frontiers

Wheat Head Detection from High Resolution RGB Images Using Convolutional Neural Network

Xinyu Yao [1] **and Selvyn Perez** [1]

[1] *Data Science, Columbian College of Arts and Science, George Washington University, Washington, DC, USA*

Correspondence*:
Xinyu Yao
xinyu_yao@gwu.edu

ABSTRACT

Wheat head density estimation is an appealing trait for plant breeders. Current manual counting is tedious and inefficient. In this study three types of CNN architectures were investigated: (i) YOLO, (ii) EfficientDet, and (iii) Faster R-CNN. Models were evaluated on the mean average precision at different intersections over union (IoU) thresholds, ranging from 0.5 to 0.75 with a step size of 0.05. Further, the number of wheat heads detected from the RGB images will be compared with the number of wheat heads labelled. YOLO got the best performance with $AP_{.5...0.75}$ of 0.75, mAP@0.5 of 0.86, and rMSE of 10%.

Keywords: Wheat head; Object detection; Object counting; Field imaging; Convolutional neural networks

1 INTRODUCTION

Wheat head density is associated with crop yield, but is a difficult and tedious trait for breeders to efficiently measure. Further, it is prone to sampling errors when the sampling area is small due to limited human resources. Computer vision approaches provide a potential solution to increase the throughput as well as the spatial representativeness, leading potentially to an improved accuracy. A number of studies based on high spatial resolution imaging systems applied to plant phenotyping under field conditions have received much attention in recent years (Li et al., 2014).

While previous studies (Milioto et al., 2018; Ubbens et al., 2018; Madec et al., 2019) have tested wheat head detection methods on individual datasets, in practice these deep learning models are difficult to scale to real-life phenotyping platforms, since they are trained on limited datasets, with expected difficulties when extrapolating to new situations. Most training datasets are limited in terms of genotypes, geographic areas and observational conditions. Wheat head morphology may significantly differ between genotypes with notable variation in head morphology, including size, inclination, color and the presence of awns (David et al., 2020). The appearance of heads and the background canopy also change significantly from emergence to maturation due to ripening and senescence (Anderegg et al., 2020). Further, planting densities and patterns vary globally across different cropping systems and environments, with possible overlap between heads for the higher head densities (David et al., 2020). To fill the need for a large and diverse wheat head dataset with consistent labelling, the Global Wheat Head Detection (GWHD) dataset (David et al., 2020) was published, which can be used to benchmark methods proposed in the computer vision

community. The GWHD dataset results from the harmonization of several datasets coming from seven countries and three continents, at different growth stages with a wide range of genotypes.

The advances in computation capacity along with the availability of very large collections of labelled images have fostered enhanced machine learning methods based on convolutional neural networks (CNNs) in the field of computer vision (LeCun et al., 2015). Existing object detectors are mostly categorized by whether they have a region-of interest proposal step (two-stage) (Girshick et al., 2014; Ren et al., 2015) or not (one-stage) (Tan et al., 2020; Bochkovskiy et al., 2020). R-CNN and Faster R-CNN have been demonstrated to be efficient for detection and analysis of wheat head for certain genotypes, geographic areas and observational conditions (Hasan et al., 2018; Madec et al., 2019). While two-stage detectors tend to be more flexible and more accurate, one-stage detectors are often considered to be simpler and more efficient by leveraging predefined anchors (Huang et al., 2017). Yolo and EfficientDet have achieved state-of-art average precision (AP) on object detection.

The main objective of this study is to evaluate deep learning approaches for high-throughput wheat head counting under field conditions. For this purpose, three types of CNN architectures were investigated: (i) YOLOv5, (ii) EfficientDet, and (iii) Faster R-CNN. Models were evaluated on the mean average precision at different intersections over union (IoU) thresholds, ranging from 0.5 to 0.75 with a step size of 0.05. Further, the number of wheat heads detected from the RGB images will be compared with the number of wheat heads labelled.

