YOLO Object Detection

B-tech Semester- VI

Prepared at

ISO 9001:2008
ISO 27001:2013
CMMI LEVEL-5

Bhaskaracharya National Institute for Space Applications & Geo-informatics

Ministry of Electronics and Information Technology, Govt. of India.

Gandhinagar

Prepared By

Shashank Mathur	**Mohammed Adil**	**Meha Deora**	**Nishtha Gehlot**
ID No. E1	ID No. E1	ID No. E1	ID No. E1

Guide:

Vishal Patel

Project Scientist

BISAG- N, Gandhinagar.

SUBMITTED TO

Jodhpur Institute of Engineering and Technology (JIET) - Jodhpur

`

CERTIFICATE

This is to certify that the project report compiled by **Mr Shashank Mathur, Mr Mohammed Adil, Ms Meha Deora, Ms Nishtha Gehlot** students of 7th Semester **Jodhpur Institute of Engineering and Technology (JIET) - Jodhpur** have completed their final Semester internship project satisfactorily. To the best of our knowledge, this is an original and bonafide work done by them. They have worked on a Web-based application for "**Yolo Object Detection**", starting from May 25th, 2021 to June 28th, 2021.

During their tenure at this Institute, they were found to be sincere and meticulous in their work. We appreciate their enthusiasm & dedication to the work assigned to them.

We wish them every success.

Vishal Patel
Project, Scientist
BISAG-N, Gandhinagar

T. P. Singh
Project, Scientist
BISAG-N Gandhinagar

COLLEGE CERTIFICATE

This is to certify that the 6th Semester Internship Project entitled **"YOLO (V3) Object Detection"** has been carried out by **Shashank Mathur, Mohammed Adil, Meha Deora** and **Nishtha Gehlot** under my guidance in fulfilment of the degree of Bachelor of Technology in Computer Science and Electronics and Communication Engineering (6th Semester) of **Jodhpur Institute of Engineering and Technology (JIET) - Jodhpur** during the academic year 2021-2022.

Guide	**Head of the Department**
Mr Sanjay Bhandari	**Prof. Rajendra Purohit**
	Prof. Avnish Bora

Jodhpur Institute of Engineering and Technology (JIET) - Jodhpur

CANDIDATE'S DECLARATION

We declare that the 6th-semester internship project report entitled **"YOLO (V3) Object Detection"** is our work conducted under the supervision of the external guide **Vishal Patel** from BISAG-N (Bhaskaracharya National Institute for Space Applications & Geo-informatics). We further declare that to the best of our knowledge the report for this project does not contain any part of the work which has been submitted previously for such project either in this or any other institutions without proper citation.

Shashank Mathur	**Mohammed Adil**	**Meha Deora**	**Nishtha Gehlot**
ID No. E1	ID No. E1	ID No. E1	ID No. E1

Submitted To:

Jodhpur Institute of Engineering and Technology (JIET) - Jodhpur

ACKNOWLEDGMENT

We are grateful to **T.P. Singh,** Director (BISAG-N) for giving us this opportunity to work under the guidance of renowned people in the field of MIS Based Portal also providing us with the required resources in the company.

We would like to express our endless thanks to our external guide **Mr Vishal Patel**, And Admin Staff **Mr Saurabh Bhabhor** and **Mr Sidhdharth Patel** at Bhaskaracharya National Institute of Space Application and Geo-informatics for their sincere and dedicated guidance throughout the project development.

Also, our hearty gratitude to our Head of Department, **Prof. Rajendra Purohit, Prof. Avnish Bora** and our internal guide **Mr Sanjay Bhandari** for giving us encouragement and technical support on the project.

Shashank Mathur	**Mohammed Adil**	**Meha Deora**	**Nishtha Gehlot**
ID No. E1	ID No. E1	ID No. E1	ID No. E1

Table Of Content

1. Introduction

1.1 Project Details:

Object detection is a computer vision technique in which a software system can detect, locate, and trace the object from a given image or video. The special attribute of object detection is that it identifies the class of objects (person, table, chair, etc.) and their location-specific coordinates in the given image. The location is pointed out by drawing a bounding box around the object. The bounding box may or may not accurately locate the position of the object. The ability to locate the object inside an image defines the performance of the algorithm used for detection.

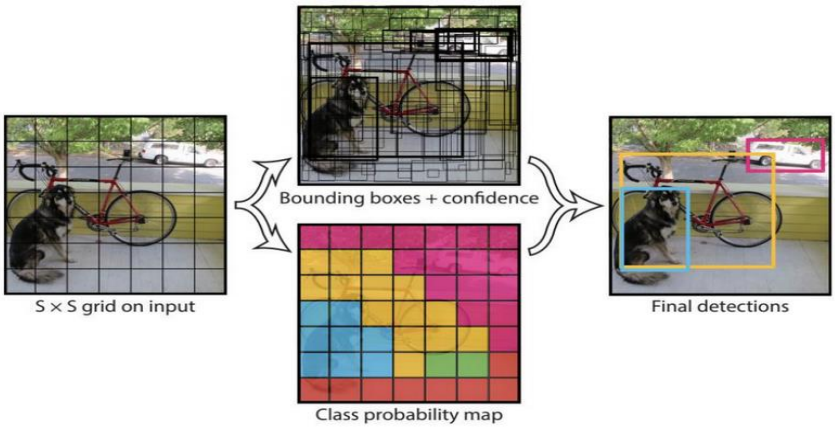
In many computer vision systems, object detection is the first task being performed as it allows to obtain further information regarding the detected object and about the scene. Once an object instance has been detected, it is possible to obtain further information, including recognizing the specific instance, tracking the object over an image sequence, and extracting further information about the object, while it is also possible to infer the presence or location of other objects in the scene and to better estimate further information about the scene, among other contextual information.

These object detection algorithms might be pre-trained or can be trained from scratch. In most use cases, we use pre-trained weights from pre-trained models and then fine-tune them as per our requirements and different use cases.

YOLO which stands for “You only look once” is a single shot detection algorithm. It gives a perfect description of this algorithm as it predicts classes and bounding boxes for the whole image in one run of the algorithm.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.



YOLO works by taking an image and splitting it into an SxS grid, within each of the grid we take m bounding boxes. For each of the bounding boxes, the network gives an output a class probability and offset values for the bounding box. The bounding boxes have the class probability above which a threshold value is selected and used to locate the object within the image. YOLO is orders of magnitude faster(45 frames per second) than any other object detection algorithms. The limitation of the YOLO algorithm is that it struggles with the small objects within the image, for example, it might have difficulties in identifying a flock of birds. This is due to the spatial constraints of the algorithm.

Object detection aims to detect all instances of objects from a known class, such as people, cars or faces in an image. Generally, only a small number of instances of the object are present in the image, but there is a very large number of possible locations and scales at which they can occur and that need to somehow be explored. Each detection of the image is reported with some form of pose information. This is as simple as the location of the object, a location and scale, or the extent of the object defined in terms of a bounding box. In some other situations, the pose information is more detailed and contains the parameters of a linear or non-linear transformation. For example, face detection in a face detector may compute the locations of the eyes, nose and mouth, in addition to the bounding box of the face. An example of bicycle detection in an image that specifies the locations of certain parts is shown in Figure 1. The pose can also be defined by a three-dimensional transformation specifying the location of the object relative to the camera. Object detection systems always construct a model for an object class from a set of training examples. In the case of a fixed rigid object in an image, only one example

may be needed, but more generally multiple training examples are necessary to capture certain aspects of class variability.

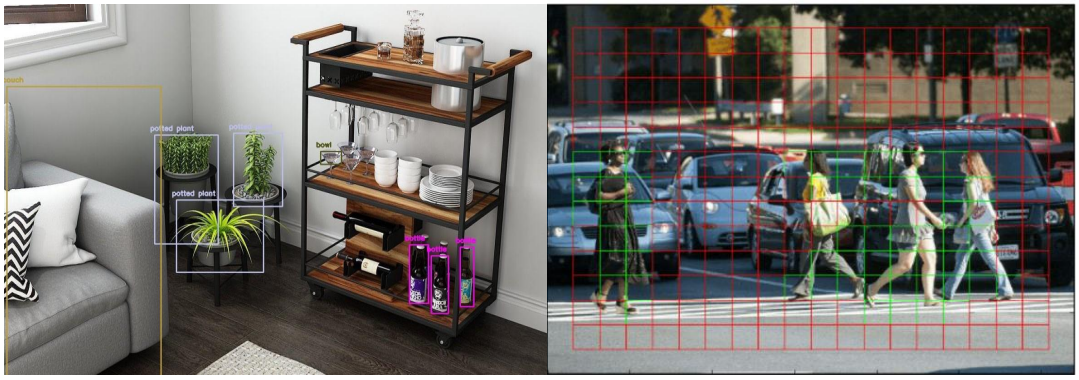


Figure 1

Convolutional implementation of the sliding windows Before we discuss the implementation of the sliding window using convents, let us analyze how we can convert the fully connected layers of the network into convolutional layers. Fig. 2 shows a simple convolutional network with two fully connected layers each of shape.

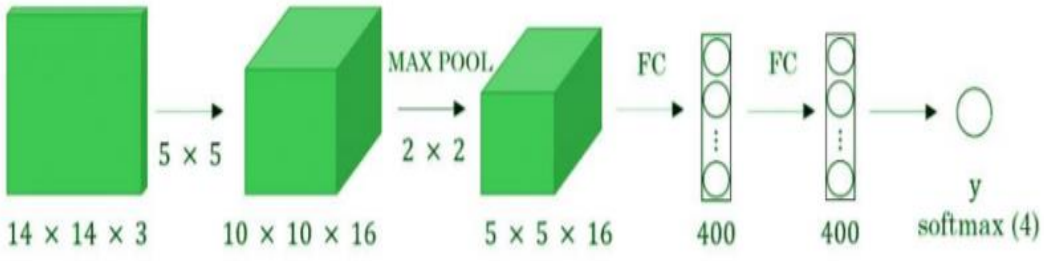


Figure 2: simple convolution network

A fully connected layer can be converted to a convolutional layer with the help of a 1D convolutional layer. The width and height of this layer are equal to one and the number of filters is equal to the shape of the fully connected layer. An example of this is shown in Fig 3.

