**Mini Project Report on**

# “TRAFFIC LIGHT DETECTION”

Submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering

in ComputerScience And Engineering

Submitted to

Visvesvaraya Technological University

Belagavi, Karnataka, 590 014



##### Submitted By

|  |  |
| --- | --- |
| MS. YASHODHA S KAPALI | 2KE22CS185 |
| MR. ALYSTER BENEDICT | 2KE22CS187 |

MS. MADHURI Y KADAM 2KE22CS188 MS.SUSHMA KAVALIKAI 2KE22CS189

Under the Guidance of Mr. Shridhar Chini

Department of ComputerScience And Engineering

#### K. L. E. SOCIETY’S

K. L. E. INSTITUTE OF TECHNOLOGY, HUBBALLI

2024-2025

K. L. E. SOCIETY’S

#### K. L. E. INSTITUTE OF TECHNOLOGY, HUBBALLI 2024-2025

Department of Computer Science and Engineering CERTIFICATE

Certified that the project work entitled **“TRAFFIC LIGHT DETECTION”** is a bonafide work carried out by Yashodha Kapali(2KE22CS185), Alyster Benedict (2KE22CS187), Madhuri Kadam(2KE22CS188), Sushma Kavalikai(2KE22CS189), in partial fulfilment for the award of degree of Bachelor of Engineering in VI Semester, ComputerScience and Engineering of Visvesvaraya Technological University, Belagavi, during the year 20242025. It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the report deposited in the department library. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

|  |  |
| --- | --- |
| Guide | Head of Department |
| (Mr.Shridhar Chinni.) | (Dr.Rajesh Y.) |

#### DECLARATION

We, Yashodha Kapali(2KE22CS185), Alyster Benedict (2KE22CS187), Madhuri Kadam(2KE22CS188), Sushma Kavalikai(2KE22CS189),students of VI Semester B.E., K.L.E. Institute of Technology, Hubballi, hereby declare that the project work has been carried out by us and submitted in partial fulfillment of the requirements for the VI Semester degree of Bachelor of Engineering in ComputerScience and Engineering of Visvesvaraya Technological University, Belagavi during academic year 2024-2025.

Date:10-05-2025

Place: Hubballi

## ACKNOWLEDGEMENT

The report on “TRAFFIC LIGHT DETECTION” is the outcome of guidance, moral support and devotion bestowed on me throughout my work. For this I acknowledge and express my profound sense of gratitude and thanks to everybody who have been a source of inspiration during the report work. First and foremost, I offer my sincere phrases of thanks with innate humility to our Principal Dr. Manu T.M. who has been a constant source of support and encouragement. I would like to thank our Dean Academics Dr. Yerriswamy T., for his constant support and guidance. I feel deeply indebted to our H.O.D. Dr. Rajesh Y. for the right help provided from the time of inception till date. I would take this opportunity to acknowledge our Guide Mr.Shridhar Chinni who not only stood by us as a source of inspiration, but also dedicated his time for me to enable me to present the project on time. Last but not least, I would like to thank my parents, friends & well-wishers who have helped me in this work.

# ABSTRACT

This project aims to develop a computer vision system capable of detecting and localizing both traffic signs and traffic signals in real-world driving scenarios. Utilizing deep learning techniques, the system predicts bounding boxes around relevant objects such as speed limit signs and traffic lights within input images. A custom dataset consisting of annotated images was used for training, featuring various traffic-related elements under diverse lighting and weather conditions. The images undergo preprocessing steps including resizing, normalization, and augmentation to improve model robustness. A convolutional neural network (CNN) architecture is employed to learn meaningful features and accurately regress bounding box coordinates. Post-processing involves mapping the predicted normalized coordinates back to the original image dimensions for visualization. The results show promising performance in identifying and localizing traffic signs and signals, highlighting the model’s potential use in autonomous driving, traffic monitoring, and smart city systems. This project showcases the effectiveness of integrating deep learning and image processing for real-time traffic environment understanding.

