VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

PROJECT WORK-4 REPORT
on

REAL TIME SIGN LANGUAGE DETECTION

Submitted by

SHIVA PRANEETH KODALI (1BM18CS100)
SHREYA LADDHA (1BM18CS103)
TANYA MISHRA (1BM18CS117)
VK PRITHVIK ANIKETH (1BM18CS119)

Under the Guidance of
Vikranth BM
Assistant Professor, BMSCE

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B. M. S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Mar-2021 to Jun-2021

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project work entitled "**REAL TIME SIGN LANGUAGE DETECTION**" carried out by **SHIVA PRANEETH KODALI (1BM18CS100), SHREYA LADDHA (1BM18CS103), TANYA MISHRA (1BM18CS117) AND VK PRITHVIK ANIKETH (1BM18CS119)** who are bonafide students of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visveswaraiah Technological University, Belgaum during the year 2021. The project report has been approved as it satisfies the academic requirements in respect of **Project Work-4 (20CS6PWPW4)** work prescribed for the said degree.

Signature of the Guide	Signature of the HOD
Vikranth BM	Dr. Umadevi V
Assistant Professor	Associate Prof. & Head, Dept. of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

External Viva

Name of the Examiner	Signature with date
1.______________________________	_________________________
2. ______________________________	__________________________

B. M. S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECALARATION

We, SHIVA PRANEETH KODALI (1BM18CS100), SHREYA LADDHA (1BM18CS103), TANYA MISHRA (1BM18CS117) AND VK PRITHVIK ANIKETH (1BM18CS119), students of 5th Semester, B.E, Department of Computer Science and Engineering, B. M. S. College of Engineering, Bangalore, here by declare that, this Project Work-1entitled "Project Title" has been carried out by us under the guidance of Dr. Selva kumar S, Assistant Professor, Department of CSE, B. M. S. College of Engineering, Bangalore during the academic semester Mar-2021-Jun-2021

We also declare that to the best of our knowledge and belief, the development reported here is not from part of any other report by any other students.

Signature

SHIVA PRANEETH KODALI (1BM18CS100)
SHREYA LADDHA (1BM18CS103)
TANYA MISHRA (1BM18CS117)
VK PRITHVIK ANIKETH (1BM18CS119)

1. Introduction

In the current situation, where all of us are living through a global pandemic, online communication has become the new normal standard for communication. It becomes very hard for mute people to communicate via video call or any online means and it's not always feasible for a third party to be present and translate. A real-time sign language translator is an important milestone in facilitating communication between the deaf community and the general public. A real-time sign language translator is an important milestone in facilitating communication between the deaf community and the general public.

Sign language recognition is a problem that has been addressed in research for years. However, we are still far from finding a complete solution available in our society. The detector aims to develop algorithms and methods to correctly identify a sequence of produced signs and to understand their meaning. Research has thus far focused on identifying optimal features and classification methods to correctly label a given sign from a set of possible signs. However, sign language is far more than just a collection of well specified gestures.

Among the works developed to address this problem, the majority of them have been based on basically two approaches: contact-based systems, such as sensor gloves; or vision-based systems, using only cameras. The latter is way cheaper and the boom of deep learning makes it more appealing.

In real-time, it is highly essential to have an autonomous translator that can process the images and recognize the signs very fast at the speed of streaming images.

2. Problem Definition and Algorithm

2.1 Task Definition

Since ages communication has served as a medium to build relationships, know people, understand technology and allow rapid growth and development on a global basis. Normal people can communicate their thoughts and ideas to others through speech. The most used method of communication for the hearing impaired community is the use of sign language. This project basically provides the mute people a medium for ease of communication.

2.2 Algorithm Definition

MODEL USED:

TENSORFLOW MODEL ZOO

TensorFlow model Zoo provide a collection of detection models pre-trained on the COCO dataset, the Kitti dataset, the Open Images dataset. These models can be useful for out-of-the-box inference if we are interested in categories already in those datasets

COCO Dataset

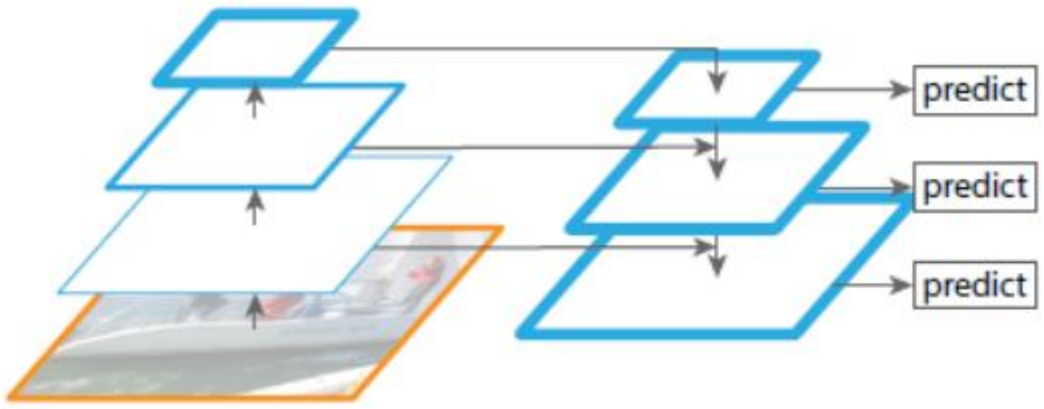
The Common Objects in Context (COCO) dataset is one of the most popular open source object recognition databases used to train deep learning programs.

TENSORFLOW MODEL ZOO- SSD_MOBILENET_V2_FPNLite

We have considered SSD-mobileNet V2 Trained on MS-COCO Data which was Released in 2019, this model is a single-stage object detection model that goes straight from image pixels to bounding box coordinates and class probabilities. This model is part of the Tensorflow object detection API.

In SSD-mobileNetV2 we have taken FPNlite that stands for Feature Pyramid Network. It's a subnetwork which outputs feature maps of different resolutions.

FPN- Feature Pyramid Network



- It combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections.

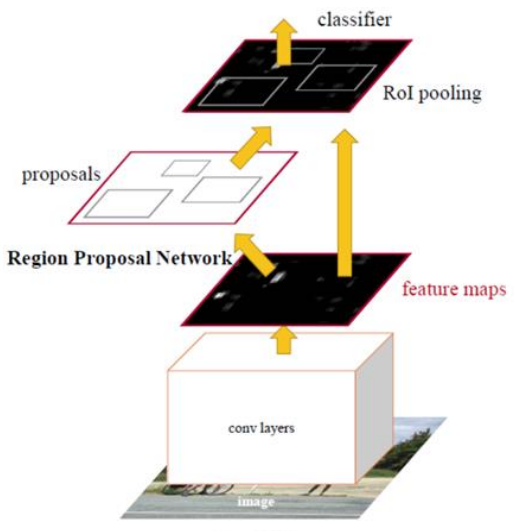
- This feature pyramid has rich semantics at all levels and is built quickly from a single input image scale, thereby without sacrificing representational power, speed, or memory.

- In the original detection network in Faster R-CNN, a single-scale feature map is used.

- Here, to detect the object, ROIs (Region of interest) of different scales are needed to be assigned to the pyramid levels.

Faster R-CNN

In Faster R-CNN both region proposal generation and objection detection tasks are all done by the same conv networks. With such a design, object detection is much faster.



Faster R-CNN

1. First, the picture goes through conv layers and feature maps are extracted.

2. Then a sliding window is used in RPN for each location over the feature map.

3. For each location, k (k=9) anchor boxes are used (3 scales of 128, 256 and 512, and 3 aspect ratios of 1:1, 1:2, 2:1) for generating region proposals.

