scientific reports

Check for updates

OPEN

YOLO-Granada: a lightweight attentioned Yolo for pomegranates fruit detection

Jifei Zhao, Chenfan Du, Yi Li, Mohammed Mudhsh, Dawei Guo, Yuqian Fan, Xiaoying Wu, Xinfa Wang & Rolla Almodfer✉

Pomegranate is an important fruit crop that is usually managed manually through experience. Intelligent management systems for pomegranate orchards can improve yields and address labor shortages. Fast and accurate detection of pomegranates is one of the key technologies of this management system, crucial for yield and scientific management. Currently, most solutions use deep learning to achieve pomegranate detection, but deep learning is not effective in detecting small targets and large parameters, and the computation speed is slow; therefore, there is room for improving the pomegranate detection task. Based on the improved You Only Look Once version 5 (YOLOv5) algorithm, a lightweight pomegranate growth period detection algorithm YOLO-Granada is proposed. A lightweight ShuffleNetv2 network is used as the backbone to extract pomegranate features. Using grouped convolution reduces the computational effort of ordinary convolution, and using channel shuffle increases the interaction between different channels. In addition, the attention mechanism can help the neural network suppress less significant features in the channels or space, and the Convolutional Block Attention Module attention mechanism can improve the effect of attention and optimize the object detection accuracy by using the contribution factor of weights. The average accuracy of the improved network reaches 0.922. It is only less than 1% lower than the original YOLOv5s model (0.929) but brings a speed increase and a compression of the model size. and the detection speed is 17.3% faster than the original network. The parameters, floating-point operations, and model size of this network are compressed to 54.7%, 51.3%, and 56.3% of the original network, respectively. In addition, the algorithm detects 8.66 images per second, achieving real-time results. In this study, the Nihui convolutional neural network framework was further utilized to develop an Android-based application for real-time pomegranate detection. The method provides a more accurate and lightweight solution for intelligent management devices in pomegranate orchards, which can provide a reference for the design of neural networks in agricultural applications.

Keywords Pomegranate growth period detection, YOLOv5, Lightweight network ShuffleNet, CBAM attention mechanism, Android deployment

Pomegranate (*Punica granatum* L.), a diploid fruit crop species (2n = 16), belongs to the Lythraceae family[1,2]. Pomegranate is considered to be a very important fruit cash crop in the semiarid tropics and is grown on approximately 550,000 hectares worldwide with a production of approximately 6.5 million tons[3]. As a nutritious, export-oriented fruit, it ensures high returns for growers and serves as a strategic crop for nutritional and livelihood security in water-scarce regions, contributing to climate change mitigation[4,5]. Reliance on manual experience management may lead to laborious processes, imprecise outcomes, reduced efficiency, and increased expenses. Automated management offers efficiency gains and cost reductions. Rapid and precise fruit crop detection, facilitated by computer vision technology, is crucial. The development of an algorithm accurately detecting pomegranate growth cycles enables tailored management interventions, holding significant research value and practical implications[6,7].

In the realm of deep learning-based object detection, two main categories emerge when considering network structure: two-stage and single-stage object detection methods. While two-stage algorithms, like RCNN, Fast RCNN and Faster RCNN[8–10], improve accuracy and speed, their high parameter count makes real-time detection computationally intensive. In contrast, single-stage object detection methods, from YOLOv1 through

College of Computer Science and Technology, Henan Institute of Science and Technology, Xinxiang, Henan, China. ✉email: rollajamil@hist.edu.cn

YOLOv7, represent advancements in single-stage detection efficiency. YOLOv1 based on GoogLeNet, used a grid-based approach, to predict object bounding boxes and class probabilities[11]. YOLOv2 switched to DarkNet19 and introduced Batch Normalization for enhanced convergence[12]. YOLOv3, which incorporates ResNet and DarkNet-53, addresses convergence challenges and integrates Feature Pyramid Networks for better small object detection[13]. YOLOv4 makes slight improvements upon YOLOv3[14]. YOLOv5 features faster speed, a smaller model size, and stronger deployment capabilities, showcasing excellent performance. YOLOv6 introduced EfficientRep and Rep-PAN, enhancing efficiency[15], and YOLOv7, while adopting a deeper Extended-ELAN structure, lacked significant improvement in speed and was unsuitable for lightweight terminals[16]. In this paper, YOLOv5 is the chosen benchmark, balancing accuracy, speed, model size, and deployment capabilities.

In agricultural applications, the choice of YOLO over two-stage object detection methods is driven by several factors, primarily its efficiency and suitability for complex environments such as those found in agriculture. YOLO's single-stage detection process, which performs object localization and classification simultaneously in one forward pass, provides a significant speed advantage for real-time detection tasks. This is crucial in agricultural settings where conditions can change rapidly, and decisions need to be made quickly. Additionally, YOLO's smaller model size and fewer parameters make it more adaptable for use on devices with limited computational power, such as mobile devices and embedded systems commonly used in agricultural operations.

However, despite these advantages, there are significant challenges in the application of even the most advanced versions like YOLOv5 in agriculture. The complexity of its network structure, the extensive number of parameters, and the high configuration requirements for training pose barriers, resulting in low frame rates for real-time detection and high deployment costs. These challenges are compounded by the fact that simplifying the YOLO network to reduce its computational demands often leads to decreased accuracy and increased error rates, which can escalate costs further. Moreover, the specific agricultural challenges such as the analysis of growth stages in crops like pomegranates are currently under-researched, indicating a gap that needs to be addressed to fully harness YOLO's capabilities in these settings. Thus, while YOLO's design is theoretically advantageous for agriculture due to its speed and efficiency, practical deployment issues and a lack of tailored research hinder its effectiveness and widespread adoption.

This study introduces effective strategies to address the challenges previously mentioned. The key contributions of this manuscript are summarized below:

1. We introduce the Attentioned-ShuffleNet YOLO-Granada deep learning model, designed to enhance detection speed and reduce network complexity. This model integrates the lightweight ShuffleNetv2 network as its backbone, ensuring computational efficiency and real-time performance. Additionally, we enhance object detection accuracy by incorporating the Convolutional Block Attention Module (CBAM) into the ShuffleNetv2 network. This simple yet effective module strengthens the output signal, improving the network's ability to distinguish between object features and thereby increasing detection accuracy. The Attentioned-ShuffleNet YOLO-Granada model thus offers a lightweight, efficient, and precise solution for detecting pomegranate fruits;
2. Our research encompasses a variety of optimization techniques for deep learning models, including 12 ablation experiments across two major groups to fine-tune and enhance the YOLO-Granada network. We compare the performance of our model against other object detection methods through 22 validation experiments, assessing its effectiveness in a cost-effective CPU environment;
3. Additionally, we have developed an Android-based real-time pomegranate detection application using the Nihui convolutional neural network (NCNN) framework. This application demonstrates the practical viability of the YOLO-Granada model for monitoring pomegranate growth and provides valuable insights for researchers in agricultural engineering and artificial intelligence.

The structure of this paper is organized as follows: The Literature Review section details previous research relevant to the topic. The Resarch Background section provides the foundational context for the study. In the Methodology section, we introduce our proposed YOLO-Granada structure. The Experiments and Results section discusses the experimental setup, evaluation metrics, and comparative results with other models. Finally, the Conclusion and Future Works section summarizes the study's findings, elaborates on the techniques introduced, and outlines directions for future research.

Literature review

The theoretical framework in the Introduction highlighted that single-stage object detection algorithms like YOLO are more lightweight compared to two-stage algorithms like RCNN. This section expands on that by analyzing the preference for YOLO over traditional RCNN from current research, especially in agricultural applications, citing key studies that demonstrate its advantages.

YOLO was first introduced in 2015, and in a short period, it gained increasing attention in the agricultural field, showing exponential growth by November 2023[17]. In agricultural applications, YOLO has demonstrated its effectiveness in various studies. Yang et al.[18] emphasizes tea bud recognition using seven mainstream algorithms, with YOLO achieving the highest classification accuracy, supporting its use in mechanical intelligent tea picking. Tian et al.[19] presents an enhanced model for detecting apples in complex orchard environments, outperforming YOLOv3 and Faster R-CNN with VGG16. Khan et al.[20] introduces a novel approach using CNN and transformer for tomato ripeness classification, achieving high accuracy but being limited by its large model size.

The versatility and effectiveness of YOLO in agriculture cover a wide range of tasks. Wang et al.[21] addresses intelligent online yield estimation for tomatoes, achieving 99.3% mAP with an enhanced YOLOv3 model. Similarly, Tang et al.[7] proposes ESPA-YOLOv5s for detecting rapeseed seedling survival rates, achieving an

mAP of 99.6%. Zheng et al.[22] introduces YOLO-BP for detecting green citrus, achieving an mAP of 91.55% and a detection speed of 18 fps. Tang et al.[23] presents YOLO-Oleifera for detecting fruits in complex orchard environments, achieving an AP of 0.9207. Dong et al.[24] enhances YOLOv5 for pest and disease detection in nine major crops (primarily rice and corn), achieving an mAP of 90.7. Lin et al.[25] modifies the backbone network to detect citrus and employs a self-attention mechanism to enhance the interaction and fusion of local and global features, significantly improving the model's occlusion detection capability, achieving an mAP of 83.2%. Omer et al.[26] uses the bottleneck CSP module to replace C3 as the backbone and neck network parts and incorporates the convolutional block attention module (CBAM) for improvements. This approach detects cucumber pests and diseases with an improved model mAP of 80.10%, occupying only 13.6 MB of memory. Lan et al.[27] introduces the EMA attention mechanism and designs a novel backbone network for rice panicle detection. References[28] and[29] discuss high-precision models for wheat ear detection and wheat flowering stage determination, respectively, incorporating attention modules and feature fusion techniques to enhance performance. Zhang et al.[30] improves the YOLOv5 model for orchard tree trunk detection by replacing it with the lightweight GhostNet V2 and adding the coordinate attention mechanism (CA) to reduce interference from irrelevant background information in images. It is evident that most current studies achieve lightweight models by replacing the backbone network or improving accuracy by adding attention mechanisms. Therefore, our research also focuses on these two aspects. Additionally, most studies focus on detecting the presence, location, and classification of targets in relatively simple environments, which highlights the difficulty of detecting pomegranate growth stages in more complex settings.