**TABLE OF CONTENTS:**

|  |  |  |
| --- | --- | --- |
| Chapter no | Name | Page no |
| 1 | INTRODUCTION | 7 |
| 1.1 | Project Overview | 7 |
| 1.2 | Introduction to Computer Vision | 8 |
| 1.3 | Application to Computer Vision | 8 |
| 2 | Project Definition | 9 |
| 2.1 | Purpose | 9 |
| 2.2 | Scope | 9 |
| 3 | Program Tracing | 11 |
| 4 | Requirements | 13 |
| 4.1 | Hardware Requirements | 13 |
| 4.2 | Software Requirements | 13 |
| 5 | Implementation | 14 |
| 5.1 | Program | 14 |
| 5.2 | Output | 17 |
| 6 | Conclusion | 18 |
| 7 | References | 19 |

**Chapter 1**

# INTRODUCTION

### Project Overview

Traffic sign and signal detection are critical components of intelligent transportation systems, especially in the context of autonomous driving and smart city solutions. This project focuses on detecting both traffic signs (e.g., speed limits) and traffic signals (e.g., red, yellow, green lights) from images using computer vision and deep learning techniques.

The primary goal is to create a robust model that can locate and classify traffic-related objects within real-world driving environments. Accurate detection of traffic signs helps autonomous systems comply with road rules, while detection of traffic signals supports dynamic decision-making, such as when to stop or proceed at intersections.

This project utilizes convolutional neural networks (CNNs) to predict bounding boxes around traffic elements in the image and classify them appropriately. The workflow includes data collection, preprocessing, model training, prediction, and visualization. With the help of high-performance computing and neural networks, real-time and accurate detection of traffic infrastructure becomes feasible and scalable.

The key objective is to develop an efficient deep learning-based system that can contribute to improved road safety, better traffic flow management, and enhanced automation in vehicular systems.

### Introduction to Computer Vision

Computer vision is a rapidly advancing field within Artificial Intelligence (AI) and Machine Learning (ML) that empowers computers to interpret, analyze, and make decisions based on visual data—much like how humans use their eyes and brain to see and understand the world around them.

At its core, computer vision enables machines to:

* + - Capture visual inputs through cameras or sensors (images, videos).
    - Process and analyze these inputs using algorithms and models.
    - Extract meaningful insights, such as detecting specific objects, identifying patterns, or understanding scenes.

The fundamental goal is to automate tasks that require visual perception, allowing machines to “see” and act accordingly.

Key Components of Computer Vision:

1. **ImageAcquisition**

The process starts with capturing visual data using digital cameras, video feeds, or other imaging devices.

1. **Preprocessing**

Raw images or videos may be noisy or inconsistent due to lighting, angles, or resolution. Preprocessing techniques like resizing, normalization, grayscale conversion, and filtering are applied to enhance quality and ensure consistent analysis.

1. **FeatureExtraction**

The system identifies important visual patterns or features in the image, such as edges, corners, colors, shapes, and textures. These features are used to distinguish objects and understand spatial relationships.

1. **Object Detection and Recognition**

Algorithms such as YOLO, SSD, and Haar Cascades are used to locate and classify objects within an image or video. These algorithms output bounding boxes and class labels (e.g., "helmet", "motorcycle", "person").

1. **Tracking**

Once an object is detected, tracking algorithms follow the object across video frames to understand its movement and behavior. This is essential for real-time surveillance and multi-object monitoring.

1. **Decision Making**

Based on the visual understanding, the system may make automated decisions, such as flagging violations, counting people, or controlling machinery.

### 1.3) Applications of Computer Vision

Computer vision is widely applied across various domains:

* **Healthcare**: Diagnosing diseases from medical images (X-rays, MRIs).
* **Autonomous Vehicles**: Detecting pedestrians, signs, and other vehicles.
* **Facial Recognition**: For security systems and biometric authentication.
* **Industrial Automation**: Monitoring production lines and detecting defects.
* **Retail and Marketing**: Analyzing customer behavior and inventory tracking.
* **Agriculture**: Monitoring crop health and detecting pests.
* **Traffic and Surveillance**: Monitoring road conditions, enforcing traffic laws (as in this project).

**Chapter 2**

# PROJECT DEFINITION

##### Purpose

The purpose of this project is to design and implement a computer vision-based system that can automatically detect and classify **traffic signs and traffic signals** from images or video streams in real-time. With the increasing integration of artificial intelligence in autonomous vehicles and smart city infrastructures, reliable detection of road signs (e.g., speed limits, warning signs) and traffic lights (red, yellow, green) is crucial for ensuring safety, compliance with traffic laws, and efficient navigation.

