Gujarat Technological University

Ahmadabad

B and B Institute of Technology, V.V. Nagar

Computer Engineering Department (GIA)

A Micro Project Report on

"Object Detection & Classification"

Under the Course

Introduction To Machine

Learning– (4350702) Submitted

By

ENROLLMENT NO.	NAME
226040307115	Patel Dhanya Ripankumar
226040307058	Kapadia Aditya Divyang

Academic Year
2024- 25

Mr. Hitesh A. Patel
Course Coordinator

Mr. Jiten P. Parmar
Head of the Department

Department of Computer Engineering

Bhailalbhai and Bhikhabhai
Institute of Technology

VALLABH VIDYA NAGAR -
388120 GUJARAT, INDIA
PHONE: 02692-237240

(Affiliated to Gujarat Technological University, Gujarat)

CERTIFICATE

This is to certify that

ENROLLMENT NO.	NAME
226040307115	Patel Dhanya Ripankumar
226040307058	Kapadia Aditya Divyang

of 5th semester Computer Engineering department, have satisfactorily completed his / her Micro Project in Introduction to Machine Learning (4350702) for the term from 27/06/2024 to 25/10/2024.

Date of Submission:

Mr. Hitesh A. Patel
(Course Coordinator)

Mr. Jiten P. Parmar
(I/C Head)

Index

1 INTRODUCTION

1.1 INTRODUCTION TO OBJECT DETECTION & CLASSIFICATION

- Object Detection: Object detection is a computer vision technique that involves identifying instances of objects from specific categories (like cars, people, animals, etc.) in images or videos. Unlike image classification, which merely assigns a label to an entire image, object detection goes a step further by identifying the location of objects within the image. This is typically achieved by drawing bounding boxes around the detected objects.
- Object detection models perform two key tasks:
  1. Localization: Determining the position of objects in the image by drawing bounding boxes.
  2. Classification: Assigning a label to the object inside each bounding box, determining which class the object belongs to (e.g., person, vehicle, cat, etc.).
- Common algorithms for object detection include:
  1. YOLO (You Only Look Once): A real-time object detection system that divides the image into a grid and predicts bounding boxes and class probabilities.
  2. Faster R-CNN: A two-stage object detection model that first proposes regions of interest and then classifies objects within those regions.
  3. SSD (Single Shot Detector): A single-stage detector that simplifies object detection by predicting both bounding boxes and classes directly from the image.
- Object Classification: Object classification refers to the task of identifying the object’s category from a set of predefined categories within an image. In this case, the entire image is analyzed, and a label (class) is assigned based on the most prominent object or feature in the image.
- In practice, classification is simpler than detection since it involves categorizing the entire image as a whole rather than localizing multiple objects. Convolutional Neural Networks (CNNs) are commonly used for classification tasks due to their ability to recognize complex features and patterns.

1.2 PROJECT OVERVIEW: OBJECT DETECTION AND CLASSIFICATION

- Object detection and classification are key areas within the field of computer vision, a branch of machine learning that focuses on enabling machines to interpret and make decisions based on visual data. The goal of this project is to develop a model capable of accurately detecting and classifying objects within images or video streams, allowing for real-time processing and recognition of multiple objects in a given scene.

- In this project, the following tasks will be performed:

  - **Object Detection**: The model will be trained to locate objects within an image or video frame. This involves not only identifying the objects present but also drawing bounding boxes around them to determine their exact position. Object detection is a critical step for tasks such as surveillance, autonomous driving, and robotics.

    - **Object Classification**: After detecting the objects, the model will classify each one into a predefined category (e.g., person, car, animal). This classification is based on the visual characteristics of the object, and it ensures that the system can distinguish between different types of objects present in a scene.

- Key Components of the Project:
    - **Data Collection and Preprocessing**: The project will begin by gathering a dataset containing labeled images that include various objects. This dataset will be used to train the detection and classification model. Preprocessing steps like resizing, normalization, and augmentation will ensure the model can generalize well to new data.

    - **Model Design and Training**: The object detection and classification model will be built using deep learning architectures such as Convolutional Neural Networks (CNNs) or pre-trained models like YOLO (You Only Look Once) or Faster R-CNN. These models are well-suited for image-related tasks due to their ability to recognize complex patterns and features.

    - **Performance Optimization**: One of the key challenges in object detection is ensuring both accuracy and speed. The project will explore ways to improve detection speed while maintaining or enhancing classification accuracy, especially for real-time applications such as video analysis.

    - **Evaluation and Testing**: The model's performance will be evaluated using standard metrics such as precision, recall, and mean Average Precision (mAP). Additionally, the system will be tested on various datasets and real-world scenarios to ensure it performs robustly under different conditions (e.g., varying object sizes, occlusion, lighting).

- Applications:
    - This project has wide-ranging applications in fields such as:

        - **Autonomous Vehicles**: For detecting pedestrians, other vehicles, traffic signs, and obstacles in real-time.
        - **Surveillance Systems**: For identifying and tracking individuals or objects of interest in live video feeds.
        - **Healthcare**: For detecting medical anomalies like tumors or lesions in radiology images.
        - **Retail Automation**: For inventory management and smart checkout systems using automated object detection.

