VIDEO BASED EVIDENCE ANALYSIS AND EXTRACTION IN DIGITAL FOREIGNSIC INVESTIGATION

A PROJECT REPORT

Submitted in the partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

K. VISHWA SAI	**17B81A05V5**
D.SUDHEER KUMAR	**17B81A05S1**
K. VISHAL REDDY	**17B81A05V3**

Under the guidance of

Mrs.M.Archana

Assistant Professor, CSE Department

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CVR COLLEGE OF ENGINEERING

(An Autonomous institution, NBA, NAAC Accredited and Affiliated to JNTUH, Hyderabad)

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M),
Rangareddy (D), Telangana-501510

May 2021

Cherabuddi Education Society's

CVR COLLEGE OF ENGINEERING

(An Autonomous Institution)

ACCREDITED BY NBA, NAAC 'A' Grade

(Approved by AICTE & Government of Telangana and Affiliated to JNTU Hyderabad)

Vastunagar, Mangalpalli (V), Ibrahimpatnam (M), R.R.District.

Web: http://www.cvr.ac.in, email: info@cvr.ac.in

Ph : 08414 – 252222, 252369, Office Telefax : 252396, Principal : 252396 (O)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**VIDEO BASED EVIDENCE ANALYSIS AND EXTRACTION IN DIGITAL FORENSIC INVESTIGATION**" being submitted by **K.Vishwasai** (17B81A05V5), **D.Sudheer Kumar** (17B81A05S1), **K.Vishal Reddy** (17B81A05V2) in partial fulfillment of the requirement for the award of degree of Bachelor of Technology in Computer Science and Engineering to the CVR College of Engineering, is a record of bonafide work carried out by them under my guidance and supervision during the year 2020-2021.

The results embodied in this project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Mrs.M.Archana	**Dr.A. Vani Vathsala**
Asst. Professor, CSE	HOD, CSE
CVR College of Engineering	CVR College of Engineering

External Examiner

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose encouragement and guidance has been a source of inspiration throughout the course of the project.

It is great pleasure to convey our profound sense of gratitude to our Principal **Dr. Nayanathara K. S**, Vice-Principal **Prof. L. C. Siva Reddy**, Head of CSE Department **Dr.A. Vani Vathsala**, CVR College of Engineering, for having been kind enough for arranging necessary facilities for executing the project in the college.

We wish to reciprocate in full measure the kindness shown by **Dr R K.SelvaKumar** ,Professor , CSE and **Ms. Sharada**, Assistant Professor, CSE who inspired us with her valuable suggestions and guiding us timely in successfully completing the project work. We shall remain grateful to her for providing us strong atmosphere by enforcing strict discipline to do the project with utmost concentration and dedication.

We deem it a pleasure to acknowledge our sense of gratitude to our project guide **Mrs. M.Archana** under whom we have carried out the project work. His incisive and objective guidance and timely advice encouraged us with constant flow of energy to continue the work.

We wish a deep sense of gratitude and heartfelt thanks to CVR Management for providing excellent lab facilities and tools. Finally, we thank all those who gave us immense support directly and indirectly in this regard.

K.VISHWASAI, 17B81A05V5

D.SUDHEER KUMAR, 17B81A05S1

K.VISHAL REDDY 17B81A05V3

ABSTRACT

Evidence Analysis and Anomaly Detection is a challenging research in the field of image processing and computer vision, especially for security systems based on the face image recognition. It is also one of the most interesting and important research field in the past two decades. The reason comes from the need of automatic recognition and surveillance systems. Evidence Analysis and Anomaly Detection is an important application of Image processing owing to its use in many fields. The project presented here was developed after study of various Evidence Analysis and Anomaly Detection methods and their Hence there is a need for some software application which detects faces in real time with minimal error and maximum accuracy the need of automatic recognition and surveillance systems is very high these days.

Especially the analysis of CCTV footage and other video data for finding suspects in the video/image data is very crucial in crime and forensic analysis.

The project presented here was developed after studies of various Evidence Analysis and Anomaly Detection methods and their efficiencies.

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

1.1 MOTIVATION

Evidence analysis and Anomaly detection has been a sought out after problem of biometrics and it has a variety of applications in modern life. The problems of Evidence Analysis and Anomaly Detection attracts researchers working in biometrics, pattern recognition field and computer vision. Several Evidence Analysis and Anomaly Detection algorithms are also used in many different applications apart from biometrics, such as video compressions, indexing etc

An efficient Evidence Analysis and Anomaly Detection system can be of great help in forensic sciences, Identification for law enforcement, surveillance and giving preferential access to authorized users i.e. access control for secured areas etc. The problem of Evidence Analysis and Anomaly Detection has gained even more importance after the recent increase in the terrorism related incidents.

The cost of the license for an efficient commercial Evidence Analysis and Anomaly Detection system ranges from 30,000 $ to 150,000 $ which shows the significant value of the problem.

Though Evidence Analysis and Anomaly Detection is considered to be a very crucial authentication and anomaly detection system but even after two decades continuous research and evolution of many Evidence Analysis and Anomaly Detection algorithms, a truly robust and efficient system that can produce good results in real time and normal conditions is still not available.

1.2 PROBLEM STATEMENT

- Due to popularity of mobile devices and low cost surveillance systems, visual data is been used in forensic investigation.
- The main goal of the project is to detect and alert , when there are criminal activities and other riots found in CCTV and other video data.

Desired Solution: The solution should focus on developing an application to capture a photo from video recording and search for the same on official websites / social media websites / internet using an optimized Evidence Analysis and Anomaly Detection algorithm and also to store the evidence in the evidence repository

1.3 PROJECT OBJECTIVES

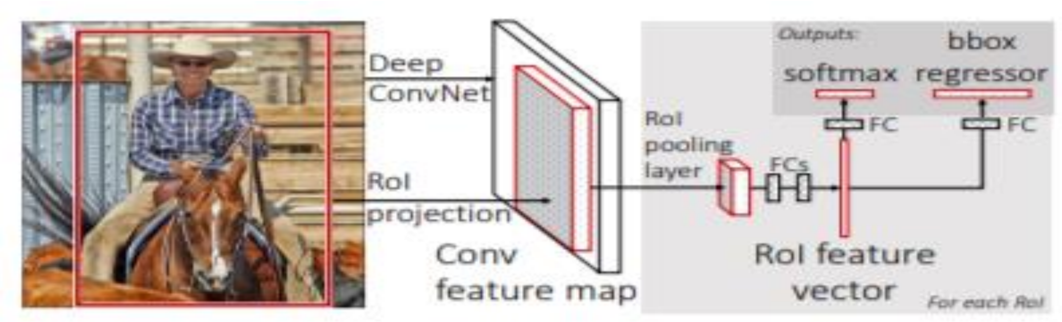
- To help in detecting criminals in public.
- To help in detecting anomaly persons in a public area
- To detect riots occurrence before hand and hence help in preventing them.

2. LITERATURE SURVEY

2.1 EXISTING SYSTEM

<u>Fast R-CNN</u>

Fast R-CNN is an improved version of R-CNN as it had some disadvantages like multistage pipelining environment, space and time expensive, slow object detection. To remove them Fast R-CNN introduced a new structure.



2.1.1Fast R-CNN Architecture, source: Fast R-CNN

It takes the entire image as an input along with the object proposals. Initially the algorithm runs a CNN on the input image and forms a feature map by the use of various conv and max pooling layers. After that for each object proposal a Region of Interest (RoI) pooling layer extracts a fixed length feature vector and inputs it into a Fully Connected (FC) Layer. This layer further branches into two output layers : one producing the SoftMax probability for each class along with a “background” class, the other layer outputs four real numbers for each of the classes which define the bounding box for that specific class.

2.2 LIMITATIONS OF THE EXISTING SYSTEM

Problem with R-CNN:

Each image needs to classify 2000 region proposals. So, it takes a lot of time to train the network.It requires 49 seconds to detect the objects in an image on GPU.

To store the feature map of the region proposal, lots of Disk space is also required.

Most of the time taken by Fast R-CNN during detection is a selective search region proposal generation algorithm. Hence, it is the bottleneck of this architecture which was dealt with in Faster R-CNN.

2.3 OUR SOLUTION

- <u>Desired Solution:</u> The solution should focus on developing an application to capture a photo from a video recording and search for the same on official websites / social media websites / internet using an optimized Evidence Analysis and Anomaly Detection algorithm to easily detect criminals.

