AUTOMATIC HELMET VIOLATION DETECTION

Prof. Dr. N.K.Karthikeyan
Head of the Department,
Dept of Information Technology
Coimbatore Institute of Technology
Coimbatore, India
karthikeyan.nk@cit.edu.in

Mr. M. Luhindaar
Student, Dept of
Information Technology
Coimbatore Institute of Technology
Coimbatore, India
whil078@gmail.com

Mr. P. Pramoth Kanna
Student, Dept of
Information Technology
Coimbatore Institute of Technology
Coimbatore, India
pramoth6302@gmail.com

Ms. B. Vedha
Student, Dept of
Information Technology
Coimbatore Institute of Technology
Coimbatore, India
vedhabaskaran09@gmail.com

Abstract— In India, the number of accidents occurring each day is increasing rapidly. The two-wheelers account 25 percent of road crash deaths because of ignoring safety measures like wearing helmets while driving. A police officer cannot manage the whole traffic and look out for rule-breakers. It would be a very tough job and will need a lot of human power to cover all the areas. Our project's main objective is to identify the two-wheeler riders who failed to wear a helmet. The system implements Deep learning and video processing techniques. The system takes a video of traffic on public roads as input. The road CCTV footage is used to detect whether a rider is wearing a helmet or not, using YOLOv3 Algorithm, a Deep Learning Technology and the respective frames are taken from the video to detect riders, riding two wheelers, who are not wearing helmets. Using this helmet detection model riders without helmet can be easily detected.

Keywords— Deep Learning, Computer Vision using OpenCV, YOLOv3, EasyOCR.

I. INTRODUCTION

A. Deep Learning

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behaviour of the human brain—albeit far from matching its ability—allowing it to "learn" from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy. Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services as well as emerging technologies. Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognize, classify, and describe objects within the data. Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called visible layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made. Another process called backpropagation uses algorithms, like gradient descent, to calculate errors in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and backpropagation allow a neural network to make predictions and correct for any errors accordingly. Over time, the algorithm becomes gradually more accurate.

B. Computer Vision Using OpenCV

Computer Vision is an interdisciplinary field that deals with how computers can be made to gain a high-level understanding from digital images or videos. The idea here is to automate tasks that the human visual systems can do. So, a computer should be able to recognize objects such as that of a face of a human being or a lamppost or even a statue. OpenCV is Python's very own Computer Vision Library. OpenCV is a Python library that was designed to solve computer vision problems. OpenCV was originally developed in 1999 by Intel, but later, it was supported by Willow Garage. It supports a wide variety of programming languages such as C++, Python, Java, etc. Support for multiple platforms including Windows, Linux, and MacOS. OpenCV Python is nothing but a wrapper class for the original C++ library to be used with Python. Using this, all of the OpenCV array structures gets converted to/from NumPy arrays. This makes it easier to integrate it with other libraries that use NumPy. For example, libraries such as SciPy and Matplotlib.

C. YOLOv

YOLO is a clever convolutional neural network (CNN) for doing object detection in real-time. The algorithm applies a single neural network to the full image, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. YOLO is popular because it achieves high accuracy while also being able to run in real-time. The algorithm "only looks once" at the image in the sense that it requires only one forward propagation pass through the neural network to make predictions. After non-max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes. With YOLO, a single CNN simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance.

II. LITERATURE SURVEY

A. *DETECTION OF TWO-WHEELERS HELMET USING MACHINE LEARNING:*

In this paper, the YOLOv5 based Helmet identification and also studied CNN. The Jupyter notebook is used to execute the python program and hence successfully compiled the program. The implemented architecture of helmet detection has been tested and implemented multiple times and verified the accuracy every time. The project can be combined with the road traffic cameras and with some modifications of software and if possible, developing an app that can be implemented to recognize helmets automatically in the real-time system. Furthermore, automatic license plate identification and make software that generates challans for not wearing helmets can be implemented.

B. *DETECTING MOTORCYCLE HELMET USE WITH DEEP LEARNING:*

In this paper, the Automated helmet use detection for motorcycle riders is a promising approach to efficiently collect large, up-to-date data on this crucial measure. When trained, the algorithm presented in this paper can be directly implemented in existing road traffic surveillance infrastructure to produce real-time helmet use data. The evaluation of the algorithm confirms a high accuracy of helmet use data, that only deviates by a small margin from comparable data collected by human observers. Observation site-specific training of the algorithm does not involve extensive data annotation. While the sole collection of data does not increase road safety by itself, it is a prerequisite for targeted enforcement and education campaigns, which can lower the rate of injuries and fatalities. The automated helmet use detection can solve the challenges of costly and time-consuming data collection by human observers.

