UNIVERSITY OF WEST LONDON
The Career University

Computer Vision- CP70072E_05

Topic – Object Detection

Student ID: [21600713]

Full Name: [Mohammed Haris Shaikh]

Table of Contents

Abstract

This report presents the development and evaluation of an object detection system for identifying three types of fruits: apples, bananas, and oranges.

The system leverages the power of YOLOv8, a state-of-the-art deep learning model specifically designed for object detection. Inspired by advancements in fruit recognition using deep learning approaches like DeepFruits, this project aims to demonstrate the effectiveness of YOLOv8 in this domain.

A dataset of 240 training images and 60 test images is utilized. Each image contains bounding box annotations for the fruit objects in XML format.

The YOLOv8 model undergoes training on the designated training dataset, while a separate validation set is employed to optimize its performance metrics. Following training and optimization, the model is evaluated on the provided test dataset. Standard performance metrics like precision, recall, and F1 score are used to assess the system's effectiveness in accurately detecting and localizing fruits within images.

This project contributes to the field of computer vision by showcasing the practical application of YOLOv8 for fruit recognition.

This technology offers potential benefits in various sectors, including agriculture, food processing, and inventory management.

Introduction

The surge in deep learning techniques has significantly impacted computer vision, enabling the creation of highly accurate and efficient object detection systems. This project delves into the domain of fruit detection within images, a field with significant applications in agriculture, food processing, and retail inventory management. Automatic identification and localization of fruits in images can streamline processes like fruit harvesting, sorting, and quality control.

The project utilizes a dataset containing images of three common fruits: apples, bananas, and oranges. This dataset comprises 240 training images and 60 test images, each annotated with bounding boxes for the fruit objects in XML format. While valuable for training and evaluating object detection models, the dataset necessitates pre-processing to conform to the specific requirements of YOLOv8.

Challenges and Pre-processing Considerations

One key challenge encountered is that YOLOv8 utilizes a .txt file format, known as a labels file, to define the object classes present in the dataset. The provided XML annotations require conversion to this format. This conversion involves extracting relevant information from the XML files, such as class labels and bounding box coordinates, and structuring them in the YOLOv8 labels file format.

Another challenge is the potential presence of missing data within the dataset. Addressing missing data might involve techniques.

Data pre-processing is crucial to ensure compatibility with YOLOv8. This process encompasses tasks like:

- **Label conversion:** Transforming XML annotations into YOLOv8 labels format (.txt files).
- **Missing data handling:** Addressing missing data through techniques like imputation, augmentation, or data removal.
- **Data formatting:** Ensuring all data (images and labels) are formatted as per YOLOv8 specifications.

By meticulously addressing these challenges, we can prepare the dataset for training an effective fruit detection model using YOLOv8.

Main Content

This project can be broadly divided into six key stages:

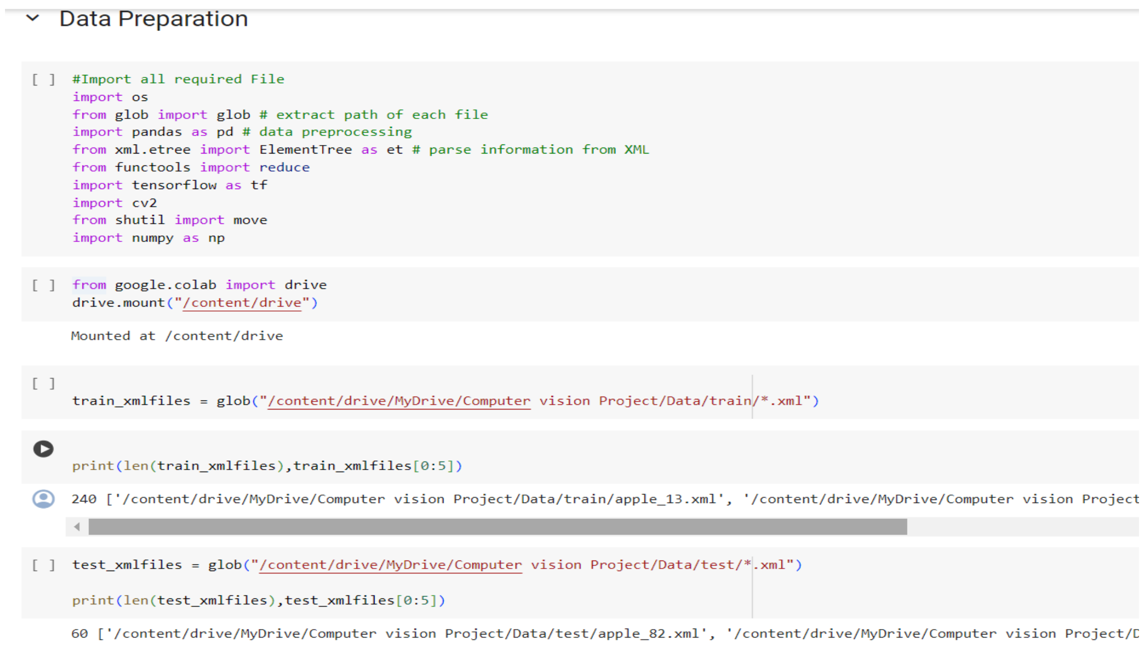
1. **Data Acquisition and Preprocessing:** This stage involved obtaining the fruit detection dataset and preparing it for training the YOLOv8 model.
2. **Model Selection and Training:** Here, a suitable YOLOv8 model variant was chosen based on desired speed and accuracy.
3. **Model Evaluation and Visualization:** This stage focused on assessing the trained model's performance on the test set using metrics like precision, recall, and F1 score. Visualization techniques like confusion matrices.
4. **Predicting on Entire Test Data:** The trained model's performance was evaluated on the entire test set to obtain final accuracy metrics.
5. **Video Prediction :** If video prediction was implemented, details about the setup, libraries used for video processing, and integration with the model for real-time detection would be included here.
6. **Conclusion and Future Work:** This final section would summarize the project's findings, reiterate the importance of fruit detection, and discuss potential future applications or areas for improvement in the system.

1. Data Preparation and Preprocessing

This section focuses on the preparation and pre-processing of the fruit detection dataset for training a YOLOv8 model.

Dataset Description:

The project utilized a pre-existing fruit detection dataset containing images of three types of fruits: oranges, bananas, and apples. The dataset was already split into training (240 images) and test sets (60 images). Each image in the training set was accompanied by an XML annotation file containing bounding box information for the fruits present.



Data Preparation

```
[ ] #Import all required File
    import os
    from glob import glob # extract path of each file
    import pandas as pd # data preprocessing
    from xml.etree import ElementTree as et # parse information from XML
    from functools import reduce
    import tensorflow as tf
    import cv2
    from shutil import move
    import numpy as np
```

```
[ ] from google.colab import drive
    drive.mount("/content/drive")

    Mounted at /content/drive
```

```
[ ]
    train_xmlfiles = glob("/content/drive/MyDrive/Computer vision Project/Data/train/*.xml")
```

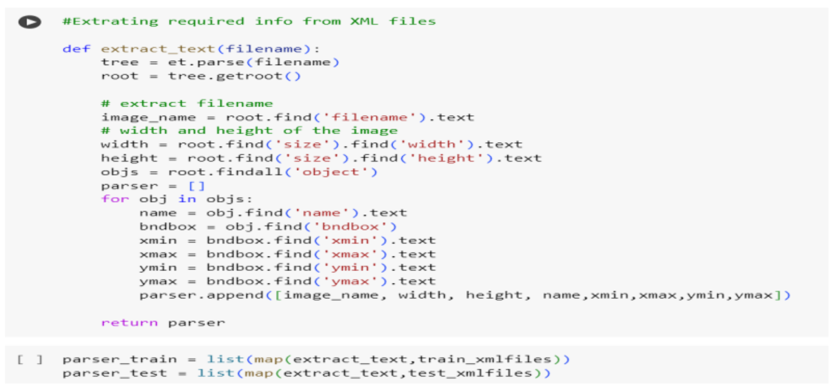
```
    print(len(train_xmlfiles),train_xmlfiles[0:5])

    240 ['/content/drive/MyDrive/Computer vision Project/Data/train/apple_13.xml', '/content/drive/MyDrive/Computer vision Project
```

```
[ ] test_xmlfiles = glob("/content/drive/MyDrive/Computer vision Project/Data/test/*.xml")

    print(len(test_xmlfiles),test_xmlfiles[0:5])

    60 ['/content/drive/MyDrive/Computer vision Project/Data/test/apple_82.xml', '/content/drive/MyDrive/Computer vision Project/D
```

