Sign Language Recognition Using Tensorflow

Report submitted to

ITM University, Raipur, Chhattisgarh

for the partial fulfilment of the award of the degree

of

Bachelors of Technology
in
Cloud Computing and information technology
Submitted by

Name of the Student	Enrollment No
Saransh Nirmalkar	**H0338**
Keshavdeep Yadav	**H0130**

Under the Supervision of
Prof. Dr. Raktim Deb

School of Engineering and Research
ITM University, Uparwara, Naya Raipur,
Raipur, Chhattisgarh-492002

DECLARATION

We **Saransh Nirmalkar and Keshavdeep Yadav** the student of Bachelor of Technology in **Cloud Technology and Information Security.** B.Tech under the School of Engineering and Research in ITM University, Naya Raipur, Chhattisgarh hereby declare that the work contained in this Project report is original and has been done by me under the guidance of my supervisor.

The work has not been submitted to any other University/Institute for any degree or diploma. I have followed the guidelines provided by the University in preparing the dissertation report.

The used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references.

Name of the students	Enrollment No.	Signature
1. Saransh Nirmalkar	H0494	
2. Kesahavdeep Yadav	H0271	

Date:

CERTIFICATE

This is to certify that the Project Report entitled, "**Sign Language Recognition Using Tensorflow**" submitted by "**Saransh Nirmalkar and Keshavdeep Yadav**" to ITM University, Raipur, India, is a record of bonafide Project work carried out by him/her under my/our supervision and guidance and is worthy of consideration for the award of the degree of Bachelor Of Technology in **Cloud Technology and Information Security.**

Signature
Prof. Dr. Raktim Deb
Project Supervisor
Department of Engineering

Date:

Signature

Department Co-ordinator
Department of Engineering
ITM University, Raipur

Forwarded By

Head
School of Engineering and Research
ITM University, Raipur.

PROJECT REPORT APPROVAL CERTIFICATE

This is to certify that dissertation work entitled "**Project Title**" carried out by the Students Name in Computer Science Engineering Department of ITM University Raipur. We hereby accepted and approved after proper evaluation as a creditable work submitted in partial fulfilment of the requirement for the award of the Degree, Bachelor of Technology in Computer Science Engineering at ITM University Raipur.

Objective of this Project report is **satisfactory / unsatisfactory** for the partial fulfilment of the requirement for the award of the Degree, **Bachelor of Technology** in **Computer Science Engineering**.

Internal Examiner: External Examiner

Name: Name:

Signature: Signature:

Date:_____

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to all those people who have given their heart willing support in making this completion a magnificent experience.

I am thankful to **Dr. S.P Makhija**, Head, School of Engineering and Research, ITM University Raipur, for providing us good and healthy environment for the preparation of this Project.

I am also thankful to my Project Supervisor **Prof. Dr. Raktim Deb,** who was in the Computer Science Engineering Department, for his timely comments and suggestions. He advised on the details of my work and provides valuable discussions. Without the guidance of my supervisor, this Project may not have well materialized.

I am overwhelmed by the constant support and needful motivation given by the Computer Science Engineering Department Faculty.

I am really grateful to my parents for their support, appreciation and encouragement. I specially acknowledge the authors of different research papers and books for providing necessary contribution to my work.

ABSTRACT

Sign language is the only tool of communication for the person who is not able to speak and hear anything. Sign language is a boon for the physically challenged people to express their thoughts and emotion. In this work, a novel scheme of sign language recognition has been proposed for identifying the alphabets and gestures in sign language. With the help of computer vision and neural networks we can detect the sign and give the respective text output.

CHAPTER 1

INTRODUCTION

Sign Languages are the native languages of the Deaf and their main medium of communication. As visual languages, they utilize multiple complementary channels to convey information. This includes manual features, such as hand shape, movement and pose as well as non-manuals features, such as facial expression, mouth and movement of the head, shoulders and torso.

The goal of this project is to show, how we use our technology to help Deaf by labelling the handsigns with their respective messages using Object Detection API Tensorflow, which can be read by those who don't know sign language. It can help to understand deaf without any need to have knowledge of Sign Language. This project is just a small step, there are many things which can be added to this project for later period.

