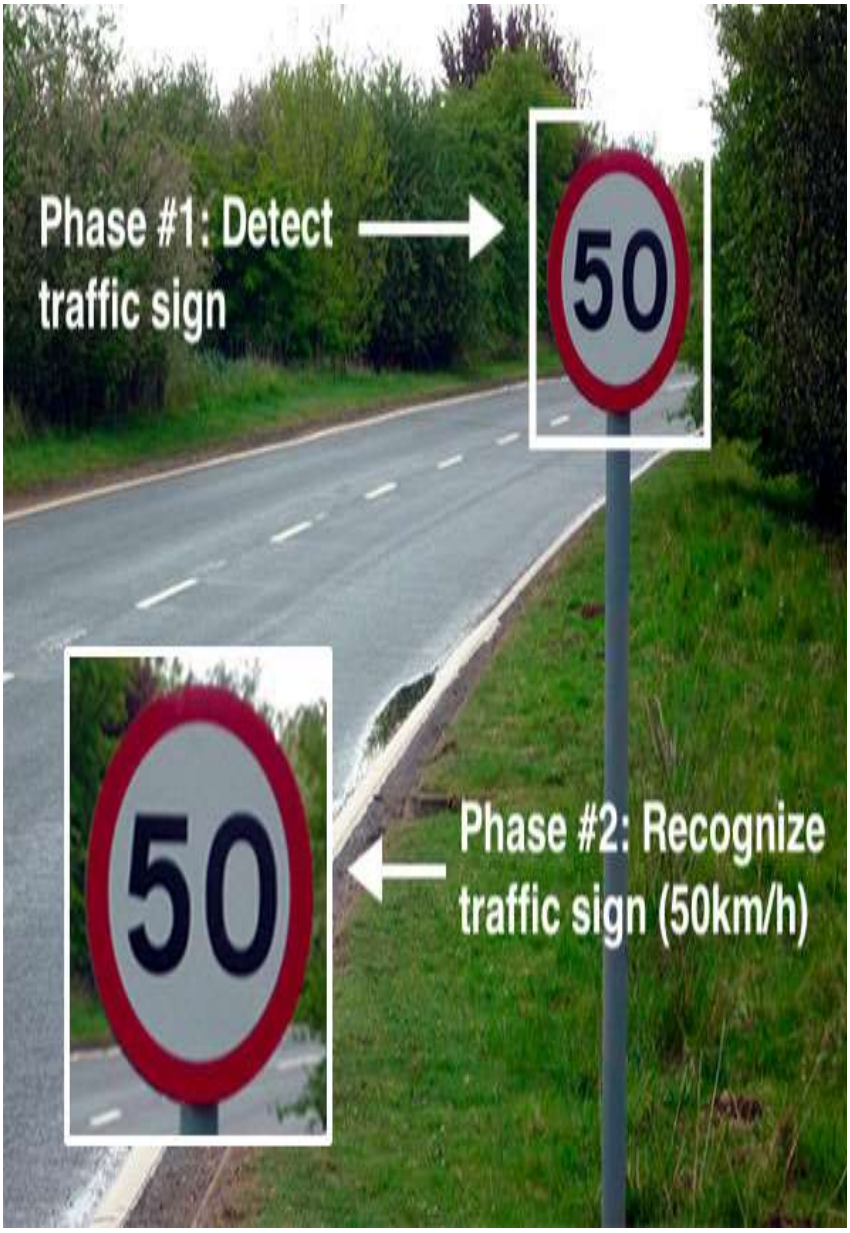
Ain-Shams University
Faculty of Computer Science and Information
Scientific Computing Department



Traffic Signs Detection and Recognition

Ain-Shams University
Faculty of Computer Science and Information
Scientific Computing Department

Traffic sign Detection and Recognition

This documentation submitted as required for the degree of bachelors

In Computer and Information Sciences.

By

Sayed Ashraf Sayed [Scientific Computing]
Ghaly Mansour Ghaly [Scientific Computing]
Tariq Ziad Hassan [Scientific Computing]
Mina Lion Fahmy [Scientific Computing]

Under Supervision of

Professor/ Doc Hala Moushir
Scientific Computing Department
Faculty of Computer Science and Information
Ain-Shams University.

Teacher Assistant / MennaTullah Mamdouh
Scientific Computing Department
Faculty of Computer Science and Information
Ain-Shams University.

Table of Contents

Chapter 6: Conclusion and Future Work

Acknowledges

All praise and thanks to ALLAH, who provided me the ability to complete this work. I hope to accept this work from me.

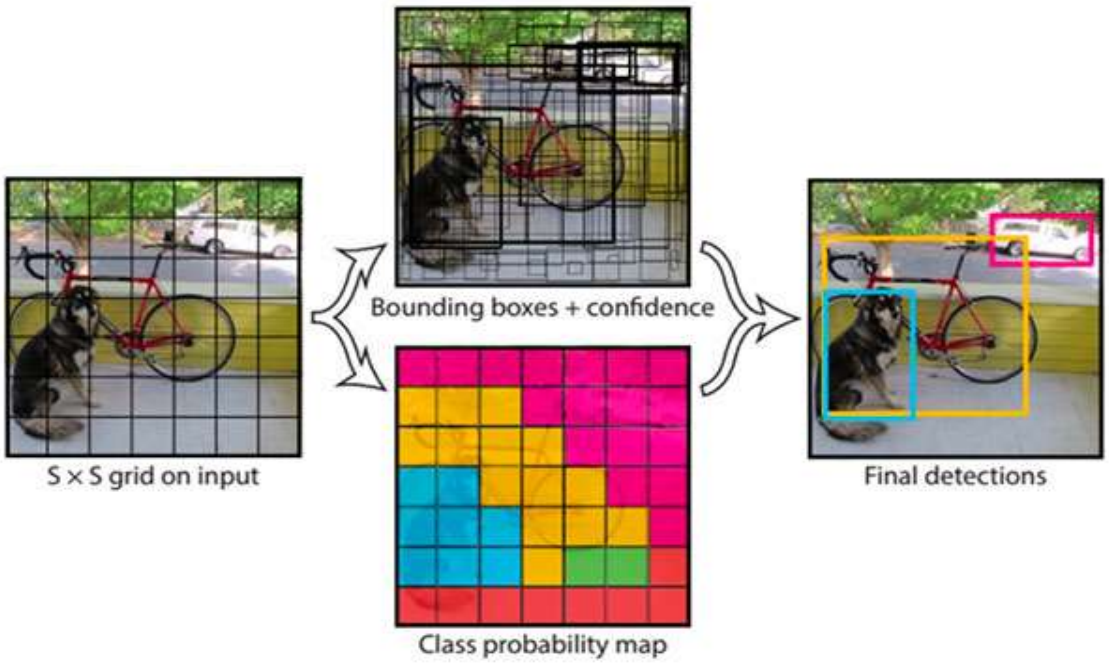
I am grateful of *my parents* and *my family* who are always providing help and support throughout the whole years of study. I hope I can give that back to them.

I also offer my sincerest gratitude to my supervisors, ***Prof. Dr. Hala Moushir*** and ***T.A MennaTullah Mamdouh*** who have supported me throughout my thesis with their patience, knowledge and experience.

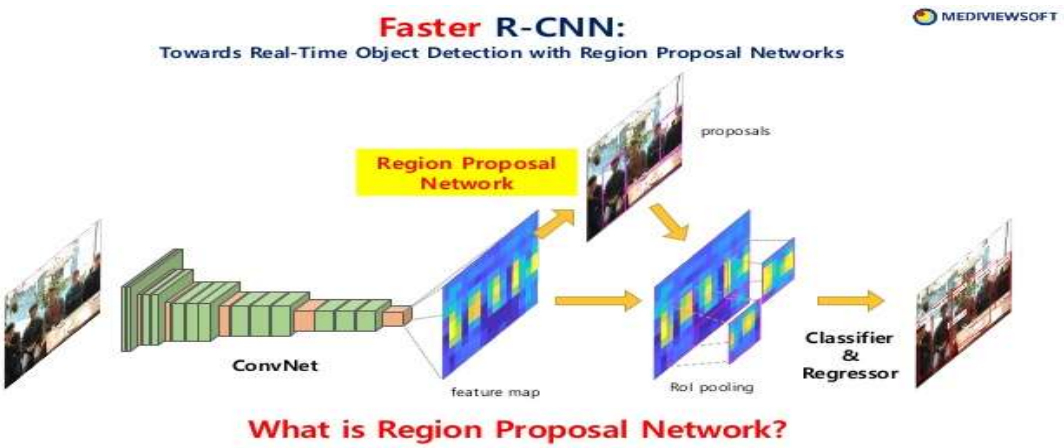
Finally, I would thank my friends and all people who gave me support and encouragement.

Abstract

Increase in the number of vehicles on road necessitates the use of automated systems for driver assistance this system forms important component of self-driving vehicles also, ***Traffic sign Detection and Recognition system*** is such an automated system provides the required contextual awareness for the self-driving or the driver. ***CNN*** based methods like ***Faster R-CNN*** for object detection provide human level accuracy and real time performance and are proven successful. Single stage detection system such as ***YOLO*** and ***SSD***, despite offering state of the real time detection speed. The enhanced ***YOLO*** version ***YOLOv2*** and ***YOLOv3*** have shown promising result with respect to accuracy and speed required for object detection problem. We will use ***YOLOv3*** that use specialized network architecture inspired form feature pyramid network and has several design changes over before tackle the low accuracy and small object detection problems.



Yolo (You Only Look Once) Object Detection



Faster R-CNN (Recurrent Convolution Nural Network) Object Classifier

List of figures

List of tables

List of abbreviations

DITS: The Data Set of Italian Traffic Signs

CNN: Convolutional neural network

SVM: Support vector machine

ROI: region of interest

CLAHE: Contrast Limited Adaptive Histogram Equalization

HSV: Hue, Saturation, Value

YOLO: you only look once

YOLOv2: you only look once version 2

YOLOv3: you only look once version 3

IoU: Intersection over Union

FPN: Feature Pyramid Network

ReLU: Rectified Linear Unit

GTSRB: German Traffic Signs Recognition Benchmark

GTSDB: German Traffic Sign Detection Benchmark

GPS: Global Positioning System

Chapter 1: Introduction

1-Problem Definition:

System for assisting the drivers to avoid accidents are becoming more and more important as the number of vehicles on road is on an exponential increase. Advanced driver assistant systems is being effectively used in automobiles for providing lane keep assistance, forward collision warning, pedestrian warning driver drowsiness detection traffic sign assist system etc. These form essential systems in autonomous cars for contextual awareness and road attribute mapping in order to control the vehicle motion trajectory. ***Traffic sign recognition*** is the core component of traffic assist system for providing timely instructions and warning to the driver regarding traffic restrictions and information. ***In self-driving cars***, the inputs from the system used to make suitable decisions by the car for examples, to reduce speed or prepare for a detour**.** Traffic sign recognition involves traffic sign detection and classification.

2-History:

Introduction:

The United Nations estimates that between 2010 and 2020 the number of road deaths will increase by up to 50%, that is, about 1.9 million people. To reverse this trend, the UN established, in 2011, the 1st ***"Decade of Action for Road Safety"***. Driver Assistance Systems can help reduce the number of accidents by automating tasks such as ***lane departure warning*** systems, traffic sign recognition. The recognition of traffic signs has received increasing attention in recent years; it is even considered as a highly important feature of intelligent vehicles. Traffic signs carry substantial useful information that might be disregarded by drivers due to driving fatigue or searching for an address reasons. These drivers are also likely to pay less attention to traffic signs on driving in threatening weather. Therefore, making enhancement initiatives, like increasing driving safety along with improving automatic detection and ***road sign recognition system***, is becoming indispensable to help decrease road death toll. These enhancements, however beneficial they may seem, meet several external non-technical challenges such as lighting variations, scale and weather conditions changes, occlusions and rotations, which may eventually decrease the traffic sign recognition systems performance. The main issue of the problem in the traffic sign recognition system is not how to detect or recognize, with high recall, a traffic sign in a fixed image. It is rather about how to obtain a high precision in videos big data. To illustrate the problem of false alarms, a traffic sign recognition system installed in a smart phone, with 30 shot frames per s, (***108,000 frames in a 1-h video***), was considered. If we suppose that in every 4min we detect a sign and that—with reference to the speed of a car—every sign spans 2 s, this means that in the course of 1 h we will find a total of 15 signs and that every sign will display in 60 frames (900 frames that contain signs while 107,100 does not). Supposing also that the system has a 1% false positive accuracy rate, this means that

there are 17 detected false alarms in 1min (1071 in 1 h) and 1 true positive and 68 false alarms in 4min by consequent. This may eventually lead to
most users disabling their applications

Traffic signs types: **a** mandatory sign, **b** temporary sign, **c** warning sign, **d** prohibition sign, **e** reservation sign

Figure 1.1

Traffic sign recognition systems consist of three main stages: localization, detection, and classification. In the case of any false alarm in the detection stage, performance will be lower in the classification one; this is due to the fact that the classifier is not usually trained on false alarms. Road signs have many discriminating features on the basis of which they are classified. According to their shapes and colors, these are five main classes: warning signs (red triangle), prohibition signs (red circular), reservation signs (rectangular blue), mandatory signs (circular blue), and temporary signs (yellow triangle). Examples of traffic signs for each of the categories mentioned above are shown in Fig. 1. The aim of this paper is to present an overview of some recent and efficient traffic sign detection and classification methods; some authors like [1–4,15,36] are also preferred to make a study on this domain. In Sect. 2, traffic sign detection methods are presented; they have been divided into three categories: color-based, shape-based, and learning-based methods, including deep learning methods. In Sect. 3, traffic sign classification methods are stressed, firstly we cite learning methods based on hand-crafted features, and then, we mention deep learning methods. Moreover, different publicly available traffic sign detection and classification datasets are also presented to help meet the goal of this paper. Section 4

describes the future research directions which can incorporate with researchers in their future works. Finally, a conclusion is expected.

Detection methods:

Asmentioned above, we can classify detection or localization methods into three fundamental classes: color-based, shapebased, and learning-based methods. According to the nature of the problem and system requirements, we can decide
upon the best method to apply; for example, methods based on color information can used with high-resolution dataset, however, not with grayscale images.

a. Color-based methods:

The dominant color-based segmentation is applied to detect regions of interest. There are specific characteristic colors of traffic signs: red, blue, and yellow. These characteristics, however, indicate sensitivity to various factors, such as the age of signs and the variation of light, which make the segmentation an arduous process. In order to overcome this problem, authors are working on different color spaces among which we mention the following:

i. RGB space:

De La Escalera et al. [5] adopt RGB space by reason of HIS formulas are nonlinear. The authors use the relation between the components as it is presented in the following expression:

$$g(x,y) = k_1 \begin{cases} R_a \le f_r(x,y) \le R_b \\ TG_a \le \frac{f_g(x,y)}{f_r(x,y)} \le TG_b \\ TB_a \le \frac{f_g(x,y)}{f_b(x,y)} \le TB_b \end{cases} \qquad (1)$$

$$g(x,y) = k_2 \quad \text{in other case}$$

fr(x, y), fg(x, y), and fb(x, y) are, respectively, the

functions that provide the red, green, and blue levels of each point of the image. Thres holding is used by several authors as [6-8]. Their approaches, however, highly related to the selected thresholds, which make the comparison of their performances a difficult task. Ruta et al. [7] applied filtering for each pixel: X = [xR, xG, xB] and S = xR + xG + xB)

$$f_R(X) = \max(0, \min(x_R - x_G, x_R - x_B)/S) \quad (2)$$

$$f_B(X) = \max(0, \min(x_B - x_R, x_B - x_G)/S)) \quad (3)$$

$$f_Y(X) = \max(0, \min(x_R - x_B, x_G - x_B)/S)) \quad (4)$$

In this approach, they generate three maps red, blue, and yellow for each RGB image. The dominant color has a high intensity while deteriorated signals have low intensities. King
et al. [9] prefer the R'G'B' space. At first, they normalize the three RGB channels by intensity ***I:***

$$I = \frac{R' + G' + B'}{3} \quad (5)$$

$$r = \frac{R'}{I}, \quad g = \frac{G'}{I}, \quad b = \frac{B'}{I} \quad (6)$$

Then, they construct four new images according to the equation proposed by [10]:

$$R = r - \frac{(g + b)}{2} \quad (7)$$

$$G = g - \frac{(r + b)}{2} \quad (8)$$

$$B = b - \frac{(r + g)}{2} \quad (9)$$

$$Y = \frac{r + g}{2} - \frac{|r - g|}{2} - b \quad (10)$$

The dominant color has a great intensity that

facilitates the extraction of the panels. Thresholding is used to binarize the four images (R, G, B, Y); morphological operations are then applied to remove the unwanted pixels. It is worth highlighting that this approach is capable of detecting up to 93.63%of the panels. King et al.'s approach is adopted in [11]. A filter is proposed to eliminate undesirable pixels with a view to reduce the execution time

ii. HSV space:

Yakimov [12] considered that is not possible to detect traffic signs in real images by applying a simple threshold directly in the RGB space due to lighting variations; this is what urged them to choose the HSV space. They used experimental method to determine the optimal threshold values for red color as it is presented in the following expression:

$$\begin{aligned} &(0.0H < 23) \parallel (350 < H < 360) \\ &(0.85 < S \leq 1) \\ &(0.85 < V \leq 1) \end{aligned} \qquad (11)$$

After segmentation, they used a modified algorithm presented in [13] to denoise segmented images. The advantage of the denoising algorithm is that only noise will be removed and the regions of interest stay unfiltered. Wang et al. [14] also choose HSV space, and they found that the classical thresholding method gives good results in many different lighting conditions except for the cases of color cast or poor lighting condition. They proposed a new thresholding method by using the color information of neighboring pixels. Firstly, the red degree of each point c is calculated to get a new image fR(c) with the following

equation:

$$f_R(c) = \begin{cases} S(c)\frac{\sin(H(c)-300°)}{\sin(60°)} & \text{if } H(C) \in [300°, 360°] \\ S(c)\frac{\sin(60°)-H(c)}{\sin(60°)} & \text{if } H(C) \in [0°, 60°] \\ 0 & \text{others} \end{cases} \quad (12)$$

Secondly, the normalized red degree fN R(x) is calculated as the following:

$$f_{NR}(x) = \frac{(f_R(x) - \mu_R(\omega_x))}{(\sigma_R(\omega_x))} \quad (13)$$

μR(ωx) and σR(ωx) are the mean and the variance of the red degrees of the pixels in the window ωx centered on x. Thirdly, the normalized intensity fN I is calculated with the following equation:

$$f_{NI}(x) = \frac{(f_I(x) - \mu_I(\omega_x))}{(\sigma_I(\omega_x))} \quad (14)$$

fI (x) is the intensity of the pixel x; μI (ωx) and σI (ωx) are the mean and the variance of the intensities of the pixels in the window ωx. Finally, the red bitmap B is given as the following:

$$B(x) = \begin{cases} 1 & \text{if } f_{NR}(x) > \max(THR1, f_{NI} + THR2) \\ 0 & \text{others} \end{cases} \quad (15)$$

Basconand et al. [16] combine thresholding on H and S components with the achromatic decomposition. This method, however simple and fast, is not robust to signal deterioration and illumination changes. Fleyeh et al. [17] use thresholding on H, S, and V

components; this method is resistant to lighting changes, but is costly in computation time. Vitabile et al. [18] on the other side use dynamic aggregation technique of pixels to segment the image.

b. *Shape-based methods*:

The authors, in this approach, do not absolutely consider color segmentation as a discriminative feature due to its sensitivity to various factors such as the distance of the target, weather conditions, time of the day, and reflection of the signs. Conversely, detection of the signs is made from the edges of the image analyzed by structural or comprehensive
approaches. Shape-based methods are generally robust than colorimetric methods by reason they can process images in grayscale and treat their gradients. However, they are costly in computation time, given the fact that the rate of treatment depends largely on the number of detected edges. However, shape-based methods can treat grayscale images; in some countries, such as Japan, there are pairs of different signs in the highway code which, when converted to grayscale, appear exactly the same. To be able to distinguish them, an amount of color information is absolutely needed [7]. On the other hand, some authors adopt the color feature to localize the region of interest and complete with shape methods in order to detect the signs position and recognize its geometric form. Vitabile et al. [18] use the totality of pixels to recognize the geometric form of the sign, where each form of a road sign is represented by a binary image of fixed size (36 × 36 pixels).
After detecting the regions of interest with the dynamic aggregation technique of pixels, they resize them on the same scale of binary image to calculate a measure of similarity with all forms of road signs using the Tanimoto coefficient: The value of this coefficient is normalized, i.e.,

the more it is close to 1, the more the model and region of interest are similar. If X and Y are considered sets of pixels of binary images to be compared, then the Tanimoto coefficient S will be defined as follows:

$$S = \frac{|x \cap y|}{|x \cup y|} \qquad (18)$$

The results obtained show that more than 86% of signs are detected among 620 images of 24 classes. Authors in [25] used template matching to filter out the regions which do not contain traffic signs. A sliding window, having the same size of the template, slides on the image to search its most similar region. The function of similarity used is the mean square error (MSE).

$$MSE = \frac{1}{MN}\sum_{y=1}^{N}[T(x,y) - I(x,y)]^2 \qquad (19)$$

• T (x, y): the intensity value of the pattern image at the position (x, y);
• I (x, y): the intensity of the input image at the position (x, y);
• M and N: the width and height of the image, respectively.

