Configuration Manual

MSc Research Project
MSc. Data Analytics

Karthik Krishnan Iyer
Student ID: x21205485

School of Computing
National College of Ireland

Supervisor: Mr. Vikas Tomer

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Karthik Krishnan Iyer
Student ID:	x21205485
Programme:	MSc. Data Analytics
Year:	2018
Module:	MSc Research Project
Supervisor:	Mr. Vikas Tomer
Submission Due Date:	14/08/2023
Project Title:	Retail Inventory Management using Deep Learning Techniques
Word Count:	XXX
Page Count:	16

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	14th August 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Karthik Krishnan Iyer
x21205485

1 Introduction

This report is the step-by-step guide for ”Retail Inventory Management Using Deep Learning Techniques” as provided in the Research Technical Report in the methodology and Implementation. The main aim of this report is to guide the reader to the steps for implementation of this whole project and get the output and results as discussed in the Technical Report. For this there are multiple steps, libraries, software, combination of various softwares and configurations are used.

1.1 Project Overview

The project is a combination of two goal. The primary goal of the project is to detect and count the objects and empty spaces in the shelf and display the count using YOLOv7 algorithm.Secondly, the displayed count of empty spaces is extracted through tesseract OCR module and the extracted text is converted to speech through gTTS(Text to speech converter). There are series of steps which needs to be followed for the desired output.

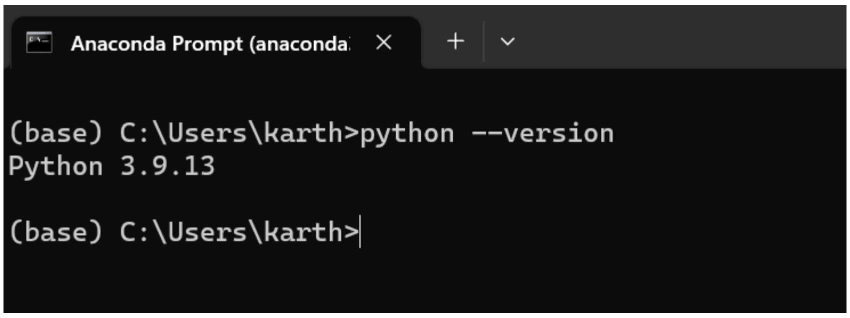
1.2 Prerequisites

1.2.1 Hardware Used:-

- Device Name - Acer Aspire 5
- Processor - 12th Gen Intel(R) Core(TM) i5-1235U 1.30 GHzn
- RAM - 16 GB
- GPU - NVIDIA GeForce MX550
- ROM - 2 GB
- OS - Windows

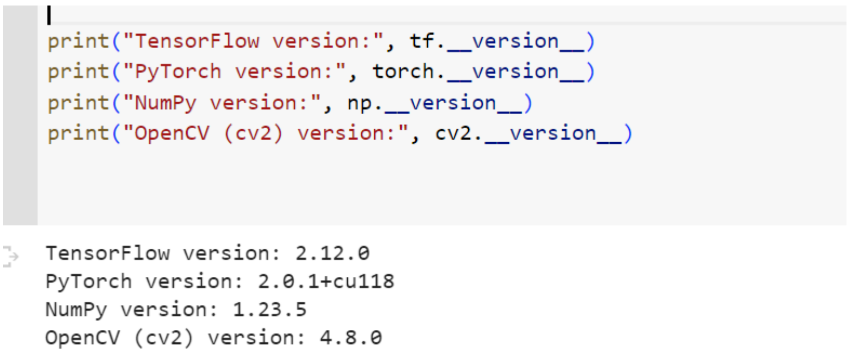
1.2.2 Software Used:-

- Programming Language:- Python
- Development Tools:- Google Colab Pro, cudnn
- Other platforms:- Google drive, labelimg, app.roboflow.com



```
(base) C:\Users\karth>python --version
Python 3.9.13

(base) C:\Users\karth>
```

Figure 1: Python Version



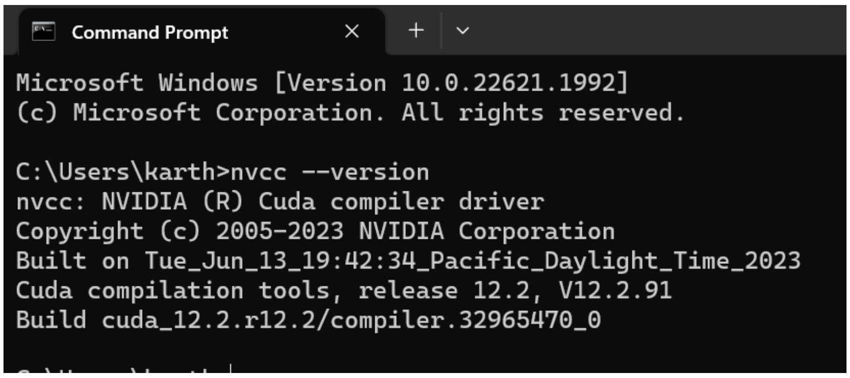
```
print("TensorFlow version:", tf.__version__)
print("PyTorch version:", torch.__version__)
print("NumPy version:", np.__version__)
print("OpenCV (cv2) version:", cv2.__version__)
```

```
TensorFlow version: 2.12.0
PyTorch version: 2.0.1+cu118
NumPy version: 1.23.5
OpenCV (cv2) version: 4.8.0
```

Figure 2: Library Versions

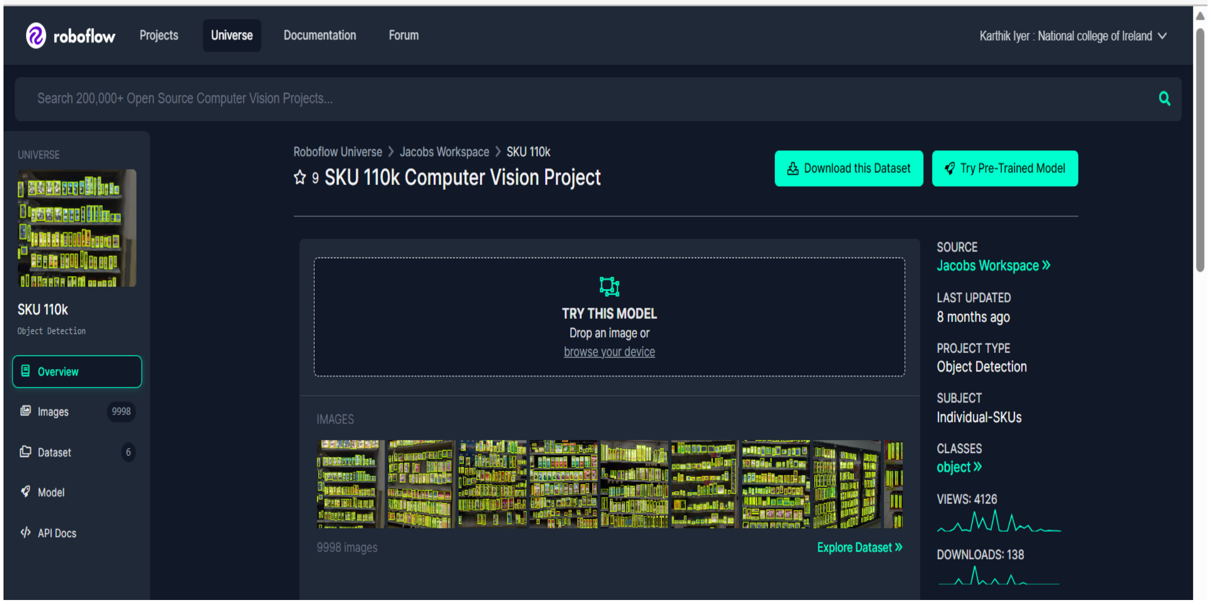
2 Software Download and setup

Python is the primary programming language used in this study.The python is downloaded and setup through anaconda environment. The downloaded version of python is 3.9.13. CUDA and cudnn gpu are essential to run this project as it consumes a lot of memory the execution speed will be less if it is executed in cpu environment.



```
Microsoft Windows [Version 10.0.22621.1992]
(c) Microsoft Corporation. All rights reserved.

C:\Users\karth>nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2023 NVIDIA Corporation
Built on Tue_Jun_13_19:42:34_Pacific_Daylight_Time_2023
Cuda compilation tools, release 12.2, V12.2.91
Build cuda_12.2.r12.2/compiler.32965470_0
```

Figure 3: CUDA Version



☆ 9 SKU 110k Computer Vision Project

SOURCE
Jacobs Workspace »

LAST UPDATED
8 months ago

PROJECT TYPE
Object Detection

SUBJECT
Individual-SKUs

CLASSES
object »

VIEWS: 4126

DOWNLOADS: 138

Figure 4: Dataset Download

2.1 Dataset Download:-

The dataset is the open source public dataset. The dataset is downloaded from roboflow website https://universe.roboflow.com/jacobs-workspace/sku-110k.

