Wheat-head Detection from Outdoor Wheat Field Images using YOLOv5

Samadur Khan and Ayatullah Faruk Mollah

Department of Computer Science and Engineering, Aliah University,
Kolkata 700160, India
samadur7726@gmail.com

Abstract. Automatic wheat head detection is an important problem having applications in wheat production estimation, wheat breeding, crop management, etc. Since the introduction of popular object detection model YOLO (You Only Look Once) in 2015, a number of advancements have come into picture. In this paper, YOLOv5, the latest of the YOLO family of models is deployed for automatic wheat head detection from outdoor wheat field images. Experimenting on a publicly available wheat head dataset, mean average precision of 92.5% is achieved, which reflects its capability in learning and prediction wheat heads. The present method also outperforms some other methods on the same dataset. YOLOv5, being open source may receive more commits in future paving the possibility of further improvement in wheat head detection performance.

Keywords: Deep learning, Wheat head detection, GWHD dataset, YOLOv5

1. Introduction

Wheat is one of the most planted crops globally and India's prime-most harvest crop, placed second only to rice. Wheat is mainly produced for consumption and it has high nutritional value. At the time of Green Revolution in 1960s, India's wheat crop production and productivity increased at a very high level. In the current scenario, 87 million tons of wheat is produced by India which is the second largest producer after China [1]. Wheat-head detection from outdoor field images can be visually tough because of two major problems - overlap of dense wheat plants and blur effect of wind on the images. These problems make it difficult to detect a single wheat-head. Even the appearance of wheat-head differs due to its color, genotype, maturity, and head orientation. On the contrary, automatic wheat-head detection may facilitate not only wheat-head identification but also wheat-production estimation, wheat breeders, crop management, etc.

Until recently, methods involving image processing technology and shallow learning were mainly applied for detection of wheat heads. Zhu et al. [2] proposed a coarse-to-fine wheat-head detection method with two steps. Binarization approach is followed to identify wheat-heads from image background by Dammer et al. [3]. Bi et al. [4] proposed an approach to detect wheat-spike characteristics by utilizing geometric

properties with around 80% accuracy. On the other hand, deep learning emphasizes on the structure of detection models with high number of layers and highlights the importance of feature learning [5]. With the development of deep learning theory and improvement in hardware performance, deep learning has emerged as the most efficient and advanced approach to computer vision applications.

Wheat-head detection may be thought of as object detection as well. Therefore, deep learning based object detection frameworks can be applied in detecting wheat-heads from images or video frames. In this paper, a popular and fast object detection model known as YOLOv5 is applied, wherein, this model is custom-trained on the training set of wheat-head images and wheat-heads are detected from the test set of images by the trained model. It is noticed that the developed model is able to identify wheat-heads from images containing plenty of wheat heads from all over the world, even if they are significantly overlapped. Experiments on the global-wheat-detection dataset [6] reflect that the developed model detects wheat-heads accurately and quickly.

2. Related Work and Motivation

In this section, some popular deep learning methods used in object detection are discussed and then the progress in wheat-head detection is also reported. Deep learning is being widely used for object detection since the last few years. The algorithms and approaches proposed in this field may be broadly divided into two categories: "two-stage detection" and "one-stage detection".

2.1 Two-stage Detection

The representatives of two-stage detectors include R-CNN [7], Fast R-CNN [8], Faster R-CNN [9], FCOS [10], etc. R-CNN applies deep learning to the field of object detection, laying a foundation for two-stage target detection. In the selection of region proposals, R-CNN uses the selective-search algorithm [11]. In the classification stage, the SVM algorithm is applied. Girshick et al. [8] proposed the Fast R-CNN on the basis of R-CNN, and its innovation was the elimination of the need to send all candidate boxes into the convolutional neural network. However, this method only needed to send one picture to the network. According to the previous experience, Ren et al. [9] proposed a two-stage object detection method, Faster R-CNN, with faster detection speed. This method introduces the region proposal network, which extracts candidate-bound boxes by setting anchor boxes with different proportions and realizes an end-to-end network.

