Project Report

for

Image Processing with Machine Learning

DA526

on

Traffic sign detection

Submitted By:

1. Abinash Kumar Ray(224101062)
2. Aviral Singh(224101010)
3. Abhishek Ranjan(224101003)
4. Amit Kumar(224101006)
5. Vivek Verma(224101059)

Submitted to:

Prof. Debanga Raj Neog

Index

Github reposetory link : Link

1.Problem Statement

Nearly everything we do in the modern world has automated tasks streamlined the process. Drivers frequently miss signs when driving because they try to concentrate on the road.they might endanger themselves and anyone around them by stopping on the side of the road. If there were a reliable means to alert the driver without requiring them to change their focus, this issue could be avoided. Not being able to interpret the sign's meaning is another frequent issue. Drivers may not even recognise the sign's meaning even if they do manage to see it. With the help of our application, the detected sign will be given to the driver in a convenient manner, ensuring that they are no longer hindered by the difficulty of deciphering what the sign is saying.

Our proposed Project **Traffic sign Detection and classification** is to develop a computer vision system that can accurately detect and recognize traffic signs from images or videos captured by a camera.
This system should be able to identify different types of traffic signs, such as stop signs, speed limit signs, and yield signs, and accurately classify them according to their meaning.

The system should also be able to detect multiple signs in a single image and provide accurate bounding boxes around them. The goal of this problem is to improve road safety by providing drivers with real-time information about the traffic signs they encounter, helping them to obey traffic laws and avoid accidents.

We suggest using the state-of-the-art object recognition and classification method known as the Region-based Convolutional Neural Network (R-CNN) algorithm to accomplish this purpose. R-CNN is made up of three primary parts: a region proposal network (RPN) for object recognition and pretrained model such as ResNet or VGG16 for feature extraction purpose and convolutional neural network for classification of traffic signals,the classification model i.e cnn followed by fully connected layer predicts the classification probability followed by bounding box regression which will be trained to find coordinates of bounding box around the traffic signals.

In order to identify prospective locations for traffic signs in the supplied image or video, our system will first produce region recommendations using the RPN. The region-based classifier will next use the features that the CNN has extracted from these regions to identify whether each region has a traffic sign and, if so, what kind of sign it is. In order to get rid of duplicate detections and improve the bounding box, we will additionally apply a non-maximum suppression approach.

The proposed project has been implemented on German Traffic Signs Database, an open source dataset and we have compared the working of our model with the recent works done in the field.
We have also investigated the effect of different hyperparameters, such as learning rate using the concept of annealing rate, batch size, and network architecture, on the performance of our system.

Overall, our proposed project aims to develop an accurate and efficient system for real-time traffic sign detection and classification using R-CNN + Resnet and compare it with recent works in the field.The project accuracy has been evaluated on average precision AP and intersection over union which evaluates how close the regressor is performing to the ground truth supplied.

By improving road safety and reducing the number of accidents caused by driver error, this project can have a significant impact on society.

2.Dataset

We use datasets of the **German Traffic Sign Detection Benchmark (GTSDB) dataset**.
There are multiple reasons for choosing this dataset over the others, including the fact that it is highly accepted and is widely used for comparing traffic sign detection approaches in the literature.
Signs are grouped in **43 categories**. The GTSDB dataset contains natural traffic scenes recorded in various types of roads (highway, rural, urban) during the daytime and at twilight, and numerous weather conditions are featured.
The images can contain one, multiple, or no signs at all, and are of varying clarity and contrast. Sign distances within the image are also not fixed.
This dataset is composed of **506 images containing 1206 traffic signs** that are split into a **training set of 406 images, testing set of 50 images and 50 images for validation**.
These are the classes, and we have sorted the images into their respective classes:
The dataset consists of various classes representing different traffic signs. These classes include

1. Speed limit (20 km/h): Indicates a maximum speed limit of 20 km/h in the area.
2. Speed limit (30 km/h): Indicates a maximum speed limit of 30 km/h in the area.
3. Speed limit (50 km/h): Indicates a maximum speed limit of 50 km/h in the area.
4. Speed limit (60 km/h): Indicates a maximum speed limit of 60 km/h in the area.
5. Speed limit (70 km/h): Indicates a maximum speed limit of 70 km/h in the area.
6. Speed limit (80 km/h): Indicates a maximum speed limit of 80 km/h in the area.
7. End of speed limit (80 km/h): Marks the end of a previously indicated maximum speed limit of 80 km/h.
8. Speed limit (100 km/h): Indicates a maximum speed limit of 100 km/h in the area.
9. Speed limit (120 km/h): Indicates a maximum speed limit of 120 km/h in the area.
10. No passing: Prohibits overtaking or passing other vehicles.
11. No passing for vehicles over 3.5 metric tons: Prohibits overtaking for vehicles weighing over 3.5 metric tons.
12. Right-of-way at the next intersection: Indicates priority at the upcoming intersection.
13. Priority road: Gives priority to vehicles on the specified road.

