List Of Figures

List Of Tables

Abbreviations

CNN	Convolutional Neural Network
CV	Computer Vision
YOLO	You only look once

Notations

No specific notation is used

Abstract

Segregating the waste material over the garbage bin even though the bin is filled is becoming a very big problem. Even the cleaning authorities won't be knowing the level of garbage bin filled and when there are limited people for the duty then it will be difficult for the authority people to check the bin regularly. So, if the garbage is not removed for a long period of time, it will raise a lot of issues like stench, harmful gas and unhygienic condition.

In order to solve the problem a camera-based detection system can be built using a convolutional neural network (CNN) Architecture is proposed with a tracking mechanism to detect the overflowing garbage bin and to report them so that some action can be taken to remove the garbage.

The main requirement for the project is collection of the dataset. Once the dataset is collected, the dataset should be divided in to train and test pair and annotation every training image should be manually. For the object detection, YOLOv5 is the state of art architecture and with the YOLOv5 we can get the weights used on the training images to detect on the testing images.

Keywords:

CNN, YOLOv5, Detection, Garbage.

Table of Content

Chapter 1

Summary of the Base Paper

Title: "Multi-Scale CNN Based Garbage Detection of Airborne Hyperspectral Data"

Publisher: IEEE

Published Date: 30 July 2019

Dumping of garbage is becoming a problem to the environment and it's increasing more every year in an exponential rate. Finding the dumped area manually will consume a lot of time and even there is a possibility that some place may go unnoticed. So, for this problem an autonomous detection system is proposed and there is no public data available so they have used hyperspectral garbage dataset from Shandong Suburb dataset which is labelled and published. They have used Multi-Scale Convolutional Neural Network to classify HIS data and using the model they generated a binary segmentation map. To locate the place, they have used unsupervised region proposal generation algorithm and none maximum suppression. They used the Indian pines dataset and Pavia University dataset for the testing purpose.

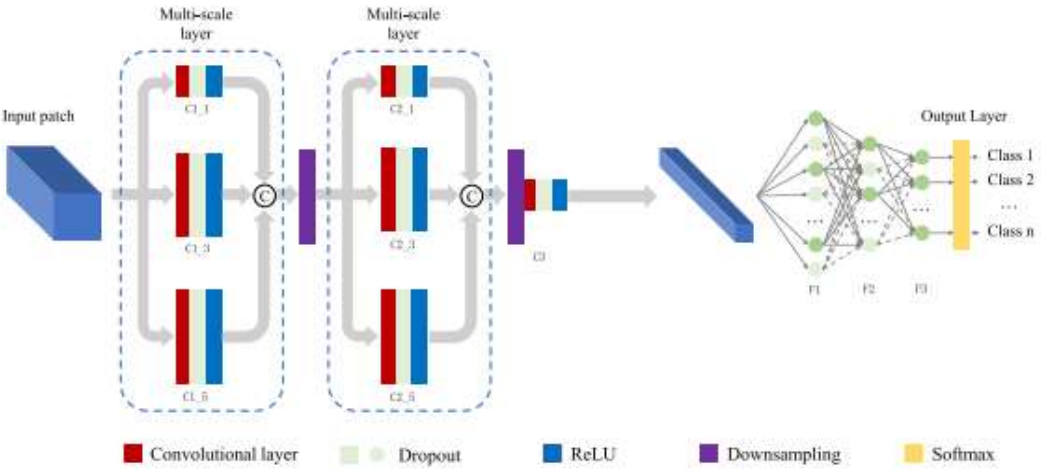

Fig 1.1. Architecture used in the model

Chapter 2

Merits and Demerits of the base paper

2.1 Merits

In this paper they have used a novel Multi layered CNN which was crated just for this application and the dataset works well for the dataset that they have used. Their location searching algorithms works very well to search for a garbage dumped location.

2.2 Demerits

This paper uses the hyperspectral data set which is difficult to implement for a small-scale area. Using an airborne image dataset would be much more efficient for the application purpose. The architecture they proposed consumes a lot of computational power.

Chapter 3

Project Phases

Dataset

The first phase of the project to detect the desired result is to collect a suitable dataset.

Dataset used for the project: Kaggle Garbage Dataset

Citation:" *G Mittal, K B Yagnik, M Garg, and N C Krishnan, Spot Garbage: Smartphone App to Detect Garbage Using Deep Learning, ACM International Joint Conference on Pervasive and Ubiquitous Computing, 940-945, 2016*"

Annotation

LabelImg package is used for annotation of the image.

