TRAFFIC SIGN RECOGNITION USING DEEP LEARNING

Hai Binh Dang,
Thanh Nam Vo,
Huu Duc Nguyen

Supervisors: Dr. Duc Ngoc Minh Dang
Dr. Cuong Tuan Nguyen

Department of ITS
FPT University

This capstone project submitted in partial fulfillment of the requirement for the *Degree of Bachelor of Artificial Intelligent in Computer Science*

FPT University February 2024

Declaration

We hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is our own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Hai Binh Dang,
Thanh Nam Vo,
Huu Duc Nguyen
February 2024

Acknowledgements

First of all, we would like to thank our two mentors, Mr. Duc and Mr. Cuong for helping us make this thesis while giving much useful information about the ways of researching articles, checking the code of the model, etc. With their help, we are able to make this thesis as detailed as possible, and also teach us more about deep learning and image detection, and help us to achieve our bachelor's degrees.

Personally, I would also like to show my appreciation to my family, and my friends, as they have always motivated and helped me whenever I had trouble and helped me push through this hardship of a time. In addition, I am very proud of our team, as everyone has done such a great job, and all deserve high praise.

Dang Hai Binh
Ho Chi Minh City
December 2023

Abstract

Recognizing the crucial role of traffic signs in ensuring road safety, their diverse shapes, colors, and symbols serve as vital cues for drivers. Overlooking or missing these signs, especially the critical ones, can lead to legal violations or catastrophic accidents, profoundly impacting various aspects. Utilizing computers for traffic sign recognition becomes imperative to enhance commuter safety. This not only plays a pivotal role in advancing technology but also holds significant value in academic research. Given the challenges posed by complex terrains and varying weather conditions, Traffic Sign Recognition (TSR) remains a high-demand problem.

This thesis aims to address the complexities of TSR by proposing a solution based on deep learning, specifically leveraging convolutional neural networks (CNN). CNNs, known for their effectiveness in image capture and processing, have demonstrated success in similar problem domains. The primary objective is to enhance the accuracy and speed of TSR. The key contributions of this thesis are:

1. **Model Improvement**: Building upon existing models employed for TSR, the thesis endeavors to enhance their performance. Through systematic improvements, the goal is to elevate the effectiveness of these models in addressing the complexities of traffic sign recognition, as also test out the ability to recognise data in a real-time situation.

2. **CNN Model for Traffic Sign Recognition in Vietnam**: Introducing a CNN model trained specifically for recognizing traffic signs in Vietnam. The thesis employs Faster R-CNN and YOLO for detection; LeNet, VGG16, VGG19 and ResNet50 for classification, conducting a comparative analysis that leads to the conclusion that YOLO and LeNet perform significantly better in real-time TSR scenarios.

By tackling these challenges and leveraging the capabilities of deep learning, particularly CNNs, the thesis aims to contribute to the advancement of TSR, fostering safer road environments through improved accuracy and speed in traffic sign recognition systems.

Contents

ACKNOWLEDGMENTS **v**

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**
1.1 Background and motivation 1
1.2 Intelligent Recognition and Detection Systems for Traffic Signs 3
1.3 Contributions 5
1.4 Objective of the thesis 6
1.5 Structure of the thesis 6

2 Literature Reviews **9**
2.1 Faster R-CNN Models 9
2.1.1 Traffic Sign Identification Using Deep Learning 9
2.1.2 Traffic Signs Detection and Recognition System using Deep Learning 10
2.1.3 Traffic sign detection method based on Faster R-CNN 11
2.2 YOLO Models 12
2.2.1 Improved YOLOv5 network for real-time multi-scale traffic sign detection 12
2.2.2 Traffic-Sign Recognition Using Deep Learning 13
2.2.3 Improved YOLO v5 with balanced feature pyramid and attention module for traffic sign detection 14
2.3 Other Models 15
2.3.1 Deep Learning for Large-Scale Traffic-Sign Detection and Recognition 15
2.3.2 Lightweight deep network for traffic sign classification 19
2.3.3 Real-time traffic sign recognition based on a general purpose GPU and deep-learning 19

2.3.4 Deep Learning Traffic Sign Detection, Recognition and Augmentation 21

3 Project Management Plan 23

4 Materials and Methods 27
4.1 Dataset . . . 27
4.1.1 German Traffic Sign Detection Benchmark (GTSDB) and German Traffic Sign Recognition Benchmark (GTSRB) . . . 27
4.1.2 Our collected data . . . 30
4.2 Detection Model . . . 30
4.2.1 Faster R-CNN Model . . . 31
4.3 YOLO Model . . . 33
4.3.1 Development of YOLO Family . . . 33
4.3.2 YOLOv8 for Traffic Sign Recognition . . . 35
4.4 Classification Model . . . 36
4.4.1 VGG 16 and 19 Model . . . 38
4.4.2 ResNet50 . . . 40
4.4.3 LeNet Model . . . 42
4.5 Our Methods . . . 43
4.6 Evaluation Methods . . . 44
4.6.1 IoU (Intersection over Union) . . . 44
4.6.2 Precision and Recall . . . 46
4.6.3 Precision Recall Curve and Average precision(AP) . . . 47

5 Result 49
5.1 Our Dataset . . . 49
5.2 Testing on models . . . 52
5.3 Project's Model Testing . . . 54

6 Discussion 57

7 Conclusion 59

Bibliography 61

List of Figures

1.1 Traffic Sign in Road 2

2.1 Each category AP results from statistics in the experiment 12
2.2 Size distribution of sign instances from the TT100K 13
2.3 Details Result on STSD 17
2.4 Evaluation on STSD 17
2.5 Results on DFG Traffic-Sign data set 18
2.6 Details Result on DFG with Difference Sizes 18

4.1 GSTDB and GSTDB's classes 27
4.2 GTSDB Dataset Classes Distribution 28
4.3 GTSRB Train Classes Distribution 29
4.4 GTSRB Test Classes Distribution 29
4.5 Object Detection Workflow 31
4.6 Process Flow of Faster R-CNN 32
4.7 Region Proposal Network 33
4.8 Comparison between each version of YOLOv5 34
4.9 Analysis of YOLO 35
4.10 YOLOv8 Framework 36
4.11 The Convolutional Layer 37
4.12 The Pooling layer 38
4.13 The VGG-16 architecture 40
4.14 ResNet50 shortcut connections 41
4.15 ResNet50 architecture 42
4.16 The LeNet architecture 43
4.17 Our Processing Method for TSR 44
4.18 Example of predicted bounding box with ground-truth bounding box 45
4.19 Formula of IoU 45

4.20 Precision Recall Curve 47
4.21 Average precision 47
4.22 mean Average precision 48

5.1 Dataset's classes 49
5.2 Example of a labeled image 50
5.3 Detection dataset's distribution 51
5.4 Classification dataset's train distribution 51
5.5 Classification dataset's test distribution 51
5.6 Some examples of our classification dataset images 52

List of Tables

2.1 Comparison with other method on the TT100K dataset 13
2.2 YOLOv5 Model Parameters 14
2.3 Comparison with the original method 15
2.4 Comparison with the others method 15
2.5 Driving Scenes 20

3.1 Project Schedule 23
3.2 Source Code and Data 25

5.1 Testing detection model on GSTDB 52
5.2 Testing on GTSDB to choose the best YOLO model version 53
5.3 YOLOv8 models' statistics 53
5.4 Testing classification model 54
5.5 Testing jointed model 54
5.6 Detection model's result 55
5.7 Classification model's result 55
5.8 YOLOv8n + Lenet model's result 55
5.9 Classification model result comparison 56
5.10 Jointed model result comparison 56

Chapter 1

Introduction

In the opening section, we introduce the motivation and contextual background that underpin our research project on Traffic Sign Recognition (TSR). Following this, in the subsequent section, we delve into the contemporary developments within TSR, offering a comprehensive overview of the current state of TSR technology and research. We also outline our research's objectives and provide a structural framework that guides the trajectory of our study, allowing readers to gain a holistic understanding of our work in the field of TSR.

1.1 Background and motivation

Cars have become an integral mode of mobility for people as the human population has grown and our societies have developed. In China, car sales increased 8.4% year on year to 2.582 million units in August 2023, resuming growth after a 1.4% decrease the previous month, aided by larger discounts and tax breaks for environmentally friendly and electric automobiles. Energy vehicle sales, in particular, increased by 27%. From January to August, new car sales increased by 8%, compared to a 1.7% gain in 2022. It is noteworthy that during this time, sales of energy vehicles increased significantly by 39.2%. The rise in car use inevitably leads to several issues with road traffic.

Even though there are traffic signs posted all over the route to guide us through the real scenario as Figure 1.1a, the growing number of vehicles complicates our traffic and makes us more perplexed. The proliferation of vehicles also causes a variety of societal issues, including the pollution caused by vehicle exhaust, which lowers air quality and endangers the ecosystem. Due to poor visibility caused by extreme weather, particularly haze, and fog as Figure 1.1b, it is simple for cars to miss crucial traffic warnings. Because of this, drivers will be more aware of breaking traffic laws, which might lead to accidents.

(a) Normal weather

(b) Foggy weather

Figure 1.1 Traffic Sign in Road

It has been observed that the frequency of traffic accidents has been increasing lately. According to the World Health Organization's (WHO) records, traffic accidents lead to the untimely demise of more than 1.3 million people annually. Furthermore, non-fatal injuries affect an additional 20 to 50 million individuals, who often develop disabilities due to the severity of their injuries.

It has been observed that the frequency of traffic accidents has been increasing lately. According to the World Health Organization's (WHO) records, traffic accidents lead to the untimely demise of more than 1.3 million people annually. Furthermore, non-fatal injuries affect an additional 20 to 50 million individuals, who often develop disabilities due to the severity of their injuries.

To combat the global surge in accidents, professionals and academics are investigating this issue [1]. Given that road transportation remains the primary mode for nations with vast geographic areas, research on traffic management systems like the Intelligent Traffic System (ITS) is becoming crucial for the future of road traffic. The foundation of TSR [2] lies in pattern recognition, image processing, computer vision, and related technologies. Its goal is to automatically collect real-time road data while a vehicle is in motion.

The correct interpretation of effective traffic sign information is essential for safe driving and maintaining smooth roads. TSR [3] [4] has garnered significant attention due to its rapid expansion. However, ecological deterioration, leading to adverse weather conditions like haze, poses a challenge to intelligent transportation systems. TSR algorithms struggle in

such conditions, primarily due to poor image quality and the loss of crucial image data [5]. Therefore, it's imperative to investigate how to recognize traffic signs in hazy environments.

As TSR progresses, machines simulate and handle vast amounts of complex data, mirroring the human visual system. Systems for comprehensive object identification and detection [6] [7], among others [8], replicate human processes.

While numerous image processing techniques exist for identifying various objects in driving assistance, recognizing road traffic signals has received less attention. Road traffic signs in developed countries adhere to advanced standards, offering intuitive information crucial for driving systems. Efficient recognition of these signs can reduce traffic congestion and lower the likelihood of accidents.

Deep learning has shown remarkable potential, especially in image recognition, and is a central focus in this thesis for the detection and recognition of road traffic signs. Given the constant evolution of the automotive industry, investigating methods to detect speed limit signs in challenging conditions, such as foggy weather, is essential. Accurately identifying traffic signs under adverse conditions holds great promise for practical applications, and advancements in deep learning techniques are vital for scientific and technological progress in transportation.

1.2 Intelligent Recognition and Detection Systems for Traffic Signs

The rapid advancement of the automotive industry has prompted most major car manufacturers, including those in the United States, and Germany, to institute specialized research centers aimed at advancing technology for self-driving cars. One notable example is the TRS system, which has significant importance. Originally, the TRS system was created to enhance road safety by preventing accidents and assisting or even replacing human drivers in various driving-related tasks.

The "Stanford Cart" project, a groundbreaking endeavor initiated in 1961, marked a significant milestone in the development of autonomous vehicles. At a time when technology was still in its infancy, the project aimed to create a self-driving vehicle that could navigate its surroundings using vehicle-mounted cameras and early artificial intelligence systems. Despite its pioneering spirit and innovative approach, the "Stanford Cart" faced formidable challenges due to the limitations of computing hardware and nascent algorithms. One of the most glaring issues was its painstakingly slow pace. Progressing at a mere meter in 20 minutes, the vehicle's sluggish movement hindered its practicality and widespread adoption.

The origin of Traffic Sign Recognition (TSR) technology can be traced back to 1987 when an innovative Japanese team introduced the world's earliest TSR system. Their primary emphasis was on the recognition of speed limit signs, a crucial element in ensuring road safety. To accomplish this, they utilized classical algorithms such as threshold segmentation and template matching, which were cutting-edge at the time. Notably, the entire recognition process was completed in a mere 0.5 seconds, as documented by Fu and Huang in 2010. As TSR technology gained global traction, it wasn't long before research teams in Europe and America entered the scene, driven by the potential advantages for enhanced road safety and autonomous driving systems.

In a significant collaborative initiative in 1994, Koblenz Landau University partnered with the esteemed German automotive company, Daimler Benz, to pioneer an advanced real-time Traffic Sign Recognition (TSR) system. This collaborative TSR system was groundbreaking, showcasing exceptional performance metrics that were unparalleled at the time. It boasted an impressive recognition speed of 3 frames per second, a remarkable achievement given the computing power available in the mid-1990s. What set this system apart was its extensive dataset, comprising approximately 40,000 experimental images, making it one of the most comprehensive collections used for TSR research. However, the true measure of its success lay in the outstanding recognition accuracy it achieved. As documented by Priese in 1994, the system's recognition accuracy soared to an astonishing 98.0%Ṫhis level of precision marked a significant advancement in the realm of TSR, laying the foundation for further progress in this critical field.

As we entered the 21st century, the field of Traffic Sign Recognition (TSR) continued to experience significant growth, with research teams worldwide increasingly dedicating their efforts to this important domain. During this period, research methods matured, leading to remarkable advancements in TSR technology.

A TSR system was created in 2005 as a result of a partnership between the Nick Barnes Institute of Automation in Australia and the research laboratory headed by Gareth Loy in Sweden. This system achieved a remarkable recognition accuracy of up to 95% by using symmetry and circular shape cues that are frequently present in photos of traffic signs to determine the centroid of the sign. The University of Massachusetts had advanced TSR technology significantly by 2010. Their system utilized the principal component analysis technique with color threshold segmentation for object detection and recognition. Notably, this system achieved a recognition accuracy as high as 99.2%İt demonstrated impressive performance even in challenging conditions such as low-visibility weather or slight object occlusion. However, a limitation was the processing time of the system, preventing it from achieving real-time TSR.

China's foray into the realm of intelligent transportation and Traffic Sign Recognition (TSR) initially lagged behind some other nations, posing challenges to research in this domain. Despite the initial hurdles, recent years have seen a significant surge in development and a heightened interest in TSR technology among Chinese universities, research institutions, and enterprises. One of the early contributions to TSR research in China was spearheaded by Fleyeh (Hasan, 2008), focusing on traffic sign detection and recognition technology. Fleyeh's methodology centered on the utilization of mathematical morphology techniques and morphological skeleton functions to extract distinctive features related to warning signs. Furthermore, the recognition and classification of traffic signs were facilitated through the implementation of a template-matching algorithm. Although the initial phase encountered difficulties and yielded lower recognition rates, China's dedication to TSR research has experienced substantial growth in recent years. The country has actively invested in advancing intelligent transportation systems, and continuous research and development endeavors are expected to result in more practical and robust TSR solutions in the foreseeable future.

Xiamen University's contribution in 2009 marked a notable milestone in the TSR journey, showcasing the potential for robust solutions. The integration of visual saliency analysis such as Histogram of Oriented Gradient (HOG) features, or Support Vector Machine (SVM) classification demonstrated progress in tackling the complexities of TSR. The remarkable outcomes, with a detection rate of 98.3% and a low error recognition rate of 5.09% in TSR research, signified a significant breakthrough. This highlighted that the algorithm developed at Xiamen University effectively addressed a spectrum of challenges, including illumination variations, scaling, changes in viewing angles, and partial occlusion of traffic signs.

