

AI-based Detection of Fully Occluded Tomatoes Empowered by Pixel-based Lock-In-Amplifier and Synthetic Feature Injection

Aiden Xu

Winter Springs High School, 130 Tuskawilla Rd, Winter Springs, FL 32708, USA; goaidenxu@gmail.com

Mentor:

ABSTRACT: As robotic operations emerge in agricultural fields, the limited perception capability of robots in detecting fully occluded fruits remains a major obstacle to efficient harvesting, accurate yield prediction, and other field operations. It was previously shown that reflected electromagnetic wave signals are different between scenarios with and without fully occluded fruits. However, three challenges persist: the baseline scenario (i.e., the scenario without fully hidden fruits) is assumed to be known a priori; the signal difference is very small and embedded in a high level of noise; and no visible boundary can be seen with the unaided eye. In this paper, two image pre-processing techniques are proposed to aid classification and object detection tasks. First, a pixel-based lock-in amplifier (PLIA) is investigated to extract weak near-infrared (NIR) signals from experimental videos and output images. Then, the average RGB values in those images are embedded as synthetic features to enhance the performance of the classification and object detection models. Seventy-seven experiments were conducted in a mock tomato orchard setup, and NIR videos were collected, processed, and labeled. Two classification models (YOLOv11 and EfficientNet) and three object detection models (YOLOv11, YOLOv12, and RF-DETR) are trained, and their performances are compared. It is shown that all classification models achieve a precision above 70%, while the mAP50 values of the detection models exceed 50%. The proposed PLIA and synthetic feature injection (SFI) strategies improve the classification and object detection performance, as demonstrated in ablation tests, while the trained models are lightweight and easily deployable.

KEYWORDS: Applied AI Models, Fully Occluded Fruit Detection and Classification, Image/Video Pre-processing, Near Infrared, Precision Agriculture.

■ Introduction

Robotics has been utilized in many agricultural applications along with the rise of Agriculture 4.0.¹ Tasks such as sowing,² irrigation, monitoring,³ yield prediction,⁴ disease classification,⁵ and harvesting,⁶ among a myriad of other tasks, have seen varying degrees of automation to streamline and enhance their processes. High-value fruit crops, like apples, strawberries, and tomatoes,⁷ in particular, have received significant research attention for the automation of their farming processes due to their labor-intensive nature. However, in order for these robots to be effective, they need unprecedented perception capabilities so that they can find fruits, either fully exposed, partially occluded, or fully obscured. Please note that "fully obscured" means that a fruit is completely behind leaves and that none of its parts can be observed by an RGB camera from any viewpoint. Finding them and in some cases classifying them are crucial for some robotic operations such as monitoring growth status,⁸ managing disease conditions,⁹ checking nutrient deficiency,¹⁰ conducting yield prediction,¹¹ and harvesting.¹²

Computer vision has been widely used in detecting and classifying fruits based on RGB images.¹³ Duong *et al.*¹⁴ compared EfficientNet,¹⁵ MixNet, and ConvNet, achieving an accuracy of 99.98%, 99.97%, and 94.52%, respectively, in the task of finding a variety of 25 different fruits and vegetables. As another example, in,¹⁶ the detection of tomatoes achieved an average precision of 99.4% using the modified YOLOv3 model. Another study¹⁷ focused on determining strawberry ripeness and

found that a combination of YOLOv8 and Lightweight-Swin Transformer can reach an accuracy of 94.4%. Xiao *et al.*¹⁸ utilized a vision transformer (Swin Transformer) and a multilayer perceptron to identify fruit ripeness and accomplished mAP50:95 values of 46.5%, 84.7%, and 83.9%, respectively. Additionally, Sa *et al.*¹⁹ employed a YOLOv5 detection model and reached an mAP50:95 of 81.2%.

All the above-mentioned computer vision models focused on finding fully exposed or partially occluded fruits. These models detect and classify fruits based on the surface information of an image only. However, as seen in Figure 1, relying only on RGB images cannot detect fruits that are fully occluded. In many commercial farms, such as fresh market tomato farms, a significant amount of fruits are hidden inside dense leaves, and only counting the fruits that are visible to RGB cameras is not enough for many robotic operations, such as yield prediction and harvesting. Additionally, the approach of using cameras from different angles is not feasible, as leaves outside of the fruits are densely distributed.

In this study, a pixel-based lock-in-amplifier (PLIA) method and a synthetic feature injection (SFI) method are combined as an image/video processing method for the captured near infrared videos in seventy-seven experiments conducted in a mock tomato orchard setup. Then, two classification models (YOLOv11²⁰ and EfficientNet¹⁵) and three object detection models (YOLOv11, YOLOv12²⁰, and RF-DETR²¹) are custom-tuned and trained based on 875 processed images, trying

to classify whether or not there are tomatoes fully occluded by leaves, as well as using bounding boxes to locate where they are.



Figure 1: Two RGB images in a mock tomato orchard setup include tomatoes fully occluded by leaves, which are invisible to the unaided human eye. There are zero and two tomatoes hidden behind leaves in the left and right images, respectively.

The main contributions of this study are as follows:

- PLIA is developed to address the issue of reflected near-infrared (NIR) signals buried in a large magnitude of noise in captured images/videos. The modulation-demodulation-low pass filtering mechanism in the original lock-in amplifier (LIA) method²² can help remove noise in all harmonic frequencies other than the DC component, which is the actual reflected NIR signal. Here, the LIA is applied to each pixel of a video along the temporal direction, so-called PLIA, and the result is an image instead of a scalar value, as shown in,²³ enabling the utilization of advanced convolutional neural network (CNN) and vision transformer models for later classification and detection tasks.
- The average RGB value of the PLIA processed image is injected into the image as a synthetic feature, a method similar to synthetic feature injection,²⁴ to enhance the performance of AI models in classification and object detection.
- A total of 875 NIR images, either collected from experiments or augmented via Roboflow®, are labeled and used in the CNN and vision transformer models' training, validation, and testing.
- CNN and vision transformer models were custom-tuned and trained to classify and detect fully hidden fruits given NIR images. For classification tasks, an EfficientNet and a YOLOv11 model were customized, and for detection tasks, a YOLO11, YOLOv12, and RF-DETR model were customized. Their performances are compared in terms of precision, recall, and F1 score for classification, and precision, recall, and mAP50 values for object detection, respectively.

It is worth noting that this study is a significant extension of the work presented in.²³ As shown in,²³ the RGB values of LIA-processed NIR images are different between scenarios with and without fully hidden fruits. However, the method proposed therein is not implementable because it assumes that the RGB value of scenarios without fully hidden fruits is known *a priori*, which is not the case under real farm conditions.

■ Related Work

Finding fruits that are fully occluded is a very challenging task, and there are only a few attempts in this area of interest. These methods can be largely grouped into two categories: mechanical approaches and electromagnetic wave approaches.

Several researchers have used mechanical approaches trying to expose the fruits hidden behind. For example, Jordi Gené-Mola *et al.*²⁵ tried to use leaf blowers to part the leaves so that they could see the fully hidden fruits behind. However, this approach was not highly effective because the leaves on most commercial tomato farms are tightly bound by strings, preventing the leaf blower from doing its intended purpose.

Reflected electromagnetic radiation, captured by a detector and converted from photons into electronic signals, can be used to determine whether fruits are fully blocked by leaves. (i) X-rays can penetrate leaves; however, the high hardware and software requirements, along with the associated cost, make them impractical for farm applications.²⁶ In addition, radiation would require safety gear to be worn, and may also damage fruits.²⁶ (ii) Although LiDAR operates in the NIR or short-wave infrared (SWIR) spectrum, existing products focus only on detecting and rendering surface information and are not capable of detecting objects behind leaves.²⁷ (iii) Millimeter radar may be a feasible approach, which has been proposed to detect hidden tomatoes with limited success in a one tomato plant setup; although this approach relies upon sophisticated hardware and signal processing software.²⁸ (iv) Mid-wave IR (MWIR) and long-wave IR (LWIR), such as thermal imagers, focus on finding objects that have temperature differences from their background. This approach may not work due to the fact that fruits and leaves do not have obvious temperature differences in general.²⁹ (v) Near infrared is shown to have relatively high transmissivity and low absorption and reflectivity for leaves and, at the same time, relatively high reflectivity and low transmissivity and absorption for tomato fruits.²³ Therefore, as shown in,²³ the reflected NIR signal is different between the scenarios with and without fully occluded fruits. However, as shown in Figure 2, it is still a very challenging task to detect if there are fully hidden fruits. (vi) Lastly, ultrasound, as a mechanical wave instead of an electromagnetic wave, has been proposed to tackle this challenging problem.³⁰ However, ultrasound signals have an insurmountable amount of noise from reflections off leaves and surroundings, and are also affected by local humidity and temperature.³⁰ Thus, it is not a feasible solution.

In brief, to the best of the author's knowledge, no effective approach has been proposed, including mechanical approaches, multi-angle vision systems, and electromagnetic approaches, that can effectively identify fruits fully hidden behind leaves.

■ Problem Definition

Based on the literature review, it is a challenging task to classify and detect whether there are fully occluded tomatoes. In this study, three questions will be answered via experiments in a mock tomato orchard setup, where the NIR emitter, the sensor detector, and the mock tomato orchard are placed in proximity.

- **Question 1:** Can we achieve a 70% or higher precision when using the proposed methods to classify whether there are fully hidden fruits behind leaves?

• **Question 2:** Can we achieve a 50% or higher mAP50 value when using the proposed methods to detect the number and locations of fully hidden fruits behind leaves?

• **Question 3:** Can we achieve a higher performance when incorporating the proposed PLIA and SFI methods?

The thresholds set in Question 1 and Question 2 are relatively lower than those in typical classification and object detection tasks. The reason is that the task of finding fully hidden fruits is extremely challenging. According to the initial study, in²³ the RGB value difference in the raw NIR images between the cases with and without fully hidden fruits is very small, about 3% in average. Furthermore, as can be seen in Figure 2, there are no clearly visible, distinct boundaries of the fully hidden fruit.



Figure 2: NIR photons were detected by a cell phone camera, which generated the NIR images. There are two tomatoes fully hidden behind leaves in both the left and right images. The image on the left is the raw image, in which the fully hidden fruits cannot be seen. The center image shows the image after the PLIA preprocessing. The image on the far right is the corresponding RGB view of the other two images. Dotted lines on those images delineate the approximate locations of the two hidden tomatoes.

Overview of the Methods

The basic process of the method is shown in Figure 3. A group of thirty 940 nm diodes emits a pulse-width-modulated (PWM) modulated NIR signal towards the mock tomato orchard. Here, a special square PWM signal is generated in Arduino with 50% duty cycle and an amplitude between 0 and 1. Then the NIR wave propagates through the leaves and to the fruit(s) hidden behind if present. The NIR wave then reflects off the fully hidden fruit(s) and propagates back through the leaves. The photons are detected by a cellphone camera, which converts them digitally to an RGB video. This video then goes through the video pre-processing method of PLIA, which returns an image. Two filters are used in this step, including an average filter and a zero-phase low-pass filter (LPF). The SFI method is then applied to inject an additional visual clue. This processed image is then given to the custom-trained AI models, which will classify and detect fully hidden fruits. The result will determine the presence, number, and location of fully hidden fruits.

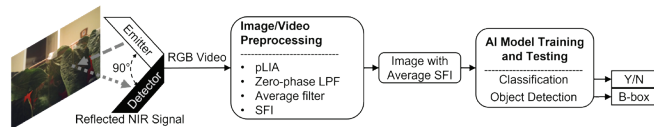


Figure 3: Overall description of the proposed method, including the experiment procedure, image/video pre-processing, and AI model training and testing.

Pre-Processing Methods

In the mock tomato orchard experiment, the PWM signal generation and video recording may have been slightly unsyn-

chronized. Therefore, before pre-processing, the experiment videos should be synchronized with the reference PWM signal.

1. PLIA for video processing:

The lock-in amplifier (LIA) method has been widely used in extracting weak signals from a noisy environment.²² Please note that the derivation here is similar to the original LIA,²² but is custom-designed and applied to each pixel of an image frame along the temporal direction in an experiment video.

