iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 10 **Issue:** V **Month of publication:** May 2022

DOI: https://doi.org/10.22214/ijraset.2022.42874

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 10 Issue V May 2022- Available at www.ijraset.com

Pedestrian Detection Based on Deep Learning

Shayan Alam[1], Satyam[2], Prakhar Vashistha[3], Javed Miya[4], Ranjit Kumar[5], Suresh Kumar[6]
[1, 2, 3]*Student, Information Technology at Galgotia college of engineering and technology (Greater Noida)*
[4, 5, 6]*Associate Professor, Information Technology at Galgotia college of engineering and technology (Greater Noida)*

Abstract: Pedestrian detection is most commonly employed in autonomous driving circumstances that necessitate rapid detection. It is also very useful for the purpose of video surveillance and other similar purpose. The primary goal of this project is to create a system that recognises pedestrians from video, or a stream of video sent to the system in the form of previously recorded video or real-time camera input.
Bounding boxes will be drawn around the things that the system has spotted along with the confidence score of prediction. This project employs Python programming and the YOLO machine learning technology.

I. INTRODUCTION

Technology has advanced to the point that modes of transportation have gotten more sophisticated, and people's travel has become more easy, but it has also introduced several safety risks. Accidents involving safety are common. Fatigued and inebriated driving are the direct causes of this phenomena. The primary cause is that the driver is unable to appropriately gauge road conditions in real time and take appropriate emergency measures. As a result, autonomous driving systems have emerged in order to reduce the occurrence of traffic accidents. These systems use computers' fast, stable, and accurate computing power to provide safe driving assurance during the driving process of vehicles, greatly reducing the incidence of traffic accidents. Pedestrian detection is critical in autonomous driving systems. It is necessary to detect not only the number and distance of pedestrians in front of the vehicle, but also their precise location, and to provide reliable data to the driving system's central processor so that measures can be taken to ensure the safe passage of vehicles and the safety of pedestrians. Detecting pedestrians is part of the object detection branch. Object detection is a hot topic these days. It can be utilised in a variety of practical applications, including industrial product detection, intelligent navigation, and security monitoring, to aid government agencies and the majority of businesses in increasing their productivity. Currently, there are two types of pedestrian detection programmes. The first type of scheme is based on background modelling, which involves first determining the object of foreground movement, then extracting features according to the specific area, and finally using a classifier to classify whether there are pedestrians. However, this method has the following issues [1]. The image's chromaticity will be affected by changes in the surroundings, such as light. When the image contains a lot of targets, the detection effect diminishes dramatically [3]. The classifier must make an accurate judgement on the background item's change, but it cannot be classified as a target object. The second type of system is based on statistical learning approaches, which are currently utilised to detect pedestrians. Classifiers for pedestrian detection are created [7].

A. Related Work

The detection of an item in a video sequence is critical in a variety of applications, including video surveillance. Pre-processing, segmentation, foreground and background extraction, and feature extraction can all be used to find objects in a video stream. Humans are quite good at detecting and identifying items in images. The human visual system is quick and accurate, and it can handle complex tasks like detecting many objects with little effort. We can now quickly train computers to detect and classify many items within an image with high accuracy thanks to the availability of vast amounts of data, faster GPUs, and better algorithms [5].
Pedestrian detection is a critical yet difficult vision skill. Many applications rely on it, including image search, image auto-annotation, scene interpretation, and object tracking [4]. One of the most important topics in computer vision was moving object tracking in video picture sequences. It has previously been used in a variety of computer vision applications, including smart video surveillance, artificial intelligence, military guidance, safety detection and robot navigation, as well as medical and biological applications [8]. In recent years, a number of successful single-object tracking systems have emerged, but object recognition becomes challenging in the presence of several objects, and when objects are wholly or partially occluded, they are obtruded from human vision, compounding the challenge of detection. Lighting and acquisition angle are both decreasing. An optimal selection of unique features, as well as the implementation of the Adaboost strong classification algorithm, make the proposed MLP-based object tracking system robust [11].

II. PEDESTRIAN DETECTION SYSTEM MODEL

Figure 1 depicts the object classification system architecture, which includes the dataset, training based on classifying objects based on their morphologic content and colour, to which a module containing train, test, and validation is built, followed by a processing unit, where the input is divided or framed for processing convenience, and the processing unit detects the object by localising the position and displaying it with classification specifications. The neural network is trained on the dataset using PyTorch [13] framework and then the trained model is tested on user input using OpenCV.