2.2 One-stage Detection

One-stage detectors include mainly YOLO family models (YOLO [12], YOLO9000 [13], YOLOv3 [14], YOLOv4 [15] and YOLOv5 [16]). The main advantage of one-stage detection is accuracy as well as high speed. The basic idea of YOLO is to divide input image into an $S \times S$ grid, and the grid generates a certain number of bounding boxes when the center of an object falls into it. Finally, the objects are framed and

classified by using non-maximum suppression to select the appropriate prediction bounding boxes.

According to the previous research of YOLO, Redmon et al. [13] proposed a novel method, YOLO9000. The main innovation of YOLO9000 is the use of multiple computer vision techniques, such as batch normalization, high-resolution classifier, location, prediction, etc. The detection accuracy of YOLO9000 is 78.6% on the VOC2007 dataset. The backbone of YOLO9000 is Darknet-19 with a 3×3 convolutional kernel and global average pooling, which reduces the computational complexity and parameters of the model. Redmon et al. [14] put forward YOLOv3, which used the Darknet-53 as the backbone network, and introduced the Feature Pyramid Network [17] to achieve the purpose of multiscale integration.

YOLOv4 was proposed by Alexey et al. [15]. The original intention of YOLOv4 was to optimize parallel computing and improve the speed of object detection. A deep neural-network object detector is composed of three parts: backbones, neck, and heads. The function of each part is different. The backbone part mainly extracts the features. The neck is used to fuse the features extracted from the main part. The role of the head is to predict, including predicting the bounding boxes and the object classification. The backbone network part of YOLOv4 applies CSPDarknet53 [18].

2.3 Progress and Motivation

As evident from above discussion, one stage detection is more powerful in terms of speed and accuracy. There are only a few related works on global wheat-head detection on public datasets such as Global Wheat-Head Dataset (GWHD). Like, it was trained on YOLOv3, YOLOv4, Faster R-CNN etc. Although, they all performed moderately well, there were some limitations. Their accuracies were not satisfactory for real-life scenario. In this paper, we apply a later version of YOLO family of models i.e. YOLOv5 to get it custom-trained for detection of wheat-heads from outdoor wheat field images.

3. Methodology

YOLOv5 [18], the latest version of YOLO family of models, supports custom training. It was introduced by Mr. Glenn Jocher (Founder & CEO of Utralytics) and it is one of the best available models for object detection at this moment. It is the first YOLO model to be implemented in Pytorch entirely. In comparison to its previous version i.e. YOLOv4, it has slightly less performance in COCO dataset, but it is superior in respect of ease of use, exportability, memory requirements, speed, good Mean Average Precision (MAP) and market size. As shown in Fig. 1, the network architecture of YOLOv5 consists of three parts: (1) Backbone: CSPDarknet, (2) Neck: PANet, and (3) Head: YOLO Layer.

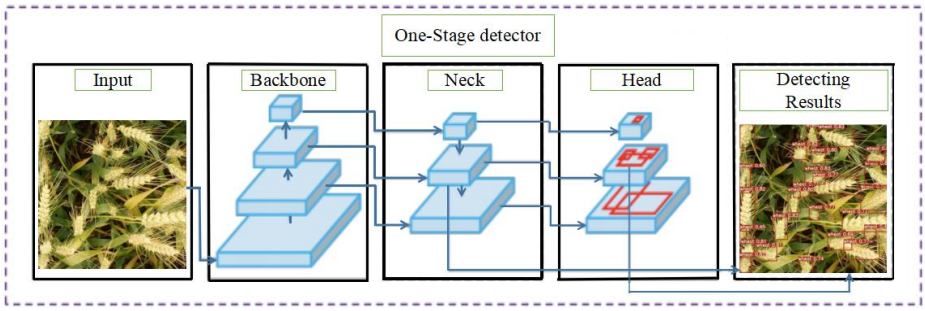


Fig. 1 YOLOv5 architecture for wheat-head detection (The backbone is used for feature extraction, the neck is used for feature fusion, and the head is used for detection).

3.1 Data Pre-processing

Now, for the training purpose, the whole data should be prepared as per requirement of YOLO. Data labeling, splitting of dataset and creation of the yaml file take place in this stage. YOLO expects one *.txt file per image having the following specifications (also demonstrated in Fig. 2).

- One row per object
- Each row likes **Class x_center y_center width height** format.
- Box coordinates must be in normalized xywh format (from 0 to 1). If our boxes are in pixels, divide x_center and width by image width, and y_center and height by image height.
- Class numbers are zero-indexed (start from 0).

