GESTURE BASED HOME AUTOMATION SYSTEM

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the Degree of

**BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE ENGINEERING**

by

**Abhishek (19BCI7058)
Divyansh(19BCD7171)
Prashant Fachara(19BCE7271)**

Under the Guidance of

DR. Y MOHAMED SIRAJUDEEN

SCHOOL OF COMPUTER SCIENCE ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

DECEMBER 2022

CERTIFICATE

This is to certify that the Capstone Project work titled "**GESTURE BASED HOME AUTOMATION SYSTEM**" that is being submitted by **ABHISHEK (19BCI7058), DIVYANSH(19BCD7171) AND PRASHANT FACHARA(19BCE7271)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. Y Mohamed Sirajudeen

Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner	External Examiner

Approved by

PROGRAM CHAIR	**DEAN**
B. Tech. CSE –AI, DA, CORE	School Of Computer Science Engineering

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to Dr. Y Mohamed Sirajudeen for his guidance and support during the development of this project. Your valuable insights and constructive feedback have been instrumental in shaping the direction and outcomes of our work. Your expertise and mentorship have been invaluable to us, and we are grateful for the opportunity to have worked with you.

We also want to thank VIT-AP University for providing us with the resources and support necessary to complete this project. Without their assistance, this project would not have been possible.

Thank you for your invaluable guidance and support.

Abhishek, Divyansh and Prashant

ABSTRACT

In this paper an efficient system is presented which can automatically detect and map the gesture made by the user to activate/act upon connected systems in their house. Hand-gesture based control has enormous potential both theoretically and for practical applications, as it is convenient and intuitive. One of these application is found in Home Automation.

Under this project we aim to present a real-time interactive control system to achieve home automation. This interactive control system shall allow wireless control of household appliances using a combination of several hand gestures and waving motions.

We aim to achieve our goal using the most feasibly available technology, a regular PC and existing wireless signal processing systems.

Lastly, the study will present the analysis and effects of gestures control for a higher up-take of smart home solutions towards designing and maintaining buildings of the future that are both user-centric and resource efficient.

TABLE OF CONTENTS

List of Tables

List of Figures

CHAPTER 1
INTRODUCTION

A gesture-based home automation system is a type of control system that allows you to control various devices in your home using gestures, rather than buttons or touchscreens. These systems typically use sensors such as cameras or infrared detectors to track a person's gestures, and then interpret those gestures to control devices such as lights, thermostats, doors, and appliances.

One way to build a gesture-based home automation system is to use a combination of computer vision, machine learning and deep learning techniques. The system would first use a camera to capture images of a person's gestures, and then use machine learning algorithms to interpret those gestures. For example, a person could train the system to recognize a specific gesture, such as a wave of the hand, as a command to turn on the lights.

Another way to build a gesture-based home automation system is to use specialized gesture recognition hardware such as a Kinect sensor, which uses infrared cameras and depth sensors to track a person's movements. This type of system can be used to control devices in the home by mapping specific gestures to different commands.

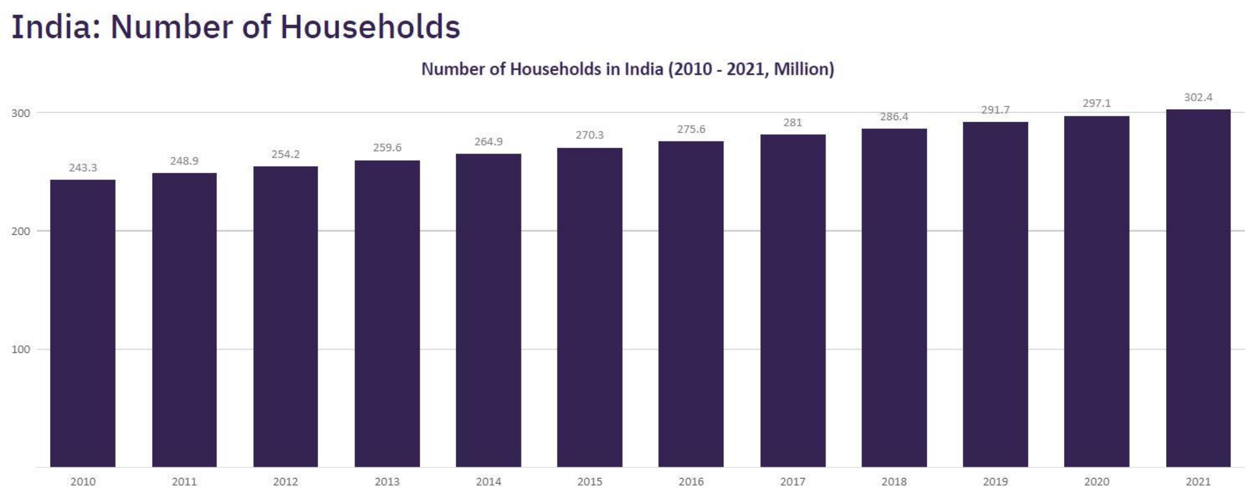


Figure 1 Number of Houses

The number of houses in India has been steadily increasing over the years, driven by factors such as population growth and urbanization. The Government of India has also launched several initiatives to increase the availability of housing, such as the Pradhan Mantri Awas Yojana (PMAY) and the Housing for All by 2022 scheme.

As per the census data in 2021, the total number of households in India is about 252 million. The number of houses in India also depends on many factors like rural-urban split, population density and availability of land, the data might vary to the actual situation. In these scenarios, the potential of this technology's application in the real world is huge.

Whichever the approach, the gesture-based home automation system is still a relatively new technology, and there are many challenges to be addressed before it can be widely adopted. These include developing robust gesture recognition algorithms that can work in a variety of lighting conditions, as well as creating user-friendly interfaces for interacting with the system.

1.1 Objectives

The objectives of the project "Gesture Based Home Automation System" are:

- Convenience: A gesture-based system aims to make controlling home devices more convenient and intuitive for users. This could include being able to turn lights on and off, close or open doors with simple hand movements.
- Accessibility: A gesture-based home automation system can also improve accessibility for people with disabilities or limited mobility. For example, it can allow people to control devices without having to reach for buttons or use a remote control.
- New User Experience: A Gesture based home automation system is a fresh and unchartered path interactions with AI, control of home appliances provides a new and more interactive user experiences.
- Smart Home Automation: With the integration of sensors and other smart devices, a gesture-based home automation system makes smart homes smarter.

1.2 Background and Literature Survey

Gesture based Home Automation Systems can be a very convenient and easy way to control devices in your home, it can be a futuristic way of control devices with a simple gesture.
There are some commercial product available such as Myo armband, it uses the electrical activity of muscles to detect the gestures.

Myo is a type of wearable technology that is worn around the arm, which uses electromyography (EMG) sensors to detect the electrical activity of muscles. These sensors are able to detect subtle muscle movements, which can be interpreted as gestures. The Myo armband can be used to control various types of devices, such as smart phones, tablets, computers, and home automation systems, using gestures such as waving, tapping, and squeezing.

One of the key features of the Myo armband is its ability to recognize specific hand gestures, such as a fist or a wave, and use those gestures to trigger different actions or commands. For example, a user could train the Myo armband to recognize a fist gesture as a command to turn off the lights, and a wave gesture as a command to turn on the music.

We referred to a paper titled "Using gestures to interact with home automation systems" by Mr. Marcel Villanueva, doctoral candidate at Centrale Supelec/ Universite Paris - Saclay. This paper talks about using hand gestures to interact with home systems.

1.3 Organization of the Report

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the proposed system, methodology, hardware and software details.
- Chapter 3 gives the cost involved in the implementation of the project.
- Chapter 4 discusses the results obtained after the project was implemented.
- Chapter 5 concludes the report.
- Chapter 6 consists of codes.
- Chapter 7 gives references.

CHAPTER 2

VEHICLE SERVICE MANAGEMENT AND LIVE MONITORING SYSTEM

This Chapter describes the proposed system, working methodology, software and hardware details.

2.1 Proposed System

The following diagram shows the system architecture of this project.

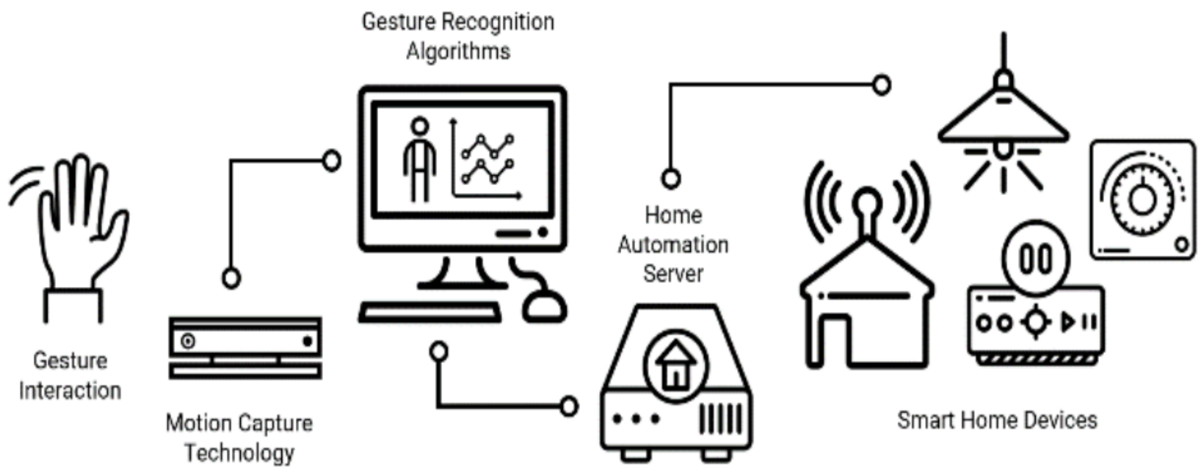


Figure 2.1 System Diagram

2.2 Working Methodology

1. **Setup TensorFlow Directory in Google Colaboratory:**

   TensorFlow-GPU allows our PC to use the video card to provide extra processing power while training. Create a folder directly in my drive and named it "tensorflow1". This working directory will contain the full TensorFlow object detection framework.