Package: "https://github.com/tzutalin/labelImg"

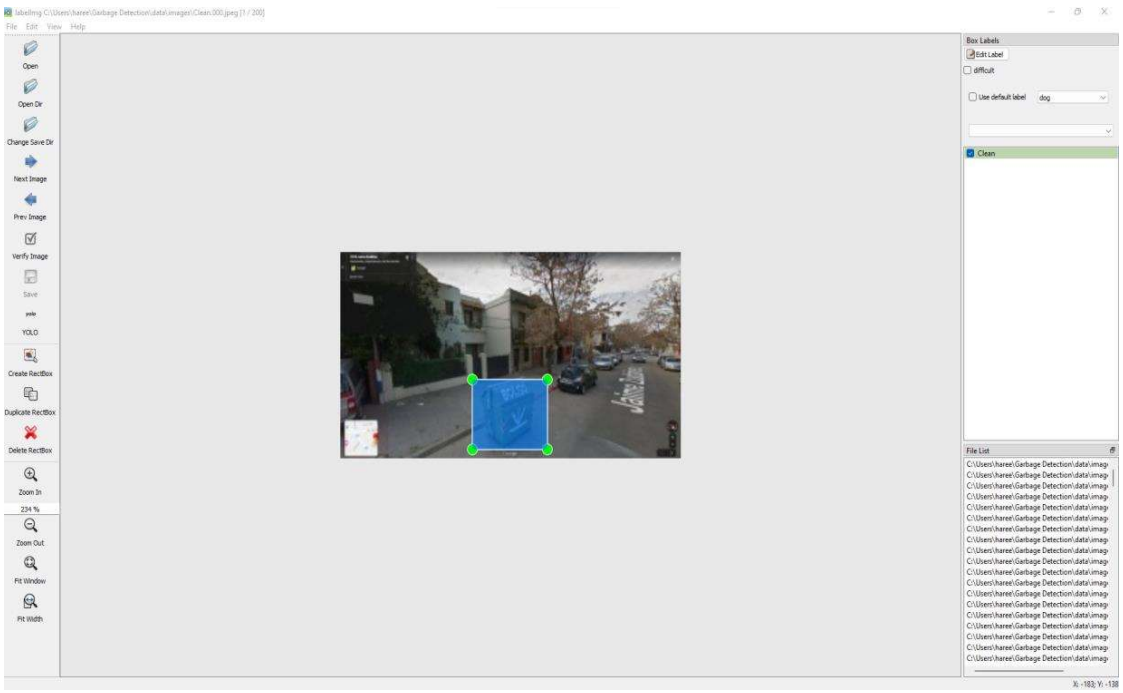

Fig 3.1. Annotation of the image

Modelling

YOLOv5 is a object detection architecture and it is pretrained on the COCO dataset. There are five levels of dense on the model (n, s, m, l, and x)

Link: " https://github.com/ultralytics/yolov5"

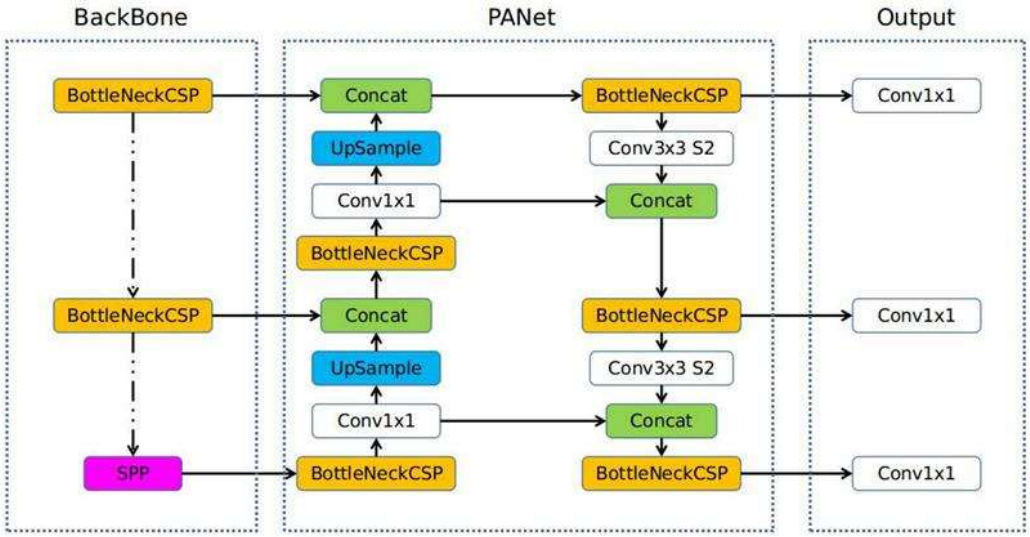

Fig 3.2. YOLOv5 Architecture

Model Comparison

Comparing the model result time.

