UAV Detection using Web Application Approach based on SSD Pre-Trained Model

Leonard Matheus Wastupranata
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
leo.matt.547@gmail.com

Rinaldi Munir
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
rinaldi@informatika.org

2021 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES) | 978-1-6654-0179-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICARES53960.2021.9665191

Abstract— **UAV development is being intensively developed by various groups to help overcome various types of problems. Object Detection is important in helping UAVs to do drone chasing and other competition that need visual approach based on image processing and deep learning. Unfortunately, the computational capabilities of the onboard processing unit that attached to the UAV are less than optimal for object detection due to storage and memory size constraints. This paper aims to create the new approach to improve the precision and recall during UAV detection by using web application to do real time detection. To decide a pre-trained model, it is necessary to compare which SSD pre-trained model is suitable to be deployed in this web application. The results obtained are that using the web application approach is better than the onboard processing approach with a high level of precision and recall with an average precision value of 0.85 and an average recall value of 0.837.**

Keywords— UAV, object detection, SSD, web application, deep learning

I. INTRODUCTION

In modern times, the use of UAVs is intended to fulfill increasingly complex human needs. From rush hour dispatch services to scanning inaccessible areas, UAVs are proving to be essential in a variety of situations where humans cannot achieve or cannot perform hazardous/risky tasks in a timely and efficient manner[1]. Now, UAV technology integrates autonomous capabilities with aerial vehicles to do a variety of tasks, including package delivery, autonomous mapping, and surveillance, as well as capturing another autonomous aircraft in flight. The expanding number of UAV competitions throughout the world, ranging from payload delivery to drone chasing, proves that UAV technology is growing rapidly today.

A drone chasing competition is one of the races which drones compete to tag each other. The first drone to be tagged loses the race. Although it is a relatively simple challenge for human-controlled drones to solve, it is a considerably more difficult problem for autonomous drones to solve since, to tag another drone, the drone must first know the precise location of other drones to tag it down. One way for a drone to learn about the location of other drones is by detection.

To help the drone learn about the location visually, TensorFlow Object Detection API[2] will be the problem solver. This API is the most widely used API for detecting objects captured on camera. Comparable or same arrangement of pixels that make up an object to the original item will be considered. To do that, Single Shot Object Detection (SSD) is the Object Detection method which is good for detecting small targets because low-level feature maps is used with high resolution process[3]. In carrying out the object detection process, there are several pre-trained models contained in the zoo model. The model has been pre-trained with general objects whose details can be seen in the detection label map file [4].

Makirin et.al. [5] proposed the real time detection of flying drones by prioritizing the concept of a lightweight model with good accuracy. The results using SSD MobileNet v2 are precision of 0.586 and sensitivity of 0.622 at IoU 0.50 - 0.95 in all-area of captured image. However, the accuracy of this model is still lacking and requires heavy computing on the onboard processing. A new approach is needed to carry the computational load so that this extra resource requirement can be reduced in the onboard processing of the UAV.

Vaddi et al. [6] also have proposed the efficient way for Real-Time UAV Object Detection. The method used is to compare models that are suitable for detecting pedestrians, vehicles, bicycles, etc., using MobileNet[7] and ResNet[8] models. The results obtained are the combined model with MobileNet as backend feature extractor gave the best results in terms of accuracy, speed and memory efficiency and is best suitable for real time object detection with drones. Unfortunately, the proposed method does not detect other UAVs in more detail so they cannot be used for tracking other UAVs.

TABLE I. LIST OF ABBREVIATIONS

Acronym	Acronym
UAV	Unmanned Aerial Vehicle
SSD	Single Shot Detector
API	Application Programming Interface
IoU	Intersection of Union
GPU	Graphics Processing Unit
COS	Cloud Object Storage

Funding was received from P3MI STEI ITB to sponsor the research

In this paper, a new web application approach will be proposed so the precision and recall can be improved for the better detection instead of onboard processing [5]. The precision obtained is expected to be greater than 80% and the successful recall number obtained is expected to be greater than 70% to be concluded as successful. In addition, this pre-trained model will be developed with the help of cloud computing and can be stored in a separate storage. It will be accessed asynchronously from internet access so the computing process will speed up.

