ARTIFICIAL INTELLIGENCE-BASED HELMET DETECTION FOR TRAFFIC CHALLAN SYSTEM

Sai Kumar[1], Sai Preethi Maisarla[2], Uday Bhaskar[2], Manya Banala[2]

[2]UG Scholar, [1,2]Department of Computer Science and Engineering

[1,2]Kommuri Pratap Reddy Institute of Technology, Ghatkesar, Hyderabad, Telangana.

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.

Keywords: Helmet, CCTV, Motorcycle, YOLO, Object Detection, Deep Learning.

1. INTRODUCTION

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.

2. LITERATURE SURVEY

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 preprocessing 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 preprocessing 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 preprocessing, 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 preprocessing 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 preprocessing, number plate localization, character segmentation and character recognition. Preprocessing 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%. Satyabhama, 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. Singh, Jaskirat, and Bharat Bhushan (2019) [10] proposed a robust technique for License Plate Detection (LPD) in the images using deep neural networks, Pre-process the detected license plate sand perform License Plate Recognition (LPR) using LSTM Tesseract OCR Engine. According to their experimental results, they have successfully achieved robust results with LPD accuracy of 99% and LPR accuracy of 95%just like commercial ANPR systems i.e., Open-ALPR and Plate Recognizer.

3. PROPOSED SYSTEM

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.

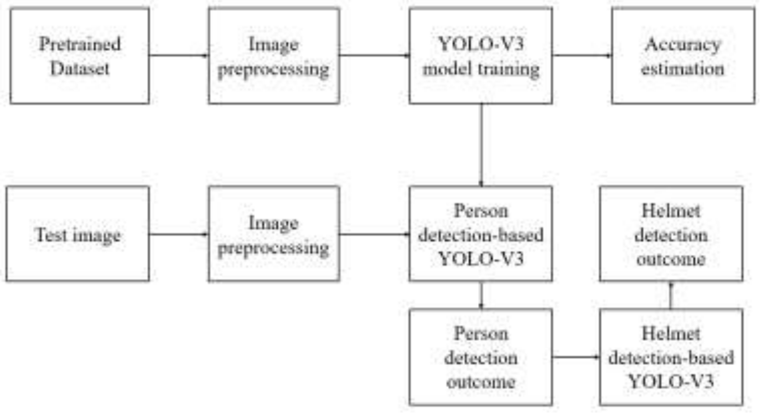

Figure 1. Proposed block diagram.

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.

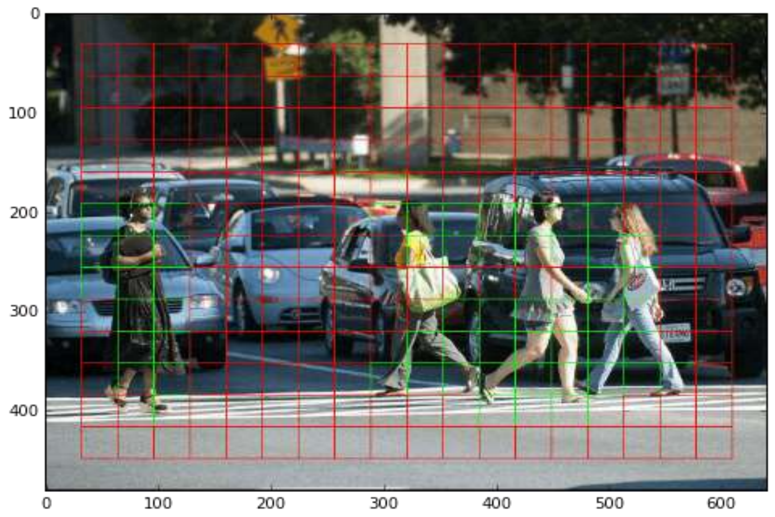

Figure 2. Example of residual blocks.

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.

