BSc Project
CS/CE/IS Department
Project ID: UQU-CS-2022F-09
Nov 2022

# JAZ
# Smart Traffic Light Network Using Computer Vision

**Sahar Ali Hakami**

**Shatha Badr Al-Hasani**

**Sara Mansour Al-Withinani**

**Nebras Abdullah Al-Shareef**

**Ghada Eidhah Al-Zahrani**

Dept. of Computer Science
Faculty of Computer and Information Systems
Umm Al-Qura University, KSA

# Contact Information

This project report is submitted to the Department of Computer Science at Umm Al-Qura University in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Engineering/Computer Science/Information Systems.

Author(s):

Sahar Ali Hakami sahar.ali.hakami@gmail.com

Shatha Badr Al-Hasani shatha.alsharef@gmail.com

Sara Mansor Al-Withinani sarahMalwithinani@gmail.com

Nebras Abdullah Al-Shareef NebrasAlsh@gmail.com

Ghada Eidhah Al-Zahrani Ghada.Eidhah@outlook.com

University supervisor(s):

Dr. Afnan M. Aldhahri

Dept. of Computer Science
Faculty of Computer and Information Systems

Umm Al Qura University
Kingdom of Saudi Arabia

Internet: http://uqu.edu.sa

## Intellectual Property Right Declaration

This is to declare that the work under the supervision of Dr. Afnan M. Aldhahri having title "JAZ

Smart Traffic Light Network Using Computer Vision" carried out in partial fulfillment of the requirements of Bachelor of Science in Computer Science, is the sole property of the Umm Al Qura University and the respective supervisor and is protected under the intellectual property right laws and conventions. It can only be considered/ used for purposes like extension for further enhancement, product development, adoption for commercial/organizational usage, etc., with the permission of the University and respective supervisor.

This above statement applies to all students and faculty members.

Date: ______________

Author(s):

| | |
|---|---|
| Sahar Ali Hakami | Signature: _____________________ |
| Shatha Badr Al-Hasani | Signature: _____________________ |
| Sara Mansor Al-Withinani | Signature: _____________________ |
| Nebras Abdullah Al-Shareef | Signature: _____________________ |
| Ghada Eidhah Al-Zahrani | Signature: _____________________ |

Supervisor(s):

Dr. Afnan M. Aldhahri

Signature: ____Afnan Aldhahri________

## Anti-Plagiarism Declaration

This is to declare that the above publication produced under the supervision of Dr. Afnan M. Aldhahri

having title “JAZ Smart Traffic Light Network Using Computer Vision” is the sole contribution of the author(s) and no part hereof has been reproduced illegally (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is proven.

Date: ______________

Author(s):

Author(s):

| | |
|---|---|
| Sahar Ali Hakami | Signature: _____________________ |
| Shatha Badr Al-Hasani | Signature: _____________________ |
| Sara Mansor Al-Withinani | Signature: _____________________ |
| Nebras Abdullah Al-Shareef | Signature: _____________________ |
| Ghada Eidhah Al-Zahrani | Signature: _____________________ |

## ACKNOWLEDGMENTS

While bringing out this project to its final form. We came across a number of people whose contributions in various ways helped our field of research and they deserve special thanks. It is a pleasure to convey our gratitude to all of them.

First, we wish to extend our greatest gratitude and sincere thanks to our supervisor Dr. Afnan Aldhahri for her invaluable guidance and advice throughout this project and who provided us with the golden opportunity to do this wonderful project on Smart Traffic light system which also helped us in doing a great deal of research and gave us a lot of new knowledge.

Our sincere thanks also go to Dr. Huda Ali Hakmai and Dr. Amira Al-harbi, for their encouragement, insightful comments, and their help in the mathematic and AI part of this project.

Last but not least, we would like to thank our friends and family for helping us to complete this semester and graduate project within the limited timeframe, it was a great comfort and relief to know that we all have our family and friends support unconditionally. Through it, we gained knowledge and skills that helped us in our careers.

## ABSTRACT

As we grew up in Saudi Arabia and grew up with technology, we became more enthusiastic to make the world smarter by utilizing new technologies to solve our daily problems.

Congestion is one of the most pressing issues in the transportation sector. According to statistics, 8 million trips are made every day in Riyadh city, with traffic on its streets lasting until late at night. Over the years, traffic has become increasingly worse in most cities around the world. And we don't seem to be able to solve the problem with the usual solutions!

We are developing a smart traffic light network for this project. Computer vision Mathematics are used to calculate the optimal green signal timing, update the red signal timing, and switch between them. Utilizing artificial intelligence will save people time and facilitate the development of a smart city.

Keywords: Traffic lights, smart traffic management, computer vision, transportation system

**TABLE OF CONTENT**

## LIST OF FIGURES

## LIST OF TABLES

# CHAPTER 1
# INTRODUCTION

# 1. INTRODUCITON

## 1.1. introduction

Nowadays, many influential countries are competing to provide innovative solutions for better smart cities, and Saudi Arabia is no exception. Improving the daily lives of people in Saudi Arabia by integrating artificial intelligence has become essential in developing the country's smart city. The transportation sector is receiving a lot of attention in this regard, with one of its main challenges worth tackling toward being improving traffic congestion In Riyadh city, 8 million trips take place each day, with traffic on its roads until late at night, according to recently released statistics by the General Directorate of Traffic [1]. Traffic management using AI is poised to reshape cities, relieving bottlenecks that snarl traffic. By reducing the time, we spend in traffic, we will not only reduce traffic congestion and travel times, but also emissions.

Traffic congestion refers to a condition in transportation that is characterized by slow speeds of transport progress, long travel times, and long queues of vehicles. In most cities around the world, traffic has been getting worse and worse over the years. The problem of traffic is a major one throughout the world without a doubt. Traffic congestion is the main obstacle for drivers in the United States, where they spend up to 58% more time stuck in traffic than they do in any other city in the world. Furthermore, such congestion leads to increased fuel consumption, air pollution, unneeded energy and time loss, chronic stress, and other physical health issues.

As the increase in vehicle traffic has not been accompanied by an increase in road capacity; this results in many congestion points which lead to a rise in accidents, affect economic growth, and increase greenhouse gas emissions. Traffic jams waste fuels, cause air pollution, and increase stress and frustration among motorists and passengers—in the United States alone, they cause 5.7 billion person-hours of delay each year.

The average driver in the United States wastes an average of 375 liters of gasoline a year due to traffic, and delays caused by intersections make up 12-55% of daily commute travel [1]. Based on the TomTom Traffic Index of 2021 [2], a slight increase in congestion levels was seen in Riyadh from 2020 to 2021, with congestion levels rising from 22% to 23%. When the congestion level is 23%, a 30-minute trip will take 7 minutes longer than in free-flow conditions.

The number one procedure for controlling traffic movement around the world is traffic light systems. Generally, traffic lights control traffic flow at crossroads, zebra crossings, and other points. Regulation of traffic at road junctions is intended to control vehicle movement in each movement group—traffic is expected to move more regularly.

Traffic lights traditionally operate sequentially based on a preprogrammed schedule, and they are effective on roads with constant vehicle density. However, vehicle density usually fluctuates in different directions. Therefore, to tackle the problem of traffic congestion more effectively, a smarter traffic light system is needed as one of the most cost-effective ways of keeping traffic moving smoothly and making streets safer.

The goal of smart traffic light management is to increase the efficiency of traffic signal control systems by utilizing improved tools, techniques, and equipment. It improves air quality, reduces fuel consumption, reduces congestion, and saves time for commercial and emergency vehicles, buses, and the public. It also reduces serious accidents as well as aggressive driving, including red-light running.

This project proposes an innovative solution to tackle the problem of traffic congestion by introducing a smart adaptive traffic light system. Through the Smart City initiative, cities will be able to improve their air quality, energy consumption, and energy efficiency. We can therefore say that traffic lights can be included in intelligent city solutions and can have a significant impact on reducing global warming and improving air quality. Through the implementation of this project and its use of artificial intelligence applications, we aim to promote the vision of 2030 and the concept of the smart city as NEOM.

## 1.2. Traffic light system

### *1.2.1. Traffic Light Purpose*

The purpose of traffic lights is to ensure the safety and efficiency of traffic. The failure of traffic lights can result in excessive delays, an increase in collisions, and harmful emissions as a result of improper installation, operation, or maintenance.

Our transportation system is directly impacted by signal timing, which impacts virtually everything in our communities. Our daily lives are influenced by the timing of our signals, from how much time we spend traveling to how safe it is to travel on roadways to how much money we spend on our trips.

### *1.2.2. Traffic Light System Mechanism*

- **Manual traffic light system**

Traffic lights were introduced in London for the first time in 1868. Their design was almost identical to railway signs, with two red and green arms used during the day to control traffic and gaslights used at night for visibility. Unlike an automatic or electronic light system, this “Manual Traffic Light Control System” required a policeman to stand beside it and operate it. As a result of the gas lighting, a policeman operating the light signal died on the night of 2nd June 1869. After this disaster, things eventually improved; an electric lighting system was developed, but it still required manual control [3]. At the present time, although the traffic light system has been changed to a fixed time system, but we notice in some places, such as the signals near universities, the officer controls the traffic lights manually, and the signal is changed according to the condition of the road, as the traffic increases in the morning times, especially before The timings of the lectures for students where the density of the street increases, so the officer opens the green signal for a longer period on each road to pass a greater number of vehicles.

- **Fixed time traffic light system**

  The fixed time or pre-timed controlling system is another mechanism used in traffic lights. In this system, the time that controls the signals is predetermined by a timer whose duration may range from 30 seconds to several minutes, depending on the intersection's historical traffic data. The main drawback of this mechanism is that it cannot adapt to different traffic situations. Several advantages can be gained from pre-timed control. The start and end of green are predictable, so it can be used for efficient coordination between adjacent pre-timed signals. Also, due to the fact that it is not dependent on detectors, it is immune to problems caused by malfunctioning detectors. In addition, setting it up and maintaining it requires very little training. Furthermore, pre-timed control is inefficient at isolated intersections where traffic arrives at random times since it cannot compensate for unplanned fluctuations in traffic flows.

- **Dynamic traffic light system**

  In a dynamic control system, meanwhile, the signals are adjusted dynamically and automatically in response to varying traffic conditions. Such systems are still in development and many researchers are still proposing methods to improve them for the purpose of minimizing the traffic congestion.

## 1.3. Project question

Our Project will be implemented to show “*How can we optimize the traffic light system and consider other lanes in the interaction using Computer Vision?”.*

## 1.4. What is JAZ

### 1.4.1. Topic area and title

We are building a Smart Traffic Light Network Using Computer Vision called JAZ that aims to enhance the transportation system and reduce traffic congestion.
JAZ is an Arabic name that means 'passing across' or 'making a transit' and is also used to signify 'being allowed', 'being permitted,' and 'saying Jaz'.

### 1.4.2. Idea generation and motivation

Smart cities are aimed at improving air quality, reducing energy consumption, and improving energy efficiency. As a result, we can say that traffic lights can contribute to reducing global warming and providing better air quality by being integrated into intelligent city solutions. As part of this project, artificial intelligence applications will be used to promote the vision of 2030 and the concept of the smart city as NEOM.

### 1.4.3. JAZ Description

JAZ is a smart traffic light system that aims to smooth traffic movement by improving the existing traffic lights system, making it into a more intelligent system by leveraging AI and computer vision. The key activity of the JAZ system is to intelligently change traffic light signals based on street conditions such as total vehicles and their type in the whole intersection at a given moment and the road condition in term of congestion, taking into consideration other lanes in the intersection.

Although the JAZ system will initially be developed and tested in Saudi Arabia, it is a universal solution that can serve all communities and help improve traffic around the world.

The JAZ system consists of two subsystems, the JAZ system and the JAZ admin system. The JAZ system uses computer vision and machine learning to determine the green light time for traffic lights. Three modules make up this system.

JAZ system modules:

- **Detection module**: In the detection module, computer vision techniques are used to count and detect vehicles based on real-time data collected from the street.

- **Scheduling module**: Calculate the green light time using a mathematical equation that generates the optimal green light time based on traffic density taking into account the entire intersection.

- **Simulation module**: the simulation module will be developed using Pygame library, a cross-platform set of Python modules designed for writing video games which is easy to use and runs in any environment. It provides music and graphics libraries created specifically for use with the Python programming language.

**JAZ admin system** is a web-based tool that provides admins with access to JAZ data collected from the detection module, such as how many vehicles are at a specific intersection on a given road. Maintenance and management processes can be assisted by this data.

### 1.4.4. JAZ value proposition

This system has several value propositions, including:

- **Reducing Travel time -** Reducing the average of 3 days and 7 hours wasted time per year in Saudi Arabia.
- **Less Pollution -** Help mitigate some of the emissions effects of starting and stopping and urban congestion.
- **Reducing Cost & Increasing Safety -** The system would be an alternative to traffic officers and that will reduce the cost and much safer for human being
- **Fewer Road Accidents -** Preventing road violation via a smarter traffic light system to minimize the temptation to bend the rules.
- **Real-Time data capturing**

## 1.5. Project management team-skills

This project will be a combination of applied computing, information technology, and business expertise that we have acquired over the last five years of our studies. We will be applying our skills to the implementation of a standard final product, which will include:

Project Management:

In order to manage the project effectively, we will use our past experience and knowledge of project management tools. Resources will be managed so that the project will be completed within the time, quality, cost, and scope constraints.

Database:

The project will gather traffic records or statistics as needed, and use a database. Both Oracle and Microsoft Access database systems have been used by us, and when required, we are able to query both of them using SQL.

System and Business Analysis:

Using business analysis techniques will help us to comprehend traffic light systems and adjust the development of the online tool.

Software Engineering Methodology:

We will use our software engineering skills to identify and apply software engineering methodologies to project development, such as analysis, specification, design, coding, and testing.

Web Development:

Online packages and the distribution requirements associated with them. As part of this project, you will need to be knowledgeable about web scripting, including PHP, Active Server Pages (ASP), hypertext markup language (HTML), and JavaScript.

Programming:

There will be some programming language skills necessary for the development of the website interface and backend connection of the website. Our knowledge and understanding of Flutter and Python have developed over time, and we are comfortable using them today. This project will allow us to study and learn more about the programming language.

Others:

An Object-Oriented Design approach is used to create a design using the Unified Modeling Language (UML).

# CHAPTER 2
# BACKGROUND

# 2. BACKGROUND

## 2.1. Traffic Engineering

Pedestrians and other vehicles such as motorcycles and buses are protected from accidents and injuries at busy intersections by traffic signals that ensure an orderly flow of traffic. Using the time sharing principle, we can resolve conflicts created by traffic movements in different directions. There are many reasons why traffic signals are beneficial, including orderly traffic movement and increased capacity at intersections. They also require only simple geometric design. Nevertheless, signalized intersections have disadvantages, such as large delays, difficulties in designing and implementing the system. It is possible for a user to experience relatively high stopped delays however, even though it may have a smaller overall delay than a rotary.

A lot of traffic lights are installed all around the world to ensure the safety, this amount of traffic light require a good and efficient signal timing and monitoring.

### 2.1.1. Terminologies

***Cycle***

A complete sequence of signal indications.

***Cycle Length***

the amount of time needed for a full run of signal indications.

***Interval***

It denotes a transition from one stage to another. There are two different sorts of intervals: change interval and clearance interval.

***Capacity***

the fastest possible speed for cars to cross the intersection under normal circumstances.

***Density***

The average number of vehicles on a stretch of road, commonly represented in terms of vehicles per mile or vehicles per mile per lane. (see also: volume-density, sometimes referred to as density timing)

***Delay***

the longer distance that a motorist, passenger, or pedestrian must travel.

***Effective green time***

The period of time that a specific traffic movement or series of movements may occur; it is determined by subtracting the effective red time from the cycle duration.

***Lost Time***

the period of time that vehicles cannot use at the start of each green period, as well as a portion of each yellow change and red clearance period.

***Travel Time (Average)***

the total amount of time taken to travel a particular distance. A single link or corridor's average trip time is an average of all runs for that link or corridor.

***Queue***

A line of vehicles, bicycles, or persons waiting to be served by the system in which the flow rate from the front of the queue determines the average speed within the queue. Usually, those joining the back of the line or slowly moving vehicles are seen as being in the line.

**Quick-Estimation Method**

A method that allows an analyst to identify the critical movements at an intersection, estimate whether the intersection is operating below, near, at, or over capacity, and approximate the amount required for each crucial maneuver in green time.

**Saturation Flow (s)**

The maximum quantity of vehicle flow that is feasible is called saturation flow. It's described as the opposite of headway.

The saturation flow is given as Saturation flow = 3600/headway in vehicles per hour if the headway is given in seconds.

**Observed Volume (v)**

The observed volume is the amount of traffic flow that is actually being seen at the intersection.. It is also represented as vehicles per unit time.

**Critical Flow Ratio**

The ratio between the observed volume of flow and the saturation flow present at all phases of an intersection is known as the critical flow ratio at a phase. It's stated as,

***Critical flow ratio at ith phase = observed volume / saturation flow = v/s at ith phase***

## 2.2. Webster’s method of traffic signal design

No matter how much or how little traffic is present on the route, fixed-time signals operate in the same sequence, always providing the same amount of time to each traffic movement. The delaying effect of fixed time signals can be caused by short-term demand fluctuations.

Traffic signals are designed rationally using Webster's method. The formulae given by Webster make it simple to apply. An intersection's optimum cycle length can be calculated using Webster's formula [4]. The optimum cycle length is also taken as the total cycle time for a signal system. Webster's formula is given as,

***Optimum cycle length*** (Co) = (1.5*L + 5) / (1 - y),

where,

L - entire red time plus all lost time,

L = (n * Lost time at a phase) + All red time

n - number of phases

Typically, all red time is regarded as zero.

A phase's lost time is often calculated as 2 seconds

y -, where y is the total of each phase's critical flow ratio

***Green Time by Webster Method***

By using Webster's approach, the green time for road "a" is given as,

Ga = (ya/y) * (Co - L),

where,

ya represents the critical flow ratio for road "a,"

y is the sum of all critical flow ratios

Co - Optimum cycle length

L - lost time including all red time

## 2.3. History of AI

Artificial intelligence (AI) consists of a set of sciences, theories, and techniques (including mathematical logic, statistics, probability, computational neurobiology, computer science) that aims to duplicate human cognitive abilities. As a result of its development in the 1940s, computers are capable of performing increasingly complex tasks, which could once only be managed by humans.

The concept of artificial intelligence was not invented by John Von Neumann and Alan Turing at the beginning of 1950, but they were the fathers of the technology that led to it: they transitioned from computerized logic to decimal logic in the 19th century, which dealt with values ranging from 0 to 9 and machines to binary logic that relies on Boolean algebra, which deals with chains of zeros or ones. As described in Turing's famous 1950 article "Computing Machinery and Intelligence", the possibility that a machine is intelligent was raised for the first time. He proposed a "game of imitation," in which a person could tell whether he was conversing with a man or a machine in a teletype dialogue.

In 1956, the Rockefeller Institute funded a conference at Dartmouth College that sparked the development of the field of artificial intelligence. [5]

## 2.4. AI in Saudi Arabia

Machines, especially computers, simulate human intelligence processes through artificial intelligence. Machine vision, natural language processing, speech recognition and expert systems are examples of specific applications of artificial intelligence.

future states, AI systems analyze huge amounts of labeled training data, determining correlations and patterns, and applying those patterns to the training data. This is how social robots, such as chatbots fed with examples of text exchanges, can learn to make lifelike interactions with people, or how image recognition tools can figure out what objects are in images by reviewing millions of them. It reflects a deeper understanding of the world around it that is reflected in intelligence. Nevertheless, all its components must work together in order for the system to qualify as AI.

In 2016, the Saudi Arabian government developed the 2030 Vision, a strategic framework that gave AI a central role in creating and integrating the city of Neom. In order to free the kingdom from its dependence on oil exports, Vision2030 reforms will focus on economic and social policies.

With the organization of a global AI summit in 2020, Saudi Arabia plans to become the world's leading AI event provider. This event brings together leaders, experts, and specialists from Saudi Arabia and around the world, including technology companies, investors, and entrepreneurs. [6]

Among the six dimensions of Saudi Arabia's National Strategy for Data and Artificial Intelligence are:

- **Ambition** – Develop and adopt data and AI technologies, and lead global dialogue and policy development, and to transform the country into a global leader in data and AI. 2030 is the target date for Saudi Arabia to rank among the top 15 countries in the field of artificial intelligence.

- **Skills** – Assist Saudi Arabians in leveraging the power of data and AI to improve their public and private sector performance through education and re-skilling.

- **Policy and regulations** – Inspire and foster data-driven businesses, open data sharing, and data collection by government agencies for the benefit of citizens through a world-class regulatory framework.

- **Investment** – Promote investment and incentivize foreign and local investors to invest in qualified Saudi Arabian opportunities.

- **Research and innovation** – To accelerate the development and commercialization of new technologies, the kingdom must build and enable core research and innovation institutions in data and artificial intelligence.

- **Ecosystem** – Build a collaborative ecosystem to promote data and AI adoption and commercialization for both public and private sectors, through the creation of a forward-looking ecosystem.

## 2.5. Machine and deep learning

**Machine learning** focuses on setting up computers to be able to conduct tasks in a non-programmed way. Structured data is fed to computers (in most cases) and they 'learn' over time how to evaluate and act upon it. Supervised, unsupervised, and reinforcement learning are three types of machine learning. In machine learning, supervised learning is one of the most basic forms. It uses labeled data to train the machine learning algorithm. Data that is unlabeled can be processed by unsupervised machine learning. In other words, the program can work on much larger datasets without requiring human labor to make them machine-readable. As humans learn from data in their everyday lives, reinforcement learning directly draws inspiration from this process. A trial-and-error method is used to improve the algorithm over time and learn from new situations [7].

An algorithm that uses **deep learning** is an algorithm based on neural networks with three or more layers. It is a subset of machine learning. By simulating the human brain's behavior - although far from its capabilities - neural networks are able to "learn" from large amounts of data. With additional hidden layers, a neural network can improve and refine its accuracy while still making approximate predictions [8].

It is common to refer to deep learning models as deep neural networks because they typically use neural network architectures. A neural network is typically described as "deep" if it has many hidden layers. As many as 150 hidden layers are possible in deep neural networks, which are more complex than traditional neural networks.

An algorithm that learns features from large sets of labeled data can be trained using large sets of labeled data and neural networks that don't require manual feature extraction. For deep learning to be successful, raw data must be automatically transformed into such representations.

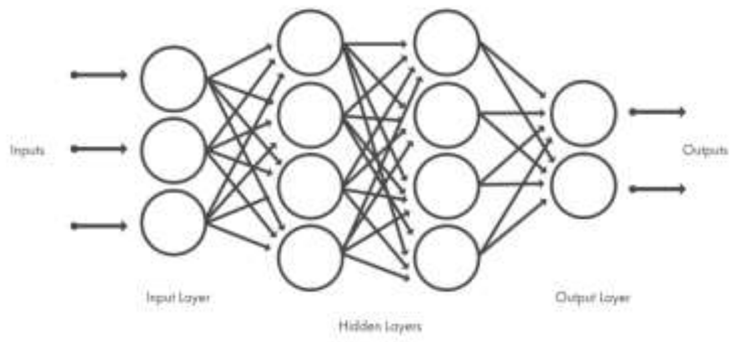


***Figure 1:*** *CNN Layers*

Convolutional neural networks (CNNs or ConvNets) are one type of deep neural network. Convolutional neural networks (CNNs) combine learned features with input data using two-dimensional convolutional layers, and are thus well-suited to processing 2D data.

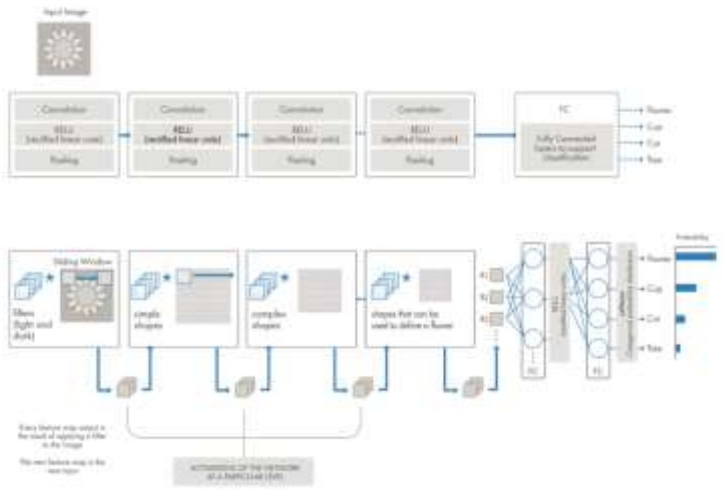


***Figure 2:*** *CNN Structure*

You don't have to identify the features used to classify images with CNNs since they eliminate the need for manual feature extraction. CNN works by extracting features directly from images. During training, the network learns relevant features from a collection of images and not from pre-trained features. For computer vision tasks, such as object classification, deep learning models can be highly accurate due to automated feature extraction.

A CNN uses tens or hundreds of hidden layers to detect different features of an image. As hidden layers are added to an image, the features become more complex. Detecting edges could be the first hidden layer and detecting complex shapes specific to the object shape could be the second hidden layer [9].

As inputs are taken into CNNs, they are processed, and the results are sent out as outputs. The image is fed as input. Arrays of image pixels are accepted as input by the input layer. In CNNs, there could be multiple hidden layers, which perform feature extraction from the image by doing calculations. the process could involve convolution, pooling, rectified linear units, and fully connected layers. Convolution is the primary layer that does highlight extraction from an input image. The fully connected layer classifies the object and distinguishes it within the yield layer. "CNNs are feedforward systems in that data stream takes put in one heading as it were, from their inputs to their yields. Fair as manufactured neural systems (ANN) are biologically inspired, so are CNNs. The visual cortex within the brain, which comprises of rotating layers of straightforward and complex cells, spurs their engineering. CNN structures come in a few varieties; in any case, in common, they comprise of convolutional and pooling (or subsampling) layers, which are gathered into modules. Either one or more completely associated layers, as in a standard feedforward neural network, take after these modules [10].

## 2.6. Image Processing

Image processing refers to the process of adding some improvements to an image or extracting useful information from it. It is a type of signal processing that includes an image as input and an image or features associated with that image as output. Image processing today is one of the fastest growing technologies. It is also one of the main research areas in engineering and computer science.

Image processing mainly involves the following three steps:

- Import the image via the image acquisition tools.
- Image analysis and processing.
- The output through which the image can be changed, or the report based on the image analysis.

There are two types of methods used for image processing, namely, analog and digital image processing. Analog image processing can be used for prints such as prints and photographs. Image analysts use different fundamentals of interpretation while using these visual techniques. Digital image processing techniques help in processing digital images using computers. The three general stages that all types of data must go through while using digital technologies are pre-processing, augmentation, presentation, and information extraction.

In this project we will use digital image processing to extract useful traffic data that will be used as an output for the scheduling algorithm.

## 2.7. Computer Vision

Artificial intelligence in the form of computer vision trains machines to understand and interpret visual data. Machines can accurately identify and classify objects with the help of digital images captured by cameras and videos and deep learning models.

CV applications:

- It is extremely time-consuming to evaluate cancerous tumors using traditional methods. Such methods are subject to subjectivity and can be subject to errors based on a limited amount of data. By leveraging artificial intelligence, Amsterdam UMC has transformed tumor evaluations. To identify cancer patients who are candidates for surgery faster, and with more precision and accuracy, doctors use deep learning models and computer vision to assess chemotherapy responses [11].

- By using Computer Vision in agriculture, plant growth and disease detection can be monitored continuously in real-time.

Computer vision employments image processing to recognize and categorize picture information. Computer vision and image processing are two closely related areas which can be considered as a work zone utilized in nearly any investigate including cameras or any picture sensor to obtain data from the scenes or working situations. Imperative data can be gotten from pictures captured from the camera at the road, there are a lot of methods utilized in computer vision to extricate these information.

### 2.7.1. Object Recognition

Computer vision is a technique used to recognize objects in images or videos. Machine learning and deep learning algorithms produce object recognition as a key output. We are able to recognize people, objects, scenes, and visual details when we look at a photograph or a video. The objective is to teach a computer to do what comes normally to people: to pick up a level of understanding of what a picture contains. Object recognition could be a key innovation behind driverless cars, empowering them to recognize a halt sign or to distinguish a person on foot from a lamppost. It is additionally valuable in an assortment of applications such as malady recognizable proof in bioimaging, mechanical review, and automated vision.

**Object Recognition in machine learning:**

In machine learning, the process of identifying the object in an image requires a set of data from the classes, followed by manually extracting their features. The object in the image is then classified using a classifier model. Machine learning for object recognition gives you the freedom to select the ideal blend of learning features and classifiers. It can produce precise results with little data.

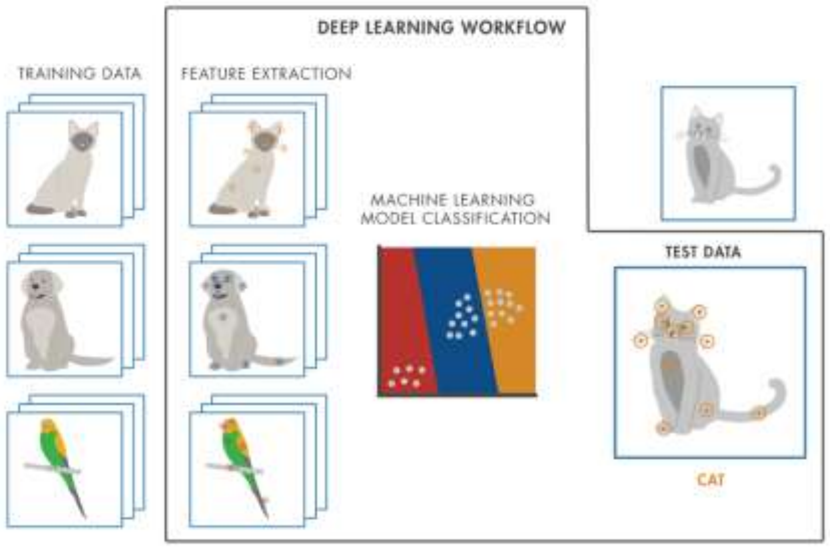


***Figure 3:*** *Deep Learning Workflow*

**Object recognition in deep learning:**

Convolutional neural networks, or CNNs, are deep learning models that are used to automatically learn an object's inborn properties in order to identify that object. Although deep learning is highly accurate, it needs a lot of data to generate reliable predictions.

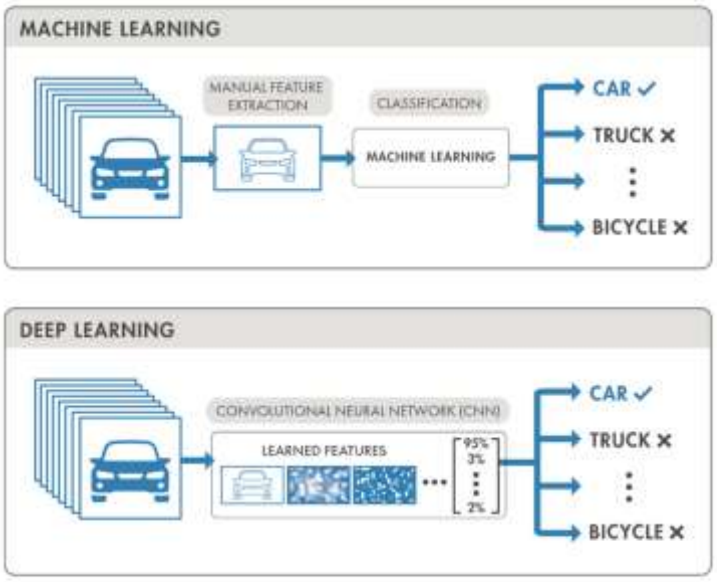


***Figure 4:*** *Machine Learning vs Deep Learning*

**Object recognition use cases:**

**Automotive industry:** Using this technology, self-driving cars will continue to be able to recognize the objects around them and make decisions that prioritize safety first.

**Security and surveillance:** Private businesses utilize smart cameras with incorporated object, face, and retina recognition technologies for public safety.

**Agriculture:** Seedlings and weeds can be distinguished by robotic instruments equipped with computer vision.

**Retail:** Customers can be recognized and their behavior in a store can be tracked using smart cameras with a visual object recognition feature. Better product and shelf placements are made using these findings.

### 2.7.2. Image Classification

It is the process of categorizing and classifying the image's elements. For instance, if you use Google to search for dog pictures, the network will return hundreds of images, including photographs, illustrations, and even sketches. The neural network now needs to scan several photos containing various items, detect them, and categorize according to the type of the item on the picture. This is an advanced version of image detection. Pattern matching with data is what classification is.

Using training data is not necessary for the fully automated unsupervised classification method. During the image processing stage, the specified qualities of an image are systematically detected using a suitable algorithm. "Image clustering" or "pattern recognition" are the classification techniques employed in this scenario. "ISODATA" and "K-mean" are two often used algorithms.

In order to generate statistical measures that can be applied to the complete image, the supervised classification approach involves visually picking samples (training data) within the image and categorizing them (e.g., roads, buildings, water bodies, vegetation, etc.). Two popular techniques for categorizing the full image using the training data are "maximum likelihood" and "minimum distance." For instance, "maximum likelihood" classification computes the mean and standard deviation of each spectral and textural index of the image before using the statistical properties of the data. The likelihood of each pixel belonging to a certain class is then calculated using some classical statistics and probabilistic relationships, taking into account a normal distribution for the pixels in each class. Finally, the pixels are labeled to a class of features that show the highest likelihood [12].

A web-based application called Teachable Machine allows building machine learning models quick, simple, and available to everyone. And we made use of it to demonstrate how to identify and categorize objects in an image [13]. This model is going to recognize and classify the type of plant in an image, for a demonstration we created 5 plant and tree classes (bamboo, botus, lemon, orange, zamia), with 50 images in each class and train this model with 60 epochs and 32 batch size.

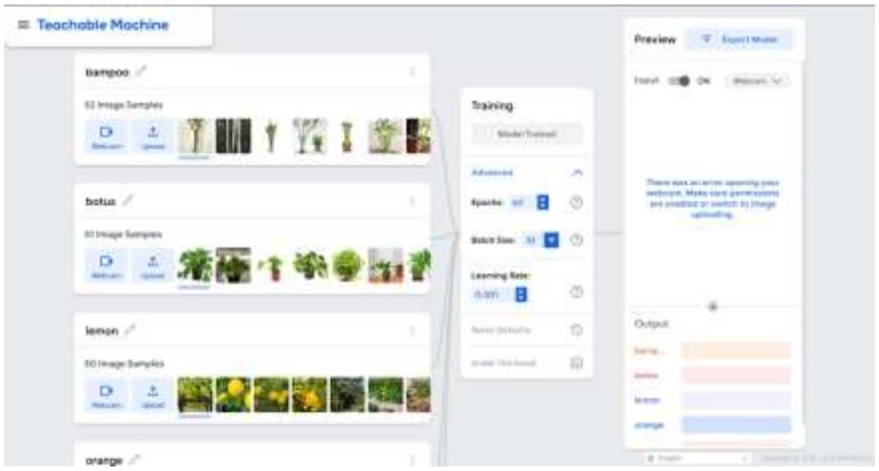


*Figure 5:* *Teachable Machine Project*

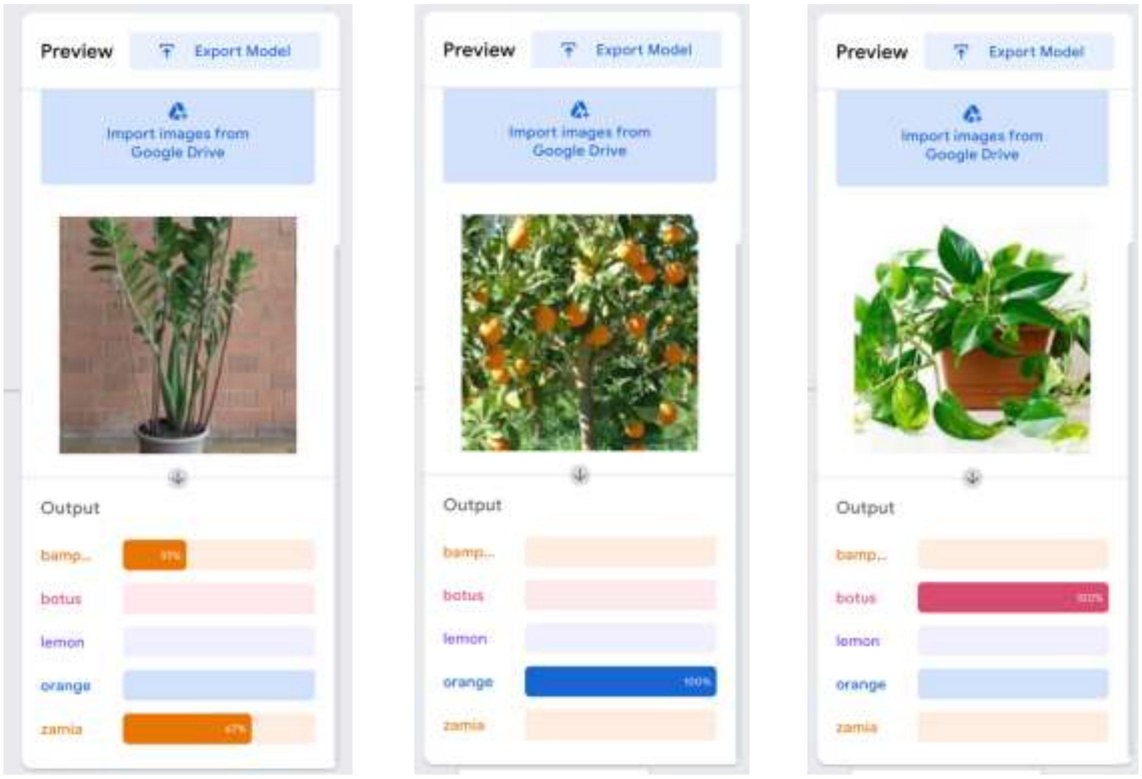


*Figure 6:* *Results Of Teachable Machine Training*

### 2.7.3. Object Localization

In order to build a bounding box around an object of interest, the object localization method predicts a set of 4 continuous numbers, namely the x coordinate, y coordinate, height, and breadth. The final layer of classification algorithms provides a probability value between 0 and 1. In contrast, because localization is a regression problem, localization methods produce results in four real numbers [14].

***Figure 7:*** *Object Localization*

### 2.7.4. Object Detection

A computer vision technique called object detection is used to find objects in pictures or movies.. Object detection algorithms commonly employ machine learning or deep learning to get relevant results. When watching photographs or movies, humans are able to quickly recognize and pin down objects of interest. Computer-based object detection makes an attempt to imitate this intelligence.

To get started with object detection using deep learning, you can pick between two major strategies:

- Make a unique item detector and train it. You must create a network architecture to learn the features for the items of interest if you want to train a custom object detector from scratch. Additionally, you need to collect a substantial amount of labeled data in order to train the CNN. A specialized object detector can deliver incredible outcomes.. Nevertheless, you must manually configure the CNN's layers and weights, which takes a lot of time and training data.

- Implement a trained object detector. Transfer learning is a technique used by many deep learning object identification procedures that enables you to start with a pretrained network and then fine-tune it for your application. Because the object detectors have already been trained on dozens or even millions of photos, this method can produce findings more quickly.

**Two-Stage Networks**

A region proposal, or a subset of the image that may include an item, is identified in the first stage of two-stage networks like R-CNN and its variations. The objects contained in the region proposals are categorized in the second step. Although two-stage networks are often slower than single-stage networks, they can produce results for object detection that are quite accurate.

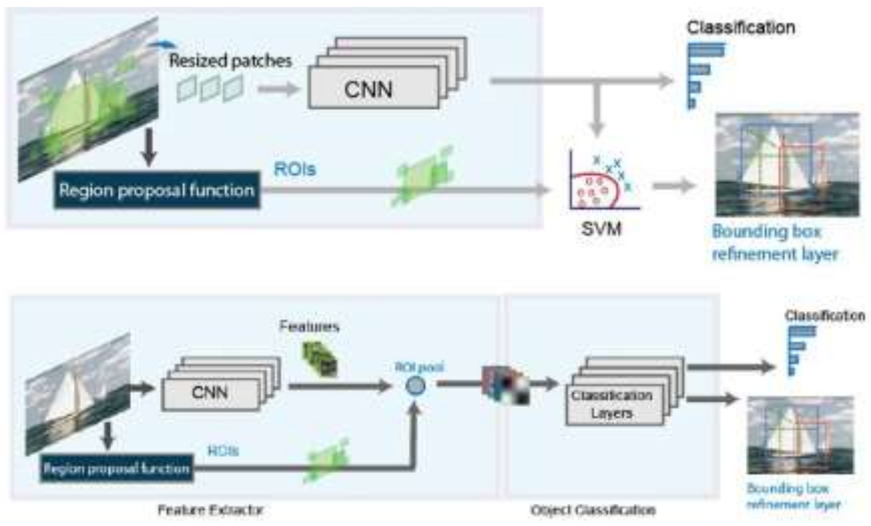


***Figure 8:*** *High-level architecture of R-CNN (top) and Fast R-CNN (bottom) object detection*

**Single-Stage Networks**

In single-stage networks, like YOLO, the CNN uses anchor boxes to construct network predictions for regions over the entire image, and the predictions are then decoded to provide the final bounding boxes for the objects. Despite being substantially faster than two-stage networks, single-stage networks may not achieve the same level of accuracy, particularly in scenarios with little objects [15].

