

## Detection and counting of wheat ear using YOLOv8

Muhammad Sabri Mas, Sofia Saidah, Nur Ibrahim

School of Electrical Engineering, Telkom University, Bandung, Indonesia

### Article Info

#### Article history:

Received Oct 24, 2023

Revised Mar 25, 2024

Accepted Apr 16, 2024

#### Keywords:

Deep learning

Detection and counting

Mean average precision

Wheat ear

YOLOv8

### ABSTRACT

Detection and calculation of wheat ears are critical for land management, yield estimation, and crop phenotype analysis. Most methods are based on superficial and color features extracted using machine learning. However, these methods cannot fulfill wheat ear detection and counting in the field due to the limitations of the generated features and their lack of robustness. Various detectors have been created to deal with this problem, but their accuracy and calculation precision still need to be improved. This research proposes a deep learning method using you only look once (YOLO), especially the YOLOv8 model with depth and channel width configuration, stochastic gradient descent (SGD) optimizer, structure modification, and convolution module along with hyperparameter tuning by transfer learning method. The results show that the model achieves a mean average precision (mAP) of 95.80%, precision of 99.90%, recall of 99.50%, and frame per second (FPS) of 22.08. The calculation performance of the wheat ear object achieved accurate performance with a coefficient of determination ( $R^2$ ) value of 0.977, root mean square error (RMSE) of 2.765, and bias of 1.75.

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



### Corresponding Author:

Muhammad Sabri Mas

School of Electrical Engineering, Telkom University

Jalan Telekomunikasi, Terusan Buah Batu, Bandung 40257, Indonesia

Email: Sabrimol91@gmail.com

## 1. INTRODUCTION

Wheat is a cultivated crop, and its yield is essential for food security. The rapid growth of the human population in the world, resulting in the demand to increase crop yields, becomes more urgent [1]. Data released by Statista in 2023 shows that the volume of world wheat production reached more than 778 million tonnes in 2021-2022, an increase of 4 million tonnes compared to the previous year. Wheat production is forecast to increase by about 286 million tonnes worldwide by 2023. Wheat yield forecasting is a critical part of the agricultural management process that can be the reference for field operations and agricultural decision-making. Identifying and counting wheat ears helps monitor growth, forecast wheat yield, and analyze crop phenotype characteristics. Automatic detection of wheat ears not only improves work efficiency but is also essential for the development of intelligent agricultural machinery. An efficient and automated algorithm for detecting and counting wheat ears is needed.

The conventional method for calculating wheat ear numbers involves a manual process that is time-consuming and susceptible to errors. The calculation is done by manual yield prediction in the field, capacity prediction, annual data prediction, and remote sensing image estimation [2]. These methods are empirical, low accuracy, and also labor intensive. The volumetric techniques are expensive and inefficient in measuring wheat density. Remote sensing is based on satellite images used as samples. Because the images are far apart, they are only suitable for large-scale processing and analysis, resulting in low accuracy of wheat prediction. On the other hand, multiple linear regression-based predictions are heavily influenced by weather factors, making their accuracy hard to ensure and unsuitable for field yield estimation.

The presence of computer vision in the field of research indicates that earlier investigations employed machine-learning techniques to identify wheat ear objects. In Xu *et al.* study [3], the K-means algorithm was applied to automatically segment wheat ear images, followed by their utilization in training and testing convolutional neural network (CNN) models. Research by Grbović *et al.* [4] also uses segmentation and thermal images and then compares them with ground truth to assess the accuracy of the system used. Fernandes-Gallego *et al.* [5] performed detection by utilizing a red-green-blue (RGB) camera and then built an automatic wheat ear calculation system with three steps, namely with Laplacian frequency filter, median filter, and find maxima segmentation. Although wheat ear recognition has achieved good results, most of these methods require artificially adjusting the image, causing insufficient accuracy in different environments or farmlands. Noise disturbances such as the level of exposure and background disturbances also make the detection and calculation of wheat ears not optimal.

Deep learning has achieved impressive results in various fields in recent years. Significant progress has been made in target detection technology, which is one of the core problems in computer vision. Deep learning-based target detection algorithms are divided into multi-stage and single-stage. All these algorithms use CNN, which is based on the convolution multiplication of the input image with a kernel and then produces a feature map that will be reprocessed to classify an object [6]. Standard target detection algorithms for multi-stage models are region-based fully convolutional network (R-FCN) [7], region-based convolutional neural network (R-CNN) [8], Fast R-CNN [9], Faster R-CNN [10], Mask-RCNN [11] and for single-stage models is you only look once (YOLO). In the multi-stage model, the first step is to determine candidate regions that may contain targets to be detected. It then performs detailed identification of the targets in each candidate region to perform classification and regression. Despite its good accuracy in object detection, the multi-stage model has a longer computation time due to its complex structure. In contrast, the single-stage model is more effective and efficient by using the help of anchors and grid boxes to localize the target region and constrain the object shape to predict the bounding box in one step. YOLO has achieved several developments and model improvements, starting with YOLO9000 [12], YOLOv3 [13], YOLOv4 [14], YOLOv5, YOLOv6 [15], YOLOv7 [16], and YOLOv8.

