B. TECH. PROJECT REPORT

On

TRAFFIC SIGN RECOGNITION SYSTEM FOR INDIAN ROAD CONDITIONS USING YOLOv4

By

ASHUTOSH PATEL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2022

Traffic sign recognition System for Indian road conditions using YOLOv4

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degrees

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by:

Ashutosh Patel (180001010)

Guided by:

Dr. Somnath Dey

(Associate Professor, Computer Science and Engineering)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

May 2022

CANDIDATE'S DECLARATION

We hereby declare that the project entitled **"TRAFFIC SIGN RECOGNITION SYSTEM FOR INDIAN ROAD CONDITIONS USING YOLOv4"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Computer Science and Engineering' completed under the supervision of **Dr. Somnath Dey, Associate Professor, Computer Science and Engineering, IIT Indore** is an authentic work.

Further, I declare that I have not submitted this work for the award of any other degree elsewhere.

Ashutosh Patel

CERTIFICATE by BTP Guide(s)

It is certified that the above statement made by the students is correct to the best of my knowledge.

Dr. Somnath Dey
Associate Professor
Department of Computer Science and Engineering
IIT Indore

Preface

This report on "**TRAFFIC SIGN RECOGNITION SYSTEM FOR INDIAN ROAD CONDITIONS USING YOLO**" is prepared under the guidance of Dr. Somnath Dey.

Through this report, we have tried to give a detailed design of our proposed method for recognizing traffic signs on Indian roads which we have developed over the course of 6 months. We have tried to the best of our abilities and knowledge to explain the content in a lucid manner with examples as suggested by our guide. We have improved the widely used YOLOv4 model to improve the overall performance. We have also included the results of our implemented method in tabular and graphical form for better understanding and visualization.

Ashutosh Patel
B.Tech. IV Year
Department of Computer Science and Engineering
IIT Indore

Contributions

The major contributions made by me are as follows:

1. Images collected for Indian Traffic Sign Dataset (ITSD) across Indore city.
2. Annotated images captured by me.
3. Used Data Augmentation technique on ITSD dataset to increase the size of the dataset.
4. Applied Generative Adversarial Network (GAN) on German Traffic Sign Recognition Benchmark (GTSRB) to make the number of instances of each class comparable.

I would also like to appreciate the contributions made by my project partners. The major contributions made by them include:

1. Proposal and Implementation of Section 5.1, Section 5.2, Section 5.3 and Section 5.4.
2. Prepocessing of MTSD, ITSD, TT100k and GTSDB datasets.
3. Conducting the Ablation Study on Individual components and Detection Block.
4. Performance evaluation and comparative analysis of the proposed models by calculating performance metrics and frames per second on MTSD and TT100k dataset.
5. Suggestion, Implementation and Optimization of image enhancement model for night-time images.
6. Implementation and conducting cross dataset validation on ITSD and GTSDB dataset.

Acknowledgements

I wish to thank our B.Tech Project supervisor **Dr. Somnath Dey** for his kind support, expertise and valuable guidance. He provided a perfect environment for critical thinking and research acumen and was always available for discussions, doubt clearance and guidance at every part of the project. He has constantly motivated us to take the project to its very culmination.

I would also like to acknowledge **Mr. Swastik Saxena** for his support and sincere cooperation. It is his guidance and help, due to which we were able to handle the delicacies involving deep learning and experimentation. He was always available for discussion and doubt clearance.

I also thank my project partners **Sundesh Gupta**, **Miten Shah** and **Jeevan Reddy Puchakayala** who contributed to the making of this project.

I am grateful to the Institute for the opportunity to be exposed to systemic research, especially Dr. Somnath Dey's Lab, for providing the necessary hardware utilities to complete the project.

Without their support, this report would not have been possible.

Ashutosh Patel
B.Tech. IV Year
Department of Computer Science and Engineering
IIT Indore

Abstract

Traffic Sign Detection and Recognition is a vital topic for autonomous driving vehicles, cruise control and adaptive driver assistance systems. Real-time prediction of the traffic signs captured from car-mounted cameras and dashcams is a challenging task. Because the images captured may contain scenarios like low illuminated images, the small size of signs relative to the image, motion blur, object occlusion, and degraded/discolored traffic signs. Also, the perspective of traffic signs can be different. To address the issues, we introduce India Traffic Sign Dataset (ITSD) to improve the performance of models on Indian Roads. We also propose improvements in the existing YOLOv4 model. First, we use the Generalised Intersection over Union (GIOU) distance metric for Anchor box calculation. Second, we introduce an extra feature scale layer for detection. Third, we change connections from the CSPDarknet-53 backbone to the PANet neck to improve the utilization of shallow features. Finally, we modify the final feature extraction step (Detection Block) and use Grouped Convolutions to represent features better. We also employ an image illumination model to tackle the problem of low-light images captured at night. The proposed model is tested on three existing datasets, namely Mapillary Traffic Sign Dataset (MTSD), Tsinghua-Tencent 100k (TT100K), and German traffic sign detection benchmark (GTSDB). We achieve a 94.73% mAP on the TT100K dataset, which outperforms the existing state-of-the-art models. We conduct ablation on each of our proposed modifications, and finally, we perform cross-dataset validation to test the robustness of our system.

Keywords: Traffic Sign Detection and Recognition; YOLOv4; Object Detection; Computer Vision; Dataset

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Traffic signs are signs erected at the side of or above roads to give instructions or provide essential information for vehicles. These traffic signs are vital as they help avoid accidents and maintain norms and regulations. A basic understanding of the traffic signs is a must while driving vehicles. Though most of the signs are bright and distinctive, the driver can miss out on the signs due to several reasons. Recent advances in technology have come up with tools that can be useful to assist the driver or advanced driver-assistance system (ADAS). The traffic sign detection and recognition (TSDR) system helps us detect the traffic signs and classify them. With the increasing popularity of self-driving cars and the importance of cruise control features increasing, the role of TSDR systems will be pivotal. Figure 1.1a shows the example of the detection of the bounding boxes around traffic telling the system that traffic exists there, while 1.1b also classifies the traffic signs along with detecting them.

One of the most critical common challenges faced in the real-life implementation of the TSDR task, as mentioned in the survey paper [30] is illumination perspectives. The illumination changes affect the color and contrast while making it almost invisible for the low-light images. The other issues mentioned were the visual perspective, motion blur, partial occlusion, etc. We proposed changes in the standard YOLOv4 [3] model to improve the performance. The dataset used for the experiments are Mapillary Traffic Sign Dataset (MTSD) [4], Tsinghua-Tencent 100K (TT100K) dataset [39], and The German Traffic Sign Detection Benchmark (GTSDB) [9]. The key contributions of our work can be summarized as follows:

- Proposed a solution based on YOLOv4 that outperforms several previous works on this task.

(a) (b)

Figure 1.1: Traffic Sign Detection and Classification [4] [39]

- Enhanced k-means clustering algorithm by selecting GIoU as a distance metric over IoU for calculating anchor boxes.
- Improved Path Aggregation Network (PANet) of the model by routing the connections to lower hierarchical layers of the backbone to improve performance for small objects. Incorporated an extra feature layer to improve detection.
- Improved the layers before detection head, i.e., detection block, for better representation using grouped convolution.
- Collected and annotated images with different illumination for the Indian Roads.
- Performed cross-dataset validation using our model to check the robustness of the same. Used a novel night-time image enhancement illumination model.

The rest of the report is organized as follows. Chapter 2 discusses some of the existing related works done on the TSDR task. Chapter 3 proposes new dataset based on Indian roads. Chapter 4 gives an overview of the literature required to understand the proposed solutions. Chapter 5 discuss the proposed model based on YOLOv4 model followed by experimental analysis in Chapter 6. We also conduct ablation on different components of our proposed model in Chapter 6. We conduct cross dataset experiments in Chapter 7 to validate the performance of our model. Finally, Chapter 8 summarizes the conclusion and future works regarding the TSDR task.

Chapter 2

Related Works

This chapter reviews the existing works done on TSDR problem. Before the adoption of deep learning and convolutional neural networks (CNNs), diverse object detection approaches were adapted for traffic-sign classification, e.g., based on SVMs [17] and sparse representations [18]. Traditional approaches for the TSDR task were generally a two-step process. First, the extraction of regions of interest (ROI) using color and shape based techniques [34], [20]. Most traffic signs have a similar pattern, such as a red border with black text having a white background and standard shapes like circles or triangular. These patterns were exploited in color segmentation techniques [34] using thresholding in RGB color space. Others employed Histogram-of-Oriented-Gradient (HOG) features [37]. Secondly, after extraction of ROI, different machine learning models like AdaBoost [34] and fuzzy regression tree framework [29] are applied for recognition of traffic signs. Later the researchers applied CNNs for both detection and recognition tasks. In [15], authors use a series of convolution layers to extract features of the image and later use SVM to classify the traffic signs. Authors of [15] use the Deep Perceptual Feature model for the TSDR task and argue that this network is highly efficient for color-based object detection tasks. Researchers in [23] do detection using a color-based region proposal with a multi-tasking CNN, and classification is done simultaneously using fully connected layers. More recent techniques are based on Region-based CNN (R-CNN) models, which is a region-based detector, and single-shot detectors (SSD) like YOLO [22, 25, 26, 3] models, which are a region free detector. [38] use a cascaded RCNN wtih a multiscale attention mechanism to improve the performance of traffic sign systems. In [32], researchers use mask R-CNN to address the full pipeline of detection and recognition with automatic end-to-end learning. Authors in [2] used YOLOv3, YOLOv4, and TinyYOLO to achieve real-time detection. In [19], authors propose Contextual YOLOv3 to utilize contextual information for the TSDR task.

Chapter 3

Dataset Collection

Without proper traffic signs, our road infrastructure would be severely lacking. If we did not have such helpful signs, there would most likely be an increase in the number of accidents that occurred. This is because drivers would not be provided with critical feedback on how fast they could go safely, nor would they be informed about roadworks, sharp turns, or school crossings that lay ahead. Traffic signs can be divided into different categories based on their function, and within each category, sub-classes with similar generic shapes and appearance but different details can be found. This suggests that the recognition of traffic signs should be done in two stages: the first stage is detection, and the second stage is classification. During the detection phase, shared data are utilised to generate bounding boxes that have the potential to contain traffic signs belonging to a particular category. On the other hand, the classification stage analyses the differences between the signs in order to establish which category each one belongs to (if any).

3.1 Characteristics of Traffic Signs

- Traffic signs have been designed so that they are easily recognizable from the natural and driving environment.
- The color for traffic signs is chosen such that, it serves different purposes and is also distinguishable for the driver while driving. The signs are represented by fixed shapes like triangles, circles, octagons, and rectangles.
- Traffic signs in India are categorized as: *WARNING (40), COMPULSORY (27), REGULATORY (10), and INFORMATORY (15).* This makes a total of 92 traffic signs altogether.

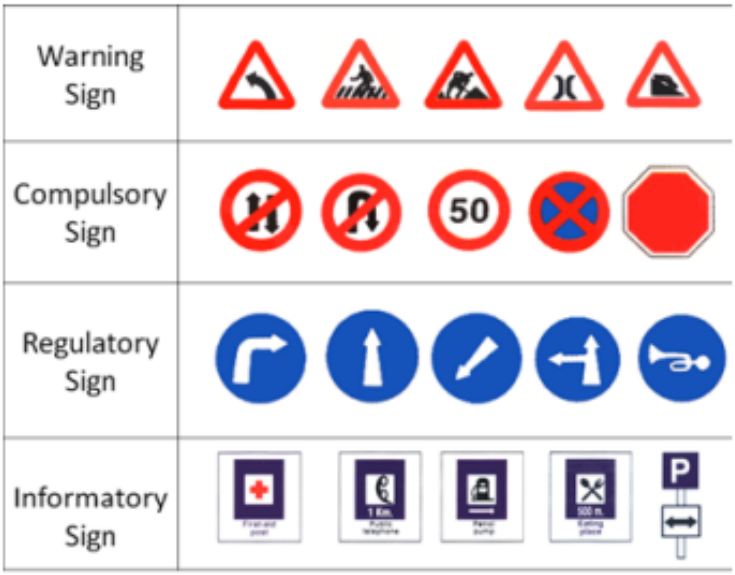


Figure 3.1: Different Categories of Traffic Signs

3.2 Need for Data Collection

Data is crucial in machine learning. It is the most important aspect that enables algorithm training. The dataset that is currently available on the internet does not contain enough information to obtain a higher level of accuracy. In addition to this, we need to ensure that we are making use of wide variety of data such as pictures taken with/without motion blur, in different lighting and weather conditions in order to achieve better results. There are only 900 images contained within the German Traffic Sign Dataset that is the most widely used, and there is not a suitable Indian Traffic Sign Benchmark Dataset. This was the motivation behind the collection of traffic signs found on Indian roads.

