

SIGN LANGUAGE RECOGNITION

A PROJECT REPORT SUBMITTED BY:

Aishwarya: 4NM18CS009

Aparna: 4NM18CS021

For the course: Machine Learning (18CS601)

UNDER THE GUIDANCE OF:

Mrs. Divya Jennifer D'souza

Assistant Professor Gd I

Department of Computer science and Engineering

In partial fulfilment of the requirement for the award of the Degree of

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE AND ENGINEERING

From

Visvesvaraya Technological University, Belagavi



NITTE
EDUCATION TRUST

N.M.A.M. INSTITUTE OF TECHNOLOGY

(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)

Nitte – 574 110, Karnataka, India

(ISO 9001:2015 Certified), Accredited with 'A' Grade by NAAC

☎: 08258 - 281039 - 281263, Fax: 08258 - 281265

Department of Computer Science and Engineering

B.E. CSE Program Accredited by NBA, New Delhi from 1-7-2018

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
N.M.A.M. INSTITUTION OF TECHNOLOGY



(An Autonomous Institution affiliated to VTU, Belagavi)
(NBA Accredited, ISO 9001:2008 Certified)
Nitte-574110, Karkala, Udupi District, Karnataka, India

Department of Computer Science and Engineering

CERTIFICATE

Certified that the Mini Project work entitled

SIGN LANGUAGE RECOGNITION

is a bonafide work carried out by

Aishwarya: 4NM18CS009

Aparna: 4NM18CS021

In partial fulfilment of requirements for the award of Bachelor of Engineering Degree in Computer Science and Engineering prescribed by Visvesvaraya Technology University,

Belgaum during the year 2020-2021

It is certified that all corrections/suggestions indicated for Internal Assessment have been

Incorporated in the report

The mini project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree

Name & Signature of Guide(s)

Mrs. Divya Jennifer D'souza
Assistant Professor, Gd I
Department of CSE

Name & Signature of HOD

Dr. K.R Udaya Kumar
Head of the Department
Department of CSE

ACKNOWLEDGMENT

We believe that our project will be complete only after we thank the people who have contributed to make this project successful.

First and foremost, our sincere thanks to our beloved principal, **Dr Niranjan N Chiplunkar** for giving us an opportunity to carry out our project work at our college and providing us with all the needed facilities.

We express our deep sense of gratitude and indebtedness to our guide **Mrs Divya Jennifer D'souza**, Assistant Professor GD I, Department of Computer Science and Engineering, for his inspiring guidance, constant encouragement, support and suggestion for improvement during the course of our project.

We sincerely thank **Dr K R Udaya Kumar Reddy**, Head of Department of Computer Science and Engineering, Nitte Mahalinga Adyantaya Memorial Institute of Technology, Nitte.

We also thank all those who have supported us throughout the entire duration of our project.

Finally, we thank the staff members of the Department of Computer Science and Engineering and all our friends for their honest opinions and suggestions throughout the course of our project.

Aishwarya (4NM18CS009)

Aparna (4NM18CS021)

TABLE OF CONTENTS

Sl. No.	Contents	Page No.
1.	Abstract	1
2.	Introduction	2 - 3
3.	Literature Survey	4 - 5
4.	Design & Implementation	6 - 12
5.	Result	13
6.	Conclusion & Future Work	14
7.	References	15

ABSTRACT

Sign language is the mode of communication which uses visual ways like expressions, hand gestures, and body movements to convey meaning. Sign language is extremely helpful for people who face difficulty with hearing or speaking. Sign language recognition refers to the conversion of these gestures into words or alphabets of existing formally spoken languages. Thus, conversion of sign language into words by an algorithm or a model can help bridge the gap between people with hearing or speaking impairment and the rest of the world.

Our project aims at extending a step forward in this field by collecting a dataset and then use various feature extraction techniques to extract useful information which is then input into various supervised learning techniques. To more easily approach the problem and obtain reasonable results, we experimented with just up to 5 different classes/words in our self-made dataset instead of all 26 possible letters.

We achieved a classification accuracy of 97% on a randomly selected set of test data using our trained model. In addition to the work we did on static images, we also created a live demo version of the project which can be run at a little less than 2 seconds per frame to classify signed hand gestures from any person.

