**HELMET DETECTION AND NUMBER PLATE RECOGNITION USING MACHINE LEARNING**

A Project Work submitted in partial fulfilment of the

requirements for the degree of

BACHELOR OF COMPUTER APPLICATIONS

to the

# PERIYAR UNIVERSITY, SALEM - 11

Submitted By

# S.YOGESH

# (Reg. No. C21UG131CAP047)

**DEPARTMENT OF COMPUTER APPLICATIONS**

# VYSYA COLLEGE

**(AFFILIATED TO PERIYAR UNIVERSITY)**

**SALEM – 636 103**

**MARCH – 2024**

**Date:**

## CERTIFICATE

This is to certify that the Project Work entitled **" HELMET DETECTION AND NUMBER PLATE RECOGNITION USING MACHINE LEARNING"** submitted in partial fulfillment of the requirements of the degree of Bachelor of Computer Applications to the Periyar University, Salem is a record of bonafide work carried out by  **YOGESH** Reg. No.**C21UG131CAP047** under my supervision and guidance.

**Head of the Department Internal Guide**

**Date of Viva-voice:**

**Internal Examiner External Examiner**

**ACKNOWLEDGEMENT**

I am highly indebted to **Thiru J. Rajendra Prasad,** Correspondent, Vysya College, Salem for their encouragement as well as for providing necessary information regarding the project & also for their support in completing the project.

My abundant thanks to **Dr.P.Venkatesan, M.Sc, Ph.D.** Principal, Vysya College, Salem who gave this opportunity to do project work.

I wish my heartfelt thanks to our Head of the Department of ComputerApplications, Vysya College, Salem for his valuable suggestion, innovative ideas, constructive criticisms and inspiring guidance had enabled me to complete the work successfully.

I Express my deep gratitude and sincere thanks to my guide and other faculty members of our department for helping me in kind manner throughout the successful completion of this work.

I would like to express my special gratitude and thanks to all above mentioned people for giving us such attention and time. My thanks and appreciations also go to our other faculties in developing the project and people who have willingly helped us out with their abilities.

## CONTENTS

|  |  |
| --- | --- |
| **CHAPTER** | **PAGE NO** |
| **COLLEGE BONAFIDE CERTIFICATE**  **ACKNOWLEDGEMENT**  **SYNOPSIS** |  |
| **1. INTRODUCTION** |  |
| * 1. SYSTEM SPECIFICATION |  |
| 1.1.1 HARDWARE CONFIGURATION |  |
| 1.1.2 SOFTWARE SPECIFICATION |  |
| **2. SYSTEM STUDY** |  |
| 2.1EXISTING SYSTEM |  |
| 2.1.1DESCRIPTION |  |
| 2.1.2 DRAWBACKS |  |
| 2.2 PROPOSED SYSTEM |  |
| 2.2.1 DESCRIPTION |  |
| 2.2.2 FEATURES |  |
| **3. SYSTEM DESIGN AND DEVELOPMENT** |  |
| 3.1 FILE DESIGN |  |
| 3.2 INPUT DESIGN |  |
| 3.3 OUTPUT DESIGN |  |
| 3.4 CODE DESIGN |  |
| 3.5 DATABASE DESIGN |  |
| 3.6 SYSTEM DEVELOPMENT |  |
| 3.6.1 DESCRIPTION OF MODULES |  |
| **4. TESTING AND IMPLEMENTATION** |  |
| **5. CONCLUSION** |  |
| **6. BIBLIOGRAPHY** |  |
| **APPENDICES** |  |
| 1. DATA FLOW DIAGRAM |  |
| 1. TABLE STRUCTURE |  |
| 1. SAMPLE CODING |  |
| 1. SAMPLE INPUT |  |
| 1. SAMPLE OUTPUT |  |

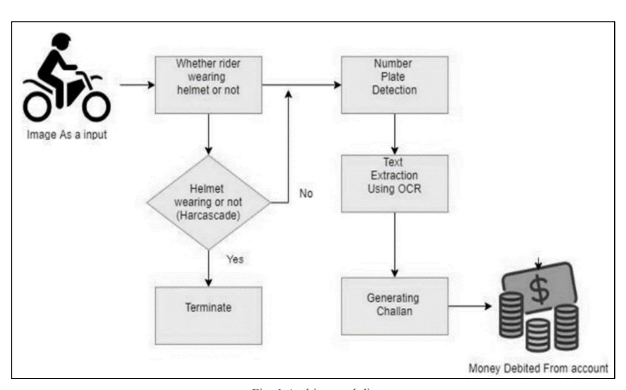
**SYNOPSIS**

Motorcycles have always been the primary mode of transportation in developing countries. Motorcycle accidents have increased in recent years. One of the main reasons for fatalities in accidents is that a motorcyclist does not wear a protective helmet. The most common way to ensure that motorcyclists wear a helmet is by traffic police to manually monitor motorcyclists at road junctions or through CCTV footage and to penalize those without a helmet. But it requires human intervention and effort. This system proposes an automated system for detecting motorcyclists who do not wear a helmet and a system for retrieving motorcycle number plates from CCTV video footage. First, the system classifies moving objects as motorcycling or non-motorcycling. In the case of a classified motorcyclist, the head portion is located and classified as a helmet or non-helmet. Finally, the motorcyclist without a helmet is identified. Further we have developed a system which identifies the number plates and extracts the characters of the number plate using OCR algorithm.

**CHAPTER - 1**

**INTRODUCTION**

Wearing helmets while riding motorcycles is mandatory by law to ensure the safety of the riders. Currently, the responsibility of monitoring helmet usage falls on the police force. However, due to the limitations of human senses and insufficient police force, this method has proven to be inefficient. CCTV surveillance-based methods are also used in major cities, but they require human assistance and are not automated. As the number of motorcycles on the road increases and concern for human safety grows, research in the domain of road transport has also increased. proposes a system that automates the monitoring of motorcyclists by detecting those not wearing helmets and retrieving their license plate numbers in real time from CCTV camera videos at road junctions using machine learning.

The problem of increasing road accidents in India has become a major concern, with a high number of deaths caused by head injuries due to a significant portion of the population not wearing helmets. To address this issue, an automated system is needed to detect individuals not wearing helmets and identify the license plates of the motorcycles to penalize the offenders. The use of machine learning in transportation systems has become increasingly popular in recent years. 

Automated systems that detect helmet usage and license plates on motorcycles are just one example of how technology can improve road safety. In addition to improving enforcement, such systems can also provide valuable data for policymakers to make informed decisions about road safety regulations. However, there are some challenges in implementing such systems. For instance, the accuracy of license plate recognition algorithms can be affected by factors such as lighting conditions, weather, and the distance between the camera and the vehicle. In addition, the implementation of such systems requires significant investment in terms of infrastructure.

Therefore, it is essential to carefully evaluate the feasibility of such systems and their potential impact on road safety. Despite these challenges, the use of machine learning and other advanced technologies has the potential to revolutionize road safety in developing countries. By improving enforcement and providing valuable data for policymakers, such systems can help reduce the number of deaths and injuries caused by motorcycle accidents. With continued research and development, these systems may become an integral part of transportation systems around the world, making roads safer for everyone. The trained YOLOv5 model is employed to detect helmets in video frames. The model performs inference, predicting bounding boxes around the detected helmets. Non-maximum suppression (NMS) is then utilized to refine the results and eliminate redundant detections, improving the overall accuracy of the helmet detection process Regions of interest (ROIs) corresponding to the detected number plates are extracted based on the helmet detection results.

