Test Document for the Image Processing Unit

Location	Ankara, Turkey
Date	May 23, 2023, Friday
Time	18.00-19.00
Description	The accuracy of the image processing subsystem on Raspberry Pi 4 device will be examined during the tests
Aim	Observing and measuring the quality of the image processing subsystem
Expected Outcome	The image processing subsystem is expected to detect the required objects with accuracy higher than %70.
Participants	Güneş (Hüseyin) Yılmaz Birsen Özge Ünal Ege Çıtak Murat Gence Ramazan Cem Çıtak

Test Devices & Tools

1. Dataset with these five classes: *pedestrian crossing*, *red* and *green pedestrian traffic light (PTL), person* and *car*.

Ground truth: A publicly available dataset at Roboflow environment is created by composing two main sources for pedestrian crossings [1] and pedestrian-related red and green traffic lights [2] as well as some photographs from our test area and METU campus. There are 396 photographs for training, 22 photographs for validation, and 20 photographs and 2 videos for testing. Labeling of the photographs was done in Roboflow environment. Test videos were taken from YouTube [3].
The dataset is at the link **https://app.roboflow.com/kartezyencorp/aid-for-the-blind/10**, which is the 8th version of our dataset since the beginning of the project.

2. Raspberry Pi (RPI) 4 Model B with 8 GB RAM

Test Environment

Since this is an on-device basic test, the test environment will only include the image processing subsystem, and *a Raspberry Pi (RPI) 4 Model B with 8 GB RAM* is used to observe the ability of this subsystem. RPI Operating System (OS) should be "*Raspberry Pi OS Bullseye 64-bit*" in order to run YOLOv5, which is also uploaded to the device before testing.
On the test environment, RPI is connected to a PC via Secure Shell (SSH) and VNC Viewer. After connecting the RPI via PuTTy program with SSH to PC, VNC Viewer will be our control and test environment because it enables to control RPI via a PC and makes it possible to share files between RPI and PC.
Also, to generate a new model, your PC should have Internet access to obtain new weights file by training.

Test Document for the Image Processing Unit

Test Parameters

When testing methods of image detection are observed, it is seen that the accuracy of the process is measured through some concepts. The detection success of a system over a database can be depicted by precision which can be calculated through a detection over the data set.

True positive (TP): A correct detection of a ground-truth bounding box

False positive (FP): An incorrect detection of a nonexistent object or a misplaced detection of an existing object

False negative (FN): An undetected ground-truth bounding box

Precision=TP/TP+FP=TP/all detections

Recall=TP/TP+FN=TP/all ground truths

Average Precision (AP): The area under the Precision-Recall curve. In the integral below, P represents the y-axis, or Precision and R represents the Recall, or x axis inside the integral. To calculate the area, the integral is in terms of Recall (R), and the function inside the integral is function of R.

$$AP = \int_0^1 P(R)dR$$

Precision, Recall and Average Precision values are between 0 and 1, which is also the reason for the limits of the above integral for AP.

The most important metric for the accuracy of the detection of a pedestrian crossing or pedestrian-related traffic light is *mean Average Precision (mAP)*.

$$mAP \ = \ \frac{1}{N}\sum_{n=1}^{N}(AP)_n$$

N is the total number of classes of objects to be detected, which are pedestrian crossings and pedestrian-related traffic lights, so it is equal to 2. (AP)n is the average precision in the nth class. By calculating mAP, the accuracy of the image processing subsystem is calculated.

Confidence is simply how confident the algorithm is about the detected object. When an object is detected, the algorithm shows the object in a box with a percentage value. This percentage value is called the confidence which is an output coming from the software.

Confidence can be seen as success of the algorithm itself per detection. While testing average confidence value can be obtained.