C. *HELMET DETECTION AND LICENSE PLATE RECOGNITION:*

In this project, A Non-Helmet Rider Detection system is developed where a video file is taken as input. If the motorcycle rider in the video footage is not wearing a helmet while riding the motorcycle, then here we are uploading an image to identify the license plate number of that motorcycle is extracted from the image and displayed. Object detection principle with YOLO architecture is used for motorcycle, person, helmet, and license plate detection. OCR is used for license plate number extraction if the rider is not wearing a helmet. Not only the characters are extracted, but also the frame from which it is also extracted so that it can be used for other purposes. All the objectives of the project are achieved satisfactorily.

D. *AN IMPROVED HELMET DETECTION METHOD FOR YOLOV5 ON AN UNBALANCED DATASET:*

In this paper, the direct use of YOLOv5 can no longer adapt to the real production situation for target detection requirements. For the unbalanced helmet dataset, the data augmentation processes the data and then train the model with the YOLOv5 algorithm, without affecting recognition speed, and finally get a 0.01-0.02 increase in the confidence level. The image detection and classification problems similar to helmet detection can draw on the results in this thesis to complete the further optimization of the recognition effect, which is of great significance for the rapid image detection and recognition of actual production sites.

E. *SAFETY HELMET WEARING DETECTION BASED ON IMAGE PROCESSING AND MACHINE LEARNING*

This paper proposed an innovative and practical safety helmet-wearing detection method based on image processing and machine learning. At first, the ViBe background modelling algorithm is exploited to detect moving objects under the view of fix surveillant camera in a power substation. After obtaining the motion region of interest, the Histogram of Oriented Gradient (HOG) feature is extracted to describe the Inner human. And then, based on the result of HOG feature extraction, the Support Vector Machine (SVM) is trained to classify pedestrians. Finally, the safety helmet detection will be implemented by colour feature recognition. Compelling experimental results demonstrated the correctness and effectiveness of this proposed method.

Fig. 1. Literature Survey

III. SYSTEM ARCHITECTURE

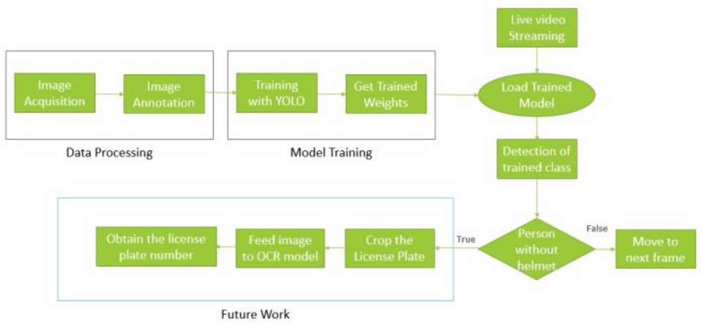


Fig. 1. Architecture Diagram

A. *Dataset*

The Dataset comes from a popular Kaggle contributor which is utilized for research taken by researchers and assistants using mobile phones. All the images are captured in different backgrounds, different lighting conditions to keep the diversity in the dataset alive. In addition to this dataset, we have included images taken from real time traffic videos. Labels of the objects are basically divided into 2 classes – with_helmet and without_helmet.

Following is a summary of information about the data from Kaggle Dataset.

Class	Number of Images
Without Helmet	250
With Helmet	200

In addition to these images, around 500 images are added from our side. Post Data Augmentation around 1000 images were generated.

Following is a sample of images from Kaggle Dataset.

_b879ee94-9ccf-11e6-a83e-3795272c0142
01HYMRM09-Helme+HY02HELMETTRAFFIC_C
1
1helmet11-500x330
02_10_0
4
5
8-2
9k= (15)
10
10bikes2
12
12-1452583178-helmet-compulsory--wearing-rule-coming-effect-...
14
20
21
22
24HELMET02
25
25oriHELMET_210457
27-1453878468-helmet-bike

Fig. 2 Sample Image of the dataset

Train dataset. The Training Dataset consists of a data from both the Kaggle Dataset and self-collected dataset. Train Dataset has around 1000 images without applying Data Augmentation. Each class on average has around 500 images.

Test dataset. The Test dataset consists of randomly selected image for each class. There are about 200 images for Test Dataset without Data Augmentation.

