Modified YOLOv4 for real-time Coconut Trees Detection from an Unmanned Aerial Vehicle

Kshitij Nair Nair
VIT-AP Campus
Sibi Chakkaravarthy S (sb.sibi@mitindia.edu)
VIT-AP Campus
Ramya Krishna Dhulipalla
VIT-AP Campus
Suresh Chandra Satapathy
KIITS University: Kalinga Institute of Industrial Technology
Abhrankash Kanungo
APSAC
Kannan E
APSAC
Prasada Rao G.
APSAC
Tata Babu Ch
APSAC

Research Article

Keywords: object detection, coconut tree, aerial detection, unmanned aerial vehicle detection, retraining

Posted Date: August 20th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-826223/v1

Modified YOLOv4 for real-time Coconut Trees Detection from an Unmanned Aerial Vehicle

[1]Kshitij Nair, [1*]Sibi Chakkaravarthy Sethuraman, [1]Dhulipalla Ramya Krishna, [2]Suresh Chandra Satapathy, [3]Abhrankash Kanungo, [3]R. Kannan, [3]G. Prasada Rao, [3]Ch. Tata Babu
[1]VIT-AP University, Andhra Pradesh, India
[2]KIIT, Bhubaneshwar
[3]Andhra Pradesh Space Applications Centre (APSAC), ITE & C Dept., Govt. of Andhra Pradesh

Abstract- In this paper, a novel system is proposed to detect coconut trees from the images taken on an unmanned aerial vehicle (UAV). We propose a model based on the YOLOv4 Detector (CSPDarknet53) to achieve accurate prediction and fast speeds that enable real-time detection and object localization for a single object class-coconut tree. The pipeline is trained on two separate datasets of images from camera embedded UAV and images from multiple sources are fed into the multi-scale detector to predict bounding boxes and labels. A simple procedure is also proposed to enable detection on a larger scale, whereby the bigger image can be cropped into multiple single images and then fed into the detector. This model is compared statistically with the other state-of-the-art deep-learning models for object detection. After field studies and experiments on the images shot via a UAV (drone), it is proved that this system can efficiently and accurately detect coconut trees on a field at a speed of 15 frames per second when trained on 500 aerial images of the coconut trees.

Keywords- object detection, coconut tree, aerial detection, unmanned aerial vehicle detection, retraining

I. Introduction

The ability of a computer to detect, or predict, the presence of instances of semantic objects belonging to a certain class from a digital video or image forms the core of the field of computer vision. This subject has gained a massive momentum in the last two decades; CCTV cameras in the house, smartphone cameras that detect object borders on-the-go, cars that can drive themselves without supervision are advancements that were made possible with the vast and ever-expanding research being done in this field.

After the development of CNNs, they were implemented for object detection but suffered from extremely high detection times owing to the time-consuming region search algorithm to look for the Region of Interest (ROI). Regional-based Convolutional Networks or RCNN solved this by using a faster selective approach for the purpose of extracting just 2000 regions from the input image as region proposals which were then fed to the CNN, instead of manually extracting all the ROIs, making this new implementation much lighter and faster than the original approach, but still not capacitating real-time implementations. Demanding for a faster approach, Fast-RNN was developed which fed the input image directly to the CNN layer rather than first extracting the ROI from regional proposals. The CNN generates a convolutional feature map that identifies the region of proposals and then warps them into squares. It works faster at approximately 2.3 seconds per image as compared with 49 seconds taken by R-CNN, because instead of feeding 2000 regions into the convolutional neural network, the operation is performed only once per image giving a feature map for object data extraction. The information of essence is here that in Fast-RNN, the in the 2.7 seconds, 2.3 seconds is taken for the entire fully-connected network detection and only 0.32 seconds excluding the region proposal network processing time, implying that this region locator bottlenecks a rather fast performance of the R-CNN network.

A minor but significant solution to this problem was put forward in Faster R-CNN that removed the selective search algorithm completely and instead used a separate Regional Proposal Network that takes the feature map as the input, thereby reducing the total detection time to a very efficient and fast 0.2 seconds per image. The sole

drawback of all these algorithms was that they used set “regions” to localize the object within the image and never looked at the complete image.

One of the earliest models that brought a breakthrough in this category was YOLOv1[1], which was inspired from GoogleNet, which though had a trade-off between speeds and detection accuracy; the possible drawbacks were outweighed by how astonishingly fast this model was, and its latest release continues to be so. You Only Look Once, or YOLO, proposed a single convolutional network for predicting both the bounding boxes as well as classification of these boxes or simply, class prediction.

This was made possible by realizing that the complex task of detection can be constructed as a regression problem while the classifier can be optimized in a single framework with an end-to-end network, designed to perform classification and bounding box regression at the same time. This simple architecture is what separates YOLO models 9000 and v3 from the other well-known complex frameworks like R-CNN and RetinaNet.

The process of localization of these bounding boxes spatially on the input images is where YOLO struggles. The YOLO model is sufficiently accurate and fast at 45fps on a Titan X GPU with an F-measure at 67%. The three-scale model of YOLO also allows detection at various scales, and is better at recognizing smaller or larger objects from the trained data set. The region proposal network also enables the model to learn general features of an object that helps generalize the detection algorithm, detecting objects of the same class even with drastic differences in textures, angles or even surrounding lighting, making it a very robust and reliable model for object detection. Combined with its fast speed which is comparable to Faster R-CNN, it makes YOLO the ideal model to be used in real-time applications for object detection.

A. Contributions in this paper

This paper explores the use of image detection to detect coconut trees from aerial images. We use the deep-learning model YOLOv4, implemented in [3] and employ the concept of retraining and fine-tuning the convolutional layers to train a model that is accurate and speedy. Our major contribution is to solve the problem of accurately detecting objects from UAV images. We do not interfere with the RGB properties of the input image as done by comparing boosting cascades, neither take a separative approach in splitting the box into foreground and background elements as proposed in [4].

We conduct the experiments on a dataset of 500 images taken manually using DJI Inspire 1 Drone with a downward pointing camera. Since the small dataset doesn’t provide much variation in the object type, a high accuracy is achievable through retraining and fine tuning. The outcome and field tests are further done on images with different backgrounds and the trade-off is discussed.

B. Novelty and Significance of the proposed model

In this paper, we provide a retrained deep-learning model based on the YOLOv4 [3] detector and achieve speeds of 16fps with a GPU achieving an accuracy of 95.65%. We further find that the trained model can identify coconut trees in images that have difference in lighting conditions and backgrounds, accurately and at the same high speeds. This generalizability has been a common issue in other popular CNN architectures such as InceptionResnet V2 CNN, VGG16 and ResNet50 [a, b, c].

The motivation for this paper comes from the simple problem of being able to count coconut trees in any given area and to do so automatically with the help of a drone camera and reduce the manpower required for the job. This concept can be applied in multiple spaces. Using this for connected surveillance cameras in parking lots can help a car to quickly find out an empty spot in a crowded setting, especially with the advancements being made in the field of automatic cars, this is another branch of technology that would expand the area of reach for artificial intelligence in future smart vehicles.

UAV drones with cameras are being used in disaster ridden areas to assess damage caused and conduct reconnaissance about the area. A model with high precision such as ours can be deployed to locate humans and animals in such areas without compromising other life by sending search parties. It ensures maximum recovery with minimum human involvement.