- This project will leverage state-of-the-art machine learning techniques to solve real-world problems involving visual data analysis, providing a scalable solution for object detection and classification tasks.

2 Tool, Platforms and Languages

2.1 Hardware Requirements

- To successfully implement object detection and classification using machine learning, certain hardware resources are essential, especially for handling the computationally intensive tasks of training and running deep learning models. Below is a list of recommended hardware components for the project:
- Processor (CPU):
  1. Recommended: Intel Core i7 or AMD Ryzen 7 and above.
  2. Minimum: Intel Core i5 or AMD Ryzen 5.
  3. The CPU is essential for general computing tasks, and while training models can be CPU-intensive, modern machine learning tasks typically benefit more from powerful GPUs.
- Graphics Processing Unit (GPU):
  1. Recommended: NVIDIA GPU with CUDA support, such as:
  2. NVIDIA RTX 3060 / 3070 / 3080 / 3090 for high performance.
  3. NVIDIA Tesla V100 / A100 for enterprise-level performance.
  4. Minimum: NVIDIA GTX 1060 or above (with at least 4 GB VRAM).
  5. The GPU is crucial for accelerating deep learning computations, especially for training convolutional neural networks (CNNs) used in object detection and classification tasks.
- RAM (Memory):
  1. Recommended: 16 GB to 32 GB DDR4 RAM.
  2. Minimum: 8 GB DDR4 RAM.
  3. Ample RAM ensures smooth performance, especially when working with large datasets, running training sessions, or multitasking.
- Storage:
  1. Recommended:512 GB to 1 TB SSD (Solid-State Drive) for fast read/write speeds and storing frequently accessed datasets and models.
  2. Minimum:256 GB SSD for essential storage needs, though larger datasets may require external storage.
  3. Additional Option: 1 TB HDD for additional storage capacity for datasets and backups.
- Power Supply Unit (PSU):
  1. Depending on the GPU and other components, ensure your PSU has enough wattage to support the hardware.

2. Recommended: 650W to 750W for systems with mid-to-high-end GPUs.
3. Minimum: 500W for standard configurations.

- Display:
  1. Recommended: A full HD (1920x1080) or higher resolution monitor for clear visualization of images and results.
  2. For tasks like monitoring object detection outputs and evaluating performance, a good display can be beneficial.
- Cooling System:
  1. High-performance hardware, especially GPUs, generates significant heat during training sessions.
  2. Recommended: A good cooling system (liquid cooling or high-performance fans) to maintain optimal system temperatures.
- External Storage/Backup:
  1. Recommended: External HDD or cloud storage (e.g., Google Drive, AWS, or Azure) for storing large datasets or backup copies of trained models.
- Additional Peripherals:
  1. Keyboard and Mouse: Standard input devices.
  2. Power Backup (UPS): For ensuring power stability during long training sessions.

2.2 SOFTWARE REQUIREMENTS

- For developing and deploying the object detection and classification project, a range of software tools and frameworks are necessary. These include the operating system, programming language, machine learning libraries, and optional development environments. Below is a detailed breakdown of the required and optional software components:
  1. Operating System:
     - Windows (Windows 10 or newer): A widely supported operating system for machine learning, especially for those who prefer a GUI-based development environment.
     - Linux (Ubuntu 20.04 or newer): Highly recommended for machine learning and deep learning tasks, offering better compatibility with libraries such as PyTorch and TensorFlow, especially for GPU usage.
     - macOS (macOS Mojave or newer): Suitable for development, though support for GPU acceleration (NVIDIA CUDA) is limited.
  2. Python:
     - Version: Python 3.7 or higher.
     - Python is the primary language used for building machine learning models. It is preferred due to its extensive ecosystem of libraries and ease of use.
     - It is recommended to install Python via the Anaconda Distribution to

simplify package management and virtual environments.

3. PyTorch:
   - Version: Compatible version (check specific requirements in the Ultralytics YOLOv8 documentation).
   - PyTorch is a popular open-source deep learning framework used for training and deploying machine learning models. It provides flexibility, dynamic computation graphs, and is well-suited for research and production use cases.
   - PyTorch supports CUDA for GPU acceleration, allowing faster training of models like YOLO (You Only Look Once).
   -
4. CUDA Toolkit (Optional, for GPU Acceleration):
   - Version: CUDA 11.x or newer.
   - The CUDA Toolkit is necessary for leveraging NVIDIA GPUs to accelerate deep learning tasks. GPU acceleration drastically reduces training times and improves performance, especially for complex models like YOLOv8.
   - CUDA should be installed alongside compatible NVIDIA GPU drivers to ensure proper functionality.
   - cuDNN (NVIDIA CUDA Deep Neural Network Library):
   - Version: cuDNN 8.x or newer.
   - cuDNN is a GPU-accelerated library specifically designed for deep neural networks. It works in conjunction with CUDA to optimize performance for tasks like training and inference in convolutional neural networks (CNNs).

5. YOLOv8 (Ultralytics):
   - YOLOv8 is the latest version of the "You Only Look Once" family of object detection algorithms. It is highly efficient and capable of real-time object detection.
   - YOLOv8 provides a ready-to-use implementation of object detection and classification with pre-trained models that can be fine-tuned for specific applications.

