PRACTICE SCHOOL-1
PROJECT REPORT

CV-BASED DRONE DETECTION USING COTS ALGORITHMS

BY

L SHIVA RUDRA	2020A8PS1246H
HEMANTH REDDY	2020A4PS0905H
KOTA VENKATA BHARGHAV	2020B2A82088G
TARUN RAJKUMAR	2020A8PS1447H

AT

THE MILITARY COLLEGE OF ELECTRONICS AND MECHANICAL ENGINEERING

A Practice School - 1 Station of

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

(JUNE, 2022)

Practice School Details

Station: Military College of Electronics and Mechanical Engineering

Centre: Secunderabad, Telangana

Date of start: 30 May 2022

Date of submission: 20 July 2022

Title of the project: CV based drone detection using COTS algorithms

Name of the PS faculty: Dr. Mithun Mondal

Name of the PS mentor: Lt Col. Anuj Singh

Project Areas: Computer Vision, Deep learning

ACKNOWLEDGEMENT

We would like to thank Prof. S.P. Regalla, the Dean of Practice School Division and BITS Practice School Division for this excellent opportunity to pursue Practice School-I program at the Military College of Electronics and Mechanical Engineering, Secunderabad, Telangana.

We would like to express our heartfelt gratitude towards our PS Faculty, Prof Dr. Mithun Mondal for his dedication towards students currently in the PS and for his quick response to all our queries.

We are also grateful to Dr. Shivendra Kumar, Dean of MCEME, for giving us an opportunity to be a part of an elite organisation and contribute to the development of the project on CV based drone detection.

We would also extend our gratitude to our project guide, Lt Col Anuj Singh and Maj Vinay Chandran, who were not only overseeing the project personally, but also guided us whenever we were in need of any help.

ABSTRACT

Unmanned Aerial Vehicles (UAVs), commonly known as drones have witnessed a massive increase in the past few years. Drones are not only being used for recreational purposes but also in a vast number of applications in engineering, disaster management, logistics, securing airports, etc. Extensive usage of drones has immediately raised security concerns due to the potential of their use in malicious activities. To address this problem, this project is aimed at analysing the available drone detection solutions which can identify drones from the day and night camera feeds in real-time. This study also aims to identify the potential challenges involved in drone detection and study the various accuracy and loss metrics in drone detection problems. Finally, the drone detection solutions will be compared and a suitable COTS drone detection solution will be suggested such that it gives the best trade-off between false alarm and miss. The main focus of this project will be on object detection methods with deep learning.

CONTENTS

LITERATURE REVIEW

General Overview of Computer Vision

Computer vision is more related to visual perception i.e., it is the science of perceiving and understanding the world through images and videos(which is the series of frames over time) and by building a physical model of the world so that an AI system can perform the desired tasks.

Computer vision trains machines to determine various characteristics of the given visual data such as detecting the object in interest, its location and the corresponding probability value. This involves loading lots of data(sometimes millions of images for example) and training them with various algorithms and then testing them with another dataset. This is because, unlike humans, machines are not that intuitive so they need more data to improve the accuracy of their predictions.

For example, if we want our computer to distinguish between a cat and dog by just loading a single image of each, we cannot expect a good prediction. This is because the algorithm that we use starts by comparing the features of both the animals. But both of them have got four legs, two eyes and many such similarities. Now if we pass an image of a fox, the system may be confused and make a wrong prediction. So, we expect it to

tweak a little bit and make a better dataset by training as many models as possible.

Deep learning

It is the subset of Machine Learning that teaches computers what is natural to humans. Deep learning models involve neural networks. There are several layers present in this network in which each layer consists of a basic unit called neurons which are like nodes through which data flows in from the previous node and flows out to the next layer or as the output.

While there are neural network models like Artificial neural networks(ANN), Recurrent neural networks(RNN); we will be focussing primarily on Convolutional neural networks(CNN).
This is because CNN is used in image recognition and processing that is specifically designed for processing pixel data.
Using CNN involves processing images in two parts:

- First, the convolution base: Here, various filters are used in different layers that are used to detect various features in the images like lines, edges, curves etc. and generate response maps which are arrays of probabilities of the presence of the respective filters in each pixel of the image. These response maps are then passed on to the next layers where further filters may be present. This goes on until a final response map is created. Here, the filters are trainable parameters.

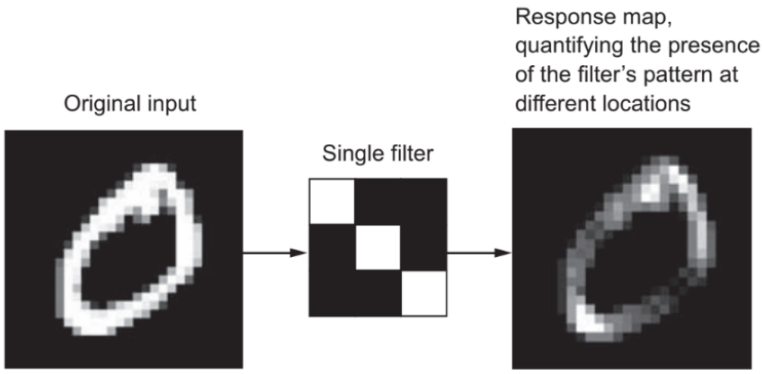
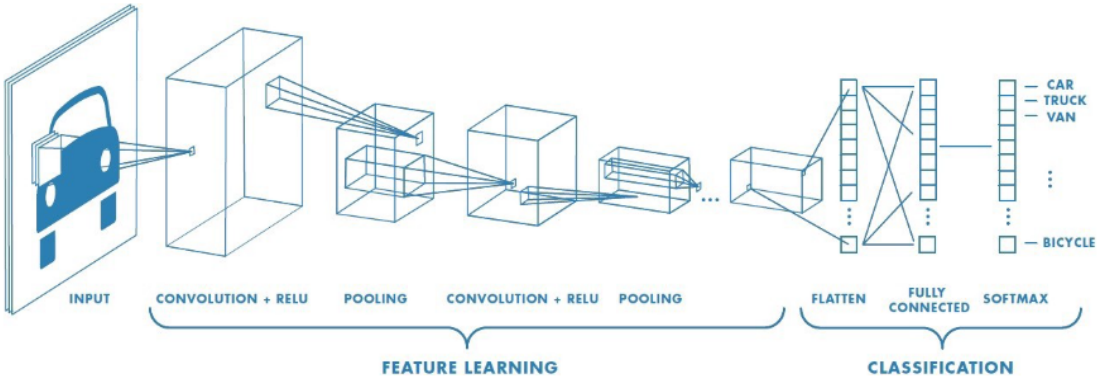


Figure 5.3 The concept of a *response map*: a 2D map of the presence of a pattern at different locations in an input

- Secondly, there is another neural network called the Dense neural network(DNN) which is used for classification purposes. It takes in the response map from the convolution base and based on this, it has several layers of neurons which compute the final class to which the image belongs. Here, weights and biases are trainable parameters.

The following is the summary of CNN:



The above image consists of the following technical terms:

- Convolution: It is the process of feature detection and making the probability of feature presence.

- ReLU(Rectifying Linear Unit): It is an activation function which is used to restrict the output between set values. Generally, it cancels out any negative values and allows the positive values to some extent.
- Pooling: Generally, ‘maximum value Pooling’ is the procedure that is used. This is just to pick up the maximum value from a set of values from a group of pixels after every convolution with a filter.
- Flatten: The final output from convolutional base is an array(since it is related with pixels and images), but to give this as input to DNN, it is flattened(converted to 1-D data).