B. Methodology

The system uses OpenCV to stream live video and capture images at regular intervals. The image captured is then passed onto the image classification model. The model is either deployed in cloud platforms or saved in local computer. The model first classifies the object into 2 different classes namely with_helmet and without_helmet. System for Helmet detection consists of a method which can train the network and allows the user to predict helmet violators in the video. The user data will be pre-processed on the fly before it gets fed into the system.

C. You Only Look Once(YOLO)

Inspired by ResNet and FPN (Feature-Pyramid Network) architectures, YOLO-V3 feature extractor, called Darknet-53 (it has 52 convolutions) contains skip connections (like ResNet) and 3 prediction heads (like FPN) - each processing the image at a different spatial compression. Like its predecessor, Yolo-V3 boasts good performance over a wide range of input resolutions. In GluonCV's model zoo you can find several checkpoints: each for a different input resolution, but in fact the network parameters stored in those checkpoints are identical. Tested with input resolution 608x608 on COCO-2017 validation set, Yolo-V3 scored 37 mAP (mean Average Precision). This score is identical to GluonCV's trained version of Faster-RCNN-ResNet50, (a faster-RCNN architecture that uses ResNet-50 as its backbone) but 17 times faster. In that model zoo the only detectors fast enough to compete with Yolo-V3.

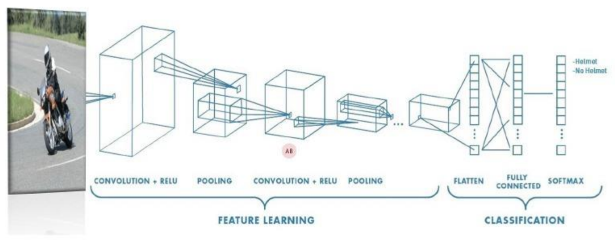


Fig. 3. CNN Architecture.

IV. SYSTEM DESIGN

A. Data Annotation

Before training the model, the training datasets need to be manually annotated with the BB information for two classes (Rider with helmet, Rider without helmet). The data annotation was done using LabelImg software [18] since YOLO needs a ground truth. The software generates starting coordinates (x0, y0) and ending coordinates (x1, y1). The coordinates were normalized between 0 and 1 to fit into YOLO format (x, y, w, h) using Eqs. (5)–(8):

$$x = (x0 + x1)\ 2 * W\ (5)$$
$$y = (y0 + y1)\ 2 * H\ (6)$$
$$w = (x1 - x0)\ W\ (7)$$
$$h = (y1 - y0)\ H(8)$$

B. Training Module

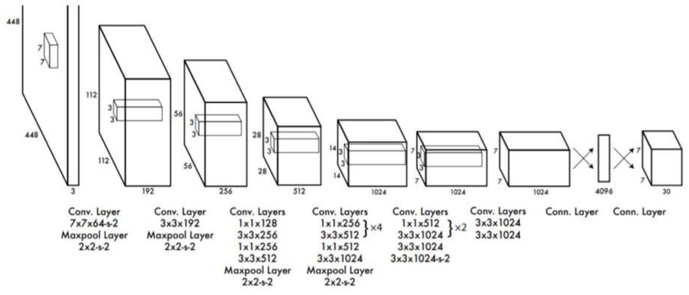
The YOLOv3 includes 19 convolutional layers and 5 max-pooling layers, followed by softmax activation functions to classify the object. The input image is divided into SxS grid cells. The grid cell, which contains the center of the object is responsible for predicting the 5-bounding box (BB) coordinates (bx , by , bw, bh, c). The coordinates (bx , by) represent the center of the object relative to the grid cell location and (bw, bh) represents the width and height of the object relative to image dimensions. The presence of object in a grid cell is given by confidence score c. The coordinates (cx, cy) represent the starting grid cell location which contains the center of a rider. The actual coordinates of the BB were normalized with relative to grid cell location using Eqs. (1)–(4):

$$bx = (bx - cx)\ cx\ (1)$$
$$by = (\ by - cy\ )\ cy\ (2)$$
$$bw = bw\ W\ (3)$$
$$bh = bh\ H\ (4)$$

where W and H are the image dimensions.

V. IMPLEMENTATION

A. YOLOv3 Architecture for the Model



448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Fig. 4. YOLOv3 Architecture.

The YOLO model is made up of three key components: the head, neck, and backbone. The backbone is the part of the network made up of convolutional layers to detect key features of an image and process them. The backbone is first trained on a classification dataset, such as ImageNet, and typically trained at a lower resolution than the final detection model, as detection requires finer details than classification. The neck uses the features from the convolution layers in the backbone with fully connected layers to make predictions on probabilities and bounding box coordinates. The head is the final output layer of the network which can be interchanged with other layers with the same input shape for transfer learning. The head is an S × S × (C + B ∗ 5) tensor and is 7 × 7 × 30 in the original YOLO research paper with a split size S of 7, 20 classes C, and 2 predicted bounding boxes B. These three portions of the model work together to first extract key visual features from the image then classify and bound them.

