

REAL-TIME SIGN LANGUAGE DETECTION TO TEXT LANGUAGE CONVERSION SYSTEM USING OPENCV AND TENSORFLOW

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

HARRISHKUMAR. L (Reg.No - 40110443)

HASHWANTH KUMAR. H (Reg.No – 40110449)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY – I University by UGC

**Accredited with Grade “A++” by NAAC|12B Status by UGC| Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600119**

APRIL - 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Harrishkumar. L (40110443) Hashwanth Kumar. H (40110449)** who carried out the Project entitled **“REAL-TIME SIGN LANGUAGE DETECTION TO TEXT LANGUAGE CONVERSION SYSTEM USING OPENCV AND TENSORFLOW”** under my supervision from November 2023 to April 2024.

Internal Guide

Ms. K. KARUNYA, M.E,

Head of the Department

Dr. L. LAKSHMANAN, M.E., Ph.D.

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I, **Hashwanth Kumar. H (40110449)** hereby declare that the Project Report entitled “**REAL-TIME SIGN LANGUAGE DETECTION TO TEXT LANGUAGE CONVERSION SYSTEM USING OPENCV AND TENSORFLOW**” done by me under the guidance of **Ms. K. Karunya, M.E**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 12/04/24

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **Sathyabama Institute Science and Technology** for their encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of Computer Science and Engineering for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. K. Karunya, M.E**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Sign language is a vital means of communication for individuals with hearing impairments. The existing Indian Sign Language Recognition systems are designed using machine learning algorithms with single and double-handed gestures but they are not real-time.

To bridge the communication gap between the deaf community and the hearing world, we present a real-time sign language detection system that converts sign language gestures into text language using state-of-the-art technologies, TensorFlow and OpenCV. In this project, we propose a method to create an Indian Sign Language dataset using a webcam and then using the transfer learning train a TensorFlow model to create a real-time Sign Language Recognition system. The system achieves a good level of accuracy even with a limited-size dataset. A real-time sign language detection system is being developed for Indian Sign Language. For data acquisition, detection system is being developed for Indian Sign Language. For data acquisition, images are captured by webcam using Python and OpenCV. The OpenCV provides functions which are primarily aimed at the real-time computer vision. It accelerates the use of machine perception in commercial products and provides a common infrastructure for the computer vision-based applications. The OpenCV library has more than 2,500 efficient computer vision and machine learning algorithms which can be used for face detection and recognition, object identification, classification of human actions, tracking camera and object movements, extracting 3D object models and many more.

OpenCV: is an open-source library for computer vision, machine learning, and image processing.

TensorFlow: is an open-source framework developed by Google to run machine learning, deep learning, and other statistical and predictive workloads.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	v
	LIST OF FIGURES	viii
1	Introduction	1
	1.1 Regression	1
	1.2 Classification	1
	1.3 Classification is preferred to regression	2
	1.4 Face Detection	2
	1.5 Sign language detection	3
2	LITERATURE SURVEY	4
	2.1 Inferences from Literature Survey	5
	2.2 Problems from the sign language	6
3	REQUIREMENT ANALYSIS	7
	3.1 Prerequisite	7
	3.1.1 Python	7
	3.1.2 Anaconda	8
	3.1.3 Visual Studio Code	11
	3.1.4 OpenCV	12
	3.1.5 TensorFlow	13
	3.1.6 Lebellmg	13
	3.1.7 SSD Mobile net V2	13
	3.2 Software requirements specification	14

4	DESCRIPTION OF PROPOSED SYSTEM	15
4.1	Description of the existing model	15
4.2	Description of the proposed model	16
5	IMPLEMENTATION DETAILS	17
5.1	System Design Architecture	17
5.2	Algorithm	17
6	RESULTS AND DISCUSSION	20
6.1	Results	20
6.2	Future Enhancements	20
7	CONCLUSION	21
	REFERENCES	22
	APPENDIX	24
A	SOURCE CODE	24
B	SCREENSHOTS	29
C	RESEARCH PAPER	32
D	CERTIFICATE	38

LIST OF FIGURES

FIGURES NO	TOPIC	PAGE NO
3.2.1	Install OpenCV	14
3.2.2	Install os-sys	14
3.2.3	Install TensorFlow	14
3.2.4	Install numpy	14
4.1	Flowchart of existing model	15
4.2	Flowchart of proposed model	16
4.7	Architecture of the model	17
B.1	Hello command	29
B.2	Yes command	29
B.3	No command	30
B.4	I like you command	30
B.5	Thankyou command	31

CHAPTER 1

INTRODUCTION

In this project, our goal is to create a program using Python to convert sign language to text language using a live video feed. Pre-trained models are being used for sign language detection because the main focus is how to implement the sign language conversion, using OpenCV.

