Configuration Manual

MSc Research Project
Data Analytics

Abhinav Bhardwaj
Student ID: x20100906

School of Computing
National College of Ireland

Supervisor: Prof Aaloka Anant

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Abhinav Bhardwaj
Student ID:	x20100906
Programme:	MSc Data Analytics
Year:	2021-22
Module:	MSc Research Project
Supervisor:	Prof Aaloka Anant
Submission Due Date:	31/01/2022
Project Title:	Configuration Manual
Word Count:	1409
Page Count:	12

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:	Abhinav Bhardwaj
Date:	31st January 2022

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

Abhinav Bhardwaj
x20100906

1 Introduction

The actions taken to carry out this research's implementation are described in full in this configuration manual. Data collection and processing, feature extraction, and model creation are all part of this process. In order to assure reproducibility, the code samples, screenshots, and step-by-step instructions are also included.

2 Hardware and Software configurations

Host machine/Operating System:	MacBook Pro/MacOS Catalina And Windows Machine (AMD Ryzen 9 5900HS with Radeon Graphics 3.30 GHz)/Windows 10 Home
RAM	8 GB, M1 chip processor And 16 GB (For Windows).
Hard Disk	256GB And 1TB SSD (For Windows)
Cloud compute (GPU)	Free GPU Tesla K80 offered by Colab with 2496 CUDA cores and 12GB RAM.

Table 1: Hardware specifications

Programming language	Python (Anaconda distribution)
IDE	Jupyter notebook.
Cloud environment	Google Collaboratory
Browser	Google chrome

Table 2: Software specifications

Data is first processed on the local workstation and saved as .tar.gz files before being transferred to Google Colab for modeling.

3 Data Preparation

The author has used self-created data. The file, post the split in training and testing, has been saved on Google Drive. Please follow the link to download the same.

Weapoon dataset link : https://drive.google.com/drive/folders/1zLBJ099QElaai0tSVLBwpdvfL8yG

3.1 Creating Environment

Open Terminal/Comman Window

- Set up a new environment with name tfod using the following command:
- !conda create –name tfod python=3.8 ; when asked for procced : press Y
- !conda activate tfod
- !python -m pip install –upgrade pip
- !pip install ipykernel
- python -m ipykernel install –user –name=tfodj

Steps to install Tensorflow :

- Kindly refer the foot note for Tensorflow Github repository[1]
- !git clone https://github.com/tensorflow/models.git
- !pip install –ignore-installed –upgrade tensorflow==2.5.0
- verify your installation :
- python -c "import tensorflow as tf;
- print(tf.reduce_sum(tf.random.normal([1000, 1000])))"

You can skip steps 15 to 21 if you only want to run the second section of the code on google colab. Disclaimer: The last piece of the code, where one have to detect an object using a live feed from one's webcam. Code only run on one's local machine. Google Colab does not have any solutions in which one may attach a webcam and execute object detection on a live feed.

- To install the CUDA Toolkit as per the local machine's requirement and built follow the link: https://developer.nvidia.co-11.2.2-download-archive?target$_os = Linuxtarget_arch = x86_64$
- To install the CUDNN follow : https://developer.nvidia.com/rdp/cudnn-download
- Create a user profile if needed and log in to select archive file for Cudnn: https://developer.nvidia.c archivea-collapse810-111
- Extract the contents of the zip file (i.e. the folder named cuda) inside INSTALL_PATH NVIDIA GPU Computing Toolkit CUDA v11.2 where INSTALL_PATH points to the installation directory specified during the installation of the CUDA Toolkit. By default INSTALL_PATH CDrive :Program Files.
- Download the latest protoc-*-*.zip release from https://github.com/protocolbuffers/protobuf/rele
- Extract the contents of the downloaded protoc-*-*.zip in a directory PATH_TO_PB of your choice (e.g. C drive Program FilesProtobuf)

[1]Tensorflow Github repository : `https://github.com/tensorflow/models`

- Add PATH_TO_PB bin to your Path environment variable.
- In a new Terminal 1, cd into TensorFlow/models/research/ directory and run the following command:!protoc object_detection/protos/*.proto –python_out=.