4. A *cls* layer outputs *2k* scores whether there is an object or not for *k* boxes.

5. A *reg* layer outputs *4k* for the coordinates (box center coordinates, width and height) of *k* boxes.

6. With a size of *W*×*H* feature map, there are *WHk* anchors in total.

3. Experimental Evaluation

3.1 Methodology

Prerequisites to install Tensor flow Object detection API

> Python 3.7

> Anaconda: Created a new Anaconda virtual environment

> Install the TensorFlow PIP package

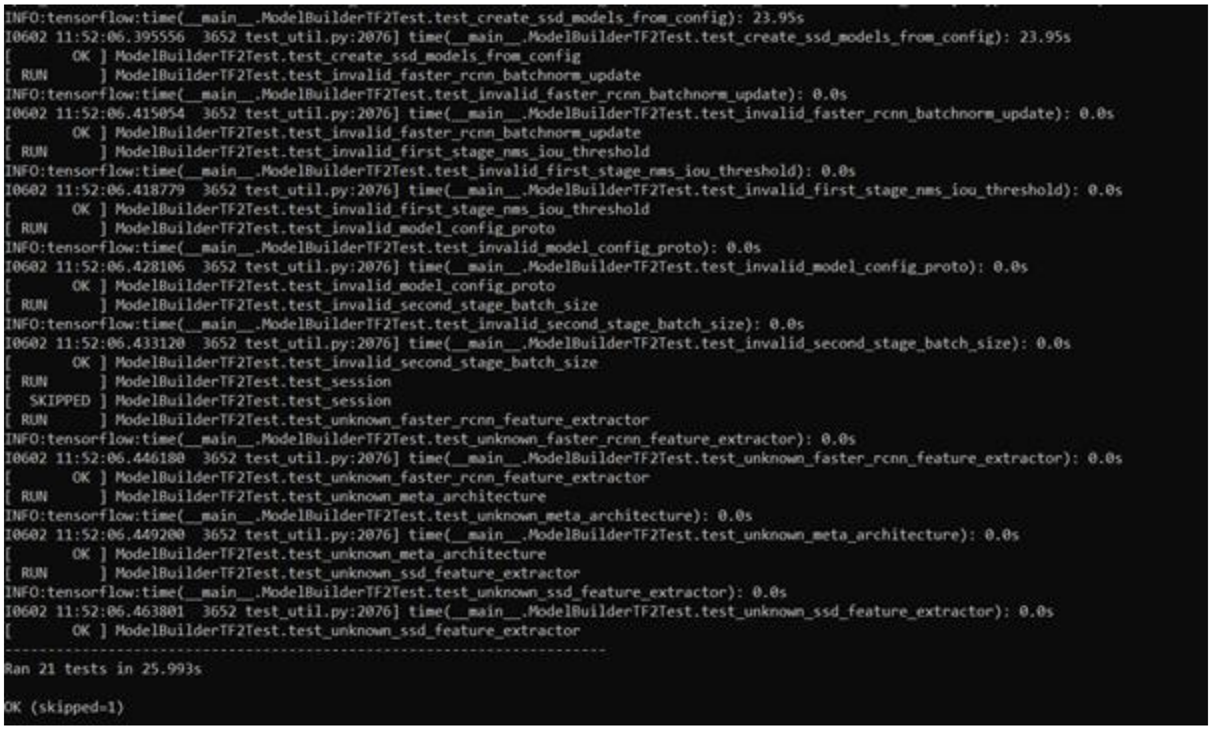
> Download models from Git to clone TensorFlow Models repository

> Install Protobufs to configure model and training parameters

> Install COCO API Installation

> Install Object Detection API

Test the installation



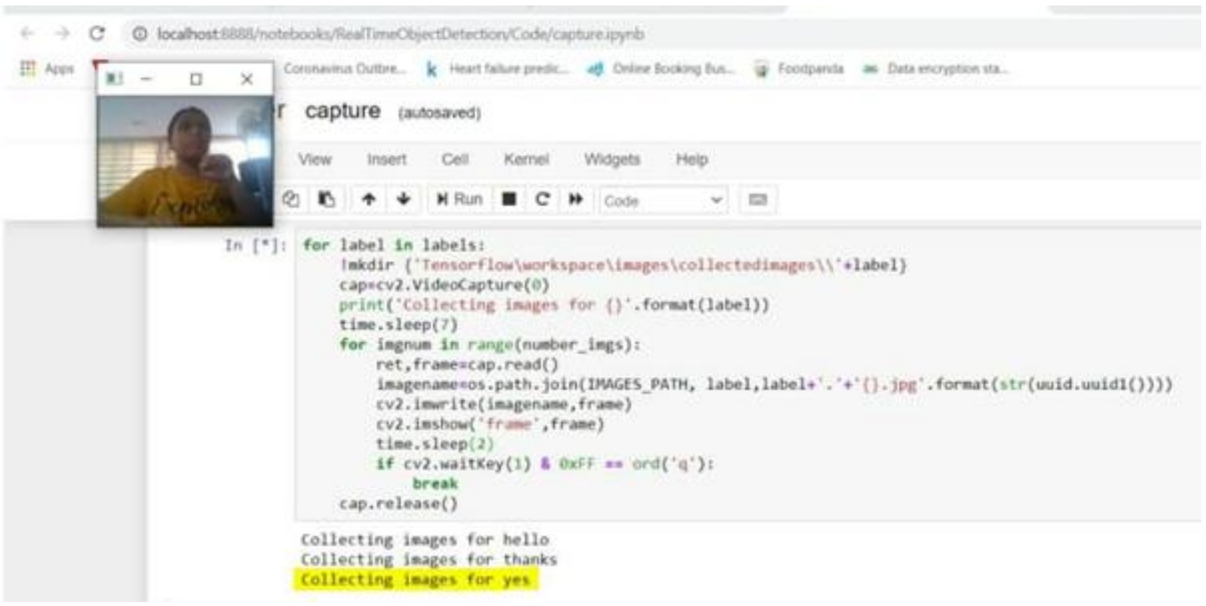
```
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_create_ssd_models_from_config): 23.95s
I0602 11:52:06.395556  3652 test_util.py:2076] time(__main__.ModelBuilderTF2Test.test_create_ssd_models_from_config): 23.95s
[       OK ] ModelBuilderTF2Test.test_create_ssd_models_from_config
[ RUN      ] ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update): 0.0s
I0602 11:52:06.415054  3652 test_util.py:2076] time(__main__.ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update): 0.0s
[       OK ] ModelBuilderTF2Test.test_invalid_faster_rcnn_batchnorm_update
[ RUN      ] ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold): 0.0s
I0602 11:52:06.418779  3652 test_util.py:2076] time(__main__.ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold): 0.0s
[       OK ] ModelBuilderTF2Test.test_invalid_first_stage_nms_iou_threshold
[ RUN      ] ModelBuilderTF2Test.test_invalid_model_config_proto
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_model_config_proto): 0.0s
I0602 11:52:06.428106  3652 test_util.py:2076] time(__main__.ModelBuilderTF2Test.test_invalid_model_config_proto): 0.0s
[       OK ] ModelBuilderTF2Test.test_invalid_model_config_proto
[ RUN      ] ModelBuilderTF2Test.test_invalid_second_stage_batch_size
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_invalid_second_stage_batch_size): 0.0s
I0602 11:52:06.433120  3652 test_util.py:2076] time(__main__.ModelBuilderTF2Test.test_invalid_second_stage_batch_size): 0.0s
[       OK ] ModelBuilderTF2Test.test_invalid_second_stage_batch_size
[ RUN      ] ModelBuilderTF2Test.test_session
[  SKIPPED ] ModelBuilderTF2Test.test_session
[ RUN      ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
I0602 11:52:06.446180  3652 test_util.py:2076] time(__main__.ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_faster_rcnn_feature_extractor
[ RUN      ] ModelBuilderTF2Test.test_unknown_meta_architecture
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
I0602 11:52:06.449200  3652 test_util.py:2076] time(__main__.ModelBuilderTF2Test.test_unknown_meta_architecture): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_meta_architecture
[ RUN      ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
INFO:tensorflow:time(__main__.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
I0602 11:52:06.463801  3652 test_util.py:2076] time(__main__.ModelBuilderTF2Test.test_unknown_ssd_feature_extractor): 0.0s
[       OK ] ModelBuilderTF2Test.test_unknown_ssd_feature_extractor
----------------------------------------------------------------------
Ran 21 tests in 25.993s

OK (skipped=1)
```

