Analysis, Design and Implementation of a Ripe Mango Detection Program in Burkina Faso

Moustapha Bikienga[1], Roland Manegaouindé Tougma[1,2(✉)], and Soumaïla Ouedraogo[1]

[1] Norbert ZONGO University, Avce Maurice Yameogo BP376, Koudougou, Burkina Faso
manegarodrol@gmail.com

[2] Joseph KI ZERBO University, CFX2 7R6 Ouagadougou, Burkina Faso
https://www.ujkz.bf/

Abstract. The analysis, concept, and implementation of a computer program capable of detecting ripe mangoes are at the core of this study. Traditional methods are often faced with calibration errors. Recently, deep learning has shown promising performance in visually guided agricultural applications. Faced with these constraints, it is necessary to establish an automatic system for robust and efficient detection of mangoes in orchards. In this study, a fast implementation system of a mango detector, distinguishing between ripe and unripe mangoes based on deep learning using the YOLOv5 algorithm, was developed. From a simple photo, the algorithm detects and counts the number of mangoes on a tree. This artificial intelligence system (deep neural network) was trained on a dataset of over 500 annotated mango images. Experimental results show that the algorithm achieves 98% precision, 98% recall, and an F1-score of 98%. This satisfactory precision in mango detection offers significant advantages in terms of efficiency and accuracy compared to traditional methods. However, it should be noted that our system has certain limitations. Nevertheless, our study demonstrates promising results in the field of ripe mango detection.

Keywords: Deep learning · YOLO algorithm · Mangoes · Performance metrics · Computer vision

1 Introduction

In BURKINA FASO, the mango is one of the six (6) so-called promising sectors identified as having strong potential for the diversification of exports and represents between 11 and 18% of West African mango production [5]. The mango constitutes about half of the national fruit production in volume [5]. It is also a very important economic, social and climatic issue in BURKINA FASO. However, the detection of fruit maturity is an essential task in agriculture and the food industry. However, traditional methods of detecting fruit maturity, such as

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2025
Published by Springer Nature Switzerland AG 2025. All Rights Reserved
A. Sere et al. (Eds.): AFRICOMM 2023, LNICST 588, pp. 181–189, 2025.
https://doi.org/10.1007/978-3-031-81573-7_15

visual observation and touch, are often subjective, slow and can lead to errors of judgment. By integrating advanced machine learning and computer vision techniques, it is possible to develop a more efficient and accurate automated and objective ripe mango detection system [5]. The objective of this study is to develop a robust and efficient ripe mango detection program capable of accurately locating ripe mangoes on trees and counting the total number using computer vision. Specifically, our study aims to ensure a better quality of the fruits offered on the market and to contribute to reducing post-harvest losses while optimizing logistics operations in the mango industry. This is why we focus on analyzing, designing and implementing a mango detection program using the You Only Look Once algorithm version 5 (YOLOv5). YOLOv5 is widely used for real-time object detection. Its lightweight architecture and high accuracy make it an ideal choice for our ripe mango detection application.

2 Similar Works

Object detection processes using artificial intelligence techniques have been practiced around the world. Several methods have been proposed in the past for the detection of different fruits, including:

R-CNN, Faster R-CNN, Fast R-CNN. Susoven jana et al. [4] proposed a deep learning method using faster R-CNN for the classification of a multi-fruit set, namely mango and pitaya fruits. The dataset used is a farmer's real catch at harvest time, which is divided into two (2) classes: mango and pitaya. In this research, the MobileNet model on the TensorFlow platform was used. The proposed method achieved good results. The accuracy score reached around 99%. There's also Ross Girshick, Microsoft Research [7] who proposes a Faster R-CNN algorithm for object detection. This algorithm trains the very deep VGG16 network 9 times faster than R-CNN, is 213 times faster at test time and achieves a higher mAP on PASCAL VOC 2012. Compared with SPPnet, Fast R-CNN forms VGG16 3 times faster, is tested 10 times faster and is more accurate. Following the same logic, Inkyu et al. [9] used a fruit detection approach using deep convolution neural networks. It applies the transfer learning method on previous work that led to the development of a state-of-the-art object detector called Faster Region-based CNN (Faster R-CNN). The performance of the implemented detector was evaluated over several fruits, and an F1 score of 83% was obtained, which is slightly higher than the results of previous work, which was 80%. The weakness of this work lies in the choice of detection method. Indeed, it should be noted that in an identical test environment, YOLOv5 performs better than the Faster R-CNN algorithm. This was demonstrated in a comparative study between the Faster R-CNN algorithm and YOLOv5 [11]. Beyond the choice, many results show a low detection score (83%).