II. Proposed Method

A. *Workflow for Model Training Initiation*

As we can see in the Figure 1, the purpose of image labeling is for training and evaluating the model so the exact location of the object that must be detected. After that, the entire image will be entered into a TFRecord file which is useful for the training process. TFRecord is a file that describes the data needed during the training and testing phases of the TensorFlow Object Detection API [9].

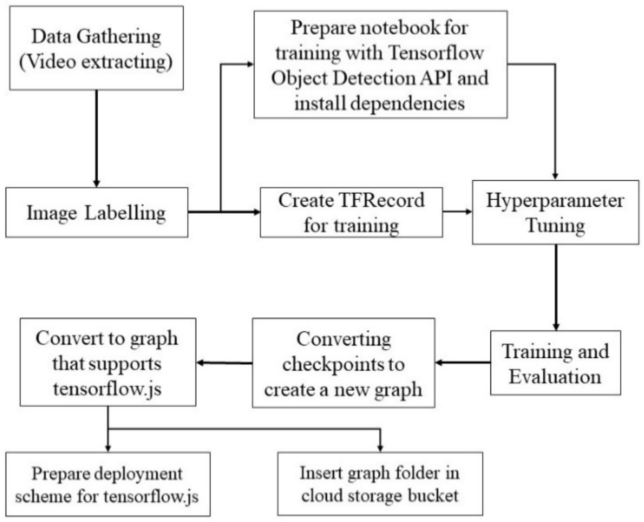

Data Gathering (Video extracting)
Prepare notebook for training with Tensorflow Object Detection API and install dependencies
Image Labelling
Create TFRecord for training
Hyperparameter Tuning
Convert to graph that supports tensorflow.js
Converting checkpoints to create a new graph
Training and Evaluation
Prepare deployment scheme for tensorflow.js
Insert graph folder in cloud storage bucket

Fig. 1. Proposed Workflow

Each different model will have a unique behavior for each different set of images. For this reason, hyperparameter tuning is needed so that when real training is carried out on a large set of images, the model will still fit [10]. After getting the right parameters for training and evaluating the model, the training process is carried out and of course will produce a new graph which will then be converted and deployed to TensorFlow.js [11]. Finally, cloud storage is set up to store a model so that the website can access the graph file after it has been released.

B. *Web Application Architecture*

From Figure 2, we can see that after the new model had been deployed, user can choose the desired model for object detection after the model deployment had done. After that, the UAV onboard camera will be used to capture images which can send directly to the website in real time [12]. After the web gets the model requested by the user and has the video from UAV webcam, then the web will send request to the TensorFlow Object Detection API to extract the image. The processing uses a model that has been stored in cloud storage so that the location of the bounding box where the detection is located is obtained. With a cross-service operation plan, a web architecture that relies on cloud storage will be chosen. As a result, this web will use IBM COS. This is done to better support cloud operations and asynchronous computing, thereby reducing long computing time [13].

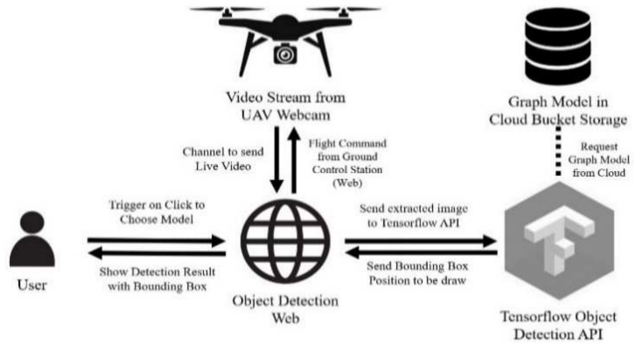
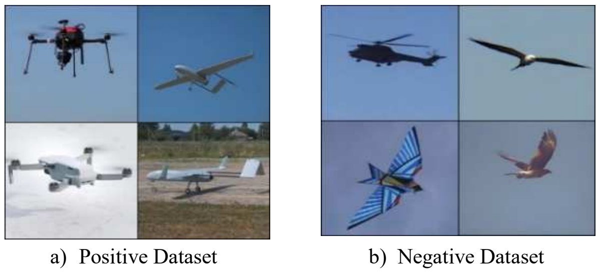

Video Stream from UAV Webcam
Graph Model in Cloud Bucket Storage
Channel to send Live Video
Flight Command from Ground Control Station (Web)
Request Graph Model from Cloud
Trigger on Click to Choose Model
Send extracted image to Tensorflow API
User
Show Detection Result with Bounding Box
Send Bounding Box Position to be draw
Object Detection Web
Tensorflow Object Detection API

Fig. 2. Proposed Architecture

Next, the result will be sent in the form of an array to the web and the web will draw a bounding box. Later, the results of this detection will be used so that the web application on the website can send commands to the UAV in real-time to carry out a mission.