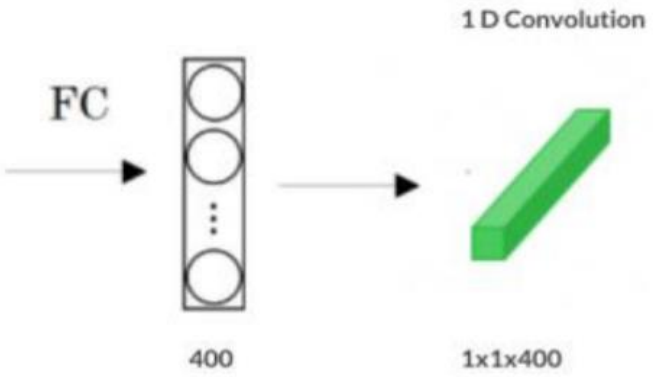


Figure 3

We can apply the concept of conversion of a fully connected layer into a convolutional layer to the model by replacing the fully connected layer with a 1-D convolutional layer. The number of filters of the 1D convolutional layer is equal to the shape of the fully connected layer. This representation is shown in Fig 4. Also, the output softmax layer is a convolutional layer of shape (1, 1, 4), where 4 is the number of classes to predict.

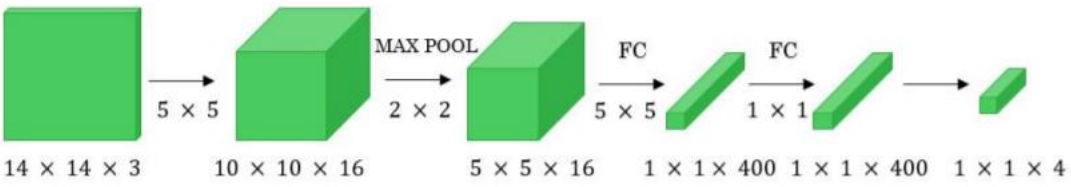


Figure 4

Now, let's extend the above approach to implement a convolutional version of the sliding window. First, let us consider the ConvNet that we have trained to be in the following representation (no fully connected layers).

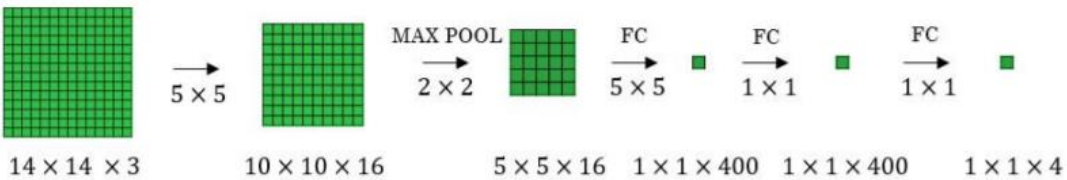


Figure 5

Let's assume the size of the input image to be 16 × 16 × 3. If we are using the sliding window approach, then we would have passed this image to the above ConvNet four times, where each

time the sliding window crops the part of the input image matrix of size 14 × 14 × 3 and pass it through the ConvNet. But instead of this, we feed the full image (with shape 16 × 16 × 3) directly into the trained ConvNet (see Fig. 6). These results will give an output matrix of shape 2 × 2 × 4. Each cell in the output matrix represents the result of the possible crop and the classified value of the cropped image. For example, the left cell of the output matrix(the green one) in Fig. 6 represents the result of the first sliding window. The other cells in the matrix represent the results of the remaining sliding window operations.

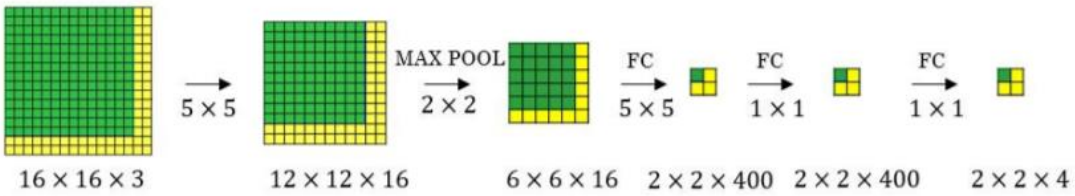


Figure 6

The stride of the sliding window is decided by the number of filters used in the Max Pool layer. In the example above, the Max Pool layer has two filters, and for the result, the sliding window moves with a stride of two resulting in four possible outputs to the given input. The main advantage of using this technique is that the sliding window runs and computes all values simultaneously. Consequently, this technique is really fast. The weakness of this technique is that the position of the bounding boxes is not very accurate. A better algorithm that tackles the issue of predicting accurate bounding boxes while using the convolutional sliding window technique is the YOLO algorithm. YOLO stands for you only look 6 once which was developed in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. It is popular because it achieves high accuracy while running in real-time. This algorithm requires only one forward propagation pass through the network to make the predictions. This algorithm divides the image into grids and then runs the image classification and localization algorithm (discussed under object localization) on each of the grid cells. For example, we can give an input image of size 256 × 256. We place a 3 × 3 grid on the image (see Fig. 7).

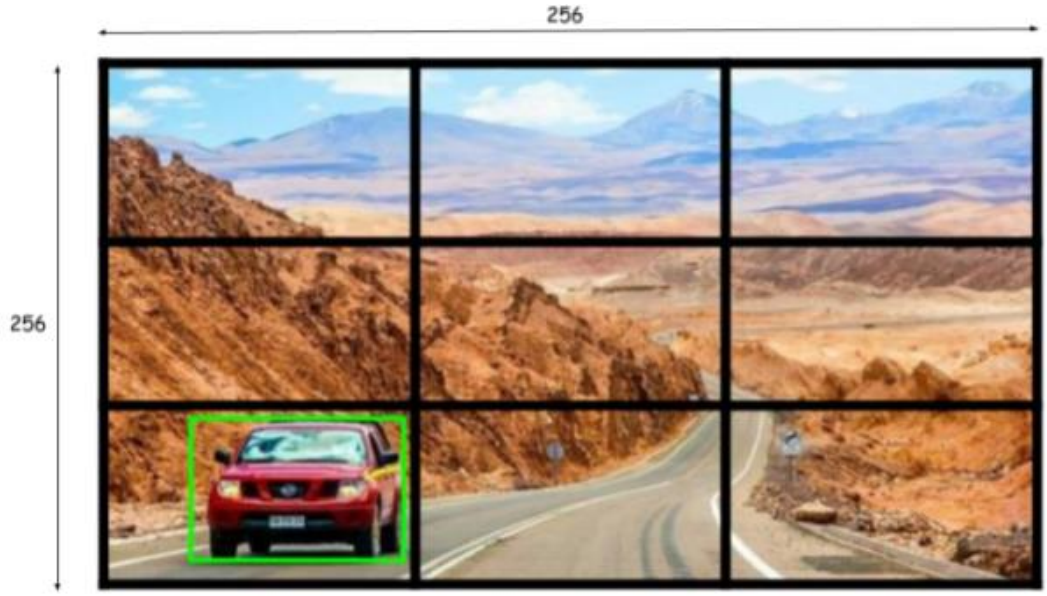


Figure 7

Next, we shall apply the image classification and localization algorithm to each grid cell. In the image of each grid cell, the target variable is defined as Yi,j=[pcbxbybhbwc1c2c3c4]T(6) Do everything once with the convolution sliding window. Since the shape of the target variable for each grid cell in the image is 1 × 9 and there are 9 (3 × 3) grid cells, the final output of the model will be:

$$Final\ Output = \underbrace{3 \times 3}_{\text{Number of grid cells}} \times \underbrace{9}_{\text{Output label for each grid cell}}$$

The advantages of the YOLO algorithm is that it is very fast and predicts much more accurate bounding boxes. Also, in practice to get the more accurate predictions, we use a much finer grid, say 19 × 19, in which case the target output is of the shape 19 × 19 × 9.

Phase I:

This is especially true for deep learning domains like computer vision. Not everyone has the computational resources to build a DL model from scratch. That's where predefined frameworks and pretrained models come in handy. And in this article, we will look at one such framework for object detection

Developing a system which generates the object report present in any image provided and stores it into a text file.

Folders provided into this can also generate the object report of multiple images provided.

Phase II :

Training the YOLO, to detect the persons i.e. Shah rukh khan, Narendra Modi, Crestiano Ronaldo and Barack Obama.

Collect the Data/images of these peoples 50 items each which makes 250 images in data , then label the data in YOLO format to provide into the algorithm, this process need an long runtime and good hardware, so to achieve good training accuracy and better resources utilized the Google Collebertry to do this process of training.

Used the concept of transfer learning technique to work efficiently. Transfer learning is an optimization that allows rapid progress or improved performance when modelling the second task. Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned.

Phase III :

Solving the complexity of the training and custom object detection model, by automating the task of training and environment setup, by developing an web-application which runs on an

GPU accelerated server, helps you to create an object detection module train, on the provided data.
It runs on the base of Darknet, Uses the concept of transfer learning technique to work efficiently. Transfer learning is an optimization that allows rapid progress or improved performance when modelling the second task. Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned.

1.2 Purpose

Object detection is a computer vision technique that allows us to identify and locate objects in an image or video. With this kind of identification and localization, object detection can be used to count objects in a scene and determine and track their precise locations, all while accurately labelling them. Object detection allows us to at once classify the types of things found while also locating instances of them within the image.

Object detection is inextricably linked to other similar computer vision techniques like image recognition and image segmentation, in that it helps us understand and analyze scenes in images or video.

But there are important differences. Image recognition only outputs a class label for an identified object, and image segmentation creates a pixel-level understanding of a scene's elements. What separates object detection from these other tasks is its unique ability to locate objects within an image or video. This then allows us to count and then track those objects.

1.3 Scope

The goal of object detection is to recognize instances of a predefined set of object classes (e.g. people, cars, bikes, animals) and describe the locations of each detected object in the image using a bounding box.

1.4 Objective

The main purpose of object detection is to identify and locate one or more effective targets from still image or video data. It comprehensively includes a variety of important techniques, such as image processing, pattern recognition, artificial intelligence and machine learning.

1.5 Technology

1) Download and install Python version 3 from the official Python Language website

https://python.org

2) Install the following dependencies via pip:

Tensorflow:
Tensorflow is an open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networksKeras. It is used for both research and production by Google. Tensorflow is developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015. Tensorflow is Google Brain's second-generation system.1st Version of TensorFlow was released on February 11, 2017. While the reference implementation runs on single devices, Tensorflow can run on multiple CPUs and GPU (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on various platforms such as 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS. The architecture of TensorFlow allows the easy deployment of computation across a variety of platforms (CPU's, GPU's, TPU's), and from desktops - clusters of servers to mobile and edge devices. Tensorflow computations are expressed as stateful dataflow graphs. The name Tensorflow derives from operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors.

pip install tensorflow -command

Numpy:

NumPy is a library of Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate over these arrays. The ancestor of NumPy, Numeric, was created by Jim Hugunin with contributions from several developers. In 2005 Travis Oliphant created NumPy by incorporating features of computing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

pip install numpy -command

OpenCV:

OpenCV is a library of programming functions mainly aimed at real-time computer vision. originally developed by Intel, it is later supported by Willow Garage then Itseez. The library is cross-platform and free to use under the open-source BSD license.

pip install opencv-python -command

Matplotlib:

Matplotlib is a Python programming language plotting library and its NumPy numerical math extension. It provides an object-oriented API to use general-purpose GUI toolkits such as Tkinter, wxPython, Qt, or GTK+ to embed plots into applications.

pip install matplotlib - command

Keras:

Keras is an open-source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

pip install keras - command

1.6 Literature Survey

In various fields, there is a necessity to detect the target object and also track them effectively while handling occlusions and other included complexities. Many researchers (Almeida and Guting 2004, Hsiao-Ping Tsai 2011, Nicolas Papadakis and Aure lie Bureau 2010) attempted various approaches in object tracking. The nature of the techniques largely depends on the application domain. Some of the research works which made the evolution to proposed work in the field of object tracking are depicted as follows.

