MACHINE LEARNING BASED REAL-TIME DETECTION OF RIPE AND UNRIPE SORGHUM

Audu, A.I.[1*] and Ndirmbula, S.M.[2]

[1]Department of Computer Engineering, University of Maiduguri, Nigeria

[2]Department of Computer Engineering, University of Maiduguri, Nigeria

*Corresponding email: Abdulkadir.Iyyaka@gmail.com

ABSTRACT

The physiological maturity, which is a conventional way of detecting the ripeness of Sorghum involves moisture detection and physical sampling of the crop. It involves plucking out a few grains from the panicle and check the point of attachment, if the tip is brownish-black, it signifies that the grain is mature. However, this method of ripeness detection is destructive, time-consuming, labor intensive particularly in large farms and inaccurate. Another important issue that causes a reduction in yields and economic losses is delayed harvesting, which can result in grain loss due to pest infestation, shatter, and sprouting. Sprouting becomes an issue when sorghum grain has reached maturity and is exposed to long periods of wet, warm weather. This work proposes to develop a machine learning (ML) based system to detect the ripeness of Sorghum and alerts the farmer to harvest at the appropriate time. It encompasses the training of a machine learning model to effectively learn patterns within a given dataset. The dataset was annotated using Roboflow and divided into training, validation and test set. The trained model is deployed onto a Raspberry pi using Python code, allowing the model to operate directly on the device to execute the model's predictions locally enabling it to make accurate predictions of the state of the crop. Finally, it Utilizes a camera sensor on the Raspberry device, to perform real-time inference with the trained model. YOLOv5 object detector model is used. Precision, Recall, F1 score and mean average precision (mAP) are the evaluation metrics used to gauge the performance of the model. The trained model is successfully deployed on the raspberry pi to achieve a real-time detection on the field. The real-time system exhibits an overall Precision of 79.5%, a Recall of 87.3%, an F1 score of 83%, and an mAP of 87% using the YOLOv5 object detector model. For future work, expanding the size of the dataset and using a more robust model would improve the accuracy and efficiency of detecting the ripeness of the crop. The model could be further integrated into robotic systems and drones for automatic harvest and to compute the yield percentage.

Keywords: Sorghum, machine learning, Raspberry pi, YOLOv5, training, object detection, convolutional neural network, Python and evaluation metrics

1 INTRODUCTION

Sorghum (Sorghum bicolor) is a well-known grain in the world food market. Of all the species of Sorghum, it is only Sorghum bicolor that is native to Africa, that is produced for human consumption. Sorghum is known for its ability to withstand adverse conditions like drought, heat, and water-logging caused by erratic rainfall. This is just one of the reasons it is considered a good alternative to planting maize [1]. It comes in a distant fifth for most produced grains—behind barley, rice, wheat, and corn. While dwarfed by better known grains (or cereals), sorghum is an important crop that has long played