Figure 3.2: GTSDB (German Traffic Sign Detection Dataset Benchmark)

3.3 Dataset Collection

3.3.1 Indian Traffic Sign Dataset (ITSD)

Class	Number of Instances
Mandatory	765
Cautionary	649
Informatory	137
Other	142

Table 3.1: Number of Instances of different classes in ITSD dataset

- The dataset includes 1000+ traffic scene images containing different kinds of signs. The images are collected at different times, under different lighting conditions, as well as with moving blur. The images were collected using the OnePlus 6 smartphone device. The images were captured during daytime/night while the vehicle was moving/stopped at different angles. A total of 500 images have been captured, excluding some common signs.

- Collected 800+ pictures that have at least one traffic signal in them from 50 hours of videos recorded using a dashcam on a car while driving inside Indore city and on the way to IITI from Indore city during different light conditions. The pictures were collected using the VIFO A129 4K Ultra Dash Cam.

Figure 3.3: Sample from ITSD dataset

3.3.2 Recognition Dataset

German Traffic Sign Recognition Benchmark (GTSRB):

More than 50,000 images of various traffic signs are included in the GTSRB. It's further divided into 43 different classes. The dataset is quite diverse, with some classes having many images and others having few. The dataset is approximately 300 MB in size. The dataset includes a train folder that contains images for each class, as well as a test folder for testing the model.

Figure 3.4: Sample from GTSRB (German Traffic Sign Recognition Dataset Benchmark)

Indian Traffic Sign Dataset (ITSD):

Same images from detection dataset are taken and annotated them with recognition labels. Signs are classified into 43 types. These pictures are annotated using makesense.ai tool. Below in the table 3.2 is the analysis of dataset.

Figure 3.5: Sample from ITSD dataset after annotations

Class	No of Instances	Class	No of Instances
cautionary-left-hand-curve	26	cautionary-right-hand-curve	63
cautionary-hump	214	cautionary-gap-in-median	11
cautionary-cross-road	14	cautionary-t-intersection	32
cautionary-men-at-work	21	cautionary-gaurded-level-crossing	0
cautionary-school-ahead	1	cautionary-pedestrian-crossing	67
cautionary-narrow-bridge	0	cautionary-give-way	15
cautionary-go-slow	24	cautionary-barrier-ahead	1
cautionary-ungaurded-level-crossing	4	cautionary-round-about	12
mandatory-stop	20	mandatory-no-entry	0
mandatory-right-turn-prohibited	0	mandatory-left-turn-prohibited	0
mandatory-u-turn-prohibited	25	mandatory-overtaking-prohibited	23
mandatory-horn-prohibited	1	mandatory-width-limit	0
mandatory-height-limit	0	mandatory-length-limit	0
mandatory-speed-limit	168	mandatory-no-parking	3
mandatory-no-stopping-or-standing	1	mandatory-compulsory-ahead	11
mandatory-compulsory-turn-left	0	mandatory-compulsory-turn-right	0
mandatory-restriction-ends	19	informatory-hospital	2
informatory-petrol-pump	7	informatory-first-aid-post	0
informatory-eating-place	0	informatory-light-refreshment	0
informatory-resting-place	0	informatory-bus-stop	14
informatory-bus-stop	14	informatory-direction-signs	0
informatory-advance-direction-sign	0	informatory-confirmatory-sign	0

Table 3.2: Number of Instances of different categories

3.4 Data Augmentation

Data Augmentation is a technique that uses images transformations(but realistic) like image rotation to increase the diversity of your training set. Offline augmentation was used on ITSD because it is appropriate for smaller datasets because it increases the size of the dataset by a factor equal to the number of transformations performed (For example, by flipping all my images, I would increase the size of my dataset by a factor of 2).

3.4.1 Generative Adversarial Network (GAN)

We noticed a significant disparity in the data across the 43 classes. There is a significant disparity between the number of instances of each class, which can lead to decreased accuracy and unsatisfactory outcomes. Some of the images in the test set are also distorted. As a result, we'll use data augmentation techniques to:

- Extend the dataset and provide additional images in various lighting settings and orientations.

Figure 3.6: Example of Data Augmentation

- Improve the model's generic ability.
- Improve the accuracy of test and validation, especially on distorted images.

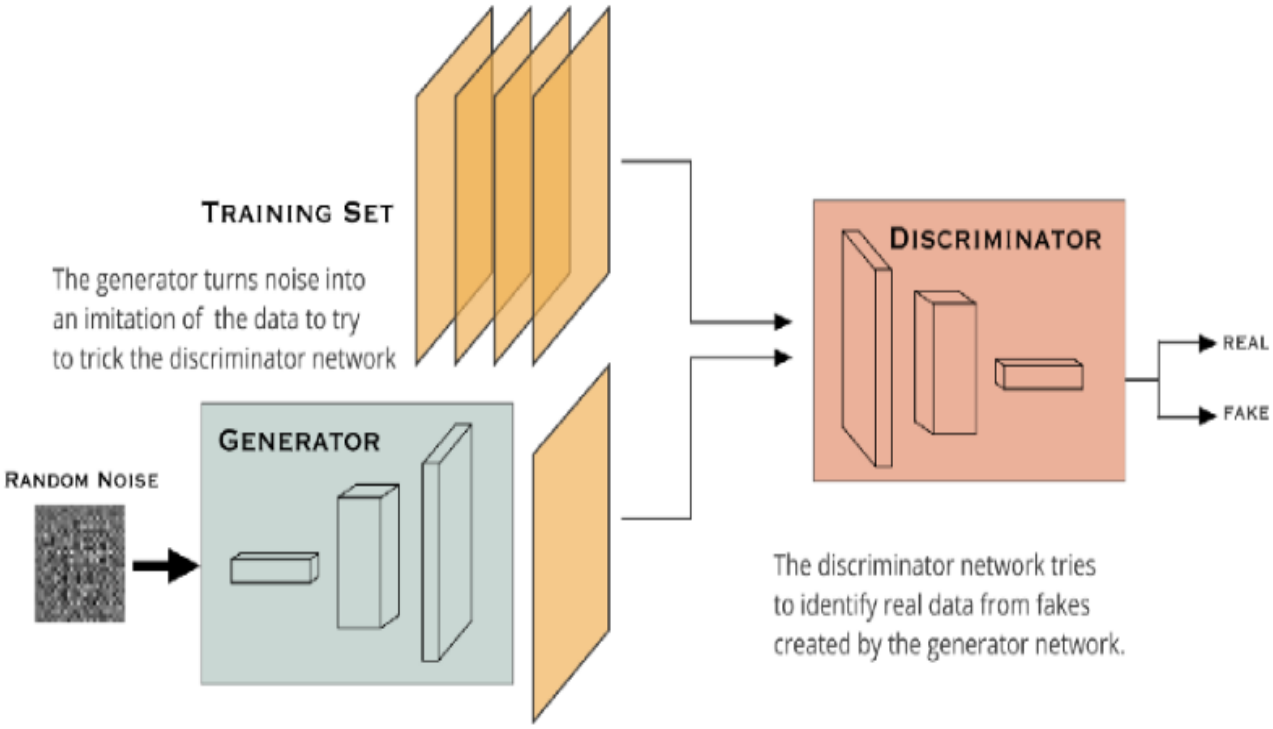


Figure 3.7: Generator and Discriminator Relationship in a GAN Network

Generative Adversarial Network (GAN): A generative adversarial network (GAN) is a class of machine learning frameworks conceived in 2014 by Ian Goodfellow and his colleagues. Two neural networks: a generator that generates an image

based on a given dataset, and a discriminator (classifier) to distinguish whether an image is real or generated, compete with each other. GANs, for example, can create images that resemble photographs of human faces, despite the fact that the faces do not belong to anyone. The discriminator and the generator are both neural networks. The discriminator input is directly connected to the generator output. The discriminator's classification provides a signal that the generator uses to update its weights via backpropagation.

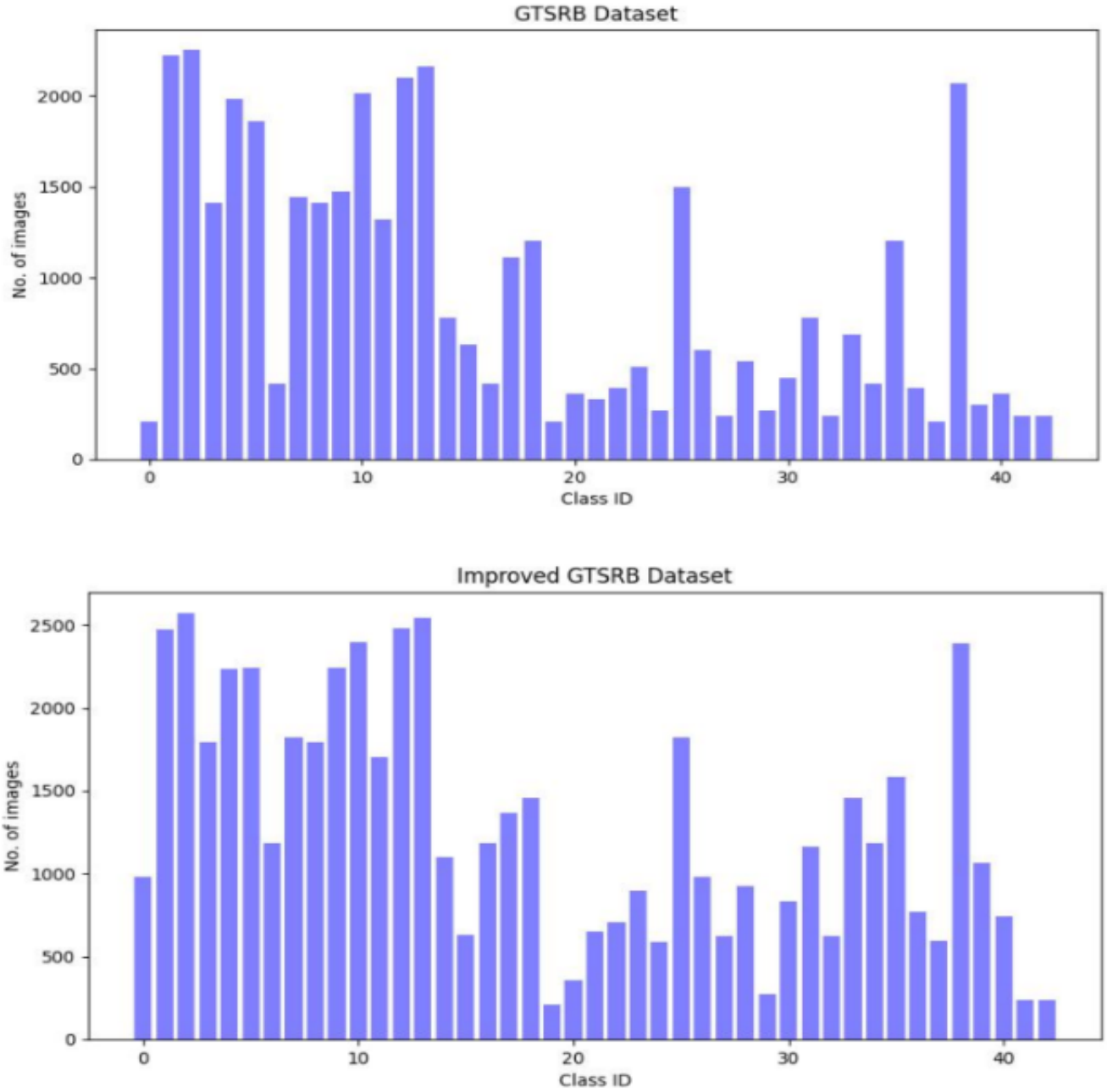


Figure 3.8: Graphical representation of GTSRB dataset (up) and Improved GTSRB dataset (down)

3.5 Annotations

Image annotation is the process of labeling features in images of a dataset to train a machine learning model which will be able to recognise them given an input. An annotation contains an array of features, each feature is described by feature class name, center coordinates, height and width of the feature. These annotations are made using a tool called makesense.ai.