1.1 IMAGE PROCESSING

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

Digital image processing consists of the manipulation of images using digital computers. Its use has been increasing exponentially in the last decades. Its applications range from medicine to entertainment, passing by geological processing and remote sensing. Multimedia systems, one of the pillars of the modern information society, rely heavily on digital image processing.

Pattern recognition, On the basis of image processing, it is necessary to separate objects from images by pattern recognition technology, then to identify and classify these objects through technologies provided by statistical decision theory. Under the conditions that an image includes several objects, the pattern recognition consists of three phases.

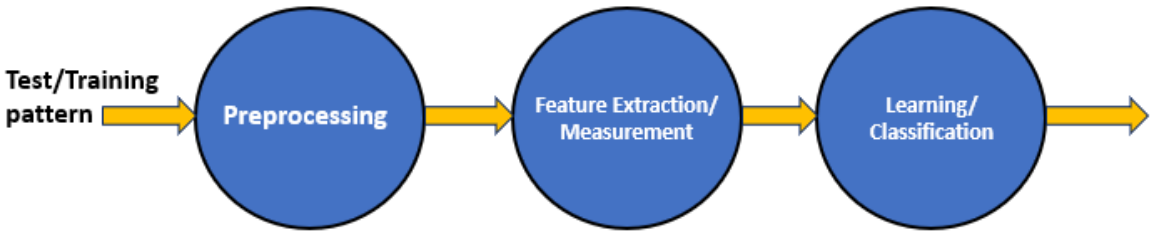


Fig.1.1Phases of pattern recognition.

1.2 SIGN LANGUAGE AND HAND GESTURE RECOGNITION

The process of converting the signs and gestures shown by the user into text is called sign language recognition. It bridges the communication gap between people who cannot speak and the general public. Image processing algorithms along with neural networks is used to map the gesture to appropriate text in the training data and hence raw images/videos are converted into respective text that can be read and understood.

1.3 Tensorflow:

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications. TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by

using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.

If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition.

Build and train state-of-the-art models without sacrificing speed or performance. TensorFlow gives you the flexibility and control with features like the Keras Functional API and Model Subclassing API for creation of complex topologies. For easy prototyping and fast debugging, use eager execution.

TensorFlow also supports an ecosystem of powerful add-on libraries and models to experiment with, including Ragged Tensors, TensorFlow Probability, Tensor2Tensor and BERT.

Features: TensorFlow provides stable Python (for version 3.7 across all platforms) and C APIs; and without API backwards compatibility guarantee: C++, Go, Java, JavaScript and Swift (early release). Third-party packages are available for C#, Haskell Julia, MATLAB,R, Scala, Rust, OCaml, and Crystal."New language support should be built on top of the C API. However, not all functionality is available in C yet." Some more functionality is provided by the Python API.

Application: Among the applications for which TensorFlow is the foundation, are automated image-captioning software, suchas DeepDream.

1.4 Opencv

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the

commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

OpenCV's application areas include:

1. 2D and 3D feature toolkits
2. Egomotion estimation
3. Facial recognition system
4. Gesture recognition
5. Human–computer interaction (HCI)
6. Mobile robotics
7. Motion understanding
8. Object identification
9. Segmentation and recognition

Stereopsis stereo vision: depth perception from 2 cameras

1. Structure from motion (SFM).
2. Motion tracking
3. Augmented reality

To support some of the above areas, OpenCV includes a statistical machine learning

library that contains:

a. Boosting
b. Decision tree learning
c. Gradient boosting trees
d. Expectation-maximization algorithm
e. k-nearest neighbor algorithm
f. Naive Bayes classifier
g. Artificial neural networks
h. Random forest
i. Support vector machine (SVM)
j. Deep neural networks (DNN)

AForge.NET, a computer vision library for the Common Language Runtime (.NET Framework and Mono).

