Detection of Freshness/Rottenness for Fruits and Vegetables

Group Members

1.	Dawit Getahun	UGR/9220/13
2.	Metsakal Zeleke	UGR/1027/13
3.	Tewodros Berhanu	UGR/9715/13
4.	Tinsae Shemalise	UGR/6075/13
5.	Yohannes Dessie	UGR/7612/13

Table of Contents

Introduction

In this paper, we will discuss how we built a computer vision system that detects the **Freshness/Rottenness of Fruits and Vegetables**. Classification of fruit and vegetable freshness plays an essential role in the food industry. Freshness is a fundamental measure of fruit and vegetable quality that directly affects the physical health and purchasing motivation of consumers. Most of the quality assurance and defect detections for fruits and vegetables are done by humans. But in a world with a growing emphasis on sustainable, efficient, and highly advanced technologies this is an inefficient and labor-intensive task. With our project, we hope to improve upon the existing system in terms of efficiency and accuracy.

We will first discuss how we collected, cleaned, and prepared our dataset. After that, we will discuss all the different models we tried out and their corresponding results. Finally, we will discuss the mobile app that we built utilizing the system we built, which is one of the many ways our model can be used in the food industry.

Following the **CRISP-DM** (Cross-Industry Standard Process for Data Mining) framework, we outline our approach. We start by outlining the business understanding and then detailing the data collection, and preparation phases. Then we will discuss modeling and evaluation. Finally, we present a mobile app as a practical application.

1. Business Understanding

The endeavor to develop a computer vision system for discerning the freshness/rottenness of fruits and vegetables stems from a critical need within the food industry. With consumer well-being and purchasing decisions closely tied to the perceived freshness of produce, efficient and accurate quality assessment is paramount. In response to the limitations of labor-intensive human inspections, our project aligns with the industry's growing emphasis on sustainability and technological advancement. By employing state-of-the-art computer vision techniques, we aim to impact fruit and vegetable quality assurance, providing a more streamlined and effective approach to freshness classification. The business implications of such an innovation extend beyond mere efficiency gains, influencing consumer trust, health standards, and the overall competitive landscape of the food industry. Understanding these business imperatives motivates our commitment to developing an advanced and practical solution.

1.1 Business Objective

The main objective is to build a system that can be used to improve food quality industry metrics. The business target of the system we are building is manufacturers and exporters in the food industry. We aim to help them reach quality standards by replacing repetitive work with an efficient computer vision system that would have been done otherwise by a human, which can be unreliable at times. Specifically, we aim to revolutionize the classification of fruit and vegetable freshness, offering a more efficient and accurate alternative to traditional, labor-intensive methods.

1.2 Situational Assessment

Requirement Assumptions, and Constraints:

- The system should provide accurate and comprehensible results. The quality of output is crucial because of the high importance of ensuring rotten/stale food is not classified as fresh.
- The system will only be put in place in real-world situations after rigorous testing and evaluation. The system will have to be licensed by the authority that oversees the subject to be able to be deployed in real-life scenarios for manufacturers.
- Availability of GPU (Graphics Processing Unit) for training is a constraint. Available free GPU access can be limited at times. Kaggle and Google provide limited access.
- The system is best suited for an IOT(Internet of Things) device, but due to the lack of resources, we have opted for a flutter mobile application.

Risks and Contingencies:

- One significant risk associated with the computer vision system is the potential misclassification of rotten or stale food items as fresh. This misclassification could lead to inaccurate quality assessments, impacting consumer health, trust, and overall food safety.
- If a specific type of misclassification is consistent there should be a way for manufacturers to provide feedback to developers.

Costs and Benefits:

- The project is developed with zero cost currently, as all the resources were acquired freely without payment. In the future, there will be licensing and deployment costs to get the system to the target manufacturers. And, future benefits after licensing and deployment will be discussed with manufacturers.