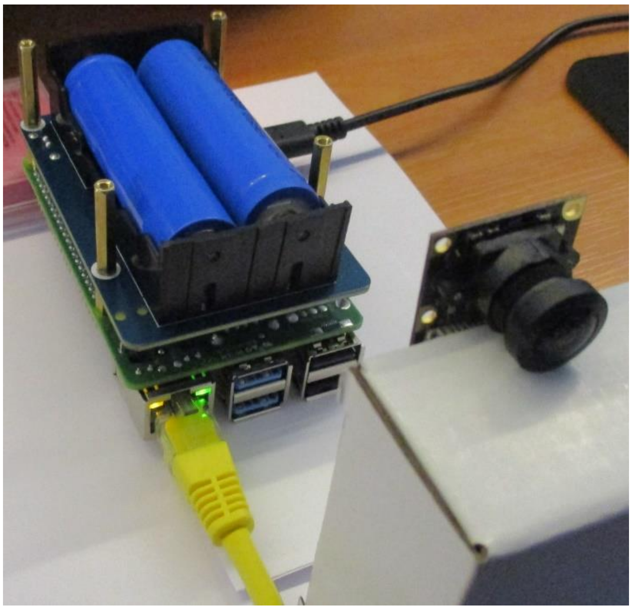
Parameter	**Range**
Precision of each detection	0.00 – 1.00
Recall of each detection	0.00 – 1.00
Mean average precision of the system	%0.00 - %100

Test Document for the Image Processing Unit

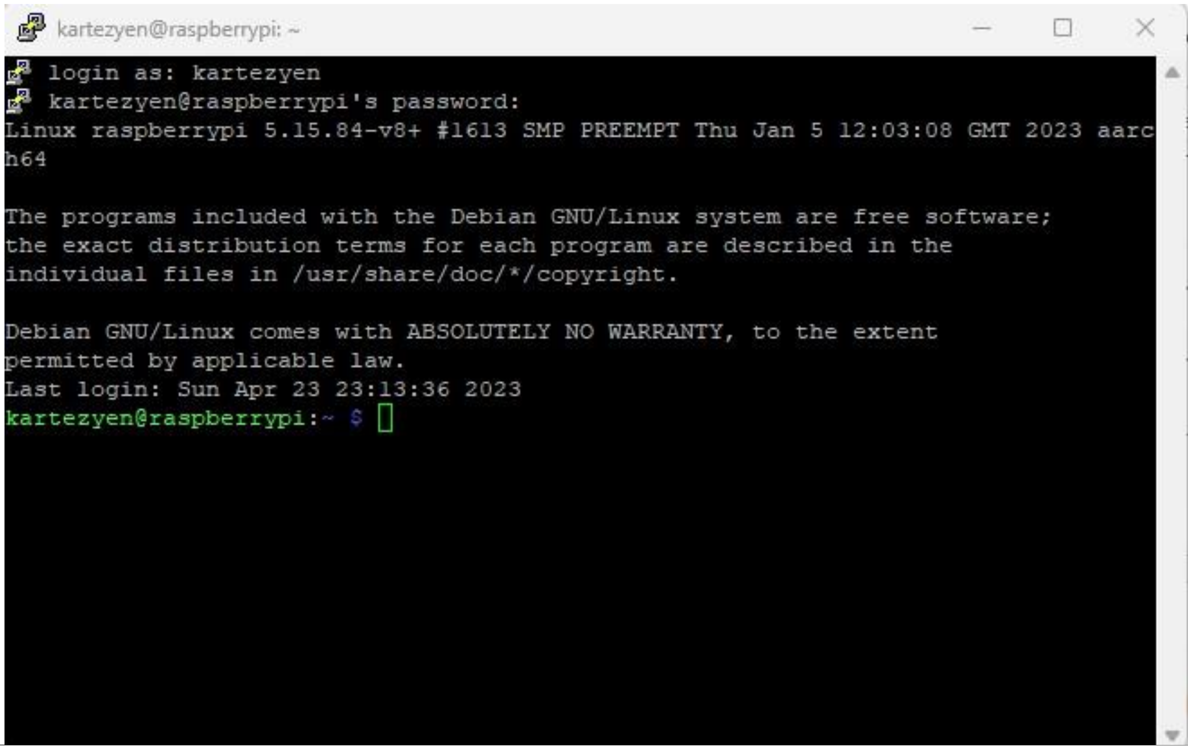
Test Procedure

Initializing RPI Device

1. Firstly, connect the ethernet cable from your computer to RPI. Then, power up the RPI. When you see green light on your RPI ethernet port as can be seen below (Ethernet cable is yellow.), and RPI is ready for SSH connection.



2. Run PuTTy program on your computer. Enter the host name (or IP address) of your RPI, and click to open. Then, SSH screen will appear, and you will enter your username and password for RPI. After a successful connection, you will see a window similar to the one below.



3. After SSH connection, run the VNC Viewer program on your computer. Click to File->New Connection. A small window will appear, and write your host name (or IP address) to VNC Server part on the top, and click to OK. Then, by double clicking to your host name (or IP address) created on the left corner of the program window, open the connection window. Username and password of the RPI should be entered there. Finally, after clicking to OK, you will see the desktop of your RPI.

