



**Bahir Dar University**

**Bahir Dar Institute of Technology**

**Faculty of Electrical and Computer Engineering**

**Department of Computer Engineering**

**PROJECT TITLE**

**AI BASED**

**TRAFFIC VIOLATION DETECTION AND REPORTING SYSTEM**

**EYUEL GASHAYE - BDU1201637**

**FIRAOL TOLESSA - BDU1201719**

**NATNAEL ADAMU - BDU1202627**

**NATNAEL BIRHANU - BDU1202631**

**Advisor: Mr Molla A**

**Date: 01/09/2016 E.C**

## **Declaration**

We are 5<sup>th</sup> year Computer-Engineering students of Bahir Dar University in Bahir Dar institute of Technology (BiT), Faculty of Electrical and Computer Engineering. The information found in this final year project proposal is our original work. And all sources of materials that will be used for the project work will be fully acknowledged.

Name	Signature
------	-----------

1. Eyuel Gashaye	_____
------------------	-------

2. Firaol Tolessa	_____
-------------------	-------

3. Natnael Adamu	_____
------------------	-------

4. Natnael Birhanu	_____
--------------------	-------

This project proposal has been submitted for examination with my approval as a university advisor.

Advisor Name: Mr Molla Atanaw

Advisors Signature: \_\_\_\_\_

Date: 01-09-2016 E.C

## **Acknowledgement**

First and foremost, we Thank God Almighty for granting us the strength, wisdom, and guidance throughout the completion of our project.

We would like to thank Mr. Sultan for his valuable advice and guidance in the selection of our final year project title. His expertise and insights provided us with a clear direction and helped us make informed decisions regarding our project's scope and objectives.

Furthermore we would also like to extend our sincere appreciation to Mr. Molla for his valuable assistance, unwavering support, and valuable suggestions during the development of our final year project proposal on the Traffic Violation Detection and Reporting System. His dedication and expertise greatly contributed to the successful completion of our project proposal.

## **Executive Summery**

In many urban areas, the rise in vehicle population has led to a surge in traffic violations, presenting a significant challenge for law enforcement. Traditional traffic management methods, reliant on manual enforcement, are struggling to cope with the growing volume of violations. This traditional approach not only fails to keep pace with the expanding traffic but also exposes the system to corruption and mismanagement.

Due to the severity of traffic violations, particularly in densely populated urban areas, extensive research has been undertaken to develop and deploy effective technological solutions. These researches have utilized different procedures such as vehicle detection through edge detection and utilized various violation detection AI models. These endeavors typically fall into two categories: those employing non-real-time data analysis and those leveraging real-time data streams. Both approaches rely on a variety of tools including cameras, sensors, and AI models to achieve their objectives.

In this project we have designed a system which incorporates a vehicle detection, license plate recognition and alerting system. The system utilizes a camera module for video data collection, a processing unit where the AI model will be deployed and a centralized data and request coordinating unit.

Our project aims to address these shortcomings by introducing an advanced Traffic Violation Detection and alerting System, leveraging cutting-edge technology to enhance law enforcement capabilities and ensure safer roads for all citizens. This project will, hopefully, contribute a great deal to the reduction of traffic violations and ensure traffic safety procedures. By the use of Flagged license plate detection procedures, the project will make law enforcement procedures more efficient and less demanding on human personal. And as data is key element in this age, the system will provide valuable insights into traffic patterns and violation trends.

## Contents

Declaration.....	I
Acknowledgement .....	II
Executive Summery.....	III
List of Figures .....	VI
List of Tables .....	VII
List of Acronyms.....	VIII
Chapter 1: Introduction .....	1
<b>1.1 Background Information</b> .....	1
<b>1.2 Statement of the Problem</b> .....	2
<b>1.3 Objectives of the Project</b> .....	2
• General Objective .....	2
• Specific Objectives .....	2
<b>1.4 Methodology Used in this Project</b> .....	3
<b>1.5 Major Assumptions Made for the Work</b> .....	5
<b>1.6 Contributions of the Project</b> .....	6
<b>1.7 Scope of the Project</b> .....	6
Chapter 2 Literature Review .....	8
Chapter 3 System Components and Operations.....	10
<b>Packet</b> .....	10
<b>Processing Unit</b> .....	11
<b>UI System Component</b> .....	13
<b>Interface Admin and Organization Interfaces</b> .....	13
<b>System</b> .....	14
Chapter 4.....	15
System Design and Analysis .....	15
<b>Interconnecting Structure</b> .....	15
<b>AI Model</b> .....	18
<b>Comparison of the models</b> .....	18
<b>YOLO</b> .....	21
<b>Training YOLO</b> .....	32

<b>License Plate Recognition and Traffic Violation Detection</b> .....	34
<b>UI Design</b> .....	36
<b>System Requirements(system design )</b> .....	36
<b>Functional Requirements</b> .....	36
<b>User Guide</b> .....	37
<b>Admin</b> .....	37
<b>Organization</b> .....	38
<b>System Workflow</b> .....	38
Chapter 5 Results and Discussions.....	41
<b>Model Evaluation and Testing</b> .....	41
<b>Vehicle Tracking Traffic Violation Detection</b> .....	44
Chapter 6.....	53
Conclusion and Recommendations for future work.....	53
Conclusion.....	53
6.2 Recommendations for Future work.....	53
<b>References</b> .....	55
Appendices.....	56
Appendix A: JSON Web Token .....	56

## List of Figures

Figure 1 System Components .....	10
Figure 2 Data flow .....	10
Figure 3 Object detection models.....	12
Figure 4 System component Interaction .....	15
Figure 5 Thread handling procedure .....	18
Figure 6 Yolo Grid Division .....	22
Figure 7 Bounding Box .....	22
Figure 8 Probability Scores(Confidence score) .....	23
Figure 9 Intersection Over Union.....	24
Figure 10 Non-Max Suppression.....	25
Figure 11 Intersection over union.....	25
Figure 12 Bounding box confidence score.....	26
Figure 13 Class probability .....	27
Figure 14 Yolo Architecture .....	28
Figure 15 Licence plate Annotation in Roboflow .....	33
Figure 16 Training in Colab .....	34
Figure 17 Detected License plate.....	35
Figure 18 Gray scale and Bilateral filter .....	35
Figure 19 Use case diagram .....	37
Figure 20 ENTITY RELATION DIAGRAM .....	40
Figure 21 FI-confidense curve .....	42
Figure 22 Precision-Confidence .....	43
Figure 23 Precision-Recall Curve.....	43
Figure 24 Recall-confidence curve .....	44
Figure 25 Vehicle tracking.....	45
Figure 26 Over Speeding Car.....	46
Figure 27 Traffic light status determining the Traffic light line color .....	47
Figure 28 Traffic light status determining the Traffic light line color .....	47
Figure 29 Red Light Violation detection.....	48
Figure 30 Lane Lines.....	48
Figure 31 Lane violation detection .....	49
Figure 32 Login page .....	49
Figure 33 Violation List.....	50
Figure 34 Update Violation record .....	50
Figure 35 Send SMS .....	51
Figure 36 Organizational Login page.....	52
Figure 37 Organizational flag request issuer .....	52

## List of Tables

<i>Table 1 Functional and non-functional requirements .....</i>	<i>4</i>
Table 2 Faster R-CNN comparison to YOLO .....	18
Table 3 Metrics comparison of Faster CNN and YOLO .....	20
Table 4 Architecture comparison of SSD and YOLO.....	20
Table 5 Metrics comparison of SSD and YOLO .....	20



## **List of Acronyms**

AI : Artificial intelligence

API : Application Programmable Interface

JWT : Json Web Tokens

# **Chapter 1: Introduction**

## **1.1 Background Information**

Traffic safety stands as a critical concern worldwide, and Ethiopia confronts its own challenges in this regard. As the number of vehicles on Ethiopian roads continues to rise, ensuring adherence to traffic regulations poses a formidable task. Beyond mere inconvenience, traffic violations imperil lives, exacerbate congestion, and impede the efficiency of transportation systems [1]. In response, the development of a Traffic Violation Detection and Reporting System, augmented by Flagged Car Detection, emerges as a pivotal endeavor to mitigate risks and enhance road safety standards in Ethiopia.

The escalating volume of vehicles traversing Ethiopian roads underscores the urgency of addressing traffic safety comprehensively. With rapid urbanization and infrastructure development driving this surge in vehicular traffic, the need for effective enforcement mechanisms becomes increasingly apparent [2]. Traffic Violation Detection and Reporting Systems offer a proactive solution by harnessing cutting-edge technologies like computer vision and data analytics to monitor and identify various infractions in real-time. From running red lights to improper lane usage, these systems autonomously detect instances of non-compliance with traffic regulations.

Moreover, the integration of Flagged Car Detection technology fortifies the enforcement capabilities of such systems. By identifying vehicles flagged for specific reasons—such as involvement in previous accidents, outstanding fines, or criminal activities—law enforcement agencies can swiftly address potential risks and uphold public safety.

In Ethiopia, where the challenges of urbanization and infrastructural development intersect with burgeoning vehicular traffic, the implementation of a robust Traffic Violation Detection and Reporting System assumes paramount importance [3]. Beyond its preventive role in averting accidents, this system furnishes law enforcement agencies with invaluable data and evidence for legal proceedings against offenders.

Our project embarks on a comprehensive exploration of the Traffic Violation Detection and Reporting System's design, implementation, and potential impact, with a particular

emphasis on its integration with Flagged Car Detection technology. Through an analysis encompassing pertinent case studies, technological frameworks, and policy implications, we endeavor to elucidate the adaptability of such systems to Ethiopia's distinct traffic management landscape. Ultimately, our aspiration is to contribute to the advancement of road safety initiatives and the cultivation of responsible driving practices for the collective welfare of road users in Ethiopia.

## **1.2 Statement of the Problem**

Urban centers worldwide are facing an unprecedented surge in the number of vehicles, leading to a corresponding increase in traffic violations. These violations not only contribute significantly to road accidents but also cause major inefficiencies in the traffic management system. Conventional methods of traffic management, which heavily rely on limited manpower for monitoring and enforcement, struggle to keep up with this growing volume. Manual enforcement is prone to human error and potential corruption, creating a critical gap between the occurrence of violations and their proper reporting. This gap undermines efforts to ensure road safety and maintain smooth traffic flow. Additionally, it is challenging to promptly identify and track vehicles associated with crimes throughout the city.

The Traffic Violation Detection and Reporting System aims to address these challenges by proposing an automated, AI-based solution. This system efficiently and accurately monitors, detects, and reports traffic violations, thereby bridging the critical gap caused by the surge in vehicle numbers. It also implements a police watch hit list to identify and report vehicles associated with crimes, enhancing overall security.

## **1.3 Objectives of the Project**

- General Objective

Implement a comprehensive traffic violation detection and reporting system to improve traffic efficiency and safety.

- Specific Objectives

- Real-Time Monitoring: To design a system which can continuously monitor traffic conditions and provide information on traffic flow.

- **Violation Detection:** To implement traffic violation detection system for various violations such as red light running, wrong way and Lane Violation using cameras.
- **Police Watch List Hit:** To implement a vehicle identifying and matching mechanism for stolen cars, or suspicious vehicles which are tied up with wanted individuals such as individuals connected to crimes by capturing and identifying flagged license plates from database.
- **Evidence Capture:** To design and implement a recording and capture of violations system that can be used for issuing citations and legal proceedings.
- **Automated Reporting and Alerting:** To implement an automated report generation for detected violations and alerting system for law enforcement personnel in real-time about high-priority violations like police watch list hits or dangerous driving behaviors.
- **Secure Database:** To design and implement a database for storing evidence for detected violations and police watch list.
- **User interface:** To design and implement a user-friendly interface for government bodies to submit capture requests and for system administrators to navigate and manage violations records effectively

## **1.4 Methodology Used in this Project**

The project follow a structured methodology comprising several key steps to ensure comprehensive and effective implementation:

### **1. Literature Review:**

We conduct systematic and comprehensive literature review to gain valuable insights into AI-powered traffic violation detection systems. The review encompasses academic journals, conference proceedings, technical reports, and industry publications. The primary objectives of the review are as follows:

- To investigate existing AI algorithms and techniques used for traffic violation detection.

- To be aware of the best practices for system design, and components interactions.

## 2. System Design

Develop a detailed system architecture that incorporates camera for video feed, image processing modules, and a central database.