The paper is further divided into sections discussing the previously done work in the field of aerial image detection where the concepts of retraining and fine-tuning are discussed, the proposed methodology where the YOLO architecture is extensively explored, the experimental setup and our final conclusions.

II. Literature Review

A. Object detection on a mobile UAV

Image sensing and object detection from a camera plays a key role in a variety of modern problems such as an unmanned aerial driving system, intelligent security monitoring, Patrol Robot based monitoring. The field of computer vision demands a sufficiently accurate target detection system for fulfilling further intellectual applications like object classification, action detection and more, which has attracted a large number of researches.

Recent publications focus on ground objects (pedestrians and vehicles) from a moving, ground-based camera or robot. An RGB-D based multi person detection and real-time tracking system is proposed in [5], which combines RGB-D visual odometry estimation and ROI processing using a Kinect sensor. A two-stream architecture of convolutional networks is proposed in [6], that extracts ROIs by a retrained LeNet-5 model [7]. For the purpose of aerial image detection, unmanned aerial vehicles (UVAs) are often used. Multi-class object detection (cars, pedestrians and trucks) from an UAV, under as low as 67ms is proposed in [8]. In [8], a retrained CNN-based model is proposed for running person detection at a rate of 12fps without a GPU. Many researchers exploring this field have contributed to civil engineering methods for the purposes of land exploration, survey and inspection. A building detection algorithm is proposed in [10] that uses images from different angles to reconstruct buildings. A modified CNN-based model is proposed in [11] to detect objects from an aerial image.

B. Various CNN models for object detection

The very fundamental requirement of computer vision is to perform object detection on input data to produce meaningful results. In recent experiments [12, 13] CNN models were found to be more time efficient and accurate as compared to other models. The pipeline first extracts such as HOG (Histogram of Oriented Gradients), LBP (Local Binary Pattern) and SIFT (scale-invariant feature transform) from the input data or in certain implementations, extracts self-learning features and then uses classifiers to identify the object.

Table 1: Results of recent detection models on MS-COCO Dataset [3]

Detectors	**mAP-50**	**Time (*ms*)**
SSD513	50.4	125
FPN + Faster RCNN	59.1	172
RetinaNet-101-800	57.5	198
YOLOv4	64.9	23

In Table 1, a comparison of top object CNN-based object detection frameworks is drawn with mAP (mean average precision) score at IOU 0.5 and time of inference (in milliseconds). They are briefly explained as follows, for pointing out some pros and cons:

Single Shot MultiBox Detector (SSD).

In this paper [14], the tasks of localization and classification are done in a single pass of the data through the network. The technique for bounding box regression is named MultiBox. A "network surgery" adds feature map discretization, the matching shape adjustment and non-maximal suppression the structural VGG16 network [15] that forms the base of this model. SSDMultiBox has a higher accuracy than Faster RCNN and is faster than YOLO, but it does not work well on small objects.

FPN + Faster RCNN. In this model [16], the bottom layers of the multiple feature map layers are considered, unlike in the previous framework. This generates better anchor-based proposals than the selective search method to find out the region proposal used in the Faster-RCNN model without Feature Pyramid Network (FPN). FPN is a feature extractor designed for generating a feature map pyramid with accuracy and speed in mind. Though FRCNN is highly accurate, it can be very slow and can only process at 7 FPS with a GPU, since this framework is a region-based object detector and not one-shot.

RetinaNet (Focal Loss). RetinaNet is built on the backbone of FPN and ResNet and introduces focal loss, which is a modified loss function with a modulating factor (1-pt) γ, which is down weighing the loss of easy samples to focus on the hard samples, thus eliminating negative feature spaces from clouding the detector. A newer adaptive focal loss modification makes the model even more accurate but it cannot be used where faster real-time applications are required due to its slow detection speed.

YOLOv4. YOLOv4 is the latest iteration of YOLO [1], which is a single, one-shot, end-to-end model consisting of a single convolutional network integrated with feature extraction, bounding box prediction, non-maximal suppression and contextual reasoning that predicts the bounding boxes and the class probabilities for these boxes. YOLO9000 [17], at the time of its release outperformed almost every framework with higher accuracy to speed ratio. YOLOv4 offers an even trade-off between accuracy and speed, being the fastest method for real-time analysis.

C. Studies done for Aerial Tree Detections

Past research has explored the objective of detecting trees in any giver region using a UAV for the purposes of recon or just curious exploration. In [18], a more classical approach, an aggregate channel features (ACF) based approach is used by adding colour inputs, gradient filters, Gabor wavelets, etc. In the same paper, a deep learning model based on YOLOv2 fails as it is unable to cope with a dense-packed scene. At optimal settings, the ACF model achieves a 90% recall with 96% precision. A study [4] proposes a framework with two classes- coconut tree and background. Using a combination of ACF model + TensorFlow, GoogleNet model and CNN Inception V3, it achieves a precision of 71% at a recall of 93% and takes 90 minutes for a 10000x10000 image. [19] proposes a proprietary deep learning CNN model with more anchor boxes at smaller scales (28x28, 14x14, 7x7) and achieves mAP of 81 at 0.5 IoU.

We thus concluded that YOLOv4 would be most suitable model for the purpose of real-time aerial detection due to the following reasons:

- As shown in Table 1, YOLOv4 has the fastest detector when compared to all the present state-of-the-art models.
- The updated feature extraction architecture present in YOLOv4, CSPDarknet53, gives YOLOv4 a 10% increase in accuracy at mAP-50 and an average of 12% faster speeds than its previous version YOLOv3.
- The detector provides detections at three scales – small, medium and large, for all the objects present in the scene. This works in our favour as different coconut trees from the same angle may have different height and in-turn, different structure from the top.
- The learning model is general in nature [2], which makes it possible for the detector to detect the object in different backgrounds and lighting conditions, maintaining the same accuracy.

D. Fine-tuning and retraining the model

In this paper, we explore how fine-tuning or modifying the parameters of the developed model help attain a more accurate target recognition capability and apply these methods to YOLOv4 for target detection as single pass networks are much faster than their sliding-window counterparts. Retraining a model involves feeding into this fine-tuned model, a newer set of data, for a specific target object with random or pre-training weights. The data space is defined as the data set over which the model is trained after performing modification operations over the dataset. The purpose of fine-tuning is to expand the target data space and/or making adjustments to the model and retraining is done to redefine the data space. Often, fine-tuning done to expand the dataset includes operations such as image transformations (scaling, rotation, etc.) or colour adjustments. Retraining the dataset plays an important role when we take a pre-trained model such as YOLOv4 and further train it on a goal-specific dataset.

As proposed by many researchers, these models are fine-tuned and retrained on specific datasets according to the required output to save training cost and time, giving even better results than the as-is model. In [20], which was the best performing submission for ILSVRC-2013, a smaller receptive windows size and smaller stride was introduced in the first convolutional layer. In VGG16 [21], a smaller 3×3 convolutional filter and pushing the depth of architecture to 16-19 layers can improve the object detection performance. [22] Fine-tuning is performed on Faster-RCNN by rotating the input data space images from 0° to 360° in incremental steps of 45°, making this approach much more efficient than the classic model.