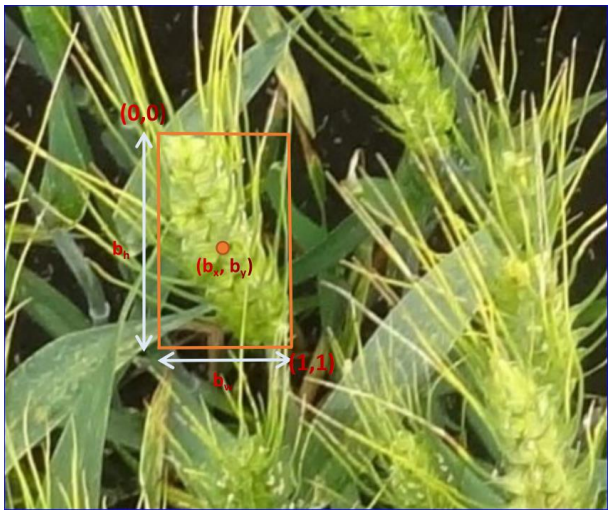


Fig. 2 YOLO formatted bounding box of an object (Wheat-head)

After labeling the images, we have to create a yaml file which contains

a) The path for training and validation images
b) Number of classes present
c) The names corresponding to those classes.

3.2 Training of Model

In order to train this model with a custom dataset, we have to take care of some specific things. Firstly, to prepare the dataset according to the YOLO format, install all the dependencies, setup the data and directories, and the YAML files. After configuring all these setups, we can begin the training process.

There are multiple parameters that can be specified which are:

img: define input image size
batch: determine batch size
epochs: define the number of training epochs.
data: set the path to the YAML file
cfg: specify our model configuration
weights: specify a custom path to weights
name: result names
nosave: only save the final checkpoint
cache: cache images for faster training

For training purpose the command used is

```
python train.py img 1024 --batch 16 --epochs 150 --data
'../data.yaml' --cfg ./models/custom_yolov5s.yaml --weights ''
'' --name yolov5s_results --cache
```

3.3 Running Inference

Using the final trained weights obtained after completion of training, we can use the model for inference. To run the model inference we can use the following command:

- **source:** input images directory or single image path or video path.
- **weights:** trained model path.
- **conf:** confidence threshold.

The command we used in this work is

```
python detect.py --weights
runs/train/yolov5s_results/weights/best.pt --img 416 --conf 0.4
--source ../test/images
```

4. Experiments and Results

Here, basically the Kaggle dataset of global-wheat-detection [6] is used which consists of 3373 images which are recorded from many locations around the world. It contains

images of outdoor wheat fields, with bounding boxes for each wheat head to be identified. Not all images include wheat heads / bounding boxes. The CSV data is simple - the image ID matches up with the filename of a given image, and the width and height of the image are included, along with a bounding box. There is a row in train.csv for each bounding box and not all images have bounding boxes. The samples have been divided in 7:2:1 ratio for training, validation and prediction respectively.

4.1 Experimental Setup

After setting up the model, the next step is to train it. The training is carried out using Google Colab platform for its faster GPU. In this project, maximum 150 epochs are taken for better performance. The libraries used are Pandas, Matplotlib >=3.2.2, Numpy >=1.18.5, Opencv-python >=4.1.2 Tensor -board >=2.2, Torch >=1.7.0, Tqdm >=4.41.0. Finally, testing of the developed model is performed using the test dataset. And then, the system compares the coordinate values of the bounding boxes of the ground truth and the outcome.

4.2 Evaluation Metrics

Intersection over Union (IoU): Intersection over Union (IoU) ratio is used as a threshold for determining whether the predicted output is true positive or false positive. According to this protocol, True Positive (TP) and False Positive (FP) are measured by using an overlap ratio of GT and detected rectangle bounding box [19,20] as shown in Fig. 3. TP is considered if the value of IoU ratio is greater than 0.5 that is defined in below equation.

$$\text{Area}\,(G_i \cap P_j) \,/\, \text{Area}\,(G_i \cup P_j) > 0.5$$

where $G = \{G_1, G_2, G_3, G_4, \ldots\ldots. G_n\}$ is the set of GT boxes and $P = \{P_1, P_2, P_3, P_4, \ldots\ldots. P_n\}$ is the set of predicted boxes, where $1 < i < n$ and $1 < j < m$. If the IoU value is zero, it is considered as no detection.