14. Yield: Instructs drivers to yield and give the right-of-way to other vehicles.
15. Stop: Requires drivers to come to a complete stop at the designated point.
16. No vehicles: Prohibits the entry of any vehicles.
17. Vehicles over 3.5 metric tons prohibited: Restricts vehicles weighing over 3.5 metric tons from entering.
18. No entry: Prohibits entry for all vehicles.
19. General caution: Warns drivers to exercise caution and be aware of potential hazards.
20. Dangerous curve to the left: Indicates an upcoming leftward curve that may be hazardous.
21. Dangerous curve to the right: Indicates an upcoming rightward curve that may be hazardous.
22. Double curve: Indicates consecutive curves ahead, which may be hazardous.
23. Bumpy road: Warns drivers of a road surface that is uneven or rough.
24. Slippery road: Alerts drivers of a road surface that is slippery or prone to reduced traction.
25. Road narrows on the right: Informs drivers that the road width decreases on the right side.
26. Road work: Notifies drivers of ongoing road construction or maintenance.
27. Traffic signals: Indicates the presence of traffic signals at the upcoming intersection.
28. Pedestrians: Warns drivers to watch out for pedestrians crossing the road.
29. Children crossing: Alerts drivers to the possibility of children crossing the road.
30. Bicycles crossing: Indicates the potential crossing of bicycles ahead.
31. Beware of ice/snow: Warns drivers of icy or snowy road conditions.
32. Wild animals crossing: Alerts drivers to the potential crossing of wild animals.
33. End of all speed and passing limits: Marks the end of previously indicated speed and passing restrictions.
34. Turn right ahead: Instructs drivers to prepare for a right turn ahead.
35. Turn left ahead: Instructs drivers to prepare for a left turn ahead.
36. Ahead only: Indicates that the driver must proceed straight ahead; no other turns are allowed.
37. Go straight or right: Allows the driver to either proceed straight or make a right turn.
38. Go straight or left: Allows the driver to either proceed straight or make a left turn.
39. Keep right: This sign instructs drivers to stay on the right side of the road or follow a specific lane on the right.
40. Keep left: This sign instructs drivers to stay on the left side of the road or follow a specific lane on the left.
41. Roundabout mandatory: This sign indicates that drivers are required to enter and navigate through a roundabout rather than proceeding straight or turning at an intersection.
42. End of no passing: This sign marks the end of a previously designated stretch of road where passing or overtaking other vehicles is prohibited.
43. End of no passing by vehicles over 3.5 metric tons: This sign indicates that the restriction on passing or overtaking specifically applies to vehicles weighing over 3.5 metric tons and that the restriction ends at this point.

We have diligently followed the annotation guidelines and methodology for German Traffic Sign Detection using the GTSDB dataset to ensure accuracy and consistency in labeling the dataset.

3.Related Works

A vast majority of existing approaches perform well on limited sets of traffic signs.As a traditional approach attempts have been made using image processing algorithms to detect handcrafted detectors and classifiers.
Miguel et.al [1] work proposes to establish a system in which traffic signs may be detected and classified at the same time under different lighting conditions. Circular prohibitions and obligations, together with triangular advertising signs, are intended to be detected by the system. The process of detection, classification and tracking is performed in three stages.Detection, classification, and tracking. In the detection stage, the Hough transform was used to identify potential traffic signs based on the edges in the image. Then, in the classification stage, a neural network was used to classify the detected signs based on their shape and color. Finally, in the tracking stage, a Kalman filter was used to track the detected signs over time, providing the system with memory to improve its accuracy.The orientation of the camera is decisive. Should a circular sign be captured non-orthogonally by the camera,it would be seen as an ellipse in the image and would not be so neatly detected.They have worked only on two classes of data speed limit and warning signs.