We used a YOLOv4 model based on CSPDarknet53, which is an open-source neural network framework written in C and CUDA that can take advantage of the GPU for processing and training the model. In this paper, we fully learn from YOLOv4 architecture and modify a few underlying parameters in the convolutional layers, and implement a retraining model. In out fine-tuning, we also define how we draw the bounding box around the coconut trees and further discuss how that effects the outcome.

III. Proposed Methodology

YOLOv4 is a model with a generalized method of learning using the shape, size and occurrence of the object would work better. Since the detector in YOLOv4 takes the general features of the objects for which is trained for, it can identify those objects in a wide variety of spaces. We will use this to our advantage and make sure that the labelled images do not contain any negative white space that is not required for the model for training. Retraining with a properly labelled dataset as discussed in Sec. C is what makes this model truly reliable.

We propose a framework as shown in Figure 1. The camera is attached to the drone, looking downwards on the site of deployment to capture the live stream. The proposed YOLOv4 Detector will run on a Raspberry Pi and will be supported with an on-board tiny GPU module to boost the performance of detection. The detected images can be sent back to a home base instantaneously using 5G hardware.

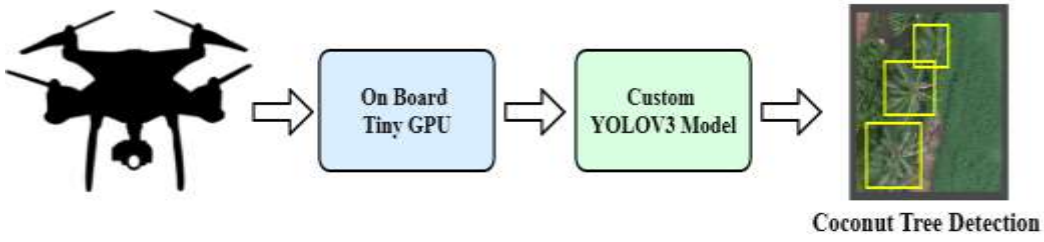

Fig. 1. System Flow Architecture of the proposed framework

A. YOLOv3 and YOLOv4 Architecture and Explanation

Since YOLOv4 is an architectural improvement over YOLOv3, it is imperative to understand what makes YOLOv3 perform detections in real-time before discussing how YOLOv4 builds on top of that architecture.

The YOLOv3 Architecture

YOLOv3 uses a deeper variant of Darknet [1], Darkner-53 which is a custom light-model deep learning framework containing 53 layers. For adding detection capability, it adds 53 more layers, giving a 106 layer fully convolutional underlying architecture. YOLOv3 takes from the concept of ResNet [23], the winner of 2015 ImageNet Classification competition. YOLOv3 introduced residual blocks to add identity mapping and residual mapping without gradient dispersion or disappearance. The identity mapping is implemented by using a short-cut connection or a skip connection. In this new architecture, upsampling is done at one scale to concatenate it with the next layer to preserve the fine-grained features to be able to detect smaller objects. Therefore, YOLO v3's Multi-Scale detector provides a hybrid approach for detecting objects at different scales.

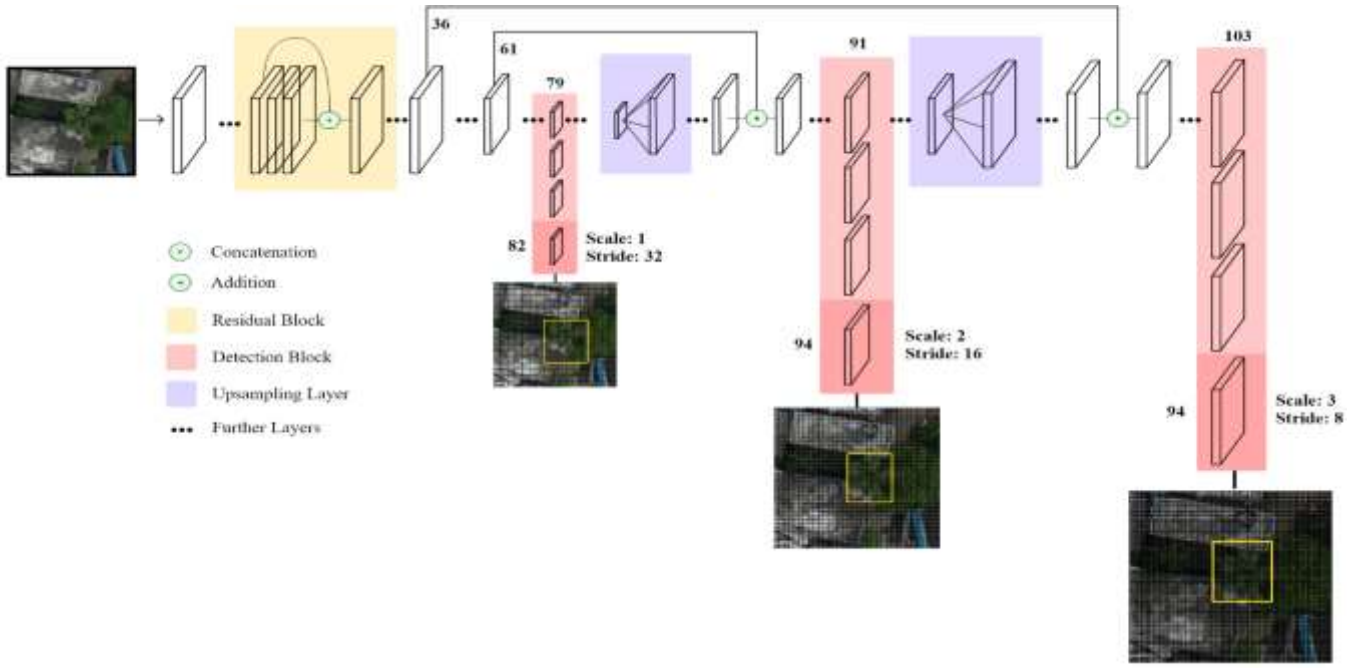

Fig. 2. The custom YOLOv3 Architecture for coconut tree detection

The framework starts by first dividing the input image into GxG grids as shown in Figure 2 and if the center of an object from the ground truth domain falls into a grid cell, then that grid cell is responsible for the detection of that object. Every cell in the grid layout predicts B boxes along with their confidence scores that indicates two things- how confident the model is that the bounding box contains, represented by P(*Object*) an object and how accurately it predicts the given box, given by IOU_{pred}^{truth}, which depicts the Intersection of Union of the prediction bounding box and the ground truth bound box, as shown in Figure 3. The bounding box holds 5 attributes- 4 spatial values x, y, w, h, where (x, y) depict the center of the bounding box, w and h represent the width and the height of the box, respectively and one object confidence p.