List of free and open source software packages

a. OpenCV Functionality
b. Image/video I/O, processing, display (core, imgproc, highgui)
c. Object/feature detection (objdetect, features2d, nonfree)
d. Geometry-based monocular or stereo computer vision (calib3d, stitching,

videostab)

e. Computational photography (photo, video, superres)
f. Machine learning & clustering (ml, flann)
g. CUDA acceleration (gpu)

Image-Processing: Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it. If we talk about the basic definition of image processing then “Image processing is the analysis and manipulation of a digitized image, especially in order to improve its quality”

Digital-Image : An image may be defined as a two-dimensional function f(x, y), where x and y are spatial(plane) coordinates, and the amplitude of fat any pair of coordinates (x, y) is called the intensity or grey level of the image at that point. In another word An image is nothing more than a two-dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function f(x, y) at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what colour it should be.

Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image.

Image processing basically includes the following three steps :

- Importing the image
- Analysing and manipulating the image
- Output in which result can be altered image or report that is based on image analysis

Applications of Computer Vision:

Here we have listed down some of major domains where Computer Vision is heavily used :

1. Robotics Application
2. Localization − Determine robot location automatically
3. Navigation
4. Obstacles avoidance
5. Assembly (peg-in-hole, welding, painting)
6. Manipulation (e.g. PUMA robot manipulator)
7. Human Robot Interaction (HRI) − Intelligent robotics to interact with and serve people
8. Medicine Application
9. Classification and detection (e.g. lesion or cells classification and tumor detection)
10. 2D/3D segmentation
11. 3D human organ reconstruction (MRI or ultrasound)
12. Vision-guided robotics surgery
13. Industrial Automation Application
14. Industrial inspection (defect detection)
15. Assembly
16. Barcode and package label reading
17. Object sorting
18. Document understanding (e.g. OCR)
19. Security Application
20. Biometrics (iris, finger print, face recognition)

21. Surveillance − Detecting certain suspicious activities or behaviors
22. Transportation Application
23. Autonomous vehicle
24. Safety, e.g., driver vigilance monitoring

1.5 Keras

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a

Google engineer. Chollet also is the author of the XCeption deep neural network model.

Features: Keras contains numerous implementations of commonly used neuralnetwork building blocks such as layers, objectives, activation functions, optimizers, anda host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

.In addition to standard neural networks, Keras has support for convolutional and recurrent neural networks. It supports other common utility layers like dropout, batch normalization, and pooling.

Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU) principally in conjunction with CUDA.

Keras applications module is used to provide pre-trained model for deep neural networks. Keras models are used for prediction, feature extraction and fine tuning. This chapter explains about Keras applications in detail.

1.6 Numpy

NumPy (pronounced /ˈnʌmpaɪ/ (NUM-py) or sometimes /ˈnʌmpi/ (NUM-pee)) is a library for the Python programming language, adding support for large, multidimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the

competing Numarray into Numeric, with extensive modifications. NumPy is opensource software and has many contributors.

Features: NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

1.7 Neural Network

A neural network is a series of algorithms that endeavors to recognize

underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so the network generates the best possible resultwithout needing to redesign the output criteria. The concept of neural networks, which has its roots in artificial intelligence, is swiftly gaining popularity in the development

of trading systems.A neural network works similarly to the human brain's neural network. A "neuron" in a neural network is a mathematical function that collects and classifies information

according to a specific architecture. The network bears a strong resemblance to statistical methods such as curve fitting and regression analysis.

1.8 Motivation

Not all normal people can understand sign language of impaired people. Our project hence is aimed at converting the sign language gestures into text that is readable for normal people.

1.9 PROBLEM STATEMENT

Speech impaired people use hand signs and gestures to communicate.

Normal people face difficulty in understanding their language. Hence there is a need of a system which recognizes the different signs, gestures and conveys the information to the normal people. It bridges the gap between physically challenged people and normal people.

CHAPTER 2

LITERATURE REVIEW

A Survey of Hand Gesture Recognition Methods in Sign Language Recognition

Sign Language Recognition (SLR) system, which is required to recognize sign languages, has been widely studied for years.The studies are based on various input sensors, gesture segmentation, extraction of features and classifcation methods.This paper aims to analyze and compare the methods employed in the SLR systems, classi cations methods that have been used, and suggests the most promising method for future research. Due to recent advancement in classification methods, many of the recent proposed works mainly contribute on the classifcation methods, such as hybrid method and Deep Learning. This paper focuses on the classifcation methodsused in prior Sign Language Recognition system.

