REAL-TIME WHEAT DETECTION BASED ON LIGHTWEIGHT DEEP LEARNING NETWORK REPYOLO MODEL
/
基于轻量级深度学习网络 RepYOLO 模型的麦穗实时检测

Zhifang BI[1], Yanwen LI[2], Jiaxiong GUAN[2], Xiaoying ZHANG[*3]
[1] Department of Basic Courses, Shanxi Agricultural University, Shanxi / China
[2] College of Information Science and Engineering, Shanxi Agricultural University, Shanxi / China
[3] School of Software, Shanxi Agricultural University, Shanxi / China
Tel: +86-15803449361; E-mail: xiaoyingzhang@sxau.edu.cn
DOI: https://doi.org/10.35633/inmateh-72-53

Keywords: *Real-time detection; Wheat; YOLOv4; RepVGG; CBAM; ATSS Algorithm*

ABSTRACT

Real-time detection has become an essential component in intelligent agriculture and industry. In this paper, a real-time wheat spike detection method based on the lightweight deep learning network RepYOLO is proposed. Addressing the small and densely packed phenotype characteristics of wheat spikes, the channel attention mechanism module CBAM from the traditional YOLOv4 algorithm is introduced and multiple convolutional kernels are merged using a structural reparameterization method. Additionally, the ATSS algorithm is incorporated to enhance the accuracy of object detection. These approaches significantly reduce the model size, improve the inference speed, and lower the memory access cost. To validate the effectiveness of the model, it is trained and tested on a large dataset of diverse wheat spike images representing various phenotypes. The experimental results demonstrate that the RepYOLO algorithm achieves an average accuracy of 98.42% with a detection speed of 8.2 FPS. On the Jetson Nano platform, the inference speed reaches 34.20 ms. Consequently, the proposed model effectively reduces the memory access cost of deep learning networks without compromising accuracy and successfully improves the utilization of CPU/MCU limited performance.

摘要

实时检测已成为智能农业的重要组成部分。本文提出了一种轻量级深度学习网络 RepYOLO 的小麦穗实时检测方法。针对小麦穗小而密的表型特征，从传统的 YOLOv4 算法中引入通道注意机制模块 CBAM，并使用结构重参数化方法合并多个卷积核。此外，结合了 ATSS 算法来提高目标检测的准确性。这些方法显著减小了模型尺寸，提高了推理速度，降低了内存访问成本。为了验证该模型的有效性，本实验在代表不同表型的小麦穗图像的大型数据集上进行了训练和测试。实验结果表明，RepYOLO 算法的平均准确率为 98.42%，检测速度为 8.2fps。在 Jetson nano 平台上，推理速度达到 34.20/ms。因此，本文提出的模型在不影响精度的前提下，有效地降低了深度学习网络的内存访问成本，并成功地提高了 CPU/MCU 有限性能的利用率。

INTRODUCTION

Drones have become an important technology in agriculture due to their advantages, including low cost, high speed, low-altitude flying capability, and efficient operation *(Radoglou-Grammatikis et al., 2020)*. They are equipped with various sensors such as RGB cameras, thermal imaging cameras, and multispectral cameras, which allow for capturing high-resolution images of wheat fields at different spatial scales, unlike remote sensing images from satellites and ground platforms *(Hassan et al., 2019)*.

Deep learning technology combined with drone detection has become the mainstream detection technology. It enables crop growth monitoring, management efficiency improvement, and precision agriculture. In the field of computer vision, deep learning has made tremendous progress. Chen et al.'s research has shown the successful application of deep learning in this field *(Chen et al., 2021)*. Alsalm et al. developed a computer vision-controlled drone monitoring system that can accurately detect crop health and pests *(Jin et al., 2017)*. Khan et al. proposed a new framework based on drones and object detection technology that can be used for precision agriculture, remote sensing, search, and rescue *(Khan et al., 2022)*.

However, methods relying on cloud computing platforms for detection have latency issues in practical applications and cannot meet real-time requirements *(Khalil et al., 2022)*. Additionally, without support from a local area network and large computing devices, application and implementation are impossible. Directly porting the inference detection program to an embedded unmanned aerial vehicle (UAV) chip is the best way to solve this problem.

However, due to the large amount of computing resources required for deep learning during the inference process, current deep learning technology cannot be applied to embedded UAV chips. Therefore, this paper proposes a target detection model suitable for embedded UAV chips - RepYOLO.

The RepYOLO model greatly reduces MAC (memory access cost) *(Ma et al., 2022)* by changing the backbone network to RepBlock *(Ding et al., 2021)* and simplifies the prediction boundary box calculation process by changing the detection head to anchor-free *(Wang et al., 2021)*. In addition, the CBAM attention mechanism (*Woo et al., 2018*) is added to the Neck network structure to further improve the model's accuracy. The ATSS algorithm *(Zhang et al., 2022)* was chosen in the training process to improve the model's inference accuracy. Because wheat data exhibits multi-phenotype characteristics and fields present complex scenes with dense distribution, overlap, and occlusion, the Global Wheat Dataset *(David et al., 2021)* and real UAV-captured wheat images from fields were used for the dataset.