6. OpenCV (Optional for Image Processing):
   - Version: OpenCV 4.x or newer.
   - OpenCV is an open-source library used for image and video processing. It allows developers to handle tasks like image resizing, transformation, and visualization, which are helpful when working with object detection models.

   Integrated Development Environment (IDE) (Optional):

7. PyCharm:
   - PyCharm is a powerful Python IDE that offers intelligent code completion, on-the-fly error checking, and advanced debugging.
   - It is well-suited for machine learning projects because of its support for Python virtual environments and integration with tools like Jupiter notebooks.
   - Available in both a community (free) and Professional (paid) version.

2.3 Details of Technology used

- This section provides a detailed overview of the technologies employed in the project, focusing on object detection and classification using modern machine learning and deep learning techniques. Each technology is crucial in building, training, and deploying the model efficiently.
  1. Python Programming Language:
     - Technology: Python is the primary programming language used in this project.
     - Details: Python is known for its simplicity and flexibility, making it the preferred language for machine learning and artificial intelligence (AI) applications. It has a rich ecosystem of libraries and frameworks such as PyTorch, TensorFlow, and OpenCV, which are essential for building machine learning models. Python's clear syntax and dynamic typing allow rapid prototyping, easy code maintenance, and testing.
     - Usage in the Project: Python is used to implement the object detection and classification algorithms, process data, and manage the workflow of the model's training and evaluation.
  2. PyTorch Deep Learning Framework:
     - Technology: PyTorch is an open-source deep learning framework developed by Facebook's AI Research lab.
     - Details: PyTorch offers dynamic computation graphs (also known as define-by-run), which makes it easy to adjust models during runtime and allows more flexibility compared to static graph-based frameworks like TensorFlow. It is widely used for research and production environments due to its ease of use and debugging capabilities. PyTorch also supports automatic differentiation, which simplifies the training of deep learning models.
     - Usage in the Project: PyTorch is used to build, train, and fine-tune the object detection and classification models, particularly YOLOv8 (You Only Look Once version 8). It also integrates with GPU acceleration via CUDA for faster computations.
  3. YOLOv8 (You Only Look Once):
     - Technology: YOLOv8 is the latest version of the YOLO family of models for real-time object detection.
     - Details: YOLO (You Only Look Once) is a state-of-the-art, real-time object detection system that processes images in a single forward pass of the neural network, allowing it to detect objects with high speed and accuracy. YOLOv8 improves upon previous versions with enhanced performance, better precision, and faster training and inference times. It is particularly effective in tasks that require detecting multiple objects in images or video streams.
     - Usage in the Project: YOLOv8 is employed to detect and classify objects within images. It is a pre-trained model that can be fine-tuned with custom datasets to improve detection accuracy for specific use cases.
  4. CUDA (Compute Unified Device Architecture):
     - Technology: CUDA is a parallel computing platform and API model created by NVIDIA.
     - Details: CUDA enables developers to utilize the processing power of NVIDIA GPUs to accelerate the computation of complex tasks, such as training deep learning models. With CUDA, tasks like matrix

multiplication, convolution operations, and gradient computations can be run on the GPU, significantly speeding up model training.
   - Usage in the Project: CUDA is used to accelerate the training and inference processes of the YOLOv8 model, making use of the parallel processing power of NVIDIA GPUs. This results in faster training times and real-time detection capabilities.
5. cuDNN (CUDA Deep Neural Network library):
   - Technology: cuDNN is a GPU-accelerated library for deep neural networks.
   - Details: cuDNN is developed by NVIDIA to optimize deep learning performance on GPUs. It provides high-performance building blocks for deep neural network architectures, including convolution, pooling, normalization, and activation layers. It is designed to work with deep learning frameworks like PyTorch and TensorFlow to maximize the performance of models on NVIDIA GPUs.
   - Usage in the Project: cuDNN is used in conjunction with PyTorch and CUDA to optimize the performance of convolutional neural networks (CNNs) used in object detection and classification.
6. OpenCV (Open Source Computer Vision Library):
   - Technology: OpenCV is an open-source library designed for real-time computer vision tasks.
   - Details: OpenCV provides a wide range of functionalities for image processing, such as reading, manipulating, and analyzing images and videos. It is widely used in tasks like object detection, face recognition, and motion tracking. OpenCV integrates well with deep learning frameworks and supports working with images and video feeds in real-time.
   - Usage in the Project: OpenCV is used to preprocess input images for object detection and classification. It also facilitates visualization of the model’s output, displaying detected objects with bounding boxes and labels on images or video streams.
7. Jupyter Notebook:
   - Technology: Jupyter Notebook is an open-source web application for creating and sharing live code, equations, visualizations, and narrative text.
   - Details: Jupyter is widely used in the data science and machine learning community for exploratory data analysis, model training, and result visualization. It provides an interactive interface where developers can run code in small chunks and immediately see the results, making it easier to experiment with models, visualize outputs, and document the process.
   - Usage in the Project: Jupyter Notebooks are used to develop and test the object detection and classification model. They allow for easy visualization of the detection results, tuning of hyperparameters, and analysis of the model's performance.
8. NumPy:
   - Technology: NumPy is a fundamental package for scientific computing with Python.
   - Details: NumPy provides support for multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. It is highly optimized for performance and is a key

  component of many machine learning workflows.
- Usage in the Project: NumPy is used for handling and manipulating image data, as well as for numerical computations needed during model training and inference.