Let us define $A_{i,j}^l(k)$ as the RGB intensity of the photon that is reflected from fully occluded fruits and reaches the detector in frame k of a video at the pixel location of (i, j) . Here, the RGB value means the sum of red, green, and blue values of a pixel in an image. $S_{i,j}^l(k)$ is $A_{i,j}^l(k)$ modulated with the PWM signal that is generated by the Arduino board as

$$S_{i,j}^l(k) = A_{i,j}^l(k) \left[\frac{1}{2} + \frac{2}{\pi} \sum_{n=1,3,5,\dots}^{\infty} \frac{1}{n} \sin(n\omega t_k) \right] \quad (1)$$

, in which the PWM signal is in its Fourier series expansion³¹ and t_k is the time associated with frame k . Also, let's define $X_{i,j}^l(k) = S_{i,j}^l(k) + n_{i,j}(k)$ where $n_{i,j}(k)$ is the noise captured by the detector and $X_{i,j}^l(k)$ is the photon signal detected by the camera at frame k and pixel location of (i, j) of a video.

The first step in PLIA is to demodulate $X_{i,j}^l(k)$ by multiplying it with the reference PWM signal, $r_{i,j}(k)$, in the form of a Fourier series for every pixel in an image frame of a video, along the temporal direction as

$$X_{i,j}(k)r_{i,j}(k) = [S_{i,j}^l(k) + n_{i,j}(k)] \times \left[\frac{1}{2} + \frac{2}{\pi} \sum_{m=1,3,5,\dots}^{\infty} \frac{1}{m} \sin(m\omega t_k) \right] \quad (2)$$

, the second line of which is the Fourier series of the reference PWM signal.³¹ Then some trigonometric operations will be conducted, and a zero-phase LPF is applied to remove those non-DC terms. After that the Basel's equation³² is used, the resultant value equals to $0.5A_{i,j}^l(k)$. Thus, we can extract the reflected photon signals from a high-noise environment.

This process will be repeated for every pixel of the experiment video. The achieved image is formed by averaging each achieved pixel value along the temporal direction, and the pixel value in the resultant image is

$$p_{i,j} = \sum_{k=1}^{N_f} 0.5A_{i,j}^l(k) / N_f \quad (3)$$

, in which N_f is the number of frames in this video. Since the reflected NIR photon is low, the converted RGB value $p_{i,j}$ is small. Therefore, to enhance the image quality, each pixel value of the image is scaled up so that the maximum $p_{i,j}$ in an image is near 255. The difference between a raw image and a PLIA processed image can be seen in Figure 2.

2. Synthetic feature injection:

As can be seen in Figure 2, the image obtained after processing an experiment video using PLIA is still not sufficiently distinct for reliable classification and object detection. As noted in,²³ the number of photons captured by the detector differs between scenarios with and without fully hidden fruits. Therefore, this characteristic is injected into the PLIA pre-processed NIR images.

Three types of SFIs were tried based on the RGB value of an image after PLIA: average SFI (the average RGB value of an image), maximum SFI (the maximum RGB value of an image), and maximum difference SFI (the maximum RGB value - the minimum RGB value). Among them, the average SFI is selected because the failure rate of indicating whether there are hidden fruits is the lowest when tried on all the collected NIR images. Let's define the total RGB value of an image to be $V_i = R_i + B_i + G_i$, where i is the i^{th} pixel in the image. Then the average SFI p_{sfi} is calculated as all V_i summed together and then divided by N , where N is the total number of pixels in an image. Since p_{sfi} in all the experiments fall within a certain range, this value is adjusted to be $a[p_{\text{sfi}} - p_{\text{base}}]$ to better scale the subsequently constructed black square box. Here, p_{base} and a are tuned based on the experiment data. This value is proportional to the side length of the black square placed at an appropriate point in the NIR image, and in this case, the top right corner.

One resultant image after these two image/video processing methods is shown in Figure 4. Since the average RGB value is proportional to the number of fully hidden fruits present, the SFI method gives additional artificial visual data injected directly into the images, thus improving the accuracy metrics of the AI models.

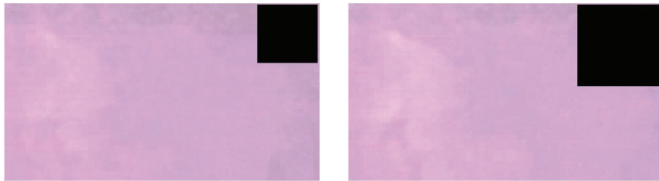


Figure 4: Images after the application of SFI. The average total RGB value is represented by the side length of the black square in the top right corner. This feeds additional visual information into the AI models, improving their performance. An image with one fruit is shown on the left, and an image with two fruits is shown on the right. There is a noticeable difference in the side lengths of the squares between the two.

3. Image/video pre-processing:

With the PLIA and SFI methods, the pre-processing steps are shown in Figure 5. Each video is downsized from 1920x1080 to 480x270 due to memory limits, as well as the computational cost consideration of the low-pass filter used. These downsized videos are converted to a 4-dimensional matrix for processing. Then, the signals are aligned to the reference PWM signal and then truncated to 4 periods. To reduce the memory needs and computational cost, only one period of the video, about 6 seconds, is used. The signals are passed to the PLIA method, which applies the LIA method to each pixel in the video and returns an RGB image. Each pixel is then multiplied by 255 over its maximum total RGB value. Then, they go through the SFI method, which injects additional visual information into the image in the form of a black square. The side length of this square is equal to 10 times the quantity of the average total RGB value minus 57, i.e., $p_{\text{base}} = 57$ and $a = 10$. One example image is shown in Figure 4.

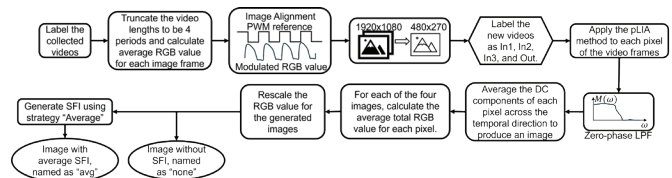


Figure 5: Procedure for image/video preprocessing. The outputs include eight pictures, with one set of four having 0, 1, 2, and 3 fruits, and the other set of four each with SFI.

■ Experiment and Video Collections

1. Testbed apparatus and setup:

The same apparatus as the one in²³ is used in this study, with a few modifications. The detailed construction of such a low-cost, mock tomato orchard and the emitter/detector therein can be found in.²³ To increase the number of photons captured by the camera, and meanwhile to maintain the low cost, thirty 940nm NIR diodes are integrated in the emitter, and all the circuits are on printed circuit boards to reduce the number of wires needed and save space. A cell phone camera is used to detect the reflected NIR wave.