A comprehensive Traffic Sign Recognition (TSR) system comprises two indispensable components: detection and recognition of speed-limit signs, each accompanied by its unique set of challenges. Speed-limit sign detection encounters difficulties in adapting to the variability in sign appearance, managing illumination variations, navigating cluttered backgrounds, and addressing scale and perspective variations. Conversely, speed-limit sign recognition must contend with diverse sign content, variations in font and style, degraded signs, and the imperative for real-time processing. Overcoming these challenges successfully is crucial for the efficient implementation of TSR systems, guaranteeing precise detection and recognition of speed-limit signs in intricate real-world driving scenarios.

1.3 Contributions

Using deep learning techniques for TSR, this thesis compares and enhances the algorithms. The following are the thesis's contributions:

Using the GTSDB and GSTRB datasets, this thesis will offer a visual means of comparing various traffic sign recognition and classification techniques. Using the most recent deep learning models (YOLO, Faster R-CNN, and multiple classification models), we ran a test for traffic sign identification and completed a comparison of all suggested algorithms. The evaluation's findings showed that LeNet and YOLOv8n performed better than the other models.

We improve the existing model by using data manipulation methods, parameter tweaking, and model combinations to try and achieve a better result. We also made a dataset that is endemic to Vietnam and could be used for future research related to TSR.

1.4 Objective of the thesis

First, we work with two traffic sign datasets: the German Traffic Sign Recognition Benchmark (GTSRB) and the German Traffic Sign Detection Benchmark (GSTDB) for detection and classification, respectively. These datasets give us access to a large range of photos of traffic signs, which is crucial for our study.

After that, we will create two different deep learning methods: one that makes use of the well-known Faster R-CNN architecture and another that makes use of YOLO, a cutting-edge object recognition model that is renowned for its effectiveness and accuracy.

After our models are operational, a detailed comparison is warranted. We will compare the performance of these two approaches in detail for both datasets. To get the whole picture, we'll examine several factors, such as identification speed and accuracy.

Finally, more development will be done on the Traffic Sign Recognition (TSR) approach which performs better in terms of recognition speed and accuracy. We will be able to construct an efficient and effective TSR system by fine-tuning and optimizing the selected technique through this iterative process.

1.5 Structure of the thesis

In the first section, we describe the topic's history and current state, as well as the status of related topics in recent years. We study the primary challenge and research contents of this subject and explain the main technical route of this thesis. Finally, we give a short description of the organizational structure.

The second section of our discussion covers the essential ideas and relevant methods related to this topic. We present the basic ideas behind convolution neural networks and how

they are used to identify and detect traffic signs. The following sections of this thesis are theoretically supported by the introduction of the pertinent road sign recognition principles.

In the third section, we show our working curriculum throughout the 14 weeks of the project, as well as our code and data links with a small description of them.

The TSR experiment is covered in the fourth section. If a one-shot model cannot be created, we use YOLO and the Faster R-CNN model for detection and LeNet, VGG16, VGG19, and ResNet50 for classification. Our approach consists of two models: one that combines the traffic sign recognition and classification in a single shot, and another that separates the two aforementioned functions. We intend to investigate whether or not there is a discernible difference between the two in terms of processing speed and accuracy. The German Traffic Sign Recognition Benchmark (GTSRB) will be used to train all of our models, allowing us to compare their accuracy to that of other papers' models.

In the fifth section, we mainly discuss the results of the recognition and classification model of traffic signs and compare our method to others' similar research.

We examine and debate the contributions made to this thesis in the sixth section. Simultaneously, we examine the drawbacks of our experiment as well as algorithmic enhancements.

In the seventh and last section, we depict the novelty and creativity of this thesis, analyze the existing problems in this field of research, and put forward new ideas for future research projects.

Chapter 2

Literature Reviews

In this section, we lay the foundation by introducing and discussing the pioneering works of distinguished researchers, including master's and doctoral thesis studies, as well as contributions from accomplished professors who have delved into the problem under consideration. Their research encompasses a diverse array of models, including but not limited to Faster R-CNN, YOLO, and other state-of-the-art methodologies. These investigations have greatly contributed to the evolving landscape of our understanding in this domain. We offer a comprehensive overview of their critical insights, methodologies, and findings, setting the stage for our own novel approach, which builds upon the lessons and advances from their pioneering work.

2.1 Faster R-CNN Models

2.1.1 Traffic Sign Identification Using Deep Learning

Published in 2019 by Ratheesh Ravindran, Mohammad Fanaei, Michael J. Santora, and Mariam Faied [9], this paper focuses on evaluating Traffic Sign Recognition (TSR) systems based on Mean Average Precision (mAP) and Frames Per Second (FPS). The authors prioritize selecting Deep Neural Networks (DNN) based on application-oriented performance.

The GTSDB dataset is utilized, and images are categorized into training, validation, and testing sets for DNN transfer learning. Certain traffic signs, such as "Right One Way," "Left One Way," and "Road Closed," are excluded. A new dataset is generated from real-time video recordings during trials on the University of Detroit Mercy campus, recorded by ROSBAG.

Challenges arise in distinguishing certain traffic signs due to factors like constraints on the training dataset, shape, and text features. To address this, the authors propose using a second text identification technique alongside the DNN. This method recognizes text from the

Region of Interest (ROI) in an image, improving traffic sign classification accuracy. Tesseract version 4.0, a Long Short-Term Memory (LSTM) artificial recurrent neural network, is employed as the text identification layer.

The study reveals that Faster R-CNN Inception V2, the underlying DNN, strikes a balanced trade-off between mAP and FPS (with mAP = 90.6% and average FPS of about 16.94). Integrating the Tesseract LSTM network enhances text detection, improving overall DNN performance in traffic sign detection. Real-time testing on the Polaris Gem e2 platform and ROSGazebo simulations demonstrate the effectiveness of the proposed architecture. The authors suggest further optimization through using the NVIDIA Profiler and tuning the GPU platform's behavior for enhanced FPS performance.

2.1.2 Traffic Signs Detection and Recognition System using Deep Learning

The 2019 paper authored by Pavly Salah Zaki, Marco Magdy William, Kerolos Gamal Alexsan and Bolis Karam Soliman[10] introduces an efficient method for real-time detection and recognition of traffic signs, tackling challenges such as adverse weather conditions, varied lighting, and limited visibility. The study concentrates on advanced multi-object detection systems, specifically Single Shot Multibox Detector (SSD) and Faster Recurrent Convolutional Neural Networks (F-RCNN), incorporating feature extractors like MobileNet v1, Tiny-YOLOv2 and Inception v2 to address traffic sign detection challenges. The authors particularly highlight Tiny YOLO v2 and F-RCNN Inception v2 for their superior performance.

The German Traffic Signs Detection Benchmark (GTSDB) dataset is utilized for training and testing, classifying traffic signs into prohibited, mandatory, and dangerous types. While acknowledging the limitations of the dataset size (900 photos in total, with 800 for training and 100 for testing), the researchers argue that it may be insufficient for complex models like F-RCNN Inception.

The paper's main contributions include the successful application of the F-RCNN Inception v2 model, delivering accurate and swift results in challenging driving scenarios with an average accuracy of 96%Ȧdditionally, the Tiny-YOLOv2 model is presented as a fast alternative with reasonable accuracy, with the authors recommending YOLOv2 or YOLOv3 for higher precision requirements. The Inception v2 model trained on the German Dataset for Recognition which is GTSRB achieves a record accuracy of 99.8% with 39,200 photos and 43 classes, following the same configuration as outlined in section III-D. The authors propose that further accuracy improvements can be attained by leveraging a powerful GPU,

significantly increasing training data (at least 40k photos with an average of 1,000 images per class), and extending the training duration.

2.1.3 Traffic sign detection method based on Faster R-CNN

In their 2019 publication [11], Houjie Li, Linxiu Wu, Xuan Chen and Jianjun He from IOP Publishing Ltd. introduce a novel method for traffic sign detection using the Faster R-CNN deep learning framework. The approach involves utilizing a convolutional neural network to automatically extract features from traffic sign images. These features are then processed by a Region Proposal Network (RPN) to filter foreground objects and regress bounding boxes. The identified regions are mapped back to the feature map, and fixed-size proposal boxes are generated using the Region of Interest pooling layer (RoI). A classification network is subsequently employed for specific classification tasks and further bounding box regression.

The method's effectiveness and robustness are assessed on the German Traffic Sign Detection Benchmark (GTSDB), demonstrating its capability to handle various challenges, including different lighting conditions, obstructions, and motion. The dataset comprises 900 images, with 600 for training and 300 for testing, all sharing a common resolution of 1360 x 800 pixels. Within this dataset, 1213 traffic signs are present, categorized into danger, mandatory, prohibitory and a miscellaneous category.

Mean Average Precision (mAP) and Average Precision (AP) are the two primary measures used in the evaluation. The area under the Precision-Recall (PR) curve is used to determine AP, and the average of these AP scores across all categories is known as mAP. The PR curve is displayed in Figure 2.1, which shows the experimental findings. The obtained mAP of 91.75% indicates outstanding performance. Furthermore, the suggested approach demonstrates resilience when confronted with obstacles including fluctuating illumination, motion blur, obscured signs, and more.

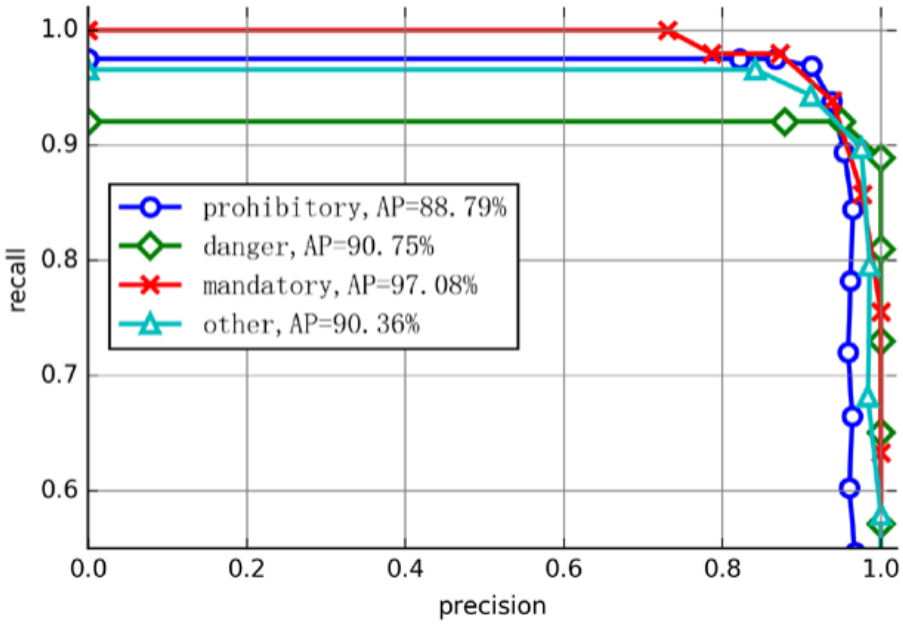


Figure 2.1 Each category AP results from statistics in the experiment

2.2 YOLO Models

2.2.1 Improved YOLOv5 network for real-time multi-scale traffic sign detection

Yi Chen, Junfan Wang, Zhenkang Dong, and Mingyu Gao[12] present an advanced feature pyramid model called AF-FPN in their paper. In order to address information loss during feature map generation, AF-FPN integrates the Adaptive Attention Module (AAM) and Feature Enhancement Module (FEM), which improves the feature pyramid's capacity for representation. AF-FPN replaces the traditional feature pyramid network in YOLOv5, improving detection performance for multi-scale targets while preserving real-time capabilities. The study additionally offers a novel automatic data augmentation technique to expand the dataset and strengthen the model's resilience, improving its applicability in real-world settings. Their approach shows better performance and greater versatility when compared to several state-of-the-art methods, as demonstrated by extensive experiments on the Tsinghua-Tencent 100K (TT100K) dataset.

The evaluation of the enhanced YOLOv5 model is carried out on the TT100K dataset, which encompasses 182 distinct types of traffic sign instances, each meticulously annotated. This dataset provides a representation of diverse real-world traffic environments. Notably,

around 42.5% of the traffic signs within TT100K are categorized as small objects, making it particularly well-suited for real-world vehicle-mounted target recognition tasks. The distribution of traffic sign sizes in the dataset is visually depicted in Figure 2.2.

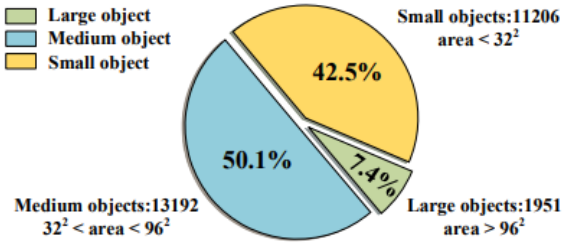


Figure 2.2 Size distribution of sign instances from the TT100K

The authors carried out a thorough assessment on the TT100K dataset, contrasting their method with a number of alternative models, in order to demonstrate the benefits of their suggested approach in traffic sign detection. The YOLOv3, the original YOLOv5, YOLOv5-Lite, YOLOv5-face, Efficientdet, M2det, and SSD models were all compared. Model size, parameters, floating-point operations per second (FLOPs), mean average precision (mAP), average precision scores for large, medium, and small-sized targets (APL, APM, APS), and frames per second (FPS) were among the metrics that were evaluated in the performance assessment. More specific findings are shown in Table 2.1.

Method	Model	Parameters	FLOPs	APs	APm	APl	mAP@50	FPS
Efficientdet-d0	15.15M	3.752M	2.50B	-	0.3980	0.5526	0.5786	26
M2det	340M	147M	16.35G	0.0300	0.3198	**0.6586**	0.4658	12
Mobilenet-SSD	118M	25.067M	29.20G	0.0275	0.2443	0.5261	0.3200	22
YOLOv3	119.6M	59.578M	158.00G	0.3889	0.4454	0.4288	0.6186	27
YOLOX-s	69.0M	9.010M	27.03G	0.4122	**0.5975**	**0.6081**	**0.6860**	59
YOLOv5-Lite-g	11.6M	5.277M	15.60G	0.3218	0.4914	0.4957	0.5431	71
YOLOv5	14.6M	7.193M	17.90G	0.3567	0.5142	0.5184	0.6018	**105**
[12]	16.6M	8.039M	17.90G	**0.4146**	**0.5783**	**0.5817**	**0.6514**	**95**

Table 2.1 Comparison with other method on the TT100K dataset

2.2.2 Traffic-Sign Recognition Using Deep Learning

In a 2021 paper by Wei Qi Yan and Zhongbing Qin [13], the authors explore the testing and application of two models, Faster R-CNN and YOLOv5, for the Traffic Sign Recognition problem. The evaluation was conducted in New Zealand using the NZ-Traffic-Sign 3K dataset, which comprises 3,436 images and a total of 3,545 instances across various traffic

sign categories which including Stop, Keep Left, Road Diverges, Road Bump, Crosswalk Ahead, Give Way at Roundabout, and Roundabout Ahead.

To mitigate overfitting, the authors introduced noise and blurring to the dataset during training using the Skimage library. For the Faster R-CNN model, a traditional CNN was employed as the basic convolutional layers for feature extraction, utilizing a pre-trained VGG16 model to assist. The data structure was divided into five parts: Annotations, ImageSets, JPEGImages, and SegmentationClass, as well as SegmentationObject.

The model's parameters include:

Parameters	Momentum	Learning rate	Max Epoch	Batch size	Weight Decay	giou	cls	cls_pw	obj	obj_pw
Setting	0.95	0.00128	200	16	0.000201	1.2	15.7	3.67	20	1.36

Table 2.2 YOLOv5 Model Parameters

Caffe is preinstalled to set up the experimental environment. CUDA and Tesla V100-SXM2-16GB are also necessary, in order to train the model with VGG16. The model returns the mAP@0.5 according to the pixel rate of image, which is 0.907 - 0.977 - 0.945, with the pixel rate is ≤ 200, $[200, 400]$, and ≥ 400, respectively.