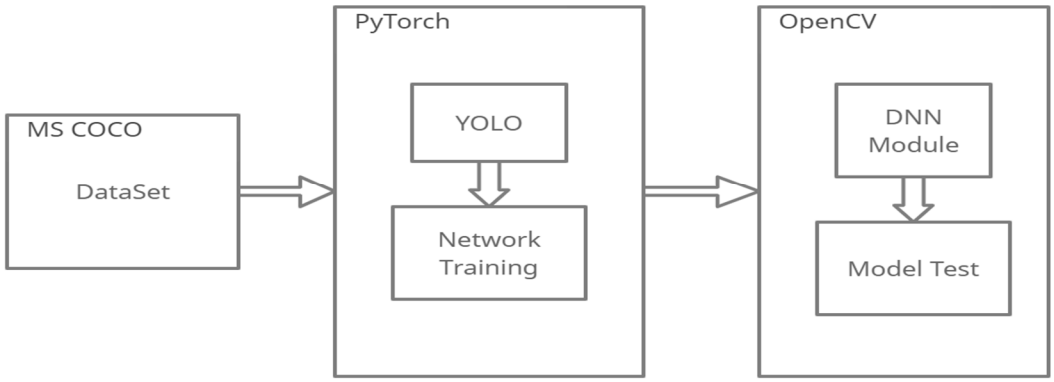


Figure 1: System architecture of object classification

A. *YOLO — You Only Look Once*

To locate the object within the image, all previous object detection techniques utilised areas. The network does not examine the entire picture. Rather, portions of the image with a high likelihood of containing the object. You Only Look Once, or YOLO, is an object detection algorithm that differs significantly from the region-based algorithms discussed previously. The bounding boxes and class probabilities for these boxes are predicted by a single neural network in YOLO [6].

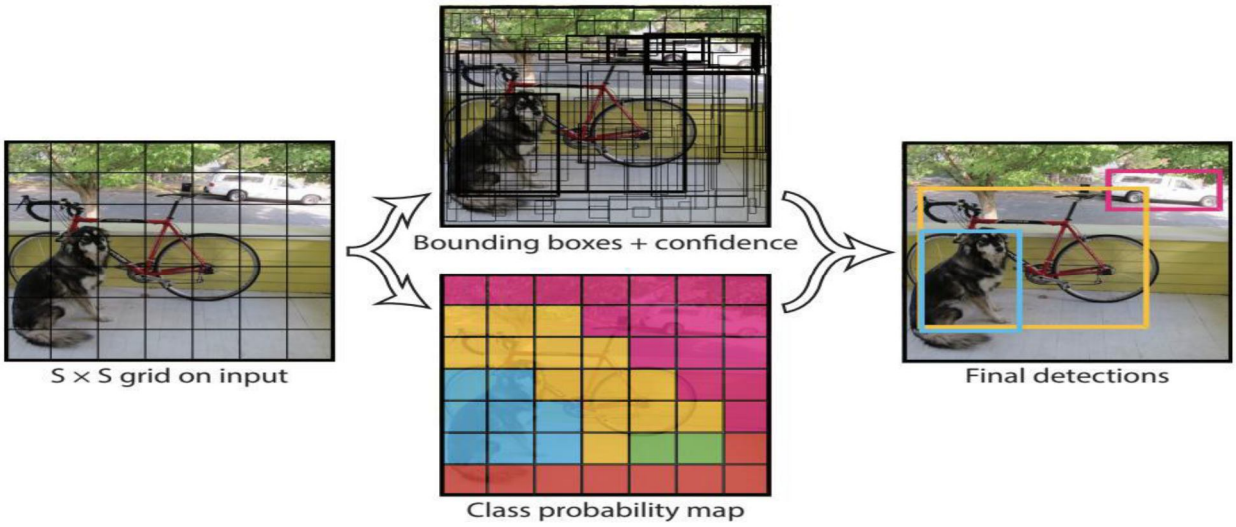


Figure 2 YOLO object detection procedure

The system is fed with images as input. Because video is only a stream of images, it may be used as an input. The input runs through the network only once, as the name implies, and the output of detected item with Bounding Boxes and Labels is obtained [6]. The input is split into a grid of dimensions SxS, and then each grid box predicts bounding boxes containing five parameters (x, y, w, h) and a box confidence score. Each grid box also predicts conditional class probabilities. Finally bounding boxes are drawn around objects with class labels and confidence scores [6].