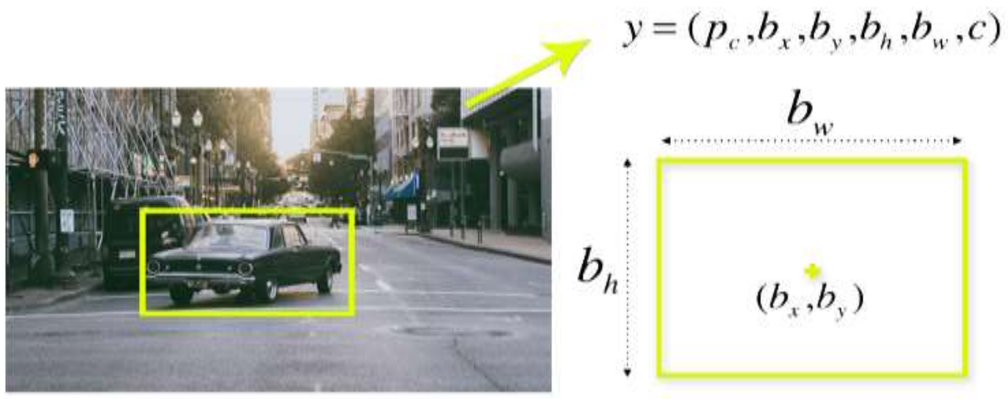

Figure 3. Bounding box regression

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.

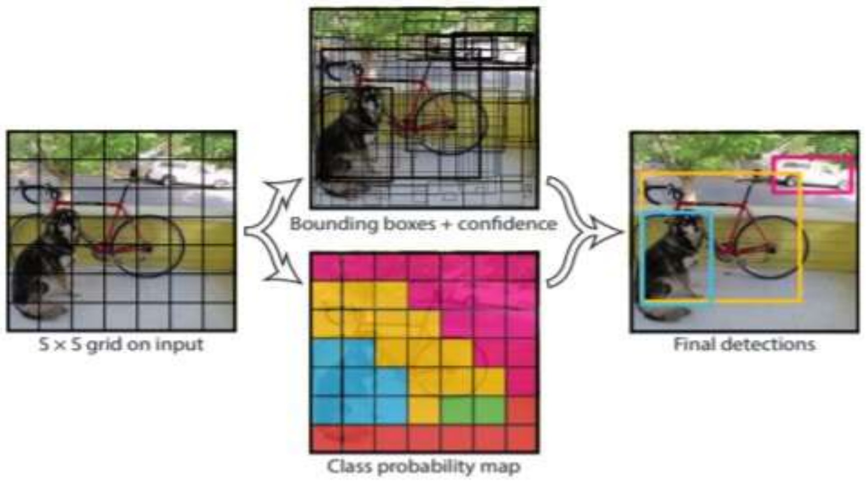

Figure 4. Combination of three modules.

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

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.

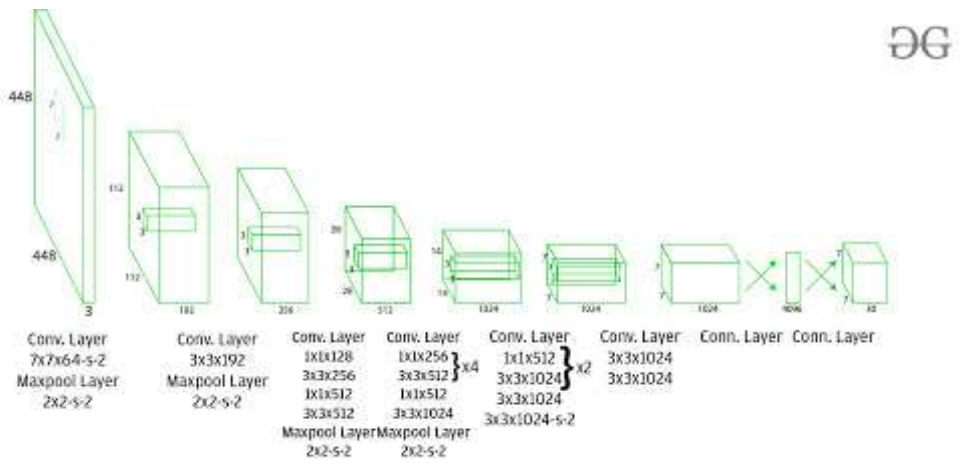

Figure 5: YOLO Layers.