## 3. Requirement Gathering

Identification of functional and nonfunctional requirements based on the literature review and current road traffic incidences are done in the next step of the Research method and presentation and presented as follows:

*Table 1 Functional and non-functional requirements*

Functional Requirements	Non Functional Requirements
Plate Detection	Accuracy
License Plate Recognition	Security
Violation Detection	User Friendly Interface
Violation Reporting	
Flagged Plate Detection	
Real-time Alerting	

## 4. Algorithm Selection and Development

In this section of the research method procedure we analyze and compare different AI models like OpenCV ,CNN , YOLO, RNN regarding their Speed of Detection accuracy ,Good generalization ,Open-source .We compared the different AI models by using different range of metrics such as

YOLO is extremely fast because it does not deal with complex pipelines. It can process images at 45 Frames Per Second (FPS). In addition, YOLO reaches more than twice the mean Average Precision (mAP) compared to other real-time systems it also works for real time detection of specific traffic violations and EasyOcr for plate detection.

## **5. Data Collection, pre processing and training:**

In this step we focus on the gathering and preprocessing the data specifically used for training the AI model in detecting license plate. The data collection phase involves identifying and acquiring relevant data sources, such as video footage from surveillance cameras and data library websites. In Pre processing phase normalization, labeling and datasets Splinting are done to the data to ensure its quality and suitability for AI training.

## **6. System Implementation**

Assemble the hardware components and integrate the software modules to create a functional prototype.

## **7. Testing and Validation**

Test the system in a controlled environment to evaluate its performance in detecting violations and identifying flagged vehicles.

### **1.5 Major Assumptions Made for the Work**

- The quality and resolution of the camera footage will be sufficient for accurate detection and recognition of car plates to extract the plate numbers.
- There will be a camera at the traffic violation detection area, especially where there is a traffic light system, and there will be bold, visible lines that are detectable by the camera.
- Adequate network infrastructure will be available to support real-time data transmission and processing between the processing system and the central system.
- There will be cooperation from local authorities and law enforcement agencies for system deployment and data sharing.
- Environmental factors such as lighting and weather conditions will not significantly affect the system's performance.

## **1.6 Contributions of the Project**

The significance of this study lies in its potential to deliver substantial improvements across several key areas such as:

- **Enhanced Traffic Safety:** Through the automated detection of red light running, and other violations, this project can significantly deter dangerous driving behaviors, leading to a demonstrable reduction in the number of accidents and associated casualties.
- **Increased Law Enforcement Efficiency:** By automating violation detection and reporting, the system frees up valuable time for law enforcement personnel. This allows them to focus on higher-priority issues, respond more quickly to emergencies, and dedicate resources to proactive crime prevention initiatives.
- **Reduced Crime:** The system's license plate recognition capabilities can identify stolen vehicles, and vehicles linked to criminal activity. This significantly aids law enforcement in crime prevention and apprehension of criminals, leading to a safer environment for the public.
- **Data-Driven Insights:** The system's data collection capabilities provide valuable insights into traffic patterns and violation trends. This data can be leveraged to inform traffic management strategies, optimize infrastructure development, and drive future improvements to the AI system itself.
- **Pushing the boundaries of AI in traffic management:** also through leveraging the power of AI we are not only showing how AI based solutions can be used to solve problems we as a community face, but also on how to push the boundaries of traffic management to the modern age.

## **1.7 Scope of the Project**

The scope of this project focuses on designing and implementing a Traffic Violation Detection and Reporting System in a demonstration environment. It will integrate components such as a camera unit, image processing module, microcontroller, and central database. It also composes of a system with license plate detection and recognition algorithms to extract license plate information.

Within a demonstration road environment, specifically at traffic light locations, our system will employ algorithms to detect various traffic violations, including red light violations, wrong way driving, and lane violations. Furthermore, the system will automatically notify traffic police upon detecting flagged license plates or any committed violations

The project will primarily focus on car vehicles as the target vehicles for detection and reporting. The system will be trained and tested using a car vehicle data set to ensure accurate recognition and violation detection. However, the project's scope is limited to the demonstration in the specified environment, to showcase its ability to detect and report violations within the specified operational environment.



## **Chapter 2**

### **Literature Review**

Traffic violations have been on the rise lately according to a study carried out by Gadisa Layo in the Urban city center. The violations have been increasing in percentage yearly due to different factors. Management of these issues has long been carried out with traditional traffic management, which is heavily reliant on manual enforcement and is struggling to keep pace with the growing volume of traffic and violations. These management strategies have resulted in an ease of susceptibility to corruption and mismanagement due to there being no advanced technology-based traffic law enforcement [1].

Due to the severity of traffic violations, especially in urban city centers, different research has been conducted on the best ways to design and implement these technologies. Some researches as proposed by Hirnaik et al. [4] have designed a simplistic data flow model on how different layers can be implemented to process video data with distinct separation of layers for step-by-step processing and the utilization of YOLO for fast vehicle detection but lacking an effective way of managing real-time data. Others such as Jin su [5] utilized edge detection and blurring filters to detect features and expanded the range of their violation detected to encompass issues such as no helmet and seatbelt while suffering the same problem as the previous and not having a feature for night time violation detection procedure.

To tackle issues with real-time tracking of vehicles in a dynamic environment Xiaoling et al [6] have used Uses wavelet transform and dynamical background to track objects in real time. However, the model efficiency is not outlined in contrast to other techniques outlined in the document. Furthermore, the studies conducted by Dr. S. Raj Anand and his colleges [7] have shown that real-time processing can be achieved and optimized further. By using techniques such as adaptive differential imaging [5]for detecting object movement real-time processing can be considered as being possible.

However, detecting violations without an effective means of tracing and recording the license plate of the vehicle for reporting procedures won't make the system robust. Tao

Zhang and Yong qi Qi [8] have done research on license plate recognition systems that track and report license plates of vehicles in real time. A study by Saman Rajebi and colleagues [9] also optimized the recognition system by implementing mechanisms for removing environmental effects and uses Hopfield's neural network for pattern recognition.

## Chapter 3

### System Components and Operations

The traffic violation detection and alerting system incorporates different system components which overall interact to achieve the objective set in chapter one. The two major components of the system include central server and the processing unit. The interaction and major functions is outline in the following figure ,

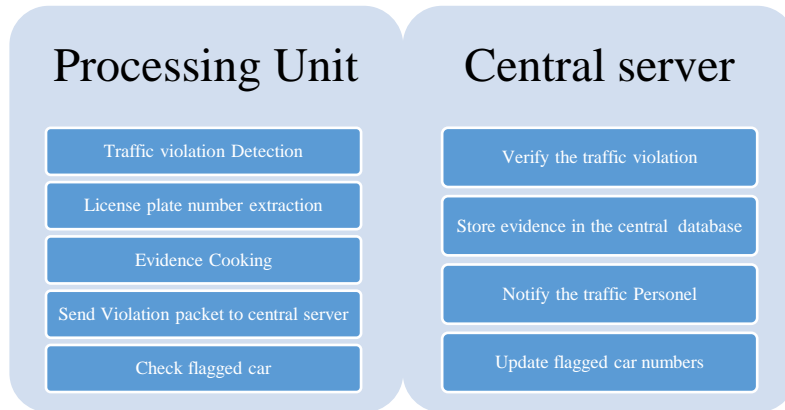


Figure 1 System Components

The data flow through the component is a major aspect considered in the design of the system. The total flow of data is designed as follows,

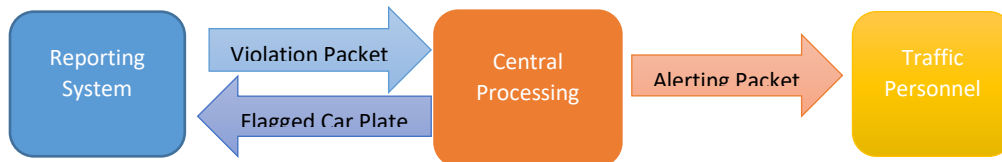


Figure 2 Data flow

#### Packet

The Packets that are sent between the component and the information they hold is,

1. Violation Packet
  - Violation Type
  - Violation date stamp
  - Car plate number
  - Violation Region

2. Flagged car plate number Packet
  - Flagged Plate Number
  - Status Of the car
    - to be removed if car is found
    - registered on search list if car is missed now
3. Alerting Packet
  - Violation Type
  - Car plate number
  - Date stamp
  - Region violation committed

## **Processing Unit**

### **Object Detection and Recognition**

Object detection is a computer vision technique that identifies and classifies a particular object in a particular setting. It involves the classification of objects and the determination of their positions through bounding boxes. The main goal of object detection is to scan digital images or video to locate instances of every object, separate them, and analyze their necessary features for real-time predictions. It is an important part of many applications, such as surveillance, self-driving cars, or robotics.

While Object recognition is a computer vision task that involves identifying and classifying objects within an image or video frame. Unlike object detection, which focuses on locating and drawing bounding boxes around objects, object recognition primarily aims to identify the object and determine its category or label. This process typically involves matching features from the image with pre-trained models to recognize objects based on their shapes, colors, textures, and other visual attributes. In this project context object recognition is applied to recognize letter from detected licence plate.

Object detection algorithms are broadly classified into two categories based on how many times the same input image is passed through a network as One Stage/Single shoot and .Two Stage/Two Shoot

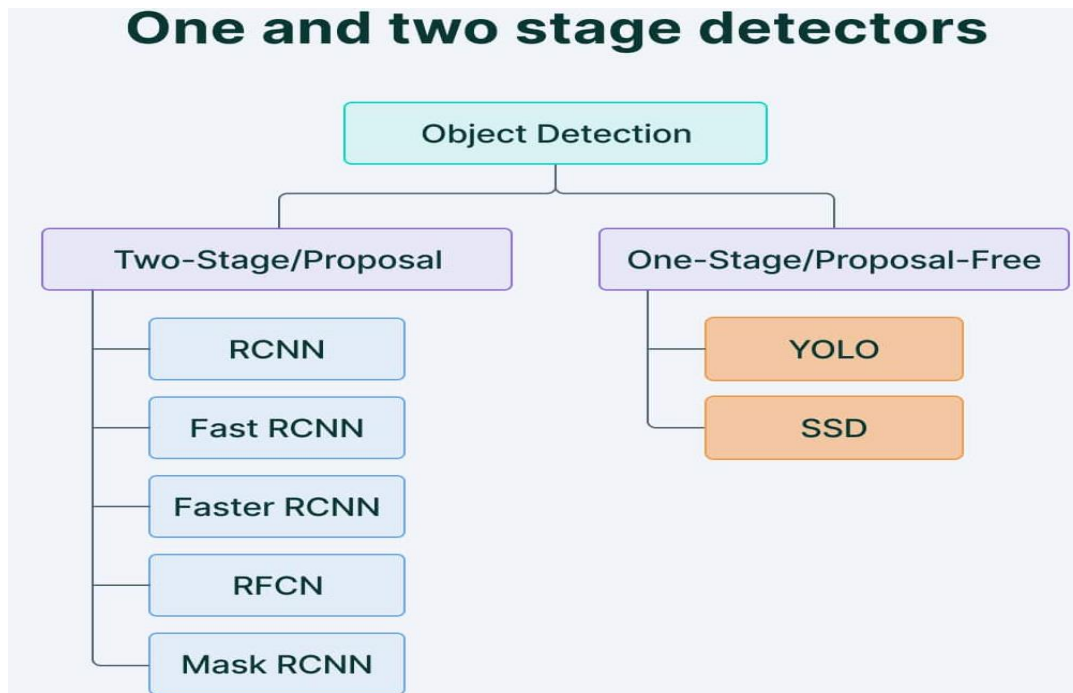


Figure 3 Object detection models

### *One Stage Object Detection*

Single-shot object detection uses a single pass of the input image to make predictions about the presence and location of objects in the image. It processes an entire image in a single pass, making them computationally efficient.

However, single-shot object detection is generally less accurate than other methods, and it's less effective in detecting small objects. Such algorithms can be used to detect objects in real time in resource-constrained environments.

### *Two Stage Object detection*

Two-shot object detection uses two passes of the input image to make predictions about the presence and location of objects. The first pass is used to generate a set of proposals or potential object locations, and the second pass is used to refine these proposals and make final predictions. This approach is more accurate than single-shot object detection but is also more computationally expensive.

Two-shot object detection like R-CNN and its variations, used a pipeline to perform this task in multiple steps. This can be slow to run and also hard to optimize, because each individual component must be trained separately.

## UI System Component

### Interface

#### Admin and Organization Interfaces

##### Admin Interface

##### Features:

- **Dashboard:** Overview of recent violations, flagged car notifications, and system status.
- **Violation Management:** View, filter, and manage reported violations. Includes options to mark violations as reviewed or dismissed.
- **Notification System:** Interface to send SMS notifications to traffic police, with pre-filled violation details.
- **User Management:** Add, remove, and manage user accounts, roles, and permissions.
- **Reports and Analytics:** Generate and view detailed reports and analytics on traffic violations.