We have to download Tensorflow 2 Object Detection API.For the same please follow the step under the same terminal/ command prompt:

- Kindly refer the foot note for Tensorflow API[2]
- Download the COCO API : !git clone https://github.com/cocodataset/cocoapi.git
- cd cocoapi/PythonAPI
- !cp -r pycocotoolsPATH_TO_TF¿/TensorFlow/models/research/
- From within TensorFlow/models/research/
- cp object_detection/packages/tf2/setup.py .
- python -m pip install –use-feature=2020-resolver .
- Test the installation : From within TensorFlow/models/research/
- !python object_detection/builders/model_builder_tf2_test.py
- If everything goes fine run : jupyter notebook
- Post that run the first files Image_Collection.

4 Project Development

PYTHON programming was used exclusively in the implementation. This research project is divided into three stages: data preparation, modeling, and evaluation. The first stage consists of data preprocessing and data selection, followed by the modeling stage, which consists of model implementation using TensorFlow, Keras, and Tensorflow Zoo model2, and finally, model evaluation using performance metrics such as average precision, average recall, and localization loss.

4.1 Data collection

- First, we will start importing the required libraries such as OpenCV2 and Time, which will help us capture the live images using our webcam. Kindly refer the figure 1
- In the following section, we will be setting up folders on our local machine. Where all our images, pre-trained models, and trained models will be saved. Kindly refer the figure 2