Pre-processing techniques, such as noise reduction, contrast adjustment, and thresholding, are applied to enhance the image quality of the number plate regions. Optical character recognition (OCR) algorithms are then used to recognize and extract alphanumeric characters from the pre-processed number plate regions. The helmet detection and number plate recognition components are seamlessly integrated into a unified real-time video processing system. The implementation is carried out using the Python programming language and the OpenCV library. Performance evaluation is conducted using standard metrics, including precision, recall, and accuracy, to assess the system's effectiveness. A comparison with existing methods reported in the literature is performed to showcase the superiority of the proposed solution. A diverse dataset of images and videos containing motorcyclists and vehicles with visible number plates is collected and annotated.

**1.1 SYSTEM SPECIFICATIONS**

**1.1.1 HARDWARE CONFIGURATION:**

* System : Pentium IV 2.4 GHz.
* Hard Disk : 500 GB.
* Monitor : 15 VGA Colour.
* Mouse : Logitech.
* RAM : 4 GB.

**1.1.2 SOFTWARE SPECIFICATION:**

* Operating system : Windows 10/11 (64Bit).
* Coding Language : Python 3.10
* IDE Tools : Visual Studio Code 1.5

**CHAPTER - 2**

**SYSTEM STUDY**

**2.1 EXISTING SYSTEM**

**2.1.1 DESCRIPTION**

The Existing system monitors the traffic violations primarily through CCTV recordings, where the traffic police have to look into the frame where the traffic violation is happening, zoom into the license plate in case rider is not wearing helmet. But this requires lot of manpower and time as the traffic violations frequently and the number of people using motorcycles is increasing day-by-day. But these works are limited with respect to efficiency, accuracy or the speed with which object detection and classification is done.

**2.1.2 DRAWBACKS**

* The risk of deaths and injuries can be minimized significantly.
* Less accuracy.
* Without helmet increase the risk of injury to the neck and spine.

**2.2 PROPOSED SYSTEM**

**2.2.1 DESCRIPTION**

The proposed system for automatic helmet detection and number plate recognition using the YOLOv5 algorithm. The system aims to enhance safety and security by accurately detecting helmets worn by motorcyclists and recognizing the number plates of vehicles in real time. They consist of two main components: helmet detection and number plate recognition. It leverages the YOLOv5 algorithm, which is a state-of-the-art object detection framework known for its high accuracy and efficiency. By combining advanced computer vision techniques and machine learning algorithms, the system can effectively detect helmets and extract number plate information from video frames.

**2.2.2 FEATURES**

* Can prevent permanent disability or death.
* Can prevent brain damage.
* Can reduce medical expenses in case of an accident.
* Reduced risk of injury to the neck and spine.

**CHAPTER -3**

**SYSTEM DESIGN AND DEVELOPMENT**

**3.1 FILE DESIGN**

**1. Data Collection and Preprocessing**

The first step in designing a file for helmet detection and number plate recognition using machine learning is to collect and preprocess the data. This involves gathering images of motorcycles with and without helmets, as well as images of vehicle number plates. The images should be labeled appropriately, indicating whether a helmet is present or not, and the location of the number plate in the image.

Once the data has been collected, it should be preprocessed to improve the quality of the images and remove any irrelevant features. This may include resizing the images, converting them to grayscale, and applying filters to enhance the features of interest. The preprocessed data should then be split into training, validation, and testing sets to ensure that the model can generalize well to new data.

**2. Model Architecture**

The next step is to design the model architecture for helmet detection and number plate recognition. For helmet detection, a convolutional neural network (CNN) can be used to identify the presence or absence of a helmet in an image. The CNN can be trained on the preprocessed data collected in the first step.

For number plate recognition, optical character recognition (OCR) techniques can be used to extract text from the number plate. This involves using a CNN to detect the location of the number plate in the image, followed by an OCR engine to extract the text from the number plate. The OCR engine can be trained on a separate dataset of number plates with corresponding text labels.

**3. Training and Evaluation**

Once the model architecture has been designed, the next step is to train and evaluate the model on the preprocessed data. During training, the model weights are updated to minimize the loss function between the predicted output and the true output. The model can be evaluated on a separate testing set to assess its performance on new data.

The performance of the model can be measured using various metrics such as accuracy, precision, recall, and F1 score. These metrics can be used to compare different models and select the best one for deployment.

**4. Deployment and Monitoring**

After training and evaluating the model, it can be deployed in a real-world setting for helmet detection and number plate recognition. The model can be integrated into a larger system such as a traffic management system or a security system.

Once deployed, the model should be continuously monitored to ensure that it is performing as expected. This involves tracking various metrics such as accuracy, latency, and throughput. If any issues are detected, the model may need to be retrained or fine-tuned on new data.

**3.2 INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.

**OBJECTIVES**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

**3.3 OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

**3.4 CODE DESIGN**

**Helmet Detection**

The helmet detection system uses object detection algorithms to identify the presence or absence of a helmet in an image or video stream. The following steps are involved in designing the helmet detection system:

**Data Collection:** The first step is to collect a dataset of images containing people wearing helmets and people not wearing helmets. This dataset can be created using publicly available datasets, such as the Open Images Dataset, or by collecting images from the internet or real-world scenarios.

**Data Preprocessing:** Once the dataset is collected, it needs to be preprocessed to prepare it for training the object detection algorithm. This involves resizing the images, converting them to grayscale, and applying data augmentation techniques to increase the size of the dataset.

**Model Selection:** The next step is to select an appropriate object detection algorithm for detecting helmets in images. Popular object detection algorithms include Faster R-CNN, YOLO, and SSD. These algorithms can be trained on the preprocessed dataset to detect helmets in images.

**Model Training:** After selecting an appropriate model, it needs to be trained on the preprocessed dataset. This involves feeding the dataset into the model and adjusting the model parameters to minimize the training loss.

**Model Evaluation:** Once the model is trained, it needs to be evaluated on a separate test dataset to measure its performance. Popular evaluation metrics for object detection include Intersection over Union (IoU), Precision, Recall, and F1 score.

**Model Deployment:** Finally, the trained model can be deployed in a real-world scenario to detect helmets in images or video streams.

**Number Plate Recognition**

The number plate recognition system uses image processing and Optical Character Recognition (OCR) algorithms to identify the number plate in an image or video stream and extract the text from it. The following steps are involved in designing the number plate recognition system:

**Image Preprocessing:** The first step is to preprocess the image to enhance the number plate region and remove noise from the background. This involves applying image processing techniques such as thresholding, edge detection, and region segmentation.

**Number Plate Localization:** After preprocessing the image, the next step is to localize the number plate region in the image. This involves applying image segmentation techniques such as Hough Transform or Connected Component Analysis (CCA) to extract the number plate region from the image.

**Character Segmentation:** Once the number plate region is localized, the next step is to segment each character in the number plate region from each other. This involves applying image segmentation techniques such as vertical line segmentation or horizontal line segmentation.