3 Data Preprocessing

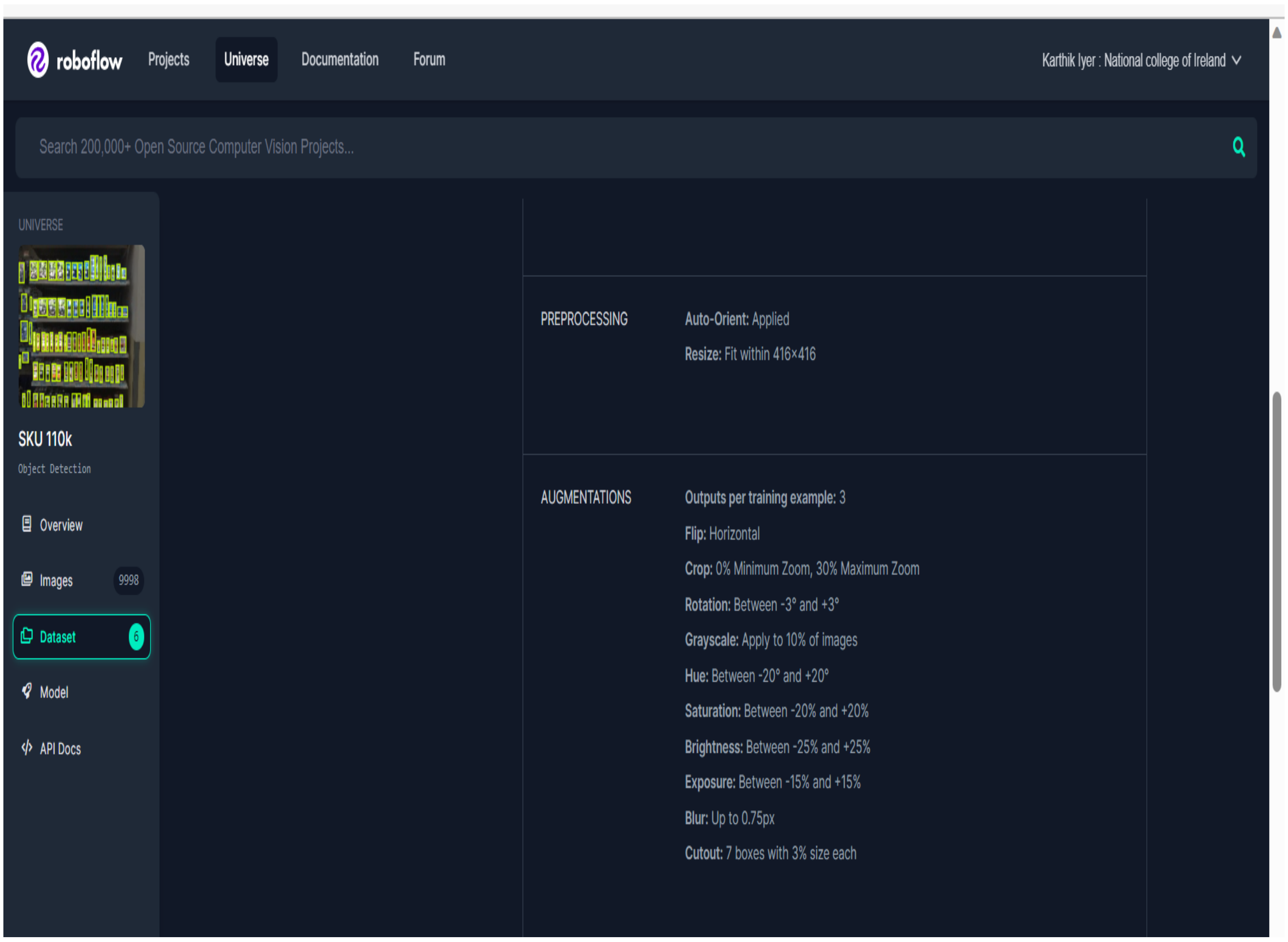
3.1 Data Augmentation

The dataset downloaded is already augmented. But some of the extra images are used with this data which are fed into roboflow which are augmented.

3.2 Data Annotation

- Firstly the downloaded dataset has predefined annotations only for objects in the shelf. So, it is essential to annotate the 'Empty' and 'AlmostEmpty' spaces.
- For this the labelimg annotation tool is used for this purpose.
- The environment was created by the name labelimg in anaconda.
- Then the labelimg was installed in the device and performed operation on the images dataset.
- The classes.txt file is created in the images folder and the empty spaces and almost empty spaces were annotated by creating bounding box.

Then the annotation performed by labelimg is saved in the format compatible with yolo with the same name as the image. Then the manual annotation by the tool for 'Empty' and 'AlmostEmpty' is combined with predefined annotation of objects to form a new file



roboflow Projects Universe Documentation Forum Karthik Iyer : National college of Ireland

Search 200,000+ Open Source Computer Vision Projects...

UNIVERSE

SKU 110k

Object Detection

Overview

Images 9998

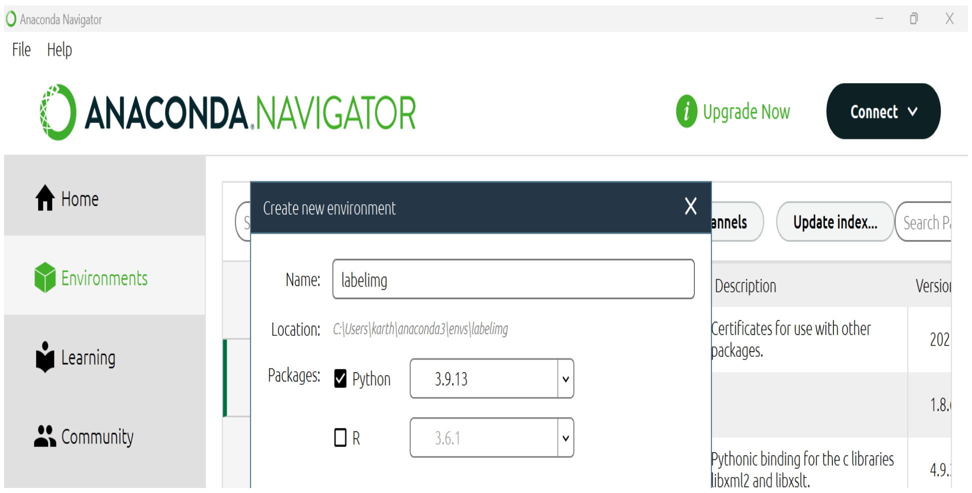
Dataset 6

Model

API Docs

PREPROCESSING	**Auto-Orient:** Applied
	Resize: Fit within 416×416
AUGMENTATIONS	**Outputs per training example:** 3
	Flip: Horizontal
	Crop: 0% Minimum Zoom, 30% Maximum Zoom
	Rotation: Between -3° and +3°
	Grayscale: Apply to 10% of images
	Hue: Between -20° and +20°
	Saturation: Between -20% and +20%
	Brightness: Between -25% and +25%
	Exposure: Between -15% and +15%
	Blur: Up to 0.75px
	Cutout: 7 boxes with 3% size each