Traditional traffic sign recognition methods can be classified into three main categories:
A color-based method, a shape-based method, and a sliding window-based method. Color-based methods are based on the fact that road signs are often red, yellow, or blue. These methods use a normalized color space such as RGB or HSI to enhance the corresponding color blob. In [2], red,blue and yellow pixels are enhanced in the given color channels to extract the corresponding color blobs.
.While these methods can provide some performance gains, they are sensitive to lighting changes and require a set of designer-set thresholds. Additionally, traffic signs can fade over time, making these methods less reliable.

Shape-based methods exploit the specific shapes of traffic signs, such as circles, triangles, and squares, to detect them. The Hough Transform is commonly used to detect circular and triangular traffic signs[3]. However, these methods can be time-consuming. To address this issue, Graeth et al.[4] proposed a radial symmetry detector to reduce processing time,The method uses the symmetric nature of these shapes, together with the pattern of edge orientations exhibited by equiangular polygons with a known number of sides, to establish possible shape centroid locations in the image. This approach is invariant to in-plane rotation and returns the location and size of the shape detected. Results on still images
show a detection rate of over 95%.However, these methods can be sensitive to noise in cluttered environments and can struggle with occluded or partially visible signs.

Some have worked in combination of color and shape features as well. Abdulrahman [5] has presented an approach by using Fuzzy Neural Network (FNN) and it includes three stages. The first stage segments the images to extract ROIs. The segmentation is usually performed based on Adaptive thresholding to overcome the color segmentation problems. The second one detects traffic shapes. Given that the

geometric form of traffic signs is limited to triangular, circular, rectangular and octagonal forms, the geometric information is used to identify traffic shapes from ROIs provided by the first stage. The third stage recognizes the traffic signs based on the information included in their pictograms. Moreover, in this work, six types of features are extracted.

4.Methods

We have used three different approaches in the project :

- Building the faster RCNN network from scratch (ResNet50 backbone) ***(Appendix A)***.
- Using a pytorch faster RCNN library function (MobileNet v2 backbone)***(Appendix B)***.
- Using a pytorch faster RCNN library function (ResNet50 backbone).

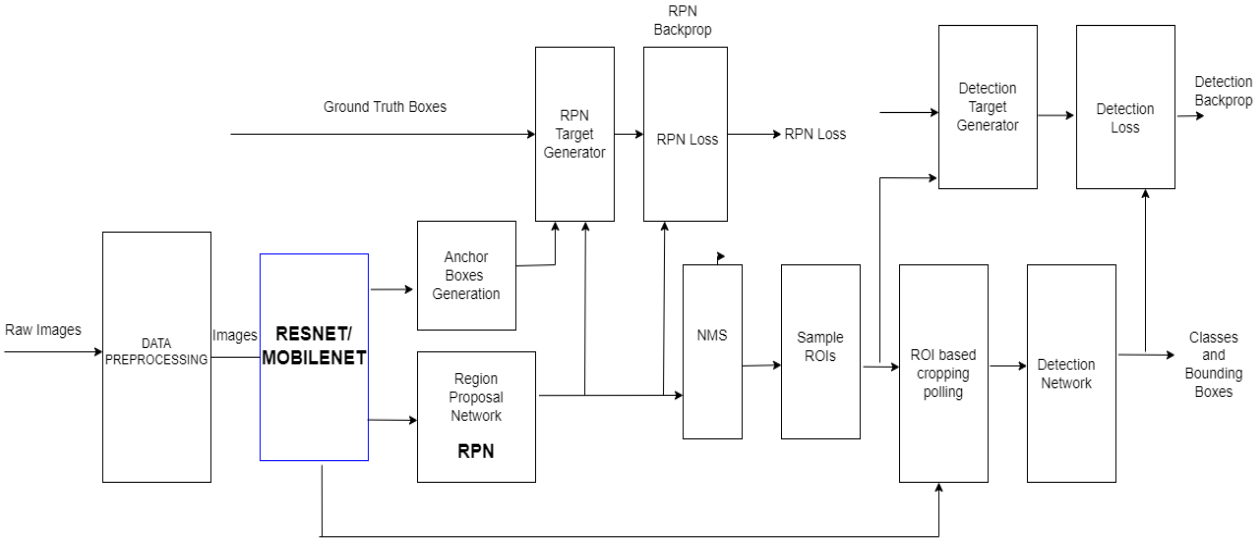

Fig. Architecture of Project workflow(RCNN +RESNET)

A.Data Preprocessing

- As some of the images were not having ground truth values and hence incomplete we couldn't simply work with those data so we performed data cleaning first by only retaining those image instances for which the class label and bounding box values are present in ground truth values.
- As there was imbalance among classes we performed data augmentation techniques with the motive of increasing the dataset and somehow nullifying the class imbalance problem.We have experimented with various data augmentation techniques in our projects.
- The techniques we have experimented with are **Image Flipping:** Flip the images horizontally or vertically to create new samples. This is particularly useful for traffic signs that are symmetrical.

