École Nationale Supérieure d'Informatique et d'Analyse des Systèmes - RABAT

Filière : Ingénierie en Data Science and IoT

Multi-Task Computer Vision System for the Classification and Detection of Fruits and Vegetables

Prepared by :

EL Yassini Safouat

Supervised by :

Mr. Moudni Alae

Jury members :

Mr. Ait skourt
Mr. Rachad

Year 2024-2025

Acknowledgments

I would like to particularly thank Allah the Almighty, as this project would never have been realized without His blessing.

I thank my dear parents for their support, patience, sacrifice, and love. You deserve all praise, for it is thanks to you that I have become who I am today. I hope to live up to your expectations. May God protect and bless you.

I also wish to express my gratitude to all those who have supported me with their presence, availability, and advice in completing this project. I would especially like to thank Mr. Alae Moudni, who honored me by being my mentor. I deeply appreciate their continuous encouragement and for always being there to listen, help, and guide me. Their wisdom and valuable advice have been of great help to me, as well as their support throughout my internship.

Abstract

With the increasing demand for automated solutions in agriculture and food industries, accurate recognition systems play a critical role in enhancing processing efficiency and quality control. This project presents a computer vision system optimized for the classification of fruits and vegetables, with supplementary object detection for images containing multiple instances of the same produce item. The primary focus is on high-accuracy classification, addressing the need for reliable identification of various produce types in real-world industrial applications.

To build a robust model, the dataset was carefully curated and preprocessed, involving extensive labeling, augmentation, and balancing to ensure that each fruit and vegetable class was well-represented. A separate detection module was incorporated to assist with images containing repeated objects, ensuring accurate identification without diminishing classification precision. Using deep learning techniques, the model achieved excellent classification accuracy across all classes and handled multi-object scenarios effectively when necessary.

Deployed via a FastAPI interface for real-time inference, the final system demonstrates both scalability and adaptability for integration into production environments. This project showcases an efficient solution tailored to the needs of the agriculture and food processing sectors, meeting industry standards for precision and speed in automated produce recognition.

Résumé

Avec la demande croissante de solutions automatisées dans les secteurs de l'agriculture et de l'alimentation, des systèmes de reconnaissance précis jouent un rôle essentiel dans l'amélioration de l'efficacité des processus et du contrôle de la qualité. Ce projet présente un système de vision par ordinateur optimisé pour la classification des fruits et légumes, avec une détection d'objets complémentaire pour les images contenant plusieurs instances du même produit. L'objectif principal est d'assurer une classification de haute précision, répondant au besoin d'une identification fiable des différents types de produits dans des applications industrielles réelles.

Pour construire un modèle robuste, l'ensemble de données a été soigneusement élaboré et prétraité, impliquant un étiquetage, une augmentation et un équilibrage approfondis afin de garantir que chaque classe de fruits et légumes soit bien représentée. Un module de détection séparé a été intégré pour aider avec les images contenant des objets répétés, assurant une identification précise sans diminuer la précision de la classification. En utilisant des techniques d'apprentissage profond, le modèle a atteint une excellente précision de classification dans toutes les classes et a géré efficacement les scénarios multi-objets lorsque cela était nécessaire.

Déployé via une interface FastAPI pour une inférence en temps réel, le système final démontre à la fois une scalabilité et une adaptabilité pour son intégration dans des environnements de production. Ce projet met en avant une solution efficace adaptée aux besoins des secteurs de l'agriculture et de la transformation alimentaire, répondant aux normes de l'industrie en matière de précision et de rapidité dans la reconnaissance automatisée des produits.

Contents

List of Figures

Introduction

The accurate classification of multiple instances of the same object within a single image is a complex yet increasingly relevant task in computer vision, particularly within agriculture and food processing.
This project focuses on developing a multi-task computer vision system to classify and detect multiple replicas of specific fruits and vegetables within a single image. Designed to meet the needs of industries where precise identification and counting of produce items are essential for quality control and inventory management, this system aims to identify repeated objects accurately, even in scenarios with high object density and variability.
By leveraging advanced deep learning models, including YOLO and other detection frameworks, the project integrates state-of-the-art techniques in object recognition and data augmentation to manage dataset imbalance and enhance model performance. This report outlines the methodologies used, model architectures applied, and the effectiveness of this system in addressing the unique challenge of detecting and classifying multiple replicas of the same item within complex image environments.

Chapter 1

Project Overview

1.1 Introduction

The agricultural and food industries are increasingly adopting automation to enhance operational efficiency and quality control. A key focus area is the development of advanced recognition systems capable of accurately identifying various fruits and vegetables. These systems play a vital role in streamlining food processing, inventory management, and quality assurance, ultimately reducing waste and increasing profitability.

In this context, the **Multi-Task Computer Vision System for the Classification and Detection of Fruits and Vegetables** was developed during my internship at **Société Tangéroise de Maintenance (STM) Group**. Established in 1998, STM is a leader in IT infrastructure integration and information systems, with over 22 years of experience, 13 strategic partnerships, and a portfolio of more than 5,800 satisfied clients. The company offers a wide range of services, including custom software development, hardware solutions, and cutting-edge technologies like computer vision and artificial intelligence.

The primary objective of this project is to create an automated system that prioritizes the accurate classification of fruits and vegetables. It also incorporates detection capabilities to handle scenarios involving multiple instances of the same item within a single image, addressing the growing demands of the agricultural sector for reliable identification solutions.

1.2 Presentation of the Host Organization

1.2.1 Introduction

Founded on July 23, 1998, **Société Tangéroise de Maintenance (STM) Group** has established itself as a leader in the design, integration, and operation of innovative IT solutions. With over 22 years of experience, STM has developed cutting-edge expertise that addresses the growing needs of the Moroccan and African markets in the field of new technologies.

1.2.2 Experience and Reach

With a robust portfolio of 5,800 satisfied clients and 13 strategic partnerships, STM has become a reference point for IT infrastructure integration and information systems. The company has built a solid market presence through its commitment to delivering high-quality services and solutions tailored to the unique requirements of its diverse clientele.

1.2.3 Expert Team

STM is powered by a dedicated team of over 50 certified engineers and technicians, specializing in various domains, including engineering, project management, technical support, and system integration. Their expertise allows STM to offer a comprehensive range of services, including:

— **IT Infrastructure Integration:** Seamlessly integrating IT systems to enhance operational efficiency.
— **Computer Hardware and Office Automation:** Providing state-of-the-art hardware solutions for modern workplaces.
— **Telephony and Communication Solutions:** Developing telecommunication systems to facilitate effective communication.
— **Human Resources Management and Accounting Solutions:** Implementing systems that streamline HR and financial operations.

— **Artificial Intelligence and Computer Vision:** Leveraging advanced technologies to develop innovative solutions that enhance business processes and decision-making.

1.2.4 Comprehensive Service Offerings

STM's expertise extends beyond standard IT services, covering critical infrastructure components such as pre-wiring, telecom networks, data centers, videoconferencing, telemonitoring, and fire detection systems. The company also offers preventive and curative maintenance contracts, IT outsourcing, equipment rentals, and support for various workstations, including printers and copiers.

1.2.5 Custom Engineering Solutions

In addition to its wide-ranging services, STM specializes in custom engineering solutions. Their highly qualified team is adept at developing both software and hardware solutions that meet the specific needs of clients, ensuring tailored support and innovation.

1.2.6 Innovation in Technology

STM is at the forefront of technological advancement, offering software engineering services to develop high-quality, scalable, and reliable software solutions. The company also specializes in computer vision and artificial intelligence technologies, providing state-of-the-art solutions that meet the demands of modern business.

1.2.7 Conclusion

With a commitment to excellence and a focus on customer satisfaction, Société Tangéroise de Maintenance (STM) Group continues to set benchmarks in the IT industry. The company's dedication to innovation, quality, and tailored solutions has earned it the trust of major economic players in the region, solidifying its position as a leader in IT infrastructure and information systems.

1.3 Problem Statement and Proposed Solution

1.3.1 Problem Statement

The project was proposed by an agricultural factory looking to improve its operational efficiency and streamline its processes. This facility is engaged in the processing and distribution of various fruits and vegetables, and they have recognized the limitations of traditional manual classification methods. Currently, workers are responsible for visually inspecting and annotating each batch of produce as it moves along the production line. This process is not only time-consuming but also prone to human error, which can lead to misclassification, reduced quality control, and ultimately increased waste.

1.3.2 Proposed Solution:

To address these challenges, the factory aims to integrate a sophisticated camera system into the final stage of their production line. This system will leverage advanced computer vision technology to automatically classify each assembly of fruits and vegetables in real-time as they pass through the inspection area. By utilizing automated recognition capabilities, the factory seeks to eliminate the need for manual intervention, significantly speeding up the classification process while enhancing accuracy. This shift to an automated solution will not only improve the efficiency of the production line but also ensure consistent quality in the classification of produce, aligning with the factory's goals of minimizing waste and maximizing profitability.

