Topic:

Object Detection

Submitted by:

Name: Shahkar Javid

Matricola: 2047305

Email: javid.2047305@studenti.uniroma1.it

Name: Syed Shahihan

Matricola: 2048059

Email: syedmohamed.2048059@studenti.uniroma1.it

Submitted To:

Prof. Alessandro Giuseppi

Prof. Luca Benvenuti

1. INTRODUCTION

In this report we try explain and implement object detection algorithm. Object detection, in terms of deep learning, refers to the task of identifying and localizing multiple objects within an image or a video. It involves detecting and classifying objects present in the input data, as well as providing their precise spatial coordinates, usually in the form of bounding boxes.

2. Description

Deep learning-based object detection methods have achieved significant advancements and are widely used due to their ability to handle complex visual patterns and variations. These methods typically employ convolutional neural networks (CNNs) as their backbone architecture, leveraging their ability to extract meaningful features from raw pixel data.

There were many different methods and architecture were introduced for this specific task but here we have chosen to work on YOLO (you only look once) architecture. YOLO (You Only Look Once) is a popular object detection architecture that can detect multiple objects in an image or video in real-time. It's known for its simplicity and efficiency, providing a good trade-off between accuracy and speed.

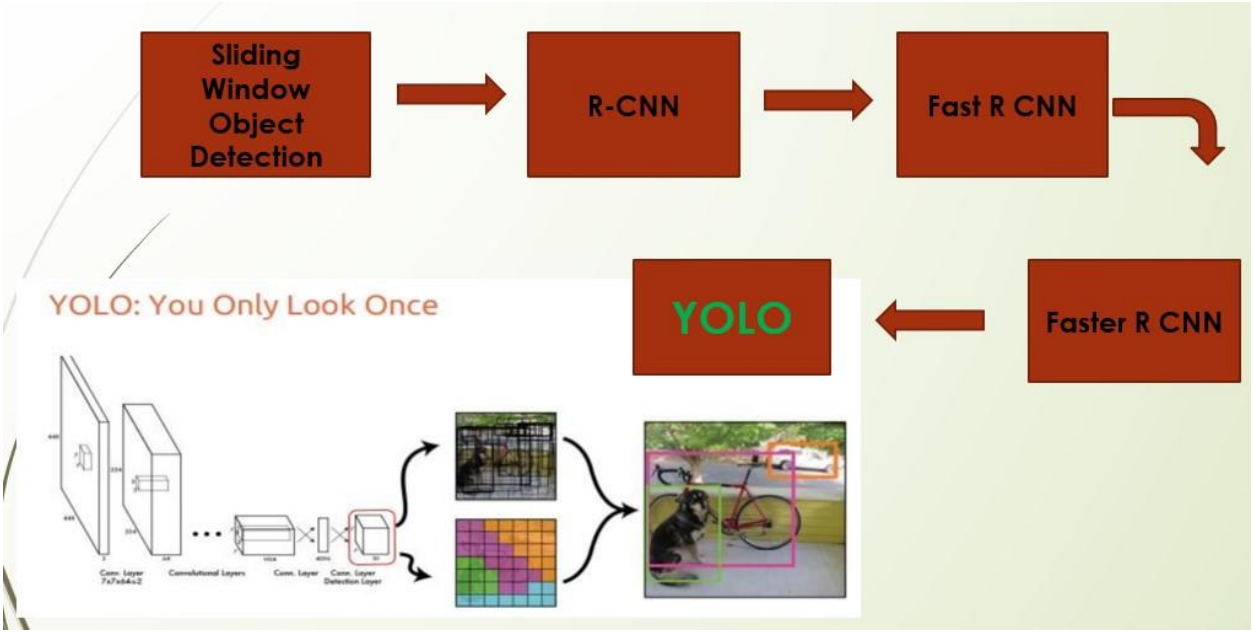
Before the advent of YOLO (You Only Look Once), several architectures were commonly used for object detection. Some of the notable ones include:

R-CNN (Region-based Convolutional Neural Networks): R-CNN is one of the pioneering architectures in object detection. It consists of three main steps: region proposal generation, feature extraction using a convolutional neural network, and classifying the proposed regions using support vector machines (SVMs). R-CNN achieved good accuracy but was relatively slow due to its multi-stage pipeline.

Fast R-CNN: Fast R-CNN was introduced as an improvement over R-CNN to address its speed limitations. Instead of extracting features separately for each region proposal, Fast R-CNN performs feature extraction on the entire image and then crops the features using region proposals. This reduces the computation time and allows for end-to-end training of the network.

