

Sabahat et al. LGURJCSIT 2025

ISSN: 2521-0122 (Online)

ISSN: 2519-7991 (Print)

LGU Research Journal of Computer Science & IT

doi: 10.54692/lgurjcsit.2025.092696

Vol (9): Issue (2), July December 2025

AI-POWERED DETECTION OF FRUIT FLIES IN AGRICULTURAL ENVIRONMENTS

Sabahat Anwar[1,*], Ms. Ayesha Akram[2], Nimra Sajjad[3], Aliza Mubayyaz Riaz[4], Mr. Ahsan Rehman Gill[5]

[1,3] Department of Computer Science, University of Agriculture, Faisalabad, Pakistan

[2] Lecturer of Computer Science. University of Agriculture, Faisalabad, Pakistan

[4]Department of Mathematics & Statistics, University of Agriculture, Faisalabad, Pakistan

[5]Visiting Lecturer of Computer Science. Constituent College Toba Tek Singh University of Agriculture, Faisalabad, Pakistan

[1,*]sabahatanwar1122@gmail.com,[2]ayeshaakram443@gmail.com,

[3]nimra325gb@gmail.com,[4]alizamubayyaz@gmail.com,[5]ahsanrehman41@gmail.com

Corresponding Author* Sabahat Anwar

Abstract

Protecting crops from harmful insects is very important for increasing food production and supporting sustainable agriculture. In recent years, the use of smart devices and sensors has made it easier to collect large amounts of data from farms. However, this data must be processed, understood correctly to help farmers make better decisions. This study presents a deep learning-based method for detecting harmful fruit flies that affect fruit quality and crop production. A new dataset with four classes of fruit flies was created by combining several binary and multi-class datasets to improve training and testing. Four different detection models were trained, tested to find out which one performs best for detecting one, two, three, and four types of fruit flies. Several image pre-processing methods, such as YMC norm, homomorphic filtering, min-max normalization, and negative training were applied before model training. These techniques were used to improve the accuracy, reliability of the models. Among all the models, YOLOv5 showed the best results in terms of precision, recall, F1-score, and mean average precision (mAP). An ESP32-CAM-based electronic trap was also added to the system. It captured real-time images and sent instant alerts through WhatsApp and email, allowing quick action against pests. This system is low-cost, easy to use, and suitable for farmers in developing countries like Pakistan. In conclusion, this approach helps reduce the overuse of pesticides and supports the production of high-quality fruits. It also offers a strong base for creating smart and automatic pest monitoring systems for future farming needs.

Keywords: Yellow Mean Color Normalization (YMC-Norm), You Only Look Once (YOLO), Dacus Image Recognition Toolkit (DIRT), Open source Computer Vision Library (OPEN-CV), Intersection Over Union (IOU).

1. INTRODUCTION

Modern agriculture is encountering growing challenges in building a sustainable future to meet the global demand for food [1]. As a key driver of economic growth, agriculture must enhance both the quality and quantity of crop production. However, various insect pests such as fruit flies, bugs, and aphids negatively impact crop metabolism, leading to reduced crop health, lower yield, and diminished quality [2]. These pest-related issues also hinder the overall progress of the agricultural sector [3]. Traditionally, farmers apply pesticides at fixed intervals rather than based on the actual presence or likelihood of pests in the field, which can be inefficient and environmentally harmful [4]. While the use of pesticides can help improve crop yield, it also poses a significant threat to the ecological environment [5]. One of the major reasons for

pesticide overuse in agriculture is the limited knowledge about the specific types of pests present in the fields. Additionally, there is often a lack of timely and accurate data regarding pest population densities. For instance, fruit flies are not uniformly distributed across crop fields. Applying excessive pesticides in areas with low fruit fly density not only harms plant health but also poses potential health risks to consumers. However, if precise and timely pest data were available, early intervention strategies could be adopted enabling targeted pest control, the implementation of effective prevention methods, and a more rational use of pesticides [6]. Traditionally, the identification and estimation of pest species and their population densities have relied on human visual assessment. Workers typically examine characteristics such as texture, shape, and colour of pests, comparing them with expert records [7].

However, these records are often incomplete or inaccurate. Moreover, certain types of pest damage especially those caused by fruit flies can be difficult to describe verbally. Manual methods of pest identification and counting are not only time-consuming and labor-intensive but also prone to errors [8]. This highlights the urgent need for fully autonomous, fast, and accurate vision-based pest recognition systems. Fortunately, with the advancement of digital imaging technology, computer vision and deep learning have emerged as effective solutions. These technologies enable high-quality images of insects to be captured using cameras installed in electronic traps. Machine vision techniques are then used to detect fruit flies in the captured images, and the farm owner is promptly alerted when a harmful species is identified.

Despite significant technological advancements in remote crop monitoring, agriculture continues to face emerging challenges. Among the most serious threats to horticultural crops are fruit flies, which cause direct damage by puncturing fruit skins to lay their eggs. This leads to fruit decay, rendering them unfit for human consumption. Typically, the affected fruits fall to the ground during or just before the maggots complete their development, resulting in substantial yield losses [9].

Since almost all fruit fly species are classified as quarantine pests, importing nations' quarantine requirements to prevent their entry also result in indirect losses from fruit flies in export-quality goods. To enable early preventive measures, electronic traps equipped with cameras and liquid attractants have been widely deployed in fruit farms for fruit fly identification. Additionally, machine learning and computer vision techniques have been integrated into the detection process to enhance the system's efficiency. However, many of the proposed methods have limitations they are often manual or only semi-automated. Fully autonomous systems have been implemented mostly in controlled environments like greenhouses, and even then, these models tend to perform poorly in real-world field conditions.

The following is a description of this study's **primary contributions**:

A novel electronic trap has been proposed, incorporating an ESP32-CAM module, a yellow sticky pad, and a liquid attractant (methyl eugenol) to capture images of Olive fruit flies and transmit them to a server for further analysis.

A multi-threaded web server has been developed using socket programming in Python to receive image data from multiple electronic traps simultaneously.

The YOLOv5m model has been trained on the DIRT (Dacus Image Recognition Toolkit) dataset with hyper-parameter optimization to detect Olive fruit flies appear in the photographs received by the server. Additionally, transfer learning has been employed to initialize the model weights effectively. This approach not only improves performance but also supports the collection of a custom dataset, benefiting from the pre-learned features.

Lastly, an automated alert system has been implemented to notify the respective farm owner via WhatsApp and email when a

harmful fly is detected in the image sent from the trap installed on their farm.

