



GIDC Degree Engineering College

Managed by GIDC Education Society

Gujarat's Public Private Partnership(PPP) Model Institute

Weapon Detection in Images using YOLOv3 & YOLOv4

Submitted By:

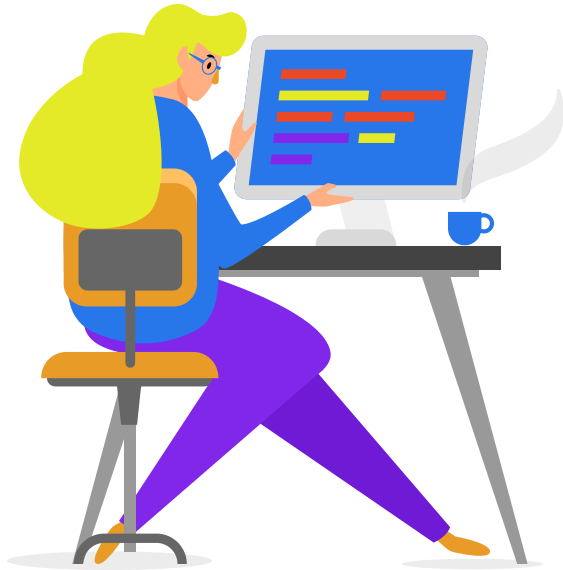
Priya Yadav(201100107524)

Project Guide:

Prof. Archana Naik



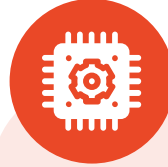
INTRODUCTION



Machine Learning Infographics



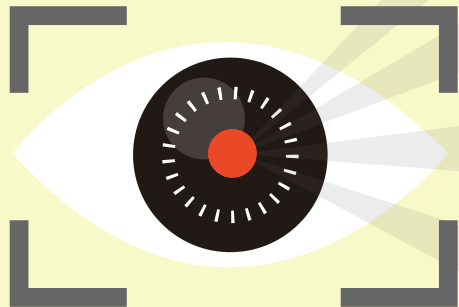
The focus of our mini project is to develop a security system which detects weapons like guns, knife, etc . in no time with less computational time and high accuracy.



It is hoped that through the design and implementation of such system, we can come closer to ensuring safety and security of everyday citizens in both public and private level.

PROJECT SCOPE

Weapon Detection



Label objects

We will label objects of weapon

YOLOv3 architecture

YOLO v3 architecture for our gun detection.

YOLOv4 architecture

YOLO v3 architecture for our knife detection

Improving model performance

We will try to get high test set accuracy.

Testing

We will test the model with real world images

PROJECT SCOPE

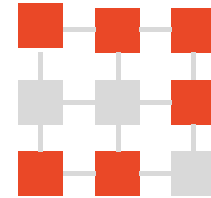
01



Scope of the project is to build a ML model which can detect weapons from captured images and videos.



02



- 1) Acquiring images containing weapons and non-weapons.
- 2) Label the images for weapon detection.
- 3) Use YOLOv3 architecture to validate our dataset to get high accuracy.
- 4) Use YOLOv4 architecture to train and validate our dataset to get high accuracy.

PROJECT REQUIREMENTS

01

Google Colab's

GPU machines to train a ML model

03

ASSUMPTIONS AND CONSTRAINTS

02

SOFTWARE REQUIREMENTS

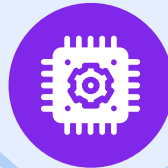
We will use Python and various open source Python based package to train a model

04

DATASET SPECIFICATIONS

We have used [this labeled dataset](#) for detecting weapons in images and videos by using pretrained model

HARDWARE REQUIREMENTS

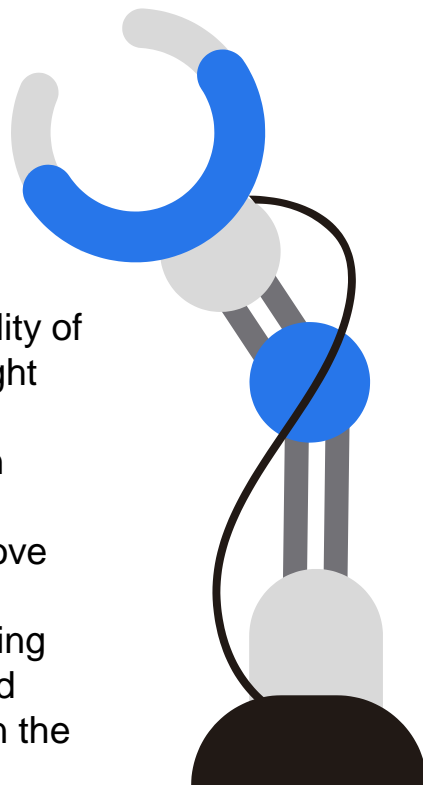


ASSUMPTIONS AND CONSTRAINTS

Weapon Detection



- ☐ The gun is in line of sight of camera and fully/partially exposed to the camera.
- ☐ There is enough background light to detect the ammunition.
- ☐ Generally the model accuracy depends on quality of training data, in real life scenario the model might fail on different lights settings.
- ☐ We will target accuracy of a model greater than 85% but real time accuracy might increase or decrease based on training data. We can improve such model by additional training.
- ☐ This project requires high performance computing power to train the model, So GPU with high-end computation power were used, to remove lag in the ammunition detection.



