Configuration Manual

MSc Research Project
Programme Name

Stephen Angelo Kumar
Student ID:21220123

School of Computing
National College of Ireland

Supervisor: Diego Lugones

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Stephen Angelo Kumar
Student ID:	x21220123
Programme:	Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Diego Lugones
Submission Due Date:	18/09/2023
Project Title:	Configuration Manual
Word Count:	1042
Page Count:	8

Signature:	STEPHEN ANGELO KUMAR
Date:	18th September 2023

Configuration Manual

Stephen Angelo Kumar
x21220123

1 Introduction

This project detects objects in traffic camera images using YOLOv3, a fast neural network for localization and recognition. The goal is to leverage a pre-trained YOLOv3 model to identify cars, pedestrians, cyclists, and other objects in the KITTI traffic camera dataset. KITTI contains real-world driving footage, providing diverse road scenes. Object detection has key applications in autonomous vehicles, traffic analysis, and safety systems. This project will run YOLOv3 inference on KITTI images, visualize detections, and evaluate performance. The results will demonstrate YOLOv3's ability to quickly and accurately detect objects of interest in traffic images. Overall, this project establishes a basis for deploying real-time object detection in traffic video analysis using a state-of-the-art deep learning model.

2 Prerequisites

2.1 Software

Python 3.x, OpenCV 4.x or later, Darknet framework, Additional Python packages like matplotlib, numpy etc.

2.2 Hardware

AMD EPYC Processors support for efficient model training and inference,

2.3 Dataset

KITTI object detection dataset images and annotations, Camera images containing road/traffic scenes, Annotation files with object bounding boxes, Classes: car, pedestrian, cyclist etc., Training and validation splits, Images should be extracted and preprocessed.

3 Enviroment Setup

3.1 Install Python Packages:

- Create and activate a Python 3.7+ virtual environment
- Install OpenCV using pip install opencv-python
- Install other packages like matplotlib, numpy, pandas etc.

3.2 Download and Compile Darknet

- Clone the Darknet repo: git clone https://github.com/pjreddie/darknet
- Enter the darknet directory and edit the Makefile to enable GPU and CUDA
- Build Darknet by running make

3.3 Get Pre-trained Weights and Config

- Download YOLOv3 weights file from GitHub
- Get yolov3.cfg configuration file from the darknet repository
- Save weights and cfg file in the darknet directory

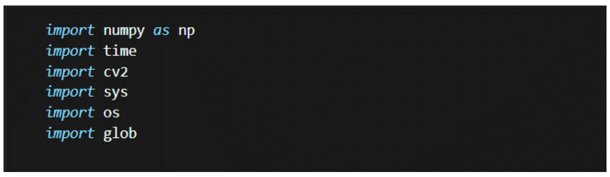
3.4 Verify Installation

- Run darknet detector test to validate Darknet build
- Load a sample image and confirm YOLOv3 model detects objects

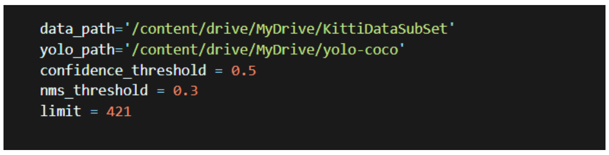
4 Implementation

4.1 Import Modules:

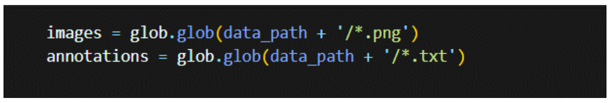
Import the necessary modules



```
import numpy as np
import time
import cv2
import sys
import os
import glob
```

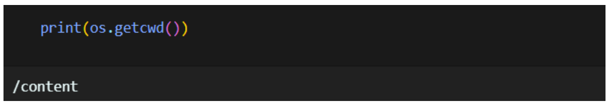


```
data_path='/content/drive/MyDrive/KittiDataSubSet'
yolo_path='/content/drive/MyDrive/yolo-coco'
confidence_threshold = 0.5
nms_threshold = 0.3
limit = 421
```