1.1 REGRESSION

Regression is a statistical method used in finance, investing, and other disciplines that attempts to determine the strength and character of the relationship between one dependent variable (usually denoted by Y) and a series of other variables (known as independent variables).

Also called simple regression or ordinary least squares (OLS), linear regression is the most common form of this technique. Linear regression establishes the linear relationship between two variables based on a line of best fit. Linear regression is thus graphically depicted using a straight line with the slope defining how the change in one variable impacts a change in the other. The y-intercept of a linear regression relationship represents the value of one variable when the value of the other is zero.

1.2 CLASSIFICATION

Classification is defined as the process of recognition, understanding, and grouping of objects and ideas into preset categories as known as “sub-populations.” With the help of these pre-categorized training datasets, classification in machine learning programs leverage a wide range of algorithms to classify future datasets into respective and relevant categories.

Classification algorithms used in machine learning utilize input training data for the purpose of predicting the likelihood or probability that the data that follows will fall into one of the predetermined categories. One of the most common applications of classification is for filtering emails into “spam” or “non-spam”, as used by today’s top email service providers.

1.3 CLASSIFICATION IS PREFERRED TO REGRESSION

Technically, there's no reason why you can't treat sign conversion as a regression task. There are even some models that do just that. The problem is that sign language is inherently subjective and based solely on hand gestures.

Communication is very important to every human being. Normal people like us communicate with other people via mouth and body language. But people who are deaf or partially dumb cannot communicate normally through voice. Instead, they communicate via sign language. Each sign language has a unique meaning.

Once you start treating a sign conversion problem as a regression problem, it becomes significantly harder for a model to accurately predict a single value representing that person's image. However, if you treat it as a classification problem, defining buckets/sign brackets for the model, our sign language convertor model becomes easier to train, often yielding substantially higher accuracy than regression-based prediction alone.

In simple words, treating sign language conversion as classification “relaxes” the problem a bit, making it easier to solve — typically, we will get a more accurate meaning for every single hand postures and gestures.

1.4 FACE DETECTION

Face detection is one of the most widely used computer vision applications. It is a fundamental problem in computer vision and pattern recognition. In the last decade, multiple face feature detection methods have been introduced. In recent years, the success of deep learning and convolutional neural networks (CNN) have recently shown great results in powering highly-accurate face detection solutions.

Face detection is a computer technology that determines the location and size of a human face in digital images. Given an image, the goal of facial recognition is to determine whether there are any faces and return the bounding box of each detected face (see object detection). Other objects like trees, buildings, and bodies are ignored in the digital image. Face detection can be regarded as a specific case of

object-class detection, where the task is finding the location and sizes of all objects in an image that belongs to a given class.

1.5 SIGN LANGUAGE DETECTION

A real-time sign language detection system that converts sign language gestures into text language using state-of-the-art technologies, TensorFlow and OpenCV. In this project, we propose a method to create an Indian Sign Language dataset using a webcam and then using the transfer learning train a TensorFlow model to create a real-time Sign Language Recognition system. The system achieves a good level of accuracy even with a limited-size dataset. A real-time sign language detection system is being developed for Indian Sign Language. For data acquisition, images are captured by webcam using Python and OpenCV. This project harnesses the power of computer vision and natural language processing to create a system capable of recognizing and interpreting sign language gestures in real-time. Sign language is an essential communication aid for the hard of hearing. Rather than being real-time, Machine learning is an essential component of today's Indian Sign Language Recognition systems techniques that consider motions made with both one and two hands. We offer an identification system for speaking with signs in real-time that uses TensorFlow and OpenCV, two cutting-edge technologies, to convert gestures from sign language to text in a quest to narrow the communication gap that exists between the hearing and deaf communities. In this project working, we suggest a technique to use a webcam to generate an Indian Sign Language dataset, which can subsequently be utilized in the TensorFlow model training process during the transferring of learning to build an Real time Sign Language Recognition system. Even with a limited amount of data, this approach can achieve a reasonable degree of accuracy.