1.3 Data Mining Goals

- Identify and recognize patterns in the dataset related to visual cues that distinguish between fresh and rotten or stale fruits and vegetables.
- Extract relevant features from the images, such as color, texture, and shape, that contribute to the accurate classification of freshness levels.
- Implement effective data cleaning techniques to handle noisy or irrelevant data, ensuring the dataset is well-prepared for training and testing machine learning models.
- Cross-Dataset Generalization: Assess the model's ability to generalize across different datasets, considering variations in image quality, lighting conditions, and fruit/vegetable types.
- Evaluate the model appropriately with different evaluation techniques
  Develop an interpretable model, allowing others to understand the decision-making process
- Optimize the model for real-time inference, considering the potential application in live scenarios, such as quality control in food processing units.

1.4 Project plan

- Project Initiation (Week 1): Define project scope and objectives, identify stakeholders, and establish project team roles and responsibilities
- Business Understanding (Week 2-3): Finalize business objectives and success criteria, perform situational assessment and identify risks and contingencies
- Data Collection and Preparation (Week 4): Search and collect open-source datasets, combine and preprocess datasets, and Data Cleaning: remove watermarked and irrelevant Images
- Model Development (Week 5-6): experiment with different models, train models, validate, and optimize hyperparameters, and perform a comparative analysis of model performance
- Results Analysis (Week 7): analyze results from different models, evaluate, and assess model confusion matrices,
- Mobile App Development (Week 8-10): design and develop the mobile app.
- Documentation and Reporting (Week 11): document model architectures, and prepare a comprehensive project report
- Licensing and Deployment (Future): get proper licensing from government authorities and approach manufacturers
- Continuous Improvement (should be ongoing and not stop): training and knowledge transfer with end users, establish feedback mechanisms, monitor

model performance in real-world scenarios, implement updates and enhancements, and address emerging challenges and user requirements.

2. Data Understanding

Data Collection, Description, and Quality Assessment:

It is a known fact that for any deep learning project, the amount and quality of data are paramount to the performance of the resulting model(system). To build our dataset what first came to mind was to collect the dataset ourselves, but this would require us to dedicate a lot of time and effort. But instead, we looked for open-source datasets from the internet.

After some searching, we found the following datasets:

- Fresh and Rotten Classification by SWOYAM SIDDHARTHNAYAX
- Fruits and Vegetables dataset by MUKHRIDDIN MUKHIDDINOV
- Classificação de Frutas e Vegetais by PROJETO FV

Between the above datasets, we had about 10 fruit and vegetable types. We got to 10 because the datasets had overlapping fruit/vegetable data, so we combined them into one large dataset by either keeping only one of each overlapping fruit/vegetable class or by selectively merging the images into a new fruit/vegetable class. After this process, we had a total of 11986 images belonging to 20 classes, i.e. fresh and rotten for each of the 20 fruit/vegetables.

We were able to utilize this initial collection for our classification task. However, as our project moved away from being a classification task to being an object detection task, the requirement for our dataset changed. The images needed to be annotated with the description of the borders of the box the fruit or vegetable is bounded by within each image. Instead of annotating every single image ourselves (thousands we were able to collect), we were able to find annotated data from the Roboflow website (which we will discuss in the model section later).

3. Data Preparation

Data cleaning

Now that we have compiled a big enough dataset we can start to clean the resulting dataset. Most of the images were good but some had watermarks and stock images that don't reflect what the fruits/vegetables look like. Examples:

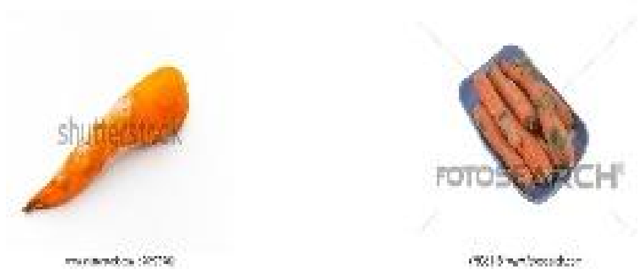


Fig. 1: Examples of faulty data

After removing such images, we finished the data cleaning step. We could have resized every image to the same dimensions but it can be done when the data is loaded and trained. That is the best practice while building such systems because putting a resize layer in the network will make our model compatible with any image size plus we won’t need to waste time resizing thousands of images while it can be done during training without that much cost on training time and resources.