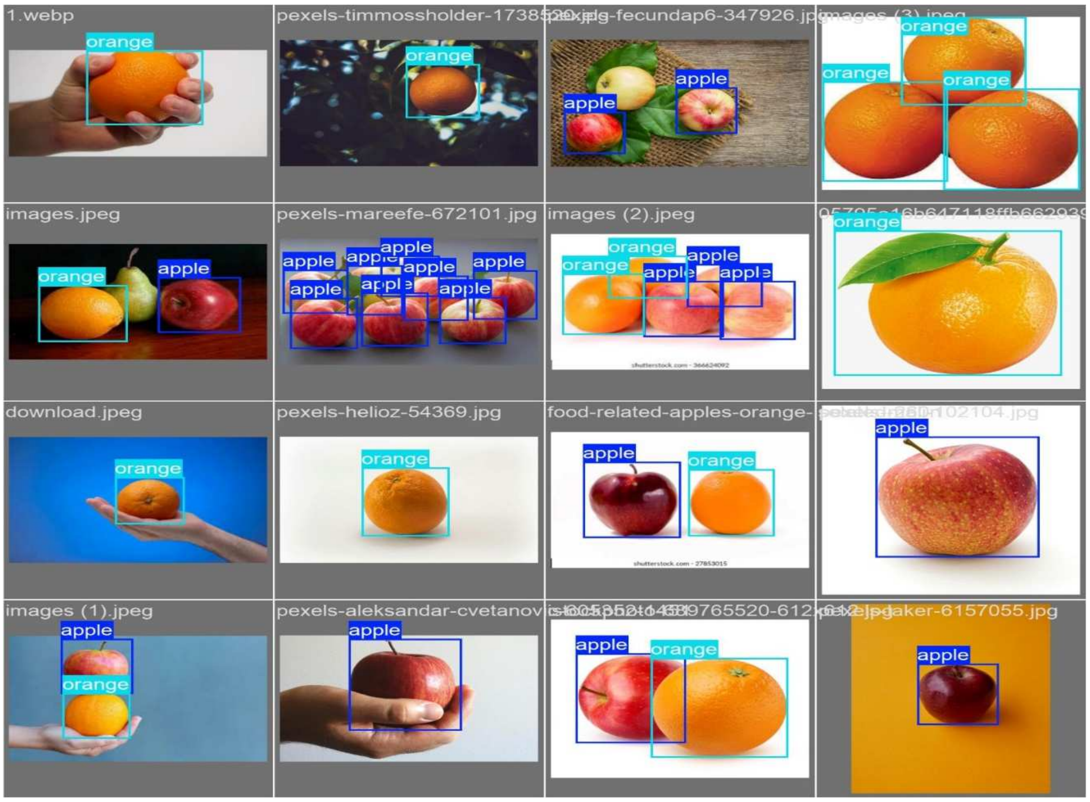
9. Pandas:
   - Technology: Pandas is a Python library for data manipulation and analysis.
   - Details: Pandas provides data structures like DataFrames that make it easier to manipulate and analyze structured data. It is widely used in machine learning projects for data preprocessing, cleaning, and exploration.
   - Usage in the Project: Pandas is used to handle datasets, particularly for loading and managing labeled images and metadata associated with object detection tasks.
10. Matplotlib & Seaborn:
    - Technology: Matplotlib and Seaborn are Python libraries for creating static, animated, and interactive visualizations.
    - Details: Matplotlib is the core plotting library, providing basic plotting capabilities, while Seaborn is built on top of Matplotlib and provides more aesthetically pleasing statistical plots.
    - Usage in the Project: Matplotlib and Seaborn are used to visualize the performance of the model, such as plotting training loss, accuracy metrics, and confusion matrices to evaluate the classification outcomes