CHAPTER 2

LITERATURE SURVEY

[1] K. Tiku, J. Maloo, A. Ramesh and I. R & 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2020, pp. 346-351, doi: 10.1109/ICIRCA48905.2020.9182877. Real-time Conversion of Sign Language to Text and Speech an android application is developed that can convert real-time ASL (American Sign Language) signs to text/speech. This paper presents an analysis of the performance of different techniques that have been used for the conversion of sign language to text/speech format.

[2] S. Thakar, S. Shah, B. Shah and A. V. Nimkar & 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT), Bangalore, India, 2022, pp. 1-5, doi: 10.1109/GCAT55367.2022.9971953. Sign Language to Text Conversion in Real Time using Transfer Learning To achieve the former a Convolution Neural Network based VGG16 architecture is used as well as a TensorFlow model for image classification The dataset which has been used is the ASL dataset which has over 87000 images and has been used to train and test the video. There has been an improvement in accuracy from 94% of CNN to 98.7% by Transfer Learning.

[3] S. Sharanyaa, S. V. A. Devi, S. Abinaya, J. K. Sakthi and S. Niranjani & 2022 1st International Conference on Computational Science and Technology (ICCST), CHENNAI, India, 2022, pp. 1-3, doi: 10.1109/ICCST55948.2022.10040293. Real-Time Video To Text Conversion of Sign Language to develop a deep-learning model that can classify live video actions of human sign language such as words and sentences. It is expected that if the LSTM approach is supplemented by the addition of continuous image sequential input, feature extraction methods and correct identification of video types, the success rate of the results produced will increase.

[4] A. Jamwal, G. Vasukidevi, T. N. Malleswari, T. Vijayakumar, L. C. S. Reddy and A. S. A. L. G. G. Gupta & 2022 Sixth International Conference on I-SMAC (IoT in

Social, Mobile, Analytics and Cloud) (I-SMAC), Dharan, Nepal, 2022, pp. 603-609, doi: 10.1109/I-SMAC55078.2022.9987362. Real-Time Conversion of American Sign Language to text with Emotion using Machine Learning This paper mainly focuses on developing a system for recognizing the different hand signs in American Sign Language and their emotions simultaneously in real-time and converting them into text Using the latest technologies like MediaPipe, an improved CNN model is developed based on the distances between each unique identified vital point. The customized model able to achieve 80 percentage of accuracy for the live image.

[5] <https://github.com/nicknochnack/RealTimeObjectDetection> - For Pre-defined models.

[6] <https://github.com/HumanSignal/labelImg> - For labelling image.

2.1 INFERENCES FROM LITERATURE SURVEY

From the above papers, we can analyze the performance of the different techniques that have been used for the conversion of sign language to text / speech format. The dataset that has been used is the ASL dataset which has over 87000 images and has been used to train and test the video. There has been an improvement in accuracy from 94% of CNN to 98.7% by Transfer Learning. It is expected that, if the LSTM approach is supplemented by the addition of continuous image sequential input, feature extraction methods, and correct identification of video types, the success rate of the results produced will increase.

Using the latest technologies like Media Pipe, an improved CNN model is developed based on the distances between each uniquely identified vital point. The customized model was able to achieve 80 percentage of accuracy for the live image.

2.2 PROBLEMS FROM THE SIGN LANGUAGE

Sign language does not use only hand gestures to communicate by also includes facial expressions, hand movements and position, and even the body postures.

Any change in them can change the entire meaning of the sign.

That is why it is generally hard for someone with no knowledge of sign language to understand them.

If the source of the light from the behind of the person who is in the front of the camera is too bright, there is a possibility of misclassification.

Learning a new vocabulary can be challenging in any language, and sign language is no exception. Remembering the signs for different words and concepts can take time and practice.

Sign languages often have their own unique grammar rules that differ from spoken languages. Understanding and applying these rules correctly can be difficult for learners.