Figure 5: Data augmentation



Anaconda Navigator

File Help

ANACONDA.NAVIGATOR Upgrade Now Connect

Home

Environments

Learning

Community

Create new environment

Name: labelimg

Location: C:\Users\karth\anaconda3\envs\labelimg

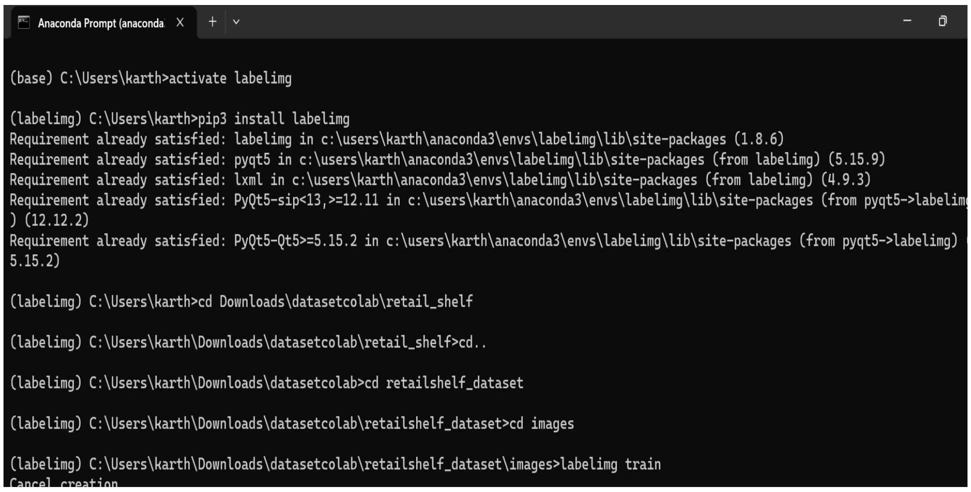
Packages: ☑ Python 3.9.13

☐ R 3.6.1

annels Update index... Search P

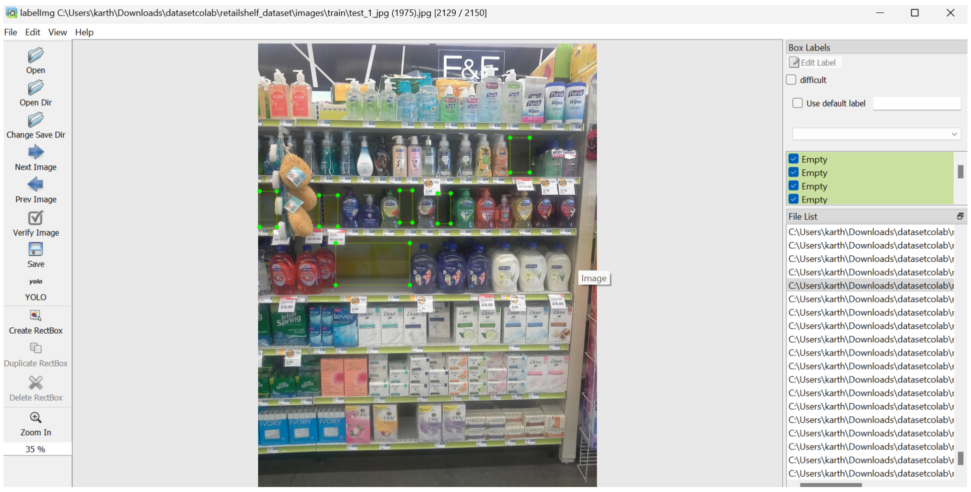
Description	Versio
Certificates for use with other packages.	202
	1.8.
Pythonic binding for the c libraries libxml2 and libxslt.	4.9.

Figure 6: Labeling Environment setup in anaconda



```
(base) C:\Users\karth>activate labelimg

(labelimg) C:\Users\karth>pip3 install labelimg
Requirement already satisfied: labelimg in c:\users\karth\anaconda3\envs\labelimg\lib\site-packages (1.8.6)
Requirement already satisfied: pyqt5 in c:\users\karth\anaconda3\envs\labelimg\lib\site-packages (from labelimg) (5.15.9)
Requirement already satisfied: lxml in c:\users\karth\anaconda3\envs\labelimg\lib\site-packages (from labelimg) (4.9.3)
Requirement already satisfied: PyQt5-sip<13,>=12.11 in c:\users\karth\anaconda3\envs\labelimg\lib\site-packages (from pyqt5->labelim
) (12.12.2)
Requirement already satisfied: PyQt5-Qt5>=5.15.2 in c:\users\karth\anaconda3\envs\labelimg\lib\site-packages (from pyqt5->labelimg)
5.15.2)

(labelimg) C:\Users\karth>cd Downloads\datasetcolab\retail_shelf

(labelimg) C:\Users\karth\Downloads\datasetcolab\retail_shelf>cd..

(labelimg) C:\Users\karth\Downloads\datasetcolab>cd retailshelf_dataset

(labelimg) C:\Users\karth\Downloads\datasetcolab\retailshelf_dataset>cd images

(labelimg) C:\Users\karth\Downloads\datasetcolab\retailshelf_dataset\images>labelimg train
Cancel creation
```

Figure 7: Labelimg installation and operation on the dataset



```
labelImg C:\Users\karth\Downloads\datasetcolab\retailshelf_dataset\images\train\test_1_jpg (1975).jpg [2129 / 2150]
```

Figure 8: Labelimg Environment setup in anaconda

of annotation. After completing this for each image in the dataset the images and their respective annotations are stored in the same folder named retailshelf.

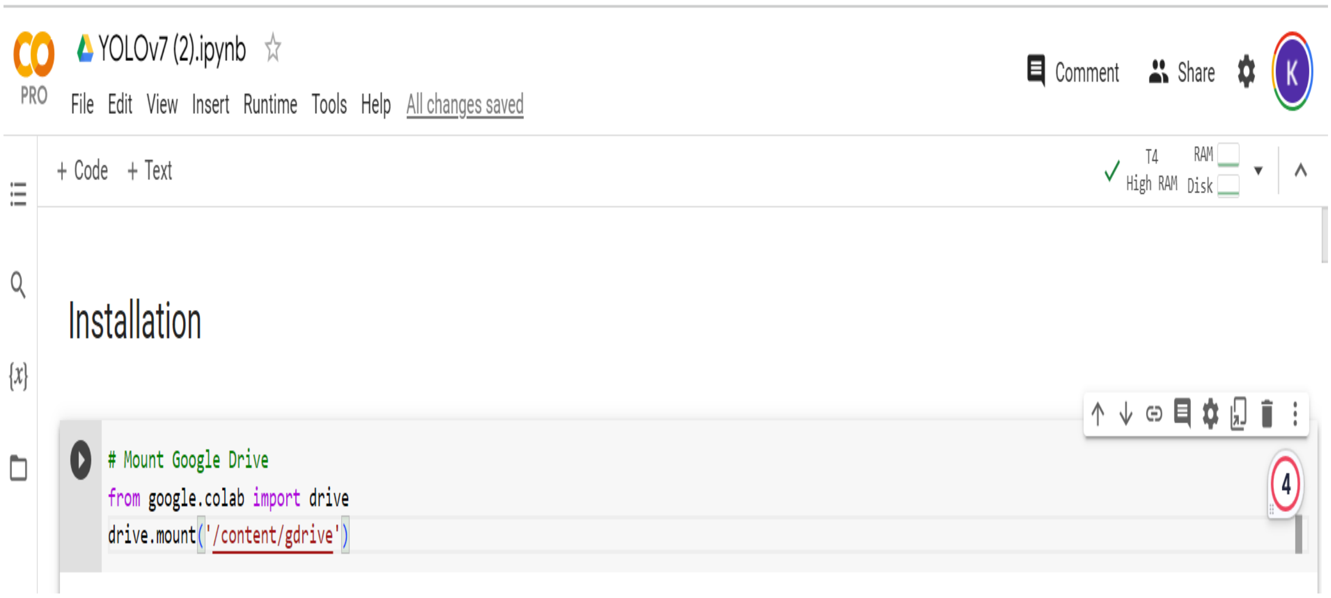
3.3 Data Splitting:-

For model training the Data splitting is an important step for obtaining effective result. The retailshelf folder which contains the images as well as annotations is splitted into train, Validation and Test. For this the splitdataset.py file is downloaded from the github. The command of split dataset is run which splits the data into train which constitutes of 80 percent, validation 10 percent and test 10 percent.The splitted data is named as retailshelfdataset.py.

As the dataset is a huge and the operation in the cpu won't be effective enough. So,the training is performed on google colab and the dataset is imported through google drive. So for uploading the dataset to google drive it is converted to .zip file.

