Image Classification Using Yolo V8

COURSE PROJECT REPORT

By

VEDANT RAI (RA2111026010342)
RAJDEEP CHOUDHARY
(RA2111026010355)

Under the guidance of

Dr. Karpagam.M

In partial fulfillment for the Course

of

18CSE390T – COMPUTER VISION
in **CINTEL**

FACULTY OF ENGINEERING AND TECHNOLOGY

SCHOOL OF COMPUTING

SRM INSTITUTE OF SCIENCE AND

TECHNOLOGY KATTANKULATHUR

MAY 2024

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this minor project report for the course **18CSE390T COMPUTER VISION** entitled in "**Image Classification using YOLO V8**" is the bonafide work of **Vedant Rai(RA2111026010342) and Rajdeep Choudhary(RA2111026010355)** who carried out the work under my supervision.

SIGNATURE

Dr.Karpagam.M
Assistant Professor
Department of Computational Intelligence

SIGNATURE

Dr. R. Annie Uthra
Head of Department
Department of Computational Intelligence

ABSTRACT

Image classification is a fundamental task in computer vision, with applications ranging from medical diagnosis to autonomous driving. This mini-project focuses on developing and implementing a deep learning-based image classification system to categorize everyday objects. With the growing availability of high- quality image datasets and advancements in deep learning techniques, image classification has become increasingly accurate and accessible.

Image classification is a fundamental task in computer vision that involves categorizing objects or scenes within an image. YOLOv8, an evolution of the popular YOLO (You Only Look Once) object detection algorithm, offers a powerful framework for image classification with real-time processing capabilities. This mini project focuses on leveraging YOLOv8 to perform image classification tasks efficiently and accurately.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Motivation

The Exploration of YOLOv8: YOLOv8 represents a cutting-edge advancement in computer vision technology. By exploring this model, you're delving into the forefront of the field, gaining hands-on experience with state-of-the-art techniques. This exploration allows you to understand the latest advancements, potential strengths, and limitations of YOLOv8, contributing to your knowledge and skill development in computer vision.

Implementation for Image Classification: While YOLOv8 is primarily known as an object detection system, its architecture and capabilities make it suitable for image classification tasks as well. By implementing YOLOv8 for image classification, you're leveraging its unique features such as real-time processing, high accuracy, and efficiency. This choice expands the scope of YOLOv8's applicability beyond traditional object detection, demonstrating its versatility and potential utility in various computer vision applications.

High Accuracy and Speed: YOLOv8 is renowned for its exceptional accuracy and speed, making it a compelling choice for real-world applications where both performance metrics are crucial. By harnessing YOLOv8's capabilities, you aim to develop an image classification system that not only achieves high accuracy in categorizing objects but also operates efficiently in real-time or near-real-time scenarios. This emphasis on performance ensures that your system can meet the demands of modern applications, such as autonomous driving, surveillance, or augmented reality.

Potential Applications: The versatility of YOLOv8 opens up a wide range of potential applications beyond the scope of traditional image classification models. By demonstrating its effectiveness in image classification tasks, you pave the way for its adoption in diverse domains, including healthcare, agriculture, retail, and more. This project's outcomes have the potential to catalyze advancements in various industries by providing robust and efficient solutions for image analysis and classification challenges.

In summary, the motivation behind your mini-project lies in the exploration and implementation of YOLOv8 for image classification, driven by its high accuracy, speed, and potential applications across different domains. Through this endeavor, you aim to contribute to the advancement of computer vision technology and its practical utilization in real-world scenarios..

1.2 Objective

The Development of Image Classification System using YOLOv8: Your primary objective is to build a robust and effective image classification system using YOLOv8. This entails implementing the necessary infrastructure, including data preprocessing, model training, and evaluation procedures, to create a functional system capable of accurately categorizing objects within images. By focusing on YOLOv8, you're leveraging its unique architecture and capabilities to develop a state-of-the-art solution for image classification tasks.

Accurate and Efficient Object Classification: The ultimate goal of your project is to ensure that the developed system can classify objects in images with high accuracy and efficiency. Accuracy refers to the system's ability to correctly identify and categorize objects, while efficiency pertains to its speed and resource utilization. By optimizing the model's architecture, parameters, and training process, you aim to achieve a balance between accuracy and efficiency, ensuring that the system can deliver reliable results in real-time or near-real-time scenarios.

Enhanced Understanding of YOLOv8 Architecture: In addition to building a functional image classification system, your project aims to deepen your understanding of the YOLOv8 architecture and its application in image classification tasks. This involves studying the underlying principles, components, and mechanisms of YOLOv8, as well as experimenting with different configurations and techniques to gain insights into its performance characteristics. By thoroughly exploring the YOLOv8 architecture, you'll not only improve your proficiency in implementing the model but also acquire valuable knowledge that can inform future research and development efforts in the field of computer vision.

In summary, the objectives of your mini-project revolve around the development of an image classification system using YOLOv8, with a focus on achieving accuracy, efficiency, and a deeper understanding of the model's architecture. Through this endeavor, you aim to contribute to the advancement of image classification technology and enhance your expertise in the field of computer vision.

1.3 Problem statement

The problem statement is to create an image classification system that can take an input image and accurately classify the objects within it. YOLOv8 will be the core technology utilized for object detection and classification.

1.4 Challenges

Several challenges are anticipated during the development of the image classification system using YOLOv8:

- Implementing YOLOv8 for object detection and classification.
- Training the model on a suitable dataset for classification tasks.
- Fine-tuning the model for optimal performance.
- Developing an intuitive user interface for interacting with the system.
- Ensuring real-time or near-real-time processing for efficient image classification.