a vital role in certain diets [2]. Nigeria is the largest producer of sorghum in West Africa, grown on about 5.9 million acre with current annual production estimated to be about 6.7 million tonnes, and third largest after the United States and Asia, globally but first in food grain production [2]. However, its yield is hindered by various factors such as weather conditions, soil moisture, and pests. To help farmers optimize their sorghum harvests and to prevent pests from infesting a sorghum harvesting alert system can be designed to provide timely alerts. Due to the fact that accurate crop recognition and categorization is a critical component of an automated harvesting system, numerous researchers have worked to create reliable algorithms. Although crop detection systems have performed considerably better in recent years, they are still far from being used in real-world situations. The unpredictable and complicated environment of the orchards, which includes naturally varying illumination, indistinguishable backdrops, extreme overlap and occlusion conditions, and low resolutions, are the obvious hurdles in designing such a crop-detecting system. ML techniques have been implemented in several crop detecting systems over time [3]. As a component of Artificial Intelligence (AI), computer vision technology permits computers and systems to take action and offer recommendations after deriving relevant information from digital images, videos, and other visual data [4]. One popular usage of computer vision is object detection. Algorithms for object detection can forecast an object's class and location inside an image, while those for image classification may distinguish between many image classes. According to [5]: Recognition of Bloom/Yield in Crop Images Using Deep Learning Models for Smart Agriculture, using natural resources in a variety of environments to increase yields is vital for precision agriculture. A crop's yield may change from year to year depending on changes in the environment, the characteristics of the soil, and the fertilizers applied. Agriculture-related automation in the post-pandemic environment, industry can improve food quality while moderating resource utilization. For crop seeding, crop monitoring, weed control, pest management, and harvesting, agricultural robots have been developed. For the purpose of estimating agricultural production, physical counting of fruitlets, flowers, or fruits at various stages of growth is a time-consuming and expensive process. Crop yield estimation and prediction are made more accurate and reliable by remote sensing technologies. automated picture analysis using deep learning and computer vision provides precise field and yield maps. It has been noted in this review that the use of deep learning techniques has improved precision for intelligent farming. Fruits including grapes, apples, oranges, and tomatoes, as well as vegetables like corn, soybeans, cucumbers, and maize, were chosen for the study. The research that was done for this study is now commercially available as products for uses like pest infestation, weed detection, and robot harvesting. The algorithms that used traditional deep learning techniques offered an accuracy of 92.51% on average. This study explains many automated methods for classifier-based virtual analysis of crop yield detection systems. A 'Real-Time Ripe Oil Palm Fresh Fruit Bunch Detection System Based on YOLOv4' was proposed by [6]. Using live camera input, a Conventional Neural Network (CNN) model named YOLOv4 was employed to detect the presence of ripe Fresh Fruit Bunches (FFBs) on the oil palm trees in real-time. Once a ripe FFB is located on the tree, a signal is transmitted over ROS to the robotic harvesting system. To train the YOLOv4 model, a vast amount of ripe FFB images were collected using an Intel Realsense Camera D435 with a 1920 x 1080 resolution. During the data gathering procedure, a subject matter expert provided assistance in classifying the FFBs as ripe or unripe. The testing phase yielded results for the mean Average Precision (mAP) and recall of 87.9% and 82%, respectively, because the detection is satisfied the Intersect over Union (IoU) with more than 0.5 after 2000 iterations and the system worked at a real-time speed of around 21 Frame Per Second (FPS). In [7], the work enhances test time augmentation for deep learning sorghum head detection. Sorghum yields can be estimated from the number of crop heads in different branching arrangements. For different crops, artificial intelligence and image processing-based techniques have shown to be successful in rapidly and autonomously collecting this kind of data. However, the application of sorghum heads to sorghum crops presents additional challenges due to their variation in size, colour, and shape. This study looked into and evaluated a method for identifying sorghum heads in photos taken by unmanned aerial vehicles. A Yolov4-Based Wheat Head Detection Technique is presented in [8]. First, two Spatial Pyramid Pooling (SPP) networks are added to improve the backbone network portion. This increases the convolutional network's receptive field and improves feature learning. Second, to obtain multilevel features, the top-down and bottom-up feature fusion procedures are used. Finally, it forecasts the

bounding box of the item using Yolov3's head structures. The findings demonstrate the improved speed and accuracy of the suggested wheat head detection method. The suggested method's detection speed is 88 frames per second, and its mean average precision is 94.5%. Based on Yolov4, the suggested deep neural network can swiftly and precisely identify the wheat head in the image. Furthermore, the suggested method is more resilient and has a wide range of application values because the training dataset is a wheat head dataset with rich variations and accurate annotations. An optimized YOLO-based object recognition model for agricultural harvesting systems was the focus of [3]. The project's objective is to collect visual data from the plantation that will be necessary for developing an automated, intelligent crop harvesting system. The goal of the project is to develop an automated detection system that is extremely accurate, low-cost to compute, and lightweight. Taking into consideration the advantages of YOLOv3 tiny, an enhanced YOLOv3 tiny network dubbed YOLO-P is shown to detect and localize three items at a palm oil plantation: a fresh fruit bunch, a grabber, and a palm tree, under a range of environmental conditions. The suggested YOLO-P model's lightweight framework, with an optimized anchor box size and multi-scale detection architecture, was constructed on top of a densely connected neural network. The results of the trial demonstrated that the proposed YOLO-P model performed well, with a mean average precision and F1 score of 98.68% and 0.97, respectively. Furthermore, the proposed model had a faster training process and generated a lightweight model of 76 MB. Tests were also conducted on the suggested model, which identified fresh fruit bunches of different ages with an accuracy rate of 98.91%. The comprehensive experimental results show that the proposed YOLO-P model can effectively perform accurate and dependable detection in the palm oil plantation. A method for detecting foxtail millet ear based on YOLOv4 and adaptive anchor box adjustment was proposed by [9]. Initially, a dataset called Ear detection was created, comprising 10,000 ear samples and 784 pictures. Additionally, a novel method for quickly and precisely identifying the foxtail millet ear in a particular box was devised, based on YOLOv4 (You Only Look Once). Several parameters, including the mean average Precision, F1-score, recall, and mAP, were used to confirm the efficacy of the suggested method. The YOLOv4 received scores of 78.99% and 83.00% in terms of mean average precision and f1 score, respectively. The Precision was 87% and the Recall was 79.00% across all criteria, which was approximately 8% better than the YOLOv2, YOLOv3, and Faster RCNN. The proposition by [10] is on 'Automatic Jujube Fruit Detection and Ripeness Inspection Method in the Natural Environment', uneven ambient variables such as lighting changes, leaf occlusion, overlapping fruits, colors or brightness, comparable plant look to the background, and so on, the ripening phases of jujube fruits are one issue preventing fruit recognition. As a result, an approach dubbed YOLO-Jujube was suggested to address these issues. The networks of Stem, RCC, Maxpool, CBS, SPPF, C3, PANet, and CIoU loss were incorporated, allowing YOLO-Jujube to automatically recognize jujube fruit for ripeness assessment. YOLO-Jujube surpassed the network of YOLOv3-tiny, YOLOv4-tiny, YOLOv5s, and YOLOv7-tiny with params of 5.2 m, GFLOPs of 11.7, AP of 88.8%, and a speed of 245 fps for detection performance, including the sorting and counting process combined. YOLO-Jujube is capable and useful for achieving the objective.