Data Preprocessing:

Several pre-processing steps were undertaken to prepare the data for YOLOv8 training:

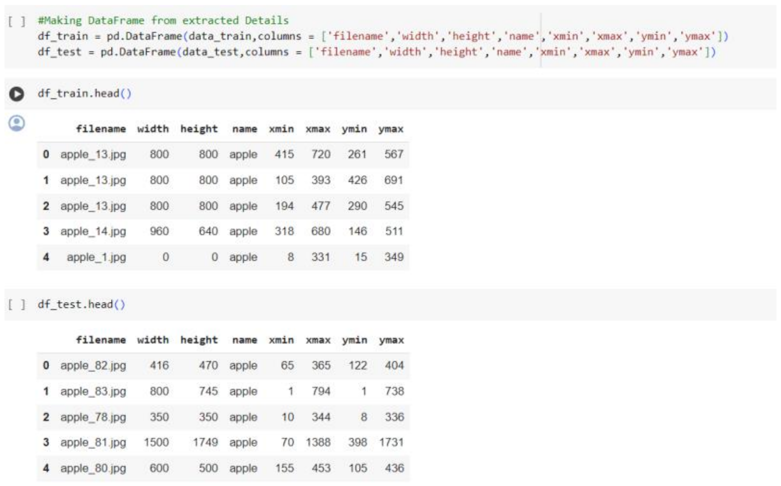
1. **Information Extraction:** Using the xml.etree library, relevant data from the XML annotations was extracted. This included object names (fruit types), bounding box coordinates (xmin, xmax, ymin, ymax), image width, and height.



```
#Extrating required info from XML files

def extract_text(filename):
    tree = et.parse(filename)
    root = tree.getroot()

    # extract filename
    image_name = root.find('filename').text
    # width and height of the image
    width = root.find('size').find('width').text
    height = root.find('size').find('height').text
    objs = root.findall('object')
    parser = []
    for obj in objs:
        name = obj.find('name').text
        bndbox = obj.find('bndbox')
        xmin = bndbox.find('xmin').text
        xmax = bndbox.find('xmax').text
        ymin = bndbox.find('ymin').text
        ymax = bndbox.find('ymax').text
        parser.append([image_name, width, height, name,xmin,xmax,ymin,ymax])

    return parser
```

```
[ ] parser_train = list(map(extract_text,train_xmlfiles))
    parser_test = list(map(extract_text,test_xmlfiles))
```

2. **Data Exploration:** The extracted information was converted into a Pandas dataframe format for easier exploration and manipulation.



```
[ ] #Making DataFrame from extracted Details
    df_train = pd.DataFrame(data_train,columns = ['filename','width','height','name','xmin','xmax','ymin','ymax'])
    df_test = pd.DataFrame(data_test,columns = ['filename','width','height','name','xmin','xmax','ymin','ymax'])
```

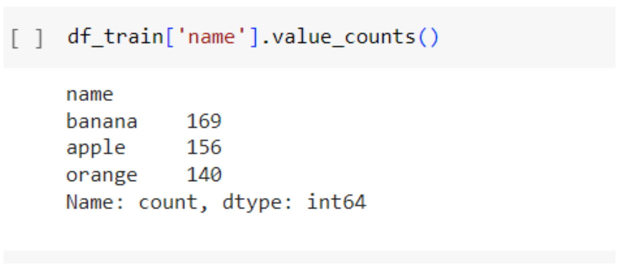
```
df_train.head()
```

	filename	width	height	name	xmin	xmax	ymin	ymax
0	apple_13.jpg	800	800	apple	415	720	261	567
1	apple_13.jpg	800	800	apple	105	393	426	691
2	apple_13.jpg	800	800	apple	194	477	290	545
3	apple_14.jpg	960	640	apple	318	680	146	511
4	apple_1.jpg	0	0	apple	8	331	15	349

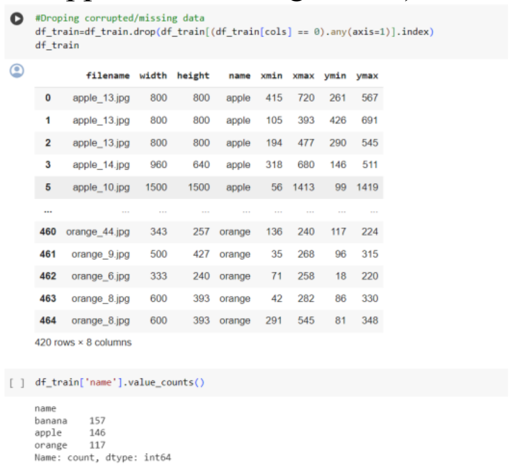
```
[ ] df_test.head()
```

	filename	width	height	name	xmin	xmax	ymin	ymax
0	apple_82.jpg	416	470	apple	65	365	122	404
1	apple_83.jpg	800	745	apple	1	794	1	738
2	apple_78.jpg	350	350	apple	10	344	8	336
3	apple_81.jpg	1500	1749	apple	70	1388	398	1731
4	apple_80.jpg	600	500	apple	155	453	105	436

3. **Missing Data Handling:** The dataframe revealed the presence of missing data. To ensure data quality, these rows were identified and removed, resulting in adjusted train class distribution (bananas: 157, apples: 146, oranges: 117).



```
[ ] df_train['name'].value_counts()

name
banana     169
apple      156
orange     140
Name: count, dtype: int64
```



```
#Droping corrupted/missing data
df_train=df_train.drop(df_train[(df_train[cols] == 0).any(axis=1)].index)
df_train
```

	filename	width	height	name	xmin	xmax	ymin	ymax
0	apple_13.jpg	800	800	apple	415	720	261	567
1	apple_13.jpg	800	800	apple	105	393	426	691
2	apple_13.jpg	800	800	apple	194	477	290	545
3	apple_14.jpg	960	640	apple	318	680	146	511
5	apple_10.jpg	1500	1500	apple	56	1413	99	1419
...	...	...	...	...	...	...	...	...
460	orange_44.jpg	343	257	orange	136	240	117	224
461	orange_9.jpg	500	427	orange	35	268	96	315
462	orange_6.jpg	333	240	orange	71	258	18	220
463	orange_8.jpg	600	393	orange	42	282	86	330
464	orange_8.jpg	600	393	orange	291	545	81	348

420 rows × 8 columns

```
[ ] df_train['name'].value_counts()

name
banana     157
apple      146
orange     117
Name: count, dtype: int64
```

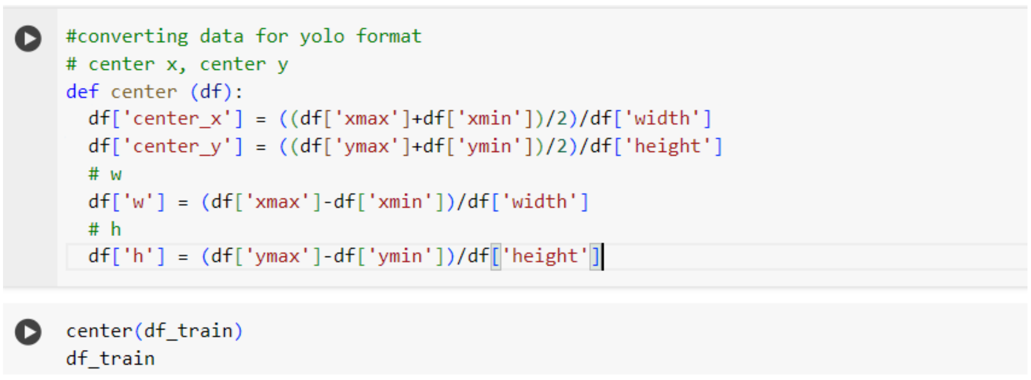
4. **YOLOv8 Format Conversion:** YOLOv8 requires bounding box information in the format of center coordinates (center_x, center_y), width (w), and height (h). These were calculated using the extracted bounding box coordinates and image dimensions.

5. **Label Encoding:** To convert the class labels (fruit types) into a numerical format suitable for YOLOv8 training, label encoding was applied.