Adding Model Weights to RPI Device

4. Open the Google Colab environment of YOLOv5 Custom Data Training which is available at https://colab.research.google.com/github/roboflow-ai/yolov5-custom-training-tutorial/blob/main/yolov5-custom-training.ipynb on your RPI or your computer.

5. Execute the first three Python code cells for the requirements of YOLOv5.

6. To use the dataset in Google Colab environment of YOLOv5 Custom Data Training, copy the below python code to the 4th cell of the Google Colab environment of YOLOv5 Custom Data Training. This code is the code that is automatically generated from the Roboflow environment using the annotated dataset. Before running this code, all the changes to dataset should be done such as adding new images.

```
!pip install roboflow

from roboflow import Roboflow
rf = Roboflow(api_key="vWwEZ5JaR2njF6H8CgOn")
project = rf.workspace("kartezyencorp").project("aid-for-the-blind")
dataset = project.version(10).download("yolov5")
```

7. To train the data set in YOLOv5, execute the 5th cell in Google Colab environment. The code for this cell is given below. The batch size and the number of epochs are the important parameters that affects the accuracy of the trained model. Batch is the number of images that are used during a training cycle, and an epoch is a training cycle that covers the entire dataset.

```
!python train.py --img 416 --batch 12 --epochs 200 --data {dataset.location}/data.yaml --weights yolov5s.pt --cache
```

8. Finally, for testing the 20 test images that are not in the training images, execute the 7th cell with the code

```
!python detect.py --weights runs/train/exp/weights/best.pt --img 416 --conf 0.1 --source {dataset.location}/test/images
```

The confidence threshold value can be changed by using this command. In the command, its value is 0.1 where `–conf 0.1`. Its range is the same as in the Test Parameters section of this Test Document.

9. Then, to see the test images with prediction annotations, execute the 8th cell for displaying. The test images will be seen in the cell. Also, by not executing this cell, all the tested images can also be accessed in the path `yolov5/runs/detect/exp`.

10. Recall and Precision values can be obtained by using the following command in Colab.

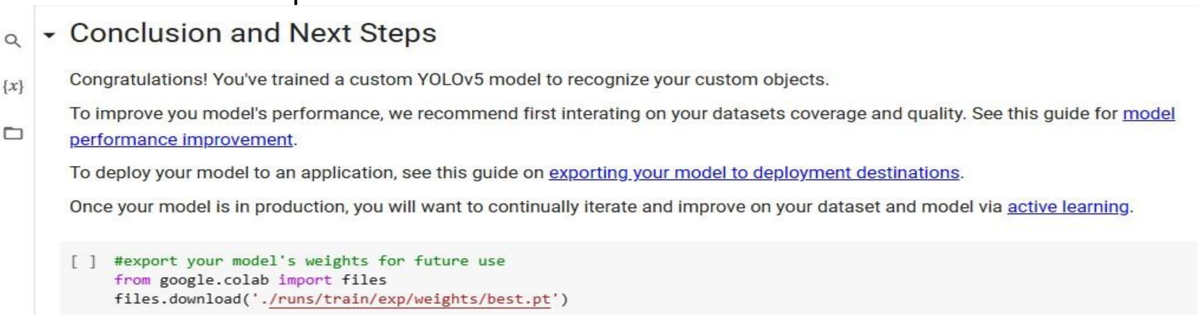
```
!python val.py --weights runs/train/exp/weights/best.pt --data {dataset.location}/data.yaml --task test
```

Before running the above code, add the below code to the end of `data.yaml` file.