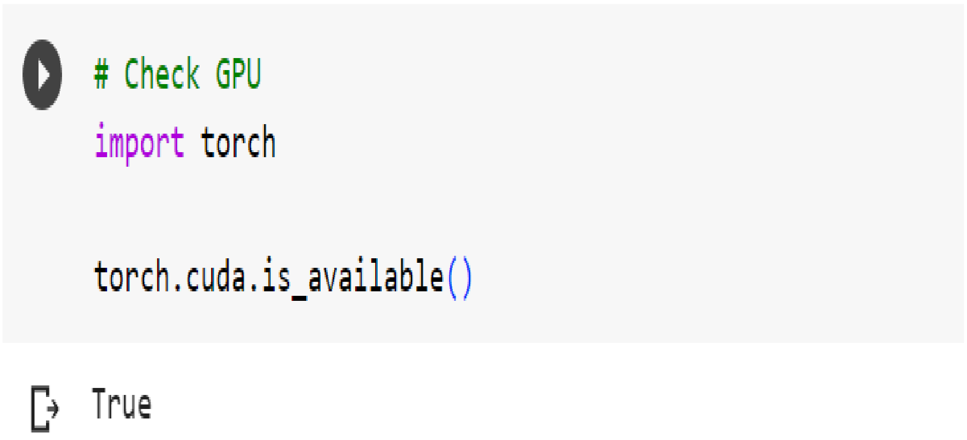
4 Implementation

4.1 Firstly the google colab pro notebook is connected to google drive.



```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/gdrive')
```

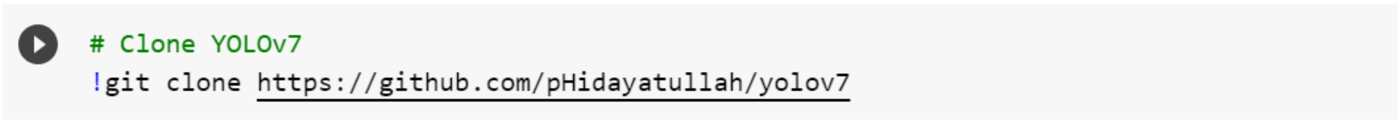
Figure 9: Mounting the google drive for model training



```
# Check GPU
import torch

torch.cuda.is_available()
```

```
True
```

Figure 10: Check for cuda



```
# Clone YOLOv7
!git clone https://github.com/pHidayatullah/yolov7
```

Figure 11: Cloning of yolov7 pretrained model with the colab pro notebook

4.2 Check if the cuda is operating.

4.3 The yolov7 pretrained model is downloaded from the github and cloned with the google drive.

4.4 This operation downloads the yolov7 folder with the required files for executing the model.

4.5 The yolov7 weights are then downloaded from the github for model training.

4.6 The .zip file is uploaded in the google drive yolov7 folder in the data subfolder.

4.7 The .zip file is extracted in colab pro notebook for performing further operation.

4.8 For performing model training the retailshelf.yaml shoulld be configured. As this file will be used model training the retailshelf.yaml file should contain the location of the dataset subfolders train, val and test.It should also contain the class names in the same order as the index of the annotation. Therefore, the index of 'AlmostEmpty' is 0, 'Empty' is 1 and 'Object' is 2. So, they are placed in that order.

4.9 Furthermore the yolov7.yaml folder present in the cfg folder also needs to be configured for training. The name of the file is changed to yolov7retailshelf.yaml. The default nc in this file is 80 that is changed to 3.

4.10 Then the model is trained with yolov7 algorithm. The command includes train.py file, batch size, retailshelf.yaml file which contains the location of the dataset subfolders with classes. Likewisely yolov7retailshelf.yaml which contains the nc information.It uses yolov7 weights for training. It is trained for 300 epochs.

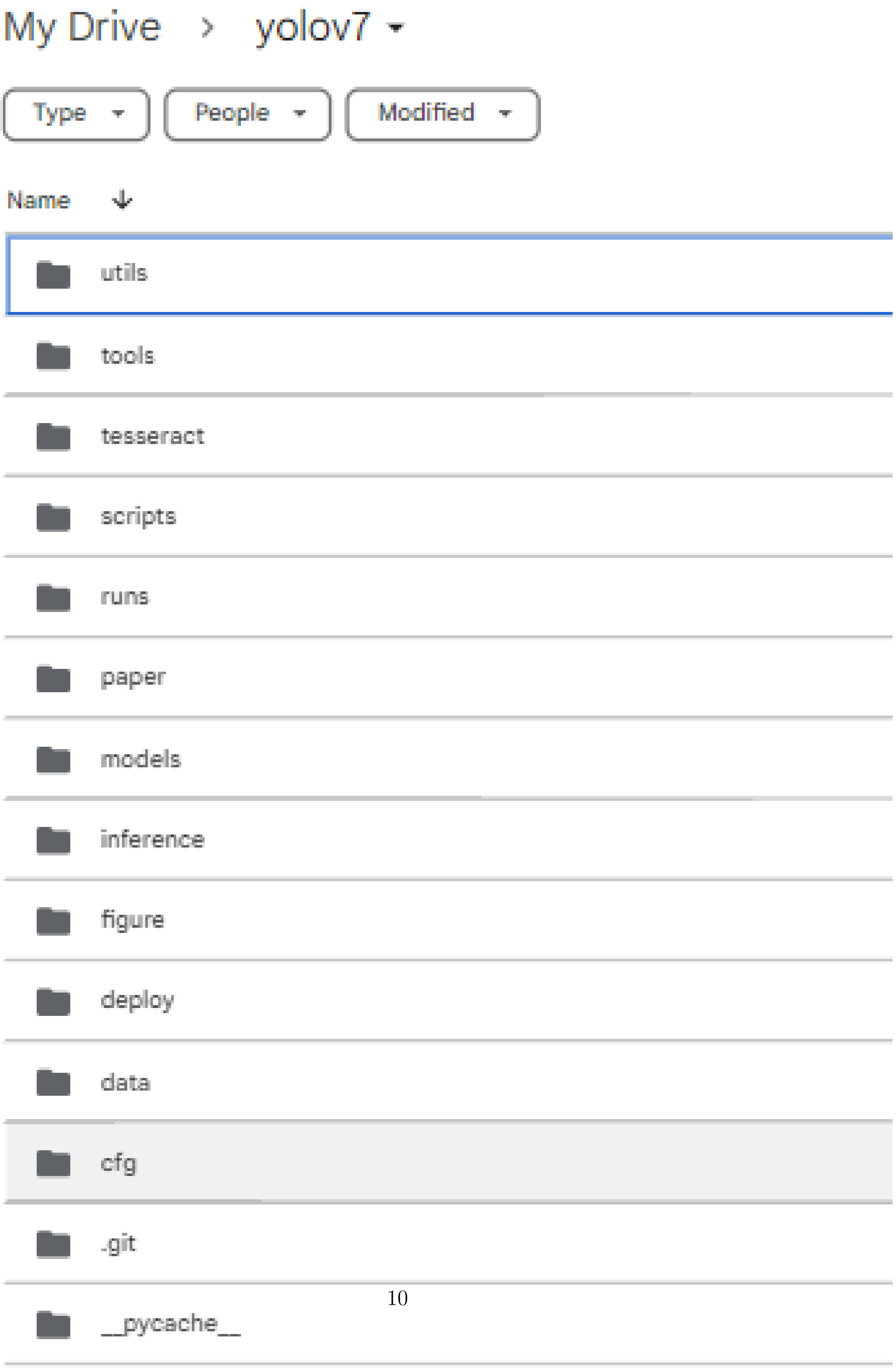
4.11 After a point the training is stopped after certain number of epochs. So,to continue the training the command below is used which used last.pt weights to start from the epochs where it stopped.

The model training result are stored in the 'runs'

4.12 So, after the model training, the object detection test is performed on some random image which is not from the dataset to check the detection results. For this the detect.py file is used. The random image is downloaded and stored in the inference folder.

4.13 The second part is object counting. So, for this the detect-andcount.py file is downloaded from github. This file has code for running through every bounding box and counting them on the basis of class labels. It also initiates the counter for count of objects and empty spaces so that the count can be displayed. There have been some alterations made to the code then it is used to detect and count the objects as well as empty spaces simultaneously.These results are stored in runs detect. This code is used to display the detected image.