Communication between Deaf-Dumb People and Normal People

Chat applications have become a powerful media that assist people to communicate in different languages with each other. There are lots of chat applications that are used different people in different languages but there are not such a chat application that has facilitate to communicate with sign languages. The developed system is based on

Sinhala Sign language. The system has included four main components as text messages are converted to sign messages, voice messages are converted to sign messages, sign messages are converted to text messages and sign messages Are converted to voice messages. Google

voice recognition API has used to develop speech character recognition for voice messages.The system has been trained for the

speech and text patterns by using some text parameters and signs of Sinhala Sign language is displayed by emoji. Those emoji and signs that are included in this system

will bring the normal people more close to the disabled people. This is a 2 way communication system but it uses pattern of gesture recognition which is not very realiable in getting appropriate output.

Intelligent Sign Language Recognition Using Image Processing

Computer recognition of sign language is an important research problem for enabling communication with hearing impaired people. This project introduces an efficient and fast algorithm for identification of the number of fingers opened in a gesture representing an alphabet of the Binary Sign Language. The system does not require the hand to be perfectly aligned to the camera. The project uses image processing system to identify, especially English alphabetic sign language used by the deaf people to communicate. The basic objective of this project is to develop a computer based intelligent system that will enable dumb people significantly to communicate with all other people using their natural hand gestures. The idea consisted of designing and building up an intelligent system using image processing, machine learning and artificial intelligence concepts to take visual inputs of sign language's hand gestures and generate easily recognizable form of outputs. Hence the objective of this project is to develop an intelligent system which can act as a translator between the sign language and the spoken language dynamically and can make the communication between people with hearing impairment and normal people both effective and efficient.

GESTURE RECOGNITION SYSTEM

Communication plays a crucial part in human life. It encourages a man to pass on his sentiments, feelings and messages by talking, composing or by utilizing some other medium. Gesture based communication is the main method for Communication for the discourse and hearing weakened individuals. Communication via gestures is dialect that utilizations outwardly transmitted motions that consolidates hand signs and development of the hands, arms, lip designs, body developments and outward appearances, rather than utilizing discourse or content, to express the individual's musings. Gestures are the expressive and important body developments that speaks to some message or data. Gestures are the requirement for hearing and discourse hindered, they pass on their message to others just with the assistance of motions. Gesture Recognition System is the capacity of the computer interface to catch, track and perceive the motions and deliver the yield in light of the caught signals. It enables

the clients to interface with machines (HMI) without the any need of mechanical gadgets. There are two sorts of sign recognition methods: image- based and sensorbased strategies. Image based approach is utilized as a part of this project that manages communication via gestures motions to distinguish and track the signs and change over them into the relating discourse and content.

Proposed System

Our proposed system is sign language recognition system using tensorflow which recognizes various hand gestures by capturing video and converting it into frames. Then the hand pixels are segmented and the image it obtained and sent for comparison to the trained model. Thus our system is more robust in getting exact text labels of letters.

Chapter 3

Methodology

3.1TRAINING MODULE:

Supervised machine learning: It is one of the ways of machine learning where the model is trained by input data and expected output data. To create such model, it is necessary to go through the following phases:

1. model construction

2. model training

3. model testing

4. model evaluation

Model construction: It depends on machine learning algorithms. In this projectscase, it was neural networks.Such an agorithm looks like:

1. begin with its object: model = Sequential()

2. then consist of layers with their types: model.add(type_of_layer())

3. after adding a sufficient number of layers the model is compiled. At this moment Keras communicates with TensorFlow for construction of the model. During model compilation it is important to write a loss function and an optimizer algorithm. It looks like: model.comile(loss= 'name_of_loss_function', optimizer= 'name_of_opimazer_alg') The loss function shows the accuracy of each prediction made by the model. Before model training it is important to scale data for their further use.

Model training: After model construction it is time for model training. In this phase, the model is trained using training data and expected output for this data. It's look this way:

model.fit(training_data, expected_output). Progress is visible on the console when the script runs. At the end it will report the final accuracy of the model.