   Clone the full TensorFlow object detection repository at :

   https://github.com/tensorflow/models

2. **Download the ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz pretrained-model**

TensorFlow provides several object detection models (pre-trained classifiers with specific neural network architectures) in its model zoo.

```
#Download the pretrained model ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-
8.tar.gz into the data folder & unzip it.

!wget http://download.tensorflow.org/models/object_detection/tf2/20200711/s
sd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz

!tar -xzvf ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
```

3. **Configure PYTHONPATH environment variable**
   A PYTHONPATH variable must be created that points to the \models, \models\research, and \models\research\slim directories.
4. **Compile Protobufs and run setup.py**
   compile the Protobuf files, which are used by TensorFlow to configure model and training parameters.
5. **Test TensorFlow setup to verify it works**
6. **Process the image by resizing and normalizing as needed**
7. **Use the pre-trained CNN model ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8 to extract features**
8. **Based on the features, detect the hand gestures**
9. **Map the gestures to act on devices**

2.3 Standards

Various standards used in this project are:

- **Data set**

The model has been trained to detect five signs and subsequently map each sign to a unique action. Each image class has 30 images, so a total of 5 x 30 = 150 images.
Out of which 26 are for training, and 4 are for testing purposes.

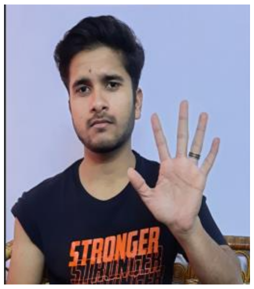
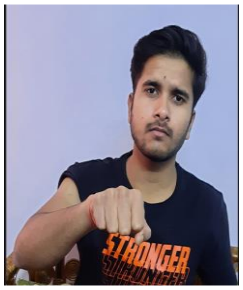
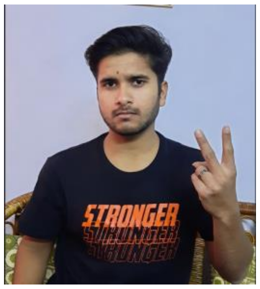
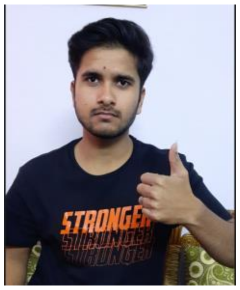





Fig 2.2.1 **Fig 2.2.2** **Fig 2.2.3** **Fig 2.2.4**

Open Gate **Close Gate** **Open Window** **Close Window**

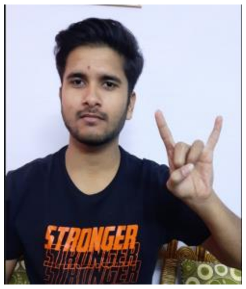


Fig 2.2.5

Start Fan

- ## Data Preparation

Image data pre-processing is an important step in preparing image data for use in machine learning and other computer vision tasks. It ensures the data selected for training and testing the model is optimum. It includes steps like:

Resizing: Resizing images to a consistent size is an important step in preparing images for analysis. This step is often performed by resizing images to a specific width and height, or by resizing images so that their aspect ratio is preserved.

Cropping: Cropping images can help to remove irrelevant parts of an image and focus on the object or area of interest. This step is typically performed by specifying a rectangular region of an image to keep.

Normalization: This step is used to normalize pixel values across all images so that they have similar values.This is often done by normalizing the pixel values to a range of 0 to 1 or -1 to 1.

Data augmentation: This step helps to create variations of the images in the dataset by performing operations like rotating, flipping, zooming, etc. This is done to increase the size of the dataset and also to improve the model's robustness to different variations.

Splitting the data: The data should be split into training, validation, and test sets in order to properly evaluate and test the model. The training set is used to train the model, the validation set is used to fine-tune the model and select the best hyper-parameters, and the test set is used to evaluate the final performance of the model.

- **Transfer Learning Using DeepCNN(DCNN)**

Transfer learning is a technique that allows you to use a pre-trained deep neural network model for a new task, rather than training a new model from scratch. This approach is particularly useful when you have a limited amount of labelled data or when the new task is similar to the task for which the pre-trained model was originally trained.

One type of transfer learning that is commonly used with deep convolutional neural networks (CNNs) is called feature extraction. This approach involves using the pre-trained model to extract features from your new data and then training a new classifier to recognize the features. The pre-trained model's convolutional layers are typically used for feature extraction, as they are able to learn useful features from images.

To use transfer learning, you will typically start by selecting a pre-trained model that has been trained on a related task. This model will have already learned some useful features and patterns from the data it was trained on, which can then be transferred to the new task

Next, you will typically fine-tune the pre-trained model by continuing to train it on the new task, using the available data. This can involve adjusting the model's architecture or the training hyper- parameters in order to better fit the new task.This is done by unfreezing some of the layers of the pre-trained model and training those layers along with new classifier layers. This can be useful when the new task is similar to the task for which the pre-trained model was originally trained.

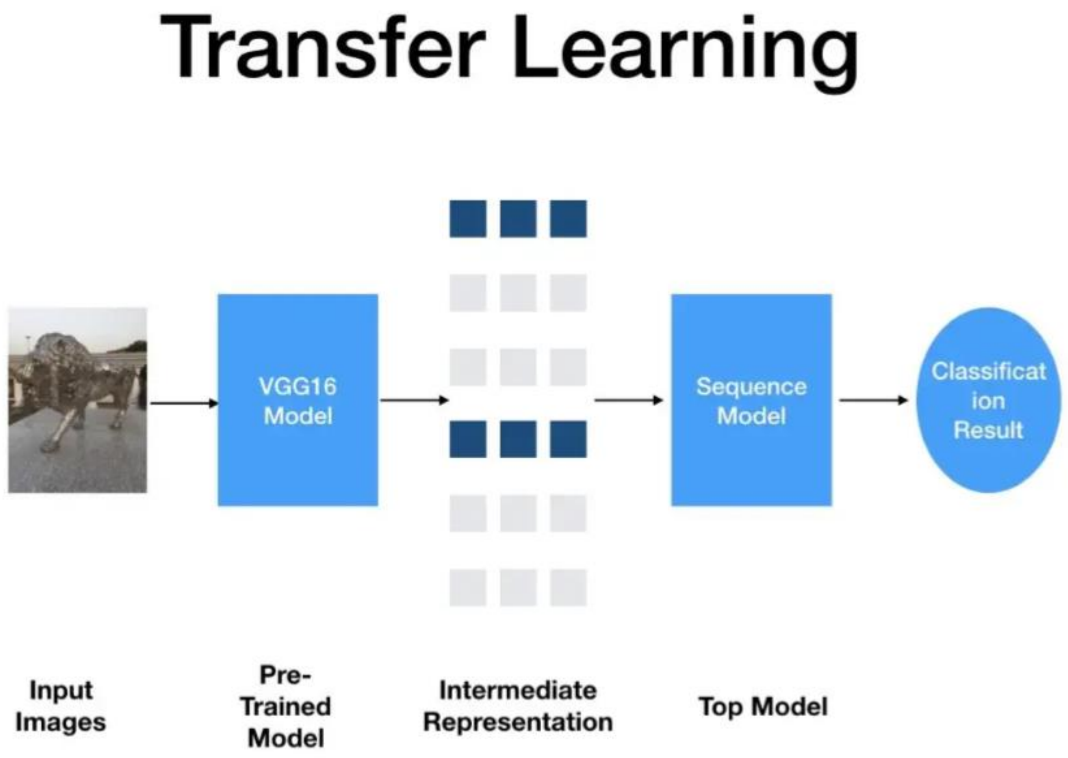


Fig 2.3 Transfer Learning Basic Idea

It is important to note that for transfer learning to be effective, the new task should be similar to the task for which the pre-trained model was originally trained. If the new task is too different from the original task, the pre-trained model may not be useful, and it may be necessary to train a new model from scratch.

It's also a good idea to use a pre-trained model that was trained on a large dataset like Imagenet, so that the model has already seen a wide range of images, and has good generalization. With the right pre-trained model, and appropriate fine-tuning, transfer learning can help you achieve good performance on your new task with less data and computation.

- **Gather and Label Pictures**

  TensorFlow needs hundreds of images of an object to train a good detection classifier. To train a robust classifier, the training images should have random objects in the image along with the desired objects, and should have a variety of backgrounds and lighting conditions. There should be some images where the desired object is partially obscured, overlapped with something else, or only halfway in the picture.

Label the images using a annotation tool, here we have used LableImg.