3.Source Code

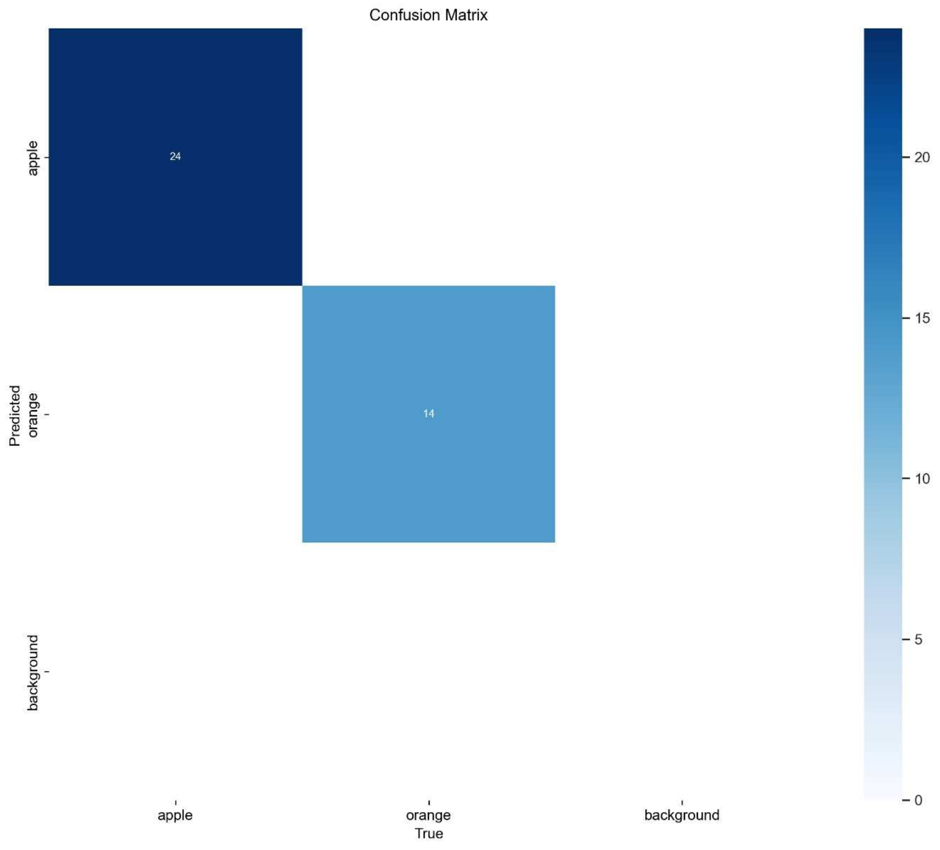
```
Config.yaml

path: D:\PycharmProjects\Object-Detection-
Yolo\YoloTrainingForBag\data
train: images\train # train images (relative to 'path')
val: images\train # val images (relative to 'path')

# Classes
names:
  0: CaproBag
```

```
Main.py

from ultralytics import YOLO

model = YOLO("yolov8l.yaml")  # Loading the YOLOv8 configuration (for
model structure)
model = YOLO("yolov8l.pt")    # Loading pre-trained weights (YOLOv8
large version)
if __name__ == '__main__':
    results = model.train(data="config.yaml", epochs=150)
    results = model.val()
    results = model("bag1.jpg")
    success = YOLO("yolov8l.pt").export(format="onnx")
```