In addition to improving operational efficiency, this project represents a significant step toward adopting innovative technologies in the agricultural sector. By implementing an automated classification system, the factory aims to set a benchmark for others in the industry, showcasing how technology can transform traditional practices and drive advancements in agricultural processing.

1.4 Methodology

The development of the automated classification and detection system followed an Agile approach, divided into iterative sprints. Each sprint had clear objectives and deliverables, ensuring continuous progress and frequent evaluation. The following sections outline each sprint, its outcomes, and the time allocated.

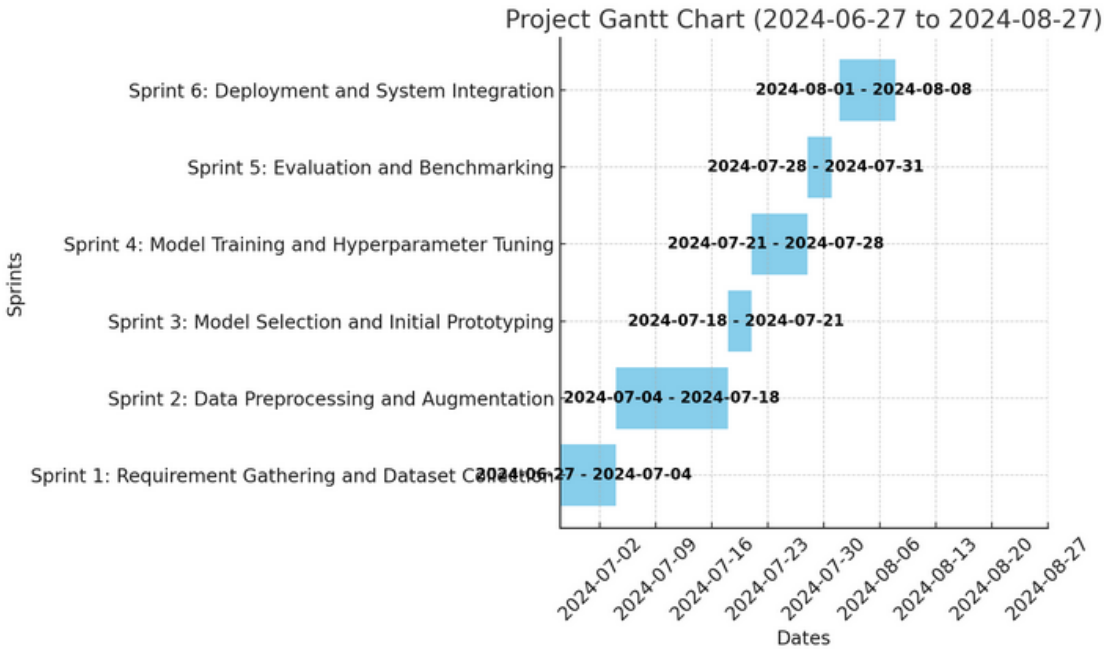

Figure 1.1 – Gantt chart

1.4.1 Sprint 1: Requirement Gathering and Dataset Collection (Duration: 1 week)

The first sprint focused on defining the project requirements and collecting initial data. Key objectives included:

— Engaging with stakeholders from the agricultural factory to understand specific classification and detection requirements.

— Sourcing a preliminary dataset of fruits and vegetables through publicly available repositories .

This sprint concluded with a detailed project plan and a labeled dataset ready for preprocessing.

1.4.2 Sprint 2: Data Preprocessing and Augmentation (Duration: 2 weeks)

This sprint aimed to prepare the dataset for model training:

— Annotating the images with bounding boxes and labels for various types of produce.

— Preprocessed images through normalization, resizing, and standardizing dimensions.

— Applied data augmentation techniques (rotation, scaling, color shifts) to increase dataset variability, improving model generalization.

Deliverables included a final, balanced dataset and an augmentation pipeline to be used in model training.

1.4.3 Sprint 3: Model Selection and Initial Prototyping (Duration: 3 days)

This sprint centered around model evaluation and prototyping:
— Selected YOLOv10 for object detection and a YOLOv8 classifier for fruit and vegetable categorization.
— Conducted preliminary testing on a small subset of data to assess feasibility.
The sprint delivered a functional prototype with baseline performance metrics.

1.4.4 Sprint 4: Model Training and Hyperparameter Tuning (Duration: 1 week)

In this sprint, full-scale training and optimization were carried out:
— Trained the model on 70% of the dataset, with 20% for validation and 10% for testing.
— Tuned hyperparameters, such as learning rate and batch size, to maximize accuracy while minimizing overfitting.
— Conducted rigorous testing to identify the best-performing model configuration.
This sprint resulted in a well-tuned model ready for evaluation.

1.4.5 Sprint 5: Evaluation and Benchmarking (Duration: 3 days)

This sprint focused on model evaluation and benchmarking:
— Evaluated model accuracy, precision, recall, and F1-score for both detection and classification tasks.
— Conducted speed tests to verify real-time performance.
— Benchmarked performance against baseline and alternative approaches.
Outcomes confirmed the model's readiness for deployment.

1.4.6 Sprint 6: Deployment and System Integration (Duration: 1 week)

The final sprint involved deploying the system within the factory environment:
— Integrated the model into a FastAPI-based application for real-time image processing.
— Developed an interface for monitoring and managing the classification system.
— Conducted on-site testing to ensure smooth production line integration.
This sprint concluded with the model achieving full operational status in the factory.

1.4.7 Sprint 7: Post-Deployment Monitoring and Improvement (Duration: Ongoing)

This sprint established a monitoring framework for long-term performance:
— Set up tracking for accuracy and detection rates to identify potential degradation.
— Scheduled periodic retraining with new data to account for produce variations.
— Collected feedback from stakeholders for future improvements.
This sprint finalized the methodology and ensured sustainability for the automated classification system.

1.5 Conclusion:

Until now, we have seen the host organization and their role in the field of IT. We have also discovered the project that I will be carrying out the internship.

Chapter 2

Data Preprocessing

2.1 Introduction

In this chapter, we delve into Data Exploration and Data preprocessing, pivotal steps in any data science project. The focus is on gaining valuable insights from the data through systematic exploration, a crucial phase in understanding and preparing the groundwork for subsequent analyses.

2.2 Dataset Overview

2.2.1 Dataset Composition and Sources

The dataset utilized in this project was sourced from **Roboflow**, where data for **82 different classes of fruits and vegetables** was collected. Each class contains labeled images with annotations specifying both the class and bounding box locations, making the dataset well-suited for both classification and detection tasks in an automated factory setting.

The images were curated to represent various conditions and perspectives, enhancing the dataset's robustness for real-time applications in industrial environments.

Figure 2.1 – Roboflow Logo

2.2.2 Class Distribution and Visualizations

An analysis of the class distribution revealed notable imbalances, with several classes having fewer than 3000 images. Visualizations of the dataset's class frequencies are provided to better understand the distribution:

— **Word Cloud of Class Distribution**: A word cloud visualization of class names illustrates the dataset composition, with the size of each class name indicating its relative frequency. This visual offers an intuitive sense of class prevalence and highlights classes with more substantial representation, as well as those with limited samples.

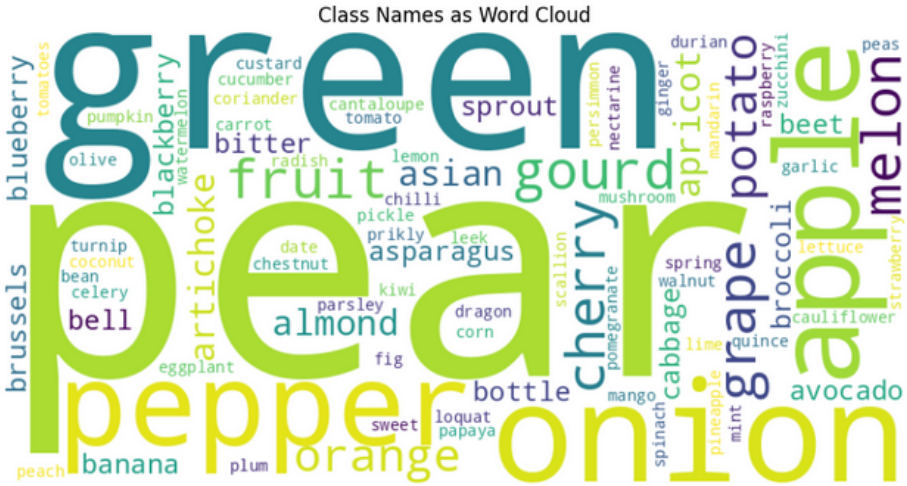

Figure 2.2 – Word Cloud of Class Distribution

— **Overall Class Frequency Bar Chart**: This bar chart offers a detailed view of image frequencies across all 82 classes. The chart reveals substantial variation in representation, with certain classes appearing considerably less frequently than others.

