

PROJECT REPORT

(Project Term August-December 2021)

REAL TIME OBJECT DETECTION USING YOLO

Submitted by

M K ALISILE ALIAS ANUSRI

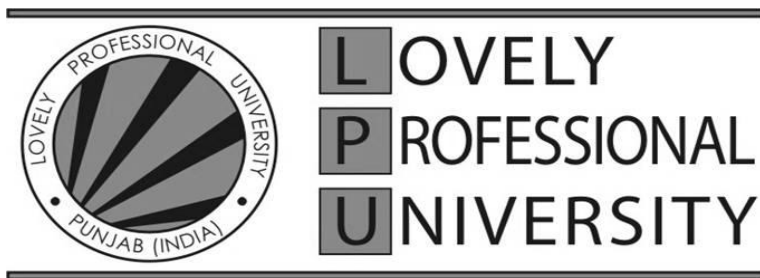
Registration Number :11912258.....

Course Code :INT246.....

Under the Guidance of

DR. SAGAR PANDE

School of Computer Science and Engineering



DECLARATION

I hereby declare that the project work entitled '**Real time Object detection using YOLO**' is an authentic record of work carried out as requirements of Project for the award of B.Tech degree in ____Computer Science____ from Lovely Professional University, Phagwara, under the guidance of Dr. Sagar Pande, during August to December 2021. All the information furnished in this project report is based on our own intensive work and is genuine.

Name of Student :M K Alisile alias Anusri.....

Registration Number:11912258.....

M K Alisile alias Anusri

20 November 2021

CERTIFICATE

This is to certify that the declaration statement made by the student is correct to the best of my knowledge and belief. She has completed this Project under my guidance and supervision. The present work is the result of her original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Project is fit for the submission and partial fulfillment of the conditions for the award of B.Tech degree in Computer Science from Lovely Professional University, Phagwara.

Dr. Sagar Pande

Professor

School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab.

Date : 20 November 2021

ACKNOWLEDGEMENT

I am are grateful to Dr.Sagar Pande sir for giving me a wonderful project. I'm really fortunate for the kind association as well as supervision from sir. Dr.Sagar Pande sir, his exemplary guidance, constant encouragement, and careful monitoring helped me through out the project. I also take this opportunity to a express a deep sense of gratitude to him for giving me an amazing experience.

TABLE OF CONTENTS

1. INTRODUCTION	1
1.1. OBJECT DETECTION	1
1.1.1. CONCEPT	1
1.1.2. SCOPE OF TECHNOLOGY	1
1.2. YOLO	2
1.2.1. TWO STEP OBJECT DETECTION	2
1.2.2. YOLO VS OTHER ALGORITHMS	2
2. PROBLEM STATEMENT	3
3. EXISTING SYSTEM	4
3.1 INTRODUCTION	4
3.2 YOLO ARCHITECTURE	4
4. SOFTWARE REQUIREMENT ANALYSIS	5
4.1 DATA SET	5
4.2 YOLO	6
4.2.1 INTERSECTION OVER UNION (IoU)	6
4.2.2 AVERAGE PRECISION	6
4.2.3 THE DIFFERENCES: YOLO, YOLOv2, YOLOv3, YOLOv4+	7
4.2.4 HOW TO WORK WITH YOLO	10
4.2.5 OVERVIEW	10
5. DESIGN	11
5.1 STEPS INVOLVED	11
6. CODE SNIPPETS	12
7. BIBLIOGRAPHY	13

1. INTRODUCTION

Real time Object detection is a computer technology involving computer vision and image processing that deals with detecting instances of semantic objects of a certain class which could be of humans, buildings, cars, etc. in digital images and videos. Well researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

1.1 OBJECT DETECTION

Object detection is an advanced form of image classification where a neural network predicts objects in an image and points them out in the form of bounding boxes. It thus refers to the detection and localization of objects in an image that belong to a predefined set of classes.

1.1.1 CONCEPT

Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found.

1.1.2 SCOPE OF TECHNOLOGY

It is widely used in computer vision tasks such as image annotation, vehicle counting, activity recognition, face detection, face recognition, video object co-segmentation. It is also used in tracking objects, for example tracking a ball during a football match, tracking movement of a cricket bat, or tracking a person in a video.

1.2. YOLO

YOLO - You Only Look Once is an algorithm proposed by by Redmond et. al in a research article published at the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) as a conference paper, winning OpenCV People's Choice Award..

As the name states, YOLO algorithm can detect images very fast whereas traditional classifiers need to slide windows over the image multiple times to detect different classes. It is an extremely fast, real-time method of not only identifying but localizing objects up to an astounding 155 frames per second.

