Techno-Science Research Journal 9(2) (2021) 60-66

Content list available at ITC

Techno-Science Research Journal

Journal Homepage: http://techno-srj.itc.edu.kh/

Techno-Science Research Journal

Helipad Detection for UAV based on YOLOv4 Transfer Learning Model

Vanyi Chao*, Sarot Srang, Morokot Sakal, Chivorn Keo

Dynamics and Control Laboratory, Department of Industrial and Mechanical Engineering, Institute of Technology of Cambodia, Russian Federation Blvd., P.O. Box 86, Phnom Penh, Cambodia.

Received: 01 June 2021; Accepted: 21 October 2021; Available online: December 2021

Abstract: *Humans have fast and accurate visual system that allows them to perform complex tasks like driving with a little consciousness. Without visual system, UAV cannot do complex tasks like humans. When UAVs are equipped with visual system and trained with fast and accurate model, they will be able to carry out even more complex tasks such as autonomous landing. Computer vision is a technique suitable for UAV visual system. In this paper, we consider a computer vision technique that uses a deep learning model to recognize the landing site (Helipad). We conducted an experiment of training the deep learning model to recognize Helipad. In order to land on the desired site safely, we proposed a detection method based on YOLOv4-tiny transfer learning model to detect the Helipad in real time. The digital images were used as training data in order for the model to learn and gain a high-level recognizing object that exists in an image. The data collection to train the model was delimited by collecting them from the internet and video's snapshot. The annotation tool was used in order to draw ground truth box for 184 training samples and 57 testing samples with 1 class. The YOLOv4-tiny model was trained on darknet framework, using YOLOv4-tiny pre-trained weight and the described input data. After training was completed with GPU acceleration, the best weights were saved in order to use in OpenCV's Deep Neural Network (DNN). The model was first validated with testing images, tested on videos and finally real-time streaming video in order to investigate its performance. We used Intersection over Union (IOU), precision, recall, miss rate and mean Average Precision (mAP) as the evaluation metrics as well as Loss-function visualization to visualize and analyze the model's performance. During real-time streaming video, we investigate frames per second (FPS) and inference time. Finally, the experimental results show that the detection method can accurately detect the Helipad in real-time video.*

Keywords: Computer vision; Real-time object detection; Daarknrt; OpenCV; You Look Once (YOLO)

1. INTRODUCTION

When humans take a glance at an image, they will instantly know what and where the objects are in that image and how they interact. The human visual system is fast and accurate that allows them to perform complex tasks like driving with a little consciousness. This is a motivation why researchers try to find a way to equip robots or machines with visual system.

General purpose object detection should be fast, and accurate. Since the introduction of neural networks, detection frameworks have become increasingly fast and accurate. However, most detection methods are still constrained to a small set of objects [1].

The early approach of the object detection model is called, sliding window approach. Most successful object recognition systems rely on binary classification, deciding only if an object is present or not, but not providing information on the actual object location. To perform localization, one can take a sliding window approach, but this strongly increases the computational cost because the classifier function has to be evaluated over a large set of candidate sub-windows [2].

The later approach to deal with computational cost was to extract some region proposals (boxes) and check if any of those boxes contain any object. The approach is called Region based Convolutional Neural Network (R-CNN), which did selective search so it used some deterministic algorithm that extracted about 2 thousand potential bounding boxes for an image before running through CNN. However, the model is still slow but has less error in localization. Later on, several improved versions have been published, refer to Fast R-CNN and Faster R-CNN [3].

* Corresponding author: Vanyi Chao
E-mail: chaovanyi45@gmail.com; Tel: +855-69 802 410

To deal with the term of speed, another approach called YOLO was introduced by [4]. YOLO performed frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. YOLO was the first object detection network to combine the problem of drawing bounding boxes and identifying class labels in one end-to-end differentiable network. Although YOLO is the fastest, it struggles to localize objects correctly compared to Fast R-CNN. There are three improved versions of YOLO with the same author [4,5], and the last version of YOLOv4 [6].

As the computing power of modern Central Processing Unit (CPU) and Graphics Processing Unit (GPU) rapidly increases in the last few years, that made it possible to train a new neural network faster using Nvidia GPU and run multiple neural networks in parallel on a smaller computer like Jetson nano for object detection.