Table 1 Results achieved by Youssef et al. [81]

Dataset	Prohibitive		Danger		Mandatory	
	Accuracy (%)	Time (ms)	Accuracy (%)	Time (ms)	Accuracy (%)	Time (ms)
GTSDB	98.67	231	96.01	234	90.43	243
DITS	97.87	198	98.12	197	89.71	200

Table 1.1

A Hough transform has been used by [26] to detect edges of the panels and select the closed contours which make their approach sensitive to noise and occlusion. This approach is able to detect 97% of the 435 speed limit signs and 94% of 312 danger signs in a time ranging between 20 and 200ms/img according to the number of processed outline. It is worth mentioning that Miura et al. [26] have also applied the transformed Hough for circular signals. Youssef et al. [81] also choose to use HOG descriptor with a 40×40 detection window (block of 10×10 and 2×2 striding). The GTSRB dataset is used for first round training, DITS (The Data Set of Italian Traffic Signs) is used to the second round, and the both are used in the third round. In the aims to reduce both the false positive and time execution, a color segmentation step with improved HSV is used before the detection step. Established results are illustrated in Table 1. In [27], the authors preferred to use Radial Symmetry Transformation to detect speed limit signs. This method, a variant of the circular Hough transform, is particularly made to detect possible occurrences of circular signs. The authors reached a detection rate of 96% for 10% false positives in a 50ms/img for a 320–240 image. Authors in [28] are also using Radial Symmetry Transform to detect other geometric shapes like octagon, square, and triangle. However, they reached a detection rate of 100% for triangle shape, and the major drawback is that it can only find the size and position of forms: It cannot distinguish a Yielding passing panel from an intersection one. In [29], a process of extracting contours with their associated tangents accuracy is made, and they also propose three algorithms: RANSAC type of quadrilateral, ellipse, and triangle detection. The final shape will be chosen based
on compatibility degrees provided by each algorithm. The method presented in [30] is adopted to estimate the center of

the shape using only three points with their tangents. Over 80% of 1400 test images are properly detected, with 5% false positive, in a processing time that ranges between 15 and 20 s
for 1980 1024 image.

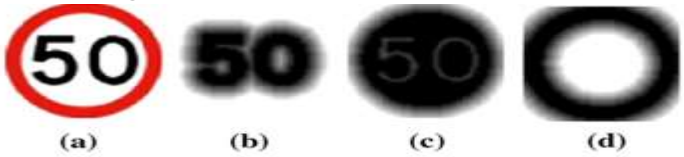


Fig. 2 CDT (*Color* Distance Transform) normalized: **a** original image in discrete *color*, **b** black CDT, **c** white CDT, **d** *red CDT*. Shadows regions mean a small distance, extract of [8] (color figure online)

Figure 1.2

Qin et al. [31] exploit Fourier descriptors (FDs) to describe the contour, the main reason of using FDs is their robustness to the rotation, scaling, and translation, and then, a process of matching FDs is applied. A database containing over 20000 images has been created by recording sequences from over 350 km of Swedish highways and city roads; this database is used to evaluate the proposed approach. The average detection rate achieved is 77.08% with 641 images in total. Several works, like [8], preferred to use the distance transform which converts a binary image to an image where each pixel value represents the distance from the pixel to the nearest pixel feature. The researchers propose a variant of DT called Color Distance Transform (CDT) in which they applied a comparison between the real image and the template one in a representation of the discrete color. To facilitate the comparison, they calculate a distance transform DT for each discrete color. Then, pixels having a discrete color are considered as feature pixels while other pixels are not. The result of this process is illustrated in Fig. 2. Qin et

al.

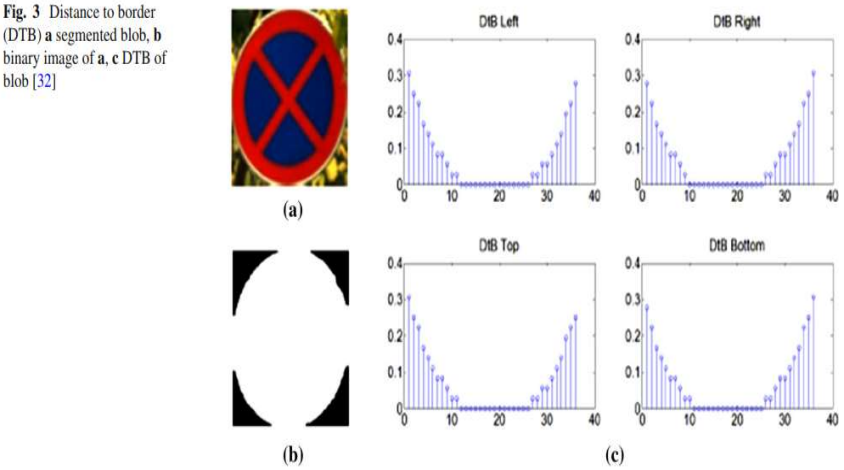


Fig. 3 Distance to border (DTB) **a** segmented blob, **b** binary image of **a**, **c** DTB of blob [32]

Figure 1.3

[32] use the vector of the distance to border (DTB): It is the distance between the outer contour and the bounding box in which they calculate, for each segmented blob, the four DTB vectors (left, right, top, and bottom) as shown in Fig. 3. This distance is robust to translations and rotations. Since the blobs should be zoomed to 36 × 36, the proposed approach is not robust to scale changes. DTB vectors will be then chosen as a feature to classify blobs with SVM. The use of four distance vectors for each segmented blob makes the DTB robust to occlusions, that is why it is also used by [63] as a feature to recognize the geometric form of signs. Another approach is proposed by [11] to recognize the shape of the sign candidate; the idea is to compare the detected patternwith the BoxOutrectangle that encompasses. A score of intersections is calculated between the contour of the pattern and the four lines of the BoxOut; it is shown in Fig. 4. This approach is capable of detecting 95.65% with 2.17 % false alarms. The used dataset consists of 48 images with 360×270 pixels containing three different traffic signs. The disadvantage of this approach is its weakness to

occlusions and noises.

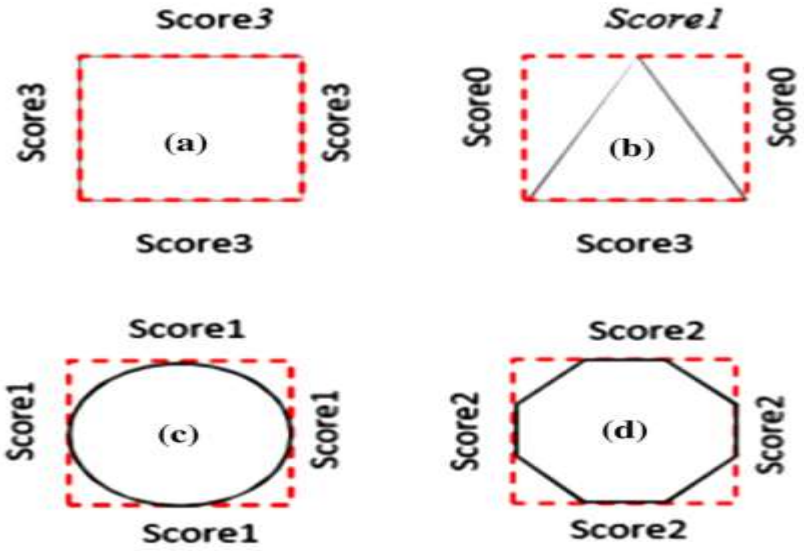


Fig. 4 The proposed detection approach of [11] **a** rectangle, **b** triangle, **c** circle, **d** octagon

Figure 1.4

Table 2 summarizes shape-based methods; it is clear from the table that the previous detection methods do not use the same database, which prevents us from comparing them to one another. Authors in [29] use a big dataset with large images, reached an acceptable detection rate that is still very far from the real-time application. The method used by [11], however having real time, its dataset is small, which keeps us from considering it a highly efficient method.

Paper	Detection rate (%)	False alarms	Used dataset	Time
Barnes et al. [27]	100	10	Speed limit signs with 320 × 240 pixels	50 ms/img
Behloul et al. [11]	95.65	2.17%	48 images with 360 × 270 pixels1	Not exceed 1 ms/image
Berkaya et al. [76]	93.78 for prohibitive signs 75.51 for mandatory signs	0.99% for prohibitive signs 2.04% for mandatory signs	GTSDB	36 ms/image
Garcia-Garrido et al. [26]	97 of speed limit	–	435 speed limit signs	20 until 200 ms/img
	94 of panels danger	–	312 danger signs	
Soheilian et al. [29]	80	5	1400 images with 1980 × 1024 pixels	15 until 20 s/img
Vitabile et al. [18]	86	–	620 img with 24 classes	–
Youssef et al. [81]	Prohibitive: 98.67		GTSDB	231ms
	Danger: 96.01			234 ms
	Mandatory: 90.43			243 ms
	Prohibitive: 97.87		DITS	198 ms
	Danger: 98.12			197 ms
	Mandatory: 89.71			200 ms

Table 1.2

c. Learning-based methods:

The previous methods share a common weakness in several factors such as lighting changes, occlusions, scale change, rotation, and translations. However, these problems could also be treated using machine learning, but it requires a large database of annotated data. Increasing complexity detectors cascade is used by Viola and Jones [33], each detector is a set of classifiers based on Haar wavelet, and these classifiers use a learning algorithm AdaBoost. Authors in [34] used the Viola–Jones detector to detect triangular traffic sign. The

detector was trained using about 1000 images of relatively poor quality. The obtained detector achieved a very high true positive rate (ranging from 90 to 96%) depending on the training set and the configuration of the detector. In their experiments [34,35], they observed two main weaknesses of the Viola–Jones detector:

• the requirement of a large number of training images;
• false positive rates.

Nevertheless, their research also indicates that the Viola–Jones detector is robust to noise and low-quality training data Bario et al. [37] proposed an attentional cascade which is composed of a set of classifiers where each entry of the cascade is the region of interest detected by the previous classifier, and Adaboost algorithm was used to learn the classifiers. The researchers also proposed a classification strategy Forest-ECOC (Error Correcting Output Codes) to overcome the multiclass problem, and the idea is to integrate several trees in the Framework ECOC. The authors obtained the following results:
• Prohibition circle: 70% of detections with 3,65% false positives per image;
• Obligation circle: 60% of detections with 0,95% false positives per image;
• Danger triangle: 65% of detections with 2,25% false positives per image;
• Triangle Right of way: 75% of detections with 2,8% false positives per image.

The rate of treatment is not given because the algorithm is operated offline. The case of rectangular panels is not addressed, and the color information is not used. Priscariu et al. [38] used Adaboost classifier based on viola and Jones detector, followed by an SVM operating on normalized RGB

channels. The system is robust to motion blur through 3D region-based tracking. Chen et al. [85] explore both exploring Adaboost and support vector regression (SVR) together to detect traffic signs. The proposed approach is evaluated over three datasets (GTSD, BTSD, STSD) using an Intel Core-i7 4770 with 8G RAM. This approach is not real time, the detection time varied from 0.05 to 0.5 s, and the training time is 16min. The recall obtained on STSD is 80.85% with a precision of 94.52%, and Table 3 shows the results achieved on GTSD and BTSD. Researchers in [19] used genetic algorithms in the detection step. The authors applied a parallel search in different directions following by an optimization process that mimics natural evolution and selection. However, their approach is robust to scale changes, rotation, weather conditions, and partial occlusion, and it is not a real-time application. The neural networks are used by [39] to recognize the shape of the panels, but this process is not linear with time 2 s/img.

Table 3 Results achieved on GTSD and BTSD by Chen et al. [85]

Categories	GTSD (%)	BTSD (%)
Mandatory	99.87	97.78
Danger	100	99.88
Prohibitory	100	93.45

Table 1.3

Table 4 Detection rate of algorithms cited in Houben et al. [43]

Algorithm	Prohibition (%)	Warning (%)	Mandatory (%)
HOG+LDA	91.3	90.7	69.2
Hough-like	55.3	65.1	34.7
Viola-Jones	98.8	74.6	67.3

Table 1.4

Zaklouta et al.[40] use the histogram of oriented gradients, proposed by Dalal and Triggs [41] for pedestrian detection due to its scale-invariance, local contrast normalization, coarse spatial binning, and weighted gradient orientations. The HOG descriptors are computed and used as a feature to train a linear SVM classifier. To improve the precision of the SVM detector, they use a morphological operator (blackhat) to filter the detected candidates. The blackhat transform is defined as the difference between the closing and the input image. A large part of the image is eliminated using this lter, and the number of false alarms is reduced. HOG transformer is also used by Wang et al. [42], and they use LDA and SVM as a classifier. The proposed approach achieves high recall and precision ratios in GtSDB dataset, it is robust to bad lighting condition, partial occlusion, low quality, and small projective deformation, but this method is not real time. In [43], the authors use the German Traffic Sign Detection Benchmark presented as a competition at IJCNN 2013 (International Joint Conference on Neural Networks) to evaluate some of the most popular detection approaches such as the Viola–Jones detector based on Haar features and a linear classifier relying on HOG descriptors, and they also evaluate a recent algorithm exploiting shape and color in a model-based Hough-like voting scheme presented in [44]. The detection rate of the three algorithms is presented in Table 4. Salti et al. [84] propose an approach based on interest regions extraction rather than sliding window detection. The authors test their approach on ground-truth dataset which contains 6580 images. The dataset is created by using 5 cameras to correctly geo-referenced the signs; however, using 5 cameras is not practical for a real-time application because it increases the execution time (Instead of processing 1 image, 5 images will be processed). They achieved 78.21% for prohibitory signs, 82.13% for danger

signs and 72.78% for mandatory signs. Integral Channel Features detector built on HOG features and boosted decision trees is used by [45], on BTSD dataset they obtained 97.96% for mandatory signs, 97.40% for warning signs and 94.44% for prohibitory signs, on GTSDB dataset they have 96.98% for mandatory signs, 100% for warning signs and 100% for prohibitory signs.

To address the multiclass traffic sign detection (TSD) problem, [48] presented traffic sign localization framework that is capable to detect multiclass traffic sign rapidly in high-resolution image. They proposed three new ideas: first, multi-block normalized local binary pattern (MN-LBP) and tilted MN-LBP (TMN-LBP) are used as discriminant features to express multiclass traffic signs effectively. Second, a tree structure called Split-Flow Cascade is designed. It uses the common features of multiple classes to build a coarse-to-fine pyramid structure named SFC tree. Third, to
design an efficient SFC-tree a Common-Finder AdaBoost (CF AdaBoost) is developed to find common features among different training sets. All these new contributions make the system work in real time with high recall, and they have good results on GTSDB dataset: 100% for prohibitory signs, 99.2% for warning signs, 98.57% for mandatory signs, and 97.24% for other signs.

In [47], a good run time has been achieved (6–8 fps on video sequences). The solution presents a novel approach, called Categories-First-Assigned Tree (CFA-Tree) where they integrate the detection and the classification phase in one module, this novel system has high accuracy about 93.5%, and however, this search tree can only detect three categories and has low efficiency in handling high-resolution images [48]. Due to the success of CNN in traffic sign classification, the authors in [77] propose a lightweight and optimized ConvNet with sliding window to detect traffic signs in high-

resolution images. The accuracy rate detection achieved is 99.89% on the German Traffic Sign Detection Benchmark dataset. Time execution on GPU (GeForce GTX 980) is 26.506ms which is equal to processing 3772 frames per s. Obtained results make this approach a real-time application. Wu et al. [46] use convolutional neural networks CNN to localize and recognize traffic sign, firstly they use support vector machine to transform the original image from RGB to grayscale to avoid falling into the problem of sensitivity to color difference due to various lighting conditions, secondly they use the fixed layer in the CNN to localize region of interest which are similar to traffic sign, and the learnable layers are used to extract discriminant features for classification. They use GTSDB as a dataset and they obtained 99.73% in warning signs and 97.62% in mandatory signs, but it is too far from a real-time application. Cascaded convolutional neural networks (CNNs) are recently used by [83] to reduce false positive regions detected
using the local binary pattern (LBP) feature detector combined with the AdaBoost classifier. Results achieved on GTSDB using an Intel Core 2 Duo 2.2GHz are illustrated in Table 5. In Table 7, we resume detection methods based on machine learning. Best accuracies (>99%) are achieved by [84], [45] and [42], and however, can really these methods be a commercial application and have a high rate and precision in ground truth with other complex conditions? However, these methods give best results, and they are not real-time
application. On the other hand, [82] achieve good results with real-time execution (35 ms). Ordinarily, most learningbased detection methods achieve a detection rate higher than 95%, so it is time to create new datasets more

complicated because current dataset is saturated.

Table 5 Results achieved by Zang et al. [83]

Categories	Rate (%)	Time (ms)	False positive
PROHIBITORY	99.45	35	1
MANDATORY	96.50	34	0
DANGER	98.33	35	0

Table 1.5

Table 6 Publicly available traffic sign detection datasets

Dataset	*Images*	Properties
GTSDB [43]	900	Image's size: 1360 × 800 pixels Country: Germany
BTSD [49]	10,000	Image's size: 1628 × 1236 pixels Contain 4 videos With annotations Country: Belgium
LISA [50]	7855	Contain videos With annotations Image's size: 640 × 480 to 1024x522 Country: USA
MASTIF [51]	10,000	Image's size: 720× pixels Composed of 3 datasets With annotations Country: Croatia
STSD [31]	20,000	Image's size: 1280 × 960 pixels With annotations Country: Sweden
DITS [81]	1887	With annotation Country: Italy

Table 1.6

Table 7 Learning-based detection methods

Methods	Dataset used	Prohibition signs	Warning signs	Mandatory signs	All signs	Time
Aghdam et al. [77]	GTSDB		–	–	99.89%	26.506 ms
Bar et al. [37]	–	70% with 3,65% false positives	65% with 2,25% FP	60% with 0,95% FP	–	–
Brkic et al. [34]	1000 images	–	90% to 96%	–	–	–
Chen et al. [85]	STSD				80.85%	
	GTSDB	100%	100%	99.87%		0.05 to 0.5 s
	BTSD	97.78%	99.88%	93.45%		
Houben et al. [43]						
HOG+LDA	GTSDB	91.3%	90.7%	69.2%	–	–
Hough-like	GTSDB	55.3%	65.1%	34.7%	–	–
Viola–Jones	GTSDB	98.8%	74.6%	67.3%	–	–
Liu et al. [47]	GTSDB	–	–	–	93.5%	Real time
Liu [48]	GTSDB	100%	92.2%	98.57%		192 ms
Mathias et al. [45]	BTSD	94.44%	97.40%	97.96%	–	–
	GTSDB	100%	100%	96.98%	–	–
Salti et al. [84]	Ground-truth	78.21%	82.13%	72.78%	–	–
Wang et al. [42]	GTSDB	99.88%	99.85%	99.85%	–	1.12 to 1.48 s
Zaklouta et al. [40]	820 images contains 11 classes	–	–	–	95%	247 ms
Zang et al. [83]	GTSDB	99.45%	98.33%	96.50%	–	35 ms

Table 1.7

Classification methods:

In this section, we highlight some recent and efficient methods in traffic sign classification. Firstly, we describe some methods use hand-crafted features such as HOG, LBP, SIFT, and BRISK; secondly, deep learning methods surpassed the human performance are cited (Table 7)

a. Learning methods based on hand-crafted features:

Fatin Zaklouta et al. [52] used different size histogram of oriented gradients (HOG) descriptors and Distance Transforms to evaluate the performance of K-d trees and random forests, and the random forests are more robust to variations in the background than the K–d trees. The rate classification achieved for random forest is 97.2% with HOG descriptors and 81.8% with Distance Transform, K–d trees improve 92.9 and 67%, respectively. Ellahyani et al. [86] calculate the histogram of oriented gradients (HOG) features in the HSI color space; then,

it combined with the local self-similarity (LSS) features. The authors prefer to use random forest as a classifier. The recognition rate achieved is 97.43% on the GTSDB and 94.21% for the entire system in 8–10 frame/s. Authors in [53] compared traffic sign recognition performance of human and machine learning methods; they also show results of a linear classifier trained by linear discriminant analysis (LDA). The performance of LDA was dependent on the feature representation. The authors achieve best results with HOG2 representation by an accuracy of 95.68, 93.18% for HOG1 and 92.34% for HOG3.