2. Project Management

2.1 Feasibility Study

The feasibility study is a major factor that contributes to the analysis and development of the system. The decision of the system analyst whether to design a system or not depends on its feasibility study. A feasibility study is undertaken whenever a possibility of the probability of improving an existing system or designing a new system. A feasibility study helps to meet user requirements.

2.1.1 Technical Feasibility

As far as machine learning is concerned a high processing system is required for maximum accuracy. Our goal is to achieve maximum throughput in less power consumption. Technically all three phases are feasible as we are having the capabilities to convert the idea into a working system. All the technical resources will be validated. The technology used here is the Yolo (v3), jupyter notebook, streamlit, labelling software, python programming language.

2.1.2 Time Schedule Feasibility

As it is the estimated time, this project will take around 20 - 25 days to be completed, including all three phases. i.e.

Phase I: Object Detection using a Pre-trained model.

Phase II: Train YOLO for custom objects.

Phase III: Web application for automated training.

2.1.4 Implementation Feasibility

The Implementation Feasibility will meet both ends as we have all the technical resources available to us, and we are capable of converting our ideas into practical ones.

Phase I: Need of the pre-trained YOLO weights to demonstrate the object detection using the YOLO which is available on the official Darknet repository of YOLO, with examples and proper documentation.

Phase II: For implementation, there is a need for a high computing GPU and the official darknet repository which will be fulfilled by Google collab for the High-speed computation Graphics processing unit, and by darknet repository by AlexeyAB/darknet respectively.
For Data preparation need to access google open images and labelling software that labels the images/objects in the desired format.

Phase III: Implementation of an automated object detection training web application needs a High-end server to deploy it as a real-time application which can be feasible nowadays using different cloud-based services, which gives us high computation power, like Amazon Sagemaker, IBM, Azure etc.

2.1.3 Operational Feasibility

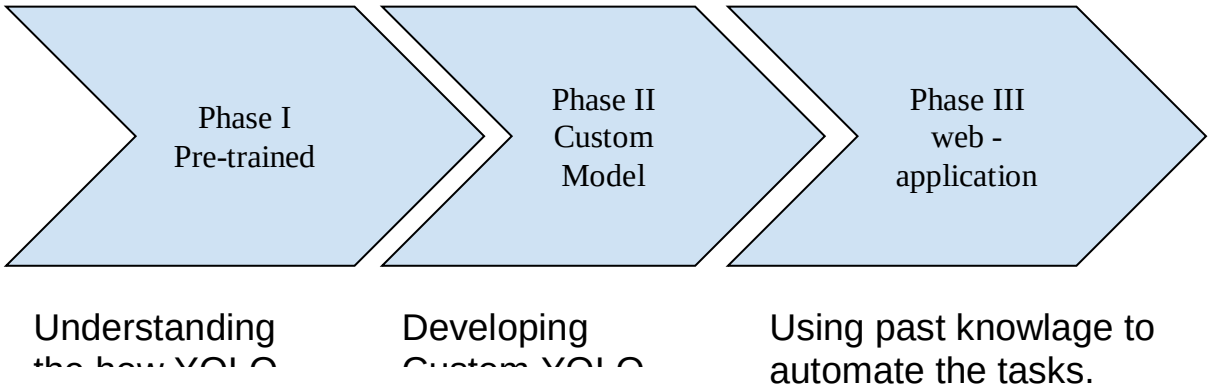
The proposed new system will solve the problem very effectively because at the end of the third phase we will develop our web application for detection that will take the data or file of objects in a zip format as input and after three to four hours it will complete the detection of the data and will return it to the user.

2.2 Project Planning

2.2.1 Project Development Approach and Justification

The approach involves a single deep convolutional neural network (originally a version of GoogLeNet, later updated and called DarkNet based on VGG) that splits the input into a grid of cells and each cell directly predicts a bounding box and object classification. A result is a large number of candidate bounding boxes that are consolidated into a final prediction by a post-processing step.

There are three main variations of the approach, at the time of writing; they are YOLOv1, YOLOv2, and YOLOv3. The first version proposed the general architecture, whereas the second version refined the design and made use of predefined anchor boxes to improve the bounding box proposal, and version three further refined the model architecture and training process. However, in this project, we have used YOLOv3.



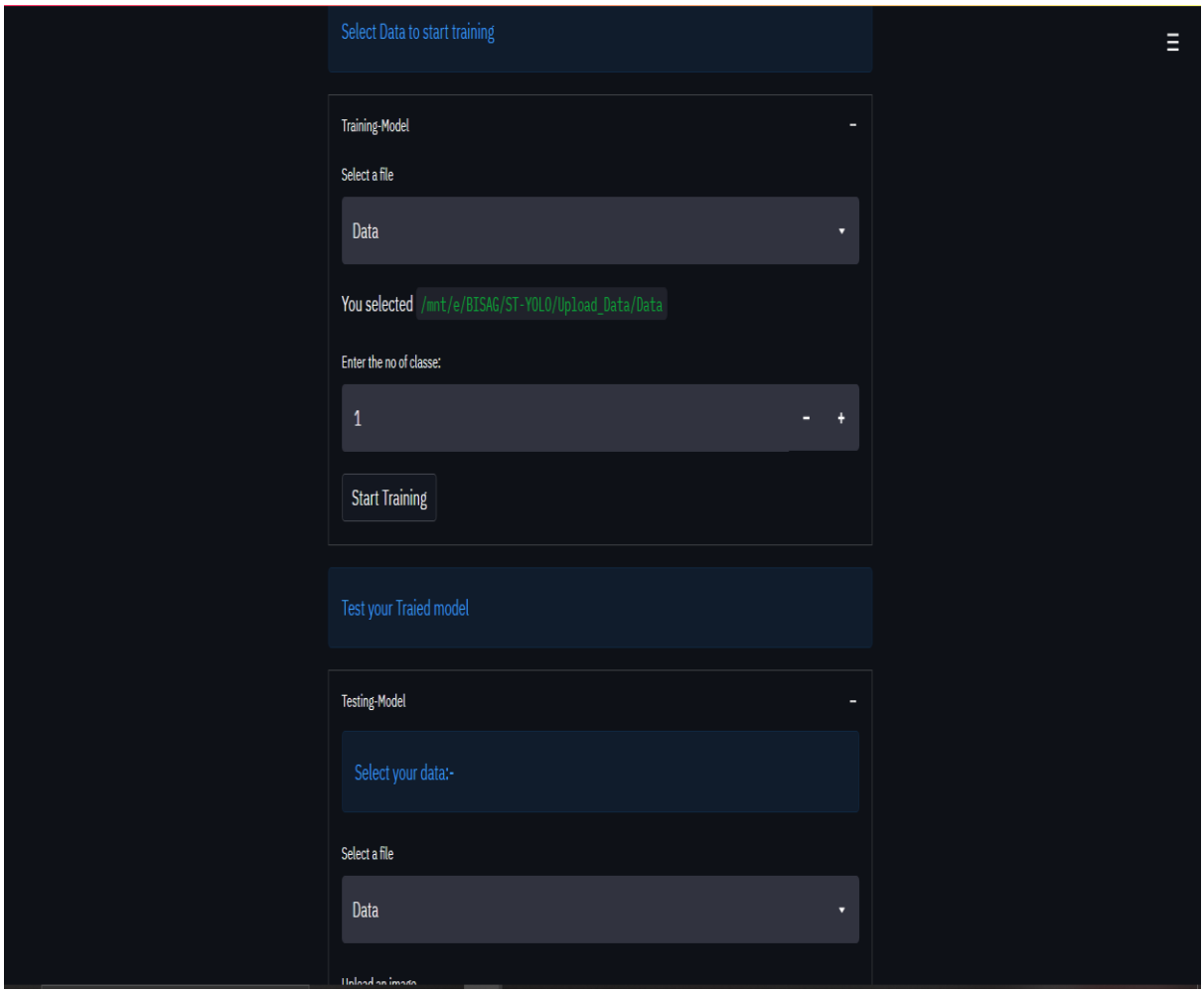
Although the accuracy of the models is close but not as good as Region-Based Convolutional Neural Networks (R-CNNs), they are popular for object detection because of their detection speed, often demonstrated in real-time on video or with a camera feed input.

2.2.2 Deliverables

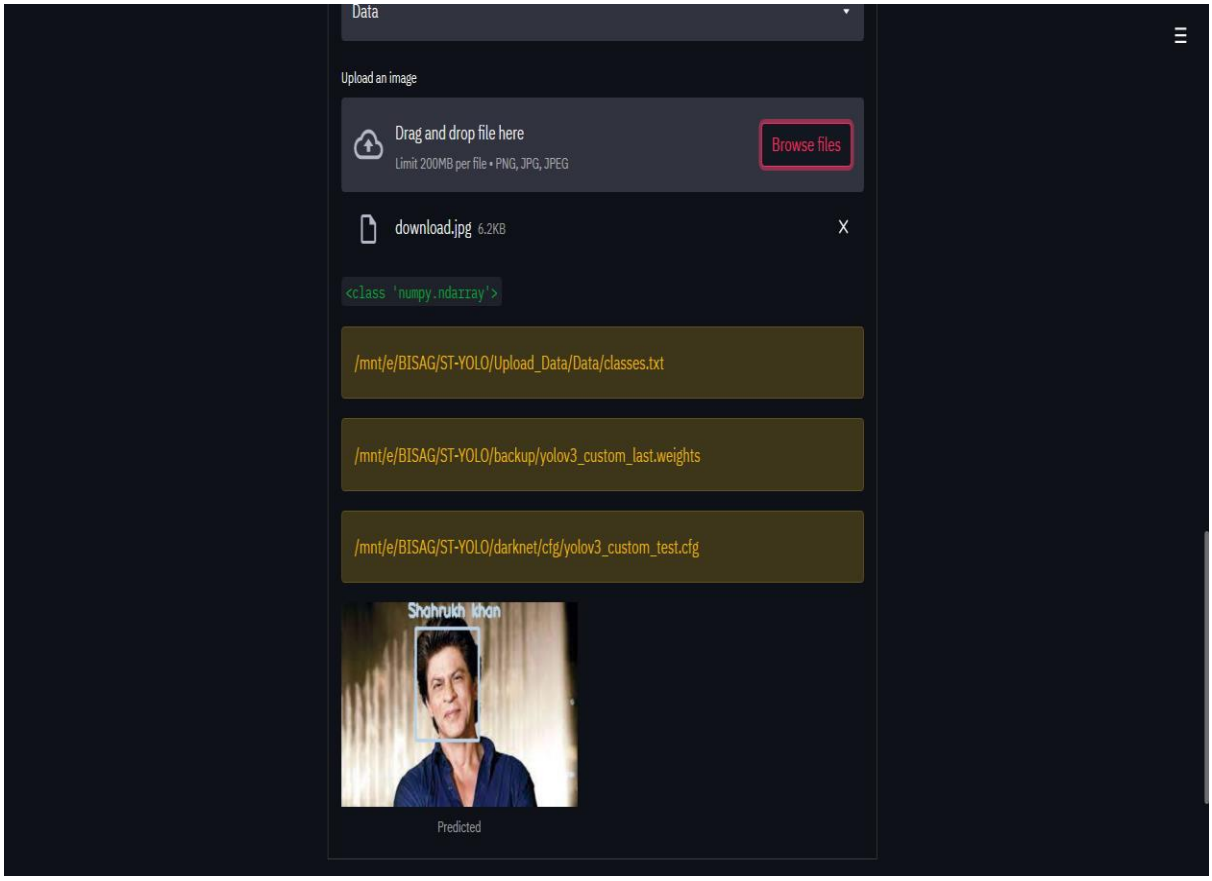
A web application that is capable of automated high computation, power to train an object detection model development and testing environment, works and is based on the YOLO algorithm.

