An optimized lightweight real-time detection network model for IoT embedded devices.

CHEN, R., WANG, P., LIN, B., WANG, L., ZENG, X., HU, X., YUAN, J., LI, J., REN, J. and ZHAO, H.

2025

scientific reports

Check for updates

OPEN

An optimized lightweight real-time detection network model for IoT embedded devices

Rongjun Chen[1,2], Peixian Wang[1], Binfan Lin[1], Leijun Wang[1], Xianxian Zeng[1,2,3], Xianglei Hu[1], Jun Yuan[1], Jiawen Li[1]✉, Jinchang Ren[1,4]✉ & Huimin Zhao[1]✉

With the rapid development of Internet of Things (IoT) technology, embedded devices in various computer vision scenarios can realize real-time target detection and recognition tasks, such as intelligent manufacturing, automatic driving, smart home, and so on. YOLOv8, as an advanced deep learning model in the field of target detection, has attracted much attention for its excellent detection speed, high precision, and multi-task processing capability. However, since IoT embedded devices typically own limited computing resources, direct deployment of YOLOv8 is a big challenge, especially for real-time detection tasks. To address this vital issue, this work proposes and deploys an optimized lightweight real-time detection network model that well-suits for IoT embedded devices, denoted as FRYOLO. To evaluate its performance, a case study based on real-time fresh and defective fruit detection in the production line is performed. Characterized by low training cost and high detection performance, this model accurately detects various types of fruits and their states, as the experimental results show that FRYOLO achieves 84.7% in recall and 89.0% in mean Average Precision (mAP), along with a precision of 92.5%. In addition, it provides a detection frame rate of up to 33 Frames Per Second (FPS), satisfying the real-time requirement. Finally, an intelligent production line system based on FRYOLO is implemented, which not only provides robust technical support for the efficient operation of fruit production processes but also demonstrates the availability of the proposed network model in practical IoT applications.

Keywords YOLOv8, Neural networks, Embedded device, IoT, Computer vision

The Internet of Things (IoT) bridges the physical and digital worlds, connecting everything through embedded devices, sensors, and intelligent systems. The massive amounts of the devices require efficient processing and analysis to enable real-time monitoring, predictive maintenance, and automated control[1]. Despite its powerful and fast data processing capabilities, traditional cloud computing may need more network latency due to transmitting data over the network to the cloud, making it unsuitable for applications with high real-time requirements. In contrast, edge computing processes data on embedded devices close to the data source, significantly reducing the need for data transmission and lowering latency, which is more suitable for applications with high real-time requirements. Usually, in an IoT architecture, edge computing devices play a crucial role, where edge computing is a technology that pushes computing tasks and data storage from the centralized cloud to the network's edge, as well as data processing and analysis in IoT embedded devices. This approach can significantly reduce data transmission latency, improve system response speed, and reduce computing pressure on the cloud. Consequently, with the rapid development of IoT technology, embedded devices in various computer vision scenarios can realize target detection and recognition tasks, such as intelligent manufacturing, automatic driving, smart home, and so on[2].

Recently, in deep learning, the development of the Convolutional Neural Network (CNN) has endowed machines with capabilities to detect objects automatically, and the deep learning-based object detection approaches can be classified into two categories: two-stage and single-stage. The two-stage networks include R-CNN[3], fast R-CNN[4], and faster R-CNN[5], which usually operate in two main phases: generate region proposals

[1]School of Computer Science, Guangdong Polytechnic Normal University, Guangzhou 510665, China. [2]Guangdong Provincial Key Laboratory of Intellectual Property and Big Data, Guangdong Polytechnic Normal University, Guangzhou 510665, China. [3]Guangdong Provincial Key Laboratory of Big Data Computing, The Chinese University of Hong Kong, Shenzhen (CUHK-Shenzhen), Shenzhen 518000, China. [4]National Subsea Centre, Robert Gordon University, Aberdeen AB21 0BH, UK. ✉email: lijiawen@gpnu.edu.cn; jinchang.ren@ieee.org; zhaohuimin@gpnu.edu.cn

at first, then classify these regions and perform bounding box regression. Although they are recognized for high detection accuracy, their multi-stage processing results in relatively slower detection speeds, making them less suitable for real-time scenarios. On the other hand, the single-stage networks adopt a more direct approach. They bypass the region proposal step and directly predict object-bound boxes and classes from the input image. Such a simplified process essentially improves detection speed. While it does sacrifice some accuracy, single-stage detection networks such as the You Only Look Once (YOLO) series[6–10] and Single Shot MultiBox Detector (SSD)[11] have shown great potential in applications demanding high-efficiency processing. Despite the outstanding performance of deep learning-based approaches, their extensive parameters and computational costs limit their deployment on resource-constrained IoT embedded devices. This difficulty is particularly pressing given the need for computer vision scenarios such as efficient fruit defect detection in the canned fruit production line, as drawn in Fig. 1. Hence, the current deep learning-based models, though powerful, are complex in structure, making it challenging to deploy for finding the subtle differences and quality states of objects in diverse IoT applications.

To address the current limitations, this work proposes and deploys an optimized lightweight YOLOv8 model well-suited for IoT embedded devices, denoted as FRYOLO, which enhances the robustness and optimizes the structural design to reduce the network complexity substantially. More importantly, to evaluate its performance for practical IoT applications, a case study based on real-time fresh and defective fruit detection in the automatic production line is conducted. In particular, the main contributions are summarized as follows:

- For method validation, this work establishes an image dataset containing two states (fresh and defective) of five fruits (apple, mango, orange, pear, and banana), which provides a high-quality data foundation for developing, optimizing, and applying the proposed method to advance the deployment in IoT embedded devices.
- To address the limitations of IoT embedded devices in computational cost for real-time detection, this work proposes FRYOLO by improving the YOLOv8, where the Distribution Focal Loss (DFL) is assigned to the post-processing stage outside the model to reduce its complexity. This operation simplifies the architecture and allows the DFL's advantages to be realized without adding inference burdens. Moreover, to accelerate the speed, the confidence scores of the output branches are summed. Based on that, the threshold screening mechanism in the post-processing stage is optimized, which enables the model to maintain high accuracy while completing the inference process rapidly, enhancing the operational efficiency in resource-constrained IoT embedded devices.
- The proposed network model demonstrates its remarkable performances, achieving a mean Average Precision (mAP) of 98.3% and a precision of 93.3%. Meanwhile, it fits between actual fruit and detection box sizes, i.e., recall exceeds 96.5%, verifying its high localization and size estimation precision. Based on that, an intelligent production line system is established, providing technical support for IoT design.

The rest of this work is organized as follows: Sect. 2 reviews the related works about fruit defect detection since the case study is in this field. Section 3 describes the experimental dataset and the FRYOLO model. Section 4 presents the performance evaluation and the intelligent production line system designed, along with a comparative study of existing methods. Finally, Sect. 5 summarizes this work.

Related work

To accomplish real-time fruit defect detection in the production line, it is necessary to detect the fruits' states at first, and the traditional methods have relied on feature extraction. Usually, it begins with a detailed analysis of the multidimensional characteristics of fruit images, including but not limited to size, shape, color distribution, and texture details. By integrating one or more features, a classifier is established to achieve detection tasks. For example, Ireri et al.[12] developed a computer vision system for grading tomatoes, utilizing Red Green Blue (RGB) images to detect calyx and stalk scars by applying a histogram threshold method based on the g-r values of these

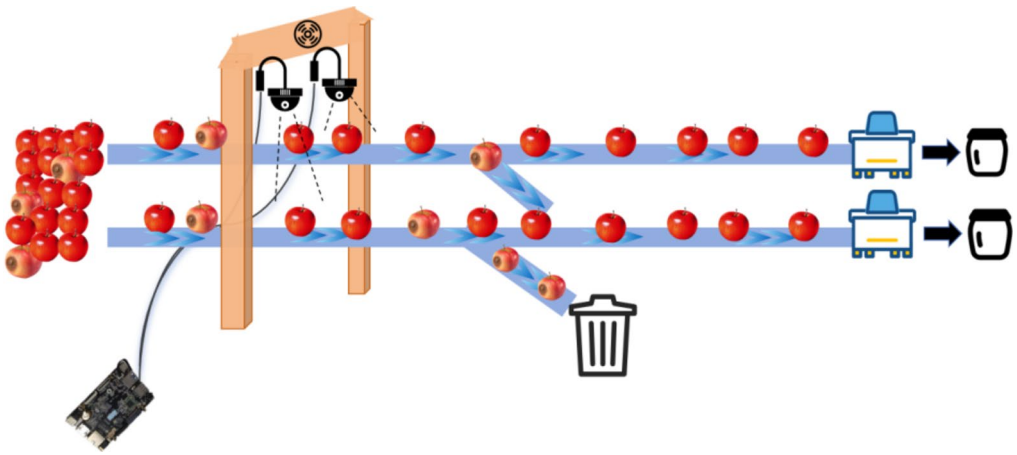

Fig. 1. A typical automatic canned fruit production line system in the field of IoT.

regions, an average detection accuracy of 0.9515 was obtained. In addition, defect detection was performed using pixel values in the Lab color space, a more accurate color space that uses three values (L, a, and b) to specify colors, and a Radial Basis Function (RBF) kernel Support Vector Machine (SVM) classifier, resulting in an overall accuracy of 0.989 after validation. Liu et al.[13] proposed another computer vision method based on the elliptical boundary model, converting images from the RGB to the YCbCr space, a family of color spaces used as a part of the color image pipeline in video and digital photography. They sampled the grapefruit peel areas from template images and computed the color regions in the Cr-Cb coordinates. The implicit quadratic polynomial of the elliptical boundary model was then fitted employing Ordinary Least Squares (OLS), and the images were segmented in the Cr-Cb color space to distinguish between immature green fruits, mature green fruits, and mature fruits with different color characteristics. Parraga-Arotinco et al.[14] designed a system based on computer vision and machine learning techniques to address the issue of automatic blueberry sorting and improve sorting efficiency. This system classified the degree of ripeness of blueberries from 10 mm to 14 mm and above, by mimicking size sorting and Hue Saturation Value (HSV) color detection. While traditional methods are relatively mature, their detection accuracy closely depends on the features and the classifiers, which incur slow detection speeds and poor applicability in complex environments.

On the other side, with advances in neural networks, some detection methods have been developed. For instance, Prakash et al.[15] designed an intelligent fruit detection system based on CNN and heterogeneous flow with bilinear pooling. Gao et al.[16] addressed the issue of reduced fruit-picking efficiency due to multiple branch occlusions in orchards by proposing a multi-class apple detection method based on an improved Faster R-CNN, achieving an mAP of 87.9% and an average detection time per image of 0.241 s. Zhai et al.[17] presented a method employing YOLOv5 combined with an attention mechanism and bidirectional feature pyramid network (BiFPN) to perform the automated detection of blueberry ripeness. By integrating the attention mechanism, the unnecessary information is removed, and the YOLOv5-SE + BiFPN model conducted an mAP of 90.5%, with a recall rate and precision of 88.5% and 88.4%, respectively. Wang et al.[18] applied the Detail-Semantics Enhancement (DSE)-YOLO model to detect multi-stage strawberries. By incorporating the DSE into the backbone to extract various horizontal and vertical features, they accomplished an mAP of 86.58% and an F1 score of 81.59%. To improve the detection accuracy in complex detection backgrounds, Yang et al.[19] developed an approach named BCo-YOLOv5, where they added the Bidirectional Cross-attention Mechanism (BCAM) between the backbone and neck to construct deeper positional feature relationships, enhancing its ability to detect fruit in the images with obtaining an mAP of 97.7%.

The above studies exhibit good results in fruit detection tasks. Nevertheless, their computational complexities are too high to be deployed on IoT embedded devices for practical applications such as real-time fruit defect detection in the production line. With the development of edge computing, more and more lightweight network model solutions have been proposed, making it possible to deploy computationally intensive object detection models on embedded devices. For example, Rastegari et al.[20] proposed a method to quantize weights to lower bit widths, while Qu et al.[21] reduced the redundancy of quantization bit widths by assigning different bit widths to different weight groups. In another work, Qu et al.[22] introduced the DRESS method, which reduces redundancy among multiple subnetworks by sharing weights and architectures for lowering model training parameters. Khaliq et al.[23] designed a systematic quality scalable design methodology, which approximates filter values in a deep learning model with 3-bit encoding, reducing the storage requirements for model parameters. Hence, converting numbers to canonical sign digit (CSD) representation minimizes the number of partial products required in multiplication operations, making it more appropriate for IoT embedded devices.

In short, deep learning-based models usually focus on coarse-grained fruit datasets in ideal environments, resulting in weak generalization capabilities that confine their abilities to detect multi-target fruits in complex environments. Such limitations need to be addressed in an optimized lightweight way, which is the motivation of this work.

Method

Dataset

Considering that oranges, pears, mangoes, bananas, and apples are popular canned fruits in the production line system, the cast study selects them as samples in the dataset. In order to thoroughly investigate the characteristics of these fruits in different states, each has been categorized into fresh and defective, resulting in a classification containing ten categories. This task benefits the model's ability to conduct quality control in practical applications. Moreover, to build up a high-quality dataset, in total of 1020 fruit images have been collected, which exhibit diverse backgrounds ranging from simple solid colors to complex environments. In addition, the images containing single and multiple fruits are included to simulate real-world scenarios. LabelImg, an annotation tool, is employed for data annotation, where each fruit in every image has been labeled with its category and position. This step is important for subsequent model training, helping the model learn accurate feature representations. Figure 2 shows fruit sample images in the dataset.

To avoid the imbalance issue in machine learning, the number of each type remains between 90 and 115, as listed in Table 1. Meanwhile, to ensure the standardization of the deep learning training process, the hold-out method is adopted, dividing the dataset into a training set (718 images), a validation set (203 images), and a test set (99 images) in a 7:2:1 ratio, which guarantees the independence of the training set and test set. To facilitate reproducible research, the dataset related to this work is freely available at https://github.com/wpx-1/fruit-yolo/tree/master.

Next, the statistics shown in Figure 3 offer an in-depth analysis of data properties. Figure 3(a) draws the distribution of instance counts across various categories. Here, the mango accounts for most cases, attributed to the numerous images having two or more mangoes. Figure 3(b) depicts the distribution of annotation box sizes and their variation in quantity across various categories. It helps to investigate the size variations of target

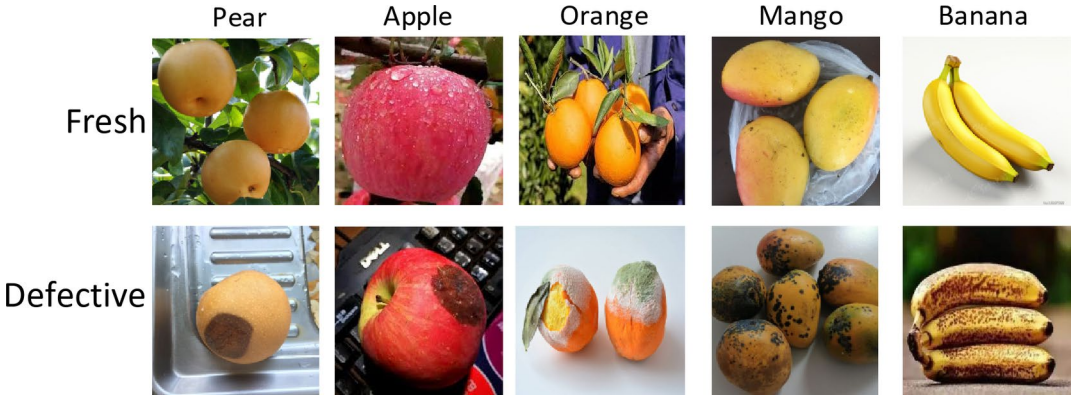

Fig. 2. Fruit sample images in the dataset.

Type	Label	Number
Fresh apple	Apple	101
Fresh mango	Mango	100
Fresh orange	Orange	98
Fresh pear	Pear	115
Fresh banana	Banana	111
Defective apple	bad_apple	97
Defective mango	bad_mango	95
Defective orange	bad_orange	101
Defective pear	bad_pear	90
Defective banana	bad_banana	112

Table 1. The number of each type in the fruit dataset.

objects within images and optimize the model's detection ability for objects of varying sizes. Figure 3(c) displays a scatter plot that illustrates the distribution of annotation box centroids relative to the entire image's position, which comprehends the spatial layout of target objects within images. Finally, Figure 3(d) analyzes the aspect ratio of target objects relative to the entire image. It not only indicates the diversity in the shapes of target fruits but also reveals morphological differences among various categories.

Overall architecture

YOLO has garnered widespread adoption in computer vision due to its efficient performance, rooted in its ingenious fusion of the deep feature extraction capabilities of CNN with the instantaneous requirements of object detection[24]. Then, YOLO adopts these constructed feature maps to promptly generate a series of candidate bounding boxes, assigning scores to potential objects within each box to quantify their confidence as belonging to specific categories. During the prediction stage, to enhance localization accuracy and mitigate the redundancy arising from multiple overlapping or proximate candidate boxes, YOLO includes the Non-Maximum Suppression (NMS) algorithm as a post-processing step[25]. NMS compares the scores of adjacent candidate boxes, retains the box with the highest score within local regions, and discards others with lower scores, addressing the issue of duplicate detection and ensuring a refined and accurate output for object detection.

As the YOLO advances, YOLOv8 is at the forefront of this evolution. It includes three vital parameters: depth, width, and max_channels, which can regulate the model's complexity. The depth means the number of layers in the network, i.e., its capacity. The width governs the breadth of each network layer, specifying the number of neurons within each layer. The max_channels allows for dynamic adjustment of the network's channel count, offering flexibility to the model. Based on these parameters, YOLOv8 has been designed into different versions, catering to various needs and computational constraints. Table 2 lists the depth, width, and max_channels for different versions.

In Table 2, YOLOv8n stands out with its extremely lightweight design, exhibiting efficient and rapid detection capabilities through minimal network depth and the least number of parameters. It is particularly suitable for limited computational resources and real-time requirements[26]. While, YOLOv8x utilizes a network structure that pursues high accuracy with heightened demands on computational resources, including more substantial computing power, larger memory space, and more extended training times. Hence, it supports applications that require high detection performances, such as medical image analysis[27] and high-precision industrial automation[28]. In practical IoT applications, the key lies in selecting the appropriate version of the YOLOv8 model based on specific task requirements and available computational resources.

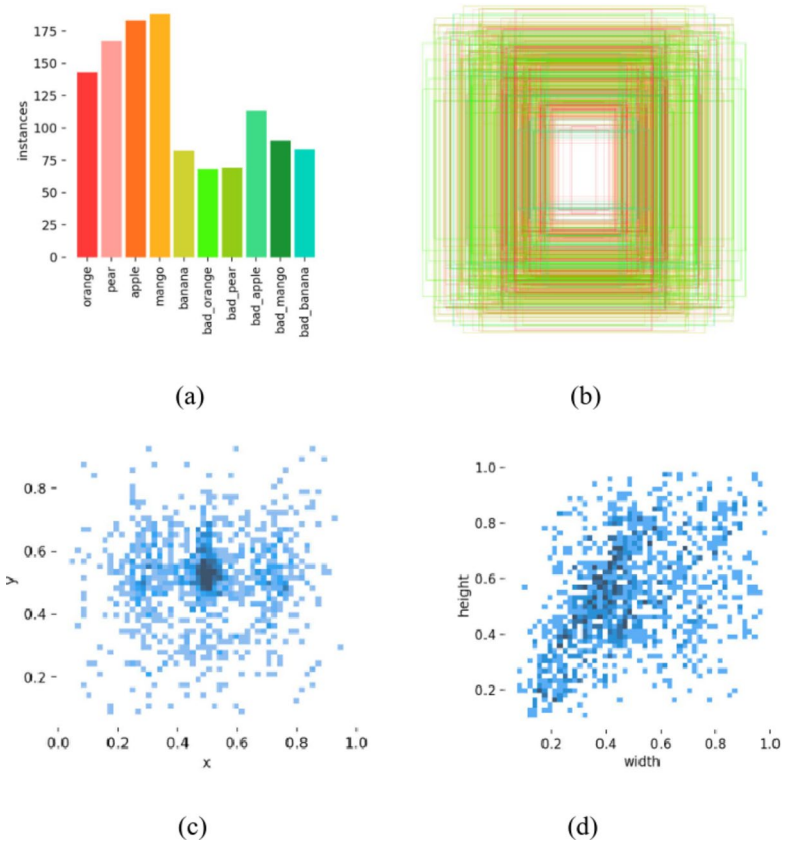

Fig. 3. Data statistics of the fruit dataset: (**a**) The distribution of instance counts across various categories; (**b**) The distribution of annotation box sizes; (**c**) The scatter plot that illustrates the distribution of annotation box centroids relative to the entire image's position; (**d**) The aspect ratio of target objects relative to the entire image.

Version	Depth	Width	Max_channels
YOLOv8n	0.33	0.25	1024
YOLOv8s	0.33	0.50	1024
YOLOv8m	0.67	0.75	768
YOLOv8l	1.00	1.00	512
YOLOv8x	1.00	1.25	512

Table 2. The parameters of depth, width, and max_channels for different versions of YOLOv8.

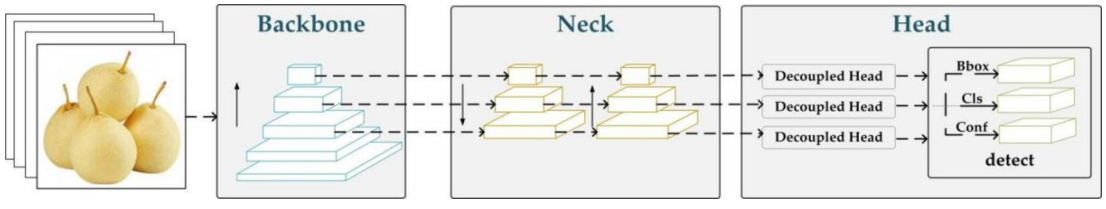

Fig. 4. The overall architecture of the proposed FRYOLO network model.

Compared to the standard YOLOv8, FRYOLO adds a layer, an output branch that calculates the sum of execution degrees, to improve model inference speed without introducing additional parameters. It also assigns the DFL to the post-processing stage outside the model to reduce complexity and enhance its operational speed on embedded devices. Through these two modifications, FRYOLO maintains high accuracy while improving the inference speed of the model. For better illustration, its overall architecture is displayed in Fig. 4. Here, the

Layer	Module	Input	Repeat	Output
1	CBS	640×640×3	1	320×320×64
2	CBS	320×320×64	1	160×160×128
3	C2f	160×160×128	3	160×160×128
4	CBS	160×160×128	1	80×80×256
5	C2f	80×80×256	6	80×80×512
6	CBS	80×80×512	1	40×40×512
7	C2f	40×40×512	6	40×40×512
8	CBS	40×40×512	1	20×20×1024
9	C2f	20×20×1024	3	20×20×1024
10	SPPF	20×20×1024	1	20×20×1024

Table 3. The composition of the backbone network modules.

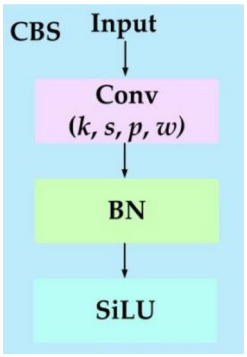

Fig. 5. The CBS module in the proposed FRYOLO network model.

backbone is responsible for feature extraction, the neck selects the features, and the head predicts the location and confidence score of the fresh or defective fruits.

Backbone

When designing a fruit defect detection network model, especially for real-time applications, maintaining a lightweight network structure while ensuring remarkable performance is vital. To this end, the proposed method adopts the feature extraction network similar to the lightweight model CSPDarknet, consisting of CBS, C2f, and Spatial Pyramid Pooling Fast (SPPF) modules, as listed in Table 3.

The CBS module contains Convolution (Conv), Batch Normalization (BN), and Sigmoid Linear Unit (SiLU) activation function, as depicted in Fig. 5. The Conv layer performs convolutional processing on the input images and increases the depth of the data channels, generating a solid foundation for feature extraction (1):

$$w' = \frac{(w + 2p - k)}{s} + 1 \tag{1}$$

where w is the input size, p denotes the padding for the convolution kernel, k indicates the size of the convolution kernel, s means the stride, and w' represents the output size.

Through a unified convolutional kernel size strategy, the CBS module reduces the number of parameters and computational burden while providing the weight-sharing mechanism, enabling effective feature extraction from various locations in the image. Subsequently, BN standardizes the distribution of each mini-batch of data, helping the stability of model training and improving generalization capabilities (2):

$$y = \frac{x - E[x]}{\sqrt{Var[x] + ?}} * \gamma + \beta \tag{2}$$

where y represents the output, x means the input, $E[x]$ is the mean of x, $Var[x]$ denotes the variance of x, ε is a default value set to 10^{-5}, γ is 1, and β is 0.

Lastly, the SiLU activation function helps neural networks to fit nonlinear data. This nonlinearity is beneficial for modeling complex relationships in data (3):

$$\mathrm{SiLU}(x) = \frac{1}{1 + \exp(-x)} \tag{3}$$

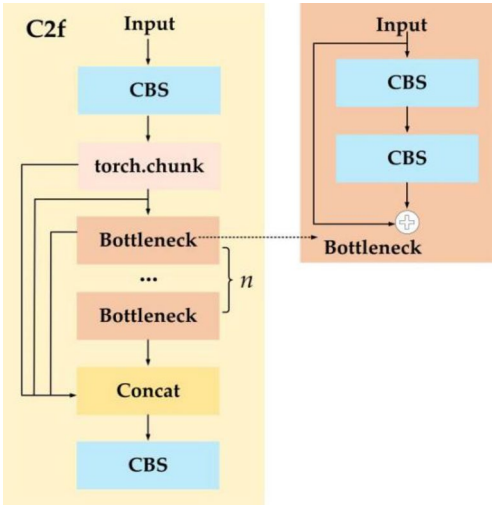

Fig. 6. The C2f module and its bottleneck in the proposed FRYOLO network model.

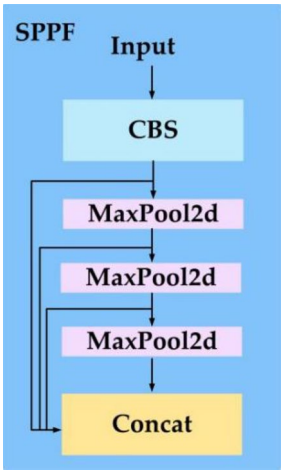

Fig. 7. The SPPF module in the proposed FRYOLO network model.

The C2f module integrates the bifurcation concept of CSPNet and the design philosophy of residual structures, achieving the transformation of information from the channel dimension to the spatial dimension within the network. This module is designed to enhance the network's ability to capture features of objects at different scales. Specifically, the C2f module first applies a 1×1 convolutional layer to process the input feature map, then divides the processed feature map into two parts. The first part serves as one output branch directly, while the second part undergoes processing through several bottleneck modules. In these bottleneck modules, residual connections directly transmit the input signal and learn the mapping relationship between the input and output differences. Finally, these two branches are concatenated, allowing the model to maintain its lightweight nature while acquiring a diverse gradient flow. The structure of the C2f module and its bottleneck are depicted in Fig. 6.

The SPPF module combines large kernel convolution and non-dilated convolution, where the large kernel convolution increases the receptive field of the model, and the non-dilated convolution maintains the spatial resolution of the convolution operation without increasing the number of parameters. Based on that, SPPF enhances the performance while maintaining low computational overhead. Specifically, when a feature map enters the SPPF module, it first processes a 1×1 convolution, then pools at different levels, generating a set of feature maps covering multi-scale information from coarse to detailed. Subsequently, SPPF concatenates these multi-scale feature maps, forming a feature vector containing multi-scale information. As the output size is fixed, the model can flexibly adapt to input images of different sizes, improving its performance on tasks with high sensitivity to scale variations, such as fine-grained classification and multi-scale object detection. Its structure is illustrated in Fig. 7.

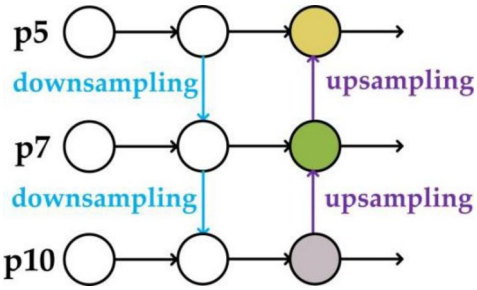

Fig. 8. The PAFPN applied in the neck receives inputs from the backbone's fifth, seventh, and tenth layers, denoted as p5, p7, and p10.

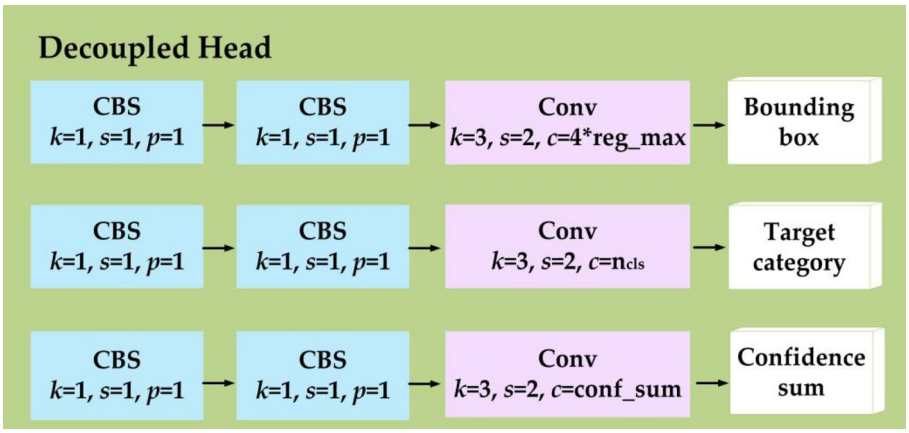

Fig. 9. The decoupled head with three outputs: bounding box, target category, and confidence sum.

Neck

Since a single-scale feature map has limited representation capabilities for objects of different sizes, leading to degraded performance when detecting small or huge objects, Pyramid Feature Network (PFN) has been developed. It aims to enhance the model's multi-scale object detection ability by constructing a feature cascade and multi-scale fusion mechanism[29]. Building upon this foundation, the Path Aggregation Feature Pyramid Network (PAFPN) adds a bottom-up pyramid after the FPN, which contains a path aggregation approach that aggregates shallow feature maps (low resolution but weaker semantic information) with deep feature maps (high resolution but rich semantic information). In addition, PAFPN enhances the multi-scale feature representation capability by transmitting feature information along specific paths and propagating the robust localization features from lower layers upwards, which helps to select the features for fresh and defective fruit detection.

The backbone outputs three different sizes of feature maps at the fifth (p5), seventh (p7), and tenth layers (p10), which are utilized for multi-scale feature fusion in the neck. Then, the FRYOLO uses the PAFPN to build feature pyramids across different layers, enabling the model to perform fruit detection at various scales, as shown in Fig. 8. It achieves this aim by fusing low-level detailed features with high-level semantic features through upsampling and downsampling, generating an enriched feature representation. This pyramid network and the fusion of features from different levels effectively extract and integrate multi-scale features, allowing the model to better detect fresh or defective fruits with various scales and sizes in complex scenes.

Head

As classification and regression tasks are coupled, which leads to interference when the model processes different tasks, the $80 \times 80 \times 256$, $40 \times 40 \times 512$, and $20 \times 20 \times 512$ feature maps from various levels of the Neck are input into decoupled head for classification and regression, respectively, as depicted in Fig. 9. The regression head calculates the positional offsets between the predicted bounding boxes and the ground truth boxes. Then, it feeds the offsets into the regression head for loss calculation, outputting a four-dimensional vector denoting the target bounding box's top-left and bottom-right coordinates (x, y). The classification head performs Region of Interest (RoI) pooling and convolution operations on each candidate box extracted by anchor-free to obtain a classifier output tensor, where the value at each position indicates the probability that the candidate box belongs to each category. Consequently, the maximum suppression method filters out the final detection results.

Furthermore, considering the real-world scenarios and real-time applications on IoT embedded devices, it is necessary to fully exploit the acceleration of the Neural Processing Unit (NPU). One solution is to relocate the model's DFL structure to the external part, i.e., the post-processing of the inference process for optimizing the

Hardware/software	Parameter/version
CPU	Intel Xeon w7-3445
GPU	GeForce RTX 4090-24 G
RAM	128 GB
Hard disk	8 TB
Embedded device	Rockchip RK3588
OS	Ubuntu Linux 20.04

Table 4. The details of experimental conditions.

Parameter	Setting
Batch size	32
Momentum	0.937
Weight decay	5×10^{-4}
Epoch	100
Number of classes	10

Table 5. The initial parameter settings for training the proposed FRYOLO network model.

computational flow and alleviating the burden on embedded devices. In addition, a new output branch named the confidence sum is applied to speed up the threshold screening process in the post-processing stage. Based on that, the proposed method can quickly identify high-confidence targets by pre-calculating and outputting the confidence sum for each detection box. Hence, unnecessary computation and processing time can be reduced. Implementing these optimizations not only improves the detection performance on IoT embedded devices but also ensures remarkable computational efficiency and response speed in resource-constrained conditions.

Results and discussion

Experimental conditions

The experimental conditions of this work are summarized in Table 4. Python 3.8 and the deep learning framework PyTorch 2.3 are the software versions, with CUDA 12.0. The Central Processing Unit (CPU) is Intel Xeon w7-3445, and the Graphics Processing Unit (GPU) is GeForce RTX 4090-24G. The Operating System (OS) is Ubuntu Linux 20.04, with a hard disk of 8 TB and 128GB Random Access Memory (RAM). In addition, the embedded device is Rockchip RK3588, which integrates 4GB RAM, a built-in Artificial Intelligence (AI) accelerator NPU, and an ARM Ma-li-G610 GPU.

Evaluation metrics

Mathematically, for classification results based on the combination of the predicted labels and the true labels, the samples can be divided into four categories: True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN). Then, they can be employed to calculate the evaluation metrics, such as precision, recall, and mAP, which are investigated in this work.

Precision evaluates the accuracy of a model's predictions, especially among samples predicted as positive. It is the ratio of TP to the sum of TP and FP. The higher the precision, the fewer cases of the model incorrectly predicting negative samples as positive. Second, recall indicates the ability of the model to identify positive samples correctly. It is the ratio of TP to the sum of TP and FN. The higher the recall, the fewer cases of the model missing positive samples. In this work, the detection task is concerned not only with whether the model can accurately identify fruits (classification accuracy) but also with the accuracy of the bounding boxes identified by the model (localization accuracy). Lastly, mAP shows the average detection accuracy of the model across various categories, taking into account both classification and localization performance. Specifically, mAP@50 denotes the mAP value when the Intersection over the Union (IoU) threshold is 0.5, where IoU refers to the degree of overlap between the predicted bounding box and the ground truth bounding box. mAP@50 is relatively lenient as it considers a detection correct if the IoU between the predicted box and the ground truth box reaches 0.5 (i.e., 50% overlap). This balanced evaluation for detection and localization accuracy suits different tasks, so the mAP in this work is mAP@50.

Performance evaluation

Regarding the training of FRYOLO, two data formats are utilized: image and their corresponding label annotation. The label annotation files are in .txt format that records information such as the type and coordinates of the targets. The network employs an end-to-end backpropagation and stochastic gradient descent optimization method to train and optimize the model. The initial parameters for training are shown in Table 5. Here, a batch size of 32 is often a balanced value, as it maintains the stability of gradient estimation while not significantly increasing memory requirements. A momentum value of 0.937, close to the commonly used 0.9, is an empirically adequate value that can accelerate convergence and reduce oscillations. Weight decay is a regularization technique used

Version	Precision (%)	Recall (%)	mAP (%)	Params (M)	FLOPs (G)
FRYOLO based on YOLOv8n	84.7	89.0	92.5	3.2	8.7
FRYOLO based on YOLOv8s	89.4	87.5	94.7	11.2	28.6
FRYOLO based on YOLOv8m	89.8	88.3	94.8	25.9	78.9
FRYOLO based on YOLOv8l	84.7	86.0	93.0	43.7	165.2
FRYOLO based on YOLOv8x	87.6	86.8	93.8	68.2	257.8

Table 6. Performances of different YOLOv8 versions in this work.

Type	Precision (%)	Recall (%)	mAP (%)
Fresh apple	94.9	92.7	97.2
Fresh mango	80.6	87.1	89.8
Fresh orange	95.5	72.8	94.7
Fresh pear	91.2	94.3	96.5
Fresh banana	78.1	100	97.9
Defective apple	85.2	81.6	88.0
Defective mango	76.9	82.4	82.9
Defective orange	82.6	89.0	93.9
Defective pear	81.6	95.9	99.5

Table 7. Performances of the proposed FRYOLO model in detecting different fresh and defective fruits.

to prevent overfitting. A weight decay value of 5×10^{-4} is a common choice that effectively controls the model's weights without excessively penalizing model complexity.

Then, the proposed FRYOLO network model with various depths and widths is trained on the dataset, and the complexity is characterized by Floating-point Operations (FLOPs) and the number of parameters (Params). The detection performance and complexity of various versions are presented in Table 6. As the model grows more complex, so do the FLOPs and Params. To achieve a balance between detection performance and model complexity, FRYOLO based on YOLOv8n is selected. Although it exhibits slightly lower precision and mAP compared to the others, its overall performance remains remarkable. More importantly, FRYOLO based on YOLOv8x, the most updated YOLOv8 version, requires 68.2 M Params and 257.8G FLOPs, rendering it unsuitable for real-time detection on IoT embedded devices. In contrast, FRYOLO based on YOLOv8n reduces the Params by 95.3% and the FLOPs by 96.6%, while only experiencing a minor decline in accuracy, i.e., it provides the smallest model size, with only 3.2 M Params and 8.7G FLOPs, making it the least parameter-heavy and computationally complex among all YOLOv8 versions. This characteristic translates to minimal storage requirements and easier deployment on resource-constrained devices, making it the suitable choice for this work. As a result, the proposed FRYOLO network model is based on YOLOv8n.

Next, Table 7 shows the performances of the proposed FRYOLO model in detecting different fresh and defective fruits. The experimental results demonstrate remarkable performances. Specifically, for apple and mango detection, both the recall rate and mAP reach exceptional levels, strongly evidencing the model's robust capability in detecting their fresh states. Nonetheless, when it comes to detecting defective fruits, while the model's map remains relatively high, there are varying degrees of decline in both recall rate and precision, particularly pronounced in detecting defective apples and mangoes. The reason may be due to the scarcity of defective samples in the dataset, incurring insufficient generalization capability of the proposed method.

The normalized confusion matrix is presented in Fig. 10, with the true labels as the horizontal axis and the predicted labels as the vertical axis. The matrix's diagonal elements denote the ratio of correctly detected samples (i.e., TP) to the total number of samples in that category. The lower left region refers to the ratio of samples missed by the model (i.e., FN) to the total number of samples in that category. The upper right region represents the ratio of samples falsely detected by the model (i.e., FP) to the total number of samples in that category. When the values in the upper right region are high, the model has more false detection, revealing the model identifies the current sample as background or other objects. When the values in the lower left region are high, the model has missed detection in these categories, indicating the model has failed to detect the true types correctly.

The normalized confusion matrix demonstrates that the proposed model generally exhibits better precision in distinguishing fresh fruits, with higher rates than defective fruits. The reason may be due to the distinctive features of fresh fruits, which are more accessible for the model to identify. Concerning the defective fruits, the model accomplishes impressive results in identifying defective oranges and pears, as it detects all instances of defective oranges and pears without any FP. Nevertheless, it encounters challenges in identifying defective apples, with a relatively lower detection accuracy than others. To solve it, one solution may increase the number of defective apple samples in the training stage to enhance the model's sensitivity.

Precision evaluates the accuracy of the model's detection, while recall gauges the model's ability to identify all TP instances. In this regard, the F1 score balances these two metrics, serving as the weighted average of precision and recall, making it highly effective in comprehensively considering the model's accuracy and recall

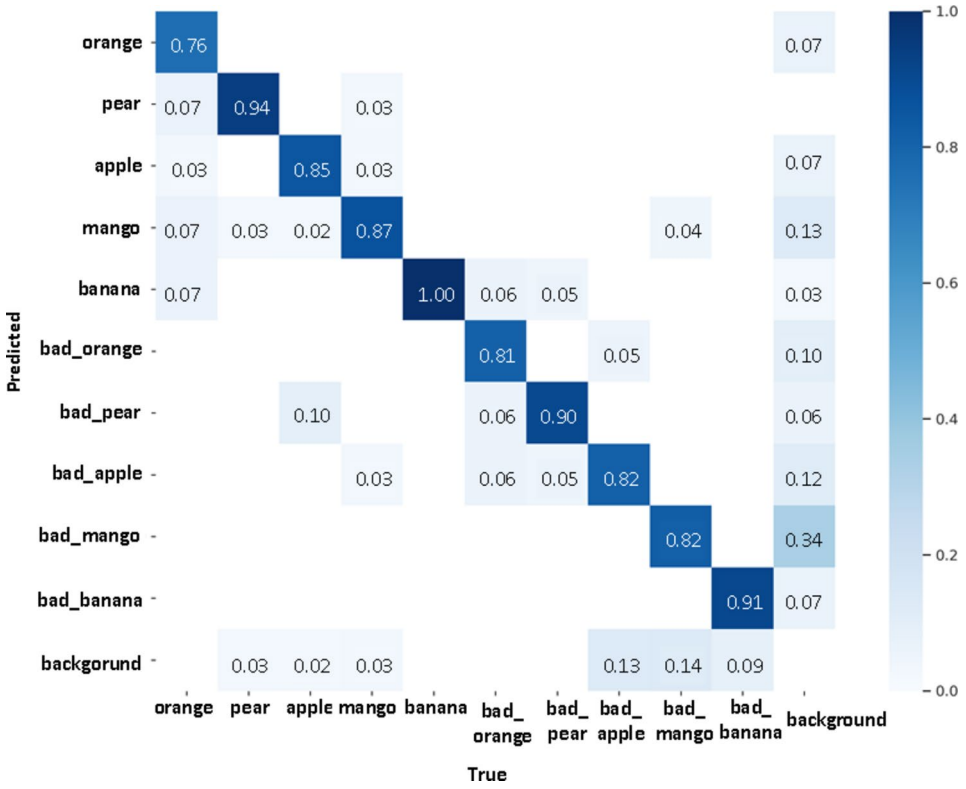

Fig. 10. The normalized confusion matrix.

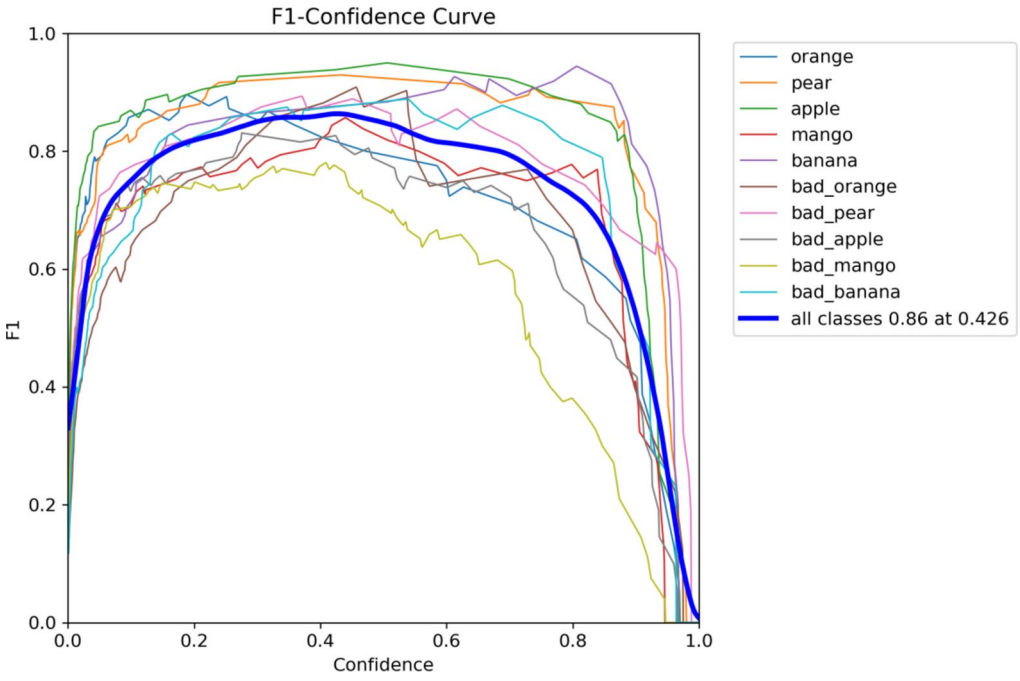

Fig. 11. The F1-confidence curve of the proposed FRYOLO network model.

capabilities. The F1-confidence curve plots the F1 score against the confidence threshold, enabling an analysis of the model's detection performance for each category and across different confidence thresholds, as illustrated in Fig. 11. It is observed that different categories achieve their peak F1 scores under varying threshold conditions. For instance, when the confidence threshold is 0.8, the mango reaches 0.9, indicating that the model detects most bananas at a confidence level of 0.8. Nonetheless, the F1 score for the bad_mango category drops, approaching

Fig. 12. Fresh and defective fruit detection results using the proposed FRYOLO network model.

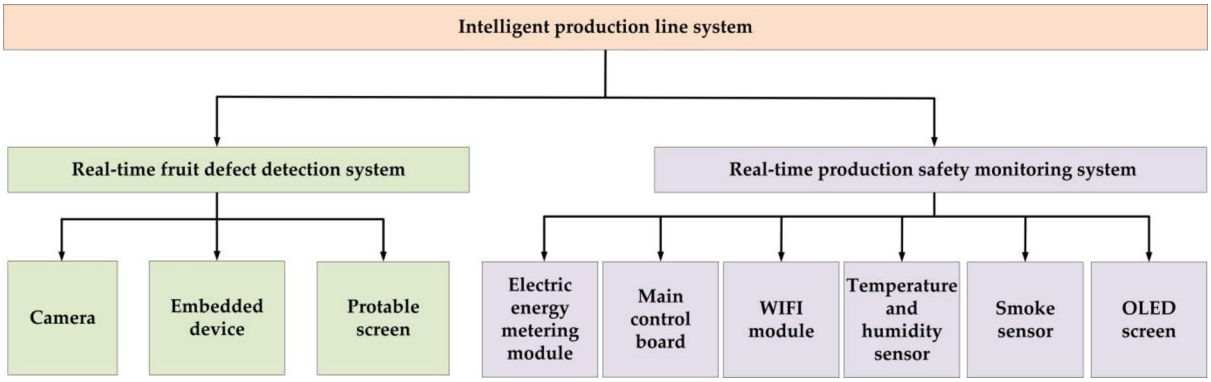

Fig. 13. The general framework of intelligent production line system based on FRYOLO.

0.4, suggesting that the model's detection capability for bad_mango is less robust at higher confidence levels, necessitating further improvements. Overall, the FRYOLO model accomplishes an optimal F1 score of 0.86 at a confidence threshold of 0.426, which is an impressive performance when considering its lightweight nature.

Lastly, Fig. 12 displays the fruit defect detection results for various cases using the proposed FRYOLO network model. As seen, the model demonstrates high stability in complex background environments, accurately identifying different fruits with high confidence. Therefore, it provides a foundation for high-precision fresh and defective fruit detection in practical IoT applications, along with powerful generalization ability.

System implementation

To demonstrate the availability of the proposed FRYOLO model in practical IoT applications, an intelligent production line system is implemented, which contains two subsystems: a real-time fruit defect detection system and a real-time production safety monitoring system. The general framework is illustrated in Fig. 13, and its hardware implementation is shown in Fig. 14.

The real-time fruit defect detection system deploys the proposed FRYOLO network model into the IoT embedded devices to perform the detection tasks. It consists of a camera, an embedded device, and a portable

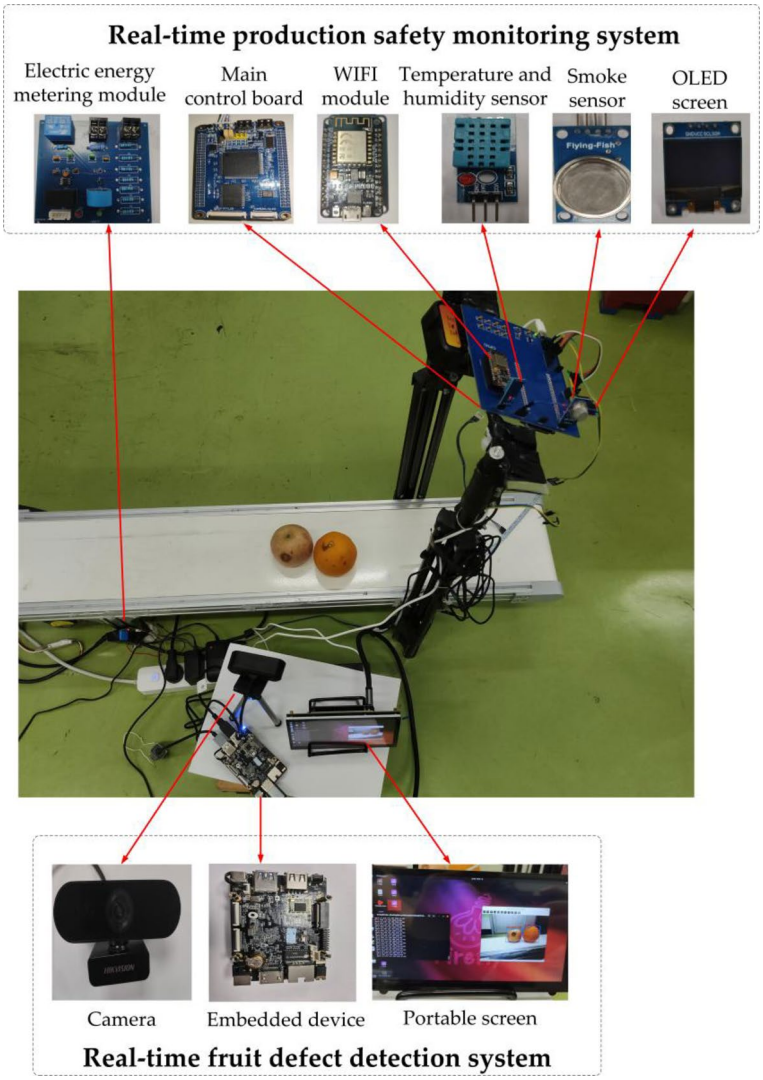

Fig. 14. The hardware implementation of intelligent production line system.

screen, where the embedded device processes the video recorded by the camera and displays the detection results on the portable screen. The real-time safety monitoring system monitors the ambient temperature, humidity, and smoke particle concentration through a temperature and humidity sensor and a smoke sensor. Then, an Organic Light Emitting Diode (OLED) display screen shows these real-time environmental conditions. In addition, an electric energy metering module is installed at the bottom of the conveyor belt. It enables real-time monitoring of the voltage, current, and power, facilitating precise control over its operation and interruption to guarantee production safety. In emergencies, such as a fire, the main control board will activate the electric energy metering module to cut off the power supply. Meanwhile, an alarm will be sent to the user through a WIFI module. Such functions enhance safety protection capabilities during the production procedure.