Landing an UAV is a very challenging problem. Pilots have to spend a huge amount of time practicing landing technique in order to minimize any risk that is involved during the landing phase.

2. METHODOLOGY

2.1 Existing methods

Refer to [7], the author proposed an approach to detect helipad in order to achieve high accuracy of pose measurements of a real-time application. The algorithm consists of 3 main steps, concerning the helipad detection and pose estimation, such as image acquisition and preprocessing, helipad mark extraction and corner detection of the helipad.

Rungta et al. [8] proposed a method to autonomously detect Helipads in real time. The author used the method called modified Image-Based Visual Servoing (IBVS). Once the tracking starts the modified IBVS method starts publishing velocity to guide the quadrotor onto the helipad.

However, those proposed approaches do not involve machine learning or deep learning model. Deep learning has many potential applications [9,10] and it has the ability to detect many types of Helipads and many classes. The reasons deep learning is suitable for this particular task are: it can detect many types of Helipads (Every image that has a sign of a circle around letter H), and use the same model to detect more classes (our future work) such as obstacles.

2.2 Proposed method

This paper considers using YOLOv4-tiny transfer learning model to detect Helipad in real time. It is also the best trade-off model for our application and simple to work with, compared to other models [6].

This paper is organized as follows: In section 2, the architecture of YOLOv4 is illustrated and we explain how the Helipad images were prepared before training. The Loss function is derived to visualize how good the model is. In section 3, the evaluation metrics are presented to validate the results. Section 4 illustrates how the transfer learning is done and the result is discussed. Finally, the conclusion is drawn in section 5.

2.3 YOLOv4 architecture

Fig. 1 illustrates the combined architecture from different methods which is divided into Backbone, Neck and Head. YOLOv4's architecture is composed of Cross Stage Partial Network (CSPDarknet53) [11] as a backbone, spatial pyramid pooling [12] as additional module, Path Aggregation Network (PANet) [13] as the network neck and finally YOLOv3 [5] as the network head. CSPDarknet53 is a novel backbone that can enhance the learning capability of CNN. The spatial pyramid pooling block is added over CSPDarknet53 to increase the receptive field and separate out the most significant context features. The PANet is used as the method for parameter aggregation for different detector levels.

2.4 YOLOv4 data augmentation

The important part that YOLOv4 has improved is in data preparation. Data augmentation is really important in computer vision technique. The image augmentation in YOLOv4 includes crop, rotation, flip, hue, saturation, exposure, aspect, mix up, cut mix, mosaic and blur. These actions are done programmatically in the YOLOv4 model.

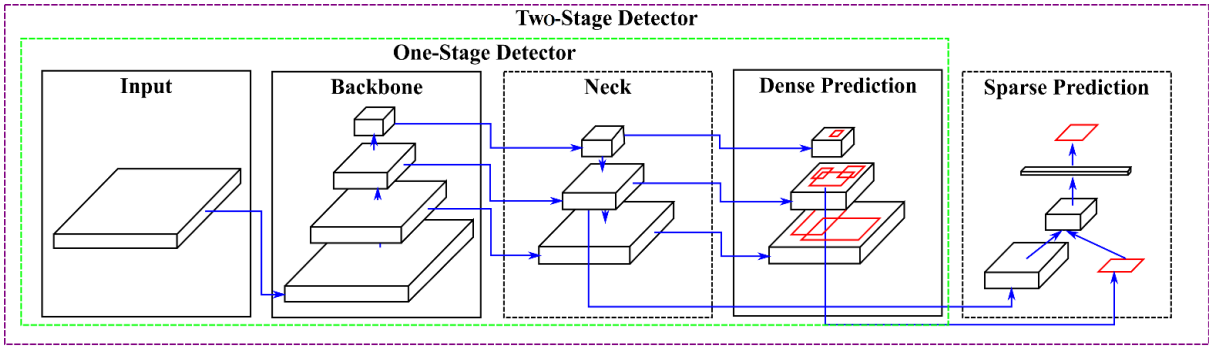


Fig. 1. YOLOv4 combined architecture

2.5 Image data pre-processing

Helipad images are collected from the internet and video's snapshots. To enhance the diversity and richness of the experimental data, the images are collected by using augmentation technique in terms of scaling, flopping, changing brightness, blurring and others. We collected 241 images which were randomly split into 76% for training samples. Thus, we got 184 images for training and 57 images for validation. We collected random size of images which were later resized to 416×416 image size to match the network dimensions. The training and validation images are labelled by using an annotation tool, called LabelImg [14], before training (Fig. 2).