Finally, the experimental results of this paper show that the RepYOLO model can achieve optimal accuracy and speed on UAV devices when detecting wheat data with complex features, and the detection performance and accuracy are higher than the YOLOv4 *(Bochkovskiy et al., 2020)* model directly ported to the UAV chip. Since the proposed model in this paper is universal, it is also suitable for detecting other targets.

MATERIALS AND METHODS

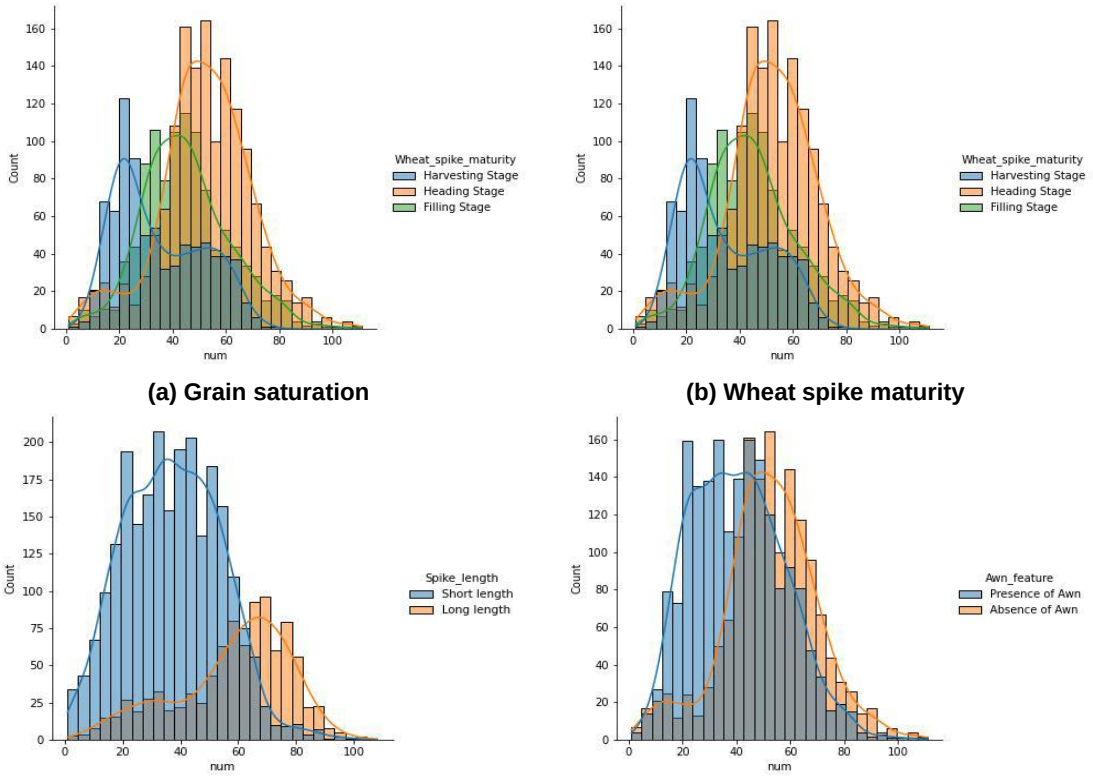
Data Acquisition and Preprocessing

The dataset used in this experiment is based on the Global Wheat Head Detection Dataset. To address the issue of uneven distribution in the sample data, increase the diversity of the dataset and mitigate the risk of trained model errors and biases, both web crawling and field data collection methods were used. With these methods, wheat image data of four phenotypic features were obtained: grain saturation, wheat spike maturity, spike length, and awn features (Fig.1).

Fig. 1 - Four Phenotypic Features

To ensure the statistical properties of the dataset, skewness, kurtosis, standard deviation, and Z-Score are also computed to evaluate whether the sample data conforms to normal distribution. During data preprocessing, multiple measures were taken to improve the convergence speed of training and the quality of the dataset. Firstly, the original images were cropped and adjusted to 416x416 pixels in size to improve the convergence speed of training. In addition, a complex frequency domain Gaussian low-pass filter method was used to eliminate image noise.

In this dataset, wheat image data containing four phenotypic traits were collected and fitted curves were plotted for the quantity distribution of different traits, as shown in Fig.2. The fitted curves reflect the statistical regularities of the quantity distribution of wheat images with different phenotypic traits in the dataset.



(a) Grain saturation **(b) Wheat spike maturity**

(c) Spike length **(d) Awn feature**

Fig. 2 - Distribution of Sample Data under Different Phenotypic Features

The skewness, kurtosis, standard deviation, Z-Score of four phenotypic features are in Table 1. Skewness is used to describe the degree of deviation from a normal distribution. Kurtosis measures the thickness of the distribution's peak compared to a normal distribution. Standard deviation reflects the dispersion of the distribution, and Z-score is primarily used to determine whether the sample data conforms to a normal distribution. When the value of Z-score is between -1.96 and 1.96, 95% of the data is included in this interval, indicating that it conforms to a normal distribution.

