drones

MDPI

Article

Automated Drone Detection Using YOLOv4

Subroto Singha [1] and Burchan Aydin [2,*]

[1] Department of Computer Science and Information Systems, Texas A&M University-Commerce, 2600 W Neal St., Commerce, TX 75428, USA; ssingha@leomail.tamuc.edu
[2] Department of Engineering and Technology, Texas A&M University-Commerce, 2600 W Neal St., Commerce, TX 75428, USA
* Correspondence: burchan.aydin@tamuc.edu; Tel.: +1-903-886-5174

Abstract: Drones are increasing in popularity and are reaching the public faster than ever before. Consequently, the chances of a drone being misused are multiplying. Automated drone detection is necessary to prevent unauthorized and unwanted drone interventions. In this research, we designed an automated drone detection system using YOLOv4. The model was trained using drone and bird datasets. We then evaluated the trained YOLOv4 model on the testing dataset, using mean average precision (mAP), frames per second (FPS), precision, recall, and F1-score as evaluation parameters. We next collected our own two types of drone videos, performed drone detections, and calculated the FPS to identify the speed of detection at three altitudes. Our methodology showed better performance than what has been found in previous similar studies, achieving a mAP of 74.36%, precision of 0.95, recall of 0.68, and F1-score of 0.79. For video detection, we achieved an FPS of 20.5 on the DJI Phantom III and an FPS of 19.0 on the DJI Mavic Pro.

Keywords: drone detection; YOLOv4; mAP; precision; recall; F1-score; FPS

Citation: Singha, S.; Aydin, B. Automated Drone Detection Using YOLOv4. *Drones* **2021**, *5*, 95. https://doi.org/10.3390/drones5030095

Academic Editors: Andrey V. Savkin and Kooktae Lee

Received: 2 August 2021
Accepted: 9 September 2021
Published: 11 September 2021

1. Introduction

According to the Federal Aviation Administration, as of 13 April 2021, there were 368,508 commercial and 500,601 recreational Unmanned Aircraft Systems (UAS) or drones registered in the US [1]. The drone industry is expanding rapidly. They are growing increasingly more accessible to the public and at cheaper prices [2]. According to their payload capability, drones can be used for various purposes, such as inspection, delivery, monitoring, photography, and among other uses [3]. However, drones can also be misused, generating safety concerns [3]. There is an increasing potential for small drones to be misused, especially by hobbyists, as well as for illegal activities such as drug smuggling, terrorist attacks, or even interfering in emergency services such as fire prevention and disaster response. Drones can also be converted into dangerous weapons by loading them with explosive materials [2]. Examples of drones being used in terrorist attacks can be found in [4,5].

It is important that the illegal use of drones be controlled in order to prevent security breaches and to ensure public safety. However, they are not easy to detect when in the air. Small drones transmit very limited electromagnetic signals, making it very difficult for conventional radar to detect them [2]. Acoustic and radio frequency detection are expensive [6] and cannot deal with the Doppler effect well [6]. Conversely, object detection using deep learning has achieved substantial success due to its high accuracy and available computing power [6]. In fact, the "You Only Look Once" (YOLO) algorithm has surpassed other object detection algorithms such as the Region-Based Convolutional Neural Network (R-CNN) and the Single-Shot Multi-box Detector (SSD) because of its highly precise real-time detection capability [6]. YOLO is superior in terms of both accuracy and speed [7]. Though it has various versions, YOLOv4 is one of the latest, achieving 10% more average precision (AP) on the Microsoft Common Objects in Context (MS COCO) dataset than the

previous version, YOLOv3 [8]. In this research, the YOLOv4 algorithm was used to detect drones because of its real-time detection capability and high level of precision.

2. Background

Deep learning is a state-of-the-art technique that has shown great promise in computer vision and pattern recognition. Techniques based on convolutional neural networks (CNNs) can predict class and bounding boxes by extracting deep high-level features from an object, a process known as object detection. CNNs are useful for performing detection, recognition, and segmentation tasks [2]. They follow layer-based approaches, where the lower layers extract low-level features such as edges, the mid-level layers extract droplet-like structures, and the top-level layers define the object [9]. Deep learning-based object detection techniques can be divided into two types: two stage and one stage. Two-stage object detectors are basically R-CNNs. R-CNNs first use a selective search algorithm to generate a large number of region proposals. Then, a CNN is used for feature extraction from each region proposal. Finally, the R-CNN classifies various classes and defines bounding boxes [10]. To speed up such calculations, Fast R-CNNs [11], Faster R-CNNs [12], and Mask R-CNNs [13] have all been proposed.

To overcome the slowness of R-CNNs, [11] developed an algorithm called the Fast R-CNN. The Fast R-CNN is different from the R-CNN in that it does not split regions into any region proposals; rather, it first applies a CNN and then allocates region estimation to the neural network's property map. For final classification, it uses Softmax rather than a support vector machine (SVM). If a CNN is used only once, the speed of the algorithm increases. To further improve the estimation time, [12] proposed Faster R-CNN. Like Fast R-CNN, it applies the CNN first and then creates a feature map. Faster R-CNN does not conduct any region recommendations (much like Fast R-CNN). Instead, it uses a selective search algorithm, and the rest of the approaches are similar to Fast R-CNN.

Unlike region-based approaches, one-stage techniques look at an image only once. YOLO is a popular example of this kind of approach. It involves a single neural network trained end-to-end, which takes an image as input and directly predicts bounding boxes and class labels. The technique offers lower predictive accuracy (e.g., more localization errors) but operates at 45 to 155 frames per second (FPS), providing a speed-optimized version of the model [14]. Two-stage object detection methods spend comparatively more time on receiving proposals. Thus, one-stage object detectors and SSDs are used in various real-time object detection applications such as in traffic scenes [15], high voltage insulators [16], and airplane detection [17]. The authors of [14] proposed the first version of YOLO, which provides class prediction with bounding boxes and class probabilities. Later, [18] proposed YOLOv2, which outperformed the first version of YOLO in terms of speed and accuracy. To further improve the accuracy, [7] proposed a bigger YOLO network known as YOLOv3, which achieved higher accuracy than the previous versions. The authors of [8] proposed YOLOv4, which considers universal parameters such as weighted residual connections, cross-stage partial (CSP) connections, cross mini-batch normalization (CmBN), self-adversarial training (SAT), Mish activation, mosaic data augmentation, DropBlock regularization, and CIoU loss, combining some to achieve state-of-the-art results: 43.5% AP (65.7% AP50) for the MS COCO dataset at a real-time speed of ~65 FPS on the Tesla V100. In fact, YOLOv4 outperforms its previous versions in terms of both accuracy and speed. For example, YOLOv4 improved upon YOLOv3's AP and FPS by 10% and 12%, respectively.

Tracking and detecting drones is challenging and a critical issue where preventing security breaches is a priority. A drone attack at London Gatwick Airport resulted in the airport being shut down for 18 h, delaying 760 flights and affecting 120,000 passengers [19]. Deep learning is a promising means of detecting and identifying drones [3]. The authors of [20] used CNN-based network architectures such as Zeiler and Fergus (ZF) and the Visual Geometry Group (VGG16) to transfer learning and to detect drones. Their results showed that VGG16 with Faster R-CNN performed better than other architectures did on a

training dataset containing five MPEG4-coded videos taken at different times. The authors of [21] proposed an audio-based drone detection technique using CNN, a recurrent neural network, convolutional recurrent neural network algorithms, and the unique acoustic fingerprints of flying drones. Their dataset consisted of audio recorded samples of drone activities. The authors of [3] used YOLOv2 to detect loaded and unloaded UAVs. The authors of [2] used YOLOv3 and 150 epochs to detect and classify various types of drones. Their dataset contained more than 10,000 images of different categories of drones. The authors of [22] proposed an OpenCV-based drone detection system, achieving 89% accuracy. Their dataset contained 2088 positive and 3019 negative examples. Reference [10] used YOLOv3 to achieve better detection accuracy and obtained more accurate bounding boxes. Their experimental configurations were as follows: a 64-bit Ubuntu 16.04 operating system with a hardware configuration comprising an Intel Xeon E5-2630 v4 processor, a GPU model NVIDIA GeForce GTX 1080 Ti, and memory for 11G. The experiment was run on the Darknet framework. The authors of [19] achieved 88.9% average accuracy in detecting drones. These researchers used input images that were 416 × 416 in size on YOLOv3, pre-trained weights, and transfer learning. The researchers integrated their trained model on a NVIDIA Jetson TX2 for real-time detection. A total of 1500 drone images were manually sorted to remove those that were irrelevant; a total of 1435 were prepared for training. Reference [6] used YOLOv4 to detect low-altitude UAVs, finding that YOLOv4 performed better than YOLOv3 in terms of accuracy and speed. Due to a lack of public low-altitude data, they built their own dataset by flying three types of drones in various conditions: the DJI-Phantom, DJI-Inspire, and XIRO-Xplorer. Later, they mixed their dataset with drone images collected from the internet. Their comparison results consisted only of these three drones. The experiment was conducted using a NVIDIA RTX2060 OC and 6G RAM. The maximum number of iterations was 100,000, the momentum and decay were 0.9 and 0.0005, respectively, and the batch size was 64. YOLOv4 achieved an accuracy of 89.32%, 5.18% higher than YOLOv3.