2. Video collection:

The experiment procedure is shown in Figure 6. In each experiment, the leaves are taped to strings that go across the front to simulate the leaf cover. Up to three tomatoes are then placed just behind the leaf cover, with the tomatoes and leaves adjacent to each other and at a total distance of roughly 12 cm from the emitter. When the experiment starts, the emitter emits an NIR signal modulated with PWM. The PWM signal has a period of 6 seconds at a duty cycle of 50%. The camera detector (Motorola Moto G Pure[®]) detects the reflected NIR photons at a frame rate of 30 frames per second and digitally converts them into an RGB video. This experiment will repeat four cases with 0, 1, 2, and 3 fully hidden tomatoes, respectively.

A total of 77 valid experiments were conducted mainly in the evening to reduce the noise level. These generated images are processed with PLIA and SFI, then labeled using labelImg[®]. The initial YOLO dataset was copied, and that copy was reformatted to be compatible with EfficientNet and RF-DETR using the Roboflow[®] platform. The datasets are also augmented in Roboflow[®] using vertical and horizontal flips, as well as 90-degree rotations.

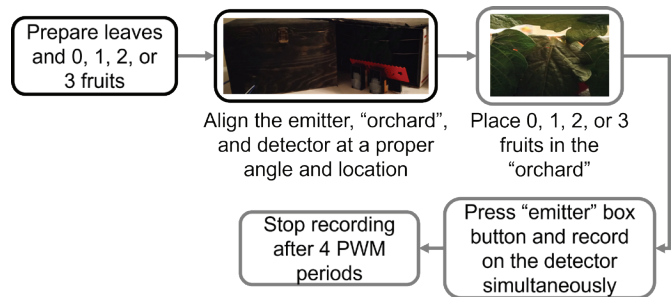


Figure 6: Procedure for collecting experimental data. The output includes four videos, with 0, 1, 2, and 3 fruits in them, respectively. These videos are passed to the image preprocessing procedure as described in Figure 5.

■ Classification and Object Detection Pipelines

In classification, as shown in Figure 7, there will be two classes: no fruit and fruits. For EfficientNet, hyperparameters including the model, image size, batch size, dropout threshold, learning rate, and the seed are carefully tuned; while for YOLOv11, hyperparameters include the model, image size, batch size, cls value, dropout threshold, learning rate, and label smoothing value.

As shown in Figure 7, the classification pipeline starts with setting up the dataset and output directories, and then the GPU is activated for YOLOv11. After that, a specific classification model is retrieved, and two classes are defined. Hyperparameters are tuned, and for EfficientNet, a random seed is picked. The model is then trained based on processed images. If the precision is less than 0.7, the model will be re-trained using new configured hyperparameters. If the precision is equal to or higher than 0.7, the result will be compared with the saved best training result. If the current model is not better than the saved one, hyperparameters will be adjusted, and the classification model will be retrained. Otherwise, the model with the best precision value will be saved.

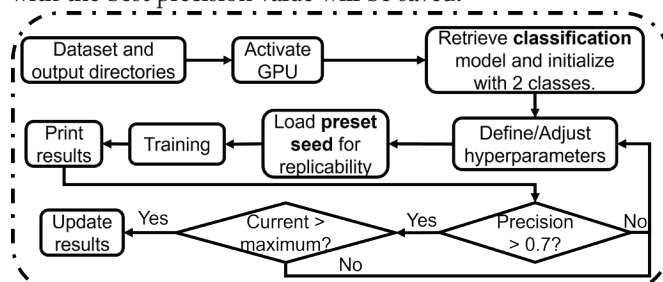


Figure 7: Classification pipeline for determining if there are fruits fully hidden behind leaves. Two AI models are trained: YOLOv11 and EfficientNet.

For object detection models, as shown in Figure 8, the base model is initialized with one class. For YOLOv11 and YOLOv12, tuned hyperparameters include the model (nano, small, medium, and large), image size, batch size, cls value, dropout threshold, learning rate, and label smoothing. For RF-DETR, hyperparameters include batch size, learning rate, backbone learning rate, resolution, and weight decay. Those hyperparameters are discussed in the results sections.

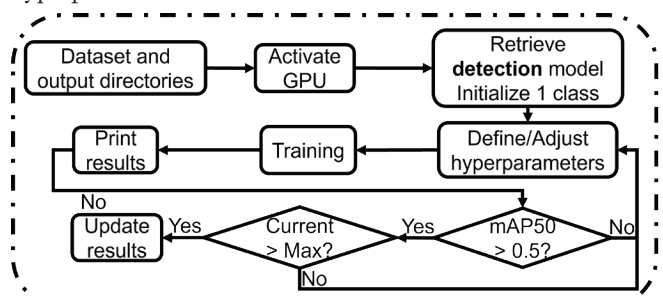


Figure 8: Object detection pipeline used to train a model to decide the number and locations of possible fruits fully hidden behind leaves. Three AI models are trained here: YOLOv11, YOLOv12, and RF-DETR.

It is worth noting that the CPU is used to train the EfficientNet model because of the need to strictly control the

random seed value for the replicability of results. GPU is used to train all four other AI models.

■ Results and Discussion

To answer the three questions, four AI models, PLIA, and SFI are teamed in five different combinations, and their performance is compared. Among them, EfficientNet and YOLOv11 are used in classification, while YOLOv11, YOLOv12, and RF-DETR are compared in terms of object detection performance.

The first two sections will discuss the best results in classification and object detection, respectively. The last section will discuss the ablation tests, demonstrating whether PLIA and SFI improve performance.

All training, validation, and testing are conducted on a laptop computer. Its CPU (for EfficientNet) is a 13th Gen Intel(R) Core™ i7-13700HX (2.10 GHz) with a RAM of 16GB, and the GPU used in YOLOv11, YOLOv12, and RF-DETR is the NVIDIA GeForce RTX 4060. The Python version used is 3.10.19 for EfficientNet and 3.11.14 for YOLOv11, YOLOv12, and RF-DETR.