1.1 Objectives

- To develop a cost-effective and user-friendly system for early detection and classification of fruit flies in fruit farms.
- To enable remote monitoring of agricultural fields using an intelligent electronic trap integrated with an ESP32-CAM module.
- To capture real-time images of fruit fly activity and transmit them to a backend server for analysis.
- To notify farmers immediately via WhatsApp and email alerts when fruit flies are detected, ensuring timely interventions.
- To create and annotate a custom dataset for model training, addressing the issue of insufficient training data.

2. BACKGROUND AND RELATED WORK

Many scientific studies were collected and filtered from major databases, including IEEE Xplore, Springer Link, ACM, Science Direct, Emerald, and Google Scholar. Studies applying review-based predictive analysis are summarized in Table 1.

Table 1: Review of Context-Oriented, Review-Based Studies

Sr. No.	Title	Citation	Year	Methodology	Limitations	Conclusion
1	Harnessing AI for Agriculture: Plant Pest Detection on Web Application Using Deep Learning	Nuankaew et al.	2025	Mobile app (Flask, Python, SQLite); images via phone; tested YOLOv5 models.	Real-time image issues; complex field data.	YOLOv5m6 best (Recall 0.8281); shows AI potential.
2	Exploring the Role of AI in Autonomous Drone Technology for Precision Pest Control in Agriculture	Verma	2025	AI + drones; ML, vision, navigation; real-time detection & spraying.	Need datasets; high cost; regulations.	Sustainable pest control; drones can revolutionize farming.
3	AI-powered crop pest detection: a visual approach	Godavarthi M et al.	2025	Leaf images via cameras; grayscale; MATLAB; SVM classifier.	Needs good images; limited scalability; specific pests only.	Early pest detection; targeted control; supports sustainability.
4	AI-Driven Crop Monitoring and Management for Sustainable Agriculture	Ghosh P et al.	2025	AI for yield + pest; acoustic detection; auto spraying; drones & robotics.	Limited data; low transparency; scalability issues.	AI boosts precision farming, food security; needs improvements.

Table 2: Review of Context-Oriented, Review-Based Studies

Sr. No.	Title	Citation	Year	Methodology	Limitations	Conclusion

5	Spot-out fruit fly algorithm with simulated annealing optimized SVM for detecting tomato plant diseases	Gangadevi et al.	2024	Hybrid FOA+SA tuned SVM; vs DT, RF, KNN, K-means.	Limited crop; complex model.	91.1% **acc.**, best vs others.
6	An annotated image dataset of pests on different coloured sticky traps acquired with different imaging devices	Ong & Høye	2024	Img dataset: DSLR, webcam, phone; 4 trap colors; 2 pests.	Only 2 species; narrow scope.	Useful for pest ID automation.
7	IoT based intelligent pest management system for precision agriculture	Ahmed et al.	2024	IoT trap + >1000 imgs; features + CNN; vs ML/DL.	Guava only; small data.	CNN best; AI+IoT effective.
8	Investigation to answer three key questions concerning plant pest identification and development of a practical identification framework	Wayama et al.	2024	334k imgs; 20 pests, 4 crops; ROI + CNN.	Needs diverse fields; cross-crop tough.	**91% acc., F1=88.5**, robust.
9	Accurate detection and identification of insects from camera trap images with deep learning	Bjerge Id et al.	2023	2M imgs, 29,960 labeled; 9 taxa; YOLO.	Tiny insects; limited taxa.	YOLOv5 top (AP 92.7, Rec 93.8).
10	Deep Learning-Enhanced Hybrid Fruit Fly Optimization for Intelligent Traffic Control in Smart Urban Communities	Kotapati et al.	2023	TTCNN (Energy Layer) + HFFO tuning; sim tests.	Sim only; no real data.	Outperformed others; smart traffic.

Table 3: Review of Context-Oriented, Review-Based Studies

Sr. No.	Title	Citation	Year	Methodology	Limitations	Conclusion
11	Fruits Classification and Detection Application Using Deep Learning	Mimma et al.	2022	YOLOv3 detect; ResNet50/VGG16 classify; 761 imgs (8 fruits) + FIDS-30 (971 imgs, 30 fruits); Flask + Android app.	Small data; monitoring hard.	High acc. (99% custom, 86% public); real-time mobile app.
12	AI-Powered Mobile Image Acquisition of Vineyard Insect Traps with Automatic Quality and Adequacy Assessment	Faria et al.	2021	DCGAN, WGAN, VAE synthesize 7 pests; train YOLOv5m.	Only Peruvian pests; realism limits.	**WGAN/VAE** best; boosted detection acc. with synthetic imgs.

3. MATERIALS AND METHODS

2.1 IOT based Framework

The IoT framework features a fully autonomous system using intelligent electronic traps based on modified McPhail designs, requiring no human involvement [22]. It is referred to as electronic because it is equipped with electronic components which are used to collect and transfer the image data. McPhail traps are actually a bowl or bucket as shown in figure 1, shaped structures which are used to store luring substance or liquid attractant [23].



Figure 1: Sample McPhail Trap

Methyl Eugenol and yellow sticky pads are used to attract fruit flies, with an ESP32 camera capturing images at regular intervals. These images are analyzed using a YOLOv5 model trained on the DIRT dataset, and if harmful flies are detected, the farm owner is alerted via WhatsApp and email. YOLO-TLP Grids Architecture as shown in figure 2. Methyl Eugenol and yellow sticky pads are used to attract fruit flies, with an ESP32 camera capturing images at regular intervals. These images are analyzed using a YOLOv5 model trained on the DIRT dataset, and if harmful flies are detected, the farm owner is alerted via WhatsApp and email.

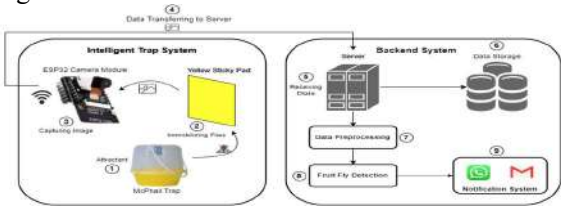


Figure 2: YOLO-TLP Grids Architecture

2.2 ESP32 Camera Module and Yellow Sticky Pad

The ESP32-CAM is a compact, low-cost IoT development board with Wi-Fi, Bluetooth, flashlight, GPIO pins, and an OV2640 2MP camera supporting JPEG, BMP, and grayscale formats up to 1600×1200 resolution. Shown in figure 3, It features a 32-bit CPU at 160 MHz, 520 KB SRAM, expandable PSRAM, TF card storage up o 4 GB, and operates on 5 V power, making it ideal for applications like pest detection, surveillance, and smart devices.