It will reduce the efforts for programming the models again and again for different data, here the application provides the convenience to train an object detection model on any data which is well prepared in the desired format.

Training:-



Testing:-



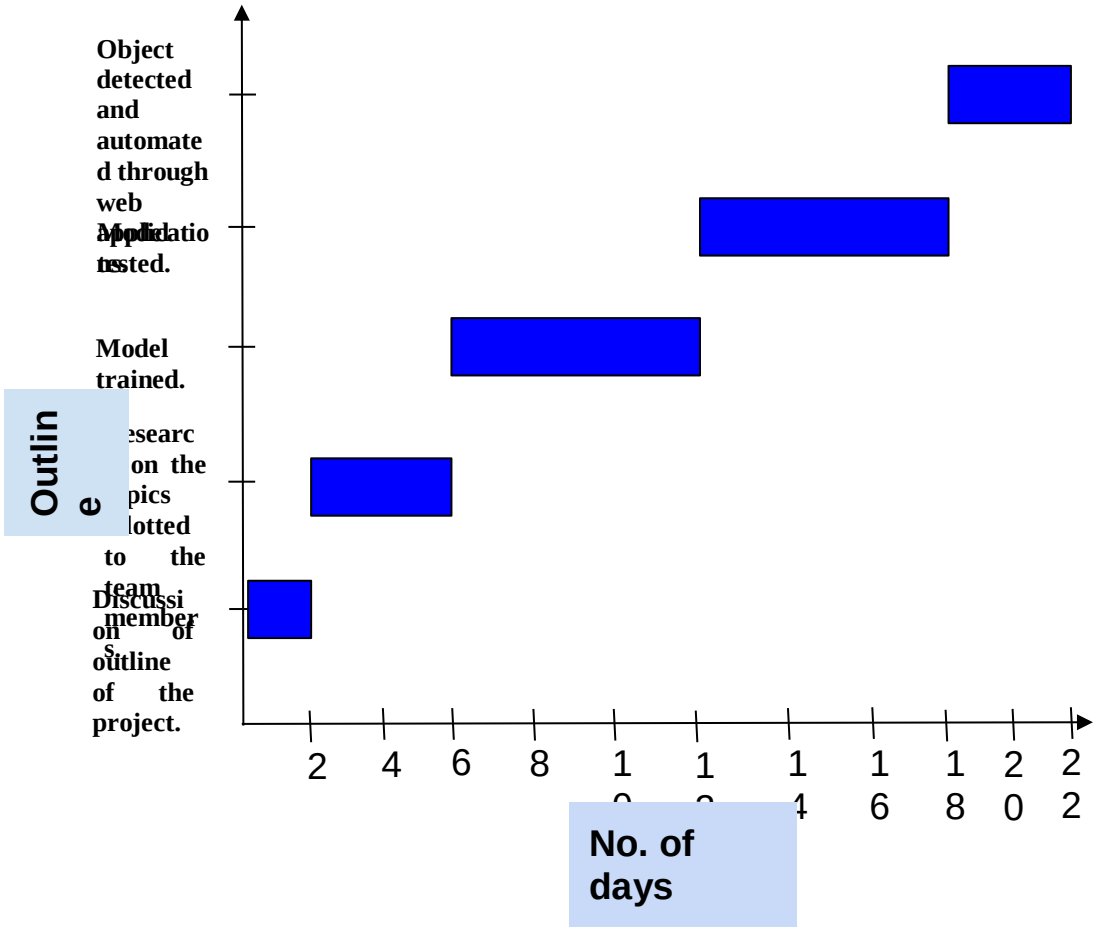
2.2.3 Roles & Responsibilities

S.No.	Name	Responsibility
1.	Shashank Mathur	Phase I, Phase II & Phase III.
2.	Meha Deora	Trained the data on YOLOv3 (Phase II), Report writing.
3.	Mohammed Adil	Trained the data on YOLOv3 (Phase II), Report writing, collecting the data.
4.	Nishtha Gehlot	Trained the data on YOLOv3 (Phase II), Report writing.

2.2.4 Group Dependencies

S.No.	Name	Dependencies
1.	Shashank Mathur	Planning and Implementation.
2.	Meha Deora	Data preparation and Documentation.
3.	Mohammed Adil	Data preparation and Documentation.
4.	Nishtha Gehlot	Data preparation and Documentation.

2.3 Project Scheduling



3. System Requirement Study

3.1 Study of Current System

Linux is one of the most popular versions of the UNIX operating System. It is open source as its source code is freely available. It is free to use. Linux was designed considering UNIX compatibility. Its functionality list is quite similar to that of UNIX.

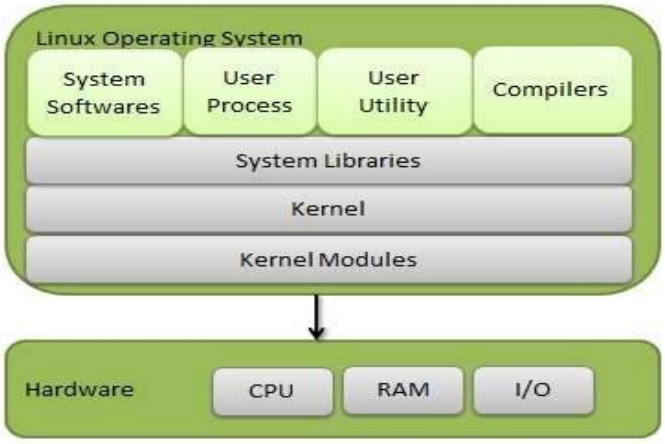
Components of Linux System

Linux Operating System has primarily three components

- Kernel − Kernel is the core part of Linux. It is responsible for all major activities of this operating system. It consists of various modules and interacts directly with the underlying hardware. The kernel provides the required abstraction to hide low-level hardware details in system or application programs.

- System Library − System libraries are special functions or programs using application programs or system utilities to access Kernel's features. These libraries implement most of the operating system’s functionalities and do not require kernel module's code access rights.

- System Utility − System Utility programs are responsible to do specialized, individual-level tasks.



Basic Features

Following are some of the important features of the Linux Operating System.

- Portable − Portability means software can work on different types of hardware in the same way. Linux kernel and application programs support their installation on any kind of hardware platform.

- Open Source − Linux source code is freely available and it is a community-based development project. Multiple teams work in collaboration to enhance the capability of the Linux operating system and it is continuously evolving.

- Multi-User − Linux is a multiuser system that means multiple users can access system resources like memory/ ram/ application programs at the same time.

- Multiprogramming − Linux is a multiprogramming system that means multiple applications can run at the same time.

- Hierarchical File System − Linux provides a standard file structure in which system files/ user files are arranged.

- Shell − Linux provides a special interpreter program that can be used to execute commands of the operating system. It can be used to do various types of operations, called application programs. etc.

- Security − Linux provides user security using authentication features like password protection/ controlled access to specific files/ encryption of data.

Properties of Linux:-

1. Open Source

As mentioned earlier Linux is an open-source operating system. This can come in handy especially for people who code and/or need to make changes to the way their computer works. It is also free to install and use unlike Windows or Mac so it could be a great alternative for someone who is on a budget but needs a computer that can adapt according to his/her needs.

2. Security

One of the best qualities of Linux is that it is extremely secure. Since it is an open-source OS, several developers in the world look out for viruses and potential threats. So, when the code needs to be upgraded or changed, it can be done in no time. Another reason for this security is that since the number of Linux users is comparatively lower than Windows users, it is not profitable to create viruses for this system. Automatically, lesser viruses equal lesser chances of your system being infected.

3. Speed

It is a fact that antivirus programs slow down a system. Since the software is constantly updating its database of viruses and checking the system for those viruses, it uses a lot of memory. Since Linux does not require the installation of this software, a huge chunk of memory is unused, creating a faster system. On top of that, Linux already works faster than Windows because of its comparatively simpler user interface.

4. Redesigning

One of the biggest advantages of Linux is that it is very easy to change according to your needs. It is also one reason why it is very popular amongst programmers. Since you can see the source code for this OS, it is easier and legal (unlike Windows), to change it according to your needs for your system.

5. Low System Specifications

Another one of the reasons why Linux is a better choice is the fact that it works on computers with very low system specifications. If you have an older computer that you use and cannot throw away, Linux is a great choice for you. It will run without a problem, unlike Windows or Mac.

3.2 Problems and weakness of the current system

Linux Cons

1. Learning Curve

Learning how to use Linux can take a while, especially if you are used to using other Operating Systems and not an open-source operating system like Linux. The user experience is a lot different in Linux so you will need to set time aside to learn the system first. This could be frustrating to some people.

2. Installing Software

A variety of applications that work on Windows or Apple OS do not work on Linux. If you are someone who needs to have access to certain applications for your work, you will probably want to see if it works in Linux. Microsoft applications are one such example of software that isn't available for Linux users.

3. Lack of Games

Statistically, the majority of computer users use either Windows or Apple OS so game developers create games suited to these operating systems. A lot of popular games aren't supported by Linux. This is one reason why Linux isn't popular among young users yet.

4. Hardware Drivers

One of the biggest problems in Linux, as reported by users, is that drivers don't exist in this operating system. Like most software, there are ways to work around that but there have been problems reported when people tried to use old hardware or certain other hardware.