Various studies on wheat ear detection using the YOLO model have shown promising results. The research conducted by Yang *et al.* [17] modified YOLOv4 by adding a convolutional block attention module (CBAM) [18] and a dual-channel (attention and spatial) module to the neck layer. The results show that the model can remove background noise and perform well on three datasets, namely WD, WEDD, and GWHD datasets, with mAP of 94.00%, 96.40%, and 93.11%, respectively. Zhao *et al.* [19] improved the YOLOv4 receptive field with spatial pyramid pooling (SPP) [20] in the feature fusion section to extract multi-scale features. The goal is for the resulting features to have solid and robust location information of the wheat object. The method showed an accuracy of 96.40% on the HRED dataset and 93.11% on the GWHD dataset. Li and Wu [21] used the YOLOv5 model and improved the perceptual field by sampling four times in the feature pyramid to improve small target detection. In addition, the research also added a CBAM module along with attention and spatial modules to overcome the problem of decreasing gradients during training. The accuracy of the model was 94.32% using the GWHD dataset. Meng *et al.* [22] used the YOLOv7 model on the 2021 GWHD dataset with a mAP of 93.86% with FPS 35.93,  $R^2$  0.9895 (low light), 0.9872 (blur) and 0.9882 (occlusion).

Some of the studies mentioned above show that models trained on several datasets from certain regions need to be more generalizable and experience a decrease in accuracy due to overlapping and varying sizes of wheat ear objects. In addition, the previous research also did not display how many total estimated objects were detected in the processed image, so rechecking the output of the generated code is required. This research uses the YOLOv8 model as the base model and then enhances it with several tests to be more suitable for detecting wheat ears on complex backgrounds. YOLOv8 will be optimized gradually by performing several test scenarios. First, the effect of convolution depth and channel width values of each YOLOv8 variant was tested. Second, several optimizers such as stochastic gradient descent (SGD), Adam, AdamW, and RMSProp are tested to determine which optimizer is suitable for wheat ear detection. Thirdly, layer modification and convolution modules are performed to strengthen detection. Finally, the hyperparameters were tuned with the transfer learning method for maximum accuracy.

## 2. METHOD

### 2.1. Dataset

We use the original distribution dataset to compare the model performance with previous research methods, where the GWHD 2020 wheat ear dataset used in this research consists of 3,422 training images and 1,276 testing images used for model performance evaluation [23]. The dataset is an RGB image with a 1024×1024 pixels resolution and an annotation file in txt format. The GWHD dataset was collected between

2016 and 2019 by 9 institutions in 10 locations covering genotypes from Europe, North America, Australia, and Asia. The GWHD dataset consists of sub-datasets obtained with different planting practices and row spacing varying from 12.5 cm (ETHZ\_1) to 30.5 cm (USASK\_1), planted with average planting density (*Arvalis\_1*, *Arvalis\_2*, *Arvalis\_3*, *INRAE\_1 part of NAU\_1*) and high seed density (*RRES\_1*, *ETHZ\_1 part of NAU\_1*). The GWHD dataset covers a wide range of pedoclimatic conditions, such as being grown in the Picardy region of France (*Arvalis\_3*), produced in mountainous or highland regions of Switzerland (*ETHZ\_1*), or the Alpes de Haute Provence (*Arvalis\_1*, *Arvalis\_2*). Figure 1 shows image samples from the GWHD dataset distribution.

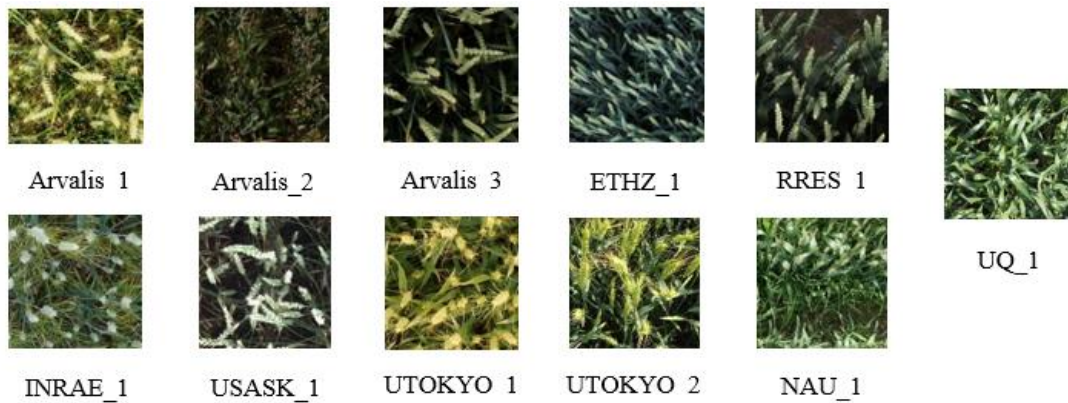


Figure 1. Some example images

## 2.2. YOLOv8 algorithm

YOLOv8 is the latest YOLO model released in 2023, precisely in January, by Ultralytics, also the developer of YOLOv5. YOLOv8 is a detector consisting of 2 main parts: the backbone network and the head network. The backbone model is used to extract features from a given input image. This model uses a modified version of the cross stage partial network, especially CSPDarknet53 architecture, as the backbone, which consists of 53 convolutional layers and uses a technique called cross-stage partial connection to improve the flow of information between the various network layers. Meanwhile, the head network contains several convolution layers and a series of fully connected layers. These layers are responsible for the predicted bounding box, object score, and class probabilities for the detected objects. In addition, there is also attention in the head network that allows the model to focus on different parts of the image and adapt various features based on their relevance to the given task.