C. *Datasets*

For a training purpose to make a new model, there are 12000 images that have been collected from YouTube, with the classification of 6000 positive images and the remaining are negative images, can be seen in Figure 3. The positive data includes images of quadrotor and fixed-wing types [14], while the negative data contains images of kites, helicopters, and birds. All datasets have been labeled according to their respective class classifications, namely the "uav" class for positive images and the "non_uav" class for negative images. Labeling is done on a flawless, flawless image so blurry images will not be used. This dataset divided into 96% Train Dataset, 2% Dev Dataset, and 2% Test Dataset for evaluation.



a) Positive Dataset b) Negative Dataset

Fig. 3. Example of Dataset

This dataset contains images that are in the Portable Network Graphic (PNG) format. This format is used because in the machine learning process, the training will use the RGB (Red, Green, Blue) color approach. With this scheme, the training process will be more accurate and can detect each pixel well [15].

D. *Pre-trained Models*

There are so many SSD pre-trained model options available in TensorFlow V2 [4]. A comparison will be made on three different models, such as SSD MobileNet, SSD ResNet50, and SSD ResNet101. MobileNet and ResNet pre-trained model was chosen because the smaller layer sizes show the faster speed. These three models were chosen because only 12GB of GPU memory of computation that can be used for training the model.

From Table II, we can see that, the results with FPN [16] are measurements made on training for familiar objects. Therefore, these selected models will be further investigated for training using the dataset described in the previous section.

TABLE II. TENSORFLOW 2 DETECTION MODEL ZOO [4]

Model name	Speed (ms)	COCO mAP	Outputs
SSD MobileNet V1 FPN 640x640	48	29.1	Boxes
SSD MobileNet V2 FPNLite 320×320[a]	22	22.2	Boxes
SSD MobileNet V2 FPNLite 640×640	39	28.2	Boxes
SSD ResNet50 V1 FPN 640×640 (RetinaNet50) [a]	46	34.3	Boxes
SSD ResNet50 V1 FPN 1024×1024 (RetinaNet50)	87	38.3	Boxes
SSD ResNet101 V1 FPN 640×640 (RetinaNet101) [a]	57	35.6	Boxes
SSD ResNet101 V1 FPN 1024×1024 (RetinaNet101)	104	39.5	Boxes
SSD ResNet152 V1 FPN 640×640 (RetinaNet152)	80	35.4	Boxes
SSD ResNet152 V1 FPN 1024×1024 (RetinaNet152)	111	39.6	Boxes

a. Selected Model to be trained

E. *Hyperparameter Tuning*

In determining the right parameters so that the resulting model can work optimally, the random search method is used because random experiments are more efficient than grid experiments for hyper-parameter optimization in the case of several learning algorithms on several data set[17]. Hyperparameter tuning is using GPU computing environment with a total of 100 000 training steps, a momentum scale of 0.9 [18] and an initial learning rate of 0.04. The selected number can be seen in the cells marked in Table III with the letter (a). The selection of this number is based on the highest precision and recall with the lowest loss so the maximum training will be generated.

TABLE III. HYPERPARAMETER TESTING

Model	Batch Size	Warm up Step	Warm up Rate	Loss	Prec	Rec
SSD MobileNet V2 FPNLite 320×320	12	5000[a]	0.013[a]	0.056	0.0347	0.045
	12	100000	0	0.077	0.0385	0.058
	12	5000	0.04	0.038	0.039	0.039
SSD ResNet50 V1 FPN 640×640 (RetinaNet 50)	12	5000	0.013	0.033	0.0232	0.028
	12	100000	0	0.176	0.088	0.132
	12	5000[a]	0.04[a]	0.057	0.0485	0.053
SSD ResNet101 V1 FPN 640×640 (RetinaNet 101)	8	5000[a]	0.013[a]	0.047	0.0302	0.039
	8	100000	0	0.25	0.125	0.188
	4	5000	0.04	0.122	0.081	0.102

a. Selected Parameter Value to be trained

F. *Hardware and Resources*

For training phase, Google Colab provides a single 12GB NVIDIA Tesla K80 GPU that can be used up to 12 hours continuously. The testing phase is using a Computer with Intel® Core™ i7-9750H CPU @ 2.60GHz, with 8.192 MB RAM. For Training. The deployment process is using Heroku with React.js. Detection process is using the model that saved in IBM Cloud with single Cloud Object Storage Service Instance in Lite scheme [19].