**Optical Character Recognition (OCR):** After segmenting each character, OCR algorithms can be applied to extract text from each character region. Popular OCR algorithms include Tesseract OCR or OCRopus OCR.

**Text Postprocessing:** Finally, postprocessing techniques such as regular expressions can be applied to extract only alphanumeric characters from the extracted text and remove any non-alphanumeric characters such as special symbols or spaces.

**System Deployment:** Finally, the number plate recognition system can be deployed in a real-world scenario to extract text from number plates in images or video streams.

**3.5 DATABASE DESIGN**

When designing a database for a project involving helmet detection and number plate recognition using machine learning, it is crucial to consider the various components that will interact with the database, the data flow, storage requirements, and the overall efficiency of the system. Below are some key considerations for designing the database:

**1. Data Schema Design:**

Define the entities involved in the project such as images, detected helmets, recognized number plates, timestamps, and any other relevant information.

Establish relationships between these entities to ensure data integrity and efficient querying.

**2. Image Storage:**

Decide on how images will be stored in the database. Consider storing images directly in the database or storing references to image files in a separate storage system.

Optimize image storage to balance between performance and storage space.

**3. Machine Learning Model Integration:**

Determine how the machine learning models for helmet detection and number plate recognition will interact with the database.

Store model parameters, training data, and prediction results effectively within the database.

**4. Real-time Data Processing:**

Plan for real-time data processing requirements if the system needs to handle live video feeds for helmet detection and number plate recognition.

Implement efficient data pipelines to process incoming data streams.

**5. Security and Privacy:**

Implement robust security measures to protect sensitive data such as images, number plates, and any personal information that may be captured.

Ensure compliance with data protection regulations regarding image and personal data handling.

**6. Scalability and Performance:**

Design the database to scale horizontally or vertically based on the expected workload.

Optimize queries and indexing strategies to enhance performance as the dataset grows.

**7. Backup and Recovery:**

Establish regular backup procedures to prevent data loss in case of system failures or corruption.

Test recovery mechanisms to ensure quick restoration of data in case of emergencies.

**3.6 SYSTEM DEVELOPMENT**

**Components of the System:**

**Helmet Detection Module:** This module uses object detection algorithms such as YOLO (You Only Look Once) or Faster R-CNN to identify and locate helmets in images or video frames. The model is trained on a dataset containing images of individuals wearing helmets and without helmets.

**Number Plate Recognition Module:** The number plate recognition module utilizes optical character recognition (OCR) techniques to extract text from vehicle number plates. This involves preprocessing the image, segmenting characters, and then recognizing the characters using machine learning models like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs).

**Integration of Modules**: Once the helmet detection and number plate recognition modules are developed, they need to be integrated into a cohesive system that can process real-time video streams or images. This integration involves designing a pipeline that feeds the output of the helmet detection module into the number plate recognition module.

**Machine Learning Models:** The success of this project heavily relies on training accurate machine learning models for both helmet detection and number plate recognition. The models need to be trained on diverse datasets to ensure robust performance in various environmental conditions.

**User Interface:** A user-friendly interface should be designed to display the results of helmet detection and number plate recognition in a clear and understandable manner. This interface could be a web application, mobile app, or desktop software depending on the project requirements.

**3.6.1 MODULE DESCRIPTION:**

* Dataset Preparation
* YOLOv5 Model Training
* Helmet Detection
* Number Plate Recognition
* Integration and Evaluation

**DATASET PREPARATION:**

A diverse dataset of images and videos containing motorcyclists and vehicles with visible number plates is collected and annotated. The dataset ensures a balanced distribution of positive and negative examples, providing a comprehensive training and testing dataset for the system.

**YOLOV5 MODEL TRAINING:**

The YOLOv5 architecture is fine-tuned using the annotated dataset. Data augmentation techniques, such as random scaling, rotation, and flipping, are applied to augment the dataset's diversity. Transfer learning is utilized by initializing the YOLOv5 model with weights from a pre-trained model trained on a large-scale dataset. Hyperparameter tuning is performed to optimize the model's configuration, including learning rate, batch size, and training iterations.

**HELMET DETECTION:**

The trained YOLOv5 model is employed to detect helmets in video frames. The model performs inference, predicting bounding boxes around the detected helmets. Non-maximum suppression (NMS) is then utilized to refine the results and eliminate redundant detections, improving the overall accuracy of the helmet detection process.

**NUMBER PLATE RECOGNITION:**

Number Plate Recognition: Regions of interest (ROIs) corresponding to the detected number plates are extracted based on the helmet detection results. Pre-processing techniques, such as noise reduction, contrast adjustment, and thresholding, are applied to enhance the image quality of the number plate regions. Optical character recognition (OCR) algorithms are then used to recognize and extract alphanumeric characters from the pre-processed number plate regions.

**INTEGRATION AND EVALUATION:**

The helmet detection and number plate recognition components are seamlessly integrated into a unified real-time video processing system. The implementation is carried out using the Python programming language and the OpenCV library. Performance evaluation is conducted using standard metrics, including precision, recall, and accuracy, to assess the system's effectiveness.

**CHAPTER - 4**

**TESTING AND IMPLEMENTATION**

**SYSTEM TESTING**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**TYPES OF TESTS:-**

**UNIT TESTING:-**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**INTEGRATION TESTING:-**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**FUNCTIONAL TEST:-**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**SYSTEM TEST**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**WHITE BOX TESTING**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**BLACK BOX TESTING**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test Strategy and Approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

**Integration Testing:**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or - one step up - software applications at the company level - interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing:**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**IMPLEMENTATION**

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it’s constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

**CHAPTER-V**

**CONCLUSION**

Using Convolutional Neural Networks (CNNs) and transfer learning, the system achieves high accuracy in identifying helmetless riders. However, the framework goes beyond just detecting these riders, as it also recognizes and stores the license plate numbers of their motorcycles. By storing the license plate numbers, the framework provides a means of identifying and penalizing riders who violate helmet laws. This comprehensive approach makes the system an effective tool for promoting road safety. The implementation of this framework can have a significant impact on road safety, especially since it can be integrated with existing CCTV networks. Furthermore, the use of transfer learning makes the system adaptable to different environments, making it a scalable solution that can be used in various locations. Overall, this project demonstrates the potential of using AI and machine learning to improve road safety. By providing a comprehensive solution for detecting helmetless riders and identifying their vehicles, this framework has the potential to reduce accidents and save lives.

