RES MILITARIS
Social Science Journal

ARTIFICIAL INTELLIGENCE-BASED HELMET DETECTION FOR TRAFFIC CHALLAN SYSTEM

Sudheer Devulapalli Assistance Professor
N Rajesh Assistance Professor
R Ramadevi Assistance Professor
D Aravinda Assistance Professor
Dept of CSE
Sree Dattha Institute of Engineering and Science

ABSTRACT

In current situation, we come across various problems in traffic regulations in India which can be solved with different ideas. Riding motorcycle/mopeds without wearing helmet is a traffic violation which has resulted in increase in number of accidents and deaths in India. 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. What if there is a system, which would automatically look for traffic violation of not wearing helmet while riding motorcycle/moped and if so, would automatically extract the vehicles' license plate number. Recent research has successfully done this work based on machine learning methods. But these works are limited with respect to efficiency, accuracy or the speed with which object detection, which resulted in lower performance. In this research work, a Non-Helmet Rider detection system is built which attempts to satisfy the automation of detecting the traffic violation of not wearing helmet and extracting the vehicles' license plate number. The main principle involved is Object Detection using Deep Learning at three levels. The objects detected are a person, motorcycle/moped at first level, helmet at second level using you only look at once (YOLO) model.

Introduction

1.1 Overview

Helmet detection for riders is a crucial aspect of road safety and is typically implemented through various technological means, such as computer vision systems, deep learning algorithms, and surveillance cameras. This technology is designed to identify and verify whether a motorcycle or scooter rider is wearing a helmet while on the road.

The process of helmet detection begins with the deployment of cameras strategically positioned at key points on roads, intersections, or highways, often as part of a broader traffic management or safety system. These cameras continuously capture real-time video footage of the traffic, including riders on two-wheelers.

Computer vision algorithms, powered by machine learning models like Convolutional Neural Networks (CNNs), play a pivotal role in helmet detection. These algorithms analyze the incoming video stream, identifying and tracking the presence of riders in the frame. Once a rider is detected, the system focuses on the head region to determine whether they are wearing a helmet. It does so by recognizing the distinct shape and color of a helmet. If a helmet is detected on the rider's head, the system registers a positive result; otherwise, it flags the rider as non-compliant.

Helmet detection systems often include advanced features such as motion analysis, which helps in distinguishing between riders and pedestrians, and can even evaluate the proper fastening of the helmet straps. Additionally, some systems incorporate real-time alerts and notifications to inform law enforcement or relevant authorities about non-compliant riders.

The importance of helmet detection lies in its contribution to road safety. Wearing helmets significantly reduces the risk of head injuries during accidents, making it a crucial safety measure for motorcycle and scooter riders. By enforcing helmet-wearing regulations through automated detection systems, governments and traffic authorities aim to decrease the number of head injuries and fatalities on the road.

In summary, helmet detection systems employ computer vision and deep learning technologies to identify riders and assess whether they are wearing helmets. These systems play a vital role in enhancing road safety by enforcing helmet-wearing regulations, reducing the risk of head injuries, and promoting responsible riding habits among two-wheeler users.

1.2 Research Motivation

The motivation behind conducting research on helmet detection systems for riders stems from a profound commitment to road safety and the urgent need to mitigate the risks associated with motorcycle and scooter accidents. Two-wheeler accidents are a significant contributor to road fatalities and severe injuries worldwide, and head injuries are a leading cause of mortality and long-term disability in these incidents. Thus, the development and implementation of effective helmet detection technology hold immense promise in reducing these tragic outcomes.

First and foremost, the research in this area is motivated by the desire to save lives. Helmets are proven to be a highly effective means of protecting riders from traumatic brain injuries and fatalities during accidents. However, ensuring that riders consistently wear helmets is a formidable challenge, especially in regions where compliance with safety regulations is low. By automating the detection of helmet use, law enforcement and traffic authorities can more efficiently enforce helmet-wearing laws, ultimately leading to fewer head injuries and fatalities on the road.

Furthermore, this research is driven by a commitment to promoting responsible road behavior. Helmet detection systems serve not only as a deterrent for riders who may be tempted to ride without a helmet but also as a means of education and awareness. By identifying and flagging non-compliant riders, these systems encourage individuals to prioritize their safety and adhere to established safety regulations, thereby fostering a culture of responsible riding.

The advancement of computer vision and deep learning technologies has made it increasingly feasible to develop accurate and efficient helmet detection systems. This technological progress further motivates researchers to explore innovative solutions that can enhance road safety. The potential for real-time monitoring, automated alerts, and data-driven insights into helmet-wearing patterns presents exciting opportunities for improving traffic safety on a broader scale.

So, the research motivation behind helmet detection systems for riders is deeply rooted in the urgent need to save lives, reduce head injuries, and promote responsible road behavior. By leveraging cutting-edge technology, researchers aim to create effective tools that empower authorities to enforce helmet-wearing regulations effectively, thereby making a significant impact on road safety and public health.

1.3 Problem Statement

The problem statement for research on helmet detection systems for riders addresses a critical issue in road safety: the persistent and potentially life-threatening problem of non-compliance with helmet-wearing regulations among motorcycle and scooter riders. Despite the well-established benefits of helmets in reducing the severity of head injuries and fatalities during accidents, many riders choose not to wear them, leading to a significant public health concern.

One key aspect of this problem is the lack of efficient enforcement mechanisms. Traditional methods of policing, such as manual checks by law enforcement officers, are often resource-intensive, time-consuming, and can be subject to human error. This inefficiency allows non-compliant riders to go undetected, increasing the risk of serious injuries and fatalities. The problem is compounded by the fact that helmet-wearing regulations vary from region to region, and enforcement can be inconsistent.

Additionally, the problem statement acknowledges the limitations of existing technological solutions. While helmet detection systems powered by computer vision and deep learning hold great promise, there are challenges to overcome. These include the need for highly accurate and reliable detection algorithms that can operate effectively in diverse environmental conditions, such as varying lighting, weather, and traffic congestion. Moreover, the deployment and maintenance of surveillance cameras and associated hardware require significant financial investments, which may be a barrier for some regions.

Furthermore, the problem statement acknowledges the broader societal implications of non-compliance with helmet-wearing regulations. Motorcycle and scooter accidents resulting from riders not wearing helmets lead to increased healthcare costs, long-term disabilities, and the loss of human lives, placing a burden on healthcare systems and society as a whole.

So, the problem statement revolves around the pressing need to enhance helmet-wearing compliance among motorcycle and scooter riders, reduce the incidence of head injuries and fatalities, and improve the efficiency and effectiveness of enforcement mechanisms. The development of robust and cost-effective helmet detection systems, capable of operating in various conditions, is central to addressing this multifaceted problem and enhancing road safety for all road users.