3.3 Hardware and Software Requirements

Phase 1:

Hardware Specification:

- Processor: I5 7TH Generation
- RAM: 8 GB
- Hard disk: 1 TB
- GPU: Nvidia 940MX 2GB graphics

Software Specification:

- Operating System: Linux.
- Technologies used: Machine learning, neural networks, TensorFlow, Deep Learning, Darknet, Yolov3, Darknet-Yolo.

Phase 2:

Hardware Specification:

- Processor: Intel(R) Xeon(R) CPU @ 2.30GHz.
- RAM: 13 GB.
- Hard disk: 100 GB
- GPU: Nvidia Tesla K80 GPU

Software Specification:

- Operating System: Linux.
- Technologies used: Machine learning, neural networks, TensorFlow, Deep Learning, Darknet, Yolov3, Darknet-Yolo.

Phase 3:

Hardware Specification:

- Processor: I5 7TH Generation
- RAM: 8 GB
- Hard disk: 1 TB

GPU: Nvidia 940MX 2GB graphics

Software Specification:

- Operating System: Linux.
- Technologies used: Machine learning, neural networks, TensorFlow, Deep Learning, Darknet, Yolov3, Darknet-Yolo, web-application frameworks.

3.4 Constraints

3.4.1 User Interface

The Linux based kernel can run a wide variety of software across many different hardware-based platforms. A computer can act as a server, which means it primarily handles data on other's behalf or can act like a desktop, which means a user will be interacting with it directly.

The system can run software or it can be used as a development PC in the process of creating any software. Linux can perform multiple roles as there is no special allocation to Linux about the role of the system; it's only a matter of configuring the present applications and how they execute.

Command Line Interface (CLI):

The Command Line Interface (CLI), is a non-graphical, text-based interface to the computer system, where the user types in a command and the computer then successfully executes it. The Terminal is the platform or the IDE that provides the user's command-line interface (CLI) environment. The CLI terminal accepts the commands that the user types and passes to a shell. The shell then receives and interprets what the user has typed into the instructions that can be executed by the OS (Operating System). If the output is produced by the specific command, then this text is displayed in the terminal. If any of the problems with the commands are found, then some error message is displayed.

Graphical and the non-Graphic Interface:

Linux has two approaches: graphically and non-graphically. In graphical mode, The actual applications live in windows that we can resize and move around according to our needs. We have the menu and tools to help us find what we're looking for. This is the point where we'll use a required web browser, our graphics editing tools, and our emails. Here we can see some examples of the graphical desktop, with a menu bar of popular applications to the left.

In Graphical Mode (GUI), we can have many shells open, which is a good thing when we are performing some tasks on multiple/remote computers. We can even login with our username/id and password/keys through the GUI. After successfully logging in, we are taken to the OS desktop where we can use the installed applications.

The non-graphical mode starts with a text-based login, As shown below. We are generally prompted for our username/ID and after entering that, we are then prompted for our password. If the login is successful, then we are taken straight to an execution shell. In the command-line interface or the CLI, there are none of the windows present to move around. Even though we have specific text editors, dedicated web browsers, and email clients, they are just texts. This is how UNIX got its start before the graphical environments became the norm. Most servers will be running in command line mode (CLI) too because a GUI is a waste of resources and dataspace.

3.4.2 Hardware Interface

Linux currently supports systems with an Intel 80386, 80486, Pentium, Pentium Pro, Pentium II, and Pentium III CPU. This includes all variations on this CPU type, such as the 386SX, 486SX, 486DX, and 486DX2. Non-Intel "clones," such as AMD and Cyrix processors, work with Linux as well.

But while working with neural network need high computational power.

For any neural network, the training phase of the deep learning model is the most resource-intensive task

While training, a neural network takes in inputs, which are then processed in hidden layers using weights that are adjusted during training and the model then spits out a prediction. Weights are adjusted to find patterns to make better predictions.

A GPU (Graphics Processing Unit) is a specialized processor with dedicated memory that conventionally perform floating-point operations required for rendering graphics
CPUs are mostly applicable for problems that require parsing through or interpreting complex logic in code, GPUs are designed to the dedicated graphical rendering workhorses of computer games, and which were later enhanced to accelerate other geometric calculations (for instance, transforming polygons or rotating verticals into different coordinate systems like 3D).

Why choose GPUs for Deep Learning:

GPUs are optimized for training artificial intelligence and deep learning models as they can process multiple computations simultaneously.
They have a large number of cores, which allows for better computation of multiple parallel processes. Additionally, computations in deep learning need to handle huge amounts of data — this makes a GPU's memory bandwidth most suitable.
There are a few deciding parameters to determine whether to use a CPU or a GPU to train a deep learning model:

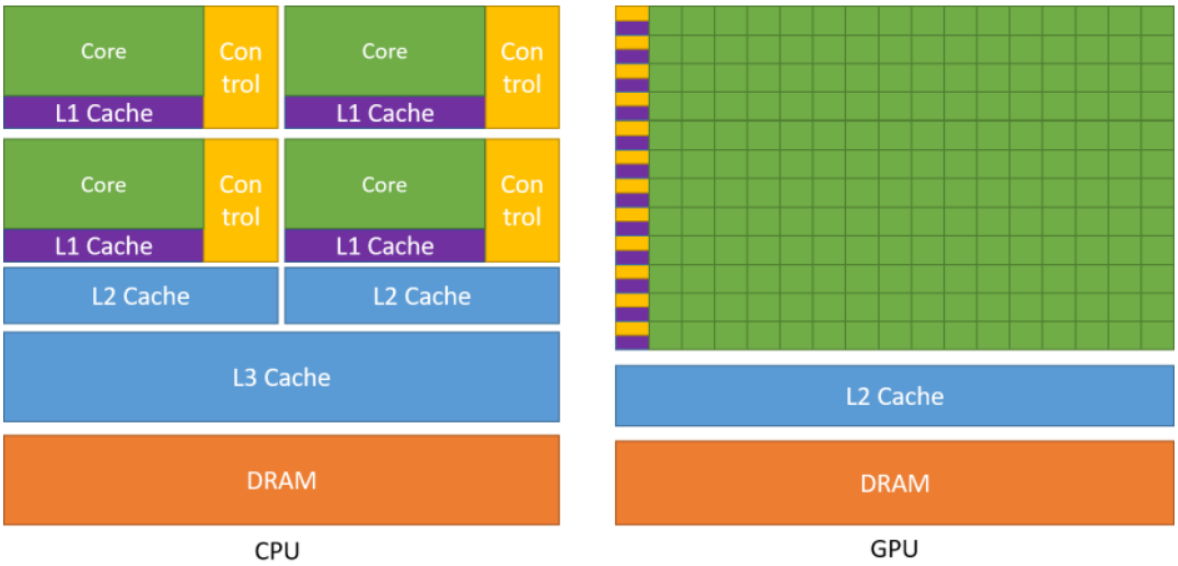
Memory Bandwidth:

Bandwidth is one of the main reasons why GPUs are faster for computing than CPUs. With large datasets, the CPU takes up a lot of memory while training the model.

Computing huge and complex jobs take up a lot of clock cycles in the CPU — CPUs take up jobs sequentially and has a fewer number of cores than their counterpart, GPU.

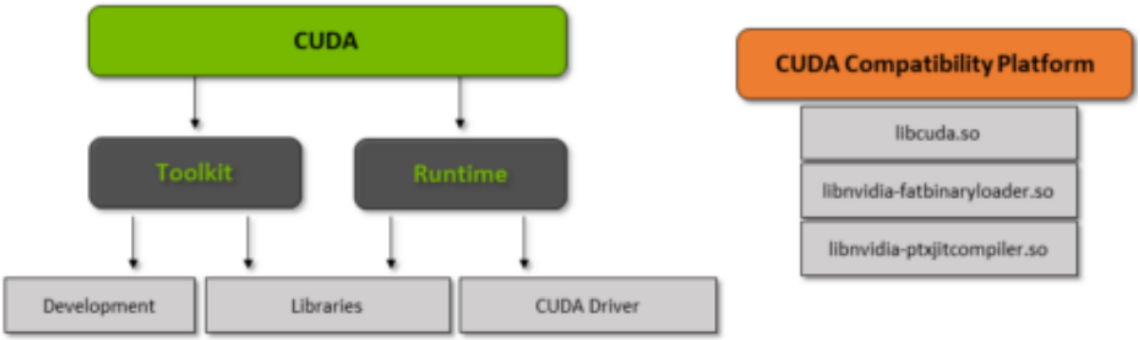
A standalone GPU, on the other hand, comes with dedicated VRAM (Video RAM) memory.

CPUs aren't good enough. CPUs are good at handling different tasks related to different operations like handling operating systems, handling spreadsheets, playing HD videos, extracting large zip files, all at the same time. These are some things that a GPU simply cannot do.



CUDA:

CUDA stands for 'Compute Unified Device Architecture' which was launched in the year 2007, it's a way in which you can achieve parallel computing and yield the most out of your GPU power in an optimized way, which results in much better performance while executing tasks.



The CUDA toolkit is a complete package that consists of a development environment that is used to build applications that make use of GPUs. This toolkit mainly contains a c/c++ compiler, debugger, and libraries. Also, the CUDA runtime has its drivers so that it can communicate with the GPU. CUDA is also a programming language that is specifically made for instructing the GPU for performing a task. It is also known as GPU programming.

3.5 Assumption and Dependencies

3.5.1 Assumptions

- Users must have basic knowledge of Machine learning and Deep learning.
- Familiar with Neural Networks.
- Prior knowledge of the YOLO algorithm.
- A system having the GPU along with CudaNN.

3.5.2 Dependencies

Graphics Card :

Training a model in deep learning requires a large dataset, hence the large computational operations in terms of memory. To compute the data efficiently, a GPU is an optimum choice. The larger the computations, the more the advantage of a GPU over a CPU.

A GPU is a parallel programming setup involving GPUs & CPUs which can process & analyze data in a similar way to an image or another graphic form. GPUs were created for better and more general graphic processing but were later found to fit scientific computing well. This is because most of the graphic processing involves applying operations on large matrices.

Cuda NN:

The CUDA toolkit is a complete package that consists of a development environment that is used to build applications that make use of GPUs. This toolkit mainly contains a c/c++ compiler, debugger, and libraries. Also, the CUDA runtime has its drivers so that it can communicate with the GPU. CUDA is also a programming language that is specifically made for instructing the GPU for performing a task. It is also known as GPU programming.

4. Requirement of Proposed System

4.1 Main Module of the System

Phase I: Object Detection.

Phase II: Deep Neural Network.

Phase III: Automation.

4.2 Module Descriptions

Phase I:
Official YOLO pre-trained weights and configuration acts as the main module of this phase, these weights and config files are used along with the Open-Cv to predict the objects on images or video.
The main module of this phase work on images to identify the object present in the image and return a dictionary of object details.

