YOLOv4 and it’s application for Helmet detection.

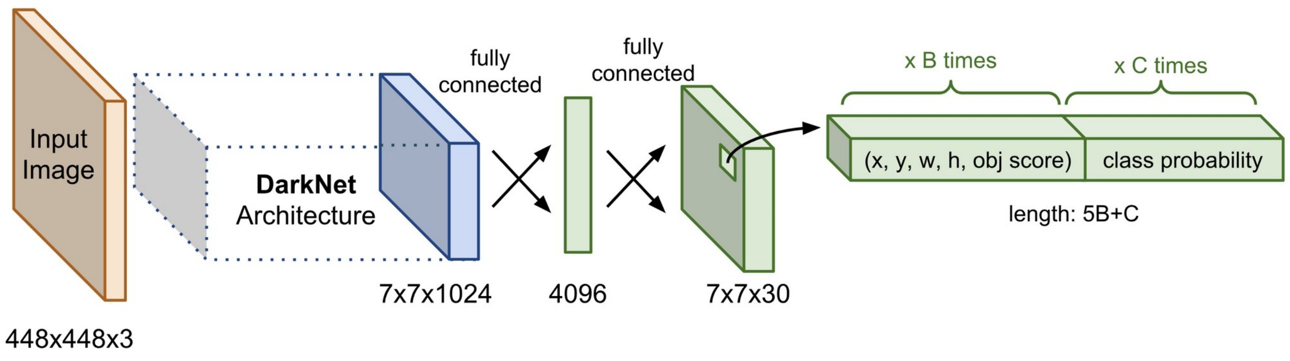
-Chanda Kumari
-IIT Roorkee

Abstact:

In Atm, we already know that nobody is allowed to enter with Helmet. So in order to maintain security purpose a helmet detection method based on improvement YOLOv4 is proposed by collection of a self-made dataset. It’s detection accuracy and detection speed were improved compared with YOLOv4, which satisfy the real time requirements of the helmet detection task.

What is yoloV4?

YOLOv4 is an object detection algorithm that is an evolution of the YOLOv3 model. The YOLOv4 method was created by Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. It is twice as fast as EfficientDet with comparable performance. These are basically updated version of YOLO algorithm. So let’s talk about Yolo first. YOLO was proposed by Joseph Redmond *et al.* in 2015. It was proposed to deal with the problems faced by the object recognition models at that time, Fast R-CNN is one of the state-of-the-art models but it has its own challenges such as this network cannot be used in real-time, because it takes 2-3 seconds to predicts an image and therefore cannot be used in real-time.
The Yolo algorithm stands for You Only Look Once, this algorithm is a state of art, which works on a real-time system, build on deep learning for solving various Object Detection as well as Object Tracking problems. The architecture of Yolo can be observed from the below Fig:



More about YOLO:

Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images. YOLO algorithm employs convolutional neural networks (CNN) to detect objects in real-time. As the name suggests, the algorithm requires only a single forward propagation through a neural network to detect objects.
This means that prediction in the entire image is done in a single algorithm run. The CNN is used to predict various class probabilities and bounding boxes simultaneously.

Why the YOLO algorithm is important:

YOLO algorithm is important because of the following reasons:

- **Speed:**This algorithm improves the speed of detection because it can predict objects in real-time.
- **High accuracy:** YOLO is a predictive technique that provides accurate results with minimal background errors.
- **Learning capabilities:** The algorithm has excellent learning capabilities that enable it to learn the representations of objects and apply them in object detection.

How the YOLO algorithm works:

YOLO algorithm works using the following three techniques:

- Residual blocks
- Bounding box regression
- Intersection Over Union (IOU)

Residual blocks

First, the image is divided into various grids. Each grid has a dimension of S x S.