To achieve optimal classification and/or object detection performance, the AI models were trained across a wide range of hyperparameter settings. For each hyperparameter setting, performance metrics such as F1 score, mAP50, and precision are recorded. Through extensive trials, we observed that model performance is mainly affected by three hyperparameters: learning rate, image size, and batch size. Therefore, based on this observation, a total of 45 hyperparameter settings are tried for two classification and two object detection models. The only exception is the RF-DETR model, which requires a long training time for each hyperparameter setting. Therefore, only a limited number of settings were evaluated for the RF-DETR model. The 45 combinations of the hyperparameter settings are: learning rate (0.00005, 0.0001, 0.0002), image size (224, 288, 320, 480, 640), and batch size (8, 16, 24).

The images in the dataset are split into three groups: training (70%), validation (20%), and testing (10%). All images from the same experiment are assigned to the same dataset partition. The dataset is split at the experiment level, rather than at the image level.

It is worth noting that approximate tomato placement is determined beforehand via ordinary RGB pictures taken of the apparatus, as shown in Figure 2. This information is utilized to determine the ground-truth bounding boxes by noting the approximate locations of each of the tomatoes, the number of tomatoes in that specific experiment, and the size of the tomatoes.

1. Classification results

Two AI models, EfficientNet and YOLOv11, enhanced by PLIA and SFI, are customized for classification tasks to determine whether there are tomatoes fully occluded by leaves in the mock tomato orchard.

The hyperparameters for "EfficientNet+PLIA+SFI" that achieved the best precision are: model (B0), image size (224x224), batch size (8), dropout threshold (0.2), learning

rate (0.00001), and seed value (80). In contrast, the hyperparameters for YOLOv11 that achieved the best precision are: model ("yolov11n-cls"), image size (288), batch size (24), cls value (1.8), dropout threshold (0.1), learning rate (0.0002), and label smoothing value (0.2), respectively.

As shown in Table 1, for both models, the precision is well above 70%, indicating that the models are effective at determining the presence of fully hidden fruits, thereby addressing Question 1. The F1 scores for them are around 0.9, showing a strong balance between precision and recall.

Table 1: Comparison of Classification Results.

Method	Precision	Recall	F1
EfficientNet + pLIA + SFI	0.8636	1.0000	0.9157
YOLOv11 + pLIA + SFI	0.9189	0.8718	0.8947

As an example, the training results of EfficientNet are shown in Figure 9. The plot on the top shows that the training and validation accuracies converge over epochs, while the figure on the bottom shows the results of the confusion matrix of the validation.

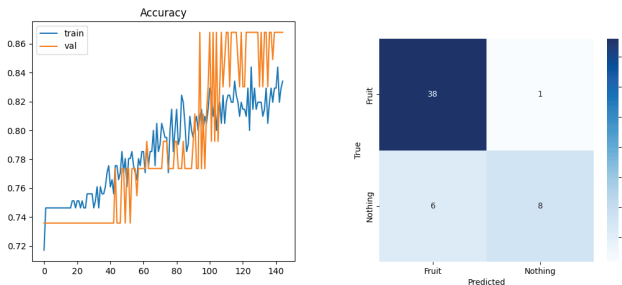


Figure 9: An example training output from EfficientNet. On top is the graph of model accuracy per epoch. At the bottom is the confusion matrix associated with the dataset that the model was trained on.

2. Object detection results:

Three models, YOLOv11, YOLOv12, and RF-DETR, empowered by PLIA and SFI, are trained to detect fully occluded fruits in the mock tomato orchard.

The best results achieved using these three models are shown in Table 2. The hyperparameters used to achieve the best YOLOv11 model are a learning rate of 0.0002, image size of 480, classification loss of 1.8, batch size of 16, dropout threshold of 0.1, and label smoothing of 0.2; whereas YOLOv12 utilizes a learning rate of 0.0001, image size of 480, classification loss of 1.8, batch size of 16, dropout threshold of 0.1, and label smoothing of 0.2. The optimal hyperparameters for RF-DETR are a learning rate of 0.0002, a backbone learning rate of 0.0000001, a batch size of 4, gradient accumulation steps of 4, an image size of 336, a weight decay of 0.0006, and 100 queries.

As shown in Table 2, the mAP50 scores of all models are greater than 65%, which means that the models are effective in determining the number and location of fully hidden fruits, addressing Question 2. For YOLOv11 and YOLOv12 cases, their precision numbers are also higher than 70% in this very challenging task, addressing Question 1.

Table 2: Comparison of Object Detection Results.

Method	Precision	Recall	mAP50
YOLOv11 + pLIA + SFI	0.7421	1.0000	0.6874
YOLOv12 + pLIA + SFI	0.7683	0.9870	0.6583
RF-DETR + pLIA + SFI	N/A	N/A	0.6620

Figure 10 shows several plots generated from the training run of YOLOv12 with one hyperparameter setting. The mAP50 value in part (a) oscillates while gradually increasing, and the model parameters corresponding to the best mAP50 value in these epochs are saved. Parts (b), (c), and (d) show three bounding box examples in which tomato locations are successfully identified. In part (e), there is only one tomato; however, YOLOv12 produced two false positives.

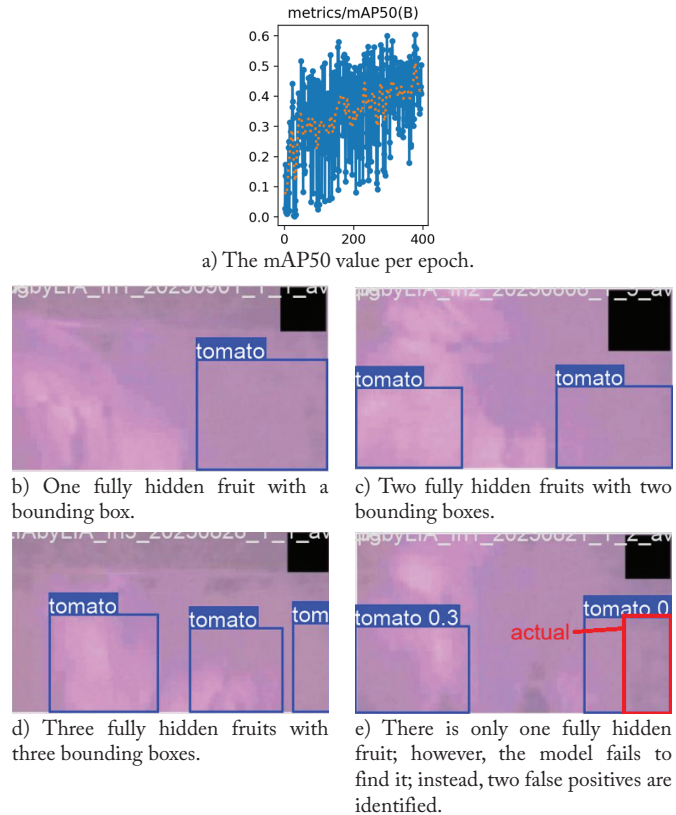


Figure 10: An example of training output from the YOLOv12 detection model.

