

Fruit Fly Detection and Classification in IoT Setup

Syed M. Fasih, Asad Ali, Talha Mabood(✉), Atif Ullah, Muhammad Hanif, and Waqar Ahmad

Faculty of Computer Science and Engineering, Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi, Swabi, Pakistan
talhamabood400@gmail.com

Abstract. In the 21st century, advancements in hardware and processing speeds have enabled the deployment of machine learning algorithms such as YOLOv5 on Internet of Things (IoT) devices. These AI-based IoT devices are powerful computers suitable for embedded applications, AI IoT, and edge computing, delivering the performance of a modern AI module at a reduced cost. This conference paper presents a modified version of YOLOv5 called YOLOv5-FlyEye, optimized for embedded systems such as the Nvidia Jetson Nano. The proposed modification achieves high performance in detecting and classifying fruit flies by reducing the computational complexity of the YOLOv5 algorithm. The study aims to demonstrate the potential of AI-based IoT devices in the field of agriculture, particularly in pest detection and monitoring, and contributes to the development of optimized algorithms for efficient and cost-effective embedded systems.

Keywords: Deep Learning · YOLOv5 · Embedded Systems · Edge Computing

1 Introduction

The conventional methods for identifying pests in orchards mostly rely on human experience and visual inspection, which can be highly subjective and inconsistent across different individuals. Moreover, orchards are typically vast in size, which makes it time-consuming and labor-intensive to inspect them solely through human eyes. To address the issues of poor objectivity and low efficiency associated with these traditional methods, there is an urgent need for a new pest identification technology for orchards. In recent years, deep learning-based computer vision has emerged as a powerful tool with various applications in fields such as facial recognition and autonomous driving. While GPU computing power limitations previously hindered the development of this technology, advancements in computing power have gradually overcome this limitation. Consequently, more and more researchers have started to apply computer vision techniques to solve problems related to image recognition and detection in agriculture. Edge computing and embedded systems are transforming the technological landscape for

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2023
O. Gervasi et al. (Eds.): ICCSA 2023 Workshops, LNCS 14108, pp. 593–607, 2023.
https://doi.org/10.1007/978-3-031-37117-2_40

future innovations. Edge computing and embedded systems are developing for hardware specific applications such as machine learning. Embedded machine learning represents a hot topic in both academic research and industry. Emerging innovations such as TinyML is contributing to the field of embedded machine learning to make this world a better place for coming generations. Earlier, machine learning algorithms were not as efficient and computer hardware's were not good enough to support machine learning deployment on embedded scale. But with recent advancements in algorithms such as, Yolov5, MobileNet SSD and TensorFlow lite which are lighter and perform the task with lower number of floating-point operations. Moreover, with improved hardware, we can deploy machine learning algorithms for edge computing. In this paper, we are using Nvidia Jetson Nano development kit for deploying machine learning algorithms to demonstrate the ability of SBC's. Nvidia Jetson nano is equipped with NVIDIA Maxwell GPU, Quad-Core ARM Cortex-A57 processor and 4GB LPDDR4 memory. It uses Nvidia's provided Jetpack-Os which runs a Linux operating system. Figure 1 shows an image of Nvidia Jetson Nano.

2 Literature Review

Various automatic systems have been created to detect and monitor insect pests, with the aim of enhancing integrated pest management in precision agriculture. There are now many automatic detection traps available for significant pests, and these technologies hold great potential for the timely identification and monitoring of dangerous and quarantined pests. Outlined below is a brief overview of several methods employed for the detection of pests, including infrared sensors, audio sensors, and classification based on images.

2.1 Automatic Monitoring of Lepidoptera Pest Species

Worldwide, insects like moths and butterflies can significantly reduce crop yields and have the capacity to lay a large number of eggs, which can result in significant losses when populations are well established [1,2]. Currently, delta traps with pheromone lures are the standard surveillance technique for these insects. However, the development of automatic detection and identification models is quite difficult due to the variety of poses that these insects can adopt when attached to sticky traps [3].