Figure 3.9: Example of an Annotation

Chapter 4

Background

In this chapter, we will describe the YOLOv4 model, which is the base of our proposed model. Afterward, we explain the MobileNetV2 architecture that forms the basis of our improved detection head.

4.1 YOLOv4

You Only Look Once (YOLO) [24] is an object detection model that predicts bounding boxes and associated class images from full images in one evaluation. YOLOv4 [3] is an improvement over YOLO and its off-springs YOLO9000 [25] and YOLOv3 [26] in terms of speed and accuracy. YOLOv4 consists of 4 components: input, backbone, neck, and dense prediction or head. Input consists of the image containing the objects we want to detect. Backbone consists of deep convolution neural network which extracts features from the image. Semantic elements are further extracted in the neck for precise predictions. The head is responsible for predicting the objectness score for each bounding box using logistic regression.

The figure 4.2 shows the detailed architecture of YOLOv4. Starting with the backbone, YOLO9000 proposed a Darknet-19 network for feature extraction. It uses 3×3 filters and doubles the number of channels after every pooling step and 1×1 filters to squeeze the feature representations between 3×3 filters. In Darknet-19, a residual module similar to the ResNet [8] network structure is presented to obtain a more powerful Darknet-53, [26] used in YOLOv3. Cross-stage partial network (CSPNet) [35] is introduced in Darknet-53 for further improvement of the network performance to form CSPDarknet-53. CSPDarknet-53 adds a cross-stage

feature fusion strategy to the Darknet-53 local network in order to improve the network's reasoning. CSPDarknet-53 also reduces the computational complexity of the backbone by preventing the repeated gradient information in different layers. This is achieved by splitting the feature map of the base layer and later merging them using the cross-stage fusion strategy. The authors of YOLOv4 thus suggest using CSPDarknet-53 due to its superior learning ability, reduced computation cost, and memory footprint, as well as improved inference speed.

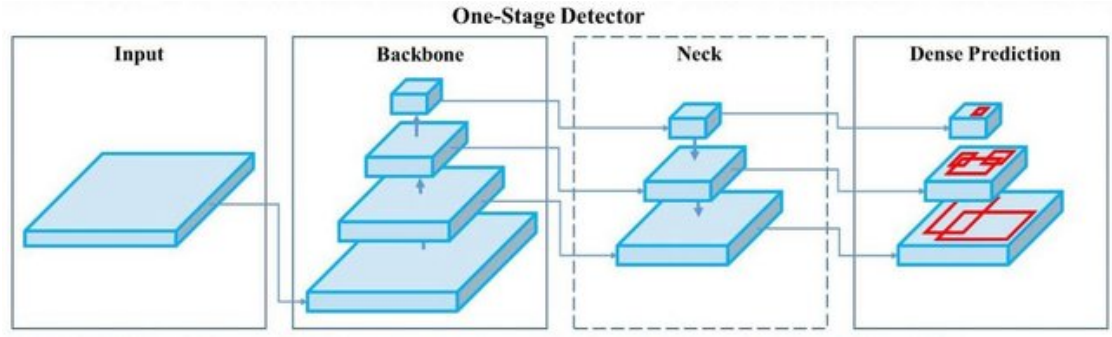


Figure 4.1: YOLOv4 object detector architecture [21]

To further improve the performance of YOLOv3, researchers added a few layers between the backbone and detection head to accumulate various features from different phases. In YOLOv4, the authors decided to call it the neck of the architecture and replaced the Feature pyramid networks (FPN) used in the previous iterations of YOLO with the Spatial Pyramid Pooling (SPP) block and Path Aggregation Network (PANet). SPP-net [7] was introduced to eliminate the requirement of the fixed-sized input image by CNNs to generate fixed-length image feature representation. It uses multi-level pooling layers to make it robust against object deformations as it helps in localizing the most significant contextual features. In YOLOv4, to further improve the receptive field, the SPP module was added on top of the CSPDarknet-53 backbone in which the outputs of max-pooling layers with kernel size k x k, here k = {1, 5, 9, 13} are concatenated. To further improve the recognition of pixel-wise masks, PANet [16] is used to integrate the low-level features from the network hierarchy. The output of the SPP block is convolved and then upsampled and concatenated with the features from the backbone. This shortens the information path between shallow and high-level features, which results in precise localization.

Finally, at the detection head, a one-stage dense predictor (e.g., YOLO) is applied to predict the bounding boxes. This is achieved by first dividing the image into equally spaced grids. The model generates predictive bounding boxes (bound-

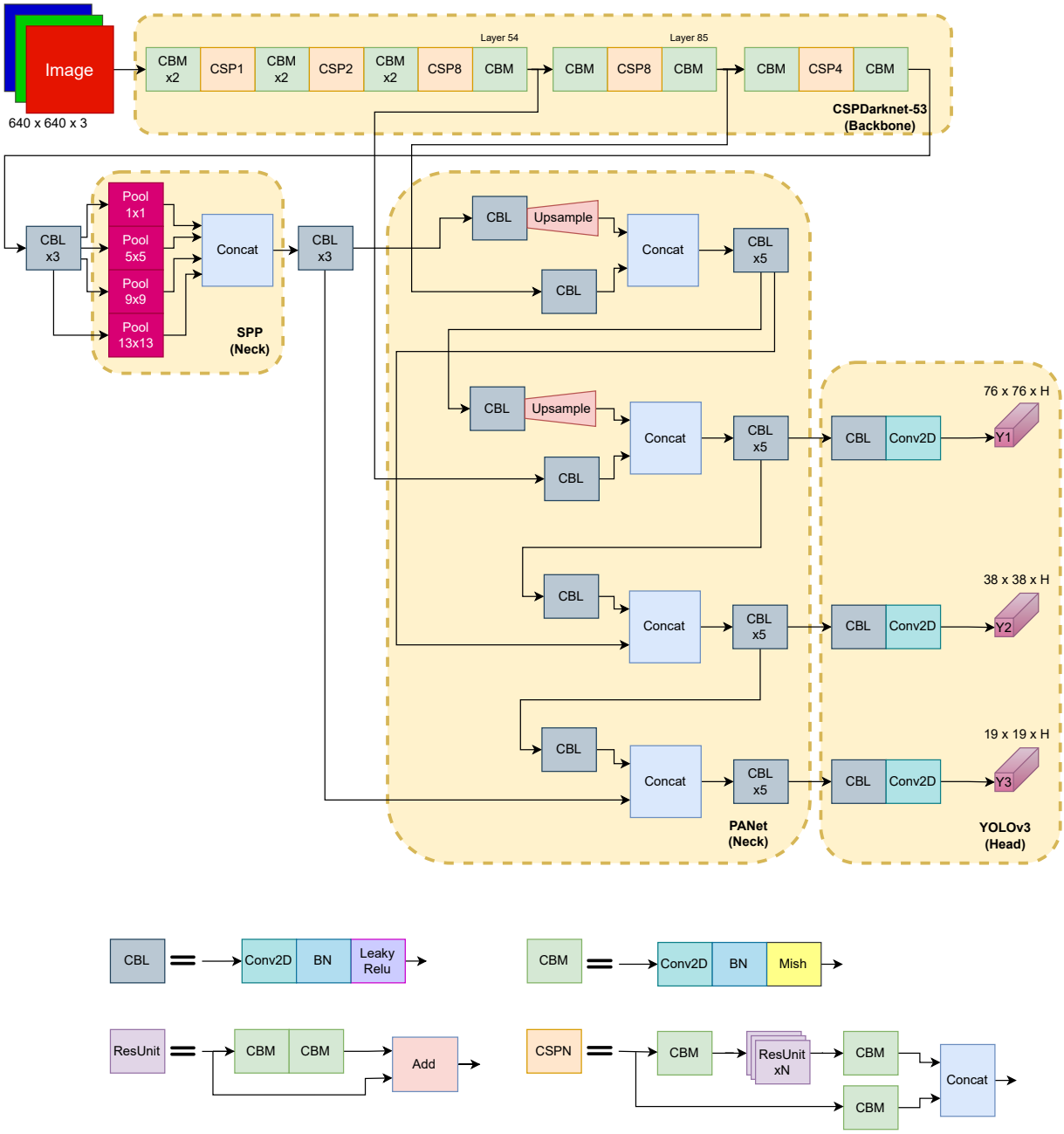


Figure 4.2: YOLOv4 network architecture

ing box coordinates along with the confidence of the bounding box) for each of the grid cells and class probabilities representing whether the center of the target object belongs within the grid cell. The bounding box confidence score tells whether the predicted bounding box contains objects and the accuracy of the position when the objects are included. The formula for confidence is as follows:

$$confidence = Pr(Object) * IoU_{pred}^{ground} \tag{4.1}$$

where, IoU_{pred}^{ground} is the Intersection over Union between the predicted box and

the ground truth, and $Pr(object)$ is the probability of the object being detected in the grid. If no object exists in a cell, its confidence score should be zero.

These bounding boxes are further filtered by non-maximum suppression (NMS) [27]. In order to further improve the generalization performance of YOLOv4, the authors have suggested techniques like focal loss to deal with the problem of class imbalance, data augmentation to deal with noisy, complex backgrounds, and different choices of loss functions for bounding box regression.

Although the techniques mentioned above perform well on most object detection tasks, there is a reasonable scope of improvement for the TSDR task. Due to its insufficient fine-grain feature extraction properties, and lower sensitivity to the positioning and color resolution, original YOLOv4 can provide low detection accuracy with many missed detection and false object predictions. Moreover, the real-world traffic environment is generally complex. The performance of traffic detection is hampered by the complex environment and similar traffic signs. Also, traffic signs are of small size relative to the image captured from inside a vehicle, which is more difficult to detect than traffic signs of large size. Most publically available datasets contain images captured in broad daylight with minimum motion blur and noise. The real-world applications of TSDR models require high-performing models consistent at night. The issues mentioned before exaggerates in low illumination scenarios. Therefore, we need a better model which can address the aforementioned issues.

4.2 Grouped Convolution

The idea of Grouped Convolution or Filter groups was first used in the AlexNet [14]. The primary motivation, as explained by the authors, was to allow the training on smaller GPUs as it reduces the memory requirements and also allows model-parallelization across the GPUs. This idea was formalized further in the Deep Roots [12] paper and also claimed that grouped convolution can have a better representation than normal convolution based on finding correlations across filters.

Compared to traditional convolution, in grouped convolutions, the input channels and the filter channels are divided into group count (g) number of separate groups, with each group having decreased channels. The convolution operation is then performed individually on these input and filter groups. For example, consider the following: if the number of input channels is 6 and the number of filter channels of 12. For a normal, ungrouped convolution, the computation operations

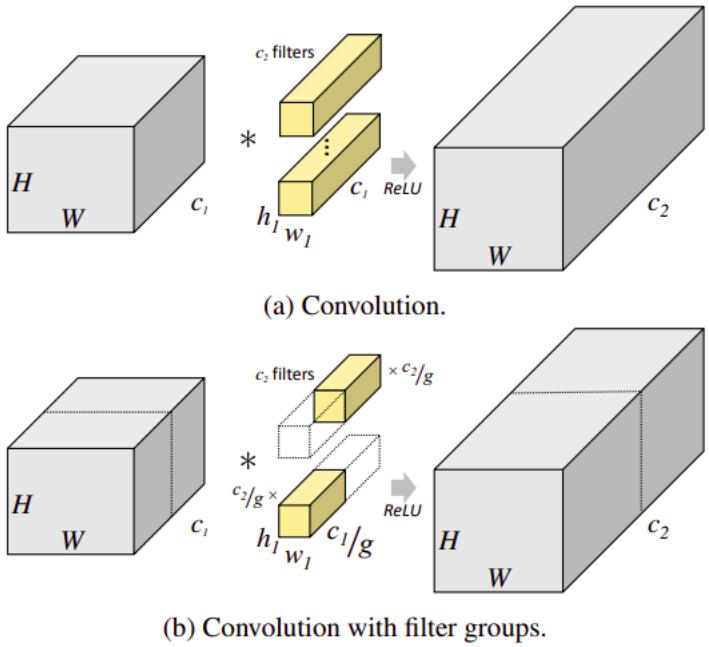


Figure 4.3: Filter Groups. (a) Convolutional filters (yellow) typically have the same channel dimension c_1 as the input feature maps (gray) on which they operate. However, (b) with filter grouping, g independent groups of $\frac{c_2}{g}$ filters operate on a fraction $\frac{c_1}{g}$ of the input feature map channels, reducing filter dimensions from $h \times w \times c_1$ to $\frac{h \times w \times c_1}{g}$ [12]

performed are $12 * 6$. If the g is set to 2, then there are two input channel groups of 3 input channels each and two filter channel groups of 6 filter channels each. As a result, each grouped convolution will perform $3 * 6$ computation operations, and two such grouped convolutions are performed. Hence the computation savings are 2x: $\frac{(12*6)}{(2*(3*6))}$. Figure 4.3 shows a generalized view of this grouped convolution.