Research on pomegranates is relatively scarce and primarily focuses on disease identification and quality grading. Vasumathi et al.[31] proposes a dragonfly optimization algorithm based on CNN-LSTM technology for identifying pomegranate diseases, achieving a classification accuracy of 92%, which improves to 97.1% with the optimization. Mitkal et al.[32] presents an automatic grading method for pomegranates using CNN, K-means, and image processing techniques to improve production efficiency and quality.

Resarch background

ShuffleNet v2

In 2018, Ningning Ma conducted comprehensive experiments highlighting the limitations of using FLOPs as the sole measure of computational complexity[33]. The study emphasized the importance of considering factors such as the memory access cost (MAC), hardware platform, and algorithm complexity. Building upon ShuffleNetv1[34], an enhanced and improved version called ShuffleNetv2 was proposed. Given a computational complexity budget of 40 MFLOPs, ShuffleNetv2 demonstrated a 3.5% and 3.7% increase in accuracy compared to ShuffleNetv1 and MobileNetv2 respectively. ShuffleNet incorporates two main innovations: group convolution to reduce computational effort and channel shuffle to enhance interaction between different channels. The network structure of ShuffleNet is illustrated in Fig. 1, with Fig. 1a representing the original ShuffleNetv1 and Fig. 1b illustrating the downsampled version. The network utilizes grouped point convolution, followed by a shuffle operation, a 3×3 depth-separable convolution, and another grouped point convolution. ShuffleNetv2 further improves upon the original design, as depicted in Fig. 1c, d. The input layer employs channel split instead of group

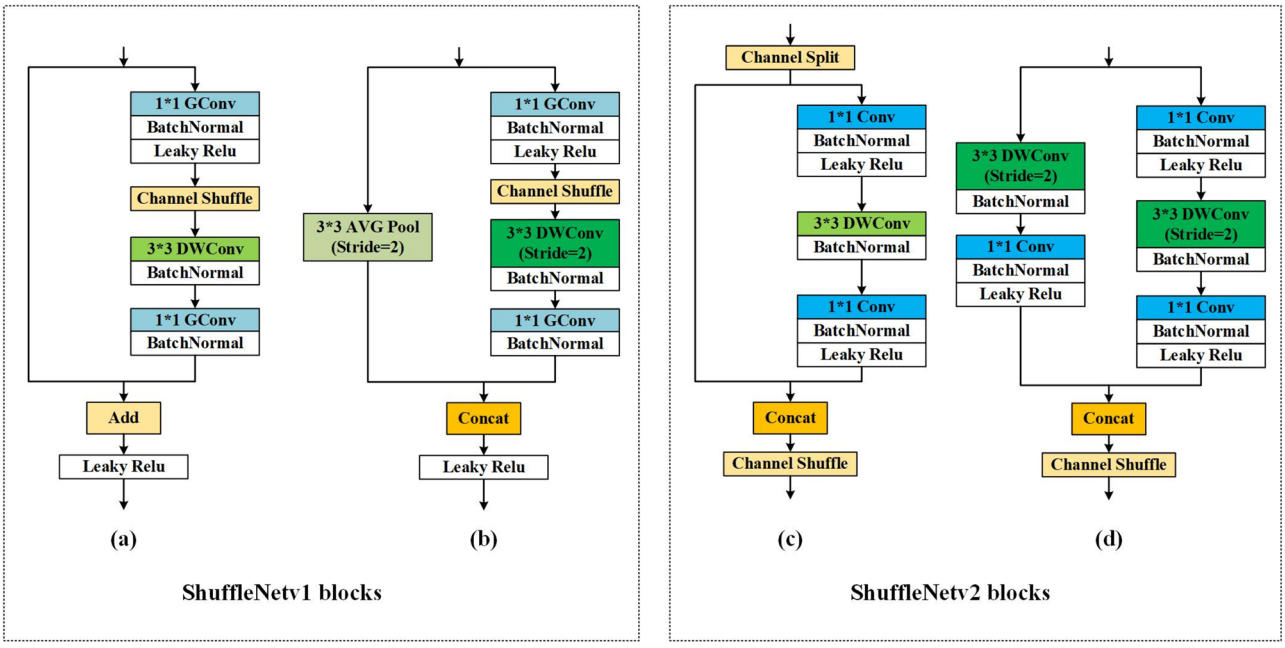

Figure 1. Building blocks of ShuffleNetv1 and ShuffleNetv2. (**a**) The basic ShuffleNetv1 unit. (**b**) The ShuffleNetv1 unit for spatial down sampling (2 ×). (**c**) The basic ShuffleNetv2 unit. (**d**) The ShuffleNetv2 unit for spatial down sampling (2 ×). DWConv: depthwise convolution. GConv: group convolution.

convolution, dividing the feature channels into two parts to enable separate computations within the channels. The output result employs concatenation instead of the add operation, and the bottleneck part no longer includes the channel shuffle operation.

The modifications in ShuffleNetv2 allow for increased accommodation of feature channels. By performing convolutional calculations on subsets of feature channels after the channel splitting operation, computational volume and parameters are reduced. Furthermore, feature channels within the network module are directly forwarded to the next module without convolutional computations, enabling feature reuse.

Convolutional block attention module

The attention mechanism revolutionizes agricultural image processing by filtering out irrelevant information and honing in on the region of interest. Originating in the 1990s, attention mechanisms gained momentum when the Google Mind team[35] applied them to RNN models for image classification in 2014. These mechanisms help neural networks suppress less salient features in the channel or space. While earlier studies overlooked the contribution factor of weights, recent research underscores its role in enhancing attention. In 2018, Sanghyun Woo introduced the convolutional block attention module (CBAM) for feed-forward convolutional neural networks, demonstrating its efficacy in adaptive feature optimization[36]. CBAM's lightweight design seamlessly integrates into all CNN architectures, allowing end-to-end training. Figure 2 illustrates the channel attention module, showcasing the steps involving global max pooling, global average pooling, and multiplication through a multilayer perceptron (MLP) to derive the essential input features for the spatial attention module.

The channel attention module compresses the spatial dimension of the feature map to obtain a one-dimensional vector and applies operations to it. This compression involves both average pooling and max pooling, which aggregate spatial information from the feature maps. The compressed spatial dimensions are then combined elementwise to generate a channel attention map. In this mechanism, channel attention focuses on determining the important elements within a single graph. Mean pooling provides feedback for every pixel point on the feature map, while maximum pooling provides feedback only for the gradients where the response is the largest in the feature map during gradient backpropagation calculations. The channel attention mechanism can be represented as follows:

$$f_{avg}(F^c) = \frac{1}{n}\sum_{i,j} F^c_{i,j} \tag{1}$$

$$f_{max}(F^c) = \max_{i,j} F^c_{i,j} \tag{2}$$

$$M_c = \sigma(f(W_\gamma, f(W_\lambda, f_{avg}(F^c))) + f(W_\gamma, f(W_\lambda, f_{max}(F^c)))) \tag{3}$$

It is important to note that the MLP weights, W_λ and W_γ are shared for both inputs. Additionally, the ReLU activation function is applied following W_λ.

Figure 2 illustrates the Channel and spatial attention module. The output feature map from the channel attention module serves as the input feature map for this module. Initially, global max pooling and global average pooling are performed along the channel dimension. These results are concatenated based on the channel. Finally, the spatial attention feature is obtained by multiplying this feature with the input feature from the module, resulting in the final generated feature.

Likewise, the spatial attention module compresses the channels by applying mean pooling and max pooling operations along the channel dimension. The AvgPool operation extracts the average value across the channels,

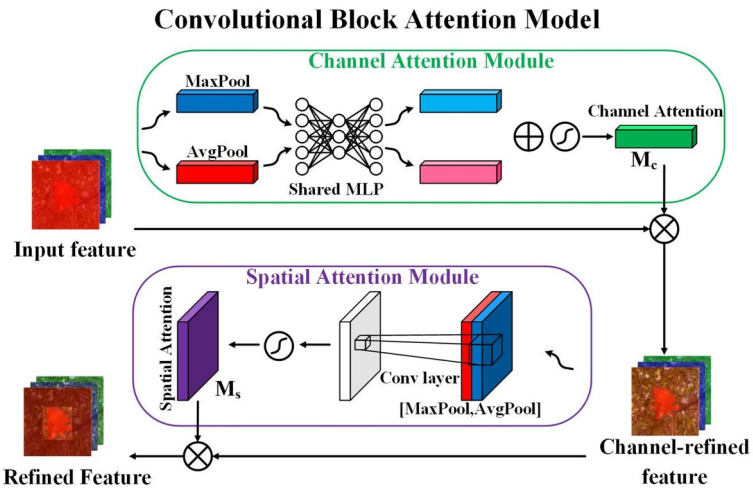

Figure 2. Channel and spatial attention mechanism.

with the number of extractions equal to the product of the height and width. Subsequently, the previously extracted feature maps, with a channel count of 1, are combined to yield a 2-channel feature map.

$$M_s = sigmiod(f_{avg}^{7*7}(F^s); f_{max}^{7*7}(F^s)) \tag{4}$$

The size of the convolution kernel is denoted by 7×7, and it has been observed that a 7×7 convolution kernel performs better compared to a 3×3 convolution kernel.