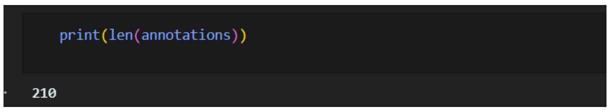


```
images = glob.glob(data_path + '/*.png')
annotations = glob.glob(data_path + '/*.txt')
```

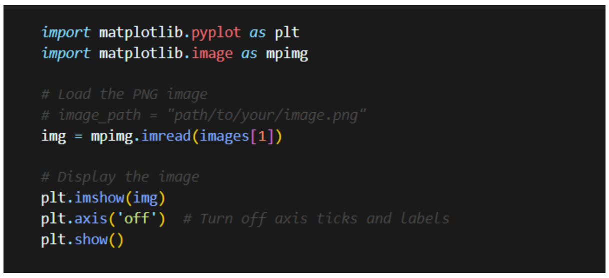
- Sets data_path and yolo_path variables to locate image dataset and YOLO model files.
- Defines confidence threshold of 0.5 and NMS threshold of 0.3 for model detection.
- Uses glob to get image and annotation file paths from data_path. Images and annotations will be used to run YOLOv3 model and evaluate detections.



```
print(os.getcwd())

/content
```



```
print(len(annotations))

210
```

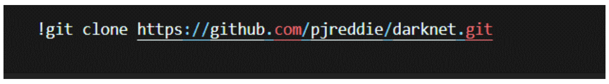


```python
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Load the PNG image
# image_path = "path/to/your/image.png"
img = mpimg.imread(images[1])

# Display the image
plt.imshow(img)
plt.axis('off')  # Turn off axis ticks and labels
plt.show()
```

-Prints the Current Working Directory.

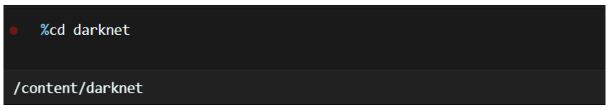
-Prints the length of annotations

-This code uses matplotlib to load and display a PNG image. The image at index 1 from the 'images' list is loaded and shown without axis ticks and labels.



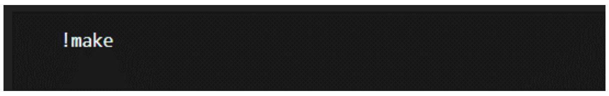
```
!git clone https://github.com/pjreddie/darknet.git
```

- Clones the Darknet repository from the provided GitHub URL.



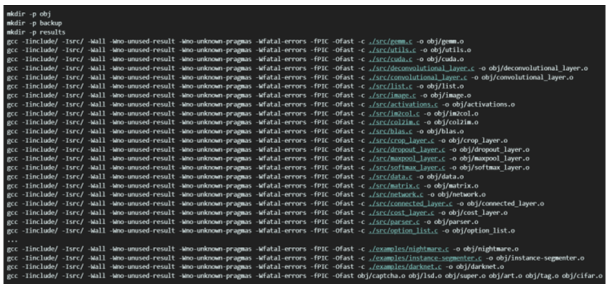
```
%cd darknet

/content/darknet
```

- Changes the current working directory to 'darknet'.



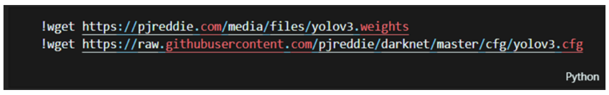
```
!make
```

- Runs the 'make' command to compile the Darknet code file.