Chapter 5

Proposed Architecture of Improved YOLOv4

In this chapter, we will try to resolve the issues mentioned above related to the real-time traffic sign detection and recognition problem. The state-of-the-art YOLOv4 algorithm is improved to enhance its ability to adapt to complex environments on Indian roads, improve the detection of adversarial conditions like noisy images, and deformed traffic signs, and enhance its sensitivity for small target detection. We propose four changes to the default YOLOv4 algorithm and fuse them in a single model, Improved YOLOv4. The complete schematic of the improved YOLOv4 network architecture is shown in Figure 5.2, and each modification is discussed in the following sections.

5.1 k-means Clustering using GIOU metric for Anchor Box calculation

YOLOv4 uses the anchor-based YOLOv3 head for detection. Anchor Boxes are also known as Boundary Box priors as they are predetermined before training of specific height and width. With the help of an anchor box, we can specialize our model to train on ideal shape, size, and location. The shape and size of the anchor boxes have a direct impact on the performance of the model as the bounding box is calculated with the help of the following formulae,

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
\tag{5.1}
$$

here, b_x, b_y, b_w and b_h are the center x, center y coordinates, width and height of the bounding box respectively. t_x, t_y, t_w and t_h are the predictions of the network, (c_x, c_y) is the coordinate of top left corner of the cell, and p_w and p_h are the height and width of the anchor box. Work by [25], employs traditional k-means clustering to calculate the best anchor boxes. By treating cluster centroid as the height, and width of the desired anchor box and the ground truth bounding boxes as the cluster data, authors use Intersection-over-Union (IoU) as the distance metric for the bounding box ($b1$) and cluster centroid ($b2$). The ratio of the area of the bounding box and predicted anchor box is used to represent the similarity of the prediction in object detection tasks. The distance between $b1$, $b2$, and IoU is represented as follows:

$$
\begin{aligned}
d(b1, b2) &= 1 - IOU(b1, b2) \\
IoU(b1, b2) &= \frac{|b1 \cap b2|}{|b1 \cup b2|}
\end{aligned}
\tag{5.2}
$$

Traffic signs are highly shaped dependent, and since anchor boxes are crucial for the shape of the detected bounding box, we propose to use GIoU (Generalized Intersection-over-Union) [28] as a distance metric for anchor box calculation. The new distance metric and GIoU are represented as follows:

$$
\begin{aligned}
d'(b1, b2) &= 1 - GIoU(b1, b2) \\
GIoU(b1, b2) &= IoU(b1, b2) - \frac{|c \setminus (b1 \cup b2|)}{|c|}
\end{aligned}
\tag{5.3}
$$

where c represents the area of smallest convex hull that encloses both $b1$ and $b2$. Figure 5.1 represents two cases while predicting the anchor box using the K-means algorithm. In the first case, both $d(b1, b2)$ and $d'(b1, b2)$ would be non-zero. But, in the second case, $d(b1, b2)$ would be zero. Moreover, irrespective of how far the blue bonding box is from the green one, $d(b1, b2)$ would be zero as long as they

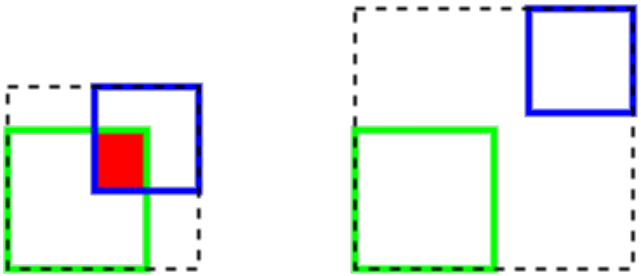


Figure 5.1: Example of overlapping vs non-overlapping bounding boxes. Green represents $b1$ and blue represents $b2$. Dotted black line represents smallest convex hull that encloses both $b1$ and $b2$

are non-intersecting. On the other hand, $d'(b1, b2)$ would be non-zero, and if the blue bounding moves further away, $|c|$ increase and consequently increase $d'(b1, b2)$. Thus, using GIoU, we can enable the k-means algorithm to push the blue bounding box in the second case to intersect the green bounding box, similar to the first case.

5.2 Improved PANet

PANet is used in YOLOv4 due to its ability to preserve spatial information, which helps in the proper localization of objects. As the neural network deepens, the complexity of the feature maps increases, and the spatial resolution of the image reduces. Due to this, the pixel-level masks cannot be identified accurately by the high-level features. Also, the location information becomes fuzzier, making the original YOLOv4 algorithm miss detecting small traffic signs precisely. Therefore, we propose a feature enhancement technique that fuses features from shallow layers of the CSPDarknet-53 backbone with semantic features of high-level layers of PANet to improve the precise positioning and sensitivity for small target detection.

The modifications carried out are highlighted in red in Figure 5.2. Compared with the original YOLOv4 (Figure 4.2), three layers are consequently fused with PANet layers instead of two. The fusion helps take advantage of spatial information from shallow layers and semantic information from deeper layers. The features of the 117th layer are fused with the 54th layer after up-sampling four times. The upsampling is necessary to match the dimensions. The features of the 127th layer are fused with the 23rd layer. We also increase the count of CBL blocks by one to account for the fact that we need to translate more amount of spatial information into high-level semantic information. As a result of the modifications, the first-scale feature has a dimension of $152 \times 152 \times H$ instead of $76 \times 76 \times H$, where H is the number

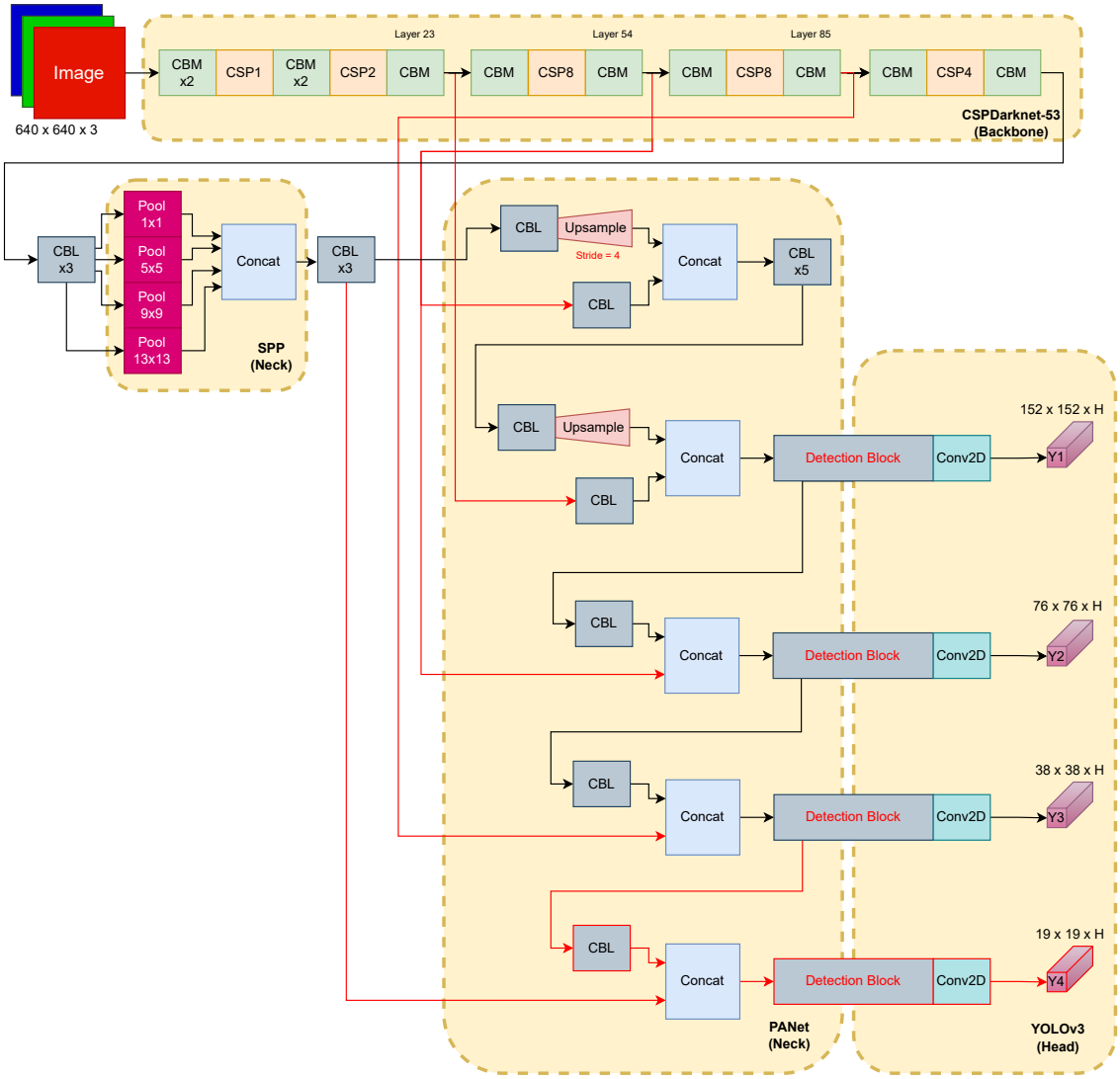


Figure 5.2: Improved YOLOv4 network architecture (Highlighted in red are the changes)

of filters depending on the count of classes of objects. Since the feature scale is large, it can efficiently detect very small objects. The second-scale feature dimensions are $76 \times 76 \times H$ responsible for small targets. The third-scale feature dimensions are $36 \times 36 \times H$, responsible for middle-sized targets. The fourth-scale or the extra-scale (Section 5.3) feature dimensions remain unchanged at $19 \times 19 \times H$ responsible for large targets. Since the proposed model does not disturb the network architecture of CSPDarknet-53, it guarantees that large traffic signs still have a high detection accuracy. At the same time, it integrates shallow features containing precise and prominent location information with the deep features containing superior semantic information, thereby improving the detection accuracy of traffic signs for most small targets.

5.3 Extra Feature Scale

The traffic sign images captured in the natural scene are affected by shape and color due to different environmental and light conditions. Moreover, in the vehicle-mounted camera, traffic signs captured in images are of small size compared to the whole image. Thus, default YOLOv4 can provide low detection accuracy, leading to an increased number of missed detection and false object prediction due to insufficient small-scale feature extraction for the TSDR problem. Therefore, motivated by [36], we add one more layer to the detection head to help the model make detections on a finer level.

The extra layer is highlighted by red in Figure 5.2. Compared to Figure 4.2, we can see that now we have four different feature scales of sizes 152×152, 76×76, 38×38, and 19×19, which helps in detecting traffic signs of various sizes. Due to changes in PANet, the size of the input feature map for the last layer of improved YOLOv4 is the same as that of the default YOLOv4. Due to the increased number of detection heads, we can extract more spatial information, but there is a trade-off in which we can lose gradient information. Since one of the inputs in the extra concat block added is from shallow layers, it removes the possibility of losing information in the extra layer added. The second input is from the third residual block of YOLOv4, which helps keep the network structure and pattern similar to previous detection heads. The concat block is followed by alternate convolution layers of 3×3 and 1×1 kernel sizes, the output of which is passed to the detection head. In this way, a fourth and extra feature scale for prediction is established.