PREPARING THE DATASET

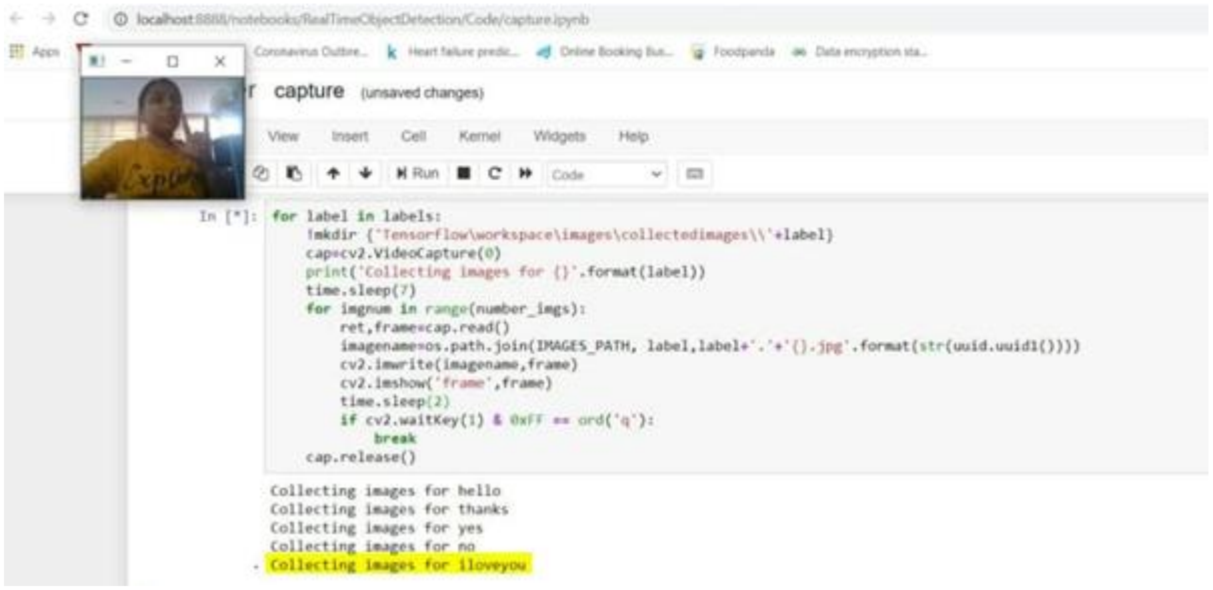
Collected all the images to be used to test our model using the following code which uses OpenCV to capture the images.

```
import cv2
import os
import time
import uuid
IMAGES_PATH='Tensorflow/workspace/images/collectedimages'
labels=['hello','thanks','yes','no','iloveyou']
number_imgs=15
for label in labels:

get_ipython().system("mkdir{'Tensorflow\\workspace\\images\\collectedimages\\\\'
+        label}")
    cap=cv2.VideoCapture(0)
    print('Collecting images for {}'.format(label))
    time.sleep(5)
    for imgnum in range(number_imgs):
        ret,frame=cap.read()
        imagename=os.path.join(IMAGES_PATH,
label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
        cv2.imwrite(imagename,frame)
        cv2.imshow('frame',frame)
        time.sleep(2)
        if cv2.waitKey(1) & 0xFF == ord('q'):
          break
    cap.release()
```

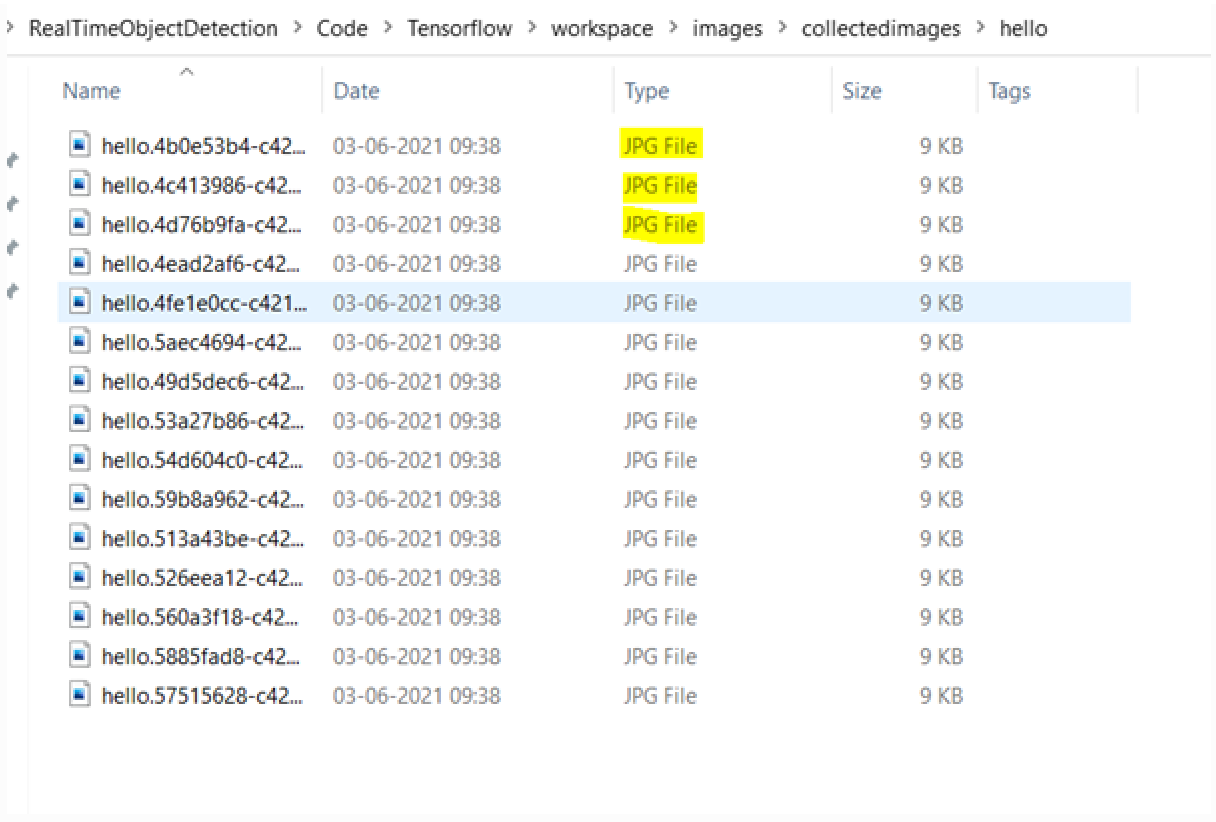
Captured images for deep learning using webcam and OpenCV.



```
In [*]: for label in labels:
            !mkdir {'Tensorflow\workspace\images\collectedimages\\'+label}
            cap=cv2.VideoCapture(0)
            print('Collecting images for {}'.format(label))
            time.sleep(7)
            for imgnum in range(number_imgs):
                ret,frame=cap.read()
                imagename=os.path.join(IMAGES_PATH, label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
                cv2.imwrite(imagename,frame)
                cv2.imshow('frame',frame)
                time.sleep(2)
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break
            cap.release()

        Collecting images for hello
        Collecting images for thanks
        Collecting images for yes
```



```
In [*]: for label in labels:
            !mkdir {'Tensorflow\workspace\images\collectedimages\\'+label}
            cap=cv2.VideoCapture(0)
            print('Collecting images for {}'.format(label))
            time.sleep(7)
            for imgnum in range(number_imgs):
                ret,frame=cap.read()
                imagename=os.path.join(IMAGES_PATH, label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
                cv2.imwrite(imagename,frame)
                cv2.imshow('frame',frame)
                time.sleep(2)
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break
            cap.release()

        Collecting images for hello
        Collecting images for thanks
        Collecting images for yes
        Collecting images for no
        Collecting images for iloveyou
```

