OBJECT DETECTION USING YOLOV8

PROJECT REPORT

Presented By
MANIK SINGH

Introduction

This project is the part of final submission of the Internship programme at **DRDO's Young Scientist Lab-AI**, which commenced on May 28, 2023. Throughout the duration of this programme, I had the privilege of being mentored by **Sc. Sumit Sarkar and Sc**. **Rahul**, to whom I would like to express my sincere gratitude for their invaluable insights and guidance.

The focus of my project revolved around the field of object detection using computer vision techniques. While there exist numerous algorithms for object detection, I specifically employed the YOLOv8 algorithm for this undertaking. The reason behind selecting YOLOv8 was its status as the latest version, boasting the highest speed for detection and currently offering the highest accuracy among available alternatives.

Object detection through computer vision plays a crucial role in various applications, enabling machines to identify and locate objects within images or video frames. By employing YOLOv8, I aimed to enhance the efficiency and effectiveness of object detection processes, leveraging the algorithm's advanced capabilities.

Throughout the course of this project, I thoroughly explored and implemented the YOLOv8 algorithm, analyzing its intricacies and optimizing its performance. I conducted extensive experiments and evaluations to validate the algorithm's accuracy and efficiency in detecting objects across diverse datasets.

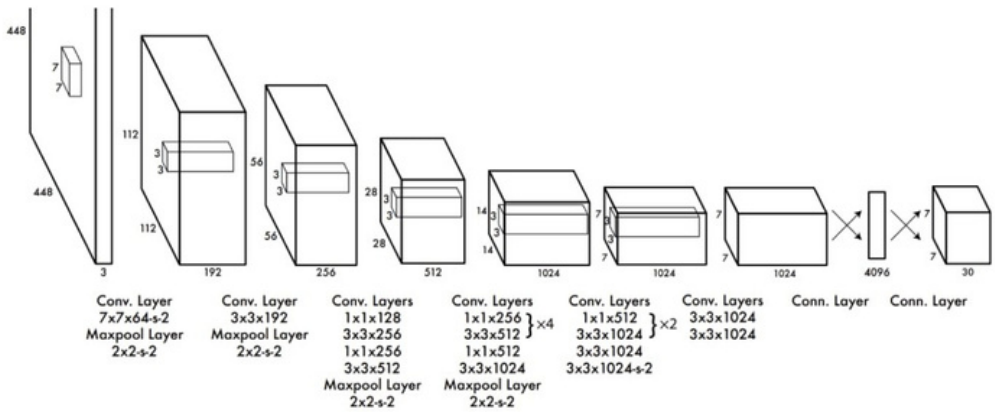
About

This project aims for detecting objects in **satellite imagery**. The ability to accurately identify and locate objects within satellite images holds immense value in a variety of fields, ranging from urban planning and environmental monitoring to disaster management and military reconnaissance.

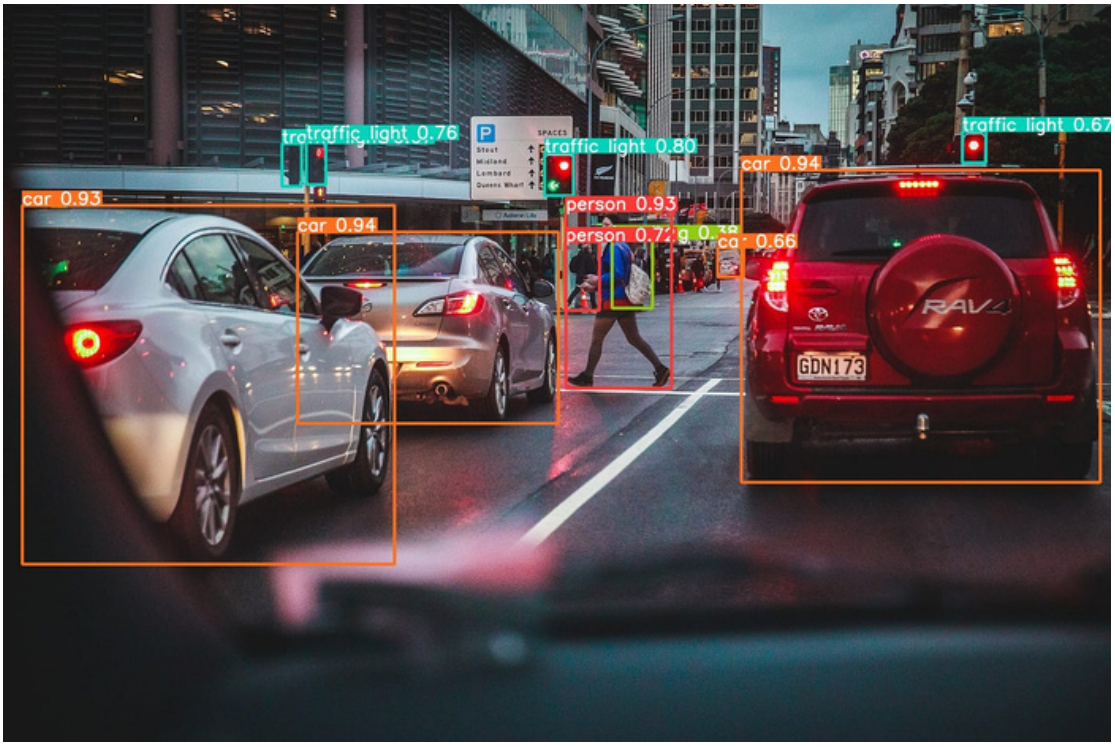
The detection of objects in satellite imagery involves the use of advanced computer vision techniques and algorithms. These techniques enable the system to analyze the vast amounts of visual data captured by satellites and identify specific objects or classes of objects with a high degree of accuracy.