Table 8 Results achieved by Yakimov [61]

Algorithm	Accuracy (%)	time (FPS)
Sliding window + SVM [43]	100	1
GHT with preprocessing [61]	97.3	43
GHT without preprocessing [61]	89.3	25
Viola–Jones [43]	90.81	15
HOG [43]	70.33	20

Table 1.8

The analysis of training data carried out by [62] demonstrates that there is an imbalance in the distribution of samples in the traffic sign classes. The biggest class can contain morethan 1000 images while the smallest class can contain only several images. This imbalance can negatively impact on the classification performance; to overcome this problem, the authors proposed a hierarchical classificationmethodfor traffic sign recognition, and the classification tree is composed of two layers. In the first layer, the Adaboost classifier combined with Aggregate Channel Features (ACF) is used to classified sings into three categories according to their geometric shape. The ACF is used for feature representation where 10 channels are used (three color channels of RGB color space, the gradient magnitudes, the six oriented gradient maps: horizontal, vertical, 30, 60, 120, and 150), and then, these features are used for

training Adaboost classifier. In the second layer, a traffic sign is identified by a random forest classifier which is trained on three features: histogram of oriented gradients (HOG), local binary pattern (LBP), and HSV; they achieved as accuracy 95.97% with GTSRB dataset and 97.94% with STSD dataset. Yakimov et al. [61] achieved a real-time traffic sign recognition using multithreaded programming technology CUDA on a mobile GPU Nvidia Tegra K1, which contains 192 graphics cores and 4 CPU cores of ARM architecture. The authors proposed a modified generalized Hough transform (GHT) algorithm to classified traffic signs, and they show that the algorithm achieved a good compromise between execution time and accuracy with preprocessed images comparing to result obtained in [43] as it is illustrated in Table 8. The German traffic sign dataset was used, but only 9987 among 50,000 images were taken into account. LBP and HOG features are used by Li et al. [74] to recognize traffic signs. The accuracy rate achieved is 95.16% for HOG and 95.38% for LBP with GTSRB dataset. Authors in [76] use three categories of feature descriptor HOG, LBP, and Gabor filter as input features to SVM, results obtained are shown in Table 9.

Table 9 Results achieved by Li et al. [74]

approach	Rate (%)	Executing time (ms)
LBP	93.36	1.21
Gabor	93.90	2.31
HOG	94.56	0.02
HOG+LBP	95.24	1.24
HOG + Gabor	97.00	2.40
Gabor + LBP	95.17	3.49
Gabor + LBP + HOG	97.04	3.60

Table 1.9

He and Dai [72] propose a new variant of local binary pattern (LBP) named multiscale center symmetric local binary pattern (MS-CSLBP) used as a local feature and the low frequency coefficients of discrete wavelet transform (DWT) as a global

feature to classified traffic signs. The great difference between LBP and CSLBP is that the latter replace the center pixel value with the pixel value that is symmetric to the center pixel which reduce the dimension of feature vectors to 2N/2 rather than 2N in LBP as it is illustrated in the following equation:

$$CSLBP_R^N = \sum_{i=0}^{\frac{N}{2}-1} S(g_i - g_{i+p/2})2^i \qquad (20)$$

$$S(x) = \begin{cases} 1 & x > \tau \\ 0 & x < \tau \end{cases}$$

A threshold τ is used to reduce the influence of the noise. CSLBP can reduce the dimension of the feature vector and computes very simple compared to HOG and SIFT; however, CSLBP feature of single scale is not enough a discriminate feature to represent traffic signs. To solve the problem, the authors propose to calculate CSLBP in multiscale (radius = 1, 2 and 3). The MS-CSLBP is used then as input feature to classified traffic signs with SVM; the results achieved by [72] with GTSRB dataset are compared with two algorithms. SIFT feature matching algorithm is used by [66], and the authors test their approach on eight videos captured in India. The range of the rate achieved is between 75 to 100% with a false positive does not exceed 2%. Hua et al. [67] also used SIFT combined with the bag of visual to construct codebook then classifying with SVM. The dataset used for evaluation consists of 130 images (50 × 50 pixels), and they obtained an accuracy rate 93% with time execution which consists of

0.098ms per image.

Table 10 Results achieved by He et al. [72] compared to results of Tang et al. [73]

Method	Rate (%)	Execution time (ms)
HOG [73]	95.92	134
DCT+LBP [73]	92.50	350
DWT+MSCSLBP [72]	97.67	121

Table 1.10

To exploit the intrinsic structure of the pre-learned visual codebook, a new feature approach using group sparse coding is proposed by [71]. The rate of recognition achieved by the proposed approach is 97.83% with GTSRB dataset. [63] use Speeded Up Robust Feature Descriptor (SURF) with Artificial Neural Network (ANN) classifier to recognize traffic signs. They create a new dataset containing 200 images which have captured from highway roads of Bangladesh in different weather and illumination conditions. The true positiverate achieved is 97% with 3%false positiverate; however, they have good results, the time execution has not mentioned, and their dataset does not contain all traffic sign types. [64] combine SURF with K-nearest neighbor (K-NN) search method, and they propose a novel feature selection strategy. Not useful interest points are eliminated by applying a threshold of the determinant of Hessian matrix, only interest points with larger determinant are considered. In order to find good matches, they propose a novel feature selection strategy by calculating the first and the second minimum distances. Then, a good match is given as the following:

$$\frac{d_1}{d_2} < t_1 \quad and \quad d_1 < t_2 \tag{21}$$

d1, d2: the first and the second minimum distances. t1, t2: relative and absolute threshold, respectively. To evaluate their approach, they create a new dataset including more than 1200 images. They have as total false recognition 12.81% and 0.99% of false classification rate. The proposed approach is not robust to blur, and they obtained a total false recognition 4% after eliminating blur images from the dataset. Chen et al. [69] use SURF as a feature to classify traffic signs; firstly, they divide template signs into eight categories based on the color and the trained Adaboost classifiers to reduce time processing. Then approximate nearest neighbor (ANN) algorithm is used for matching step. The recognition accuracy achieved is 92.7% with 200 images containing 281 traffic signs. SURF descriptor is also chosen by [11] due to the efficient of SURF in execution time and robustness to changes in lighting comparing to SIFT and PCA-SIFT. The authors create for each class of signs a template model to eliminate the interest points detected in the background and keep only the points inside the sign. The recognition rate obtained is 97.72% with 48 images. Hoferlin et al. [70] present an architecture for recognition of circular traffic signs; it consists of two multilayer perceptrons (MLP). The first layer uses SIFT as input feature, and the second layer uses SURF. System performance is evaluated in 30min sequence which contains 133 traffic signs, and they achieved a rate of 96.4%.

Table 11 Results achieved by Malik et al. [65]

Method	Recognition rate (%)		Matching time(s)
	Scenario 1 (%)	Scenario (%) 2	
SIFT	100	93.75	9.03
SURF	93.75	81.25	6.02
BRISK	93.75	87.5%	4.3

Table 1.11

A comparative analysis of three feature matching techniques SIFT, SURF and Binary Robust Invariant Scalable Key (BRISK) points is presented by [65]. The authors create a new dataset containing 172 images classified to 32 categories; after evaluation, they observed that SIFT is outperformed SURF and BRISK. In execution time, however, BRISK is almost twice as fast as SIFT, but it is not in real time. The authors evaluate the system in two different scenarios: in the first scenario, signs are manually segmented, while in the second one, signs are the result of segmentation step. Table 11 shows results obtained by the authors which demonstrate that classification performance depends on results of segmentation step. Recently, SURF was chosen by [68] because they consider that SURF is a method that achieved the best results in a reasonable time. The accuracy rate achieved is 94.28% with 179 images of the created dataset, and execution time obtained is 0.04 ms for a single SURF keypoint and 5 ms per image. In this approach, objects detected in a similar area of the scene are merged to one object, so that two superimposed signs can be considered as a single sign which can influence the matching performances.
Learning methods based on hand-crafted features are less accurate than ConvNets; however, both methods are not scalable and they cannot classify signs of other region or novel dataset. Due to the scalability of visual attributes, they are used by [89], combined with Bayesian networks which can estimate the probable class of a novel input using the observations. Rate

classification achieved for each class of GTSRB is 97.01% for the speed limit, 97.09% for mandatory, 96.31% for danger and other class so in average they achieved 98.04%. we summarize classification methods based on handcrafted features in Table 12. However, all methods achieve a good rate classification, and there is not a method with high rate classification and less false positive process in real time it still an open field of

Table 12 Traffic sign classification methods based on hand-crafted features

Authors	Dataset used	Feature	Rate (%)
Abedin et al. [63]	200 images	SURF+ANN	97
Aghdam et al. [89]	GTSRB	Visual attributes with Bayesian network	98.04
Ali et al. [11]	48 images	SURF	97.72
Berkaya et al. [76]	GTSRB	GABOR	93.90
		LBP	93.36
		HOG	94.56
		HOG+LBP	95.24
		HOG+GABOR	97.00
		Gabor+LBP	95.17
		HOG+LBP+GABOR	97.04
Chen et al. [69]	200 images	SURF+ANN	92.7
Ellahyani et al. [86]	GTSRB	HOG in HSI+LSS with random forest	97.43
Han et al. [64]	1200 images	SURF+KNN	96
He and Dai [72]	GTSRB	MS-CSLBP+DWT with SVM	97.67
Hoferlin et al. [70]	30 min sequence	SIFT+SURF with MLP	96.4
Houben et al. [43]	GTSRB	Sliding window with SVM	100
		Viola–Jones	90.81
		HOG	70.33
Hua et al. [67]	130 images	SIFT+bag of visual	93
Lasota et al. [68]	179 images	SURF	94.28
Li et al. [74]	GTSRB	HOG	95.16
		LBP	95.38
Liu et al. [71]	GTSRB	Group sparse coding	97.83
Malik et al. [65]	172 images	SIFT	93.75
		SURF	81.25
		BRISK	87.5
Qu et al. [62]	GTSRB	ACF with Adaboost	95.97
	STSD		97.94
Sathish et al. [66]	Videos captured in India	SIFT	75 to 100
Stallkamp et al. [53]	GTSRB	LDA+HOG1	93.18
Tang et al. [73]	GTSRB	HOG	95.92
Yakimov [61]	GTSRB	GHT with preprocessing	97.3
		GHT without preprocessing	89.3

Table 1.12

b. Deep learning methods:

The traditional hand-crafted features have a limited representation power and they are strongly related to expert knowledge; consequently, they cannot be discriminative with a very large dataset. To overcome this problem and push the recognition performance, deep features are necessary. Pierre

Sermanet and Yann Le Cun [54] use Convolutional Networks (ConvNets) to learn invariant features of traffic sign in a supervised way using 32x32 color input images of GTSRB dataset, and they reached an accuracy of 98.97%, which is above then the human performance(98.81%).Moreover, they increased their networks capacity and depth by ignoring color information and they established a new record of 99.17%.

Table 13 Accuracies and time processing obtained by Aghdam et al. [58] compared with results of [60]

Method	Accuracy (%)	CPU time execution (ms)	GPU time execution (ms)
1 ConvNet [58]	–	12.96	1.06
1 compact ConvNet [58]	–	12.47	1.03
1 ConvNet [60]	–	14.47	1.45
5 ConvNet [58]	99.23	64.8	5.30
2 Compact ConvNet [58]	99.61	24.94	2.06
20 ConvNet [60]	99.65	289.4	29.0

Table 1.13

The authors obtained the best result without color information, so they suspected that normalized color channels may be more informative than raw color, and this method is still far from a real-time application. Fully connected layers in convolutional neural network (CNN) are trained through back-propagation, and the sensitivity of back-propagation and the over training of fully connected layers can make the generalization performance of CNN sub-optimal. Ordinarily, authors use CNN as a feature extractor and a classifier, it is true, and they are getting impressive results, however, with a vast and complex network on a huge dataset. On the other hand, the authors in [57] propose a new approach where CNN works as a deep feature extractor, which means that only the first eight layers are retained and eliminating the fully connected layers. Extreme learning machine (ELM) is used then as a classifier due to its performance generalization, and it receives the input from CNN. The proposed method takes 5–6 h in training without GPU implementation and achieves a recognition rate of 99.40%

without any data augmentation and preprocessing like in [56], but this method is not robust to motion blur. Qian et al. [87] also use CNN as a feature extractor and multilayer perceptron (MLP) as a classifier. Comparing with the classical ConvNet, in themax pooling layer the authors do not use the max values, conversely, their position. The max pooling positions (MPPs) consist to encoded each max value position to a binary value of 4 bits, then concatenating all max positions values to obtain the MPPs feature. The accuracy achieved using MPPs is boosted to 98.86% on GTSRB.
Xie et al. [88] observe that 80% of misclassified signs have the same color, shape, and pictogram. To overcome this problem, they propose a two-stage cascaded CNN. The first stage CNN is trained on the class label, while the second stage is trained on super-classes separately according to the shape and pictogram. The accuracy of the proposed method is 97.94% on GTSRB, and they decrease the misclassified number with the cascaded CNN from 430 to 202. The time execution is not mentioned.
A new ConvNet architecture proposed by [58] can reduce the number of parameters 27, 22 and 3% compared with ConvNets used by [54,59,60], respectively; then they propose a compact ConvNet which reduces the number of parameters 52% compared to the ConvNet proposed. To improve the accuracy of classification, they also proposed a new method for creating an optimal ConvNets by selecting an optimal number of ConvNets with the highest possible accuracy and with less number of arithmetic operations 88 and 73% compared with [59,60], respectively. The accuracy achieved by their proposed method on GTSRB dataset is 99.23% with 2 ConvNets (compact ConvNet) and 99.61% with only 5 ConvNets which greatly reduces the execution time as it is illustrated in Table 13. On the other hand, [54] have as accuracy 99.46% with an ensemble of 25 ConvNets, and [60] achieve an accuracy 99.65% using 20 ConvNets. To test the scalability and the cross-dataset performance of the new ConvNet proposed by [58], they use the

ConvNet already trained on GTSRB to identify the traffic signs in the BTSC dataset, and the accuracy obtained is 92.12%. A simple architecture of deep neural network is used by [80] to recognize circular traffic signs, and they achieved a recognition rate of 97.5% on GTSRB. Two other architectures of CNN are proposed by [81], and single-scale architecture consists of two stages of convolutional layers and two local fully connected layers followed by a softmax classifier. In multiscale architecture, the output of the first convolutional layer is considered as input for both the second convolutional layer and the first fully connected one. Two architectures are evaluated on GTSRB dataset and their novel dataset DITS (Data Set of Italian Traffic Signs). The accuracy rates achieved on GTSRB are 97.2% for singlescale architecture and 98.2% for the multistate one, and for DITS dataset the accuracy rate achieved is 93.1% for single scale and 95.0% for multistate. Dan Cirean et al. [55] used a GPU implementation of a convolutional neural network and they applied preprocessing on input images by resizing all images to 48x48 pixels and they test three types of normalization to overcome high contrast variation and they obtain a recognition rate of 99.15% by using a committee of multilayer perceptrons (MLPs) trained with HOG feature descriptors and CNNs trained on raw pixel intensities to boost recognition performance. After they won the final phase of the German traffic sign recognition benchmark by a recognition rate 99.46% in [56], a multi-column deep neural networks (MCDNN) by averaging the output activations of several DNN columns. Aghdam et al.
[78] propose a new architecture of CNN which sets a new record of classification accuracy 99.51%. The proposed architecture reduces 85% of the number of the parameters and 88% of multiplications comparing to the winner network reported in the German Traffic Sign Benchmark competition [56]. This reduction is achieved by using Leaky Rectified Linear Units (ReLU) [79] as an activation function that use only

one comparison and one multiplication to compute the output A new record of traffic sign classification is achieved again by Aghdam et al. [77], and they propose a new variant of their previous CNN proposed in [78]. The authors replace color images with that grayscale, and they remove the linear transformation layer. And to increase the flexibility a fully connected layer is added to the network and they reduce the size of the first and the middle kernels also the input images are resized to 44 × 44 pixels to reduce time processing. The new best accuracy achieved is 99.55% with single CNN and 99.70% with an ensemble of 3 CNN. The new CNN is real time with time processing of 0.7 ms per image.
In Table 14, we summarize deep learning methods

Table 14 Deep learning classification methods

Authors	Dataset used	Method	Rate (%)
Aghdam et al. [58]	GTSRB	New ConvNet	99.23
		Compact ConvNet	99.61
	BTSC	ConvNet trained on GTSRB	92.12
Aghdam et al. [78]	GTSRB	New CNN	99.51
Aghdam et al. [77]	GTSRB	Single CNN	99.55
		3 CNN	99.70
Cirean et al. [56]	GTSRB	MCDNN	99.46
Cirean et al. [55]	GTSRB	MLPs + HOG + CNNs	99.15
Eickeler [80]	GTSRB	New DNN	97.5
Jin [60]	GTSRB	20 ConvNet	99.65
Qian et al. [87]	GTSRB	CNN+MLP	98.86
Qu et al. [62]	GTSRB	ACF with Adaboost	95.97
	STSD		97.94
Sermanet et al. [54]	GTSRB	Convnets	98.97
		Human performance	98.8
		Convnets without color	99.17
Xie et al. [88]	GTSRB	Cascaded CNN	97.94
Youssef [81]	GTSRB	Single-scale CNN	97.2
Zeng et al. [57]	GTSRB	CNN+ELM	99.40

Table 1.14

Table 15 Available classification datasets

Dataset	Classes	Images	Properties
GTSRB [53]	43	50000	Image's size: 15 × 15 to 222 × 193 pixels Country: Germany
BTSC [49]	62	6000	Image's size: NA Country: Belgium
Revised MASTIF [51]	30	6000	Image's size: 64 × 64 pixels Country: Croatia
DITS [81]	58	9254	Country: Italy

Table 1.15

1. Traffic Signs Recognition and Classification based on Deep Feature Learning

Convolutional Neural Network:

Our basic model follows the classic LeNet-5 network, which consists of three convolutional layers ($C1$, $C3$ and $C5$) and two sub-sampling layers ($S2$ and $S4$). The input of each layer represents a small group of local units from the upper layer. Each convolution layer contains multiple convolution kernels. These convolution kernels are able to scan the image features via different expressions, based on this, we can acquire various feature maps in different locations. The subsampling layer, following the convolution layer, is mainly used to reduce the resolution of the feature map, to extract the existing image features and to determine the features' relative location.

CNN Training:

The training of the layer-concatenated CNN includes 4 main parts, i.e., forward propagation, error calculation, back propagation and weights adjustment. The forward propagation represents a progress of information delivery from the input layer to the output layer. Since initialization of the weighting parameters is random, the results obtained by forward propagation tend to be deviated. To modify this deviation, error estimation and parameters adjustment are included between each forward and backward propagation.
More specifically, it runs with the following procedures:

• Extract a sample from the training data set and send it into the training network.

• Each layer's output, generated by the activation function, are continuously led into the next hidden layer till the output layer.

• Calculate the deviation matrix of the output.

• Conduct a layer-by-layer reverse calculation according to gradient descent algorithm.

• Acquire the updated weight and gradient values.

• Repeat the forward propagation to start the next iteration.

YCbCr Color Space for CNN's Feature Extraction:

The previous traffic signs recognition and classification methods usually take CNN training in RGB space. In that space, the color information and the luminance information are mixed among channels, generating the variance of the extracted features. Nonetheless, the color distribution in RGB space is not uniform. Some subtle changes in color are captured difficultly.

YCbCr color space is mainly used for continuous image processing in the video. We can calculate each component by the transform formula shown as:

$Y = 0.299R+0.587G+0.114B$ (1)

$Cb = 0.564(B -Y)$ (2)

$Cr = 0.713(R -Y)$, (3)

where Y is luminance component, Cb represents the

blue chrominance component and Cr denotes the red chrominance component.
In order to choose a better color space for feature extraction, we conduct an error evaluation among
three typical color space, i.e., Grayscale, RGB and YCbCr. To do this, we send the training data set for CNN training. Taking the bn as the number of wrong samples in each batch training, the n as the number of batch training time, the S as the batch size. We are able to calculate the training error in each batch training which represented by En:
En = bn / S

The results of different training errors shown as Table 1, we can find that the corresponding result of YCbCr space is better than the others.

Color Space	Training Error
Grayscale	0.138
RGB	0.105
YCbCr	0.073

Table 1.16

PARAMETERS ADJUSTMENT BASED ON CNN-SVM:

In this section, we will take training data set and testing street-view images to conduct parameters adjustment and image preprocessing in training and testing
progress respectively. The goal of parameters adjustment is to increase training accuracy and to shorten

the training time in the training parts. While in the testing progress, the image preprocessing operation is mainly used to eliminate the negative impacts, e.g., insufficient illumination and similar background, so as to acquire ROIs precisely.