3. Model size:

One of the hyperparameters being tuned is the image size for each model. As shown in Table 3, the image size needed for classification is much smaller than that of object detection. Furthermore, the sizes of the trained models are in general very small, ranging from 5.23MB to 19.9M, except the trained RF-DETR model, of which the size is 352MB. This reflects the deployability of each model on small devices. For example, a model such as YOLOv11 for object detection, with a size of only 5.23 MB, can be more easily deployed than a model like RF-DETR, which has a size of 352 MB. This is due to hardware constraints, as a model like RF-DETR may be too computationally heavy to run on a basic Raspberry Pi™.

4. Ablation tests:

Four ablation tests are conducted, aiming to illustrate the contributions of PLIA and SFI to the key performance of the two classification and two object detection models. In each ablation test, two hyperparameter settings are shown: the best result reported previously and a selected (non-best) result. Here, "baseline" refers to training an AI model using raw im-

ages without PLIA and SFI. "PLIA" denotes the case where the raw images are processed using the PLIA method, while "PLIA+SFI" refers to the case when both the PLIA and SFI video/image pre-processing methods are applied.

Table 3: Comparison of Model Complexity.

	Classification		Object Detection		
	YOLOv11	EfficientNet	YOLOv11	YOLOv12	RF-DETR
Image Size (pixels)	288	320	480	480	336
Model Size (MB)	3.04	15.8	5.25	5.25	352

It is worth mentioning that "precision" and "mAP50" are the primary performance metrics focused on during the training for classification and object detection, respectively.

The ablation test result for classification using YOLOv11 is listed in Table 4. The first three rows compare the performance improvement achieved using the proposed video/image pre-processing methods with the hyperparameters that yield the best precision, as discussed previously. "PLIA" outperforms the "baseline" across all three metrics, and "PLIA+SFI" further improves precision and F1 score. These indicate that the proposed methods effectively enhance the models' ability to make more accurate predictions. The last three rows in Table 4 show the ablation test for one selected hyperparameter setting, in which learning rate, image size, batch size, drop rate, and label smoothing are set to 0.0002, 640, 8, 0.1, and 0.2, respectively. As shown, the "PLIA+SFI" strategy improves the precision and F1 score, whereas "PLIA" only achieves the same precision and F1 score as "baseline."

The EfficientNet ablation test is shown in Table 5. The first set of three rows represents the best results for the "baseline", "PLIA", and "PLIA+SFI" cases. These show that the model improves the precision value from the "baseline" case to the "PLIA" case, and from the "PLIA" case to the "PLIA+SFI" case. The second set of three rows represents the results achieved with a set list of hyperparameters, which are a learning rate of 0.0002, an image size of 320, and a batch size of 8. This shows that the addition of "PLIA" and "PLIA+SFI" improves both precision and F1 score.

Table 4: Ablation Test 1 – Classification (YOLOv11).

	Precision	% Increase	Recall	% Increase	F1	% Increase
Baseline*	0.8500	N/A	0.8718	N/A	0.8608	N/A
pLIA*	0.8750	2.9%	0.8974	2.9%	0.8861	2.9%
pLIA+SFI*	0.9189	5.0%	0.8718	-2.9%	0.8947	1.0%
Baseline**	0.7358	N/A	1.0000	N/A	0.8478	N/A
pLIA**	0.7358	0.0%	1.0000	0.0%	0.8478	0.0%
pLIA+SFI**	0.7500	1.9%	1.0000	0.0%	0.8571	1.1%

(*: The best result) (**: Another selected result)

The ablation tests for object detection are shown in Tables 6 and 7. First, the performance comparison for the YOLOv11 object detection mode is displayed in Table 6. The best results for "baseline", "PLIA", and "PLIA+SFI" with their respective hyperparameters listed previously are shown in the first three rows. From these, it can be observed that "PLIA" improves from "baseline" and "PLIA+SFI" improves from "PLIA" across all three metrics. Therefore, the proposed methods are effective at enhancing the model's performance. The last three rows in Table 6 were trained with the same hyperparameters to conduct an ablation test. The hyperparameters are learning rate 0.0002, image size 320, classification loss 1.8, batch size 8, dropout 0.1, and label smoothing 0.2, respectively. As shown here, the "PLIA+SFI" strategy yielded an improvement for precision and mAP50, while the method with only "PLIA"

yielded an improvement on mAP50 but a decline in precision from "baseline".

Second, a similar ablation test was conducted for the YOLOv12 model, as shown in Table 7. Rows 1-3 show the best results for the "baseline", "PLIA", and "PLIA+SFI" cases, each with its own distinct set of hyperparameters as shown previously. These results show that each addition of the proposed method improves the mAP50 of the model. Rows 4-6 show the performance of the "baseline", "PLIA", and "PLIA+SFI" methods on the same selected set of hyperparameters, with 0.0002 (learning rate), 480 (image size), 1.8 (classification loss), 24 (batch size), 0.1 (dropout threshold), and 0.2 (label smoothing), respectively. The results unveil that the proposed method enhances the precision, recall, and mAP50.

Table 5: Ablation Test 2 – Classification (EfficientNet).

	Precision	% Increase	Recall	% Increase	F1	% Increase
Baseline*	0.8478	N/A	1.0000	N/A	0.9176	N/A
pLIA*	0.8485	0.1%	1.0000	0.0%	0.9127	-5.3%
pLIA+SFI*	0.8636	1.8%	1.0000	0.0%	0.9157	3.3%
Baseline**	0.8125	N/A	1.0000	N/A	0.8966	N/A
pLIA**	0.8444	3.9%	0.9828	-1.7%	0.9084	1.3%
pLIA+SFI**	0.8636	2.3%	0.9744	-0.9%	0.9157	0.8%

(*: The best result) (**: Another selected result)

Table 6: Ablation Test 3 – Object Detection (YOLOv11).