3. SOFTWARE AND HARDWARE SPECIFICATIONS

3.1 HARDWARE REQUIREMENTS:

- **CPU :** Core i5(2.8 GHz or above, at least quad core)
- **RAM :** 8GB minimum (16 GB recommended)
- **GPU :** NVidia graphics (minimum 3GB or more like 8GB)
  (Like GeForceGTX 1080Ti)

3.2 SOFTWARE REQUIREMENTS:

- **Language :** Python
- **Frameworks :** YOLO darknet repository
- **Library used** : NumPy, Matplotlib, OS, pandas etc.

4. DESIGN

4.0 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object - oriented computer software. In its current form UML is comprised of two major components: A Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.
The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing object- oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

<u>GOALS:</u>

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.

6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

4.1USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted

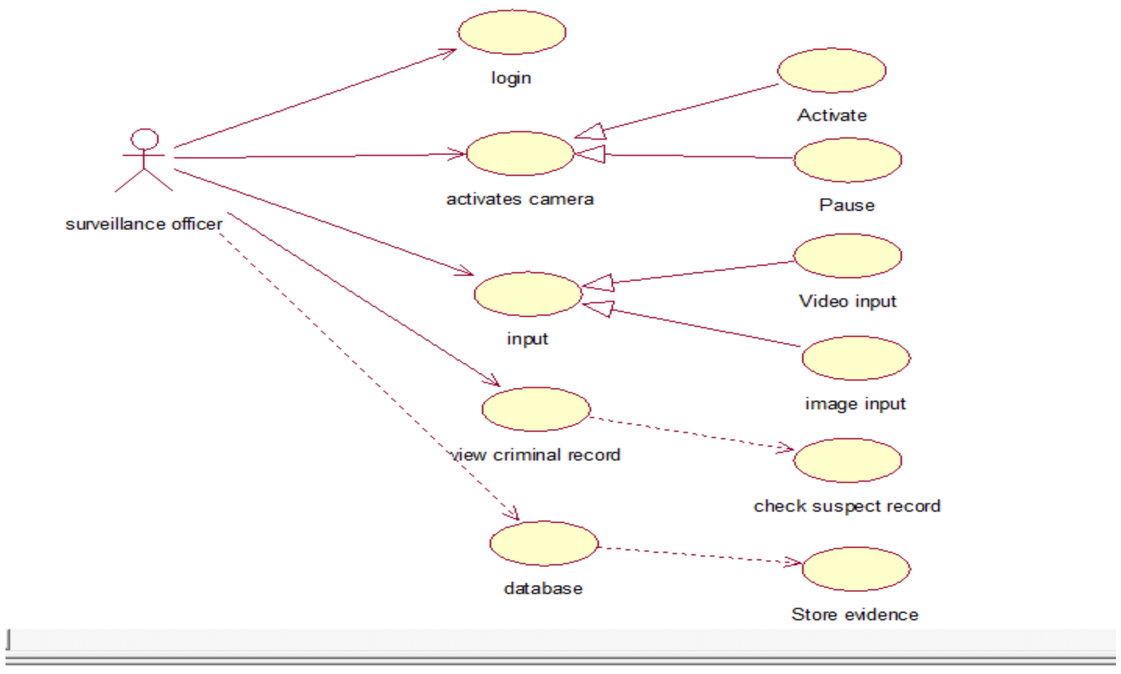


Figure 4.1 Use Case Diagram of Evidence Analysis and Anomaly Detection System

4.2 SEQUENCE DIAGRAM:

-

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

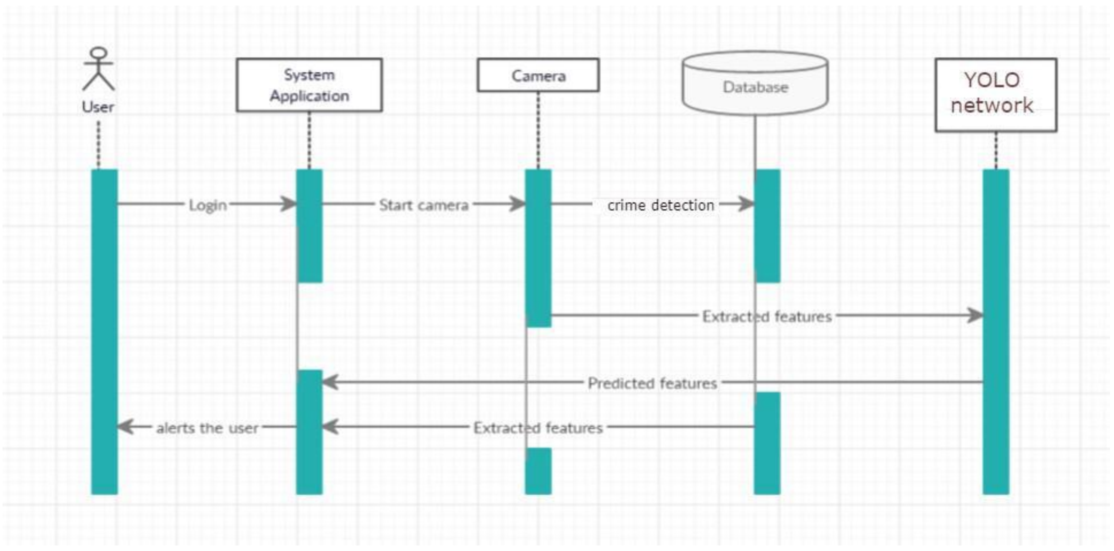


Figure 4.2 Sequence Diagram of Evidence analysis and Anomaly detection System

4.3 ACTIVITY DIAGRAM:

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another as shown in below.