2 DATASET AND EXPLORATORY ANALYSIS

2.1 Dataset

All images share a common format of 1024*1024 px with a resolution of 0.1-0.3mm per pixel. The training dataset corresponds to 3422 images from Europe and North America representing 73 percent of the whole GWHD dataset images. To evaluate model performance, including robustness against unseen images, the test data set includes all the images from Australia, Japan, and China, representing 1276 images.

The field in China, where the images are collected, has higher sowing density (seeds*m-2), therefore a data augmentation method: mixup is considered to train models with images of higher wheat head density. However, the fields in Japan and Australia, where the images are collected, have relatively lower sowing density (seeds*m-2), therefore CoarseDropout is considered to train models with images of lower wheat head density. This dataset is diverse in terms of developmental stages when the images were collected, from flowering to ripening, which resulted in the color of wheat heads and canopy ranging from green to yellow. Therefore, randomly changing hue, saturation, and value (HSV) of training images was considered to train models with images of various colors of wheat heads and canopy. Although all images were acquired at nadir-viewing direction, half the field of view along the image diagonal varies from 10° to 46°. Some geometric distortions may be observed for few sub-datasets due to the different lens characteristics of the cameras used, excerpts of the acquired images have different. Images collected from Japan and Switzerland are particularly affected by this issue. Therefore, PiecewiseAffine, HorizontalFlip, VerticalFlip, and RandomRotate90 were considered to train models with images of different perspectives.

2.2 Exploratory Analysis

2.2.1 Distributions of brightness and contrast of training images

RGB images are read using cv2 and then converted to grayscale, where the mean and standard deviation of all the pixels is calculated as the brightness and contrast of the image. The distributions of brightness

and contrast are relatively broad, so randomly changing brightness and contrast of input images were considered for training image augmentation (Fig. 1).

2.2.2 Distribution of Intersection over Union (IoU) of labelled bounding boxes

Intersection over Union (IoU) of labelled bounding boxes is calculated as:

$$IoU(A, B) = (A \cap B)/(A \cup B) \tag{1}$$

There are a total of 138831 labelled bounding boxes, of which over half have overlap with at least one bounding box (Fig. 2). This high occurrence of overlapping and occluded objects is unique to the GWHD Dataset and makes detection more challenging.

3 METHOD

This research framework consists of five consecutive steps: (i) splitting training and validation sample sets, (ii) image augmentation, (iii) training detection models, and (iv) model evaluation.

The specific algorithm (workflow) is as follows:

Step 1. Images in the training dataset were splitted into 5 folders, 4 folders were used for training models and 1 folder was used for model evaluation;

Step 2. HorizontalFlip, VerticalFlip, RandomRotate90, RandomBrightnessContrast, HueSaturationValue, Blur, PeicewiseAffine, CoarseDropout, and MixUp were performed while loading images to model training;

Step 3. YOLO, EfficientDet and Faster R-CNN were implemented using Pytorch and trained on NVIDIA Tesla P100;

Step 4. Models were evaluated on the mean average precision at different IoU threshold, ranging from 0.5 to 0.75 with a step size of 0.05, as well as root mean squared error (RMSE) and the relative RMSE (rRMSE).

3.1 Image Augmentation

Though the GWHD dataset results from the harmonization of several datasets coming from seven countries and three continents, at different growth stages with a wide range of genotypes, there are still limitations in terms of genotypes, geographic areas and observational conditions compared to the worldwide situation. Therefore, heavy augmentations from Albumentations library were performed while loading images to model training.

3.1.1 HorizontalFlip, VerticalFlip, and RandomRotate90

HorizontalFlip, VerticalFlip, and RandomRotate90 (Fig. 3) were performed on 50% of training images in each epoch to train models with images of different perspectives. Although all images were acquired at nadir-viewing direction, half the field of view along the image diagonal varies from 10° to 46°. Some geometric distortions may be observed for few sub-datasets due to the different lens characteristics of the cameras used, excerpts of the acquired images have different. Images collected from Japan and Switzerland are particularly affected by this issue. Therefore, HorizontalFlip, VerticalFlip, and RandomRotate90 were performed.