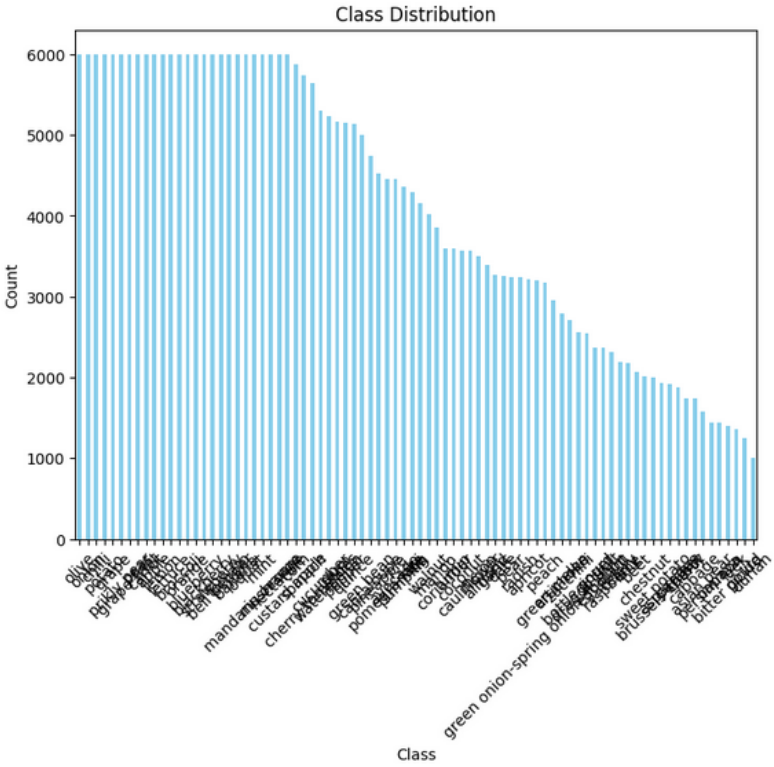

Figure 2.3 – Overall Class Frequency Bar Chart

— **Pie Chart of Classes Under 3000 Images**: This pie chart shows the proportion of images in classes with fewer than 3000 samples. As indicated, a significant portion of classes fall below this threshold, highlighting the need for specific augmentation strategies to balance the dataset.

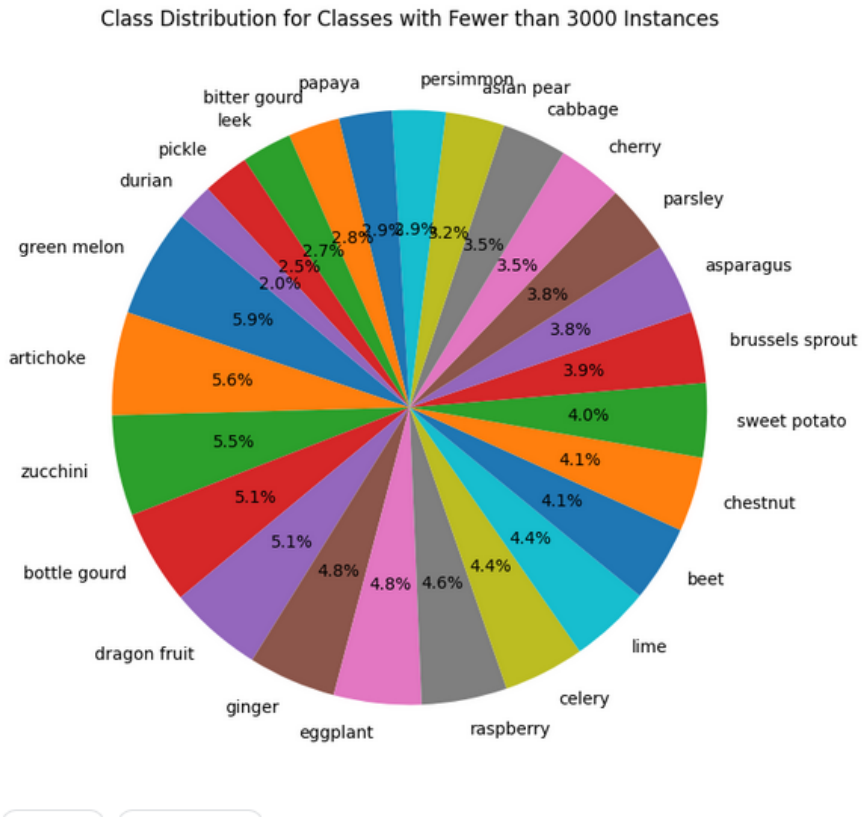

Figure 2.4 – Pie Chart of Classes Under 3000 Images

2.2.3 Addressing Imbalances

To address the identified class imbalances, particularly for those classes with fewer than 3000 images, we propose a two-fold approach: downsampling classes with more than 3000 labels and oversampling those with fewer than 3000 labels. Specifically, we considered several techniques:

— **Data Augmentation**: Augmentation techniques will be applied to classes with lower representation to increase sample diversity and help balance the dataset.

— **Data Labeling for Augmented Data**: Ensuring that the newly created augmented samples are accurately labeled is essential for maintaining the integrity of the dataset. This step will help in effectively training the model on a more balanced dataset.

These steps are crucial for ensuring reliable performance across all classes and minimizing bias, thereby enhancing the model's ability to generalize effectively.

2.3 Data Labeling :

In computer vision, data labeling is the process of annotating images or videos with information that identifies objects or regions of interest. This can include classifying images, drawing bounding boxes for object detection, or segmenting images. Accurate labeling is essential for training machine learning models, providing the ground truth needed for effective learning.

It is recommended to label data manually for greater accuracy; however, there are now models available that assist in automatically labeling images with high confidence.These automated models can generate labels with high confidence, significantly speeding up the process and reducing human effort, though they may still require some level of manual verification to ensure accuracy.

2.3.1 Segement Anything Model:

The SAM (Segment Anything Model) is an advanced AI model designed for image segmentation tasks created by Meta AI researcher. It can automatically generate high-quality segmentation masks for objects in images, making it particularly useful in computer vision applications. SAM operates by leveraging a large dataset to learn how to segment various objects effectively, allowing it to perform well across diverse scenarios without the need for extensive fine-tuning.

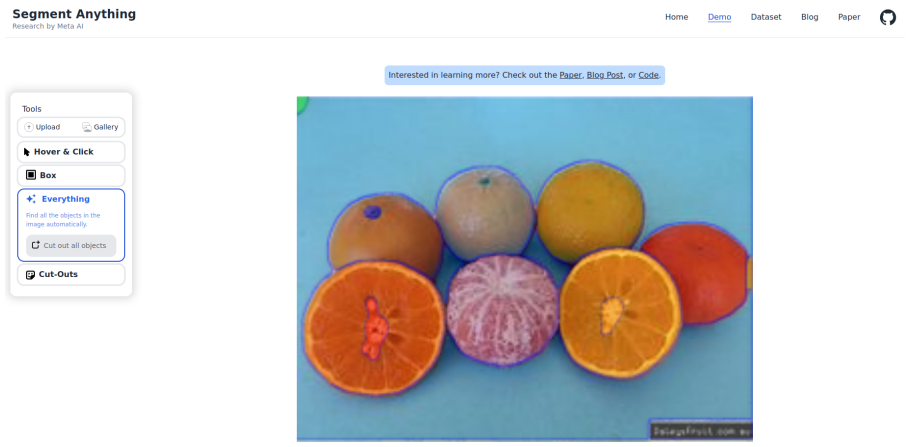

Figure 2.5 – Segement Anything Model

2.3.2 Grounded Dino Model(Detect Anything):

it's an open-set object detector, called Grounding DINO, by marrying Transformer-based detector DINO with grounded pre-training, which can detect arbitrary objects with human inputs such as category names or referring expressions.