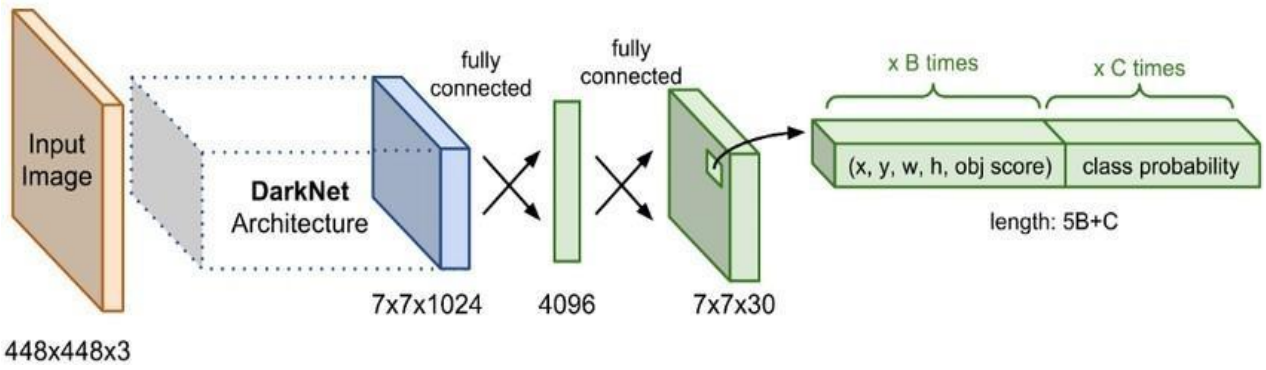
2. LITERATURE SURVEY

YOLO was introduced to the computer vision community via a paper release in 2015 by Joseph Redmon et al titled 'You Only Look Once: Unified, Real-Time Object Detection'. The paper reframed object detection, presenting it essentially as a single pass regression problem, initiating with image pixels and moving to bounding box and class probabilities. The proposed approach based on the 'unified' concept enabled the simultaneous prediction of multiple bounding boxes and class probabilities, improving both speed and accuracy. Since its inception in 2016 until the present year (2023), the YOLO family has continued to evolve at a rapid pace. Although the initial author (Joseph Redmon) halted further work within the computer vision domain at YOLO-v3 the effectiveness and potential of the core 'unified' concept have been further developed by several authors, with the latest addition to the YOLO family coming in the form of YOLO-v8.

For implementation of object detection, each grid cell would predict bounding boxes along with the dimensions and confidence scores. The confidence score was indicative of the absence or presence of an object within the bounding box. Therefore, the confidence score can be expressed as Equation (1):

confidence score =p(object)*IoUtruth pred (1)

Each bounding box consisted of five components (x,y,w,h, and the confidence score) with the first four components corresponding to center coordinates (x,y,width, and height) of the respective bounding box as shown in Figure:



Introduction of YOLO and Unified Object Detection Concept: YOLO's introduction marked a significant departure from traditional object detection approaches. By treating object detection as a regression problem, YOLO pioneered the concept of unified object detection, where bounding box predictions and class probabilities are generated simultaneously. This approach eliminated the need for computationally expensive region proposal algorithms, leading to faster and more efficient object detection pipelines.

Evolution of YOLO: The evolution of the YOLO family has been characterized by continuous innovation and refinement. YOLOv2 introduced improvements such as batch normalization and anchor boxes, which enhanced both accuracy and convergence speed. YOLOv3 further raised the bar by incorporating a feature pyramid network and introducing multiple scales for object detection. Subsequent iterations, including YOLOv4 and YOLOv5, have built upon these foundations, leveraging advancements in architecture design, optimization techniques, and training strategies to achieve state-of-the-art performance on object detection benchmarks.

Unified Object Detection Concept in YOLOv8: YOLOv8 represents the culmination of years of research and development in the YOLO family. Building upon the unified object detection concept, YOLOv8 integrates cutting-edge innovations to push the boundaries of performance and efficiency. By optimizing model architectures, leveraging advanced training methodologies, and harnessing the latest hardware capabilities, YOLOv8 delivers unparalleled accuracy and real-time processing speeds, making it a top choice for various object detection applications.

Implementation of Object Detection in YOLO: In YOLO-based object detection, the image is divided into a grid, and each grid cell predicts bounding boxes, confidence scores, and class probabilities. The confidence score reflects the likelihood of an object being present within a bounding box, while the class probabilities indicate the predicted object category. YOLO's streamlined architecture and efficient inference process enable rapid and accurate object detection, making it well-suited for use cases requiring real-time performance, such as autonomous driving, surveillance, and augmented reality.

Recent Advances in Object Detection:

Efficient Object Detection Architectures: Recent research has focused on developing efficient object detection architectures that strike a balance between accuracy and computational complexity. Models like EfficientDet and MobileDet leverage novel design principles, such as compound scaling

and efficient feature fusion, to achieve state-of-the-art performance with minimal computational resources. These architectures are particularly well-suited for deployment in resource-constrained environments, including mobile devices and embedded systems.

Attention Mechanisms: Attention mechanisms have emerged as a powerful tool for improving object detection performance by enabling models to focus on relevant regions of the input image. Transformer-based architectures, such as DETR (DEtection TRansformer), leverage self-attention mechanisms to capture long-range dependencies and contextual information, leading to more accurate object localization and classification. By directly predicting object bounding boxes and class labels, attention-based models offer a flexible and efficient alternative to traditional anchor-based methods.

Self-Supervised Learning and Semi-Supervised Approaches: Self-supervised learning techniques have gained traction in object detection research for leveraging large-scale unlabeled datasets to pretrain models on auxiliary tasks. By learning meaningful representations from unlabeled data, self-supervised pretraining can improve model generalization and robustness to domain shifts and variations in input data. Semi-supervised approaches, which combine labeled and unlabeled data during training, offer a practical solution for leveraging limited annotation resources to train high-performance object detectors.

Domain Adaptation and Few-Shot Learning: Domain adaptation methods aim to improve the generalization capabilities of object detectors by adapting models to new target domains using labeled data from related source domains. Techniques such as adversarial training and domain-specific adaptation modules enable models to learn domain-invariant representations and effectively transfer knowledge across domains. Few-shot learning approaches, on the other hand, enable object detectors to generalize to new object classes with only a few annotated examples, making them more adaptable to evolving environments and novel scenarios.

By exploring recent advancements in object detection research, we gain a deeper understanding of the latest trends, challenges, and opportunities in the field. Incorporating insights from both the evolution of the YOLO family and recent research findings enriches our understanding of object detection methodologies and paves the way for further advancements in computer vision technology**.**

3. REQUIREMENTS

3.1 Requirement Analysis

Data Collection:

- Determine the sources from which you will collect the images for training and testing.
- Specify the number of images required for a meaningful classification model.
- Ensure that the data collected is diverse and representative of real-world scenarios

Data Preprocessing:

- Detail the preprocessing steps, including resizing, normalization, and data augmentation if necessary.
- Ensure that the data is labeled properly for supervised learning.

Model Selection:

- Justify your choice of YOLOv8 and explain how it is suitable for your problem.
- Compare it with other models if necessary.

Evaluation Metrics:

- Define the performance metrics to assess the model's accuracy (e.g., mAP, precision, recall).
- Specify the minimum acceptable performance.

Software and Tools:

- List the software and tools required for developing and deploying the YOLOv8 model (e.g., Python, YOLO framework, deep learning libraries, image processing libraries).

Budget and Resources:

- Estimate the budget required for the project, including hardware costs, if applicable.
- Identify the human resources needed for data labeling, model development, and evaluation.

3.2 Hardware Requirement

GPU Acceleration:

- YOLO models are computationally intensive. Ensure you have access to a GPU for training the model efficiently.
- Specify the GPU model and VRAM (Video RAM) capacity required.

Storage:

- Have sufficient storage to store the dataset, model checkpoints, and intermediate results.
- SSDs are recommended for faster data access during training.