Data augmentation
Data augmentation is a technique employed to enhance the training set by generating modified copies of a dataset through the manipulation of existing data. It encompasses a variety of methods aimed at improving both the attributes and size of training datasets. This approach proves particularly beneficial in the context of images, where it serves to safeguard against overfitting by preventing the model from learning "irrelevant" information, such as object position and orientation. In our specific case, we leveraged data augmentation as a preventive measure against overfitting.

4. Modeling and Evaluation

For this project, we have explored different deep-learning techniques and algorithms. For each of the modeling techniques we used, we provide an explanation and evaluation of the results as to why some of them were not selected.

It is well known that Deep Learning, specifically Convolutional Neural Networks(CNNs) are the best tools to help build different computer vision applications and services. So naturally we decided to use CNNs to help us build our computer vision system. Currently, there are a lot of CNN architectures we could have chosen from. Besides, libraries like TensorFlow make it easier to build any architecture of our design. Therefore from this point onward, we will discuss the different models we tried out and their performance results.

4.1 An architecture we built ourselves

We first tried to build our model, mostly to understand how it is done and if needed, to be able to customize the other built-in models later on.

As you can see, in the ***'training_1.ipynb'*** notebook we have tried to use Max Pooling, and Batch Normalization to combat overfitting and help our model generalize. The performance of the model was as follows:

- **Accuracy: 81%**

We have visualized the training and validation loss changes over the epochs, and it shows that our model didn't overfit and generalized well:

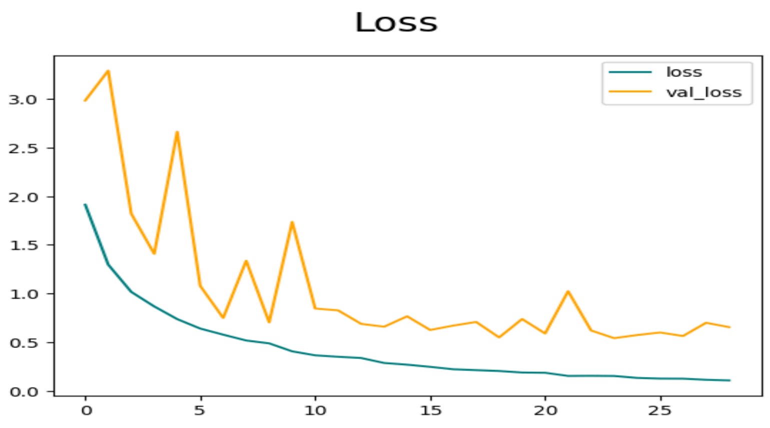


Fig. 2: Training and Validation Loss progression over the epochs

Even if the results we got for the model were good we didn't see those as the result we desired (the model failed to generalize outside of the sample). We saw it as an indicator that other well-designed existing models have so we continued to test out other architectures below.

4.2. VGG-16 architecture

VGG16 is a deep convolutional neural network architecture that was introduced by the Visual Geometry Group at the University of Oxford. It's characterized by its depth, featuring 16 layers which are depicted in the figure below.