Due to shape similarities while in the air, drone and bird images are often used in combination. The authors of [23] trained a combined drone-bird dataset using three machine learning models: CNN, SVM, and k-nearest neighbor. The authors achieved the highest accuracy using the CNN model, achieving an accuracy of 93%. The authors of [24] presented a second edition of "drone-vs-bird" detection, in which they summarized the four best-performing models. All four were based on the CNN algorithm, and the best achieved a 0.73 F1-score. Reference [25] prepared a drone-bird dataset in order to detect drones using the YOLOv2 object detector algorithm. The authors in [26] created a large database of drones and birds in order to classify them using a CNN algorithm, achieving 99.6% validation and 94.4% test accuracy.

In this study, we chose to use the state-of-the-art in object detection, the YOLO-v4 algorithm, because of its real-time detection capabilities, high speed, and accuracy. To train this neural network architecture, we collected 479 images of 300 species of birds and 1916 images of drones from public resources. We prepared our own drone dataset to verify the drone detection capability. We chose to use bird images due to their similarity to drones. We used mean average precision (mAP) as our evaluation metric to evaluate the object detector's performance. Using the collected and prepared dataset, the trained YOLOv4 neural network was evaluated in terms of its detection ability, location precision, and mAP. The next section discusses the YOLOv4, our chosen methodology, in detail.

3. Materials and Methods

YOLOv4 [8] introduces new universal features (i.e., WRD, CSP, CmBN, SAT, Mish activation, mosaic data augmentation, DropBlock regularization, and CIoU loss) in combination to achieve high AP and FPS. YOLOv4 follows a one-stage detector architecture comprised of four parts: input, backbone, neck, and dense prediction or head. The input is the set of data we want to detect. The backbone is responsible for extracting features and uses the image dataset to make the object detector scalable and robust. It is comprised

three parts: bag of freebies (BoF), bag of specials (BoS), and CSPDarknet53. The head uses same strategy as YOLOv3 [7].

3.1. Bag of Freebies

BoF is a strategy used to train the object detector offline without increasing inference cost. There are various strategies available in computer vision to achieve the goal of BoF, but YOLOv4 uses specific techniques for both the backbone and the detector. Important BoF strategies used in YOLOv4 include CutMix, mosaic data augmentation, label smoothing, IoU loss, and DropBlock regularization.

Data augmentation is used to improve the robustness of the object detection model. The result is an increase in the variability of images so that an unknown environment will not create any issues for the detector model. Adjusting the brightness, contrast, hue, saturation, and noise of an image assists with overcoming photometric distortion. Random flipping, scaling, cropping, and rotating are used to overcome geometric distortions. Other than such pixel-wise adjustments, random erase, MixUp, CutMix, style transfer, GAN, etc., can also be used.

BoF uses focal loss (FL) to deal with the issue of data imbalance. In classification problems, the cross entropy (CE) loss function is used; however, it cannot smoothly handle misclassified targets. Thus, FL is introduced, which is basically a modified version of CE. In FL, an additional co-efficient $(1-\rho_t)^{\gamma}$ is used.

$$\mathrm{CE}(\rho_t) = -\log(\rho_t) \tag{1}$$

$$\mathrm{FL}(\rho_t) = -(1-\rho_t)^{\gamma}\log(\rho_t) \tag{2}$$

Label smoothing is introduced in YOLOv4, which is basically the concept of distillation. Label smoothing converts hard labels into soft labels, producing robustness in the model. Another important improvement in YOLOv4 is the inclusion of IoU loss. In conventional object detection models, l_1 or l_2 losses are calculated in order to evaluate the bounding box prediction; this tends to minimize errors on small objects and large bounding boxes. IoU loss overcomes this issue in YOLOv4 because of its mathematical representation.

$$l_2 \text{ loss} = ||\,(\text{Predicted Bounding Box}) - (\text{Ground truth Bounding Box})||_2^2 \tag{3}$$

$$\text{IoU loss} = -\ln\frac{\text{Intersection}\,(\text{Predicted Bounding Box}, \text{Ground truth Bounding Box})}{\text{Union}\,(\text{Predicted Bounding Box}, \text{Ground truth Bounding Box})} \tag{4}$$

3.2. Bag of Specials

YOLOv4 introduces a set of strategies called BoS to improve object detection accuracy by increasing a small amount of inference costs. Various techniques are incorporated in order to implement BoS, but the most significant improvements include Mish activation, CSP connections, SPP-block, and PAN path-aggregation block. Mish activation considers the negative information, thus solving the dying ReLU phenomenon and providing strong regularization effects during training to overcome the overfitting issue. The Mish activation function is shown in Figure 1.

3.3. CSPDarknet53

YOLOv4 uses CSPDarknet53 as its detection architecture. Though CSPResNext50 performs better for classifying objects in ILSVRC20212 (ImageNet), CSPDarknet53 performs better when detecting objects in the MS COCO datasets [8]. A performance graph of the original YOLOv4 can be found in Figure 2. In Figure 2, performance evaluation metrics, fps, and AP, are compared with other methods. The performance of YOLOv4 is shown in green, and it is labeled as "YOLOv4 (ours)". Figure 2 from the original YOLOv4 paper shows superior performance compared to the other object detection methodologies and was one of the reasons behind our choice to use YOLOv4 for detection purposes. CSPDarknet53

consists of 29 layers of 3 × 3 filters, 725 × 725 receptive fields, and 27.6 M parameters. This architecture has proven to be superior to its competitor architecture, CSPResNext50 [8]. The addition of an SPP block over the CSPDarknet53 significantly increases the receptive field performance by bringing out contextual features. YOLOv3's FPN is replaced by PANet in YOLOv4 as a parameter aggregation method. The final YOLO head is based on the strategy of YOLOv3. In short, the YOLO head works in three steps. First, it divides the entire image into N × N grids. Each grid has five parameters (i.e., x, y, w, h, and c; object_confidenc_score), where (x, y) is the offset value between the prediction box and the respective grid cell bound, (w, h) is the width and height from the prediction box to the entire image, and object_confidence_score expresses the probability of the class object. A CNN extracts the feature and predicts classes with class probability scores. Finally, non-maximum suppression is used to eliminate the repetitive bounding boxes and to produce a single bounding box for each class. The overall detection architecture for YOLOv4 is given below in Figure 3.

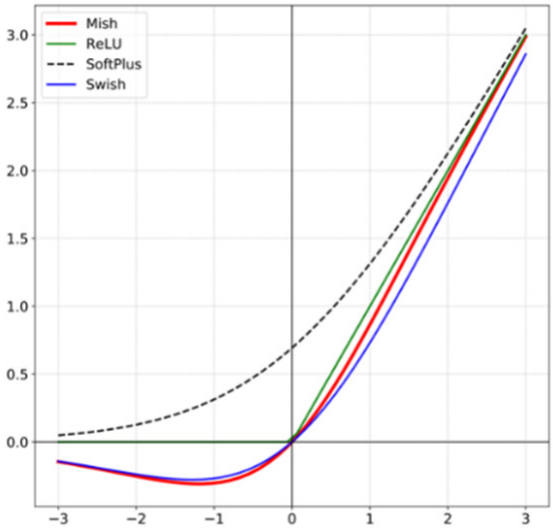


Figure 1. Mish activation comparison with ReLU, Swish, and SoftPlus [27].