To achieve the desired outcomes, I have implemented **YOLO(you only look once) algorithm** specifically designed for precise analysis. It proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection. Methods that use Region Proposal Networks perform multiple iterations for the same image, while YOLO gets away with a single iteration.

About the algorithm 'YOLO':
The YOLO algorithm takes an image as input and then uses a simple deep convolutional neural network to detect objects in the image. The architecture of the CNN model that forms the backbone of YOLO is shown below.



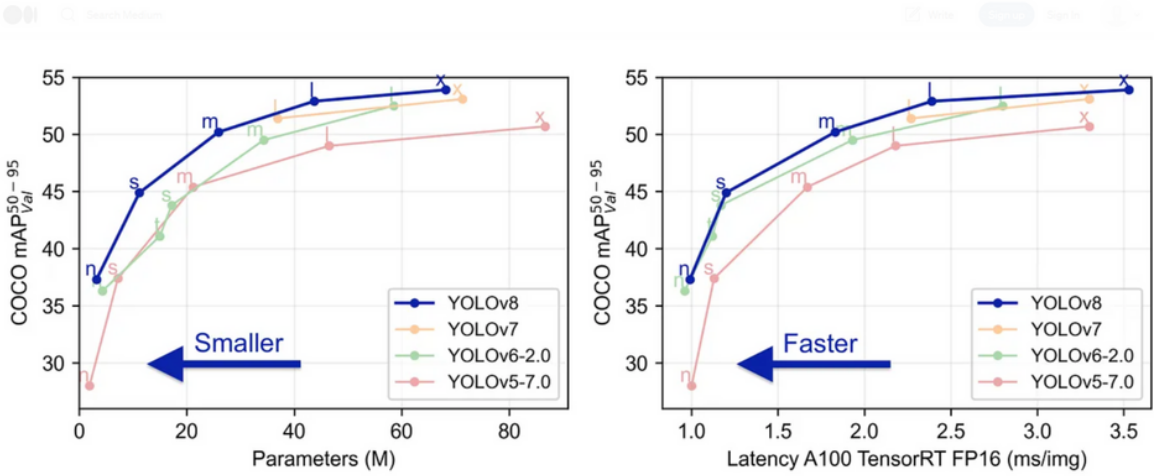
The first **20 convolution layers** of the model are pre-trained using ImageNet by plugging in a temporary average pooling and fully connected layer. Then, this pre-trained model is converted to perform detection since previous research showcased that adding convolution and connected layers to a pre-trained network improves performance. YOLO's final fully connected layer predicts both class probabilities and bounding box coordinates.