Memory (RAM):

- Depending on the size of your dataset and model, you may need a considerable amount of RAM for data loading and model training.

CPU:

- A modern multi-core CPU is required for data preprocessing, inference, and general system tasks.

Operating System:

- Specify the OS you'll be using. Linux-based systems are commonly preferred for deep learning tasks.

Testing and Debugging Equipment:

- Have a monitor, keyboard, and mouse available for setup and debugging, especially in a development environment.

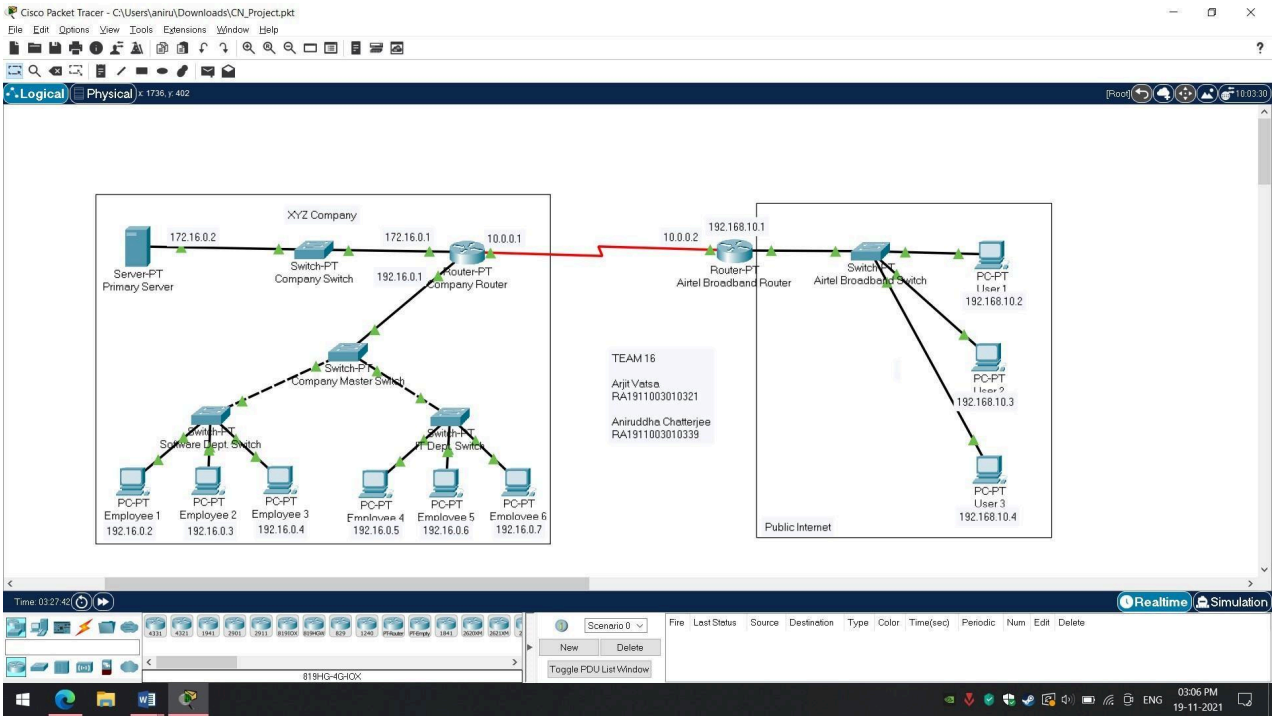
Security and Access Control:

- Implement security measures to protect sensitive data and model checkpoints.

4. ARCHITECTURE AND DESIGN

4.1 Network Architecture

The network architecture is as follows:



The architecture consists of three major networks:

- Company Network(s)
- Public Internet
- Network maintained by the Internet Service Provider

1. Company Network(s):

Purpose: This private network serves as the backbone for all the company's IT infrastructure. It securely connects various devices and resources used by employees, such as:

Servers: These house critical data and applications used by the company (e.g., Server-PT).
Switches: These intelligent devices manage data flow within the network, directing information to the intended devices (e.g., Switch-PT and Switch).
Routers: These act as traffic directors, intelligently forwarding data packets between different networks, including the company network and the internet (e.g., Router-PIT and Router-P).
PCs (Personal Computers): These are the workstations used by employees to access company resources and the internet (multiple PCs).
Security: The company network is typically isolated from the public internet by firewalls and other security measures. This helps protect sensitive data and resources from unauthorized access.

2. Public Internet:

Definition: This vast, interconnected global network allows communication and resource sharing between devices across the world. It's represented by the cloud labeled "Public Internet" in the diagram.

Components: The public internet is a complex web of interconnected networks owned and operated by various entities like internet service providers (ISPs), governments, and educational institutions. It consists of:

Backbone networks: These high-speed fiber optic cables form the core of the internet, carrying massive amounts of data traffic.
Regional networks: These connect smaller networks to the backbone networks, creating a more distributed infrastructure.
Local networks: These are smaller networks within organizations or communities that connect to regional networks.
Accessibility: The public internet allows access to various resources and services, including:

Websites: These provide information, entertainment, and online services.
Email: This enables electronic communication between individuals and organizations.

Cloud storage: This allows storing data remotely and accessing it from anywhere.
3. Network Maintained by the Internet Service Provider (ISP):

Role: The ISP acts as a bridge between the company network and the public internet. Devices in the ISP network (e.g., "Airtel Broadband Router" and "Airtel Broadband Bit") provide internet access to the company. These devices typically include:

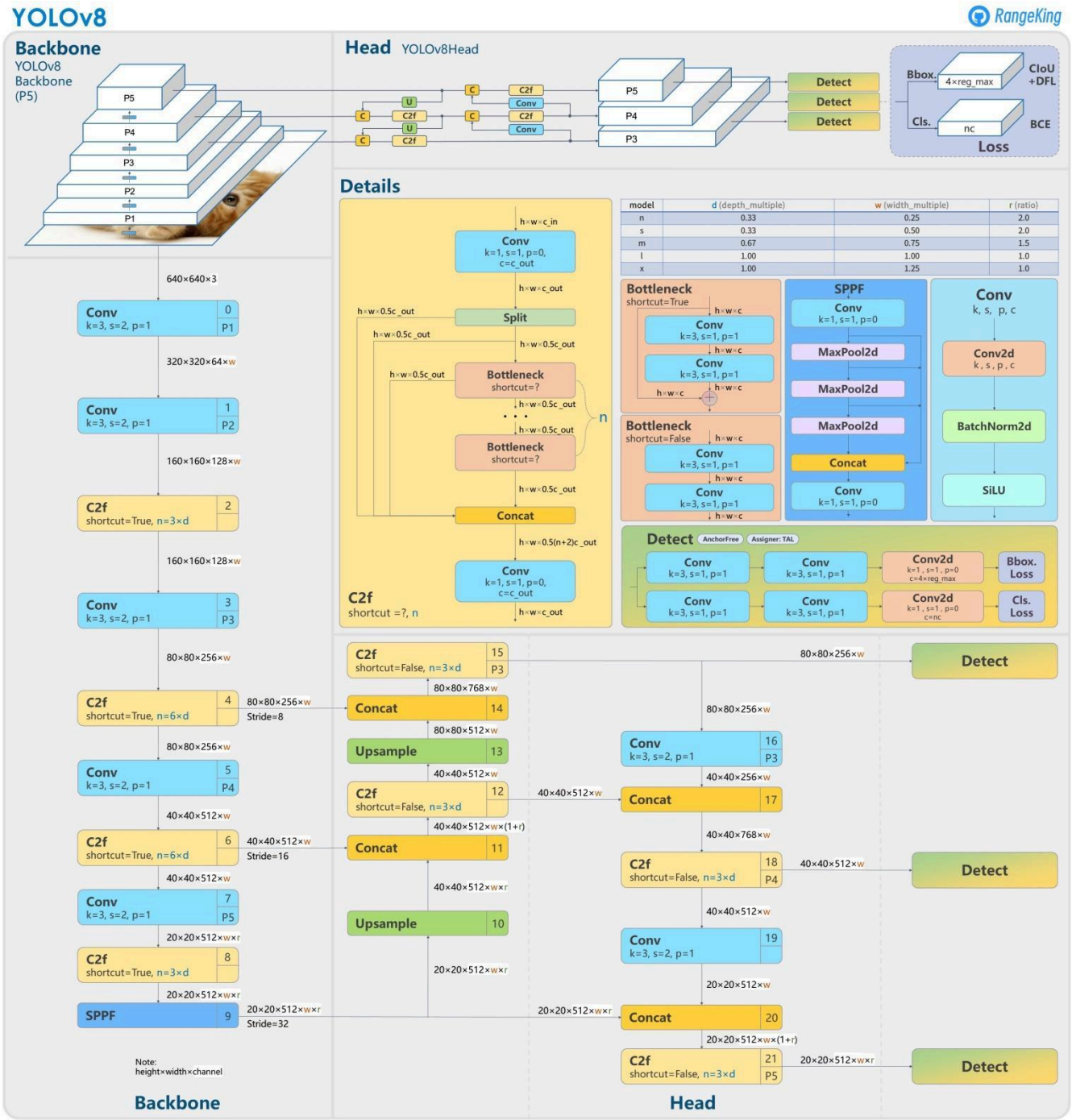
Routers: These route data packets between the company network and the ISP's network.
Modems: These convert the signal format between the company network and the ISP's network (depending on the connection type).
Services: The ISP offers various internet access plans with different speeds and data caps depending on the company's needs.

Interconnections - How It Works:

Internal Communication: Devices within the company network (e.g., PCs) communicate directly with each other using switches and routers.
Internet Access: When a company device needs to access resources on the internet (e.g., accessing a website), it sends a data packet to its designated router (e.g., Router-P).
Routing: The router examines the destination address of the data packet and determines the most efficient route to send it. It likely forwards the packet to the ISP's router (Airtel Broadband Router) connected to the company network.
Reaching the Internet: The ISP's router further routes the data packet through its network and eventually to the public internet backbone. The packet finally reaches its destination website or server.
Response: The requested information or resource from the internet travels back to the company device through the reverse path, following the routing decisions made by the routers involved.
Benefits of this Architecture:

Security: Isolating the company network from the public internet enhances data security.
Control: Organizations can manage internet access and usage by employees.

4.2 MODEL ARCHITECTURE



Main Blocks

The first step to understanding the YOLO architecture is to understand that there are 3 essential blocks in the algorithm and everything will occur in these blocks, which are: Backbone, Neck and Head. The function of each block is described below.

Backbone:

Function: The backbone, also known as the feature extractor, is responsible for extracting meaningful features from the input.
Activities:
- Captures simple patterns in the initial layers, such as edges and textures.
- Can have multiple scales of representation as you go, capturing features from different levels of abstraction.
- Will provide a rich, hierarchical representation of the input.

Neck:

Function: The neck acts as a bridge between the backbone and the head, performing feature fusion operations and integrating contextual information. Basically the Neck assembles feature pyramids by aggregating feature maps obtained by the Backbone, in other words, the neck collects feature maps from different stages of the backbone.
Activities:
- Perform concatenation or fusion of features of different scales to ensure that the network can detect objects of different sizes.
- Integrates contextual information to improve detection accuracy by considering the broader context of the scene.
- Reduces the spatial resolution and dimensionality of resources to facilitate computation, a fact that increases speed but can also reduce the quality of the model.

Head:

Function: The head is the final part of the network and is responsible for generating the network's outputs, such as bounding boxes and confidence scores for object detection.
Activities:
- Generates bounding boxes associated with possible objects in the image.
- Assigns confidence scores to each bounding box to indicate how likely an object is present.
- Sorts the objects in the bounding boxes according to their categories.

Conv:

The YOLO architecture adopts the local feature analysis approach instead of examining the image as a whole, the objective of this strategy is mainly to reduce computational effort and enable real-time detection. To extract feature maps, convolutions are used several times in the algorithm.

Convolution is a mathematical operation that combines two functions to create a third. In computer vision and signal processing, convolution is often used to apply filters to images or

signals, highlighting specific patterns. In convolutional neural networks (CNNs), convolution is used to extract features from inputs such as images. Convolutions are structured by Kernels (K), Strides (s) and paddings (p).

5. IMPLEMENTATION

Implementing a YOLO (You Only Look Once) version 8-based image classification project can be a complex task, as YOLO is primarily designed for object detection rather than traditional image classification.

Prerequisites:

Before running the code, ensure that you have the following prerequisites set up:

Python Environment:

Make sure you have a Python environment set up on your machine or Colab notebook.

NVIDIA GPU:

The code utilizes a GPU for accelerated processing. Ensure that you have access to a GPU, such as an NVIDIA GPU, to run the code efficiently.

Roboflow Account: You need a Roboflow account and an API key to download the dataset used in the code. You can sign up for a Roboflow account at Roboflow website.

Ultralytics Library: Install the Ultralytics library, which provides easy-to-use APIs for object detection tasks.

install it using pip:

!pip install ultralytics

Roboflow Library:

Install the Roboflow library to interact with Roboflow API and

download datasets:

!pip install roboflow

YOLOv8 Pre-trained Model:

Download or obtain the YOLOv8 pre-trained model (e.g., yolov8s.pt) from a reliable source. You can use the model provided in the code or train your own model using Roboflow datasets.

Importing Libraries and Checking GPU:

The code starts by importing necessary libraries and checks for the availability of an NVIDIA GPU using !nvidia-smi.