2 MATERIALS AND METHOD

2.1 SOFTWARE/LIBRARIES

The softwares employed towards the realization of set objectives are mentioned. The details of the implementation tools are presented.

2.1.1 Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python offers a wide range of libraries and frameworks specifically designed for computer vision and object detection projects.

2.1.2 *Roboflow*

Roboflow is an end-to-end computer vision platform that simplifies the process of building computer vision models. It offers detailed statistics for dataset quality improvement, dataset class breakdown, and annotation count visualize image sizes and aspect ratio breakdowns and visual representation of annotation locations within your dataset.

2.1.3 *TensorFlow*

TensorFlow is used to create a fast-numerical calculation and was released by Google. This is a latest Python library. Create deep learning models directly, using this foundation library. Also, the wrapper libraries which help to create top of TensorFlow. This is one type of math library and machine learning applications like neural networks. There are different types of deep learning models available and install TensorFlow using pip command. TensorFlow helps in Data augmentation before it begins model training. It is also used to make algorithms prediction efficiency perfect, and then download pre-trained image net weights.

2.1.4 *YOLOv5*

In YOLOv5 is a one-stage object detection architecture. One-stage object detection architectures treat object detection as a regression problem. It estimates the class probability and the coordinates of the bounding box that will contain the object in a single step on the input image. Like other one stage object detection architectures (SSD, YOLOv3, YOLOv4, RetinaNet, etc.), it consists of three basic parts: backbone, neck and head as show in Fig. 1. The head layer is also called the YOLO layer. The task of the model backbone is to reveal the distinctive features from the given image. CSPNet (Cross Stage Partial Networks) structure is used as the model backbone in YOLOv5. In large-size neural network backbones, the gradient information is copied in updating the layer weights and most of them have to be learned over and over again. This is a situation that negatively affects model size and inference speed. In the CSPNet approach, the feature map in the base layer is divided into two, some of them reach the transition layer through the dense block, and the other part is directly combined with the transition layer. In this way, not only the model size is reduced, but also the inference speed increases [11].

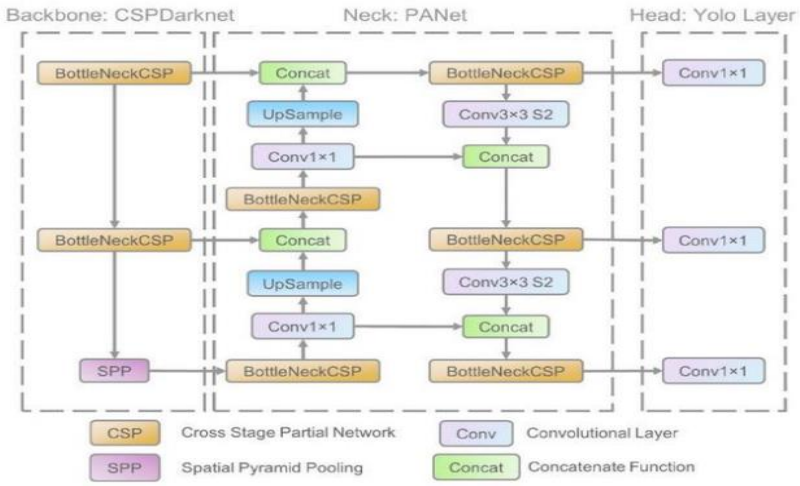


Figure 1. YOLOv5 Architecture (Source:https://www.ijans.org/index.php/ijans/article/view/550)

2.1.5 *Pytorch*

Pytorch is an open-source ML framework based on the Python programming language and the Torch library. Torch is an open-source ML library used for creating deep neural networks and is written in the Lua scripting language. It's one of the preferred platforms for deep learning research. The framework is built to speed up the process between research prototyping and deployment.

2.1.6 OpenCV

OpenCV is a library of programming functions mainly aimed on real time computer vision. originally developed by Intel, it is later supported by Willow Garage then Itseez. The library is across-platform and free to use under the open-source BSD license.

2.1.7 Raspbian OS

Raspberry Pi OS is a free, open-source Debian Linux-based operating system engineered for use on Pi boards. Additionally, several ARM-based single-board computers also run Raspberry Pi OS. The first version, then known as Raspbian, debuted in 2013, and from 2015 onwards the Raspberry Pi Foundation offered it as an officially-sanctioned Pi distro. Developers Peter Green and Mike Thompson are responsible for creating Raspbian, initially an independent endeavor.