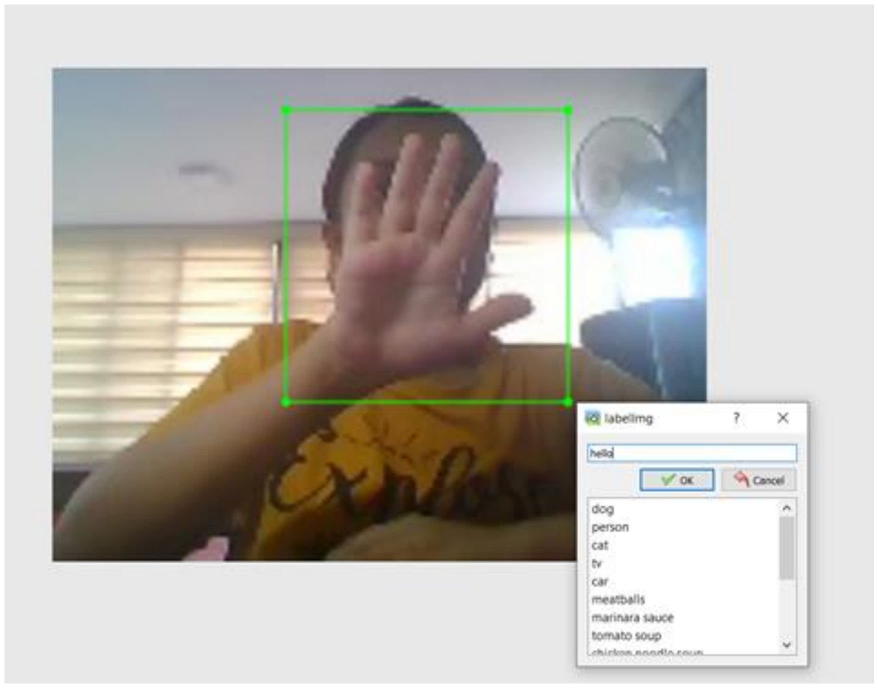


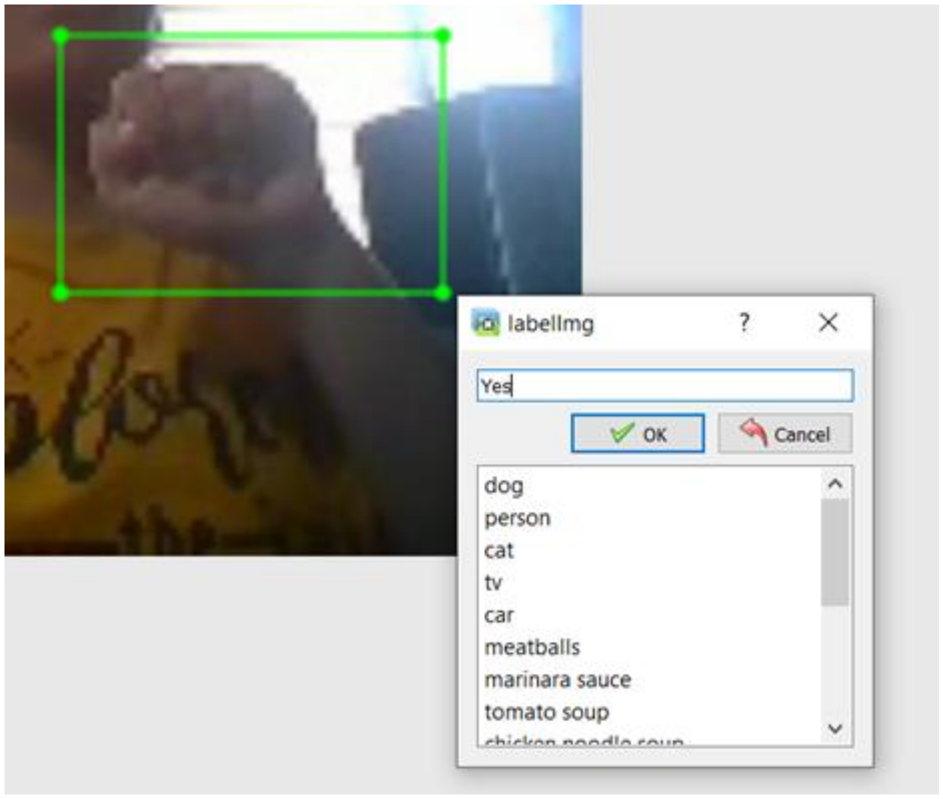
› RealTimeObjectDetection › Code › Tensorflow › workspace › images › collectedimages › hello

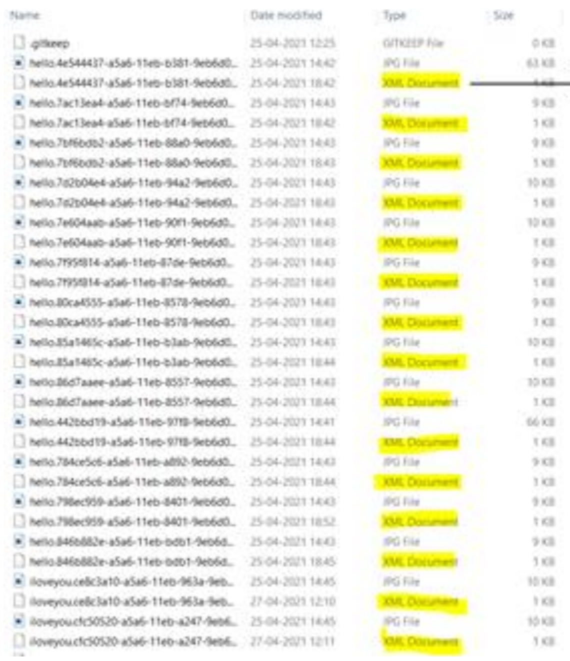
Name	Date	Type	Size	Tags
hello.4b0e53b4-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.4c413986-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.4d76b9fa-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.4ead2af6-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.4fe1e0cc-c421...	03-06-2021 09:38	JPG File	9 KB	
hello.5aec4694-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.49d5dec6-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.53a27b86-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.54d604c0-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.59b8a962-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.513a43be-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.526eea12-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.560a3f18-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.5885fad8-c42...	03-06-2021 09:38	JPG File	9 KB	
hello.57515628-c42...	03-06-2021 09:38	JPG File	9 KB	

Label Image

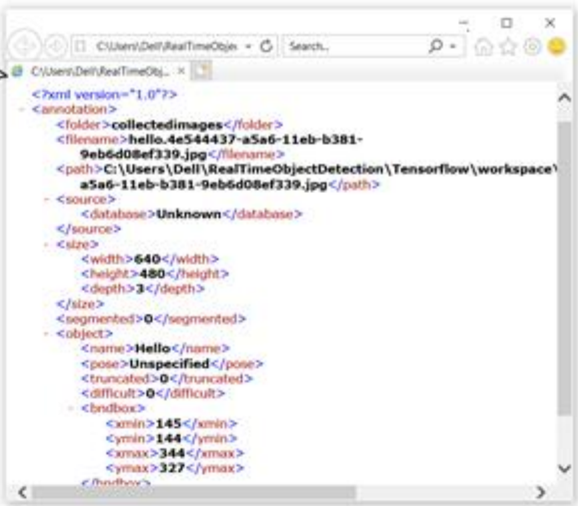
Labelled images using LabelImg and created the xml records for each image.







Name	Date modified	Type	Size
.gitkeep	25-04-2021 12:25	GITKEEP File	0 KB
hello.4e544437-a5a6-11eb-b381-9eb6d0...	25-04-2021 14:42	JPG File	63 KB
hello.4e544437-a5a6-11eb-b381-9eb6d0...	25-04-2021 18:42	XML Document	1 KB
hello.7ac13ea4-a5a6-11eb-bf74-9eb6d0...	25-04-2021 14:43	JPG File	9 KB
hello.7ac13ea4-a5a6-11eb-bf74-9eb6d0...	25-04-2021 18:42	XML Document	1 KB
hello.7bf6bdb2-a5a6-11eb-88a0-9eb6d0...	25-04-2021 14:43	JPG File	9 KB
hello.7bf6bdb2-a5a6-11eb-88a0-9eb6d0...	25-04-2021 18:43	XML Document	1 KB
hello.7d2b04e4-a5a6-11eb-94a2-9eb6d0...	25-04-2021 14:43	JPG File	10 KB
hello.7d2b04e4-a5a6-11eb-94a2-9eb6d0...	25-04-2021 18:43	XML Document	1 KB
hello.7e604aab-a5a6-11eb-90f1-9eb6d0...	25-04-2021 14:43	JPG File	10 KB
hello.7e604aab-a5a6-11eb-90f1-9eb6d0...	25-04-2021 18:43	XML Document	1 KB
hello.7f95f814-a5a6-11eb-87de-9eb6d0...	25-04-2021 14:43	JPG File	9 KB
hello.7f95f814-a5a6-11eb-87de-9eb6d0...	25-04-2021 18:43	XML Document	1 KB
hello.80ca4555-a5a6-11eb-8578-9eb6d0...	25-04-2021 14:43	JPG File	9 KB
hello.80ca4555-a5a6-11eb-8578-9eb6d0...	25-04-2021 18:43	XML Document	1 KB
hello.85a1465c-a5a6-11eb-b3ab-9eb6d0...	25-04-2021 14:43	JPG File	10 KB
hello.85a1465c-a5a6-11eb-b3ab-9eb6d0...	25-04-2021 18:44	XML Document	1 KB
hello.86d7aaee-a5a6-11eb-8557-9eb6d0...	25-04-2021 14:43	JPG File	10 KB
hello.86d7aaee-a5a6-11eb-8557-9eb6d0...	25-04-2021 18:44	XML Document	1 KB
hello.442bbd19-a5a6-11eb-97f8-9eb6d0...	25-04-2021 14:41	JPG File	66 KB
hello.442bbd19-a5a6-11eb-97f8-9eb6d0...	25-04-2021 18:44	XML Document	1 KB
hello.784ce5c6-a5a6-11eb-a892-9eb6d0...	25-04-2021 14:43	JPG File	9 KB
hello.784ce5c6-a5a6-11eb-a892-9eb6d0...	25-04-2021 18:44	XML Document	1 KB
hello.798ec959-a5a6-11eb-8401-9eb6d0...	25-04-2021 14:43	JPG File	9 KB
hello.798ec959-a5a6-11eb-8401-9eb6d0...	25-04-2021 18:52	XML Document	1 KB
hello.846b882e-a5a6-11eb-bdb1-9eb6d...	25-04-2021 14:43	JPG File	9 KB
hello.846b882e-a5a6-11eb-bdb1-9eb6d...	25-04-2021 18:45	XML Document	1 KB
iloveyou.ce8c3a10-a5a6-11eb-963a-9eb...	25-04-2021 14:45	JPG File	10 KB
iloveyou.ce8c3a10-a5a6-11eb-963a-9eb...	27-04-2021 12:10	XML Document	1 KB
iloveyou.cfc50520-a5a6-11eb-a247-9eb6...	25-04-2021 14:45	JPG File	10 KB
iloveyou.cfc50520-a5a6-11eb-a247-9eb6...	27-04-2021 12:11	XML Document	1 KB



```
<?xml version="1.0"?>
- <annotation>
    <folder>collectedimages</folder>
    <filename>hello.4e544437-a5a6-11eb-b381-
      9eb6d08ef339.jpg</filename>
    <path>C:\Users\Dell\RealTimeObjectDetection\Tensorflow\workspace\
      a5a6-11eb-b381-9eb6d08ef339.jpg</path>
  - <source>
      <database>Unknown</database>
    </source>
  - <size>
      <width>640</width>
      <height>480</height>
      <depth>3</depth>
    </size>
    <segmented>0</segmented>
  - <object>
      <name>Hello</name>
      <pose>Unspecified</pose>
      <truncated>0</truncated>
      <difficult>0</difficult>
    - <bndbox>
        <xmin>145</xmin>
        <ymin>144</ymin>
        <xmax>344</xmax>
        <ymax>327</ymax>
      </bndbox>
```