The real-time performance of models deployed on the IoT embedded devices has been evaluated across various complex environmental backgrounds and lighting conditions, as depicted in Fig. 15. The model consistently demonstrates robust performance under these testing scenarios. As seen, it can accurately detect whether a single fruit or multiple fruits with high confidence. Even when dealing with complex environments, it performs well, revealing its reliability. More importantly, by adopting multithreading techniques, the real-time frame rate can be increased to 33 FPS, which satisfies the real-time requirements of canned fruit production in practical IoT applications. However, during the experiments, it can be observed that when the camera is at the same height as the banana and the banana's orientation is parallel to the camera's view, the model struggles to accurately detect the banana. So, the system's physical structure needs to be further improved.

Comparative study

On one hand, to extensively assess the advantages of this work in terms of metrics, a comparative study with the recent state-of-the-art works is conducted, as outlined in Table 8. Regarding precision, recall, and mAP, the proposed FRYOLO network model accomplishes 84.7%, 89.0%, and 92.5%, respectively. Consequently, when considering the deployment of the IoT embedded devices as a primary concern, this work exhibits robustness

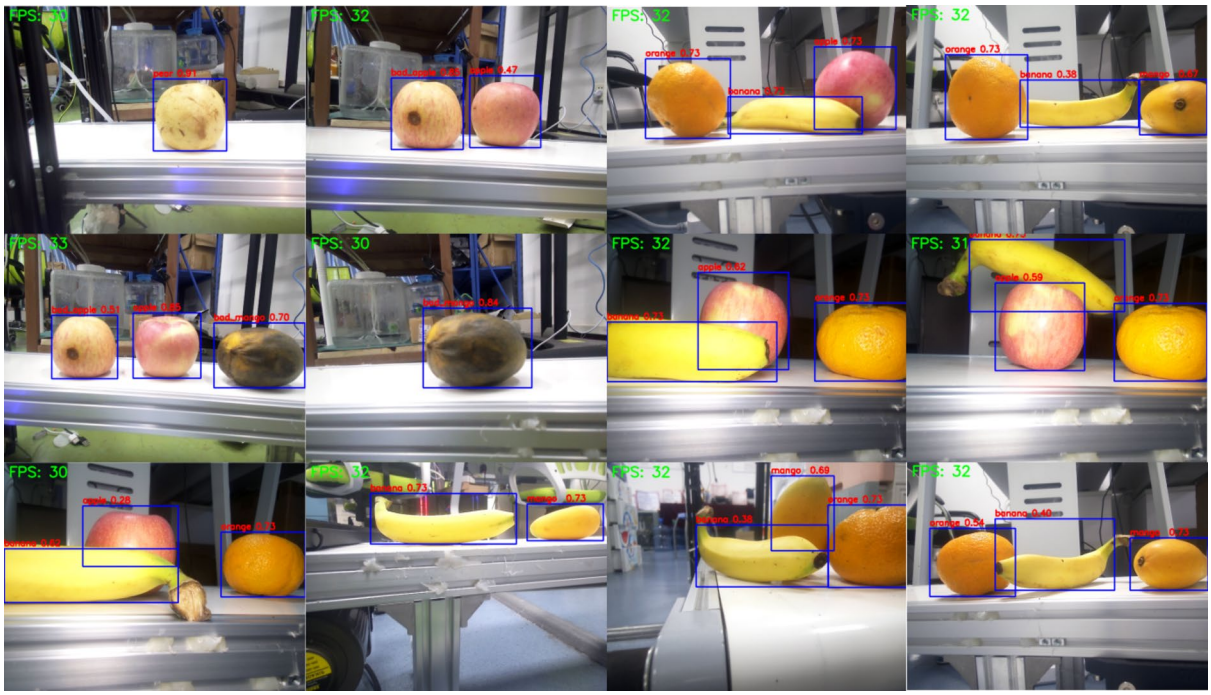

Fig. 15. Fresh and defective fruit detection results using the designed intelligent production line system that deploys FRYOLO.

Method	Precision (%)	Recall (%)	mAP (%)	Params (M)	FLOPs (G)
Liu et al.[11]	–	74.2	92.5	3.2	0.7
Redmon and Farhadi[30]	79.4	84.0	88.7	81.9	104.5
Li et al.[31]	84.9	79.0	88.2	4.2	11.8
Ren et al.[32]	–	73.8	96.4	41.8	180.0
Lin[33]	–	81.1	93.3	20.0	107.5
Chen et al.[34]	–	79.4	85.8	32.2	153.0
Zhang et al.[35]	–	84.8	90.3	47.6	238.6
Wang et al.[36]	79.8	67.4	80.8	2.7	8.2
Zhang et al.[37]	–	74.8	88.3	41.4	179.2
This work	84.7	89.0	92.5	3.2	8.7

Table 8. A comparative study with recent works in terms of metrics.

Method	Memory usage (M)	Power consumption (W)	FPS
Redmon and Farhadi[30]	900	5.77	12
Li et al.[31]	268	5.22	29
Ren et al.[32]	123	10.40	6
Liu et al.[11]	80	10.20	13
Wang et al.[36]	245	5.06	29
Zhang et al.[37]	129	11.60	7
This work	237	5.66	33

Table 9. Performance comparison of different methods deployed on IoT embedded devices.

with a lightweight nature, which contributes valuable competitiveness in real-world scenarios and would open a novel direction in the lightweight model development for IoT.

On the other hand, to comprehensively evaluate the performance of this work on IoT embedded devices, some typical models have been selected to deploy on the designed intelligent production line system, where these deployed models can be categorized into two-stage and single-stage types, as presented in Table 9. By

comparing memory usage, power consumption, and FPS, it can be found that two-stage models have higher power consumption and lower FPS than one-stage models. In addition to the SSD model proposed by Liu et al.[11], other single-stage models tend to have higher memory usage. Overall, the proposed FRYOLO network model shows benefits when considering good FPS results under proper memory usage and power consumption.

Conclusion

To address the issue of existing neural network models being overly complex and unable to deploy in IoT appropriately embedded devices, a lightweight solution based on the improved YOLOv8n, i.e., FRYOLO, has been proposed in this work. As for the method validation, a fruit dataset is established by conducting image acquisition, manual annotation, and dataset segmentation. It contains five fruits in two states with various backgrounds, solving the need for high-quality fruit defect detection tasks. The results reveal that the FRYOLO model achieves up to 84.7% in recall and 89.0% in mAP, along with a precision of 92.5%. Therefore, using the proposed method as a core, an intelligent production line system that supports the efficient operation of the canned fruit production process is constructed, which demonstrates the availability of the proposed FRYOLO network model in real-world scenarios. In the future, the dataset will be expanded by adding more fruit varieties and complex backgrounds to enhance its applicability further. Furthermore, other advanced deep learning models or real-time detection techniques[38–40] will be investigated to deploy into edge devices employing the optimized lightweight solution proposed in this work.

Data availability

The datasets generated and analyzed during the current study are available at https://github.com/wpx-1/fruit-yolo/tree/master.