2.1.8 Vonage

Vonage is a cloud communications provider. it is a global cloud communications platform that offers developers a range of communication tools and APIs (Application Programming Interfaces) to include messaging, video, phone, and other communication features into their goods, services, or apps. When it changed its name to Vonage, company was earlier known as Nexmo.

2.2 *Hardware*

2.2.1 Raspberry pi4 model B

Raspberry Pi is the name of a series of single-board computers made by the Raspberry Pi Foundation, a UK charity that aims to educate people in computing and create easier access to computing education. it is a very cheap computer that runs Linux, but it also provides a set of 20GPIO (general purpose input/output) pins, allowing you to control electronic components for physical computing and explore the Internet of Things (IoT). The Raspberry Pi 4 Model B was released in June 2019 with a 1.5 GHz 64-bit quad core ARM Cortex-A72 processor, on-board 802.11ac Wi-Fi, Bluetooth 5, full gigabit Ethernet (throughput not limited), two USB 2.0 ports, two USB 3.0 ports, 1, 2, 4, or 8 GB of RAM, and dual-monitor support via a pair of micro-HDMI (HDMI Type D) ports for up to 4K resolution.

2.2.2 Micro SDCard

Micro SD cards are small removable memory cards in which data can be stored. They are a specific type of SD (Secure Digital) card. The SD card is a key part of the Raspberry Pi; it provides the initial storage for the Operating System and files.

Figure 2. Raspberry pi4 micro-controller.

2.2.3 Raspberry pi Camera

The Raspberry Pi Camera Board is a custom designed add-on module for Raspberry Pi hardware. It attaches to Raspberry Pi hardware through a custom CSI interface. The sensor has 5 megapixel native resolution in still capture mode. In video mode it supports capture resolutions up to 1080p at 30 frames per second.

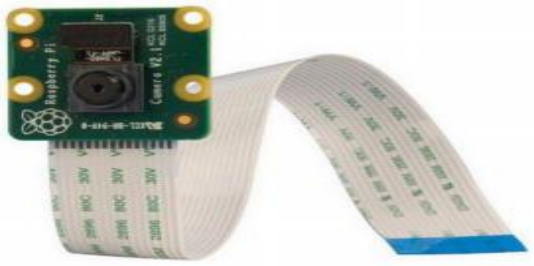


Figure 3. Raspberry pi camera module.

2.3 *Method*

The details of the implementation stages and tools are presented.

2.3.1 Block Diagram

Figure 4 shows the composition of the system.

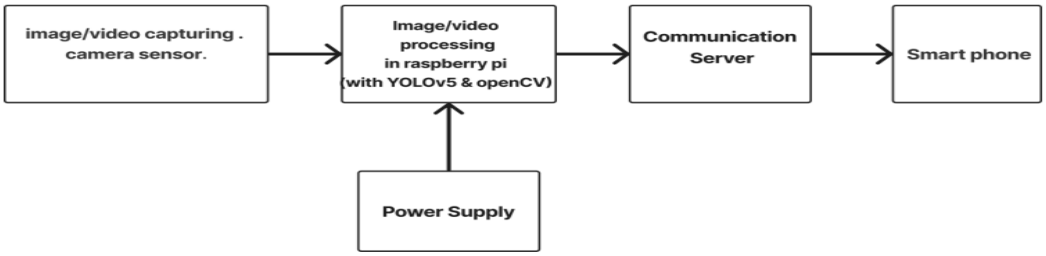


Figure 4. Block diagram

2.3.2 Data Collection

The ripe and unripe Sorghum (Bicolor L.) photos and videos used in this project were captured in various parts of the Moduganari Area. Borno State, Maiduguri Town. The pictures were taken in the morning, midday, and afternoon at varying distances and angles using the cameras on the Infinix S5 and Samsung Galaxy Tab A tablets, which have 720 x 1600 and 1920 x 1200pixel resolutions, respectively. With unequal settings such leaf occlusion, overlap occlusion, similar visual appearance to the backdrop image, dense targets, rear light, front light, and other natural situations, a total of 266 photographs and 2 films of length 98.4MB and 10MB, respectively, were acquired and stored in JPG and MP4 format. As was done by [12]. Some of the images that were taken under different settings are shown in Fig 5. To create 167 images for model training and testing paradigms, these images are randomly divided in Roboflow into 70% for the train set, 20% for the valid set, and 10% for the test set.

(a) (b) (c) (d)

Figure 5. Sample images: (a) Front light, (b) Back light, (c) Leaf occlusion, and (d) Overlap.

2.3.3 Data Preparation/Annotation

A total of 367 Sorghum heads were annotated and were divided into 177 for ripe sorghum and 190 for unripe sorghum respectively with an average annotation of 1.7 per image, average image size of 15.93mp and a median image ratio of 3456 X 4608 across two classes. Details on the dataset annotation are given in the Tab. 1 and 2. Using the Roboflow bounding box tool and the annotation files saved in the YOLOv5 text format, the ground truth bounding boxes within each image were manually constructed. The target class, coordinates, height, and width are required by the YOLOv5 text format. Fig. 6 shows the annotations made on images of ripe and unripe sorghum in purple and red, respectively.