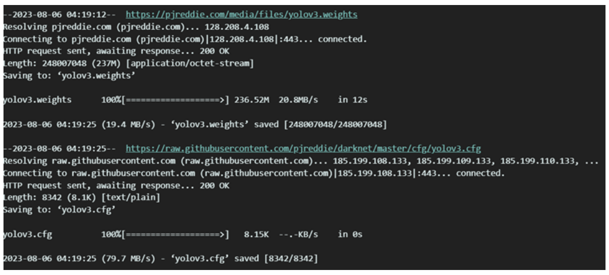
```
mkdir -p obj
mkdir -p backup
mkdir -p results
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/gemm.c -o obj/gemm.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/utils.c -o obj/utils.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/cuda.c -o obj/cuda.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/deconvolutional_layer.c -o obj/deconvolutional_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/convolutional_layer.c -o obj/convolutional_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/list.c -o obj/list.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/image.c -o obj/image.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/activations.c -o obj/activations.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/im2col.c -o obj/im2col.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/col2im.c -o obj/col2im.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/blas.c -o obj/blas.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/crop_layer.c -o obj/crop_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/dropout_layer.c -o obj/dropout_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/maxpool_layer.c -o obj/maxpool_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/softmax_layer.c -o obj/softmax_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/data.c -o obj/data.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/matrix.c -o obj/matrix.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/network.c -o obj/network.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/connected_layer.c -o obj/connected_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/cost_layer.c -o obj/cost_layer.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/parser.c -o obj/parser.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./src/option_list.c -o obj/option_list.o
...
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./examples/nightmare.c -o obj/nightmare.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./examples/instance-segmenter.c -o obj/instance-segmenter.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast -c ./examples/darknet.c -o obj/darknet.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-errors -fPIC -Ofast obj/captcha.o obj/lsd.o obj/super.o obj/art.o obj/tag.o obj/cifar.o
```

Going back to the content folder.

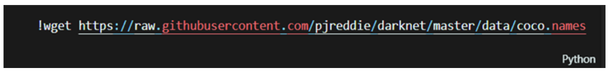


```
!wget https://pjreddie.com/media/files/yolov3.weights
!wget https://raw.githubusercontent.com/pjreddie/darknet/master/cfg/yolov3.cfg
```

Downloads YOLOv3 pre-trained weights and configuration file using wget.

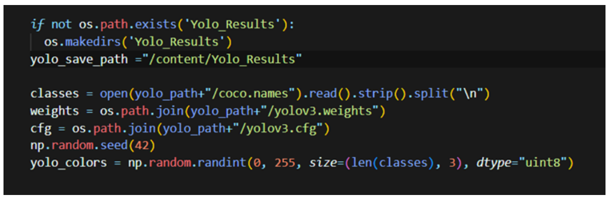


```
--2023-08-06 04:19:12--  https://pjreddie.com/media/files/yolov3.weights
Resolving pjreddie.com (pjreddie.com)... 128.208.4.108
Connecting to pjreddie.com (pjreddie.com)|128.208.4.108|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 248007048 (237M) [application/octet-stream]
Saving to: ‘yolov3.weights’

yolov3.weights      100%[===================>] 236.52M  20.8MB/s    in 12s

2023-08-06 04:19:25 (19.4 MB/s) - ‘yolov3.weights’ saved [248007048/248007048]

--2023-08-06 04:19:25--  https://raw.githubusercontent.com/pjreddie/darknet/master/cfg/yolov3.cfg
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8342 (8.1K) [text/plain]
Saving to: ‘yolov3.cfg’

yolov3.cfg          100%[===================>]   8.15K  --.-KB/s    in 0s

2023-08-06 04:19:25 (79.7 MB/s) - ‘yolov3.cfg’ saved [8342/8342]
```



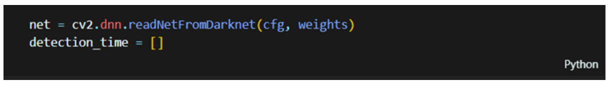
```
!wget https://raw.githubusercontent.com/pjreddie/darknet/master/data/coco.names
```

Downloads ’coco.names’ file from the given URL using wget.