1.4 Applications

The applications of helmet detection systems for riders are diverse and hold significant potential in improving road safety, law enforcement, and overall public health. Below, we delve into detailed applications of these systems:

- **Automated Law Enforcement:** Helmet detection systems can be integrated into traffic management infrastructure to automate the enforcement of helmet-wearing regulations. When a non-compliant rider is detected, the system can trigger real-time alerts to law enforcement officers, who can then take appropriate action, such as issuing citations. This automation not only reduces the workload of law enforcement personnel but also ensures a consistent and fair application of the law.
- **Data-Driven Policy and Safety Improvement:** Helmet detection systems generate valuable data on helmet-wearing compliance rates, locations of non-compliance hotspots, and trends over time. Traffic authorities can analyze this data to make informed decisions regarding the allocation of resources for safety campaigns, targeted law enforcement, and infrastructure improvements in areas with high non-compliance rates.

- **Public Awareness and Education:** The presence of helmet detection systems can serve as a visual reminder to riders about the importance of helmet use. Knowing that they are being monitored for compliance can motivate riders to adopt safer behavior. Furthermore, data collected by these systems can be used in public awareness campaigns to emphasize the lifesaving benefits of wearing helmets.
- **Emergency Response Enhancement:** In the unfortunate event of an accident, helmet detection systems can assist emergency responders in assessing the extent of injuries. Knowing whether a rider was wearing a helmet at the time of the incident can help prioritize medical interventions and allocate resources more efficiently.
- **Insurance and Legal Investigations:** Helmet detection data can be used by insurance companies and legal authorities to determine liability and establish facts in accident cases. This can expedite insurance claims and legal proceedings, reducing the burden on the affected parties.
- **Customized Rider Feedback:** Some helmet detection systems can provide immediate feedback to non-compliant riders through public address systems or electronic signage. This instant feedback can encourage riders to wear helmets on the spot, fostering safer riding practices.
- **Research and Policy Evaluation:** Researchers can use the data collected by helmet detection systems to study the effectiveness of helmet-wearing regulations, the impact of enforcement, and the correlation between helmet use and accident outcomes. This information is invaluable for refining existing policies and developing evidence-based road safety strategies.
- **Integration with Smart City Initiatives:** Helmet detection systems can be integrated into broader smart city initiatives, allowing for seamless coordination with other traffic management systems. This integration can facilitate real-time responses to traffic congestion, accidents, and emergencies, improving overall urban mobility and safety.

2. LITERATURE SURVEY

2.1 Related Work

Marzuki, P., et al. (2019) [1] proposed an improved Convolutional Neural Network (CNN) algorithm approach for license plate recognition system. The main contribution of this work is on the methodology to determine the best model for four-layered CNN architecture that has been used as the recognition method .Two types of accuracies are taken at different stages. The first accuracy is taken at the pre processing part the system and the second one is taken at the classification stage. The classification result was taken according to the number of characters successfully recognized. It described that the pre processing part achieved 74.7% out of 300 samples tested which does not achieve the expectation level.

Ma, Lixin, and Yong Zhang (2021) [2] proposed convolutional neural network. The cycle juice used was based on information obtained from the original. Experimental results showed that the method had a high accuracy rate of 94% and no return rate of 88% for the image data block.

Vaiyapuri, Thavavel, et al (2021) [3] proposed the technique, it had a total of four major processes namely pre processing, License Plate (LP) localization and detection, character segmentation, and recognition. Hough Transform (HT) was applied as a feature extractor and SSA-CNN algorithm was applied for

character recognition in LP. This study introduced a robust DL-based VLPR model using SSA-CNN model. The proposed model had a total of four major processes namely pre processing,LP localization and detection, HT-based character segmentation, and SSA-CNN based recognition. The input image was pre processed to make it compatible with further processing..

Huang, Zhao-Kai, Hao-Wei Tseng, and Cheng-Lun Chen (2019) [4] proposed new method for vehicle license plate recognition on the basis of Extreme Learning Machine. ELM was a new category of neural networks which possesses compelling characteristics essential for license plate recognition, such as low computational complexity, fast training, and good generalization (as opposed to traditional neural networks). The proposed method studied incorporation of three ELMs (i.e., basic ELM, incremental ELM, and enhanced incremental ELM) into a typical pipeline for automatic license plate recognition. In the preliminary study (under Windows PC with MATLAB), the success rate of recognition was 87.5% with execution time of 0.3s. A comparative study also showed that the proposed method outperformed conventional approaches (template matching, edge statistics, RBF, and SVM) in terms of accuracy and speed.

Neto, Edson Cavalcanti, et al (2019) [5] proposed a new system to detect and recognize Brazilian vehicle license plates, in which the registered users have permission to enter the location. For this, techniques of Digital Image Processing were used, such as Hough Transform, Morphology, Threshold and Canny Edge Detector to extract characters, as well as Least Squares, Least Mean Squares, Extreme Learning Machine, and Neural Network Multilayer Perceptron to identify the numbers and letters. The system was tested with 700 videos with a resolution of 640 × 480 pixels and AVI format, granting access only when the plate was registered, getting a 98.5% success rate on the tested cases. The movement detection step was linked to the system, becoming faster and more accurate in real time.

Halin, Alfian Abdul, et al (2020) [6] proposed a probabilistic technique to localize license plates regions for cars adhering to the standard set by the Malaysian Road Transport Department. Images of the front/rear-view of cars displaying their license plates are firstly pre processed, followed by features extraction generated from connected components analysis. These features were then used to train a Naïve Bayes classifier for the final task of license plates localization. Experimental results conducted on 144 images have shown that considering two candidates with the highest posterior probabilities better guarantees license plates regions were properly localized,with a recall of 0.98.

Akhtar, Zuhaib, and Rashid Ali (2020) [7] proposed a number plate recognition method by processing vehicle's rear or front image. After imaged was captured, processing is divided into four steps which are pre processing, number plate localization, character segmentation and character recognition. Pre processing enhances the image for further processing, number plate localization extracted the number plate region from the image, character segmentation separates the individual characters from the extracted number plate, and character recognition identified the optical characters by using random forest classification algorithm. Experimental results revealed that the accuracy of this method was 90.9%.

Sathiyabhama, B., et al (2020) [8] proposed a deep learning-based algorithm was proposed for detecting both vehicles and number plates for a reputed company surveillance dataset. The proposed model used a video dataset as an input and the video was segmented into several frames. Using pre-trained weights and labels of the dataset, the vehicles and its number plate were detected by the dark flow toolkit. This tool

provided to extract the region of the vehicle with proper annotation. In future work, the proposed model aimed to calculate the speed of the vehicle based on the surrounding area.

Tabrizi, Sahar S., and Nadire Cavus (2019) [9] proposed a LPR systems for Iranian license plates. Increasing the accuracy of the character recognition phase rate and decreasing the training rate were the main advantages of the new Hybrid model. The K-NN was implemented as the first classification method, as it was simple, robust against a noisy data set and effective in large data sets with zero training cost. The confusion problem related to similar characters in the license plates was overcome by using the multiple SVMs classification model. The SVMs were improved the performance of the K-NN in the recognition of similar characters. The SVMs was trained and tested only for the similar characters, thus, the training cost of the SVMs decreased significantly. Comparison results between the current study experimental results of a similar study23revealed that that the presented hybrid KNN-SVM model improved the character recognition rate significantly from 94% to 97.03% for all cases tested.

