زانکۆی پۆلیتەکنیکی هەولێر
ERBIL POLYTECHNIC UNIVERSITY

Improving Traffic Flow for Emergency Vehicles Using Deep Learning Techniques

A Dissertation

Submitted to the Council of the College of Erbil Technical Engineering College at Erbil Polytechnic University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Information System Engineering

By

Kamaran Hussein Khdir Manguri

B.Sc. in Computer Systems Engineering (2011)

M.Sc. in Electronics and Computer Engineering (2016)

Supervised by

Prof. Dr. Aree Ali Mohammed

Erbil, Kurdistan
August 2024

DECLARATION

I declare that the PhD. Dissertation entitled “Improving Traffic Flow for Emergency Vehicles Using Deep Learning Techniques” is my own original work, and hereby I certify that unless stated, all work contained within this dissertation is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgment is made in the text.

Signature:

Student Name: Kamaran Hussein Khdir Manguri

Date: August/ 2024

CERTIFICATE OF PROOFREADING

This is to certify that this dissertation entitled: “Improving Traffic Flow for Emergency Vehicles Using Deep Learning Techniques” written by the postgraduate student (Kamaran Hussein Khdir Manguri) has been proofread and checked for grammatical, punctuation, and spelling mistakes. Therefore, after making all the required corrections by the student for further improvement, I confirm that this last copy of the dissertation is ready for submission.

Signature:
Name: Asst. Prof. Dr. Salih Ibrahim Ahmed
Phone No.: 07701502771
Email Address: salih.ahmed@uor.edu.krd
Date: 01/07/2024

SUPERVISOR CERTIFICATE

This dissertation has been written under my supervision and has been submitted for the award of the degree of Doctor of Philosophy in Information System Engineering with my approval as supervisor.

Signature
Name: Prof. Dr. Aree Ali Mohammed

Date: /   / 2024

I confirm that all requirements have been fulfilled.

Signature:

Name: Byad A. Ahmed

Head of the Department of Information Systems Engineering

Date: /   / 2024

I confirm that all requirements have been fulfilled.

Postgraduate Office

Signature:

Name:

Date:

EXAMINING COMMITTEE CERTIFICATION

We certify that we have read this Dissertation "Improving Traffic Flow for Emergency Vehicles Using Deep Learning Techniques" and as an examining committee examined the student (Kamaran Hussein Khdir Manguri) in its content and what related to it. We approve that it meets the standards of a dissertation for the degree of Doctor of Philosophy in Information System Engineering.

Signature:	Signature:
Name: Prof. Dr. Mazen R. Khalil	Name: Asst. Prof. Dr. Moayad Y. Potrus
Chairman	Member
Date: / / 2024	Date: / / 2024
Signature:	Signature
Name: Assist. Prof. Dr. Ismael K. Abdulrahman (Member)	Name: Assist. Prof. Dr. Azhin T. Sabir
	Member
Date: / / 2024	Date: / / 2024
Signature:	Signature
Name: Assist. Prof. Dr. Shahab W. Kareem (Member)	Name: Prof. Dr. Aree A. Mohammed
	Supervisor
Date: / / 2024	Date: / / 2024

Signature

Name: Prof. Dr. Ayad Z. Sabir Agha

Dean of Erbil Technical Engineering College

Date: / / 2024

DEDICATION

This Dissertation is dedicated to:

My merciful parents

My best friend and lovely wife (Chopy) for her beliefs and supports

My Son, Kovan

My siblings.

ACKNOWLEDGMENTS

Expressing my gratitude to **Allah** for His blessings and guidance throughout this journey, I want to extend deep thanks to all who supported me in completing my PhD dissertation. A special acknowledgment goes to my supervisor, **Prof. Dr. Aree Ali Mohammed**, for their invaluable guidance and unwavering encouragement.

I also wish to express heartfelt appreciation to my dissertation committee members for their significant contributions and insightful feedback.

Furthermore, I would like to express my gratitude for the crucial support provided by the **Department of Information System Engineering** at **Erbil Polytechnic University**, which significantly aided in facilitating my research. Additionally, I appreciate the opportunity given by the **University of Raparin**. I am grateful for the camaraderie and encouragement from colleagues and friends, as well as the constant support from my family.

Lastly, I wish to thank all individuals who participated in this research project, as their contributions greatly enriched the outcome of my dissertation.

ABSTRACT

The world's population has exponentially grown, which has an effect on usage of vehicles by individuals and leads to an increase in the number of cars in urbans. With the direct relationship between population and car usage, traffic management has become an important issue to be solved. For this purpose, an intelligent traffic signaling with a rapid urbanization is required to overcome the traffic congestions, and reduce cost and time of traveling. To overcome these problems, emerging computer vision and deep learning are vital candidates to handle this issue because they take an important role for managing and controlling traffic signals with great success. Nevertheless, detecting and distinguishing between objects are helpful for counting vehicles and other objects which avoid crowds and controlling signals in the traffic areas. Besides, detecting emergency vehicles and giving the priority to them is required for intelligent traffic signaling system.

The main objective of this study is to design and implement an efficient system for traffic signal systems based on custom vehicle detection. Furthermore, the proposed system involves four phases; the first one is capturing images from both simulated and real time cameras from the roads. In the second phase, different image preprocessing algorithms are performed to the captured images as a pre-processing step. In addition, the deep learning techniques are applied to detect objects such as (regular car, police car, ambulance, and firefighter, etc..). In the last phase, the proposed system is tested to evaluate the performance accuracy of the detected vehicles.

A modified transfer learning approach has been applied to the DenseNet201 model for multiple classifications, including non-emergency cars, ambulances, police, and firefighters. The approach involves freezing the architecture of the model's layers. A high accuracy rate is obtained with this model and reaches 98.6%. Also, various optimization methods, including (Adam, Adamax, Nadam, and RMSprob) are used to improve the detection performance based on the best optimizer selection and yielded an accuracy of 98.84%. In addition, a modified version of YOLOv5 was proposed for vehicle detection, which aims to enhance the mean average precision (mAP) detection by 3%. Finally, the proposed system was simulated to reduce the waiting time at traffic signal. The experimental results demonstrate a significant reduction in waiting time, ranging from 30 to 100 seconds depending on the status.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Meaning
Adagrad	Adaptive Gradient Algorithm
Adam	Adaptive Moment Estimation
ARM	Advanced Reduced Instruction Set Computing
AVR	Atmel Alf and Vegard's RISC Processor
BoI	Blocks of Interest
BS	Background Subtraction
CBL	The Convolutional Layer
CCTV	Closed-Circuit Television
CNN	Convolutional Neural Network
COCO	Objects in Context
CSVM	Cubic Support Vector Machine
CTSD	Chinese Traffic Sign Dataset
CUDA	Compute Unified Device Architecture
CVIS	Cooperative Vehicle Infrastructure Systems
DenseNet	Densely Connected Convolutional Networks
EVs	Emergency Vehicles
FPN	Feature Pyramid Network
GPU	Graphics Processing Unit
GTSDB	German Traffic Sign Detection Benchmark
HIS	Hue-Saturation-Intensity
HMM	Hidden Markov Model
HOG	Histogram of Oriented Gradients
ITS	Intelligent Transportation Systems
KRG	Kurdistan Region of Iraq
LaRA	Laboratório de Robótica e Automação
LISA	Intelligent & Safe Automobiles
LSS	Local Self-Similarity Machine

mAP	Mean Average Precision
MaxGT	Maximum Green Time
MFCC-SVM	Mel-frequency Spectral Coefficients Combined with Support Vector Machine
MinGT	Minimum Green Time
MobileNet	Mobile Networks
MSFF	Multi-Scale Feature Fusion
Nadam	Nesterov-accelerated Adaptive Moment Estimation
OpenCV	Open Source Computer Vision Library
PSO	Particle Swarm Optimization
QL	Q-learning
R-CNN	Region-based Convolution Neural Network
ResNet	Residual Network
RF	Random Forest
RFID	Radio-Frequency Identification
RISC	Reduced Instruction Set Computing
RMSprop	Root Mean Square Propagation
RNN	Recurrent Neural Network
RoI	Regions of Interest
SGD	Stochastic Gradient Descent
SGW	Simplified Gabor Wavelets
SORT	Simple Online and Real-time Tracking algorithm
SPP	Spatial Pyramid Pooling
SPPF	Spatial Pyramid Pooling - Fast
SSD	Single-Shot Detection
STL	Smart Traffic Light
STS	Swedish Traffic Signs
SVM	Support Vector Machine
TSC	Traffic Signal Control

TUIC	Temporal Unknown Incremental Clustering
VANETs	Vehicular Ad-Hoc Networks
VGG	Visual Geometry Group
YOLO	You Only Look Once

CHAPTER ONE

1. INTRODUCTION

1.1 Overview

In recent years, there has been a notable increase in the number of cars (Jain et al., 2019) leading to a widespread problem of traffic congestion (Biswas et al., 2019) that presents various challenges worldwide. Consequently, there has been a rise in car accidents and a worrying escalation in yearly CO2 emissions (Coelho et al., 2005), both of which threaten the sustainability of future transportation (Guo et al., 2019). Moreover, effective traffic management heavily relies on the deployment of manpower (Kumaran et al., 2019a). Traffic control systems, operating on a time-dependent basis, have been designed to facilitate smooth traffic movement in all directions. However, it is important to note that the transition of traffic signals from green to red during turns may sometimes cause traffic congestion in one direction, with minimal improvement in traffic flow in the opposite direction (Malhi et al., 2011).

Congestions resulting from traffic signals could have adverse effects on the transportation economy, primarily due to increased fuel consumption (Lakshmi and Kalpana, 2017) and time expenditure (Jing et al., 2017). Additionally, road congestion contributes to environmental issues such as noise and air pollution (Qadri et al., 2020). Moreover, accidents occurring in congested traffic conditions can lead to injuries or even fatalities (Lakshmi and Kalpana, 2017). Conversely, reducing congestion offers economic, environmental, and social benefits. Among various solutions in urban settings, signalized intersections have been identified as effective means for addressing prevalent bottlenecks, thus playing a significant role in urban traffic management (Wei et al., 2019). To this end, the concept of smart cities revolves around Intelligent Transportation Systems (ITS), which are pivotal in modern urban planning (Yuan et al., 2019, Zhang et al., 2011). Throughout history, transportation systems have been vital components of national infrastructure. Research indicates in 2011, approximately 40% of the global population was spent at least an hour commuting daily (Zhang et al., 2011). Consequently, managing the increasing number of vehicles has become increasingly challenging without the assistance of technology (Veres and Moussa, 2019).

Furthermore, emergency vehicles are crucial in critical situations, but traffic congestion can be a significant threat to patients' lives, causing more than 20% of fatalities during ambulance transport. In densely populated areas, traffic jams are frequent during peak hours, slowing down emergency vehicles such as police cars, ambulances, and firefighters and worsening life-threatening situations. To address this problem, it is essential to prioritize these vehicles and introduce an automated traffic system that can recognize and clear their path (Roy and Rahman, 2019).

In general, in order to make the optimization problem manageable, several assumptions have to be made. The main problem here that arises is that these assumptions deviate and sometimes do so significantly from the real world. Meanwhile, many factors have effects on drivers in real word traffics such as driver's preference interactions with vulnerable road users (e.g., pedestrians, cyclists, etc.), weather and road conditions (Wei et al., 2019). Also, this system needs to accurately detect emergency vehicles among regular ones. Advanced techniques like Convolutional Neural Networks (CNNs) based on deep and transfer learning have shown great potential in computer vision, delivering results comparable to, and sometimes even better than, human expertise.

Computer vision is crucial in effectively managing and regulating traffic signals (Kumaran et al., 2019a), contributing to significant success (Jeon et al., 2018). In general, in busy urban areas, the best way to control traffic flow is by using intelligent traffic signal systems(Kumaran et al., 2019a). These systems can estimate density, detect and recognize traffic signals, identify emergency and police vehicles, and detect accidents. Even though a better infrastructure can improve the traffic flow(Wang et al., 2018), quieter intersections are usually managed by human controllers or automated systems (Kumaran et al., 2019a). Cameras are already present in many congested areas for various purposes, but they can be utilized for analyzing traffic scenes with specific hardware, instead of replacing existing closed-circuit television (CCTV) cameras. These cameras are commonly used for security and vehicle detection. This multi-functional use of cameras optimizes resources and enhances overall traffic management efficiency (Khan and Ullah, 2019).

Besides on computer vision, deep learning techniques have capability to distinguish vehicle classes, and counting number of vehicles for different classes in a traffic video (Asha and Narasimhadhan, 2018). Scientists and researchers have proposed many approaches for traffic density estimation which will be done by using images acquired from live cameras positioned near the traffic junction and several machine learning algorithms (Ikiriwatte et al., 2019).

1.2 Urban Traffic Light System

Usually, each traffic light contains three color lights: green, yellow and red lights. These lights are positioned in the four directions—north, south, east, and west—at intersections (Huang and Chung, 2009b). Figure 1.1 illustrates a typical intersection formed by two perpendicular and parallel lanes. The traffic signal comprises three signal lights: red (R), yellow (Y), and green (G), and additionally features five lights: red (R), yellow (Y), a left turn arrow illuminated during green (GL), a right turn arrow illuminated during green (GR), and a straight-ahead arrow illuminated during green (GS) (Huang and Chung, 2009a).

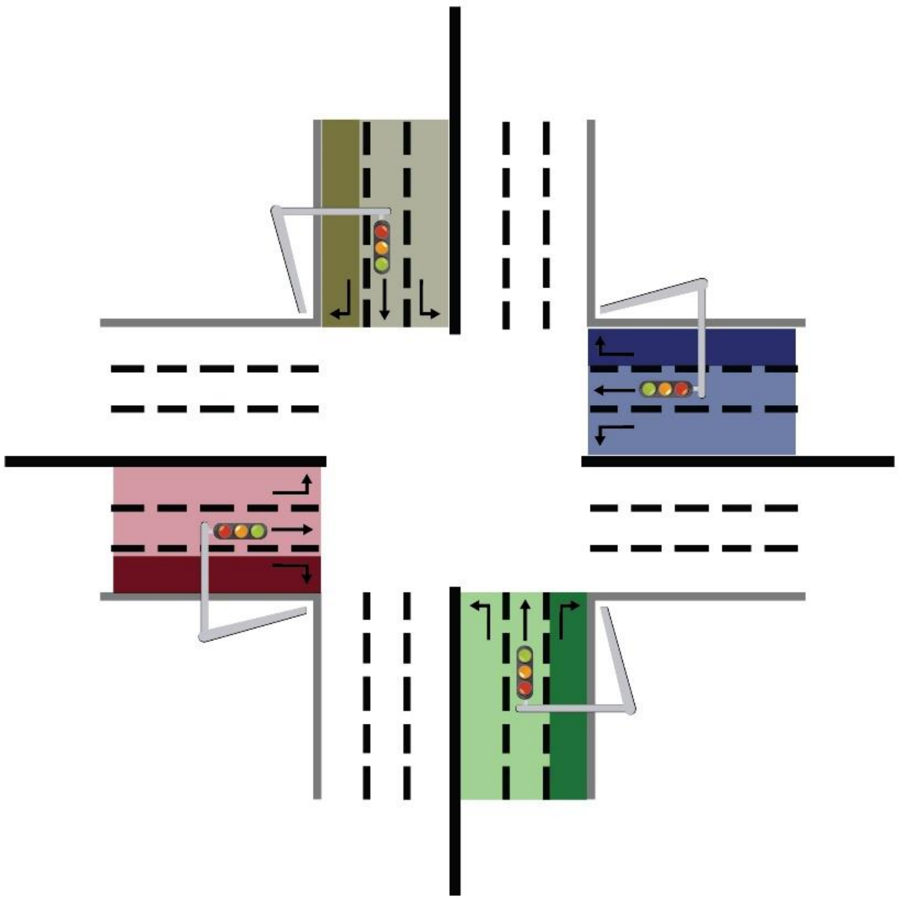


Figure 1.1 Four lanes intersection (Manguri et al., 2023)

The configuration of traffic lights can lead to different phase transitions based on the interactions among the number of signal lights (Huang and Chung, 2009a). A state diagram in Figure 1.2 displays the sequence of 4 phase transitions. The diagram shows a cycle that starts in phase 1 and ends in phase 4, providing an alternative series of operations. During phase 1, the southern traffic light enables the GL, GR, and GS signals, allowing vehicles to turn left, right, or go straight in the southbound direction. Similarly, during phase 4, the GL, GR, and GS signals are activated at the southern traffic light, allowing vehicles to turn left, right, or go straight in the southbound direction. The system follows predetermined timing intervals with fixed durations.

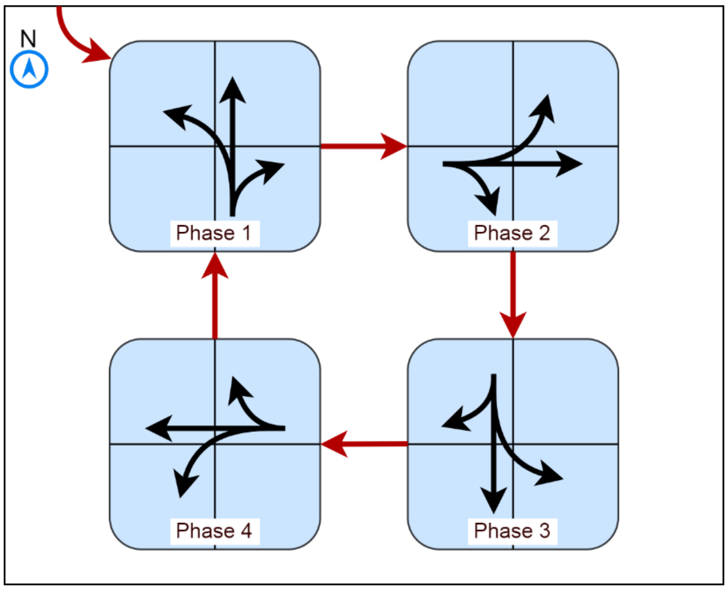


Figure 1.2 Phase Transitions (Huang et al., 2005)

Internationally, drivers understand traffic lights. A green signal allows vehicles in that lane to proceed while all other directions show red, signaling to stop (Wei et al., 2019). The management of traffic lights and monitoring of traffic conditions using computer vision technologies require CCTV cameras at intersections. Figure 1.3 shows a simulation of traffic control at a crossroad (Manguri, 2016).

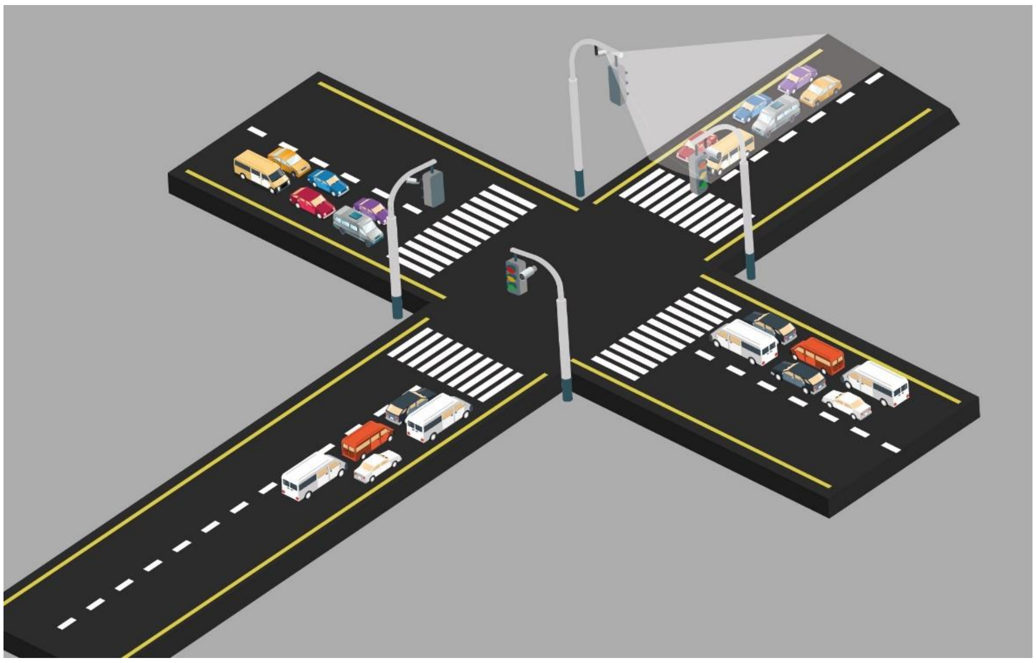


Figure 1.3 Crossroad Model and Cameras (Manguri et al., 2023)

1.3 Problem Statement

The global population is rapidly increasing and is projected to exceed 8 billion by 2023 and reach 10 billion by 2056 (Chamie, 2020). As a result, existing transport infrastructures are being strained, leading to worsened traffic congestion (Ariffin et al., 2021). This congestion has significant consequences, including high financial costs (Lee and Chiu, 2020) and decreased quality of life (Ghazali et al., 2019). One critical issue that arises from this situation is the inadequate response of traditional traffic systems to emergency vehicles, such as ambulances, fire trucks, and police cars. These vehicles often experience delays due to congestion, which can hinder their ability to quickly reach emergency scenes. The challenge lies in developing traffic systems that effectively prioritize emergency vehicles by coordinating traffic signals to stop non-priority traffic, ensuring safe and uninterrupted passage. Addressing this problem is crucial for improving emergency response times and overall public safety.

1.4 Research Objectives

This study aims at designing and implementing an efficient traffic signaling system based on custom vehicle detection.

The objectives include:

1. Saving time by reducing waiting time at intersections, especially for emergency vehicles.
2. Density estimation and counting cars on the road for reducing traffic congestion.
3. Changing side signal to green signal in case of detection ambulance, firefighters and police cars.

4. Testing the proposed system and evaluating its performance with other related works.

1.5 Research Contributions

Researchers mostly use deep learning architecture for vehicle classification and detection. However, in this study, a transfer based deep learning system using Densely Connected Convolutional Networks with 201 Layers (DenseNet201) was used as a main contribution to this work. The significant key contributions of the proposed model are as follows:

1. Testing varying dimensions of input images (64*64, 128*128, and 224*224) to select the optimal image size for the training phase.
2. Applying data augmentation to make the datasets balanced among emergency vehicles' types (police, ambulance, and firefighters).
3. Modifying some layers in DenseNet-201 to improve performance accuracy.
4. Testing other deep learning models such as (MobileNet, VGG-19, ResNet-101, etc.) on the same datasets to produce a fair comparison.
5. Choosing the best optimizer that increases the detection accuracy of the model.
6. Applying You Only Look Once version 5 (YOLOv5) for controlling real time signaling.
7. Modifying the YOLOv5 to improve the model accuracy.
8. Proposing a mathematical formulation for optimizing traffic signaling.
9. Designing and implementing an integrated system for smart traffic signaling based on Arduino.

1.6 Research Challenges and Limitations

Collecting data from real traffic area officially requires a permission from traffic police center. Due to the non-availability of public datasets, a customized dataset has been created. The main challenge of the intersection traffic signal control (TSC) problem is to determine an optimal configuration of traffic signaling system that allows a maximum traffic flow in a network. Creating a standard dataset for emergency vehicles is not feasible due to the lack of standardized colors, sizes, and other characteristics. In this regard, an accurate system for traffic congestion reduction is required to build, which is the main goal of this study.

Furthermore, deep learning methods such as Region-based Convolution Neural Network (R-CNN), Fast R-CNN, and Faster R-CNN necessary large amount of training data and computational power. The limitation of Faster R-CNN was its inference time. Moreover, SSD (Single-Shot Detection) and YOLO (You Only Look Once) were developed to solve the Faster R-CNN limitation. Generally, SSD and YOLOv5 methods have fast detection speed and high efficiency but a low accuracy.

1.7 Dissertation Structure

The rest of this dissertation is organized as follows:

Chapter Two: Literature Review and Theoretical Background:

In this chapter, a comprehensive review has been done based on recent researches and describes the theoretical background of all traffic signal phases (custom dataset preparation, data preprocessing, train DenseNet201 model, object detection, and system evaluation).

Chapter Three: Methodology, Research Design, Materials and Methods:

The proposed method and used materials are given in chapter three.

Chapter Four: Implementation**,** Results and Discussion:

This chapter provides an implementation of the developed intelligent traffic signaling system. Also, it discusses the obtained results.

Chapter Five: Conclusions and Future Works:

The summary of this study can be found on the chapter five, as well as suggestions for future work.

CHAPTER TWO

2. THEORETICAL BACKGROUND AND LITERATURE REVIEW

2.1 Introduction

This chapter provides the theoretical background of this study such as object detection and its techniques are discussed. Furthermore, literature review of the recent studies carried out on computer vision for managing and controlling traffic signals in this chapter.

2.2 Theoretical Background

Traffic signaling control can be categorized into several types based on the technology and methods used. This study provides detailed information on common types of traffic signaling control. However, it is important to note that some traffic signaling control systems, such as Pedestrian Signals, Traffic Signs and Road Markings, Traffic control for construction zones, Railway Crossings, and Traffic Roundabouts, are not directly related to this study.

2.1.1 Fixed-Time Traffic Signals

Fixed-time control involves the use of predetermined green and cycle durations that remain constant, regardless of variations in traffic volumes throughout the day. This approach primarily assigns more green time to the busiest traffic movements based on historical data. Some fixed-time systems may incorporate distinct preset time intervals for morning rush hours, evening peak hours, and other periods of high traffic. Nevertheless, this method is not well-suited for handling unexpected shifts in traffic demand. To address this limitation, vehicle-actuated traffic signals are implemented. Traditional vehicle-actuated signals essentially respond to the need for green time and green extension based on vehicle detection, provided it falls within predetermined limits and as long as traffic flow allows for accurate measurement. The cycle duration becomes variable, and consequently, adjacent signalized intersections can only be coordinated in specific scenarios (Wahlstedt, 2013).

2.1.2 Actuated Traffic Signals

The actuated control method allows for dynamic modification of the green time duration and total cycle length in response to real-time traffic needs. These needs are determined using loop detectors or other conventional traffic sensors. Within this control scheme, critical parameters such as minimum green time (MinGT), maximum green time (MaxGT), and cycle duration can be adjusted based on actual traffic conditions, often using data collected by loop detectors (Shirvani Shiri and Maleki, 2017, Zheng et al., 2010, Zhang and Wang, 2010, Viti and Van Zuylen, 2010). Furthermore, actuated signal phase timing schemes are constrained by their inability to accurately adapt to changing traffic conditions. This means that parameters such as MinGT, MaxGT, and cycle length are still determined using predefined values. These values rely on a limited amount of traffic data and recommended experimental values (Nie et al., 2021).

2.1.3 Adaptive Traffic Signals

The adaptive traffic control system, known as UTOPIA, is designed to optimize traffic flow while granting selective priority to public transport, all without compromising travel times for private vehicles (Wahlstedt, 2013, Samadi et al., 2012). UTOPIA represents an innovation in Urban Traffic Control (UTC) and is characterized by its hierarchical, adaptive, distributed, and open traffic control system (Taranto, 2012).

Hierarchical: UTOPIA employs goal-related coordination and incorporates cooperative control.

Adaptive: The system continually monitors real-time traffic conditions and adjusts signal plans to ensure rapid responses to changes in traffic demand.

Distributed: This approach involves problem decomposition, forward-looking strategies, strong interaction, and a focus on terminal costs. The network optimization is broken down into coordinated junction problems, which are addressed by the intersection units (SPOT) in collaboration with the central system.

2.1.4 Traffic Control by Law Enforcement

Law enforcement officers are entrusted with the duty of manually regulating traffic, particularly in situations such as events, emergencies, or construction zones. They rely on hand signals and stop signs to provide guidance and exercise control over the flow of vehicles.

2.1.5 Smart Traffic Signals

The definition of a Smart Traffic Light (STL) involves using sensors, cameras, and artificial intelligence to control the flow of traffic. This novel method depends on the traffic light playing a central role in overseeing the movement of vehicles and reducing congestion on the roads. To meet the criteria of being labeled as a smart traffic light, it must incorporate a mechanism for automatically adjusting signals in real-time.

Various techniques can be employed for developing smart traffic signals, including the installation of cameras at traffic intersections. These cameras capture images of the vehicles on the road from a preset distance away from the traffic signal. Subsequently, image analysis techniques are applied to assess the number of vehicles present on the road. The information derived from this analysis is then transmitted to the traffic light, enabling it to adjust signals (red, yellow, or green) accordingly (Choudekar et al., 2011).

Under the concept of STL, sensors are used on the sides of roads and vehicles are interconnected. The collected data undergoes analysis, and based on this information, concert commands are generated. These commands are then transmitted to the traffic light, facilitating a responsive adjustment of signals to optimize traffic flow (Hussain et al., 1995).

2.2 Deep and Transfer Learning Models

Deep learning, a subset of AI, is specialized in building substantial neural network models proficient in making accurate decisions from data. Its strength lies in situations marked by complex data and access to large datasets (Kelleher, 2019). Therefore, conducting training with a wide range of potential outcomes can reduce the chances of making errors during testing (Dong et al., 2021). But the current moment, the limitation on neural network complexity was tied to available computing power. However, progress in big data analytics has empowered the development of more intricate neural networks. As a result, computers can now rapidly observe, learn, and respond to complex situations more swiftly than humans (Kuutti et al., 2020).

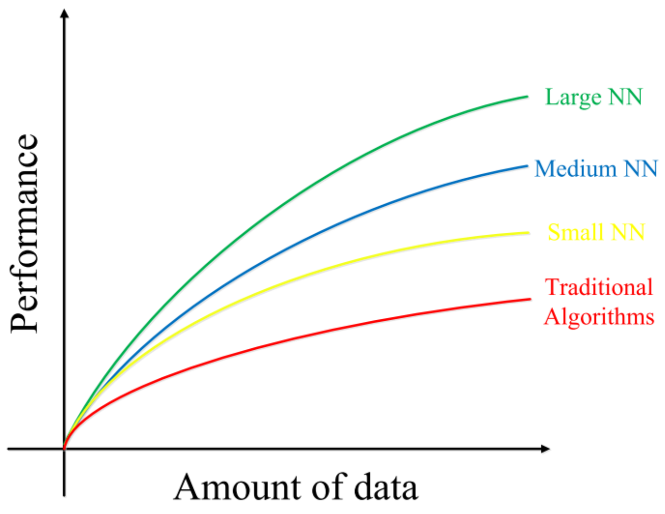


Figure 2.1 Scale driving Deep Learning progress (Dong et al., 2021).

Figure 2.1 illustrates how the performance of the Deep Neural Network improves as the data size increases (Dong et al., 2021).

The main focuses of deep learning techniques in the ITS including Density estimations, traffic signal control, accident detection, traffic sign detection and recognition, etc.(Dong et al., 2021).

Deep learning neural networks use both data inputs and weights, which can be numerical or learned by machines, then multiplied by inputs. They adjust themselves based on differences between predicted output and training inputs.

Additionally, they make use of activation functions, mathematical equations controlling neuron activity, and bias, in algorithms skews results by either encouraging or discouraging certain ideas. These elements collectively mimic the human brain (Kelleher, 2019, Chen et al., 2021), allowing for precise recognition, classification, and description of data objects. As a result, multiple layers serve as the backbone of all neural network types, working together to construct a neural network model (Dong et al., 2021), however, the exact number of layers can vary.

Input layer: Depicts the dimensions of the input vector.

Hidden layer: Intermediate nodes that partition the input space into distinct boundaries. An activation function generates an output based on weighted inputs.

Output layer: Represents the final output of the neural network.

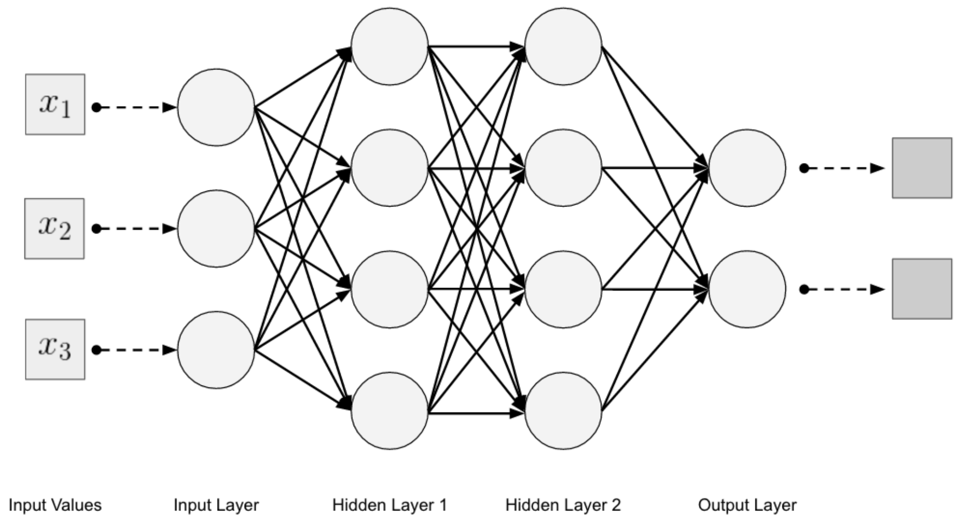


Figure 2.2 Multi-Layer Neural Networks (Ozturk and Fthenakis, 2020)

2.2.1 Deep Learning Factors

Deep learning techniques encompass various factors such as:

2.2.1.a Neural Network Architecture

The Neural Network Architecture is defined as arrangement and composition of layers, nodes, and interconnections within the network. The Architecture relies on three primary elements that emulate the usual procedures of feature extraction, matching, simultaneous inlier detection, and model parameter estimation. These components are designed to be trainable in an end-to-end manner. To enhance performance, it usually requires a larger network capacity. A high-performing network architecture often encompasses a wide range of potential setups concerning the number of layers, hyperparameters within each layer, and the layer types. This complexity makes a thorough manual search impractical, relying significantly on expert knowledge and experience to craft successful networks. As a result, the automated and intelligent construction of networks remains an ongoing challenge yet to be resolved (Zhong et al., 2018).

2.2.1. b Activation Functions

Another crucial element within a neural network involves activation functions, modeled after the firing mechanism of human neurons—signaling either activation or inactivity. These functions serve to establish nonlinear connections between input and output, mirroring the structure of the human brain due to their nonlinearity, multiple nodes, and layers, thus earning the term "neural network." Several activation functions, as illustrated in Figure 2.3, exist, including commonly used ones like Sigmoid, Hyperbolic tangent, and Relu. Their primary function lies in transforming and abstracting data into a format more conducive to classification (Dong et al., 2021).

