision using all point interpolation was 0.66 or 66 percent.(Fig.5.4)

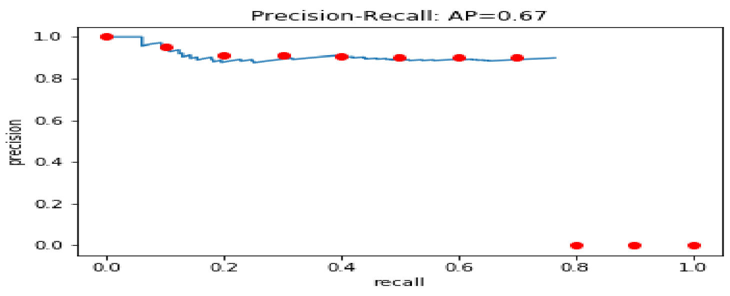

Figure 5.1: 11 point interpolation for IOU > 0.5

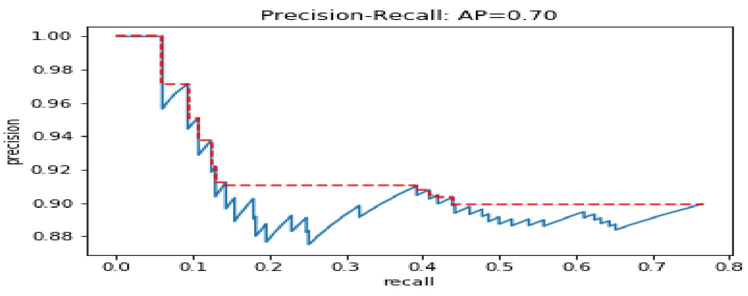

Figure 5.2: All point interpolation for IOU > 0.5

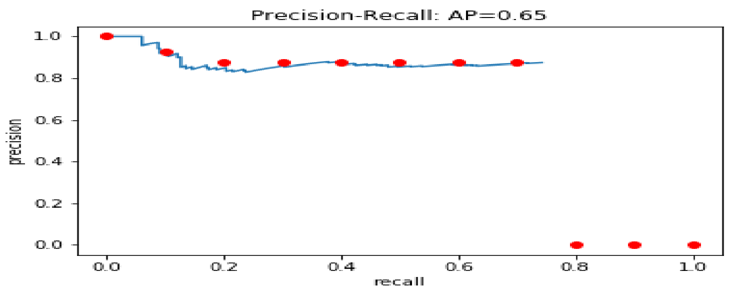

Figure 5.3: 11 point interpolation for IOU > 0.7

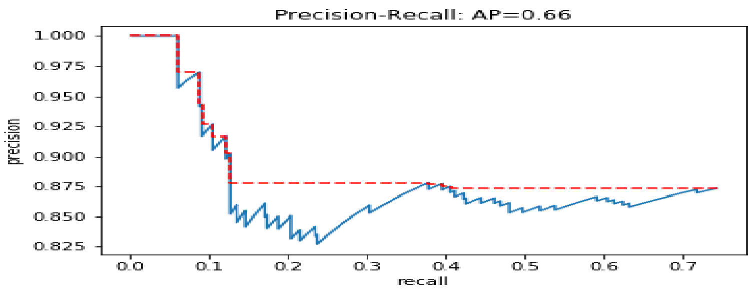

Figure 5.4: All point interpolation for IOU > 0.7

5.2 Result of Recognition task

The testing dataset of GTSRB dataset consisted of 43 classes. For evaluating the classifier using F1-Score, the Precision and Recall values for every class were calculated. Since, there were 43 different classes in the testing dataset, 43 different Precision and Recall values were obtained. Also, the F1-score for every individual class was calculated by taking the harmonic mean of Precision and Recall values. To have a overall recognition accuracy of the classifier, the F1-Score was calculated which was obtained by taking the harmonic mean of the Precision and Recall values.

The F1-Score for this classifier was obtained as **88%**.

The table 5.1 shows Precision, Recall and F1-score values for some of the classes in the testing dataset.