III. RESULTS AND ANALYSIS

Loss is the distance between the detection result and ground labeled for image detection [20]. The higher loss number, the more errors will be generated during training. From the Figure 4, MobileNet has a loss rate of only 0.09, while ResNet50 has a loss rate, around 0.101, and finally ResNet 101 has the highest loss rate of all, which is 0.142. MobileNet V2 has the lowest level of loss value so the detection results with the ground label are not much different. On the other hand, ResNet101 has the highest loss rate, so many errors can be seen during detection.

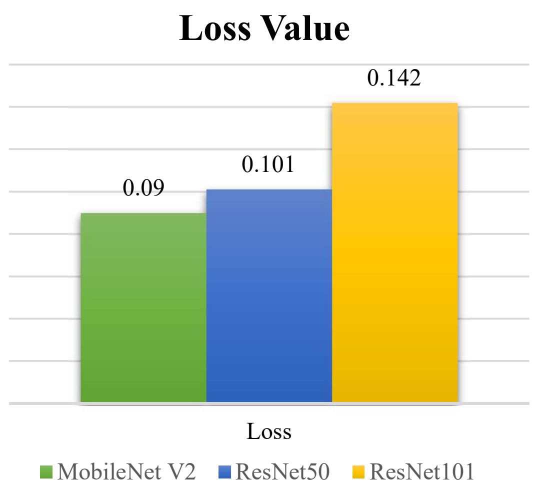


Fig. 4. Loss Value for all Models

Some metrics is used to measure the Object Detection Models. A true positive is an outcome where the model correctly predicts the positive class. Similarly, a true negative is an outcome where the model correctly predicts the negative class. A false positive is an outcome where the model incorrectly predicts the positive class. And a false negative is an outcome where the model incorrectly predicts the negative class. To calculate the precision of the object detection results, it can be seen in equation (1).

$$Precision = \frac{TP}{TP+FP} \tag{1}$$

TP means True Positive, and FP means True Negative. The greater the precision value, the higher the object detection accuracy in recognizing and vice versa.

The model can be said to be successful if the precision number is greater than 80% and the successful recall number is greater than 70% and it is categorized as "Knowledge High" [21]. IoU is also called the Intersection of Union, which means the intersection of the detection results with the square that should be at the same time, also known as the detection confidence.

In the detection of large and medium-sized objects as we can see in Figure 5, the difference in results between the three models is not so significant, but the results of ResNet101 rank the lowest. The same thing happened at the scale of IoU = 0.75 and IoU = 0.50, and IoU = 0.50:0.95, again ResNet101 has the lowest precision in performing detection on a certain confidence scale. The precision value for small object detection size should be highlighted here; the MobileNet V2 Model has the highest precision. Therefore, if small-scale UAV object identification is done frequently, the precision should be considered.

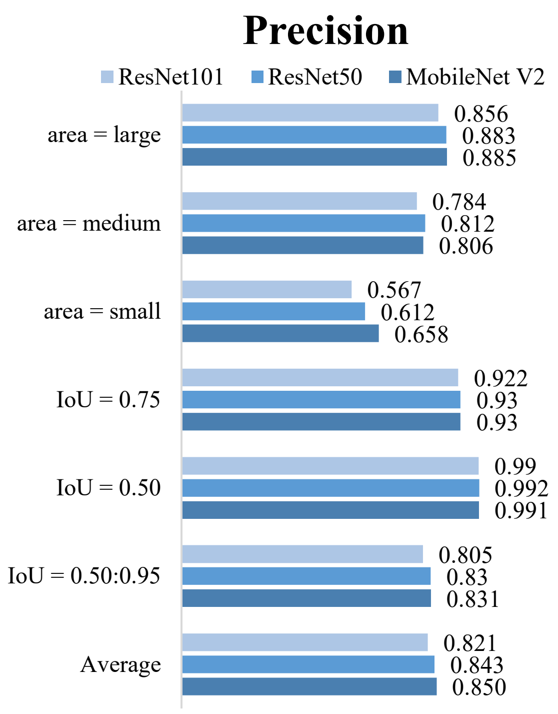


Fig. 5. Precision Value for all Models

To calculate the sensitivity or Recall Value of the object detection results, it can be seen in equation (2).

