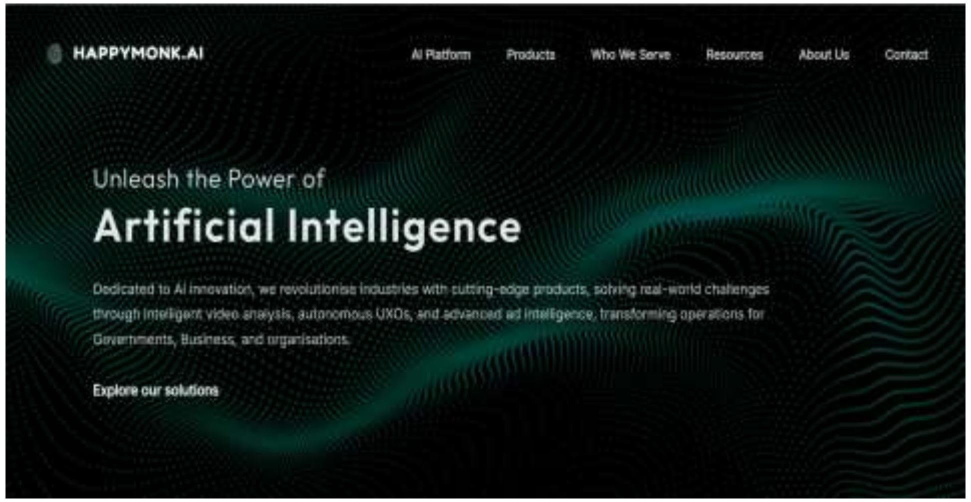
LINK TO GITHUB REPO

DOCUMENTATION

INTERNSHIP : HAPPYMONK.AI

CANDIDATE : SUDHANSHU

FIELD : DATA SCIENCE AND COMPUTER VISION



Unleash the Power of

Artificial Intelligence

Dedicated to AI innovation, we revolutionise industries with cutting-edge products, solving real-world challenges through intelligent video analysis, autonomous UXOs, and advanced ad intelligence, transforming operations for Governments, Business, and organisations.

KEY POINTS FOR DOCUMENTATION

- All the code , source images/videos , result images/videos are available in the github repository itself according to the tasks given .
- In task 2 , YOLO v5 is used instead of v8 due to some technical issues but it is working fine as desired according to the task
- Models are not trained for longer due to time issue , hence it affects their accuracy but if they will be trained longer they will be far more accurate.
- Also big models are not used . Only edge device preferable models are used like YOLO nano/small or MobileNet .
- I haven't zipped the files , you can directly clone the mentioned github repo and see the all files as they are organized in a systematic way.
- I have not made 4 video proofs as I received the mail late , So I have to hurry that's why I was only able to give two inferences of each task . Hope you understand

TASK 1 – (A)

QUESTION :

Implement YOLOv8 object detection model to identify and locates pecifically person class(No other class should be displayed on the screen in the output) in images.

Library Used : Ultralytics (YOLOV8) , OpenCV , Numpy , CVZONE

It contains 2 folders : source video and results video

It contains 2 files : task.py and yolov8.pt

Since YOLO is trained on COCO dataset , I have to get all class names of COCO dataset and then select only the person class name to get detected by model .

TASK DIFFICULTY : Very Easy

TASK 1 – (B)

QUESTION :

Use a pre-trained model, fine-tune it on a custom dataset containing images of your choice, with at least 4 types of augmentation(Include the augmentation snippet in the source code) applied on the original images.

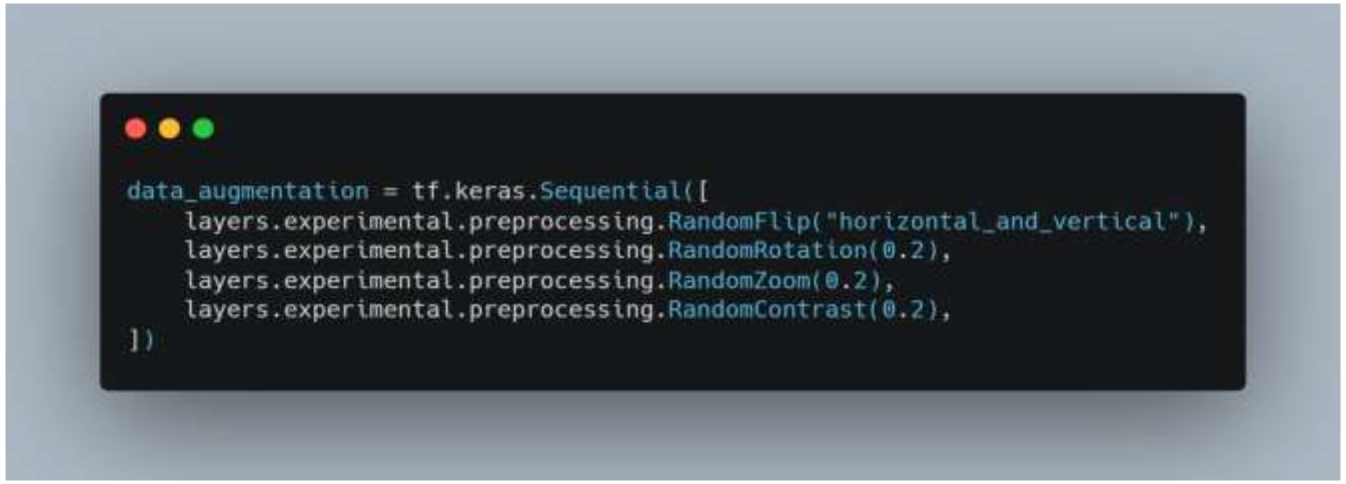
Library Used : Tensorflow , OpenCV , Numpy , Matplotlib , MobileNetV2