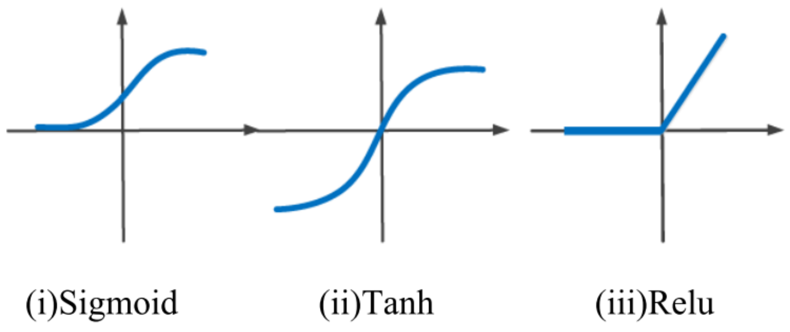


Figure 2.3 Activation Functions (Dong et al., 2021).

2.2.1. c Parameter learning

As a traditional machine learning classifiers, deep learning classifiers also require learning parameters utilizing mathematical tools such as gradient descent. The gradient descent algorithm is particularly valuable for parameter learning in convex functions, which possess a single absolute minimum or maximum. Convex functions simplify parameter learning, while nonconvex functions require mathematical techniques to transform them into convex ones. Neural network optimization is considered a non-convex optimization problem, implying the existence of multiple optimum points (minima/maxima). Learning occurs by minimizing the error between predicted and actual values (Dong et al., 2021).

2.2.1. d Optimization algorithms

Optimization algorithms, crucial in improving neural network performance, play a significant role in adjusting a model's hyperparameters in a standard fashion. Hyperparameters, such as the learning rate, direct the update method, defining the optimizer's behavior. Any two optimizers can be distinguished by the combination of their hyperparameters and update rule (Fatima and Journal, 2020).

In the Deep Learning models, loss function is provided (could be considered as the function we aim to optimize for our specific problem). This, coupled with the optimization algorithm, becomes a mandatory parameter for compiling our model. The optimizer's function involves adjusting the weights and learning rate of our model's nodes during training to effectively minimize the loss function. In essence, the primary objective of an optimizer is to decrease the training error (Fatima and Journal, 2020). All optimizers use a mathematical formula to update the weights, along with specific learning rate values.

$$w_x' = w_x - \propto \left(\frac{\partial error}{\partial w_x}\right) \qquad 2.1$$

Where w_x' denotes the updated weights while w_x represents the old weights, and $\propto$ indicates the learning rate. The term $\left(\frac{\partial \mathrm{error}}{\partial \mathrm{w}_\mathrm{x}}\right)$ represents the derivative of the error concerning the weights. Different algorithms implement adjustable learning rates.

I. Root Mean Square Propagation (RMSprop)

RMSprop, despite its widespread use, is an optimizer that was never officially published. Geoff Hinton, a prominent figure in backpropagation, introduced it during his online course on Neural Networks for machine learning. RMSprop and Adadelta emerged concurrently but separately, aiming to address the diminishing learning rates observed in Adagrad. RMSprop, a gradient-based optimizer, diverges from considering the learning rate as a fixed hyperparameter, opting for an adaptive learning rate that is adjusted dynamically during training.

II. Adaptive Moment Estimation (Adam)

Adam is one of the most popular optimizers and is closely related to RMSprop and Adagrad. It is known for its high efficiency, adaptability, and fast convergence. Adam uses the L2 norm or Euclidean norm for optimization. Like RMSprop, it incorporates squared gradients to adjust the learning rate. Additionally, it leverages momentum by calculating the moving average of the gradient, which is similar to Stochastic Gradient Descent with momentum. The name "Adam" stems from "adaptive moment estimation," signifying its computation of individual learning rates for diverse parameters. This is achieved by calculating the first and second moments of gradients and adjusting the learning rate for each weight in the network.

III. Adamax

Adamax could be described as an Adam optimizer variant that utilizes the infinity or max norm for optimization. In scenarios where data exhibits traditional noise in gradient updates, such as datasets with multiple outliers, Adamax tends to outperform Adam.

IV. Nesterov-accelerated Adaptive Moment Estimation (Nadam)

Nadam amalgamates Adam and NAG (Nesterov Accelerated Gradient) techniques. It incorporates Nesterov to update the gradient a step ahead, hence named "Nesterov-accelerated Adaptive Moment Estimation." Nadam finds utility in handling noisy or highly curved gradients. Accelerating the learning process, it speeds up by aggregating the exponential decay of moving averages from the previous and current gradients.

In addition, there are other optimization techniques available that were not detailed in this study due to the good results obtained from tests, such as Stochastic Gradient Descent (SGD), Adaptive Gradient Algorithm (Adagrad), and AdaDelta.

2.2.1. e Data Preprocessing

Preparing data involves converting raw information into a format suitable for training machine learning models, which is a time-consuming process. In the case of image data, preprocessing typically encompasses basic transformations like cropping, filtering, rotating, or flipping images. Presently, data scientists manually determine, based on their expertise, which transformations and their sequence are best suited for a specific image dataset. This manual approach not only poses a bottleneck in real-world data science projects but can also result in suboptimal outcomes. Relying on intuition or trial-and-error methods to explore various image transformations might hinder the discovery of the most effective ones (Minh et al., 2018).

2.2.1. f Hyperparameter Tuning

Optimizing or tuning hyperparameters is a core technique for improving the performance of machine learning models. Hyperparameters are parameters that are adjusted during the learning process to fine-tune and optimize the model's performance. Various constraints, weights, or learning rates may be required for the same machine learning model to effectively capture different patterns in data. These adjustable parameters, known as hyperparameters, are determined through a process of trial and error to ensure that the model performs at its best in carrying out the machine learning task (Pon et al., 2021).

2.2.2 Deep Learning-Based Classification Techniques

Image classification refers to sorting an entire image into predefined classes or categories, like determining if an image depicts a cat or a dog. Conversely, object detection not only categorizes objects within an image but also pinpoints their positions by outlining bounding boxes around them. Object detection offers a more detailed comprehension by identifying and localizing multiple objects within an image, whereas image classification concentrates on assigning a single label to the entire image.

2.2.2.a Image Classification

Image classification is an essential part of image processing. Its objective involves assigning images to predefined categories (Lillesand et al., 2015). There are two primary types of classification: supervised and unsupervised classification. Image classification involves a two-step procedure: training and testing. During training, the framework extracts distinctive attributes from the images to form a unique representation for each class. This process is repeated for all classes, depending on the nature of the classification problem—be it binary or multi-class (Jaswal et al., 2014, He et al., 2016, Liu et al., 2015, He and Sun, 2015). During the testing phase, the aim is to classify test images into the classes for which the system was trained. The class assignment is based on the partitioning between classes, considering the training features (Aggarwal and vision, 2018).

In recent decades, CNNs have demonstrated significant potential across various domains within the computer vision community (LeCun et al., 2015). Also, many techniques are available for image classifications such as DenseNet, ResNet, Mobile Net. VGG, Inception, AlexNet, and etc. The used deep learning techniques during this study to standardize the proposed dataset are detailed below:

I. VGG Networks

The VGG network is a convolutional neural network architecture known for its depth and simplicity. The VGG network is composed of very small convolutional filters, comprising 13 convolutional layers and three fully connected layers (Simonyan and Zisserman, 2014). Its design emphasizes simplicity in constructing the system. The VGG network is primarily a deep convolutional neural network created with careful consideration of the optimal layer depth to avoid excessive network complexity (Haque et al., 2019). The architecture of the VGG network is illustrated in Figure 2.4.

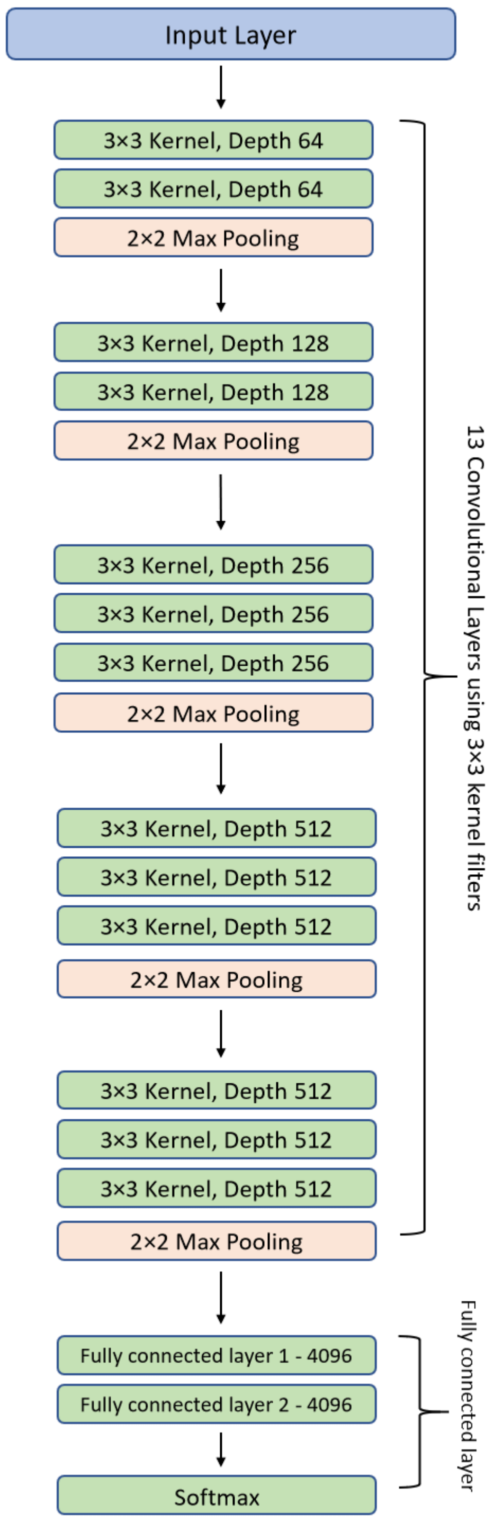


Figure 2.4 VGG network Architecture (Tammina and Publications, 2019)

II. ResNet

Deep residual networks show significant performance in computer vision and image processing tasks like image classification and segmentation. These networks resemble a set of filters, utilizing smaller convolutional filters to simplify the overall network architecture. ResNet is structured with a series of 1x1 and 3x3 convolutional layers, functioning as filters (He et al., 2016). These layers play a crucial role in reducing network complexity while extracting high-level features. The network's output relies on a wealth of information used to activate fully connected layers, effectively preserving extensive information that pinpoints the precise location of objects (Haque et al., 2019). Figure 2.5 shows the Basic ResNet Architecture.

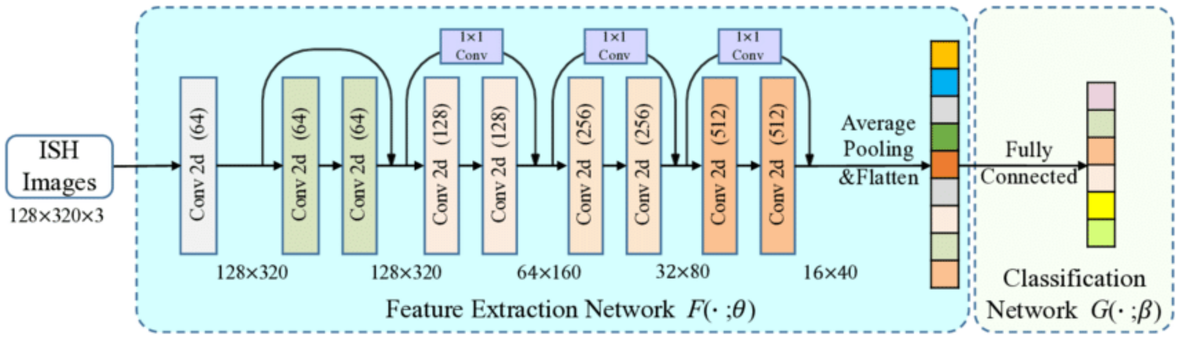


Figure 2.5 ResNet Network Architecture (Cai et al., 2021)

III. MobileNet

The MobileNet model uses depthwise separable convolutions, which are a type of factorized convolution. This convolution splits a standard convolution into two parts: a depthwise convolution and a 1×1 pointwise convolution. In MobileNets, the depthwise convolution applies a single filter to each input channel, and then the pointwise convolution combines the outputs from the depthwise convolution. Unlike a standard convolution that filters and combines inputs in one step, the depthwise separable convolution separates this process into two layers: one for filtering and one for combining. This factorization significantly reduces both computational load and model size (Howard et al., 2017). The architecture of MobileNet presented in Figure 2.6.

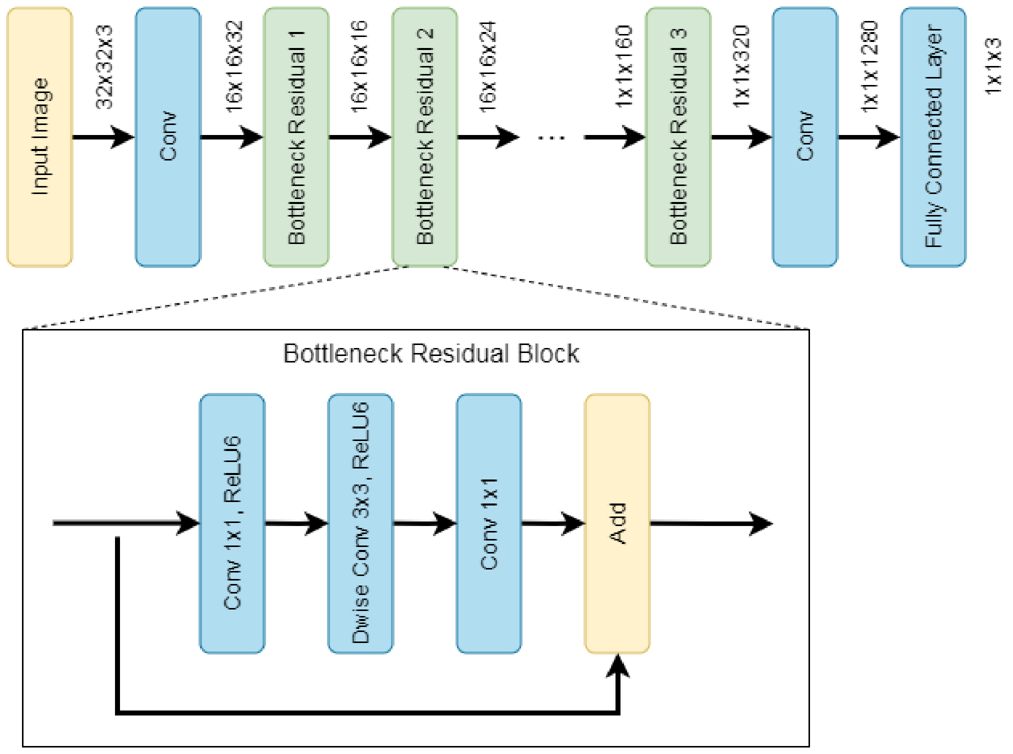


Figure 2.6 MobileNet Architecture (Shobeiri et al., 2021)

IV. DenseNet

The evolution of DenseNet201 proposes the remarkable potential of the dense architectural framework to achieve cutting-edge outcomes. This advancement is particularly evident in scenarios where a modest growth rate is adopted, resulting in the perception of feature maps as a comprehensive network-wide resource. As a result, each sequential layer enjoys unfettered access to the entirety of feature maps stemming from preceding layers. The incremental contribution of K feature maps to the global network state transpires within each layer, defining the cumulative input feature maps at the 1th layer as:

$$(FM)^l = K^0 + K(l-1) \qquad 2.2$$

where, K^0 signifies the input layer's channels. The enhancement of computational efficiency is portrayed in Figure 2.7, wherein a (1*1) convolutional layer precedes each (3*3) convolutional layer, thereby curtailing the volume of input feature maps. Serving as a bottleneck layer, the (1*1) convolutional layer engenders the production of 4K feature maps. For classification purposes, the incorporation of two dense layers housing 128 and 64 neurons, respectively, is realized. The modified feature extraction network, encompassing a truncated DenseNet201 architecture, is augmented by a sigmoid activation function for binary classification, thereby supplanting the conventional softmax activation function is employed within the established DenseNet201 framework (as depicted in Figure 2.7). The sigmoid function's definition is as follows:

$$y = \frac{1}{1+e^{-(\sum_i w_{i*}x_i)}} \qquad 2.3$$

Where y is the output of the neuron, w_i *and* $x_i$represent the weights and inputs, respectively.

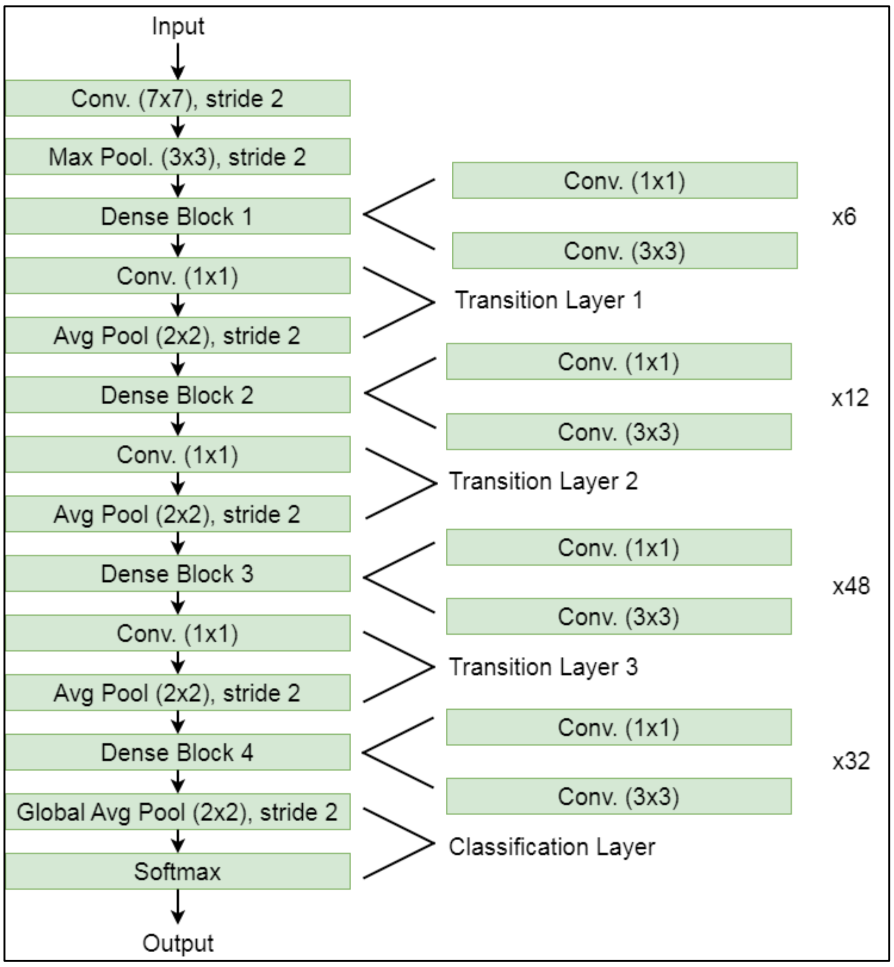


Figure 2.7 DenseNet201 architecture (Jaiswal et al., 2021)

2.2.3 Deep Learning-Based Object Detection Techniques

Object detection is a highly researched subject in the field of computer vision. Numerous methods have been developed to address the growing demand for precise models in object detection (Hu et al., 2004). The Viola-Jones framework (Viola and Jones, 2001) gained popularity primarily for its successful implementation in solving face-detection challenges (Padilla et al., 2012). Subsequently, it was adapted for various subtasks such as pedestrian (Ohn-Bar and Trivedi, 2016) and car (Sun et al., 2006) detections. In recent times, with the rise of CNN (Krizhevsky et al., 2012, Szegedy et al., 2015, LeCun et al., 1998, He et al., 2016) and GPU-accelerated deep-learning frameworks, an innovative perspective emerged in the development of object-detection algorithms (Hinton et al., 2006, Hinton and Salakhutdinov, 2006). Works like Overfeat (Sermanet et al., 2013), R-CNN (Girshick et al., 2014), Fast R-CNN (Girshick, 2015), Faster R-CNN (Ren et al., 2015), R-FCN (Dai et al., 2016), SSD (Liu et al., 2016), and YOLO (Redmon et al., 2016) significantly elevated the performance benchmarks in this field.

2.2.3.a YOLO

YOLO is an object detection technique known for its efficiency and accuracy in real-time object detection tasks. YOLO distinguishes itself from earlier methods used in object detection. Unlike previous practices that repurpose classifiers for detection tasks, YOLO reframes object detection as a regression challenge to predict bounding boxes and class probabilities within an image. This method employs a single neural network to directly generate these predictions from entire images in a single evaluation. As the entire detection process operates within a single network, it allows for direct optimization specifically for detection performance.

The unified architecture is highly efficient in terms of speed. The standard YOLO model is capable of processing images in real-time at 45 frames per second. In comparison, a smaller version of this network, called Fast YOLO, achieves an impressive speed of 155 frames per second while also improving the mean average precision (mAP) compared to other real-time detectors. Though YOLO may demonstrate more localization errors when compared to cutting-edge detection systems, it is less prone to making false positive predictions in the background (Redmon et al., 2016).

The YOLOv5 network is shown in Figure 2.8 and comprises three main segments: the backbone, neck, and head (output result). The backbone, positioned immediately after the input image, primarily handles feature extraction. Following this, the CNN's neck processes data to perform resolution feature aggregation, and the final predictions based on object resolution are generated in the head (Khasawneh et al., 2023, Kateb et al., 2021).

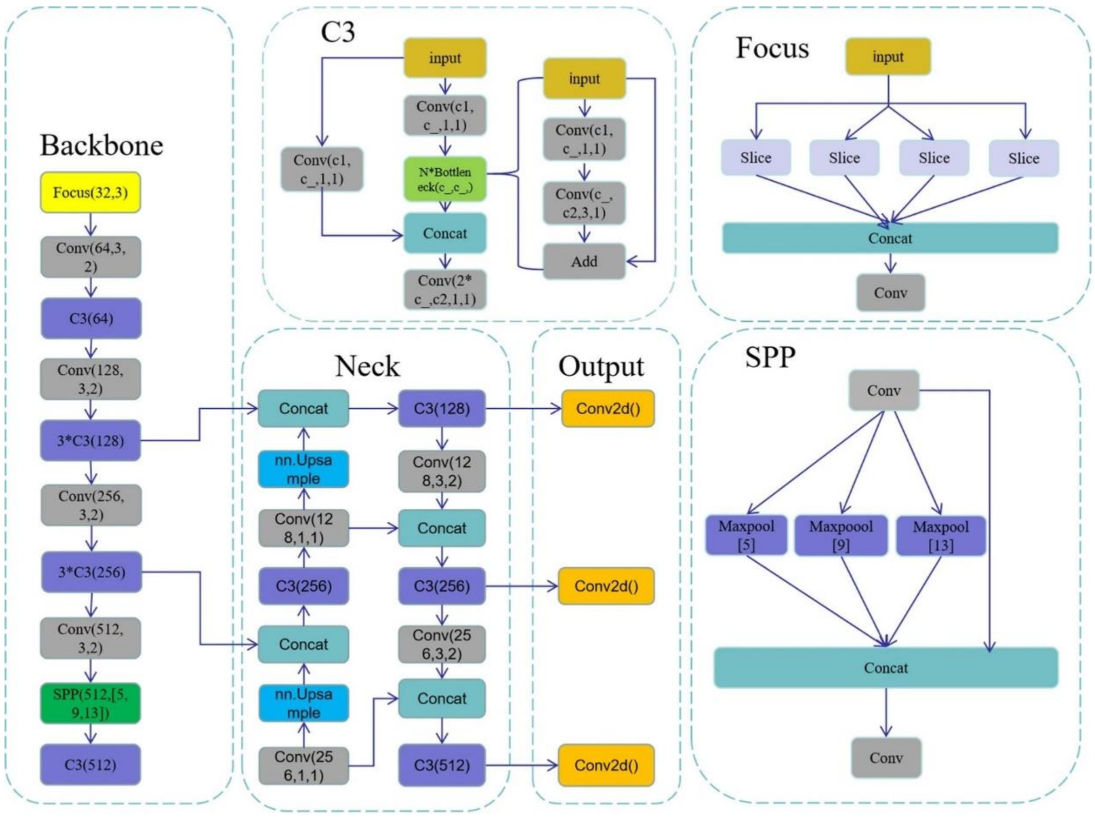


Figure 2.8 The YOLOv5 network framework (Guo and Zhang, 2022)

In the backbone part, the input image, initially sized at 640 × 640 × 3, is transformed by the focus layer for space-to-depth conversion. Using the slice operator, this process reshapes the image to 320 × 320 × 12, creating a feature map. This map is then passed through the convolution operator, which utilizes 32 convolution kernels, resulting in a feature map of size 320 × 320 × 32. The Convolutional Layer (CBL) involves the application of the convolutional kernel to the input layer, resulting in Conv2D outputs, batch normalization (BatchNormal), and LeakyRELU activation (Zhou et al., 2021). BottleneckCSP plays a pivotal role in feature extraction from the map and stands out for its ability to diminish information gradient duplication during neural network optimization, constituting a substantial part of the entire YOLOv5 network's parameters (Wu et al., 2021).

The research explores the variations in width and depth of the BottleneckCSP segment, which led to the development of four YOLOv5 models: YOLOv5s (small network), YOLOv5m (medium-sized network), YOLOv5l (large network), and YOLOv5x (very large network) (Glučina et al., 2023, Zhou et al., 2021) (See Figure Y). The Spatial Pyramid Pooling (SPP) module enhances the network's receptive field and captures additional features of diverse scales. YOLOv5 also integrates a bottom-up pyramid structure based on the feature pyramid network (FPN) (Chen et al., 2022). The FPN layer is essential for transmitting semantic features and enhancing the robustness of bottom-up positioning features. It aggregates features from various layers to improve the network's performance and its ability to detect targets at multiple scales. This is evident towards the end of the image, where the classification results and object coordinates are shown (Glučina et al., 2023).

2.3 Simulation Model Components

The simulation model requirements for a smart traffic system based on Arduino and computer vision vary. They include a range of essential components that are crucial for the system to operate effectively and efficiently which include Arduino UNO board, a breadboard, LED Traffic Light Module Board, three 220-ohm resistors, and jumper wires for connectivity.

2.3.1 Arduino Board

The Arduino microcontroller is a platform that is open-source and designed to be easily programmed and updated at any time. Originally intended for professionals and students, it enables the creation of devices that can interact with the environment using sensors.

Arduino microcontrollers feature both inputs and outputs that enable the acquisition of information, and based on the received data, Arduino can generate the output signal.

The Arduino platform comprises two main components: hardware and software. The hardware component involves the use of the Arduino development board, while the software component involves the Arduino IDE (Integrated Development Environment) for code development. Arduino utilizes either 8-bit Atmel Alf and Vegard's RISC processor (AVR) microcontrollers or 32-bit Atmel Advanced Reduced Instruction Set Computing (RISC) Machine (ARM) microcontrollers. These microcontrollers can be easily programmed using the C or C++ language within the Arduino IDE (Ismailov et al., 2022).

There are several advantages to choosing Arduino microcontrollers over other options, as highlighted by Massimo Banzi, the Co-founder of Arduino (Ismailov et al., 2022):

- ***Active User Community:*** Arduino has a vibrant user community where individuals can share their experiences and seek help when encountering issues. This collaborative environment promotes problem-solving and knowledge-sharing among users.
- ***Inexpensive Hardware:*** Arduino microcontrollers are known for their affordability, making them a perfect choice for beginners who want to start projects without spending too much money.
- ***Growth of Arduino:*** Another benefit is that the Arduino platform itself is free to use through the official website, and users only need to invest in Arduino hardware.
- ***Multi-platform Environment:*** The Arduino Integrated Development Environment (IDE) is compatible across multiple platforms, including Microsoft, Linux, and Mac OS X, which further expands the user base and accessibility of the platform.

Various types of Arduino boards are available (As shown in Table 2.2), each differs from others in terms of their microcontroller types, crystal frequencies, and the presence of auto-reset functionality.

Table 2.1 Different Types of Arduino Boards

Arduino Board	Microcontroller Type	Crystal Frequency	Auto-Reset Availability	Memory Type
Arduino Uno	ATmega328P	16 MHz	Yes	Flash: 32 KB
Arduino Mega 2560	ATmega2560	16 MHz	Yes	Flash: 256 KB
Arduino Nano	ATmega328P	16 MHz	Yes	Flash: 32 KB
Arduino Due	ATSAM3X8E	84 MHz	Yes	Flash: 512 KB
Arduino Pro Mini	ATmega328P	16 MHz	No	Flash: 32 KB
Arduino Leonardo	ATmega32U4	16 MHz	Yes	Flash: 32 KB

For this study, the Arduino Uno has been selected as it meets the requirements of the traffic signaling model in terms of processing power, size and form factor, cost, features and connectivity.

2.3.2 LED Traffic Light Signal Module

The LED traffic light signal module is a digital traffic light module that provides a signal output. This module is known for its high brightness, making it perfect for creating models of traffic light systems. Additionally, it is small in size, easy to wire, and comes with a targeted design and customizable installation options.

2.3.3 Breadboard or Prototyping Board

A breadboard or prototyping board is an essential tool for assembling circuitry, allowing you to connect LEDs, resistors, and microcontroller elements. It provides a convenient and adaptable platform designed specifically for testing and prototyping electronic circuits.

2.3.4 Jumper Wires

Jumper wires are used to establish connections between components on a breadboard or prototyping board. They provide a versatile and easily adaptable method for interconnecting circuitry.

2.3.5 Camera

Cameras are essential components in smart traffic systems where they are responsible for managing traffic signals. By continuously capturing images of roads and lanes, they provide valuable data that allows for precise control of traffic signals.

2.3 Literature Review

In efforts to improve the control and monitoring of traffic signaling, scientists and researchers have put forth numerous methodologies drawing upon machine vision techniques. The architecture of traffic signal control and monitoring based on computer vision includes various stages, such as image acquisition, preprocessing, and the application of advanced computer vision techniques like density estimation, traffic sign detection and recognition, accident detection, emergency vehicle detection, and the like. This section explores selected papers during recent years that introduce proposed methods for controlling and monitoring traffic signals.

(Kumaran et al., 2019b) introduced a smart traffic signal control system that utilizes computer vision algorithms. They also proposed a new method for traffic signal timing based on traffic flow, using the Temporal Unknown Incremental Clustering (TUIC) model to cluster moving vehicles according to optical flow features. Test results show that their approach leads to reduced waiting times and increased throughput compared to other signal timing algorithms. (Jing et al., 2017) conducted a thorough review of adaptive traffic signal control in congested traffic conditions. Their system effectively reduces urban traffic congestion by minimizing delays. They evaluated performance by comparing different methods based on prior research findings. Moreover, they developed a systematic framework for adaptive traffic signal regulation that uses linked cars to inform future research paths. More research is required to create more widely applicable adaptive traffic signal control algorithms for a connected car environment.

In real-time wireless network simulation, (Faraj et al., 2020) proposed an intelligent microcontroller circuit-based system for controlling cars in traffic congestion. This system is both efficient and cost-effective, offering an improvement over traditional traffic signal systems in terms of managing and controlling traffic flow. Notably, it can dynamically adjust the timing of traffic light signals and rapidly respond to road traffic conditions, particularly during peak hours, with the aim of reducing congestion. The implementation incorporates hardware components such as a server, cameras, and a microcontroller board, along with a wireless network infrastructure connecting traffic lights. Experimental findings demonstrate enhanced accuracy when utilizing YOLOv3 as a machine learning tool and OpenCV as a preprocessing algorithm. Additionally, the system successfully reduces average waiting times at traffic intersections by over 55%.

An efficient reinforcement learning approach for traffic management systems introduced by (Joo et al., 2020). Their method tackles the issue of traffic congestion through an adaptive Traffic Signal Control (TSC) technique, designed to optimize the number of cars crossing an intersection and distribute signals evenly across roads using Q-learning (QL). Their research findings indicate that this proposed method outperforms other studies utilizing QL in terms of reducing waiting times and minimizing the standard deviation of the shortest queue lengths.

(Sharma et al., 2021) presented an intelligent framework for traffic light control systems using deep learning techniques. Their method combines the YOLO deep learning model to identify objects in a video stream with the Simple Online and Real-time Tracking algorithm (SORT) to track these objects across consecutive frames. Their primary focus is on tackling the intricate traffic conditions found on roads in India. The system was tested in real-time traffic situations, showing positive results. Additionally, they efficiently managed traffic issues at a low cost by adjusting signal timings dynamically based on the current road conditions. In cases of heavy traffic congestion in interconnected road networks, the system successfully reduced the risk of total jam. To address this challenge, (Sun et al., 2020) recommended a predictive model using the Hidden Markov Model (HMM) to predict congestion patterns in highly congested traffic areas. This model establishes a connection between observed external traffic conditions and hidden internal traffic conditions in these busy zones. By refining and analyzing floating vehicle trajectory data, the HMM was adjusted to forecast congestion patterns. Experimental results indicate a predictive accuracy of up to 83.4%, outperforming moving average and autoregressive models by 5.8%. These findings highlight the effectiveness and feasibility of this method in predicting congestion patterns.

(Ke et al., 2018) have demonstrated an innovative method for detecting road congestion in intelligent transportation systems, using multidimensional visual characteristics and CNN. The approach starts by determining foreground object density with a gray-level co-occurrence matrix. Next, the Lucas-Kanade optical flow, along with a pyramid scheme, is used to evaluate the speed of moving objects. Subsequently, a Gaussian mixture model is applied for background modeling, enabling accurate identification of the main foreground amidst potential foreground candidates through CNN. The efficacy of the proposed technique is confirmed through thorough quantitative and qualitative evaluations, showing significant improvements over existing road-traffic congestion detection methods due to the integration of multidimensional features via CNN. Regarding traffic flow optimization, this section is subdivided into density estimation, traffic sign detection and recognition, accident detection, and emergency vehicle detection.

2.3.1 Density Estimation

Estimating traffic density is essential for automating traffic signal control and reducing congestion at intersections. The following section provides an overview of the different methods researchers use to estimate traffic density:

(Garg et al., 2016) presented a method for estimating traffic density through vision, which is a fundamental component in traffic monitoring systems. In response to the limitations of current techniques, such as inaccuracies in vehicle counting and tracking, vulnerability to lighting changes, occlusions, and congestion, the authors developed strategies to address these challenges. They also tackled the issue of high computational complexity found in holistic approaches, which hindered real-time implementation. To overcome this hurdle, they suggested a block processing method for calculating density in busy road segments. This method involves two main steps: first, identifying Regions of Interest (ROI), creating Blocks of Interest (BOI), and establishing background; second, using an iterative process that includes updating the background, detecting occupied blocks, removing shadow blocks, and estimating traffic density. The effectiveness of the proposed methods was assessed and verified using the TrafficDB dataset.