Methodology

In this research endeavor, the YOLOv5s architecture served as the fundamental network, wherein a series of enhancements were meticulously incorporated to address the primary objectives of model lightweightness and compatibility with CPU utilization. The specific improvements encompassed the following aspects: (1) utilization of the backbone structure from Attentioned-ShuffleNet as a substitute for the CSPDarknet-53 network serving as the backbone in YOLOv5; and (2) integration of the CBAM. To provide a visual representation of the proposed YOLO-Granada object detection network, we present the comprehensive framework diagram depicted in Fig. 3. This diagram encompasses four principal components, namely, the input, backbone, neck, and prediction components, each of which plays a crucial role in the overall functioning of the network.

In Fig. 3, Conv represents the convolution operation and DWConv represents the packet convolution operation. BatchNormal represents the data normalization method adopted as batch normalization. Leaky Relu and Leaky Silu are activation features of the network. Concat means the stitching of the feature map, which increases the dimension of the feature, and Concat does not increase the amount of parameter operation compared to the Add operation.

Input

The input section of the network encompasses image preprocessing and enhancement. In the preprocessing step, the input image is resized to match the network's specifications, and the pixel values are normalized from the [0-255] range to the [0,1] range. In the enhancement process, the preprocessed images are further improved using mosaic data and adaptive anchor box calculations utilizing k-means. The network model incorporates the recorded anchor box information to enhance its performance.

Backbone: attentioned-ShuffleNet

In the proposed YOLO-Granada, the backbone architecture consists of a ShuffleNetv2 Which utilizes pointwise group convolution and channel shuffle operations to reduce computational costs. The backbone is further enhanced with three CBAMs. The purpose of these CBAMs is to introduce attention mechanisms and promote information flow within the network. Each CBAM is placed after a downsampling operation and a convolutional block.

In the proposed model, the input feature maps are initially processed through a convolutional block. Let's denote the input feature maps as X_{in}. The output feature maps from the first convolutional block can be represented as X_{conv1}.

Following the convolutional block, a channel shuffle operation is applied to X_{conv1}. We can denote the channel shuffle operation as $S(\bullet)$, which rearranges the feature maps along the channel dimension. Mathematically, the channel shuffle operation can be represented as:

$$X_{shuffled} = S(X_{conv1}) \tag{5}$$

The shuffled feature maps, $X_{shuffled}$, are then passed through the first CBAM attention module. This module computes attention weights, A, for each spatial location in the feature maps. Let $A_{i,j}$ represent the attention weight at spatial location i and channel j. The attention weights are calculated by comparing the features at each spatial location with all other spatial locations in the feature maps. Mathematically, the proved attention weights can be computed as:

$$A_{i,j} = M_s(M_c(X_{shuffled,i,j}) \otimes X_{shuffled,i,j}) \otimes (M_c(X_{shuffled,i,j}) \otimes X_{shuffled,i,j}) \tag{6}$$

Once the attention weights $A_{i,j}$ have been computed, they are used to scale the feature maps. The feature maps are multiplied elementwise by the attention weights, effectively scaling the importance of each feature based on its relevance to other features in the network. Mathematically, the scaled feature maps denoted as X_{scaled}, can be calculated as:

$$X_{scaled,i,j} = X_{shuffled,i,j} \cdot A_{i,j} \tag{7}$$

The scaled feature maps, X_{scaled}, are then passed to the second convolutional block, where the process repeats. Furthermore, the scaled feature maps are also fed to the neck of YOLO-Granada, specifically between the upsampling block and the CSP block.

Let $X_{upsampled}$ represent the feature maps obtained after the upsampling block, and X_{CSP} represent the feature maps obtained after the CSP block in the YOLO-Granada neck. The scaled feature maps, X_{scaled}, are combined with $X_{upsampled}$ to form the input for the CSP block. This operation can be represented as:

$$X_{CSP} = CSP(X_{upsampled} \oplus X_{scaled}) \tag{8}$$

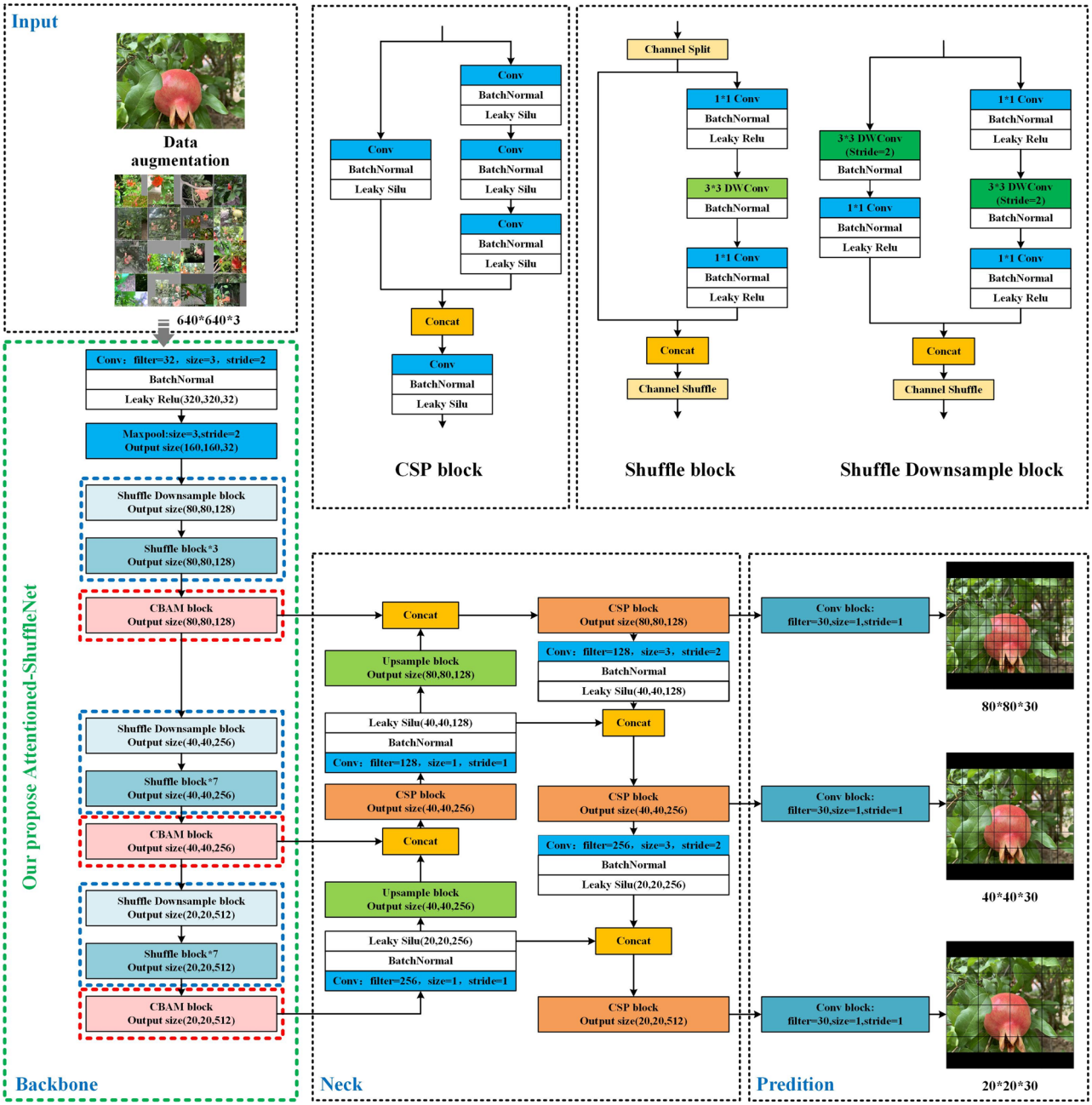

Figure 3. Structure diagram of YOLO-Granada.

where $\bigoplus$ denotes the concatenation operation along the channel dimension, and $CSP(\bullet)$ represents the CSP block. The CSP block processes the concatenated feature maps and produces the final feature maps, X_{out}, which are then used for subsequent operations in the YOLO-Granada network.

Finally, the globally pooled feature maps, X_{CSP}, are passed through a fully connected layer, resulting in the final classification scores.

Neck

The neck module in the YOLO-Granada algorithm is responsible for fusing features from different scales and generating high-quality feature maps, which are essential for accurate object detection. The neck module consists of multiple convolutional layers that are designed to merge features from different stages of the backbone network. In particular, the neck module in YOLO-Granada uses a feature pyramid network (FPN) architecture to fuse features from different stages of the backbone network. The FPN is a popular architecture for object detection that is designed to generate feature maps at multiple scales. These feature maps are then fused using lateral connections to produce a final set of feature maps that are well-suited for object detection.

Overall, the neck module plays a critical role in the YOLO-Granada algorithm by fusing features from different scales to generate high-quality feature maps that are essential for accurate object detection.