Sign language relies heavily on facial expressions, body movements, and spatial referencing to convey meaning. Learners may struggle to incorporate these elements into their signing effectively.

Just like spoken languages, sign languages can vary significantly between different regions or countries. This can pose challenges for learners who may encounter different signs or variations of signs depending on where they are.

Fluent signers can sign at a rapid pace, making it challenging for beginners to keep up with conversations or presentations.

Sign language is not just about learning hand gestures; it also involves understanding the culture and community associated with the language. This includes understanding social norms, etiquette, and cultural nuances.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 PREREQUISITE

To develop real-time sign language detection, the basic concepts of Python, OpenCV, and TensorFlow. So, the major prerequisite is to install OpenCV and all the other pre-trained models to run the program successfully.

3.1.1 Python

Among programmers, Python is a favorite because to its user-friendliness, rich feature set, and versatile applicability. Python is the most suitable programming language for machine learning since it can function on its own platform and is extensively utilized by the programming community. Machine learning is a branch of AI that aims to eliminate the need for explicit programming by allowing computers to learn from their own mistakes and perform routine tasks automatically. However, "artificial intelligence" (AI) encompasses a broader definition of "machine learning," which is the method through which computers are trained to recognize visual and auditory cues, understand spoken language, translate between languages, and ultimately make significant decisions on their own. The desire for intelligent solutions to real-world problems has necessitated the need to develop AI further in order to automate tasks that are arduous to program without AI.

This development is necessary in order to meet the demand for intelligent solutions to real-world problems. Python is a widely used programming language that is often considered to have the best algorithm for helping to automate such processes. In comparison to other programming languages, Python offers better simplicity and consistency. In addition, the existence of an active Python community makes it simple for programmers to talk about ongoing projects and offer suggestions on how to improve.

3.1.2 Anaconda

Anaconda is an open-source package manager for Python and R. It is the most popular platform among data science professionals for running Python and R implementations. There are over 300 libraries in data science, so having a robust distribution system for them is a must for any professional in this field. Anaconda simplifies package deployment and management. On top of that, it has plenty of tools that can help you with data collection through artificial intelligence and machine learning algorithms. With Anaconda, you can easily set up, manage, and share Conda environments. Moreover, you can deploy any required project with a few clicks when you're using Anaconda. There are many advantages to using Anaconda and the following are the most prominent ones among them: Anaconda is free and open-source. This means you can use it without spending any money. In the data science sector, Anaconda is an industry staple. It is open-source too, which has made it widely popular. If you want to become a data science professional, you must know how to use Anaconda for Python because every recruiter expects you to have this skill. It is a must-have for data science.

It has more than 1500 Python and R data science packages, so you don't face any compatibility issues while collaborating with others. For example, suppose your colleague sends you a project which requires packages called A and B but you only have package A. Without having package B, you wouldn't be able to run the project. Anaconda mitigates the chances of such errors. You can easily collaborate on projects without worrying about any compatibility issues. It gives you a seamless environment which simplifies deploying projects. You can deploy any project with just a few clicks and commands while managing the rest. Anaconda has a thriving community of data scientists and machine learning professionals who use it regularly. If you encounter an issue, chances are, the community has already answered the same. On the other hand, you can also ask people in the community about the issues you face there, it's a very helpful community ready to help new learners. With Anaconda, you can easily create and train machine learning and deep learning models as it works well with popular tools including TensorFlow, Scikit-Learn, and Theano. You can create visualizations by using Bokeh, Holoviews, Matplotlib, and Datashader while using Anaconda.

(i) How to Use Anaconda for Python

Now that we have discussed all the basics in our Python Anaconda tutorial, let's discuss some fundamental commands you can use to start using this package manager.

(ii) Listing All Environments

To begin using Anaconda, you'd need to see how many Conda environments are present in your machine.

```
conda env list
```

It will list all the available Conda environments in your machine.