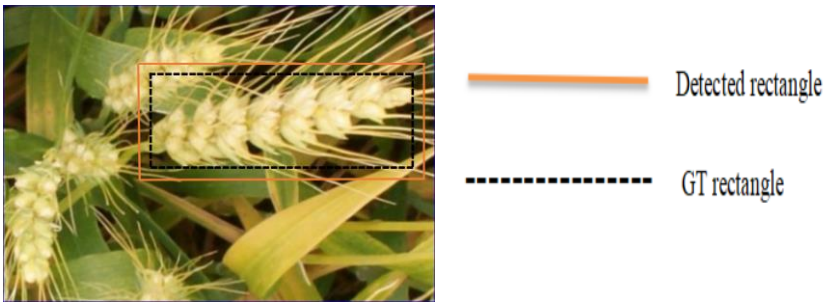


Fig. 3 Comparison of Ground Truth rectangle and detected rectangle

Precision: Precision is the ratio of the number of true positives to the total number of positive prediction. For example, if our model detects 100 objects, and 85 are correct, then precision is 85 percent.

$$\text{Precision} = (\text{True Positive})/(\text{True Positive} + \text{False Positive})$$

Recall: Recall is the ratio of the number of true positives to the total number of actual objects. For example, if our model correctly detects 75 objects in an image, and there are actually 100 objects in the image, the recall is 75 percent.

$$\text{Recall} = (\text{True Positive})/(\text{True Positive} + \text{True Negative})$$

True Positive (TP) means the samples are predicted to be correct and actually positive.

False-Positive (FP) represents that samples are predicted to be positive but actually negative.

And, **False-Negative (FN)** represents those samples that are predicted to be negative but positive actually.

Mean Average Precision (mAP): It means the average accuracy of all categories is added, and divided by the number of categories. Here, we use two range of mAP, **mAP_{50} and mAP_{95}**. **mAP_{50}** represents the average accuracy of the confidence threshold of 50% and **mAP_{95}** represents the average accuracy of the confidence threshold of 50% - 95% .

4.3 Detection Performance

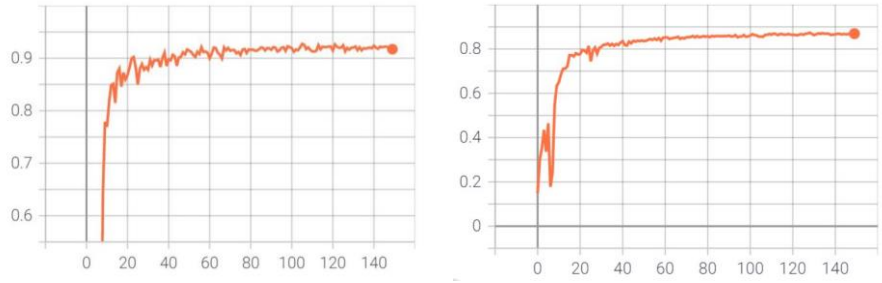
In this section, the results of the developed model are presented. It may be noticed from Table 1 and Fig. 4 that the model detected the wheat head(s) from the images reasonably accurately.

Table 1. Performance of our Model for different epochs

Epochs	Precision (%)	Recall (%)	mAP_{50} (%)	mAP_{95} (%)	Required time to train the model
10	73.1	62.7	67.6	28.5	5 min 51s
50	90.8	84.1	90.6	46	18 min 20s
100	91.4	86.2	91.7	48	36 min 59s
150	91.7	86.9	92.5	48.8	54 min 55s

Fig. 4 Detected Wheat heads from two samples images of the test set.

Here, basically the statistical graphs are shown about the precision and recall of the model with increasing epochs. From Fig. 5, it may be noticed that the model is performing well as the precision and recall values increase over time up to a certain range during the training process.



(a) (b)

Fig. 5 Graphical representation of varying (a) Precision, and (b) Recall with increasing epochs.