LITERATURE REVIEW

01

**Anuj Mahajan and
Simran Gupta**

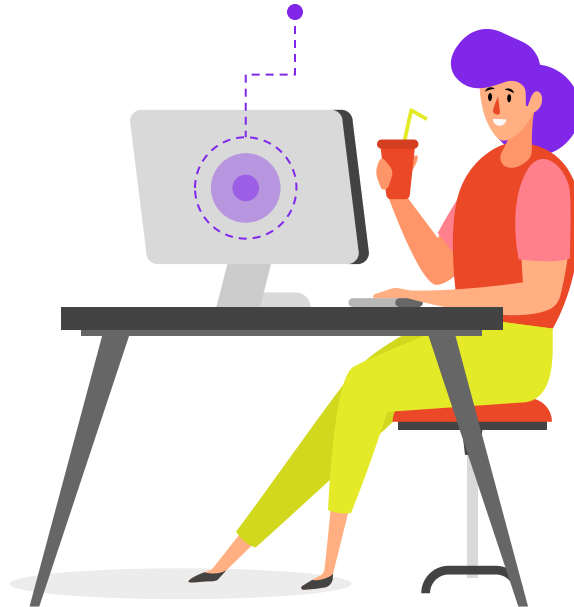
“Weapon Detection in Video
Surveillance using Computer
Vision Techniques”,

03

**Dr. N Geetha, Akash
Kumar, K.S. Akshita**

“Weapon Detection in
Surveillance System”

ML



**Michal Grega, Andrzej
Matiolanski, Piotr Guzik
and Mikolaj Leszczuk**

“Automated Detection of
Firearms and Knives in a
CCTV Image”,

05

**Tahir Bhatti, Gufran Khan
and Masood Aslam**

“Weapon Detection in Real
Time CCTV Videos using
Deep Learning”

04

PROJECT FEASIBILITY STUDY

Our mini project is Technical feasible because the technologies needed in our project are readily available in market.

We have opted Google Colab for storage and good processing speed.

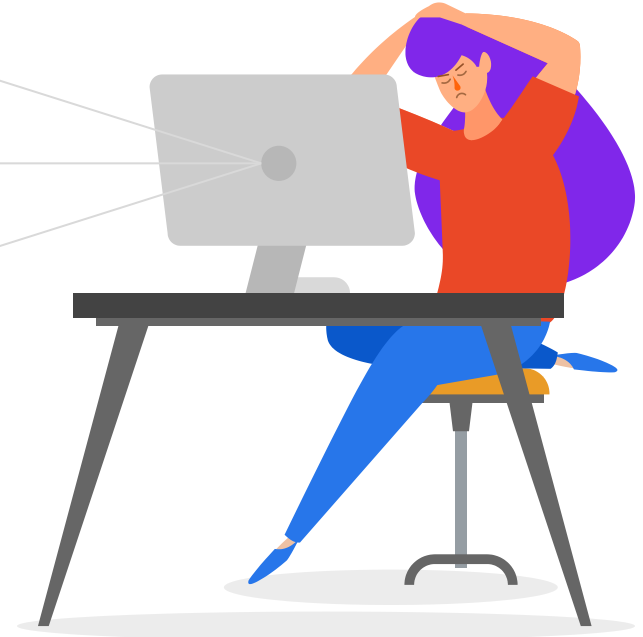
We have used Python and its libraries like Jupyter Notebook to run code

01

02

03

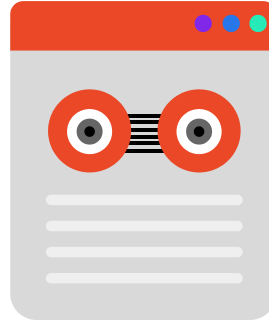
Technical Feasibility



Economic Feasibility

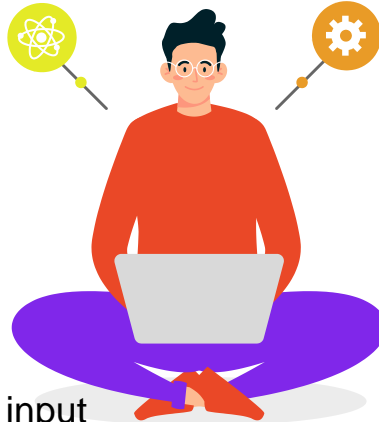
Our project is economically feasible because the system is developed by the team of 2 people

The development tools used are free and open source, hence the cost of development only accounts for electricity bills and internet bills for laptops which is very minimum



Operational Feasibility

Our system will take pictures or video as input and detect weapons using ML algorithms



Operational Feasibility

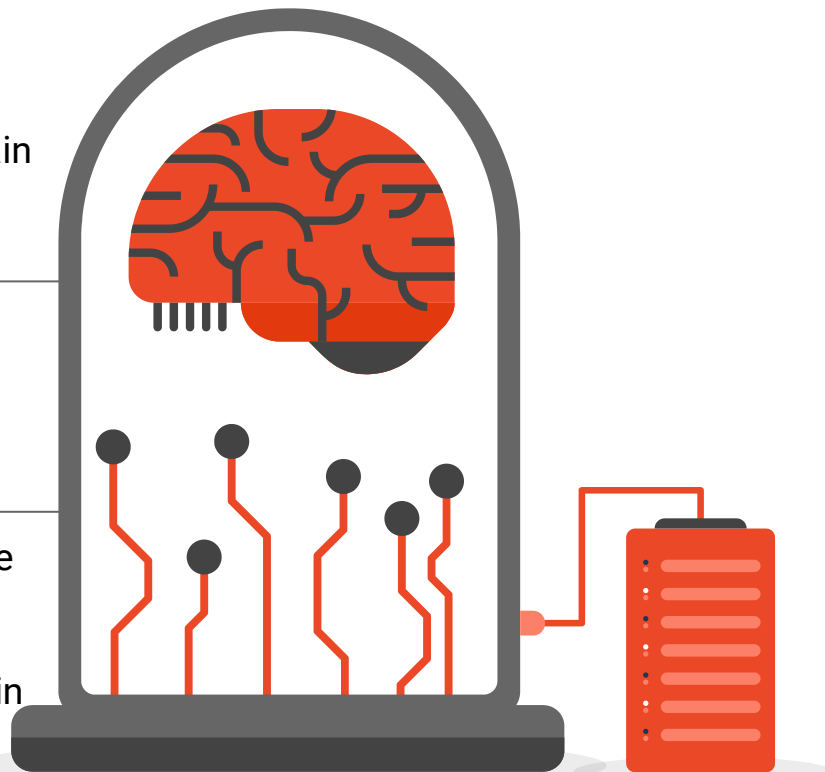
The system will also classify the type of weapons

PROJECT REQUIREMENT GATHERING

01 We need to work with various stockholders in both private and public services to understand the pain Points

02 There is a long challenge in adoption of new technology and privacy, we will work with stockholders to overcome the same.