**CHAPTER-VI**

**BIBLIOGRAPHY**

1. **S. Senthilkumar**, **K. Gomathi**, and **R. Jayaparvathy**, "**Helmet detection and license plate recognition for intelligent traffic management system**," in 2018 International Conference on Smart City and Emerging Technology (ICSCET), 2018, pp. 1-5. [DOI: 10.1109/ICSCET.2018.8478799]
2. **K. Zhang**, **Z. Zhang**, **Z. Li**, and **Y. Qiao**, "**Joint face detection and alignment using multitask cascaded convolutional networks,**" IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499-1503, 2016. [DOI: 10.1109/LSP.2016.2603342]
3. **Y. LeCun**, **Y. Bengio**, and **G. Hinton**, "**Deep learning**," Nature, vol. 521, no. 7553, pp. 436-444, 2015. [DOI: 10.1038/nature14539]
4. **Y. Bengio**, **A. Courville**, and **P. Vincent**, "**Representation learning: A review and new perspectives,**" IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798-1828, 2013. [DOI: 10.1109/TPAMI.2013.50]
5. **J. Redmon**, **S. Divvala**, **R. Girshick**, and **A. Farhadi**, "**You only look once: Unified, real-time object detection**," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788. [DOI: 10.1109/CVPR.2016.91]
6. **R. Girshick**, "**Fast R-CNN**," in Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1440-1448. [DOI: 10.1109/ICCV.2015.169]
7. **J. Deng**, **W. Dong, R. Socher**, **L.-J. Li**, **K. Li**, **and L. Fei-Fei**, "**ImageNet: A large-scale hierarchical image database,**" in 2009 IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 248-255. [DOI: 10.1109/CVPR.2009.5206848]
8. **O. Russakovsky et al**., "**ImageNet Large Scale Visual Recognition Challenge,**" International Journal of Computer Vision, vol. 115, no. 3, pp. 211-252, 2015. [DOI: 10.1007/s11263-015-0816-y]
9. **Krizhevsky**, **I. Sutskever**, and **G. Hinton**, "**ImageNet classification with deep convolutional neural networks**," in Advances in Neural Information Processing Systems 25 (NIPS 2012), 2012, pp. 1097-1105.
10. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6517-6525. [DOI: 10.1109/CVPR.2017.690]

**APPENDICES**

1. **DATA FLOW DIAGRAM:**
2. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
3. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
4. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

Start

Stop

Input Video

Frames Reading

Feature Matching

Person with Helmet

Number Plate Detection

Accuracy Prediction

**USE CASE DIAGRAM**

User

1. **TABLE STRUCTURE**

**Image Data Table:**

Image ID (Primary Key)

Image Path

Helmet Detection Result

Number Plate Recognition Result

**Helmet Detection Results Table:**

Detection ID (Primary Key)

Image ID (Foreign Key)

Helmet Detected (Boolean)

Confidence Level

**Number Plate Recognition Results Table:**

Recognition ID (Primary Key)

Image ID (Foreign Key)

Number Plate Text

Confidence Level

**Helmet Data Table:**

Helmet ID (Primary Key)

Helmet Type

Color

Size

Manufacturer

**Number Plate Data Table:**

Plate ID (Primary Key)

Plate Text

Vehicle Type

State

Training Data Table:

Training ID (Primary Key)

Image ID (Foreign Key)

Features Extracted

Label (Helmet Detected/Not Detected, Number Plate Recognized/Not Recognized)

**Model Performance Table:**

Model ID (Primary Key)

Model Name

Training Accuracy

Testing Accuracy

**User Feedback Table:**

Feedback ID (Primary Key)

Image ID (Foreign Key)

User Comments

User Ratings

1. **SAMPLE CODING**

**Detect.py:**

import cv2

import numpy as np

import os

from PIL import Image

import time

import imutils

from keras.models import load\_model

os.environ['TF\_FORCE\_GPU\_ALLOW\_GROWTH'] = 'true'

net = cv2.dnn.readNet("yolov3-custom\_7000.weights", "yolov3-custom.cfg")

net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_CUDA)

net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CUDA)

model = load\_model('helmet-nonhelmet\_cnn.h5')

print('model loaded!!!')

cap = cv2.VideoCapture('video.mp4')

COLORS = [(0,255,0),(0,0,255)]

layer\_names = net.getLayerNames()

output\_layers = [layer\_names[i - 1] for i in net.getUnconnectedOutLayers()]

fourcc = cv2.VideoWriter\_fourcc(\*"XVID")

writer = cv2.VideoWriter('output.avi', fourcc, 5,(888,500))

def helmet\_or\_nohelmet(helmet\_roi):

try:

helmet\_roi = cv2.resize(helmet\_roi, (224, 224))

helmet\_roi = np.array(helmet\_roi,dtype='float32')

helmet\_roi = helmet\_roi.reshape(1, 224, 224, 3)

helmet\_roi = helmet\_roi/255.0

return int(model.predict(helmet\_roi)[0][0])

except:

pass

ret = True

while ret:

ret, img = cap.read()

img = imutils.resize(img,height=500)

# img = cv2.imread('test.png')

height, width = img.shape[:2]

blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

net.setInput(blob)

outs = net.forward(output\_layers)

confidences = []

boxes = []

classIds = []

for out in outs:

for detection in out:

scores = detection[5:]

class\_id = np.argmax(scores)

confidence = scores[class\_id]

if confidence > 0.3:

center\_x = int(detection[0] \* width)

center\_y = int(detection[1] \* height)

w = int(detection[2] \* width)

h = int(detection[3] \* height)

x = int(center\_x - w / 2)

y = int(center\_y - h / 2)

boxes.append([x, y, w, h])

confidences.append(float(confidence))

classIds.append(class\_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

for i in range(len(boxes)):

if i in indexes:

x,y,w,h = boxes[i]

color = [int(c) for c in COLORS[classIds[i]]]

# green --> bike

# red --> number plate

if classIds[i]==0: #bike

helmet\_roi = img[max(0,y):max(0,y)+max(0,h)//4,max(0,x):max(0,x)+max(0,w)]

else: #number plate

x\_h = x-60

y\_h = y-350

w\_h = w+100

h\_h = h+100

cv2.rectangle(img, (x, y), (x + w, y + h), color, 7)

# h\_r = img[max(0,(y-330)):max(0,(y-330 + h+100)) , max(0,(x-80)):max(0,(x-80 + w+130))]

if y\_h>0 and x\_h>0:

h\_r = img[y\_h:y\_h+h\_h , x\_h:x\_h +w\_h]

c = helmet\_or\_nohelmet(h\_r)

cv2.putText(img,['helmet','no-helmet'][c],(x,y-100),cv2.FONT\_HERSHEY\_SIMPLEX,2,(0,255,0),2)

cv2.rectangle(img, (x\_h, y\_h), (x\_h + w\_h, y\_h + h\_h),(255,0,0), 10)

writer.write(img)

cv2.imshow("Image", img)

if cv2.waitKey(1) == 27:

break

writer.release()

cap.release()

cv2.waitKey(0)

cv2.destroyAllWindows()

**Helmet Code:**

import cv2

import numpy as np

import os

from PIL import Image

import time

import imutils

from keras.models import load\_model

os.environ['TF\_FORCE\_GPU\_ALLOW\_GROWTH'] = 'true'

net = cv2.dnn.readNet("yolov3-custom\_7000.weights", "yolov3-custom.cfg")

net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_CUDA)

net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CUDA)

model = load\_model('helmet-nonhelmet\_cnn.h5')

print('model loaded!!!')

cap = cv2.VideoCapture('video.mp4')