According to (Biswas et al., 2017), density estimation is conducted through car counting, employing the Background Subtraction (BS) method and the OverFeat framework. The accuracy of the proposed system is assessed through manual car counting, and a comparative study is carried out before and after the implementation of the OverFeat framework. The results show a significant improvement in accuracy, with an average accuracy of 96.55% achieved after integrating the OverFeat framework, compared to 67.69% for Placemeter and 63.14% for BS, respectively. Additionally, this study demonstrates the versatility of the OverFeat framework, as it is shown to have other applications beyond density estimation. Furthermore, the study examines the advantages and limitations of the BS method, analyzing six individual traffic videos from various perspectives, including camera angles, weather conditions, and time of day, in conjunction with the OverFeat framework.

(Biswas et al., 2019), applied SSD and MobileNet-SSD to estimate traffic density using fifty-nine individual traffic cameras. They evaluated the strengths and weaknesses of these frameworks by comparing their accuracy with manually estimated density. The experiments showed that the SSD framework, in particular, demonstrated significant potential in traffic density estimation, achieving high detection accuracy rates of 92.97% for SSD and 79.30% for MobileNet-SSD.

K.-H. N et al (Bui et al., 2020), developed a method that uses advanced computer vision technologies to analyze traffic flow by extracting data from video surveillance. The technique involves collecting data from video surveillance to estimate traffic density at intersections. YOLO and DeepSORT techniques were utilized to detect, track, and count vehicles, allowing for the calculation of road traffic density. The effectiveness of the method was tested using real-world traffic data obtained from CCTV footage gathered over a day.

A new technique for estimating traffic density utilizing a macroscopic approach has been developed by Kurniawan et al. (2017). The method proposed comprises two main components: background construction and a traffic density estimation algorithm. Background construction involves identifying non-moving vehicles in front of or behind others, while the image background is determined using edge detection techniques. Density estimation is achieved by calculating the ratio between the number of Regions of Interest (ROI) containing objects and the total number of ROI.

(Eamthanakul et al., 2017) introduced an image processing technique for congestion detection comprising three components: (1) image background subtraction to distinguish vehicles from the background, (2) application of morphological techniques to eliminate image noise, and (3) calculation of traffic density based on the obtained image from CCTV. The outcomes of this process are subsequently transmitted to the transport plan database.

Finally, Table 2.2 presents a summary of various studies on traffic density estimation.

Table 2.2 Summary of Different Studies on Traffic Density Estimation

Reference(s)	Algorithm(s)	Dataset(s)	Accuracy %	Contribution(s)
(Garg et al., 2016)	Block Variance	The TrafficDB	93.70	Traffic density estimation with the low computational cost.
(Biswas et al., 2017)	Background Subtraction, Over Feat framework, and Place meter	ImageNet	96.55	Defining ROI by Over Feat framework
(Biswas et al., 2019)	Detection (SSD) and MobileNet-SSD	Data collected from cameras with different places	92.97 (SSD), 79.30 (MobileNet-SSD)	New path opened for real time traffic density estimation
(Bui et al., 2020)	YOLO and DeepSORT	Collected data from CCTV	87,88 (Day, Congestion) 93,88 (Day, Normal) 82,1 (Night, Normal)	Detecting, tracking and counting vehicles
(Kurniawan et al., 2017)	ROI and edge detection	-	N/A	New technique developed for estimating traffic density.
(Eamthanakul et al., 2017)	Background subtraction and Morphological techniques	-	N/A	Traffic density estimated.

2.3.2 Traffic Sign Detection and Recognition

Recognition of traffic signs is integral to driver assistance systems and intelligent autonomous vehicles, contributing significantly to road safety. Additionally, it can facilitate the automation of traffic signals, thereby helping to prevent intersections from being crossed when signals indicate red.

Novel approaches proposed in (Kaplan Berkaya et al., 2016) for the detection and recognition of traffic signs. Specifically, a novel technique called the circle detection algorithm has been developed to identify traffic signs. In addition, RGB-based color thresholding technique was proposed by Kaplan Berkaya et al. (2016). Moreover, three algorithms, namely histogram of oriented gradients, local binary patterns, and Gabor features, were utilized within a Support Vector Machine (SVM) classification framework to recognize traffic signs. The effectiveness of these methods for both detection and recognition was assessed using the German Traffic Sign Detection Benchmark (GTSDB) dataset. According to the experimental findings, the proposed system surpassed existing literature and could be employed in real-time operations.

(Yang et al., 2016) presented a method for detecting and recognizing traffic signs, consisting of three primary steps. Firstly, they use thresholding of HSI color space components to segment the image. Secondly, they use blobs extracted from the previous step to detect traffic signs. A significant aspect of their approach in the initial step is the use of invariant geometric moments for shape classification instead of machine learning algorithms. Thirdly, they propose a novel recognition method inspired by existing features. They extend the Histogram of Oriented Gradients (HOG) features to the HSI color space and combine them with LSS features to create the descriptor. They test Random Forest (RF) and SVM classifiers in conjunction with the new descriptor. The effectiveness of the proposed system is evaluated using the GTSDB and STS datasets, and its performance is compared with that of existing techniques.

Salti et al. (2015), combined solid image analysis and pattern recognition techniques to identify traffic signs in mobile mapping data. Unlike other existing systems that use sliding window detection, their system focuses on extracting regions of interest. Despite facing challenges like varying illumination, partial occlusions, and large scale variations, the system showed strong performance. It targeted the detection of three main categories of traffic signs - mandatory, prohibitory, and danger signs - following the experimental setup of the recent GTSDB competition. The system not only excelled in the online competition but also proved effective when tested on a demanding dataset of Italian signs in mobile mapping, indicating its potential for successful real-world deployment.

(Du et al., 2017) developed a robust and efficient classifier-based detector aimed at achieving fast performance. They introduced two algorithms for detection and classification. Firstly, they proposed aggregate channel features, which are based on three types of features: color features, gradient magnitude, and gradient histograms. Secondly, they presented a boosted trees classifier for a multi-scale and multi-phase detector, based on the Real AdaBoost algorithm. Experimental results from this study demonstrated high average-recall and speed when evaluated on the Daimler, Laboratory for Intelligent & Safe Automobiles (LISA), and Laboratório de Robótica e Automação (LaRA) datasets.

Real-time detection and recognition of traffic signs are crucial for improving the intelligence of smart vehicles. To tackle this challenge, (Shao et al., 2018) proposed a new approach with two main steps. Firstly, images captured from the road scene are converted into grayscale images. Then, Simplified Gabor Wavelets (SGW) filters are used to optimize the parameters of the grayscale images. Additionally, traffic signs are delineated using edge detection to prepare the data for the following step. In the second stage, the maximally stable extremal areas approach is used to identify the region of interest, and the superclass of traffic signs is classified using SVM. CNN is used for subclass classification of traffic signs, using input from simplified Gabor feature maps and the same parameters used in the detection phase. Finally, the proposed method is assessed on the GTSDB and Chinese Traffic Sign Dataset (CTSD) datasets, with experimental results showing rapid processing speed at 6.3 frames per second and high accuracy of 99.43%.

(Kaplan Berkaya et al., 2016), introduced innovative approaches to improve traffic efficiency by enhancing object recognition and problem detection using colorful graphics. They improved two digital image processing methods, the Circle Detection Algorithm and RGB, which rely on simple image segmentation techniques to boost traffic sign detection capabilities. Additionally, they developed a classification framework called the SVM by integrating three main attributes - Gabor features, histogram of oriented gradients, and local binary patterns - into the intelligent system. The effectiveness of their proposed technique was confirmed using the GTSDB datasets. Practical results showed that their technique significantly outperformed the methods mentioned in the paper and performed consistently in real-time operations.

A new method for detecting and identifying traffic signs, as suggested in (Ellahyani et al., 2016) consists of three main steps. Firstly, the image is segmented based on color space components using Hue-Saturation-Intensity (HIS) thresholding. Next, blobs identified in the previous step are used to detect traffic signs. Finally, the recognized traffic signs are classified in the last step. In their research, two different methods are used for sign classification. Initially, forms are classified using invariant geometric moments rather than machine learning methods. New recognition features are then suggested, drawing inspiration from current ones. The HSI colour space, obtained from HOG features, is merged with local self-similarity (LSS) features to provide the descriptor for the new approach. Finally, the suggested method's efficacy is evaluated and tested against the German Traffic Sign Recognition Benchmark (GTSRB), German Traffic Sign Detection Benchmark (GTSDB), and Swedish Traffic Signs (STS) datasets.

The CNN machine learning algorithm is known for its high effectiveness in object recognition, thanks to its superior recognition rate and efficient execution time. In a study by (Shustanov and Yakimov, 2017) traffic sign recognition was implemented using CNN, where different CNN architectures were compared. The training was conducted using the TensorFlow library, and a massively parallel architecture was adopted for multithreaded programming with Compute Unified Device Architecture (CUDA). The entire process of detecting and recognizing traffic signs was carried out in real-time on a mobile Graphics Processing Unit (GPU). The method's efficiency was then assessed using the GTSRB dataset, resulting in an impressive classification accuracy of 99.94% for images.

Table 2.3 provides a summary of various studies that focus on different approaches for detecting and recognizing traffic signs.

Table 2.3 Summary of Various Studies on the Detection and Recognition of Traffic Signs.

Reference(s)	Algorithm(s)	Dataset(s)	Accuracy %	Contribution(s)
(Yang et al., 2016)	HSI, HOG, LSS and SVM	GTSDB, CTSD	98.24 (GTSDB), 98.77 (CTSD)	Developed Circle detection algorithm and an RGB-based color thresholding technique.
(Kaplan Berkaya et al., 2016)	HOG, LSS, Random Forest and SVM	GTSDB	97.04	In the first step, machine learning algorithms not used classify shapes instead of this invariant geometric moments have been used. Second, method has been proposed for the recognition.
(Salti et al., 2015)	ROI, HOG, SVM and Context Aware Filter	GTSDB	99.43 (Prohibitory) 95.01 (Mandatory) 97.22 (Danger)	On-line detecting mandatory, prohibitory and danger traffic signs
(Du et al., 2017)	Aggregate Channel Features and Boosted Trees Classifier	Daimler, LISA and LaRA	84.314 (Daimler), 90.33 (LISA),	Proposed the high average-recall and speed method

			92.048 (LaRA)	
(Ellahyani et al., 2016)	HOG, LSS and SVM	GTSRB, GTSDB and TST	97.43	Shapes classified by using invariant geometric moments
(Shao et al., 2018)	SGW and SVM	GTSDB and CTSD	99.43	Speed of detection and classification improved which is more than 6 frames per second
(Shustanov and Yakimov, 2017)	CNN	GTSRB	99.94	CNN process described
(Liu et al., 2021)	Proposed model named CapsNet	TL_Dataset		The proposed CapsNet is employed for traffic sign recognition.

2.3.3 Accident Detection

One crucial aspect of traffic monitoring beside of others is identifying and tracking vehicles, which helps in reporting and detecting incidents at traffic junctions. This section also covers methods for predicting and detecting accidents.

(Tian et al., 2019) developed a Cooperative Vehicle Infrastructure Systems (CVIS) and introduced a machine-based vision system capable of automatically detecting car accidents. The study consisted of two phases: In the first phase, the CAD-CVIS database was created to improve the accuracy of accident detection. This database, CAD-CVIS, includes various types of accidents, weather conditions, and accident locations, representing different traffic scenarios. In the second phase, a deep neural network model named YOLO-

CA, based on CAD-CVIS and deep learning algorithms, was created for accident detection. Moreover, to enhance the model's performance in detecting small objects, Multi-Scale Feature Fusion (MSFF) and a loss function with dynamic weights were utilized. The results showed that the proposed method surpassed previous approaches, being able to detect car accidents within milliseconds with a very high average precision of 90.02%. Finally, the proposed method was compared with existing approaches, demonstrating improved accuracy and real-time performance compared to other models.

Neoteric framework is presented in (Ijjina et al., 2019), for accident detection. This framework introduces the use of Mask R-CNN for accurate object detection and is supported by a centroid-based object tracking algorithm for surveillance footage efficiency. The main idea is to identify accidents by spotting irregularities in vehicle speed and trajectory once vehicles intersect. This framework proves to be superior and paves the way for real-time versatile vehicular accident detection algorithm development. Performance evaluation and validation of this framework were conducted using a dataset with diverse weather conditions.

(Saini et al., 2017), suggested a novel vehicle tracking technique utilizing image processing, which eliminates the need for background subtraction to delineate the region of interest. Instead, the study advocates for a hybrid methodology that integrates feature detection and region matching, facilitating the estimation of vehicle trajectories across successive frames. As vehicles traverse through an intersection, the system monitors the tracked direction for any potential events. The research concludes that the proposed method exhibits proficiency in detecting accidents involving two vehicles.

According to (Wenqi et al., 2017), the TAP-CNN model was introduced for accident prediction on highways by utilizing convolutional neural networks. This model combines traffic state and CNN architecture to create a state matrix that includes various accident factors like traffic flow, weather, and lighting. The researchers also investigated ways to improve the TAP-CNN model's accuracy through multiple iterations. They gathered accident data for training purposes and to assess the model's performance. The experimental results conclusively show that the TAP-CNN model outperforms conventional neural network models in accurately predicting traffic accidents.

(Dogru and Subasi, 2018) proposed an intelligent accident detection system in which automobiles share microscopic vehicle variables. Using vehicle speed and coordinates obtained from Vehicular Ad-Hoc Networks (VANETs), data are collected and simulated in the proposed system, which then sends traffic alerts to drivers. The study also demonstrates the utilization of machine learning techniques for accident detection on freeways within ITS. Position and velocity values of each vehicle serve as crucial parameters for easy accident analysis and detection. Moreover, the proposed method is evaluated using the OOB dataset, with results indicating that the RF algorithm outperforms ANN and SVM algorithms, achieving accuracies of 91.56%, 88.71%, and 90.02%, respectively.

As can be seen in (Yu et al., 2019), vision-based algorithms are used to detect traffic accidents by applying an ST-IHT algorithm to improve the robustness and sparsity of spatio-temporal features. Furthermore, a weighted extreme learning machine detector is used to distinguish between traffic accidents and normal traffic. The study also presents a two-point search technique designed to dynamically locate candidate values for Lipschitz coefficients to improve tuning accuracy. To assess the efficiency of the suggested approach, 30 traffic videos from the YouTube website are used for testing and evaluation. The results indicate that the suggested technique surpasses current approaches in terms of traffic accident detection performance.

The accelerometer method is widely utilized for crash detection. In this respect (Borisagar et al., 2018) state that the accelerometer values undergo calibration to detect accidents based on acceleration. However, due to the limitations in accelerometer accuracy and the need for efficient accident detection, the researchers turned to the CNN machine learning algorithm. While image classification techniques are typically used for accident detection, CNNs require significant time, data, and computational resources for training. To address these challenges, transfer learning techniques were creatively employed. This involved retraining a pre-trained network, specifically the Inception-v3 classifier developed by Google for image tasks for accident detection purposes. The efficiency of the proposed method was then compared to traditional accelerometer-based techniques, resulting in an accuracy of 84.5% for the Transfer Learning algorithm.

Summary of Various Studies Conducted in the Field of Accident Detection Presented in Table 2.4

Table 2.4 Summary of Studies Conducted in the Field of Accident Detection

Reference(s)	Algorithm(s)	Dataset(s)	Accuracy %	Contribution(s)
(Tian et al., 2019)	Deep neural network model YOLO-CA	CAD-CVIS	90.02	CAD-CVIS dataset created and the proposed method more fast and accurate.
(Ijjina et al., 2019)	Mask R-CNN	Proposed	71	Developing vehicular accident detection algorithms in real-time.
(Saini et al., 2017)	Hybrid of feature detection and Region matching	Real world dataset	N/A	Accident detection between two vehicles
(Wenqi et al., 2017)	CNN	Accident data collected	78.5	Accident predicted by using CNN
(Dogru and Subasi, 2018)	ANN, SVM and Random Forests (RF)	OOB data set	91.56 (RF), 88.71 (ANN), 90.02 (SVM)	The proposed method can provide estimated geographical location of the possible accident
(Yu et al., 2019)	ST-IHT, Spatio-Temporal Features and W-ELM	Collected dataset	87.4 ± 0.3 (SVM), 94.3 ± 0.2 (ELM), 95.5 ± 0.3 (W-ELM)	(i) Robust Fractures extraction proposed based on OF-DSIFT and ST-IHT ii) detect imbalance between traffic accident and normal traffic
(Ghahremannezhad et al., 2022)	YOLOv4	video sequences collected from YouTube	N/A	presents a new efficient framework for accident detection

2.3.4 Emergency Vehicles Detection

The success of law enforcement and public safety is the timely arrival of first responders at emergency scenes. These responders usually consist of ambulance, firefighter, and police vehicles. The following section reviews several suggested techniques for detecting these emergency vehicles.

(Raji et al., 2022) have developed an innovative strategy to manage emergency situations, especially in congested traffic scenarios. Addressing the problem of blocked emergency vehicle movement during peak traffic hours, the system successfully improves timing inefficiencies and reduces traffic congestion. As an emergency vehicle moves through a specific lane, a Radio-Frequency Identification (RFID) transmitter captures and transmits signal data. This allows an RFID receiver to change the traffic signal from red to green, thus clearing the lane for the emergency vehicle. To adapt to changes in traffic density, the system dynamically adjusts signal timing intervals to either 10 or 6 seconds. This adjustment leads to decreased congestion, reduced travel time, and potentially saving lives. By utilizing RFID technology for accurate motion detection and identification, the proposed system can consistently evaluate vehicular density and grant automatic priority to emergency vehicles. Both traditional and deep neural networks are commonly used to classify regular and emergency vehicles. In this respect, (bin Che Mansor et al., 2021) presented a classification technique specifically designed to identify emergency vehicles that frequently get stuck in congested traffic areas. Detecting emergency vehicles on city roads can help improve their prompt arrival. They employed the VGG-16 model as a pre-trained base, adjusting the convolutional layer and filter size to boost performance. The experimental results showed that the proposed method achieved an accuracy rate of 95%. In (Haque et al., 2022), developed and implemented an automated system to identify emergency vehicles, distinguishing them from non-emergency vehicles. They used YOLOv4 for initial object detection with the ROI strategy, then trained the detected objects using CNN and VGG-16 by fine-tuning the model parameters. The system reached an average accuracy of 82.03% when tested on the Emergency Vehicle Identification v1 dataset.

According to (Kaushik et al., 2020), two computer vision techniques are used to identify and locate emergency vehicles. These methods include object detection and instance segmentation. More precisely, the process includes using Faster RCNN for object detection and Mask RCNN for instance segmentation. The results demonstrate the proposed approach's effectiveness, particularly its accuracy and suitability for detecting emergency vehicles in chaotic traffic conditions. Furthermore, a custom dataset of 400 images was employed for emergency vehicle detection, carefully labeled using the LabeMe tool.

(Roy and Rahman, 2019) have developed a model aiming at identifying emergency vehicles like ambulances and fire trucks in crowded road CCTV footage. This model gives priority to these vehicles, ensuring the emergency lane is cleared to help them pass through traffic intersections smoothly. When traffic police encounter difficulties in determining which lanes to open for emergency vehicles, this model provides an automated solution. By employing deep convolutional neural networks and the Common Objects in Context (COCO) dataset, the method they propose shows promising results in effectively detecting and recognizing different types of emergency vehicles.

(Jonnadula and Khilar, 2020b) introduced a hybrid architecture for emergency vehicle detection, blending image processing and computer vision elements. They also reduce the search space by using region of interest techniques.

To reduce casualties in road emergencies, (Lin et al., 2020) introduced a novel approach that utilizes machine learning techniques. Various features are extracted through multi-faceted methods to accurately represent ambulance characteristics. The effectiveness of predicting next-day demand was tested through experiments involving cutting-edge machine learning techniques and ambulance demand prediction methodologies. This testing was conducted using actual ambulance and demographic data from Singapore. Additionally, the accuracy of the proposed method was validated across different machine learning techniques and data types, using the SCDF-Engineered-Socio dataset.

The current traffic light system often lacks responsiveness during emergencies involving ambulances, firefighters, and police vehicles. In response to this issue, (Suhaimy et al., 2020) have developed an embedded machine learning application. This application involves data acquisition, feature extraction, exploration of various algorithms, tuning, and model deployment to achieve optimal performance in a simulation environment. Specifically, they created a classifier for ambulance siren sounds, sorting them into 'Ambulance Arrive' and 'No Ambulance Arrive' categories. This classifier allows the traffic light system to detect and monitor ambulance arrivals during emergencies. The approach proposed utilizes Mel-frequency Spectral Coefficients Combined with Support Vector Machine (MFCC-SVM) implemented on MATLAB R2017b. Additionally, other researchers have investigated incorporating optimization algorithms into deep learning models in similar research efforts.

(Alhudhaif et al., 2022) utilized a hybrid approach that combined a pre-trained CNN GoogleNet and particle swarm optimization (PSO) - an algorithm inspired by nature - to classify autonomous vehicles. They trained the model using a Kaggle dataset comprising vehicle images that were enhanced through various transformations. Subsequently, the model underwent classification using different classifiers, with the Cubic Support Vector Machine (CSVM) emerging as the most effective model. The CSVM exhibited superior performance in terms of both time efficiency and accuracy, achieving an impressive accuracy rate of 94.8%. Both empirical and statistical evaluations confirm the model's superiority over similar approaches, not only in accuracy (94.8%) but also in training duration (82.7 seconds) and speed in forecasting

(380 observations per second).

The studies conducted in the field of emergency vehicle detection are summarized in Table 2.5.

Table 2.5 Summary of Studies Conducted on Emergency Vehicle Detection

Reference(s)	Algorithm(s)	Dataset(s)	Accuracy %	Contribution(s)
(Kaushik et al., 2020)	Faster RCNN and Mask RCNN	Custom dataset	81 (Object Detection), 92 (Iinstance Segmentation)	The computational and accuracy for emergency vehicle detection are suitable
(Roy and Rahman, 2019)	Deep convolutional neural network	COCO	97.97	Detecting and identifying all kinds emergency cars
(Jonnadula and Khilar, 2020a)	YOLO + ResNet	COCO	N/A	Hybrid architecture presented for detection of emergency vehicles in a real time
(Lin et al., 2020)	SVR, MLP, RBFN, and LightGBM	SCDF-Engineered-Socio	N/A	Varying degrees to the model training in LightGBM
(Suhaimy et al., 2020)	MFCC-SVM	-	97	Effectively distinguish audio events from audio signals

2.3.5 Transfer Learning and Optimization Techniques

Researchers have conducted some studies in order to improve the accuracy of emergency vehicle detection. Their main focus has been on models that utilize deep learning techniques. Also, they use of transfer learning for traffic signal systems and reducing traffic congestion. In this respect, some other researchers have conducted researches of using optimization algorithms in deep learning models. (Alhudhaif et al., 2022) proposed a method that combines a pre-trained CNN GoogleNet with particle swarm optimization (PSO), a nature-inspired optimization algorithm, to classify autonomous vehicles. The model was trained using a Kaggle dataset that included vehicle images enhanced with various transformations. After training, the model was subjected to classification using different classifiers, with the Cubic support vector machine (CSVM) proving to be the most effective. It exhibited superior performance in terms of both time efficiency and accuracy, achieving an accuracy rate of 94.8%. The results of empirical and statistical evaluations clearly demonstrate that this model not only surpassed similar approaches in terms of accuracy (94.8%) but also excelled in training duration (82.7 seconds) and speed forecasting (380 observations per second).

In another work, (Haque et al., 2022) automated an approach which is deployed to detect emergency vehicles. Ambulances and fire trucks are categorized as emergencies, while other vehicles are considered non-emergency. The process begins by identifying multiple vehicles within an image using the YOLOv4 object detector. These identified vehicles are then further investigated. Also, the method distinguishes between emergency and non-emergency vehicles. Finally, the research contributes by developing a model that incorporates a convolutional neural network (CNN) with a viral algorithm in deep learning. Transfer learning with a fine-tuned VGG16 model is also utilized for emergency vehicle detection. On the Emergency Vehicle Identification v1 dataset, this model achieves an average accuracy of 82.03%.

CHAPTER THREE:

3. METHODOLOGY, RESEARCH DESIGN, MATERIALS AND METHODS

3.1 Introduction

This chapter introduces a comprehensive methodology that specifically addresses vehicle classification and object detection using advanced deep learning techniques. The primary objective is to enhance accuracy and offer readers a practical guide for efficient research design and material utilization. Moreover, this chapter provides detailed explanations of the methods used and the intricate modifications applied to deep learning techniques. Additionally, the chapter offers thorough explanations of the employed methods and the modifications made to deep learning techniques.

3.2 Research Framework

The process of implementing research is divided into five distinct phases, each with its own objectives and tasks. Visual representations are utilized to enhance the clarity and flow of each phase within the framework. Table 3.1 presents a block diagram that provides an overview of the entire study framework.

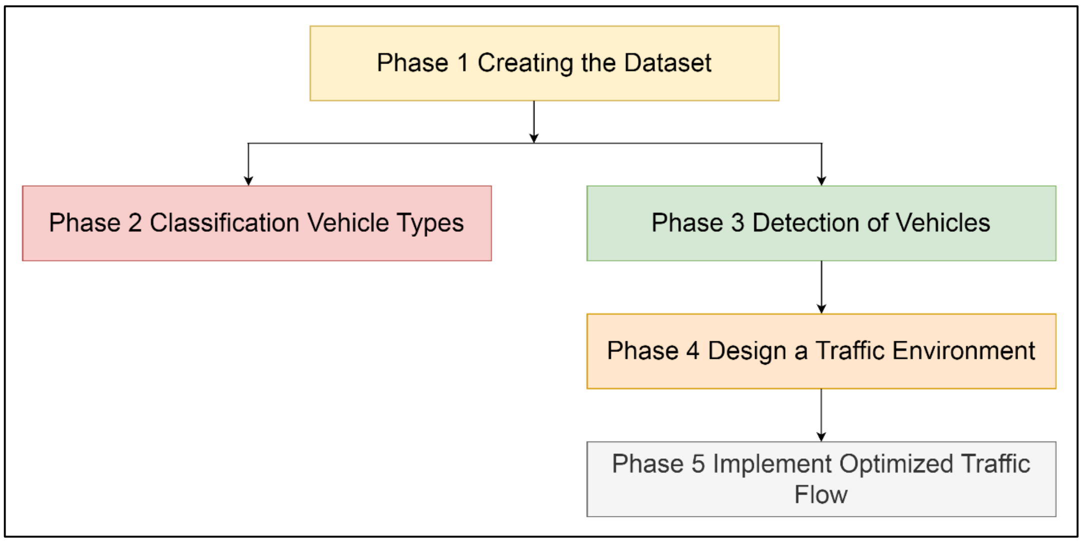

Phase 1 Creating the Dataset
Phase 2 Classification Vehicle Types
Phase 3 Detection of Vehicles
Phase 4 Design a Traffic Environment
Phase 5 Implement Optimized Traffic Flow

Figure 3.1 General Framework of Proposed System

Figure 3.2 specifically illustrates Phase 1, where the customized dataset is created, highlighting the inclusion of various vehicle types such as police cars, ambulances, firefighters, and non-emergency vehicles.

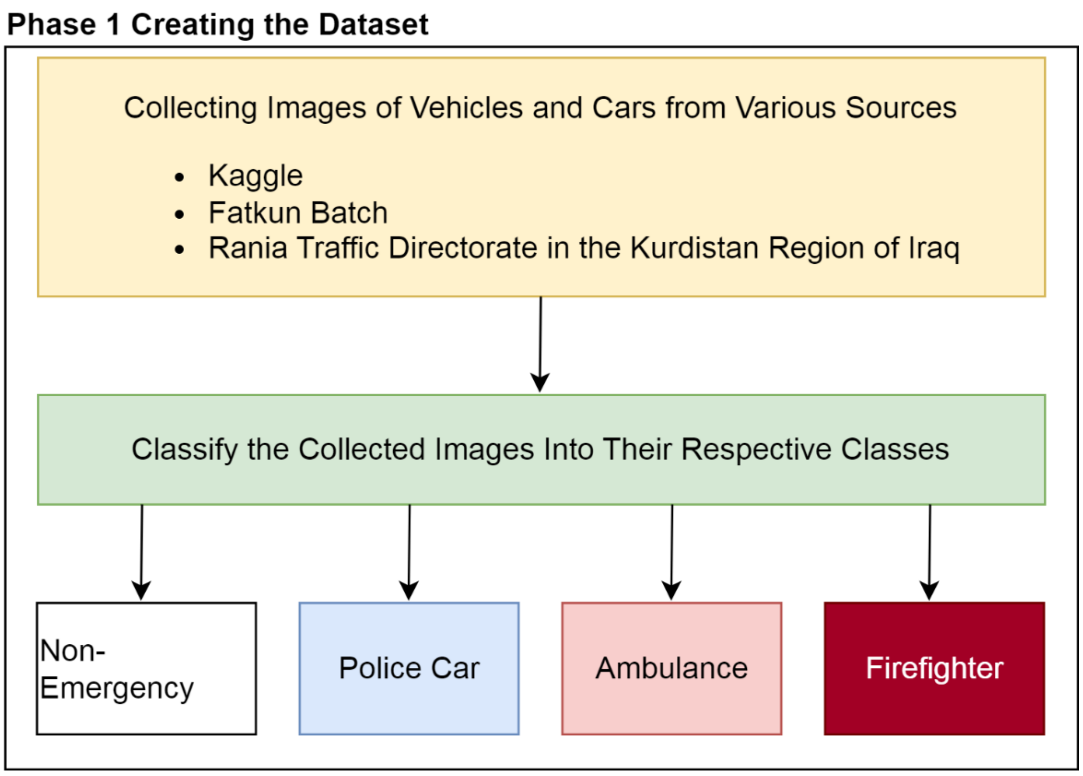

Phase 1 Creating the Dataset
Collecting Images of Vehicles and Cars from Various Sources
Kaggle
Fatkun Batch
Rania Traffic Directorate in the Kurdistan Region of Iraq
Classify the Collected Images Into Their Respective Classes
Non-Emergency
Police Car
Ambulance
Firefighter

Figure 3.2 Phase 1 Creating the Dataset

Phase 2 illustrates the development of the classification system. This system utilizes a modified deep learning technique to accurately distinguish the different types of vehicles shown in Figure 3.3.

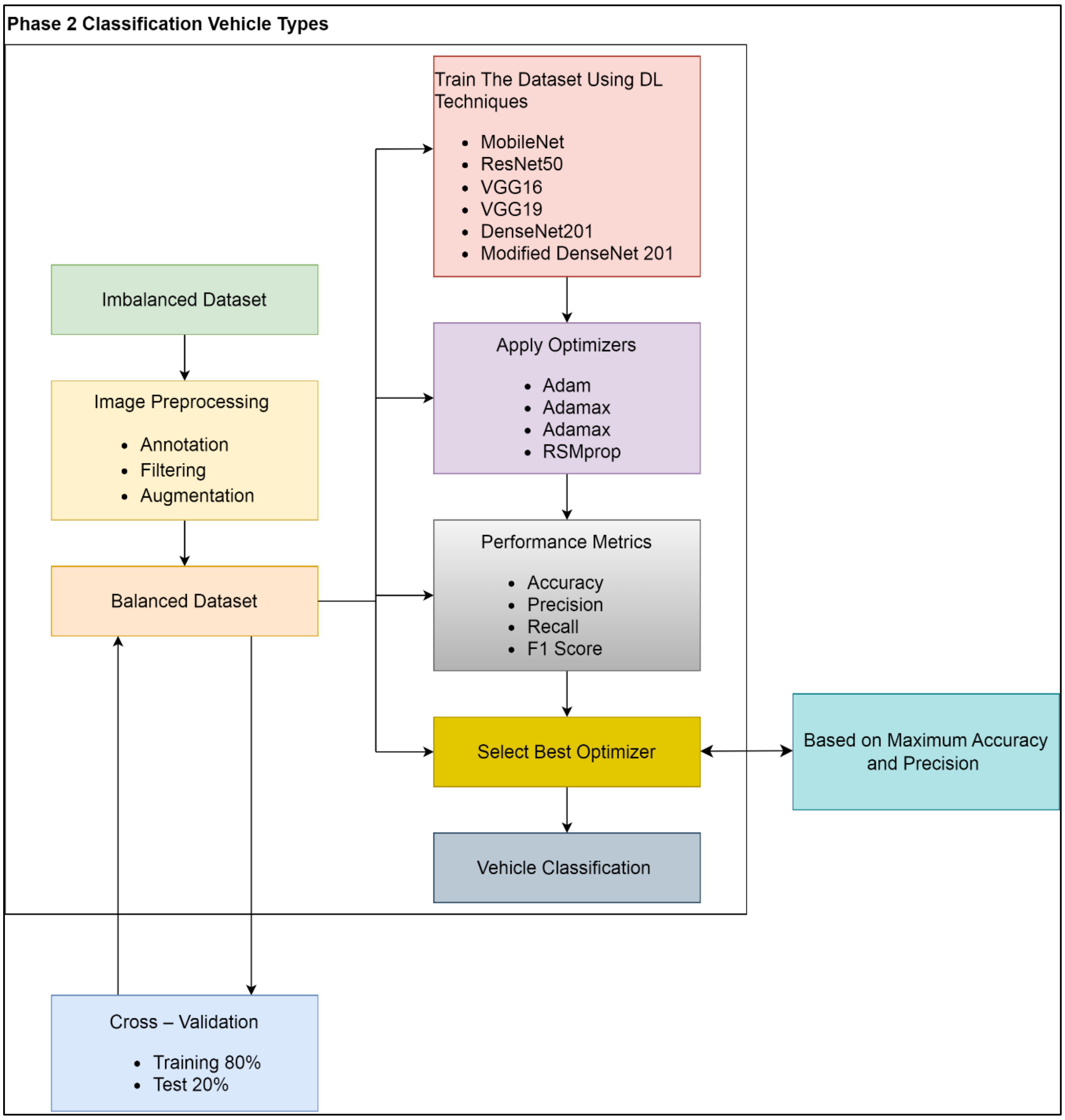


Figure 3.3 Phase 2 Classification Vehicle Types

Figure 3.4 represents Phase 3, which focuses on refining the vehicle detection system using YOLOv5. This system plays a vital role in the decision-making process for traffic signals.