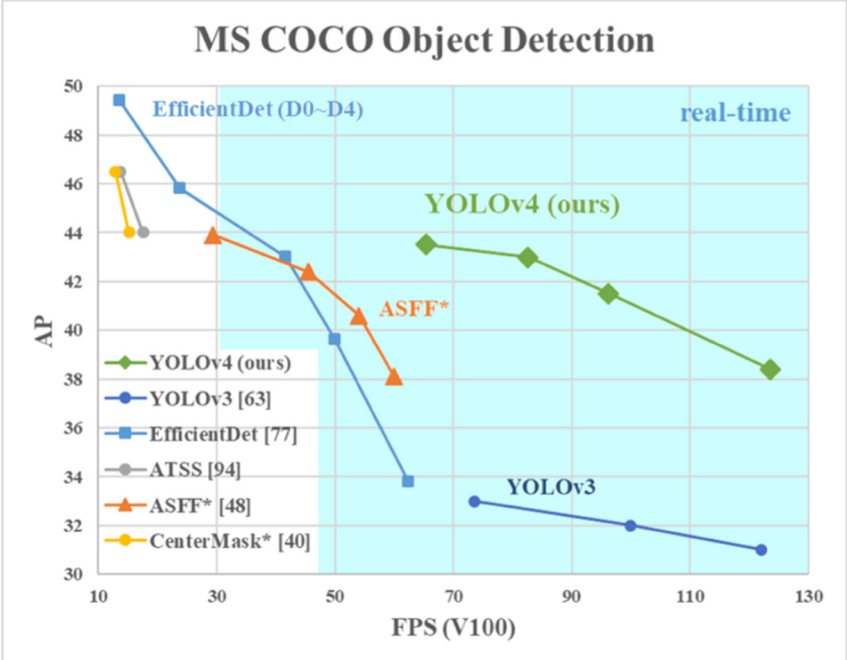


Figure 2. Performance comparison of YOLOv4 (original) with other object detectors [8].

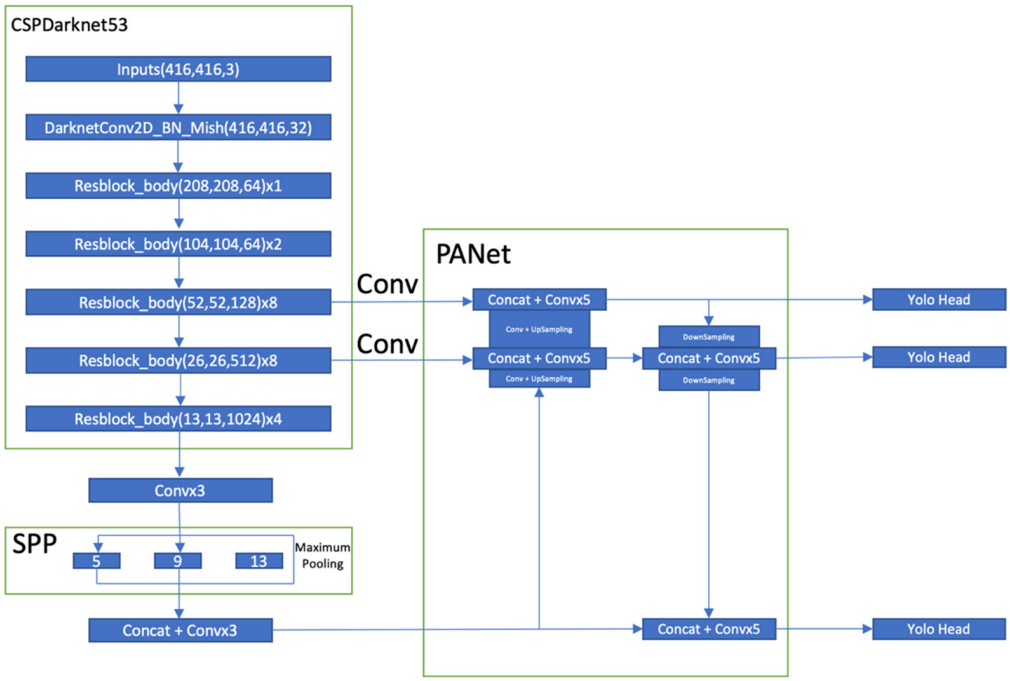


Figure 3. YOLOv4 detection architecture [6].

4. Results

4.1. Construction of Experiment and Data Acquisition

Due to the scarcity of drone and bird images, we collected images from various resources such as Google and Kaggle. The drone images that were collected were from various altitudes and angles. We used images of around 300 species of birds. Altogether, we collected 2395 images consisting of 479 birds and 1916 drones, as we mentioned previously. We split the dataset into a 90/10 train_test_split. For image annotation, we used the LabelImg tool and manually annotated the images using two classes; drone was the "first class", and bird was the "zero class". For YOLO implementation, we saved the annotated images in a .txt format.

To evaluate the performance of the trained YOLOv4, two types of drones were flown, and flying videos were captured at three altitudes: 60 feet, 40 feet, and 20 feet. Drone models DJI Mavic Pro and DJI Phantom III were used. The drones were flown at three altitudes to evaluate the detection speed and capability of the trained YOLOv4. The reason why we selected those altitudes is that above 60 ft, the drones look almost invisible to cameras. In addition, the reason why we used these drone models is their popularity among drone hobbyists.

The experiment was conducted using a Darknet framework and Google deep learning VM. We used a Tesla K80 graphic processing unit (GPU) to train the Darknet. A cuDNN 7.6.5 was used to run processes on the NVIDIA GPU. For video detection, Google CoLab was used with a GPU Tesla T4 and OpenCV version 3.2.0. We configured and fine-tuned the YOLOv4 architecture for our custom dataset. The main source code of the Darknet framework was prepared by [8], and in our research, we used the transfer learning technique to make the framework compatible with our custom dataset. We fine-tuned the last three YOLO and convolutional layers for our certain number of classes. The original Darknet was trained on 80 classes; thus, we changed the number of classes into two, namely "drone" and "bird". Before each three YOLO layers, there were three convolutional layers in order to build high-level feature map of the objects. In convolutional layers, filters are used to extract features. For the original Darknet, they used 255 filters. The number of filters is calculated using the formula: (number of classes + 5) $\times$ 3. Thus, we changed the number of filters to 21 in the three convolutional layers before the YOLO layers. We kept rest of the layers among the same 162 layers in our implementation. To address the

data scarcity issue, YOLOv4 introduces various data augmentation techniques that we discussed previously. We turned the MOSAIC flag on to automate the data augmentation process. We tuned the number of batches, which was set to 64. Depending on the GPU, there are various numbers to be tried for subdivision, starting from 8 to multiple of 8. In our case, subdivision = 32 worked. We set the image width × height = 608 × 608 pixels. Other hyperparameters such as learning rate = 0.001, momentum = 0.949, decay = 0.0005, hue = 0.1, batch normalization = 1, activation = mish, etc., were kept as default values. Further, we fine-tuned the maximum number of batches and set it to 4000, which was calculated using the formula (number of classes × 2000). Steps were calculated using the formula (80% and 90% of maximum batches). Thus, we set the step range between 3200 and 3600. Finally, we trained the YOLOv4 on Google's deep learning VM and later tested it on our testing images and videos. We trained YOLOv4 for 4000 iterations and saved the trained weights for each 1000 iterations and later constructed a number of iterations versus the mAP curve at four different points as weights that had been saved at 1000, 2000, 3000, and 4000 iterations by the default Darknet framework. A flowchart of the overall experiment is shown in Figure 4.