Saving Labels and Images:

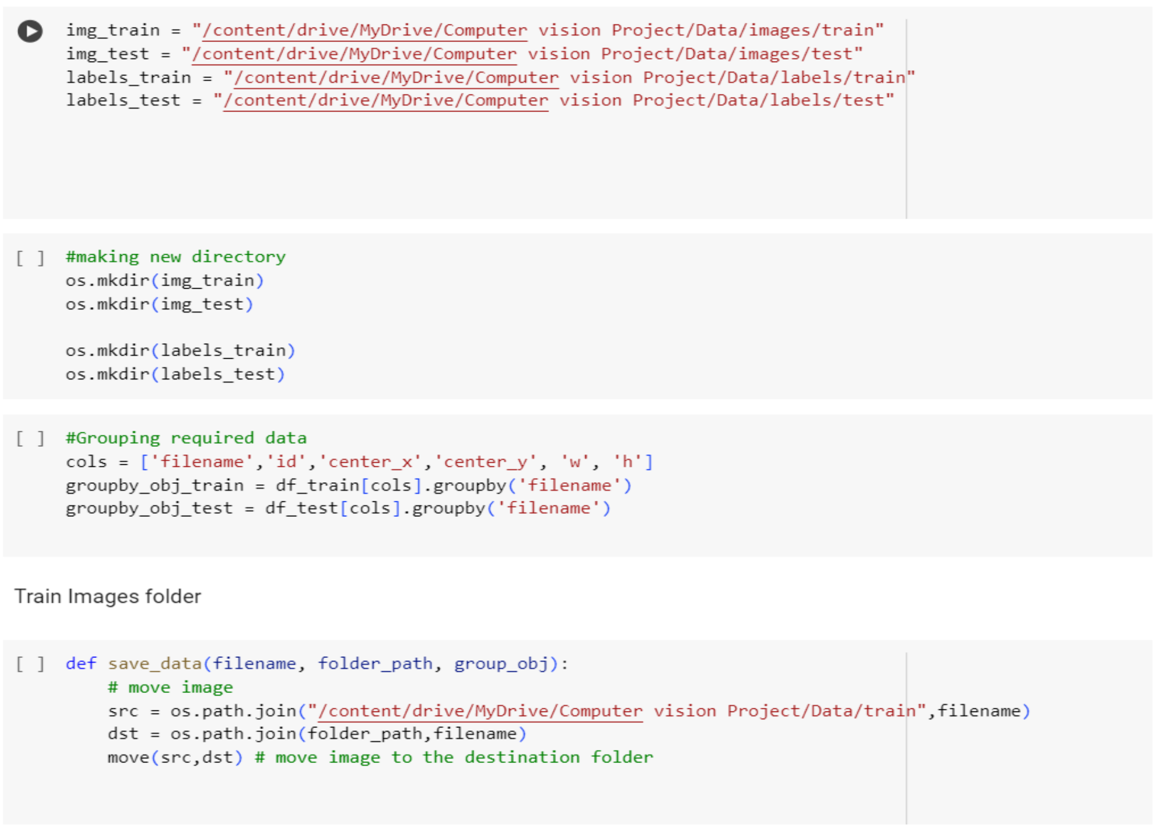
Following pre-processing, the dataset was reorganized for efficient training:

1. **Directory Creation:** Using the os library, separate directories were created for training and test images, as well as their corresponding labels files.
2. **Data Relocation:** Training and test images were moved to their respective directories.
3. **Label File Generation:** For each image, a label file (.txt format) was created in the corresponding label directory. These label files contained only the following information for each fruit object detected:
   - Filename (of the corresponding image)
   - Class ID (encoded numerical representation of the fruit type)
   - Center X coordinate (normalized)
   - Center Y coordinate (normalized)
   - Width (normalized)
   - Height (normalized)



```
#converting data for yolo format
# center x, center y
def center (df):
  df['center_x'] = ((df['xmax']+df['xmin'])/2)/df['width']
  df['center_y'] = ((df['ymax']+df['ymin'])/2)/df['height']
  # w
  df['w'] = (df['xmax']-df['xmin'])/df['width']
  # h
  df['h'] = (df['ymax']-df['ymin'])/df['height']
```

```
center(df_train)
df_train
```



```
img_train = "/content/drive/MyDrive/Computer vision Project/Data/images/train"
img_test = "/content/drive/MyDrive/Computer vision Project/Data/images/test"
labels_train = "/content/drive/MyDrive/Computer vision Project/Data/labels/train"
labels_test = "/content/drive/MyDrive/Computer vision Project/Data/labels/test"
```

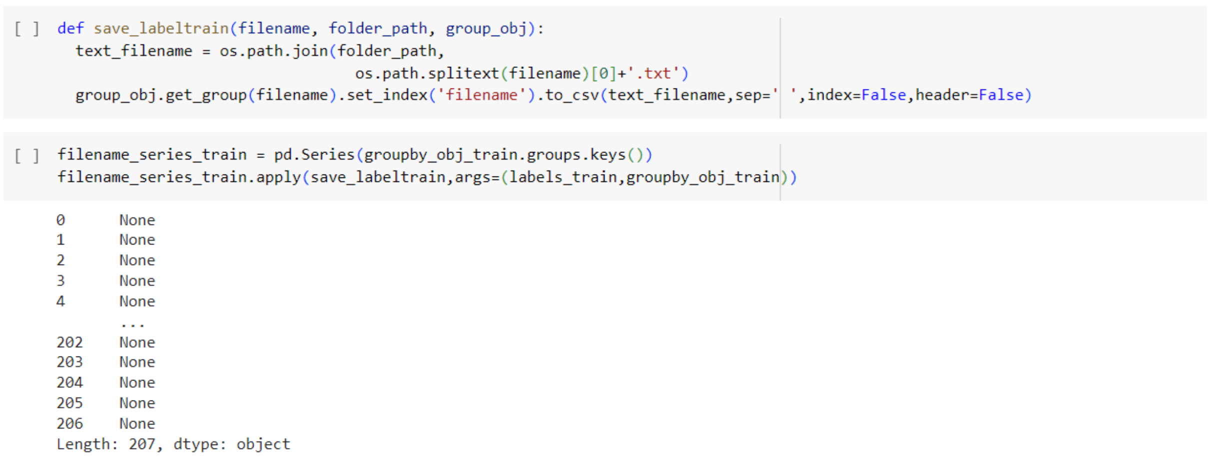
```
[ ] #making new directory
    os.mkdir(img_train)
    os.mkdir(img_test)

    os.mkdir(labels_train)
    os.mkdir(labels_test)
```

```
[ ] #Grouping required data
    cols = ['filename','id','center_x','center_y', 'w', 'h']
    groupby_obj_train = df_train[cols].groupby('filename')
    groupby_obj_test = df_test[cols].groupby('filename')
```

Train Images folder

```
[ ] def save_data(filename, folder_path, group_obj):
        # move image
        src = os.path.join("/content/drive/MyDrive/Computer vision Project/Data/train",filename)
        dst = os.path.join(folder_path,filename)
        move(src,dst) # move image to the destination folder
```

Train label Folder

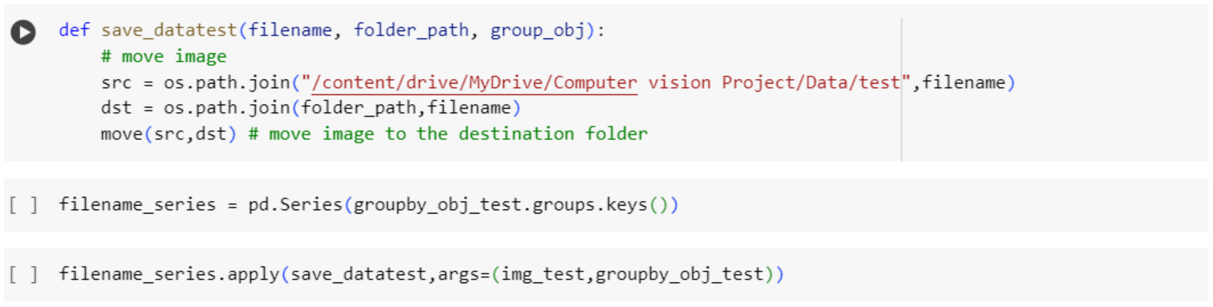


```
[ ] def save_labeltrain(filename, folder_path, group_obj):
      text_filename = os.path.join(folder_path,
                                   os.path.splitext(filename)[0]+'.txt')
      group_obj.get_group(filename).set_index('filename').to_csv(text_filename,sep=' ',index=False,header=False)
```

```
[ ] filename_series_train = pd.Series(groupby_obj_train.groups.keys())
    filename_series_train.apply(save_labeltrain,args=(labels_train,groupby_obj_train))
```

```
0      None
1      None
2      None
3      None
4      None
       ...
202    None
203    None
204    None
205    None
206    None
Length: 207, dtype: object
```