(a) (b) (c)

Figure 6. Annotation of sorghum. (a) Ripe, (b) Unripe, (c) Ripe and Unripe.

Table 1. Splitting of the dataset into training, validation, and test sets.

Total number of images	Training	Validation	Test
266	186	53	27
100%	70%	20%	10%

Table 2. Distribution of annotations between the two classes used.

Total number of annotations	Ripe sorghum	Unripe sorghum
367	177	190
100%	48.25%	51.81%

2.3.4 Training of Dataset

The YOLOv5 model was cloned from the YOLOv5 repository on GitHub, and a custom annotated dataset from Roboflow was imported into Google Colab using the API for training. A total of 266 images with an annotation of 367 for both classes were utilized for training the YOLOv5s model, with a photo size set at 640 pixels, a batch size of 16, and a training duration using 150 epochs. The training process was conducted on Google Colab, leveraging the Tesla K80 GPU. Tensor Board was initiated after the training in order to provide graphical displays of the training progress and performance metrics, which will allow for a thorough visualization of the convergence and optimization of the YOLOv5 model.

2.3.5 Validation

For the validation of the trained model, we used a dataset of 53 images. The validation of all the trained models was performed on Google Colab using Tesla K80 107.7 GB GPU. As in the training phase, the image size of 640 px was used. Upon the completion of the training and validation phases, the file encapsulating the model weights that produced the most favorable validation outcomes during training, denoted as 'best.pt,' is obtained and saved locally on the computer in PyTorch file format. This 'best.pt' file is earmarked as the designated model for real-time inference in the on going project.

2.3.6 Deployment to Raspberry pi

For real-time inference on the Raspberry Pi 4, the optimized PyTorch model, represented by the 'best.pt' file, is easily included into the Python project. The procedure for deployment involves moving the

model file to the Raspberry Pi 4, where the Python code is carefully set up to load and run the 'best.pt' file properly. In addition, the inference pipeline is optimized to take advantage of the unique computing properties of the Raspberry Pi 4 hardware, guaranteeing peak performance in real-time inference.

2.3.7 Integration with Vonage

The project is connected with the Vonage communication platform to improve its usefulness. With the help of Vonage, SMS messaging can be conducted with ease. The system can instantly notify a chosen smartphone when particular objects of interest are spotted. The integration entails connecting to the Vonage API via the Python SDK. A notification event is triggered by the Python code when it detects predefined objects. Vonage is then used to send a personalized SMS alert to the specified phone number. The user will receive a brief but informative update via SMS message if pertinent information about the object is recognized. The project becomes much more versatile with this connection, which also adds a useful layer of real-time communication that keeps users updated about items or events that the system detects.

3.3.8 Flowchart

The flow chart of Figure 7 explains graphically the process of detecting and alerting if the sorghum is ripe. The process starts when a camera sensor is used to start taking images or videos in order to record visual information about the sorghum crop. The next stage involves a decision point where the system uses a computer vision algorithm to determine whether the sorghum is ripe or not. Upon checking the condition of the sorghum, the flow chart divides into two routes; a positive route where the sorghum is ripe or a negative route where the sorghum is unripe. When a ripe sorghum is detected, an immediate alert is sent to a smartphone via the Twilio communication platform. After the alert has been sent, the flowchart converges at a checkpoint where it checks if action has been taken. If the algorithm checks that the sorghum is unripe or not yet ripe, it continues without triggering any alert.

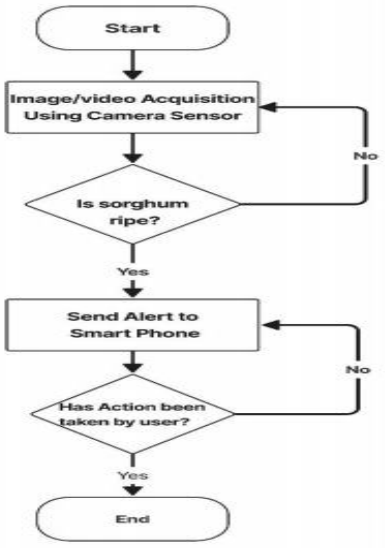


Figure 7. Flowchart of project

2.3.9 Convolutional Neural Network (CNN)

Convolutional neural network (CNN), also known as ConvNet, is a kind of artificial neural network (ANN) that has a deep feed-forward architecture and amazing generalizing ability in comparison to other networks with FC layers. It can learn highly abstracted features of objects, particularly spatial data, and can more effectively identify them. A deep CNN model is made up of a finite number of processing layers that are capable of learning many levels of abstraction for different input data features (such as picture features). Higher abstraction is learned and extracted by the deeper layers, whilst lesser abstraction is learned and

extracted by the initiatory layers for high level features. In CNN, the classification layer and the feature extraction layers collaborate to learn, which improves the model's output organization and increases its reliance on the extracted features. Today, CNN has become a tool for generating promising outcomes in a variety of computer vision based applications, including image classification, object detection, face detection, speech recognition, vehicle recognition, facial expression recognition, text recognition, and many more.