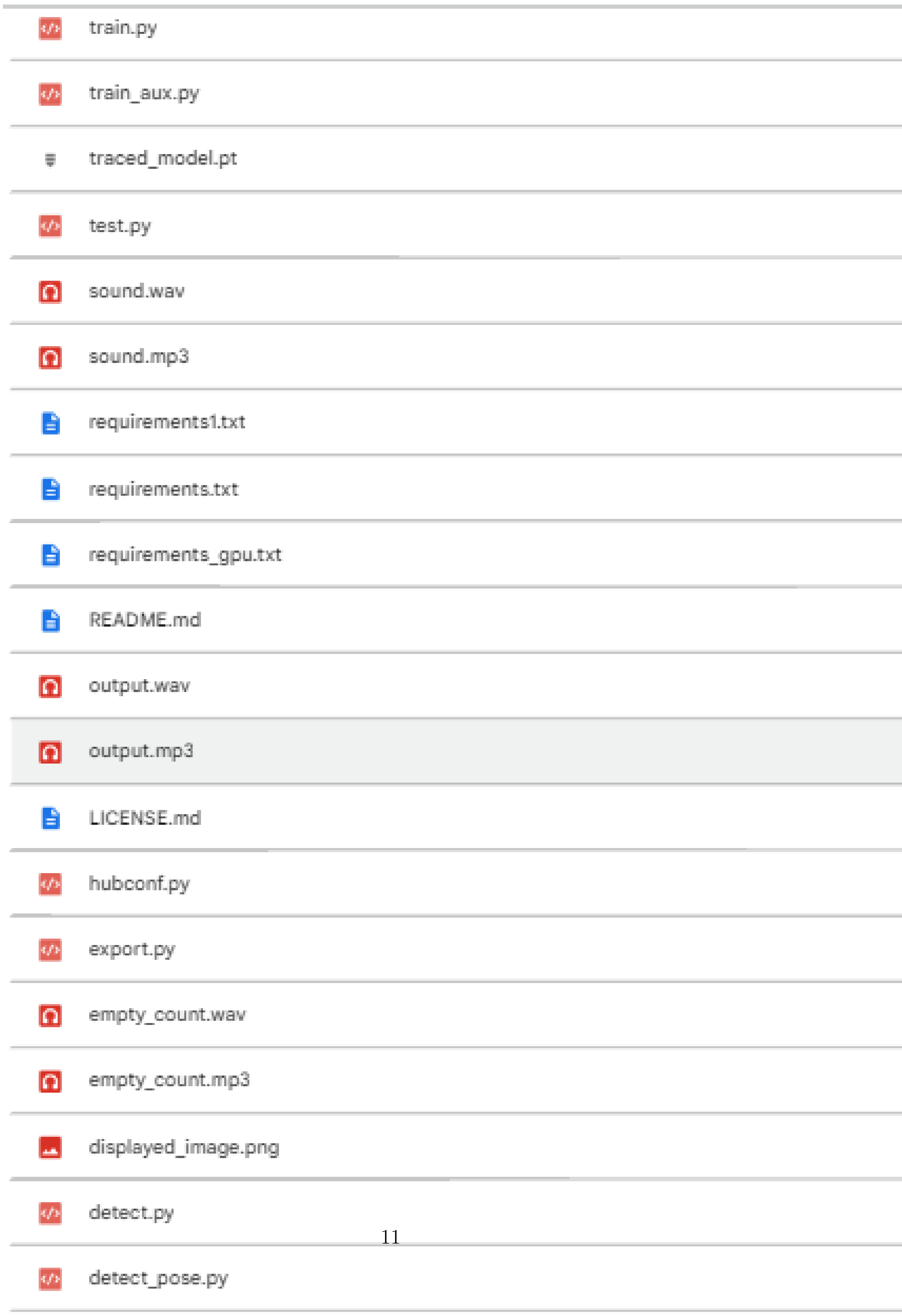
4.14 Now the final stage is to extract the count of the empty spaces and convert it to speech so that the system can speak out to the store management that how many empty spaces are present in the shelf.This involves importing some important libraries which include tesseract OCR for text extraction, gTTS for text to speech conversion. cv2 and IPython.Display is used to show the image simultaneously with the audio.


My Drive › yolov7

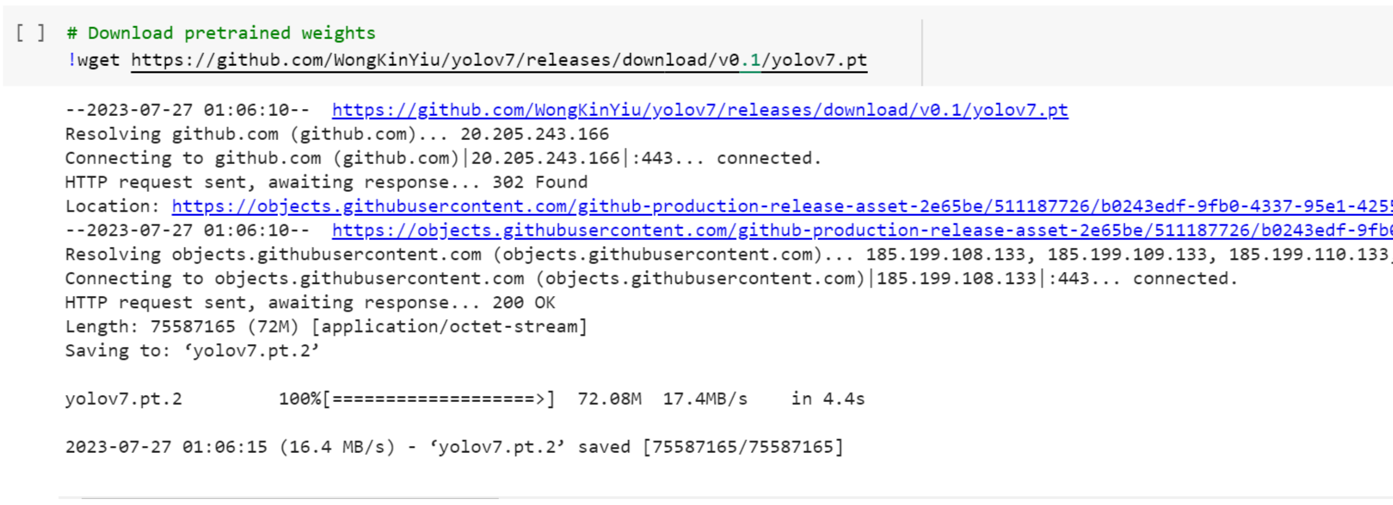
Type People Modified

Name

utils
tools
tesseract
scripts
runs
paper
models
inference
figure
deploy
data
cfg
.git
__pycache__

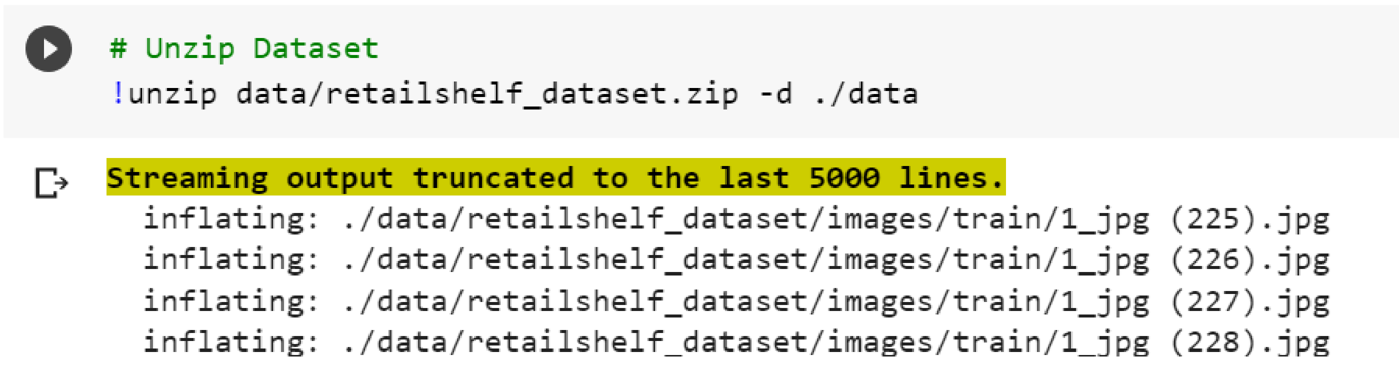


train.py
train_aux.py
traced_model.pt
test.py
sound.wav
sound.mp3
requirements1.txt
requirements.txt
requirements_gpu.txt
README.md
output.wav
output.mp3
LICENSE.md
hubconf.py
export.py
empty_count.wav
empty_count.mp3
displayed_image.png
detect.py
detect_pose.py