The authors used the LabelImg tool to label the dataset and indicated that the activation functions in YOLOv5 are sigmoid and LReLU in their evaluation of YOLOv5. The sigmoid function is added to the final detection layer, and LReLU is integrated into the middle/hidden layers. Furthermore, they switched from SGD to ADAM as the optimizer. The experiment, carried out on Colab with a Tesla V100-SXM2-16GB, produced results that were displayed in a pixelated format identical to that of the Faster R-CNN: 0.883 - 0.976 - 0.931.

Summarily, the Faster R-CNN model produced superior results, but its processing time of 37 seconds renders it unsuitable for real-time recognition. On the other hand, YOLOv5 exhibited faster processing times, completing recognition in 0.011 seconds for a 2,074-frame testing video. Consequently, the authors recommend using Faster R-CNN for recognition tasks without time constraints, while YOLOv5 is deemed more suitable when rapid processing times are essential.

2.2.3 Improved YOLO v5 with balanced feature pyramid and attention module for traffic sign detection

An improved version of the YOLOv5 (You Only Look Once) network model was introduced in 2022 by Hui Liu, Linfeng Jiang, Guangjian Zhang, and Hong Zhu [14]. In order to improve feature fusion and extraction, they integrated a global context block and a balanced feature pyramid structure. They carried out extensive experiments with the Tsinghua-Tencent-100K

(TT100K) dataset, which consists of almost 10,000 high-resolution photos, to validate their approach. For their experiments, they carefully chose 45 classes totaling more than 100 samples. The improved model is assessed and compared to the original YOLO v5 in the paper using metrics like recall, precision, and mean average precision, or mAP. The improved YOLO v5 and the original YOLO v5 were first compared by their groups.

	P	R	mAP@0.5	mAP@0.5:0.95
YOLOv5	0.850	0.828	87.8%	67.2%
[14]	0.874	0.861	89.7%	69.3%

Table 2.3 Comparison with the original method

Based on the data presented in Table 2.3, it is apparent that when comparing their model with the original YOLO v5 network, their model demonstrates significant improvements. Specifically, their model achieves a 2.4% improvement in precision and a 3.3% boost in recall. Moreover, there is a corresponding enhancement in mAP@.5 and mAP@.5:0.95, with improvements of 1.9% and 2.1% respectively. Furthermore, they also include a comparison with state-of-the-art object detection methods, and the results of this comparison are detailed in Table 2.4.

	SSD300	Faster RCNN	YOLOv3	YOLOv4	YOLOv5	[14]
mAP@0.5	63.71%	79.10%	82.40%	86.80%	87.8%	89.70%

Table 2.4 Comparison with the others method

2.3 Other Models

2.3.1 Deep Learning for Large-Scale Traffic-Sign Detection and Recognition

In an article published in 2019 on traffic sign detection in Slovenia, Domen Tabernik and Danijel Skocaj [15] used a Convolutional Neural Network (CNN) approach on their modified dataset using mask and Faster R-CNN. Developing a model appropriate for practical use in inventory management of traffic signs was the aim.

The DFG Consulting d.o.o. organization provided the dataset, which goes by the name DFG traffic-sign dataset, which was collected with the intention of monitoring the road sign inventory on Slovenian roads. The collection consists of 6,957 photos with 13,239 closely annotated examples of traffic signs, representing 200 categories. Less than 200

photos have a lower resolution (720 x 576), while the majority have a high resolution (1920 x 1080). In order to compare their results with a traditional dataset, they also conducted tests on the Swedish traffic-sign dataset (STSD), which comprises roughly 20 categories with straightforward traffic signs in over 19,236 photos.

There are two modules in both the Mask R-CNN and Faster R-CNN models. The first module generates a set of rectangular object proposals with an objectness score using a deep fully convolutional network called a Region Proposal Network (RPN). The second module, called Fast R-CNN, is a region-based CNN that divides the suggested regions into pre-established groups.

They used a Caffe2-based Python implementation of Detectron for both models. A VGG16 network with 13 convolutional layers and 3 fully-connected layers was used by the Faster R-CNN. The Mask R-CNN, on the other hand, employed a ResNet-50 architecture with 16 convolutional layers that had kernel sizes of at least 3x3, and the remaining convolutional layers had kernel sizes of 1x1. A Feature Pyramid Network (FPN) was also integrated into the Mask R-CNN in order to gather data from tiny objects. Notably, they tested the ResNet-101 architecture as well, but decided to concentrate on the ResNet-50 instead because the results were similar but required a slower processing time.

Both approaches trained with a model that had already been pre-trained on ImageNet and used comparable learning hyper-parameters. A weight decay of 0.0005 and a learning rate of 0.001 were employed by the Faster R-CNN and 0.0025 and 0.0001, respectively, by the Mask R-CNN. The momentum used in both methods was 0.9. They underwent 95 epochs of end-to-end training, with a factor of 10 learning rate reduction at the 50th and 75th epochs. For the STSD dataset, which uses two GPUs per batch, and the DFG dataset, which uses four GPUs per batch, the training used two images per batch per GPU for the STSD and eight images per batch for the DFG.

The following are the result of both models testing on the Swedish traffic-sign data set (STSD):

Traffic Sign	FCN [6]		Faster R-CNN			Mask R-CNN (ResNet-50)					
						No adaptations			With adaptations (our)		
	Prec.	Rec.	Prec.	Rec.	AP^{50}	Prec.	Rec.	AP^{50}	Prec.	Rec.	AP^{50}
PED. CROS.	100.0	95.2	92.6	92.6	94.1	100.0	97.5	98.2	99.2	97.6	97.6
PASS RIGHT SIDE	95.3	93.8	98.1	98.1	99.5	94.8	98.2	98.6	100.0	98.2	99.8
NO STOP/STAN	100.0	75.0	92.3	92.3	86.5	81.2	100.0	95.4	86.7	100.0	83.9
50 SIGN	100.0	100.0	81.2	92.9	90.3	87.5	100.0	97.5	90.0	96.4	96.9
PRIORITY ROAD	100.0	98.9	98.7	95.1	92.1	97.5	97.5	96.9	98.7	92.9	89.8
GIVE WAY	96.7	96.7	100.0	94.1	94.1	100.0	91.4	91.4	100.0	94.1	94.1
70 SIGN	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
80 SIGN	94.4	77.3	100.0	95.2	95.2	95.2	100.0	99.8	100.0	100.0	100.0
100 SIGN	90.5	100.0	94.1	88.9	92.5	100.0	61.1	74.8	100.0	93.8	93.8
NO PARKING	100.0	92.1	96.8	90.9	98.5	96.7	90.6	95.9	100.0	93.9	96.5
Averaged	**97.7**	92.9	95.4	94.0	94.3	95.3	93.6	94.9	97.5	**96.7**	**95.2**

Figure 2.3 Details Result on STSD

Average	R-CNN [6]	FCN [6]	Faster R-CNN	Mask R-CNN (ResNet-50)	
				No adapt.	Adapt. (ours)
Precision	91.2	**97.7**	95.4	95.3	97.5
Recall	87.2	92.9	94.0	93.6	**96.7**
F-measure	88.8	95.0	94.6	93.8	**97.0**
mAP^{50}	/	/	94.3	94.9	**95.2**

Figure 2.4 Evaluation on STSD

The following figure shows the outcomes of testing the two models using the DFG dataset as well as their average precision per class. The green and red bars show changes in performance (an increase in green and a decrease in red) compared to the base Mask R-CNN (ResNet-50) without their improvements, while the blue bars represent Mask R-CNN (ResNet-50) with their enhancements and data augmentation:

	Faster R-CNN	Mask R-CNN (ResNet-50)		
		No adapt.	With adapt.	With adapt. and data augment.
mAP^{50}	92.4	93.0	95.2	**95.5**
$mAP^{50:95}$	80.4	82.3	82.0	**84.4**
Max recall	93.8	94.6	**96.5**	**96.5**

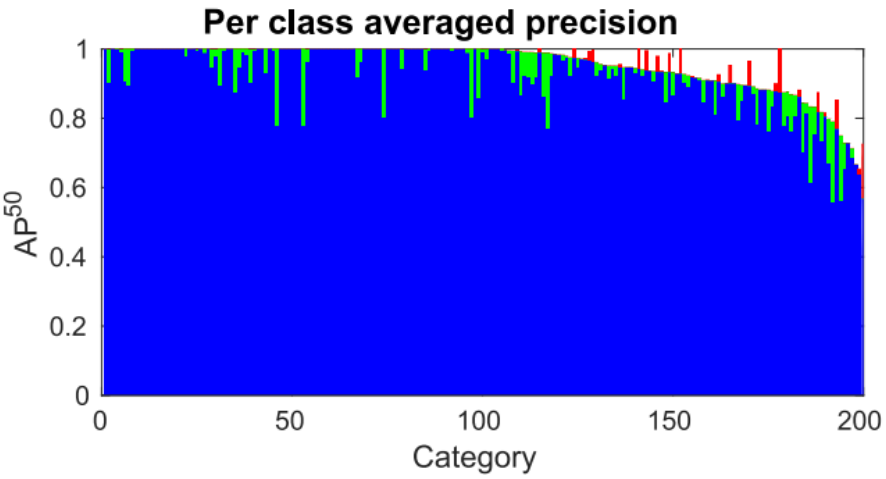


Figure 2.5 Results on DFG Traffic-Sign data set

The following is the precision of the model base on the traffic sign size:

Traffic-sign size (% signs retained)	Faster R-CNN		Mask R-CNN				Mask R-CNN with adapt. and data augmentation (ours)			
			ResNet-50		ResNet-101		ResNet-50		ResNet-101	
	Max recall	mAP^{50}	Max recall	mAP^{50}	Max recall	mAP^{50}	Max recall	mAP^{50}	Max recall	mAP^{50}
min 30 px (100%)	93.8	92.4	94.6	93.0	94.8	93.2	**96.5**	**95.5**	96.1	95.2
min 40 px (89%)	96.1	95.0	96.8	95.3	96.8	95.3	**97.4**	**96.7**	97.0	96.4
min 50 px (80%)	96.6	95.0	96.7	94.9	96.8	95.2	**97.2**	**96.0**	96.8	95.5

Figure 2.6 Details Result on DFG with Difference Sizes

In summary, although there have been a few minor inaccuracies (2–3%) in real detection instances, the researchers affirm that deep learning is an appropriate method for this issue. For numerous traffic-sign categories, including several complex ones with substantial intra-class variability, the models performed remarkably well. They stress that peculiar viewing angles, occlusions, and the similarity of some categories are the main causes of the errors.

2.3.2 Lightweight deep network for traffic sign classification

Two cutting-edge lightweight networks were introduced in 2019 by Wei Wang, Jianming Zhang, Jin Wang, Chaoquan Lu, and Arun Kumar Sangaiah [16]. They were designed to be deployed in resource-constrained environments where network efficiency is essential. The goal was to reduce the number of trainable parameters in the models while still achieving higher recognition precision. The proposed approach uses knowledge distillation, which is the process of moving information and understanding from a teacher network that has already received training to a smaller student network. Additionally, a new module is added to the teacher network that combines two streams of highly connected feature channels in order to enhance traffic sign recognition precision. With its simple end-to-end architecture that includes a fully connected layer and five convolutional layers, the student network is designed to be easily deployed on mobile devices.

The German Traffic Sign Recognition Benchmark (GTSRB) dataset, which presents difficulties for traffic sign classification tasks, is used for the evaluation. It consists of single-image, multi-class data. 51,839 instances in all, ranging in size and shape from 15 by 15 to 250 by 250, make up the dataset. There are 43 classes (such as prohibited, dangerous, and mandatory signs) with between 100 and 1000 images in each. There are 39,209 images in the training set and 12,630 images in the testing set. Furthermore, the Belgian Traffic Sign Classification dataset (BTSC) is utilized, which comprises 4533 training images, 2562 testing images, and 62 different types of traffic signs. With fewer training samples and a wider variety of traffic signs, the BTSC dataset makes accurate classification more difficult because of occlusions, variable weather, and other factors.

The student model exhibits an improved average recognition rate of 99.61% while the teacher model achieves an impressive 99.23% recognition rate. The teacher network achieves 93.16% accuracy on the CIFAR-10 general dataset in training and testing. The student network uses only 0.8 million parameters to achieve high accuracy rates of 99.61% and 99.13% on the GTSRB and BTSC datasets, respectively, thanks to knowledge transfer from the teacher network. The results of these experiments demonstrate how useful the lightweight networks that have been suggested could be for implementing deep convolutional neural networks (CNN) on mobile embedded devices.

2.3.3 Real-time traffic sign recognition based on a general purpose GPU and deep-learning

A real-time traffic sign detection and recognition system based on a General Purpose Graphics Processing Unit (GPGPU) that demonstrates resilience against illumination changes was

introduced in 2017 by Yongwon Hong, Kwangyong Lim, Hyeran Byun, and Yeongwoo Choi [17]. The suggested approach performs reliably in low light, attaining an impressive 0.97 F1-score on an extensive dataset that includes traffic rules from the Vienna Convention (South Korea and Germany). Real-time hierarchical recognition and detection are achieved with this technique.

Two different datasets were used for this methodology evaluation: one was a real-world driving dataset that captured scenes straight from a car-mounted camera, and the other was the LISA traffic sign dataset. The LISA dataset includes Southern California driving scenes with a variety of US traffic signs, warnings, stops, and no-turn signs, as well as speed limits. Interestingly, the traffic rules in Vienna are not the same as the LISA dataset's. In order to address this, the researchers collected driving scenes from Germany and Korea, two countries with Vienna traffic laws in place, guaranteeing a thorough assessment in both cases.

Name	Type	Location	Frame	Signs	Resolution
LISA-SPEED	Daylight, Cloudy	USA	641	637	1280 x 960
DE-D	Daylight, Rain, Cloudy	Germany	4765	2001	1280 x 672
KR-D	Daylight, Rain, Cloudy	Korea	4527	2294	1280 x 672
KR-N	Night	Korea	3086	1343	1280 x 672

Table 2.5 Driving Scenes

The suggested GPGPU-based method uses a real-time traffic sign detection and recognition technique to overcome obstacles. In order to identify candidate regions, the process starts with extracting Byte-MCT features from the input driving scene. Unlike AdaBoost-based techniques, which rely on multiple sliding-windows for particular shapes, such as triangles or circles, the suggested method makes use of landmark-based parallel-windows. Multiple traffic sign candidates can be detected with this strategy, illustrating robust performance even under difficult circumstances like dim lighting or nighttime situations. In order to ensure real-time processing, the method also uses a GPGPU-based parallel window searching technique that runs in parallel across every divided grid of the input image.

The feature vectors, which are concatenations of Byte-MCT histograms, are then subjected to the SVM verification procedure, enabling the differentiation of traffic signs with comparable shapes. The found windows are categorized by the SVM into triangles, circles, and other shapes. Convolutional Neural Networks are used in the final recognition stage to classify different traffic signs (CNN).

On the VIVA test-set, the suggested method outperforms the ACF (Aggregated Channel Feature) result of 84.3% with an accuracy of 89.5%Ȧdditionally, the approach achieves

an average accuracy of 97.9% on datasets that adhere to Vienna Convention traffic rules, a notable 26.8% improvement over the ACF result, which averages 71.1%Ṫhe suggested approach has potential for use in smart car applications, especially with GPGPU devices such as DRIVER PX. Therefore, it could be incorporated into the automotive sector afterwards.

2.3.4 Deep Learning Traffic Sign Detection, Recognition and Augmentation

In a ground-breaking work released in 2017 [18], Aref Meddeb and Lotfi Abdi presented a novel real-time method for effective and precise traffic sign recognition using Augmented Reality (AR) and Cascade Deep Learning. This creative technique allows for the overlay of augmented virtual objects on real-world scenes, accommodating a range of driving situations, including inclement weather. They show through extensive experiments that combining deep convolutional neural networks with Haar Cascade greatly improves detection capabilities without sacrificing real-time performance.