Layer	Type	Neurons	Kernel
0	Input	48×48	
1	Convolutional	44×44	5×5
2	Subsampling	22×22	2×2
3	Convolutional	18×18	5×5
4	Subsampling	9×9	2×2
5	Convolutional	5×5	5×5
6	Fully	1×600	

Table 1.17

2. Traffic-Sign Detection and Classification under Challenging Conditions: A Deep Neural Network Based Approach

Methods:

A. Analysis of Dataset:

A video dataset was provided for the IEEE VIP Cup 2017, in order to test and train the traffic-sign detection model. A brief summary of the dataset is given Table I.
There are a total of 98 unique video sequences in the dataset, half of which are real and the other half synthesized. 14 types of traffic signs are present in the dataset, and, 12 types of challenges have been simulated in the video sequences. The challenge types are categorized and stated below :

• No Challenge (00)

• Color Type: Decolonization (01)
• Blur Type: Lens blur (02), Gaussian blur (07)

• Illumination Type: Darkening (04), Shadow (10), Haze (12),Exposure (06)Weather Type: Rain (09), Snow (11)

• Other: Codec Error (03), Dirty Lens (05), Noise (08)

Note: 5 levels of severances have been simulated for each challenge type. Thus, with 300 frames per video, the total number of images for each specific challenge type and level is
$98 \times 300 = 29{,}400$.

B. Model Proposal:
1) First, we detect the type of challenge present in the image by using a

CNN, similar to the VGG-16 architecture

2) Then, we preprocess the image to enhance traffic-sign features depending on the type of challenge detected. We adopt a variant of Histogram-Equalization for preprocessing images affected with challenges of Illumination Type and Decolorization. We also use a CNN of ResNettype architecture to denoise images affected with Rain

3) Next, we localize traffic-sign proposals from the preprocessed image. There are 3 localizing networks in this stage - one for Lens Blur and Gaussian Blur, one for Noise and a Common Localizer for all of the other types of challenges. The task of localizing sign-proposals is performed by using a deep CNN of U-Net Architecture.

4) Finally, we extract the sign-proposals predicted by the localizer from the preprocessed image, and, pass them to yet another CNN for classification. The architecture of this CNN is similar to that used in detecting the challenge-type in the first step, but, with more depth, so that it can attain higher classification accuracy.

C. Architecture

1) Challenge Detection: The description of the Challenge Detection CNN is given in Table II. It consists of four blocks, each with 2 convolution layers, a ReLU Activation Layer and a Max-Pooling layer. The output after these 4 blocks is then flattened, and, a fully connected layer is used to map this flattened feature vector to 12 classes. These 12 classes represent the 12 types of challenge that are to be detected.

2) Selective Preprocessing: Before passing the image on to the corresponding localizer for detecting traffic-signs, it is first preprocessed. The preprocessing method that we choose to use is a variant of Histogram-Equalization, called, Contrast Limited Adaptive Histogram Equalization (CLAHE). This version of histogram equalization works on a local neighborhood of the image and limits contrast to prevent enhancement of noise.
To do this, we first convert the image from RGB to HSV (Hue, Saturation, Value) space, and then, equalize the histogram of the proper HSV channel. The equalizing operations that are performed for the different challenge types are given in Table III. The table states the number of applications of CLAHE to the Saturation or Value channel, and, the Contrast Clip-Limit used. The result of our preprocessing method, performed on level 4 noisy images

3) Traffic-Sign Localization: This stage of the model is implemented by using a Fully Convolutional Neural Network inspired by the U-Net Architecture.The detailed network architecture is illustrated in Fig. 4. The network consists of 2 stages: a downsampling stage and an upsampling stage, which are connected by using a Residual Learning strategy.

a) Down-sampling:

This stage consists of several convolution blocks, each with 2 convolution layers, a Batch Normalization layer and a ReLU Activation layer. After each block, a
Max-Pooling operation is performed for down-sampling. The number of feature channels is doubled at each down-sampling step.

b) Up-sampling:

During upsampling of the feature map,
at each step, it is concatenated with a corresponding cropped feature map from the downsampling stage. Here, cropping is necessary for the loss of border pixels in every convolution. Finally, two more convolution layers are added, which map each component feature vector from the previous layer to 2 classes The motivation for using the U-Net architecture is its robustness against low-resolution features of the image during training. In the up-sampling layers employed within it, there are
a large number of feature channels which allow the network to propagate context information to higher resolution layers. This is quite an attractive feature for localizing objects in noisy conditions where the target object may not be dominantly visible in the image.

The U-Net architecture produces the location of the object by performing a pixel-wise image-segmentation. It predicts a probability for each pixel based on its likelihood of being present in a region corresponding to one of 2 classes (TrafficSign and Background). So, the traffic signs are detected as islands of high-probability regions in the image. The bounding box of these regions are the predicted traffic signs in this image.

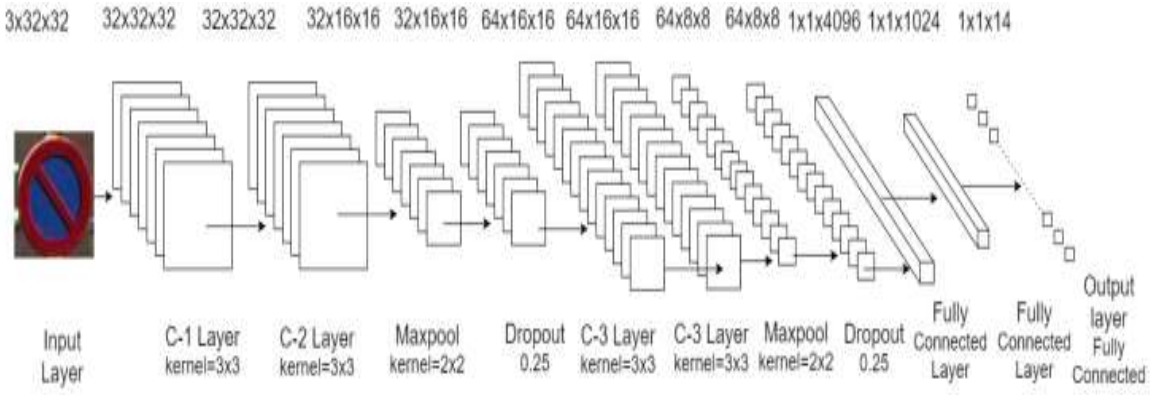


Figure 1.5

4) Classification: The architecture of the CNN used for classifying the traffic sign proposals is shown in Fig. 5. It has 2 Convolution Blocks, each with two consecutive Convolution layers, a ReLU activation layer, and a Max-Pooling layer. A dropout layer is added after each block. Finally, there are 2 fully connected layers - the first one has a hidden size of 1024 with ReLU activation, and, the latter has a hidden size of 14 with Softmax activation. The output of this layer are the classprobabilities for each of the 14 sign-classes

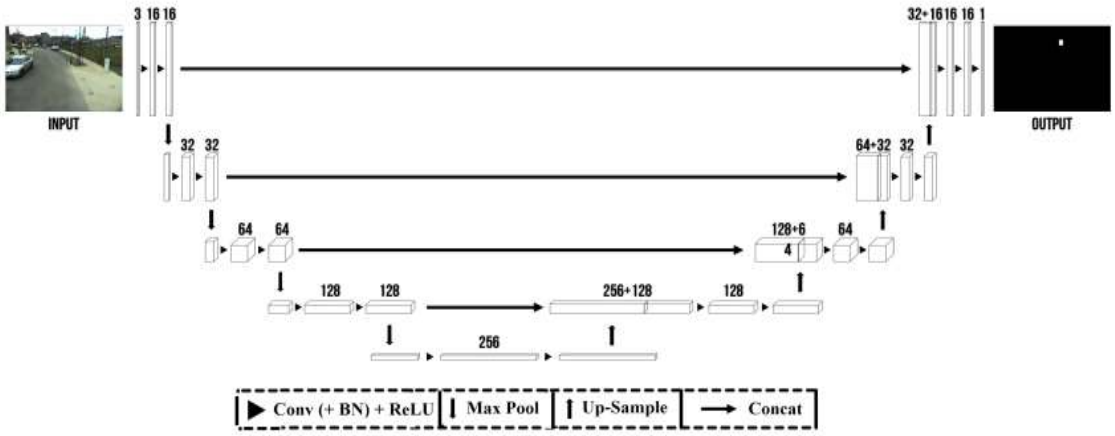


Figure 1.6

3. Real-Time Traffic Sign Recognition using YOLOv3 based Detector

Approach:

Our approach for traffic sign recognition based on YOLOv3 is presented. The traffic sign recognition pipeline, consists of YOLOv3 detector trained for detecting the candidate traffic signs, a bounding box pre-processor, which enlarges the detected bounding box, crops and resizes the boxes containing candidate traffic signs, and a CNN based classifier which classifies the candidate traffic sign as belonging to one of the 43 classes.

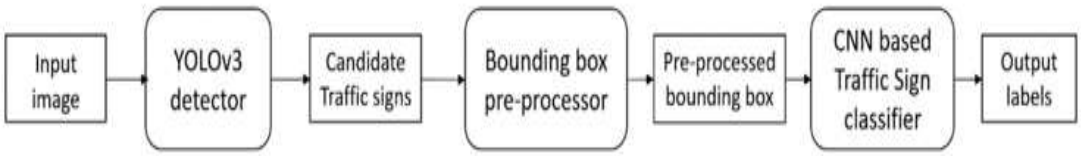


Figure 1.7

A. YOLOv3 based Traffic Sign Detection:

YOLO is a CNN-based object detection network. It offers real-time object detection performance by considering detection as a regression problem rather than several classification problems. A single stage detection network is used to take in image pixels as input and to predict bounding box coordinates and class probabilities. The whole detection process is performed in a

single evaluation of the image. YOLO divides the input image into grids and predicts bounding boxes, confidence of those boxes, and class probabilities simultaneously. YOLOv2 and YOLOv3 are improved

versions of the base YOLO network with several enhancements to increase the accuracy and detection speed. One of the important improvements in YOLOv3 is the multi- scale prediction, which helped in overcoming the difficulty in small object detection existed in its predecessors. Also, it uses logistic regression to predict the objectness score and uses multiple independent logistic classifier per class.

YOLOv3 is used as the traffic sign detector in the proposed method. YOLOv3 uses a base network which can be considered as a hybrid between YOLOv2 network, DarkNet- 19 and a Residual network. The resulting network consists of
53 convolutional layers and therefore called DarkNet-53, which is the feature extractor. YOLOv3 network considers the image as an S × S grid of sub-images or cells and predicts bounding boxes and class probabilities for each sub-image.

The network structure of YOLOv3 based traffic sign detector is shown in the Fig. 3. It consists a total of 75 convolutional layers with no pooling or fully connected layers. Instead of pooling layers, convolutional layers with stride 2 are used to down-sample the feature map. The structure allows to predict traffic signs from images of any size. Residual connections similar to ResNet [19] and a multiscale detection mechanism similar to Feature Pyramid Network (FPN) [4] are used for accuracy improvement and small object detection in YOLOv3. Through this architecture, network becomes capable of identifying traffic signs of different sizes present in the image. For this, detection is done in feature maps of different scales in YOLOv3. To make the features suitable for detection at different scales, the feature maps from deeper levels are up-sampled and merged before detection at a particular scale. In YOLOv3 detector, a 1 × 1 convolutional layer with logistic regression is used for enabling multi-label classification, instead of SoftMax.

YOLOv3 uses DarkNet53 based feature extractor network and fully convolutional detector networks at three different scales. Each block shows the type of the layer, the stride, number of filters and the filter size.

1. DarkNet53 feature extractor: DarkNet53 network is used for feature extraction purpose. It uses a fully convolutional network

with residual connections. Every convolutional layer is followed with a Batch Normalization layer and uses Leaky ReLU activation. Residual blocks with convolutional layers and shortcut paths make the network learn features when the network goes deeper. DarkNet53 layers are indicated in the network structure.

2. Detector subnetwork at various scales: The extracted features are used by convolutional subnetworks for detection of traffic signs at various scales, which correspond to big, small and medium sized signs. The last stage feature maps are combined with feature map outputs from earlier stages and the concatenated activations are used by the convolutional networks for different scales to generate the output tensors of size given by: Number of bounding boxes × (4 + l + Number of classes) = 3 × (4 + l + l) = l8

3. Loss Function: YOLOv3 loss function is a multi-task loss function given by: L = LSoc + LcSc The first two terms of the loss function signify the localization loss (regression loss). The last three terms constitute the classification loss. Localization loss is the squared error between predicted bounding box coordinates, (x^t, y^t, w^t, h^) and the ground truth box coordinates, (xi, yi, wi, hi). Classification losses are cross entropy loss terms. Of the last three terms, the first one penalizes the objectness score prediction for those bounding boxes responsible for predicting objects, the second one for bounding box with no objects and the last one penalizes the class prediction for bounding box which predicts the objects. In the loss function equation, Zcoord is the scale parameter which controls how much to increase loss from bounding box coordinate predictions, Znoobj is the scale parameter which controls how much to decrease the loss of confidence score predications for boxes with no objects 1iobj indicates whether the grid cell i contains an object, 1 obj indicates whether jth bounding box of ith grid cell is responsible for prediction of object, Ci is the confidence score of ith cell, C^ is the predicted confidence score, classes indicate the set of all classes (for our case, it is

ij

t

1 since only traffic sign class is available), pi(c) is the conditional probability of whether ith cell contains object of class c c classes, p^i(c) is the predicted conditional class probability.

IEEE - 45670

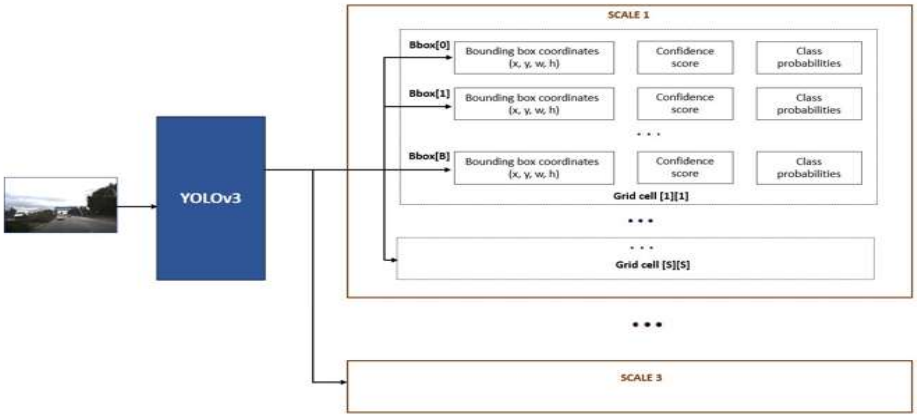


Figure 1.8

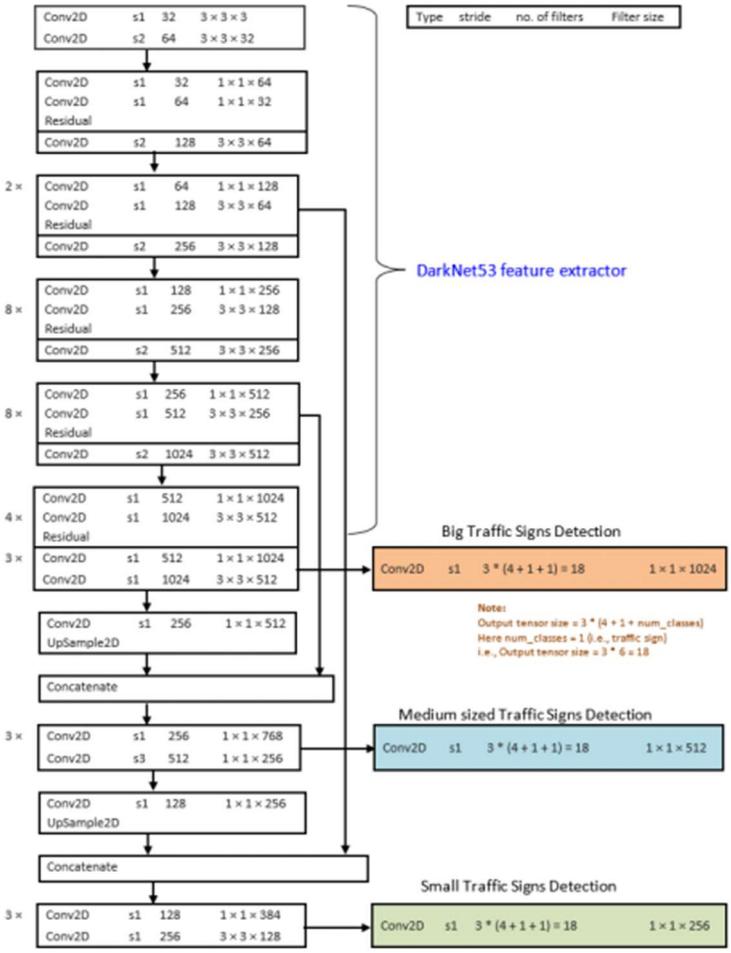


Figure 1.9

B. Bounding Box Pre-processor:

Bounding Box Pre-processor stage extracts and prepares the detected candidate traffic sign boxes for classification. The center of the regressed bounding box is determined and the box is enlarged by 25% to compensate for any regression errors to make sure the traffic sign is completely enclosed in the region. The enlarged boxes are cropped and resized to 48×48 size and fed to the classifier network for recognition of the traffic sign class.

C. Traffic Sign Classifier:

The classifier network structure is in Table I. In this architecture, an n × n convolution is replaced by an n × 1 convolution followed by a 1 × n convolution, which reduces both the number of convolution operations and the network parameters. This leads to computational cost reduction and increased speed.

Batch Normalization and ReLU layers follows all layers other than the final dense layer. The sixth layer forms an inception module where kernels of different sizes are used to extract information from feature map output of the previous layer. The output feature maps from the inception layer are concatenated to combine the feature maps. Dropout layers are also used to regularize the activations of the final stages. To recognize the 43 traffic sign classes, fully-connected layer of an output size of 43 with Softmax activation is used as the last layer.

Layer	Type		Filter Size/ Parameter	Filters/ Stride	Output shape
1	Conv		3 × 3	32, s1	48 × 48 × 32
2	Conv		7 × 1	48, s1	48 × 48 × 48
3	Conv		1 × 7	48, s1	48 × 48 × 48
4	MaxPool		2 × 2	s2	24 × 24 × 48
5	DropOut		0.2		24 × 24 × 48
6-1	Inception	Conv	3 × 1	64, s1	24 × 24 × 64
7-1	Inception	Conv	1 × 3	64, s1	24 × 24 × 64
6-2	Inception	Conv	1 × 7	64, s1	24 × 24 × 64
7-2	Inception	Conv	7 × 1	64, s1	24 × 24 × 64
8	Inception	Concatenate	-	-	24 × 24 × 128
9	MaxPool		2 × 2	s2	12 × 12 × 128
10	DropOut		0.2	-	12 × 12 × 128
11	Conv		3 × 3	128, s1	12 × 12 × 128
12	Conv		3 × 3	256, s1	12 × 12 × 256
13	MaxPool		2 × 2	s2	6 × 6 × 256
14	Dropout		0.3	-	6 × 6 × 256
15	Dense		256	-	256
16	Dropout		0.4	-	256
17	Dense - Softmax		43	-	43

Table 1.18

Traffic Sign Architecture (Number of layers, Type of each layer, Size of each layer, Filters and Strides and The output shape)

Comparison between Mentioned Papers

Paper	Accuracy	Data Set
Traffic Signs Recognition and Classification based on Deep Feature Learning	98.6% Classification	German Traffic Signs Recognition Benchmark (GTSRB)
Traffic-Sign Detection and Classification under Challenging Conditions: A Deep Neural Network Based Approach	62% Detection and Classification over all	IEEE VIP Cup 2017
Real-Time Traffic Sign Recognition using YOLOv3 based Detector	99.59% Classification And 92.2% Detection	German Traffic Sign Detection Benchmark (GTSDB) and German Traffic Sign Recognition Benchmark (GTSRB)

Table 1.19

3-Applications:

Traffic sing detection and Classification is used in many applications and the main of these applications is the self-driving cars.

Self-Driving Cars:

Through a camera which in the self-driving cars, I takes images form the street and get them as input to the application, then the application detect the traffic signs in the image and classify them to make the car control itself according to the sing and the road.

Prevent Accidents:

Now days, many drivers do not notice the traffic sing on the road and some others do not know the meaning of the sing specially if you are not a citizen form the country so, this help the drivers on the different roads to understand the meaning and notice each one of the traffic signs.

4-Motivation:

- According to ***global road crash statistics***, nearly 1.3 million people die in road related accidents each year, which is 3,287 deaths per day on average ***[ASIRT, 2014]***.

- Sadly, driver error due to drunk driving, reckless driving, and fatigue and driver distraction remain the leading causes of deaths on the road.