Table 1

Kurtosis, Skewness, Standard Deviation and Z-Score of Different Phenotypic Data

Phenotypic features	Grain saturation		Wheat spike maturity		
	Shriveled type	Saturated type	Heading stage	Filling stage	Harvesting stage
Skewness	0.2694	0.2587	-0.1933	0.4382	0.4672
Kurtosis	0.0559	0.0857	0.6251	0.3949	-0.9842
Standard Deviation	0.4732	0.4007	0.5181	0.5201	0.6903
Z_1-Score	0.5694	0.6457	-0.3732	0.8426	0.6767
Z_2-Score	0.1180	0.2138	1.2066	0.4687	-1.4257
Phenotypic features	**Spike length**		**Awn feature**		
	Short length	**Long length**	**Absence of Awn**	**Presence of Awn**	
Skewness	0.2123	-0.6443	-0.1933	0.3841	
Kurtosis	-0.2248	-0.2099	0.6251	-0.2027	
Standard Deviation	0.4523	0.5127	0.5019	0.3140	
Z_1-Score	0.4694	-1.2568	1.2455	1.2232	
Z_2-Score	-0.4789	-0.4094	1.2045	-0.6457	

Figure 2 shows the bimodal analysis distribution fitting curves of the four feature samples. It can be seen from the figure that the data distribution roughly conforms to the normal distribution. Table 1 provides a quantitative analysis of the sample data, including skewness, kurtosis, standard deviation, and Z-Score calculation. The results of the calculation indicate that the sample distribution of the four phenotypic features conforms to the normal distribution, as the values of Z_1-Score and Z_2-Score are both between -1.96 and 1.96.

In order to enhance the generalization ability of deep learning models and avoid overfitting, data augmentation techniques were utilized, including Gaussian blurring, Random noise, Hue, Exposure, Cutout, and Mosiac (Fig. 3. The aforementioned data augmentation techniques were used to process RGB images with a size of 416x416 pixels, generating a total of 11,541 images. These images were randomly shuffled and divided into training, validation, and testing sets with a ratio of 7:2:1. The training set was used to train the model, the validation set was used to tune the hyperparameters, and the testing set was used to evaluate the performance of the model. The specific parameter settings for data augmentation are shown in Table 2.

(a) Original **(b) Gaussian blurring** **(c) Random noise** **(d) Hue**

(e) Exposure **(f) Cutout** **(g) Mosiac** **(h) Combination**

Fig. 3 - Data Enhancement

Table 2

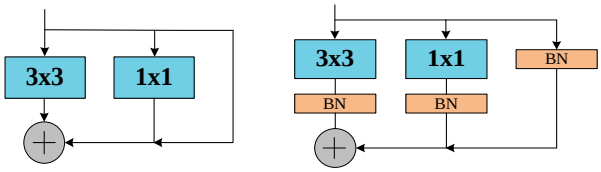
Data Enhancement Setting

Gaussian blurring	Random noise	Hue
Up to 2px	Up to 1% of pixels	-40° ~ +40°
Exposure	**Cutout**	**Mosiac [1]**
-5% ~ +5%	3 boxes with 5% size	Applied

RepYOLO structure

RepYOLO is a detection algorithm that is improved based on YOLOv4. Similar to other methods in the YOLO series, RepYOLO can output detection boxes containing information such as coordinates, categories, and confidence scores. This study mainly focuses on modifying the backbone, head, and neck of the algorithm to maximize the utilization of limited computing resources on drone chips. Structural reparameterization technology can effectively improve the utilization of computer resources (*Ding et al., 2021, Zagoruyko et al., 2017)*. Inspired by the RepVGG classification network, RepBlock was used instead of the original CSPDarknet in the backbone network to increase detection speed. Previous studies have shown that anchor-based detection heads significantly decrease network speed compared to anchor-free detection heads, but this is due to the uneven distribution of positive and negative samples during training. To further improve the utilization of limited computing resources on drones, an anchor-free detection head was used instead of the original one in the head network and the ATSS algorithm was adopted to replace the original positive and negative sample matching scheme. At the same time, the CBAM attention mechanism module was inserted in the neck network to further integrate advanced semantic features and low-level spatial information, and improve the accuracy of the detection model.