The proposed method can be comprehensively evaluated thanks to the utilization of the free and open-source Caffe deep learning package in the implementation. The German Traffic Sign Recognition Benchmark (GTSRB) dataset, which consists of 51,839 images divided into 43 classes for training and testing, is used to evaluate real-time performance. A highly effective classifier is built using a seven-stage cascade, with a minimum hit rate of 0.995 and a maximum false positive rate of 0.5. The algorithm's effectiveness is demonstrated by the results, which show an astounding average recall rate of 98.22%and precision rate of 98.81%.

With an overall accuracy of 99.36%,the classification task on the GTSRB testing photos demonstrates the classification module's strong performance. The methodology consists of projecting the appropriate 3D object sign using a sparse dictionary, then classifying the projected vector using support vector machines (SVMs). Operating on positive Regions of Interest (ROIs), the multi-class sign classifier achieves 99.36% classification accuracy. The suggested Cascade Deep Learning-based method is a promising development for increasing driver comfort, lowering the risk of traffic accidents, and increasing the accuracy of traffic sign detection because it not only performs on par with state-of-the-art methods, but also shows reduced computing complexity and training time.

Chapter 3

Project Management Plan

Our project has two primary objectives, each with distinct core values. The first objective (1) entails the creation of a custom traffic sign dataset, meticulously crafted by our team. The second objective (2) involves the development of a prototype solution to tackle the Traffic Sign Recognition (TSR) problem, which will be achieved by fine-tuning a model. These endeavors are to be completed within a defined time frame spanning 15 weeks, commencing on the 4th of September and concluding in the middle of December. The comprehensive project plan is detailed in the Table 3.1, while the data employed in our project will be presented in the subsequent which is Table 3.2.

Table 3.1 Project Schedule

Task Name	Priority	Owner	Start Date	End Date	Status	Issues
Find Documents	High	Team	9/4/2023	9/10/2023	Finished	None
Review Papers	Medium	Team	9/4/2023	9/10/2023	Finished	None
Write "Introduction" and "Related Work"	High	H.Binh & T.Nam	9/11/2023	9/20/2023	Finished	None
Clean dataset	Medium	H.Duc	9/11/2023	9/17/2023	Finished	None

Write extra detail for the thesis	Medium	H.Binh	9/11/2023	9/20/2023	Finished	None
Initialize detection model	High	Team	9/18/2023	10/8/2023	Finished	Due to the equipment and big datasets some models can't finished
Prepare for Review 1	High	T.Nam	9/25/2023	9/28/2023	Finished	None
Adjust per Review 1's Feedback	High	Team	10/2/2023	10/4/2023	Finished	None
Write "Project management plan"	Medium	Binh	9/4/2023	9/10/2023	Finished	None
Gather and label data	High	Team	10/9/2023	10/15/2023	Finished	None
Experiment	Medium	Team	10/16/2023	10/22/2023	Finished	None
Compare Result	Low	Team	10/17/2023	10/22/2023	Finished	None
Write "Result" and "Conclusion"	High	H.Binh & T.Nam	11/1/2023	11/25/2023	Finished	None
Building prototype	Medium	Team	11/29/2023	12/01/2023	Finished	None
Prepare for Review 2	High	T.Nam	10/24/2023	10/28/2023	Finished	None

Review and condense gathered data	Low	Team	10/27/2023	11/04/2023	Finished	None
Adding extra data	Low	Team	11/13/2023	11/18/2023	Finished	None
Prepare for Review 3	High	Team	11/23/2023	11/26/2023	Finished	None
Final adjustment	High	Team	11/27/2023	12/01/2023	Finished	None
Final look back	Medium	Team	12/02/2023	12/02/2023	Finished	None
Prepare for the thesis defend	High	Team	12/04/2023	12/15/2023	Finished	None

Table 3.2 Source Code and Data

Item	Link	Description
GTSDB	https://benchmark.ini.rub.de/gtsdb_dataset.html	German Traffic Sign Detection Benchmark
Recorded dataset	https://drive.google.com/drive/folders/1oUkIgwitlpFd8nJC3THlkIk2cHD_y5Xl?usp=sharing	Group recorded dataset
Source code	https://www.kaggle.com/code/osiris543/yolov5-in-gtsdb	Source code of the YOLOv5 detection model run on GTSDB dataset
Source code	https://github.com/sidthoviti/Traffic-Sign-Detection-with-Faster-R-CNN-using-PyTorch	Source code of the Faster R-CNN detection model run on GTSDB dataset
Source code	https://github.com/0infinitive0/traffic-sign-detection-combine.git	Source code for running real-time with jointed model

Source code	https://www.kaggle.com/nguyenhuuduck16hcm/gtsrb-sign-detection-cnn-vs-vgg19	Source code of classification on GTSRB dataset
Source code	https://www.kaggle.com/nguyenhuuduck16hcm/manual-data-sign-detection-cnn-vs-vgg19	Source code of classification on manual dataset

Chapter 4

Materials and Methods

Within this section, we delve into a comprehensive exploration of the model and datasets employed in our project. We will discuss not only the pre-existing datasets but also the proprietary data collected for our research. Additionally, we will delve into the specifications and characteristics of the final model, encompassing its inherent values and parameters, offering a holistic view of the model's attributes and properties.

4.1 Dataset

4.1.1 German Traffic Sign Detection Benchmark (GTSDB) and German Traffic Sign Recognition Benchmark (GTSRB)

Figure 4.1 GSTDB and GSTDB's classes

German Traffic Sign Detection Benchmark (GTSDB)

The German Traffic Sign Detection Benchmark (GTSDB) holds significant importance in the realms of computer vision, pattern recognition, and image-based driver assistance. Launched during the IEEE International Joint Conference on Neural Networks in 2013, this benchmark serves as a fundamental resource for researchers. Comprising 900 images meticulously categorized into 600 for training and 300 for testing, the GTSDB addresses the challenge of single-image traffic sign detection. The dataset classifies signs into three distinct types and is designed to accommodate varying sign counts, sizes ranging from 16x16 to 128x128 pixels, and challenging lighting and perspective conditions.

One of the standout features of GTSDB is its online evaluation system, which facilitates real-time result analysis. This system provides a robust platform for assessing detection algorithms across diverse scenarios. The dataset's images are stored in PPM format, while annotations are meticulously detailed in CSV files. These annotations include crucial informa tion such as Region of Interest (ROI) coordinates and sign class IDs, offering comprehensive insights for researchers and practitioners. The GTSDB, with its rich variety and detailed annotations, stands as a cornerstone dataset for advancing research in the field of traffic sign detection.

The Figure 4.2 below is the label and distribution of the GTSDB dataset:

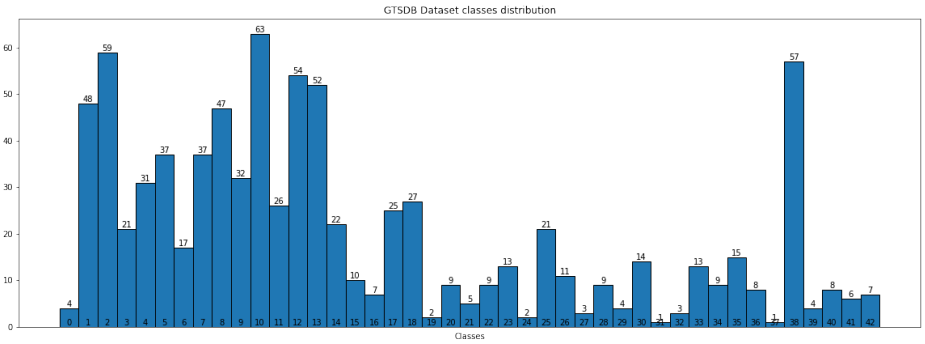


Figure 4.2 GTSDB Dataset Classes Distribution

German Traffic Sign Recognition Benchmark (GTSRB)

The German Traffic Sign Recognition Benchmark, initially introduced at the International Joint Conference on Neural Networks (IJCNN) in 2011, stands as a prominent evaluation plat form for single-image, multi-class classification challenges. Featuring over 40 distinct classes and a dataset comprising 50,000 realistic images, this benchmark welcomes participants

from diverse fields without requiring specialized domain knowledge. The primary focus of this benchmark is to address the complexities associated with single-image classification, particularly in the context of traffic signs.

The dataset is meticulously designed to exhibit real-world diversity, with unique instances capturing various scenarios. The training set is organized by class, and annotations are pro vided in CSV files, offering detailed information for each track. For edge-based approaches, each track consists of 30 images of a single physical traffic sign, with a 10% border around the sign. The images are stored in the PPM format, featuring sizes ranging from 15x15 to 250x250 pixels. The annotation format includes essential details such as filename, dimen sions, and bounding box coordinates, facilitating a comprehensive evaluation of classification algorithms on a diverse and realistic traffic sign database.

The Figure presented below illustrates two distributions pertaining to the training and testing subsets of the dataset:

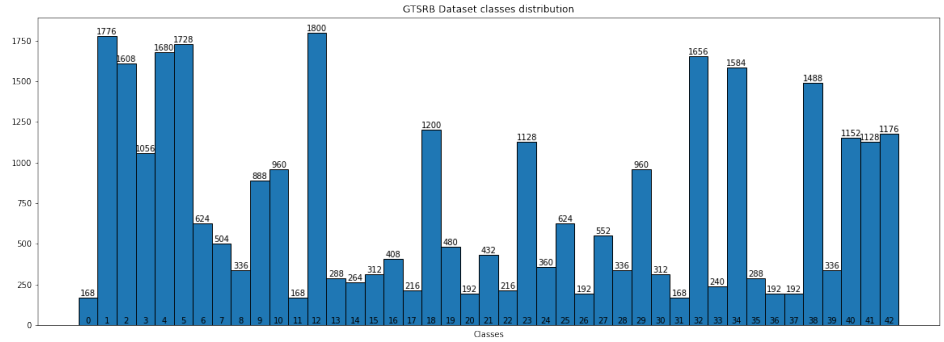


Figure 4.3 GTSRB Train Classes Distribution

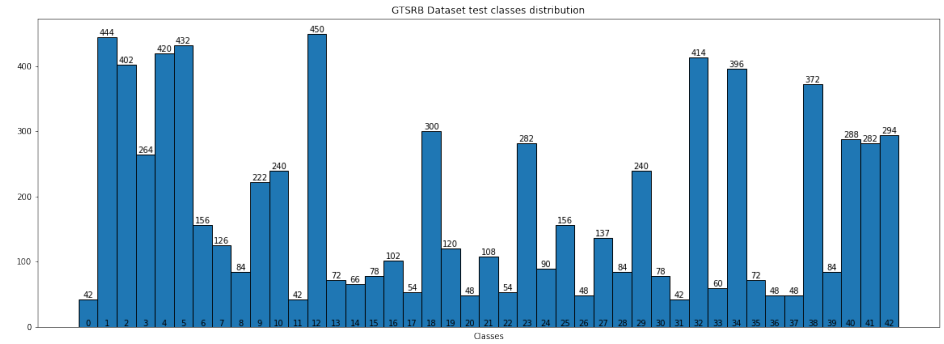


Figure 4.4 GTSRB Test Classes Distribution

4.1.2 Our collected data

For our collected data, we use a GoPro 10, which have values of 1920x1080 and 2704 x 1520 for frame width and height, all are recorded at a rate of 60 FPS. We recorded these data from 11 AM to 4 PM in various regions of Ho Chi Minh City, mainly in district 1 and 3. After encode back into the H.254 codec using HandBrake, we eventually get a raw dataset consisting of 27 videos that weigh around 45GB. After we segmented our video into frames and got rid of those that are not relatable to what we needed, mainly scenes that do not have any sign in it, we have reduced it down to 11.5GB of data, which is about 22000 images.

The details of the dataset is written in the 5.1 section of the *Result* chapter, as it is one of our objectives for this thesis project.

4.2 Detection Model

Visual object detection is one area where deep learning has numerous practical applications. Its objective is to identify and locate objects within images, a process involving two primary steps: (1) determining the object's position within the image, and (2) determining its category. However, this process is complex due to the wide range of object sizes, orientations, and placements in high-resolution images.

Traditionally, the approach to this problem is to create a deep neural network that takes image data, object labels, and positions as input. The network employs convolutional neural layers and two fully connected layers to classify objects, precisely locate them, and provide the object's exact coordinates within the image.

Figure 4.5 provides an illustration of the workflow. The first fully connected layer generates a classification score indicating the detected object's category, while the second fully connected layer provides the object's exact coordinates within the image.

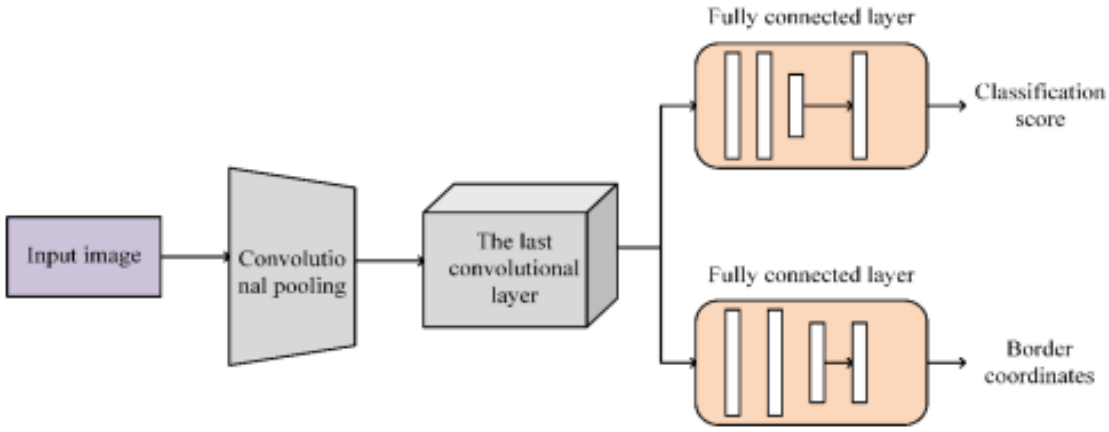


Figure 4.5 Object Detection Workflow

To enhance the effectiveness of object detection, numerous deep learning models have been introduced, including R-CNN [19] and its subsequent iterations such as Fast R-CNN [20], Faster R-CNN [21], and regression-based techniques like YOLO and SSD. This section is dedicated to offering a comprehensive overview and analysis of the Faster R-CNN model, a type of deep learning network. Additionally, suggestions for improvements within the context of this framework will be discussed.

4.2.1 Faster R-CNN Model

Faster R-CNN [21] is a significant improvement over its predecessor, Fast R-CNN [20]. Instead of using the traditional selective search algorithm to extract possible object regions, Faster R-CNN uses the RPN (Region Proposal Network) [22] to identify candidate regions. The RPN system delivers a mix of candidate regions with varying qualities, inspired by the region-wise feature pooling from SPP-Net [23]. This strategic change enhances the network's recall performance, recognition rate and precision.

