



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Technical Answers for Real World Problems (TARP)

ECE3999

Winter Semester 2020-21

Project Report

Sign Language Recognition Using Python

Teammates:

1. G V Ganesh Maurya – 18BEC0128
2. Gyan Vallabh K – 18BEC0899
3. Sheelavant N Sai Krishna -18BEC0905

Under the guidance of

Dr. VINOTH BABU K

School of Electronics Engineering

VIT,Vellore

May, 2021

INTRODUCTION:

Sign language is composed of visual gestures and sign which is most widely used by deaf and dumb people for communicating purpose. Now-a-days without technology it is becoming very difficult for the deaf-and-dumb to communicate properly. So good communication leads to better understanding. But for normal people it will be bit tough to learn, understand sign-language. There are 143 various sign languages globally and each country have their own sign-language. Each sign has a specific meaning. But we are using Indian Sign Language to make this model.

In this project we are implementing a **“Sign-language recognition using python with the help of Convolutional Neural Networks (CNN) Algorithm”** which detects alphabets. Here we requires the rear camera on a mobile phone. When the person shows the sign with his hand in front of camera, the camera will automatically detect the alphabet and returns the meaning of the sign in text format on the screen. With this even the normal people can communicate very easily with the deaf/dumb people. With this advancement in technology educating these deaf-dumb people will become very easy and the reach of this technology will be vast. Also the quality life of these people will improve.

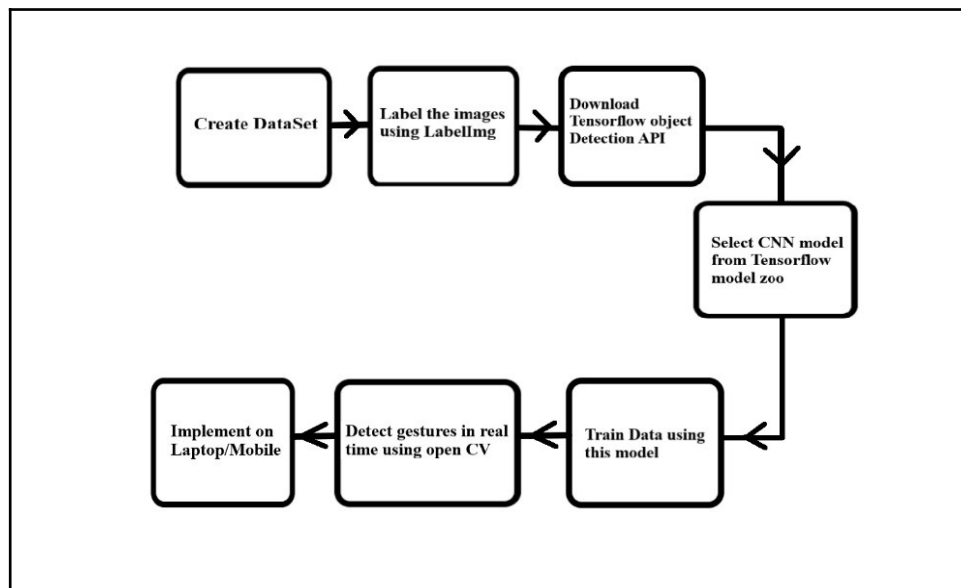
LITERATURE SURVEY:

- In the year 2014 the process was mainly focused on deaf and dumb people in and around society. Their research was summed up to basic gesture control keeping the tech in the mind at that time.
- Here, a cyberglove which mainly detects the motion of the hands and having a fixed point about rotations or movement making it more rigid than moving out hand freely to have a brighter approach to the problem
- Here the glove was to be worn the whole time where gesture tracking happens and solutions were visible to others from the results obtained.
- Problem: He/She can't wear the glove every time and the results obtained were obtained on a different portal which was slightly tough to understand and recognise also.
- As time passed there was drastic change where deep learning and machine algorithms to over there by eliminating the use of cyberglove. In the year 2016 in real time American sign language recognition proposed that these methods can be used to implement on real time basis and capturing would also be done on live with maximum accuracy. Here the

images are trained with certain data set and then its used to find out the desired image which would be captured on a practical scale

- It also had a future scope of finding out things when shown in a sequence of series or so called real time non static detection.
- In the following year there was much greater increment in the previous work which was confined to very few sign languages detections such as phrases and letters in order to tell out few emotions or for few basis usages words and most of the signs was actually based on Americans generally taken into account, but now there was a border perspective like Indian languages were also taken into account.
- But now our project brings up in identifying the Indian sign languages or phrases and etc. (comparatively very less thesis are on this part of our detections) and moreover we able to obtain much better results with higher accuracy around 98% on real time detection by using CNN algorithm.

BLOCK DIAGRAM:



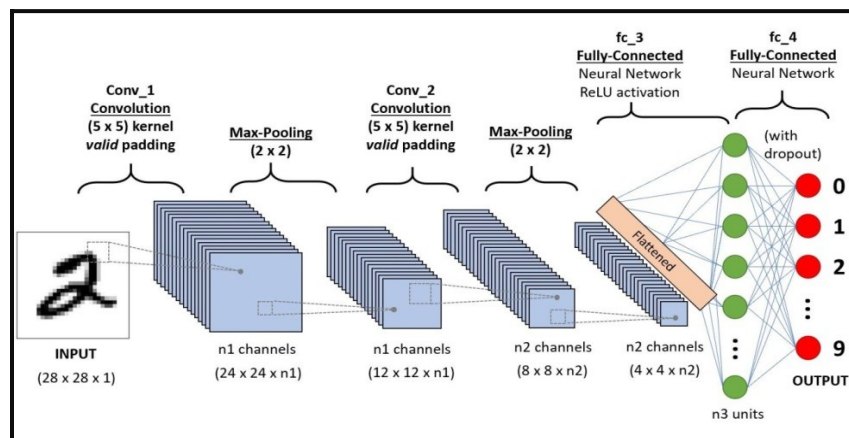
SOFTWARES USED:

1. Tensorflow
2. OpenCV
3. Python
4. Labelling

BACKGROUND:

1. Introduction to CNN:

- In general artificial intelligence is a wide branch of computer science which involves a similar process on how a normal brain think and reacts.
- The main aim of artificial intelligence is to mimic this process, and now lets see how this is done by one of the most prominent and leading methods.
- We can see that pattern differentiation is the main process involved in CNN algorithm.
- Lets us try to understand these hidden layers:
 - The last layer defines or gives us a chance to predict the desired output from various process which was passed on through hidden layers with respective inputs.
 - These hidden layers are generally called as convolutional layers.
 - Like mentioned about how these work on pattern detection we have these special terms called as filters.
 - These filters are mainly responsible for these detections and moreover there are also defines by how many number of filters are present in each convolution layer.
 - To understand why these filters are required in convolutional layers lets us take an example of an image.



- Now as we can see there are specific filters for detections of specific images inside a image for example:
 - a. Patterns detection for edges,
 - b. Patterns detection for corners, pattern detection for circle etc...
- Therefore one type of pattern that could be detected by the filters may be termed as edge filter, corner filter, circle filter and so on.
- Now as we move further into deep detection these layers or filters become and more sophisticated there by are which are used to detect higher level integrated objects or images like eyes, ears, flowers birds etc.



2. Working Aspect:

Since we have understood what are layers, filters and patterns lets try to understand by an illustration in technical manner. Filters: these filters are generally termed as matrices which can be 2×2 , 3×3 , 4×4 respectively based on our assumption. Hence as seen more numbers matrix's means more number of filters.

Step 1: Defining Matrix

-1	-1	-1
1	1	1
0	0	0

-1	1	0
-1	1	0
-1	1	0

0	0	0
1	1	1
-1	-1	-1

0	1	-1
0	1	-1
0	1	-1

- Lets consider four filters with 3×3 matrices.
- Our example of detection is number 7 and the image is like as follows:



Step 2: Now lets define are object of detection.

Step 3: Defining colours for our matrices correspondingly.

- These matrices values can be resembled in the form of colours as follows-1 representing black, 1 representing white, and 0 representing grey.



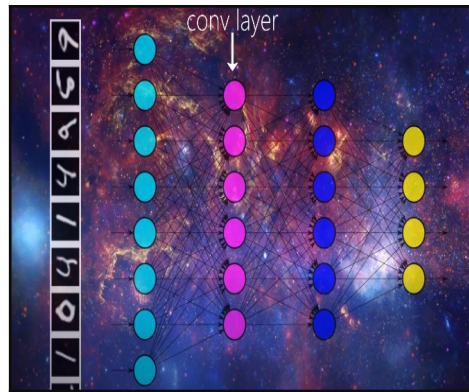
Step 4: Since we know there are various convolution layers hence the base image 7 is convolved with these following filters individually and hence we are able to obtain the following images as possible

Note: All these modified colour filters when convolved with base image mainly detects the edges.



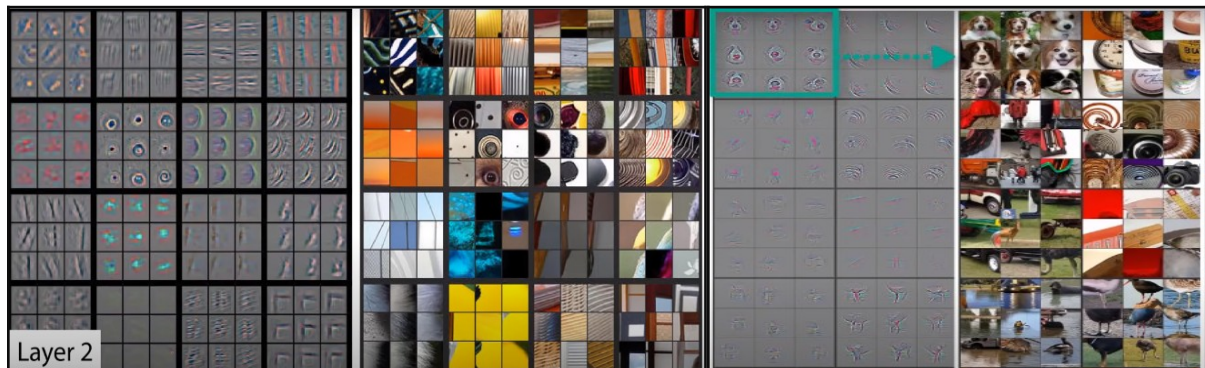
Since we have mentioned about that it mainly concentrates on edges we can see that all the image edges correspondingly are much brighter. Now summing up all the points: We can say that the corresponding base 7 for detection may be confine to one of the corresponding layer with various number of filters as shown below. This process and be multiplied for other numbers also lets see how that is done pictorially since we have understood the process.

-1	1	0
-1	1	0
-1	1	0



Thus these matrix(filters) can be slid over each input base numbers for detection of numbers corresponding to each layers this process of sliding is generally termed as convolution and hence this network of understanding is similar to our brain classification hence the name neural network.