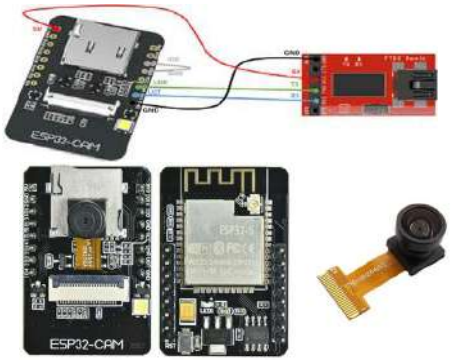


Figure 3: ESP32-CAM Module and OV2640 Camera

The ESP32-CAM includes an LED flash for clear image capture, operates between -20°C and 85°C, and supports data storage from -40°C to 90°C. It offers multiple sleep modes to reduce power consumption and features a schematic pinout for the AI-Thinker module as shown in figure 4.

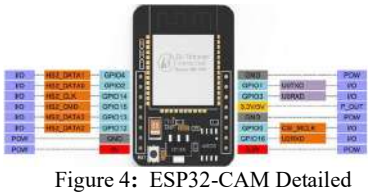


Figure 4: ESP32-CAM Detailed PinOut Diagram

Since the ESP32-CAM lacks a USB port, an external FTDI adapter is used for programming, set to 5V to match the device's requirements. Connection details between the ESP32-CAM and FTDI adapter are illustrated in the figure 5. Jumper wires are used to connect the ESP32-CAM to the FTDI programmer, with "IO0" linked to "GND" for code uploading. The "5V", "U0R", "U0T", and "GND" pins of the ESP32-CAM connect to the FTDI's "VCC", "TX", "RX", and "GND" respectively see in Figure 5.

Figure 5: Connection between FTDI Adapter & CAM

Yellow sticky pads attract fruit flies due to their natural preference for yellow, and work best with a nearby liquid lure. Once flies land on the pad, they stick, allowing images to be captured for monitoring.

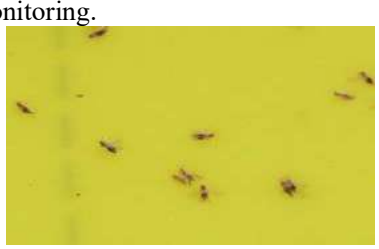


Figure 6: Sample Yellow Sticky Pad

2.3 Server

A server provides services to clients in a client/server model, handling requests via a public IP. It can be physical or virtual, running an OS and application, often using multi-threading for multiple clients. In this work, ESP32-CAM modules capture images and send them to the server, which stores files, runs **YOLOv5 inference**, and sends alerts on fruit fly detections. Inference is server-side due to ESP32-CAM's limited RAM (520 KB) and risks of overheating, power drain, and chip damage with edge computing [24].

2.4 DIRT-Dataset Detailed Description

The DIRT dataset, collected in Greece from 2015–2017 using McPhail traps, contains mostly olive fruit fly images. Data was gathered via the e-Olive smartphone app, which lets users upload trap images and submit fly capture reports. The DIRT dataset contains 848 non-standardized images, mostly with at least one olive fruit fly. Blurry, clustered, or irrelevant insect images were discarded. Only floating olive fruit flies were manually annotated using Label-Img by an entomology expert, while other species were treated as background. The DIRT dataset contains 848 non-standardized images, mostly with at least one olive fruit fly. Blurry, clustered, or irrelevant insect images were discarded [25]. Only floating olive fruit flies were manually annotated using Label-Img by an entomology expert, while other species were treated as background.



Figure 7: Sample Images from DIRT Dataset

Easily distinguishable olive fruit flies were annotated, while clustered ones were not. Unlike the original dataset, male and female



flies were treated as a single class, making the task a binary classification: olive fruit flies (positive) vs. everything else (negative). Sample annotated images from DIRT dataset are see in Figure.

Figure 8: Sample Annotated Images from DIRT Dataset

[26] The original dataset images were large, so previous work sliced them into four parts to detect small objects, but this caused truncated flies appearing in multiple images. Instead, the authors resized images to 1280×1280 and used YOLOv5, as it can detect small objects without slicing.

2.5 Data Pre-processing

Data preprocessing prepares raw data for efficient model training and better generalization. In vision tasks, it reduces complexity and boosts performance. Grayscale is useful when color is less important (e.g., cars, people), but not in domains like medical imaging [27]. For the DIRT dataset, preprocessing included color normalization, homomorphic filtering, grayscale conversion, and min-max normalization using OpenCV.

2.6 Yellow Mean Color Normalization

DIRT dataset images face light variation from sun and shadows. Stored in RGB (0–255), large pixel differences hinder training. Since fruit flies are more attracted to yellow traps, **a** Yellow Mean Color Normalization (YMCN) method was proposed. It unifies the yellow backdrop by averaging red and green channels, then applies multiplicative gains to RGB channels for illumination invariance. These gains adjust pixel intensities in each channel, and the effect of this normalization is demonstrated on sample DIRT images is shown in figure 9.



Figure 9**:** Sample Images (a,c) before YMC Normalization and (b,d) after YMC Normalization

After applying Yellow Mean Color normalization, pixel value ranges shrink and become more centered, reducing the mean and standard deviation of each channel. This helps the learning algorithm generalize better and extract more meaningful features from the images.

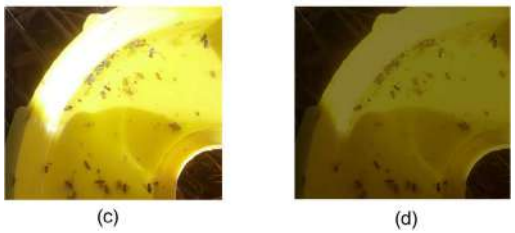


Figure 10**:** Sample Images (a,c) before YMC Normalization and (b,d) after YMC Normalization

2.7 Homomorphic Filtering

Homomorphic filtering, developed in the 1960s, is a nonlinear technique that transforms signals into a different domain for linear filtering and then back again. In image processing, it is widely used for enhancement and correcting non-uniform illumination.

Homomorphic filtering enhances images by improving contrast and normalizing brightness while also removing multiplicative noise. It models an image as the product of illumination (lighting conditions) and reflectance (object properties), with pixel intensity representing their combined effect.

$$\text{Image}(a,b) = L(a,b) \times R(a,b)$$

In images, illumination (L) changes gradually, while reflectance (R) can vary sharply, especially at edges. Their multiplication causes multiplicative noise. To separate them, the image is transformed into the log domain, converting the multiplicative components into additive ones.

$$\text{in}(\text{Image}(a,b)) = \ln(L(a,b) \times R(a,b))$$

$$\ln(\text{Image}(a,b)) = \ln(L(a,b)) + \ln(R(a,b)))$$

After converting to the log domain, a high-pass filter is applied to retain high-frequency reflectance details while removing low-frequency illumination effects. This works

because illumination varies gradually, while reflectance changes sharply, especially at object edges and boundaries are shown in figure 11. Homomorphic filtering was applied after converting images to gray-scale using OpenCV, as different Python libraries handle gray-scale conversion differently in accuracy, computation time is below Table 3.