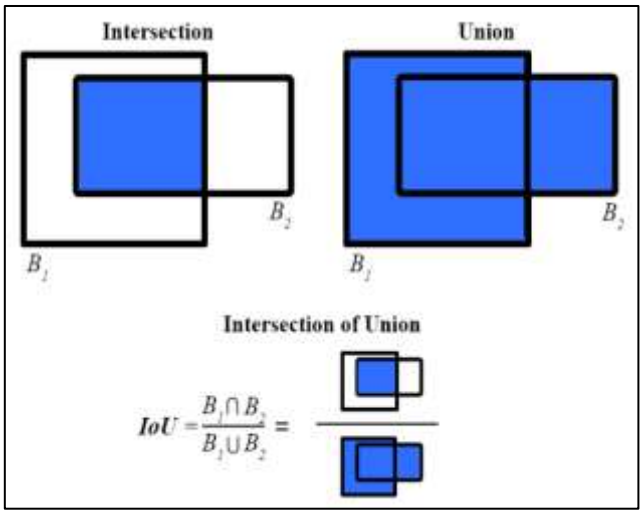

Fig. 3. A visual representation of the calculation of IOU value.

From Figure 3, we get,

$$IoU = \frac{B_1 \cap B_2}{B_1 \cup B_2} \quad (1)$$

Each grid cell also calculates the conditional class probability $P(label\ _i|object)$, given a grid cell contains an object. The final prediction is done on the numerical value calculated by finding the product of the conditional class probability and the confidence of the individual bounding box, which theoretically tells us how accurate the predicted box fits the object, given an object of a class exists inside that box. Mathematically, it gives us (2):

$$P(object) * P(label_i|object) * IOU_{pred}^{truth} = P(label_i) * IOU_{pred}^{truth} \quad (2)$$

The detection is performed by a 1x1 size kernel on the feature maps of the input image at three different sizes. The full shape of the detection kernel is $[1 * 1 * (B * (5 + C))]$, where B is the total number of bounding boxes that a cell on the feature map can predict (3, here), C is the number of classes present (1, here) and 5 is the 4 bounding box attributes and one object confidence value. This gives us a detection kernel of size 1x1x18 to perform the detection on the aerial images.

The input image that we will be feeding into the model will be of 544x544 size. YOLOv3 model downsamples this input at three strides-32, 16 and 8. A stride is defined as the ratio by which the image is downsampled in a training network. The first 81 layers in the model downsamples the 544x544 image to by a stride of 32 giving us a resultant feature map of 17x17. At the 82nd layer, the first detection is made with the 1x1x18 detection kernel defined above, giving us a detection feature map of dimensions 17x17x18. The feature map from layer 79 undergoes more convolutional layers before being upsampled by 2x to 34x34x. The feature map vector from layer 61 is then depth contacted with this layer and is subjected to 1x1 convolutional layers to combine the features from layer 61 and the second detection is made at the 94th layer, giving us a feature vector of 34x34x18. The same procedure is carried again, this time the feature map vector from layer 91 is processed by convolutional layers and is then depth concatenated with feature vector from layer 36. A few 1x1 convolutional layers combine the feature information of both the vectors and the final detection is then made at the 106th layer. From this the feature vector is of the shape 68x68x18. This process is depicted graphically in Figure 2.

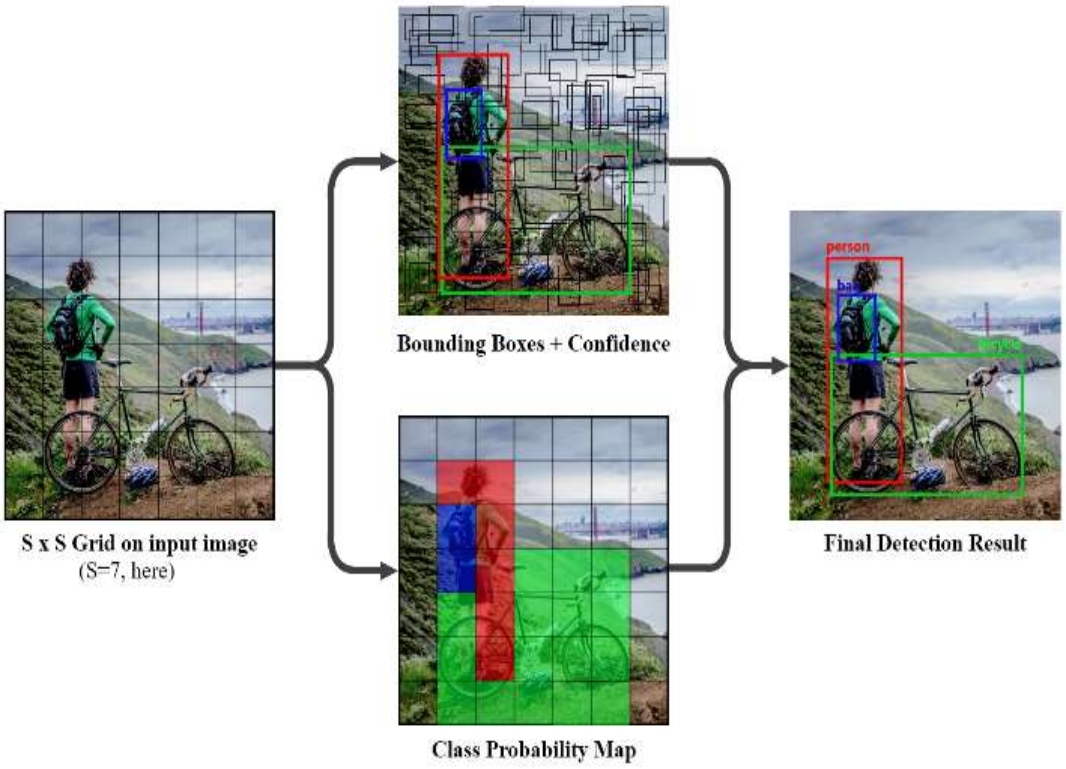

Fig 4. Dividing the input image into GxG grids

In Fig 4, the input is divided into an SxS grid and bounding box confidence predictions and class probabilities are calculated for each grid cell, which are then multiplied to generate the final detection result, encoded as an S x S x (B * 5 + C) tensor.

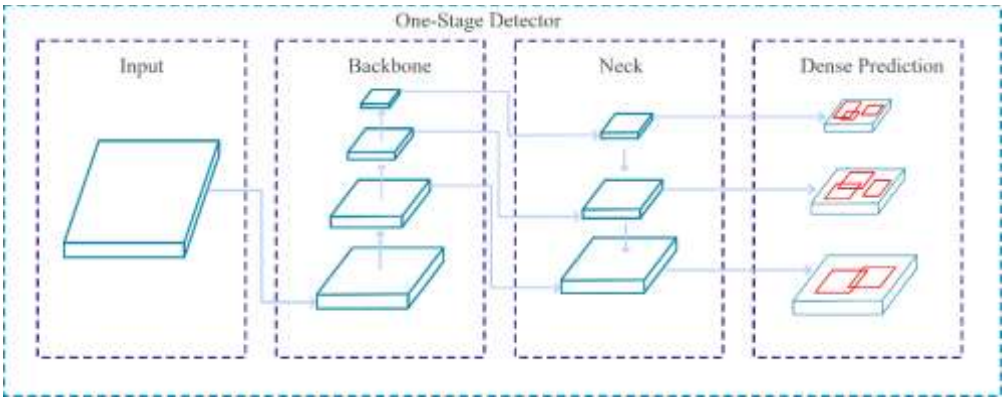

Fig 5. General Image Detector Architecture

The YOLOv4 Architecture

The YOLOv4 model was developed an upgrade for YOLOv3 [3]. Any image detection architecture can be split into 4 distinct blocks- the backbone, the neck, the head as shown in Fig. 5. Here, we have discussed, in general, the methods that are added to YOLOv3 at the backbone and neck, that improve the performance in YOLOv4.