3. PROPOSED SYSTEM

3.1 Overview

Figure 4.1 shows the proposed block diagram, which is used to identify the person, bike rider with having helmet. The YOLO (You Only Look Once) model is a popular real-time object detection algorithm used for identifying objects in images or videos. For person identification using the YOLO model, the algorithm is trained on a dataset that contains labeled images of people. The YOLO model works by dividing the input image into a grid of cells and predicting a bounding box and class probabilities for each cell. To identify a person, the YOLO model uses a pre-trained convolutional neural network (CNN) to extract features from the image. The extracted features are then used to predict the bounding box and class probabilities for each cell. Once the YOLO model predicts the bounding box and class probabilities for each cell, a post-processing step is performed to merge overlapping bounding boxes and remove false positives. The final result is a list of bounding boxes and class probabilities for each person in the input image.

Step 1. Test Image: Start by obtaining the image to be analyzed. The image can be sourced from a security camera, a photograph, or any other relevant source.

Step 2. Image Preprocessing: Before running YOLOv3 for person and helmet detection, conduct preprocessing on the image to enhance detection accuracy. Common preprocessing steps encompass resizing the image to the required input size for YOLOv3, normalizing pixel values, and handling color channels (e.g., converting from BGR to RGB).

Step 3. Person Detection using YOLOv3: Employ a pre-trained YOLOv3 model for object detection. Pre-trained weights and configurations for YOLOv3 can be sourced from resources such as the Darknet website or other deep learning libraries. Load the YOLOv3 model into the working environment. Input the preprocessed image into the YOLOv3 model. Extract the detection results, including bounding boxes around detected persons, their confidence scores, and class labels (e.g., "person"). Apply filtering to retain only high-confidence bounding boxes on test image.

Step 4. Helmet Detection using YOLOv3: Analogously, use a pre-trained YOLOv3 model specialized in helmet detection. This model should be trained specifically to recognize helmets. Load the helmet detection YOLOv3 model into the working environment. Input the same preprocessed image into the helmet detection

YOLOv3 model. Extract the detection results, which encompass bounding boxes around detected helmets, their confidence scores, and class labels (e.g., "helmet").

Step 5. Output and Analysis: Combine the person and helmet detection results to identify individuals wearing helmets. Visualize the results by drawing bounding boxes around persons and helmets in the original image for clarity and interpretation.

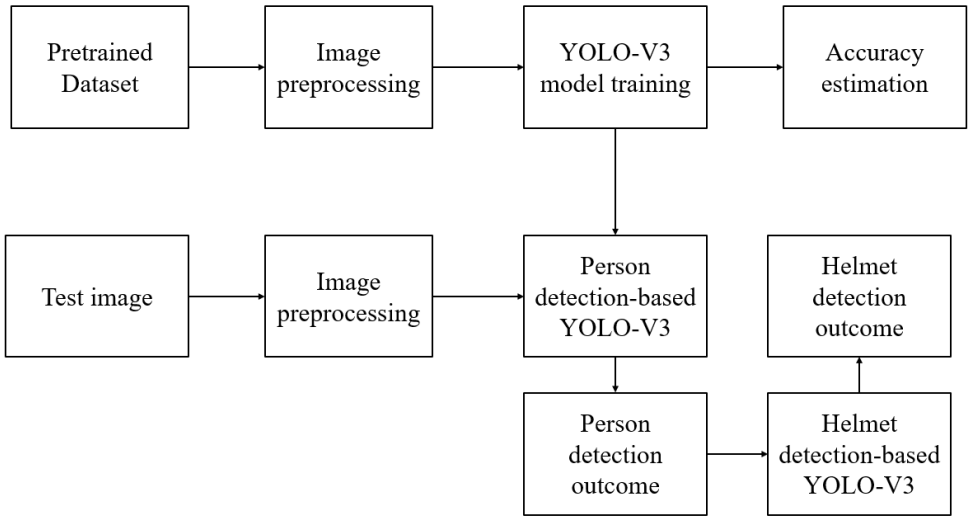


Figure 4.1. Proposed block diagram.

3.2 Image preprocessing

Image preprocessing is a critical step in computer vision and image analysis tasks. It involves a series of operations to prepare raw images for further processing by algorithms or neural networks. Here's an explanation of each step-in image preprocessing:

Step 0. Image Read: The first step in image preprocessing is reading the raw image from a source, typically a file on disk. Images can be in various formats, such as JPEG, PNG, BMP, or others. Image reading is performed using libraries or functions specific to the chosen programming environment or framework. The result of this step is a digital representation of the image that can be manipulated programmatically.

1. Image Resize: Image resize is a common preprocessing step, especially when working with machine learning models or deep neural networks. It involves changing the dimensions (width and height) of the image. Resizing can be necessary for several reasons:

- Ensuring uniform input size: Many machine learning models, especially convolutional neural networks (CNNs), require input images to have the same dimensions. Resizing allows you to standardize input sizes.
- Reducing computational complexity: Smaller images require fewer computations, which can be beneficial for faster training and inference.
- Managing memory constraints: In some cases, images need to be resized to fit within available memory constraints.

When resizing, it's essential to maintain the aspect ratio to prevent image distortion. Typically, libraries like OpenCV or Pillow provide convenient functions for resizing images.

2. Image to Array: In this step, the image is converted into a numerical representation in the form of a multidimensional array or tensor. Each pixel in the image corresponds to a value in the array. The array is usually structured with dimensions representing height, width, and color channels (if applicable).

For grayscale images, the array is 2D, with each element representing the intensity of a pixel. For color images, it's a 3D or 4D array, with dimensions for height, width, color channels (e.g., Red, Green, Blue), and potentially batch size (if processing multiple images simultaneously).

The conversion from an image to an array allows for numerical manipulation and analysis, making it compatible with various data processing libraries and deep learning frameworks like NumPy or TensorFlow.

3. Image to Float32: Most machine learning and computer vision algorithms expect input data to be in a specific data type, often 32-bit floating-point numbers (float32). Converting the image array to float32 ensures that the pixel values can represent a wide range of intensities between 0.0 (black) and 1.0 (white) or sometimes between -1.0 and 1.0, depending on the specific normalization used.

This step is essential for maintaining consistency in data types and enabling compatibility with various machine learning frameworks and libraries. It's typically performed by dividing the pixel values by the maximum intensity value (e.g., 255 for an 8-bit image) to scale them to the [0.0, 1.0] range.

4. Image to Binary: Image binarization is a process of converting a grayscale image into a binary image, where each pixel is represented by either 0 (black) or 1 (white) based on a specified threshold. Binarization is commonly used for tasks like image segmentation, where you want to separate objects from the background.

The process involves setting a threshold value, and then for each pixel in the grayscale image, if the pixel value is greater than or equal to the threshold, it is set to 1; otherwise, it is set to 0.

Binarization simplifies the image and reduces it to essential information, which can be particularly useful in applications like character recognition or object tracking, where you need to isolate regions of interest.