COLORS = [(0,255,0),(0,0,255)]

layer\_names = net.getLayerNames()

output\_layers = [layer\_names[i - 1] for i in net.getUnconnectedOutLayers()]

fourcc = cv2.VideoWriter\_fourcc(\*"XVID")

writer = cv2.VideoWriter('output.avi', fourcc, 5,(888,500))

def helmet\_or\_nohelmet(helmet\_roi):

try:

helmet\_roi = cv2.resize(helmet\_roi, (224, 224))

helmet\_roi = np.array(helmet\_roi,dtype='float32')

helmet\_roi = helmet\_roi.reshape(1, 224, 224, 3)

helmet\_roi = helmet\_roi/255.0

return int(model.predict(helmet\_roi)[0][0])

except:

pass

ret = True

while ret:

ret, img = cap.read()

img = imutils.resize(img,height=500)

# img = cv2.imread('test.png')

height, width = img.shape[:2]

blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

net.setInput(blob)

outs = net.forward(output\_layers)

confidences = []

boxes = []

classIds = []

for out in outs:

for detection in out:

scores = detection[5:]

class\_id = np.argmax(scores)

confidence = scores[class\_id]

if confidence > 0.3:

center\_x = int(detection[0] \* width)

center\_y = int(detection[1] \* height)

w = int(detection[2] \* width)

h = int(detection[3] \* height)

x = int(center\_x - w / 2)

y = int(center\_y - h / 2)

boxes.append([x, y, w, h])

confidences.append(float(confidence))

classIds.append(class\_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

for i in range(len(boxes)):

if i in indexes:

x,y,w,h = boxes[i]

color = [int(c) for c in COLORS[classIds[i]]]

# green --> bike

# red --> number plate

if classIds[i]==0: #bike

helmet\_roi = img[max(0,y):max(0,y)+max(0,h)//4,max(0,x):max(0,x)+max(0,w)]

else: #number plate

x\_h = x-60

y\_h = y-350

w\_h = w+100

h\_h = h+100

cv2.rectangle(img, (x, y), (x + w, y + h), color, 7)

# h\_r = img[max(0,(y-330)):max(0,(y-330 + h+100)) , max(0,(x-80)):max(0,(x-80 + w+130))]

if y\_h>0 and x\_h>0:

h\_r = img[y\_h:y\_h+h\_h , x\_h:x\_h +w\_h]

c = helmet\_or\_nohelmet(h\_r)

cv2.putText(img,['helmet','no-helmet'][c],(x,y-100),cv2.FONT\_HERSHEY\_SIMPLEX,2,(0,255,0),2)

cv2.rectangle(img, (x\_h, y\_h), (x\_h + w\_h, y\_h + h\_h),(255,0,0), 10)

writer.write(img)

cv2.imshow("Image", img)

if cv2.waitKey(1) == 27:

break

writer.release()

cap.release()

cv2.waitKey(0)

cv2.destroyAllWindows()

Helmet Position Prediction:

import cv2

import numpy as np

import os

from PIL import Image

import time

import imutils

from keras.models import load\_model

os.environ['TF\_FORCE\_GPU\_ALLOW\_GROWTH'] = 'true'

net = cv2.dnn.readNet("yolov3-custom\_7000.weights", "yolov3-custom.cfg")

net.setPreferableBackend(cv2.dnn.DNN\_BACKEND\_CUDA)

net.setPreferableTarget(cv2.dnn.DNN\_TARGET\_CUDA)

model = load\_model('helmet-nonhelmet\_cnn.h5')

print('model loaded!!!')

cap = cv2.VideoCapture('video.mp4')

COLORS = [(0,255,0),(0,0,255)]

layer\_names = net.getLayerNames()

output\_layers = [layer\_names[i - 1] for i in net.getUnconnectedOutLayers()]

fourcc = cv2.VideoWriter\_fourcc(\*"XVID")

writer = cv2.VideoWriter('output.avi', fourcc, 5,(888,500))

def helmet\_or\_nohelmet(helmet\_roi):

try:

helmet\_roi = cv2.resize(helmet\_roi, (224, 224))

helmet\_roi = np.array(helmet\_roi,dtype='float32')

helmet\_roi = helmet\_roi.reshape(1, 224, 224, 3)

helmet\_roi = helmet\_roi/255.0

return int(model.predict(helmet\_roi)[0][0])

except:

pass

ret = True

while ret:

ret, img = cap.read()

img = imutils.resize(img,height=500)

# img = cv2.imread('test.png')

height, width = img.shape[:2]

blob = cv2.dnn.blobFromImage(img, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

net.setInput(blob)

outs = net.forward(output\_layers)

confidences = []

boxes = []

classIds = []

for out in outs:

for detection in out:

scores = detection[5:]

class\_id = np.argmax(scores)

confidence = scores[class\_id]

if confidence > 0.3:

center\_x = int(detection[0] \* width)

center\_y = int(detection[1] \* height)

w = int(detection[2] \* width)

h = int(detection[3] \* height)

x = int(center\_x - w / 2)

y = int(center\_y - h / 2)

boxes.append([x, y, w, h])

confidences.append(float(confidence))

classIds.append(class\_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)

for i in range(len(boxes)):

if i in indexes:

x,y,w,h = boxes[i]

color = [int(c) for c in COLORS[classIds[i]]]

# green --> bike

# red --> number plate

if classIds[i]==0: #bike

helmet\_roi = img[max(0,y):max(0,y)+max(0,h)//4,max(0,x):max(0,x)+max(0,w)]

else: #number plate

x\_h = x-60

y\_h = y-350

w\_h = w+100

h\_h = h+100

cv2.rectangle(img, (x, y), (x + w, y + h), color, 7)

# h\_r = img[max(0,(y-330)):max(0,(y-330 + h+100)) , max(0,(x-80)):max(0,(x-80 + w+130))]

if y\_h>0 and x\_h>0:

h\_r = img[y\_h:y\_h+h\_h , x\_h:x\_h +w\_h]

c = helmet\_or\_nohelmet(h\_r)

cv2.putText(img,['helmet','no-helmet'][c],(x,y-100),cv2.FONT\_HERSHEY\_SIMPLEX,2,(0,255,0),2)

cv2.rectangle(img, (x\_h, y\_h), (x\_h + w\_h, y\_h + h\_h),(255,0,0), 10)

writer.write(img)

cv2.imshow("Image", img)

if cv2.waitKey(1) == 27:

break

writer.release()

