Assignment 7

**1. Aim**

To implement an object detection system using the YOLO (You Only Look Once) algorithm with a pretrained model to identify and localize objects within images or video streams.

**2. Objectives**

* To understand the YOLO algorithm and its architecture for real-time object detection.
* To utilize a pretrained YOLO model to detect objects in images and videos.
* To evaluate the model’s performance in terms of accuracy and processing speed.
* To implement techniques for visualizing detected objects and their bounding boxes.

**3. Theory**

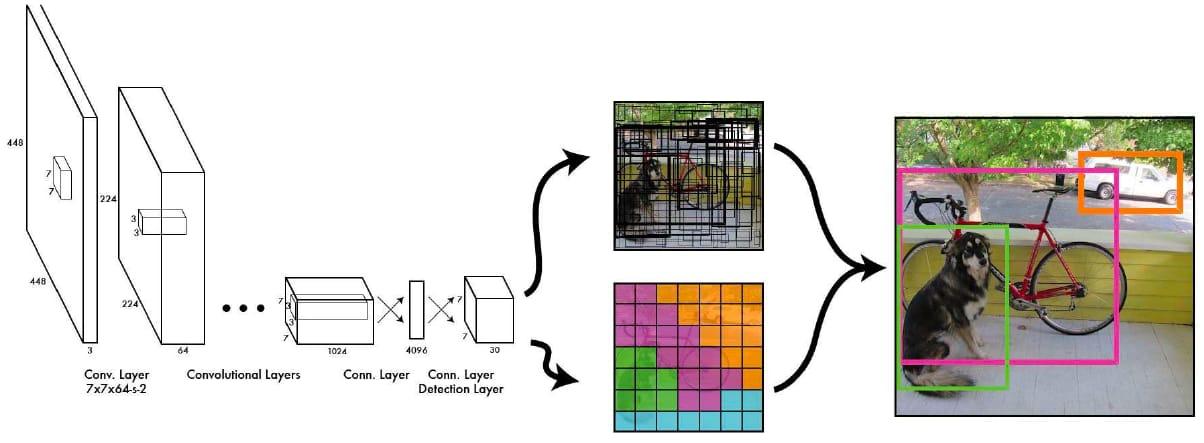
**3.1 Object Detection**

Object detection is a computer vision task that involves identifying and localizing objects within images or video streams. The goal is to classify objects while simultaneously providing their locations using bounding boxes.