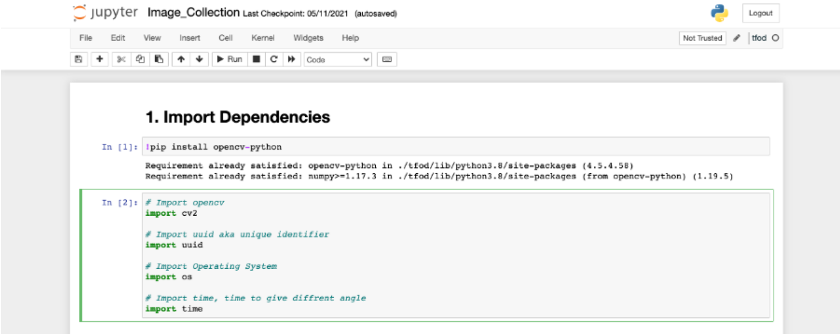

Figure 1: Import Lib

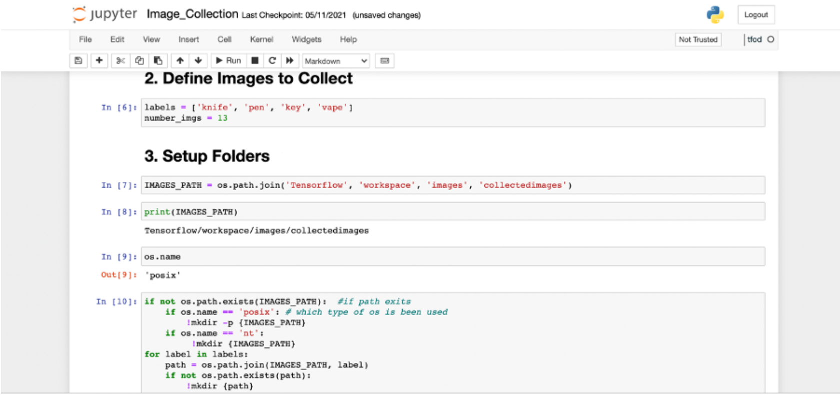

Figure 2: Setting up folders

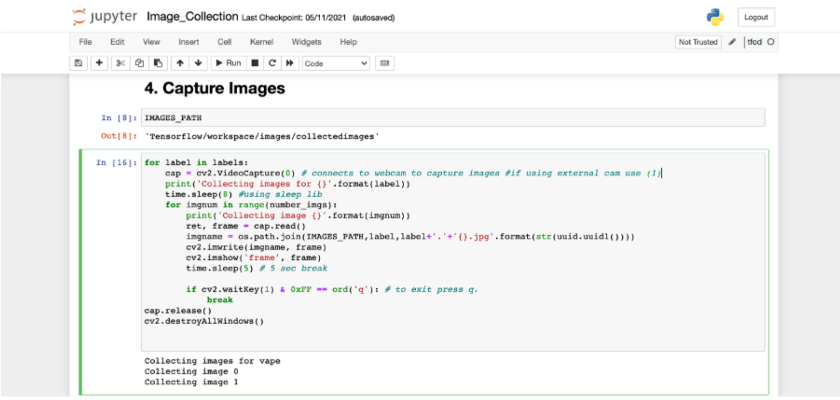

Figure 3: Image capture for Data collection

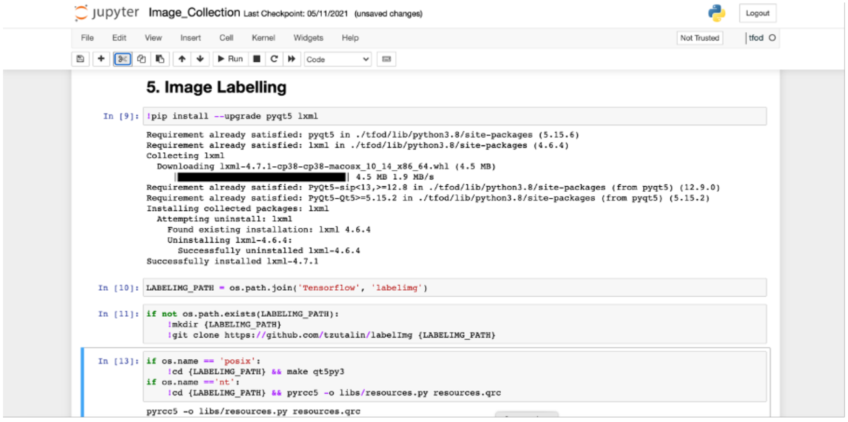

Figure 4: Image labelling

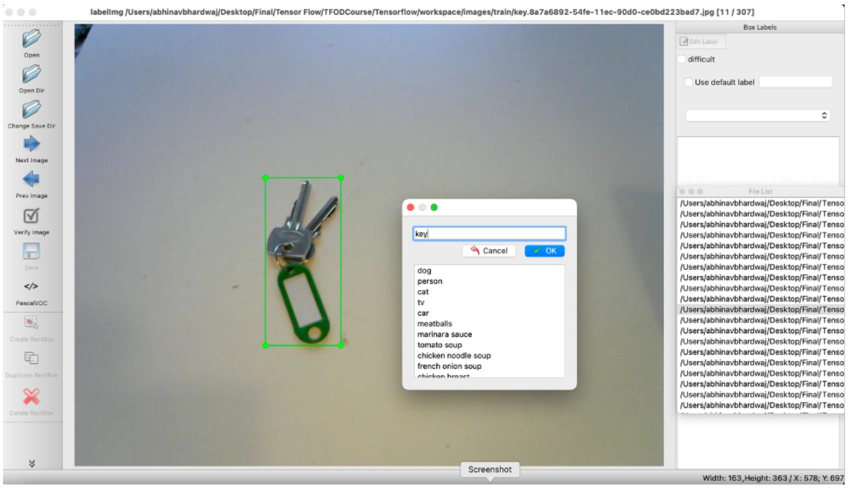

Figure 5: Labelling tool

- In this step, we are ready to capture the live images from our webcam using the open CV2 library. We can set up the timer as per our need for capturing the images.Kindly refer the figure 3
- We have to label the captured image using a graphical image annotation tool created by Darren Tzutalin from his public GitHub repository named LabelImg. On saving the each labbeled image it will also save the xml file with the same name as of file. The file will contains the x and y coordinates of the labelled image. Kindly refer the figure 4. Kindly refer the footnote for git hub repo [3].
- Referring the figure 5, is an example of how one can access the local directory and start labelling the images.
- Finally we have images in our said folders of Knife, Key, Pen, Vape. We have to move them into the testing and training folders Manually along with their XML

[2]Tensorflow API installation : https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html