Phase II:
Deep neural network-based YOLO algorithm, acts as the main module for the whole and sole of the project, supported along with the official YOLO repository Darknet, which helps in setting up the configuration environment to work with the Training Object detection model by providing it with the data.

Phase III:
The main module automated application is always at its heart, that is the automation of training, the main module acts/works to train the object detection model on the provided data, along with preparing the automated configuration and environment according to the provided data.

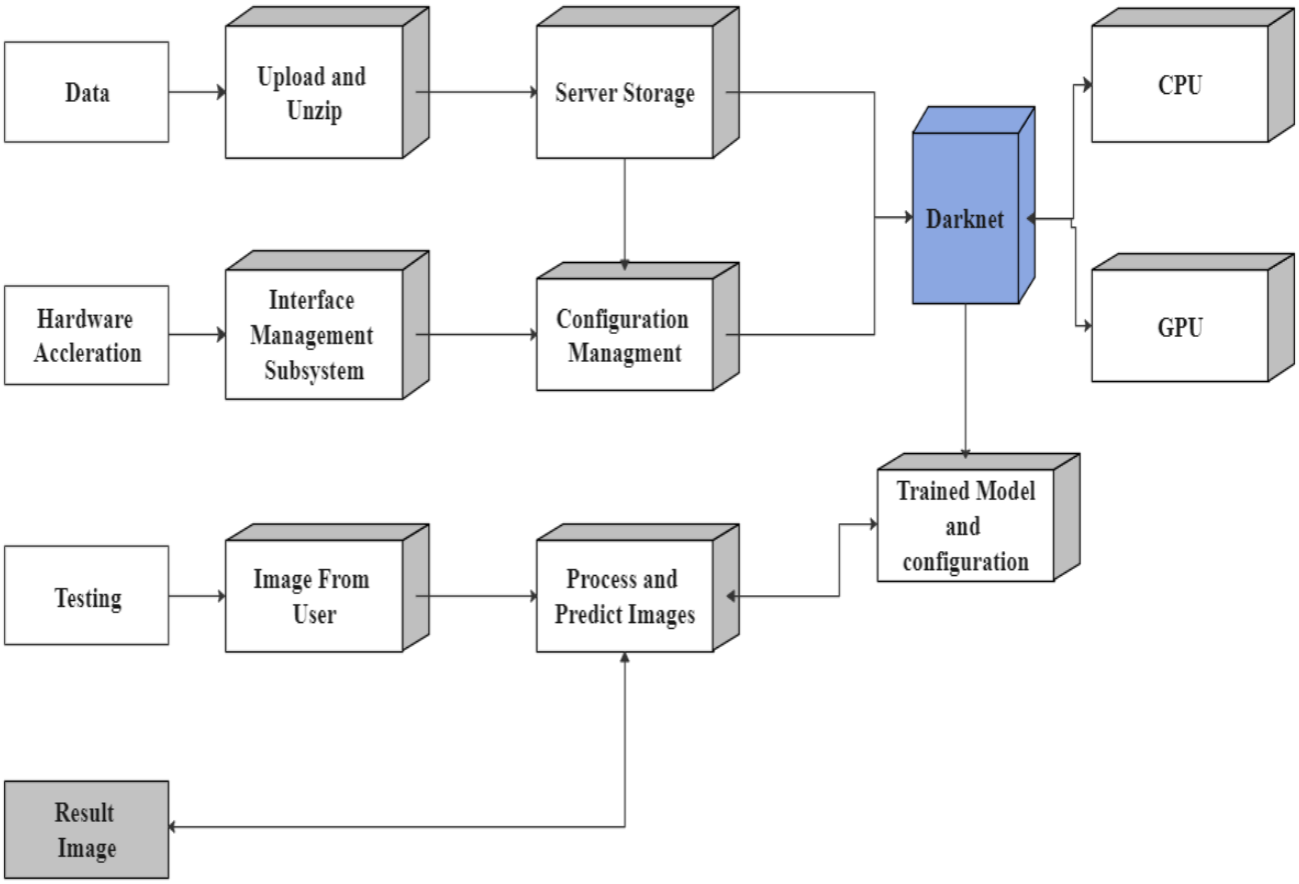
4.3 Features of New System

- Makes the object detection Process easy.

- No need to prepare the training environment for different data.
- Makes the training automated.

5. System Design

5.1 System Architecture Design



5.2 Snapshots

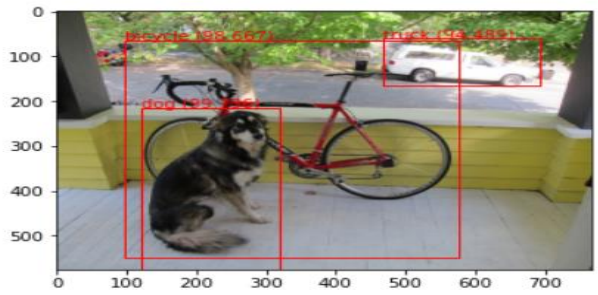
Phase I: Pre-trained Model:

```
In [2]:
# model = tf.keras.models.load_model("model/yolo")
model = load_model('model/yolo.h5')
model.summary()
```

conv_0 (Conv2D)	(None, None, None, 3 864	input_1[0][0]
bnorm_0 (BatchNormalization)	(None, None, None, 3 128	conv_0[0][0]
leaky_0 (LeakyReLU)	(None, None, None, 3 0	bnorm_0[0][0]
zero_padding2d (ZeroPadding2D)	(None, None, None, 3 0	leaky_0[0][0]
conv_1 (Conv2D)	(None, None, None, 6 18432	zero_padding2d[0][0]
bnorm_1 (BatchNormalization)	(None, None, None, 6 256	conv_1[0][0]
leaky_1 (LeakyReLU)	(None, None, None, 6 0	bnorm_1[0][0]
conv_2 (Conv2D)	(None, None, None, 3 2048	leaky_1[0][0]
bnorm_2 (BatchNormalization)	(None, None, None, 3 128	conv_2[0][0]

```
In [7]:
detect_objects('images/dog.jpg')
```

```
truck
bicycle
dog
```



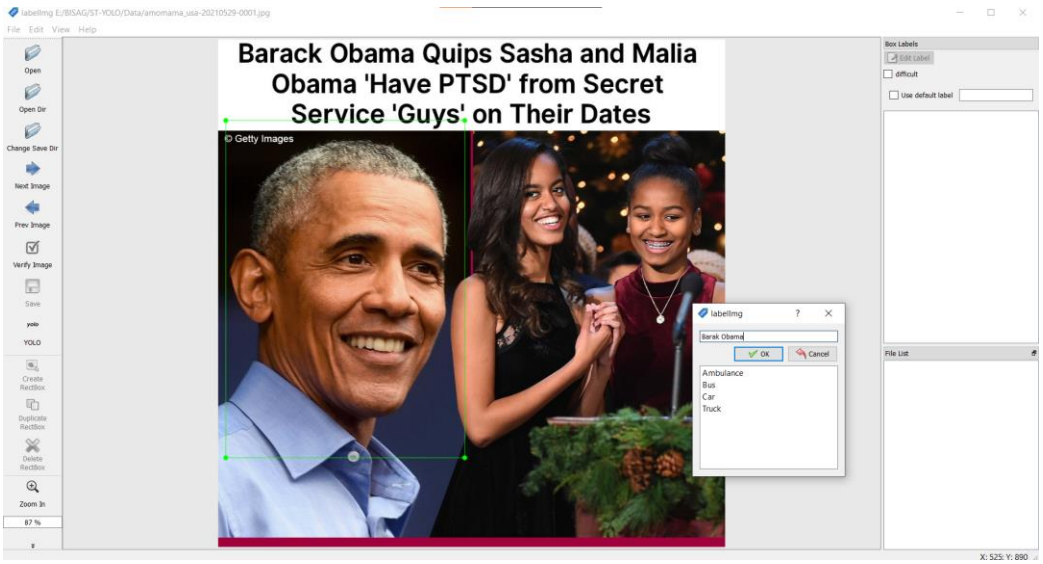

```
Out[7]: {'truck': 1, 'bicycle': 1, 'dog': 1}
```

```
In [10]:
detect_objects('images/houseofobjects.in-20210527-0001.jpg')
```

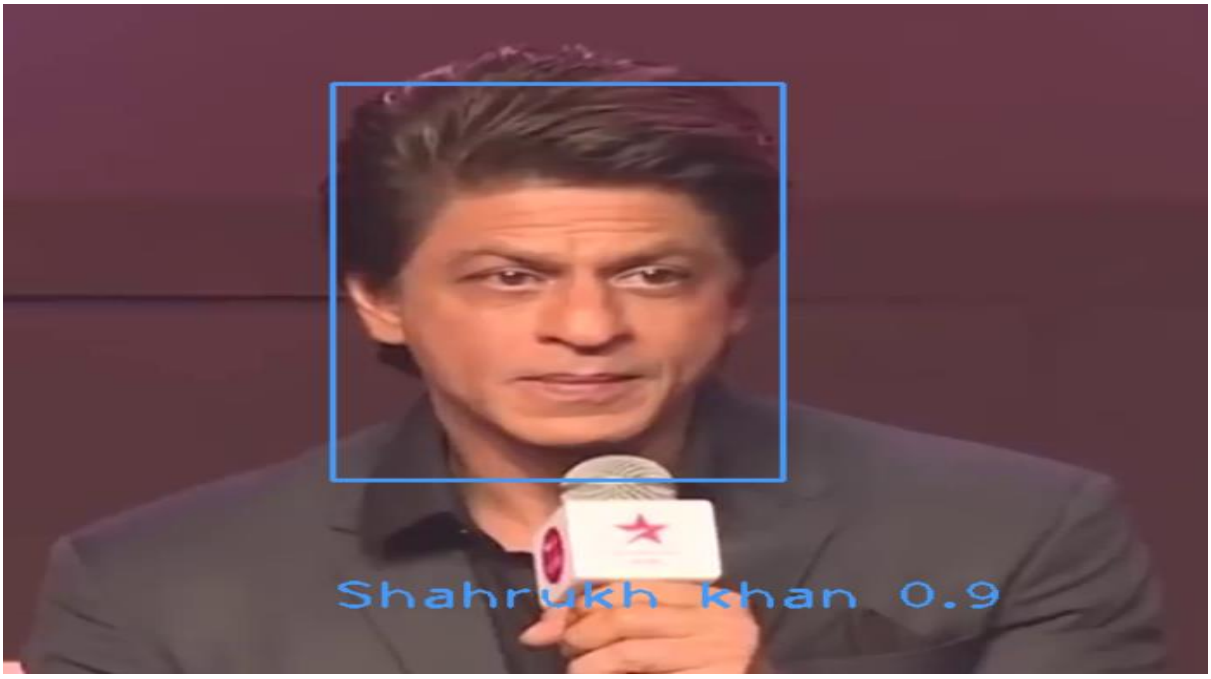
```
couch
potted plant
potted plant
potted plant
bottle
bottle
```


```
Out[10]: {'couch': 1, 'potted plant': 3, 'bottle': 2}
```