Test Images Folder

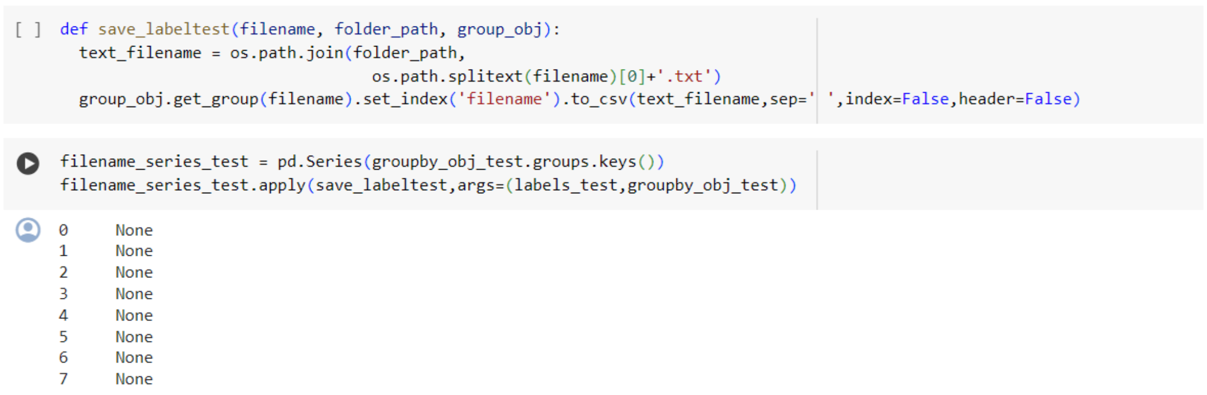


```
def save_datatest(filename, folder_path, group_obj):
    # move image
    src = os.path.join("/content/drive/MyDrive/Computer vision Project/Data/test",filename)
    dst = os.path.join(folder_path,filename)
    move(src,dst) # move image to the destination folder
```

```
[ ] filename_series = pd.Series(groupby_obj_test.groups.keys())
```

```
[ ] filename_series.apply(save_datatest,args=(img_test,groupby_obj_test))
```

Test Label Folder



```
[ ] def save_labeltest(filename, folder_path, group_obj):
      text_filename = os.path.join(folder_path,
                                   os.path.splitext(filename)[0]+'.txt')
      group_obj.get_group(filename).set_index('filename').to_csv(text_filename,sep=' ',index=False,header=False)
```

```
filename_series_test = pd.Series(groupby_obj_test.groups.keys())
filename_series_test.apply(save_labeltest,args=(labels_test,groupby_obj_test))
```

```
0      None
1      None
2      None
3      None
4      None
5      None
6      None
7      None
```

2. Importing and Training YOLO Model

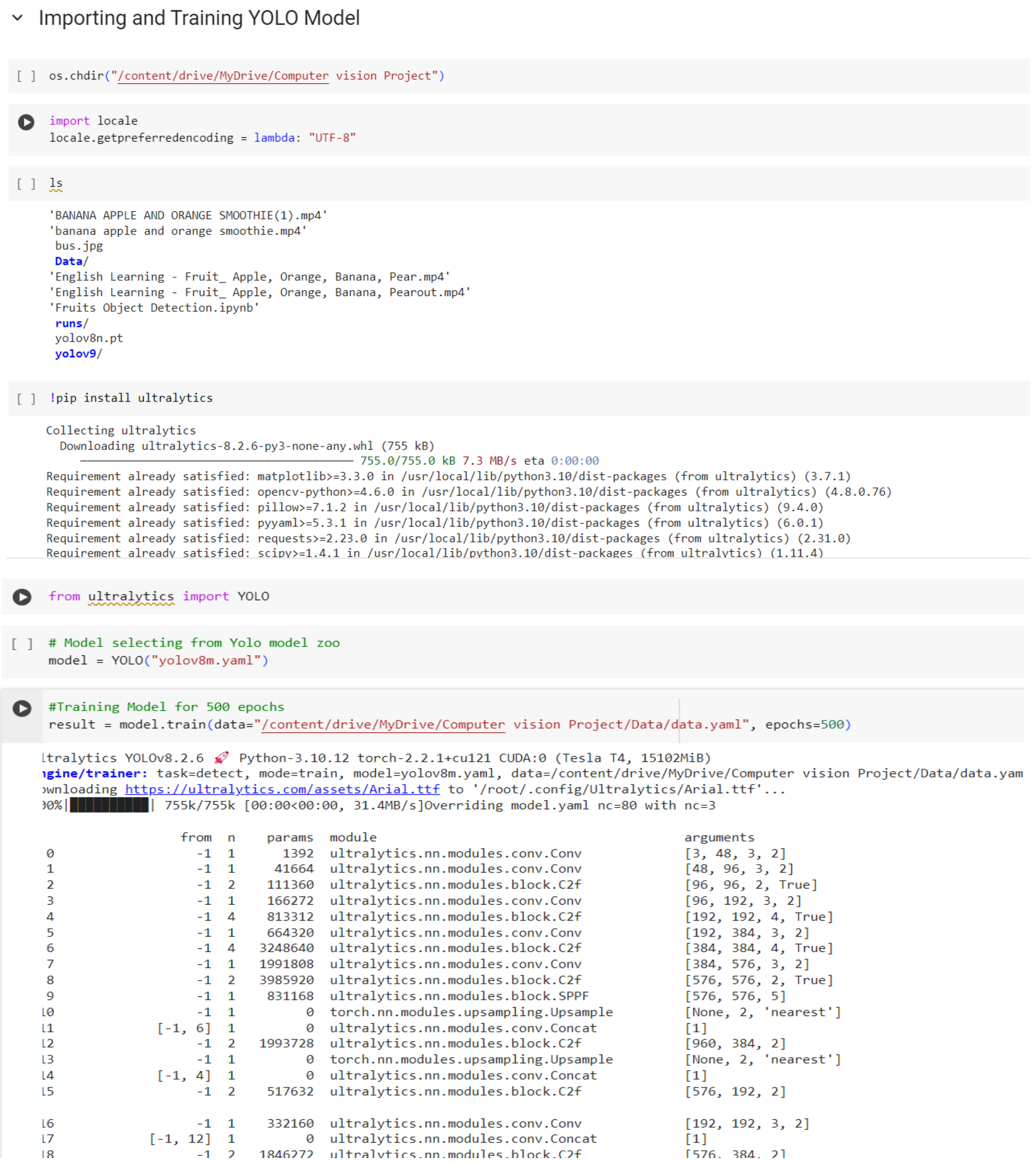
This section details the process of importing, configuring, and training the YOLOv8 model for fruit detection.

Model Selection and Library:

The Ultralytics library, providing functionalities for YOLOv8 models, was installed and imported. The chosen YOLOv8 model variant was yolov8m, offering a balance between speed and accuracy for the specific needs of this project.

Training Process:

- The pre-processed training data, consisting of images and corresponding label files, was utilized for model training.
- The training process ran for a maximum of 500 epochs, representing iterations of training the model on the entire dataset.
- To prevent overfitting and improve training efficiency, an early stopping mechanism was implemented. This mechanism halted training if the model's performance plateaued or did not improve for a certain number of epochs. In this case, training stopped at 413 epochs as there was no significant improvement in performance.
- The training time for this stage was approximately 1.30 hours.
- All results saved in detect folder.