(iii) Creating a New Environment

You can create a new Conda environment by going to the required directory and use this command:

```
conda create -n <your_environment_name>
```

You can replace <your_environment_name> with the name of your environment. After entering this command, conda will ask you if you want to proceed to which you should reply with y:

```
proceed ([y])/n)?
```

On the other hand, if you want to create an environment with a particular version of Python, you should use the following command:

```
conda create -n <your_environment_name> python=3.6
```

Similarly, if you want to create an environment with a particular package, you can use the following command:

```
conda create -n <your_environment_name> pack_name
```

Here, you can replace pack_name with the name of the package you want to use.

If you have a .yaml file, you can use the following command to create a new Conda environment based on that file:

```
conda env create -n <your_environment_name> -f <file_name>.yaml
```

We have also discussed how you can export an existing Conda environment to a .yaml file later in this article.

(iv) Activating an Environment

You can activate a Conda environment by using the following command:

```
conda activate <environment_name>
```

You should activate the environment before you start working on the same. Also, replace the term <environment_name> with the environment name you want to activate. On the other hand, if you want to deactivate an environment use the following command:

```
conda deactivate
```

(v) Installing Packages in an Environment

Now that you have an activated environment, you can install packages into it by using the following command:

```
conda install <pack_name>
```

Replace the term <pack_name> with the name of the package you want to install in your Conda environment while using this command.

(vi) Exporting an Environment Configuration

Suppose you want to share your project with someone else (colleague, friend, etc.). While you can share the directory on Github, it would have many Python packages, making the transfer process very challenging. Instead of that, you can create an environment configuration .yaml file and share it with that person. Now, they can create an environment like your one by using the .yaml file.

For exporting the environment to the .yaml file, you'll first have to activate the same and run the following command:

```
conda env export >> <file_name>.yaml
```

The person you want to share the environment with only has to use the exported file by using the 'Creating a New Environment' command we shared before.

(vii) Removing a Package from an Environment

If you want to uninstall a package from a specific Conda environment, use the following command:

```
conda remove -n <env_name><package_name>
```

On the other hand, if you want to uninstall a package from an activated environment, you'd have to use the following command:

```
conda remove <package_name>
```

(viii) Deleting an Environment

Sometimes, you don't need to add a new environment but remove one. In such cases, you must know how to delete a Conda environment, which you can do so by using the following command:

```
conda env remove --name <env_name>
```

The above command would delete the Conda environment right away.

3.1.3 Visual Studio code

Visual Studio Code (VS Code) offers a range of features that make it a popular choice among developers:

Cross-platform: VS Code runs on Windows, macOS, and Linux, ensuring consistency across different operating systems.

IntelliSense: Provides smart code completion, suggestions, and auto-imports based on the context, language, and libraries being used.

Debugging: Built-in debugging support for multiple languages with breakpoints, call stacks, and an interactive console.

Extensions: A vast marketplace of extensions allows users to customize and extend VS Code's functionality according to their needs, including themes, language support, and productivity tools.

Version Control: Integrated Git support enables version control workflows, including staging changes, committing, and branching directly within the editor.

Terminal Integration: An integrated terminal allows developers to run command-line tools and scripts without leaving the editor.

Task Runner: Provides a task runner interface to automate repetitive tasks, such as building, testing, and deployment.

Workspace Management: Supports the organization of projects into workspaces, facilitating multitasking and collaboration.

Live Share: Collaborative editing and debugging in real-time, enabling developers to work together remotely on the same codebase.

Customization: Extensive customization options, including keybindings, settings, and themes, allowing users to tailor the editor to their preferences and workflow.

These features, along with its performance and community support, contribute to Visual Studio Code's popularity among developers across different domains.

3.1.4 OpenCV

OpenCV (Open-Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting with 2011, OpenCV features GPU acceleration for real-time operations.

It is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects,

faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify an image pattern and its various features we use vector space and perform mathematical operations on these features.

3.1.5 TensorFlow

It is an open-source artificial intelligence package that builds models using data flow graphs. It enables developers to build large-scale neural networks with several layers. TensorFlow is mostly used for classification, perception, comprehension, discovery, prediction, and creation. It is developed by Google to run machine learning, deep learning, and other statistical and predictive workloads. TensorFlow can be used to develop models for various tasks, including natural language processing, image recognition, handwriting recognition, and different computational based simulations such as partial differential equations.