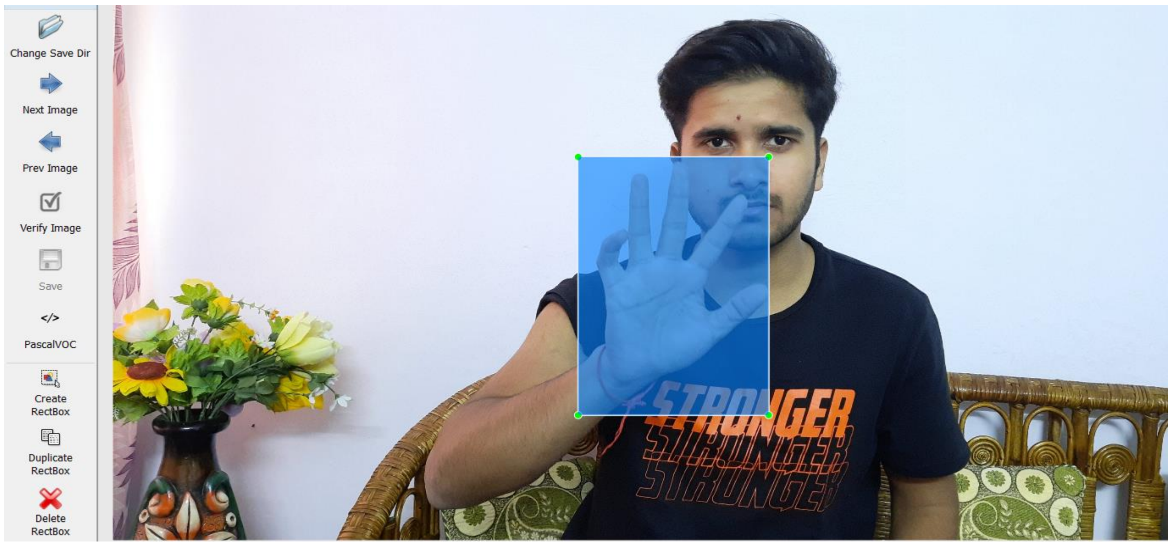


Fig 2.4 Label Images

- **Generate Training Data**

First we use xml_to_csv.py script to generate .csv files for test and train images, with the detail of image's annotation in the csv file.

Xml_to_csv.py script:

```
#adjusted from: https://github.com/datitran/raccoon_dataset

def xml_to_csv(path):

  classes_names = []

  xml_list = []



  for xml_file in glob.glob(path + '/*.xml'):

    tree = ET.parse(xml_file)

    root = tree.getroot()

    for member in root.findall('object'):

      classes_names.append(member[0].text)
      value = (root.find('filename').text  ,
               int(root.find('size')[0].text),
               int(root.find('size')[1].text),
               member[0].text,
               int(member[4][0].text),
               int(member[4][1].text),
               int(member[4][2].text),
               int(member[4][3].text))
      xml_list.append(value)
  column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']
  xml_df = pd.DataFrame(xml_list, columns=column_name)
  classes_names = list(set(classes_names))
  classes_names.sort()
  return xml_df, classes_names


for label_path in ['test_labels', 'train_labels']:
  image_path = os.path.join('/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/Labels/' + label_path)
  xml_df, classes = xml_to_csv(image_path)
  xml_df.to_csv(f'{label_path}.csv', index=None)
  print(f'Successfully converted {label_path} xml to csv.')
```

Now we run the **generate_tfrecord.py** script with necessary modification, to generate TFRecord files namely **train.record** .

- **Create Label Map and Configure Training**

  A label map, also known as a label mapping or annotation map, is a file that maps object labels to integer values in the context of object detection and semantic segmentation.

  For object detection, a label map file typically defines a mapping between object class names and integer class IDs, where each object class corresponds to a unique class ID.

```
item {
   name:'Open_Gate'
   id:1
}
item {
   name:'Close_Gate'
   id:2
}
item {
   name:'Open_Window'
   id:3
}
item {
   name:'Close_Window'
   id:4
}
item {
   name:'Start_Fan'
   id:5
}
```

Configuring the object detection pipeline. It defines which model and what parameters will be used for training.

Following are the changes made in the pipeline.cofig file:

pipeline_config.model.ssd.num_classes = 5

pipeline_config.train_config.batch_size = 8

pipeline_config.train_config.fine_tune_checkpoint= "/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint"

pipeline_config.train_config.fine_tune_checkpoint_type=”detection”
pipeline_config.train_input_reader.label_map_path = "/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/label_map.pbtxt"

pipeline_config.train_input_reader.tf_record_input_reader.input_path = "/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/train.record"

pipeline_config.eval_input_reader[0].label_map_path = "/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/label_map.pbtxt"

pipeline_config.eval_input_reader_[0].tf_record_input_reader.input_path = "/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/test.record"

- **Run the training**

  From the \object_detection directory issue the following command:

  !python model_main_tf2.py --pipeline_config_path=/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.config **--model_dir**=/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/Training **--alsologtostderr**

- **Evaluation and Testing**

  Using appropriate evaluation metrics: It is important to use appropriate evaluation metrics to measure the performance of the model. Common evaluation metrics for machine learning models include accuracy, precision, validation loss ,Training loss and time taken . It is also important to consider the specific requirements of the application and choose evaluation metrics that are relevant to the task at hand

  Test the model on a separate test set: It is important to test the model on a separate test set in order to evaluate its performance on new data. This will help to ensure that the model is not overfitting to the training data and is able to generalize to new data.

2.4 System Details

This section describes the software and hardware details of the system:

2.4.1 Software Details

Python programming language, TensorFlow, Labelmg, are used.

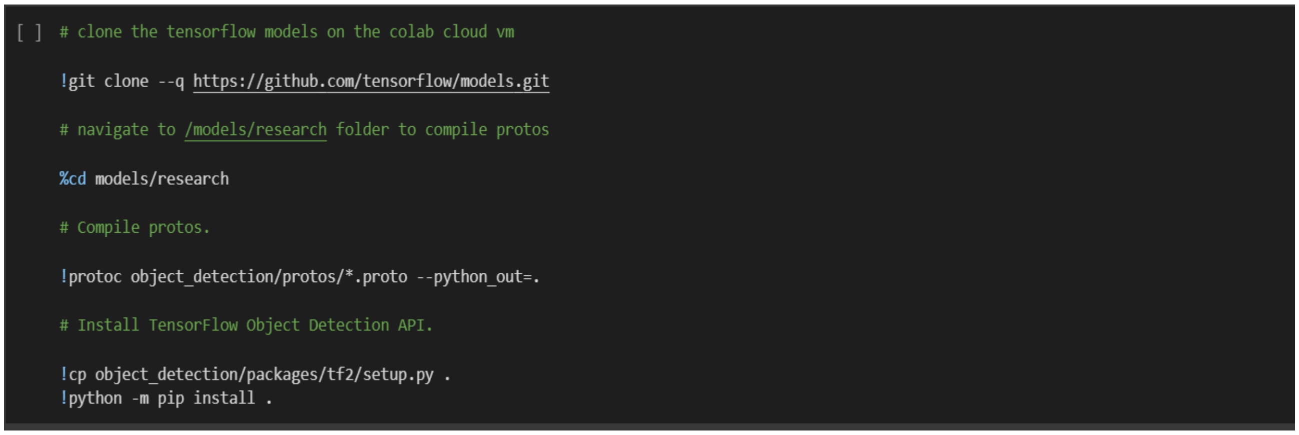
i) Python

Python is a high-level, general-purpose programming language that was first released in 1991. It is designed to be easy to read and write, and it is often used for web development, scientific computing, data analysis, artificial intelligence, and more.
With the combination of its simplicity, powerful libraries, and community support, Python has become one of the most popular programming languages, and is widely used in a variety of applications such as web development, data science, machine learning, AI, scientific computing and many more.

ii) TensorFlow

TensorFlow is an open-source software library for machine learning and artificial intelligence. It was developed by researchers and engineers working on the Google Brain team, and it was released under the Apache 2.0 open source license in 2015. TensorFlow can be used for a wide range of tasks such as training and deploying machine learning models, running complex mathematical computations, and even building and training deep neural networks.

TensorFlow is widely used in various industries such as healthcare, finance, and retail, and it is also used by many researchers and engineers working on new machine learning and AI algorithms. With its flexibility and performance, it has become one of the most popular and widely used machine learning libraries.



```
[ ] # clone the tensorflow models on the colab cloud vm

    !git clone --q https://github.com/tensorflow/models.git

    # navigate to /models/research folder to compile protos

    %cd models/research

    # Compile protos.

    !protoc object_detection/protos/*.proto --python_out=.

    # Install TensorFlow Object Detection API.

    !cp object_detection/packages/tf2/setup.py .
    !python -m pip install .
```

Fig 2.5 : Cloning TensorFlow Repository

iii) LableImg

TensorFlow is widely used in various LabelImg is a graphical image annotation tool that is written in Python and uses Qt for its graphical interface. It is open source and available on GitHub. It allows users to draw bounding boxes around objects in an image, and label them with a class name.

LabelImg is commonly used to create training data for object detection models, where the bounding boxes and class labels are used to train the model to identify the objects in new images. LabelImg also supports annotation of pixel-level masks and points.

The tool includes a simple user interface that allows users to open an image, draw rectangles around objects of interest, and label them with a class name. Once the image is annotated, it can be saved in the PASCAL VOC format, which is a commonly used format for storing image annotations, and can be easily used for training object detection models using libraries such as Tensorflow or Caffe.industries such as healthcare, finance, and retail, and it is also used by many researchers and engineers working on new machine learning and AI algorithms. With its flexibility and performance, it has become one of the most popular and widely used machine learning libraries.