Another Technique has been developed by Akshay Ramesh et al. [2] to identify the different ripening stages of climacteric fruits such as mango, using the

Arduino IDE which has the role of sending the detection result remotely via the GSM module. The process begins by capturing the image with a camera, which is then sent to MATLAB for further processing. MATLAB uses the HSV color space algorithm, which provides pixel information in the form of digital HSV values. This numerical value is compared with ideal sample values pre-stored in the database using MATLAB. Once the color of an image is known according to the different stages of ripeness of the mango, MATLAB sends the unique code of this color to the Arduino. If the mango is ripe, then we have quality detection; the model simply checks the brownish color threshold and makes a decision. The proposed method was tested on around 200 samples, of which 193 samples were accurately identified, representing a percentage of 96.5%. The weaknesses of this study are that the Arduino IDE can handle the tests, but will be limited to the actual deployment by its processing and storage capacity. Accuracy can also be improved.

Akshay Ramesh et al. [2] chose to identify the different ripening stages of climacteric fruits such as mango using the Arduino IDE, which has the role of sending the detection result remotely via the GSM module. The process begins by capturing the image with a camera, which is then sent to MATLAB for further processing. MATLAB uses the HSV color space algorithm, which provides pixel information in the form of digital HSV values. This numerical value is compared with ideal sample values pre-registered in the database using MATLAB. Once the color of an image is known according to the different stages of ripeness of the mango, MATLAB sends the unique code of this color to the Arduino. If the mango is ripe, quality is detected, and the model simply checks the brownish color threshold and makes a decision. The proposed method has been tested on around 200 samples, 193 of which have been identified with precision, i.e. a percentage of 96.5%. The weaknesses of this work are that the Arduino IDE can handle the tests, but will be limited to the actual deployment by its processing and storage capacity. Accuracy can also be improved.

3 Methodology

To achieve the announced downstream objective, we will address several key aspects. First, we will perform an in-depth analysis of the object detection features and specifications. Next, we will proceed to the design of the detection system, defining the steps and components necessary to achieve optimal performance. We will then implement our program using YOLOv5, adjusting parameters and performing experiments to improve the results. Finally, we will evaluate the performance of our program using metrics such as precision, recall, and F1 score.

3.1 Data Preparation

In the course of our study, we collected images of mangoes at several sites, notably in mango orchards in REO, the capital of the Sanguie Province in the

Central-West Region of BURKINA FASO. In addition, we also exploited several image banks available on the Roboflow platform, which offers free public image datasets of mangoes [3,6,8,10]. Images were also collected from sites such as Depositphotos and Alamy. After collecting the images, we annotated them using two tools: Roboflow's online tool and labelImg for local annotation. The annotations were made in the YOLO format, corresponding to the YOLO algorithm. We used two classes of objects for annotation: "ripe" for ripe mangoes and "unripe" for unripe mangoes. Once the images were correctly annotated, we performed pre-processing on them. We resized the images to 640×640 to match the model's input size, and we normalized the images to enable effective model training. Our dataset consists of 500 mango images, with 80% of the images used for training, 10% for validation, and the remaining 10% for testing. To organize the dataset and specify the paths to the training, testing, and validation image folders, we created a .yaml configuration file. This file defines the root directory of the dataset and the relative paths to the folders containing the training, testing, and validation images (Fig. 1).

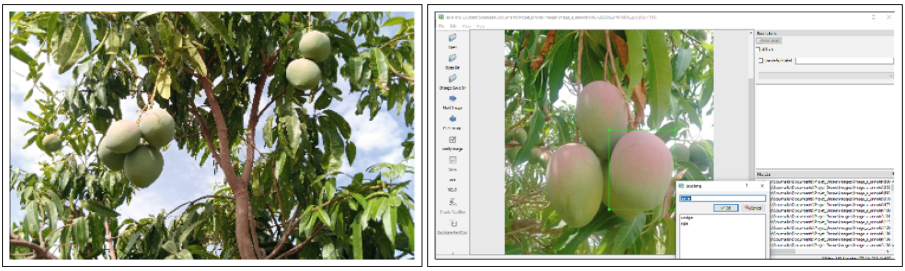

Fig. 1. On the left are the images taken with REO and on the right the annotations made with LabelImg