To understand the above with code, let us consider the following example taken from the “TensorFlow” website:

```
import tensorflow as tf

from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
#  Importing all the necessary libraries

#  Loading and splitting dataset
(train_images, train_labels), (test_images, test_labels) =
datasets.cifar10.load_data()

# Normalise pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0

class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']

# Let's look at a one image
IMG_INDEX = 7  # change this to look at other images

plt.imshow(train_images[IMG_INDEX] ,cmap=plt.cm.binary)
plt.xlabel(class_names[train_labels[IMG_INDEX][0]])
```

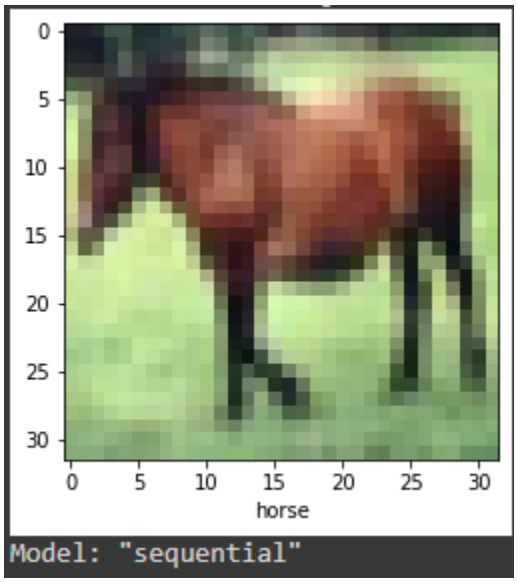
```
plt.show()
# The 'plt' is used for plotting purposes


# The following lines are used to model the entire network(Convolutional base +
Dense NN)
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32,
3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))

# The following lines are for building the Dense NN
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))

model.summary()
# model.summary() shows the model history
```

We get the following image by using ‘plt’ :



And model.summary() gives:

```
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)              (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928

 flatten (Flatten)            (None, 1024)              0

 dense (Dense)                (None, 64)                65600

 dense_1 (Dense)              (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

Now, we add the following code to train and compile the model we built above:

```
model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=4,
                    validation_data=(test_images, test_labels))
```

And the following lines to test the model:

```
test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
print(test_acc)
```

The output we get on running the above code is as follows:

```
313/313 - 4s - loss: 0.9751 - accuracy: 0.6593 - 4s/epoch - 12ms/step
0.6593000292778015
```

Thus, we get the accuracy as 65.9 percent which can be further improved by using better algorithms and feeding in more data and tuning how many times the same data is fed in.

Existing research on drone detection

In recent years, drone detection, recognition, and identification have gained a lot of attention. Conventionally, the dataset for drone detection is obtained using active and passive sensors. In the case of active sensors, Radar and LIDAR sensors were used. However, the problem with active sensors includes high costs and limited integration into small drones. In addition, the use of thermal sensors results in lower accuracy due to low spatial resolution. Therefore, to overcome the above-mentioned limitations, visible imagery was used in the context of passive sensors that do not have weight limitations when integrated into small drones.

However, drone detection and recognition have other complications associated with them. Issues such as unpredictable movement and speed of the drones, long-distance of the drone, its close resemblance to birds, its small size, the presence of hidden areas in images, crowded backgrounds, the problem with light in visible images, and various weather conditions challenge drone detection. For this reason, new methods of deep learning are used to solve these challenges. In

recent years, researchers detected drones in a set of visible images using artificial intelligence-based methods and using RPN (Region Proposal Network), CNN (Convolutional Neural Networks), and YOLOv2 neural networks. The limitation of these studies was the low accuracy in detecting drones.

In 2019, drone detection was performed using YOLO (You Only Look Once), Faster-RCNN, and SSD methods. RCNN (Region-based Convolutional Neural Networks) and SSD (Single Shot Detector) methods were used to detect drones in video datasets, with the RCNN method showing better accuracy. The use of the YOLOv3 deep learning network in this study has resulted in improved accuracy and precision of drone detection compared to other methods due to its lightweight architecture and appropriate depth. In 2020, drones were detected using YOLOv3, YOLOv2, tiny-YOLOv3, Faster-RCNN, and SSD networks, and the results were compared. The YOLOv2 and YOLOv3 deep learning networks had the best accuracy.

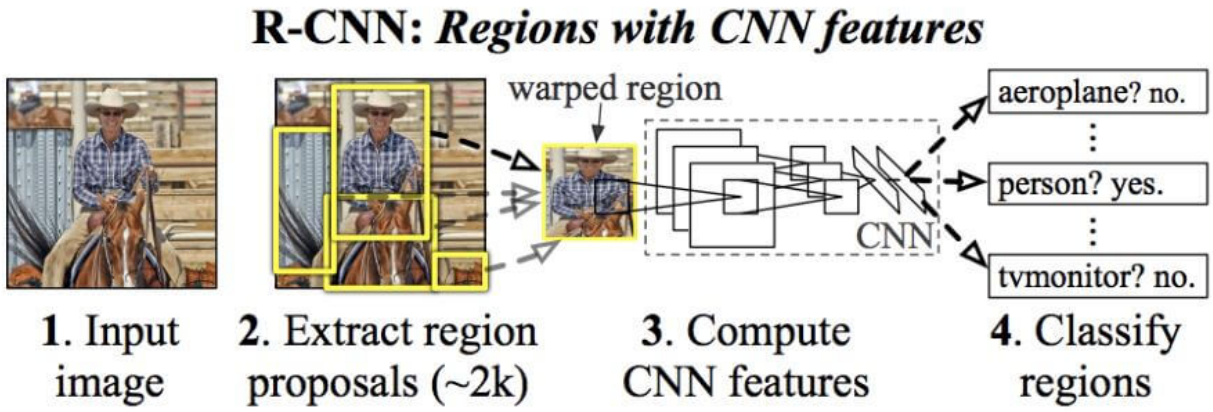
In 2021, the challenges in drone detection were examined in more detail. This year, segmentation-based methods were used to detect drones in crowded backgrounds, and another study detected drones in real-time using the YOLOv3 network on NVIDIA Jetson TX2 hardware. The use of this method has provided good accuracy and speed and is capable of detecting drones of various sizes. Other methods used to detect drones

include Faster-RCNN, SSD, YOLOv3, and whose performance was examined in a series of visible drone images. All the methods used in this study performed well in detecting drones, but YOLOv3 provided the best precision. Researchers have also recently used YOLOv4. The use of YOLOv4 in the initial study provided sustainable drone detection results compared to similar studies and had better accuracy. Furthermore, in the next study, the network used had good accuracy as well as good performance in detecting small and fast drones. Therefore, the pruned YOLOv4 method gave better performance compared to the above-discussed methods.

Popular Object Detection Algorithms

1) Region-based Convolutional Neural Networks (R-CNN)

Region-based Convolutional Neural Networks are approaches that apply deep models to object detection. R-CNN models first select several proposed regions from an image (for example, anchor boxes are one type of selection method) and then label their categories and bounding boxes. These labels are created based on predefined classes given to the program. They then use a convolutional neural network to perform forward computation to extract features from each proposed area. In R-CNN, the inputted image is first divided into nearly two thousand region proposals, i.e bounding boxes for image classification. R-CNN creates these bounding boxes using a process called ‘Selective Search’ and then, for each bounding box, image classification is done through CNN. Finally, each bounding box is refined using regression.



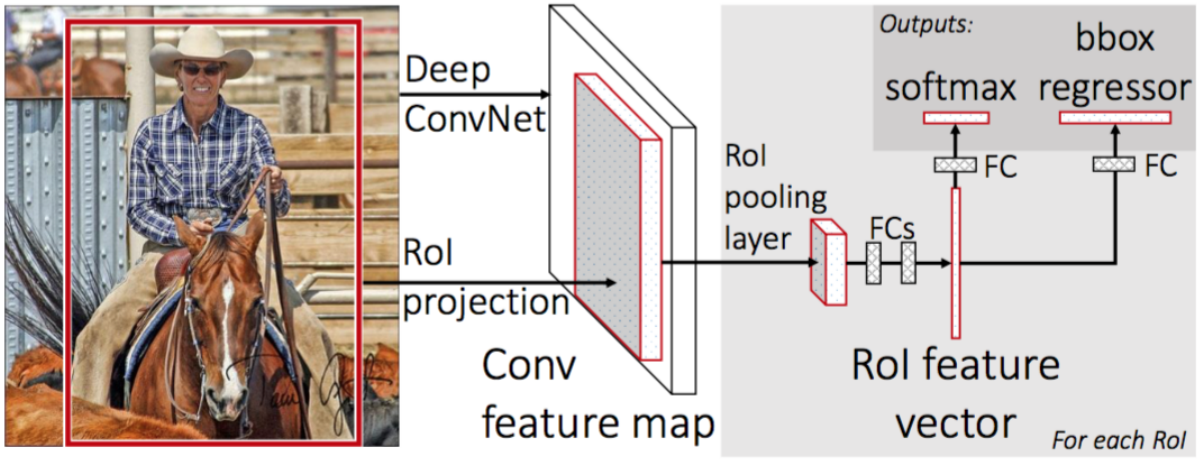
Drawbacks of R-CNN :

- Time taken to train the network is huge as the network has to classify around 2000 region proposals per image.
- It cannot be implemented in real time as it takes around 47 seconds for each test image.
- The 'Selective Search' algorithm is a fixed algorithm. Therefore, no learning is happening at that stage, This could potentially lead to generation of bad regions of proposal.

2) Fast R-CNN

It is computationally extensive to train and even test the image using R-CNN. To deal with this problem Fast R-CNN was proposed. The approach to Fast R-CNN is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, we feed the input image to the CNN to generate a convolutional feature map. From the convolutional feature map,

we identify the region of proposals and warp them into squares and by using an RoI pooling layer we reshape them into a fixed size so that it can be fed into a fully connected layer. From the RoI feature vector, we use a softmax layer to predict the class of the proposed region and also the offset values for the bounding box.



Fast R-CNN architecture

Advantages of Fast R-CNN :

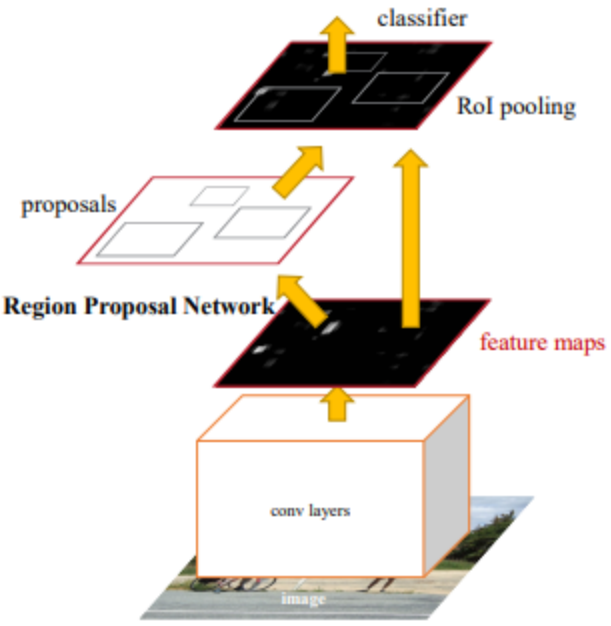
- Fast R-CNN drastically improves the training (8.75 hrs vs 84 hrs) and detection time from R-CNN.
- It also improves the Mean Average Precision (mAP) slightly as compared to R-CNN.

Drawbacks of Fast R-CNN :

- The bottleneck of the Fast R-CNN architecture is selective search. Since it needs to generate 2000 proposals per image, it constitutes a major part of the training time of the whole architecture.

3) Faster R-CNN

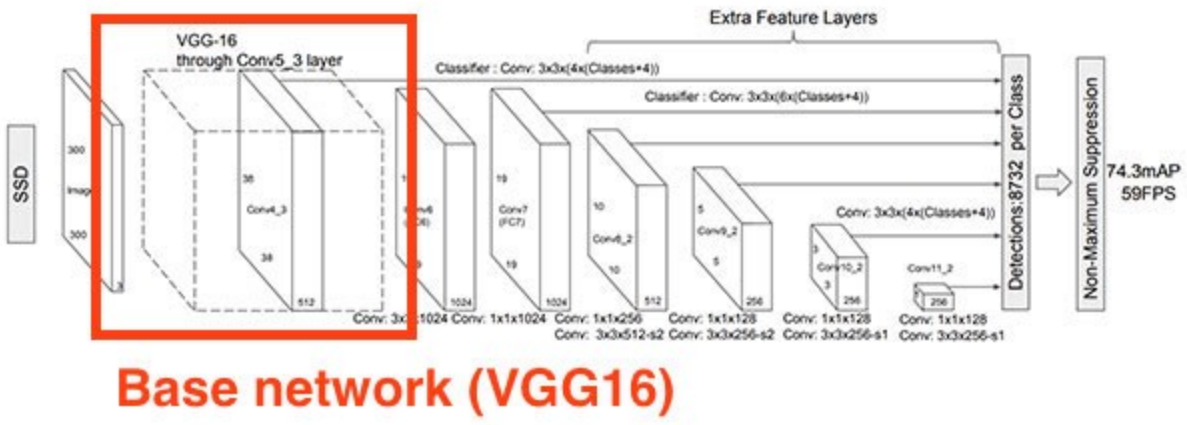