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.

Convolutional Neural Network

Convolutional Neural Networks (CNNs) represent a pivotal advancement in deep learning, particularly well-suited for tasks involving structured grid-like data, such as images. Their architecture and design draw inspiration from biological processes, specifically how the visual cortex processes visual information. CNNs have revolutionized fields like computer vision, enabling breakthroughs in image classification, object detection, and segmentation, among others.

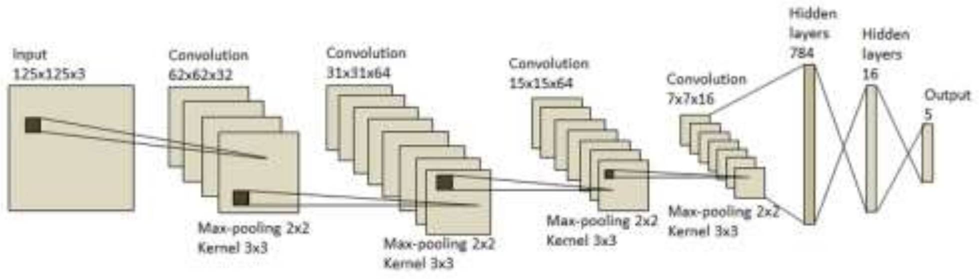

Figure 6: Architectural Diagram of CNN Model.

Components of a CNN

1. Input Layer

The input to a CNN is typically an image represented as a grid of pixel values. For a color image, this might be a 3D array (height x width x 3) where 3 represents the RGB channels.

2. Convolutional Layer

The convolutional layer applies filters (kernels) to the input image. Each filter performs convolution operation, which involves sliding the filter matrix over the input image and computing dot products to produce a feature map. Mathematically, for a 2D input image I and a filter K, the convolution operation at a specific position (i,j) is given by:

$$(I*K)(i,j) = \sum m \sum n I(m,n) \cdot K(i-m,j-n)$$

where I(m,n) represents the pixel intensity at position (m,n) in the input image.

3. Activation Function

After convolution, an activation function like ReLU (Rectified Linear Unit) is applied element-wise to introduce non-linearity into the model:

$$\mathrm{ReLU}(x)=\max(0,x)$$

ReLU helps CNNs learn complex patterns and improves convergence during training.

4. Pooling Layer

Pooling layers (e.g., max pooling or average pooling) follow convolutional layers to reduce spatial dimensions of the feature maps, thereby decreasing computational complexity and controlling overfitting.

For max pooling, the operation extracts the maximum value from each patch of the feature map:

$$\text{MaxPooling}(I)(i,j)=\text{maxm,n}I(m,n)$$

5. Flattening

After several convolutional and pooling layers, the resulting feature maps are flattened into a vector. This step converts the 2D or 3D matrix representation into a 1D vector that can be input to fully connected layers.

6. Fully Connected (Dense) Layers

These layers process the flattened features. Each neuron in a fully connected layer is connected to every neuron in the previous layer. Mathematically, for a layer l with output vector $h^{(l)}$ and weights $W^{(l)}$ the output $z^{(l+1)}$ of the next layer is computed as:

$$z^{(l+1)} = W^{(l)}h^{(l)}+b^{(l)}$$

where $b^{(l)}$ is the bias vector.

7. Output Layer

The final layer of the CNN produces the network's output. For classification tasks, this typically involves applying a softmax activation function to the output of the last fully connected layer. Softmax normalizes the outputs into probabilities across multiple classes, enabling the model to make predictions.