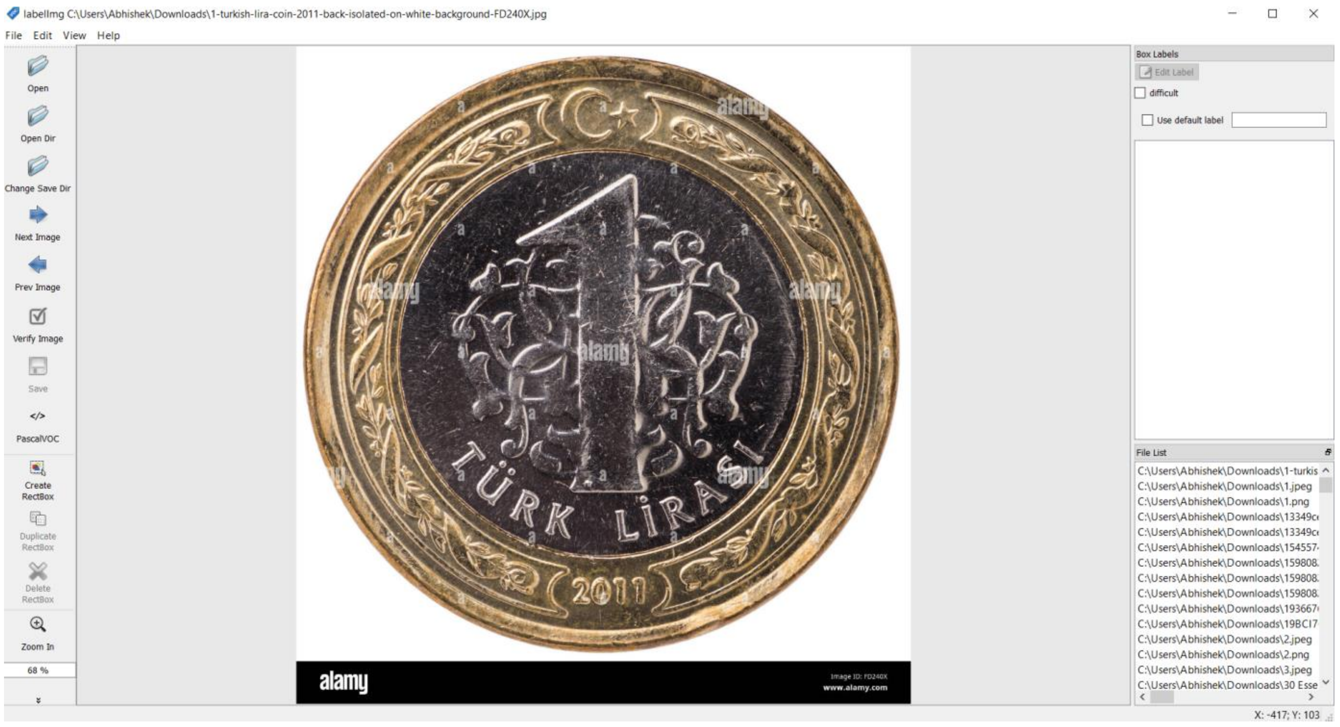

labelImg C:\Users\Abhishek\Downloads\1-turkish-lira-coin-2011-back-isolated-on-white-background-FD240X.jpg
File Edit View Help
Open
Open Dir
Change Save Dir
Next Image
Prev Image
Verify Image
Save
PascalVOC
Create RectBox
Duplicate RectBox
Delete RectBox
Zoom In
68 %
Box Labels
Edit Label
difficult
Use default label
File List
C:\Users\Abhishek\Downloads\1-turkis
C:\Users\Abhishek\Downloads\1.jpeg
C:\Users\Abhishek\Downloads\1.png
C:\Users\Abhishek\Downloads\13349c
C:\Users\Abhishek\Downloads\13349c
C:\Users\Abhishek\Downloads\154557
C:\Users\Abhishek\Downloads\159808
C:\Users\Abhishek\Downloads\159808
C:\Users\Abhishek\Downloads\159808
C:\Users\Abhishek\Downloads\193667
C:\Users\Abhishek\Downloads\19BC17
C:\Users\Abhishek\Downloads\2.jpeg
C:\Users\Abhishek\Downloads\2.png
C:\Users\Abhishek\Downloads\3.jpeg
C:\Users\Abhishek\Downloads\30 Esse
X: -417; Y: 103
TÜRK LİRASI
2011
alamy
www.alamy.com

Fig 2.6 : LabelImg Tool

2.4.2 Hardware Details

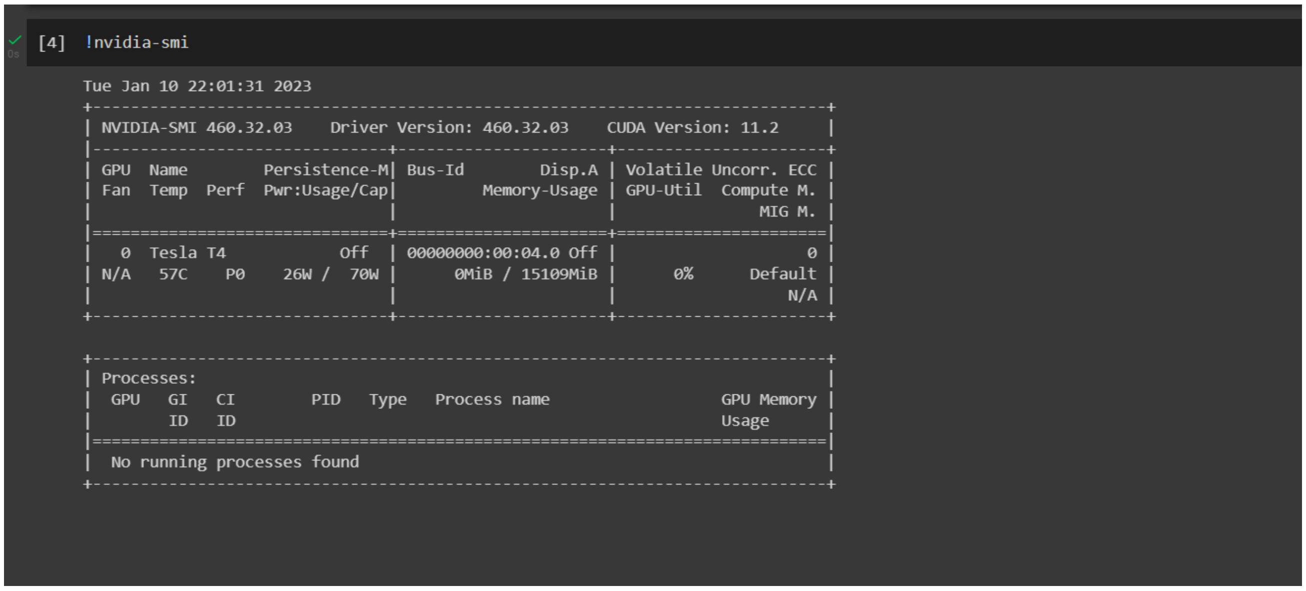
As shown in Figure 1 we have various hardware components being used in this system. The details of each component is as follows

i) Google Colaboratory

Google Colab, or Google Colaboratory, is a free, web-based platform that allows you to write, run, and share code in Python and other programming languages. It is especially useful for machine learning, data science, and scientific computing.

One of the key benefits of using Colab is that it provides access to free GPU resources, which can significantly speed up the training of large machine learning models. It also allows you to easily share notebooks with others, making collaboration on projects much simpler.