a. Input: This is the image that is provided to the model that goes through many layers of convolution, pooling and other augmentation techniques to generate the final feature map for detection.

b. The Backbone: This is the feature extractor present, which is CSPDarknet53 with all the convolution layers. It stands for Cross-Spatial -Partial connections, which is a method using which the current layer is split in two, one passing through the convolutions while the other one does not and the results are aggregated at the end. This is where all the essential features are extracted from the input image in stages. Techniques like data augmentation (photometric and geometric distortion, MixUp, CutMix, etc.) and other functions such as Mish Activation, which is an activation function that works better than ReLU.

c. The Neck: This is where the feature maps from different stages are collected from the Backbone. This section is very similar to the ResNet architecture present in YOLOv3 but uses a modified path aggregation network (Spatial Pyramid Pooling), better spatial attention module and spatial pyramid pooling which all perform aggregation to improve the performance. This is responsible for the detections that take place at three different scales.

d. The Head: This section is responsible for locating the bounding boxes for location. This process is the same as for YOLOv3 as described in the section above. In addition, YOLOv4 adds broadly two techniques- bag of freebies and a bag of specials. The former helps during training, data augmentation and dropout, and the bag of freebies includes the neck, non-max suppression, etc. that help the performance of the model. The components of the YOLOv4 model are also mentioned in Fig 6.

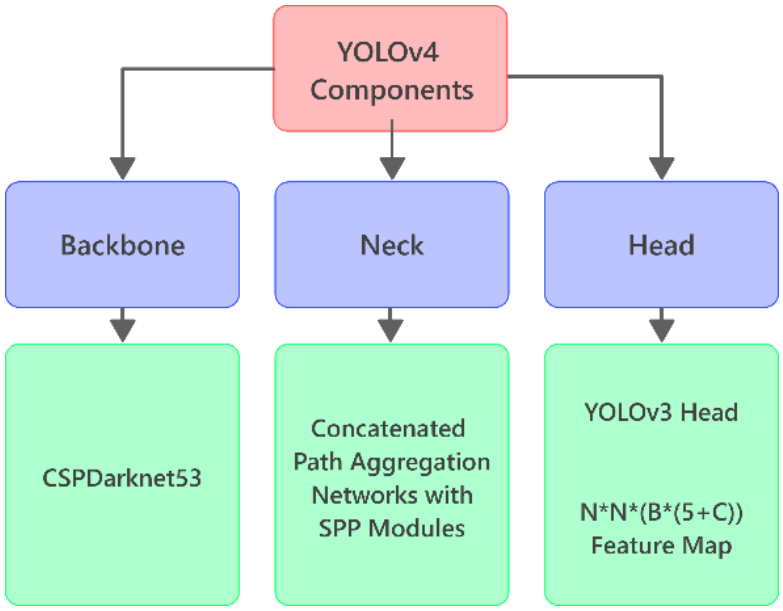

Fig 6. The components of YOLOv4 Architecture

Notably, instead of using Darknet53, YOLOv4 uses CSPDarknet53 in the backbone. At the neck, it performs multiple operations to modify the image for training to better improve the performance from a “bag of freebies” and a “bag of specials” and at the head, it uses the same architecture as YOLOv3 and generates a feature map of the same dimension as well. This results in a performance increase which makes YOLOv4 10% more accurate and 12% faster [3].

Generalizability

The primary reason for choosing the YOLOv4 architecture over other models was the performance of the model in objects that are detectable and observable by the human-eye as belonging to a particular class, but not by even the faster R-CNN object detection models, owing to the previous generation YOLOv3 architecture. This happens due to the fact that models such as the R-CNN use Selective Search Algorithm for bounding box proposals [24]. At this classifier step, the algorithm sees only small regions and needs good proposals, thus learning a pattern of pixels from

the same class to identify images, which gives a significant drop in detection AP [2] when applied to extreme scenarios like artworks. With the same artwork dataset [25], YOLOv3 performs much better that its counterparts because of its ability to model the shape and size of the object as well as capture the relationship between objects and where they commonly appear. This is evidenced from Fig.7. where the YOLOv3 model trained on VOOC 2007 dataset is able to detect objects of the class "person" [2] from People-Art Dataset and Picasso-Dataset. The model is able to not just detect a person at an unorthodox angle (third image, second row) but also from paintings (top row). This ability of the YOLOv3 model to detect objects based on the relationship between the size/shape and occurrence on the object rather than their pixel arrangements is what stands out and remains consistent in YOLOv4.

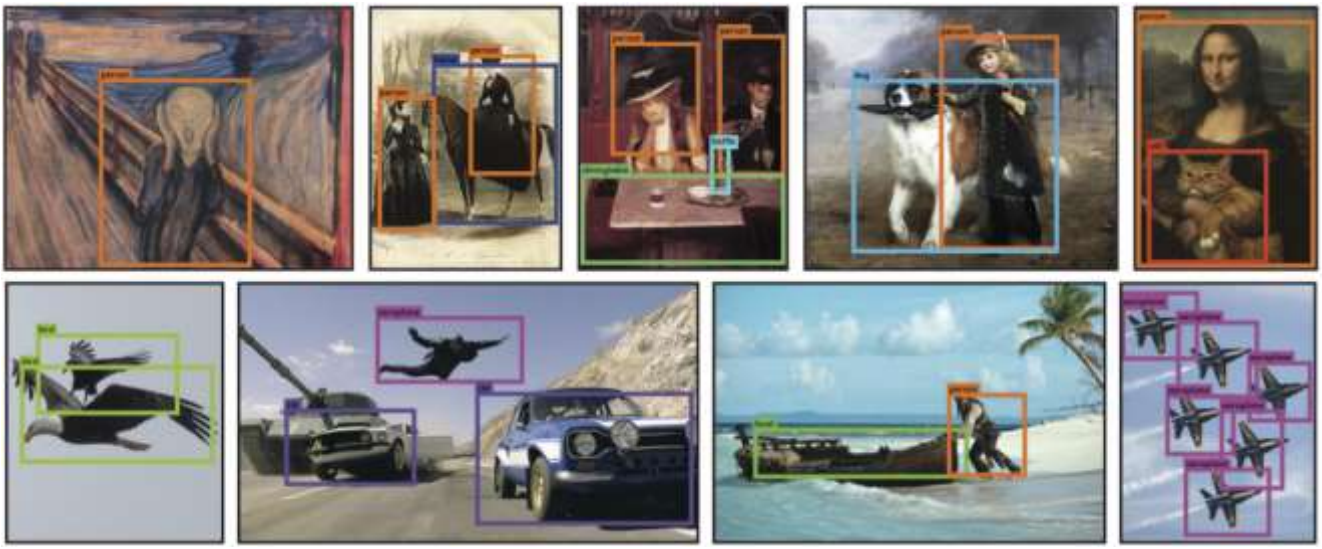

Fig 7. YOLOv3 Generalizability as presented in [2]

B. Collection of Data

For the purpose of training the model, we have two different datasets:

a. 4036 Aerial Coconut Images by Dave Luo with MC and MBB labels as CSV. - 416x416 size images
b. Drone Captured Images, self MC and MBB - 3000x4000 size images

The model was trained on both the models exclusively to see how each of them performs on the given dataset. The predictions made were to test the generalizability of the model in detecting the coconut trees irrespective of their geographical location and persistence with different deformities (due to heavy wind, rain or thunder), in which case the trees are only 30-50% observable from an aerial view, within the same kind. An example of this is shown in Fig. 8.