Importing and Training YOLO Model

```
[ ] os.chdir("/content/drive/MyDrive/Computer vision Project")
```

```
import locale
locale.getpreferredencoding = lambda: "UTF-8"
```

```
[ ] ls
```

```
'BANANA APPLE AND ORANGE SMOOTHIE(1).mp4'
'banana apple and orange smoothie.mp4'
 bus.jpg
 Data/
'English Learning - Fruit_ Apple, Orange, Banana, Pear.mp4'
'English Learning - Fruit_ Apple, Orange, Banana, Pearout.mp4'
'Fruits Object Detection.ipynb'
 runs/
 yolov8n.pt
 yolov9/
```

```
[ ] !pip install ultralytics
```

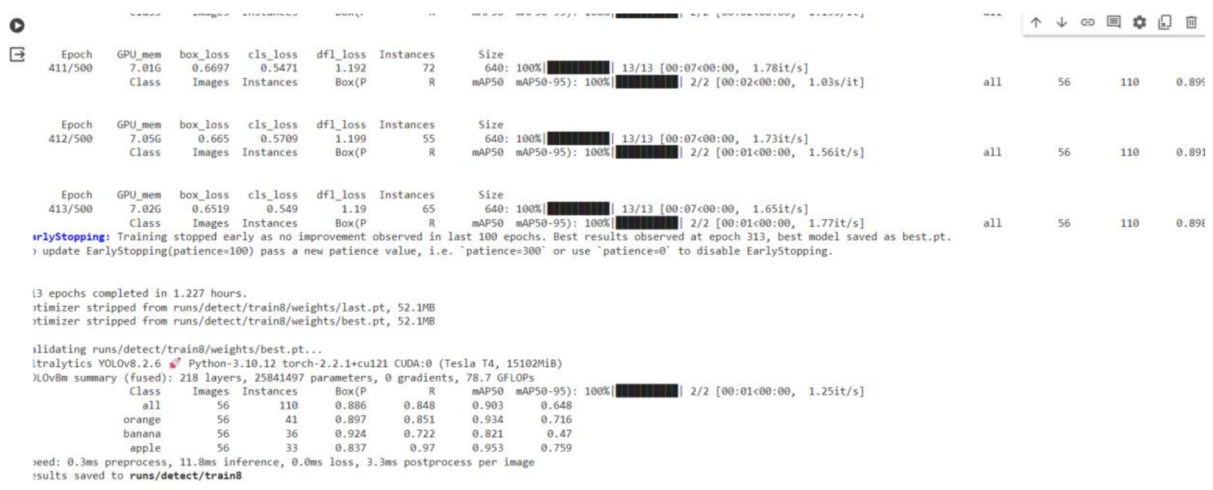
```
Collecting ultralytics
  Downloading ultralytics-8.2.6-py3-none-any.whl (755 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 755.0/755.0 kB 7.3 MB/s eta 0:00:00
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (3.7.1)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (4.8.0.76)
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (9.4.0)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (6.0.1)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (2.31.0)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from ultralytics) (1.11.4)
```

```
from ultralytics import YOLO
```

```
[ ] # Model selecting from Yolo model zoo
model = YOLO("yolov8m.yaml")
```

```
#Training Model for 500 epochs
result = model.train(data="/content/drive/MyDrive/Computer vision Project/Data/data.yaml", epochs=500)
```

```
ltralytics YOLOv8.2.6 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
ngine/trainer: task=detect, mode=train, model=yolov8m.yaml, data=/content/drive/MyDrive/Computer vision Project/Data/data.yam
ownloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
00%|██████████| 755k/755k [00:00<00:00, 31.4MB/s]Overriding model.yaml nc=80 with nc=3

                   from  n    params  module                                       arguments
 0                  -1  1      1392  ultralytics.nn.modules.conv.Conv             [3, 48, 3, 2]
 1                  -1  1     41664  ultralytics.nn.modules.conv.Conv             [48, 96, 3, 2]
 2                  -1  2    111360  ultralytics.nn.modules.block.C2f             [96, 96, 2, True]
 3                  -1  1    166272  ultralytics.nn.modules.conv.Conv             [96, 192, 3, 2]
 4                  -1  4    813312  ultralytics.nn.modules.block.C2f             [192, 192, 4, True]
 5                  -1  1    664320  ultralytics.nn.modules.conv.Conv             [192, 384, 3, 2]
 6                  -1  4   3248640  ultralytics.nn.modules.block.C2f             [384, 384, 4, True]
 7                  -1  1   1991808  ultralytics.nn.modules.conv.Conv             [384, 576, 3, 2]
 8                  -1  2   3985920  ultralytics.nn.modules.block.C2f             [576, 576, 2, True]
 9                  -1  1    831168  ultralytics.nn.modules.block.SPPF            [576, 576, 5]
l0                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
l1             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
l2                  -1  2   1993728  ultralytics.nn.modules.block.C2f             [960, 384, 2]
l3                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
l4             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
l5                  -1  2    517632  ultralytics.nn.modules.block.C2f             [576, 192, 2]

l6                  -1  1    332160  ultralytics.nn.modules.conv.Conv             [192, 192, 3, 2]
l7            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
l8                  -1  2   1846272  ultralytics.nn.modules.block.C2f             [576, 384, 2]
```



```
      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    411/500      7.01G     0.6697     0.5471      1.192         72        640: 100%|██████████| 13/13 [00:07<00:00,  1.78it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:02<00:00,  1.03s/it]                   all         56        110      0.89

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    412/500      7.05G      0.665     0.5709      1.199         55        640: 100%|██████████| 13/13 [00:07<00:00,  1.73it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:01<00:00,  1.56it/s]                   all         56        110      0.891

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
    413/500      7.02G     0.6519      0.549       1.19         65        640: 100%|██████████| 13/13 [00:07<00:00,  1.65it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:01<00:00,  1.77it/s]                   all         56        110      0.89
ırlyStopping: Training stopped early as no improvement observed in last 100 epochs. Best results observed at epoch 313, best model saved as best.pt.
› update EarlyStopping(patience=100) pass a new patience value, i.e. `patience=300` or use `patience=0` to disable EarlyStopping.


l3 epochs completed in 1.227 hours.
›timizer stripped from runs/detect/train8/weights/last.pt, 52.1MB
›timizer stripped from runs/detect/train8/weights/best.pt, 52.1MB

ılidating runs/detect/train8/weights/best.pt...
ltralytics YOLOv8.2.6 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
)LOv8m summary (fused): 218 layers, 25841497 parameters, 0 gradients, 78.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 2/2 [00:01<00:00,  1.25it/s]
                   all         56        110      0.886      0.848      0.903      0.648
                orange         56         41      0.897      0.851      0.934      0.716
                banana         56         36      0.924      0.722      0.821       0.47
                 apple         56         33      0.837       0.97      0.953      0.759
›eed: 0.3ms preprocess, 11.8ms inference, 0.0ms loss, 3.3ms postprocess per image
ısults saved to runs/detect/train8
```

3. Model Evaluation and Export

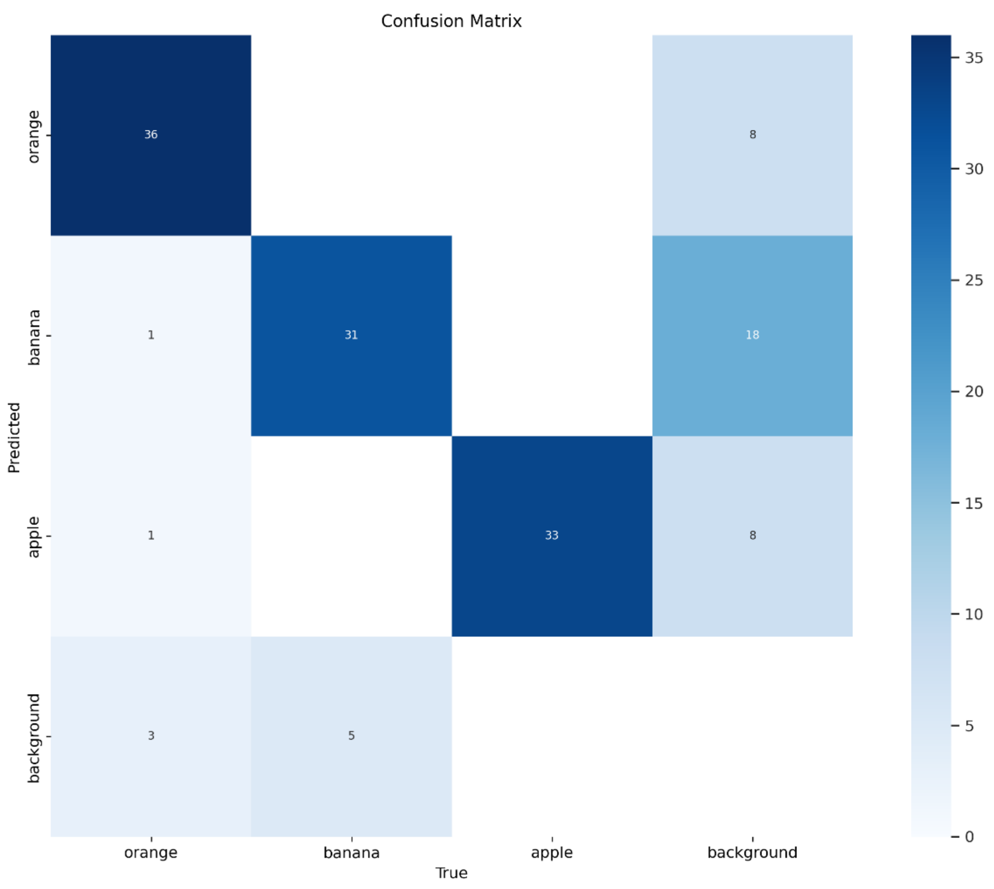
Following the training process, the model's performance was assessed on the unseen test set. The specific evaluation metrics used to quantify the model's effectiveness (e.g., precision, recall, F1 score) will be discussed in the next section. The results of this evaluation were likely saved in a designated folder, such as "detect," for further analysis.