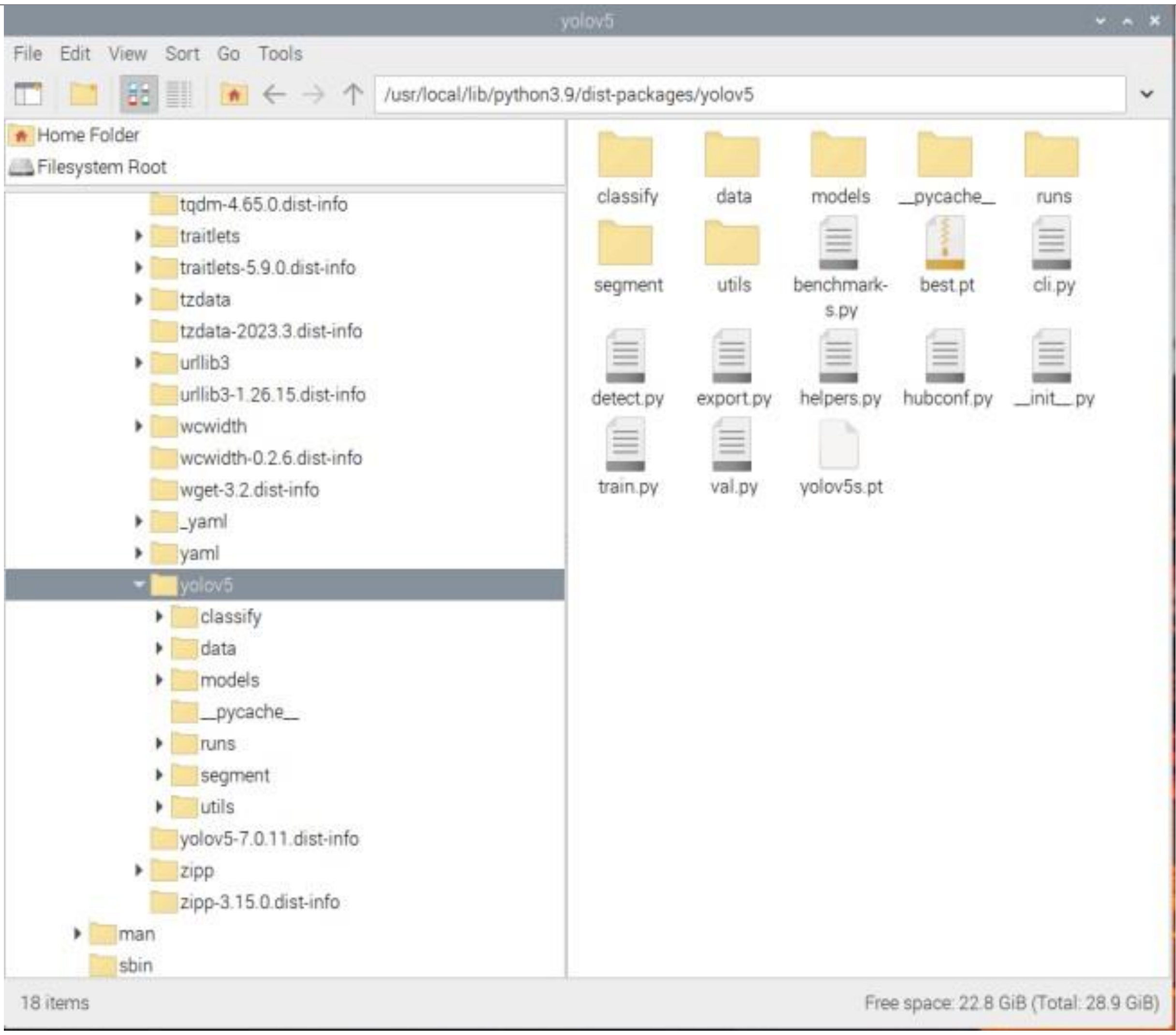
```
test: /content/datasets/Aid-for-the-Blind--10/test/images
```

The default value for the threshold of intersection over union is 0.6 for this code for Recall values. It can be adjusted by writing `–iou-thres` where `thres` is the value for the threshold. This part should be repeated for each class.

11. Finally, to download weights file, which is `best.pt`, run the last code cell in Google Colab as in seen in the below. `best.pt` file can be sent via VNC File Transfer to RPI Device.