III. METHODOLOGY

This model is trained on google Collab because it was having lots of problems when trying to install locally. The system used to hang a lot, so the project implementation shifted to google Collab because they provide a GPU processor for doing high GPU required processing.

First, we installed darknet. Then we will split the dataset into two parts, testing and training. We can train data on our model, and after that, we will do the testing. The following things are required for the training and experiment to be conducted.

A. NumPy

It's a Python package that helps with multi-dimensional arrays and matrices. To work with these arrays, it also has high-level mathematical algorithms. In all academic domains, it is utilised to analyse pipelines. Because of NumPy [12], the scientific Python ecosystem has grown tremendously. It serves as the interface between libraries and APIs. It's a fantastic tool for mathematical and scientific study. Images are used in our project and will be transformed to NumPy arrays. The labels are later applied to the image dataset.

B. OpenCV

The entire form of OpenCV is an open-source computer vision library; we're utilising it since we need to install a webcam in the car to identify things in real-time for our project [5]. It's a computer vision library that works in real time [3]. Intel developed this library. This library has the advantage of being open-source and cross-platform. [4] Gaussian blur Method, Canny Edge Detection, and Hough Space are just a handful of the OpenCV algorithms available.

C. PyTorch

PyTorch is an open-source machine learning (ML) framework based on the Torch library and the Python programming language. It is one of the most popular deep learning research platforms. The framework is designed to accelerate the transition from research prototyping to implementation. Tensor computation and functional deep neural networks are two of PyTorch's most notable capabilities [13].

D. Dataset

Data collecting is an important aspect of the research in order to create an effective model. The accuracy of the model is heavily influenced by the data [5]. Microsoft created and maintained the COCO dataset, which we used. Its full form is common objects in context. Although the COCO dataset has 80 different object classes, we are only concerned in the human class [14]. As a result, we used a tool called FiftyOne (which is officially supported by the COCO dataset) to extract the images with the human label and create our own dataset from them.

E. Training

Model training entails training in accordance with the project's requirements. The training will take place in Google Colab Editor, which has GPU support for training the model on the data set. Install the required dependencies first, then upload the dataset to Google Colab. Then we need to make a config file with values that correspond to the classes.

Every algorithm interprets the labelling in its own way. We may now begin the training. Now we must comprehend that when the average loss does not alter, we must cease training. If the loss does not diminish after several iterations, we should cease training. These results can be seen using mAP@.

F. Testing

Testing is the final phase in the process. We'll utilise the same images we used for training for testing after we've trained our own custom data set. The weights that we generate throughout training will be used. To test the model, we will utilise OpenCV, which allows us to receive user input and make predictions based on the model that has been trained on our dataset. Testing can be done in three ways. The first step is to supply an image (png, jpeg, etc.) and receive a labelled image with bounding boxes and accuracy percentage as an output. The second option is to insert a video as input, which will then label the pedestrians and produce a labelled video. The third form, which uses a camera to provide real-time input, is the most essential and widely used.

IV. RESULT AND ANALYSIS

The results of the experiment conducted is shown in this section. The experiment is conducted on two different forms of input, first being an image in standard format (jpeg, png, etc.) and the second is live video stream from web-camera.

A. Providing an Image as Input to the Model

On inserting the following image into the detector, it detects all the pedestrians in it and creates rectangular bounding boxes around them. The bounding boxes are labelled with the 'person' tag along with the confidence score of the prediction. Figure 3 represents the output from the model when an image is provided as input to the model.

B. Providing Real-time Input through Web Camera

In practical applications, intelligent security, automatic driving and other fields need to use pedestrian detection, which has extremely high requirements for the speed and real-time performance of pedestrian detection. Therefore, this section will discuss the real-time performance of the detector. The test is performed through the built-in camera of the laptop. The input is provided as a live video stream from the web-camera of laptop. Bounding boxes are successfully drawn around moving or stationary persons that are encountered within the camera frame. Figure 4 represents the output generated by the model on providing live video stream.