Prediction

When using the YOLO generic prediction layer, the feature map of the feature pyramid layer is first processed using the header model to predict the bounding box containing the target, the target class, and the object score. The prediction frames are then filtered using the distance intersection over union (DIoU) and non-maximum suppression (NMS) algorithm to exclude highly overlapping predictions. Finally, complete intersection over union (CIoU) is used as the loss function for the output-side bounding box to achieve better and faster convergence.

To generate prediction frames on the feature map, anchor frames are applied to the feature map. Therefore, the total size of the output vector can be represented using formula (9):

$$S_{out} = Num_{af} \times (Num_{trlp} + 1 + Num_{tc}) \tag{9}$$

where: S_{out} is the output size; Num_{af} is the number of anchor frames; Num_{trlp} is the number of target rectangle location parameters; 1 represents the target confidence, and Num_{tc} is the number of target categories.

For the pomegranate detection model, with feature maps upsampled to size of 80×80, 40×40 and 20×20, and detecting 5 different target categories of pomegranates, the number of target rectangle location parameters is 4. Therefore, the output sizes of each feature map for the pomegranate detection model are [80, 80, 30], [40, 40, 30] and [20, 20, 30], respectively.

Experiments and results

Data sets

To detect the growth stages of pomegranates, we employed a dataset specifically compiled for observing pomegranate development, as detailed in[37]. This dataset consisted of 5857 images. We allocated 4685 images (80%) for training the network, 585 images (10%) for validation and parameter tuning, and the remaining 587 images (10%) for performance assessment. Each image in the dataset was labeled to represent different developmental stages of pomegranate growth, categorized as "bud," "flower," "early-fruit," "mid-growth," or "ripe," as illustrated in Fig. 4. This labeling enabled the model to learn from a wide range of examples of pomegranate growth phases, improving its accuracy in classifying new images. The structured method allowed systematic training, validation, and testing, ensuring the network's robust performance in practical applications.

Performance evaluation metrics for network models

Different evaluation metrics reflect the performance of the detection algorithm in different aspects. The evaluation metrics for this study are shown in Table 1.

The precision, recall, F1 score, and average precision were used to assess the model detection accuracy. Model complexity and scalability are measured by weight size, params, and FLOPs. Speed and efficiency are gauged through the FPS. The precision (P) is the ratio of correctly detected pomegranates to the sum of correctly and incorrectly detected pomegranates. Recall (R) is the ratio of correctly detected pomegranates to the sum of correctly detected and undetected pomegranates. TP and FP denote correct and incorrect detections, respectively, while FN represents true pomegranates not detected. The F1 score is the harmonic mean of the precision and recall. The mAP is the mean average accuracy across all categories.

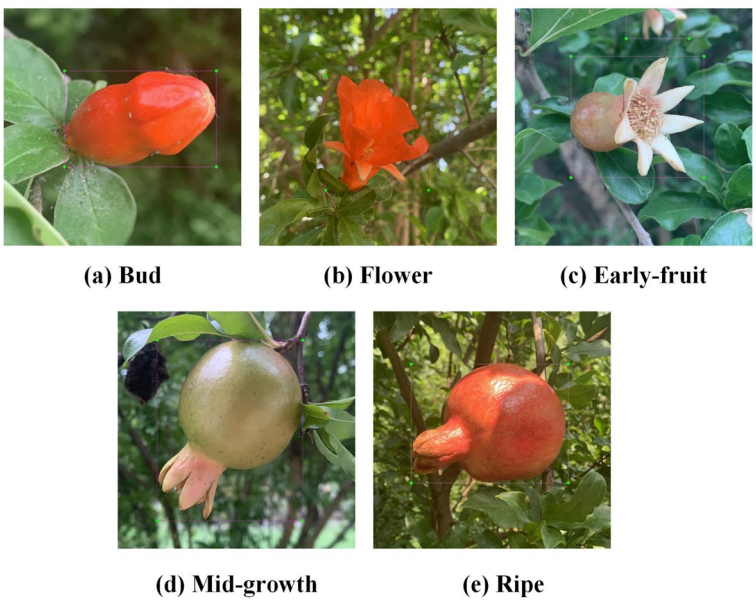

Figure 4. Images of pomegranates in different growth stages.

Items	Formulas
Detection accuracy	
P	$Precision = TP - (TP + FP)$
R	$Recall = TP/(TP + FN)$
F1	$F1scores = (2 \times P \times R)/(P + R)$
mAP@0.5	$mAP = \left(\sum_{i=1}^{k} AP_i\right)/k$
Model complexity	
Size	$Modelweightsize$
Params	$Params = \sum(K_h \times K_w \times C_{in} \times C_{out})$
FLOPs	$FLOPs = \sum(K_h \times K_w \times C_{in} \times C_{out} \times H \times W)$
Detection speed	
FPS	$FPS = 1000/(pretreated + inference + NMS)$

Table 1. Model evaluation metrics.

Params refers to the total number of trainable parameters in the model. FLOPs represents the number of floating point operations. C_{in} and C_{out} represent the input and output channel counts, respectively. K_h and K_w denote the width and height of the convolution kernel, and H and W represent the width of the feature map. FPS (frames per second) is a metric used to assess the actual detection speed of a model. Higher FPS values indicate better real-time performance. The components that contribute to the processing time include: (1) Image pre-processing time (*pretreated*): It involves tasks such as maintaining the aspect ratio during image scaling and padding, channel transformation (HWC to CHW), and dimensioning; (2) Inference time (*inference*): It represents the time taken from feeding the pre-processed image to the model until obtaining the output result; (3) Post-processing time (*NMS*): It includes tasks such as transforming the model's output result. The time unit for these components is milliseconds (ms).

Selection of hyperparameters and operating conditions

Selection of hyperparameters

The network models were trained in a consistent environment, employing dynamic learning rate adjustment. The initial learning rate, recurrent learning rate, learning rate momentum, weight decay coefficient, batch size, and maximum number of iterations (epochs) were set to 0.01, 0.01, 0.937, 0.0005, 32 and 300 respectively. In the image augmentation section, a mosaic operation was performed with a probability of 1. Flipping the image vertically was set to a probability of 0, while flipping it horizontally was set to a probability of 0.5.

Training configuration

The training environment was created by Anaconda 3 and configured with Python 3.7.12, PyTorch 1.11 and Torch Vision 0.11.1 artificial neural network libraries. In addition, the CUDA 11.2 deep neural network acceleration library was used.

Configuration of the comparison test

In order to verify the effect of model size on terminals, we conducted comparative experiments using different models on CPUs. In the comparison experiments for each model in Detection Results, the model was built on the deep learning framework PyTorch 1.11, and the program code was written in Python 3.7.12. Our detection experiments were performed on inexpensive devices and did not use GPUs, so we conducted detection experiments on a computer platform consisting of an Intel(R) Core(TM) i5-10400F CPU. The batch size is set to 1, i.e., one image is detected at a time; the confidence threshold is set to 0.001; and the NMS IoU threshold is set to 0.6.

Mobile deployment

To deploy the YOLO-Granada model on mobile devices efficiently, we followed these steps:

1. Export weight files to the ONNX format for model generalization across platforms.
2. Use the NCNN framework for deployment, converting ONNX format models to the ".ncnn" format for optimized execution on cell phones.
3. Simplify the ONNX model using the ".onnx-simplifier" tool to enhance the efficiency of end devices.
4. Configuration of the NCNN parameters, optimizing the network structure and runtime parameters, potentially involving tuning with ".param" files.
5. Deploy the model to Android mobile terminals, managing dependencies and building with Gradle 7.2.0, ensuring compatibility with specific libraries and tools, using JDK 18 as the development toolkit, and targeting Android 10 for stability. This setup allows efficient execution of the YOLO model on Android phones.

Through these steps, we successfully exported the YOLO-Granada model to ONNX format, deployed it on a cell phone via the NCNN framework, and configured relevant parameters for efficient mobile operation. The detailed test experiments are presented in Results for Cell Phone Terminals.

Ablation experiments

The study employs the original YOLOv5s, a widely used version with a 13.76 MB model size, as the baseline for experiments. Focusing on lightweight tasks, we primarily replace the backbone network. Ablation experiments on the YOLOv5s benchmark, maintaining default parameters, assess various improvements. We test six mainstream lightweight networks to substitute for CSPDarknet-53, and the results are displayed in Table 2. The table comprehensively compares their performance in terms of parameters, computations, model size, and accuracy.

Table 2 presents a thorough evaluation of six mainstream lightweight networks replacing YOLOv5's original CSPDarknet-53 , MobileNetv3-small and ShuffleNetv2 emerged as superior alternatives, boasting compact model sizes of only 6.20 and 7.65 respectively. These smaller sizes not only demonstrate minimal resource requirements but also maintain high accuracy, effectively underscoring the success of the lightweight design.

Our focus is on replacing the backbone network for model lightweighting, which typically resultis in decreased accuracy. To address this, we adopt a strategy of enhancing the backbone network to improve the accuracy.

Incorporating attention mechanisms helps suppress less prominent features in channels or spatial dimensions. We conduct experiments with five attention mechanisms (NAM[38], GAM[39], SE[40], SimAM[41] and CBAM) integrated into the backbone network after feature extraction. This allows the model to allocate greater attention to more prominent features.

The study outcomes are presented in Table 3, which showcased efforts to enhance the backbone network.