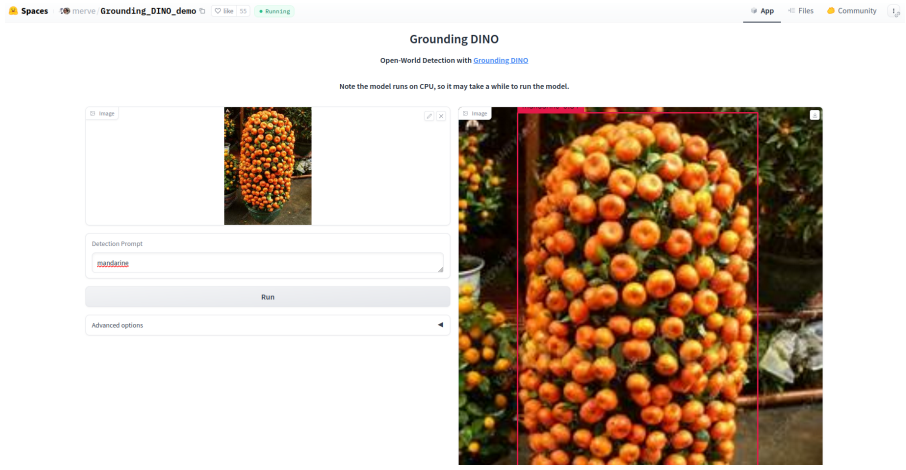

Figure 2.6 – Grounded Dino

2.3.3 Grounded SAM:

Grounded SAM combines Grounded DINO and the Segment Anything Model (SAM) to create a powerful tool for object detection and segmentation. This approach allows objects to be identified based on a text prompt and generates a mask for the specified object, which helps isolate it from the rest of the image. Although segmentation may not always be necessary for certain labeling tasks, using it in Grounded SAM is recommended as it improves the visualization process and ensures more accurate labeling, particularly when working with complex scenes.

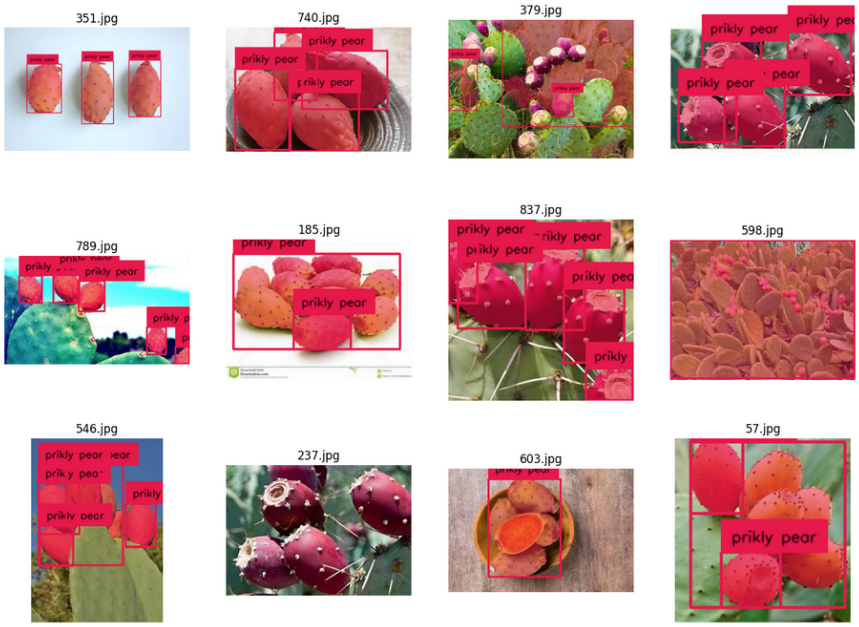

Figure 2.7 – AutoLabeling Images

2.4 Data Augmention :

In Computer Vision,Data augmentation is essential for enhancing model robustness by artificially expanding the dataset with transformations like **random cropping, shifting, flipping, rotation, zooming, shearing, and adding shadows**. These techniques increase the diversity of training data, allowing the model to learn to recognize fruits and vegetables from various perspectives. We intentionally avoided altering the colors of the images to preserve the natural color characteristics, ensuring that the model accurately associates specific colors with their respective classes. This approach helps improve the model's performance in real-world applications, where color recognition is critical for effective classification.

2.4.1 Random Crop and Resize:

— **Random Crop**: This technique involves randomly selecting a portion of an image to keep while discarding the rest. The size of the crop can either be fixed or randomly chosen within a specified range. This helps the model learn to focus on different parts of the image and makes it more robust to variations in object position.
— **Resize**: After cropping, the cropped image is usually resized to a standard size (e.g., 224x224 pixels for YOLO models). This ensures that all input images have the same dimensions, which is necessary for batch processing in neural networks.

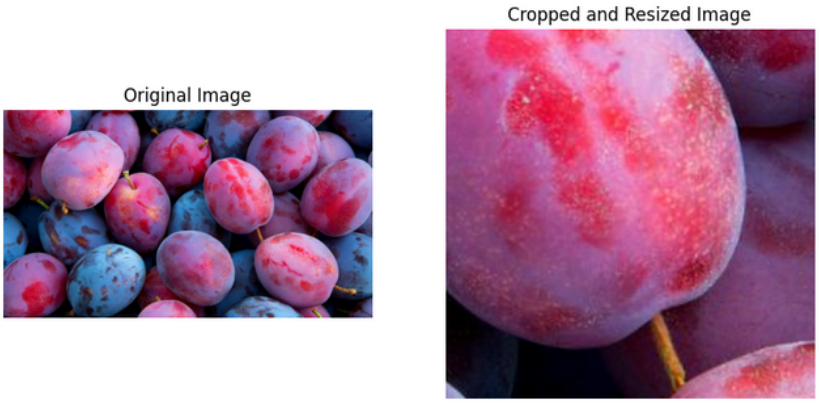

Figure 2.8 – Random Crop

2.4.2 Random Flip:

— **Horizontal/Vertical Flip**: Random flipping involves flipping an image along a specified axis (horizontal or vertical). This technique can help the model

become invariant to the orientation of the objects, allowing it to recognize them even when they appear in a mirrored form.

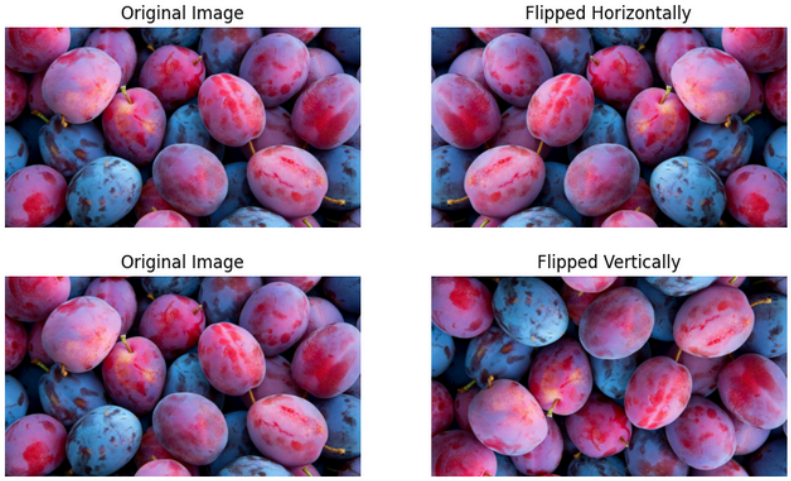

Figure 2.9 – Random Flip

2.4.3 Random Shadow:

This technique simulates the effect of shadows on the image by randomly adding dark patches (shadows) over parts of the image. This helps the model learn to be invariant to lighting conditions and can improve robustness by exposing it to variations in illumination.

Figure 2.10 – Random Shadow

2.4.4 Random Zoom:

Random zoom involves scaling the image up or down randomly. This can simulate the effect of the object being closer or further away from the camera. The image is resized to the original size after zooming to maintain consistency in input dimensions.

Figure 2.11 – Random Zoom

2.4.5 Random Shift:

This technique involves randomly translating (shifting) the image along the x and/or y axes. It helps the model learn to recognize objects that are not centered in the frame and can improve generalization by exposing it to various object placements.

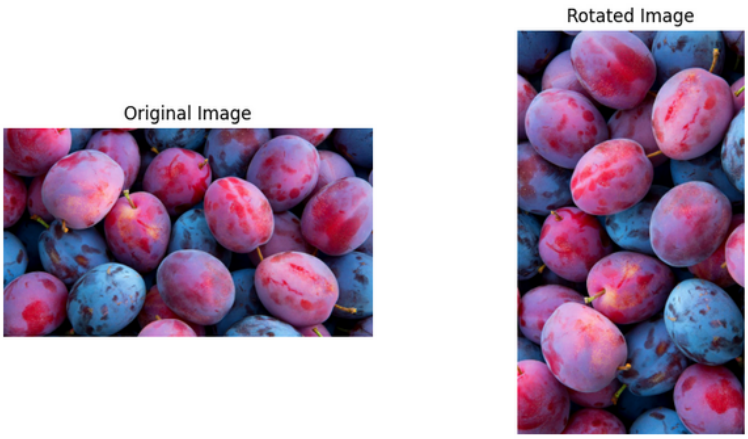

Figure 2.12 – Random Shift

2.4.6 Random Shear:

Shearing is a transformation that distorts the image in such a way that the image appears slanted or skewed. Random shear applies this effect by altering the angles of the image's axes. It helps the model learn to handle variations in perspective and object shapes.

Figure 2.13 – Random Shear

2.5 Data Splitting

After labeling and augmenting the dataset, as well as downsampling classes with more than 3000 images, we proceeded to split the dataset into training, validation, and test sets. This split ensures that each subset accurately represents the data distribution while providing sufficient samples for effective model training, evaluation, and testing.