4.4 Discussion

Performance of the present wheat head detection cum localization model is compared with some of the report methods on the same wheat head dataset i.e. the GWHD dataset [6] in Table 2. It may be noted that the said comparison is reported in terms of standard metrics, viz. mAP_{50} and mAP_{95} adopted for quantifying wheat head detection performance. In case of mAP with threshold 50, the present method performs very well with 92.5% precision. But, at mAP with threshold 95, our method performs slightly less than YOLOv4.

Table 2. Performance of our Model Comparison with other Methods

Method	Datasets	mAP_{50}	mAP_{95}
YOLOv3 [14]	GWHD	90.5%	47.7%
YOLOv4 [15]	GWHD	91.4%	**51.2%**
Faster R-CNN [9]	GWHD	76.6%	49.1%
Our Method (YOLOv5)	GWHD	**92.5%**	48.8%

Moreover, as reported in literature, YOLOv5 is much faster than YOLOv4 and it is a one-stage object detection model. Thus, YOLOv5 appears to be a prospective trainable model for wheat head detection problems as well.

5. Conclusion

In this paper, YOLOv5, the latest model of YOLO family is deployed on a publicly available wheat head dataset and the accuracy obtained with the trained model is 92.5%. Compared to some other methods that reported results on the same dataset, the present method is found to detect wheat heads effectively and it marginally outperforms other methods as well. Moreover, the built-in Pytorch framework of YOLOv5 is user-friendly and it may receive more contributions in near future as it is an open source model, which in turn may further improve wheat head detection performance.

References

1. Geography and You. 2017. Available online: https://geographyandyou.com/wheat-crop/ (accessed on March 18, 2017)
2. Zhu, Y.; Cao. Z.; Lu, H.; Li, Y.; Xiao, Y. In-field automatic observation of wheat heading stage using computer vision. Biosyst. Eng. 2016, 143, 28-42.
3. Dammer, K.; Möller, B.; Rodemann, B.; Heppner, D. Detection of head blight (Fusarium ssp.) in winter wheat by color and multispectral image analyses. Crop Prot. 2011, 30, 420-428.
4. Bi, K.; Jiang, P.; Li, L.; Shi, B.; Wang, C. Non-destructive measurement of wheat spike characteristics based on morphological image processing. Trans. Chin. Soc. Agric. Eng. 2010, 26, 212-216.
5. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. Nature 2015, 521, 436-444.
6. Global Wheat Head Detection (GWHD) Dateset, Available in Online: https://www.kaggle.com/c/global-wheat-detection/data
7. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate detection and semantic segmentation. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June 2014, pp. 580–587.
8. Girshick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440-1448.
9. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Trans. Pattern Anal. Mach. Intell. 2017, 39, 1137-1149.

10. Tian, Z.; Shen, C.; Chen, H.; He, T. FCOS: Fully Convolutional One-Stage Object Detection. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27-28 October 2019; pp. 9626-9635.
11. Uijlings, J.R.R.; Van De Sande, K.E.A.; Gevers, T.; Smeulders, A.W.M. Selective search for object recognition. Int. J. Comput. Vis. 2013, 104, 154-171.
12. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27-30 June 2016; pp. 779-788.
13. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21-26 July 2017; pp. 6517-6525.
14. Redmon,J.; Farhadi, A. YOLOv3:An Incremental Improvement. arXiv 2018, arXiv:1804.02767 Available online: http://arxiv. org/abs/1804.02767 (accessed on 29 December 2020).
15. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv 2020, arXiv:2004.10934
16. Ultralytics.YOLOv5.2020. Available online: https://github.com/ultralytics/yolov5#readme
17. Lin, T.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21-26 July 2017; pp. 936-944.
18. Wang, C.; Liao, H.M.; Wu, Y.; Chen, P.; Hsieh, J.; Yeh, I. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. In: Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Seattle, WA, USA, 14-19 June 2020; pp. 1571-1580.
19. Khan, T., Sarkar, R. & Mollah, A.F. Deep learning approaches to scene text detection: a comprehensive review. Artif Intell Rev 54, 3239-3298 (2021).
20. Saha, S., Chakraborty, N., Kundu, S., Paul, S., Mollah, A.F., Basu, S., & Sarkar, R. (2020). Multi-lingual scene text detection and language identification. Pattern Recognition Letters, 138, 16-22 (2021).