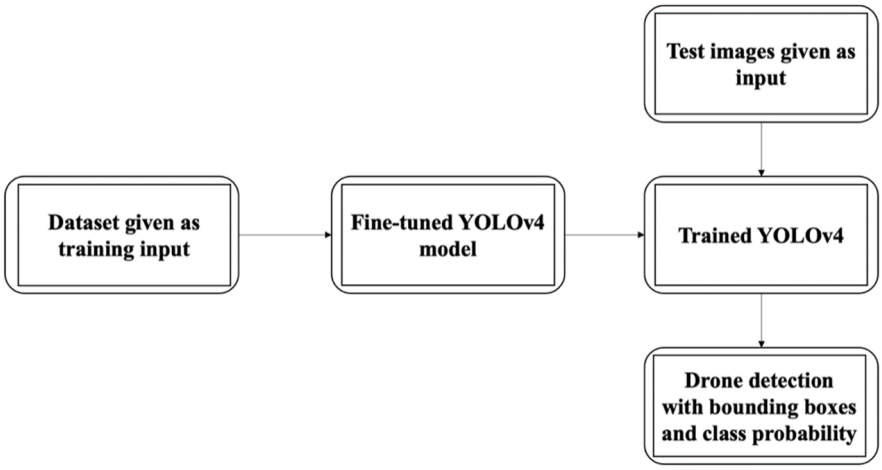


Figure 4. Flowchart of the experiment.

4.2. Evaluation

The trained YOLOv4 was evaluated using mAP, precision, recall, and F1-score. In addition, the FPS was calculated in order to check the detection speed of the model for the captured videos. The precision, recall, and F1-scores of the trained YOLOv4 are shown in Table 1. Table 2 shows the mAP and FPS values for two captured videos. Primarily, an evaluation was performed for the testing images of birds and drones. In addition, our testing was performed considering a complex background, different weather conditions (cloudy, sunset, etc.), and multiple objects in one image. We plotted a curve for tracking mAP improvement over iterations, and the mAP values were computed at four different iterations that were mentioned previously. Figure 5 shows the iterations versus mAP curve. The highest mAP was achieved during 4000 iterations, and the mAP was 74.36%. Drone and bird detection with class probabilities are shown in Figure 6. Due to space and better clarity, more detection images are shown in Appendix A. Figures 7 and 8 show the drone detection for the videos.

Table 1. Calculated precision, recall, and F1-score.

Precision	Recall	F1-Score
0.95	0.68	0.79

Table 2. Evaluation results of YOLOv4 for video detection.

Model	mAP	Video	Drone Type	Video Detection FPS
YOLOv4	74.36%	Video_1.mp4	DJI Phantom III	20.5
		Video_2.mp4	DJI Mavic Pro	19.0

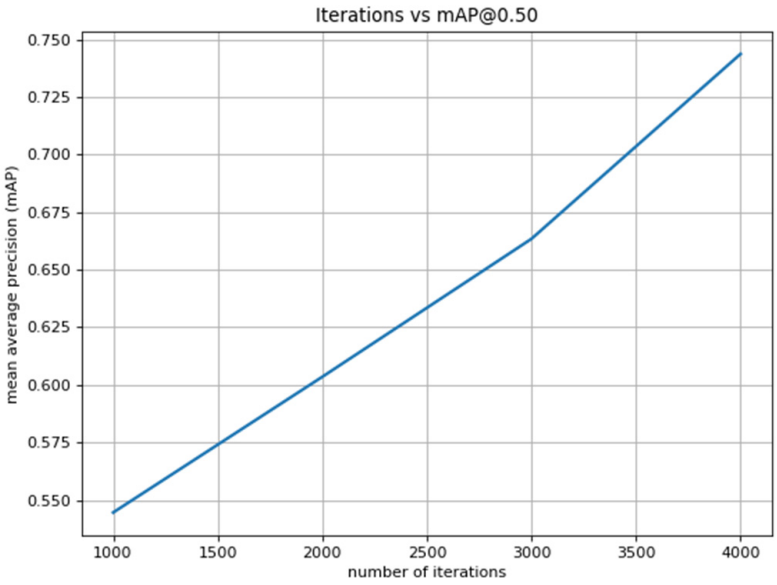
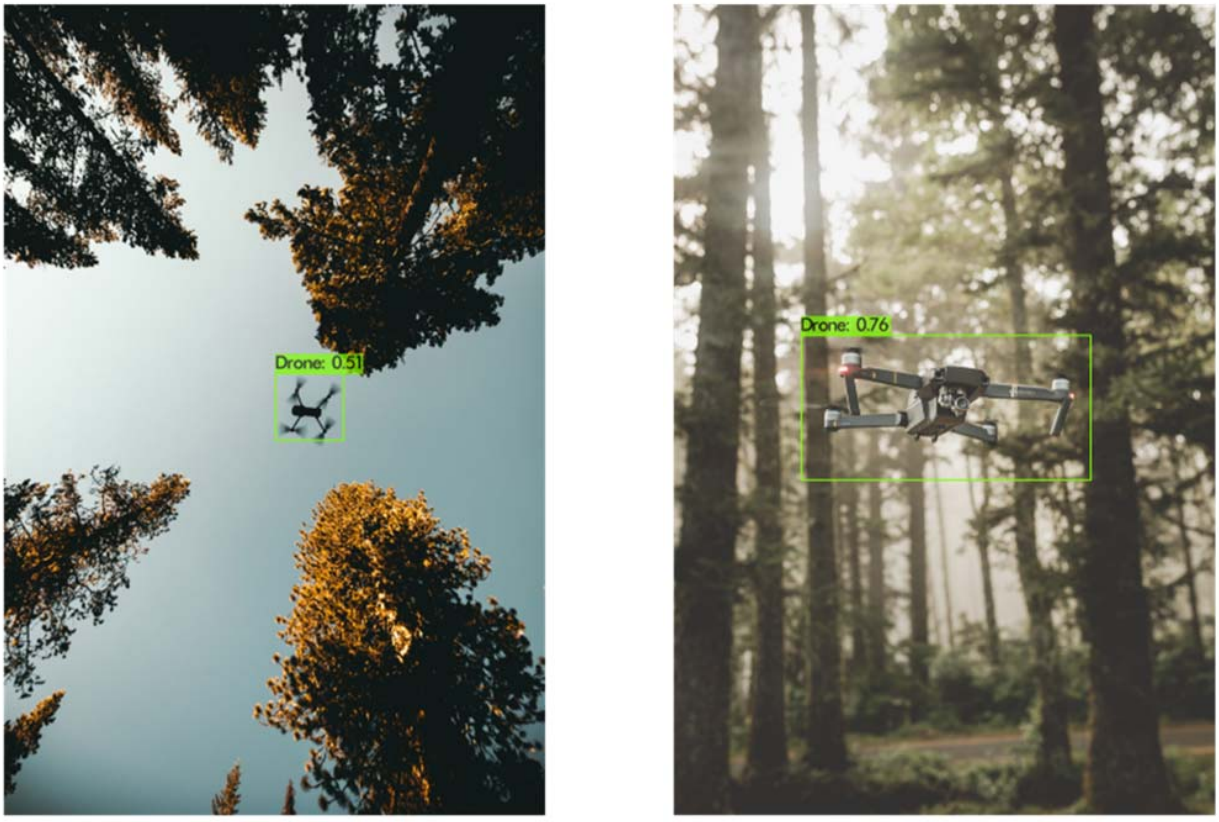
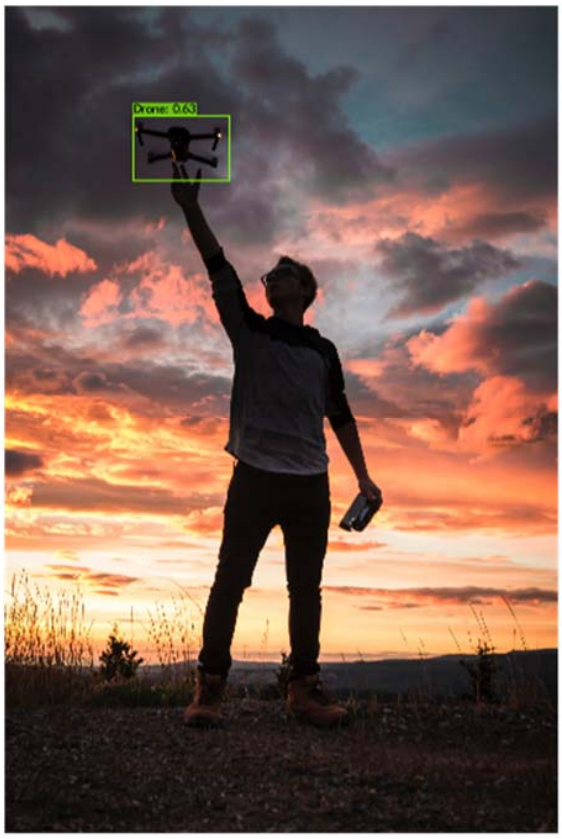


Figure 5. Iterations vs. mAP curve.