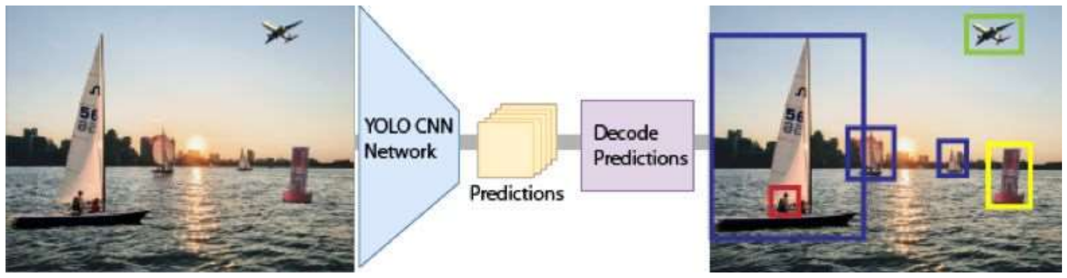


***Figure 9:*** *Overview of YOLO v2 object detection*

# CHAPTER 3
# RELATED WORK

# 3. Related work

## 3.1. Software based

The authors in [16] propose a better system called the Traffic Lights Expert System (TLES). TLES uses rule-based knowledge representation, with evidential reasoning as the inference engine. The main drawback of this work are the limitations of TLES, such as difficult knowledge acquisition, maintenance costs and development costs.

In this paper [17], the authors propose a history-based traffic management algorithm that relies on the previous all-year traffic information to predict the traffic flow on congested streets of a crowded city. Thus, they refer to this proposed 19 techniques as a calendar-based traffic congestion management system. The main idea is to use the recorded traffic history information to compute the green/red times for each direction on a congested intersection with a traffic light controller. A robust heuristic is proposed to use the history information in predicting the future traffic load on each street leading to an intersection controlled by a traffic light

Smart Traffic Light Scheduling in Smart City Using Image and Video Processing [18] is a system using the combination of IoT and image and video processing techniques. They implemented two models for the scheduling, the first model takes into account the density of the vehicle and the second model is based on the number of vehicles. The images are processed in Raspberry Pi by separating the edge from the original images, white pixels are counted after displaying the edge image as white color and creating the image black background the overlapping percentage of the instantaneous traffic image with the reference image is obtained. Then, information is given to the scheduling algorithm to decide dynamically on the timing of the green light of each direction but without taking into account other directions in the intersection. The second model uses the live video for video processing, so that the number of vehicles passing through the main streets leading to the crossroads is determined. Therefore, the location of the camera to determine the number of vehicles is far from the crossroad and on the path leading to the crossroad and video processing is performed using the video background removal method.

The authors in [19] propose a self-adaptive approach to building a smart traffic light management system for dealing with intersections. This system relies on multi-agent systems architecture and can support a distributed and collaborative mechanism of regulation while considering dynamic changes in the traffic flow. The main limitation of this paper is that it doesn’t take the type of vehicles and the other lanes into consideration when determining the green light time.

In [20], the proposed system utilizes live images from the cameras at traffic junctions for traffic density calculation using image processing and AI. It also focuses on the algorithm for switching the traffic lights based on vehicle density in order to reduce congestion. The main limitation of this system is that how long it gives a green light to one lane is not dependent on the traffic in other lanes.

## 3.2. Hardware based

The authors in this paper [21] proposed a development of a system to handle traffic in a smart way by automatically adjusting its timing based on traffic density using Arduino Uno ATMega 328. In this, traffic is sensed using digital IR Sensors and IR Sensors detect vehicles further based on the signal reflected from them. Sensors placed adjacent to the road to control the traffic density by changing traffic signals appropriately. All IR Sensors are interfaced with Arduino Uno, and it reads data from IR Sensors. Traffic Signal for the system is designed using LEDs and each signal consists of two LEDs for each lane. Using this system development at traffic junctions we need not to worry about handling the traffic manually and also consumes less time as compared to the conventional traffic system. We harness solar power from solar panels, and this is used to build a prototype working model of a smart traffic signal which automatically adjusts its timing based on traffic direction.

The proposed smart traffic light system implemented using Arduino, camera and IR sensors [22], this system implementation is based on sensing the current number of vehicles at each side of an intersection and assigning the green light when the number vehicles reach a specific number. The sensors are placed in all the four roads, the sensors are installed at 30-meter distance from the intersection at each side. When a road reaches for example 30 cars the green light will be triggered and will assign the green light to this road, when another road reaches the 30 cars, red light will be assigned to the current signal and allocate the green light to the new 30 cars stacked road. If two or more roads have reached the 30 cars stacked, the system will store the road ID in a queue and then follow the queue priority scheme First In First Out (FIFO) with 3 minute time to each road.

The paper [23] explores the utilization of RFID technology for preventing traffic congestion. It suggests finding the blockage at any intersection of the street with an RFID reader and labels as sensors. The main limitation of this system is that it requires every vehicle to be prepared with a passive RFID tag.

| RELATED RANK | PAPERS TITLE | PAPER DESCRIPTION | TECHNIQUES |
|---|---|---|---|
| **Software methods** | | | |
| 1 | Smart control of traffic lights using artificial intelligence | Utilizes live images from the cameras at traffic junctions for traffic density calculation using image processing and AI. It also focuses on the algorithm for switching traffic lights in order to reduce congestion. | Image Processing + switching algorithm |
| 2 | Smart Traffic Light Scheduling in Smart City Using Image and Video Processing | Used the combination of IoT and image and video processing techniques and implemented two models for the scheduling, the first model takes into account the density of the vehicle and the second model is based on the number of vehicles. | Image and Video Processing + IOT using Raspberry Pi |
| 3 | Adaptive and collaborative agent-based traffic regulation using behavior trees. | This system relies on multi-agent systems architecture and can support a distributed and collaborative mechanism of regulation while considering dynamic changes in the traffic flow. | Agent-based system + perception using behavior tree |
| 4 | intelligent traffic light scheduling technique using calendar-based history information | A history-based traffic management algorithm that relies on the previous all-year traffic information to predict the traffic flow on congested streets of a crowded city. Thus, they refer to this proposed 19 techniques as a calendar-based traffic congestion management system. | history-based system + perception |
| 5 | Modeling and Controlling Smart Traffic Light System Using a Rule Based System | This paper proposes a better system called the Traffic Lights Expert System (TLES). TLES uses rule-based knowledge representation, with evidential reasoning as the inference engine. | rule-based knowledge representation |

| Hardware methods | | | |
|---|---|---|---|
| 6 | Smart density-based traffic light system. | Adjusting the traffic light time based on traffic density using Arduino Uno ATMega 328. In this, traffic is sensed using digital IR Sensors and IR Sensors detect vehicles further based on the signal reflected from them | timing based of traffic density + IR sensor, Arduino |
| 7 | Design and implementation of a smart traffic light management system controlled wirelessly by Arduino | This system implementation is based on sensing the current number of vehicles at each side of an intersection and assigning the green light when the number vehicles reach a specific number | IR sensor, wireless system by Arduino |
| 8 | An adaptive approach: Smart traffic congestion control system | The paper explores the utilization of RFID technology for preventing traffic congestion. It suggests finding the blockage at any intersection of the street with an RFID reader and labels as sensors. | Radio-frequency identification + IOT |

***Table 1:*** *Related work*

# CHAPTER 4

# SYSTEM ANALYSIS AND REQUIRMENTS SPECIFICATION

# 4. System Analysis and Requirements Specifications

This chapter dives deeper into the details of the proposed system where the system requirements are specified by the functional and the non-functional requirements and multiple perspectives of the system are modeled through different diagrams. Finally, both proposed and future solutions to the system are shown.

## 4.1. Project management methodology

**Development Method**

In this project, we will be using the waterfall methodology to develop JAZ. This is because of the nature of the graduation project process, which is a well-defined process in which cost, design, and time requirements need to be known upfront. In the primary waterfall process, five phases are involved

**Requirement Analysis Phase**

In this phase, we collected the dataset that will be used in the training of the vehicle detection module from public computer vision cars on the Kaggle website, Furthermore, the JAZ system is clearly defined, and the details of the functionality have been specified and documented both functional and non-functional requirements to specify how the system should be able to work and perform. After the system was identified, the UML and DFD diagrams were used to show and illustrate the system structure and behaviour.

**Design Phase**

Once all of the system requirements have been identified in the analysis phase, we will define and designs the architecture of the overall system as well as the hardware and software requirements. As part of this phase, we will also plan project management strategies and future enhancements as well as define the system components in detail.

**Implementation Phase**

In the implementation phase, three modules will be developed. The detection module will involve training a deep learning model called YOLOv5 (You Only Look Once) in Real-Time Object Detection to detect the vehicles. The main benefit of adopting YOLO is its incredible speed; it can process 45 frames per second.. YOLO also understands generalized object representation [24]. We will use PyTorch, an open-source machine learning framework that accelerates the path from research prototyping to production deployment. PyTorch is one of the top deep learning tools; its use is increasing in both industry and research as it is flexible and fast and makes it easy to get a project up and running. The second module involves a signal-switching algorithm that is used to determine the best time for a signal based on a mathematical equation.

In addition to the JAZ system, we will develop the admin system. We will begin by creating the user interface (UI) using Flutter, a Google open-source framework for creating stunning, natively compiling, cross-platform applications from a single code source, run by Dart, a programming language designed for quick programs on any platform, as well as Firebase Database, a cloud-hosted NoSQL database that supports real-time data syncing across Android and iOS clients, build rich, collaborative apps by allowing direct, safe access to the database from client-side code. and store all the data in JSON groups, and any changes to the data are instantly reflected by performing a sync activity through all the stages

**Testing Phase**

In this phase, the system will be tested on a simulation to test the equation and show its result. The simulation module will be developed using Pygame library, a cross-platform set of Python modules designed for writing video games that is easy to use and runs in any environment It provides music and graphics libraries created specifically for use with the Python programming language.

**Maintenance Phase**

The system will be checked and observed for regular maintenance to make sure it functions smoothly along the way. Maintaining traffic signal timing schedules and the traffic signal system requires good precise data management. When records are not kept properly, efficiency and cost savings will be lost in the future if the process must be recreated or followed. We are going to organize and manage the collected data, which is needed for tracking changes in the controller, or other signal infrastructure, as well as for documenting maintenance measures and that will result in a lot of benefits in the future. Once the project is delivered, we intend to publish it as an open source for future researchers so we can improve the functionality and maximize the benefit from the project to make progress towards the 2030 vision.

## 4.2. JAZ over all architecture

**Algorithm to be used**

Algorithm for switching the signal - As the vehicle detection module returns traffic density information, the signal switching algorithm adjusts the green light timer. It also updates the red-light timers. According to the timer, the signals are also cycled through.
In Detection module, by using deep learning model called YOLOv5 to count and detect vehicles based on real time data.

Detection module information is used as input to the equation, based on this input, an analysis is conducted of each vehicle class as a whole to determine the number of vehicles. Calculate the green light time from a mathematical equation that generates the optimal green light time based on traffic density. After the green time has been calculated for the signal, it is associated with all the other signals, and the red times are automatically adjusted. During the development of the equation, the following factors were considered: In order to calculate traffic density, an equation has to calculate how long it will take for a photo to be taken. Based on this, the number of lanes, number of cars, and their type as trucks, buses, etc., is determined. based on the factors listed above, traffic density can be calculated.

**Reuse of existing software**

We have discussed the design constraints, including the hardware and software environments, we will describe the implementation and software components used in our system. In one model, a mathematical equation is used to determine the optimal time for a signal. For detecting and counting vehicles, Yolo is used in the detection module. An algorithm that employs two-stage object detection breaks the object detection problem statement into two phases: 1. Regions of potential objects are detected. 2. Sorting the image into object classes based on those regions. With this approach, accurate object detection with high mean average precision is achieved. However, this method requires multiple iterations, which slows down detection speed and prevents real-time detection. As a result of YOLO's end-to-end neural network, bounding boxes and class probability predictions are made all at once. As compared to other algorithms, the "You Only Look Once" (YOLO) algorithm provides fast performance for this purpose. This algorithm runs at speeds as high as 45 frames per second, which is significantly faster than its counterparts. 43

## 4.3. Project management plan

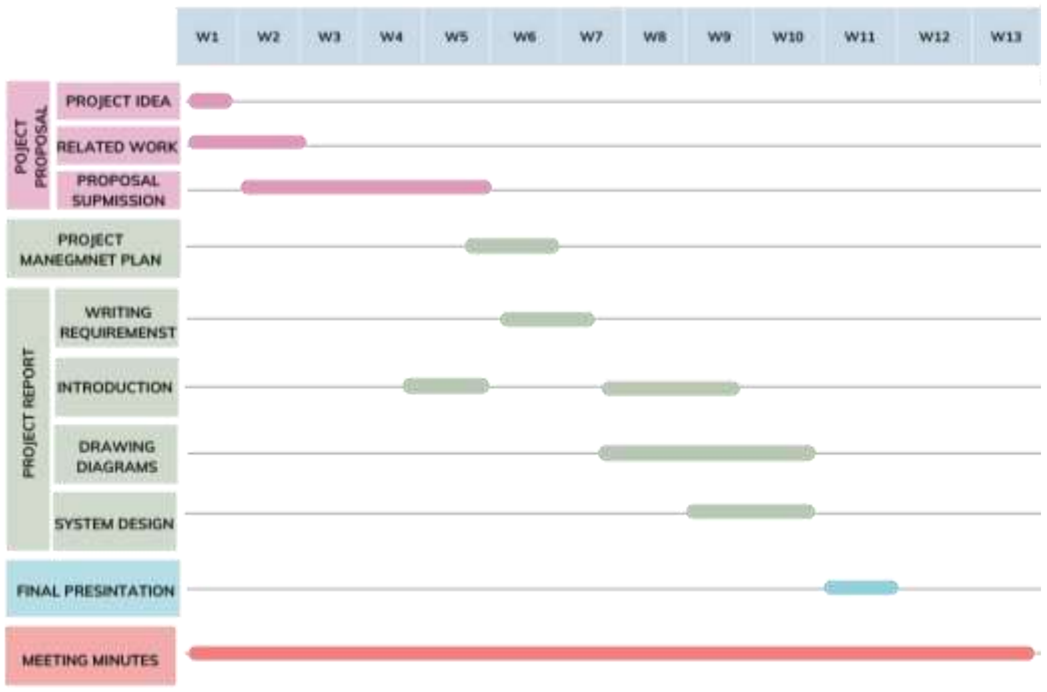
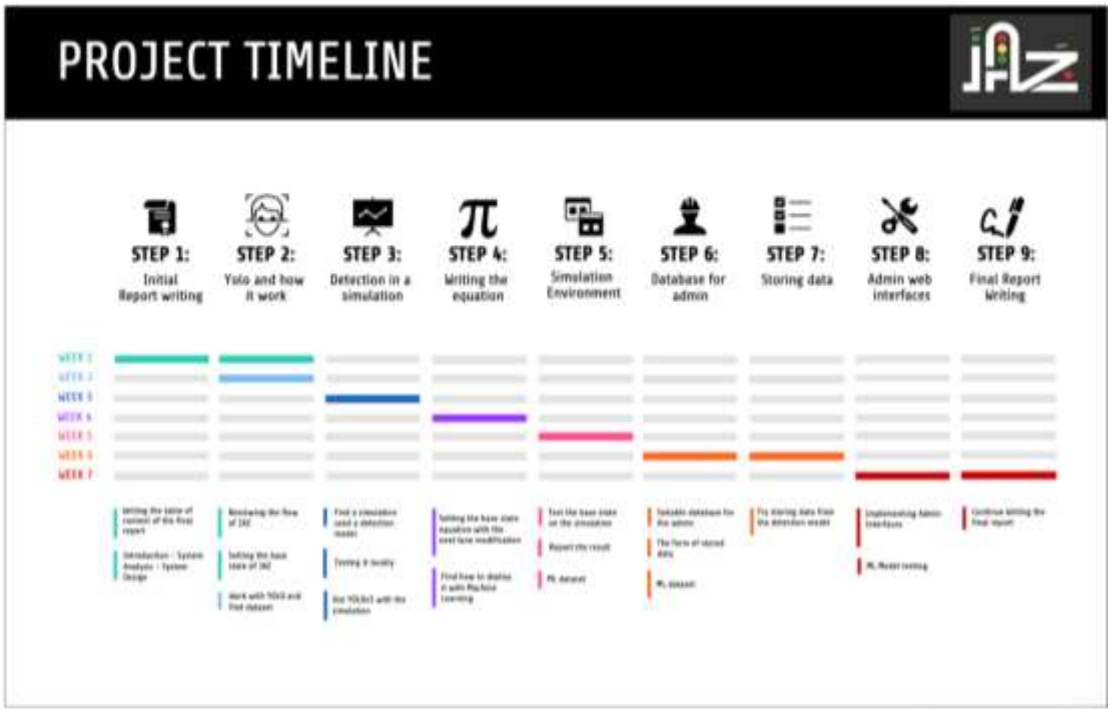


***Figure 10:*** *Project Management Plan*

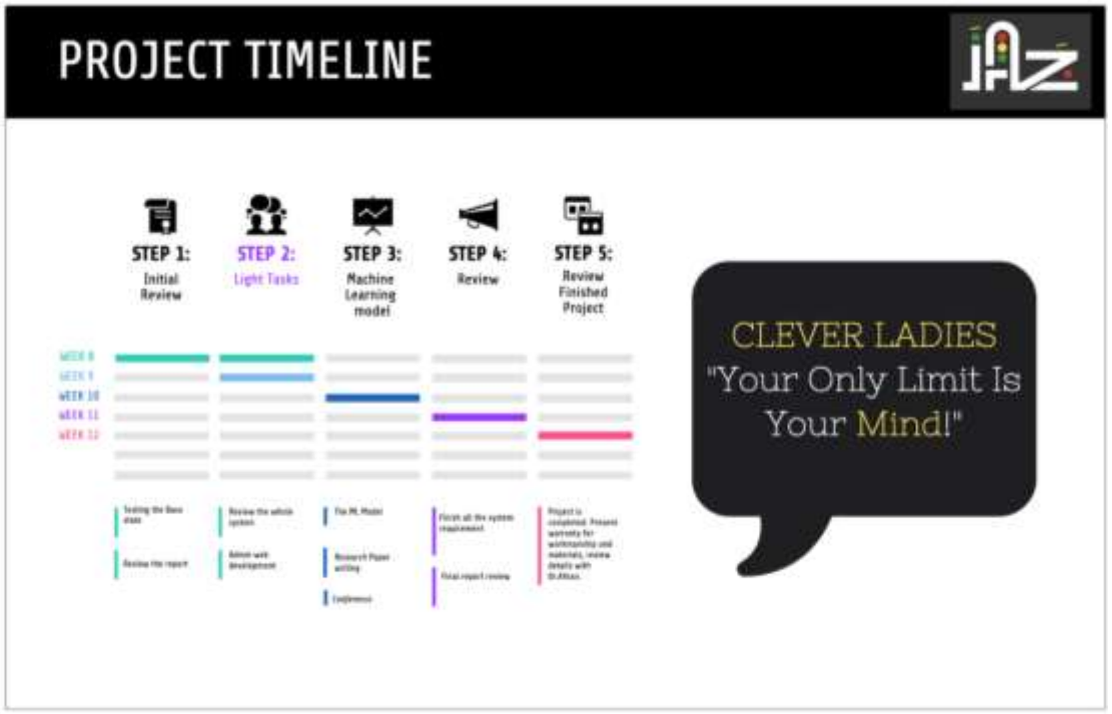


## PROJECT TIMELINE

STEP 1: Initial Report writing
STEP 2: Yolo and how it work
STEP 3: Detection in a simulation
STEP 4: Writing the equation
STEP 5: Simulation Environment
STEP 6: Database for admin
STEP 7: Storing data
STEP 8: Admin web interfaces
STEP 9: Final Report Writing

WEEK 1
WEEK 2
WEEK 3
WEEK 4
WEEK 5
WEEK 6
WEEK 7

- Writing the table of content of the final report
- Introduction - System Analysis - System Design
- Reviewing the flow of [illegible]
- Setting the base state of [illegible]
- Work with YOLO and find dataset
- Find a simulation used a detection model
- Testing it locally
- Use YOLOv3 with the simulation
- Setting the base state equation with the [illegible] modification
- Find how to deploy it with Machine Learning
- Test the base state on the simulation
- Report the result
- ML dataset
- Suitable database for the admin
- The form of stored data
- ML dataset
- Try storing data from the detection model
- Implementing Admin interfaces
- ML Model testing
- Continue writing the final report



## PROJECT TIMELINE

STEP 1: Initial Review
STEP 2: Light Tasks
STEP 3: Machine Learning model
STEP 4: Review
STEP 5: Review Finished Project

WEEK 8
WEEK 9
WEEK 10
WEEK 11
WEEK 12

- Testing the base state
- Review the report
- Review the whole system
- Admin web development
- The ML Model
- Research Paper writing
- Conference
- Finish all the system requirement
- Final report review
- Project is completed. Present warranty for workmanship and materials, review details with Dr.[illegible]

*Figure 11:* Project Timeline

## 4.4.Stockholder

An individual, a group, or an organization that is actively involved in a software project, can influence its success or failure, and has interests that are affected by its outcome. In the context of software development, stakeholders may have varying degrees of authority and responsibilities.

1. **Primary stakeholders in JAZ**

   Software projects are directly impacted by primary stakeholders. It is their voice that can make or break their income and they have the largest influence.

   - **Supervisor**
     Leading the development team and supervise the project implementation processes, making necessary adjustments.

   - **JAZ Developer Team**
     Manages the delivery of software and estimations on a timely basis. Assess the scope of work and the resources needed for the implementation of business ideas. User-friendly and understandable interfaces are the key to making the product successful.

   - **General Department of Traffic**

   - **The graduation project committee**
     The department of computer science and information technology who provided the guidelines for the project.

2. **Secondary stakeholders in JAZ**

   Software development processes have indirect relationships with secondary stakeholders. As far as your software product and its reputation are concerned, they may not directly affect the project or company, but indirectly may influence it.

   - **Citizens**

   - **Competitors**
     Competitors always implement new features and create industry trends affecting the market. Thus, they influence the prioritization of tasks creating unexpected new challenges to respond to.

   - **Government**

   - **Municipality**

## 4.5. Functional Requirements

There will be two systems:

- Traffic light System (Main system)
- Admin System (Sub-system)

### 1. *Functional Requirement for the traffic system*

***1.*** The admin should be able to set and install the CCTV cameras at traffic lights.
   - The CCTV cameras should be able to capture images of vehicles.
   - The CCTV cameras should be able to cover the whole area of an intersection.

***2.*** The system should pass the captured images to the detection model.
   - The detection model should be trained to detect the different vehicle types.
   - The detection model should be able to detect the type of vehicle successfully.
   - The system should be able to extract traffic data such as {the number of vehicles, and the vehicle types} from the detection model.

3. The system should pass the traffic data to the scheduling algorithm
   - The scheduling algorithm should use the past traffic data to calculate the total number of vehicles of each type in the whole intersection in order to generate the best signal time for each lane.

4. The system should set the optimized green signal timer.
   - The system should count down the timer of the current green signal. When the green timer of the current signal becomes zero, the next signal should become green for time set by the algorithm.

5. The system should update the red signal timers of other traffic lights.
   - The system should be able to update the red signal timers of other traffic lights after the current green signal time is calculated using the scheduling algorithm.

6. The system should switch between the signals.
   - The system should be able to switch between the signals cyclically according to the timers.

### *2. Functional requirement for the admin system*

The admin subsystem is just a small part we'd like to help others with! It doesn't make any difference to our main project, but it will definitely affect other people's projects positively.

In fact, we noticed that we do not have any shareable public data set about vehicles, streets, and statistics in Saudi Arabia. That's why we made sure to include the admin subsystem for extracting useful information in the report format as a nice gesture to save other developers time and effort.

1. The admin should be able to login to the system.
    - The admin should enter username and password.
    - The system should validate the admin's inputs.
    - The system should display an error message if the inputs are not validated.
    - The system should display the home page if the login was successful.

2. The admin should be able to view reports.
    - The system should allow the admin to view daily, weekly, monthly, and yearly reports of the traffic data.
    - The system will display the reports as tables containing the traffic data which is {number of vehicles, number of vehicle types, date, time, waiting time, intersection number}.

3. The admin should be able to print reports.
    - The system should allow the admin to print daily, weekly, monthly, and yearly reports of the traffic data.
    - The system should allow the admin to print the reports as CSV files.

4. The admin should be able to update the system.
    - The admin should be able to update the system whenever needed.

5. The admin should be able to log out.
    - The admin should be able to logout the system when pressing the log out button.
    - The system should redirect the admin to the landing page.

## 4.6. Non-Functional Requirements

1. **Portability requirements**

   The system shall be compatible with different platforms since it operates on web browsers.

2. **Reusability requirements**

   A. The system will be published as an open source for future enhancement from other researchers
   B. JAZ is divided into 3 different module, which help increasing the reusability

3. **Maintainability requirements**

   A. The system organization shall allow to new functionality changes easily.
   B. The system will be checked and observed for regular maintenance to make sure it functions smoothly along the way

4. **Security requirements**

   A. The system validates the administrator's input by checking it from the database to allow only the authorized person to enter the system.

5. **Performance (or Efficiency) requirement**

   A. The computation and processing speed of YOLO is quite high, especially in real-time compared to most of the other training methods and object detection algorithms. The system will use YOLO detection model for faster running at as high as 45 FPS
   B. The system response time should not exceed 6 seconds.
   C. The system's initial loading of the webpages should not exceed 4 seconds from the time the user requests the page
   D. The system's refresh time for the webpages should not exceed 2 seconds.

6. **Robustness requirement**

   A. The system should be able to recover quickly after failure occurs.

7. **Accuracy requirement**

   A. The system will use YOLO algorithm which provides high overall accuracy while reducing background errors seen in other methods

## 8. Dependability (reliability) requirement

A. The system's failure time should not exceed 3 hours as an average.
B. In the event of a system failure for any reason, it will directly switch to fixed time signals to avoid road conflict
C. The system probability of failure per month should not exceed 10%.

## 9. Availability requirement

A. The system should be available to the users 24 hours

## 4.7.UML system model

### *4.7.1. Use case diagram*

A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application.

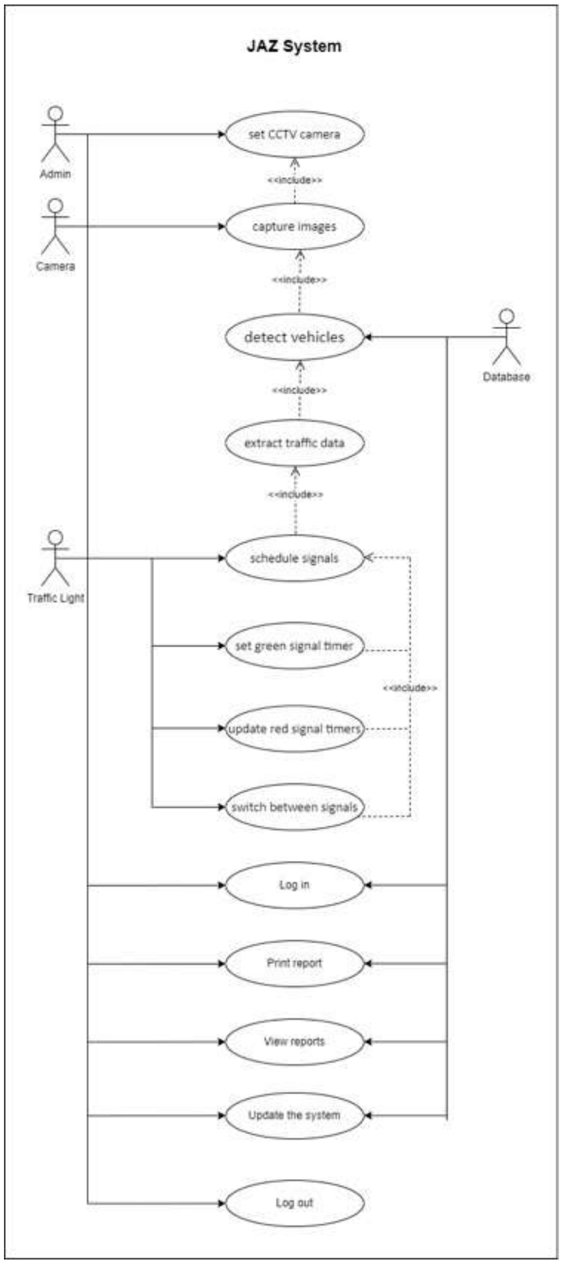


***Figure 12:*** *The Admin and JAZ Systems Use Case Diagram*

### *4.7.2. Use Case Scenarios*

Use case tables are **a structured format for documenting use cases in text form**. They can be a very simple list of basic information about use cases or contain more detailed information.

| Set CCTV camera | |
|---|---|
| Actors | Admin. |
| Descriptions | The CCTV cameras must be installed at traffic lights to start capturing images of vehicles. |
| Data | vehicles. |
| Stimulus | None. |
| Response | The CCTV cameras will start capturing images of vehicles. |

***Table 2:*** *Use case scenario – Set CCTV camera*

| Detect vehicles | |
|---|---|
| Actors | Database. |
| Descriptions | The system will pass the captured images to the YOLO model-based algorithm in order to detect vehicles. |
| Data | The captured images. |
| Stimulus | The system needs to capture the images of vehicles first. |
| Response | The system will extract traffic data from vehicles images. |

***Table 3:*** *Use case scenario – Detect vehicles*

| Schedule signals | |
|---|---|
| Actors | Traffic light. |
| Descriptions | The scheduling algorithm sets the signals timer according to traffic data returned by the vehicle detection model.<br>When the algorithm is first run, the default time is set for the first signal of the first cycle and the times for all other signals of the first cycle and all signals of the subsequent cycles are set by the algorithm. A separate thread is started which handles the detection of vehicles for each direction and the main thread handles the timer of the current signal. |
| Data | The extracted traffic data from the detection model. |

| | |
|---|---|
| Stimulus | The system needs to extract traffic data from the captured vehicles images first. |
| Response | The system will produce the total number of vehicles of each type to generate the best signal time for each direction. |

***Table 4:*** *Use case scenario – Schedule signals*

| Set green signal timer | |
|---|---|
| Actors | Traffic light. |
| Descriptions | The system will set the optimized green time according to the scheduling algorithm while counting down the timer of the current green signal. Once the green timer of the current signal becomes zero, the next signal becomes green for the amount of time set by the algorithm. |
| Data | The total number of vehicles of each type. |
| Stimulus | The system needs to calculate the total number of vehicles of each type first. |
| Response | The system will parse and set the timer of the next green signal. |

***Table 5:*** *Use case scenario – Set green signal timer*

| Update red signal timers | |
|---|---|
| Actors | Traffic light. |
| Descriptions | The system will update the red signal time of the next signal accordingly after calculating the time of the green signal of the current direction. |
| Data | The time of the green signal of current direction. |
| Stimulus | The system needs to calculate the time of the green signal of current direction. |
| Response | The system will update the red signal time. |

***Table 6:*** *Use case scenario – Update red signal timers*

| Switch between signals | |
|---|---|
| Actors | Traffic light. |
| Descriptions | The system will switch between the signals cyclically according to the timers. |
| Data | The green and red signals time. |

| | |
|---|---|
| Stimulus | The system needs to calculate the time of the green and red signal time. |
| Response | The system will switch between signals according to the signals time. |

***Table 7:*** *Use case scenario – Switch between signals*

| Log in | |
|---|---|
| Actors | Database, Admin. |
| Descriptions | The system will let the admin access the system interfaces. |
| Data | Admin's information (username, password). |
| Stimulus | None. |
| Response | The admin will be able to access the system. |

***Table 8:*** *Use case scenario – Log in*

| View reports | |
|---|---|
| Actors | Database, Admin. |
| Descriptions | The system will let the admin view the traffic data in report format. |
| Data | The traffic data. |
| Stimulus | The system database should store the traffic data. |
| Response | The system database will retrieve the traffic data. |

***Table 9****: Use case scenario – View reports*

| Print report | |
|---|---|
| Actors | Database, Admin. |
| Descriptions | The system will let the admin print the traffic data in report format as CSV file. |
| Data | The traffic data. |
| Stimulus | The system database should store the traffic data. |
| Response | The system database will retrieve the traffic data. |

***Table 10:*** *Use case scenario – Print report*

| Update the system | |
|---|---|
| Actors | Database, Admin. |
| Descriptions | The system will be updated by the admin whenever needed. |
| Data | The CCTV camera.<br>The detection model.<br>The scheduling algorithm. |
| Stimulus | The system needs to set in advance. |
| Response | The system will be updated. |

***Table 11:*** *Use case scenario – Update the system*

| Log out | |
|---|---|
| Actors | Admin. |
| Descriptions | The system will let the admin exit the system. |
| Data | None. |
| Stimulus | The admin needs to logged in. |
| Response | The admin will be able to exit the system. |

***Table 12:*** *Use case scenario – Log out*

### *4.7.3. Class diagram*

It represents the types of objects residing in the system and the relationships between them. A class consists of its objects, and also it may inherit from other classes. A class diagram is used to visualize, describe, document various different aspects of the system, and also construct executable software code. It shows the attributes, classes, functions, and relationships to give an overview of the software system. It constitutes class names, attributes, and functions in a separate compartment that helps in software development. Since it is a collection of classes, interfaces, associations, collaborations, and constraints, it is termed as a structural diagram.

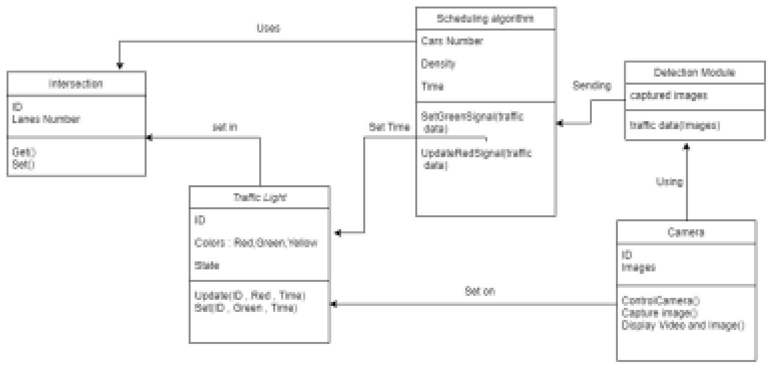


***Figure 13:*** *JAZ class diagram*

### *4.7.4. Data flow diagram*

**Components of DFD**

The Data Flow Diagram has 4 components:

- **Process**
  Input to output transformation in a system takes place because of process function. The symbols of a process are rectangular with rounded corners, oval, rectangle or a circle. The process is named a short sentence, in one word or a phrase to express its essence
- **DataFlow**
  Data flow describes the information transferring between different parts of the systems. The arrow symbol is the symbol of data flow. A relatable name should be given to the flow to determine the information which is being moved. Data flow also represents material along with information that is being moved. Material shifts are modeled in systems that are not merely informative. A given flow should only transfer a single type of information. The direction of flow is represented by the arrow which can also be bi-directional.
- **Warehouse**
  The data is stored in the warehouse for later use. Two horizontal lines represent the symbol of the store. The warehouse is simply not restricted to being a data file rather it can be anything like a folder with documents, an optical disc, a filing cabinet. The data warehouse can be viewed independent of its implementation. When the data flow from the warehouse it is considered as data reading and when data flows to the warehouse it is called data entry or data updation.
- **Terminator**
  The Terminator is an external entity that stands outside of the system and communicates with the system. It can be, for example, organizations like banks, groups of people like customers or different departments of the same organization, which is not a part of the model system and is an external entity. Modeled systems also communicate with terminator.

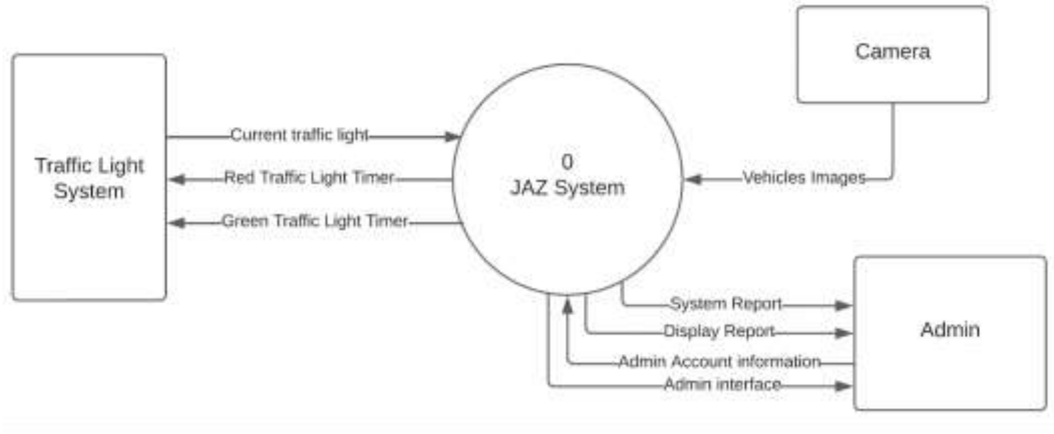


***Figure 14:*** *Data Flow Diagram DFD - Context Diagram (Level 0)*

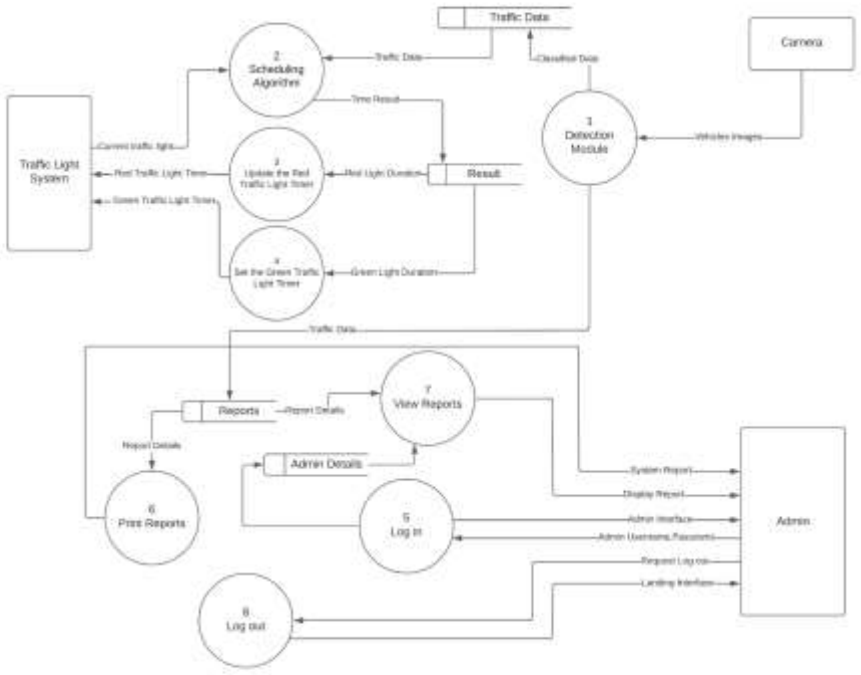
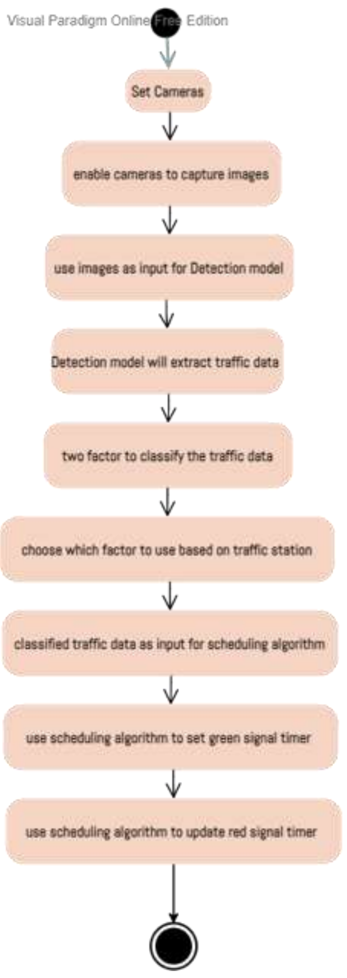


***Figure 15:*** *Data Flow Diagram DFD (Level 1)*

### 4.7.5. *Activity diagram*

It captures the dynamic behavior of the system. Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques.