To automatically identify various kinds of moths and butterflies using machine learning techniques, several researchers have created tools and models. The nymphalid butterfly Bicyclus anynana has eyespot patterns that can be automatically recognised by Silveira and Monteiro using a machine learning system with features based on circularity and symmetry. Other butterfly species' patterns could also be recognised by the software with success [4].

Other researchers created a pyramidal stacked de-noising auto-encoder (IpS-DAE) that can produce a deep neural network for moth identification regardless of the pose of insects using a mix of form, colour, texture, and numerical features

retrieved for moth description. This model was able to identify 98.13% of moth genera without categorising the species [3,5].

2.2 Automatic Monitoring of Sucking Insects

Growers in greenhouses are particularly concerned about sucking bugs since they can result in considerable crop losses. Among the pests that cause the most trouble in congested greenhouse environments are thrips, aphids, and whiteflies. Sticky traps are frequently set up in greenhouses to keep an eye on the populations of these pests. However, given the small size and intricate morphology of these pests, in addition to the low effectiveness of weary or inexperienced human observers, this can be a challenging task for non-specialized professionals [6].

To monitor and identify pests, useful integrated pest management (IPM) strategies have been developed in addition to image processing techniques. A model for estimating silver leaf whitefly Bemisia tabaci density was put out by Qiao et al. and is based on an image processing system that makes use of yellow sticky traps from greenhouses. In order to score and identify Bemisia tabaci and the western flower thrips Franklinella occidentalis on sticky traps, Sun et al. applied a unique smart vision method using two-dimensional Fourier transformed spectra, achieving good correlation with human vision accuracy [6].

For the highly accurate identification and monitoring of Franklinella occidentalis and Bemisia tabaci, Espinoza et al. integrated artificial neural networks with image processing (segmentation, morphological and colour property estimates). To analyse both whiteflies and thrips with an accuracy of 85% to 95%, Lu et al. presented a convolutional network (CNN) classifier model in conjunction with a generative adversarial network (GAN) picture augmentation. A model-fitting embedded system for the inexpensive detection of aphids, whiteflies, and thrips was proposed by Xia et al. [6,7].

2.3 Automatic Identification and Monitoring of Fruit Flies

One of the major obstacles to the cultivation of horticulture and fruit in the Mediterranean, tropical, and subtropical regions of the world is the presence of frugivorous fruit flies (Diptera: Tephritidae) [8]. Each year, these pests result in crop losses reaching billions of dollars worldwide, with Brazil alone experiencing losses of USD 242 million [9]. Fruit fly damage can result in crop losses of up to 80% to 100% depending on the crop and the lack of control measures. The fruit fly is a member of the Dacini tribe and includes 932 recognised species, 10% of which are currently considered to be pests of commercial fruit and vegetable production, leading to quarantine problems and trade embargos [8].

A limited number of highly polyphagous and invasive pest species that, if they enter and establish in a place, competitively dominate the native fauna are aggravating the problem of fruit flies worldwide [10–12]. Sprays of pesticides, mass trapping with pheromone lures, and the release of sterile males can all be used to control these pests. The timing of treatment determines the effectiveness of the control approach, which is more effective when the pest is discovered

early. An automatic detection system will be helpful to avoid and monitor the infestation of these quarantine pests in a quicker manner due to the low threshold of control of these pests [12].

3 Machine Learning Algorithms

Machine learning algorithms are a type of artificial intelligence that enable computers to learn from data and make predictions or decisions without being explicitly programmed. These algorithms use statistical models and optimization techniques to identify patterns and relationships in data, and can be used for a wide range of tasks, from image and speech recognition to natural language processing and predictive analytics.

The goal of machine learning is to create models that can make accurate predictions or decisions on new, unseen data. This involves a process of training the algorithm on a large dataset, tuning its parameters, and testing its performance on a separate validation dataset. With the increasing availability of large datasets and powerful computing resources, machine learning algorithms are rapidly advancing and being applied to an ever-growing number of domains and industries. For the detection and classification purpose, YOLO algorithm is used.

3.1 YOLOv5

YOLO (You Only Look Once) is an object detection algorithm that uses deep convolutional neural networks to detect objects in images and videos. YOLOv5 is the version of the algorithm, released in 2020, and is an improvement over previous versions in terms of accuracy and speed.