Since YOLO handles multiple tasks, we need to adapt the dataset specifically for each task:

2.5.1 Classification Dataset

For the classification task, we focused on images that contain only a single, clearly labeled object. Each image in this dataset was assigned one label, streamlining the model's objective to classify one object per image. We applied specific criteria for selecting images, ensuring that each image represented only one class. Images containing overlapping objects or ambiguity were excluded to maintain high labeling precision.

Class balancing was prioritized to avoid biases toward certain classes. In cases where class imbalance was detected, we adjusted the number of samples per class using either data augmentation or selective downsampling to bring uniformity. The final classification dataset was split into training, validation, and test sets, with attention to representing each class evenly across splits to enhance model generalization.

Figure 2.14 – classification dataset

2.5.2 Detection Dataset

For the detection task, the dataset included images with multiple objects, each annotated with bounding boxes that specify their location and class. This task required unique labeling conventions: every object in each image was labeled with a bounding box, using the (x, y) coordinates and width and height dimensions to mark each object's boundaries.

During the split, we ensured that each subset (training, validation, and test) contained a balanced representation of all classes to allow the model to recognize and detect objects across varied contexts. By maintaining class representation across splits, we provided the model with exposure to different combinations of objects and variations within classes, facilitating robust detection capabilities across real-world scenarios

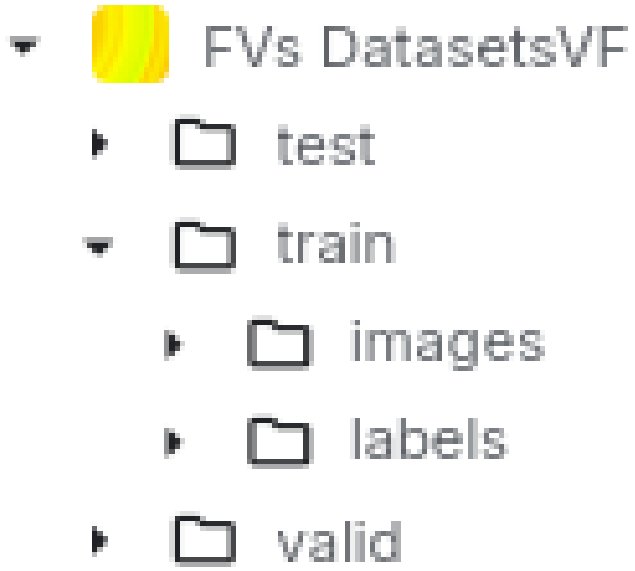

Figure 2.15 – detection dataset structure

2.5.3 Conclusion

In this chapter, we examine the initial dataset gathered from Roboflow, detailing the automated labeling process for missing images, data augmentation, and data balancing by downsampling classes with over 3000 samples. Finally, we discuss the dataset split for each task (classification or detection).

Chapter 3

Models and Performance Evaluation

3.1 Model for classification task

3.1.1 Introduction

Computer vision classification focuses on categorizing images by identifying the main object or concept they contains.

Common architectures for image classification include convolutional neural networks (CNNs) such as YOLOv8 , ResNet, VGG, and AltCLIP

In the next section, we will delve into a detailed discussion of the YOLOv8 model and evaluate its performance.

3.1.2 YOLOv8 model

YOLOv8 is a state-of-the-art object detection and image segmentation model created by Ultralytics, the developers of YOLOv5. YOLOv8 has native support for image classification tasks, too. YOLOv8, launched on January 10, 2023, features:

A new backbone network; A design that makes it easy to compare model performance with older models in the YOLO family; A new loss function and; A new anchor-free detection head

there are many YOLOv8 classication models released. These include:
— YOLOv8n-cls (Nano): Approximately 3.2 million parameters
— YOLOv8s-cls (Small): Approximately 11.2 million parameters
— YOLOv8m-cls (Medium): Approximately 25.9 million parameters
— YOLOv8l-cls (Large): Approximately 43.7 million parameters
— YOLOv8x-cls (Extra Large): Approximately 68.2 million parameters

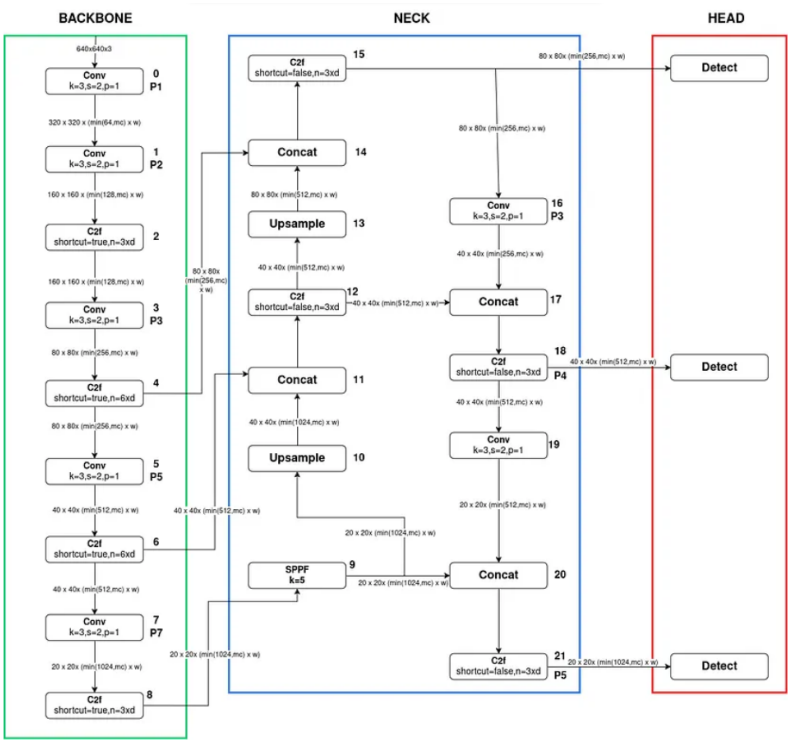

Figure 3.1 – YOLOv8 Architecture

3.1.3 Model Training

To train the model, a data.yaml file should be created to specify the classes and the localization paths for the training and validation folders. We chose to train for 80 epochs and used the Adam optimizer with a batch size of 16 due to resource constraints. The image size was set to 224 pixels, and we kept the default learning rate of 0.001 and a momentum of 0.937.

```
task: classify
mode: train
model: yolov8s-cls.pt
data: /kaggle/input/fvsaugmented/fvsaugmented_classification
epochs: 50
time: null
patience: 100
batch: 16
imgsz: 224
save: true
save_period: -1
cache: false
device: null
workers: 8
project: null
name: yolov8s
exist_ok: false
pretrained: true
optimizer: auto
verbose: true
seed: 0
deterministic: true
single_cls: false
rect: false
cos_lr: false
close_mosaic: 10
resume: false
amp: true
fraction: 1.0
profile: false
freeze: null
multi_scale: false
overlap_mask: true
mask_ratio: 4
dropout: 0.0
val: true
split: val
save_json: false
save_hybrid: false
conf: null
iou: 0.7
max_det: 300
half: false
dnn: false
plots: true
source: null
vid_stride: 1
stream_buffer: false
visualize: false
augment: false
agnostic_nms: false
classes: null
```

