WEP Warwick Evans Publishing

International Journal of Computer Science and Information Technology

ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 2, Number 2, Year 2024
DOI: https://doi.org/10.62051/ijcsit.v2n2.01
Journal homepage: https://wepub.org/index.php/IJCSIT/index

High-Precision Tomato Maturity Detection Using a Genetic Algorithm Optimized Swin-YOLO Network

Lujie Fan[1, 2, *]

[1]Graduate School, University of the East, Manila, Philippines

[2]College of Computer, Weifang University of Science and Technology, Shouguang, China

*Corresponding Author: Lujie Fan

ABSTRACT

In the realm of precision agriculture, the determination of tomato maturity stages plays a crucial role in optimizing harvest timings, ensuring produce quality, and maximizing yields. Traditional approaches, largely reliant on manual inspection, are not only labor-intensive but also subject to human error, making them unsuitable for modern, large-scale agricultural operations. Addressing these challenges, this study pioneers the use of a Genetic Algorithm (GA) optimized Swin-YOLO (You Only Look Once) network, aiming to automate and enhance the precision of tomato maturity detection. By integrating the advanced capabilities of the Swin Transformer for feature extraction with the efficiency of the YOLO object detection framework, the proposed Swin-YOLO model is meticulously designed to identify varying stages of tomato maturity from complex agricultural imagery. The introduction of a GA facilitates the systematic optimization of the network's architecture and hyperparameters, focusing on improving accuracy and computational efficiency across diverse environmental conditions. Our research involved compiling a comprehensive dataset of annotated tomato images, reflecting a wide array of maturity stages under different lighting and occlusional challenges. The findings from our study indicate a significant leap in performance, with the GA-optimized Swin-YOLO network outshining existing deep learning models and conventional manual methods in terms of both detection accuracy and processing speed. Notably, the model exhibits exceptional adeptness at handling images affected by variable lighting and partial occlusions, marking a substantial advancement in automated tomato maturity detection. The implications of this research are profound, offering a scalable, efficient, and highly accurate solution to one of agriculture's longstanding challenges. This breakthrough not only stands to reduce labor costs and enhance operational efficiency but also contributes to the sustainability of tomato farming practices by facilitating more informed and timely harvesting decisions. Furthermore, the successful application of GA for network optimization underscores the potential of combining deep learning with evolutionary algorithms to tackle complex agricultural challenges, setting the stage for future innovations in precision agriculture and beyond.

KEYWORDS

Tomato Maturity Detection, Swin-YOLO, Genetic Algorithm, Deep Learning, Agricultural Automation

1. INTRODUCTION

1.1. Background and Significance

The efficient detection of tomato maturity stages is a cornerstone in agricultural production, directly affecting the marketability, taste, and nutritional value of the yield. In the era of precision agriculture, the ability to accurately assess the maturity of tomatoes not only ensures the quality of produce

reaching consumers but also maximizes yield and minimizes waste, thereby enhancing sustainability in food production. Traditional methods, reliant on manual inspection, are fraught with challenges including labor intensity, subjectivity, and inconsistency, highlighting a pressing need for automation in maturity detection processes.

1.2. Literature Review

Recent advancements in technology have ushered in a plethora of research exploring automated systems for agricultural tasks, with a significant focus on tomato maturity detection. Traditional approaches have ranged from colorimetric analysis to complex biochemical assays, each with its limitations in terms of scalability, accuracy, and applicability in field conditions. With the advent of deep learning, a shift has been observed towards leveraging computer vision and machine learning techniques. Models such as Convolutional Neural Networks (CNNs) and the YOLO (You Only Look Once) framework have demonstrated potential in identifying and classifying crops and their maturity stages directly from images. However, despite these advancements, challenges remain in terms of real-time processing capabilities and adaptability to varied environmental conditions and tomato breeds.

1.3. Research Gap

While existing methodologies have laid a solid foundation for automated tomato maturity detection, a significant gap persists in achieving high precision and real-time detection in diverse agricultural settings. Many deep learning models still struggle with the variability present in natural environments, including changes in lighting, occlusions, and the physical diversity among tomato species. Furthermore, the optimization of these models often requires extensive computational resources and expert knowledge, limiting their accessibility and scalability for widespread agricultural use.

1.4. Objectives

This study aims to bridge the aforementioned gaps by developing and implementing a Genetic Algorithm (GA) optimized Swin-YOLO network specifically for tomato maturity detection. By harnessing the strengths of Swin Transformers for effective feature extraction and the real-time detection capabilities of the YOLO framework, coupled with the optimization prowess of GAs, this research seeks to achieve unparalleled accuracy and efficiency in detecting tomato maturity stages across various conditions.

1.5. Contributions

The contributions of this study are manifold:

Innovation: Introduction of a novel GA-optimized Swin-YOLO network, marking a significant leap in applying deep learning to agriculture.

Accuracy and Efficiency: Demonstration of a system capable of high-precision, real-time tomato maturity detection, surpassing existing models in both laboratory and field tests.

Scalability:Provision of a scalable solution that can be adapted to different tomato varieties and growing conditions without the need for extensive retraining or manual optimization.

Sustainability: Contribution towards sustainable agriculture practices by enabling more precise harvest timing, reducing waste, and ensuring the highest quality of produce.

Research Implications: Establishment of a benchmark for future research in agricultural automation and deep learning applications, offering a comprehensive framework for addressing similar challenges in precision agriculture.

In summary, this research not only addresses a critical need in agricultural production but also paves the way for future innovations in the field, blending deep learning with genetic algorithms to overcome current limitations in crop maturity detection.

2. MATERIALS

2.1. Dataset Collection

The dataset for this study was meticulously assembled to ensure a comprehensive representation of tomato maturity stages, facilitating the development and validation of a highly accurate maturity detection model. This section details the process involved in dataset collection, including the sources, the criteria for selection, and the methodologies employed to prepare the dataset for model training and testing.

2.1.1. Sources:

The dataset comprises over 10,000 high-resolution images of tomatoes, sourced from a variety of settings to capture the diversity in appearance attributable to different maturity stages, varieties, and environmental conditions. These settings included:

- **Commercial Farms:** Partnering with local agricultural establishments, images were captured across different seasons to include a wide range of maturity stages from green to fully ripe tomatoes.
- **Research Greenhouses:** Collaborations with agricultural research institutions provided access to controlled environments where tomatoes of specific varieties were grown and imaged at predetermined maturity stages.
- **Public Databases:** A selection of images was also sourced from publicly available agricultural image databases to supplement the dataset with additional variability in tomato appearances and backgrounds.

2.1.2. Selection Criteria:

The inclusion of images in the dataset was based on several criteria to ensure the quality and relevance of the data:

- **Resolution and Quality:** Only high-resolution images that clearly showed the tomato and its surrounding environment were included. This criterion was essential for ensuring that the model could identify and learn from detailed features indicative of maturity stages.
- **Maturity Stage Diversity:** The dataset was designed to evenly cover all key maturity stages of tomatoes, from immature green to fully ripe. This diversity is critical for training the model to accurately classify tomatoes across the entire maturity spectrum.
- **Environmental Variability:** To enhance the model's robustness and generalizability, the dataset included images captured under a variety of environmental conditions, including different lighting conditions, weather scenarios, and backgrounds.

2.1.3. Preparation:

Once collected, the images underwent a series of preprocessing steps to prepare them for use in training and testing the model:

- **Annotation:** Each image was manually annotated by agricultural experts to label the maturity stage of the tomato. This step involved categorizing each tomato into one of five maturity stages, ensuring the accuracy of training labels.

- **Standardization:** Images were resized to a uniform dimension to facilitate efficient processing by the model. This step involved scaling down larger images and cropping them to a standard size while maintaining the aspect ratio to avoid distorting the features of interest.
- **Augmentation:** To further increase the diversity of the dataset and improve the model's ability to generalize to new, unseen images, data augmentation techniques were applied. These included rotations, flips, and variations in brightness and contrast, simulating different capturing conditions.

The comprehensive and meticulously prepared dataset serves as the foundation for developing a robust and accurate tomato maturity detection model, capable of performing well across a variety of real-world conditions.

2.2. Data Annotation

2.2.1. Overview

Data annotation is a crucial step in preparing datasets for machine learning models, especially in tasks requiring precise object detection and classification, such as tomato maturity detection. This process involves labeling the data with information that models can learn from. For the tomato dataset, annotation entailed marking each image with the maturity stage of the tomato and, where necessary, drawing bounding boxes around each tomato to facilitate object detection.

2.2.2. Annotation Process

1. **Defining Categories**: The first step was to define clear, distinct categories for tomato maturity stages based on agricultural standards. These stages were identified as immature green, mature green, breaker, turning, pink, and red, providing a comprehensive spectrum of tomato maturity.
2. **Tool Selection**: We utilized a combination of automated tools and manual annotation processes. For initial bounding box placement and preliminary stage classification, an automated tool pre-processed images based on color segmentation and shape recognition algorithms. This step significantly reduced the manual workload.
3. **Manual Verification and Correction**: Each automated annotation was manually checked and corrected by a team of agricultural experts and data annotators. This ensured high accuracy, as human judgment is crucial for nuanced distinctions between maturity stages and for verifying the correct placement of bounding boxes.
4. **Quality Control**: To maintain consistency and accuracy across the dataset, a quality control protocol was implemented. This involved random checks of annotated images by a senior annotator and periodic training sessions for the annotation team to calibrate their understanding of maturity stages.
5. **Annotation Format**: Annotations were stored in a structured format compatible with the training framework used for the Swin-YOLO model. Each image's file was accompanied by an XML or JSON file containing the coordinates of bounding boxes and the corresponding maturity stage labels.

2.2.3. Challenges and Solutions

- **Variability in Tomato Appearance**: The diverse appearance of tomatoes, even within the same maturity stage, posed a significant challenge. Solution: Enhanced training for annotators on variability and introduced a validation step where annotations were reviewed by agricultural experts.
- **Occlusion and Overlap**: In some images, tomatoes were partially occluded or overlapping, making accurate annotation difficult. Solution: Adopted a protocol for handling occlusions,

including marking the visible parts of tomatoes and using predictive judgment for partially visible maturity indicators.

- **Scalability**: Manually annotating thousands of images was time-consuming. Solution: Leveraged semi-automated annotation tools to pre-annotate images, focusing manual efforts on verification and correction, thus improving scalability and efficiency.

2.2.4. Impact on Model Training

The meticulously annotated dataset played a pivotal role in training the Swin-YOLO model. Accurate and consistent annotations allowed the model to learn the nuanced differences between maturity stages effectively, leading to high precision and recall in maturity stage detection. This rigorous annotation process ensures the model's reliability and applicability in real-world agricultural settings, demonstrating the critical importance of data annotation in developing AI solutions for precision agriculture.

2.3. Data Enhancement

2.3.1. Overview

Data enhancement, often referred to as data augmentation, is a technique used to increase the diversity of a dataset without actually collecting more data. This is particularly crucial in machine learning and computer vision tasks, where models benefit from exposure to a wide range of variations of the training data. In the context of tomato maturity detection, data enhancement involves applying a series of transformations to the images in the dataset to simulate different capturing conditions, thereby improving the model's generalization capability.

2.3.2. Techniques Employed

1. **Rotation and Flipping**: Images were randomly rotated by angles up to 30 degrees and flipped horizontally or vertically. This simulates the varying orientations in which tomatoes might be captured in real-world scenarios.
2. **Rescaling and Cropping**: Random rescaling and cropping of images were applied to mimic the effect of tomatoes being at different distances from the camera. This helps the model learn to recognize tomatoes of various sizes and partial occlusions.
3. **Brightness and Contrast Adjustments**: Variations in lighting conditions were simulated by adjusting the brightness and contrast of the images. This ensures the model's robustness against changes in natural and artificial lighting when deployed in diverse agricultural environments.
4. **Color Jittering**: Slight alterations in the color properties of the images (hue, saturation, and value) were introduced to account for the natural variation in tomato colors due to different growth conditions and camera sensors.
5. **Noise Injection**: To mimic the effect of different camera qualities and environmental conditions, a small amount of random noise was added to some images. This enhances the model's ability to perform well even in less-than-ideal photographic conditions.

2.3.3. Implementation

The data enhancement techniques were implemented using a combination of custom scripts and libraries such as OpenCV and TensorFlow's Keras preprocessing utilities. The transformations were applied on-the-fly during the model training process, which means each image was randomly transformed differently at each epoch of training. This approach significantly increased the effective size of the dataset without the need for additional storage space.

2.3.4. Benefits

- **Improved Model Generalization**: By training on enhanced data, the model becomes less prone to overfitting and better at generalizing to new, unseen data, which is critical for real-world applications.
- **Increased Dataset Robustness**: Data enhancement introduces the model to a broader spectrum of variations, making it more robust to variations in new datasets.
- **Cost-Effective Dataset Expansion**: It provides a cost-effective way to expand the dataset, bypassing the need for additional data collection and annotation efforts.

2.3.5. Impact on Model Performance

Data enhancement played a pivotal role in the successful development of the GA-optimized Swin-YOLO network for tomato maturity detection. The enhanced dataset ensured that the model was exposed to a wide range of image variations, closely mimicking the diversity of real-world conditions. This led to noticeable improvements in the model's accuracy, precision, and recall, demonstrating the effectiveness of data enhancement in preparing datasets for training high-performing machine learning models.

3. OPTIMIZATION OF NETWORK STRUCTURE BASED ON GA

The methodology section outlines the systematic approach undertaken to develop, optimize, and evaluate the performance of a GA-optimized Swin-YOLO network for the task of high-precision tomato maturity detection. This process encompasses dataset preparation, network architecture design, optimization through genetic algorithms, and comprehensive evaluation metrics.

The main idea of genetic algorithm to optimize neural network is to let the neural network continuously generate and evolve individuals in the population, and determine the best design method of Swin-YOLO network by finding an optimal individual.

The process of using genetic algorithm to optimize Swin-YOLO neural network in this experiment is roughly divided into five steps: (1) population initialization, using the preset hyperparameters of YOLOv5 and the structural hyperparameter values of Swun-YOLO, a custom initial hyperparameter file is constructed as the initialization population. (2) Calculate the fitness function of each individual and evaluate its performance on the object detection task; (3) Selection operation, according to the size of the fitness function, select some excellent individuals to enter the next generation according to a certain probability; (4) Mutation operation, which randomly changes some hyperparameters of the selected individuals to increase the diversity of the population; 5.Repeat steps 2-4 until the maximum number of iterations is reached. Pseudocode for optimizing the Swin -YOLO network by GA In this paper, the process of optimizing the Swin -YOLO neural network by GA is roughly divided into five steps: (1) Population initialization, the preset hyperparameters of YOLOv5 and the structural hyperparameter values of Swin -YOLO were used to construct a custom initial hyperparameter file as the initialization population. (2) Calculate the fitness function of each individual and evaluate its performance on the object detection task; (3) Selection operation, according to the size of the fitness function, select some excellent individuals to enter the next generation according to a certain probability; (4) Mutation operation, which randomly changes some hyperparameters of the selected individuals to increase the diversity of the population; 5.Repeat steps 2-4 until the maximum number of iterations is reached. The specific pseudo-code of GA optimized Swin-YOLO network is shown in Table 1.

Table 1. Pseudocode for GA optimized Swin-YOLO network

Algorithm: Optimizing Swin-YOLO network by GA.

: hyperparameters H, metadata M, num generations n, mutation probability m, standard_deviation s

Output: results R

```
Assign mutation_scale g = M0
Assign fitness f = calculate_fitness(H,M)
Assign best_fitness bf = f
for each generation e do
  Assign parents p = choose best n from H
  Assign combination c = weighted selection based on f from H
  Initialize mutation_vector v to 1
  for v = 1 do
    for each vi ∈ v do
      if random(1) < m then
        Compute vi = gi * normal_distribution(0,s) + 1
      else
        Assign vi = 1
      end if
    end for
  end for
  Compute c = c * v
  Assign c = adjust (c, M)
  Assign R = train_model (c)
  Assign f = calculate_fitness(R)
  if f > bf then
      Assign bf = f
  end if
  Add c to H
end for
return R
```

In the setting of hyperparameter optimization, considering that the sliding window self-attention mechanism realizes information fusion through W-MSA and SW-MSA, it needs to be used continuously and alternately to obtain better results. However, W-MSA and SW-MSA modules do not necessarily need to appear in pairs, so W-MSA is set as the first submodule in SwinTR module to optimize the total number of two submodules of W-MSA and SW-MSA. For the interior of SwinTR module, the optimal values of Dropout probability of MLP layer, attention layer, and residual connection will change with the total number of two sub-modules of W-MSA and SW-MSA. In addition, the size of the partition window is also critical to the detection performance. Therefore, they will also be added to the list of hyperparameter optimization.

In summary, the objectives of GA selected in this paper to optimize the hyperparameters of SW-YOLO network include four aspects: (1) the number of W-MSA modules and SW-MSA modules in the sliding window self-attention modules of each layer; (2) the size of the window in the sliding window self-attention module; (3) Dropout probability of MLP layer, attention layer and residual

connection in sliding window self-attention module; (4) hyperparameters such as learning rate, optimizer, loss function, data augmentation, etc.

3.1. Population initialization

In order for the GA to be able to search in the solution space to find the optimal solution, a certain number of individuals, each of which is a solution vector, should be randomly generated, and these individuals constitute the initial population.

In this paper, prior information is used to obtain better optimization effects. The initial hyperparameters of YOLOv5 pre-optimized in the COCO dataset are used as the initial population of the genetic algorithm, and the size of the initial population is 1. Different from the traditional genetic algorithm with random initialization, the genetic algorithm in this paper does not keep the population size unchanged in each generation, but dynamically increases the population size according to the number of iterations of the algorithm, and then uses truncation selection to control the amount of calculation to improve the poor search ability caused by the small initial population.

The encoding method of individuals is an important parameter in genetic algorithms. Each individual in the population is represented using a chromosome encoding of a certain length. There are many ways to encode chromosome, such as binary code, real code, tree code and so on. Considering that the hyperparameters of the neural network to be optimized contain integer and floating point numbers, the data types can be easily converted by using the round() number, so the real number encoding method is used to encode the neural network related hyperparameters into chromosomes to improve the flexibility and efficiency of the optimization process. Real encoding encodes each hyperparameter of the neural network as a real number, and then concatenates all the real numbers into a long vector as a chromosome. Where, each hyperparameter on a chromosome is called a gene. The initialization of the chromosome structure of Swin-YOLO in GA optimization is shown in FIG 1 below. In the figure, SC1, SC2 and SC3 represent the number of W-MSA submodules and SW-MSA submodules of SwinTR module in layers 16, 21 and 25 respectively, and WZ represents the size of partition window in winTR module. DR, ADR, and DPR represent the Dropout probability of MLP layer, attention layer, and residual connection in SwinTR module, respectively.

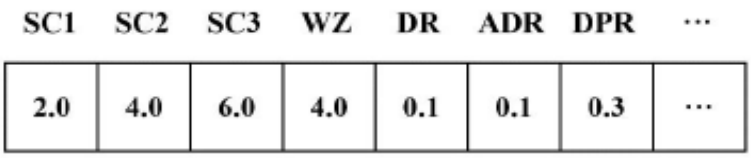

Figure 1. Initialization of the chromosome structure in the genetic algorithm optimization using Swin-YOLO

3.2. Individual fitness value evaluation

In the genetic algorithm, for each individual, it is necessary to give the individual fitness value evaluation according to its performance. The purpose is to select a certain individual of the population in order to reproduce the next generation.

In neural networks, individual fitness evaluation specifically refers to the training of each set of hyperparameters during the optimization process, and the network model is put on the validation set for testing after the training is completed. The individual fitness value evaluation in the neural network is realized by the fitness() function, which balances the four evaluation indexes of Precision, Recall, mAP@0.5, mAP@0.5:0.95 to calculate the final fitness score. FIG 2 shows the calculation process of the individual fitness value evaluation of the neural network.

Figure 2 Calculation process of individual fitness value evaluation of neural network

3.3. Updating individuals

In this paper, the selection operator and mutation operator in genetic algorithm are used to update the Swin-YOLO network model individually.

(1) Selection process: The selection process in this paper combines two selection operators: truncated selection and roulette wheel selection. In order to prevent the high computational cost caused by the large population size of the proposed GA after several iterations, the truncated selection operator is preferentially used in the selection process. By filtering out the top n individuals with the highest fitness value in the population and eliminating the rest of the individuals, the number of individuals participating in the subsequent update process is limited. Then, the truncated selected individuals were selected by roulette wheel, and the probability weights were given according to the fitness value, and an individual was randomly selected. The purpose is to give individuals with higher fitness values a greater chance of being selected and passing on genetic material to the next generation. The selection process of GA in Ann is shown in FIG. 3.

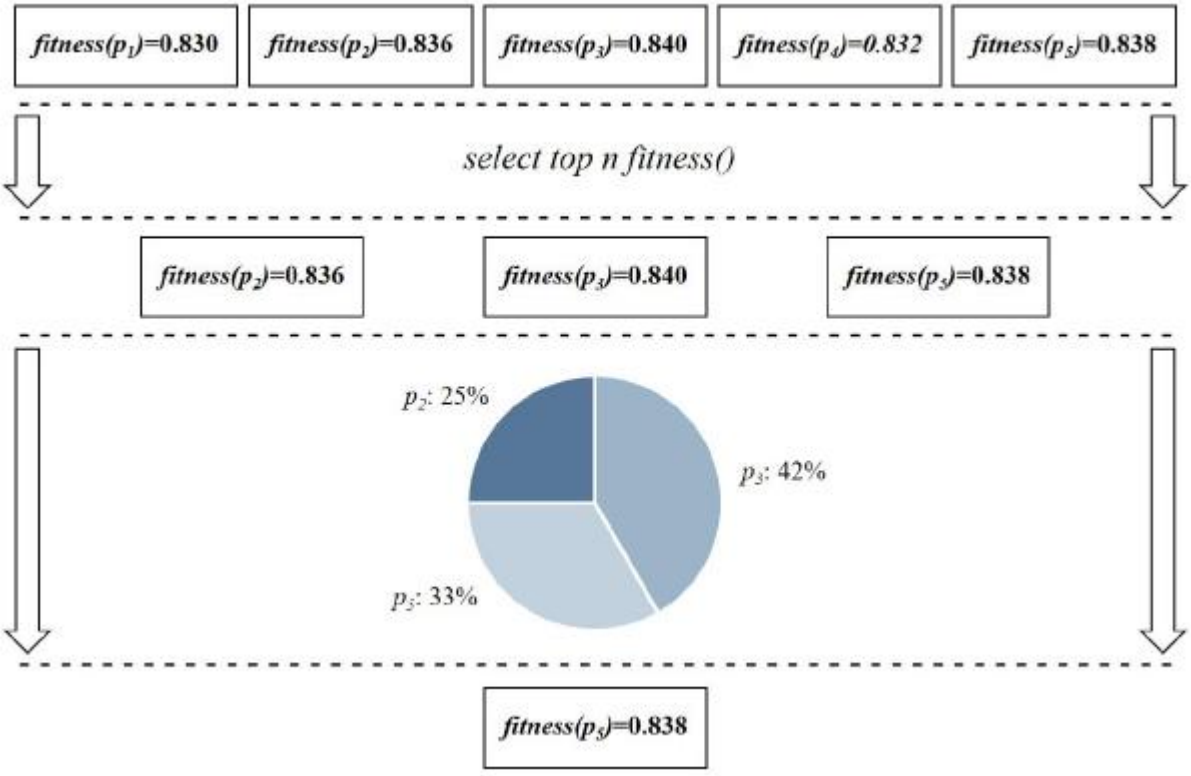

Figure 3. Diagram of the selection process of Swin-YOLO in GA

(2) Mutation process: The method of real number coding is selected to optimize the genetic algorithm of Swin-YOLO in this paper. The commonly used real number coding mutation methods include real number mutation, normal mutation, non-consistency mutation and adaptive mutation. Due to the unimodal and symmetrical nature of normal mutation, the mutated individual is easier to approach the optimal solution, so this method is selected for mutation in this paper. Firstly, the mutation operator set the mutation scale, mutation probability mp and standard deviation s, and generated a normal distribution with mean 0 and standard deviation s. According to the mutation scale, mutation

probability, normal distribution to determine whether mutation and mutation value size. If a mutation occurs, this mutation value is added to the original hyperparameter to obtain the new mutated hyperparameter. In the mutation process, at least one parameter of a set of hyperparameters will be mutated, otherwise the process of normal mutation will be repeated until the mutation occurs. The mutation process of Swin-YOLO in GA is shown in FIG 4.

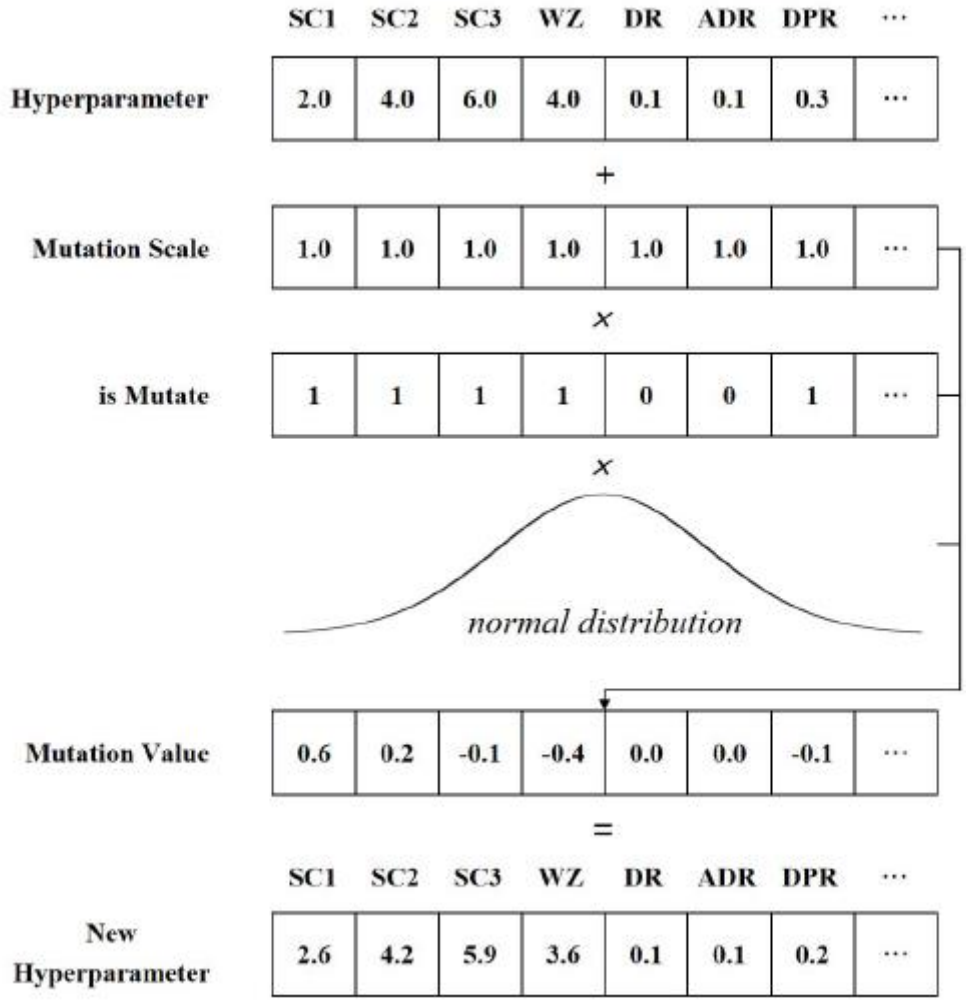

Figure 4. Mutation process diagram of Swin-YOLO in GA

The first row of chromosomes in the figure represents a set of hyperparameters obtained after selection, and according to the mutation probability mp, DR And ADR are randomly determined not to mutate in this round. For the other hyperparameters, the obtained normal distributed random value is multiplied with its variation scale to obtain the variation value. Finally, the new hyperparameter list was obtained by adding the hyperparameter list of the first row with the mutation value array.

4. MAKING OF THE TOMATO DATASET

4.1. Image acquisition

In the image acquisition work, cameras, mobile phones and other devices were used to take pictures of tomato plants in multiple tomato planting bases. After filtering out poor quality images, 1810 tomato images were obtained. The image format includes PNG and JPG, and the image resolution has four specifications: 400×500, 3000×3000, 3024×4032, and 3120×4160. Some of the images from the tomato dataset are shown in Figure 5.

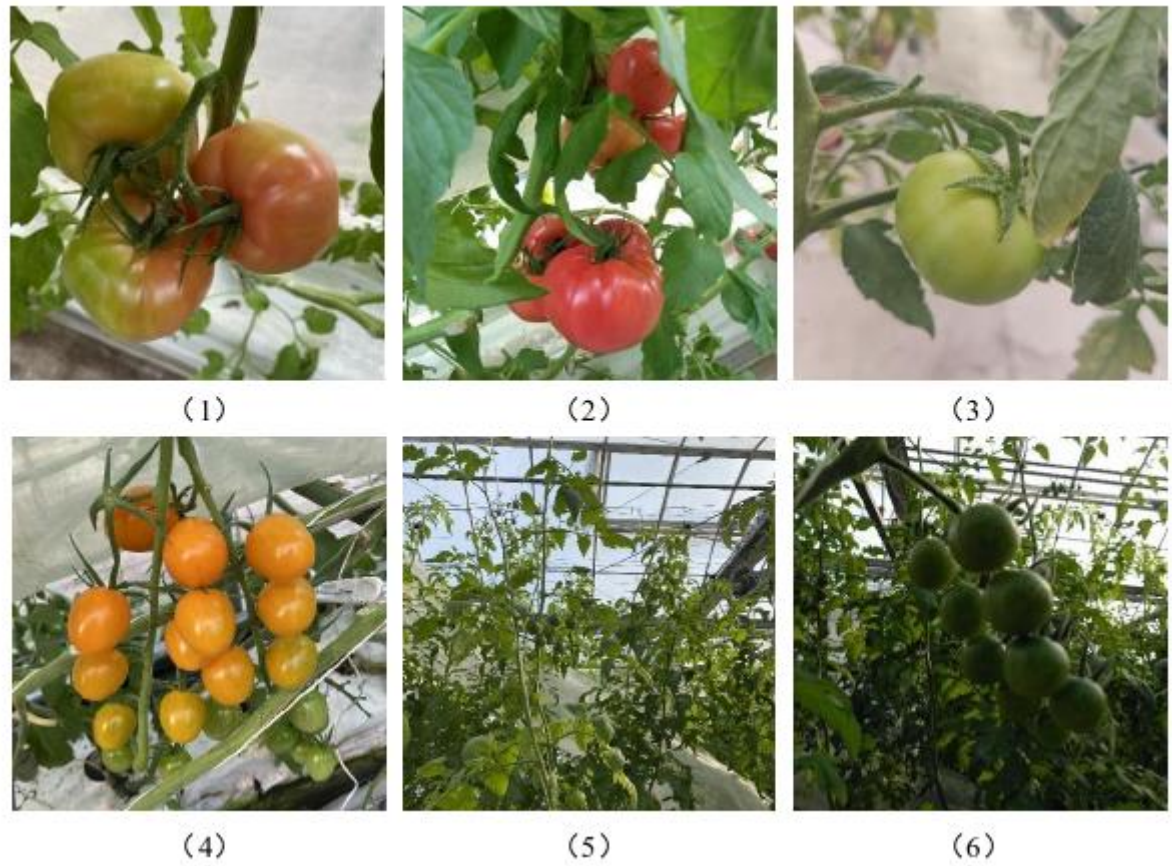

Figure 5. Partial images of the tomato dataset: (1) no occlusion; (2) there is occlusion; (3) single target; (4) multi-objective;5.Small goals; (6) Lack of light

4.2. Dataset Production

Based on the self-collected tomato images and public data, the collected 1810 tomato images were divided according to the standard that the training set accounted for 70% and the test set accounted for 30%, and the required data set was obtained, including 1267 training sets and 543 test sets. labelImg was used to manually label the dataset, producing an XML file with a 4-tuple argument list specifying the four coordinate pairs of a labeled object bounding box. FIG. 6 shows the sample tomato image labeled with labelImg and its corresponding XML file. This dataset contains 21020 labeled samples, including 14899 in the training set and 6121 in the test set.

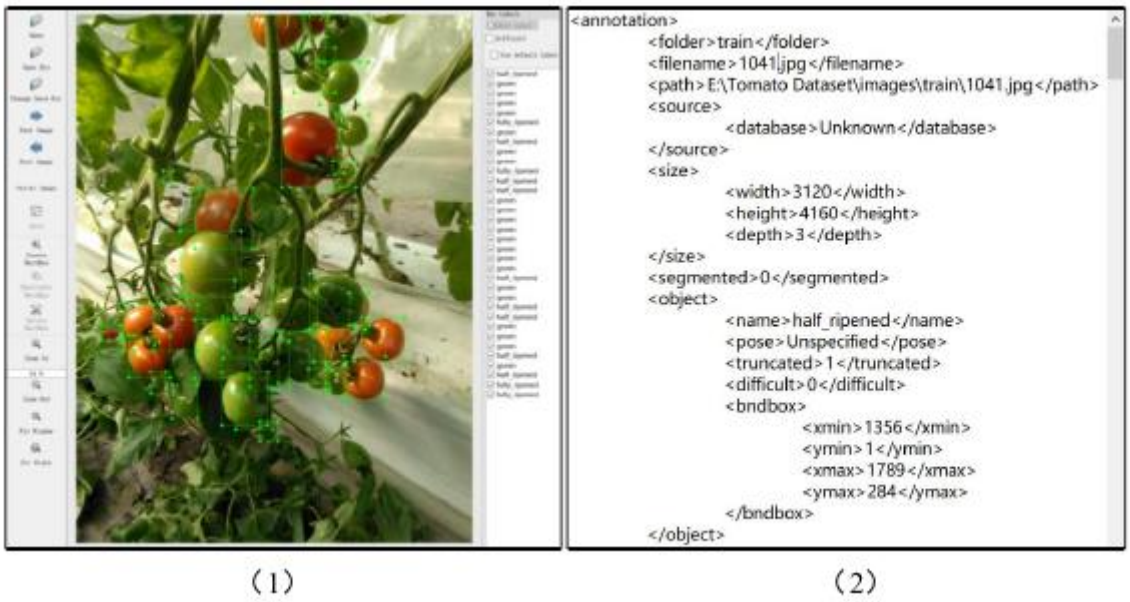

Figure 6. Labelimg labeling tomato images: (1) labeling process; (2)xml files

In order to verify the quality of the produced tomato data set, the distribution of labeled size of tomato fruits in the data set is statistically analyzed, as shown in Figure 7. In this distribution map, tomato fruit targets are divided into three categories: small targets, medium targets, and large targets. The classification criterion is determined based on the proportion of the area of the labeled box to the total area of the graph. When the area of the annotated box accounted for less than 0.3% of the total area, it was defined as a small target. Between 0.3% and 2.7%, defined as the medium target; When it is greater than 2.7%, it is defined as a large target. As you can see from the figure, the distribution of label sizes in the training and test sets is basically the same. The number of small targets in the whole data set is as many as 8776, accounting for 41.6% of all targets, which is only slightly lower than the medium target. Therefore, this tomato data set has certain difficulties in recognition, and requires high discrimination and recognition ability of the network model.

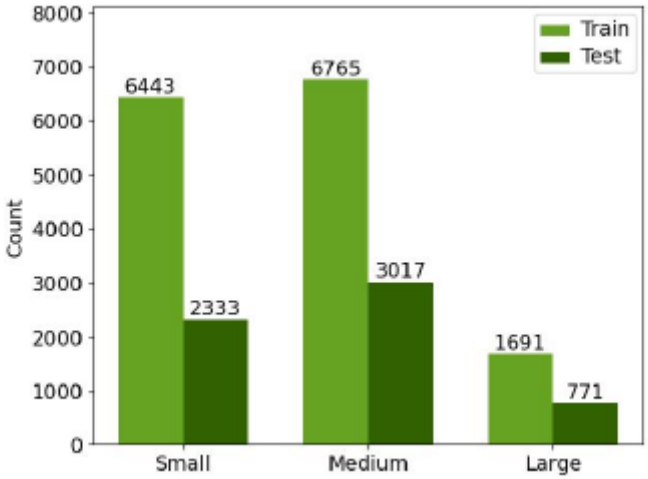

Figure 7. Target label size distribution plot for the tomato dataset

In the process of dataset annotation, according to the different maturity of tomato fruits, it is marked as unripe, semi-ripe and ripe, and the number of these three types of tomato fruits in the training set and test set is shown in Table 2 below.

Table 2. Number of tomatoes in the dataset for each of the three ripens

	Crudity	Half-ripe	Ripe
Training set	8294	3749	2856
Test set	3381	1599	1141

5. SWINGA-YOLO NETWORK ARCHITECTURE

Based on the characteristics of deep learning object detection algorithms, this chapter proposes a sliding window self-attention tomato fruit detection model based on GA optimization, named SwinGA-YOLO. On the basis of Swin-YOLO, the model uses genetic algorithm to optimize the hyperparameters of the network to further improve the adaptability of the network to the tomato data set. The default hyperparameter file hyp.scratch-low.yaml of YOLOv5s model and the structure hyperparameters of Swin-YOLO were used as initial values, and the Swin-YOLO was optimized. The number of rounds of optimization was 300, and 35 items of optimized hyperparameters were selected. The list of hyperparameter optimizations is shown in Table 3.

Table 3. List of hyperparameter optimizations

Hyper-parameters	Initial value	Evolutionary scope	Optimal value
lr0	0.001	[1e-5， 0.1]	0.00115
lrf	0.001	[0.01， 1.0]	0.01195
momentum	0.937	[0.6， 0.98]	0.98
weight_decay	0.0005	[0.0， 0.001]	0.0054
warmup_epochs	3.0	[0.0， 0.5]	1.9015
warmup_monmentum	0.8	[0.0， 0.95]	0.95
warmup_bias_lr	0.1	[0.0， 0.2]	0.14629
box	0.05	[0.02， 0.2]	0.04852
cls	0.5	[0.2， 4.0]	0.48968
cls_pw	1.0	[0.5， 2.0]	0.7401
obj	1.0	[0.2， 4.0]	0.75538
obj_pw	1.0	[0.5， 2.0]	0.90761
anchor_t	4.0	[2.0， 8.0]	2.4482
hav-h	0.015	[0.0， 0.1]	0.01727
hav-s	0.7	[0.0， 0.9]	0.42075
hav-v	0.4	[0.0， 0.9]	0.27128
degrees	0.0	[0.0， 45.0]	0.0
translate	0.1	[0.0， 0.9]	0.13987
scale	0.5	[0.0， 0.9]	0.69297
shear	0.0	[0.0， 10.0]	0.0
flipud	0.0	[0.0， 1.0]	0.0
mosaic	1.0	[0.0， 1.0]	0.60018
mixup	0.0	[0.0， 1.0]	0.0
copy_paste	0.0	[0.0， 1.0]	0.0
swin_1	2	[0.0， 10.0]	3.591
swin_2	4	[0.0， 10.0]	4.7001

Hyper-parameters	Initial value	Evolutionary scope	Optimal value
Swin window size	44	[2.0, 8.0]	3.2557
Swin drop rate	0.1	[0.0, 0.5]	0.12127
Swin attn drop rate	0.1	[0.0, 0.5]	0.07758
Swin drop pathr ate	0.3	[0.0, 0.5]	0.2856

The obtained optimal values of some hyperparameters were rounded by the round() function, and the total number of W-MSA submodules and SW-MSA submodules of the 16th, 21st, and 25th layer SwinTR module were finally determined to be 4, 5, and 4, respectively, and the window partition size was set to 3.

6. EXPERIMENT AND RESULT ANALYSIS

6.1. Experimental Setup

In order to ensure the fairness of ablation experiments and comparison experiments, self-made tomato data sets are used for training and testing in the following experiments, and run on this equipment: Processor Intel(R) Core(TM) i7-7800X CPU @3.50GHz, graphics card NVIDIA GeForce GTX 3090, running memory 32GB, hard disk capacity 1.25T, Windows10 64-bit operating system.

6.2. Transfer learning and fine-tuning

Transfer Learning is a method of using existing domain knowledge to solve a problem in a related but different domain. In the training of neural network, it accelerates the training of new network model by transferring the parameters of the already trained model, which avoids the problem that the network learns from zero, which is difficult and slow. At the same time, transfer learning also solves the problem of insufficient training samples that may exist in self-made datasets. Therefore, the SSD, Faster R-CNN, YOLOv3, and YOLOv5 network models trained on COCO were transferred to the tomato dataset for training.

The size, quantity, and quality of the images in the dataset affect the setting of the training parameters. Since the parameters of these networks are not directly set based on the tomato dataset, we need to do some fine-tuning of the network parameters: change the output class of the network to 3, indicating unripe, medium ripe, and ripe fruit; The number of training iterations was set to 200. The training batch size is set to 16.

6.3. The SwinGA-YOLO ablation experiment

In order to verify the effectiveness of the improved strategy proposed in this paper for YOLOv5 in tomato detection, the ablation test will be carried out on the tomato dataset. Transfer learning and fine-tuning are used in the experimental process, and the weights of YOLOv5s model pre-trained on COCO dataset are used to accelerate the convergence of the model. Based on the original network, we improve the following five aspects: (1) add a sliding window self-attention SwinTR module to the Neck region; (2) Modify the Concat structure of the 19th layer of the original network into three branches; (3) Replace the original loss function with SIoU; (4) Optimize the network structure by GA; (5) Weights pre-trained on COCO dataset using GA optimized network structure. Tables 4 show the results of the SwinGA-YOLO ablation experiments, where "√" indicates that the corresponding improvement is used and "-" indicates that the corresponding improvement is not used.Other parameters are kept unchanged during training.

Table 4. Swinga-Yolo ablation experiments

Self attention	Three branches	SIoU	GA optimization	New weight	Accuracy	recall rate	Average precision
-	-	-	-	-	81.6	74.7	81.8
√	-	-	-	-	80.3	77.6	82.7
-	√	-	-	-	80.9	76.7	82.6
-	-	√	-	-	81.3	75.0	82.1
√	√	-	-	-	83.4	76.5	83.0
√	√	√	-	-	81.9	77.4	83.0
-	√	√	-	-	84.3	76.2	82.8
√	√	√	-	-	82.8	76.6	83.1
√	√	√	√	-	84.0	76.4	84.3
√	√	√	√	√	84.2	77.3	84.7

It can be seen from the above table that when the sliding window self-attention module, the three-branch Concat structure, and the SIoU loss function are used to improve the original network separately, the precision rate will be reduced and the recall rate will be improved. At the same time, the average precision of the network is increased by 0.9 percentage points, 0.8 percentage points and 0.3 percentage points respectively. Sliding window self-attention helps the network capture broader contextual information, while the three-branch Concat structure provides richer feature representation. These improvements help the network to better understand the semantics of the image and the context of the target object, thereby improving the recall rate of tomato fruit detection. The SIoU loss function further improves the recall rate of object detection by improving the regression quality of the object box. However, when using the sliding window self-attention module and the three-branch Concat structure at the same time, the average accuracy of recognition is only improved by 1.2 percentage points, which is less than the theoretical additive value when improved separately. The analysis shows that the use of sliding window self-attention mechanism in Neck can improve the ability of the network model to collect and integrate feature information in the post-processing stage, and pay attention to the semantic information that is easy to be ignored. The Concat structure is modified at the 19th layer, which also acts in the deep layer of the network model. And the improved target problem of adding a sliding window attention module, strengthening the communication between location information and semantic information, and cross-complemecting with the sliding window attention mechanism. Under the action of the two, the precision and recall rate are improved compared with the original network. Similarly, when the three improvement methods are used together, they can make up for each other and improve the precision, recall and average precision, reaching 83.1% recognition accuracy on the tomato dataset. This model is the Samin-YOLO network introduced in Chapter III. After the GA optimization algorithm is added, the fitness of the network model to the tomato data set is greatly strengthened by optimizing the network structure and other hyperparameters, and a new network model SwinGA-YOLO is obtained, which is in line with tomato fruit detection. The recognition effect of the network model is increased by 1.2 percentage points compared with that before optimization.

Since the pre-trained weights of the original network are used in the above experiments, which are different from the network parameters of SwinGA-YOLO, the network weights are retrained according to the network structure of SwinGA-YOLO and applied to the tomato dataset. Finally, under the joint use of the five improved methods, the recognition effect achieves the best performance, with an average accuracy of 84.7%. Compared with the original network YOLOv5, the accuracy and

recall rate are increased by 2.6 percentage points, and the average accuracy is increased by 2.9 percentage points.

6.4. Comparison of tomato detection performance of different network models

In order to verify the effectiveness and superiority of the SwinGA-YOLO network model proposed in this paper for tomato detection, the mainstream object detection models SSD, Faster R-CNN, YOLOv3, YOLOv5m, YOLOv5s are selected to compare with them on the tomato dataset. During the experiment, each detection model used its own pre-trained weight. The YOLO series model used the latest version provided by Ultralytics, and the SSD and Faster R-CNN models were trained based on the MMDetection detection library. The performance pairs of tomato detection for different network models are shown in Table 5.

Table 5 Comparison of tomato detection performance of different network models

Network model	Input Size/pixel	Model size/M	Detection speed/ms	Average precision/%
SSD	512	24.39	24.9	74.4
Faster R-CNN	640	60.11	40.7	76.1
YOLOv3	640	61.53	12.5	84.6
YOLOv5m	640	21.02	8.9	85.2
YOLOv5s	640	7.02	5.5	81.8
Swin-YOLO	640	13.02	7.2	83.1
SwinGA YOLO	640	12.04	6.9	84.7

It can be seen from the table that SwinGA-YOLO, the network model proposed in this study, has obvious advantages in model size, detection speed and detection accuracy compared with SSD and Faster R-CNN. Compared with YOLOv3, SwinGA-YOLO has the same detection performance, but the model size is only 1/5 of YOLOv3, and the detection speed is nearly doubled. Compared with YOLOv5m target detection model, SwinGA-YOLO reduces the size of the network model by about 43%, greatly saves memory overhead, and improves the detection speed by 22%, with the average detection accuracy only slightly lower by 0.5%. Considering model size, detection speed, and average precision, SwinGA-YOLO adds some performance overhead in exchange for more performance improvement. Compared with Swin-YOLO, SwinGA-YOLO optimized by genetic algorithm has a smaller model, faster detection speed, and better recognition effect on tomato fruits. In the field of tomato fruit target detection, SwinGA-YOLO is a target detection model with relatively stronger comprehensive performance, which can provide more choices for the research and development of tomato intelligent equipment.

Comparison before and after network improvement under different occlusion conditions

In order to study the performance of the improved model under different occlusion conditions, it was decided to analyze from two aspects: the degree of fruit occlusion and the relation of fruit occlusion.

(1) Fruit occlusion degree: The tomato test set is divided into a less occlusion test set and a more occlusion test set. Images with average tomato fruit occlusion rate less than or equal to 30% were classified as less occlusion, and images with average tomato fruit occlusion rate greater than 30% were classified as more occlusion. The test sets with less occlusion and more occlusion contain 343 and 200 data respectively. The network models before and after improvement were tested on the less occlusion and more occlusion test sets respectively, and the results are shown in Table 6.

Table 6 Comparison of network before and after improvement under different degrees of occlusion

Network model	Be lessshelteredl/%	Shelter from view/%
YOLOv5s	84.2	77.7
SwinGA-YOLO	86.3	81.9

As can be seen from the table, YOLOv5s achieves 84.2% and 77.7% recognition accuracy with less and more occlusion, respectively, while SwinGA-YOLO achieves 86.3% and 81.9% recognition accuracy with less and more occlusion, respectively. Compared with YOLOv5s, SwinGA-YOLO has the fruit recognition accuracy improved by 2.1 percentage points and 4.2 percentage points respectively in the cases of less occlusion and more occlusion.

Therefore, according to the above experimental results, it can be concluded that the SwinGA-YOLO network model has better recognition ability in the case of serious fruit occlusion, and can better deal with more complex scenes.

(2) Fruit occlusion relationship: In order to further explore the performance differences between the SwinGA-YOLO network proposed in this paper and the original network in different occlusion relationships, 50 images with only inter-fruit occlusion, background occlusion, and compound occlusion are selected from the test set, respectively, where the compound occlusion refers to the presence of both inter-fruit occlusion and background occlusion. FIG. 8 shows an example of a tomato picture under three occlusion relations.

Figure 8. Annotated tomato images with three occlusion relationships: (1) occlusion between fruits; (2) background occlusion; (3) Compound occlusion

The comparison between before and after network improvement under different occlusion relationships is shown in Table7.

Table 7 Comparison of network before and after improvement under different occlusion relationships

Network mode l	Fruit shade/%	Background masking/%	Composite shelter/%
YOLOv5s	90.8	85.4	79.3
SwinGA-YOLO	91.5	89.6	82.0

It can be seen from the table that the recognition rates of SwinGA-YOLO network in inter-fruit occlusion, background occlusion, and composite occlusion are 91.5%, 89.6%, and 82.0%, respectively, which are 0.7%, 4.2%, and 2.7% higher than those of YOLOv5s. The network improvement accuracy of SwinGA-YOLO under composite occlusion is equivalent to the improvement effect of the complete test set, which can roughly prove that this data set generally contains a variety of occlusions and has high complexity. The recognition rate of the two networks in the occlusion between fruits is generally high, and the degree of improvement is relatively small. SwinGA-YOLO has a great improvement in background occlusion, and can effectively resist recognition disturbances caused by complex environments.

