UMKC Project Report:

Smart Shopping Check-out Using Deep Learning Methods

Department of Computer science University of Missouri Kansas City

COMP-SCI-5542-0001-45477-2021-Big Data Analytics & Apps

Prof. Syed Jawad Shah

GROUP SUBMISSION:

Team number:17
Team Members Names:
Haritha Thumukuntla - htmrc@umsystem.edu
Shambhavi Sompally - ss472@umsystem.edu
SaiDurga Maheshwari Kanuganti - skbfz@umsystem.edu
Sandeep Reddy chevula - scv7n@umsyste.edu

Domain of the project

Standing in long queues and waiting for checkout in any store has become a very big problem in our daily busy life schedule. As technology is also booming every day, we came up with an idea to implement this shopping checkout system in a smart way by detecting the object or the items present in any store. Object detection is an important, yet challenging vision task. It is a critical part of many applications such as image search, image auto-annotation, scene understanding, and object tracking. Object localization refers to identifying the location of one or more objects in an image and drawing a bounding box around their extent. Object detection does work by combining these two tasks and localizes and classifying one or more objects in an image.

Introduction

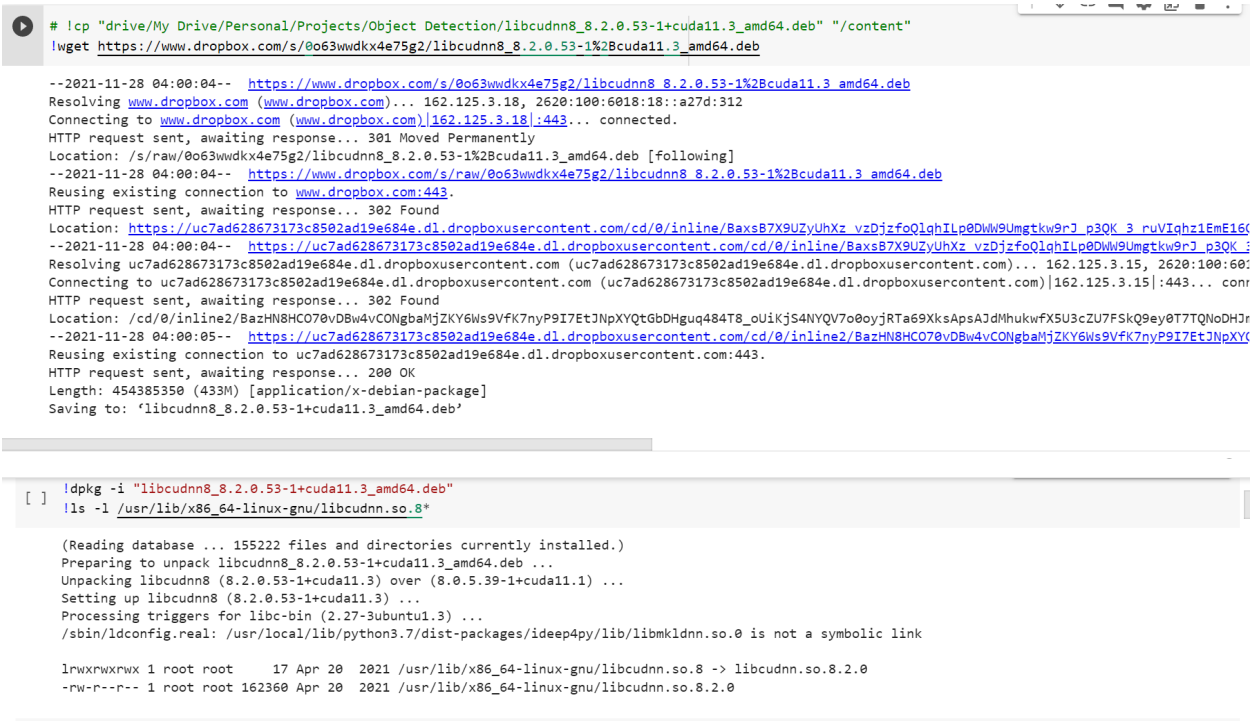
The goal of product identification is to make product administration easier and to improve the shopping experience for customers. At the moment, barcode recognition is the most extensively utilized technique, not only in research but also in industries that use automatic goods identification. The existing problem is Product management may be enhanced by scanning barcode markings on each product packaging. Almost every item on the market, in most cases, has a corresponding barcode. However, owing to the ambiguity of the barcode's printing location, it frequently takes time to manually locate the barcode and aid the machine in detecting the barcode at the checkout counter. Recognizing items in order to undertake goods supervision needed a vast quantity of manual effort and a huge percentage of the effort.