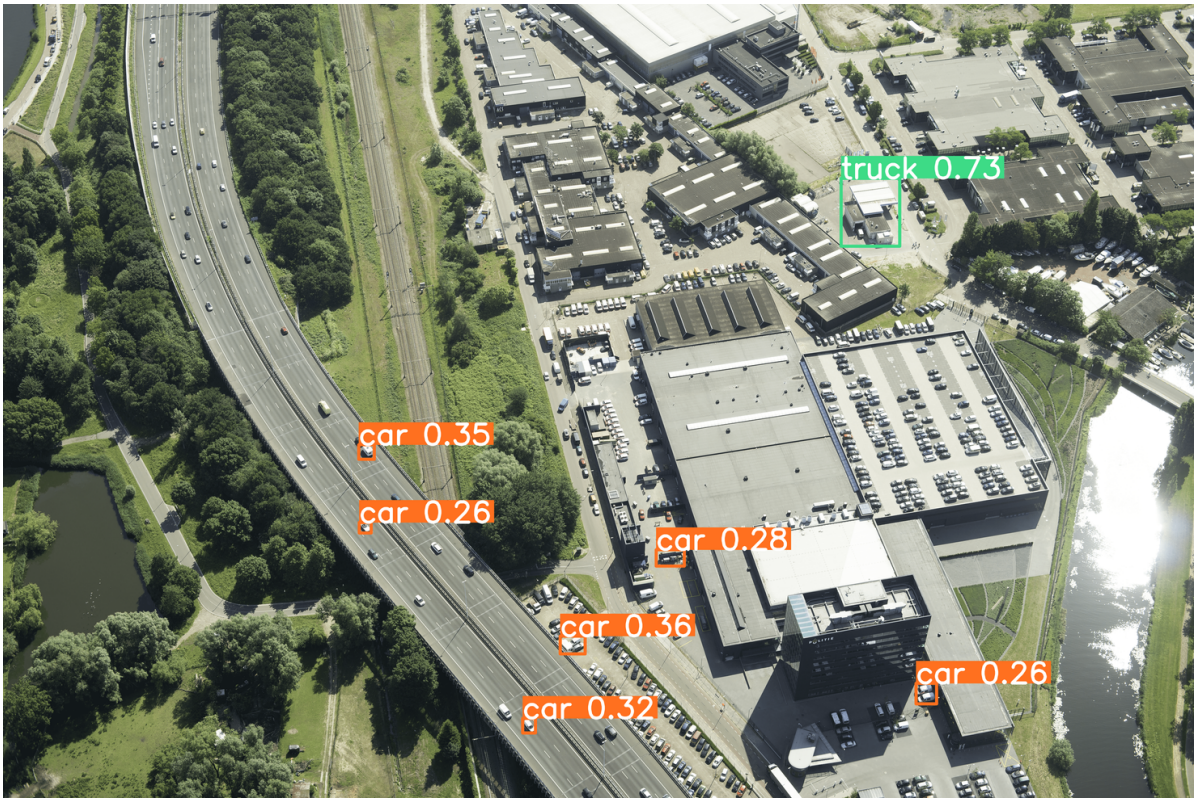
YOLO divides an input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and how accurate it thinks the predicted box is.

YOLO predicts multiple bounding boxes per grid cell. At training time, we only want one bounding box predictor to be responsible for each object. YOLO assigns one predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at forecasting certain sizes, aspect ratios, or classes of objects, improving the overall recall score.
This algorithm has undergone several iterations and improvements over time. In this particular project, the selected model for implementation is YOLOv8.

YOLOv8 represents the latest version and a notable advancement of the YOLO algorithm. It stands out among other models due to its exceptional performance in terms of both speed and accuracy. By leveraging cutting-edge techniques and architectural enhancements, YOLOv8 has further refined the capabilities of object detection, making it a compelling choice for this project



Project Prediction



Contributions

The project was trained using a unique dataset that was specifically created for its objectives. The dataset creation process involved utilizing an online labeling tool called "CVAT," and the resulting dataset can be accessed via the following link: 'https://app.cvat.ai/tasks/224205/jobs/224516 '.
The custom dataset, generated through the aforementioned labeling tool, played a crucial role in training the model. By incorporating this dataset during the training phase, the model was able to learn from the labeled examples and improve its ability to accurately detect objects.

```python
from ultralytics import YOLO
import multiprocessing

def main():
    # Load a model
    model = YOLO("yolov8x.yaml")  # build a new model from scratch
    #model = YOLO("yolov8n.pt")  # load a pretrained model (recommended for training)

    # Use the model
    model.train(data="config.yaml", epochs=100)  # train the model

if __name__ == '__main__':
    multiprocessing.freeze_support()
    main()
```

For the training of YOLOv8, there are two available approaches: using the command-line interface (CLI) or employing Python scripts. In this project, the Python script was utilized for training the custom model. The script provided the necessary instructions and configurations to train the YOLOv8 model with the specifically curated dataset, enabling the model to adapt to the target object detection task effectively.

```yaml
path: C:\Users\Manik\OneDrive\Desktop\EXP2  # dataset root dir
train: C:\Users\Manik\OneDrive\Desktop\EXP2\images\train  # train images (relative to 'path')
val: C:\Users\Manik\OneDrive\Desktop\EXP2\labels\train  # val images (relative to 'path')

# Classes
names:
  0: Building
  1: car
  2: pearson
  3: tree
  4: boat
  5: dock
  6: ground
  7: trees
  8: water
```

References:

1. YOLOv8 repository by Ultralytics
2. About Yolo
3. Dataset
4. Custom Train
5. My Project