>Captured 15 images for each Sign, 13 for training and 2 for testing.

Generated TF Records

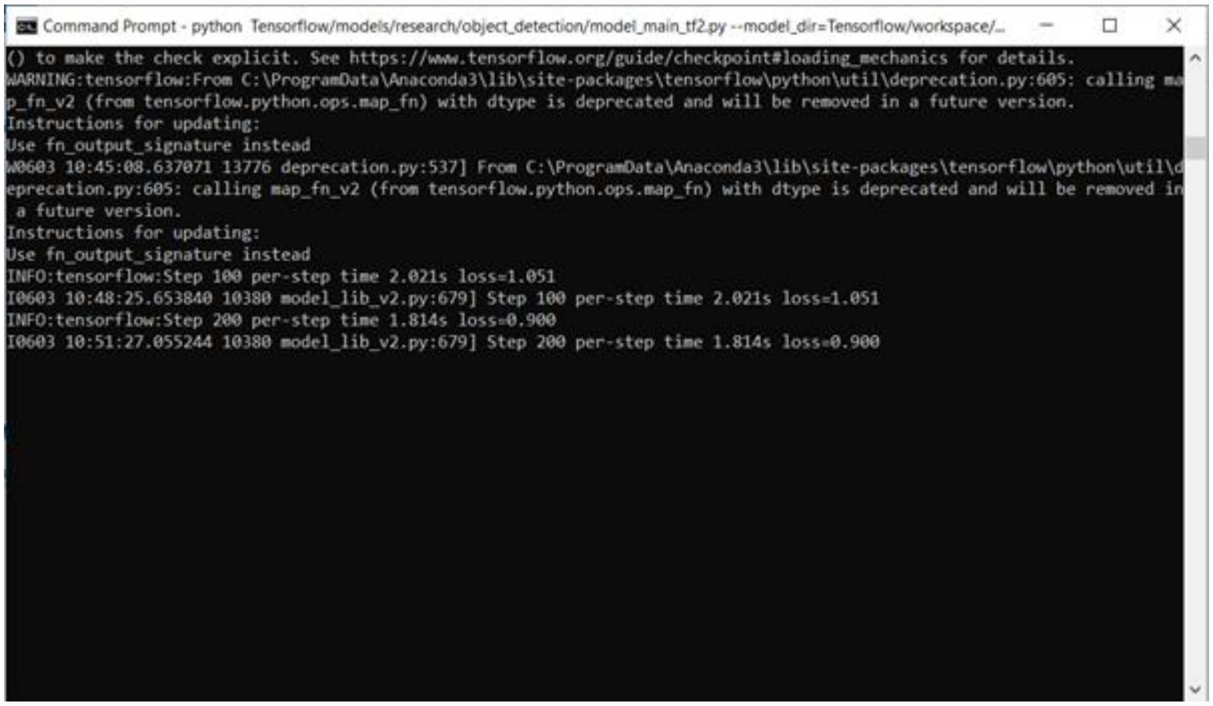
The **TF Record** format is a simple format for storing a sequence of binary **records** as it's the easiest way to understand a message type by the TensorFlow API to train the dataset.

Training Model

As discussed we used TensorFlow Model ZOO ssd_mobnet_v2_fpnLite as our model for object detection which has an in built model_main_v2.py which creates a script from the pipeline.config file path that we provide with changes of our class name and training records path to train the model for 5000 steps.

STEP 100

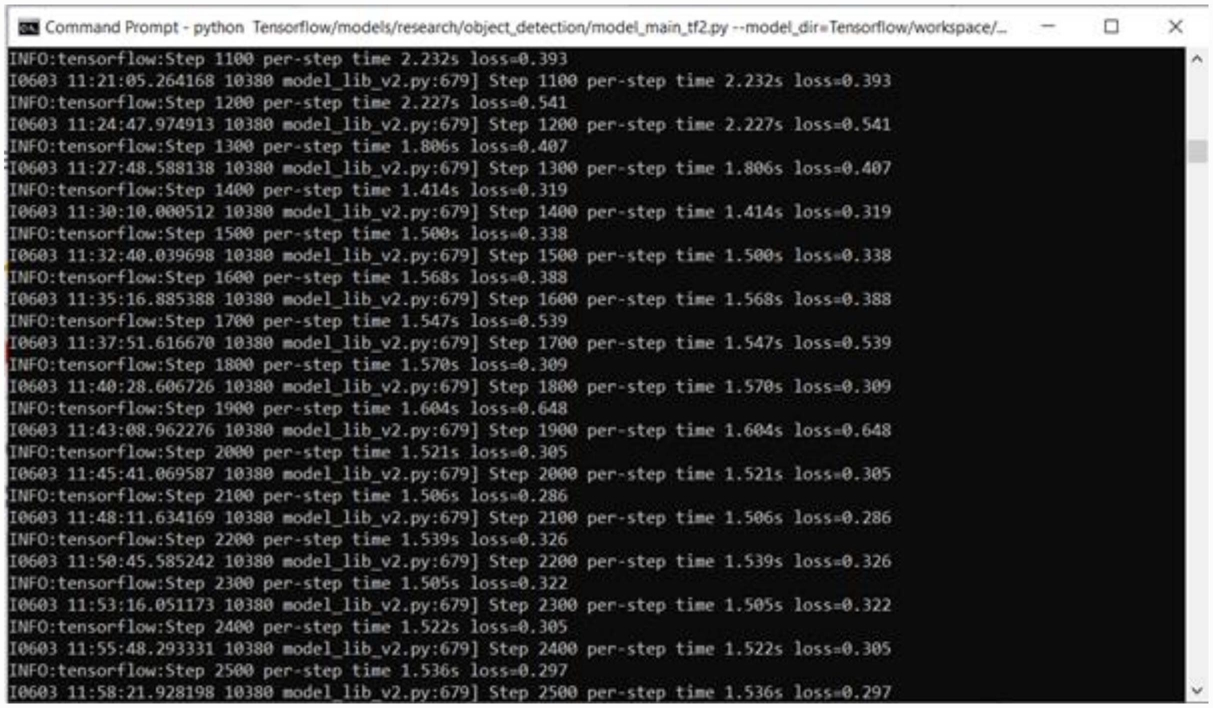
Loss: 1.051



```
Command Prompt - python  Tensorflow/models/research/object_detection/model_main_tf2.py --model_dir=Tensorflow/workspace/...
() to make the check explicit. See https://www.tensorflow.org/guide/checkpoint#loading_mechanics for details.
WARNING:tensorflow:From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\util\deprecation.py:605: calling ma
p_fn_v2 (from tensorflow.python.ops.map_fn) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Use fn_output_signature instead
W0603 10:45:08.637071 13776 deprecation.py:537] From C:\ProgramData\Anaconda3\lib\site-packages\tensorflow\python\util\d
eprecation.py:605: calling map_fn_v2 (from tensorflow.python.ops.map_fn) with dtype is deprecated and will be removed in
 a future version.
Instructions for updating:
Use fn_output_signature instead
INFO:tensorflow:Step 100 per-step time 2.021s loss=1.051
I0603 10:48:25.653840 10380 model_lib_v2.py:679] Step 100 per-step time 2.021s loss=1.051
INFO:tensorflow:Step 200 per-step time 1.814s loss=0.900
I0603 10:51:27.055244 10380 model_lib_v2.py:679] Step 200 per-step time 1.814s loss=0.900
```