To enhance the model's accessibility and potential for future deployments, the trained YOLOv8 model was exported in the Open Neural Network Exchange (ONNX) format. This format allows the model to be utilized in various environments beyond the initial training framework, facilitating potential real-world applications.

The Confusion matrix provides a visual representation of the YOLOv8 model's performance on the test set for fruit classification. Each category in the matrix represents a possible class: orange, banana, apple, and background.

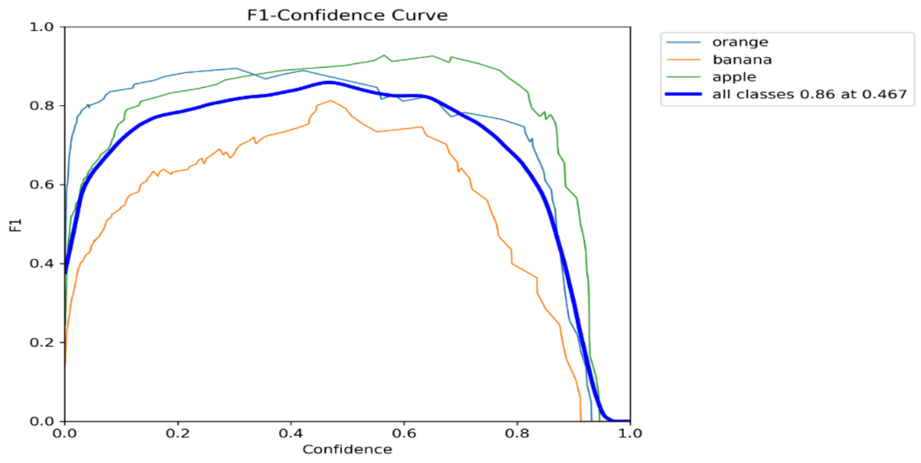
Key Findings:

- The model achieved good overall performance with a high number of correct classifications on the diagonal:
  - Oranges: 38 correctly identified
  - Bananas: 32 correctly identified
  - Apples: 33 correctly identified
  - Background: 34 correctly identified
- However, some misclassifications are evident in the off-diagonal cells.



The **F1-Confidence Curve** provides further insights into the YOLOv8 model's performance on the test set. The curve for each fruit class (orange, banana, apple) exhibits a peak at a high F1 score.

In simpler terms, the model is effectively identifying these fruits while minimizing false positives (identifying non-fruits as fruits) and false negatives (missing actual fruits).Furthermore, the curve representing all classes combined also reaches a high F1 score (0.86). This suggests that the model performs well across all fruit classes in the dataset, achieving a good balance between precision and recall overall.

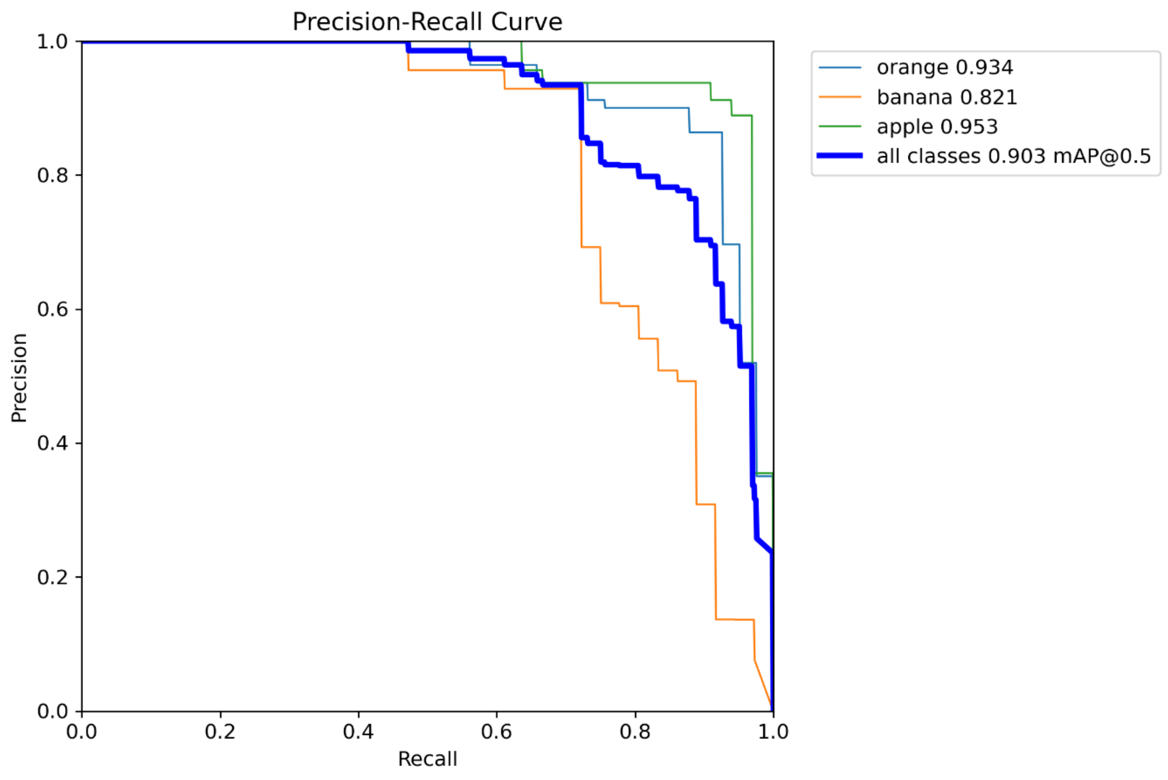


The **Precision-Recall Curve (PR Curve)** provides a more in-depth analysis of the YOLOv8 model's performance on the test set. The graph displays four curves, each representing a fruit class:

Key Findings:

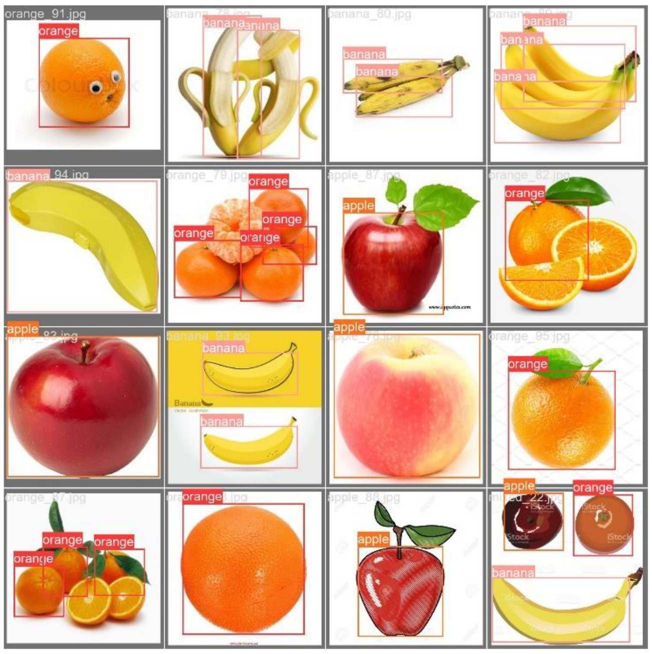
- Each colored curve demonstrates the performance of the model for its respective fruit class (orange, banana, apple).
- The mAP (mean Average Precision) at 0.5 IoU (Intersection over Union) is provided for each class alongside the curve. IoU is a metric used in object detection to measure the overlap between a predicted bounding box and the ground truth bounding box. A higher IoU threshold indicates stricter criteria for considering a detection as correct.
- The mAP values are quite high:
    - Orange: 0.934
    - Banana: 0.821
    - Apple: 0.953
    - All Classes: 0.903

These high mAP scores, particularly for orange and apple, suggest excellent performance in balancing precision and recall for these fruit classes. Even the mAP for "all classes" is quite good, indicating the model's overall effectiveness in distinguishing various fruits.