In Fast R-CNN, the bottleneck of the architecture is selective search. Since it needs to generate *2000* proposals per image, it constitutes a major part of the training time of the whole architecture. In Faster R-CNN, it was replaced by the Region Proposal Network (RPN). First of all, in this network, we pass the image into the backbone network. This backbone network generates a convolution feature map. These feature maps are then passed into the region proposal network. The region proposal network takes a feature map and generates the anchors (the centre of the sliding window with a unique size and scale). These anchors are then passed into the classification layer (which classifies that there is an object or not) and the regression layer (which localize the bounding box associated with an object). In terms of Detection time, Faster R-CNN is faster than both R-CNN and Fast R-CNN. The Faster R-CNN also has better mAP than both the previous ones.



Faster R-CNN architecture

4) Single Shot Detection (SSD)

Single Shot Detector (SSD) is a method for detecting objects in images using a single deep neural network. The SSD approach discretises the output space of bounding boxes into a set of default boxes over different aspect ratios. After discretizing, the method scales per feature map location. The Single Shot Detector network combines predictions from multiple feature maps with different resolutions to naturally handle objects of various sizes.The tasks of object localization and classification are done in a single forward pass of the network.



SSD Architecture

The base network is just one of the many components that fit into the overall deep learning object detection framework — the figure at the top of this section depicts the VGG16 base network inside the SSD framework.

Advantages of SSD:

- SSD attains a better balance between swiftness and precision. SSD runs a convolutional network only once on the input image and computes a feature map.
- Easy to train and straightforward to integrate into systems that require a detection component.

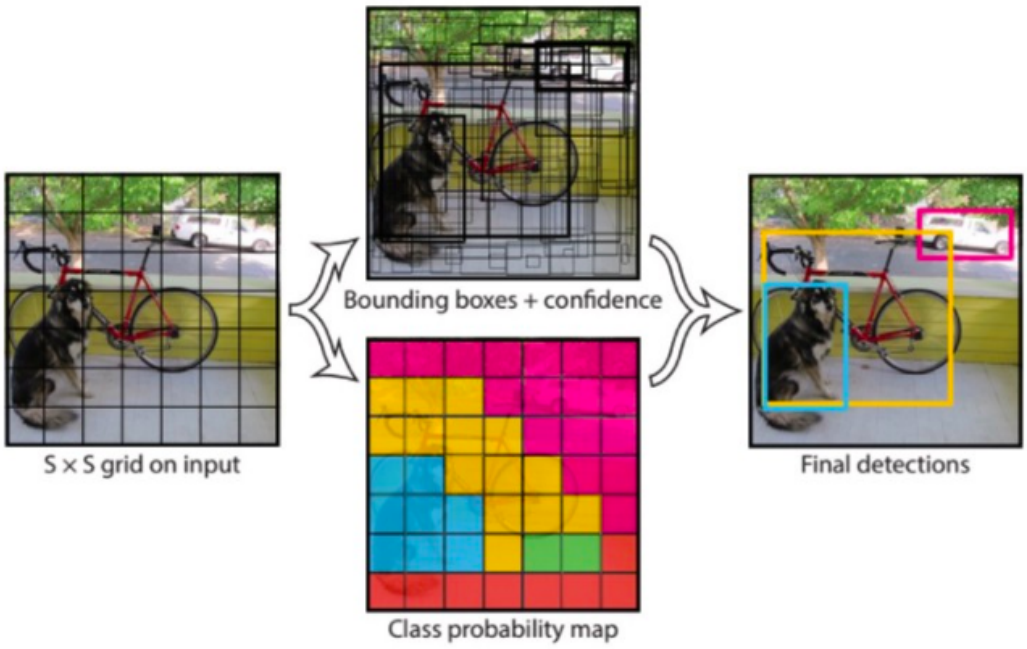
5) YOLO

All of the previous object detection algorithms use regions to localize the object within the image. The network does not look at the complete image. Instead, parts of the image which have high probabilities of containing the object. YOLO or You Only Look Once is an object detection algorithm much different from the region based algorithms seen above. In YOLO a single convolutional network predicts the bounding boxes and the class probabilities for these boxes.

How YOLO works is that we take an image and split it into an SxS grid, within each of the grid we take m bounding boxes. For each of the bounding box, the network outputs a class probability and offset values for the bounding box. The bounding boxes having the class probability above a threshold value is selected and used to locate the object within the image.

Advantages and disadvantages of YOLO:

- YOLO is orders of magnitude faster(45 frames per second) than other object detection algorithms.
- The limitation of YOLO algorithm is that it struggles with small objects within the image, for example, it might have difficulties in detecting a flock of birds. This is due to the spatial constraints of the algorithm.



6) Mask R-CNN

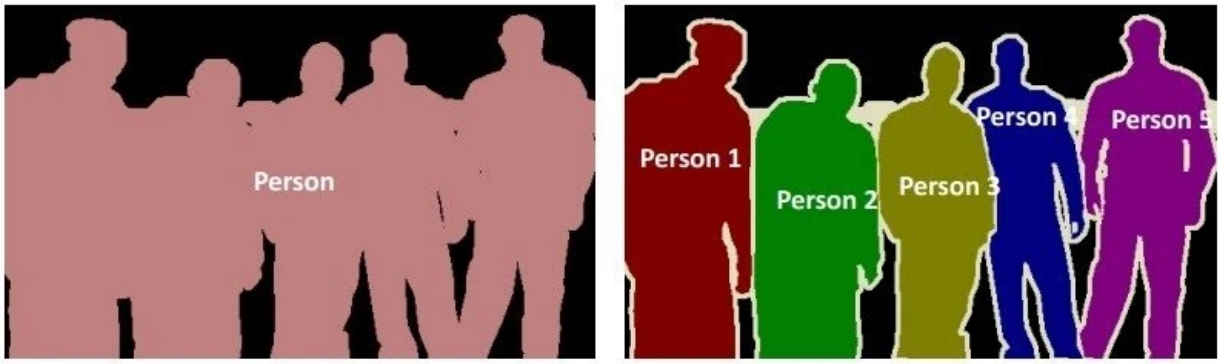
The first step to understanding how Mask R-CNN work requires an understanding of the concept of Image Segmentation

The computer vision task Image Segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as image objects). This segmentation is used to locate objects and boundaries (lines, curves, etc.).

There are 2 main types of image segmentation that fall under Mask R-CNN:

1. Semantic Segmentation(deals with the classification of similar objects as a single class from the pixel level)

2. Instance Segmentation (deals with the correct detection of all objects in an image while also precisely segmenting each instance)

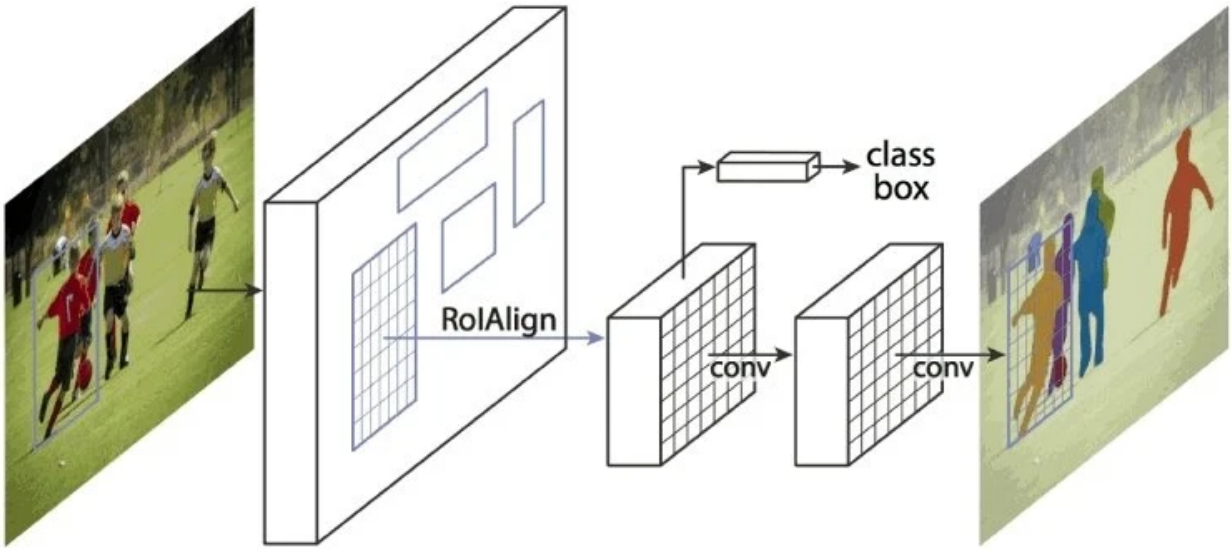


Semantic Segmentation **Instance Segmentation**

Mask R-CNN was built using Faster R-CNN. While Faster R-CNN has 2 outputs for each candidate object, a class label and a bounding-box offset, Mask R-CNN is the addition of a third branch that outputs the object mask. The additional mask output is distinct from the class and box outputs, requiring the extraction of a much finer spatial layout of an object.

Advantages of Mask R-CNN:

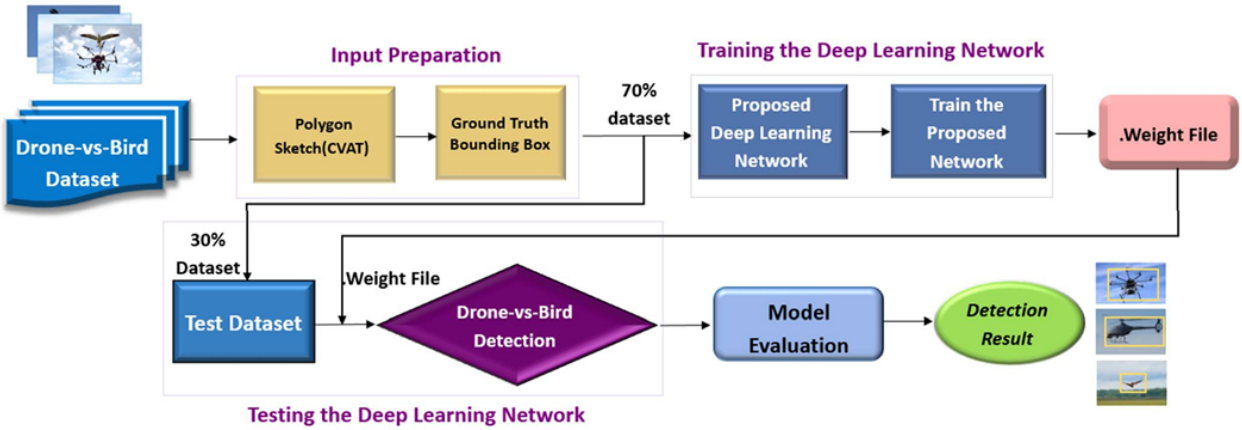
- Mask R-CNN is simple to train
- Mask R-CNN outperforms all existing single-model entries on every task.
- The method is very efficient and adds only a small overhead to Faster R-CNN (Mask-RCNN can run at 5fps).



Mask R-CNN Architecture

METHODOLOGY

Due to the challenges in drone detection and recognition such as crowded background, close resemblance to birds, smaller size of drones, longer distance, and lighting problems in the image, in this study, a deep learning-based method is proposed. After comprehensively understanding the available object detection algorithms, we concluded that YOLOv5 would give the best trade-off between detection speed and detection accuracy. The proposed drone detection and recognition process consists of four main steps. The first step is to prepare the data properly as the input of the proposed architecture. The second step is the network training phase which is implemented to detect and recognize drones and birds. Then, in the third step, the trained model is tested using a large variety of drone and bird datasets. Finally, the performance of the model is evaluated, and the detection and recognition process is performed on the input test data.



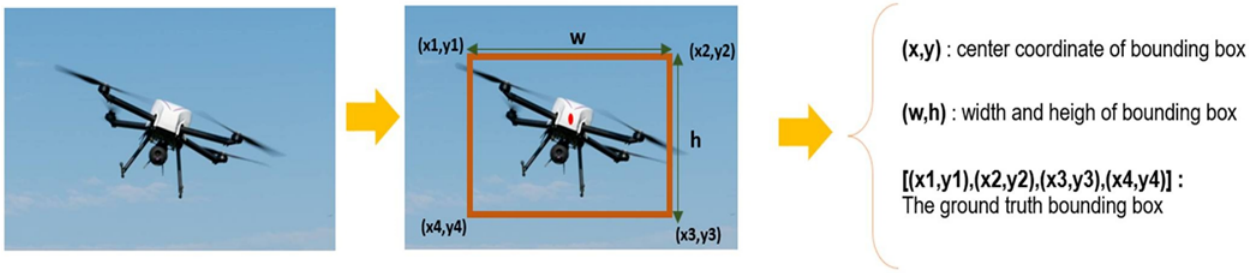
Input Preparation

- In order to train the network, a set of drone and bird visible images are prepared to be fed into the proposed network. The drone dataset used for training includes more than 800 drone and bird images. In total, 70% of the images are used for training and the rest for validation.
- Preparation of the input data involves drawing the ground truth bounding box around drones and birds in the images. This process is known as Data labelling. For this purpose, we used a tool called 'MakeSense.AI'. After the labelling of all the images is complete, a .txt file with the same name is created for each image file in the same directory. Each .txt file contains the annotations for the corresponding image file, that is object class number, object coordinates, height and width.

  `Annotation syntax in YOLOv5: <object-class> <x> <y> <width> <height>`

  For each object, a new line is created. In our case , we have two objects-(i) Drone and (ii) Bird. All the drones have been given an object class number '0' and all the birds have been given an object class number '1'. So let's say an image has 3 drones and 2 birds, the .txt file of the corresponding image will have the following annotation :