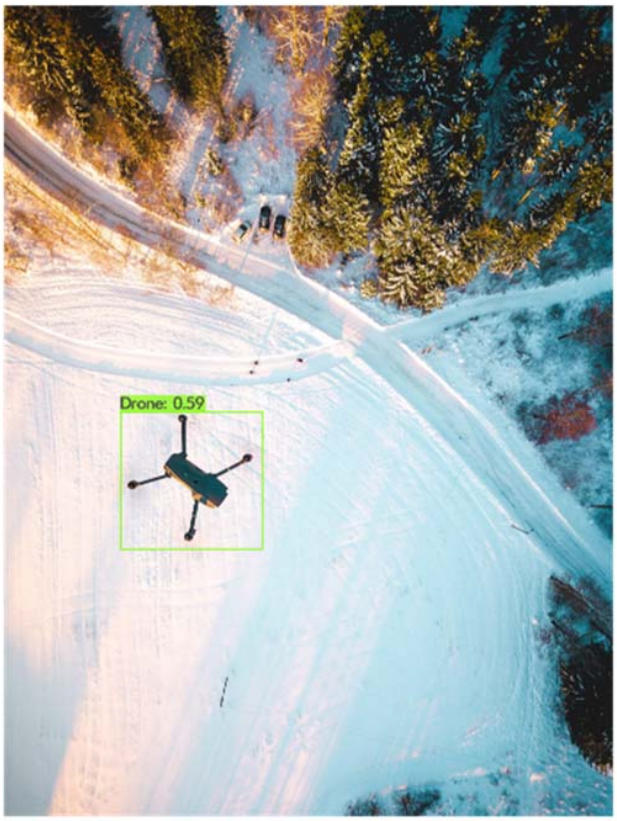


(**a**) (**b**)

Figure 6. *Cont.*



(**c**)



(**d**)

Figure 6. Drone and bird detection with bounding boxes and class probabilities. (**a**) Bottom-view of drone; (**b**) shaded forest background; (**c**) man with drone during sunset; (**d**) top-view of drone.

(**a**)

Figure 7. *Cont.*

(**b**)

(**c**)

Figure 7. Drone detection in Video_2.mp4 (drone type: DJI Mavic Pro). (**a**) Drone detection at an altitude of 20 ft; (**b**) drone detection at an altitude of 40 ft; (**c**) drone detection at an altitude of 60 ft.

(**a**)

(**b**)

Figure 8. *Cont.*

(**c**)

Figure 8. Drone detection in Video_1.mp4 (drone type: DJI Phantom III). (**a**) Drone detection at an altitude of 20 ft; (**b**) drone detection at an altitude of 40 ft; (**c**) drone detection at an altitude of 60 ft.

5. Discussion

Previous studies on this topic have mostly focused on drone detection. Few have considered both drones and birds as detection classes. In fact, "Drone vs. Bird" is a popular challenge competition in which participants are asked to design object detectors to detect drone and bird classes [24]. Our main goal was to detect drones and drone-like objects such as birds in order to compare our work with previous studies such as [20,24]. In [20], the researchers used various deep CNN architectures. Their methodology employed Faster RCNN for object detection, an anchor-based algorithm similar to YOLO. Using ZF, VGG16, and VGG_M_1024, the authors achieved mAP values of 0.61, 0.66, and 0.60, respectively. Their proposed methodology successfully detected both drones and birds. VGG16 performed better because they fine-tuned the architecture and trained their dataset on top of the ImageNet model. In our methodology, a trained YOLOv4 weighted on the MS-COCO dataset was used. Our neural network architecture was tuned to only fit our custom dataset. In addition, our model considered drone detection at three defined altitudes, and the trained model was evaluated using our own drone videos. In addition to mAP, FPS, precision, recall, and F1-score were used as evaluation metrics. Our methodology outperformed the highest mAP of 0.66, achieving a mAP of 0.7436. The authors of [24] compared the top four teams' neural network models in the Drone vs. Bird detection challenge. The authors found that the top four teams achieved F-1 scores of 0.12, 0.41, 0.68, and 0.73. In our methodology, an F-1 score of 0.79 was achieved.

In our study, fps was one of the key parameters to be considered while evaluating the performance of the detector. This parameter is widely used for testing the speed of detection in videos. Depending on the available computing resources, such as random-access memory (RAM), graphical processing unit (GPU), or central processing unit (CPU), this parameter varies a lot. Even the length of the video may play a role while comparing fps. Here, we considered similar research for comparison purposes only. The authors

of [28,29] performed drone and bird detection in real time. Their methodologies calculated the fps while evaluating their performances. In [28], they introduced two methodologies, Deep Residual CNN with Skip Connection and Network in Network (DCSCN) and a compact version of DCSCN (c-DCSCN), for "drone", "bird", and "rest" classes detection. They incorporated a super-resolution technique for long range video surveillance. Using the methodologies, they successfully detected drones in videos, and their average fps was 0.32 and 0.58, respectively, for DCSCN and c-DCSCN. In their techniques, a Faster-RCNN was trained for 70k iterations. Other hyperparameters were learning rate (0.001) and stochastic gradient descent (momentum: 0.9 and weight decay: 0.0004). They used a NVIDIA GeForce TITAN XP with 12 GB memory. Their fps is shown in Figure 9 with red bars. Similarly, in [29], they trained a MobileNetV2 CNN to detect drone and objects similar to drones such as birds or airplanes. They incorporated a background subtraction technique to increase the accuracy and speed. They achieved an average detection speed of 9 fps. They used the SGD optimization algorithm with a learning rate of 0.05, a momentum of 0.9, and weight decay of 0.001. They used a NVIDIA GeForce GT 1030 2 GB GPU for training and detection. Only the fps of their performance is shown in Figure 9, which is depicted in red and is labeled as MobileNetV2 CNN. In terms of fps, our performance is shown in green bars for two videos. Comparing other methodologies, we used free and publicly available Google CoLab RAM of 12 GB. We used free GPU from Google, which was dynamic in nature, i.e., based on the availability that Google provides the configuration. At the time of implementation, we used a Tesla T4 with unknown memory. Using these low-cost resources, our fps was 20.5 and 19.0 for video 1 and video 2, respectively. A comparison of the fps parameter is shown in Figure 9.

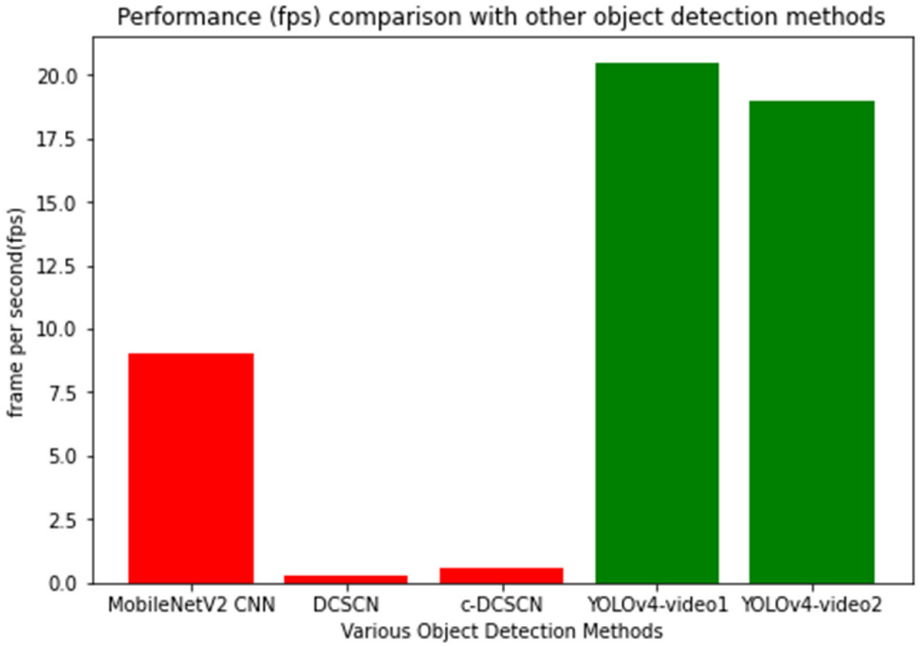


Figure 9. Frame per second (fps) comparison with other methods.