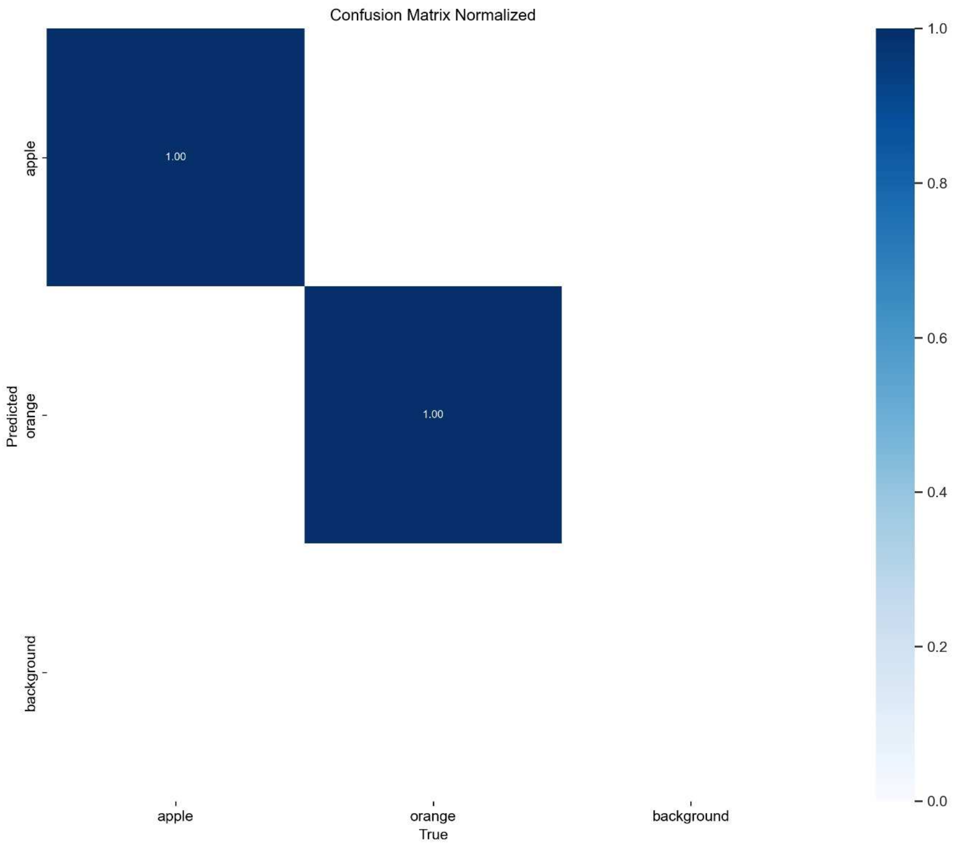
4. SnapShot

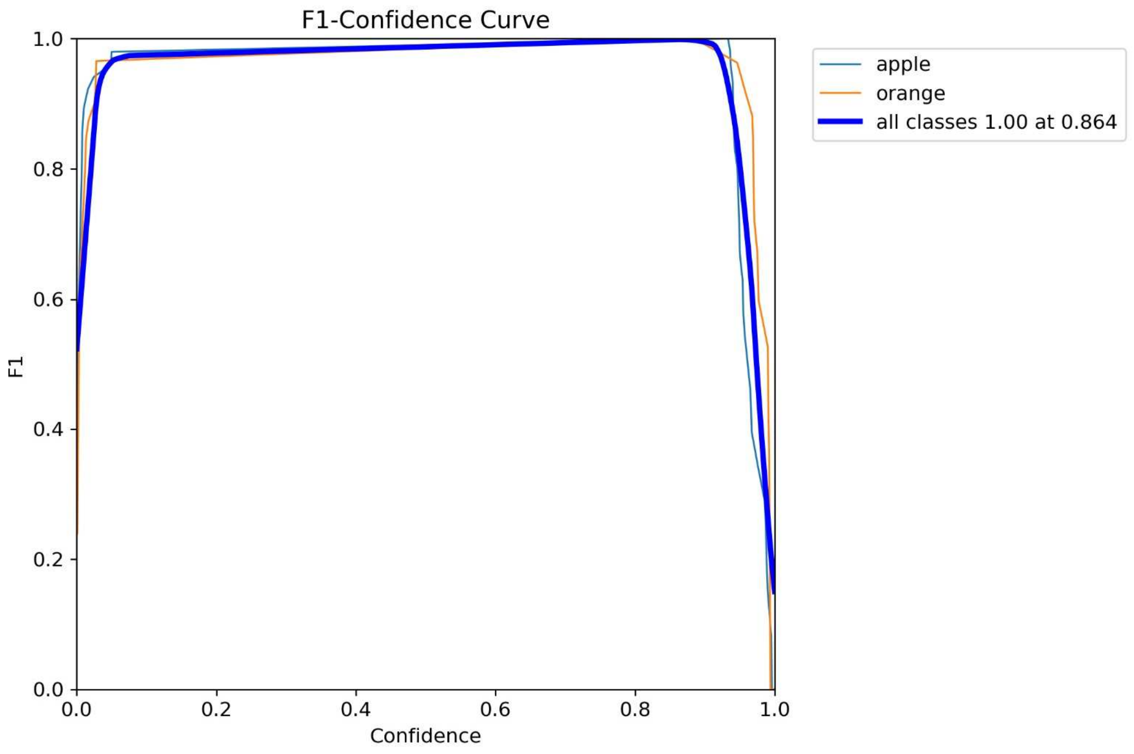


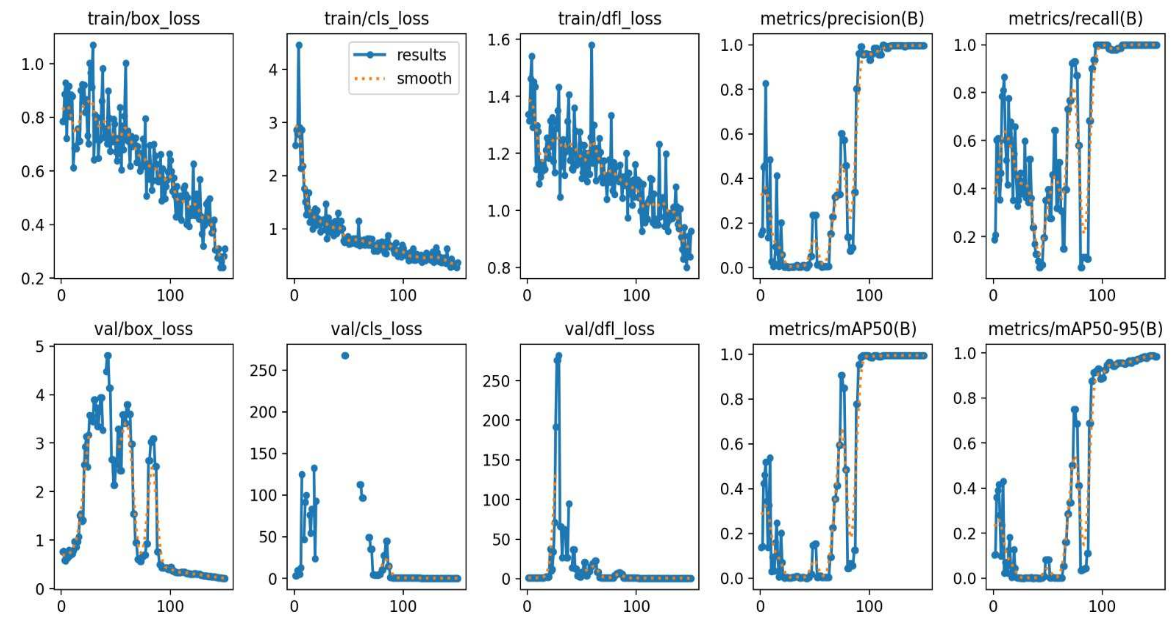
1.webp
orange
pexels-timmossholder-1738520.jpg
orange
pexels-fecundap6-347926.jpg
apple
apple
images (3).jpeg
orange
orange
orange
images.jpeg
orange
apple
pexels-mareefe-672101.jpg
apple
apple
apple
apple
apple
images (2).jpeg
orange
orange
apple
apple
apple
download.jpeg
orange
pexels-helioz-54369.jpg
orange
food-related-apples-orange-
apple
orange
images (1).jpeg
apple
orange
pexels-aleksandar-cvetanovic
apple
orange
pexels-jupdaker-6157055.jpg
apple