- **Translation**: Shift the position of the traffic sign within the image randomly. This simulates different spatial placements and improves the model's ability to locate signs in various positions.
- **Image Normalization**: Normalizing the pixel values of the images can help improve the performance of the model by ensuring that the inputs have a similar scale.

B. Data Loading

- We have also experimented with load data in batches of different sizes as batch size is an important hyperparameter in object detection models that determines the number of training samples processed in each iteration of the training process.A **larger batch size** allows for more samples to be processed simultaneously, leading to faster training times. This is because parallelization can be effectively utilized on modern hardware, such as GPUs, to process multiple samples in parallel.However their is memory requirement when we work with batch of larger size.Larger batch sizes require more memory to store the activations, gradients, and intermediate results during the forward and backward passes. In our case considering both the tradeoff between training efficiency and memory requirement , we have worked with a batch of size 2.

The process or method in the detection of the traffic signal in the image given is done in steps that are mentioned below. Step-1 : Preparation of the data as per the requirements Step-2 : Developing and training the model.

In the step-1, we prepare the data as per our requirement, the following sub steps have been followed. Initially, we initialize a python dictionary with empty value and then store the image name as key along with the coordinates of the traffic sign and class value of the sign as values of that particular key. And in the next process, when checking the gt.txt file, we found out that not all images contain the sign coordinates and hence we only use the images with the sign coordinates and class value in the model training and so we separate them into another folder. Once the dataset is ready, we define a custom pytorch dataset class to load the data and then a class for data augmentation using pytorch transforms. Once that everything is coded and ready, the dataset is loaded by calling the corresponding classes.

In step-2, we develop the model that will use the data we input for the training dataset. For this, the following sub steps. First we split the dataset into training and validation datasets. And then we define the main model that is required to train the data using the Faster R-CNN technique with ResNet50 as backbone for feature extraction. Later in the model development, we define a few additional parameters that are mentioned below.

Parameter	Value
Optimiser	Stochastic Gradient Descent
Learning Rate Scheduler	CosineAnnealingWarmRestarts
Number of epochs	720
Learning Rate	0.0005

Table-1 : Parameters and values defined in the model

We trained the model for 720 epochs using the P100 GPU available on kaggle. Errors during each epoch were recorded for model analysis.

5.Experiments and results

We have used COCO detection evaluation metrics to evaluate the developed model. The metrics that were used include Average Precision (AP), Average Recall (AR) which are defined as follows.
True Positive (TP): When the Intersection over union (IoU) over the predicted bounding box and ground truth is greater than or equal to the threshold.
False Positive (FP): When the IoU over predicted bounding box and ground truth is less than threshold. Average Precision (AP) is the number of true positives in the resulting bounding boxes. Average Recall (AR) is the proportion of true positives out of possible positives.
The below image represents evaluation of the model on test data.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.819
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.936
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.930
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.846
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.859
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.825
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.777
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.844
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.844
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.851
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.866
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.825
```

Fig-1: Sample of the evaluation output (ResNet 50 backbone)

The AP @ IoU=0.5:0.95 for area = large is 0.800 which means that when the model detects an object with a large area, 80% of the time it matches the ground truth objects. The AR @IoU=0.5:0.95 for area = large is 0.800 which means that the model detects 80% of objects with large area, correctly.