$$Recall = \frac{TP}{TP+FN} \tag{2}$$

TP means True Positive, and FN means False Negative. The greater the recall value, the higher the sensitivity of object detection in triggering decision making.

MaxDets is the maximum number of detections indicating the maximum number of detections in a single image only. For all MaxDets values as shown in Figure 6, there is no significant difference in detection sensitivity, with ResNet101 having the lowest sensitivity. The sensitivity of detection on large objects, there is also no significant difference. The interesting thing is the recall number in the detection of medium-sized objects, where ResNet50 occupies the highest number even though the difference is not much different from others. Unfortunately, ResNet50 has a lower detection sensitivity than MobileNet V2 for small objects.

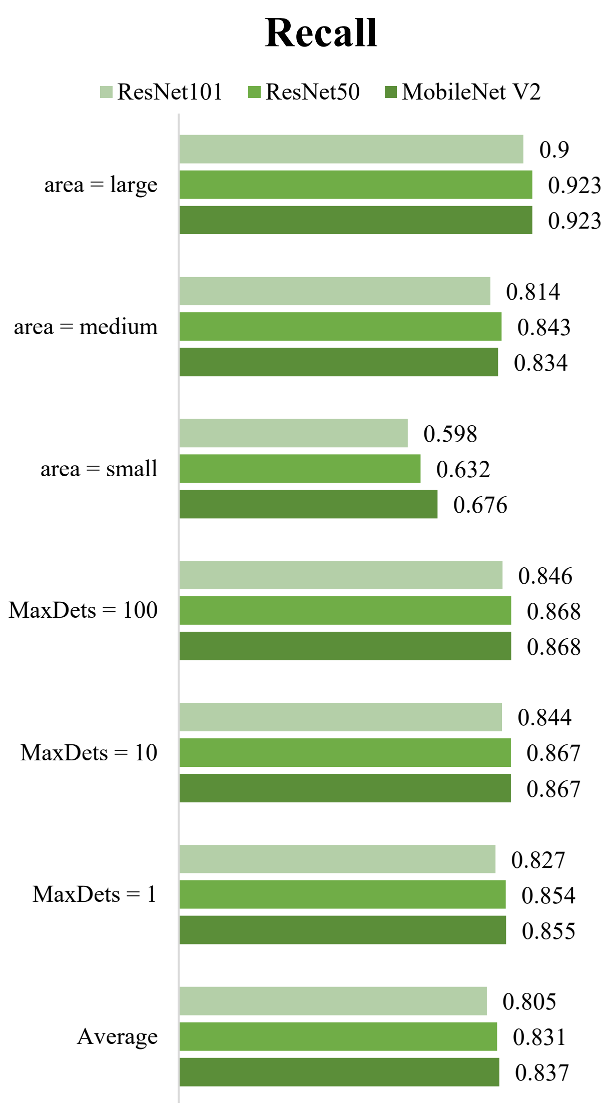
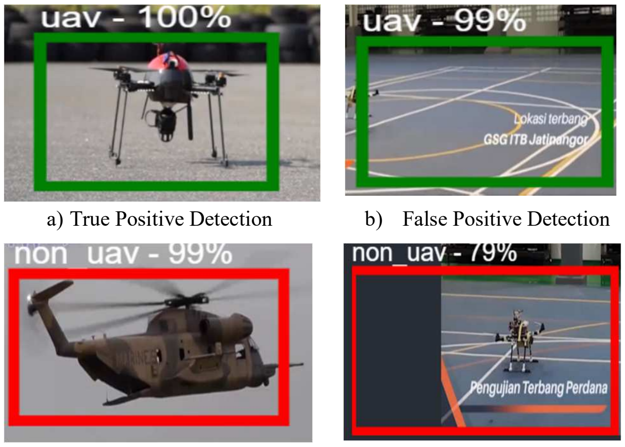