##### Technologies Used:

- **Frontend:** React
- **Backend:** Laravel
- **Authentication:** Secure login with role-based access control (RBAC)

##### Organization Interface

##### Features:

- **Login:** Organization logs in to the system using secure credentials.

- **Violation Reporting:** Interface to add report flagged vehicles to the admin.
- **Report Generation:** Create detailed reports with license plate, location, reason, and photos.
- **System Configuration:** Manage camera settings, detection thresholds, and other system parameters.

### Technologies Used:

- **Frontend:** React
- **Backend:** Laravel
- **Authentication:** Secure login with role-based access control (RBAC)

## System

### Reporting and Notification System

The Reporting and Notification System ensures timely communication of traffic violations to traffic police using Twilio for SMS notifications.

### Components:

- **Twilio Integration:** API integration for sending SMS notifications.
- **Notification Logs:** Records of all sent notifications for auditing and tracking.

### Backend and Database Management

The backend is built using Laravel, a robust PHP framework, and the system uses MySQL for database management. The backend handles all API requests, user authentication, data processing, and storage.

### Backend Responsibilities:

- **API Management:** Handle requests from the frontend and provide appropriate responses.
- **User Authentication:** Secure login and session management.
- **Data Processing:** Process and store violation data from the AI Detection Unit.
- **Twilio Integration:** Interface with Twilio API for sending SMS notifications.

## Chapter 4

### System Design and Analysis

#### Interconnecting Structure

The system architecture consists of two major blocks, namely :

1. Processing unit
2. Central server

As this machines run on separate environment there needs to be a reliable way of sharing data between the two blocks. The processing unit is deployed in the traffic region and actively tracks vehicles and checks violation cases , while the central server processes the request coming from the processing unit and regional admins as outlined in the system design.

The system interconnection consists of the transfer of packets , namely called as Meri packets. These packets can originate from the processing or central server block and are transferred through API architecture in the form of JWT.

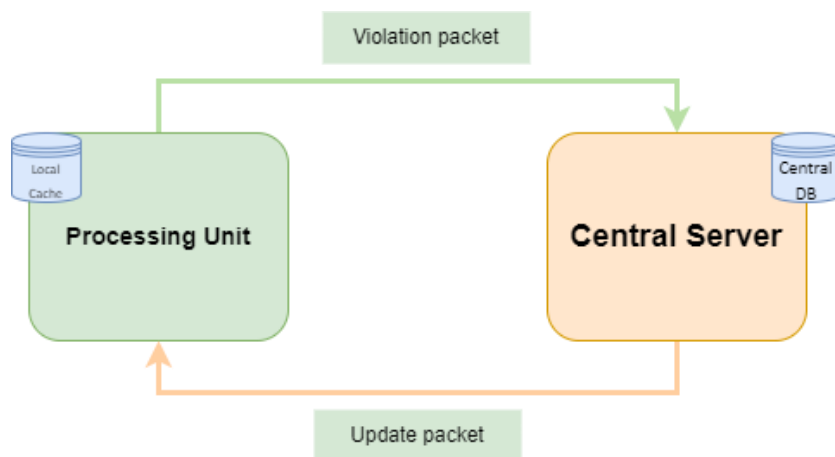


Figure 4 System component Interaction

#### Central server to processing unit

The main objective in regards to this communication is to send update packet requests from the central server to the processing unit. This update packet holds the following structure in the form of JWT. The JWT helps in authenticating and authorizing the request generated



by the server and helps filter requests with malicious intent. It does this by using a common 256-bit symmetric key.

JWT format specification

Header,

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

Payload

```
{  
  "sub": "Meri_Central_Server",  
  "name": "Update_pck",  
  "cmd": COMMAND  
}
```

The local cache holds values of flagged license plates in a local database which is updated regularly from the central server. The update interval is set arbitrarily and once the time is reached JWT encoded packet is constructed in the central server which is signed with a symmetric 256-bit key. Once the packet is constructed it is sent to the processing unit through its POST method running over the network.

The API structure looks like :

*Request(POST) :*

*127.0.0.1:8000/cmd*

*Response:*

```
{  
  "message": "command received"  
}
```

*Error conditions:*

```
{  
  "message": "command not received"  
}
```

```
{  
  "message": "Signature invalid"  
}
```

## Processing unit to Central server

As outlined in the system components the processing unit plays a vital role in this system as it ensures in providing vital traffic data such as violations and license plate images. The processing unit mainly generates two type of packets that will be sent to the central server, violation and flagged packet. The requests are in the form of JWT and a symmetric key is used to validate the request to be sent to the central.

### JWT format specification

```
{
  "alg": "HS256",
  "typ": "JWT"
}
payload = {
  "sub": device_id,
  "name": "violation_pck",
  "type": vtype,
  "lp" : licensePlate,
  "time": timeStamp,
  "location": device_location,
  # Evidence Information
  "evidence": evidence photo
}
```

The API structure looks like:

*Request(POST) :*

*127.0.0.1:8000/pck*

*Response:*

```
{
  "message": "Packet received"
}
```

*Error conditions:*

```
{
  "message": "Packet not received",
  "message": "Signature invalid"
}
```

The handling of requests needs to be redundant as loss of a packet significantly undermines the performance of the system as such proper handling of requests and responses have been set in place to allow for smooth operation. To handle such instances we have incorporated a buffer system which once connection is established and ready to receive it will send them iteratively. The handler of this request runs in a separate thread.

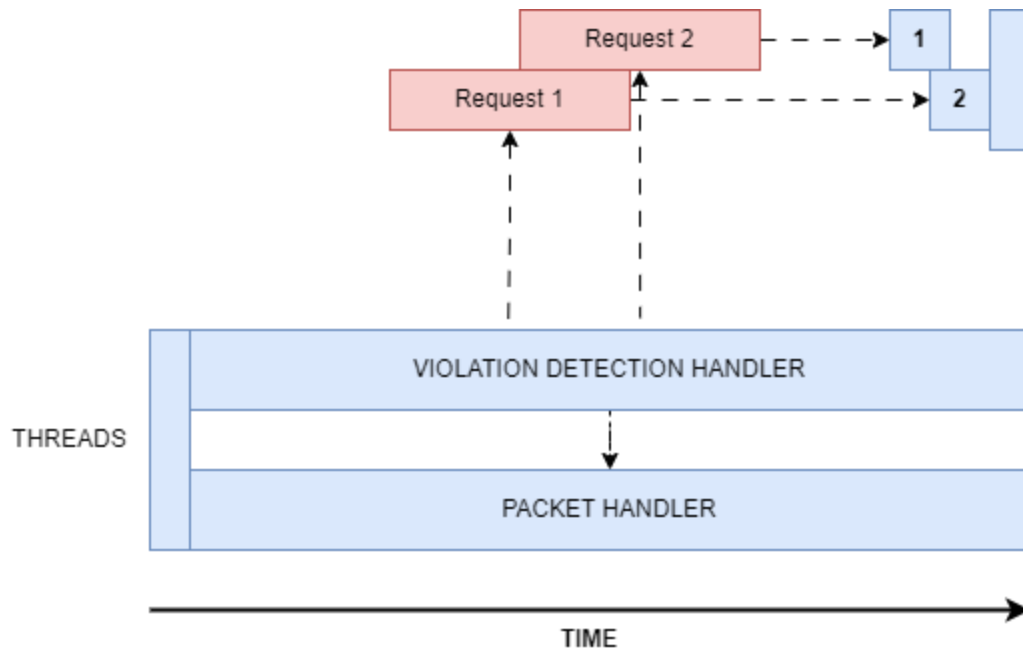


Figure 5 Thread handling procedure

## AI Model

### Comparison of the models

#### YOLO Vs Faster R-CNN

YOLO (You Only Look Once) and Faster R-CNN are both popular object detection algorithms, but they have different approaches, architectures, and trade-offs. Here's a detailed comparison:

Table 2 Faster R-CNN comparison to YOLO

Faster R-CNN	YOLO
Architecture	
<ul style="list-style-type: none"> <li>• <b>Two-Stage Detector:</b> Faster R-CNN is a two-stage detector. <ul style="list-style-type: none"> <li>◦ Stage 1: Region Proposal Network (RPN) generates region proposals.</li> <li>◦ Stage 2: The proposals are fed into a Fast R-CNN network that classifies the proposals and refines their bounding boxes.</li> </ul> </li> <li>• Components:</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Single-Stage Detector:</b> YOLO is a single-stage detector. <ul style="list-style-type: none"> <li>◦ <b>Single Forward Pass:</b> The entire image is processed in a single forward pass through the network.</li> </ul> </li> <li>• <b>Grid-Based Approach:</b> <ul style="list-style-type: none"> <li>◦ The image is divided into a grid, and each grid cell predicts bounding boxes and confidence scores for objects within the cell.</li> </ul> </li> </ul>

<ul style="list-style-type: none"> <li>○ Region Proposal Network (RPN): Generates candidate object proposals.</li> <li>○ Fast R-CNN: Processes these proposals to classify them and refine the bounding boxes.</li> </ul>	<ul style="list-style-type: none"> <li>○ Simultaneously predicts multiple bounding boxes and class probabilities for each grid cell.</li> </ul>
Speed	
<ul style="list-style-type: none"> <li>• <b>Slower:</b> Because it processes region proposals in two stages, it tends to be slower than single-stage detectors.</li> <li>• <b>Higher Latency:</b> More computations due to the two-stage approach lead to higher latency, making it less suitable for real-time applications.</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Faster:</b> Designed for speed, processes the entire image in a single forward pass.</li> <li>• <b>Lower Latency:</b> Highly optimized for real-time performance, making it suitable for applications that require fast processing times.</li> </ul>
Accuracy	
<ul style="list-style-type: none"> <li>• <b>Higher Accuracy:</b> Generally more accurate than YOLO, especially for small objects and challenging detection tasks.</li> <li>• <b>Better Localization:</b> More precise in localizing objects due to the two-stage approach and refinement of bounding boxes.</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Good Accuracy:</b> Achieves good accuracy but generally not as high as Faster R-CNN.</li> <li>• <b>Trade-offs:</b> Sacrifices some accuracy for speed, which can affect detection of small objects and fine details.</li> </ul>
Complexity	
<ul style="list-style-type: none"> <li>• <b>More Complex:</b> The two-stage architecture makes it more complex to implement and train.</li> <li>• <b>Requires More Computation:</b> More computational resources needed due to the two-stage processing.</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Simpler:</b> Single-stage architecture is simpler and more straightforward to implement.</li> <li>• <b>Less Computationally Intensive:</b> More efficient and requires fewer computational resources, making it easier to deploy on edge devices.</li> </ul>
Use Cases	
<ul style="list-style-type: none"> <li>• <b>High-Accuracy Applications:</b> Suitable for applications where accuracy is critical, such as medical imaging, detailed object</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Real-Time Applications:</b> Ideal for real-time applications such as surveillance, autonomous driving, and live video processing.</li> </ul>

<p>recognition, and high-resolution images.</p> <ul style="list-style-type: none"> <li>• <b>Offline Processing:</b> Better suited for scenarios where processing speed is less critical, and accuracy is the priority.</li> </ul>	<ul style="list-style-type: none"> <li>• <b>Resource-Constrained Environments:</b> Works well on devices with limited computational power, like mobile phones and embedded systems.</li> </ul>
---	--

### Metrics for Comparison

Table 3 Metrics comparison of Faster CNN and YOLO

Metric	Faster R-CNN	YOLO
Inference Time	Higher (slower)	Lower (faster)
Mean Average Precision (mAP)	Generally higher	Competitive but usually lower
Localization Accuracy	High	Moderate to high
Recall	High	Moderate
Precision	High	Moderate to high
Flexibility	High (better with small objects)	Moderate to high
Ease of Implementation	Moderate to complex	Simpler

### YOLO Vs SSD

The comparison of compare SSD (Single Shot MultiBox Detector) and YOLO (You Only Look Once), two popular single-stage object detection algorithms, is outlined in this table.

### General Architecture

Table 4 Architecture comparison of SSD and YOLO

Aspect	YOLOv8	SSD
<b>Architecture</b>	Single-stage detector: Improved grid-based approach	Single-stage detector: Multi-scale feature maps
<b>Speed</b>	Very fast, real-time processing	Fast, real-time processing
<b>Accuracy</b>	High, better than previous YOLO versions	High, especially for small and varying-sized objects

### Metrics for Comparison

Table 5 Metrics comparison of SSD and YOLO

Metric	YOLOv8	SSD
--------	--------	-----

<b>Inference Time</b>	Lower (faster)	Low (fast)
<b>Mean Average Precision (mAP)</b>	High, improved over previous YOLO versions	High, especially with small objects
<b>Localization Accuracy</b>	High	High
<b>Recall</b>	High	High
<b>Precision</b>	High	High
<b>Flexibility</b>	High, handles various object sizes well	High, especially with anchor boxes
<b>Ease of Implementation</b>	Simple to implement and train	Moderate, due to anchor box tuning

## YOLO

YOLO was developed by Joseph Redmond that offers end-to-end network YOLO, which stands for "You Only Look Once," is a state of the art, real-time object detection system. It frames object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities.