[3]Image labelling: https://github.com/tzutalin/labelImg

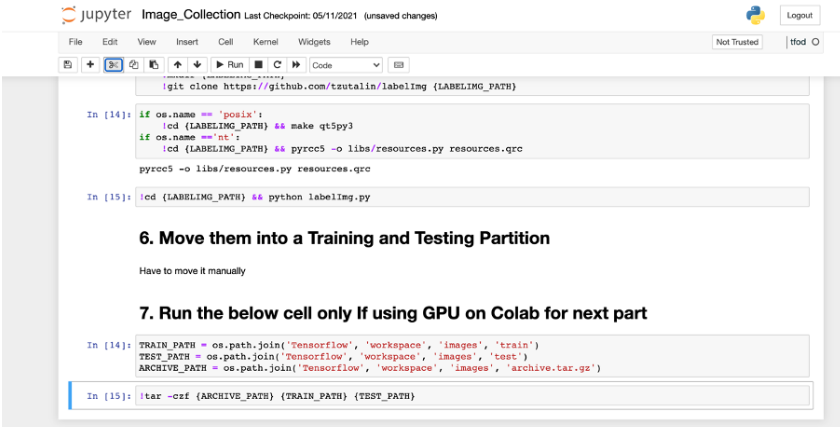

Figure 6: Moving to Test and train

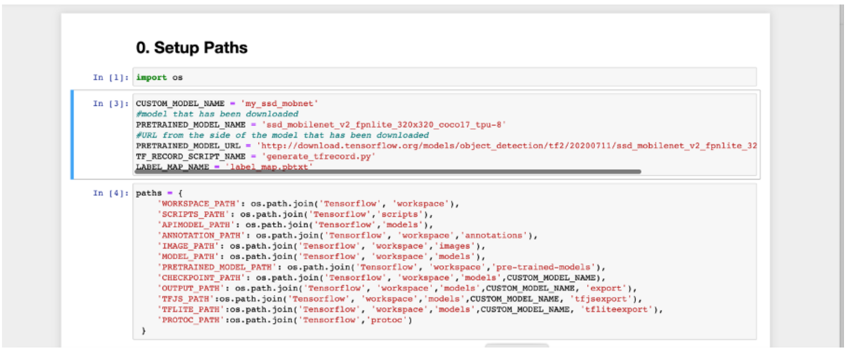

Figure 7: Setting up paths for Pre-Trained models

files. For this project the was a split of 75 training and 25 testing data split.

Using the step 7 is only for those who are running the code on google colab. This will create an archive forlder for your test and train dataset. Which needs to be uplodaed on colab.Kindly refer the figure 6

4.2 Training and Detection

- Starting with importing the required library and the pre-trained model from the Tensorflow zoo github repository. Paths has been defined for all the pre-trained models. Kindly refer the figure 7 and figure 8
- Downloading the pre-trained models from and setting up on the paths and location for extraction. Kindly refre the figure 9
- running the verification script for TensorFlow, will give us go ahead of rest of the code. We need OK from post we execute the script.Kindly refer the figure 10
- Finally one can extract the pre-trained models it will look like the one in referred the figure 11
- This step required to create the label maps for all the object. please note that this is case-sensitive and make sure one should use the same name as it has been