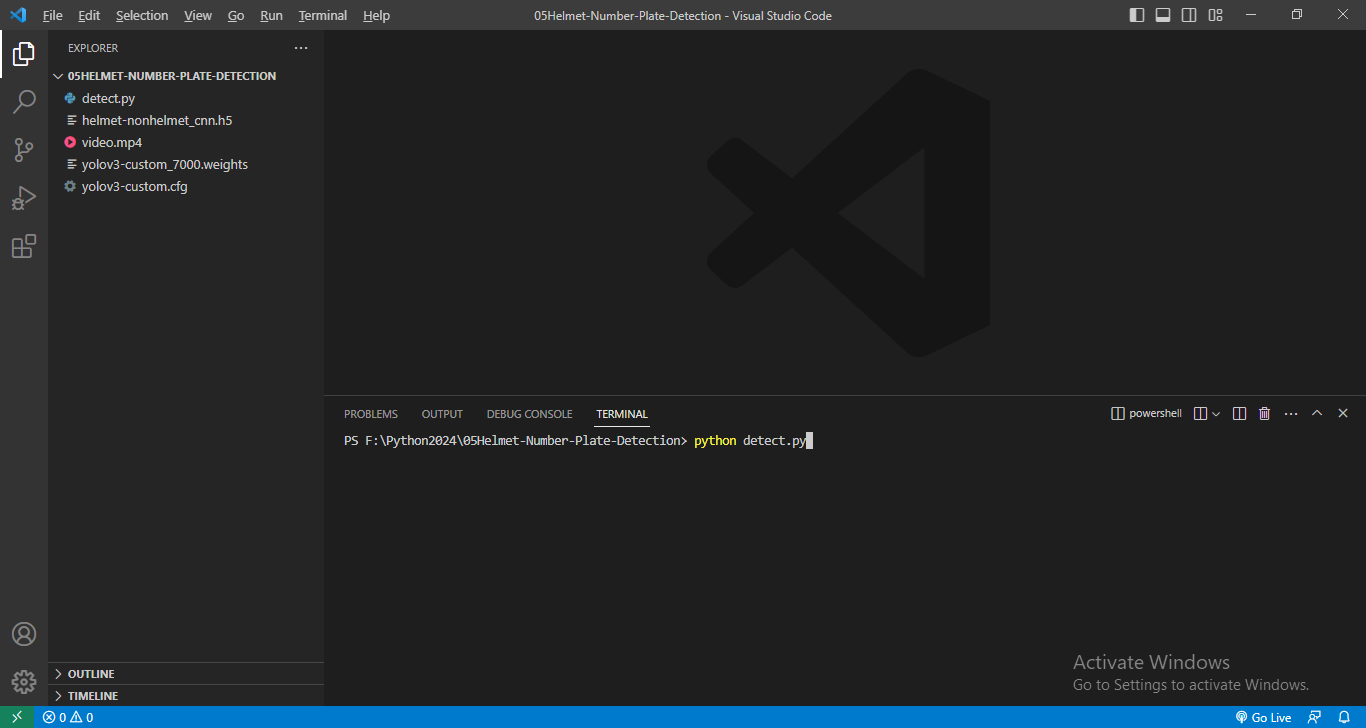
cap.release()

cv2.waitKey(0)

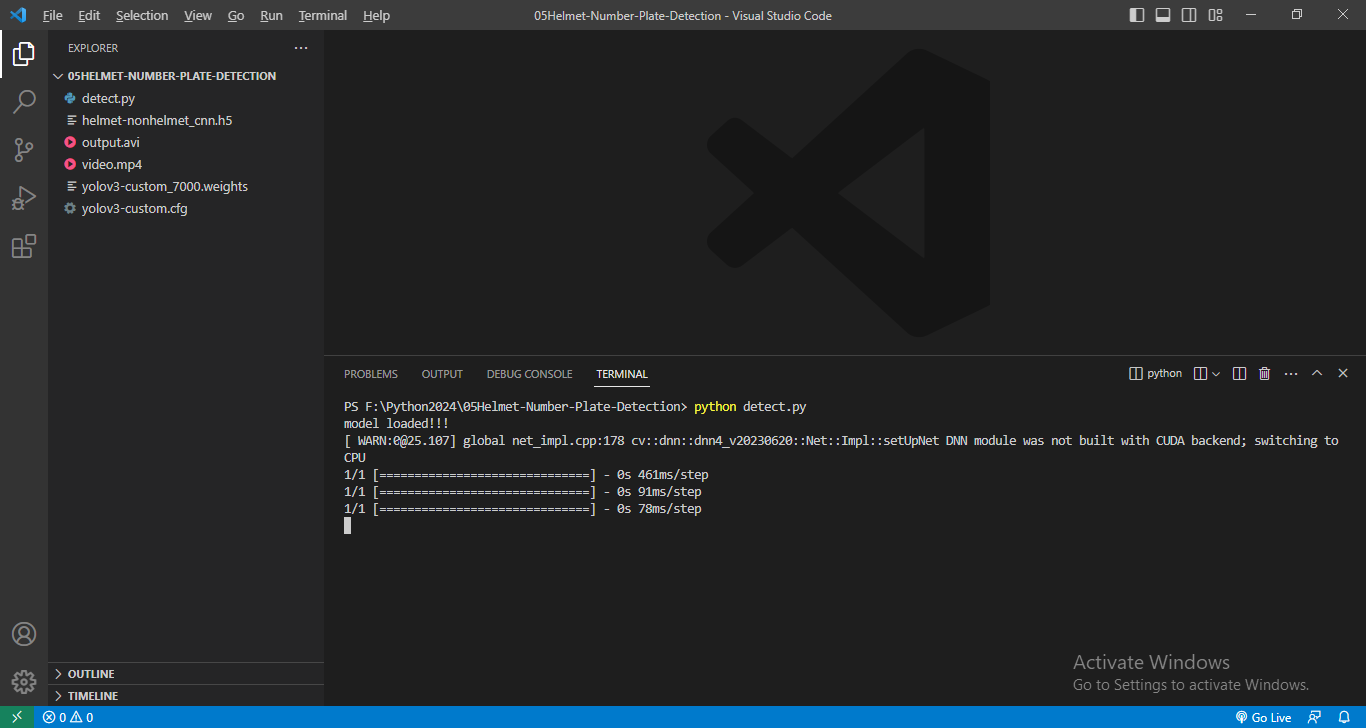
cv2.destroyAllWindows()