03 To get this project working, we might need some real time setting like Low visibility areas, bad camera positions, bad resolution etc. We will try and procure additional such images to be used in training data



PROJECT SRS

01

**USE CASE
DIAGRAM**

02

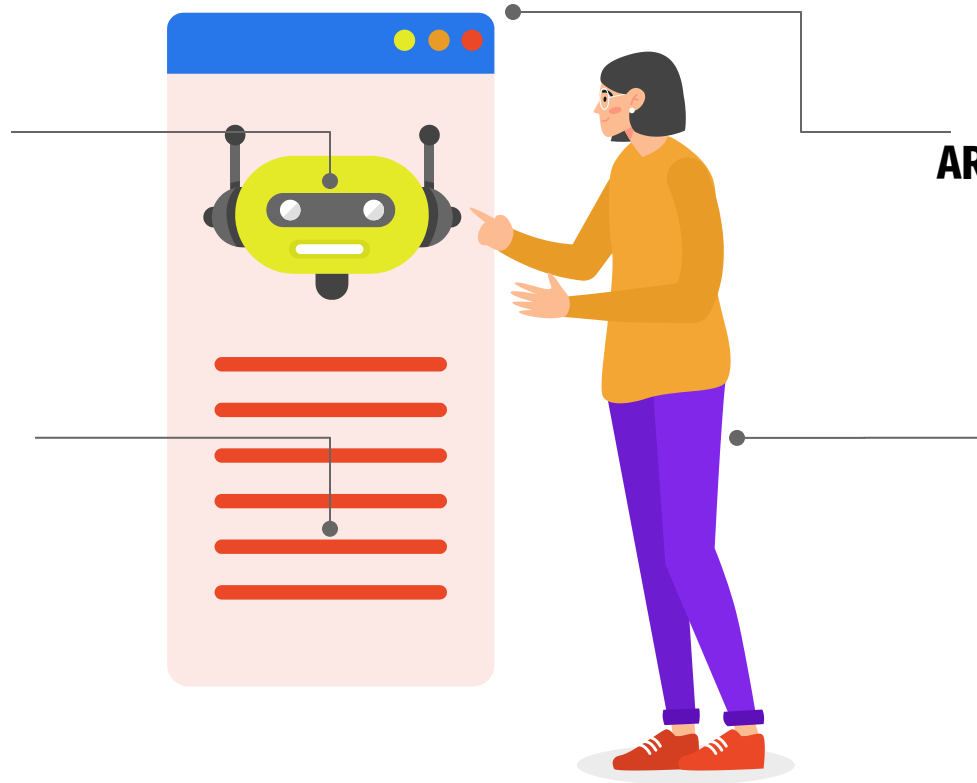
**YOLOv4
ARCHITECTURE**

03

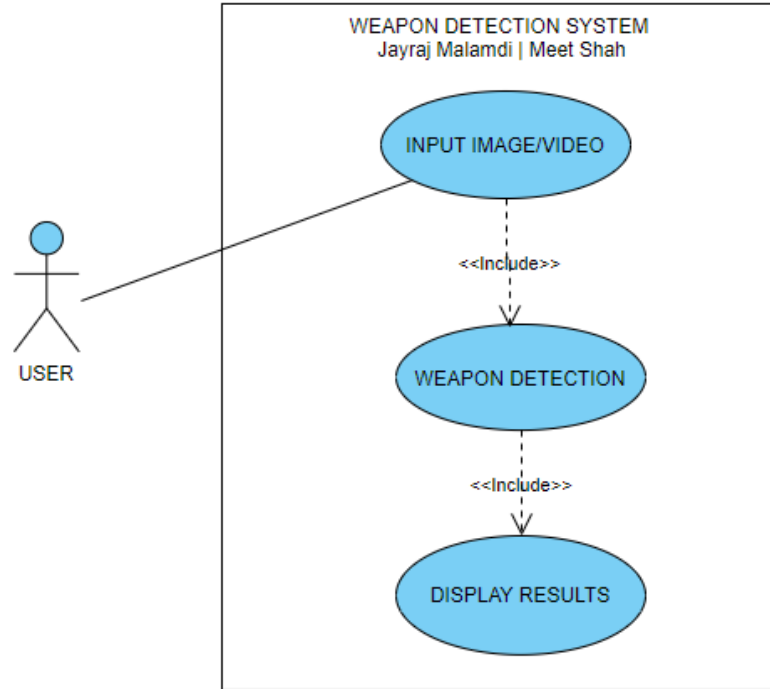
**YOLOv3
ARCHITECTURE**

04

**MODEL
TRAINING
FLOW**



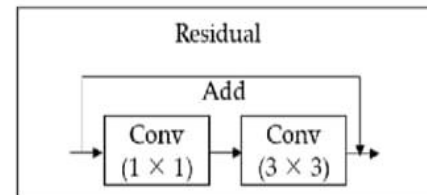
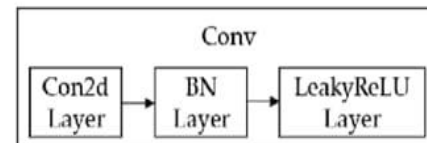
USE CASE DIAGRAM



