SIGN2SPEECH- A Sign Language Recognition Model

A CAPSTONE PROJECT REPORT

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

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

By
B Neha Chowdary(19BCE7097)
Talasila Sri Harsha(19BCE7490)
K Dhanush Kumar(19BCE7639)

Under the Guidance of

Dr. R. Nandha Kumar

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

DECEMBER 2022

CERTIFICATE

This is to certify that the Capstone Project work titled “ **SIGN2SPEECH- A SIGN LANGUAGE RECOGNITION MODEL** ” that is being submitted by **BEZAWADA NEHA CHOWDARY (19BCE7097) , TALASILA SRI HARSHA (19BCE7490)** and **KOGANTI DHANUSH KUMAR(19BCE7639)** 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. R. Nandha Kumar
Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner External Examiner

Approved by

PROGRAM CHAIR **DEAN**

B. Tech. CSE School Of Computer Science and Engineering

ACKNOWLEDGEMENTS

We would like to express our gratitude to **Dr. G. Viswanathan,** Chancellor, **Dr. Sekar Viswanathan,** Vice president, **Dr. S. V. Kota Reddy,** Vice Chancellor and **Dr. Sudha S V**, Dean, School of Computer Science and Engineering, for providing with an environment to work during the tenure of the course.

In jubilant mood we express ingeniously our whole-hearted thanks to **Dr. Saroj Kumar Panigrahy** Program chair and Professor, School of Computer Science and Engineering and our pleasure to express with deep sense of gratitude to **Dr. R. Nandha Kumar**, Associate Professor, School of Computer Science and Engineering, VIT-AP, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in our endeavor.

It is indeed a pleasure to thank all faculty members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showeredon us with zeal, which prompted the acquirement of the requisite knowledge to finalize our course study successfully. We would like to thank our respective parentsfor their support.

At last, but not least, we express our gratitude and appreciation to all those who have helped us directly or indirectly toward the successful completion of this project.

Place: Amaravati

Date:

B. Neha Chowdary
T. Sri Harsha
K. Dhanush Kumar

ABSTRACT

The use of Sign Language may assist in overcoming various obstacles that might arise during interaction among hearing and non-hearing individuals. The recognition of sign language helps non-hearing minorities become more integrated into society. Hand gestures are used in sign language; each has a specific meaning that might shift depending on the context, location, and even the time of day. A system is necessary to remove this obstacle that stands in the way of disabled persons and society.

Image analysis or sensor analysis are the two approaches that may be used in the process of sign language recognition. Both approaches are possible. The image-based method uses one or more cameras to catch pictures of the sign language interpreter's ability to perform the sign and then uses photo conversion to understand the sign; however, these approaches are very precise.   The instrumental gloves built with detectors are used in the sensor-based technique to monitor the hand's representations. These are incredibly precise. However, they come at a high price and are difficult to get.

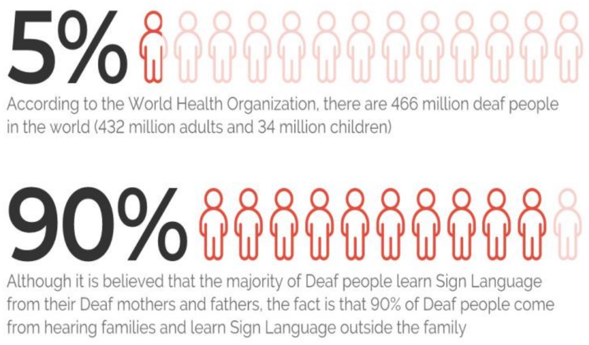
In the context of this project, an image-based model is used to recognize sign language and then transform it into a voice dispatch library. Taking a picture is the first step, followed by image preprocessing, which involves scaling the picture to 640 by 640 pixels, feature extraction, sign classification, and voice translation. The model that is being utilized is called Faster-RCNN-Resnet, and it is a pre-trained model that is obtained from the TensorFlow model zoo.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Sign language is a visual language that uses gestures and facial expressions to transmit message between individuals. Sign language is a form of communication that does not rely on spoken words and is used only by deaf people to communicate with their hearing peers and the outside world. It relies on visual signals from the hands, eyes, face, lips, and body of the person doing the reading. The hand movements or symbols used in sign language are arranged in a manner that is linguistic. Fingerspelling, hand gestures, body language, facial expressions, timing, touch, and everything else that may transmit thoughts or ideas without the need for words are all a part of non-verbal communication, which is a rich mix.



1.Infographic

Sign language is a technique for people unable to speak normally to communicate with one another; it offers a voice to those who do not have one. A survey indicated that around 466 million individuals are deaf, out of whom 34 million are adults, yet 90 percent of them learn sign language from people who are not related to them in any way. It is difficult to learn any language, but sign language is incredibly challenging since it is the only method to interact with individuals who are deaf. As a result, there is a pressing need for a role model capable of understanding disabled people and acting as their voice.

After completing this project, the deaf will be able to communicate more effectively with others. This project uses a quicker RCNN resnet, a pre-trained model that has been tweaked to meet the project's requirements. The model is trained on ten classes comprising different keywords, such as " Hello, Help, Ok, Goodluck, Party time, One, three, Sorry, Super, and Victory."

1.1 OBJECTIVES

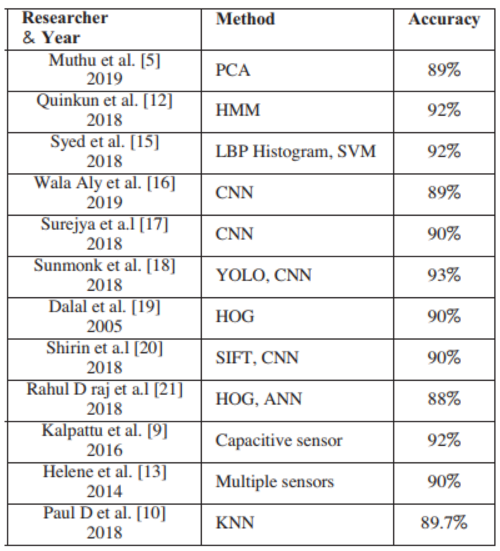
In this project the objectives met were:

- To design an efficient system that can convert Sign language to speech, using Deep Neural Networks and other libraries.
- A system that is efficient, accurate and cheap and easily accessible.

1.2 BACKGROUND AND LITERATURE SURVEY

The research article titled ***"A Translator for American Sign Language to Text and Speech"*** served as the idea for the creation of this project. It presented a method that automatically recognizes the static hand signals of the alphabet in American Sign Language (ASL). They merged the ideas behind the AdaBoost and Haar-like classifiers to accomplish this goal. This image-based method is constructed with the help of an extensive dataset of sign images, and it reached an efficiency of 98.7% with 26 classes. Following the detection of the hand sign and the extraction of its characteristics using a Haar-like classifier, AdaBoost was used to classify the sign. A Windows software development kit is used to transform the voice into text.

In the study titled ***"A Brief Review of the Recent Trends in Sign Language Recognition"*** various models are evaluated based on how accurately they detected sign language. At the end of the study, the authors came to the following conclusion based on their findings: HMM, LBP Histogram, and SVM perform the best when contrasted to other models in identifying sign language.



Researcher & Year	Method	Accuracy
Muthu et al. [5] 2019	PCA	89%
Quinkun et al. [12] 2018	HMM	92%
Syed et al. [15] 2018	LBP Histogram, SVM	92%
Wala Aly et al. [16] 2019	CNN	89%
Surejya et a.l [17] 2018	CNN	90%
Sunmonk et al. [18] 2018	YOLO, CNN	93%
Dalal et al. [19] 2005	HOG	90%
Shirin et a.l [20] 2018	SIFT, CNN	90%
Rahul D raj et a.l [21] 2018	HOG, ANN	88%
Kalpattu et al. [9] 2016	Capacitive sensor	92%
Helene et al. [13] 2014	Multiple sensors	90%
Paul D et al. [10] 2018	KNN	89.7%

2. Algorithm Comparison

In the research described in the paper titled *"**Sign Language to Speech Conversion**"* a flex sensor-based sign language recognition module has been established to notice English letters of the alphabet and a couple of words, and a Text-to-Speech synthesizer based on HMM has been constructed to convert the relating text. Flex sensors were utilized so that the hand motion could recognize. This study uses a text-to-speech synthesis system (HTS) based on a Hidden Markov Model (HMM) to generate synthetic speech in the English language. There are 1,141 unique words used during the model's training process, and the accuracy was 87.5%.

In the article titled *"**Sign Languages to Speech Conversion Prototype using the SVM Classifier**"* the authors describe a glove fitted with sensors and used to identify hand gestures. The model was trained on data from both ISL and ASL using SVM. It achieved an accuracy of 100 percent for ISL and 98.91 for ASL using 75 percent train data and 25 percent test data across 11 courses. This approach is entirely accurate; however, it is costly and challenging. Additionally, it is not readily available. The prototype could identify American Sign Language (ASL) and Indian Sign Language (ISL).

*"**Real-Time Hand Gesture Identification Based on Deep Learning YOLOv3 Model**"* is a publication that presents a lightweight model for gesture recognition based on YOLO (You Only Look Once) v3 and DarkNet-53 convolutional neural networks. This model is intended to recognize hand gestures in real time, and they trained the model without extra preprocessing, image filtering, or improvement. The suggested model was tested using both PASCAL VOC ANS YOLO formats using the labeled dataset, and the results are compared. It produced excellent results by extracting characteristics from the hand and a few hand motions, achieving an accuracy score of 97.68, a precision score of 94.88, a recall score of 98.66, and an F-1 score of 96.70%, respectively. In addition, the accuracy of this model is evaluated in comparison to that of the Single Shot Detector (SSD) and the Visual Geometry Group (VGG16) models, both of which attained an accuracy of between 82 and 85%.

A deepGesture algorithm is proposed in the paper *"deepGesture: Deep learning-based gesture recognition scheme using motion sensors."* This algorithm builds a new hand gesture recognition method based on gyroscope and accelerometer sensors and uses deep convolution and recurrent neural networks. This approach to automating the learning of features from raw sensor data included four layers of deep convolution. It was possible to get excellent and accurate results using the suggested method. As an outcome of the experiment, it was shown that the mean F1-score climbs by more than 6% for nine specified gestures and rises by 9% in the case of a counterclockwise-drawing circle. In addition, the confusion matrix reveals that the prediction accuracy for every class has increased by 6%.

1.3 ORGANIZATION OF THE REPORT

The remaining chapters of this report are described as follows:

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

CHAPTER-2

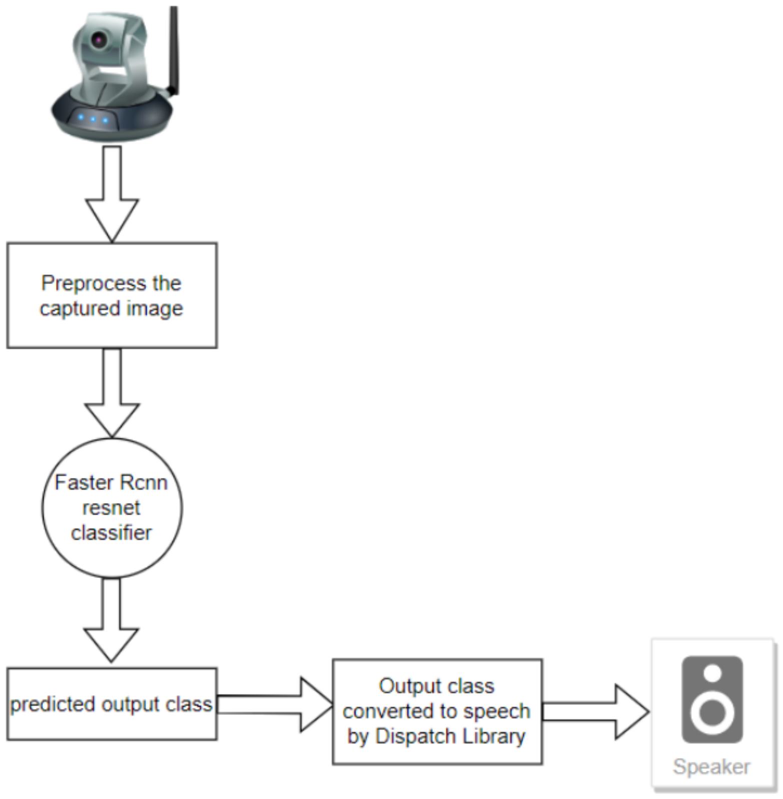
SIGN2SPEECH CONVERSION

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

2.1 PROPOSED SYSTEM

The major component of the system that is being offered is the software portion, which will convert sign language to speech through transfer learning and with the assistance of a pre-trained Faster RCNN Resnet model. This model is currently being used to train on a dataset created by the user. The dataset has 160 photos for the train set and 40 images for the test set. (Food, Goodluck, Hello, Help, Hungry, It’s party time, please come, Sorry, Super, and Victory)

The following block diagram shows the system architecture of this project.


Preprocess the captured image
Faster Rcnn resnet classifier
predicted output class
Output class converted to speech by Dispatch Library
Speaker

3. Proposed System’s Block Diagram

2.2 WORKING METHODOLOGY

First, we will take some input images with 10 different classes. To preprocess the captured image, we train the dataset using the pretrained model. After performing the preprocessing, we will pass the images into the FASTER RCNN Resnet algorithm.

The faster RCNN model is used to identify the hand motions inside the ROI, and the resent model categorizes the sign or the hand gesture. After that, the dispatch library is used to turn the newly created text class into voice. In this project, Faster RCNN RESNET is used for image detection, where faster RCNN is used for the purpose of image localization and Resnet is also known as residual network is used for the purpose of classification of an image. This pretrained model was trained on 90 different object classes, and gave the mean average precision value of 31.8, This model's pretrained weights will be used to train the user-defined dataset, to learn on the bounding boxes of the images and the unique features required to identify the classes or hand gestures in this case.

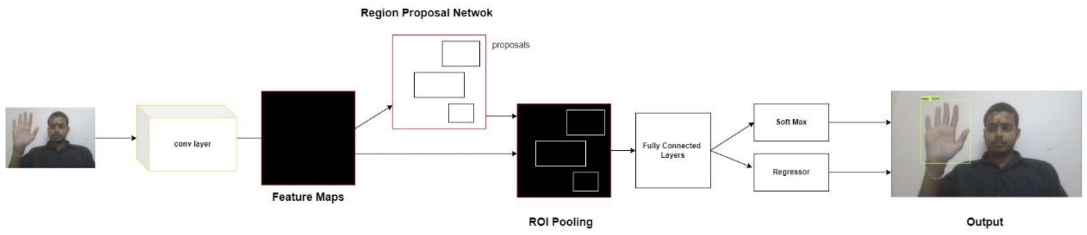
Hence, Algorithm will detect the hand gestures of sign language and predict the respective output class and will display the text and accuracy. After predicting the text, we will convert the text into speech dispatch library. Then we can listen the speech of that class.

To design an efficient system that can convert Sign language to speech, using Deep Neural Networks and other libraries. A system that is efficient, accurate and cheap and easily accessible. Additional features include converting the text to speech using a windows dependent library.

2.3 FASTER RCNN Architecture:

Faster RCNN is an object detection model presented by Ross Girshick, Shaoqing Ren, Kaiming He and Jian Sun in 2015, it is one of the famous object detection models that uses convolution neural networks.

Faster RCNN has 3 parts :

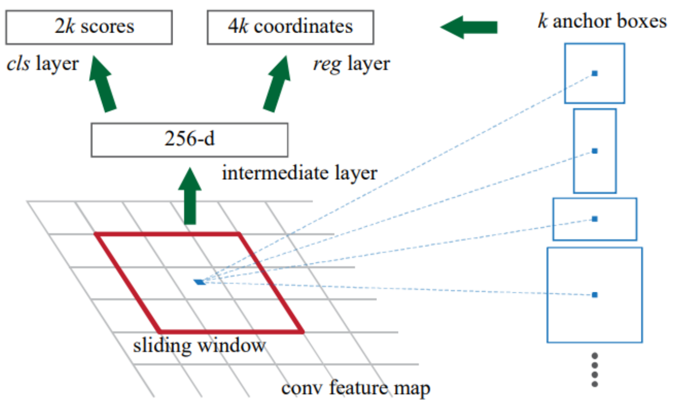


Convolution Layers

In this layer, the filters are trained to extract the appropriate features from the image. This convolution neural network consists of a Convolution layer, a pooling layer, followed by an activation function. All these together are termed as one convolution layer, these layers can be added based on the requirement, and in the last, all these are connected to form a fully connected layer, which is then flattened, and a dropout function is used to reduce overfitting, followed by an output layer with an appropriate activation function like sigmoid function is used.

Region Proposal Network

This layer is followed by the convolution layer, which detects whether there is an object present in the image. If yes, then it identifies its bounding boxes. The region proposals are now generated using a network that could be trained and customized according to the detection task. Because the proposals are generated using a network, this can be trained end-to-end to be customized on the detection task. Thus, it produces better region proposals compared to generic methods like Selective Search and Edge Boxes. The RPN processes the image using the same convolutional layers used in the Fast R-CNN detection network. Thus, the RPN does not take extra time to produce the proposals compared to the algorithms like Selective Search. Due to sharing the same convolutional layers, the RPN and the Fast R-CNN can be merged/unified into a single network. Thus, training is done only once.



Region proposal Network(RPN)

Anchors At each sliding-window location, we simultaneously predict multiple region proposals, where the number of maximum possible proposals for each location is denoted as k. So the reg layer has 4k outputs encoding the coordinates of k boxes, and the cls layer outputs 2k scores that estimate the probability of an object or not an object for each proposal. The k proposals are parameterized relative to k reference boxes, which we call anchors. "Region" is a generic term we only consider rectangular regions, as is common for many methods. "Object ness" measures membership to a set of object classes vs background. 4. For simplicity, we implement the class layer as a two-class SoftMax layer. Alternatively, one may use logistic regression to produce k scores.

The RPN works on the output feature map returned from the last convolutional layer shared with the Fast R-CNN. This is shown in the next figure. Based on a rectangular window of size nxn, a sliding window passes through the feature map. For each window, several candidate region proposals are generated. These proposals are not the final proposals as they will be filtered based on their "objectness score."

Classes and Bounding Box Prediction

We took the output generated from the region proposal as input and passed it into the RoI pooling layer, and this RoI pooling layer has the same function as it performed in Fast R-CNN, to make different sizes region proposals generated from RPN into a fixed-size feature map.

SoftMax and Bounding Box Regression Layer:

The feature map size generated in RoI pooling is then sent to two fully connected layers. These fully connected layers flatten the feature maps and then send the output into two parallel fully connected layers, each with a different task assigned to them. The first layer is a softmax layer output parameter that predicts the objects in the region proposal. The second layer is a bounding box regression layer. This layer regresses the bounding box location of the object in the image. This layer takes its input from the RPN layer, and from the object and its bounding boxes, it classifies the image.

FASTER RCNN RESNET

In this project, Faster RCNN RESNET is used for image detection, where faster RCNN is used for the purpose of image localization and Resnet is also known as residual network is used for the purpose of classification of an image. This pretrained model was trained on 90 different object classes, and gave the mean average precision value of 31.8, This model’s pretrained weights will be used to train the user-defined dataset, to learn on the bounding boxes of the images and the unique features required to identify the classes or hand gestures in this case.