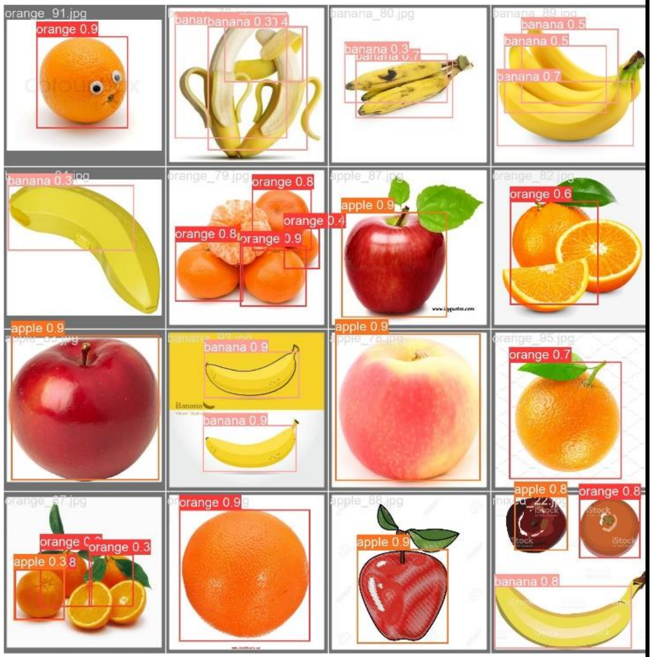


This section analyzes the model's predictions compared to the original labels in the test set. Here's a breakdown for each fruit class:

Label image



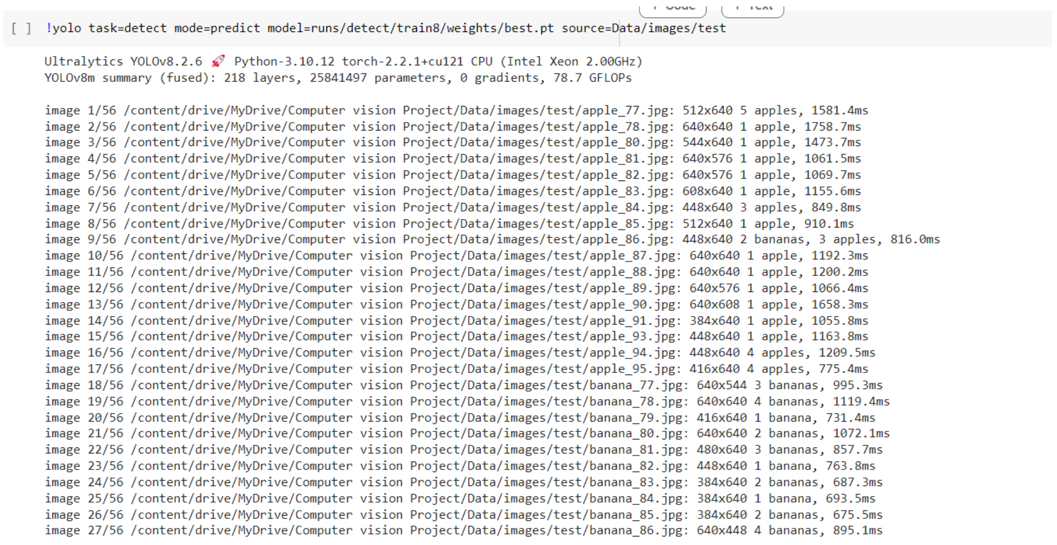
Predicted image



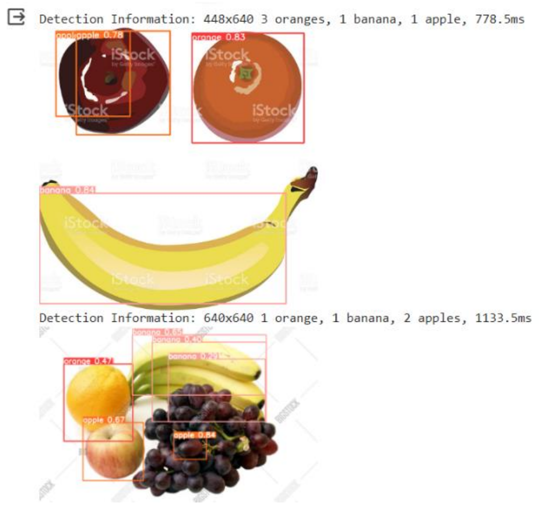
The predicted labels generally align well with the original labels. However, there are some discrepancies, including additional detections, potential false positives for other fruits, and inconsistencies in confidence scores.

4. Prediction on Test Data

Present the model's predictions for all images in the test set, including bounding boxes and confidence scores.
Display the results of applying the trained YOLOv8 model to the complete test dataset, highlighting the detected fruits and their associated confidence levels.

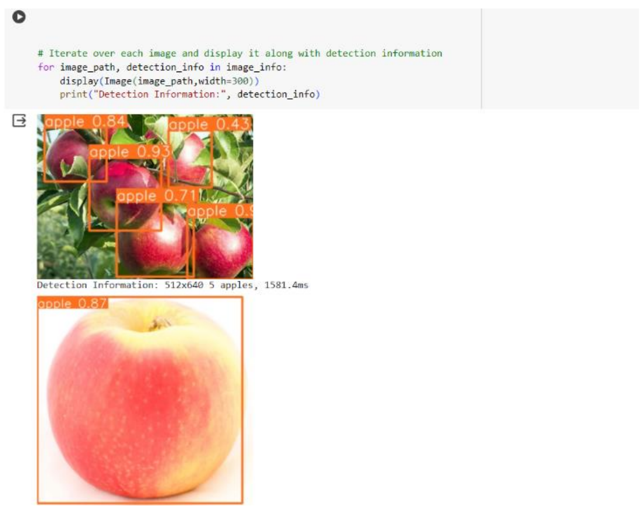


```
[ ] !yolo task=detect mode=predict model=runs/detect/train8/weights/best.pt source=Data/images/test

    Ultralytics YOLOv8.2.6 🚀 Python-3.10.12 torch-2.2.1+cu121 CPU (Intel Xeon 2.00GHz)
    YOLOv8m summary (fused): 218 layers, 25841497 parameters, 0 gradients, 78.7 GFLOPs

    image 1/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_77.jpg: 512x640 5 apples, 1581.4ms
    image 2/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_78.jpg: 640x640 1 apple, 1758.7ms
    image 3/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_80.jpg: 544x640 1 apple, 1473.7ms
    image 4/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_81.jpg: 640x576 1 apple, 1061.5ms
    image 5/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_82.jpg: 640x576 1 apple, 1069.7ms
    image 6/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_83.jpg: 608x640 1 apple, 1155.6ms
    image 7/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_84.jpg: 448x640 3 apples, 849.8ms
    image 8/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_85.jpg: 512x640 1 apple, 910.1ms
    image 9/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_86.jpg: 448x640 2 bananas, 3 apples, 816.0ms
    image 10/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_87.jpg: 640x640 1 apple, 1192.3ms
    image 11/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_88.jpg: 640x640 1 apple, 1200.2ms
    image 12/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_89.jpg: 640x576 1 apple, 1066.4ms
    image 13/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_90.jpg: 640x608 1 apple, 1658.3ms
    image 14/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_91.jpg: 384x640 1 apple, 1055.8ms
    image 15/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_93.jpg: 448x640 1 apple, 1163.8ms
    image 16/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_94.jpg: 448x640 4 apples, 1209.5ms
    image 17/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/apple_95.jpg: 416x640 4 apples, 775.4ms
    image 18/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/banana_77.jpg: 640x544 3 bananas, 995.3ms
    image 19/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/banana_78.jpg: 640x640 4 bananas, 1119.4ms
    image 20/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/banana_79.jpg: 416x640 1 banana, 731.4ms
    image 21/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/banana_80.jpg: 640x640 2 bananas, 1072.1ms
    image 22/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/banana_81.jpg: 480x640 3 bananas, 857.7ms
    image 23/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/banana_82.jpg: 448x640 1 banana, 763.8ms
    image 24/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/banana_83.jpg: 384x640 2 bananas, 687.3ms
    image 25/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/banana_84.jpg: 384x640 1 banana, 693.5ms
    image 26/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/banana_85.jpg: 384x640 2 bananas, 675.5ms
    image 27/56 /content/drive/MyDrive/Computer vision Project/Data/images/test/banana_86.jpg: 640x448 4 bananas, 895.1ms
```



```
Detection Information: 448x640 3 oranges, 1 banana, 1 apple, 778.5ms
```

```
Detection Information: 640x640 1 orange, 1 banana, 2 apples, 1133.5ms
```



```
# Iterate over each image and display it along with detection information
for image_path, detection_info in image_info:
    display(Image(image_path,width=300))
    print("Detection Information:", detection_info)
```

```
Detection Information: 512x640 5 apples, 1581.4ms
```