Previous research has indicated that multi-branch networks typically exhibit better classification performance than single-path networks, but this can lead to an increase in inference latency (*Szegedy et al., 2019, He et al., 2016*). In this study, inspired by RepVGG, the limited computing performance of unmanned aerial vehicles was considered. To balance accuracy and speed, an efficient and parameterizable backbone network, known as RepBlock, was utilized. The RepBlock network leverages hardware computing capabilities, and after model training, inference latency was significantly reduced by converting the multi-branch topology to a 3x3 convolution layer (RepConv) with a ReLU activation function. The reparameterization method outlined in the RepVGG paper, which separates the multi-branch topology during training from the single-path network during inference, was employed. Previous studies have shown that the Batch Normalization (BN) layer normalizes data within each mini-batch. This normalization ensures stable means and variances, and reduces the coupling between adjacent layers. Consequently, the BN layer makes network training more stable. In addition, the BN layer acts as a regularization technique to prevent overfitting. As shown in Fig.4, the RepBlock Training Structure introduces the BN layer.



(a) Original Multi-branch (b) Multi-branch with BN

Fig. 4 - Introduction of BN Layer

The YOLOv4 algorithm for object detection traditionally uses an anchor-based detection head. To improve detection performance and reduce computation, this paper proposes the RepYOLO algorithm, which utilizes an anchor-free detection head and the ATSS algorithm to define positive and negative training samples. In contrast to the IoU threshold allocation scheme used in the YOLOv4 algorithm, the RepYOLO method can better adapt to objects of different scales and shapes while also effectively reducing computation.

Previous studies have demonstrated that anchor-free object detection yields lower accuracy compared to anchor-based algorithms. However, recent research has highlighted that the fundamental disparity between anchor-based and anchor-free algorithms lies in how they define positive and negative training samples, leading to a performance gap during the training process. If these algorithms implement identical definitions for positive and negative samples during training, regardless of whether it is based on box or point regression, there will be no conspicuous difference in their final performance *(Ge et al., 2021)*. The latest scientific research has proposed the ATSS algorithm for positive and negative sample matching, which successfully addresses the issue of reduced accuracy in anchor-free object detection algorithms, making its accuracy comparable to that of anchor-based algorithms *(Zhang et al., 2020)*.

To compare the performance of anchor-free and anchor based algorithms, the two algorithms were calculated in terms of detection accuracy, computation speed, and the definition of positive and negative samples used in the training process. In the model performance evaluation section, it will be demonstrated, from the perspectives of detection accuracy and speed, that our RepYOLO algorithm outperforms YOLOv4 in both aspects and is more adaptable to different scenarios. This will prove the practicality and effectiveness of our proposed algorithm for unmanned aerial vehicle object detection.

In order to further enhance the feature extraction and fusion abilities of the PAN layer, the CBAM attention mechanism was integrated into PAN in our study. Traditional convolutional neural networks (CNNs) have limitations in capturing channel and spatial information, which can affect their feature representation ability. The Convolutional Block Attention Module (CBAM) selectively emphasizes important features and suppresses unimportant ones, improving the accuracy of feature representation. Compared to other attention mechanisms, CBAM has relatively smaller computational costs while still preserving limited UAV performance. Therefore, the CBAM attention mechanism was introduced into the PAN layer to further optimize the model's performance.

The CBAM module comprises the Channel Attention Module (CAM) and the Spatial Attention Module (SAM). CAM is responsible for learning the weights of channel attention and computing the attention map between channels to enable the network to selectively emphasize informative channels and suppress uninformative ones. On the other hand, SAM learns the weights of spatial attention and computes the attention map between spatial positions, enabling the network to selectively focus on significant spatial regions and suppress irrelevant ones.

The final attention map is obtained by multiplying the attention maps of the two modules. This final attention map is then applied to the original feature map to obtain the ultimate output. The structure of CBAM is illustrated in Fig.5.

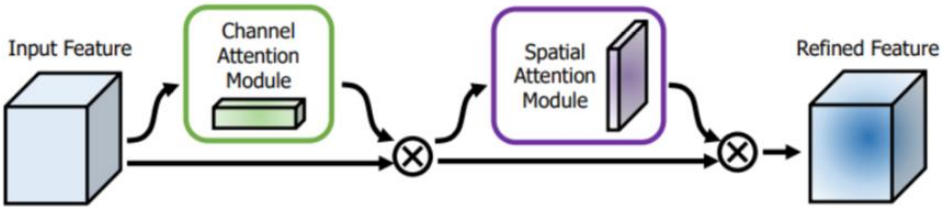


Fig. 5 - The Structure of CBAM

To compare the performance differences between the CBAM-inserted algorithm and the one without CBAM, the focus was on comparing them based on detection accuracy. To further validate the performance differences, the Grad-CAM heatmap visualization method was employed.

Experimental Environment

The experimental training was conducted on a machine equipped with an Intel ® Xeon(R) Platinum 8255C processor and an NVIDIA GeForce RTX 3090 graphics processor (GPU). The operating system used was Ubuntu 20.04. The training framework was CUDAv11.5, cuDNNv8.2.0, and PyTorch 1.6.1.

The training parameters were set as shown in Table 3.