It contains 3 folders : test images , test video and results video

It contains 3 files : dataset.txt , notebook.ipynb and model.h5

TASK DIFFICULTY : Medium

- Firstly there were some necessary library and dependancies import .
- Then dataset was chosen by me . I decided to work on PlantVillage Dataset Tomato Class . Link to dataset is provided in dataset.txt file .
- Then data preparation and data visualization was done in order to understand and view data/images.
- Then according to question I have to do Data Augmentation , So I preffered to do this using a tensorflow layer. Code Snippet is given below :



```
data_augmentation = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2),
    layers.experimental.preprocessing.RandomZoom(0.2),
    layers.experimental.preprocessing.RandomContrast(0.2),
])
```

- Now it was a high time for me to select a good model for my problem . As I know I have enough amount of data I don't need a big model So I decided to go with MobileNetV2 which is approx 13MB in size and is well suited for mobile and edge devices .
- Then I trained the model for 15 EPOCHS with a Early Stopping callback . Training was stopped at 14 epoch and I got vaidation accuracy of 87% and train accuracy of 86% . So I knew my model was not overfitting and neither underfitting .
- Then I also got test accuracy of approx. 87% only .
- Then It was time for testing model , So firstly I test it through images and it performed extremely well . Then I leverage the model with OPENCV for realtime detections too and there also it performed good .

TASK 2– (A)

QUESTION :

Train a YOLOV8 object detection model for face detection using the dataset provided in the link below.

Library Used : Ultralytics (YOLOV5) , OpenCV , Numpy , PyTorch

It contains 2 folders : test and results

It contains 2 files : MASK DETECTOR.ipynb and yolov5 (github repo)

TASK DIFFICULTY : Easy

Since we have the pre-made dataset , It made my work lot easier as now I have to just make a dataset.yaml file with correct file path and then custom train my YOLO model on that dataset . It was tested on both images and videos and it performed exceptionally well .

YOLO nano model was used and was trained for about **16 epochs** that got us accuracy of **74% maP**

TASK 2– (B)

QUESTION :

Then train a Image classification model which takes the input as a crop from the object detection model(face crop) to classify emotions in categories [‘Happy’, ‘Sad’, ‘Neutral’] and display the class name on the face class bounding box on 4 videos.

Library Used : Ultralytics (YOLOV5) , OpenCV , Numpy , PyTorch

It contains 4 folders : test , results , data and weights

It contains 1 files : EMOTION DETECTOR.ipynb

TASK DIFFICULTY : TOUGH

- Firstly I install YOLO github repo and its dependancies .
- Since I was using PyTorch some more imports were given .
- Now since I don’t have data I have to collect data all by myself only . To tackle this out I used OpenCV and my face to make data . I took 10 images for each class ie Happy , Neutral and Sad using my webcam . Data is very less and it is not preferable by Deep Learning models .
- Now next task was to annotate my images . For this I used a tool named LabelImg to annotate images in YOLO format .
- Setting up dataset.yaml file to manage training process
- Now custom trained my YOLO model on the dataset for 100 epochs then did realtime detections using OpenCV and to my surprise It performed extremely well for two classes but poor for 1 class that is “Sad” . If we train this for more epochs and with more data , it will be performing extremely well .

THE MOST CHALLENGING PART IN THIS TASK WAS DATA COLLECTION and DATA ANNOTATION . It was my 2nd favourite task .

TASK 4

QUESTION :

Vehicle Counting and Speed Estimation: Use YOLOv8 object detection to exclusively identify vehicles, count the number of vehicles present within that video frame, and implement speed estimation for the detected vehicles in the video.

Library Used : Ultralytics (YOLOV8) , OpenCV , Numpy , CVZONE , TIME , SORT (github repo)

It contains 2 folders : source video and result video

It contains 5 files : car_counter.py , co-ordinates.py , requirements.txt , sort.py and yolov8.pt

TASK DIFFICULTY : VERY TOUGH

- Firstly importing some necessary libraries and dependancies .
- Since YOLO is trained on COCO dataset , I limited the classes to detect only vehicles and not other classes .
- Then I prepared a mask on canva so that YOLO only detects vehicle in that certain region only and joined it to original image using bitwise_and operator .
- Then I gave a unique id to each vehicle using a algorithm called SORT (Simple Online Realtime Tracking) that is available as a github repository .
- Now I made a line joining both end of highway and I calculated the co-ordinates of this line by using co-ordinates finder.py file that is present in my repository .
- Now I calculated the centre point of each vehicle and display it using a solid circle with help of cv2.circle .
- Now every time the centre point of vehicle passes through the line I made , it counts the vehicle

THE MOST CHALLENGING PART TO ME IN THIS PROJECT WAS TO FIND SPEEED OF VEHICLE

- I thought of many possible solutions like :
  - Creating two lines , once vehicle touches the first line a timer will get started and when it touches second line it get stopped . Now we have the time we can calculate speed using Distance , Time Formula . But problem here was I don’t know the distance of road as it was a random video .

- But finally while surfing the internet I found out that pixels in image can also be converted to metres so I thought to give it a try that’s why I used this on my project . But there is one problem with this approach that we have to cap the frames to a fixed rate othwerwise it will give speed variably on a same video on two different devices (one that is using CPU and other using GPU) .
- But if we know the distance between two points on a highway in real life we will go for 1st method not this one .
- Then I used OpenCV for detection on a video and it performed good.

THANK YOU !