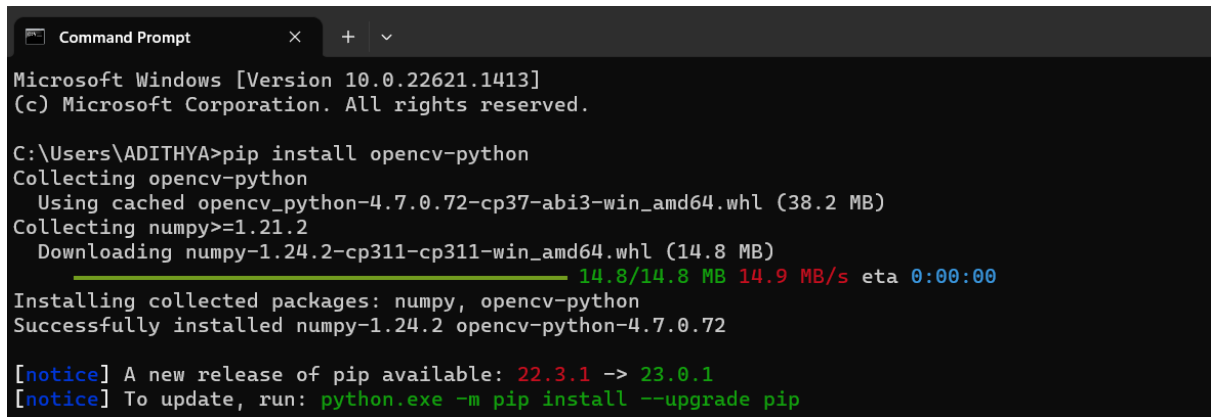
3.1.6 Labellmg

Labellmg is a graphical image annotation tool that labels the bounding boxes of objects in the pictures. With the help of the labeling images, We can Select a particular part of an image and give the exact meaning of that symbol.

3.1.7 SSD Mobile net V2

The Mobile Net SSD model is a single-shot multi box detection (SSD) network that scans the pixels of an image that are inside the bounding box coordinates and class probabilities to conduct object detection. In contrast to standard residual models, the model's architecture is built on the notion of inverted residual structure, in which the residual block's input and output are narrow bottleneck layers. In addition, nonlinearities in intermediate layers are reduced, and lightweight depthwise convolution is applied. The TensorFlow object detection API includes this model.

3.2 SOFTWARE REQUIREMENTS

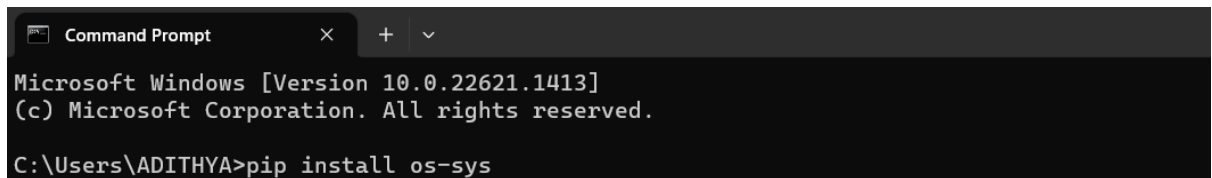


```
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADITHYA>pip install opencv-python
Collecting opencv-python
  Using cached opencv_python-4.7.0.72-cp37-abi3-win_amd64.whl (38.2 MB)
Collecting numpy>=1.21.2
  Downloading numpy-1.24.2-cp311-cp311-win_amd64.whl (14.8 MB)
    14.8/14.8 MB 14.9 MB/s eta 0:00:00
Installing collected packages: numpy, opencv-python
Successfully installed numpy-1.24.2 opencv-python-4.7.0.72

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

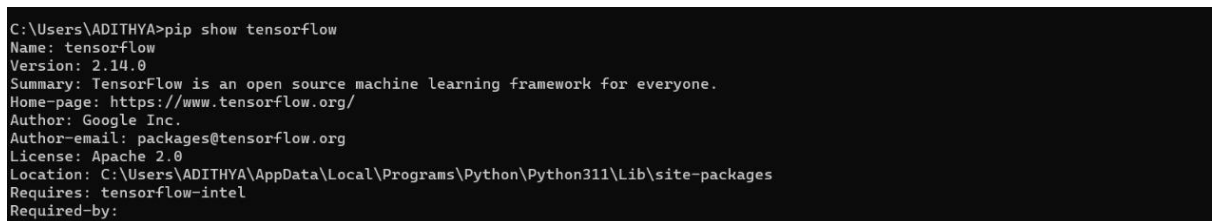
Fig 3.2.1 Install OpenCV