Faster R-CNN: Faster R-CNN further enhances the speed and accuracy of object detection by introducing a Region Proposal Network (RPN). The RPN generates region proposals directly from the convolutional feature maps, eliminating the need for an external region proposal generation step. The RPN and the object classification network are trained jointly, enabling end-to-end learning.

These architectures were influential in the field of object detection and paved the way for the development of YOLO, which introduced a novel and efficient approach by framing object detection as a regression problem.



2.2 ARCHITECTURE

The YOLO architecture approaches object detection as a regression problem, enabling it to detect objects in real-time with a single pass through the neural network.Here is a detailed explanation of the YOLO architecture:

Input Processing:

YOLO takes an input image and resizes it to a fixed size, typically dividing it into a grid.

The resized image is divided into cells, and each cell is responsible for predicting bounding boxes and object classes.

Backbone Network:

YOLO uses a convolutional neural network (CNN) as its backbone. The backbone network processes the resized input image and extracts meaningful features through several convolutional layers. These convolutional layers capture low-level features like edges and textures, as well as higher-level features like shapes and object parts.

Grid Division:

The feature map generated by the backbone network is divided into an S x S grid, where S represents the number of cells. Each cell in the grid is responsible for predicting bounding boxes and object classes.

Bounding Box Prediction:

Each grid cell predicts a fixed number of bounding boxes. For each bounding box, YOLO predicts four coordinates: (x, y) representing the center of the box relative to the cell, width (w), and

height (h) relative to the entire image.The predicted coordinates are normalized between 0 and 1, where (0,0) represents the top-left corner of the image, and (1,1) represents the bottom-right corner.

Objectness Score:

Each bounding box prediction also includes an objectness score that represents the likelihood of an object being present in that box.The objectness score indicates how well the bounding box overlaps with ground truth objects in the training data.

Class Prediction:

Each grid cell predicts the probabilities of different object classes present in the bounding boxes. YOLO utilizes a softmax function to calculate the class probabilities, ensuring that the sum of class probabilities in each cell is equal to 1.

Non-Maximum Suppression (NMS):

To eliminate duplicate detections and refine the final set of predictions, YOLO applies a post-processing step called NMS.NMS compares the abjectness scores of bounding boxes and removes redundant overlapping boxes based on a predefined threshold. It selects the bounding box with the highest objectness score as the final prediction for an object in the scene.

Training:

YOLO is trained on a labeled dataset where each object is annotated with its bounding box coordinates and class labels. During training, YOLO calculates the loss based on the predicted bounding boxes, objectness scores, and class probabilities, comparing them to the ground truth annotations.

The loss function incorporates multiple components, including localization loss (regression loss for bounding box coordinates), confidence loss (objectness score), and classification loss (class probabilities).The network parameters are updated using backpropagation and gradient descent to minimize the loss.

By combining efficient architecture design, grid-based prediction, and regression-based approach, YOLO achieves impressive speed and accuracy for object detection tasks. It has been widely adopted in various applications, including autonomous driving, surveillance, and real-time video analysis.

2.3 Training on a Custom Data

Dataset Annotation:

We train our model for two task one for PPE detection(helmet, safety glasses and vest) and one for satellite object detection.

For training we need annotated images with bounding boxes coordinate of each class in an image
Labelimg was used for annotating images of my task.[1] LabelImg is a popular open-source graphical annotation tool used for labeling and annotating object detection datasets. It provides an intuitive user interface that allows users to manually annotate images by drawing bounding boxes around objects of interest and assigning corresponding class labels. LabelImg generates annotations in formats compatible with YOLO

Here's how LabelImg annotates image data for YOLO:

Installation and Setup: Download and install LabelImg on your system from the official GitHub repository or other trusted sources.Launch the application and specify the folder where your images are located.

Image Loading: Open an image in LabelImg. The image will be displayed in the main window.

Annotation Process: Select a class label from the available list or add a new label if necessary. Examples of class labels could be "car," "person," "dog," etc.Using the annotation tools provided by LabelImg, draw a bounding box around an object of interest in the image. Adjust the position and size of the bounding box as needed to tightly enclose the object. Repeat the process to annotate all instances of the same class in the image. f there are multiple classes in the image, switch to the corresponding class label and annotate objects of other classes. Continue annotating objects in subsequent images in the dataset until all desired images are labeled.