Table 3

Parameters of Training Setting

Optimizer	Momentum	Initial Learning rate	L2 Regularization
SGD/ADMW	0.9	0.01	0.001
Training Epochs	**Warmup Epochs**	T_{max}	**Batch Size**
300	10	290	8

In this study, model transfer experiments were conducted using two unmanned aerial vehicles (UAVs) with different configurations. The first UAV was equipped with a Raspberry Pi 4B processor (ARM Cortex-A72 CPU, 8G Broadcom Video-Core VI DSP (GPU), 8GB LPDDR4-3200 SDRAM RAM), running on Ubuntu 18.04 with the PyTorch v1.7.0 framework. The second UAV was equipped with a Jetson nano processor (ARM Cortex-A57 CPU, 128-core NVIDIA Maxwell GPU, 8GB LPDDR4-3200 SDRAM RAM).

Evaluation indexes of the model

Precision, Recall, F1 value, and mean Average Precision (mAP) were used as evaluative metrics. Precision is the area under the Precision-Recall curve; mAP is the average detection accuracy of the model overall categories of wheat ear targets; and Frames per Second (FPS) is real-time evaluation. The more excellent the value, the better the model's real-time performance.

MAC (Memory Access Cost) and FLOPs (Floating Point Operations) are two metrics used to measure the computational complexity of deep learning models. These metrics have a significant impact on the model inference speed. MAC represents the number of memory accesses when a model performs inference, while FLOPs refer to the total number of floating-point operations during the entire inference process, including matrix multiplication and other mathematical operations.

RESULTS

Grad-CAM is a visualization technique commonly used in convolutional neural networks. It identifies important spatial locations and presents the corresponding attention regions. By analyzing the key focus regions identified by the target detection network, a more intuitive understanding of how the network extracts features and applies them to target detection can be gained.

To evaluate the performance of RepYOLO, its visualization results were compared with the YOLOv4 baseline. Figure 6 displays that the Grad-CAM heatmap mask of RepYOLO more effectively covers the target object region. This indicates that RepYOLO exhibits better detection performance and stronger robustness in wheat with different phenotypic features.