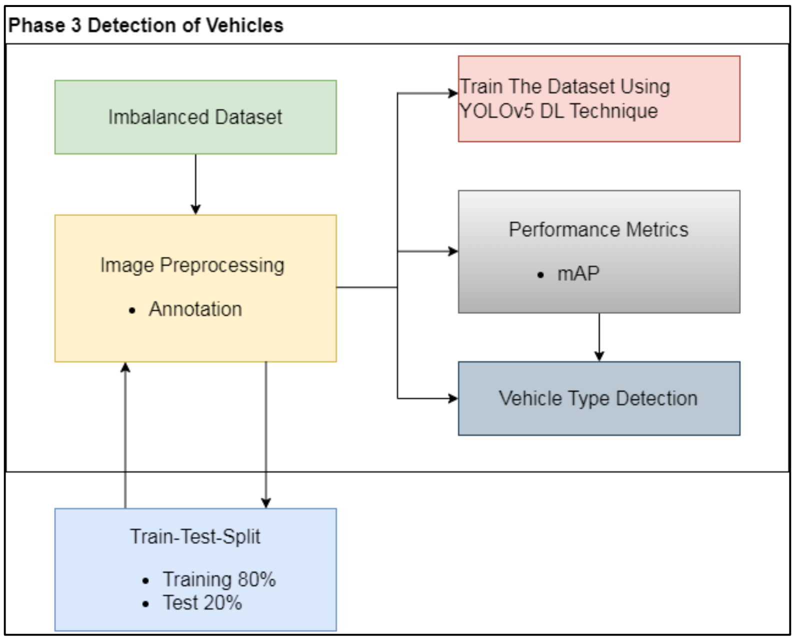


Figure 3.4 Phase 3 Detection of Vehicles

Figure 3.5 details Phase 4, where the simulation model is designed. This figure illustrates the various components and requirements incorporated into the simulated traffic environment.

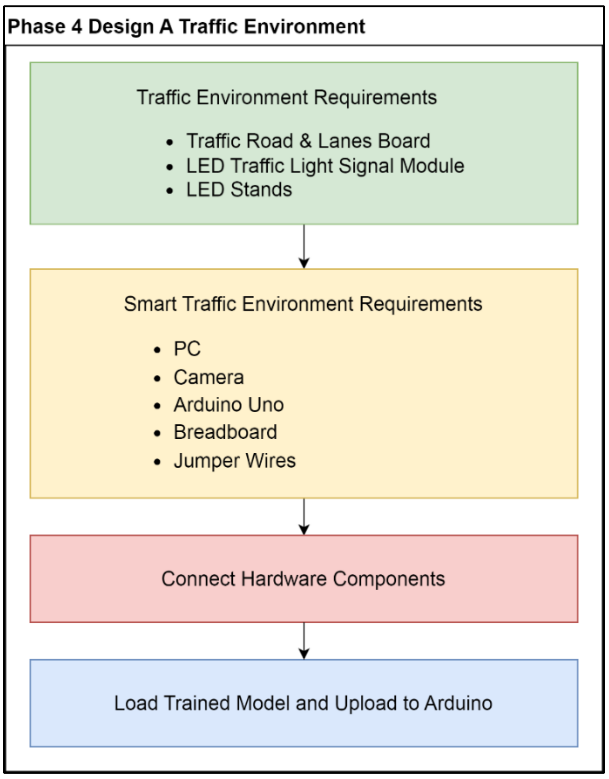
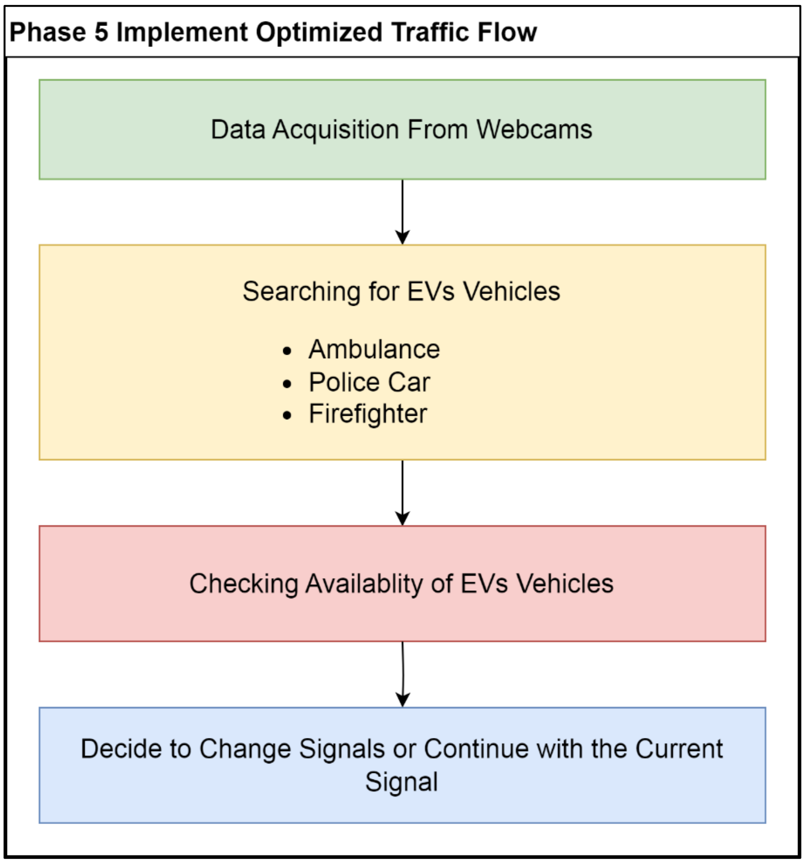

Phase 4 Design A Traffic Environment
Traffic Environment Requirements
Traffic Road & Lanes Board
LED Traffic Light Signal Module
LED Stands
Smart Traffic Environment Requirements
PC
Camera
Arduino Uno
Breadboard
Jumper Wires
Connect Hardware Components
Load Trained Model and Upload to Arduino

Figure 3.5 Phase 4 Design a Traffic Environment

Phase 5 is dedicated to showcasing the implementation of the optimized traffic signal system. Its primary objective is to improve traffic flow and prioritize emergency vehicles, as depicted in Figure 3.6.



Phase 5 Implement Optimized Traffic Flow
Data Acquisition From Webcams
Searching for EVs Vehicles
• Ambulance
• Police Car
• Firefighter
Checking Availablity of EVs Vehicles
Decide to Change Signals or Continue with the Current Signal

Figure 3.6 Phase 5 Implement Optimized Traffic Flow

3.2.1 Creating a New Dataset

This research used a dataset that included images of emergency and non-emergency vehicles. Since there was no existing dataset available specifically for emergency vehicles like police cars, ambulances, and firefighters, a custom dataset was created for this study. The vehicle images were gathered from different sources, including Kaggle (www.kaggle.com), Fatkun Batch, and the

Rania traffic directorate in the Kurdistan region of Iraq. It is important to mention that the vehicle images vary in dimension and have an unbalanced distribution across different classes. For more information about the dataset for emergency and non-emergency vehicles, please refer to Table 3.1.

Table 3.1Unbalanced Datasets

Vehicle types	Total number of images
Ambulance	322
Firefighters	526
Police car	700
Non-emergency	1670
Total	3218

3.2.2 Data Preprocessing

Preprocessing algorithms involve a range of techniques that aim to improve the quality of images through specific operations. The initial steps include resizing the images to meet model requirements, such as dimensions of 64x64, 128x128, and 224x224. After resizing, the images are categorized into emergency and non-emergency vehicle classes. Often, the quality of the images is affected by factors like electronic device effects and lighting conditions. Therefore, implementing preprocessing algorithms is crucial for preparing the images for the classification process.

This study highlights the importance of image sharpening, smoothing, and contrast enhancement in enhancing image quality. These enhancements provide essential support for downstream tasks such as image segmentation, detection, and classification. For these purposes, Albumentations library python data augmentation is used for balancing the dataset. Albumentations library has 12 transformations which can be applied to easy and fast augmentation. Table 3.2 shows a balanced dataset consisting of both emergency and non-emergency vehicles.

Table 3.2 Used Preprocessing Techniques

Vehicle types	Used Techniques
Ambulance	Vertical Flip, Sharpen Horizontal Flip, Sharpen Sharpen, Random Brightness Contrast Random Brightness Contrast, Median Blur
Firefighters	Vertical Flip, Sharpen Horizontal Flip, Sharpen Sharpen, Random Brightness Contrast
Police car	Horizontal Flip, Sharpen

Because our dataset is unbalanced, balancing scaling is used for balance classes and is shown in Table 3.3. The obtained dataset size after the augmentation process is 6222 images (An example of augmented data for Ambulance, Firefighters, and Police Car is shown in Figures 3.7, 3.8, and 3.9, respectively). The results achieved by applying preprocessing techniques to balance the dataset collected for this study are presented in Table 3.3.

Table 3.3 Balanced Datasets

Vehicle types	Balancing Scale	Total number of Images
Ambulance	5.19844358	1610
Firefighters	3.165876777	1682
Police car	2.385714286	1260
Non-emergency	1	1670
Total		6222

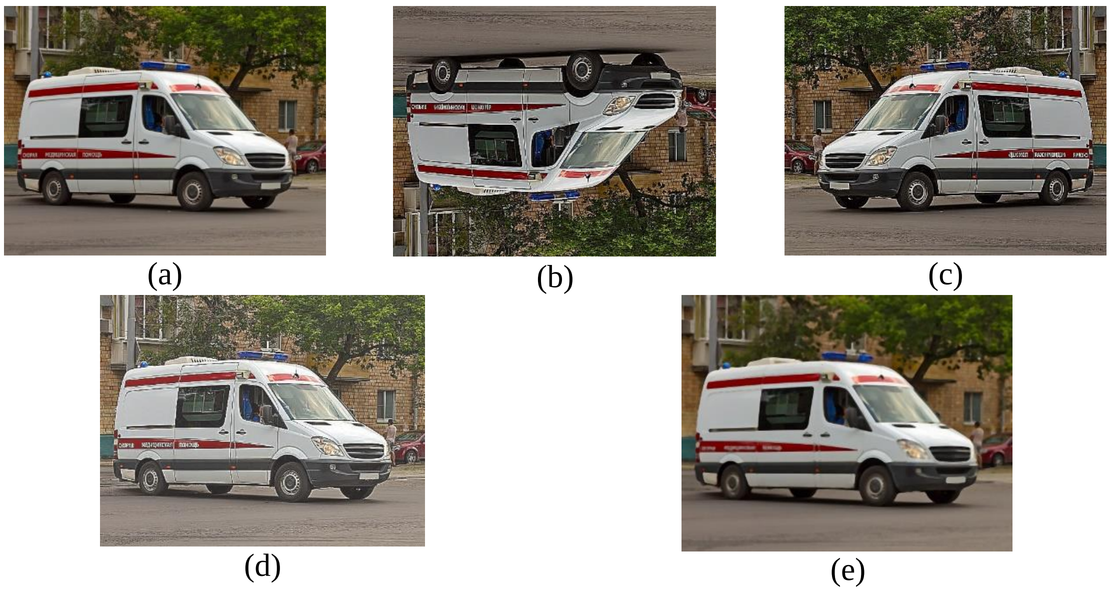


Figure 3.7 Augmented data (Ambulance): (a) Original Image, (b) Vertical Flip, Sharpen, (c) Horizontal Flip, Sharpen, (d) Sharpen, Random Brightness Contrast, (e) Random Brightness Contrast, Median Blur

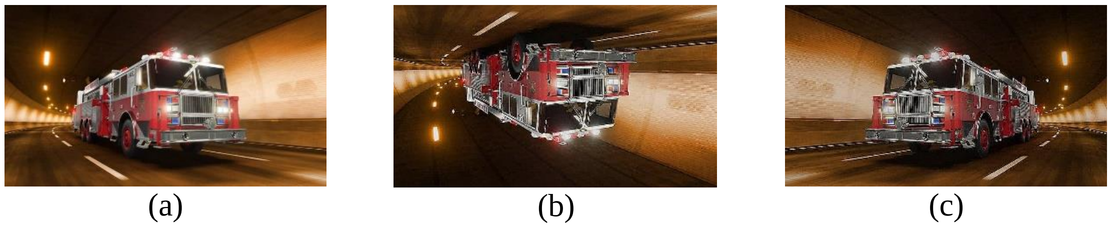


Figure 3.8 Augmented data (Firefighters): (a) Original Image, (b) Vertical Flip, Sharpen, (c) Horizontal Flip, Sharpen

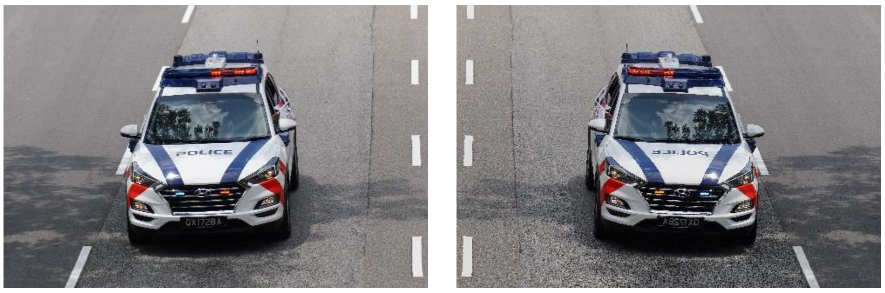

(a) (b)

Figure 3.9 Augmented data (Police): (a) Original Image, (b) Horizontal Flip, Sharpen

3.2.3 Image Data Annotation

Annotations play a crucial role in object detection tasks. There are several free tools available for annotating datasets, such as MakeSense, LabelImg, CVAT, LabelMe, VoTT, ImgLab, and some more. For this study, the MakeSense online tool was used to label and annotate the images. The annotated information is stored in both text and XML files, which can be used with various object detection techniques like YOLO, R-CNN, SSD, and others. Figure 3.10 provides an illustrative example of image labeling.

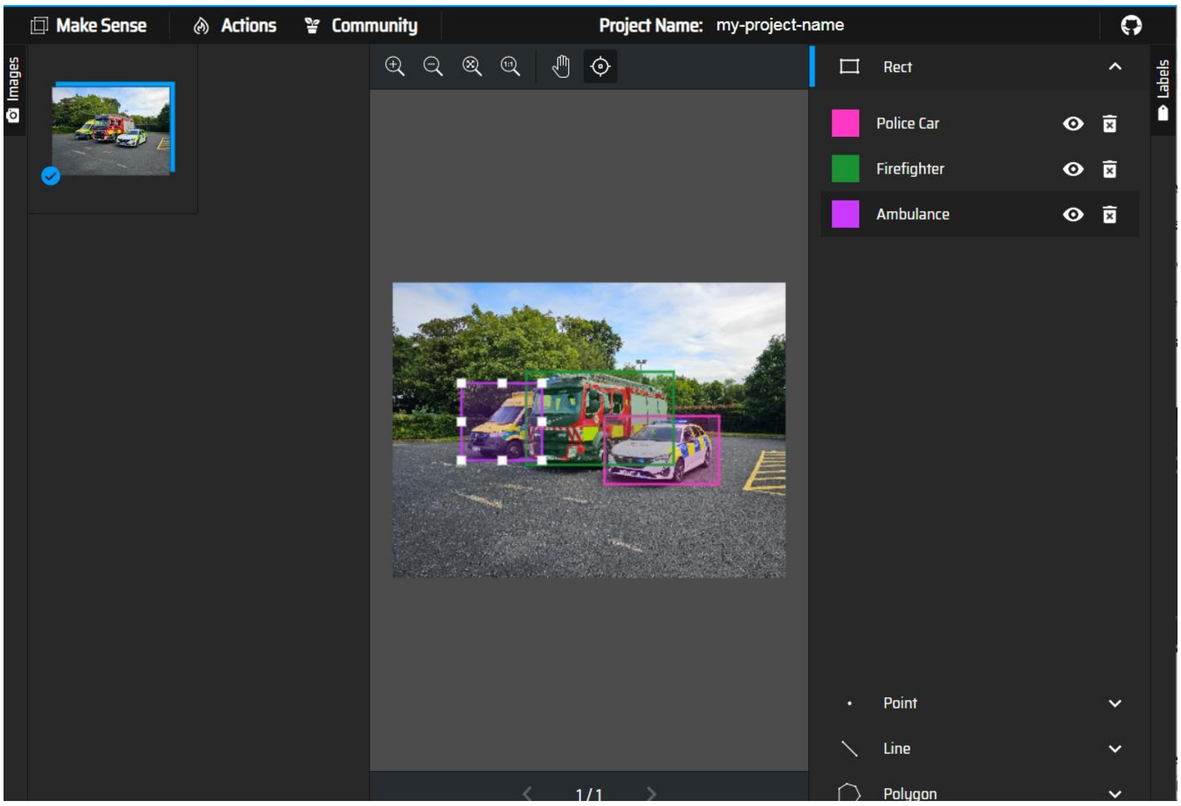


Figure 3.10 Image Labeling

3.3 Vehicle Types Classification

To improve the accuracy of the vehicle classification system, several classification techniques are applied to the balanced dataset. These techniques encompass ResNET, MobileNet, VGG16, VGG19, DenseNet201. Transfer learning was valuable in object classification, particularly when working with limited datasets, as it provided valuable insights. A visual representation of the proposed system’s classification can be seen in Figure 3.11. This study successfully employed a variety of deep learning techniques to precisely classify vehicles as they pass through a traffic intersection.

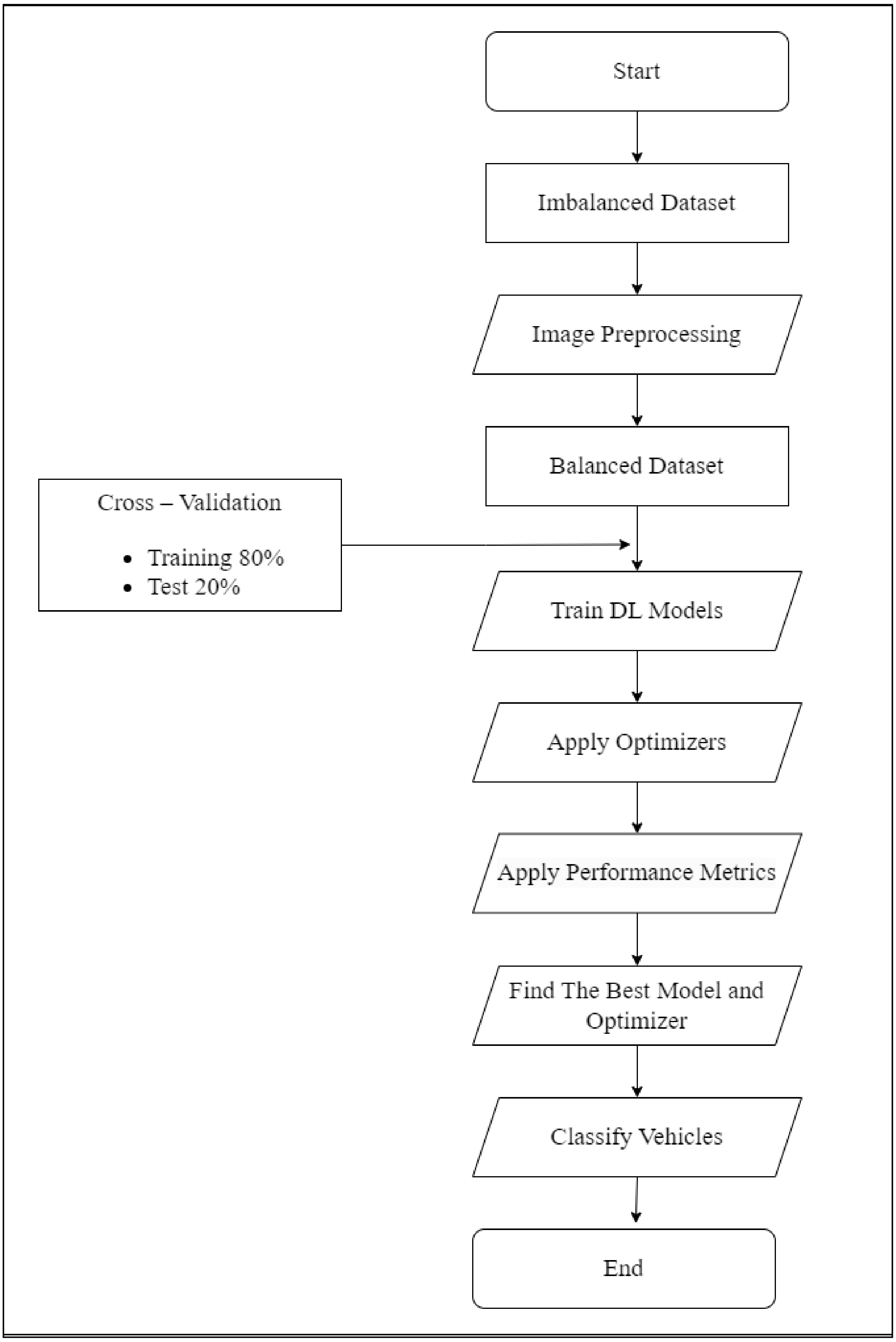


Figure 3.11 Vehicle Classification Process

Table 3.4 showcases the top results attained, along with the optimizers utilized for this dataset.

Table 3.4 Results of DL Techniques

Techniques	Optimizer	Accuracy (%)
VGG16	RMSprop	95
MobileNet	RMSprop	95.3
ResNet	RMSprop	95.3
DenseNet	Adam	97

Based on the results obtained from the tests and the selection of the best technique for vehicle classification, DenseNet has been chosen for modification.

Furthermore, fine-tuning the deep transfer learning DenseNet201-based model can be employed to further improve the results. This process involves repurposing a pre-trained model, allowing the application of knowledge gained from a larger dataset to a smaller one, as depicted in Figure 3.12. The methodologies utilized in this study encompass various processes, as illustrated in Figure 3.13. The block diagram of DenseNet 201 modification comprises the following stages:

1. Data Compilation: images are collected from publicly available datasets and local traffic offices in the KRG, Iraq, to compile the data on vehicles.
2. Image Labeling: Vehicles are categorized into emergency (Ambulance, Police, and Firefighter) and non-emergency groups through annotation. This results in four types of vehicles that are used for model training.

3. Improvement of Image Quality: Preprocessing algorithms are applied to enhance the quality of the images. This includes resizing, sharpening, smoothing, and contrast enhancement.
4. Data Augmentation: Various image transformations are utilized to address overfitting and balance the datasets.
5. Dataset Partitioning: The data is divided into training and testing/validation sets for cross-validation. 80% of the data is allocated for training, while 20% is reserved for testing and validation.
6. Proposed Transfer Model Training: The model is constructed and certain parameters are adjusted for training.
7. Vehicles Classification: Vehicles are classified into emergency and non-emergency categories.
8. Evaluation of Performance Metrics: Performance is assessed using loss-accuracy curves, a confusion matrix, precision, recall, F1-score, and average accuracy.

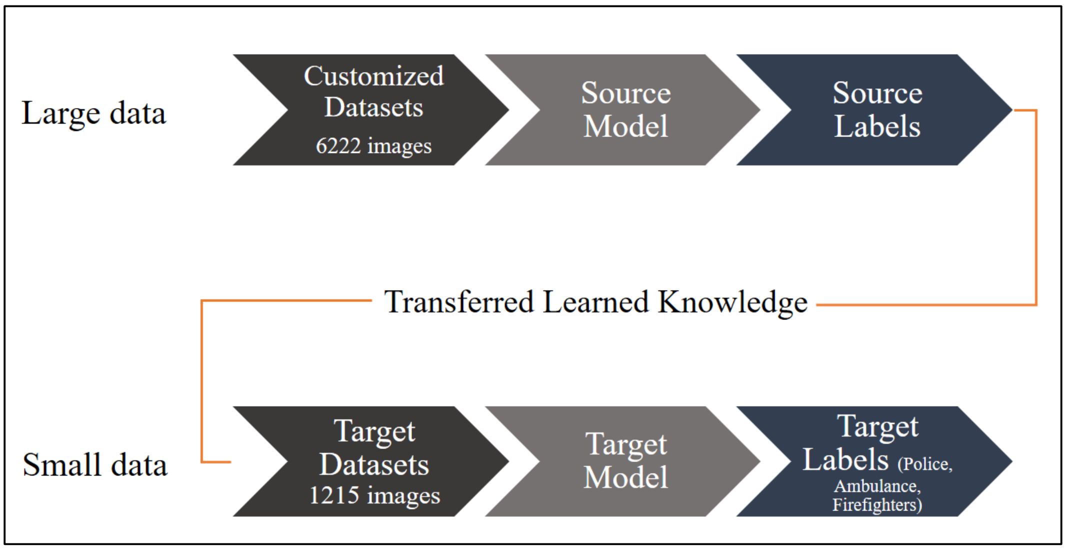

Large data
Customized Datasets 6222 images
Source Model
Source Labels
Transferred Learned Knowledge
Small data
Target Datasets 1215 images
Target Model
Target Labels (Police, Ambulance, Firefighters)

Figure 3.12 Transfer knowledge learning-based processes

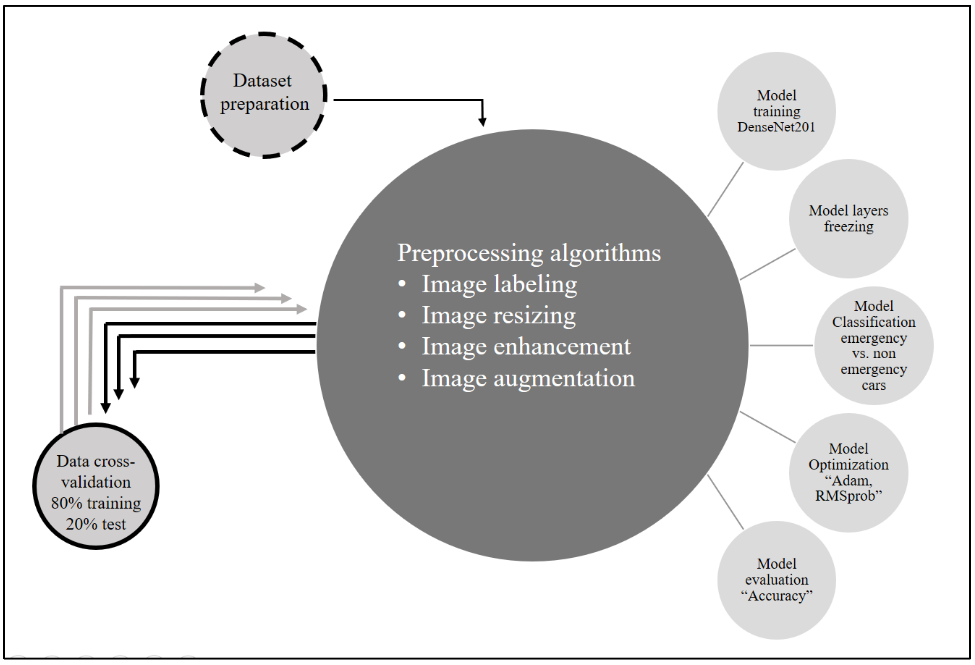

Dataset preparation
Preprocessing algorithms
Image labeling
Image resizing
Image enhancement
Image augmentation
Data cross-validation 80% training 20% test
Model training DenseNet201
Model layers freezing
Model Classification emergency vs. non emergency cars
Model Optimization "Adam, RMSprob"
Model evaluation "Accuracy"

Figure 3.13 Steps of DenseNet201 Modification

3.3.1 DenseNet201

The proposed model uses the DenseNet201 for classification. The design and implementation of the proposed approach consists of three steps: apart of preprocessing, feature extraction, classification, and optimization. Figure 3.14 displays the stages of the proposed model architectures.

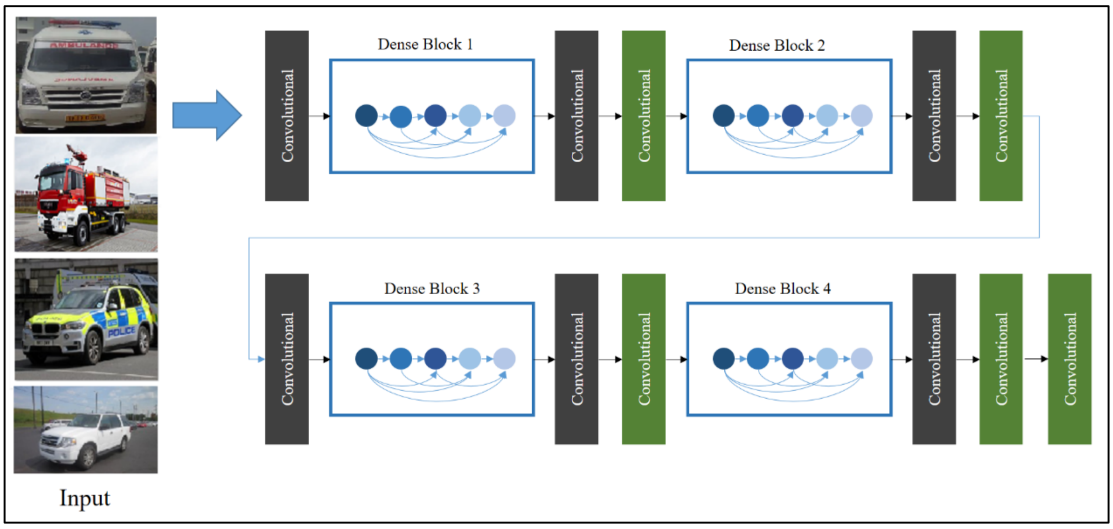


Figure 3.14 DenseNet201 with CNN for classification

3.3.2 Modified DenseNet201 (Freezing Layers)

In the field of neural networks, freezing layers refers to the control of weight updates. When a layer is frozen, its weights remain unchanged during further processing. This simple, yet effective, technique helps reduce computational costs during training without significantly affecting detection accuracy. Therefore, freezing layers is a strategic approach to speed up neural network training by gradually immobilizing hidden layers. In this study, we will apply multiple freezing layers to the DenseNet201 layers to show their impact on performance accuracy as depicted in Figure 3.15.

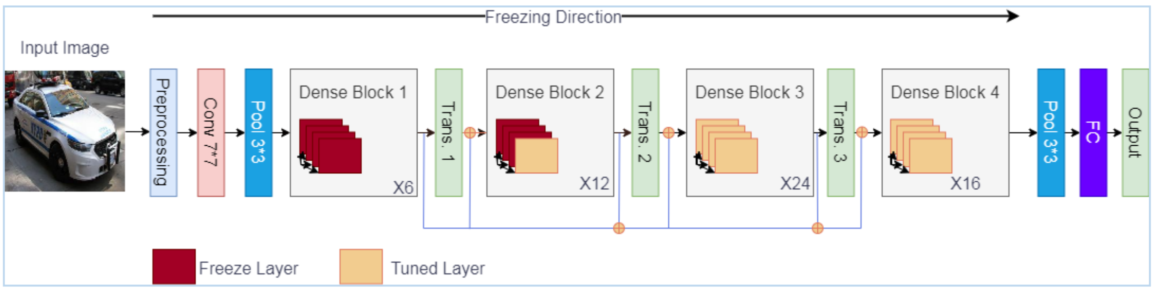


Figure 3.15 Modified DenseNet201 Architecture

3.3.3 Proposed Optimized Selection Algorithm

We train each model using input sizes of 64x64, 128x128, and 224x224. The models are then simulated with different epoch numbers. Performance metrics such as accuracy, precision, recall, and F1-score are calculated for each model. Next, we sort the results to find the optimal values of accuracy and precision. Our selection process is based on choosing the best optimizer when accuracy or precision reach their peak, as shown in Figure 3.16.

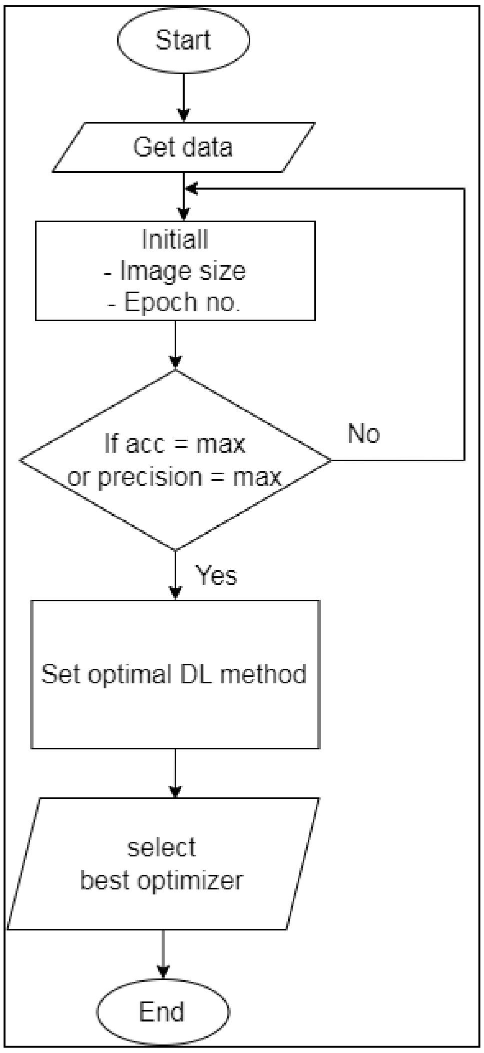


Figure 3.16 Proposed Search Mechanism

3.3.4 Selecting Optimizers

Many optimization techniques discussed in the existing literature have been widely used in recent studies on deep transfer learning. These techniques are designed to reduce the loss function and modify the weights during back-propagation. Gradient descent is a commonly used method for finding local minima of different functions. It calculates the gradient by evaluating the loss function across the entire dataset.

This study investigates how different optimizers affect the accuracy of deep learning DenseNet201 models, both with and without freezing layers. Figure 3.17 depicts the selection of the optimal optimizer.