## How does Yolo Works

### Image Division and Grid System

#### Grid Division

YOLO divides an input image into an  $S \times S$  grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts  $B$  bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

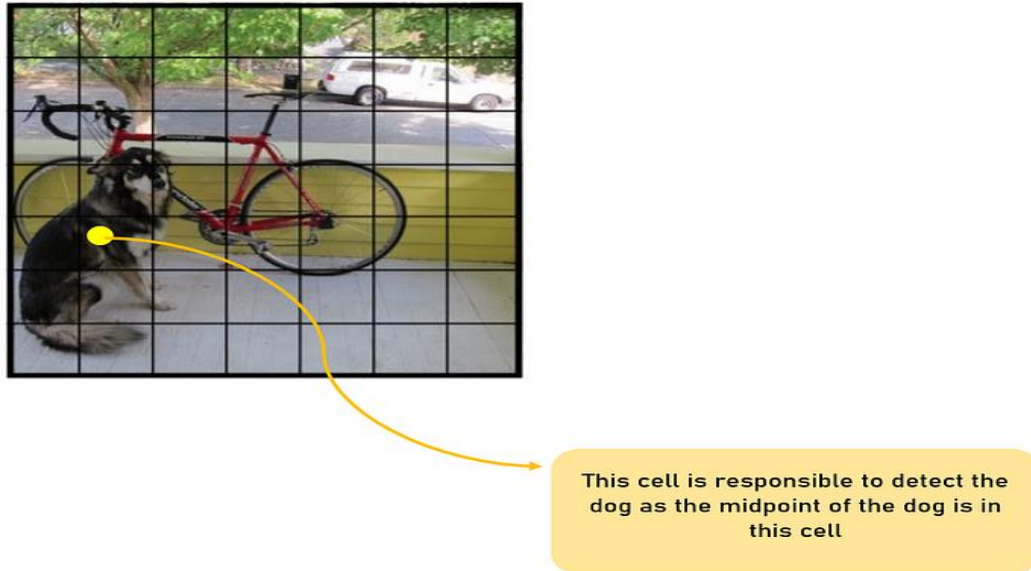


Figure 6 Yolo Grid Division

### Bounding Box Regression

The next step is to determine the bounding boxes which correspond to rectangles highlighting all the objects in the image . We can have as many bounding boxes as there are objects within a given image. YOLO determines the attributes of these bounding boxes using a single regression module in the following format, where Y is the final vector representation **for each bounding box**.

$$Y = [pc, bx, by, bh, bw, c1, c2, c3 \dots cn]$$

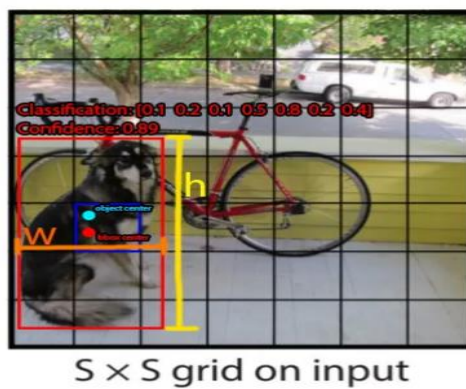


Figure 7 Bounding Box

For  $S \times S$  grid system bounding boxes anchors  $B=2$  and there

The system models detection as a regression problem. It divides the image into an  $S \times S$  grid and for each grid cell predicts  $B$  bounding boxes(anchor Boxes) in the figure above we have 2 anchor box  $B$  because we have red and blue bounding box to work with.

we have 5 parameters in the vector representation of  $Y$  confidence for those boxes, and we can have  $n$  different classes  $C$  to be predicted.

These predictions are encoded as an  $Y=S \times S(B * 5 + C)$  tensor.

$p_c$  corresponds to the probability score or confidence score of the grid containing an object. For instance, in the following picture all the grids in red will have a probability score higher than zero in. The image on the right is the simplified version since the probability of each yellow cell is zero (insignificant)

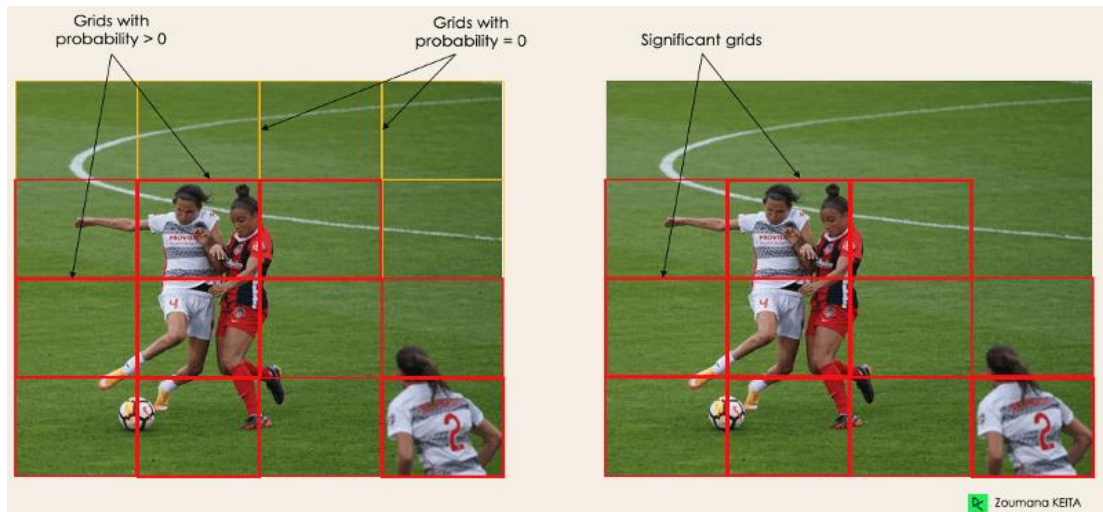


Figure 8 Probability Scores(Confidence score)

$bx$ ,  $by$  are the  $x$  and  $y$  coordinates of the center of the bounding box **with respect to the enveloping grid cell** for instance if the grid cell's size is 1 unit and the center of the bounding box is at the center of the grid cell, then  $bx$  and  $by$  would both be 0.5  **$bx$  and  $by$**  are normalized to be between 0 and 1, where 0,0 represents the left/top edge of the cell, and 1,1 represents the right/bottom edge of the cell.  **$bh$ ,  $bw$**  correspond to the height and



the width of the bounding box with respect to the normalized(between 0 and 1) to the entire image. **c1 and c2** correspond to the two classes **Player and Ball** in the above image case. We can have as many classes we want to predict.

### Intersection Over Unions or IOU

Most of the time, a single object in an image can have multiple grid box(object detection models are responsible to draw this bounding boxes) candidates for prediction, even though not all of them are relevant. IOU is a measure of the overlap between two bounding boxes. A threshold is typically set for IOU to determine if two boxes are referring to the same object (e.g., if  $\text{IOU} > 0.5$ , the boxes are considered to overlap significantly). The goal of the IOU (a value between 0 and 1) is to discard such grid boxes to only keep those that are relevant. Here is the logic behind it: The user may defines its IOU selection threshold, which can be, for instance, 0.5. Then YOLO computes the IOU of each grid cell which is the Intersection area divided by the Union Area. Finally, it ignores the prediction of the grid cells having an  $\text{IOU} \leq \text{threshold}$  and considers those with an  $\text{IOU} > \text{threshold}$ . Iou is used in the calculation of confidence score and in the Non maximum suppression stage

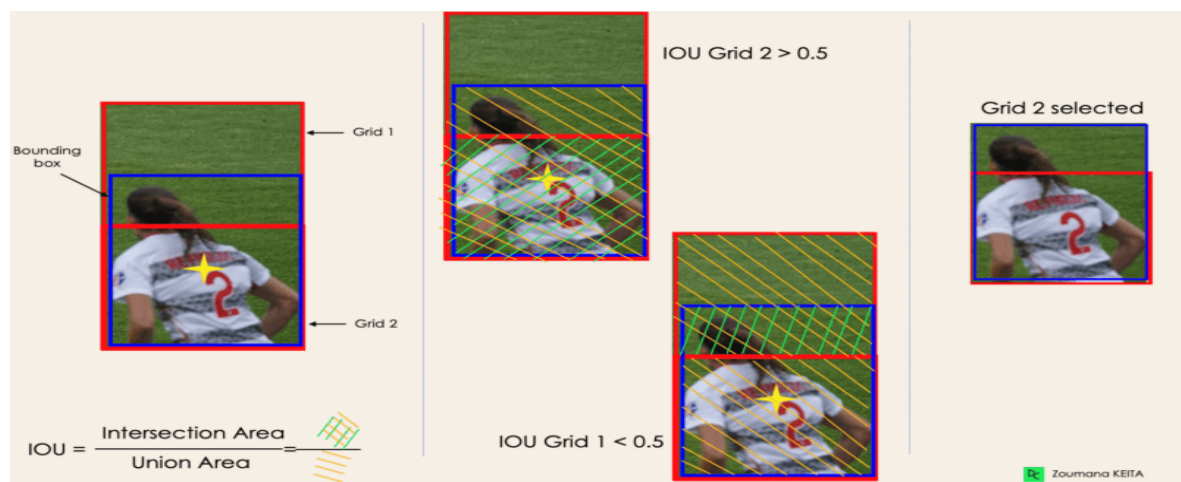


Figure 9 Intersection Over Union

### Combining Predictions and Non-Max Suppression or NMS

The final detection involves combining the bounding box predictions and class probabilities as seen in the example 1 in figure 3. The class-specific confidence score is

calculated by multiplying the confidence score for each bounding box by the class probabilities.

Setting a threshold for the IOU is not always enough because an object can have multiple boxes with IOU beyond the threshold, Non-Maximum Suppression (NMS) is a post-processing technique used in object detection algorithms to reduce the number of overlapping bounding boxes and improve the overall detection quality. *Object detection algorithms typically generate multiple bounding boxes around the same object with different confidence scores.* NMS filters out redundant and irrelevant bounding boxes, keeping only the most accurate ones.



Figure 10 Non-Max Suppression

In the final prediction stages if we have  $n$  classes to predict non max suppression is applied to each independently and run  $n$  times based on the number of class that we are going to predict

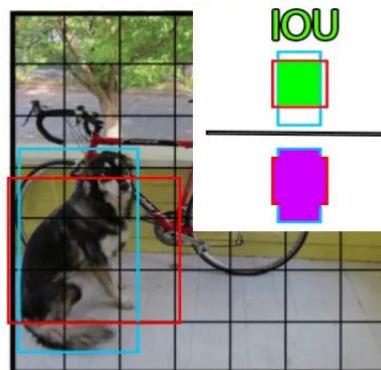


Figure 11 Intersection over union

The above image described IOU area of intersection of the ground truth and predicted box in green divided by the area of the union of the two boxes, in purple. This will be between 0 and 1, 0 if they don't overlap at all, and 1 if they are the same box. Therefore, a higher IOU is better as it is a more accurate prediction

### Example

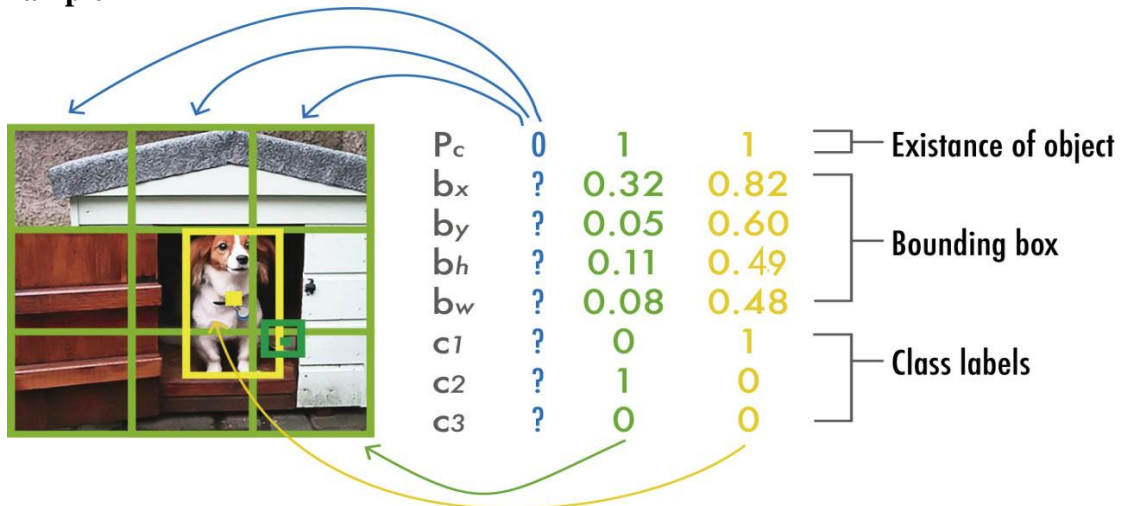


Figure 12 Bounding box confidence score

In the above example the model is have class probability of zero for the upper right grid so that if any arbitrary value is predicted these numbers are considered as insignificant. If we see the predicted Bounding Box 2 of the above image we can clearly see that

- $P_c=1$ : The confidence score that an object exists in this bounding box.
- $b_x=0.82$ : The x-coordinate of the bounding box center is 82% across the grid cell width from the left of the cell.
- $b_y=0.60$ : The y-coordinate of the bounding box center is 60% down the grid cell height from the top of the cell.
- $b_w=0.48$ : The width of the bounding box is 48% of the total image width.
- $b_h=0.49$ : The height of the bounding box is 49% of the total image height.
- $c_1, c_2, c_3$ . These represent the probabilities that the object detected belongs to a particular class in the above image scenario the  $c_2$  class has the probability one which shows the Dog which have class 2 certainly found .