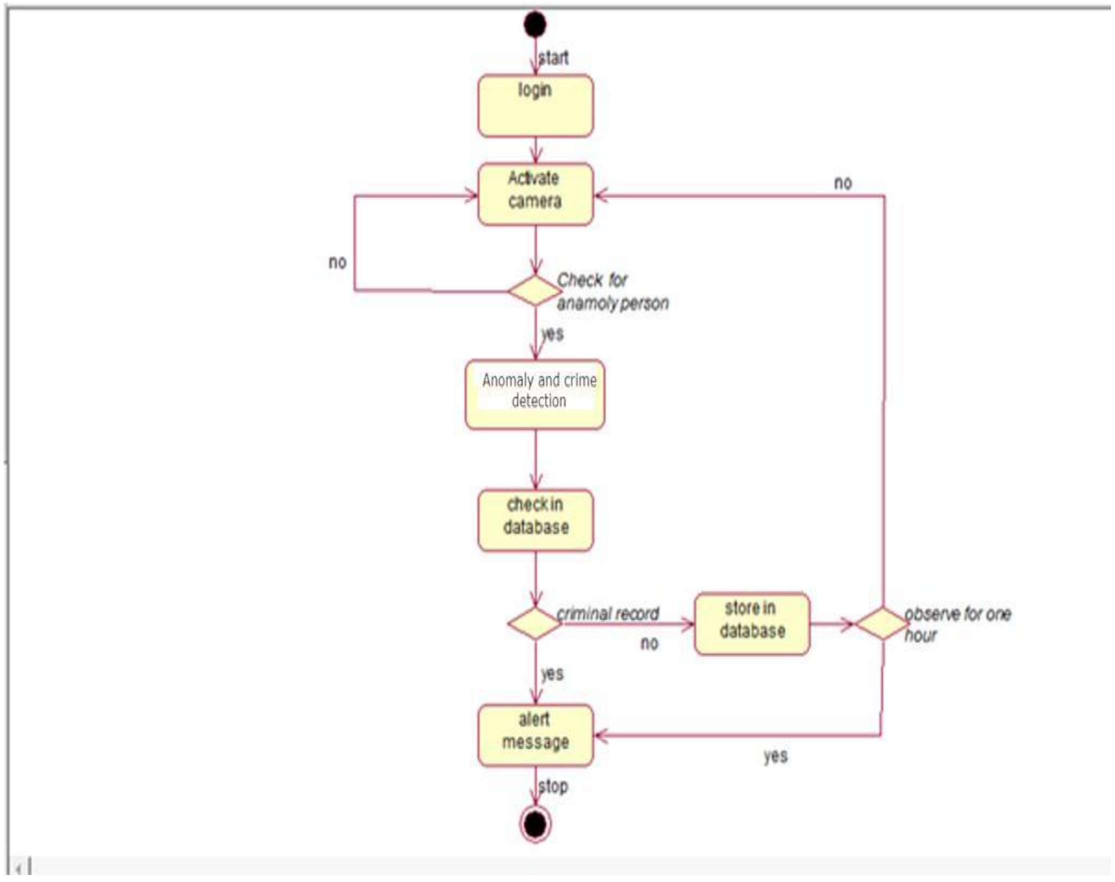


Figure 4.3 Activity Diagram Of Evidence Analysis and Anomaly Detection system

4.4 SYSTEM ARCHITECTURE DIAGRAM:

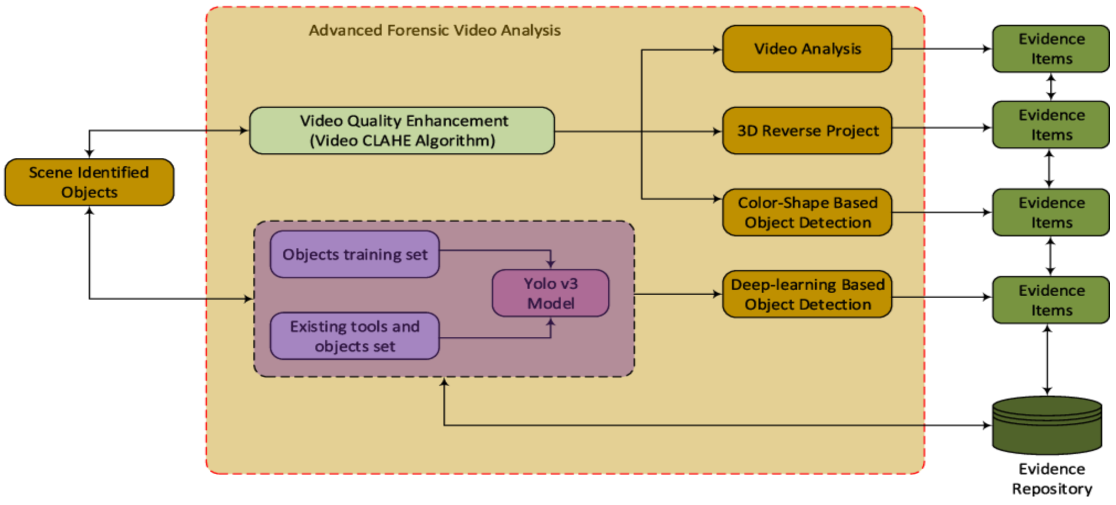

Advanced Forensic Video Analysis
Scene Identified Objects
Video Quality Enhancement (Video CLAHE Algorithm)
Objects training set
Yolo v3 Model
Existing tools and objects set
Video Analysis
3D Reverse Project
Color-Shape Based Object Detection
Deep-learning Based Object Detection
Evidence Items
Evidence Items
Evidence Items
Evidence Items
Evidence Repository

Figure 4.4 System Architecture Diagram of Evidence Analysis and Anomaly Detection System

4.5 Technology Description

YOLO Algorithm

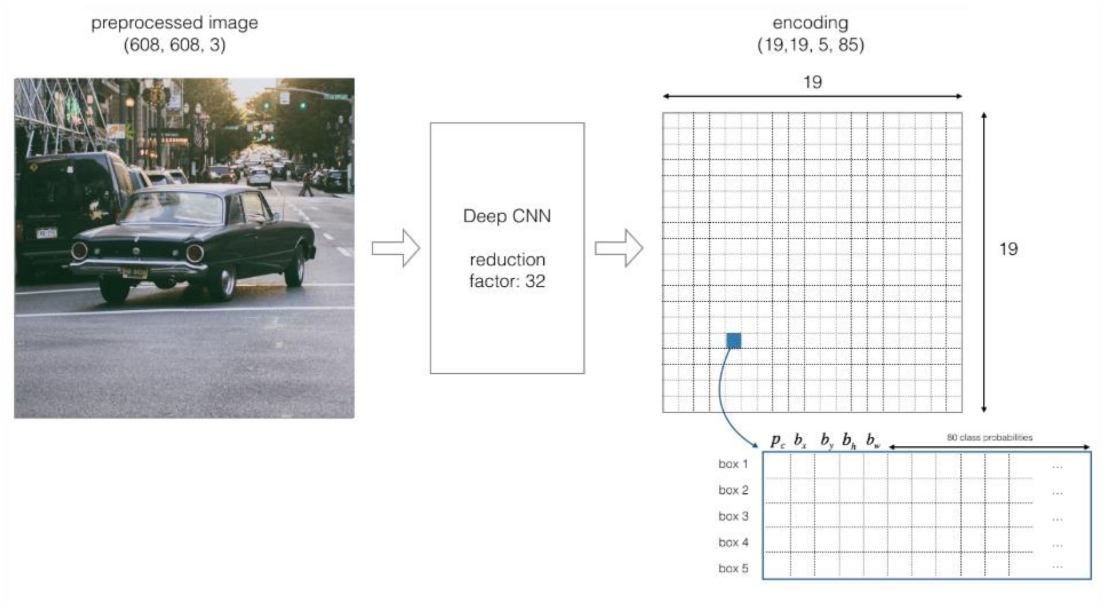
YOLO algorithm gives a much better performance on all the parameters with a high fps for real-time usage. YOLO algorithm is an algorithm based on regression, instead of selecting the interesting part of an Image, it predicts classes and bounding boxes for the whole image in **one run of the Algorithm.**

To understand the YOLO algorithm, first we need to understand what is actually being predicted. Ultimately, we aim to predict a class of an object and the bounding box specifying object location. Each bounding box can be described using four descriptors:

1. Center of the box (**bx, by**)
2. Width (**bw**)
3. Height (**bh**)
4. Value **c** corresponding to the class of an object

Along with that we predict a real number **pc**, which is the probability that there is an object in the bounding box.

YOLO doesn't search for interested regions in the input image that could contain an object, instead it splits the image into cells, typically 19x19 grid. Each cell is then responsible for predicting K bounding boxes.



4.5.1Deep CNN

An Object is considered to lie in a specific cell only if the center co-ordinates of the anchor box lie in that cell. Due to this property the center co-ordinates are always calculated relative to the cell whereas the height and width are calculated relative to the whole Image size.

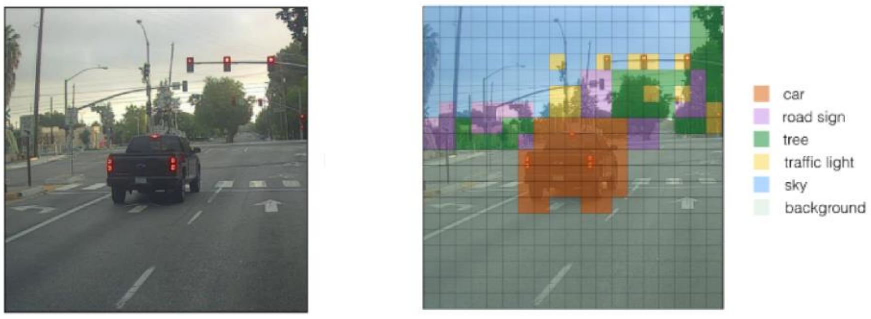
During the one pass of forwards propagation, YOLO determines the probability that the cell contains a certain class. The equation for the same is :

$$score_{c,i} = p_c \times c_i$$

Probability that there is an object of certain class ‘c’

The class with the maximum probability is chosen and assigned to that particular grid cell. Similar process happens for all the grid cells present in the image.

After computing the above class probabilities, the image may look like this :

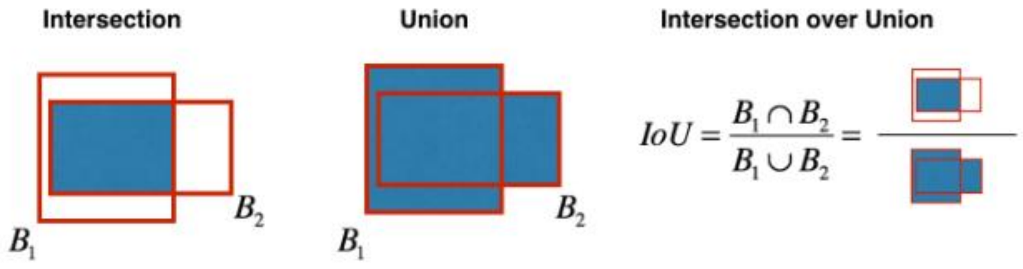


4.5.2 Bounding Box

This shows the before and after of predicting the class probabilities for each grid cell. After predicting the class probabilities, the next step is Non-max suppression, it helps the algorithm to get rid of the unnecessary anchor boxes, like you can see that in the figure below, there are numerous anchor boxes calculated based on the class probabilities.