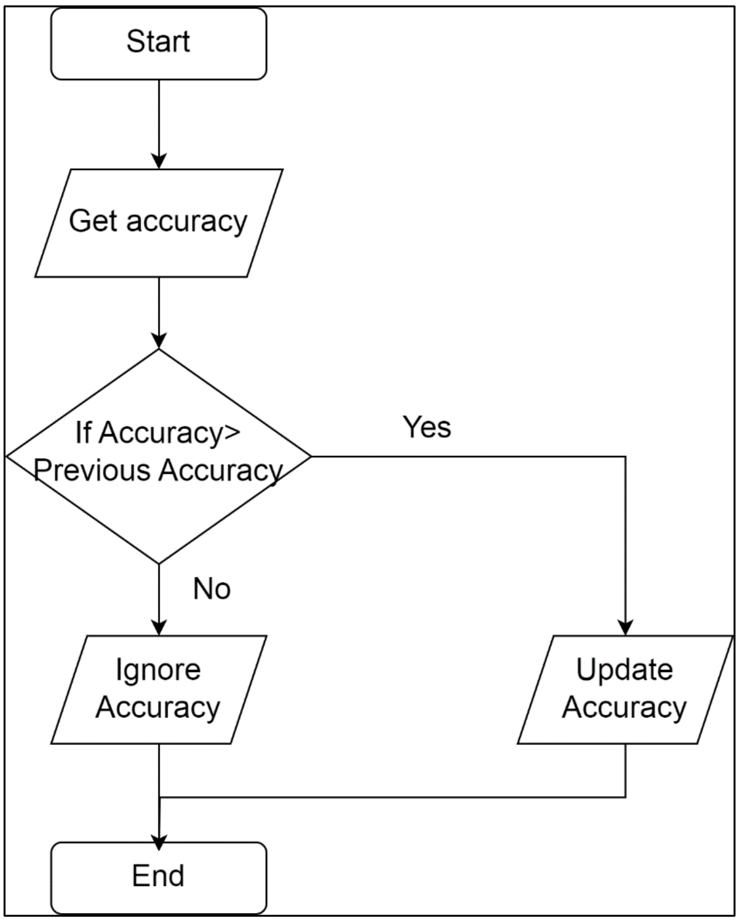


Figure 3.17 Selecting Best Optimizer

3.4 Vehicle Types Detection

Deep learning techniques can be employed for the purpose of detecting and categorizing vehicles, with a specific focus on distinguishing emergency vehicles (EVs) such as police cars, ambulances, and firefighters from non-emergency vehicles. This involves training a deep learning model to not only detect vehicles, but also identify them accurately. Figure 3.18 displays a flowchart depicting the processes used to detect vehicle types.

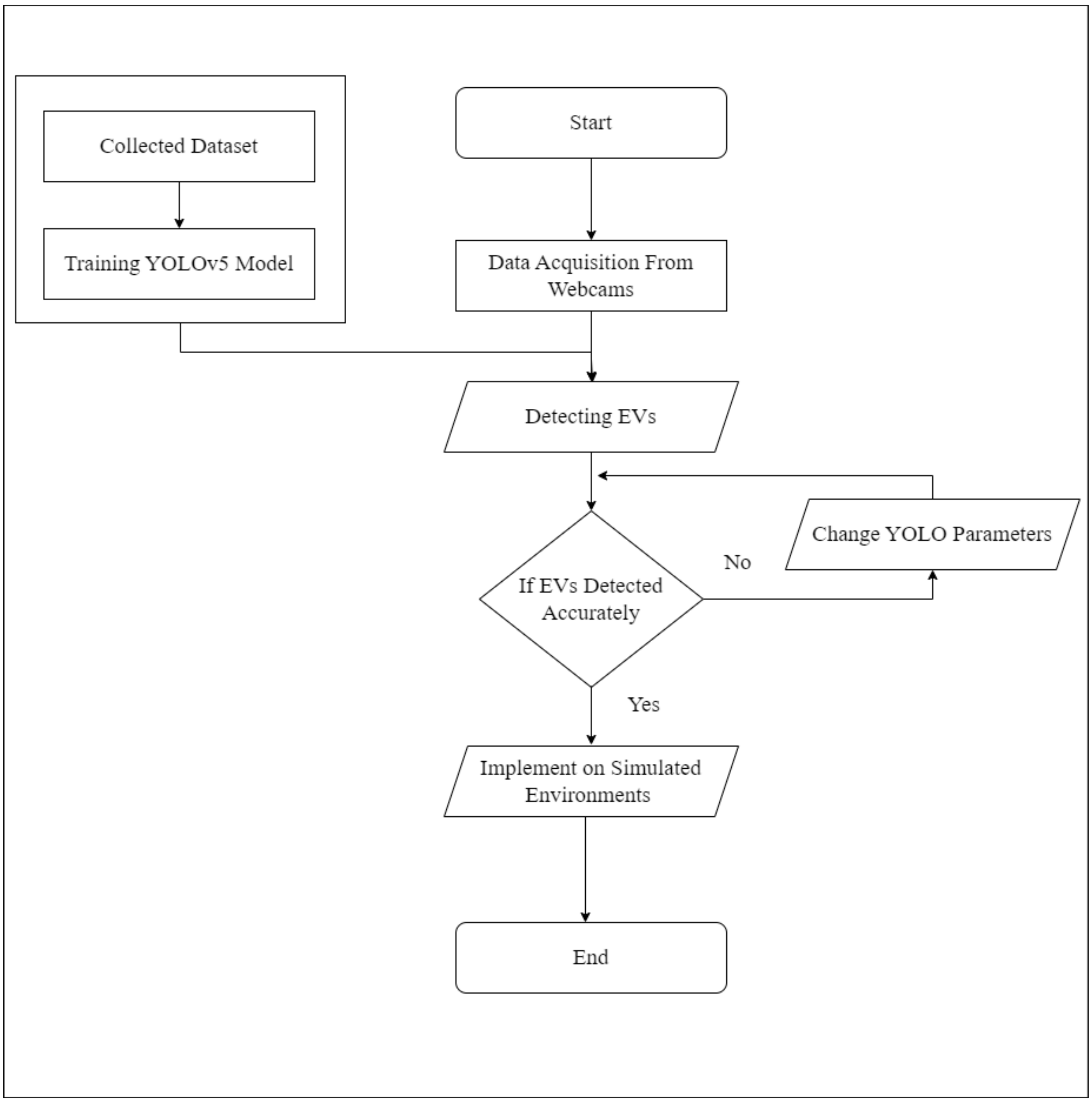

Collected Dataset
Training YOLOv5 Model
Start
Data Acquisition From Webcams
Detecting EVs
Change YOLO Parameters
No
If EVs Detected Accurately
Yes
Implement on Simulated Environments
End

Figure 3.18 Vehicle Types Detection Processes

3.4.1 Modified YOLOv5s

The changes are made to the YOLOv5s backbone are intended to improve the network's ability to learn and represent features more efficiently. In this study, the original C3 layers are replaced with BottleneckCSP layers and substituted the Spatial Pyramid Pooling - Fast (SPPF) module with the Spatial Pyramid Pooling (SPP) module in the modified backbone of the YOLOv5s architecture and named to YOLOv5sm. Modified version of YOLOv5 is illustrated in Figure 3.19.

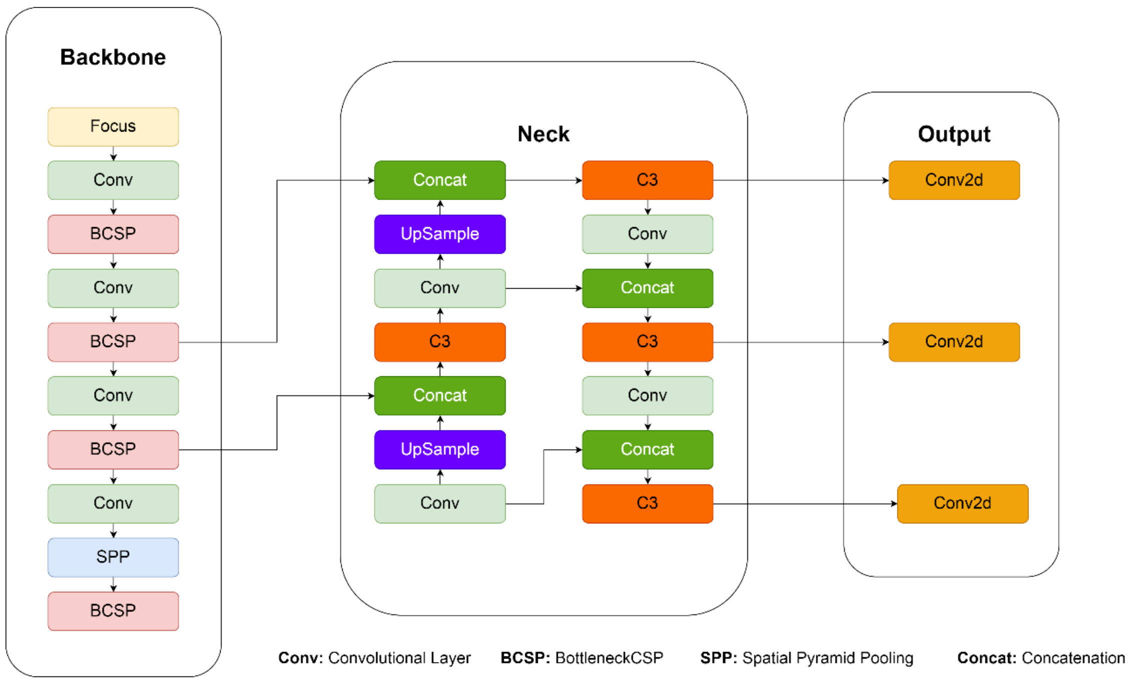


Figure 3.19 Modified YOLOv5

3.4.2 Integrate YOLO with Arduino

The architecture of the proposed system is divided into two distinct components hardware and software. In terms of hardware, it includes a webcam for video input, a PC with a GPU for training and executing the YOLO model, an LED Traffic Light Signal Module, and an Arduino Uno for signal control. As for the software component, it involves the use of the Python programming language and its associated libraries. The working mechanism of the proposed system for signalling control is illustrated in figure 3.20.

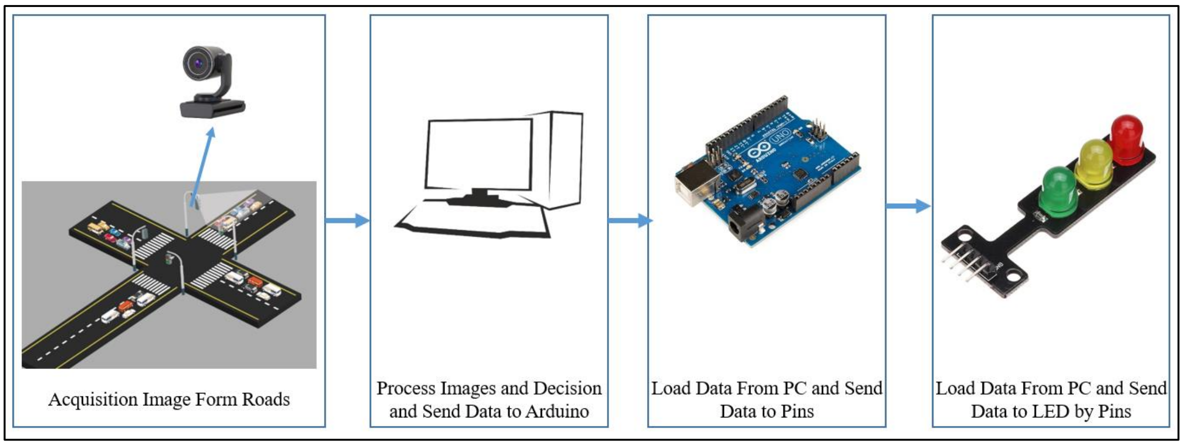

Acquisition Image Form Roads
Process Images and Decision and Send Data to Arduino
Load Data From PC and Send Data to Pins
Load Data From PC and Send Data to LED by Pins

Figure 3.20 System Architecture

3.4.3 Optimized Traffic Flow

The use of deep learning methods allows for the optimization of traffic flow and the distinction between emergency and non-emergency vehicles. By utilizing advanced neural network architectures, such as YOLO, it becomes possible to analyze traffic dynamics and classify vehicles in real-time. This allows for the accurate identification of emergency vehicles, such as police cars, ambulances and firefighters, within regular traffic, which in turn enables the implementation of dynamic traffic management strategies. These strategies prioritize the passage of emergency vehicles, thereby improving response times during critical situations. In addition, deep learning techniques can analyze traffic flow patterns and adjust signal timings at intersections to reduce congestion and improve overall traffic efficiency. By integrating deep learning into traffic management systems, smarter and more adaptable control mechanisms can be implemented, leading to safer and more efficient transportation networks. A general block diagram of using optimized traffic flow is shown in the Figure 3.21.

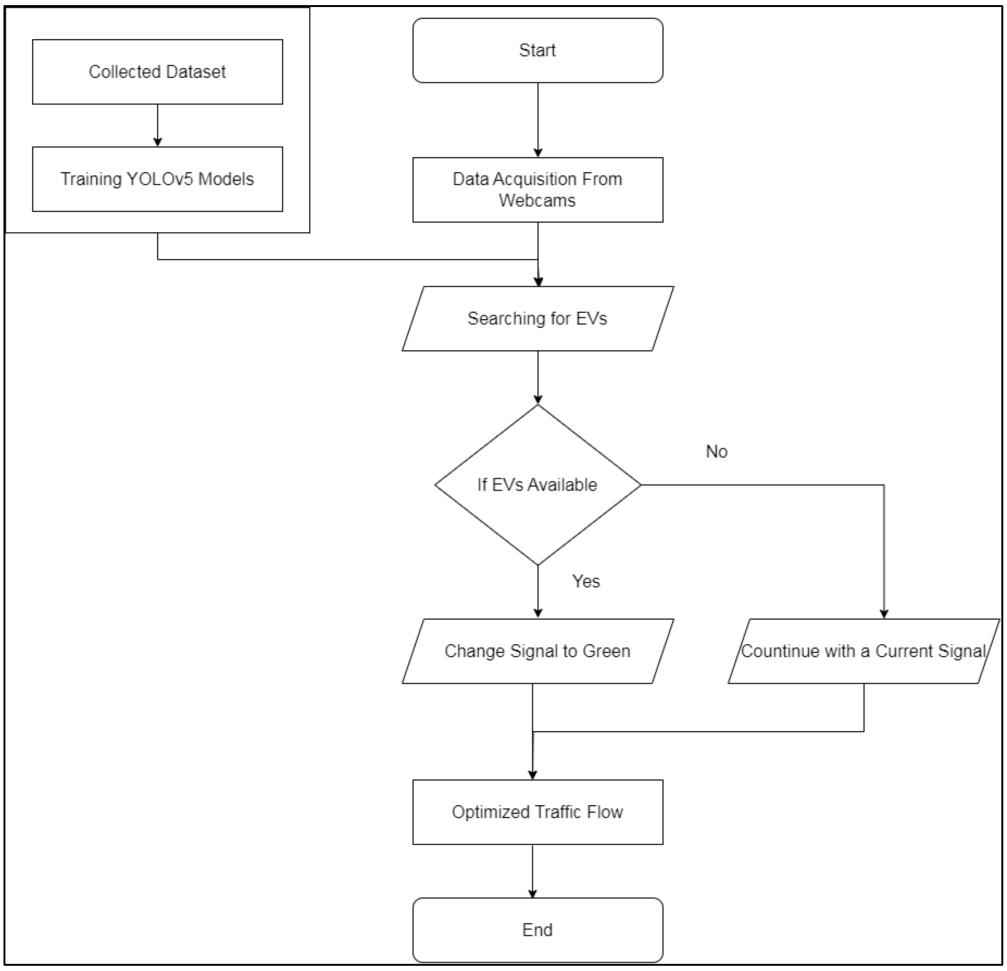


Figure 3.21 General Block Diagram of Optimized Traffic Flow

3.5 Mathematical Formulation of Optimized Traffic Flow

This study presents a novel mathematical formulation that aims to optimize the traffic flow. The proposed approach utilizes advanced algorithms and real-time traffic data analysis to minimize congestion by adjusting traffic signals in the presence of emergency vehicles.

Normally, allowed time (ATi) for one lane is equal to:

$$AT_i = G_i + Y_i \tag{3.1}$$

Where,

- *AT_i : Allowed time for passing vehicles of Lane i (in seconds)*
- *G_i : Duration of green signal for Lane i (in seconds)*
- *Y_i : Duration of yellow signal for Lane i (in seconds)*

That means the total times for one cycle is equal to:

$$C = \sum_{i=0}^{n} AT_i \qquad 3.2$$

Where

- *C: Total cycle length (in seconds)*

The expanded equation of 3.2 is equal to:

$$C = \sum_{i=0}^{n} G_i + Y_i \qquad 3.3$$

To incorporate the preemption (P_i) duration for emergency vehicles into the equation for the total cycle length *C*, we need to add the preemption duration for each lane *i* when an emergency vehicle appears.

$$C = \sum_{i=0}^{n} G_i \pm P_i + Y_i \quad \begin{cases} + P_i \; if \; EVs \; apperared \\ - P_i \; if \; road \; Cleard \; and \; no \; vehcile \; on \; the \; Lane \end{cases} \qquad 3.4$$

Then, the static signaling time is converted from static to dynamic value based on factors including appearance EVs and not cars in the lane while still in green.

$$C = \sum_{i=0}^{n} \Delta(G_i \pm P_i + Y_i) \begin{cases} + P_i \; if \; EVs \; apperared \\ - P_i \; if \; road \; Cleard \; and \; no \; vehcile \; on \; the \; Lane \end{cases} \quad 3.5$$

3.6 Performance Metrics

To evaluate the performance of the models, several metrics are used. Machine learning tasks are typically categorized into either classification or object detection. It is crucial to select appropriate metrics to evaluate performance, as not all metrics are suitable for every type of problem. The metrics used to evaluate the proposed models in this study are divided into two parts: classification performance and object detection.

The precision-recall curve offers a comprehensive assessment of a system's performance, often consolidated into a single metric by calculating the average precision across various standard recall levels or document numbers (Goutte and Gaussier, 2005).

Precision stands for a model's capability to accurately identify pertinent objects, denoting the percentage of correct positive predictions (Kamal and Hamid, 2023).

$$Precision = \frac{TP}{TP+FP} \qquad 3.6$$

Sensitivity (Recall) represents a model's ability to detect all relevant instances, signifying the percentage of accurate positive predictions among all provided ground truths (Kamal and Hamid, 2023).

$$Recall = \frac{TP}{TP+FN} \qquad 3.7$$

Accuracy is defined as the proportion of correctly predicted observations in a dataset compared to the total number of observations (Kamal and Hamid, 2023).

$$Accuracy = \frac{number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \qquad 3.8$$

Or

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad 3.9$$

F1 Score is a metric that combines precision and recall, providing a balanced evaluation by considering both metrics (Kamal and Hamid, 2023).

$$F1\ score = \frac{2.TP}{2TP+FP+FN} \qquad 3.10$$

Specificity which refers to the rate of true negatives, is a measure used to assess the accuracy of correctly identifying negative data (Kamal and Hamid, 2023).

$$Specificity = \frac{TN}{TP+FP} \qquad 3.11$$

The ***Confusion Matrix*** is a structured tabular representation displaying the results of predictions in binary classification. It offers a comprehensive insight into a classification model's performance when evaluated against a dataset with known true values (Padilla et al., 2020, Gong, 2021). This metric helps to assess and delineate the accuracy and efficiency of classification predictions. Table 3.5 demonstrates a typical confusion matrix for binary classification, yet it can be expanded to accommodate classification involving more than two classes.

Table 3.5 Standard tabular confusion matrix (Narkhede, 2018)

		Actual	
		Positive (**P**)	Negative (**N**)
Predicted	Positive	True Positive (**TP**)	False Positive (**FP**)
	Negative	False Negative (**FN**)	True Negative (**TN**)

In the displayed table, columns represent the predicted values, while rows indicate the actual values. The table distinguishes between two potential classes: Positive and Negative. For example, in predicting the presence of an emergency vehicle in an image, a positive prediction column suggests the image containing a change signal to green, while a negative prediction column implies the no changing signals. The table is divided into four categories:

- True Positive (TP): Signifies a correct prediction aligned with reality.
- True Negative (TN): Indicates an accurate negative prediction.
- False Positive (FP): Represents an incorrect positive prediction.
- False Negative (FN): Denotes an incorrect negative prediction.

Average Precision (AP), in object detection challenges and across the scientific community, the AP (Average Precision) stands out as the primary metric used to assess the accuracy of detections among diverse annotated datasets. It is important to note that in object detection, a true negative (TN) result is irrelevant. This is due to the fact that there are numerous bounding boxes in any given image that should not be detected (Padilla et al., 2020).

The provided definitions necessitate defining "correct detection" and "incorrect detection." A prevalent method to achieve this is by utilizing the intersection over union (IOU), which is a metric derived from the Jaccard Index—a measure of similarity between two data sets (Jaccard, 1901).

In object detection, the IOU measures the overlap between the predicted bounding box Bp and the ground-truth bounding box Bgt, defined as the ratio of the area of their intersection to the area of their union

$$J(B_p, B_{gt}) = IOU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \qquad 3.12$$

as illustrated in Figure 3.22.

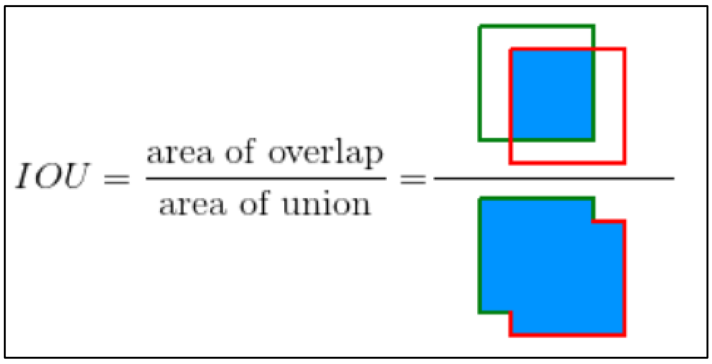

$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$

Figure 3.22 Intersection Over Union (IOU).

To determine whether a detection is correct or incorrect, we compare the IOU with a given threshold, t. If the IOU is greater than or equal to t, the detection is considered correct. If the IOU is less than t, the detection is considered incorrect.

As mentioned earlier, object detection frameworks do not utilize true negatives (TN). Therefore, metrics such as true positive rate (TPR), false positive rate (FPR), and ROC curves (Hanley and McNeil, 1982) are not used. Instead, the evaluation of object detection methods primarily revolves around precision (P) and recall (R) (as previous mentioned). Precision is defined as the proportion of correctly detected objects, while recall is the proportion of actual objects that are correctly detected.

Precision measures a model's ability to correctly identify relevant objects, expressed as the percentage of accurate positive predictions. Recall, on the other hand, gauges a model's ability to detect all relevant instances, including all ground-truth bounding boxes, and is calculated as the percentage of correct positive predictions relative to all ground truths provided.

The mean average precision (mAP) is a metric utilized to assess the accuracy of object detectors across all classes in a given database. The mAP is essentially the average precision calculated for each class (Ren et al., 2015, Liu et al., 2016), that is

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \qquad 3.13$$

with *APi* representing the average precision in the i-th class, and N being the total number of classes evaluated.

CHAPTER FOUR:

4. IMPLEMENTATION, RESULTS AND DISCUSSION

4.1 Introduction

This chapter focuses on using advanced deep learning techniques to achieve vehicle classification and detection. The main goal is to achieve high accuracy and provide valuable insights to readers. In addition, simulation environments of real traffic intersections are developed, which are crucial for the application of these techniques in real situations. Additionally, this chapter provides an overview of the architecture of implemented and modified deep learning techniques. It contains a detailed description of the methods used and changes made. Finally, the results of this study are presented and discussed.

4.2 Deep Learning-Based Models' Results

The problem of vehicles passing through a road junction was considered, and the proposed model, which comprises three main elements, was introduced. First, a robust deep learning-based classification method was developed. More precisely, the deep learning model was improved using the efficiency of DenseNet201, which achieved greater accuracy than other models. The purpose is to enhance vehicle classification during crossing at a junction. In the second part, YOLOv5 was employed for vehicle identification and detection tasks. With this approach, it is possible to accurately and efficiently determine the vehicle type at any given moment when they pass through the intersection, achieving complete traffic analysis of the area. Finally, the system features an innovative algorithm for efficient signal optimization that aims to minimize average travel time by dynamically regulating traffic signals to match observed congestion levels in order to maximize flow rate and prevent queue build-up.

4.2.1 Vehicle Types Classification Results

As explained in previous sections, one of the main goals of this work is to find high accuracy DL technique and an optimal optimizer. Accordingly, the pre-trained models are compiled using the customized dataset for different epoch numbers (i.e., iterations). Based on the results other hyper-parameters are fixed for all methods, such as (filter size, dropout rate, batch size, and learning rate). The description of value of the fixed hyper parameters are shown in the Table 4.1

Table 4.1 Fixed Hyper-parameters for DL Techniques

Hyper Parameter	Description
Filter Size	7*7
Dropout Rate	0.5
Batch Size	16
Learning Rate	0.0001

The tests conducted involved evaluating performance metrics such as average accuracy, precision, recall, and F1-Score. These metrics were measured over a span of 15 to 30 epochs, a range chosen based on observations from preliminary experiments. When the model was trained for fewer than 15 epochs, its performance consistently suffered. Conversely, extending training beyond 30 epochs did not lead to noticeable improvement in performance metrics. Figure 4.1. and 4.2. show the accuracy and precision results for the different input image sizes when the best optimizer has been chosen. In this case, a modified DenseNet201 with 120 layers freezing is used.

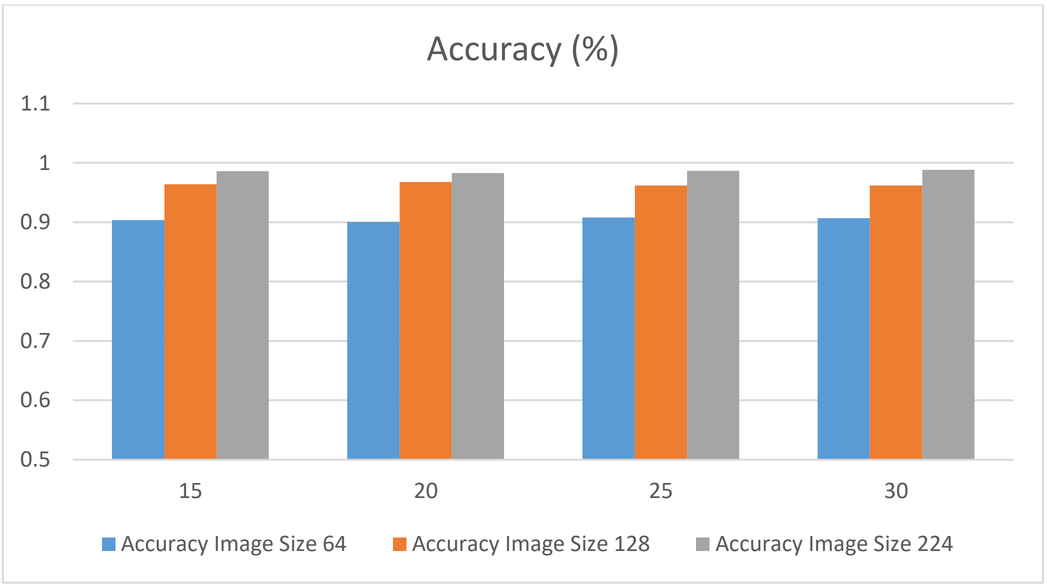


Figure 4.1 Accuracy versus image size (DensNet201-120_freeze_layers)

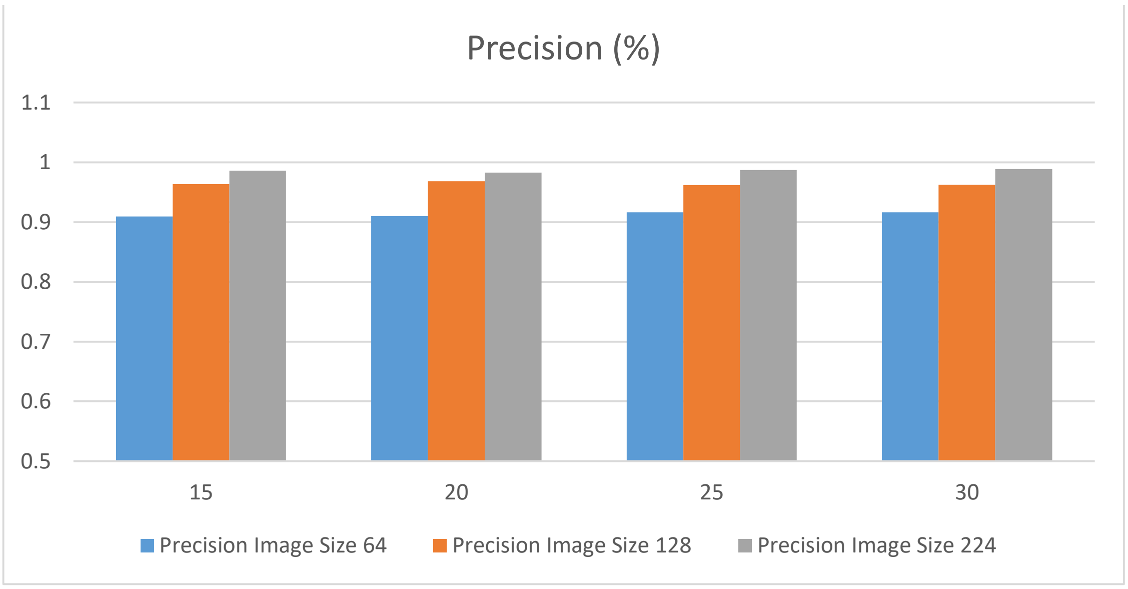


Figure 4.2 Precision versus image size (DensNet201-120_freeze_layers)

Experimental results in Figures 4 and 5 reveal that the image size 224*224 gives better accuracy and precision values. Therefore, for further tests, merely used this size with all techniques and optimizers in this research.

Table 4.2 and 4.3 present the accuracy and precision metrics results to select the best optimizer using our new search strategy and other tests are detailed in the Appendix A.

Table 4.2 Accuracy Tests

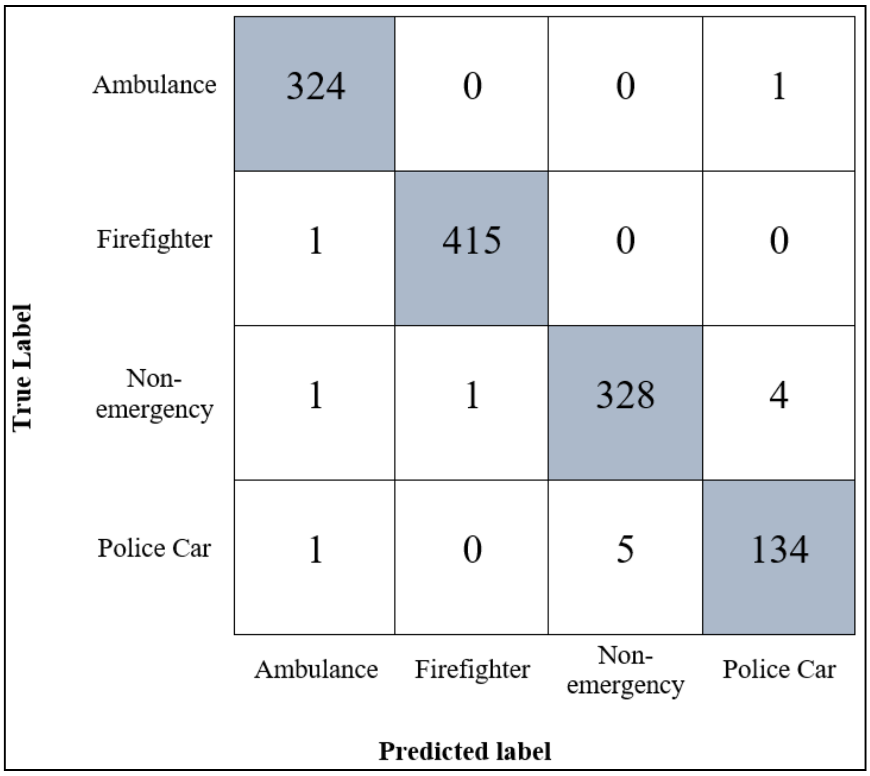
Image Size	Epoch No.	Models	Optimizers	Accuracy (%)
64	15	VGG19	Adam	90.37
	20	VGG19	Adam	90.04
	25	VGG16	RMSprop	90.78
	30	VGG19	Nadam	90.69
128	15	DenseNet201 Freeze 0	Nadam	96.37
	20	DenseNet201 Freeze 0	RMSprop	96.79
	25	DenseNet201 Freeze 0 - 30	Adam	96.21
	30	DenseNet201 Freeze 0 - 90	Adam	96.21
224	15	DenseNet201 Freeze 0 - 30	Adam	98.06
	20	DenseNet201 Freeze 0 - 60	RMSprop	98.27
	25	DenseNet201 Freeze 0 - 150	RMSprop	98.68
	30	**DenseNet201 Freeze 0 - 120**	**RMSprop**	**98.84**

The highlighted rows in the above tables indicate that the more accurate model is the DenseNet201 with 120 layers freezing and the best optimizer is RMSprop for both accuracy and precision tests when the input image size is 224*224 and the epoch No. is 30.

Table 4.3 Precision Tests

Image Size	Epoch No.	Models	Optimizers	Accuracy (%)
64	15	VGG16	Adam	90.96
	20	VGG19	Adam	91.01
	25	VGG16	RMSprop	91.63
	30	VGG19	Nadam	91.65
128	15	DenseNet201 Freeze 0	Nadam	96.37
	20	DenseNet201 Freeze 0	RMSprop	96.84
	25	DenseNet201 Freeze 0 - 30	Adam	96.21
	30	DenseNet201 Freeze 0 - 90	Adam	96.26
224	15	DenseNet201 Freeze 0 - 30	Adam	98.61
	20	DenseNet201 Freeze 0 - 60	RMSprop	98.27
	25	DenseNet201 Freeze 0 - 150	RMSprop	98.71
	30	**DenseNet201 Freeze 0 - 120**	**RMSprop**	**98.85**

In addition, the confusion matrix of the proposed emergency vehicles classification transfer-based model is plotted, which is displayed in Figure 4.3. The matrix diagonal represents the proposed model performance accuracy for different types of emergency cars.



True Label \ Predicted label	Ambulance	Firefighter	Non-emergency	Police Car
Ambulance	324	0	0	1
Firefighter	1	415	0	0
Non-emergency	1	1	328	4
Police Car	1	0	5	134

Figure 4.3 Confusion matrix for the optimal model and optimizer

The model's (DenseNet201 with 120 layers freezing) loss and accuracy for the training and test validation are depicted in Figure 4.4 and Figure 4.5 respectively.