5.4 Detection block using Grouped Convolution

The last CBL blocks just before the detection Head in Figure 4.2 are used for feature propagation from complex semantic features outputted by the concat layer to features required by the linear block in the YOLO detection head. But, the CBL block is a straightforward block of Convolution followed by Batch Normalization and Leaky Relu activation function. There is a scope for improvement in this area where we can learn better and more efficient representations of the feature map. Due to previous changes, we can see that the parameters of the networks have been significantly increased. Motivated by [12], to overcome these issues and to make our network more efficient, we introduce Grouped Convolutions in Detection Block, as shown in Figure 5.2. We call Detection Block the combination of all the CBL

blocks used before the detection head of default YOLOv4. Figure 6.7a shows the detection block containing CBL blocks as per original YOLOv4. Figure 6.7d shows the improved detection block using Grouped Convolution that will be used in our proposed YOLOv4 model. In the last four layers of the detection block, we add grouped convolutions to the layers having kernel size 1×1. The convolutions of kernel size 3×3 are kept unchanged.

The above four changes combined is our proposed YOLOv4 model for the TSDR task. The comparison of our network with the other state-of-the-art models is shown in Chapter 6.

Chapter 6

Experimental Analysis

In this chapter, we discuss the results and analysis of the model. We start by mentioning the datasets used for the experiment and the metrics used for evaluating the performance, followed by the experimental setup and results. We carry ablation study to learn more about the proposed changes.

6.1 Datasets

Figure 6.1: Different traffic signs present in MTSD dataset (The sizes are relative to the number of samples within MTSD)[4]

To verify the efficiency and performance of our model for the TSDR task, we selected three different established datasets, viz. Mapillary Traffic Sign Dataset (MTSD) [4], Tsinghua-Tencent 100K (TT100K) dataset [39], and The German Traffic Sign Detection Benchmark (GTSDB) [9]. These datasets are further discussed in the sections below.

Table 6.1: Details of MTSD Dataset

	Classes	Images
Training	401	36589
Validation	381	5320
Testing	400	10544

6.1.1 Mapillary Traffic Sign Dataset (MTSD)

This is one of the most recent and largest dataset in the field of TSDR tasks. It is a global and diverse dataset encapsulating comprehensive coverage of geographical locations and varying weather and lighting conditions. Since this dataset is new, no prior research has been conducted on this dataset. We use this dataset to test the robustness and adaptability of our model to traffic signs from different countries. Table 6.1 shows the details of this dataset. This dataset consists of 52K labeled images that covers more than 300 manually annotated traffic sign classes and more than 82K instances of traffic signs. The dataset is split into three sets: training, validation, and testing, as shown in Table 6.1. Since there are more than 300 classes in this dataset, we have divided each class into four groups, namely, warning, information, regulatory and complementary, to make the labels applicable across the GTSDB dataset for Cross Dataset Validation (Chapter 7).

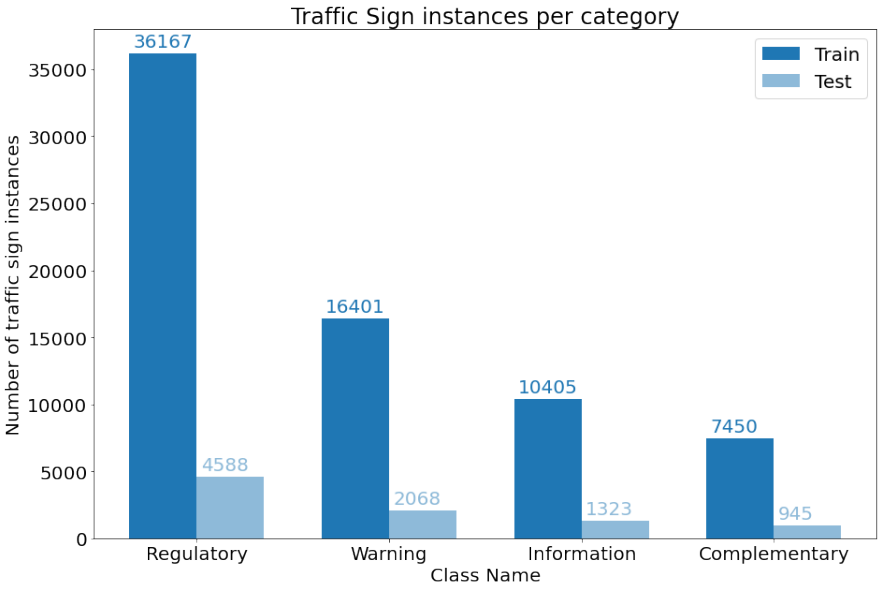


Figure 6.2: Number of traffic sign instances per category in each of the 4 classes of MTSD dataset

6.1.2 Tsinghua-Tencent 100K (TT100K) dataset

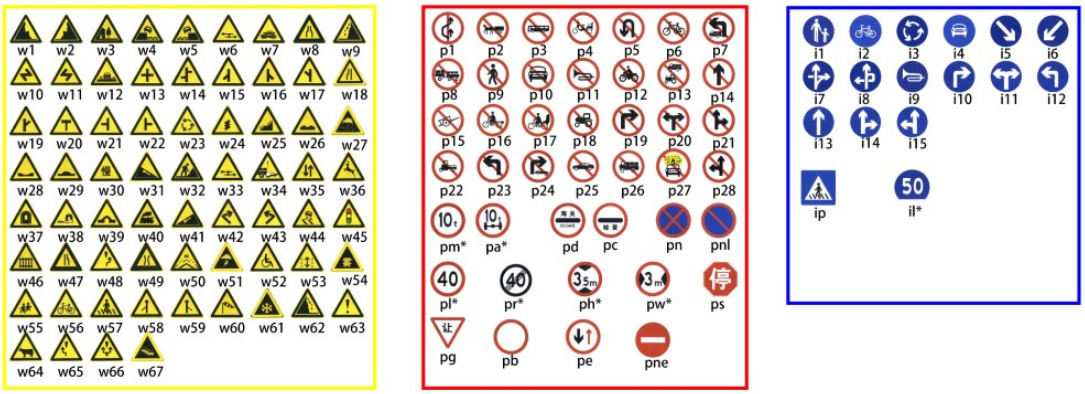


Figure 6.3: TT100K classes (Signs in yellow, red and blue boxes are warning, prohibitory and mandatory signs respectively) [39]

This is a Chinese Traffic Sign dataset and is popularly used in TSDR tasks. It is used to compare our model with other state-of-the-art networks and prior works. This dataset consists of 6k+ images for training and 2k+ images for testing. Each traffic sign has a unique label, and there are 200 different labels of traffic signs. Some signs symbolize a family (e.g., speed limit signs for different speeds). Such signs are generically denoted above (e.g. 'pl*'); the unique label is determined by replacing '*' with a specific value (e.g. 'pl40' for a 40 km/h speed limit sign). Although there are 200 different classes of traffic signs, we take the top 44 categories with the most traffic signs for a fair comparison with previous works [36, 5] and to deal with the severe imbalance in the number of instances in each category. Table 6.2 shows the number of images/instances in each category.

6.1.3 German Traffic Sign Detection Benchmark (GTSDB)

This is one of the first datasets for the TSDR task. It consists of 900 images, split into two parts, a training set with 600 images and a testing set with 300 images. The classes in this dataset are divided into four categories which are the prohibitory, danger, mandatory and other. Due to the small size of the dataset, we use this dataset for only Cross Dataset Validation (Chapter 7), with a total of 900 images. This will help test the robustness of our proposed model in a real scenario since the distribution of the training and testing datasets would be different.

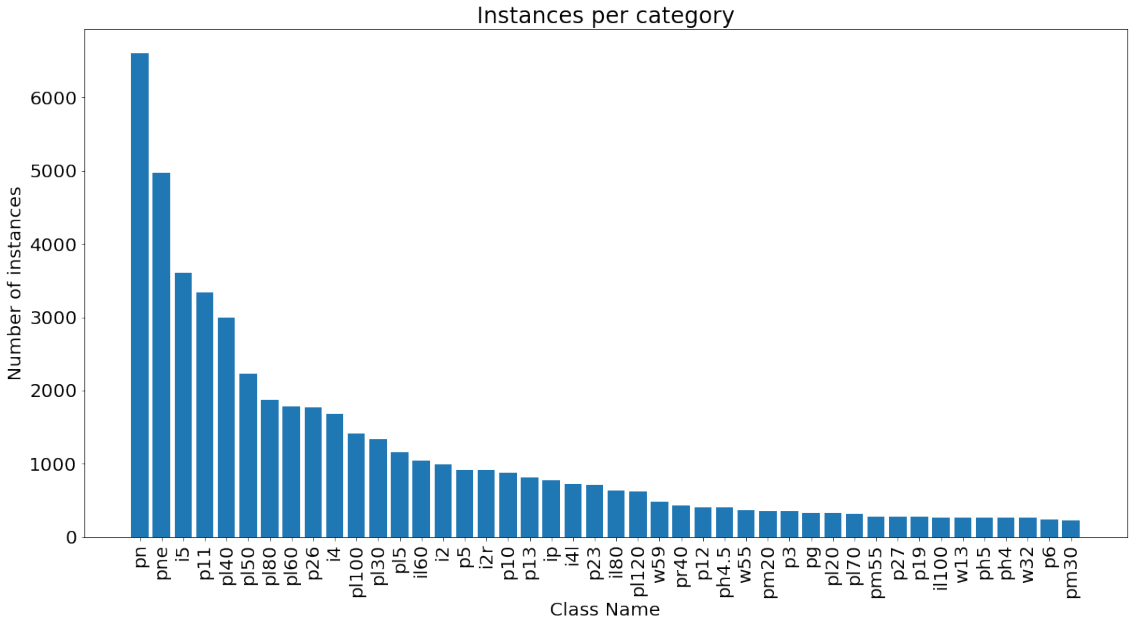


Figure 6.4: Number of images per category in each of the 44 classes of TT100K dataset

Table 6.2: Number of images in each class of TT100K dataset

Class	**Number of images**	**Class**	**Number of images**	**Class**	**Number of images**	**Class**	**Number of images**
pn	6601	pl30	1334	il80	629	pl70	314
pne	4974	pl5	1155	pl120	620	pm55	281
i5	3605	il60	1041	w59	488	p27	278
p11	3338	i2	989	pr40	428	p19	274
pl40	2992	p5	914	p12	404	il100	270
pl50	2232	i2r	911	ph4.5	402	w13	269
pl80	1877	p10	874	w55	369	ph5	268
pl60	1777	p13	814	pm20	359	ph4	266
p26	1771	ip	771	p3	354	w32	263
i4	1679	i4l	727	pg	334	p6	238
pl100	1419	p23	710	pl20	327	pm30	231

6.2 Performance Metric

To measure the performance of our proposed model, we use several traditional metrics [6] such as Precision, Recall, F1-score, Average Precision (AP), and mean Aver-

age Precision (mAP). For each class, we calculate True Positive (TP) as the object is present, and the model successfully detects the object with the IoU of the predicted bounding box against the ground truth box above a certain IoU threshold. False Positive (FP) as the object is present, but the IoU of the predicted box against the ground truth box is less than the IoU threshold or model predicts a box even though there are no objects present. False Negative (FN) as the object is present, but the model is not able to predict the object. In the object detection task, True Negative (TN) is not defined. Precision (P), Recall (R), and F1-score (F) are defined by the following formulae.

$$\begin{aligned} P &= \frac{TP}{TP + FP} \\ R &= \frac{TP}{TP + FN} \\ F &= \frac{2PR}{P + R} \end{aligned} \tag{6.1}$$

High precision implies that most predicted objects match ground truth objects, while high recall indicates most ground truth objects were correctly predicted. F1-score tells us about the degree of precision. For a particular task, the Precision-Recall curve (P-R curve) is defined as the curve with recall on the x-axis (abscissa) and precision on the y-axis (ordinate) plotted for certain confidence thresholds. AP Average Precision is the area under the P-R curve.

$$\begin{aligned} AP &= \int_0^1 P(R)dR \\ mAP &= \frac{1}{N} \sum^{N} AP \end{aligned} \tag{6.2}$$

here, N is the number of classes. We define APα as the value of AP at IoU threshold α [13]. $AP_{50:95}$ or AP is defined as the average precision over of the range of 0.50:0.05:0.95 which means values of IoU threshold ranging from 50% to 95% with step size 5%. mAP is defined as the mean of average precision for each class.