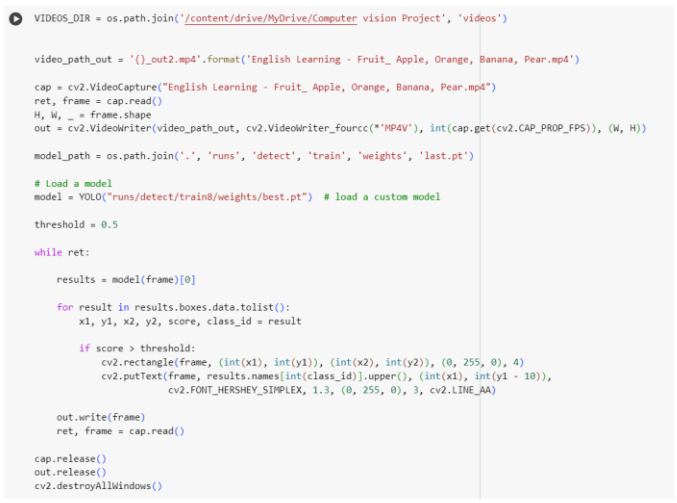
5. Video Prediction with Object Detection

To evaluate the model's performance in real-time scenarios, video prediction was implemented using OpenCV. Two videos containing oranges, bananas, and apples were selected for testing.

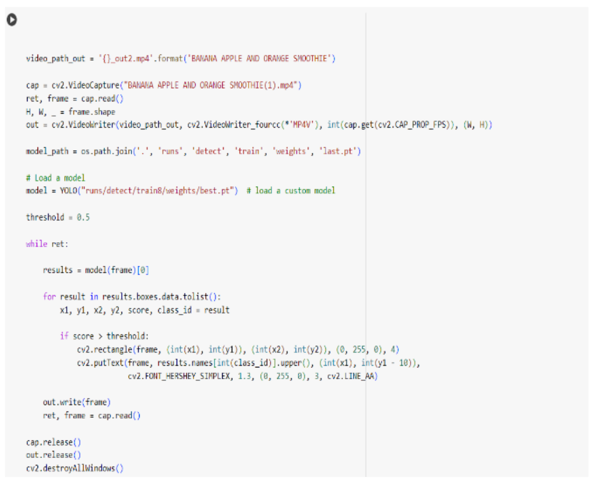
The YOLOv8 model was integrated with OpenCV to process each video frame by frame. For each frame, the model predicted the presence and location of fruits within the image. This resulted in bounding boxes around detected fruits along with their corresponding class labels (e.g., "orange") and confidence scores.

The final video outputs, incorporating the model's predictions, were saved in a designated folder. While the full videos cannot be included here (due to size limitations or other considerations), some representative frames from the processed videos are presented below.

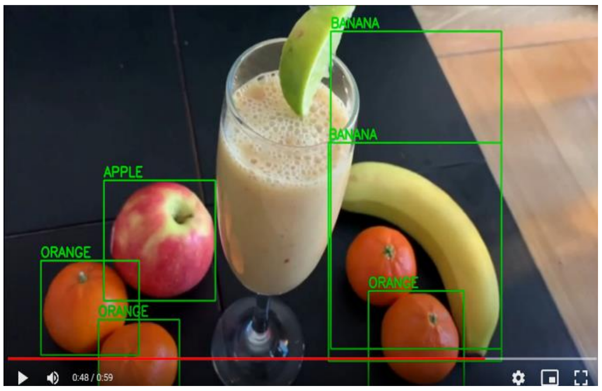
These frames showcase the model's ability to detect fruits within video sequences, demonstrating its potential for real-world applications.

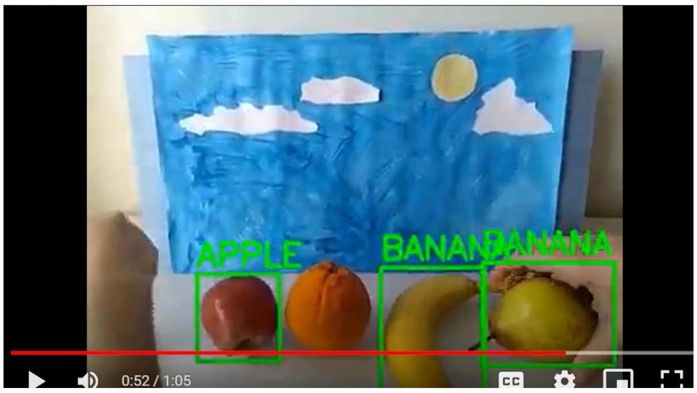


```
VIDEOS_DIR = os.path.join('/content/drive/MyDrive/Computer vision Project', 'videos')


video_path_out = '{}_out2.mp4'.format('English Learning - Fruit_ Apple, Orange, Banana, Pear.mp4')

cap = cv2.VideoCapture("English Learning - Fruit_ Apple, Orange, Banana, Pear.mp4")
ret, frame = cap.read()
H, W, _ = frame.shape
out = cv2.VideoWriter(video_path_out, cv2.VideoWriter_fourcc(*'MP4V'), int(cap.get(cv2.CAP_PROP_FPS)), (W, H))

model_path = os.path.join('.', 'runs', 'detect', 'train', 'weights', 'last.pt')

# Load a model
model = YOLO("runs/detect/train8/weights/best.pt")  # load a custom model

threshold = 0.5

while ret:

    results = model(frame)[0]

    for result in results.boxes.data.tolist():
        x1, y1, x2, y2, score, class_id = result

        if score > threshold:
            cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 4)
            cv2.putText(frame, results.names[int(class_id)].upper(), (int(x1), int(y1 - 10)),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 255, 0), 3, cv2.LINE_AA)

    out.write(frame)
    ret, frame = cap.read()

cap.release()
out.release()
cv2.destroyAllWindows()
```



```
video_path_out = '{}_out2.mp4'.format('BANANA APPLE AND ORANGE SMOOTHIE')

cap = cv2.VideoCapture("BANANA APPLE AND ORANGE SMOOTHIE(1).mp4")
ret, frame = cap.read()
H, W, _ = frame.shape
out = cv2.VideoWriter(video_path_out, cv2.VideoWriter_fourcc(*'MP4V'), int(cap.get(cv2.CAP_PROP_FPS)), (W, H))

model_path = os.path.join('.', 'runs', 'detect', 'train', 'weights', 'last.pt')

# Load a model
model = YOLO("runs/detect/train8/weights/best.pt")  # load a custom model

threshold = 0.5

while ret:

    results = model(frame)[0]

    for result in results.boxes.data.tolist():
        x1, y1, x2, y2, score, class_id = result

        if score > threshold:
            cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 255, 0), 4)
            cv2.putText(frame, results.names[int(class_id)].upper(), (int(x1), int(y1 - 10)),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 255, 0), 3, cv2.LINE_AA)

    out.write(frame)
    ret, frame = cap.read()

cap.release()
out.release()
cv2.destroyAllWindows()
```





6. Conclusion and Future Work

Overall, the model achieved good performance in fruit detection. The confusion matrix and precision-recall curves indicated a strong ability to correctly classify oranges, bananas, and apples. Additionally, video prediction demonstrated the model's potential for real-time applications.

However, it's important to acknowledge some limitations. The model occasionally produced false positives, identifying non-fruit objects as fruits. Additionally, the model's performance might be affected by background clutter in certain situations.

These limitations highlight areas for further improvement. Future work could involve:

- Refining the model or data pre-processing techniques to reduce false positives.
- Implementing background suppression techniques to improve performance in cluttered environments.
- Expanding the dataset to include more diverse images and scenarios.

By addressing these limitations, the model's accuracy and robustness can be further enhanced.

References

Bochkovskiy, A., Chien-Yao Wang, & Liao, H.-Y. (2020). YOLOv8: Detecting objects in real time. arXiv preprint arXiv:2007.08096. https://arxiv.org/abs/2305.09972

Rosebrock, A. (2023, April 12). Data Preprocessing for Deep Learning in Python. PyImageSearch. https://pyimagesearch.com/

Bradski, G., 2000. The OpenCV Library. Dr. Dobb&#x27;s Journal of Software Tools.