Analyzing Table 3 shows the performance of MobileNetv3-small and ShuffleNetv2 with various attention mechanisms, revealing that common attention mechanisms do not significantly enhance the performance of MobileNetv3-small. Specifically, both the F1 scores and mAP@0.5 values for MobileNetv3-small with these mechanisms are close to or slightly below those without any attention mechanisms. In contrast, integrating CBAM with ShuffleNetv2 markedly boosts its performance. As indicated in Fig. 5 and Table 3, ShuffleNetv2 + CBAM achieves an F1 score of 87.9% and an mAP@0.5 of 0.922, the highest among all combinations tested. This suggests that CBAM effectively enhances the attention mechanism in ShuffleNetv2, making it the preferred choice for our lightweight detection model. We have therefore incorporated CBAM into ShuffleNetv2 and utilized it as the backbone for our proposed YOLO-Granada model.

The structural differences between ShuffleNetv2 and MobileNetv3-small contribute to the varying impacts of CBAM. ShuffleNetv2's design, which includes pointwise group convolutions and channel shuffling, provides an ideal framework for CBAM to optimize feature extraction and attention weighting. The group convolutions allow

Model(YOLO)	Params	GFLOPs	Size (MB)	F1 (%)	mAP@0.5
v5s(CSPDarknet-53)	7,033,114	15.9	13.76	88.9	0.928
MobileNetv2	4,603,738	10.0	9.20	87.1	0.917
MobileNetv3-large	5,583,072	10.8	11.05	88.5	0.92
MobileNetv3-small	3,586,312	6.5	7.20	87.3	0.911
EfficientNetv1	5,760,722	11.4	11.41	88.5	0.928
EfficientNetv2	27,298,066	35.8	52.74	89.6	0.933
ShuffleNetv2	3,803,738	8.1	7.65	87.1	0.916

Table 2. Performance comparison of YOLOv5s with different backbone networks.

Model	Params	GFLOPs	Size (MB)	F1	mAP@0.5
MobileNetv3-small	3,586,312	6.5	7.20	87.3%	0.911
+NAM	3,591,934	6.5	7.25	87.4%	0.908
+GAM	5,287,642	11.5	11.47	87.6%	0.913
+SE	3,601,940	6.6	7.30	87.5%	0.910
+SimAM	3,586,330	6.5	7.21	87.4%	0.911
+CBAM	3,602,984	6.6	7.35	87.5%	0.912
ShuffleNetv2	3,803,738	8.1	7.65	87.1%	0.916
+NAM	3,805,530	8.0	7.66	87.0%	0.913
+GAM	6,087,642	12.2	12.03	86.3%	0.914
+SE	3,846,746	8.1	7.74	87.4%	0.916
+SimAM	3,803,738	8.1	7.65	87.4%	0.915
+CBAM(ours)	3,847,995	8.0	7.74	87.9%	0.922

Table 3. Performance comparison with different attention mechanisms in modified backbone networks.

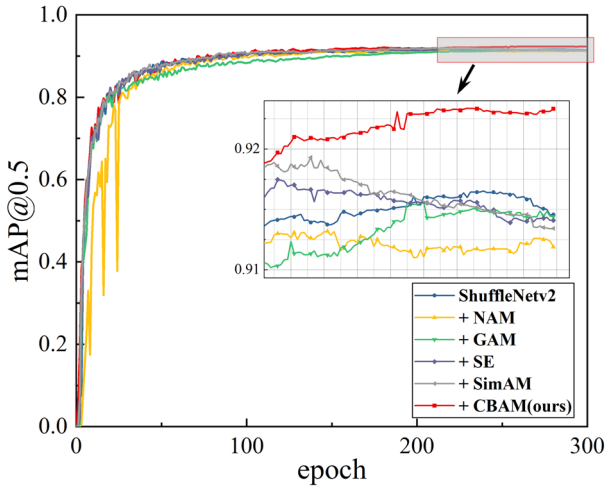

Figure 5. Performance comparison with different attention mechanisms in modified backbone networks.

CBAM to more effectively emphasize important feature channels and suppress the less relevant ones, leading to substantial performance improvements. On the other hand, MobileNetv3-small, which employs depthwise separable convolutions, already maximizes efficiency in feature extraction and channel separation due to its lightweight design. Consequently, additional attention mechanisms like CBAM offer negligible benefits, as the model's inherent efficiency and streamlined architecture leave little scope for further enhancement by these mechanisms.

It is also noteworthy that, apart from GAM, the other four attention mechanisms (NAM, SE, SimAM, and CBAM) do not significantly alter model complexity, increase size, or require additional computational resources. GAM introduces a small feedforward neural network, thereby increasing trainable parameters and model size. The convolutional block attention module (CBAM), in contrast, improves attention functionalities in CNNs without notably expanding parameter count. CBAM leverages existing convolution operations, extracts features, and computes attention weights through a designed attention subnetwork, distinguishing itself as an effective and lightweight attention mechanism that enhances performance while keeping computational and memory demands low.

Detection results of YOLO-Granada model

To benchmark YOLO-Granada against other state-of-the-art methods, we selected 22 renowned neural networks, such as various versions of YOLO (including YOLOv3 to YOLOv7) as baseline methods for comparative experiments.

As depicted in Table 4, YOLOv3 achieves 89.85% accuracy (F1) and 0.936 mAP on the dataset, validating in 694.7 s with an FPS of 1.43. YOLOv3-spp, a variant, has a slightly lower accuracy and mAP@0.5 but a similar validation time and FPS. YOLOv4 and YOLOv4-CSP exhibit lower accuracy and mAP@0.5 than YOLOv3, with comparable validation times and FPS. YOLOv4-tiny has lower GFLOPs and size but lower accuracy, mAP@0.5, higher validation time, and reduced FPS.

YOLOv5 series, despite higher GFLOPs and size, achieves similar or better accuracy and mAP@0.5 with higher FPS. YOLOv5s, with the lowest GFLOPs and size, achieves 89.37% accuracy and 0.929 mAP@0.5. YOLOv5m/l increases in GFLOPs and size, accuracy, and mAP@0.5 but decreases FPS. YOLOv5x has the highest GFLOPs, size, 90.07% accuracy, and 0.937 mAP@0.5.

YOLOv6 (s, m, l, x series) surpasses v5 in volume, parameters, and GFLOPs without accuracy improvement (mAP@0.5 0.925-0.929 vs. v5 0.929-0.937). Thus, v5 is preferred. YOLOv7 and R series exhibit accuracy over 0.94 but have larger size, parameters, and GFLOPs (>10 times ours). Not suitable for our lightweight task.

In Table 4, our method attains 0.922 precision, ensuring accurate object detection with a high recall of 0.833 and a balanced F1 score of 87.84%. The table provides a comprehensive comparison of detection results for the YOLO-Granada model and various other models, including the original YOLOv5s. Notably, YOLO-Granada has the smallest model size among all the models, at 7.74 MB, which is a reduction of 56.3% compared to YOLOv5s's 13.76 MB. This significant reduction in model size makes YOLO-Granada more efficient for real-time applications on devices with limited computational resources.

Our analysis shows that while YOLO-Granada achieves a slightly lower mAP@0.5 (0.922) and F1 score (87.84%) compared to YOLOv5s (mAP@0.5 of 0.929 and F1 score of 88.37%), it offers significant improvements in speed and model size. Specifically, YOLO-Granada is approximately 17.3% faster, achieving 8.66 FPS compared to 7.19 FPS for YOLOv5s. Additionally, the parameters, floating-point operations, and model size of YOLO-Granada are compressed to 54.7%, 51.3%, and 56.3% of those in YOLOv5s, respectively. These trade-offs highlight the effectiveness of YOLO-Granada in balancing accuracy, speed, and model size, making it suitable for practical deployments in agricultural monitoring and other similar tasks.

Model	Parameters	GFLOPs	Size (MB)	*P*	R	F1 (%)	mAP@0.5	Val Time	FPS (f/s)
v3	61,518,970	154.6	117.83	0.900	0.897	89.85	0.936	06:49	1.43
v3-spp	62,568,058	155.4	119.84	0.922	0.878	89.95	0.937	07:04	1.38
v3-tiny	8,675,932	12.9	16.64	0.901	0.841	87.00	0.911	00:54	10.67
v4	60,412,730	130.7	115.90	0.928	0.866	89.59	0.934	07:46	1.25
v4-csp	52,492,090	119.0	100.73	0.942	0.865	90.19	0.940	06:43	1.45
v4-tiny	5,880,252	16.1	11.33	0.897	0.826	86.00	0.903	00:53	10.93
v5s	7,023,610	15.8	13.76	0.921	0.868	89.37	0.929	01:21	7.19
v5m	20,869,098	47.9	40.26	0.933	0.863	89.66	0.933	02:50	3.43
v5l	46,129,818	107.7	88.56	0.936	0.863	89.80	0.937	05:27	1.79
v5x	86,200,330	203.8	165.13	0.936	0.868	90.07	0.937	08:52	1.10
v5s6	12,323,608	16.2	23.95	0.911	0.860	88.48	0.932	01:28	6.61
v5-fpn	39,047,066	90.1	74.98	0.919	0.873	89.54	0.935	04:52	2.00
v5-BiFPN	46,391,962	108.5	89.06	0.922	0.864	89.21	0.931	05:32	1.76
v6s	16,298,399	44.0	31.33	0.920	0.832	87.38	0.925	01:24	6.90
v6m	51,980,095	161.2	99.50	0.920	0.849	88.31	0.927	03:51	2.53
v6l	110,865,631	391.2	211.96	0.928	0.857	89.11	0.926	08:17	1.17
v6x	172,985,727	610.3	330.49	0.908	0.862	88.44	0.929	12:20	0.79
v7	36,501,466	103.2	71.37	0.950	0.863	90.44	0.942	05:44	1.70
v7x	70,807,066	188.1	135.56	0.936	0.876	90.50	0.945	08:42	1.12
v7-tiny	6,017,434	13.1	11.72	0.922	0.842	88.02	0.924	01:14	7.82
R-csp	52,487,060	119.0	100.62	0.927	0.865	89.49	0.943	06:24	1.52
R-csp-x	96,390,380	224.9	184.50	0.936	0.865	89.91	0.946	08:25	1.06
ours	3,843,771	8.1	7.74	0.929	0.833	87.84	0.922	01:07	8.66

Table 4. Comparison of detection results of the YOLO-Granada model.