6.3 Experimental Settings

The experiments are performed on Ubuntu 20.04 operating system with a processor of Intel(R) Xeon(R) Silver 4214 and Nvidia RTX 3090 24GB GPU. For the imple-

mentation of Darknet, we are using a publicly available repository [3] written in C. For training, we are using a batch size of 64 and 32 subdivisions. The number of training steps is 20,000, and the step decay rate scheduling strategy is used with an initial learning rate of 0.001 and a decay of factor 0.1 at steps 16,000 and 18,000. The momentum and decay are set to 0.949 and 0.0005, respectively. We are also using the mosaic data augmentation technique along with changes in saturation, exposure, and hue.

Table 6.3: Size of anchor boxes using IoU and GIoU as distance metric on TT100K dataset

Feature Scale	Distance Metric	Anchor Boxes
3 Layer	IoU	7, 8, 11, 12, 15, 16, 13, 26, 21, 22, 28, 32, 39, 44, 55, 60, 89, 87
	GIoU	8, 9, 11, 12, 15, 17, 13, 28, 21, 23, 29, 32, 39, 44, 55, 59, 88, 86
4 Layer	IoU	7, 8, 10, 11, 12, 15, 17, 19, 13, 27, 22, 24, 27, 31, 36, 37, 40, 50, 55, 54, 65, 76, 99, 91
	GIoU	8, 9, 11, 12, 14, 16, 12, 25, 19, 21, 25, 26, 31, 33, 25, 52, 40, 43, 54, 54, 65, 77, 103, 95

Table 6.3 shows the anchor boxes that are used in the experiments. We achieve an average IoU of 83.15% and 82.15% for the 4-layer feature scale using IoU and GIoU as distance metrics, respectively. For the 3-layer model, we get 80.96% for the IoU distance metric and 80.82% for the GIoU distance metric. We are using greedy NMS for Non-Max Suppression and CIoU for bounding box regression loss.

6.4 Results and Model Analysis

In order to demonstrate the effectiveness of our model, we compare our results with previous state-of-the-art models in Table 6.4 on the TT100k dataset. In the detection task, various different metrics are used to compare works, and to overcome this challenge; we use a mean Average Precision (mAP) as our common performance metric for a fair comparison. We use the following baselines:

Table 6.4: Comparison of Different Models on TT100K dataset

Model	mAP (%)
Zhu et. al. [39]	88
Original YOLOv4 [3]	93.00
Luo et. al. [19]	94
Tang et. al. [33]	94.5
Proposed Method	**94.73**

- Zhu et. al. [39]: In this research work, TT100K dataset is proposed along with a robust convolutional neural network (CNN) model.

- Original YOLOv4 [3]: We use the original implementation of YOLOv4 to serve as the baseline. For the calculation of the Anchor box, we are using the k-means clustering method with IoU as a distance metric.

- Luo et. al. [19]: In order to improve accuracy on smaller objects, Contextual-YOLOv3 is proposed with utilizes a contextual relationship matrix combined with the classification probability of YOLOV3.

- Tang et. al. [33]: Based on statistical features of traffic signs, Integrated Feature Pyramid Network with Feature Aggregation (IFA-FPN) is proposed in this paper with Cascade-RCNN as detector head.

Table 6.4 shows the results of these baselines and our proposed model. We can see that our model outperforms the existing models on the TSDR task. Zhu et. al. [39] used traditional CNN for the TSDR task. Since CNN networks are known to perform poorly for object detection tasks, it achieves a low accuracy of 88%. On the other hand, the original YOLOv4 performs much better and achieves an accuracy of 93%. Luo et. al. [19] achieve 94% mAP, but they considered fewer classes for the TT100K dataset, making the model more straightforward. Our improvements concurrently give an increase of 0.23% mAP over the current state-of-the-art model Tang et al. [33]. But, this model was computationally expensive and achieved around two frames per second during inference. Our model achieves high frames per second, shown in Table 6.9, indicating that our model has an edge in both speed and accuracy over the current state-of-the-art.

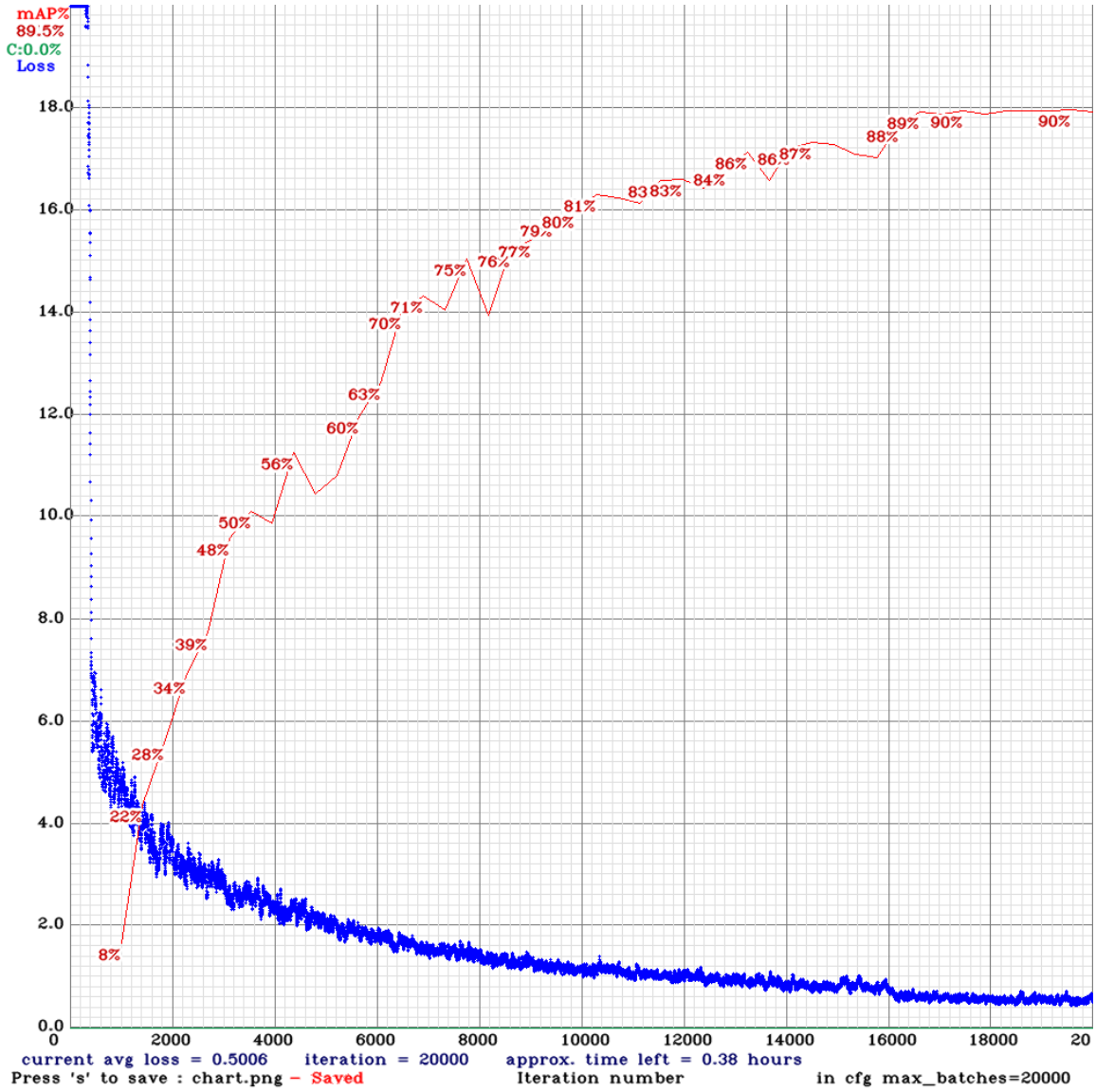


Figure 6.5: Training Accuracy and Loss on TT100K dataset for the proposed model.

Figure 6.5 shows the training trend on accuracy and loss of our proposed model for 20k steps. Our model achieves a high training accuracy of 90%, and the testing accuracy of 94.73% suggests a high generalization capability of our model.

Table 6.5 shows the detailed description of precision, recall, and AP for each of the classes on the TT100K dataset. Classes with more number of images have a higher AP, indicating that given there are sufficient traffic sign instances during training, our model will learn to predict those signs accurately. Moreover, recall is generally higher than precision, meaning there are fewer false negatives than false positives. TSDR tasks tend to require high recall than precision, and our model satisfies this condition.

Table 6.5: Classwise Results on TT100K dataset

Class Name	Average Precision (AP %)	Precision	Recall
pn	97.70	0.93	0.96
pne	97.98	0.96	0.98
i5	98.14	0.95	0.97
p11	95.91	0.88	0.93
pl40	96.32	0.87	0.95
pl50	95.86	0.89	0.93
pl80	97.77	0.88	0.96
pl60	95.95	0.89	0.93
p26	94.98	0.86	0.94
i4	97.12	0.92	0.96
pl100	99.66	0.94	1.00
pl30	95.98	0.80	0.93
pl5	96.55	0.82	0.98
il60	99.61	0.93	0.99
i2	94.26	0.85	0.93
p5	97.81	0.88	0.98
i2r	94.17	0.83	0.93
w57	97.41	0.88	0.97
p10	93.67	0.88	0.92
p13	90.59	0.81	0.91
ip	97.49	0.94	0.95
i4l	97.61	0.83	0.96
p23	96.78	0.80	0.96
il80	99.28	0.89	0.99
pl120	99.21	0.97	0.99
w59	90.49	0.77	0.97
pr40	99.77	0.88	1.00
p12	94.11	0.63	0.96
ph4.5	93.18	0.60	0.95
w55	86.56	0.75	0.86
pm20	93.34	0.47	0.94
p3	88.53	0.84	0.80
pg	96.84	0.90	0.96
pl20	92.63	0.73	0.91
pl70	85.23	0.77	0.80
pm55	95.41	0.67	0.95
p27	96.30	0.83	0.96
p19	92.27	0.55	0.91
il100	98.57	0.93	1.00
w13	89.33	0.60	0.90
ph5	76.91	0.50	0.79
ph4	88.22	0.47	0.89
w32	96.21	0.86	0.97
p6	96.79	0.55	0.97
pm30	94.19	0.40	0.97

Figure 6.6: Experimental results on different test images of TT100k dataset (Label shows class name, confidence score)

Figure 6.6 shows some examples of our model predictions on a test set of the TT100K dataset. In the first example, we can see that the size of traffic signs is small relative to the image size. But, after zooming into the image, we can see that our model accurately predicts the "No Honking (p11) sign" with 100% confidence. It also predicts the "Speed Limit 30" sign though the size of the traffic sign is small. Our model is accurate in predicting even small signs, but in this example, it had lower confidence of 31%, signifying the importance of the size of traffic signs for prediction. In the second example, we can see that our model correctly identifies all three traffic signs with high confidence even though they are in a small vicinity, signifying the high localization capability of our model. In the third example, we can see that our model can distinguish between traffic signs and directions accurately and predicts the signs again with high confidence.

Table 6.6: Classwise Results on MTSD dataset

Class Name	Average Precision (AP %)	Precision	Recall
regulatory	84.25	0.74	0.84
warning	87.97	0.75	0.95
information	72.48	0.64	0.72
complementary	78.12	0.67	0.78

Since there is no existing work on the MTSD dataset, we cannot compare it with the existing works. But, our best accuracy on this dataset is 80.71%. The aggregate results are further discussed during the ablation study. (Section 6.5). Table 6.6 shows the classwise AP, precision and recall for MTSD dataset. The AP for MTSD dataset is slightly lower than TT100k due to a variety of reasons. First, because of the change in labels, the distribution of images within a single class has been changed. Secondly, it is difficult to learn the pattern for a single class owing to the complexity of the MTSD dataset, such as signs from different nations for the same class. But, the model has a higher recall than precision, similar to the results for the TT100K dataset.

6.5 Ablation Study

In this section, we analyze the effect of individual components on our proposed model. We further mention about the study on detection block using different

techniques like MobileNet, DenseNet and Grouped Convolution and report accuracy using mAP and speed using Frames Per Second (FPS).