```
0 0.703659 0.322439 0.334146 0.573659
0 0.525610 0.543360 0.597561 0.615718
0 0.511364 0.457386 0.971591 0.528409
1 0.335366 0.314602 0.236585 0.234724
1 0.479268 0.443257 0.134146 0.106164
```



Ground truth bounding box sketch.

Training YOLOv5 on the prepared dataset

- First the entire dataset containing the images and labels are compressed in a zip file.This entire file is uploaded in the **google colab** file containing the code to train the Yolov5 model.The link to this colab file is available in the official github page for Yolov5.
- Then we executed the following code to install dependencies required to run this model.

```
!git clone https://github.com/ultralytics/yolov5  # clone
%cd yolov5
%pip install -qr requirements.txt  # install

import torch
import utils
display = utils.notebook_init()  # checks
```

- Next, we uploaded the zip file and download the coco128.yaml file from the data folder of the Yolov5 folder.

- Modify the coco128.yaml file to include the number and name of the classes and the appropriate file paths where the train and val folders are stored in the dataset. We set the number of classes to 2 and the class names as ‘Drone’ and ‘Bird’. After that, we uploaded the file back to the data section of the Yolov5 folder.

- After that, we unzipped the zip file using this command:

```
!unzip -q ../file_name.zip -d ../
```

- Then, we trained the model for the required number of epochs and batch size using this command:

```
!python train.py --img 640 --batch 16 --epochs 60 --data
file_name.yaml --weights yolov5s.pt --cache
```

- After the training is done, we uploaded a test image/video and tested it using this command(use the appropriate file extension):