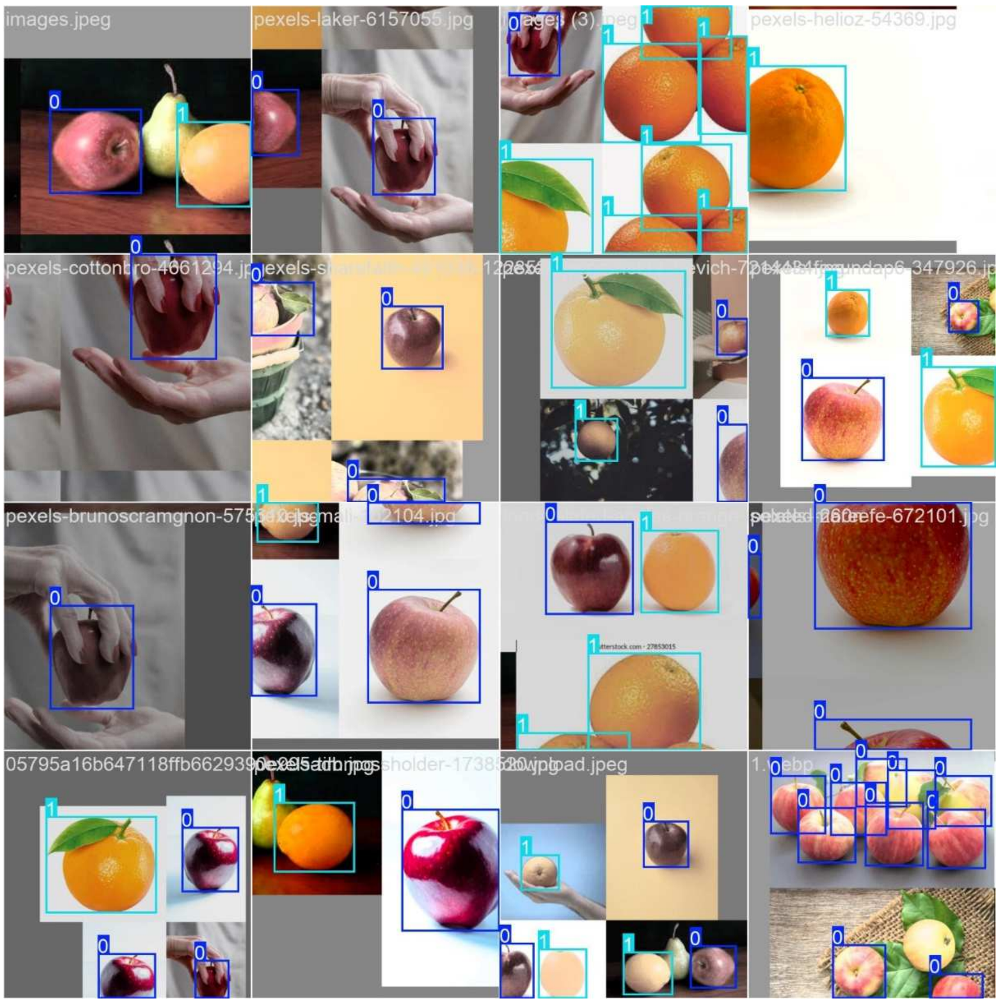


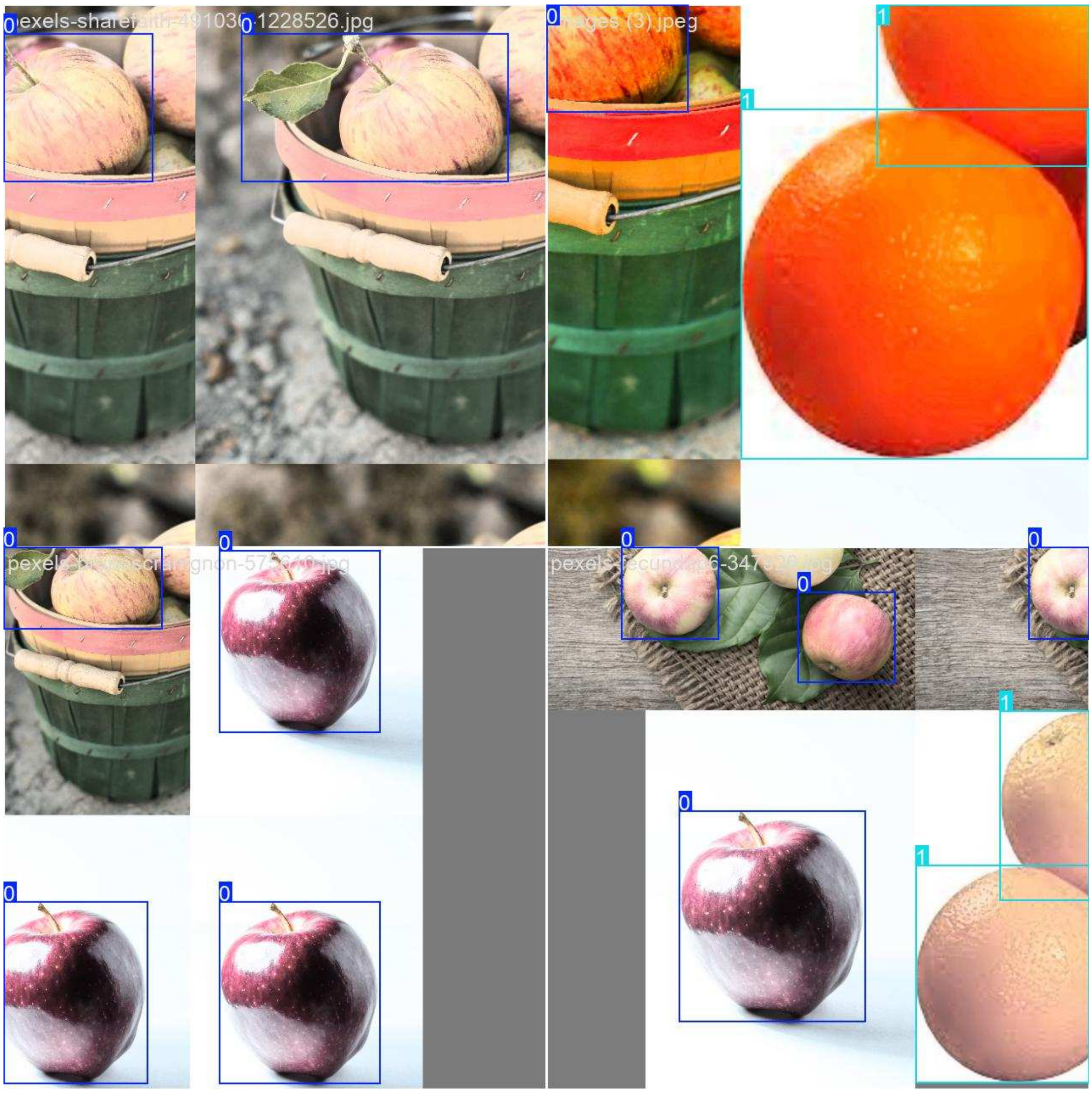
Confusion Matrix
Predicted
apple
orange
background
True
apple
orange
background
24
14
0
5
10
15
20


Confusion Matrix Normalized
1.00
1.00
apple
orange
background
Predicted
apple
orange
background
True
1.0
0.8
0.6
0.4
0.2
0.0


F1-Confidence Curve
apple
orange
all classes 1.00 at 0.864
F1
Confidence
0.0
0.2
0.4
0.6
0.8
1.0


train/box_loss
train/cls_loss
results
smooth
train/dfl_loss
metrics/precision(B)
metrics/recall(B)
val/box_loss
val/cls_loss
val/dfl_loss
metrics/mAP50(B)
metrics/mAP50-95(B)


images.jpeg
pexels-laker-6157055.jpg
pexels-helioz-54369.jpg
pexels-brunoscramgnon-575
pexels-cottonbro-4061294.jp
05795a16b647118ffb66293
1.webp


pexels-sharefaith-491036-1228526.jpg
images (3).jpeg
pexels-pixabay-573910.jpg
pexels-fecundap6-347926.jpg
0
1

5. Future Enhancement

1. **Improve Model Accuracy**: Use advanced architectures like EfficientDet or Detectron2 for better performance.
2. **Model Ensemble:** Combine multiple models to enhance detection and classification accuracy.
3. **Real-Time Integration**: Optimize the model for real-time applications using TensorRT or similar tools.