3.1.2 RandomBrightnessContrast

RandomBrightnessContrast (brightness_limit=0.4, contrast_limit=0.85) (Fig. 4) was performed on 90% of training images in each epoch. The distributions of brightness and contrast are relatively broad (Fig.

1), so randomly changing brightness and contrast of input images were considered for training image augmentation.

3.1.3 HueSaturationValue

HueSaturationValue (hue_shift_limit=0.4, sat_shift_limit=0.4, val_shift_limit=0.4) (Fig. 5) was performed on 90% of training images in each epoch. This dataset is diverse in terms of developmental stages when the images were collected, from flowering to ripening, which resulted in the color of wheat heads and canopy ranging from green to yellow. Therefore, randomly changing hue, saturation, and value (HSV) of training images was considered to train models with images of various colors of wheat heads and canopy.

3.1.4 CoarseDropout

CoarseDropout (max_holes=8, max_height=8, max_width=8, min_holes=None, min_height=None, min_width=None, fill_value=1, always_apply=False) (Fig. 6) was performed on 50% of training images in each epoch. The fields in Japan and Australia, where the images are collected, have relatively lower sowing density (seeds*m-2), therefore CoarseDropout is considered to train models with images of lower wheat head density.

3.1.5 Mixup

Mixup combines pairs of images and their labelled bounding boxes and feeds to the model training25. The original image and a randomly selected image from the training dataset are added pixel-wise to generate the mixup image (Fig. 7), where labelled bounding boxes are also augmented. The field in China, where the images are collected, has higher sowing density (seeds*m-2), therefore mixup is considered to train models with images of higher wheat head density.

3.1.6 PiecewiseAffine

Apply affine transformations that differ between local neighbourhoods. This augmenter places a regular grid of points on an image and randomly moves the neighbourhood of these points around via affine transformations, which leads to local distortions (Fig. 8). Some geometric distortions have been observed in images collected from Japan and Switzerland due to the field of view along the image diagonal varying from 10° to 46°. Therefore, PiecewiseAffine was considered to train models with images containing geometric distortions.

3.1.7 Blur

As images are taken from the field under natural conditions, motion or wind could cause possible blurring, therefore, Blur is performed on training image dataset (Fig. 9) to increase the robustness of the models.

3.2 Training YOLO

The open-source repository created by Ultralytics was cloned from YOLOv5, and requirements.txt dependencies are installed, including Python¿=3.8 and PyTorch¿=1.6. Dataset.yaml was then created, which defines: a path to a directory of training images (or path to a *.txt file with a list of training images); the same for our validation images; the number of classes; a list of class names. In addition to image augmentation mentioned above, YOLOv5 training pipeline employs some other augmentations, such as CutMix, Mosaic, etc. YOLOv5x was trained on images sizes of 1024*1024, batch size of 2, with APEX for 100 epochs. Learning rate was set as 0.01, with momentum of 0.937, and weight_decay of 0.0005. The model with the highest AP on validation dataset was saved and evaluated on test dataset.

3.3 Training EfficientDet-D5

EfficientDet was trained on images sizes of both 512*512 and 1024*1024. Random search was performed for hyperparameter tuning. Learning rate was randomly selected from loguniform (0.001, 0.00001), learning rate optimizer of Adam, AdamW, and SGD was randomly selected, and learning rate scheduler of OneCycleLR, ReduceLROnPlateau, and CosineAnnelingLR was randomly selected. The best

combination of hyperparameter is learning rate of 0.0002, with optimizer of AdamW, and optimizer of ReduceLROnPlateau. Model was trained for 40 epochs, and the model with the highest AP on validation dataset was saved and evaluated on the test dataset.