2.3.10 Evaluation Metrics

The Precision P, Recall R, F1 score and Mean average, mAP are the evaluation metrics utilized for the model's performance, which are expressed (1) through to (4). Precision is a metric that quantifies the accuracy of an object detection algorithm by measuring the proportion of correctly predicted positive instances out of all the instances predicted as positive. It indicates how reliable the algorithm is in correctly identifying objects [13]. Recall, also known as sensitivity or true positive rate, measures the ability of an object detection algorithm to find all the positive instances. It calculates the proportion of correctly predicted positive instances out of all the actual positive instances in the image or video [13]. The F1 Score is a harmonic mean of Precision, P and Recall, R. It provides a single metric that balances both precision and recall [14], giving an overall measure of the algorithm's performance. Mean Average Precision, mAP averages the Precision and Recall scores for each object class to determine the overall accuracy of the object detector. For a given query, q, we calculate its corresponding Average Precision AP, and then the mean of all these AP scores would give us a single number, called the mAP, which quantifies how good our model is at performing the query [15].

$$\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \tag{1}$$

$$\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP+FN}} \tag{2}$$

$$\mathrm{F1} = \frac{2\mathrm{RP}}{\mathrm{R+P}} \tag{3}$$

$$\mathrm{mAP} = \frac{\sum_{j=1}^{C}(\mathrm{AP_j})}{\mathrm{C}} \tag{4}$$

TP is true positive (correct detections), FN is false negative (missed detections), FP is false positive (incorrect detections), and P(R) denotes that P is a function of R. F1 is the trade-off between P and R, AP is the P−R curve of a single class, mAP is all the AP values averaged over different classes, C is the number of classes, j is the serial number [12].

3 RESULT AND DISCUSSION

The outcome of the training and results of the trained sorghum detection model when put to the real-time inference are presented. Also, how well the model adapts to different conditions and environments, showing its flexibility in practical farming scenarios is highlighted.

3.1 Trained Performance of the YOLOv5s Model

The YOLOv5s model has undergone a 150 epochs training program, and its performance has been thoroughly evaluated. A thorough analysis of the model's object detection performance is provided below, along with graphical interpretations of the results in Tab. 3. A strong correlation is found between the number of training iterations and the detection performance of the YOLOv5s model. The model's capacity to precisely detect and pinpoint ripe and unripe sorghum has improved over time as a consequence of repeated refining, leading to a significant improvement in overall performance metrics.

Table 3. Distribution of annotations between the two classes used.

MODEL	PRECISION (%)	RECALL (%)	F1 SCORE (%)	mAP (%)
Roboflow 3.0 Object Detection (Fast)	77.3	73.5	75.3	**78.1**
YOLOv5s	79.5	87.3	83.0	**87.0**

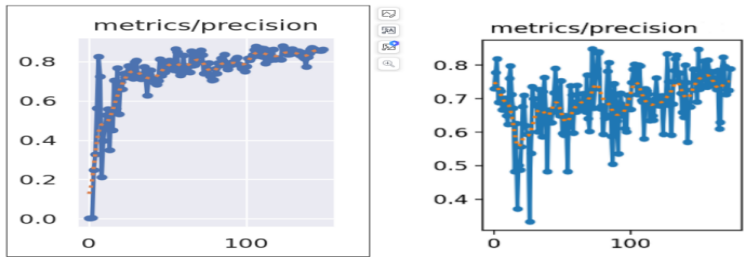
An interesting viewpoint on the development of important metrics over the training period is offered by the graphical analysis. Notably, the YOLOv5s model's precision is a noteworthy 0.795, demonstrating its effectiveness in reducing false positives. Concurrently, an excellent recall rate of 0.873 highlights the model's ability to accurately represent a significant percentage of true positive cases. The F1 score, which is the harmonic mean of memory and precision, achieves a respectable 0.83, indicating a fair trade-off between recall and precision. Furthermore, an 87% Mean Average Precision (mAP) score sums up the model's performance. This statistic provides a thorough indication of the overall detection accuracy of the model by measuring the precision-recall curve across different confidence thresholds. The mAP of 87% that was reached confirms the YOLOv5s model's effectiveness in producing strong and trustworthy findings. The obtained findings confirm the efficiency of the YOLOv5s model and highlight its applicability to many applications that require high-precision object detection capabilities. These results strengthen the model's usefulness in real-world scenarios where precise and effective object recognition is crucial by providing insightful information about the model's generalization capabilities across a variety of datasets and circumstances.