YOLOv5 is an object detection algorithm that uses a single-stage architecture with a focus on speed and accuracy. It is built on top of the EfficientDet-D7 backbone, which is a high-performance convolutional neural network (CNN) designed for object detection. YOLOv5 also incorporates several key features that improve its performance, such as anchor-based predictions, focal loss, and class-balanced sampling. The YOLOv5 algorithm uses a single neural network to detect objects in an image. The input image is divided into a grid of cells, and each cell is responsible for predicting a set of bounding boxes and object classes. The predicted bounding boxes are then refined and filtered based on their confidence scores and non-maximum suppression is used to eliminate redundant detections.

The YOLOv5 algorithm is based on a deep neural network architecture called CSP (Cross-Stage Partial Network). This architecture consists of a series of convolutional layers that extract features from the input image, followed by a set of detection layers that make predictions based on the extracted features. As it involves CSP (Cross-Stage Partial Network) into Darknet, creating CSPDarknet as its backbone. CSPNet addresses the problem of repeated gradients information in largescale backbones. It also integrates gradient changes into feature maps of the convolved images, thereby decreasing the parameters and number of Floating-Point Operations (FLOPs). A decrease in FLOPs is considered a crucial element in embedded deep learning because a lower value of FLOPs ensures inference speed and accuracy while reducing the model size.

Deep learning models that are smaller, faster, and more efficient, such as YOLOv5 yield great performance on jetson Nano's GPU. The YOLOv5 uses a path aggregation network (PANet) as its neck to boost information flow. PANet involves a feature pyramid network (FPN) structure with an improved bottom-up path, this results in better propagation of low-level features. At the same time, adaptive feature pooling, which links the feature grid and all feature levels, is used to make useful information in each feature level propagate directly to the following subnetwork. PANet improves the utilization of accurate localization signals in lower layers, which can enhance the location accuracy of the object. Further, YOLOv5 also address the problem of multi-scale detection. Such detections involve variations in the size of objects, for example, when detecting crops using a UAV, the footage is subjected to certain scaling, perspective distortion and other factors. There are some variants of YOLOv5. These are as follows:

YOLOv5s. YOLOv5s is a variant of the YOLOv5 algorithm that is designed to be small and fast while still maintaining high accuracy. The "s" in YOLOv5s stands for "small," and it is intended for use cases where computational resources and memory are limited, but real-time object detection is still required.

Compared to the larger YOLOv5 models (YOLOv5m and YOLOv5l), YOLOv5s has fewer layers and parameters, which allows it to be deployed on low-power edge devices with limited resources. Specifically, YOLOv5s has 12 convolutional layers and a total of 7.5 million parameters, which is significantly fewer than the 44 layers and 88 million parameters of YOLOv5m.

One of the key features of YOLOv5s is its use of a CSP (Cross Stage Partial) architecture, which allows for faster and more efficient computation. This architecture is based on the idea of splitting the input data into two streams, which are then processed in parallel by two separate convolutional blocks. The outputs of these blocks are then concatenated and passed through another set of convolutional layers to produce the final predictions. Another important feature of YOLOv5s is its use of a dynamic anchor assignment strategy. This strategy involves adjusting the anchor boxes during training based on the statistics of the training data, which helps to improve the model's accuracy on objects of different sizes and aspect ratios.

YOLOv5n. Edge computing has become an increasingly important trend in computer vision, as more and more applications require real-time object detection on low-power devices. YOLOv5n is an optimized version of the popular YOLOv5 object detection model, designed specifically for edge devices with limited computing resources. In order to achieve high performance and efficiency on these devices, several technical considerations must be taken into account when optimizing the model architecture and training process.

– Precision and accuracy: One of the key trade-offs when optimizing YOLOv5 for edge devices is achieving a balance between precision and accuracy. Since YOLOv5n is designed for low-power devices, it may use lower precision arithmetic to reduce the computational and memory requirements of the model.