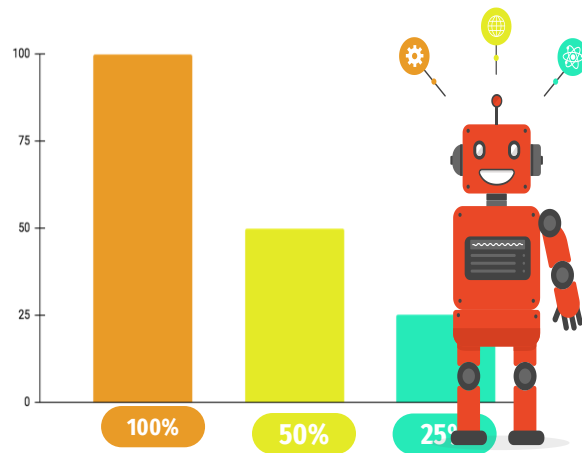
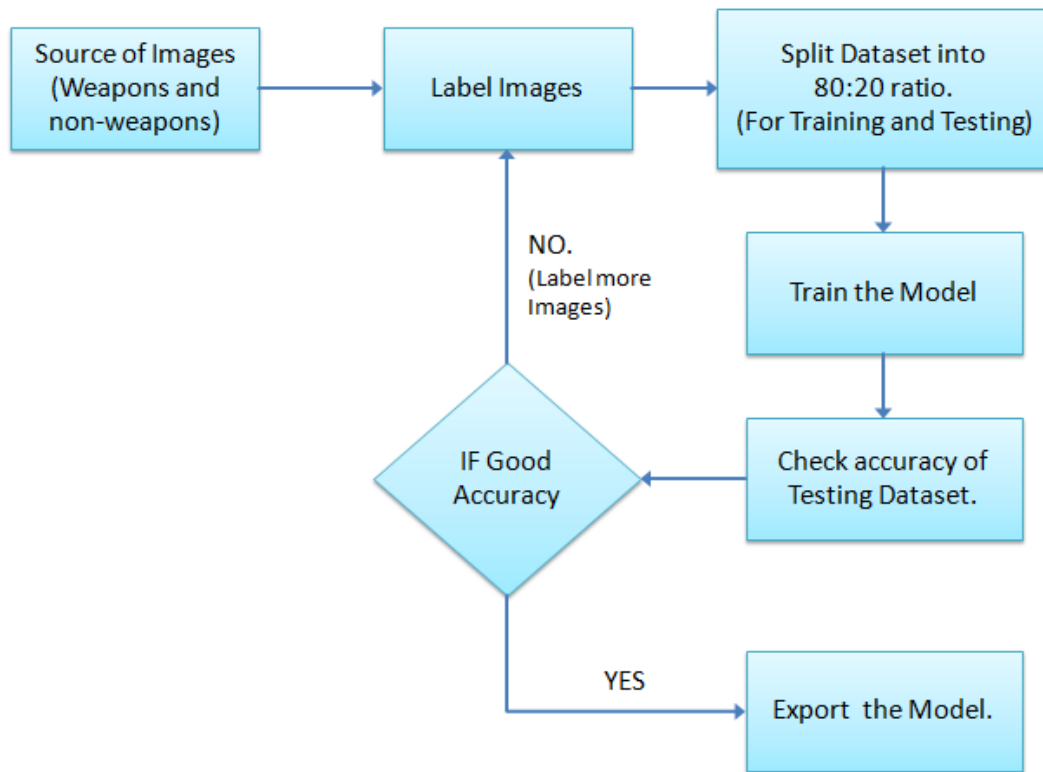
YOLOv3 ARCHITECTURE



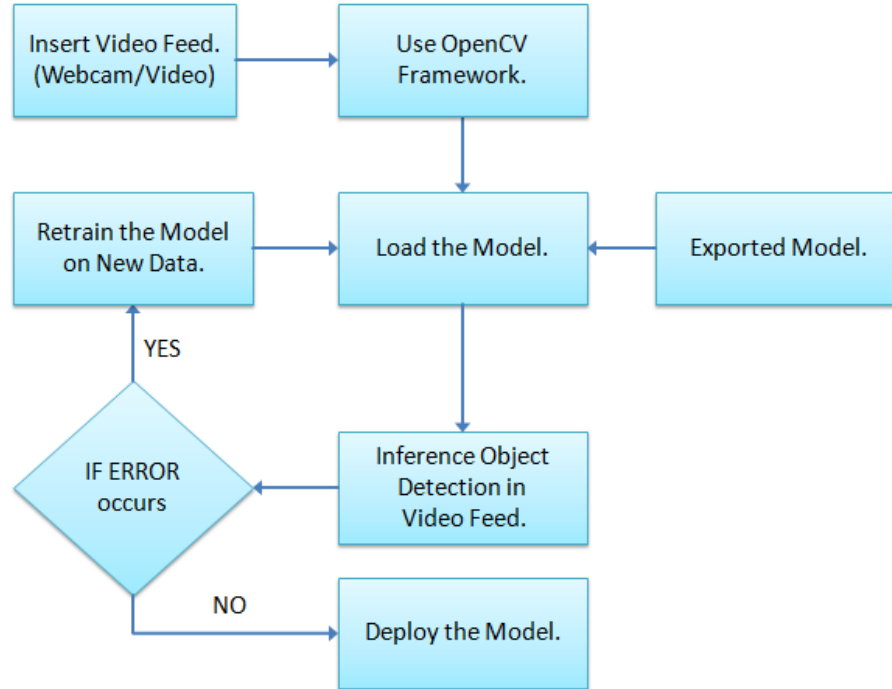
Layer	Filters size	Repeat	Output size
Image			416×416
Conv	$32 \ 3 \times 3/1$	1	416×416
Conv	$64 \ 3 \times 3/2$	1	208×208
Conv	$32 \ 1 \times 1/1$	<div> <div>Conv</div> <div>Conv</div> <div>Residual</div> </div> $\times 1$	208×208
Conv	$64 \ 3 \times 3/1$		208×208
Residual			208×208
Conv	$128 \ 3 \times 3/2$	1	104×104
Conv	$64 \ 1 \times 1/1$	<div> <div>Conv</div> <div>Conv</div> <div>Residual</div> </div> $\times 2$	104×104
Conv	$128 \ 3 \times 3/1$		104×104
Residual			104×104
Conv	$256 \ 3 \times 3/2$	1	52×52
Conv	$128 \ 1 \times 1/1$	<div> <div>Conv</div> <div>Conv</div> <div>Residual</div> </div> $\times 8$	52×52
Conv	$256 \ 3 \times 3/1$		52×52
Residual			52×52
Conv	$512 \ 3 \times 3/2$	1	26×26
Conv	$256 \ 1 \times 1/1$	<div> <div>Conv</div> <div>Conv</div> <div>Residual</div> </div> $\times 8$	26×26
Conv	$512 \ 3 \times 3/1$		26×26
Residual			26×26
Conv	$1024 \ 3 \times 3/2$	1	13×13
Conv	$512 \ 1 \times 1/1$	<div> <div>Conv</div> <div>Conv</div> <div>Residual</div> </div> $\times 4$	13×13
Conv	$1024 \ 3 \times 3/1$		13×13
Residual			13×13