4.5.3 Anchor boxes

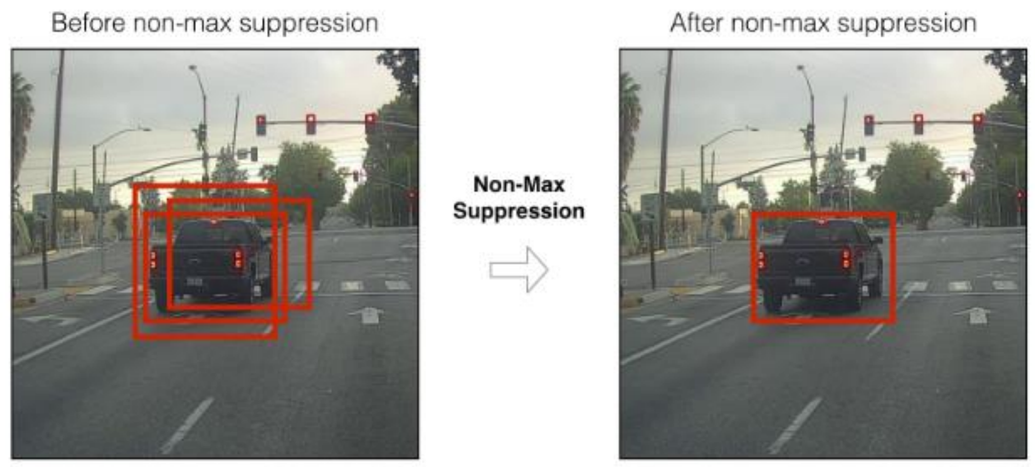
To resolve this problem Non-max suppression eliminates the bounding boxes that are very close by preforming the IoU (Intersection over Union) with the one having the highest class probability among them.



4.5.4 IoU operation

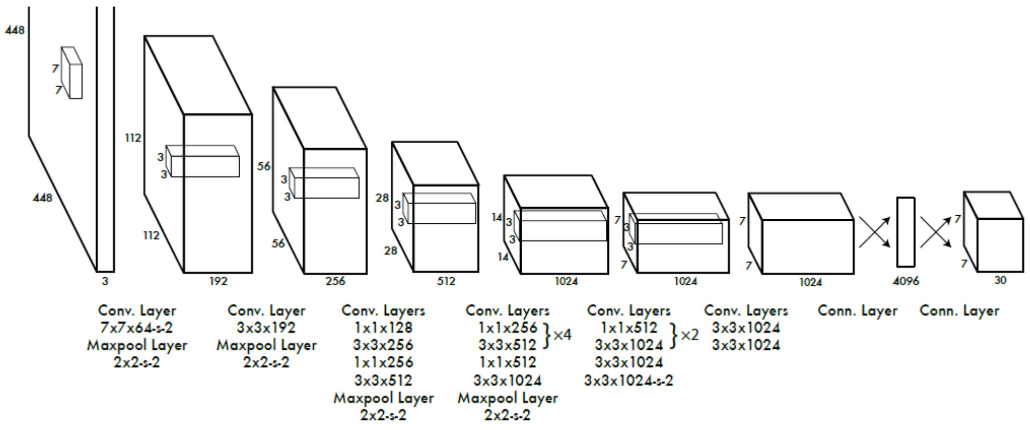
It calculates the value of IoU for all the bounding boxes respective to the one having the highest class probability, it then rejects the bounding boxes whose value of IoU is greater than a threshold. It signifies that those two bounding boxes are covering the same object but the other one has a low probabilty for the same, thus it is eliminated.

Once done, algorithm finds the bunding box with next highest class probabilities and does the same process, it is done until we are left with all the different bounding boxes.



4.5.5 Before and After of Non-max suppression

After this, almost all of our work is done, the algorithm finally outputs the required vector showing the details of the bounding box of the respective class. The overall architecture of the algorithm can be viewed below :



4.5.6 YOLO Architecture

Also, the most important parameter of the Algorithm, its Loss function is shown below. YOLO simultaneously learns about all the four parameters it predicts (discussed above).

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Loss function for YOLO

So this was all about the YOLO Algorithm. We discussed all the aspects of Object detection along with the challenges we face in that domain. We then saw some of the algorithms that tried to solve some of these challenges but were failing in the most crucial one-Real time detection (speed in fps). We then studied the YOLO algorithm which outperforms all the other models in terms of the challenges faced, its fast-can work well in real-time object detection, follows a regression approach.

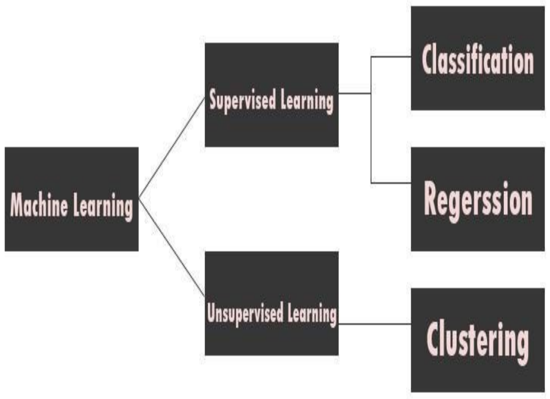
Still improvements are being made in the algorithm. We currently have four generations of the YOLO Algorithm from v1 to v4, along with a slightly small version of it YOLO-tiny, it is specifically designed to achieve a incredibly high speed of 220fps.

IMPORTANT TERMS

MACHINE LEARNING

.

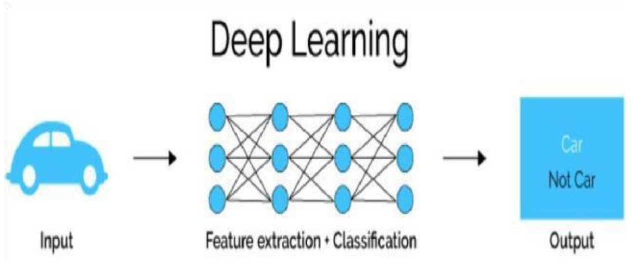
ML or Machine Learning is a study of computer algorithms that improve automatically through experience. It deals with provision of human intelligence to machine system. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms.



4.5.7*: Machine Learning Techniques*

DEEP LEARNING

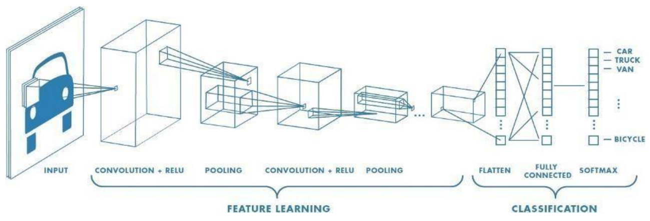
Deep Learning is a family of artificial intelligence. Deep Learning has much architecture in them, such as Deep Neural Network, Convolutional Neural Network, and Recurrent Neural Network etc. Deep Learning is a modern variation concerned with an unbounded number of layers of a bounded size that permit practical application and optimized implementation. Deep Learning is also called hierarchical learning or deep structured learning. In artificial intelligence in the healthcare setting, the agency noted that some deep learning algorithms have already produced transformational outcomes. Deep Learning provides the healthcare industry to analyse data at exceptional speeds without compromising on accuracy. Deep Learning in healthcare has already leftist marks. Deep learning could be reducing the admin load while increasing insight into patient care and requirements. Artificial Intelligence is transforming the operation of medicine. Diagnose diseases from X-Rays and 3D MRI brain images. X-rays are the most used diagnostic imaging test and are widely available.



4.5.8: Deep Learning Technique

CNN MODEL

In deep learning a (CNN or Convnet) Convolutional neural network is a class of deep neural network is the class of deep neural network most commonly applied to analysing image. Convolutional Neural Network are regularized version of multilayer perceptions Multilayer perceptions usually means fully connected network that is every one neuron in one layer is connected to all neuron in a next layer. CNN or Convolutional Neural Network image classification take an input image, process it and classifies it under categories (Egg: Dog, Cat, Car, Medical Field). A computer sets an input image an array of pixels, and it depends on its image resolution. Artificial Intelligence has been observing a monumental growth in bridging the gap between humans and machines' capabilities.