B. Model Training

The ImageNet 1000-class competition dataset and pre-trained 20 out of the 24 convolution layers followed by an average-pooling and fully connected layer is used. Then 4 more convolutions to the model as well as 2 fully connected layers as it has been shown that adding both convulsions and fully connected layers increases performance were added. The resolution from 244 × 244 to 448 × 448 pixels as detection requires finer details were also increased. The final layer, which predicts both class probabilities and bounding box coordinates, uses a linear activation function while the other layers use a leaky ReLU function.Data augmentation and dropout were used to prevent overfitting, with a dropout layer with a rate of 0.5, used between the first and second fully connected (preventing co-adaptation).
The loss function is the simple squared sum, but it must be modified. Without modification, the model will weight localization error, the difference between predicted and true bounding box coordinates, and class prediction error the same. Additionally, when a grid cell doesn't contain an object, its confidence score tends towards 0 which can overpower the gradients from other cells that do contain objects. Both issues are solved by using two coefficients, λcoord and λnoobj, which multiply the loss for the coordinates and the object losses respectively. These are set to λcoord = 5 and λnoobj = 0.5, increasing the weight of detection and decreasing the importance of no object loss. Finally, to weight small bounding box equality as much as large boxes, the width and height difference is square-rooted rather than used directly. This makes sure that the error is treated the same as in large and small boxes, which would otherwise discourage the model from predicting large boxes. For example, if the predicted width of the bounding box is 10 and the actual width is 8, and this equation is used

$$(w_i - \hat{w}_i)^2$$

So the loss is 4. However, a difference of 2 out of the true 98 is negligible compared to a difference of 2 out of 8. Therefore, the loss between 10 and 8 should be much larger than the loss between 100 and 98. So this equation was used instead:

$$(\sqrt{w_i} - \sqrt{\hat{w}_i})^2$$

Using this new equation, the loss for 10 and 8 is 0.111 while the loss for 100 and 98 is 0.010. So the fact that 0.111 is much smaller than 4 doesn't matter, but what does matter is that the difference between loss for the large and small widths is 0% for the squared difference while the difference is 90.99% for the squared rooted difference.
Each grid cell predicts multiple bounding boxes, but only one bounding box is responsible for detecting the object. The responsible bounding box is determined by choosing the predicted bounding box with the highest IOU.
Loss function:

$$\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}$$

The double summation merely means to sum across all of the grid cells (an S × S square) and all of the bounding boxes B.

$$\mathbb{1}_{ij}^{\text{obj}}$$

This is an identity function, set to 1 if there is an object in cell i, and bounding box j is responsible for the prediction.

$$(w_i - \hat{w}_i)^2$$

This represents the squared difference between the actual x coordinate and the predicted coordinate in cell i. This is repeated for both x and y coordinates, finding the squared difference between the overall midpoint. Finally, the identity function is `0` when there is no object or the current bounding box isn't the responsible one. The first line of the equation is the sum of squared differences between the predicted and actual midpoints of the objects in all grid cells which have an object in them and are the responsible bounding box. The second line is the sum of the squared differences between the square roots of the predicted and actual widths and heights in all grid cells which have an object in them. The third line is just the squared difference between the predicted class probabilities and the actual class probabilities in all cells that contain an object. The fourth line is the same but for all cells that don't have an object. These two lines are summed across all bounding boxes because each bounding box also predicts a confidence score in addition to coordinates. The last line: the first summation goes through every grid cell which has an object in it. Then, for that single grid cell, the squared difference between the predicted class vector and the actual vector is found.

$$pred = \begin{bmatrix} 0.1 \\ 0.6 \\ 0.3 \\ 0.1 \\ 0.4 \end{bmatrix} \quad true = \begin{bmatrix} 0.0 \\ 0.1 \\ 0.0 \\ 0.0 \\ 0.0 \end{bmatrix}$$

$$loss = (-0.1)^2 + (0.4)^2 + (-0.3)^2 + (-0.1)^2 + (-0.4)^2$$

Finally, the last line is the squared difference between the predicted and actual class for all cells that have an object. This is calculated just across grid cells and not across each bounding box as each cell predicts only a single classification regardless of the number of bounding boxes it also predicts. Finally, all of these are summed together, and the first two lines are multiplied by a coordinate coefficient to weight them more heavily and line four is multiplied by a smaller no object coefficient to weight it less.