3.4 Training Faster R-CNN

A Faster R-CNN constructs a single, unified model composed of RPN (region proposal network) and Fast R-CNN with shared convolutional feature layers. Previous algorithms (R-CNN and Fast R-CNN) use selective search to find out the region proposals. Selective search performs poorly in comparison to Faster R-CNN. Instead of using a selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals.

Faster-RCNN Workflow

1. Image Input - Images are passed through a pre-trained CNN up until an intermediate layer (Transfer Learning)
2. Feature Extractor - A Region Proposal Network (RPN) uses the features that the CNN computed to find up to a predefined number of regions (bounding boxes), which may contain objects
3. Region of Interest (RoI) Pooling - Extract previous features which would correspond to the relevant objects into a new tensor
4. R-CNN module - Classifies the content in the bounding box (or discard it, using "background" as a label) and adjusts the bounding box coordinates (so it better fits the object)

Faster R-CNN was trained on image sizes of both 512*512 and 1024*1024. Stochastic Gradient Descent (SGD) vs Adam SGD is an optimization algorithm that is used to update network weights iteratively based on training data. Adam is an optimization algorithm that can be used instead of the classic. The Adam optimizer performed better than the SGD optimizer during a smaller period of time. The following parameters were used for SGD within the PyTorch function: learning rate=0.0001, momentum=0.0005, weight decay=0.0005, nesterov=True The following parameters were used for Adam within the PyTorch function: learning rate=0.0001, betas=(0.9, 0.999), epsilon=1e-08, weight_decay=0.0005, amsgrad=False

	SGD		Adam	
Epoch	Loss	Precision	Loss	Precision
0	.6842	.6703	.7415	.6110
4	.6768	.6721	.6513	.6589
9	.6700	.6720	.6326	.6913

3.5 Criteria to Evaluation Algorithm

Models are evaluated on the mean average precision at different IoU thresholds. The IoU of a set of predicted bounding boxes and ground truth bounding boxes is calculated as:

$$IoU(A, B) = (A \cap B)/(A \cup B) \tag{2}$$

The metric sweeps over a range of IoU thresholds, at each point calculating an average precision value. The threshold values range from 0.5 to 0.75 with a step size of 0.05. In other words, at a threshold of 0.5, a predicted object is considered a "hit" if its intersection over union with a ground truth object is greater than 0.5. At each threshold value t, a precision value is calculated based on the number of true positives (TP),

false negatives (FN), and false positives (FP) resulting from comparing the predicted object to all ground truth objects:

$$\frac{TP(t)}{TP(t) + FP(t) + FN(t)} \tag{3}$$

A true positive is counted when a single predicted object matches a ground truth object with an IoU above the threshold. A false positive indicates a predicted object had no associated ground truth object. A false negative indicates a ground truth object had no associated predicted object. If there are no ground truth objects at all for a given image, ANY number of predictions (false positives) will result in the image receiving a score of zero, and being included in the mean average precision. The average precision of a single image is calculated as the mean of the above precision values at each IoU threshold:

$$\frac{1}{|thresholds|} \sum_{t} \frac{TP(t)}{TP(t) + FP(t) + FN(t)} \tag{4}$$

Bounding boxes will be evaluated in order of their confidence levels in the above process. This means that bounding boxes with higher confidence will be checked first for matches against solutions, which determines what boxes are considered true and false positives. Lastly, the score returned by the competition metric is the mean taken over the individual average precisions of each image in the test dataset.

The head counting performances were also quantified using root mean squared error (RMSE) and the relative RMSE (rRMSE):

$$RMSE = \sqrt{\frac{1}{N}\Sigma_{k=1}^{n}(t_k - c_k)^2} \tag{5}$$

$$rRMSE = \sqrt{\frac{1}{N}\Sigma_{k=1}^{n}\left(\frac{t_k - c_k}{t_k}\right)^2} \tag{6}$$

Where N denotes the number of test images, *tk* and *ck* are respectively the reference and estimated counts for image k.