Figure 11: Homomorphic Filtering Process

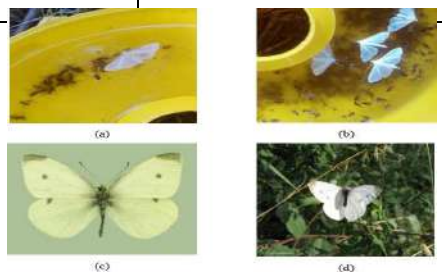
2.8 Gray-scale Conversion

Gray-scale images use a single channel (0–255 intensity) with multiple shades of gray, unlike binary black–white images. They often improve classification accuracy and reduce computation compared to RGB, though effectiveness depends on context. For tasks where color is irrelevant, gray-scale boosts efficiency, [28] but for data like satellite imagery, color loss can reduce accuracy. In this study, DIRT dataset images were converted to gray-scale using OpenCV in Python.2.9 YOLO9000 (YOLOv2):

YOLOv2, introduced in December 2016 as an improved version of YOLOv1, could detect over 9000 object classes (YOLO9000). On the VOC 2007 dataset, it achieved 76.8 mAP at 67 FPS, with key improvements aimed at enhancing recall, reducing localization errors, and preserving accuracy.

Table 4: The mean and standard deviation of RGB channels

	Before Yellow Mean Color		**After Yellow Mean Color**	
Blue Channel	**36.9 Mean (μ)**	**33.402 Normalization Standard Deviation (σ)**	**3.689 Mean (μ)**	**5.923 Normalization** Standard Deviation (σ)
Red	184.05	44.726	101.383	**24.724**
Green	**148.451**	**40.768**	**65.763**	**18.075**



2.10 Min-Max Normalization using OpenCV

Images were normalized using OpenCV's min-max normalization, where the minimum pixel value maps to alpha (lower limit), the maximum to beta (upper limit), and all other values are linearly scaled in between. A sample image see in Figure 12, represents the effect of min-max normalization using OpenCV.



Figure 12: Sample DIRT images: (a) Original Image, (b) min-max Normalized Image

2.11 Exploiting Negative Training using Leeds Butterfly Dataset

False alarms (low recall) can be reduced by adding non-target objects during training. To improve recall, the **Leeds Butterfly dataset** (832 PNG images of 10 butterfly species

collected via Google search) was used as negative samples, since butterflies also appear in some DIRT images.

Figure 13: Sample DIRT Images containing Butterflies (a,b)

3.1 Object Detection Models

Object detection identifies and classifies objects in images [29]. Traditional methods use manual features like SIFT and HOG with classifiers, while modern CNN-based models learn features automatically with higher accuracy. Widely applied in surveillance, traffic, face recognition, and pest detection, [30] popular models include R-CNN, YOLO, and SSD. This work employs **YOLOv5** for its efficiency and strong small-object detection.

3.1.1 YOLOv1

You Only Look Once (YOLO) [31], is a fast object detector (up to 155 FPS) with good mAP but weaker accuracy for small objects. Its architecture has 24 conv + 2 FC layers, dividing images into grids where each cell predicts bounding boxes, confidence, and class probabilities. Tested on Pascal VOC (20 classes), YOLOv1 was limited to one–two objects per grid cell and max 98 objects per image, making it unsuitable for detecting multiple small flies in this work.

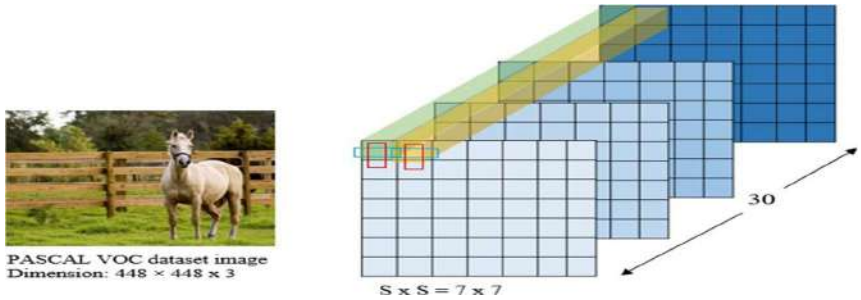
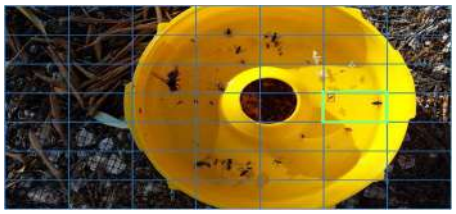


Figure 14: Grid architecture of YOLOv1 model

YOLOv1 divides images into 7×7 grids (7×7×30 output). However, it struggles with small objects since grid size often exceeds object dimensions. For example, in a DIRT dataset image (1552×2050), the grid cell size is ~221.7×292.8, while a fruit fly measures



151×137, making detection difficult.

3.1.2 YOLOv3

YOLO9000 introduced **Darknet-19/53**, anchor boxes, multi-scale training, and batch normalization, improving accuracy and predicting up to 845 objects per image [32].

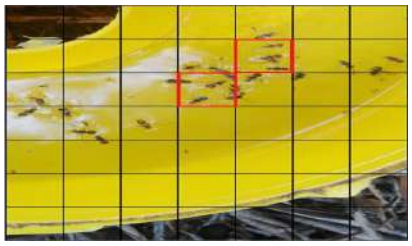


Figure 15: Multiple flies in same Grid cell (Red highlighted)

However, it struggles with very small objects like fruit flies as show in figure 16, making it unsuitable for the DIRT dataset [33]. YOLOv3 improved further with multi-scale feature extraction (3 scales), 9 anchors, and higher AP (57.9 vs 44 of YOLOv2), allowing prediction of up to 1521 objects.