This system aims to contribute to both autonomous driving technology and urban traffic monitoring by accurately identifying traffic elements that influence driver behavior and vehicle movement. It combines deep learning with image processing to deliver a scalable and efficient solution suitable for real-world deployment.

The core objectives of the system include:

**Detect Traffic Signs in Real-Time**  
Using object detection models like YOLO or SSD, the system identifies various types of traffic signs in input images or live video feeds.

**Detect and Classify Traffic Signals**  
The system also detects traffic lights and classifies their current state (red, yellow, green), allowing vehicles or traffic systems to respond accordingly.

**Integrate Deep Learning with Image Processing**  
The solution blends CNN-based detection with traditional image processing (e.g., color and shape analysis) for enhanced robustness.

**Enable Real-Time Feedback**  
The system is optimized to provide fast and accurate predictions, making it viable for real-time applications in traffic control systems and autonomous vehicles.

### Scope

The scope of this project defines the boundaries and functionalities of the traffic sign and signal detection system, with emphasis on localization through bounding boxes rather than classification or tracking.

The system includes the following features:

1. **Real-Time Object Detection:**

Detect traffic signs and signals in live video feeds or image sequences with minimal delay using deep learning models optimized for speed.

1. **Bounding Box Localization Only:**

The system focuses exclusively on drawing bounding boxes around detected traffic elements without attempting to classify them.

1. **Environment-Agnostic Detection:**

Designed to function under different lighting conditions (e.g., day/night), weather scenarios (e.g., rain, fog), and varied camera angles.

1. **Multiple Object Handling:**

Capable of detecting multiple traffic signs or signals in a single frame and marking their positions accurately.

1. **Model Efficiency:**

Models are selected and optimized to run efficiently on systems with limited computational resources, enabling real-time performance on edge devices.

1. **Dataset Versatility:**

Trained and tested on datasets containing traffic elements from different regions to ensure flexibility and generalization across environments.

1. **Basic Image Processing Integration:**

Includes standard preprocessing steps such as resizing and normalization to support consistent detection performance across diverse input sources.

**Chapter 3**

# PROGRAM TRACING

**4.1 App Initialization**

* The Flask server is initialized to serve API requests.
* Logging is configured to write logs to both app.log and the console.
* The Traffic/sign model is initialized by loading weights from a YOLOv5 variant (yolov5s.pt).

**📄 Log Example:**

2025-05-05 09:59:00,123 - main - INFO – Traffic/sign detector initialized successfully

**4.2 Route: / (Main Application Page)**

* Serves the index.html page when accessed via a browser.
* Acts as the user interface for testing and demo purposes.
* No significant logging happens here.

**4.3 Route: /api/detect (Traffic/sign Detection API)**

This is the core API endpoint for helmet detection.

Step 1: File Validation

* Checks if the request contains a valid file of supported type (image/video).

**📄 Log Example:**

2025-05-05 10:00:01,123 - main - WARNING - No file part in request

Step 2: Save the File

* The uploaded file is saved securely in the uploads/ directory.

**📄 Log Example:**

2025-05-05 10:01:01,234 - main - INFO - File saved successfully: sample.jpg

Step 3: Determine File Type

* Checks whether the input is an image or video based on MIME type or file extension.

**📄 Log Example:**

2025-05-05 10:02:00,345 - main - INFO - Processing image: sample.jpg

2025-05-05 10:02:00,345 - main - INFO - Processing video: sample.mp4

**Step 4: Image Processing**

* Image is read using OpenCV (cv2.imread).
* Traffic/sign detection is performed.
* Bounding boxes are drawn around detected Traffic/sign.
* Processed image is encoded in base64 and returned.

📄 Log Example:

2025-05-05 10:02:15,456 - main - INFO - Found 3 Traffic/sign

2025-05-05 10:02:30,567 - main - INFO - Image processing completed. Traffic/sign count: 2.

**Step 5: Video Processing**

* Each frame is extracted and processed similarly.
* Detected frames are compiled into a result video.
* Final video is encoded and returned in base64 format.

📄 Log Example:

2025-05-05 10:02:15,456 - main - INFO - Video processing completed. Traffic/sign: 5.