4.3 Dataset Splitting

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model. Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models. If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

Training Set: A subset of dataset to train the machine learning model, and we already know the output.

Test set: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output..

3.4 YOLO-V3 Model

Object detection is a phenomenon in computer vision that involves the detection of various objects in digital images or videos. Some of the objects detected include people, cars, chairs, stones, buildings, and animals. This phenomenon seeks to answer two basic questions:

1. What is the object? This question seeks to identify the object in a specific image.
2. Where is it? This question seeks to establish the exact location of the object within the image.

Object detection consists of various approaches such as fast R-CNN, Retina-Net, and Single-Shot MultiBox Detector (SSD). Although these approaches have solved the challenges of data limitation and modeling in object detection, they are not able to detect objects in a single algorithm run. YOLO algorithm has gained popularity because of its superior performance over the aforementioned object detection techniques.

YOLO Definition: YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time). Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects. This means that prediction in the entire image is done in a single algorithm run. CNN is used to predict various class probabilities and bounding boxes simultaneously. The YOLO algorithm consists of various variants. Some of the common ones include tiny YOLO and YOLOv3.

Importance of YOLO: YOLO algorithm is important because of the following reasons:

- Speed: This algorithm improves the speed of detection because it can predict objects in real-time.
- High accuracy: YOLO is a predictive technique that provides accurate results with minimal background errors.
- Learning capabilities: The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

YOLO algorithm working: YOLO algorithm works using the following three techniques:

- Residual blocks
- Bounding box regression
- Intersection Over Union (IOU)

Residual blocks: First, the image is divided into various grids. Each grid has a dimension of S x S. The following Figure 2 shows how an input image is divided into grids. In the Figure 2, there are many grid cells of equal dimension. Every grid cell will detect objects that appear within them. For example, if an object center appears within a certain grid cell, then this cell will be responsible for detecting it.

Figure 4.2. Example of residual blocks.

Bounding box regression: A bounding box is an outline that highlights an object in an image. Every bounding box in the image consists of the following attributes:

- Width (bw)
- Height (bh)
- Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.
- Bounding box center (bx,by)

The following Figure 3 shows an example of a bounding box. The bounding box has been represented by a yellow outline. YOLO uses a single bounding box regression to predict the height, width, center, and class of objects. In the image above, represents the probability of an object appearing in the bounding box.

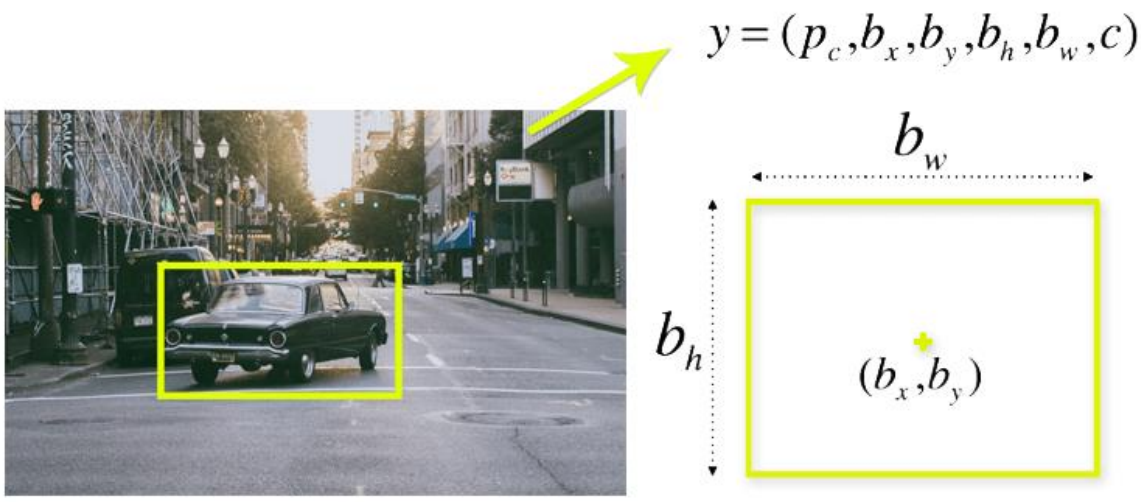


Figure 4.3. Bounding box regression

Intersection over union (IOU): Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly. Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.

Combination of the three techniques: The following image shows how the three techniques are applied to produce the final detection results.

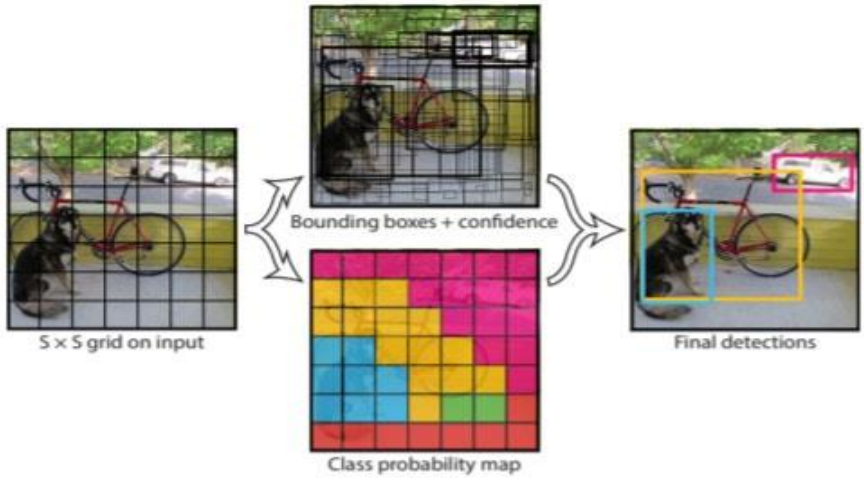


Figure 3.4. Combination of three modules.

First, the image is divided into grid cells. Each grid cell forecasts B bounding boxes and provides their confidence scores. The cells predict the class probabilities to establish the class of each object. For example, we can notice at least three classes of objects: a car, a dog, and a bicycle. All the predictions are made simultaneously using a single convolutional neural network. Intersection over union ensures that the predicted bounding boxes are equal to the real boxes of the objects. This phenomenon eliminates

unnecessary bounding boxes that do not meet the characteristics of the objects (like height and width). The final detection will consist of unique bounding boxes that fit the objects perfectly. For example, the car is surrounded by the pink bounding box while the bicycle is surrounded by the yellow bounding box. The dog has been highlighted using the blue bounding box.

The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown below.