However, lower precision can also reduce the accuracy of the model, so it is important to carefully evaluate the impact of different precision levels on the model's performance.
- Compression techniques: In addition to quantization and pruning, there are several other compression techniques that can be used to reduce the size of the YOLOv5n model. For example, weight sharing, which involves using the same weights for multiple layers of the network, can significantly reduce the number of parameters in the model without sacrificing performance.
- Data augmentation: Data augmentation is a technique used to artificially increase the size of the training dataset by applying random transformations to the input images. This can help to improve the robustness of the YOLOv5n model and reduce the risk of overfitting, which can be a concern when working with limited training data.
- Quantization-aware training: Quantization-aware training is a technique that involves training the YOLOv5n model with quantization in mind, rather than quantizing the model after training. By taking into account the impact of quantization on the model during training, it is possible to achieve better performance and accuracy on low-power devices.

4 Methodology

4.1 Deep Learning Approach

Introducing an efficient approach to address the problem of low inference speed on IoT devices (e.g. NVIDIA Jetson Nano) running YOLOv5 while maintaining sufficient accuracy. We propose modifications to an existing YOLOv5 architecture to achieve superior results while reducing the floating point operations (FLOPs) significantly.

In this paper we discarded the use of standard convolution operation and used Ghosted convolution. Ghosted convolution is a technique used in convolutional neural networks (CNNs) that was introduced in the GhostNet architecture. In traditional CNNs, convolutions are performed on the entire input tensor, resulting in a large number of parameters and computations. Ghosted convolution, on the other hand, are a type of convolution that involves splitting the input feature maps into two parts, the primary feature map and the ghost feature map. The ghost feature map is generated by convolving the primary feature map with a set of lower-dimensional convolution filters. The output of this convolution is then upsampled and added back to the primary feature map to produce the final output [13].

The use of ghosted convolutions in GhostNet allows for more efficient feature extraction while reducing the computational complexity and memory requirements of the network. This is because the ghost feature maps are generated using lower-dimensional convolution filters, which reduces the number of parameters needed to learn and the computational cost of the convolution operation.

Overall, the use of intrinsic feature maps and ghosted convolutions in GhostNet allows for a more efficient and effective deep neural network architecture.

The key idea behind GhostNet is the use of a Ghost module, which is a modified version of a standard convolutional layer. The Ghost module splits the input into two parts, with one part being passed through a cheap operation (e.g. depth-wise convolution), while the other part is passed through a regular convolutional layer. The cheap operation is then used to refine the output of the regular convolution, resulting in a more accurate output while using fewer computational resources.

The Ghost module is defined as follows, where $F(x)$ represents a regular convolution, $G(x)$ represents a cheap operation, and W represents the learnable parameters of the layer:

$$H(x) = F(x) + GW(x)$$

To illustrate this with an example, let's consider an image classification task. Suppose we have an input image x and we want to apply the Ghost module to extract features from it. Regular Convolution (F(x)), we first pass the input image x through a regular convolution operation, which applies a set of learnable filters to extract spatial features. This operation captures complex patterns and details in the image. Cheaper Operation (G(x)), alongside the regular convolution, we perform a cheaper operation that requires fewer computational resources. This operation might involve simpler operations like downsampling, pooling, or applying a small number of filters. The purpose of this operation is to capture more general features or global information from the image. Combining the Results, after obtaining the outputs from both the regular convolution and the cheaper operation, we apply the Ghost module. We multiply the output of the cheaper operation, GW(x), by the learnable parameters W. This multiplication scales the cheaper operation's output based on the learned weights. Finally, we add this scaled output to the output of the regular convolution, F(x), resulting in the final output H(x) of the Ghost module. By combining both the expensive and cheaper operations, the Ghost module aims to strike a balance between computational efficiency and capturing detailed and global features. The learnable parameters W allow the module to adaptively control the influence of the cheaper operation, enabling it to learn the most effective combination of features for a given task.

In the GhostNet architecture, ghosted convolution can be used in conjunction with other techniques such as squeeze-and-excitation (SE) blocks and mobile inverted bottleneck (MBConv) blocks to create a lightweight and efficient CNN. The resulting network has fewer parameters and computations compared to traditional CNNs, while still achieving high accuracy on a variety of computer vision tasks.

In summary, ghosted convolution is a technique that can be used in CNNs to reduce the number of parameters and computations while maintaining accuracy. It achieves this by splitting the input tensor into smaller tensors, which are then convolved separately. The GhostNet architecture uses ghosted convolution in combination with other techniques to create a lightweight and efficient CNN [13,14].