The YOLOv8 classification loss function uses BCE loss. Regression loss takes the form of CIOU loss+DFL and VFL, which proposes an asymmetric weighting operation. The DFL of the box position is modeled as a general distribution. The network focuses faster on the distribution of locations close to the object location, and the probability density is as relative as possible to that location. The following is shown in (1).

$$DFL_{(s_i s_{i+1})} = -((y_{i+1} - y) \log(s_i) + (y - y_i) \log(s_{i+1})) \quad (1)$$

where  $s_i$  is the sigmoid output for the network,  $y_i$  and  $y_{i+1}$  are the interval orders, and  $y$  is the label. Compared to previous models, YOLOv8 can be easily extended and contains a framework that can support earlier versions of YOLO and switch between different versions [24]. In addition, YOLOv8 implements free anchor boxes where the model detects objects directly without looking at the offset of known anchor boxes. Anchor boxes are predefined boxes with a specific width and height that see object classes with the desired scale and aspect ratio. They are selected based on the size of the objects in the training dataset and are arranged on the image during detection. Figure 2 shows the architecture of YOLOv8 when extracting input images.

It can be seen in Figure 2 that first, the input image is given, and then the feature extraction process is carried out in stages following the concept of FPN. Each layer in P1, P2, P3, P4, and P5 has a different layer size and depth. It is intended to find the location and position of the object. Then, the last three layers perform extraction again before entering the object detection section. The detection head applies the localization metric as previously described.

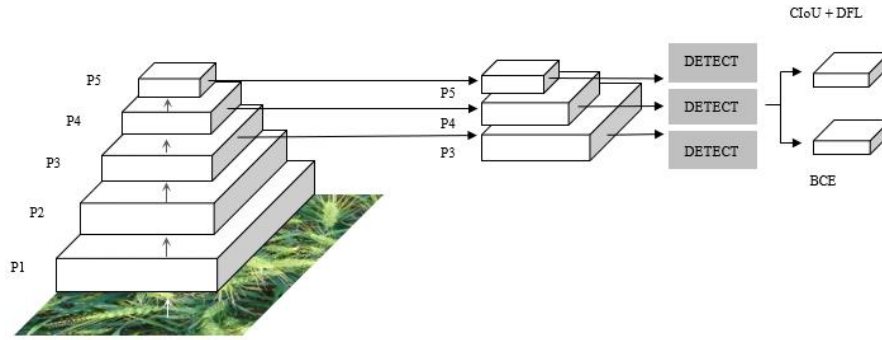


Figure 2. YOLOv8 architecture

**2.3. System design**

The training and testing process is shown in Figure 3. Based on the flowchart, training starts with preparing the train data and the YOLOv8 model and then preparing the model configuration, such as selecting parameters such as training iterations of 600 epochs and selecting an image size of 1,024 following the original size of the dataset. During the training process, a fine-tuning method is also performed to find the proper adjustment to the object. The training process produces pre-trained weights, which will then be reused in the testing process. Object detection and calculation based on evaluation metrics are performed in this phase. In addition, adjustments or model configurations are made again to optimize the performance of the YOLOv8 base model. Some of the configurations carried out in this study include the following.

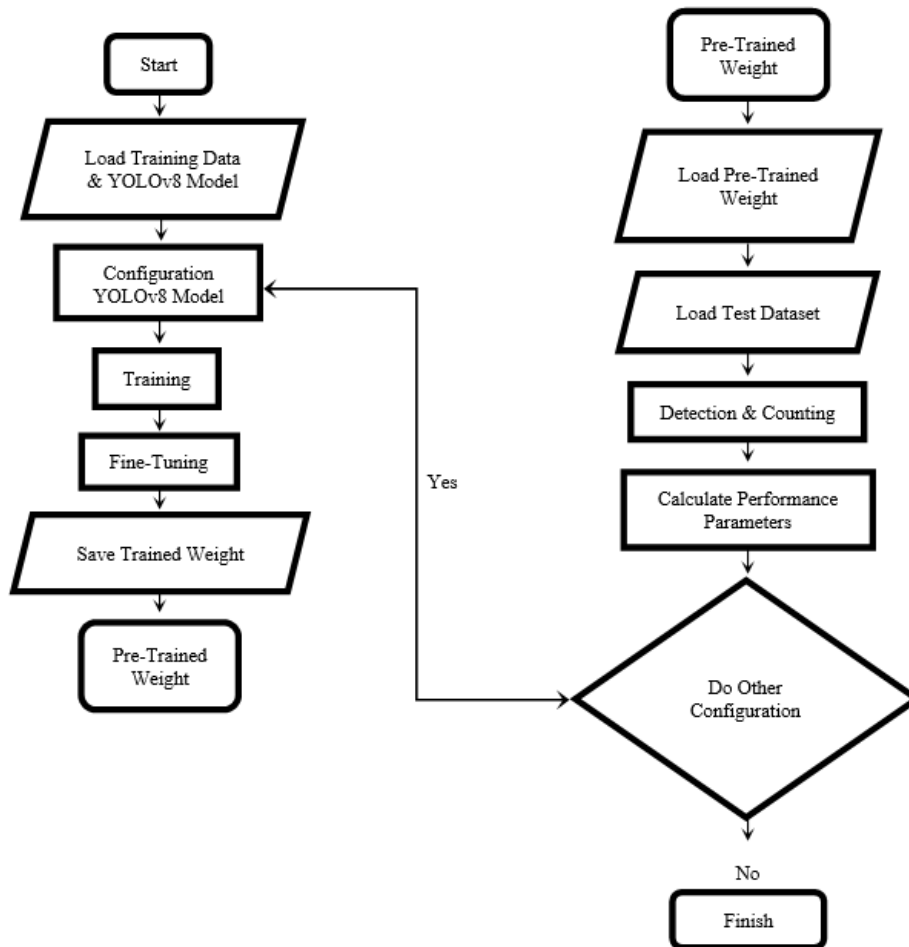


Figure 3. Flowchart training and testing

### 2.3.1. Testing the depth multiple and width multiple values

In the first test, the model training is performed on the depth multiple variables, which is a multiple of the channel depth and width multiple, which is a multiple of the model layer channel. In general, the YOLOv8 model consists of 5 models namely YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l and YOLOv8x. Larger multiples of depth indicate adding more layers to the neural network, while multiples of width add more filters to the layers, adding more channels to the layer outputs. The addition of both makes the model larger and requires more computing power. The depth multiple and width multiple values of each of the five models are shown in Table 1.

Table 1. Comparison of depth multiple and width multiple values

Model	Depth multiple	Width multiple
YOLOv8n	0.33	0.25
YOLOv8s	0.33	0.50
YOLOv8m	0.67	0.75
YOLOv8l	1.00	1.00
YOLOv8x	1.00	1.25

### 2.3.2. Testing the optimizer

This test aims to compare optimizers against evaluation metrics. The tested optimizers consist of SGD [25], Adam [26], AdamW [27] and RMSProp [28]. The primary role of using the right optimizer is to minimize the model's error or loss of function, thereby improving performance. The best optimizer will be selected to optimize the hyperparameter value in the following test scenario.

### 2.3.3. Testing of architecture layers and modules

This test uses several modules of the YOLOv8 architecture with several different structures. The modules used in this testing are those available in the YOLOv8 repository. Additionally, we made convolutional layer adjustments to several models under test to assess the impact of depth on model performance. We experimented with multiple available models, subsequently modifying convolution and attempting combinations of convolution modules in the backbone and head networks. The aim is to identify which model exhibits stronger feature extraction and achieves the highest accuracy.

### 2.3.4. Hyperparameter tuning and transfer learning

Hyperparameter tuning refers to finding optimal values for the model's hyperparameters [29]. This test aims to find the right combination to improve the model's performance and generalization ability significantly. In addition, transfer learning is also performed to ease the heavy computational burden [30]. The results are analyzed based on several performance metrics, such as precision, recall, F1-score, and FPS. In addition, ear of grain calculation metrics such as  $R^2$ , RMSE, and bias are used.

$$P = \frac{TP}{TP + FP} \quad (2)$$

$$R = \frac{TP}{TP + FN} \quad (3)$$

$$F1 - Score = \frac{2 \times P \times R}{P + R} \quad (4)$$

$$mAP = \frac{\sum_{i=1}^N AP_i}{N} \quad (5)$$

Among these metrics, the true positive (TP) represents instances where the detection correctly identifies a wheat ear, matching the actual presence of a wheat ear. Conversely, the false positive (FP) occurs when the detection falsely identifies an ear of wheat when it is part of the background, indicating incorrectly counted wheat ears. The false negative (FN) corresponds to situations where the background is mistakenly labeled as a wheat ear, signifying uncounted wheat ears. Precision (P) quantifies the proportion of correct wheat ear detections for all predictions. Recall (R) calculates the ratio of true positives to the total actual wheat ears, revealing how effectively the model identifies positive cases among all real positives. The F1-score offers a means of assessing the method's performance by balancing the importance of precision and recall. In this context, mAP represents the mean average precision achieved in this evaluation.

$$R = \frac{TP}{TP + FN} \quad (6)$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (m_i - c_i)^2} \quad (7)$$

$$Bias = \frac{1}{n} \sum_{i=1}^n (m_i - c_i) \quad (8)$$

Coefficient of determination ( $R^2$ ), and root mean square error (RMSE) bias are used as evaluation indicators to measure the counting performance of the model. Where  $n$  represents the number of wheat ear images,  $m_i - c_i$  represents the number of wheat ears manually labeled and counted by the model in the  $i$  image, and  $\bar{m}$  represents the average number of wheat ears. The bias signifies the mean count for each detected image and the actual discrepancy.

### 3. RESULTS AND DISCUSSION

Model training and validation were performed using the original distribution of the GWHD dataset. The hardware used in this experiment is an Intel i5-12400F processor and NVIDIA GeForce RTX 3080 GPU with 12 GB VRAM, which is implemented using PyTorch deep learning framework and Python programming. In addition, the CUDA 11.8 parallel computing framework and CUDNN 8.9.4 deep neural network acceleration library are used in this research.

#### 3.1. Experiment result

##### 3.1.1. Test results for depth multiple and width multiple values

Based on the test results of the depth multiple and width multiple values, it can be seen in Table 2 that the YOLOv8x model has the best accuracy (mAP) of 91.00%. In addition, the YOLOv8x model also has a precision value of 99.50%, recall of 99.10%, and F1-score of 99.30%, where these values are the highest values of the other four models. Despite having the best performance, the YOLOv8x model has the lowest FPS value of 24.27. This is due to the increased convolution and channel width. The model has a larger depth multiple and width multiple values, so the model is heavier. In contrast, the lightest YOLOv8n model has the highest FPS of 188.68, but the accuracy is very low at 68.60%.

Table 2. Test results for depth multiple and width multiple values

Model	Depth multiple	Width multiple	mAP	Precision	Recall	F1-score	FPS
YOLOv8n	0.33	0.25	68.60%	95.10%	94.40%	94.75%	188.68
YOLOv8s	0.33	0.5	68.60%	95.10%	94.40%	94.75%	192.31
YOLOv8m	0.67	0.75	87.20%	99.20%	99.10%	99.15%	60.98
YOLOv8l	1	1	87.60%	99.00%	98.80%	98.90%	40.49
YOLOv8x	1	1.25	91.00%	99.50%	99.10%	99.30%	24.27

##### 3.1.2. Test result for optimizer

The following test scenario looks at the effect of the optimizer on the model. The model chosen for this test is based on the previous test, namely the YOLOv8x model, which has a depth multiple of 1 and a width multiple of 1.25. Some of the optimizers tested were SGD, Adam, AdamW and RMSProp. Table 3 shows the results of testing several types of optimizers.

Table 3. Test results for optimizer

Optimizer	mAP	Precision	Recall	F1-score	FPS
SGD	89.20%	99.30%	99.10%	99.20%	24.45
Adam	69.30%	95.40%	95.10%	95.25%	24.57
AdamW	84.90%	98.70%	98.70%	98.70%	24.45
RMSProp	22.30%	62.10%	48.00%	54.15%	20.20

Based on the table, the SGD optimizer is suitable for the model and has the highest mAP value of 89.20%. Likewise, the precision, recall, and F1-score values outperform other optimizers, with 99.30%, 99.10%, and 99.10%, respectively. This test also shows that the SGD, Adam, and AdamW optimizers do not experience overfitting, while the RMSProp optimizer experiences overfitting.

### 3.1.3. Test results for architecture layers and modules

This test uses the configuration from the previous best test. The three models tested shown in Table 4 used the same configuration: a depth multiple values of 1, a width multiple values of 1.25, and the SGD optimizer. The YOLOv8x-P2 model has 1 more extensive detection layer added before the original 3 layers in the head layer. The YOLOv8x-P6 model uses 1 extra smaller detection layer after the initial 3 layers in the head layer consisting of. The Mod-YOLOv8x-P6 model has the same structure as the YOLOv8x-P6 model, with deeper convolution layers in some backbone layers.

Based on the table, it can be seen that the YOLOv8x-P2 model has the lowest accuracy, where its accuracy has decreased by 3.4% mAP from the original YOLOv8x model with 91.00% mAP. In addition, it also has a very low FPS of 15.4. The model produces a less robust feature map despite having a deep convolution layer to extract objects. The YOLOv8x-P6 model has a 0.8% mAP improvement from the original model (YOLOv8x), and the FPS has decreased by 1.77. By modifying the convolution layer and applying the C2F module to the backbone and head, which previously the head network used the C2 module, the YOLOv8x-P6 modified model was shown to have an increase of 1.5% from the original model (YOLOv8x) with a mAP of 92.50% along with an increase in precision, recall, and F1-score of 99.60%, 99.30%, and 99.40% respectively and an FPS value of 22.1.

Table 4. Test results for architecture layers and modules

Model	mAP	Precision	Recall	F1 score	FPS
YOLOv8x-P2	87.60%	98.80%	98.70%	98.70%	15.4
YOLOv8x-P6	91.80%	99.50%	99.30%	99.40%	22.5
Mod-YOLOv8x-P6 (proposed model)	92.50%	99.60%	99.30%	99.40%	22.1

### 3.1.4. Hyperparameter tuning results and transfer learning method

The previously trained Mod-YOLOv8x-P6 model will be retrained by adjusting the hyperparameters. Transfer learning will also be carried out in preparing the model by utilizing the pre-trained weights from the previously trained YOLOv8x-P6 model. The results can be shown in Table 5 with the hyperparameter configuration used in this test is the initial learning rate of 0.0025, the final learning rate of 0.001, the momentum of 0.1, and the weight decay of 0.0005. It can be seen from Figure 4 that the model accuracy curve increases gradually with increasing iterations. After passing 300 iterations, precision, recall, and mAP show a flat curve, which means no more improvement and indicates the model has reached the optimal ability.

Table 5. Hyperparameter tuning results and transfer learning method

Model	mAP	Precision	Recall	F1-score	FPS
Proposed model	95.80%	99.90%	99.50%	99.90%	22.08

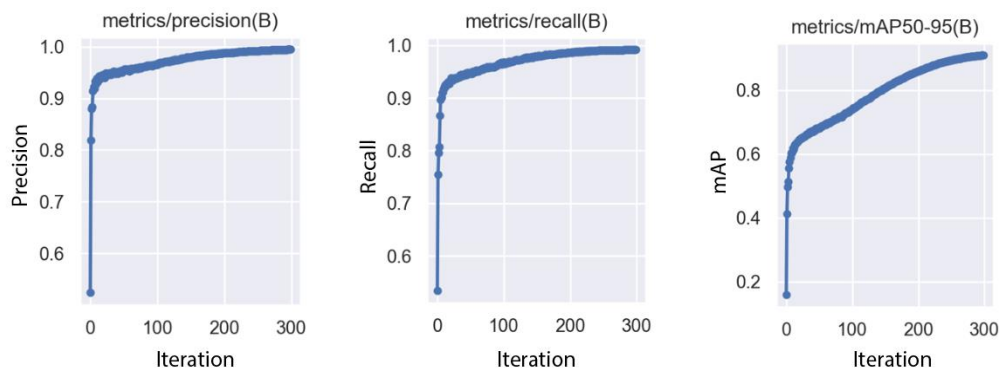


Figure 4. Proposed model curve

### 3.2. Calculation of wheat ears

When testing the detection effect of the proposed model, 10 images with wheat ear objects aligned or called slightly oriented and 10 images with wheat ear objects more varied or heavily oriented are selected randomly from the test dataset, as exemplified in Figure 5. Then, the proposed model will be used to detect these datasets and determine the success of the model. Examples of detection results can be observed in

Figure 6. Figures 6(a) and 6(b) are images with slightly oriented wheat ear objects, while 6(c) and 6(d) are images with heavily oriented wheat ear objects.