Received: 30 September 2024; Accepted: 28 January 2025
Published online: 30 January 2025

References

1. Li, J. et al. An AIoT-based assistance system for visually impaired people. *Electronics* **12**, 3760 (2023).
2. Zhao, P., Zhou, W. & Na, L. High-precision object detection network for automate pear picking. *Sci. Rep.* **14**, 14965 (2024).
3. Girshick, R., Donahue, J., Darrell, T. & Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June*, pp. 580–587 (2014).
4. Girshick, R. & Fast, R-C-N-N. December. In *Proc. of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13*, 1440–1448 (2015).
5. Jiang, H. & Learned-Miller, E. June. Face detection with the faster R-CNN. In *Proc. of the IEEE International Conference on Automatic Face and Gesture Recognition, Washington, DC, USA, 30 May–3*, 650–657 (2017).
6. 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 (2022).
7. Parico, A. I. B. & Ahamed, T. Real time pear fruit detection and counting using YOLOv4 models and deep SORT. *Sensors* **21**, 4803 (2021).
8. Cardellicchio, A. et al. Detection of tomato plant phenotyping traits using YOLOv5-based single stage detectors. *Comput. Electron. Agric.* **207**, 107757 (2023).
9. Alam Soeb, J. et al. Tea leaf disease detection and identification based on YOLOv7 (YOLO-T). *Sci. Rep.* **13**, 6078 (2023).
10. Ma, B. et al. Using an improved lightweight YOLOv8 model for real-time detection of multi-stage apple fruit in complex orchard environments. *Artif. Intell. Agric.* **11**, 70–82 (2024).
11. Liu, W. et al. SSD: Single shot multibox detector. In *Proc. of European Conference on Computer Vision, Amsterdam, Netherlands, 11–14 October*, 21–37. (2016).
12. Ireri, D., Belal, E., Okinda, C., Makange, N. & Ji, C. A computer vision system for defect discrimination and grading in tomatoes using machine learning and image processing. *Artif. Intell. Agric.* **2**, 28–37 (2019).
13. Liu, T. H., Ehsani, R., Toudeshki, A., Zou, X. J. & Wang, H. J. Identifying immature and mature pomelo fruits in trees by elliptical model fitting in the Cr–Cb color space. *Precis Agric.* **20**, 138–156 (2019).
14. Parraga-Arotinco, E. A., Cárdenas-Martinez, R. P., Misari-Baldeón, J. L. & Vilchez-Baca, H. A. Automated system for sorting blueberries by size and degree of ripeness. In *Proc. of the International Conference on Control, Robotics and Informatics, Danang, Vietnam, 26–28 May*, 74–80 (2023).
15. Prakash, A. J. & Prakasam, P. An intelligent fruits classification in precision agriculture using bilinear pooling convolutional neural networks. *Vis. Comput.* **38**, 1–17 (2022).
16. Gao, F. et al. Multi-class fruit-on-plant detection for apple in SNAP system using faster R-CNN. Comput. *Electron. Agric.* **176**, 105634 (2020).
17. Zhai, X. et al. Detection of maturity and counting of blueberry fruits based on attention mechanism and bi-directional feature pyramid network. *J. Food Meas. Charact.* 1–16 (2024).
18. Wang, Y. et al. DSE-YOLO: Detail semantics enhancement YOLO for multi-stage strawberry detection. *Comput. Electron. Agric.* **198**, 107057 (2022).
19. Yang, R., Hu, Y., Yao, Y., Gao, M. & Liu, R. Fruit target detection based on BCo-YOLOv5 model. *Mob. Inf. Syst.* **2022**, 8457173 (2022).
20. Rastegari, M., Ordonez, V., Redmon, J. & Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proc. of European Conference on Computer Vision, Amsterdam, Netherlands, 11–14 October*, 580–587 (2016).
21. Qu, Z., Zhou, Z., Cheng, Y. & Thiele, L. Adaptive loss-aware quantization for multi-bit networks. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June*, 7988–7997 (2020).
22. Qu, Z. et al. B. DRESS: Dynamic real-time sparse subnets. Preprint at https://arXiv.org/2207.00670 (2022).
23. Abdul, K. S. & Hafiz, R. Quality scalable quantization methodology for deep learning on edge. Preprint at https://arXiv.org/2407.11260 (2024).
24. Ahmad, T., Ma, Y., Yahya, M., Ahmad, B. & Nazir, S. Object detection through modified YOLO neural network. *Sci. Program.* **2020**, 8403262 (2020).
25. Zaghari, N., Fathy, M., Jameii, S. M. & Shahverdy, M. The improvement in obstacle detection in autonomous vehicles using YOLO non-maximum suppression fuzzy algorithm. *J. Supercomput.* **77**, 13421–13446 (2021).

26. Fan, Y., Zhang, L. & Li, P. A lightweight model of underwater object detection based on YOLOv8n for an edge computing platform. *J. Mar. Sci. Eng.* **12**, 697 (2024).
27. Elhanashi, A., Saponara, S. & Zheng, Q. Classification and localization of multi-type abnormalities on chest X-rays images. *IEEE Access* **11**, 83264–83277 (2023).
28. Luo, B., Kou, Z., Han, C. & Wu, J. A. Hardware-friendly foreign object identification method for belt conveyors based on improved YOLOv8. *Appl. Sci.* **13**, 11464 (2023).
29. Liu, H. et al. Upgrading swin-B transformer-based model for accurately identifying ripe strawberries by coupling task-aligned one-stage object detection mechanism. *Comput. Electron. Agric.* **218**, 108–674 (2024).
30. Redmon, J. & Farhadi, A. Yolov3: an incremental improvement. Preprint at https://arXiv.org/180402767 (2018).
31. Li, C. et al. *YOLOv6: A single-stage Object Detection Framework for Industrial Applications*. Preprint at https://arXiv.org/arXiv:2209.02976 (2022).
32. Ren, S., He, K., Girshick, R., Sun, J. & Faster, R-C-N-N. Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **39**, 1137–1149 (2016).
33. Lin, T. Focal loss for dense object detection. Preprint at https://arXiv.org/1708.02002 (2017).
34. Chen, Z. et al. Disentangle your dense object detector. In *Proc. of the ACM International Conference on Multimedia, Virtual, 19–23 October*, 4939–4948 (2021).
35. Zhang, H. et al. Dino: Detr with improved denoising anchor boxes for end-to-end object detection. Preprint at https://arXiv.org/2203.03605 (2022).
36. Wang, A. et al. YOLOv10: Real-time end-to-end object detection. Preprint at https://arXiv.org/2405.14458 (2024).
37. Zhang, H. et al. Towards high quality object detection via dynamic training. In *Proc. of European Conference on Computer Vision, Amsterdam, Netherlands, 11–14 October*, 260–275 (2016).
38. Chen, R. et al. Rapid detection of multi-QR codes based on multistage stepwise discrimination and a compressed MobileNet. *IEEE Internet Things J.* **10**, 15966–15979 (2023).
39. Moulik, S. RESET: a real-time scheduler for energy and temperature aware heterogeneous multi-core systems. *Integration* **77**, 59–69 (2021).
40. Ren, J. et al. Effective extraction of ventricles and myocardium objects from cardiac magnetic resonance images with a multi-task learning U-Net. *Pattern Recognit. Lett.* **155**, 165–170 (2022).

Acknowledgements

The authors would like to appreciate the special contributions from Digital Content Processing & Security Technology of Guangzhou Key Laboratory.

Author contributions

Conceptualization: Rongjun Chen, Peixian Wang, Binfan Lin, Jiawen Li, Jinchang Ren, and Huimin Zhao; Funding acquisition: Rongjun Chen, Leijun Wang, Xianxian Zeng, Jiawen Li, Jinchang Ren, and Huimin Zhao; Methodology: Rongjun Chen, Peixian Wang, Binfan Lin, Leijun Wang, and Jiawen Li; Project administration: Leijun Wang, Xianxian Zeng, Xianglei Hu, Jun Yuan, and Jinchang Ren; Resources: Rongjun Chen, Jiawen Li, Jinchang Ren, and Huimin Zhao; Software: Peixian Wang, Binfan Lin, Xianxian Zeng, Xianglei Hu, and Jun Yuan; Validation: Rongjun Chen, Peixian Wang, Binfan Lin, Xianxian Zeng, and Jiawen Li; Writing - original draft: Rongjun Chen, Peixian Wang, Binfan Lin, and Jiawen Li; Writing - review and editing: Peixian Wang, Leijun Wang, Xianglei Hu, Jun Yuan, and Jiawen Li.All authors reviewed the manuscript.

Funding

This work was supported in part by the National Natural Science Foundation of China under Grant 62072122, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2024A1515010219, in part by the Special Projects in Key Fields of Ordinary Universities of Guangdong Province under Grant 2021ZDZX1087, in part by the Guangzhou Science and Technology Plan Project under Grants 2024B03J1361, 2023B03J1327, 2023A04J0362, and 2023A04J0361, in part by the Open Research Fund of Guangdong Provincial Key Laboratory of Big Data Computing under Grant B10120210117-OF08, in part by the Guangdong Province Ordinary Colleges and Universities Young Innovative Talents Project under Grants 2023KQNCX036 and 2022KQNCX038, in part by the Key Discipline Improvement Project of Guangdong Province under Grants 2022ZDJS015 and 2021ZDJS025, in part by the Scientific Research Capacity Improvement Project of the Doctoral Program Construction Unit of Guangdong Polytechnic Normal University under Grant 22GPNUZDJS17, in part by the Graduate Education Demonstration Base Project of Guangdong Polytechnic Normal University under Grant 2023YJSY04002, and in part by the Research Fund of Guangdong Polytechnic Normal University under Grants 2023SDKYA004 and 2022SDKYA015.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to J.L., J.R. or H.Z.