	Precision	% Increase	Recall	% Increase	F1	% Increase
Baseline*	0.6182	N/A	1.0000	N/A	0.6190	N/A
pLIA*	0.7179	16.1%	1.0000	0.0%	0.6648	7.4%
pLIA+SFI*	0.7421	3.4%	1.0000	0.0%	0.6874	3.4%
Baseline**	0.8167	N/A	1.0000	N/A	0.4807	N/A
pLIA**	0.6761	-14.1%	1.0000	0.0%	0.5358	5.5%
pLIA+SFI**	0.8681	19.2%	1.0000	0.0%	0.6112	7.5%

(*: The best result) (**: Another selected result)

Table 7: Ablation Test 4 – Object Detection (YOLOv12).

	Precision	% Increase	Recall	% Increase	F1	% Increase
Baseline*	0.6987	N/A	1.0000	N/A	0.5910	N/A
pLIA*	0.8384	20.0%	0.9091	-9.1%	0.6224	5.3%
pLIA+SFI*	0.7683	-8.4%	0.9870	8.6%	0.6583	5.8%
Baseline**	0.6279	N/A	1.0000	N/A	0.5460	N/A
pLIA**	0.7299	16.2%	1.0000	0.0%	0.5888	7.8%
pLIA+SFI**	0.8787	20.4%	1.0000	0.0%	0.6046	2.7%

(*: The best result) (**: Another selected result)

In summary, the proposed PLIA and SFI strategies improve the precision and mAP50 values in classification and object detection tasks, respectively.

■ Limitations

- The current mock tomato orchard only has 1 or 2 layers of leaves covering occluded fruits. In real farm conditions, leaf covers can have more than 2 leaves overlapped on top of a fruit. The used 940nm NIR diodes, which only have a 70mW maximum power, may not penetrate thick leaf covers, and the reflected photons may also scatter. Therefore, in real scenarios, a more powerful NIR diode or laser, or even a millimeter radar, can be utilized. However, the image/video pre-processing methods and the AI models should still be effective.

- The computational cost of the LPF used in PLIA is high because it must iterate through every pixel of an image twice. The current PLIA software is implemented in MATLAB, and in certain scenarios, the execution time of the applied LPF can extend to several minutes. A potential solution is to adopt a computationally efficient LPF and mitigate any resulting performance degradation in the AI models by training on a larger image dataset.

- The performance of the current AI models, while exceeding the thresholds, has not reached 0.95 in precision or mAP50. The simplest way to address this issue is to increase the size

of the image dataset. Alternatively, new SFI methods can be developed to improve model precision. An instance of this involves predicting the average RGB values of scenarios without fully hidden fruits, which are treated as baseline numbers. The SFI method is then to compute the difference between the average RGB value and that baseline.

■ Conclusion

In this study, a combination of NIR, image/video pre-processing approaches, and deep AI learning models is proposed to tackle a very challenging problem: detecting and classifying tomatoes that are fully hidden behind leaves. Unlike many other object detection problems, NIR signals reflected from fully occluded fruits are extremely weak and embedded in a large magnitude of noise. To enhance the performance of AI models, a PLIA is developed to extract weak NIR signals from noise and produce an image. After that, the average RGB value of the generated image is used as the side length of a black square, which is injected into the image. This step provides the subsequent AI models with an additional visual cue indicating whether tomatoes are fully hidden behind leaves. Seventy-seven experiment videos obtained in a custom-built, mock tomato orchard are used in training the selected CNN and vision transformer models. All these AI models are able to tell whether there are fully occluded fruits more than 86% of the time. The mAP50 values for object detection are above 0.65. To the best of the authors' knowledge, this approach is low-cost and effective in determining the presence of fully hidden fruits. The ablation tests demonstrate the effectiveness of the proposal PLIA and SFI pre-processing methods in improving the performance of the AI models for classification and object detection.

■ Acknowledgments

The author would like to thank Dr. Shawn Putnam for the introduction of the lock-in amplifier and coupled image analysis concept. The author would also like to thank Dr. Chen Chen for his thorough proofreading and editing of the manuscript.

■ References

1. Zhai, Z., Martínez, J., Beltran, V., Martínez, N., Decision support systems for agriculture 4.0: survey and challenges. *Computers and Electronics in Agriculture*, **2020**, 170, 105256. DOI: 10.1016/j.compag.2020.105256.
2. Chandrakala, P., Badholia, R., Arunkumar, E., Victor, M., Maheswaran, M., Choubey, A., A design of agricultural robotics for the use of sowing and planting. *Proceedings of the 4th International Conference on Advanced Computing, Innovation Technology and Engineering (ICACITE)*, **2024**, 345–349. DOI: 10.1109/ICACITE60783.2024.10616892.
3. Esfandiari, Y., Nagasubramanian, K., Fotouhi, F., Schnable, P., Ganapathysubramanian, B., Sarkar, S., Distributed deep learning for persistent monitoring of agricultural fields. *Neural Information Processing Systems (NeurIPS 2021 AI for Science Workshop)*, Virtual, **2021**. <https://openreview.net/forum?id=rctFXFsFvbl>.
4. Sharma, N. A. P., Dadheech, P., Aneja, S., Predicting agriculture yields based on machine learning using regression and deep