Visual Paradigm Online Free Edition
Set Cameras
enable cameras to capture images
use images as input for Detection model
Detection model will extract traffic data
two factor to classify the traffic data
choose which factor to use based on traffic station
classified traffic data as input for scheduling algorithm
use scheduling algorithm to set green signal timer
use scheduling algorithm to update red signal timer

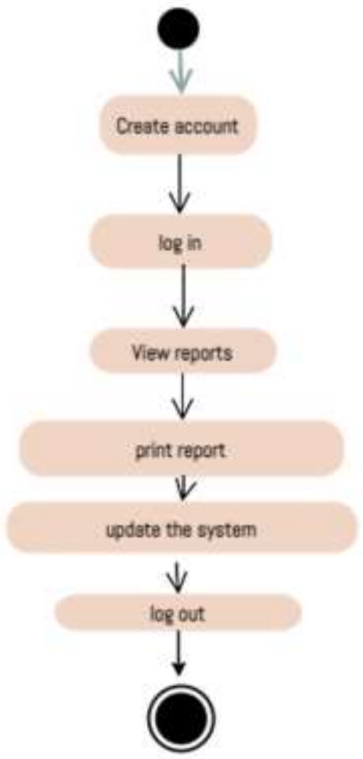


***Figure 16:*** *Activity Diagram of JAZ System*

***Figure 17:*** *Activity Diagram of the Admin System*

### 4.7.6. State diagram

A state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli.

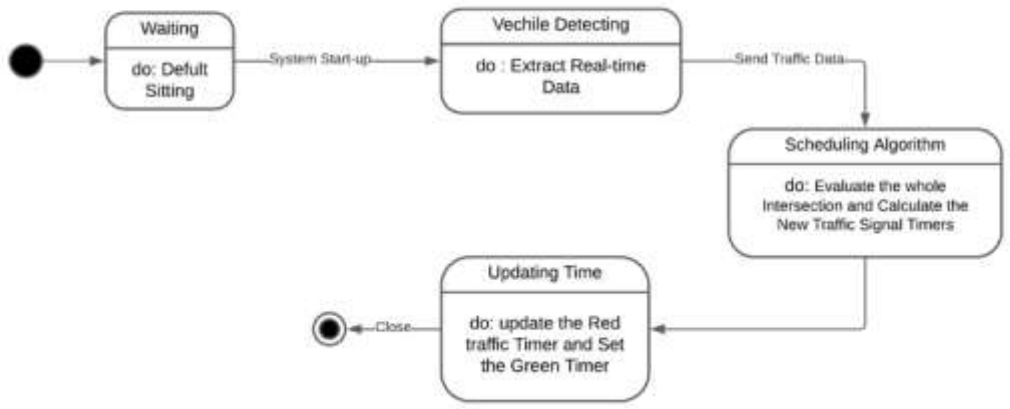

Waiting
do: Defult Sitting
System Start-up
Vechile Detecting
do : Extract Real-time Data
Send Traffic Data
Scheduling Algorithm
do: Evaluate the whole Intersection and Calculate the New Traffic Signal Timers
Updating Time
do: update the Red traffic Timer and Set the Green Timer
Close

***Figure 18:*** *State Diagram – JAZ System*

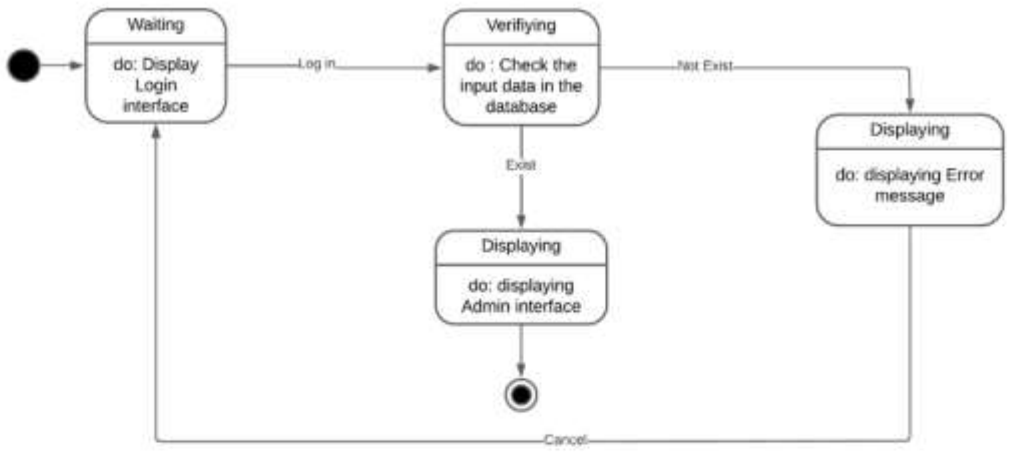

Waiting
do: Display Login interface
Log in
Verifiying
do : Check the input data in the database
Not Exist
Displaying
do: displaying Error message
Exist
Displaying
do: displaying Admin interface
Cancel

***Figure 19:*** *State Diagram – Admin Login*

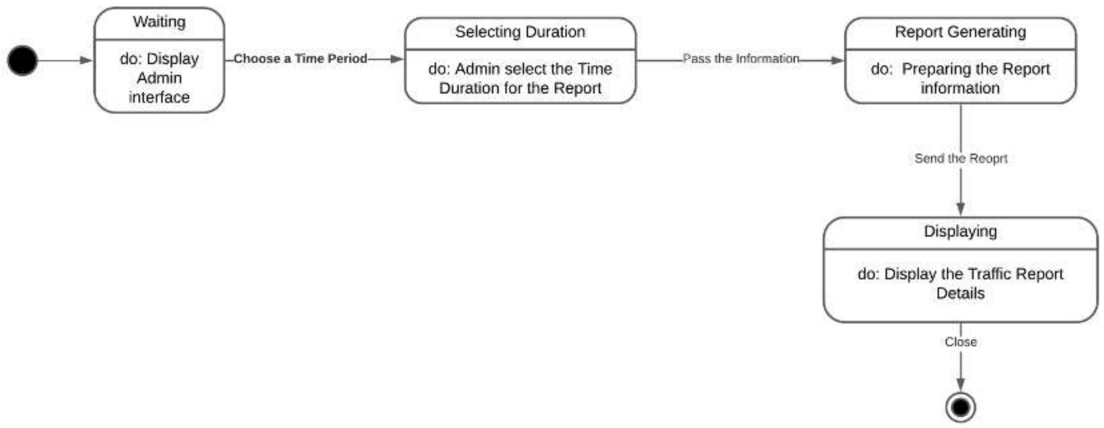

Waiting
do: Display Admin interface
Choose a Time Period
Selecting Duration
do: Admin select the Time Duration for the Report
Pass the Information
Report Generating
do: Preparing the Report information
Send the Reoprt
Displaying
do: Display the Traffic Report Details
Close

***Figure 20:*** *State Diagram – Admin View Report*

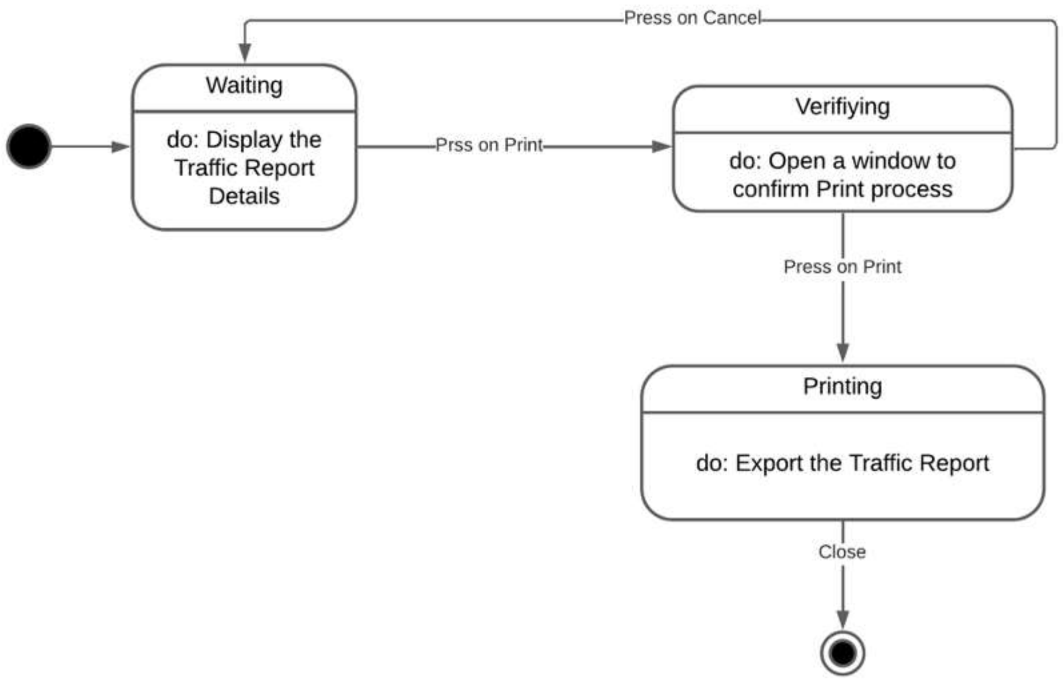

Press on Cancel
Waiting
do: Display the Traffic Report Details
Prss on Print
Verifiying
do: Open a window to confirm Print process
Press on Print
Printing
do: Export the Traffic Report
Close

***Figure 21:*** *State Diagram – Admin Print Report*

### 4.7.7. Sequence diagram

Sequence Diagrams are time focus, and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

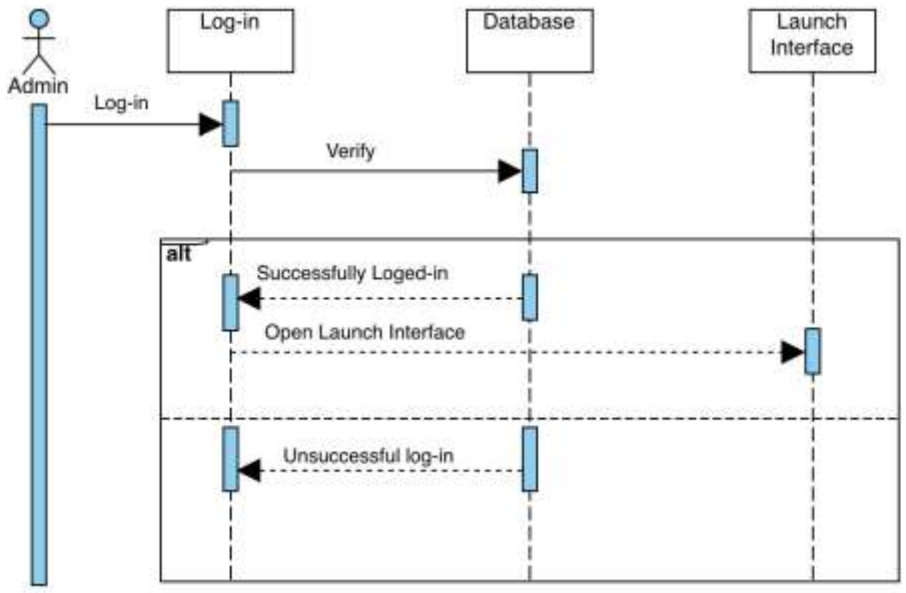


***Figure 22:*** *Admin login sequence diagram*

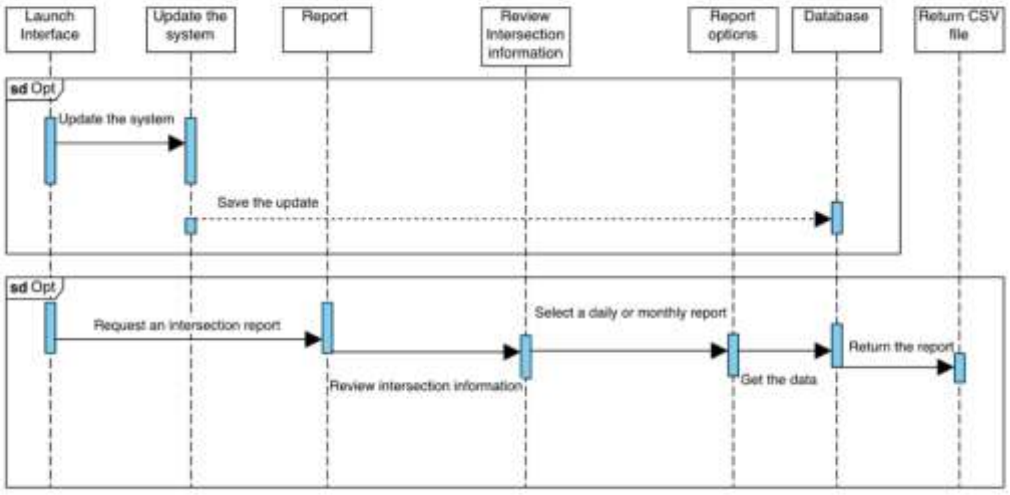


***Figure 23:*** *Admin viweing and printing report's sequence diagram*

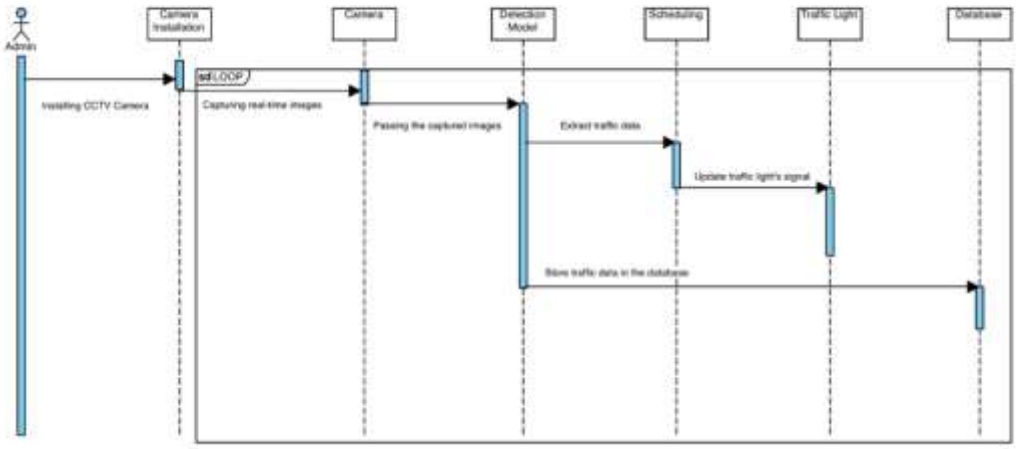


***Figure 24:*** *JAZ System sequence diagram*

## 4.7.8. COMPONENT DIAGRAM

**Component diagram** shows components, provided and required interfaces, ports, and relationships between them.

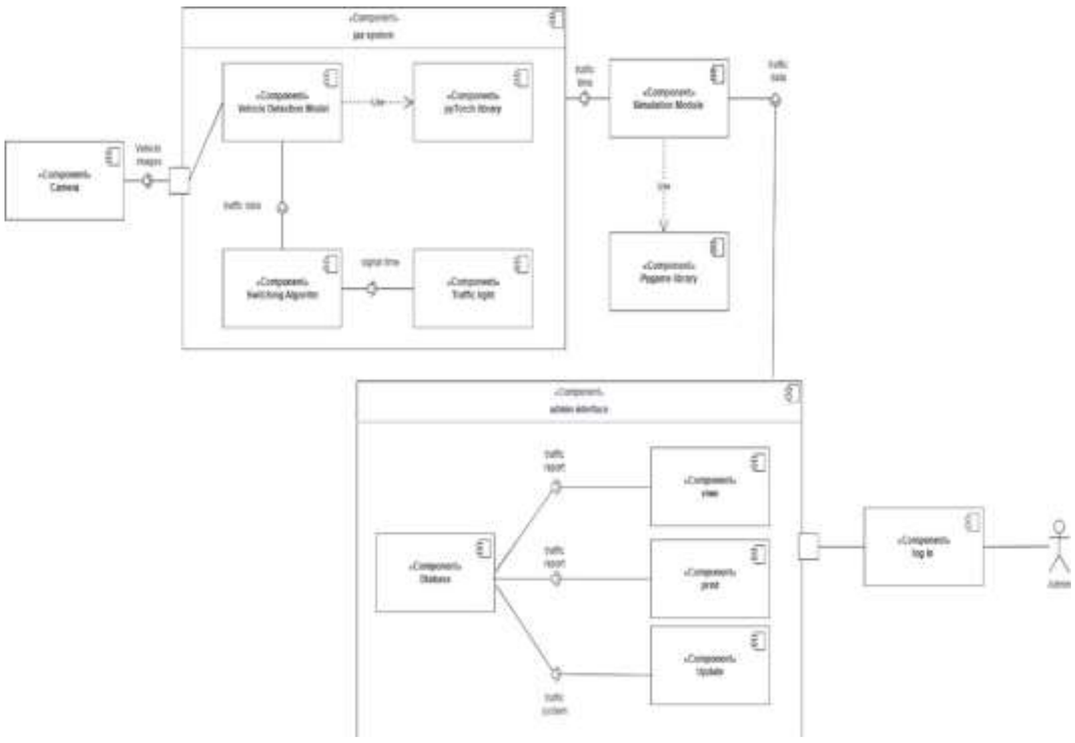


***Figure 25:*** *Component Diagram of the Admin and JAZ System*

# CHAPTER 5

# SYSTEM IMPLEMENTATION

# 5. SYSTEM IMPLEMENTATION

## 5.1. SYSTEM CONSTRAINTS

### *5.1.1. Hardware and software environment*

- **Hardware environment**

For the development of the Jaz system, members of the Jaz team will use their personal computers.

| PC | System Type | RAM | PROCESSOR | STORAGE | GRAPHICS |
|---|---|---|---|---|---|
| 1 | 64-bit operating system, x64-based processor | 16.0 GB | Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz | 1TB HDD + 256GB SSD | NAVIDIA GeForce MX250 + Intel(R) UHD Graphics 620 |
| 2 | 64-bit operating system, x64-based processor | 16.0 GB | Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 1.99 GHz | 1TB HDD + 256GB SSD | NAVIDIA GeForce MX250 + Intel(R) UHD Graphics 620 |
| 3 | macOS Monterey | 16GB | 1.7GHz quad-core Intel Core i7<br><br>Turbo Boost up to 4.5GHz | 512GB SSD | Intel Iris Plus Graphics 645 |
| 4 | macOS | 8 GB 2133 MHz LPDDR3 | Dual-Core Intel Core i5<br><br>2.3 GHz | 128 GB | Intel Iris Plus Graphics 640 1536 MB |

| | | | | | |
|---|---|---|---|---|---|
| 5 | x64-based PC | 16GB | Processor Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 2592 Mhz, 6 Core(s), 12 Logical Processor(s) | 256 GB | Intel(R) UHD Graphics 630 +NVIDIA GeForce GTX 1660 Ti |

***Table 13:*** *Hardware environment*

- Software environment

A variety of technologies will be used to implement the project. This section describes the technologies used in this project.

| | |
|---|---|
| **Interface design tool** | **Figma** - Figma is a collaborative interface design web application with additional offline functionality enabled by the desktop application. |
| **Full-Stack-Web development** | **Python** - Another reason to use Python is that it is not only useful for applications because it provides dynamic writing capabilities and is the first choice for programming artificial intelligence applications, but it is also an open-source product. is a highly adaptable and efficient programming language.<br><br>**Pytorch** - PyTorch is an open-source machine learning (ML) framework based on the Python programming language and Torch library. It is one of the suitable platforms for deep learning research. This framework is designed to speed up the process between research prototyping and deployment.<br><br>**Flutter -** Flutter platform is Google's open-source application development platform for mobile, desktop, and web apps. Flutter SDKs allow you to build native mobile applications with only one codebase, unlike other popular solutions. Therefore, you can create two different apps with the same programming language and codebase |

| | |
|---|---|
| **Database** | **Firebase -** Firebase is a Google “development platform” served as a database for the capstone project. Besides it providing Realtime database, it offers many other products such as Analytics, Cloud Storage, Hosting, and many other great useful products for development. |
| **Integration** | **GitHub -** GitHub, Inc., is an Internet hosting service for software development and version control using Git. It provides the distributed version control of Git plus access control, bug tracking, software feature requests, task management, continuous integration, and wikis for every project.<br><br>**Google Colaboratory -** Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs |
| **IDE** | **Visual Studio Code -** Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. |
| **Detection model** | **YOLO -** The detection module will involve training a deep learning model called YOLO (You Only Look Once) in Real-Time Object Detection to detect the vehicles |
| **Simulation** | **Pygame –** a cross-platform set of Python modules designed for writing video games which is easy to use and runs in any environment |

***Table 14:*** *Software environment*

## 5.2.System Flow

1. **Dataset**
   A vehicle dataset in YOLO format, to train the model.
   Labeling and augmenting more images in some classes help in getting higher result and avoid data biassing.

2. **Deep Learning Models**
   Train YOLOv5 and YOLOv7 on the dataset and finetune the model to achieve high accuracy.

3. **Extracted Information**
   From the detection model, get important information from the intersection, such as the type of each vehicle and the total vehicle in the given image.

4. **Scheduling Algorithm**
   Use the data from the detection model to calculate the optimal green light time based on the intersection condition in term of congestion.

5. **Simulation Testing**
   Test our equation on a simulation to measure its performance and help reporting and finetune the scheduling algorithm.

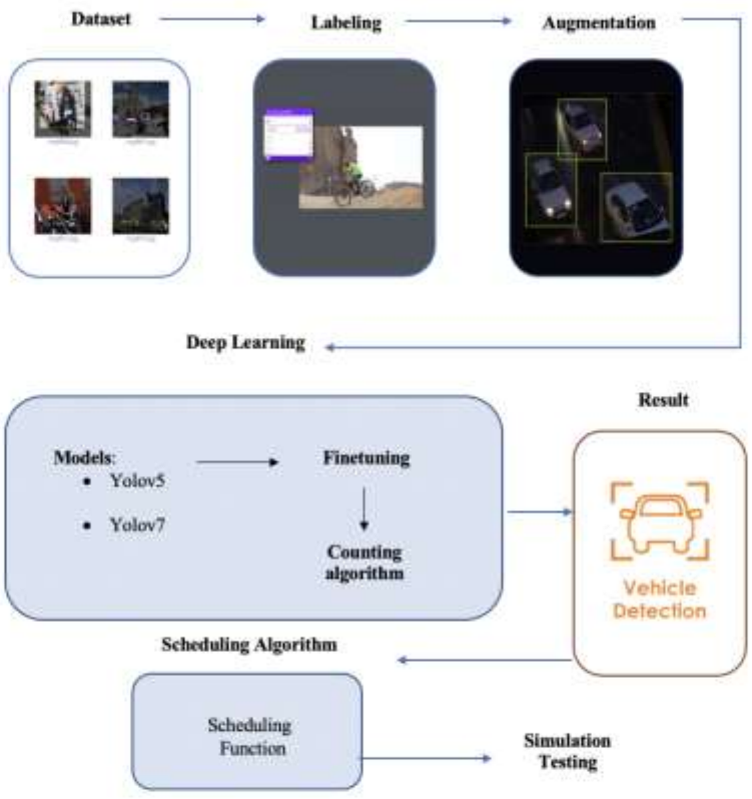


***Figure 26:*** *System Flow*

## 5.3. Detection Model

### 5.3.1. JAZ Detection Model

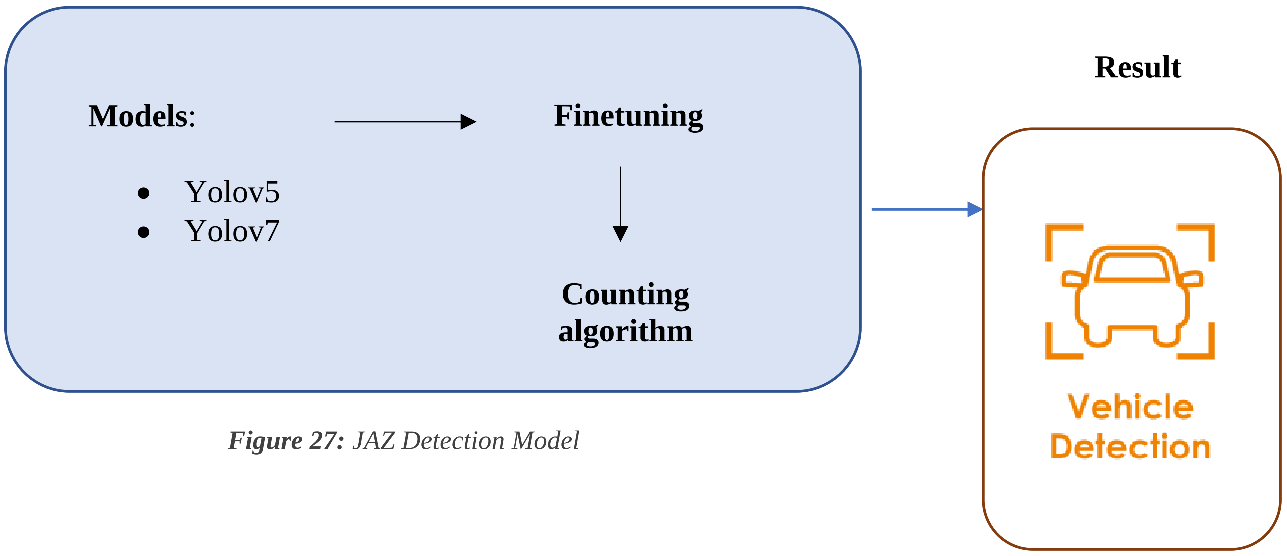


***Figure 27:*** *JAZ Detection Model*

JAZ Detection model is mainly developed to detect vehicles at the intersection and take advantage from the detection model to take the right choice of traffic light timing based on the intersection condition, whether the next street is at a congestion or not, and calculate the green light time based on this information. A proper dataset for this model is found to train a YOLOv5 and YOLOv7 models.

The dataset is labeled with 5 classes, Car, Bus, Truck, Bike, Bicycle. Robowflow is used at the dataset stage to finetune the dataset and label more data for some of the classes in the dataset.

After training YOLO on the custom dataset, and writing its result, another python code is added to count the number of vehicles at the intersection and display the counting at the screen. The result from this model is used in the scheduling algorithm to determine the optimal green light time for each traffic light.

### 6.3.2. YOLO and how it works

YOLO is an algorithm that provides real-time object detection using neural networks. The popularity of this algorithm is due to its accuracy and quickness. It has been applied in a variety of ways to identify animals, humans, parking meters, and traffic lights. YOLO is an abbreviation for the term ‘You Only Look Once’. This program identifies and finds different things in an image (in real-time). The class probabilities of the discovered photos are provided by the object identification process in YOLO, which is carried out as a regression problem.

Convolutional neural networks (CNN) are used by the YOLO method to recognize items instantly. The approach just needs one forward propagation through a neural network to identify objects, as the name would imply. This indicates that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.

***The important of YOLO algorithm***

YOLO algorithm is important because of the following reasons:
Some of the reasons why YOLO is leading the competition include its:

- Speed
- Detection accuracy
- Good generalization
- Open source

**1. Speed**

Because it doesn't deal with complicated pipelines, YOLO is incredibly quick. At 45 frames per second, it can process pictures (FPS). YOLO is a fantastic choice for real-time processing since it can achieve more than double the mean Average Precision (mAP) compared to other real-time systems.

With 91 FPS, YOLO is clearly superior to the other object detectors, as shown in the graph below.

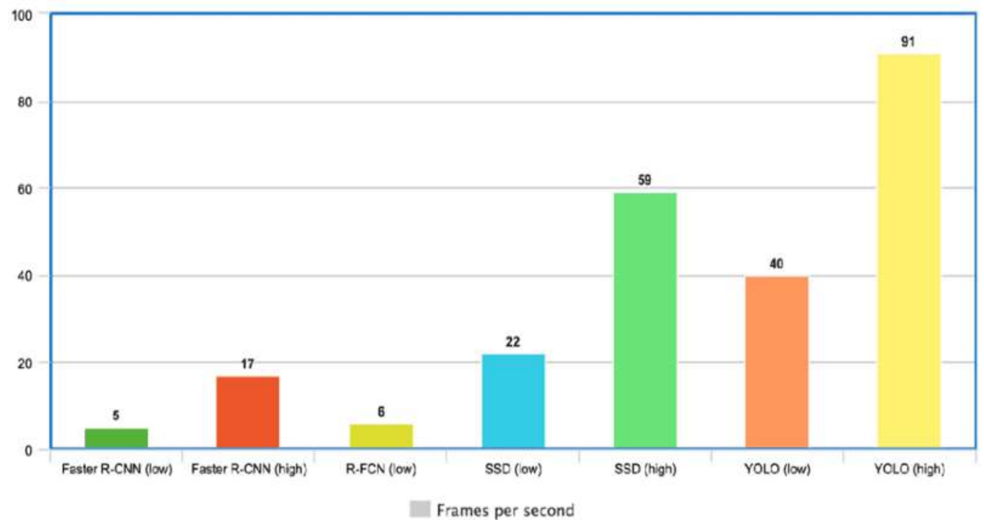


***Figure 28:*** *YOLO Speed compared to other state-of-the-art object detectors*

**2. High detection accuracy**

With a very low amount of background mistakes, YOLO's accuracy greatly outpaces that of other cutting-edge models.

**3. Better Generalization**

This is particularly true for the updated YOLO versions. With those improvements, YOLO went a bit further and offered improved generalization for new domains, making it ideal for applications that require quick and reliable object identification. The Automatic Identification of Melanoma with Yolo Deep Convolutional Neural Networks article, for instance, demonstrates that, when compared to YOLOv2 and YOLOv3, the initial version of YOLOv1 has the lowest mean average precision for the automatic detection of melanoma disease

4. **Open source**

Making YOLO open source led the community to continuously enhance the model. This is among the factors that have contributed to YOLO's rapid rise.

### 6.3.3. YOLO Architecture

YOLO architecture is similar to **GoogleNet**. As illustrated below, it has overall 24 convolutional layers, four max-pooling layers, and two fully connected layers.

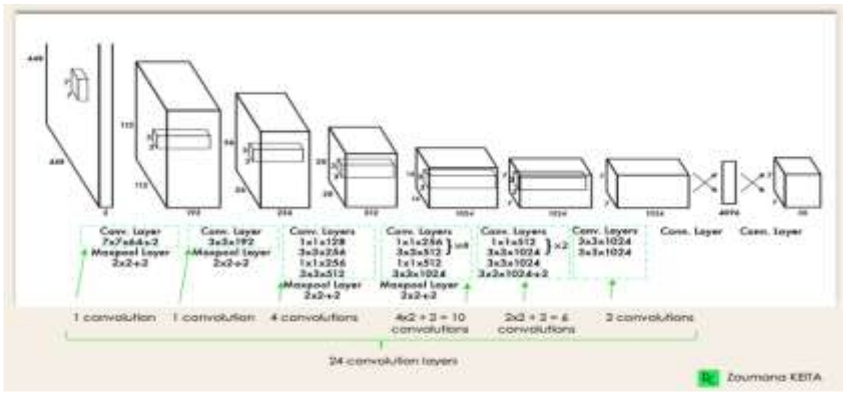

Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer
1 convolution
1 convolution
4 convolutions
4x2 + 2 = 10 convolutions
2x2 + 2 = 6 convolutions
2 convolutions
24 convolution layers
Zoumana KEITA

*Figure 29*: *YOLO Architecture from the original paper*

The architecture works as follows:

- Before passing the input picture through the convolutional network, it is resized to 448x448.

- To create a cuboidal output, a 3x3 convolution is used after a 1x1 convolution to minimize the number of channels.

- Some extra approaches, such as batch normalization and dropout, respectively regularize the model and prevent it from overfitting.

- The activation function used internally is ReLU, with the exception of the last layer, which employs a linear activation function.

**How Does YOLO Object Detection Work?**

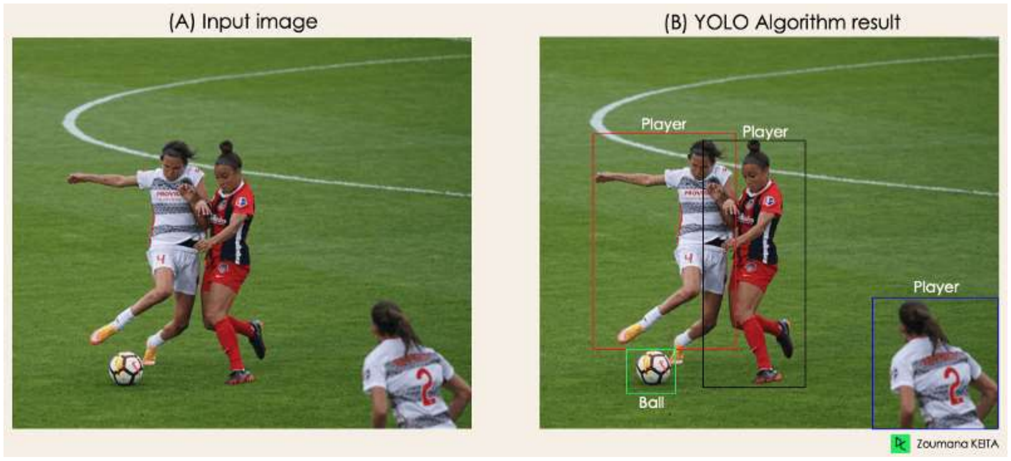


***Figure 30:*** *YOLO Algorithm Result*

The algorithm works based on the following four approaches:

- Residual blocks
- Bounding box regression
- Intersection Over Unions or IOU for short
- Non-Maximum Suppression.

**1- Residual blocks**

In this initial phase, the original image (A) is divided into NxN grid cells with equal shapes, where N in this example is 4 as indicated on the right image. The class of the item that each grid cell covers must be predicted locally, together with the probability/confidence value.

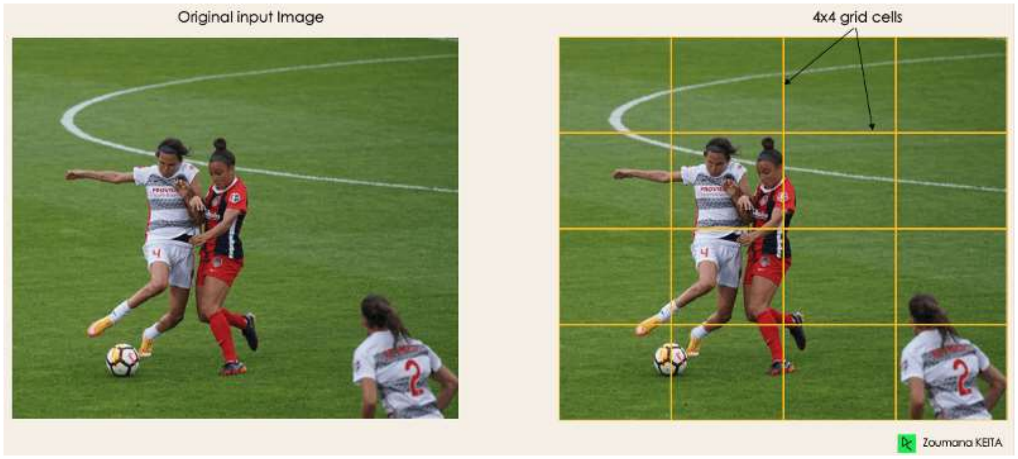


**Figure 31:** YOLO Residual blocks

### 2- Bounding box regression

The next step is to identify the bounding boxes, which are rectangles that highlight all of the image's objects. As many bounding boxes as, there are objects in a given image is possible.
The properties of these bounding boxes are determined by YOLO using a single regression module, where Y is the final vector representation of each bounding box.

$$Y = [pc, bx, by, bh, bw, c1, c2]$$

This is particularly crucial during the model's training phase.
• The probability score of the grid containing an object is represented by pc. For example, every red grid will have a probability value greater than zero. The simpler form is shown on the right since there is no chance that any of the cells will be yellow (insignificant).

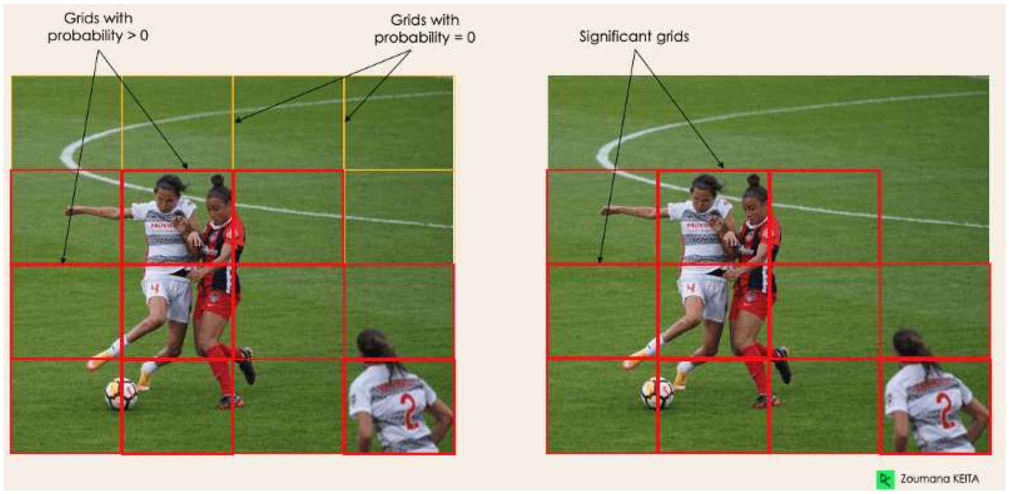


**Figure 12:** YOLO Bounding box

• The bounding box's centre's x and y coordinates in relation to the surrounding grid cell are given by bx, by.

• bh, bw stand for the bounding box's height and breadth in relation to the surrounding grid cell.

• The two classes Player and Ball are represented by c1 and c2. If your use case calls for more classes, we can have them.

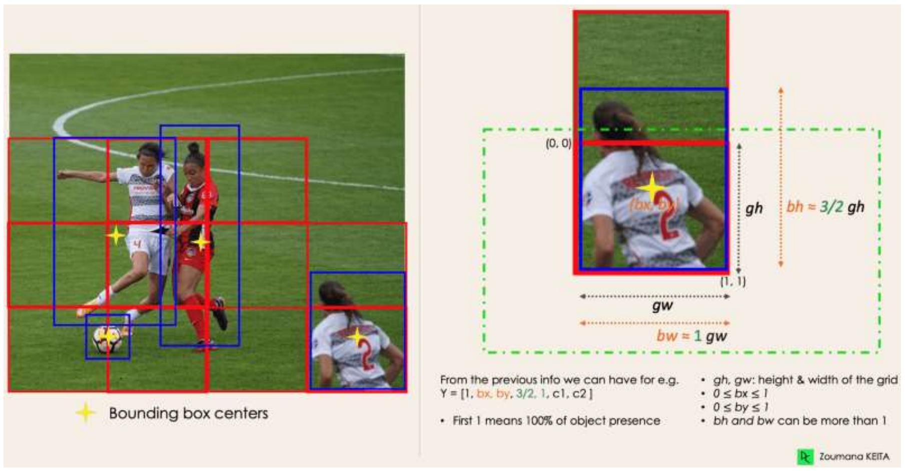

Bounding box centers
(0, 0)
(bx, by)
gh
bh = 3/2 gh
(1, 1)
gw
bw = 1 gw
From the previous info we can have for e.g.
Y = [1, bx, by, 3/2, 1, c1, c2 ]
• First 1 means 100% of object presence
• gh, gw: height & width of the grid
• 0 ≤ bx ≤ 1
• 0 ≤ by ≤ 1
• bh and bw can be more than 1
Zoumana KEITA

**Figure 33:** YOLO Bounding box regression

### 3- Intersection Over Unions or IOU

Despite not all of them being significant, a single item in a picture might frequently have many grid box possibilities for prediction. The IOU's (a value between 0 and 1) purpose is to eliminate such grid boxes and retain just the necessary ones. Here is the reasoning for it:

• The IOU selection threshold is set by the user and can be, for example, 0.5.

• After that, YOLO determines each grid cell's IOU, which is calculated by dividing the intersection area by the union area.

• Finally, it evaluates grid cells with an IOU > threshold rather than those predicted to have an IOU threshold.

An example of using the grid selection method on the item in the bottom left is shown below. We can see that the item had two potential grid options at first, but only "Grid 2" was ultimately chosen.