- An on-board ***computer vision*** system that could detect and identify road-signs may help avoid accidents by assisting the driver in a number of ways.

- The on-board ***vision system*** could serve to augment reality and display upcoming warning signs early on, or keep them displayed on a screen even once the sign has passed.
  This would decrease the likelihood that the driver failed to see an important sign**.**

- The ***vision system*** may also connect to the mechanics of the car, automatically slowing the car to the speed limit or even slowing the car before sharp bends in the road.

- Navigation Another motivation for sign reading capabilities in vehicles may be to navigate in dense urban environments with limited ***GPS*** availability.

- A precise location could be determined by identifying unique road signs and looking up their relocation in an image database.

- Although this may be attractive for marketing purposes, road safety and convenience remain the leading motivations for traffic sign recognition.

5-Objectives:

- Aim to propose a method for sign detection that is able to pass regions of interest to a classifier at more than ***25 frames per second.***
- The regions of interest should contain minimal false positives and must have ***associated shape classifications*** to make the sign classification more efficient.
- The proposed system settings should perform in typical midday sunny conditions. This restriction is imposed to ensure high accuracy in a subset of ***illumination conditions and serves as a proof of concept***.
- Future work may apply different detection pre-sets based on the ***global illumination*** in a given frame. Unlike surveillance systems where the camera remains fixed in its relocation, this optical sensor is expected to move through space.
- This makes ***detection more challenging*** because the background is constantly changing, meaning the relationships and patterns between pixels are in constant change.
- It is therefore a requirement that the identification of the position of an object of interest is robust to changes in the background in the given illumination environment. The ***lighting conditions*** are expected to change drastically from day to day, and system settings will need to be based on the ***global illumination*** in order to account for these larger ***lighting variations***.
- This approach will attempt to solve problems with ***detection*** associated with daytime conditions driving at high speed, and naturally, small changes in lighting must be accounted for.

6-Time Plan:

Here is Our Time Plan in Each Phase of the Project.

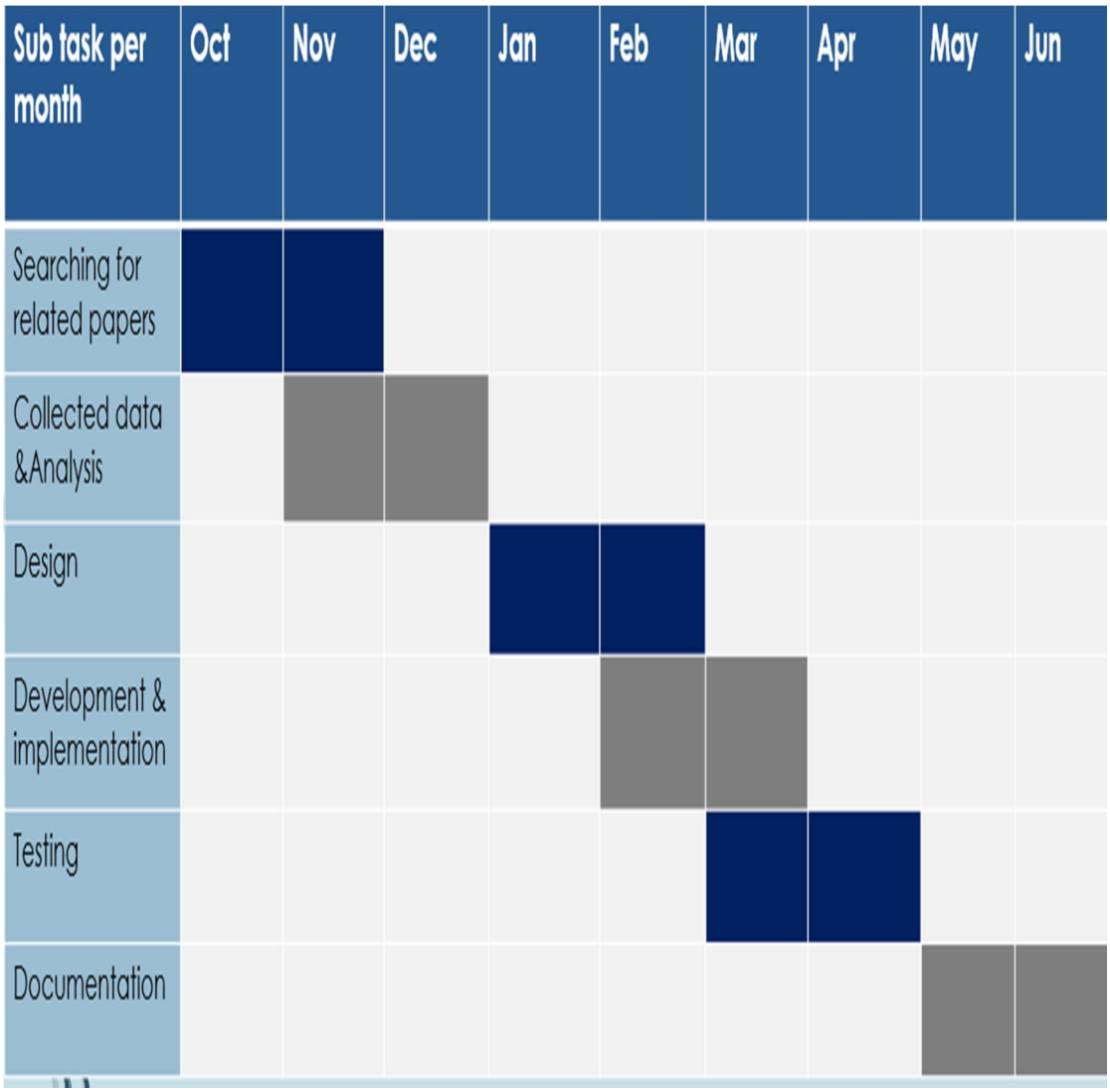


Figure 1.10

7-Document Outline:

Here is the Outline (Content) of the documentation

Chapter 2: Background:

In this chapter, we will define all the information about our system including:

1. System Overview
2. Mathematical Background
3. Algorithms
4. Required Techniques

Chapter 3: System Architecture:

In this chapter, we will show the architecture of our system including the main components and the relations between them.

Chapter 4: System Implementation:

In this chapter, we will describe the implementation of the system in details.

Chapter 5: System Testing:

In this chapter, we will describe how to test the system with the input and show the output (results) with screen shots.

Chapter 6: Conclusion and Future Work:

In this chapter, we will discuss about two points:

1-Conclusion: here we will summarize the project and list the advantages and disadvantages.

2-Future Work: here we will list points that will help in the evolving the project and get the best results in the next year.

Tools:

In this section, we will list all the software and hardware tools we used in building our system.

References:

In this section, we will list all the books, links, papers and researches we used to build our system.

Chapter 2: Background

1-System Overview:

Our System consists of two models:

1- Detection Model:

This model is responsible of detecting all the traffic sings in the input image or video through, using the ***YOLOv3 Technique***. The output of this model is that, ***bounding boxes*** surrounding the sings in the image or the frames of the video (***Localization***) through using techniques of ***deep learning*** and ***probability.***

2- Recognition Model:

This model is responsible of ***classifying*** the signs. First, it cut the input image according to the ***surrounding box*** around the traffic sing. Get each sub image from the original image and classify each sign using ***Convolution layers***, ***Batch Normalization layers*** and the ***RELU*** Notify each Box according to the class with a label (***the number of the class which is the type of the traffic sign***) ***The last output*** is image with all ***labeled Boxes***. ***In case of videos***, we combine all frames with labeled Boxes.

2-Mathmatical Background:

1- Detection Model:

- ***The RELU Activation function*** is the ***squashing function*** we use in this model to detect the traffic sings in the Input image as we can see the function is given as

  f(x) = n, if n >= 0
  or
  0, if n < 0

  as shown in the figure:

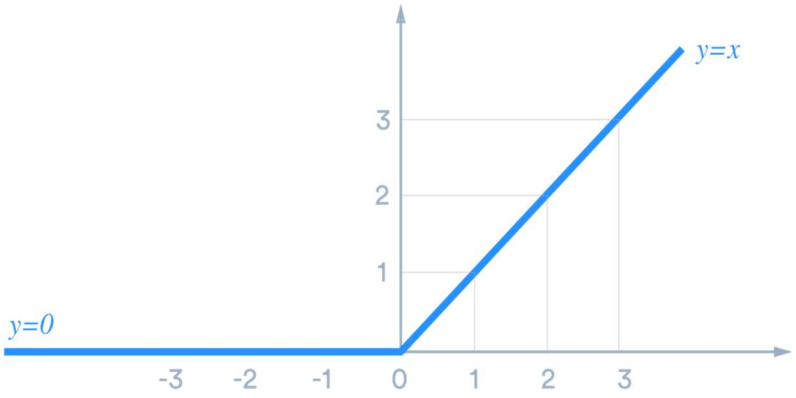


Figure 2.1

- We use the ***RELU*** as it prevents the ***weight vanishing*** that, the weight may reach ***zero*** and then the model will learn nothing.
- We use advanced version of ***RELU*** which is the ***Leaky RELU*** and this to handle the ***negative section*** output numbers to prevent the ***RELU Explosion*** (***That may the output be a large negative number so when become Zero, there will be a big loss in the information the model need to learn***).

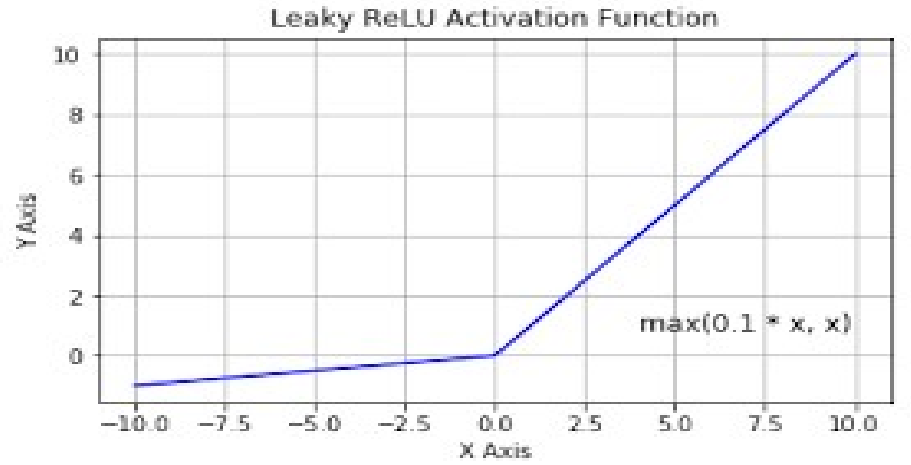


Figure 2.2

- The input image is from dimensions of W * H * D (Width * Height * Depth) and we calculate the new depth after getting boxes using the YOLO which is calculated using the equation of:

  D = (5 + number of boxes) * number of classes
  Number of Boxes: is the number of desired boxes around each sign.

  Number of Classes: is the number of classes in our case, there are 43 different traffic signs.

  Five: indicates for the parameters of each box and they are (X, Y, W, H, P).
  X: the x-axis coordinate of the box.
  Y: the y-axis coordinate of the box.
  H: the x-axis coordinate of the total image.
  W: the y-axis coordinate of the total image.
  P: the probability that, the box contains the center of the object we want to detect.

- We use the ***YOLOv3*** that we use the ***pertained model DarkNet*** and this enhance the performance of the detection in the image for better classification.

- Loss Function: YOLOv3 loss function is a multi-task loss function given by:

$$L = LSoc + LcSc$$

- The first two terms of the loss function signify the localization loss (regression loss). The last three terms constitute the classification loss. Localization loss is the squared error between predicted bounding box coordinates, (x^t, y^t, w^t, h^) and the ground truth box coordinates, (xi, yi, wi, hi). Classification losses are cross entropy loss terms. Of the last three terms, the first one penalizes the objectness score prediction for those bounding boxes responsible for predicting objects, the second one for bounding box with no objects and the last one penalizes the class prediction for bounding box which predicts the objects. in the loss function equation, Zcoord is the scale parameter which controls how much to increase loss from bounding box coordinate predictions, Znoobj is the scale parameter which controls how much to decrease the loss of confidence score predications for boxes with no objects 1iobj indicates whether the grid cell i contains an object, 1 obj indicates whether jth bounding box of ith grid cell is responsible for prediction of object, Ci is the confidence score of ith cell, C^ is the predicted confidence score, classes indicate the set of all classes (for our case, it is 1 since only traffic sign class is available), pi(c) is the conditional probability of whether ith cell contains object of class c c classes, p^i(c) is the predicted conditional class probability. S^2 .

- ***Bounding Box Pre-processor stage*** extracts and prepares the detected candidate traffic sign boxes for classification. The center of the regressed bounding box is determined and the box is ***enlarged by 25%*** to compensate for any ***regression errors*** to make sure the traffic sign is completely enclosed in the region. The enlarged boxes are cropped and resized ***to 48 × 48 size and fed to the classifier network for recognition of the traffic sign class***.

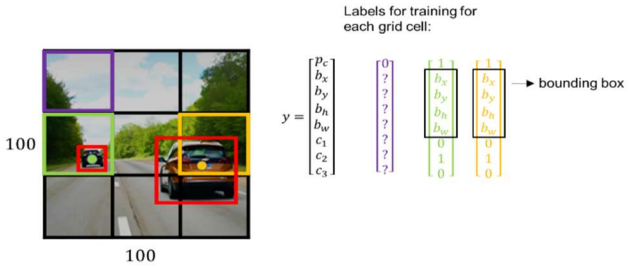


Figure 2.3

2- Recognition Model:

- We here have a ***43 Classes*** of different Traffic signs.

- By using ***convolution layers***, ***fully connected layers***, ***batch normalization layers***, ***dropout***, ***RELU layers*** and the suitable number of ***epochs***, we can extract the most important ***features***.

According To The Detection Model

- We also use the ***DarkNet53*** to extract the ***unique features*** of each sign and this help us to predict the output sign in the ***architecture***.

- The Dimensions of the output of each layer is the ***same width and height of the input but the depth is the number of layers in the batch***.

- We use also ***the Intersection over Union Evolution Criteria***, as they may be overlap boxes resulted from the ***YOLOv3*** and then we must use threshold to get the right box ***surrounding the traffic sign for the recognition***.

- The ***IoU threshold is 0.85*** (***the score of the overlapped boxes must be more than 0.85 to accept***.)

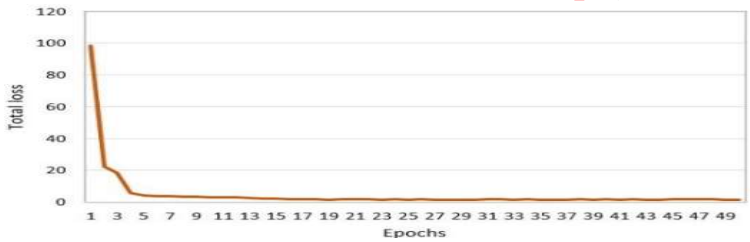


Figure 2.4

3-Algorithms and Techniques:

Here we will list the Algorithms and the efficient techniques we used for building our system

- ***Convolution Network:***

The convolution layers are used to extract the features form the input sub images.

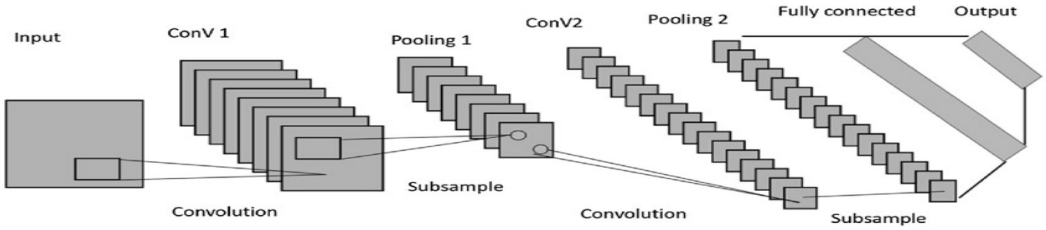


Figure 2.5

- ***Batch Normalization:***

The Batch Normalization layers are used to save the normalization of the input through the layers by trainable parameters.

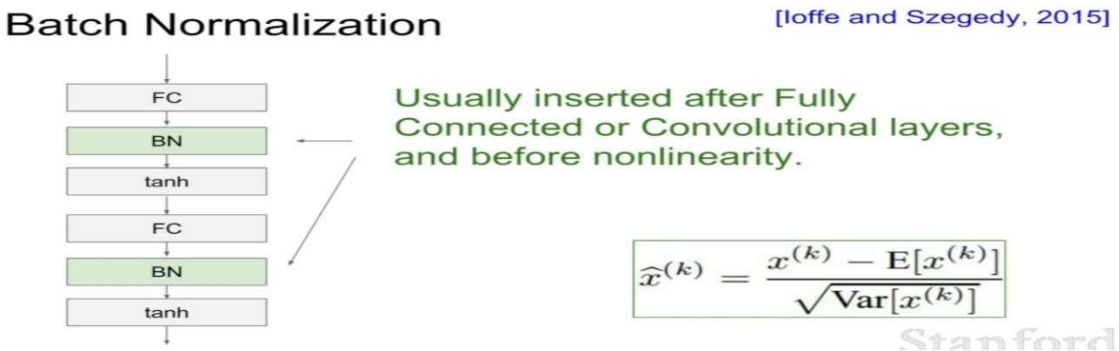


Figure 2.6

- ***RELU Layers:***

The RELU activation function is the best activation function in our situation and we use the leaky one as mentioned before to get the output to classify the sub images in the recognition model.

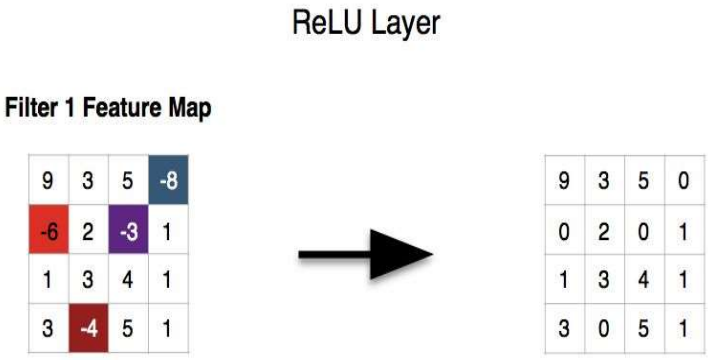


Figure 2.7

- ***Max Pool Layers:***

Max Pool Layers are used to get the non-linear features form the input of convolution layers and get the most important features through getting the max values.

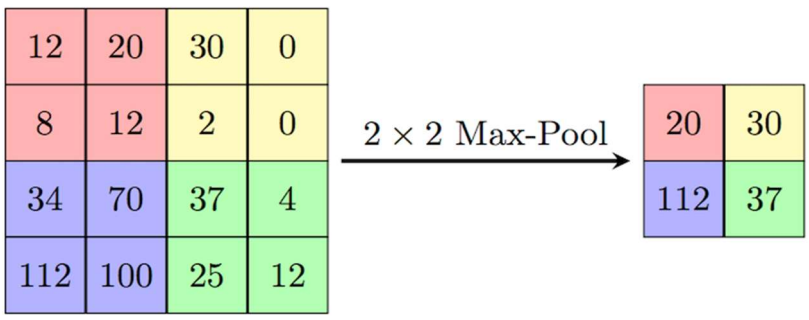


Figure 2.8

- ***Residual:***

  This technique is used to get the input of some convolution layers not only form the previous layer but from past layers also and this help us to avoid the over fitting by getting more features in the future layers.

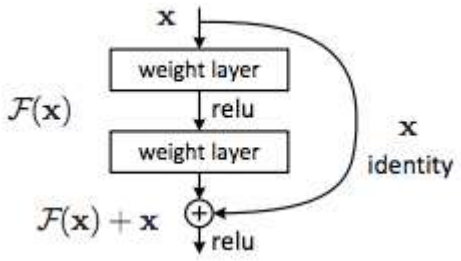


Figure 2.9

- ***Soft Max Activation Function:***

  This function is used in the fully connected layer to get the probability on the input at each class then take the largest one to be the class of the input sub image.

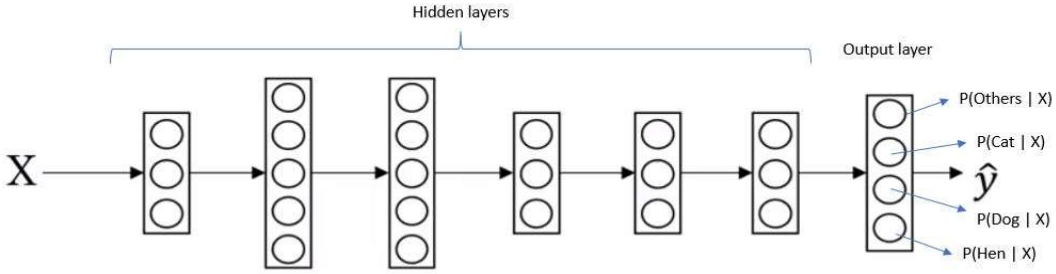


Figure 2.10

- ***Fully Connected Layer:***

This is the layer of 43 Class and the last layer to classify the input by using the dense of the input and the soft max activation function.