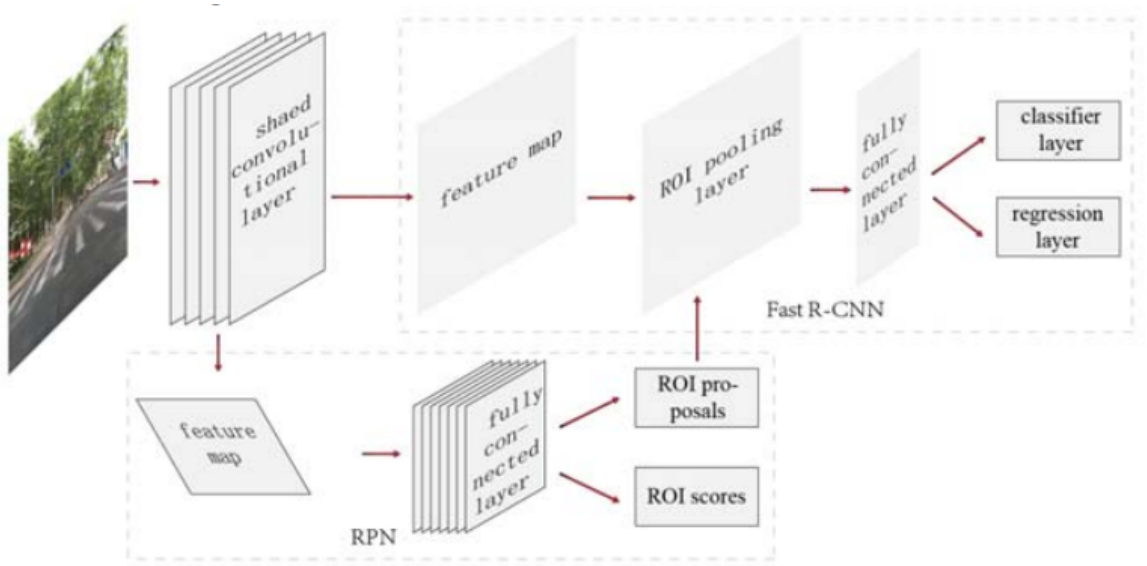


Figure 4.6 Process Flow of Faster R-CNN

The process structure of Faster R-CNN is shown in Figure 4.6, and the specific process is described below:

1. When an image is fed into the CNN network, it results in the derivation of a convolu tional layer. This convolutional layer serves a dual purpose: it acts as input for the RPN network and is also directed towards a specific, fixed convolutional layer to produce a more intricate convolutional layer feature map.

2. The RPN network delivers both scores and corresponding region recommendations. These scores are further refined using non-maximum suppression (NMS) [24] with a threshold of 0.7. This process yields fewer than 300 high-quality candidate frames, which are then standardized in size through the ROI pooling layer.

3. The ROIs selected in step 2 and the high-dimensional feature map from step 1 are directed into the ROI pooling layer, which enables the extraction of region-specific features.

4. The candidate features from step 3 are fed into the fully connected layer, which computes scores and boundary regression for each candidate region. This involves a joint training process, combining soft-max loss and smooth L1 loss to handle both classification and boundary regression.

RPN (Region Proposal Network) uses bounding box regressions and anchors, also known as candidate boxes. It slides over the final convolutional layer to generate candidate regions

at multiple scales. Figure 4.7 provides a visual representation of the RPN network structure. The underlying network architecture is ZF [25], and the output dimension of the conv5 layer is 256, corresponding to 256 feature maps. For each sliding window, RPN predicts two candidate regions. The classification layer outputs two scores, representing the likelihood of an object being present, and provides four sets of coordinates that correspond to the object's boundary.

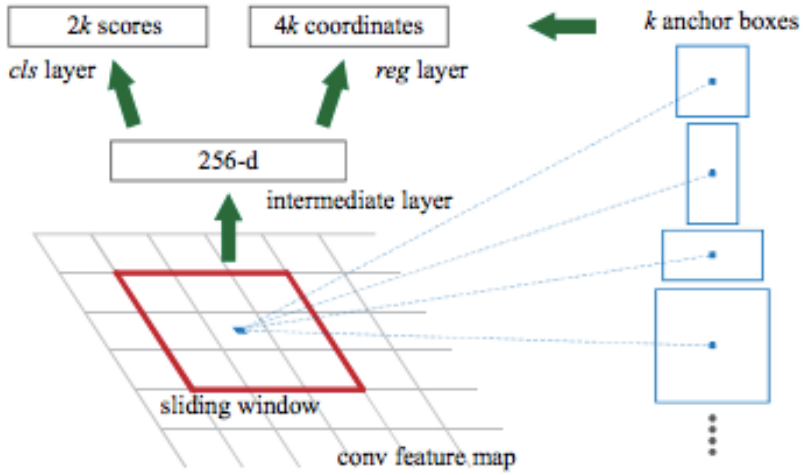


Figure 4.7 Region Proposal Network

4.3 YOLO Model

4.3.1 Development of YOLO Family

Divvala, Redmon, Farhadi, and Girshick (2016) created the You Only Look Once (YOLO) algorithm [26], which is a superb regression algorithm for deep learning object detection. One example of a single-stage algorithm is YOLO. Over time, the YOLO team made improvements to the algorithm and released YOLOv2 [27] and YOLOv3 [28]. Researchers released YOLOv4 in April 2020 [29], which greatly enhanced its functionality. More than a month after YOLOv4 was released, YOLOv5 was made available on the GitHub platform, and YOLOv5.1.0 was made available on the official website in June.

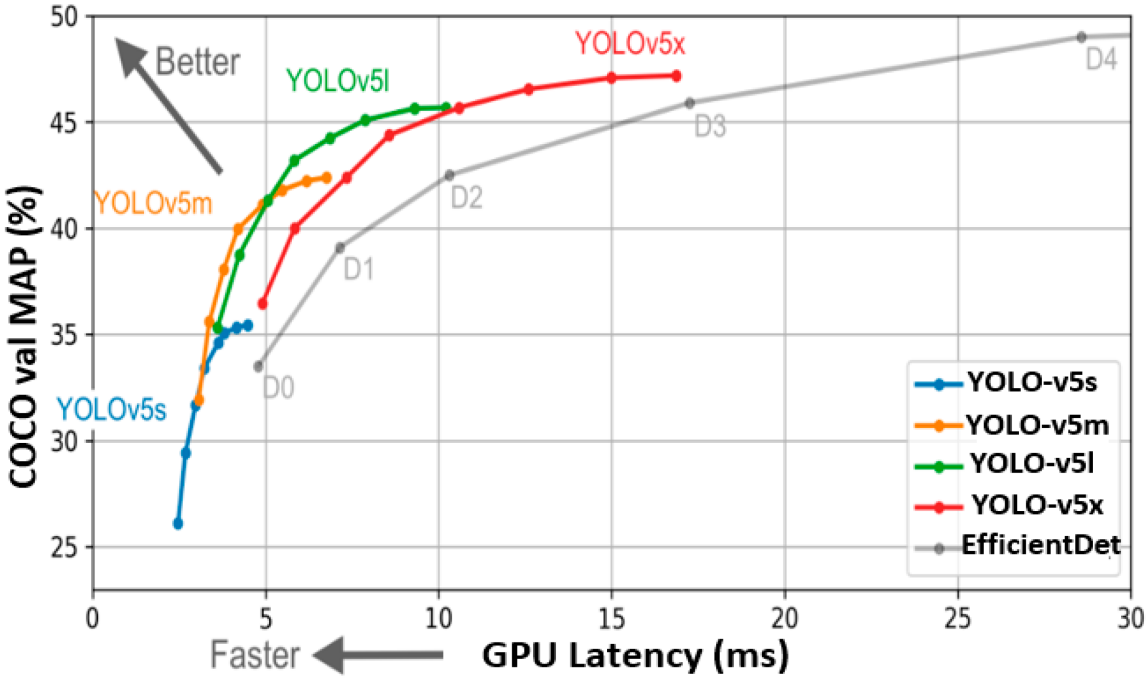


Figure 4.8 Comparison between each version of YOLOv5

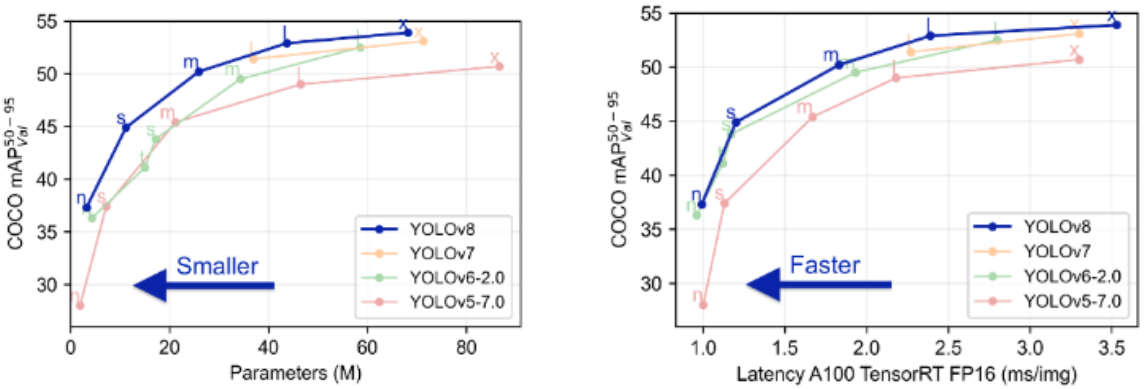
YOLOv5 has four net structures of various depths and widths. A comparison of map and reasoning speed between YOLOv5 and EfficientDet on the COCO data set is shown in Figure 4.8. We can see that YOLOv5 excels in both speed and precision, making it a highly performing algorithm.

In June 2023, one year after the release of YOLOv5, Meituan researchers introduced a su perior iteration called YOLOv6. YOLOv6 demonstrated notable performance improvements when benchmarked against the MS COCO data set.

Just a month later, the same research team introduced YOLOv7, also known as MT-Y OLOv6. This model established a new state-of-the-art for real-time object detection, show casing significant advancements in the field.

Finally, the most recent version, YOLOv8, was launched on January 10, 2023. Devel oped by Ultralytics, the team behind YOLOv5, this version introduces several innovations, including a new backbone network, a design that simplifies performance comparisons with previous YOLO models, a novel loss function, and an anchor-free detection head.

As shown in the Figure below, with content similar to Figure 4.8, the graph compares YOLO versions from v5 to v8:



(a) Number of parameters in different versions of YOLO architecture

(b) Inference speed for each YOLO version

Figure 4.9 Analysis of YOLO

4.3.2 YOLOv8 for Traffic Sign Recognition

Unlike some counterparts, YOLO doesn't involve a regional proposal step. In contrast to traditional Convolution Neural Networks (CNNs), YOLO streamlines the task into a single regression problem, whereas methods like Faster R-CNN require handling both classification and regression separately. This streamlined approach makes YOLO exceptionally fast and efficient, making it suitable for real-time applications like autonomous driving and video surveillance. YOLOv8 is an evolution of the YOLOv5 framework, introducing a range of enhancements in terms of architecture and developer usability. This new version offers improved speed and accuracy compared to YOLOv5 and unifies the framework for training models capable of object detection, instance segmentation, and image classification.

The architecture of YOLOv8 improves upon its predecessors, employing a convolutional neural network with two key components: the backbone and the head. The backbone utilizes a modified CSPDarknet53 architecture, consisting of 53 convolutional layers and incorporating cross-stage partial connections to enhance information flow between layers. In the head, there are multiple convolutional layers followed by fully connected layers that predict bounding boxes, object scores, and class probabilities for detected objects. YOLOv8 distinguishes itself by introducing a self-attention mechanism in the head, allowing the model to focus on different regions of the image and adjust the importance of features based on task relevance. Additionally, the model excels in multi-scaled object detection through a feature pyramid network, which includes multiple layers detecting objects at different scales, facilitating the identification of both large and small objects within an image.

Here is the framework of YOLOv8:

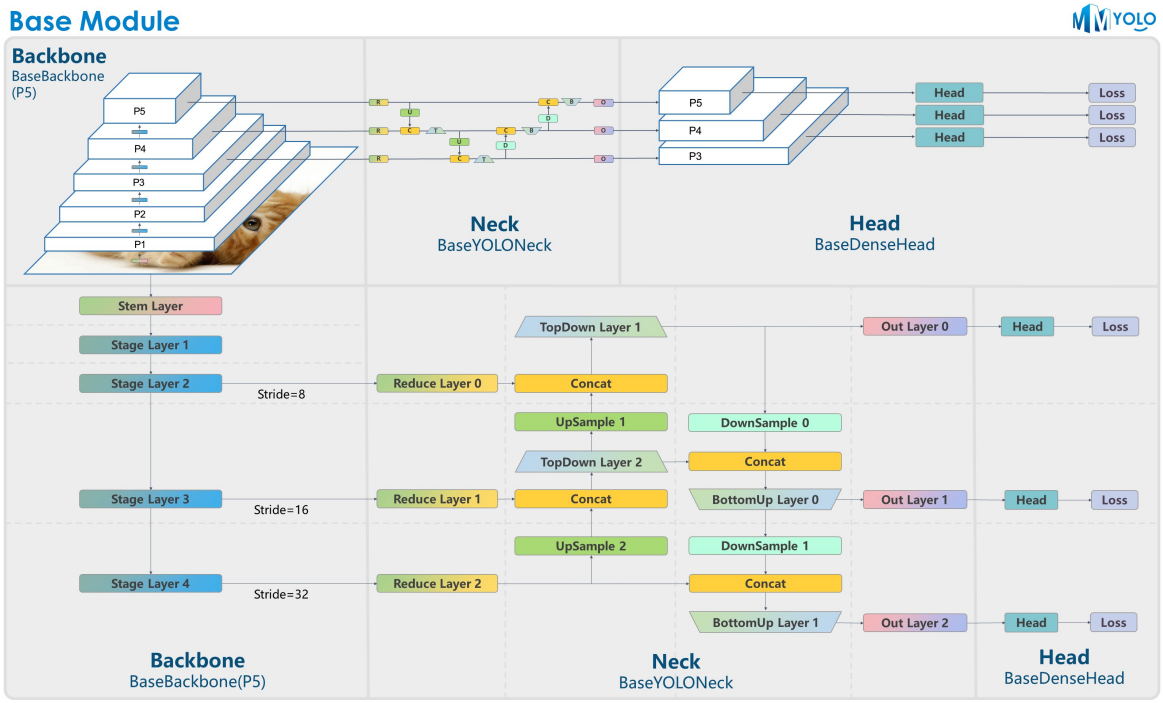


Figure 4.10 YOLOv8 Framework

4.4 Classification Model

In the 1980s, Yann LeCun, a postdoctoral computer science researcher, developed the pioneering Convolutional Neural Network (CNN) with a focus on recognizing handwritten digits. Named LeNet in honor of its creator, this initial CNN featured a straightforward architecture, consisting of five layers of 5x5 convolutional layers and 2x2 max-pooling. Despite its simplicity, the broader adoption of CNNs faced challenges due to factors like the need for extensive training data and computational resources. Additionally, the basic architecture limited its effectiveness with low-resolution images.

The pivotal moment for CNNs in computer vision emerged in 2012 with the introduction of AlexNet, a sophisticated CNN that leveraged GPUs for training. AlexNet's diverse array of kernels and impressive performance outpaced non-neural models, leading to its victory in the ImageNet Challenge 2012.

In the realm of deep learning, Convolutional Neural Networks (CNNs) have become prominent for processing and interpreting visual data. Unlike traditional neural networks relying on matrix multiplications, ConvNets introduce convolution as a distinctive operation. Mathematically, convolution involves operating on two functions, resulting in a third function that illustrates the transformation of one function's shape by the other.

A CNN typically consists of three essential layers: the convolutional layer, the pooling layer, and the fully connected (FC) layer. The convolutional layer initiates the network, while the FC layer constitutes the final stage.

The progressive complexity of a CNN from the convolutional layer to the FC layer allows it to systematically recognize larger segments and more intricate features within an image, ultimately leading to the identification of complete objects.

The convolutional layer plays a vital role in a CNN, serving as the focal point for most computations. Within this layer, a kernel or filter systematically moves across the image's receptive fields, scanning for the existence of distinctive features. This process involves multiple iterations, where the kernel traverses the entire image, performing a dot product operation between the input pixels and the filter. The outcome is a feature map or convolved feature, essentially transforming the image into numerical values. This transformation facilitates the CNN's ability to interpret the input data and extract meaningful patterns that contribute to its overall understanding of the visual information.