Figure 8: Part 2 of setting up paths for Pre-Trained models

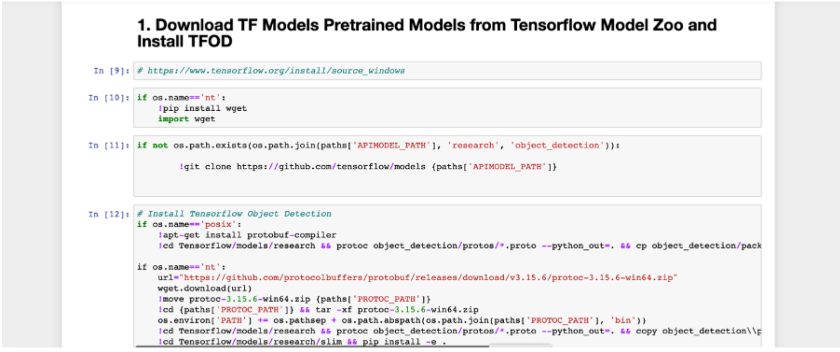

Figure 9: setting up paths for Pre-Trained models

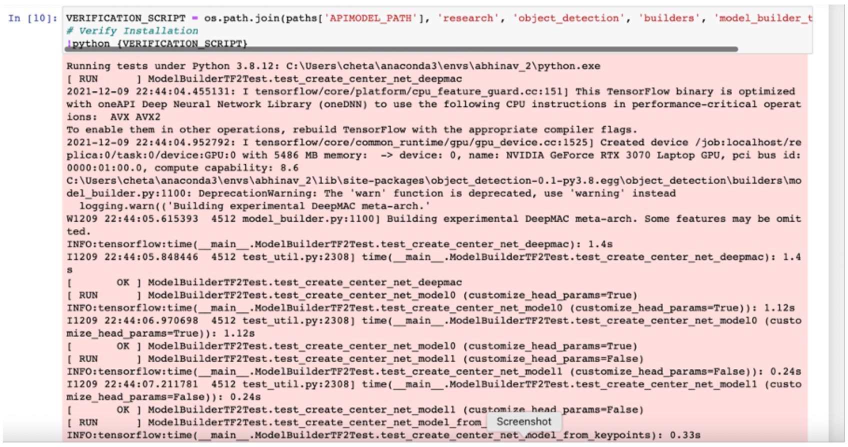

Figure 10: Verification Script

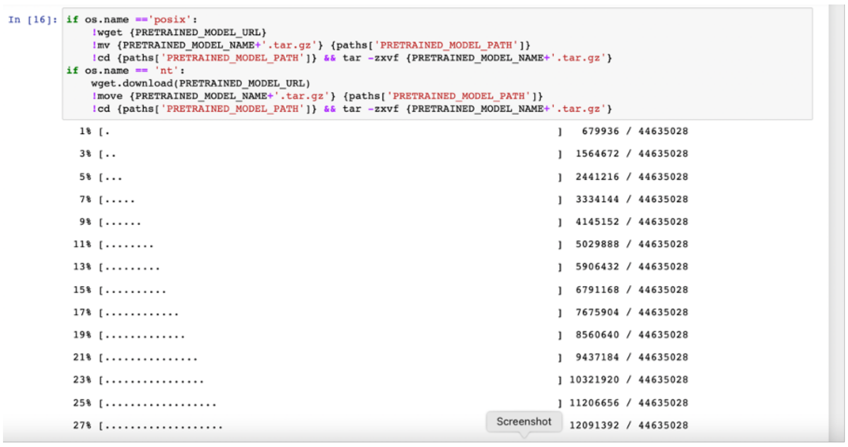

Figure 11: setting up paths for Pre-Trained models

Figure 12: Creating label maps and TF records

used while creating the folders and labelling them. WE are also creating the TF records and setting up tf training and testing scripts for the model. Kindly refer the figure 12

- once we have the tf scripts for testing and training data. we need to now configure the pre-tranined models to the refereed paths.Kindly refer the figure 13
- Training of model will require the command prompt to see the process of model training. Once we have the command printed paste it on the Tensflow terminal. Kindly refer the figure 14 and figure 15
- Under the evualtion part we have to follow the procecess of copying the command to the tensorflow terminal which will generate the following optput. Kindly refer the figure 16 and figure 17
- To visualize TensorBoard from Train folder kinldy follow the path: Tensorflow workspace models my_ssd_mobnet train¿tensorboard –logdir=.
- To visualize TensorBoard from Eval folder kindly follow the path: Tensorflow workspace models my_ssd_mobnet eval tensorboard –logdir=.