INTRODUCTION

The sign language is used widely by people who are deaf-dumb these are used as a medium for communication. A sign language is nothing but composed of various gestures formed by different shapes of hand, its movements, orientations as well as the facial expressions. There are around 466 million people worldwide with hearing loss and 34 million of these are children. 'Deaf' people have very little or no hearing ability. They use sign language for communication. People use different sign languages in different parts of the world.

Fingerspelling	Word level sign vocabulary	Non-manual features
Used to spell words letter by letter .	Used for the majority of communication.	Facial expressions and tongue, mouth and body position.

Machine learning

Machine learning is a subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn" (i.e., progressively improve performance on a specific task) with data, without being explicitly programmed. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome.

Supervised learning

The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.

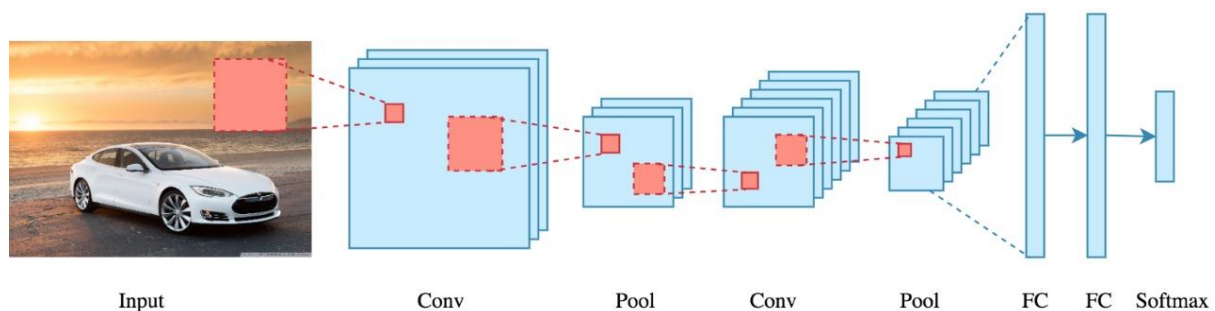
APPROACHES AND METHODS

TensorFlow:

TensorFlow is an open source software library for numerical computation. First, we define the nodes of the computation graph, then inside a session, the actual computation takes place. TensorFlow is widely used in Machine Learning.

CNN:

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyse visual imagery. Now when we think of a neural network, we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.



OpenCV:

Open Source Computer Vision is an open source library of programming functions used for real-time computer-vision. It is mainly used for image processing, video capture and analysis for features like face and object recognition. It is written in C++ which is its primary interface, however bindings are available for Python, Java, MATLAB/OCTAVE.

LITERATURE SURVEY

In the recent years there has been tremendous research done on the hand gesture recognition.

With the help of literature survey done we realized the basic steps in hand gesture recognition are:

- Data acquisition
- Data pre processing
- Feature extraction
- Gesture classification

Data acquisition:

The different approaches to acquire data about the hand gesture can be done in the following ways:

1. Use of sensory devices

It uses electromechanical devices to provide exact hand configuration, and position. Different glove-based approaches can be used to extract information. But it is expensive and not user friendly.

2. Vision based approach

In vision-based methods computer camera is the input device for observing the information of hands or fingers. The Vision Based methods require only a camera, thus realizing a natural interaction between humans and computers without the use of any extra devices. These systems tend to complement biological vision by describing artificial vision systems that are implemented in software and/or hardware.

The main challenge of vision-based hand detection is to cope with the large variability of human hand's appearance due to a huge number of hand movements, to different skin-colour possibilities as well as to the variations in viewpoints, scales, and speed of the camera capturing the scene.

Data pre-processing and Feature extraction for vision-based approach:

We can also extract necessary image which is to be trained by applying a filter

called Gaussian blur. The filter can be easily applied using open computer vision also known as OpenCV and is described in [2].

Feature extraction :

For extracting necessary image which is to be trained we can use instrumented gloves as mentioned in [3]. This helps reduce computation time for pre-processing and can give us more concise and accurate data compared to applying filters on data received from video extraction.

We tried doing the hand segmentation of an image using colour segmentation techniques but as mentioned in the research paper skin colour and tone is highly dependent on the lighting conditions due to which the output, we got for the segmentation we tried to do were no so great. Moreover we have a huge number of symbols to be trained for our project many of which look similar to each other like the gesture for symbol 'V' and digit '2', hence we decided that in order to produce better accuracies for our large number of symbols, rather than segmenting the hand out of a random background we keep background of hand a stable single colour so that we don't need to segment it on the basis of skin colour. This would help us to get better results.