4.5.9: Convolutional Neural Network

The architecture of a convert is analogous to that of the connectivity pattern of neurons in the human brain and was encouraged by the organization. Deep Learning method utilizing deep convolutional neural networks have been applied to medical image analysis providing promising results. The application covers the whole spectrum of medical image analysis, including Detection, Segmentation, and Classification.

5. IMPLEMENTATION & TESTING

5.1 CODE IMPLEMENTATION

The goal of the coding or programming phase is to translate the design of the system produced during the design phase into code in a given programming language, which can be executed by a computer and that performs the computation specified by the design.

The coding phase affects both testing and maintenance. The goal of coding is not to reduce the implementation cost but the goal should be to reduce the cost of later phases. In other words, the goal is not to simplify the job of programmer. Rather the goal should be to simplify the job of the tester and maintainer

Code of the program

We coded on Google colab for faster inference and runtime

1)Training the model

```
from google.colab import drive
drive.mount('/content/gdrive',force_remount=True)


# this creates a symbolic link so that now the path /content/gdri
ve/My\ Drive/ is equal to /mydrive
!ln -s /content/gdrive/My\ Drive/ /mydrive
!ls /mydrive
```

Cloning and Setting Up Darknet for YOLOv4

We will be using the famous AlexeyAB's darknet repository in this tutorial to perform YOLOv4 detections.

```
!git clone https://github.com/AlexeyAB/darknet


%cd darknet
!sed -i 's/OPENCV=0/OPENCV=1/' Makefile
!sed -i 's/GPU=0/GPU=1/' Makefile
```

```
!sed -i 's/CUDNN=0/CUDNN=1/' Makefile
!sed -i 's/CUDNN_HALF=0/CUDNN_HALF=1/' Makefile
!sed -i 's/LIBSO=0/LIBSO=1/' Makefile

# make darknet (builds darknet so that you can then use the darkn
et.py file and have its dependencies)
!make
```

```
# get bthe scaled yolov4 weights file that is pre-
trained to detect 80 classes (objects) from shared google drive
!wget --load-
cookies /tmp/cookies.txt "https://docs.google.com/uc?export=downl
oad&confirm=$(wget --quiet --save-cookies /tmp/cookies.txt --
keep-session-cookies --no-check-
certificate 'https://docs.google.com/uc?export=download&id=1V3vsI
axAlGWvK4Aar9bAiK5U0QFttKwq' -O- | sed -rn 's/.*confirm=([0-9A-
Za-z_]+).*/\1\n/p')&id=1V3vsIaxAlGWvK4Aar9bAiK5U0QFttKwq" -
O yolov4-csp.weights && rm -rf /tmp/cookies.txt
```

Download pre-trained YOLOv4 weights

YOLOv4 has been trained already on the coco dataset which has 80 classes that it can predict. We will grab these pretrained weights so that we can run YOLOv4 on these pretrained classes and get detections.

```
!wget https://github.com/AlexeyAB/darknet/releases/download/darkn
et_yolo_v4_pre/yolov4-tiny.weights
```

Helper Functions

Here are a few helper functions defined that will be used to easily convert between different image types within our later steps.

```
# define helper functions
def imShow(path):
  import cv2
  import matplotlib.pyplot as plt
  %matplotlib inline

  image = cv2.imread(path)
  height, width = image.shape[:2]
  resized_image = cv2.resize(image,(3*width, 3*height), interpola
tion = cv2.INTER_CUBIC)

  fig = plt.gcf()
  fig.set_size_inches(18, 10)
  plt.axis("off")
  plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
  plt.show()

# use this to upload files
def upload():
  from google.colab import files
  uploaded = files.upload()
  for name, data in uploaded.items():
    with open(name, 'wb') as f:
      f.write(data)
      print ('saved file', name)

# use this to download a file
def download(path):
  from google.colab import files
  files.download(path)


# function to convert the JavaScript object into an OpenCV image
def js_to_image(js_reply):
  """
  Params:
          js_reply: JavaScript object containing image from webca
m
  Returns:
          img: OpenCV BGR image
  """
  # decode base64 image
  image_bytes = b64decode(js_reply.split(',')[1])
  # convert bytes to numpy array
  jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)
  # decode numpy array into OpenCV BGR image
  img = cv2.imdecode(jpg_as_np, flags=1)

  return img

# function to convert OpenCV Rectangle bounding box image into ba
se64 byte string to be overlayed on video stream
def bbox_to_bytes(bbox_array):
  """
  Params:
          bbox_array: Numpy array (pixels) containing rectangle t
o overlay on video stream.
  Returns:
        bytes: Base64 image byte string
  """
  # convert array into PIL image
  bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
  iobuf = io.BytesIO()
  # format bbox into png for return
  bbox_PIL.save(iobuf, format='png')
  # format return string
  bbox_bytes = 'data:image/png;base64,{}'.format((str(b64encode(i
obuf.getvalue()), 'utf-8')))

  return bbox_bytes
```

Download pre-trained weights for the convolutional layers.

This step downloads the weights for the convolutional layers of the YOLOv4 network. By using these weights it helps your custom object detector to be way more accurate and not have to train as long. You don't have to use these weights but trust me it will help your modle converge and be accurate way faster. USE IT

```
!wget https://github.com/AlexeyAB/darknet/releases/download/darkn
et_yolo_v4_pre/yolov4-tiny.conv.29
```

```
%cd /content/gdrive/My Drive/yolo/yolov4/darknet
!ls
```

```
%cd darknet/

!chmod +x ./darknet

!./darknet detector train training/obj.data training/yolov4-
tiny.cfg training/yolov4-tiny.conv.29  -dont_show -map
```

Darknet for Python

In order to utilize YOLOv4 with Python code we will use some of the pre-built functions found within darknet.py by importing the functions into our workstation. Feel free to checkout the darknet.py file to see the function definitions in detail!

```
# import darknet functions to perform object detections
from darknet import *
# load in our YOLOv4 architecture network
network, class_names, class_colors = load_network("cfg/yolov4-
csp.cfg", "cfg/coco.data", "yolov4-csp.weights")
width = network_width(network)
height = network_height(network)

# darknet helper function to run detection on image
```

```
def darknet_helper(img, width, height):
  darknet_image = make_image(width, height, 3)
  img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
  img_resized = cv2.resize(img_rgb, (width, height),
                              interpolation=cv2.INTER_LINEAR)

  # get image ratios to convert bounding boxes to proper size
  img_height, img_width, _ = img.shape
  width_ratio = img_width/width
  height_ratio = img_height/height

  # run model on darknet style image to get detections
  copy_image_from_bytes(darknet_image, img_resized.tobytes())
  detections = detect_image(network, class_names, darknet_image)
  free_image(darknet_image)
  return detections, width_ratio, height_ratio
```

YOLOv4 on Webcam Images

Running YOLOv4 on images taken from webcam is fairly straight-forward. We will utilize code within Google Colab's **Code Snippets** that has a variety of useful code functions to perform various tasks.

We will be using the code snippet for **Camera Capture** which runs JavaScript code to utilize your computer's webcam. The code snippet will take a webcam photo, which we will then pass into our YOLOv4 model for object detection.

Below is a function to take the webcam picture using JavaScript and then run YOLOv4 on it.

```
def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({v
ideo: true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();
      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElemen
t.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)

  # get photo data
  data = eval_js('takePhoto({})'.format(quality))
  # get OpenCV format image
  img = js_to_image(data)

  # call our darknet helper on webcam image
  detections, width_ratio, height_ratio = darknet_helper(img, wid
th, height)

  # loop through detections and draw them on webcam image
  for label, confidence, bbox in detections:
    left, top, right, bottom = bbox2points(bbox)
    left, top, right, bottom = int(left * width_ratio), int(top *
 height_ratio), int(right * width_ratio), int(bottom * height_rat
io)
    cv2.rectangle(img, (left, top), (right, bottom), class_colors
[label], 2)
    cv2.putText(img, "{} [{:.2f}]".format(label, float(confidence
)),
                      (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX,
0.5,
                      class_colors[label], 2)
  # save image
  cv2.imwrite(filename, img)

  return filename


try:
  filename = take_photo('photo.jpg')
  print('Saved to {}'.format(filename))

  # Show the image which was just taken.
  display(Image(filename))
except Exception as err:
  # Errors will be thrown if the user does not have a webcam or i
f they do not
  # grant the page permission to access it.
  print(str(err))
```

YOLOv4 on Webcam Videos

Running YOLOv4 on webcam video is a little more complex than images. We need to start a video stream using our webcam as input. Then we run each frame through our YOLOv4 model and create an overlay image that contains bounding box of detection(s). We then overlay the bounding box image back onto the next frame of our video stream.