Google Colab is a very powerful platform that can be used for a wide range of tasks such as data preprocessing, building and training models and running simulations. With the support of free GPU and its integration with Google Drive, it makes the development and experimentation of complex models easier, and more accessible to the community.



```
[4] !nvidia-smi

Tue Jan 10 22:01:31 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    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 T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   57C    P0    26W /  70W |      0MiB / 15109MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

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

Fig 2.7 : Google Colab Specs

CHAPTER 3
COST ANALYSIS

3.1 List of components and their cost

The costs of the various components used in this project are given below in Table 3.1.

Table 3.1 List of components and their costs

COMPONENT	**COST**
Python	₹ 0(Open Source)
TensorFlow Repository	₹ 0(Open Source)
LabelImg	₹ 0(Open Source)
Google Colaboratory	₹ 0 (Provide By Google Free)
TOTAL	₹ 0/-

CHAPTER 4

RESULTS AND DISCUSSIONS

The final output of the trained model has been satisfactory. It was trained for 10,000 epochs and the model performed fairly well on the test samples.

It was able to correctly detect the gestures.

Input:

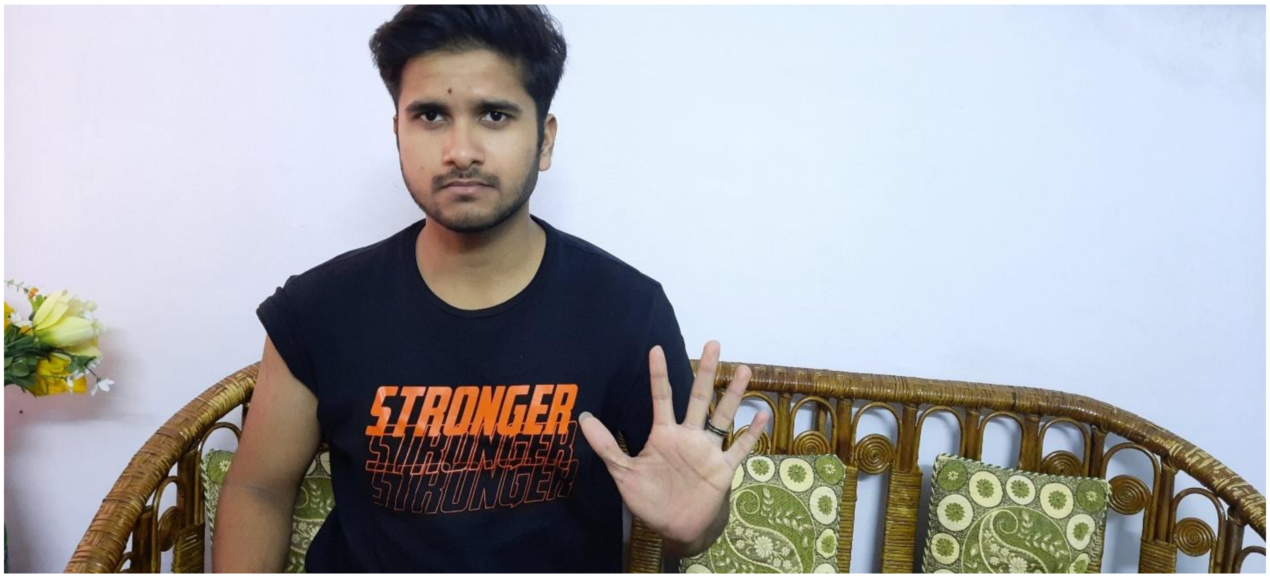


Fig 4.1 : Open Gate Input Image

Output :

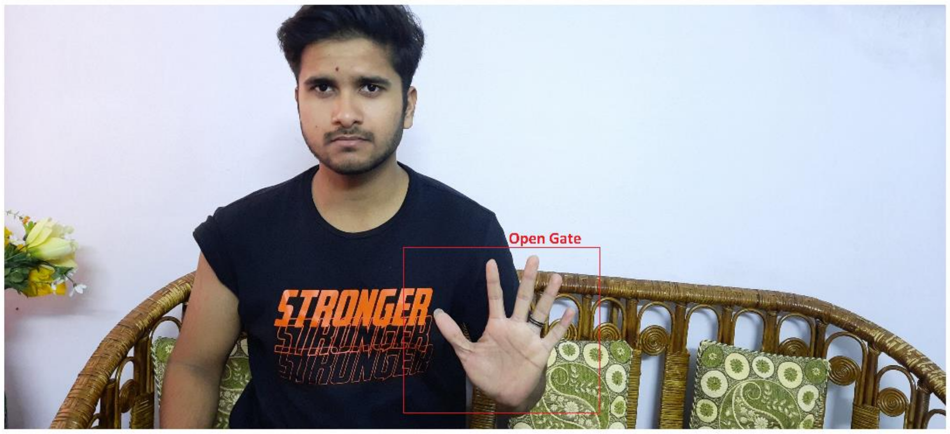


Fig 4.2 : Open Gate Output Image

Input :

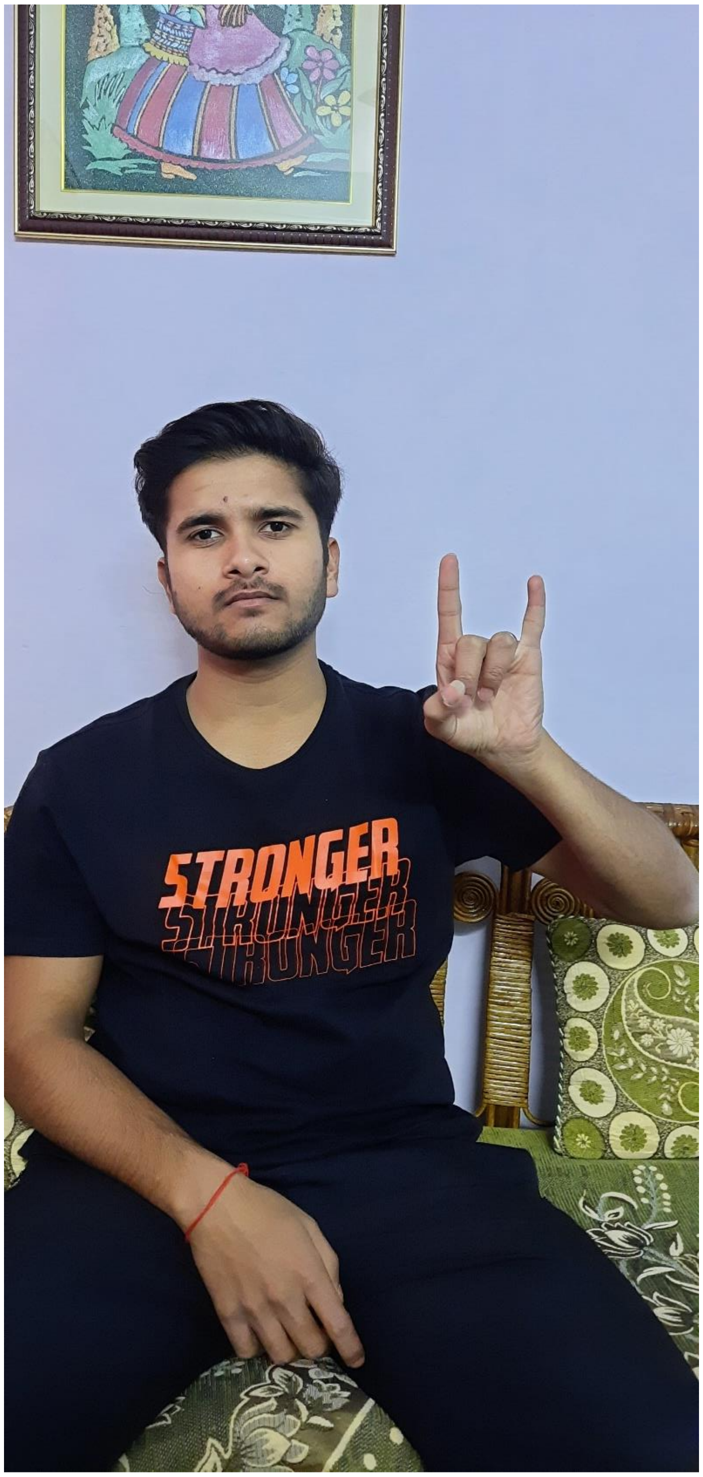


Fig 4. 3 : Start Fan Input

Output:

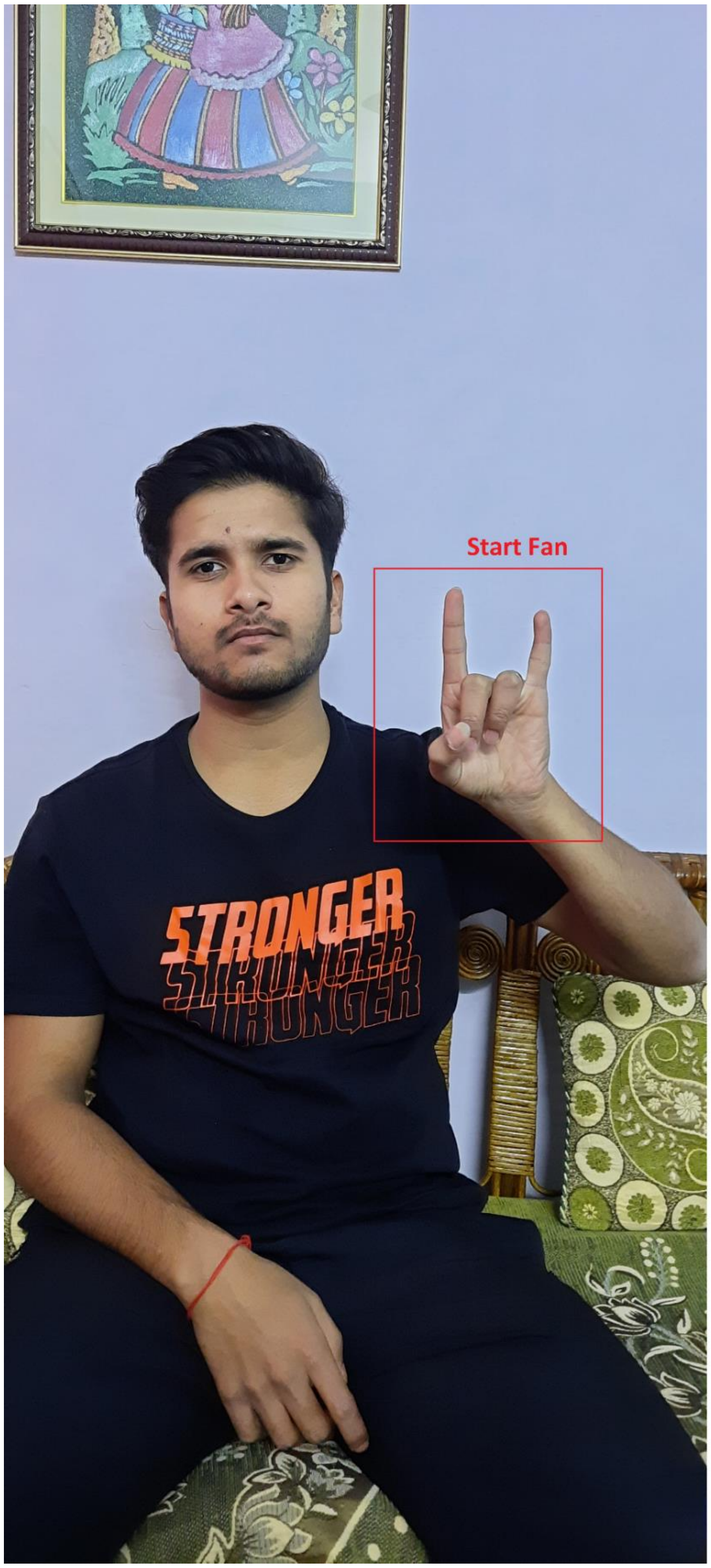


Fig 4.4 : Start Fan Output