6.5. Comparison before and after network improvement under different maturity levels

In order to verify the detection performance of the improved network model on different maturity tomato fruits, the recognition rates of tomato fruits in the test set in the immature, semi-ripe and mature stages will be counted respectively, as shown in Table 8.

Table 8 Comparison of network before and after improvement under different fruit maturity

Network model	Unripe fruit/%	Partially ripe fruit/%	Ripe fruit/%
YOLOv5s	84.6	75.9	84.7
SwinGA-YOLO	86.6	79.4	87.9

It can be seen from Table 8 that SwinGA-YOLO has improved the recognition accuracy under different fruit maturity compared with the original model YOLOv5s. Among them, the recognition accuracy of unripe fruit, semi-ripe fruit trees, and ripe fruit reaches 86.6%, 79.4%, and 87.9%, respectively, which is 2.0%, 3.5%, and 3.2% higher than that of YOLOv5s.

The semi-ripe period is between immature and mature, which is a difficult developmental stage to define. SwinGA-YOLO has the largest improvement in the recognition accuracy of fruit semi-ripe time, which proves that the network contains more abundant semantic information and has stronger discrimination in complex classification problems that are easy to confuse.

6.6. Comparison of visualization effects before and after model improvement

In order to feel the difference before and after the improved model more intuitively, the prediction results of YOLOv5s and SwinGA-YOLO on the test set are output. The following show only partial results, as shown in Figure 9.

(1) (2) (3)

(4) (5) (6)

Figure 9. Comparison of visualization effects before and after model improvement: (1), (2), and (3) are the detection effects of YOLOv5s;(4), (5), and (6) are the detection renderings of SwinGA-YOLO

In Figure (1), YOLOv5s mistakenly detected leaves as tomato fruits, and could not distinguish the maturity of some tomato fruits in the figure, which did not eliminate redundant predictions well. SwinGA-YOLO avoided these problems. In Figure (2), YOLOv5s incorrectly predicted small green branches and stems as immature tomato fruits, and there were also some cases of wrong prediction of fruit maturity. However, in SwinGA-YOLO, there was only one wrong prediction of fruit maturity, which greatly reduced the original problem. In Figure (3), YOLOv5s has poor recognition effect on fruits with background occlusion, missed detection occurs many times, and the leaves are predicted as immature tomato fruits again. At the same time, it focuses on unlabeled super-vision targets, indicating that the fitting ability of the data set is weak.

Combined with the above detection comparison figure, it can be roughly seen that YOLOv5s and SwinGA-YOLO perform similar when tomato fruit is a large target, less occlusion and easy to distinguish maturity. When the target is small, the degree of occlusion is large or the maturity is difficult to define, YOLOv5s is prone to miss detection and false detection to a certain extent. SwinGA-YOLO can better improve these problems and has stronger robustness.

7. SUMMARY

This paper proposes a sliding window self-attention YOLO tomato target detection algorithm based on GA optimization. In order to improve the fitness of the model in the tomato data set, the hyperparameters of the model are optimized. The filtered and annotated tomato images are used for training and testing of the experiment. Experimental results show that the accuracy of the improved model SwinGA-YOLO on the tomato dataset can reach 84.7%, which is 2.9%higher than the original model YOLOv5s. Compared with SSD, Fast-RCNN, YOLOv3, YOLOv5m, the superiority of SwinGA-YOLO can also be found. SwinGA-YOLO performs well in complex scenes, and effectively improves the problem of missed detection and false detection in the tomato detection process. When the average occlusion rate of tomato fruit exceeds 30%, the detection accuracy is improved by 4.2%. When the tomato fruit is occluded by the background, the recognition rate is also improved by 4.2%. In different stages of tomato development, the recognition rates of unripe, semi-ripe and ripe fruits were increased by 2.0%, 3.5% and 3.2%, respectively.

CONFLICTS OF INTEREST

The authors of this study declare that there is no direct or indirect commercial conflict of interest in the research, writing and publication of this paper. This paper is supported by the research group,

subject number 2023KJ07: Research on key technologies of machine vision for the whole industry chain of tomato.

ACKNOWLEDGEMENTS

I would like to extend my deepest appreciation to the professors and staff at the Eastern University of the Philippines for their unwavering support and assistance during the course of my research and writing of this paper. Their expertise and insights have been incredibly valuable to the success of this project.

REFERENCES

[1] Yan Gong, Suzhou Institute of Biomedical Engineering and Technology, Chinese Academy of Sciences, et al. "Swin-Transformer-Based YOLOv5 for Small-Object Detection in Remote Sensing Images", Sensors, Vol. 23, No. 7, pp. 3634, 2023. https://doi.org/10.3390/s23073634

[2] Gi-Sang Choi, Department of Electrical and Computer Engineering, Graduate School, University of Seoul, et al. "Improved YOLOv5: Efficient Object Detection Using Drone Images under Various Conditions", Applied Sciences, Vol. 12, No. 14, pp. 7255, 2022. https://doi.org/10.3390/app12147255

[3] YANG R, HU Y, YAO Y, et al. Fruit target detection based on BCo-YOLOv5 model. Mobile Information Systems, 2022, 2022: 1-8.

[4] ZHANG W, XIA X, DU J, et al. Recognition and detection of wolfberry in the naturalbackground based on improved YOLOv5 network. 2022 3rd International Conference on Computer Vision, Image and Deep Learning&International Conference on Computer Engineering and Applications(CVIDL&ICCEA), 2022: 256-262.

[5] WANG J, ZHANG Z, LUO L, et al. SwinGD: A Robust Grape Bunch Detection Model Based on Swin Transformer in Complex Vineyard Environment[J]. Horticulturae, 2021, 7(11).

[6] ZHENG Z, WANG P, LIU W, et al. Distance-IoU loss: Faster and better learning for bounding box regression[C]. Proceedings of the AAAI conference on artificial intelligence, 2020, 34(07): 12993-13000.

[7] ZHENG Z, WANG P, REN D, et al. Enhancing geometric factors in model learning and inference for object detection and instance segmentation[J]. IEEE Transactions on Cybernetics, 2021, 52(8): 8574-8586.

[8] CARON M, TOUVRON H, MISRA I, et al. Emerging properties in self-supervised vision transformers. Proceedings of the IEEE/CVF international conference on computer vision, 2021: 9650-9660.

[9] RANFTL R, BOCHKOVSKIY A, KOLTUN V, et al. Vision Transformers for Dense Prediction. 18th IEEE/CVF International Conference on Computer Vision(ICCV), 2021: 12159-12168.

[10] WANG C-Y, LIAO H-Y M, WU Y-H, et al. CSPNet: A new backbone that can enhance learning capability of CNN. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, 2020: 390-391.

[11] LIU Z, LIN Y, CAO Y, et al. Swin transformer: Hierarchical vision transformer using shifted windows. Proceedings of the IEEE/CVF international conference on computer vision, 2021:10012-10022.

[12] DOSOVITSKIY A, BEYER L, KOLESNIKOV A, et al. An image is worth 16x16 words: Transformers for image recognition at scale.ar Xiv preprint ar Xiv: 201011929, 2020.

[13] PAN X, GE C, LU R et al. On the integration of self-attention and convolution. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022: 815-825.