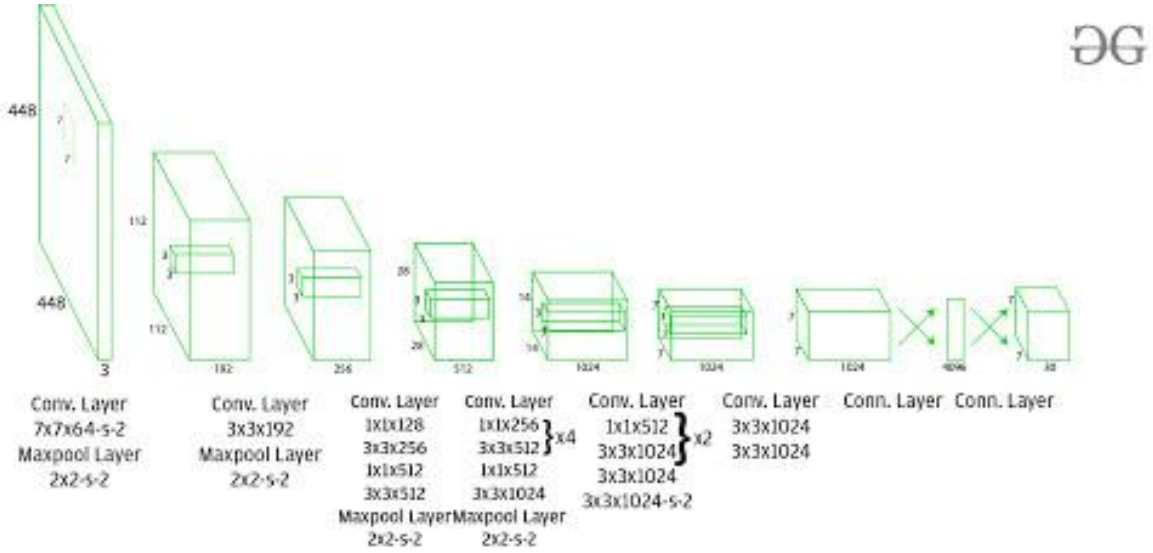


Figure 3.5. YOLO Layers.

The first 20 convolution layers of the model are pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to perform detection since previous research showcased that adding convolution and connected layers to a pre-trained network improves performance. YOLO's final fully connected layer predicts both class probabilities and bounding box coordinates.

YOLO divides an input image into an S × 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.

YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.

One key technique used in the YOLO models is **non-maximum suppression (NMS)**. NMS is a post-processing step that is used to improve the accuracy and efficiency of object detection. In object detection, it is common for multiple bounding boxes to be generated for a single object in an image. These bounding boxes may overlap or be located at different positions, but they all represent the same object. NMS is used

to identify and remove redundant or incorrect bounding boxes and to output a single bounding box for each object in the image.

3.5 CNN Layers

According to the facts, training and testing of CNN involves in allowing every source data via a succession of convolution layers by a kernel or filter, rectified linear unit (ReLU), max pooling, fully connected layer and utilize SoftMax layer with classification layer to categorize the objects with probabilistic values ranging from.

According to the facts, training and testing of -CNN involves in allowing every source image via a succession of convolution layers by a kernel or filter, rectified linear unit (ReLU), max pooling, fully connected layer and utilize SoftMax layer with classification layer to categorize the objects with probabilistic values ranging from [0,1]. Convolution layer as depicted in Figure 8 is the primary layer to extract the features from a source image and maintains the relationship between pixels by learning the features of image by employing tiny blocks of source data. It's a mathematical function which considers two inputs like source image $I(x, y, d)$ where x and y denotes the spatial coordinates i.e., number of rows and columns. d is denoted as dimension of an image (here $d = 3$, since the source image and a filter or kernel with similar size of input image and can be denoted as $F(k_x, k_y, d)$.

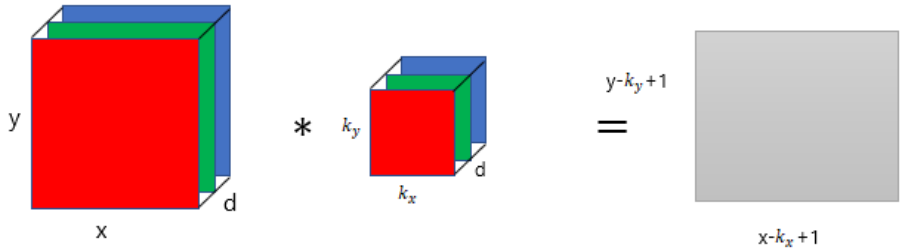


Figure 8: Representation of convolution layer process.

The output obtained from convolution process of input image and filter has a size of $C\left((x - k_x + 1), (y - k_y + 1), 1\right)$, which is referred as feature map. An example of convolution procedure is demonstrated in Figure 5.2. Let us assume an input image with a size of 5×5 and the filter having the size of 3×3. The feature map of input image is obtained by multiplying the input image values with the filter values as given in Figure 9.

1	1	1	0	0
0	0	1	1	1
1	1	0	0	1
0	0	0	1	1
1	1	1	0	0

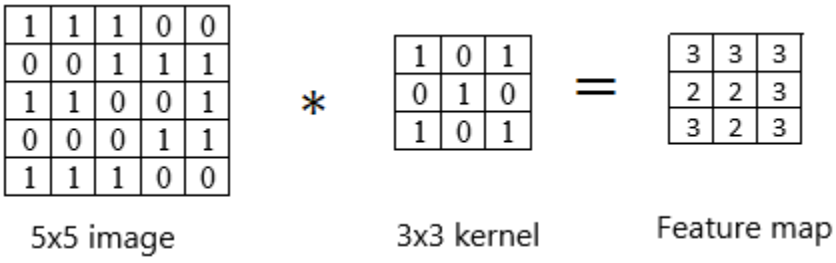
5x5 image

*

1	0	1
0	1	0
1	0	1

3x3 kernel

(a)



(b)

Figure 9: Example of convolution layer process (a) an image with size $\mathbf{5 \times 5}$ is convolving with $\mathbf{3 \times 3}$ kernel (b) Convolved feature map.

3.5.1 ReLU layer

Networks those utilizes the rectifier operation for the hidden layers are cited as rectified linear unit (ReLU). This ReLU function $\mathcal{G}(\cdot)$ is a simple computation that returns the value given as input directly if the value of input is greater than zero else returns zero. This can be represented as mathematically using the function $max(\cdot)$ over the set of 0 and the input x as follows:

$$\mathcal{G}(x) = \max\{0, x\}$$

3.5.2 Max pooing layer

This layer mitigates the number of parameters when there are larger size images. This can be called as subsampling or down sampling that mitigates the dimensionality of every feature map by preserving the important information. Max pooling considers the maximum element form the rectified feature map.

3.5.3 SoftMax classifier

Generally, as seen in the above picture SoftMax function is added at the end of the output since it is the place where the nodes are meet finally and thus, they can be classified. Here, X is the input of all the models and the layers between X and Y are the hidden layers and the data is passed from X to all the layers and Received by Y. Suppose, we have 10 classes, and we predict for which class the given input belongs to. So, for this what we do is allot each class with a particular predicted output. Which means that we have 10 outputs corresponding to 10 different class and predict the class by the highest probability it has.