Gesture classification :

In [1] Naïve Bayes Classifier is used which is an effective and fast method for static hand gesture recognition. It is based on classifying the different gestures according to geometric based invariants which are obtained from image data after segmentation. Thus, unlike many other recognition methods, this method is not dependent on skin colour. The gestures are extracted from each frame of the video, with a static background.

According to paper on "Human Hand Gesture Recognition Using a Convolution Neural Network" by Hsien-I Lin, Ming-Hsiang Hsu, and Wei-Kai Chen graduates of Institute of Automation Technology National Taipei University of Technology Taipei, Taiwan, they construct a skin model to extract the hand out of an image and then apply binary threshold to the whole image. After obtaining the threshold image they calibrate it about the principal axis in order to centre the image about it. They input this image to a convolutional neural network model in order to train and predict the outputs. They have trained their model over 7 hand gestures and using their model they produce an accuracy of around 95% for those 7 gestures.

DESIGN AND IMPLEMENTATION

STEPS TO BUILD THE MODEL

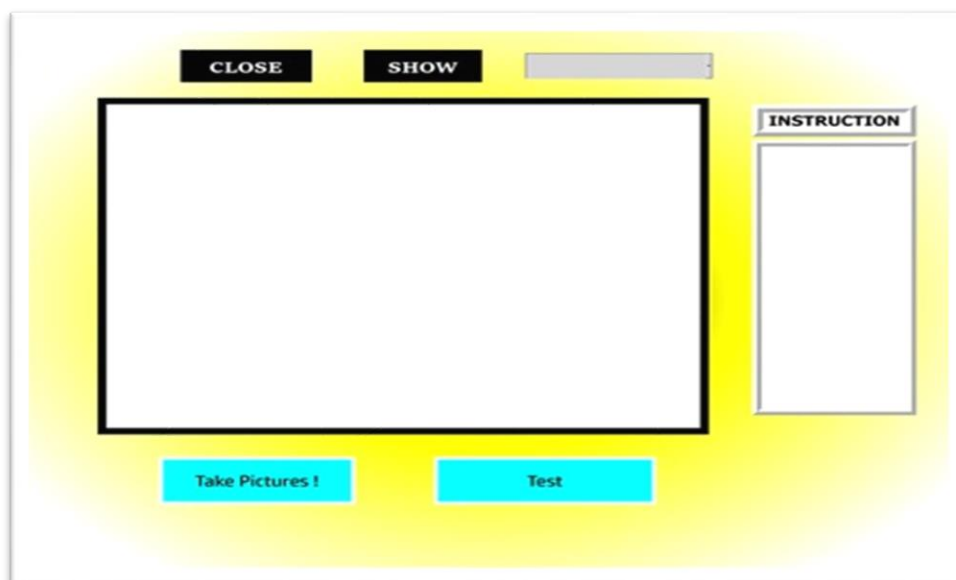
1. Collect the images/dataset for deep learning using your webcam and OpenCV.
2. Image pre-processing & labelling for sign language detection.
3. Setup Tensor-flow Object Detection API.
4. Use transfer learning to train a deep learning model.
5. Detect sign language in real time using OpenCV.

STEP 1: COLLECTING DATASET

Image Capture:

This is the first step in sign recognition. Camera interfacing is a very critical part. Web camera is used to capture the hand gesture. Now web camera is also in built in laptops & one can use external camera for interfacing. But captured images need to be in high definition which is done with the help of OpenCV. So, selection of good webcam & its interfacing is an important task of this method.

UI for this project has been developed using PyQt5.



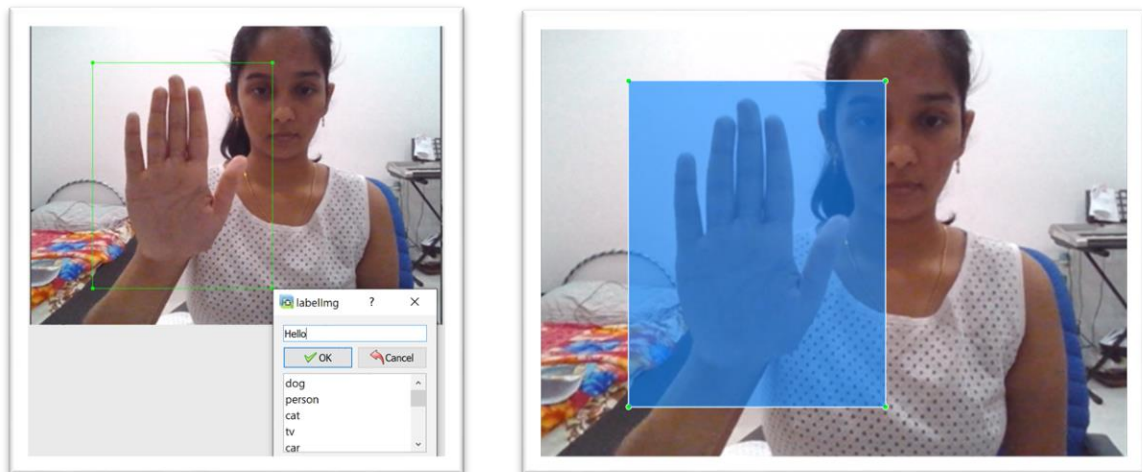
STEP 2: IMAGE PRE-PROCESSING AND LABELLING

Image Pre-processing:







Contains cropping, filtering, brightness & contrast adjustment & many more. To do such process Image enhancement, Image cropping & Image Segmentation methods are used. Captured Images are in the form of RGB. So, the first step is to convert RGB images to binary images then cropping of image is to be done so that unwanted part of images can be removed. And now enhancement can be done in certain selected area. In Image segmentation, Edge detection method is used which can detect the boundary of cropped images which is further used for feature extraction method.

Image Labelling:

Labelling the Images with the help of LabelImg :



Storing the labelled and cropped images in the xml format.

 Bye_1	05-04-2021 04:02 AM	PNG File	1,057 KB
 Bye_1	05-04-2021 05:04 AM	XML Document	1 KB
 Bye_2	05-04-2021 04:02 AM	PNG File	1,065 KB
 Bye_2	05-04-2021 05:03 AM	XML Document	1 KB
 Bye_3	05-04-2021 04:02 AM	PNG File	1,064 KB
 Bye_3	05-04-2021 04:56 AM	XML Document	1 KB

The above is done for every image taken as input for each category.

```

<?xml version="1.0"?>
- <annotation>
  <folder>imgs</folder>
  <filename>Bye_1.png</filename>
  <path>D:\projects\signlangdetection\Tensorflow\workspace\imgs\Bye_1.png</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>1280</width>
    <height>720</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>Bye</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    + <bndbox>
  </object>
</annotation>

```

This is how the XML form of our PNG images look like.

STEP 3: SETTING UP TENSORFLOW OBJECT DETECTION API

The TensorFlow Model Zoo is a collection of pre-trained object detection architectures that have performed tremendously well on the COCO dataset.

Mobile Net is an object detector released in 2017 as an efficient CNN architecture designed for mobile and embedded vision application. This architecture uses proven depth-wise separable convolutions to build lightweight deep neural networks.

1) Download the model file from the TensorFlow model zoo.

- a) Download the MobileNetV2 pre-trained model to your machine
- b) Move it to the object detection folder.
- c) Create a main.py python script to run the real-time program.

2) Setting up the configuration file and model pipeline.

- a) Identifying the path to the pipeline config of our MobileNetV2 model. This configuration file defines the model architecture and params.
- b) Specifying the checkpoint file of the model to be used (model_dir).
- c) Initialize model prediction by passing in the config path of the model.
- d) Use TensorFlow to restore the model's last checkpoint by specifying the checkpoint directory.

```
# Load Train Model From Checkpoint
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

# 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-6')).expect_partial()
```

STEP 4: USE TRANSFER LEARNING TO TRAIN A DEEP LEARNING MODEL

Transfer Learning is a machine learning technique where models are trained on (usually) larger data sets and refactored to fit more specific or niche data. This is done by recycling a portion of the weights from the pre-trained model and reinitializing or otherwise altering weights at shallower layers. The primary benefits of such a technique are its less demanding time and data requirements.

Transfer learning can be done in two ways : **Last layers-only retraining & Full model retraining**

$$Loss = \frac{1}{N} \sum_{i=1}^N -\log \left(\frac{e^{f_i, y_i}}{\sum_{j=1}^C e^{f_i, j}} \right) \quad (1)$$

$$f_j(z) = \frac{e^{z_j}}{\sum_{k=1}^C e^{z_k}} \quad (2)$$

N = total number of training examples
C = total number of classes

The TensorFlow Object Detection API requires the data to be in the somewhat obscure TFRecord format. Understanding TFRecord and getting it right is not an easy task and may take some time.

1) Set up Paths and Create Label Map.

```
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/'
```

```
labels = [
    {'name': 'Hello', 'id': 1},
    {'name': 'Bye', 'id': 2},
    {'name': 'Okay', 'id': 3},
    {'name': 'Yes', 'id': 4},
    {'name': 'No', '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')
```

2) Create TF Records

```
!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'}
```

3) Download TF Models Pretrained Models from Tensorflow Model Zoo.

```
!cd Tensorflow && git clone https://github.com/tensorflow/models
```

4) Copy Model Config to Training Folder.

```
CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
!mkdir {'Tensorflow\workspace\models\\'+CUSTOM_MODEL_NAME}
!cp {PRETRAINED_MODEL_PATH + '/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/pipeline.config'}
{MODEL_PATH + '/' + CUSTOM_MODEL_NAME}
```


5) Update Config For Transfer Learning.

```
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_PATH = MODEL_PATH + '/' + CUSTOM_MODEL_NAME + '/pipeline.config'
config = config_util.get_configs_from_pipeline_file(CONFIG_PATH)

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)

pipeline_config.model.ssd.num_classes = 2
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/ckpt-0'
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']

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

6) Train the model. (retraining **the whole-model** therefore **5000 steps**)

```
print("""python {}/research/object_detection/model_main_tf2.py --model_dir={} --pipeline_config_path={} --num_train_steps=5000""".format(
    API_MODEL_PATH, MODEL_PATH, CUSTOM_MODEL_NAME, MODEL_PATH, CUSTOM_MODEL_NAME))
```

7) Accuracy of our model

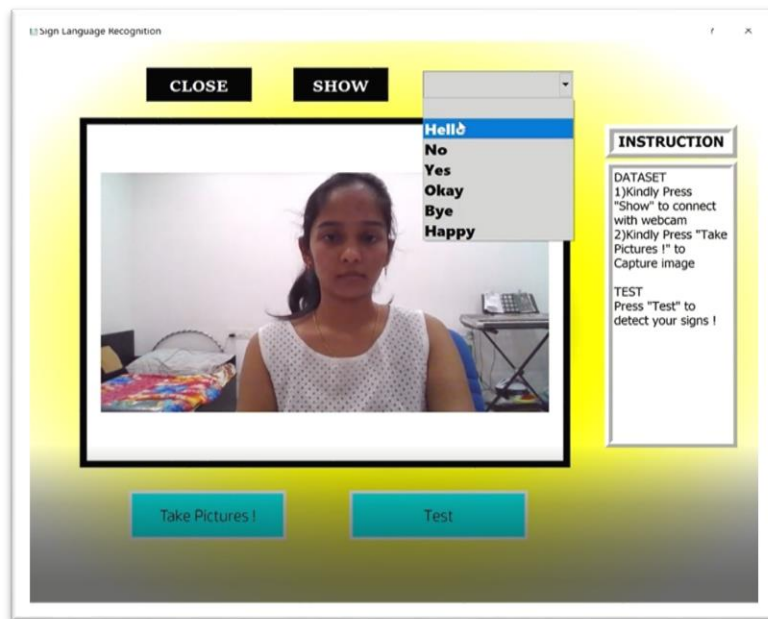
Last step of training (5000th step) -> time taken = **0.922s**, loss=**0.339**

```
INFO:tensorflow:Step 5000 per-step time 0.922s loss=0.339
I0426 16:42:35.023545 19256 model_lib_v2.py:682] Step 5000 per-step time 0.922s loss=0.339
```

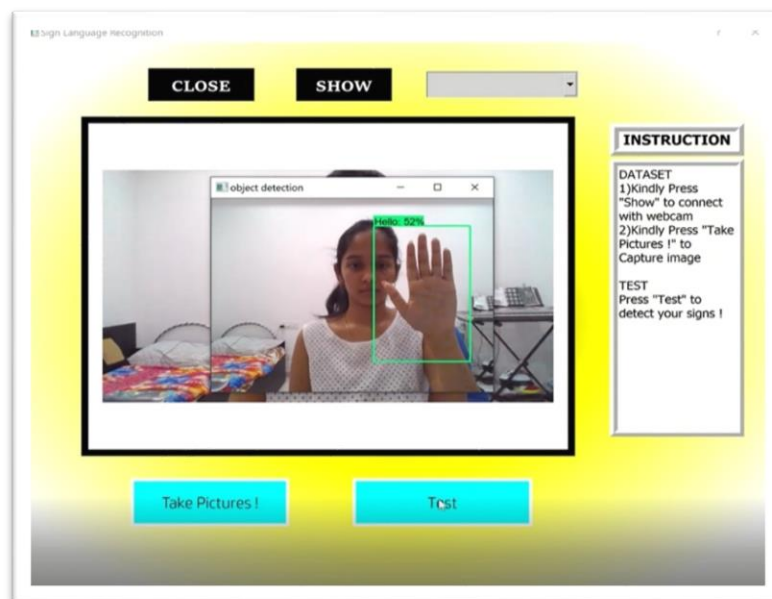
STEP 5: REAL TIME DETECTION

Our UI sends images to our server one by one. Each time, the server classifies the image and presents probabilities for each word. It keeps a running cache of classified images. When it feels confident about the sign being made by the user, it records the top most likely word based on the cache.

Collecting dataset



Testing the Model

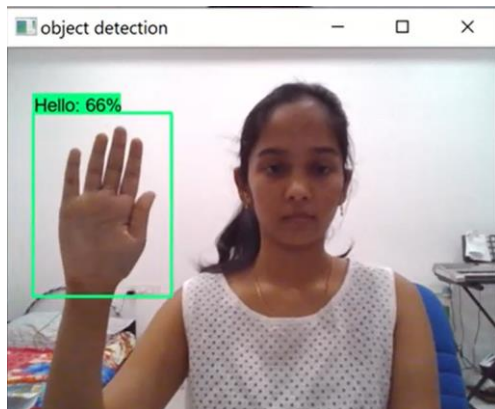


We have successfully developed sign language detection project. This is an interesting machine learning python project to gain expertise. This can be further extended for detecting various other English words.

RESULT

1. Obtaining video of the user signing (input)
2. Classifying each frame in the video to a word
3. Reconstructing and displaying the most likely word along with its classification scores/percentage (output).

Result for HELLO



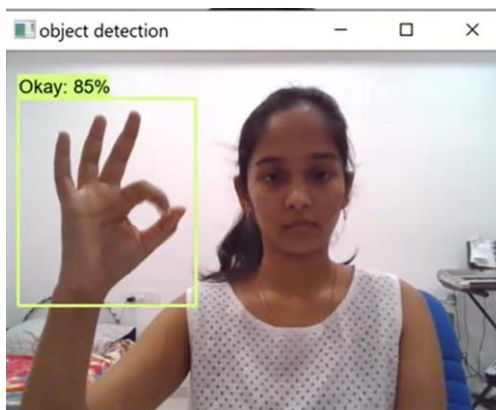
Result for BYE



Result for NO



Result for OKAY



Time Taken to detect = 0.922s

Accuracy of our model = 97%

CONCLUSION AND FUTURE WORK:

Conclusion:

Sign language recognition is a hard problem if we consider all the possible combinations of gestures that a system of this kind needs to understand and translate. That being said, probably the best way to solve this problem is to divide it into simpler problems, and the system presented here would correspond to a possible solution to one of them.

The system performed reasonably well and it was demonstrated that we can build a first-person sign language translation system using only cameras and convolutional neural networks.

However, the challenge in transfer learning stems from the differences between the original data used to train and the new data being classified. Larger differences in these data sets often require re-initializing or increasing learning rates for deeper layers in the net.

Future Work:

Additional Models: We focused our efforts on optimizing Mobile Net, but it would be worth exploring different nets that have also been proven effective at image classification.

Image pre-processing: We believe that the classification task could be made much simpler if there is very heavy pre-processing done on the images. This would include contrast adjustment, background subtraction and potentially cropping. A more robust approach would be to use another CNN to localize and crop the hand.

REFERENCES

- [1] Pujan Ziaie, Thomas Müller, Mary Ellen Foster, and Alois Knoll “A Naive Bayes Munich, Dept. of Informatics VI, Robotics and Embedded Systems, Boltzmannstr. 3, DE-85748 Garching, Germany.

- [2] https://docs.opencv.org/2.4/doc/tutorials/imgproc/gaussian_median_blur_bilateral_filter/gaussian_median_blur_bilateral_filter.html

- [3] Mohammed Waleed Kalous, Machine recognition of Auslan signs using Power Gloves Towards large-lexicon recognition of sign language.