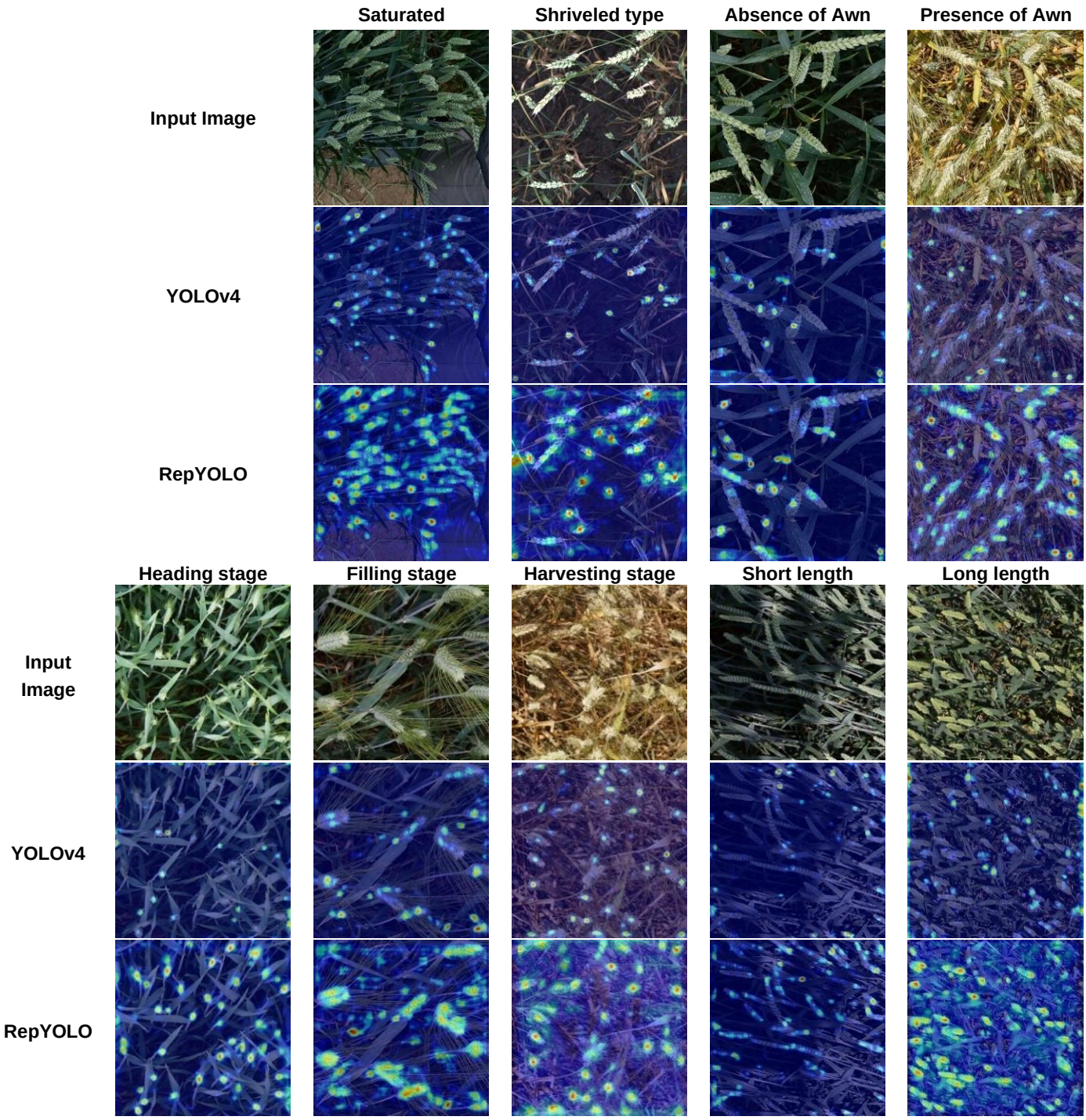


Fig. 6 - Grad-CAM visualization result of different traits

In the analysis of experimental results, Precision was used to evaluate the detection accuracy, while Recall, F1, and mAP were used to evaluate whether the training was sufficient. The small model (with half the standard number of neurons per layer) was represented as "-n", the standard model as "-s", and the large model (with twice the standard number of neurons per layer) as "-l". Through the analysis of the experimental results in Table 4, it was found that the RepYOLO standard model improved by 4.67%, 1.37%, 1.51%, and 1.68% in Precision, Recall, F1, and mAP, respectively, compared to the YOLOv4 standard model. These results suggest that RepYOLO significantly improved in terms of detection accuracy performance, which is consistent with the results of Grad-CAM heatmap visualization.

Based on these findings, it can be speculated that the CBAM feature refinement process in RepYOLO enables the network to better utilize the given features and improve detection performance. In section Ablation Study, the focus is on analyzing the impact of CBAM on the accuracy of RepYOLO. Our goal is to demonstrate the significant improvement in detection performance after integrating CBAM. This section will provide a detailed analysis and exploration of CBAM to verify its effectiveness in RepYOLO.

Table 4

Precision Evaluation Metrics of YOLOv4 and RepYOLO

Base Model	Size	P/%	R/%	F1/%	mAP/%
YOLOv4	-n	91.62	80.69	87.10	80.11
	-s	93.55	82.34	88.88	81.74
	-l	93.89	82.63	89.19	82.03
RepYOLO	-n	96.75	82.45	89.03	82.17
	-s	98.22	83.71	90.39	83.42
	-l	98.71	84.13	90.83	83.84

From the perspective of ShuffleNetV2, the use of FLOPs as an indicator of model speed has limitations. The experimental results reveal that although ShuffleNetV2 has higher FLOPs than MobileNetV2, its MAC is much smaller than MobileNetV2 *(Ma et al., 2018, Sandler et al., 2018)*. Moreover, the inference speed of ShuffleNetV2 is significantly faster than MobileNetV2, suggesting that MAC has a greater impact on inference speed than FLOPs. Therefore, this study focuses on analyzing the MAC issue *(Ma et al, 2018)*.

The aim of this paper is to reduce the impact of Elemwise operations on MAC and further improve speed by adopting the structure reparameterization and multi-branch structure design of RepVGG based on the efficient network criteria proposed by ShuffleNetV2, converting the multi-branch structure into a single path.

Figure 7 illustrates the significant reduction in Elemwise operations of RepVGG compared to YOLOv4 on different inference platforms. On GPU, Raspberry 4B, and Jetson Nano, RepVGG's Elemwise operations cause 50.0%, 40.2%, and 49.2% of the inference delay of YOLOv4, respectively. This indicates that the RepVGG design effectively solves the MAC problem inherent in YOLOv4.