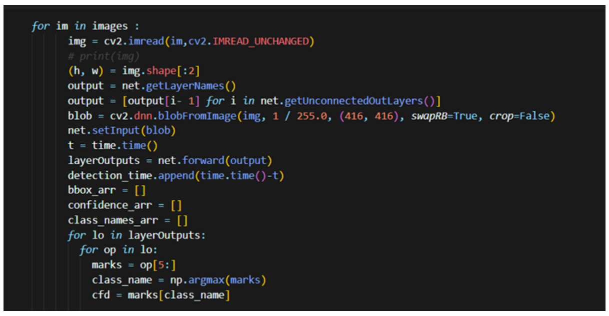
```
if not os.path.exists('Yolo_Results'):
  os.makedirs('Yolo_Results')
yolo_save_path ="/content/Yolo_Results"

classes = open(yolo_path+"/coco.names").read().strip().split("\n")
weights = os.path.join(yolo_path+"/yolov3.weights")
cfg = os.path.join(yolo_path+"/yolov3.cfg")
np.random.seed(42)
yolo_colors = np.random.randint(0, 255, size=(len(classes), 3), dtype="uint8")
```

This code creates a directory ’Yolo_Results’, sets up paths for YOLOv3 detection using COCO model, reads class names, loads pre-trained weights, and generates random colors for visualization.

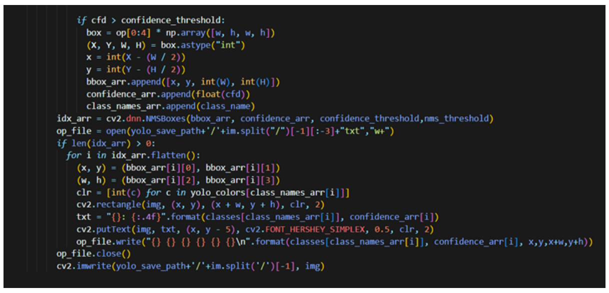


```
net = cv2.dnn.readNetFromDarknet(cfg, weights)
detection_time = []
```

In this code snippet, a YOLO neural network model is loaded using the OpenCV library. The ‘cv2.dnn.readNetFromDarknet‘ function is used to read a YOLO model from its configuration file (‘cfg‘) and corresponding pre-trained weights (‘weights‘). The ‘detection_time‘ list is initialized to store the time taken for each detection operation. This code sets up the YOLO model for further object detection operations.

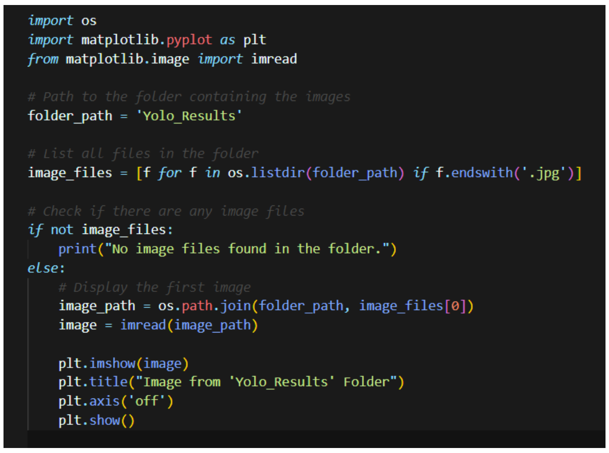


```
for im in images :
      img = cv2.imread(im,cv2.IMREAD_UNCHANGED)
      # print(img)
      (h, w) = img.shape[:2]
      output = net.getLayerNames()
      output = [output[i- 1] for i in net.getUnconnectedOutLayers()]
      blob = cv2.dnn.blobFromImage(img, 1 / 255.0, (416, 416), swapRB=True, crop=False)
      net.setInput(blob)
      t = time.time()
      layerOutputs = net.forward(output)
      detection_time.append(time.time()-t)
      bbox_arr = []
      confidence_arr = []
      class_names_arr = []
      for lo in layerOutputs:
        for op in lo:
          marks = op[5:]
          class_name = np.argmax(marks)
          cfd = marks[class_name]
```



```
            if cfd > confidence_threshold:
              box = op[0:4] * np.array([w, h, w, h])
              (X, Y, W, H) = box.astype("int")
              x = int(X - (W / 2))
              y = int(Y - (H / 2))
              bbox_arr.append([x, y, int(W), int(H)])
              confidence_arr.append(float(cfd))
              class_names_arr.append(class_name)
    idx_arr = cv2.dnn.NMSBoxes(bbox_arr, confidence_arr, confidence_threshold,nms_threshold)
    op_file = open(yolo_save_path+'/'+im.split("/")[-1][:-3]+"txt","w+")
    if len(idx_arr) > 0:
      for i in idx_arr.flatten():
        (x, y) = (bbox_arr[i][0], bbox_arr[i][1])
        (w, h) = (bbox_arr[i][2], bbox_arr[i][3])
        clr = [int(c) for c in yolo_colors[class_names_arr[i]]]
        cv2.rectangle(img, (x, y), (x + w, y + h), clr, 2)
        txt = "{}: {:.4f}".format(classes[class_names_arr[i]], confidence_arr[i])
        cv2.putText(img, txt, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, clr, 2)
        op_file.write("{} {} {} {} {} {}\n".format(classes[class_names_arr[i]], confidence_arr[i], x,y,x+w,y+h))
    op_file.close()
    cv2.imwrite(yolo_save_path+'/'+im.split('/')[-1], img)
```