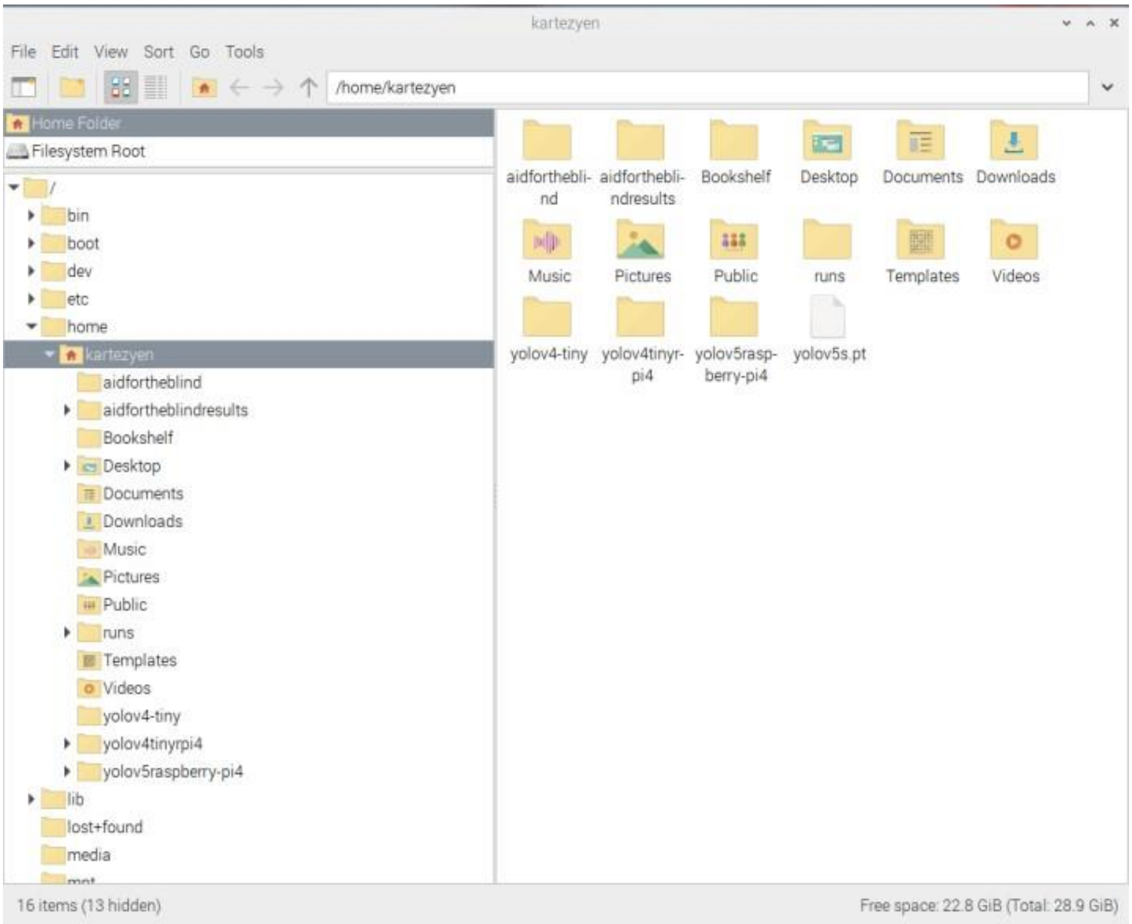
The file path to save `best.pt` file can be seen below on the RPI OS.



Now, RPI is ready to make completely offline (without Internet connection) inferences using YOLO algorithm. To upload new weights to RPI, changing the `best.pt` file is sufficient.

Using YOLO on RPI for Testing

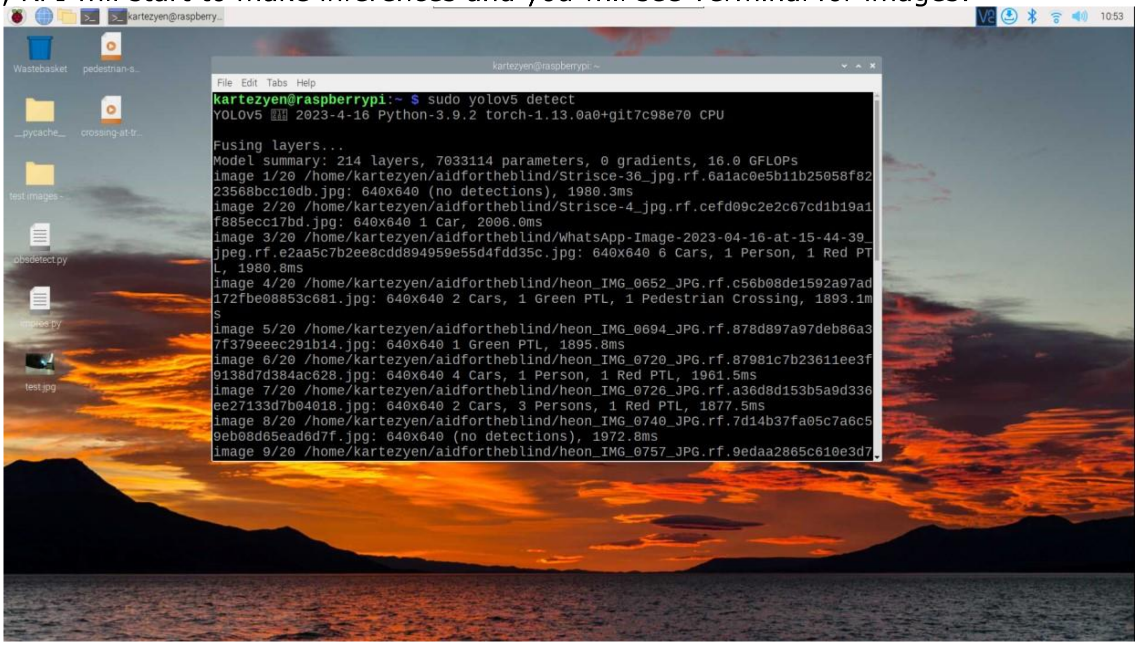
12. In YOLOv5 program, `detect.py` file can be used where to put your test images and results. Therefore, create two different folders. In our tests, we determined `aidfortheblind` for test files and `aidfortheblindresults` for results as two different folders as seen in below.