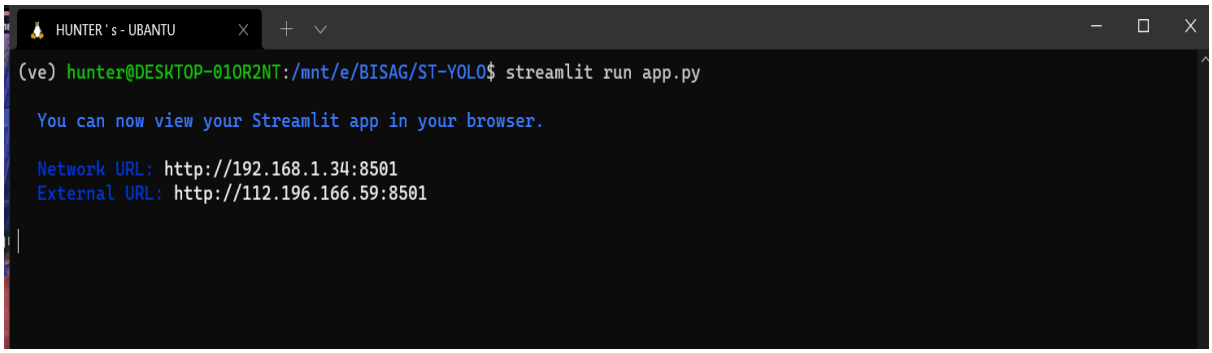
Phase II: Trained for Custom Objects:

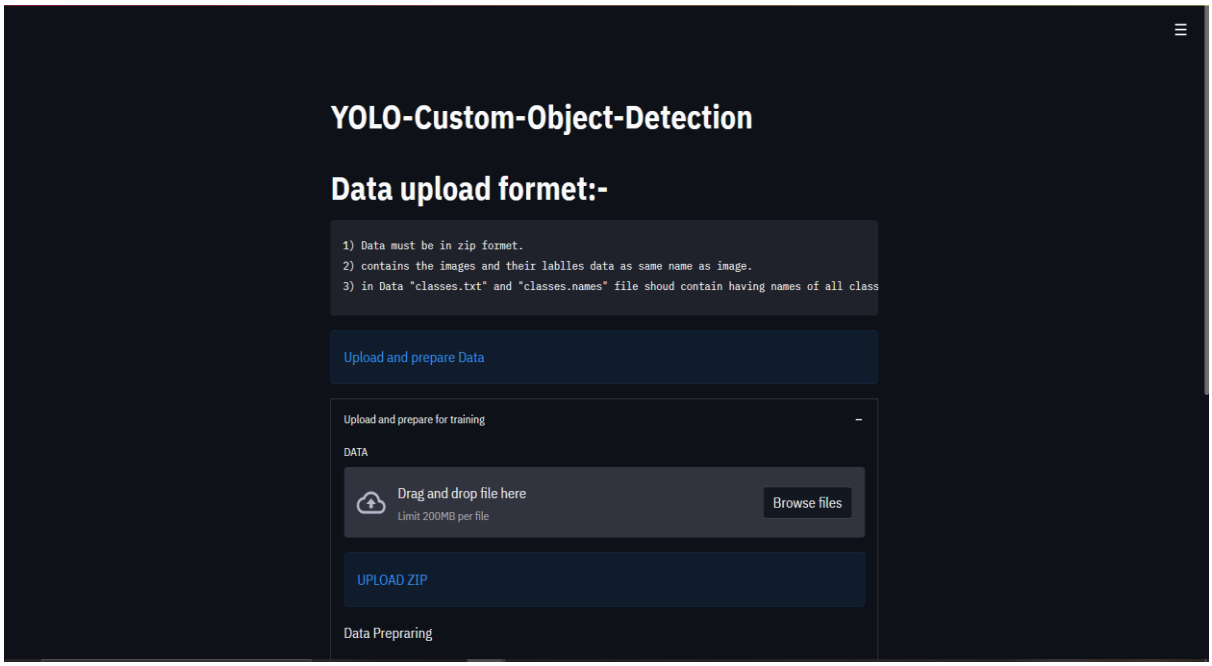


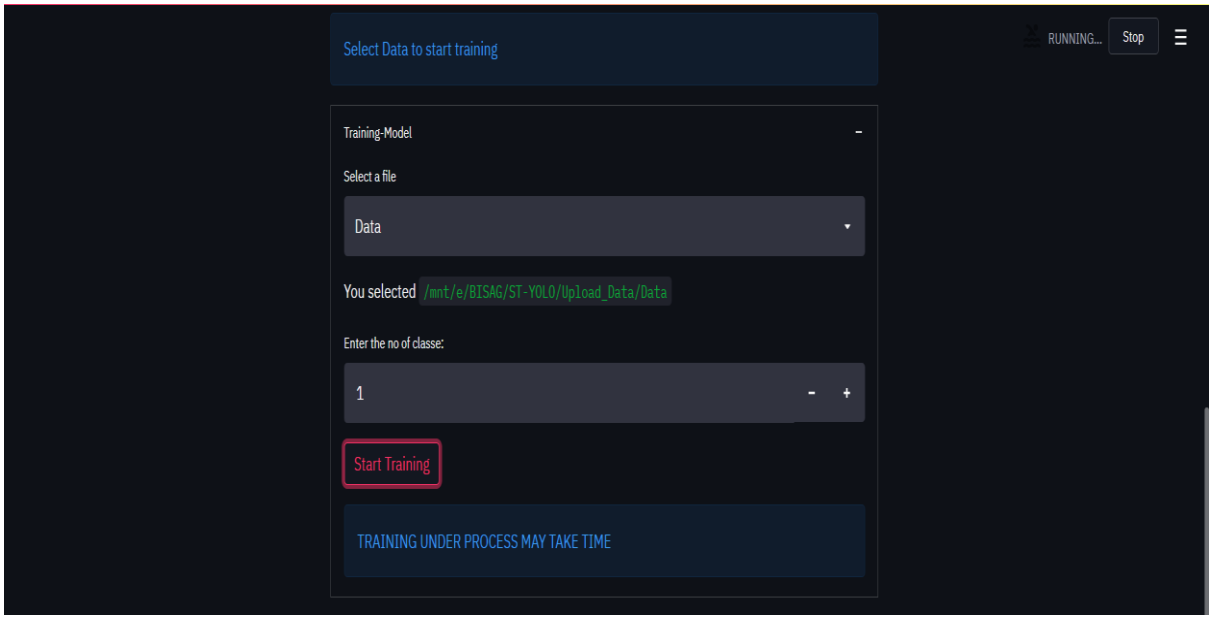

```
%%time
!darknet/darknet detector train Data/labelled_data.data darknet/cfg/yolov3_custom.cfg Custom_weights/darknet53.conv.74 -dont_sh
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000
 total_bbox = 139369, rewritten_bbox = 0.000000 %

 2221: 0.084809, 0.099398 avg loss, 0.001000 rate, 5.997635 seconds, 142144 images, 10.128417 hours left
Loaded: 0.000048 seconds
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.821489), count: 3, class_loss = 0.104325, iou_loss = 0.096840, total_loss = 0.201165
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.764157), count: 1, class_loss = 0.001477, iou_loss = 0.033274, total_loss = 0.034751
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.339067), count: 1, class_loss = 0.456583, iou_loss = 0.788732, total_loss = 1.245315
 total_bbox = 139374, rewritten_bbox = 0.000000 %
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 82 Avg (IOU: 0.707260), count: 3, class_loss = 0.186234, iou_loss = 0.168126, total_loss = 0.354360
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 94 Avg (IOU: 0.911600), count: 1, class_loss = 0.017911, iou_loss = 0.008418, total_loss = 0.026329
v3 (mse loss, Normalizer: (iou: 0.75, obj: 1.00, cls: 1.00) Region 106 Avg (IOU: 0.000000), count: 1, class_loss = 0.000000, iou_loss = 0.000000, total_loss = 0.000000
 total_bbox = 139378, rewritten_bbox = 0.000000 %
```

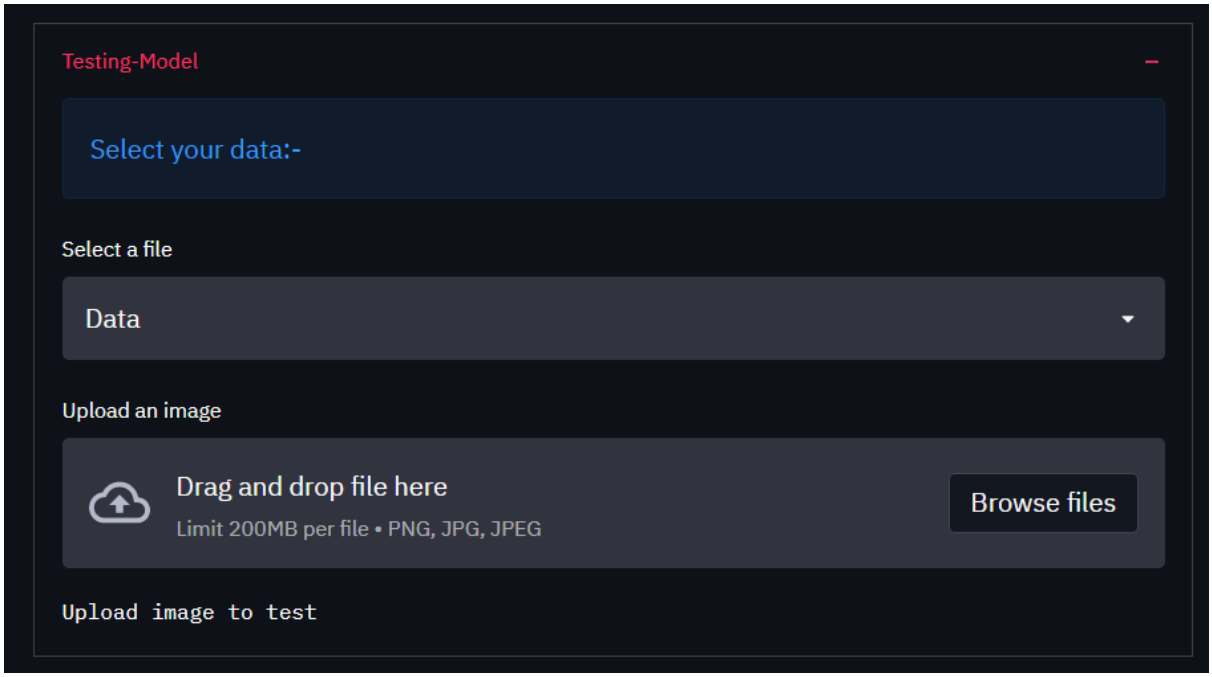


Phase III: Automated Web Application:









6. Implementation Planning

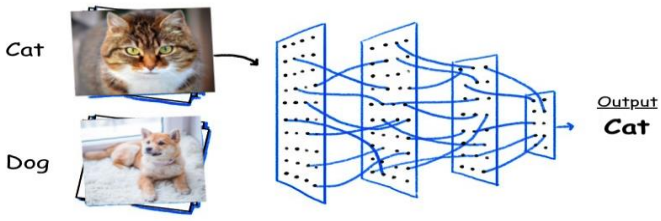
Phase I:

We are using YOLOv3, in particular, YOLO trained on the COCO dataset. The COCO dataset consists of 80 labels, including, but not limited to: People Bicycles Cars and trucks Airplanes Stop signs and fire hydrants Animals, including cats, dogs, birds, horses, cows, and sheep, to name a few Kitchen and dining objects, such as wine glasses, cups, forks, knives, spoons, etc.