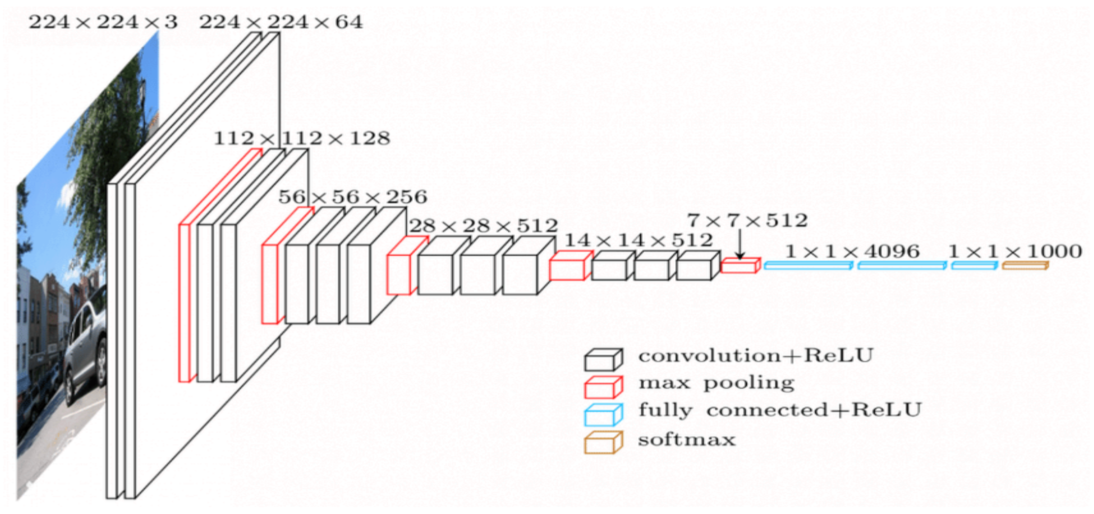


Fig. 3: Architecture of VGG-16

Tensorflow provides a built-in implementation of this architecture so all we had to do was feed our data to it. You can find the code for this experiment in the ***'training_2.ipynb'*** notebook. After training the performance of our model was:

- **Accuracy: 89%**

We have visualized the training and validation loss changes over the epochs, and it shows that our model didn't overfit and generalized well within the data sample we compiled:

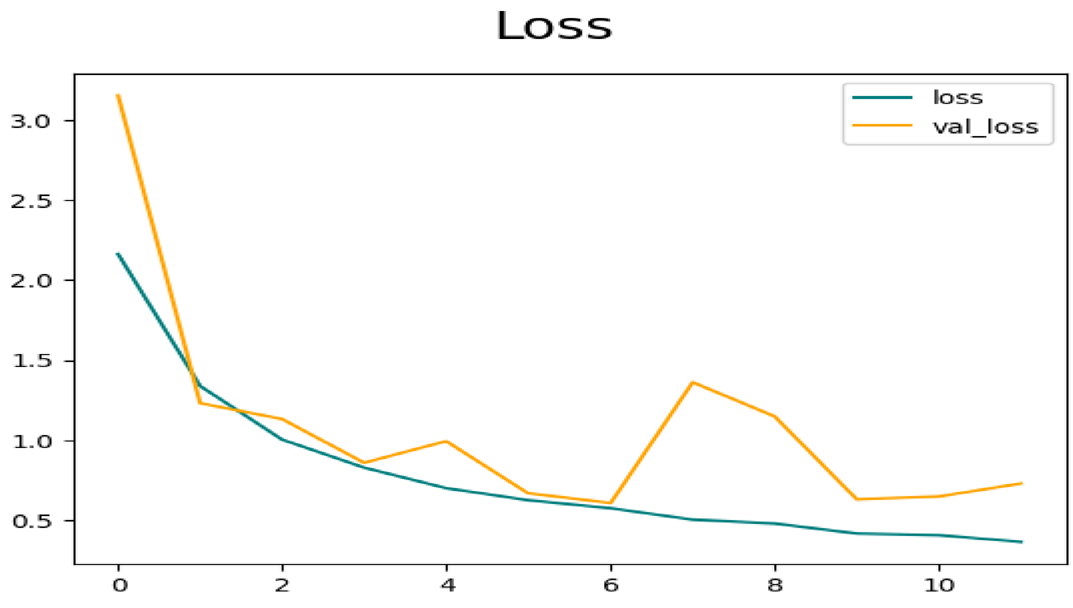


Fig. 4: Training and Validation Loss progression over the epochs

But when we decided to test the model on pictures taken from our phones, surprisingly the results weren’t really good. We can see from the validation and training loss that there isn’t any apparent overfitting problem.

So the next thing we tried was to utilize early stopping but nothing changed, our training and testing accuracy stayed pretty much the same. Yet when we tested the model with the photos we took (i.e. real-life data), there wasn’t much improvement.

4.3. Resnet - 18 & 50

ResNet 18 and 50 are two types of convolutional neural networks (CNNs) that use residual blocks to learn deep features from images. Residual blocks are composed of two or three convolutional layers that have a shortcut connection that skips over some layers. This helps to avoid the vanishing gradient problem and improve the accuracy of the network.