1.2.1 TWO STEP OBJECT DETECTION

Two-stage object detection refers to the use of algorithms that break down the object detection problem statement into the following two-stages:

1. Detecting possible object regions.
2. Classifying the image in those regions into object classes.

Popular two-step algorithms like Fast-RCNN and Faster-RCNN typically use a Region Proposal Network that proposes regions of interest that might contain objects.

YOLO on the other hand is proposal free and does object detection in one step.

1.2.1 YOLO VS OTHER ALGORITHMS

Compared to the approach taken by object detection algorithms before YOLO, which repurpose classifiers to perform detection, YOLO proposes the use of an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once.

Following a fundamentally different approach to object detection, YOLO achieves state-of-the-art results beating other real-time object detection algorithms by a large margin.

2. Problem Statement

To detect object instances from a image or a video in real time using the fast and recent technology of YOLO by creating a custom data set to process trained weights thereby creating a neural network model, under broad categories which are as follows:

1. Watch
2. Sea
3. Sky
4. Person
5. Bike

3. EXISTING SYSTEM

3.1 INTRODUCTION

The YOLO algorithm works by dividing the image into N grids, each having an equal dimensional region of $S \times S$. Each of these N grids is responsible for the detection and localization of the object it contains. Correspondingly, these grids predict B bounding box coordinates relative to their cell coordinates, along with the object label and probability of the object being present in the cell.

This process greatly lowers the computation as both detection and recognition are handled by cells from the image, but it brings forth a lot of duplicate predictions due to multiple cells predicting the same object with different bounding box predictions. YOLO makes use of Non Maximal Suppression to deal with this issue.

In Non Maximal Suppression, YOLO suppresses all bounding boxes that have lower probability scores. YOLO achieves this by first looking at the probability scores associated with each decision and taking the largest one. Following this, it suppresses the bounding boxes having the largest Intersection over Union with the current high probability bounding box.

This step is repeated till the final bounding boxes are obtained.

3.1 YOLO ARCHITECTURE

Inspired by the GoogleNet architecture, YOLO's architecture has a total of 24 convolutional layers with 2 fully connected layers at the end.

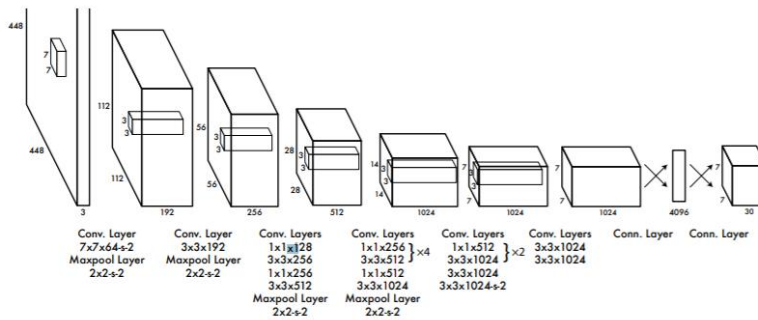
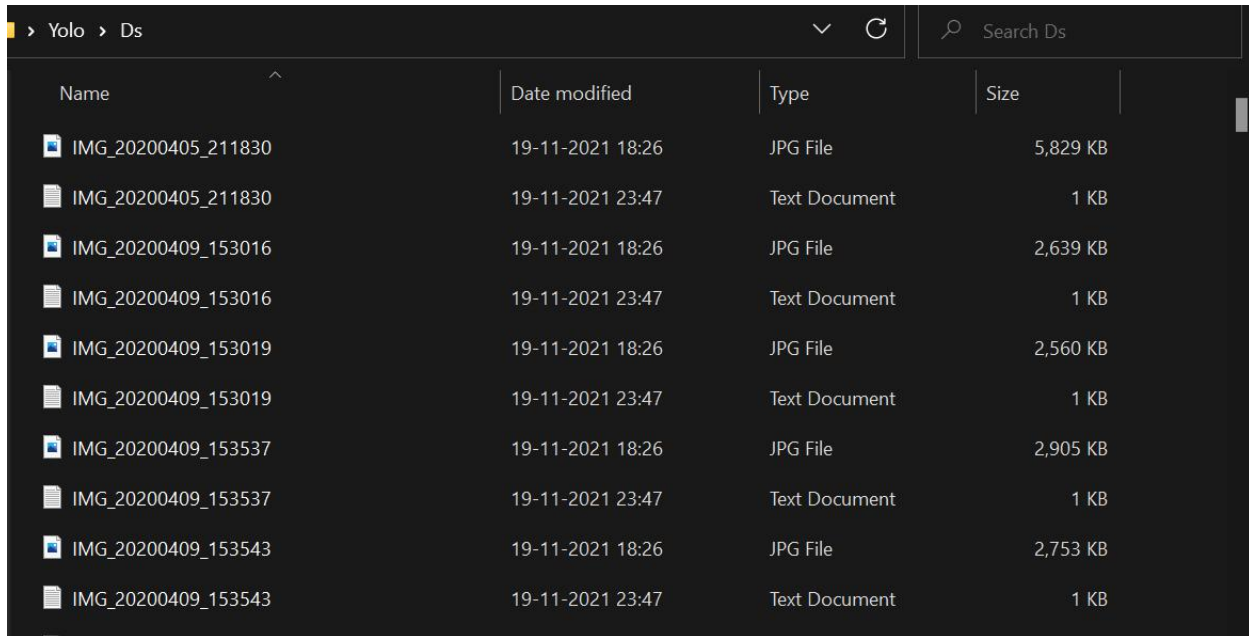