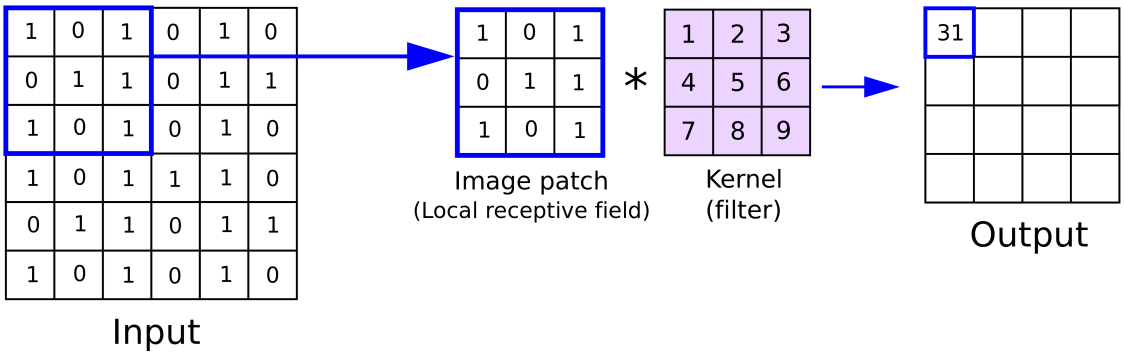


Figure 4.11 The Convolutional Layer

The Pooling layer plays a pivotal role in reducing the spatial size of the convoluted feature, a critical step in minimizing the computational power required for data processing by decreasing dimensions. Two prevalent types of pooling, namely average pooling and max pooling, are commonly employed in this process.

In Max Pooling, the method entails identifying the maximum value of a pixel within a specific region of the image covered by the kernel. Max Pooling serves as a noise suppressor, eliminating noisy activations and contributing to denoising while concurrently achieving dimensionality reduction.

On the other hand, Average Pooling calculates the average of all values within the designated section of the image covered by the kernel. While the primary objective of Average

Pooling is dimensional reduction, it also functions as a noise-suppressing mechanism. It is noteworthy that Max Pooling often exhibits superior performance compared to Average Pooling, particularly in its effectiveness as a noise-suppressing technique.

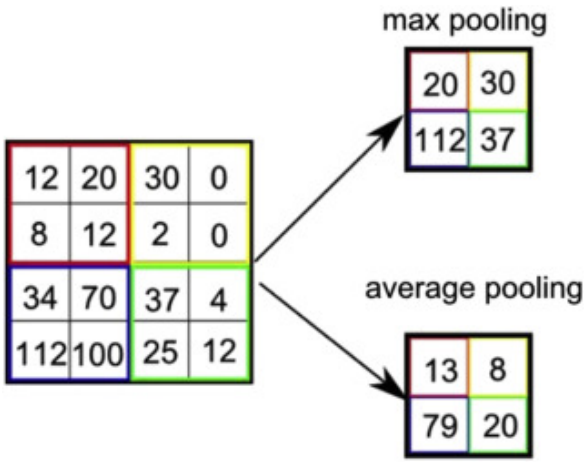


Figure 4.12 The Pooling layer

The FC layer is the stage where image classification takes place, leveraging the features extracted in earlier layers. In this layer, the term "fully connected" indicates that all inputs or nodes from one layer form connections with each activation unit or node in the following layer. However, not all layers in the CNN are fully connected, a design choice made to prevent excessive network density, manage losses, maintain output quality, and avoid computational inefficiencies.

4.4.1 VGG 16 and 19 Model

VGG16, a convolutional neural network devised by K. Simonyan and A. Zisserman at the University of Oxford, is frequently denoted as the VGG model. Described in the paper "Very Deep Convolution Networks for Large-Scale Image Recognition," this architecture is composed of 16 layers [25]. It gained acclaim for achieving an impressive top-5 test accuracy of approximately 92.7% on the ImageNet dataset, which comprises over 14 million images spanning nearly 1000 classes. VGG16 prominently participated in the ILSVRC-2014 competition.

Distinguishing itself from its predecessors, VGG16 replaces larger kernel-sized filters with a sequence of smaller 3×3 kernel-sized filters, leading to substantial improvements compared to AlexNet. This design choice enhances the network's ability to capture intricate features and patterns in images.

The training of VGG16 involved leveraging NVIDIA Titan Black GPUs over several weeks, underscoring the computational demands associated with deep learning tasks. Fea turing 16 layers, VGGNet-16 excels in classifying images across 1000 object categories, encompassing a diverse range of items such as keyboards, animals, pencils, mice, and more. Notably, the model processes images with a standard input size of 224-by-224 pixels.

The VGG network, particularly the VGG-16 model, is characterized by its utilization of small 3×3 convolution filters, constituting a total of 13 convolution layers and 3 fully connected layers:

- **Input**: VGGNet is designed for a 224×224 image input size. During ImageNet, creators cropped the central 224×224 patch from each image for consistent processing.
- **Convolution layers**: VGG's convolution layers employ a compact receptive field size of 3×3, capturing essential up/down and left/right information—the smallest size capable of this. Furthermore, 1×1 convolution filters serve as linear transformations, incorporating a ReLU unit introduced by AlexNet. This innovation reduces training time. ReLU, or rectified linear unit, is a piece-wise linear function outputting the input for positive values; otherwise, it yields zero. The convolution stride is consistently set at 1 pixel to maintain spatial resolution post-convolution.
- **Hidden Layers**: VGG's hidden layers uniformly utilize ReLU activation, and the network deliberately avoids employing Local Response Normalization (LRN). This decision is rooted in the observation that LRN tends to elevate memory consumption and training time without contributing to an enhancement in overall accuracy.
- **Pooling Layers**: After successive convolution layers, pooling layers play a pivotal role in reducing the dimensions and parameters of the feature maps generated during each convolutional step. This becomes especially crucial as the number of filters escalates rapidly, advancing from 64 to 128, 256, and ultimately reaching 512 in the concluding layers.
- **Fully-Connected Layers**: The VGGNet consists of three fully connected layers. The first two layers have 4096 channels each, while the third layer has 1000 channels, corresponding to one channel for each class in the classification task.

VGG19 is an extension of VGG16 with 19 weight layers, featuring 16 convolutional layers and three fully connected layers. Similar to VGG16, it employs 3x3 filters in the convolution layers and 2x2 max pooling layers. VGG19 also incorporates five max-pooling layers in its

architecture. This design contributes to the model's ability to capture intricate features in the input data through a hierarchical series of convolution and pooling operations.

For this project, we modified the model input to 32x32x3.

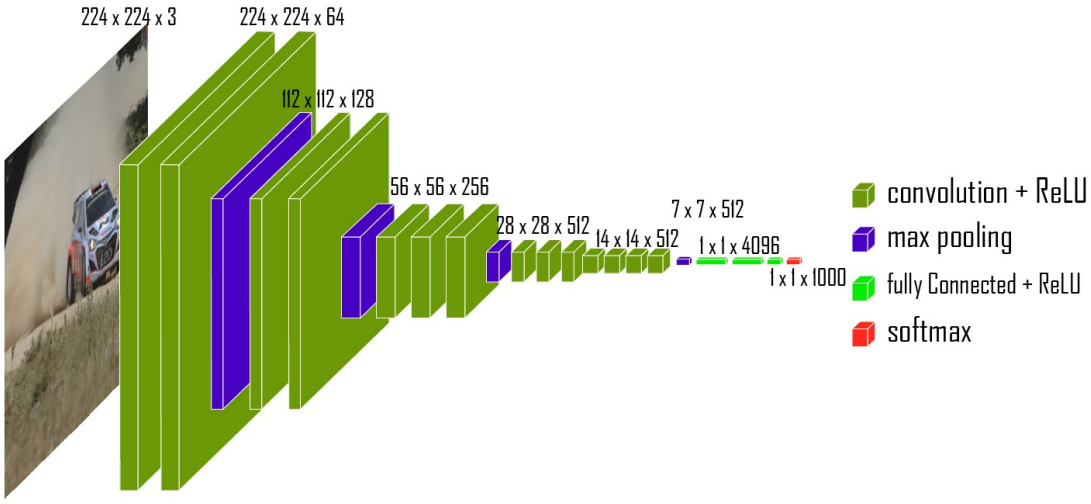


Figure 4.13 The VGG-16 architecture

4.4.2 ResNet50

The Residual Network, commonly known as ResNet, was introduced in a pivotal 2015 research paper titled "Deep Residual Learning for Image Recognition" by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun [22]. This innovative neural network achieved remarkable success, securing first place in the ILSVRC 2015 classification competition with an impressively low error rate of 3.57%. It also emerged victorious in the 2015 ILSVRC and COCO competitions across various tasks, including ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

The initial ResNet architecture, ResNet-34, comprised 34 weighted layers. Unlike VG GNet, which utilized 3×3 filters in each convolutional network, ResNet adopted a simpler design with fewer filters. The ResNet architecture adheres to two key design principles: each layer maintains the same number of filters based on the size of the output feature map, and if the feature map's size is halved, the number of filters is doubled to preserve temporal complexity.

Numerous ResNet variations exist, differing in the number of layers while adhering to the same design principles. One such variant is ResNet50, designed to operate with 50 neural network layers. For ResNet50 and higher versions, a minor modification was introduced:

shortcut connections, which used to skip two layers, now skip three layers. Additionally, a 1x1 convolution layer was added.

In our project, a notable modification was made by adjusting the model input to 32x32x3, showcasing the flexibility of ResNet in handling diverse input sizes.

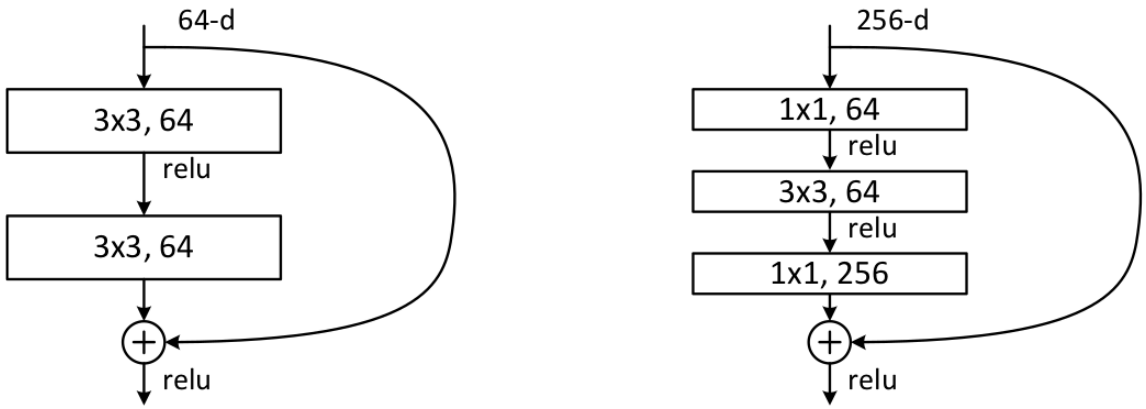


Figure 4.14 ResNet50 shortcut connections

The ResNet 50 architecture contains the following elements:

- A convolutional layer with a 7x7 kernel employs 64 additional filters and a stride of 2.
- A max pooling layer is incorporated with a stride size of 2.
- 9 additional layers, comprising three sets of repetitions: one with 1x1 dimensions and 64 kernels, another with 3x3 dimensions and 64 kernels, and a third with 1x1 dimensions and 256 kernels.
- An extra 12 layers, organized into four iterations of the following sequence: a layer with 1x1 dimensions and 128 kernels, followed by a layer with 3x3 dimensions and 128 kernels, and finally, a layer with 1x1 dimensions and 512 kernels.
- 18 additional layers, and these consist of six iterations of the following pattern: a layer with 3x3 dimensions and 256 cores, another layer with 1x1 dimensions and 1024 cores, and a final layer with 1x1 dimensions and 256 cores.
- An additional 9 layers, with three repetitions of the following pattern: a layer with 1x1 dimensions and 512 cores, followed by a layer with 3x3 dimensions and 512 cores, and concluding with another layer of 1x1 dimensions and 2048 cores.
- Initially, average pooling is applied, followed by the utilization of the softmax activation function to establish a fully connected layer comprising 1000 nodes.

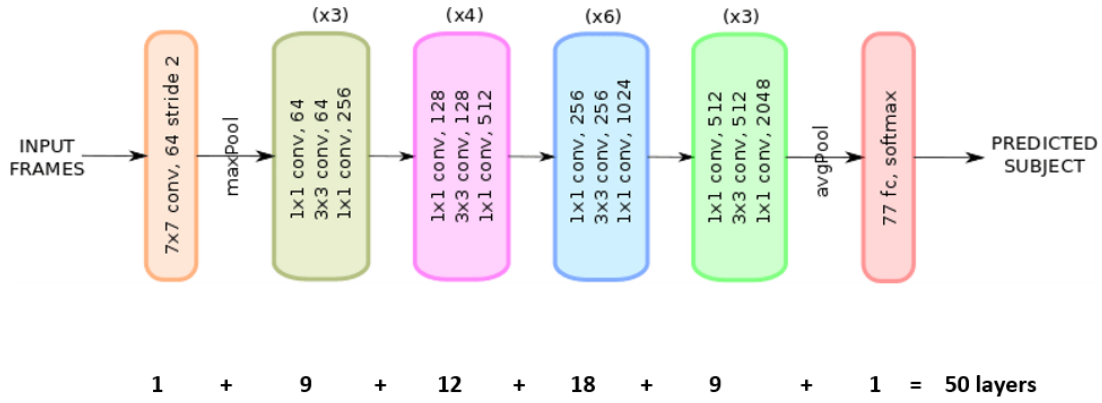


Figure 4.15 ResNet50 architecture

4.4.3 LeNet Model

The Lenet-5 model, introduced in 1998 by Yann LeCun and collaborators, is a pioneering pre-trained model known for its simplicity and effectiveness in handwritten and machine-printed character recognition. This multi-layer convolutional neural network (CNN) has five layers with learnable parameters, making it a foundational milestone in the development of CNNs for visual recognition.

The architecture of Lenet-5 consists of three sets of convolution layers interspersed with average pooling operations, allowing it to extract hierarchical features from input images. Two fully connected layers follow the convolution and pooling layers, enabling complex pattern recognition. The model concludes with a Softmax classifier for image categorization.

In our modified Lenet-5 model, it processes 32x32 RGB images with three channels for input data. The initial convolution operation involves sixty 5x5 filters, resulting in feature maps of size 28x28x60 and 24x24x60. After the first convolution, a max pooling operation is applied, halving the feature map size while maintaining the number of channels. The second convolution layer uses thirty 3x3 filters, producing feature maps of dimensions 10x10x30 and 8x8x30. Another max pooling layer follows, reducing the feature map to 4x4x30.

The output is then flattened to produce 480 values. A fully connected layer with 500 neurons is employed, and the final output layer with 29 neurons corresponds to the twenty-nine classes in the dataset. This modified Lenet-5 model is designed for image classification tasks, showcasing its adaptability and continued relevance in the field.

Here is the final architecture of the Lenet-5 model:

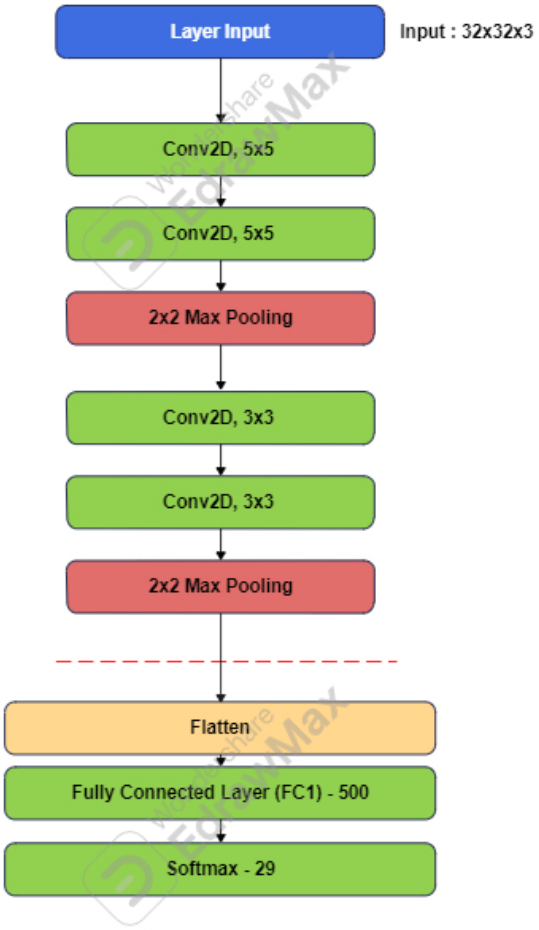


Figure 4.16 The LeNet architecture

4.5 Our Methods

For our project, we also decided to develop our own method on handling this problem, by having a two jointed model, in which the first model is the detection model, and the second model will be the classification model. The main reason is because after we tested on several dataset with multiple different value of classes, we notice that as the class have a small value, mainly around 3-4, as they only recognise the shape of the sign, the mAP value is much

higher than theirs of the 30-40 classes, as the model must be able to recognise that kind of particular sign.