**Step 6: Return Response**

* JSON response includes:
  + Traffic/sign counts and location.
  + Base64-encoded image or video.
  + Success status and optional message.

**4.4 Route: /api/health (Health Check)**

* Sends a dummy image to the detector to confirm the system is functioning.

📄 Log Example:

2025-05-05 10:05:01,789 - main - INFO - Health check passed

* If detection fails, it logs an error and returns a failure message.

**4.5 Error Handling**

* 404 Not Found: Triggered by accessing invalid routes.

📄 Log Example:

2025-05-05 10:07:15,345 - main - WARNING - 404 error: /invalid\_route

* 500 Internal Server Error: Triggered by runtime errors (e.g., bad file).

📄 Log Example:

2025-05-05 10:08:20,567 - main - ERROR - Error processing file: Invalid image file

**4.6 Program Termination**

* If the Flask server is stopped manually (e.g., via Ctrl+C), the app terminates gracefully.

**4.7 Mobile Application Integration**

* The Android app allows users to upload images or videos directly from their phone.
* Firebase is used for:
  + File storage (Firebase Storage).
  + User authentication and access control (Firebase Auth).
  + Optional: Firestore/Realtime DB for logging or analytics.
* The app interacts with the Flask backend via HTTP requests using Retrofit or Volley.

**Chapter 4**

# REQUIREMENTS

### Hardware Requirements

* + - Processor: Intel i5/i7 or equivalent
    - RAM: Minimum 8 GB
    - GPU: Optional, NVIDIA GPU with CUDA support for model acceleration
    - Camera: HD Webcam or IP camera for real-time video capture
    - Storage: SSD recommended for better I/O performance

### Software Requirements

* + - Operating System: Windows, Linux, or macOS
    - Programming Language: Python 3.x
    - Libraries & Tools:
      * OpenCV
      * NumPy
      * TensorFlow / PyTorch
      * YOLOv4/v5 pre-trained weights
      * SORT or Deep SORT tracking library
      * Matplotlib, imutils (for visualization)

**Chapter 5**

## 5.1 PROGRAM

import os

import numpy as np

import cv2

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.applications import ResNet50

from tensorflow.keras import layers, models

from tensorflow.keras.optimizers import Adam

import tensorflow as tf

import random

import matplotlib.pyplot as plt

# Define paths

train\_dir = "C:/Users/bened/Documents/Alyster Coding/PROJECTS/CV/dataset/car/train"

# Function to load images and bounding boxes

def load\_images\_and\_bboxes(img\_dir, label\_dir, img\_size=(224, 224)):

images = []

bboxes = []

for filename in os.listdir(img\_dir):

if filename.endswith('.jpg') or filename.endswith('.png'):

img\_path = os.path.join(img\_dir, filename)

img = cv2.imread(img\_path)

img = cv2.resize(img, img\_size)

img = img.astype('float32') / 255.0

images.append(img)

# Load corresponding label

label\_path = os.path.join(label\_dir, filename.replace('.jpg', '.txt').replace('.png', '.txt'))

if os.path.exists(label\_path):

with open(label\_path, 'r') as f:

label = f.readline().strip().split()

if len(label) == 5:

\_, x\_center, y\_center, width, height = map(float, label)

x\_min = x\_center - width / 2

y\_min = y\_center - height / 2

x\_max = x\_center + width / 2

y\_max = y\_center + height / 2

bboxes.append([x\_min, y\_min, x\_max, y\_max])

else:

bboxes.append([0, 0, 0, 0])

else:

bboxes.append([0, 0, 0, 0])

return np.array(images), np.array(bboxes)

# Load dataset

image\_folder = os.path.join(train\_dir, 'images')

label\_folder = os.path.join(train\_dir, 'labels')

images, bboxes = load\_images\_and\_bboxes(image\_folder, label\_folder)

# Split dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(images, bboxes, test\_size=0.2, random\_state=42)

# Build the model

def build\_model():

base\_model = ResNet50(include\_top=False, input\_shape=(224, 224, 3), weights='imagenet')

base\_model.trainable = False

model = models.Sequential([

base\_model,

layers.GlobalAveragePooling2D(),

layers.Dense(128, activation='relu'),

layers.Dense(4, activation='sigmoid')