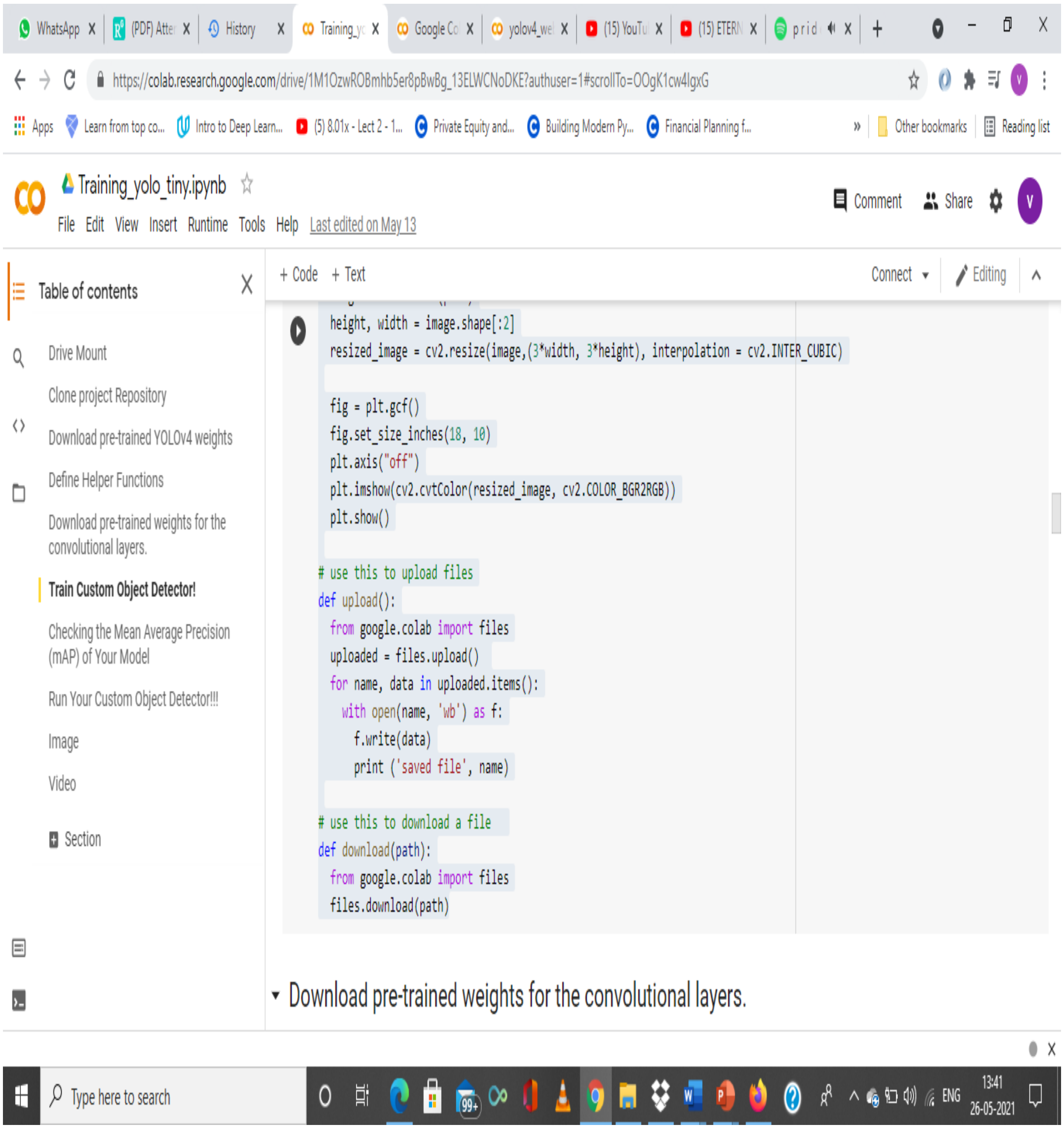
YOLOv4 is so fast that it can run the detections in real-time!

```
def video_stream():

  js = Javascript('''
    var video;
    var div = null;
    var stream;
    var captureCanvas;
    var imgElement;
    var labelElement;

    var pendingResolve = null;
    var shutdown = false;

    function removeDom() {
       stream.getVideoTracks()[0].stop();
       video.remove();
       div.remove();
       video = null;
       div = null;
       stream = null;
       imgElement = null;
       captureCanvas = null;
       labelElement = null;
    }

    function onAnimationFrame() {
      if (!shutdown) {
        window.requestAnimationFrame(onAnimationFrame);
      }
      if (pendingResolve) {
        var result = "";
        if (!shutdown) {
          captureCanvas.getContext('2d').drawImage(video, 0, 0, 6
40, 480);
          result = captureCanvas.toDataURL('image/jpeg', 0.8)
        }
        var lp = pendingResolve;
        pendingResolve = null;
        lp(result);
      }
    }

    async function createDom() {
      if (div !== null) {
        return stream;
      }

      div = document.createElement('div');
      div.style.border = '2px solid black';
      div.style.padding = '3px';
      div.style.width = '100%';
      div.style.maxWidth = '600px';
      document.body.appendChild(div);

      const modelOut = document.createElement('div');
      modelOut.innerHTML = "<span>Status:</span>";
      labelElement = document.createElement('span');
      labelElement.innerText = 'No data';
      labelElement.style.fontWeight = 'bold';
      modelOut.appendChild(labelElement);
      div.appendChild(modelOut);

      video = document.createElement('video');
      video.style.display = 'block';
      video.width = div.clientWidth - 6;
      video.setAttribute('playsinline', '');
      video.onclick = () => { shutdown = true; };
      stream = await navigator.mediaDevices.getUserMedia(
          {video: { facingMode: "environment"}});
      div.appendChild(video);

      imgElement = document.createElement('img');
      imgElement.style.position = 'absolute';
      imgElement.style.zIndex = 1;
      imgElement.onclick = () => { shutdown = true; };
      div.appendChild(imgElement);

      const instruction = document.createElement('div');
      instruction.innerHTML =
          '<span style="color: red; font-weight: bold;">' +
          'When finished, click here or on the video to stop this
demo</span>';
      div.appendChild(instruction);
      instruction.onclick = () => { shutdown = true; };

      video.srcObject = stream;
      await video.play();

      captureCanvas = document.createElement('canvas');
      captureCanvas.width = 640; //video.videoWidth;
      captureCanvas.height = 480; //video.videoHeight;
      window.requestAnimationFrame(onAnimationFrame);

      return stream;
    }
    async function stream_frame(label, imgData) {
      if (shutdown) {
        removeDom();
        shutdown = false;
        return '';
      }

      var preCreate = Date.now();
      stream = await createDom();

      var preShow = Date.now();
      if (label != "") {
        labelElement.innerHTML = label;
      }

      if (imgData != "") {
        var videoRect = video.getClientRects()[0];
        imgElement.style.top = videoRect.top + "px";
        imgElement.style.left = videoRect.left + "px";
        imgElement.style.width = videoRect.width + "px";
        imgElement.style.height = videoRect.height + "px";
        imgElement.src = imgData;
      }

      var preCapture = Date.now();
      var result = await new Promise(function(resolve, reject) {
        pendingResolve = resolve;
      });
      shutdown = false;

      return {'create': preShow - preCreate,
              'show': preCapture - preShow,
              'capture': Date.now() - preCapture,
              'img': result};
    }
    ''')

  display(js)

def video_frame(label, bbox):
  data = eval_js('stream_frame("{}", "{}")'.format(label, bbox))
  return data
```