Setting Up Paths and Installing Ultralytics:

The code sets up the home directory path and installs the Ultralytics library using pip.
Importing Ultralytics and Clearing Output:
Ultralytics library is imported, and any previous outputs are cleared.

Running Object Detection with YOLOv8:

The code performs object detection using YOLOv8 on a sample image (0001.jpg). The yolo command-line interface (CLI) is used to specify the task (detect), mode (predict), model (yolov8n.pt), confidence threshold (conf=0.25), and source image path (source=/0001.jpg).

Downloading Dataset from Roboflow:

The code downloads the dataset from Roboflow using the Roboflow library. It fetches the dataset associated with the specified Roboflow project and version and saves it to the local directory.

Training Object Detection Model with YOLOv8:

The code trains an object detection model using YOLOv8 on the downloaded dataset. It specifies the task (detect), mode (train), model (yolov8s.pt), data configuration file (data={dataset.location}/data.yaml), number of epochs (epochs=20), and image size (imgsz=800).

Visualizing Training Results:

The code displays various training results, including a confusion matrix, training progress plot, and sample training batch image.

Running Object Detection on Validation Set:

The code evaluates the trained model on the validation set using the YOLO CLI. It specifies the task (detect), mode (val), trained model path, and data configuration file.

Running Object Detection on Test Set:

The code performs object detection on the test set images using the trained model. It specifies the task (detect), mode (predict), trained model path, confidence threshold, and source image directory.

Visualizing Object Detection Results:

The code displays sample object detection results on test set images, showcasing the model's performance in detecting objects.

Running Object Detection on Custom Image:

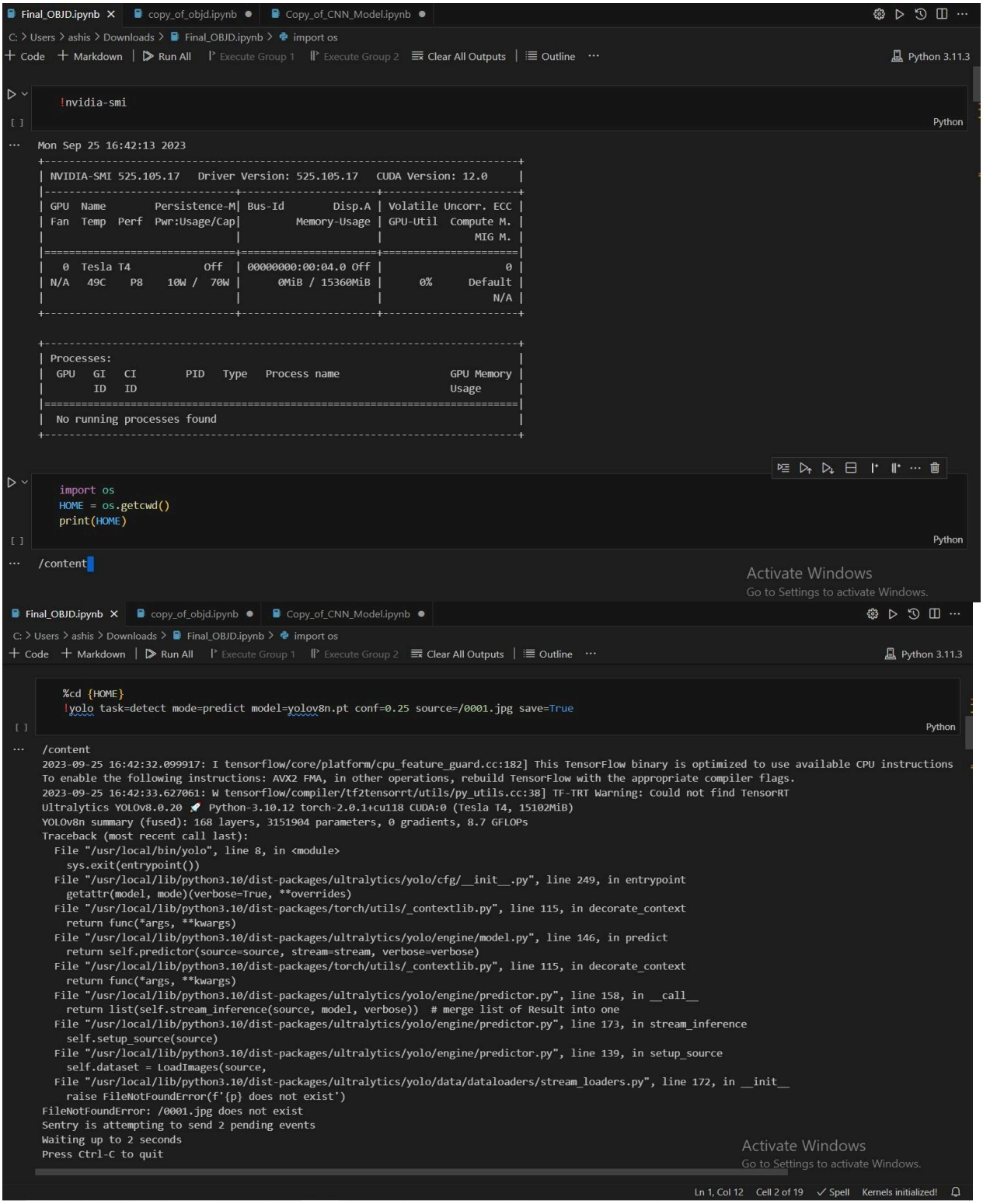
Lastly, the code performs object detection on a custom image (UAVS_286_jpg.rf.f6110a73c6a1bd33115104864f68b23e.jpg) using the trained model. It specifies the task (detect), mode (predict), trained model path, confidence threshold, and source image path.

CODE:

```
# -*- coding: utf-8 -*-
"""Final_OBJD

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1yXciCdg170T5X3pzJ1z47mQXEKRlhTtb
"""
!nvidia-smi
import os
HOME = os.getcwd()
print(HOME)

# Pip install method (recommended)

!pip install ultralytics==8.0.20

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()

from ultralytics import YOLO

from IPython.display import display, Image

# Commented out IPython magic to ensure Python compatibility.
# %cd {HOME}
!yolo task=detect mode=predict model=yolov8n.pt conf=0.25 source=/0001.jpg
save=True

model = YOLO(f'{HOME}/yolov8n.pt')
```

```
results =
model.predict(source='https://media.roboflow.com/notebooks/examples/dog.jpeg',
conf=0.25)

!mkdir {HOME}/datasets

!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="hlDXIF9H34eRdsDoUfHI")
project = rf.workspace("ashish-maurya-rk50z").project("object_detection_1-sbzfe")
dataset = project.version(11).download("yolov8")

!yolo task=detect mode=train model=yolov8s.pt data={dataset.location}/data.yaml
epochs=20 imgsz=800 plots=True

!ls /content/datasets/runs/detect/train


Image(filename=f'/content/datasets/runs/detect/train/confusion_matrix.png',
width=600)


Image(filename=f'/content/datasets/runs/detect/train/results.png', width=600)


Image(filename=f'/content/datasets/runs/detect/train/train_batch0.jpg',
width=600)



!yolo task=detect mode=val
model=/content/datasets/runs/detect/train/weights/best.pt
data={dataset.location}/data.yaml


!yolo task=detect mode=predict
model=/content/datasets/runs/detect/train/weights/best.pt conf=0.25
source={dataset.location}/test/images save=True

!yolo task=detect mode=predict model=/content/datasets/yolov8s.pt
source="/content/UAVS_286_jpg.rf.f6110a73c6a1bd33115104864f68b23e.jpg" save=True

Image(filename=f'/content/runs/detect/predict2/istockphoto-471196487-
2048x2048_jpg.rf.32a6510bd6c8231582d86560692c1618.jpg', width=600)

Image(filename=f'/content/runs/detect/predict4/UAVS_286_jpg.rf.f6110a73c6a1bd3311
5104864f68b23e.jpg', width=800)
```