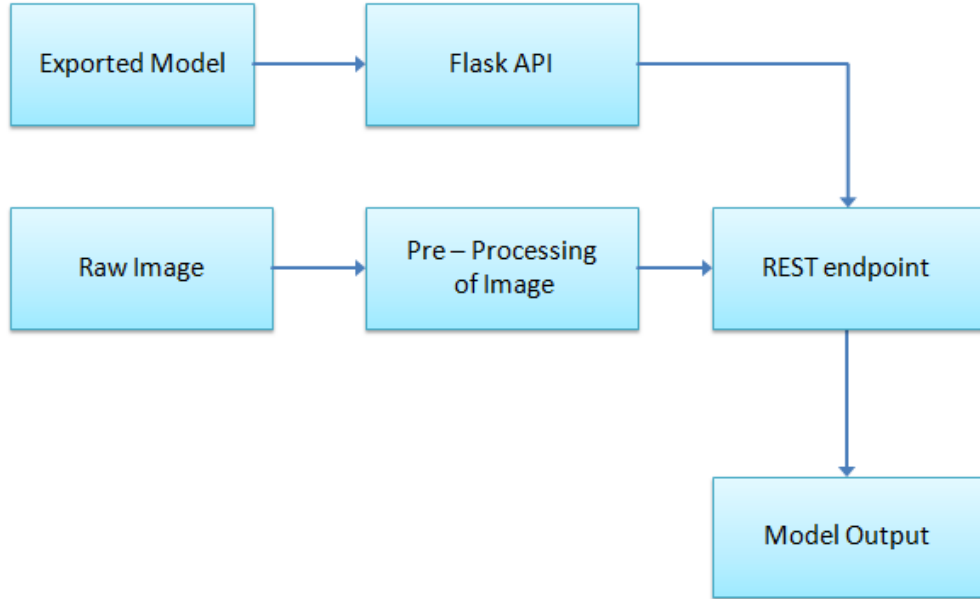
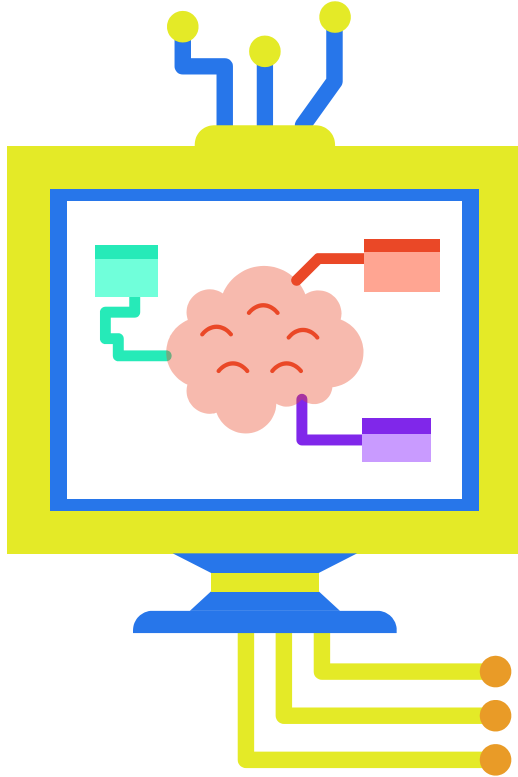
MODEL TRAINING FLOW



MODEL DEPLOYMENT



MODEL VALIDATION ON IMAGES



PROJECT IMPLEMENTATION FOR GUN DETECTION

Importing Libraries

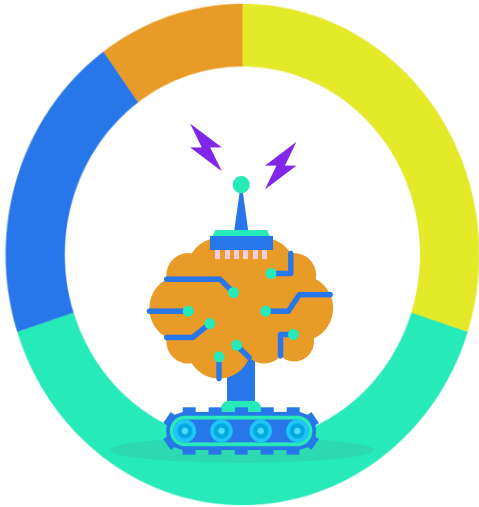
```
In [19]: # import libraries

import numpy as np
import time
import cv2
import os
import matplotlib.pyplot as plt
%matplotlib inline
```