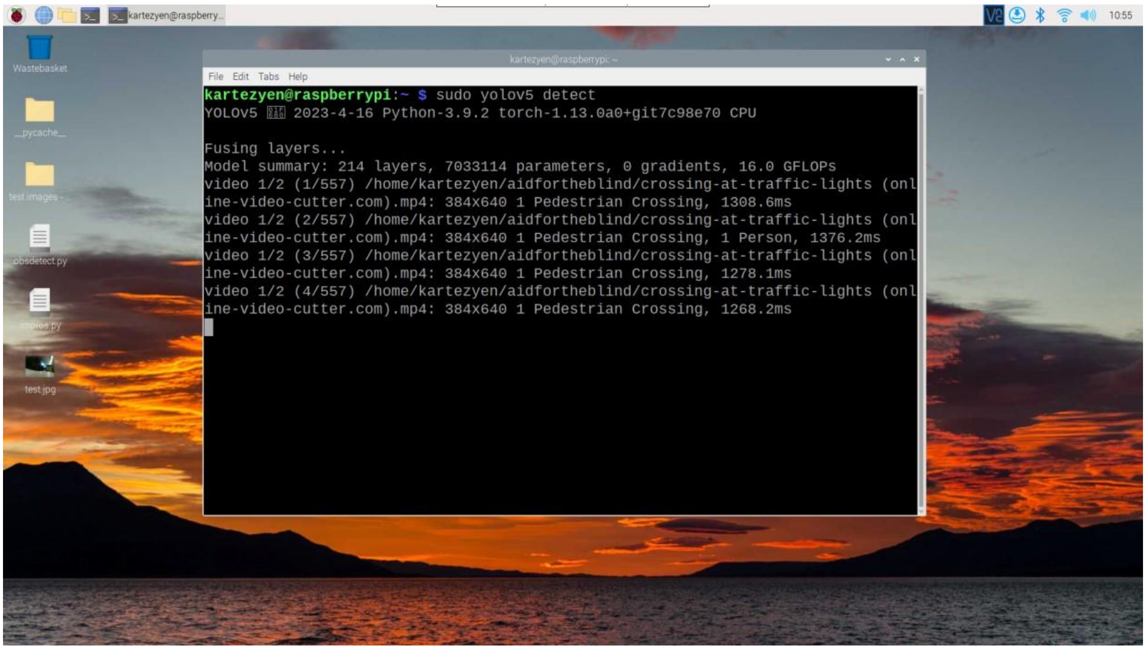
13. Upload your test media such as images and videos to `aidfortheblind` folder. Then, run this command on the RPI OS Terminal:

`sudo yolov5 detect`

Then, RPI will start to make inferences and you will see Terminal for images:

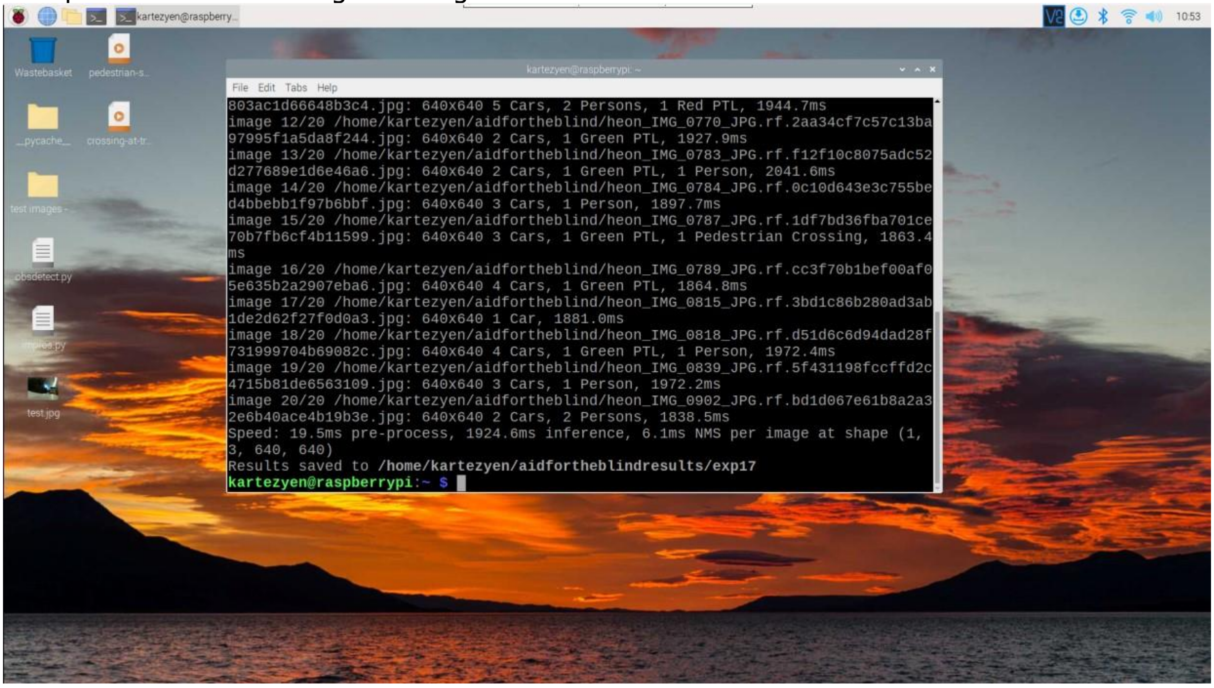


For videos:



Test Document for the Image Processing Unit

Completion of inferencing for images:

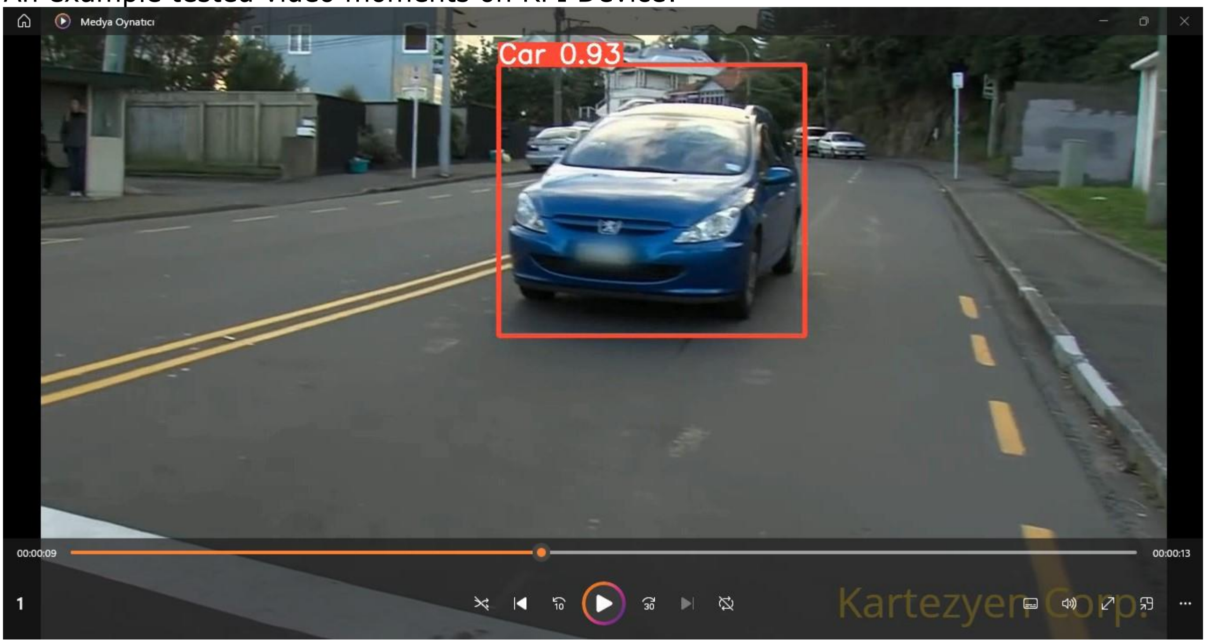


Completion of inferencing for videos is similar to this. The detection scripts seen on Terminal can be saved as text, and they will be used as an input to other subsystems such as Audio Warning Subsystem.