Proposed YOLOv5 Architecture. The customized YOLOv5 architecture namely Yolov5-FlyEye has two main parts, the backbone and the head. The backbone consists of a series of convolutional layers that are used to extract features from the input image. The backbone starts with a 6-layer Convolutional Neural Network (CNN) with 32 output channels, a kernel size of 6, a stride of 2, and padding of 2. This layer is followed by a Ghost convolutional layer with 64 output channels, a kernel size of 3, and a stride of 2, which downsamples the feature maps by a factor of 2.

The backbone then consists of a series of C3Ghost modules arranged in specific manner to form the Yolov5-FlyEye.

In a YOLOv5 model, the "head" refers to the top part of the architecture that takes in features from the backbone and outputs predictions for object detection. The head of the YOLOv5 model is responsible for predicting the bounding boxes and class probabilities for the objects in the input image.

The head consists of a series of layers, including GhostConv (a type of convolutional layer that uses less computation), Upsample (to increase the size of the feature maps), Concat (to concatenate feature maps from different layers), and C3Ghost (a type of GhostConv layer with three parallel branches).

At the prediction level we have removed the P5/32-large and P6/64-xlarge layer from the architecture because out dataset includes the detection of objects that may vary between small and medium sizes. Therefore we used the P3/8-small and P4/16-medium layers at the prediction level.

In the YOLOv5 head, the small prediction level is represented by feature maps from the P3 layer of the backbone network that have been downsampled by a factor of 8. These feature maps have a resolution of 80×80 pixels and are processed by three convolutional layers with ghost convolution filters to generate object detection predictions.

The medium prediction level is represented by feature maps from the P4 layer of the backbone network that have been downsampled by a factor of 16. These feature maps have a resolution of 40×40 pixels and are processed by three convolutional layers with ghost convolution filters to generate object detection predictions.

The object detection predictions for the small and medium prediction levels are then combined using a "Detect" operation that applies anchor-based object detection to the concatenated feature maps. These predictions are used to generate the final bounding box and class predictions for the YOLOv5 model.

Overall, the head architecture is designed to combine information from multiple resolution levels and perform detection on objects of varying sizes, while minimizing computational cost.

5 Dataset

In any model, dataset plays vital role in performance irrespective of algorithm. In our case we are having two species of fruit flies. On which our data is trained. If any other specie of fly is detected then it would be classified as foreign fly. This

custom dataset has been collected from Bactrocera zonata and Bactrocera dorsalis. Bactrocera zonata, also known as the peach fruit fly, is known to attack a variety of fruits, including peaches, plums, pomegranates, and citrus fruits. It has also been reported to attack some vegetables, such as eggplant and tomato. Bactrocera dorsalis, also known as the Oriental fruit fly, is known to infest over 250 different types of fruit, including citrus, mangoes, bananas, guava, and papaya. It can also attack some vegetables, such as peppers and tomatoes. The dataset was taken from [15].

We also used a dataset from [16], which consists of 200 images of yellow fruit flies in the trap are consolidated into two parts, namely training dataset and test dataset, with the proportion of 75% and 25%, respectively.

Overall, both Bactrocera zonata and Bactrocera dorsalis are highly destructive pests that can cause significant damage to fruit and vegetable crops. Effective control measures, such as the use of insecticides, biological control agents, and quarantine measures, are essential for preventing the spread of these pests and minimizing their impact on agricultural production.

In order to classify rest of the flies from fruit flies, we also collected data of butterflies from [17]. That will be considered as foreign fly. We aquired all the above mentioned dataset from three sources and merged them to create a single dataset.

After the dataset is created, we implemented the algorithm. The dataset details are given below having testing set and validation set as follows (Table 1).

Table 1. Table showing dataset split ratio

Data split	Image Count	Image Percentage
Training Set	2500	84%
Validation Set	399	13%
Testing Set	85	3%

It is necessary to preprocess the data. The goal of preprocessing is to enhance the quality of the data and to optimize its use for a specific machine learning algorithm. By applying preprocessing techniques, machine learning models can be trained more accurately and efficiently, leading to better predictions and insights.