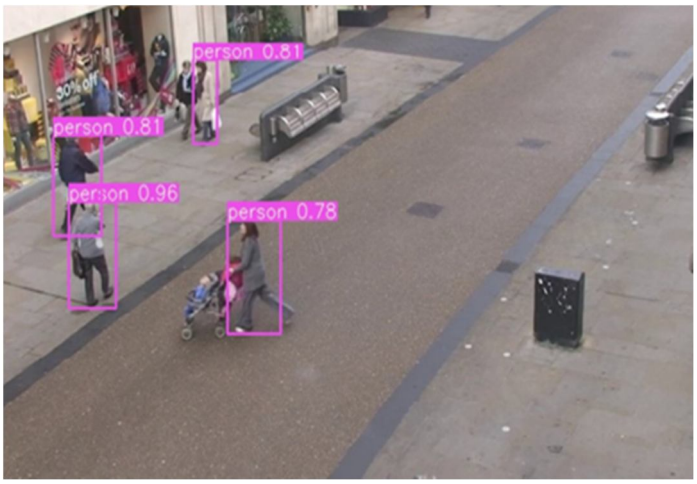


Figure 3: Prediction from input image

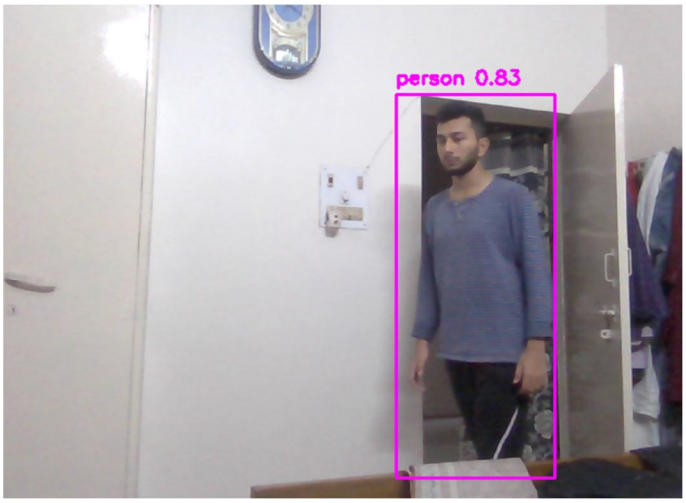


Figure 4: Real-time prediction from web cam

The result of the experiment is analysed in Table 1 on parameters like type of input, expected and actual result and the average confidence of prediction.

Type of Input	Expected result	Actual result	Average confidence
Image Input	Image with bounding box around the objects and predicted class	Successful	84%
Real-Time Input from web camera	Objects detected in the real time with bounding box, confidence score and predicted class	Successful	83%

Table 1: Analysis of the result obtained

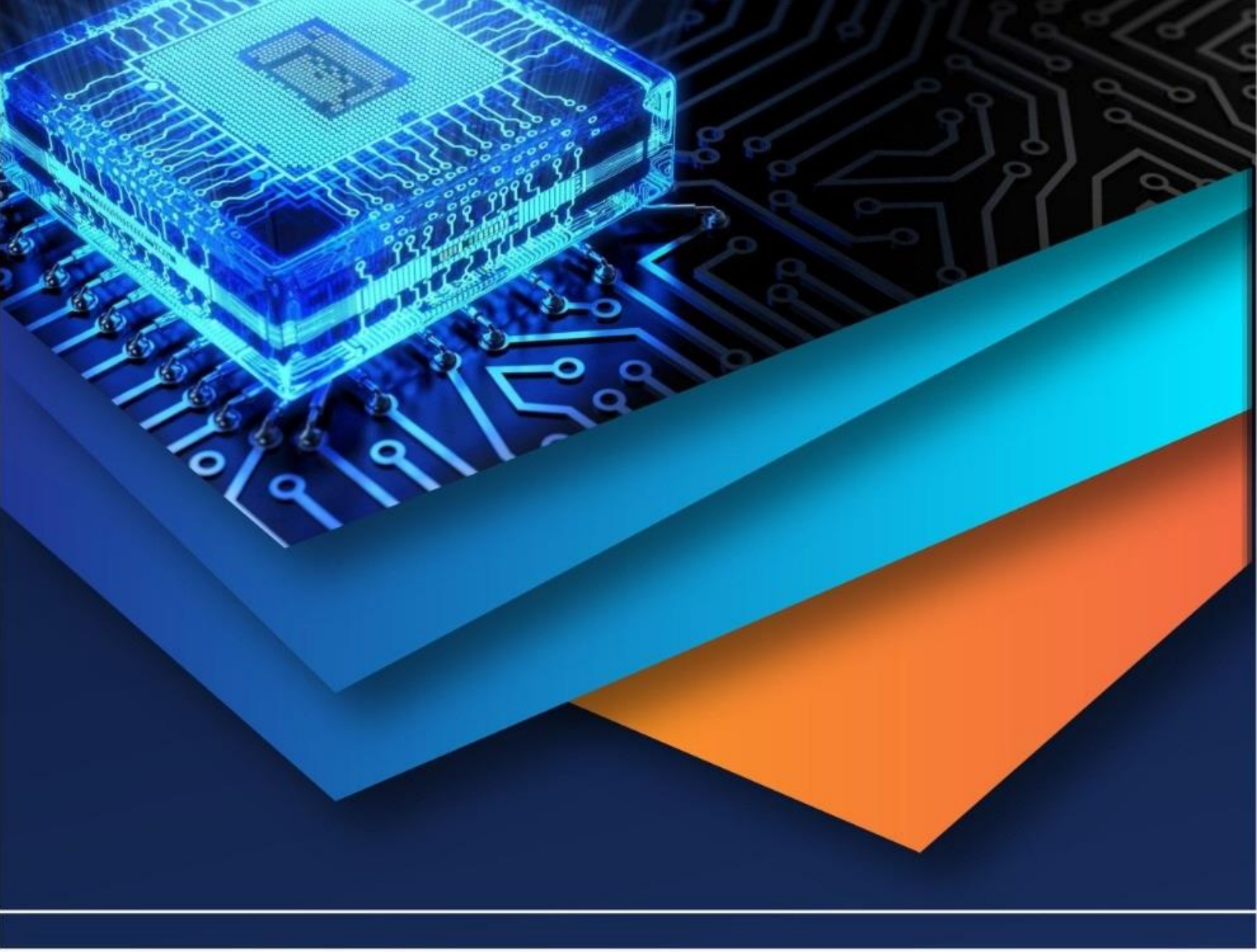
V. CONCLUSION

This paper shows how we are using Pytorch framework to train the model on our custom dataset in the google colab editor on a GPU for higher precision and recall rate. After going through numerous iterations of training we were able to achieve high accuracy. And then we used the weights of the trained model and OpenCV to make predictions on input images and real-time input from the user. The model showed up to 83% accuracy even after making the predictions on a laptop CPU using its web camera.

REFERENCES

We have used the following research papers as basis of our research and project work:

[1] Fruit Classification Comparison Based on CNN and YOLO: Riyanshu Raj et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1187 012031
[2] Detection of natural disaster victims using You Only Look Once (YOLO): M Sarosa et al 2021 IOP Conf. Ser.: Mater. Sci. Eng. 1098 032076
[3] Detection and Content Retrieval of Object in an Image using YOLO: B Vinoth Kumar et al 2019 IOP Conf. Ser.: Mater. Sci. Eng. 590 012062
[4] V. Gajjar, A. Gurnani and Y. Khandhediya, "Human Detection and Tracking for Video Surveillance: A Cognitive Science Approach," in 2017 IEEE International Conference on Computer Vision Workshops.
[5] Javed Miya & M. A. Ansari (2020) Medical images performance analysis and observations with SPIHT and wavelet techniques, Journal of Information and Optimization Sciences, 41:1, 273-282, DOI: 10.1080/02522667.2020.1721616 .
[6] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 779-788).
[7] Ren S Q, He K M, Girshick R, et al.Faster R-CNN: towards real-time object detection with region proposal networks[J].IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(6): 1137-1149
[8] Javed Miya & M. A. Ansari (2020) Wavelet Techniques for Medical Image Performance Analysis and Observations with EZW and Underwater Image Processing, Wireless Personal Communications, doi.org/10.1007/s11277-020-07238-w Springer Nature: https://rdcu.be/b3eSe.
[9] Huang J, Yuan Z, Zhou X. A Learning Framework for Target Detection and Human Face Recognition in Real Time. International journal of technology and human interaction, vol. 15, no. 3, pp. 63-76, 2019.
[10] Shahriar Shakir Sumit1, Junzo Watada1, Anurava Roy1 and DRA Rambli1 In object detection deep learning methods, YOLO shows supremum to Mask R-CNN: Shahriar Shakir Sumit et al 2020 J. Phys.: Conf. Ser. 1529 042086.
[11] Javed Miya & M. A. Ansari (2015) Observations and Performance Evaluation of various Techniques on Medical Images, International Journal of Applied Engineering Research, ISSN 0973-4562 Vol. 10 No.94 pp. 47-51, DOI:10.37622/000000.
[12] NumPy is the fundamental package for scientific computing in Python. Its documentation can be found at https://numpy.org/doc/stable/
[13] PyTorch is the framework used for training the neural network, its documentation can be found at https://pytorch.org/docs/stable/index.html
[14] Microsoft coco (common objects in context) dataset can be found at https://cocodataset.org//#download



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY