

Digital Food Experiencing using AR/VR Report

Team members:

Arun Raghav S

Anurag Kumar Bharti

Kashvi Jain

Object detection using Tensorflow

Tensorflow: It is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but focuses on training and inference of deep neural networks.

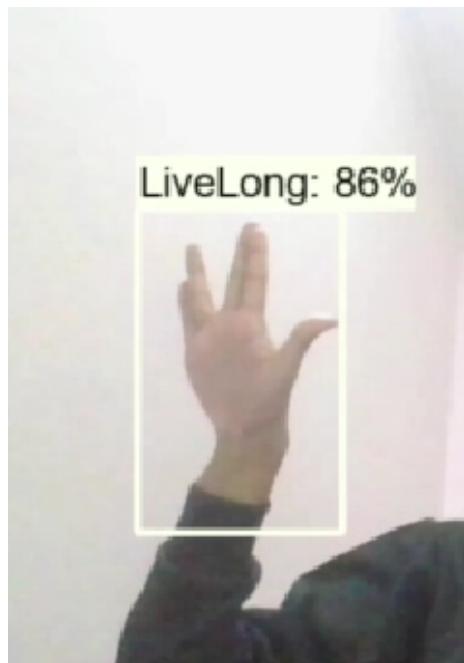
Learning Curve:

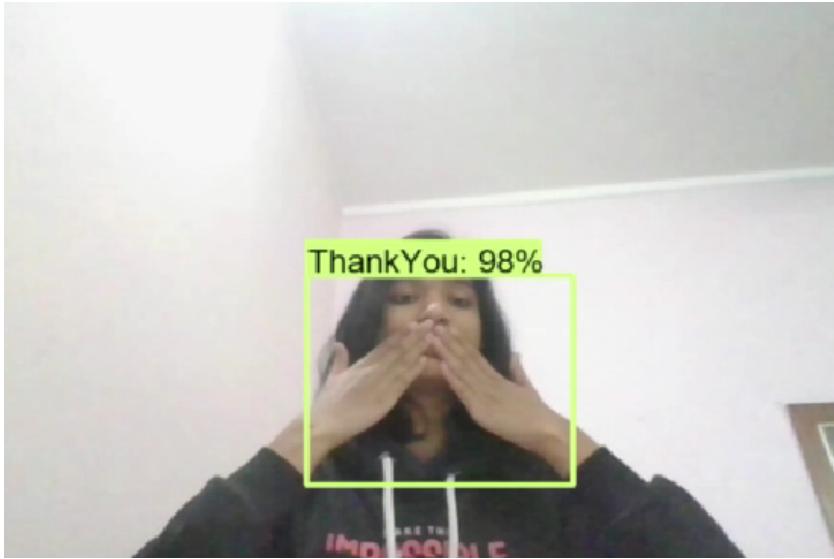
We followed the course on object detection on [youtube](#) and trained our own model to detect different hand gestures in images and real-time using a webcam.

Followed the steps provided by [Github Repository](#).

The model was trained to detect four classes of hand gestures: Thumbs Up, Thumbs down, Thank You, and Livelong.

The model could then detect hand gestures in real time. Here are some screenshots of the same:





Here is the video link for the real-time detection: [link](#)

But this model didn't work well when we trained it for different food models as it identified objects incorrectly. So, we used a pre-trained model on the coco dataset.

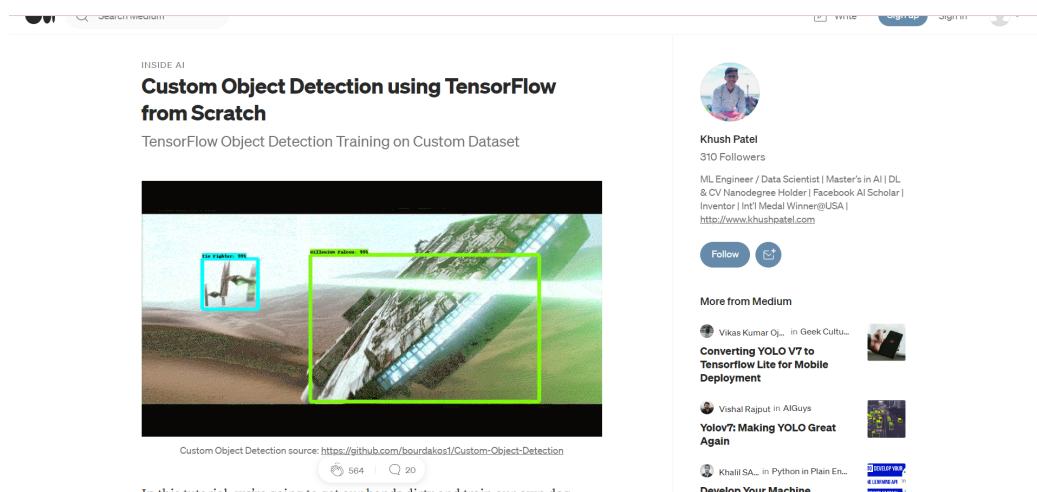
COCO-DATASET

COCO stands for Common Objects in Context, as the image dataset was created to advance image recognition. The COCO dataset contains challenging, high-quality visual datasets for computer vision, primarily state-of-the-art neural networks. For example, COCO is often used to benchmark algorithms to compare the performance of real-time object detection. The format of the COCO dataset is automatically interpreted by advanced neural network libraries. The COCO dataset classes for object detection and tracking include the following pre-trained 80 objects:

```
'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck',  
'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parking meter', 'bench',  
'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra',  
'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee',  
'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove',  
'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork',  
'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli',  
'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch', 'potted plant',  
'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard',  
'cell phone', 'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book',  
'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush'
```

We then worked on three different models.

1) First model



LINK:-<https://towardsdatascience.com/custom-object-detection-using-tensorflow-from-scratch-e61da2e10087>

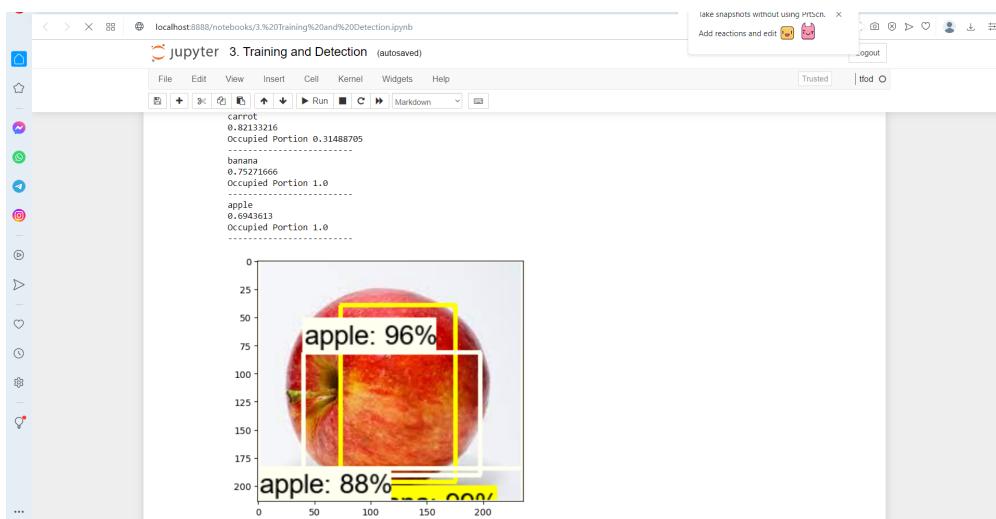
In this model, `ssd_mobilenet_v2_coco`, the object was detected with low accuracy and fluctuated labeling (i.e other objects were also detected with the specified object). We also tried to rectify the problem by increasing the number of steps and training the model with more images, but there were no changes in the output.

localhost:8888/notebooks/3.%20Training%20and%20Detection.ipynb

jupyter 3. Training and Detection (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted tfod

```
In [2]: import os
In [3]: CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_320x320_coco17_tpu-8.tar.gz'
# PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/efficientdet_d0_coco17_tpu-32.tar.gz'
# PRETRAINED_MODEL_URL = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v1_fpn_640x640_coco17_tar'
LABEL_MAP_NAME = 'label_map.pbtxt'
In [4]: PRETRAINED_MODEL_URL
Out[4]: 'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_320x320_coco17_tpu-8.tar.gz'
In [5]: paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'SCRIPTS_PATH': os.path.join('Tensorflow', 'scripts'),
    'APIMODEL_PATH': os.path.join('Tensorflow', 'models'),
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace', 'annotations'),
    'IMAGE_PATH': os.path.join('Tensorflow', 'workspace', 'images'),
    'MODEL_PATH': os.path.join('Tensorflow', 'workspace', 'models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace', 'pre-trained-models'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME, 'export'),
    'TFLITE_PATH': os.path.join('Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME, 'tfjsexport'),
    'TFLITE_PATH': os.path.join('Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME, 'tfliteexport'),
    'PROTOC_PATH': os.path.join('Tensorflow', 'protoc')
}
In [6]: files = {
    'PIPELINE_CONFIG': os.path.join('Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME, 'pipeline.config'),
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}
```



2) Second model

Real-time Object Detection using SSD MobileNet V2 on Video Streams

An easy workflow for implementing pre-trained object detection architectures on video streams



Photo by Christopher Burns on Unsplash

169 1



Odemakinde Elisha
105 Followers
Powering the next generation of AI solutions in the African Ecosystem.

Follow  

More from Medium

 Vikas Kumar Ojha in Geek Cultur...
Converting YOLO V7 to Tensorflow Lite for Mobile Deployment

 Vishal Rajput in AIGuys
YOLOv7: Making YOLO Great Again

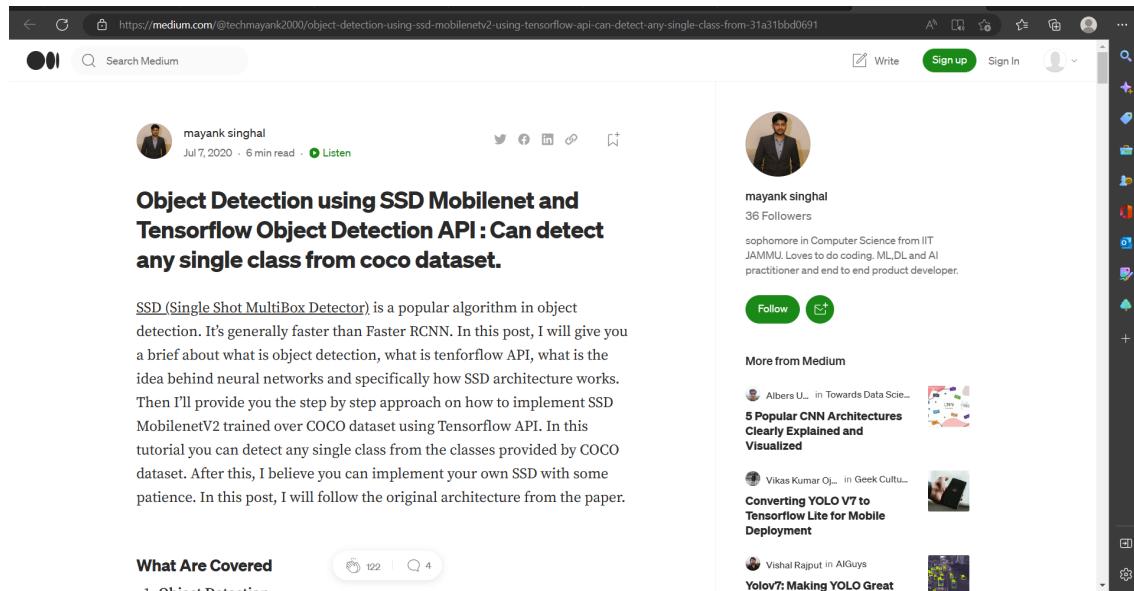
 Albers Uebel in Towards Data Scie...
5 Popular CNN Architectures Clearly Explained and Visualized

 Remy Villalobos in Level Up Coding
Face recognition with OpenCV

LINK:-<https://heartbeat.comet.ml/real-time-object-detection-using-ssd-mobilenet-v2-on-video-streams-3bfc1577399c>

We were not able to use this model because we were getting a particular error which couldn't be resolved despite all our efforts. A screenshot of the error faced is attached below:

3) Third model



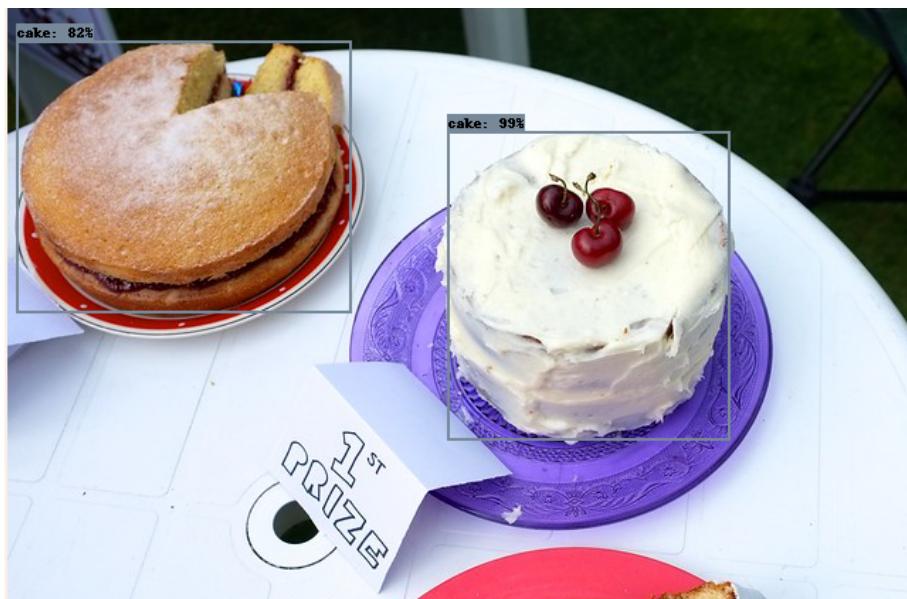
SSD (Single Shot MultiBox Detector) is a popular algorithm in object detection. It's generally faster than Faster RCNN. In this post, I will give you a brief about what is object detection, what is tensorflow API, what is the idea behind neural networks and specifically how SSD architecture works. Then I'll provide you the step by step approach on how to implement SSD MobilenetV2 trained over COCO dataset using Tensorflow API. In this tutorial you can detect any single class from the classes provided by COCO dataset. After this, I believe you can implement your own SSD with some patience. In this post, I will follow the original architecture from the paper.

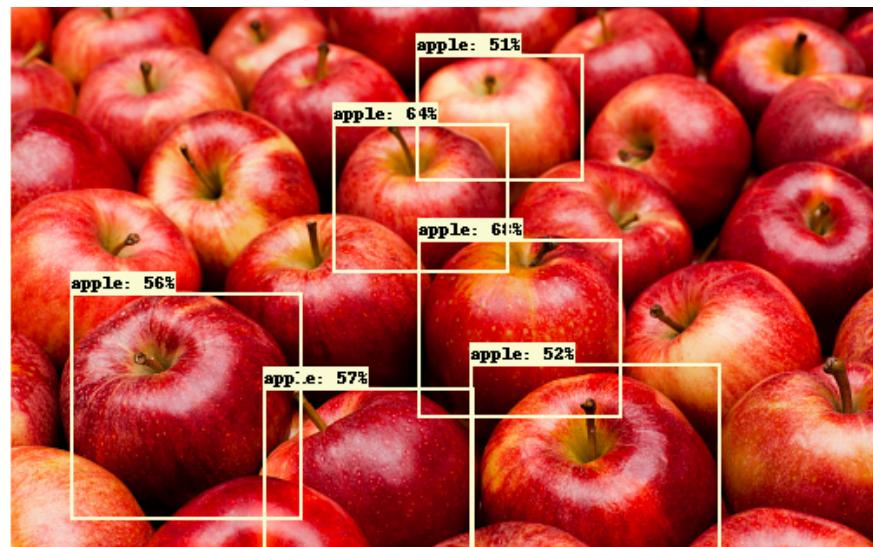
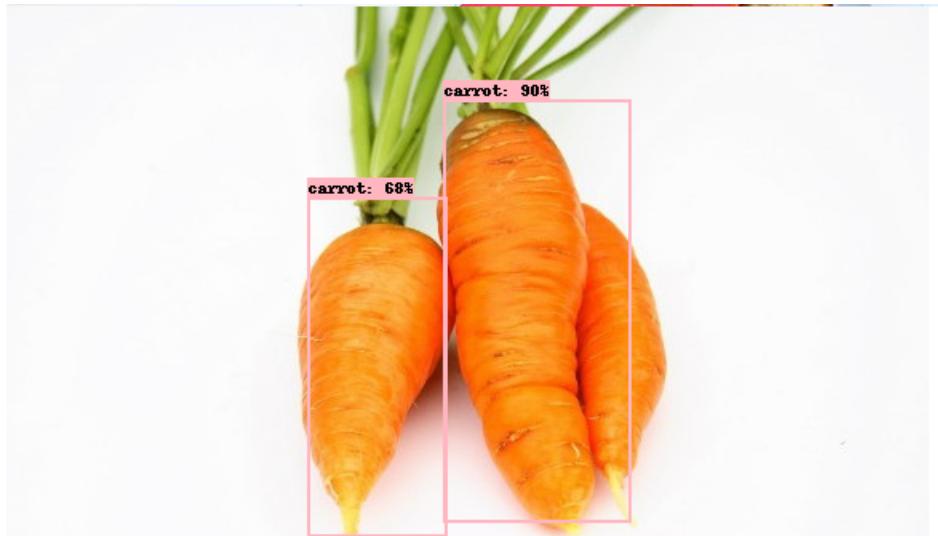
What Are Covered

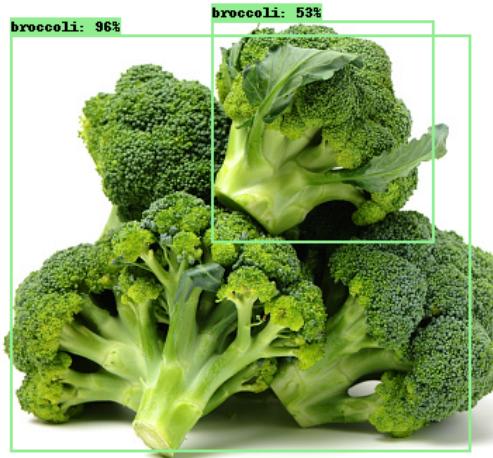
- 1. Overview
- 2. What is Object Detection
- 3. What is TensorFlow API
- 4. What is the idea behind neural networks
- 5. How SSD architecture works
- 6. Step by step approach to implement SSD
- 7. Conclusion

[Object Detection using SSD Mobilenet and Tensorflow Object Detection API : Can detect any single class from coco dataset. | by mayank singhal | Medium](https://medium.com/@techmayank2000/object-detection-using-ssd-mobilenet2-using-tensorflow-api-can-detect-any-single-class-from-coco-dataset-31a31bb0691)

Out of all three models, the third one worked fine and detected objects with high accuracy. So, we decided to proceed with that. We've attached screenshots of the image detection of different food items present in the coco dataset by the model. We detected the objects like cake, apple, carrots, bananas, and broccoli as shown below.



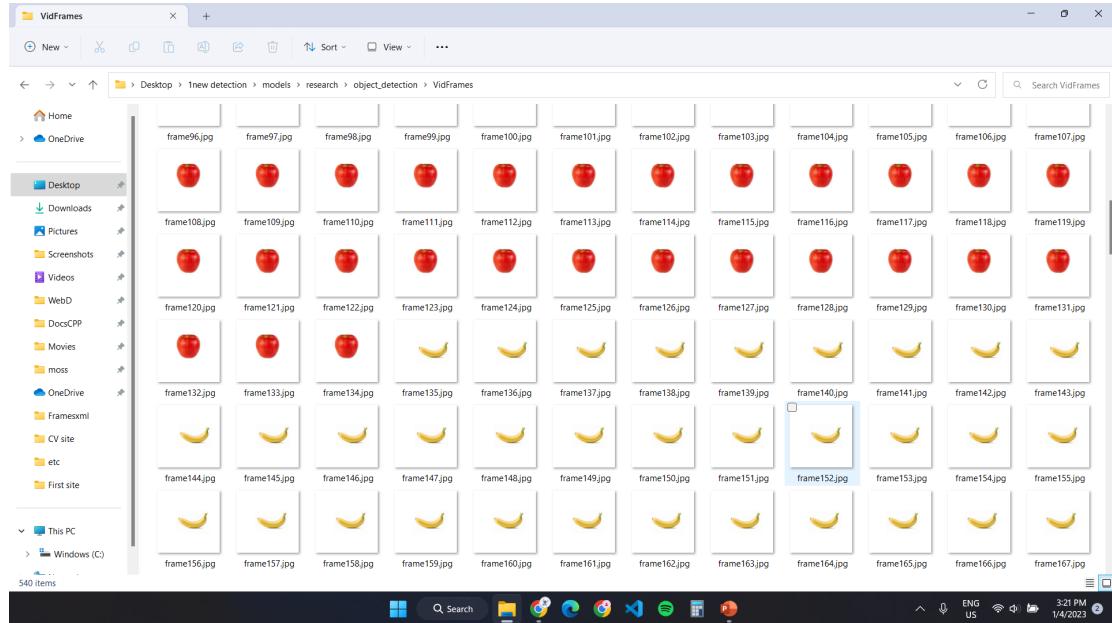




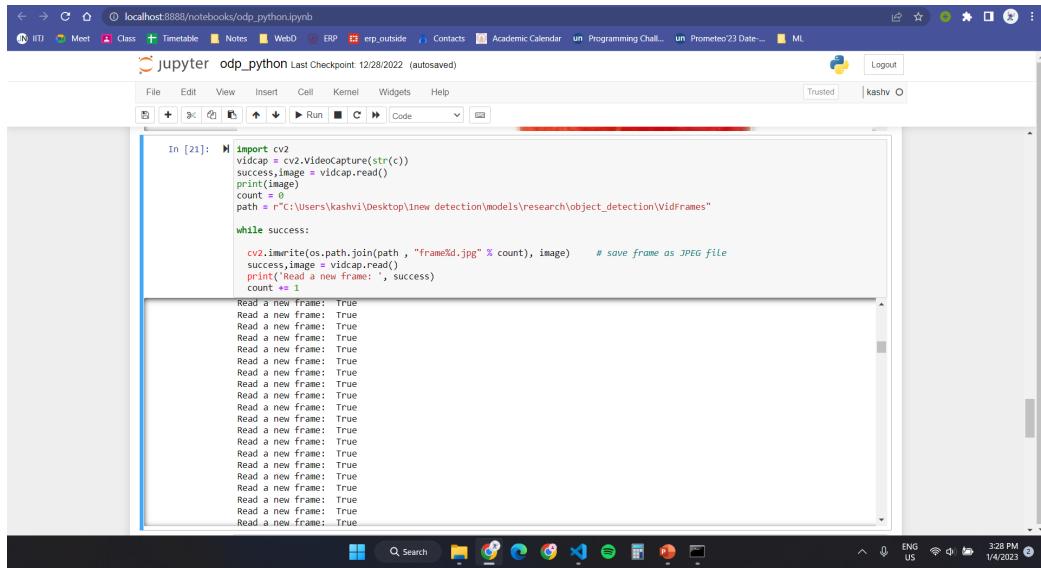
Object detection in video

We then wrote the code for saving frames from the video in jpg format. Then, using the code for XML generation for images, we generated the XML files for video frames.

Below are the attached screenshots showing the frames of the videos saved in jpg format and the XML files generated for the frames:



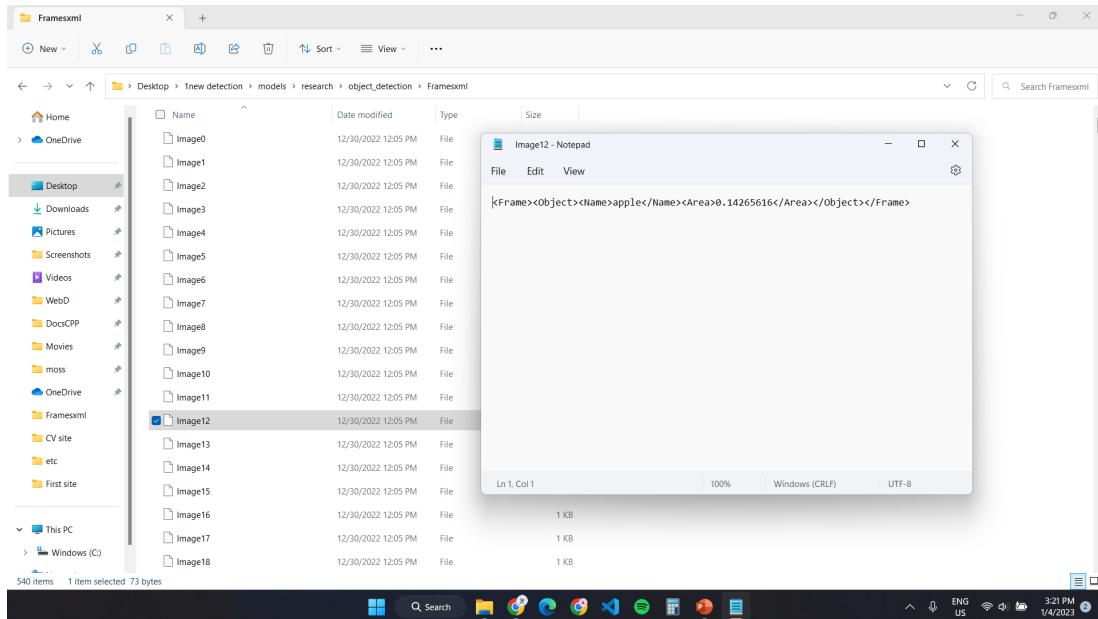
And here is a screenshot of the code:



```
In [21]: import cv2
vidcap = cv2.VideoCapture(str(c))
success,image = vidcap.read()
print(image)
count = 0
path = "C:\Users\kashvi\Desktop\1new detection\models\research\object_detection\VidFrames"

while success:
    cv2.imwrite(os.path.join(path , "frame%d.jpg" % count), image)    # save frame as JPEG file
    success,image = vidcap.read()
    print('Read a new frame: ', success)
    count += 1
```

xml generated:



Thank You