Matplotlib for plotting

```
: # using matplotlib here for plotting and show image in output  
  
def display_img(img,cmap=None):  
    fig = plt.figure(figsize = (12,12))  
    plt.axis(False)  
    ax = fig.add_subplot(111)  
    ax.imshow(img,cmap)
```

Load YOLOv3 architecture & pre-trained weights



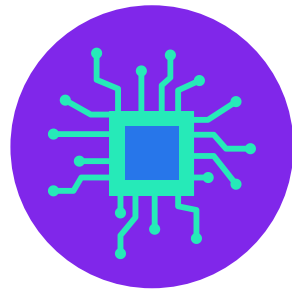
```
labelsPath = (r"C:\Users\DELL\weapon detection inference\yolo_gun.names")
LABELS = open(labelsPath).read().strip().split("\n")
print("----Label to predict---", LABELS)
```

```
----Label to predict--- ['gun']
```

```
weightsPath = os.path.join(r"C:\Users\DELL\weapon detection inference\yolov3_900.weights")
configPath = os.path.join(r"C:\Users\DELL\weapon detection inference\yolov3_custom_train.cfg.txt")
```

```
# Loading the neural network framework Darknet (YOLO was created based on this framework)
net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
```

Detecting & Creating Bounding Box



```
# Create the function which predict the frame input
```

```
def predict(image):
```

```
# initialize a list of colors to represent each possible class label
```

```
np.random.seed(42)
```

```
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3), dtype="uint8")
```

```
(H, W) = image.shape[:2]
```

```
# determine only the "ouput" layers name which we need from YOLO
```

```
ln = net.getLayerNames()
```

```
ln = [ln[i - 1] for i in net.getUnconnectedOutLayers()]
```

```
# construct a blob from the input image and then perform a forward pass of the YOLO object detector  
# giving us our bounding boxes and associated probabilities
```

```
blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416), swapRB=True, crop=False)
```

```
net.setInput(blob)
```

```
layerOutputs = net.forward(ln)
```

```
boxes = []
```

```
confidences = []
```

```
classIDs = []
```

```
threshold = 0.2
```

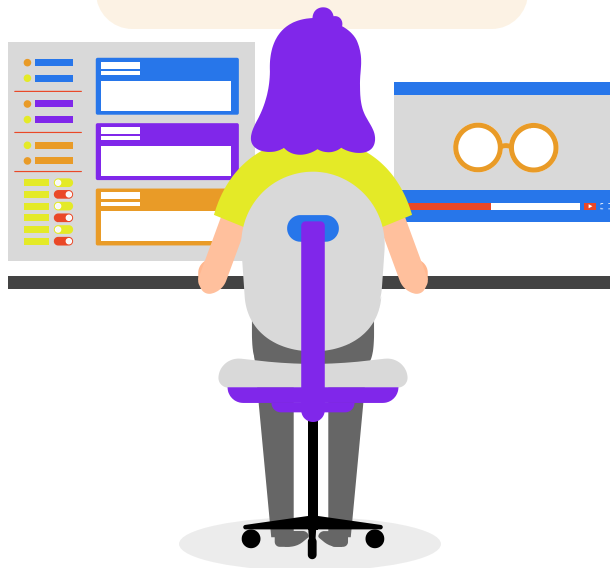
output

02

**REAL TIME CCTV
IMAGE**

01

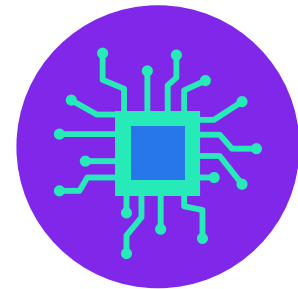
HD IMAGE



03

WEAKNESSES

HD IMAGE



```
In [29]: # Execute prediction on a single image
img = cv2.imread(r"C:\Users\DELL\weapon detection inference/sample images/sample3.jpg")
#img = cv2.resize(img, (416, 416))
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
display_img(predict(img))
```



REAL TIME CCTV IMAGE

```
In [28]: # Execute prediction on a single image
img = cv2.imread(r"C:\Users\DELL\weapon detection inference/cctv_sample1.jpg")
#img = cv2.resize(img, (416, 416))
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
display_img(predict(img))
```

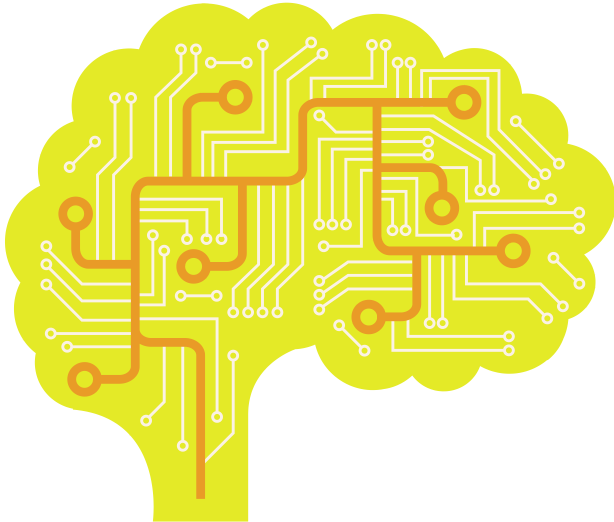


WEAKNESSES

First Weakness



```
In [9]: # Execute prediction on a single image
img = cv2.imread(r"C:\Users\DELL\weapon detection inference/sample images/sample5.jpg")
#img = cv2.resize(img, (416, 416))
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
display_img(predict(img))
```