In this study, YOLOv4 performed better due to the capability of detecting objects in real-time. The YOLO algorithm predicts a class with localization using only a single pass over an image. Further, YOLOv4 introduces various new features such as WRC, CSP, CmBN, mish activation, mosaic data augmentation, and complete intersection over union loss (CIoU loss), and these new features make it fast. In our case, we fine-tuned the original architecture and trained on top of the YOLOv4 weights, which made it accurate. MOSAIC = 1 performed the data augmentation and provided an artificial augmented dataset on top of our collected dataset. Using the bird dataset further strengthened the

classifier while detecting drones against similar objects. The training time was long enough, which also helped to ensure accurate prediction.

6. Conclusions

In this research, YOLOv4 was trained to detect drones and drone-like objects (i.e., birds). Our model performed better than those of previous similar studies. Drone detection is necessary, considering that drone intervention is frequent in unauthorized and emergency tasks. However, detecting drones at various altitudes can be difficult, especially due to their small size and high altitude and speed as well as the existence of drone-like objects. Drone and bird image databases were compiled in this research by collecting images from available public resources. Using those collected images, a YOLOv4 model was trained and evaluated via our own drone videos. The performance of the trained YOLOv4 was tested in real time at three different altitudes: 20 ft, 40 ft, and 60 ft. The mAP and FPS evaluation metrics were then calculated to check the performance. Using a Tesla T4 GPU and OpenCV (3.2.0), the YOLOv4 achieved a mAP of 74.36% at an IoU of 50 and FPS of 19.0 for the DJI MAVIC Pro and an FPS of 20.5 for the DJI Phantom III. This study was limited to YOLO implementation only since various object detection algorithms require datasets to be labeled in certain formats, which is time consuming. In addition, speed was one of our considerations while choosing algorithms. In future work, a more diverse image dataset will be used to further improve the results. In addition, other object detection algorithms such as R-CNN, mobilenet, SSD, etc., will be trained and compared. YOLOv5 has already been released; thus, we will further use this version to see if the speed and accuracy improve. More objects will be added alongside the drone and bird images.

Author Contributions: Conceptualization, S.S. and B.A.; methodology, B.A. and S.S.; software, S.S.; validation, B.A. and S.S.; formal analysis, S.S and B.A.; investigation, B.A. and S.S.; resources, S.S.; data curation, S.S.; writing—original draft preparation, S.S.; writing—review and editing, B.A.; visualization, S.S.; supervision, B.A.; project administration, S.S. and B.A.; funding acquisition, B.A. Both authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets used or analyzed during the current study are available from the corresponding author upon reasonable request.

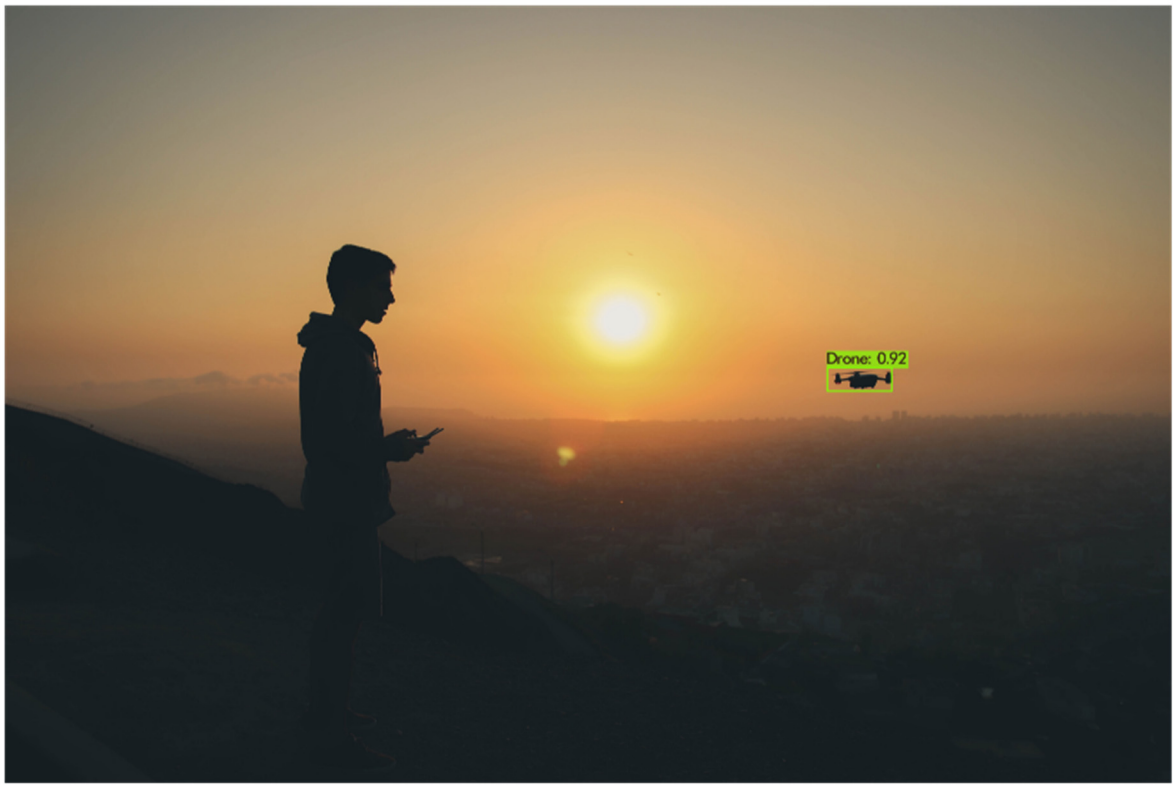
Acknowledgments: We are grateful to the Presidential Graduate Assistant Research initiative program of Texas A&M University—Commerce that made the research possible.

Conflicts of Interest: We declare that there is no conflict of interest.

Appendix A

Our classifier detected drone and bird objects successfully in various images. Images with complex backgrounds in different weather conditions were tested. Here, we have included the detection results, where the images are shown with their class probabilities and with their respective class names.

(**a**)