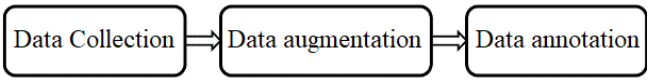


Fig. 2. Image pre-processing

2.6 Loss function

Loss function defines how the prediction is good or bad. It plays a very important role in any statistical model. Loss function defines an objective which the performance of the model is evaluated against and the parameters learned by the model are determined by minimizing a loss function.

Deep learning discovers a very complicated structure in large datasets by using optimization algorithms to optimize objective function and then the internal parameters, which are used to represent in each layer, are updated in order to minimize the loss function. Therefore, object detection problems can be cast as the optimization of the loss function requiring minimization with respect to all parameters. In another word, when the parameters are updated, we hope that the loss function will converge to zero for the best model.

The YOLOv4 loss function consists of three parts such as regression loss, classification loss and confidence loss. Classification loss and confidence loss remain the same as the YOLOv3 model. However, Complete Intersection over Union (CIoU) [15] is used to replace the Mean Squared Error (MSE) to optimize the regression loss (Fig. 3).

The CIoU loss can be defined as

$$Loss_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \qquad \text{(Eq. 1)}$$

where:

$\rho^2(b, b^{gt})$ = the Euclidean distance between the center points of the prediction box and the ground truth box.

c = the diagonal distance of the smallest closed area that can simultaneously contain the prediction box and the ground truth box, shown in Fig 3.

α = trade-off parameter and v measures the consistency of aspect ratio.

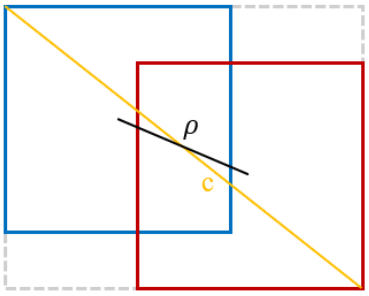


Fig. 3. Complete Intersection over Union (CIoU)

And α and v can be defined as

$$\alpha = \frac{v}{(1 - IoU) + v} \qquad \text{(Eq. 2)}$$

$$v = \frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^2 \qquad \text{(Eq. 3)}$$

The confidence loss can be defined as

$$\begin{aligned} Loss_{conf} = &-\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[\hat{C}_i \log(C_i) \\ &+ (1 - \hat{C}_i)\log(1 - C_i)] \\ &- \lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{noobj}[\hat{C}_i \log(C_i) \\ &+ (1 - \hat{C}_i)\log(1 - C_i)] \end{aligned} \qquad \text{(Eq. 4)}$$

The classification loss can be defined as

$$\begin{aligned} Loss_{cls} = -\sum_{i=0}^{S^2} I_{ij}^{obj} \sum_{c \in classes} &[\hat{p}_i(c)\log(p_i(c)) \\ &+ (1 - \hat{p}_i(c)\log(1 - p_i(c))] \end{aligned} \qquad \text{(Eq. 5)}$$

Finally, the total loss function is:

$$Loss = Loss_{CIoU} + Loss_{conf} + Loss_{cls} \qquad \text{(Eq. 6)}$$

S^2 represents $S \times S$ grids of an image, each grid generates B candidate boxes, each candidate box gets corresponding bounding boxes through the network. If there is no object (noobj) in the box, only the confidence loss of the box is calculated. The confidence loss function uses cross entropy error and is divided into two parts: there is the object (obj) and no object (noobj). The loss of noobj increases the weight coefficient λ, which is to reduce the contribution weight of the noobj calculation part. The classification loss function also uses cross entropy error. When the j-th anchor box of the *i*-th grid is responsible for certain

ground truth, then the bounding box generated by this anchor box will calculate the classification loss function.