For the first model, after testing with multiple models, we decided that using YOLO is one of the best choices for traffic sign detection, as they are fast and have a high mAP value. They are also much more up to date than the Faster R-CNN model.

As for the classification layer, we currently have two models on our mind, LeNet and MicroNet, both have a high performance value for this problem, and we want to test both of them if they are capable of running in this type of architecture.

The way this method processes data is the data (mainly image) is input into the first model, and the detection model will point out where the sign is, and make a bounding box at the position of the sign. After that, the image will be reduced into the part that has the bounding box, and run through the second model, where the sign will be classified into a specific category, and output the prediction of the sign. The process can be viewed in the figure below.

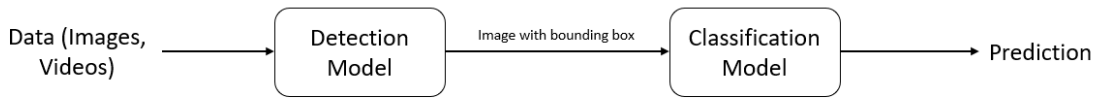


Figure 4.17 Our Processing Method for TSR

Finally, we decided to choose to mix several models together, as we want to show out the best combination to test in a real-time situation. All of these values that help us to pick out these models will be informed in the *Result* section.

4.6 Evaluation Methods

In this section, we employ a range of evaluation techniques to assess our model's performance. In the context of Object Detection tasks, models are typically evaluated based on mAP (mean Average Precision). Before delving into the concept and calculation of mAP, we need to understand the relevant terms and concepts.

4.6.1 IoU (Intersection over Union)

The Intersection over Union (IOU), also known as the Jaccard index, serves as a metric to assess the accuracy of an object detector when applied to a dataset. To formally employ IOU for evaluating any object detector, two crucial components are required:

- Ground-truth bounding boxes: which are manually annotated bounding boxes derived from the test set, indicating the actual location of the object within the image.
- Predicted bounding boxes: are bounding boxes generated by our model during the detection process.

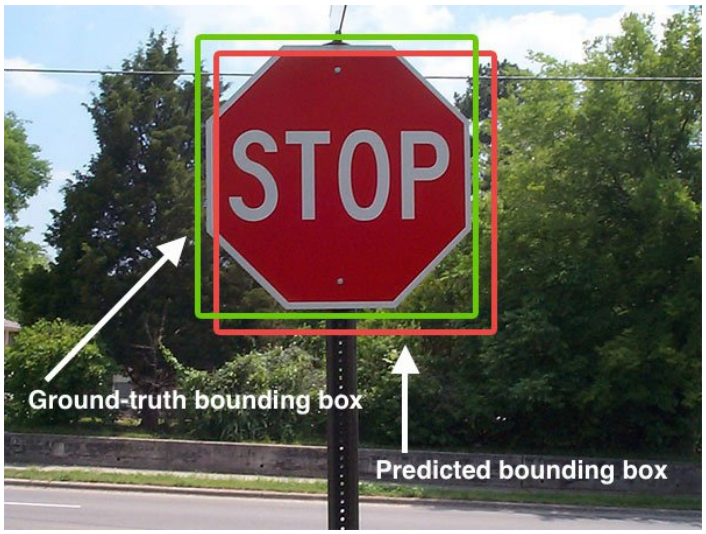


Figure 4.18 Example of predicted bounding box with ground-truth bounding box

With these two sets of bounding boxes as illustrated in 4.18, we can calculate Intersection over Union:

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$

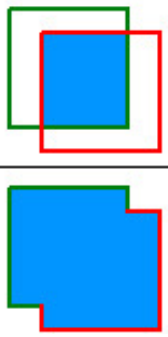


Figure 4.19 Formula of IoU

In which:

- Area of Overlap: This denotes the region where the predicted bounding box and the ground-truth bounding box intersect.

- Area of Union: This represents the collective area encompassed by both the predicted bounding box and the ground-truth bounding box.

For a given problem, an IOU threshold is usually established, ranging from 0 to 1. A prediction is deemed accurate if the IOU exceeds the specified threshold, commonly set at 0.5 in many cases.

The confusion matrix comprises four essential elements: True Positive (TP), True Neg ative (TN), False Positive (FP), and False Negative (FN). To determine these components accurately, it is crucial to define a threshold (referred to as α) based on Intersection over Union (IoU).

- True Positive (TP) - This corresponds to a case where the classifier correctly identifies a positive instance when the actual truth is indeed positive. In other words, it's a detection for which the Intersection over Union (IoU) $\geq \alpha$.
- False Positive (FP) - Type I Error - This represents an incorrect positive detection, where the classifier predicts a positive instance, which the IoU $< \alpha$.
- False Negative (FN) - Type II Error - This is when the classifier fails to detect an actual positive instance.
- True Negatives (TN): In object detection, the concept of True Negative is less relevant because there are many potential predictions that should not be considered detections in an image. Therefore, TN includes all the incorrect predictions that were not detected, as there can be various ways to make incorrect predictions that are not directly comparable to TP, FP, or FN.

4.6.2 Precision and Recall

Precision - assesses a classifier's capability to correctly identify relevant objects. It represents the ratio of accurate positive predictions which is given by

$$\text{Precision} = \frac{TP}{TP+FP} \tag{4.1}$$

Recall - measures a classifier's ability to locate all relevant cases, which means finding all the ground-truth instances. It is the proportion of true positives identified among all the ground-truth instances and is defined

$$\text{Recall} = \frac{TP}{TP+FN} \tag{4.2}$$

4.6.3 Precision Recall Curve and Average precision(AP)

Precision and Recall values vary with different Confidence thresholds. To visualize the corresponding Precision and Recall values at various thresholds, we utilize a Precision-Recall Curve, which links all the data points containing (recall, precision) pairs for each threshold.

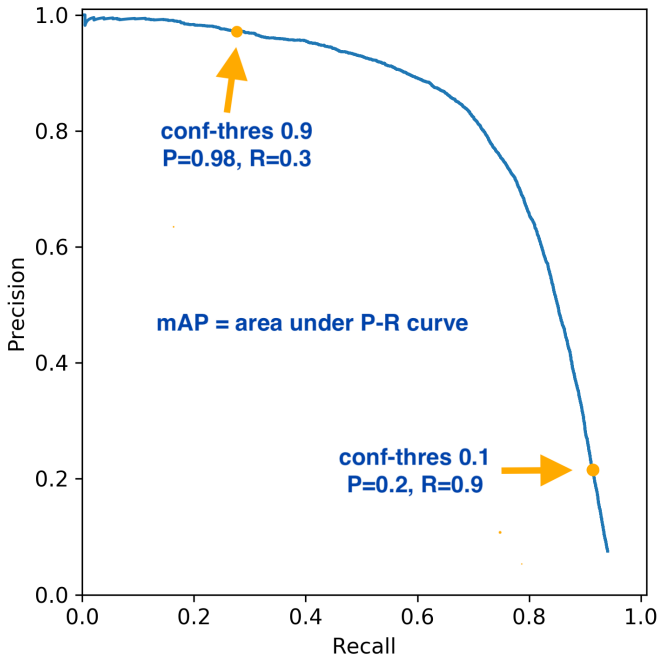


Figure 4.20 Precision Recall Curve

The AUC (Area Under the Curve) serves as a metric utilized in assessing model perfor mance. Specifically, within the Precision-Recall Curve context, the AUC is alternatively referred to as Average Precision (AP). The calculation of AP involves the following formula:

$$AP = \sum_{k=1}^{n} (R_k - R_{k-1}) * P_k$$

Figure 4.21 Average precision

Where R_k and P_k are Recall and Precision values corresponding to threshold k and n is the total number of thresholds.

A high AP suggests a large AUC, indicating that the curve tends to be closer to the upper right corner, signifying that both Precision and Recall are relatively high across various thresholds. This signifies that the model is performing well.

Conversely, a low AP implies that both Precision and Recall are relatively low, indicating poor model performance.

And in the context of Object Detection, whether it's YOLO or any other method, mAP (mean Average Precision) is defined as the average of the individual Average Precision (AP) values for all classes. It can be expressed as:

$$mAP = \frac{1}{n} \sum_{i \in C} AP(i)$$

Figure 4.22 mean Average precision

Where C is is the set of all classes and n is the total number of classes.

A higher mAP suggests that most individual AP values for different classes are high, indicating a better-performing model. This is why mAP is an ideal metric for evaluating a model because it captures the overall performance across all classes. Consequently, when training a model, the goal is to maximize the mAP, which implies achieving the highest possible individual AP values for each class.

Lastly, we use milliseconds (ms) to measure the speed of our models. The faster our models can process an image, the faster they can operate in real-time. Achieving such speed calls for optimized hardware and software solutions, and it is an important factor for developers and users who want to deliver smooth and immersive real-time applications.

Chapter 5

Result

In this chapter, we will give out information about our testing on the model, which will give out the reason why we choose and put it to use in the project. We will also give out detailed information of our collected dataset, and announce the result of our tested model, and its performance on it.

5.1 Our Dataset

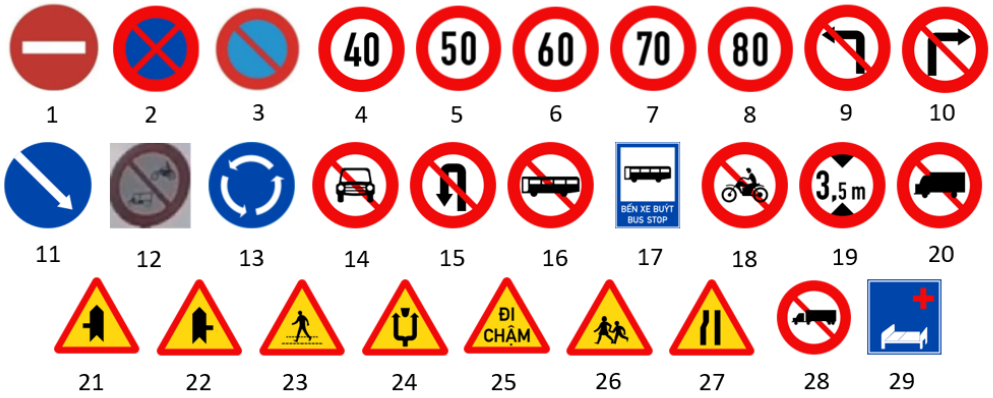


Figure 5.1 Dataset's classes

As shown in the project register, we also wanted to make a traffic sign dataset, and we are able to finish it. We recorded these data on the 18th of October, 2023, at around 11AM to 4PM, using a GoPro 10, and they were recorded in several locations, most of them were in District 3, District 1 and District 9 of Ho Chi Minh City. The dataset includes 1088 images, which contain 2051 annotations, divided into 29 classes in total. These images were carefully

sorted within approximately 22000 frames recorded, and were labeled by hand using the Labelbox application.

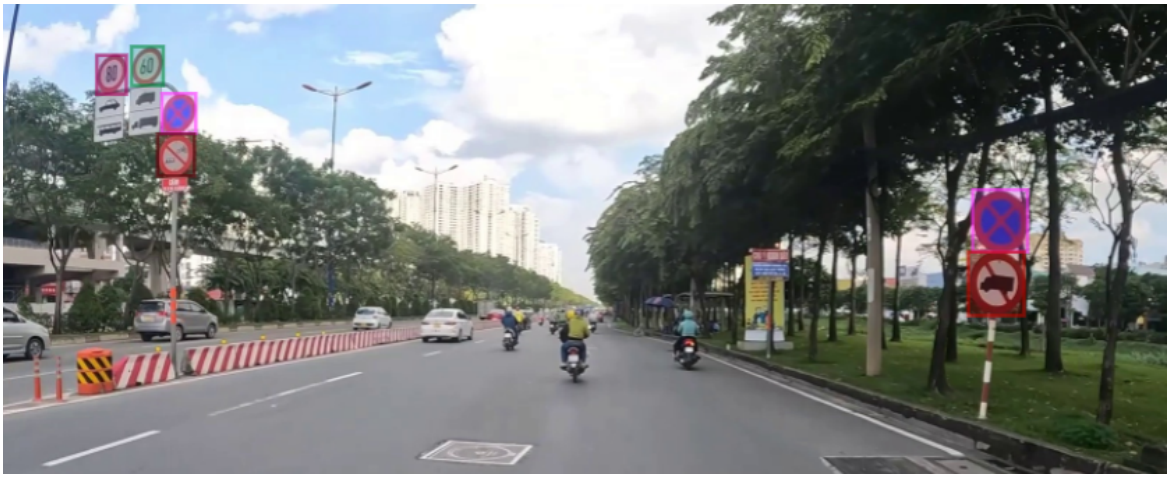


Figure 5.2 Example of a labeled image

Here is the list of classes and their quantities that we gathered:

Name of Traffic Sign	**Quantity**	**Name of Traffic sign**	**Quantity**
No stopping or parking	349	Zebra crossing / crosswalk ahead	41
No parking	182	No buses	37
Do not enter	162	Maximum speed 80 km/h	35
Maximum speed 60km/h	157	Roundabout ahead	34
Keep right	134	No cars	33
No 2-3 wheel vehicles	98	Height restriction	31
Slow down	90	Maximum speed 70 km/h	30
No right turn	88	Hospital nearby	27
No heavy vehicles	75	School zone ahead	22
No trailers	70	No motorcycles	20
Right road junction with priority	57	Maximum speed 50 km/h	19
No U-turn	55	Traffic obstruction ahead - may pass on either side	18
Maximum speed 40 km/h	53	Left road junction with priority	16
No left turn	53	Road narrows ahead on the left side	15
Bus stop	50		

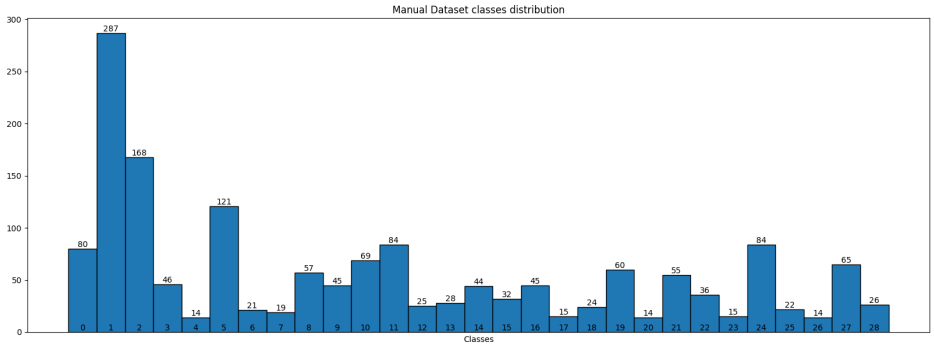


Figure 5.3 Detection dataset's distribution

For the classification model, we gather images of cropped traffic signs from the internet. We gathered about 28000 images, consisting of 29 classes as above.