Models	Performance time per frame
YOLOv5n	63ms
YOLOv5s	82ms
YOLOv5m	113ms
YOLOv5l	147ms
YOLOv5x	184ms

Table 3.1. Performance Comparison

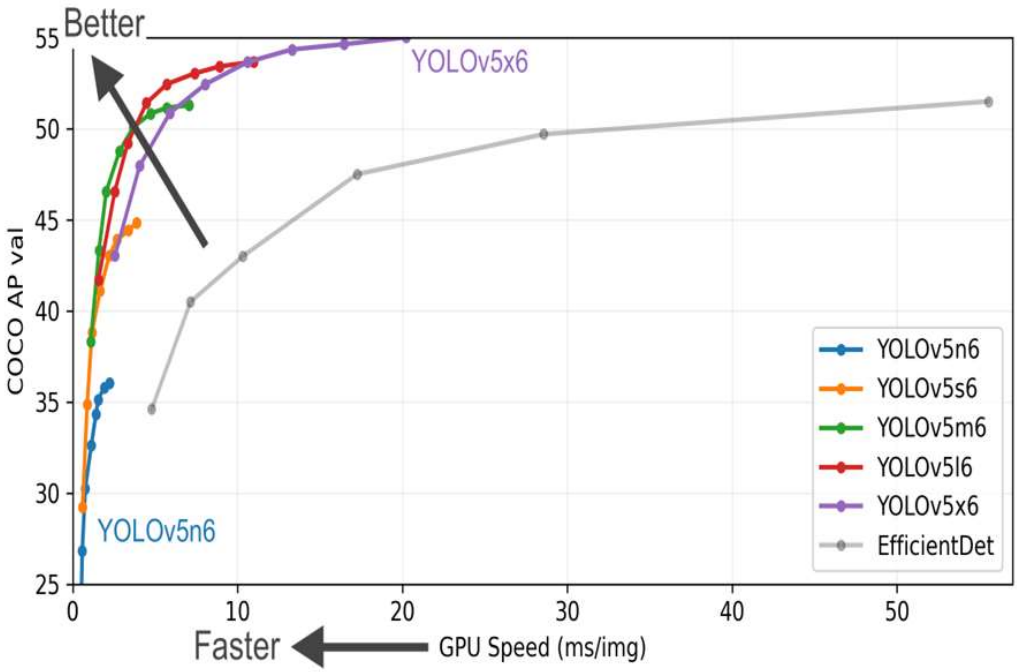

Fig 3.3. Graph for comparison

Performance metric

The following metrics are used to measure the performance of the model.

- Precision
- Recall score
- Mean average precision
- Box loss
- Object loss
- Class loss
- Information retrieval