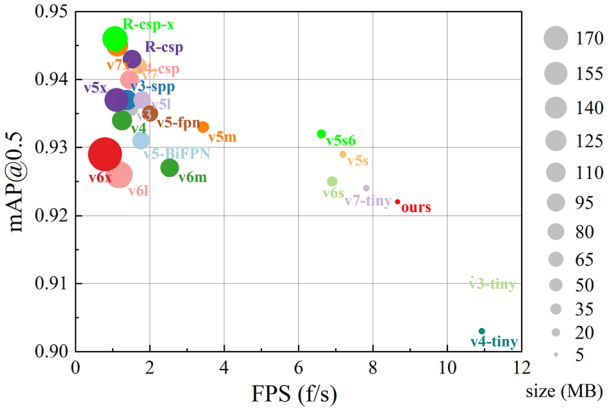

Figure 6. Comparison of detection results of the YOLO-Granada model.

As depicted in Fig. 6, the model's performance is primarily influenced by three key factors: FPS (frames per second), mAP@0.5 (average accuracy), and model size (parameters, floating-point operations, and model size in MB). The horizontal axis represents FPS, while the vertical axis represents mAP@0.5, with the size of each data point corresponding to the model's size (in MB).

Our model's size is the smallest among all compared models in Table 4, with parameters, floating-point operations, and model size compressed to 54.7%, 51.3%, and 56.3% of those in YOLOv5s, respectively. This significant reduction makes our model highly efficient for deployment on devices with limited computational resources, which is crucial for agricultural applications.

While emphasizing FPS, models like "v3-tiny" and "v4-tiny" exhibit higher speeds at 10.67 and 10.93 frames per second, respectively. However, their relatively lower accuracy, with mAP@0.5 values of 0.911 and 0.903, implies a potentially higher rate of missed or false detections. Although the mAP@0.5 value of 0.922 for the "ours" model is slightly lower than some other models, this trade-off is acceptable given the improved FPS of 8.66 frames per second.

The focus on minimizing model size while maintaining acceptable accuracy and speed ensures that YOLO-Granada is well-suited for real-time applications on resource-constrained devices. These trade-offs highlight the

effectiveness of YOLO-Granada in balancing accuracy, speed, and model size, making it suitable for practical deployments in agricultural monitoring and other similar tasks.

The choice of our model architecture involved a careful trade-off between performance and computational resources. While models with a larger number of parameters can potentially achieve better performance, they require significantly more computational resources and longer inference times. On the other hand, models with fewer parameters can achieve decent performance while being computationally efficient, making them more practical for deployment in real-world scenarios.

In our case, we aimed to strike a balance between model performance and computational resources by designing a model architecture that leverages a relatively small number of parameters while maintaining a high level of accuracy. Our model architecture includes several optimization techniques, including the ShuffleNet backbone network, which is highly efficient while maintaining high accuracy. And neural attention mechanism (CBAM), which helps to reduce the number of false positives generated by the model. By using these techniques, we were able to create a custom model that is highly accurate while remaining efficient.

The proposed model is highly efficient and suitable for deployment in resource-constrained environments, such as mobile devices or embedded systems. It is particularly well-suited for users who do not have access to a GPU, as it can operate efficiently on a CPU. Moreover, the experiments were conducted on low-cost CPUs with a main frequency of only 2.90 GHz. Given that many devices, including microcontrollers and cell phones, feature CPUs with similar or even higher main frequencies, our model can be seamlessly deployed on such inexpensive devices.

Table 5 shows the evaluation metrics for a multiclass object detection model across five class labels: "bud", "flower", "early-fruit", "mid-growth" and "ripe", and overall for all classes. The metrics include the precision (P), recall (R), and mean average precision at an IoU threshold of 0.5 (mAP@0.5).

The model achieved mAP@0.5 values of 0.872, 0.945, 0.91, 0.899 and 0.987 for "bud", "flower", "early-fruit", "mid-growth" and "ripe" respectively. The precision and recall scores for these classes ranged from 0.901-0.956 and 0.727-0.969, respectively.

Overall, the model achieved an mAP@0.5 of 0.922 with a precision of 0.929 and a recall of 0.833 across all the classes, highlighting its effectiveness in accurately detecting objects across different classes.

The loss values for both the training and validation sets, as shown in Fig. 7, undergo a rapid decrease initially, indicating effective early learning by the model. After this initial sharp decline, the loss gradually stabilizes at approximately 0.04, demonstrating that the model has reached a point of stable convergence. However, The gap between the training and validation loss, although small, suggests that the model is slightly better at modeling the training data compared to the unseen validation data. This is indicative of overfitting where the model starts to learn the noise and specific details of the training data rather than generalizing from it.

Class	Labels	*P*	R	F1	mAP@0.5
bud	216	0.918	0.727	81.14%	0.872
flower	293	0.956	0.846	89.76%	0.945
early-fruit	143	0.901	0.827	86.24%	0.910
mid-growth	294	0.925	0.796	85.57%	0.899
ripe	159	0.947	0.969	95.79%	0.987
all	1105	0.929	0.833	87.84%	0.922

Table 5. YOLO-Granada test results.

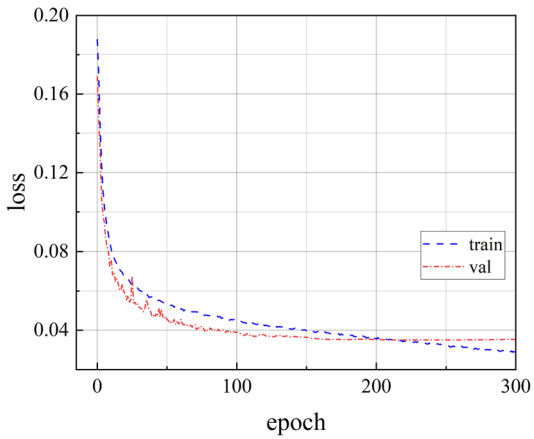

Figure 7. Training and validation loss curves over epochs.

To address this concern, we have implemented L2 regularization in our model, which helps to prevent overfitting by penalizing larger weights. This encourages the model to develop smaller, more generalizable weights that better capture the underlying patterns without fitting excessively to the training data's noise. Furthermore, we employed early stopping, configured to halt training after 270+ epochs if no improvement in validation loss is observed. These two strategies, early stopping and L2 regularization, are crucial in preventing the model from over-learning from the training set and ensuring that it remains effective when exposed to new, unseen data.

Results for cell phone terminals: weak and strong outcomes

To verify the model's scalability and practicality, it's deployed on an Android cell phone terminal. The application interface (Fig. 8) displays the "YOLO-Granada" model name, a still image, and function buttons. The "Image" button selects static images (Fig. 8), while the "Live" button enables real-time inspection via the cell phone camera (Fig. 9).

Figure 8. Test results for static images.

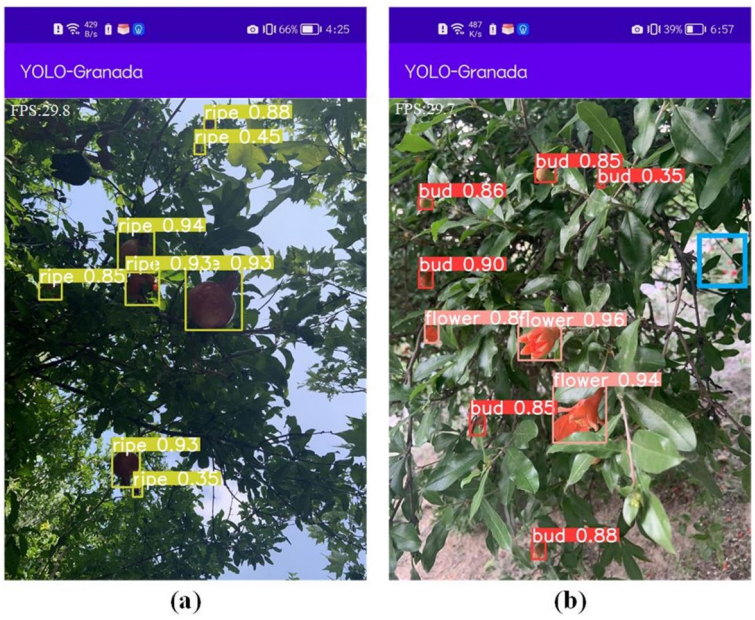

Figure 9. Test results for video streams.

Test results for static images

The application analyzes pomegranate ripeness in static images (Fig. 8). Users can detect pomegranates in captured or selected images, viewing model results for decision-making.