D. Performance evaluation and metrics

a. Confusion matrix

The performance of our ensemble model was measured using a confusion matrix. Two or more types of classes were obtained as an outcome of the confusion matrix. ACTUAL and PREDICTED were the two dimensions of the confusion matrix. If the value obtained for actual and predicted class is 1, then it falls under true positives (TP). If the value obtained for actual and predicted class is 0, then it falls under the category of true negatives (TN). If the value obtained for actual and predicted class is 0 and 1 respectively, then it falls under false positives (FP). If the value obtained for actual and predicted class is 1 and 0 respectively, then it comes under false negatives (FN).

b. Accuracy

It is the most often used statistic for assessing the performance of classification algorithms. It is expressed as the number of right predictions [4]. The following formula [5] is used to compute.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)} \qquad (1)$$

For determining, accuracy of our classification model13], accuracy score function has been used.

b. Precision

The number of correct documents put back by our proposed model can be termed as precision. The formula which is used to determine precision based on a confusion matrix [5][13].

$$\text{Precision} = TP / (TP + FP) \qquad (2)$$

c. Recall

Recall is nothing but the number of positives put back by our proposed model. For the determination of recall using the confusion matrix [5][13], the formula is

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \qquad (3)$$

d. F1 score

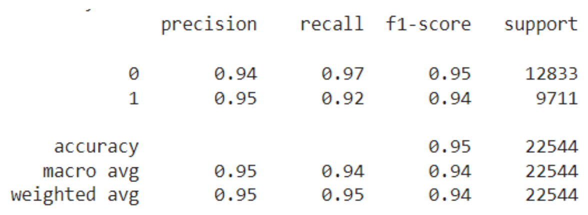
The F1 score is calculated as average of precision and recall. F1 will have a highest value of 1 and a worst value of 0. The formula used to compute F1 score is [1] [5]

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision + Recall}} \qquad (4)$$

e. Support

Number of samples of the genuine response that fall into each class of target values is termed as support [13].

E. Comparison



```
              precision    recall  f1-score   support

           0       0.94      0.97      0.95     12833
           1       0.95      0.92      0.94      9711

    accuracy                           0.95     22544
   macro avg       0.95      0.94      0.94     22544
weighted avg       0.95      0.95      0.94     22544
```