For  $P_c$  The confidence score for the bounding box.  **$b_x, b_y, b_h, b_w$**  bounding box parameters.  **$c_1, c_2, c_3$**  The class probabilities.

In the above example for the yellow bounding box

**Confidence Score (pc) = P(Object) × IoU** where

P(Object) is the probability that an object is present in the bounding box.

IoU (Intersection over Union) measures the overlap between the predicted bounding box and the actual ground truth box.

The confidence score (1 in this case) is used in conjunction with the class probabilities to determine the final prediction.

The model will predict a bounding box around the object with coordinates (0.82,0.60) and size (0.49,0.48), and it will classify this object as a dog (class 2) with high confidence.

The diagram below is another image of all the bounding boxes and class predictions that would actually be made and their final result.

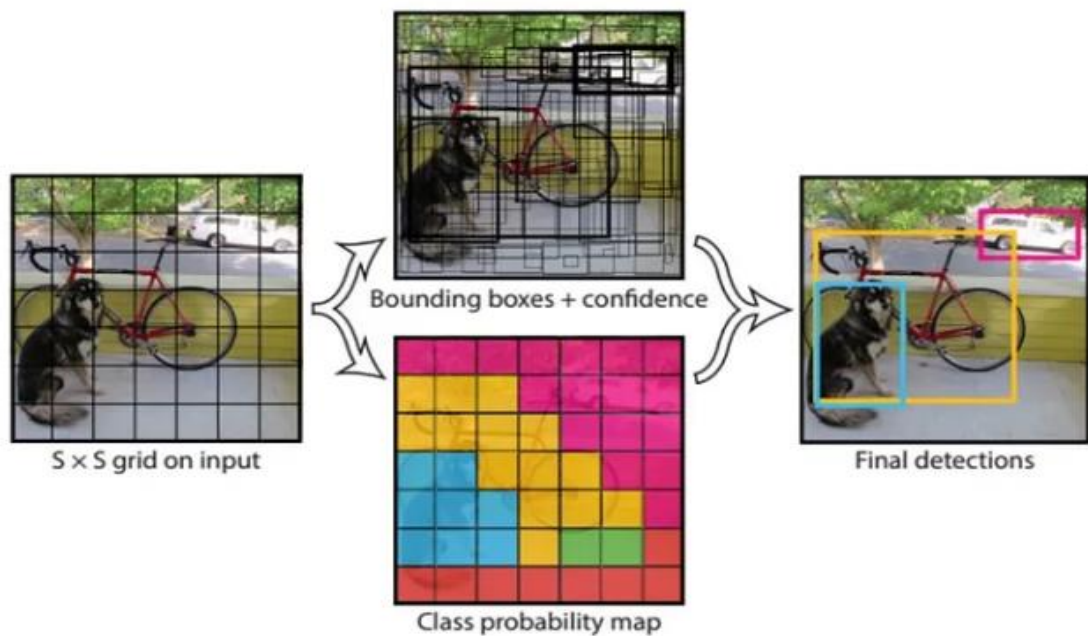


Figure 13 Class probability

### YOLO Overall Architecture

YOLO's architecture consists of a single convolutional neural network (CNN) that processes the entire image in one go. This end-to-end approach is what makes YOLO significantly faster than other object detection systems.

The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network(CNN) to detect objects in the image. In the YOLO architecture, the convolutional layers play a crucial role in extracting features from the input image. These features are then used to predict bounding boxes and class probabilities for object detection. The architecture of the CNN model that forms the backbone of YOLO is shown below.

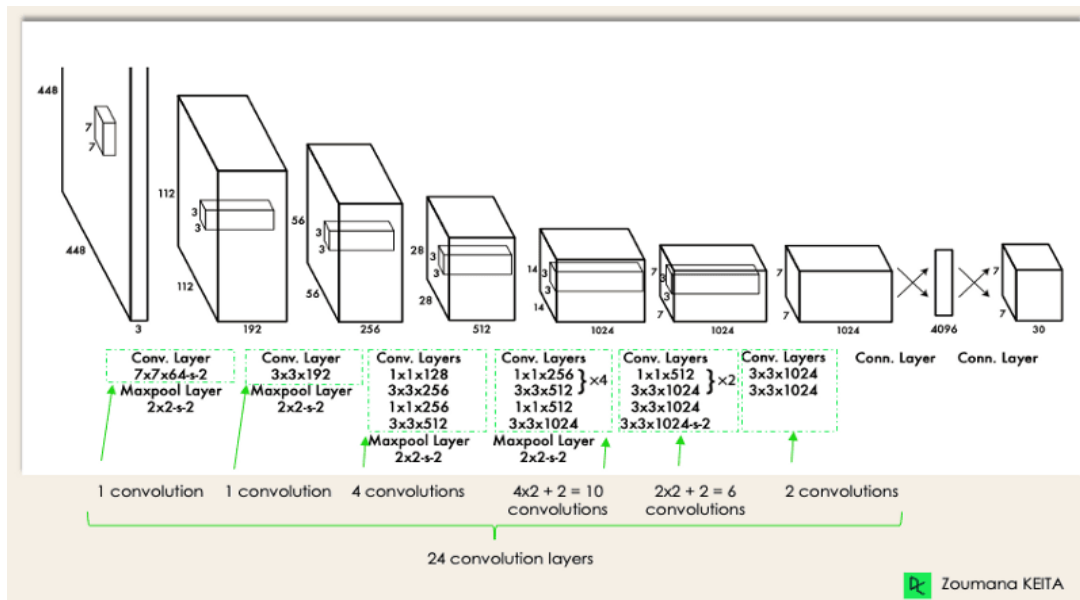


Figure 14 Yolo Architecture

The YOLO architecture is composed of a series of convolutional layers followed by fully connected layers, designed to perform object detection in a single forward pass through the network. There are common terminologies that are used to explain the yolo architecture which are max pool and filter

**Max pooling** :is a down sampling technique commonly used in convolutional neural networks (CNN) to reduce the spatial dimensions of the input, which helps in reducing the computational cost and controlling over fitting. It works by dividing the input into non-overlapping regions (usually squares) and taking the maximum value from each region

**Filters (n x n)**

Filters, also known as kernels, are small matrices (e.g., 3x3, 5x5) that move across the input image to detect various features and creates a feature map by capturing the presence of specific features at different spatial locations.

Each filter detects a specific type of feature, such as edges, textures, or patterns. The filter performs an element wise multiplication (convolution operation) with the input data and sums the results to produce a single output value. Multiple filters are used to generate multiple feature maps, each highlighting different aspects of the input image.

### **Input Layer**

The input size of the network is a 448 x 448 RGB image. This size is chosen to provide a balance between computational efficiency and detection accuracy.

### **Convolutional Layers**

The network consists of a total of 24 convolutional layers, organized into different blocks. Convolutional layers are responsible for feature extraction from the input image.

#### **First Convolutional Block:**

- Convolutional Layer: 7 x 7 filter, stride 2, 64 filters
- Maxpool Layer: 2x2, stride 2
- Output Size: 112 x 112 x 64

The purpose is to initial feature extraction with a large receptive field to capture more context.

#### **Second Convolutional Block:**

- Convolutional Layer: 3x3 filter, 192 filters
- Maxpool Layer: 2x2, stride 2
- Output Size: 56 x 56 x 192

The purpose is to Increase the depth of the network while reducing spatial dimensions to focus on important features.

### **Third Convolutional Block**

➤ Convolutional. Layer

1 x 1 filter, 128 filters

3 x 3 filter, 256 filters

1 x 1 filter, 256 filters

3 x 3 filter, 512 filters

➤ Maxpool Layer: 2x2, stride 2

Output Size: 28 x 28 x 512

The purpose is to further increase the depth and complexity of the feature maps.

### **Fourth Convolutional Block**

➤ Convolutional. Layer

1x1 filter, 256 filters x 4

1x1 filter, 512 filters

3x3 filter, 512 filters x 4

3x3 filter, 1024 filters

➤ Maxpool Layer: 2x2, stride 2

➤ Output Size: 14 x 14 x 1024

The purpose is to Enhance the network's ability to detect objects at different scales and handle more complex features.

#### **Fifth Convolutional Block:**

- Convolutional Layers

1 x 1 filter ,512 filters x 2

3 x 3 filter, 1024 filters x 3

- Output Size: 14 x 14 x 1024

The purpose of this layer is to deepen the network to improve the ability to detect intricate patterns.

#### **Sixth Convolutional Block**

- Convolutional Layers

3x3 filter, 1024 filters x 2

- Output Size: 7x7x1024

This layer serve as a final layer for feature extraction before transitioning to fully connected layers.

#### **Fully Connected Layers**

- **Flattening:** The output from the final convolutional layer is flattened to a 1D vector.
- Fully Connected Layers
  - ✓ Layer 1: 4096 units, activation function (typically ReLU)
  - ✓ Dropout: Dropout layer with a probability of 0.5 to prevent over fitting.



- ✓ Layer 2: 4096 units, activation function (typically ReLU)
- ✓ Output Layer: Size corresponds to the number of bounding boxes and class probabilities (e.g.,  $7 \times 7 \times (2 \times 5 + 20)$  for  $7 \times 7$  grid, 2 boxes per grid, 5 values per box, and 20 classes).

The final output consists of **bounding box coordinates**, **confidence scores** for each **bounding box**, and **class probabilities** for each predicted object.

## Training YOLO

### *Data Set Preparation and Annotation*

#### **Data Set Collection**

The backbone of the yolo model is pre-trained object detection model on coco data set Here the task was to train the model with a licence plate data set. The data set was collected from various sources on the internet like keggel, roboflow and google (links are provided in the appendix section) and compile together for per-processing. The selection criteria included diversity in lighting conditions, angles, blurred and far vehicles to ensure a robust model and accuracy.

#### **Annotation Process**

The collected images were uploaded to Roboflow for annotation. Roboflow provides an intuitive interface for labeling images, making it easy to annotate large datasets accurately. Using the following steps

1. Uploading prepared data set Images to Roboflow platform
2. Pre-Process the data set by cropping, adaptive thresholding (light adjustments) and others were done in this step to make the data fit for training
3. Then annotating Images using Roboflow's annotation tools, vehicles Licence plate were labeled in the images. The interface allowed for precise and efficient annotation of the data set.
4. Classify the data set to test, train and valid data set using 80:10:10 proportion by diversifying the datasets to each group based on its proportion

5. Exporting the annotated dataset: Once the annotation was complete, the data set was exported in the YOLOv8 format. This export included the necessary configuration for training the YOLOv8 model.