Running on Webcam Video

```
# start streaming video from webcam
video_stream()
# label for video
label_html = 'Capturing...'
# initialze bounding box to empty
bbox = ''
count = 0
while True:
    js_reply = video_frame(label_html, bbox)
    if not js_reply:
        break

    # convert JS response to OpenCV Image
    frame = js_to_image(js_reply["img"])

    # create transparent overlay for bounding box
    bbox_array = np.zeros([480,640,4], dtype=np.uint8)

    # call our darknet helper on video frame
    detections, width_ratio, height_ratio = darknet_helper(frame,
 width, height)

    # loop through detections and draw them on transparent overla
y image
    for label, confidence, bbox in detections:
      left, top, right, bottom = bbox2points(bbox)
      left, top, right, bottom = int(left * width_ratio), int(top
 * height_ratio), int(right * width_ratio), int(bottom * height_r
atio)
      bbox_array = cv2.rectangle(bbox_array, (left, top), (right,
 bottom), class_colors[label], 2)
      bbox_array = cv2.putText(bbox_array, "{} [{:.2f}]".format(l
abel, float(confidence)),
                        (left, top - 5), cv2.FONT_HERSHEY_SIMPLEX
, 0.5,
                        class_colors[label], 2)

    bbox_array[:,:,3] = (bbox_array.max(axis = 2) > 0 ).astype(in
t) * 255
    # convert overlay of bbox into bytes
    bbox_bytes = bbox_to_bytes(bbox_array)
    # update bbox so next frame gets new overlay
    bbox = bbox_bytes
```

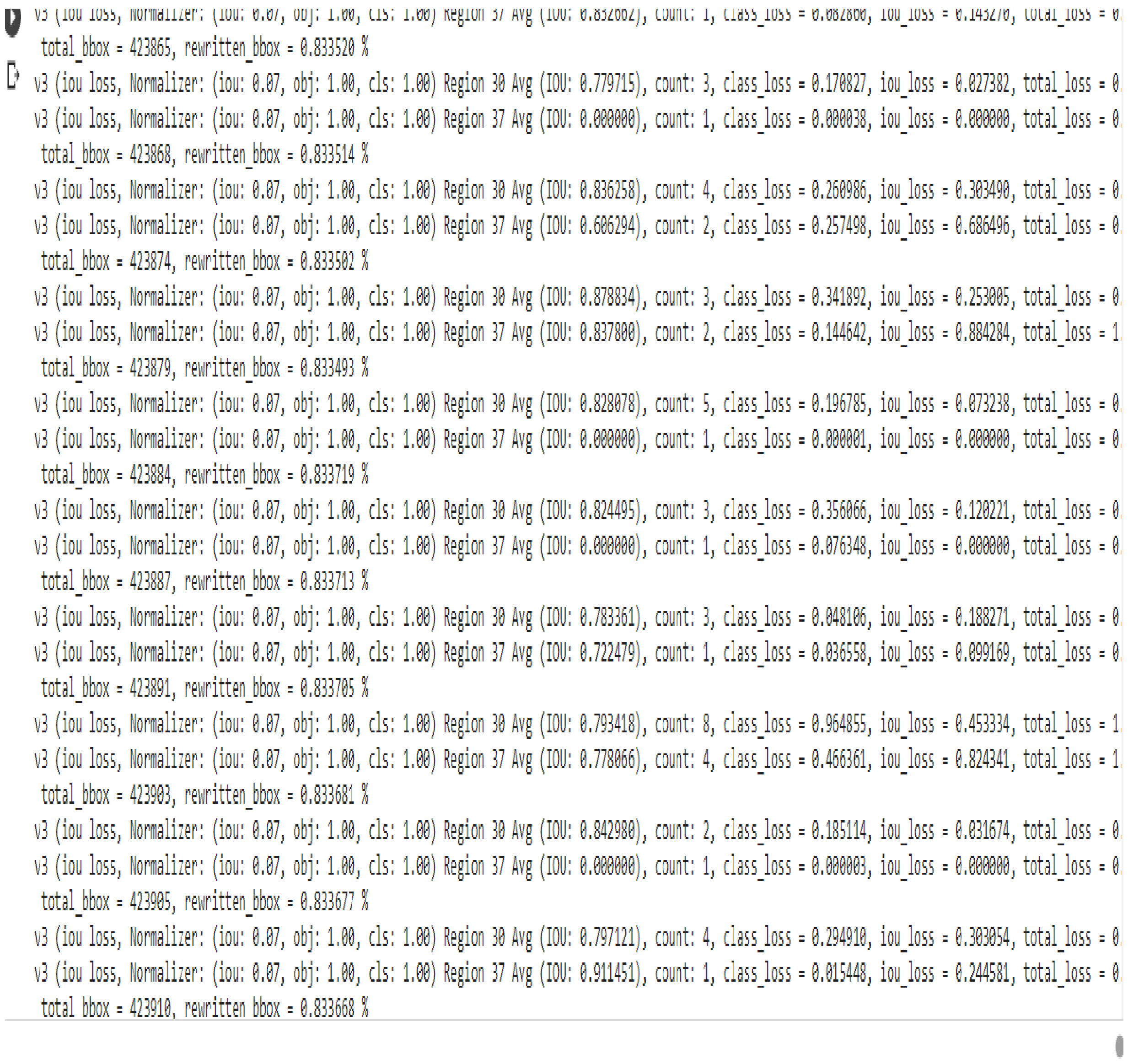
5.2 SCREENSHOTS


Training_yolo_tiny.ipynb
File Edit View Insert Runtime Tools Help Last edited on May 13
Comment Share
+ Code + Text
Connect Editing
Table of contents
Drive Mount
Clone project Repository
Download pre-trained YOLOv4 weights
Define Helper Functions
Download pre-trained weights for the convolutional layers.
Train Custom Object Detector!
Checking the Mean Average Precision (mAP) of Your Model
Run Your Custom Object Detector!!!
Image
Video
Section

```
    height, width = image.shape[:2]
    resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)

    fig = plt.gcf()
    fig.set_size_inches(18, 10)
    plt.axis("off")
    plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
    plt.show()

# use this to upload files
def upload():
    from google.colab import files
    uploaded = files.upload()
    for name, data in uploaded.items():
      with open(name, 'wb') as f:
        f.write(data)
        print ('saved file', name)

# use this to download a file
def download(path):
    from google.colab import files
    files.download(path)
```

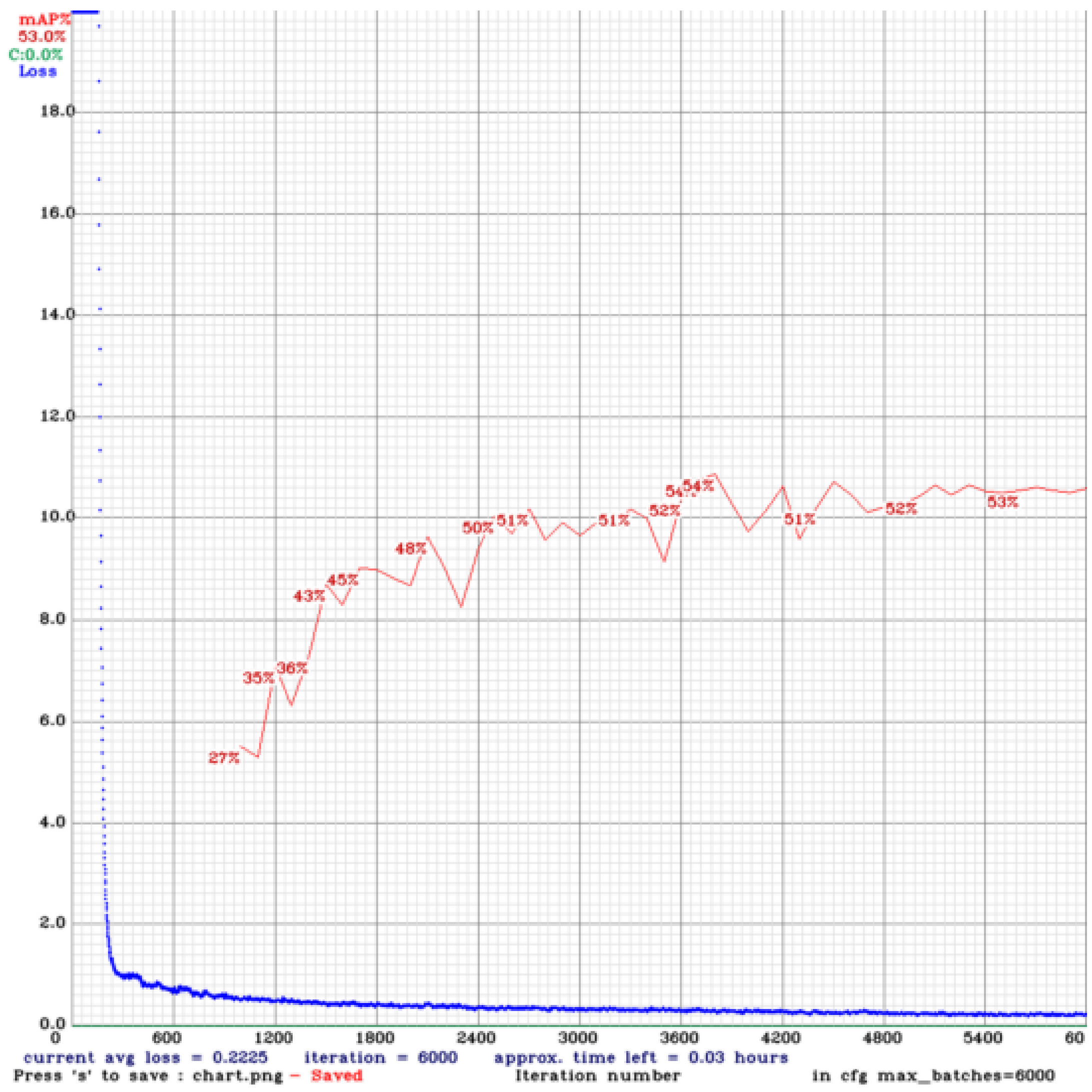
Download pre-trained weights for the convolutional layers.