Classes	Precision	Recall	F1-Score
Speed limit (20Km/h)	1.00	0.50	0.67
Speed limit (30Km/h)	0.97	0.94	0.96
Speed limit (50Km/h)	0.99	0.93	0.96
Speed limit (120Km/h)	0.74	0.92	0.82
No Passing	0.99	0.79	0.88
Dangerous curve to the right	0.53	0.96	0.69
Pedestrians	0.56	0.50	0.53
Children crossing	0.77	0.97	0.86
Bumpy road	1.00	0.70	0.82
Slippery road	0.63	0.83	0.72
Road work	0.98	0.94	0.96
No vehicle	0.99	0.56	0.72
No entry	0.99	0.90	0.94
Keep right	0.92	0.99	0.95

Table 5.1: Recognition report

5.3 Result of Recognition of Images from the Detection task

The evaluation metric used in this task was F1-Score. Following are the results:

- The F1 Score for the Task 3 was found out to be 69 percent.
- The Accuracy for the Task 3 was found out to be 52.75 percent.
- The Precision for the Task 3 was found out to be 55 percent.
- The Recall for the Task 3 was found out to be 90 percent.

6 Conclusion

Traffic sign detection and recognition are two separate tasks which are beneficial for ADAS systems in current world. This project performed the evaluation of detection and recognition model separately as well as together. Detection results were obtained on the GTSDB testing dataset containing 300 images. A different dataset with more testing images can be used to evaluate the performance of a network.

Evaluation of traffic sign detection was accomplished in this project. Evaluation results obtained showed that pre-trained R-FCN network can be used for traffic sign detection where accuracy is not major deciding factor. Evaluation results varied for different values of IoU. Average precision varied by 4 percent when IoU value was changed from 0.5 to 0.7.

The Traffic Sign Recognition task was successfully completed and evaluated using F1-Score as an evaluation metric. The result of F1-Score showed that the classifier trained on GTSRB dataset could correctly recognise and classify the traffic signs to the respective classes of the dataset. However, for some of the classes like 'Pedestrians' and 'Dangerous curve to the right' the calculated F1-Score was less since these classes were unbalanced i.e. had less number of training samples. Hence, in order to have a better F1-Score class skewing should be avoided.

For the task 3, the images from the GTSDB dataset were cropped, which resulted in degradation of the Image quality. This ultimately resulted in a poor recognition. To avoid this, a model which does both Detection and Recognition should be built. To enhance the image quality, some pre-processing techniques can be used like gaussian blur, transformation and reduction. This can help to improve the training accuracy of the model to perform object recognition.

Bibliography

[AGAGSM18] Alvaro Arcos-Garcia, Juan A Alvarez-Garcia und Luis M Soria-Morillo: *Evaluation of deep neural networks for traffic sign detection systems*, *Neurocomputing*, Bd. 316:S. 332–344, 2018.

[DLHS16] Jifeng Dai, Yi Li, Kaiming He und Jian Sun: *R-fcn: Object detection via region-based fully convolutional networks*, in *Advances in neural information processing systems*, S. 379–387, 2016.

[fsc19] *F-Score*, `https://deepai.org/machine-learning-glossary-and-terms/f-score`, 2019, [Online; accessed 7-June-2020].

[HSS$^+$13] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing und Christian Igel: *Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark*, `http://benchmark.ini.rub.de/?section=gtsdb&subsection=news`, 2013, [Online; accessed 27-April-2020].

[Ros16] Adrian Rosebrock: *Intersection over Union (IoU) for object detection*, `https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/`, 2016, [Online; accessed 5-June-2020].

[Ros19] Adrian Rosebrock: *3 ways to create a Keras model with TensorFlow 2.0*, `https://www.pyimagesearch.com/2019/10/28/3-ways-to-create-a-keras-model-with-tensorflow-2-0-sequential-functi`
2019, [Online; accessed 6-June-2020].

[SSSI11] Johannes Stallkamp, Marc Schlipsing, Jan Salmen und Christian Igel: *The German Traffic Sign Recognition Benchmark: A multi-class classification competition*, `http://benchmark.ini.rub.de/?section=gtsrb&subsection=news`, 2011, [Online; accessed 27-April-2020].

[WLC$^+$20] Yiqiang Wu, Zhiyong Li, Ying Chen, Ke Nai und Jin Yuan: *Real-time traffic sign detection and classification towards real traffic scene*, *Multimedia Tools and Applications*, S. 1–19, 2020.