Despite this, YOLOv3 still fails on extra-small flies and on very large objects, so it is also unsuitable here [34]. YOLOv3 distributes 9 anchor priors across 3 scales and significantly improves performance, achieving 57.9 AP50 vs 44 AP50 of YOLOv2

on MS COCO at 608×608 resolution. Its predictions are encoded as (S × S) × (B × (5+C)), with S=13, B=9, and C=80, resulting in an output tensor of (13×13) × 225. This allows the model to predict up to 1521 objects per image. YOLOv3 can predict up to 9 objects per grid cell, but struggles with very small objects. In the DIRT dataset, grid dimensions (≈86.8 × 49.1) are much larger than fruit flies (≈26 × 40), making precise detection difficult. While multi-scale training improves small object detection, extra-small objects like fruit flies remain challenging for the model. YOLOv3 performs poorly on medium and especially large objects. For example, in a 1080 × 1032 image, a fruit fly of 918 × 888 size nearly matches the image dimensions, making detection inaccurate. Due to such limitations, YOLOv3 is unsuitable for this problem scenario

Figure 16: Very Small fly in a grid cell

3.1.3 YOLOv4

In 2020 deep learning framework for object detection that builds on prior work while introducing new ideas. Its main goal is to increase CNN speed and optimize parallel computation. Modern detectors consist of a backbone (feature extraction), a neck (feature aggregation), and a head (object prediction). YOLOv4 uses CSPDarknet53 as backbone to expand receptive field without reducing speed, PANet for feature aggregation instead of FPN, and the YOLOv3 head for predictions. It also employs Bag of Freebies (accuracy improvements during training) and Bag of Specials (post-processing techniques) to further boost performance. Overall, YOLOv4 aims to balance speed, accuracy, and scalability for detecting objects of different sizes.

YOLOv4 improves detection accuracy with a slight drop in inference speed by using data augmentation techniques (CutMix, MixUp, CutOut, cropping, translation, rotation), feature aggregation methods, and architectural enhancements like MiWRC, CSP, Mish activation, residual and weighted residual connections, skip connections, dropout, and batch normalization.

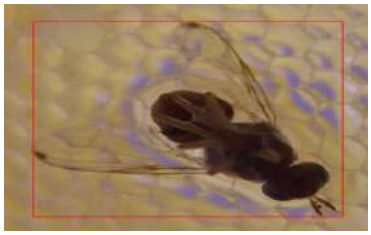


Figure 17: No Fly Detection Scenario

YOLOv4 introduces Mosaic augmentation (stitching 4 images for context diversity) and Self-Adversarial Training (SAT) to improve robustness. It achieves 65.7 AP50, outperforming YOLOv3 (57.9) and YOLOv2 (44) at 608×608 resolution as shown in Figure 4.6, Here's a concise summary: YOLOv4 struggles to detect fruit flies due to mismatched grid and object dimensions, despite using 12 anchors. Given these limitations and the availability of the more robust and deployment-friendly YOLOv5, YOLOv4 was not chosen.



Figure 19: YOLOV4 Limitation Scenario

3

. RESULTS AND DISCUSSION

3.1 YOLOv5

YOLOv5 is a widely used deep learning-based object detection algorithm that reframes detection as a regression problem, predicting class probabilities and object coordinates via a loss function. Building on the YOLO framework, it incorporates modern training optimizations and divides the input image into

a 19 × 19 grid for detection as see in Figure 20. YOLOv5 performs detection at three levels with three bounding boxes per level. Key advancements include auto-learning bounding box shapes via evolutionary algorithms, mosaic data augmentation, and Cross-Stage Partial Networks, enabling efficient multi-stage feature extraction. Its architecture is divided into four stages: input (with preprocessing and augmentation), backbone, neck, and head [35]. In YOLOv5, auto-anchors are selected before training to match the dataset. The backbone then extracts features using techniques such as Spatial Pyramid Pooling (SPP) for improved representation. cross-stage partial networks (CSP) [36] at different sizes. These techniques utilize multiple convolutional as well as pooling operations.

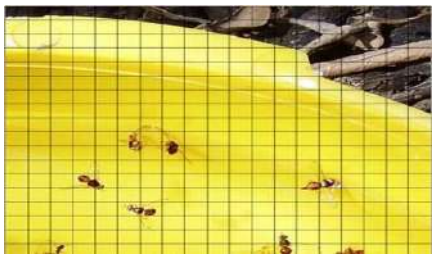


Figure 20: 19 × 19 grid applied to a sample image of DIRT Dataset

The YOLOv5 backbone extracts features at three scales to improve accuracy, while the neck combines low- and high-level features, and the head predicts targets of varying sizes. YOLOv5 comes in variants (n, s, m, l, x), differing in feature extraction layers, convolution layers, input size, and kernel size.

YOLOv5 versions differ in size, accuracy, and computational needs see in Table 4. YOLOv5n is lightweight and suited for low-compute devices but has lower accuracy (mAP 45.7). Larger versions achieve higher mAP but require more parameters and computational resources, measured in GFLOPS.

Table 4**:** Difference between different YOLOv5 versions

Model	Parameters (million)	GFLOPS	mAP0.5 validation
YOLOv5n	1.9	4.5	45.7
YOLOv5s	7.2	16.5	56.8
YOLOv5m	21.2	49	64.1
YOLOv5l	46.5	109.1	67.3
YOLOv5x	86.7	205.7	68.9

3.2 Computing Platforms for Model Training

A modified YOLOv5 model implemented in PyTorch was used for this study, trained and validated on the DIRT dataset. Experiments were conducted on Google Colab Pro, utilizing an Intel Xeon 2.2 GHz CPU, 25 GB RAM, and a Tesla T4 GPU.

3.3 Dataset Brief Overview

The DIRT dataset containing olive fruit fly images from traps in Greece (2015–2017) was used to train the YOLOv5m model. Images varied in dimensions due to different cameras, and expert entomologists annotated them for accuracy. A 5-fold cross-validation was applied, with 80% for training and 10% each for validation and testing. From 900 images, about 720 were used for training, and 90 each for validation and testing per fold. In Table 5 provides the data distribution for a single fold in the cross-validation process.

Table 5: DIRT-Dataset Distribution for 1-fold of Cross-Validation

	DIRT Images
Total images	**900**
Train Set	**720**
Validation Set	**90**
Test Set	**85**
Number of classes	**1**

3.4 Evaluation Metrics

Object detection models are evaluated using precision, recall, F1-score, and mean average precision (mAP), which are derived from True positive, True Negative, False negative, and False positive classifications**.** IoU is a key metric that measures the overlap between predicted and ground truth bounding boxes to determine detection accuracy.

3.5 Hyper-parameters for training YOLOv5

The YOLOv5m6 model was trained and evaluated using a 5-fold cross-validation strategy with **transfer learning** from MS-COCO pre-trained weights. The model, with **485 layers and 36.2M parameters**, was trained on **1280×1280 images** for **350 epochs** using **SGD optimizer**. Hyper parameters were largely adapted from the original YOLOv5, optimized via genetic algorithms. **Auto anchor** selected 30 anchor boxes. Key settings included: batch size 8, learning rate 0.0115, momentum 0.97, weight decay 0.00028, IoU threshold 0.25, patience 30, and 4 warmup epochs. Details are also presented in Table 6.