Bounding box regression

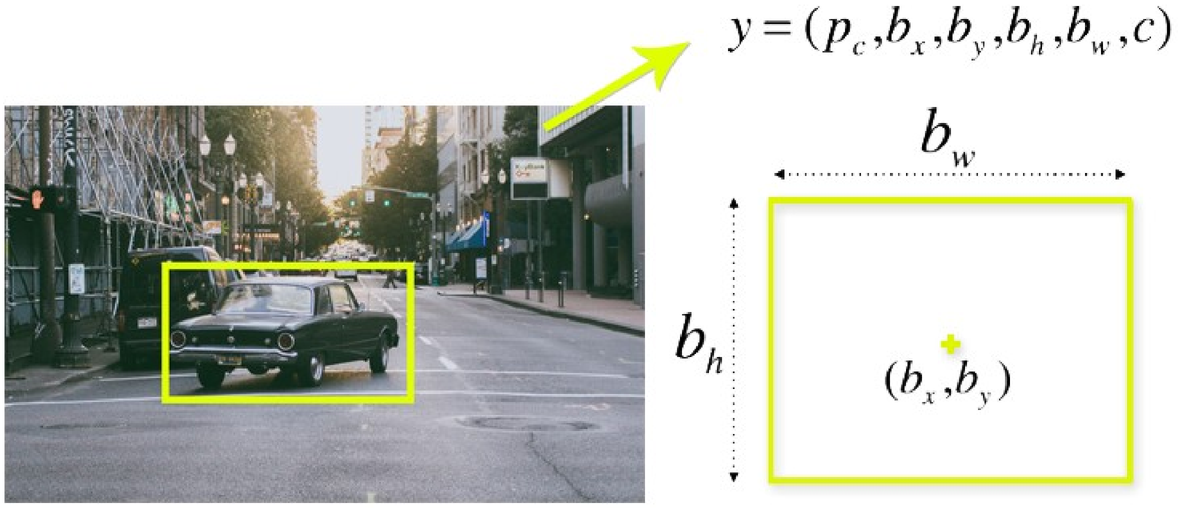
A bounding box is an outline that highlights an object in an image.

Every bounding box in the image consists of the following attributes:

- Width (bw)
- Height (bh)
- Class (for example, person, car, traffic light, etc.)- This is represented by the letter c.

•Bounding box center (bx,by)

The following image shows an example of a bounding box. The bounding box has been represented by a yellow outline.

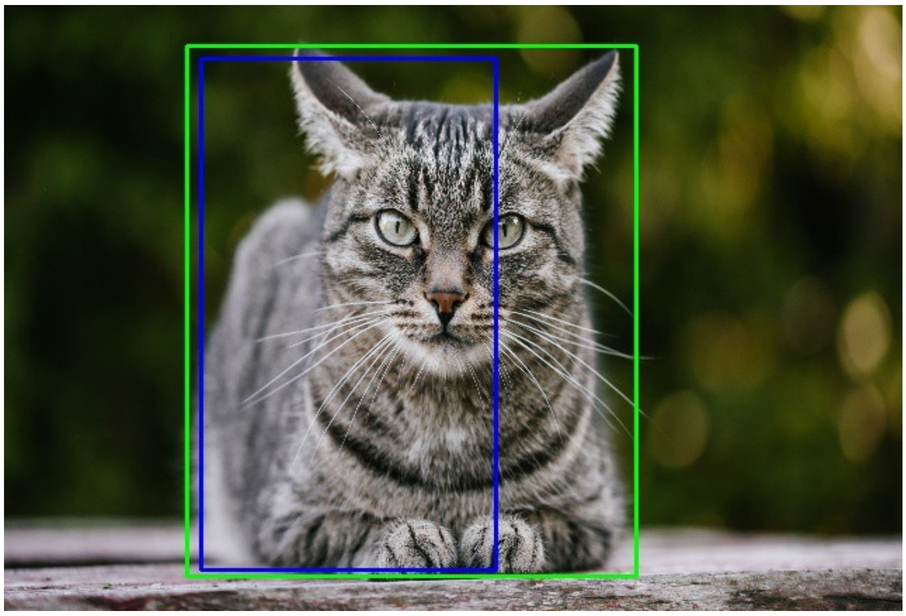


Intersection over union (IOU)

Intersection over union (IOU) is a phenomenon in object detection that describes how boxes overlap. YOLO uses IOU to provide an output box that surrounds the objects perfectly.