14. Now, tested images and videos are saved in `aidfortheblindresults` folder and ready for analysis.

An example tested image on RPI Device from our test set:

An example tested video moments on RPI Device:



15. For analysis of test images, use Google Colab as in step 10 and fill the tables as in that step. On the other hand, for videos, a qualitative analysis is sufficient. Also, inference times on the Terminal screen on RPI OS should be analyzed.

Test Document for the Image Processing Unit

Test Data

Table 1 will be filled using the average of the mAPs of the three classes as well as calculating the error relative to the expected result.

```
Fusing layers...
Model summary: 157 layers, 7031701 parameters, 0 gradients, 15.8 GFLOPs
test: Scanning /content/datasets/Aid-for-the-Blind--10/test/labels... 43 images, 0 backgrounds, 0 corrupt: 100% 43/43 [00:00
test: New cache created: /content/datasets/Aid-for-the-Blind--10/test/labels.cache
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 2/2 [00:02<00:00,  1.42s/it]
                   all         43         94      0.769       0.68      0.759      0.455
                   Car         43         22      0.313      0.727      0.339      0.203
             Green PTL         43          9      0.877      0.667      0.741      0.229
            Green TL C         43          6      0.826      0.833      0.904      0.706
  Pedestrian Crossing C        43         20      0.838       0.26      0.798      0.361
    Pedestrian Crossing        43         14      0.883      0.543      0.851      0.505
                Person         43         16      0.532      0.438       0.45      0.263
               Red PTL         43          4          1      0.975      0.995      0.442
              Red TL C         43          3      0.886          1      0.995      0.929
Speed: 0.2ms pre-process, 11.5ms inference, 8.0ms NMS per image at shape (32, 3, 640, 640)
```

Test Results on Colab where class all is mAP50 of 0.759 (the classes with C are cardboard version of the object for indoor tests)

Table 1: Mean Average Precision Value of the Detection On RPI Device

Parameter Value	Actual Performance	Expected Performance	Error
Mean Average Precision	75.9 %	>70%	-

Data Analysis

Firstly, from Table 1, we see that we achieved an accuracy of greater than 70 % by obtaining 75.9 % mAP, which is the main metric for object detection accuracy used in the literature [4].

By Table 6, the best class of objects in our model is the one with highest mAP and lower Recall values. Person class has the lowest mAP, and the Red PTL class has the highest Recall, so this means that although person class is underfitting, Red PTL class is overfitting. Thus, we need more labeled images on person, Red PTL and car classes to able to obtain more accurate detection results.

Overall, we achieved the requirement of our device in detections.

Results and Discussion

Since we have changed batch size from 16 to 12, and epoch number from 150 to 200, we obtained significant increases in mAP value when compared to the CDRR tests. Car and person classes, however, are underfitting as can be seen in Colab results of 43 test images. We need a few more images for these two classes of objects at the final version of our device.

References

[1] xN1ckuz, (2022), *Crosswalks Detection using YoloV5* [Github repository]. Available: https://github.com/xN1ckuz/Crosswalks-Detection-using-YoloV5 [Accessed: Jun. 4, 2023].

[2] samuelyu2002, (2019) ImVisible: Pedestrian Traffic Light Dataset, *LytNet Neural Network, and Mobile Application for the Visually Impaired* (Version 1.0) [Github repository]. Available: https://github.com/samuelyu2002/ImVisible [Accessed: Jun. 4, 2023].

[3] PublicResourceOrg. "Pedestrian Safety," *YouTube,* Oct. 5, 2010 [Video file]. Available: https://www.youtube.com/watch?v=CJm7FbpDFsE [Accessed: Jun. 4, 2023].

[4] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," 2020 *International Conference on Systems, Signals and Image Processing* (IWSSIP), Jul. 2020. Available: https://doi.org/10.1109/IWSSIP48289.2020.9145130 [Accessed: Jun. 4, 2023].