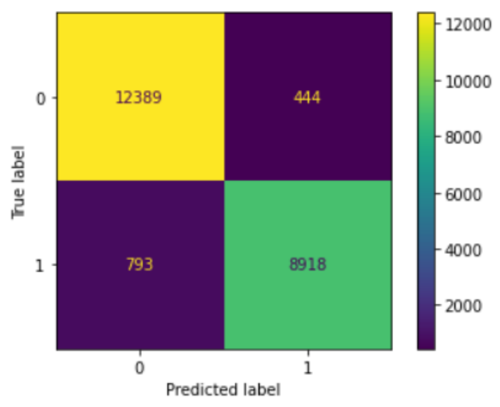


Fig. 5. Regression’s confusion matrix

F. ROC Graph

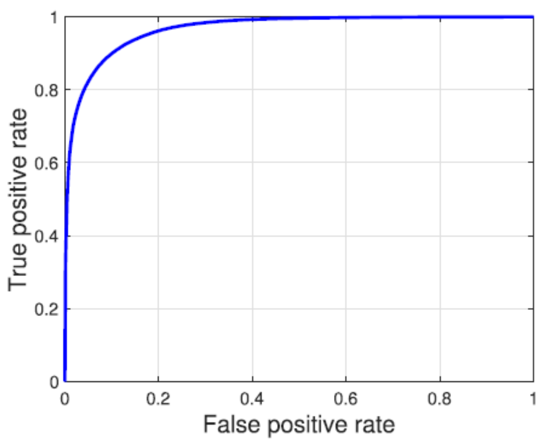


Fig. 6. ROC Graph

G. LOSS Graph

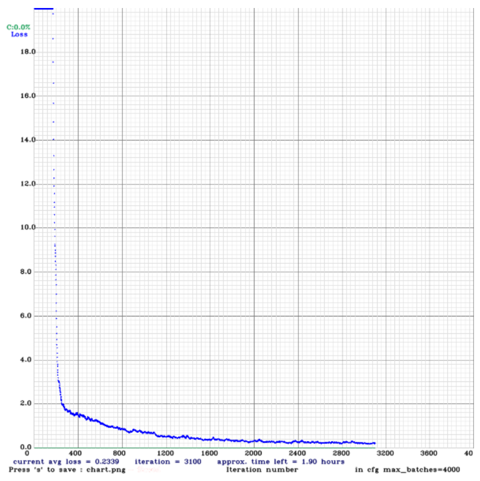


Fig. 7. LOSS Graph

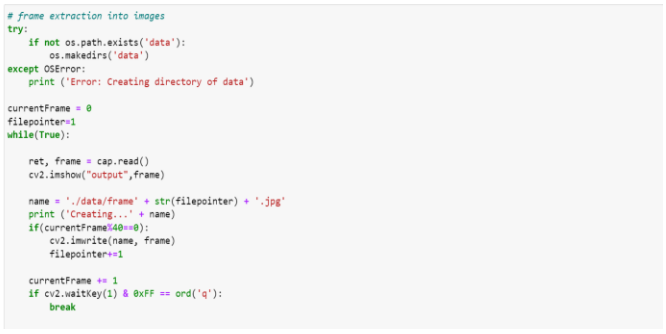
VI. CONCLUSION

The obtained value of accuracy is 82% with a better detection of the riders without helmet. From the results shown above it is evident that the YOLO object detection is well suited for real-time processing and was able to accurately classify and localize all the object classes. The proposed end-to-end model was developed successfully and has all the capabilities to be automated and deployed for monitoring. All the libraries and software used in our project are open source and hence is very flexible and cost efficient. The project was mainly built to solve the problem of non-efficient traffic management. Hence at the end of it we can say that if deployed by any traffic management departments, it would make their job easier and more efficient.

VII. FUTURE WORK

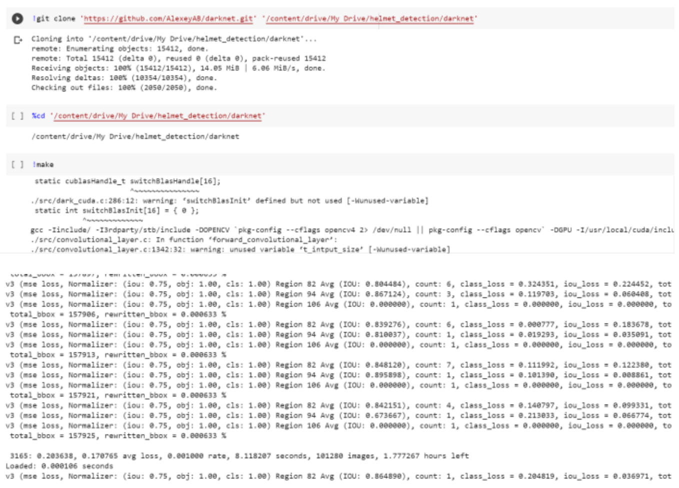
The model can further be improved with better training for the image classifier. Due to lack of resources the accuracy obtained is limited. This model can also be used in such a way that a motorbike does not start unless the rider wears a helmet. By incorporating a portable model to the ignition switch the power supply can be killed until a helmet is detected. This is one of the future developments that can be made to this model. Even though the vehicle cost will increase a bit than its standards, safety can be ensured. The system can further be automated by using an image to text classifier to identify number plates and generate a fine automatically.

VIII. RESULTS



```
# frame extraction into images
try:
    if not os.path.exists('data'):
        os.makedirs('data')
except OSError:
    print ('Error: Creating directory of data')

currentFrame = 0
filepointer=1
while(True):

    ret, frame = cap.read()
    cv2.imshow("output",frame)

    name = './data/frame' + str(filepointer) + '.jpg'
    print ('Creating...' + name)
    if(currentFrame%40==0):
        cv2.imwrite(name, frame)
        filepointer+=1

    currentFrame += 1
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

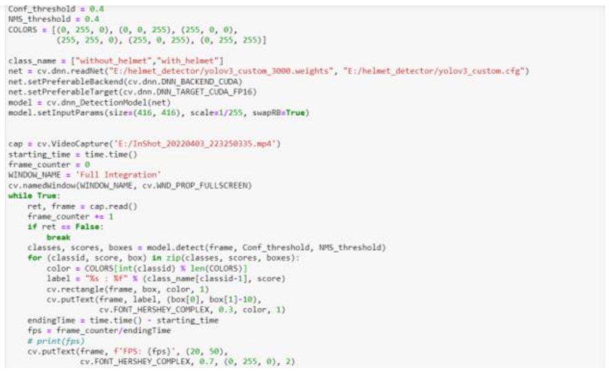
Fig. 8. Frame Extraction



```
!git clone 'https://github.com/AlexeyAB/darknet.git' '/content/drive/My Drive/helmet_detection/darknet'

Cloning into '/content/drive/My Drive/helmet_detection/darknet'...
remote: Enumerating objects: 15412, done.
remote: Total 15412 (delta 0), reused 0 (delta 0), pack-reused 15412
Receiving objects: 100% (15412/15412), 14.05 MiB | 6.06 MiB/s, done.
Resolving deltas: 100% (10354/10354), done.
Checking out files: 100% (2050/2050), done.