```
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

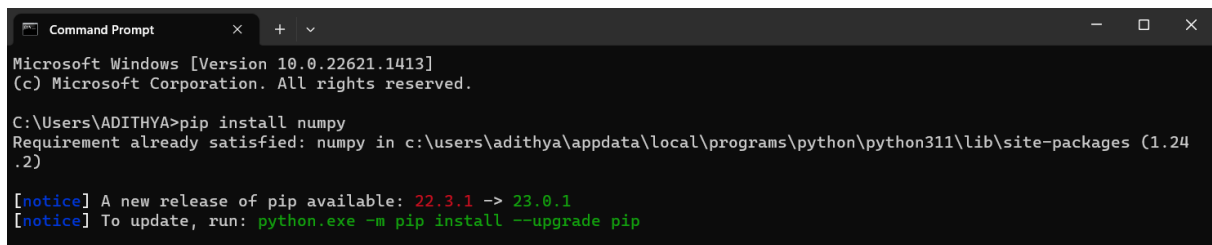
C:\Users\ADITHYA>pip install os-sys
```

Fig 3.2.2 Install os-sys



```
C:\Users\ADITHYA>pip show tensorflow
Name: tensorflow
Version: 2.14.0
Summary: TensorFlow is an open source machine learning framework for everyone.
Home-page: https://www.tensorflow.org/
Author: Google Inc.
Author-email: packages@tensorflow.org
License: Apache 2.0
Location: C:\Users\ADITHYA\AppData\Local\Programs\Python\Python311\Lib\site-packages
Requires: tensorflow-intel
Required-by:
```

Fig 3.2.3 Install TensorFlow



```
Microsoft Windows [Version 10.0.22621.1413]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ADITHYA>pip install numpy
Requirement already satisfied: numpy in c:\users\adithya\appdata\local\programs\python\python311\lib\site-packages (1.24.2)