Thus like mentioned more and more deep layers involves in more and more sophisticated filters for detections of more and more sophisticated objects example few are as follows.



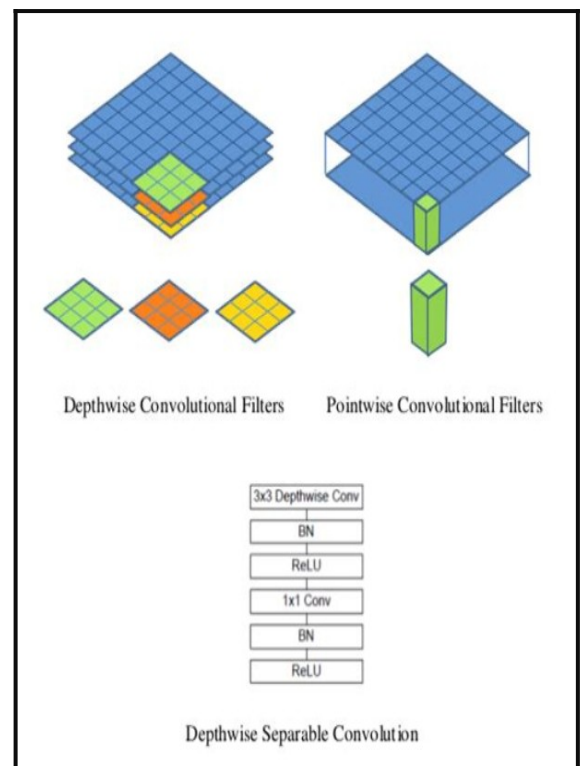
Thus these are few complex image for detection. Therefore concluding CNN this methodology is defined and performed as follows, similarly we can use to detect one of the most prominent problems ie., sign language.

3. About Tensorflow Object Detection API:

- The TensorFlow Object Detection API is an open-source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. There are already pre-trained models in their framework which are referred to as Model Zoo.
- It includes a collection of pre-trained models trained on various datasets such as:
 - COCO (Common Objects in Context) dataset,
 - KITTI dataset, and
 - the Open Images Dataset.
- The various models have different architecture and thus provide different accuracies but there is a trade-off between speed of execution and the accuracy in placing bounding boxes. Tensorflow bundles together Machine Learning and Deep Learning models and algorithms. It uses Python as a convenient front-end and runs it efficiently in optimized C++.

4. Mobile-Net:

- The MobileNet model is based on depthwise separable convolutions which are a form of factorized convolutions. These factorize a standard convolution into a depthwise convolution and a 1×1 convolution called a pointwise convolution.
- For MobileNets, the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs of the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step.
- The depthwise separable convolution splits this into two layers - a separate layer for filtering and a separate



layer for combining. This factorization has the effect of drastically reducing computation and model size.

SAMPLE CODE AND METHODOLOGY:

```
In [5]: import cv2  ## opencv
import os
import time
import uuid

In [2]: IMAGES_PATH = 'Tensorflow\workspace\images\collectedimages' ➡ Create a path to save the images

In [3]: labels = ['hello', 'thanks', 'yes', 'no', 'iloveyou']
number_imgs = 15

In [6]: for label in labels:
    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()

A subdirectory or file Tensorflow\workspace\images\collectedimages\hello already exists.

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

- This is the code for creating the dataset.
- We use functions of OpenCV software in order to automate the process of collecting webcam data.
- Label the images using LabelImg software.
- We then obtain XML files for each image which defines the bounding box of every image.
- We divide the images into training and test data in the ratio 8:2 and store them in separate folders.

0. Setup Paths

```
In [1]: WORKSPACE_PATH = 'Tensorflow/workspace'
        SCRIPTS_PATH = 'Tensorflow/scripts'
        API_MODEL_PATH = 'Tensorflow/models'
        ANNOTATION_PATH = WORKSPACE_PATH + '/annotations'
        IMAGE_PATH = WORKSPACE_PATH + '/images'
        MODEL_PATH = WORKSPACE_PATH + '/models'
        PRETRAINED_MODEL_PATH = WORKSPACE_PATH + '/pre-trained-models'
        CONFIG_PATH = MODEL_PATH + '/my_ssd_mobnet/pipeline.config'
        CHECKPOINT_PATH = MODEL_PATH + '/my_ssd_mobnet/'
```

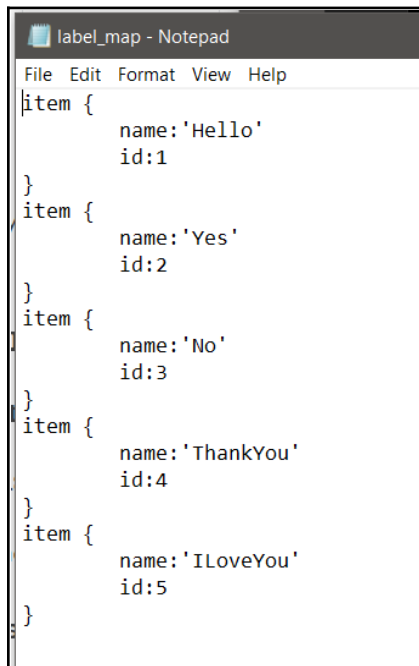
- First step is setup the paths of our workspace.

1. Create Label Map

```
In [2]: labels = [{'name': 'Hello', 'id': 1},
                  {'name': 'Yes', 'id': 2},
                  {'name': 'No', 'id': 3},
                  {'name': 'ThankYou', 'id': 4},
                  {'name': 'ILoveYou', 'id': 5}]

        with open(ANNOTATION_PATH + '\\label_map.pbtxt', 'w') as f:
            for label in labels:
                f.write('item { \n')
                f.write('\\tname:\\'{}\\'\\n'.format(label['name']))
                f.write('\\tid:{}\\n'.format(label['id']))
                f.write('}\\n')
```

- The next step is to create a label map which will be used to as an input to the pipeline.config file which helps in training the dataset.