Figure 13: Configuring Pre-trained models

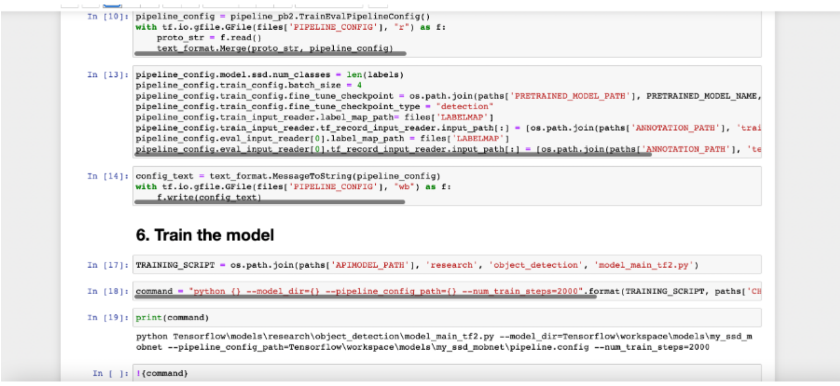

Figure 14: Model Traning

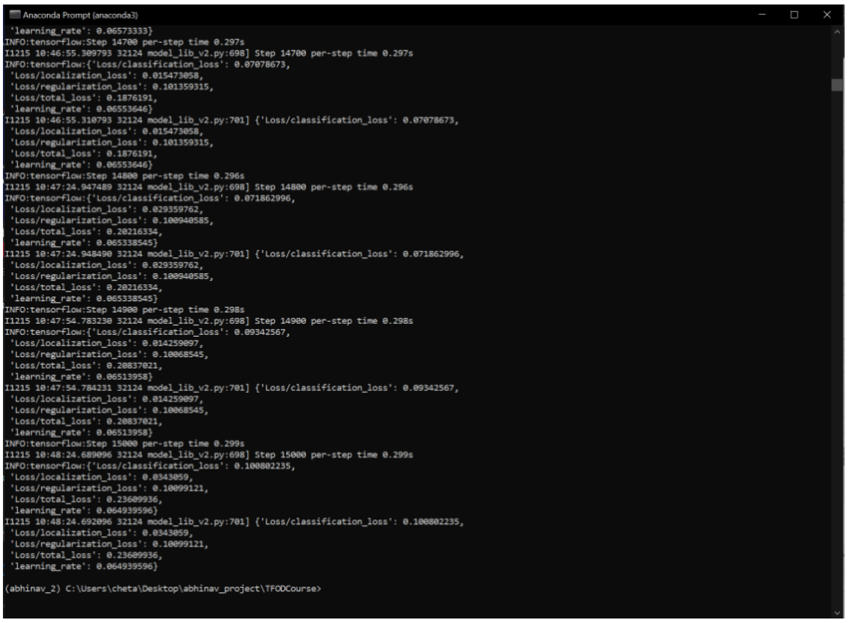

Figure 15: Traning Steps

Figure 16: Evaluation

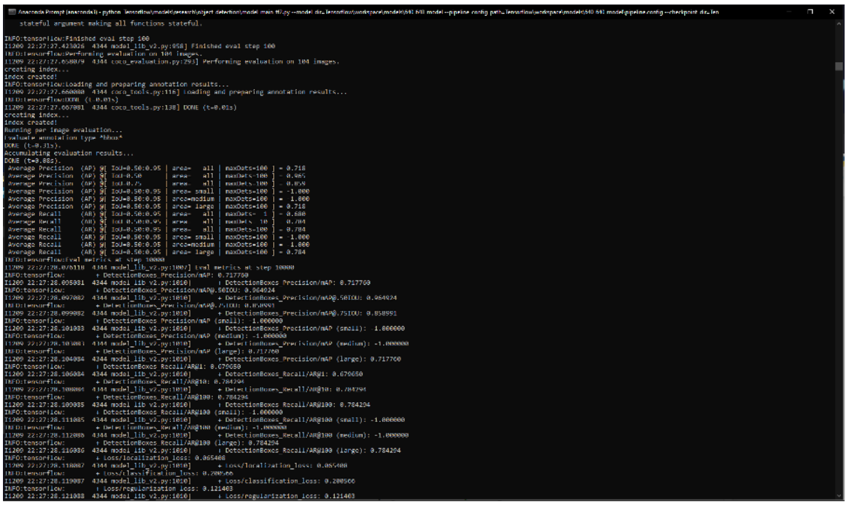

Figure 17: Evaluation on Terminal

- For Google Colab to run Tensorboard from train folder kindly follow the path:
  reload_ext tensorboard
  load_ext tensorboard
  !cd
  /Tensorflow/workspace/models/ssd_640_640/train/
  tensorboard –logdir .

- For Google Colab to run Tensorboard from eval folder: reload_ext tensorboard
  load_ext tensorboard

  !cd /Tensorflow/workspace/models/ssd_640_640/eval/
  tensorboard –logdir .

- Kindly refer the following figure 18 and figure 19 to see the TensorBoard outputs.