3. EVALUATION METRICS

3.1 Intersection over Union (IoU)

The ground truth box is drawn by hand when labelled the image. Thus, it is sure that the object exists in the box. Assuming that it is a blue box in Fig. 4.

The predicted bounding box is generated from the model detector that indicates the location of the object predicted. Assuming that it is a red box in Fig. 4.

The IoU metric consists in calculating how much the predicted bounding box overlaps with the ground-truth one.

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union} \quad \text{(Eq. 7)}$$

Fig. 4 illustrates the overlap between ground truth box and predicted bounding box from poor to perfect. When IoU is closer to one, it is perfect. In contrast, it is poor when IoU is getting closer to zero.

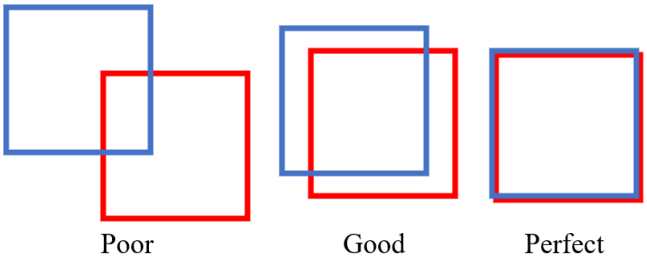


Fig. 4. Intersection over Union

3.2 Precision, Recall and mean Average Precision (mAP)

- **TP** (True Positive) is when the model detects object when the object is present and the $IoU > 0.5$.

- **TN** (True Negative) is when the model does not detect object when the object is absent.

- **FP** (False Positive) is when the model detects object when the object is absent or the $IoU \leq 0.5$.

- **FN** (False Negative) is when the model does not detect object when the object is present. All shown in Fig. 5.

$$Precision = \frac{TP}{TP + FP} \quad \text{(Eq. 8)}$$

$$Recall = \frac{TP}{TP + FN} \quad \text{(Eq. 9)}$$

$$Miss\ rate = 1 - Recall \quad \text{(Eq. 10)}$$

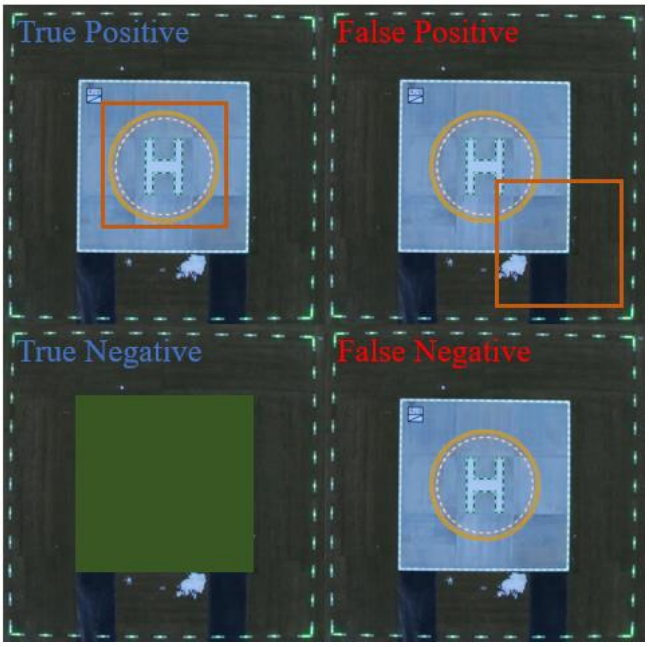


Fig. 5. Predicted bounding generated on Heliport

The average precision (AP) will be the area of a plotted graph of precision versus recall for one class. And the mean average precision (mAP) is the mean value of average precision of all classes. In this paper, we predict only one class, thus, the average precision and mAP is exactly the same.

4. RESULTS AND DISCUSSION

4.1 Environment and requirements

The implementation environment to train the model was a personal laptop, MSI GS63 7RD that runs on Windows 10 operating system with Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz and 16GB of RAM. The model was trained by using a NVIDIA GPU (GTX 1050, 2GB of VRAM). With this environment, the model runs at 22 average frames per second.

The model was trained on a darknet framework which was adapted from [16]. The system is coded in Python language.

Table 1 is the requirement dependencies which have to be installed in order to train the model on GPU and GPU acceleration to save time when training and decrease the time inference when running the model.