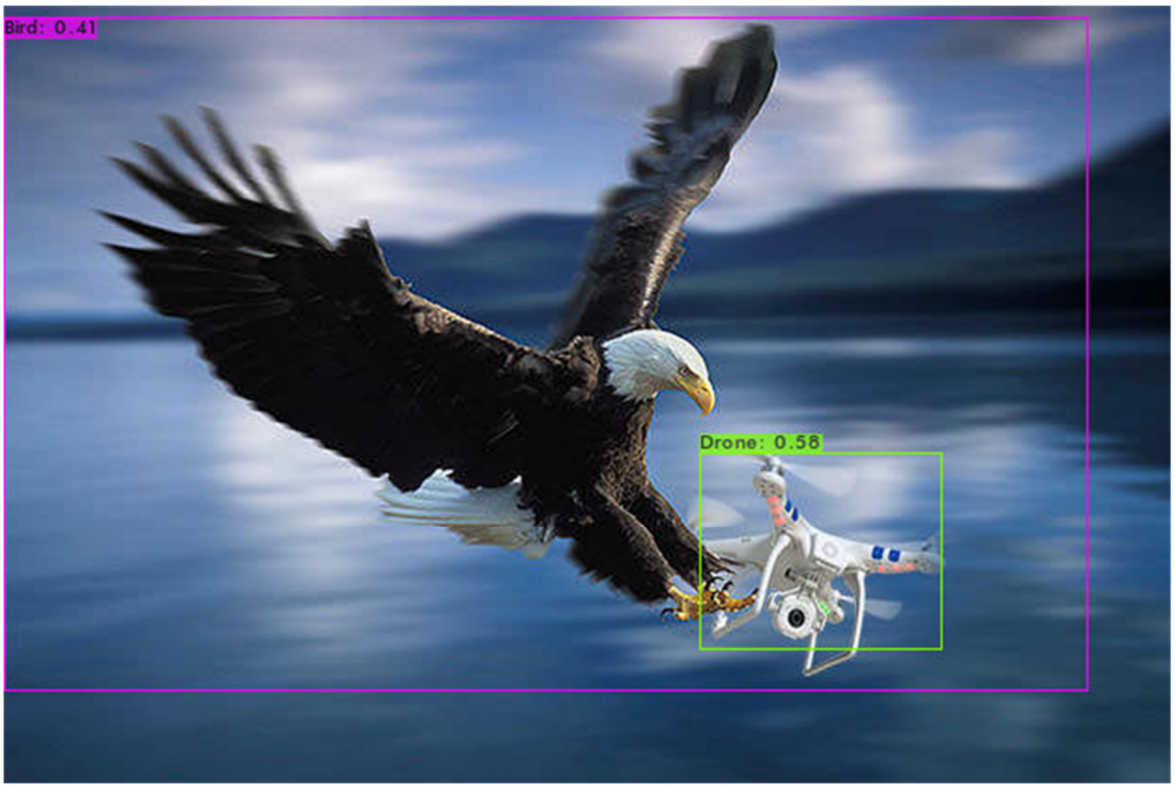
(**b**)

Figure A1. *Cont.*

(**c**)

(**d**)

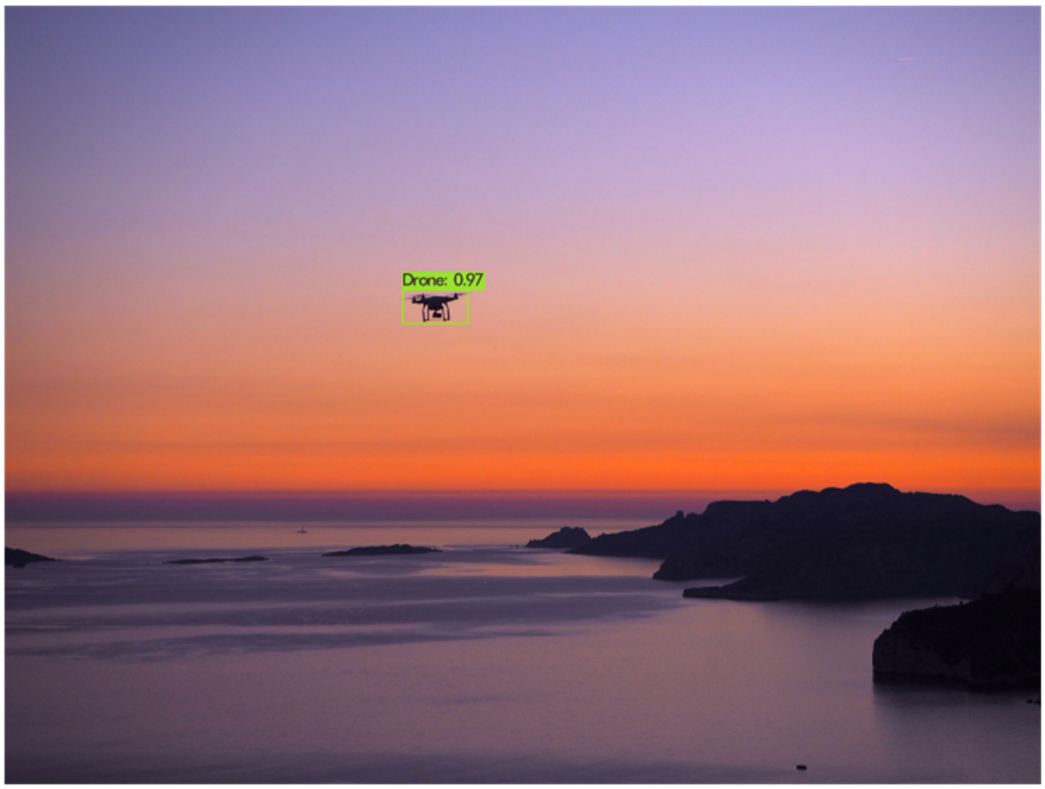
Figure A1. *Cont.*



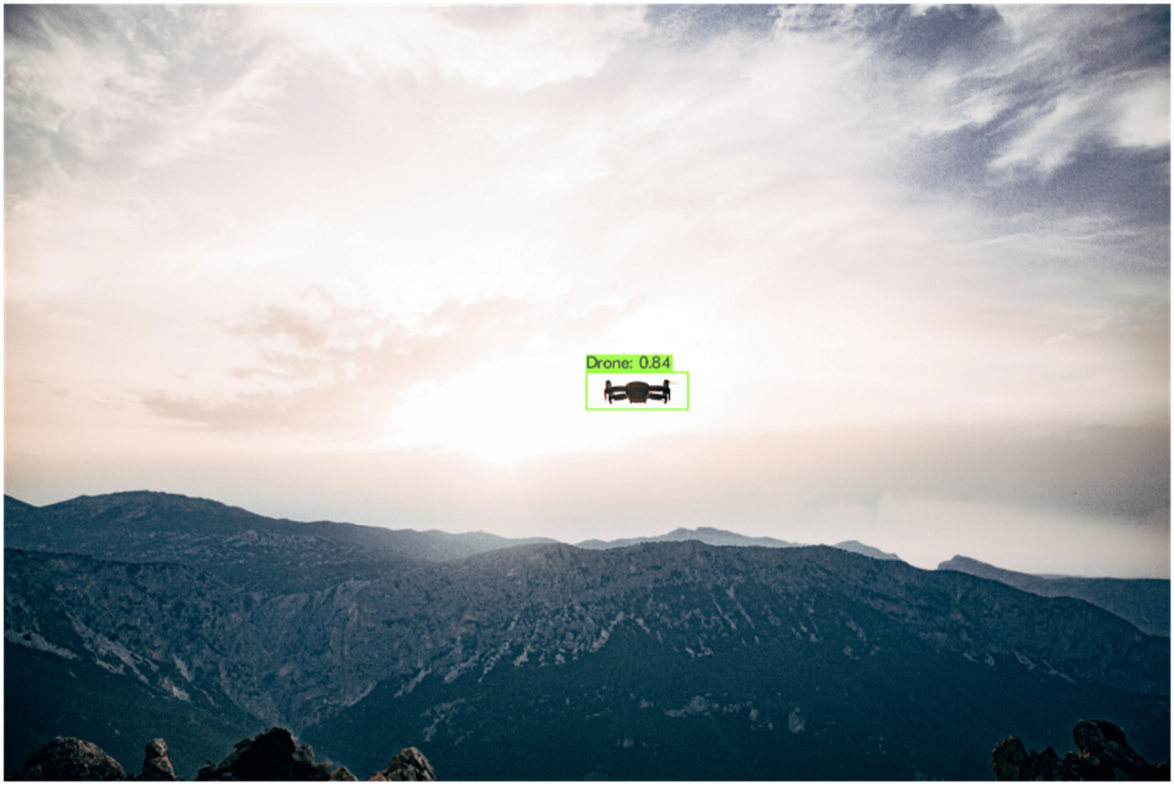
(**e**)

(**f**)