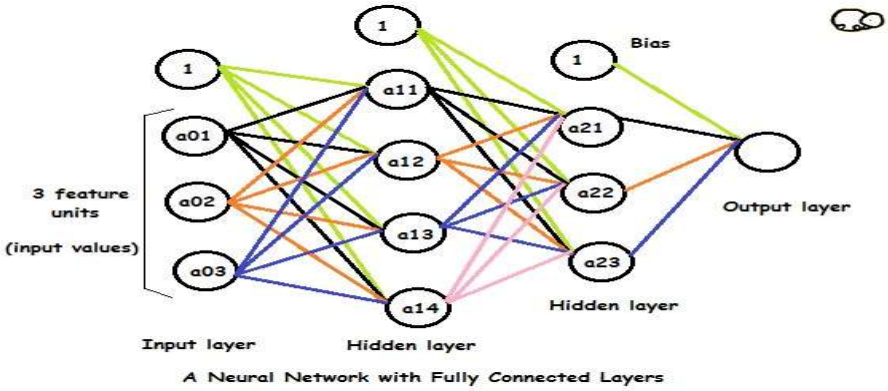
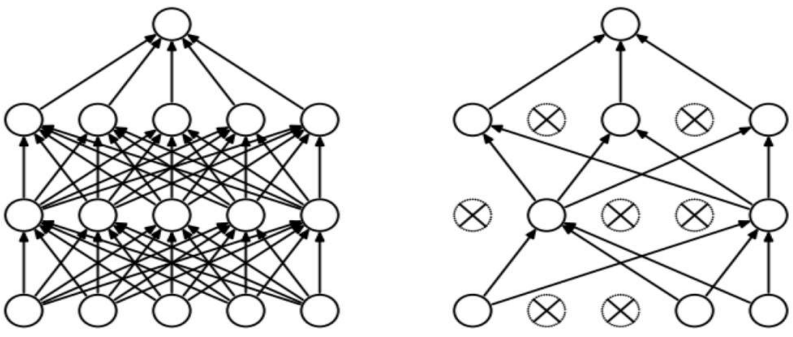


Figure 2.11

- ***Dropout:***

This technique is used to avoid the over fitting by cancel some nodes and work with the other according to the probability of each one and compare it with the given threshold. The probability that is less than or equal to the threshold will be canceled in this epoch.



(a) Standard Neural Net (b) After applying dropout.

Figure 2.12

- ***YOLOv3:***

  we use the YOLOv3 in detecting the traffic sings in the given images by making virtual boxes surrounding the sing.

- ***DarkNet53:***

  The Dark Net 53 is a pre-trained model which is used in extracting the features of the given sub image in the recognition model.

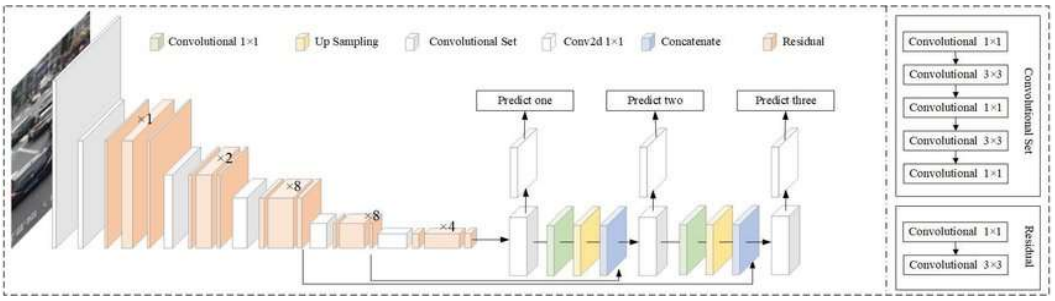


Figure 2.13

- ***Intersection over Union (IoU):***

  This technique is used when the output of the YOLOv3 in some areas of the image is that, to overlapped boxes so we need to get the right box around the traffic sign by using this approach.

Figure 2.14

- ***Non Maxima Suppression:***

This algorithm is used for the accuracy of the YOLOv3, as some times there is boxes with probability, which is 0.2, 0.25, or lower than that, so we need to get rid of those boxes for the recognition model.

so this algorithm take a threshold of 0.5 and all boxes with probability that less than of equals to this threshold will be eliminated.

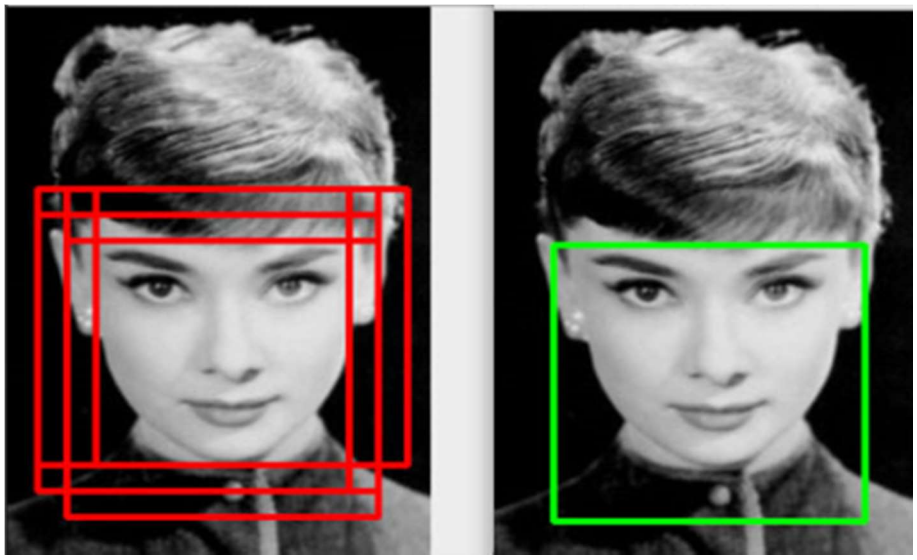


Figure 2.15

Chapter 3: System Architecture

Now we will discuss about the architecture of our system we used to get the best accuracy.

We here will discuss the architecture through three main points

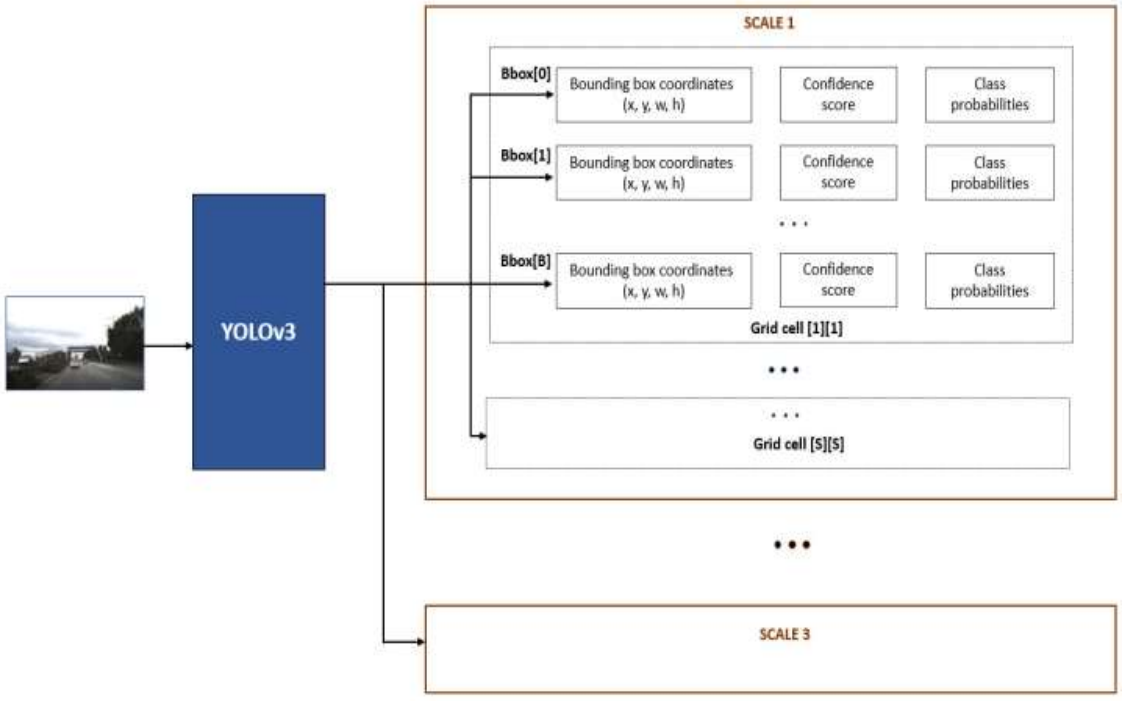


Figure 3.1

Design for the First Stage Architecture

1-Overview Approach:

- Our approach is that; take the input image (***or the frame of the video***) as an ***input*** to the ***YOLOv3***.

- The output of this process from the ***YOLOv3*** with the ***DarkNet53*** is a collection of ***Bounding Boxes surrounding the Traffic Sings*** in the image.

- Each Bounding Box is defined by:

  - ***Bounding Box Coordinates:***
    1. ***X:*** the x-axis coordinate of the box.
    2. ***Y:*** the y-axis coordinate of the box.
    3. ***W:*** the width of the image.
    4. ***H:*** the height of the image.

  - ***Confidence Score:***
    The Score of confidence that the box contains the desired object we want to detect which is the traffic sign.

  - ***Class Probability:***
    The probability that the object belong to specific class (specific type of traffic sign).

2-Detecition Model (YOLO Architecture):

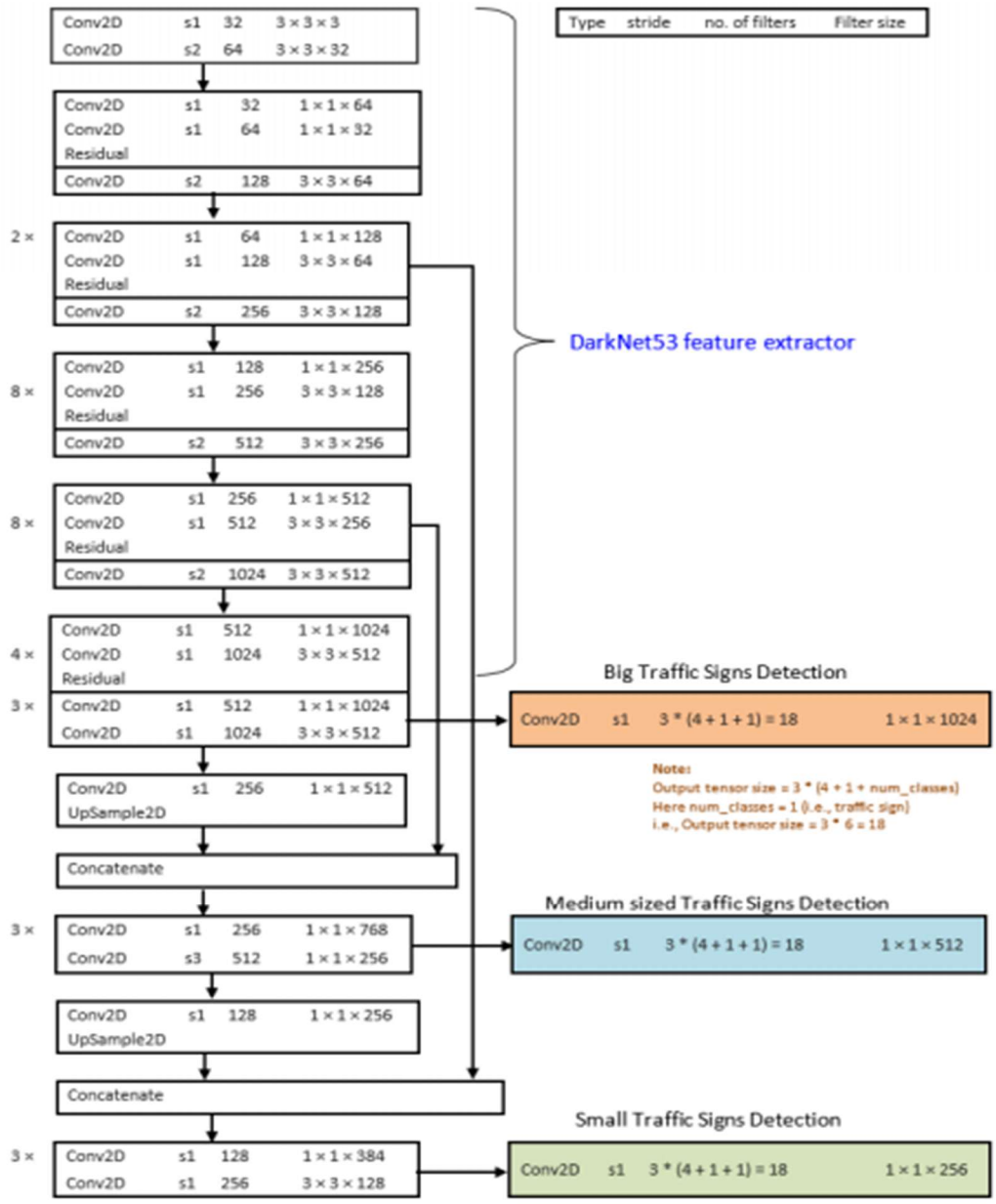


Figure 3.2

- First, we have the ***DarkNet53*** that consists of ***2D convolution networks the first one with 32 layers with size of 3*3*3 and the second one with 64 layers with size of 3*3*32***.

- There is two other convolution networks ***with residual technique to feed the concatenate layers with 32 layers with size of 1*1*64 and the other one with 64 layers with the size of 1*1*32 and the combine of the two sizes is 128 layers with size of 3*3*64.***

- The third stage consists of two ***convolution networks, the first one is 64 layers with size of 1*1*128 and the second one is 128 layers with size of 3*3*64***, also we use the ***residual technique*** to feed the ***concatenate layers***, with ***the combined output 256 layers and the size of 3*3*128.***

- The next two convolution networks are 128 layers 1*1*256 and the second one with 256 layers with the size of 3*3*128 we use the ***residual technique*** to feed the ***concatenate layers***, with ***the combined output 512 layers and the size of 3*3*256.***

- The next two convolution networks are 256 layers 1*1*512 and the second one with 512 layers with the size of 3*3*256 we use the ***residual technique*** to feed the ***concatenate layers***, with ***the combined output 1024 layers and the size of 3*3*512.***

- The next two convolution networks are 512 layers 1*1*1024 and the second one with 1024 layers with the size of 3*3*512 we use the residual technique to feed the concatenate layers.
  The outputs of two convolution network with 512 layers, the size of 1*1*1024, 1024 layers, and the size of 3*3*512.
  This helps us in the detection of the big traffic signs in the image.

- Then use a single network convolution with 256 layers and size of 1*1*512.
  In addition, we use the up sampling in 2D to get some number of features for the good training of the model.

- ***After the concatenate layer***, there are two convolution networks; the first one is 256 layers with size of 1*1*768 and the second one 512 layers wit size of 1*1*256.
  This helps us to detect the medium size of the traffic sings in the image.

- Then use a single network convolution with 128 layers and size of 1*1*256.
  In addition, we use the up sampling in 2D to get some number of features for the good training of the model.

- Then we get to the ***concatenate layer***.
  This is fed by the residual technique in the past layers.
  This help us to adjacent the number of features to train the model to get high accuracy in the detection phase.

- There is two other convolution networks ***with 32 layers with size of 1*1*64 and the other one with 64 layers with the size of 1*1*32 and the combine of the two sizes is 128 layers with size of 3*3*64. This helps us to detect the small size of the traffic sings in the image.***

- ***In the Big, Small and Medium Traffic Signs the number of boxes is the same which is 3*(4 + 1 + 1) = 18 Bounding Boxes***

- In the ***Big*** Signs there is Convolution layer with ***1*1*1024***
- In the ***Medium*** Signs there is Convolution layer with ***1*1*512***
- In the ***Small*** Signs there is Convolution layer with ***1*1*256***
- ***This helps us the adjacent the size of the bounding box according to the size of the traffic sign in the image.***

3-Recognition Architecture:

Layer	Type		Filter Size/ Parameter	Filters/ Stride	Output shape
1	Conv		3 × 3	32, s1	48 × 48 × 32
2	Conv		7 × 1	48, s1	48 × 48 × 48
3	Conv		1 × 7	48, s1	48 × 48 × 48
4	MaxPool		2 × 2	s2	24 × 24 × 48
5	DropOut		0.2		24 × 24 × 48
6-1	Inception	Conv	3 × 1	64, s1	24 × 24 × 64
7-1	Inception	Conv	1 × 3	64, s1	24 × 24 × 64
6-2	Inception	Conv	1 × 7	64, s1	24 × 24 × 64
7-2	Inception	Conv	7 × 1	64, s1	24 × 24 × 64
8	Inception	Concatenate	-	-	24 × 24 × 128
9	MaxPool		2 × 2	s2	12 × 12 × 128
10	DropOut		0.2	-	12 × 12 × 128
11	Conv		3 × 3	128, s1	12 × 12 × 128
12	Conv		3 × 3	256, s1	12 × 12 × 256
13	MaxPool		2 × 2	s2	6 × 6 × 256
14	Dropout		0.3	-	6 × 6 × 256
15	Dense		256	-	256
16	Dropout		0.4	-	256
17	Dense - Softmax		43	-	43

Table 3.1

Note: **S1 and S2 mean the Stride of the Convolution Layers.**

Note: **Inception means Convolution network.**

Note: **Filter means the number of Layers.**

Note: **Size is the size of the layer.**

Chapter 4: System Implementation

In this chapter, we will see the implementation of each component of our system.

The components are:

- ***Classifier Model:*** That will classify the Traffic Signs.
- ***Detection Model:*** That will detect the locations of the traffic sign in the image.
- ***Connection of Detection and Classifier Models:*** That will connect the output of the detection with as the input of the classifier (the locations of the traffic signs in the images), then the final output is the notation of each location (box) surrounding the traffic sign.

1. *Classifier Model:*

This model is for the classification of the Traffic signs according to the ***boxes surrounding*** each traffic sign ***(the output of the detection YOLOv3 model)***.