Table 1 List of dependencies

Dependencies	Used version	Recommended version
Cmake	3.20	≥ 3.18
CUDA	11.2	≥ 10.2
cuDNN	8.1.1	≥ 8.0.2
OpenCV	4.5.1	≥ 2.4
GPU CC	6.1	≥ 3.0

For OpenCV, we built from source with CUDA and cuDNN enabled. Table 2. shows the requirement VRAM during training with batch size of 64 and 416x416 input image size. Since the computer mentioned in 4.1 has only the VRAM of 2GB about 1907 MiB, the YOLOv4-tiny has been chosen as the training model configuration and it is possible to use 8 subdivisions.

Table 2 VRAM requirement during training (MiB)

Subdivisions	64	32	16	8	4
YOLOv4	4236	6246	?	?	?
YOLOv4-tiny	814	956	1321	1752	2770
YOLOv4-tiny-3l	830	1085	1282	1862	2982

4.2 Training

Since YOLOv4-tiny was originally created to be trained on a COCO dataset containing 80 classes, some small changes had to be made to the structure of YOLO. This was achieved by changing the number of filters and classes that existed in each YOLO layer. It is also necessary to change the number of maximum batches and steps.

$$filters = (4 + 1 + C) * 3 \quad \text{(Eq. 11)}$$

$$max_{batches} = 2000 * C \quad \text{(Eq. 12)}$$

$$Steps = 0.8 * max_{batches}, 0.9 * max_{batches} \quad \text{(Eq. 13)}$$

where C is the number of classes.

Table 3 Hyper parameters used during training

Hyper parameters	Value
Max batches	4000 (minimum 4000)
Batch	64
subdivisions	8
Weight decay	0.005
momentum	0.9
Learning rate	0.00261
Steps	3200,3600

Since we have only one class for landing site, we got the number of $filters = 18$. The maximum batches should not be less than 4000 even $C = 1$, thus, we choose $max\_batches = 4000$.

Every layer of YOLOv4-tiny uses leaky ReLU as the activation function except for the last 2 convolutional layers of [yolo].Eq. 11, Eq. 12, Eq. 13, and Table 3 are modified and saved in yolov4-tiny-train.cfg and yolov4-tiny-test.cfg, shown in Fig. 6.

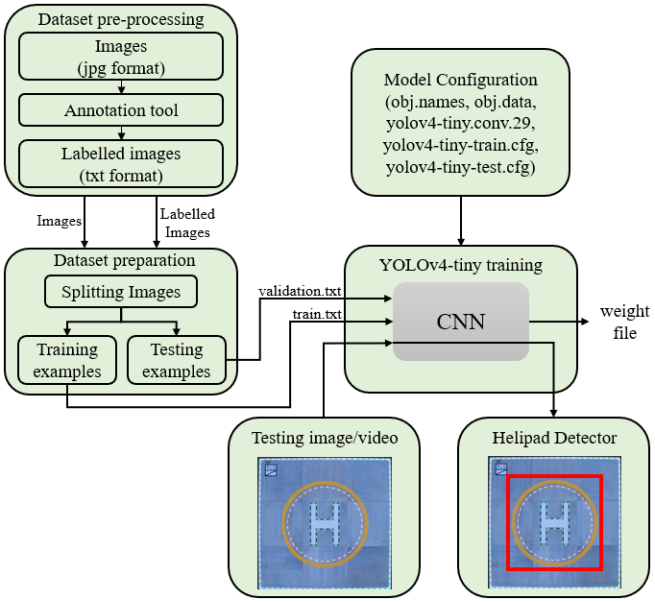


Fig. 6. Flow process of YOLOv4-tiny transfer learning

While training, we use the pre-trained weight yolov4-tiny.conv.29. The weight file is saved once every 1000 iterations. After the training is complete, the final weight and the best weight are saved. After that, the best weight is used to replace yolov4-tiny-conv.29 for testing images and video.

4.3 Results

Fig. 7 illustrates the average loss function and mAP for 4000 iterations. The result shows that the average loss converges to zero. At the 4000th iteration the average loss is 0.0138 which shows that the model is really good. Moreover, the mAP at the end of training is 100% which is the best for one class detection.