Each grid cell is responsible for predicting the bounding boxes and their confidence scores. The IOU is equal to 1 if the predicted bounding box is the same as the real box. This mechanism eliminates bounding boxes that are not equal to the real box.

The following image provides a simple example of how IOU works.



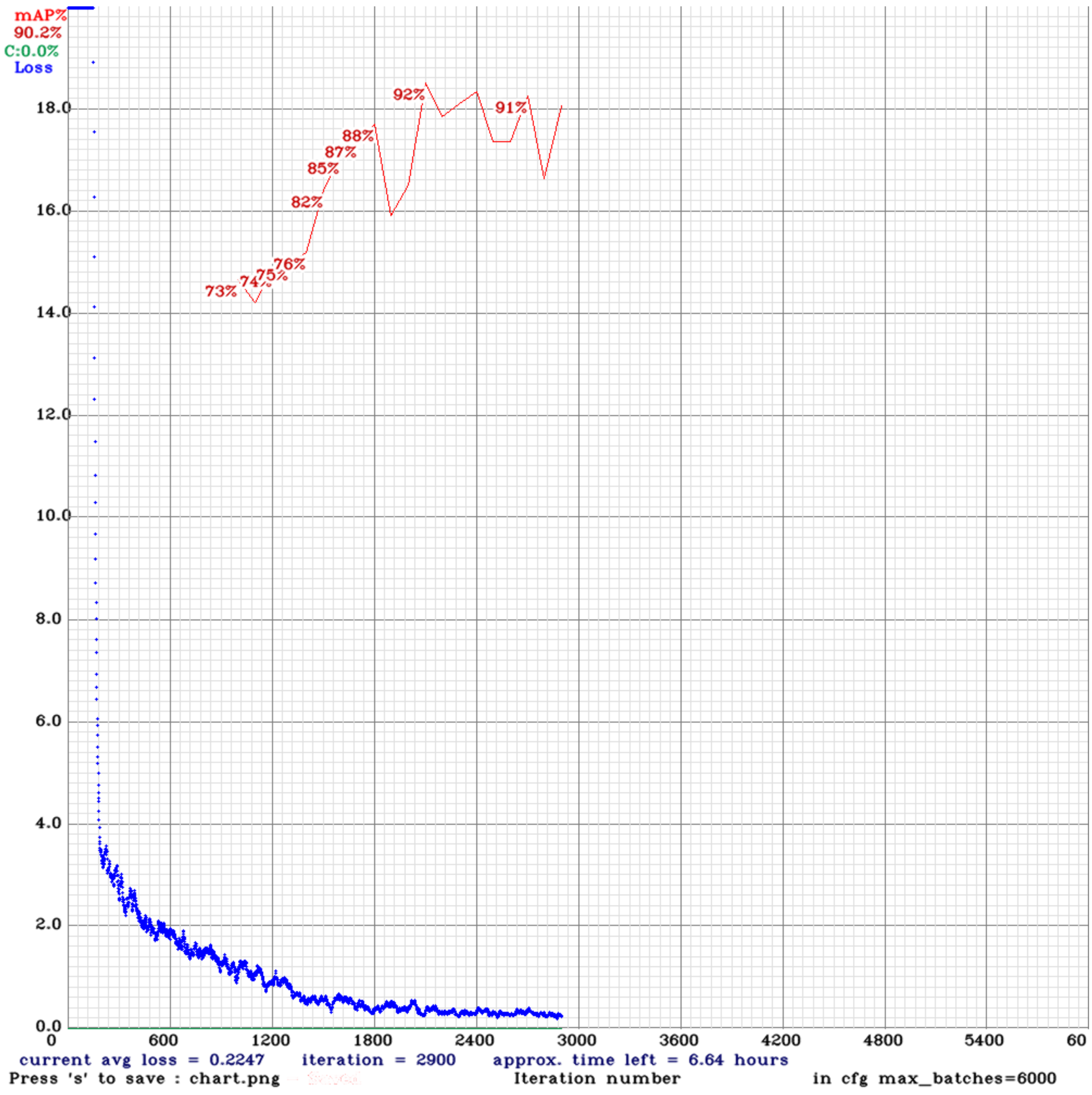
The processes of my project:

As I didn’t had the pretrained weight file for Helmet detection so I started from the beginning and done the following processes

1. Collected the dataset and did labeling.
2. Trained the dataset using Google colab because it provids free GPU.
3. Used OpenCV for detection.

I have uploaded the training part on github repository-https://github.com/chanda27/Helmet-detection-using-yolov4, you can get it here.

After training the accuracy chart of model:



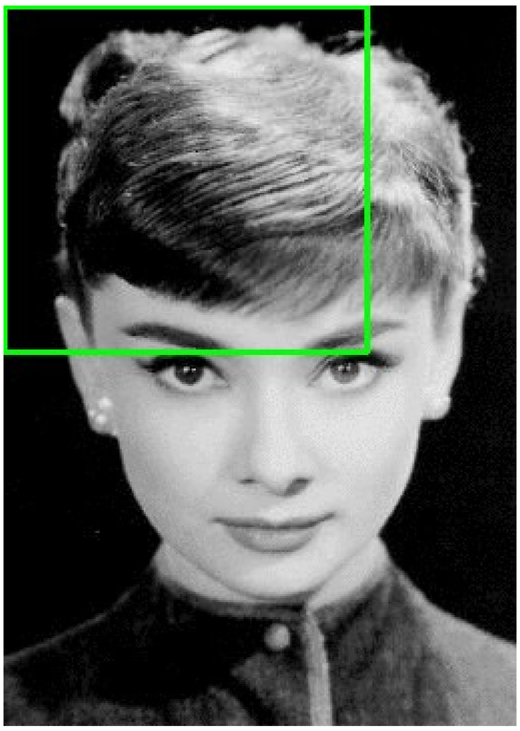
After training the dataset, got the weight file. Using OpenCV I did the detection part. Code is uploaded on github which you can get it here.
https://github.com/chanda27/Helmet-detection-using-yolov4/blob/main/Helmet%20detection%20using%20live.ipynb

Problems I faced during doing this:

When I research about it, like what are some algorithm available for object detection so I found out an algorithm called "Haar cascade classifier" and so I started to work on this. So let's talk about this.

What is Haar cascade classifier?

It is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features proposed by Viola and Jones in their research paper "Rapid Object Detection using a Boosted Cascade of Simple Features" published in 2001.



In this **figure**, we can see that we are sliding a **fixed size window** across our image at **multiple scales**. at each of these phases, our window stops, computes some features, and then classifies the region as *yes, this region does contain a face, or no, this region does not contain a face.*

This requires a bit of machine learning. We need a classifier that is trained in using positive and negative samples of a face:

1. Positive data points are examples of regions containing a face
2. Negative data points are examples of regions that *do not* contain a face

This is basically a machine learning approch for object detection and it works only when you will have OpenCV version less than OpenCV 3.0 but during installation I faced several problem and didn't get that version So my mentor told me to work on YOLOv3 algorithm, which follow a deep learning approach for object detection and it is also very fast and so then I started to work on this.

In YOLOv3 for training the custom dataset we use transfer learning which is pretrained for some sort of classes and we try to optimize on top of our own dataset which was darknet53.con74 weight file. But when I was training it was taking so much time like approx 400 hours which is so huge and also my colab got crashed. I was then studying about it and got to know about another pre-trained weights for the convolutional layers which was using in algorithm YOLOv4 and it worked nice. So then finaly I trained my model using YOLOv4 only and did the rest of this using OpenCV.