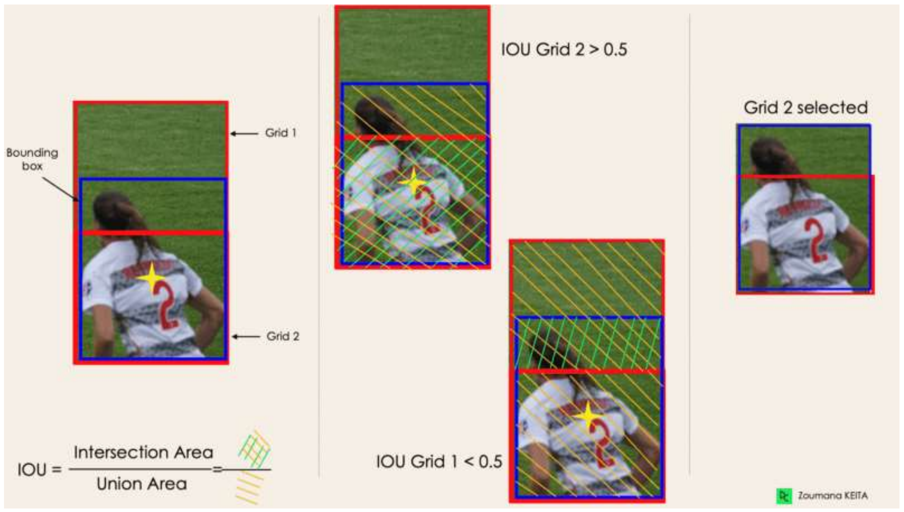


**Figure 34:** YOLO Intersection Over Unions or IOU

**4- Non-Max Suppression or NMS**

Setting an IOU threshold is not always sufficient since an item may have numerous boxes with IOU that exceeds the threshold and keeping all of those boxes open might result in the inclusion of noise. Here, NMS may be used to keep just the boxes with the highest likelihood of being discovered. [25]

### 6.3.4. YOLOv5

The YOLOv5 model family, which was developed using the COCO dataset, consists of compound-scaled object detection models. It has basic capabilities for Test Time Augmentation (TTA), model assembling, hyperparameter evolution, and export to ONNX, CoreML, and TFLite.

A cutting-edge, real-time object detection system called YOLO was developed in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. It was pre-trained using the COCO dataset. To process a whole image, it only employs one neural network. The program divides the image into areas, and for each region, it forecasts probability and bounding boxes.

Yolo is well renowned for its speed and accuracy, and it has been utilized in numerous fields like self-driving automobiles, healthcare, and security monitoring. The Ultralytics team has been trying to improve this model since 2015, and numerous iterations have been released since then.

Here are two different kinds of object detection models: single-stage and two-stage models. As illustrated in the diagram below, single-stage object detectors (like YOLO architecture) consist of three parts: a backbone, a neck, and a head.

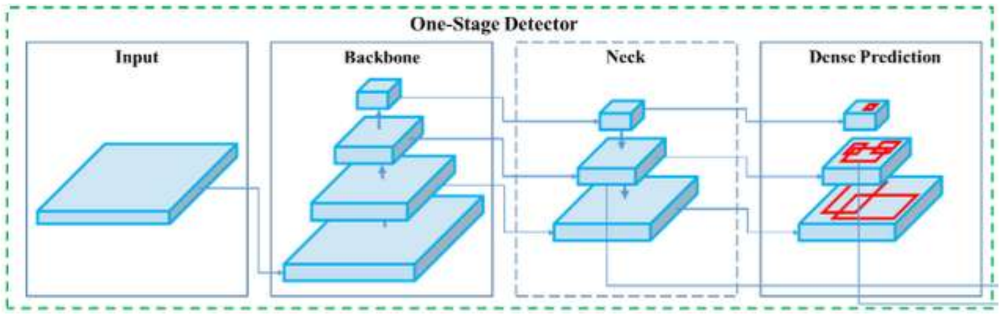


***Figure 35:*** *Single-Stage Detector Architecture*

**Model Backbone**

A pre-trained network serves as the backbone and is used to extract rich feature representation for images. This aids in lowering the image's spatial resolution and raising its feature (channel) resolution.

**Model Neck**

Pyramids of feature data are extracted from the model neck. This makes it easier for the model to generalize to objects of various sizes and scales.

The final stage operations are carried out on the model head. It renders the finished product, which includes classes, objectness scores, and bounding boxes, by applying anchor boxes to feature maps.

As with other single-stage object detectors, YOLO v5 is a single-stage object detector and contains three key components.

The main purpose of Model Backbone Model Neck Model Head Model Backbone is to extract significant information from the input image. To extract valuable, important features from an input image in YOLO v5, the CSP — Cross Stage Partial Networks are employed as the backbone. The primary purpose of Model Neck is to produce feature pyramids. Pyramids of features enable models to scale objects successfully in general. The ability to recognize the same thing in various sizes and scales is helpful.

Models that use feature pyramids perform well on unobserved data. Other models, such as FPN, BiFPN, PANet, etc., employ other feature pyramid methodologies. PANet is utilized in YOLO v5 as a neck to obtain feature pyramids. To learn more about features of pyramids

The final detecting step is primarily carried out using the model Head. It used anchor boxes on the features and produced final output vectors that included bounding boxes, objectness scores, and class probabilities.

The head of the YOLO v5 model is identical to the heads of the YOLO V3 and V4 models. [26]

**Activation Function**

Any deep neural network's selection of activation functions is extremely important. Many activation functions, such Leaky ReLU, mish, and swish, have recently been introduced. The Leaky ReLU and Sigmoid activation function was chosen by the authors of YOLO v5.

In YOLO v5, the final detection layer uses the sigmoid activation function whereas the middle/hidden layers use the Leaky ReLU activation function.

**Optimization Mechanism**

We have two choices for the optimization function in YOLO v5.

1. SGD
2. Adam

**SGD** is the standard optimization function for training in YOLO v5.

Nevertheless, you can change it to **Adam** by using the command-line parameter "— — adam".

**Loss function versus cost function**

A compound loss is calculated for the YOLO family of algorithms based on the objectness score, class probability score, and bounding box regression score.

For the loss computation of class probability and object score, Ultralytics employed the Binary Cross-Entropy with Logits Loss function from PyTorch. To calculate the loss, we also have the option of using the Focal Loss function. You have the option to use the fl gamma hyper-parameter to train with focal loss. [27]

### 6.3.5. YOLOv7

The recent release of YOLOv7 [28], where the researcher claimed that it outperforms all known object detectors in both speed and accuracy and has the highest accuracy 56.8% AP among all known real-time object detectors, stands out among the many different object detection models that are effective for specific use cases. In terms of accuracy and speed, the suggested YOLOv7 version E6 outperformed transformer-based detectors like SWINL Cascade-Mask R-CNNR-CNN. Scaled-YOLOv4, YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, and Vit-Adapter-B all underperformed YOLOv7.

The most recent member of the YOLO (You Only Look Once) model family is the v7 model. Object detectors using YOLO models have a single stage. Image frames are enhanced in a YOLO model by a backbone. The network's head receives these traits after they have been merged and mingled in the neck. Bounding boxes should be drawn around specific locations and object types according to YOLO.

**The different:**

By developing a network architecture that could predict bounding boxes more correctly than its competitors at comparable inference speeds, the YOLOv7 developers aimed to advance the state of the art in object identification.

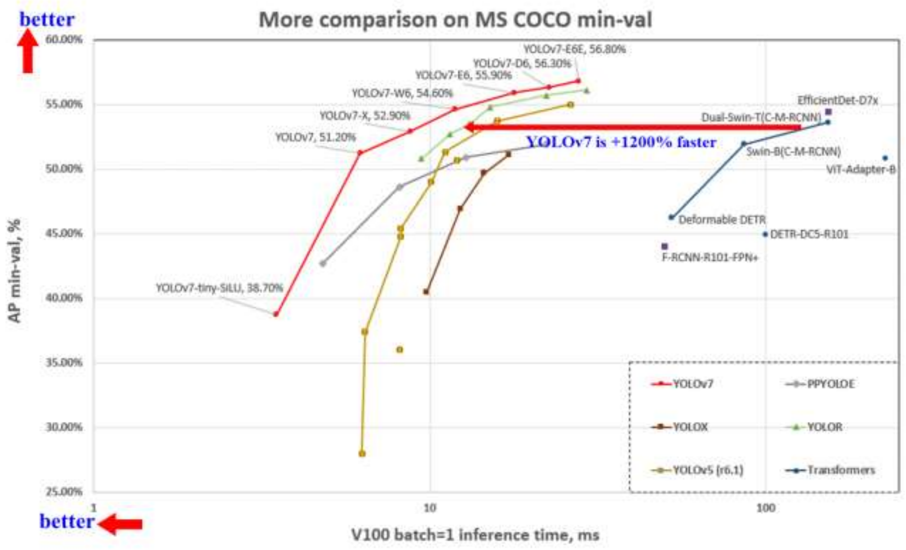


*Figure 36:* *YOLO Versions Comparison*

Faster and more accurately than its peer networks, YOLOv7 assesses in the top left.

The YOLOv7 authors made a number of adjustments to the YOLO network and training procedures in order to get these findings. The three significant advancements in computer vision research that were made in the YOLOv7 study are discussed here.

**Model re-parameterization**

Model re-parameterization is designed to reduce inference time by combining several computing modules into one during the inference stage. Model-level ensemble and module-level ensemble are the two primary divisions of model re-parameterization approaches. In the first case, weights from several trained models are averaged after training numerous identical models with various training sets. The latter is responsible for calculating a weighted average of model weights over various iteration counts. Module-level re-parameterization has become very popular in recent years. Additionally, certain re-parameterization approaches are not architecture agnostic, which means that only a select few architectures may use them. A new type of re-parameterization that addresses the weakness of earlier approaches is introduced in YOLO v7.

**Model Scaling**

Model scaling is a technique for scaling up or down an existing model to accommodate any computer platform. As a result, the architecture is quite flexible. There are several forms of scaling, including stage scaling, depth scaling (number of layers), width scaling (number of channels), and resolution scaling (size of the input picture) (number of feature pyramids). In essence, model scaling reveals the trade-off between accuracy and speed.
Model scaling techniques like network architecture search (NAS) are frequently employed. Without establishing overly complex criteria, NAS may automatically scan the search area for appropriate scaling factors. The drawback of NAS is that it needs expensive computing to finish looking for model scaling factors. The primary issue with NAS is the independent analysis of scaling variables. To address the issue of separate assessments, YOLO v7 provides a new scaling technique.

**Model Architecture**
**Extended efficient layer aggregation networks**

The size, quantity, and computing density of a model are the main considerations in the construction of an efficient architecture. The VovNet model takes a step further and examines how the input/output channel ratio, the number of architectural branching, and the element-wise operation affect the performance of network inference. The next significant advancement in architectural search is known as ELAN, and YOLO v7 extends this to become E-ELAN. The ELAN article came to the conclusion that a deeper network can successfully learn and converge if the shortest longest gradient path is controlled.

Large-scale ELAN has achieved stability regardless of the gradient path length or the number of computing blocks stacked. This stable condition could be lost if infinitely more computing blocks are piled, and the rate of parameter consumption will drop. Expand, shuffle, and merge cardinality are methods used by E-ELAN to constantly improve the network's learning capacity while preserving the original gradient route. E-ELAN solely modifies the architecture of the computing block; the transition layer's design remains unaltered. Group convolution is used in the E-

ELAN technique to increase the channel and cardinality of computing blocks. It gives all the computational blocks in a computational layer the same group parameter and channel multiplier.

In accordance with the predetermined group parameter g, the feature map produced by each computing block is subsequently divided into g groups and concatenated. Currently, each set of feature maps will have the same amount of channels as there were in the original design. Finally, execute merge cardinality by adding g groups of feature maps. Along with upholding the original ELAN design architecture, E-ELAN may direct other collections of computational blocks to pick up additional, more varied functionalities.

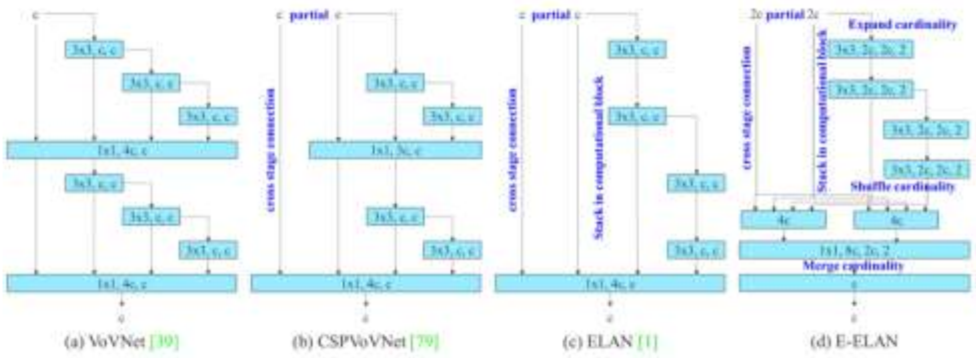


***Figure 37:*** *E-ELAN architecture*

**Model scaling for concatenation-based models**

Model scaling is mostly used to modify certain model properties and produce models at various sizes to accommodate various inference rates. They scale between width, depth, and resolution in EfficeintNet, a well-known Google architectural design. Later, however, researchers attempted to determine the impact of group and vanilla convolution on the quantity of parameters and computation while executing scale.

Concatenation-based architecture is not a good fit for EfficientNet's approach because when scaling up or scaling down is done on depth, the in-degree of a translation layer that comes right after a concatenation-based computing block would change. For instance, scaling-up depth will modify the ratio between a transition layer's input channel and output channel, which might reduce the model's hardware use. The suggested approach should also figure out how the output channel of a computational block will vary as its depth factor is adjusted. The outcome is displayed in the picture below after applying width factor scaling to the transition layers with the same amount of change. The optimum structure and the model's original attributes can both be preserved via the YOLO v7 compound scaling approach.

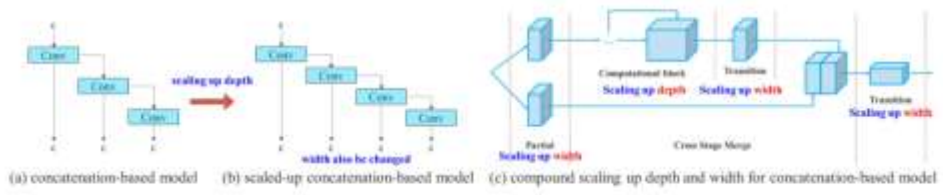

scaling up depth
width also be changed
Computational block
Transition
Scaling up depth
Scaling up width
Partial
Cross Stage Merge
(a) concatenation-based model
(b) scaled-up concatenation-based model
(c) compound scaling up depth and width for concatenation-based model

***Figure 38:*** *Model Scaling*

**Trainable bag-of-freebies**
**Planned re-parameterized convolution**

Let's take a closer look at RepConv, a different kind of convolutional block, before we get into more depth about this. Similar to Resnet, RepConv has one identification connection as well as a second connection with a 1x1filter in between.

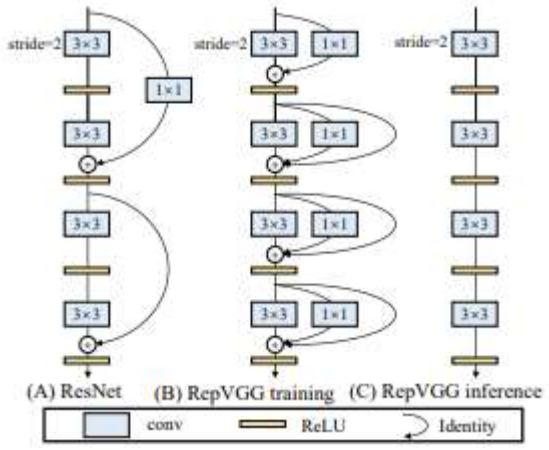


***Figure 39:*** *RepConv being used in VGG*

Researchers from YOLOv7 examined how re-parameterized convolution should be coupled with various networks using gradient flow propagation routes. The placement of the conv blocks is shown in the diagram below. Four of the eight combinations are excellent (marked with a green tick in the below image).

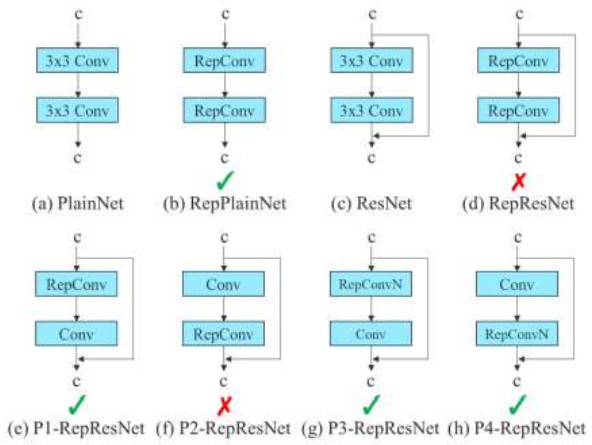


***Figure 40:*** *YOLOv7 Conv Blocks*

**Coarse for auxiliary and fine for lead loss**

Deep supervision is a method that is frequently applied while deep networks are being trained. The shallow network weights using assistant loss as the guidance, and the main idea is to increase the number of auxiliary heads in the middle levels of the network. Deep supervision may nevertheless considerably enhance the model's performance on many tasks, even for designs like ResNet and DenseNet that often converge well. The object detector architecture is depicted below in both its "without" and "with" deep supervision states. In the YOLOv7 design, the lead head is in charge of producing the output, while the auxiliary head is in charge of assisting in training.

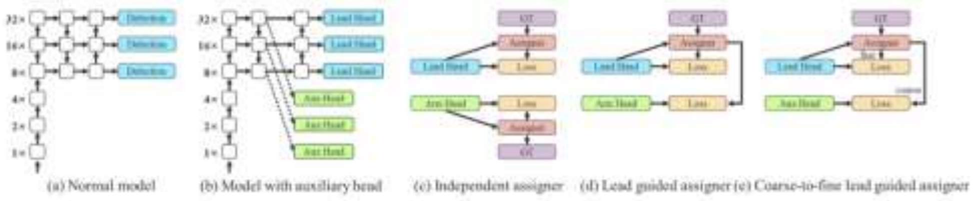


***Figure 41:*** *object detector architecture "without" and "with" deep supervision states*

In the past, label assignment during deep network training often made direct reference to the ground truth and produced hard labels in accordance with the prescribed criteria. However, if we use object identification as an example, researchers now frequently include the quality and distribution of the network's prediction output in addition to the ground truth when using various calculation and optimization approaches to produce a trustworthy soft label. How can you, however, attach a soft label to the lead head and auxiliary head? To create coarse-to-fine hierarchical labels for auxiliary head and lead head learning, respectively, YOLOv7 uses lead head prediction as guidance. The accompanying graphic displays the two suggested deep supervision label assignment mechanisms.

**Assigned lead head directed label**: The lead head will be better able to concentrate on learning residual information that has not yet been learnt by having the shallower auxiliary head directly absorb the information that the lead head has learned**.**

**Assigned coarse-to-fine lead head guided label:** This technique makes the optimizable upper bound of the fine label consistently higher than the coarse label and enables dynamic adjustment of the relevance of the fine label and coarse label throughout the learning process**.**

### 6.3.6. YOLO Use Cases

#### In Medicine (Melanoma Detection Using Yolo v1, Yolo v2, and Yolo v3)

In the past three years, deep convolutional neural networks (DCNNs) have achieved promising results in skin cancer detection. However, improving the accuracy and efficiency of automatic skin cancer detection remains urgent due to the visual similarity of benign and malignant dermatoscopic images.
Deep convolutional neural networks (DCNNs) have shown encouraging results in skin cancer diagnosis during the last three years. However, due to the visual resemblance between benign and malignant dermatoscopic images, it is still necessary to increase the accuracy and effectiveness of automatic skin cancer detection.

Additionally, there is a need for quick, computationally efficient mobile applications that are aimed towards caretakers and homes. Only one You Look method (Yolo), which is based on DCNNs and used to identify skin cancer. The methods used by Yolo, such as YoloV1, YoloV2, and YoloV3, resize the input image first before dividing it into a number of cells.

Skin disorders seriously endanger the health of patients, and skin illnesses continue to be a leading cause of disability globally. In 2013, the global burden of 306 illnesses and injuries— measured in disability-adjusted life years (DALYs)—was made up of 1.79% by skin problems. The prevalence of skin diseases is influenced by both geographic and age-related factors, with melanoma being the most frequent diagnosis in areas with abundant resources like Australia, New Zealand, Northern Europe, and North America. Untreated malignant melanoma can be lethal if it spreads to other body organs and is a very aggressive cancer. Early melanoma detection usually allows for surgical removal of the entire tumor. Deep learning has recently been extremely important in the early identification of cancer. This study employ deep convolutional neural networks (CNNs) to identify melanoma.

The researchers came across three key detector families for object detection using deep learning: RCNN, SSD, and Yolo. You Only You Look One (Yolo) series, Single Shot Detector (SSD), and CNN Region (R-CNN). All of these methods approach object detection as a regression issue, learning the bounding box (BBox) coordinates and associated class label probabilities simultaneously from a given image. R-CNN and its variations, which include the original R-CNN, Fast R-CNN, and Faster R-CNN, are two-stage detectors and are typically quite accurate but incredibly slow. SSD provides a nice balance between precision and speed.

However, the goal of this work is to create a mobile melanoma detector in the future, and for this application, speed is essential. Yolo algorithm, a single-stage detector, is the best option for this use. Yolo is faster than R-CNN but less accurate. Yolo unquestionably has some shortcomings and restrictions. For instance, It struggles with little items and works poorly when objects are gathered closely together. However, these behaviors are hardly observed throughout the melanoma detection process.

## METHODS

Yolo is simply a unified detection model that uses the entire image as the network input and divides it into a s×s grid without the usage of a convoluted processing pipeline. The model immediately outputs the location of the object border and the appropriate category in the output layer following the selection from the network, however, is ineffective at picking up small populations and close objects. YoloV2 and YoloV3 adapted certain concepts from SSD and Faster R-CNN to address these issues in the original YoloV1 network. Both the speed and accuracy of the detection increase significantly as a result of these integrated methods. YoloV2 and YoloV3 are better versions of YoloV1 that weren't developed with sophisticated networks but instead incorporated a wide range of concepts from other deep learning models and how they were applied, including batch normalization, anchor boxes, and fine-tuning with high-resolution classifiers.

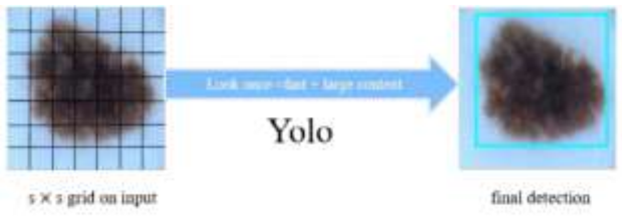


***Figure 42: The structure of yolo detection with sxs grid input image, after yolo selected , it find the final location of the melanoma***

### A. YoloV1

YoloV1 converts the target detection issue into a regression problem that extracts BBoxes and class probabilities straight from the image using a single convolutional neural network. Three parts make up the complete inspection process: scaling the image to a resolution of 448 448; detection and classification using grid cells; and Non-Maximum Suppression (NMS), which is used to eliminate redundant detection possibilities. Yolo divides the image into s s grid cells specifically in the first step. The grid is in charge of identifying targets when their centers fall within it. As shown in Figure 3, each grid cell predicts two BBoxes, one box confidence score, and two conditional class probabilities (a). Five values are contained in each BBox: the target's confidence value, the coordinates x, y, w, and h. The relevant cell's offsets are indicated by the x and y coordinates. The BBox width and height are indicated by the w and h coordinates, respectively. Yolo has two completely connected layers after 24 convolutional layers.

### B. YoloV2

The YoloV2 deep learning framework, an updated version of YoloV1, is advised for modeling the architecture and training the model. For the multi-layer convolution and pooling processes, YoloV2 changes the initial 77 grid to a 1313 grid. The larger grid leads to a denser grid and a higher BBox density. When a picture contains a large number of items, especially little ones, the extended network can increase the number of retrieved objects. This increases the detection's precision. But computation and complexity of the model will both increase. The correct grid size must be chosen in order to achieve a good balance between accuracy and detection speed. The Darknet-19 model's final convolution layer is changed to three 3×3 convolution layers and 1024 channels for YoloV2 detection network to extract features from. To compress the information and quicken the process, an 1×1 convolution layer is applied after each convolution. Convolution doesn't need to be molded, hence the spatial information is effectively retained. YoloV2 makes use of the idea of an anchor. The accuracy and speed of the model's BBox position detection are influenced by the sizes and numbers of the anchors. The anchor box foretells the kind of object and its coordinates.

### C. YoloV3

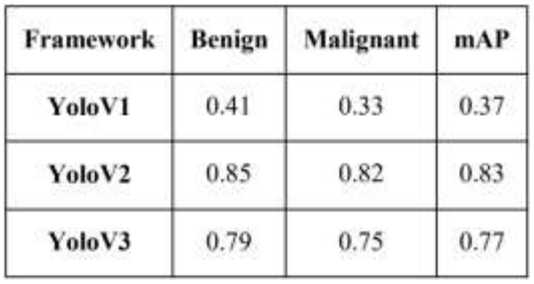
YoloV3 proposes a new deeper and larger feature extraction network named Darknet-53, which contains nine anchor boxes, by incorporating Darknet-19 merges from YoloV2.

### D. Dataset description

This study used a dataset that it created on melanoma detection. The ISIC dataset, a widely accessible database of skin diseases, served as the primary source of data. The Yolo detector requires explicitly labeled data because it is a supervised deep learning system. To maintain balanced classifications, only 200 photos were chosen, of which half were benign and half were malignant. There were 160 training images, 20 validation images, and the final 20 testing images.

## RESULTS

***Figure 42*** compares the performance of the three versions of Yolo. YoloV3 has a mAP of 0.770, which is 0.064 lower than YoloV2, but YoloV1 has the lowest mAP at 0.371. As a result, the proposed YoloV2 better balances the interaction between the problem of network overfitting and feature representation capacity. YoloV3 is a model that is suggested for small item detection because it performs poorly on large objects.



| Framework | Benign | Malignant | mAP |
|---|---|---|---|
| **YoloV1** | 0.41 | 0.33 | 0.37 |
| **YoloV2** | 0.85 | 0.82 | 0.83 |
| **YoloV3** | 0.79 | 0.75 | 0.77 |

***Figure 43:*** *Melanoma MEAN AVERAGE PRECISION (MAP) COMPARISON OF YOLO VERSIONS*

## CONCLUSION

According to the Yolo comparison of three DCNN networks, YoloV2 had the highest classification accuracy, reaching about 83%. Darknet-19 feature extraction and the wider grid that suited the photos in the test dataset helped YoloV2 perform better.

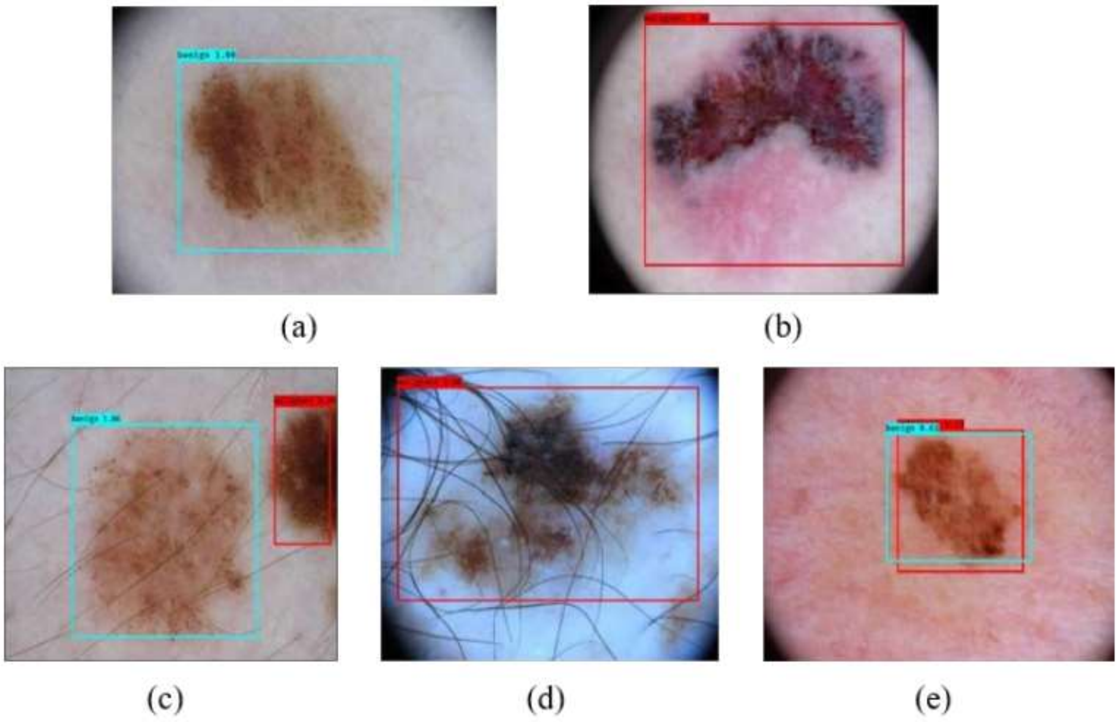


*Figure 44:* *Melanoma YOLO Detection Results*

### In Marine (Marine Object Detection Using Yolo v4)

Marine monitoring is a crucial responsibility today. It is necessary for a variety of tasks like sea surface monitoring, managing marine areas, preventing ship collisions, conducting search and rescue operations, battling illegal immigration, stopping fishing and smuggling, etc.
Technologies for computer vision have been adopted to help with naval surveillance. Marine image processing has covered a wide range of subjects. Marine organism detection and taxonomy, however, remain unresolved issues. Due to the constantly changing background, recurring tides and waves, the existence of small floating objects, and the unpredictable movement of marine objects, the marine environment poses additional difficulties. The lighting is also influenced by the weather (fog, haze, rain, bright sunlight, twilight, etc.) and the angle at which the sun's rays strike the lake surface. The time of day (day, night, sunrise, sunset, etc.), as well as the weather, affect the hue of the sea.

Artificial intelligence (AI) developments offer effective methods for classifying and detecting objects. Deep neural network-based emerging technologies have successfully solved previously intractable computer vision issues. This software uses the advantages of top view scenes taken by satellites or drones to monitor aquatic species with the help of built-in artificial intelligence.

An overview of the discovery-assisted method using best-vision sights is shown in Figure 1. You Only Look Once (YOLO), a recently developed object detection technique, focuses on finding marine life utilizing photos that offer top-down perspectives. YOLO was initially presented as an effective, all-encompassing approach for real-time object identification. There are various YOLO variants available with various network architectures and characteristics. They utilized YOLOv4 and the 2020-introduced YOLOv4 Small Formats in this implementation. The YOLOv4 model is enhanced over earlier iterations thanks to the use of CSPDarknet-53, which uses residual connectivity as its backbone. The short YOLOv4 form is a condensed version of the full YOLOv4 form and has two detection headers as opposed to three.

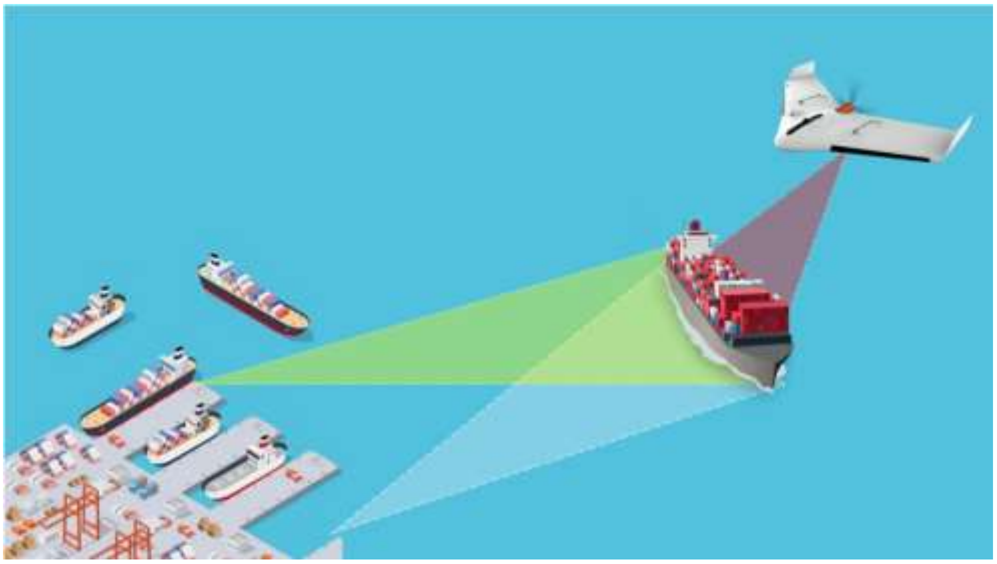


***Figure 45:*** *Overview of the marine proposed approach*

**METHODS**

**A. Collected Dataset**

A sizable collection of photos displaying aquatic things from above has been compiled. Two classes are the main subjects of this essay: (1) seagoing vessels and (2) people. Other aquatic objects like jet skis and floatable buoys can be seen in the gathered pictures. The gathered photographs were either taken by drones or satellites. They also make use of publicly accessible datasets. All of the gathered photographs have been edited. Images that lack annotations are manually labeled. Existing labels are upgraded by converting them to YOLO format and changing the bounding boxes to match the precise measurements of the intended marine objects. Examples of the finished dataset's photos are shown in ***Figure 47*** The pictures depict aquatic things from various angles and sizes. Altitudes, camera shooting angles, and lighting all vary amongst the photographs. Negative samples from the dataset that lack tagged objects are included. These examples are used to make sure that the target classes are not mistakenly assigned to other objects in the images (boats and humans). The

| Model | Number of Layers | Activation function | Model Weights' Volume (MB) |
|---|---|---|---|
| YOLOv4 | 162 | Mish | 256.2 |
| YOLOv4 Tiny | 38 | Mish | 23.5 |

obtained dataset is divided into 90% for training and 10% for validation. The details of the generated dataset are shown in ***Figure 46*** It displays the quantity of photos and the breakdown of annotations by class.

***Figure 46:*** *SPECIFICATIONS OF THE TARGETED YOLOV4 MODELS*

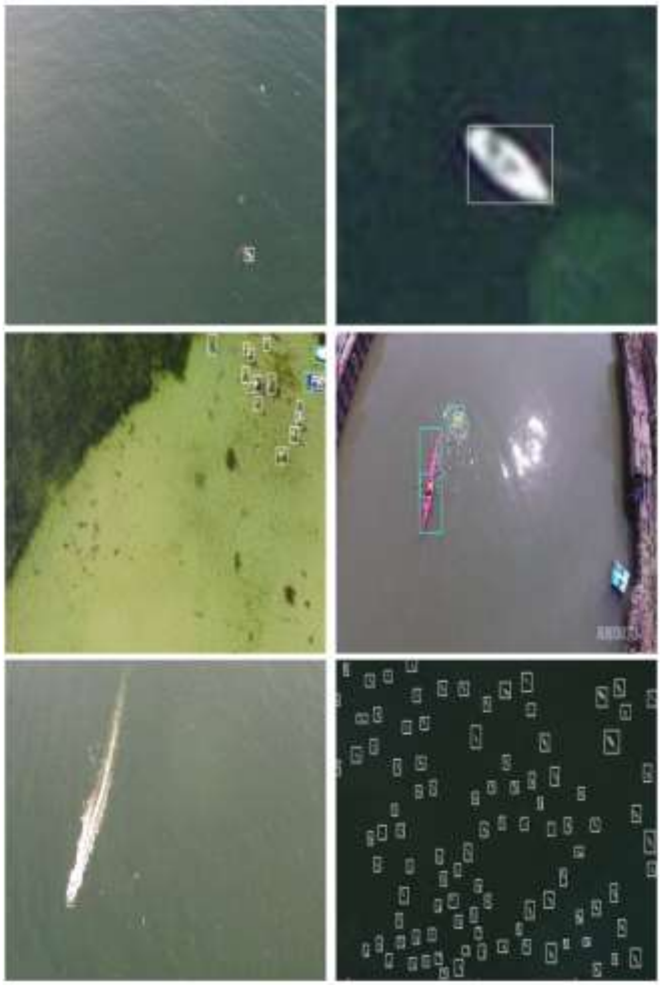


***Figure 47:*** *Sample of images from the marine dataset*

**B. Training and Validation**

They specifically addressed YOLOv4 and YOLOv4 small models in the paper. The Darknet framework, an open-source neural network framework created in C and CUDA, is used to rain the target models. It supports both CPU and GPU computing, is quick, and is simple to install. It can also categorize pictures for the 1000-class ImageNet challenge.

Using the generated dataset, pre-trained YOLO models are modified for the target classes (Boats and Humans). Transfer learning is a machine learning technique that allows users to solve other issues that are comparable using the knowledge they have learned while training a set of problems.

The validation dataset is used to check the trained model during training. For each of the four epochs, the mean average precision (mAP) is determined starting with the first 1000 iterations. The training and validation results of the YOLOv4 model are shown in ***Figure 48*** The red curve represents the computed mAP values, whereas the blue curve represents the training loss.

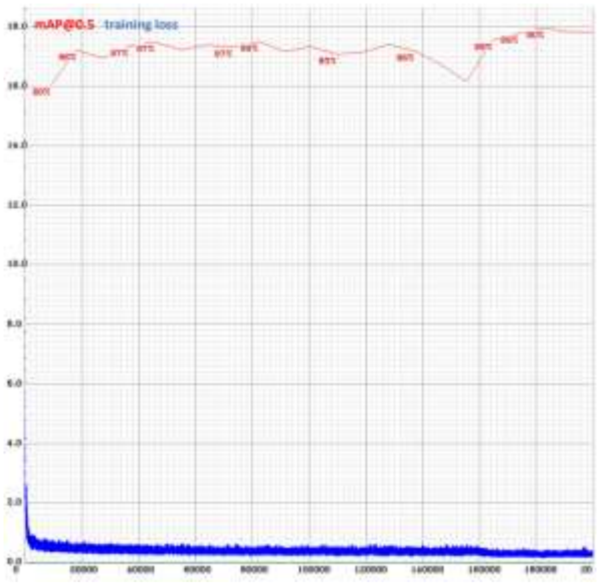


***Figure 48:*** *Training and validation performances of YOLOv4 model*

**RESULTS**

The trained models are tested using a dataset that includes images that are not seen before. The test dataset includes 1114 images with 1672 annotations. Fig. 4 presents sample detection results. The figure shows that trained models were able to accurately detect and classify the presence of marine objects with high confidence ratio in different maritime environments.

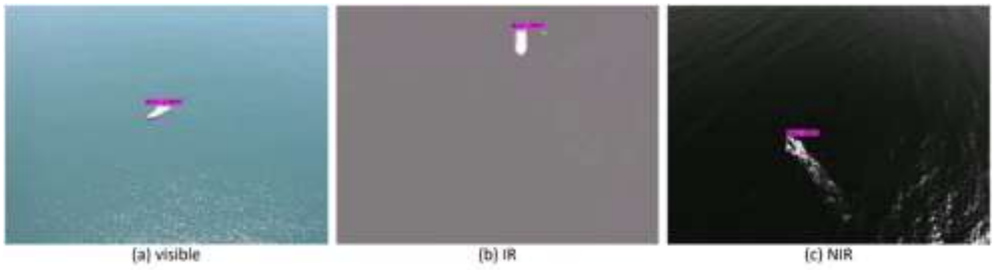


*Figure 49: Samples of the obtained detection results in video*

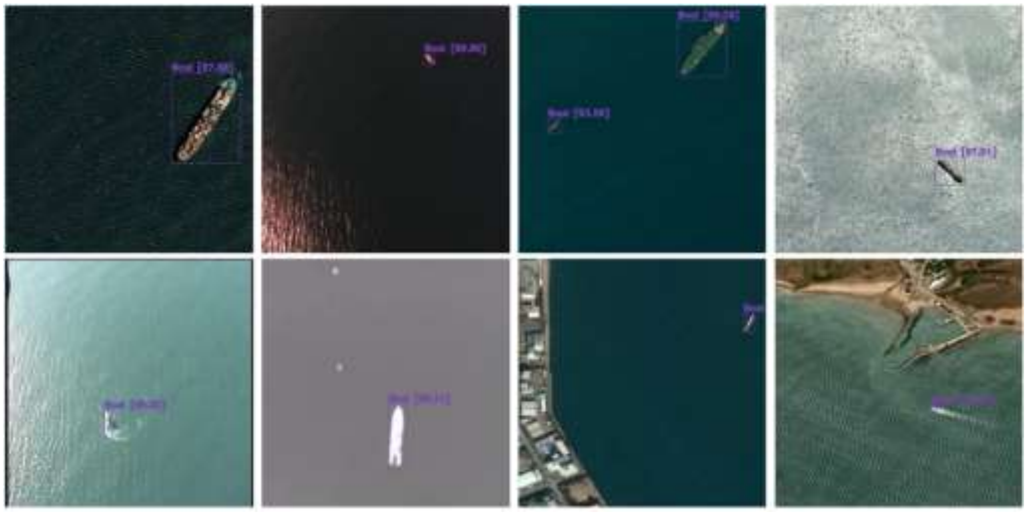


***Figure 50:*** *Sample detection results in testing dataset images*

## CONCLUSION

The research introduces a novel dataset that has the greatest number of photos of people and floating boats taken from above. It is described how the target models were trained, validated, and evaluated. The findings obtained demonstrate that YOLOv4 models are capable of 90% accuracy and a mAP value of 0.89.