```
label_map - Notepad
File Edit Format View Help
[
  {
    name: 'Hello'
    id: 1
  },
  {
    name: 'Yes'
    id: 2
  },
  {
    name: 'No'
    id: 3
  },
  {
    name: 'ThankYou'
    id: 4
  },
  {
    name: 'ILoveYou'
    id: 5
  }
]
```

- This is the label map file.
- The names of the images and ID's can be seen here.

2. Create TF records

```
In [3]: !python {SCRIPTS_PATH + '/generate_tfrecord.py'} -x {IMAGE_PATH + '/train'} -l {ANNOTATION_PATH + '/label_map.pbtxt'} -o {ANNOTATION_PATH + '/train.record'}
!python {SCRIPTS_PATH + '/generate_tfrecord.py'} -x {IMAGE_PATH + '/test'} -l {ANNOTATION_PATH + '/label_map.pbtxt'} -o {ANNOTATION_PATH + '/test.record'}

Successfully created the TFRecord file: Tensorflow/workspace/annotations/train.record
Successfully created the TFRecord file: Tensorflow/workspace/annotations/test.record
```

- This command is used to create training and test record files.
- This is done using the tfrecord.py file.
- This file stores the image location and other data, and also the XML file data which was obtained after labeling images using LabelImg.

3. Download TF Models Pretrained Models from Tensorflow Model Zoo

```
In [12]: !cd Tensorflow && git clone https://github.com/tensorflow/models

^C

In [6]: #wget.download('http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz')
#mv ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz {PRETRAINED_MODEL_PATH}
#!cd {PRETRAINED_MODEL_PATH} && tar -zxvf ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8.tar.gz
```

- Download the Tensorflow Object Detection API from Github and also download SSD MobileNet v2 320x320 CNN model from Tensorflow Model Zoo.
- This is the model that will train our data.

Model name	Speed (ms)	COCO mAP	Outputs
CenterNet HourGlass104 512x512	70	41.9	Boxes
CenterNet HourGlass104 Keypoints 512x512	76	40.0/61.4	Boxes/Keypoints
CenterNet HourGlass104 1024x1024	197	44.5	Boxes
CenterNet HourGlass104 Keypoints 1024x1024	211	42.8/64.5	Boxes/Keypoints
CenterNet Resnet50 V1 FPN 512x512	27	31.2	Boxes
CenterNet Resnet50 V1 FPN Keypoints 512x512	30	29.3/50.7	Boxes/Keypoints
CenterNet Resnet101 V1 FPN 512x512	34	34.2	Boxes
CenterNet Resnet50 V2 512x512	27	29.5	Boxes
CenterNet Resnet50 V2 Keypoints 512x512	30	27.6/48.2	Boxes/Keypoints
CenterNet MobileNetV2 FPN 512x512	6	23.4	Boxes
CenterNet MobileNetV2 FPN Keypoints 512x512	6	41.7	Keypoints
EfficientDet D0 512x512	39	33.6	Boxes
EfficientDet D1 640x640	54	38.4	Boxes
EfficientDet D2 768x768	67	41.8	Boxes
EfficientDet D3 896x896	95	45.4	Boxes
EfficientDet D4 1024x1024	133	48.5	Boxes
EfficientDet D5 1280x1280	222	49.7	Boxes
EfficientDet D6 1280x1280	268	50.5	Boxes
EfficientDet D7 1536x1536	325	51.2	Boxes
SSD MobileNet v2 320x320	19	20.2	Boxes
SSD MobileNet V1 FPN 640x640	48	29.1	Boxes

- The CNN model is selected based on speed and accuracy.
- Even though EfficientDet has the best accuracy, we choose SSD MobileNet because of it's very high speed.
- Another reason is that SSD MobileNet can be easily implement on Android and iOS devices which is the ultimate goal of this project.

4. Copy Model Config to Training Folder

```
In [17]: !cd Tensorflow

In [4]: CUSTOM_MODEL_NAME = 'my_ssd_mobnet'

In [5]: !mkdir {'Tensorflow\workspace\models\'+CUSTOM_MODEL_NAME}
!copy [PRETRAINED_MODEL_PATH + '/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/pipeline.config'] [MODEL_PATH+'/' +CUSTOM_MODEL_NAME

A subdirectory or file Tensorflow\workspace\models\my_ssd_mobnet already exists.
The system cannot find the file specified.
```

5. Update Config For Transfer Learning

```
In [6]: import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format

In [9]: CONFIG_PATH = MODEL_PATH+'/' +CUSTOM_MODEL_NAME+'pipeline.config'

In [11]: config = config_util.get_configs_from_pipeline_file(CONFIG_PATH)

In [13]: pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(CONFIG_PATH, "r") as f:
    proto_str = f.read()
    text_format.Merge(proto_str, pipeline_config)

In [14]: pipeline_config.model.ssd.num_classes = 5
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint = PRETRAINED_MODEL_PATH+'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 = ANNOTATION_PATH + '/label_map.pbtxt'
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = [ANNOTATION_PATH + '/train.record']
pipeline_config.eval_input_reader[0].label_map_path = ANNOTATION_PATH + '/label_map.pbtxt'
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = [ANNOTATION_PATH + '/test.record']

In [15]: config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(CONFIG_PATH, "wb") as f:
    f.write(config_text)
```

- Here we have to update the pipeline.config with the number of classes, batch size, fine tune checkpoint, checkpoint type, label map, train and test record files.
- This file will then be used for training the dataset.
- The pipeline.config file is shown below:

```
1  model {
2    ssd {
3      num_classes: 1 # Set this to the number of different label classes
4      image_resizer {
5        fixed_shape_resizer {
6          height: 640
7          width: 640
8        }
9      }
10     feature_extractor {
```

```

147     optimizer {
148         momentum_optimizer {
149             learning_rate {
150                 cosine_decay_learning_rate {
151                     learning_rate_base: 0.03999999910593033
152                     total_steps: 25000
153                     warmup_learning_rate: 0.013333000242710114
154                     warmup_steps: 2000
155                 }
156             }
157             momentum_optimizer_value: 0.8999999761581421
158         }
159         use_moving_average: false
160     }
161     fine_tune_checkpoint: "pre-trained-models/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8/checkpoint/
162     num_steps: 25000
163     startup_delay_steps: 0.0
164     replicas_to_aggregate: 8
165     max_number_of_boxes: 100
166     unpad_groundtruth_tensors: false
167     fine_tune_checkpoint_type: "detection" # Set this to "detection" since we want to be training
168     use_bfloat16: false # Set this to false if you are not training on a TPU
169     fine_tune_checkpoint_version: V2
170 }
171 train_input_reader {
172     label_map_path: "annotations/label_map.pbtxt" # Path to Label map file
173     tf_record_input_reader {
174         input_path: "annotations/train.record" # Path to training TFRecord file
175     }
176 }
177 eval_config {
178     metrics_set: "coco_detection_metrics"
179     use_moving_averages: false
180 }
181 eval_input_reader {
182     label_map_path: "annotations/label_map.pbtxt" # Path to Label map file
183     shuffle: false
184     num_epochs: 1
185     tf_record_input_reader {
186         input_path: "annotations/test.record" # Path to testing TFRecord
187     }
188 }

```

6. Train the model

```

In [16]: print("""python {}research/object_detection/model_main_tf2.py --model_dir={} --pipeline_config_path={} --r
python Tensorflow/models/research/object_detection/model_main_tf2.py --model_dir=Tensorflow/workspace/models/my_ssd_mobnet --pi
pipeline_config_path=Tensorflow/workspace/models/my_ssd_mobnet/pipeline.config --num_train_steps=20000

```

- Model is trained for 20000 steps.

7. Load Train Model From Checkpoint

```
In [17]: import os
        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

In [18]: # Load pipeline config and build a detection model
        configs = config_util.get_configs_from_pipeline_file(CONFIG_PATH)
        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(CHECKPOINT_PATH, 'ckpt-3')).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
```

- The trained model is then loaded from the latest checkpoint and is used for detecting the images.

8. Detect in Real-Time

```
In [19]: import cv2
        import tensorflow as tf
        import numpy as np

In [20]: category_index = label_map_util.create_category_index_from_labelmap(ANNOTATION_PATH+'label_map.pbtxt')

In [21]: # Setup capture
        cap = cv2.VideoCapture(0)
        width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
        height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
```

- OpenCV is used to access the webcam of the laptop and detect the images in real time.


```
In [22]: while True:
    ret, frame = cap.read()
    image_np = np.array(frame)

    input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
    detections = detect_fn(input_tensor)

    num_detections = int(detections.pop('num_detections'))
    detections = {key: value[0, :num_detections].numpy()
                  for key, value in detections.items()}
    detections['num_detections'] = num_detections

    # detection_classes should be ints.
    detections['detection_classes'] = detections['detection_classes'].astype(np.int64)

    label_id_offset = 1
    image_np_with_detections = image_np.copy()