Product detection is a problem that needs the detection, identification, and classification of an object or entity. In order to complete this task, we must first use our machine learning model to determine whether or not an object of interest is present in the image. If the object(s) in the image is present, create a bounding box around them. Finally, the model must categorize the item represented by the bounding box. This work needs quick object detection in order to be implemented in real-time. One of its most important uses is in real-time object detection. Here we are connecting the Dataset from the Dropbox that we have created for the product to be detected. The data set contains the women’s footwear, women’s shirts, women’s pants similarly to the men. Here we would like to read the Dataset which is in Dropbox. The dataset which is in dropbox is in a zip file, We need to unzip the file. Here in this project, we took 600 images and we label them manually by installing python 3.8.10, and selecting the dataset folder the in command prompt we have to run labelImg then we get a window there we have labeled the images and defined the boundary boxing width, height it created the XML file with all the widths and height of the image.

Implementation

Here for this particular project, we used the tensor flow model for object detection. The TensorFlow Object Detection API is an open-source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. Next, we have installed python 3.8.10 and run the labeling command to label the images.

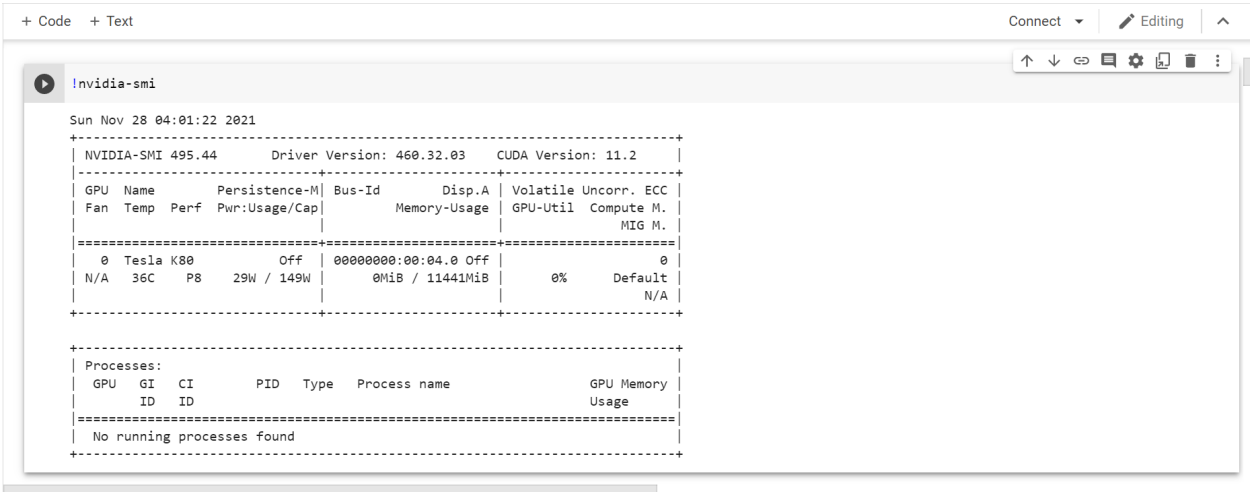
Before running the actual code Firstly we have to install CuDNN which is used by deep learning researchers and framework developers all around the world for high-performance GPU acceleration. It enables them to devote more time to training neural networks and designing software applications rather than low-level GPU performance adjustments.



```
# !cp "drive/My Drive/Personal/Projects/Object Detection/libcudnn8_8.2.0.53-1+cuda11.3_amd64.deb" "/content"
!wget https://www.dropbox.com/s/0o63wwdkx4e75g2/libcudnn8_8.2.0.53-1%2Bcuda11.3_amd64.deb

--2021-11-28 04:00:04--  https://www.dropbox.com/s/0o63wwdkx4e75g2/libcudnn8_8.2.0.53-1%2Bcuda11.3_amd64.deb
Resolving www.dropbox.com (www.dropbox.com)... 162.125.3.18, 2620:100:6018:18::a27d:312
Connecting to www.dropbox.com (www.dropbox.com)|162.125.3.18|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/0o63wwdkx4e75g2/libcudnn8_8.2.0.53-1%2Bcuda11.3_amd64.deb [following]
--2021-11-28 04:00:04--  https://www.dropbox.com/s/raw/0o63wwdkx4e75g2/libcudnn8_8.2.0.53-1%2Bcuda11.3_amd64.deb
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc7ad628673173c8502ad19e684e.dl.dropboxusercontent.com/cd/0/inline/BaxsB7X9UZyUhXz_vzDjzfoQlqhILp0DWW9Umgtkw9rJ_p3QK_3_ruVIqhz1EmE16(
--2021-11-28 04:00:04--  https://uc7ad628673173c8502ad19e684e.dl.dropboxusercontent.com/cd/0/inline/BaxsB7X9UZyUhXz_vzDjzfoQlqhILp0DWW9Umgtkw9rJ_p3QK_3
Resolving uc7ad628673173c8502ad19e684e.dl.dropboxusercontent.com (uc7ad628673173c8502ad19e684e.dl.dropboxusercontent.com)... 162.125.3.15, 2620:100:601
Connecting to uc7ad628673173c8502ad19e684e.dl.dropboxusercontent.com (uc7ad628673173c8502ad19e684e.dl.dropboxusercontent.com)|162.125.3.15|:443... conn
HTTP request sent, awaiting response... 302 Found
Location: /cd/0/inline2/BazHN8HCO70vDBw4vCONgbaMjZKY6Ws9VfK7nyP9I7EtJNpXYQtGbDHguq484T8_oUiKjS4NYQV7o0oyjRTa69XksApsAJdMhukwfX5U3cZU7FSkQ9ey0T7TQNoDHJr
--2021-11-28 04:00:05--  https://uc7ad628673173c8502ad19e684e.dl.dropboxusercontent.com/cd/0/inline2/BazHN8HCO70vDBw4vCONgbaMjZKY6Ws9VfK7nyP9I7EtJNpXY(
Reusing existing connection to uc7ad628673173c8502ad19e684e.dl.dropboxusercontent.com:443.
HTTP request sent, awaiting response... 200 OK
Length: 454385350 (433M) [application/x-debian-package]
Saving to: 'libcudnn8_8.2.0.53-1+cuda11.3_amd64.deb'
```

```
[ ] !dpkg -i "libcudnn8_8.2.0.53-1+cuda11.3_amd64.deb"
    !ls -l /usr/lib/x86_64-linux-gnu/libcudnn.so.8*

(Reading database ... 155222 files and directories currently installed.)
Preparing to unpack libcudnn8_8.2.0.53-1+cuda11.3_amd64.deb ...
Unpacking libcudnn8 (8.2.0.53-1+cuda11.3) over (8.0.5.39-1+cuda11.1) ...
Setting up libcudnn8 (8.2.0.53-1+cuda11.3) ...
Processing triggers for libc-bin (2.27-3ubuntu1.3) ...
/sbin/ldconfig.real: /usr/local/lib/python3.7/dist-packages/ideep4py/lib/libmkldnn.so.0 is not a symbolic link

lrwxrwxrwx 1 root root     17 Apr 20  2021 /usr/lib/x86_64-linux-gnu/libcudnn.so.8 -> libcudnn.so.8.2.0
-rw-r--r-- 1 root root 162360 Apr 20  2021 /usr/lib/x86_64-linux-gnu/libcudnn.so.8.2.0
```

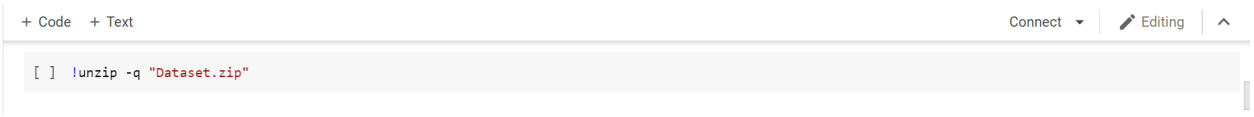
The NVIDIA A100 enables enterprise AI and deep learning acceleration. To address complicated computer difficulties, the GPU includes high-performance computing (HPC), increased acceleration, and data analytics. Because of its tremendous power, it can scale up to hundreds of GPUs and distribute the workload over numerous instances.



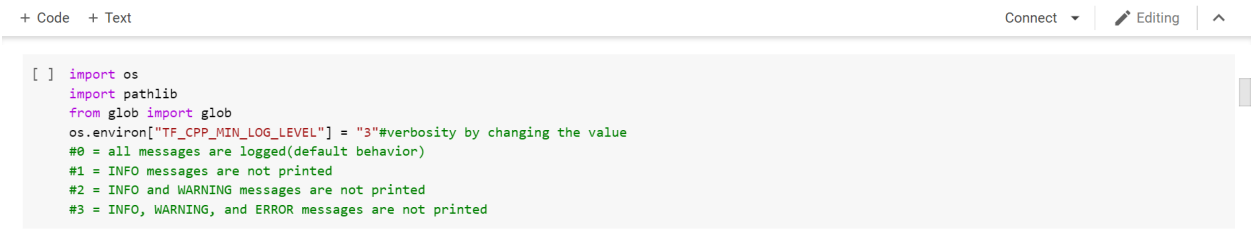
```
+ Code + Text                                                    Connect  Editing

!nvidia-smi

Sun Nov 28 04:01:22 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.44       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   36C    P8    29W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Then we call the dataset which is uploaded in the dropbox followed by unzipping the dataset.

```
+ Code + Text                                                    Connect  Editing

[ ] !wget https://www.dropbox.com/s/bj742qjgnp3zhh2/Dataset.zip

--2021-11-28 04:01:25--  https://www.dropbox.com/s/bj742qjgnp3zhh2/Dataset.zip
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.18, 2620:100:601b:18::a27d:812
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.18|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/bj742qjgnp3zhh2/Dataset.zip [following]
--2021-11-28 04:01:25--  https://www.dropbox.com/s/raw/bj742qjgnp3zhh2/Dataset.zip
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location: https://uc5dcb3a8027b2ffad45327170e1.dl.dropboxusercontent.com/cd/0/inline/BazEhVm-Zl4owQ0ZfutEfQY4lb0jM3gTQdUOQ6Wc0W2mU9xUMfBXE50m56-HO7cnyj
--2021-11-28 04:01:25--  https://uc5dcb3a8027b2ffad45327170e1.dl.dropboxusercontent.com/cd/0/inline/BazEhVm-Zl4owQ0ZfutEfQY4lb0jM3gTQdUOQ6Wc0W2mU9xUMfB
Resolving uc5dcb3a8027b2ffad45327170e1.dl.dropboxusercontent.com (uc5dcb3a8027b2ffad45327170e1.dl.dropboxusercontent.com)... 162.125.3.15, 2620:100:601
Connecting to uc5dcb3a8027b2ffad45327170e1.dl.dropboxusercontent.com (uc5dcb3a8027b2ffad45327170e1.dl.dropboxusercontent.com)|162.125.3.15|:443... conn
HTTP request sent, awaiting response... 302 Found
Location: /cd/0/inline2/Baz2z3b2eC3ltHPfDPHh97TsPkeMx6nQpUqwTWPtjlBvDnCvttSKwXuxCUGTZGx1cV20wVtLCg12TNdRr4mpJUICwNjKdpve01es_fO-ilZy5E-RHIdvMmlnNEbRXWE
--2021-11-28 04:01:25--  https://uc5dcb3a8027b2ffad45327170e1.dl.dropboxusercontent.com/cd/0/inline2/Baz2z3b2eC3ltHPfDPHh97TsPkeMx6nQpUqwTWPtjlBvDnCvtt
Reusing existing connection to uc5dcb3a8027b2ffad45327170e1.dl.dropboxusercontent.com:443.
HTTP request sent, awaiting response... 200 OK
Length: 104510473 (100M) [application/zip]
Saving to: 'Dataset.zip'

Dataset.zip         100%[===================>]  99.67M  87.0MB/s    in 1.1s
```

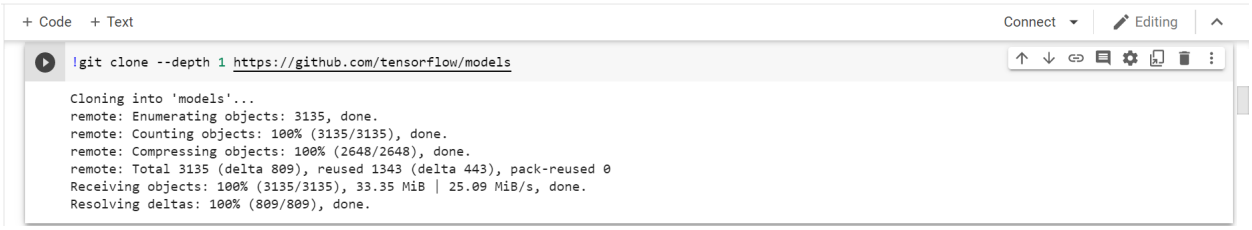


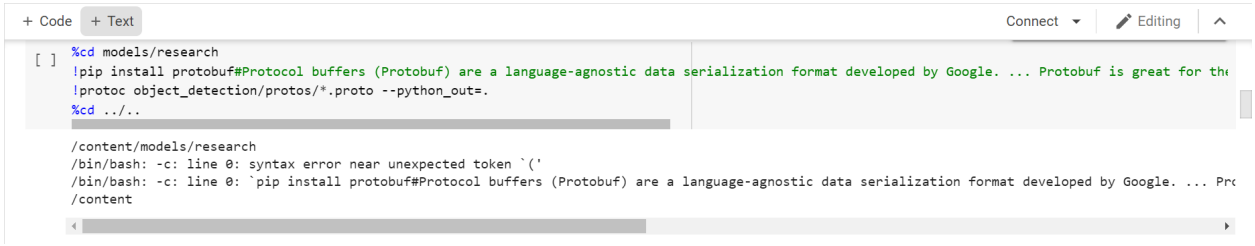
```
+ Code + Text                                                    Connect  Editing

[ ] !unzip -q "Dataset.zip"
```

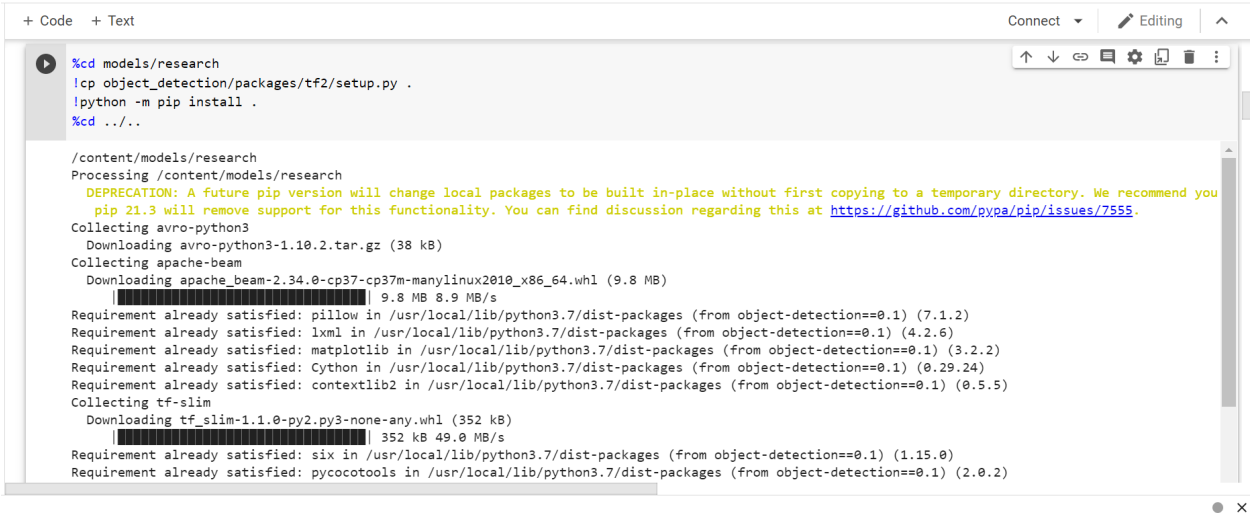


```
+ Code + Text                                                    Connect  Editing

[ ] import os
    import pathlib
    from glob import glob
    os.environ["TF_CPP_MIN_LOG_LEVEL"] = "3"#verbosity by changing the value
    #0 = all messages are logged(default behavior)
    #1 = INFO messages are not printed
    #2 = INFO and WARNING messages are not printed
    #3 = INFO, WARNING, and ERROR messages are not printed
```

Imported the required libraries and installed protocol buffers to allow serialization and deserialization of structured data and also to provide a better way, compared to XML, to make systems communicate.







Imported all the required libraries for this project to detect the object

```
[ ] import numpy as np
    import os
    import sys
    import tarfile
    import tensorflow as tf
    import zipfile
    import cv2
    import json
    import pandas as pd
    import glob
    import os.path as osp
    from pathlib import Path
    import datetime
    import random
    import shutil
    from io import StringIO, BytesIO
    from PIL import Image
    from IPython.display import display
    import re
```

```
[ ] %matplotlib inline
    import matplotlib.pyplot as plt
    import seaborn as sns
    from matplotlib import rcParams
    sns.set(rc={"font.size":9,"axes.titlesize":15,"axes.labelsize":9,
                "axes.titlepad":11, "axes.labelpad":9, "legend.fontsize":7,
                "legend.title_fontsize":7, 'axes.grid' : False})

    from sklearn.model_selection import train_test_split

    ## Import object detection module
    from object_detection.utils import ops as utils_ops
    from object_detection.utils import label_map_util
    from object_detection.utils import visualization_utils as vis_utils
    from object_detection.protos.string_int_label_map_pb2 import StringIntLabelMap, StringIntLabelMapItem
    from object_detection.utils import config_util
    from object_detection.builders import model_builder
```

We have given the path as shown in the screenshot below:

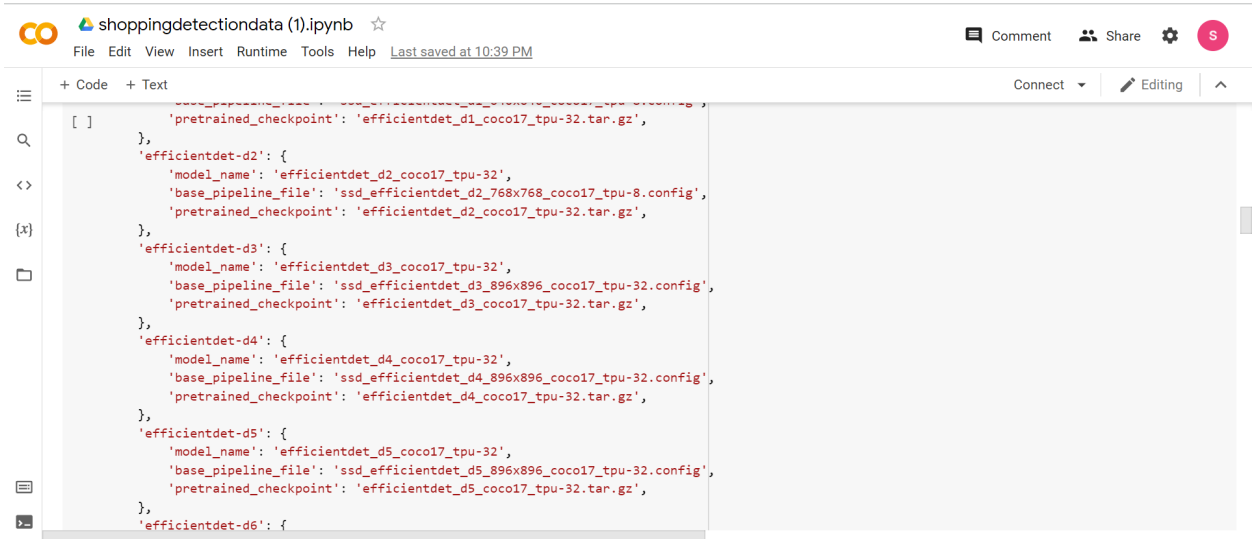
```
[ ] PATH_TO_LABELS = 'models/research/object_detection/data/mscoco_label_map.pbtxt'
    category_index = label_map_util.create_category_index_from_labelmap(PATH_TO_LABELS, use_display_name=True)
```

```
[ ] pretrained_dir = "training_job/model/"

    if not os.path.exists(pretrained_dir):
        os.makedirs(pretrained_dir)
        print('Pretrainined Model Directory:', pretrained_dir)
```

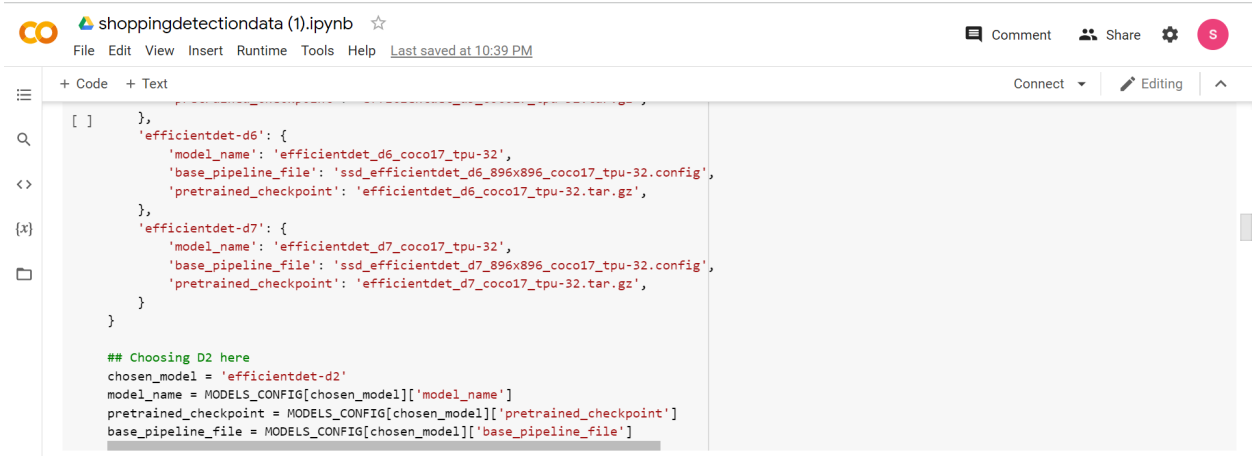
We have taken the pre-trained model here and used efficeintdet configuration.

Efficeintdet configuration:

EfficientDet is a type of object detection model, which utilizes several optimizations and backbone tweaks, such as the use of a BiFPN, and a compound scaling method that uniformly scales the resolution, depth, and width for all backbones, feature networks, and box/class prediction networks at the same time.

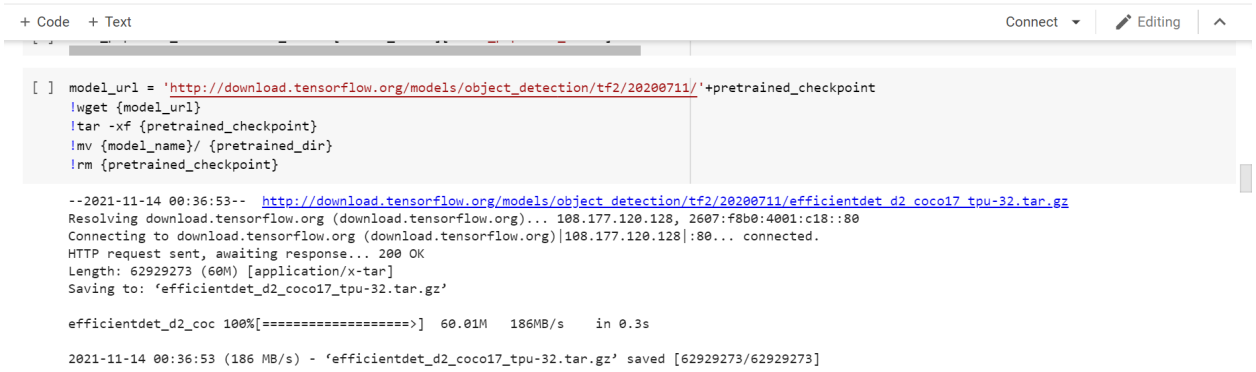


We have used the model d2 for training purposes.



Checkpoints capture the exact value of all parameters used by a model.

Checkpoints do not contain any description of the computation defined by the model and thus are typically only useful when source code that will use the saved parameter values is available.



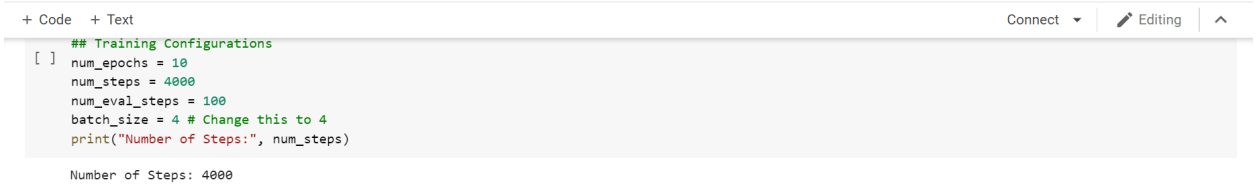
```
+ Code  + Text                                                    Connect  Editing

[ ] model_url = 'http://download.tensorflow.org/models/object_detection/tf2/20200711/'+pretrained_checkpoint
    !wget {model_url}
    !tar -xf {pretrained_checkpoint}
    !mv {model_name}/ {pretrained_dir}
    !rm {pretrained_checkpoint}

    --2021-11-14 00:36:53--  http://download.tensorflow.org/models/object_detection/tf2/20200711/efficientdet_d2_coco17_tpu-32.tar.gz
    Resolving download.tensorflow.org (download.tensorflow.org)... 108.177.120.128, 2607:f8b0:4001:c18::80
    Connecting to download.tensorflow.org (download.tensorflow.org)|108.177.120.128|:80... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 62929273 (60M) [application/x-tar]
    Saving to: 'efficientdet_d2_coco17_tpu-32.tar.gz'

    efficientdet_d2_coc 100%[===================>]  60.01M   186MB/s    in 0.3s

    2021-11-14 00:36:53 (186 MB/s) - 'efficientdet_d2_coco17_tpu-32.tar.gz' saved [62929273/62929273]
```

Here we are labeling the images with Men pants, Men shirts, Men Footwear, Women's pants, Women's shirts, Women footwear and defining there Id's

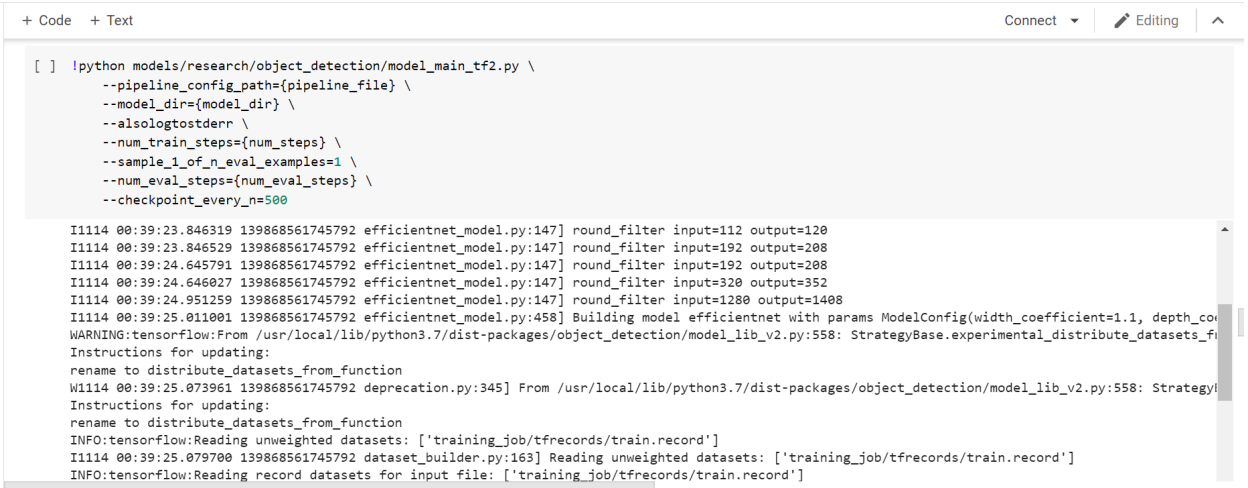
```
+ Code  + Text                                                    Connect  Editing

[ ] def convert_classes(classes, start=1):
        msg = StringIntLabelMap()
        for id, name in enumerate(classes, start=start):
            msg.item.append(StringIntLabelMapItem(id=id, name=name))
        text = str(text_format.MessageToBytes(msg, as_utf8=True), 'utf-8')
        return text

    labels =  [
                "BoysShirt",
                "GirlsShirt",
                "BoysPant",
                "GirlsPant",
                "MenFootWear",
                "WomenFootWear"
                ]

    txt = convert_classes(labels)
    print(txt)
    with open('labelmap.pbtxt', 'w') as f:
        f.write(txt)
```