Model Testing: During this phase a second set of data is loaded. This data set has never been seen by the model and therefore it's true accuracy will be verified. After the model training is complete, and it is understood that the model shows the right result, it can be saved by: model.save("name_of_file.h5"). Finally, the saved model can be used in the real world. The name of this phase is model evaluation. This means that the model can be used to evaluate new data.

3.2Preprocessing:

Uniform aspect ratio : *Understanding aspect ratios*: An aspect ratio is a proportional relationship between an image's width and height. Essentially, it describes an image's shape. Aspect ratios are written as a formula of width to height, like this: For example, a square image has an aspect ratio of 1:1, since the height and width are the same. The image could be 500px × 500px, or 1500px × 1500px, and the aspect ratio would still be 1:1. As another example, a portrait-style image might have a ratio of 2:3. With this aspect ratio, the height is 1.5 times longer than the width. So the image could be 500px × 750px, 1500px × 2250px, etc.

Cropping to an aspect ratio :Aside from using built in site style options , you may want to manually crop an image to a certain aspect ratio. For example, if you use product images that have same aspect ratio, they'll all crop the same way on your site.

Option 1 - Crop to a pre-set shape

Use the built-in Image Editor to crop images to a specific shape. After opening the

editor, use the crop tool to choose from preset aspect ratios.

Option 2 - Custom dimensions

To crop images to a custom aspect ratio not offered by our built-in Image Editor, use a third-party editor. Since images don’t need to have the samedimensions to have the same aspect ratio, it’s better to crop them to a specific ratio than to try to match their exact dimensions. For best results, crop the shorter side based on the longer side.

• For instance, if your image is 1500px × 1200px, and you want an aspect ratio of 3:1, crop the shorter side to make the image 1500px × 500px.

• Don't scale up the longer side; this can make your image blurry.

Image scaling:

• In compuer graphics and digital imaging , image scaling refers to the resizing of a digital image. In video technology, the magnification of digital material is known as upscaling or resolution enhancement .

• When scaling a vector graphic image, the graphic primitives that make up the image can be scaled using geometric transformations, with no loss of image quality. When scaling a raster graphics image, a new image with a higher or lower number of pixels must be generated. In the case of decreasing the pixel number (scaling down) this usually results in a visible quality loss. From the standpoint of digital signal processing, the scaling of raster graphics is a two dimensional example of sample-rate conversion, the conversion of a discrete signal from a sampling rate (in this case the local sampling rate) to another.

3.3SEGMENTATION

Image segmentation is the process of partitioning a digital image into multiple segments(sets of pixels, also known as image objects). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyse. Modern image segmentation techniques are powered by deep learning technology.

Why does Image Segmentation even matter?

If we take an example of Autonomous Vehicles, they need sensory input devices like cameras, radar, and lasers to allow the car to perceive the world around it, creating a digital map. Autonomous driving is not even possible without object detection which itself involves image classification/segmentation.

How Image Segmentation works?

Image Segmentation involves converting an image into a collection of regions of pixels that are represented by a mask or a labeled image. By dividing an image into segments, you can process only the important segments of the image instead of processing the entire image.

3.4Testing

The purpose of testing is to discover errors. Testing is a process of trying to discover every conceivable fault or weakness in a work product.

It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner.

Software testing is an important element of the software quality assurance and represents the ultimate review of specification, design and coding. The increasing feasibility of software as a system and the cost associated with the software failures are motivated forces for well planned through testing.

Testing Objectives:

There are several rules that can serve as testing objectives they are:

- Testing is a process of executing program with the intent of finding an error.
- A good test case is the one that has a high probability of finding an undiscovered error.

System Testing:

Involves in house testing of the entire system before delivery to the user. The aim is to satisfy the user the system meets all requirements of the client's specifications. It is conducted by the testing organization if a company has one. Test data may range from and generated to production.

Requires test scheduling to plan and organize:

- Inclusion of changes/fixes.
- Test data to use

One common approach is graduated testing: as system testing progresses and (hopefully) fewer and fewer defects are found, the code is frozen for testing for increasingly longer time periods.