Fig. 6. Recall Value for all Models

The Bounding Box on the Web is drawn using the HTML Canvas Rendering Context, usually symbolized by "ctx". The created line follows the number that appears at the detection angle of the tensorflow.js layer generated on the web backend [22]. The screenshot of UAV detection on the web can be seen in Figure 7.



a) True Positive Detection b) False Positive Detection

c) True Negative Detection d) False Negative Detection

Fig. 7. Result of UAV Detection in Web

To deploy a new model to a display page as shown in Figure 8, Runtime Testing is based on two stages, such as Load Model Time and Average Detection Time, measured in seconds. Load Model Time is measured based on the time span required by the backend to access the graph model from cloud storage with all its preparations. It can be seen in Table IV, MobileNet V2 has the fastest processing time from Cloud Storage to the website backend. This is influenced by the size of the extracted bin file from the training results, where the size of the converted graph is the smallest than the others.

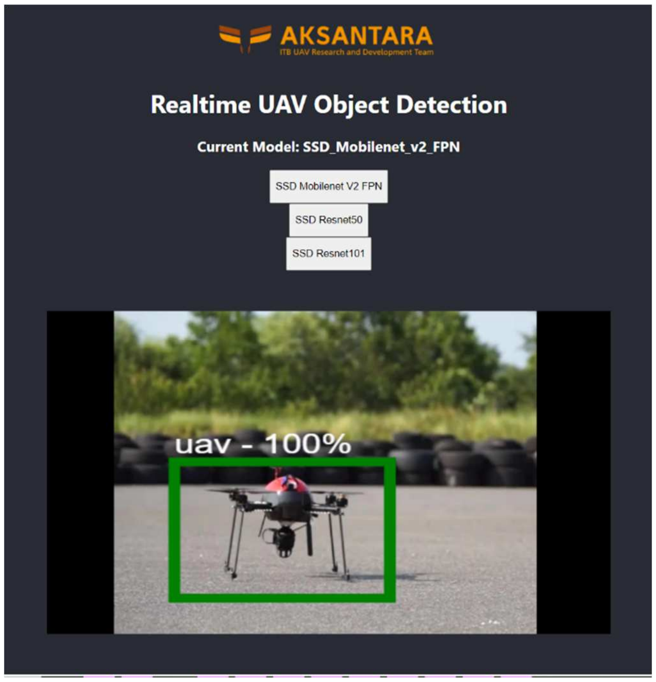

AKSANTARA
ITB UAV Research and Development Team
Realtime UAV Object Detection
Current Model: SSD_Mobilenet_v2_FPN
SSD Mobilenet V2 FPN
SSD Resnet50
SSD Resnet101
uav - 100%

Fig. 8. Web Interface of UAV Object Detection using a New Model

TABLE IV. RUNTIME MODEL IN WEB MEASUREMENT

Runtime Model Measurement (s)	**Model**		
	MobileNet V2	***ResNet50***	***ResNet101***
Load Model Time	12.27[a]	240.94	209.21
Average Detection Time	1.5[a]	17.47	25.90

a. Minimum Value

Furthermore, Average Detection Time is measured based on the time span of layer-by-layer processing that exists in each pre-trained model and the average detection time is obtained from ten times. Again, the results show that MobileNet V2 has the fastest time span in performing layer-by-layer detection so that the detection results on the web backend finally released.

IV. CONCLUSION

In this paper, a new web application approach has been generated to obtain an optimal object detection model to overcome the constraint in terms of parameters, computational environment, and other various constraints. The new pre-trained model also runs well after the deployment on the web and can classify various flying objects from the generated bounding box class. In the web application approach, SSD MobileNet v2 FPN precision value is 0.85 and recall value is 0.837. Compared with SSD MobileNet v2 with onboard processing, the precision value is only 0.586 and the recall value is only 0.622. These are proofing that the precision and recall values in the web application approach are better than UAV object detection using onboard processing. However, the processing runtime on the web application is slower than the onboard processing of the UAV.