epoch	train/box_loss	train/obj_loss	train/cls_loss	metrics/precision	metrics/recall	metrics/mAP_0.5	metrics/mAP_0.5:0.95	val/box_loss	val/obj_loss	val/cls_loss	x/lr0	x/lr1	x/lr2
0	0.078962	0.048293	0.056422	0.08359	0.21943	0.078534	0.014545	0.046939	0.031228	0.039594	0.00325	0.00325	0.07075
1	0.062648	0.030181	0.040788	0.63012	0.255	0.13275	0.023629	0.051691	0.015299	0.019797	0.0065182	0.0065182	0.040685
2	0.064021	0.018864	0.027455	0.067194	0.1783	0.05694	0.012566	0.049208	0.0097024	0.015808	0.0097203	0.0097203	0.010554
3	0.063424	0.016517	0.021434	0.10426	0.20434	0.067353	0.014403	0.053946	0.0089501	0.017711	0.009703	0.009703	0.009703
4	0.062451	0.016199	0.016785	0.27883	0.32	0.23508	0.044726	0.054381	0.0089832	0.0096862	0.009703	0.009703	0.009703
5	0.057855	0.015923	0.015656	0.84939	0.32478	0.33533	0.10778	0.049042	0.0093523	0.010038	0.009604	0.009604	0.009604
6	0.057275	0.016471	0.014537	0.78681	0.30944	0.28589	0.075347	0.049219	0.0094704	0.0077791	0.009505	0.009505	0.009505
7	0.053104	0.017004	0.013369	0.86647	0.32933	0.3488	0.11036	0.044061	0.011213	0.0065526	0.009406	0.009406	0.009406
8	0.048511	0.016748	0.011615	0.23704	0.50047	0.31	0.10846	0.040239	0.010985	0.0065085	0.009307	0.009307	0.009307
9	0.052459	0.01858	0.010704	0.33564	0.4216	0.25404	0.067576	0.040117	0.010509	0.0062792	0.009208	0.009208	0.009208
10	0.052362	0.018343	0.013094	0.73747	0.21	0.22711	0.062268	0.044892	0.0095463	0.0050023	0.009109	0.009109	0.009109
11	0.045107	0.017671	0.0094115	0.22715	0.58208	0.30361	0.1086	0.033418	0.01087	0.0039305	0.00901	0.00901	0.00901
12	0.050584	0.017079	0.0096496	0.83848	0.305	0.33357	0.14478	0.045877	0.0097885	0.0049536	0.008911	0.008911	0.008911
13	0.048598	0.018124	0.0091846	0.43063	0.50547	0.46274	0.17043	0.032044	0.012388	0.0046389	0.008812	0.008812	0.008812
14	0.049706	0.018991	0.01005	0.44376	0.49286	0.44724	0.15822	0.033929	0.011827	0.0045465	0.008713	0.008713	0.008713
15	0.046637	0.016227	0.008984	0.52488	0.47991	0.48217	0.14722	0.032677	0.0098013	0.006457	0.008614	0.008614	0.008614
16	0.04752	0.017599	0.0086251	0.35037	0.58377	0.42784	0.13354	0.035249	0.009929	0.0031845	0.008515	0.008515	0.008515
17	0.044662	0.018009	0.0074778	0.22036	0.61679	0.37439	0.12188	0.03254	0.0092035	0.0027764	0.008416	0.008416	0.008416
18	0.044678	0.016257	0.0060803	0.89788	0.3896	0.47346	0.17866	0.03533	0.0084839	0.0028235	0.008317	0.008317	0.008317
19	0.047568	0.016725	0.0069218	0.42877	0.51007	0.48858	0.22265	0.030141	0.0096868	0.0026638	0.008218	0.008218	0.008218
20	0.041122	0.015384	0.0068664	0.4114	0.60292	0.48374	0.20449	0.027276	0.0091494	0.0023724	0.008119	0.008119	0.008119
21	0.040085	0.014889	0.0049399	0.4637	0.58118	0.51241	0.1536	0.029951	0.009213	0.0025499	0.00802	0.00802	0.00802
22	0.041337	0.017464	0.0063365	0.55295	0.59098	0.5815	0.23736	0.026815	0.0091502	0.0025235	0.007921	0.007921	0.007921
23	0.038551	0.0177	0.0064342	0.59148	0.49047	0.53699	0.16648	0.032681	0.0089836	0.0030691	0.007822	0.007822	0.007822
24	0.038734	0.017361	0.0058173	0.53771	0.56462	0.54594	0.20933	0.027141	0.0091297	0.0033217	0.007723	0.007723	0.007723