Here our classification dataset's distribution:

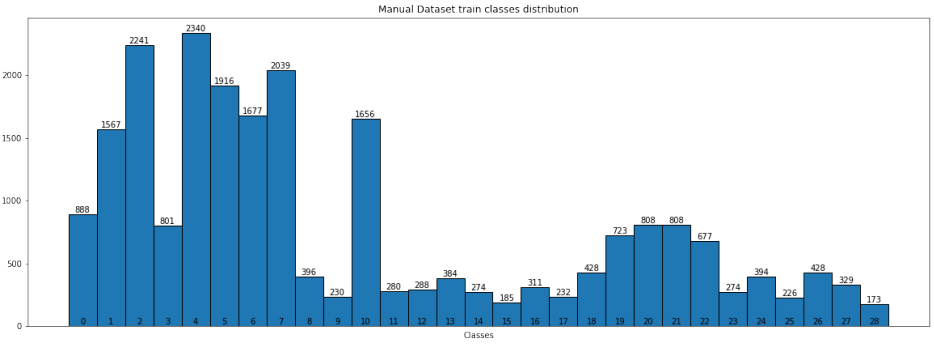


Figure 5.4 Classification dataset's train distribution

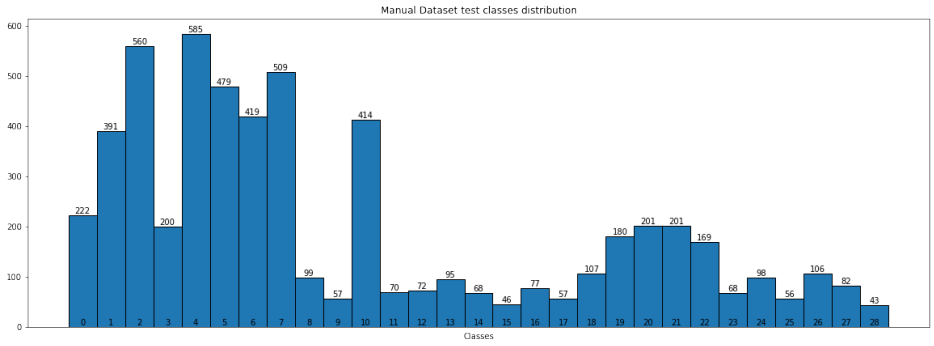


Figure 5.5 Classification dataset's test distribution

Figure 5.6 Some examples of our classification dataset images

5.2 Testing on models

In our process to select the perfect model for the TSR problem, we wanted to test if a model could able to detect and classify at the same time, as our plan at first was to have 2 core idea: make a model that could run both detection and classification, or stack 2 model to do that, and make it so that it could run on real time.

After this, we try to test out on the two main models that were recommended throughout our research on papers before: YOLO and Faster R-CNN on the GSTDB dataset. Both of which was able to give out several different result, base on the number of class we tested with it:

Model	Number of classes	Epoch	mAP@50	Parameters (M)	Speed GPU (ms)
Faster R-CNN	4	100	70.6	42.0	183.6
YOLOv5	4	100	88.0	7.2	12.4
Faster R-CNN	43	100	49.4	42	197.8
YOLOv5	43	216	40.0	7.2	43.7

Table 5.1 Testing detection model on GSTDB

So, as the data suggested, the more classes got put into the detection model, the lower their mAP rating was. This is crucial as this could be used to point out if we want to do the stacking model plan or not. And this shows that if we want to have an AI model that is able to do both detection and classification, the detection part must run at high precision, for it to go to the next part, classification. This is when we thought of building a joint model for this project. The number showed that YOLO is one of the better models, as the model shows its strength and handling the detection problem much better than Faster R-CNN when it comes to a smaller number of classes.

After we were able to pick YOLO for the detection problem, we decided to test with other several YOLO models, to see if we would be able to get a better model, as the latest version of YOLO, YOLOv8, was released on the 10th of January at the beginning of this year. Here are the results when testing on the GSTDB dataset:

Model	Number of classes	Epoch	mAP@50	Parameters (M)	Speed GPU (ms)
YOLOv5s	4	100	88.0	7.2	12.4
YOLOv7	1	300	98.5	36.9	26.1
YOLOv8m	1	100	99.4	25.9	28.6
YOLOv8n	1	100	98.1	3.2	11.4

Table 5.2 Testing on GTSDB to choose the best YOLO model version

It is worth mentioning here that after the second review of the thesis, our two review ers/teachers, Nguyen Quoc Trung and Nguyen Trong Tai, pointed out that we should use only one class when it comes to detection, as a group all the data into one "traffic sign" class, instead of separating them into groups of 4, which we originally wanted to classify them into the type of traffic sign, at the first place. This helped a lot in enhancing the model, as their mAP@50 value made quite a jump, from 0.88 to their lowest base of 0.98. And as the graph has shown, YOLOv8m gives out a high mAP value of 0.994. However, one of the pre-train models, YOLOv8n, also gives out quite the result, scoring 0.981 on the mAP value, but can process the data faster.

Model	Size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A0 TensorRT (ms)	Params (M)	FLOPs
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Table 5.3 YOLOv8 models' statistics

For more information, YOLOv8n is one of the 5 pre-trained detection models, using the COCO (Common Object in Context) dataset. According to these statistics of the model, which was taken at the YOLOv8 GitHub page, YOLOv8n even though it has the lowest mAP value of 37.3, it can put out a higher processing speed with a much lower number of parameters and operations (FLOPs). With our result on testing with the model, we decided to choose YOLOv8n as our detection problem solver, given the fact that speed is necessary for our project.

For testing on the classification part, we have several options, which are VGG16, LeNet, and MicroNet, which excel in recognition. Here are the results when testing on them using the GSTRB dataset:

Model	Epoch	Accuracy	Parameters (M)
LeNet	10	98.2	0.4
VGG16	20	97.6	15.0
VGG19	15	96.3	20.3
ResNet50	20	79.0	24.7

Table 5.4 Testing classification model

According to the results shown above, LeNet, even though it is the oldest model of all 3, can put out an accuracy value of 98% and we chose it as our main classification layer.

After testing on a separate model, we join our 2 models together, in which we take the best combination to compare with others' research paper's results. Here are the results when testing on them using the GSTDB dataset:

Model	Number of classes	mAP@50	Parameters (M)	Speed GPU (ms)
YOLOv5s+LeNet	43	98.0	7.6	39.4
YOLOv7+LeNet	43	97.0	37.3	53.1
YOLOv8m+LeNet	43	98.5	26.3	55.6
YOLOv8n+LeNet	43	98.1	3.6	38.4

Table 5.5 Testing jointed model

As we are able to choose the model structure to compare, which is the YOLOv8n + Lenet model, we began to collect the dataset, which will be mentioned in the next section.

5.3 Project's Model Testing

After testing on the GSTDB dataset, the model is able to give out the result and we wanted to compare it with several other models from various papers, which can be shown below:

We also test and compare our classification model on the GSTRB dataset with several other papers. The result is shown as follows:

Model's name	Classes	mAP@50	Speed (ms)	Research paper	Year
Faster R-CNN	4	91.7		[11]	2019
YOLOv3	4	89.7	60.6	[30]	2021
TFD Architecture	43	84.9	222.2	[31]	2022
YOLOv8n (our model)	1	98.1	11.4		2023
YOLOv8n + LeNet (our model)	43	98.1	38.4		2023

Table 5.6 Detection model's result

Model's name	Accuracy	Parameters (M)	Research paper	Year
HLSGD	99.6	23.20	[32]	2018
CDNN	98.5	1.54	[32]	2018
CNN	99.6	0.73	[33]	2019
Modified Lenet-5 (our model)	98.2	0.37		2023

Table 5.7 Classification model's result

As the statistics have revealed, just to compare the separate model alone, the YOLov8n model is really suitable for this project, as their mAP@50 value is staggeringly high, meaning it is able to do a great job of detecting traffic signs. It is the same with our LeNet classification model, as they can provide a high accuracy model without needing a large amount of parameters.

After comparing with other research papers' results, we joined the two models to test their abilities on detection and classification in a real-time situation. The result is shown below:

Dataset	Number of classes	mAP@50
GSTDB	43	98.0
Our dataset	29	90.7

Table 5.8 YOLOv8n + Lenet model's result

Looking at the result, it is safe to say that the jointed model can provide a high precision and still run with reasonable speed, as 38ms meaning the model is in a position to detect and provide results every 2-3 meters in distance, as the average speed of a moving car tend to be around 50 to 60 kilometers per hour. In addition, the mAP@50 value is quite close together, in the range of 90% and above, meaning that the model have a good generalization ability.

To improve our model result, we also try to apply histogram equalization to the image before classifying. Histogram equalization enhances contrast and helps extract more informa tion from an image. But in our case, it might enhance the background more instead of the sign. After testing, we noticed that the result did not improve but worsened.

Classification model input	Accuracy	Parameters (M)
32x32, grayscale	98.1	0.373
32x32, RGB	99.2	0.376
64x64, RGB	98.8	2.296
32x32, RGB with Histogram equalization	98.9	0.376
64x64, RGB with Histogram equalization	98.3	2.296

Table 5.9 Classification model result comparison

As the result has shown in Table 5.9, our LeNet model performs better with smaller input shapes and without pre-processing. Our theory is that because our classification model is simple with only 4 convolution layers, bigger input shapes might introduce too much noise that reduces our model's accuracy and the number of important features that it can learn.

Applying the same models to our dataset yields the expected result. Our combined model does worse with gray scale as the input for the classification model. After changing to color input, the result became much better.

Classification model input	mAP@50
32x32, grayscale	61.7
32x32, RGB	90.7
64x64, RGB	84.8
32x32, RGB + Histogram equalization	89.8
64x64, RGB + Histogram equalization	82.9

Table 5.10 Jointed model result comparison

Chapter 6

Discussion

As our results have been shown in the previous chapter, it would be best to say that we have passed all the objectives we pointed out since day one. With this, we can announce that our project can be used in Vietnam, with minor adjustments and improvements that can be made to support it in the future, especially the real-time perspective of this project.

The fact that there are many papers related to TSR, and we skim through many of them and just be able to find a partial solution to this problem, as models wanting to handle and solve this problem, need to have the ability to process both detection and classification. While this problem in real life, which can be implemented in self-driving cars, also needs to have the function to process it in real-time. Still, this proves that there is not much focus on this problem, and this could be a research category that many of us can come back to and improve in the future.

Chapter 7

Conclusion

Following the Artificial Intelligence trend, the need for a driver-less traffic condition is becoming more and more necessary, as reducing traffic accidents is one of many problems that artificial intelligence can reduce. Thus, a part of the problem, traffic sign recognition is also in great need. Our thesis is to make a dataset for traffic signs in Vietnam, and make a deep learning model that is able to detect and classify traffic signs in real-time, all of which are made in a controlled environment.

After finishing researching, we have come up with the idea of joining 2 models, a detection and classification, together, as using only 1 model that has to process both of them does not produce a high accuracy. We are able to come up with the best combination of models, YOLOv8n for detection and LeNet for classification.

We also offered a dataset for this project, which was collected in a sunny weather environment, and produced a set of 1088 images, that contained 2051 annotations, which can be categorized into 29 classes after sorting and labeling.

Bibliography

[1] J. Wu and L. Zhong, "A new data aggregation model for intelligent transportation system," vol. 671-674, 2013, pp. 2855–2859.

[2] L. Hazelhoff, I. M. Creusen, and P. H. de With, "Exploiting street-level panoramic images for large-scale automated surveying of traffic signs," *Machine Vision and Applications*, vol. 25, pp. 1893–1911, 9 2014.

[3] R. Timofte, K. Zimmermann, and L. Van Gool, "Multi-view traffic sign detection, recognition, and 3d localisation," vol. 25, 12 2009, pp. 1–8.

[4] C. Ai and Y. C. J. Tsai, "Critical assessment of an enhanced traffic sign detection method using mobile lidar and ins technologies," *Journal of Transportation Engineering*, vol. 141, 5 2014.

[5] D. Wang and J. Zhu, "Fast smoothing technique with edge preservation for single image dehazing," *IET Computer Vision*, vol. 9, pp. 950–959, 12 2015.

[6] S. Xiangbin, F. Xuejian, Z. Deyuan, and G. Zhongqiang, "Image classification based on mixed deep learning model transfer learning," *Journal of System Simulation*, vol. 28, no. 1, p. 167, 2016. [Online]. Available: https://www.china-simulation.com/EN/abstract/article_1590.shtml

[7] X. Ma, A. Fu, J. Wang, H. Wang, and B. Yin, "Hyperspectral image classification based on deep deconvolution network with skip architecture," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, pp. 4781–4791, 8 2018.

[8] C. Pan, M. Sun, Z. Yan, J. Shao, X. Xu, and D. Wu, "Vehicle logo recognition based on deep learning architecture in video surveillance for intelligent traffic system," 01 2013, pp. 132–135.

[9] R. Ravindran, M. J. Santora, M. Faied, and M. Fanaei, "Traffic sign identification using deep learning." Institute of Electrical and Electronics Engineers Inc., 12 2019, pp. 318–323.

[10] J. S. F. of Computer, I. Sciences, I. of Electrical, and E. Engineers, *ICICIS 2019 : Ninth IEEE International Conference on Intelligent Computing and Information Systems : Cairo, Egypt, December 8-9, 2019*, 2019.

[11] L. Wu, H. Li, J. He, and X. Chen, "Traffic sign detection method based on faster r-cnn," *Journal of Physics: Conference Series*, vol. 1176, no. 3, p. 032045, mar 2019. [Online]. Available: https://dx.doi.org/10.1088/1742-6596/1176/3/032045

[12] J. Wang, Y. Chen, Z. Dong, and M. Gao, "Improved yolov5 network for real-time multi-scale traffic sign detection," *Neural Computing and Applications*, vol. 35, pp. 7853–7865, 4 2023.

[13] Z. Qin and W. Q. Yan, "Traffic-sign recognition using deep learning," vol. 1386 CCIS. Springer Science and Business Media Deutschland GmbH, 2021, pp. 13–25.

[14] Jiang, Linfeng, Liu, Hui, Zhu, Hong, and Zhang, Guangjian, "Improved yolo v5 with balanced feature pyramid and attention module for traffic sign detection," *MATEC Web Conf.*, vol. 355, p. 03023, 2022. [Online]. Available: https://doi.org/10.1051/matecconf/202235503023

[15] D. Tabernik and D. Skocaj, "Deep learning for large-scale traffic-sign detection and recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, pp. 1427–1440, 4 2019.

[16] J. Zhang, W. Wang, C. Lu, J. Wang, and A. K. Sangaiah, "Lightweight deep network for traffic sign classification," *Annales des Telecommunications/Annals of Telecommunications*, vol. 75, pp. 369–379, 8 2019.

[17] K. Lim, Y. Hong, Y. Choi, and H. Byun, "Real-time traffic sign recognition based on a general purpose gpu and deep-learning," *PLoS ONE*, vol. 12, 3 2017.

[18] L. Abdi and A. Meddeb, "Deep learning traffic sign detection, recognition and aug mentation," vol. Part F128005. Association for Computing Machinery, 4 2017, pp. 131–136.

[19] D. Rumelhart, G. Hinton, and J. Mcclelland, "A general framework for parallel dis tributed processing," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, 01 1986.

[20] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083

[21] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[23] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013. [Online]. Available: http://arxiv.org/abs/1311.2901

[24] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6469–6477.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:14124313

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[27] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525.

[28] ——, "Yolov3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:4714433

[29] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[30] M. Manawadu and U. Wijenayake, "Voice-assisted real-time traffic sign recognition system using convolutional neural network," 07 2021.

[31] B. Vaidya and C. Paunwala, "Lightweight hardware architecture for object detection in driver assistance system," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 36, 02 2022.

[32] A. Wong, M. J. Shafiee, and M. St. Jules, "Micronnet: A highly compact deep convolu tional neural network architecture for real-time embedded traffic sign classification," *IEEE Access*, vol. 6, pp. 59 803–59 810, 2018.

[33] J. Zhang, W. Wang, C. Lu, J. Wang, and A. Kumar, "Lightweight deep network for traffic sign classification," *Annals of Telecommunications*, vol. 75, 07 2019.