### In Production (Weed Detection Using Yolo v3, Yolo v4, and Yolo v5)

One of the biggest risks to the production of cotton is weeds. Herbicide resistance in weeds has been increased by the overuse of herbicides for weed control, raising worries about the environment, the safety of food, and human health. The integrated and sustainable control of weeds has drawn a lot of interest to machine vision systems for automated or robotic weeding. However, creating reliable techniques for the identification and detection of weeds remains a difficult issue due to the unregulated field settings and significant weed biodiversity. The lack of specialized, extensive image data sets for crop-producing (cotton) weeds and the lack of machine learning algorithms for weed detection are the two main barriers to solving this problem. In this study, a new dataset of weeds significant to US cotton production systems is presented. This dataset consists of 5,648 images from 12 weed categories with a total of 9,370 square annotations, all of which were collected in cotton fields under natural light conditions and at various stages of weed growth. In order to find weeds in the dataset, a thorough benchmark collection of 18 advanced YOLO object detectors, including YOLOv3, YOLOv4, Scaled-YOLOv4, and YOLOv5, was developed. By YOLOv3-tiny, detection accuracy for mAP@50 varied from 88.14% to 95.22%, and by YOLOv4, detection accuracy for mAP@50 [0.5:0.95] ranged from 68.18% to 89.72%. All demonstrated YOLO models, particularly YOLOv5n and YOLOv5s, offer excellent promise for real-time weed detection, and data augmentation can improve weed detection accuracy.

In the United States, cotton is a significant row crop commercially. 10.0 million acres of cotton worth around $7.5 billion were harvested in the United States in 2021. By competing with crops for resources (such as water and nutrients) and acting as hosts for diseases and pests, weeds pose a threat to the production of cotton. If weeds are not controlled, poor weed management can result in yield losses of up to 90%. By 2020, 96% of the cotton-growing land in the United States will have been planted with HR seeds since trans-genetic, herbicide- resistant (HR) crops were first made available for commercial use in 1996. Herbicides are currently mostly used in broadcast applications for cotton weed control. Concerns regarding the environment, food safety, and human health have also arisen as a result of the overuse and repetitive application of herbicides. Therefore, to develop a more sustainable weed management system, the U.S. cotton industry must enhance weed control methods.

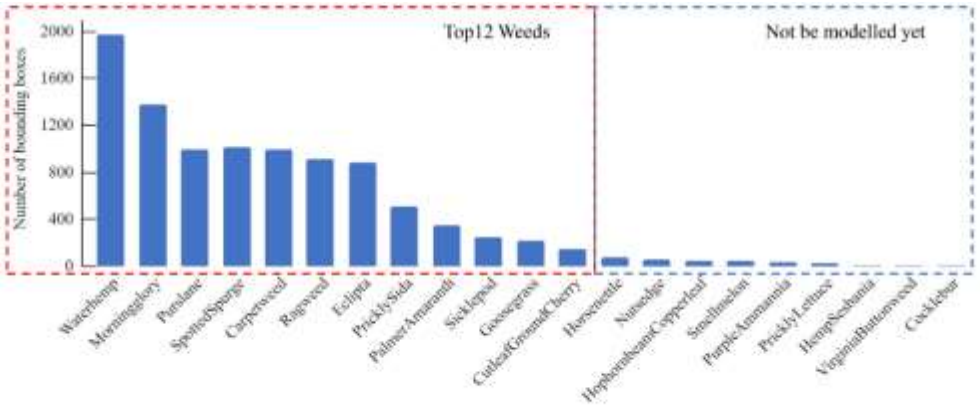
Future weed management strategies that incorporate a range of control techniques will require sustainable and integrated management of weeds, and machine vision technology will be essential to this. Machine vision systems are able to differentiate between weeds and crops, identify weed species, and provide precise and even targeted weed control using imaging sensors and computer algorithms. For in-season weed control of crops like cotton, there is growing interest in creating robotic systems based on machine vision. Once weeds have been identified and located, the proper weed removal techniques can be used, such as thermal procedures or topical spraying of flowing herbicides. All of these weeding techniques dramatically minimize the need for herbicides, which is essential to reduce weed herbicide resistance. For machine vision-based selective weed removal to be effective while inflicting the least amount of crop harm, accurate weed recognition and detection are essential.
Commercial weeders that recognize crop row patterns are available. This weed is not ideal for in-row weed control since it does not distinguish between crops and weeds for precise handling. It is notoriously difficult to distinguish between crops and weeds and to determine their species. This has been explained by the significant biological diversity of weeds and crops as well as the unmanaged field conditions. The practical implementation of machine vision weeding systems is hampered by the fact that modern machine vision systems still are unable to attain weed recognition rates that are substantially above 95%. The number and quality of the image data used to train the model as well as the weed recognition algorithms are the two key variables that have a substantial impact on weed detection performance.

## METHODS

### A. Collected Dataset

The weed dataset for this study was gathered from cotton fields at Mississippi State University research farms utilizing several hand-held smartphones and digital color cameras (with a resolution of more than 10 megapixels). Images were taken in various weather situations (such as sunny, cloudy, and overcast) using natural field lighting at various stages .of weed development in various field locations

Trained staff members who create bounding boxes for marijuana instances in images categorized the gathered photographs. Instead of scientific nomenclature, the weed specimens were identified by their vernacular names. The same weed class classification was given to several weed species that belong to the same plant genus or family and have similar weed management practices. In the end, this produced a dataset with 5960 weed photos from 21 different weed classifications and 9898 bounding box annotations. Only the top 12 weed classes—representing 5648 photos with 9370 bounding box annotations—were taken into consideration in this study



because to the uneven distribution of weed data, as seen in ***Figure 51*** (which contains more than 140 bounding boxes per class).

***Figure 51:*** *Bar plot of the cotton weed detection dataset consisting of 5960 images of 21 weed classes with 9898 bounding boxes. Because of weed class imbalance, only the top 12 weed classes with 9370 bounding boxes are used for weed detection in this study*

## B. YOLO MODELS

For constructing weed identification models for the cotton weed dataset, the five types of YOLO object detectors—YOLOv3, YOLOv4, Scaled-YOLOv4, YOLOR, and YOLOv5— were chosen in this study. Although earlier studies tested a few of these weed detectors, they only evaluated a single or a pair of YOLO models for a single or a very small number of cannabis classes. There hasn't been any investigation into the effectiveness of a variety of YOLO detectors for identifying multi-class weeds important to cotton production.

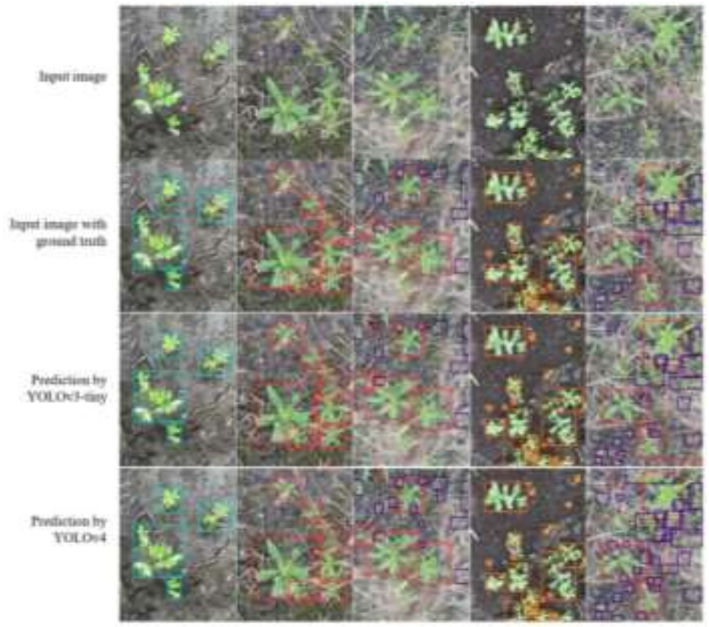


*Figure 52: Examples of weed images with predicted bounding boxes. The bounding boxes with the same color represent the same weed class. The green, red, purple, and orange bounding boxes represent the detection of Purslane, Waterhemp, Spotted Spurge, and Carpetweed, respectively.*

## RESULTS

The training curves for the top 10 detectors in terms of weed identification accuracy are shown in Figure 5 as mAP@0.5 and mAP@[0.5:0.95]. The training curves for other models are not displayed here because to space restrictions. With 90% mAP@0.5 and 80% mAP@[0.5:0.95] obtained within 100 training epochs, these models demonstrated overall promising training performance in terms of quick convergence and high detection accuracies. Training revealed that accuracy levels for all models (including those not shown in ***Figure 53***) peaked at 200 epochs, supporting the finding that training for 300 epochs was adequate for this study.

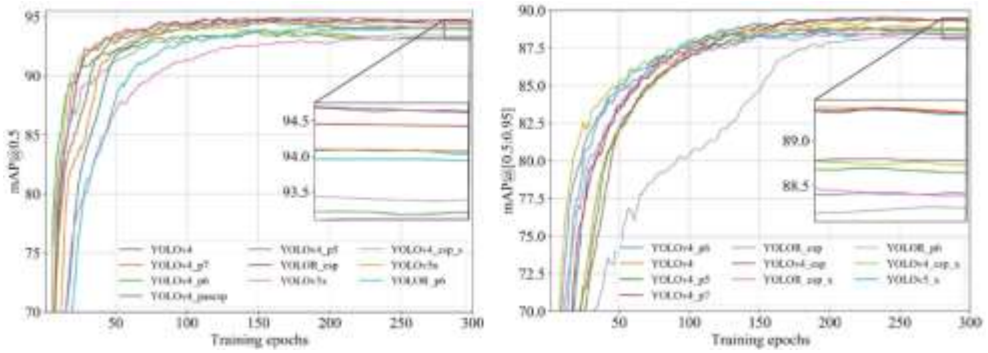


***Figure 53**: Training curves of mAP@0.5 and mAP@[0.5:0.95] for the top 10 YOLO models for weed detection.*

## CONCLUSION

Scores for YOLO discovery models ranged from 88.14% for YOLOv3-tiny to 95.22% for YOLOv4. Overall, YOLOv4-based detection models were more accurate than YOLOv3 and YOLOv5 models, whereas YOLOv5n and YOLOv5s models shown advantages in quick inference times while attaining equivalent detection accuracy. Furthermore, the performance of YOLO detectors like YOLOv3-tiny improved with more data.

### 6.3.7. Other Models and comparisons

**R-CNN**

"Rich feature hierarchies for accurate object detection and semantic segmentation" This paper examines the use of high-order feature hierarchies in order to detect objects accurately as well as segment them semantically. Known as R-CNN, this method was developed by them.

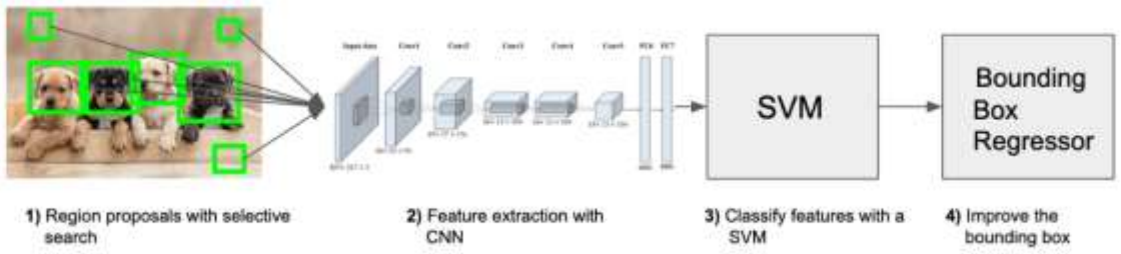

SVM
Bounding Box Regressor
1) Region proposals with selective search
2) Feature extraction with CNN
3) Classify features with a SVM
4) Improve the bounding box

*Figure 54: R-CNN system*

**The R-CNN system**

The sliding window technique was being used to solve the problem of object detection. Using a sliding window, we can detect a feature under an image or an object under an image.

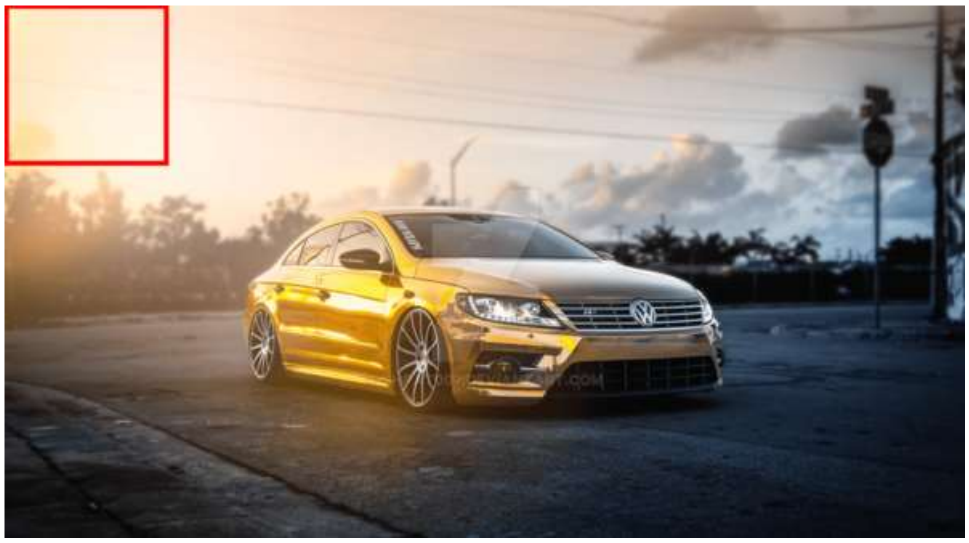


*Figure 55: Sliding window in a car image*

Since an image is a 2-dimensional array, we have to slide the window horizontally and vertically. A convolutional neural network classifies each sub-image contained in a window according to whether it contains an object of interest. The problem is you will have a giant number of smaller images to look at [29].

R-CNN extracts information about a region of interest from an input image by using a mechanism called selective search. Region of interest can be represented by the rectangle boundaries. The number of regions of interest can exceed 2000 depending on the scenario. CNN produces output features from this region of interest. SVM (support vector machine) classifiers use these output features to classify objects.

There are various issues with the R-CNN model:

- In order to later train the SVMs, it stores the pre-trained CNN's extracted features on disk. Numerous hundred Gigabytes of storage are needed for this.
- R-CNN relies on the Selective Search method, which is time-consuming, to generate region recommendations. Additionally, this approach cannot be altered to fit the detection issue.
- The CNN is supplied individually with each area suggestion for feature extraction. R-CNN cannot be operated in real time as a result [30].

## Feature Descriptor for the Histogram of Oriented Gradients (HOG)

Feature descriptors take an image and compute feature descriptors/vectors. As a result, these features can be compared using their numerical "fingerprints." The Histogram of Oriented Gradients(HOG) algorithm counts the occurrences of gradient orientation in localized portions of an image. The HOG algorithm generates the image gradient along the x-axis and y-axis for each pixel in each cell after dividing the image into tiny connected sections called cells.

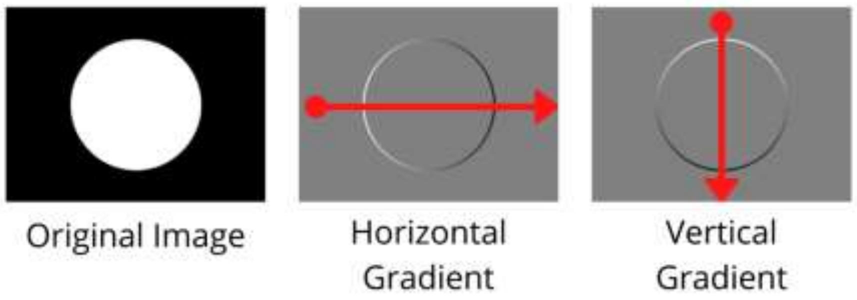


***Figure 56**: HOG*

Pixel-wise gradients visualized

These gradient vectors are mapped from 0 to 255, and pixels with negative changes are black, while those with significant positive changes are also black, and those with no changes are grey. By conducting vector addition on these two numbers, the final gradient is determined.

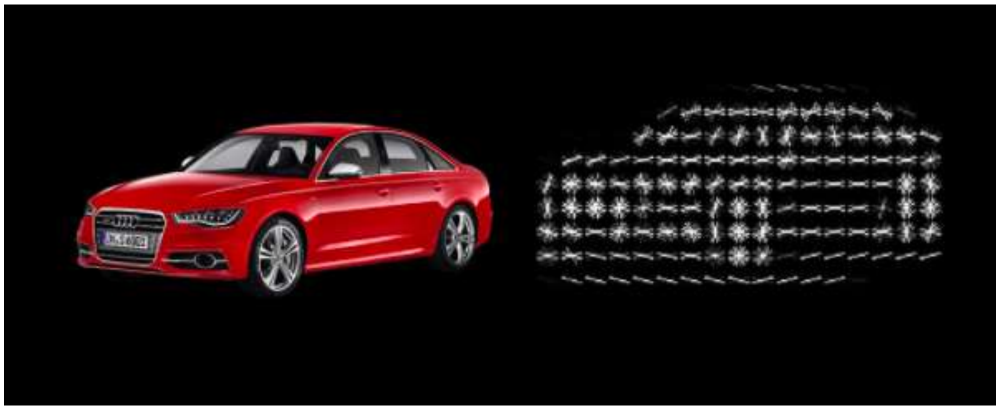


*Figure 2:HOG features for a car*

Using 8x8 pixel-sized cells, each cell is divided into angular bins after the final gradient direction and magnitude for the 64 pixels have been determined. There are nine 20-degree bins, one for each of the gradient's directions (0–180). The Histogram of Oriented Gradients (HOG) technique is able to do this and condense 64 vectors down to just 9 values. To accomplish object detection, HOG is typically used with classification methods like Support Vector Machines (SVM) [31].

**The best real-time object detection algorithm (Accuracy)**

On the MS COCO dataset and based on the Average Precision (AP), the best real-time object detection algorithm in 2022 is YOLOv7,

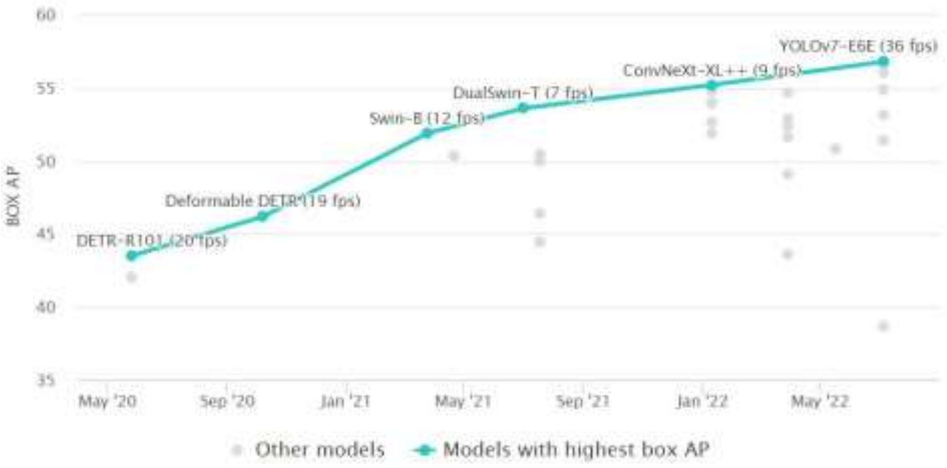


*Figure 58: best real-time object detection algorithm in 2022*

## The fastest real-time object detection algorithm (Inference time)

Inference time (ms/Frame, lower is preferable) or frames per second is a crucial benchmark parameter (FPS, higher is better). When examining inference time comparisons, the tremendous advancements in computer vision technologies are clearly discernible [32].

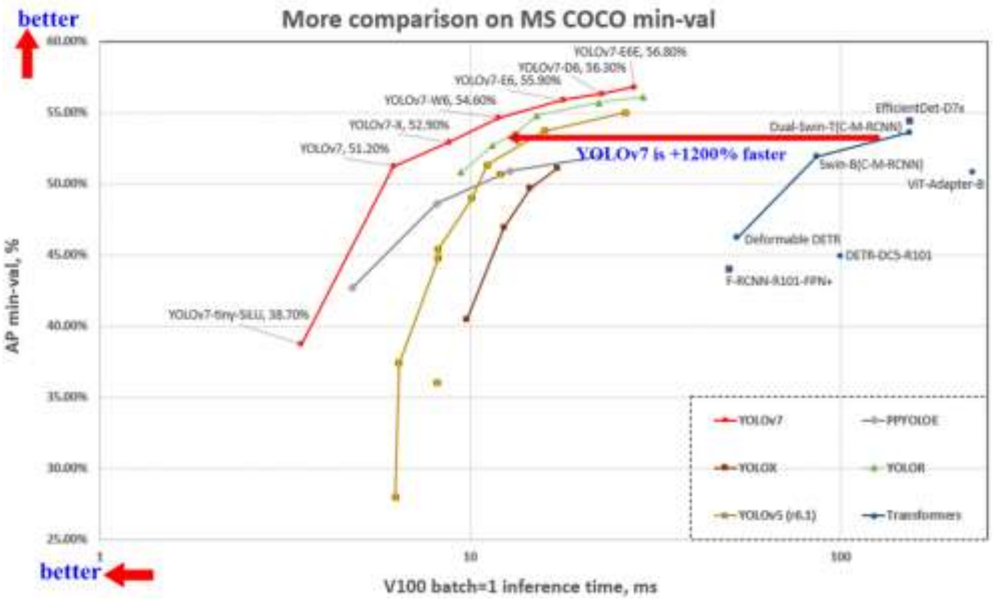


*Figure 59: The fastest real-time object detection algorithm (Inference time)*

### 6.3.8. Dataset for JAZ

For JAZ detection model, we needed a vehicle dataset for the detection model to be able to train YOLO on a custom dataset. After much research for dataset that is best suitable for our application where we need to detect different type of vehicle in order for our scheduling algorithm to give better result and use the helpful information captured from the camera, we found a published vehicle dataset from Maryam Boneh on GitHub [33]. The images' size in the dataset are 640. The images are assigned to training, validating, testing sets in a ratio of 70:20:10 for preparing object detection model using YOLO.

**The Training Set**

The data set is what teaches the model how to find hidden patterns or features in the data. The neural network architecture continuously learns the features of the training data by feeding it the same data every epoch. In order to train the model in all scenarios and be able to predict any unseen data sample, the training set should contain a diverse set of inputs.

**The Validation Set**

To validate our model performance during training, we use the validation set, a separate data set from the training set. Through this validation process, we can adjust hyperparameters and configurations to fit the model's needs. This is the equivalent of a critic telling us whether the training is going in the right direction. Each epoch, a model evaluation is performed on the validation set in parallel with the model training on the training set. In order to avoid overfitting, we divide the dataset into a validation set to prevent the model from becoming very good at classifying the samples in the training set, but not able to generalize to new data.

**The Test Set**

After the model has been trained, it is tested on a separate set of data. In terms of accuracy, precision, etc., it provides an unbiased metric for assessing the final model performance. The model's performance is determined by how well it answers the question, "How well does it perform?"

The dataset represents the 5 main type of vehicles ('Car', 'Motorcycle', 'Truck', 'Bus', 'Bicycle'), that we care about for our scheduling algorithm.

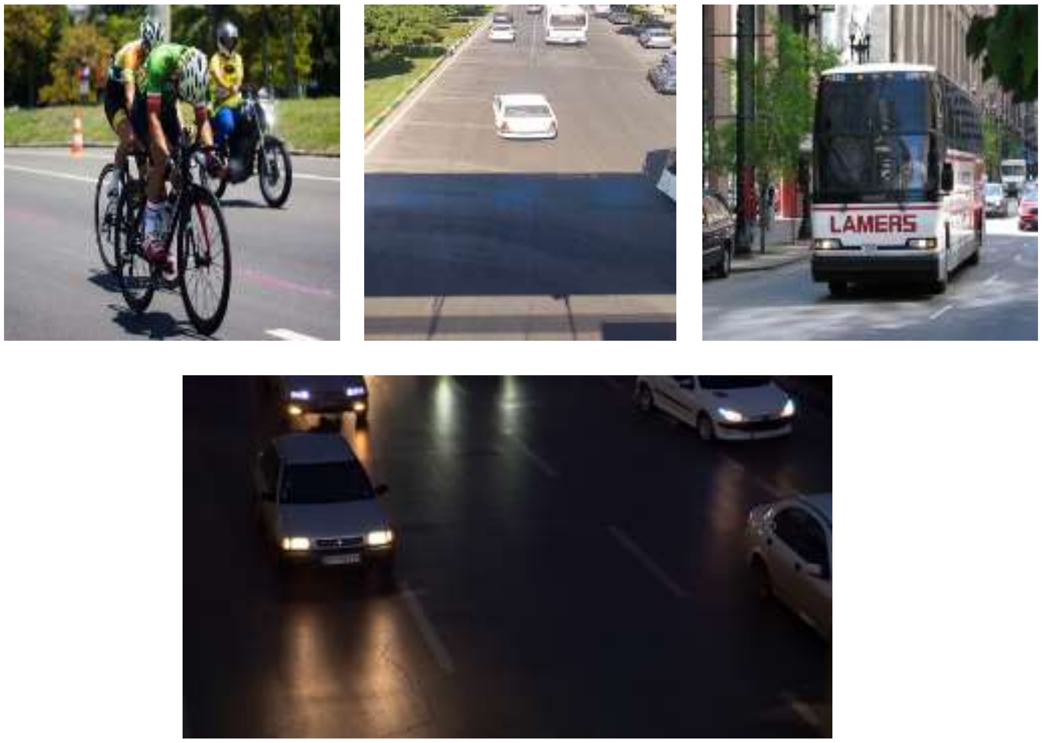


*Figure 60: Dataset classes*

We uploaded the dataset in roboflow to be able to see and take a look at the images and be able to know the number of images in the dataset and also to know the number of images that has a specific class label. There were 1251 images in the dataset, 886 in the training set, 243 for the validation set, and 122 for the testing. The number of images that contain car label is 926, and 189 for the bus label, the “2” was annotated in 113 images, and 146 images contain “3”, lastly, 58 images have bicycle label in it. And finally, the data format is exported as YOLO format. The YOLO format involves a single text file for each image in the dataset. No text file exists for an image if it does not contain objects. The following information is included in each row of the text file: (class_id, x_centre, y_centre, width, height).

- Both image files in the images folder and its relative text file in the labels folder must have the same filename.
  For example:images→0001.jpg, labels→ 0001.txt

- Each text file must fulfill all the properties of the YOLO format text file which are the following: **1**. The first element of each row is a class id, then bounding box properties (x, y, width, height). **2.** Bounding box properties must be normalized(0–1).
  **3.** (x, y) should be the mid-points of a box [34].

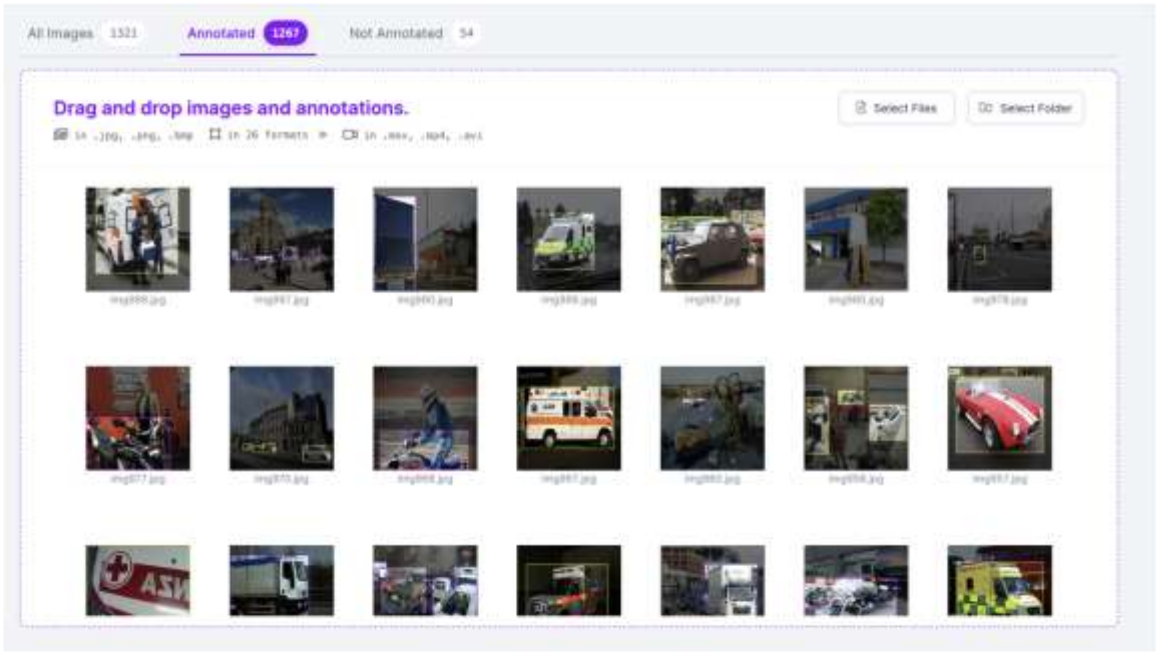


*Figure 61: Dataset in Roboflow*

### 6.3.9. Training and Fine tuning

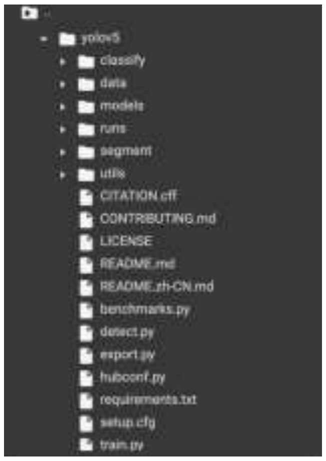
#### 6.3.9.1. Training YOLO model

The training part of the project follows the documentation for training YOLOv5 and v7 using Google Colab's Free GPU Notebook. Prepare the dataset folder by uploading it to the drive. There are a number of different flags that need to be used for training settings, such as:

- ***img-size***: This option is used to specify the image's size in pixels.
- *batch*: The number of image-caption pairings processed during training is represented by the batch size.
- ***epochs***: The number of epochs for which to train. This value represents the total number of rounds the model will make through the whole dataset during training.
- ***data***: set the path to our YAML file.
- ***weights***: Pre-trained model weights from which we will re-start training. If we also want to start from scratch, we can simply write --weights' '.
- ***name***: The path to different training outputs such as train logs. Training weights, logs, and samples of batch images are stored in a subfolder named runs/train/<name>.
- ***cfg***: The model architecture's configuration file.
- ***hyp***: An YAML file describing model hyperparameter selections. The data/hyp.scratch.yaml file contains examples of how to define custom hyperparameters.
- ***workers***: Defines the number of CPU workers assigned to training.
- ***cache***: cache images for faster training [35].

**- Train YOLOv5 on our custom vehicle dataset:**

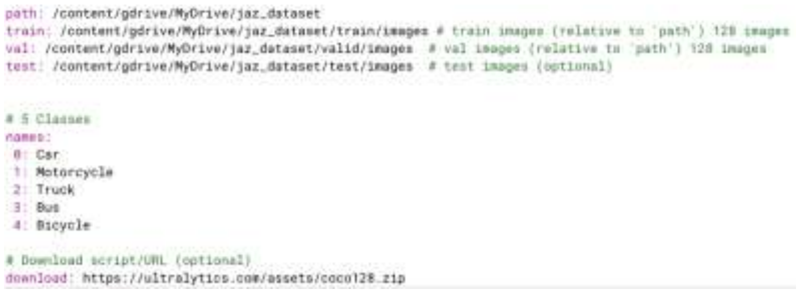
First step after preparing our dataset is to clone the yolov5 repo and install requirements.txt in the Colab.



```
..
yolov5
  classify
  data
  models
  runs
  segment
  utils
  CITATION.cff
  CONTRIBUTING.md
  LICENSE
  README.md
  README.zh-CN.md
  benchmarks.py
  detect.py
  export.py
  hubconf.py
  requirements.txt
  setup.cfg
  train.py
```

***Figure 62:*** *the yolov5 folder after install all the required files*

Then specified our data configuration YAML file was named jazData.yaml. We defined the parameters: train, test, and val. The paths for the images of the train, test, and validation datasets were also written, along with nc, which indicates the number of classes in the dataset, and last names which contain the names of the class labels in the dataset, and then place the the file in the data folder.



```
path: /content/gdrive/MyDrive/jaz_dataset
train: /content/gdrive/MyDrive/jaz_dataset/train/images # train images (relative to 'path') 128 images
val: /content/gdrive/MyDrive/jaz_dataset/valid/images  # val images (relative to 'path') 128 images
test: /content/gdrive/MyDrive/jaz_dataset/test/images  # test images (optional)

# 5 Classes
names:
 0: Car
 1: Motorcycle
 2: Truck
 3: Bus
 4: Bicycle

# Download script/URL (optional)
download: https://ultralytics.com/assets/coco128.zip
```

***Figure 63:*** *jazData.yaml file*

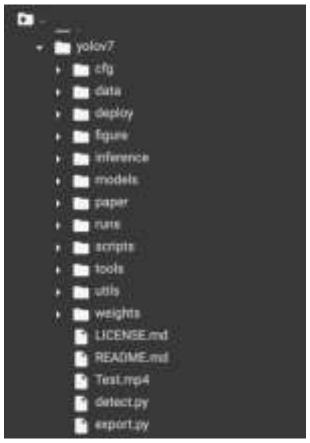
we used the Yolov5 default configuration file, *data/hyp.scratch.yaml*, and also pretrained model weight *yolov5s.pt*.

Once we have completed setting up, we can begin training our model on Colabs Notebook by executing the code cell below [36]:

```
!python train.py --batch-size 3 --epochs 100  --img-size 320 --data data/jazData.yaml --cfg models/yolov5n.yaml --hyp data/hyp.scratch.custom.yaml --weights yolov5s.pt --name yolov5-result
```

**- Train YOLOv7 on our custom vehicle dataset:**

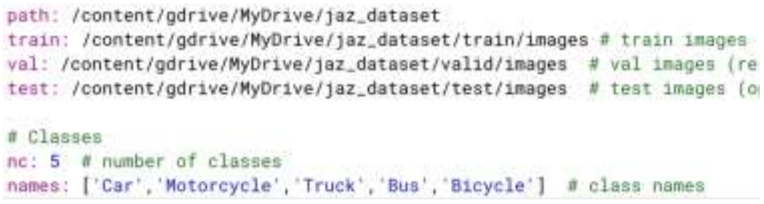
Similar to what we done with Yolov5 we'll train our Yolov7 model, we setup and clone Yolov7 repository and requirements.txt first.



```
..
yolov7
  cfg
  data
  deploy
  figure
  inference
  models
  paper
  runs
  scripts
  tools
  utils
  weights
  LICENSE.md
  README.md
  Test.mp4
  detect.py
  export.py
```

***Figure 64:*** *the yolov7 folder after install all the required files*

After that we edited the data config YAML file to the new paths to the images of the train, test, and val parameters, and then place it inside the data folder.



```
path: /content/gdrive/MyDrive/jaz_dataset
train: /content/gdrive/MyDrive/jaz_dataset/train/images # train images
val: /content/gdrive/MyDrive/jaz_dataset/valid/images  # val images (re
test: /content/gdrive/MyDrive/jaz_dataset/test/images  # test images (o

# Classes
nc: 5  # number of classes
names: ['Car','Motorcycle','Truck','Bus','Bicycle']  # class names
```

***Figure 65:*** *yolov7-dataset.yaml file*

We used the Yolov7 default configuration file, *data/hyp.scratch.yaml*, along with the pre-trained model weight yolov7.pt, and define our own custom *yolov7-dataset.yaml* architecture file and change the number of classes to our classes.

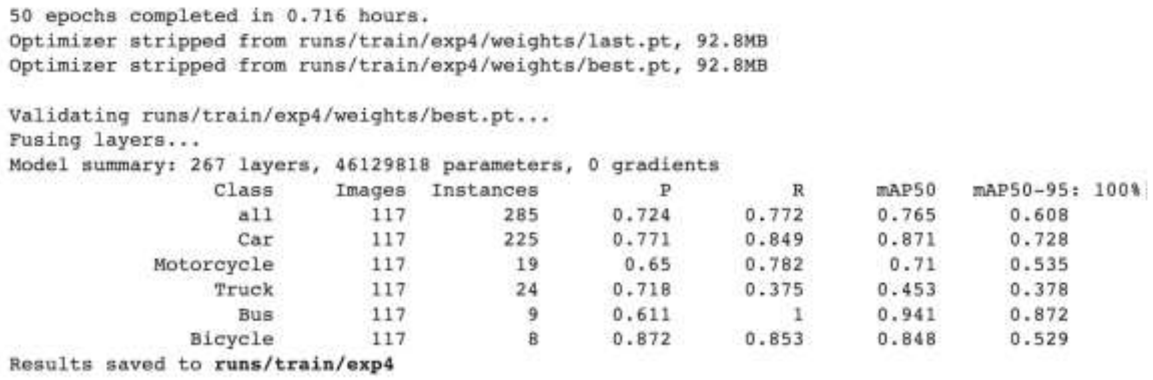
In Colabs Notebook, we can begin training our model by executing the following code cell [37]:

!python train.py --batch-size 3 --epochs 100 --img-size 320 --data data/jazData.yaml --cfg cfg/training/yolov7-custom.yaml --hyp data/hyp.scratch.custom.yaml --weights yolov7.pt --name yolov7-result

### 6.3.9.2. Fine Tuning

#### 6.3.9.2.1. Dataset

After training YOLOv5s model with the default configuration hyperparameters and weights on the above dataset for 50 epochs. We achieved the following result.



```
50 epochs completed in 0.716 hours.
Optimizer stripped from runs/train/exp4/weights/last.pt, 92.8MB
Optimizer stripped from runs/train/exp4/weights/best.pt, 92.8MB

Validating runs/train/exp4/weights/best.pt...
Fusing layers...
Model summary: 267 layers, 46129818 parameters, 0 gradients
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100%
                   all        117        285      0.724      0.772      0.765      0.608
                   Car        117        225      0.771      0.849      0.871      0.728
            Motorcycle        117         19       0.65      0.782       0.71      0.535
                 Truck        117         24      0.718      0.375      0.453      0.378
                   Bus        117          9      0.611          1      0.941      0.872
               Bicycle        117          8      0.872      0.853      0.848      0.529
Results saved to runs/train/exp4
```

***Figure 66:*** *YOLOv5s result with the default configuration for 50 epochs*

From the result above we can see that we got a 0.765 mAP. This Mean Average Precision is a performance metrics in YOLO which we will discuss it in the Evaluation chapter. An important thing we noticed that the Truck class have the least mAP due the small number of truck instance in the dataset.

According to Yolo official GitHub guides to get the best result on the model is to start with the default configuration provided by YOLO. It is always recommended to change and make the dataset sufficiently large and well labeled to achieve good result.

To avoid some of the biases from the dataset provided and the lack of images in both truck and bicycle classes, we added some annotation to these classes (100 images). The new images were annotated using Robowflow and exported the same format for YOLO.

In addition, we enhanced the dataset's photos. The technique of developing new training examples from the ones that already exist is known as image augmentation. You gently alter the original image to create a new sample. Examples of changes to the original image that will result in a new training sample include flipping, cropping, adjusting contrast, and adding noise.
The performance of deep neural networks for computer vision tasks like classification, segmentation, and object recognition is

improved by augmentations, which also serve to enhance the dataset's quantity of pictures and combat overfitting.

Robowflow annotation tools have the feature of augmenting the data you have and gives you the list of all the possible transformation to the images. We did add noises to the image and rotate the images and make gray images, these augmentations help in making the detection model more robust and perform well under various condition [38].