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

4. Software Requirement Analysis

4.1 Data Set

All the custom images to be trained into a data set needs to be in the YOLOv5 accepted format. Every image file is to be associated with a corresponding text file which denoted the box coordinates of the object.



Name	Date modified	Type	Size
IMG_20200405_211830	19-11-2021 18:26	JPG File	5,829 KB
IMG_20200405_211830	19-11-2021 23:47	Text Document	1 KB
IMG_20200409_153016	19-11-2021 18:26	JPG File	2,639 KB
IMG_20200409_153016	19-11-2021 23:47	Text Document	1 KB
IMG_20200409_153019	19-11-2021 18:26	JPG File	2,560 KB
IMG_20200409_153019	19-11-2021 23:47	Text Document	1 KB
IMG_20200409_153537	19-11-2021 18:26	JPG File	2,905 KB
IMG_20200409_153537	19-11-2021 23:47	Text Document	1 KB
IMG_20200409_153543	19-11-2021 18:26	JPG File	2,753 KB
IMG_20200409_153543	19-11-2021 23:47	Text Document	1 KB

Fig 4.1

Every Image and corresponding text file.

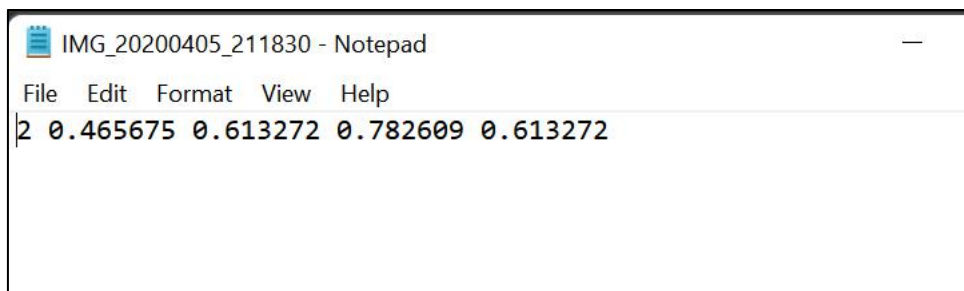


Fig 4.2

Text File for Image

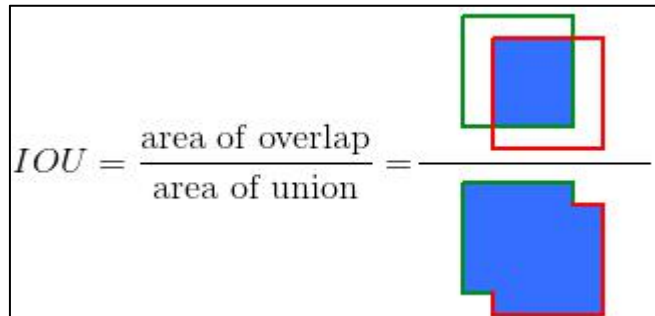
4.2 YOLO

4.2.1 Intersection over Union (IoU)

Intersection over Union is a popular metric to measure localization accuracy and calculate localization errors in object detection models.

To calculate the IoU with the predictions and the ground truth, we first take the intersecting area between the bounding boxes for a particular prediction and the ground truth bounding boxes of the same area. Following this, we calculate the total area covered by the two bounding boxes—also known as the *Union*.

The intersection divided by the Union, gives us the ratio of the overlap to the total area, providing a good estimate of how close the bounding box is to the original prediction.



4.2.2 Average Precision (AP)

Average Precision is calculated as the area under a precision vs recall curve for a set of predictions. Recall is calculated as the ratio of the total predictions made by the model under a class with a total of existing labels for the class.