4. **Transfer Learning with Larger Datasets**: Fine-tune on larger datasets like COCO or OpenImages for better generalization.
5. **Edge Device Optimization**: Deploy on low-power devices (e.g., Raspberry Pi, NVIDIA Jetson) using TensorFlow Lite or ONNX.
6. **Object Tracking:** Add object tracking capabilities with algorithms like SORT or DeepSORT.
7. **Multi-Class Segmentation**: Incorporate instance segmentation (e.g., Mask R-CNN) for pixel-wise accuracy.
8. **Advanced Data Augmentation**: Use techniques like CutMix or AutoAugment for improved model robustness.
9. **Adversarial Defense**: Implement protection against adversarial attacks to ensure model reliability.
10. **Active Learning**: Continuously improve model accuracy through an active learning pipeline.
11. **Multi-Scale Learning**: Train for better detection of objects at various scales.
12. **Cloud-Based Deployment**: Deploy the model on cloud platforms with scalable APIs for wide access.

6. Bibliography

- https://www.youtube.com/watch?v=iy34dSwfEsY&t=455s
- https://docs.ultralytics.com/
- https://youtu.be/m9fH9OWn8YM?si=H8IPvPtaWYjMbUPO
- https://colab.research.google.com/
- https://www.cvat.ai/
- https://www.youtube.com/watch?v=WgPbbWmnXJ8&t=13594s
- https://www.pexels.com/search/apple/