INTERNATIONAL JOURNAL OF INTEGRATED ENGINEERING
ISSN: 2229-838X e-ISSN: 2600-7916

IJIE

Vol. 16 No. 1 (2024) 262-271
https://publisher.uthm.edu.my/ojs/index.php/ijie

Detection and Measurement System for Button Mushrooms Using Convolutional Neural Network

Lio Wei Yong, Radzi Ambar[1]*, Mohd Helmy Abd Wahab[1], Muhammad Mahadi Abd Jamil[1], Chew Chang Choon[1]

[1] *Department of Electronic Engineering, Faculty of Electrical and Electronic Engineering Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johor, MALAYSIA*

*Corresponding Author: aradzi@uthm.edu.my

DOI: https://doi.org/10.30880/ijie.2024.16.01.021

Article Info

Received: 13 December 2023
Accepted: 28 February 2024
Available online: 22 May 2024

Keywords

Button mushroom, detection and measurement system, convolutional neural network, YOLOv4

Abstract

In Malaysia, the button mushroom is recognized as a vegetable with high nutritional value and is easy to cultivate. Monitoring mushroom growth requires farmers to regularly inspect their crops, which is time-consuming and inefficient. Hence, an automated detection and measurement system for button mushrooms has been developed using image processing techniques based on convolutional neural network (CNN) algorithm model known as YOLOv4. The algorithm was utilized to train the system using button mushroom images to create training models. The performance of the YOLOv4 models was evaluated across various iterations ranging from 1000 to 6000 iterations. The model with 2000 iterations demonstrated the most effective performance based on Recall, Precision, F1-score, Time and Mean Average Precision metrics. The model was used in a small-scale experimental setup to evaluate the button mushroom detection and measurement system's performance. Based on the results obtained from the experiments, the detection and measurement system demonstrated high accuracy in locating the position of each button mushroom with only a 5% deviation error in predicting the size of each button mushroom.

1. Introduction

Since the 19th century, the global issue of overpopulation has intensified, leading to various challenges such as climate change and severe food shortages. The increasing population has eliminated large amounts of anthropogenic greenhouse gases, contributing to the extreme weather [1]. The greenhouse effect will affect agriculture, leading to a decrease in the production of essential foods such as cereals, vegetables and fruits. This effect will exacerbate the problem of food shortages, making it more critical. Among vegetables, mushrooms are recognized for their high nutritional value, and white button mushrooms are known as the fastest-growing mushrooms, especially in Malaysia. Mushrooms are highly beneficial for human health due to their low-fat content, low-fiber content and cholesterol-free [2]. However, climate change also affects mushroom production as it is highly dependent on specific temperature and humidity conditions. Since mushrooms are sensitive to the environment, most of the mushrooms are grown traditionally in greenhouses.

A greenhouse is an enclosed space equipped with various equipment and sensors which create a fully functional system capable of controlling its interior conditions and environment. For a mushroom greenhouse, monitoring and controlling the humidity and temperature are essential for maintaining the optimal growth conditions of mushrooms [3]. Traditionally, button mushroom farmers had to manually check their crops' growth in greenhouses, a labor-intensive and time-consuming process. Thankfully, advancements in modern technology offer the potential for automation in monitoring and maintaining mushrooms' growth and harvesting. However, to achieve this, necessary technical infrastructure is needed, such as automated growth measurement systems and robotic harvesting mechanisms.

Artificial Intelligence (AI) has been the most popular technology in recent years due to its ability to accomplish impressive tasks and jobs. AI has been utilized in various fields such as medicine, industries and agriculture. One of the most popular areas of AI is computer vision, as it enables automated processes for various tasks such as image recognition, image analysis and speech recognition demonstrating outstanding performance. Deep learning, a method in AI, has been extensively developed in the field of computer vision. In deep learning, an algorithm called Convolutional Neural Networks (CNN/ConvNets) can differentiate between various objects in an image automatically [4].

The origins of CNN algorithms can be traced back to 1990 with the introduction of LeNet-5, the first multi-layer artificial neural network specifically designed for handwritten digit recognition [5]. This breakthrough paved the way for a new era of computer vision. Since then, researchers have developed numerous fast and accurate CNN architectures, including Fast R-CNN for object detection and segmentation in 2013 [6]; You-Only-Look-Once (YOLO), known for its real-time processing speed in 2013 [7]; GoogLeNet and AlexNet in 2014, which incorporated innovative inception module [8]; ResNet and Faster R-CNN in 2015, known for addressing vanishing gradient problems [9, 10]; Mask R-CNN and YOLOv2 in 2017, enabling instance segmentation with bounding boxes and masks [11, 12]; YOLOv3 in 2018, offering good balance between speed and accuracy [13, 14]; YOLOv4 in 2020, further refining the model for various applications [15]; YOLOv5 in 2020, that gains popularity for its open-source nature and ease of use [16]; and more recently, YOLOv6, YOLOv7 released in 2022 [17, 18]; and YOLOv8 in 2023 [19], showcasing the continuous evolution of the YOLO family. These advancements have continuously improved efficiency and recognition rates in image recognition tasks. Among these CNN architectures, the YOLO family has seen significant growth due to several factors. Firstly, it focuses speed, making it suitable for real-time applications in various fields such as autonomous vehicles and robotics. Secondly, it maintains a focus on accuracy that ensures reliable object detection performance. Thirdly, the open-source nature of several YOLO versions, particularly YOLOv4 and later, fosters a collaborative development environment where researchers and developers can contribute and customize models for specific needs. Within the YOLO family, YOLOv4 is considered one of the most stable versions due to its wider adoption and testing history.

CNN has been successfully applied in agriculture for various computer vision applications. Sladojevic et al. introduced a plant disease recognition model using CNN which can recognize 13 types of plant diseases based on the analysis of healthy leaves condition [20]. CNN was performed using Caffe framework and produced 96.3% on the final overall accuracy of the trained model. Another research by Sa et al. utilized Faster R-CNN, a fruit detection system on seven fruits: rock melon, strawberry, apple, avocado, mango, orange, and sweet pepper [21]. The system produced detection performance up to a 0.83 F1-score based on a field farm dataset with fast detection rate. Then, Bargoti et al. proposed a deep fruit detecting system using also Faster R-CNN framework to detect, identify, and differentiate different orchard types, apples, mangoes and almonds [22]. The study achieved a detection performance of more than 0.9 F1-score for apples and mangoes. Tu et al. proposed a passion fruits detection and identifying system utilizing Faster R-CNN using RGB-D images with a SVM classifier for the identification of fruit maturity in various phases [23]. The proposed method achieved an accuracy of 92.7% for detection and 91.5% for maturity classification. Wang et al. proposed a remote apple growth monitoring system in an orchard called fused convolution feature network based on ResNet-50 and a fusion of convolutional features [24]. The system achieved in measuring the mean average absolute error of 0.90 mm for an apples' horizontal diameter. Liu et al. developed a tomato detection system using YOLOv3 [15]. The system is based on two approaches, a dense architecture for feature extraction, and replacing the traditional R-Bbox with a proposed C-Bbox for greater precision of tomato shape identification. Experimental results under slight occlusion conditions showed 94.58% correct identification rate. Based on the above, much research has implemented two-stage object detector or feed forward network such as Fast R-CNN that utilizes regions to localize the object within an image. The detector identifies high probabilities portions of the image which contains the object. However, YOLO models are faster by directly forecasting the bounding boxes and their corresponding classes using a single-stage object detector. Furthermore, there are very few studies on mushroom detection and measurement using CNN.

This paper presents the development of a detection and measurement system for button mushrooms using CNN algorithm called YOLOv4. YOLOv4 is chosen due to its superior performance and accuracy compared to its predecessor, YOLOv3 and Faster R-CNN, as evidenced by previous research, and being one of the most stable versions in the YOLO family [14]. Hence, the CNN algorithm that is used to construct the proposed button mushroom detection and measurement system in this work is YOLOv4. The proposed system aims to automate the monitoring and measurement of button mushrooms that can reduce the needs for manual checks and optimizing resource utilization. The remainder of this paper is organized as follows. Section 2 presents an in-depth overview of the detection method. Section 3 describes the experimental results based on the proposed method, and conclusions of this paper are reported in section 4.

2. Methodology

The proposed mushroom detection and measurement system was developed using the object detection CNN algorithm, specifically YOLOv4. The YOLOv4 algorithm was utilized to train the system involving feeding the algorithm a large dataset of button mushroom images. Each image within the dataset is manually labeled with bounding boxes surrounding the mushrooms using dedicated software. These bounding boxes indicate the location and size of each button mushroom in the image. During training, the YOLOv4 model learns to identify the distinctive visual features of button mushrooms within the images. This allows the model to differentiate button mushrooms from other objects in the image and accurately predict their location and size.

Following the training process, the performance of multiple trained YOLOv4 models is evaluated. This evaluation involves assessing metrics such as Recall, Precision, F1-score, Time and Mean Average Precision (mAP) in detecting button mushrooms. The model demonstrating the most optimal performance based on these evaluation metrics will be chosen as the final model for the button mushroom detection and measurement system. Fig. 1 illustrates the flowchart for developing the button mushroom detection and measurement system's model. A detailed explanation of each step within the flowchart is provided in the subsequent subsections.

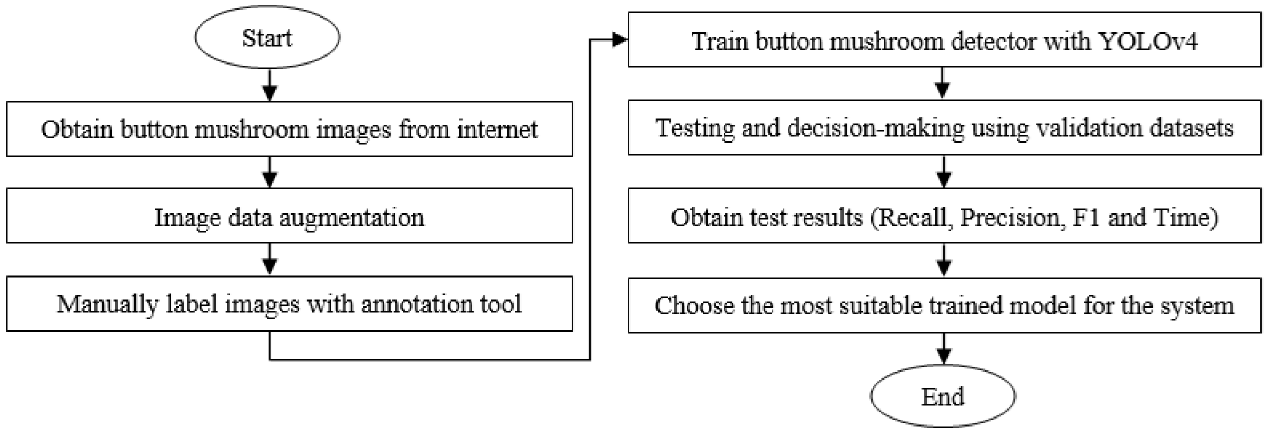

Start
Obtain button mushroom images from internet
Image data augmentation
Manually label images with annotation tool
Train button mushroom detector with YOLOv4
Testing and decision-making using validation datasets
Obtain test results (Recall, Precision, F1 and Time)
Choose the most suitable trained model for the system
End

Fig. 1 *Flowchart to train button mushroom detection and measurement system*

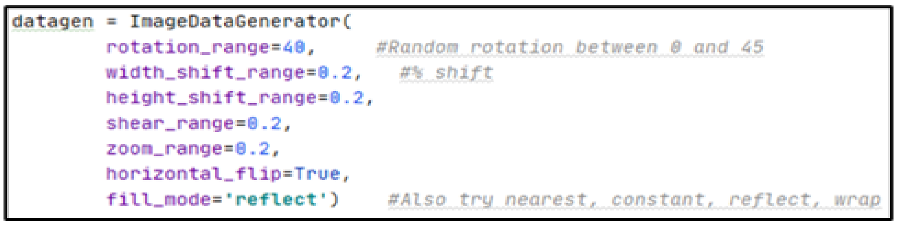
2.1 Obtaining Images and Data Augmentation

Based on Fig. 1, first, dataset images for object detection models were obtained from internet sources such as Google and Bing. In this work, a total of 27 original images obtained from the internet are used to generate 540 new images through the data argumentation technique. Data augmentation techniques are often used to generate more images or data to enhance computer vision systems' performance. Data augmentation has several techniques, but only the geometric transformations technique is used in this project. This technique involves randomly flipping, cropping and rotating the original image to create new images. The parameters used for data augmentation in this work are listed in Table 1.

Table 1 *Parameter of data augmentation*

Parameter	Status
Rotation range	40 °
Width shift	20 %
Height shift	20 %
Shear range	20 %
Zoom range	20 %
Horizontal flip	True
Fill mode	Reflect

As shown in Table 1, in data augmentation, the rotation range randomly rotates the image, with the maximum rotation not exceeding 40°. The width and height shift randomly move the image, not exceeding 20% from the original dimensions. Shear and zoom enable cutting and either zooming in or out of the image, also limited to a maximum of 20%. Horizontal flip is set to true, allowing the image to undergo mirror reflection. Fill mode is used to address the blank spaces that occur after rotation, shift, shear or zoom. For example, if the image is shifted 20% to the left, it may create a blank space without any pixels. By applying the fill mode with reflection function, a reflection of the image will fill into the blank space. Fig. 2 shows a section of the Python script for data augmentation, where the parameter values can be manually set.



```
datagen = ImageDataGenerator(
        rotation_range=40,      #Random rotation between 0 and 45
        width_shift_range=0.2,    #% shift
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='reflect')    #Also try nearest, constant, reflect, wrap
```

Fig. 2 *Flowchart to train button mushroom detection and measurement system*

2.2 Labelling and Annotation of Images with Annotation Tool

Labelling and annotation of images are important steps before converting them into datasets for training purposes. To create labels and annotations for button mushroom images, LabelImg was used. It is a graphical image annotation tool written in Python, and it supports multiple formats including YOLO format. In LabelImg, the input images are shown in the interface, and users can label the object in that image with its class names. Fig. 3 shows a screen capture of LabelImg with an image containing button mushrooms. As seen in the figure, all button mushrooms are labelled with green bounding boxes. The labelled mushrooms are named as class ‘Mushroom’. These labelling processes are done manually. After labelling, the annotations of each image are saved to a folder as .txt files in YOLO format. A file named ‘classes.txt’ will also be saved into that folder, which defines the class name ‘Mushroom’ that YOLO label refers to. A total of 540 images are labelled and formed as training datasets.

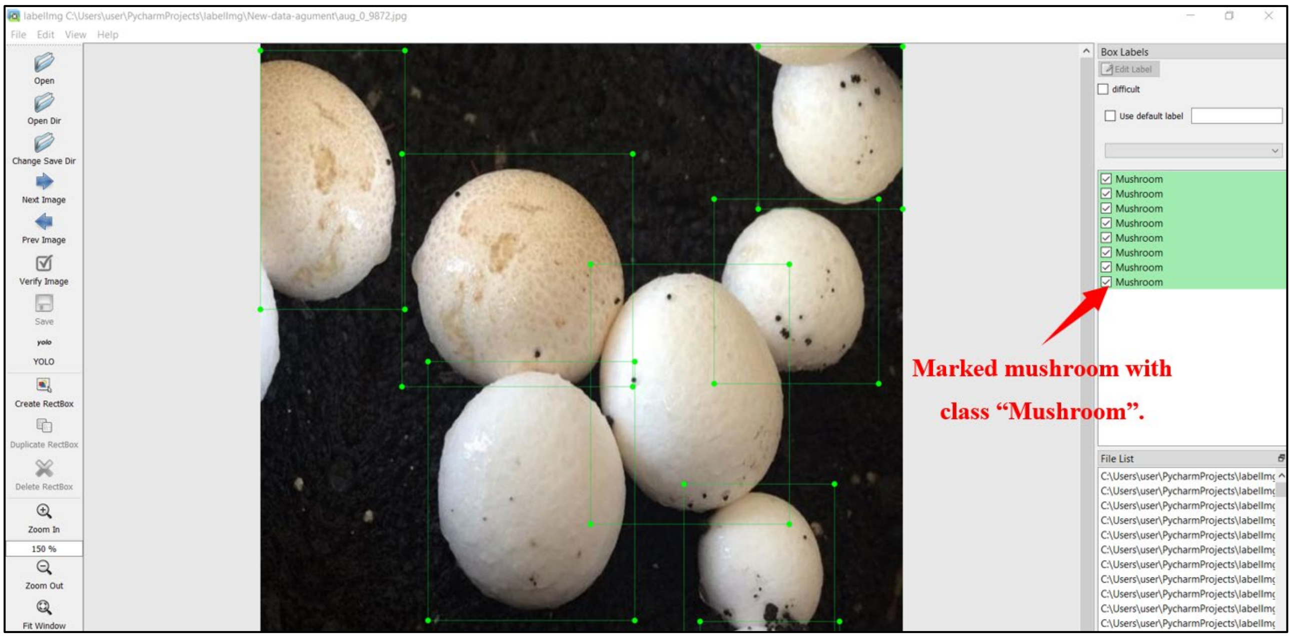


Fig. 3 *After data augmentation, LabelImg is used on button mushroom images for labelling the target objects.*

2.3 Button Mushroom Detector Based on YOLOv4

Fig. 4 shows the architecture of the button mushroom detector based on YOLOv4. As shown in the figure, the YOLOv4 detector comprises of three (3) parts: backbone, neck, and heads [14].

After augmenting and labeling the images, they are fed into the YOLOv4 backbone known as CSPDarknet53. CSPDarknet53 functions as the backbone network of YOLOv4, extracting image features for training. The YOLOv4 neck consists of two layers: Spatial Pyramid Pooling (SSP) and Path Aggregation Network (PANet). These layers gather feature maps from various levels and are comprised of several bottom-up paths and top-down paths. The SPP network enhances the receptive field and help separates contextual features, while the PANet shortens the path connecting low-level and high-level information via bottom-up and top-down paths, then, concatenating or fusing the information at different levels before being fed to the head. YOLOv3 head is implemented as the head for YOLOv4 which is used to predict the classes and bounding boxes of the objects at three different scales. To develop the button mushroom detector based on YOLOv4, pre-trained weights for the convolutional layers of the YOLOv4 network were used during training. These pre-trained weights contribute to a more accurate system and speed up the training process. The YOLOv4 network is configured with the parameters shown in Table 2.

The dataset used in this study consists of two parts: the training dataset and validation dataset. The training dataset is used for training purposes, while the validation dataset serves to thoroughly test the button mushroom detection and measurement system after training. The validation dataset size typically ranges from 20% to 30% of the training dataset size. In this case, the training dataset contains 420 button mushroom images, and the validation dataset comprises 120 button mushroom images.

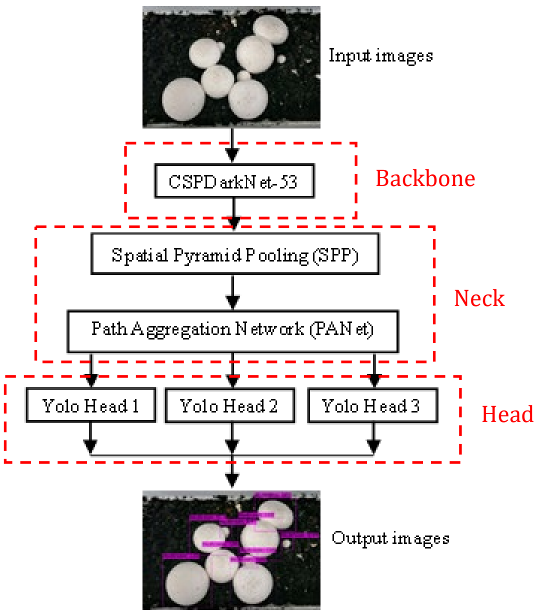


Fig. 4 *Architecture of the mushroom detector based on YOLOv4*

Table 2 *Parameter of data augmentation*

Parameter	Status
Input size	416 × 416
Batch	64
Learning rate	0.001
Classes	1
Iterations	6000

2.4 Size Measuring Methods

The proposed button mushroom detection and measurement system will begin by capturing an image before proceeding with the detection and measurement algorithm. The original image will be saved into a designated file. During the process, each mushroom in the image will be detected and localized. Every mushroom detected by the YOLOv4 detector will be labeled in the original image and saved into the same designated file. After obtaining the necessary details, the system will estimate the size of each detected mushroom using the size measuring method.

The size measuring method starts by calculating the unit area of a button mushroom using the size of its bounding box. The bounding box's width, W and height, H are multiplied to form the unit area for a button mushroom size S (pixels), as shown in Equation 1.

$$S = W \times H \tag{1}$$

Before calculating the actual size of the mushrooms, the scale between the bounding box area and the actual button mushroom area size needs to be determined. The scale ratio, R (cm2/pixel) and the actual size of the mushrooms, S̄ (cm2) are denoted as in Equation 2.

$$\bar{S} = S \times R \tag{2}$$

After predicting the size of each mushroom, the data will be saved into an excel file where the tile is named automatically using a specified format (Year-month-day (Hour' Minute' Second)), for the user to easily manage the file. Fig. 5 shows the flowchart of size measuring method.

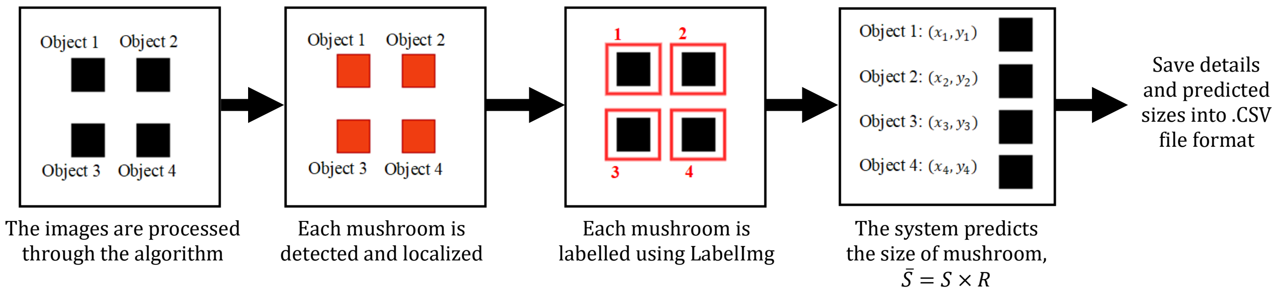

Object 1
Object 2
Object 3
Object 4
The images are processed through the algorithm
Each mushroom is detected and localized
1
2
3
4
Each mushroom is labelled using LabelImg
Object 1: (x_1, y_1)
Object 2: (x_2, y_2)
Object 3: (x_3, y_3)
Object 4: (x_4, y_4)
The system predicts the size of mushroom, $\bar{S} = S \times R$
Save details and predicted sizes into .CSV file format

Fig. 5 *Flowchart of size measuring method*

The estimated size (measured size) for each button mushroom will be compared with the actual size to obtain the percentage error. The percentage error is calculated using the equation in Equation 3.

$$Percentage\ Error = \frac{|Estimated\ size - Actual\ size|}{Actual\ size} \tag{3}$$

3. Methodology

3.1 Evaluation of the YOLOv4 Model Performance with Different Iteration

The evaluation of the performance for YOLOv4 model's performance at different iteration has been carried out to identify the most effective and appropriate model to serve as the core of the detection and measurement system. This evaluation is carried out by comparing the score of Recall, Precision, F1-score, Time and Mean Average Precision (mAP [0.50]) [13]. Table 3 shows the result of Recall, Precision, F1-score and Time for each iteration. Fig. 6 shows the graph depicting the mAP trend with increasing iterations.

Table 3 *Results of recall, precision, F1-score and time for each iteration*

Iteration	Recall	Precision	F1	Time (s)
1000	0.98	0.76	0.86	3
2000	0.96	0.84	0.90	2
3000	0.96	0.84	0.90	2
4000	0.96	0.84	0.90	2
5000	0.96	0.85	0.90	3
6000	0.95	0.85	0.90	3

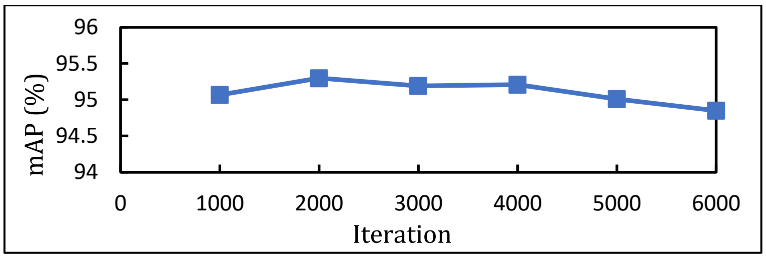


Fig. 6 *Result of mAP with different iterations*

By referring Table 3 and Fig. 6, the model with the 2000 iterations exhibits the shortest detecting time, the highest F1-score, and the second highest recall and precision values, differing only by 0.02 and 0.01, respectively, compared to the 5000-iteration result. Fig. 6 shows that the iteration with the highest mAP was 2000, and the occurrence of overfitting phenomena is observed when the iterations exceed 4000. Overfitting phenomena causes the mAP to decrease with increasing iterations. Hence, based on the analysis results, the model with 2000 iterations is the most suitable model for the button mushroom detection and measurement system.

3.2 Small-scale Field Experiment

Fig. 7 shows the small-scale experimental setup of the button mushroom detection and measurement system. The implementation of the system was carried out on actual hardware to further demonstrate its performance. As shown in Fig. 7, the setup consists of a webcam connected to an MSI laptop that runs the detection and measurement system. The webcam is positioned above a small-scale button mushroom field contained in a black tray. The webcam captures images of the small-scale button mushroom field, and feeds these images to the laptop for detecting, counting and measurement of the size of button mushrooms within each captured image.

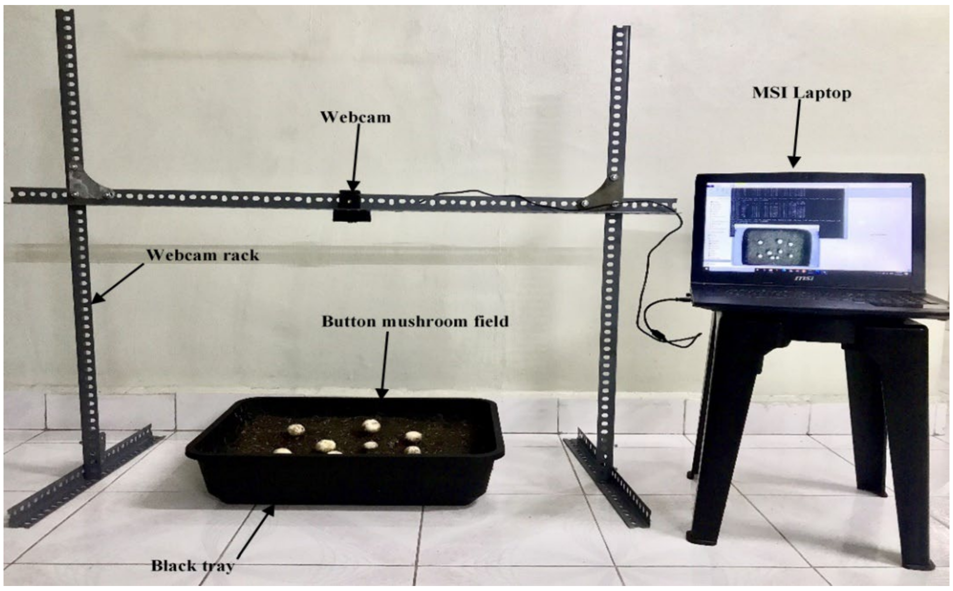


Fig. 7 *Experimental setup*

The rack for the webcam is built by using slotted angle bar with the dimension of 65 cm of height, 100 cm of length and 50 cm of width. For the small-scale button mushroom field, it uses a black tray that has the dimension of 13 cm of height, 56 cm of length and 42 cm of width. The small-scale button mushroom field is filled up with organic soil to grow the white button mushrooms. Before running through the experiment, the system needs to undergo calibration to enable precise estimation of the detected button mushroom size. By referring Equation 2, the scale ratio (R) is obtained for the system able to estimate actual size of the detected button mushrooms ($\bar{S}$). Table 4 shows the multiplication results of width (W) and height (H) of the detected button mushroom to derive the unit area (S) and the actual size unit area of each button mushroom. From the data obtained by the system, the scale ratio (R) that is used to estimate the actual size of the detected button mushrooms ($\bar{S}$) was 0.005155954 ≈ 0.005156.

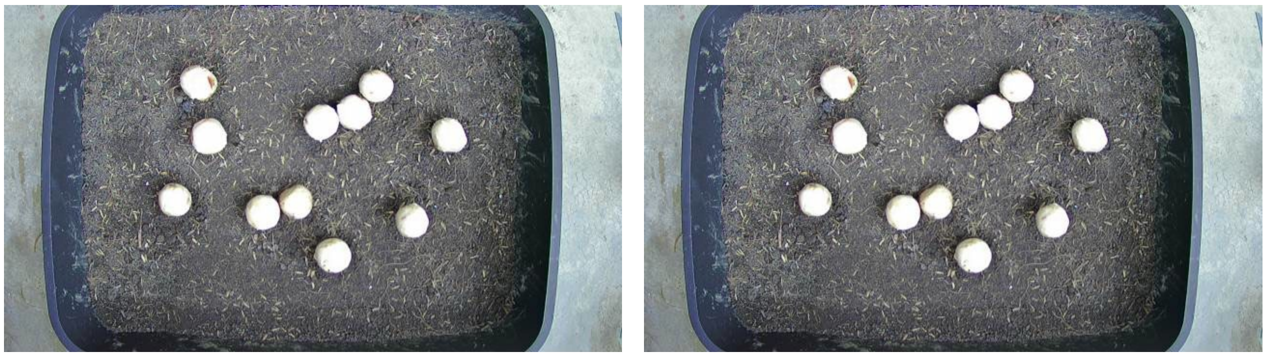
Table 4 *Results of the size (actual total area) of each button mushroom*

No	Width (W)	Height (H)	Total area (S)	Actual total area ($\bar{S}$) (cm^2)	Ratio (R)
1	43	46	1978	10.89	0.00549545
2	40	54	2160	10.55	0.005018519
3	41	50	2050	11.22	0.005258537
4	40	55	2200	11.56	0.005259091
5	40	46	1840	9.92	0.005190217
6	42	53	2226	10.51	0.005130279
7	45	51	2295	11.24	0.005102397
8	45	50	2250	10.56	0.004702222
9	37	55	2035	10.56	0.005194103
10	42	52	2184	11.55	0.005100733
11	39	51	1989	10.92	0.005263952
				Average	0.005155954

3.3 Experimental result of button mushroom detection system in a small-scaled field

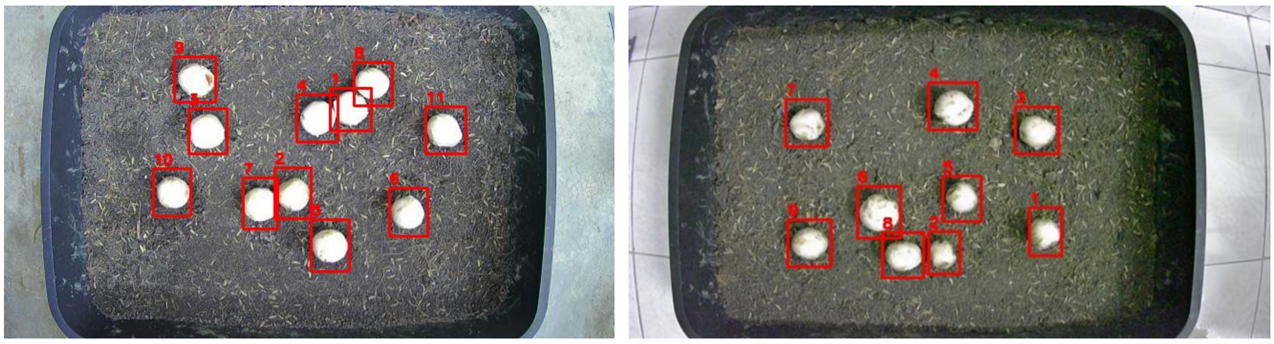
There were two (2) different setups for the small-scale button mushroom fields experiments that have been conducted, as shown in Fig. 8. The scale ratio (R), 0.005156, was set for estimating the button mushroom size in the experiments. Figs. 8(a) and 8(b) show the images of the first and second setups, denoted as setup a and setup B, respectively. These images represent the state before detecting and measuring the size of the mushrooms. On the other hand, Fig. 9 depicts the same setups, but with the result obtained after running through the detection and measurement system. The figures show how the button mushrooms were detected and measured correctly, and each detected mushroom is enclosed in red bounding boxes and labeled with a corresponding number.

Tables 5 and 6 show the results produced by the detection and measurement system for setup a and setup B, respectively, after the detection process.



(a) Setup A **(b) Setup B**

Fig. 8 *Images of two different experimental setups before undergoing detection using the proposed system*



(a) Setup A **(b) Setup B**

Fig. 9 *Images of two different experimental setups after undergoing detection using the proposed system*

Table 5 *Data of each detected and measured button mushroom size for experiment in setup A*

No.	Name	Width (cm)	Height (cm)	Measured Area (cm^2)	Actual Area (cm^2)	Percentage error (%)
1	Mushroom 1	3.087615	3.30303	10.19848	10.56	3.42
2	Mushroom 2	2.72859	3.949275	10.77595	10.55	2.14
3	Mushroom 3	2.8722	3.518445	10.10568	10.56	4.30
4	Mushroom 4	3.087615	3.662055	11.30702	11.24	0.60
5	Mushroom 5	2.8722	4.02108	11.54935	11.56	0.09
6	Mushroom 6	2.8722	3.73386	10.72439	10.51	2.04
7	Mushroom 7	2.58498	3.949275	10.2088	10.56	3.33
8	Mushroom 8	2.8722	3.374835	9.693201	9.92	2.29
9	Mushroom 9	3.087615	3.518445	10.8636	10.56	2.88
10	Mushroom 10	2.800395	3.662055	10.2552	10.24	0.15
11	Mushroom 11	3.01581	3.662055	11.04406	11.55	4.38
					Average	2.33

Table 6 *Data of each detected and measured button mushroom size for experiment in setup B*

No.	Name	Width (cm)	Height (cm)	Measured Area (cm^2)	Actual Area (cm^2)	Percentage error (%)
1	Mushroom 1	2.29776	3.73386	8.579514154	8.9	3.60
2	Mushroom 2	2.513175	3.30303	8.301092420	7.95	4.42
3	Mushroom 3	3.087615	3.662055	11.30701595	11.22	0.78
4	Mushroom 4	3.44664	3.662055	12.62178525	12.95	2.53
5	Mushroom 5	2.72859	3.518445	9.600393843	9.3	3.23
6	Mushroom 6	3.087615	4.02108	12.41554692	12.96	4.20
7	Mushroom 7	3.01581	3.662055	11.04406209	10.54	4.78
8	Mushroom 8	2.944005	3.518445	10.35831967	10.54	1.72
9	Mushroom 9	3.087615	3.662055	11.30701595	11.22	0.78
					Average	2.89

The experiment data described in the previous subchapter was collected, and analysis has been done to find the percentage error between the area results measured by the system and the area results measured manually based on equation 3. Tables 5 and 6 also show the percentage errors for each mushroom in setup a and setup B, respectively. Based on Table 5, it is found that the maximum percentage error in setup A is 4.38% and minimum percentage error is 0.09%. On the other hand, based on experiment in setup B as shown in Table 6, it is found that the maximum percentage error is 4.78% and minimum percentage error is 0.78%. Since all the percentage error results for each button mushroom in both setups are below 5%, the results obtained from the system are acceptable.

4. Conclusion

This study demonstrates the development of a detection and measurement system for button mushrooms using the YOLOv4 algorithm, a deep learning-based CNN model. The proposed system aims to automate the detection and measurement of button mushrooms, reducing the need for manual checks and optimizing resource utilization. The results of evaluation of the YOLOv4 model with different iterations showed that the model with 2000 iterations is the most suitable for the button mushroom detection and measurement system. It exhibited the shortest detecting time, highest F1-score and second-highest recall and precision values, with a negligible difference compared to the 5000-iteration result. The occurrence of overfitting phenomena was observed when the iterations exceeded 4000 that resulted in a decrease in mean average precision (mAP). Furthermore, the implementation of the system on actual hardware for two small-scale experiments demonstrated the successful detection and measurement of button mushrooms the images captured using the system correctly detected and measured each button mushrooms with a maximum percentage error below 5%.