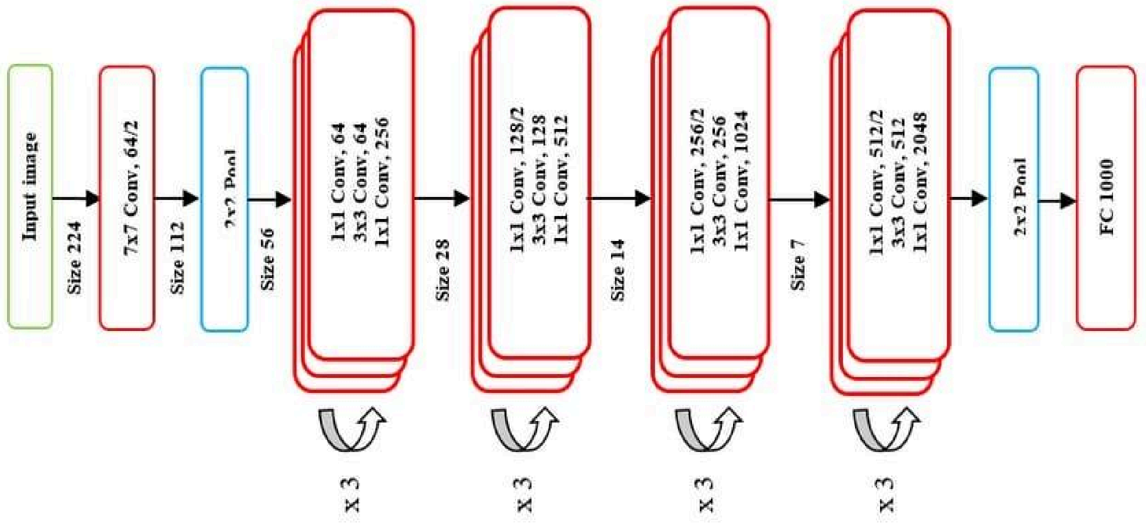


Fig. 5: Architecture of ResNet

Again Tensorflow provides built-in implementation of this architecture and the experiment can be found in the ***‘training_3.ipynb’***. The results we got for this architecture were more or less the same. So we felt the need to look for a different strategy, and we opted for building an object detection model rather than an image classification one.

4.4. YOLOv8

YOLO ("You Only Look Once"), a real-time object detection model known for its speed and accuracy, unlike other methods that scan images repeatedly, it analyzes the entire image in one go using a single neural network. This network predicts bounding boxes and probabilities for objects within the image, making it efficient for tasks like self-driving cars and video surveillance. And, we opted to use the latest YOLO version, YOLOv8 with faster inference, better precision, and a simpler design for our task.

One challenge was that we could not use the thousands of data we had collected so far to train our model because YOLO requires the bounding boxes for each object in each image to be annotated in a specific format. Luckily we were able to find a YOLOv8 style annotated data from the Roboflow website.

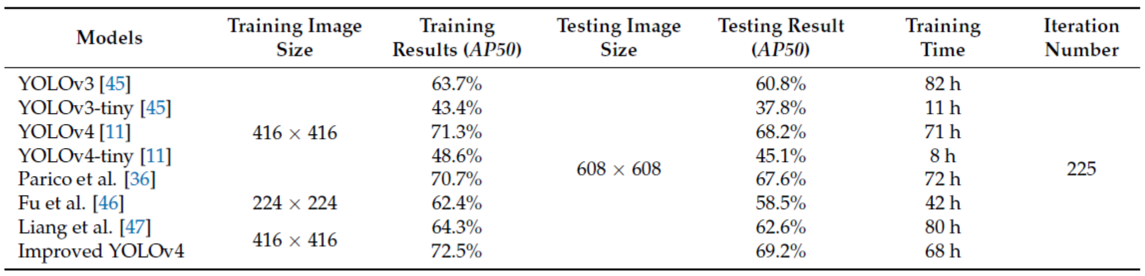
- Dataset source:- found compiled and annotated here.
- Scope of data:- Focus on a small-scale implementation, concentrating on specific varieties of fruits and vegetables commonly found in local markets. (Oranges, Mangoes, Bananas, Apples, Cucumbers, Carrots, Potatoes)

Reasons for switching to YOLOv8 rather than the models we have been trying earlier are mainly: bad performance on unseen, real-life data, and poor generalizing capabilities. So basically we found ourselves at a crossroads shifting the scope of this project to include live object detection. And because of its single-shot detection capability, we found YOLO to be more suitable than other techniques like the sliding-window method.