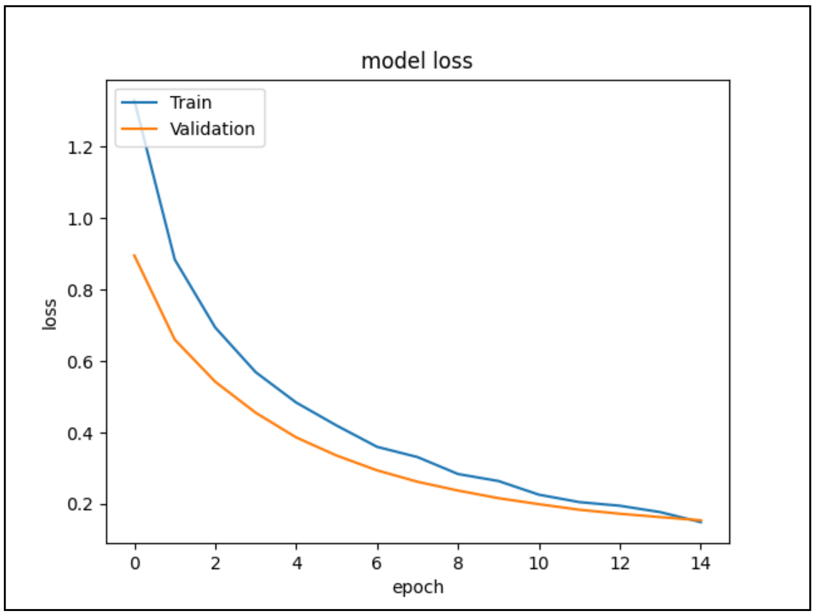


Figure 4.4 Model's loss (DensNet201 – 120 layers freezing)

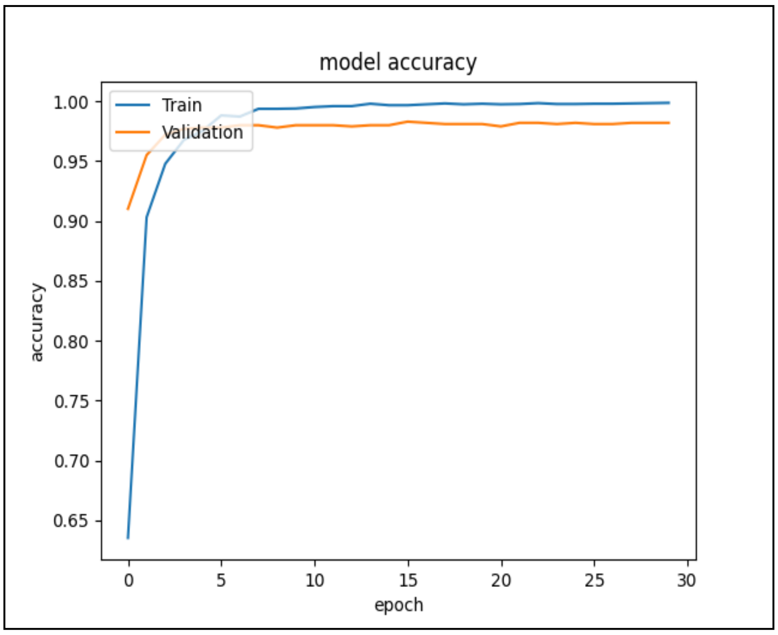


Figure 4.5 Model's accuracy (DensNet201 – 120 layers freezing)

4.2.2 Vehicle Types Detection Results

Another objective of this study is to apply the YOLO deep learning technique for detecting vehicle types. The hyperparameters, such as filter size, dropout rate, batch size, and learning rate, are kept constant across all methods using the YOLO model, except image size. The Table 4.4 displays the Description and Value of the Fixed Hyperparameters.

Table 4.4 Fixed Hyper-parameters for YOLOv5 Techniques

Hyper-parameter	Description
Batch Size	16
Learning Rate	0.01
Optimizer	SGD

Figures 4.6 and 4.7 show examples for batch training and prediction, respectively.

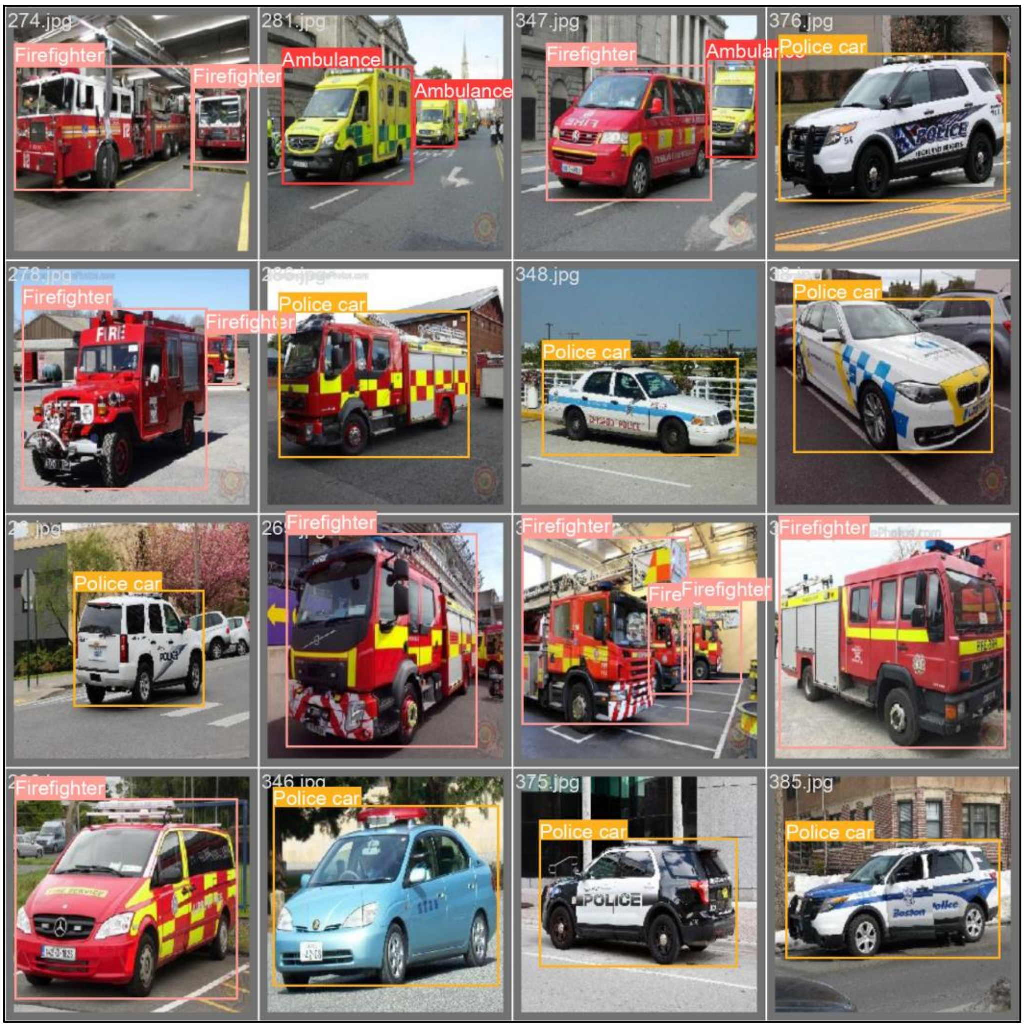


Figure 4.6 Batch Training

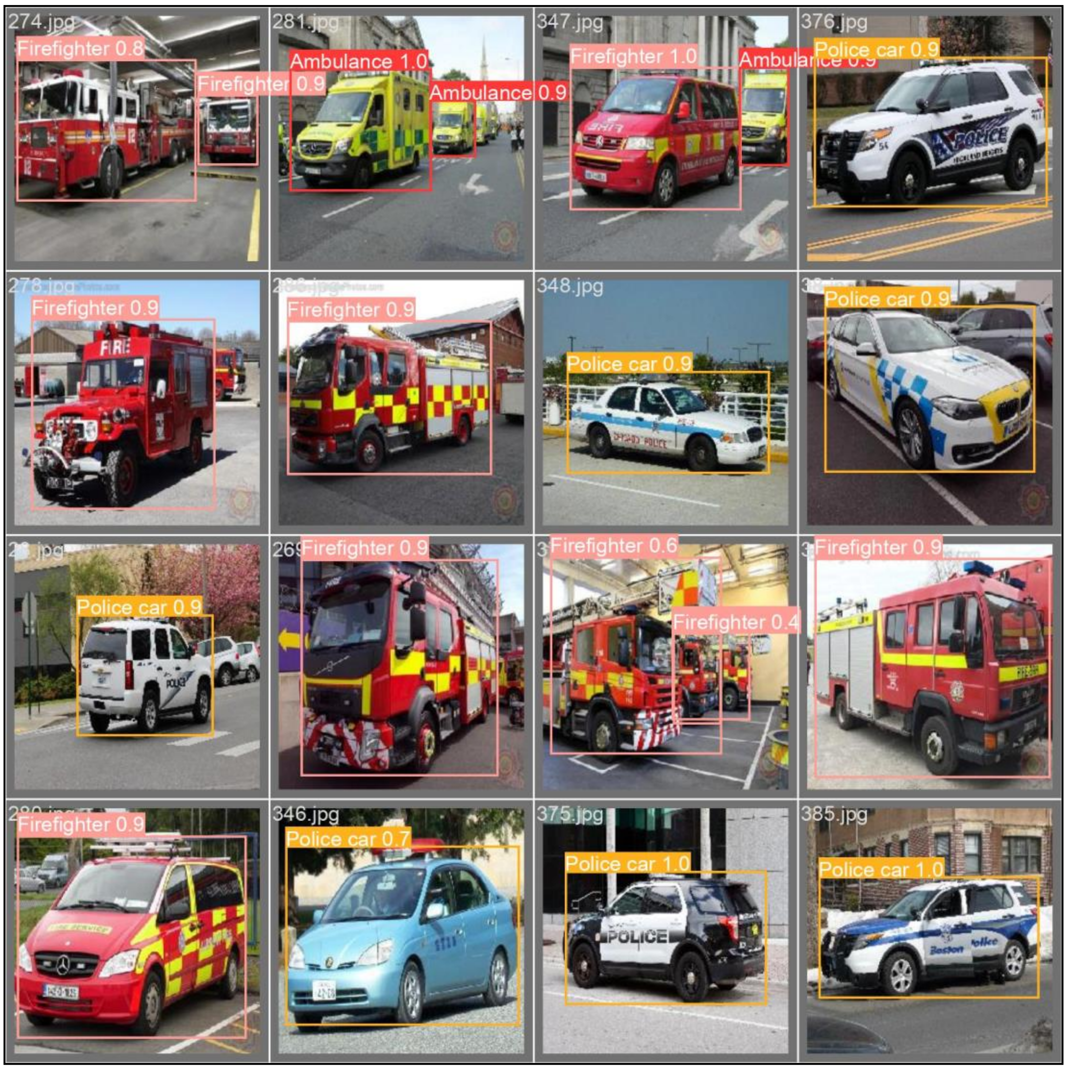


Figure 4.7 Batch Prediction

The results are obtained from evaluating different input image sizes using performance metrics such as mAP, recall, and F1-Score are presented in Table 4.5. The highlighted rows in the tables indicate the superior performance of the YOLOv5sm model.

Table 4.5 Object Detection Using YOLOv5 and Modified YOLOv5 Results

Model	Image Size	No. of Epochs	F1 Score (%)	Precision (%)	Recall (%)	mAP@0.5 (%)
YOLOv5s	640	50	86	100	99	91
		75	87	100	99	90.2
		100	88	100	98	92.2
	512	100	89	100	99	91.7
YOLOv5sm	640	100	0.87	100	98	90.9
(Proposed)	**416**	**100**	**89**	**100**	**99**	**93.2**

The tests conducted evaluate performance metrics, including average F1-Score, precision, recall, mAP, and confusion matrix are shown in Figures 4.9, 4.10, 4.11, and 4.13 Respectively.

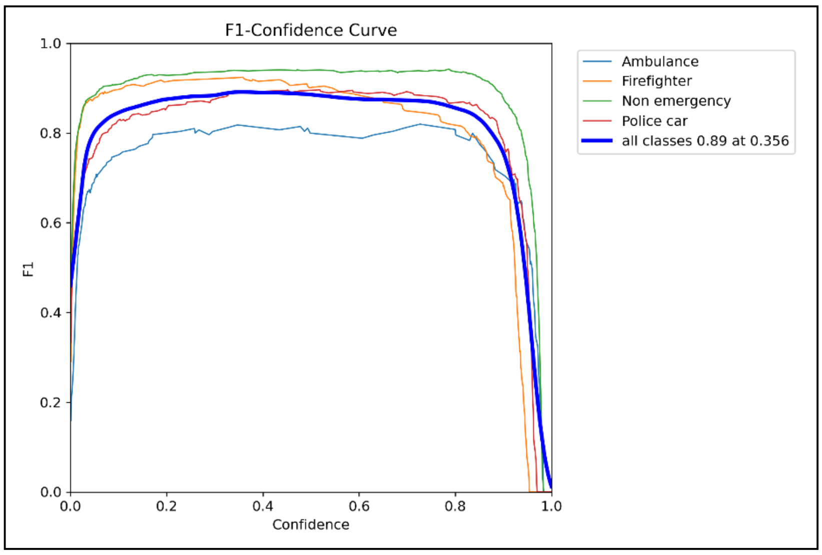


Figure 4.8 F1 Score

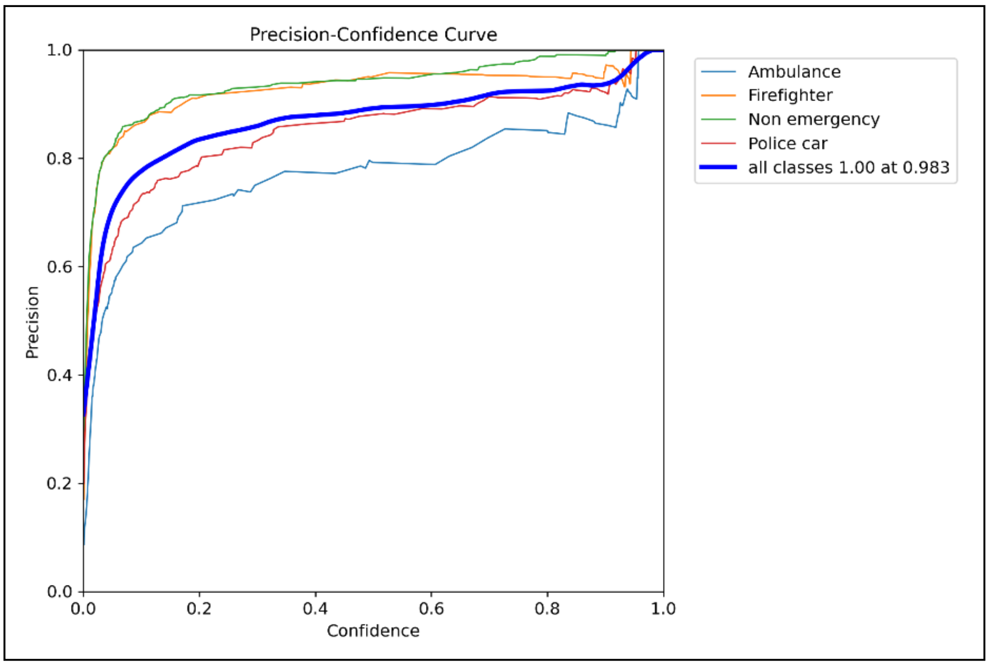


Figure 4.9 Precision

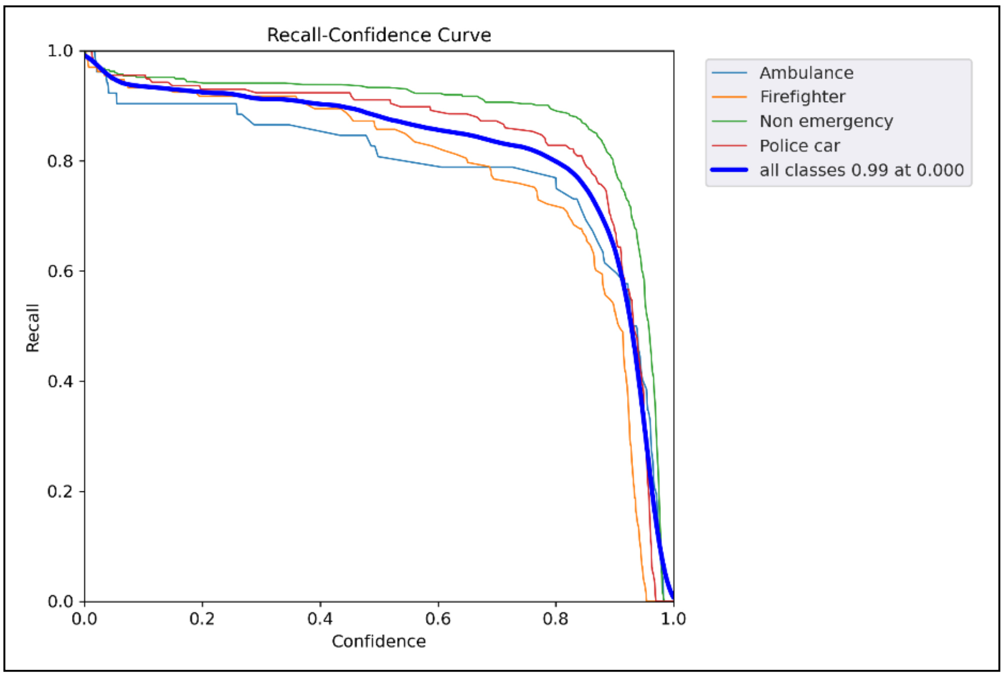


Figure 4.10 Recall

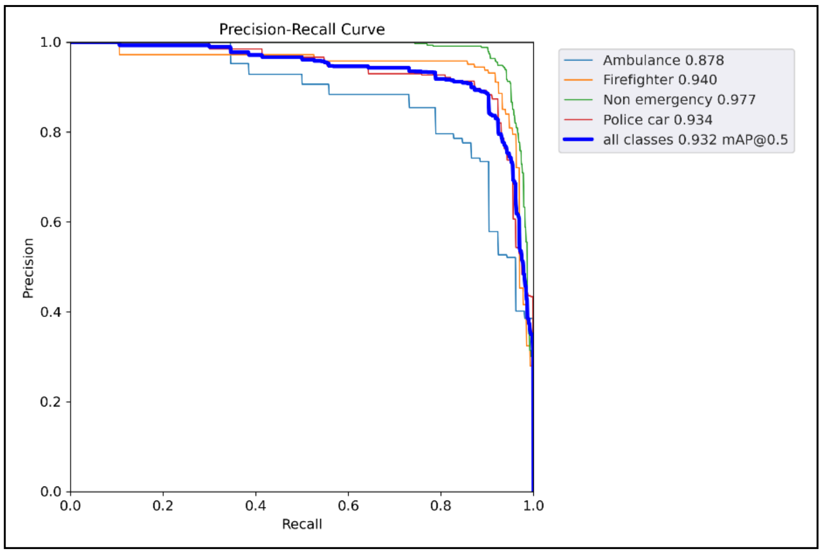


Figure 4.11 mAP@0.5

Confusion Matrix

	Ambulance	Firefighter	Non-Emergency	Police Car
Ambulance	81	2	0	4
Firefighter	6	91	0	2
Non-Emergency	0	2	93	2
Police Car	13	3	2	92

Figure 4.12 Confusion Matrix

Figures 4.14 and 4.15 show the loss functions for the training and validation sets. These figures depict both the unmodified (YOLOv5s) and modified (YOLOv5sm) versions, utilizing the same hyperparameters.

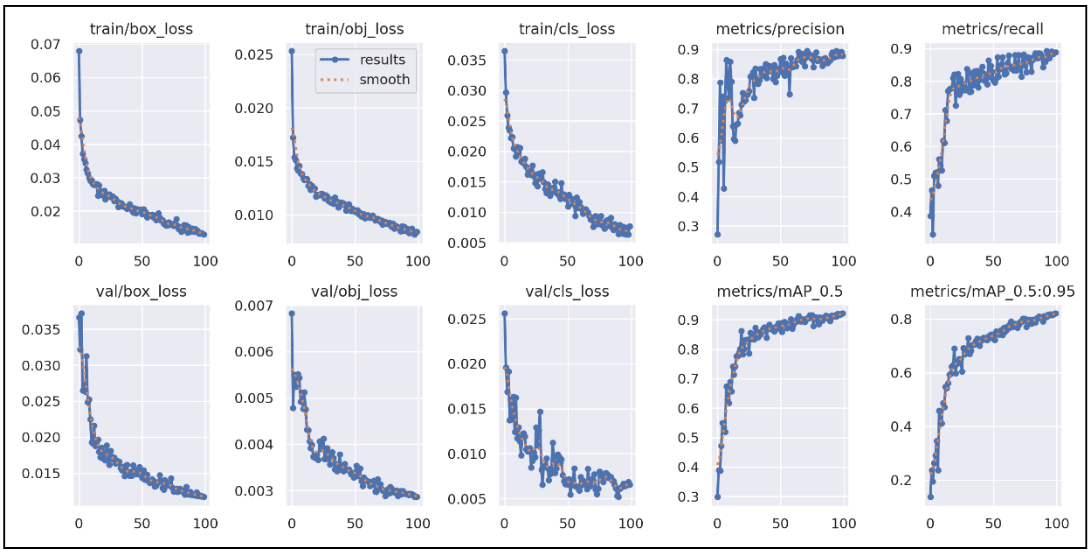


Figure 4.13 The loss functions for the training and validation sets of original (YOLOv5s)

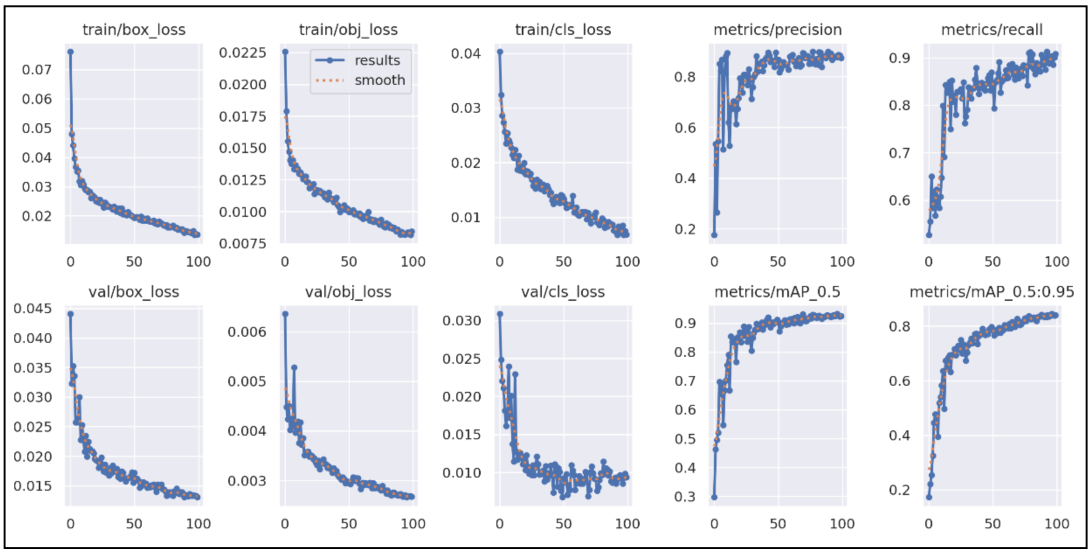


Figure 4.14 The loss functions for the training and validation sets for modified (YOLOv5sm)

4.2.3 Simulated Environment Results

Due to difficulties implementing smart traffic system on the roads and obtaining permission from the relevant security offices, a simulated version of a real traffic environment was developed for this study. As previously mentioned, the equipment in the simulated environment includes a webcam for acquiring images from simulated roads, a PC for processing the captured images and uploading them to the model, and an Arduino with LEDs for controlling traffic signals as shown in the Figure 4.16.

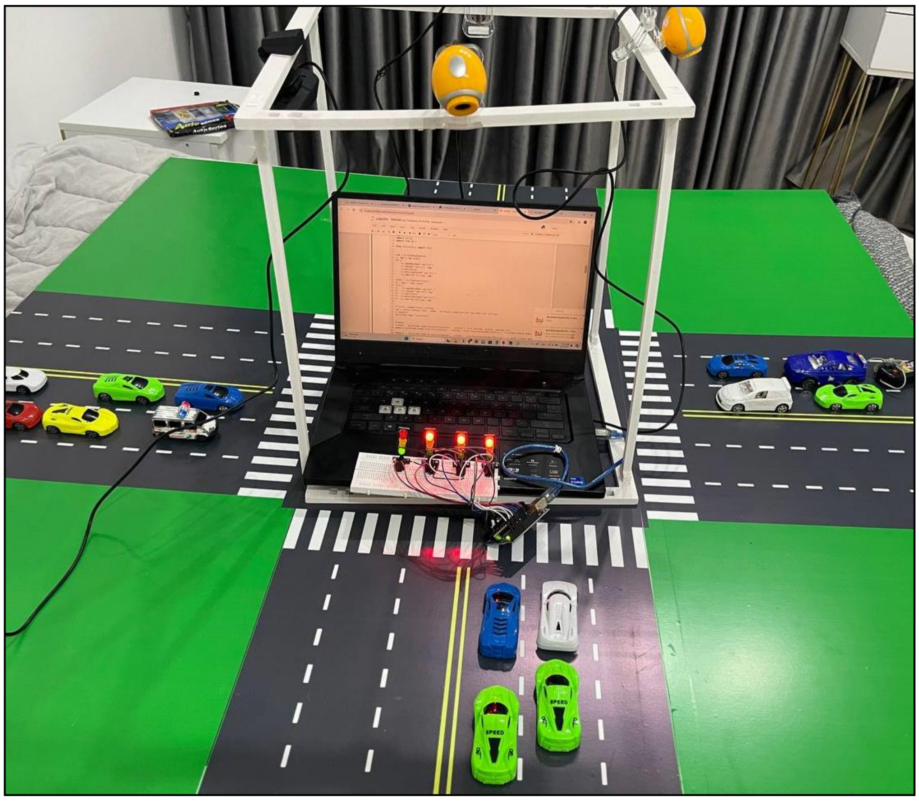


Figure 4.15 Simulated Environment of Traffic Intersection

A semi-truth table has been used to control and optimize traffic flow. This table represents the values of A (Ambulance), F (Firefighter), P (Police Car), and C (Crowded Lane) in relation to Lanes (L1, L2, L3, and L4), with the goal of determining availability.

The factors, which can affect the traffic signals with respecting lanes have been coded and read by Arduino as shown in the Table 4.5.

Table 4.6 Lanes and Factors

	Traffic Lanes				
Factors		L1	L2	L3	L4
	A	1	5	a	A
	F	2	6	b	B
	P	3	7	c	C
	C	4	8	d	D

For testing the proposed traffic flow some scenario created which are detailed below:

- **Case 1:** Assume that:
  - Set the duration of one cycle = 140 Seconds (120 for Green Signals with 30 Seconds for each Lane and 20 Seconds for Red with 5 Seconds for each Lane).
  - The green signal turned on for Lane 3, indicating that it can continue and try to finish within its allocated time.
  - An ambulance appeared in Lane 2, which is currently under the red signal as shown in Table 4.7.
  - The proposed method involves changing the signal of L3 to

yellow for 5 seconds, which will enable the ambulance to pass through the traffic area by opening L2.

- After the yellow signal has finished, the green signal returns to L3 and traffic cycles continue until another factor appears.

Table 4.7 Case 1

		Traffic Lanes			
Factors		L1	L2	L3	L4
	A	🟥 X	🟥 ✓	🟩 X	🟥 X
	F	🟥 X	🟥 X	🟩 X	🟥 X
	P	🟥 X	🟥 X	🟩 X	🟥 X
	C	🟥 X	🟥 X	🟩 X	🟥 X

Where:

✓ represents that the factor occurs on the level.

X represents that the factor does not occur on the level.

- **Case 2:** Assume that:
  - Set the duration of one cycle = 140 Seconds (120 for Green Signals with 30 Seconds for each Lane and 20 Seconds for Red with 5 Seconds for each Lane).

- The green signal turned on for Lane 3, indicating that it can continue and try to finish within its allocated time.
- A Firefighter appeared in Lane 1, which is currently under the red signal as shown in Table 4.8.
- The proposed method involves changing the signal of L3 to yellow for 5 seconds, which will enable the Firefighter to pass through the traffic area by opening L1.
- After the yellow signal has finished, the green signal returns to L3 and traffic cycles continue until another factor appears.

Table 4.8 Case 2

		Traffic Lanes			
		L1	L2	L3	L4
Factors	A	🟥 X	🟥 X	🟩 X	🟥 X
	F	🟥 ✓	🟥 X	🟩 X	🟥 X
	P	🟥 X	🟥 X	🟩 X	🟥 X
	C	🟥 X	🟥 X	🟩 X	🟥 X

Where:

✓ *represents that the factor occurs on the level.*

X represents that the factor does not occur on the level.

- **Case 3:** Assume that:
    - Set the duration of one cycle = 140 Seconds (120 for Green Signals with 30 Seconds for each Lane and 20 Seconds for Red with 5 Seconds for each Lane).
    - The green signal turned on for Lane 4, indicating that it can continue and try to finish within its allocated time.
    - A Police Car appeared in Lane 1, which is currently under the red signal as shown in Table 4.9.
    - The proposed method involves changing the signal of L4 to yellow for 5 seconds, which will enable the Firefighter to pass through the traffic area by opening L1.
    - After the yellow signal has finished, the green signal returns to L4 and traffic cycles continue until another factor appears.

Table 4.9 Case 3

		Traffic Lanes			
		L1	L2	L3	L4
Factors	A	🟥 X	🟥 X	🟥 X	🟩 X
	F	🟥 X	🟥 X	🟥 X	🟩 X
	P	🟥 ✓	🟥 X	🟥 X	🟩 X
	C	🟥 X	🟥 X	🟥 X	🟩 X

Where:

✓represents that the factor occurs on the level.

X represents that the factor does not occur on the level.

The results for cases are shown in the Table 4.10.

Table 4.10 Optimized Traffic Flow Results

Case	Normal Time to Pass (in Seconds)	Optimized Time (in Seconds)	Reduced Time (in Seconds)
1	105	5	100
2	70	5	65
3	35	5	30

4.3 Discussion

The analysis of the obtained results focuses on three key areas: classification processes, vehicle detection processes, and traffic flow optimization. Each process demonstrates significant improvements in accuracy due to the implementation of smart traffic signaling.

4.3.1 Discussion of Vehicle Types Classification Results

The performance of the DenseNet201 model can be greatly affected by the number of frozen layers during training and the quality of the training data. This impact is evident when comparing two configurations of DenseNet201, as shown in Table 4.2, which achieved higher accuracy compared to others. The results highlight the notable effect of augmented data on increasing accuracy.

In the first configuration, the DenseNet201 model was trained by freezing the first 30 layers using an imbalanced dataset of 3218 images. Despite the imbalance, the model achieved an impressive accuracy of 96.7%.

In contrast, the second configuration which involved training the model by freezing the first 120 layers and using a balanced dataset of 6222 images. This setup resulted in a significantly higher accuracy of 98.84%, showcasing the combined benefits of freezing more layers and using balanced data. A balanced dataset ensures equal representation of each class, which generally improves model generalization. It is worth noting that the same RMSProb optimizer was used in both scenarios. The fixed Hyper-parameters for DL techniques are shown in Table 4.11.

Table 4.11 Fixed Hyper-parameters for DL Techniques

Model	Optimizer		Accuracy (%)
DenseNet201 with Freezing 0-30 Layers	RMSProb	Imbalanced data (3218 images)	96.7
DenseNet201 with Freezing 0-120 Layers	RMSProb	Balanced data (6222 images)	98.84

The significant improvement in accuracy, from 96.7% to 98.84%, can be attributed to two key factors. First, freezing a larger number of layers, specifically 120 instead of just 30, allows the model to retain more of its pre-trained features. This aids in better feature extraction and reduces the risk of overfitting. Second, using a balanced dataset with data augmentation techniques likely contributed to the improvement. Data augmentation enhances the diversity and representativeness of the training data, allowing the model to learn more generalized features through varied training examples.

In summary, the results clearly demonstrate the crucial roles played by layer freezing and the quality of the training data in the performance of the DenseNet201 model. Freezing more layers (0-120) and using a balanced dataset significantly increase accuracy, as evidenced by the rise to 98.84%. This highlights the effectiveness of these strategies in improving model performance and emphasizes the importance of data augmentation and appropriate model adjustments in achieving high accuracy in machine learning tasks.

4.3.2 Discussion of Vehicle Detection Results

The performance of the YOLOv5s model and its proposed modified YOLOv5sm, has been analyzed across different image sizes and training epochs using key metrics such as F1 Score, Precision, Recall, and mAP@0.5.

The YOLOv5s model, tested with an image size of 640*640, shows a gradual improvement in F1 Score from 86% at 50 epochs to 88% at 100 epochs. Precision remains consistently perfect at 100% across all epochs, indicating no false positives. Recall slightly decreases from 99% to 98% as epochs increase, while mAP@0.5 improves from 91% at 50 epochs to 92.2% at 100 epochs. This technique demonstrates that longer training leads to better overall performance accuracy.

When the image size is reduced to 512*512 and the model is trained for 100 epochs, the F1 Score reaches 89%, Precision remains at 100%, Recall at 99%, and mAP@0.5 at 91.7%. This configuration suggests that a smaller image size with sufficient training can yield high performance, but it does not surpass the performance metrics of the proposed model.

The proposed YOLOv5sm model, tested with an image size of 640* 640 for 100 epochs, achieves an F1 Score of 87%, Precision of 100%, Recall of 98%, and mAP@0.5 of 90.9%. However, with an image size of 416*416 at 100 epochs, it significantly outperforms the other configurations with an F1 Score of 89%, Precision of 100%, Recall of 99%, and mAP@0.5 of 93.2%. This demonstrates that the proposed model's modifications are particularly effective at this smaller image size, achieving the highest mAP@0.5 among all tested configurations.

In summary, both models exhibit high precision across all configurations, but the proposed YOLOv5sm with an image size of 416*at 100 epochs delivers the best balance of metrics, particularly excelling in mAP@0.5. This indicates its superior capability in object detection tasks, making it a preferred choice for applications requiring high accuracy and reliability. Further validation on diverse datasets and real-world conditions would ensure the robustness of these findings.

4.3.3 Discussion of Optimized Traffic Flow

The analysis presents three scenarios that compare the time it takes to complete a process before and after optimization. These scenarios provide insights into the efficiency gains achieved.

In the first scenario, the initial time to complete the process is 105 seconds. After optimization, this time is drastically reduced to 5 seconds, resulting in a reduction of 100 seconds. This indicates an approximately 95.24% improvement in efficiency, showcasing the significant impact of the optimization technique.