The Implementation:

a. First we download the data set of ***German Classification Benchmark*** and mount it to the ***Google drive*** (to make it ***faster and more secure).***
b. Get the Data From the Google Drive.

c. Include All libraries we need to make the model like:
   i. ***CV2:***
      This library used to manipulate the images or the frames of the video.
   ii. ***Pickle:***
      This library used to save the files of the model with weights and the CSV files.
   iii. ***OS:***
      This library used to manipulate the paths of the images or the frames of the video.
   iv. ***Keras and Tensorflow:***
      These libraries are used to build the model with different types of layers.
   v. ***Sklearn:***
      This library is used to train the model according to the error correction methods.
   vi. ***In addition, more libraries are used…***
d. Define the Hyper parameters we will use in our implementation as:
   i. ***Train and Test Directories:***
      The paths of the Test and Train data (images).

ii. ***Train .npy and Train label .npy:***
The Paths of the Train .npy file and the class of each image.

iii. ***Test .npy and Test label .npy:***
The Paths of the Test .npy file and the class of each image.

iv. ***Classes:***
List of all Numbers of the 43 Classes Numbered from 0 to 42.

v. ***IMG_SIZE:***
The size of the image we used for the preprocessing.

vi. ***BATCH_SIZE:***
The batch size we use to build the model.

vii. ***Normalize Flag:***
This flag to indicate If we will normalize the data or not.

viii. ***The classifier flag:*** this indicates which classification model we will use form those (VGG, inception model or hyped inception)
Note we use the hypered inception model with api in the last accuracy of 99.2%

e. Now we will do the ***image processing*** operations to the images according to the following:

i. ***Create Label Function:***
This function to create the label (the class of the image) according to the name of the image as :
index = index(folder name in the classes list)
res = 1 if the idx is the same as index 0 else
return res

ii. ***Process Function:***
this function is used for processing the images or the frames of the video,
if ***Normalized Flag is True Then***
do histogram equalization using the hsv image scale as

this:
hsv = Convert rgb to hsv(image)
hsv_image = histograme_qualization(hsv)
img = Convert hsv to rgb(hsv_image)
After this we pad the image to make them all the same size

iii. ***Make the data:***
loop on each npy file and use the Create label function to make the lists of: train_data, train_label_data, test_data, test_label_data.

f. We use the plot function to plot the ***validation data, training data according to the number of epochs.***
We also plot the accuracy of the test and validation data with time and the number of epochs.

g. Then we build the Classifier model:
layer_1 = Conv2D(32, (3,3), padding='same', activation='relu', kernel_initializer='he_normal')***(input_img)***

layer_1_b = BatchNormalization(epsilon=1e-06, axis=3) ***(layer_1)***

layer_2 = Conv2D(48, (7,1), with the same padding, actibation function is the relu ,kernel_initializer='he_normal') ***(add to layer_1_b)***

layer_2_b = BatchNormalization(epsilon=1e-06, axis=3) ***(layer_2)***

layer_3 = Conv2D(48, (1,7), with the same padding, actibation function is the relu ,kernel_initializer='he_normal') ***(add to layer_2_b)***

layer_3_b = BatchNormalization(epsilon=1e-06, axis=3) ***(layer_3)***

max_1 = MaxPool2D(pool_size=(2,2))***(layer_3_b)***

drop_1 = Dropout(rate=0.2) ***(max_1)***

inc_1 = Conv2D(64, (3,1), padding='same', activation='relu' ,kernel_initializer='he_normal') ***(drop_1)***

inc_1 = BatchNormalization(epsilon=1e-06, axis=3) ***(inc_1)***

inc_1 = Conv2D(64, (1,3), padding='same', activation='relu' ,kernel_initializer='he_normal') ***(inc_1)***

inc_1 = BatchNormalization(epsilon=1e-06, axis=3) ***(inc_1)***

inc_2 = Conv2D(64, (1,7), padding='same', activation='relu' ,kernel_initializer='he_normal') ***(drop_1)***

inc_2 = BatchNormalization(epsilon=1e-06, axis=3) ***(inc_2)***

inc_2 = Conv2D(64, (7,1), padding='same', activation='relu' ,kernel_initializer='he_normal') ***(inc_2)***

inc_2 = BatchNormalization(epsilon=1e-06, axis=3) ***(inc_2)***

then merge the two layers of inc1 and inc2
merge_layer = concatenate([inc_1, inc_2])

max_2 = MaxPool2D(pool_size=(2,2)) ***(merge_layer)***
drop_2 = Dropout(rate=0.2) ***(max_2)***

layer_4 = Conv2D(128, (3,3), padding='same', activation='relu' ,kernel_initializer='he_normal') ***(drop_2)***

layer_4_b = BatchNormalization(epsilon=1e-06, axis=3) ***(layer_4)***

layer_5 = Conv2D(256, (3,3), padding='same', activation='relu' ,kernel_initializer='he_normal') ***(layer_4_b)***

layer_5_b = BatchNormalization(epsilon=1e-06, axis=3) ***(layer_5)***

max_3 = MaxPool2D(pool_size=(2,2)) ***(layer_5_b)***
drop_3 = Dropout(rate=0.3) ***(max_3)***

After This make the flatten layer:
flat_1 = Flatten() ***(drop_3)***

dense_1 = Dense(256, activation='relu', kernel_initializer='he_normal') ***(flat_1)***

ldense_1_b = BatchNormalization() ***(dense_1)***

drop_4 = Dropout(rate=0.4) ***(ldense_1_b)***

output = Dense(43, activation='softmax', kernel_initializer='he_normal') ***(drop_4)***

return model

h. In the Main Section, we only train the model with the specified number of epochs

first; we load the train and test data.

Then, we load the model and start train it.

In the data we use the data Generator to increase the number of data set images by using some operations like:

i. ***Zoom***
ii. ***Shear***
iii. ***Rotation***
iv. ***Width Shift***
v. ***Height Shift***
vi. ***Feature wise center***

We train the model without data augmentation for 200 epochs and then with 300 epochs with data augmentation and the plots show the loss and the accuracy along the epochs

Without Data Augmentation:

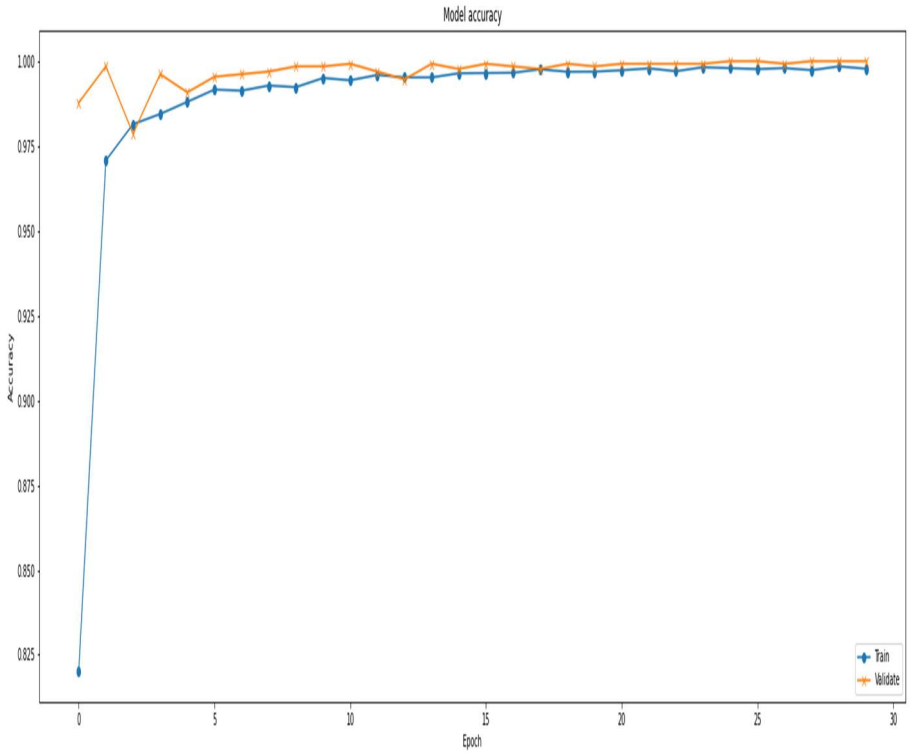


Figure 4.1

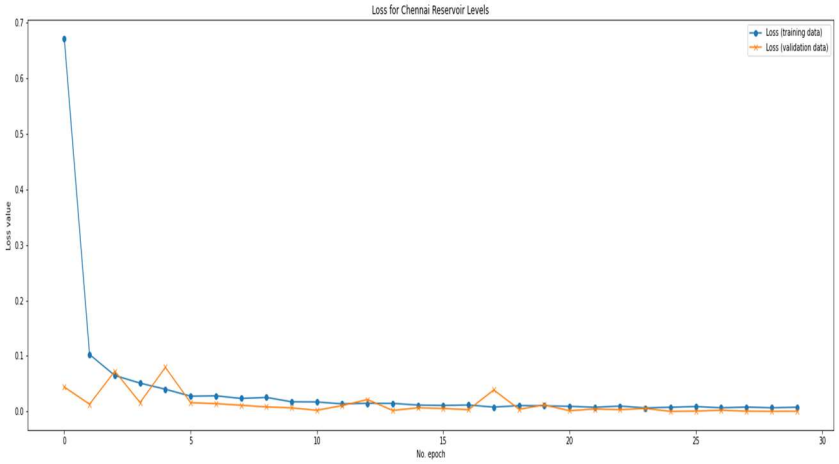


Figure 4.2

With Data Augmentation:

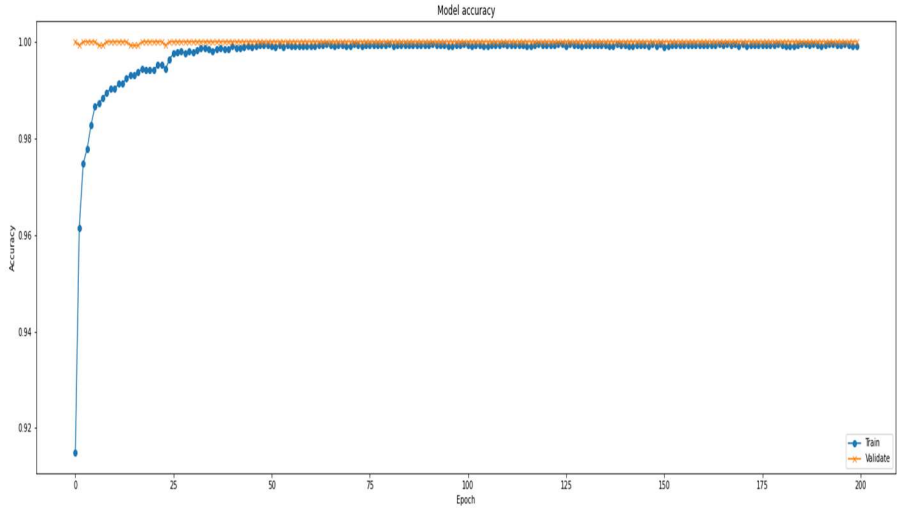


Figure 4.3

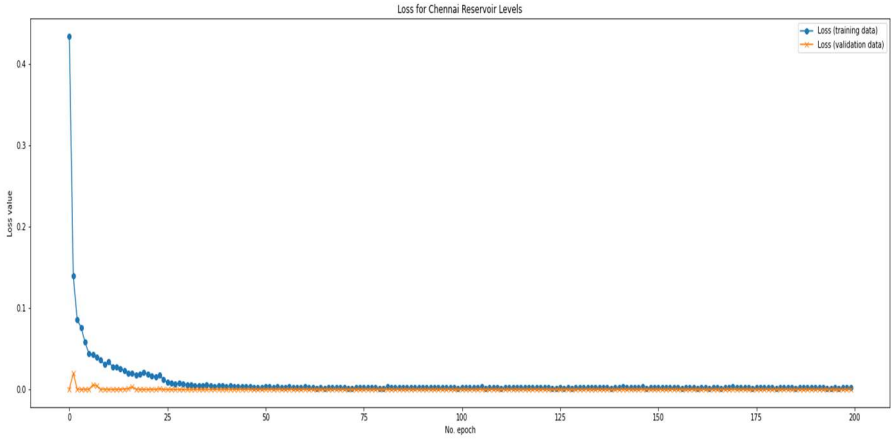


Figure 4.4

After Training the model and we see the validation error according to the number of epochs,

we save the model file use the pickle as:
pk.dump("model.h5")

and then we plot the diagrams of the error and loss with the number of epochs.

if the flag test is true then we take each box form the image and enter It to the model to classify

2. Detection Model:

This model is using the ***YOLOv3*** to detect the traffic signs in the input image and then, make ***a boundary box surrounding the sign***.

The Implementation:

a. First we include libraries we will need in the code like:
   - ***i. CV2:***
     This library used to manipulate the images or the frames of the video.
   - ***ii. Pickle:***
     This library used to save the files of the model with weights and the CSV files.
   - ***iii. OS:***
     This library used to manipulate the paths of the images or the frames of the video.
   - ***iv. Keras and Tensorflow:***
     These libraries are used to build the model with different types of layers.
   - ***v. Sklearn:***
     This library is used to train the model according to the error correction methods.
   - ***vi. Tqdm:*** this library to loop on the features of the training file.npy.
   - ***vii. In addition, more libraries are used…***

b. Use the Function ***get_sing_descriptor*** to make the ***labels*** of all the sign classes ***(43 Class) from 0 to 42*** as:
   0 is Maximum Speed Limit 20 MPH.
   1 is Maximum Speed Limit 30 MPH.
   27 is Warning for pedestrians.
   And so on.

c. Then get the data from the directories by mounting the ***Google drive to Colab*** and then specify the ***directories of the training and testing files***.

d. Define the hyper parameters we will need in the code like:
   i. ***Modelh5:*** The path to the pre-trained model with the data augmentation.
   ii. ***Weights_YOLO:*** The path to the weights of the YOLO model we train.
   iii. ***IMG_SIZE:*** The size of the image or frames of the video we will use to train the model.

e. After that for each frame of the video, we use the model weights and the pre-trained model in detecting the traffic signs we want from the image as:
   net = cv2.dnn.readNet(weightsYOLO,CFG)

f. After that first we loop on the outputs and inside this loop we loop on the detected objects as:
   for out in outputs_YOLO:
   for detection in out:

g. We use the scores of the detection list ***from 5 to the end of the list (as the 5 parameters of the box x, y ,w, h and the probability)*** and get the class id index as binary code from the scores as:
   class_idx = max_number(scores)

h. After this, we take the ***confidence*** that this box contain object we want and if the ***confidence is more than 0.2 that means the box contains a traffic sign***.

i. In case the confidence is more than 0.2, we get the center of the x and y of the box as:
   center x = detection_list[0] * width of the frame

center y = detection_list[1] * height of the frame

j. Get the width and the height of the box as:
width_box = detection_list[2] * width of the frame
height_box = detection_list[3] * height of the frame

k. Rectangle the coordinates to make the box by defining the x and y coordinates as:
x = (center x – w) / 2 as integer
y = (center y – h) / 2 as integer

l. Append the class_id, confidence and the list of box parameters to their lists.

Figure 4.5

3. *Connection of Detection and Classifier Models:*

We here connect the output of the detection model as the input of the classifier model to make the original image with no cropping as classified traffic signs image.

The Implementation:

a. First we get the output list of the detection model form the ***YOLOv3*** model

b. The output is a list if the ***cropped images in the original image***

c. After we have the traffic signs, we enter them to the ***classifier model and get the id of each image***

d. We get the ***name*** of the class of each ***ID*** we get form the ***classifier model***

e. After that, we take the name of the class to place it as a ***text*** on the boundary box surrounding the image in the original form.

f. Get the indexes of the boxes by using the pre-trained model as:
indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.8, 0.3)

g. Start to preprocess the boxes by cutting the image form the original image and append the coordinates to the output list as:
frame2gray = convert frame to gray scale(frame)
width_per_frame = w * 0.25
height_per_frame = h * 0.25

h. Crop the image to get the sign and make a list with all cropped signs to train the classifier as:

i. ***crop_img = frame2gray[y - int(height_per / 2):y + h + int(height_per / 2, x - int(width_per / 2):x + w + int(width_per / 2)]***

Figure 4.6

Chapter 5: System Testing

The System Testing is a pipeline of operations as the following:

1. First we enter the original image in the full size to the first model the ***detection model (YOLOv3)***

2. In the detection model we use the pre-trained model to detect the traffic signs in the image using the weights of the YOLO as:
   net = cv2.dnn.readNet (weightsYOLO,CFG)

3. We capture the input video ***(Test video)*** and then, adjust the resolution of the video by converting the intensity of the pixels in the frames to integers as:
   Frame_Width = integer(video[3])
   Frame_Height = integer(video[4])
   Then, get the number of the frames per second as:
   Frames_Per_Second = inteher(video[5])
   and make the size of the frames as:
   Size = (Frame_Width, Frame_Height)

4. We will loop on each ***frame in the video we captured*** and in each one we will loop on the ***detected boxes*** and for each box we will loop on the ***detected objects*** as:
   for out in the outs:
   for detected in out:

5. We use the scores of the detection list ***from 5 to the end of the list (as the 5 parameters of the box x, y ,w, h and the probability)*** and get the class id index as binary code from the scores as:
   class_idx = max_number(scores)

6. After this, we take the ***confidence*** that this box contain object we want and if the ***confidence is more than 0.2 that means the box contains a traffic sign***.

7. In case the confidence is more than 0.2, we get the center of the x and y of the box as:
   center x = detection_list[0] * width of the frame
   center y = detection_list[1] * height of the frame

8. Get the width and the height of the box as:
   width_box = detection_list[2] * width of the frame
   height_box = detection_list[3] * height of the frame

9. Rectangle the coordinates to make the box by defining the x and y coordinates as:
   x = (center x – w) / 2 as integer
   y = (center y – h) / 2 as integer

10. Append the class_id, confidence and the list of box parameters to their lists.

11. Get the indexes of the boxes by using the pre-trained model as:
    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.8, 0.3)

12. Start to preprocess the boxes by cutting the image form the original image and append the coordinates to the output list as:
    frame2gray = convert frame to gray scale(frame)

width_per_frame = ***w * 0.25***
height_per_frame = ***h * 0.25***

13. Crop the image to get the sign and make a list with all cropped signs to train the classifier as: ***crop_img = frame2gray[y - int(height_per / 2):y + h + int(height_per / 2, x - int(width_per / 2):x + w + int(width_per / 2)]***

14. Now, we have list with all the cropped images as they are the traffic signs images, we will enter these images to the classifier model

15. After the prediction we will compare the actual testing label with the by using the argmax function in the numpy library as:
Prediction = predict(features of image)
Predicted_class = argmax_function(prediction)
if Predicted_class is the output actual class Then increament the accepted results by 1 to calculate the final accuracy.

16. ***The final accuracy is 99.2%.***

Figure 5.1

Figure 5.2

Our testing system pass by two phases

Phase one:

Connecting YOLO with the classifier is done by using OpenCV (1-sec video take almost 120 sec), approximate FPS was 0.64, Pretty bad but the problem was in OpenCV library itself because it didn't use Colab GPU so we changed the version of the library with (4.4.0) dev according to **"Led , "by dlib's Davis King, and implemented by Yashas Samaga** OpenCV 4.2 now supports NVIDIA GPUs for inference using OpenCV's dnn module, improving inference speed by up to 1549%. When we tried ,)1-sec video take 67 sec(to test the speed improved by 200% as approximate FPS was 0.83

Phase two:

We tried to train yolo on 43 classes but we achieved 65% mAP, So we tried to train only on the categories of the classes and there are 4 classes and we achieved 93% mAP

Note: our classifier accuracy is 99.2 and the paper classifier is 99.5

Our detector got 93, paper detector was 92.2%

Chapter 6: Conclusion and Future Work

Here we will discuss the conclusion form the system we build and the future work we insist to do.

1-Conclusion:

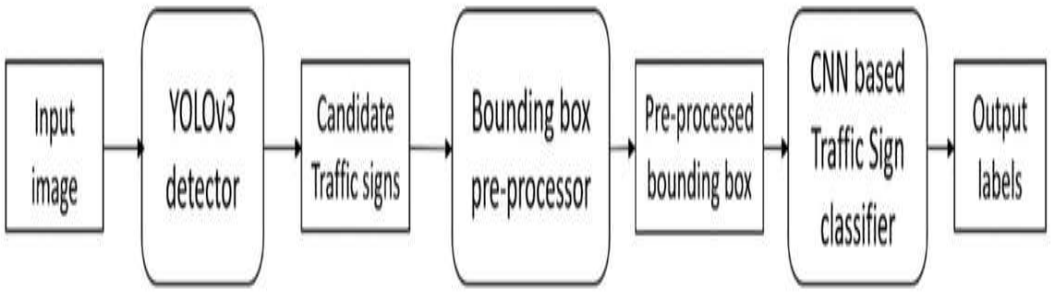


Figure 6.1

Our System Pipeline Approach

- A traffic sign recognition system based on ***YOLOv3*** based detector and a custom ***CNN*** based classifier is proposed.
- The detection performance, with traffic-sign being considered as a single class, is found to be superior to the previous detectors, based on the detection speed.
- ***It could achieve a speed of almost 10 frames/second with a high Map performance of 92.2%.***
- The used detector is able to ***detect almost all categories*** of traffic signs and could regress ***accurate bounding boxes*** for most of the detected signs.
- ***A few very small traffic signs could not be accurately localized and there were a few false positive detections.***
- The traffic sign classifier used is simple in architecture with very high accuracy.
- The ***YOLOv3*** based detector and ***CNN*** based classifier completes the ***traffic sign recognition pipeline***.
- If we manage to make more images in the data set specially for the detection of the traffic signs we will manage to improve the total accuracy.

2-Future Work:

- As a future step, simultaneous detection and classification of traffic signs using single stage detectors could be explored.
- This method could help avoiding the use of an additional traffic sign classification network.
- Fine tuning of networks to learn the intricacies of similar looking signs is required in this case and it would be challenging to tune the network to give an accuracy performance exhibited by the detector-classifier based recognition pipeline.
- But it would be worth exploring this aspect in future.
- Also a future work we can put a H/W in our project by putting a camera for experiment in a self-driving car and test the accuracy and how the car control itself in the different situations according to the traffic sign.

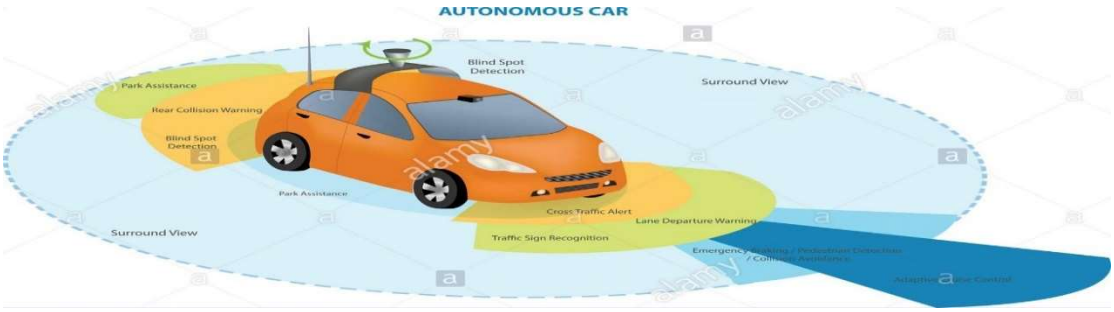


Figure 6.2

Tools

Here we will discuss the tools we used in building our system and the advantages of each one

1. *Python Programming Language:*

Advantages of this language are:

- Dynamically Typed.
- Has Garbage Collector.
- Provide ***Tenser Flow*** libraries, which are important in building deep learning models.
- Easy to write and high-level language.
- Use concepts in writing the code like ***Flat is better than nested.***

2. *Google Colab:*

Advantages of using this Online IDE are:

- Provide about 22GB Ram and about 64GB of Disk.
- Use the run of type GPU is fast.
- We can put the ***source code, textual explanation and the output of each section alone.***
  This make the code for the human respective not the machine respective.