Input:

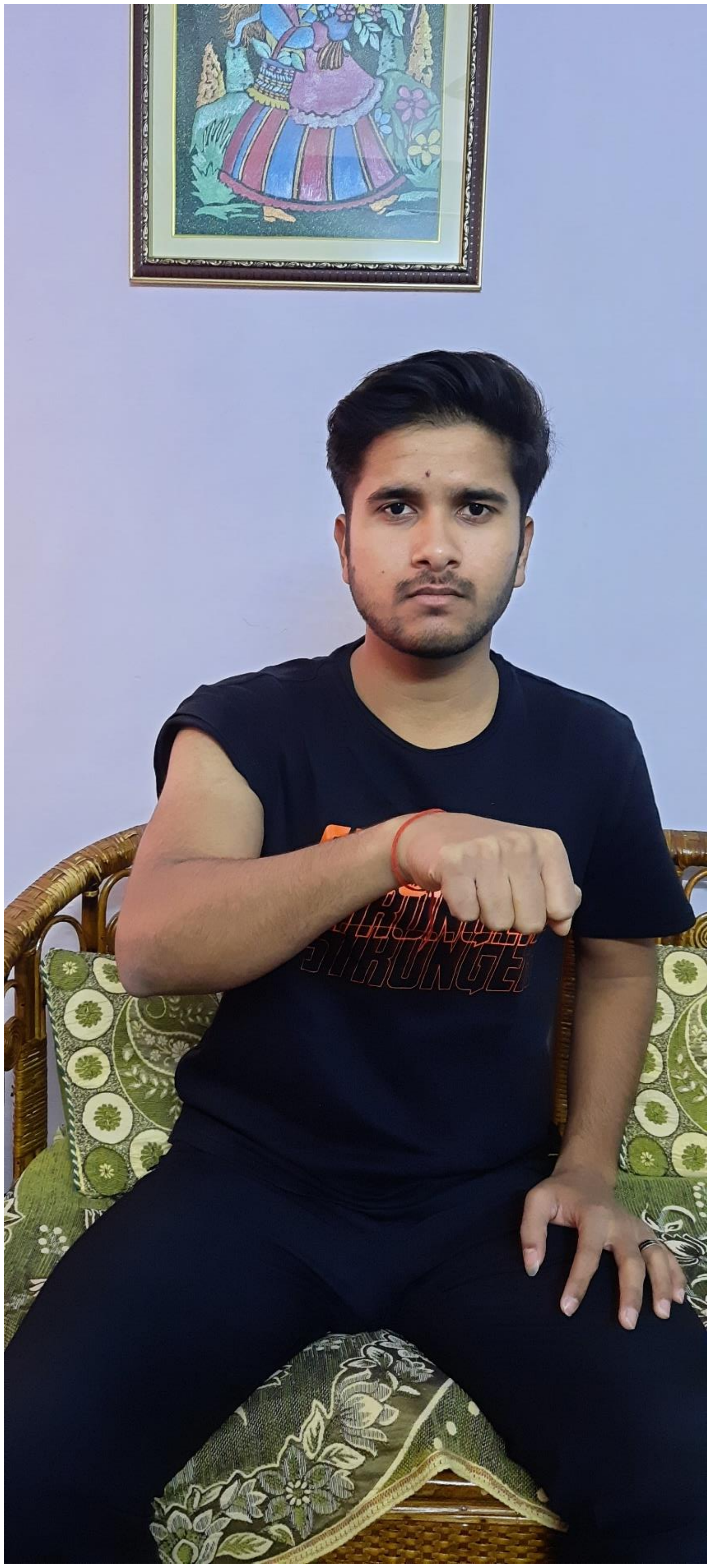


Fig 4.5: Close Gate Input

Output:

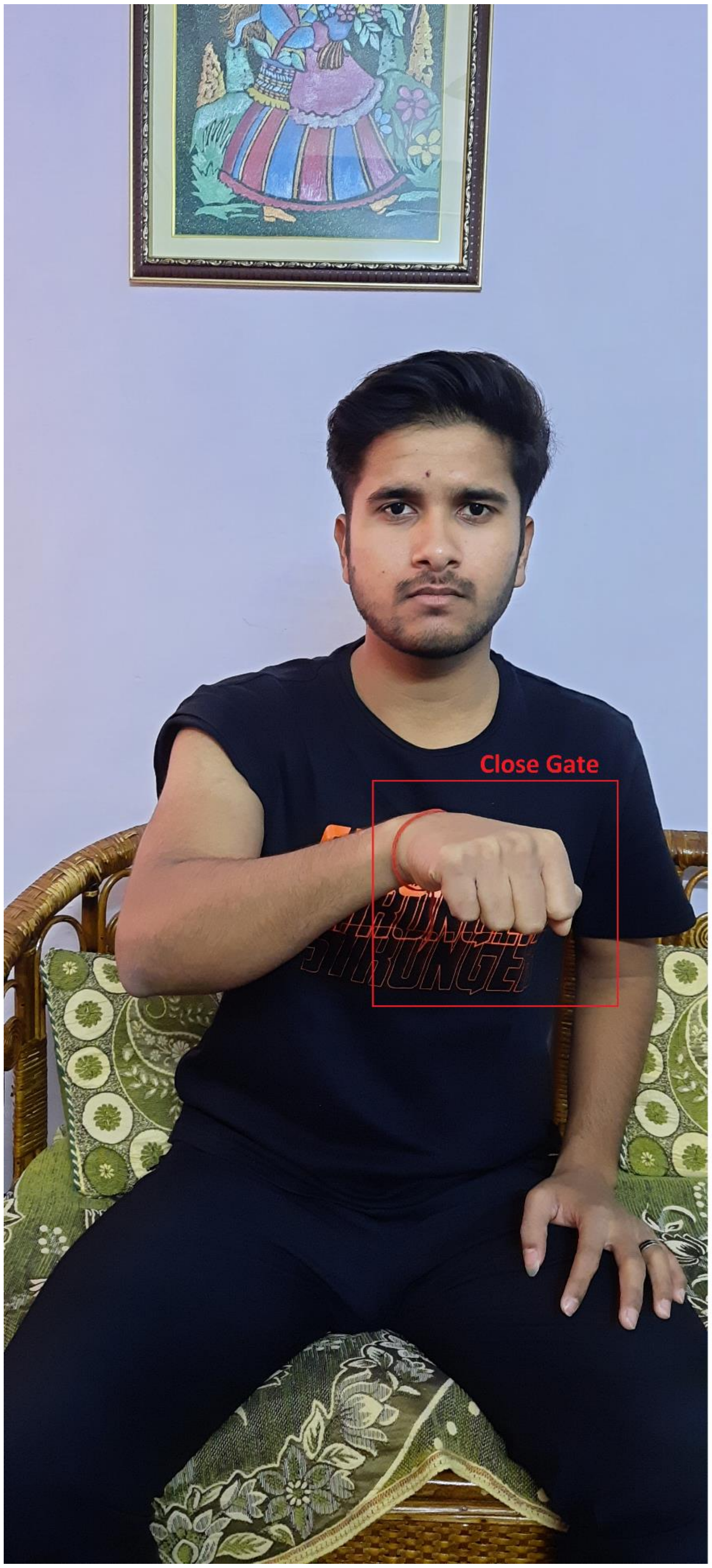


Fig 4.6: Close Gate Ouput

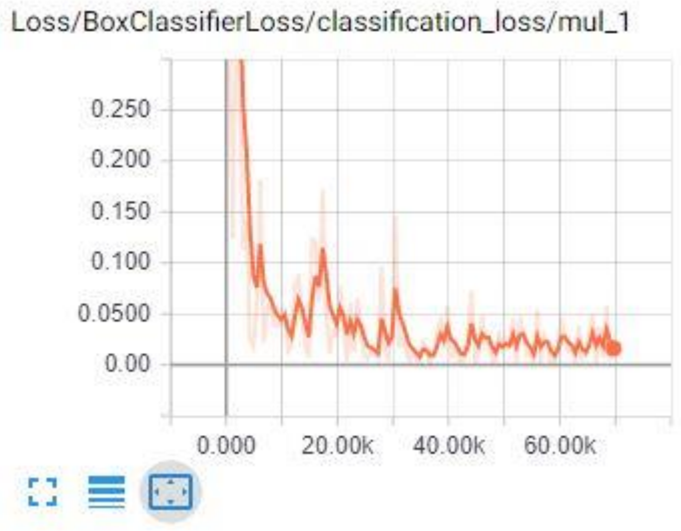


Fig 4.7 : Training Loss Graph

CHAPTER 5

CONCLUSION AND FUTURE WORK

The number of internet user and the absolute population of the entire world is increasing, and there seem to be no horizon to be seen. Under such circumstances, it becomes highly thoughtful to come up with efficient and effective automation processes. Gesture Based Home Automation System has potential to revolutionize the smart home concept in the future, we shall be contributing to it. There always is a scope of improvement, and this model of ours is no different.

There may be ways to improve the performance of the model, such as using more advanced feature extraction techniques, incorporating additional data sources, or using more sophisticated optimization algorithms.

The current system is trained on just five gestures but with right training and effort it has the potential to recognize as many gesture as the user wants. There may be alternative approaches or algorithms that could be used to achieve similar or better results in the fashion recommendation task. It may be interesting to explore these approaches and compare their performance to the current system.

Lot can be done in this area. There is a large scope which could be ventured, and new designs or system could be made to improve the conditions and efficiency of the model and by using more sophisticated algorithms and better data pre-processing we can figure out new dimensions to improve.

CHAPTER 6

APPENDIX

Python Code

