

Oil palm unstripped bunch detector using modified faster regional convolutional neural network

Wahyu Sapto Aji, Kamarul Hawari Bin Ghazali, Son Ali Akbar

Fakulty of Electrical and Electronics Engineering Technology, Universiti Malaysia Pahang, Pekan, Malaysia

Article Info

Article history:

Received Sep 27, 2021

Revised Dec 9, 2021

Accepted Dec 29, 2021

Keywords:

Faster regional convolutional neural network

Object detector

Unstripped bunch

Visual geometry group 16

ABSTRACT

The palm oil processing industry in Malaysia and Indonesia is significant and plays a vital role in the community's welfare. The efficiency of palm oil mills is characterized by the low number of unstripped bunch (USBs), so USB detection is essential in the palm oil production process. So far, USB detection is done manually and is often ignored because it is labor-intensive. We developed a USB detector based on faster regional convolutional neural network with a modified visual geometry group 16 (VGG16) backbone to solve this problem. To see the performance of our proposed USB detector, we compared it to the faster region based convolutional neural networks (R-CNN) USB detector with the VGG16 standard backbone. Based on the validation test, the USB faster R-CNN detector with modified VGG16 can improve the performance of the USB faster R-CNN detection system based on the original VGG 16 backbone. The proposed system can work faster (100% faster) with an mAP value of 0.782 (7.42% more precise) than the USB Detector with the original VGG16. In the training process, the proposed system on the speed parameter has better training parameters, which is 58.9% faster, the total loss is smaller (43.4% smaller), and the proposed system has better best accuracy (98%) than the previous system (93%). Still, it has a smaller overlap bounding box (23.91% less).

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Wahyu Sapto Aji

Faculty of Electrical and Electronics Engineering Technology, University Malaysia of Pahang

Pekan Campus, 26600 Pekan, Malaysia

Email: wahyusa@ee.uad.ac.id

1. INTRODUCTION

One of the causes of losses in palm oil mills is oil palm unstripped bunch (USB). In Malaysia, losses caused by USB are estimated at an average of 0.05% [1]. In some cases where the bunches are not appropriately processed, losses due to USB can be up to 40% [2]. Based on this, it can be concluded that the primary system performance of palm oil mills can be seen based on the presence of USB. Unfortunately, no method can perform USB observations automatically. So far, USB monitoring is still done manually and is often ignored. An automatic USB counting system must be developed to solve this problem as the first step in USB monitoring; a faster region-based convolutional neural networks (R-CNN) object detection system with a modified visual geometry group 16 (VGG16) feature extractor is proposed. The performance of the proposed USB detection system will be compared with the faster R-CNN USB detector system with the original VGG16.

USB is loosely defined as an oil palm fruit bunch that still has oil palm fruit fruitlets. Figure 1 shows an example of a USB, while Figure 2 shows an example of an empty fruit bunch (EFB). EFB itself is in oil palm bunches without sticking fruit, as shown in Figure 2. Some researchers have given their definitions of USB specifically. According to Hassan *et al.*, a USB is an empty bunch with more than 20 fruits still attached [3].

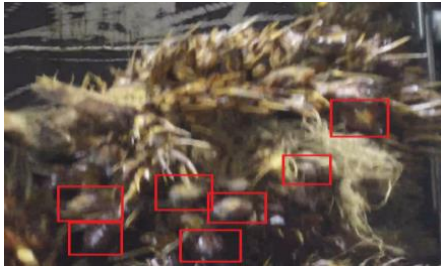


Figure 1. A USB with fruitlets (red box) attached to it



Figure 2. An EFB

2. RELATED WORK

Deep learning has gained much interest in artificial intelligence (AI) applications, especially object detection systems. That is inseparable from the ability of deep learning to recognize objects, where their abilities exceed human abilities [4]. Several researchers realized this deep learning ability and developed an object recognition application based on deep learning. In the health sector, Ghaderzadeh *et al.* take advantage of deep learning for the detection of the coronavirus disease (COVID-19), wherein their research, deep learning-based object recognition can reduce false positives and negatives in the detection and diagnosis of the COVID-19 virus and offers a unique opportunity to provide fast, inexpensive diagnostic services, and safe for the patient [5]. Deep learning applications for object detection in animal husbandry and agriculture also attracted the attention of several researchers. Barbedo *et al.* in 2019, succeeded in developing a deep learning-based cattle detection system using Unmanned aerial vehicle (UAV) [6]. In 2019, Aravind *et al.* developed deep learning-based eggplant disease detection [7]. In a recent effort, Liu *et al.* successfully developed palm tree detection based on faster R-CNN [8]. However, no researcher has yet developed a USB detection system that oil palm mills operators need.

3. RESEARCH METHOD

The research methodology carried out includes several steps. The first step is the creation of the USB dataset; the second step is fine-tuning and original VGG16 modification. After the above process, training and validation are carried out using both the original R-CNN Faster and the modified RCNN faster. The results of the training and validation processes are then compared.

3.1. USB dataset

USB detection research requires the availability of a USB database, and the USB database is not yet publicly available. Therefore, the first step in carrying out this research is to create a USB database. The USB database was compiled using image data obtained from surveillance cameras. This surveillance camera is placed above a USB conveyor.

The custom datasets used are the USB and EFB datasets. The images of the datasets are captured from video surveillance mounted on top of the exhaust conveyor of a palm oil mill. This video is taken from PT. Sawit Arum Madani, a palm oil mill located in Blitar, Indonesia in May 2019. The number of images available is 500. Four hundred sixty images (92%) were used as train data, and 40 (8%) images were used as test data. Figures 3 and 4 respectively show examples of USB and EFB labeling. For USB and EFB classes, the EFB class is 255, and the USB class is 245. LabelImg [9] is used for labeling images.



Figure 3. USB labeling example

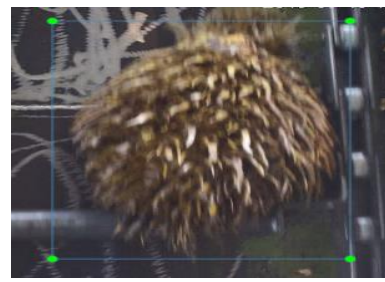


Figure 4. EFB labeling example

3.2. Faster R-CNN

Faster R-CNN is one well-known two-stage object detector [10]. A faster R-CNN object detection network is composed of a feature extraction network which is typically a pre-trained convolutional neural networks (CNN), followed by a proposal network (RPN), which is, as its name suggests, used to generate object proposals, and the second is used to predict the actual class of the object. The final layer is the detection network or classification network, as shown in Figure 5, which depicts faster R-CNN's main component [11]. This study uses a modified VGG16 as a feature extractor.

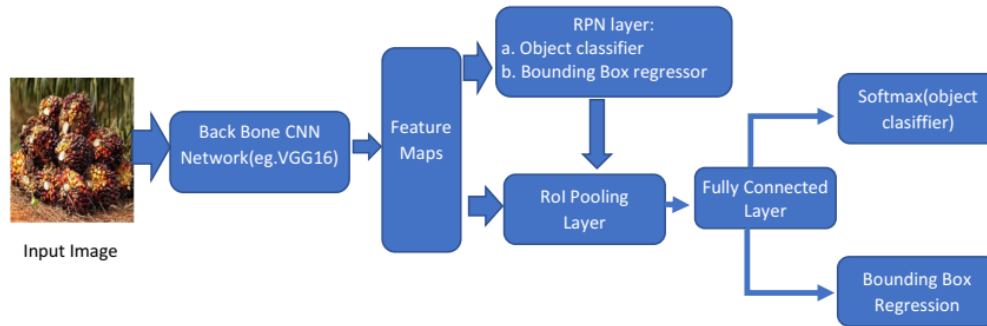


Figure 5. Faster R-CNN basic architecture

3.3. Original VGG16

VGG16 is a convolutional neural network model proposed by Simonyan and Zisserman [12]. The architecture can be seen in Table 1. It consists of five blocks, where there are piles of convolutional layers and a max-pooling layer. That is followed by three fully connected layers and a softmax layer. All convolutional layers are also equipped with ReLU, which stands for rectified linear unit. In the original configuration, VGG16 does not have a batch normalization layer. The original VGG16 model has 136,688,504 parameters, all trainable parameters.

Table 1. Original VGG16 architecture

Block	Layer	Kernel size	Number of kernels	Activation	Padding
1	Conv2D	3×3	64	Relu	Same
	Conv2D	3×3	64	Relu	Same
2	MaxPooling2D				
	Conv2D	3×3	128	Relu	Same
	Conv2D	3×3	128	Relu	Same
	MaxPooling2D				
3	Conv2D	3×3	256	Relu	Same
	Conv2D	3×3	256	Relu	Same
	Conv2D	3×3	256	Relu	Same
	MaxPooling2D			Relu	Same
4	Conv2D	3×3	512	Relu	Same
	Conv2D	3×3	512	Relu	Same
	Conv2D	3×3	512	Relu	Same
	MaxPooling2D				
5	Conv2D	3×3	512	Relu	Same
	Conv2D	3×3	512	Relu	Same
	Conv2D	3×3	512	Relu	Same

3.4. Fine-tuning and modified faster RCNN

To get a better USB detector, we propose improvements and modifications of Faster R-CNN. We propose resizing the image, adding an HPF filter, and adding a batch normalization layer. In detail, the above steps are:

- a. Minimal resizing of the sides of the image. According to Huang *et al.* reducing the size of the input image by half will speed up the inference time, although it will decrease the accuracy [13]. To increase the inference speed, in this study, we propose the minimum size of the image size is 300 pixels from the previous 600 pixels. That is important because, in its application, this system will be applied to moving images or video files.

- b. Image augmentation using Gaussian high pass filter (HPF). The accuracy of detection and classification in the object detection process usually depends on the feature expression of the detection object. The accuracy of detection and classification in the object detection process usually depends on the feature expression of the object [14]. The detection performance is also related to whether the features are diminished after a certain number of traditional convolutional operations [15]. Augmentation techniques can be used to increase the expression of the features of an object. The augmentation technique is commonly used to increase the performance of a object detector [16]. Some researchers have used this technique to increase the mAP value of the object detector [17]. In the proposed model, additional image augmentation is carried out in the form image sharpening process. Image sharpening refers to any enhancement technique that highlights an image's edges and fine details. Widely used image sharpening enhances local contrast and amplifies high-frequency pixels. This process highlights the unique features found in USB, namely the features that appear with high frequencies. As seen in Figures 1 and 2, EFB is dominated by pixels that form a line, while USB is dominated by pixels that form a circle. The image sharpening method used is the sharpening method with a Gaussian filter; this is because the high-frequency gaussian filter Gaussian high pass filter (GHPF) can sharpen images better than some other filters butterworth high pass filter (BHPF) [18]. The GHPF matrix used in this study is shown in Figure 6.

$$\begin{bmatrix} 0.11074074 & 0.11129583 & 0.11074074 \\ 0.11129583 & 0.1118537 & 0.11129583 \\ 0.11074074 & 0.11129583 & 0.11074074 \end{bmatrix}$$

Figure 6. Gaussian high pass filter (GHPF) kernel with size 3x3 and standard deviation 10

- c. Added a batch normalization layer in VGG16. Batch normalization [19] is another regularization technique that normalizes the set of activations in layers. Normalization works by subtracting the batch average from each activation and dividing by the batch standard deviation [20]. This normalization is a standard technique in pixel value pre-processing. The addition of batch normalization in several studies able to speed up the training process by eliminating internal covariate shift problems [21], is also able to increase the mAP value by more than 2% [22]. Table 2 shows the modified VGG architecture used as the backbone of USB detection with Faster RCNN. In this modified VGG16 architecture, there are 136,705,400 parameters with 136,696,952 trainable parameters and 8448 non-trainable parameters.

Table 2. Modified VGG16 with batch normalization layers added

Block	Layer	Kernel size	Dropout	Activation	Batch normalization	Padding
1	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	MaxPooling2D					
2	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	MaxPooling2D					
3	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	MaxPooling2D			Relu		Same
4	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	MaxPooling2D					
5	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same
	Conv_block	3×3	Yes	Relu	Yes	Same

With Conv_block as a layer wrapper of: x = dense(), x= batch_normalization()
x =drop_out(), x=activation()

3.5. Training performance parameters

The researchers used several parameters to assess the training performance of the deep learning-based object detector model. This study uses parameters such as loss, accuracy, and speed. A brief explanation of the parameter we:

- a. Region proposal network (RPN) layer loss: RPN layer loss is the sum of classification loss and bounding box regression loss. Classification loss uses cross-entropy loss to punish misclassified boxes. Regression loss uses the distance function between the true regression coefficients (calculated using the closest foreground anchor box to match the ground truth box) and the regression coefficients predicted by the network.
- b. Detection network losses: detection network losses also consist of two parts, regression loss, and classification loss. The regression and classification losses are also computed similarly to the RPN, except the regression coefficients are class-specific. The network calculates the regression coefficient for each object class. The cross-entropy loss is directly used for the classification loss, and the regression loss, the smooth L1 loss, is used, but the loss is only calculated for the positive sample.
- c. Total loss: the total loss is the sum of the losses in the RPN and detection network layers. As shown in (1) the equation for calculating the total loss.

$$Total_{loss} = RPN_{loss} + Detection_Network_{loss} \quad (1)$$

- d. Training speed: Speed is an indicator of object detection system [23]. The speed indicator is important, especially for hardware systems with limited resources. Unfortunately, this indicator is often opposite to other indicators such as accuracy and accuracy, so it is often necessary to balance the choice of performance indicators.

3.6. Testing (validation) performance parameters

The parameters used to assess validation performance are slightly different from those used to assess training performance. Here we use the mAP (mean precision average) parameter. This mAP parameter aims to assess the precision of the model for all classes. The following is an explanation of the parameters we use.

- a. Mean average precision (mAP): mean average precision (mAP) is a popular metric in measuring the accuracy of object detectors such as Faster R-CNN and SSD [24]. Mean Average Precision is used to measure the accuracy of object detectors (object position and object class) in all classes in a specific database [25]. As shown in (2) is the basic equation for calculating mAP [26].

$$mAP = \frac{\sum_{i=1}^K AP_i}{K} \quad (2)$$

where:

AP = Average Precision Class i

K = The number of classes evaluated

In (2), it can be seen that mAP is the average value of AP. Average precision (AP) is the most commonly used metric, derived from precision and recall, for evaluating object detection performance [27]. AP is evaluated on a specific object category. That is means that it is computed for each separate object category.

- b. Speed: inference speed is one of the performance parameters of the detection system. Of course, the ideal system is a system that is instantaneous but has high precision. To measure speed, the standard unit used is fps (frames per second) [28], [29]. Also, this is called the frame rate or frame frequency. If the speed is higher, it has better performance for handling more images. As shown in (3) and (4) are the equations used to calculate the detection time/image.

$$T = \left(\frac{test_time}{image} \right) = (total \frac{time}{amount\ of\ data\ in\ the\ test\ dataset}) \quad (3)$$

$$FPS = \frac{1}{T} \quad (4)$$

4. RESEARCH IMPLEMENTATION

The research was conducted in three steps. The first step is training the faster R-CNN USB detector system with the original VGG16 feature extractor. The second step is to train the USB detector system with a modified VGG16 and Gaussian filter addition during fine-tuning. The third step is testing and comparison between the two USB detector models. All these steps use the same dataset

The software used to compile is Anaconda3-2019 with Python 3.6.8, supported by the Python library for machine learning (Numpy, Panda, and Scikit), Keras version 2.2.4, and the artificial intelligence framework TensorFlow with version 1.14.0. The hardware uses an Intel core i-7 desktop, 8 GB RAM, and Nvidia RTX2080 graphic card. Figures 7 and 8 show the flow of the training and validation processes of the proposed USB detector system.

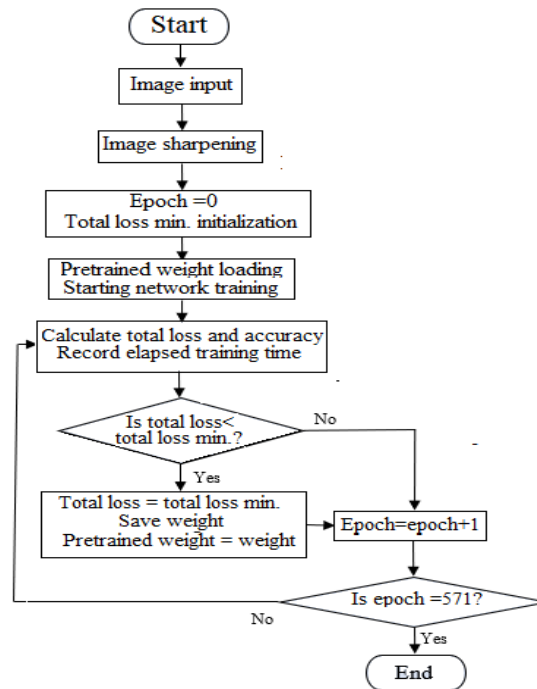


Figure 7. Workflow for USB detector training

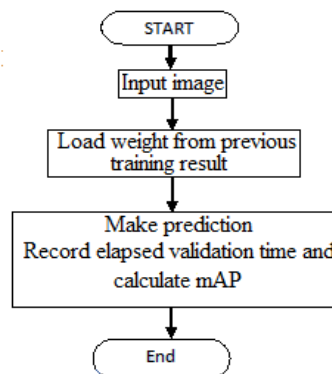


Figure 8. Workflow for USB detector validation

5. RESULTS AND DISCUSSION

5.1. The performance of USB detector using faster R-CNN based on original VGG16

USB detection performance based on the original VGG16 is divided into training and validation performance. Figures 9 to 18 show a performance graph of the USB detection system using the original VGG16. The training performance is shown in Figures 9 to 16, while the validation performance is shown in Figures 17 and 18.

5.1.1. Original VGG16 based faster R-CNN USB detector training performance

The training performance parameters for the RPN network layer are shown in Figures 9 to 16. Based on Figures 9 to 16, it can be seen that the losses during the training process have successfully converged. Figure 9 shows the initial RPN classifier loss is 2; this loss consistently decreases and reaches a value of 0.01 after 500 epochs and is stable. The same results are also shown for RPN regression loss; the initial value of RPN regression loss is 0.11, then consistently decreases to 0.02 after 500 epochs as shown in Figure 10.

The losses of the detection network layer also show the same pattern, where the initial regression loss of the detection network layer is 0.35; this loss consistently continues to decrease and is stable at 0.05

after 500 epochs as shown in Figure 11. The classification loss also converges; as shown in Figure 12, the initial classification loss is 0.8. This value continued to decrease and stabilize at a value of 0.14 after 500 epochs. It causes the total loss during training to be also convergent. The initial total loss value is 3.3 and converges to a total loss of 0.203 as shown in Figure 13.

The time required for 570 epochs (460 images per epoch) was 2349.1 minutes or 39.15 hours as shown in Figure 14; this means the average time required for one epoch is 4.12 minutes or 0.54 seconds per image (1.85 fps). Figure 15 shows the accuracy performance during training, and it can be seen in Figure 15 that the class detection accuracy is 93%. The average bounding box with a score of 24 was obtained during training, as shown in Figure 16.

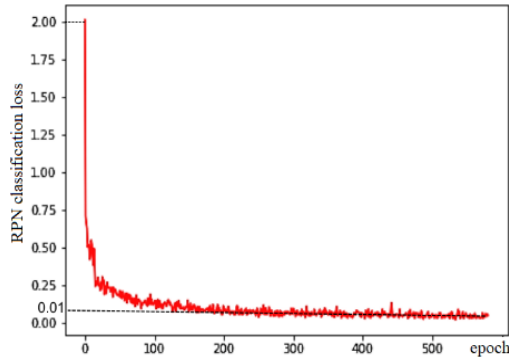


Figure 9. RPN classification loss of original Faster R-CNN during training

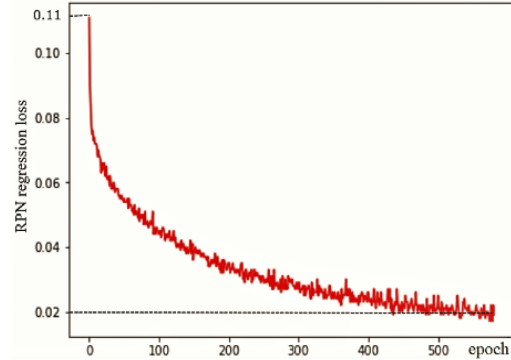


Figure 10. RPN regression loss of original Faster R-CNN during training

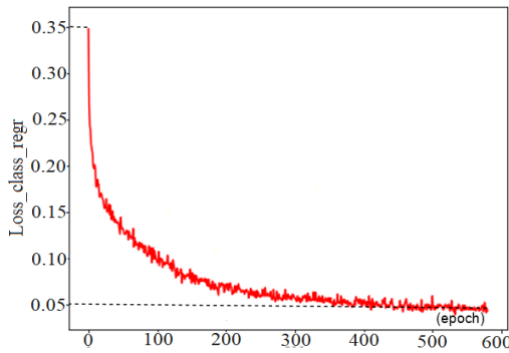


Figure 11. Regression loss of the original faster R-CNN detection network layer

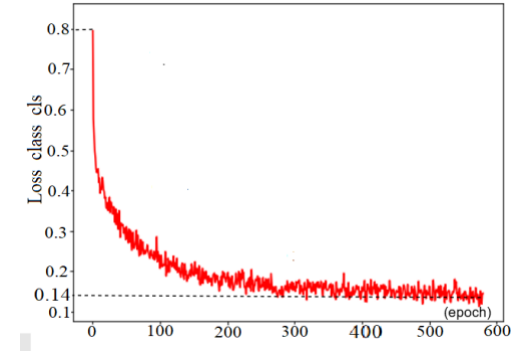


Figure 12. Classification loss of the original faster R-CNN detection network layer

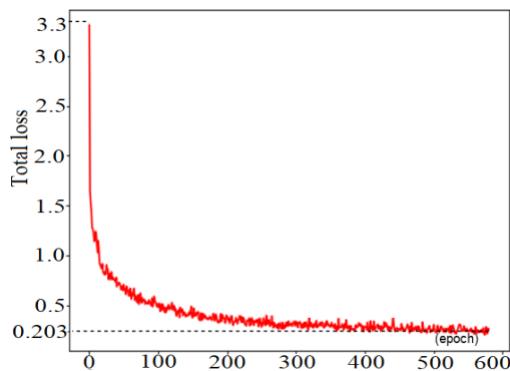


Figure 13. Total loss loss of original faster R-CNN

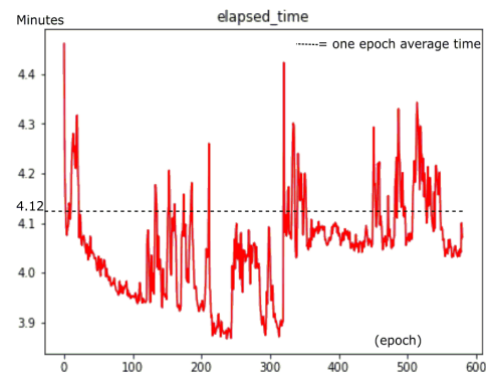


Figure 14. Time elapsed/epoch of original faster R-CNN

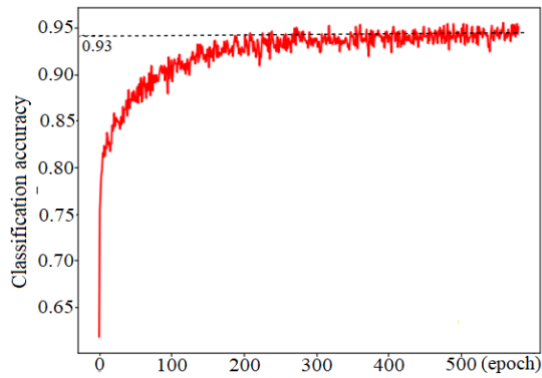


Figure 15. Classification accuracy of original faster R-CNN

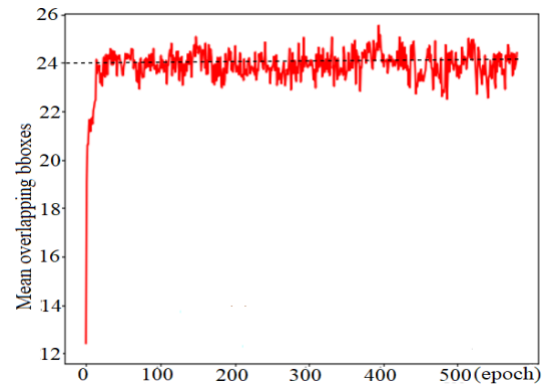


Figure 16. Mean overlapping BBox of original faster R-CNN

5.1.2. Original VGG16 based Faster R-CNN USB detector validation performance

Validation was carried out using test data which contained 40 images. In this test, the mAP average data obtained was 0.728 as shown Figure 17. The time needed to detect the testing dataset is 21.61 seconds as shown in Figure 18, which means that the average detection time for one image is 0.54 seconds (1.85 fps)

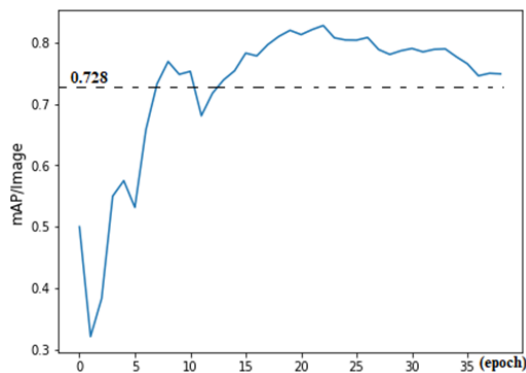


Figure 17. mAP/Image during testing

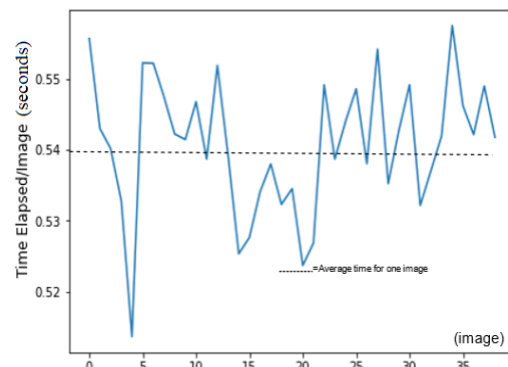


Figure 18. Time elapsed/image during testing

5.2. USB detector performance using modified VGG16 based faster R-CNN

As with the performance of the original VGG16-based UBS detector, the modified VGG16-based USB detector is also divided into training and validation performance. Figures 19 through 28 show a performance graph of the USB detection system using modified VGG16, image filtering, and image resizing. The training performance is shown in Figures 19 to 26, while the validation performance is shown in Figures 27 and 28.

5.2.1. Modified VGG16 based faster R-CNN USB detector training performance

The USB detector system training was carried out using the same dataset as the previous training. The tools and datasets used are identical to the tools and datasets used in the previous training. The hyperparameters used are also the same except for the input image size. Figures 19 to 26 show the training performance (regression and classification loss, respectively) of the USB detector based on the modified VGG16 Faster R-CNN.

RPN network layer training performances are depicted in Figures 19 to 20. Figure 19 shows that the initial value of the RPN classification loss is 1.695. This loss decreases consistently, reaches a value of 0.014 after 500 epochs, and is stable. The same results are also shown for RPN regression losses. The initial value of RPN losses is 0.092, then consistently decreases to 0.005 after 500 epochs as shown in Figure 20.

The losses of the classifier layer also show the same pattern, where the initial value of the regression loss of the classifier layer is 0.424. This loss consistently decreases and is stable at 0.03 after 500 epochs as

shown in Figure 21. The classification loss during training is convergent as shown in Figure 22, the initial classification loss of the detection network layer is 0.883, and this value continues to decrease and stabilize at a value of 0.104 after 500 epochs. Its causes the overall loss during training for (RPN layer and detection network layer losses) is also convergent. The initial total loss is 3.1 and converges to 0.103 as shown in Figure 23. The total time required for 570 epochs is 1509.52 minutes or 25.15 hours as shown in Figure 24; this means that the average time needed for one epoch is 2.64 minutes or 0.34 seconds per image (2.94 fps). It can be seen in Figure 25 that the best class detection accuracy is 98%, for the average overlapping bounding box is 17.8, as depicted seen in Figure 26.

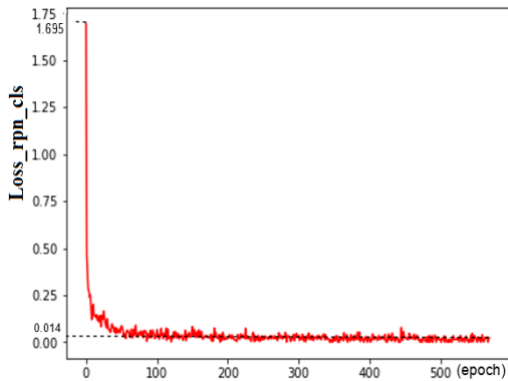


Figure 19. RPN layer classification loss

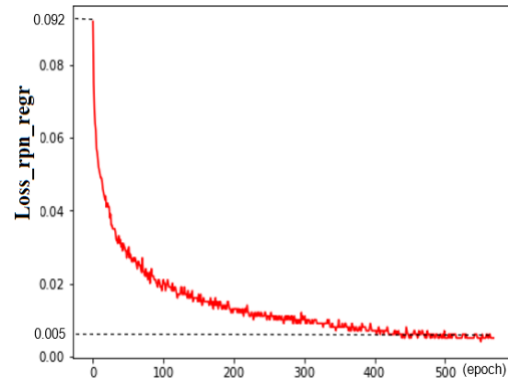


Figure 20. RPN layer regression loss

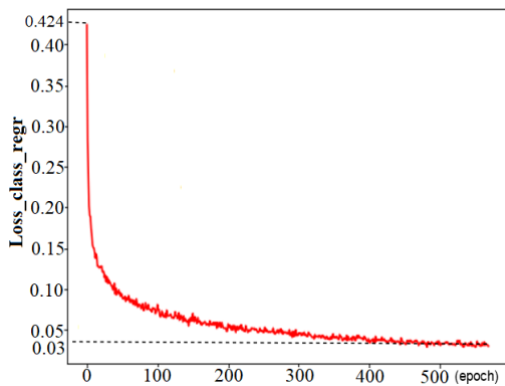


Figure 21. Regression losses of the detection network layer

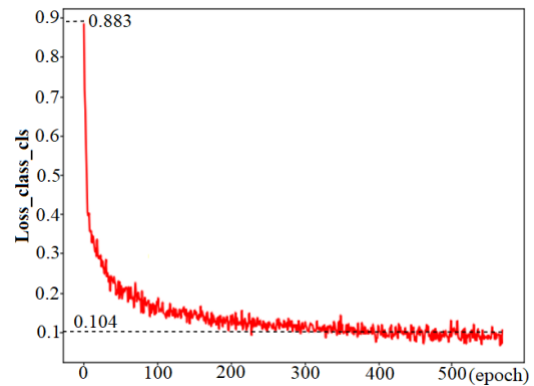


Figure 22. Classification losses of the detection network layer

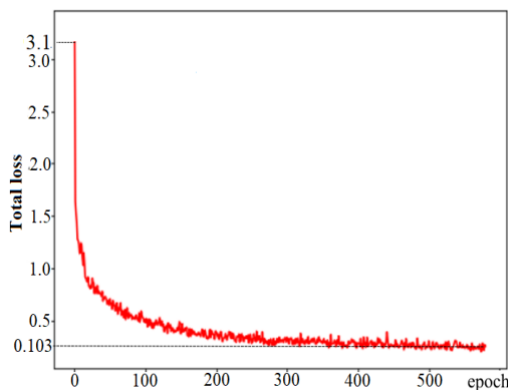


Figure 23. Total loss loss of modified faster RCNN

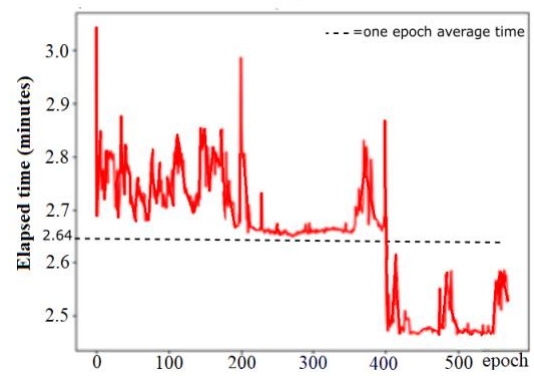


Figure 24. Time elapsed/epoch of the modified faster R-CNN

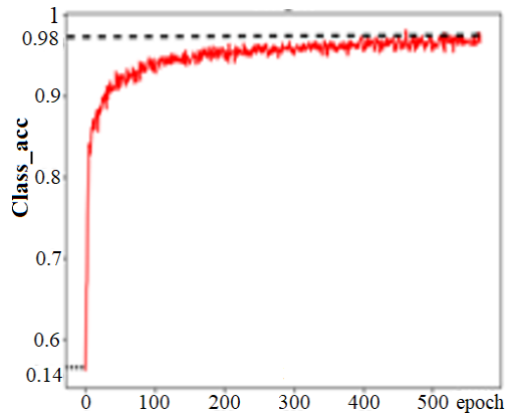


Figure 25. Classification accuracy of the modified faster R-CNN

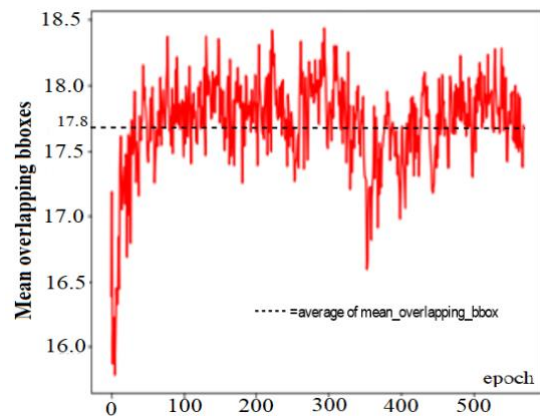


Figure 26. Mean overlapping bboxes

5.2.2. Modified VGG16 based Faster R-CNN USB detector validation performance

Tests to obtain training performance are carried out using the same dataset for the previous validation test. Figures 27 and 28 show the performance of the USB detector system validation test after being modified. In this test, the average mAP data obtained is 0.782 Figure 27 with an overall detection time of 10.92 seconds Figure 28, which means the average detection time for one image is 0.27 seconds (3.7 fps). The values above are improvements from the values obtained previously.