We use 57 testing examples to validate the model. The result shows in Table 4 and 5. We validated with $IoU = 0.5$ and $IoU = 0.75$. Thus, mAP is calculated with different IoU. We got 100% mAP at $IoU = 0.5$ and 94.48% mAP at $IoU = 0.75$.

Table 4 Validation with $IoU = 0.5$, $(mAP@0.5) = 1$

TP	FP	FN	Precision	Recall	Miss rate	AP
57	3	0	0.95	1	0	1

Table 5 Validation with $IoU = 0.75$, $(mAP@0.75) = 0.944$

TP	FP	FN	Precision	Recall	Miss rate	AP
54	6	3	0.9	0.95	0.05	0.9448

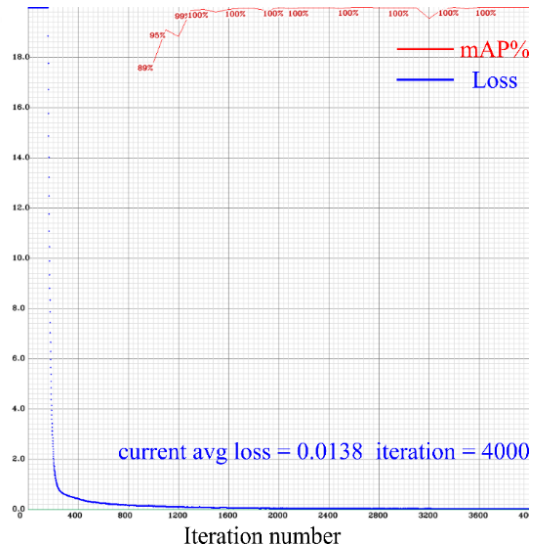
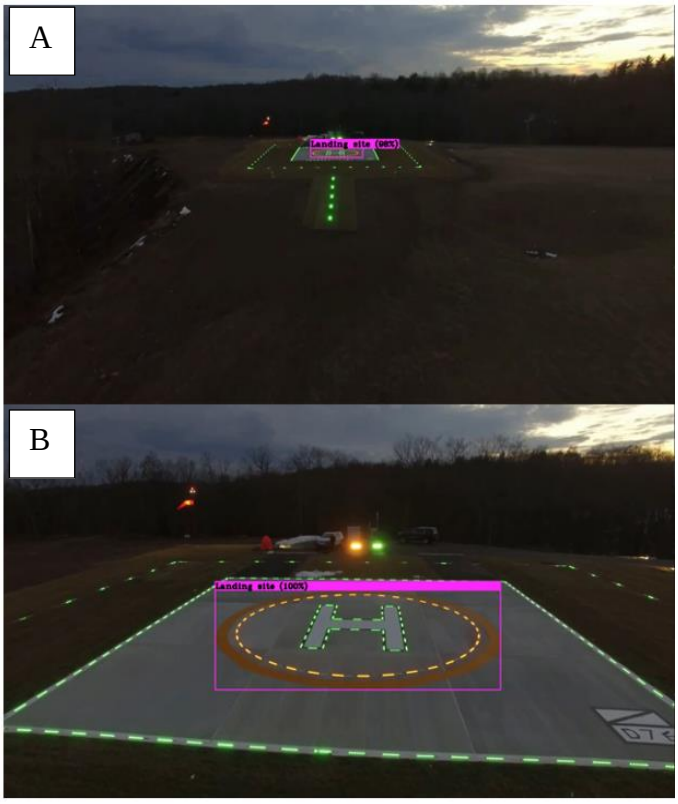


Fig. 7. Visualization of Loss (blue) and mAP (red)