Annotation Saving: After annotating an image, click on the "Save" button to save the annotations. LabelImg saves the annotations as an XML file or in the Darknet format, depending on the selected output format. The annotations include the class label, coordinates of the bounding box (x, y, width, height), and other relevant information.

Exporting Annotations for YOLO: Once you have labeled and annotated all the images in your dataset, you can export the annotations in the YOLO-compatible format. LabelImg allows you to export annotations as text files with the same name as the corresponding images, but with different extensions (e.g., ".txt"). Each exported text file contains lines representing annotated objects, where each line consists of the class index followed by the normalized bounding box coordinates (center x, center y, width, height).

By using LabelImg to annotate image data, you can generate the necessary annotations required for training a YOLO-based object detection model. These annotations provide the ground truth information about the objects present in the images, allowing the model to learn to detect and classify objects accurately.

Loading Pretrained YOLO Model: After annotation of our dataset we load pretrained model of yolov5 from github [2] and install required dependencies like pytorch, cudnn, pip, torchvision, opencv, pillow, scipy , pandas etc etc.

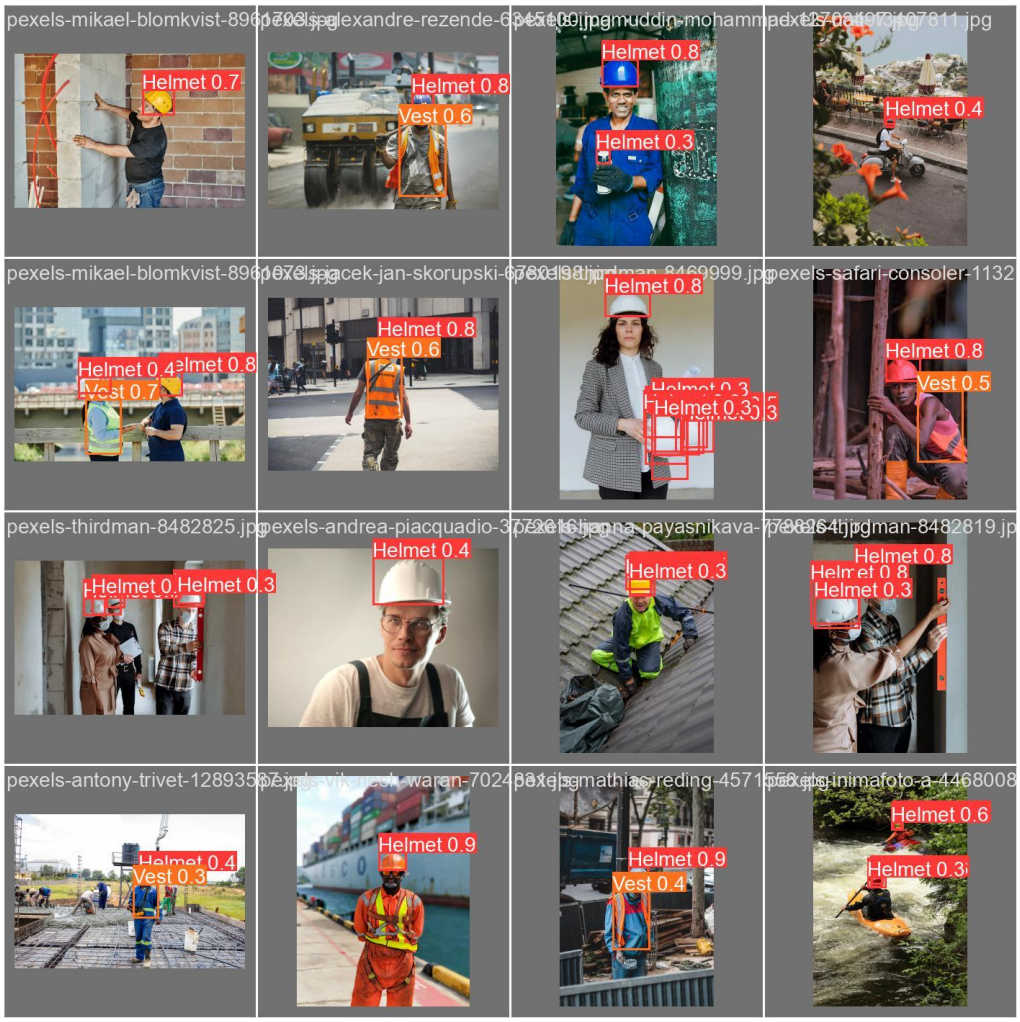
Testing pretrained model: After loading a pretrained model , we test the model on a random image to see if it is working or not. Below is the a=inage of cars on highway which were identified decently by our yolo model.