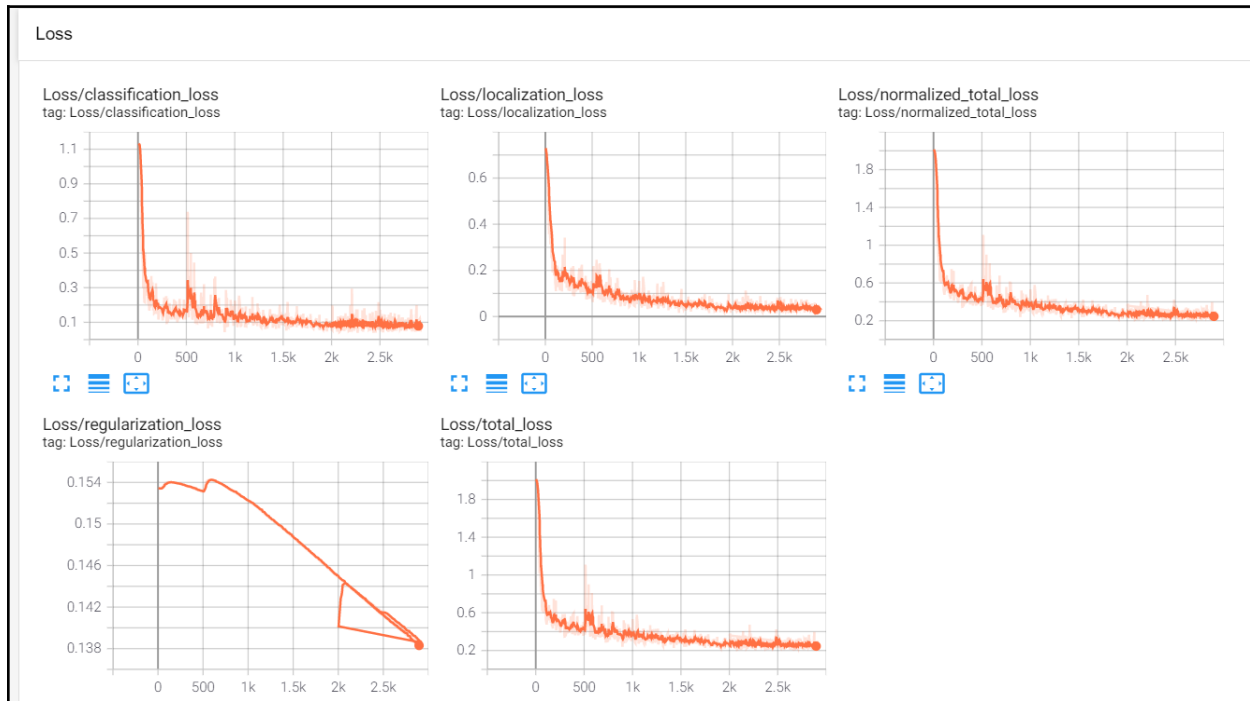
    viz_utils.visualize_boxes_and_labels_on_image_array(
        image_np_with_detections,
        detections['detection_boxes'],
        detections['detection_classes']+label_id_offset,
        detections['detection_scores'],
        category_index,
        use_normalized_coordinates=True,
        max_boxes_to_draw=5,
        min_score_thresh=.5,
        agnostic_mode=False)

    cv2.imshow('object detection', cv2.resize(image_np_with_detections, (800, 600)))