Table 6: Hyper-parameters

Hyper-parameters	Value
Input Image Size	1280×1280
Epochs	300
Patience Value	30
Optimizer	Stochastic Gradient Descent (SGD)
Batch-Size	8
Learning-Rate	0.0115
Momentum	0.97
Weight-Decay	0.00028
IOU-Threshold	0.25
Warmup-Epochs	4
Warmup-Momentum	0.95

3.6 Results

The purpose of this study's experimental aimed to validate the proposed model and compare its performance with state-of-the-art algorithms**,** using different data preprocessing and normalization methods while evaluating results through standard performance metrics.

3.7 YOLOv5m results on original DIRT Dataset

The refined YOLOv5 model was evaluated using **mAP, precision, recall, and F1-score**. Trained with SGD over 300 epochs at 1280×1280 resolution, it achieved 86.4% precision, 84.1% recall, 85.2% F1-score, and 87.9% mAP**.** All other hyper-parameters were retained. For results see in Figure. These results show strong performance, though improvements in recall and overall accuracy are still possible.

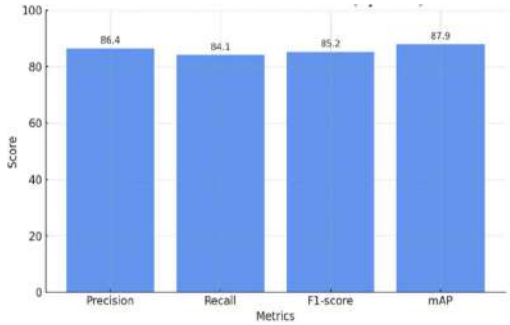


Figure 21: YOLOv5 Results Illustration on Original DIRT Dataset

The fine-tuned YOLOv5m model**,** trained with 1280×1280 images for 300 epochs using SGD, showed superior precision compared to other models on the same dataset. It outperformed YOLOv4 by 12.2% and achieved precision nearly equal to the modified YOLOv4, confirming YOLOv5m's effectiveness and competitive advantage in precision performance.

3.8 Comparison of recall between YOLOv5 and other models

The **YOLOv5m model**, trained with 1280×1280 images for 300 epochs using SGD, showed **lower recall performance** compared to other models. It was outperformed by **YOLOv4 by 7.6%**, with the **modified YOLOv4** achieving the best recall results on the dataset.

3.9 Comparison of f1-score between YOLOv5 and other models

The YOLOv5m model, trained with 1280×1280 images for 300 epochs using SGD and pre-trained MS-COCO weights, showed strong F1-score performance, surpassing YOLOv4 by 3.4%**.** However, its F1-score was slightly lower than the modified YOLOv4, suggesting further improvements could be achieved with techniques like data augmentation**.**

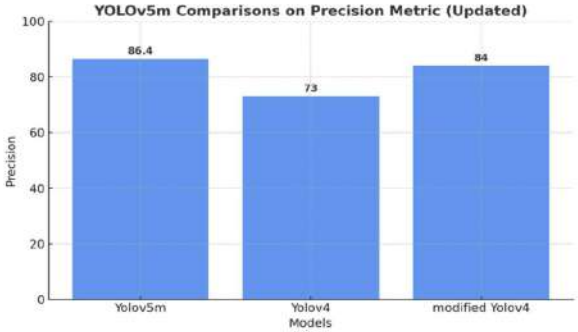


Figure 22: Comparison of Precision between YOLOv5 and other Model

A performance comparison with YOLOv2, YOLOv3, and YOLOv4 under identical training settings (1280×1280 input size, 300 epochs, SGD optimizer, and MS-COCO pre-trained weights) showed that YOLOv5 consistently outperformed the other models across all four evaluation metrics, demonstrating its superior effectiveness on the dataset.

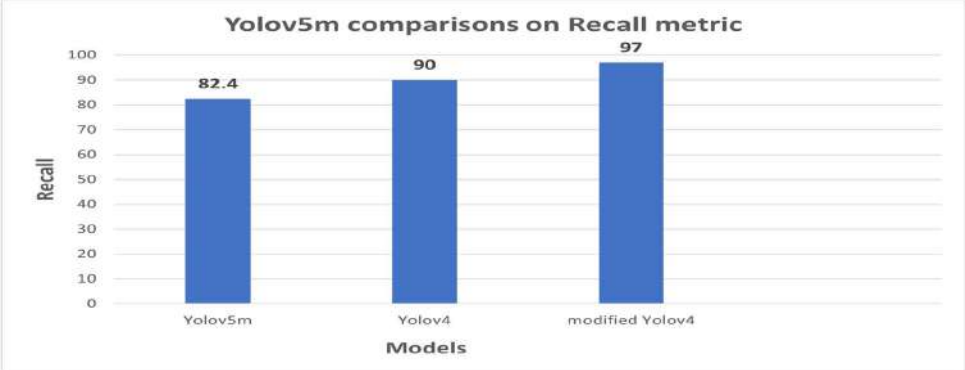


Figure 23: Comparison of recall between YOLOv5 and other Model

3.10 Results of Baseline experiment (yolov5m), Individual components, and overall proposed Framework

The experimental results Table 4.4 present the baseline performance of YOLOv5m on the original dataset, followed by evaluations of individual framework components, including YMC normalization, OpenCV normalization, negative training with the Leeds Butterfly dataset, transfer learning, and data augmentation**.** Finally, the outcomes of the fully integrated framework are reported. Using 5-fold cross-validation for robustness, the results reflect the average performance across folds. Training was conducted with 1280×1280 input resolution, 300 epochs, and the SGD optimizer**,** while other hyper-parameters remained unchanged.

Table 7**:** Experimentation results of YOLOv5m, individual components and proposed integrated framework

Model / Component	Precision	Recall	F1 score	mAP
Baseline (Original YOLOv5m with DIRT)	86.1	83.6	84.8	87.5
YOLOv5m with YMC NORM	88.0	86.3	87.1	89.1
YOLOv5m with Homomorphic Filtering	84.7	80.5	82.5	85.4
YOLOv5m with min-max Norm	84.3	83.1	83.7	86.9
YOLOv5m with DIRT + Leeds Butterfly	83.9	79.2	81.4	85.3
YOLOv5m with YMC Norm and Augmented DIRT	93.2	96.5	94.8	96.8

3.11 YOLOv5 performance evaluation on two-class dataset

A publicly available dataset was used, though only **520 images** were accessible instead of the claimed 1500. The data was split into **416 training, 52 validations, and 52 testing images**. YOLOv5, trained for **120 epochs with transfer learning (MS-COCO weights), 960×960 input resolution, batch size 10, and learning rate 0.011**, achieved strong results: **precision 0.947, recall 0.951, F1-score 0.949, and mAP 0.961**. Despite dataset limitations and no augmentation, the model performed reliably in olive fruit fly detection.