This code iterates through a list of image file paths. For each image, it reads and processes the image using OpenCV's Deep Neural Network (DNN) module and a pre-trained YOLO model. Detected objects with confidence above a threshold are extracted, and non-maximum suppression is applied to remove redundant detections. Bounding boxes are drawn on the image and saved along with detection information in a text file. The processed image is saved to an output folder.



```
import os
import matplotlib.pyplot as plt
from matplotlib.image import imread

# Path to the folder containing the images
folder_path = 'Yolo_Results'

# List all files in the folder
image_files = [f for f in os.listdir(folder_path) if f.endswith('.jpg')]

# Check if there are any image files
if not image_files:
    print("No image files found in the folder.")
else:
    # Display the first image
    image_path = os.path.join(folder_path, image_files[0])
    image = imread(image_path)

    plt.imshow(image)
    plt.title("Image from 'Yolo_Results' Folder")
    plt.axis('off')
    plt.show()
```

The code reads image files from a specified folder using the ‘os‘ library, displays the first image using ‘matplotlib‘, and provides a title while removing the axis labels for visualization.

Image from 'Yolo_Results' Folder

Launched an EC2 instance on AWS to host the model and data for inference. After SSH into the server, the main folders are:

input - Contains raw video files to run object detection on. For example, trafficcam.mp4.

datadata - Stores extracted frames from the input videos in JPEG format.

cfgcfg - Holds the YOLOv3 model configuration file.

weightsweights - Contains pre-trained weights for the YOLOv3 model.

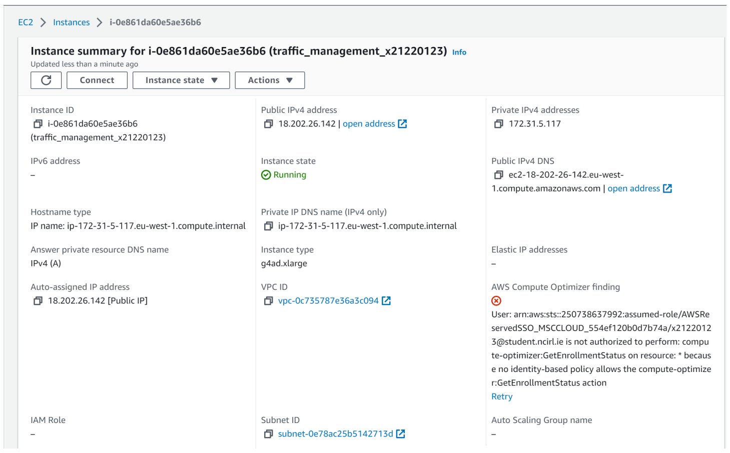
sortresults- Output folder where object detection results on frames are saved.

The main command to run is:

python3 extract.py –video input/trafficcam.mp4 –cfg cfg/yolov3 –weights yolov3.weights

This extracts all frames from trafficcam.mp4 into the data folder as JPEGs. Then it runs YOLOv3 object detection on each frame using the cfg and weights files. The detections are saved as bounding box images in sortresults.