Preprocessing techniques may include removing irrelevant or duplicated data, handling missing values, scaling or normalizing data, reducing noise, and transforming data into a different format or representation (Table 2).

After preprocessing is done, augmentation is performed on the dataset. In augmentation, blur of 0.75px and three outputs per training example are produced (Table 3).

Table 2. Table with augmentation and detail columns.

Type of Preprocessing	Detail
Auto-Orient	Applied
Resize	Stretch to 640×640

Table 3. Table with augmentation and detail.

Augmentation	Detail
Blur	Applied up to 0.75px

Annotation group is also a vital component in any models. So the table below contain the annotation group as well as details about the dataset (Table 4).

Table 4. Table showing dataset version detail.

Attribute	Detail
Version ID	3
Annotation Group	Flies

The dataset was labeled using Roboflow, which is an online platform for managing, annotating, and augmenting computer vision datasets. The following image shows a collage of the labeled dataset.

6 Results

To ensure how well our customized version of Yolov5 namely Yolov5-FlyEye performs against the two Yolov5n and Yolov5s variants. We found out that Yolov5-FlyEye outperforms both Yolov5s and Yolov5n by slight margin when subjected to same training hyperparameter values. The following shows the comparison of mean average precision at iou 0.5–0.95 against the Floating point operations (GFLOPs) (Fig. 2).

The Fig. 3 shows how the Yolov5-FlyEye performs against the Yolov5n and Yolov5s for precision vs epoch metric.

The hyper-parameters for the training of the three yolov5 variants were the same. The Table 5 highlights the hyper-parameters used for the training.

Since we are deploying the Yolov5-FlyEye on embedded device namely Nvidia Jetson Nano, we aim to lower the computational complexity of the model. We aimed to lower the floating point operations (FLOPs) which determine the computational complexity of the model while achieving superior performance than other variants used for comparison. The Table 6 shows the GFLOPs, the GPU used for training and the sizes of the weight files produced from the three Yolov5



Fig. 1. Labeled Dataset

Table 5. Hyper-parameters

Hyper-parameter	YOLOv5 FlyEye	YOLOv5n	YOLOv5s
Input resolution	640×640	640×640	640×640
Number of layers	271	157	157
Training batch size	16	16	16
Epochs	177	191	192
Patience	30	30	30
Optimizer	Adam	Adam	Adam
Learning rate	0.001	0.001	0.001
Momentum	0.937	0.937	0.937
Weight decay	0.0005	0.0005	0.0005
IoU threshold	0.55	0.55	0.55

variants. The comparison shows that our proposed Yolov5 architecture namely Yolov5-FlyEye outperforms the Yolov5n and Yolov5s. The following table shows the comparison. If we look at the Table 6, we can see that our proposed Yolov5 architecture has 23% lower computational complexity with respect to Yolov5n and 78% lower computational complexity than the Yolov5s.

After training we performed inference to check how well object detection works on the test images. The Fig. 4 shows the detection results.

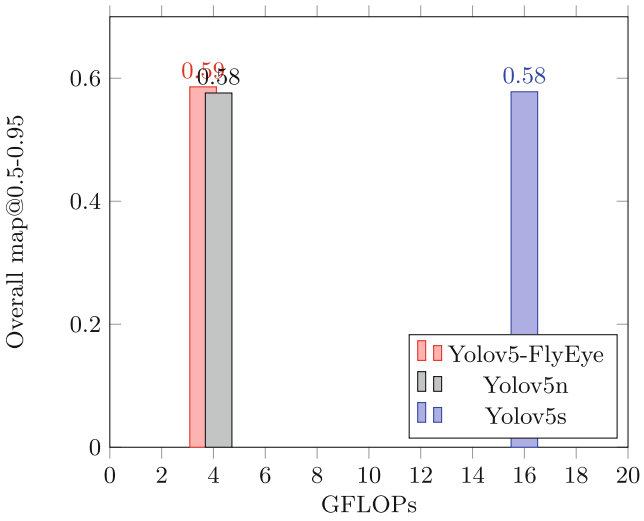

Fig. 2. Bar chart showing the performance of Yolov5-FlyEye, Yolov5n, and Yolov5s models on a detection task as a function of GFLOPs.