8. Loss Function

A loss function quantifies how well the network's predictions match the actual target values during training. For classification tasks, cross-entropy loss is commonly used.

9. Optimization

To minimize the loss function and improve the network's performance, optimization algorithms like stochastic gradient descent (SGD) or its variants (e.g., Adam, RMSProp) are employed. These algorithms adjust the weights and biases of the network iteratively based on the gradients computed during backpropagation.

10. Backpropagation

Backpropagation is a key mechanism in CNNs for computing the gradients of the loss function with respect to each parameter in the network. It propagates these gradients backward through the network, allowing efficient updates of the model's weights and biases,

4. RESULTS AND DISCUSSION

4.1 Implementation Description

Research implements 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.

4.2 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. Figure 7 Uploading the image 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 9 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 10 Shows the object detection YOLO model 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.

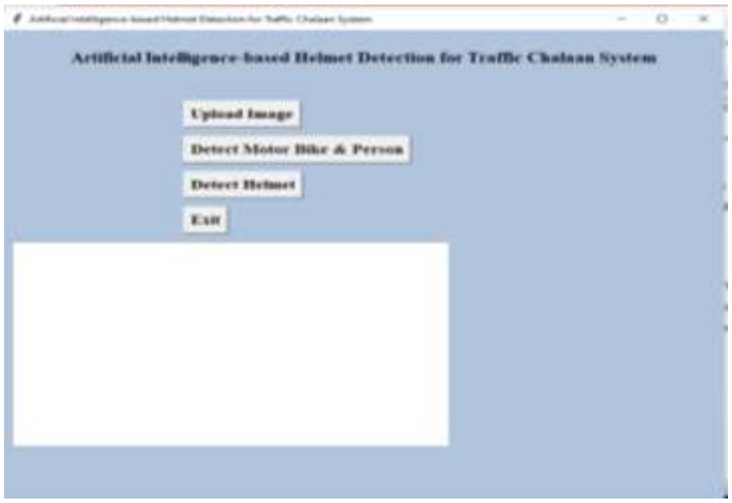
Artificial Intelligence-based Helmet Detection for Traffic Chalaan System
Upload Image
Detect Motor Bike & Person
Detect Helmet
Exit

Figure 7: Uploading an image.

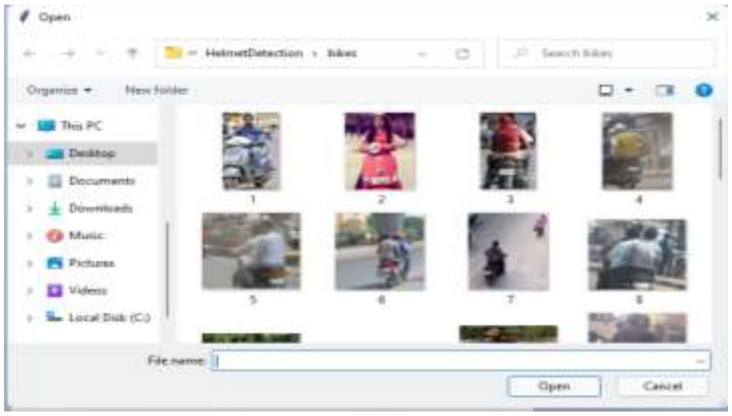

Figure 8: Selecting an image.