```
Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.146
Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.338
Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.111
Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.080
Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.236
Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.433
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.191
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.276
Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.276
Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.188
Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.373
Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.433
```

Fig-2: Sample of the evaluation output (MobileNet V2 backbone)

Loss Box Reg is the measure of how tightly the model predicted the bounding box around the true object. Loss RPN Box Reg measures the performance of the network for retrieving the region proposals. Loss Classifier measures the performance of the object classification for detected bounding boxes. Loss Objectness measures the performance of a network for retrieving bounding boxes which contain an object. The respective plots are shown below after the data is trained on the model for 720 epochs.

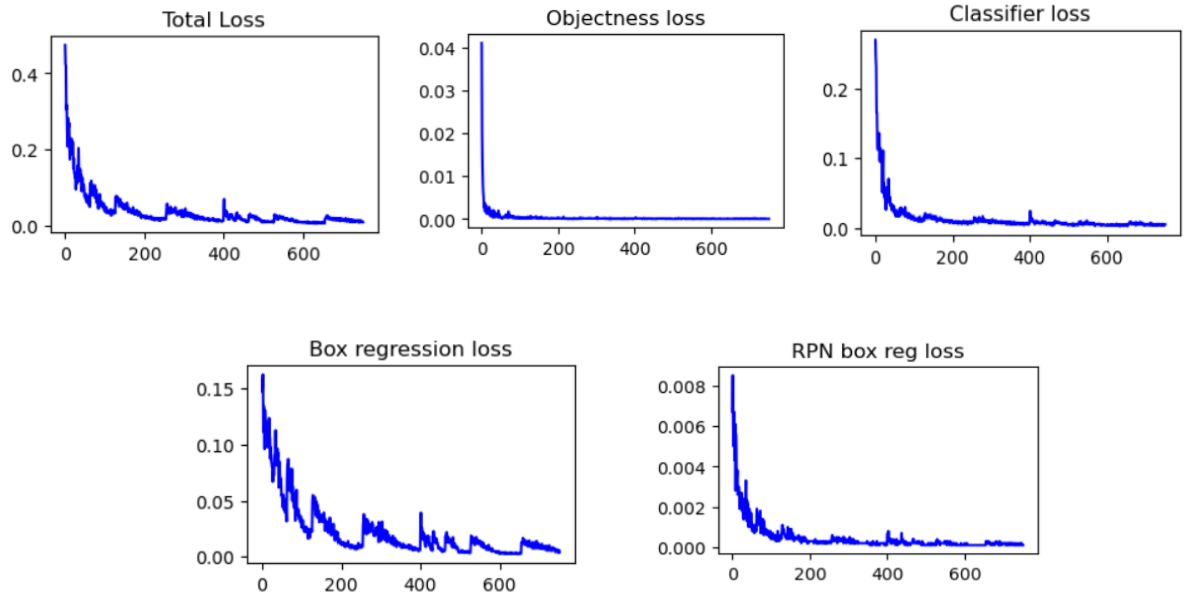

Fig-2: Various Loss Curves used in evaluation after training for 720 epochs (ResNet50 backbone)

(a) (b)

Fig-4: (a) Prediction using model with MobileNet V2 backbone (fails to detect multiple objects).
(b) Prediction using model with ResNet50 backbone

Result with ResNet50 backbone

Fig-5: Model (ResNet 50 backbone) predicting correctly multiple signs in single image with probabilities

6.Conclusions

In conclusion, our traffic sign detection and prediction project utilizing RCNN (Region-based Convolutional Neural Networks) combined with ResNet on the German Traffic Sign Database has yielded several noteworthy outcomes.

1. Accurate Detection: The RCNN model with ResNet has proven to be highly effective in detecting traffic signs within images. The combination of region proposals and convolutional neural networks allows for precise localization and classification of traffic signs, leading to reliable detection results.

2. Robust Performance: Through extensive testing and evaluation, we have observed consistent and robust performance of our model across various scenarios. It demonstrates a high level of accuracy in identifying different types of traffic signs, even under challenging conditions such as low lighting, occlusions, and variations in scale.

3. Efficient Processing: Our chosen architecture, utilizing ResNet, has demonstrated efficient processing capabilities. By leveraging the residual learning approach, ResNet enables deeper networks while avoiding the degradation of performance typically associated with deeper architectures. This allows for faster inference times and real-time or near-real-time performance in traffic sign detection.

4. Generalizability: The model trained on the German Traffic Sign Database has showcased the potential for generalization to other datasets and real-world scenarios. Although our focus was on German traffic signs, the underlying principles and techniques can be adapted to other traffic sign datasets from different countries, making it a versatile solution.

5. Potential for Real-World Applications: The successful implementation of our traffic sign detection and prediction project holds great potential for real-world applications. It can be utilized in advanced driver assistance systems (ADAS) and autonomous vehicles to enhance road safety by providing timely and accurate information about traffic signs, enabling intelligent decision-making by the vehicle.

6. Room for Improvement: While our project has achieved impressive results, there is always room for improvement. Future work could involve exploring alternative architectures or training techniques to further enhance detection accuracy, investigating ways to handle complex scenarios with multiple signs, and expanding the dataset to encompass a wider variety of traffic sign types and conditions.

Overall, our project demonstrates the effectiveness of RCNN combined with ResNet in traffic sign detection and prediction, showcasing its potential impact on road safety and its applicability in real-world scenarios.

7.Appendix A

Using **ResNet50** as a backbone of our network, we were trying to implement **Faster RCNN** from scratch. We have shown the steps in the above Method module. Steps to do so are described as follows:

1. Region Proposal Network(RPN): The RPN generates a set of candidate object bounding box proposals by sliding a small network (typically a CNN) over the convolutional feature map. For each anchor box at different spatial positions, the RPN predicts two values for each anchor: objectiveness score (foreground or background) and bounding box regression offsets (to refine the anchor box).

2. Region Of Interest(RoI) Pooling: The RoI pooling layer takes the candidate object bounding box proposals generated by the RPN and extracts fixed-size feature maps from the feature map produced by the CNN backbone network. These features are then fed into separate branches for classification and bounding box regression.

3. Classification: The classification branch takes the RoI features and applies fully connected layers and softmax activation to classify each RoI into different object categories. The number of classes corresponds to the total number of object categories that the model has been trained to detect.

4. Regression: The regression branch takes the RoI features and predicts refined bounding box coordinates (offsets) for each object proposal. These offsets are applied to the initial bounding box coordinates to obtain more accurate localization of the objects.

Result: After training for approx 7 hours, this model gives incorrect output.

8.Appendix B

In the project we also try to focus on reducing the computational cost as this project involves training the parameters of a neural network such that the algorithm learns to discern between object classes , by feeding many images of labeled data to the neural network, while updating the parameters to increase performance on a smooth objective function , but the drawback is that a large number of parameters are used, compared to more traditional algorithms.

MOBILENET

MobileNet is a type of neural network architecture that was designed to be lightweight and efficient, making it well-suited for deployment on devices with limited computational resources.

It uses depthwise separable convolutional layers to reduce the number of parameters and computations required for the network. This type of layer applies a separate convolutional filter to each channel of the input, followed by a pointwise convolution that combines the outputs from the depthwise convolution. It also includes a technique called "bottlenecking" that reduces the dimensionality of the feature maps before the depthwise convolutional layers. This helps to further reduce the number of parameters and computations required. Despite its lightweight design, MobileNet can achieve high accuracy on image classification tasks. It has been shown to outperform other neural network architectures.

But it may be more prone to overfitting than other neural network architectures, especially when trained on smaller datasets and it may be difficult to incorporate additional layers or modify the architecture without sacrificing its efficiency. This can result in decreased accuracy and poor performance on unseen data.

QUANTIZATION

This is a method to bring the neural network to a reasonable size, while also achieving high performance accuracy. It tries to reduce the number of bits used to represent activations and gradients within the network throughout training. It is the process of approximating a neural network that uses floating-point numbers by a neural network of low bit width numbers. The forward pass of the neural network uses a scheme for rounding float-precision parameters to discrete levels, and in the backwards pass, the float-precision parameters are updated using gradients calculated during the forward pass. This helps to reduce both the memory requirement and computational cost of using neural networks.

9.Reference

1. Garcia-Garrido, M. A., Sotelo, M. A., & Martin-Gorostiza, E. (2006, September). Fast traffic sign detection and recognition under changing lighting conditions. In *2006 IEEE Intelligent Transportation Systems Conference* (pp. 811-816). IEEE.

2. Ruta, A., Li, Y., & Liu, X. (2010). Real-time traffic sign recognition from video by class-specific discriminative features. *Pattern Recognition*, *43*(1), 416-430.

3. García-Garrido, M. Á., Sotelo, M. Á., & Martín-Gorostiza, E. (2005). Fast road sign detection using hough transform for assisted driving of road vehicles. In *Computer Aided Systems Theory–EUROCAST 2005: 10th International Conference on Computer Aided Systems Theory, Las Palmas de Gran Canaria, Spain, February 7–11, 2005, Revised Selected Papers 10* (pp. 543-548). Springer Berlin Heidelberg.

4. Loy, G., & Barnes, N. (2004, September). Fast shape-based road sign detection for a driver assistance system. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)* (Vol. 1, pp. 70-75). IEEE.