Training with YOLO:
- Input: The dataset has to be prepared in a unique format. A text file containing the class of the object, the coordinates of the center of the object, and its dimensions (height and width)
- Output: A model capable of detecting an object from any image, labeling it based on the training data, and drawing its coordinates.

Previous Work with YOLOv4

In October 2022, the Department of Computer Engineering at Gachon University, South Korea, undertook a comprehensive exploration of the limitations associated with YOLOv4 in the context of food quality control.



Models	Training Image Size	Training Results (*AP50*)	Testing Image Size	Testing Result (*AP50*)	Training Time	Iteration Number
YOLOv3 [45]	416 × 416	63.7%	608 × 608	60.8%	82 h	225
YOLOv3-tiny [45]		43.4%		37.8%	11 h	
YOLOv4 [11]		71.3%		68.2%	71 h	
YOLOv4-tiny [11]		48.6%		45.1%	8 h	
Parico et al. [36]		70.7%		67.6%	72 h	
Fu et al. [46]	224 × 224	62.4%		58.5%	42 h	
Liang et al. [47]	416 × 416	64.3%		62.6%	80 h	
Improved YOLOv4		72.5%		69.2%	68 h	

Fig 6. Figure showing the results obtained from the researchers from Korea in table format

Our team faced limitations in terms of both time and resources during the experimentation with YOLOv8. The use of Google's free GPU access, with a maximum allowance of 12 hours per day, posed a constraint. Additionally, frequent interruptions due to internet connectivity issues curtailed our training sessions, limiting us to only 50 epochs compared to the extensive 225 epochs undertaken by the Korean researchers. Furthermore, our uninterrupted training time was a mere 2 hours, in stark contrast to the 80 hours achieved by the researchers mentioned above.

Results with YOLOv8

Nonetheless, the results we managed to achieve were promising. The mean average precision at 50% Intersection over Union (mAP50), a critical metric measuring the overlap between predicted and ground truth bounding boxes, exhibited encouraging signs. Our training mAP50 reached an impressive 88.2%, showcasing the effectiveness of YOLOv8. During testing, the model maintained a strong mAP50 score of 63.6%.

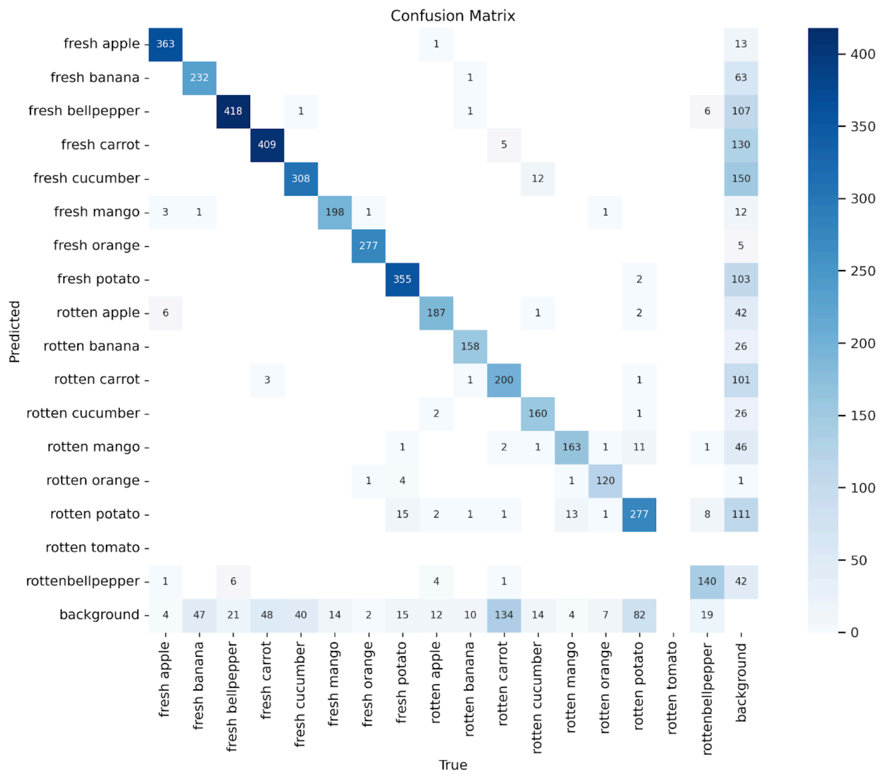


Fig 7. Figure showing the confusion matrix obtained after the first 25 epochs.