On the other hand, Precision refers to the ratio of true positives with respect to the total predictions made by the model.

The area under the precision vs recall curve gives us the Average Precision per class for the model. The average of this value, taken over all classes, is termed as mean Average Precision (mAP).

In object detection, precision and recall are not for class predictions, but for predictions of boundary boxes for measuring the decision performance. An IoU value > 0.5 is taken as a positive prediction, while an IoU value < 0.5 is a negative prediction.

4.2.3 The differences: YOLO, YOLOv2, YOLO9000, YOLOv3, YOLOv4+

YOLOv2

YOLOv2 was proposed to fix YOLO's main issues—the detection of small objects in groups and the localization accuracy.

YOLOv2 increases the mean Average Precision of the network by introducing *batch normalization*. Batch Norm increases the mAP value by as much as 2 percent.

A much more impact addition to the YOLO algorithm, as proposed by YOLOv2, was the addition of anchor boxes. YOLO, as we know, predicts a single object per grid cell. While this makes the built model simpler, it creates issues when a single cell has more than one object, as YOLO can only assign a single class to the cell.

YOLOv2 gets rid of this limitation by allowing the prediction of multiple bounding boxes from a single cell. This is achieved by making the network predict 5 bounding boxes for each cell.

The number 5 is empirically derived as having a good trade-off between model complexity and prediction performance. DarkNet-19 containing a total of 19 convolutional layers and 5 max-pooling layers is used as the backbone for the YOLOv2 architecture.

YOLO9000

Using a similar network architecture as YOLOv2, YOLO9000 was proposed as an algorithm to detect more classes than COCO as an object detection dataset could have made possible.

The object detection dataset that these models were trained on (COCO) has only 80 classes as compared to classification networks like ImageNet which has 22.000 classes.

To enable the detection of many more classes, YOLO9000 makes use of labels from both ImageNet and COCO, effectively merging the classification and detection tasks to only perform detection.

Since some classes of COCO can be referred to as superset classes of some classes of ImageNet, YOLO9000 makes use of a hierarchical classification-based algorithm inspired by WordNet, where classes and their sub-classes are represented in a tree-based fashion.

While YOLO9000 provides a lower mean Average Precision as compared to YOLOv2, it is capable of detecting more than 9000 classes, making it a powerful algorithm.

YOLOv3

While YOLOv2 is a super fast network, various alternatives that offer better accuracies—like Single Shot Detectors—have also entered the scene. Although much slower, they outstrip YOLOv2 and YOLO9000 in terms of accuracy.

To improve YOLO with modern CNNs that make use of residual networks and skip connections, YOLOv3 was proposed.

Here's how it works as presented by Joseph Redmon.

While YOLOv2 uses the DarkNet-19 as the model architecture, YOLOv3 uses a much more complex DarkNet-53 as the model backbone— a 106 layer neural network complete with residual blocks and up sampling networks.

YOLOv3's architectural novelty allows it to predict at 3 different scales, with the feature maps being extracted at layers 82, 94, and 106 for these predictions..

By detecting features at 3 different scales, YOLOv3 makes up for the shortcomings of YOLOv2 and YOLO, particularly in the detection of smaller objects. With the architecture allowing the concatenation of the up sampled layer outputs with the features from previous layers, the fine-grained features that have been extracted are preserved thus making the detection of smaller objects easier.

YOLOv3 only predicts 3 bounding boxes per cell (compared to 5 in YOLOv2) but it makes three predictions at different scales, totaling up to 9 anchor boxes.

YOLOv4, YOLOv5, YOLACT, and future YOLOs

Joseph Redmond left the AI community a few years back, so YOLOv4 and other versions past that are not his official work. Some of them are maintained by co-authors but none of the releases past YOLOv3 is considered the “official” YOLO.

However, the legacy continues through new researchers.

YOLOv4 was proposed by Bochkovskiy et. al. in 2020 as an improvement to YOLOv3. The algorithm achieves state-of-the-art results at 43.5 % Average Precision running at 65 FPS on a Tesla v100 GPU.

These results are achieved by including a combination of changes in architectural design and training methodologies of YOLOv3.

YOLOv4 proposes the addition of Weighted Residual Connections, Cross Mini Batch Normalization, Cross Stage Partial Connections, Self Adversarial Training, and Mish Activation as methodological changes amongst modern methods of regularization and data augmentation. The authors also make available a YOLOv4 Tiny version that provides faster object detection and a higher FPS while making a compromise in the prediction accuracy.