```
[ ] item {
      name: "BoysShirt"
      id: 1
    }
    item {
      name: "GirlsShirt"
      id: 2
    }
    item {
      name: "BoysPant"
      id: 3
    }
    item {
      name: "GirlsPant"
      id: 4
    }
    item {
      name: "MenFootWear"
      id: 5
    }
    item {
      name: "WomenFootWear"
      id: 6
    }
```

Here we are training the data with epoch 10 and batch size 4 for each elevation step is 100 it will train up to 4000 steps.

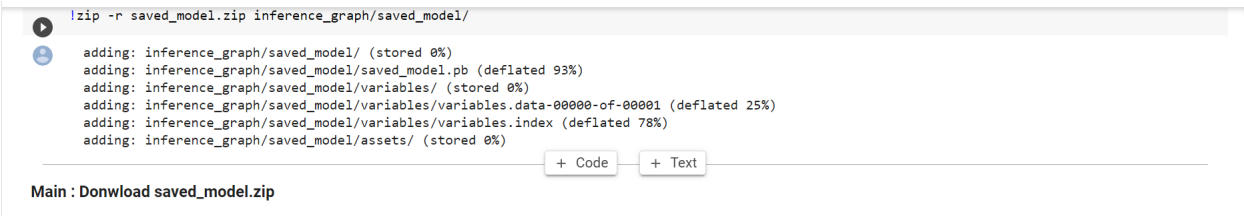


```
+ Code  + Text                                                    Connect  Editing

[ ]  ## Training Configurations
     num_epochs = 10
     num_steps = 4000
     num_eval_steps = 100
     batch_size = 4 # Change this to 4
     print("Number of Steps:", num_steps)

     Number of Steps: 4000
```



```
+ Code  + Text                                                    Connect  Editing

[ ]  !python models/research/object_detection/model_main_tf2.py \
         --pipeline_config_path={pipeline_file} \
         --model_dir={model_dir} \
         --alsologtostderr \
         --num_train_steps={num_steps} \
         --sample_1_of_n_eval_examples=1 \
         --num_eval_steps={num_eval_steps} \
         --checkpoint_every_n=500

     I1114 00:39:23.846319 139868561745792 efficientnet_model.py:147] round_filter input=112 output=120
     I1114 00:39:23.846529 139868561745792 efficientnet_model.py:147] round_filter input=192 output=208
     I1114 00:39:24.645791 139868561745792 efficientnet_model.py:147] round_filter input=192 output=208
     I1114 00:39:24.646027 139868561745792 efficientnet_model.py:147] round_filter input=320 output=352
     I1114 00:39:24.951259 139868561745792 efficientnet_model.py:147] round_filter input=1280 output=1408
     I1114 00:39:25.011001 139868561745792 efficientnet_model.py:458] Building model efficientnet with params ModelConfig(width_coefficient=1.1, depth_co
     WARNING:tensorflow:From /usr/local/lib/python3.7/dist-packages/object_detection/model_lib_v2.py:558: StrategyBase.experimental_distribute_datasets_f
     Instructions for updating:
     rename to distribute_datasets_from_function
     W1114 00:39:25.073961 139868561745792 deprecation.py:345] From /usr/local/lib/python3.7/dist-packages/object_detection/model_lib_v2.py:558: Strategy
     Instructions for updating:
     rename to distribute_datasets_from_function
     INFO:tensorflow:Reading unweighted datasets: ['training_job/tfrecords/train.record']
     I1114 00:39:25.079700 139868561745792 dataset_builder.py:163] Reading unweighted datasets: ['training_job/tfrecords/train.record']
     INFO:tensorflow:Reading record datasets for input file: ['training_job/tfrecords/train.record']
```

Here we will get the trained model file in the format of zip. We have to download that file so that we can able to detect multiple objects.



```
!zip -r saved_model.zip inference_graph/saved_model/

  adding: inference_graph/saved_model/ (stored 0%)
  adding: inference_graph/saved_model/saved_model.pb (deflated 93%)
  adding: inference_graph/saved_model/variables/ (stored 0%)
  adding: inference_graph/saved_model/variables/variables.data-00000-of-00001 (deflated 25%)
  adding: inference_graph/saved_model/variables/variables.index (deflated 78%)
  adding: inference_graph/saved_model/assets/ (stored 0%)

+ Code  + Text

Main : Donwload saved_model.zip
```

We are detecting the object with the label and the boundary box

```
+ Code  + Text                                                    Connect  Editing

[ ] import pandas as pd
    test = pd.read_csv('/content/testing.csv')
    images = list(test['filename'][0:20])

    outputs = test['class'].values

[ ] for e, image_name in enumerate(images):
      print(outputs[e])
      image_np = load_image_into_numpy_array('/content/ShoppingDetectionData/Images/' + image_name)
      output_dict = run_inference_for_single_image(model, image_np)
      vis_util.visualize_boxes_and_labels_on_image_array(
          image_np,
          output_dict['detection_boxes'],
          output_dict['detection_classes'],
          output_dict['detection_scores'],
          category_index,
          instance_masks=output_dict.get('detection_masks_reframed', None),
          use_normalized_coordinates=True,
          line_thickness=8)
```

We have run the code which is trained and saved in the before step and we upload that folder in dropbox for detecting the multiple objects at a time with the price.

We have also added the price tag for each object so that we can get the price for that object which is detected.

The above screenshot shows the final output of the detected object along with the price.

THANK YOU