    if cv2.waitKey(1) & 0xFF == ord('q'):
        cap.release()
        break
```

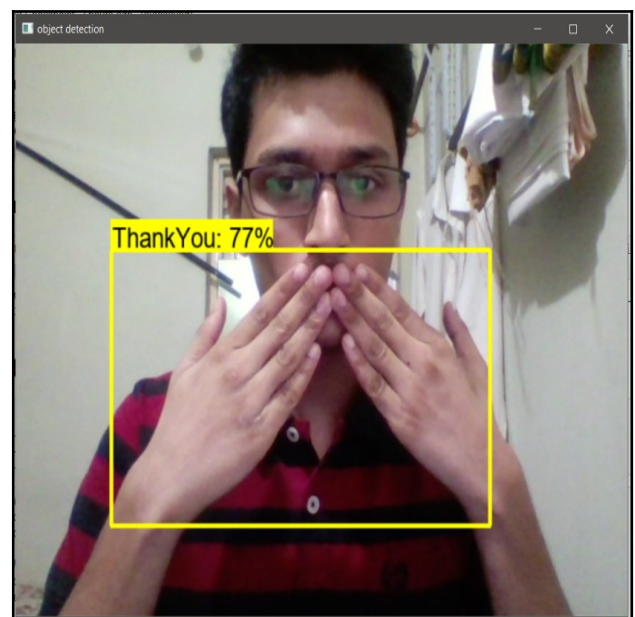
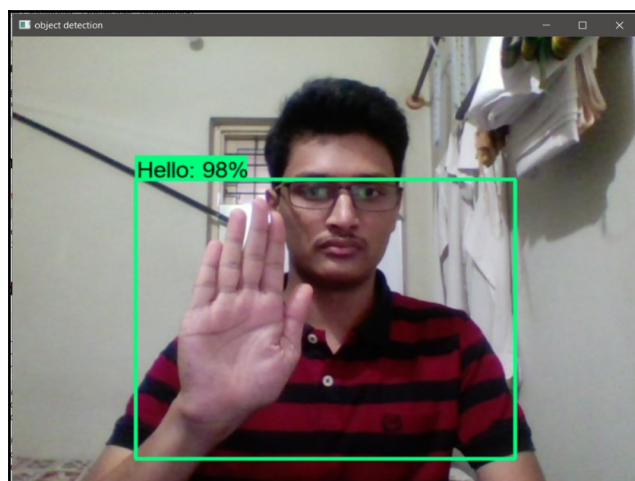
RESULT AND INFERENCE:

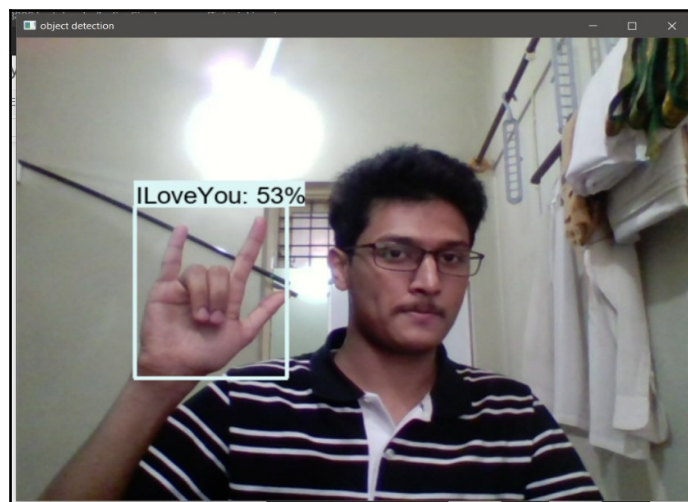
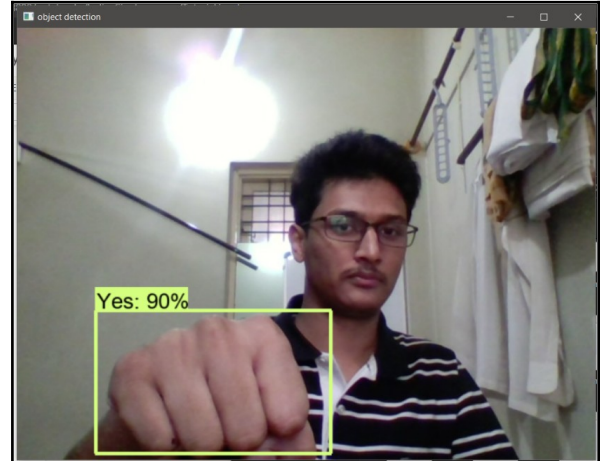
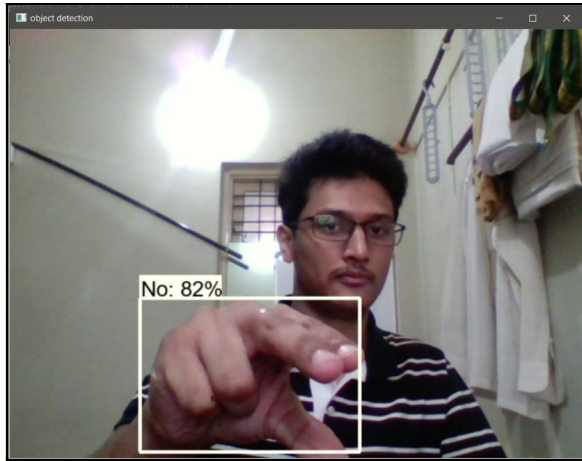
```
Anaconda Powershell Prompt (anaconda3)
INFO:tensorflow:Step 18700 per-step time 2.437s loss=0.154
I0517 15:09:57.222467 10108 model_lib_v2.py:679] Step 18700 per-step time 2.437s loss=0.154
INFO:tensorflow:Step 18800 per-step time 2.517s loss=0.193
I0517 15:13:55.255656 10108 model_lib_v2.py:679] Step 18800 per-step time 2.517s loss=0.193
INFO:tensorflow:Step 18900 per-step time 1.877s loss=0.209
I0517 15:17:11.679709 10108 model_lib_v2.py:679] Step 18900 per-step time 1.877s loss=0.209
INFO:tensorflow:Step 19000 per-step time 2.083s loss=0.201
I0517 15:20:27.462373 10108 model_lib_v2.py:679] Step 19000 per-step time 2.083s loss=0.201
INFO:tensorflow:Step 19100 per-step time 1.802s loss=0.263
I0517 15:23:35.956325 10108 model_lib_v2.py:679] Step 19100 per-step time 1.802s loss=0.263
INFO:tensorflow:Step 19200 per-step time 1.943s loss=0.166
I0517 15:26:47.225286 10108 model_lib_v2.py:679] Step 19200 per-step time 1.943s loss=0.166
INFO:tensorflow:Step 19300 per-step time 1.852s loss=0.173
I0517 15:30:14.496715 10108 model_lib_v2.py:679] Step 19300 per-step time 1.852s loss=0.173
INFO:tensorflow:Step 19400 per-step time 1.702s loss=0.213
I0517 15:33:24.140622 10108 model_lib_v2.py:679] Step 19400 per-step time 1.702s loss=0.213
INFO:tensorflow:Step 19500 per-step time 1.781s loss=0.187
I0517 15:36:19.516501 10108 model_lib_v2.py:679] Step 19500 per-step time 1.781s loss=0.187
INFO:tensorflow:Step 19600 per-step time 1.982s loss=0.202
I0517 15:39:31.704467 10108 model_lib_v2.py:679] Step 19600 per-step time 1.982s loss=0.202
INFO:tensorflow:Step 19700 per-step time 2.471s loss=0.114
I0517 15:43:47.377850 10108 model_lib_v2.py:679] Step 19700 per-step time 2.471s loss=0.114
INFO:tensorflow:Step 19800 per-step time 2.297s loss=0.191
I0517 15:47:49.372654 10108 model_lib_v2.py:679] Step 19800 per-step time 2.297s loss=0.191
INFO:tensorflow:Step 19900 per-step time 2.398s loss=0.163
I0517 15:51:51.465071 10108 model_lib_v2.py:679] Step 19900 per-step time 2.398s loss=0.163
INFO:tensorflow:Step 20000 per-step time 1.788s loss=0.163
I0517 15:55:33.501947 10108 model_lib_v2.py:679] Step 20000 per-step time 1.788s loss=0.163
(base) PS C:\Users\ganes\Final>
```



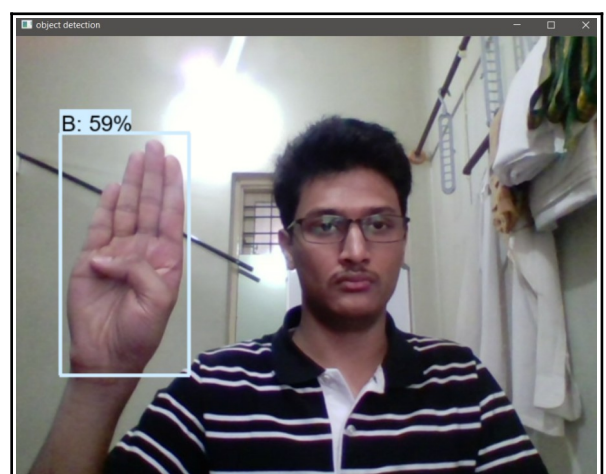
- The loss is obtained in between 0.15 and 0.25, which is the correct loss metric.
- We can also see different loss metrics plotted on the graph above.

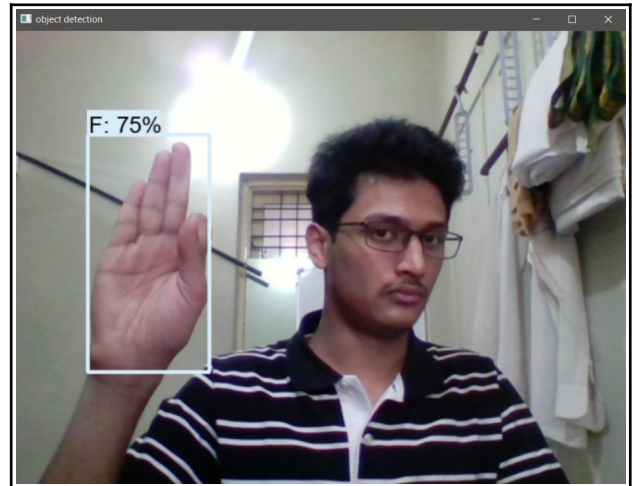
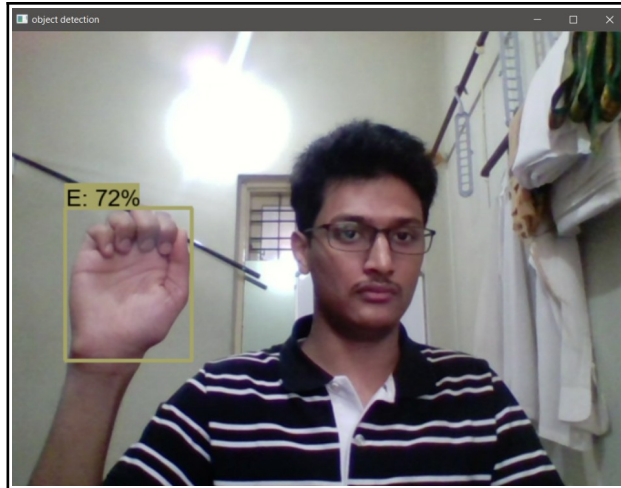
Phrases:





Alphabets:





CONCLUSION:

Hence we were able to implement Sign Language Detection using Tensorflow model. Most of the projects done in this field implemented only American Sign Language, but our project focuses on Indian Sign Language. Unlike other implementations which include detection when no background is present, our detector is able to differentiate between the hand and other body parts. It even detects multiple gestures at the same time.