3.12 YOLOv5 performance evaluation on three-class dataset

A multi-class dataset was created by merging 467 images (B. zonata and B. dorsalis) with 848 DIRT dataset images, totaling **1315** labeled images. The data was split into 80% training, 10% validation, and 10% testing**.** YOLOv5 was chosen due to its strong performance in earlier binary classification. For this three-class task, YOLOv5 was trained for 100 epochs using transfer learning with MS-COCO pre-trained weights**.**

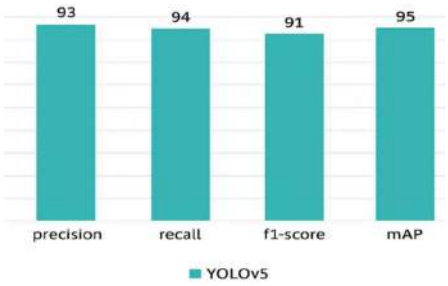


Figure 24**:** YOLOv5 performance on three Class Data

The model was trained with an **image resolution of 1024 × 1024**, **batch size of 8**, and **learning rate of 0.01**, without additional preprocessing or augmentation. YOLOv5 achieved **precision 0.835, recall 0.826, F1-score 0.830, and mAP 0.847** on the three-class dataset. Despite dataset imbalance, results were satisfactory, with room for improvement through future enhancements.

3.13 YOLOv5 Performance Evaluation on Four-Class Dataset

A four-class dataset was built by combining the DIRT (848 olive fruit fly images), AlertTrap (200 mango fly images), and a public dataset (467 Bactrocera zonata and dorsalis images), totaling **1515 labeled images**. Data was split into **80% training, 10% validation, and 10% testing**. YOLOv5 was trained for **100 epochs** with **transfer learning (MS-COCO weights)**, **1024 × 1024 resolution**, **batch size 8**, and **learning rate 0.011**. No preprocessing or augmentation was applied due to time constraints, though the

dataset's **diversity and imbalance** suggest this as a future improvement area.

YOLOv5 performed well on the **four-class detection task**, achieving precision 0.844**,** recall 0.851**,** F1-score 0.847**, and** mAP 0.839**, indicating** encouraging results**.**

3.14 Comparison of Different Detection Models on Three-Class Dataset

A three-class detection dataset of about 1515 labeled images was created by combining DIRT, Alert Trap, and other fruit fly datasets. Four models (YOLOv5, YOLOv4, YOLOv3, YOLOv2) were trained for **100 epochs** with transfer learning. With an input size of **1024 × 1024**, batch size **8**, and learning rate **0.01**, **YOLOv5 outperformed all other models**, achieving the best **precision, recall, F1-score, and mAP**, proving its effectiveness for multi-class fruit fly detection. Comparison of Different Detection Models on Four-Class Dataset. A four-class detection dataset **of about** 1515 labeled images was built by merging the DIRT, Alert Trap, and a two-class fruit fly dataset. The data was split into 80% training, 10% validation, and 10% testing**.** Four models YOLOv5, YOLOv4, YOLOv3, and YOLOv2 were trained for 100 epochs with transfer learning, using an input size of **1024 × 1024**, batch size **8**, and learning rate 0.01**.** YOLOv5 outperformed all other models**,** achieving the best **precision, recall, F1-score, and mAP** for the four-class detection task



Figure 25: Combined Images Visual results of YOLOv5 on three-class dataset

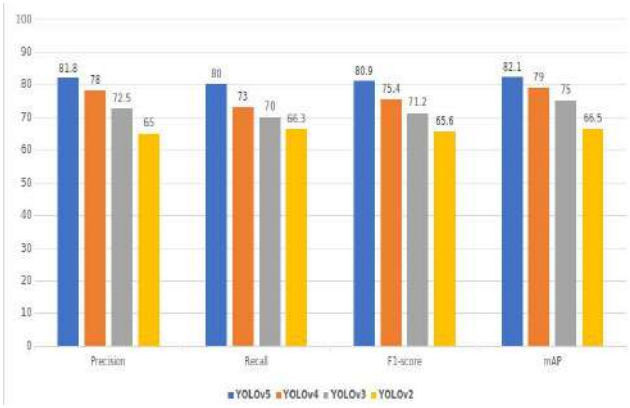


Figure 26: Comparison of different detection models on three-class Dataset

4. SUMMARY

Fruit flies are serious agricultural pests, and manual monitoring is laborious and error-prone. This study uses **YOLOv5** for automatic detection of multiple fruit fly species (*B. oleae*, *B. zonata*, *B. dorsalis*, mango fly) across datasets (DIRT, AlertTrap, two-class). YOLOv5 was selected for its **real-time accuracy and edge-device compatibility**. Models were trained with transfer learning (MS-COCO) and consistent settings, evaluated by precision, recall, F1, and mAP. An integrated framework combining normalization, augmentation, and negative training achieved up to **93.2% precision, 96.5% recall, 94.8% F1, and 96.8% mAP**, outperforming earlier YOLO versions. Results confirm YOLOv5's effectiveness for pest detection and its potential for smart traps, though larger datasets, advanced augmentation, and field testing are needed for further improvement.

In conclusion, this thesis demonstrates that deep learning models, specifically YOLOv5, offer a reliable and efficient solution for automated fruit fly detection. The work lays the foundation for the development of scalable, intelligent pest monitoring systems that can contribute significantly to sustainable agriculture and effective pest management practices.

5. REFERENCES

[1] Parker, M.A. 2013. Beyond the standard model. Phys Scr T158.

[2] Zhong, Y., J. Gao, Q. Lei and Y. Zhou. 2018. A vision-based counting and recognition system for flying insects in intelligent agriculture. Sensors (Switzerland) 18.

[3] Unnevehr, L.J. 2007. Causes of and constraints to agricultural and economic development: Discussion. Am J Agric Econ 89:1168.

[4] Hazarika, L.K., M. Bhuyan and B.N. Hazarika. 2009. Insect pests of tea and their management. Annu Rev Entomol 54:267–284.

[5] Miller, G.T., S. Spoolman, S. The, E. Download, P.D.F. Sustaining, T. Earth, P. Download, F. Download, S. The, E. Full, V.G. Tyler, S. Spoolman and S. The. n.d. PDF Sustaining The Earth CLICK FOR DOWNLOAD.

[6] Cho, J., J. Choi, M. Qiao, C. Ji, H. Kim, K. Uhm and T. Chon. 2007. Automatic identification of whiteflies, aphids and thrips in greenhouse based on image analysis. Iinternational Journal of Mathematics and Computers in Simulation 1:46–53.