```
!python detect.py --weights runs/train/exp2/weights/best.pt --img
640 --conf 0.25 --source ../file_name.jpg
```

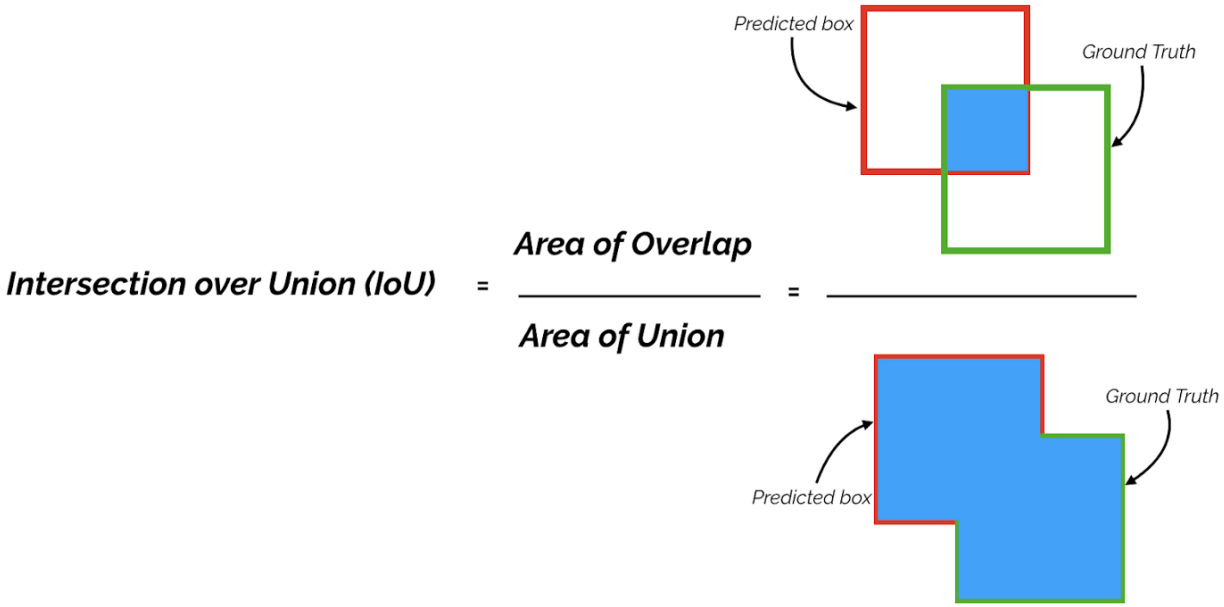
Testing the trained YOLOv5 model

To test the capabilities of the trained YOLOv5 model in the detection and recognition of drones and to distinguish drones from birds in visible imagery, the generated weight file, which is the result of the training stage, is applied. The proposed technique also uses the non-maximum suppression (NMS) method to select the best bounding box containing the drone or bird from several predicted bounding boxes. This method is used to remove possible bounding boxes and select the best bounding box that contains the drone or bird. Finally, the final bounding box containing the target objects and the output parameters of the bounding box are presented.

Evaluation of the model

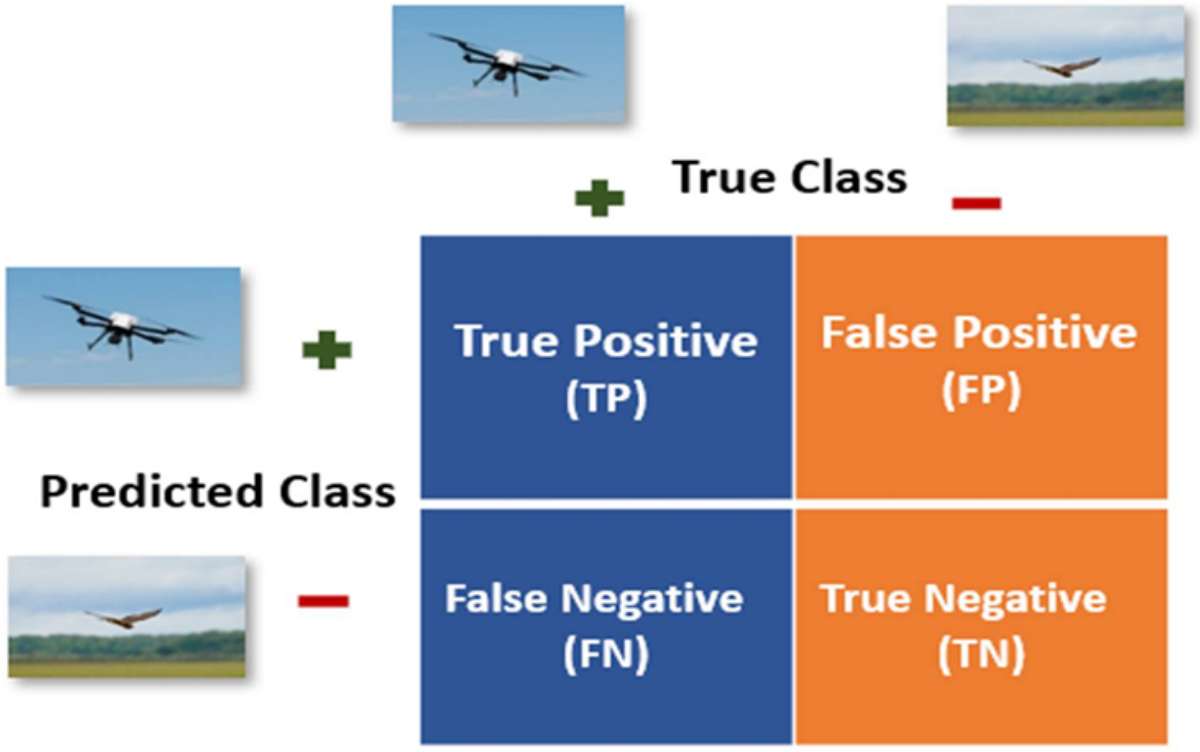
To evaluate the potential of the proposed method, the IoU, precision, mAP, recall, accuracy, and F1-score are used. We believe this evaluation strategy will give us a better understanding of how the model works.

- **Intersection over Union:** This evaluation metric means the degree of overlap between the predicted bounding box and the ground truth bounding box. In this project, a threshold of 0.65 was used to classify the input data. This means that if the IoU value is more than 0.65, the classification is True Positive (TP) and otherwise False Positive (FP). Using the number of these values, a complexity matrix was formed, and the rest of the evaluation metrics were calculated using it.



True positive (TP)
A test result that correctly indicates the presence of a condition or characteristic
True negative (TN)
A test result that correctly indicates the absence of a condition or characteristic
False positive (FP)
A test result which wrongly indicates that a particular condition or attribute is present
False negative(FN)
A test result which wrongly indicates that a particular condition or attribute is absent

- **Confusion matrix:** This is a matrix of size n × n (n = number of classes) to show how accurate the model works . The columns of this matrix represent the true class of intended objects, which in this case includes two classes- drones and birds. On the other hand, the rows of this matrix represent the predicted classes by the proposed YOLOv5 model. Since this study involves two classes, the confusion matrix is a 2x2 matrix. The positive class is related to drones, and the negative class is related to birds. Precision, recall, F1-score, and accuracy can be calculated using FN, TN, TP, and FP values.



Sample confusion matrix in the proposed method