6. RESULTS AND DISCUSSION



```python
from roboflow import Roboflow
rf = Roboflow(api_key="hlDXIF9H34eRdsDoUfHI")
project = rf.workspace("ashish-maurya-rk50z").project("object_detection_1-sbzfe")
dataset = project.version(11).download("yolov8")
```

```
mkdir: cannot create directory ‘/content/datasets’: File exists
/content/datasets
Requirement already satisfied: roboflow in /usr/local/lib/python3.10/dist-packages (1.1.7)
Requirement already satisfied: certifi==2022.12.7 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2022.12.7)
Requirement already satisfied: chardet==4.0.0 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.0.0)
Requirement already satisfied: cycler==0.10.0 in /usr/local/lib/python3.10/dist-packages (from roboflow) (0.10.0)
Requirement already satisfied: idna==2.10 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.10)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.4.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from roboflow) (3.7.1)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.23.5)
Requirement already satisfied: opencv-python-headless==4.8.0.74 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.8.0.74)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from roboflow) (9.4.0)
Requirement already satisfied: pyparsing==2.4.7 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.4.7)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.8.2)
Requirement already satisfied: python-dotenv in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.0.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.31.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.16.0)
Requirement already satisfied: supervision in /usr/local/lib/python3.10/dist-packages (from roboflow) (0.14.0)
Requirement already satisfied: urllib3>=1.26.6 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.0.4)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.66.1)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (6.0.1)
Requirement already satisfied: requests-toolbelt in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.0.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (1.1.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (4.42.1)
```

Activate Windows
Go to Settings to activate Windows.

Ln 1, Col 12 Cell 2 of 19 Spell Kernels initialized!

+ Code + Markdown | Run All | Execute Group 1 | Execute Group 2 | Clear All Outputs | Outline ... Python 3.11.3

```python
!yolo task=detect mode=train model=yolov8s.pt data={dataset.location}/data.yaml epochs=20 imgsz=800 plots=True
```

```
Downloading https://github.com/ultralytics/assets/releases/download/v0.0.0/yolov8s.pt to yolov8s.pt...
100% 21.5M/21.5M [00:00<00:00, 168MB/s]

Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
yolo/engine/trainer: task=detect, mode=train, model=yolov8s.yaml, data=/content/datasets/object_detection_1-11/data.yaml, epochs=20, patience=50, batch
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Arial.ttf...
100% 755k/755k [00:00<00:00, 14.4MB/s]
2023-09-25 16:43:22.402713: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Overriding model.yaml nc=80 with nc=8

                   from  n    params  module                                       arguments
  0                  -1  1       928  ultralytics.nn.modules.Conv                  [3, 32, 3, 2]
  1                  -1  1     18560  ultralytics.nn.modules.Conv                  [32, 64, 3, 2]
  2                  -1  1     29056  ultralytics.nn.modules.C2f                   [64, 64, 1, True]
  3                  -1  1     73984  ultralytics.nn.modules.Conv                  [64, 128, 3, 2]
  4                  -1  2    197632  ultralytics.nn.modules.C2f                   [128, 128, 2, True]
  5                  -1  1    295424  ultralytics.nn.modules.Conv                  [128, 256, 3, 2]
  6                  -1  2    788480  ultralytics.nn.modules.C2f                   [256, 256, 2, True]
  7                  -1  1   1180672  ultralytics.nn.modules.Conv                  [256, 512, 3, 2]
  8                  -1  1   1838080  ultralytics.nn.modules.C2f                   [512, 512, 1, True]
  9                  -1  1    656896  ultralytics.nn.modules.SPPF                  [512, 512, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11            [-1, 6]  1         0  ultralytics.nn.modules.Concat                [1]
 12                  -1  1    591360  ultralytics.nn.modules.C2f                   [768, 256, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
...
                  Tree        530        450      0.551      0.567      0.565      0.302
               Vehicle        530        387      0.604      0.687      0.667      0.416
Speed: 2.3ms pre-process, 6.6ms inference, 0.0ms loss, 3.4ms post-process per image
Results saved to runs/detect/train
```

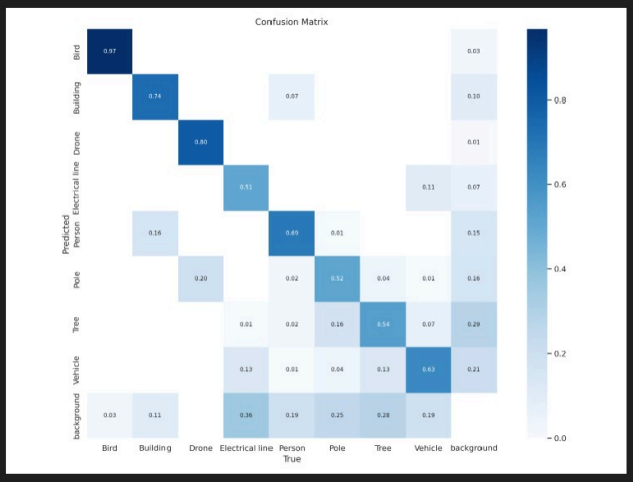
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Activate Windows
Go to Settings to activate Windows.

```
%cd {HOME}
Image(filename=f'/content/datasets/runs/detect/train/confusion_matrix.png', width=600)
```

Python

```
/content
```



+ Code + Markdown

Activate Windows
Go to Settings to activate Windows.

Ln 1, Col 12 Cell 2 of 19 ✓ Spell Kernels initialized!