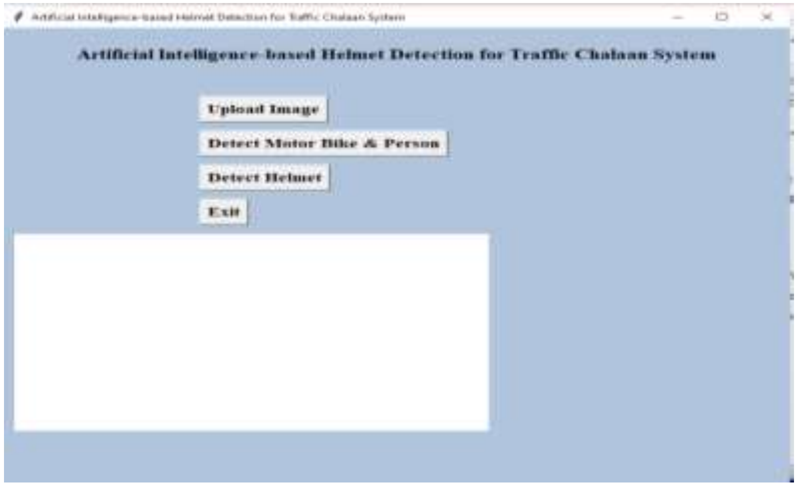
Artificial Intelligence-based Helmet Detection for Traffic Chalaan System
Upload Image
Detect Motor Bike & Person
Detect Helmet
Exit

Figure 9: Detecting motor bike and person

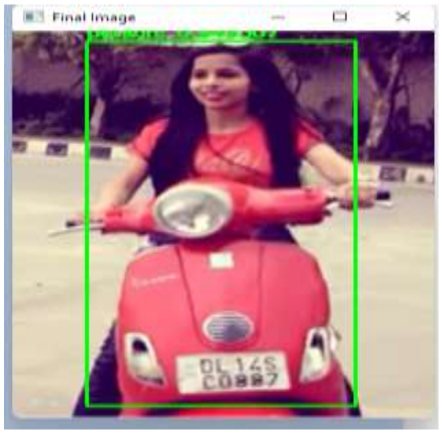

Figure 10: Detected person and motor bike

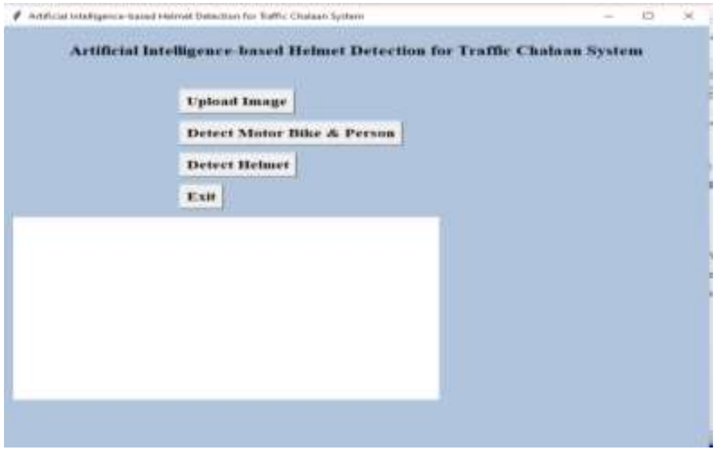
Artificial Intelligence-based Helmet Detection for Traffic Chalaan System
Upload Image
Detect Motor Bike & Person
Detect Helmet
Exit

Figure 11: Detecting helmet

Figure 12 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.

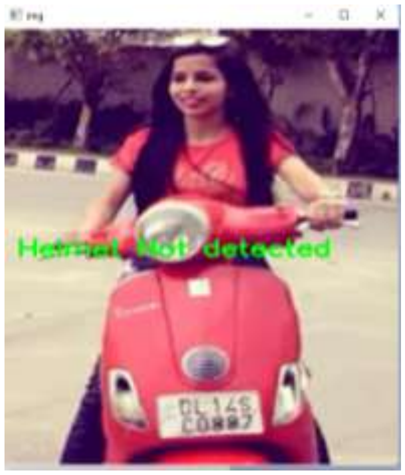

Figure 12: Person without helmet.

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.

[10] Singh, Jaskirat, and Bharat Bhushan. "Real time Indian license plate detection using deep neural networks and optical character recognition using LSTM tesseract." 2019 international conference on computing, communication, and intelligent systems (ICCCIS). IEEE, 2019.