- **Precision:** This means that among the inputs whose class is predicted to be positive, what percentage of them are actually positive class members . According to Equation below, the value of this metric is between zero and one. Precision is calculated separately for each of the classes. In this study, precision is defined in each of the drone, and bird classes. For instance, the precision of the drone class means that among all the inputs projected as drones, what percentage are actually drones. Similarly, these criteria are defined for other classes.

Precision = TP/(TP + FP)

- **mAP** is determined by calculating the average precision of the drone and bird classes. In other words, the mAP evaluation metric compares the ground truth bounding box with the predicted bounding box of the targets and calculates a certain value as the score. An increase in this number indicates the more accurate performance of the proposed model in detection and recognition

$$mAP = \frac{1}{n}\sum_{k=1}^{k=n} AP_k$$

$AP_k =$ *the AP of class k*

$n =$ *the number of classes*

- **Recall:** Indicates the percentage of the total data in the positive class, which is predicted to be positive. Similar to the concept of precision, recall is calculated separately for each class. For example, recall in the bird class means that among all the entries that are birds, what percentage of them are correctly detected and recognized as birds.

$$\textbf{Recall} = TP/(TP + FN)$$

- **F1-score** is the harmonic average of recall and precision and is calculated separately for each of the classes . According to the equation below, this measure performs well on unbalanced data because it considers false negative and false positive values .

$$F1\_score = 2TP/(2TP + FP + FN)$$

- **Accuracy** shows the overall performance of the model . Accuracy means the percentage of data the model correctly detects and recognizes as truly positive and negative

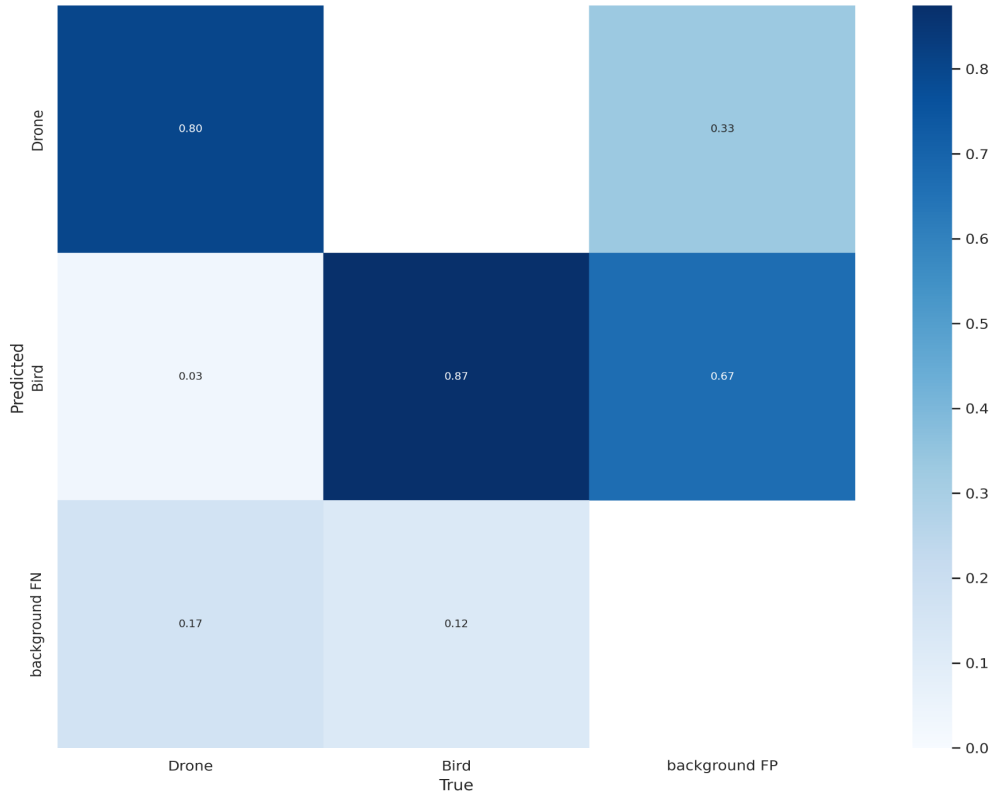
$$\textbf{Accuracy} = (TP + TN)/(TP + TN + FP + FN)$$

RESULTS AND OBSERVATIONS

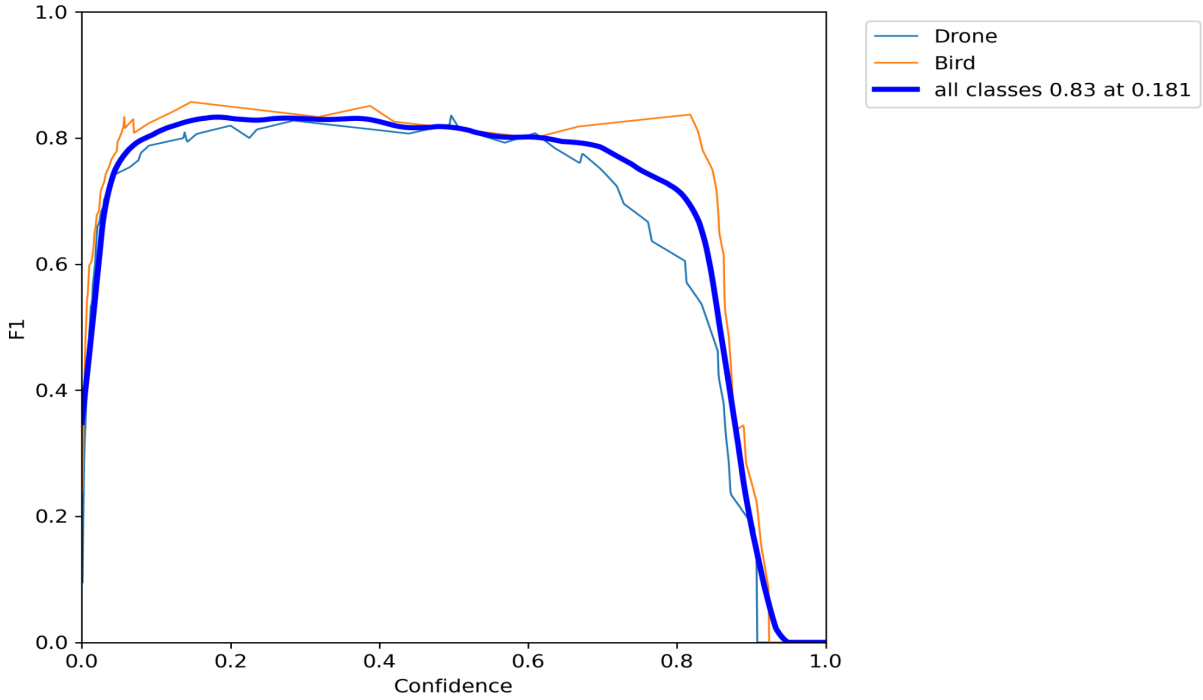
After training the YOLOv5 model on our dataset with a batch size of 16 and 60 epochs and letting it run for around 55 minutes, we got the following results :

Class	Images	Labels	Precision	Recall	mAP@0.5
All	834	1752	0.822	0.845	0.872
Drone	834	606	0.806	0.829	0.852
Bird	834	1146	0.838	0.861	0.892

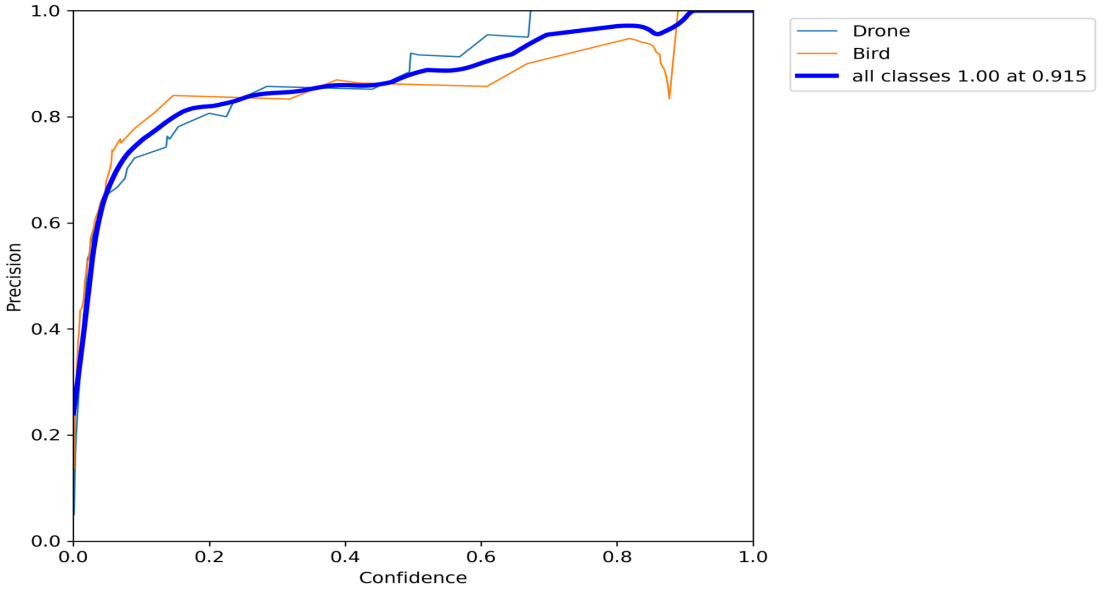
Confusion matrix:



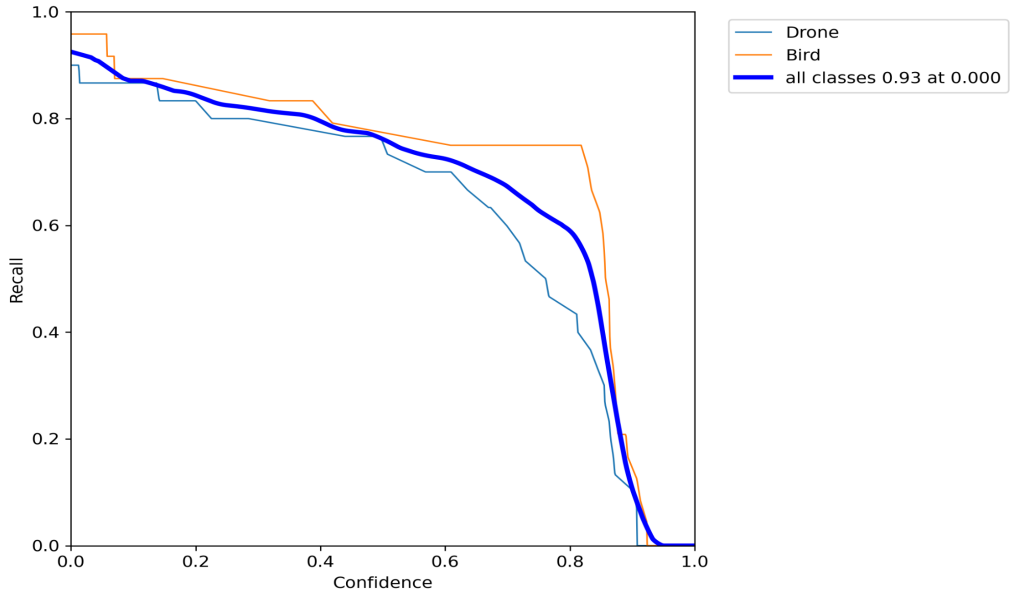
F1- Score Curve:



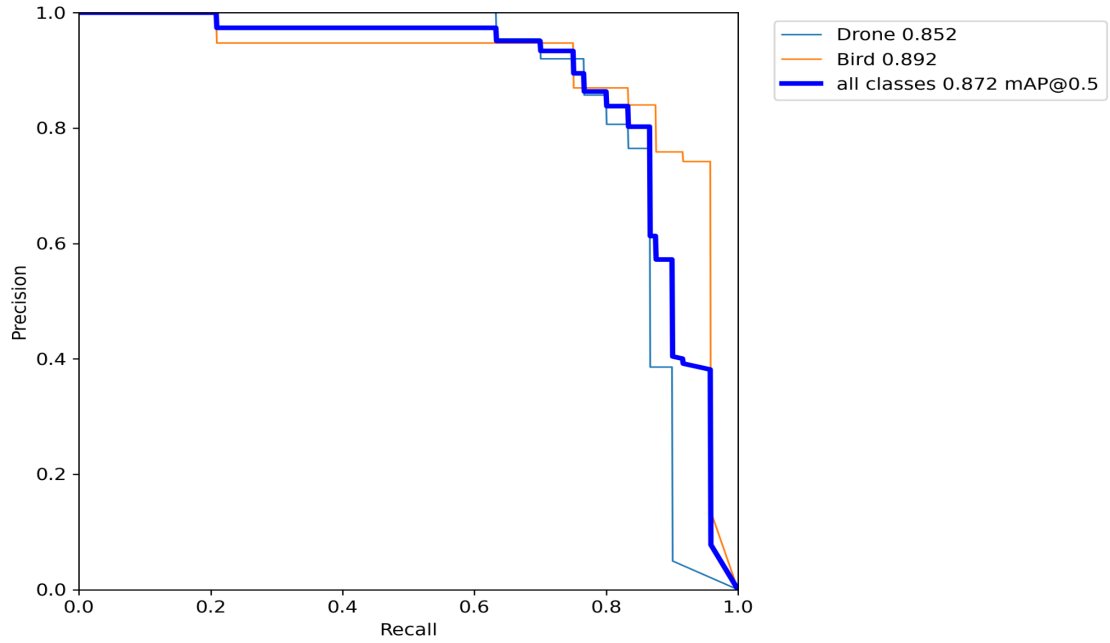
Precision Curve:



Recall Curve:



Precision-Recall Curve:

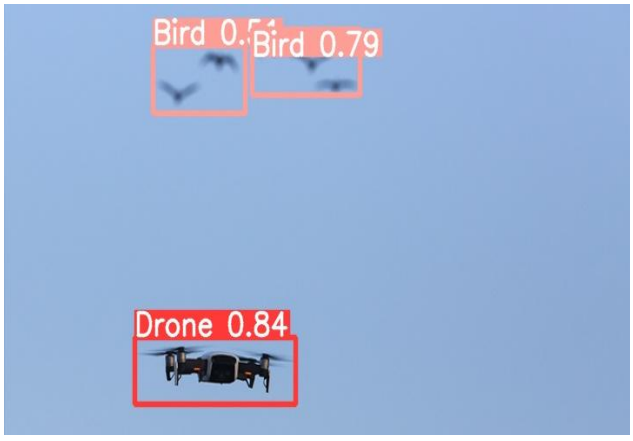


As mentioned earlier in the report, drone detection and recognition always face challenges such as the inability to isolate the background, crowded backgrounds, lighting issues within the image, and the presence of occluded areas. In addition to these, the small size of the drone and its far distance often causes it to be confused with closely related objects such as birds. This results in the reduced accuracy of the diagnosis. The proposed algorithm, YOLOv5 can overcome a variety of challenges in drone detection and recognition, such as detecting drones in crowded backgrounds and distinguishing between birds and drones even at longer ranges.