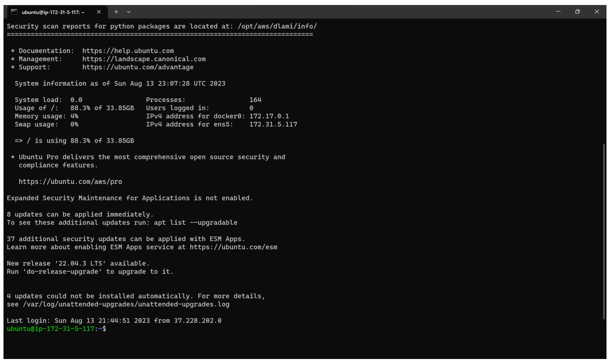
This pipeline allows new traffic camera videos to be continually processed on EC2 using the YOLOv3 model. The sortresults folder provides the output with object detections visualized on each frame. Storing the data and results on S3 could enable more scalable processing.



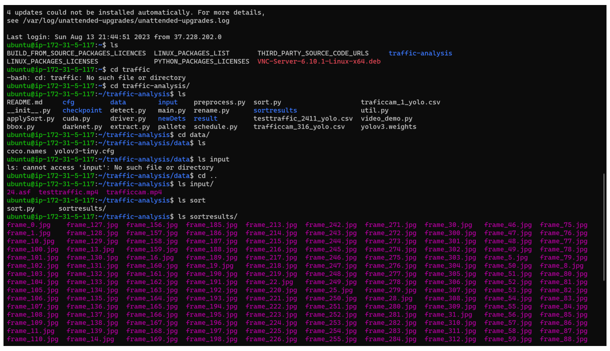
Instance summary for i-0e861da60e5ae36b6 (traffic_management_x21220123)
Updated less than a minute ago

Instance ID i-0e861da60e5ae36b6 (traffic_management_x21220123)	Public IPv4 address 18.202.26.142 | open address	Private IPv4 addresses 172.31.5.117
IPv6 address –	Instance state Running	Public IPv4 DNS ec2-18-202-26-142.eu-west-1.compute.amazonaws.com | open address
Hostname type IP name: ip-172-31-5-117.eu-west-1.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-5-117.eu-west-1.compute.internal	
Answer private resource DNS name IPv4 (A)	Instance type g4ad.xlarge	Elastic IP addresses –
Auto-assigned IP address 18.202.26.142 [Public IP]	VPC ID vpc-0c735787e36a3c094	AWS Compute Optimizer finding User: arn:aws:sts::250738637992:assumed-role/AWSReservedSSO_MSCCLOUD_554ef120b0d7b74a/x21220123@student.ncirl.ie is not authorized to perform: compute-optimizer:GetEnrollmentStatus on resource: * because no identity-based policy allows the compute-optimizer:GetEnrollmentStatus action Retry
IAM Role –	Subnet ID subnet-0e78ac25b5142713d	Auto Scaling Group name –

Figure 1: Ec2 instance



```
Security scan reports for python packages are located at: /opt/aws/dlami/info/
=============================================================================

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Aug 13 23:07:28 UTC 2023

  System load:  0.0                Processes:                164
  Usage of /:   88.3% of 33.85GB   Users logged in:          0
  Memory usage: 4%                 IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                 IPv4 address for ens5:    172.31.5.117

  => / is using 88.3% of 33.85GB

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

8 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

37 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


4 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log

Last login: Sun Aug 13 21:44:51 2023 from 37.228.202.0
ubuntu@ip-172-31-5-117:~$
```