3.5 Use Case Diagram

Use Case during requirement elicitation and analysis to represent the

functionality of the system. Use case describes a function by the system that yields a visible result for an actor. The identification of actors and use cases result in the definitions of the boundary of the system i.e., differentiating the tasks accomplished by the system and the tasks accomplished by its environment. The actors are outside

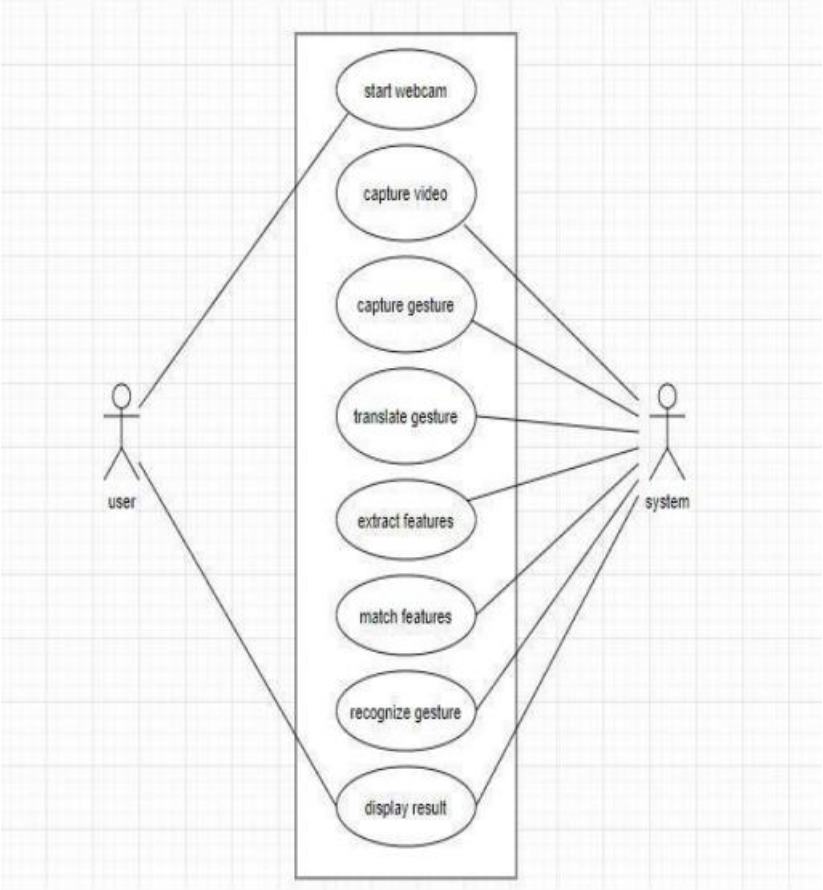
the boundary of the system, whereas the use cases are inside the boundary of the system. Use case describes the behaviour of the system as seen from the actor's point of view. It describes the function provided by the system as a set of events that yield a visible result for the actor.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified. When the initial task is complete, use case diagrams are modelled to present the outside view.

In brief, the purposes of use case diagrams can be said to be as follows –

- Used to gather the requirements of a system.
- Used to get an outside view of a system.
- Identify the external and internal factors influencing the system.
- Show the interaction among the requirements are actors

Fig 3.5 Usecase diagram of sign language recognition System



Chapter 4

RESULTS AND DISCUSSIONS

4.1 SYSTEM CONFIGURATION

Software requirements-

Operating System: Windows,Mac,Linux

SDK: OpenCV, TensorFlow, Keros, Numpy, Jupyter notebook.