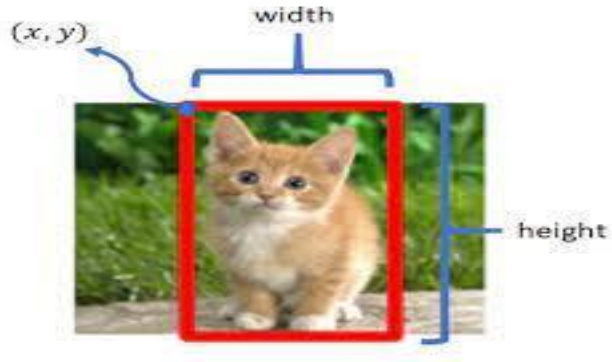
Object Detection

To explore the concept of object detection it is useful, to begin with, image classification. It goes through levels of incremental complexity.

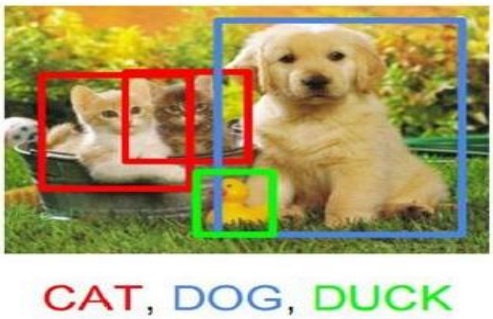
Image classification aims at assigning an image to one of several different categories (e.g. car, dog, cat, human, etc.), essentially answering the question “What is in this picture?”. One image has only one category assigned to it.



Object localization then allows us to locate our object in the image, so our question changes to "What is it and where it is?". In a real real-life scenario, we need to go beyond locating just one object but rather multiple objects in one image. For example, a self-driving car has to find the location of other cars, traffic lights, signs, humans and take appropriate action based on this information.



Object detection provides the tools for doing just that – finding all the objects in an image and drawing the so-called bounding boxes around them. There are also some situations where we want to find the exact boundaries of our objects in the process called instance segmentation, but this is a topic for another post.



Phase II:

Gathering Data

One of the crucial parts of building machine learning systems is gathering high-quality dataset. It is expected to spend a significant amount of time on data. It is essential because our model is

only as good as the data it learns from. Here we are going to detect **'Narendra Modi', 'Barack Obama', 'Shahrukh khan' and 'Cristiano Ronaldo'**.

Labelling

Once we gather data, the next step is to label/annotate them. In the context of object detection, labelling means drawing bounding boxes around the objects that we are interested in detecting in the images and associating them with corresponding object classes/categories so that we can show them to the machine clearly. This is the most labour-intensive process.

There are a lot of tools available to help us annotate the images. One of the widely used Open Source tools is LabelIng. The good thing about LabelIng is, it lets us save the annotations directly into YOLO format. Some tools don't directly do that. We need to convert the annotations ourselves into the format YOLO requires.

Labelling can be easily installed from: https://tzutalin.github.io/labelImg/

Generating train.txt

The last configuration file needed before we begin to train our custom detector is the train.txt file which holds the relative paths to all our training images.

```
from glob import glob
imgs_list = glob("Data/*jpg")
file = open("Data/train.txt",'w')
file.write("\n".join(imgs_list))
file.close()
```

Phase III:

Automation the process of training needs to have prior knowledge of object detection and Deep Learning.

To develop this phase there is a need for a technology that can provide us with good front end web application development. Which is being fulfilled by the Framework Named **Streamlit** provides convenience to develop a basic web application.

The next Challenge id automation is to automate the making of the configuration environment which involves some of the commands/shell commands, to solve that Shell scripting is the most feasible and optimized solution which can be used to automate the shell commands to run and create a work feasible environment.

Subprocess and the popular library of python that is used to run shell commands in the background while running another process in front.

7. Limitations and Future Extensions

The object recognition system can be applied in the area of the surveillance system, face recognition, fault detection, character recognition etc. The objective of this thesis is to develop an object recognition system to recognize the 2D and 3D objects in the image. The performance of the object recognition system depends on the features used and the classifier employed for recognition. This research work attempts to propose a novel feature extraction method for extracting global features and obtaining local features from the region of interest. Also, the research work attempts to hybrid the traditional classifiers to recognize the object.

It is important to mention the difficulties observed during the experimentation of the object recognition system due to several features present in the image. The research work suggests that the image is to be preprocessed and reduced to a size of 128 x 128. The proposed feature extraction method helps to select the important feature. To improve the efficiency of the classifier, the number of features should be less in number. Specifically, the contributions towards this research work are as follows

- An object recognition system is developed that recognizes two-dimensional and three-dimensional objects.

- The feature extracted is sufficient for recognizing the object and marking the location of the object. x

- The proposed classifier can recognize the object with a recognition system lossless computational cost. The proposed global feature extraction requires less time, compared to the traditional feature extraction method.

- The performance of the SVM-kNN is greater and promising when compared with the BPN and SVM.

- The performance of the One-against-One classifier is efficient. 38 Global features extracted from the local parts of the image.

- Local feature PCA-SIFT is computed from the blobs detected by the Hessian-Laplace detector.

- Along with the local features, the width and height of the object computed through the project option met are with use.

The methods presented for feature extraction and recognition are common and can be applied to any application that is relevant to object recognition. The proposed object recognition method combines the state-of-art classifier SVM and k-NN to recognize the objects in the image. The multiclass SVM is used to hybridize with the k-NN for recognition. The feature extraction method proposed in this research work is efficient and provides unique information for the classifier. The image is segmented into 16 parts, from each part the Hu's Moment invariant is computed and it is converted into the Eigen component. The local feature of the image is obtained by using the Hessian-Laplace detector. This helps to obtain the object's features easily and mark the object location without much difficulty.

As the scope for future enhancement:-

- Features either local or global used for recognition can be increased, rease the efficiency of the object recognition system.

- Geometric properties of the image can be included in the feature vector for recognition. 150 Using an unsupervised classifier instead of a supervised classifier for recognition of the object.

- The proposed object recognition system uses grey-scale image and discards the colour information.

- The colours information in the image can be used for the recognition of the object. Colour based object recognition plays a vital role in Robotics Although the visual tracking algorithm proposed here is robust in many of the conditions, it can be made more robust by eliminating some of the limitations as listed below:

- In the Single Visual tracking, the size of the template remains fixed for tracking. If the size of the object reduces with time, the background becomes more dominant than the object being tracked. In this case, the object may not be tracked.

- A fully occluded object cannot be tracked and considered as a new object in the next frame.

- Foreground object extraction depends on the binary segmentation which is carried out by applying threshold techniques. So blob extraction and tracking depend on the threshold value.

- Splitting and merging cannot be handled very well in all conditions using the single-camera due to the loss of information of a 3D object projection in 2D images.

- For Nighttime visual tracking, night vision mode should be available as an inbuilt feature in the CCTV camera. To make the system fully automatic and also to overcome the above limitations, in future, multi-view tracking can be implemented using multiple cameras.
- Multi-view tracking has the obvious advantage over single view tracking because of the wide coverage range with different viewing angles for the objects to be tracked.

- In the proposed method, a background subtraction technique has been used that is simple and fast. This technique is applicable where there is no movement of the camera.

- For robotic application or automated vehicle assistance system, due to the movement of the camera, backgrounds are continuously changing leading to the implementation of some different segmentation techniques like single Gaussian mixture or multiple Gaussian mixture models.

- Object identification task with motion estimation needs to be fast enough to be implemented for the real-time system. Still, there is a scope for developing faster algorithms for object identification. Such algorithms can be implemented using FPGA or CPLD for fast execution.

8. Conclusion & Reference

8.1 Conclusion

While Training a custom object detection algorithm, there are a lot of challenges and knowledge is needed, and there are fewer resources available.
To Train a model there is a need to set up a whole environment which takes time and also has a high probability of things don't work properly.
So, to overcome that problem, the proposed web application contains the automation of this whole process, only need to prepare the data in the giver format and upload it.
The application provides an interface to train the model and gives you a place to test it as well.
This will reduce the efforts taken to train an object detection model to 80%.

8.2 Reference

- https://github.com/theAIGuysCode/YOLOv3-Cloud-Tutorial/blob/master/YOLOv3_Tutorial.ipynb
- https://www.youtube.com/watch?v=4eIBisqx9_g&t=841s

- https://www.youtube.com/watch?v=h56M5iUVgGs&t=1078s
- https://www.youtube.com/watch?v=ag3DLKsl2vk
- https://www.youtube.com/watch?v=b59xfUZZqJE&t=1720s
- https://www.youtube.com/watch?v=hTCmL3S4Obw
- https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/
- https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framewor-python/
- https://pjreddie.com/darknet/yolo/
- https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006
- https://www.researchgate.net/publication/337464355_OBJECT_DETECTION_AND_IDENTIFICATION_A_Project_Report
- https://github.com/AlexeyAB/darknet
- https://arxiv.org/abs/2004.10934
- https://www.ijeat.org/wp-content/uploads/papers/v8i3S/C11240283S19.pdf

Bhaskaracharya National Institute for Space Applications and Geo-informatics

MeitY, Government of India

Phone: 079 - 23213081 Fax: 079 - 23213091

E-mail: info@bisag.gujarat.gov.in, website: https://bisag-n.in/

BISAG-N

ISO 9001:2008
ISO 27001:2013
CMMI LEVEL-5

Report Verification Procedure

Date:

Project Name: YOLO Object Detection

Student Name	Student ID/Team ID
Shashank Mathur	E1
Mohammed Adil	E1
Meha Deora	E1
Nishtha Gehlot	E1

	Soft Copy	Hard Copy
Report Format		
Project Index		

Sign by Training Coordinator **Sign by Project Guide**