4 RESULTS

4.1 Results of YOLO

On the validation dataset, the model got $AP_{50...75}$ of 0.75, mAP@0.5 of 0.86, MSE of 4.33, and rMSE of 10.3%. On the test dataset, the model got $AP_{50...75}$ of 0.70. The model performed well on detecting wheat heads (Fig. 10). The head count estimated with model YOLO was in relatively good agreement with the head count labelled with R2 of 0.96 (Fig. 11).

4.2 Results of EfficientDet

EfficientDet trained on images sizes of 512*512 had better performance than trained on images sizes of 1024*1024. On the validation dataset, the model got $AP_{.5....75}$ of 0.72, mAP@0.5 of 0.81, MSE of 3.44, and rMSE of 10.6%. On the test dataset, the model got $AP_{.5....75}$ of 0.63. The head count estimated with model YOLO was in relatively good agreement with the head count labelled with R2 of 0.97 (Fig. 13).

4.3 Results of Faster R-CNN

Faster R-CNN results after a condensed period of time (10 epochs) with the following parameters:

Optimizer	Learning Rate	Momentum	Weight Decay	Betas	Epsilon	Precision
SGD	.001	.005	.0005			.672453
SGD	.0001	.95	.0005			.672577
SGD	.0001	.0005	.0005			.672613
Adam	.0001		.0005	(0.9, 0.999)	1e-08	.718781
Adam	.001		.0005	(0.9, 0.999)	1e-08	.327536

Experimenting on a longer time (50 epochs) for both SGD and Adam resulted in the following:

Optimizer	Learning Rate	Momentum	Weight Decay	Betas	Epsilon	Precision
SGD	.0001	.005	.0005			.748047
Adam	.0001		.0005	(0.9, 0.999)	1e-08	.761707

5 DISCUSSION

5.1 Image acquisition recommendations

Near nadir viewing directions that limit the overlap between heads are recommended (David et al., 2020), especially in the case of high-density head population1. However, image plots from an oblique perspective as opposed to the more common nadir perspective were recommended by other researchers (Hasan et al., 2018). In an oblique view a significant number of spike features such as texture, color, shape etc. can be discerned easily. These features can be more readily extracted for the purposes of various plant phenotyping applications such as spike counting, spike shape measurement, spike texture, disease detection, grain yield estimation etc. We would recommend training CNN models on images dataset with both nadir and oblique views, so the models could have robust performance in different situations and unseen fields and images.

5.2 Models trained on different sizes of images

YOLO and Faster R-CNN trained on images sizes of 1024*1024 got better performance compared to being trained on images sizes of 512*512, which agrees with other research (Eggert et al., 2017), that Faster-RCNN has difficulties with small objects. EfficientDet got better performance when trained on images sizes of 512*512.

5.3 SGD and Adam Optimizer Comparisons

The results after 10 epochs find that lowering the learning rate of the SGD optimizer did not alter the validation precision in a significant way. Changing the momentum of the SGD as well did not alter the results in a significant way. However, using the different optimizer, Adam, is where the differences started to be realized. Using a learning rate of .0001, the precision using the Adam optimizer is significantly higher than the SGD optimizer. Even though the alteration of the learning rate for SGD yielded similar precision results, this was not the same for altering the learning rate of the Adam optimizer. Increasing the learning rate by a thousandth resulted in a significant drop in performance.

Even though performance on Adam was still better after 50 epochs, which took a little over 2 hours, the SGD seemed to start to catch up to Adam. The learning rates continued to be the same. The idea behind this experiment was to keep the similar parameters where possible and to observe the final precision.

A possible conclusion to the reason the SGD seems to perform similar to the Adam optimizer overtime is due to the fact that the nesterov parameter is set to True. Nesterov Momentum (also called Nesterov Accelerated Gradient/NAG) are slight variations of normal gradient descent that can speed up training and improve convergence. This hypothesis was tested by setting nesterov to False and a precision of .731011 was observed.