WEAKNESSES

Second Weakness

```
In [30]: # Execute prediction on a single image
img = cv2.imread(r"C:\Users\DELL\weapon detection inference/cctv_sample2.jpg")
#img = cv2.resize(img, (416, 416))
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
display_img(predict(img))
```



WEAKNESSES

Third Weakness

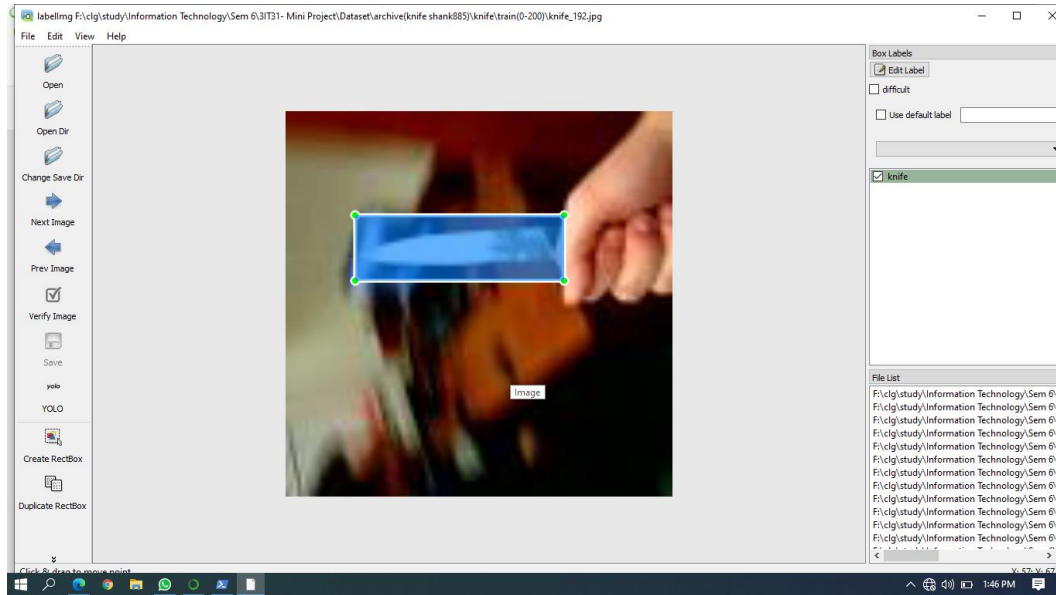
```
In [10]: # Execute prediction on a single image
img = cv2.imread(r"C:\Users\DELL\weapon detection inference/sample images/sample6.jpg")
#img = cv2.resize(img, (416, 416))
img = cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
display_img(predict(img))
```