The second scenario shows a similar trend. Initially, the process takes 70 seconds. After optimization, the time drops to 5 seconds, achieving a 65-second reduction, which translates to a 92.86% increase in efficiency. This further underscores the effectiveness of the optimization method employed.

In the third scenario, the process originally takes 35 seconds to complete. With optimization, the time is reduced to 5 seconds, resulting in a 30-second decrease and an 85.71% improvement in efficiency. This again highlights the substantial benefits of the optimization process.

The obtained results clearly illustrate the significant impact of optimization on reducing process times across all scenarios. The optimized time consistently drops to 5 seconds in each case, suggesting a standardized optimization approach.

The value of improvement is noteworthy, with time reductions ranging from 85.71% to 95.24%. This consistency in achieving a 5-second completion time indicates the effectiveness of the optimization process and its potential for broad application.

In summary, the analysis demonstrates the profound effect of optimization on reducing process time with significant efficiency gains observed in each scenario.

CHAPTER FIVE:

5. CONCLUSIONS AND FUTURE WORKS

5.1 Conclusions

In recent years, the issue of traffic congestion has become increasingly critical due to the growing number of vehicles on the roads. This study presents its findings in three main phases:

The first phase focused on classifying vehicle types using various deep learning techniques. Among the models tested, the DenseNet201 model proved to be the most effective, outperforming the others. By implementing enhancements such as freezing the first 120 layers and using a balanced dataset with data augmentation, the accuracy increased significantly from 96.7% to 98.84%. This improvement highlights the importance of balanced data and model optimization for achieving high accuracy in machine learning tasks.

The second phase involved vehicle detection using the YOLOv5s and YOLOv5sm models. The YOLOv5s model demonstrated high precision and recall across different image sizes and training epochs. Specifically, the YOLOv5sm model, trained with an image size of 416x416 for 100 epochs, achieved impressive results: An F1 score of 89%, precision of 100%, recall of 99%, and mAP@0.5 of 93.2%. These findings highlight the effectiveness of the YOLOv5sm model, particularly when using smaller image sizes, and recommend its use for accurate and reliable vehicle detection.

The final phase involved testing the proposed system in a simulated traffic environment with the goal of optimizing traffic flow. Three scenarios were evaluated based on the presented mathematical formulation, and each scenario showed significant reductions in process times. In the first scenario, the time was reduced from 105 seconds to 5 seconds, representing a 95.24% improvement. The second scenario achieved a reduction from 70 seconds to 5 seconds, resulting in a 92.86% increase in efficiency. Lastly, the third scenario reduced the time from 35 seconds to 5 seconds, leading to an 85.71% improvement. These findings emphasize the substantial impact of the proposed system in reducing process times, suggesting that its standardized application could effectively optimize traffic flow in real-world settings.

5.2 Future Works

In order to improve the intelligent traffic systems, the following future works must be taken into consideration:

- ***Enhanced Dataset Development:*** It is important to continuously update and expand datasets to include a wider range of diverse and challenging scenarios. This will help to improve the robustness of trained models.
- ***Applying another DL technique:*** It is necessary to apply and explore more deep learning architectures and optimization techniques. This exploration will may enhance the accuracy and efficiency of traffic signaling systems.
- ***Real-World Implementation***: Implement the proposed system in the real traffic environment.

- ***Finding the EVs car positions***: Applying another technique for improving the traffic signaling based on the EVs positions. In this case, the lane with EV must be preempted depending on the number of cars until the traffic signal turn from yellow to green.

REFERENCES

AGGARWAL, V. J. A. T. O. I. P. & VISION, C. 2018. A review: deep learning technique for image classification. 4**,** 21.

ALHUDHAIF, A., SAEED, A., IMRAN, T., KAMRAN, M., ALGHAMDI, A. S., ASEERI, A. O. & ALSUBAI, S. J. C. S. S. E. 2022. A Particle Swarm Optimization Based Deep Learning Model for Vehicle Classification. 40**,** 223-235.

ARIFFIN, W. N. S. F. W., KEAT, C. S., PRASATH, T., SURIYAN, L., NORE, N. A. M., HASHIM, N. B. M. & ZAIN, A. S. M. Real-time Dynamic Traffic Light ControlSystem with Emergency Vehicle Priority. Journal of Physics: Conference Series, 2021. IOP Publishing, 012063.

ASHA, C. & NARASIMHADHAN, A. Vehicle counting for traffic management system using YOLO and correlation filter. 2018 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT), 2018. IEEE, 1-6.

BIN CHE MANSOR, M. A. H., KAMAL, N. A. M., BIN BAHAROM, M. H. & BIN ZAINOL, M. A. Emergency Vehicle Type Classification using Convolutional Neural Network. 2021 IEEE International Conference on Automatic Control & Intelligent Systems (I2CACIS), 2021. IEEE, 126-129.

BISWAS, D., SU, H., WANG, C., BLANKENSHIP, J. & STEVANOVIC, A. J. S. 2017. An automatic car counting system using OverFeat framework. 17**,** 1535.

BISWAS, D., SU, H., WANG, C., STEVANOVIC, A. & WANG, W. 2019. An automatic traffic density estimation using Single Shot Detection (SSD) and MobileNet-SSD. *Physics and Chemistry of the Earth, Parts A/B/C,* 110**,** 176-184.

BORISAGAR, P., AGRAWAL, Y. & PAREKH, R. Efficient Vehicle Accident Detection System using Tensorflow and Transfer Learning. 2018

International Conference on Networking, Embedded and Wireless Systems (ICNEWS), 27-28 Dec. 2018 2018. 1-6.

BUI, K.-H. N., YI, H., JUNG, H. & CHO, J. Video-Based Traffic Flow Analysis for Turning Volume Estimation at Signalized Intersections. *In:* NGUYEN, N. T., JEARANAITANAKIJ, K., SELAMAT, A., TRAWIŃSKI, B. & CHITTAYASOTHORN, S., eds. Intelligent Information and Database Systems, 2020// 2020 Cham. Springer International Publishing, 152-162.

CAI, L., WANG, Z., KULATHINAL, R., KUMAR, S., JI, S. J. I. T. O. N. N. & SYSTEMS, L. 2021. Deep low-shot learning for biological image classification and visualization from limited training samples. 34**,** 2528-2538.

CHAMIE, J. J. Y. G. O. 2020. World population: 2020 overview.

CHEN, L., LEI, C. J. D. L. & MINDSPORE, P. W. 2021. Deep learning basics. 17-28.

CHEN, Z., WU, R., LIN, Y., LI, C., CHEN, S., YUAN, Z., CHEN, S. & ZOU, X. J. A. 2022. Plant disease recognition model based on improved YOLOv5. 12**,** 365.

CHOUDEKAR, P., BANERJEE, S. & MUJU, M. Implementation of image processing in real time traffic light control. 2011 3rd International Conference on Electronics Computer Technology, 2011. IEEE, 94-98.

COELHO, M. C., FARIAS, T. L., ROUPHAIL, N. M. J. T. R. P. D. T. & ENVIRONMENT 2005. Impact of speed control traffic signals on pollutant emissions. 10**,** 323-340.

DAI, J., LI, Y., HE, K. & SUN, J. J. A. P. A. 2016. R-FCN: object detection via regionbased fully convolutional networks, CoRR abs/1605.06409.

DOGRU, N. & SUBASI, A. Traffic accident detection using random forest classifier. 2018 15th Learning and Technology Conference (L&T), 25-26 Feb. 2018 2018. 40-45.

DONG, S., WANG, P. & ABBAS, K. J. C. S. R. 2021. A survey on deep learning and its applications. 40**,** 100379.

DU, X., LI, Y., GUO, Y. & XIONG, H. Vision-Based Traffic Light Detection for Intelligent Vehicles. 2017 4th International Conference on Information Science and Control Engineering (ICISCE), 21-23 July 2017 2017. 1323-1326.

EAMTHANAKUL, B., KETCHAM, M. & CHUMUANG, N. The traffic congestion investigating system by image processing from cctv camera. 2017 International Conference on Digital Arts, Media and Technology (ICDAMT), 2017. IEEE, 240-245.

ELLAHYANI, A., ANSARI, M. E. & JAAFARI, I. E. 2016. Traffic sign detection and recognition based on random forests. *Applied Soft Computing,* 46**,** 805-815.

FARAJ, M. A., BOSKANY, N. W. J. U. J. O. S. & TECHNOLOGY 2020. Intelligent Traffic Congestion Control System using Machine Learning and Wireless Network. 4**,** 123-131.

FATIMA, N. J. A. A. I. D. C. & JOURNAL, A. I. 2020. Enhancing performance of a deep neural network: A comparative analysis of optimization algorithms. 9**,** 79-90.

GARG, K., LAM, S.-K., SRIKANTHAN, T. & AGARWAL, V. Real-time road traffic density estimation using block variance. 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), 2016. IEEE, 1-9.

GHAHREMANNEZHAD, H., SHI, H. & LIU, C. Real-Time Accident Detection in Traffic Surveillance Using Deep Learning. 2022 IEEE International Conference on Imaging Systems and Techniques (IST), 21-23 June 2022 2022. 1-6.

GHAZALI, W., ZULKIFLI, C. & PONRAHONO, Z. The effect of traffic congestion on quality of community life. 4th International Conference on Rebuilding Place, 2019.

GIRSHICK, R. 2015. Fast r-cnn. *IEEE International Conference on Computer Vision*. IEEE.

GIRSHICK, R., DONAHUE, J., DARRELL, T. & MALIK, J. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, 2014. 580-587.

GLUČINA, M., ANĐELIĆ, N., LORENCIN, I. & CAR, Z. J. E. 2023. Detection and Classification of Printed Circuit Boards Using YOLO Algorithm. 12**,** 667.

GONG, M. J. I. J. O. M. I. T. V. 2021. A novel performance measure for machine learning classification. 13.

GOUTTE, C. & GAUSSIER, E. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. European conference on information retrieval, 2005. Springer, 345-359.

GUO, G. & ZHANG, Z. 2022. Road damage detection algorithm for improved YOLOv5. *Scientific Reports,* 12**,** 15523.

GUO, Q., LI, L. & BAN, X. 2019. Urban traffic signal control with connected and automated vehicles: A survey. *Transportation Research Part C: Emerging Technologies,* 101**,** 313-334.

HANLEY, J. A. & MCNEIL, B. J. J. R. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. 143**,** 29-36.

HAQUE, M. F., LIM, H.-Y. & KANG, D.-S. Object detection based on VGG with ResNet network. 2019 International Conference on Electronics, Information, and Communication (ICEIC), 2019. IEEE, 1-3.

HAQUE, S., SHARMIN, S. & DEB, K. Emergency vehicle detection using deep convolutional neural network. Proceedings of International Joint

Conference on Advances in Computational Intelligence: IJCACI 2021, 2022. Springer, 535-547.

HE, K. & SUN, J. Convolutional neural networks at constrained time cost. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015. 5353-5360.

HE, K., ZHANG, X., REN, S. & SUN, J. Deep residual learning for image recognition. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016. 770-778.

HINTON, G. E., OSINDERO, S. & TEH, Y.-W. J. N. C. 2006. A fast learning algorithm for deep belief nets. 18**,** 1527-1554.

HINTON, G. E. & SALAKHUTDINOV, R. R. J. S. 2006. Reducing the dimensionality of data with neural networks. 313**,** 504-507.

HOWARD, A. G., ZHU, M., CHEN, B., KALENICHENKO, D., WANG, W., WEYAND, T., ANDREETTO, M. & ADAM, H. J. A. P. A. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications.

HU, W., TAN, T., WANG, L., MAYBANK, S. J. I. T. O. S., MAN, & CYBERNETICS, P. C. 2004. A survey on visual surveillance of object motion and behaviors. 34**,** 334-352.

HUANG, Y.-S., CHUNG, T.-H. & CHEN, C.-T. Modeling traffic signal control systems using timed colour Petri nets. 2005 IEEE International Conference on Systems, Man and Cybernetics, 2005. IEEE, 1759-1764.

HUANG, Y. S. & CHUNG, T. H. 2009a. Modeling and analysis of urban traffic light control systems. *Journal of the Chinese Institute of Engineers,* 32**,** 85-95.

HUANG, Y. S. & CHUNG, T. H. J. J. O. T. C. I. O. E. 2009b. Modeling and analysis of urban traffic light control systems. 32**,** 85-95.

HUSSAIN, T. M., BAIG, A. M., SAADAWI, T. N. & AHMED, S. A. J. I. T. O. V. T. 1995. Infrared pyroelectric sensor for detection of vehicular traffic using digital signal processing techniques. 44**,** 683-689.

IJJINA, E. P., CHAND, D., GUPTA, S. & GOUTHAM, K. Computer Vision-based Accident Detection in Traffic Surveillance. 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 6-8 July 2019 2019. 1-6.

IKIRIWATTE, A., PERERA, D., SAMARAKOON, S., DISSANAYAKE, D. & RUPASIGNHE, P. Traffic density estimation and traffic control using convolutional neural network. 2019 International Conference on Advancements in Computing (ICAC), 2019. IEEE, 323-328.

ISMAILOV, A. S., JO‘RAYEV, Z. B. J. S. & EDUCATION 2022. Study of arduino microcontroller board. 3**,** 172-179.

JACCARD, P. J. B. S. V. S. N. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. 37**,** 547-579.

JAIN, N. K., SAINI, R., MITTAL, P. J. S. C. T. & SOCTA, A. P. O. 2019. A review on traffic monitoring system techniques. 569-577.

JAISWAL, A., GIANCHANDANI, N., SINGH, D., KUMAR, V., KAUR, M. J. J. O. B. S. & DYNAMICS 2021. Classification of the COVID-19 infected patients using DenseNet201 based deep transfer learning. 39**,** 5682-5689.

JASWAL, D., VISHVANATHAN, S., KP, S. J. I. J. O. S. & RESEARCH, E. 2014. Image classification using convolutional neural networks. 5**,** 1661-1668.

JEON, H., LEE, J. & SOHN, K. J. J. O. I. T. S. 2018. Artificial intelligence for traffic signal control based solely on video images. 22**,** 433-445.

JING, P., HUANG, H. & CHEN, L. J. I. 2017. An adaptive traffic signal control in a connected vehicle environment: A systematic review. 8**,** 101.

JONNADULA, E. P. & KHILAR, P. M. A New Hybrid Architecture for Real-Time Detection of Emergency Vehicles. *In:* NAIN, N., VIPPARTHI, S. K. & RAMAN, B., eds. Computer Vision and Image Processing, 2020// 2020a Singapore. Springer Singapore, 413-422.

JONNADULA, E. P. & KHILAR, P. M. A New Hybrid Architecture for Real-Time Detection of Emergency Vehicles. Computer Vision and Image Processing: 4th International Conference, CVIP 2019, Jaipur, India, September 27–29, 2019, Revised Selected Papers, Part II 4, 2020b. Springer, 413-422.

JOO, H., AHMED, S. H. & LIM, Y. J. C. C. 2020. Traffic signal control for smart cities using reinforcement learning. 154**,** 324-330.

KAMAL, K. & HAMID, E.-Z. 2023. A comparison between the VGG16, VGG19 and ResNet50 architecture frameworks for classification of normal and CLAHE processed medical images.

KAPLAN BERKAYA, S., GUNDUZ, H., OZSEN, O., AKINLAR, C. & GUNAL, S. 2016. On circular traffic sign detection and recognition. *Expert Systems with Applications,* 48**,** 67-75.

KATEB, F. A., MONOWAR, M. M., HAMID, M. A., OHI, A. Q. & MRIDHA, M. F. J. A. 2021. FruitDet: Attentive feature aggregation for real-time fruit detection in orchards. 11**,** 2440.

KAUSHIK, S., RAMAN, A. & RAO, K. V. S. R. Leveraging Computer Vision for Emergency Vehicle Detection-Implementation and Analysis. 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 1-3 July 2020 2020. 1-6.

KE, X., SHI, L., GUO, W. & CHEN, D. J. I. T. O. I. T. S. 2018. Multi-dimensional traffic congestion detection based on fusion of visual features and convolutional neural network. 20**,** 2157-2170.

KELLEHER, J. D. 2019. *Deep learning,* MIT press.

KHAN, S. D. & ULLAH, H. 2019. A survey of advances in vision-based vehicle re-identification. *Computer Vision and Image Understanding,* 182**,** 50-63.

KHASAWNEH, N., FRAIWAN, M. & FRAIWAN, L. J. C. C. 2023. Detection of K-complexes in EEG signals using deep transfer learning and YOLOv3. 26**,** 3985-3995.

KRIZHEVSKY, A., SUTSKEVER, I. & HINTON, G. E. J. A. I. N. I. P. S. 2012. Imagenet classification with deep convolutional neural networks. 25.

KUMARAN, S. K., MOHAPATRA, S., DOGRA, D. P., ROY, P. P. & KIM, B.-G. 2019a. Computer vision-guided intelligent traffic signaling for isolated intersections. *Expert Systems with Applications,* 134**,** 267-278.

KUMARAN, S. K., MOHAPATRA, S., DOGRA, D. P., ROY, P. P. & KIM, B.-G. J. E. S. W. A. 2019b. Computer vision-guided intelligent traffic signaling for isolated intersections. 134**,** 267-278.

KURNIAWAN, F., SAJATI, H., DINARYANTO, O. J. I. J. O. E. & TECHNOLOGY 2017. Image processing technique for traffic density estimation. 9**,** 1496-1503.

KUUTTI, S., BOWDEN, R., JIN, Y., BARBER, P. & FALLAH, S. J. I. T. O. I. T. S. 2020. A survey of deep learning applications to autonomous vehicle control. 22**,** 712-733.

LAKSHMI, C. J. & KALPANA, S. Intelligent traffic signaling system. 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), 10-11 March 2017 2017. 247-251.

LECUN, Y., BENGIO, Y. & HINTON, G. J. N. 2015. Deep learning. 521**,** 436-444.

LECUN, Y., BOTTOU, L., BENGIO, Y. & HAFFNER, P. J. P. O. T. I. 1998. Gradient-based learning applied to document recognition. 86**,** 2278-2324.

LEE, W.-H. & CHIU, C.-Y. J. S. 2020. Design and implementation of a smart traffic signal control system for smart city applications. 20**,** 508.

LILLESAND, T., KIEFER, R. W. & CHIPMAN, J. 2015. *Remote sensing and image interpretation*, John Wiley & Sons.

LIN, A. X., HO, A. F. W., CHEONG, K. H., LI, Z., CAI, W., CHEE, M. L., NG, Y. Y., XIAO, X., ONG, M. E. H. J. I. J. O. E. R. & HEALTH, P. 2020. Leveraging Machine Learning Techniques and Engineering of Multi-Nature Features for National Daily Regional Ambulance Demand Prediction. 17**,** 4179.

LIU, P.-H., SU, S.-F., CHEN, M.-C. & HSIAO, C.-C. Deep learning and its application to general image classification. 2015 international conference on informative and cybernetics for computational social systems (ICCSS), 2015. IEEE, 7-10.

LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S., FU, C.-Y. & BERG, A. C. Ssd: Single shot multibox detector. Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14, 2016. Springer, 21-37.

LIU, X., YAN, W. Q. J. M. T. & APPLICATIONS 2021. Traffic-light sign recognition using Capsule network. 80**,** 15161-15171.

MALHI, M. H., ASLAM, M. H., SAEED, F., JAVED, O. & FRAZ, M. Vision based intelligent traffic management system. 2011 Frontiers of Information Technology, 2011. IEEE, 137-141.

MANGURI, K. H. K. 2016. *Traffic Sıgnalıng Control At Hıghway Intersectıons Usıng Morphologıcal Image Processıng Technıque.* Hasan Kalyoncu Üniversitesi.

MANGURI, K. H. K., MOHAMMED, A. A. J. U. J. O. S. & TECHNOLOGY 2023. A Review of Computer Vision–Based Traffic Controlling and Monitoring. 7**,** 6-15.

MINH, T. N., SINN, M., LAM, H. T. & WISTUBA, M. J. A. P. A. 2018. Automated image data preprocessing with deep reinforcement learning.

NARKHEDE, S. 2018. *Understanding Confusion Matrix* [Online]. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 [Accessed July 22 2024].

NIE, C., WEI, H., SHI, J. & ZHANG, M. J. T. R. I. P. 2021. Optimizing actuated traffic signal control using license plate recognition data: Methods for modeling and algorithm development. 9**,** 100319.

OHN-BAR, E. & TRIVEDI, M. M. To boost or not to boost? on the limits of boosted trees for object detection. 2016 23rd international conference on pattern recognition (ICPR), 2016. IEEE, 3350-3355.

OZTURK, S. & FTHENAKIS, V. J. E. 2020. Predicting frequency, time-to-repair and costs of wind turbine failures. 13**,** 1149.

PADILLA, R., COSTA FILHO, C., COSTA, M. J. W. A. O. S., ENGINEERING & TECHNOLOGY 2012. Evaluation of haar cascade classifiers designed for face detection. 64**,** 362-365.

PADILLA, R., NETTO, S. L. & DA SILVA, E. A. A survey on performance metrics for object-detection algorithms. 2020 international conference on systems, signals and image processing (IWSSIP), 2020. IEEE, 237-242.

PON, M. Z. A., KK, K. P. J. S. T. O. A. I. & COMPUTING, Q. 2021. Hyperparameter tuning of deep learning models in keras. 1**,** 36-40.

QADRI, S. S. S. M., GÖKÇE, M. A. & ÖNER, E. 2020. State-of-art review of traffic signal control methods: challenges and opportunities. *European Transport Research Review,* 12**,** 55.

RAJI, C., VP, F. F. & KT, S. S. Emergency Vehicles Detection during Traffic Congestion. 2022 6th International Conference on Trends in Electronics and Informatics (ICOEI), 2022. IEEE, 32-37.

REDMON, J., DIVVALA, S., GIRSHICK, R. & FARHADI, A. You only look once: Unified, real-time object detection. Proceedings of the IEEE conference on computer vision and pattern recognition, 2016. 779-788.

REN, S., HE, K., GIRSHICK, R. & SUN, J. J. A. I. N. I. P. S. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. 28.

ROY, S. & RAHMAN, M. S. Emergency vehicle detection on heavy traffic road from cctv footage using deep convolutional neural network. 2019 international conference on electrical, computer and communication engineering (ECCE), 2019. IEEE, 1-6.

SAINI, A., SUREGAONKAR, S., GUPTA, N., KARAR, V. & PODDAR, S. Region and feature matching based vehicle tracking for accident detection. 2017 Tenth International Conference on Contemporary Computing (IC3), 10-12 Aug. 2017 2017. 1-6.

SALTI, S., PETRELLI, A., TOMBARI, F., FIORAIO, N. & DI STEFANO, L. 2015. Traffic sign detection via interest region extraction. *Pattern Recognition*, 48**,** 1039-1049.

SAMADI, S., RAD, A. P., KAZEMI, F. M. & JAFARIAN, H. J. J. O. T. T. 2012. Performance evaluation of intelligent adaptive traffic control systems: A case study. 2**,** 248.

SERMANET, P., EIGEN, D., ZHANG, X., MATHIEU, M., FERGUS, R. & LECUN, Y. J. A. P. A. 2013. Overfeat: Integrated recognition, localization and detection using convolutional networks.

SHAO, F., WANG, X., MENG, F., RUI, T., WANG, D. & TANG, J. J. S. 2018. Real-time traffic sign detection and recognition method based on simplified Gabor wavelets and CNNs. 18**,** 3192.

SHARMA, M., BANSAL, A., KASHYAP, V., GOYAL, P. & SHEIKH, T. H. Intelligent traffic light control system based on traffic environment using deep learning. IOP Conference Series: Materials Science and Engineering, 2021. IOP Publishing, 012122.

SHIRVANI SHIRI, M. & MALEKI, H. R. J. I. J. O. F. S. 2017. Maximum green time settings for traffic-actuated signal control at isolated intersections using fuzzy logic. 19**,** 247-256.

SHOBEIRI, S., AAJAMI, M. J. A. J. O. S. & ENGINEERING 2021. Shapley value in convolutional neural networks (CNNs): A Comparative Study. 2**,** 9-14.

SHUSTANOV, A. & YAKIMOV, P. 2017. CNN Design for Real-Time Traffic Sign Recognition. *Procedia Engineering,* 201**,** 718-725.

SIMONYAN, K. & ZISSERMAN, A. J. A. P. A. 2014. Very deep convolutional networks for large-scale image recognition.

SUHAIMY, M. A., HALIM, I. S. A., HASSAN, S. L. M. & SAPARON, A. Classification of ambulance siren sound with MFCC-SVM. AIP Conference Proceedings, 2020. AIP Publishing LLC, 020032.

SUN, T., HUANG, Z., ZHU, H., HUANG, Y. & ZHENG, P. J. I. A. 2020. Congestion pattern prediction for a busy traffic zone based on the Hidden Markov Model. 9**,** 2390-2400.

SUN, Z., BEBIS, G., MILLER, R. J. I. T. O. P. A. & INTELLIGENCE, M. 2006. On-road vehicle detection: A review. 28**,** 694-711.

SZEGEDY, C., LIU, W., JIA, Y., SERMANET, P., REED, S., ANGUELOV, D., ERHAN, D., VANHOUCKE, V. & RABINOVICH, A. Going deeper with convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition, 2015. 1-9.

TAMMINA, S. J. I. J. O. S. & PUBLICATIONS, R. 2019. Transfer learning using vgg-16 with deep convolutional neural network for classifying images. 9**,** 143-150.

TARANTO, D. J. P., FACULTY OF TECHNICAL SCIENCES, BITOLA, MACEDONIA 2012. UTOPIA Urban Traffic Control Overview.

TIAN, D., ZHANG, C., DUAN, X. & WANG, X. J. I. A. 2019. An automatic car accident detection method based on cooperative vehicle infrastructure systems. 7**,** 127453-127463.

VERES, M. & MOUSSA, M. J. I. T. O. I. T. S. 2019. Deep learning for intelligent transportation systems: A survey of emerging trends. 21**,** 3152-3168.

VIOLA, P. & JONES, M. Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001, 2001. Ieee, I-I.

VITI, F. & VAN ZUYLEN, H. J. J. T. R. P. C. E. T. 2010. A probabilistic model for traffic at actuated control signals. 18**,** 299-310.

WAHLSTEDT, J. Evaluation of the two self-optimising traffic signal systems Utopia/Spot and ImFlow, and comparison with existing signal control in Stockholm, Sweden. 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), 2013. IEEE, 1541-1546.

WANG, Y., YANG, X., LIANG, H. & LIU, Y. J. J. O. A. T. 2018. A review of the self-adaptive traffic signal control system based on future traffic environment. 2018.

WEI, H., ZHENG, G., GAYAH, V. & LI, Z. J. A. P. A. 2019. A survey on traffic signal control methods.

WENQI, L., DONGYU, L. & MENGHUA, Y. A model of traffic accident prediction based on convolutional neural network. 2017 2nd IEEE International Conference on Intelligent Transportation Engineering (ICITE), 1-3 Sept. 2017 2017. 198-202.

WU, T.-H., WANG, T.-W. & LIU, Y.-Q. Real-time vehicle and distance detection based on improved yolo v5 network. 2021 3rd World Symposium on Artificial Intelligence (WSAI), 2021. IEEE, 24-28.

YANG, Y., LUO, H., XU, H. & WU, F. 2016. Towards Real-Time Traffic Sign Detection and Classification. *IEEE Transactions on Intelligent Transportation Systems,* 17**,** 2022-2031.

YU, Y., XU, M. & GU, J. J. I. I. T. S. 2019. Vision-based traffic accident detection using sparse spatio-temporal features and weighted extreme learning machine. 13**,** 1417-1428.

YUAN, Y., XIONG, Z. & WANG, Q. ACM: Adaptive cross-modal graph convolutional neural networks for RGB-D scene recognition. Proceedings of the AAAI conference on artificial intelligence, 2019. 9176-9184.

ZHANG, G. & WANG, Y. J. I. T. O. I. T. S. 2010. Optimizing minimum and maximum green time settings for traffic actuated control at isolated intersections. 12**,** 164-173.

ZHANG, J., WANG, F.-Y., WANG, K., LIN, W.-H., XU, X. & CHEN, C. J. I. T. O. I. T. S. 2011. Data-driven intelligent transportation systems: A survey. 12**,** 1624-1639.

ZHENG, X., RECKER, W. & CHU, L. J. J. O. I. T. S. 2010. Optimization of control parameters for adaptive traffic-actuated signal control. 14**,** 95-108.

ZHONG, Z., YAN, J., WU, W., SHAO, J. & LIU, C.-L. Practical block-wise neural network architecture generation. Proceedings of the IEEE conference on computer vision and pattern recognition, 2018. 2423-2432.

ZHOU, F., ZHAO, H. & NIE, Z. Safety helmet detection based on YOLOv5. 2021 IEEE International conference on power electronics, computer applications (ICPECA), 2021. IEEE, 6-11.

APPENDIX A: RESULTS OF CLASSIFICATION PROCESS

Image Size: 64*64
No of epochs: 15

Model	Optimizer	accuracy	precision	recall	f1-score
MobileNet	Adam	0.713580247	0.7194	0.7136	0.7146
	Adamax	0.57037037	0.5674	0.5704	0.5665
	Nadam	0.716049383	0.7198	0.716	0.7164
	RMSprop	0.728395062	0.7334	0.7284	0.7255
ResNet50	Adam	0.81399177	0.8272	0.814	0.8187
	Adamax	0.670781893	0.6913	0.6708	0.6793
	Nadam	0.823045267	0.8289	0.823	0.8255
	RMSprop	0.83127572	0.8455	0.8313	0.8363
VGG16	Adam	0.863374486	0.9096	0.8634	0.8742
	Adamax	0.902880658	0.909	0.9029	0.9051
	Nadam	0.902057613	0.9045	0.9021	0.903
	RMSprop	0.902880658	0.9094	0.9029	0.9051
VGG19	Adam	0.903703704	0.9083	0.9037	0.9048
	Adamax	0.863374486	0.8751	0.8634	0.8671
	Nadam	0.890534979	0.9077	0.8905	0.8952
	RMSprop	0.86255144	0.8972	0.8626	0.872
DenseNet201 Freeze 0	Adam	0.824261275	0.8278	0.8243	0.8258
	Adamax	0.702954899	0.7082	0.703	0.7015
	Nadam	0.828926905	0.8302	0.8289	0.8292
	RMSprop	0.841368585	0.8457	0.8414	0.843
DenseNet201 Freeze 0-30	Adam	0.846034215	0.8499	0.846	0.8472
	Adamax	0.712286159	0.7175	0.7123	0.7131
	Nadam	0.833592535	0.8335	0.8336	0.8333
	RMSprop	0.836702955	0.8382	0.8367	0.8371
DenseNet201 Freeze 0-60	Adam	0.842923795	0.8482	0.8429	0.8444
	Adamax	0.727838258	0.7296	0.7278	0.725
	Nadam	0.810264386	0.8121	0.8103	0.8107
	RMSprop	0.822706065	0.8211	0.8227	0.8211
DenseNet201 Freeze 0-90	Adam	0.839813375	0.8372	0.8398	0.8381
	Adamax	0.724727838	0.7278	0.7247	0.7238
	Nadam	0.822706065	0.8249	0.8227	0.8235
	RMSprop	0.816485226	0.8157	0.8165	0.8157
DenseNet201 Freeze 0-120	Adam	0.786936236	0.7898	0.7869	0.7877
	Adamax	0.702954899	0.7091	0.703	0.7042
	Nadam	0.804043546	0.802	0.804	0.8028
	RMSprop	0.828926905	0.8271	0.8289	0.8276
DenseNet201 Freeze 0-150	Adam	0.800933126	0.7967	0.8009	0.7984
	Adamax	0.707620529	0.6987	0.7076	0.699

	Nadam	0.804043546	0.8024	0.804	0.8026
	RMSprop	0.808709176	0.8132	0.8087	0.809

Image Size: 64*64
No of epochs: 20

Model	Optimizer	accuracy	precision	recall	f1-score
MobileNet	Adam	0.734979424	0.7437	0.735	0.7368
	Adamax	0.595884774	0.5957	0.5959	0.5893
	Nadam	0.753909465	0.7592	0.7539	0.7552
	RMSprop	0.748971193	0.7601	0.749	0.7525
ResNet50	Adam	0.810699588	0.8247	0.8107	0.8142
	Adamax	0.719341564	0.7305	0.7193	0.724
	Nadam	0.825514403	0.8281	0.8255	0.8265
	RMSprop	0.826337449	0.8425	0.8263	0.832
VGG16	Adam	0.883950617	0.8917	0.884	0.8857
	Adamax	0.869958848	0.8928	0.87	0.8758
	Nadam	0.891358025	0.8977	0.8914	0.8935
	RMSprop	0.882304527	0.9079	0.8823	0.8885
VGG19	Adam	0.900411523	0.9044	0.9004	0.9017
	Adamax	0.891358025	0.9031	0.8914	0.8953
	Nadam	0.885596708	0.9054	0.8856	0.8917
	RMSprop	0.89382716	0.9101	0.8938	0.898
DenseNet201 Freeze 0	Adam	0.872427984	0.8857	0.8724	0.8771
	Adamax	0.795884774	0.8179	0.7959	0.8027
	Nadam	0.795884774	0.8179	0.7959	0.8027
	RMSprop	0.869958848	0.8766	0.87	0.872
DenseNet201 Freeze 0-30	Adam	0.865843621	0.8755	0.8658	0.8694
	Adamax	0.803292181	0.8116	0.8033	0.8059
	Nadam	0.86255144	0.8723	0.8626	0.8663
	RMSprop	0.872427984	0.8885	0.8724	0.8779
DenseNet201 Freeze 0-60	Adam	0.868312757	0.8752	0.8683	0.871
	Adamax	0.786831276	0.8016	0.7868	0.7901
	Nadam	0.855967078	0.8632	0.856	0.8587
	RMSprop	0.867489712	0.8727	0.8675	0.8688
DenseNet201 Freeze 0-90	Adam	0.874897119	0.8766	0.8749	0.8756
	Adamax	0.799176955	0.8101	0.7992	0.8027
	Nadam	0.852674897	0.863	0.8527	0.8566
	RMSprop	0.86255144	0.8687	0.8626	0.8651
DenseNet201 Freeze 0-120	Adam	0.852674897	0.8659	0.8527	0.857
	Adamax	0.790946502	0.8008	0.7909	0.7944
	Nadam	0.857613169	0.8694	0.8576	0.8614
	RMSprop	0.857613169	0.8702	0.8576	0.8618

DenseNet201 Freeze 0-150	Adam	0.84691358	0.8544	0.8469	0.8499
	Adamax	0.786831276	0.7909	0.7868	0.7876
	Nadam	0.855144033	0.8596	0.8551	0.8565
	RMSprop	0.854320988	0.8601	0.8543	0.8562