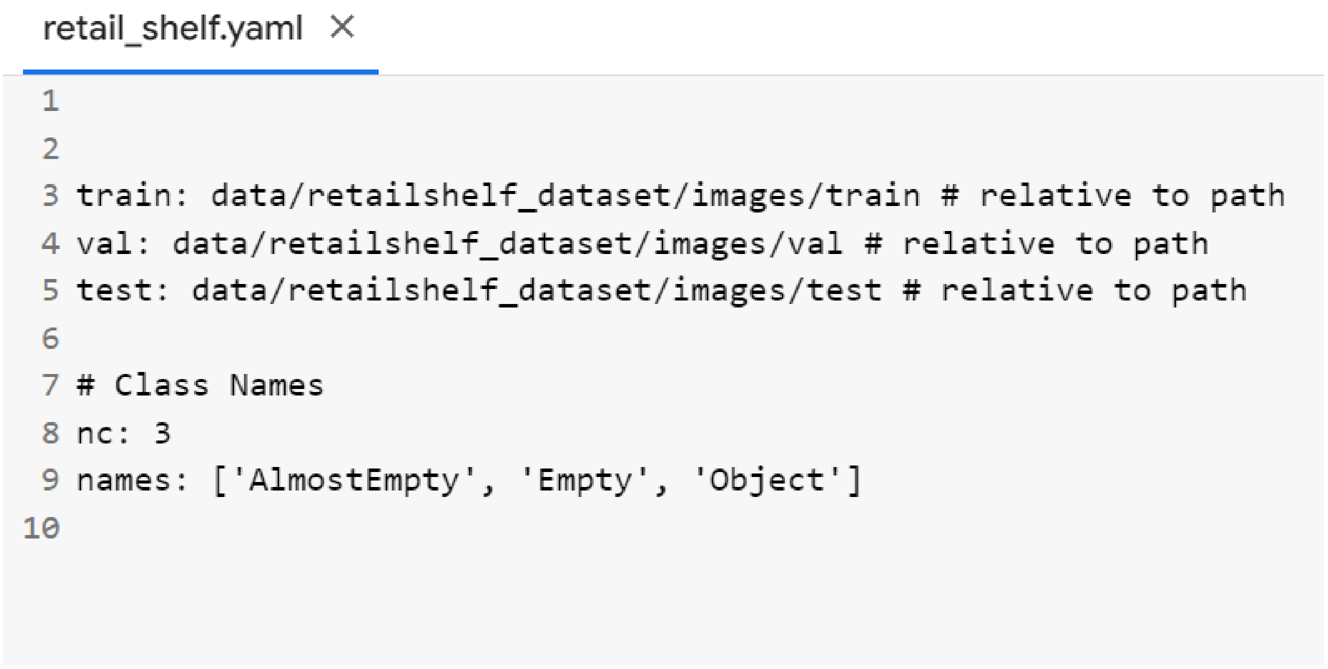
```
[ ]  # Download pretrained weights
     !wget https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7.pt

     --2023-07-27 01:06:10--  https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7.pt
     Resolving github.com (github.com)... 20.205.243.166
     Connecting to github.com (github.com)|20.205.243.166|:443... connected.
     HTTP request sent, awaiting response... 302 Found
     Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/511187726/b0243edf-9fb0-4337-95e1-4255
     --2023-07-27 01:06:10--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/511187726/b0243edf-9fb0
     Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133,
     Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.108.133|:443... connected.
     HTTP request sent, awaiting response... 200 OK
     Length: 75587165 (72M) [application/octet-stream]
     Saving to: ‘yolov7.pt.2’

     yolov7.pt.2         100%[==================>]  72.08M  17.4MB/s    in 4.4s

     2023-07-27 01:06:15 (16.4 MB/s) - ‘yolov7.pt.2’ saved [75587165/75587165]
```

Figure 14: Cloning of yolov7 pretrained model with the colab pro notebook



```
# Unzip Dataset
!unzip data/retailshelf_dataset.zip -d ./data

Streaming output truncated to the last 5000 lines.
  inflating: ./data/retailshelf_dataset/images/train/1_jpg (225).jpg
  inflating: ./data/retailshelf_dataset/images/train/1_jpg (226).jpg
  inflating: ./data/retailshelf_dataset/images/train/1_jpg (227).jpg
  inflating: ./data/retailshelf_dataset/images/train/1_jpg (228).jpg
```

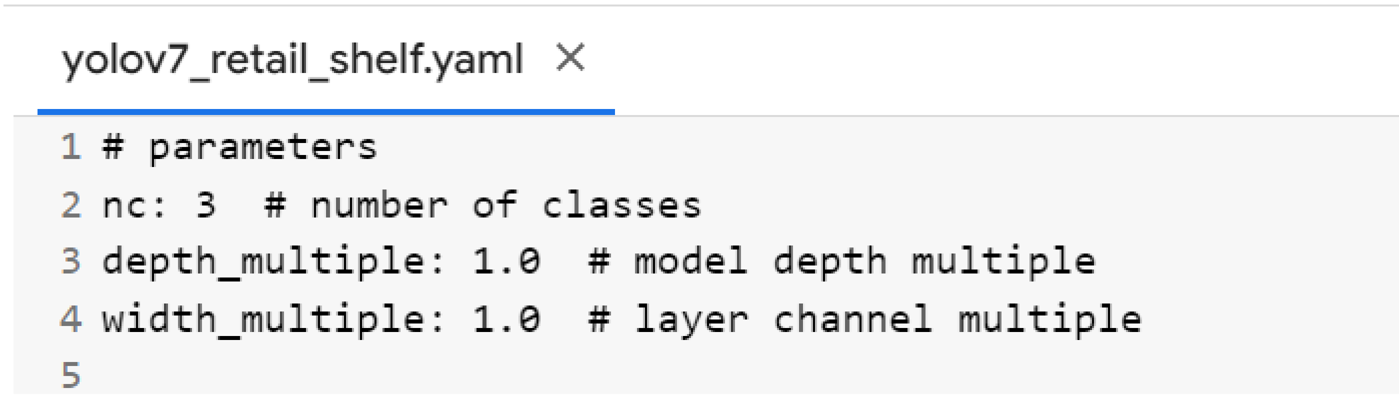
Figure 15: Extracting the dataset folder



retail_shelf.yaml

```


train: data/retailshelf_dataset/images/train # relative to path
val: data/retailshelf_dataset/images/val # relative to path
test: data/retailshelf_dataset/images/test # relative to path

# Class Names
nc: 3
names: ['AlmostEmpty', 'Empty', 'Object']

```

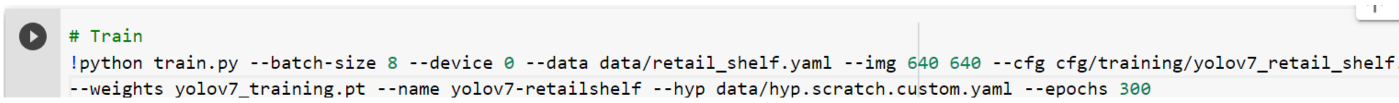
Figure 16: Changing the retailshelf.yaml file to configure model training



yolov7_retail_shelf.yaml

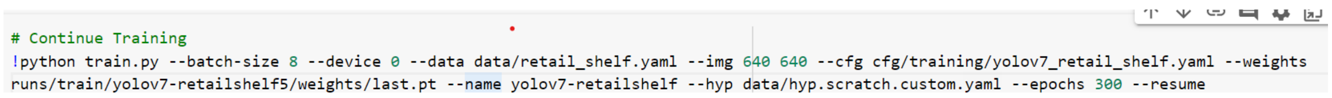
```
# parameters
nc: 3  # number of classes
depth_multiple: 1.0  # model depth multiple
width_multiple: 1.0  # layer channel multiple

```

Figure 17: Changing the configuration file



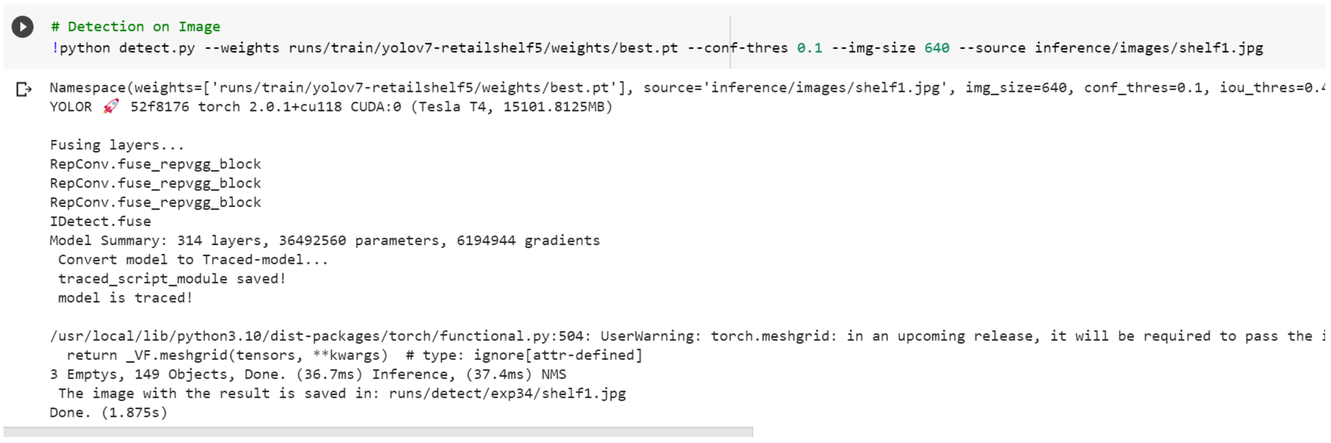
```
# Train
!python train.py --batch-size 8 --device 0 --data data/retail_shelf.yaml --img 640 640 --cfg cfg/training/yolov7_retail_shelf
--weights yolov7_training.pt --name yolov7-retailshelf --hyp data/hyp.scratch.custom.yaml --epochs 300
```