Table 3.2. Performance Metric

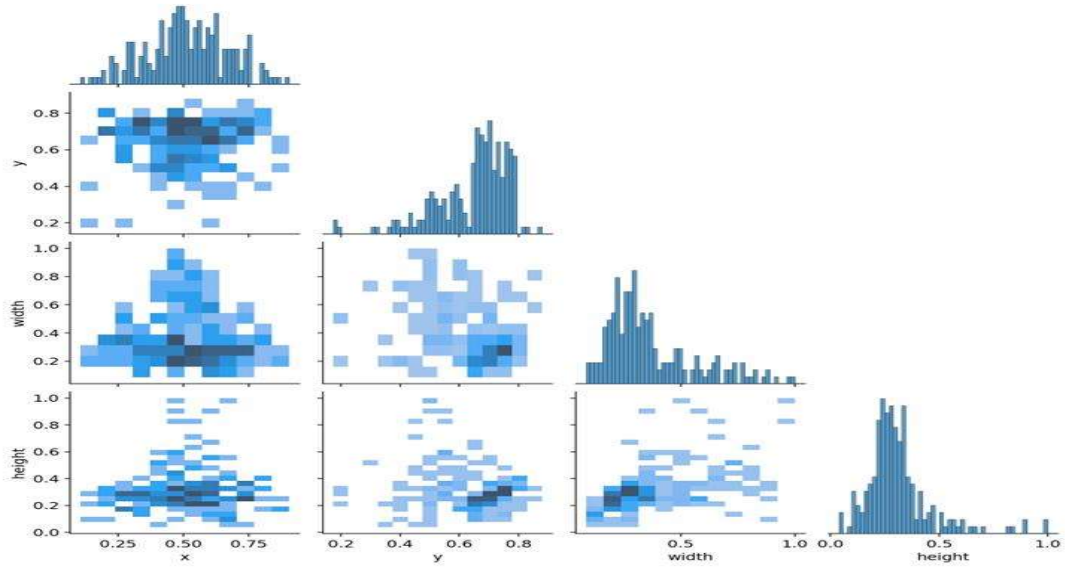

Fig 3.4. Model Correlogram

Project Execution Flow Chart

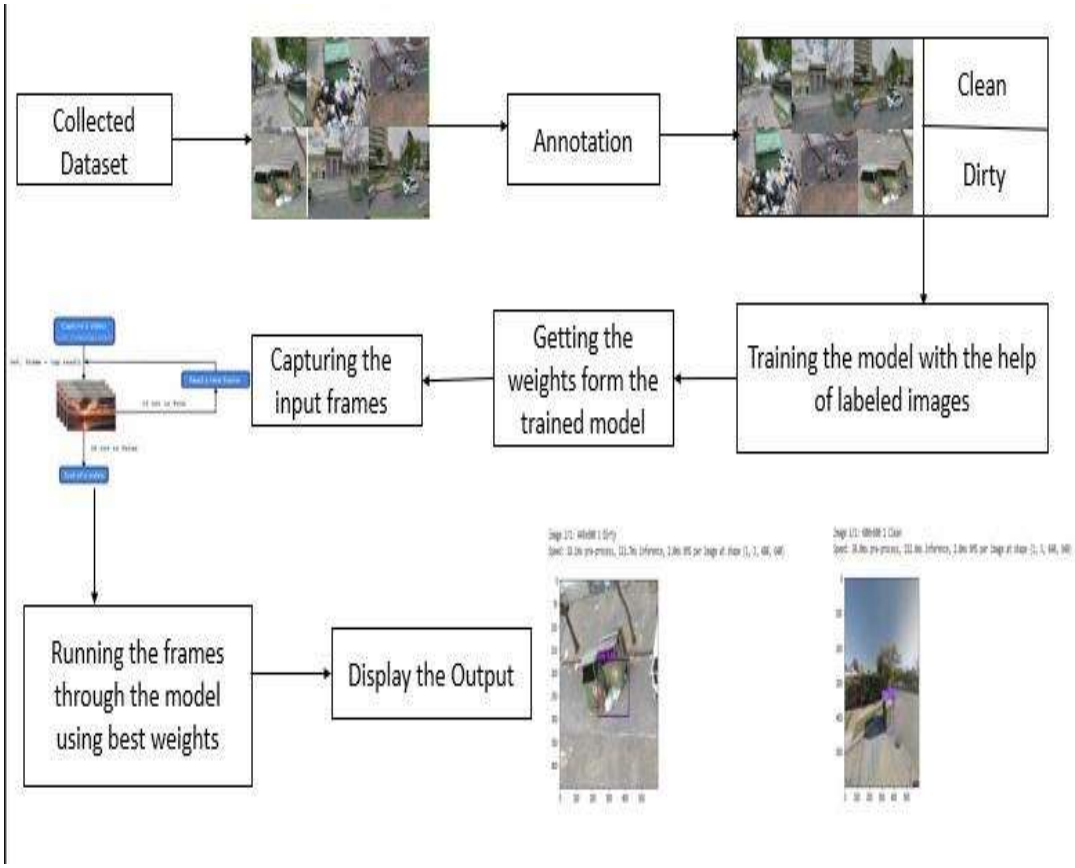

Fig 3.5. Project Flow Chart

Chapter 4

Source Code

Install all necessary packages

```
# Installing the pytorch package
!pip install torch==1.8.1+cu111 torchvision==0.9.1+cu111 torchaudio===0.8.1 -f
https://download.pytorch.org/whl/lts/1.8/torch_lts.html
```

```
!pip install gradio
```

```
#clonning YOLOV5 model from github
!git clone https://github.com/ultralytics/yolov5
```

```
# Installing all the requirement for the YOLOv5
!cd yolov5 & pip install -r requirements.txt
```

```
# Annotation package
!git clone https://github.com/tzutalin/labelImg
```

```
!pip install anyio>=3.4.0
```

```
!pip install starlette
```

```
!pip install pyqt5 lxml --upgrade
!cd labelImg && pyrcc5 -o libs/resources.py resources.qrc
#python labelImg.py
```

```
from PIL import Image
from matplotlib import cm
```