6 CONCLUSION

In this study three types of CNN architectures were investigated: (i) YOLO, (ii) EfficientDet, and (iii) Faster R-CNN. Models were evaluated on the mean average precision at different intersections over union (IoU) thresholds, ranging from 0.5 to 0.75 with a step size of 0.05. Further, the number of wheat heads detected from the RGB images will be compared with the number of wheat heads labelled. YOLO got the best performance with AP of 0.75, mAP@0.5 of 0.86, and rMSE of 10%.

CONFLICT OF INTEREST STATEMENT

The research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

AUTHOR CONTRIBUTIONS

XY and SP collaborated on image augmentation and drafting the article. XY trained YOLO and EfficientDet, and SP trained Faster R-CNN. All authors gave final approval for publication.

ACKNOWLEDGMENTS

We thank very much Dr. Jafari for his kind support.

DATA AVAILABILITY STATEMENT

The scripts for this study can be found in the GitHub.

REFERENCES

Anderegg, J., Yu, K., Aasen, H., Walter, A., Liebisch, F., and Hund, A. (2020). Spectral vegetation indices to track senescence dynamics in diverse wheat germplasm. *Frontiers in Plant Science* 10, 1749

Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*

David, E., Madec, S., Sadeghi-Tehran, P., Aasen, H., Zheng, B., Liu, S., et al. (2020). 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. *arXiv preprint arXiv:2005.02162*

Eggert, C., Brehm, S., Winschel, A., Zecha, D., and Lienhart, R. (2017). A closer look: Small object detection in faster r-cnn. In *2017 IEEE international conference on multimedia and expo (ICME)* (IEEE), 421–426

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 580–587

Hasan, M. M., Chopin, J. P., Laga, H., and Miklavcic, S. J. (2018). Detection and analysis of wheat spikes using convolutional neural networks. *Plant Methods* 14, 100

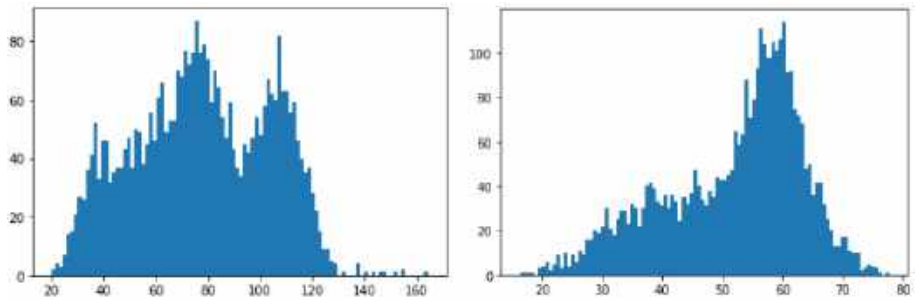

Figure 1. Distribution of brightness **(left)** and contrast **(right)** of training images.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., et al. (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7310–7311

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature* 521, 436–444

Li, L., Zhang, Q., and Huang, D. (2014). A review of imaging techniques for plant phenotyping. *Sensors* 14, 20078–20111

Madec, S., Jin, X., Lu, H., De Solan, B., Liu, S., Duyme, F., et al. (2019). Ear density estimation from high resolution rgb imagery using deep learning technique. *Agricultural and forest meteorology* 264, 225–234

Milioto, A., Lottes, P., and Stachniss, C. (2018). Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in cnns. In *2018 IEEE international conference on robotics and automation (ICRA)* (IEEE), 2229–2235

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*. 91–99

Tan, M., Pang, R., and Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10781–10790

Ubbens, J., Cieslak, M., Prusinkiewicz, P., and Stavness, I. (2018). The use of plant models in deep learning: an application to leaf counting in rosette plants. *Plant methods* 14, 6

FIGURE CAPTIONS

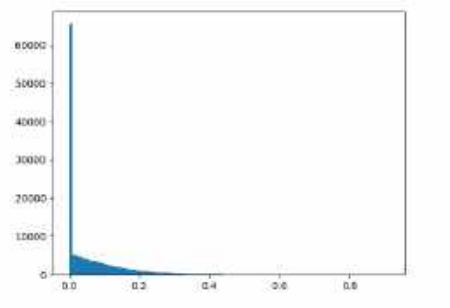

Figure 2. Distribution of IoU of labelled bounding boxes.