[ ] %cd '/content/drive/My Drive/helmet_detection/darknet'

/content/drive/My Drive/helmet_detection/darknet

[ ] !make

 static cublasHandle_t switchBlasHandle[16];
                       ^~~~~~~~~~~~~~~~
./src/dark_cuda.c:286:12: warning: 'switchBlasInit' defined but not used [-Wunused-variable]
 static int switchBlasInit[16] = { 0 };
            ^~~~~~~~~~~~~~
gcc -Iinclude/ -I3rdparty/stb/include -DOPENCV `pkg-config --cflags opencv4 2> /dev/null || pkg-config --cflags opencv` -DGPU -I/usr/local/cuda/inclu
./src/convolutional_layer.c: In function 'forward_convolutional_layer':
./src/convolutional_layer.c:1342:32: warning: unused variable 't_intput_size' [-Wunused-variable]
```

```
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.804484), count: 6, class_loss = 0.324351, iou_loss = 0.224452, tot
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.867124), count: 3, class_loss = 0.119703, iou_loss = 0.060408, tot
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, to
 total_bbox = 157906, rewritten_bbox = 0.000633 %
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.839276), count: 6, class_loss = 0.000777, iou_loss = 0.183678, tot
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.810037), count: 1, class_loss = 0.019293, iou_loss = 0.035091, tot
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, to
 total_bbox = 157913, rewritten_bbox = 0.000633 %
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.848120), count: 7, class_loss = 0.111992, iou_loss = 0.122380, tot
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.895898), count: 1, class_loss = 0.101390, iou_loss = 0.008861, tot
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, to
 total_bbox = 157921, rewritten_bbox = 0.000633 %
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.842151), count: 4, class_loss = 0.140797, iou_loss = 0.099331, tot
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.673667), count: 1, class_loss = 0.213033, iou_loss = 0.066774, tot
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, to
 total_bbox = 157925, rewritten_bbox = 0.000633 %

 3165: 0.203638, 0.170765 avg loss, 0.001000 rate, 8.118207 seconds, 101280 images, 1.777267 hours left
Loaded: 0.000106 seconds
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.864890), count: 1, class_loss = 0.204819, iou_loss = 0.036971, tot
```

Fig. 9. Model Training



```
Conf_threshold = 0.4
NMS_threshold = 0.4
COLORS = [(0, 255, 0), (0, 0, 255), (255, 0, 0),
          (255, 255, 0), (255, 0, 255), (0, 255, 255)]

class_name = ["without_helmet","with_helmet"]
net = cv.dnn.readNet("E:/helmet_detector/yolov3_custom_3000.weights", "E:/helmet_detector/yolov3_custom.cfg")
net.setPreferableBackend(cv.dnn.DNN_BACKEND_CUDA)
net.setPreferableTarget(cv.dnn.DNN_TARGET_CUDA_FP16)
model = cv.dnn_DetectionModel(net)
model.setInputParams(size=(416, 416), scale=1/255, swapRB=True)