Detected wheat ears are marked with a red box with an object label. The calculation results of wheat ear objects are shown in each image's upper right corner area. This estimates how many objects are detected, making the calculation more efficient when the model is applied to the camera. Then, each saw wheat ear object has a label to ensure no repeated calculations occur. In Figure 6(a), 31 objects were detected, and 3 failed, with 91.17% accuracy. Figure 6(b) detected 14 objects, 0 failures, and 100% accuracy. Figure 6(c) saw 29 objects, 1 failure, and 96.66% accuracy. Figure 6(d) detected 25 objects, 3 fails, and 89.28% accuracy. According to the results, only a few wheat ear objects are left behind. This is because some objects overlap with each other. The sample images represent some randomly selected sample images. To determine the success of the object, model performance metrics such as  $R^2$ , RMSE, and bias are used, as shown in Figure 7. Here is a graph that compares the actual value with the estimated value.

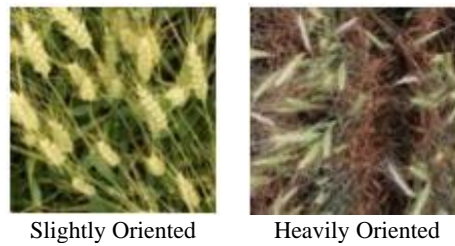


Figure 5. Random sample image

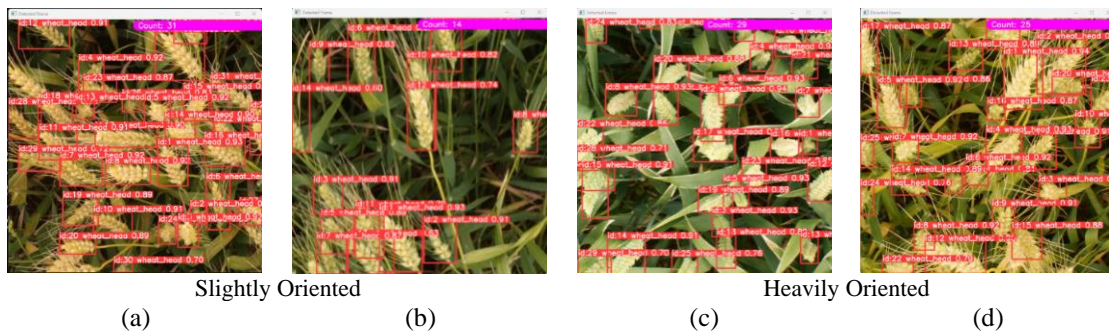


Figure 6. Some of the sample images, such as 6(a) and 6(b), are slightly oriented, while 6(c) and 6(d) are heavily oriented

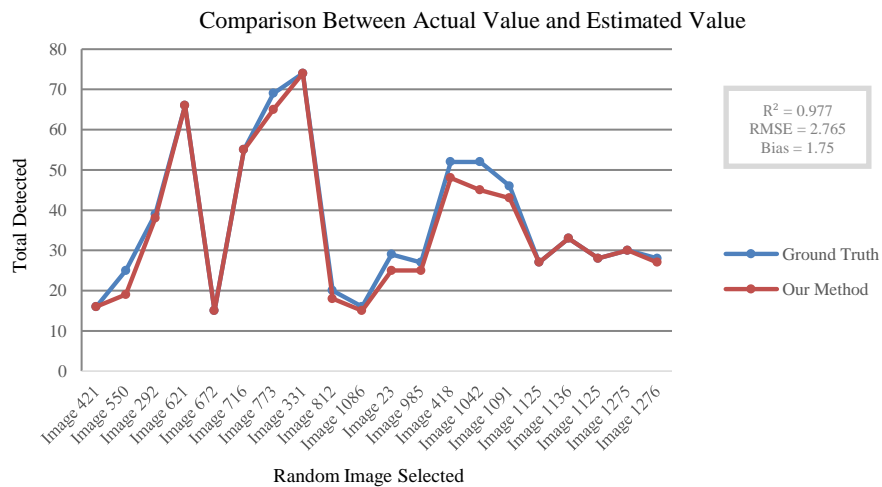


Figure 7. Sample detection results



### 3.3. Comparison with previous methods

After going through the training, testing, and calculating objects, comparisons with previous research methods, including CBAM-YOLOv4, SPP-YOLOv4, and CBAM-YOLOv5, using the GWHD 2020 dataset results are shown in Table 6. The table reveals that the proposed model attains the second-best mAP value of 95.80%, outperforming both the CBAM-YOLOv4 and CBAM-YOLOv5 models, although it lags behind SPP-YOLOv4 by a marginal 0.6%. In addition, the precision, recall, and F1-score values outperformed the 3 previous models with 99.90%, 99.50%, and 99.90%, respectively.

Table 6. Comparison with previous method

Model	mAP	Precision	Recall	F1-score	FPS
CBAM-YOLOv4 [17]	93.11%	87.55%	91.01%	89.25%	N/A
SPP-YOLOv4 [19]	96.40%	96.40%	90.24%	93.00%	51.01
CBAM-YOLOv5 [21]	94.32%	88.52%	98.06%	93.05%	N/A
Our method	95.80%	99.90%	99.50%	99.90%	22.08

The FPS value generated by the modified YOLOv8 model is 28.93 behind the SPP-YOLOv4 model, which has an FPS value of 51.01. This is due to using different GPUs, and our model has a more profound architecture. The proposed model can still detect the wheat ear object well. The other two models, CBAM-YOLOv4 and CBAM-YOLOv5, do not include FPS values in their research.