3.2 Model Selection and Training

Version 5 of the YOLO algorithm has several sub-versions, including 'n', 's', 'm', 'l' and 'x'. We decided to use the YOLOv5s version as it is designed to be faster for object detection in latency-critical applications. Once the model has been chosen, before moving on to training, it's necessary to set the hyperparameters that have an impact on model performance. These include learning rate, batch size set at 640, number of epochs and many other parameters. After fine-tuning the hyperparameters, the model is then trained with our custom dataset prepared on google colab thanks to free access to the GPU. Training is carried out by successive iterations on the training images, and the model adjusts its weights to improve detection performance. We evaluated the model's performance on validation data after training, using various measures including precision, recall and f1-score to ensure the model's ability to generalize to new data (Fig. 2).

Model	size (pixels)	mAP^{val} 0.5:0.95	mAP^{val} 0.5	Speed CPU b1 (ms)	Speed V100 b1 (ms)	Speed V100 b32 (ms)	params (M)	FLOPs @640 (B)
YOLOv5n	640	28.0	45.7	**45**	**6.3**	**0.6**	**1.9**	**4.5**
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7
YOLOv5n6	1280	36.0	54.4	153	8.1	2.1	3.2	4.6
YOLOv5s6	1280	44.8	63.7	385	8.2	3.6	12.6	16.8
YOLOv5m6	1280	51.3	69.3	887	11.1	6.8	35.7	50.0
YOLOv5l6	1280	53.7	71.3	1784	15.8	10.5	76.8	111.4
YOLOv5x6 + TTA	1280 1536	55.0 **55.8**	72.7 **72.7**	3136 -	26.2 -	19.4 -	140.7 -	209.8 -

Fig. 2. All the YOLOv5 models [1]

Accuracy. This metric measures the percentage of relevant detection results. This can be determined using the following equation: Accuracy = TP/(TP/FP) where TP (True Positive) represents the number of correctly detected objects in a given class. FP (False Positive) is when the model incorrectly identifies a region of the image as a positive object, when in reality there is no object of that class in that region.

Recall. This metric measures the percentage of total results correctly classified. It is determined using the following formula: R = TP/(TP+FN) where FN (False Negative) is when the model fails to detect a positive object in an image.

3.3 Model Deployment

The trained model was retrieved on my local machine by downloading a file summarizing the weights of the neurons, in other words the "educated" network. Thanks to this "educated" weight, we were able to perform mango detection locally on a personal machine. The advantage of this approach is that object detection can be performed more quickly and lightly on a personal machine, without the need for the heavy infrastructure required for initial model training. Using this pre-trained weight, we designed a visualization interface with the streamlit framework, enabling mango detection in an image, a video, or using the webcam.

3.4 Development Tools

The complete system requires different types of software, tools and frameworks for its implementation and deployment. Python was used as the development

language; Google-Colab Colab or Collaboratory was used to train our model, thanks to free access to the GPU, which speeds up training time. As for data preparation, Roboflow and LabelImg were used for images. Finally, Pytorch was integrated into YOLOv5 to implement our program, not forgetting Streamlit, which was used to design our interface for visualizing the detections made by the model (Fig. 3).

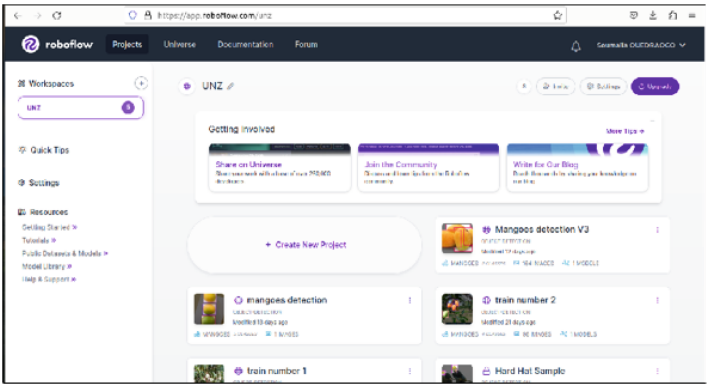

Fig. 3. We've created a project called UNZ on roboflow, which contains our datasets of mango images used to train our model.