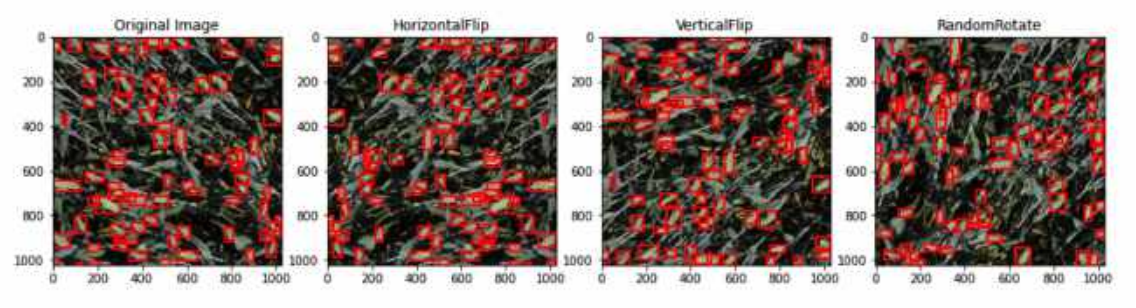

Figure 3. Image augmentation of HorizontalFlip, VerticalFlip, and RandomRotate

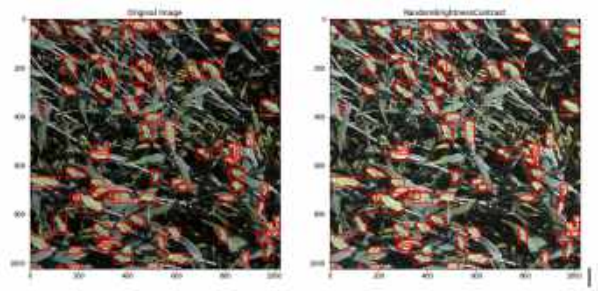

Figure 4. Image augmentation of RandomBrightnessContrast

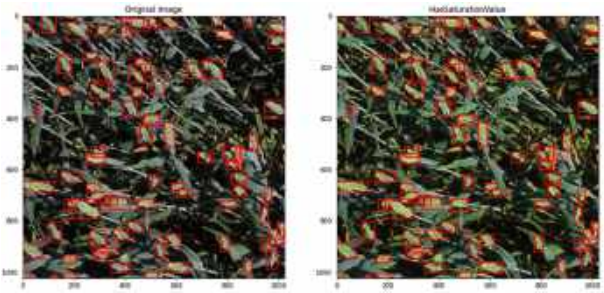

Figure 5. Image augmentation of HueSaturationValue

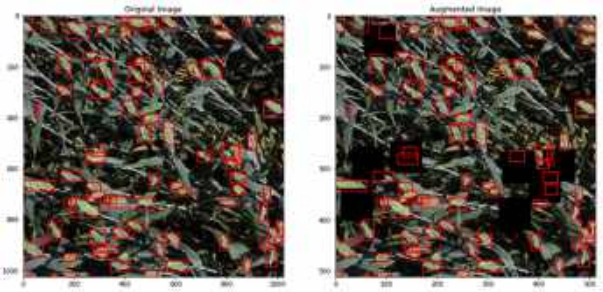

Figure 6. Image augmentation of CoarseDropout

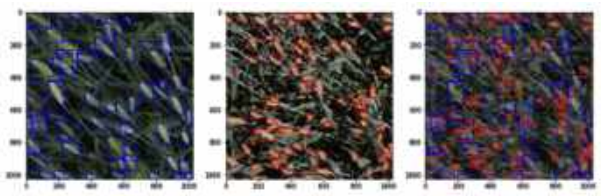

Figure 7. Image augmentation of Mixup, using random selected image **(left)** and original image **(middle)** to generate the augmented image **(right)**.