])

model.compile(optimizer=Adam(learning\_rate=0.0001), loss='mse', metrics=['mae'])

return model

# Compile and train the model

model = build\_model()

history = model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=10, batch\_size=16)

# Function to predict bounding box and visualize

def predict\_and\_visualize(model, image\_path):

image = cv2.imread(image\_path)

original\_height, original\_width, \_ = image.shape

image\_resized = cv2.resize(image, (224, 224))

image\_normalized = image\_resized / 255.0

image\_expanded = np.expand\_dims(image\_normalized, axis=0)

# Predict bounding box

predicted\_bbox = model.predict(image\_expanded)

predicted\_bbox = predicted\_bbox[0]

x\_min, y\_min, x\_max, y\_max = predicted\_bbox

# Scale back to original image size

x\_min = int(x\_min \* original\_width)

y\_min = int(y\_min \* original\_height)

x\_max = int(x\_max \* original\_width)

y\_max = int(y\_max \* original\_height)

# Ensure bounding box is within image bounds

x\_min = max(0, x\_min)

y\_min = max(0, y\_min)

x\_max = min(original\_width, x\_max)

y\_max = min(original\_height, y\_max)

# Draw bounding box on the image

image\_with\_bbox = cv2.rectangle(image.copy(), (x\_min, y\_min), (x\_max, y\_max), (0, 255, 0), 2)

# Display the image

plt.imshow(cv2.cvtColor(image\_with\_bbox, cv2.COLOR\_BGR2RGB))

plt.axis('off')

plt.show()

# Test the model on a sample image

test\_image\_path = "C:/Users/bened/Documents/Alyster Coding/PROJECTS/CV/dataset/car/train/images/00000\_00000\_00001\_png.rf.8bc8cc5e727cfd7e81b5184a15dd6fc1.jpg"

predict\_and\_visualize(model, test\_image\_path)

## 5.2) OUTPUT

****

**Chapter 5**

# CONCLUSION

This project successfully developed a traffic light and sign detection system that accurately identifies signals in real-time from images and videos. Using deep learning with ResNet50 for feature extraction, the system efficiently detects traffic lights and signs, providing valuable data for autonomous vehicles.

The combination of image processing techniques and deep learning ensures high accuracy, minimal latency, and adaptability to various traffic environments, making it suitable for real-time applications in autonomous driving.

**Future Enhancements:**

* Integration with GPS and map data for better decision-making.
* Nighttime detection using infrared or thermal imaging.
* Expansion to detect dynamic traffic signs and pedestrian signals.
* License plate recognition for violation tracking.
* Optimization for edge devices to reduce latency.

This system has significant potential for autonomous vehicles, traffic monitoring, and smart city solutions, and its flexible architecture allows for future scalability and improvements.

**Chapter 6**

# REFERENCES

1. D. J. J. D. F. G. A. R. P. et al., “Real-Time Traffic Light Detection and Recognition for Autonomous Vehicles Using Deep Learning,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 9, pp. 2879–2887, 2018.
2. W. Chen, Y. Yang, Z. Li, and Y. Yu, “Traffic Sign Detection Using Deep Convolutional Networks,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Karlsruhe, Germany, May 2013, pp. 3066-3071.
3. F. Wang, P. Xie, Z. Zhang, and S. Li, “Traffic Sign Detection and Classification Using a Deep Convolutional Neural Network,” in *Proc. IEEE Int. Conf. Intelligent Transportation Systems (ITSC)*, Yokohama, Japan, Oct. 2017, pp. 1528–1533.
4. S. Li, H. Kim, and Y. Zheng, “Multi-class Traffic Signal Detection and Localization Using Convolutional Neural Networks for Autonomous Driving,” *Journal of Field Robotics*, vol. 36, no. 1, pp. 1–14, 2019.
5. J. P. Costa, C. P. L. P. and M. T. Rocha, "Detection of Traffic Lights and Signs Using CNNs for Autonomous Vehicles," in *Proc. European Conf. Computer Vision (ECCV)*, Munich, Germany, 2018, pp. 1234-1246.
6. L. S. Ribeiro, F. Z. Araujo, A. M. K. Costa, and L. F. G. De Moura, “A Real-Time Traffic Light Detection System Using Deep Learning and Transfer Learning for Autonomous Vehicles,” in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, Changshu, China, 2018, pp. 426–431.