2.4 SYSTEM DETAILS

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

2.4.1 SOFTWARE DETAILS

Object Detection API, Pretrained model from TensorFlow model zoo (Faster RCNN resnet model), Dispatch library, Open cv, TensorFlow

- **Object Detection API:** The TensorFlow object detection API is an open-source library built using TensorFlow, is used to train, build, and deploy multi-object detection models. In this project this API is used to convert the user-defined data, into pertain models compatible format, and for localization and detection of the classes.
- **Pretrained Faster RCNN Resnet model:** A pretrained model for multiple object detection trained on 90 different classes. It gives highly accurate results, along with greater confidence levels. The disadvantages include that it consumes a huge amount of time, storage space, and computation power.
- **Dispatch Library:** A windows specific library, for the purpose of converting text to speech, It works without internet, but is native to windows OS.
- **Open CV:** This Python library is used to perform image processing and related operations, and to access images from the webcam.
- **TensorFlow:** To build and analyze the metrics of the trained model.

2.4.2 SYSTEM REQUIREMENTS

- WINDOWS OS
- Python 3.X
- TensorFlow 2.X
- NumPy Latest version

CHAPTER 3
RESULTS AND DISCUSSIONS

Initially, a dataset with ten classes, each with 20 images, is created, where 16 are used for training and 4 for testing. Using this data, the images are labelled using LabelImg software to draw bounding boxes and label images with their respective class. Once this process is completed, a label map is created for the class labels and is stored in the annotations. Then the training and test data are converted to TensorFlow records for the pre-trained model to train on. Edit the config file to accommodate the custom dataset. Now start the training, all the latest checkpoints are saved to the system with the trained model. To predict using this model, the latest checkpoint can be loaded and can be used to predict a single image or from the webcam.

The TensorFlow model zoo models use the coco metrics, where the results are given in terms of Precision and Recall.

IoU (Intersection over Union): It is used to decide whether a prediction is correct or not with respect to an object. It can be defined as the intersection between the predicted bounding box and actual bounding box divided by their union. A prediction is considered to be True Positive if its IoU value > threshold, and False Positive if IoU value< threshold.

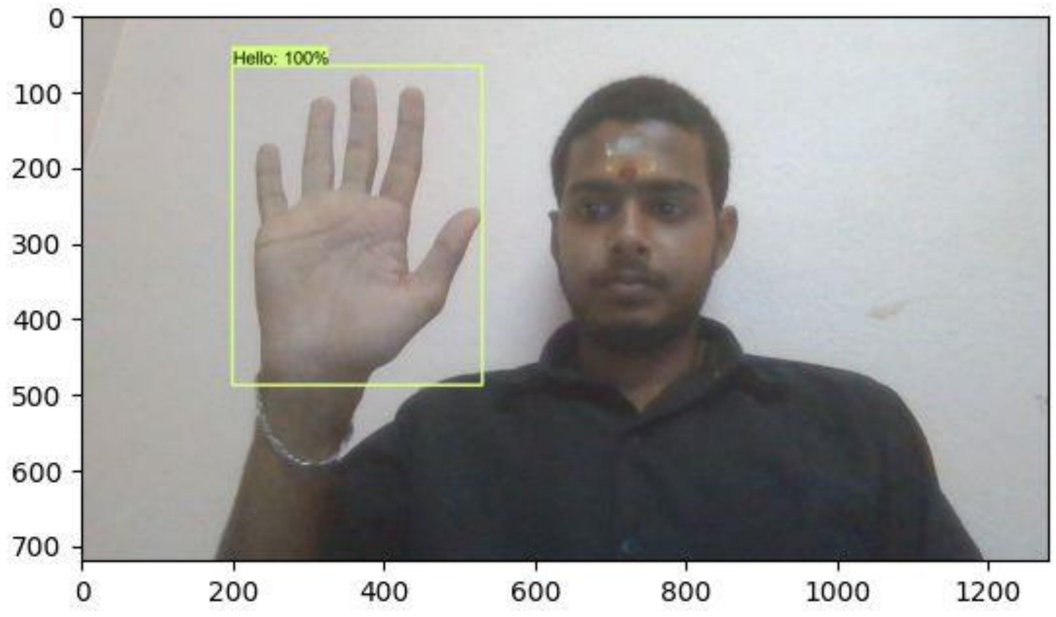
Recall: Recall is defined as the True Positive Rate .In other words, out of all the actual positives, how many of them are True positive predictions.

Precision: Precision is defined as the Positive prediction value .In other words, out of all the positive predictions made, how many of them are True positive predictions.

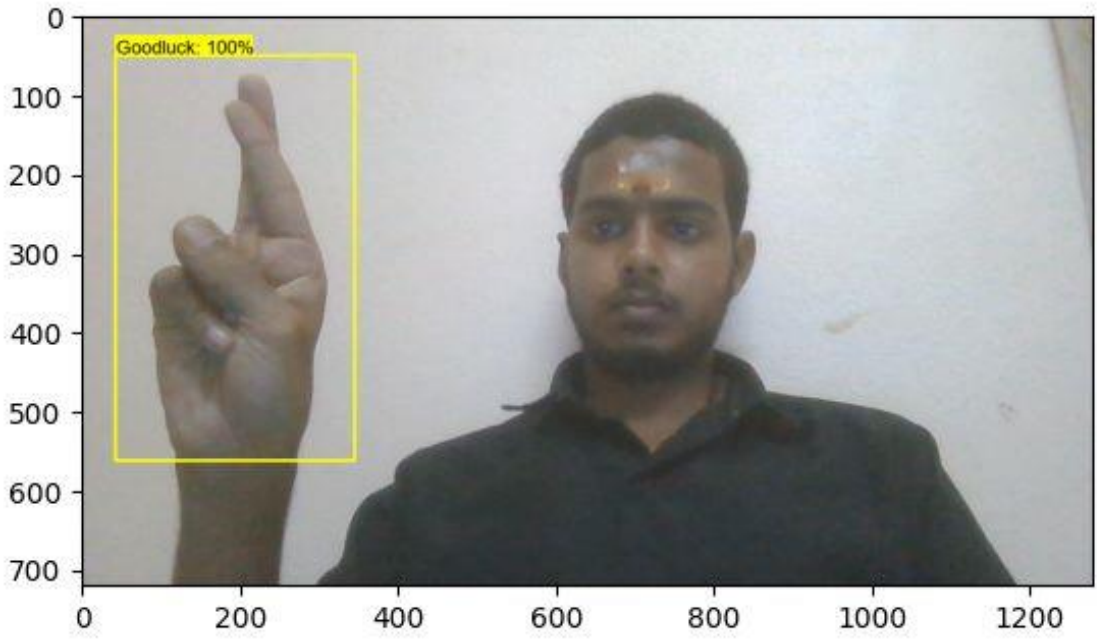
The current model obtained a precision of 0.74 and a recall value of 0.77.

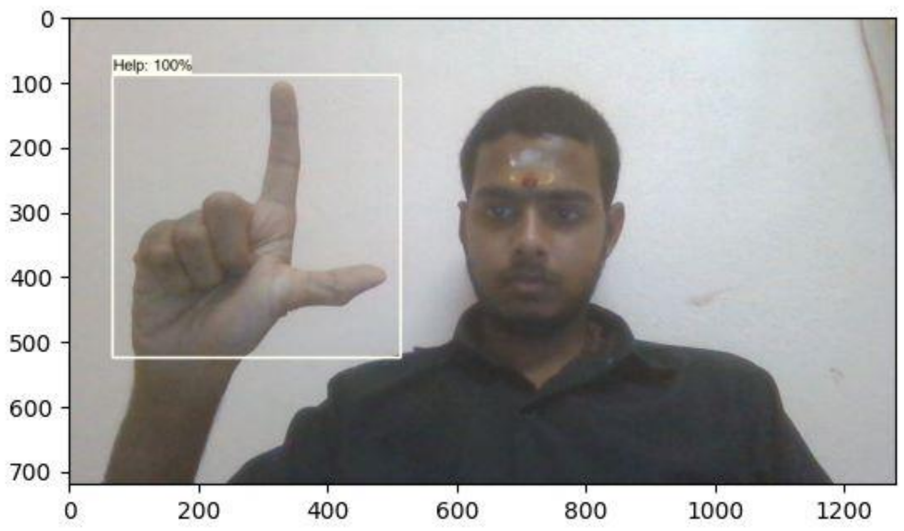
The model was tested against an individual image and using a webcam, both ways the results obtained were good.

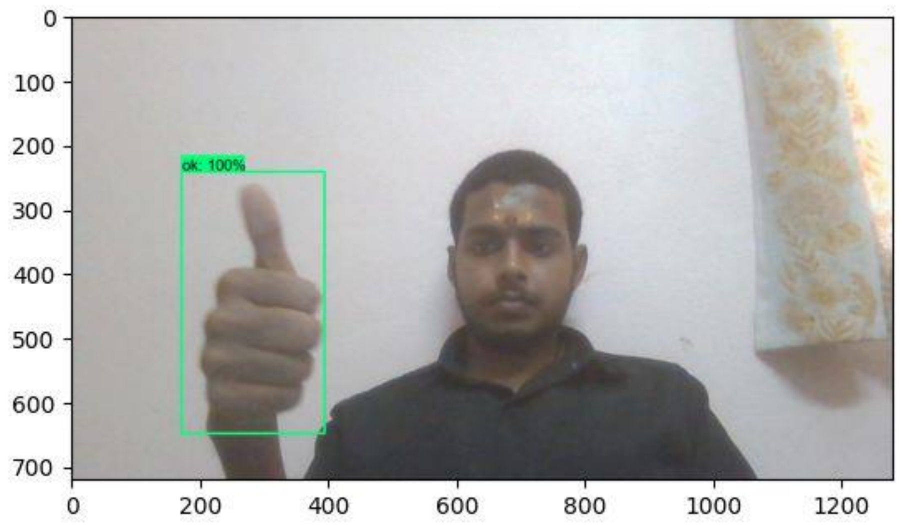
The image prediction obtained for various classes is as follows:

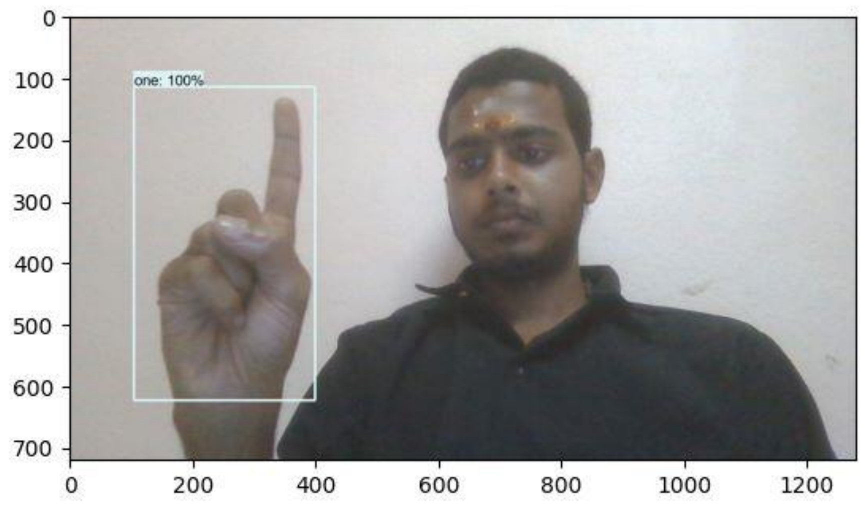


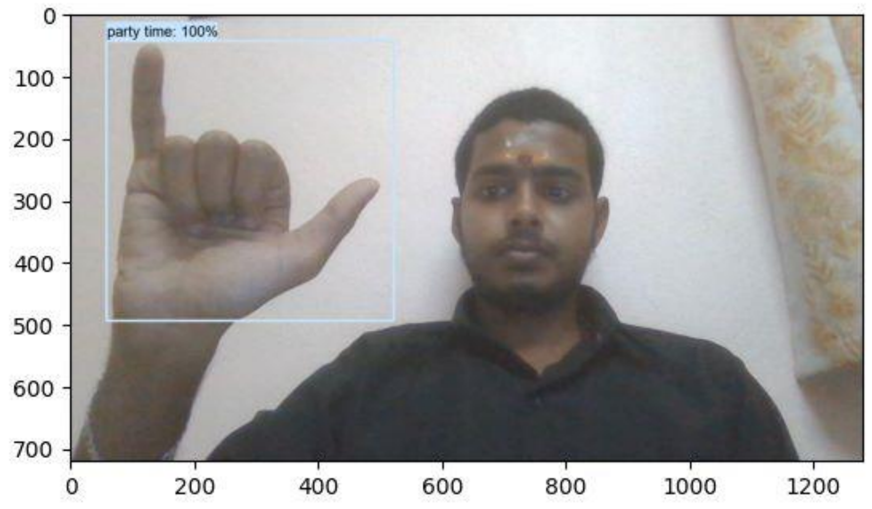
5. Single Image Detection

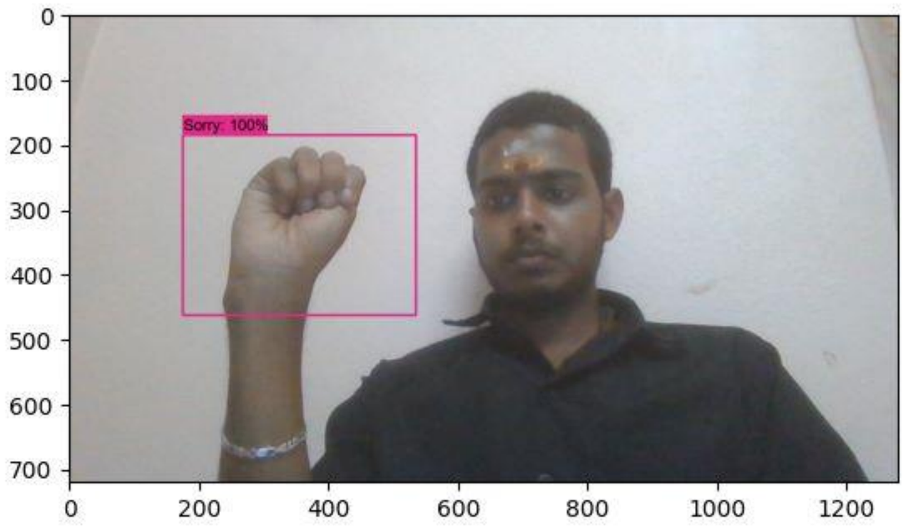


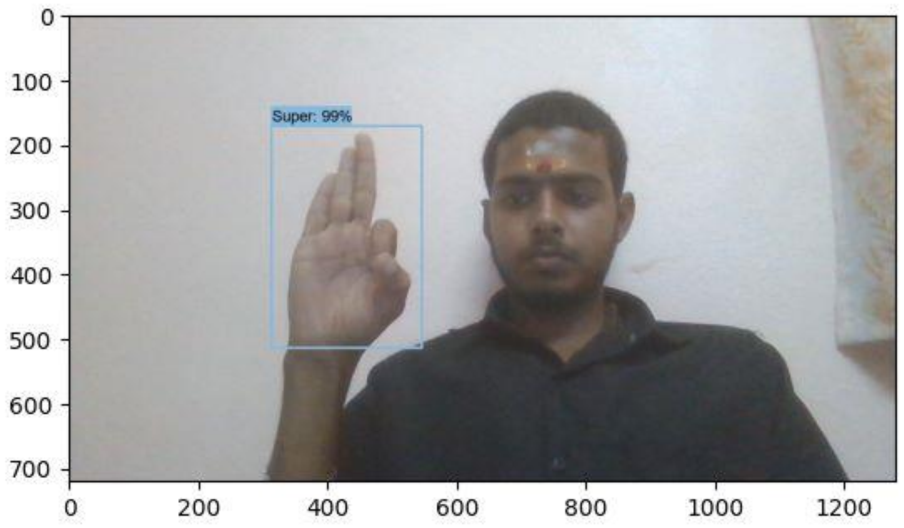

Help: 100%
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

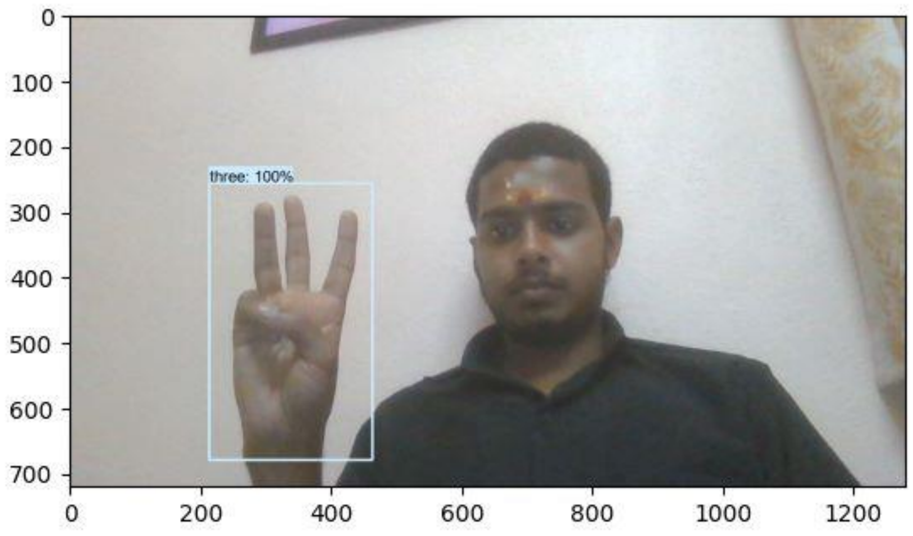

ok: 100%
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

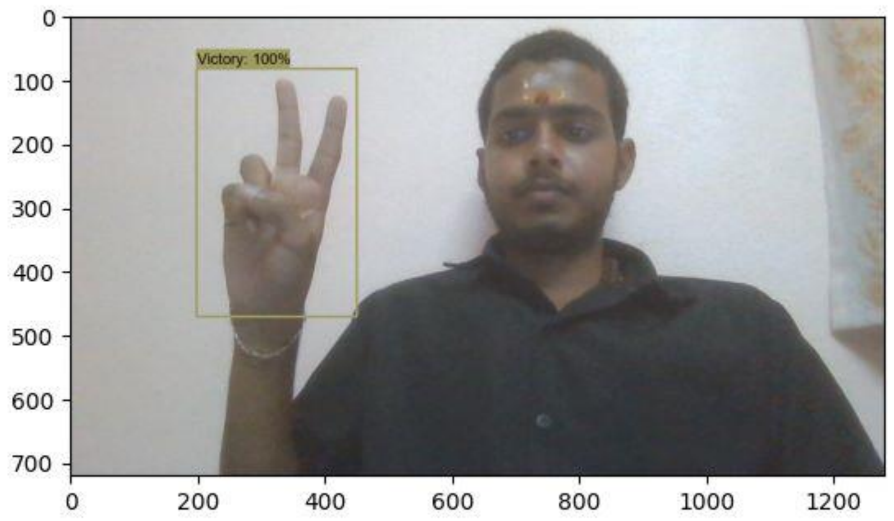

one: 100%
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200


party time: 100%

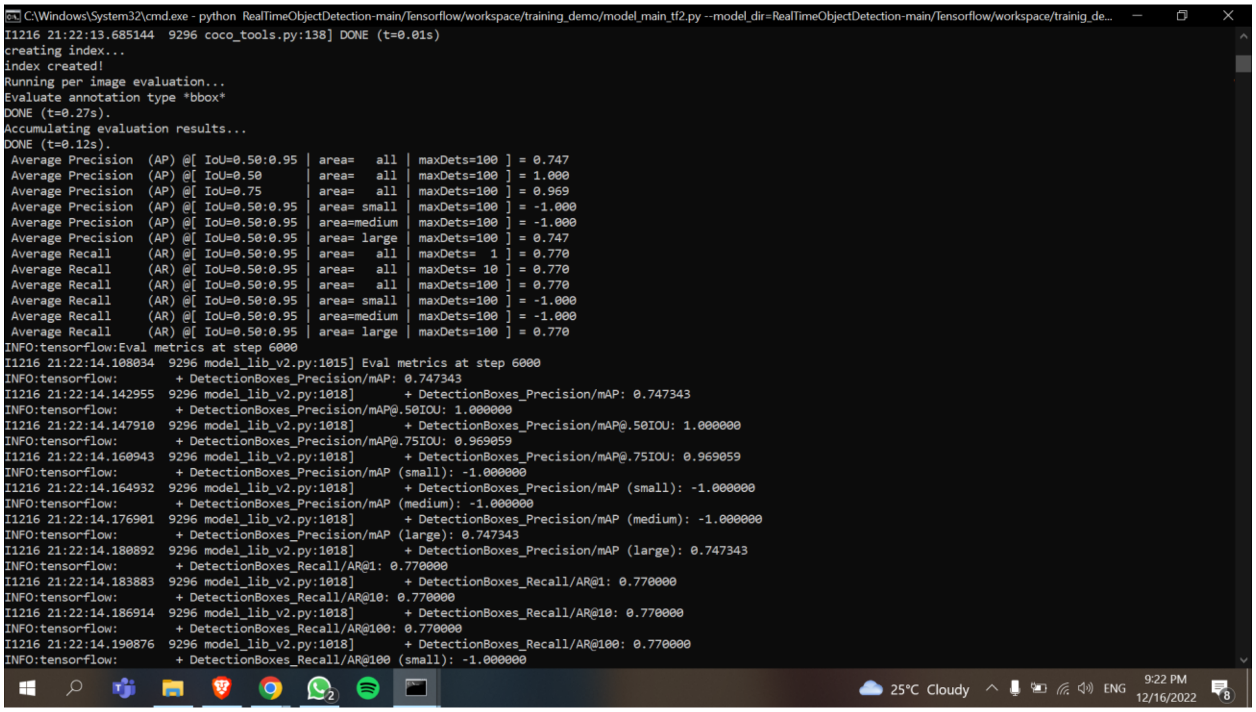

Sorry: 100%


Super: 99%

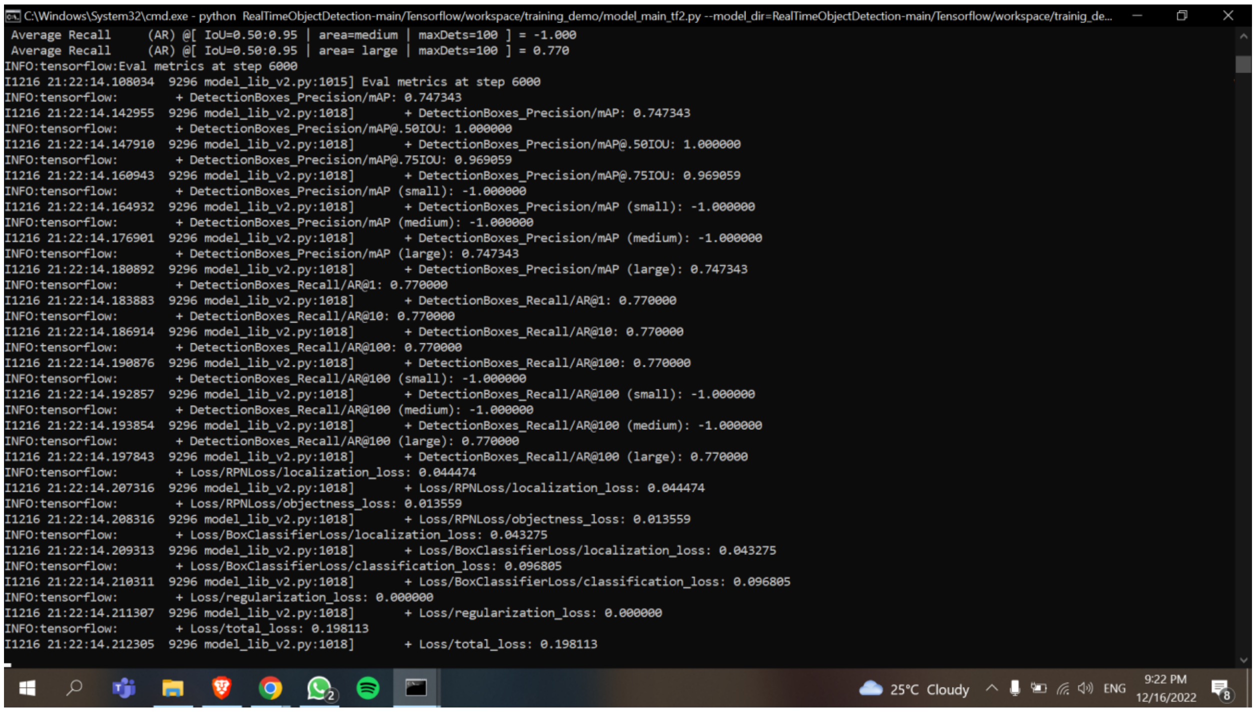

three: 100%
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200


Victory: 100%
0
100
200
300
400
500
600
700
0
200
400
600
800
1000
1200

The evaluation metrics are as follows:


C:\Windows\System32\cmd.exe - python RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/model_main_tf2.py --model_dir=RealTimeObjectDetection-main/Tensorflow/workspace/trainig_de...
25°C Cloudy
ENG
9:22 PM
12/16/2022

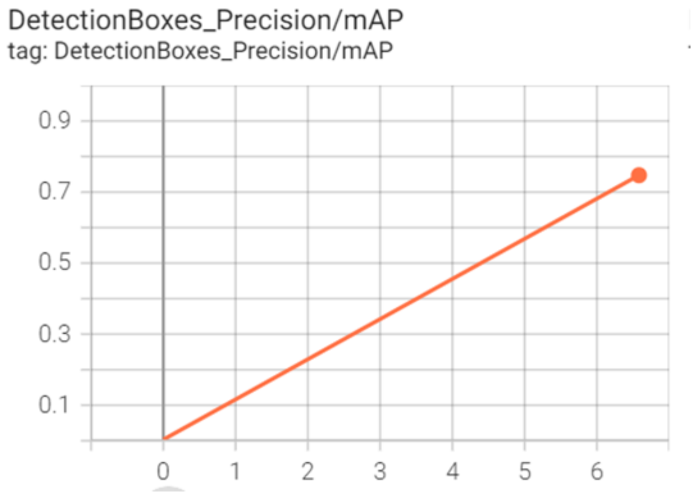
```
I1216 21:22:13.685144  9296 coco_tools.py:138] DONE (t=0.01s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *bbox*
DONE (t=0.27s).
Accumulating evaluation results...
DONE (t=0.12s).
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.747
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 1.000
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.969
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.747
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.770
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.770
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.770
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.770
INFO:tensorflow:Eval metrics at step 6000
I1216 21:22:14.108034  9296 model_lib_v2.py:1015] Eval metrics at step 6000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP: 0.747343
I1216 21:22:14.142955  9296 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP: 0.747343
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.50IOU: 1.000000
I1216 21:22:14.147910  9296 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP@.50IOU: 1.000000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.75IOU: 0.969059
I1216 21:22:14.160943  9296 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP@.75IOU: 0.969059
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (small): -1.000000
I1216 21:22:14.164932  9296 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (small): -1.000000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (medium): -1.000000
I1216 21:22:14.176901  9296 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (medium): -1.000000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (large): 0.747343
I1216 21:22:14.180892  9296 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (large): 0.747343
INFO:tensorflow:        + DetectionBoxes_Recall/AR@1: 0.770000
I1216 21:22:14.183883  9296 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@1: 0.770000
INFO:tensorflow:        + DetectionBoxes_Recall/AR@10: 0.770000
I1216 21:22:14.186914  9296 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@10: 0.770000
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100: 0.770000
I1216 21:22:14.190876  9296 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100: 0.770000
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (small): -1.000000
```

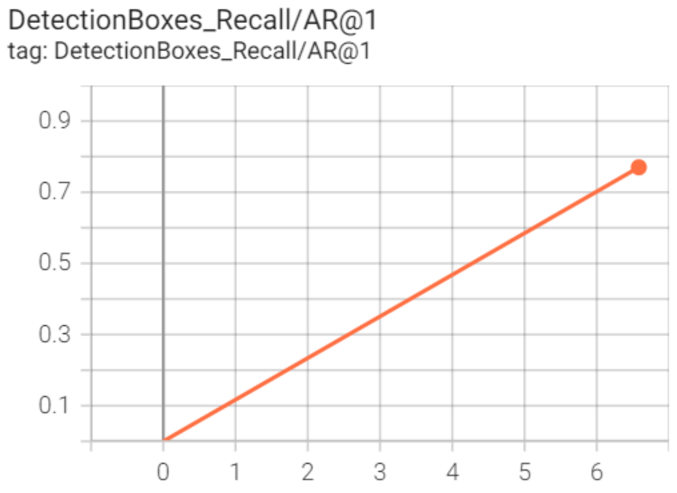

C:\Windows\System32\cmd.exe - python RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/model_main_tf2.py --model_dir=RealTimeObjectDetection-main/Tensorflow/workspace/trainig_de...
25°C Cloudy
ENG
9:22 PM
12/16/2022

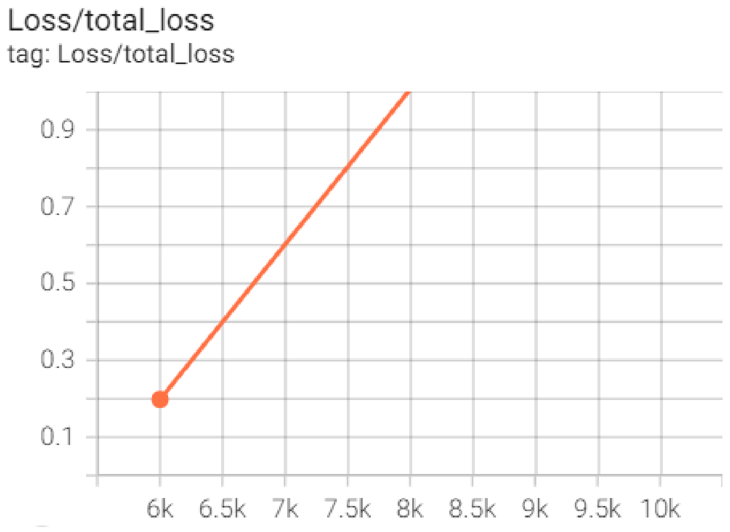
```
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.770
INFO:tensorflow:Eval metrics at step 6000
I1216 21:22:14.108034  9296 model_lib_v2.py:1015] Eval metrics at step 6000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP: 0.747343
I1216 21:22:14.142955  9296 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP: 0.747343
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.50IOU: 1.000000
I1216 21:22:14.147910  9296 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP@.50IOU: 1.000000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP@.75IOU: 0.969059
I1216 21:22:14.160943  9296 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP@.75IOU: 0.969059
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (small): -1.000000
I1216 21:22:14.164932  9296 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (small): -1.000000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (medium): -1.000000
I1216 21:22:14.176901  9296 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (medium): -1.000000
INFO:tensorflow:        + DetectionBoxes_Precision/mAP (large): 0.747343
I1216 21:22:14.180892  9296 model_lib_v2.py:1018]       + DetectionBoxes_Precision/mAP (large): 0.747343
INFO:tensorflow:        + DetectionBoxes_Recall/AR@1: 0.770000
I1216 21:22:14.183883  9296 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@1: 0.770000
INFO:tensorflow:        + DetectionBoxes_Recall/AR@10: 0.770000
I1216 21:22:14.186914  9296 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@10: 0.770000
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100: 0.770000
I1216 21:22:14.190876  9296 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100: 0.770000
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (small): -1.000000
I1216 21:22:14.192857  9296 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100 (small): -1.000000
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (medium): -1.000000
I1216 21:22:14.193854  9296 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100 (medium): -1.000000
INFO:tensorflow:        + DetectionBoxes_Recall/AR@100 (large): 0.770000
I1216 21:22:14.197843  9296 model_lib_v2.py:1018]       + DetectionBoxes_Recall/AR@100 (large): 0.770000
INFO:tensorflow:        + Loss/RPNLoss/localization_loss: 0.044474
I1216 21:22:14.207316  9296 model_lib_v2.py:1018]       + Loss/RPNLoss/localization_loss: 0.044474
INFO:tensorflow:        + Loss/RPNLoss/objectness_loss: 0.013559
I1216 21:22:14.208316  9296 model_lib_v2.py:1018]       + Loss/RPNLoss/objectness_loss: 0.013559
INFO:tensorflow:        + Loss/BoxClassifierLoss/localization_loss: 0.043275
I1216 21:22:14.209313  9296 model_lib_v2.py:1018]       + Loss/BoxClassifierLoss/localization_loss: 0.043275
INFO:tensorflow:        + Loss/BoxClassifierLoss/classification_loss: 0.096805
I1216 21:22:14.210311  9296 model_lib_v2.py:1018]       + Loss/BoxClassifierLoss/classification_loss: 0.096805
INFO:tensorflow:        + Loss/regularization_loss: 0.000000
I1216 21:22:14.211307  9296 model_lib_v2.py:1018]       + Loss/regularization_loss: 0.000000
INFO:tensorflow:        + Loss/total_loss: 0.198113
I1216 21:22:14.212305  9296 model_lib_v2.py:1018]       + Loss/total_loss: 0.198113
```

5. Evaluation Metrics

Results:


DetectionBoxes_Precision/mAP
tag: DetectionBoxes_Precision/mAP
0.9
0.7
0.5
0.3
0.1
0
1
2
3
4
5
6


DetectionBoxes_Recall/AR@1
tag: DetectionBoxes_Recall/AR@1
0.9
0.7
0.5
0.3
0.1
0
1
2
3
4
5
6


Loss/total_loss
tag: Loss/total_loss
0.9
0.7
0.5
0.3
0.1
6k
6.5k
7k
7.5k
8k
8.5k
9k
9.5k
10k

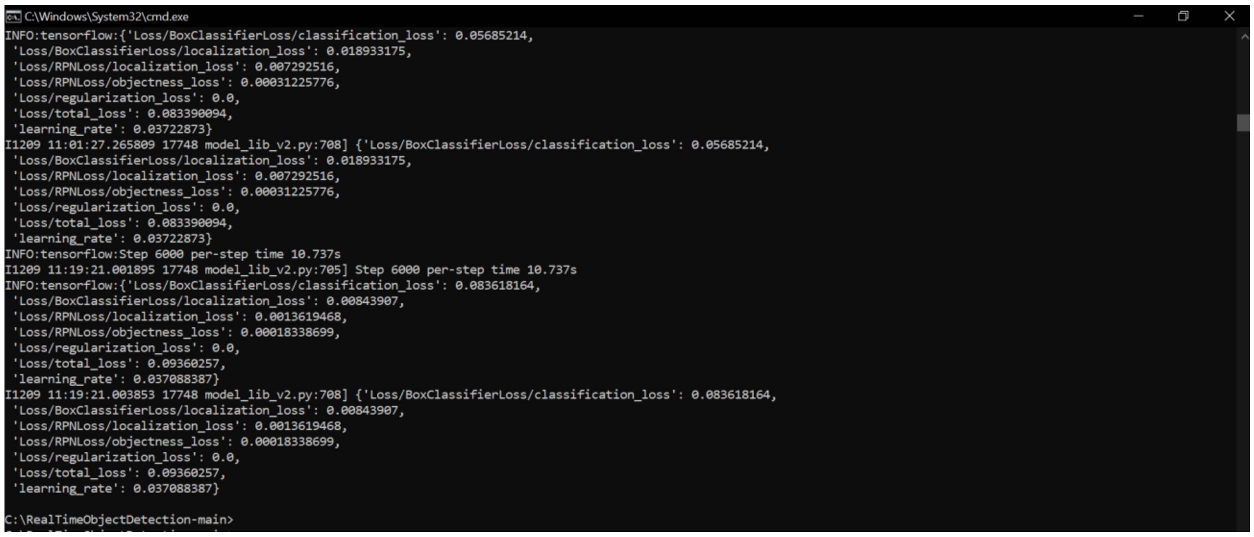
CHAPTER 4

CONCLUSION AND FUTURE WORK

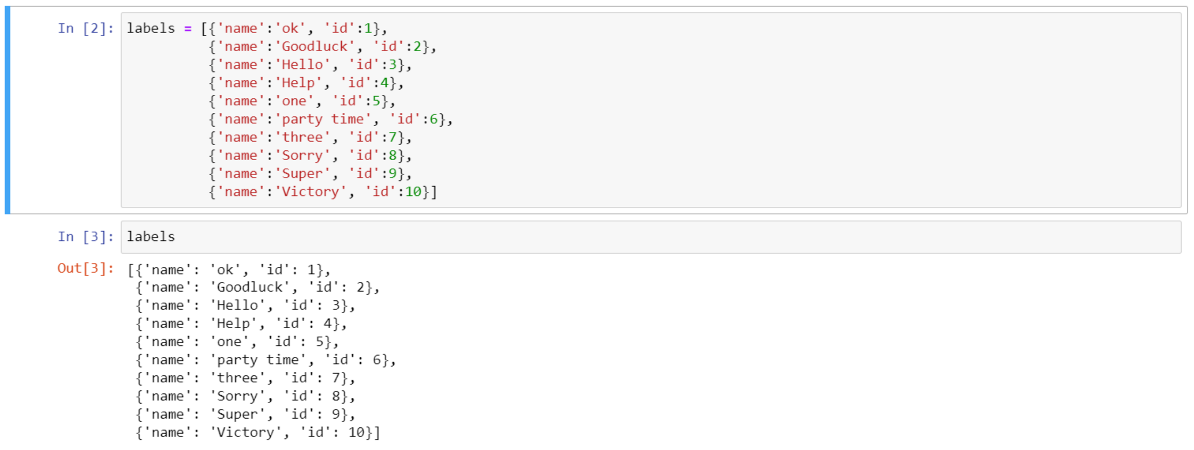
To conclude, In this project, a deep learning model was trained using the Pretrained faster RCNN Resnet model to classify and convert sign language to speech. The dataset consists of 10 classes. Each with 20 images, where 16 were used to train the model and 4 for testing the model. The model gave an overall precision of 0.74 and a recall of 0.77, which were good results. The confidence obtained for individual classification is chosen to be above 85% to convert the text to speech using the dispatch library. Transfer learning was used as it requires less data and gives the best results with high confidence in less time. This model can break the barrier between the physically impaired and the other. In the future, this model can be integrated into AR glasses or into a mobile app similar to google lens to work in real-time. The scope can also be increased by adding more classes and words and training on various data classes for real-time use. Also, new languages can be added so that sign language can be translated into different languages. Also, a feature can be provided to choose the tone of voice, gender of the voice etc.

CHAPTER 5

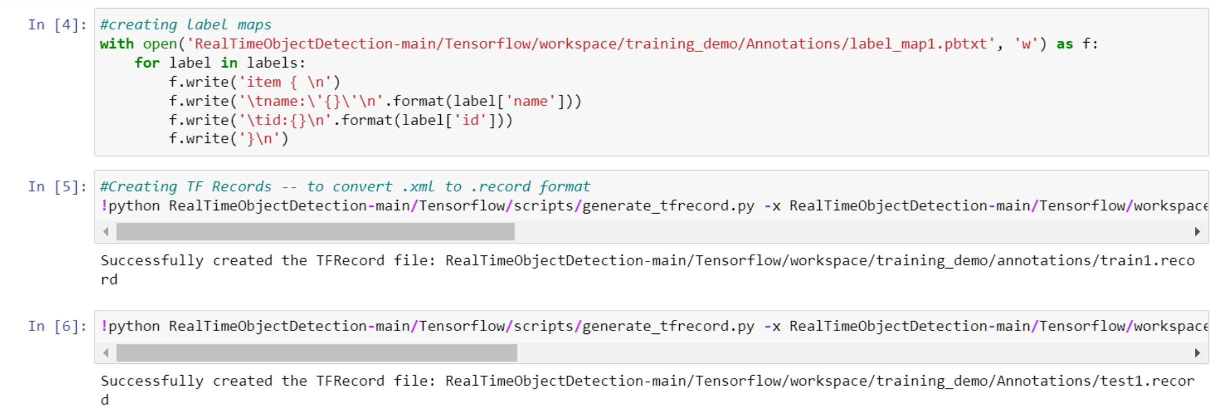
IMPLEMENTATION



```
C:\Windows\System32\cmd.exe
INFO:tensorflow:{'Loss/BoxClassifierLoss/classification_loss': 0.05685214,
 'Loss/BoxClassifierLoss/localization_loss': 0.018933175,
 'Loss/RPNLoss/localization_loss': 0.007292516,
 'Loss/RPNLoss/objectness_loss': 0.00031225776,
 'Loss/regularization_loss': 0.0,
 'Loss/total_loss': 0.083390094,
 'learning_rate': 0.03722873}
I1209 11:01:27.265809 17748 model_lib_v2.py:708] {'Loss/BoxClassifierLoss/classification_loss': 0.05685214,
 'Loss/BoxClassifierLoss/localization_loss': 0.018933175,
 'Loss/RPNLoss/localization_loss': 0.007292516,
 'Loss/RPNLoss/objectness_loss': 0.00031225776,
 'Loss/regularization_loss': 0.0,
 'Loss/total_loss': 0.083390094,
 'learning_rate': 0.03722873}
INFO:tensorflow:Step 6000 per-step time 10.737s
I1209 11:19:21.001895 17748 model_lib_v2.py:705] Step 6000 per-step time 10.737s
INFO:tensorflow:{'Loss/BoxClassifierLoss/classification_loss': 0.083618164,
 'Loss/BoxClassifierLoss/localization_loss': 0.00843907,
 'Loss/RPNLoss/localization_loss': 0.0013619468,
 'Loss/RPNLoss/objectness_loss': 0.00018338699,
 'Loss/regularization_loss': 0.0,
 'Loss/total_loss': 0.09360257,
 'learning_rate': 0.037088387}
I1209 11:19:21.003853 17748 model_lib_v2.py:708] {'Loss/BoxClassifierLoss/classification_loss': 0.083618164,
 'Loss/BoxClassifierLoss/localization_loss': 0.00843907,
 'Loss/RPNLoss/localization_loss': 0.0013619468,
 'Loss/RPNLoss/objectness_loss': 0.00018338699,
 'Loss/regularization_loss': 0.0,
 'Loss/total_loss': 0.09360257,
 'learning_rate': 0.037088387}

C:\RealTimeObjectDetection-main>
```



```
In [2]: labels = [{'name':'ok', 'id':1},
                  {'name':'Goodluck', 'id':2},
                  {'name':'Hello', 'id':3},
                  {'name':'Help', 'id':4},
                  {'name':'one', 'id':5},
                  {'name':'party time', 'id':6},
                  {'name':'three', 'id':7},
                  {'name':'Sorry', 'id':8},
                  {'name':'Super', 'id':9},
                  {'name':'Victory', 'id':10}]

In [3]: labels

Out[3]: [{'name': 'ok', 'id': 1},
         {'name': 'Goodluck', 'id': 2},
         {'name': 'Hello', 'id': 3},
         {'name': 'Help', 'id': 4},
         {'name': 'one', 'id': 5},
         {'name': 'party time', 'id': 6},
         {'name': 'three', 'id': 7},
         {'name': 'Sorry', 'id': 8},
         {'name': 'Super', 'id': 9},
         {'name': 'Victory', 'id': 10}]
```



```
In [4]: #creating label maps
        with open('RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/Annotations/label_map1.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')

In [5]: #Creating TF Records -- to convert .xml to .record format
        !python RealTimeObjectDetection-main/Tensorflow/scripts/generate_tfrecord.py -x RealTimeObjectDetection-main/Tensorflow/workspace

        Successfully created the TFRecord file: RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/annotations/train1.reco
        rd

In [6]: !python RealTimeObjectDetection-main/Tensorflow/scripts/generate_tfrecord.py -x RealTimeObjectDetection-main/Tensorflow/workspace

        Successfully created the TFRecord file: RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/Annotations/test1.recor
        d
```

Updating the pipeline.config file for the FAST RCNN RESNET Model

In [1]:
```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
path="RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/models/my-fast-rcnn-resnet/pipeline.config"
config=config_util.get_configs_from_pipeline_file(path)
#config
```

In [2]:
```
pipeline_config = pipeline_pb2.TrainEvalPipelineConfig()
with tf.io.gfile.GFile(path, "r") as f:
    proto_str = f.read()
    text_format.Merge(proto_str, pipeline_config)
```

In [3]:
```
pipeline_config.model.faster_rcnn.num_classes = 10
pipeline_config.train_config.batch_size = 2
pipeline_config.train_config.fine_tune_checkpoint = 'RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/pre-trained-
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path= 'RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/annotations/l
pipeline_config.train_input_reader.tf_record_input_reader.input_path[:] = ['RealTimeObjectDetection-main/Tensorflow/workspace/tra
pipeline_config.eval_input_reader[0].label_map_path = 'RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/annotatior
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[:] = ['RealTimeObjectDetection-main/Tensorflow/workspace/t
```

In [4]:
```
config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(path, "wb") as f:
    f.write(config_text)
```

In [5]:
```
#python RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/model_main_tf2.py --model_dir=RealTimeObjectDetection-mai
```

Load the latest model from the checkpoint

In [6]:
```
import os
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util
```

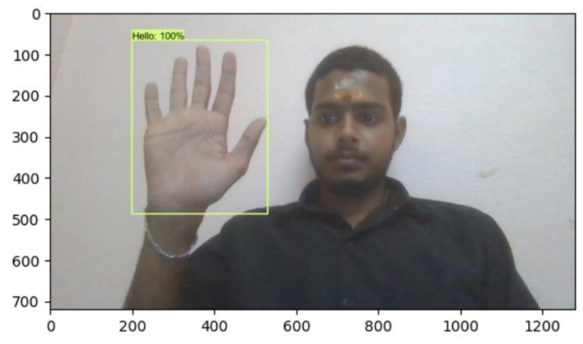
In [7]:
```
# Load pipeline config and build a detection model
configs = config_util.get_configs_from_pipeline_file('RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/models/my-f
rcnn_resnet = model_builder.build(model_config=configs['model'], is_training=False)

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=rcnn_resnet)
ckpt.restore(os.path.join('RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/models/my-fast-rcnn-resnet', 'ckpt-9')
#latest model

@tf.function
def detect_fn(image):
    image, shapes = rcnn_resnet.preprocess(image) #to resize the image
    prediction_dict = rcnn_resnet.predict(image, shapes)
    detections = rcnn_resnet.postprocess(prediction_dict, shapes)
    return detections
```

Single image detection

In [8]:
```
import cv2
import numpy as np
from win32com.client import Dispatch

from matplotlib import pyplot as plt
%matplotlib inline
category_index = label_map_util.create_category_index_from_labelmap('RealTimeObjectDetection-main/Tensorflow/workspace/training_c
IMAGE_PATH = 'RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/images/train1/21.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'))
print (num_detections)
detections = {key: value[0, :num_detections].numpy()
              for key, value in detections.items()}
detections['num_detections'] = num_detections #detections regions
print(detections['num_detections'])

# detection_classes should be ints.
detections['detection_classes'] = detections['detection_classes'].astype(np.int64) #index of the class starting from 0;

image_np_with_detections = image_np.copy()
```

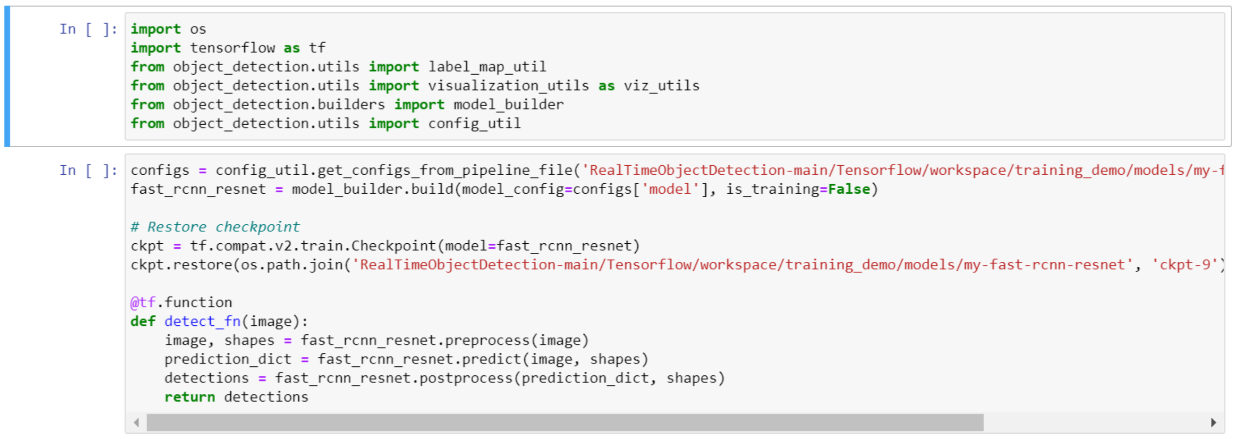
```
viz_utils.visualize_boxes_and_labels_on_image_array(
            image_np_with_detections,
            detections['detection_boxes'],
            detections['detection_classes']+1,
            detections['detection_scores'],
            category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=5,
            min_score_thresh=.8,
            agnostic_mode=False)
plt.imshow(cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB))
#plt.resize(640, 640)
plt.show()

if detections['detection_scores'][0]*100>85:
        speech=(detections['detection_classes']+1)[0]
        speak = Dispatch("SAPI.SpVoice").Speak
        speak(category_index[speech]['name'])
        print(category_index[speech]['name'])
```

```
INFO:tensorflow:depth of additional conv before box predictor: 0
WARNING:tensorflow:From c:\users\talas\appdata\local\programs\python\python38\lib\site-packages\tensorflow\python\autograph\imp
l\api.py:459: Tensor.experimental_ref (from tensorflow.python.framework.ops) is deprecated and will be removed in a future vers
ion.
Instructions for updating:
Use ref() instead.
300
300
```



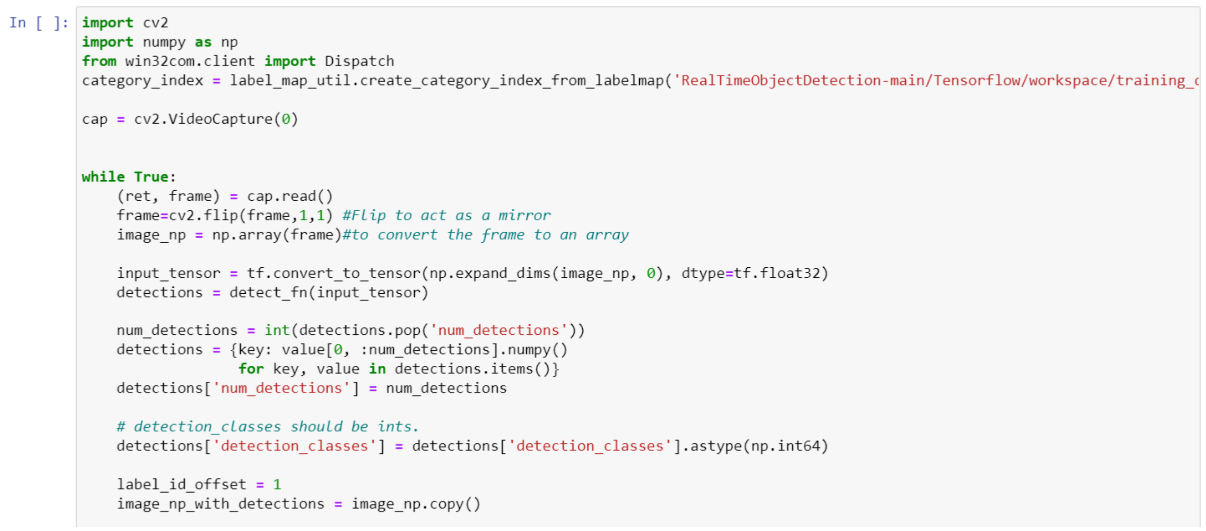
```
Hello
```

In [ ]:
```
#python RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/model_main_tf2.py --model_dir=RealTimeObjectDetection-mai
```

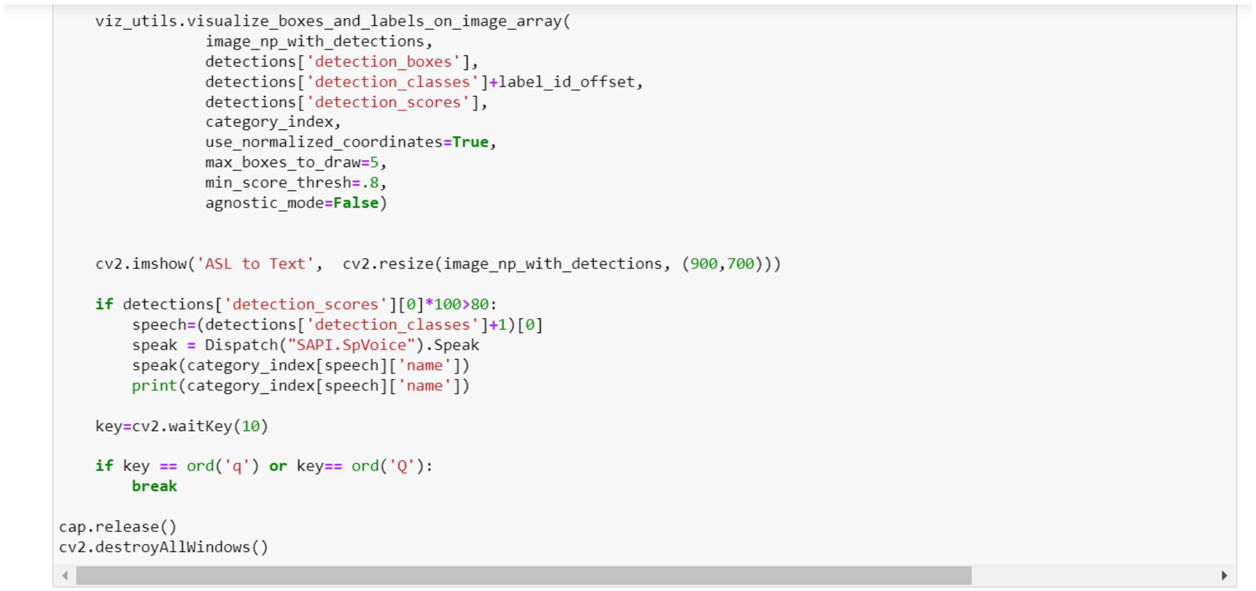


```
In [ ]: import os
        import tensorflow as tf
        from object_detection.utils import label_map_util
        from object_detection.utils import visualization_utils as viz_utils
        from object_detection.builders import model_builder
        from object_detection.utils import config_util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

```
In [ ]: configs = config_util.get_configs_from_pipeline_file('RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/models/my-f
        fast_rcnn_resnet = model_builder.build(model_config=configs['model'], is_training=False)

        # Restore checkpoint
        ckpt = tf.compat.v2.train.Checkpoint(model=fast_rcnn_resnet)
        ckpt.restore(os.path.join('RealTimeObjectDetection-main/Tensorflow/workspace/training_demo/models/my-fast-rcnn-resnet', 'ckpt-9')

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



```
In [ ]: import cv2
        import numpy as np
        from win32com.client import Dispatch
        category_index = label_map_util.create_category_index_from_labelmap('RealTimeObjectDetection-main/Tensorflow/workspace/training_c

        cap = cv2.VideoCapture(0)


        while True:
            (ret, frame) = cap.read()
            frame=cv2.flip(frame,1,1) #Flip to act as a mirror
            image_np = np.array(frame)#to convert the frame to an array

            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)


    cv2.imshow('ASL to Text',  cv2.resize(image_np_with_detections, (900,700)))

    if detections['detection_scores'][0]*100>80:
        speech=(detections['detection_classes']+1)[0]
        speak = Dispatch("SAPI.SpVoice").Speak
        speak(category_index[speech]['name'])
        print(category_index[speech]['name'])

    key=cv2.waitKey(10)

    if key == ord('q') or key== ord('Q'):
        break

cap.release()
cv2.destroyAllWindows()
```

CHAPTER 6

REFERENCES:

- V. N. T. Truong, C. Yang and Q. Tran, "A translator for American sign language to text and speech," 2016 IEEE 5th Global Conference on Consumer Electronics, 2016, pp. 1-2, doi: 10.1109/GCCE.2016.7800427.
- K. Nimisha and A. Jacob, "A Brief Review of the Recent Trends in Sign Language Recognition," 2020 International Conference on Communication and Signal Processing (ICCSP), 2020, pp. 186-190, doi: 10.1109/ICCSP48568.2020.9182351.
- P. Vijayalakshmi and M. Aarthi, "Sign language to speech conversion," 2016 International Conference on Recent Trends in Information Technology (ICRTIT), 2016, pp. 1-6, doi: 10.1109/ICRTIT.2016.7569545.
- M. M. Chandra, S. Rajkumar and L. S. Kumar, "Sign Languages to Speech Conversion Prototype using the SVM Classifier," TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019, pp. 1803-1807, doi: 10.1109/TENCON.2019.8929356.
- Mujahid, A.; Awan, M.J.; Yasin, A.; Mohammed, M.A.; Damaševičius, R.; Maskeliūnas, R.; Abdulkareem, K.H. Real-Time Hand Gesture Recognition Based on Deep Learning YOLOv3 Model. Appl. Sci. 2021, 11, 4164.
- Ji-Hae Kim, Gwang-Soo Hong, Byung-Gyu Kim, Debi P. Dogra, “deepGesture: Deep learning-based gesture recognition scheme using motion sensors” , Displays, Volume 55, 2018, Pages 38-45, ISSN 0141-9382.

BIO DATA

Team Member 1

NAME: BEZAWADA NEHA CHOWDARY

MOBILE NUMBER: 8688148463

EMAIL: neha.19bce7097@vitap.ac.in

PERMANENT ADDRESS: ONGOLE

Team Member 2

NAME: TALASILA SRI HARSHA

MOBILE NUMBER: 9347891532

EMAIL: harsha.19bce7490@vitap.ac.in

PERMANENT ADDRESS: VIJAYAWADA

Team Member 3

NAME: KOGANTI DHANUSH KUMAR

MOBILE NUMBER: 7337456147

EMAIL: dhanush.19bce7639@vitap.ac.in

PERMANENT ADDRESS: VIJAYAWADA