3.2 Trained Performance using Roboflow 3.0 Object Detection (Fast) model

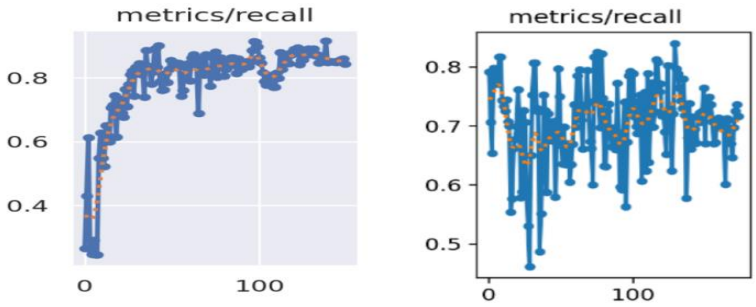
In addition to being trained using the primary YOLOv5s model for the project, the dataset is also, thoroughly trained using Roboflow 3.0 (Fast) model. The purposeful use of this dual-model training strategy was to improve the project's robustness and precision in a number of ways. The project's main building block is the YOLOv5s model, which is well-known for its accuracy and effectiveness in real-time object identification applications. But since it's necessary to vary training approaches in order to achieve the best possible model performance, Roboflow 3.0 (Fast) became an essential part of the training pipeline. The model's accuracy, capacity to identify true positives, and ability to strike a fair balance between precision and recall are all highlighted by these measurements. The outcomes demonstrate Roboflow 3.0 (Fast)'s applicability for tasks demanding accurate and dependable object recognition, it achieved a recall of 0.735, precision of 0.773., an F1 score of 0.753, and a global mAP of 78.1%.

3.3 Comparison of Results between YOLOv5 and Roboflow 3.0 Object Detection (Fast) models

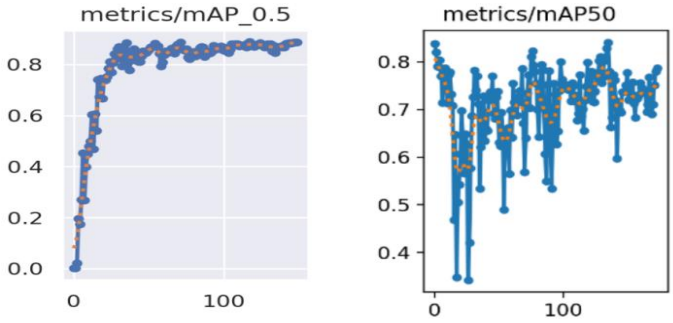
Based on the results obtained from the training, the YOLOv5s model achieved a Precision of 0.795, Recall of 0.873, F1 score of 0.83, a mAP of 87%, and the Roboflow 3.0 Object Detection (Fast) model achieved a Precision of 0.773, Recall of 0.735, F1 score of 0.753, a mAP of 78.1% respectively. The mAP serves as the primary metric for assessing the performance of the two models and it is observed and computed that the YOLOv5s model with a mAP of 87% outperformed the Roboflow 3.0 model, which has a mAP of 78.1%. Figure 8 shows graphic representation of matrics.



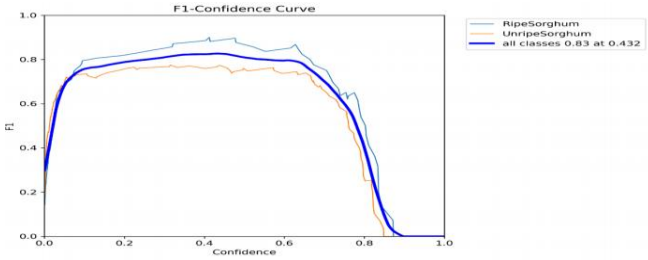
(a)



(b)



(c)



(d)

Figure 8. Graphical metric representation: (a) Graphs of Precision vs no. of Epochs, (b) Graphs of Recall vs no. of Epochs, (c) Graphs of mAP vs no. of Epochs, (d) Graph of F1 Score vs Confidence.

3.4 Pictorial Results of Inferences

Figure 9 show four test images. Figure 10, 11 and 12 illustrate how the trained model is able to infer both ripe and unripe sorghum concurrently in a single instance. It demonstrates the model is effectively trained to differentiate between ripe and unripe sorghum. Figure 10 illustrates the model's ability to effectively detect the unripe state of the sorghum, the model shows its ability to recognize the characteristics associated with the unripe class. Figure 11 illustrates the model's ability in detecting the ripe class of the sorghum exhibiting its training on the distinctive characteristics that set the ripe sorghum distinctive from the unripe. A more complex condition where both ripe and unripe sorghum are illustrated in a single instance is shown in Figure 12. The model effectively identifies and distinguishes between the two classes

in real time, giving each sorghum instance the proper label. This multi-class detection example highlights how well the model handles scenarios in which different object classes occur in the same visual context.

(a) (b)

(c) (d)

Figure 9. Test Images.