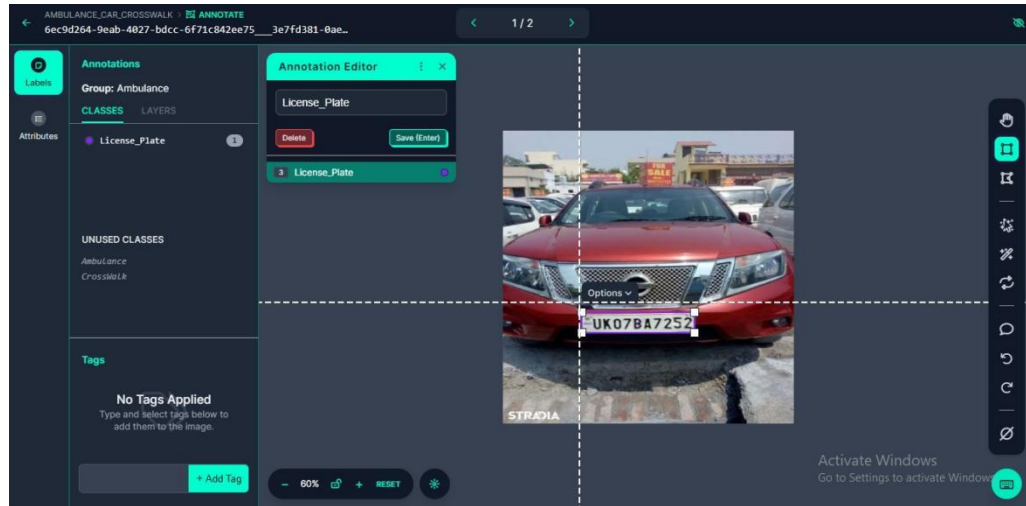
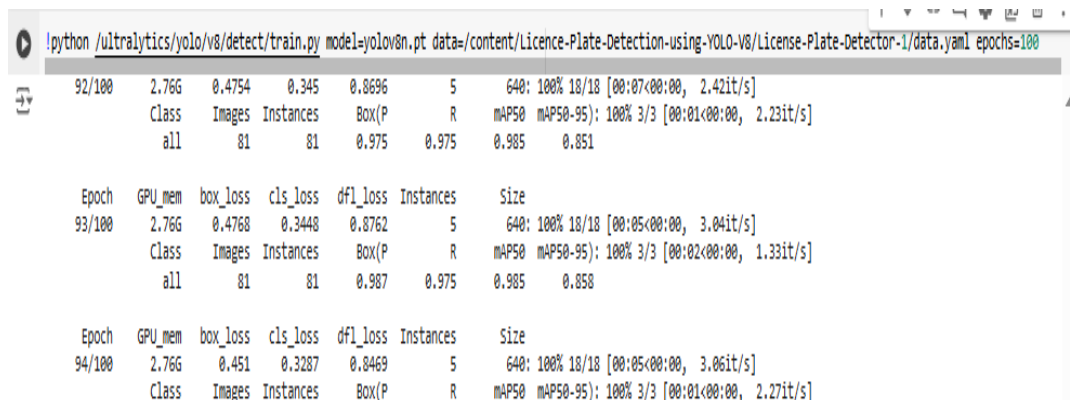


Figure 15 Licence plate Annotation in Roboflow

## Training Process

The training process of the YOLOv8 model was conducted in Google Co lab because the platform provide free GPU make the training process fast and efficient. Training the custom per-trained YOLOv8 model was conducted as follows.

1. Setting up Google colab then open a new notebook and change the run time to T4-Gpu `to accelerate the training process.
2. Install the required package and dependencies in the requirement.txt file of the which includes like Ultralytics ,OpenCv,numpy,panda,pytesseract,hydra-core,tensor board and other that are required in the training process.
3. Downloading the annotated datasets in YOLOv8 format directly to google colab using the roboflow script.
4. Using the train.py python script and the per-trained yolov8.pt and the data.yaml file of the the data set as a parameter by adjusting the image and batch size training the model for 100 epochs



```
python /ultralitics/yolo/v8/detect/train.py model=yolov8n.pt data=/content/Licence-Plate-Detection-using-YOLO-V8/License-Plate-Detector-1/data.yaml epochs=100
```

92/100	2.76G	0.4754	0.345	0.8696	5	640: 100% 18/18 [00:07:00:00, 2.42it/s]
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 3/3 [00:01:00:00, 2.23it/s]
all	81	81	0.975	0.975	0.985	0.851
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
93/100	2.76G	0.4768	0.3448	0.8762	5	640: 100% 18/18 [00:05:00:00, 3.04it/s]
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 3/3 [00:02:00:00, 1.33it/s]
all	81	81	0.987	0.975	0.985	0.858
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
94/100	2.76G	0.451	0.3287	0.8469	5	640: 100% 18/18 [00:05:00:00, 3.06it/s]
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 3/3 [00:01:00:00, 2.27it/s]

Figure 16 Training in Colab

After training for 100 epochs the weight of the training is saved to the run folder of the Current folder inside run/detect/weights folder as best.pt then this model is latter use to predict the bounding box of vehicles license plate in the ANPR Stage that will be explained in the coming section

## License Plate Recognition and Traffic Violation Detection

### *License Plate Detection and Recognition*

This section, focused on the detection of license plates from traffic videos sample used . We utilize a per-trained YOLOv8 model with custom-trained weights (best.pt) that results from the above training specifically designed for license plate detection.

### **License Plate Detection**

The best.pt weights, trained on a data set of vehicle images with annotated license plates, are used to detect license plates in the input frames. Each frame is processed, and bounding boxes are drawn around detected license plates with confidence scores. In the testing phase



*Figure 17 Detected License plate*

#### Pre-processing the Detected License Plate

- The detected license plate region is converted to a **Grayscale** image to simplify the image and reduce computational complexity.
- Using Noise reduction techniques such as **Bilateral Filter** are applied to remove any unwanted noise from the image.
- Edge detection algorithms like **Canny Edge Detection** are applied to highlight the boundaries of the characters on the license plate.
- Thresholding methods like **Otsu's Thresholding** are applied to create a binary image, making the characters more distinct and easier for OCR to read.



*Figure 18 Gray scale and Bilateral filter*

#### Optical Character Recognition using Tesseract OCR

Once the license plate regions are pre-processed, they are passed to Tesseract OCR for text recognition. The process involves

## **UI Design**

### **System Requirements(system design )**

#### **Functional Requirements**

1. **Violation Detection and Reporting System**
  - **Red Light Running Detection**
    - Detect vehicles that run red lights.
    - Capture license plate, location, date/time, and image of violation.
  - **Line Violation Detection**
    - Detect vehicles that cross lane lines improperly.
    - Capture license plate, location, date/time, and image of violation.
  - **Flagged Cars Detection**
    - Identify and capture information of flagged cars.
    - Capture license plate, location, date/time, and image of flagged cars.
2. **Data Processing and Storage**
  - Store detected violations with details: license plate, violation type, location, date/time, and image.
  - Store flagged car details similarly.
3. **Admin Interface**
  - **Violation Review**
    - Allow admin to view and review reported violations.
    - Provide filtering options based on violation type, date, location, etc.
  - **Flagged Car Review**
    - Allow admin to view and review flagged cars.
    - Provide filtering options based on car status, date, location, etc.
  - **Notification System**
    - Send SMS notifications to traffic police using Twilio for violations.
    - Generate and send detailed reports via SMS to traffic police including license plate, location, reason, and photo.
4. **Organization Reporting**
  - Generate violation reports.
  - Send reports to relevant organizations including license plate, location, reason, and photo.
5. **User Interface**
  - Develop a responsive front-end using React.
  - Provide intuitive navigation for viewing and managing violations and flagged cars.
  - Provide real-time updates and notifications.

## Non-Functional Requirements

### 1. Performance

- Ensure real-time processing of violation detection and reporting.
- Optimize database queries to handle large volumes of data.

### 2. Security

- Ensure secure data transmission and storage by applying jwt and token and major it more secure.
- Implement authentication and authorization mechanisms to restrict access to sensitive data.

### 3. Usability

- Design a user-friendly interface.
- Provide clear instructions and feedback for user actions.

### 4. Compliance

- Ensure the system complies with local traffic laws and regulations.

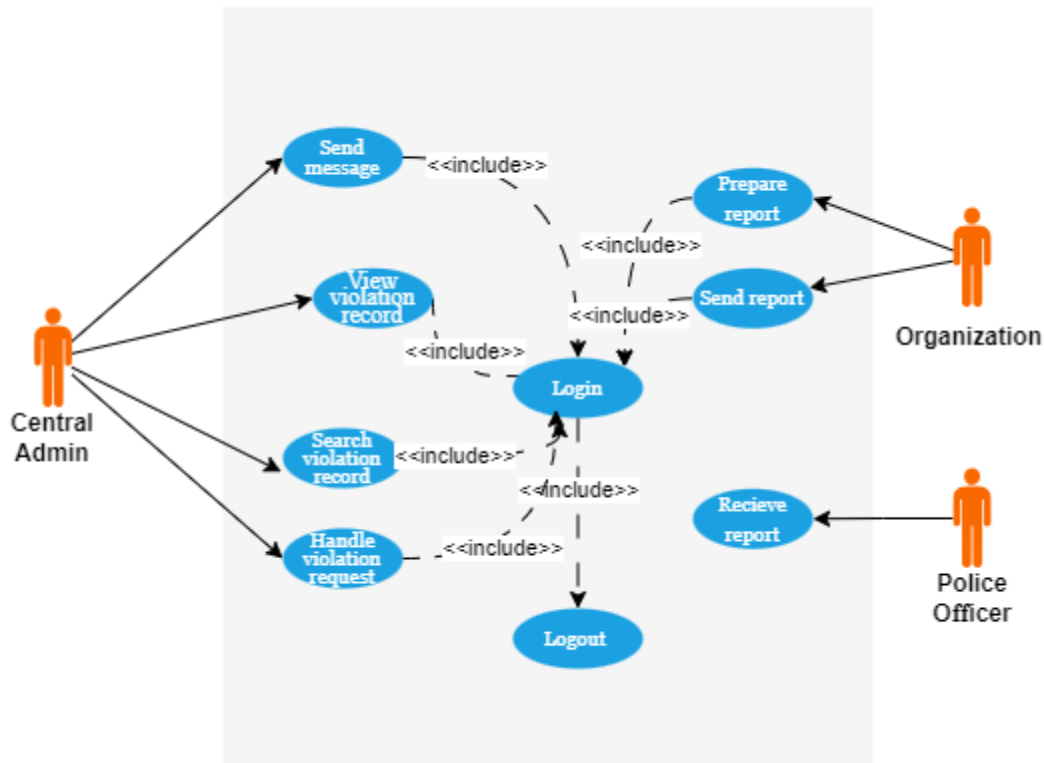


Figure 19 Use case diagram

## User Guide

### Admin

#### 1. Login:

- Access the admin interface and log in using your credentials.

2. **View Violations:**
  - Navigate to the violations dashboard to view all reported violations.
3. **Send Notifications:**
  - Select a violation and click "Send Notification" to send an SMS to the traffic police.
4. **Manage violation :**
  - remove violation and manage them .

## 5.Handle requests :

Receive reports of flagged cars from organizations.

### Organization

1. **Login:**
  - Access the organization interface and log in using your credentials.
2. **Report Violations:**
  - Review detected violations and generate reports.
  - Send reports to the admin for further action.

### Police officer

1. **Receive report :**

Get sms from the central admin when the violation or the flagged car is detected by the system.

## System Workflow

1. **Detection:**
  - The AI Detection Unit captures a traffic violation and records the license plate, violation type, location, and image.
2. **Reporting:**
  - The detected flagged is registered by the organization's interface.
  - The organization reviews the violation and generates a detailed report.
3. **Admin Review:**
  - The admin receives the flagged report.
  - The admin retrieve the violation from the central server.
  - The admin verifies the details and sends an SMS notification to the traffic police via Twilio.
4. **Notification:**
  - The traffic police receive an SMS with details of the violation and flagged vehicles for further action.

### Entities and Relationships

## 1. Admin

- **Attributes:**
  - admin\_id (Primary Key)
  - name
  - email
  - password
  - phone\_number
- **Relationships:**
  - Manages Violation
  - Manages FlaggedCar

## 2. Violation

- **Attributes:**
  - violation\_id (Primary Key)
  - license\_plate
  - violation\_type (e.g., red light running, line violation)
  - location
  - date\_time
  - image\_url
  - status (e.g., pending, processed)
  - processed\_by (Foreign Key to Admin)
- **Relationships:**
  - Handled by Admin (processed\_by)

## 3. FlaggedCar

- **Attributes:**
  - flagged\_car\_id (Primary Key)
  - license\_plate
  - status (e.g., flagged, cleared)
  - location
  - date\_time
  - image\_url
  - reason
  - processed\_by (Foreign Key to Admin)
- **Relationships:**
  - Handled by Admin (processed\_by)

## 4. Organization

- **Attributes:**
  - organization\_id (Primary Key)
  - name
  - email
  - phone\_number
- **Relationships:**
  - Reports FlaggedCar to Admin

## 5. Report



- **Attributes:**
  - report\_id (Primary Key)
  - organization\_id (Foreign Key to Organization)
  - flagged\_car\_id (Foreign Key to FlaggedCar)
  - details
- **Relationships:**
  - Sent by Organization
  - Relates to FlaggedCar

## 6. Message/notification Table:

### Attributes:

- id: Unique identifier for each notification.
- violation\_id: Identifier of the related violation.
- recipient: Phone number of the traffic police.
- message: Content of the SMS notification.