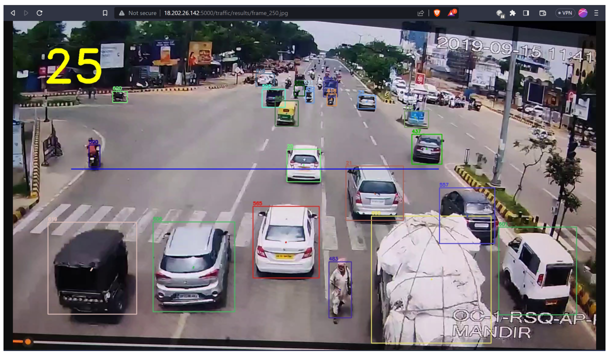
```
4 updates could not be installed automatically. For more details,
see /var/log/unattended-upgrades/unattended-upgrades.log

Last login: Sun Aug 13 21:44:51 2023 from 37.228.202.0
ubuntu@ip-172-31-5-117:~$ ls
BUILD_FROM_SOURCE_PACKAGES_LICENCES  LINUX_PACKAGES_LIST       THIRD_PARTY_SOURCE_CODE_URLS     traffic-analysis
LINUX_PACKAGES_LICENSES              PYTHON_PACKAGES_LICENSES  VNC-Server-6.10.1-Linux-x64.deb
ubuntu@ip-172-31-5-117:~$ cd traffic
-bash: cd: traffic: No such file or directory
ubuntu@ip-172-31-5-117:~$ cd traffic-analysis/
ubuntu@ip-172-31-5-117:~/traffic-analysis$ ls
README.md     cfg         data        input    preprocess.py  sort.py                    traficcam_1_yolo.csv
__init__.py   checkpoint  detect.py   main.py  rename.py      sortresults                util.py
applySort.py  cuda.py     driver.py   newDets  result         testtraffic_2411_yolo.csv  video_demo.py
bbox.py       darknet.py  extract.py  pallete  schedule.py    trafficcam_316_yolo.csv    yolov3.weights
ubuntu@ip-172-31-5-117:~/traffic-analysis$ cd data/
ubuntu@ip-172-31-5-117:~/traffic-analysis/data$ ls
coco.names  yolov3-tiny.cfg
ubuntu@ip-172-31-5-117:~/traffic-analysis/data$ ls input
ls: cannot access 'input': No such file or directory
ubuntu@ip-172-31-5-117:~/traffic-analysis/data$ cd ..
ubuntu@ip-172-31-5-117:~/traffic-analysis$ ls input/
24.asf  testtraffic.mp4  trafficcam.mp4
ubuntu@ip-172-31-5-117:~/traffic-analysis$ ls sort
sort.py      sortresults/
ubuntu@ip-172-31-5-117:~/traffic-analysis$ ls sortresults/
frame_0.jpg    frame_127.jpg  frame_156.jpg  frame_185.jpg  frame_213.jpg  frame_242.jpg  frame_271.jpg  frame_30.jpg   frame_46.jpg  frame_75.jpg
frame_1.jpg    frame_128.jpg  frame_157.jpg  frame_186.jpg  frame_214.jpg  frame_243.jpg  frame_272.jpg  frame_300.jpg  frame_47.jpg  frame_76.jpg
frame_10.jpg   frame_129.jpg  frame_158.jpg  frame_187.jpg  frame_215.jpg  frame_244.jpg  frame_273.jpg  frame_301.jpg  frame_48.jpg  frame_77.jpg
frame_100.jpg  frame_13.jpg   frame_159.jpg  frame_188.jpg  frame_216.jpg  frame_245.jpg  frame_274.jpg  frame_302.jpg  frame_49.jpg  frame_78.jpg
frame_101.jpg  frame_130.jpg  frame_16.jpg   frame_189.jpg  frame_217.jpg  frame_246.jpg  frame_275.jpg  frame_303.jpg  frame_5.jpg   frame_79.jpg
frame_102.jpg  frame_131.jpg  frame_160.jpg  frame_19.jpg   frame_218.jpg  frame_247.jpg  frame_276.jpg  frame_304.jpg  frame_50.jpg  frame_8.jpg
frame_103.jpg  frame_132.jpg  frame_161.jpg  frame_190.jpg  frame_219.jpg  frame_248.jpg  frame_277.jpg  frame_305.jpg  frame_51.jpg  frame_80.jpg
frame_104.jpg  frame_133.jpg  frame_162.jpg  frame_191.jpg  frame_22.jpg   frame_249.jpg  frame_278.jpg  frame_306.jpg  frame_52.jpg  frame_81.jpg
frame_105.jpg  frame_134.jpg  frame_163.jpg  frame_192.jpg  frame_220.jpg  frame_25.jpg   frame_279.jpg  frame_307.jpg  frame_53.jpg  frame_82.jpg
frame_106.jpg  frame_135.jpg  frame_164.jpg  frame_193.jpg  frame_221.jpg  frame_250.jpg  frame_28.jpg   frame_308.jpg  frame_54.jpg  frame_83.jpg
frame_107.jpg  frame_136.jpg  frame_165.jpg  frame_194.jpg  frame_222.jpg  frame_251.jpg  frame_280.jpg  frame_309.jpg  frame_55.jpg  frame_84.jpg
frame_108.jpg  frame_137.jpg  frame_166.jpg  frame_195.jpg  frame_223.jpg  frame_252.jpg  frame_281.jpg  frame_31.jpg   frame_56.jpg  frame_85.jpg
frame_109.jpg  frame_138.jpg  frame_167.jpg  frame_196.jpg  frame_224.jpg  frame_253.jpg  frame_282.jpg  frame_310.jpg  frame_57.jpg  frame_86.jpg
frame_11.jpg   frame_139.jpg  frame_168.jpg  frame_197.jpg  frame_225.jpg  frame_254.jpg  frame_283.jpg  frame_311.jpg  frame_58.jpg  frame_87.jpg
frame_110.jpg  frame_14.jpg   frame_169.jpg  frame_198.jpg  frame_226.jpg  frame_255.jpg  frame_284.jpg  frame_312.jpg  frame_59.jpg  frame_88.jpg
```