For future recommendations, to scale up for commercial application, the system should undergo testing in larger mushroom farms. This will require further scalability testing and optimization under various environmental conditions and a more extensive dataset. Moreover, integrating the system with robotic harvesting mechanism will complete the automation process. This integration would enable the system to not only monitor mushroom growth, but also automatically harvest mature mushrooms, thereby reducing labor costs and increasing efficiency. In conclusion, the developed detection and measurement system for button mushrooms demonstrates promising results and holds the potential to revolutionize mushroom cultivation practices. The introduction of automated detection and measurement processes can contribute to more sustainable and efficient mushroom farming practices, addressing the challenges of overpopulation and food shortages in an environmentally friendly manner.

Acknowledgement

This research was supported by Universiti Tun Hussein Onn Malaysia (UTHM) through Tier 1 (vot Q368) and Ministry of Higher Education (MOHE) through Fundamental Research Grant Scheme for Research Acculturation of Early Career (FRGS-RACER) (RACER/1/2019/TK04/UTHM//5).

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Radzi Ambar, Lio Wei Yong;* ***data collection:*** *Lio Wei Yong;* ***analysis and interpretation of results:*** *Lio Wei Yong, Radzi Ambar, Mohd Helmy*

Abd Wahab, Muhammad Mahadi Abd Jamil; ***draft manuscript preparation:*** *Lio Wei Yong, Chew Chang Choon, Radzi Ambar. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] Human Population Growth and Climate Change (2020). Retrieved March 20, 2024, from https://www.biologicaldiversity.org/programs/population_and_sustainability/climate/

[2] Rabin, R. C. (2018, January 19). What Is the Health and Nutritional Value of Mushrooms?. The New York Times. Retrieved March 21, 2024 from https://www.nytimes.com/2018/01/19/well/eat/what-is-the-health-and-nutritional-value-of-mushrooms.html.

[3] Pravinth Raja, S., P.Roger Rozario, A., Nagarani, S., & S.Kavitha, N. (2018). Intelligent Mushroom Monitoring System. *International Journal of Engineering & Technology*, 7(2.33), 1238-1242. https://doi.org/10.14419/ijet.v7i2.33.18110

[4] Sumit, S. (2018, December 16). Data Science. Towards Data Science. Retrieved February 11, 2024, from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[5] LeCun, Yann, Boser, Bernhard, E., Denker, S. John, Henderson, Donnie, Howard, E. Richard, Hubbard, E. Wayne, Jackel, & D. Lawrence (1990). Handwritten digit recognition with a back-propagation network. In D. Touretzky (Ed.), *Advances in Neural Information Processing Systems*, 2. 369-404.

[6] Girshick, R (2015). Fast R-CNN. 2015 IEEE International Conference on Computer Vision (ICCV), 1440-1448. https://doi.org/10.1109/ICCV.2015.169

[7] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 779-788. https://doi.org/10.1109/CVPR.2016.91

[8] Araujo, R. M., & Ballester, P. L. (2016). On the performance of GoogLeNet and AlexNet applied to sketches. Thirtieth AAAI Conference on Artificial Intelligence. https://doi.org/10.1609/aaai.v30i1.10171

[9] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778. https://doi.org/10.1109/CVPR.2016.90

[10] Ren, S., He, K. Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137-1149. https://doi.org/10.1109/TPAMI.2016.2577031

[11] He, K., Gkioxari, G., Dollar, P., & Girshick, R. (2017). Mask R-CNN. IEEE International Conference on Computer Vision (ICCV), 2961-2969. https://doi.org/10.1109/ICCV.2017.322

[12] Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 6517-6525. https://doi.org/10.1109/CVPR.2017.690

[13] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. Computer Science, arXiv: 1804.02767. https://doi.org/10.48550/arXiv.1804.02767

[14] Liu, G., Nouaze, J., C., Mbouembe, P. L. T., & Kim, J., H. (2020). YOLO-Tomato: A Robust Algorithm for Tomato Detection Based on YOLOv3. *Sensors*, 20 (7), 2145. https://doi.org/10.3390/s20072145

[15] Bochkovskiy, A., Wang, C.-Y., & Liao, H.,-Y., M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934. https://doi.org/10.48550/arXiv.2004.10934

[16] Ultralytics. YOLOv5. Retrieved January 23, 2024, from https://github.com/ultralytics/yolov5

[17] Meituan. YOLOv6. Retrieved January 24,2024, from https://github.com/meituan/YOLOv6

[18] Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv:2207.02696. https://doi.org/10.48550/arXiv.2207.02696

[19] Ultralytics. YOLOv8. Retrieved January 24, 2024, from https://github.com/ultralytics/ultralytics

[20] Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification. *Computational Intelligence and Neuroscience*, 1-11. https://doi.org/10.1155/2016/3289801

[21] Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., & Mccool, C. (2016). DeepFruits: A Fruit Detection System Using Deep Neural Networks. *Sensors*, 16(6), 1222. https://doi.org/10.3390/s16081222

[22] Bargoti, S., & Underwood, J. (2017). Deep fruit detection in orchards. 2017 IEEE International Conference on Robotics and Automation (ICRA), 3626-3633. https://doi.org/10.1109/ICRA.2017.7989417

[23] Tu, S., Xue, Y., Zheng, C., Qi, Y., Wan, H., Mao, L. (2018). Detection of passion fruits and maturity classification using Red-Green-Blue Depth images. Biosyst. Eng. 175, 156–167. https://doi.org/10.1016/j.biosystemseng.2018.09.004

[24] Wang, D. et al. (2020). Deep learning approach for apple edge detection to remotely monitor apple growth in orchards. *IEEE Access*, 8, 26911–26925. https://doi.org/10.1109/ACCESS.2020.2971524