Importing the required Libraries

```
from PIL import Image
from matplotlib import cm
```

```
import torch
import matplotlib
from matplotlib import pyplot as plt
matplotlib.use('TkAgg')
%matplotlib inline
import numpy as np
import cv2
import os
import gradio as gr
```

Training the model with custom dataset

```
model = torch.hub.load('ultralytics/yolov5', 'custom', path='yolov5/runs/train/exp/weights/best.pt', force_reload=True)
```

Modelling with the new weights

```
!cd yolov5 && python train.py --img 320 --batch 16 --epochs 500 --data dataset.yml --weights yolov5s.pt --workers 2
```

Performance with Images

```
def imageclass(img):
    #img = os.path.join('data', 'images', 'Clean.083.jpg')
    results = model(img)
    results.print()
    x=np.squeeze(results.render())
    #im = Image.fromarray(np.uint8(cm.gist_earth(x)*255))
    #return im
    return x
imag = gr.inputs.Image()
lab = gr.outputs.Image()
gr.Interface(fn=imageclass,inputs=imag,outputs=lab).launch()
```

Performance with video/live cam

```
gr.Image(source='webcam')
cap = cv2.VideoCapture(0)
while cap.isOpened():
    ret, frame = cap.read()

    # Make detections
    results = model(frame)

    cv2.imshow('YOLO', np.squeeze(results.render()))

    if cv2.waitKey(10) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Chapter 5

Snapshots

GUI page to upload the images

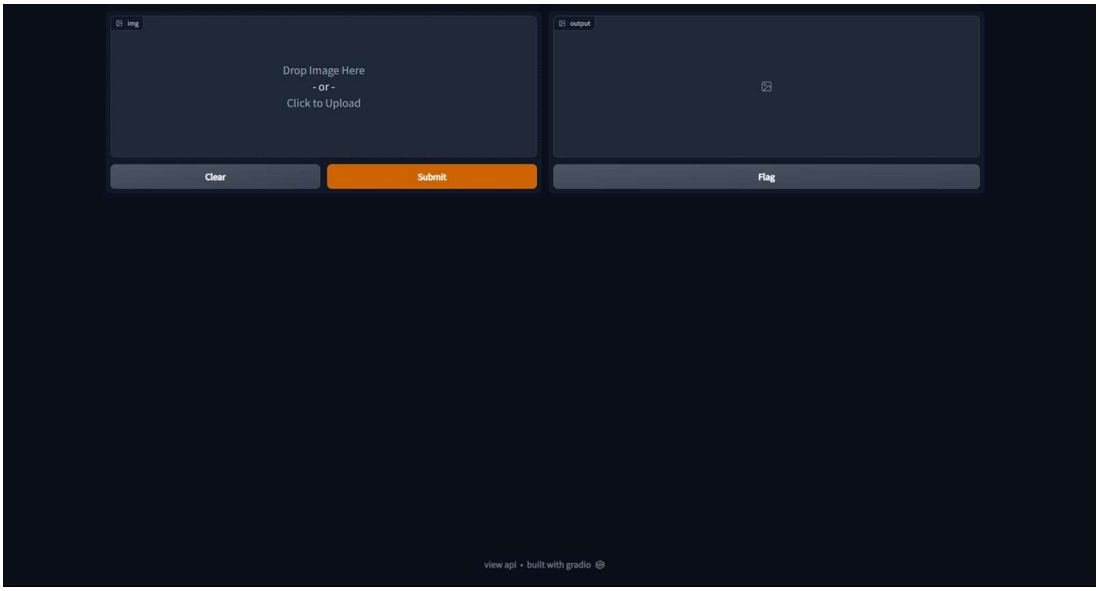

Fig 5.1.GUI Initial Page

GUI after uploading the image

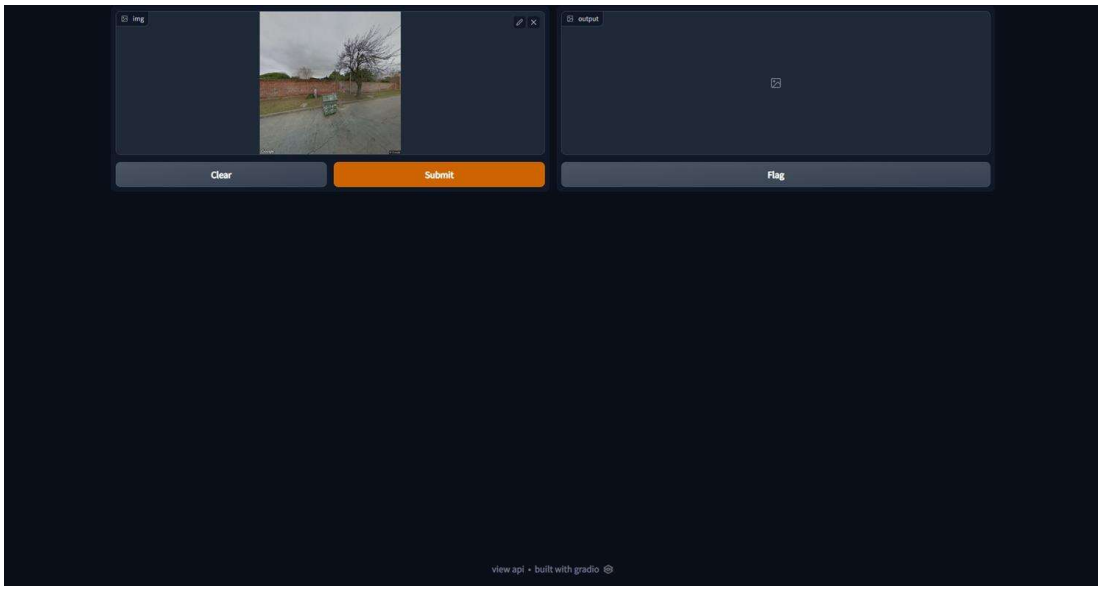

Fig 5.2. GUI Image Upload

Output GUI page

Fig 5.3.GUI Output