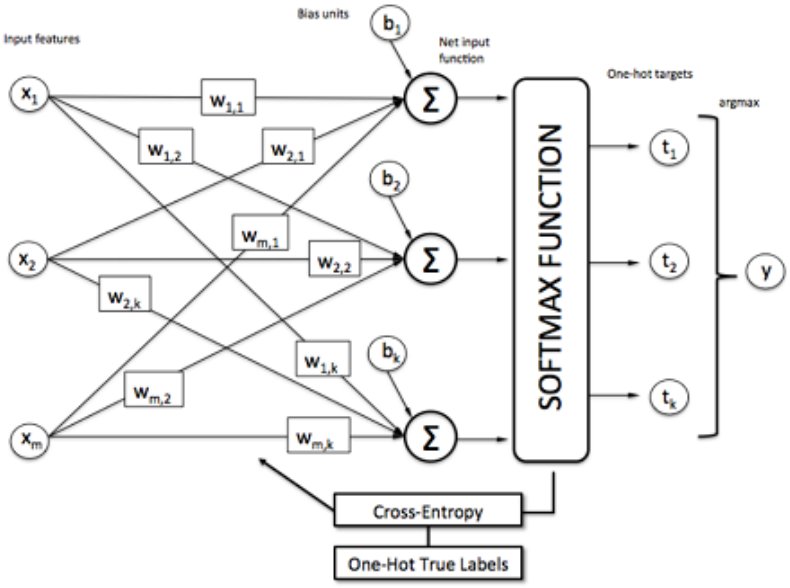


Figure 10: Emotion prediction using SoftMax classifier.

In Figure 11, and we must predict what is the object that is present in the picture. In the normal case, we predict whether the emotion is A. But in this case, we must predict what is the object that is present in the picture. This is the place where softmax comes in handy. As the model is already trained on some data. So, as soon as the picture is given, the model processes the pictures, send it to the hidden layers and then finally send to softmax for classifying the picture. The softmax uses a One-Hot encoding Technique to calculate the cross-entropy loss and get the max. One-Hot Encoding is the technique that is used to categorize the data. In the previous example, if softmax predicts that the object is class A then the One-Hot Encoding for:

Class A will be [1 0 0]

Class B will be [0 1 0]

Class C will be [0 0 1]

From the diagram, we see that the predictions are occurred. But generally, we don't know the predictions. But the machine must choose the correct predicted object. So, for machine to identify an object correctly, it uses a function called cross-entropy function. So, we choose more similar value by using the below cross-entropy formula.

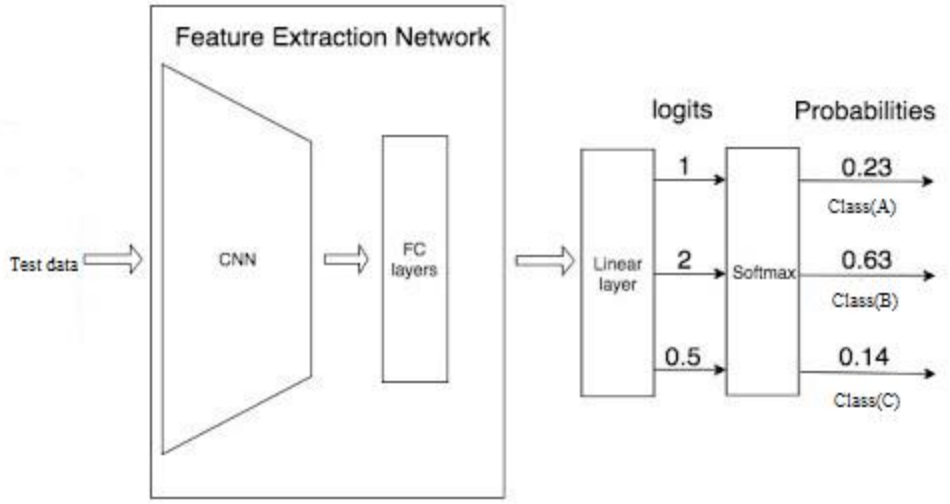


Figure 11: Example of SoftMax classifier.

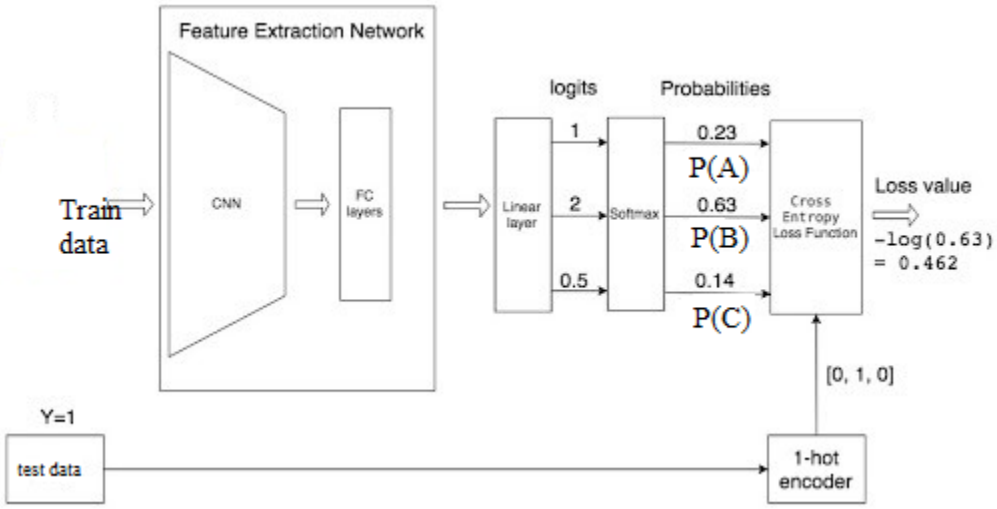


Figure 12: Example of SoftMax classifier with test data.

In the above example we see that 0.462 is the loss of the function for class specific classifier. In the same way, we find loss for remaining classifiers. The lowest the loss function, the better the prediction is. The mathematical representation for loss function can be represented as: -

$$LOSS = np.sum(-Y * np.log(Y\_pred))$$

3.6 Advantages

The procedure outlined above for image testing, preprocessing, person detection, and helmet detection using YOLOv3 offers several distinct advantages in various applications. Firstly, it provides a robust and efficient approach to object detection in images. YOLOv3, known for its real-time capabilities, enables the simultaneous detection of multiple objects within an image. This efficiency is particularly beneficial in scenarios where quick responses are critical, such as security surveillance, industrial safety, or traffic management.

Secondly, the image preprocessing step enhances the quality and consistency of the input data, leading to improved detection accuracy. Resizing the image to a standardized format ensures that YOLOv3 can work consistently across different image sizes. Normalizing pixel values and color channel adjustments help in reducing noise and ensuring that the model generalizes well to various lighting conditions.

Furthermore, the procedure's modularity allows for flexibility in customizing the object detection task. By using pre-trained models, users can easily adapt the system to their specific needs, whether it's person detection, helmet detection, or other object recognition tasks. This adaptability extends to different types of images, making it suitable for diverse applications. Another advantage lies in the post-processing step, where the results from person and helmet detection can be integrated and analyzed together. This integration facilitates the identification of safety violations, such as individuals not wearing helmets in a hazardous environment. Such insights are valuable for safety monitoring and compliance enforcement in industries like construction, manufacturing, or sports.