Figure A1. *Cont.*

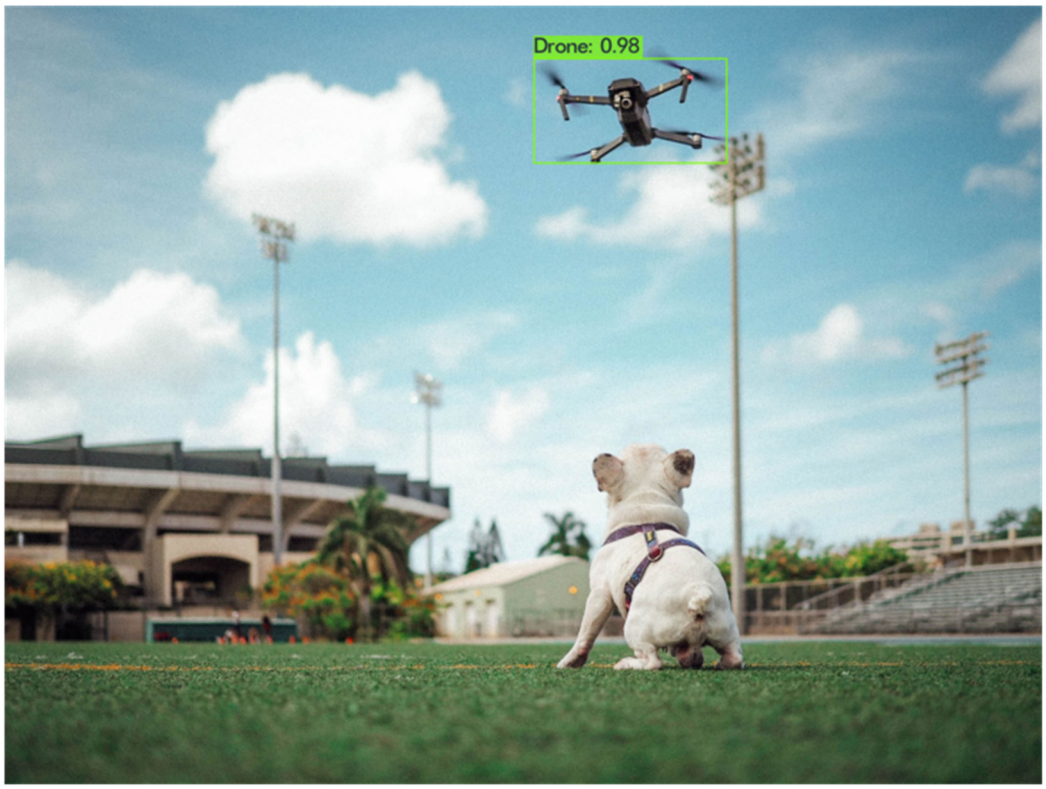


(**g**)



(**h**)

Figure A1. *Cont.*



(**i**)

Figure A1. Drone and bird detection with bounding boxes and class probabilities. (**a**) Drone in low-light at a far distance; (**b**) drone in bright light background during sunset; (**c**) drone at night with bright artificial light; (**d**) drone during cloudy weather; (**e**) bird grabbing drone; (**f**) background where the environment is too sunny; (**g**) drone flying during sunset at a beach; (**h**) bright-sunny background on top of mountain; (**i**) multiple objects with blurry background.

References

1. UAS by the Numbers. Available online: https://www.faa.gov/uas/resources/by_the_numbers/ (accessed on 28 July 2021).
2. 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.
3. Seidaliyeva, U.; Alduraibi, M.; Ilipbayeva, L.; Almagambetov, A. Detection of Loaded and Unloaded UAV Using Deep Neural Network. In Proceedings of the 2020 Fourth IEEE International Conference on Robotic Computing (IRC), Taichung, Taiwan, 9–11 November 2020; pp. 490–494.
4. Rohan, A.; Rabah, M.; Kim, S.-H. Convolutional Neural Network-Based Real-Time Object Detection and Tracking for Parrot AR Drone 2. *IEEE Access* **2019**, *7*, 69575–69584. [CrossRef]
5. Kumawat, H.C.; Raj, A.B. Extraction of Doppler Signature of Micro-to-Macro Rotations/Motions Using Continuous Wave Radar-Assisted Measurement System. *IET Sci. Meas. Technol.* **2020**, *14*, 772–785. [CrossRef]
6. Shi, Q.; Li, J. Objects Detection of UAV for Anti-UAV Based on YOLOv4. In Proceedings of the 2020 IEEE 2nd International Conference on Civil Aviation Safety and Information Technology (ICCASIT), Weihai, China, 14–16 October 2020; pp. 1048–1052.
7. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
8. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
9. Albawi, S.; Mohammed, T.A.; Al-Zawi, S. Understanding of a Convolutional Neural Network. In Proceedings of the 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 21–23 August 2017; pp. 1–6.
10. Hu, Y.; Wu, X.; Zheng, G.; Liu, X. Object Detection of UAV for Anti-UAV Based on Improved YOLO V3. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 8386–8390.
11. Girshick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
12. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]

13. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988.
14. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
15. Tao, J.; Wang, H.; Zhang, X.; Li, X.; Yang, H. An Object Detection System Based on YOLO in Traffic Scene. In Proceedings of the 2017 6th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 21–22 October 2017; pp. 315–319.
16. Sadykova, D.; Pernebayeva, D.; Bagheri, M.; James, A. IN-YOLO: Real-Time Detection of Outdoor High Voltage Insulators Using UAV Imaging. *IEEE Trans. Power Deliv.* **2020**, *35*, 1599–1601. [CrossRef]
17. Kharchenko, V.; Chyrka, I. Detection of Airplanes on the Ground Using YOLO Neural Network. In Proceedings of the 2018 IEEE 17th International Conference on Mathematical Methods in Electromagnetic Theory (MMET), Kyiv, Ukraine, 2–5 July 2018; pp. 294–297.
18. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.
19. Wei Xun, D.T.; Lim, Y.L.; Srigrarom, S. Drone Detection Using YOLOv3 with Transfer Learning on NVIDIA Jetson TX2. In Proceedings of the 2021 Second International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP), Bangkok, Thailand, 20–22 January 2021; pp. 1–6.
20. Saqib, M.; Daud Khan, S.; Sharma, N.; Blumenstein, M. A Study on Detecting Drones Using Deep Convolutional Neural Networks. In Proceedings of the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 29 August–1 September 2017; pp. 1–5.
21. Al-Emadi, S.; Al-Ali, A.; Mohammad, A.; Al-Ali, A. Audio Based Drone Detection and Identification Using Deep Learning. In Proceedings of the 2019 15th International Wireless Communications Mobile Computing Conference (IWCMC), Tangier, Morocco, 24–28 June 2019; pp. 459–464.
22. Lee, D.; Gyu La, W.; Kim, H. Drone Detection and Identification System Using Artificial Intelligence. In Proceedings of the 2018 International Conference on Information and Communication Technology Convergence (ICTC), Busan, Korea, 22–24 October 2014; pp. 1131–1133.
23. Mahdavi, F.; Rajabi, R. Drone Detection Using Convolutional Neural Networks. In Proceedings of the 2020 6th Iranian Conference on Signal Processing and Intelligent Systems (ICSPIS), Mashhad, Iran, 23–24 December 2020; pp. 1–5.
24. Coluccia, A.; Fascista, A.; Schumann, A.; Sommer, L.; Ghenescu, M.; Piatrik, T.; De Cubber, G.; Nalamati, M.; Kapoor, A.; Saqib, M.; et al. Drone-vs-Bird Detection Challenge at IEEE AVSS2019. In Proceedings of the 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Taipei, Taiwan, 18–21 September 2019; pp. 1–7.
25. Aker, C.; Kalkan, S. Using Deep Networks for Drone Detection. In Proceedings of the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 29 August–1 September 2017; pp. 1–6.
26. Rahman, S.; Robertson, D.A. Classification of Drones and Birds Using Convolutional Neural Networks Applied to Radar Micro-Doppler Spectrogram Images. *IET Radar Sonar Navig.* **2020**, *14*, 653–661. [CrossRef]
27. Misra, D. Mish: A Self Regularized Non-Monotonic Activation Function. *arXiv* **2020**, arXiv:1908.08681.
28. Magoulianitis, V.; Ataloglou, D.; Dimou, A.; Zarpalas, D.; Daras, P. Does Deep Super-Resolution Enhance Uav Detection? In Proceedings of the 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Taipei, Taiwan, 18–21 September 2019; IEEE: Piscataway Township, NJ, USA, 2019; pp. 1–6.
29. Seidaliyeva, U.; Akhmetov, D.; Ilipbayeva, L.; Matson, E.T. Real-Time and Accurate Drone Detection in a Video with a Static Background. *Sensors* **2020**, *20*, 3856. [CrossRef] [PubMed]