Hardware Requirements-

The Hardware Interfaces Required are:

Camera: Good quality,3MP

Ram: Minimum 8GB or higher

GPU: 4GB dedicated

Processor: Intel Pentium 4 or higher

HDD: 10GB or higher

Monitor: 15” or 17” colour monitor

Mouse: Scroll or Optical Mouse or Touch Pad

Keyboard: Standard 110 keys keyboard

4.2 Code

Datacollection.ipynb-

```
import cv2
import os
import time
import uuid
IMAGES_PATH = 'Tenserflow/workspace/images/collectedimages'
labels = ['hello','thanks','yes','no','iloveyou']
number_imgs=15
for label in labels:
    !mkdir {'Tensorflow\workspace\images\collectedimages\\'+label}
    cap = cv2.VideoCapture(0) #0 means the device num of webcam
    print('Collecting images for {}'.format(label))
    time.sleep(5)#time to wait to collect image
    for imgnum in range(number_imgs):
        ret, frame = cap.read()
        imgname = os.path.join(IMAGES_PATH, label, label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
        cv2.imwrite(imgname, frame)
        cv2.imshow('frame',frame)
        time.sleep(2)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
```

main.ipynb-

#setuppath

```
WORKSPACE_PATH = 'Tensorflow/workspace'
SCRIPTS_PATH = 'Tensorflow/scripts'
APIMODEL_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/'
```

#createlabelmap

```
labels = [{'name':'hello', 'id':1},
        {'name':'thanks', 'id':2},
        {'name':'yes', 'id':3},
        {'name':'no', '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')
# 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'}
# Download TF Models Pretrained Models from Tensorflow Model Zoo
```

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


#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}


# 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 = 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/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)

#Train the model

print("""python {}/research/object_detection/model_main_tf2.py --model_dir={}/{} --
pipeline_config_path={}/{}/pipeline.config --
num_train_steps=10000""".format(APIMODEL_PATH,
MODEL_PATH,CUSTOM_MODEL_NAME,MODEL_PATH,CUSTOM_MODEL_NAME))


#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()


@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
```

```
# Detect in Real-Time

import cv2

import numpy as np

category_index =
label_map_util.create_category_index_from_labelmap(ANNOTATION_PATH+'/label_map.p
btxt')

# Setup capture

cap = cv2.VideoCapture(0)

width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))

height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

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
detections = detect_fn(input_tensor)
from matplotlib import pyplot as plt
```

Chapter 5

Application of project work

Sign language recognition systems are developed to facilitate communication between deaf and normals. Recent field of research is intended to focus on effectively recognizing signs under computing power constraints. The proposed sign language recognition system used to recognize sign language letters can be further extended to recognize gestures facial expressions. Instead of displaying letter labels it will be more appropriate to display sentences as more appropriate translation of language.This also increases readability. The work primarily includes recognizing sign languages using discrete cosine transforms, principal component analysis, and hidden Markov models.

Chapter 6

Conclusion

Nowadays, applications need several kinds of images as sources of information for elucidation and analysis. Several features are to be extracted so as to perform various applications. When an image is transformed from one form to another such as digitizing, scanning, and communicating, storing, etc. degradation occurs. Therefore ,the output image has to undertake a process called image enhancement, which contains of a group of methods that seek to develop the visual presence of an image.

Image enhancement is fundamentally enlightening the interpretability or awareness of information in images for human listeners and providing better input for other automatic image processing systems. Image then undergoes feature extraction using

various methods to make the image more readable by the computer.Sign language recognition system is a powerful tool to preparae an expert knowledge, edge detect and the combination of inaccurate information from different sources. the intend of convolution neural network is to get the appropriate classification

Chapter 7

Future scope

The proposed sign language recognition system used to recognize sign language letters can be further extended to recognize gestures facial expressions. Instead of displaying letter labels it will be more appropriate to display sentences as more appropriate translation of language. This also increases readability. The scope of different sign languages can be increased. More training data can be added to detect the letter with more accuracy. This project can further be extended to convert the signs to speech.

References

Base Paper:

[1] http://cs231n.stanford.edu/reports/2016/pdfs/214_Report.pdf

[2] http://www.iosrjen.org/Papers/vol3_issue2%20(part-2)/H03224551.pdf

Other References:

[3]http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.734.8389&rep=rep1&type=pdf

[4]http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.734.8389&rep=rep1&type=pdf

[5] https://ieeexplore.ieee.org/document/7507939

[6] https://www.youtube.com/watch?v=pDXdlXlaCco&t=501s

[7] https://www.youtube.com/watch?v=IOI0o3Cxv9Q&t=0s

[8] https://tensorflow-object-detection-api-tutorial.readthedocs.io/en/latest/install.html