Image Size: 64*64
No of epochs: 25

Model	Optimizer	accuracy	precision	recall	f1-score
MobileNet	Adam	0.769547325	0.7853	0.7695	0.7757
	Adamax	0.635390947	0.6317	0.6354	0.6334
	Nadam	0.771193416	0.7805	0.7712	0.7749
	RMSprop	0.78600823	0.7867	0.786	0.7856
ResNet50	Adam	0.827983539	0.8341	0.828	0.8305
	Adamax	0.827983539	0.8341	0.828	0.8305
	Nadam	0.826337449	0.8307	0.8263	0.8275
	RMSprop	0.832098765	0.8403	0.8321	0.835
VGG16	Adam	0.903703704	0.9116	0.9037	0.9061
	Adamax	0.897119342	0.901	0.8971	0.8983
	Nadam	0.879012346	0.9049	0.879	0.8867
	RMSprop	0.90781893	0.9163	0.9078	0.9103
VGG19	Adam	0.90617284	0.9069	0.9062	0.9059
	Adamax	0.887242798	0.8911	0.8872	0.8886
	Nadam	0.887242798	0.8928	0.8872	0.8885
	RMSprop	0.905349794	0.9024	0.9053	0.9023
DenseNet201 Freeze 0	Adam	0.865020576	0.8763	0.865	0.8692
	Adamax	0.81399177	0.8179	0.814	0.815
	Nadam	0.869135802	0.8802	0.8691	0.8723
	RMSprop	0.867489712	0.8845	0.8675	0.8731
DenseNet201 Freeze 0-30	Adam	0.869135802	0.872	0.8691	0.8698
	Adamax	0.817283951	0.8273	0.8173	0.8209
	Nadam	0.875720165	0.8885	0.8757	0.8799
	RMSprop	0.888065844	0.8903	0.8881	0.8887
DenseNet201 Freeze 0-60	Adam	0.855967078	0.8679	0.856	0.86
	Adamax	0.811522634	0.8183	0.8115	0.8137
	Nadam	0.865843621	0.8758	0.8658	0.8694
	RMSprop	0.885596708	0.8905	0.8856	0.8876
DenseNet201 Freeze 0-90	Adam	0.864197531	0.8736	0.8642	0.8675
	Adamax	0.809053498	0.8248	0.8091	0.8155
	Nadam	0.851028807	0.8676	0.851	0.857
	RMSprop	0.863374486	0.873	0.8634	0.866
DenseNet201 Freeze 0-120	Adam	0.86090535	0.8681	0.8609	0.8636
	Adamax	0.791769547	0.8078	0.7918	0.7982
	Nadam	0.849382716	0.8678	0.8494	0.8552

	RMSprop	0.856790123	0.8647	0.8568	0.8594
DenseNet201 Freeze 0-150	Adam	0.849382716	0.8581	0.8494	0.8521
	Adamax	0.776954733	0.7913	0.777	0.7816
	Nadam	0.860082305	0.8715	0.8601	0.8637
	RMSprop	0.855967078	0.8646	0.856	0.8592

Image Size: 64*64
No of epochs: 30

Model	Optimizer	accuracy	precision	recall	f1-score
MobileNet	Adam	0.775308642	0.775309	0.7753	0.77531
	Adamax	0.667489712	0.671	0.6675	0.6684
	Nadam	0.795884774	0.8067	0.7959	0.7999
	RMSprop	0.800823045	0.8114	0.8008	0.8049
ResNet50	Adam	0.846090535	0.8579	0.8461	0.8495
	Adamax	0.778600823	0.7893	0.7786	0.782
	Nadam	0.84526749	0.851	0.8453	0.8472
	RMSprop	0.82962963	0.8419	0.8296	0.8328
VGG16	Adam	0.897942387	0.9044	0.8979	0.9
	Adamax	0.895473251	0.8986	0.8955	0.8967
	Nadam	0.902880658	0.9105	0.9029	0.9048
	RMSprop	0.898765432	0.9161	0.8988	0.9034
VGG19	Adam	0.902057613	0.9042	0.9021	0.9011
	Adamax	0.888065844	0.8965	0.8881	0.8911
	Nadam	0.906995885	0.9165	0.907	0.9101
	RMSprop	0.880658436	0.8868	0.8807	0.8779
DenseNet201 Freeze 0	Adam	0.872427984	0.8795	0.8724	0.874
	Adamax	0.824691358	0.8306	0.8247	0.827
	Nadam	0.87654321	0.8875	0.8765	0.8799
	RMSprop	0.880658436	0.8996	0.8807	0.8868
DenseNet201 Freeze 0-30	Adam	0.868312757	0.8835	0.8683	0.8729
	Adamax	0.834567901	0.8406	0.8346	0.8364
	Nadam	0.865843621	0.8794	0.8658	0.8706
	RMSprop	0.879012346	0.8942	0.879	0.8839
DenseNet201 Freeze 0-60	Adam	0.874897119	0.8857	0.8749	0.8788
	Adamax	0.819753086	0.8312	0.8198	0.824
	Nadam	0.861728395	0.8734	0.8617	0.8661
	RMSprop	0.873251029	0.8861	0.8733	0.8773
DenseNet201 Freeze 0-90	Adam	0.861728395	0.8643	0.8617	0.8626
	Adamax	0.821399177	0.8305	0.8214	0.8248
	Nadam	0.871604938	0.8756	0.8716	0.8733
	RMSprop	0.874074074	0.8856	0.8741	0.8781
DenseNet201 Freeze 0-120	Adam	0.855144033	0.8692	0.8551	0.8603
	Adamax	0.81399177	0.8257	0.814	0.8187

	Nadam	0.857613169	0.8676	0.8576	0.8612
	RMSprop	0.847736626	0.8681	0.8477	0.8542
DenseNet201 Freeze 0-150	Adam	0.864197531	0.8709	0.8642	0.8665
	Adamax	0.788477366	0.8079	0.7885	0.7956
	Nadam	0.84526749	0.8579	0.8453	0.8487
	RMSprop	0.863374486	0.8634	0.8634	0.8632

Image Size: 128*128
No of epochs: 15

Model	Optimizer	accuracy	precision	recall	f1-score
MobileNet	Adam	0.950617284	0.9508	0.9506	0.9507
	Adamax	0.930041152	0.9291	0.93	0.9292
	Nadam	0.947325103	0.947	0.9473	0.947
	RMSprop	0.950617284	0.9503	0.9506	0.9504
ResNet50	Adam	0.922633745	0.9261	0.9226	0.9236
	Adamax	0.914403292	0.9151	0.9144	0.9146
	Nadam	0.925925926	0.9275	0.9259	0.9266
	RMSprop	0.924279835	0.9245	0.9243	0.9241
VGG16	Adam	0.945679012	0.946	0.9457	0.9452
	Adamax	0.942386831	0.9429	0.9424	0.9426
	Nadam	0.948148148	0.9504	0.9481	0.9484
	RMSprop	0.9218107	0.9295	0.9218	0.9221
VGG19	Adam	0.946502058	0.9461	0.9465	0.9461
	Adamax	0.93744856	0.941	0.9374	0.9384
	Nadam	0.95473251	0.9552	0.9547	0.9549
	RMSprop	0.940740741	0.9423	0.9407	0.9412
DenseNet201 Freeze 0	Adam	0.956378601	0.9574	0.9564	0.9567
	Adamax	0.928395062	0.9293	0.9284	0.9285
	Nadam	0.963786008	0.9637	0.9638	0.9637
	RMSprop	0.959670782	0.9598	0.9597	0.9597
DenseNet201 Freeze 0-30	Adam	0.956378601	0.9573	0.9564	0.9567
	Adamax	0.935802469	0.9356	0.9358	0.9356
	Nadam	0.960493827	0.9604	0.9605	0.9604
	RMSprop	0.953909465	0.955	0.9539	0.9541
DenseNet201 Freeze 0-60	Adam	0.950617284	0.9527	0.9506	0.9514
	Adamax	0.935802469	0.9386	0.9358	0.9368
	Nadam	0.958024691	0.9592	0.958	0.9584
	RMSprop	0.95308642	0.9543	0.9531	0.9535
DenseNet201 Freeze 0-90	Adam	0.951440329	0.9517	0.9514	0.9515
	Adamax	0.922633745	0.9223	0.9226	0.9222
	Nadam	0.95473251	0.9557	0.9547	0.955
	RMSprop	0.940740741	0.946	0.9407	0.9422
DenseNet201 Freeze 0-120	Adam	0.950617284	0.9518	0.9506	0.9509

	Adamax	0.926748971	0.9289	0.9267	0.9276
	Nadam	0.944855967	0.9468	0.9449	0.9454
	RMSprop	0.943209877	0.9454	0.9432	0.9438
DenseNet201 Freeze 0-150	Adam	0.951440329	0.9524	0.9514	0.9517
	Adamax	0.930041152	0.9295	0.93	0.9295
	Nadam	0.939917695	0.942	0.9399	0.9404
	RMSprop	0.949794239	0.9511	0.9498	0.9502

Image Size: 128*128
No of epochs: 20

Model	Optimizer	accuracy	precision	recall	f1-score
MobileNet	Adam	0.95473251	0.9551	0.9547	0.9549
	Adamax	0.942386831	0.9417	0.9424	0.9419
	Nadam	0.941563786	0.9417	0.9416	0.9416
	RMSprop	0.95308642	0.9531	0.9531	0.9531
ResNet50	Adam	0.922633745	0.9286	0.9226	0.9243
	Adamax	0.914403292	0.915	0.9144	0.9146
	Nadam	0.928395062	0.9302	0.9284	0.929
	RMSprop	0.926748971	0.9291	0.9267	0.9276
VGG16	Adam	0.95308642	0.9545	0.9531	0.9531
	Adamax	0.948148148	0.9511	0.9481	0.949
	Nadam	0.943209877	0.945	0.9432	0.9438
	RMSprop	0.950617284	0.9532	0.9506	0.9515
VGG19	Adam	0.955555556	0.9562	0.9556	0.9558
	Adamax	0.936625514	0.9382	0.9366	0.9372
	Nadam	0.938271605	0.938	0.9383	0.9371
	RMSprop	0.950617284	0.9526	0.9506	0.9513
DenseNet201 Freeze 0	Adam	0.956378601	0.9563	0.9564	0.9562
	Adamax	0.941563786	0.944	0.9416	0.9425
	Nadam	0.956378601	0.9566	0.9564	0.9565
	RMSprop	0.967901235	0.9684	0.9679	0.9681
DenseNet201 Freeze 0-30	Adam	0.958024691	0.9586	0.958	0.9582
	Adamax	0.943209877	0.9448	0.9432	0.9437
	Nadam	0.955555556	0.9569	0.9556	0.956
	RMSprop	0.956378601	0.9566	0.9564	0.9562
DenseNet201 Freeze 0-60	Adam	0.952263374	0.9534	0.9523	0.9527
	Adamax	0.930864198	0.9312	0.9309	0.931
	Nadam	0.948148148	0.949	0.9481	0.9483
	RMSprop	0.956378601	0.9566	0.9564	0.9565
DenseNet201 Freeze 0-90	Adam	0.947325103	0.9487	0.9473	0.9477
	Adamax	0.941563786	0.9427	0.9416	0.942
	Nadam	0.953909465	0.955	0.9539	0.9543
	RMSprop	0.949794239	0.9507	0.9498	0.95

DenseNet201 Freeze 0-120	Adam	0.95308642	0.9538	0.9531	0.9533
	Adamax	0.930864198	0.9353	0.9309	0.9324
	Nadam	0.953909465	0.9569	0.9539	0.9548
	RMSprop	0.955555556	0.9568	0.9556	0.956
DenseNet201 Freeze 0-150	Adam	0.948148148	0.9506	0.9481	0.9487
	Adamax	0.938271605	0.9385	0.9383	0.9383
	Nadam	0.953909465	0.9542	0.9539	0.9539
	RMSprop	0.948148148	0.95	0.9481	0.9486

Image Size: 128*128
No of epochs: 25

Model	Optimizer	accuracy	precision	recall	f1-score
MobileNet	Adam	0.953909465	0.954	0.9539	0.9539
	Adamax	0.941563786	0.9412	0.9416	0.9413
	Nadam	0.948148148	0.9481	0.9481	0.948
	RMSprop	0.949794239	0.9513	0.9498	0.9503
ResNet50	Adam	0.926748971	0.9319	0.9267	0.9282
	Adamax	0.919341564	0.9195	0.9193	0.9193
	Nadam	0.930864198	0.9338	0.9309	0.9318
	RMSprop	0.9218107	0.924	0.9218	0.9224
VGG16	Adam	0.95473251	0.9552	0.9547	0.9548
	Adamax	0.929218107	0.9335	0.9292	0.9299
	Nadam	0.948971193	0.9491	0.949	0.9488
	RMSprop	0.95308642	0.9553	0.9531	0.9538
VGG19	Adam	0.934156379	0.9358	0.9342	0.9343
	Adamax	0.938271605	0.9385	0.9383	0.9384
	Nadam	0.952263374	0.9522	0.9523	0.952
	RMSprop	0.947325103	0.9478	0.9473	0.947
DenseNet201 Freeze 0	Adam	0.960493827	0.961	0.9605	0.9607
	Adamax	0.941563786	0.9427	0.9416	0.9419
	Nadam	0.953909465	0.9552	0.9539	0.9541
	RMSprop	0.947325103	0.9486	0.9473	0.9478
DenseNet201 Freeze 0-30	Adam	0.962139918	0.9621	0.9621	0.962
	Adamax	0.948971193	0.9494	0.949	0.9488
	Nadam	0.958024691	0.9582	0.958	0.9581
	RMSprop	0.947325103	0.949	0.9473	0.9479
DenseNet201 Freeze 0-60	Adam	0.951440329	0.9527	0.9514	0.9519
	Adamax	0.948148148	0.95	0.9481	0.9488
	Nadam	0.949794239	0.9504	0.9498	0.9498
	RMSprop	0.93744856	0.938	0.9374	0.9375
DenseNet201 Freeze 0-90	Adam	0.953909465	0.9546	0.9539	0.9541
	Adamax	0.944855967	0.9455	0.9449	0.9451
	Nadam	0.953909465	0.9547	0.9539	0.9542

	RMSprop	0.93744856	0.9386	0.9374	0.9378
DenseNet201 Freeze 0-120	Adam	0.949794239	0.9521	0.9498	0.9506
	Adamax	0.936625514	0.9392	0.9366	0.9375
	Nadam	0.953909465	0.955	0.9539	0.9543
	RMSprop	0.941563786	0.9439	0.9416	0.9424
DenseNet201 Freeze 0-150	Adam	0.949794239	0.951	0.9498	0.9502
	Adamax	0.942386831	0.9432	0.9424	0.9427
	Nadam	0.95473251	0.9553	0.9547	0.9547
	RMSprop	0.922633745	0.9292	0.9226	0.9248

Image Size: 128*128
No of epochs: 30

Model	Optimizer	accuracy	precision	recall	f1-score
MobileNet	Adam	0.950617284	0.9505	0.9506	0.9504
	Adamax	0.945679012	0.9458	0.9457	0.9456
	Nadam	0.950617284	0.9522	0.9506	0.9512
	RMSprop	0.944032922	0.9452	0.944	0.9444
ResNet50	Adam	0.938271605	0.9394	0.9383	0.9387
	Adamax	0.904526749	0.9054	0.9045	0.9047
	Nadam	0.934156379	0.9364	0.9342	0.9349
	RMSprop	0.924279835	0.9254	0.9243	0.9247
VGG16	Adam	0.934156379	0.9394	0.9342	0.935
	Adamax	0.948148148	0.948	0.9481	0.9479
	Nadam	0.95308642	0.9526	0.9531	0.9526
	RMSprop	0.955555556	0.9558	0.9556	0.9556
VGG19	Adam	0.935802469	0.9466	0.9358	0.9384
	Adamax	0.939917695	0.9404	0.9399	0.94
	Nadam	0.958847737	0.9605	0.9588	0.9593
	RMSprop	0.944032922	0.9492	0.944	0.945
DenseNet201 Freeze 0	Adam	0.958847737	0.9595	0.9588	0.9591
	Adamax	0.944855967	0.9456	0.9449	0.9451
	Nadam	0.958847737	0.9594	0.9588	0.9589
	RMSprop	0.950617284	0.9517	0.9506	0.9506
DenseNet201 Freeze 0-30	Adam	0.958024691	0.9586	0.958	0.958
	Adamax	0.939917695	0.9417	0.9399	0.9404
	Nadam	0.957201646	0.9588	0.9572	0.9577
	RMSprop	0.952263374	0.9539	0.9523	0.9527
DenseNet201 Freeze 0-60	Adam	0.955555556	0.9576	0.9556	0.9563
	Adamax	0.933333333	0.9366	0.9333	0.9344
	Nadam	0.949794239	0.9501	0.9498	0.9497
	RMSprop	0.948148148	0.9523	0.9481	0.9494
DenseNet201 Freeze 0-90	Adam	0.962139918	0.9626	0.9621	0.9623
	Adamax	0.93744856	0.9392	0.9374	0.9379

	Nadam	0.957201646	0.9577	0.9572	0.9574
	RMSprop	0.949794239	0.952	0.9498	0.9504
DenseNet201 Freeze 0-120	Adam	0.95308642	0.9547	0.9531	0.9533
	Adamax	0.93909465	0.9393	0.9391	0.9391
	Nadam	0.947325103	0.9491	0.9473	0.9479
	RMSprop	0.948971193	0.9517	0.949	0.9495
DenseNet201 Freeze 0-150	Adam	0.95308642	0.9541	0.9531	0.9534
	Adamax	0.93909465	0.9413	0.9391	0.9398
	Nadam	0.949794239	0.9505	0.9498	0.9499
	RMSprop	0.95308642	0.953	0.9531	0.9529

Image Size: 224*224
No of epochs: 15

Model	Optimizer	accuracy	precision	recall	f1-score
MobileNet	Adam	0.960493827	0.9608	0.9605	0.9606
	Adamax	0.953909465	0.9541	0.9539	0.954
	Nadam	0.963786008	0.9636	0.9638	0.9635
	RMSprop	0.957201646	0.9574	0.9572	0.9573
ResNet50	Adam	0.967078189	0.9676	0.9671	0.9672
	Adamax	0.960493827	0.9604	0.9605	0.9604
	Nadam	0.967901235	0.9683	0.9679	0.9681
	RMSprop	0.961316872	0.9617	0.9613	0.961
VGG16	Adam	0.960493827	0.9619	0.9605	0.961
	Adamax	0.942386831	0.9423	0.9424	0.9423
	Nadam	0.952263374	0.9533	0.9523	0.951
	RMSprop	0.944032922	0.955	0.944	0.9464
VGG19	Adam	0.957201646	0.9591	0.9572	0.9569
	Adamax	0.950617284	0.9504	0.9506	0.9502
	Nadam	0.972839506	0.9726	0.9728	0.9726
	RMSprop	0.966255144	0.9665	0.9663	0.9657
DenseNet201 Freeze 0	Adam	0.973662551	0.9739	0.9737	0.9737
	Adamax	0.977777778	0.9777	0.9778	0.9777
	Nadam	0.978600823	0.9787	0.9786	0.9786
	RMSprop	0.979423868	0.9794	0.9794	0.9793
DenseNet201 Freeze 0-30	Adam	0.98600823	0.9861	0.986	0.986
	Adamax	0.976954733	0.9771	0.977	0.977
	Nadam	0.983539095	0.9835	0.9835	0.9835
	RMSprop	0.976954733	0.978	0.977	0.9773
DenseNet201 Freeze 0-60	Adam	0.980246914	0.9805	0.9802	0.9803
	Adamax	0.972839506	0.9729	0.9728	0.9728
	Nadam	0.977777778	0.9777	0.9778	0.9777
	RMSprop	0.974485597	0.9747	0.9745	0.9746
DenseNet201 Freeze 0-90	Adam	0.980246914	0.9805	0.9802	0.9803

	Adamax	0.973662551	0.9738	0.9737	0.9737
	Nadam	0.975308642	0.9752	0.9753	0.9752
	RMSprop	0.976131687	0.9765	0.9761	0.9762
DenseNet201 Freeze 0-120	Adam	0.980246914	0.9802	0.9802	0.9802
	Adamax	0.973662551	0.9741	0.9737	0.9738
	Nadam	0.981069959	0.9811	0.9811	0.9811
	RMSprop	0.977777778	0.978	0.9778	0.9777
DenseNet201 Freeze 0-150	Adam	0.977777778	0.9778	0.9778	0.9778
	Adamax	0.96872428	0.9686	0.9687	0.9686
	Nadam	0.980246914	0.9804	0.9802	0.9803
	RMSprop	0.978600823	0.9789	0.9786	0.9787

Image Size: 224*224
No of epochs: 20

Model	Optimizer	accuracy	precision	recall	f1-score
MobileNet	Adam	0.967901235	0.9677	0.9679	0.9677
	Adamax	0.944855967	0.9453	0.9449	0.945
	Nadam	0.96872428	0.9685	0.9687	0.9686
	RMSprop	0.946502058	0.9467	0.9465	0.9465
ResNet50	Adam	0.97037037	0.9706	0.9704	0.9705
	Adamax	0.961316872	0.9613	0.9613	0.9613
	Nadam	0.974485597	0.9745	0.9745	0.9745
	RMSprop	0.967901235	0.9679	0.9679	0.9679
VGG16	Adam	0.967901235	0.9678	0.9679	0.9677
	Adamax	0.944032922	0.9436	0.944	0.9436
	Nadam	0.969547325	0.9699	0.9695	0.9696
	RMSprop	0.948971193	0.9592	0.949	0.9513
VGG19	Adam	0.920164609	0.9254	0.9202	0.92
	Adamax	0.942386831	0.9461	0.9424	0.9429
	Nadam	0.962962963	0.9633	0.963	0.9631
	RMSprop	0.97037037	0.9718	0.9704	0.9707
DenseNet201 Freeze 0	Adam	0.979423868	0.9794	0.9794	0.9794
	Adamax	0.975308642	0.9755	0.9753	0.9754
	Nadam	0.980246914	0.9803	0.9802	0.9803
	RMSprop	0.981893004	0.9818	0.9819	0.9819
DenseNet201 Freeze 0-30	Adam	0.967901235	0.9688	0.9679	0.9681
	Adamax	0.972016461	0.9722	0.972	0.9721
	Nadam	0.974485597	0.9746	0.9745	0.9745
	RMSprop	0.976131687	0.9762	0.9761	0.9758
DenseNet201 Freeze 0-60	Adam	0.980246914	0.9803	0.9802	0.9802
	Adamax	0.975308642	0.9754	0.9753	0.9753
	Nadam	0.981069959	0.9813	0.9811	0.9812
	RMSprop	0.982716049	0.9827	0.9827	0.9827

DenseNet201 Freeze 0-90	Adam	0.982716049	0.9827	0.9827	0.9827
	Adamax	0.97037037	0.9705	0.9704	0.9704
	Nadam	0.980246914	0.9805	0.9802	0.9803
	RMSprop	0.972839506	0.9743	0.9728	0.9732
DenseNet201 Freeze 0-120	Adam	0.975308642	0.9754	0.9753	0.975
	Adamax	0.974485597	0.9746	0.9745	0.9745
	Nadam	0.981069959	0.981	0.9811	0.981
	RMSprop	0.974485597	0.9754	0.9745	0.9747
DenseNet201 Freeze 0-150	Adam	0.981069959	0.981	0.9811	0.981
	Adamax	0.976131687	0.9762	0.9761	0.9762
	Nadam	0.97037037	0.971	0.9704	0.9706
	RMSprop	0.978600823	0.9787	0.9786	0.9786

Image Size: 224*224
No of epochs: 25

Model	Optimizer	accuracy	precision	recall	f1-score
MobileNet	Adam	0.962139918	0.9621	0.9621	0.962
	Adamax	0.953909465	0.9539	0.9539	0.9538
	Nadam	0.958024691	0.9578	0.958	0.9576
	RMSprop	0.959670782	0.9606	0.9597	0.96
ResNet50	Adam	0.967901235	0.9679	0.9679	0.9678
	Adamax	0.961316872	0.9611	0.9613	0.9611
	Nadam	0.967078189	0.9672	0.9671	0.9671
	RMSprop	0.972839506	0.973	0.9728	0.9729
VGG16	Adam	0.929218107	0.9376	0.9292	0.9311
	Adamax	0.93744856	0.951	0.9374	0.9408
	Nadam	0.958024691	0.9648	0.958	0.9596
	RMSprop	0.927572016	0.9366	0.9276	0.9211
VGG19	Adam	0.967901235	0.9689	0.9679	0.9682
	Adamax	0.952263374	0.9527	0.9523	0.952
	Nadam	0.963786008	0.9643	0.9638	0.9639
	RMSprop	0.952263374	0.9555	0.9523	0.9527
DenseNet201 Freeze 0	Adam	0.981069959	0.981	0.9811	0.981
	Adamax	0.976954733	0.9769	0.977	0.9769
	Nadam	0.979423868	0.9795	0.9794	0.9795
	RMSprop	0.978600823	0.9786	0.9786	0.9786
DenseNet201 Freeze 0-30	Adam	0.979423868	0.9794	0.9794	0.9793
	Adamax	0.979423868	0.9796	0.9794	0.9795
	Nadam	0.981893004	0.9819	0.9819	0.9818
	RMSprop	0.981069959	0.981	0.9811	0.981
DenseNet201 Freeze 0-60	Adam	0.981893004	0.9819	0.9819	0.9819
	Adamax	0.977777778	0.9779	0.9778	0.9778
	Nadam	0.983539095	0.9835	0.9835	0.9835

	RMSprop	0.983539095	0.9835	0.9835	0.9835
DenseNet201 Freeze 0-90	Adam	0.98436214	0.9844	0.9844	0.9843
	Adamax	0.966255144	0.9665	0.9663	0.9663
	Nadam	0.982716049	0.9828	0.9827	0.9828
	RMSprop	0.982716049	0.9828	0.9827	0.9827
DenseNet201 Freeze 0-120	Adam	0.976131687	0.9763	0.9761	0.9762
	Adamax	0.978600823	0.9787	0.9786	0.9786
	Nadam	0.976954733	0.9773	0.977	0.9771
	RMSprop	0.975308642	0.9753	0.9753	0.9752
DenseNet201 Freeze 0-150	Adam	0.975308642	0.9752	0.9753	0.9753
	Adamax	0.976954733	0.977	0.977	0.9769
	Nadam	0.981069959	0.981	0.9811	0.981
	RMSprop	0.986831276	0.9871	0.9868	0.9869

Image Size: 224*224
No of epochs: 30

Model	Optimizer	accuracy	precision	recall	f1-score
MobileNet	Adam	0.967078189	0.9673	0.9671	0.9672
	Adamax	0.961316872	0.9614	0.9613	0.9609
	Nadam	0.96872428	0.9688	0.9687	0.9688
	RMSprop	0.963786008	0.9638	0.9638	0.9636
ResNet50	Adam	0.967078189	0.9673	0.9671	0.9671
	Adamax	0.967078189	0.9671	0.9671	0.9671
	Nadam	0.960493827	0.9611	0.9605	0.9607
	RMSprop	0.959670782	0.9615	0.9597	0.9602
VGG16	Adam	0.967078189	0.9685	0.9671	0.9675
	Adamax	0.95308642	0.9553	0.9531	0.9538
	Nadam	0.966255144	0.9678	0.9663	0.9667
	RMSprop	0.967901235	0.9708	0.9679	0.9686
VGG19	Adam	0.969547325	0.9694	0.9695	0.9688
	Adamax	0.948971193	0.9484	0.949	0.9481
	Nadam	0.967901235	0.9688	0.9679	0.9681
	RMSprop	0.963786008	0.9647	0.9638	0.9639
DenseNet201 Freeze 0	Adam	0.976131687	0.9763	0.9761	0.9762
	Adamax	0.973662551	0.9737	0.9737	0.9737
	Nadam	0.98436214	0.9844	0.9844	0.9844
	RMSprop	0.976131687	0.9763	0.9761	0.9762
DenseNet201 Freeze 0-30	Adam	0.978600823	0.9786	0.9786	0.9783
	Adamax	0.978600823	0.9786	0.9786	0.9786
	Nadam	0.978600823	0.9785	0.9786	0.9785
	RMSprop	0.977777778	0.9778	0.9778	0.9778
DenseNet201 Freeze 0-60	Adam	0.98436214	0.9844	0.9844	0.9843
	Adamax	0.976131687	0.9765	0.9761	0.9762

	Nadam	0.97037037	0.9713	0.9704	0.9707
	RMSprop	0.982716049	0.9828	0.9827	0.9827
DenseNet201 Freeze 0-90	Adam	0.979423868	0.9795	0.9794	0.9794
	Adamax	0.976954733	0.9769	0.977	0.9769
	Nadam	0.980246914	0.9803	0.9802	0.9803
	RMSprop	0.985185185	0.9852	0.9852	0.9851
DenseNet201 Freeze 0-120	Adam	0.982716049	0.9827	0.9827	0.9827
	Adamax	0.972839506	0.9732	0.9728	0.973
	Nadam	0.983539095	0.9837	0.9835	0.9836
	RMSprop	0.988477366	0.9885	0.9885	0.9885
DenseNet201 Freeze 0-150	Adam	0.976954733	0.977	0.977	0.9769
	Adamax	0.969547325	0.97	0.9695	0.9697
	Nadam	0.976954733	0.977	0.977	0.9767
	RMSprop	0.975308642	0.9751	0.9753	0.9752

پوختە

ژمارەی دانیشتوانی جیهان بە شێوەیەکی بەرچاو زیادی کردووە، ئەمەش کاریگەری لەسەر بەکارهێنانی ئۆتۆمبێل لەلایەن تاکەکانەوە هەیە و دەبێتە هۆی زیادبوونی ژمارەی ئۆتۆمبێل لە شارەکاندا. بەهۆی هەبونی پەیوەندی ڕاستەوخۆی نێوان دانیشتووان و بەکارهێنانی ئۆتۆمبێل، ئەمەش وادەکات کە بەڕێوەبردنی هاتوچۆ بووەتە پرسێکی گرنگ کە پێویستە چارەسەر بکرێت. بۆ ئەم مەبەستە، ئاماژەدانی هاتوچۆی زیرەک بە ناو شارەکان خێرا پێویستە بۆ زاڵبوون بەسەر قەرەباڵغی هاتوچۆ، و کەمکردنەوەی تێچوون و کاتی گەشتکردن. بۆ زاڵبوون بەسەر ئەم کێشانەدا، بینینی کۆمپیوتەر (Computer Vision) و فێربوونی قووڵ (Deep Learning) بژاردەی گرنگن بۆ مامەڵەکردن لەگەڵ ئەم پرسەدا چونکە ڕۆڵێکی گرنگ دەگێڕن بۆ بەڕێوەبردن و کۆنترۆڵکردنی سیگناڵەکانی هاتوچۆ. سەرەڕای ئەوەش، دۆزینەوە و جیاکردنەوەی نێوان شتەکان(Objects) یارمەتیدەرە بۆ ژماردنی ئۆتۆمبێلەکان و شتەکانی تر کە خۆیان لە قەرەباڵغی و کۆنترۆڵکردنی سیگناڵەکان لە ناوچەکانی هاتوچۆدا بەدوور دەگرن. جگە لەوەش دۆزینەوەی ئۆتۆمبێلی فریاگوزاری و پێدانی ئەولەویەت بۆیان کە پێویستە بۆ سیستەمی زیرەکی ئاماژەدانی هاتوچۆ .

ئامانجی سەرەکی ئەم توێژینەوەیە دیزاینکردن و جێبەجێکردنی سیستەمێکی کارامەیە بۆ سیستەمی سیگناڵی هاتوچۆ لەسەر بنەمای دۆزینەوەی ئۆتۆمبێل لە ترافیکەکان. جگە لەوەش، سیستەمی پێشنیارکراوی چوار قۆناغ لەخۆدەگرێت؛ یەکەمیان گرتنی وێنە لە هەردوو کامێرای هاوشێوە و کاتی ڕاستەقینە لە ڕێگاکانەوە. لە قۆناغی دووەمدا ئەلگۆریتمەکانی جیاوازی پێش پرۆسێسکردنی وێنە بۆ وێنە گیراوەکان وەک هەنگاوێکی پێش پرۆسێسکردن ئەنجام دەدرێن. جگە لەوەش، تەکنیکەکانی فێربوونی قووڵ بەکاردەهێنرێن بۆ دیاریکردنی شتەکانی وەک ئۆتۆمبێلەکانی (ئاسایی، پۆلیس، فریاگوزاری، و ئاگرکوژێنەوە و هتد..) لەکاتێکدا لە قۆناغی کۆتاییدا، سیستەمی پێشنیارکراوی تاقیدەکرێتەوە بۆ هەڵسەنگاندنی وردی کارایی ئۆتۆمبێلە دۆزراوەکان .

ڕێبازێکی فێربوونی گواستنەوەی دەستکاریکراو بۆ مۆدێلی DenseNet201 بەکارهێنراوە بۆ چەندین پۆلێنکردن، لەوانەش ئۆتۆمبێلەکانی (ئاسایی، فریاگوزاری، ئەمبولانس، پۆلیس و ئاگرکوژێنەوە). ڕێبازەکە بریتییە لە بەستنی هەندێ لە چینەکانی مۆدێلەکە. ڕێژەی وردبینی بەرز بەم مۆدێلە بەدەست دێت و دەگاتە 98.6%. هەروەها، شێوازە جیاوازەکانی باشکردن، لەوانە (Adam, Adamax, Nadam, and RMSprob) بەکاردەهێنرێن بۆ باشترکردنی ئەدای دیاریکردن لەسەر بنەمای باشترین هەڵبژاردنی باشترکەر و وردبینی 98.84%ی بەدەستهێنا. سەرەڕای ئەوەش، وەشانێکی دەستکاریکراوی YOLOv5 پێشنیارکرا بۆ دۆزینەوەی ئۆتۆمبێل، کە ئامانجی بەرزکردنەوەی مامناوەندی وردبینی مامناوەندە (mAP) بە ڕێژەی 3%. لە کۆتاییدا، سیستەمی

پێشنیارکراوی هاوشێوە کرا بۆ کەمکردنەوەی کاتی چاوەڕوانی لە سیگناڵی هاتوچۆ. ئەنجامی تاقیکردنەوەکان کەمبوونەوەی بەرچاو لە کاتی چاوەڕوانیدا نیشان دەدەن، کە لە نێوان ٣٠ بۆ ١٠٠ چرکەدایە بەپێی دۆخەکان.