5 Results

After running object detection on the video frames, the results are saved as JPEG images in the sortresults folder on the EC2 instance.

We can view an individual detection result image by accessing a URL like:

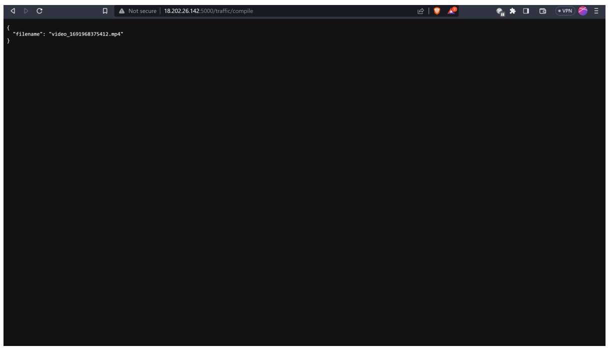
http://18.202.26.142:5000/traffic/results/frame_50.jpg
This will display frame 50 with the bounding boxes drawn around detected objects.



Not secure | 18.202.26.142:5000/traffic/results/frame_250.jpg
25
2019-09-15 11:41
OC-1-RSQ-AP-I
MANDIR

To compile the frame images into a final output video, we can trigger the compilation process:

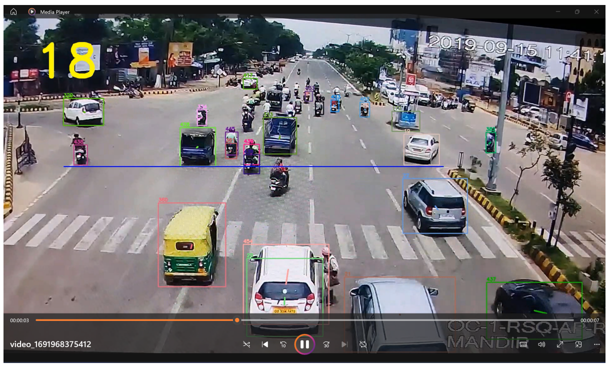
http://18.202.26.142:5000/traffic/compile
This will combine all the separate frame results into an MP4 video written to compiled/video_XXX.mp4.



```
{
  "filename": "video_1691968375412.mp4"
}
```

Finally, we can stream the compiled video:

http://18.202.26.142:5000/traffic/playVideo/video_XXX.mp4
When accessed, this URL will download the MP4 video with object detections to the local machine.


Media Player
18
2019-09-15 11:41
video_1691968375412