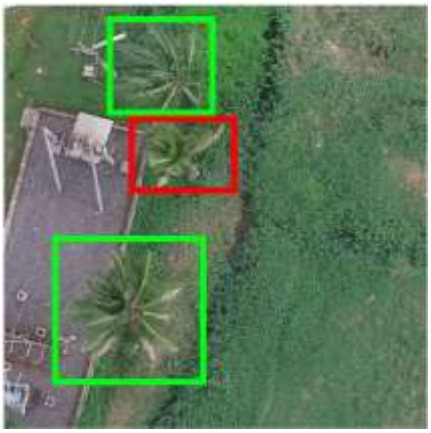

Fig 8. The coconut trees in green box depict a normal and fully observable canopy. The coconut tree in the red box depicts a canopy that is partially destroyed.

C. Pre-Processing of Data

Type (a) dataset was already labelled and provided with a bounded box from Dave Luo [19]. For Type (b), the labelling was done manually for 50 images using LabelImg. The images captured from the drone were of 3000x4000 pixel size and were cropped manually to AxA size (A should be divisible by 32). This was done for two reasons:

a. To ensure the images are compatible with the input for the YOLOv4 pipeline and
b. To optimally minimize the total number of images and reduce the amount of information being given as input to the model during training.

The drone captured images were manually cropped to 544x544 around the sections of the image where the trees were present as shown in Figure 7. We chose A = 544 as it effectively captured multiple coconut trees in a single frame. Another possibility would be to choose 2976x2976 [328x93], which would give us larger pictures and theoretically take less time to train, but in implementation, at each scale, a large number of bounding boxes would be predicted and that would produce further classification issues, as discussed in Sec. VI. We also used multiple drone shots of the same coconut tree to produce various parallax versions of the same tree to increase the efficiency at which the model can identify a coconut tree in an aerial image, even from a certain inclination. The labelling was done with a single class - “coconut”, and we had 500 images from dataset (a) and 490 such images with more than 900 individually labelled coconut trees from dataset (b).

D. Retraining of YOLOv4 Model

YOLOv4 has an upper hand over other detectors not only in terms of speed but also because it introduced detection at multiple scales, which makes it an ideal architecture for our purposes where an aerial shot from a fixed height may contain trees of varied heights resulting in different sizes of adjacent trees. The model is pre-trained with ImageNet 1000-class images. We changed the training set subdivisions to 32 instead of 16 in the original model to further make the training faster. At the end of each YOLO layer we changed the number of filters to 18 in each convolutional layer.

Fig. 9: Dividing 3000x4000 size images into 544x544 size images with the bounding box as shown

IV. Experimental Setup

The training and testing processes were carried out using Google Collaboratory which let us access an AI/ML specialized high performance 1.5GB Tesla T4 Tensor GPU with an Intel(R) Xeon(R) CPU @ 2.20GHz and 13GB with 69GB of disk space.

(a) Coconut tree dataset (b) VITAP-APSAC dataset

Fig 10. Depicting how the individual coconut trees are labelled in the two obtained datasets (a) and (b)

The images used in training the model was done by combining two datasets- (a) was a publicly available dataset from Dave Luo while (b) was obtained from field experiments done using a DJI Matrix 100 – industrial grade drone with Zenmus 10 (downward pointing sensor (camera)), a 64x zoom RGB camera, taken at the site of deployment to compare and realize if there were any major differences within the two datasets. These images were taken from an approximate altitude of 100m~120m above local Mean Sea Level (MSL) with 8cm Ground Sample Distance (GSD). The bounding boxes were manually drawn as presented in Figure 9. Further, Figure 10 gives a visual comparison of the bounding boxes drawn in the two datasets. The YOLOv4 model was modified as mentioned in Sec. III-D.

V. Results and Discussion

A. Outcome and Improvement

With our proposed methodology, we were able to train the YOLOv4 model on our custom images, but the model could only reduce the loss value to 0.569, which resulted in an overall inaccurate model, giving accuracy in the range of 40%-60% based on testing multiple random images on the obtained weights, with an overall training accuracy of almost 47% using Area-Under-Curve (AUC) for classification. This accuracy was too low to be implemented into any final product and had to be studied to figure out why the model was failing. Upon inspection we concluded that the low accuracy was a result of how different set (a) and set (b) were, not just in the specific subspecies, which produced more fronds in (a) and lower in (b); also, the bounding boxes drawn on (a) were also inclusive of the surrounding buffer area with overlapping canopy, thus introducing unnecessary whitespace in the bounding box. The custom boxes made for set (b) were drawn with the center of the tree falling approximately in the centre of the box with no whitespace occurring randomly around the coconut trees inside the bounding box. We then changed the training dataset and performed the training with the same parameters on only set (b) images for approximately 4600 iterations in batches of 16, after which the loss value was constant. With the retrained model, we tested out not just images from our own dataset (a and b) but also multiple aerial shots of coconut trees from images that we obtained from the internet, for checking if the model was accurate in the general use case. As seen in Fig. 9(b), the model trained on our set (b) was able to detect trees present in set (a) (Dave Luo Dataset); it was also able to pick out the coconut trees from images that had a different colour dynamic as compared to ours [Fig. 11(b) bottom-left] and well as detect coconut trees in a larger high-resolution image [Fig. 11(b) top-left]. A performance analysis is further done in Section B.

(a) (b)

Fig. 11. Output of the detector after running it on various images

B. Performance Analysis

The model training on Google Colab took an estimated 11.4 hours on the listed specifications. The various calculated metrics of the model are discussed below.

Loss Calculation. The final layer in YOLOv4 uses a linear activation function and uses a sum-squared error in the output of its model. The loss incoming from the bounding box coordinate predictions and confidence predictions are multiplied by a parameter and $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.5$, which increases the loss from the box prediction while reducing the loss from confidence prediction for boxes that do not contain any identified objects. Only one predictor, or bounding box, is responsible for predicting an object which is chosen on the basis of the highest IOU with the

ground truth. The loss function (See Figure 12) in YOLOv4 is a multi-part equation that only penalizes classification error if an object is present in that grid cell and the bounding box coordinate error if that predictor is liable for the ground truth box.

The model, at the end of 4600 iterations, gave an output loss of 0.0514, which implies the general learning and predictive effectiveness of the implemented model, as compared to a loss of 0.569 produced by the prior dataset.

Training and Validation Metrics. The model, after training, yielded a training accuracy of 99.01% at mean average precision (mAP) at 0.50 or 50%. A validation accuracy of 95.65% was achieved at mAP of 0.5 and each unique recall using Area-Under-Curve (AUC) for classification.