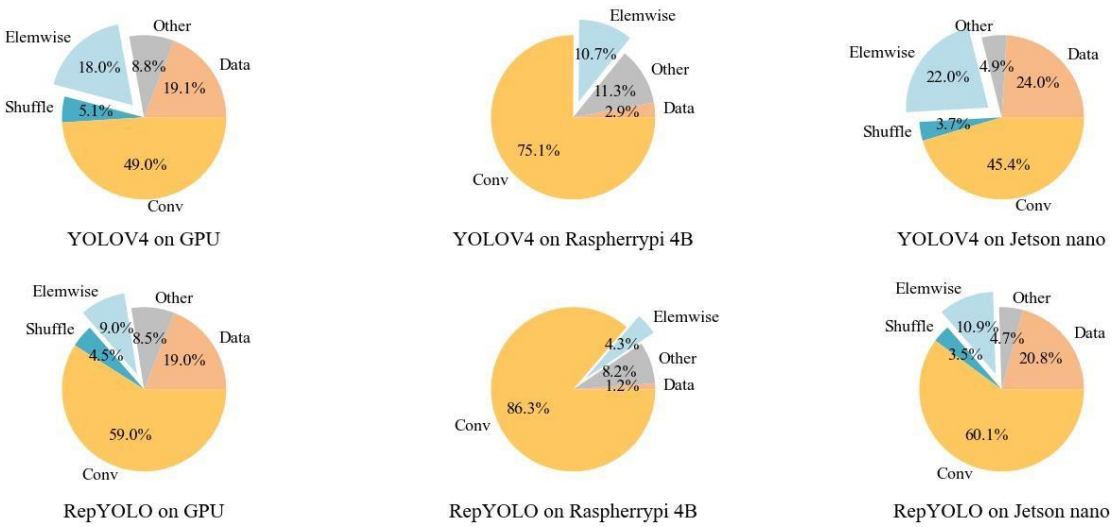


Fig. 7 - Run time decomposition on YOLOv4 and RepYOLO

The analysis of experimental results in Table 5 reveals that the RepYOLO standard model reduces inference time by 35.22%, 37.02%, and 41.44% on GPU, Raspberry 4B, and Jetson Nano, respectively, compared to the YOLOv4 standard model. These results demonstrate that the structures and reparameterization used in RepYOLO effectively reduce MAC and significantly improve inference speed.

Table 5

Speed Evaluation Metrics of YOLOv4 and RepYOLO

Base Model	Size	FLOPs/G	GPU/ms	4B/ms	nano/ms
YOLOv4	-n	8.2	225	126.7	79.2
	-s	16.5	49.0	276.0	126.7
	-l	33.00	112.0	469.2	215.4
RepYOLO	-n	8.2	14.4	81.1	34.2
	-s	16.0	31.7	173.8	74.2
	-l	32.9	72.5	281.5	160.2

In the subsequent Ablation Study section, the contribution of anchor-free to inference speed will be further analyzed. This analysis is necessary due to the influence of various factors on inference speed delay.

In order to enhance our comprehension of the influence of different modules and algorithms on the RepYOLO algorithm, a qualitative analysis was conducted in this section.

Table 6 presents the accuracy metrics of the RepYOLO algorithm with different modules and algorithms removed ((w/o) represents word without). Removing the ATSS algorithm significantly decreased the accuracy indicators of the RepYOLO algorithm with the same scale, even lower than the standard YOLOv4 detection algorithm (refer to table 5). This highlights the crucial importance of the ATSS algorithm in our proposed model and is consistent with previous research showing that anchor-free detection heads with positive and negative sample allocation algorithms have lower accuracy than anchor-based algorithms. Additionally, removing the CBAM algorithm resulted in decreased accuracy indicators. Then, heat maps were generated and compared with the original RepYOLO algorithm. Figure 6 shows that from the heat map visualization, it is evident that the CBAM algorithm plays a crucial role in improving the expression ability of the feature map. Simultaneously removing both algorithms resulted in a significant decrease in model performance, with accuracy much lower than the current model detection results, and even lower than the YOLOv4 detection performance.

Table 6

Ablation Study of RepYOLO Precision

Base Model	Size	P/%	R/%	F1/%	mAP/%
RepYOLO	-n	96.75	82.45	89.03	82.17
	-s	98.22	83.71	90.39	83.42
	-l	98.71	84.13	90.83	83.84
(w/o)CBAM and ATSS	-n	89.40	77.75	83.17	78.48
	-s	91.29	80.36	85.48	80.08
	-l	91.95	80.98	87.11	80.39
(w/o)ATSS	-n	92.53	80.62	86.17	80.03
	-s	94.36	82.18	87.85	81.58
	-l	95.04	82.76	88.48	82.16
(w/o)CBAM	-n	92.28	81.26	86.42	80.67
	-s	93.03	82.75	87.59	82.15
	-l	95.43	83.04	88.80	82.75

Table 7 presents the speed indicators of the RepYOLO algorithm with and without the Anchor-free detection head on various hardware platforms, denoted by (w/o) for the removed algorithm. Removing the Anchor-free detection head and using the Anchor-based detection head led to a significant increase in the inference latency of the RepYOLO algorithm with the same scale, with an approximately 8-12% increase in inference time. This finding suggests that the Anchor-free detection head can simplify the prediction of bounding boxes' calculation process, thereby enhancing computational efficiency and inference speed

Table 7

Ablation Study of RepYOLO Speed

Base Model	Size	FLOPs /G	GPU /ms	4B /ms	nano /ms
RepYOLO	-n	8.2	14.4	81.1	34.2
	-s	16.0	31.7	173.8	74.2
	-l	33.0	72.5	281.5	160.2
(w/o)Anchor-free	-n	8.9	15.7	111.1	38.0
	-s	17.5	34.9	203.8	81.4
	-l	36.6	79.8	310.5	179.4

CONCLUSIONS

In order to fully utilize the arithmetic capability of edge chips, the basic structure of the YOLO algorithm was modified in our work, using structural reparameterization and anchor-free detection head to make RepYOLO suitable for drone-friendly devices. However, changing the structure reintroduces the problem of reduced detection accuracy. Therefore, the CBAM attention mechanism and ATSS training method were adopted to enhance the original structure, enabling our model and structure to ultimately outperform the original YOLOV4 model on edge devices for drones.