Figure 8 depicts model detection across five growth cycles. The colored box lines signify different cycles, with confidence levels denoting the likelihood of an object being in a bounding box and its location accuracy. The model excels, achieving even 0.98 confidence. Figure 8a shows two classifications with small, obscured targets. In Fig. 8b, the distant early-fruit is small and blurred. Figure 8c, d exhibit shading and mid-growth colors akin to background leaves, yet our model identifies well with confidence levels consistent near 0.97.

Test results for video streams

Our application extends beyond still image recognition to support real-time pomegranate ripeness analysis, dynamically detect pomegranates from video streams, and evaluate ripeness, as demonstrated in Fig. 9. Real-time detection relies on the efficient processing of consecutive frames. We adopted a multithreaded approach to fully utilize the multicore processing power of Android devices, ensuring FPS of 29.7 and 29.8, meeting task requirements.

In Fig. 9, the proposed model's detection in complex environments is showcased. Figure 9(a) displays results for scenes with varied pomegranate growth cycles, small targets, and occlusions. Colored box lines denote different growth cycles, with confidence levels ranging from 0.85 to 0.88, even amidst challenges. Notably, the model accurately distinguishes between similar objects, as seen in Fig. 9b, where it avoids misidentifying an ordinary flower in the blue box as part of the pomegranate's "flower" period. Additionally, Fig. 9b highlights the model's performance under insufficient sunlight conditions, accurately detecting small targets with confidence levels ranging from 0.35 to 0.45.

Weak outcomes

Although the proposed model demonstrates impressive detection results in complex environments, it should be noted that there were some misidentifications and underchecking of objects, as shown in Fig. 10. These issues indicate that our research should continue to improve the model's performance.

In summary, using Attentioned-ShuffleNet as a lightweight backbone network for YOLOv5 in object detection tasks may lead to false positives and false negatives due to the following reasons:

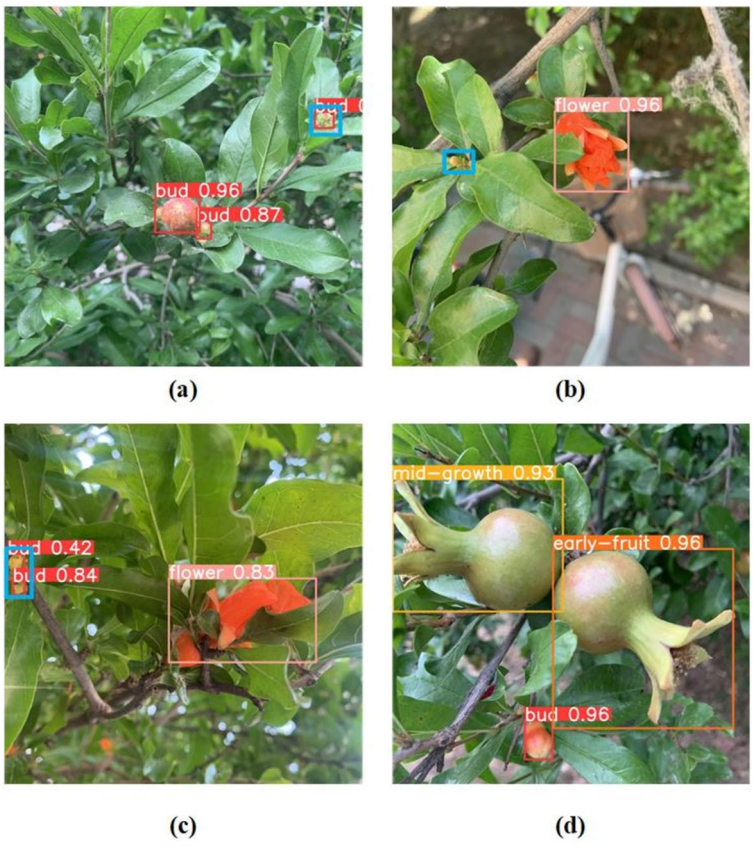

Figure 10. Pictures of poor model results.

1. Insufficient feature representation capability: ShuffleNetv2 is a lightweight convolutional neural network that reduces parameter and computational complexity by using group convolutions and channel shuffling. However, these techniques may decrease the network's feature representation capability, resulting in inaccurate object representation and leading to false positives and false negatives.
2. Excessive resolution down-sampling: YOLOv5 uses multiple down-sampling layers in the backbone network to decrease the input image resolution, which improves the receptive field and semantic representation of the feature map. However, excessive use of down-sampling layers in ShuffleNetv2 may result in a low-resolution feature map, leading to missed small objects and fine details and causing false negatives.
3. Non-aligned feature alignment problem: The channel shuffling technique used in ShuffleNetv2 causes a change in the channel order of the feature map, resulting in non-aligned feature maps. This affects the receptive field of the convolutional layers and the positional relationship of the pixels within the receptive field, thus impacting the object representation and localization accuracy, leading to false positives and false negatives.
4. Dataset characteristics: Additionally, the characteristics of the dataset can also affect the object detection results. For example, if the training set contains many poorly defined objects, it may be difficult to achieve high accuracy in object detection even with more complex networks.

Therefore, to address false positives and false negatives in object detection tasks, YOLOv5 with ShuffleNetv2 is used as the lightweight backbone network, and corresponding optimization and improvement should be performed in model design and dataset processing.

Overall, the proposed YOLO-Granada model demonstrated robustness and adaptability in detecting pomegranate objects in complex scenes with different growth cycles, small targets, and occlusions. The model's ability to detect pomegranate objects under suboptimal lighting conditions also holds great promise for its practical application in real-world scenarios.

Conclusion and future works

To implement the growth cycle detection of pomegranate on mobile, CPU, and other low-cost low-computing-power platforms. We propose a lightweight algorithm, YOLO-Granada, for pomegranate growth period detection based on the improved You Only Look Once version 5 (YOLOv5) algorithm. A lightweight Attentioned-ShuffleNet network is used as the backbone to extract pomegranate features, using ShuffleNetv2 to reduce the number and size of parameters of the model and thus increase the detection speed. Adding CBAM to the feature fusion process results in the use of the contribution factor of weights to enhancing attention and optimizing object detection accuracy. The improved network achieved an average accuracy of 0.922, which was less than 1% lower than that of the original model (YOLOv5s = 0.929) but also increased the speed and compressed the model size. Moreover, the detection speed is 17.3% faster than that of the original network. The parameters, floating-point operations, and model size of this network are compressed to 54.7%, 51.3%, and 56.3%, respectively, of those of the original network. In addition, the algorithm detects 8.66 images/second, achieving real-time results. The experimental results show that the improved algorithm effectively reduces the computational power requirement and speeds up detection, allowing real-time detection of pomegranates on an Android platform with low arithmetic power and low cost without relying on high computing power or expensive graphics processing units (GPUs). This study advances the deployment of object detection algorithms on the ground, which has positive implications for the promotion of smart agriculture and may provide ideas for similar work. In addition, the design process of the algorithm can provide a reference for the design of neural networks in agricultural applications.

The YOLO-Granada algorithm has many potential applications in the agricultural industry, and future research could focus on the following areas to enhance its effectiveness and applicability:

1. Improving Model Accuracy: Further research could be directed towards refining the model to improve its detection accuracy, particularly in complex environments with varying lighting conditions and occlusions. This could involve experimenting with different network architectures, attention mechanisms, and optimization techniques.
2. Expanding the Dataset with Different Pomegranate Varieties: To increase the robustness and generalizability of the model, it is essential to expand the dataset to include images of various pomegranate varieties. This will ensure that the model can accurately detect and analyze pomegranates across different types and growth stages.
3. Incorporating UAV-based Imagery: To better evaluate and enhance the model's performance in real-world agricultural monitoring scenarios, future work should include the collection and integration of UAV-based images and videos. Utilizing UAVs for data capture will provide aerial perspectives that are crucial for comprehensive farm monitoring, allowing for the detection of pomegranate growth stages over larger areas and from different angles.

Additionally, integrating the algorithm into a mobile app could make it more accessible for farmers to use. The algorithm could also have applications in domains beyond agriculture, such as surveillance and autonomous driving. Finally, the algorithm can be applied to other computer vision tasks, leading to further advancements in deep learning.

Supplementary information

This section provides detailed information about the sources and creation of the figures used in this manuscript. Figures 1 to 4 and 8 to 10 were generated using Microsoft Visio. Microsoft Visio is a diagramming and vector graphics application that is widely used for creating detailed diagrams and flowcharts. The version used for generating these figures was Microsoft Visio 2021. For more information on Microsoft Visio, please visit the official website: https://www.microsoft.com/en-us/microsoft-365/visio/flowchart-software. Figures 5 to 7 were created using Origin, a comprehensive data analysis and graphing software used in various scientific fields. Origin was chosen for its robust capabilities in producing high-quality graphs and its ease of use in data visualization. The version used was Origin 2021. For more details on Origin, please refer to the official website: https://www.originlab.com/. All figures were produced in accordance with the standard practices in their respective software, ensuring clarity and accuracy in the representation of the data.

Data availability

The full dataset for the current study is available in Mendeley Data, accessed at: https://data.mendeley.com/datasets/kgwsthf2w6/5.