Architecture of YOLO. Yolov5 improved the performance and its architecture, basednon high-level Object detection architecture (Fig. 4):

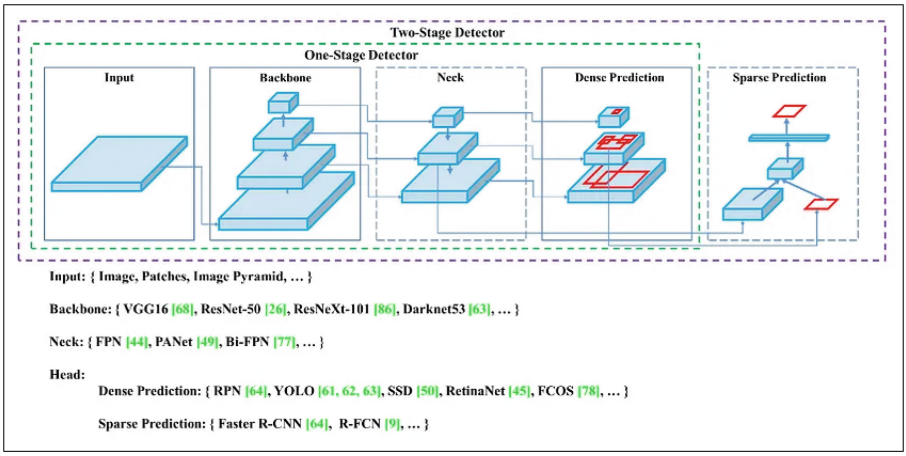

Fig. 4. Architecture of YOLO

General Object Detector will have a backbone for pre-training it and a head to predict classes and bounding boxes. The Backbones can be running on GPU or CPU platforms. The Head can be either one-stage (e.g., YOLO, SSD, RetinaNet) for Dense prediction or two-stage (e.g., Faster R-CNN) for the Sparse prediction object detector. Recent Object detectors have some layers (Neck) to collect feature maps, and it is between the backbone and the Head.

4 Results and Discussion

4.1 Results

Users can upload their own images or videos, submit them to the mango detection algorithm and visualize the results in a clear and comprehensible way. Thanks to Streamlit, we were able to quickly develop a high-performance web application, while offering an attractive and easy-to-use user interface for mango detection (Fig. 5).

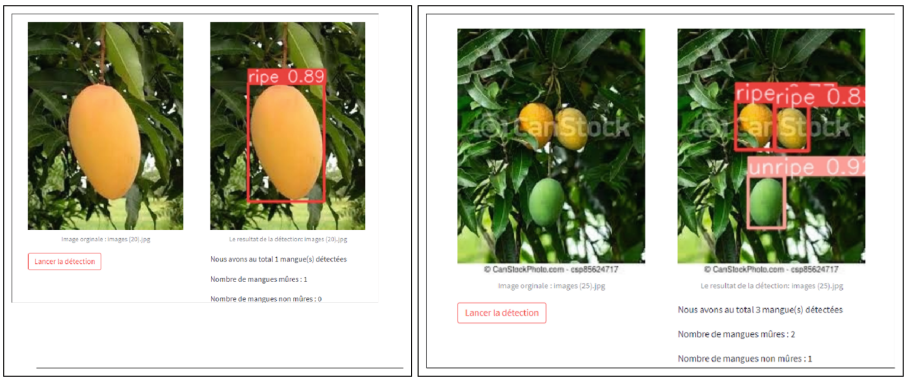

Fig. 5. The program detects ripe and unripe mangoes

Evaluating the algorithm's performance through the confusion matrix, we obtained a detection accuracy of 98% for ripe mangoes and 98% for unripe mangoes. At the end of our study, we found that our ripe mango detection algorithm has several strong points compared with existing work. Firstly, we obtained a detection accuracy of 98% for ripe mangoes, which is a promising result. However, we recognize that the detection accuracy for unripe mangoes is 98%, indicating potential room for improvement to minimize false detections. By focusing our efforts on fine-tuning the hyperparameters and increasing the size of the training dataset, we could improve the overall accuracy of our model (Fig. 6).

Evaluating the algorithm's performance through the confusion matrix, we obtained a detection accuracy of 98% for ripe mangoes and 98% for unripe

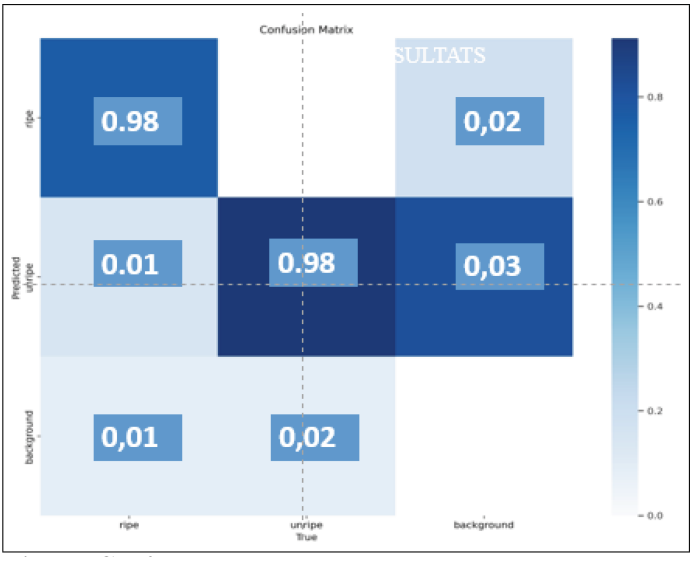

Fig. 6. Confusion matrix

mangoes. At the end of our study, we found that our ripe mango detection algorithm has several strong points compared with existing work. Firstly, we obtained a detection accuracy of 98% for ripe mangoes, which is a promising result. However, we recognize that the detection accuracy for unripe mangoes is 98%, indicating potential room for improvement to minimize false detections. By focusing our efforts on fine-tuning the hyperparameters and increasing the size of the training dataset, we could improve the overall accuracy of our model.

4.2 Discussion

One of the strengths of our algorithm is its ability to detect and classify ripe and unripe mangoes. This specificity is crucial for farmers, enabling them to determine the optimum harvesting time and ensure the quality of their produce. What's more, our model is adapted to the different types of mango varieties in BURKINA FASO, making it more versatile and applicable in different agricultural mango orchards. Another advantage of our model is that it does not require significant computing resources to apply detection. Thanks to the use of the YOLOv5s algorithm, our system is designed to be fast and efficient, enabling it to be deployed on devices with limited computing capacity. Compared with existing work, our approach offers a more flexible and efficient solution for the detection of ripe mangoes. By exploiting the capabilities of deep learning, our model can adapt to a variety of conditions, such as changes in brightness and variations in mango shapes and colors, and performance can be continually improved.

5 Conclusion and Future Projects

Mango detection can be carried out using the YOLOv5 algorithm. This algorithm performed well, producing a high accuracy value of 98% for ripe mangoes and 98% for unripe mangoes. This research can be improved by applying other detection algorithms to compare the best performances in mango detection. In future work, the program could perform detection based on ripening percentage, where it is up to growers to define their harvest percentage. The program could also specify mango quality. Our detection visualization interface could be made more interactive for growers, who could apply the detection or comment on it, so that we can adjust it to their needs, all automated by a robotic arm to make mango picking safer and faster.

References

1. Yolo v5 model architecture. https://iq.opengenus.org/yolov5/
2. Amrutkar, A.R., Jaisingpure, H.B., Bhujade, P.A.: Ripening and quality detection of mango using arduino (2018)
3. New-workspace b3mpu: mango dataset (2022). https://universe.roboflow.com/new-workspace-b3mpu/mango-yxaa7. Accesses 14 June 2023
4. Basri, H., Syarif, I., Sukaridhoto, S.: Faster r-cnn implementation method for multi-fruit detection using tensorflow platform. In: 2018 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC), pp. 337–340. IEEE (2018)
5. Boureima Barry, S.B.B.: Fiche sectorielle: Mangue du burkina faso (2023). https://www.apexb.bf/assets/pages/Fiche. Accés le 20 Mai 2023
6. New-workspace c1vsu: mango dataset dataset (2022). https://universe.roboflow.com/new-workspace-c1vsu/mango-dataset. Accessed 14 June 2023
7. Girshick, R.: Fast r-cnn. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448 (2015). https://doi.org/10.1109/ICCV.2015.169
8. KGP: Mango object detection dataset (2023). https://universe.roboflow.com/kgp-w7w3l/mango-object-detection-r1c0y. Accessed 14 June 2023
9. Sa, I., Ge, Z., Dayoub, F., Upcroft, B., Perez, T., McCool, C.: Deepfruits: a fruit detection system using deep neural networks. Sensors **16**(8), 1222 (2016)
10. Work: Mango2 dataset (2023). https://universe.roboflow.com/work-54ewq/mango2-7fjpw. Accessed 01 June 2023
11. Yusro, M.M., Ali, R., Hitam, M.S.: Comparison of faster r-cnn and yolov5 for overlapping objects recognition. Baghdad Sci. J. **20**(3), 0893–0893 (2022)