To evaluate the performance of our model, we chose a few images in which we felt accurate detection might be a challenge. Below are the output images after feeding them to our trained YOLOv5 model. We believe that this will adequately prove our model evaluation in addressing the aforementioned challenges.

- Challenge - Confusion with birds

  Output images:

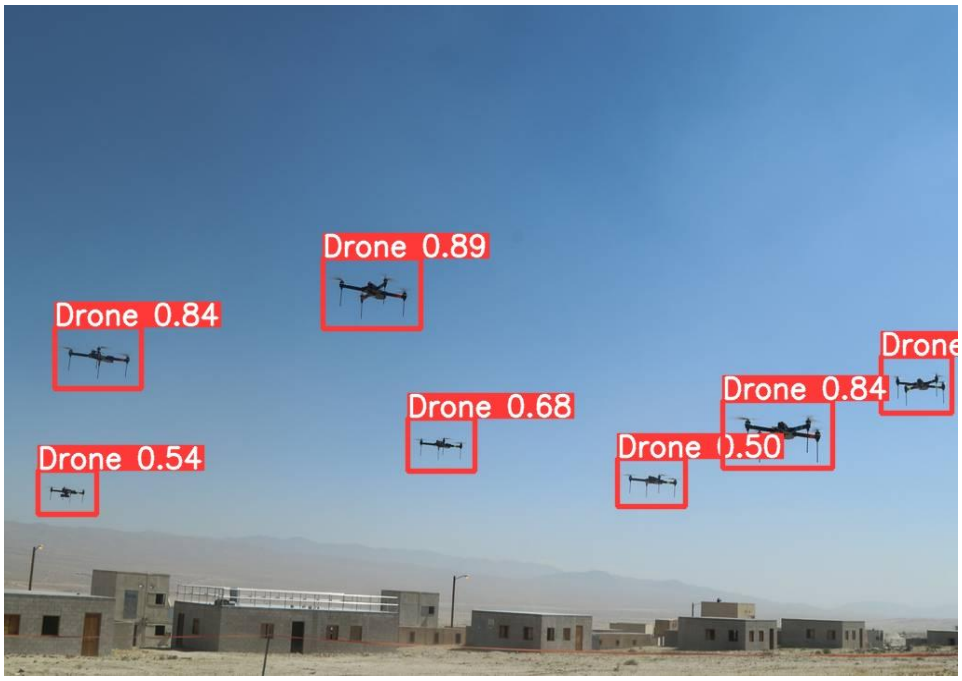


- Challenge - Crowded backgrounds

  Output images:

- Challenge - Small drone size
  Output images:

- Challenge - Lack of scalability
  Output images:



CONCLUSION

Due to the emergence and extensive application of drones and the security threats associated with their presence in sensitive locations such as airports, drone detection and recognition has attracted much attention. Due to the close resemblance of drones and birds in the sky, including their high speed and problems such as crowded backgrounds, the presence of hidden areas, lighting problems in the images, and the small size of drones at long distances, this paper proposes a deep learning-based approach where we used the YOLOv5 algorithm (which according to us, after comprehensively understanding various object detection algorithms, gave the best trade-off between a false prediction and a miss)for detecting and recognizing drones and birds to solve the problems caused by their unauthorised existence.

In this study, drone and bird images were extracted from videos and images. Around 4000 images were collected and manually annotated. The training, testing, and evaluation of the model were performed on the collected dataset and was implemented on **Google Colab**. From this model, we were able to get an overall mAP of 87.2%. Although this is quite satisfactory, the model can further be improved by feeding more images(around 20000 or higher), which we weren't able to do because of GPU constraints in our laptop. However, the performance and evaluation of our current model are quite good with healthy Precision, Recall and mAP values. We believe this will be sufficient to tackle most of the potential challenges involved in drone detection.

BIBLIOGRAPHY

https://ipsjcva.springeropen.com/articles/10.1186/s41074-019-0059-x

https://neptune.ai/blog/object-detection-with-yolo-hands-on-tutorial

https://learnopencv.com/custom-object-detection-training-using-yolov5/

https://viso.ai/deep-learning/object-detection/

https://towardsdatascience.com/how-to-train-a-custom-object-detection-model-with-yolo-v5- 917e9ce13208

https://www.sciencedirect.com/science/article/pii/S2666307421000267

https://github.com/ultralytics/yolov5