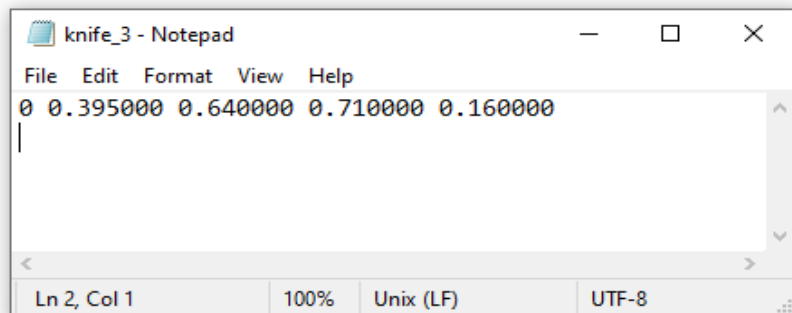


PROJECT IMPLEMENTATION FOR KNIFE DETECTION

Label The Knife Dataset using LabelImg Tool

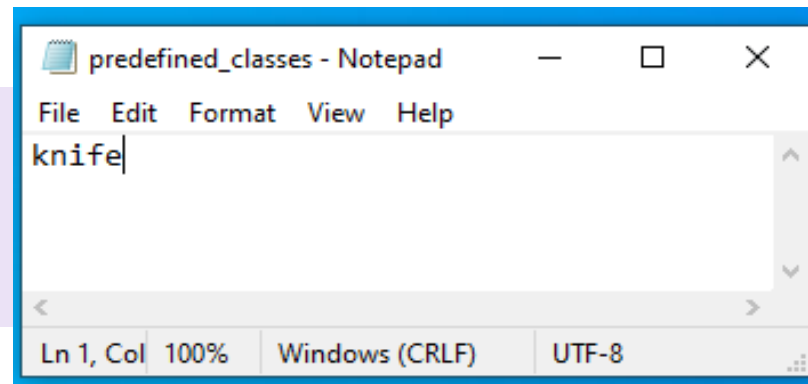


**Coordinates of the bound
Image**



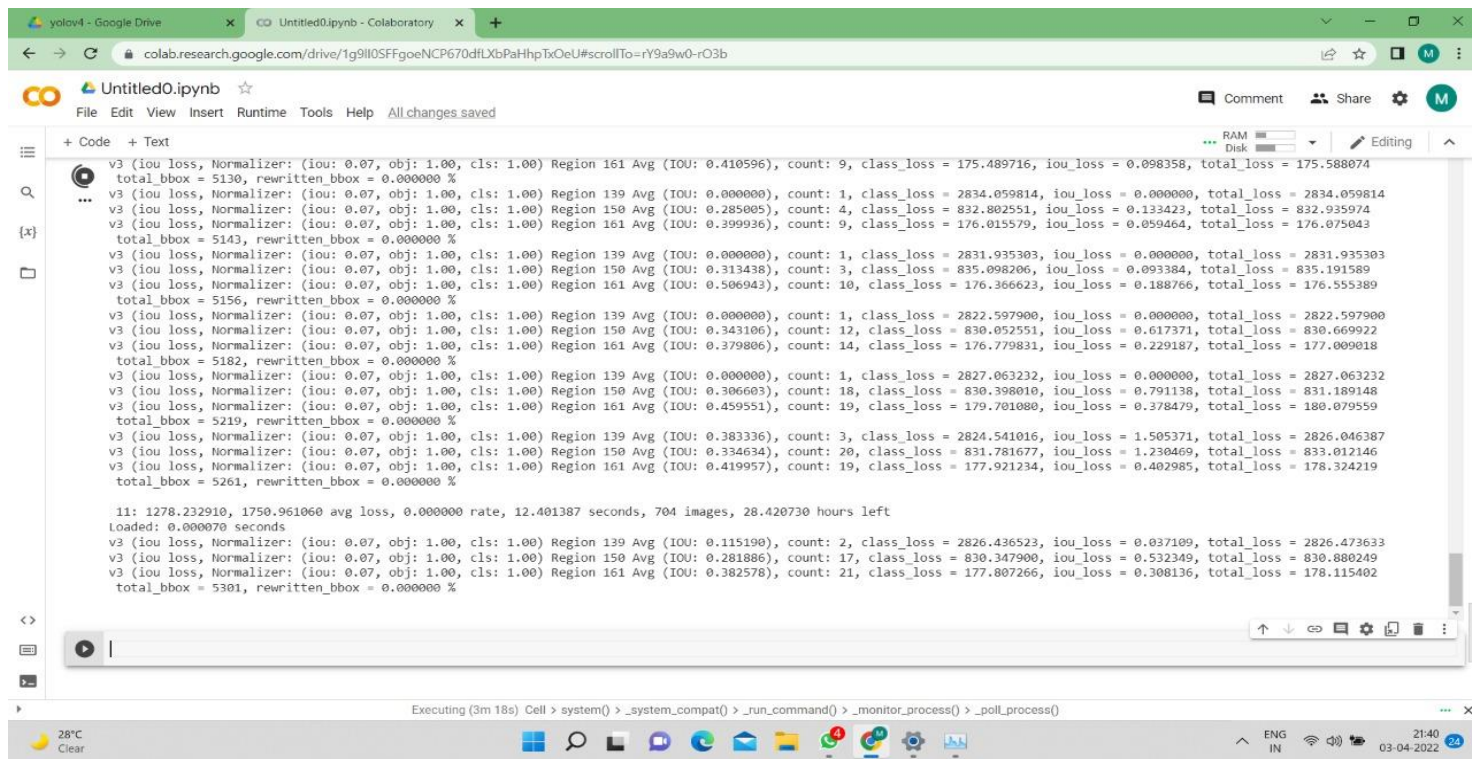
```
knife_3 - Notepad
File Edit Format View Help
0 0.395000 0.640000 0.710000 0.160000
Ln 2, Col 1 100% Unix (LF) UTF-8
```

**The classes used
while labeling**



```
predefined_classes - Notepad
File Edit Format View Help
knife
Ln 1, Col 100% Windows (CRLF) UTF-8
```

Train the Model on Google Colab using YOLOv4 architecture.



```
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.410596), count: 9, class_loss = 175.489716, iou_loss = 0.098358, total_loss = 175.588074
total_bbox = 5130, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.000000), count: 1, class_loss = 2834.059814, iou_loss = 0.000000, total_loss = 2834.059814
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.285005), count: 4, class_loss = 832.802551, iou_loss = 0.133423, total_loss = 832.935974
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.399936), count: 9, class_loss = 176.015579, iou_loss = 0.059464, total_loss = 176.075043
total_bbox = 5143, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.000000), count: 1, class_loss = 2831.935303, iou_loss = 0.000000, total_loss = 2831.935303
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.313438), count: 3, class_loss = 835.098206, iou_loss = 0.093384, total_loss = 835.191589
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.506043), count: 10, class_loss = 176.366623, iou_loss = 0.188766, total_loss = 176.555389
total_bbox = 5156, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.000000), count: 1, class_loss = 2822.597900, iou_loss = 0.000000, total_loss = 2822.597900
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.343106), count: 12, class_loss = 830.052551, iou_loss = 0.617371, total_loss = 830.669922
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.379806), count: 14, class_loss = 176.779831, iou_loss = 0.229187, total_loss = 177.009018
total_bbox = 5182, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.000000), count: 1, class_loss = 2827.063232, iou_loss = 0.000000, total_loss = 2827.063232
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.306603), count: 18, class_loss = 830.398010, iou_loss = 0.791138, total_loss = 831.189148
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.459551), count: 19, class_loss = 179.701080, iou_loss = 0.378479, total_loss = 180.079559
total_bbox = 5219, rewritten_bbox = 0.000000 %
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.383336), count: 3, class_loss = 2824.541016, iou_loss = 1.505371, total_loss = 2826.046387
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.334634), count: 20, class_loss = 831.781677, iou_loss = 1.230469, total_loss = 833.012146
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.419957), count: 19, class_loss = 177.921234, iou_loss = 0.402985, total_loss = 178.324219
total_bbox = 5261, rewritten_bbox = 0.000000 %

11: 1278.232910, 1750.961060 avg loss, 0.000000 rate, 12.401387 seconds, 704 images, 28.420730 hours left
Loaded: 0.000070 seconds
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 139 Avg (IOU: 0.115190), count: 2, class_loss = 2826.436523, iou_loss = 0.037109, total_loss = 2826.473633
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 150 Avg (IOU: 0.281886), count: 17, class_loss = 830.347900, iou_loss = 0.532349, total_loss = 830.880249
v3 (iou loss, Normalizer: (iou: 0.07, obj: 1.00, cls: 1.00) Region 161 Avg (IOU: 0.382578), count: 21, class_loss = 177.807266, iou_loss = 0.308136, total_loss = 178.115402
total_bbox = 5301, rewritten_bbox = 0.000000 %
```


Thank You 😊