In addition, we also compare this study with the ear of wheat calculation metric in Figure 8. This comparison is done to see if the proposed model has a good ear of wheat calculation performance when compared to the three previous studies. Figure 8 displays the outcomes of the comparison.

In Figure 8, the  $R^2$ , RMSE, and bias values of CBAM-YOLOv4 are 0.988, 2.097, and -3.3, respectively. The SPP-YOLOv4 model with  $R^2$  and RMSE of 0.937 and 4.810 did not have any bias values in its study. CBAM-YOLOv5 model with  $R^2$ , RMSE, and bias values of 0.976, 3.178, and 2.37, respectively. The proposed model with  $R^2$ , RMSE, and bias values are 0.977, 2.765, and 1.75, respectively.

Based on the above, the proposed model's performance is the second-best after CBAM-YOLOv4 for the coefficient of determination and RMSE. Both parameters mean that the model more perfectly explains the variance in the target variable and provides insight into the accuracy of the prediction and how closely the model matches the actual data. As for the bias value, the proposed model has the smallest value of 1.75 compared to the other models, which indicates that the model consistently predicts higher values than the actual values.

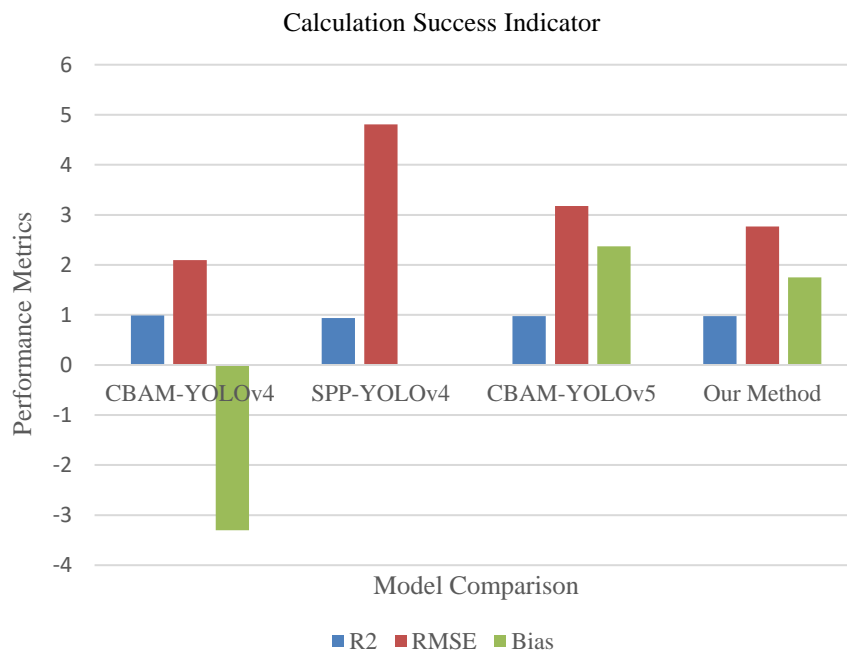


Figure 8. Comparison between  $R^2$ , RMSE and bias

#### 4. CONCLUSION

Accurately detecting and counting wheat ears provides a good reference value for food yield estimation. Farmland and lighting conditions sometimes make the detection results not optimal. Through this research, the optimized deep-learning model YOLOv8 is used to detect and count wheat ears. After the model was trained and tested on the GWHD dataset with different conditions, varieties, lighting, and farmland levels, the model achieved good accuracy. The model was gradually optimized with several configurations, such as using depth multiple and width multiple values of 1 and 1.25, respectively, using SGD optimizer, increasing the convolution layer and applying C2F module to the modified backbone and head models as well as tuning the hyperparameters and transfer learning method. The results show an increase in performance metrics, namely mAP of 95.80%, precision of 99.90%, recall of 99.50%, and F1-score of 99.90%. In addition, the FPS value obtained by the model is 22.08. The proposed model can also automatically calculate the detected wheat ear object equipped with a name label. The model's accuracy is indicated by the  $R^2$  value of 0.977, RMSE of 2.765, and bias of 1.75. The proposed model fulfills the need for automatic wheat ear detection and calculation effectively and efficiently. The model can also be integrated with drone cameras for direct field implementation.





#### REFERENCES

- [1] G. A. Slafer, R. Savin, and V. O. Sadras, "Coarse and fine regulation of wheat yield components in response to genotype and environment," *Field Crops Research*, vol. 157, pp. 71–83, Feb. 2014, doi: 10.1016/j.fcr.2013.12.004.
- [2] L. Salazar, F. Kogan, and L. Roytman, "Use of remote sensing data for estimation of winter wheat yield in the United States," *International Journal of Remote Sensing*, vol. 28, no. 17, pp. 3795–3811, Aug. 2007, doi: 10.1080/01431160601050395.
- [3] X. Xu *et al.*, "Wheat ear counting using K-means clustering segmentation and convolutional neural network," *Plant Methods*, vol. 16, no. 1, Aug. 2020, doi: 10.1186/s13007-020-00648-8.
- [4] Z. Grbović, M. Panić, O. Marko, S. Brdar, and V. Crnojević, "Wheat ear detection in RGB and thermal images using deep neural networks," *International Conference on Machine Learning and Data Mining, MLDM 2019*, 2019.
- [5] J. A. Fernandez-Gallego, S. C. Kefauver, N. A. Gutiérrez, M. T. Nieto-Taladriz, and J. L. Araus, "Wheat ear counting in-field conditions: High throughput and low-cost approach using RGB images," *Plant Methods*, vol. 14, no. 1, Mar. 2018, doi: 10.1186/s13007-018-0289-4.
- [6] Y. N. Fu'adah, S. Sa'idah, I. Wijayanto, N. Ibrahim, S. Rizal, and R. Magdalena, "Computer aided diagnosis for early detection of glaucoma using convolutional neural network (CNN)," in *Lecture Notes in Electrical Engineering*, vol. 746 LNEE, Springer Singapore, 2021, pp. 467–475.
- [7] Z. Li, Y. Chen, G. Yu, and Y. Deng, "R-FCN++: towards accurate region-based fully convolutional networks for object detection," *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, vol. 32, no. 1, pp. 7073–7080, Apr. 2018, doi: 10.1609/aaai.v32i1.12265.
- [8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 580–587, doi: 10.1109/CVPR.2014.81.
- [9] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, Dec. 2015, vol. 2015 Inter, pp. 1440–1448, doi: 10.1109/ICCV.2015.169.
- [10] 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.
- [11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, Oct. 2017, vol. 2017-October, pp. 2980–2988, doi: 10.1109/ICCV.2017.322.
- [12] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 6517–6525, doi: 10.1109/CVPR.2017.690.
- [13] J. Redmon and A. Farhadi, "YOLOv3: an incremental improvement," *arXiv preprint arXiv:1804.02767*, pp. 1–6, 2018.
- [14] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.
- [15] C. Li *et al.*, "YOLOv6: a single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, Sep. 2022.
- [16] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 7464–7475, doi: 10.1109/cvpr52729.2023.00721.
- [17] B. Yang, Z. Gao, Y. Gao, and Y. Zhu, "Rapid detection and counting of wheat ears in the field using YOLOv4 with attention module," *Agronomy*, vol. 11, no. 6, Jun. 2021, doi: 10.3390/agronomy11061202.
- [18] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "CBAM: convolutional block attention module," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11211 LNCS, Springer International Publishing, 2018, pp. 3–19.
- [19] F. Zhao, L. Xu, L. Lv, and Y. Zhang, "Wheat ear detection algorithm based on improved YOLOv4," *Applied Sciences (Switzerland)*, vol. 12, no. 23, Nov. 2022, doi: 10.3390/app122312195.
- [20] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015, doi: 10.1109/TPAMI.2015.2389824.
- [21] R. Li and Y. Wu, "Improved YOLOv5 wheat ear detection algorithm based on attention mechanism," *Electronics (Switzerland)*, vol. 11, no. 11, May 2022, doi: 10.3390/electronics11111673.
- [22] X. Meng, C. Li, J. Li, X. Li, F. Guo, and Z. Xiao, "YOLOv7-MA: improved YOLOv7-based wheat head detection and counting," *Remote Sensing*, vol. 15, no. 15, Jul. 2023, doi: 10.3390/rs15153770.





- [23] E. David *et al.*, “Global wheat head detection (GWHD) dataset: a large and diverse dataset of high-resolution RGB-labelled images to develop and benchmark wheat head detection methods,” *Plant Phenomics*, vol. 2020, Jan. 2020, doi: 10.34133/2020/3521852.
- [24] H. Lou *et al.*, “DC-YOLOv8: small-size object detection algorithm based on camera sensor,” *Electronics (Switzerland)*, vol. 12, no. 10, May 2023, doi: 10.3390/electronics12102323.
- [25] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747v2*, Sep. 2016.
- [26] D. P. Kingma and J. Ba, “Adam: a method for stochastic optimization,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–15, Dec. 2014.
- [27] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, Nov. 2017.
- [28] R. Zaheer and H. Shaziya, “A study of the optimization algorithms in deep learning,” in *Proceedings of the 3rd International Conference on Inventive Systems and Control, ICISC 2019*, Jan. 2019, pp. 536–539, doi: 10.1109/ICISC44355.2019.9036442.
- [29] T. Yu and H. Zhu, “Hyper-parameter optimization: a review of algorithms and applications,” *arXiv preprint arXiv:2003.05689*, Mar. 2020.
- [30] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” in *Proceedings of the International Conference on Artificial Neural Networks*, 2018, pp. 270–279.

## BIOGRAPHIES OF AUTHORS







**Muhammad Sabri Mas**     received a bachelor’s degree in telecommunication engineering from Telkom University, Indonesia in 2023. He has interest in deep learning, image processing and computer vision. He can be contacted at email: [sabrimol91@gmail.com](mailto:sabrimol91@gmail.com).



**Sofia Saidah**     received the B.S. and M.S. degree in telecommunication engineering from Telkom Institute of Technology, Bandung, Indonesia in 2012 and 2014 respectively. She is currently a lecturer in School of Electrical Engineering Telkom University. Her research interest includes, image processing, audio processing, biomedical engineering, steganography and watermarking. She can be contacted at email: [sofiasaidahsfi@telkomuniversity.ac.id](mailto:sofiasaidahsfi@telkomuniversity.ac.id).



**Nur Ibrahim**     was born in Bandung, Indonesia in 1987. He received a bachelor’s degree in telecommunication engineering from Institut Teknologi Bandung, Indonesia, in 2009 and a master’s degree from Institut Teknologi Bandung, Indonesia, in 2011. He is currently a doctoral candidate at the Department of Electrical Engineering, Universitas Indonesia, Indonesia. His research interests are related to artificial intelligence, machine learning, and signal processing. He can be contacted at email: [nuribrahim@telkomuniversity.ac.id](mailto:nuribrahim@telkomuniversity.ac.id).