Figure 3.2 – Configuration file

3.1.4 Performance Evaluation

3.1.4.1 Confusion Matrix

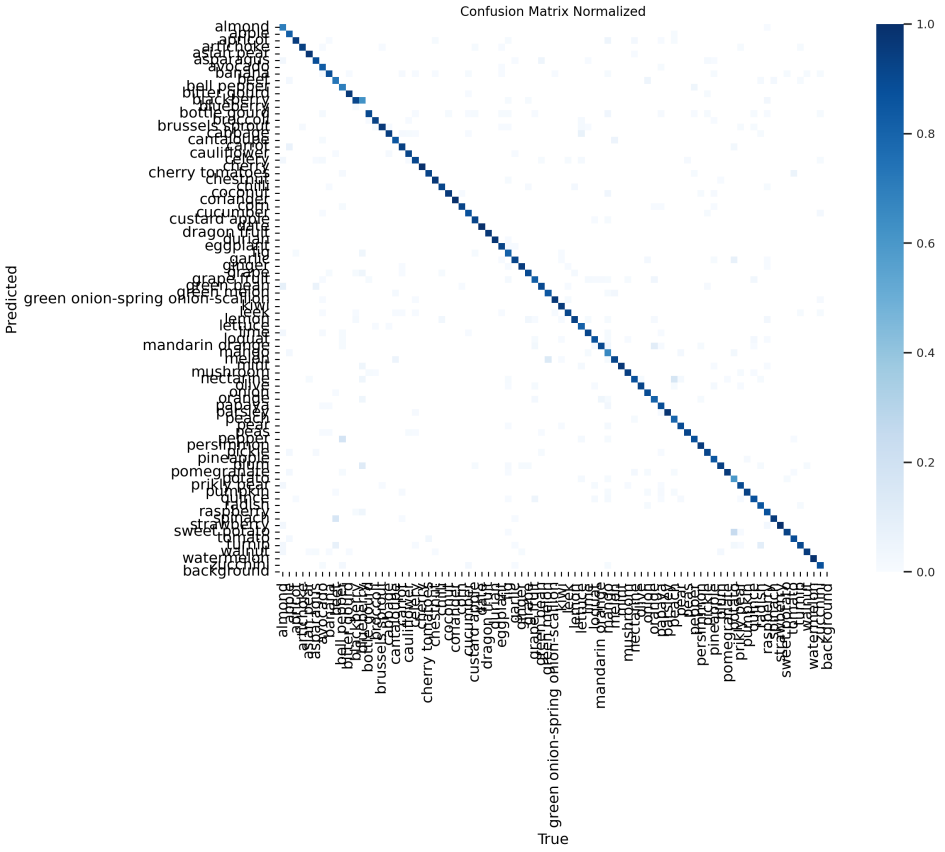

Figure 3.3 – Confusion Matrix

This confusion matrix visualization demonstrates exceptional classification performance across all categories in the fruits and vegetables recognition model. A pronounced dark blue diagonal line dominates the matrix, indicating consistently accurate predictions across the diverse range of classes, which include various fruits (such as apple, banana, cherry, mandarin), vegetables (bell pepper, cauliflower, cucumber), and other produce items. The minimal presence of off-diagonal elements, shown by the predominantly white spaces outside the main diagonal, indicates negligible misclassification between different categories. This suggests that the model has successfully learned distinctive features for each class, enabling clear differentiation even between potentially similar items. The inclusion of a background class, coupled with its strong diagonal representation, demonstrates the model's robust ability to not only correctly identify target objects but also to appropriately handle images without target objects. This comprehensive performance across all categories underscores the model's reliability and effectiveness in produce classification tasks.

3.1.4.2 Training Metrics

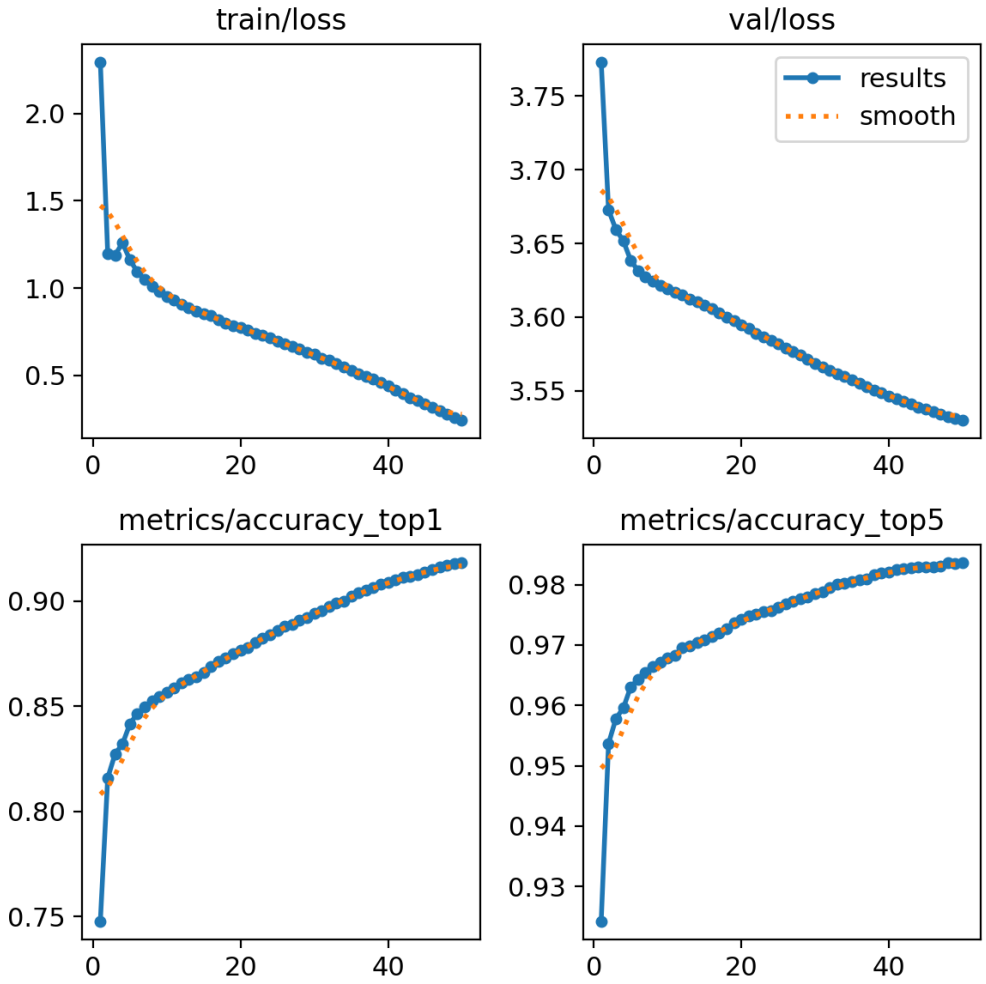

Figure 3.4 – Training Metrics

The training metrics demonstrate robust model performance and learning characteristics across all key indicators. The training loss exhibits a healthy descent from 2.0 to 0.3, while the validation loss shows a corresponding decrease from 3.75 to 3.53, indicating effective generalization without overfitting.
The model achieves impressive accuracy metrics, with top-1 accuracy improving substantially from 75% to 92%, and top-5 accuracy reaching an exceptional 98.5% from an initial 92.5%. The smooth, consistent progression of all metrics, coupled with the parallel trends between training and validation losses, suggests stable and effective training dynamics. The continued upward trajectory of accuracy metrics, combined with steadily decreasing loss values, indicates that the model successfully learned to extract meaningful features from the training data while maintaining strong generalization capabilities on unseen examples.

3.2 Model for detection Task

3.2.0.1 introduction

Object Detection is a fundamental task in computer vision where the goal is to identify and locate objects within an image or video. It involves drawing bounding boxes around detected objects and classifying them into predefined categories.

3.2.0.2 YOLOv10 Model

Real-time object detection aims to accurately predict the categories and positions of objects in images with low latency. The YOLO series has been at the forefront of this research due to its balance between performance and efficiency. However, reliance on NMS and architectural inefficiencies have hindered optimal performance. YOLOv10 addresses these issues by introducing consistent dual assignments for training without NMS and a holistic model design strategy focused on efficiency and accuracy.

3.2.0.3 Training Model

We used the same parameters as before for classification, but we changed the task to detection and the type of detection dataset

3.2.0.4 Performance Evaluation

3.2.0.5 Loss Metrics

The model demonstrates effective initial optimization, with box loss decreasing sharply from 5.87 to 2.54 within the first three epochs. After this rapid decline, the box loss continues to decrease more gradually, reaching 1.06 by epoch 34. The steady downward trend in loss, without notable fluctuations, suggests stable training and limited risk of overfitting.

3.2.0.6 Precision and Detection Metrics

Precision shows significant improvement, starting from nearly zero (0.00036) and increasing to 0.93 by the final epoch. The mAP@50 metric also demonstrates consistent enhancement, rising from 0.00189 to 0.82318, indicating progressively

accurate object detection over time. A marked improvement in precision around epoch 9 (jumping from 0.20 to 0.78) points to a critical phase in the model's learning progression.