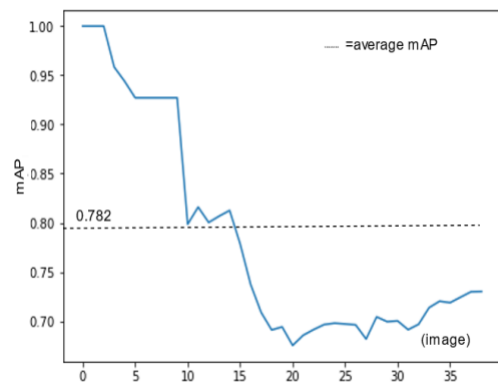


Figure 27. mAP/Image of the modified faster R-CNN

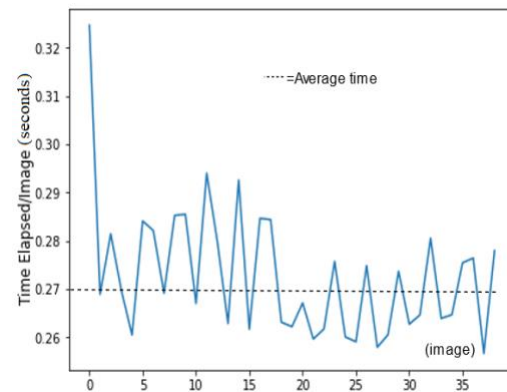


Figure 28. Time elapsed/image of the modified faster R-CNN

5.3. Performance comparison

The training performance and validation performance of the original and modified VGG16-based Faster R-CNN USB detector are shown in Tables 2 and 3. Based on the training performance table as shown Table 2, it can be seen that the proposed system has improved performance, namely 58.9% faster, 43.4% smaller total losses with 5.4% better class prediction accuracy. However, the bbox (bounding box) overlap is 23.91% less.

Table 2. Comparison of training performance between USB detection systems using the original faster R-CNN and with the proposed system

Performance parameters	USB detector based original system	USB detector based proposed system	Remark
Speed (FPS)	1.85	2.94	58.9% faster
Total loss	0.203	0.103	43.4% smaller
Class accuracy	93%	98%	5.4% more accurate
BBox Overlap	23	17,5	23.91% less

While the test performance as shown Table 3, the proposed system also shows a significant improvement where the proposed system has a speed of 3.7 fps which is faster than the previous system, which has a speed of 1.85 fps. The difference of 1.85 fps means that the proposed system is 100% faster. The proposed system has better performance for accuracy, with an mAP value of 0.782; this value is higher than the previous system, which has an mAP value of 0.728. This difference of 0.054 means an increase in accuracy performance of 7.42%.

Table 3. Comparison of validation performance between USB detection systems using the original faster R-CNN and with the proposed system

Performance parameters	USB detection system using original Faster R-CNN VGG16	USB detection system using Faster R-CNN based on modified VGG16	Comparison of the proposed system with the original system
Speed (fps)	1.85	3.7	100% faster
mAP	0.728	0.782	7.42% more precise

6. CONCLUSION

From the results, it can be concluded that the augmentation method, minimal image resizing, and adding a batch normalization layer can improve the performance of the faster R-CNN-based USB detection system using the modified VGG16 as a feature extractor. The proposed system on the speed parameter gives better results in the training process, which is 58.9% faster; the total loss is lesser (43.4% less) and has better best accuracy (98%) but had a lower average bounding box overlap (BBox) 23.91% less. While the testing (validation) performance, the proposed system can work faster (100% faster) and with better precision (7.42% more precise).

ACKNOWLEDGEMENTS

The author would like to thank Nuryono Satya Widodo. ST, M.Eng, Head of the Department of Electrical Engineering, Ahmad Dahlan University, Yogyakarta, facilitated this research. We also thank the management of PT. Sawit Arum Madani allowed us to retrieve data.