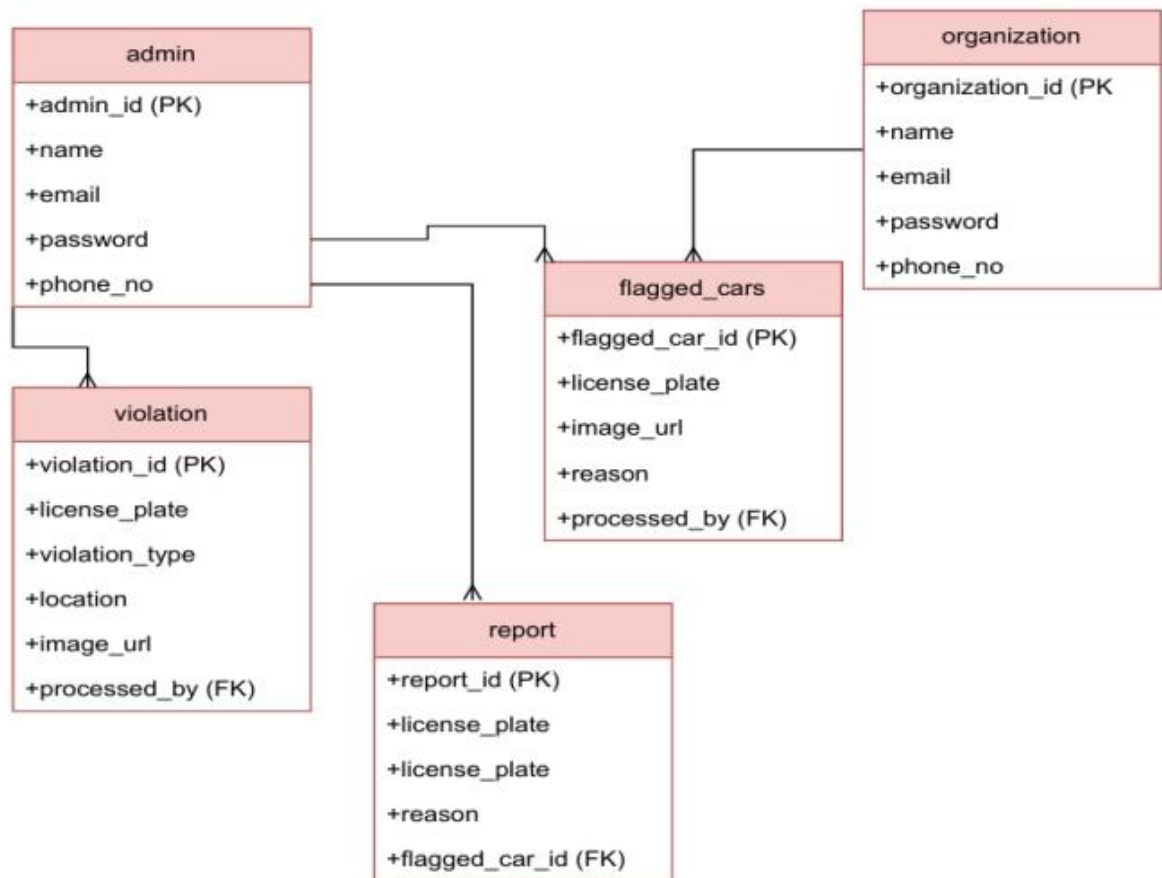


Figure 20 ENTITY RELATION DIAGRAM

## Chapter 5

### Results and Discussions

#### Model Evaluation and Testing

The evaluation of the trained YOLOv8 model was conducted to assess its performance.

##### *Evaluation Metrics*

To evaluate the performance of the trained YOLOv8 model, several key metrics were used

- **Precision (P):** The ratio of correctly predicted positive observations to the total predicted positives. High precision indicates fewer false positives.
- **Recall (R):** The ratio of correctly predicted positive observations to all observations in the actual class. High recall indicates fewer false negatives.
- **mean Average Precision (mAP50 and mAP50-95):** The average of the maximum precision at different recall values. mAP50 refers to the average precision at IoU threshold of 50%, and mAP50-95 is the average precision across IoU thresholds from 50% to 95%.

##### *Testing on a Validation Data Set*

The model was tested on a validation data set to measure its performance objectively. The validation set contained 110 images that were not used during training to ensure an unbiased evaluation. The trained YOLOv8 model was used to make predictions on the validation set. The predictions included bounding boxes, confidence scores, and class probabilities for each object detected in the images.

The results of the evaluation metrics are as follows

- Precision: 1.0
- Recall: 1.0
- mAP50: 0.995
- mAP95: 0.796

## Graphical Analysis

To provide a visual representation of the model's performance, several key graphs.

### A. F1-Confidence Curve:

The F1-Confidence Curve shows the relationship between the confidence threshold and the F1 score. The model achieves an F1 score of approximately 0.98 at a confidence threshold of 0.532, indicating a high balance between precision and recall at this threshold.

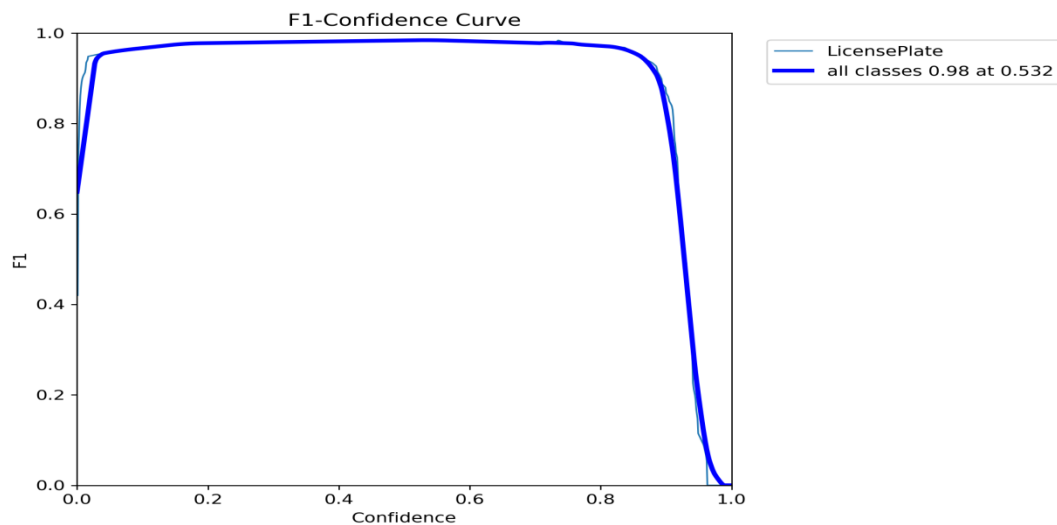


Figure 21 F1-confidence curve

### B. Precision-Confidence Curve:

The Precision-Confidence Curve illustrates how the precision varies with different confidence thresholds. The model achieves a precision of 1.0 at a confidence threshold of 0.761, indicating that at this threshold, all positive predictions made by the model are correct.

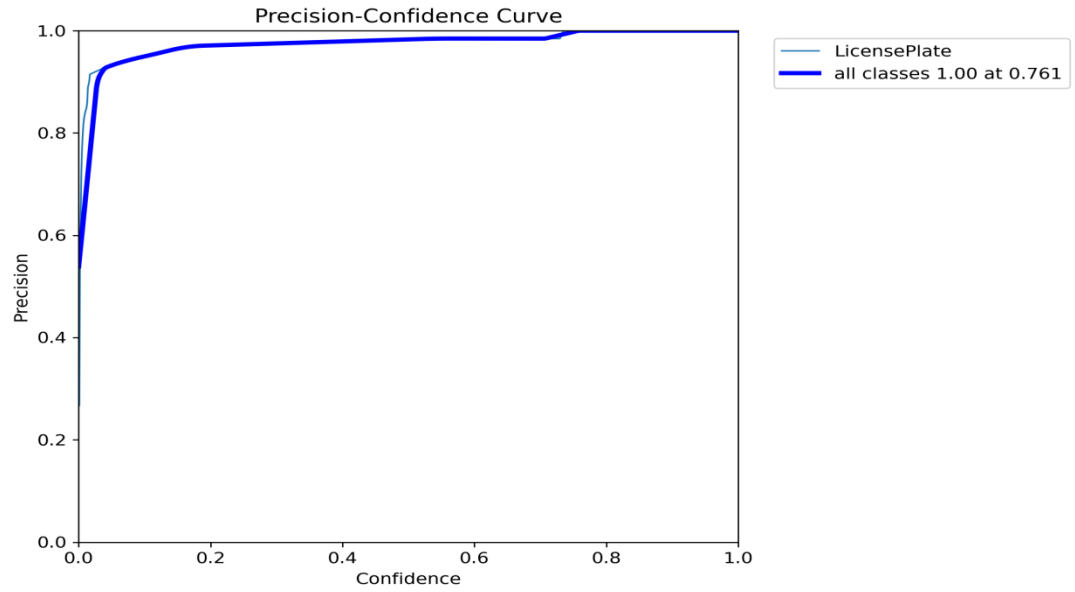


Figure 22 Precision-Confidence

### C. Precision-Recall Curve

The Precision-Recall Curve demonstrates the trade off between precision and recall for different thresholds. The model maintains a high precision of 0.986 while achieving a recall of 0.986, suggesting that the model is both accurate and comprehensive in detecting the positive instances.

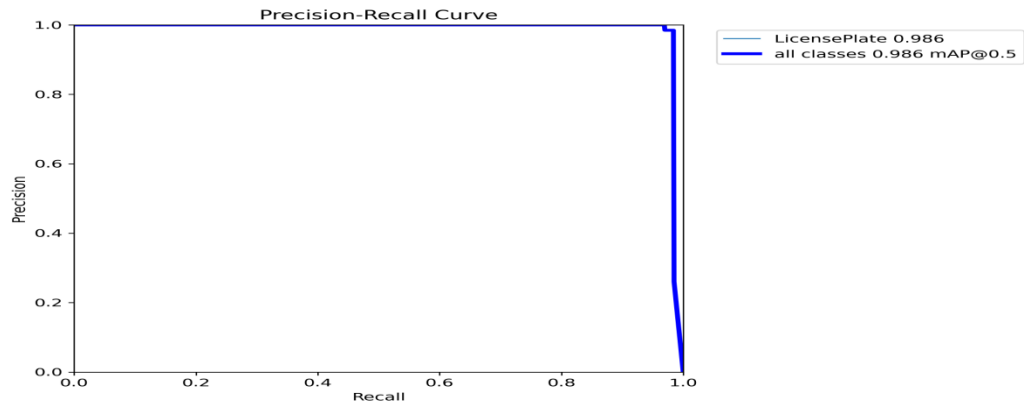


Figure 23 Precision-Recall Curve

#### D. Recall-Confidence Curve

The Recall-Confidence Curve shows the relationship between the confidence threshold and the recall. The model achieves a recall of 0.98 at a confidence threshold of 0.0, indicating that the model is able to detect most of the positive instances at very low confidence thresholds.

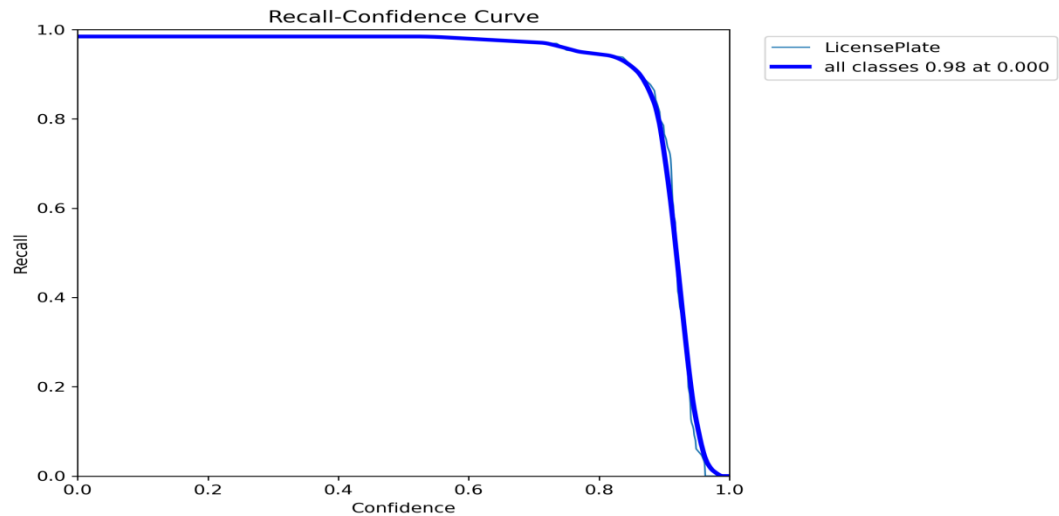


Figure 24 Recall-confidence curve

#### *Interpretation of Results*

The evaluation results and the graphical analysis indicate that the custom trained YOLOv8 model performs exceptionally well on the validation data set. The high precision and recall values, along with the high mAP scores, suggest that the model is both accurate and reliable in detecting license plates. The graphical analysis further supports this conclusion by showing that the model maintains a high F1 score and precision across a range of confidence thresholds, with minimal trade-offs in recall.