GUI for clear function

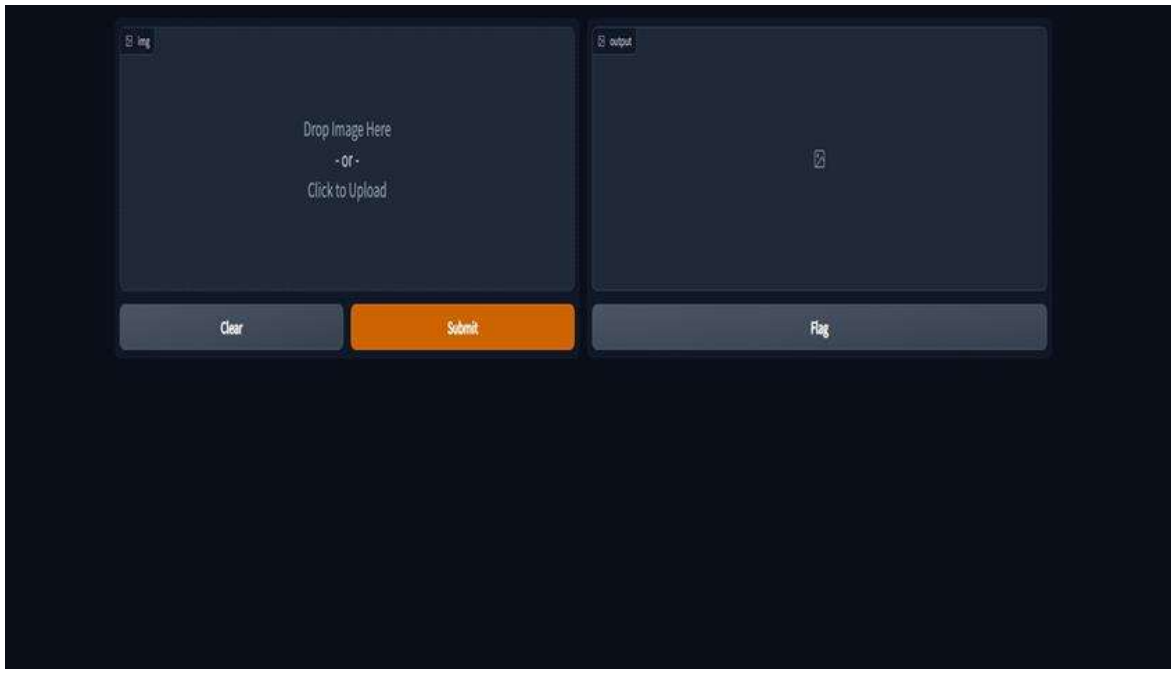

Fig 5.4.GUI Clear function

Output

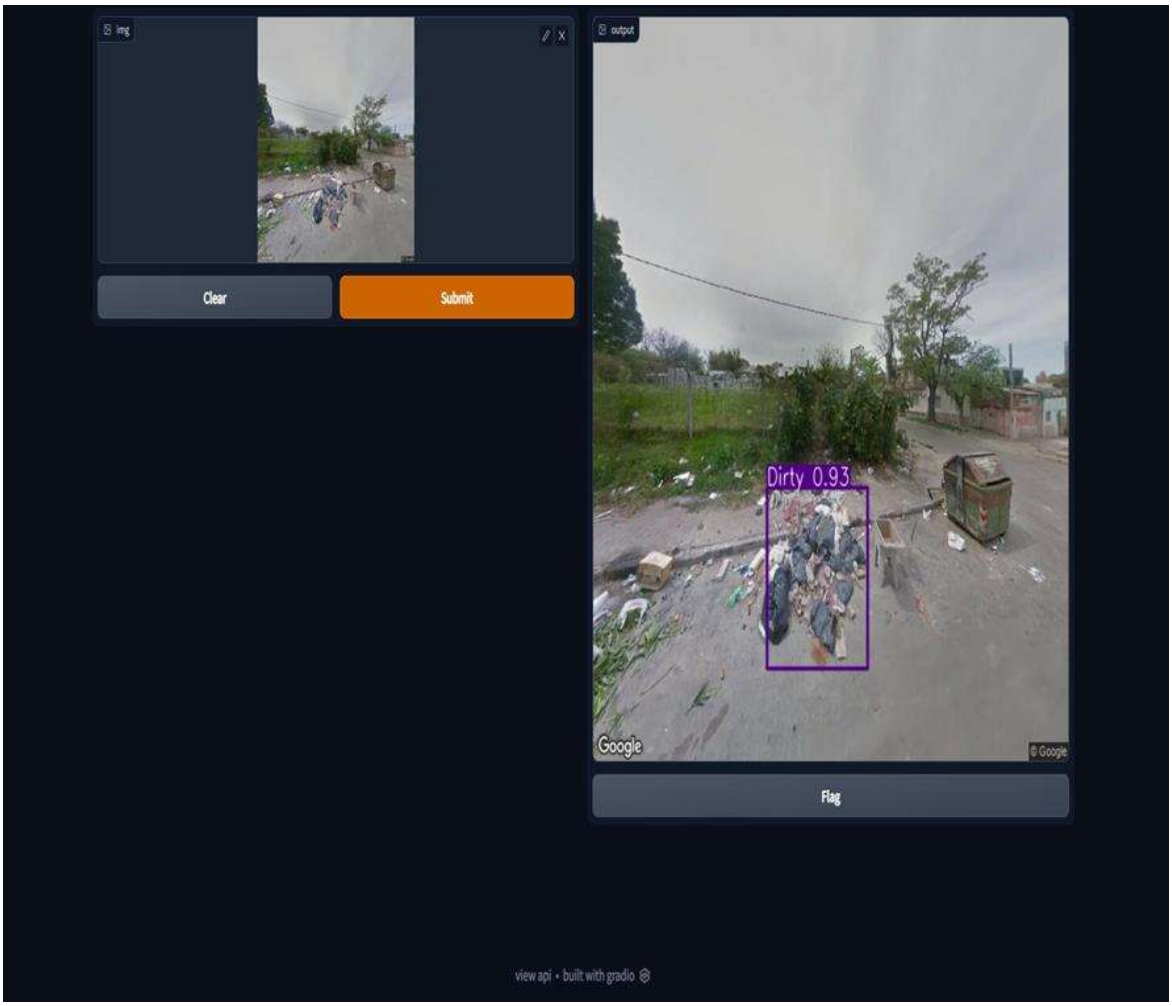

Fig 5.5.GUI Output 2

Chapter 6

Conclusion and Future Plans

YOLOv5 is a state of art object detection algorithm and it performs very well for the given problem and the dataset. YOLOv5 gives us the privilege of choosing the dense of the network so that we can compromise between accuracy and the speed of detection.

When it comes to a video-based input YOLOv5 alone won't be sufficient for detection and tracing the image and the result will be poor.

To solve this problem, we can use DeepSort algorithm along with YOLOv5 and this will enhance the detection with tracking. Secondary camera can be used for application implementing purpose