Figure 18: Model training using yolov7



```
# Continue Training
!python train.py --batch-size 8 --device 0 --data data/retail_shelf.yaml --img 640 640 --cfg cfg/training/yolov7_retail_shelf.yaml --weights
runs/train/yolov7-retailshelf5/weights/last.pt --name yolov7-retailshelf --hyp data/hyp.scratch.custom.yaml --epochs 300 --resume
```

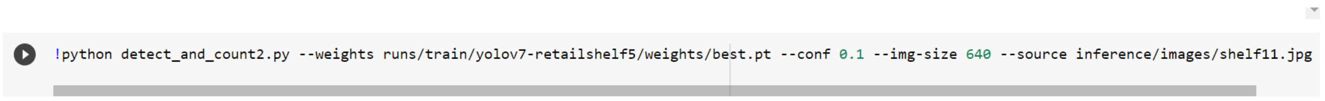
Figure 19: Continue training



```
# Detection on Image
!python detect.py --weights runs/train/yolov7-retailshelf5/weights/best.pt --conf-thres 0.1 --img-size 640 --source inference/images/shelf1.jpg
```

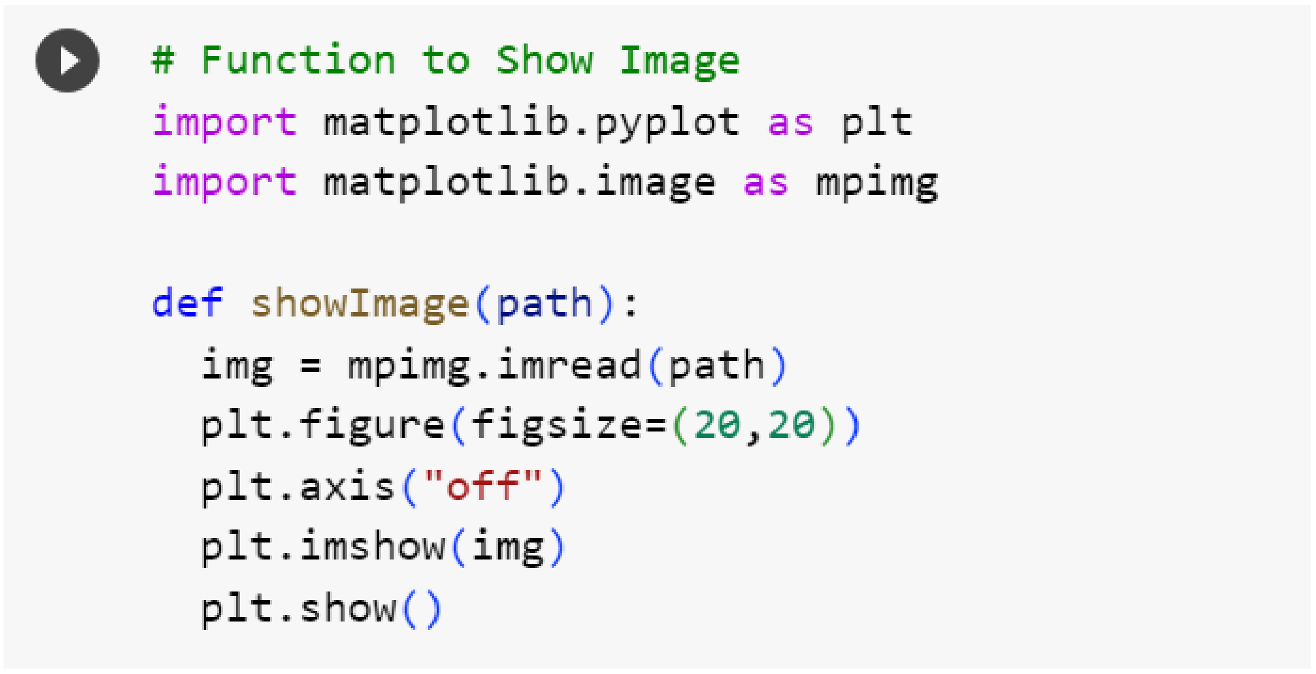
```
Namespace(weights=['runs/train/yolov7-retailshelf5/weights/best.pt'], source='inference/images/shelf1.jpg', img_size=640, conf_thres=0.1, iou_thres=0.4
YOLOR 🚀 52f8176 torch 2.0.1+cu118 CUDA:0 (Tesla T4, 15101.8125MB)

Fusing layers...
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
RepConv.fuse_repvgg_block
IDetect.fuse
Model Summary: 314 layers, 36492560 parameters, 6194944 gradients
 Convert model to Traced-model...
 traced_script_module saved!
 model is traced!

/usr/local/lib/python3.10/dist-packages/torch/functional.py:504: UserWarning: torch.meshgrid: in an upcoming release, it will be required to pass the i
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-defined]
3 Emptys, 149 Objects, Done. (36.7ms) Inference, (37.4ms) NMS
 The image with the result is saved in: runs/detect/exp34/shelf1.jpg
Done. (1.875s)
```

Figure 20: Object Detection



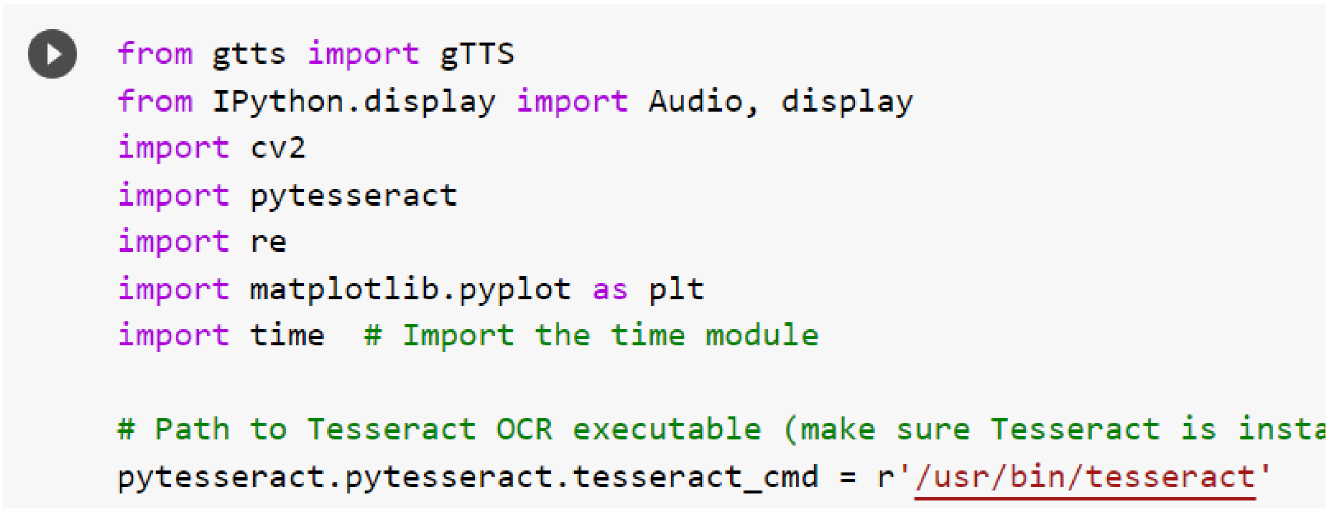
```
!python detect_and_count2.py --weights runs/train/yolov7-retailshelf5/weights/best.pt --conf 0.1 --img-size 640 --source inference/images/shelf11.jpg
```