Figure 10. Detection of unripe Sorghum.

Figure 11. Detection of ripe Sorghum.

Figure 12. Detection of ripe and unripe Sorghum.

4 CONCLUSIONS

Using the Raspberry Pi and the Raspberry Camera Module, the model was smoothly incorporated into a real-time detection system for our practical application. The camera sensor maintained 30 frames per second (fps), which guarantees quick and seamless real-time detection capabilities. As a computational power house, the Raspberry Pi implemented the object detection model quickly and effectively. After a rigorous training and optimization process, the model showed off its ability to precisely recognize and locate ripe sorghum in every frame that the Raspberry Camera Module acquired. In conclusion, the aim and objective have been achieved with the system exhibiting an overall Precision of 79.5%, a Recall of 87.3%, F1 score of 83% and mAP of 87% using the YOLOv5 object detector model. The trained model was successfully deployed on the raspberry pi to achieve a real-time detection on the field. Assessing the YOLOv5 model's results generally, it can be concluded that ML can be used as a reliable and efficient method to address challenges in agriculture. In further study, more precise, accurate and dependable results can be obtained by expanding the size of the dataset and the complexity of the model. A significant expansion of the dataset would provide the algorithm access to a wider and more varied sample set, which would improve its capacity to generalize and identify subtle patterns in the yield, ripeness, and maturity of cereal crops. Concurrently, the application of a more sophisticated model, furnished with improved computational capacities and sophisticated algorithms, would greatly improve the accuracy and efficacy of the identification procedure.

REFERENCES

1. Du Toit, M. (2024). It Makes Sense to Include More Sorghum in Diets. https://www.farmersweekly.co.za/crops.

2. Ajeigbe, H.A., Angarawai, I.I., Akinseye, F., Inuwa, A.H., AbdulAzeez, T. and Vabi, M.B. (2020). Handbook on Improved Agronomic Practices for Sorghum Production in North East Nigeria. pp. 1-15.

3. Junos, M.H., Khairuddin, A.S.M., Thannirmalai, S. and Dahari, M. (2021). An Optimized YOLO-Based Object Detection Model for Crop Harvesting System. *IET Image Processing*, vol. 15, no. 9, pp. 2112–2125.

4. Matsuzaka, Y. and Yashiro, R. (2023). AI-Based Computer Vision Techniques and Expert Systems. *Artificial Intelligence*, vol. 4, no. 1, pp. 289–302.

5. Darwin, B., Dharmaraj, P., Prince, S., Popescu, D.E. and Hemanth, D.J. (2021). Recognition of Bloom/Yield in Crop Images Using Deep Learning Models for Smart Agriculture: A Review. *Agronomy*, vol. 11, no. 4, pp. 1-22.

6. Lai, J.W., Ramli, H.R., Ismail, L.I. and Hasan, W.Z.W. (2022). Real-Time Detection of Ripe Oil Palm Fresh Fruit Bunch Based on YOLOv4. *IEEE Access*, vol. 10, pp. 95763-95770.

7. Gonzalo-Martín, C., García-Pedrero, A. and Lillo-Saavedra, M. (2021). Improving Deep Learning Sorghum Head Detection Through Test Time Augmentation. *Computers and Electronics in Agriculture*, vol. 186, pp. 1-9.

8. Gong, B., Ergu, D., Cai, Y. and Ma, B. (2020). A Method for Wheat Head Detection Based on Yolov4. *Research Square*, pp. 1-24.

9. Wangli, H.A.O., Peiyan, Y.U., Fei, H.A.O., Meng, H.A.N., Jiwan, H.A.N., Weirong, S.U.N. and Fuzhou, I. (2021). Foxtail Millet Ear Detection Approach Based on YOLOv4 and Adaptive Anchor Box Adjustment. *Smart Agriculture*, vol. 3, no. 1, pp. 63-74.

10. Xu, D., Zhao, H., Lawal O. M., Lu, X., Ren, R. and Zhang, S. (2023). An Automatic Jujube Fruit Detection and Ripeness Inspection Method in the Natural Environment. *Agronomy*, vol. 13, no. 2, pp. 1-12.

11. Önler, E. (2021). Real Time Pest Detection using YOLOv5. *International Journal of Agricultural and Natural Sciences*, vol. 14, no. 3, pp. 232-246.

12. Lawal, O.M., Zhu, S. and Cheng, K. (2023). An Improved Yolov5s Model using Feature Concatenation with Attention Mechanism for Real-Time Fruit Detection and Counting. *Frontiers in Plant Science*, vol. 14, pp. 1-11.

13. Powers, D. (2011). Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation. *Journal of Machine Learning Technologies*, vol. 2, no.1, pp. 37-63.

14. Kanstrén, T. (2023). A Look at Precision, Recall, and F1-Score - Towards Data Science. https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec.

15. Tan, R.J. (2024). Mean Average Precision (mAP) Explained: Built In. https://builtin.com/articles/mean-average-precision.