Final_OBJD.ipynb × copy_of_objd.ipynb ● Copy_of_CNN_Model.ipynb ●

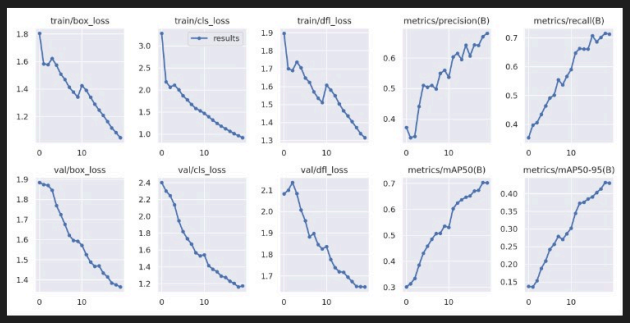
C: > Users > ashis > Downloads > Final_OBJD.ipynb > import os

+ Code + Markdown | Run All Execute Group 1 Execute Group 2 Clear All Outputs | Outline ··· Python 3.11.3

```
%cd {HOME}
Image(filename=f'/content/datasets/runs/detect/train/results.png', width=600)
```

Python

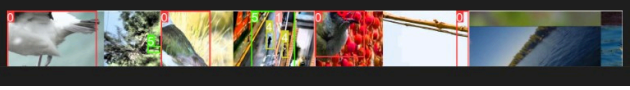
```
/content
```




```
%cd {HOME}
Image(filename=f'/content/datasets/runs/detect/train/train_batch0.jpg', width=600)
```

Python

```
/content
```



Activate Windows
Go to Settings to activate Windows.

Ln 1, Col 12 Cell 2 of 19 ✓ Spell Kernels initialized!

Final_OBJD.ipynb | copy_of_objd.ipynb | Copy_of_CNN_Model.ipynb

C: > Users > ashis > Downloads > Final_OBJD.ipynb > import os

+ Code | + Markdown | Run All | Execute Group 1 | Execute Group 2 | Clear All Outputs | Outline | Python 3.11.3

```
%cd {HOME}
Image(filename=f'/content/datasets/runs/detect/train/train_batch0.jpg', width=600)
```

Python

```
/content
```

Activate Windows
Go to Settings to activate Windows.

Ln 1, Col 12 | Cell 2 of 19 | Spell | Kernels initialized!

```
!yolo task=detect mode=val model=/content/datasets/runs/detect/train/weights/best.pt data={dataset.location}/data.yaml
```

Python

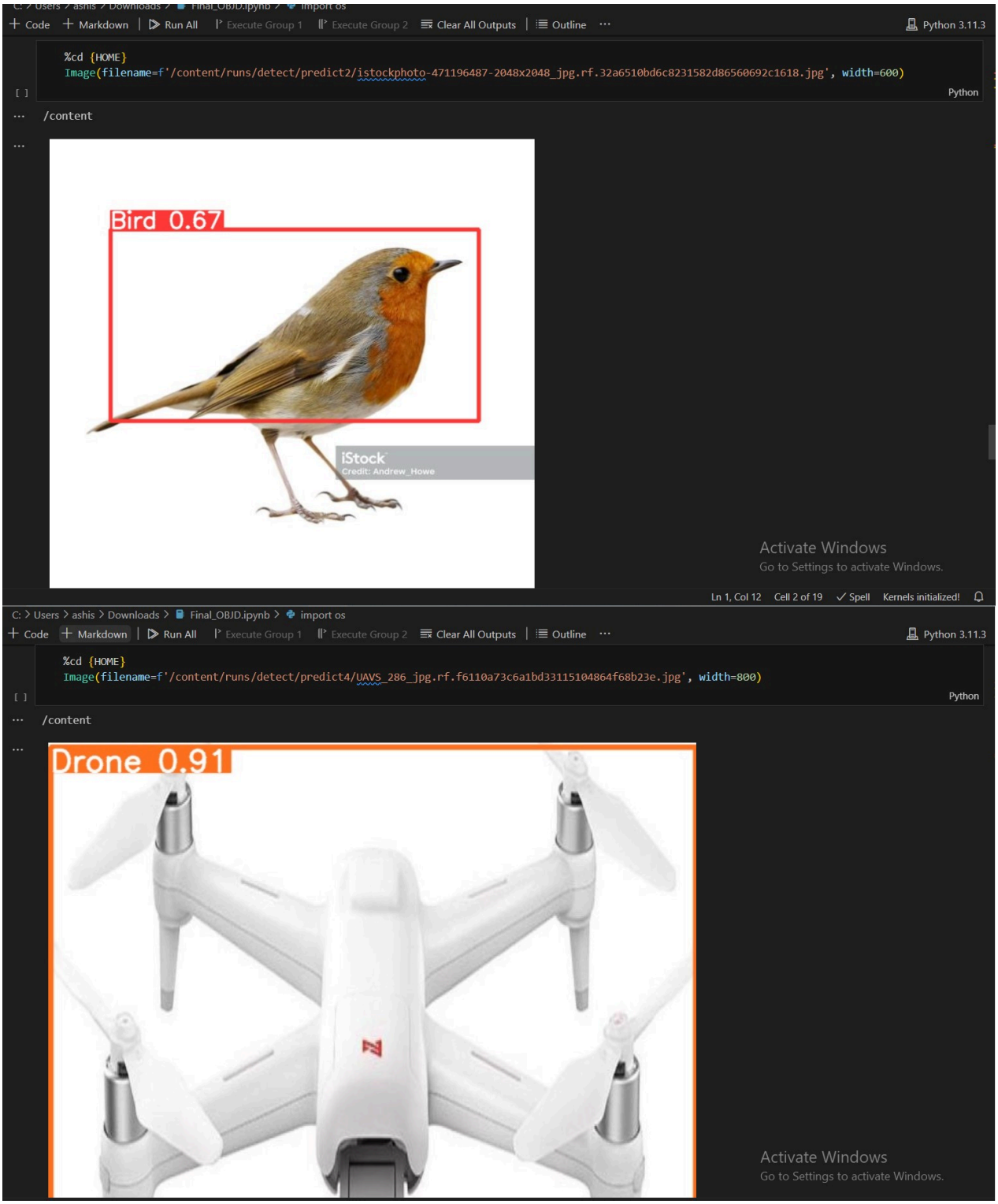
```
2023-09-25 18:26:52.538543: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11128680 parameters, 0 gradients, 28.5 GFLOPs
val: Scanning /content/datasets/object_detection_1-11/valid/labels.cache... 530 images, 0 backgrounds, 0 corrupt: 100% 530/530 [00:00<?, ?it/s]
WARNING ⚠ Box and segment counts should be equal, but got len(segments) = 201, len(boxes) = 1834. To resolve this only boxes will be used and all segm
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 34/34 [00:16<00:00,  2.08it/s]
                   all        530       1834      0.669      0.716      0.703      0.432
                  Bird        530        130      0.849      0.954       0.96       0.52
              Building        530        223      0.759      0.821      0.807      0.586
                 Drone        530         15        0.8      0.867      0.957      0.628
      Electrical line        530        132      0.538      0.568      0.467      0.344
                Person        530        236      0.612      0.675      0.629      0.357
                  Pole        530        261      0.635      0.586      0.571      0.298
                  Tree        530        450      0.553      0.567      0.566      0.303
               Vehicle        530        387      0.605       0.69      0.667      0.418
Speed: 1.3ms pre-process, 13.0ms inference, 0.0ms loss, 2.4ms post-process per image
```

```
%cd {HOME}
!yolo task=detect mode=predict model=/content/datasets/runs/detect/train/weights/best.pt conf=0.25 source={dataset.location}/test/images save=True
```