YOLOv5 is an open-source project that consists of a family of object detection models and detection methods based on the YOLO model pre-trained on the COCO dataset. It is maintained by Ultralytics and represents the organization's open-source research into the future of Computer Vision works.

YOLACT (You Only Look At Coefficients) proposed by Bolya is an application of the YOLO principle for real-time instance segmentation.

In other words, YOLACT proposes an end-to-end convolutional network for instance segmentation that achieves 29.8 mean Average Precision at 33.5 FPS on a single Titan Xp, which is significantly faster than other instance segmentation algorithms.

YOLACT performs instance segmentation by generating a set of prototype masks and per-instance mask coefficients. A linear combination of the two steps is performed to generate the final instance masks.

4.2.4 How to work with YOLO

We won't get into the nitty-gritty of working with YOLO in this article, but here's a detailed guide for training YOLOv5 on your personalized dataset: [YOLOv5 Training Guide](#).

You can create and export datasets with V7 and train YOLOv5 for detecting specific category objects.

Additionally, there are pre-trained models available for download that you can use right away.

You can also have a look at this list of [65+ Best Free Datasets for Machine Learning](#) to find relevant data for training your models.

4.2.5 Overview

YOLO provided a super fast and accurate object detection algorithm that revolutionized computer vision research related to object detection.

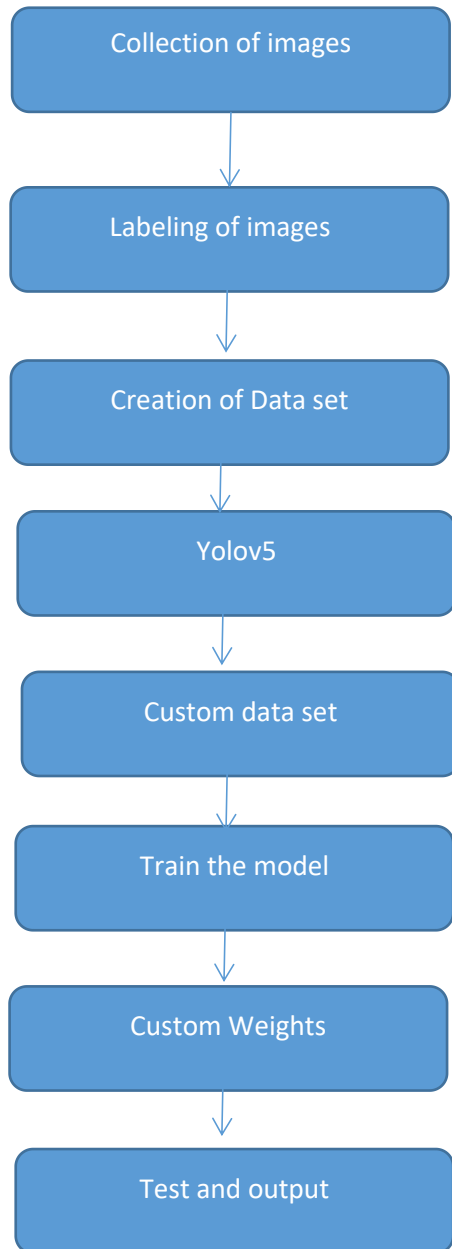
With over 5 versions (3 official) and cited more than 16 thousand times, YOLO has evolved tremendously ever since it was first proposed in 2015.

YOLO has large-scale applicability with thousands of use cases, particularly for autonomous driving, vehicle detection, and intelligent video analytic.

However, Like almost all tech, YOLO (and object detection in general), can have both positive and negative societal impact, which is why its usage should be regulated.

5. Design

5.1 Steps involved



6. Code Snippets

Some of the code snippets

```
▼ Setup

!git clone https://github.com/ultralytics/yolov5 # clone
%cd yolov5
!pip install -qr requirements.txt # install

from yolov5 import utils
display = utils.notebook_init() # checks

YOLOv5 🚀 v6.0-100-g5185981 torch 1.10.0+cu111 CUDA:0 (Tesla K80, 11441MiB)
Setup complete ✅
```

```
!unzip -q ../Yolov5.zip -d ../
```

```
!python detect.py --weights yolov5s.pt --img 640 --conf 0.25 --source data/images
display.Image(filename='runs/detect/exp/zidane.jpg', width=600)
```


7. Bibliography

Following are the websites that were used as references.

<https://thecleverprogrammer.com/2020/12/22/object-detection-with-python/>

<https://towardsdatascience.com/yolov5-end-to-end-object-detector-project-on-custom-dataset-5d9cc2c95921>

<https://towardsdatascience.com/real-time-object-detection-with-yolo-9dc039a2596b>