F1 Score and Recall. The F1 Score achieved by the model is 0.9379 with a recall value of 92% for all the validation images. The recall value indicates the sensitivity or the true positive rate, which is very accurate for our practical application as the dataset is very niche. The F1 Score conveys the balance between the precision and the recall.

For practical applications based on a UAV system, we also need to consider the speeds at which the detector can detect or more precisely, the rate of prediction in frames per second (FPS or fps) of the model. This allows us to discern whether the model is fast enough to be used in a real-time deployed application. This model can detect objects in an image in approximately 62.77ms, which gives us an FPS rate of 15.93fps or more appropriately 15fps, implying that this detector can run and produce outputs on 15 images in a second with 95.65% precision. Figure 11(a) shows some examples of the images taken from the field for testing. In comparison with the previous work on coconut tree detection done by Dave Luo [19] that achieved an F-1 Score of 0.83, ours was 13% higher, hence implying a more accurate real-life model. The detector not only detects images from the trained sample, but accurately

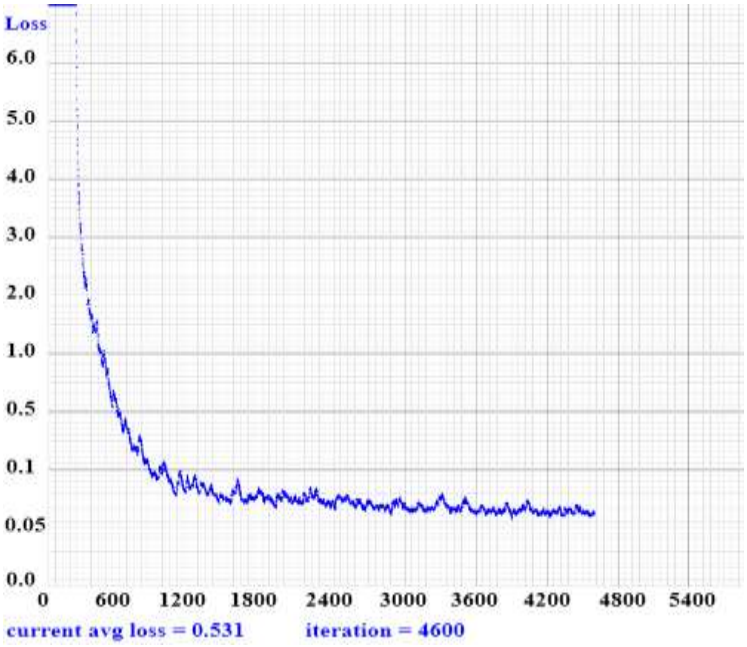

Fig. 12. The loss graph

identifies coconut trees in different background. In the related work conducted by Dave Luo, this was not accounted for as none of the test images present in the dataset had coconut trees of this nature. In Figure 11(a) (top-left) A coconut tree with some percentage of its top destroyed (say in heavy rain or storm) is also detected by the model with 78% accuracy. Figure 11(b) shows the detector detecting the same trees in different images. These images have different lighting conditions, background soil, resolution and are of different sizes, thus further strengthening our hypothesis. YOLO's more generalized method of learning to recognize an object is evident here as it is able to recognize a variety of trees, albeit with lesser accuracy and few false positives. In a similar work done by Dave Luo,

this is not the case, as the entire dataset used for training has the same lighting conditions and so the validation sets as we could find from the source.

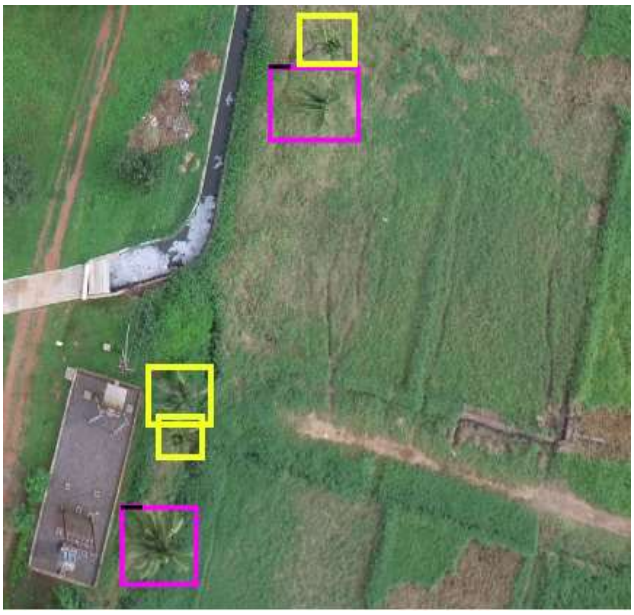

Fig. 13. Limitations of the detector when tested on image of size 750x1000

VI. Comparisons with other Frameworks

Other CNN models for deep learning have a steep disadvantage over YOLOv4 in a key aspect that YOLOv4's pipeline learns the properties of the given ground truth object in a distinctively general form rather than performing more image processing concepts on it [2]. Thus, even an inanimate object belonging to the same class (say a coconut tree toy), will be detected more accurately by the YOLOv4 model as compared to its counterparts like Faster-RCNN or RetinaNet as seen in the case of "person" class in two artwork datasets [2]. Faster-RCNN cannot be deployed for real-time UAV applications as it can provide a maximum detection speed of 6fps only. The only advantage that it has over YOLOv4 is in accurately detecting small-sized objects even in a densely packed scene.

In Table 2, we see that as compared to YOLOv1 + VGG16, the newer YOLOv4 model outperforms it in accuracy but the former is 25% faster at detections. This trade-off is justified as the accuracy of the latter is 44% more accurate. The ACF Model and the model based on CNN Inception V3 are both very accurate but turn out to be very slow while detecting; the former uses 10 minutes and the latter 90 minutes for a 10000x10000 image as both are slower, sliding window detectors. Out model is 18% more accurate and about 60% faster than Dave Luo's aerial coconut detection model.

Limitations of the detector. The multi-scale detector in YOLOv4 though uses a three-scale sampling technique, still fails at times for images that contain objects that are too small for the model as show in Fig. 13. The purple bounding boxes are detected by the trained detector and the yellow boxes are missed positive coconut trees. The kernels and bounding box generated are far too big to accommodate the objects and calculate the confidence in accordance with the ground truth.

Table 2. Comparison of our model with the other available models from a comparable domain of work.

Model	Precision	Image Size (px)	Time (s)
YOLO + VGG16	66.40%	448x448	0.0476
ACF Model [18]	96%	10000x10000	600
CNN Inception V3 Model [4]	93%	10000x10000	5400
Dave Luo Model [19]	81%	224x244	0.153
Dike et al. [9]	86.50%	480x640	0.086
Wu and Zhou Model [8]	-not provided-	1080x640	0.067
Our Model	**96.65%**	**544x544**	**0.062**

VII. Conclusion

In this paper, we propose a modified YOLOv4 algorithm for aerial detection of coconut trees using UAVs. After fine tuning the model by changing the subset division to 16, customizing the input image size to optimize the window of detection and retraining it on a custom dataset, the detector is able to detect the trees in a scene with an accuracy of 96.65% in 0.062s or 62ms which gives the model a speed of 15.93FPS, given that the scene is not densely populated with small-sized objects. Comparing this to the standard YOLOv4 model with no parameter modifications that took over 18 hours to train, it had slightly lesser detection speeds at 0.109s or 9FPS, but with similar accuracy. When this standard model was trained on a dataset without our precise bounding box modification, the training accuracy also fell to 92% and validation accuracy to 89.4%. This is a strong indication that the fine-tuning and retraining on the modified dataset are major contributing factors that improved both detection time as well as accuracy.