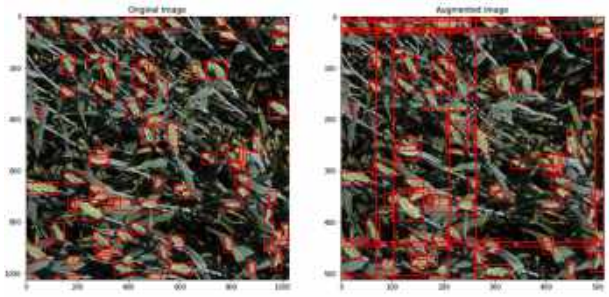

Figure 8. Image augmentation of PiecewiseAffine

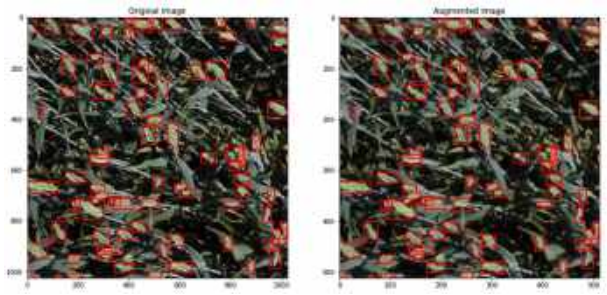

Figure 9. Image augmentation of Blur

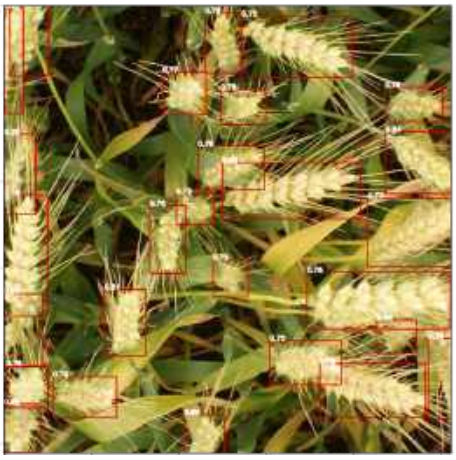

Figure 10. An example of output image using YOLO.

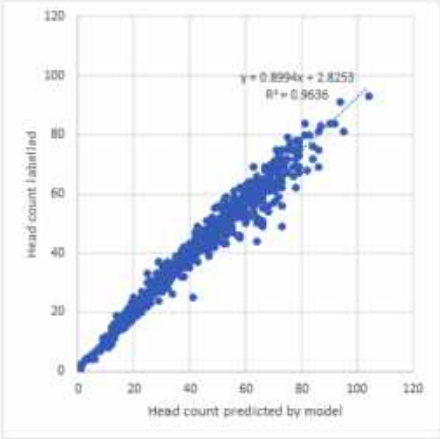

Figure 11. Comparison between the number of head count in each image visually labelling and that detected using YOLO.

Figure 12. An example of output image using EfficientDet.

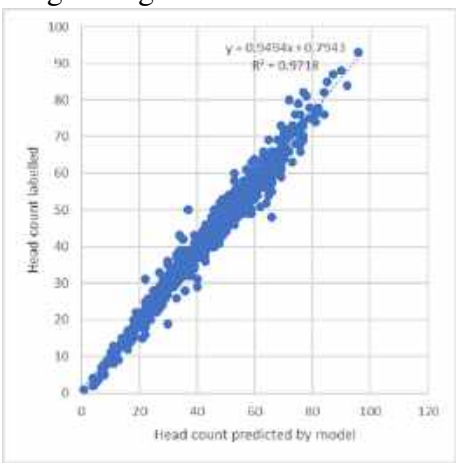

Figure 13. Comparison between the number of head count in each image visually labelling and that detected using EfficientDet.