STEP 2500 : (5000/2)

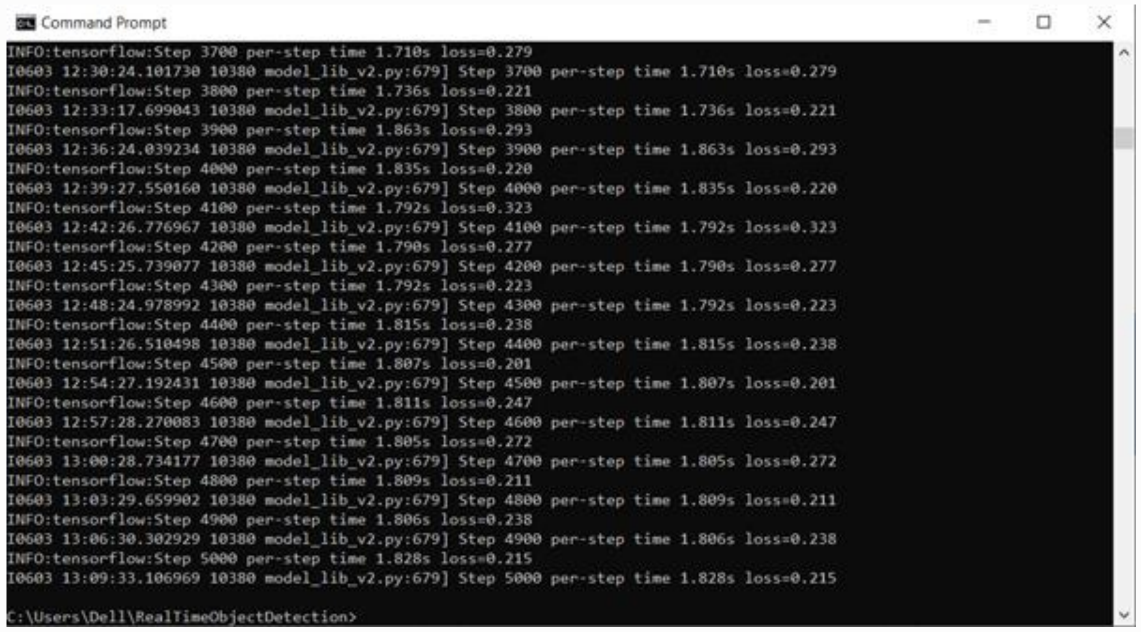
Loss: 0.297 (improved by 71.7% from Step 100)


Command Prompt - python Tensorflow/models/research/object_detection/model_main_tf2.py --model_dir=Tensorflow/workspace/...

```
INFO:tensorflow:Step 1100 per-step time 2.232s loss=0.393
I0603 11:21:05.264168 10380 model_lib_v2.py:679] Step 1100 per-step time 2.232s loss=0.393
INFO:tensorflow:Step 1200 per-step time 2.227s loss=0.541
I0603 11:24:47.974913 10380 model_lib_v2.py:679] Step 1200 per-step time 2.227s loss=0.541
INFO:tensorflow:Step 1300 per-step time 1.806s loss=0.407
I0603 11:27:48.588138 10380 model_lib_v2.py:679] Step 1300 per-step time 1.806s loss=0.407
INFO:tensorflow:Step 1400 per-step time 1.414s loss=0.319
I0603 11:30:10.000512 10380 model_lib_v2.py:679] Step 1400 per-step time 1.414s loss=0.319
INFO:tensorflow:Step 1500 per-step time 1.500s loss=0.338
I0603 11:32:40.039698 10380 model_lib_v2.py:679] Step 1500 per-step time 1.500s loss=0.338
INFO:tensorflow:Step 1600 per-step time 1.568s loss=0.388
I0603 11:35:16.885388 10380 model_lib_v2.py:679] Step 1600 per-step time 1.568s loss=0.388
INFO:tensorflow:Step 1700 per-step time 1.547s loss=0.539
I0603 11:37:51.616670 10380 model_lib_v2.py:679] Step 1700 per-step time 1.547s loss=0.539
INFO:tensorflow:Step 1800 per-step time 1.570s loss=0.309
I0603 11:40:28.606726 10380 model_lib_v2.py:679] Step 1800 per-step time 1.570s loss=0.309
INFO:tensorflow:Step 1900 per-step time 1.604s loss=0.648
I0603 11:43:08.962276 10380 model_lib_v2.py:679] Step 1900 per-step time 1.604s loss=0.648
INFO:tensorflow:Step 2000 per-step time 1.521s loss=0.305
I0603 11:45:41.069587 10380 model_lib_v2.py:679] Step 2000 per-step time 1.521s loss=0.305
INFO:tensorflow:Step 2100 per-step time 1.506s loss=0.286
I0603 11:48:11.634169 10380 model_lib_v2.py:679] Step 2100 per-step time 1.506s loss=0.286
INFO:tensorflow:Step 2200 per-step time 1.539s loss=0.326
I0603 11:50:45.585242 10380 model_lib_v2.py:679] Step 2200 per-step time 1.539s loss=0.326
INFO:tensorflow:Step 2300 per-step time 1.505s loss=0.322
I0603 11:53:16.051173 10380 model_lib_v2.py:679] Step 2300 per-step time 1.505s loss=0.322
INFO:tensorflow:Step 2400 per-step time 1.522s loss=0.305
I0603 11:55:48.293331 10380 model_lib_v2.py:679] Step 2400 per-step time 1.522s loss=0.305
INFO:tensorflow:Step 2500 per-step time 1.536s loss=0.297
I0603 11:58:21.928198 10380 model_lib_v2.py:679] Step 2500 per-step time 1.536s loss=0.297
```

STEP 5000

Loss : 0.215(improved by 79.5% from Step 100)



```
INFO:tensorflow:Step 3700 per-step time 1.710s loss=0.279
I0603 12:30:24.101730 10380 model_lib_v2.py:679] Step 3700 per-step time 1.710s loss=0.279
INFO:tensorflow:Step 3800 per-step time 1.736s loss=0.221
I0603 12:33:17.699043 10380 model_lib_v2.py:679] Step 3800 per-step time 1.736s loss=0.221
INFO:tensorflow:Step 3900 per-step time 1.863s loss=0.293
I0603 12:36:24.039234 10380 model_lib_v2.py:679] Step 3900 per-step time 1.863s loss=0.293
INFO:tensorflow:Step 4000 per-step time 1.835s loss=0.220
I0603 12:39:27.550160 10380 model_lib_v2.py:679] Step 4000 per-step time 1.835s loss=0.220
INFO:tensorflow:Step 4100 per-step time 1.792s loss=0.323
I0603 12:42:26.776967 10380 model_lib_v2.py:679] Step 4100 per-step time 1.792s loss=0.323
INFO:tensorflow:Step 4200 per-step time 1.790s loss=0.277
I0603 12:45:25.739077 10380 model_lib_v2.py:679] Step 4200 per-step time 1.790s loss=0.277
INFO:tensorflow:Step 4300 per-step time 1.792s loss=0.223
I0603 12:48:24.978992 10380 model_lib_v2.py:679] Step 4300 per-step time 1.792s loss=0.223
INFO:tensorflow:Step 4400 per-step time 1.815s loss=0.238
I0603 12:51:26.510498 10380 model_lib_v2.py:679] Step 4400 per-step time 1.815s loss=0.238
INFO:tensorflow:Step 4500 per-step time 1.807s loss=0.201
I0603 12:54:27.192431 10380 model_lib_v2.py:679] Step 4500 per-step time 1.807s loss=0.201
INFO:tensorflow:Step 4600 per-step time 1.811s loss=0.247
I0603 12:57:28.270083 10380 model_lib_v2.py:679] Step 4600 per-step time 1.811s loss=0.247
INFO:tensorflow:Step 4700 per-step time 1.805s loss=0.272
I0603 13:00:28.734177 10380 model_lib_v2.py:679] Step 4700 per-step time 1.805s loss=0.272
INFO:tensorflow:Step 4800 per-step time 1.809s loss=0.211
I0603 13:03:29.659902 10380 model_lib_v2.py:679] Step 4800 per-step time 1.809s loss=0.211
INFO:tensorflow:Step 4900 per-step time 1.806s loss=0.238
I0603 13:06:30.302929 10380 model_lib_v2.py:679] Step 4900 per-step time 1.806s loss=0.238
INFO:tensorflow:Step 5000 per-step time 1.828s loss=0.215
I0603 13:09:33.106969 10380 model_lib_v2.py:679] Step 5000 per-step time 1.828s loss=0.215

C:\Users\Dell\RealTimeObjectDetection>
```

TensorFlow has an in-built function to take the detected image’s frame as an array and postprocess it into, shape, colour and prediction score, which we display it on the screen.

3.2 Results

As seen in the pictures below, the real time sign detector is successfully able to detect the gestures and convert it to the corresponding text.

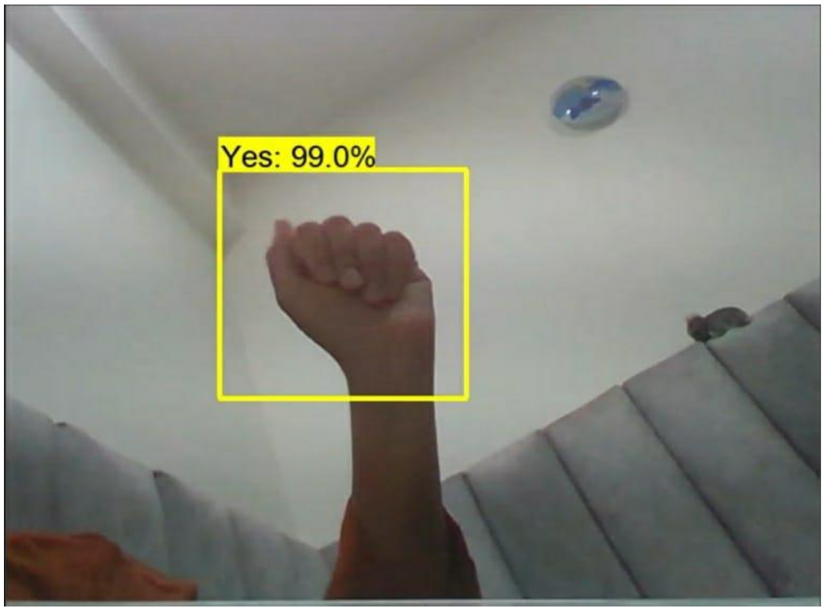


FIG.1

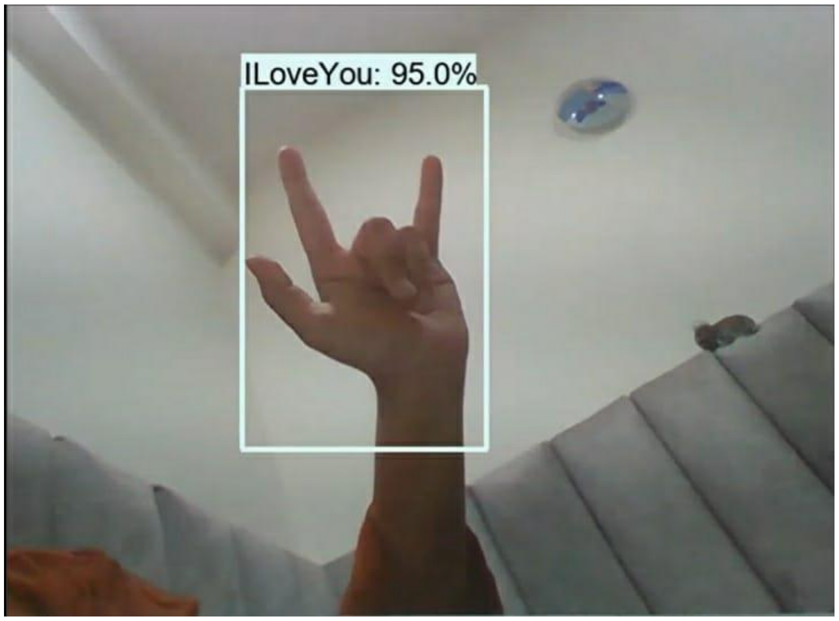


FIG.2

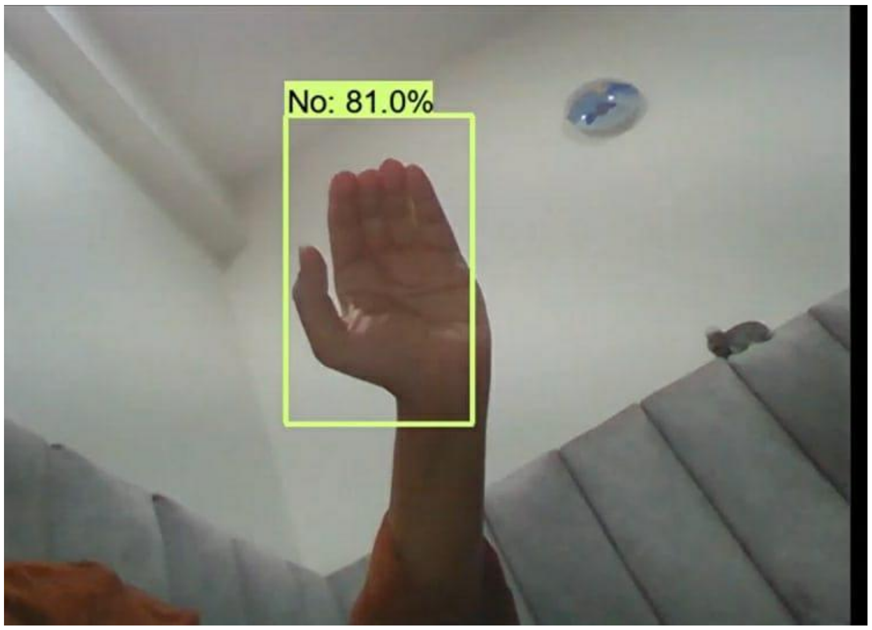


FIG.3

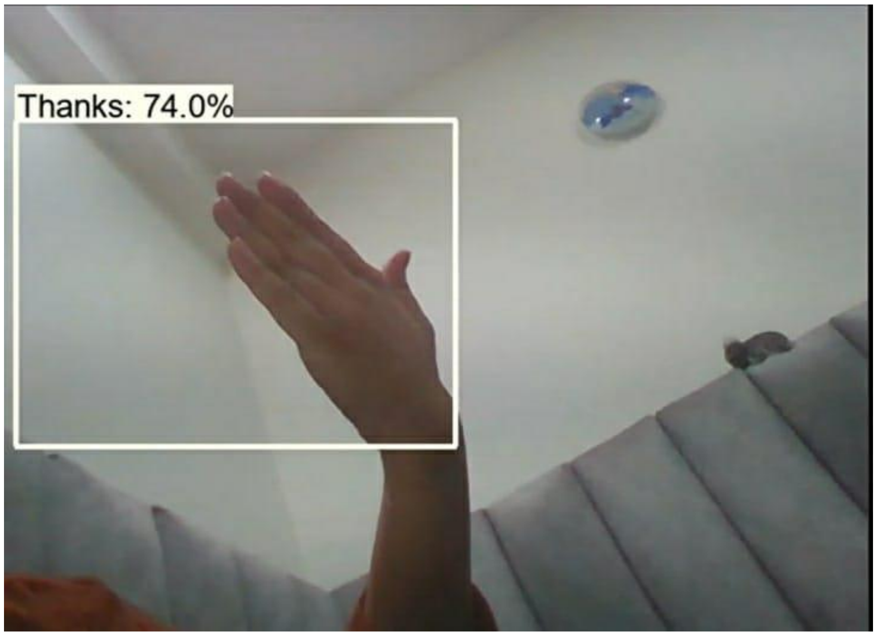


FIG.4

steps_per_sec

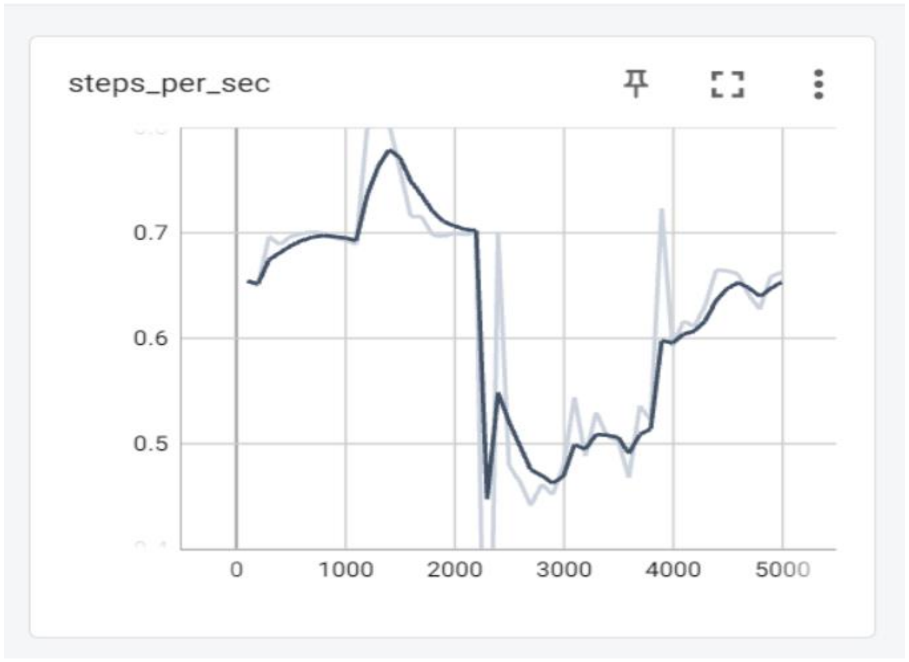
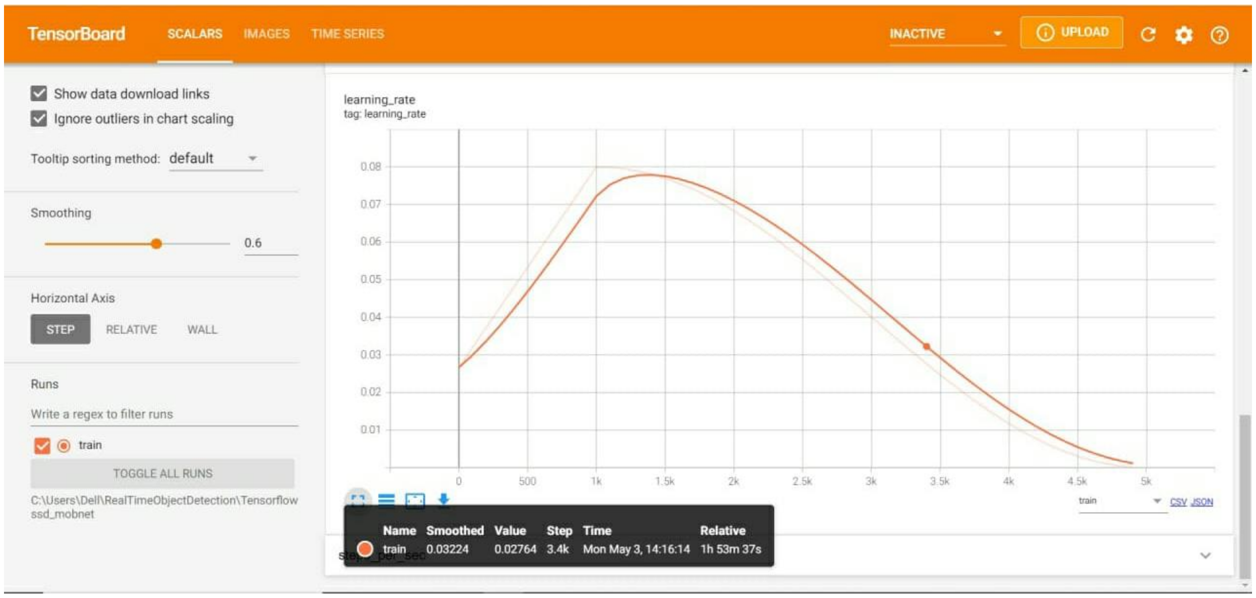


FIG.5

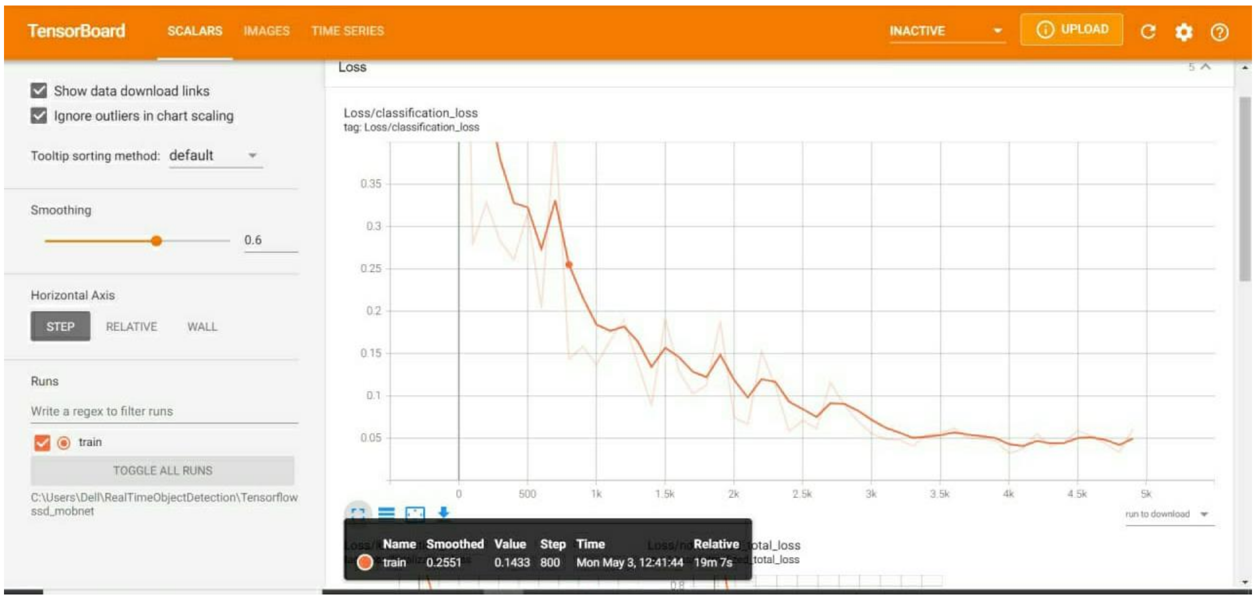


☑ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis: STEP
Runs
Write a regex to filter runs
☑ train
C:\Users\Dell\RealTimeObjectDetection\Tensorflow
ssd_mobnet

learning_rate
tag: learning_rate

Name	Smoothed	Value	Step	Time	Relative
train	0.03224	0.02764	3.4k	Mon May 3, 14:16:14	1h 53m 37s

FIG.6



☑ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis: STEP
Runs
Write a regex to filter runs
☑ train
C:\Users\Dell\RealTimeObjectDetection\Tensorflow
ssd_mobnet

Loss/classification_loss
tag: Loss/classification_loss

Name	Smoothed	Value	Step	Time	Relative
train	0.2551	0.1433	800	Mon May 3, 12:41:44	19m 7s

FIG.7

4. Other Related Work

The real-time sign language recognition system is developed for recognizing the gestures of Indian Sign Language (ISL). Generally, sign languages consist of hand gestures and facial expressions. For recognizing the signs, the Regions of Interest (ROI) are identified and tracked using the skin segmentation feature of OpenCV. The training and prediction of hand gestures are performed by applying fuzzy c-means clustering machine learning algorithm. The gesture recognition has many applications such as gesture-controlled robots and automated homes, game control, Human-Computer Interaction (HCI) and sign language interpretation. The proposed system is used to recognize the real-time signs. Hence it is very much useful for hearing and speech impaired people to communicate with normal people.

5. Future Work

The current version of the software requires a large ram to compute the data and make an accurate prediction. The time required to train the entire data set is very large and can reduce the life of RAM due to long runtime.

>The above shortcoming can be overcome by altering the training algorithm to a more efficient one.

The requirement for a higher RAM calibration can be avoided by optimizing the program.

Like spoken language, sign languages developed naturally through different groups of people interacting with each other, so there are many varieties. English for example, has three varieties: American Sign Language (ASL), British Sign Language (BSL) and Australian Sign Language (Auslan).

The current software does not recognize signs of different languages.

>The above shortcoming can be overcome by expanding the dataset.

The current version of the software converts signs to the English language. People around the globe might not be well versed in the English language, it would cater to a larger group of people if the software was able to converse to multiple languages.

6.Conclusion

In the pandemic, we can see that online communication has become essential for day-to-day life. It is a challenge for mute people to communicate owing to this. To overcome this issue, our project aims to create a platform where the signs they make are appropriately mapped and the corresponding English terms are displayed. It was also of high priority that the software converts the sign to English in real time to hold up a conversation without interruptions.

Bibliography:

- https://data-flair.training/blogs/sign-language-recognition-python-ml-opencv/
- https://towardsdatascience.com/sign-language-recognition-using-deep-learning-6549268c60bd
- https://heartbeat.fritz.ai/exploring-sign-language-recognition-techniques-with-machine-learning-d564262d87d3
- H. Muthu Mariappan and V. Gomathi, "Real-Time Recognition of Indian Sign Language," *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, 2019, pp. 1-6, doi: 10.1109/ICCIDS.2019.8862125.
- https://www.pyimagesearch.com/opencv-tutorials-resources-guides/
- https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e
- https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html