```
import os
import glob
import xml.etree.ElementTree as ET
import pandas as pd
import tensorflow as tf
import sys

%cd /content/drive/MyDrive/HomeAutomationSystem/TensorflowModel


# clone the tensorflow models on the colab cloud vm

!git clone --q https://github.com/tensorflow/models.git

# navigate to /models/research folder to compile protos

%cd models/research

# Compile protos.

!protoc object_detection/protos/*.proto --python_out=.

# Install TensorFlow Object Detection API.

!cp object_detection/packages/tf2/setup.py .
!python -m pip install .


# testing the model builder
!python object_detection/builders/model_builder_tf2_test.py


sys.path.insert(1, "/content/drive/MyDrive/HomeAutomationSystem/TensorflowM
odel/models")
```

```
sys.path.insert(2, "/content/drive/MyDrive/HomeAutomationSystem/TensorflowM
odel/models/research")
sys.path.insert(3, "/content/drive/MyDrive/HomeAutomationSystem/TensorflowM
odel/models/research/slim")
sys.path.insert(4,"/content/drive/MyDrive/HomeAutomationSystem/TensorflowMo
del/models/research/object_detection" )


!echo $PYTHONPATH
print(sys.path)

%cd /content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/i
mages/test


!unzip /content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspac
e/images/test/All.zip -d .


#xml_to_csv.py

#adjusted from: https://github.com/datitran/raccoon_dataset
def xml_to_csv(path):
  classes_names = []
  xml_list = []

  for xml_file in glob.glob(path + '/*.xml'):
    tree = ET.parse(xml_file)
    root = tree.getroot()
    for member in root.findall('object'):
      classes_names.append(member[0].text)
      value = (root.find('filename').text  ,
               int(root.find('size')[0].text),
               int(root.find('size')[1].text),
               member[0].text,
               int(member[4][0].text),
               int(member[4][1].text),
               int(member[4][2].text),
               int(member[4][3].text))
      xml_list.append(value)
  column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'x
max', 'ymax']
  xml_df = pd.DataFrame(xml_list, columns=column_name)
  classes_names = list(set(classes_names))
  classes_names.sort()
  return xml_df, classes_names
```

```
for label_path in ['test_labels', 'train_labels']:
  image_path = os.path.join('/content/drive/MyDrive/HomeAutomationSystem/Te
nsorflowModel/workspace/Labels/' + label_path)
  xml_df, classes = xml_to_csv(image_path)
  xml_df.to_csv(f'{label_path}.csv', index=None)
  print(f'Successfully converted {label_path} xml to csv.')

'''label_map_path = os.path.join("label_map.pbtxt")
pbtxt_content = ""

for i, class_name in enumerate(classes):
    pbtxt_content = (
        pbtxt_content
        + "item {{\n    id: {0}\n    name: '{1}'\n}}\n\n".format(i + 1, cla
ss_name)
    )
pbtxt_content = pbtxt_content.strip()
with open(label_map_path, "w") as f:
    f.write(pbtxt_content)
    print('Successfully created label_map.pbtxt ')  '''


pip install tensorflow-object-detection-api


#generate_tfrecord.py

from __future__ import division
from __future__ import print_function
from __future__ import absolute_import

import os
import io
import pandas as pd
import tensorflow as tf
import argparse

from PIL import Image
from tqdm import tqdm
from object_detection.utils import dataset_util
from collections import namedtuple, OrderedDict


def __split(df, group):
   data = namedtuple('data', ['filename', 'object'])
   gb = df.groupby(group)
```

```
    return [data(filename, gb.get_group(x)) for filename, x in zip(gb.groups
.keys(), gb.groups)]


def create_tf_example(group, path, class_dict):
    with tf.io.gfile.GFile(os.path.join(path, '{}'.format(group.filename)),
'rb') as fid:
        encoded_jpg = fid.read()
    encoded_jpg_io = io.BytesIO(encoded_jpg)
    image = Image.open(encoded_jpg_io)
    width, height = image.size

    filename = group.filename.encode('utf8')
    image_format = b'jpg'
    xmins = []
    xmaxs = []
    ymins = []
    ymaxs = []
    classes_text = []
    classes = []

    for index, row in group.object.iterrows():
        if set(['xmin_rel', 'xmax_rel', 'ymin_rel', 'ymax_rel']).issubset(set
(row.index)):
            xmin = row['xmin_rel']
            xmax = row['xmax_rel']
            ymin = row['ymin_rel']
            ymax = row['ymax_rel']

        elif set(['xmin', 'xmax', 'ymin', 'ymax']).issubset(set(row.index)):
            xmin = row['xmin'] / width
            xmax = row['xmax'] / width
            ymin = row['ymin'] / height
            ymax = row['ymax'] / height

        xmins.append(xmin)
        xmaxs.append(xmax)
        ymins.append(ymin)
        ymaxs.append(ymax)
        classes_text.append(str(row['class']).encode('utf8'))
        classes.append(class_dict[str(row['class'])])

    tf_example = tf.train.Example(features=tf.train.Features(
        feature={
            'image/height': dataset_util.int64_feature(height),
            'image/width': dataset_util.int64_feature(width),
            'image/filename': dataset_util.bytes_feature(filename),
            'image/source_id': dataset_util.bytes_feature(filename),
            'image/encoded': dataset_util.bytes_feature(encoded_jpg),
            'image/format': dataset_util.bytes_feature(image_format),
            'image/object/bbox/xmin': dataset_util.float_list_feature(xmins)
,
            'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs)
,
            'image/object/bbox/ymin': dataset_util.float_list_feature(ymins)
,
            'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs)
,
            'image/object/class/text': dataset_util.bytes_list_feature(class
es_text),
            'image/object/class/label': dataset_util.int64_list_feature(clas
ses), }))
   return tf_example


def class_dict_from_pbtxt(pbtxt_path):
   # open file, strip \n, trim lines and keep only
   # lines beginning with id or display_name

   with open(pbtxt_path, 'r', encoding='utf-8-sig') as f:
      data = f.readlines()

   name_key = None
   if any('display_name:' in s for s in data):
      name_key = 'display_name:'
   elif any('name:' in s for s in data):
      name_key = 'name:'

   if name_key is None:
      raise ValueError(
          "label map does not have class names, provided by values with the
 'display_name' or 'name' keys in the contents of the file"
      )

   data = [l.rstrip('\n').strip() for l in data if 'id:' in l or name_key i
n l]

   ids = [int(l.replace('id:', '')) for l in data if l.startswith('id')]
   names = [
       l.replace(name_key, '').replace('"', '').replace("'", '').strip() fo
r l in data
       if l.startswith(name_key)]

   # join ids and display_names into a single dictionary
```

```
    class_dict = {}
    for i in range(len(ids)):
        class_dict[names[i]] = ids[i]

    return class_dict


if __name__ == '__main__':
    parser = argparse.ArgumentParser(
        description='Create a TFRecord file for use with the TensorFlow Obje
ct Detection API.',
        formatter_class=argparse.RawDescriptionHelpFormatter)
    parser.add_argument('csv_input', metavar='csv_input', type=str, help='Pa
th to the CSV input')
    parser.add_argument('pbtxt_input',
                        metavar='pbtxt_input',
                        type=str,
                        help='Path to a pbtxt file containing class ids and
display names')
    parser.add_argument('image_dir',
                        metavar='image_dir',
                        type=str,
                        help='Path to the directory containing all images')
    parser.add_argument('output_path',
                        metavar='output_path',
                        type=str,
                        help='Path to output TFRecord')

    args = parser.parse_args()

    class_dict = class_dict_from_pbtxt(args.pbtxt_input)

    writer = tf.compat.v1.python_io.TFRecordWriter(args.output_path)
    path = os.path.join(args.image_dir)
    examples = pd.read_csv(args.csv_input)
    grouped = __split(examples, 'filename')

    for group in tqdm(grouped, desc='groups'):
        tf_example = create_tf_example(group, path, class_dict)
        writer.write(tf_example.SerializeToString())

    writer.close()
    output_path = os.path.join(os.getcwd(), args.output_path)
    print('Successfully created the TFRecords: {}'.format(output_path))
```

```
#Usage:
#!python generate_tfrecord.py output.csv output_pb.txt /path/to/images outp
ut.tfrecords

#For train.record
!python generate_tfrecord.py train_labels.csv  label_map.pbtxt /content/dri
ve/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/images/imagedatas
et/Training_Images train.record

#For test.record
!python generate_tfrecord.py test_labels.csv label_map.pbtxt  /content/driv
e/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/images/imagedatase
t/Test_Images test.record


#Download the pre-
trained model ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-
8.tar.gz into the data folder & unzip it.