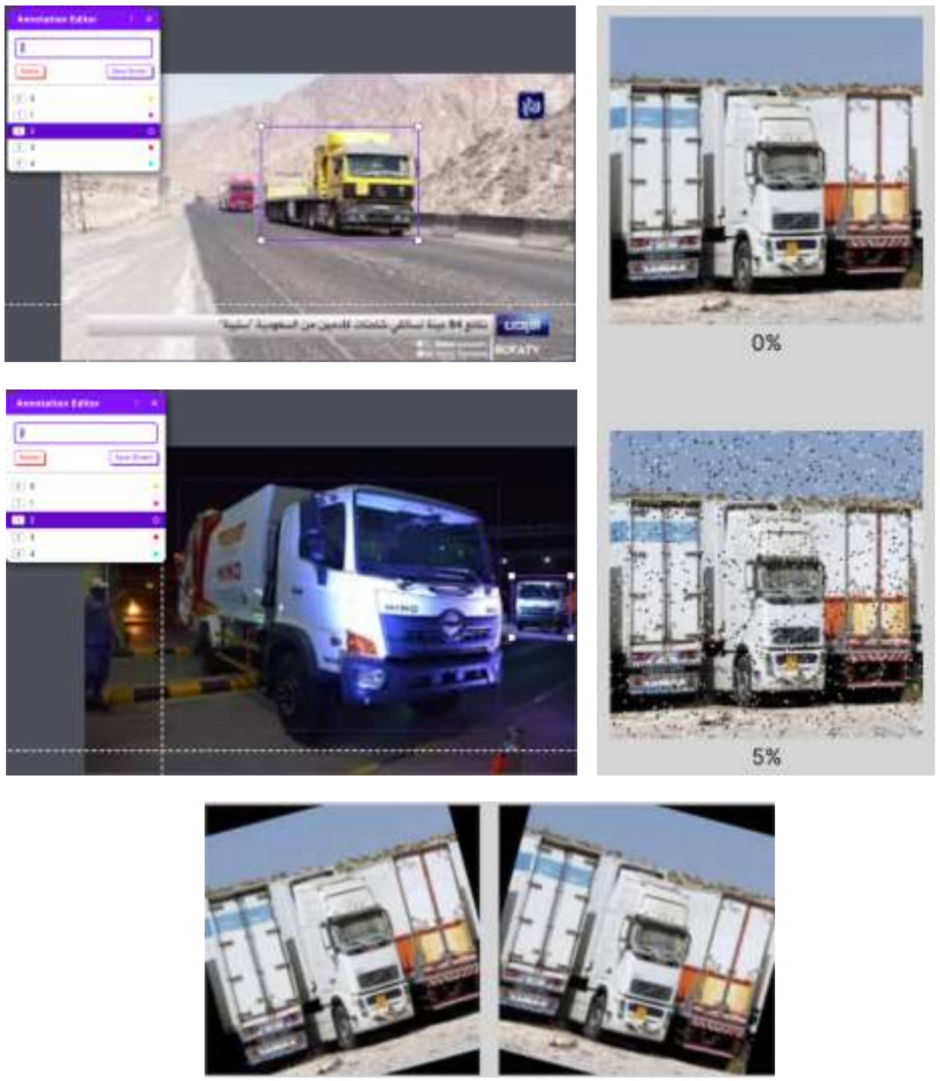


***Figure 67:*** *The new dataset after editing*

We ended up with a new dataset with a total of 9145 images, 6400 images in the training set, 1800 images in the validation set, and 912 testing set images.

#### 6.3.9.2.2. Model Selection

Larger model like YOLOv5x produce better result in all cases but have more parameters and require more CUDA memory to train and slow to run. For small to medium project deployment, it is recommended to use YOLOv5s/m models.

After finetuning the dataset, we selected the **YOLOv5m** and **YOLOv7** model, starting with the pretrained weights. With 640 image size, and 20 epochs due to the high result we got in this range of epoch. The same hyperparameter was used but with the change of the yolov5m.yaml file to take only the 5 classes we are training the model on rather than the 80 classes from the COCO dataset.

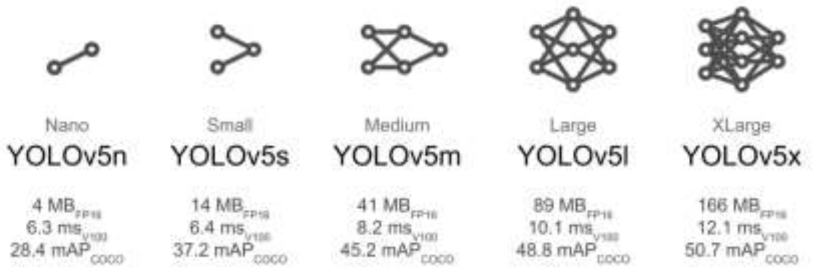


*Figure 68: YOLOv5 versions*

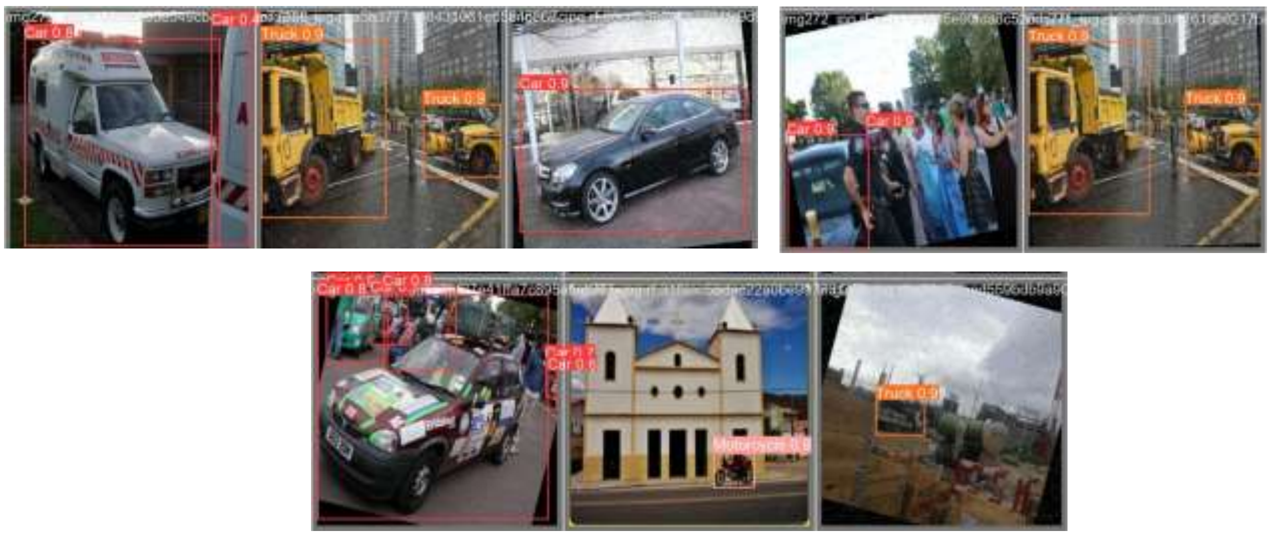
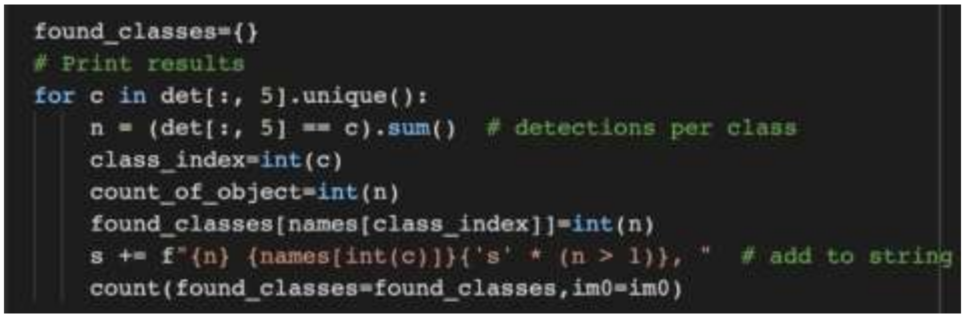


*Figure 69: Models prediction on the validation images*

### 6.3.10. Counting with YOLO

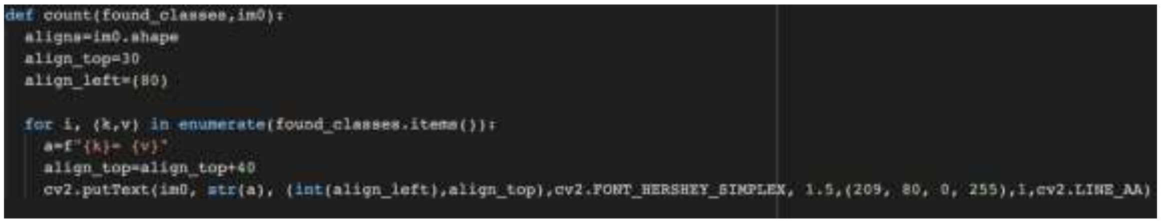
In the detect.py file in the yolov5/7 folder, we are adding a function that do count the number of detection classes and the number of objects per class detected and write it as a text on the image we are detecting on.



```
found_classes={}
# Print results
for c in det[:, 5].unique():
    n = (det[:, 5] == c).sum()  # detections per class
    class_index=int(c)
    count_of_object=int(n)
    found_classes[names[class_index]]=int(n)
    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string
    count(found_classes=found_classes,im0=im0)
```

***Figure 70:*** *detect.py file found classes function*

This is responsible to count the number of classes (n) and the objects in it. The added found_classes dictionary will store the class name and the total number of objects for each class.



```
def count(found_classes,im0):
  aligns=im0.shape
  align_top=30
  align_left=(80)

  for i, (k,v) in enumerate(found_classes.items()):
    a=f"{k}= {v}"
    align_top=align_top+40
    cv2.putText(im0, str(a), (int(align_left),align_top),cv2.FONT_HERSHEY_SIMPLEX, 1.5,(209, 80, 0, 255),1,cv2.LINE_AA)
```

***Figure 71:*** *detect.py file count function*

our count function is going to display the dictionary on the image we are detecting the object from. Using cv2.putText function we are able to write on a given image. We can provide the following arguments to this function:

- image on which you can write the text.
- text you want to write on image.
- position: distance along horizontal and vertical axis from top left corner of the image.
- font family
- font size
- font color
- font stroke width

We are displaying the counter on the top left of the image, so we assigned the alignment to the image shape which is [640,640] image shape. And we are updating the top alignment to avoid text overlapping.

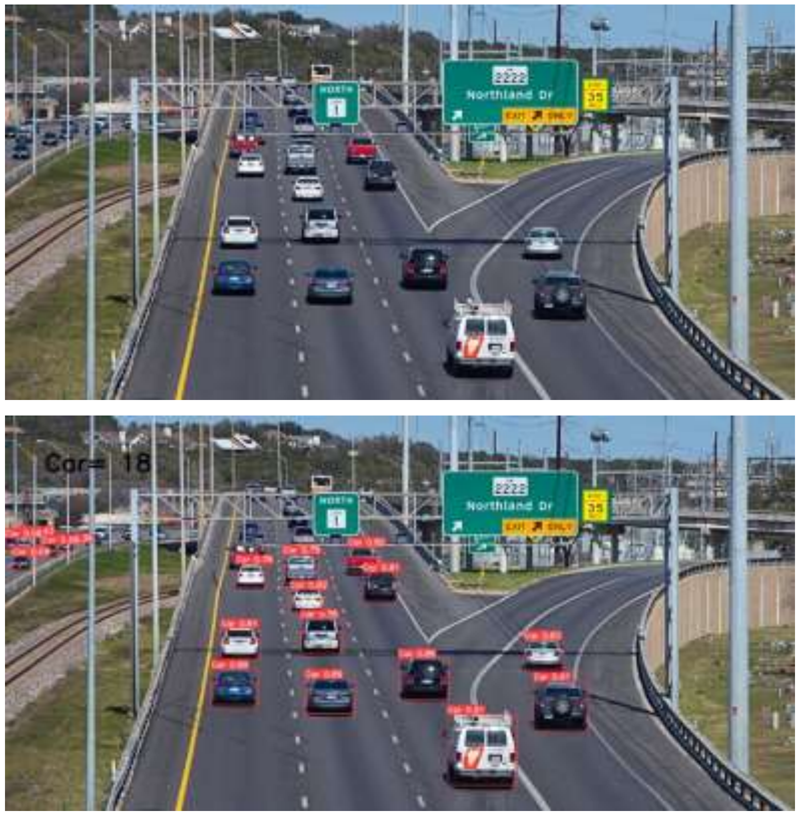


***Figure 72:*** *Counting function result*

## 6.4. Scheduling Algorithm

### 6.4.2. Objectives of Basic Signal Timing Parameters and Settings

During signal timing, distinct procedures are combined into one interrelated procedure management of data, improvement of signal timing, field deployment, and performance assessment.

In addition to cycle length, movement green time, and clearance intervals, signal timing parameters affect intersection efficiency. Traffic movements may be delayed, and vehicles may stop more if the green time is extended. The increase in green time for one movement usually comes at the expense of increased delays and stops for another movement. As a result, a good signal timing plan allocates time according to the demands at the intersection while keeping cycle lengths to a minimum. Timing settings are implemented at an intersection by the traffic signal controller. It is designed to meet the objectives of the responsible agency as well as respond to the need of users at the intersection.

**Questions related to signal timing include determining:**

- Prioritizing all type of vehicle or not
- Reviewing and updating the system

- Whether certain movements that go beyond what is considered coordinated would receive preferential attention.
- How to treat the capacity of the intercession
- What metrics will be utilized to evaluate the success of the timing plan? (car pauses, network delays, speed of arterial traffic, guessed person delays, guessed fuel consumption, guessed transit speeds, etc.) and how they will be gathered.

### 6.4.3. JAZ Timing System

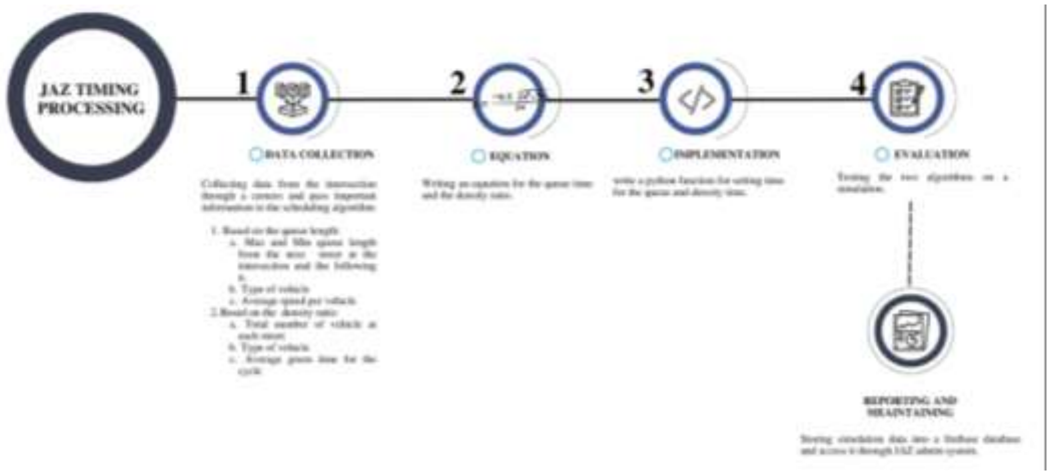

JAZ TIMING PROCESSING
1
DATA COLLECTION
Collecting data from the intersection through a camera and pass important information to the scheduling algorithm.
1. Based on the queue length
a. Max and Min queue length from the next street at the intersection and the following
b. Type of vehicle
c. Average speed per vehicle
2. Based on the density ratio
a. Total number of vehicle at each street
b. Type of vehicle
c. Average green time for the cycle
2
EQUATION
Writing an equation for the queue time and the density ratio.
3
IMPLEMENTATION
write a python function for setting time for the queue and density time.
4
EVALUATION
Testing the two algorithms on a simulation.
REPORTING AND MAINTAINING
Storing simulation data into a firebase database and access it through JAZ admin system.

*Figure 73:* *JAZ Timing System*

### 6.4.4. Based on the queue length

By predicting the number of cars per lane that will be handled in a particular cycle and then selecting a proper green time to handle that amount of traffic, green times for individual movements can frequently be roughly determined. This method is effective for predicting green times for small movements when clearing the line is the main goal and where holding green time for cars to arrive and depart on green as one might do for coordinated through movements is not the intended outcome.

This equation will get the maximum queue at a lane in the road (i), and apply the following equation:

$$GreenTime = StartupLostTime + (QueueLength * AverageTimePerVehicleClass)$$

And with the same process, calculate the green time for the road (i+1).
In our equation, we set 10 as the maximum queue length to detect and count.
Then we classified the length of the queue in the road, whether it was medium or long queue. The same thing is done to the road (i+1).

Afterwards, we are checking the following:

```
  IF (greenTime2>=greenTime):

    IF ((both roads are long) or (both roads medium)):

      greenTime= (greenTime/2)

    Else IF (road i is medium and road i+1 is long):
      greenTime=greenTime-timediff

  ELSE IF (greenTime>=greenTime2):

    IF ((both roads are long) or (both roads medium)):

        greenTime=(greenTime/2)

   ELSE IF (road i is long and road i+1 is medium):

      greenTime=greenTime-diff

  else:
    SAME

Where,

greenTime is the time for road i,
greenTime2 is the time for road i+1,
timediff is (greenTime2-greenTime),
diff is (queue for road i – queue for road i+1)
```

***Figure 74:*** *Green Time Code*

### 6.4.5. Based on the density ratio

This equation is a modified version from the density equation researcher has written [20].

Factors considered while developing the algorithm:

1. Number of lanes
2. Total count of vehicles of each class like cars, trucks, motorcycles, etc on a street.
3. Traffic density determined by the criteria mentioned above
4. Time added as a result of the latency each car has when starting up, as well as the non-linear rise in lag experienced by the vehicles in the rear.
5. The average speed of each class of vehicle when the green light begins, or the average amount of time it takes for each class of vehicle to cross the signal.
6. The minimum and maximum allotted time for the length of the green light, in order to avoid starving

**Working of the algorithm**

Count the number of vehicles in each class that are in the lane, determine the green signal time, and then set the green signal time for this signal and the red signal time for the following signal accordingly. An estimate of the average time each class of vehicle takes to cross an intersection was found using the green signal time based on the number of vehicles of each class at a signal, the average speeds of vehicles at startup, and their acceleration times. The green signal time is then calculated using the formula below.

$$GST = \frac{\sum_{vehicleClass} (NoOfVehicles_{vehicleClass} * AverageTime_{vehicleClass})}{(NoOfLanes\ +\ 1)}$$

***Figure 75:*** *Green Time Equation*

where:

- GST is green signal time
- noOfVehiclesOfClass is the number of vehicles of each class of vehicle at the signal as detected by the vehicle detection module,
- averageTimeOfClass is the average time the vehicles of that class take to cross an intersection, and noOfLanes is the number of lanes at the intersection.

The signals do not transition in the direction with the most density first; instead, they do so in cycles. This is in line with the current system, which sees the signals turn green sequentially in a predictable fashion without requiring people to change their behavior or create any confusion. The yellow signals have also been taken into account, and the signal sequence is the same as it is in the existing system.

**Modification on the density equation:**

Jaz takes care to give the green time for each signal by taking into account the rest of the intersection situation in terms of congestion and density in all the intersections and not just one road, so that if the first street is crowded and gives it a value for the green signal, Jazz is interested in looking at the rest of the roads and seeing if there are roads Crowded other than the road that will be opened this time.

## How JAZ timing works:

To calculate the green light time for *road(i):*
First, we calculate the total number of vehicles at each intersection, and then with the same previous equation, we calculate the green time needed by each road at the intersection (the four roads), the estimated time for this cycle at the intersection is:

$$CycleTime = GSTi + GSTi{+}1 + GST\ i{+}2 + GSTi{+}3$$

But it can change at any moment after opening the traffic light for the road i, for this we use JAZ, we take care of taking the updated data at the intersection each time we calculate the green time.

Then, calculate the average green light time at this iteration:

*avGreen=math.ceil((CycleTime /4) + TotalYellowTime)*

Following Webster's method of taking the ratio of a road (i) and divide it by the total ratio and multiplying it by the cycle time. We will calculate the density of this road i in relation to the total density of the intersection in the 4 roads:

*GreenTime=math.ceil((NoOfVehicle (i)/TotalNoOfVehicle) *(avGreen))*

Minimum and maximum green light were assigned to be 10 sec, 60 sec, minimum green is used to allow drivers to react to the start of the green interval and meet

## 6.5. Admin System Development

### 6.5.2. Admin System

The addition of this system will benefit the research community. We are willing to share open traffic data from our project so that other projects can benefit from it. Making traffic data available to other researchers and developers will enable them to create more accurate models and simulations. This will result in more efficient and reliable systems. Additionally, the data can be used to identify patterns in traffic behavior that can be used to inform public policy decisions.

The Admin Subsystem is a website for managing, sharing, and preserving traffic data from our system. We built it to extract useful information about street-level statistics and vehicles in Saudi Arabia. Detection and Scheduling Models in the JAZ system generate traffic data such as the total number of vehicle passed from the whole intersection during a time t, and the vehicle passed from each lane at the intersection. There is an option within the system to export these data in the form of a report. Other researchers and developers will benefit from this as it will save time and effort for those who will need to download public data. This is true for research as well as development.

### 6.5.3. System Architecture

The central JAZ system and admin will sync to exchange traffic data. The JAZ system will store traffic data in one of the databases, such as the number of vehicles on each lane, the time it takes for red and green lights to turn on in each cycle, and the flow rate at which the vehicles go. The admin system will take the traffic data from the database after it has been stored, and prepare it in a way that can be used by the system.

### 6.5.4. Developing tools

#### Flutter

A software development kit (SDK) for creating mobile apps and websites that are fast and precise is called Flutter. The Flutter framework makes it simple to create user interfaces that respond fluidly to touch thanks to its robust graphics and animation libraries.

Because Flutter is based on the Dart programming language, you can quickly iterate on your code thanks to its efficient development cycle and support for hot reloading.

And one of the features is Expressive and flexible UI: Flutter's UI elements are constructed following the same concepts as Google's Material Design standards, providing you with an expressive and flexible method to design attractive apps.
Additionally, Flutter applications may be created to function on a variety of screen sizes and aspect ratios, which is one of the benefits. Because of this, making apps that look beautiful on both phones and tablets is simple.

#### Firebase

A hosted backend service named Firebase provides real-time databases, cloud storage, authentication, crash reporting, machine learning, remote setup, hosting for static files, and other backend services.
Firebase works with Flutter.

And the tools of firebase we employ in the admin system are:

* * **Authentication -** Firebase Authentication simplifies the development of secure authentication systems for developers and improves user sign-in and onboarding. This feature offers a complete identity solution with support for email and password accounts, phone auth, Google, Facebook, GitHub, Twitter login, and more.

* * **Realtime database -** The Firebase Realtime Database is a NoSQL database hosted in the cloud that provides real-time data synchronization and storage between users. When an app is offline, the data is still accessible since it is continuously synchronized across all clients.

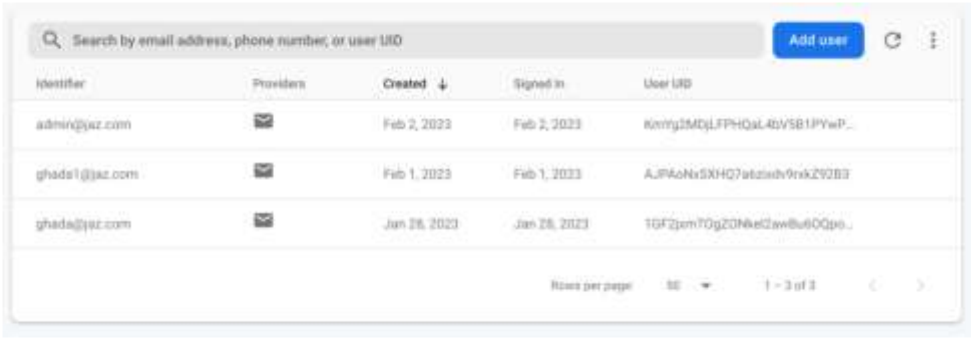
#### Visual Studio Code

The web-based and desktop source code editor Visual Studio Code is free, lightweight, and compatible with Windows and Mac.
It features built-in support for JavaScript, TypeScript, and Node.js and a robust community of extensions for additional programming languages (including C++,

C#, Java, Python, PHP, and Go), runtimes (like.NET and Unity), environments (like Docker and Kubernetes), clouds, and other environments (such as Amazon Web Services, Microsoft Azure, and Google Cloud Platform).

In addition to the general idea of being lightweight and starting quickly, Visual Studio Code has built-in source code control, including Git support, linting, multi-cursor editing, parameter hints, and other powerful editing features. It also has IntelliSense code completion for variables, methods, and imported modules. This was mostly modified using Visual Studio technology.

Integrating the tools above, we developed the admin system. For the **authentication** part, the admin is able to login to the system with his email and password. Using the Authentication function from the firebase, not registered email and incorrect password will not be able to login to the Admin Home Page.

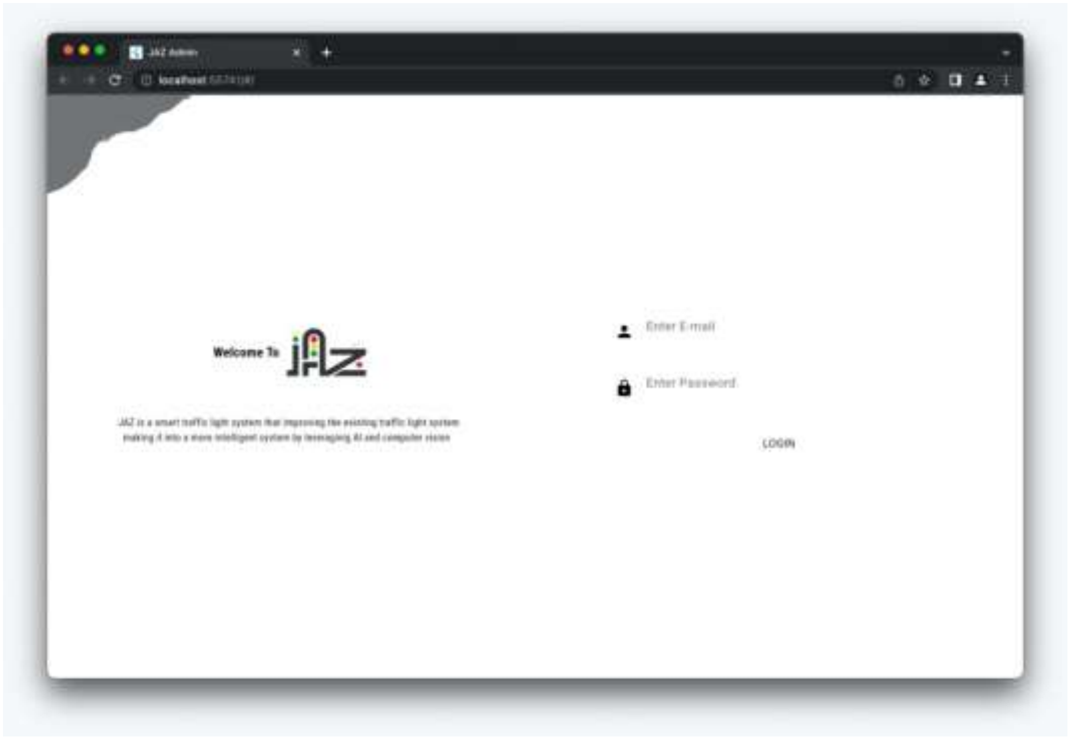


| Identifier | Providers | Created ↓ | Signed In | User UID |
|---|---|---|---|---|
| admin@jaz.com | ✉ | Feb 2, 2023 | Feb 2, 2023 | KmYg2MDjLFPHQaL4bVSB1PYwP... |
| ghada1@jaz.com | ✉ | Feb 1, 2023 | Feb 1, 2023 | AJPAoNxSXHQ7a6zixh9nkZ92B3 |
| ghada@jaz.com | ✉ | Jan 28, 2023 | Jan 28, 2023 | 1GF2pm7OgZONkel2awBu6OQpo... |

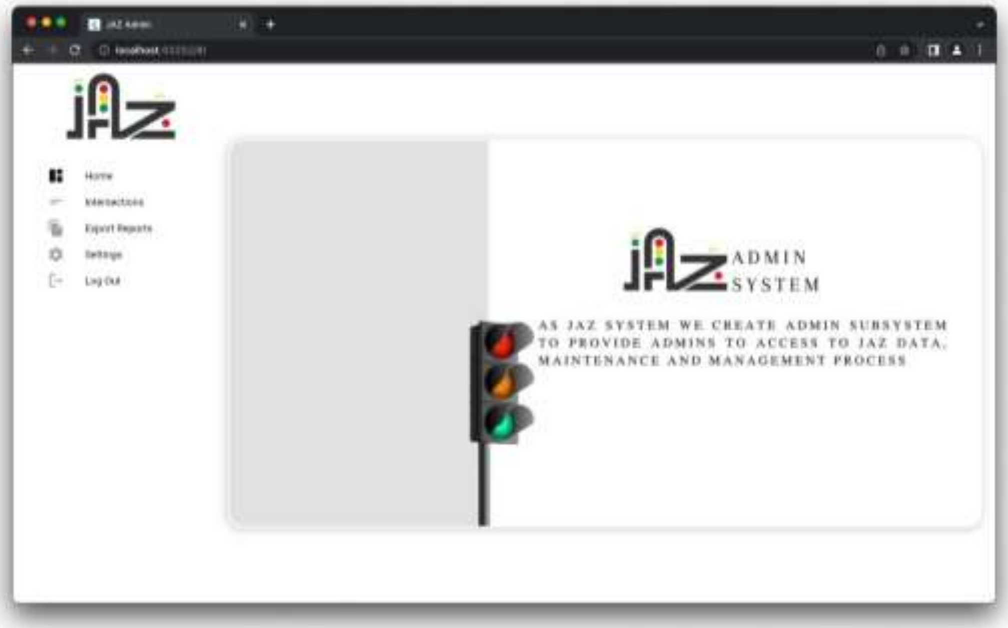
Rows per page: 50 1 – 3 of 3

*Figure 76: Admin **account authentication***

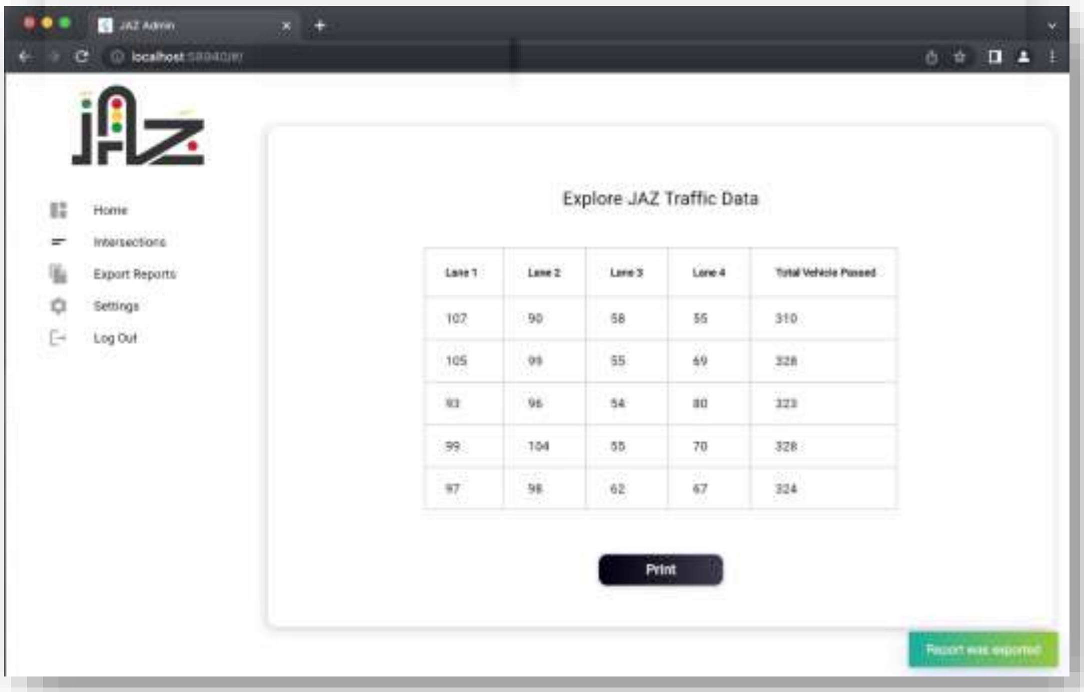
## 6.5.5. Interfaces



JAZ is a smart traffic light system that improving the existing traffic light system making it into a more intelligent system by leveraging AI and computer vision



AS JAZ SYSTEM WE CREATE ADMIN SUBSYSTEM TO PROVIDE ADMINS TO ACCESS TO JAZ DATA, MAINTENANCE AND MANAGEMENT PROCESS



Explore JAZ Traffic Data

| Lane 1 | Lane 2 | Lane 3 | Lane 4 | Total Vehicle Passed |
|---|---|---|---|---|
| 107 | 90 | 58 | 55 | 310 |
| 105 | 99 | 55 | 69 | 328 |
| 93 | 96 | 54 | 80 | 323 |
| 99 | 104 | 55 | 70 | 328 |
| 97 | 98 | 62 | 67 | 324 |

***Figure 77:*** *Admin interfaces*

## 6.6. Implementation Challenges

While using google Colab for the detection model, we found some drawback as:

- **No live editing:** We can cooperate by writing a piece of code and sharing it with the team. However, Google Colab does not offer the option for live editing, which limits the number of concurrent code writers or editors to two. As a result, there is a lot of back and forth sharing.

- **Saving & Storage Issues**: Since Google Colab does not offer permanent storage, uploaded files are deleted when the session is reopened.

- **Limited Space**: The Google Colab platform only has 15GB of free storage for files, which makes it challenging to use when working on larger datasets. The majority of the sophisticated functions can then be stored and carried out by this.

- **Limited Time**: Google Colab users are only permitted to use their notebooks for a maximum of 12 hours per day; however, those who need to work for longer periods of time must access Colab Pro, a paid version that enables programmers to remain connected for 24 hours.

Another challenges we faced while implementing JAZ admin system is, due to the variety in the operating system among the team, for connecting firebase database with flutter on macOS, we need to change the configuration to make the connection.

# CHAPTER 6
# EVALUATION AND TESTING

# 6. EVALUATION AND TESTING

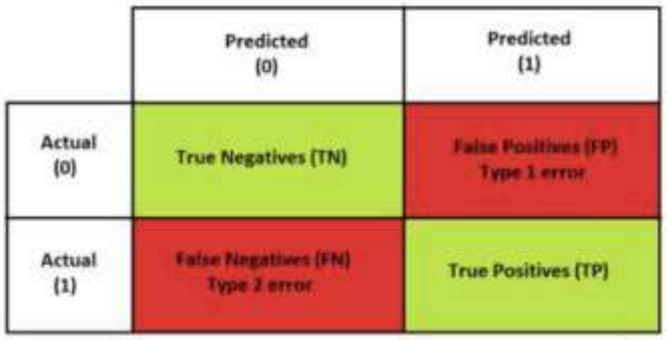
## 6.1. Evaluation of JAZ Detection Module

### 6.1.1. YOLO Evaluation Metrics

Performance metrics measure the accuracy or appropriateness of a model to the extent that all stakeholders involved in utilizing the model are satisfied. There are many situations in which performance metrics can be crucial since they can prevent many issues from arising. For example, a model may have high accuracy, but its true positive rate may be low, and the false negative rate may be high for a cancer classification model. This is problematic since patients that have cancer may not be appropriately detected even if their ratio is low in comparison to how many were identified correctly. The model has to be adjusted in this case so that recall goes up but precision goes down. Performance metrics' significance is undeniable. In the end, they could even save lives! Furthermore, they provide a level of assurance that may not have existed otherwise, making it possible to make appropriate business decisions and predict where an object will be in the future. It is crucial when an error cannot be tolerated. This is why performance metrics deserve to be part of ML classification models Different machine learning algorithms are assessed using various performance indicators. It's crucial to carefully consider the metrics you use to assess your machine learning model. How machine learning algorithm performance is assessed and evaluated depends on the metrics used.

#### Confusion Matrix

The Confusion matrix is one of the most intuitive and easiest metrics used for finding the correctness and accuracy of the model. It is utilized for classification problems where there are two or more possible class outputs. The confusion matrix is a two-dimensional table. ("Actual" and "Predicted") and sets of "classes" in both dimensions. The Actual classifications are columns and Predicted ones are Rows. It gives a holistic view of how well the classification model is performing and what kinds of errors it is making. The confusion matrix is not a performance measure as such, but almost all of the performance metrics are based on confusion Matrix and the numbers inside it.



| | Predicted (0) | Predicted (1) |
|---|---|---|
| Actual (0) | True Negatives (TN) | False Positives (FP) Type 1 error |
| Actual (1) | False Negatives (FN) Type 2 error | True Positives (TP) |

*Figure 78:* *Binary confusion matrix*

A balanced confusion matrix means that all classes have the same number of samples. The “normalized” term means that each class is represented as having 1.00 sample.

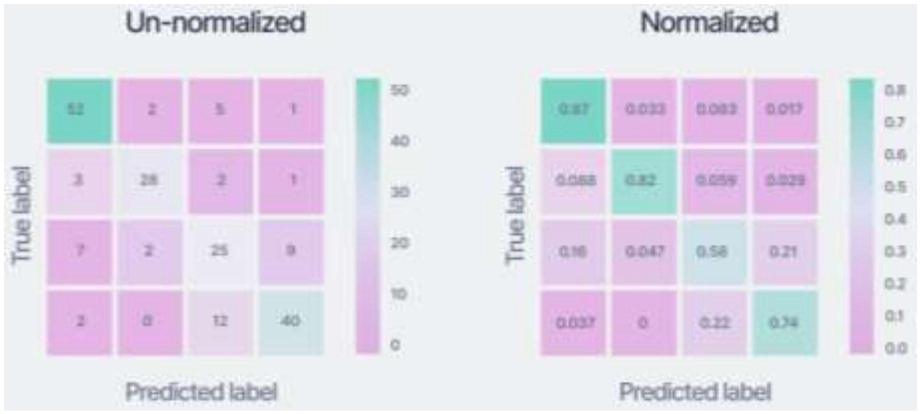


***Figure 79:*** *Types of confusion matrix*

Terms associated with Confusion matrix:

**True Positive (TP)**

- The predicted value matches the actual value
- The actual value was positive and the model predicted a positive value

**Negative (TN)**

- The predicted value matches the actual value
- The actual value was negative and the model predicted a negative value

**False Positive (FP) – Type 1 error**

- The predicted value was falsely predicted
- The actual value was negative but the model predicted a positive value

**False Negative (FN) – Type 2 error**

- The predicted value was falsely predicted
- The actual value was positive but the model predicted a negative value
- Also known as the Type 2 error

## Accuracy

The most widely used and straightforward performance metric, it is just the proportion of properly predicted observations to all observations. Usually, if there is a high accuracy, then the model is at its optimum level of operation. However, accuracy is only great when there is symmetric datasets with almost equal false positives and false negatives. Accuracy = True Positive + True Negative / (True Positive +False Positive + False Negative + True Negative)

## Precision

The term "positive predictive value" can also be used to describe precision. It contrasts the overall number of positives predicted by your model with the number of true positives predicted by your model. A measurement of quality, exactness, or accuracy is also known as precision

*Precision = True Positives / (True Positives + False Positives)*
*True Positives + False Positives = Total Predicted Positives*

## Recall or Sensitivity

Sensitivity or true positive rate are other names for recall. It contrasts the actual number of positives in the data with the right positives that the model anticipated. The term "measure of completeness" can also be used to refer to recall.

*Recall = True Positives / (True Positives + False Positives)*
*True Positives + False Positive = True Actual Positives*

## Mean Average Precision (mAP)

AP (Average Precision) is a popular metric in measuring the accuracy of object detectors such as Fast R-CNN, YOLO, Mask R-CNN, etc. For recall values greater than or equal to 0, average precision calculates the average precision value. The weighted mean of precisions at each threshold is used to compute average precision. The recall gain above the earlier criterion is what determines the weight. The mean of each class's AP is known as the mean average precision.

mAP formula is based on the following sub metrics:

**• Confusion Matrix • Recall • Precision**

Finding the Average Precision (AP) for each class and then averaging it over several courses is how the mAP is determined. The mAP takes into account both false positives (FP) and false negatives (FN), and accounts for the trade-off between accuracy and recall (FN). Due to this characteristic, the majority of detecting applications may use mAP as a measure.

## F1 Score

Since they both characterize a model's efficacy in complementary ways, precision and recall are often employed in conjunction. These two make up the weighted harmonic mean of recall and precision, or F1, respectively.