1. **SAMPLE INPUT**

Run: Detect.py

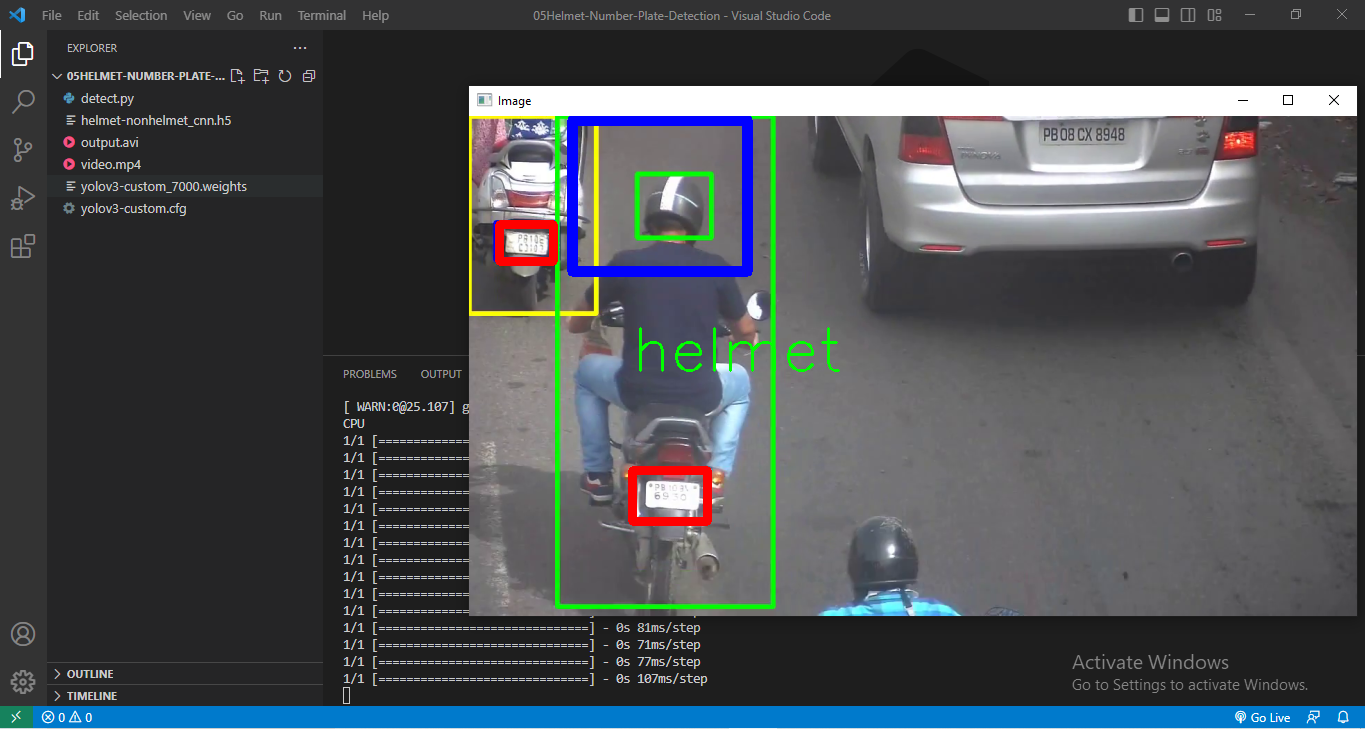


Video Frame Check:

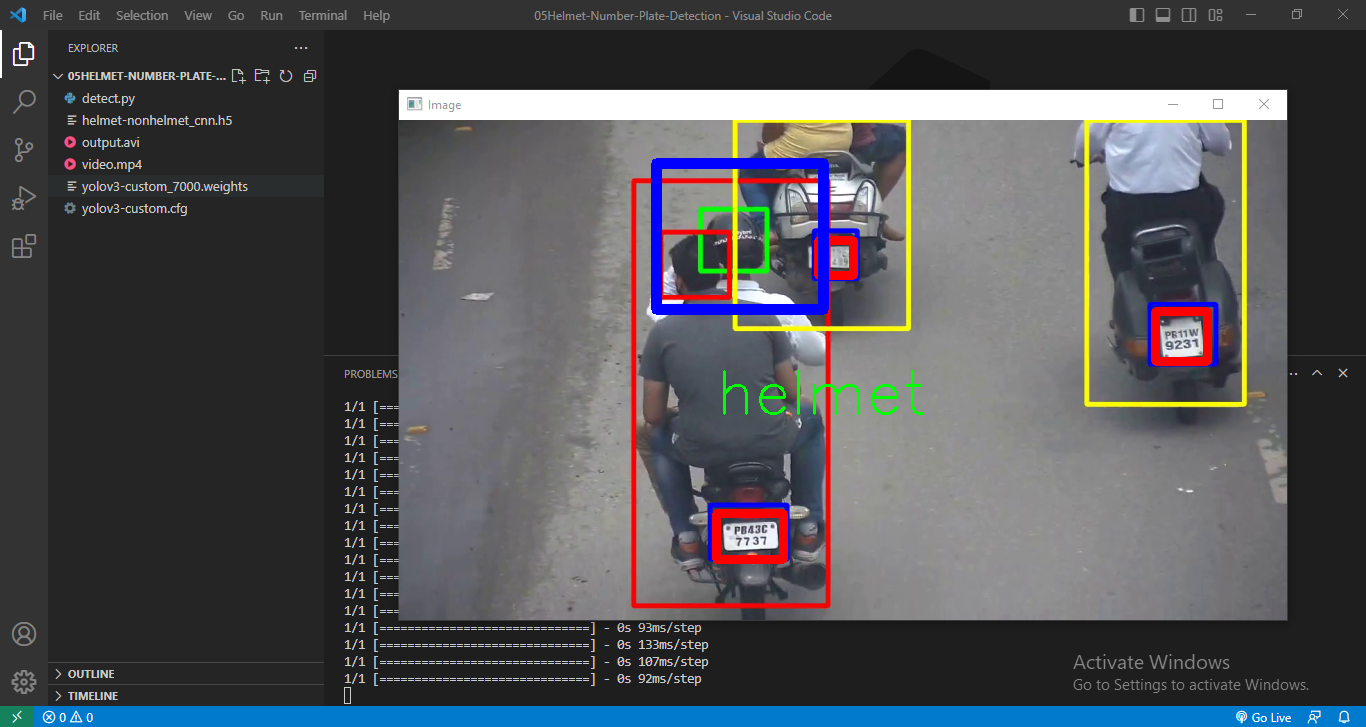


1. **SAMPLE OUTPUT**

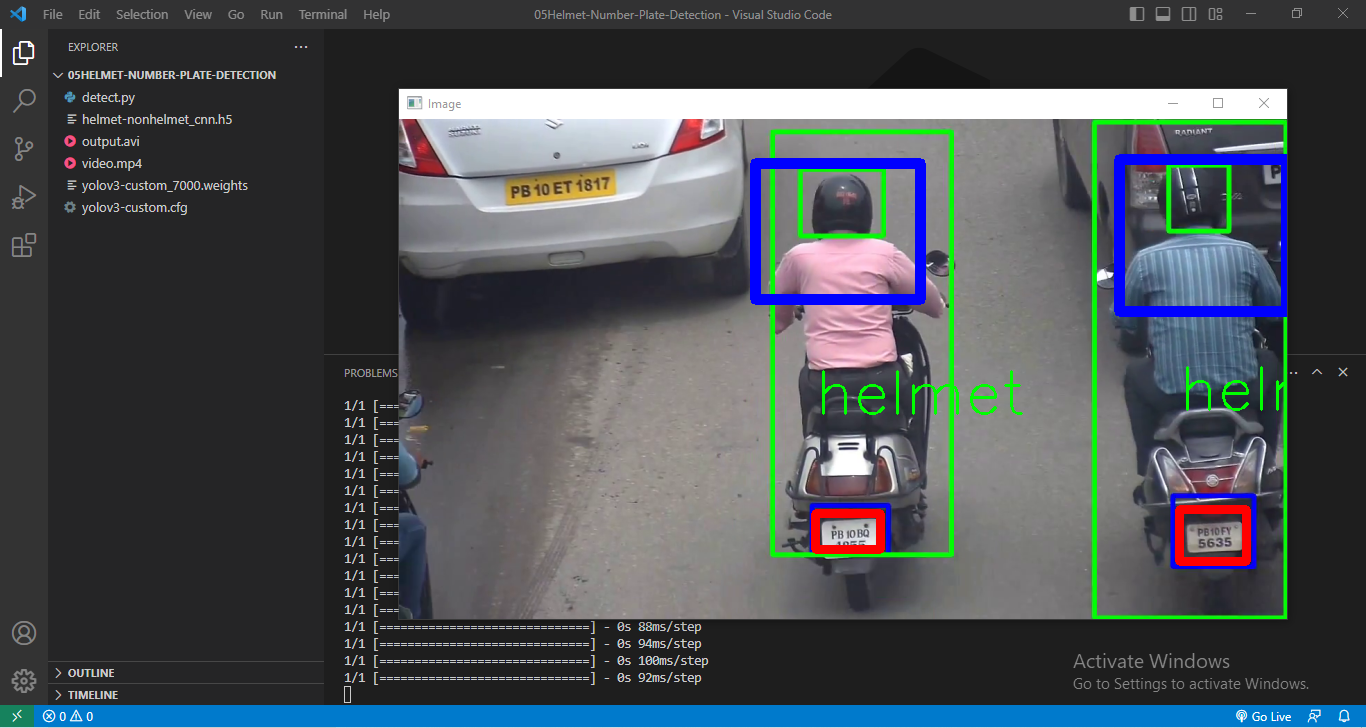
Helmet Predict:



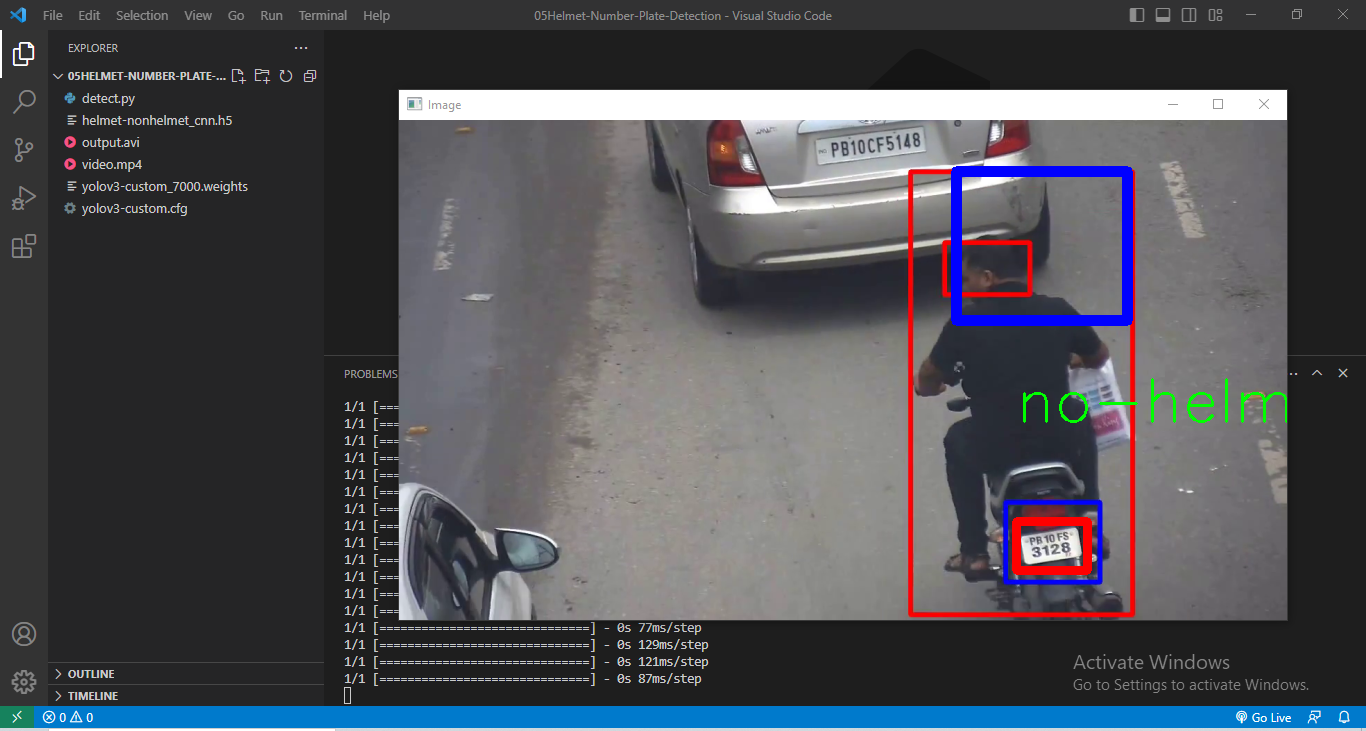
Helmet Predict:



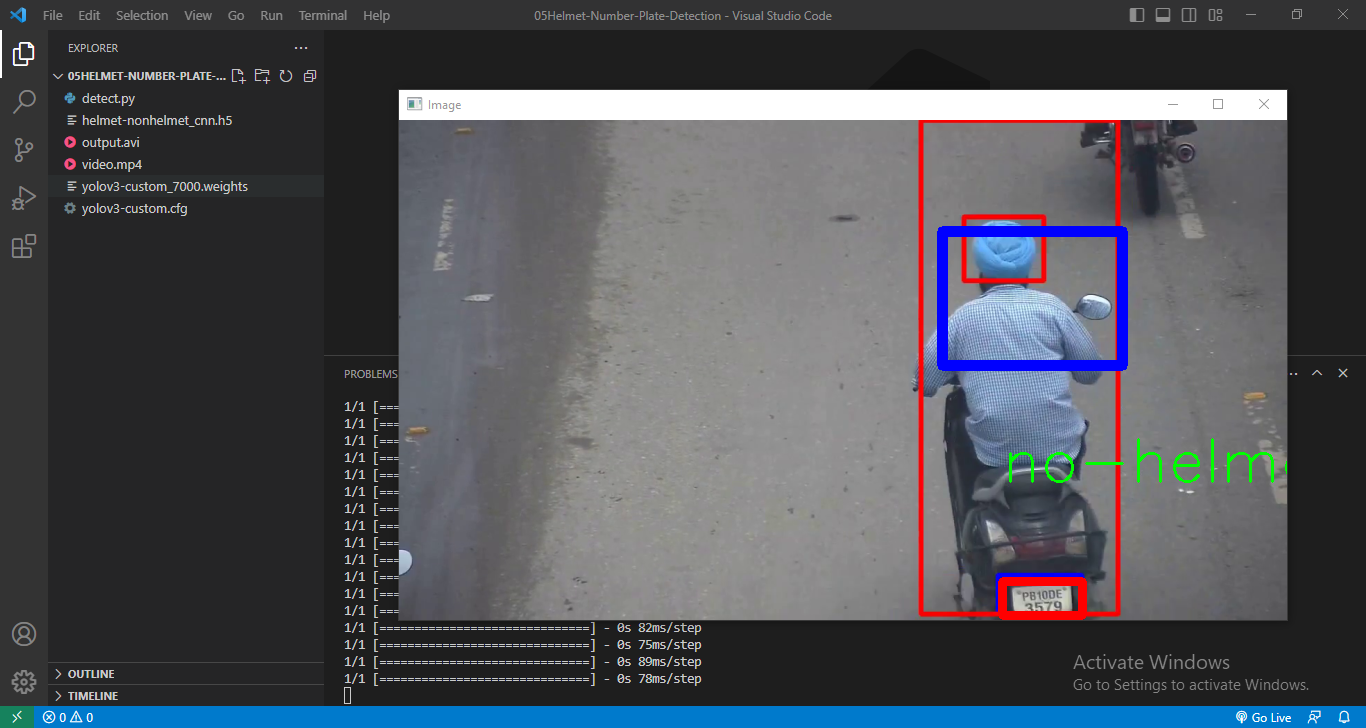
Helmet and Number Plate Predict:



No-Helmet Predict:



No-Helmet Predict:



No-Helmet Predict:

