Real-time Drone Detection using YOLOv5 on custom dataset

Prajakta Musale, Siddhi Patil and Siddhesh Patil

Department of Engineering, Sciences and Humanities
Vishwakarma Institute of Technology

Bibewadi, Pune, India
{Prajakta.musale,siddhi.patil211 &
siddhesh.patil212}@vit.edu

DOI Number: 10.14704/nq.2022.20.13.NQ88431 **NeuroQuantology2022;20(13): 3508-3514**

Abstract—*Drones have recently occupied a central position in the operations of many businesses and governmental bodies.From quick deliveries at rush hour to scanning an unreachable military base, drones are proving to be very useful in areas that humans cannot access or where they are unable to complete tasks quickly and effectively. But with their advancement, drones are turning out to be a huge security threat. In order to distinguish between drones and other flying objects such as birds, kites, airplanes, and more in a variety of light and weather conditions and distances, this paper suggests a model based on deep learning. A custom dataset and YOLOv5 have been used to achieve the above parameters. For user friendly access the paper proposes a GUI which helps to deploy the model on images, videos and real-time feed.*

Keywords *— convolutional neural network, custom dataset, deep learning,drone detection, graphical user interface, object detection, YOLOv5*

I. INTRODUCTION

Recently, the drone technology has been rapidly introduced [3]. Currently drones have advanced fields like delivery, medicine, agriculture, and filming.

Nonetheless, drones also pose a notable challenge in terms of security and privacy within society (for both individuals and organizations) [4]. Small drones have an increasing chance of being mishandled, especially by hobbyists, for nefarious purposes like drug trafficking, terrorist attacks, or even interfering with emergency services like disaster response and firefighting. Drones can also be converted into dangerous weapons by loading them with explosive materials [6]. This makes detecting drones vital in today's world.

Huge drones which are size of aircrafts are detected using strong military level radar systems. Small drones transmit very limited electromagnetic signals as they have small c ross-sectional area, making it very difficult for conventional radar to detect them. Acoustic and radio frequency detection are expensive and cannot deal with the Doppler effect well [2].

Conversely, video-based object detection has attracted many research interests [7,8] due to high detection accuracy, computing power and long effective range [1]. In our work, we employ video-based detection based on deep learning approach which uses convolutional neural networks. It is concerned with the function and structure of a brain which is known as an artificial neural network [3]. The convolutional neural network (CNN) is gaining the interest of researchers because its detection is automatic for important features at the time of its learning phase. In image classification and object detection CNN shows excellent performance [9]. The CNN capability in extracting the abstract features is the main reason behind its excellent performance [10].

The "You Only Look Once" (YOLO) algorithm has surpassed other CNN based object detection algorithms because of its highly precise real-time detection capability [2]. In 2015, the YOLO (You Only Look Once) algorithm was born with a new approach, reframing object detection as a regression problem and performing in a single neural network [11]. YOLO can achieve very fast detection, which is 67 frames per second (FPS) and can realize real-time

detection. Additionally, YOLO has significantly fewer computer configuration requirements than many detectors based on deep learning, which typically require 4 GB or even 1 GB GPU-RAM for the tiny model [1]. The fifth generation, YOLOv5, is the latest YOLO version but not developed by its original author. The performance of YOLOv5 is highest of all generations in terms of both accuracy and speed [11]. Therefore, we choose YOLOv5 to train a model for detection of drones.

In this paper, we have created a custom dataset and manually cleaned and labeled it. The model we have trained differentiates between drones (UAVs) and other drone-like flying objects like birds, planes and more. This is done in various light and weather conditions and distances. A GUI is proposed in this paper for user-friendly experience which does successful detection locally on images, videos and real-time feed.

II. METHODOLOGY

YOLO commonly used for real-time object detection, applies single neural network to the full image and divides the image into regions and predicts each regions bounding boxes and probabilities.

In the procedure of YOLO training, the image is divided into N ⇥ N cells as a grid. Each cell is in charge of foreseeing potential "B" bounding boxes. The five main characteristics of a bounding box are given as follows. Coordinates x, y at the center of the bounding box, h, w height and width of the bounding box, cs - the confidence which determines the confidence about the presence of an object in the bounding box and the accuracy of the shape of the box. The architecture of YOLO for UAV detection can be seen [Fig. 1.].

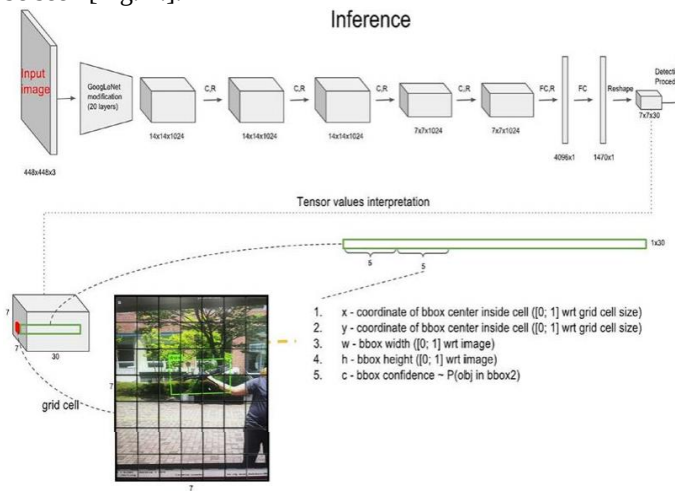

Fig. 1. Architecture of YOLO

The flowchart of YOLO [Fig. 2.] describes how YOLO detects an object when it receives the image. The input image is resized by YOLO to 416 x 416 and then divided into a grid of 5 x 5. If an object center falls on a grid, then the responsibility of the grid is to predict the object with determining bounding box confidence score and conditional probability.

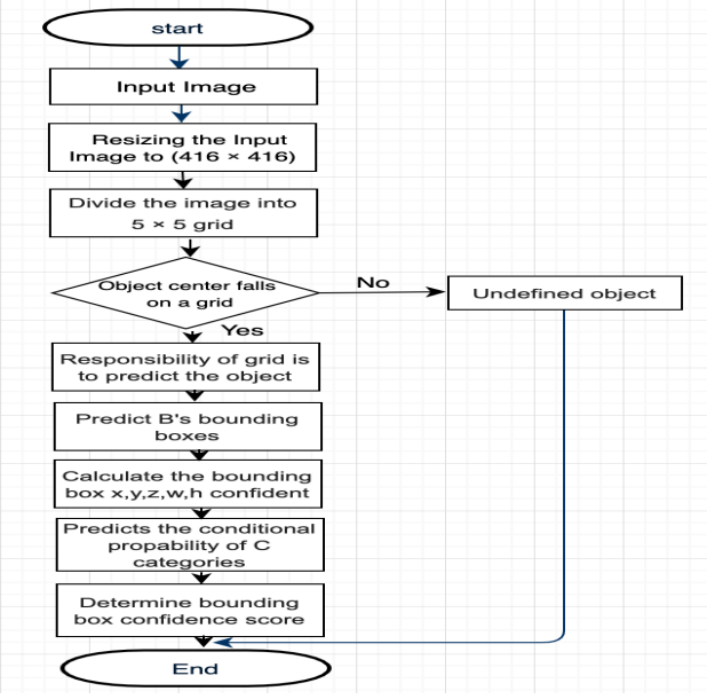

Fig. 2. Flowchart of YOLO model

To train and create a successful model for object detection some specific steps are to be followed [4]. Stages of establishing the drone detection system [Fig. 3.] consist of data collecting, annotation, training and testing.

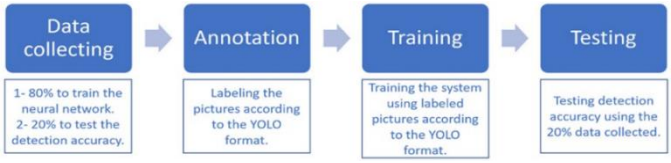

Fig. 3. Stages of Establishing the Drone Detection System

A. *Data Collection*

A custom dataset was created for detecting thedrone and differentiating it from other drone-like aerial objects. The dataset consists of 5 classes namely, airplanes, birds, drones, helicopters and kites. There are images which have objects of multiple classes in one frame [Fig. 4(a)(b)] to increase the performance of the model and help in the differentiating process. The dataset has wide range of types, categories and brands of each class. It was necessary to include data for classes other than drone so that the model understands the other classes thus leading to aptly differentiating between themselves and the drone. This reduces the error margin of the model getting confused between, suppose, a bird and a drone. The collected and gathered dataset had images of all theabove-mentioned classes of various resolutions, angles,

distances, light and weather conditions [Fig. 4(c)]. This helps prepare the model to detect flying objects in any scenario.

The dataset was cleaned manually which was a highly important part [13]. Images which were unclear (extremely low resolution) and might have had confused the model were deleted. Duplicate images were discarded as well.

A total of 4376 images [Fig. 5.] of various category were collected out of which images of drones were maximum since the objective was to train the model to detect drones. Appropriate sorting of data is essential and thus the dataset was divided by 20% for testing and 80% for the detection system training [4]. The model was tested on a video as well.

Fig. 4(a). Image from dataset having bird and drone

Fig. 4(b). Image from dataset having airplane and drone in the same frame

Fig. 4(c). Image from dataset having multiple drones from various distances and unique light intensity

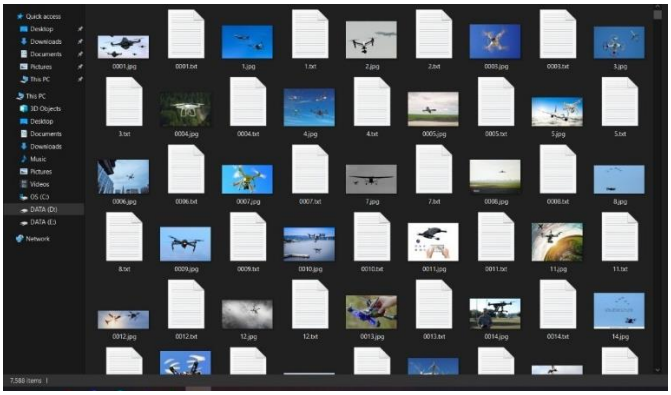

Fig. 5. Dataset size gathered

B. *Annotation*

Next step was annotation, which is labelling the images.For the custom dataset created in this paper 5 labels have been used, one for each class. For this step we used labelImg, an annotation method for graphical images. It is written in Python and its graphical interface uses Qt [4]. We manually labelled each image in the train and validation dataset [Fig. 6.]. The output was a .txt file [Fig. 5.] containing the bounding boxes' coordinates. A YAML file was written containing the paths and number of classes as a list which is a requirement to train a model.

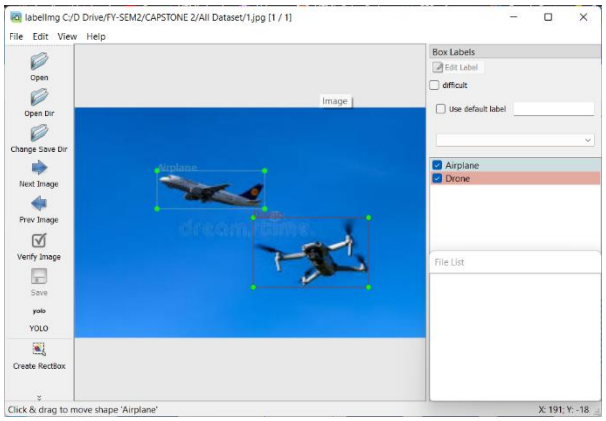

Fig. 6. Annoting images for various classes using labelImg

C. *Training*

Training is a step where the model learnshowto analyze a predetermined set of data and make predictions about what it means. It involves a lot of trial and error until the network is able to accurately draw conclusions based on the desired outcome. Backpropagation is the most common training algorithm for neural networks. To train the model we have used Google Collaboratory since it is fast and offers an online CUDA GPU. We have used the YOLOv5s (small) weights since it is faster for small datasets [12] like ours. Image size used is 416 to run train the model faster and batch size used is 8 since the dataset is small.

The model was first trained at 10 epochs, followed by 50 and was finally settled at 38 epochs. The

validation results for 10 epochs [Fig. 7(a).] show that the results are getting better with every epoch (x-axis) but the final values of precision, mean average precision and recall (y-axis) are quite less. Taking into consideration that there was scope for increasing the values of parameters by increasing the number of iterations, the model was trained at 50 epochs. The validation results for 50 epochs [Fig. 7(b).] were however quite varying. The values of the results had increased but were not the highest. If observed closely a conclusion could be derived that somewhere between 35 to 40 epochs the results for almost every parameter were high (note the highest peak of the graphs). Hence it was finalized to train the model at 38 epochs.

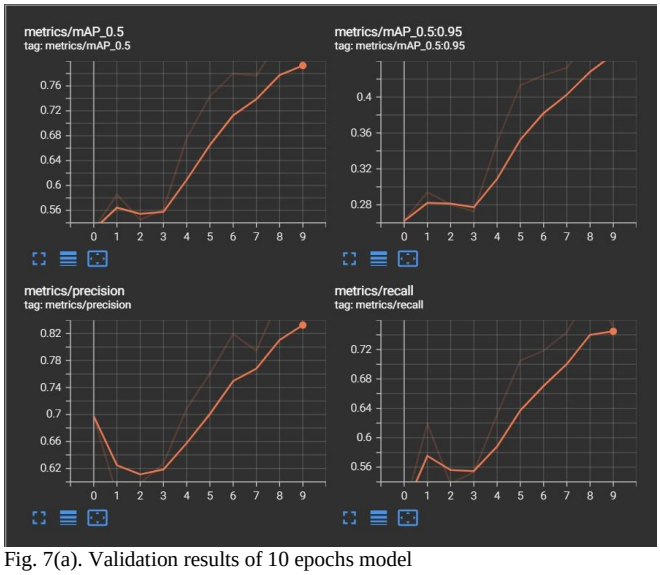

Fig. 7(a). Validation results of 10 epochs model

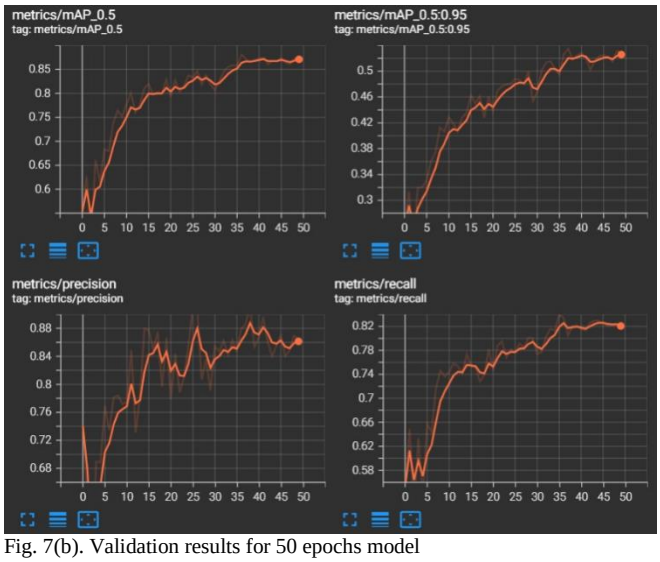

Fig. 7(b). Validation results for 50 epochs model

Part of training is validation. The model we trained runs on a new pre-labelled dataset which gives the results on how our model performed. Using these results one can make changes to the train part of code and decrease error margin. The three parameters considered here are precision, recall and mean average precision. Precision is a quality metric. Higher precision indicates that an algorithm returns more relevant results than irrelevant one. Recall is a measure of quantity. High recall indicates that an algorithm returns the majority of the relevant results. The mAP (mean average precision) incorporates the tradeoff of both metrics and maximizes the effect of both precision and recall. It provides us with a clearer picture of the model's overall accuracy.

Comparing the above-mentioned parameters for the three epochs we can conclude that the model trained at 38 epochs gives the best results thus performing the best.

Epochs	Precision	Recall	maP
10	0.845	0.757	0.799
38	0.869	0.846	0.865
50	0.849	0.821	0.858

Table.I. Validation results of models trained at different epochs

D. Testing

The final step is testing the system with the remaining 20% of dataset. The model was tested on images [Fig. 8(a)(b)(c)(d)] and on a video as well [Fig. 9.]. YOLO divides the video into frames and then runs the model on each frame. The model thus detects drones in real-time.

Fig. 8(a). Drone detected from a significant distance

Fig. 8(b). Drone detected behind an obstacle

Fig. 8(c). Drone and helicopter detected

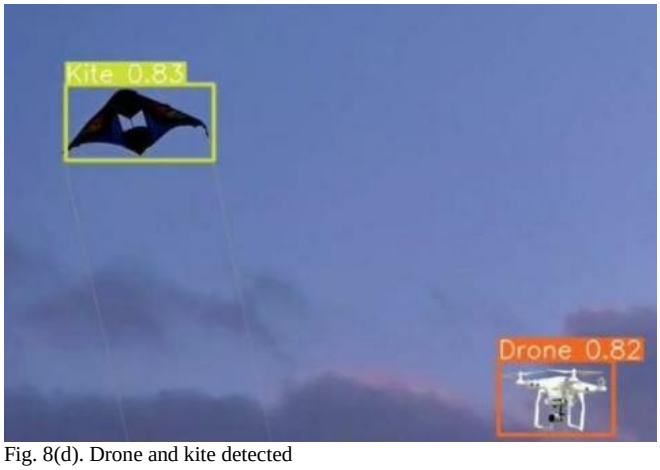

Fig. 8(d). Drone and kite detected

Fig. 9. Drone detected in video in extreme light intensity

To make things user friendly we deployed our model locally through a GUI (graphical user interface) [Fig. 10.]. The GUI was made using python library, Tkinter. Using some functions from OpenCV drones were detected in images [Fig. 11(a).], videos [Fig. 11(b).] and real-time feed [Fig. 11(c).].

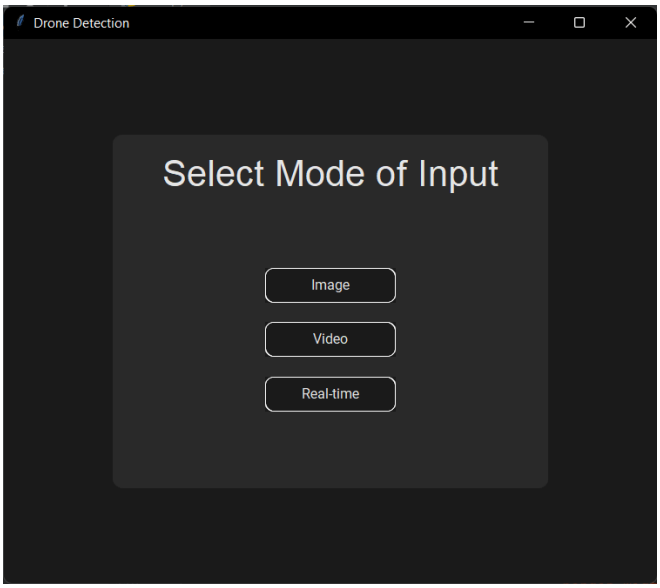

Fig. 10. GUI to detect drones locally

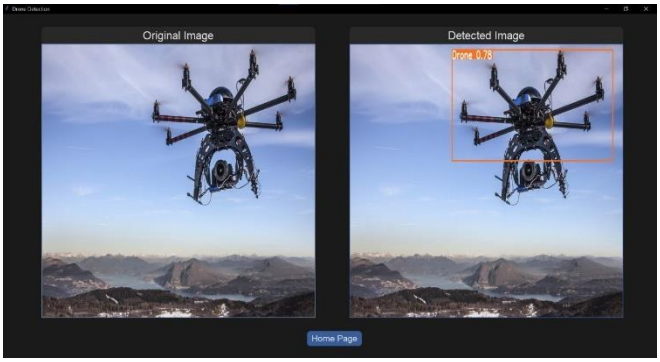

Fig. 11(a). Drone detected locally in image through GUI

Fig. 11(b). Drone detected locally in video through GUI

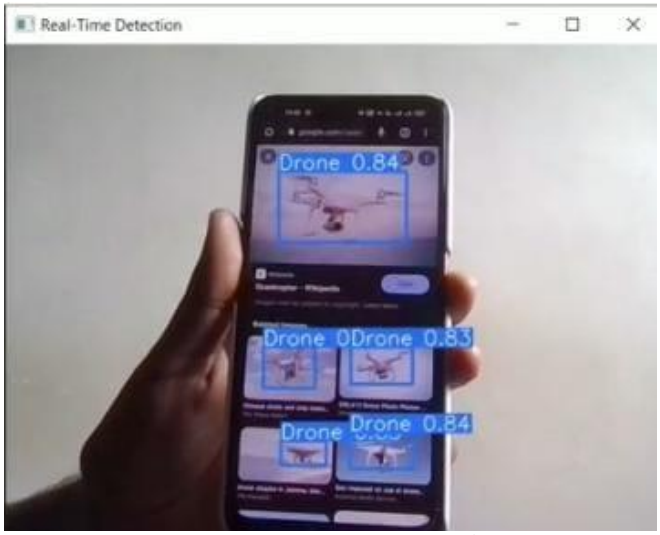

Fig. 11(c). Drone detected locally in real-time camera feed through GUI

III. Results and Discussions

As can be seen from the test and validation results, the model trained at 38 epochs achieved the goals and purposes of drone detection. The model detects the various flying objects with an average accuracy of 80%.

The graphs of various parameters and classes show that the overall performance of the model is good. The graphs show how the model performed for each class and combining all classes as well [Fig. 12(a)(b)(c)(d)].

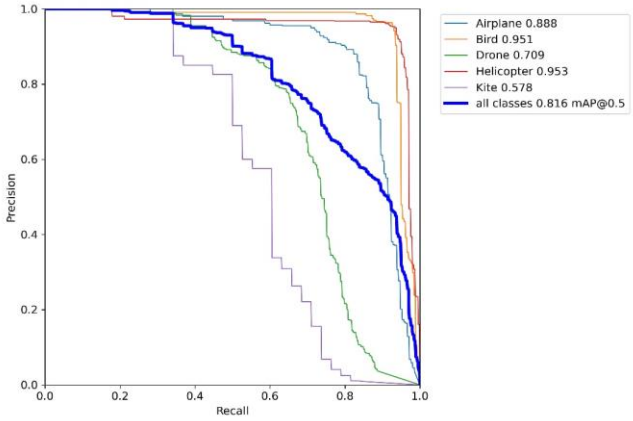

Fig. 12(a). Precision-Recall graph for all classes

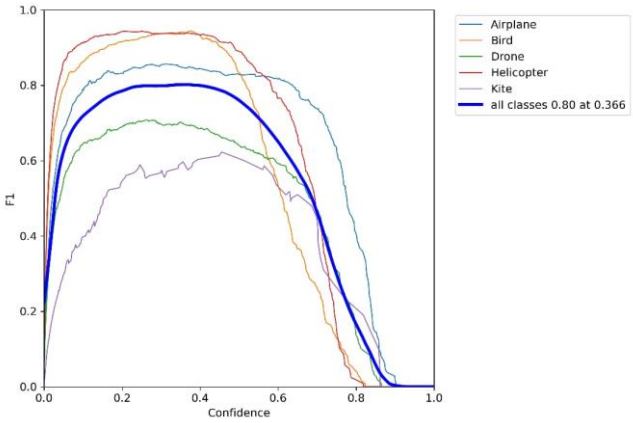

Fig. 12(b). F1-Confidence graph for all classes

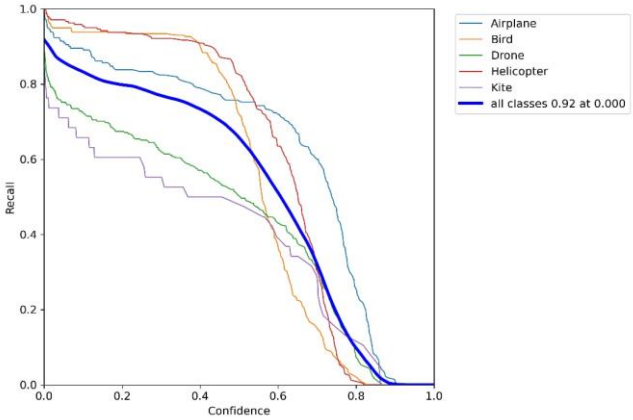

Fig. 12(c). Recall-Confidence graph for all classes

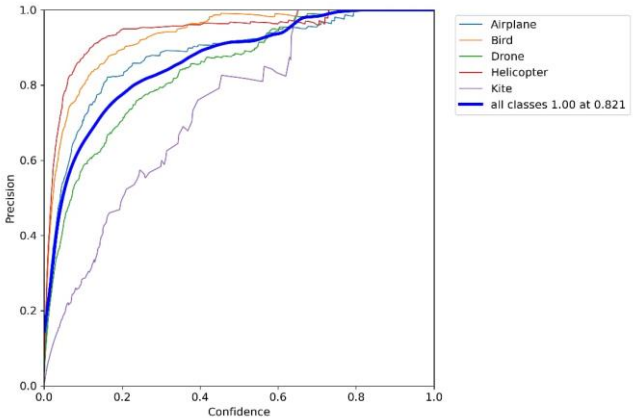

Fig. 12(d). Precision-Confidence graph for all classes

The graph for validation results of the model trained at 38 epochs [Fig. 13.]shows improvements in the parameters as well.

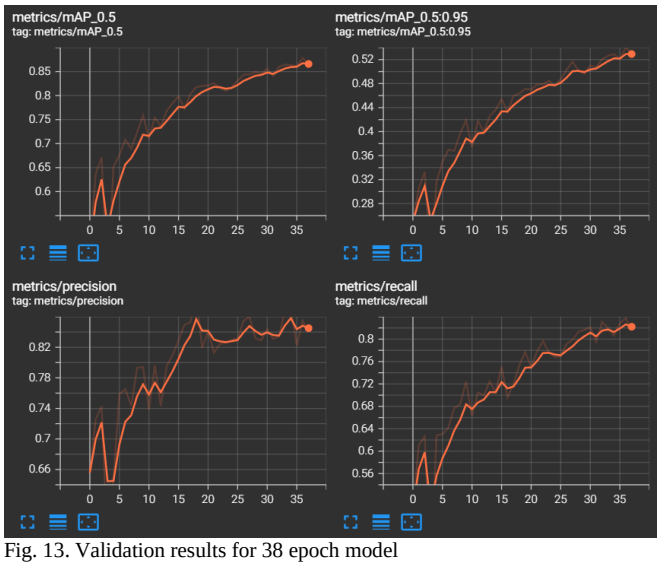

Fig. 13. Validation results for 38 epoch model

In a range of difficulties, including occlusion, lighting changes, weather changes, shades and partial visibility, the model was able to work. The model successfully differentiated between the various classes with above average accuracy. The model did all of the above in real-time as well.

IV. Conclusion

All in all, for real-time drone detection a deep learning approach, YOLOv5 is the simplest and best architecture for any kind of detection of real-time objects. YOLOv5 is easy to incorporate, easy to comprehend and stable.

Drone detection is necessary, considering that drone intervention is frequent in unauthorized and emergency tasks. The model trained in this paper satisfies the problem-statement of detecting drones in various conditions, differentiating drone from other drone-like aerial objects and to do so in real-time.

Due to the scarcity of drone data sets, we make an extensive effort to create our own custom dataset. It can be concluded that the model can be made better by increasing the dataset.

Functionalities like alerting the authorities on drone being detected and predicting the drone path (receding or approaching) can be added in the future to make the output more practical.

X.Acknowledgment

Guidance throughout the project, in researching,while creating our custom dataset and training the model was provided by Prof. P Musale. Our sincere gratitude to VIT Pune DESH department for this opportunity.

References

[1] Manjia Wu1, WeigeXie, Xiufang Shi, Panyu Shao, and Zhiguo Shi, “Real-Time Drone Detection Using Deep Learning Approach,” ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2018 L. Meng and Y. Zhang (Eds.): MLICOM 2018, LNICST 251, pp. 22–32, 2018

[2] Subroto Singha, andBurchan Aydin, “Automated Drone Detection Using YOLOv4,” 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution

[3] Syed Ali Hassan, Tariq Rahim, and Soo Young Shin, “Real-time UAV Detection based on Deep Learning Network,” 978-1-7281-0893-3/19/$31.00 ©2019 IEEE

[4] AbdulAziz Bin Shuhaywin, Faisal AlMeshkhas, and Dr. SobhiMejjaouli, “Establishing Drone Detection System by Using Deep Learning and YOLOv3 Formatting,” Proceedings of the 11th Annual International Conference on Industrial Engineering and Operations Management Singapore, March 7-11, 2021

[5] Divya Joshi, “Drone technology uses and applications for commercial, industrial and military drones in 2020 and the future,” Business Insider India

[6] Behera, D.K., Raj, A.B., “Drone Detection and Classification Using Deep Learning,” In Proceedings of the 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 13–15 May 2020; IEEE: Piscataway Township, NJ, USA; pp. 1012–1016

[7] Sevil, H.E., Dogan, A., Subbarao, K., Huff, B., “Evaluation of extant computer vision techniques for detecting intruder sUAS,” 2017 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE (2017), pp. 929–938

[8] Sunyou Hwang1, Jaehyun Lee1, Sungwook Cho1, David Hyunchul Shim1, and Heemin Shin, “Aircraft detection using deep convolutional neural network in small unmanned aircraft systems,” 2018 AIAA Information Systems-AIAA Infotech@ Aerospace, p. 2137 (2018)

[9] He, K., Zhang, X., Sun, J., and Ren, S., “Deep residual learning for image recognition,” proceedings of the IEEE conference on computer vision and pattern recognition (2016), pp. 770-778

[10] Shi, W., Caballero, J., Huszar, F., A. P., Aitken, Totz, J., Bishop, R., and Wang, Z., “Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network,” proceedings of the IEEE conference on computer vision and pattern recognition (2016), pp. 1874-1883

[11] Do Thaun, “Evolution of YOLO Algorithm and YOLOv5: The State-of-the-art Object Detection Algorithm,” Oulu University of Applied Sciences

[12] Glen Jocher, “YOLOv5 Documentation,” https://docs.ultralytics.com/

[13] Inés Roldós, “8 Effective Data Cleaning Techniques for Better Data,” https://monkeylearn.com/blog/data-cleaning-techniques/