भारतीय प्रौद्योगिकी संस्थान जम्मू
INDIAN INSTITUTE OF TECHNOLOGY JAMMU
विद्याधनं सर्वधन प्रधानम्

DIP MID SEMESTER REPORT (Group 2)

ISL(Indian sign language) recognition

1. A narrative of the held discussions with the mentor

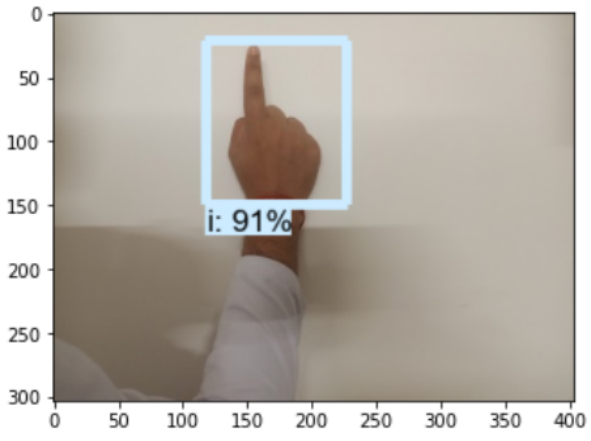
- Data should be prepared as generalized as possible. For example, some pictures can be intentionally taken shaky or blurred since we would not always be able to capture a perfect image from the subject, and our model should still be able to classify the image.
- The dataset should be prepared to avoid problems such as overfitting and underfitting.
- Even after building and training the model, it should be continuously refined as much as possible for better and better prediction accuracy. This can be done by tweaking different parameters, pre-defined functions etc. within the model.

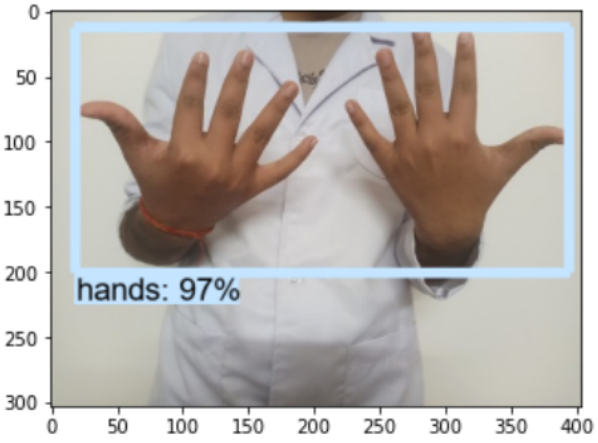
2. Contributions/Efforts in the present semester

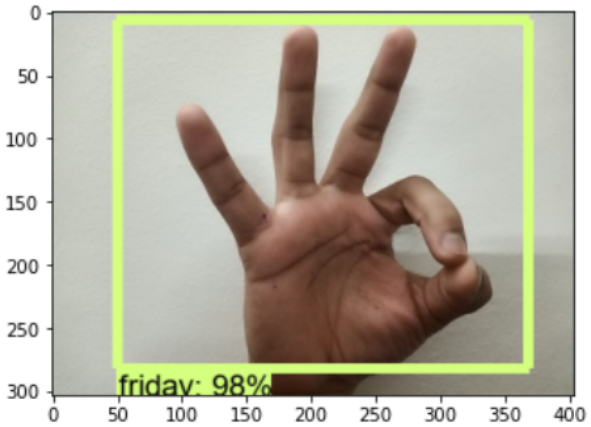
- Building of the deep-learning model, and divided the dataset into train(80%) and test(20%).
- Setup a development environment, and used tf.records in order to save space and faster I/O operations, which is useful when we are working on GPUs.
- Training of the deep learning model on collected symbols (10 up till now as given below).
SSD MobileNet V2 FPNLite 320x320	22	22.2	Boxes
- Performed transfer learning on the above deep learning model which has average speed(that is time taken for one forward propagation) and decent COCO mAp(mean Average precision).

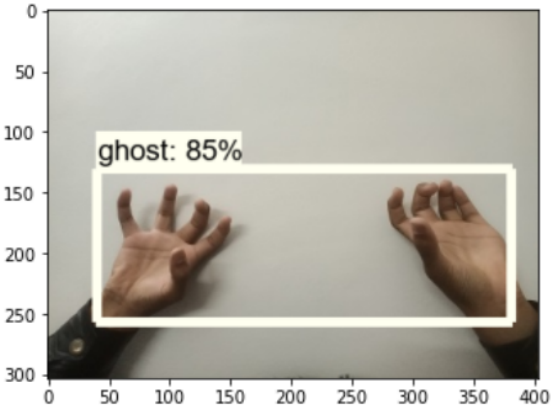
3. Present status of the project/product.

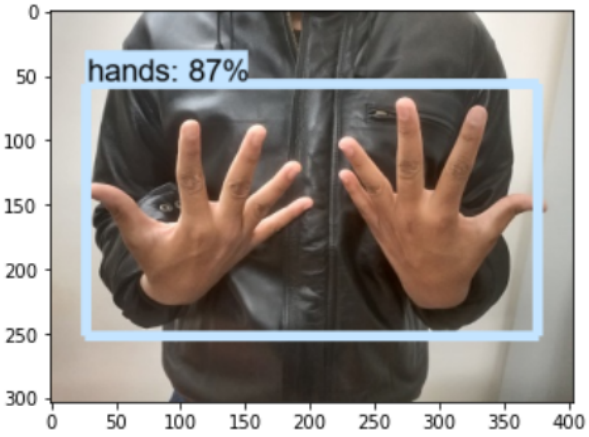
- All the images have been collected and all the labellings of the dataset have been completed.
- The first two-part i.e. till data processing are completed and the last two part of our project i.e. training and testing of our project is partially completed and we are working on that.
- A deep learning model has been built using TensorFlow and Keras by applying Convolutional Neural Network (CNN).
- Deep learning Model has been trained for 10 following class labels: award,bowl,friday,ghost,good,hands,i,like,little,meet.
- The model had been trained only for 2000 epochs with 10 images of each class label, only on CPU.
- Following are some of the detections of our model along with the accuracy of prediction.

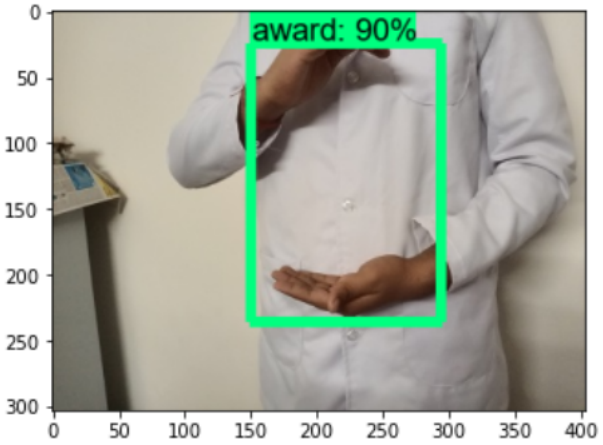












- Evaluation of our current deep learning pipeline.

```
INFO:tensorflow:Eval metrics at step 2000
I0321 10:37:57.609546 20168 model_lib_v2.py:1015] Eval metrics at step 2000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP: 0.643805
I0321 10:37:57.638083 20168 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP: 0.643805
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.50IOU: 0.932692
I0321 10:37:57.639343 20168 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP@.50IOU: 0.932692
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.75IOU: 0.739652
I0321 10:37:57.640341 20168 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP@.75IOU: 0.739652
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (small): -1.000000
I0321 10:37:57.641339 20168 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (small): -1.000000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (medium): -1.000000
I0321 10:37:57.642430 20168 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (medium): -1.000000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (large): 0.643805
I0321 10:37:57.644673 20168 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (large): 0.643805
INFO:tensorflow:        + DetectionBoxes_Recall/AR@1: 0.699167
I0321 10:37:57.645673 20168 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@1: 0.699167
INFO:tensorflow:        + DetectionBoxes_Recall/AR@10: 0.723333
I0321 10:37:57.646670 20168 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@10: 0.723333
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100: 0.723333
I0321 10:37:57.647668 20168 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100: 0.723333
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (small): -1.000000
I0321 10:37:57.648664 20168 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100 (small): -1.000000
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (medium): -1.000000
I0321 10:37:57.649662 20168 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100 (medium): -1.000000
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (large): 0.723333
I0321 10:37:57.653344 20168 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100 (large): 0.723333
INFO:tensorflow:        + Loss/localization_loss: 0.095722
I0321 10:37:57.654348 20168 model_lib_v2.py:1018]       + Loss/localization_loss: 0.095722
INFO:tensorflow:        + Loss/classification_loss: 0.226874
I0321 10:37:57.655310 20168 model_lib_v2.py:1018]       + Loss/classification_loss: 0.226874
INFO:tensorflow:        + Loss/regularization_loss: 0.146546
I0321 10:37:57.656309 20168 model_lib_v2.py:1018]       + Loss/regularization_loss: 0.146546
INFO:tensorflow:        + Loss/total_loss: 0.469142
I0321 10:37:57.657307 20168 model_lib_v2.py:1018]       + Loss/total_loss: 0.469142
```

4. A concrete plan of approach on-campus.

- Our first aim on campus is within one week we will try to allot a PC with good processing in the CS lab, in order to improve our precision by training it on more complex models.
- Training of our model for the remaining class labels and the rest of the images.
- We will be training it on a high-end desktop.
- Our goal is to train our dataset by performing transfer learning on the model which is given below

EfficientDet D7 1536x1536	325	51.2	Boxes

As was discussed before, it has a much higher mAp(mean Average precision) than the model which we are currently using. But as it required much higher computational power we have to train it on a powerful system.

- If the time permits, we will be developing a web-based sign detection app (and possibly a mobile application) that can be used anywhere, anytime.

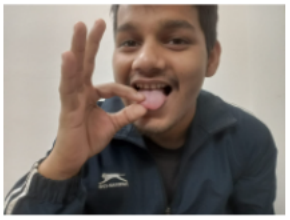
5. Finalized specifics (model and make, etc.) of the items required for the project.

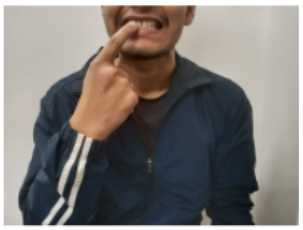
- A high-end pc will be required to train our data preferably along with GPU.

6. Any other details as applicable, if any.

Symbols Selected:
https://docs.google.com/document/d/1hOa80f_Vfl4ukaVDBfETfqPw7ZnnNqGi1mSjZQ3ILx4/edit?usp=sharing


Hands

Tongue



Teeth

Me



You

I


Limit

Busy

Like

Little

Lock

Wood

(Few of the symbols are shown above from the collected dataset)

Our Collected Dataset:
https://drive.google.com/drive/folders/1pLklcbqWJ1rWacFJBKDY6Fp0EOF8_IM0?usp=sharing

Skin segmentation code:

1.https://github.com/abhayishere/Sign-Language-Recognition-Model/blob/main/skin_segmentation.ipynb

```
import numpy as np
import cv2 as cv
from google.colab.patches import cv2_imshow
img = cv.imread("/content/sample3.JPG")
imghsv = cv.cvtColor(img,cv.COLOR_BGR2HSV)
lowerHS = np.array([0, 45, 80], dtype = "uint8")
upperHS = np.array([20, 255, 255], dtype = "uint8")
imgHSV = cv.inRange(imghsv, lowerHS, upperHS)
cv2_imshow(imgHSV)
kernel = cv.getStructuringElement(cv.MORPH_ELLIPSE, (11, 11)) #to get a desired
kernel shape
imgHSV = cv.erode(imgHSV, kernel, iterations = 2)
imgHSV = cv.dilate(imgHSV, kernel, iterations = 4)
cv2_imshow(imgHSV)
```

2.https://github.com/nsachin08/opencv/blob/main/Hand.py

```
import cv2 as cv
import numpy as np
real = cv.VideoCapture(1)
while True :
    ret,frame = real.read()
    frame = cv.flip(frame,1)
    kernel = np.zeros((3,3),dtype='uint8')
    box = frame[200:500, 200:500]

    cv.rectangle(frame,(200,200),(500,500),(255,0,0),0)
    hsv = cv.cvtColor(box,cv.COLOR_BGR2HSV)

    lowerHS = np.array([0,20,70], dtype="uint8")
    upperHS = np.array([20,255,255], dtype="uint8")

    msk = cv.inRange(hsv,lowerHS,upperHS)
    msk = cv.dilate(msk,kernel , iterations=10)

    msk = cv.GaussianBlur(msk,(3,3),200)

    cv.imshow('Mask',msk)
    cv.imshow('frame',frame)

    if cv.waitKey(1) & 0XFF == ord('q'):
        break

real.release()
cv.destroyAllWindows()
```

Link of our deep learning model pipeline.
https://github.com/abhayishere/Sign-Language-Recognition-Model/blob/main/Training%20and%20Detection.ipynb

```
import os
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
PRETRAINED_MODEL_NAME = 'ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8'
PRETRAINED_MODEL_URL =
'http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_mobilenet_v2_fpnlite
_320x320_coco17_tpu-8.tar.gz'
TF_RECORD_SCRIPT_NAME = 'generate_tfrecord.py'
LABEL_MAP_NAME = 'label_map.pbtxt'
paths = {
    'WORKSPACE_PATH': os.path.join('Tensorflow', 'workspace'),
    'SCRIPTS_PATH': os.path.join('Tensorflow','scripts'),
    'APIMODEL_PATH': os.path.join('Tensorflow','models'),
    'ANNOTATION_PATH': os.path.join('Tensorflow', 'workspace','annotations'),
    'IMAGE_PATH': os.path.join('Tensorflow', 'workspace','images'),
    'MODEL_PATH': os.path.join('Tensorflow', 'workspace','models'),
    'PRETRAINED_MODEL_PATH': os.path.join('Tensorflow', 'workspace','pre-trained-models'),
    'CHECKPOINT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME),
    'OUTPUT_PATH': os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME,
'export'),
    'TFJS_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME,
'tfjsexport'),
    'TFLITE_PATH':os.path.join('Tensorflow', 'workspace','models',CUSTOM_MODEL_NAME, 'tflite
export'),
    'PROTOC_PATH':os.path.join('Tensorflow','protoc')
 }
files = {
    'PIPELINE_CONFIG':os.path.join('Tensorflow', 'workspace','models', CUSTOM_MODEL_NAME,
'pipeline.config'),
    'TF_RECORD_SCRIPT': os.path.join(paths['SCRIPTS_PATH'], TF_RECORD_SCRIPT_NAME),
    'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
}
for path in paths.values():
    if not os.path.exists(path):
        if os.name == 'posix':
            !mkdir -p {path}
        if os.name == 'nt':
            !mkdir {path}
import object_detection
labels = [
    {'name':'award', 'id':1},
    {'name':'bowl', 'id':2},
    {'name':'friday', 'id':3},
    {'name':'ghost', 'id':4},
    {'name':'good', 'id':5},
    {'name':'hands', 'id':6},
    {'name':'i', 'id':7},
    {'name':'like', 'id':8},
    {'name':'little', 'id':9},
    {'name':'meet', 'id':10}]
```

```
with open(files['LABELMAP'], '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')
if not os.path.exists(files['TF_RECORD_SCRIPT']):
    !git clone https://github.com/nicknochnack/GenerateTFRecord {paths['SCRIPTS_PATH']}
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'train')} -l
{files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'], 'train.record')}
!python {files['TF_RECORD_SCRIPT']} -x {os.path.join(paths['IMAGE_PATH'], 'test')} -l
{files['LABELMAP']} -o {os.path.join(paths['ANNOTATION_PATH'], 'test.record')}
!copy {os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME,
'pipeline.config')} {os.path.join(paths['CHECKPOINT_PATH'])}
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
config = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "r") as f:
    proto_str = f.read()
    text_format.Merge(proto_str, pipeline_config)
pipeline_config.model.ssd.num_classes = len(labels)
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint =
os.path.join(paths['PRETRAINED_MODEL_PATH'], PRETRAINED_MODEL_NAME, 'checkpoint', 'ckpt-0')
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= files['LABELMAP']
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] =
[os.path.join(paths['ANNOTATION_PATH'], 'train.record')]
pipeline_config.eval_input_reader[0].label_map_path = files['LABELMAP']
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] =
[os.path.join(paths['ANNOTATION_PATH'], 'test.record')]
config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(files['PIPELINE_CONFIG'], "wb") as f:
    f.write(config_text)
TRAINING_SCRIPT = os.path.join(paths['APIMODEL_PATH'], 'research', 'object_detection',
'model_main_tf2.py')
command = "python {} --model_dir={} --pipeline_config_path={}
--num_train_steps=2000".format(TRAINING_SCRIPT,
paths['CHECKPOINT_PATH'],files['PIPELINE_CONFIG'])
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util
configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-6')).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

import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
IMAGE_PATH = os.path.join(paths['IMAGE_PATH'], 'test', 'award (2).jpg')
img = cv2.imread(IMAGE_PATH)
image_np = np.array(img)

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=.8,
            agnostic_mode=False,
            line_thickness=8)

plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
plt.show()
```

Reference from where we have selected the standard symbols according to the Indian Government :
http://www.islrtc.nic.in/
https://drive.google.com/drive/folders/1fAruK7eS-SexL6SOPIrWsEqsnGq4AT9i