Python

```
/content
2023-09-25 18:27:45.989447: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11128680 parameters, 0 gradients, 28.5 GFLOPs
image 1/489 /content/datasets/object_detection_1-11/test/images/0028_jpg.rf.98e8644bb79adf70949560b7808c31ab.jpg: 800x800 1 Bird, 24.7ms
image 2/489 /content/datasets/object_detection_1-11/test/images/0041_jpg.rf.ccf79f1a35ba3877fc15bbd2907bf781.jpg: 800x800 1 Bird, 24.7ms
image 3/489 /content/datasets/object_detection_1-11/test/images/0047_jpg.rf.747f2f5d1ca266bd878a07b92e091509.jpg: 800x800 1 Bird, 24.8ms
image 4/489 /content/datasets/object_detection_1-11/test/images/0069_jpg.rf.0eac026fda0d2eb7248eb6eff9f24a32.jpg: 800x800 1 Bird, 24.8ms
image 5/489 /content/datasets/object_detection_1-11/test/images/0079_jpg.rf.91781c805b74a4d5489a2dca9b9c8b2c.jpg: 800x800 1 Bird, 24.7ms
```

Activate Windows
Go to Settings to activate Windows.



Explaination: In the context of the YOLO object detection framework, the initial 2015 paper by Joseph Redmon and the subsequent evolution of YOLO versions until YOLOv8 introduced a unified approach to real-time object detection. The key innovation was treating object detection as a single-pass regression problem, starting with image pixels and ending with bounding box coordinates and class probabilities.

One of the central concepts of YOLO is its ability to predict multiple bounding boxes and

class probabilities simultaneously, significantly improving both speed and accuracy. This was achieved through the "unified" concept, which is the hallmark of YOLO. The ability to predict multiple bounding boxes and class probabilities in one pass is particularly beneficial in real-time applications, where speed is crucial.

From YOLO's inception in 2016 to the present year (2023), the YOLO family of object detection models has continued to evolve rapidly. Although the original author, Joseph Redmon, ceased his work within the computer vision domain after YOLOv3, the core "unified" concept has been further developed by several authors. The latest addition to the YOLO family is YOLOv8, which builds on the foundations of its predecessors.

The key contribution of the YOLO framework can be summarized in the following points:

Unified Approach: YOLO revolutionized object detection by proposing a unified approach that predicts multiple bounding boxes and class probabilities in a single pass.

Real-Time Performance: The single-pass design significantly improved the speed of object detection, making it suitable for real-time applications.

Evolution: YOLO has evolved over the years, with various versions enhancing its capabilities, with YOLOv8 being the latest addition.

7. CONCLUSION & FUTURE ENHANCEMENT

7.1 Conclusion

This Mini project proposed a small-size object detection algorithm based on a camera sensor; different from traditional camera sensors, we combined a camera sensor and artificial intelligence. Then, some problems in the newly released YOLOv8 and existing small-size object detection algorithms were analyzed and solved. New feature fusion methods and network architectures were proposed. It greatly improved the learning ability of the network. The test and comparison were carried out on the Visdrone dataset, the Tinyperson dataset, and the PASCAL VOC2007 dataset. Through an analysis and experiments, the feasibility of each part of the optimization was proved. DC-YOLOv8 outperformed other detectors in both accuracy and speed. Small targets in various complex scenes were easier to capture.At present, the application of our proposed algorithm in traffic safety detection is more efficient.the accuracy of car detection is the highest, so application in traffic safety effect is the best, such as traffic signs, vehicles, pedestrians, etc.In the future, we will continue to conduct in-depth research on camera sensors and will strive to outperform existing detectors in detection accuracy at various sizes as soon as possible.

7.2 Future Enhancement

1. **Fine-tuning the Model:** Continuously fine-tune the YOLOv8 model on new data to improve its accuracy and adaptability to different scenarios.
2. **Multi-Class Object Detection:** Extend the model to detect and classify multiple classes of objects in an image, not just a single class.
3. **Real-Time Processing:** Optimize the model and the processing pipeline to achieve real-time object detection and classification on video streams or live camera feeds.
4. **Improved Accuracy:** Experiment with different model architectures, hyperparameters, and training strategies to further improve the accuracy of the model.
5. **Data Augmentation:** Apply advanced data augmentation techniques to increase the diversity of the training data and improve the model's generalization ability.
6. **Model Compression:** Explore techniques for model compression to reduce the size of the model and improve its efficiency for deployment on resource-constrained devices.
7. **Integration with Other Technologies:** Integrate the YOLOv8 model with other technologies such as robotics, drones, or augmented reality for more advanced applications.
8. **User Interface Improvements:** Enhance the user interface to make it more intuitive and user-friendly, allowing users to easily interact with the model and visualize its output.
9. **Deployment on Edge Devices:** Optimize the model for deployment on edge devices to enable on-device object detection and classification without the need for a constant internet connection.

8. REFERENCES

1. Zou, M.Y.; Yu, J.J.; Lv, Y.; Lu, B.; Chi, W.Z.; Sun, L.N. A Novel Day-to-Night Obstacle Detection Method for Excavators based on Image Enhancement and Multi-sensor Fusion. *IEEE Sens. J.* **2023**, *23*, 10825–10835. [Google Scholar]

2. Liu, H.; Member, L.L. Anomaly detection of high-frequency sensing data in transportation infrastructure monitoring system based on fine-tuned model. *IEEE Sens. J.* **2023**, *23*, 8630–8638. [Google Scholar] [CrossRef]

3. Zhu, F.; Lv, Y.; Chen, Y.; Wang, X.; Xiong, G.; Wang, F.Y. Parallel Transportation Systems: Toward IoT-Enabled Smart Urban Traffic Control and Management. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 4063–4071. [Google Scholar] [CrossRef]

4. Thevenot, J.; López, M.B.; Hadid, A. A Survey on Computer Vision for Assistive Medical Diagnosis from Faces. *IEEE J. Biomed. Health Inform.* **2018**, *22*, 1497–1511. [Google Scholar] [CrossRef] [PubMed]

5. Abadi, A.D.; Gu, Y.; Goncharenko, I.; Kamijo, S. Detection of Cyclist's Crossing Intention based on Posture Estimation for Autonomous Driving. *IEEE Sens. J.* **2023**, *2023*, 1. [Google Scholar] [CrossRef]