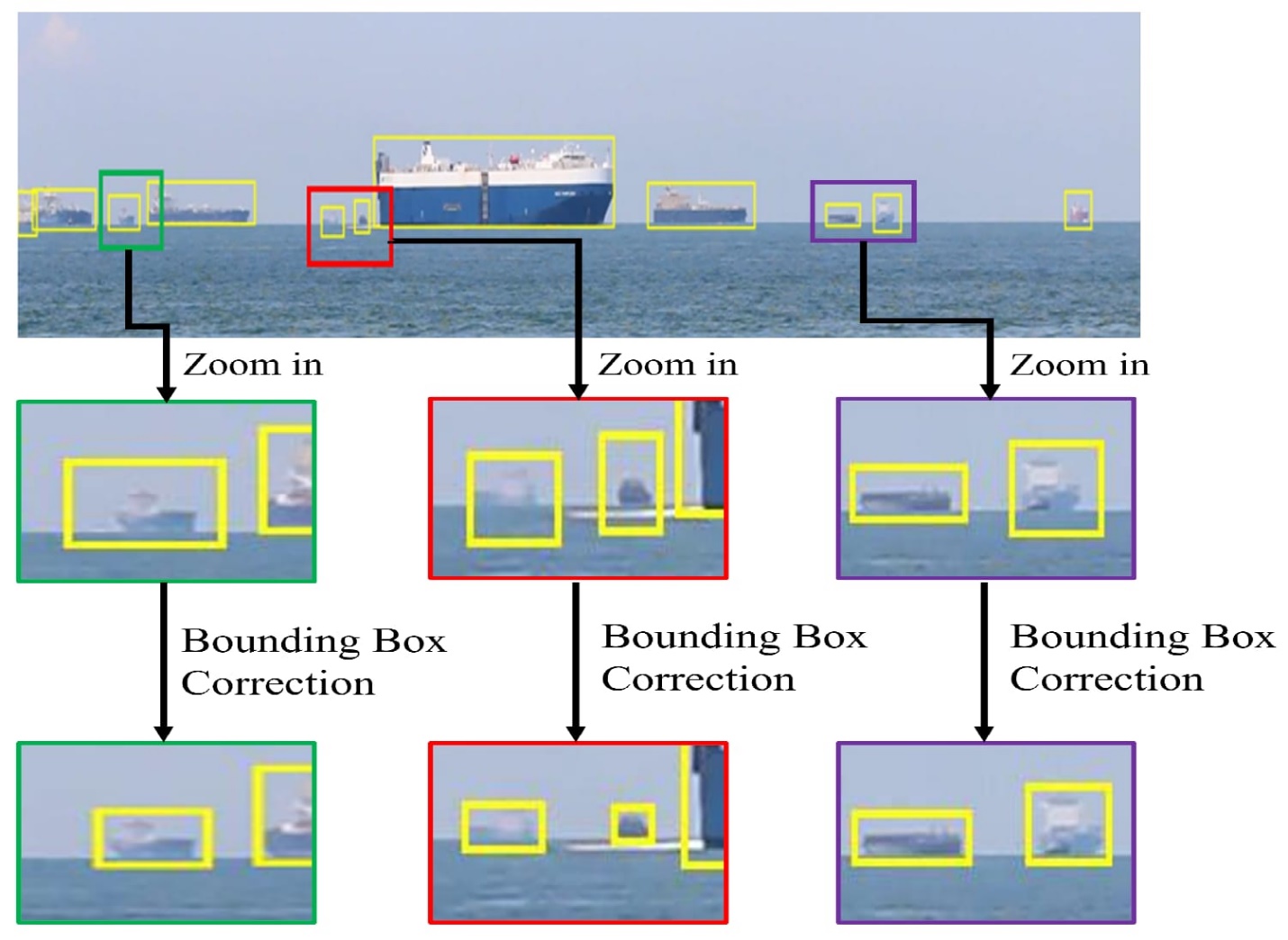
**3.2 YOLO (You Only Look Once)**

YOLO is a state-of-the-art object detection system that operates in real-time. Unlike traditional object detection methods that apply classifiers to various parts of the image, YOLO treats object detection as a single regression problem. It predicts bounding boxes and class probabilities directly from full images in a single evaluation.

* **Architecture:**
  + **Grid Division:** The image is divided into an S×SS \times SS×S grid. Each grid cell is responsible for predicting a certain number of bounding boxes and confidence scores.
  + **Bounding Box Predictions:** Each cell predicts a fixed number of bounding boxes and their corresponding confidence scores. The confidence score reflects the likelihood that a bounding box contains an object and how accurate that box is.
  + **Class Prediction:** Each grid cell also predicts class probabilities for the objects that appear in that cell.
  + **Non-Max Suppression:** This technique is used to eliminate duplicate bounding boxes, retaining only the box with the highest confidence score for each detected object.
* **Versions of YOLO:**
  + YOLO has evolved through various versions (YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, and YOLOv6), each improving upon the last in terms of accuracy and speed.



**YOLOv1—You Only Look Once**



**4. Working/Algorithm Used**

**4.1 Data Collection:**

* Use datasets like COCO (Common Objects in Context) or Pascal VOC, which are commonly used for training object detection models. Pretrained models can also be loaded that have already been trained on these datasets.

**4.2 Implementation Steps:**

1. **Environment Setup:**
   * Install the required libraries such as OpenCV, TensorFlow, or PyTorch, depending on the YOLO version used.
2. **Load Pretrained Model:**
   * Load the pretrained YOLO model weights and configuration files. For instance, YOLOv3 can be loaded using Darknet or OpenCV.
3. **Image Preprocessing:**
   * Resize the input image to the required dimensions (e.g., 416x416 for YOLOv3) while maintaining the aspect ratio.
   * Normalize pixel values and convert the image to a format suitable for the model.
4. **Object Detection:**
   * Pass the preprocessed image through the YOLO model to obtain predictions.
   * The model outputs bounding boxes, class probabilities, and confidence scores.
5. **Post-processing:**
   * Apply non-max suppression to filter out duplicate bounding boxes based on their confidence scores.
   * Draw bounding boxes and labels on the original image.
6. **Visualizing Results:**
   * Use libraries such as Matplotlib or OpenCV to display the image with detected objects and their corresponding bounding boxes.

**4.4 Evaluation Metrics:**

* **Mean Average Precision (mAP):** A common metric used to evaluate the accuracy of object detectors. It considers both precision and recall.
* **Intersection over Union (IoU):** Measures the overlap between the predicted bounding box and the ground truth bounding box.
* import torch
* import cv2
* import matplotlib.pyplot as plt
* # Load YOLOv5 model (pretrained on COCO dataset)
* model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)
* # Load the image
* img\_path = '/content/PHOTO (1).jpg'  # Replace with the path to your image
* img = cv2.imread(img\_path)
* # Perform object detection
* results = model(img)
* # Display results
* results.show()  # This will show the image with bounding boxes and labels.
* # Optionally, save the result image
* results.save('output/')  # Save the results with bounding boxes
* # To get more detailed results, such as class labels, confidence scores, and bounding boxes:
* df = results.pandas().xyxy[0]  # Results as pandas dataframe
* print(df)



**5. Conclusion**

In this project, we successfully implemented an object detection system using the YOLO algorithm with a pretrained model. The YOLO model demonstrated efficient and accurate real-time object detection capabilities, enabling the identification and localization of multiple objects within images. The results showed that YOLO is a robust and scalable solution for various object detection tasks. Further improvements could involve fine-tuning the model on specific datasets or exploring the latest YOLO versions for enhanced performance.