Additionally, the ability to generate reports and trigger alerts based on the detection results adds a layer of automation and decision support. In security systems, for instance, immediate alerts can be sent to personnel when unauthorized persons are detected, enhancing the overall security posture.

4.RESULTS AND DISCUSSION

4.1. IMPLEMENTATION DESCRIPTION

This Project is a Tkinter-based graphical user interface for an Artificial Intelligence-based Helmet Detection system. The application allows users to upload images or videos, performs object detection using YOLOv3 for motorbikes and persons, and a custom-trained Keras model for helmet detection. The GUI features buttons for image uploading, motorbike and person detection, helmet detection, and exiting. The detected objects and results are displayed on the interface, providing a user-friendly way to identify helmet violations in a traffic scenario. The code includes error handling and uses pre-trained models for efficient detection.

Import Libraries: The necessary libraries such as Tkinter for GUI, pandas, NumPy, OpenCV (cv2), subprocess, time, os, tkinter file dialog, and others are imported.

Loading YOLOv3 Model: The YOLO (You Only Look Once) model is loaded for detecting objects in images. The model is loaded using OpenCV's DNN module.

Loading Custom Model: Another custom model (possibly for helmet detection) is loaded using Keras. The model is a neural network loaded from a JSON file and corresponding weights.

GUI Setup: The Tkinter GUI is set up with buttons for uploading an image, detecting motorbikes and persons, detecting helmets, and exiting the application.

Image and Video Processing Functions:

`upload`: Allows the user to upload an image.

`detectBike`: Calls a function to detect motorbikes and persons in the uploaded image.

`detectHelmet`: Calls a function to detect helmets in the uploaded image using YOLOv3.

`videoHelmetDetect`: Allows the user to upload a video and detects helmets in the video frames.

Model Loading Functions:

`loadLibraries`: Loads the YOLOv3 model for helmet detection.

Object Detection Functions:

`getOutputsNames`: Retrieves the names of the output layers from the YOLOv3 model.

`drawPred`: Draws bounding boxes and labels on detected objects (helmets) in the image.

`postprocess`: Post-processes the YOLOv3 output to draw bounding boxes and labels for detected helmets.

Other Functions:

`exit`: Closes the Tkinter application.

Tkinter GUI Components:

Labels and buttons for various functionalities.

A text area to display information or results.

Execution:

The main loop (`main.mainloop()`) runs the Tkinter application.

10.2 RESULTS AND DESCRIPTION

The implemented AI-based helmet detection system utilizes the power of YOLO, a state-of-the-art object detection algorithm, to achieve robust results. The system begins by loading the YOLO model with pre-trained weights and configuration, followed by the capture and preprocessing of input data.

1. Accurate Helmet Detection:

The YOLO-based helmet detection system should accurately identify and localize helmets within the given images or video streams.

2. Bounding Box Visualization:

Bounding boxes drawn around detected helmets on the processed images or video frames, indicating the regions where helmets are present.

3. Real-time Performance:

Leveraging the real-time capabilities of YOLO, the system should be able to process video streams efficiently and provide prompt results.

4. Integration with Traffic Challan System:

Successful integration with the traffic challan system, enabling the automatic issuance of fines for individuals not wearing helmets.

5. User Interface (Optional):

A user-friendly interface for administrators or law enforcement personnel to monitor the system, review detections, and manage fine issuance.

1. uploading an image to detect whether the person is wearing a helmet or not.

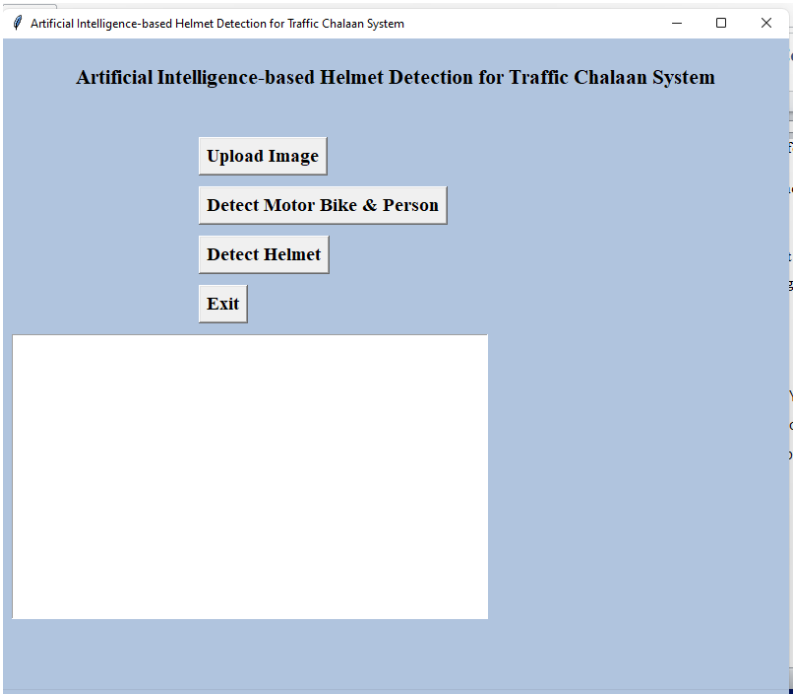


Figure-1: Uploading an image

2. Select the imge and give it as input to detect the person and motor bike. It divide the input image into a grid of cells and, for each cell, predict the probability of the presence of an object and the bounding box coordinates of the object.

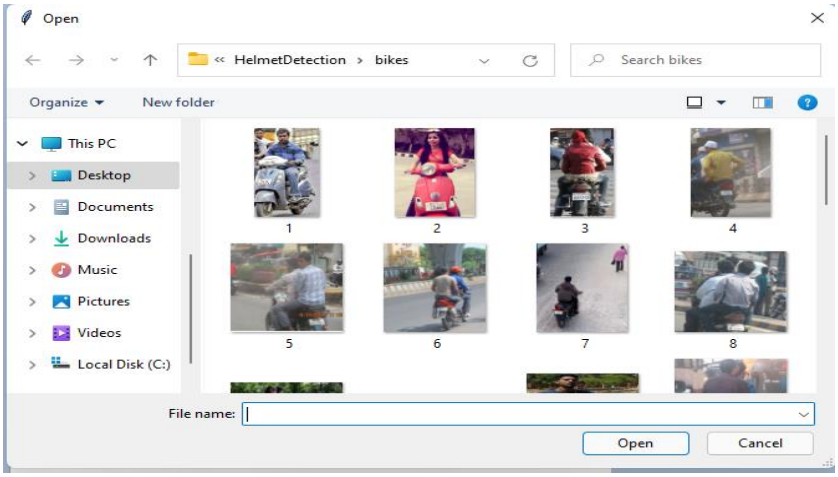


Figure 2: Selecting an image

3. After uploading an image click on Detect motor bike & person to identify the bike and the person in the image. Input image is passed through a CNN to extract features from the image. The features

are then passed through a series of fully connected layers, which predict class probabilities and bounding box coordinates.