3.2.0.7 Training Stability

The learning rate appears to be well-tuned, as indicated by:
— Smooth convergence patterns
— Minimal oscillations across key metrics
— Sustained improvements, even in later training stages

3.2.0.8 Performance Plateaus

The model shows signs of plateauing around epoch 25, with diminishing incremental improvements. Despite this plateau, minor gains persist through the final epochs, suggesting that extended training could further enhance model performance.

epoch	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B	val/box_loss	val/cls_loss	val/dfl_loss	lr/pg0	lr/pg1	lr/pg2
1	5.8736	19.06	7.8843	0.00036	0.02872	0.00189	0.00092	4.9344	11.127	6.2055	0.0033332	0.0033332	0.0033332
2	3.1432	7.4491	4.2098	0.58862	0.35562	0.36893	0.26415	2.7684	5.7202	4.2514	0.0066005	0.0066005	0.0066005
3	2.5422	4.6372	3.5638	0.60199	0.35684	0.3721	0.26783	2.8763	5.9015	4.4106	0.0098018	0.0098018	0.0098018
4	2.2582	3.6757	3.2766	0.14041	0.14259	0.12531	0.08037	3.6295	10.223	5.4923	0.009703	0.009703	0.009703
5	2.0257	3.0876	3.0501	0.06823	0.03256	0.04619	0.02704	3.8043	11.671	5.8509	0.009604	0.009604	0.009604
6	1.8997	2.7983	2.9248	0.13292	0.05533	0.08758	0.04898	3.6673	11.457	5.7032	0.009505	0.009505	0.009505
7	1.819	2.63	2.8454	0.18967	0.16219	0.1642	0.10313	3.4178	10.224	5.3122	0.009406	0.009406	0.009406
8	1.7578	2.5173	2.7859	0.20201	0.37554	0.29988	0.20601	3.0789	8.3676	4.7974	0.009307	0.009307	0.009307
9	1.6975	2.4002	2.7282	0.78147	0.3187	0.4494	0.33404	2.711	6.3383	4.2798	0.009208	0.009208	0.009208
10	1.6359	2.3117	2.67	0.77282	0.47903	0.56659	0.44568	2.392	4.799	3.8146	0.009109	0.009109	0.009109
11	1.5738	2.2223	2.611	0.77214	0.57085	0.63584	0.52159	2.1311	3.8262	3.4425	0.00901	0.00901	0.00901
12	1.5155	2.1604	2.563	0.81971	0.61581	0.68604	0.58133	1.9424	3.238	3.1585	0.008911	0.008911	0.008911
13	1.4608	2.0926	2.5161	0.86475	0.64258	0.7194	0.62313	1.7635	2.8341	2.9354	0.008812	0.008812	0.008812
14	1.4115	2.044	2.4819	0.87239	0.67048	0.73864	0.65289	1.6247	2.5629	2.7708	0.008713	0.008713	0.008713
15	1.3662	1.9894	2.4491	0.89773	0.68987	0.75378	0.67614	1.5053	2.3449	2.6398	0.008614	0.008614	0.008614
16	1.3362	1.9435	2.4255	0.9219	0.69371	0.76429	0.69482	1.4043	2.1887	2.5353	0.008515	0.008515	0.008515
17	1.3066	1.9247	2.4098	0.92622	0.69549	0.77324	0.71044	1.3451	2.1067	2.4609	0.008416	0.008416	0.008416
18	1.2869	1.8998	2.3989	0.90697	0.72199	0.78306	0.72545	1.251	1.9949	2.3909	0.008317	0.008317	0.008317
19	1.2632	1.8771	2.3847	0.91366	0.72681	0.79021	0.73662	1.1829	1.9055	2.3331	0.008218	0.008218	0.008218
20	1.2402	1.844	2.368	0.92225	0.72662	0.7967	0.74503	1.1273	1.8305	2.2894	0.008119	0.008119	0.008119
21	1.2242	1.8209	2.3588	0.91185	0.72545	0.79917	0.74822	1.0892	1.7986	2.2595	0.00802	0.00802	0.00802
22	1.2084	1.805	2.3489	0.88558	0.74683	0.80469	0.75391	1.0585	1.7369	2.2385	0.007921	0.007921	0.007921
23	1.1892	1.7794	2.3361	0.90982	0.7445	0.80862	0.75826	1.0381	1.7006	2.2197	0.007822	0.007822	0.007822
24	1.1761	1.7594	2.3265	0.92854	0.7414	0.81098	0.76131	1.0197	1.6697	2.2061	0.007723	0.007723	0.007723
25	1.1593	1.7445	2.3199	0.91728	0.74159	0.81127	0.76266	0.99933	1.6599	2.192	0.007624	0.007624	0.007624
26	1.1483	1.7283	2.3126	0.91777	0.73867	0.81172	0.76384	0.98566	1.6436	2.1811	0.007525	0.007525	0.007525
27	1.1348	1.7037	2.3044	0.92585	0.74132	0.81492	0.76699	0.97349	1.6128	2.1717	0.007426	0.007426	0.007426
28	1.1248	1.6854	2.2974	0.92166	0.74755	0.81618	0.76842	0.96232	1.5977	2.1641	0.007327	0.007327	0.007327
29	1.1145	1.6827	2.2923	0.92498	0.75032	0.81862	0.77083	0.95333	1.5732	2.158	0.007228	0.007228	0.007228
30	1.1094	1.6622	2.2899	0.92911	0.74614	0.8193	0.77202	0.94178	1.5653	2.1502	0.007129	0.007129	0.007129
31	1.095	1.6504	2.2819	0.90278	0.75739	0.81916	0.77193	0.93276	1.5555	2.1435	0.00703	0.00703	0.00703
32	1.0881	1.6416	2.2786	0.88914	0.76521	0.81972	0.77254	0.92709	1.5443	2.1355	0.006931	0.006931	0.006931
33	1.0744	1.6248	2.2702	0.93144	0.75063	0.82241	0.77557	0.92083	1.5251	2.1291	0.006832	0.006832	0.006832
34	1.0648	1.6142	2.2691	0.93082	0.75142	0.82318	0.77641	0.91569	1.5142	2.1239	0.006733	0.006733	0.006733

Figure 3.5 – Metrics of the first 43 epochs

3.3 Conclusion

In Chapter 3, we presented the approach taken for model selection, training, and evaluation for fruit and vegetable classification and detection tasks. For classification, YOLOv8 was chosen due to its efficient handling of single-object

images, providing a balance between speed and accuracy for classification tasks. For detection, YOLOv10 was selected for its ability to detect multiple objects within an image, making it ideal for identifying diverse produce types with high precision.

During training, hyperparameters were carefully tuned to optimize performance on each task, with training duration and computational resources balanced to maximize efficiency. Performance was assessed through metrics such as accuracy, precision, and recall, showing strong results across validation and test data for both models. Visualizations of the model's predictions further highlighted the accuracy in detection and classification, with minimal misclassifications. Overall, this evaluation demonstrates the models' suitability for real-world applications, ensuring reliable and accurate detection and classification of fruits and vegetables.

Chapter 4

Deployment of Inference Model

4.1 Deployment Strategy

In this section, we outline the deployment environment and the setup process for handling real-time inference requests. This includes details on the frameworks, containerization, and specific challenges faced during deployment.

4.1.1 Deployment Environment

4.1.1.1 FastAPI Framework

FastAPI was chosen as the backend framework due to its asynchronous capabilities and efficient handling of real-time requests. It enables fast processing of requests, which is essential for real-time inference in applications involving video and image processing.

Figure 4.1 – FastAPI

4.1.1.2 Docker Containerization

To ensure consistent deployment across different environments, Docker was used to containerize the application. Docker provides an isolated and replicable environment, which simplified dependency management and streamlined the deployment process. This also enables the application to scale easily across multiple

servers if needed.

Figure 4.2 – Docker logo

4.1.2 API Endpoints and Real-Time Inference

4.1.2.1 Endpoint Setup

Four main endpoints were created, each designed to handle specific tasks related to image and video classification or detection. These endpoints are as follows:

— /**classify**/**v1**: Classifies objects in a single uploaded image and returns predictions as JSON.

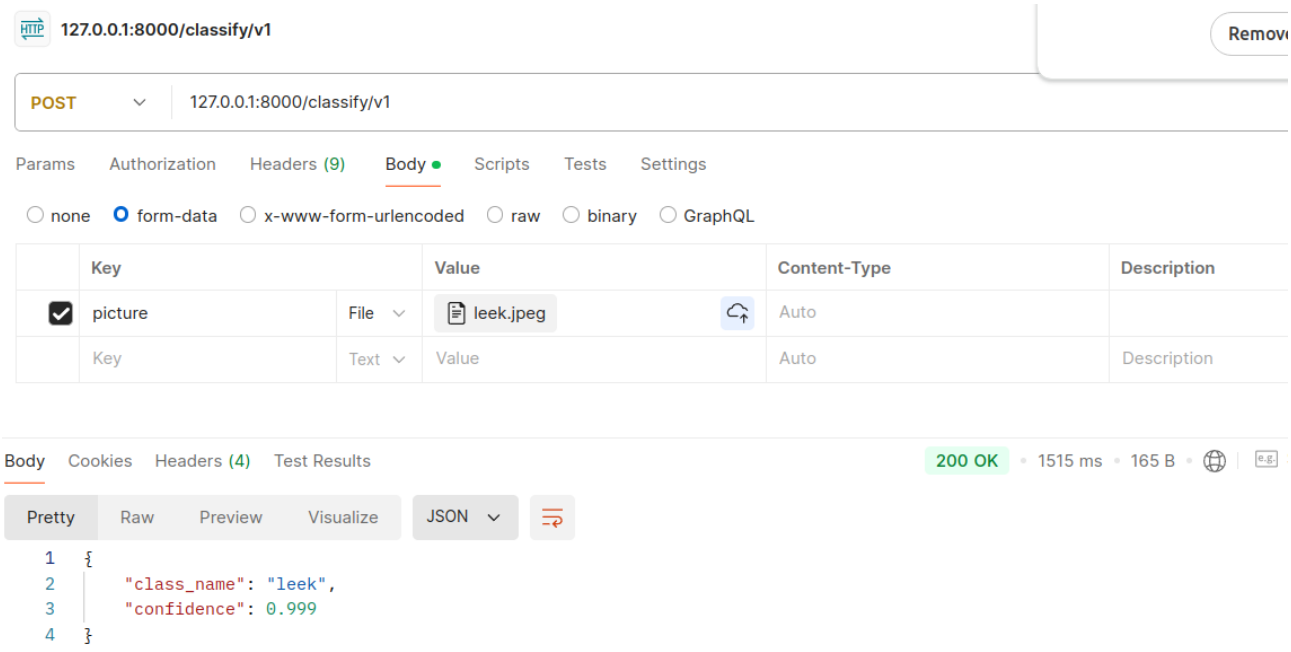

Figure 4.3 – Classification Endpoint

— /**classify**/**v2**: Performs classification on each frame of an uploaded video and streams the predictions as a JSON response.