The confusion matrix for our model shows that, while having good accuracy in accurately identifying fresh products, it also excels in minimizing false positives for fresh products. However, challenges arise when dealing with rotten or stale products, where the model tends to struggle to accurately distinguish them.

5. Development

Now that we are done with building the model, we would like to utilize it to showcase how one of its applications would look like. To do this we created a simple mobile app that allows you to do object detection from your live camera feed, take pictures, or choose a picture from your gallery and determine whether the picture contains fruits/images and also tell if they are rotten or fresh.

Since our final model was built not just for image classification but for object detection, we can detect different types of fruits/images within an image and individually determine whether they are fresh or rotten. The only thing to take note of

is that the images we take should be a little bit up close and have good lighting and quality. The source code for the app can be found here. Currently, the app is only for Androids, because one of the key dependencies used for model inference (the flutter package flutter_vision) has not been made available for IOS devices yet.
Below are screenshots of the app layout:

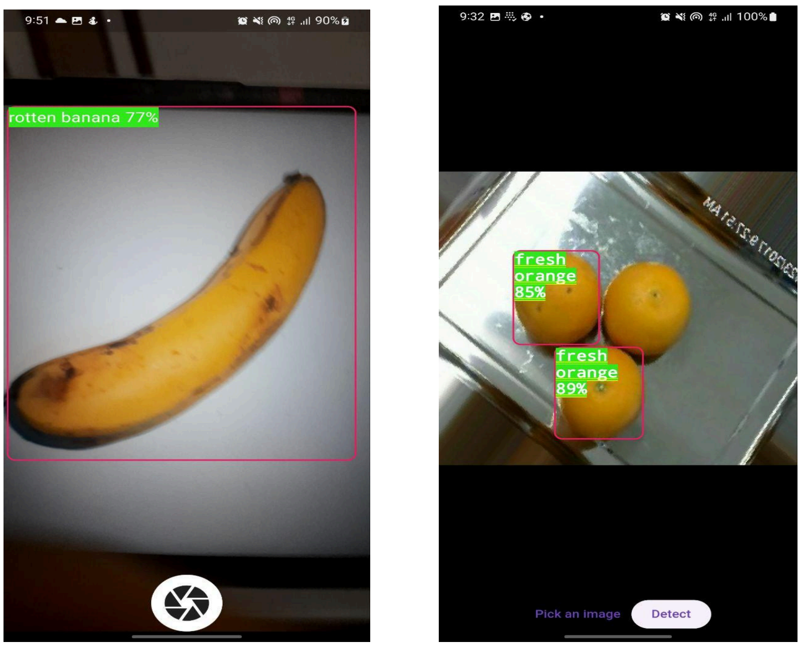


Fig. 8: Screenshots of our app. On the left is a screen that allows us to detect fruits/vegetables from the camera feed, and on the right from the gallery. The boxes are drawn on the detected fruits or vegetables with their respective confidence score

6. Conclusion

The findings suggest that implementing our proposed system, built on the foundation of YOLOv8, has the potential to enhance existing food quality control measures in Ethiopia, provided adequate resources are available. Moreover, the system can be extended to include other locally-grown and widely consumed products such as onions, tomatoes, papayas, etc. Furthermore, the applicability of this system extends to other agricultural products, such as coffee, a significant contributor to the country's exports, and teff.

This comparative analysis underscores the adaptability and potential impact of YOLOv8 in advancing object detection capabilities for agricultural and food quality control applications.

7. References

1. Mukhiddinov, M.; Muminov, A.; Cho, J. Improved Classification Approach for Fruits and Vegetables Freshness Based on Deep Learning. *Sensors* **2022**, *22*, 8192. https://doi.org/10.3390/s22218192

2. Bhargava, A.; Bansal, A. Fruits and vegetables quality evaluation using computer vision: A review. *J. King Saud Univ. Comput. Inf. Sci.* **2021**, *33*, 243–257. [link]