- learning. *IEEE Access*, **2023**, 11, 111255–111264. DOI: 10.1109/ACCESS.2023.3321861.
5. Barburiceanu, B. O. R. L. S., Meza, S., Terebes, R., Convolutional neural networks for texture feature extraction: applications to leaf disease classification in precision agriculture. *IEEE Access*, **2021**, 9, 160085–160103. DOI: 10.1109/ACCESS.2021.3131002.
6. Vougioukas, S., Agricultural robotics, *Annual Review of Control, Robotics, and Autonomous Systems*, **2019**, 2(1), 365–392. DOI: 10.1146/annurev-control-053018-023617.
7. Fontani, M., Luglio, S. M., Gagliardi, L., Peruzzi, A., Frascioni, C., Raffaelli, M., and Fontanelli, M., A systematic review of 59 field robots for agricultural tasks: applications, trends, and future directions, *Agronomy*, **2025**, 15(1), 2185. DOI: 10.3390/agronomy15092185.
8. Houetohossou, S., Houndji, V., Hounmenou, C., Sikirou, R., Kakai, R., Deep learning methods for biotic and abiotic stresses detection in fruits and vegetables: state of the art and perspectives. In *Artificial Intelligence in Agriculture*, **2023**, 9, 46–60. DOI: 10.1016/j.aiaa.2023.08.001.
9. Moldvai, L., Nyéki, A., Innovative computer vision methods for tomato (*Solanum lycopersicum*) detection and cultivation: a review. *Discover Applied Science*, **2025**, 7(975). DOI: 10.1007/s42452-025-07613-x.
10. Cho, J. W., Tin, T. T., Tu, L. H. T., Park, G. S., Chien, V. D., Kim, J. W., A nutrient deficiency prediction method using deep learning on development of tomato fruits. in *Proceedings of IEEE Conference*, **2018**, 338–341. DOI: 10.1109/iFUZZY.2018.8751688.
11. Koirala, A., Walsh, K., Wang, Z., McCarthy, C., Deep learning method overview and review of use for fruit detection and yield estimation. *Computers and Electronics in Agriculture*, **2019**, 162, 219–234. DOI: 10.1016/j.compag.2019.04.017.
12. Xiao, F., Wang, H., Xu, Y., Zhang, R., Fruit detection and recognition based on deep learning for automatic harvesting: an overview and review. *Agronomy*, **2023**, 13(6), 1625. DOI: 10.3390/agronomy13061625.
13. Guo, J., Yang, Y., Lin, X., Memon, M. S., Liu, W., Zhang, M., Sun, E., Revolutionizing agriculture: real-time ripe tomato detection with the enhanced Tomato-YOLOv7 system. *IEEE Access*, **2023**, 11, 133086–133098. DOI: 10.1109/ACCESS.2023.3336562.
14. Duong, L., Nguyen, P., Sipio, C., Ruscio, D., Automated fruit recognition using EfficientNet and MixNet. *Computers and Electronics in Agriculture*, **2020**, 171, 105326. DOI: 10.1016/j.compag.2020.105326.
15. Tan, M., Le, Q. V., EfficientNetV2: smaller models and faster training, arXiv preprint, 2021. <https://arxiv.org/abs/2104.00298>.
16. Lawal, M., Tomato detection based on modified YOLOv3 framework. *Scientific Reports*, **2021**, 11, 1447. DOI: 10.1038/s41598-021-81216-5.
17. Yang, S., Wang, W., Gao, S., Deng, Z., Strawberry ripeness detection based on YOLOv8 algorithm fused with LW-Swin transformer. *Computers and Electronics in Agriculture*, **2023**, 215, 108360. DOI: 10.1016/j.compag.2023.108360.
18. Xiao, B., Nguyen, M., Yan, W., Fruit ripeness identification using transformers. *Applied Intelligence*, **2023**, 53, 22488–22499. DOI: 10.1007/s10489-023-04799-8.
19. Sa, I., Lim, J., Ahn, H., MacDonald, B., DeepNIR: datasets for generating synthetic NIR images and improved fruit detection system using deep learning techniques, *Sensors*, **2022**, 22(13), 4721. DOI: doi.org/10.3390/s22134721.
20. Jocher, G., Qiu, J., Chaurasia, A., *Ultralytics YOLO*, Software, 2023. <https://github.com/ultralytics/ultralytics>.
21. Robinson, I., Robicheaux, P., Popov, M., *RF-DETR*, Software, 2025. <https://github.com/roboflow/rf-detr>

22. PerkinElmer™ Instruments, *Low level optical detection using lock-in amplifier techniques*
23. Xu, A., An integrated lock-in amplifier and near infrared method for finding fruits fully behind leaves in a homemade testbed, *International Journal of High School Research*, **2026**, 6(2), 1–6. DOI: 10.36838/v8i2.1.
24. Ghiasi, G., Cui, Y., Srinivas, A., Qian, R., Lin, T., Cubuk, E., Le, Q. V., Zoph, B., Simple copy-paste is a strong data augmentation method for instance segmentation, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, **2021**, 2917–2927. https://openaccess.thecvf.com/content/CVPR2021/html/Ghiasi_Simple_Copy-Paste_Is_a_Strong_Data_Augmentation_Method_for_Instance_CVPR_2021_paper.html?ref=https://githubhelp.com
25. Gené-Mola, J., Gregorio, E., Cheein, F. A., Guevara, J., Llorens, J., Sanz-Cortella, R., Escolà, A., Rosell-Polo, J., Fruit detection, yield prediction and canopy geometric characterization using LiDAR with forced air flow, *Computers and Electronics in Agriculture*, **2020**, 168, 105121. DOI: doi.org/10.1016/j.compag.2019.105121.
26. Couture, A., Adams, C., Fonseca, R., Schubert, J., Mastronardi, R., Backscatter imaging for precision agriculture, *US Patent US10712293B2*, **2020**.
27. Mahmud, M., Zahid, A., He, L., Choi, D., Krawczyk, G., Zhu, H., LiDAR-sensed tree canopy correction in uneven terrain conditions using a sensor fusion approach for precision sprayers. *Computers and Electronics in Agriculture*, **2021**, 191, 106565. DOI: 10.1016/j.compag.2021.106565.
28. Shiraz, Z., Khan, U., Shahzad, M., FruitSight: a millimeter wave radar based approach to detect occluded fruits. *Proceedings of IEEE Conference*, **2024**, 270–278. DOI: 10.1109/MASS62177.2024.00044
29. Kim, S., Kim, D., Fruit quantity measurement system and method. *KR Patent 12-541969*, **2020**.
30. Parr, B., Legg, M., Bradley, S., Alam, F., Occluded grape cluster detection and vine canopy visualization using an ultrasonic phased array, *Sensors*, **2021**, 21(6), 2182. DOI: 10.3390/s21062182
31. Weisstein, E. W., Fourier series square wave, *Wolfram MathWorld*. <https://mathworld.wolfram.com/FourierSeriesSquareWave.html> (accessed April 13, 2026).
32. Basel problem, *Wikipedia*. https://en.wikipedia.org/wiki/Basel_problem (accessed April 13, 2026).

■ Author

Aiden Xu is a high school student at Winter Springs High School, Winter Springs, FL. He is interested in conducting different types of research during his spare time and has been participating in the Science and Engineering Fair since the 6th grade.