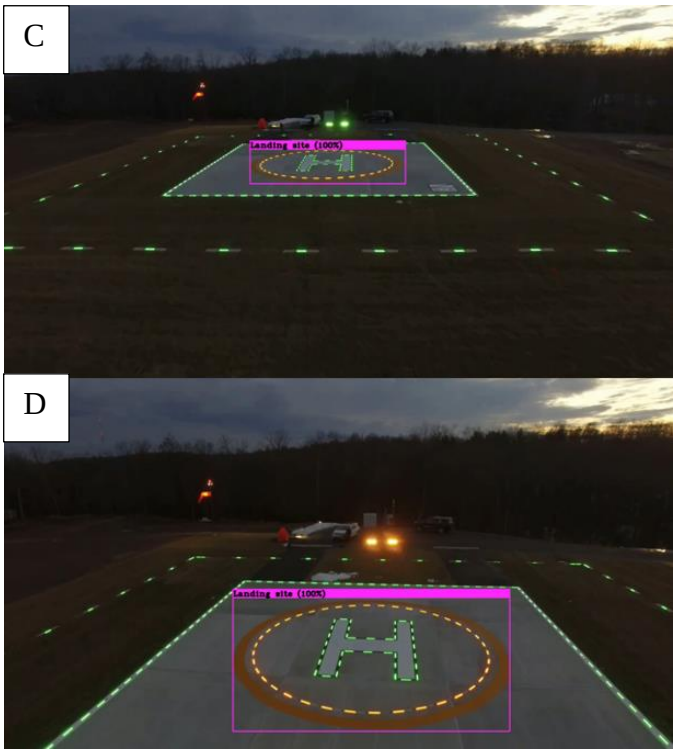


Fig. 8. Helipad detection of 4 frames extracted from a testing videod (A, B C and D).

Fig. 8 shows Helipad detection of some frames in a testing video. The model can correctly detect the Helipad from far away (small object) with 98% confidence and 100% confidence for others.

5. CONCLUSIONS

In this paper, the Landing site (Helipad or Heliport) detection based on YOLOv4-tiny transfer learning model is proposed. The model looks at the image only once to perform detection. The testing images prove that the model is good at detecting the landing site with 0% of miss rate, 100% mAP at IoU=0.5 and 5% miss rate, 94.48% mAP at IoU=0.75. Moreover, the testing video shows that the model accurately detects the landing site with the speed of 22 average FPS. For future work, we should consider implementing Helipad detection on an actual UAV and using the model to detect more classes.

REFERENCES

[1] Redmon, J., & Farhadi, A. (2017). Yolo9000: Better, faster, stronger. Proceedings of the IEEE conference on computer vision and pattern recognition, 7263–7271.

[2] Lampert, C. H., Blaschko, M. B., & Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient

subwindow search. 2008 IEEE conference on computer vision and pattern recognition, 1–8.

[3] Girshick, R. (2015). Fast rcnn. Proceedings of the IEEE international conference on computer vision, 1440–1448.

[4] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, realtime object detection. Proceedings of the IEEE conference on computer vision and pattern recognition, 779–788.

[5] Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767. Ren, S.,

[6] Bochkovskiy, A., Wang, C.Y., & Liao, H.¬Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

[7] Patruno, C., Nitti, M., Stella, E., & D'Orazio, T. (2017). Helipad detection for accurate uav pose estimation by means of a visual sensor. International Journal of Advanced Robotic Systems, 14(5), 1729881417731083.

[8] Rungta, A., Soni, Y., Agarwal, P., Ghosh, B., & Kumar, S. (2020). Realtime and autonomous detection of helipad for landing quadrotors by visual servoing. arXiv preprint arXiv:2008.02236.

[9] Zhong, M., & Meng, F. (2019). A yolov3 based nonhelmet use detection for seafarer safety aboard merchant ships. Journal of Physics: Conference Series, 1325(1), 012096.

[10] Borngrund, C. (2019). Machine vision for automation of earthmoving machines: Transfer learning experiments with yolov3.

[11] Wang, C.Y., Liao, H.Y. M., Wu, Y.H., Chen, P.Y., Hsieh, J.W., & Yeh, I.H. (2020). Cspnet: A new backbone that can enhance learning capability of cnn. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, 390–391.

[12] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE transactions on pattern analysis and machine intelligence, 37(9), 1904–1916.

[13] Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, 8759–8768.

[14] Tzutalin. (2018). Retrieved from https://github.com/tzutalin /labelImg

[15] Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-iou loss: Faster andbetter learning for bounding box regression.Proceedings of the AAAI Conference onArtificial Intelligence,34(07), 12993–13000.

[16] AlexeyAB. (2020). Darknet. Retrieved from https://github.com/AlexeyAB/darknet