These results demonstrate the effectiveness of the custom trained YOLOv8 model weight(best.pt) in the context of license plate detection, making it a suitable choice for real world applications in vehicle monitoring and traffic violation detection.

## Vehicle Tracking Traffic Violation Detection

### Vehicle Tracking

Vehicles are tracked from the moment they cross the green line until they cross the blue line(in the case of the image below for the road in the left side)

Each car entering the region will acquire a unique id and list down{} or up{} is updated by appending this new id as seen in the image below.

The time taken for the vehicle to travel between the green line and the blue line is calculated as the  $ET = CT - VT$  where **CT** is the Current time and **VT** is the time record starting from the entry time or its simply Entry time + Elapsed time



Figure 25 Vehicle tracking

### Over Speeding Vehicles Detection

After Feeding the video footage using an **OpenCV** python line will be drawn on the video frame using the coordinates in the **config.json** file

The speed of the vehicle is calculated using the formula **Speed = Distance / Elapsed time** ,where distance is the ground **Distance** between the green and blue line and the **Elapsed time** is the time required by a particular car to cross the blue line as initiated from the green light entry point as seen in the left side road of the above image.

If any vehicle cross the second line as seen from either of the side of the road with exceeded speed limit it will be marked and the speed will be highlighted in red as seen in the picture below the vehicle exceeded the 30Km/h speed limit then ANPR will be called

automatically, the frame will be captured and saved to file as id\_yyy/mm/dd\_s in the Violation folder (id will be replaced with license plate in future and s implies it is a )



Figure 26 Over Speeding Car

### Red Light Violation Detection

In the video frame as seen in the image below the **Traffic light** line is drawn in the video frame using the coordinate on the **config.json** file and the color is controlled by the status variable in the **lightstatus.json** file the status variable in this file is updated via serial communication with Arduino when the color of the Traffic Light LED's connected to Arduino (red, yellow, green) are changing their status it is updated to python and the light\_status variable in the lightstatus.json file is updated for the purpose of detecting red light violation.

Using the light status on the light-status.config file the **Traffic Light** line as seen in the picture below will be changed from red to green or vice versa





Figure 27 Traffic light status determining the Traffic light line color



Figure 28 Traffic light status determining the Traffic light line color

Any Vehicles crossing the red line when the traffic light is **red** are flagged as violators. These vehicles are captured and saved with the Folder Violations in the format: **id\_dd/mm/yyyy\_r** (id will be replaced by license plate in the future and **r** implies it is a red light violation)





Figure 29 Red Light Violation detection

## Lane Violation Detection

Similarly the yellow lane line are drawn in the video frame using the coordinate in the config.json file as seen in the figure below for lane violation detection.

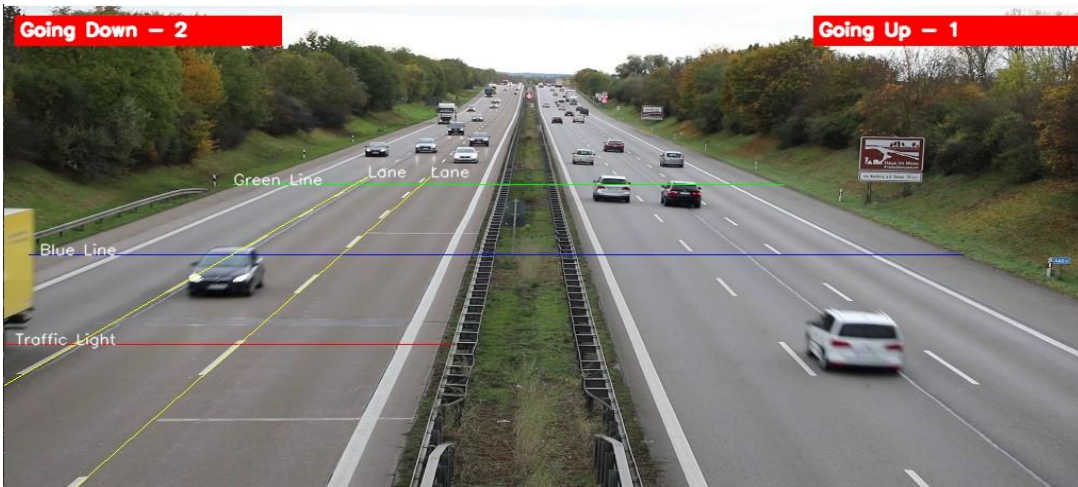


Figure 30 Lane Lines

Any vehicle that cross lane markings are flagged (if it were a solid line) as violators. These vehicles are captured and saved with the filename format: **id\_dd/mm/yyyy\_l** (id will be replaced by license plate in the future and l implies it is a lane violation)



Figure 31 Lane violation detection

## UI Result

[Dashboard](#)[Login](#)[Register](#)

### Login Here

Figure 32 Login page




violation List				
Search here by licence..				Search
violation type	Description	license plate	Image	Actions
Red light Violation	The Vehicle Violates Red Light	NRJ218 AA		<div>DeleteUpdateSMS</div>
Over Speed	The Vehicle Excidet the Speed limit	NBR678 AA		<div>DeleteUpdateSMS</div>
Lane Violation	The Vehicle Crossed the Solid Lane	BVE120 AA		<div>DeleteUpdateSMS</div>

Figure 33 Violation List

Update violation


Red light Violation

The Vehicle Violates Red Light

NRJ218

Choose File

No file chosen



Update

Figure 34 Update Violation record

### Send SMS

Red light Violation

The Vehicle Violates Red Light

NRJ218

Enter your message

+251 911234567

Send

Figure 35 Send SMS

#### Organization Interface:

- Login/Registration: Secure authentication for organization users.
- Dashboard: have a pannel for registering flagged vehicles .
- Report Generation: Tools for creating detailed flagged reports.
- System Configuration: Interface for managing system settings and parameters.

Dashboard

[Login](#)[Register](#)

Login Here

Sign In

Figure 36 Organizational Login page

Dashboard

Solomon ▾

Flag Licence Plate

Report

Figure 37 Organizational flag request issuer

## Chapter 6

### Conclusion and Recommendations for future work

#### Conclusion

In this study, we designed and implemented an AI based Traffic Violation Detection and Alerting System aimed at enhancing law enforcement capabilities and ensuring safer roads. The system integrates vehicle detection, license plate recognition, and alerting mechanisms using a camera module, a processing unit with AI model deployment, and a centralized data and request coordinating unit. Our approach addresses the shortcomings of traditional traffic management methods, which are heavily reliant on manual enforcement and are prone to corruption and inefficiency. By leveraging cutting-edge technology, our system contributes significantly to the reduction of traffic violations and improves overall traffic safety. The system also provides valuable insights into traffic patterns and violation trends, aiding law enforcement in making informed decisions.

#### 6.2 Recommendations for Future work

While the system has shown promising results, there are several areas for future improvement and expansion:

- **Emergency Vehicle Detection:** Incorporate specific algorithms to detect and prioritize emergency vehicles such as ambulances, police cars, and fire trucks. This enhancement will ensure that these vehicles are not delayed by traffic enforcement measures and can respond to emergencies promptly.
- **Localized License Plate Considerations:** Adapt the system to recognize and process Ethiopian license plates more accurately, considering the unique characteristics and formats of local license plates. This will improve the system's accuracy and reliability in a specific geographical context.
- **Integration with Broader Traffic Violations:** Extend the system's capabilities to integrate with broader traffic management systems for a more comprehensive approach to traffic monitoring and control. This should include the ability to detect and manage a wider range of traffic violations, such as unauthorized lane changes,

illegal U-turns, pedestrian crossing violations, seat belt usage infractions, speeding, driving under the influence (DUI), running stop signs, improper turns, and mobile phone usage while driving. Integrating with existing traffic management infrastructure and data sources will enable a more holistic view of traffic conditions and enhance the effectiveness of enforcement strategies.

By addressing these recommendations, the Traffic Violation Detection and Alerting System can become more robust, efficient, and effective in enhancing road safety and supporting law enforcement efforts.

## References

- [1] G. L. MOSISA, “ANALYSIS OF ROAD TRAFFIC VIOLATIONS IN ADDIS ABABA CITY,” Addis Ababa University, Addis Ababa, 2018.
- [2] E. E. Dube, “Urbanization in Ethiopia: Challenges, opportunities and Future Research Directions,” Dilla, 2019.
- [3] T. Bireda, “Intelligent Transport System in Ethiopia: Status and the Way Forward,” in *Springer International Publishing*, Cham, 2018.
- [4] M. G. P. R. O. R. A. H. S. ., V. G. P. D. S. B. Hirnaik Ashlesha Padmakar, “Traffic Rules Violation Detection System,” *International Journal of Innovative Science and Research Technology*, vol. 8, no. 6, p. 6, 2023.
- [5] M. K. S. B. P. Jin Su Kim, “A study on implementation of real-time intelligent video surveillance system based on embedded module,” *EURASIP Journal on Image and Video Processing* , p. 22, 2021.
- [6] L.-M. M. B. Z. ., K.-L. D. Xiaoling Wang, “A Video-based Traffic Violation Detection System,” Hangzhou, China, 2013.
- [7] D. N. K. D. D. U. S. R. K. ., S. N. T. Dr. S. Raj Anand, “TRAFFIC SIGNAL VIOLATION DETECTION USING ARTIFICIAL INTELLIGENCE AND DEEP LEARNING,” *International Journal of Advanced Research in Engineering and Technology (IJARET)*, vol. 12, no. 2, p. 11, 2021.
- [8] Y. q. Q. Tao Zhang, “Research and Application of License Plate Recognition System”.
- [9] S. P. R. M. Saman Rajebi, “A License Plate Recognition System with Robustness against Adverse Environmental Conditions Using Hopfield’s Neural Network,” *Axioms*, p. 12, 2023.



## Appendices

### Appendix A:

#### JSON Web Token

A JSON web token(JWT) is JSON Object which is used to securely transfer information over the web(between two parties). It can be used for an authentication system and can also be used for information exchange. The token is mainly composed of header, payload, signature. These three parts are separated by dots(.). JWT defines the structure of information we are sending from one party to the another, and it comes in two forms – Serialized, Deserialized. The Serialized approach is mainly used to transfer the data through the network with each request and response. While the deserialized approach is used to read and write data to the web token.

#### Header

A header in a JWT is mostly used to describe the cryptographic operations applied to the JWT like signing/decryption technique used on it. It can also contain the data about the media/content type of the information we are sending. This information is present as a JSON object then this JSON object is encoded to BASE64URL. The cryptographic operations in the header define whether the JWT is signed/unsigned or encrypted and are so then what algorithm techniques to use. A simple header of a JWT looks like the code below:

```
{  
  
  "typ":"JWT",  
  
  "alg":"HS256"
```

```
}
```

The 'alg' and 'typ' are object key's having different values and different functions like the 'typ' gives us the type of the header this information packet is, whereas the 'alg' tells us about the encryption algorithm used. HS256 and RS256 are the two main algorithms we make use of in the header section of a JWT.

### **Payload**

The payload is the part of the JWT where all the user data is actually added. This data is also referred to as the 'claims' of the JWT. This information is readable by anyone so it is always advised to not put any confidential information in here. This part generally contains user information. This information is present as a JSON object then this JSON object is encoded to BASE64URL. We can put as many claims as we want inside a payload, though unlike header, no claims are mandatory in a payload. The JWT with the payload will look something like this:

```
{  
  
  "userId": "b07f85be-45da",  
  
  "iss": "Meri_server",  
  
  "sub": "auth/some-hash-here",  
  
  "exp": 153452683  
  
}
```

### **Signature**

This is the third part of JWT and used to verify the authenticity of token. BASE64URL encoded header and payload are joined together with dot(.) and it is then hashed using the hashing algorithm defined in a header with a secret key. This signature is then appended to header and payload using dot(.) which forms our actual token ,

HEADER	PAYLOAD	SIGNATURE
--------	---------	-----------