V. FUTURE WORKS

Runtime performance of detection processing can be improved by exploring another web architecture, pre-trained models, and method. The web application approach can be further developed to be integrated with Ground Control Station of the UAV. The web application also allows the detection and classification of other flying objects instead of detecting only "uav" and "non_uav" objects. UAVs can also coordinate with each other from multiple camera angles and transmit object detection results to multiple computers and devices in real time. With runtime improvement, this web application can also be integrated to measuring the UAV speed and anti-theft detection.

ACKNOWLEDGMENT

I would like to thank P3MI STEI ITB for providing sponsorship and funding. I also want to thank Aksantara ITB for giving advice so that this research can run well.

REFERENCES

[1] H. Shakhatreh *et al.*, "Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[2] M. M. Naushad Ali, M. Abdullah-Al-Wadud, and S. L. Lee, "Moving Object Detection and Tracking Using Different Image Processing Approaches," *International Journal of Pure and Applied Mathematics*, vol. 321–324, no. 6, pp. 1200–1204, 2013.

[3] W. Liu *et al.*, "SSD : Single Shot MultiBox Detector," *European conference on computer vision*, pp. 21–37, 2016.

[4] TensorFlow, "TensorFlow 2 Detection Model Zoo," *GitHub*, 2021. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md (accessed Jun. 08, 2021).

[5] M. K. Makirin, L. M. Wastupranata, and A. Daffa, "Onboard Visual Drone Detection for Drone Chasing and Collision Avoidance," *AIP Conference Proceedings*, vol. 2366, 2021.

[6] S. Vaddi, D. Kim, C. Kumar, S. Shad, and A. Jannesari, "Efficient Object Detection Model for Real-time UAV Application," *Computer and Information Science*, vol. 14, no. 1, p. 45, 2021.

[7] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[9] O. Campesato, *TensorFlow 2 Pocket Primer*. Mercury Learning and Information, 2019.

[10] M. Claesen and B. de Moor, "Hyperparameter Search in Machine Learning," *The XI Metaheuristics International Conference*, pp. 10–14, 2015.

[11] D. Smilkov *et al.*, "TensorFlow.js: Machine Learning for the Web and Beyond," *Proceedings of the 2nd SysML Conference*, 2019.

[12] S. N. Kizar and G. S. R. Satyanarayana, "Object Detection and Location Estimation using SVS for UAVs," *International Conference on Automatic Control and Dynamic Optimization Techniques, ICACDOT 2016*, pp. 920–924, 2017.

[13] A. Levin *et al.*, "AIOps for a Cloud Object Storage Service," *Proceedings - 2019 IEEE International Congress on Big Data, BigData Congress 2019 - Part of the 2019 IEEE World Congress on Services*, pp. 165–169, 2019.

[14] A. Jaimes, S. Kota, and J. Gomez, "An Approach to Surveillance an Area Using Swarm of Fixed Wing and Quad-Rotor Unmanned Aerial Vehicles UAV(S)," *2008 IEEE International Conference on System of Systems Engineering, SoSE 2008*, pp. 8–13, 2008.

[15] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning Rich Features from RGB-D Images for Object Detection and Segmentation," *Lecture Notes in Computer Science*, vol. 8695 LNCS, pp. 345–360, 2014.

[16] X. Li, T. Lai, S. Wang, Q. Chen, C. Yang, and R. Chen, "Feature Pyramid Networks for Object Detection," *Proceedings - 2019 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking, ISPA/BDCloud/SustainCom/SocialCom 2019*, pp. 1500–1504, 2019.

[17] J. Bergstra and Y. Bengio, "Random Search for Hyper-Parameter Optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[18] S. Ruder, "An Overview of Gradient Descent Optimization Algorithms," pp. 1–14, 2016.

[19] L. Stadtmueller, "Which Cloud Storage Service Delivers the Performance You Need ? Comparing IBM Cloud Object Storage and Amazon S3 An Executive Brief Sponsored by IBM," 2016.

[20] S. Jiang, H. Qin, B. Zhang, and J. Zheng, "Optimized Loss Functions for Object detection: A Case Study on Nighttime Vehicle Detection," *arXiv*, pp. 1–18, 2020.

[21] F. Habryn, *Customer Intimacy Analytics: Leveraging Operational Data to Assess Customer Knowledge and Relationships and to Measure their Business Impact*. 2012.

[22] N. Renotte, "Deployable Computer Vision App," 2020. https://github.com/nicknochnack/DeployableComputerVisionApp (accessed Jun. 02, 2021).