!wget http://download.tensorflow.org/models/object_detection/tf2/20200711/s
sd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
!tar -xzvf ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz


#load tensorboard

%load_ext tensorboard
%tensorboard --
logdir '/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/Trainin
g'


!python setup.py build
!python setup.py install


#pipeline.cofig
model {
  ssd {
    num_classes: 5
    image_resizer {
      fixed_shape_resizer {
        height: 320
        width: 320
      }
    }
    feature_extractor {
      type: "ssd_mobilenet_v2_fpn_keras"
      depth_multiplier: 1.0
      min_depth: 16
      conv_hyperparams {
        regularizer {
          l2_regularizer {
            weight: 3.9999998989515007e-05
          }
        }
        initializer {
          random_normal_initializer {
            mean: 0.0
            stddev: 0.009999999776482582
          }
        }
        activation: RELU_6
        batch_norm {
          decay: 0.996999979019165
          scale: true
          epsilon: 0.0010000000474974513
        }
      }
      use_depthwise: true
      override_base_feature_extractor_hyperparams: true
      fpn {
        min_level: 3
        max_level: 7
        additional_layer_depth: 128
      }
    }
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
        use_matmul_gather: true
      }
    }
    similarity_calculator {
      iou_similarity {
      }
    }
    box_predictor {
      weight_shared_convolutional_box_predictor {
        conv_hyperparams {
          regularizer {
            l2_regularizer {
              weight: 3.9999998989515007e-05
            }
          }
          initializer {
            random_normal_initializer {
              mean: 0.0
              stddev: 0.009999999776482582
            }
          }
          activation: RELU_6
          batch_norm {
            decay: 0.996999979019165
            scale: true
            epsilon: 0.0010000000474974513
          }
        }
        depth: 128
        num_layers_before_predictor: 4
        kernel_size: 3
        class_prediction_bias_init: -4.599999904632568
        share_prediction_tower: true
        use_depthwise: true
      }
    }
    anchor_generator {
      multiscale_anchor_generator {
        min_level: 3
        max_level: 7
        anchor_scale: 4.0
        aspect_ratios: 1.0
        aspect_ratios: 2.0
        aspect_ratios: 0.5
        scales_per_octave: 2
      }
    }
    post_processing {
      batch_non_max_suppression {
        score_threshold: 9.99999993922529e-09
        iou_threshold: 0.6000000238418579
        max_detections_per_class: 100
        max_total_detections: 100
        use_static_shapes: false
      }
      score_converter: SIGMOID
    }
    normalize_loss_by_num_matches: true
    loss {
      localization_loss {
        weighted_smooth_l1 {
        }
      }
      classification_loss {
        weighted_sigmoid_focal {
          gamma: 2.0
          alpha: 0.25
        }
      }
      classification_weight: 1.0
      localization_weight: 1.0
    }
    encode_background_as_zeros: true
    normalize_loc_loss_by_codesize: true
    inplace_batchnorm_update: true
    freeze_batchnorm: false
  }
}
train_config {
  batch_size: 8
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    random_crop_image {
      min_object_covered: 0.0
      min_aspect_ratio: 0.75
      max_aspect_ratio: 3.0
      min_area: 0.75
      max_area: 1.0
      overlap_thresh: 0.0
    }
  }
  sync_replicas: true
  optimizer {
    momentum_optimizer {
      learning_rate {
        cosine_decay_learning_rate {
          learning_rate_base: 0.07999999821186066
          total_steps: 50000
          warmup_learning_rate: 0.026666000485420227
          warmup_steps: 1000
        }
      }
      momentum_optimizer_value: 0.8999999761581421
    }
    use_moving_average: false
  }
  fine_tune_checkpoint:
"/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/ssd_
mobilenet_v2_fpnlite_320x320_coco17_tpu-8/checkpoint"
  num_steps: 10000
  startup_delay_steps: 0.0
  replicas_to_aggregate: 8
  max_number_of_boxes: 100
  unpad_groundtruth_tensors: false
  fine_tune_checkpoint_type: "detection"
  fine_tune_checkpoint_version: V2
}
train_input_reader {
  label_map_path:
"/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/labe
l_map.pbtxt"
  tf_record_input_reader {
    input_path:
"/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/trai
n.record"
  }
}
eval_config {
  metrics_set: "coco_detection_metrics"
  use_moving_averages: false
}
eval_input_reader {
  label_map_path:
"/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/labe
l_map.pbtxt"
  shuffle: false
  num_epochs: 1
  tf_record_input_reader {
    input_path:
"/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/workspace/test
.record"
  }
}



# Run the command below from the content/models/research/object_detection d
irectory
"""
PIPELINE_CONFIG_PATH=path/to/pipeline.config
MODEL_DIR=path to training checkpoints directory
NUM_TRAIN_STEPS=50000
SAMPLE_1_OF_N_EVAL_EXAMPLES=1

python model_main_tf2.py -- \
  --model_dir=$MODEL_DIR --num_train_steps=$NUM_TRAIN_STEPS \
  --sample_1_of_n_eval_examples=$SAMPLE_1_OF_N_EVAL_EXAMPLES \
  --pipeline_config_path=$PIPELINE_CONFIG_PATH \
  --alsologtostderr
"""

!python model_main_tf2.py --
pipeline_config_path=/content/drive/MyDrive/HomeAutomationSystem/Tensorflow
Model/workspace/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.config --
model_dir=/content/drive/MyDrive/HomeAutomationSystem/TensorflowModel/Train
ing --alsologtostderr
```

```
#label_map.pbtxt

item {
      name:'Open_Gate'
      id:1
}
item {
      name:'Close_Gate'
      id:2
}
item {
      name:'Open_Window'
      id:3
}
item {
      name:'Close_Window'
      id:4
}
item {
      name:'Start_Fan'
      id:5
}
```

```
#An annotation generated XML file
<annotation>
      <folder>collectedImages</folder>
      <filename>Close_Gate_1.jpg</filename>
      <path>C:\Users\Abhishek\Desktop\RealTimeObjectDetection\Tensorflow\wo
rkspace\images\collectedImages\Close_Gate_1.jpg</path>
      <source>
            <database>Unknown</database>
      </source>
      <size>
            <width>4624</width>
            <height>2084</height>
            <depth>3</depth>
      </size>
      <segmented>0</segmented>
      <object>
            <name>Close_Gate</name>
            <pose>Unspecified</pose>
            <truncated>0</truncated>
            <difficult>0</difficult>
            <bndbox>
                  <xmin>2435</xmin>
                  <ymin>960</ymin>
                  <xmax>2941</xmax>
                  <ymax>1591</ymax>
            </bndbox>
      </object>
</annotation>
```

REFERENCES

[1] M. -C. Su, J. -H. Chen, A. M. Arifai, S. -Y. Tsai and H. -H. Wei, "Smart Living: An Interactive Control System for Household Appliances," in IEEE Access, vol. 9, pp. 14897-14904, 2021, doi: 10.1109/ACCESS.2021.3051253.

[2] A. Cenedese, G. A. Susto, G. Belgioioso, G. I. Cirillo and F. Fraccaroli, "Home Automation Oriented Gesture Classification From Inertial Measurements," in IEEE Transactions on Automation Science and Engineering, vol. 12, no. 4, pp. 1200-1210, Oct. 2015, doi: 10.1109/TASE.2015.2473659.

[3] Marcel Villanueva, “Using gestures to interact with home automation systems”, Jan. 2019, doi: 10.5185/amp.2019.1443

[4] Nicholas Renotte : Real Time Sign Language Detection with Tensorflow Object Detection and Python | Deep Learning SSD, YouTube

[5] Training a model for custom object detection(TF 2.X), www.techzizou.com

[6] Smruti Kshirsagar , Srushti Sachdev , Navjyot Singh , Anushka Tiwari , Yugchhaya Dhote ,Gesture Controlled Home Automation System Using Internet of Things, “Real Time Databases for Applications”, International Research Journal of Engineering and Technology (IRJET) Real Time Databases for Applications, Volume: 04 Issue: 06 | June -2017

[7] GONGFA LI(Member, IEEE), HAO WU1 , GUOZHANG JIANG, SHUANG XU AND HONGHAI LIU, (Senior Member, IEEE), Dynamic Gesture Recognition in the Internet of Things

[8] LabelImg for Labeling Object Detection Data, https://blog.roboflow.com/labelimg/

[9] TensorFlow Setup: https://www.tensorflow.org/install

[10] Python Setup and Docs : https://www.python.org/downloads/ , https://www.python.org/doc

[11] Stack Overflow, https://stackoverflow.com

BIODATA

Name	: Abhishek
Registration No.	: 19BCI7058
Mobile Number	: 9863145336
E-mail	: abhishek.19bci7058@vitap.ac.in

Name	: Divyansh
Registration No.	: 19BCD7171
Mobile Number	: 898845719
E-mail	: divyansh.19bcd7171@vitap.ac.in

Name	: Prashant Fachara
Registration No.	: 19BCE7271
Mobile Number	: 9727860378
E-mail	: prashant.19bce7271@vitap.ac.in

Thank You