- We can even detect the object by inputting an image. Kindly refer the figure 20

- In the final portion of the code we can run our our live feed cam to detect an image using our webcam. Kindly refer the figure 21, figure 22 and figure 23

- we can finally save the model and the checkpoint so that one dose not have to run the complete program again. Kindly refer the figure 24

Figure 18: Tensorboard Loss and Learning graph

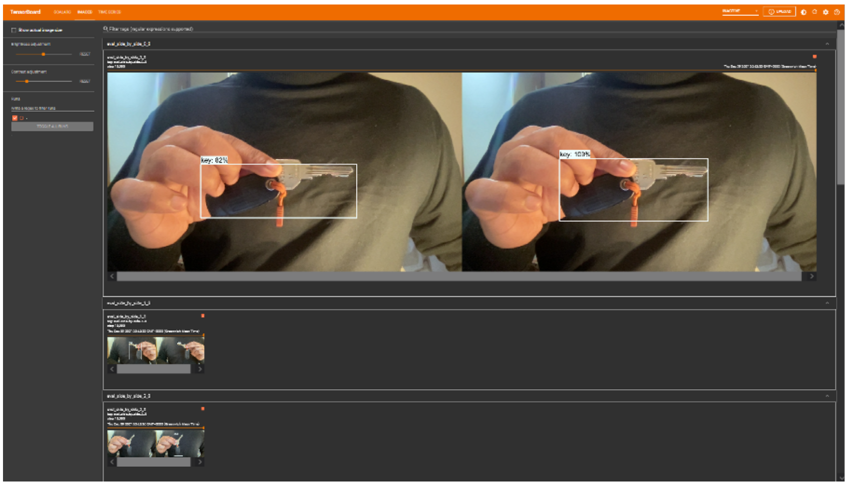

Figure 19: Testing using an image from test folder

Figure 20: Detecting the object from an image

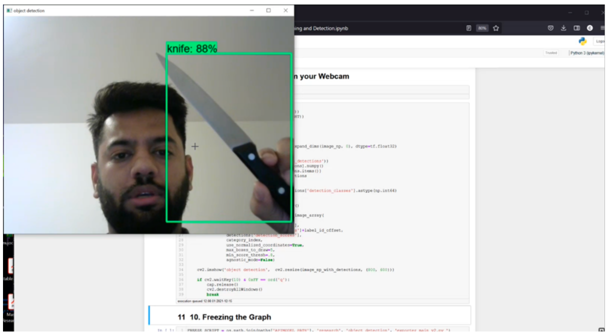

Figure 21: Detecting and object using live feed 1

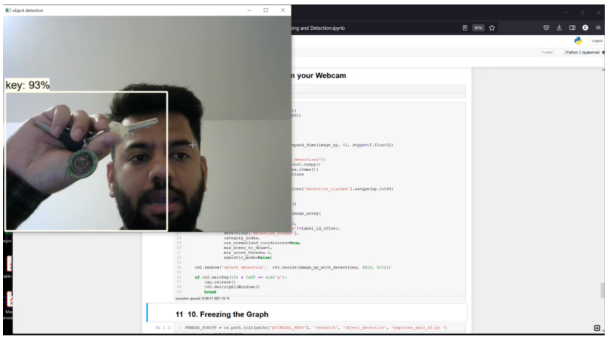

Figure 22: Detecting and object using live feed 2

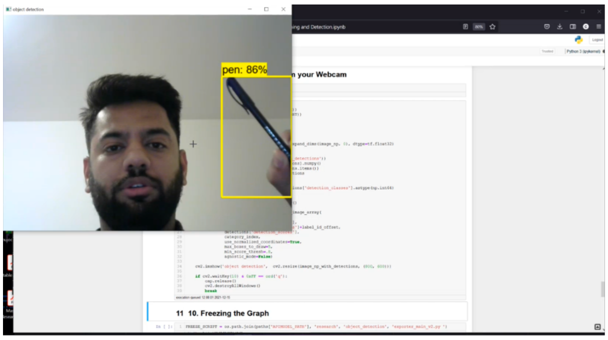

Figure 23: Detecting and object using live feed 3

8. Load Train Model From Checkpoint

In [7]:
```
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util
```

In [11]:
```
# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-5')).expect_partial()

@tf.function
def detect_fn(image):
    image, shapes = detection_model.preprocess(image)
    prediction_dict = detection_model.predict(image, shapes)
    detections = detection_model.postprocess(prediction_dict, shapes)
    return detections
```

Figure 24: Load the checkpoints and save model