REFERENCES

- [1] U. K. H. M. Nadzim, R. Yunus, R. Omar, and B. Y. Lim, "Factors contributing to oil losses in crude palm oil production process in malaysia: a review," *International Journal of Biomass and Renewables*, vol. 9, no. 1, 2020.
- [2] O. Walat and N. S. Bock, "Palm oil mill OER and total oil losses," *POEB*, no. 108, pp. 11–16, 2013.
- [3] A. Hassan, N. H. Muhammad, Z. A. Rahman, R. M. Halim, H. Alias, and M. Sabtu, "Improving mill oil extraction rate under the malaysian national key economic area," *POEB*, no. 103, pp. 33–47, 2012.
- [4] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 1026–1034, doi: 10.1109/ICCV.2015.123.
- [5] M. Ghaderzadeh and F. Asadi, "Deep learning in the detection and diagnosis of COVID-19 using radiology modalities: a systematic review," *Journal of Healthcare Engineering*, vol. 2021, pp. 1–10, Mar. 2021, doi: 10.1155/2021/6677314.
- [6] J. G. A. Barbedo, L. V. Koenigkan, T. T. Santos, and P. M. Santos, "A study on the detection of cattle in uav images using deep learning," *Sensors*, vol. 19, no. 24, p. 5436, Dec. 2019, doi: 10.3390/s19245436.
- [7] K. R. Aravind, P. Raja, R. Ashiwin, and K. V. Mukesh, "Disease classification in Solanum melongena using deep learning," *Spanish Journal of Agricultural Research*, vol. 17, no. 3, Nov. 2019, doi: 10.5424/sjar/2019173-14762.
- [8] X. Liu, K. H. Ghazali, F. Han, and I. I. Mohamed, "Automatic detection of oil palm tree from UAV images based on the deep learning method," *Applied Artificial Intelligence*, vol. 35, no. 1, pp. 13–24, Jan. 2021, doi: 10.1080/08839514.2020.1831226.
- [9] M. License, "Tzutalin Labellmg," <https://github.com/tzutalin/labellmg> (accessed Dec. 20, 2020).
- [10] L. Du, R. Zhang, and X. Wang, "Overview of two-stage object detection algorithms," *Journal of Physics: Conference Series*, vol. 1544, no. 1, May 2020, doi: 10.1088/1742-6596/1544/1/012033.
- [11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.
- [12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv:1409.1556*, Apr. 2014.
- [13] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3296–3297, Apr. 2017.
- [14] Y. Xiao, X. Wang, P. Zhang, F. Meng, and F. Shao, "Object detection based on faster R-CNN algorithm with skip pooling and fusion of contextual information," *Sensors*, vol. 20, no. 19, p. 5490, Sep. 2020, doi: 10.3390/s20195490.
- [15] H. K. Leung, X.-Z. Chen, C.-W. Yu, H.-Y. Liang, J.-Y. Wu, and Y.-L. Chen, "A deep-learning-based vehicle detection approach for insufficient and nighttime illumination conditions," *Applied Sciences*, vol. 9, no. 22, p. 4769, Nov. 2019, doi: 10.3390/app9224769.
- [16] B. Zoph, E. D. Cubuk, G. Ghiasi, T.-Y. Lin, J. Shlens, and Q. V. Le, "Learning Data Augmentation Strategies for Object Detection," 2020, pp. 566–583.
- [17] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learnin," *Journal of Big Data*, vol. 6, no. 1,




- p. 60, Dec. 2019, doi: 10.1186/s40537-019-0197-0.
- [18] A. Dogra and P. Bhalla, "Image sharpening by gaussian and butterworth high pass filter," *Biomedical and Pharmacology Journal*, vol. 7, no. 2, pp. 707–713, Dec. 2014, doi: 10.13005/bpj/545.
- [19] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*, 2015, vol. 37, pp. 448–456.
- [20] S. Ioffe and C. Szegedy, "renormalization: towards reducing minibatch dependence in batch-normalized models," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 1942–1950.
- [21] Y. Zhou, C. Yuan, F. Zeng, J. Qian, and C. Wu, "An object detection algorithm for deep learning based on batch normalization," 2018, pp. 438–448.
- [22] L. Jiao *et al.*, "A Survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128837–128868, 2019, doi: 10.1109/ACCESS.2019.2939201.
- [23] W. Li, "Analysis of object detection performance based on faster R-CNN," *Journal of Physics: Conference Series*, vol. 1827, no. 1, Mar. 2021, doi: 10.1088/1742-6596/1827/1/012085.
- [24] H. Mao, X. Yang, and W. J. Dally, "A delay metric for video object detection: what average precision fails to tell," *arXiv:1908.06368*, Nov. 2019.
- [25] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electronics*, vol. 10, no. 3, p. 279, Jan. 2021, doi: 10.3390/electronics10030279.
- [26] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, Jul. 2020, pp. 237–242, doi: 10.1109/IWSSIP48289.2020.9145130.
- [27] K. Oksuz, B. C. Cam, S. Kalkan, and E. Akbas, "One metric to measure them all: localisation recall precision (LRP) for evaluating visual detection tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1, 2021, doi: 10.1109/TPAMI.2021.3130188.
- [28] L. Liu *et al.*, "Deep Learning for generic object detection: a survey," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 261–318, Feb. 2020, doi: 10.1007/s11263-019-01247-4.
- [29] L. Lang, K. Xu, Q. Zhang, and D. Wang, "Fast and accurate object detection in remote sensing images based on lightweight deep neural network," *Sensors*, vol. 21, no. 16, p. 5460, Aug. 2021, doi: 10.3390/s21165460.

BIOGRAPHIES OF AUTHORS






Wahyu Sapto Aji    is a Ph.D. student in the Faculty of Electrical and Electronic Engineering Technology, University of Malaysia Pahang, Malaysia. Wahyu S Aji obtained a bachelor's and master's degree in Electrical Engineering from the Electrical Engineering Department of Gadjah Mada University, Yogyakarta, Indonesia. He has been working as a lecturer in the electrical engineering department at Ahmad Dahlan University since 2014. His current research interests include Image Processing, Deep Learning. He can be contacted at email: wahyusa@ee.uad.ac.id.



Kamarul Hawari bin Ghazali    is a Professor in the Faculty of Electrical and Electronic Engineering Technology, Universiti Malaysia Pahang. His major research areas include Machine Vision System, Image Processing, Signal Processing, Intelligent system, Vision Control, Thermal Imaging Analysis. (in all related applications – Electrical, Medical, Environment), Deep Learning for Image and Signal Classification. Major sponsor research includes Fundamental Research Grant Scheme (FRGS), International Grant, Long Term Research Grant Schemes (LRGS) and short-term grant from UMP. Currently a Professional Engineer (Ir), Board of Engineer Malaysia (BEM). He can be contacted at email: kamarul@ump.edu.my



Son Ali Akbar    is a Ph.D. student in the Faculty of Electrical and Electronics Engineering Technology, University Malaysia of Pahang, Malaysia. He has been working as a lecturer in the electrical engineering department at Ahmad Dahlan University since 2014. His current research interests include Image Processing, Deep Learning. He can be contacted at email: son.akbar@te.uad.ac.id.