*F1 Score = 2 * (Precision * Recall) / (Precision + Recall)*

### 6.1.2. YOLOv5 Result

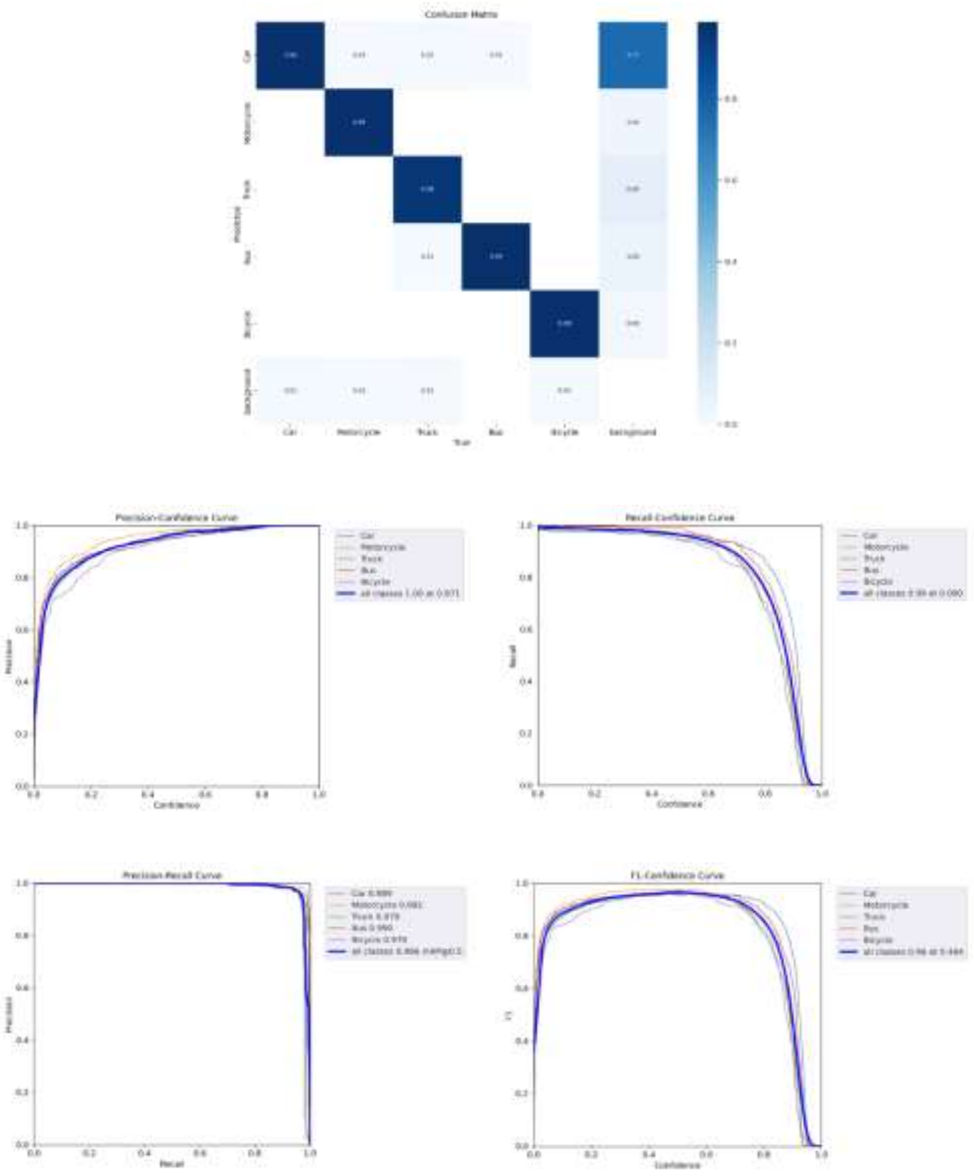


***Figure 80:*** *YOLOv5 Graphs*

In ***Figure 80:*** The label (a) shows the normalized(N) confusion(C) matrices(M) graph which represent all the True Positives values along the diagonal and the other values will be False Positives or False Negatives, (b) shows the precision(P) versus confidence(C) graph, (c) the recall(R) versus confidence(C), (d) is the mean average precision(mAP) based on comparing the truth bounding box and detection box, and (e) the F1 score at 96% with confidence of 0.484, which advocates the balance between P and R based on vehicles dataset. The mAP for all classes is high and accurately modeling detections at 98.6% with a threshold of 0.5. The P and R are high at 100% and 99%, respectively, and more confidence at 0.971 and 0.0 respectively for all classes.

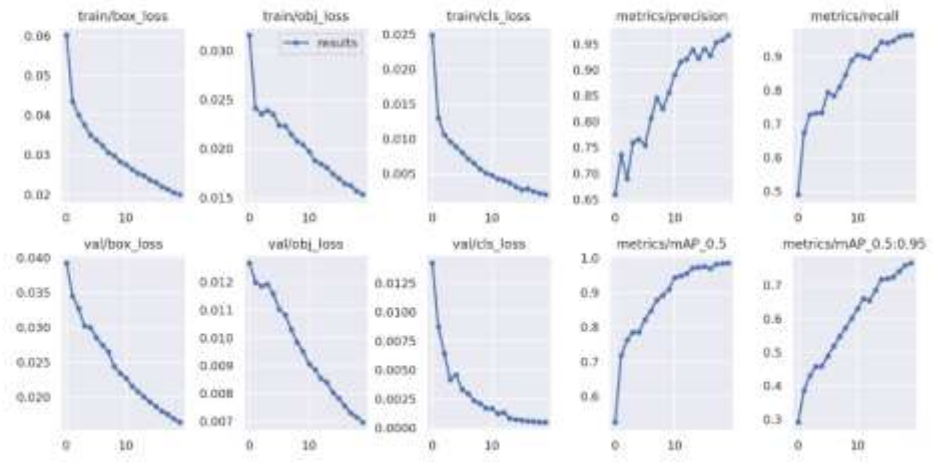


***Figure 81:*** *Training and validations losses of the YOLOv5*

***Figure 81:*** Shows both training and validations losses of the YOLOv5 algorithm object detection and classification on 20 epochs for vehicles dataset.

***Figure 82:*** *validation sample set of YOLOv5*

*Figure 82:* Shows the validation sample set of YOLOv5. The first column (**a**) are the label samples, and the second column (**b**) are the predicted samples. As a result of the model's high detection rate, it is considered precise and efficient.

### 6.1.3. YOLOv7 Result

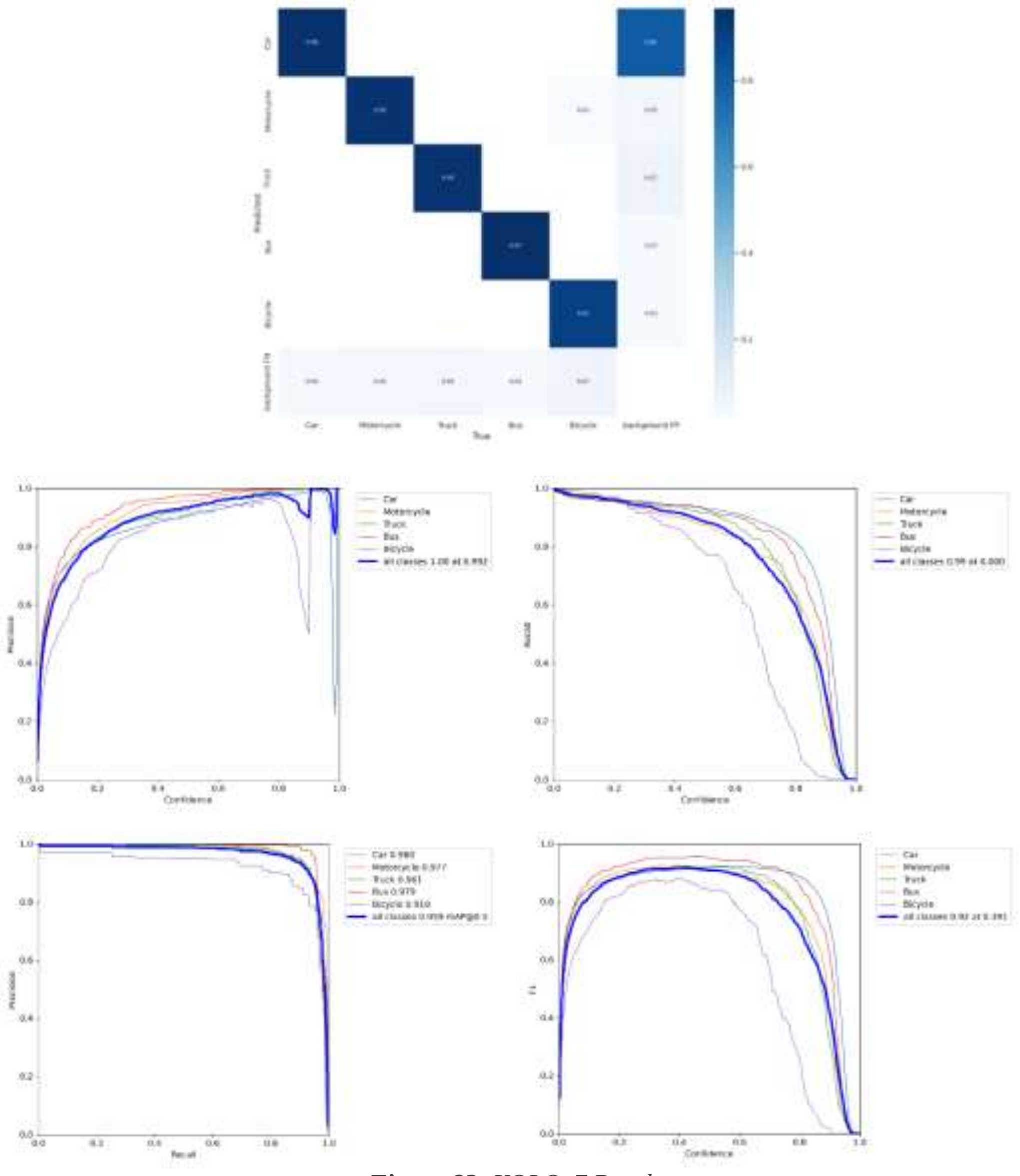


***Figure 83:*** *YOLOv7 Results*

*Figure 83:* The label (a) shows the normalized(N) confusion(C) matrices(M) graph which represent all the True Positives values along the diagonal and the other values will be False Positives or False (**b**) shows the precision(P) versus confidence(C) graph, (**c**) the recall(R) versus confidence(C), (**d**) is the mean average precision(mAP) based on comparing the truth bounding box and detection box, and (**e**) the F1 score at 92% with confidence of 0.391, which

advocates the balance between P and R based on vehicles dataset. The mAP for all classes is high and accurately modeling detections at 95.9% with a threshold of 0.5. The P and R are high at 100% and 99%, respectively, and more confidence at 0.992 and 0.0 respectively for all classes.

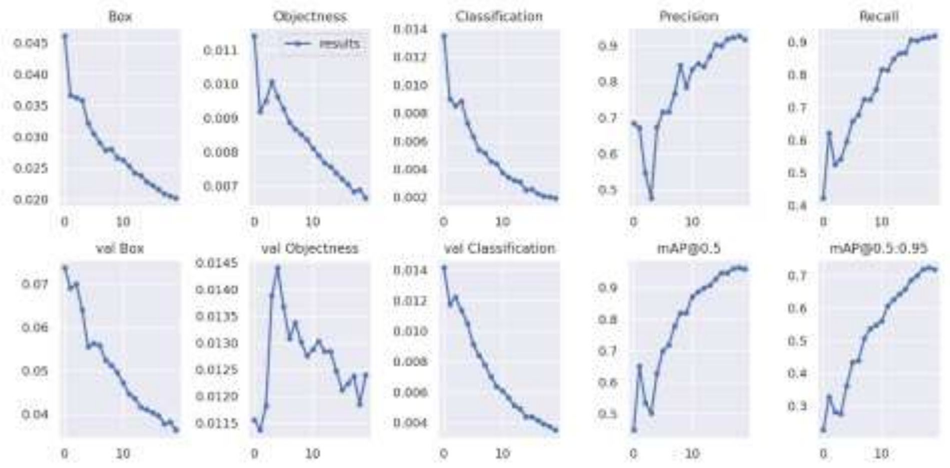


***Figure 84:*** *Training and validations losses of the YOLOv7*

***Figure 84:*** Shows both training and validations losses of the YOLOv7 algorithm object detection and classification on 20 epochs for vehicles dataset.

**Comparison between YOLOv5 and YOLOv7 results:**

| Model | Precision | Recall | mAP | F1 |
|---|---|---|---|---|
| YOLOv5 | 100% | 99% | 98.6% | **96%** |
| YOLOv7 | 100% | 99% | 95.9% | **92%** |

***Table 15:*** *Comparison between YOLOv5 and YOLOv7 results*

YOLOv5 has proven to be effective with high performance in precision and recall, accompanied by the high confidence values on the dataset. It achieved a greater balance between precision and recall, with a mean average precision of 98.6% at a 0.5 threshold for all classes. This demonstrated that the algorithm could be trusted for accurately detecting and correctly classifying the objects of interest.

The main reason why YOLOv5 achieve better results than YOLOv7 due to its accuracy and speed on normal GPU system comparing with YOLOv7 which works better on latest high- speed GPUs. In addition,YOLOv7 training speed is slow on custom data because it uses more floating point operations (more computational) as compared to YOLOv5. Lastly, YOLOv7 is still in the development phase. While YOLOv5 uses stable and efficient memory while training on custom data, YOLOv7 is still unable to use efficient GPU memory.

## 6.2. Evaluation of JAZ Scheduling Algorithm

### 6.2.1. Pygame Simulation

The simulation was taken and modified from a developed version that was created from scratch using Pygame to simulate actual traffic. It helps with system visualization and comparison with the current static system. There are 4 traffic lights at a 4-way intersection there. Each signal has a timer on top that displays the amount of time until it changes from green to yellow, yellow to red, or red to green. The quantity of vehicles that have passed through the intersection is also shown next to each light. Cars, bikes, buses, trucks, rickshaws, and other types of vehicles arrive from all directions.

Some of the vehicles in the rightmost lane turn to cross the intersection to increase the realism of the simulation. When a vehicle is generated, random numbers are also used to determine whether or not it will turn. It also has a timer that shows how much time has passed since the simulation began.
The scheduling will start in the time of 5 in the yellow light time by first detecting and counting all the vehicles in the lane and continue until it measures the adaptive green light time as it is stated.

In order to improve efficiency and measure the green light duration based on the simulation, the *setTime* method that was used to set the green time for the intersection was modified and enhanced by the scheduling algorithm.

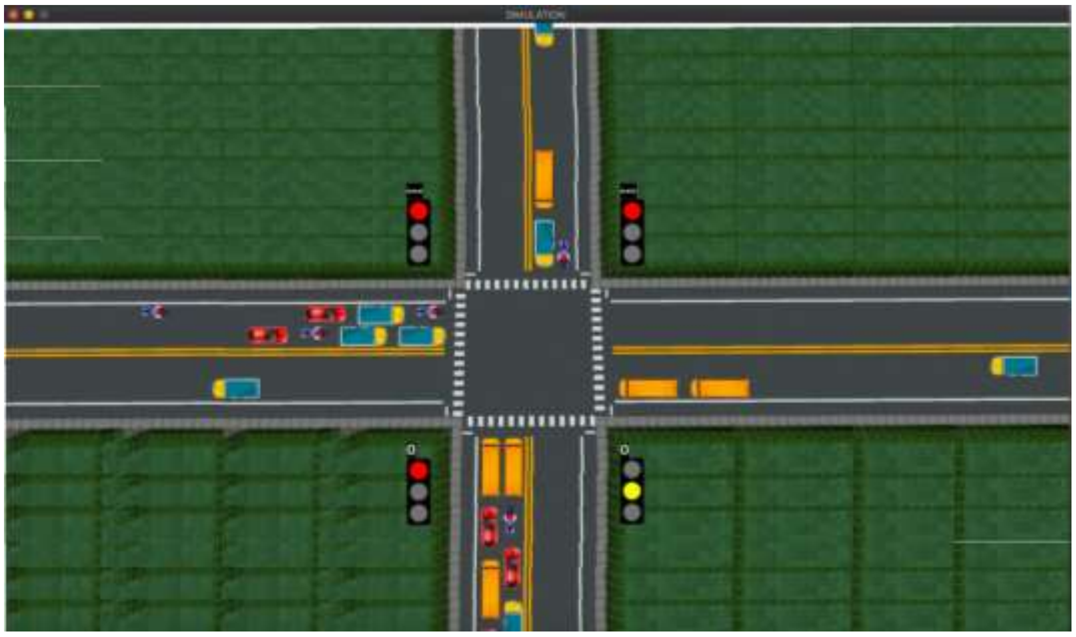
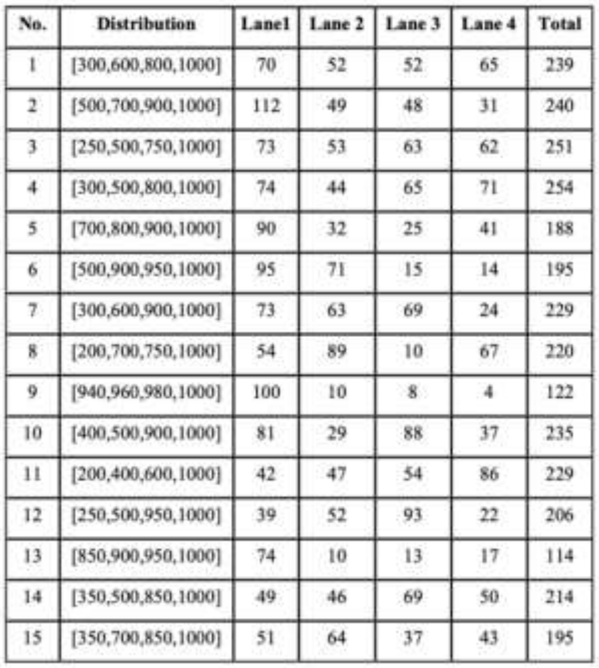


***Figure 85:*** *JAZ simulation demo*

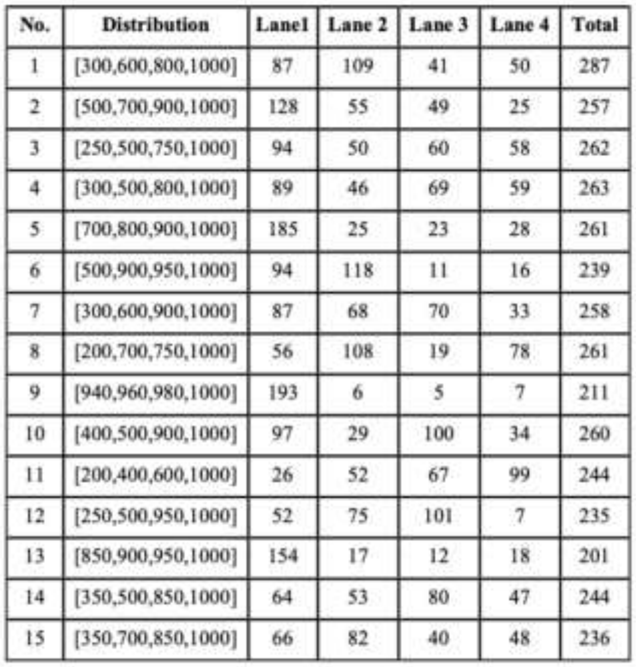
### 6.2.2. Previous Results

The researcher in [20] proposed an adaptive traffic light timing by calculating the density of a single road at the intersection and assign the green light time to it, ignoring the 3 other roads in the intersection. His performance is measured in the term of total passing vehicle per 5 minutes. Researcher ran the simulation 15 times with different distribution of the vehicle to be generated which help in imitation the real-life situation at the intersection.

His result shows an improvement when the distribution of vehicles in the lanes is moderately or sharply skewed and different.



| No. | Distribution | Lane1 | Lane 2 | Lane 3 | Lane 4 | Total |
|---|---|---|---|---|---|---|
| 1 | [300,600,800,1000] | 70 | 52 | 52 | 65 | 239 |
| 2 | [500,700,900,1000] | 112 | 49 | 48 | 31 | 240 |
| 3 | [250,500,750,1000] | 73 | 53 | 63 | 62 | 251 |
| 4 | [300,500,800,1000] | 74 | 44 | 65 | 71 | 254 |
| 5 | [700,800,900,1000] | 90 | 32 | 25 | 41 | 188 |
| 6 | [500,900,950,1000] | 95 | 71 | 15 | 14 | 195 |
| 7 | [300,600,900,1000] | 73 | 63 | 69 | 24 | 229 |
| 8 | [200,700,750,1000] | 54 | 89 | 10 | 67 | 220 |
| 9 | [940,960,980,1000] | 100 | 10 | 8 | 4 | 122 |
| 10 | [400,500,900,1000] | 81 | 29 | 88 | 37 | 235 |
| 11 | [200,400,600,1000] | 42 | 47 | 54 | 86 | 229 |
| 12 | [250,500,950,1000] | 39 | 52 | 93 | 22 | 206 |
| 13 | [850,900,950,1000] | 74 | 10 | 13 | 17 | 114 |
| 14 | [350,500,850,1000] | 49 | 46 | 69 | 50 | 214 |
| 15 | [350,700,850,1000] | 51 | 64 | 37 | 43 | 195 |



| No. | Distribution | Lane1 | Lane 2 | Lane 3 | Lane 4 | Total |
|---|---|---|---|---|---|---|
| 1 | [300,600,800,1000] | 87 | 109 | 41 | 50 | 287 |
| 2 | [500,700,900,1000] | 128 | 55 | 49 | 25 | 257 |
| 3 | [250,500,750,1000] | 94 | 50 | 60 | 58 | 262 |
| 4 | [300,500,800,1000] | 89 | 46 | 69 | 59 | 263 |
| 5 | [700,800,900,1000] | 185 | 25 | 23 | 28 | 261 |
| 6 | [500,900,950,1000] | 94 | 118 | 11 | 16 | 239 |
| 7 | [300,600,900,1000] | 87 | 68 | 70 | 33 | 258 |
| 8 | [200,700,750,1000] | 56 | 108 | 19 | 78 | 261 |
| 9 | [940,960,980,1000] | 193 | 6 | 5 | 7 | 211 |
| 10 | [400,500,900,1000] | 97 | 29 | 100 | 34 | 260 |
| 11 | [200,400,600,1000] | 26 | 52 | 67 | 99 | 244 |
| 12 | [250,500,950,1000] | 52 | 75 | 101 | 7 | 235 |
| 13 | [850,900,950,1000] | 154 | 17 | 12 | 18 | 201 |
| 14 | [350,500,850,1000] | 64 | 53 | 80 | 47 | 244 |
| 15 | [350,700,850,1000] | 66 | 82 | 40 | 48 | 236 |

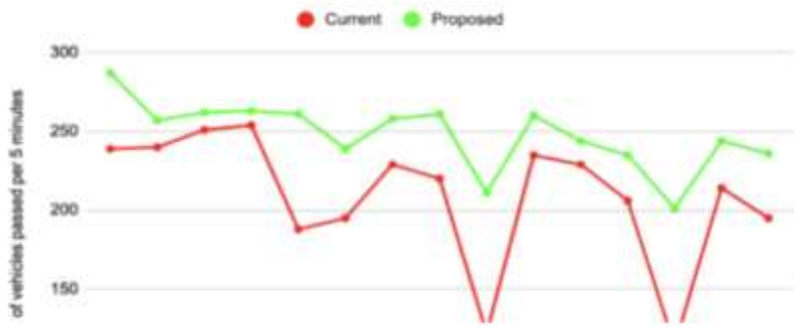
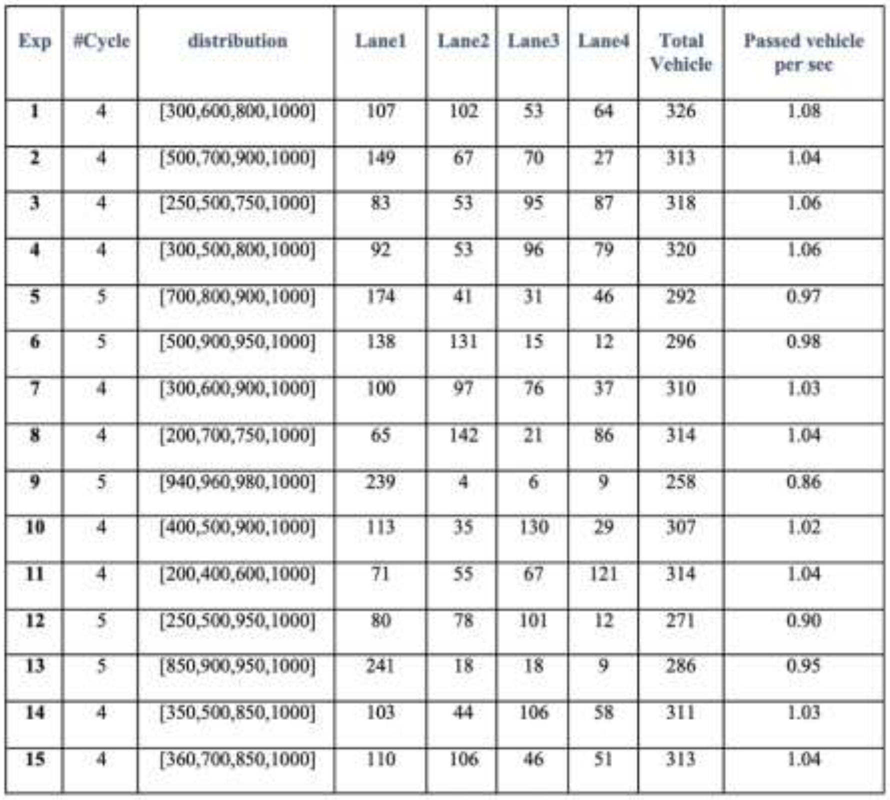


*Figure 86: Previous results of adaptive traffic light timing by calculating the density of a single road at the intersection and assign the green light time*

### 6.2.3. Queue Length Algorithm

Using the simulation above, the setTime function was modified in order to set the time based on the queue length of road i and road i+1. Afterward, comparing the time needed for both roads and take the decision based on the condition specified. This method showed an improvement to the static timing systems where the time is always the same for all the roads under all conditions. The simulation ran 15 experiments, each experiment is 5 minutes long with different vehicle distribution with every experiment.

On comparison the queue length timing function with the static timing system, the queue method increased the number of passing vehicle per 5 minutes. This is mainly because this method will check the time assigned to road i and road i+1 and see if roadi+1 have longer queue, then it is better to lower my given time so that the next road won't have much longer queue length and to slightly decrease their waiting time.  Queue algorithm result in 4-5 full cycles in the 5 minutes, having more cycles help increasing the flow at the intersection and minimize the total delay.



| Exp | #Cycle | distribution | Lane1 | Lane2 | Lane3 | Lane4 | Total Vehicle | Passed vehicle per sec |
|---|---|---|---|---|---|---|---|---|
| **1** | 4 | [300,600,800,1000] | 107 | 102 | 53 | 64 | 326 | 1.08 |
| **2** | 4 | [500,700,900,1000] | 149 | 67 | 70 | 27 | 313 | 1.04 |
| **3** | 4 | [250,500,750,1000] | 83 | 53 | 95 | 87 | 318 | 1.06 |
| **4** | 4 | [300,500,800,1000] | 92 | 53 | 96 | 79 | 320 | 1.06 |
| **5** | 5 | [700,800,900,1000] | 174 | 41 | 31 | 46 | 292 | 0.97 |
| **6** | 5 | [500,900,950,1000] | 138 | 131 | 15 | 12 | 296 | 0.98 |
| **7** | 4 | [300,600,900,1000] | 100 | 97 | 76 | 37 | 310 | 1.03 |
| **8** | 4 | [200,700,750,1000] | 65 | 142 | 21 | 86 | 314 | 1.04 |
| **9** | 5 | [940,960,980,1000] | 239 | 4 | 6 | 9 | 258 | 0.86 |
| **10** | 4 | [400,500,900,1000] | 113 | 35 | 130 | 29 | 307 | 1.02 |
| **11** | 4 | [200,400,600,1000] | 71 | 55 | 67 | 121 | 314 | 1.04 |
| **12** | 5 | [250,500,950,1000] | 80 | 78 | 101 | 12 | 271 | 0.90 |
| **13** | 5 | [850,900,950,1000] | 241 | 18 | 18 | 9 | 286 | 0.95 |
| **14** | 4 | [350,500,850,1000] | 103 | 44 | 106 | 58 | 311 | 1.03 |
| **15** | 4 | [360,700,850,1000] | 110 | 106 | 46 | 51 | 313 | 1.04 |

***Figure 87:*** *Queue Length Algorithm*

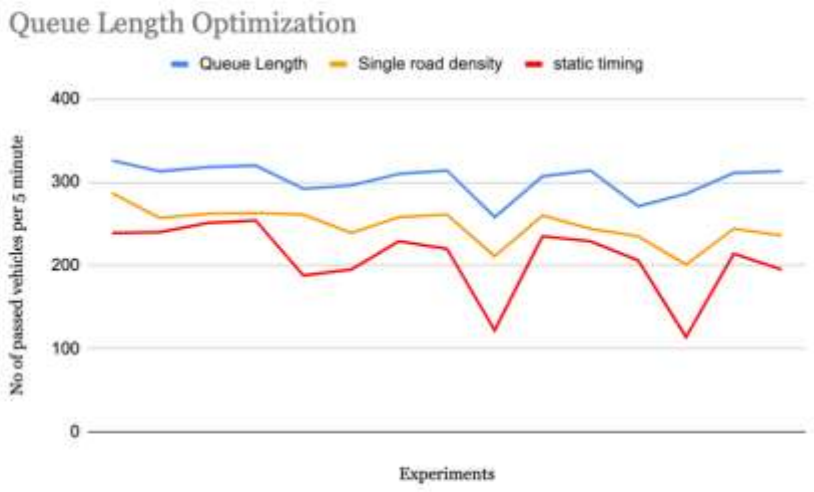
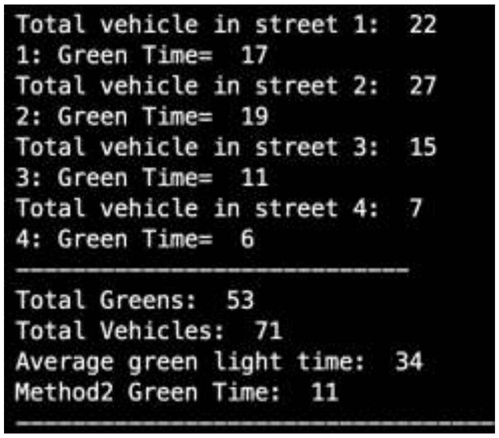


***Figure 88:*** *Queue Length Optimization*

### 6.2.4. Density Ratio Algorithm

In the density ratio algorithm, the *setTime* function was responsible to get the intersection data in order to calculate the optimal green light time for each lane dependent on the density ratio for road I to the density in the whole intersection.

This algorithm showed a great improvement to both, researcher proposed system and the static system under all conditions and vehicles distributions. It shows that with knowing the total number of vehicle in the intersection, and taking into account the average green light time needed, if road I density ratio is relatively small to other road density ratio, this road will not be needed long green light time, and if other road density is larger, meaning that road i is small comparing to them, the method will lower the green light time in order to other congested road to pass more quickly. Where if road I is relatively large, then longer amount of green light would be assigned for that road than roads with small density ratio. This result in giving the right time for each road considering the whole intersection which leads in increasing the total number of passing vehicles.



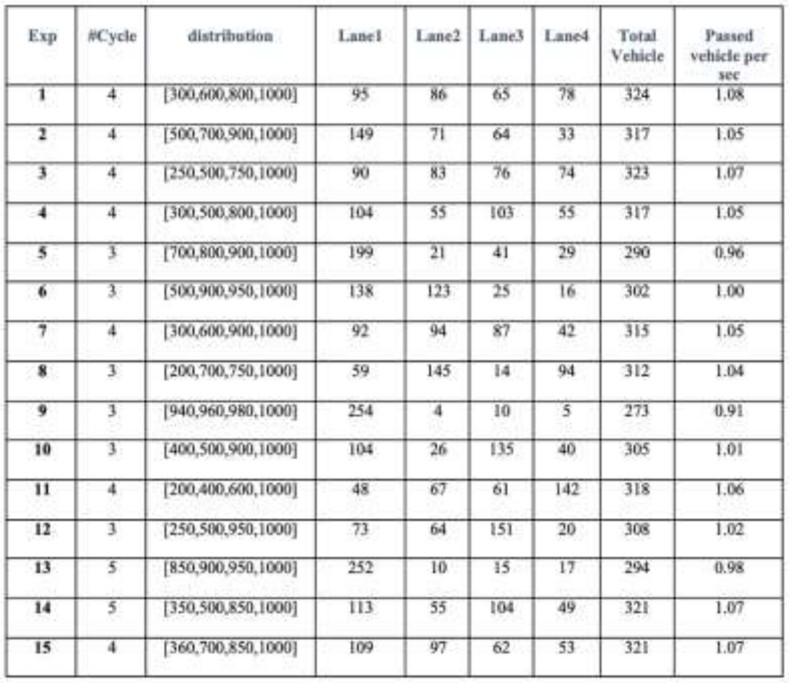
```
Total vehicle in street 1:  22
1: Green Time=  17
Total vehicle in street 2:  27
2: Green Time=  19
Total vehicle in street 3:  15
3: Green Time=  11
Total vehicle in street 4:  7
4: Green Time=  6
------------------------------
Total Greens:  53
Total Vehicles:  71
Average green light time:  34
Method2 Green Time:  11
```

This is a sample from an experiment, the street 1 is the one going to be opened that is when we get the intersection information and the total number of vehicles and the average green time for the whole intersection, then for street 1 calculate the green time based on the equation in [20].

This example showed that streets 2 and 3 are relatively large so based on the density ratio of street 1, the green light time will decrease from 17 sec to 11 sec.

*GreenStreet1= (22/71) *34=11 second*

*Figure 89:*



| Exp | #Cycle | distribution | Lane1 | Lane2 | Lane3 | Lane4 | Total Vehicle | Passed vehicle per sec |
|---|---|---|---|---|---|---|---|---|
| **1** | 4 | [300,600,800,1000] | 95 | 86 | 65 | 78 | 324 | 1.08 |
| **2** | 4 | [500,700,900,1000] | 149 | 71 | 64 | 33 | 317 | 1.05 |
| **3** | 4 | [250,500,750,1000] | 90 | 83 | 76 | 74 | 323 | 1.07 |
| **4** | 4 | [300,500,800,1000] | 104 | 55 | 103 | 55 | 317 | 1.05 |
| **5** | 3 | [700,800,900,1000] | 199 | 21 | 41 | 29 | 290 | 0.96 |
| **6** | 3 | [500,900,950,1000] | 138 | 123 | 25 | 16 | 302 | 1.00 |
| **7** | 4 | [300,600,900,1000] | 92 | 94 | 87 | 42 | 315 | 1.05 |
| **8** | 3 | [200,700,750,1000] | 59 | 145 | 14 | 94 | 312 | 1.04 |
| **9** | 3 | [940,960,980,1000] | 254 | 4 | 10 | 5 | 273 | 0.91 |
| **10** | 3 | [400,500,900,1000] | 104 | 26 | 135 | 40 | 305 | 1.01 |
| **11** | 4 | [200,400,600,1000] | 48 | 67 | 61 | 142 | 318 | 1.06 |
| **12** | 3 | [250,500,950,1000] | 73 | 64 | 151 | 20 | 308 | 1.02 |
| **13** | 5 | [850,900,950,1000] | 252 | 10 | 15 | 17 | 294 | 0.98 |
| **14** | 5 | [350,500,850,1000] | 113 | 55 | 104 | 49 | 321 | 1.07 |
| **15** | 4 | [360,700,850,1000] | 109 | 97 | 62 | 53 | 321 | 1.07 |

***Figure 90:*** *Density Ratio Algorithm*

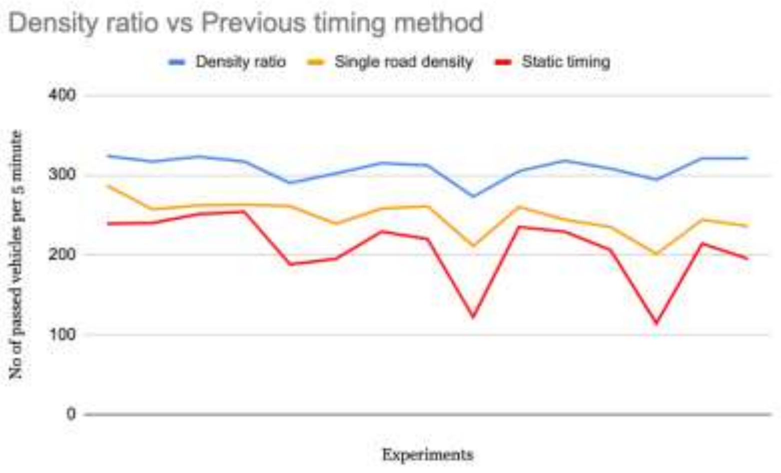


***Figure 91:*** *Density Ratio Algorithm vs Previous Timing Method*

### 6.1.1. Queue Length vs Density Ratio

Both algorithms showed a great improvement and provide more practical solution than the existing developed models. But as shown in the graph bellow, density ratio is slightly better than the queue length as it takes into consideration the whole intersection which help in increase the total of passing vehicles.

Another reason for the density ratio to be better is because it is more dynamic and can react to short term demand changes and decrease an excessive delay.

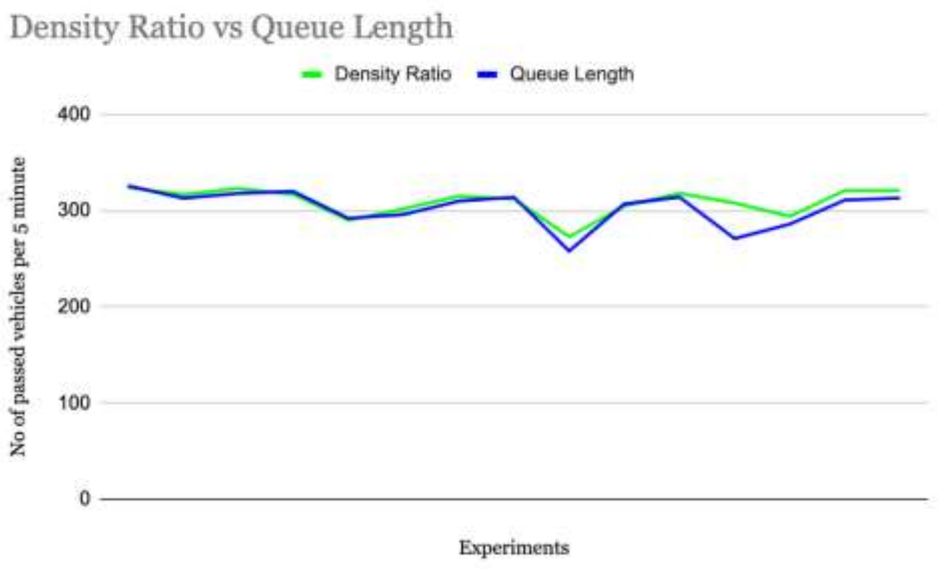


***Figure 92:*** *Density Ratio Algorithm vs Queue Length Algorithm*

### 6.1.2. Comparisons against prior works

The distribution of traffic is different in each experiment, and as it shows in the tables above. the distribution can be divided into an equal or almost equal distribution among the 4 lanes as in exp 1,2,3,4,10,11,12, moderately skewed when some of the lanes have high traffic as in exp 5,6,7,8,14,15, lastly, sharply skewed when a lane is significantly high in traffic and the rest of the 3 lanes are not as in exp 9,13.
The improvement depends on the skewness in the distribution. The more skewed a distribution, the more percentage increasing the improvement show.

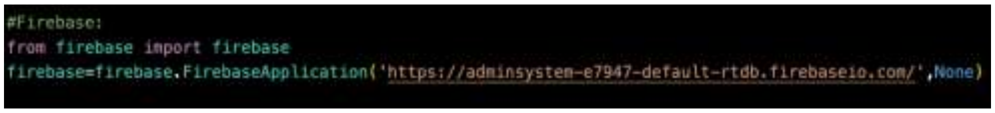
| **Density Ratio and Static Timing** | **Density Ratio and Single Density** | **Queue Length and Static Timing** | **Queue Length and Single Density** |
|---|---|---|---|
| Equal Distribution | | | |
| **33%** | **22%** | 31% | 20% |
| Moderately Skewed | | | |
| **50%** | **24%** | 47% | 22% |
| Sharply Skewed | | | |
| **140%** | **37%** | 130% | 32% |

***Table 16:*** *Comparisons against prior works*

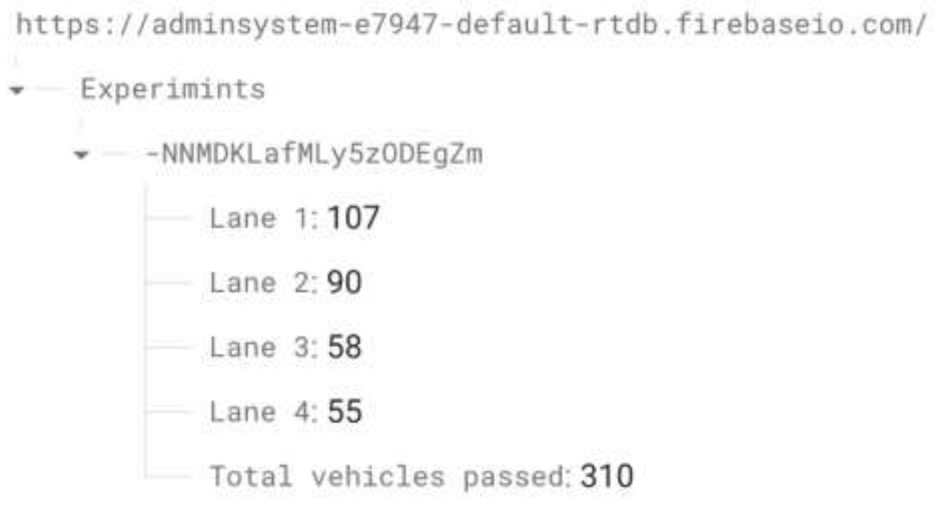
## 6.1. Evaluation of JAZ Admin System

The monitoring of signal timing operations and maintenance is included as the last step of the signal timing environment and can take place in a variety of ways. Storing every experiment from the simulation python file, help us to know the number of passing vehicle from each lane, which is great way to know the distribution of the vehicle among the lanes. Admin who can view this data is responsible to monitor the system and evaluate the timing algorithm used by the total vehicle passed factor.