5.2.1 The colab notebook we worked on



```
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.832662), count: 1, class_loss = 0.082860, iou_loss = 0.143270, total_loss = 0
 total_bbox = 423865, rewritten_bbox = 0.833520 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.779715), count: 3, class_loss = 0.170827, iou_loss = 0.027382, total_loss = 0.
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 0.000038, iou_loss = 0.000000, total_loss = 0.
 total_bbox = 423868, rewritten_bbox = 0.833514 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.836258), count: 4, class_loss = 0.260986, iou_loss = 0.303490, total_loss = 0.
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.606294), count: 2, class_loss = 0.257498, iou_loss = 0.686496, total_loss = 0.
 total_bbox = 423874, rewritten_bbox = 0.833502 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.878834), count: 3, class_loss = 0.341892, iou_loss = 0.253005, total_loss = 0.
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.837800), count: 2, class_loss = 0.144642, iou_loss = 0.884284, total_loss = 1.
 total_bbox = 423879, rewritten_bbox = 0.833493 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.828078), count: 5, class_loss = 0.196785, iou_loss = 0.073238, total_loss = 0.
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 0.000001, iou_loss = 0.000000, total_loss = 0.
 total_bbox = 423884, rewritten_bbox = 0.833719 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.824495), count: 3, class_loss = 0.356066, iou_loss = 0.120221, total_loss = 0.
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 0.076348, iou_loss = 0.000000, total_loss = 0.
 total_bbox = 423887, rewritten_bbox = 0.833713 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.783361), count: 3, class_loss = 0.048106, iou_loss = 0.188271, total_loss = 0.
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.722479), count: 1, class_loss = 0.036558, iou_loss = 0.099169, total_loss = 0.
 total_bbox = 423891, rewritten_bbox = 0.833705 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.793418), count: 8, class_loss = 0.964855, iou_loss = 0.453334, total_loss = 1.
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.778066), count: 4, class_loss = 0.466361, iou_loss = 0.824341, total_loss = 1.
 total_bbox = 423903, rewritten_bbox = 0.833681 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.842980), count: 2, class_loss = 0.185114, iou_loss = 0.031674, total_loss = 0.
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.000000), count: 1, class_loss = 0.000003, iou_loss = 0.000000, total_loss = 0.
 total_bbox = 423905, rewritten_bbox = 0.833677 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 30 Avg (IOU: 0.797121), count: 4, class_loss = 0.294910, iou_loss = 0.303054, total_loss = 0.
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 37 Avg (IOU: 0.911451), count: 1, class_loss = 0.015448, iou_loss = 0.244581, total_loss = 0.
 total_bbox = 423910, rewritten_bbox = 0.833668 %
```

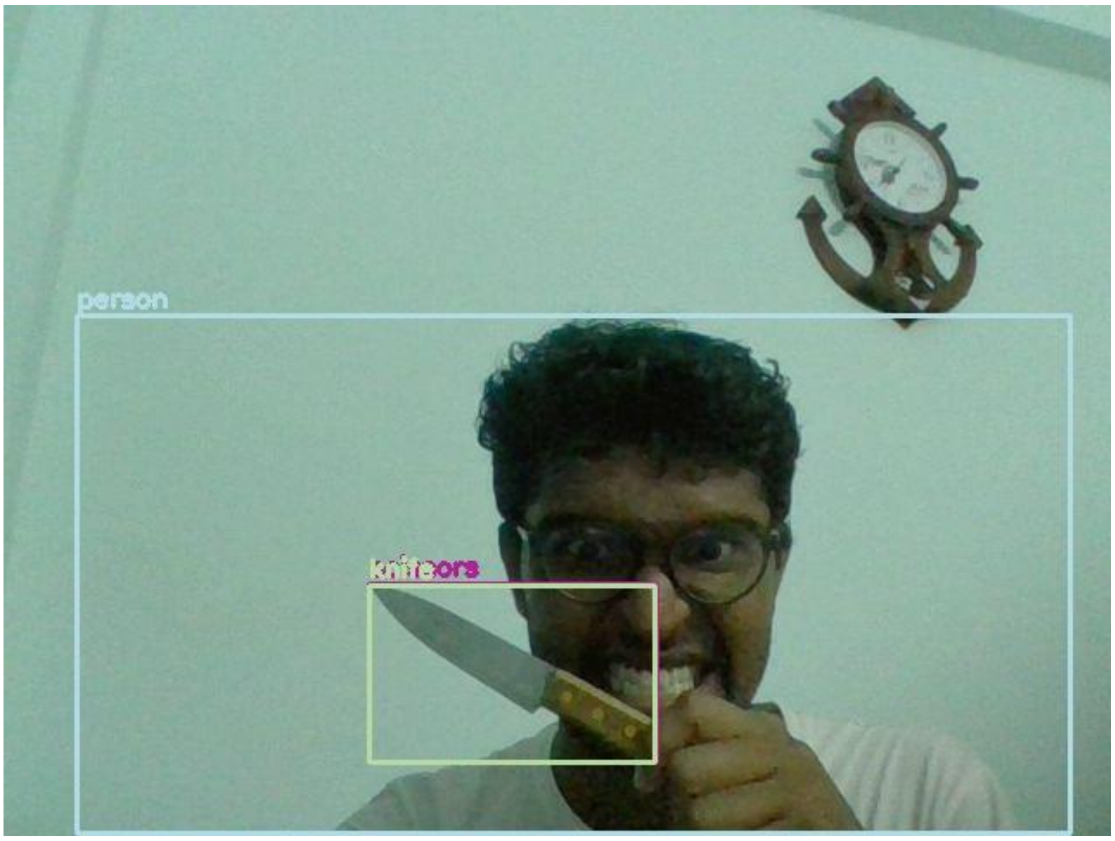
5.2.2 Training



5.2.3 Trained model statistics

5.3 TESTING & TESTCASES

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.



5.3.1 *The system recognising the Anomaly event and displaying on the screen*

6. CONCLUSION & FUTURE SCOPE

6.1 CONCLUSION

- In our project user friendly coding has been adopted.
- This project has given us an opportunity to explore more about Evidence Analysis and Anomaly Detection.
- This project provided us with the pros and cons of many recognition systems and the trade-off associated with them.

6.2 FUTURE SCOPE

This Real time solution can be further extended for applications such as:

- Anomaly detection tasks such as detecting terrorists from group of people.
- Restricted access to restricted places.
- Can be extended for detecting the persons without mask in real-time.

7.REFERENCES

- H. Jeong, K. Park, and Y. Ha, "Image preprocessing for efficient training of yolo deep learning networks," in 2018 IEEE International Conference on Big Data and Smart Computing (BigComp), Jan 2018, pp. 635–637
- S. Saikia, E. Fidalgo, E. Alegre, and L. Fernández-Robles, "Object detection for crime scene evidence analysis using deep learning," " in International Conference on Image Analysis and Processing. Springer, 2017, pp. 14–24
- L. Zeng, J. Chen, L. Tong, B. Yan, and X. Ping, "Image contrast enhancement based on histogram similarity," in 2013 IEEE International Conference on Medical Imaging Physics and Engineering, Oct 2013, pp. 269–273