After successfully testing we load our custom data for our detection task both for PPE and satellite separately. We trained our model for around 100 epoch for PPE and for around 50 epoch for setelite object detection

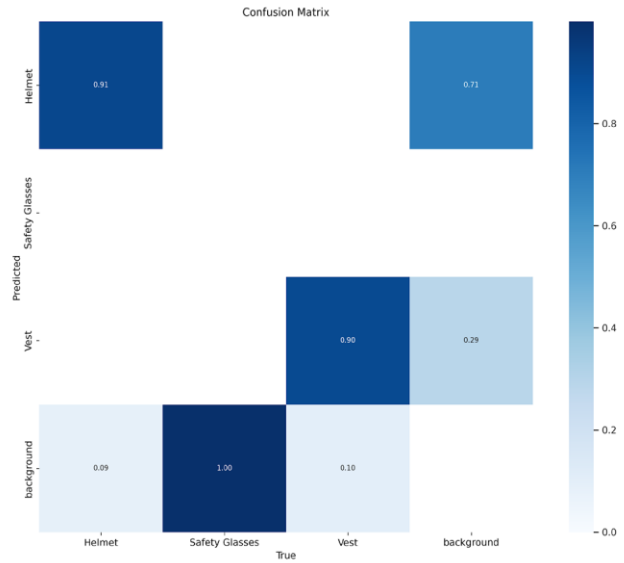
3. Results and Conclusion

After completion of epoch the visual result that can be seen below are decent and acceptable. In F1 confidence curve we can see that category of safety glasses weren't as much detected as other classes. The only reason is because of the dataset and annotation. We do not have the enough dataset for safety glasses. We also tried our modle on video which was also working pretty nice.

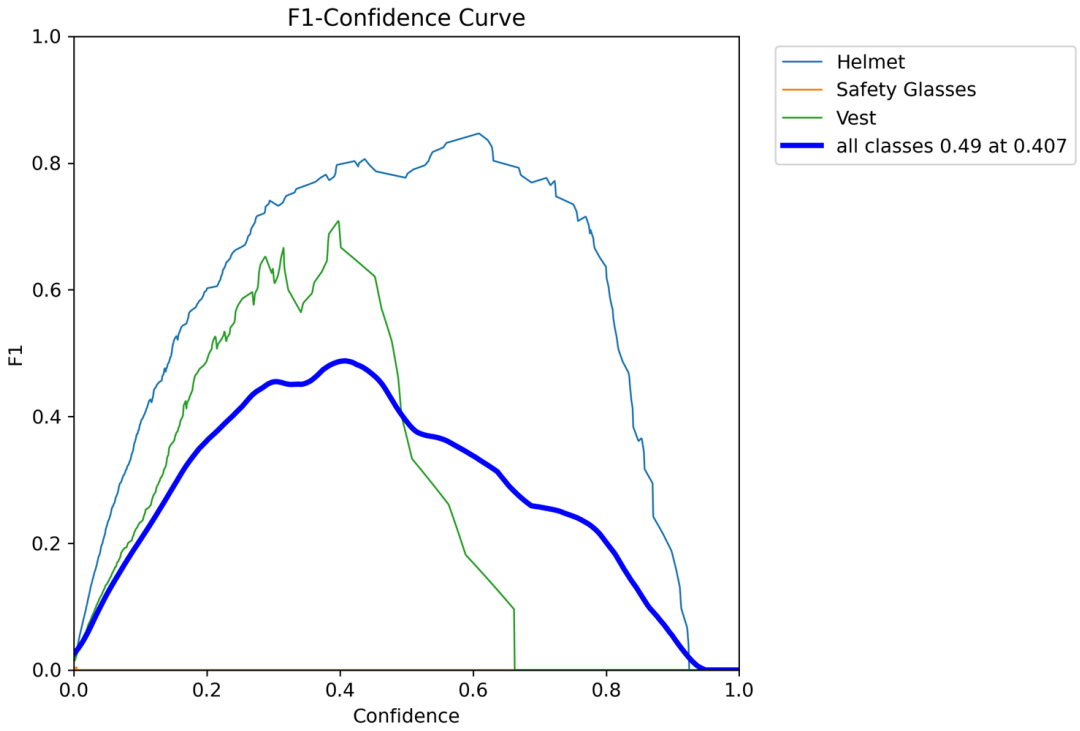
Helmet, Glasses and Vest Detection:



Confusion Matrix:



F1 Score Graph:



Satellite object Detection:

We did same for our setelite object detection and below result was observed.