The important part of the development was connecting the simulation code which is the python program we modify in order to test our equation; this file is discussed in evaluation chapter. **Storing the intersection data** during the experiment in firebase Realtime database, then from flutter we connect the web app with the same database and were able to display the stored information in a table on the web interface.



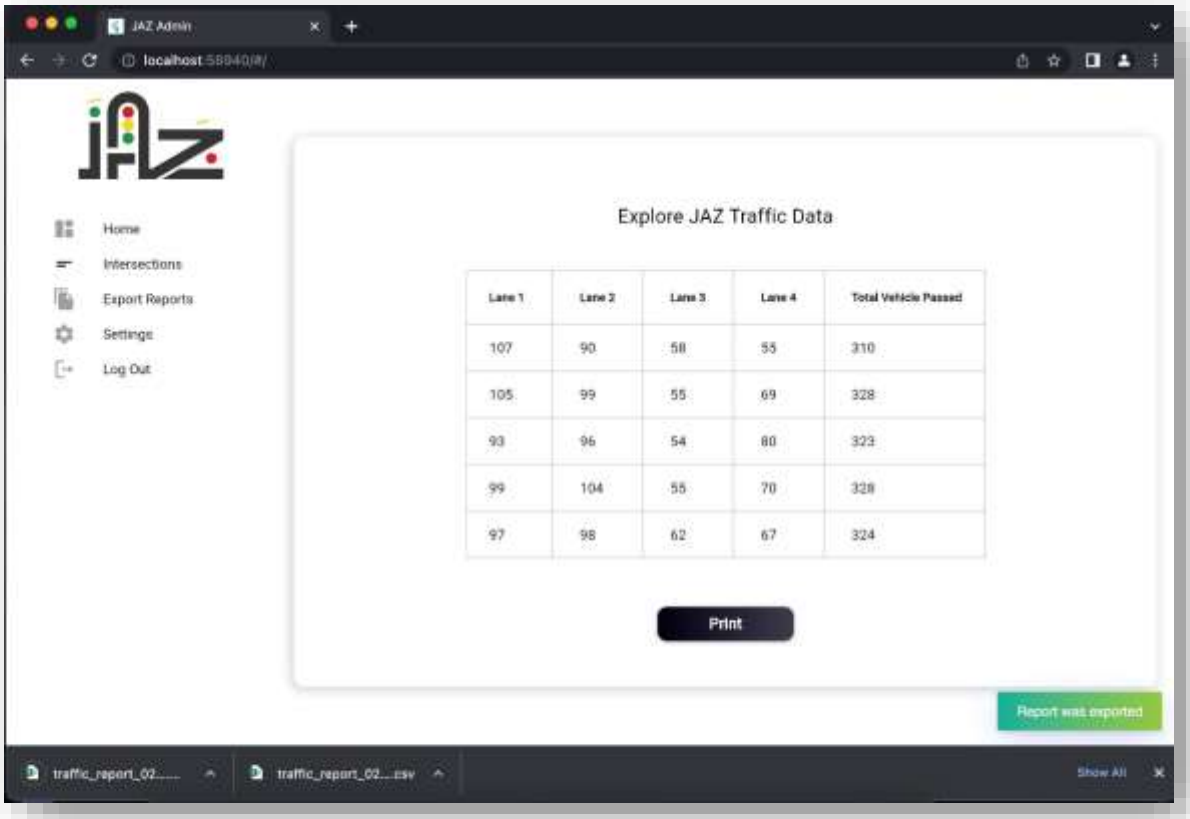
```
#Firebase:
from firebase import firebase
firebase=firebase.FirebaseApplication('https://adminsystem-e7947-default-rtdb.firebaseio.com/',None)
```



```
https://adminsystem-e7947-default-rtdb.firebaseio.com/
▾ — Experimints
    ▾ — -NNMDKLafMLy5zODEgZm
          — Lane 1: 107
          — Lane 2: 90
          — Lane 3: 58
          — Lane 4: 55
          — Total vehicles passed: 310
```

***Figure 93:*** *Fiber base for the admin system*

The data from the simulation code is stored in the real time database successfully. Afterward, we displayed these data into the admin into JAZ traffic data interface in a table format.

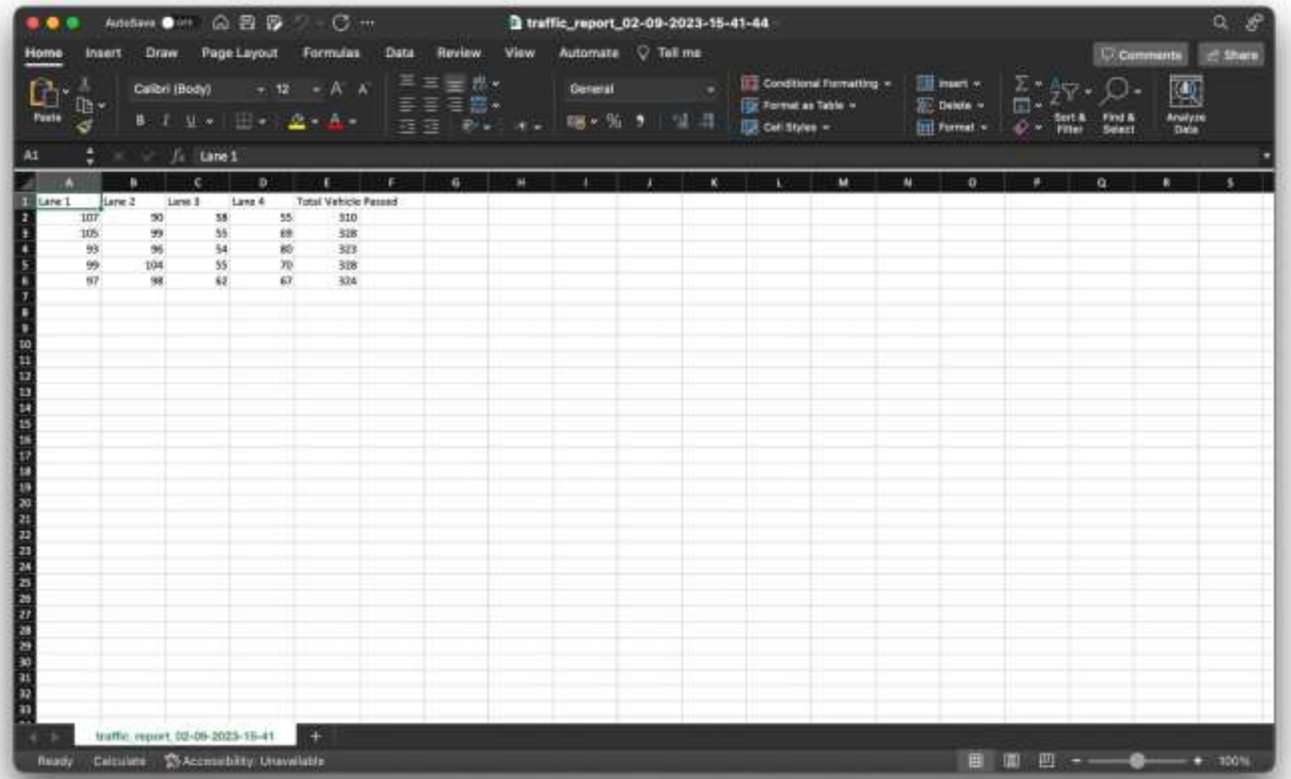


Explore JAZ Traffic Data

| Lane 1 | Lane 2 | Lane 3 | Lane 4 | Total Vehicle Passed |
|---|---|---|---|---|
| 107 | 90 | 58 | 55 | 310 |
| 105 | 99 | 55 | 69 | 328 |
| 93 | 96 | 54 | 80 | 323 |
| 99 | 104 | 55 | 70 | 328 |
| 97 | 98 | 62 | 67 | 324 |

***Figure 94:*** *Exploring JAZ traffic data interface*

Afterward, we Export traffic JAZ traffic data from JAZ traffic data interface into a CSV (Comma-separated values) File as report .


traffic_report_02-09-2023-15-41-44

| Lane 1 | Lane 2 | Lane 3 | Lane 4 | Total Vehicle Passed |
|---|---|---|---|---|
| 107 | 90 | 58 | 55 | 310 |
| 105 | 99 | 55 | 69 | 328 |
| 93 | 96 | 54 | 80 | 323 |
| 99 | 104 | 55 | 70 | 328 |
| 97 | 98 | 62 | 67 | 324 |

*Figure 95:* Traffic Report as CSV File

# CHAPTER 7

# CONCLUSION AND FUTURE WORKS

# 7. CONCLUSION AND FUTURE WORKS

## 7.1. Conclusion

Our project, Jaz smart traffic light management, aims to improve the effectiveness of traffic signal control systems via the use of newer methods, technologies, and tools. By using density ratio equation to set the green light time, we get the advantage from the information and data from the whole intersection after obtaining it from the detection model. Using YOLO detection model, we achieved a 0.95 mAP, and we increased the total of passing vehicle per 5 minutes comparing it with previous related work with maximum percentage increase in sharp distribution, 140% to the static system and 37% increase to the adaptive system in [20], by taking into account the whole intersection condition in term of congestion which help reduce the red-light time at congested street at the intersection.

Technology is developing rapidly and will only become better in the near future. This begs the concerns of whether we can keep up with the ever-increasing rate of technology and, even if we can, what will happen to flying vehicles. Will there even be a need for traffic signals soon? The significance of intelligent traffic signals is astounding, and we must accept that they enhance and save lives. Future cities can only be intelligent if intelligent traffic signal systems are implemented.

Due to an increase in the number of drivers on the roads during the past twenty years, a lot of focus has been placed on improving road safety. Governments and manufacturers will embrace any improvements in technology since safety is and will always be the first concern. People are driving for an increasing number of hours, which not only adds to the commotion but also pollutes the environment. As a result, we developed **Jaz**, a better traffic light system for smart cities. as Traffic lights are part of a cities infrastructure and therefore present a perfect opportunity to assist in creating a "Smart City."

We have a good chance that our system, Jaz, will have a big impact on traffic jams, air pollution, travel time, and even traffic accidents. Given how extensively traffic signal systems are already employed, we believe that doing so improves air quality, lowers fuel consumption, and decreases congestion. As a result, given these advantages, one can only make educated guesses regarding the favorable impacts on a city's economy, the improvement in traffic safety, and the advantages for users, who number millions globally. By carrying out this project and using artificial intelligence technologies in its implementation, we seek to promote the 2030 vision and the concept of the smart city.

By the end of this project there is still room for improvement, therefore further design, implementation, and testing suggestions for the enhancement will be defined and discussed in the following on future work.

## 7.2. Future works

The system could be further enhanced by adding the following features:

- **Weather-responsive traffic lights**: one of the enhancements that could make the traffic light smarter in taking the decisions is modifying the timing of traffic signals to take road conditions into account, may increase safety during bad weather.

- **Give priority to public transport and improve scheduling:** since the public transportation is widely used currently in our kingdom's cities so in order to enhance the effectiveness and quality of the traffic flow signals may detect if there is a public transport vehicle in the road and take the timing decisions that help the public transport to move in dynamic flow.

- **Provide special controls for emergency services vehicles**: Allowing emergency vehicles to go through red traffic signals or have a kind of control on the smart traffic light that help them to save the time and move faster or the traffic light may also detect the emergency vehicles to take the timing decisions that help the emergency services to save the time and save lives.

- **smart traffic light systems automatically notify emergency services of the accident**: since the detection playing a huge part in our project so we can enhance the detection algorithm by adding accident detection feature to the smart traffic light and after detecting the accident then notify the emergency services of the accident information to save time and road safety.

- **Enhance the traffic flow for larger region**: combined to work with GPS, laser radars, sensors to either increase the safety of pedestrians or to warn against potential vehicle traffic threats like irregular traffic flow.

# REFERENCES

[1] [Online]. Available: https://mot.gov.sa/ar/MediaCenter/News/Pages/news912.aspx.

[2] T. index, “riyadh traffic,” [Online]. Available: https://www.tomtom.com/traffic-index/riyadh-traffic#statistics.

[3] “The History of traffic lights,” [Online]. Available: https://www.stoneacre.co.uk/blog/the-history-of-traffic-lights.

[4] APSEd, “traffic signal design websters formula for optimum cycle length,” [Online]. Available: https://www.apsed.in/post/traffic-signal-design-webster-s-formula-for-optimum-cycle-length.

[5] C. o. Europe, “History of Artificial Intelligence,” [Online]. Available: https://www.coe.int/en/web/artificial-intelligence/history-of-ai.

[6] mepei, “Saudi arabia and artificial intelligence,” [Online]. Available: https://mepei.com/saudi-arabia-and-artificial-intelligence/.

[7] P. Analytics, “ What Is Machine Learning: Definition, Types, Applications And Examples,” [Online]. Available: https://www.potentiaco.com/what-is-machine-learning-definition-types-applications-and-examples/..

[8] ibm, “Deep learning,” [Online]. Available: https://www.ibm.com/cloud/learn/deep-learning..

[9] mathworks, “Deep Learning,” [Online]. Available: https://www.mathworks.com/discovery/deep-learning.html.

[10] “Convolutional Neural Networks (CNNs),” [Online]. Available: https://www.happiestminds.com/insights/convolutional-neural-networks-cnns/.

[11] sas, “Computer vision,” [Online]. Available: https://www.sas.com/en_sa/insights/analytics/computer-vision.html.

[12] S. Direct, “Image Classification,” [Online]. Available: https://www.sciencedirect.com/topics/engineering/image-classification.

[13] “Teachable Machine,” [Online]. Available: https://teachablemachine.withgoogle.com.

[14] eInfochips, “Understanding Object Localization with Deep Learning,” [Online]. Available: https://www.einfochips.com/blog/understanding-object-localization-with-deep-learning/.

[15] Mathworks, “Object Detection,” [Online]. Available: https://www.mathworks.com/discovery/object-detection.html.

[16] I. a. A.-N. S. Albatish, “Modeling and controlling smart traffic light system using a rule based system.,” *IEEE,* 2019.

[17] K. M. A. A. S. a. M. L. Yousef, “"Intelligent traffic light scheduling technique using calendar-based history information.",” *Future Generation Computer Systems 91,* 2019.

[18] M. M. H. a. R. S. Razavi, “"Smart traffic light scheduling in smart city using image and video processing.",” *International Conference on Internet of Things and Applications ,* 2019.

[19] A. B. A. a. E. F.-S. Casals, “Adaptive and collaborative agent-based traffic regulation using behavior trees.,” *International Foundation for Autonomous Agents and Multiagent Systems.*

[20] M. S. D. D. R. a. B. N. Gandhi, “Smart control of traffic light using artificial intelligence.,” *IEEE,* 2020.

[21] A. a. V. N. Firdous, “"Smart density based traffic light system.",” *International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions),* 2020.

[22] A. A. I. A. H. M. I. &. A. F. Alaidi, “ Design and implementation of a smart traffic light management system controlled wirelessly by arduino,” 2020.

[23] A. A. S. K. M. A. G. a. F. U. Atta, “An adaptive approach: Smart traffic congestion control system,” *Journal of King Saud University-Computer and Information Sciences,* 2020.

[24] V7, 7 October 2022. [Online]. Available: https://www.v7labs.com/blog/yolo-object-detection#h1.

[25] “yolo object detection explained,” [Online]. Available: https://www.datacamp.com/blog/yolo-object-detection-explained.

[26] https://cutt.us/computervisionYOLOV5, “COMPUTER VISION : YOLO V5— Explained and Demystified,” July 1, 2020. [Online].

[27] https://iq.opengenus.org/yolov5/, “YOLO v5 model architecture,” [Online].

[28] “ YOLOv7,” [Online]. Available: https://arxiv.org/pdf/2207.02696.pdf.

[29] “Foundations of Deep Learning for Object Detection: From Sliding Windows to Anchor Boxes,” [Online]. Available: https://programmathically.com/foundations-of-deep-learning-for-object-detection-from-sliding-windows-to-anchor-boxes/.

[30] “Faster R-CNN Explained for Object Detection Tasks,” [Online]. Available: https://blog.paperspace.com/faster-r-cnn-explained-object-detection/.

[31] A. Singh. [Online]. Available: https://medium.com/augmented-startups/top-6-object-detection-algorithms-b8e5c41b952f.

[32] Viso, “Object Detection,” [Online]. Available: https://viso.ai/deep-learning/object-detection/.

[33] M. Boneh, “Vehicle Detection,” [Online]. Available: https://github.com/MaryamBoneh/Vehicle-Detection.

[34] “Converting a custom dataset from COCO format to YOLO format,” [Online]. Available: https://medium.com/red-buffer/converting-a-custom-dataset-from-coco-format-to-yolo-format-6d98a4fd43fc.

[35] “Roboflow Yolo training,” [Online]. Available: https://blog.roboflow.com/yolov7-custom-dataset-training-tutorial/.

[36] “Yolov5 Official website,” [Online]. Available: https://github.com/ultralytics/yolov5.

[37] “Yolov7 Official website,” [Online]. Available: https://github.com/WongKinYiu/yolov7.

[38] Robowflow. [Online]. Available: https://roboflow.com.

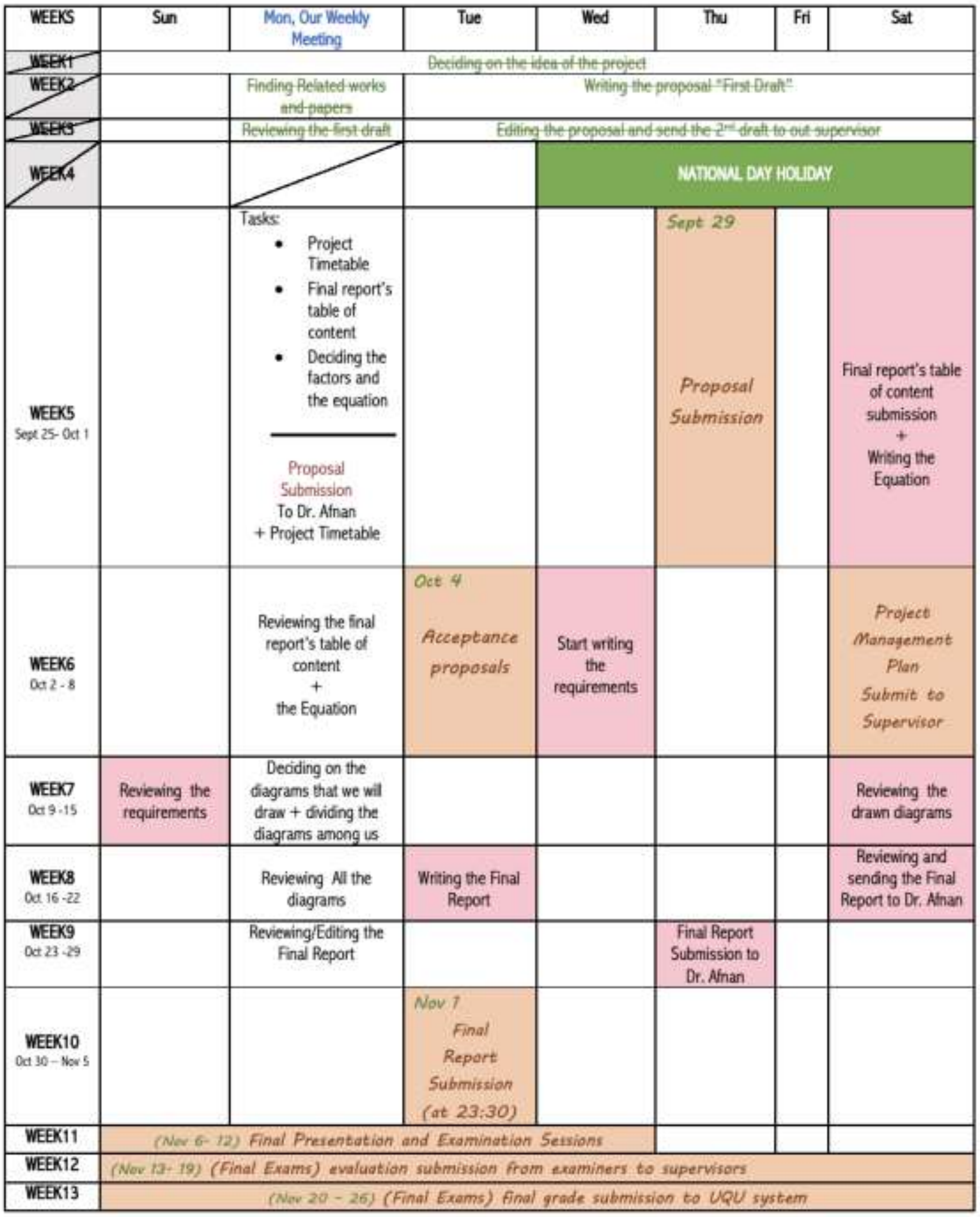
[39] “yolo object detection explained,” [Online]. Available: https://www.datacamp.com/blog/yolo-object-detection-explained.

[40] “yolo object detection explained,” [Online]. Available: //www.datacamp.com/blog/yolo-object-detection-explained.

[41] “YOLOv7,” [Online]. Available: https://arxiv.org/pdf/2207.02696.pdf.

# APPENDIX

## First Semester Timetable

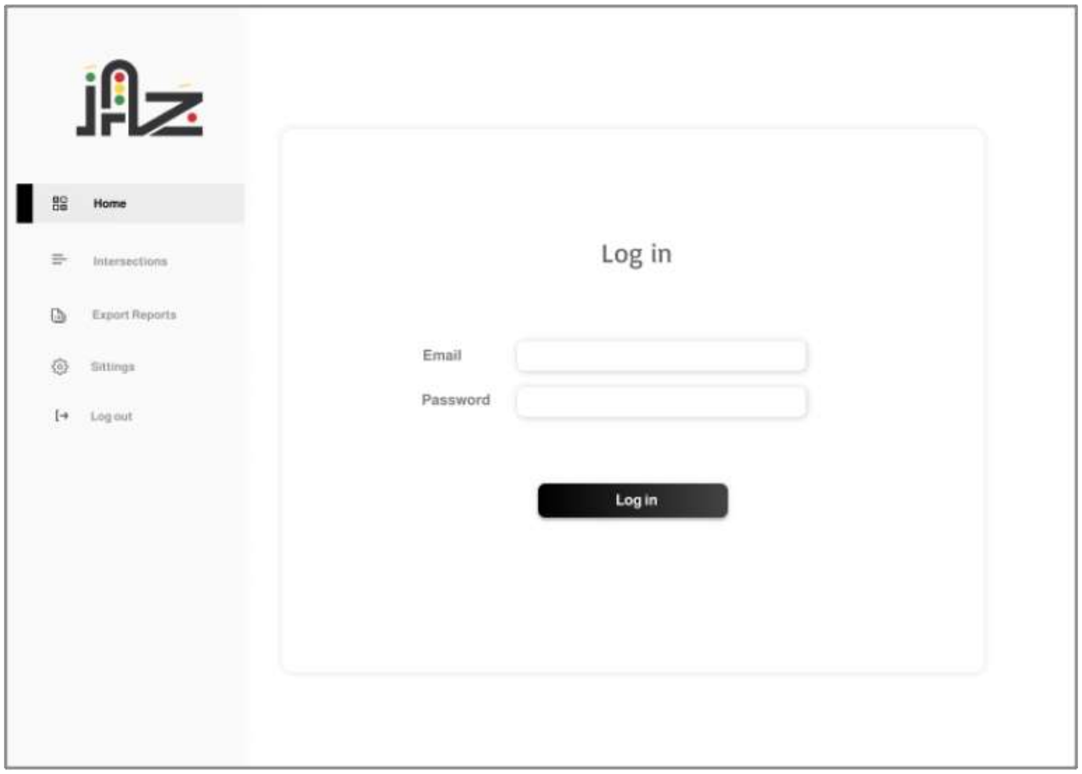


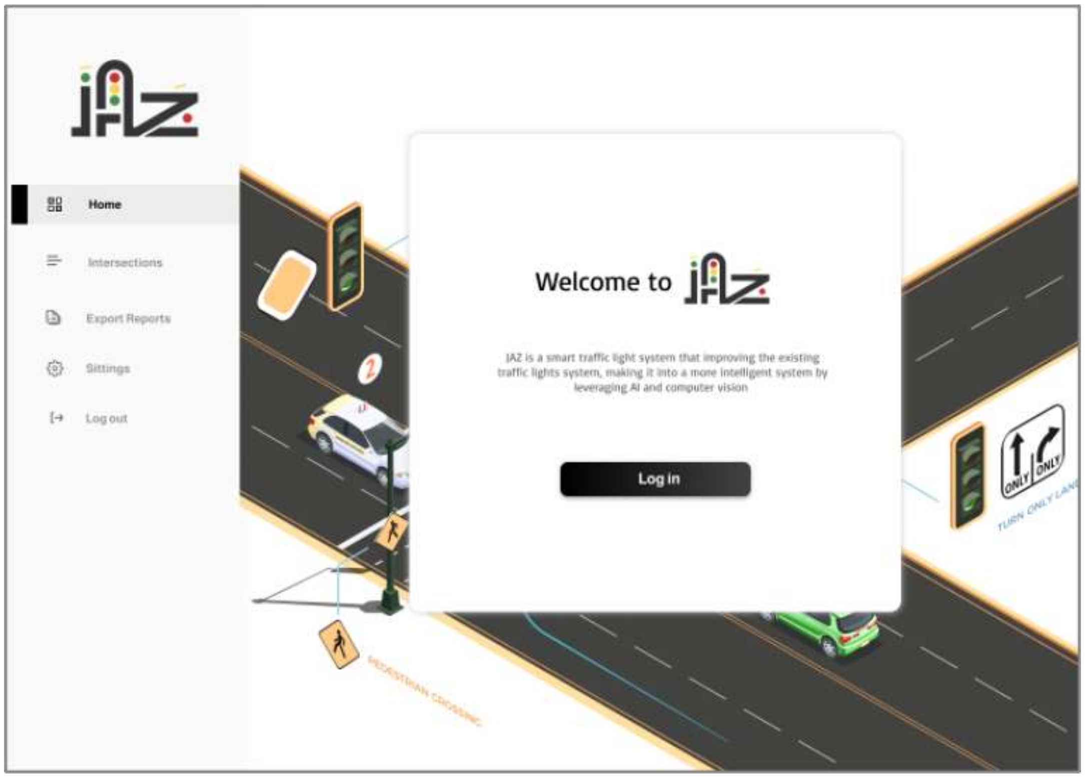
| WEEKS | Sun | Mon, Our Weekly Meeting | Tue | Wed | Thu | Fri | Sat |
|---|---|---|---|---|---|---|---|
| ~~WEEK1~~ | ~~Deciding on the idea of the project~~ | | | | | | |
| ~~WEEK2~~ | | ~~Finding Related works and papers~~ | ~~Writing the proposal "First Draft"~~ | | | | |
| ~~WEEK3~~ | | ~~Reviewing the first draft~~ | ~~Editing the proposal and send the 2nd draft to out supervisor~~ | | | | |
| ~~WEEK4~~ | | | | NATIONAL DAY HOLIDAY | | | |
| WEEK5 Sept 25- Oct 1 | | Tasks: • Project Timetable • Final report's table of content • Deciding the factors and the equation<br>Proposal Submission To Dr. Afnan + Project Timetable | | | Sept 29 *Proposal Submission* | | Final report's table of content submission + Writing the Equation |
| WEEK6 Oct 2 - 8 | | Reviewing the final report's table of content + the Equation | Oct 4 *Acceptance proposals* | Start writing the requirements | | | *Project Management Plan Submit to Supervisor* |
| WEEK7 Oct 9 -15 | Reviewing the requirements | Deciding on the diagrams that we will draw + dividing the diagrams among us | | | | | Reviewing the drawn diagrams |
| WEEK8 Oct 16 -22 | | Reviewing All the diagrams | Writing the Final Report | | | | Reviewing and sending the Final Report to Dr. Afnan |
| WEEK9 Oct 23 -29 | | Reviewing/Editing the Final Report | | | Final Report Submission to Dr. Afnan | | |
| WEEK10 Oct 30 – Nov 5 | | | Nov 7 *Final Report Submission (at 23:30)* | | | | |
| WEEK11 | *(Nov 6- 12) Final Presentation and Examination Sessions* | | | | | | |
| WEEK12 | *(Nov 13- 19) (Final Exams) evaluation submission from examiners to supervisors* | | | | | | |
| WEEK13 | *(Nov 20 – 26) (Final Exams) final grade submission to UQU system* | | | | | | |

# JAZ Timetable

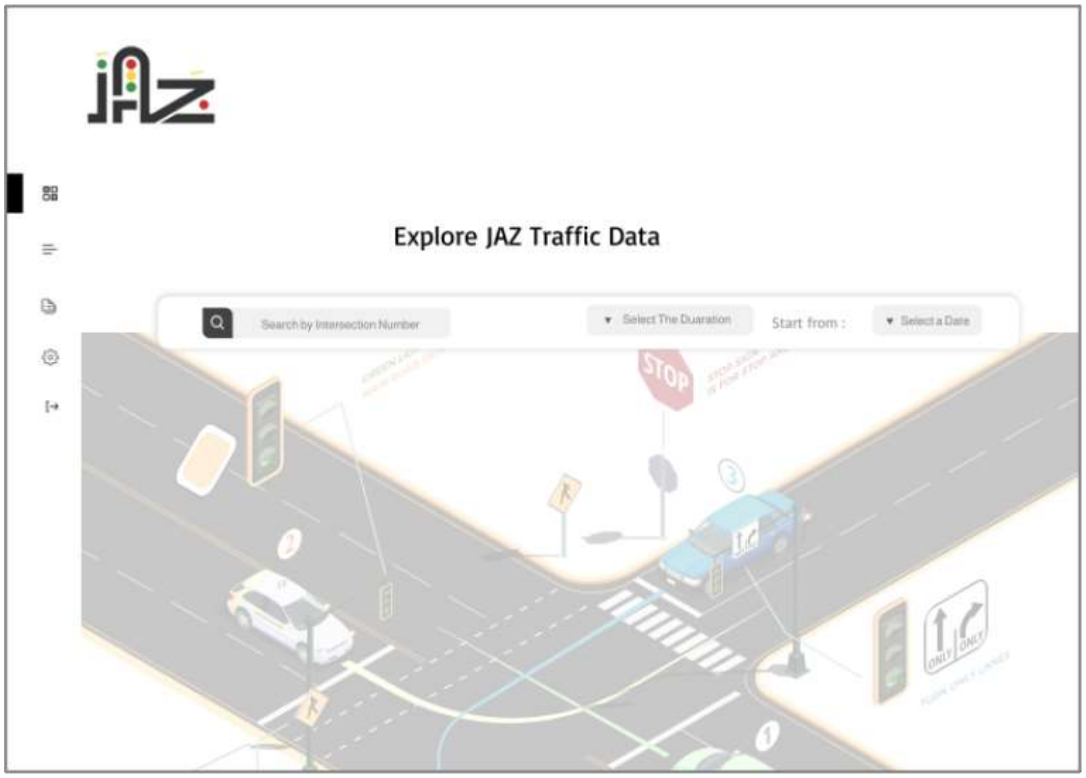
| WEEKS | Sun | Mon | Tues | Weds | Thurs | Fri | Sat |
|---|---|---|---|---|---|---|---|
| W1 (20-25) | | | | • Review the table of content<br>Writing the first 3 chapters | | | |
| W2 (27-3) | Review the Report | | | • Review the flow of JAZ<br>Setting the Base State of JAZ | | | |
| W3 (4-10) | Work with YOLO and find vehicle datasets | | | • View vehicle dataset<br>YOLO Detection and Counting Demo | | | |
| W4 (11-17) | • Run different simulation<br>• Use YOLO in it | Report: Ch5, Detection model YOLO | Report: Ch5, Simulation | • Decide which simulation to use | | "Search for Traffic Density equations and factor" | |
| W5 (18-24) | • Setting the base state equation "Decide which factor for density"<br>• How to deploy it in ML | | | • Equation Review<br>ML Technique Review, ML Dataset | | | |
| W6 (25-30) | • Test the base state on equation | | Report test's result | Write the base state of Scheduling in the Report | | Work on ML Tech | |
| W7 (1-7) | • Review ML Tech<br>• Store data from Detection model into a database | | | Start working on admin's interfaces | | | |
| W8 (8-14) | ML Model Testing<br>Review the report | | | Review the Whole system | | Research paper template and outline | |
| W9 (15-21) | Conference<br>Testing the system | | | Report Ch5, Admin Implementation | | Report Ch6, testing and maintenance | |
| W10 (22-28) | | Report Ch6, Result | | | | Report Ch8, Conclusion + Ch9, Codes | |
| W11 (29-4) | JAZ POSTER | | | JAZ POSTER REVIEW | | | |
| W12 (5-11) | JAZ POSTER + REPORT | | | REVIEWING JAZ SYSTEM | | | |

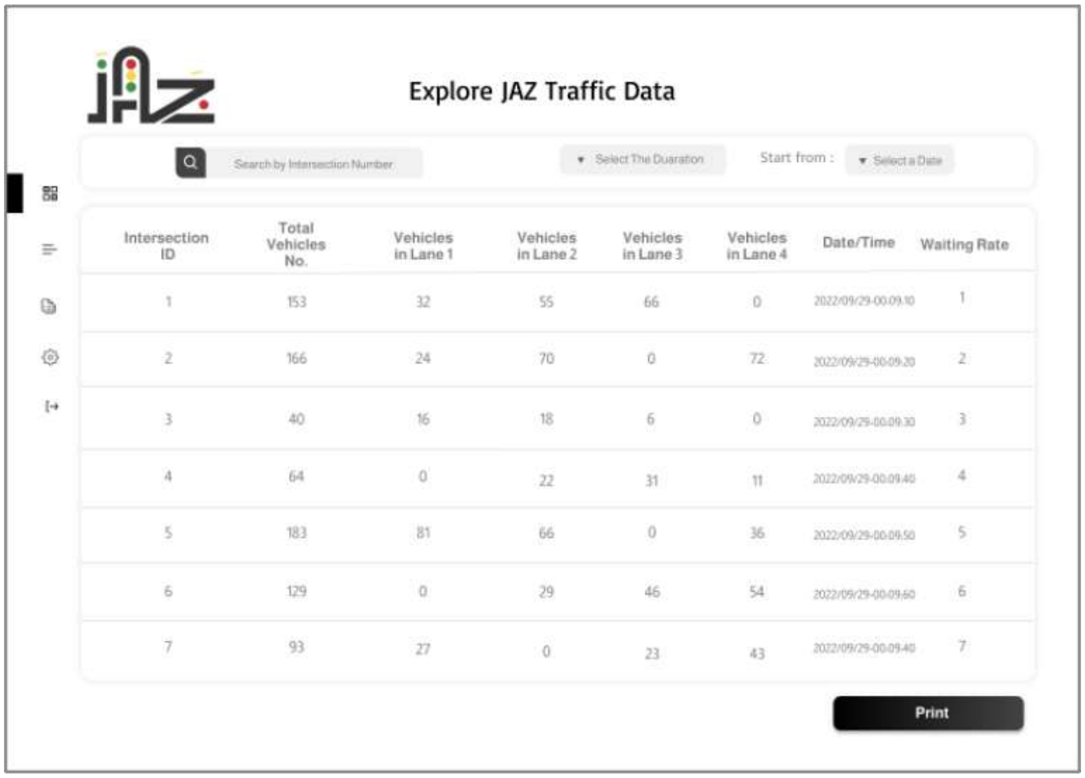
## Admin System Prototype:



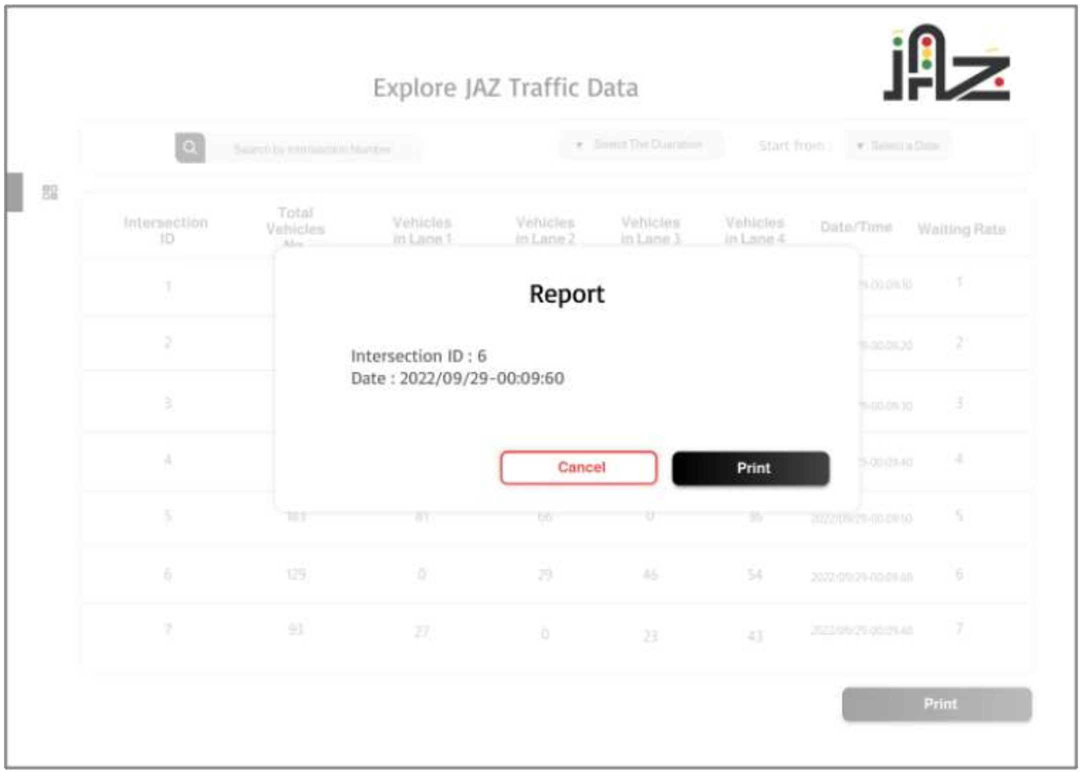


JAZ is a smart traffic light system that improving the existing traffic lights system, making it into a more intelligent system by leveraging AI and computer vision





| Intersection ID | Total Vehicles No. | Vehicles in Lane 1 | Vehicles in Lane 2 | Vehicles in Lane 3 | Vehicles in Lane 4 | Date/Time | Waiting Rate |
|---|---|---|---|---|---|---|---|
| 1 | 153 | 32 | 55 | 66 | 0 | 2022/09/29-00.09.10 | 1 |
| 2 | 166 | 24 | 70 | 0 | 72 | 2022/09/29-00.09.20 | 2 |
| 3 | 40 | 16 | 18 | 6 | 0 | 2022/09/29-00.09.30 | 3 |
| 4 | 64 | 0 | 22 | 31 | 11 | 2022/09/29-00.09.40 | 4 |
| 5 | 183 | 81 | 66 | 0 | 36 | 2022/09/29-00.09.50 | 5 |
| 6 | 129 | 0 | 29 | 46 | 54 | 2022/09/29-00.09.60 | 6 |
| 7 | 93 | 27 | 0 | 23 | 43 | 2022/09/29-00.09.40 | 7 |



Report

Intersection ID : 6
Date : 2022/09/29-00:09:60

| Intersection ID | Total Vehicles No. | Vehicles in Lane 1 | Vehicles in Lane 2 | Vehicles in Lane 3 | Vehicles in Lane 4 | Date/Time | Waiting Rate |
|---|---|---|---|---|---|---|---|
| 1 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | 1 |
| 2 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | 2 |
| 3 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | 3 |
| 4 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | 4 |
| 5 | 183 | 81 | 66 | 0 | 36 | 2022/09/29-00:09:10 | 5 |
| 6 | 129 | 0 | 29 | 46 | 54 | 2022/09/29-00:09:60 | 6 |
| 7 | 93 | 27 | 0 | 23 | 43 | 2022/09/29-00:09:40 | 7 |