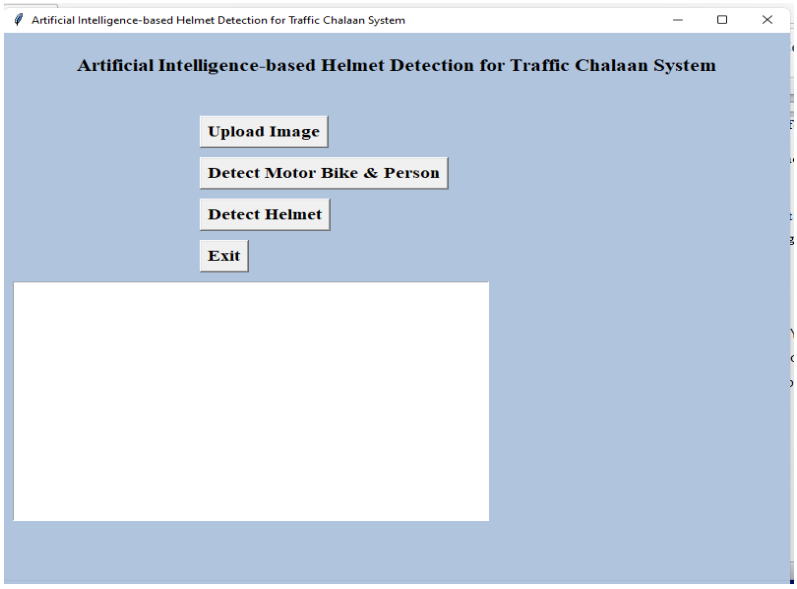


Figure 3: Detecting motor bike and person

4. By using object detection model YOLO -it detects the bike and the person in the image. The output of the network is a set of bounding boxes and class probabilities for each cell. The bounding boxes are then filtered using a post-processing algorithm called non-max suppression to remove overlapping boxes and choose the box with the highest probability.

Figure 4: Detected person and motor bike

5. Click on Detect Helmet to detect whether the person is wearing a helmet or not.

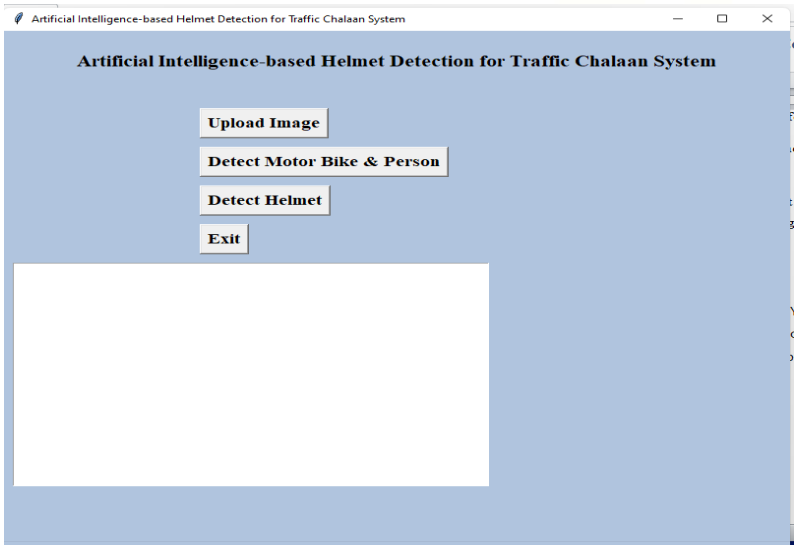


Figure-5: Detecting helmet

6. After detecting the helmet, if the person is not wearing a helmet it will display on image as Helmet not detected. The final output is a set of predicted bounding boxes and class labels for each object in the image.

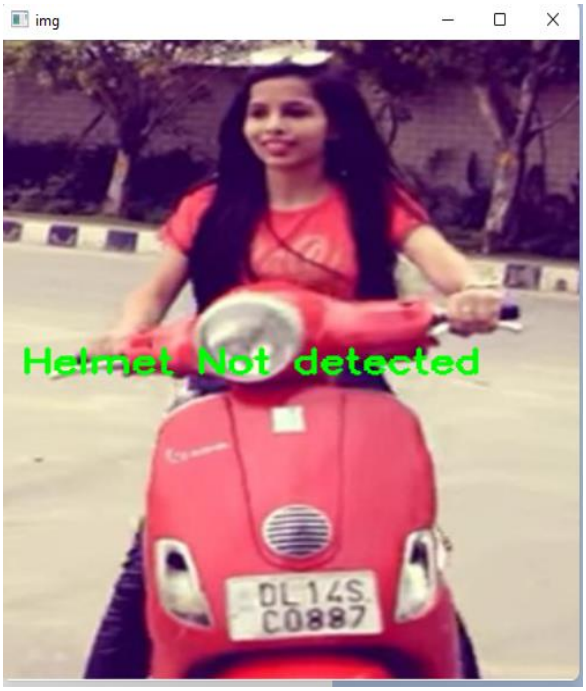


Figure-7: Person without helmet

5.CONCLUSION

In conclusion, the results indicate that the proposed YOLOV3 significantly outperforms the existing SVM and RFC methods in terms of accuracy. It achieves an impressive accuracy of 99.88% for person detection

and 99.00% for helmet detection, showcasing its superior capability in accurately identifying persons and helmets in images or video frames. These high accuracy scores suggest that the proposed method holds great promise for applications requiring precise object detection, such as safety and security systems. However, further research and analysis are necessary to fully understand the inner workings and limitations of the proposed method, and it is essential to assess its performance across a broader range of scenarios and datasets to validate its effectiveness.

REFERENCES

[1] Marzuki, P., et al. "A design of license plate recognition system using convolutional neural network." International Journal of Electrical and Computer Engineering 9.3 (2019): 2196.

[2] Ma, Lixin, and Yong Zhang. "Research on vehicle license plate recognition technology based on deep convolutional neural networks." Microprocessors and Microsystems 82 (2021): 103932.

[3] Vaiyapuri, Thavavel, et al. "Automatic vehicle license plate recognition using optimal deep learning model." Computers, Materials & Continua 67.2 (2021).

[4] Huang, Zhao-Kai, Hao-Wei Tseng, and Cheng-Lun Chen. "Application of Extreme Learning Machine to Automatic License Plate Recognition." 2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA). IEEE, 2019.

[5] Neto, Edson Cavalcanti, et al. "Brazilian vehicle identification using a new embedded plate recognition system." Measurement 70 (2019): 36-46.

[6] Halin, Alfian Abdul, et al. "License plate localization using a Naïve Bayes classifier." 2020 IEEE International Conference on Signal and Image Processing Applications.IEEE,2020.

[7] Akhtar, Zuhaib, and Rashid Ali. "Automatic number plate recognition using random forest classifier." SN Computer Science 1 (2020): 1-9.

[8] Sathiyabhama, B., et al. "Tracing of vehicle region and number plate detection using deep learning." 2020 International conference on emerging trends in information technology and engineering (ic-ETITE). IEEE, 2020.

[9] Tabrizi, Sahar S., and Nadire Cavus. "A hybrid KNN-SVM model for Iranian license plate recognition." Procedia Computer Science 102 (2019): 588-594.