In applications, software and hardware jointly limit the development of embedded IoT drones. The same network structure has different inference speeds on different hardware embedded platforms. The storage size of neural network parameter weight files in edge devices, memory sticks, and flash memories, as well as the limited computational performance of CPUs, limit the inference speed of neural network models. MCUNet proposed the inference framework of TinyNAS and TinyEngine, which can be used without relying on the cloud *(Lin et al., 2020)*. MCUNetv2 proposed a new memory bottleneck to solve the problem of memory allocation imbalance in CNN convolutional neural networks *(Lin et al., 2021)*. Compared with TF-lite and CMSIS-NN, MCUNet has achieved a fundamental transformation from mobile CPU inference to microcontrollers in terms of inference speed and accuracy.

ACKNOWLEDGEMENT

This research was supported by Fundamental Research Program of Shanxi Province (No.202203021212450 and No.202303021212115).

REFERENCES

[1] 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

[2] Chen, X.; Luo, X.; Weng, J.; Luo, W.; Li, H.; Tian, Q. (2021). Multi-view gait image generation for cross-view gait recognition. *IEEE Transactions on Image Processing*, 30, 3041-3055.

[3] David, E.; Serouart, M.; Smith, D.; Madec, S.; Velumani, K.; Liu, S.; Wang, X.; Pinto, F.; Shafiee, S.; Tahir, I.S. (2021). Global wheat head detection 2021: An improved dataset for benchmarking wheat head detection methods. *Plant Phenomics*. https://doi.org/10.34133/2021/9846158

[4] Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. (2021). Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*. https://doi.org/10.48550/arXiv.2107.08430

[5] Hassan, M.A.; Yang, M.; Rasheed, A.; Yang, G.; Reynolds, M.; Xia, X.; Xiao, Y.; He, Z. (2019). A rapid monitoring of NDVI across the wheat growth cycle for grain yield prediction using a multi-spectral UAV platform. *Plant science*, 282, 95-103. https://doi.org/10.1016/j.plantsci.2018.10.022

[6] He, K.; Zhang, X.; Ren, S.; Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*; pp. 770-778.

[7] Jin, X.; Liu, S.; Baret, F.; Hemerlé, M.; Comar, A. (2017). Estimates of plant density of wheat crops at emergence from very low altitude UAV imagery. *Remote Sensing of Environment*, 198, 105-114. https://doi.org/10.1016/j.rse.2017.06.007

[8] Khan, S.; Tufail, M.; Khan, M.T.; Khan, Z.A.; Iqbal, J.; Wasim, A. (2022). A novel framework for multiple ground target detection, recognition and inspection in precision agriculture applications using a UAV. *Unmanned Systems*, 10, 45-56. https://doi.org/10.1142/S2301385022500029

[9] Khalil, M.; Khomonenko, A.; Matushko, M. (2022). Measuring the effect of monitoring on a cloud computing system by estimating the delay time of requests. *Journal of King Saud University-Computer and Information Sciences*, 34, 3968-3972.10.https://doi.org/1016/J.JKSUCI.2021.02.001

[10] Lin, J.; Chen, W.-M.; Lin, Y.; Gan, C.; Han, S. (2020). Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems*, 33, 11711-11722.

[11] Radoglou-Grammatikis, P.; Sarigiannidis, P.; Lagkas, T.; Moscholios, I. (2020). A compilation of UAV applications for precision agriculture. *Computer Networks*, 172, 107148.

[12] Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*; pp. 4510-4520. https://doi.org/10.48550/arXiv.1801.04381

[13] Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, 2017; pp. 618-626. https://doi.org/10.1007/s11263-019-01228-7

[14] Shapiro, S.S.; Wilk, M.B. An analysis of variance test for normality (complete samples). *Biometrika* 1965, 52, 591-611. https://doi.org/10.1080/01621459.1972.10481232

[15] Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1-9. https://doi.org/10.48550/arXiv.1409.4842

[16] Wang, H.; Wu, X.; Liu, J.; Li, J. (2021). Research on Flame Detection Based on Anchor-Free Algorithm FCOS. In Proceedings of the Neural Information Processing: 28th International Conference, ICONIP 2021, Sanur, Bali, Indonesia, *Proceedings*, Part V 28, 2021; pp. 124-131. https://doi.org/10.48550/arXiv.1904.01355

[17] Woo, S.; Park, J.; Lee, J.-Y.; Kweon, I.S. (2018). Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 3-19.

[18] Zagoruyko, S.; Komodakis, N. Diracnets: Training very deep neural networks without skip-connections. arXiv preprint arXiv:1706.00388 2017. https://doi.org/10.48550/arXiv.1706.00388

[19] Zhang, S.; Chi, C.; Yao, Y.; Lei, Z.; Li, S.Z. (2020). Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9759-9768. https://doi.org/10.1109/CVPR42600.2020.00978