6.5.1 Ablation study of Individual components

We evaluate the performance of each individual improvement on the TT100K and MTSD datasets. The results are summarized in Table 6.7 and Table 6.8. Original YOLOv4 is taken as the baseline for our study. The results of each of the components on the TT100k dataset are as follows:

- k-means Clustering using GIOU metric for Anchor Box calculation: With the inclusion of this change, we achieve a performance improvement of over 0.75% over the baseline model. With this, we infer that anchor boxes calculated with GIoU as distance metric are more generalized for detection tasks compared to IoU metric.
- Improved PANet: By changing the connections in the PANet, we specifically improved the performance for fine-grained objects, due to which resulted in a 1.25% gain in detection accuracy over the baseline model.
- Extra Feature Scale: Addition of the Extra-Feature Scale is useful as it helps in increasing the number of detections by the model on pixel-wise feature masks. This improves the performance by 0.15% over the baseline model.
- Detection block using Grouped Convolution: By using Grouped Convolution in detection block instead of normal convolution, we can increase the overall performance of our model. We can further use different architectures like MobileNet [10] and DenseNet [11] for the detection blocks. The results for this are discussed in Section 6.5.2.

Table 6.7: Ablation study of individual components on TT100K dataset

Anchor Boxes using GIoU	Improved PANet	Extra Feature Scale	Grouped Convolution	Precision	Recall	F1-Score	mAP @0.50 (%)
-	-	-	-	0.82	0.95	0.88	93.00
✓	-	-	-	0.84	0.94	0.89	93.75
-	✓	-	-	0.86	0.94	0.90	94.25
-	-	✓	-	0.83	0.95	0.88	93.15
-	-	-	✓	0.83	0.94	0.88	93.09
✓	-	✓	-	0.85	0.94	0.89	93.33
✓	✓	✓	-	0.84	0.95	0.89	94.50
✓	✓	✓	✓	0.86	0.95	0.90	**94.73**

For the detection on the MTSD dataset, we make incremental improvements since the size of the dataset is large and training is time expensive. As we can see in the table 6.8, we achieve the best result of 80.71% when we only apply the first three improvements to the model. Overall, we achieved an improvement of 1.09% compared to our baseline model. While adding grouped convolution to the former gains, we get an accuracy drop of 0.04%, but since the difference is quite small, we can infer that the effect of the change is quite negligible.

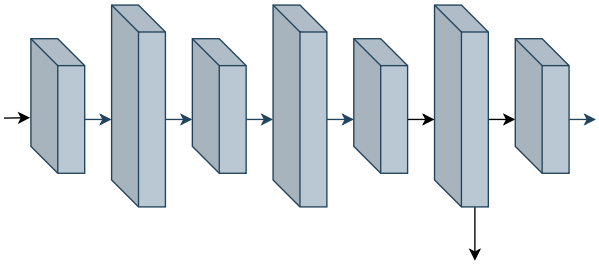
Table 6.8: Ablation study on individual components on MTSD dataset

Anchor Boxes using GIoU	Improved PANet	Extra Feature Scale	Grouped Convolution	Precision	Recall	F1-Score	mAP @0.50 (%)
-	-	-	-	0.73	0.83	0.77	79.62
✓	-	-	-	0.72	0.84	0.78	79.94
✓	✓	-	-	0.73	0.83	0.78	80.38
✓	✓	✓	-	0.72	0.84	0.78	**80.71**
✓	✓	✓	✓	0.72	0.84	0.78	80.67

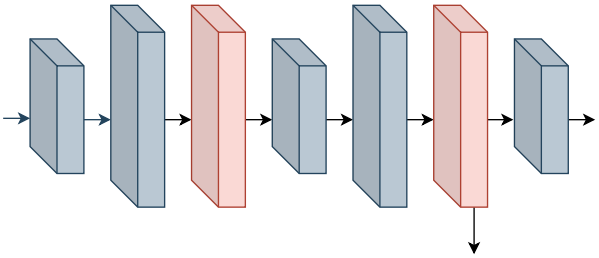
6.5.2 Ablation study on Detection Block

We conduct ablation on different detection blocks to study the effect of changing detection blocks for the TSDR task. The detection block is the last stage of feature extraction, after which the features are passed through the detection head, YOLOv3 in our case, which generates the final output. Therefore, detection block is a crucial part of detection and recognition. Consequently, we have experimented with different models in order to improve semantic feature extraction and better transmission of data from shallow layers. Figure 6.7 shows the various detection blocks tested. These detection blocks are discussed below.

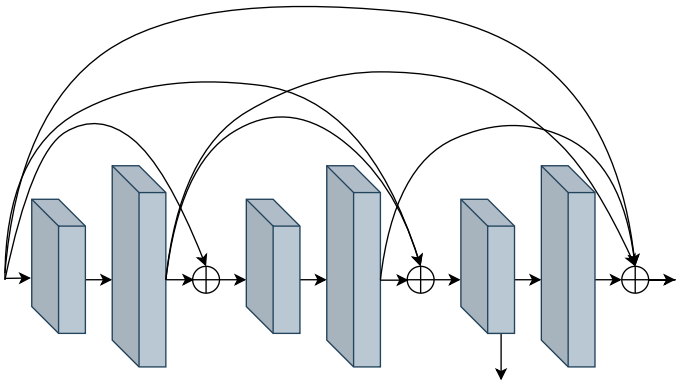
- Default: We call default detection block as the connections implemented in the original YOLOv4. This consists of 5 alternate CBL blocks with kernel size 1×1 and 3×3 followed by 1 CBL block (as shown in Figure 4.2). The dimension of feature matrix is also doubled after each alternate CBL block as shown in Figure 6.7a.
- MobileNet: Motivated by [10], we introduce MobileNet like blocks wherein the first block is normal convolution with kernel size 1×1, followed by normal convolution with kernel size 1×1 but with feature map size doubled and finally grouped convolution block with kernel size 3×3. This set of three blocks is repeated twice, as shown in 6.7c.



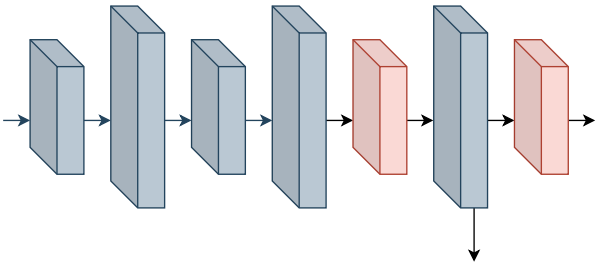
(a) Default



(b) MobileNet



(c) DenseNet



(d) Grouped Convolution

Figure 6.7: Different Detection Blocks (Grey blocks are CBL blocks, and Red blocks are CBL blocks with grouped convolution)

- DenseNet: Motivated by [11], we use a different connectivity pattern on the default detection block. We use direct connections from one set of two consecutive CBLs block to all subsequent sets, as shown in Figure 6.7c. We implement dense connections for the set of two consecutive blocks, which have kernels of

size 1×1 and 3×3, respectively.

- Grouped Convolution: As explained in Section 5.4, we have used grouped convolutions for blocks with kernel size 1×1 as shown in 6.7d.

Table 6.9: Ablation study on Detection Block (TT100K dataset)

Detection Block	Precision	Recall	F1-Score	mAP @0.50 (%)	FPS
Default	0.82	0.95	0.88	93.00	28.69
MobileNet	0.82	0.94	0.88	92.98	28.97
DenseNet	0.83	0.95	0.88	94.30	27.62
Grouped Convolution	0.83	0.94	0.88	93.09	**30.44**
Proposed Method	0.86	0.95	0.90	**94.73**	19.50

The precision, recall, F1-score, and mAP of each of the detection blocks used in YOLOv4 models are shown in Table 6.9. We have changed only the detection blocks for the first four experiments, keeping the rest of the model the same as the original YOLOv4. In contrast, the Proposed Model consists of all the changes mentioned in Chapter 5. We have also reported Frames per Second (FPS) as a measure of the speed of models corresponding to each detection block. The detection block using MobileNet has a high FPS of 28.97 as MobileNet architecture is efficient with fewer parameters but has a slight trade-off with accuracy and obtains a drop of 0.02% in mAP compared to the Default detection block. On the other hand, DenseNet has a drop in FPS to 27.62 but has a higher accuracy of 94.30% owing to the dense connections leading to better information flow but more parameters to handle. Grouped Convolution detection block has the highest FPS of 30.44. It performs better in accuracy than Default and MobileNet-based models, but when merged with other proposed modifications, it even outperforms the DenseNet-based model. Since the Proposed Method has an extra feature scale and changes in PANet, it incurs a drop in speed, giving 19.50 FPS.

Chapter 7

Cross Dataset Validation

In this chapter, we conduct cross dataset experiments to test the efficiency and robustness of our model on the data, which has different distribution compared to training data. We use models trained on the MTSD dataset since this dataset is global, and the model trained on the MTSD dataset has a better chance of predicting traffic signs from GTSDB and ITSD datasets. We also use Image Enhancement techniques for the ITSD dataset since there are many images with low illumination and noise in this dataset. The details of each of the experiments are described below.

7.1 Image Enhancement for Night-Time Images

Traffic sign detection and recognition problems depend highly on the features obtained from the traffic sign image, and it is necessary to have an excellent illuminated input image. The publically available datasets have input images captured during the daytime and mainly have a clear vision of traffic signs. But the real world implementations of TSDR, such as autonomous driving and advanced driver-assistance systems, require high-performance models even at night. Therefore, we included nighttime images in our ITSD dataset to improve the performance of existing and proposed systems in actual scenarios. The illumination of the images captured at night is not enough for a model to extract the characteristics of traffic signs. To overcome the problem in nighttime images, we use a Nighttime low illumination image enhancement technique proposed in [31] to pre-process our ITSD images for conducting cross dataset experiments (Section 7.3). Figure 7.1 shows the framework for nighttime illumination. It uses a single image-based bright dark channel prior method for image enhancement. This method is based on the concerned image's

dark channel-prior and bright channel-prior. It utilizes the bright channel prior to get an initial transmission estimate and then uses the dark channel as a complementary channel to correct potentially inaccurate transmission estimates acquired from the bright channel prior. Figure 7.1 shows the method used for the illumination of images. The advantage of using this model is that it increases the illumination and slightly lowers noise from the images using Gaussian blur.

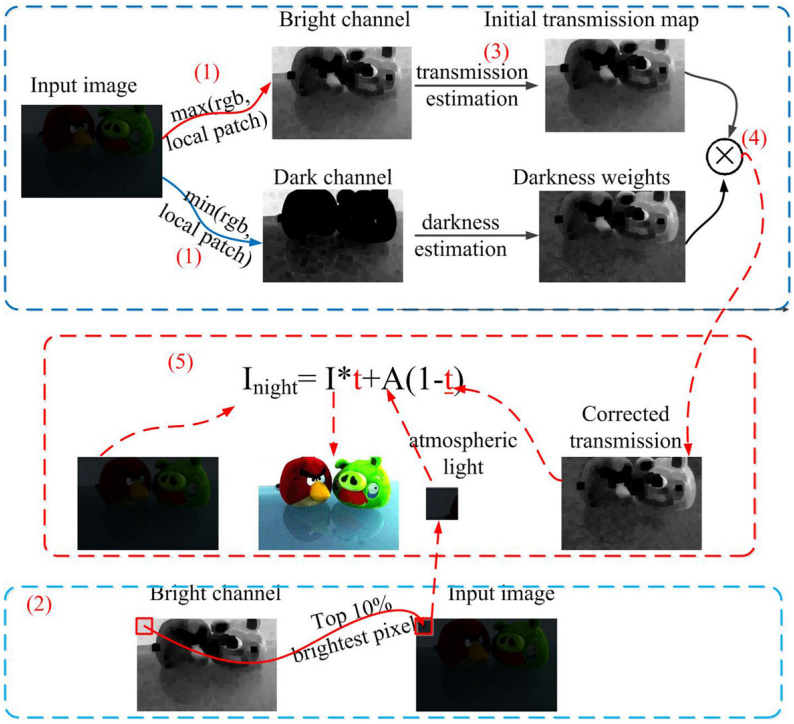


Figure 7.1: Nighttime Illumination model [31]

Table 7.1: Speedup for Nighttime Illumination Model. The time reported is amortized time per image of size 800×1360.

Improvement	Time (in seconds)	Speedup
-	46.70s	1x
Vectorized Operations	2.17s	21x
Parallel Computing	0.11s	420x

One disadvantage of using this model is that the open source implementation [1] takes 46 seconds to process an RGB (3 channel) image of size 800×1360. This time is unsuitable as our model works in real-time with high frames per second. Therefore, we have further optimized the implementation of the nighttime illumination model using vectorized and parallel operations. The speedup is mentioned in Table 7.1.