References

1. Gudigar A, Chokkadi S, Raghavendra U (2016) A review on automatic detection and recognition of traffic sign. Multimed Tools Appl 75(1):333–364

2. Fu MY, Huang YS (2010) A survey of traffic sign recognition. In: International conference on wavelet analysis and pattern recognition (ICWAPR), 2010, IEEE, pp 119–124

3. Wali SB, Hannan MA, Hussain A, Samad SA (2015) Comparative survey on traffic sign detection and recognition: a review. Przegld Elektrotechniczny, ISSN: 0033-2097

4. Escalera S, Bar X, Pujol O, Vitri J, Radeva P (2011) Background on traffic sign detection and recognition. In: Traffic-sign recognition systems. Springer, London, pp 5–13

5. De La Escalera A, Moreno LE, Salichs MA, Armingol JM (1997) Road traffic sign detection and classification. IEEE Trans Indus Electron 44(6):848–859

6. Benallal M, Meunier J (2003) Real-time color segmentation of road signs. In: Canadian conference on electrical and computer engineering, vol 3, 2003, IEEE CCECE 2003, IEEE, pp 1823–1826

7. Ruta A, Li Y, Liu X (2010) Real-time traffic sign recognition from video by class-specific discriminative features. Pattern Recogn 43(1):416–430

8. Ruta A, Porikli F,Watanabe S, Li Y (2011) In-vehicle camera traffic sign detection and recognition. Mach Vis Appl 22(2):359–375

9. Lim KH, Ang LM, Seng KP (2009) New hybrid technique for traffic sign recognition. In: International symposium on intelligent signal processing and communications systems, 2008. ISPACS 2008, IEEE, pp 1–4

10. Itti L, Koch C, Niebur E (1998) A model of saliency-based visual attention for rapid scene analysis. IEEE Trans Pattern Anal Mach Intell 20(11):1254–1259

11. Behloul A, Saadna Y (2014) A Fast and Robust Traffic Sign Recognition. Int J Innov Appl Stud 5(2):139

12. Yakimov P (2015) Traffic signs detection using tracking with prediction. In: International conference on E-business and telecommunications Colmar, France, Springer, pp 454–467

13. Yakimov PY (2015) Preprocessing digital images for quickly and reliably detecting road signs. Pattern Recogn Image Anal 25(4):729–732

14. Wang G, Ren G, Jiang L, Quan T (2014) Hole-based traffic sign detection method for traffic signs with red rim. Vis Comput 30(5):539–551

15. Mogelmose A, Trivedi MM, Moeslund TB (2012) Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey. IEEE Trans Intell Transp Syst 13(4):1484–1497

16. Maldonado-Bascon S, Lafuente-Arroyo S, Gil-Jimenez P, GomezMoreno H, Lpez-Ferreras F (2007) Road-sign detection and recognition based on support vector machines. IEEE Trans Intell Transp Syst 8(2):264–278

17. Fleyeh H (2006) Shadow and highlight invariant color segmentation algorithm for traffic signs. In: IEEE conference on cybernetics and intelligent systems, 2006, IEEE, pp 1–7

18. Vitabile S, Pollaccia G, Pilato G, Sorbello F (2001) Road signs recognition using a dynamic pixel aggregation technique in the HSV color space. In: Proceedings of the 11th international conference on image analysis and processing, 2001, IEEE, pp 572–577

19. De la Escalera A, Armingol JM, Mata M (2003) Traffic sign recognition and analysis for intelligent vehicles. Image Vis Comput 21(3):247–258

20. Fang CY, Fuh CS, Yen PS, Cherng S, Chen SW (2004) An automatic road sign recognition system based on a computational model of human recognition processing. Comput Vis Image Underst 96(2):237–268

21. Miura J, Kanda T, Nakatani S, Shirai Y (2002) An active vision system for on-line traffic sign recognition. IEICE TRANSACTIONS on Inf Syst 85(11):1784–1792

22. Broggi A, Cerri P, Medici P, Porta PP, Ghisio G (2007) Real time road signs recognition. In: IEEE intelligent vehicles symposium, 2007, IEEE, pp 981–986

23. Gmez-Moreno H, Maldonado-Bascn S, Gil-Jimnez P, LafuenteArroyo S (2010) Goal evaluation of segmentation algorithms for traffic sign recognition. IEEE Trans Intell Transp Syst 11(4):917–930

24. Liu C, Chang F, Chen Z, Liu D (2016) Fast traffic sign recognition via high-contrast region extraction and extended sparse representation. IEEE Trans Intell Transp Syst 17(1):79–92

25. Hechri A, Hmida R, Mtibaa A (2015) Robust road lanes and traffic signs recognition for driver assistance system. Int J Comput Sci Eng 10(1–2):202–209

26. Garcia-Garrido MA, Sotelo MA, Martin-Gorostiza E (2006) Fast traffic sign detection and recognition under changing lighting conditions. In: IEEE intelligent transportation systems conference, 2006 ITSC'06, IEEE, pp. 811–816

27. Barnes N, Zelinsky A (2004) Real-time radial symmetry for speed sign detection. In: IEEE intelligent vehicles symposium, 2004, IEEE, pp 566–571

28. Loy G, Barnes N (2004) Fast shape-based road sign detection for a driver assistance system. In: Proceedings of the 2004 IEEE/RSJ international conference on intelligent robots and systems, 2004 (IROS 2004) Vol 1, IEEE, pp 70-75

29. Soheilian B, Arlicot A, Paparoditis N (2010) Extraction de panneaux de signalisation routire dans des images couleurs. In: Reconnaissance des Formes et Intelligence Artificielle, pp 1–8

30. Zhang SC, Liu ZQ (2005) A robust, real-time ellipse detector. Pattern Recogn 38(2):273–287

31. Larsson F, Felsberg M (2011) Using Fourier descriptors and spatial models for traffic sign recognition. In: Scandinavian conference on image analysis, Springer, Berlin, Heidelberg, pp 238–249

32. Qin F, Fang B, Zhao H (2010) Traffic sign segmentation and recognition in scene images. In: Chinese conference on pattern recognition (CCPR), pp 1–5

33. Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition, CVPR 2001, vol 1, IEEE, p I

34. Brkic K, Pinz A, Segvic S (2009) Traffic sign detection as a compo- ˇ nent of an automated traffic infrastructure inventory system. Stainz, Austria

35. Brki K , Segvi S, Kalafati Z, Sikiri I, Pinz A (2010) Genera- ˇ tive modeling of spatio-temporal traffic sign trajectories. In: IEEE computer society conference on computer vision and pattern recognition workshops (CVPRW), 2010, IEEE, pp 25–31

36. Brkic K (2010) An overview of traffic sign detection methods. Department of Electronics, Microelectronics, Computer and Intelligent Systems Faculty of Electrical Engineering and Computing Unska, 3, 10000

37. Bar X, Escalera S, Vitri J, Pujol O, Radeva P (2009) Traffic sign recognition using evolutionary adaboost detection and forestECOC classification. IEEE Trans Intell Transp Syst 10(1):113–126

38. Prisacariu VA, Timofte R, Zimmermann K, Reid I, Van Gool L (2010) Integrating object detection with 3d tracking towards a better driver assistance system. In: 20th International conference on pattern recognition (ICPR), 2010, IEEE, pp 3344–3347

39. Fang CY, Chen SW, Fuh CS (2003) Road-sign detection and tracking. IEEE Trans Veh Technol 52(5):1329–1341

40. Zaklouta F, Stanciulescu B (2011) Warning traffic sign recognition using a HOG-based Kd tree. In: IEEE intelligent vehicles symposium (IV), 2011. IEEE, pp 1019–1024

41. Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: IEEE computer society conference on computer vision and pattern recognition, CVPR 2005, vol 1, IEEE pp 886–893

42. Wang G, Ren G, Wu Z, Zhao Y, Jiang L (2013) A robust, coarse-tofine traffic sign detection method. In: The 2013 international joint conference on neural networks (IJCNN), IEEE, pp 1–5

43. Houben S, Stallkamp J, Salmen J, Schlipsing M, Igel C (2013) Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In: The 2013 international joint conference on neural networks (IJCNN), IEEE, pp 1–8

44. Houben S (2011) A single target voting scheme for traffic sign detection. In: IEEE intelligent vehicles symposium (IV), 2011, IEEE, pp 124–129

45. Mathias M, Timofte R, Benenson R, Van Gool L (2013) Traffic sign recognitionHow far are we from the solution?. In: The 2013 international joint conference on neural networks (IJCNN), IEEE, pp 1–8

46. Wu Y, Liu Y, Li J, Liu H, Hu X (2013) Traffic sign detection based on convolutional neural networks. In: The 2013 international joint conference on neural networks (IJCNN), pp 1–7, IEEE

47. Liu C, Chang F, Chen Z, Li S (2013) Rapid traffic sign detection and classification using categories-first-assigned tree. J Comput Inf Syst 9(18):7461–7468

48. Liu C, Chang F, Chen Z (2014) Rapid multiclass traffic sign detection in high-resolution images. IEEE Trans Intell Transp Syst 15(6):2394–2403

49. Timofte R, Zimmermann K, Van Gool L (2009) Multi-view traffic sign detection, recognition, and 3d localisation. In: Workshop on applications of computer vision (WACV), 2009, IEEE, pp 1–8

50. Mogelmose A, Trivedi MM, Moeslund TB (2012) Learning to detect traffic signs: comparative evaluation of synthetic and real-world datasets. In: 21st International conference on pattern recognition (ICPR), 2012, IEEE, pp 3452–3455

51. Segvic S, Brki K, Kalafati Z, Stanisavljevi V, evrovi M, Budimir D, ˇ Dadi I (2010) A computer vision assisted geoinformation inventory for traffic infrastructure. In: 13th International IEEE conference on intelligent transportation systems (ITSC), 2010, IEEE, pp 66–73

52. Zaklouta F, Stanciulescu, B., Hamdoun, O. (2011) Traffic sign classification using kd trees and random forests. In: The 2011 international joint conference on neural networks (IJCNN), pp 2151–2155, IEEE

53. Stallkamp J, Schlipsing M, Salmen J, Igel C (2012) Man vs. computer: benchmarking machine learning algorithms for traffic sign recognition. Neural networks 32:323–332

54. Sermanet P, LeCun Y (2011) Traffic sign recognition with multiscale convolutional networks. In The 2011 international joint conference on neural networks (IJCNN), IEEE, pp 2809–2813

55. Cirean D, Meier U, Masci J, Schmidhuber J (2011) A committee of neural networks for traffic sign classification. In: The 2011 international joint conference on neural networks (IJCNN), IEEE, pp 1918–1921

56. CireAn D, Meier U, Masci J, Schmidhuber J (2012) Multi-column deep neural network for traffic sign classification. Neural Netw 32:333–338

57. Zeng Y, Xu X, Fang Y, Zhao K (2015) Traffic sign recognition using extreme learning classifier with deep convolutional features. In: The 2015 international conference on intelligence science and big data engineering (IScIDE 2015), Suzhou, China

58. Aghdam HH, Heravi EJ, Puig D (2017) A practical and highly optimized convolutional neural network for classifying traffic signs in real-time. Int J Comput Vis 122(2):246–269

59. Ciregan D, Meier U, Schmidhuber J (2012) Multi-column deep neural networks for image classification. In IEEE conference on computer vision and pattern recognition (CVPR), 2012, IEEE, pp 3642–3649

60. Jin J, Fu K, Zhang C (2014) Traffic sign recognition with hinge loss trained convolutional neural networks. IEEE Trans Intell Transp Syst 15(5):1991–2000

61. Yakimov PY (2016) Real-time road signs recognition using mobile GPU. In: CEUR workshop proceedings, vol 1638, pp 477–483

62. Qu Y, Yang S, Wu W, Lin L (2016) Hierarchical traffic sign recognition. In: Pacific rim conference on multimedia, Springer, pp 200–209

63. Abedin MZ, Dhar P, Deb K (2016) Traffic Sign Recognition using SURF: speeded up robust feature descriptor and artificial neural network classifier. In: 9th International conference on electrical and computer engineering (ICECE), 2016, IEEE, pp 198–201

64. Han Y, Virupakshappa K, Oruklu E (2015) Robust traffic sign recognition with feature extraction and k-NN classification methods. In: IEEE international conference on electro/information technology (EIT), 2015, IEEE, pp 484–488

65. Malik Z, Siddiqi I (2014) Detection and recognition of traffic signs from road scene images. In: 12th International conference on frontiers of information technology (FIT), 2014, IEEE, pp 330–335

66. Sathish P, Bharathi D (2016) Automatic road sign detection and recognition based on SIFT feature matching algorithm. In: Proceedings of the international conference on soft computing systems. Springer India, pp 421–431

67. Hua X, Zhua X, Lia D, Li H (2010) Traffic sign recognition using Scale invariant feature transform and SVM. In: A special joint symposium of ISPRS technical commission IV and AutoCarto in conjunction with ASPRS/CaGIS fall specialty conference November, pp 15–19

68. Lasota M, Skoczylas M (2016) Recognition of multiple traffic signs using keypoints feature detectors. In: International conference and exposition on electrical and power engineering (EPE), 2016, IEEE, pp 535–540

69. Chen L, Li Q, Li M, Mao Q (2011) Traffic sign detection and recognition for intelligent vehicle. In: IEEE intelligent vehicles symposium (IV), 2011, IEEE, pp 908–913

70. Hoferlin B, Zimmermann K (2009) Towards reliable traffic sign recognition. In: IEEE Intelligent vehicles symposium, 2009, IEEE, pp 324–329

71. Liu H, Liu Y, Sun F (2014) Traffic sign recognition using group sparse coding. Inf Sci 266:75–89

72. He X, Dai B (2016) A new traffic signs classification approach based on local and global features extraction. In: International conference on information communication and management (ICICM), IEEE, pp 121–125

73. Tang S, Huang LL (2013) Traffic sign recognition using complementary features. In: 2nd IAPR Asian conference on pattern recognition (ACPR), 2013, IEEE, pp 210–214

74. Li C, Yang C (2016) The research on traffic sign recognition based on deep learning. In: 16th International symposium on communications and information technologies (ISCIT), 2016, IEEE, pp 156–161

75. Akinlar C, Topal C (2013) EDCircles: a real-time circle detector with a false detection control. Pattern Recogn 46(3):725–740

76. Berkaya SK, Gunduz H, Ozsen O, Akinlar C, Gunal S (2016) On circular traffic sign detection and recognition. Expert Syst Appl 48:67–75

77. Aghdam HH, Heravi EJ, Puig D (2016) A practical approach for detection and classification of traffic signs using Convolutional Neural Networks. Robot Auton Syst 84:97–112

78. Aghdam HH, Heravi EJ, Puig D (2016) Recognizing traffic signs using a practical deep neural network. In: Robot 2015: 2nd Iberian robotics conference, Springer, pp 399–410

79. Maas AL, Hannun AY, Ng AY (2013) Rectifier nonlinearities improve neural network acoustic models. In: Proceedings of the ICML, vol 30, No 1

80. Eickeler S, Valdenegro M, Werner T, Kieninger M (2016) Future computer vision algorithms for traffic sign recognition systems. In: Advanced microsystems for automotive applications 2015, Springer, pp 69–77

81. Youssef A, Albani D, Nardi D, Bloisi DD (2016) Fast traffic sign recognition using color segmentation and deep convolutional networks. In: International conference on advanced concepts for intelligent vision systems, Springer, pp 205–216

82. Huang Z, Yu Y, Gu J, Liu H (2016) An efficient method for traffic sign recognition based on extreme learning machine. IEEE Trans Cybern 47(4):920–933

83. Zang D, Zhang J, Zhang D, Bao M, Cheng J, Tang K (2016) Traffic sign detection based on cascaded convolutional neural networks. In: 17th IEEE/ACIS international conference on software engineering, artificial intelligence, networking and parallel/distributed computing (SNPD), 2016, IEEE, pp 201–206

84. Salti S, Petrelli A, Tombari F, Fioraio N, Di Stefano L (2015) Traffic sign detection via interest region extraction. Pattern Recogn 48(4):1039–1049

85. Chen T, Lu S (2016) Accurate and efficient traffic sign detection using discriminative adaboost and support vector regression. IEEE Trans Veh Technol 65(6):4006–4015

86. Ellahyani A, El Ansari M, El Jaafari I (2016) Traffic sign detection and recognition based on random forests. Appl Soft Comput 46:805–815

87. Qian R, Yue Y, Coenen F, Zhang B (2016) Traffic sign recognition with convolutional neural network based on max pooling positions. In: 12th International conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD), 2016, IEEE, pp 578–582

88. Xie K, Ge S, Ye Q, Luo Z (2016) Traffic sign recognition based on attribute-refinement cascaded convolutional neural networks. In: Pacific rim conference on multimedia, Springer, pp 201–210

89. Aghdam HH, Heravi EJ, Puig D (2015) Traffic sign recognition using visual attributes and Bayesian network. In: International joint conference on computer vision, imaging and computer graphics, Springer, pp 295–315

90. Jia Li and Zengfu Wang, "Real-Time Traffic Sign Recognition Based on Efficient CNNs in the Wild", IEEE Trans. Intell. Transp. Syst., vol. 20, no. 3, pp. 975–984, Mar. 2019.

91. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE Conf. Comput.Vis. Pattern Recognit., pp. 779–788, Jun. 2016

92. W. Liu et al., "SSD: Single shot multibox detector," in Proc. Eur. Conf Comput. Vis. Cham, Switzerland: Springer, 2016.

93. Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie, "Feature Pyramid Networks for Object Detection ", [Online] Available: https://arxiv.org/abs/1612.03144, 2016

94. S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German traffic sign detection benchmark," in Proc. IEEE Int. Joint Conf. Neural Netw., pp. 1–8, Aug. 2013.

95. D. Kingma and J. Ba. (2014). "Adam: A method for stochastic optimization." [Online]. Available: https://arxiv.org/abs/1412.6980

96. Joseph Redmon, Ali Farhadi, "YOLO9000: Better, Faster, Stronger", [Online] Available: https://arxiv.org/abs/1612.08242, 2016

97. Joseph Redmon, Ali Farhadi, "YOLOv3: An Incremental Improvement", [Online] Available: https://arxiv.org/abs/1804.02767, 2018

98. Y. Yang, H. Luo, H. Xu, and F. Wu, "Towards real-time traffic sign detection and classification," IEEE Trans. Intell. Transp. Syst., vol. 17, no. 7, pp. 2022–2031, Jul. 2016.

99. Y. Zhu, C. Zhang, D. Zhou, X. Wang, X. Bai, and W. Liu, "Traffic sign detection and recognition using fully convolutional network guided proposals," Neurocomputing, vol. 214, pp. 758–766, Nov. 2016.

100. Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in Proc. IEEE Conf. Comput.Vis. Pattern Recognit., pp. 2110–2118, Jun. 2016.

101. E. Peng, F. Chen, and X. Song, "Traffic sign detection with convolutional neural networks," in Proc. Int. Conf. Cogn. Syst. Signal Process., Singapore: Springer, pp. 214–224, 2016.

102. R. Girshick, "Fast R-CNN," in Proc. IEEE Int. Conf. Comput. Vis., pp. 1440–1448, Dec. 2015.

103. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real- time object detection with region proposal networks," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

104. A. G. Howard et al. (2017). "MobileNets: Efficient convolutional neural networks for mobile vision applications." [Online]. Available: https://arxiv.org/abs/1704.04861

105. D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 3642–3649, Jun. 2012.

106. P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks," in Proc. IEEE Int. Joint Conf. Neural Netw., pp. 2809–2813, Aug. 2011.

105. T. L. Yuan. GTSRB_Keras_STN. Accessed: Nov. 1, 2017 [Online] Available: https://github.com/hello2all/GTSRB_Keras_STN

106. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition", [Online] Available: https://arxiv.org/abs/1512.03385

107. K. Audhkhasi, O. Osoba, and B. Kosko, "Noise-enhanced convolutional neural networks", Neural Networks 2016, Vol.78, pp. 15 – 23

Thank You……..