FUTURE SCOPE:

- In general the scope or research in this area would be nearly small or very minimal but we all have small incremental changes in our topic.
- As we all know its not impossible to use or incorporate these software's into devices and place them at each sector of use.
- We have used CNN to show case how fast its compared with other deep and machine learning algorithms present.
- This part might be bit biased because on paper there might lot algorithms better than CNN but CNN being simple and having natural Selectiveness of desired objects makes it special.
- As we have seen there are many people who are impaired in this aspect most of the people at use these sign language as a medium of communication.
- If we have having a keen observation we can see clearly that in any sectors these days either at billing, shops, banks and etc. are the major places where communication is needed. And since computer are ruling now there these small WEB cams present for no reason.

- Now a days these webcams also have inbuilt software's which have their own structure based on their use.
- Existing area in incorporating our sign language detection methodology of CNN would add value. In spite of having greater use mobile phones almost 30% still at our place is left uneducated.
- But as we all know communications is main medium maybe incorporation of these at computers present at these areas would have a better value.
- And our software's does not require older versions and they don't need any money to set up also it is all done online with ease and cost friendly.



Impact of AI

- As mentioned we are confined to static detections of our sign language we can always look up to detecting our sign language on a video approach or taking up live examples and tuning our algorithm along with AI to adapt to situations when need few images are not present in our data set.
- Thus this eliminates the option of data set related problems.

REFERENCES:

1. Sahoo, Ashok & Mishra, Gouri & Ravulakollu, Kiran. (2014). Sign language recognition: State of the art. ARPN Journal of Engineering and Applied Sciences. 9. 116-134.
2. http://cs231n.stanford.edu/reports/2016/pdfs/214_Report.pdf
3. <https://www.iosrjournals.org/iosr-jce/papers/Vol122-issue3/Series-1/B2203011419.pdf>
4. Tolentino, Lean Karlo & Serfa Juan, Ronnie & Thio-ac, August & Pamahoy, Maria & Forteza, Joni & Garcia, Xavier. (2019). Static Sign Language Recognition Using Deep Learning. International Journal of Machine Learning and Computing. 9. 821-827. 10.18178/ijmlc.2019.9.6.879.
5. https://digitalcommons.kennesaw.edu/cgi/viewcontent.cgi?article=1024&context=cs_etd

6. https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md
7. <https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/training.html#>
8. <https://arxiv.org/abs/1704.04861>
9. <https://keras.io/api/applications/mobilenet/>
10. <https://www.youtube.com/watch?v=pDXdlXlaCco>
11. https://www.youtube.com/watch?v=V0Pk_dPU2lY&t=2756s
12. <https://www.youtube.com/watch?v=IOI0o3Cxx9Q>
13. <https://www.youtube.com/watch?v=Rgpfk6eYxJA&t=905s>
14. https://www.youtube.com/watch?v=dZh_ps8gKgs&t=1443s