Figure 21: Object Detection and counting



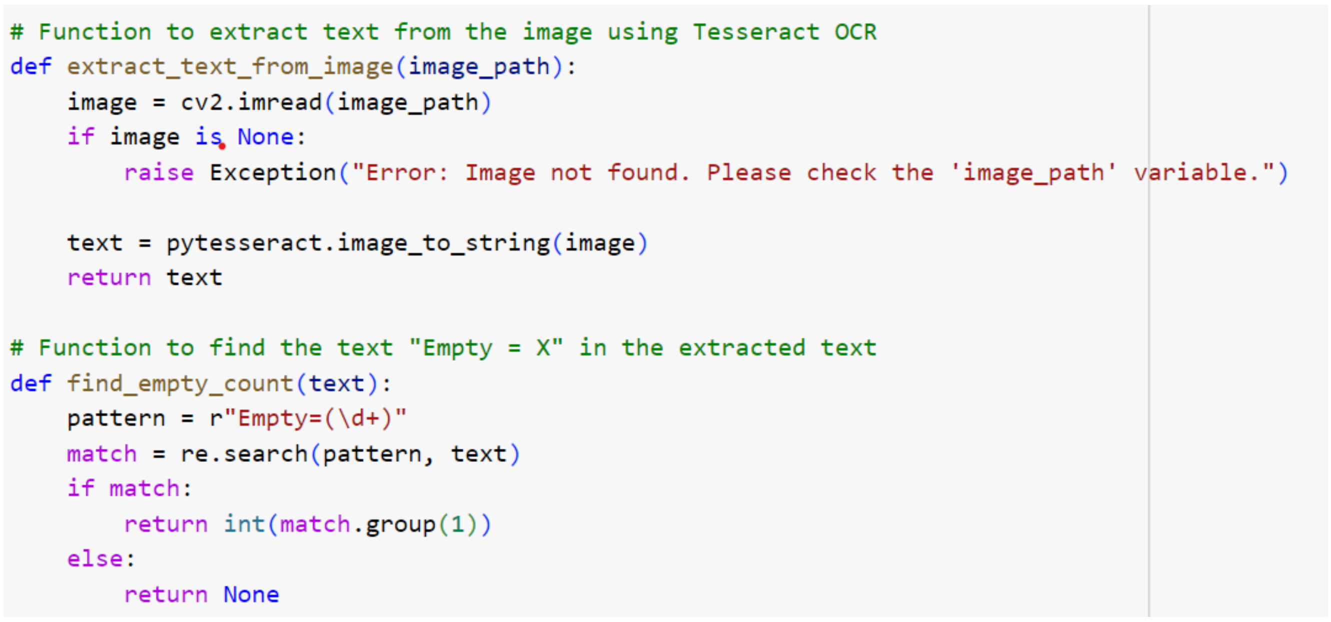
```
# Function to Show Image
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

def showImage(path):
  img = mpimg.imread(path)
  plt.figure(figsize=(20,20))
  plt.axis("off")
  plt.imshow(img)
  plt.show()
```

Figure 22: Displaying the image



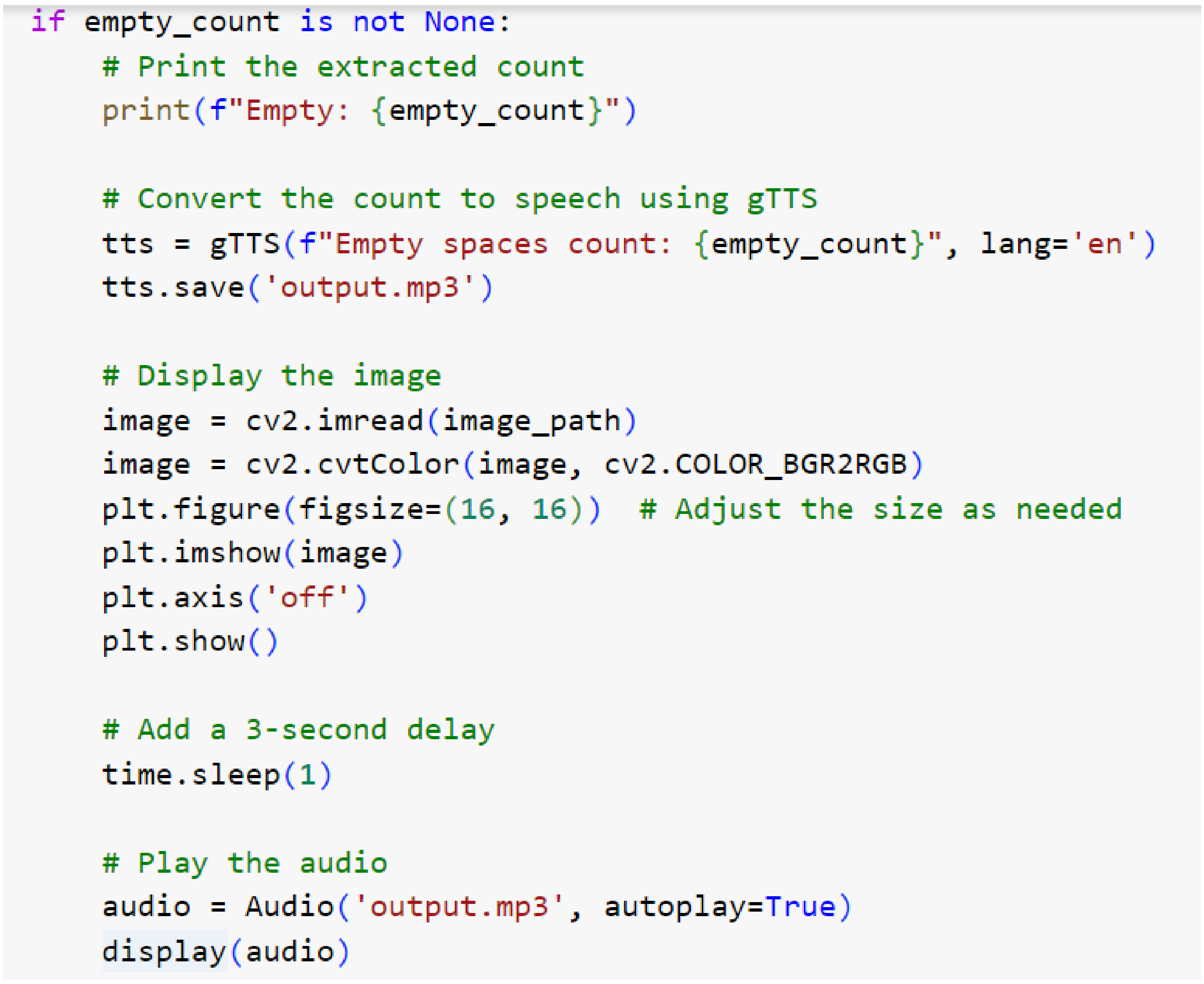
```
from gtts import gTTS
from IPython.display import Audio, display
import cv2
import pytesseract
import re
import matplotlib.pyplot as plt
import time  # Import the time module

# Path to Tesseract OCR executable (make sure Tesseract is insta
pytesseract.pytesseract.tesseract_cmd = r'/usr/bin/tesseract'
```

Figure 23: Text extarction and text to speech conversion libraries



```
# Function to extract text from the image using Tesseract OCR
def extract_text_from_image(image_path):
    image = cv2.imread(image_path)
    if image is None:
        raise Exception("Error: Image not found. Please check the 'image_path' variable.")

    text = pytesseract.image_to_string(image)
    return text

# Function to find the text "Empty = X" in the extracted text
def find_empty_count(text):
    pattern = r"Empty=(\d+)"
    match = re.search(pattern, text)
    if match:
        return int(match.group(1))
    else:
        return None
```

Figure 24: Code for extracting the count of empty spaces



```python
if empty_count is not None:
    # Print the extracted count
    print(f"Empty: {empty_count}")

    # Convert the count to speech using gTTS
    tts = gTTS(f"Empty spaces count: {empty_count}", lang='en')
    tts.save('output.mp3')

    # Display the image
    image = cv2.imread(image_path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    plt.figure(figsize=(16, 16))  # Adjust the size as needed
    plt.imshow(image)
    plt.axis('off')
    plt.show()

    # Add a 3-second delay
    time.sleep(1)

    # Play the audio
    audio = Audio('output.mp3', autoplay=True)
    display(audio)
```

Figure 25: Converting the extracted text to audio