— /**detect**/**v1**: Detects objects in a single uploaded image and returns an annotated image with bounding boxes.

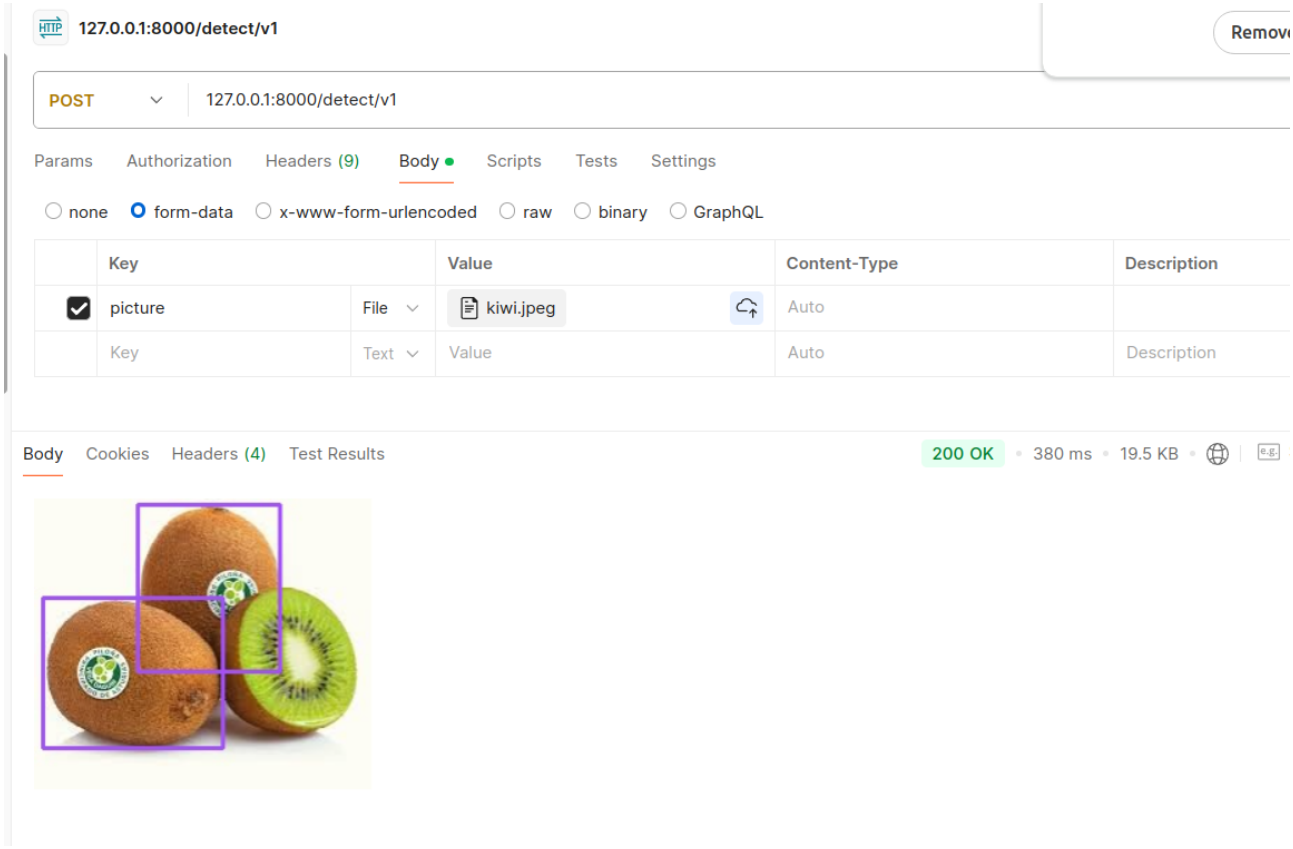

Figure 4.4 – Detection Endpoint

— **/detect_video/v1**: Detects objects in each frame of an uploaded video, annotates frames with bounding boxes, and streams the annotated video back.

Each endpoint uses the POST method to accept images or videos and process them based on the task requirements.

4.1.2.2 Real-Time Processing

For real-time inference on video streams, the endpoints were configured to process frames individually and return predictions as they are available. This setup prevents latency issues caused by waiting for the entire video to process, offering a smoother and more responsive user experience.

4.1.3 Challenges and Solutions

4.1.3.1 Handling Large Files

One major challenge was managing large video files, as processing them all at once could lead to memory issues and high latency. This was mitigated by implementing frame-by-frame processing for videos, allowing predictions to be streamed back as they were generated.

4.1.3.2 Ensuring Consistent Performance

To ensure the model's performance remained stable under load, the system was monitored and adjusted to handle high numbers of requests. Docker container limits were also set to manage memory and CPU usage efficiently, ensuring optimal performance for all endpoints.

This deployment strategy provided a scalable, responsive system for handling classification and detection requests in both image and video formats, tailored to meet real-time inference requirements in a robust environment.

4.2 Inference and Optimization

In this section, we analyze the model inference times for classification and detection tasks and describe the optimization strategies implemented to improve processing speed and resource efficiency.

4.2.1 Model Inference Times

During deployment, the model inference times were recorded for both the classification and detection tasks:

— **Detection Inference Time:** The detection model, based on YOLOv10, achieved an average inference time of 380 ms per image.

— **Classification Inference Time:** The classification model, based on YOLOv8 and ResNet, recorded a longer inference time of 1515 ms per image.

These times highlight the efficiency of the detection model, while the classification model's inference time indicates potential areas for improvement, particularly if deployed for real-time applications.

4.2.2 Optimization Strategies

To enhance inference speed and optimize resource usage, several strategies were implemented:

4.2.2.1 Input Resizing

One of the simplest yet effective techniques to reduce inference time was input resizing. By reducing the input image resolution, the model processed fewer

pixels, which decreased computation time. This resizing strategy was carefully balanced to retain sufficient detail for accurate detection and classification.

4.2.2.2 Hardware Acceleration

For faster computation, the deployment environment leveraged hardware acceleration where available. The models were optimized to run on GPUs, which are well-suited for handling the large matrix operations involved in deep learning inference. This provided a significant speedup, especially in handling batched inputs and real-time video inference tasks.

4.2.2.3 Batch Processing

To improve efficiency during batch inference, images were processed in batches rather than individually. This approach allowed for better utilization of GPU resources, reducing the overall inference time when handling multiple requests simultaneously.

4.2.2.4 Quantization and Model Pruning

Further optimizations included exploring model quantization and pruning. These techniques reduce model size by using lower precision representations (e.g., 8-bit integers instead of 32-bit floating points) or removing less important neurons, which resulted in a smaller model size and faster inference times without a significant loss in accuracy.

These optimizations collectively contributed to a more efficient deployment, ensuring the models could handle inference tasks with lower latency and resource consumption, while maintaining accuracy and reliability in predictions.

4.2.3 Conclusion

In this chapter, we discussed the deployment of the inference model, focusing on setting up a robust environment using FastAPI and Docker for scalable and reliable service. We outlined the architecture of our deployment pipeline, including the endpoints for handling image and video inputs in both classification and detection tasks.

The deployment strategy effectively manages real-time inference requests by optimizing response times through input resizing, hardware acceleration, batch

processing, and model optimization techniques such as quantization and pruning. These improvements enabled us to achieve a balance between model accuracy and speed, ensuring the system could deliver prompt and precise results in practical applications.

Overall, this deployment approach provides a solid foundation for efficient and scalable image and video processing, demonstrating the model's readiness for real-world scenarios where low latency and high throughput are critical. Future enhancements may involve further model tuning and integration with advanced hardware solutions to maintain optimal performance as demands grow.

References

https://medium.com/axinc-ai/grounded-sam-segmented-any-object-from-text-77
https://www.tensorflow.org/tutorials/images/data_augmentation https:
//docs.ultralytics.com/tasks/classify/

https://docs.ultralytics.com/fr/models/yolov10/

https://fastapi.tiangolo.com/