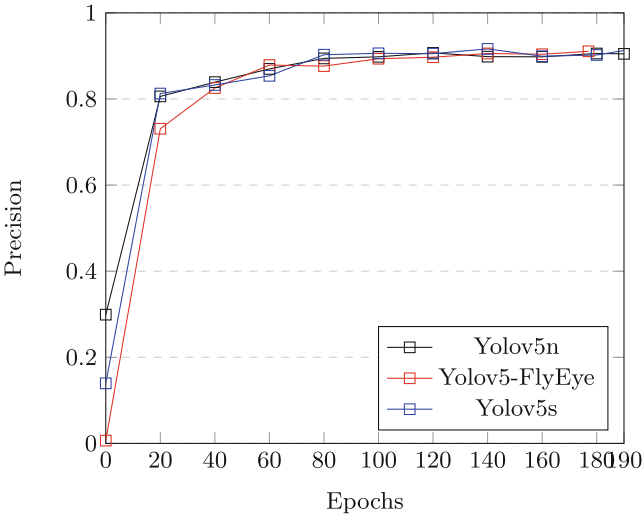

Fig. 3. Graph showing the precision of Yolov5-FlyEye, Yolov5n, and Yolov5s models on a detection task as function of epochs

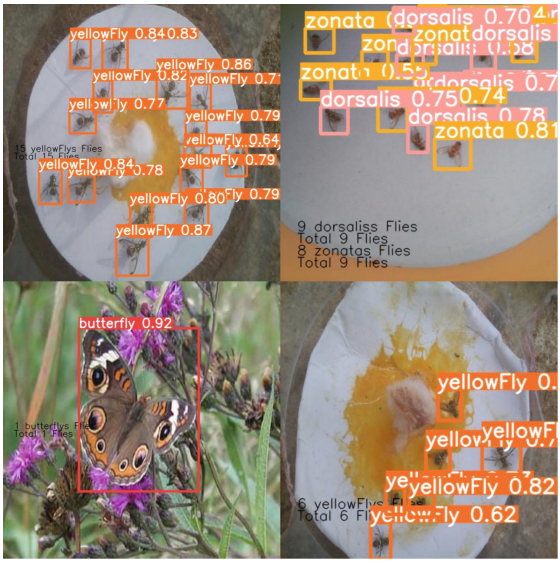

Fig. 4. Inference on test images

Table 6. Yolov5 Training Details

Model	GPU	Weight file size	GFLOPs
Yolov5-FlyEye	NVIDIA RTX A2000-12GB	3.2 MB	3.5
Yolov5n	NVIDIA RTX A2000-12GB	4.2 MB	4.6
Yolov5s	NVIDIA RTX A2000-12GB	14.7 MB	15.8

7 Conclusion

In conclusion, this conference paper highlights the potential of AI-based Internet of Things (IoT) devices in the field of agriculture by presenting the modified version of YOLOv5 called YOLOv5-FlyEye. The results demonstrate that the proposed modification achieves high performance in detecting and classifying fruit flies on embedded systems such as the Nvidia Jetson Nano. The optimized algorithm reduces the computational complexity of YOLOv5, making it suitable for efficient and cost-effective embedded systems. These findings contribute to the development of AI-based IoT devices for pest detection and monitoring in agriculture, showcasing the power of modern AI modules in edge computing applications. The advancements in hardware and processing speeds have enabled the deployment of machine learning algorithms such as YOLOv5 on IoT devices,

paving the way for a new era of embedded AI computing. Moving forward, future work can explore the application of the YOLOv5-FlyEye algorithm on a wider range of embedded systems and devices to evaluate its performance in diverse environments. Further optimization of the algorithm can be carried out to improve its accuracy and speed, making it even more efficient for real-time pest detection and monitoring. There are a number of things that should be done better for IoT's future work. Real-time monitoring could be one potential avenue for advancement. The Flytrap application now depends on data kept on the Think talk server, which may cause delays in getting crucial information. By sending data from the Jetson board to the mobile application directly, the system can be improved to enable real-time monitoring. Users would be able to get alerts and notifications in real-time as a result, getting more timely and useful information. The creation of a centralised board that can link to numerous cameras dispersed throughout an orchard or farming area represents another potential future avenue for IoT in the context of Flytrap. This would make it possible to develop a monitoring system that is more thorough, can gather information from many sources, and can give a more thorough account of pest activity in various locations.