4. Limitation of YOLO Model

- Since each grid has to decide whether it contains an object or not, very small objects may not be recognized as distinct from the rest of the image at all.
- Each grid can only contain a small number of objects, so if there are several objects close together, it may only recognize one of them.
- Despite these disadvantages, YOLO is a compelling object detection system due to its speed and the fact that it sacrifices very little accuracy in return. It does a great job even with fast-moving video files

5. Application of Object Detection

There are countless application nowadays of object detection some of them are mentioned below.

- **Autonomous Driving**: The invention of object detection is what has allowed self-driving cars to function on roads. Vehicles rely on this technology to inform them when pedestrians are crossing and when other cars are nearby.

- **Counts the Number of People/Cars/traffic**: Unless you have someone willing to sit and count every person/car that is in a space, you can never know for certain the number of of dynamic objects. The crowd counting feature can be helpful if you work in law enforcement and you need to control theme parks, concerts, and other big venues.

- **Healthcare Medical Detection**: Healthcare organizations can also benefit from object detection. As many medical diagnostics require images to study, scans for patients and other photographs this technology is a breakthrough.
- **Inventory Management:** Object detection applications trained to identify every item in a store's inventory can instantly alert employees when items need to be restocked.

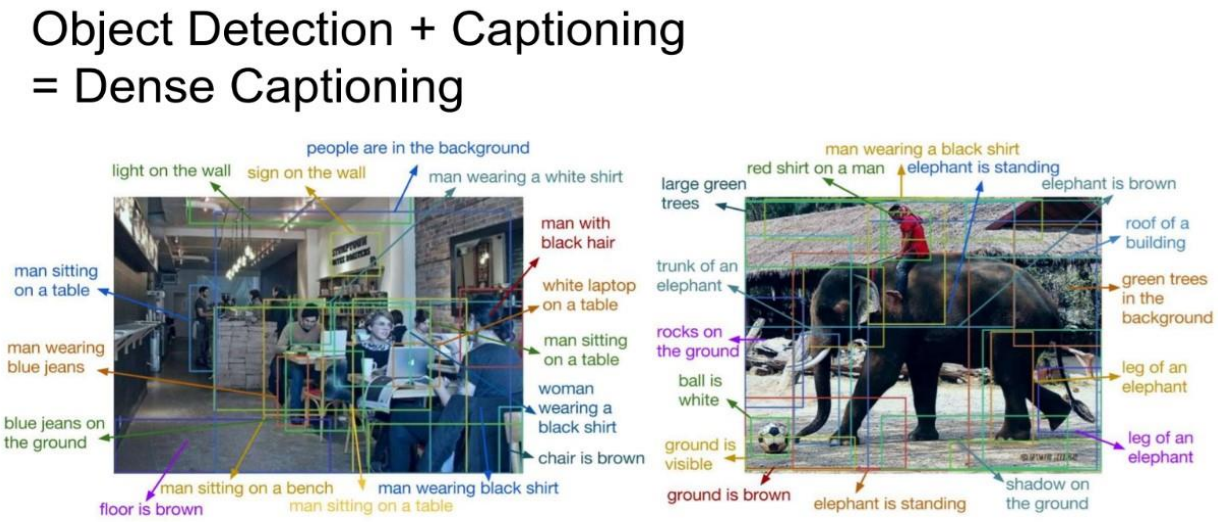
- **Anomaly and Defect Detection**: Object detection can identify parts and finished products that don't meet quality standards.

- And Many more…….

6. Further Extension Object Detection Task

Dense Captioning:

The goal of dense image captioning is to not only generate a caption but also localize and describe specific objects, activities, or relationships within the image. This requires the model to have an understanding of the image's content and spatial relationships between objects.



Instance Segmentation and Object Detection:

Instance segmentation and object detection can be combined in a single model to perform both tasks simultaneously. This combined approach is often referred to as instance-aware object detection or instance segmentation with object detection.

7. REFRENCES

Below are the references that were used in this project;

[1]. https://github.com/heartexlabs/labelImg

[2]. https://github.com/ultralytics/yolov5