[7] Zhang, W. and S.M. Swinton. 2009. Incorporating natural enemies in an economic threshold for dynamically optimal pest management. Ecological Modelling 220:1315–1324.

[8] Ding, W. and G. Taylor. 2016. Automatic moth detection from trap images for pest management. Comput Electron Agric 123:17–28.

[9] Hahn, F., S. Valle, R. Rendón, O. Oyorzabal and A. Astudillo. 2023. Mango Fruit Fly Trap Detection Using Different Wireless Communications. Agronomy 13.

[10] Nuankaew, W.S., Y. Yangwattana, S. Kamwaree, T. Sararat and P. Nuankaew. 2025. Harnessing AI for Agriculture: Plant Pest Detection on Web Application Using Deep Learning. 2025 Jt. Int. Conf. Digit. Arts, Media Technol. with ECTI North. Sect. Conf. Electr. Electron. Comput. Telecommun. Eng. (ECTI DAMT & NCON) 588–593.

[11] Verma, S.K. 2025. Exploring the Role of AI in Autonomous Drone Technology for Precision Pest Control in Agriculture. Int. Res. J. Eng. Appl. Sci. | irjeas.org 13:19–28.

[12] Godavarthi, M.S. and K.C. Lekha. 2025. AI-POWERED CROP PEST DETECTION: A VISUAL APPROACH. Int. J. Manag. Res. Bus. Strateg. 15:1–8

[13] Ghosh, P., A. Markad and C. Maharana. 2025. AI-Driven Crop Monitoring and Management for Sustainable Agriculture. Transform. Agric. through Artif. Intell. Sustain. Food Syst. 39–53.

[14] Gangadevi, E., R.S. Rani, R.K. Dhanaraj and A. Nayyar. 2024. Spot-out fruit fly algorithm with simulated annealing optimized SVM for detecting tomato plant diseases. Neural Computing and Applications 36:4349–4375.

[15] Ong, S.Q. and T.T. Høye. 2024. An annotated image dataset of pests on different coloured sticky traps acquired with different imaging devices. Data in Brief 55:110741.

[16] Ahmed, S., S.N.K. Marwat, G. Ben Brahim, W.U. Khan, S. Khan, A. Al-Fuqaha and S. Koziel. 2024. IoT based intelligent pest management system for precision agriculture. Sci Rep 14:1–15.

[17] Wayama, R., Y. Sasaki, S. Kagiwada, N. Iwasaki and H. Iyatomi. 2024. Investigation to answer three key questions concerning plant pest identification and development of a practical identification framework. Computers and Electronics in Agriculture 222:109021.

[18] Bjerge Id, K., J. Alison, M. Dyrmann, C.E. Frigaard, H.M.R. Mann and T.T. Høye. 2023. Accurate detection and identification of insects from camera trap images with deep learning. PLOS Sustainability and Transformation 2:e0000051.

[19] Kotapati, G., M.A. Ali and R. Vatambeti. 2023. Mechatronics and Intelligent Transportation Systems Deep Learning-Enhanced Hybrid Fruit Fly Optimization for Intelligent Traffic Control in Smart Urban Communities. Intell Transp. Syst 2:89–101

[20] Mimma, N.E.A., S. Ahmed, T. Rahman and R. Khan. 2022. Fruits Classification and Detection Application Using Deep Learning. Scientific Programming 2022:4194874.

[21] Faria, P., T. Nogueira, A. Ferreira, C. Carlos and L. Rosado. 2021. AI-Powered Mobile Image Acquisition of Vineyard Insect Traps with Automatic Quality and Adequacy Assessment. Agronomy 2021, Vol. 11, Page 731 11:731.

[22] Xiong, B., D. Li, Q. Zhang, N. Desneux, C. Luo and Z. Hu. 2024. Image detection model construction of Apolygus lucorum and Empoasca spp. based on improved YOLOv5. Pest Management Science 80:2577–2586.

[23] Cardoso, B., C. Silva, J. Costa and B. Ribeiro. 2022. Internet of Things Meets Computer Vision to Make an Intelligent Pest Monitoring Network. Applied Sciences (Switzerland) 12

[24] Almubarak, A.A. 2017. The Effects of Heat on Electronic Components. Adel Ahmed Almubarak. Int. Journal of Engineering Research and Application www.ijera.com 7:52–57.

[25] Lebrini, Y. and A. Ayerdi Gotor. 2024. Crops Disease Detection, from Leaves to Field: What We Can Expect from Artificial Intelligence. Agronomy 14.

[26] Wang, J. and K. Markert. n.d. Learning Models for Object Recognition from Natural Language Descriptions Mark Everingham 2021.

[27] Subha, S., D.C.J.W. Wise, S. Srinivasan, M. Preetham and B. Soundarlingam. 2020. Detection and Differentiation of Skin Cancer from Rashes. Proceedings of the International Conference on Electronics and Sustainable Communication Systems, ICESC 2020 389–393.

[28] Bui, H.M., M. Lech, E. Cheng, K. Neville and I.S. Burnett. 2016. Using grayscale images for object recognition with convolutional-recursive neural network. 2016 IEEE 6th International Conference on Communications and Electronics, IEEE ICCE 2016 321–325.

[29] M., H., A. S., A. Mohammed and R. Jennane. 2017. Histogram of Oriented Gradients and Texture Features for Bone Texture Characterization. International Journal of Computer Applications 165:23–28.

[30] Valueva, M. V., N.N. Nagornov, P.A. Lyakhov, G. V. Valuev and N.I. Chervyakov. 2020. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. Mathematics and Computers in Simulation 177:232–243.

[31] Plastiras, G., C. Kyrkou and T. Theocharides. 2018. Efficient convnet-based object detection for unmanned aerial vehicles by selective tile processing. ACM International Conference Proceeding Series, doi: 10.1145/3243394.3243692;WGROUP:STRING:ACM.

[32] Redmon, J. and A. Farhadi. 2018. YOLOv3: An Incremental Improvement.

[33] Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg and L. Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision 115:211–252.

[34] Zhao, Y., R. Han and Y. Rao. 2019. A new feature pyramid network for object detection. Proceedings - 2019 International Conference on Virtual Reality and Intelligent Systems, ICVRIS 2019 428–431.

[35] Yang, W. and X. Qiu. 2024. A Novel Crop Pest Detection Model Based on YOLOv5. Agriculture 2024, Vol. 14, Page 275 14:275.

[36] Wang, C.-Y., H.-Y.M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh and I.-H. Yeh. 2020. CSPNet: A New Backbone That Can Enhance Learning Capability of CNN.