Acknowledgment. We would like to express my sincere gratitude to my supervisor Dr Muhammad Hanif and co-supervisor Dr Waqar Ahmed for their invaluable support and guidance throughout the process of writing this conference paper. Their expertise and insights were instrumental in shaping the direction of our research and in refining our ideas. Their feedback and constructive criticism were immensely helpful in improving the quality of our manuscript. We are deeply grateful for the time and effort they have dedicated to mentoring us and helping us grow. Without their guidance, this paper would not have been possible.

References

1. Bradshaw, C.J., et al.: Massive yet grossly underestimated global costs of invasive insects. Nat. Commun. **7**, 12986 (2016)
2. Gautam, M.P., Singh, H., Kumar, S., Kumar, V., Singh, G., Singh, S.N.: Diamondback moth, Plutella xylostella (Linnaeus) (Insecta: Lepidoptera: Plutellidae) a major insect of cabbage in India: a review. J. Entomol. Zool. Stud. **6**, 1394–1399 (2018)
3. Wen, C., Wu, D., Hu, H., Pan, W.: Pose estimation-dependent identification method for field moth images using deep learning architecture. Biosyst. Eng. **136**, 117–128 (2015)
4. Silveira, M., Monteiro, A.: Automatic recognition and measurement of butterfly eyespot patterns. Biosystems **95**, 130–136 (2009)
5. Guarnieri, A., Maini, S., Molari, G., Rondelli, V.: Automatic trap for moth detection in integrated pest management. Bull. Insectol. **64**, 247–251 (2011)
6. Wang, J., Lin, C., Ji, L., Liang, A.: A new automatic identification system of insect images at the order level. Knowl. Based Syst. **33**, 102–110 (2012)
7. Thenmozhi, K., Reddy, U.S.: Crop pest classification based on deep convolutional neural network and transfer learning. Comput. Electron. Agric. **164**, 104906 (2019)

8. Hendrichs, J., Vera, M.T., De Meyer, M., Clarke, A.R.: Resolving cryptic species complexes of major tephritid pests. Zookeys **540**, 5–39 (2015)
9. Oliveira, C.M., Auad, A.M., Mendes, S.M., Frizzas, M.R.: Economic impact of insect pests in Brazilian agriculture. J. Appl. Entomol. **137**, 1–15 (2012)
10. Duyck, P., David, P., Quilici, S.: A review of relationships between interspecific competition and invasions of fruit flies (Diptera: Tephritidae). Ecol. Enthomol. **29**, 511–520 (2004)
11. Duyck, P., David, P., Junod, G., Brunel, C., Dupont, R., Quilici, S.: Importance of competition mechanisms in successive invasions by polyphagous tephritis in La Reunion. Ecology **87**, 1770–1780 (2006)
12. Potamitis, I., Rigakis, I., Tatlas, N.: Automated surveillance of fruit flies. Sensors **17**, 110 (2017)
13. Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., Xu, C.: GhostNet: more features from cheap operations, pp. 1577–1586 (2020)
14. Zhang, Y., Cai, W., Fan, S., Song, R., Jin, J.: Object detection based on YOLOv5 and GhostNet for orchard pests. Information **13**, 548 (2022)
15. Tariq, S., Hakim, A., Siddiqi, A.A., Owais, M.: An image dataset of fruit fly species. Mendeley Data **V1** (2022). https://doi.org/10.17632/hgz2n5jxhp.1
16. Le, A.D., Pham, D.A., Pham, D.T., Vo, H.B.: AlertTrap: a study on object detection in remote insect trap monitoring system using on-the-edge deep learning platform (2022)
17. Wang, J., Markert, K., Everingham, M.: Learning models for object recognition from natural language descriptions. In: Proceedings of the 20th British Machine Vision Conference (BMVC2009), September 2009