cap = cv.VideoCapture('E:/InShot_20220403_223250335.mp4')
starting_time = time.time()
frame_counter = 0
WINDOW_NAME = 'Full Integration'
cv.namedWindow(WINDOW_NAME, cv.WND_PROP_FULLSCREEN)
while True:
    ret, frame = cap.read()
    frame_counter += 1
    if ret == False:
        break
    classes, scores, boxes = model.detect(frame, Conf_threshold, NMS_threshold)
    for (classid, score, box) in zip(classes, scores, boxes):
        color = COLORS[int(classid) % len(COLORS)]
        label = "%s : %f" % (class_name[classid-1], score)
        cv.rectangle(frame, box, color, 1)
        cv.putText(frame, label, (box[0], box[1]-10),
                   cv.FONT_HERSHEY_COMPLEX, 0.3, color, 1)
    endingTime = time.time() - starting_time
    fps = frame_counter/endingTime
    # print(fps)
    cv.putText(frame, f'FPS: {fps}', (20, 50),
               cv.FONT_HERSHEY_COMPLEX, 0.7, (0, 255, 0), 2)
```

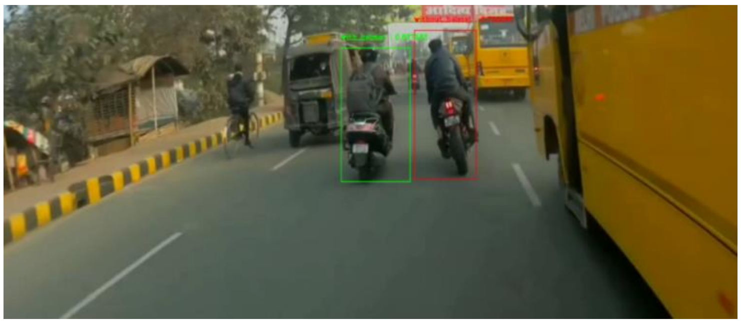


Fig. 10. Implementation of the Trained Model

IX. REFERENCES

[1] M.V.D. Prasad, E. Kiran Kumar, S.V.N.P Vamsi Krishna, M. Santosh Kumar, P. Sri Harsha, SK. Hasane Ahammad - “Detection of Two Wheelers Helmet Using Machine Learning” – 2020 IEEE Turkish Journal of Physiotherapy and Rehabilitatio (2020). IEEE, 2020.

[2] Felix Wilhelm Siebert, Hanhe Lin - “Detecting motorcycle helmet use with deep learning” – 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017.

[3] Thirunavukkarasu.M, Bugade Amoolya, Bulusu Vyagari Vaishnavi - “Helmet Detecting and license plate recognition “- IJCSMC, 2021.

[4] Rui Geng, Yixuan Ma, Wanhong Huang- “An improved helmet detection method for YOLOv5 on an unbalanced dataset” - 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC), IEEE, 2019.

[5] Jie Li, Huanming Liu, Tianzheng Wang - “ Safety helmet wearing detection based on image processing and machine learning” - 2017 IEEE Ninth International Conference on Advanced Computational Intelligence (ICACI) - IEEE, 2017.

[6] Fangbo Zhou, Huailin Zhao, Zhen Nie - “ Safety Helmet Detection Based on YOLOv5 “ - 2021 IEEE International Conference on Power Electronics, Computer Applications (ICPECA) - IEEE, 2021.

[7] Kunal Dahiya, Dinesh SINGH, C. Krishna Mohan - “Automatic Detection of bike riders without helmet using surveillance videos in real time” – 2016 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020

[8] C. Vishnu, Dinesh Singh, C. Krishna Mohan and Sobhan Babu – “Detection of Motorcyclists without Helmet in Videos using Convolutional Neural Network – IEEE, 2020

[9] Madhuchhanda Dasgupta, Oishila Bandyopadhyay, Sanjay Chatterji, "Automated Helmet Detection for Multiple Motorcycle Riders using CNN", Information and Communication Technology (CICT) 2019 IEEE, 2019.

[10] C.Y. Wen, S.H. Chiu, J.J. Liaw and C.P. Lu“ The Safety Helmet Detection for ATM’s Surveillance System via the Modified Hough transform”. Security Technology, IEEE 37th Annual International Carnahan Conference, pp. 364–369. IEEE (2003).

[11] Romuere R.V e Silva, Kelson R.T Aires, Rodrigo de M.S Veras - “Helmet Detecting using Image Descriptors and Classifiers” – IEEE, 2019.

[12] Meenu R, Sinta Raju, Smrithi P Paul, Swathy Sajeev, Alphonsa Jhony “Detection of Helmetless rider using faster RCNN” -IEEE, 2020.

[13] Lokesh Allamki, Manjunath Panchakshari, Ashish Sateesha, K. S Pratheek - “Helmet Detection using Machine Learning and Automatic License Plate Recognition” - IJERT, 2019.

[14] M.V.D. Prasad, S.V.N.P. Vamshi Krishna, M. Santhosh Kumar, P. Sri Harsha - “Helmet Detecting using ML & IoT” - IJERT, 2020.

[15] Romuere Rodrigues Veloso e Silva, Kelson Romulo Teixeira Aires, Rodrigo de Melo Souza Veras - ”Helmet Detection on Motorcyclists Using Image Descriptors and Classifiers” - 27th SIBGRAPI Conference on Graphics, Patterns and Images – IEEE 2017.