Received: 2 March 2024; Accepted: 12 July 2024
Published online: 22 July 2024

References

1. Saparbekova, A., Kantureyeva, G., Kudasova, D., Konarbayeva, Z. & Latif, A. Potential of phenolic compounds from pomegranate (*Punica granatum* L.) by-product with significant antioxidant and therapeutic effects: A narrative review. *Saudi J. Biol. Sci.* **30**, 103553. https://doi.org/10.1016/j.sjbs.2022.103553 (2023).
2. Berger, B. A., Kriebel, R., Spalink, D. & Sytsma, K. J. Divergence times, historical biogeography, and shifts in speciation rates of myrtales. *Mol. Phylogenet. Evol.* **95**, 116–136. https://doi.org/10.1016/j.ympev.2015.10.001 (2016).
3. Mo, Y. *et al.* Pomegranate peel as a source of bioactive compounds: A mini review on their physiological functions. *Front. Nutr.* **9**, 887113. https://doi.org/10.3389/fnut.2022.887113 (2022).
4. Jiang, Z. *et al.* Pomegranate-like ato/sio2 microspheres for efficient microwave absorption in wide temperature spectrum. *J. Mater. Sci. Technol.* **174**, 195–203. https://doi.org/10.1016/j.jmst.2023.08.013 (2024).
5. Roopa Sowjanya, P. *et al.* Reference quality genome sequence of Indian pomegranate cv.'bhagawa' (*Punica granatum* L.). *Front. Plant Sci.* **13**, 947164. https://doi.org/10.3389/fpls.2022.947164 (2022).
6. Yang, X. *et al.* The nutritional and bioactive components, potential health function and comprehensive utilization of pomegranate: A review. *Food Rev. Int.* **39**, 6420–6446. https://doi.org/10.1080/87559129.2022.2110260 (2023).
7. Tang, Y. *et al.* Recognition and localization methods for vision-based fruit picking robots: A review. *Front. Plant Sci.* **11**, 510. https://doi.org/10.3389/fpls.2020.00510 (2020).
8. Girshick, R., Donahue, J., Darrell, T. & Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE conference on computer vision and pattern recognition*, pp. 580–587. https://doi.org/10.1109/CVPR.2014.81 (2014).
9. Girshick, R. Fast r-cnn. In *2015 IEEE International conference on computer vision (ICCV)*, 1440–1448, https://doi.org/10.1109/ICCV.2015.169 (2015).
10. Ren, S., He, K., Girshick, R. & Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Adv. Neural Inf. Process. Syst.* **28** (2015).
11. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on computer vision and pattern recognition (CVPR)*, pp. 779–788. https://doi.org/10.1109/CVPR.2016.91 (2016).
12. Redmon, J. & Farhadi, A. Yolo9000: Better, faster, stronger. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, 6517–6525. https://doi.org/10.1109/CVPR.2017.690 (2017).
13. Redmon, J. & Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint*arXiv:1804.02767. *https://doi.org/10.48550/arXiv.1804.02767 (2018).*
14. Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint*arXiv:2004.10934. *https://doi.org/10.48550/arXiv.2004.10934 (2020).*
15. Li, C. *et al.* Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint*arXiv:2209.02976. *https://doi.org/10.48550/arXiv.2209.02976 (2022).*
16. Wang, C., Bochkovskiy, A. & Liao, H. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arxiv 2022. *arXiv preprint*arXiv:2207.02696. *https://doi.org/10.48550/arXiv.2207.02696 (2022).*
17. Badgujar, C. M., Poulose, A. & Gan, H. Agricultural object detection with you only look once (yolo) algorithm: A bibliometric and systematic literature review. *Comput. Electron. Agric.* **223**, 109090. https://doi.org/10.1016/j.compag.2024.109090 (2024).
18. Yang, M., Yuan, W. & Xu, G. Yolox target detection model can identify and classify several types of tea buds with similar characteristics. *Sci. Rep.* **14**, 2855. https://doi.org/10.1038/s41598-024-53498-y (2024).
19. Tian, Y. *et al.* Apple detection during different growth stages in orchards using the improved yolo-v3 model. *Comput. Electron. Agric.* **157**, 417–426. https://doi.org/10.1016/j.compag.2019.01.012 (2019).
20. Khan, A. *et al.* Tomato maturity recognition with convolutional transformers. *Sci. Rep.* **13**, 22885. https://doi.org/10.1038/s41598-023-50129-w (2023).
21. Wang, X., Vladislav, Z., Viktor, O., Wu, Z. & Zhao, M. Online recognition and yield estimation of tomato in plant factory based on yolov3. *Sci. Rep.* **12**, 8686. https://doi.org/10.1038/s41598-022-12732-1 (2022).
22. Zheng, Z. *et al.* A method of green citrus detection in natural environments using a deep convolutional neural network. *Front. Plant Sci.* **12**, 705737. https://doi.org/10.3389/fpls.2021.705737 (2021).
23. Tang, Y., Zhou, H., Wang, H. & Zhang, Y. Fruit detection and positioning technology for a camellia oleifera c. abel orchard based on improved yolov4-tiny model and binocular stereo vision. *Expert Syst. Appl.***211**, 118573. https://doi.org/10.1016/j.eswa.2022.118573 (2023).
24. Dong, Q., Sun, L., Han, T., Cai, M. & Gao, C. Pestlite: A novel yolo-based deep learning technique for crop pest detection. *Agriculture*, **14**. https://doi.org/10.3390/agriculture14020228 (2024).
25. Lin, Y., Huang, Z., Liang, Y., Liu, Y. & Jiang, W. Ag-yolo: A rapid citrus fruit detection algorithm with global context fusion. *Agriculture* **14**. https://doi.org/10.3390/agriculture14010114 (2024).
26. Omer, S. M., Ghafoor, K. Z. & Askar, S. K. Lightweight improved yolov5 model for cucumber leaf disease and pest detection based on deep learning. *SIViP* **18**, 1329–1342. https://doi.org/10.1007/s11760-023-02865-9 (2024).
27. Lan, M. *et al.* Rice-yolo: In-field rice spike detection based on improved yolov5 and drone images. *Agronomy* **14**. https://doi.org/10.3390/agronomy14040836 (2024).

28. Zhang, P. & Li, D. Epsa-yolo-v5s: A novel method for detecting the survival rate of rapeseed in a plant factory based on multiple guarantee mechanisms. *Comput. Electron. Agric.* **193**, 106714. https://doi.org/10.1016/j.compag.2022.106714 (2022).
29. Wang, Z., Jin, L., Wang, S. & Xu, H. Apple stem/calyx real-time recognition using yolo-v5 algorithm for fruit automatic loading system. *Postharvest Biol. Technol.* **185**, 111808. https://doi.org/10.1016/j.postharvbio.2021.111808 (2022).
30. Zhang, J., Tian, M., Yang, Z., Li, J. & Zhao, L. An improved target detection method based on yolov5 in natural orchard environments. *Comput. Electron. Agric.* **219**, 108780. https://doi.org/10.1016/j.compag.2024.108780 (2024).
31. Vasumathi, M. & Kamarasan, M. An lstm based cnn model for pomegranate fruit classification with weight optimization using dragonfly technique. *Indian J. Comput. Sci. Eng.* **12**, 371–384. https://doi.org/10.21817/indjcse/2021/v12i2/211202051 (2021).
32. Mitkal, P. S. & Jagadale, A. Grading of pomegranate fruit using cnn. *Age***3**. https://doi.org/10.48175/IJARSCT-13039 (2023).
33. Ma, N., Zhang, X., Zheng, H.-T. & Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131. https://doi.org/10.48550/arXiv.1807.11164 (2018).
34. Zhang, X., Zhou, X., Lin, M. & Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6848–6856. https://doi.org/10.48550/arXiv.1707.01083 (2018).
35. Mnih, V., Heess, N., Graves, A. *et al.* Recurrent models of visual attention. *Adv. Neural Inf. Process. Syst.* **27** (2014).
36. Woo, S., Park, J., Lee, J.-Y. & Kweon, I. S. Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, 3–19. https://doi.org/10.48550/arXiv.1807.06521 (2018).
37. Zhao, J., Almodfer, R., Wu, X. & Wang, X. A dataset of pomegranate growth stages for machine learning-based monitoring and analysis. *Data Brief* **50**, 109468. https://doi.org/10.1016/j.dib.2023.109468 (2023).
38. Liu, Y., Shao, Z., Teng, Y. & Hoffmann, N. Nam: Normalization-based attention module. *arXiv preprint*arXiv:2111.12419. *https://doi.org/10.48550/arXiv.2111.12419 (2021).*
39. Liu, Y., Shao, Z. & Hoffmann, N. Global attention mechanism: Retain information to enhance channel-spatial interactions. *arXiv preprint*arXiv:2112.05561. *https://doi.org/10.48550/arXiv.2112.05561 (2021).*
40. Hu, J., Shen, L., Albanie, S., Sun, G. & Wu, E. Squeeze-and-excitation networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **42**, 2011–2023. https://doi.org/10.1109/TPAMI.2019.2913372 (2020).
41. Yang, L., Zhang, R.-Y., Li, L. & Xie, X. Simam: A simple, parameter-free attention module for convolutional neural networks. In *International Conference on Machine Learning*, pp. 11863–11874 (PMLR, 2021).

Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Author contributions

J.Z.: Conceptualization, Methodology, Software, Writing - original draft. R.A.: Conceptualization, Funding acquisition, Resources, Supervision, Writing - review & editing. C.D.: Restructuring the manuscript, Significant contributions to the organization and logical flow. Y.L.: Conceptualization, Methodology, Deployments, Writing - original draft. M.M.: Literature review, Data analysis, Writing - original draft. D.G. , Y.F. , X.W(Xiaoying Wu) and X.W.(Xinfa Wang): Data curation, Investigation. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to R.A.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.