When compared to an algorithm of comparable accuracy, the ACF Model, our model is 99.5% faster because of omission of the sliding-window approach and cropping bigger resolution images into smaller blocks for training. The CNN Inception V3 Model which also targeted detection of coconut trees from aerial shots also had similar accuracy, but because of the large image size, and considering the advancements made since then, our model gets the detections in about $1/10^{th}$ of that time; theoretically, if we were to divide the entire 10000x10000 into a size suitable to us (say, 500x500), then those 400 images would take us approximately 25 seconds, not including the time for cropping the individual images. The model proposed by Dike et al was used in detecting running persons (a single class) from drone shots, and keeping that as the common factor, our model was 11.7% more accurate and faster at 0.062s compared to the former mode's 0.086s by 28%. The Wu and Zhou model, which was trained for aerial detection for detection of objects for three classes performs better in terms of real-time detection speeds but the accuracy is defined as "sufficient", which is much too subjective for any authentic comparison. From the experimental results, it is noted that the proposed model gives speeds at which real-time detection is achievable. The model is able to detect true positives even in scenes with low or high contrasts between the coconut tree and the background image with accuracy.

Therefore, in the future, this work is easily adaptable to technologies like 5G equipped UAVs with a Raspberry Pi, a modular on-board. In field implementations, 5G is designed to reach speeds of 10Gbits/s, ideally. A UAV equipped with the capable 5G hardware will thus be able to send these detections at instantaneous speeds or send back the captured video stream to be detected at the server-end in real-time. In both these scenarios, the model will prove to be real-time capable as the transfer latency is only about 7.5% of the total detection time, which is negligible

in real-time. A UAV of this configuration, with the high-accuracy model will find use in multiple applications- such as assessment of disaster ridden areas, helping farmers monitor crops at an inexpensive cost, etc.

Acknowledgement

The authors wish to acknowledge Andhra Pradesh Space Applications Centre (APSAC), ITE & C Dept. Govt. of Andhra Pradesh for extending the support, expertise and sample UAV images for this research.

References

1. J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.
2. Redmon, Joseph & Farhadi, Ali. (2018). YOLOv3: An Incremental Improvement. *arXiv* 1804.02767
3. Bochkovskiy, A., Wang, C., & Liao, H. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* Published. https://arxiv.org/abs/2004.10934
4. M. Zacharova, “Automated Coconut Tree Detection In Aerial Imagery Using Deep Learning”, M.Sc. thesis, Katholieke Universiteit Leuven, Lueven, Belgium, 2017.
5. W. Jia, W. Yi, J. Saniie and E. Oruklu, "3D image reconstruction and human body tracking using stereo vision and Kinect technology," *2012 IEEE International Conference on Electro/Information Technology*, Indianapolis, IN, 2012, pp. 1-4, doi: 10.1109/EIT.2012.6220732.
6. Q. Wu, H. Guo, X. Wu, S. Cai, T. He and W. Feng, "Real-time running detection from a moving camera," *2016 IEEE International Conference on Information and Automation (ICIA)*, Ningbo, 2016, pp. 1370-1375, doi: 10.1109/ICInfA.2016.7832033.
7. Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.
8. Q. Wu and Y. Zhou, "Real-Time Object Detection Based on Unmanned Aerial Vehicle," *2019 IEEE 8th Data Driven Control and Learning Systems Conference (DDCLS)*, Dali, China, 2019, pp. 574-579, doi: 10.1109/DDCLS.2019.8908984.
9. H. U. Dike, Q. Wu, Y. Zhou and G. Liang, "Unmanned Aerial Vehicle (UAV) Based Running Person Detection from a Real-Time Moving Camera*," *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO),* Kuala Lumpur, Malaysia, 2018, pp. 2273-2278, doi: 10.1109/ROBIO.2018.8665167.
10. Fradkin, Maxim & Roux, Michel & Matre, H. (1999). Building Detection From Multiple Views.
11. Radovic, M.; Adarkwa, O.; Wang, Q. Object Recognition in Aerial Images Using Convolutional Neural Networks. *J. Imaging* 2017, *3*, 21.
12. Sharma, Neha & Jain, Vibhor & Mishra, Anju. (2018). An Analysis Of Convolutional Neural Networks For Image Classification. *Procedia Computer Science*. 132. 377-384. 10.1016/j.procs.2018.05.198.
13. Xin, Mingyuan & Wang, Yong. (2019). Research on image classification model based on deep convolution neural network. *EURASIP Journal on Image and Video Processing*. 2019. 10.1186/s13640-019-0417-8.
14. Liu, Wei & Anguelov, Dragomir & Erhan, Dumitru & Szegedy, Christian & Reed, Scott & Fu, Cheng-Yang & Berg, Alexander. (2016). SSD: Single Shot MultiBox Detector. *Proceedings of the European Conference on Computer Vision (ECCV).* 9905. 21-37. 10.1007/978-3-319-46448-0_2.
15. Simonyan, Karen & Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556.
16. T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature Pyramid Networks for Object Detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* Honolulu, HI, 2017, pp. 936-944, doi: 10.1109/CVPR.2017.106.
17. J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* Honolulu, HI, 2017, pp. 6517-6525, doi: 10.1109/CVPR.2017.690.
18. Puttemans, Steven & Van Beeck, Kristof & Goedemé, Toon. (2018). Comparing Boosted Cascades to Deep Learning Architectures for Fast and Robust Coconut Tree Detection in Aerial Images, *International Conference on Computer Vision Theory and Applications (VISAPP),* Portugal. 10.5220/0006571902300241.
19. D. Luo, “Detecting Coconut Trees from the Air with Fast.ai”, March 2018. [Online]. Available: https://forums.fast.ai/t/detecting-coconut-trees-from-the-air-with-fast-ai-notebooks-dataset-available/14194. [Accessed: Mar. 12, 2020]

20. Zeiler M.D., Fergus R. (2014) Visualizing and Understanding Convolutional Networks. In: Fleet D., Pajdla T., Schiele B., Tuytelaars T. (eds) Computer Vision – ECCV 2014. ECCV 2014. *Lecture Notes in Computer Science*, vol 8689. Springer, Cham. https://doi.org/10.1007/978-3-319-10590-1_53
21. Simonyan, Karen & Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* 1409.1556.
22. Q. Wu, H. Quo, X. Wu, Y. Zhou and N. Li, "Fast action localization based on spatio-temporal path search," *2017 IEEE International Conference on Image Processing (ICIP),* Beijing, 2017, pp. 3350-3354*, doi: 10.1109/ICIP.2017.8296903.*
23. K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016,* pp. 770-778, doi: 10.1109/CVPR.2016.90.
24. Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177,* 2018.
25. H. Cai, Q. Wu, T. Corradi, and P. Hall. The crossdepiction problem: Computer vision algorithms for recognising objects in artwork and in photographs. *arXiv preprint* arXiv:1505.00110, 2015.