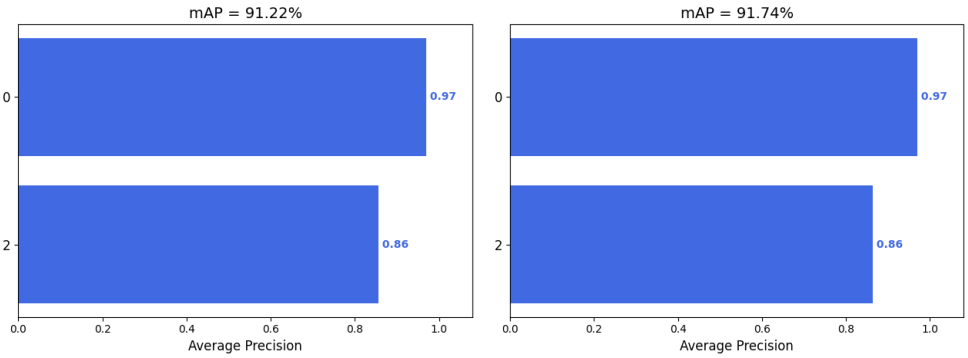
Figure 7.2: Sample Images from ITSD dataset. Left side are original images and Right side are images after passing through night-time illumination model

7.2 Results on GTSDB Dataset

GTSDB dataset has labels that are different from the labels of the MTSD dataset on which our model is trained. Therefore, we introduce a mapping of GTSDB labels to MTSD labels required for evaluation. We calculate this mapping by first feeding all the images of the GTSDB dataset to our trained model. Then this model gives detection results for each image. We then use the ground truth values, along with the detection values of each image, to calculate the IoU_{true}^{pred}. Then the class label of GTSDB is assigned a class from one of the 4 classes of MTSD dataset having the maximum IoU_{true}^{pred}. This helps to ensure a one-to-one mapping of each traffic sign instance from the GTSDB dataset to one of the labels in the MTSD dataset. Table 7.2 shows the final mapping calculated.

Table 7.2: Class label mapping from GTSDB dataset to MTSD dataset

GTSDB Label	**MTSD Label**
Prohibitory	Regulatory
Danger	Warning
Mandotory	Warning
Other	Regulatory and Warning



(a) YOLOv4 (b) Proposed Method

Figure 7.3: Comparison of classwise mAP on GTSDB dataset

Figure 7.3 shows the comparison of results of our proposed model and traditional YOLOv4. Even though both models were unfamiliar with the GTSDB dataset, we can see that YOLOv4 trained on the MTSD dataset efficiently detected traffic signs across the GTSDB dataset with an mAP of 91.22%. We can also see that our proposed model performs better than traditional YOLOv4 and has an improvement of 0.5% mAP.

Figure 7.4: Experimental results of Cross Dataset Validation on GTSDB dataset (Label shows class name, confidence score)

7.3 Results on ITSD Dataset

ITSD is our curated dataset on Indian Traffic signs. We have tested our proposed model on this dataset. For this dataset, we have calculated the detection accuracy only rather than classwise results since this dataset is at a nascent stage and class labels are yet to be verified. The results are shown in Table 7.3. We can see that

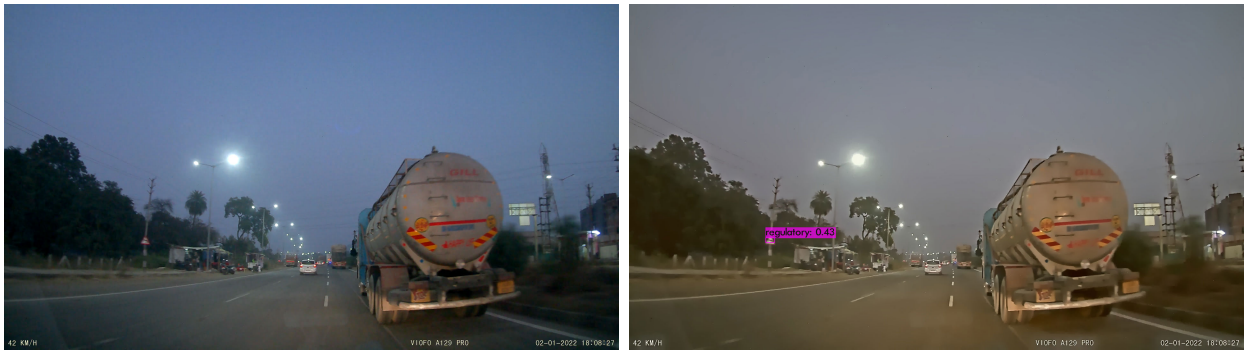
the models with and without night illumination processing give comparable results. This is because the ITSD consists of daytime images on which the performance of the night illumination model is not optimal.

Table 7.3: Results for cross-dataset experiment on ITSD

Model	Precision	Recall	F1-Score	Accuracy (%)
Proposed Method	0.77	0.81	0.79	64.86
Proposed Method with night illumination	0.77	0.80	0.79	64.65

Some sample images and their detection results for this dataset are shown in Figure 7.5. We have captured many images in night conditions and images with motion blur, which is a challenging task for object detection models. In Figure 7.5a, we can see there is a sign on the right side of the original image that is blurred, but after using the night illumination model, In Figure 7.5b, the model can predict the traffic sign. Similarly, In Figure 7.5c, the traffic sign is dark, due to which our model is unable to predict the sign, but In Figure 7.5d, we can see the sign is brightened due to illumination model and thus our model predicts the traffic sign. For daytime image in Figure 7.5e and in Figure 7.5f, there is a slight improvement in the confidence score. Finally, In Figure 7.5h, we can see that our illumination model hinders the performance of the model compared to the original image in Figure 7.5g, and one traffic sign is missed due to our illumination model.

(a) (b)



(c) (d)

(e) (f)

(g) (h)

Figure 7.5: Experimental results of Cross Dataset Validation on ITSD dataset (Label shows class name, confidence score). Left side are original images and Right side are images after passing through night-time illumination model

Chapter 8

Conclusion and Future Work

In our work, we proposed an Improved YOLOv4-based model that performs well on Traffic Sign Detection and Recognition (TSDR) tasks. We used Tsinghua-Tencent 10 (TT100k) and Mapillary Traffic Sign Data (MTSD) datasets and achieved an mAP of 94.73% and 80.71%, respectively. We showed that our model outperforms several previous state-of-the-art models on the TT100k dataset. Further to study the improvements, we carried out a detailed ablation of the experiments on individual components of the proposed changes and detection block. Lastly, to check the robustness of our model, we performed Cross-Dataset Validation on German Traffic Sign Detection Benchmark (GTSDB) and Indian Traffic Sign Dataset (ITSD). We verified that our model is powerful and can identify traffic signs belonging to different distribution than training data. As training on the MTSD dataset was expensive, we think that we can further enhance the model by hyper-parameter tuning. We can further work on improving frames per second (FPS) and better illumination models for the low-light images. We can improve the quality of the ITSD dataset and make it standardized with proper experimentation and verification of labels.

Bibliography

[1] V. Aggarwal and L. Patnaik. Improving illumination in night time images, Online Available: https://learnopencv.com/improving-illumination-in-night-time-images/. 2021.

[2] Aleksej Avramović, Davor Sluga, Domen Tabernik, Danijel Skočaj, Vladan Stojnić, and Nejc Ilc. Neural-network-based traffic sign detection and recognition in high-definition images using region focusing and parallelization. *IEEE Access*, 8:189855–189868, 2020.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.

[4] Christian Ertler, Jerneja Mislej, Tobias Ollmann, Lorenzo Porzi, and Yubin Kuang. Traffic sign detection and classification around the world. *CoRR*, abs/1909.04422, 2019.

[5] Zhang Gan, Li Wenju, Chu Wanghui, and Su Pan. Traffic sign recognition based on improved yolov4. In *2021 6th International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS)*, volume 6, pages 51–54. IEEE, 2021.

[6] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In David E. Losada and Juan M. Fernández-Luna, editors, *Advances in Information Retrieval*, pages 345–359, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[9] Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288, 2013.

[10] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[11] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016.

[12] Yani Ioannou, Duncan Robertson, Roberto Cipolla, and Antonio Criminisi. Deep roots: Improving cnn efficiency with hierarchical filter groups. 07 2017.

[13] Kiprono Elijah Koech. Object detection metrics with worked example, Online Available: https://towardsdatascience.com/on-object-detection-metrics-with-worked-example-216f173ed31e. 2020.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[15] Feng Lin, Yan Lai, Lan Lin, and Yuxin Yuan. A traffic sign recognition method based on deep visual feature. In *2016 Progress in Electromagnetic Research Symposium (PIERS)*, pages 2247–2250, 2016.

[16] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018.

[17] Ke Lu, Zhengming Ding, and Sam Ge. Sparse-representation-based graph embedding for traffic sign recognition. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1515–1524, 2012.

[18] Ke Lu, Zhengming Ding, and Sam Ge. Sparse-representation-based graph embedding for traffic sign recognition. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1515–1524, 2012.

[19] Han-Wu Luo, Cheng-Song Zhang, Fu-Cheng Pan, and Xiao-Ming Ju. Contextual-yolov3: Implement better small object detection based deep learn-

ing. In *2019 International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI)*, pages 134–141, 2019.

[20] Jun Miura, Tsuyoshi Kanda, and Yoshiaki Shirai. An active vision system for on-line traffic sign recognition. volume E85d, pages 52 – 57, 02 2000.

[21] Agus Mulyanto, Wisnu Jatmiko, Petrus Mursanto, Purwono Prasetyawan, and Rohmat Indra. A new indonesian traffic obstacle dataset and performance evaluation of yolov4 for adas. *Journal of ICT Research and Applications*, 14:286–298, 03 2021.

[22] Y.-Y. Nguwi and A.Z. Kouzani. Automatic road sign recognition using neural networks. In *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, pages 3955–3962, 2006.

[23] Rongqiang Qian, Bailing Zhang, Yong Yue, Zhao Wang, and Frans Coenen. Robust chinese traffic sign detection and recognition with deep convolutional neural network. In *2015 11th International Conference on Natural Computation (ICNC)*, pages 791–796, 2015.

[24] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[25] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[26] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.

[28] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union. June 2019.

[29] Andrzej Ruta, Yongmin Li, and Xiaohui Liu. Robust class similarity measure for traffic sign recognition. *IEEE Transactions on Intelligent Transportation Systems*, 11(4):846–855, 2010.

[30] Banhi Sanyal, Ramesh Kumar Mohapatra, and Ratnakar Dash. Traffic sign recognition: A survey. In *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*, pages 1–6, 2020.

[31] Zhenghao Shi, Bin Guo, Minghua Zhao, Changqing Zhang, et al. Nighttime low illumination image enhancement with single image using bright/dark channel prior. *EURASIP Journal on Image and Video Processing*, 2018(1):1–15, 2018.

[32] Domen Tabernik and Danijel Skočaj. Deep learning for large-scale traffic-sign detection and recognition. *IEEE Transactions on Intelligent Transportation Systems*, 21(4):1427–1440, 2020.

[33] Qing Tang, Ge Cao, and Kang-Hyun Jo. Integrated feature pyramid network with feature aggregation for traffic sign detection. *IEEE Access*, 9:117784–117794, 2021.

[34] Radu Timofte, Karel Zimmermann, and Luc Van Gool. Multi-view traffic sign detection, recognition, and 3d localisation. In *2009 Workshop on Applications of Computer Vision (WACV)*, pages 1–8, 2009.

[35] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh. Cspnet: A new backbone that can enhance learning capability of CNN. *CoRR*, abs/1911.11929, 2019.

[36] Huibai Wang and Hao Yu. Traffic sign detection algorithm based on improved yolov4. In *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 9, pages 1946–1950, 2020.

[37] Fatin Zaklouta and Bogdan Stanciulescu. Real-time traffic-sign recognition using tree classifiers. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1507–1514, 2012.

[38] Jianming Zhang, Zhipeng Xie, Juan Sun, Xin Zou, and Jin Wang. A cascaded r-cnn with multiscale attention and imbalanced samples for traffic sign detection. *IEEE Access*, 8:29742–29754, 2020.

[39] Zhe Zhu, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. Traffic-sign detection and classification in the wild. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.