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip
```

Fig 3.2.4 Install numpy

CHAPTER 4

DESCRIPTION OF THE PROPOSED MODEL

4.1 DESCRIPTION OF THE EXISTING MODEL

In the already existing model, the images of the sign language and text languages are taken already. They are stored in different folders and they are being processed using a geometric approach, an image processing technique to identify different facial expressions based on the distinguished key points on a face. The images are considered in a pixel format and are processed using the Convolutional neural network (CNN) based approach for any kind of feature extraction. The output of CNN will be compared with corresponding ideal values that are stored in a database. In this model, the images are already taken in the folders and files and then it is being retrieved by the system call. So this model cannot take live data that is being viewed on the web camera.

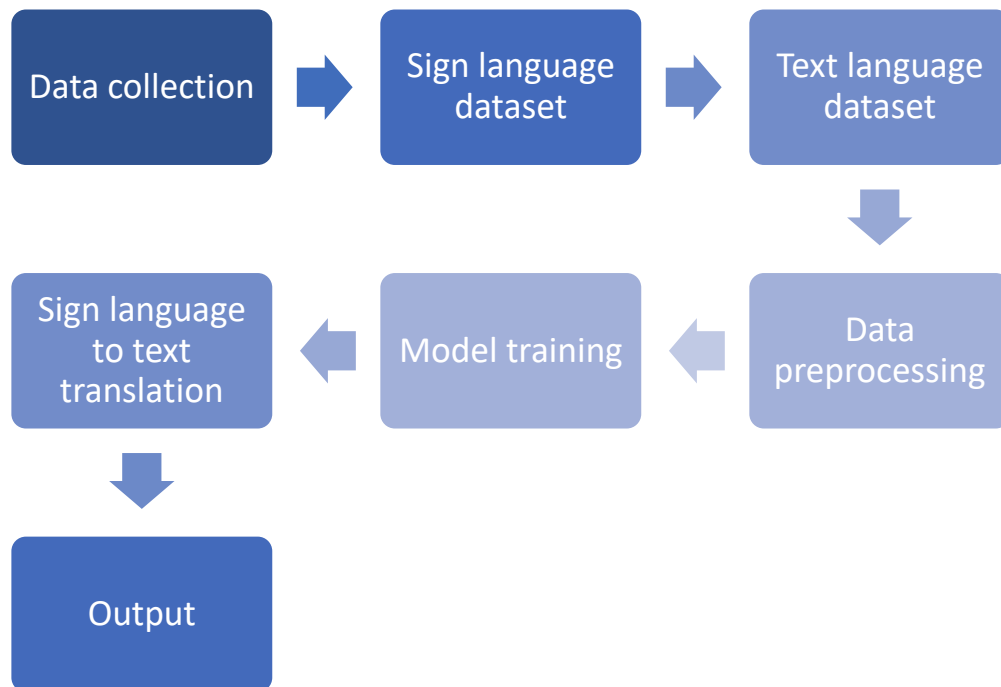


Fig 4.1 Flowchart of the existing model

4.2 DESCRIPTION OF THE PROPOSED MODEL

In this model, the images of the sign language are taken with the help of the program which sends the images to the assigned locations. After that, with the help of the labeling program label the images with their appropriate meaning. Then the dataset is divided into testing sets, training sets, and validation sets. Later, the model is trained by the Tensorflow and Convolutional neural network(CNN) algorithm. Finally, it is being tested on live detection with the help of the web camera. So the drawback of the existing model which is that the images are already taken in the folders and files and then it is retrieved by the system call. So this model cannot take live data that is being viewed on the web camera.

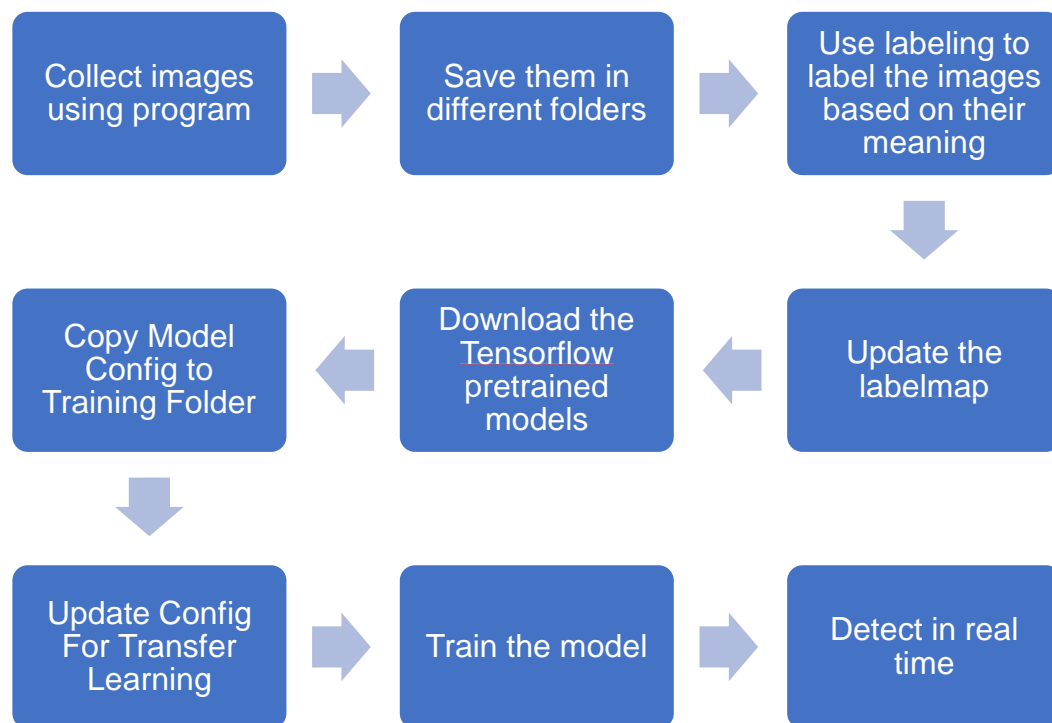


Fig 4.2 Flowchart of the proposed model

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 SYSTEM DESIGN ARCHITECHTURE

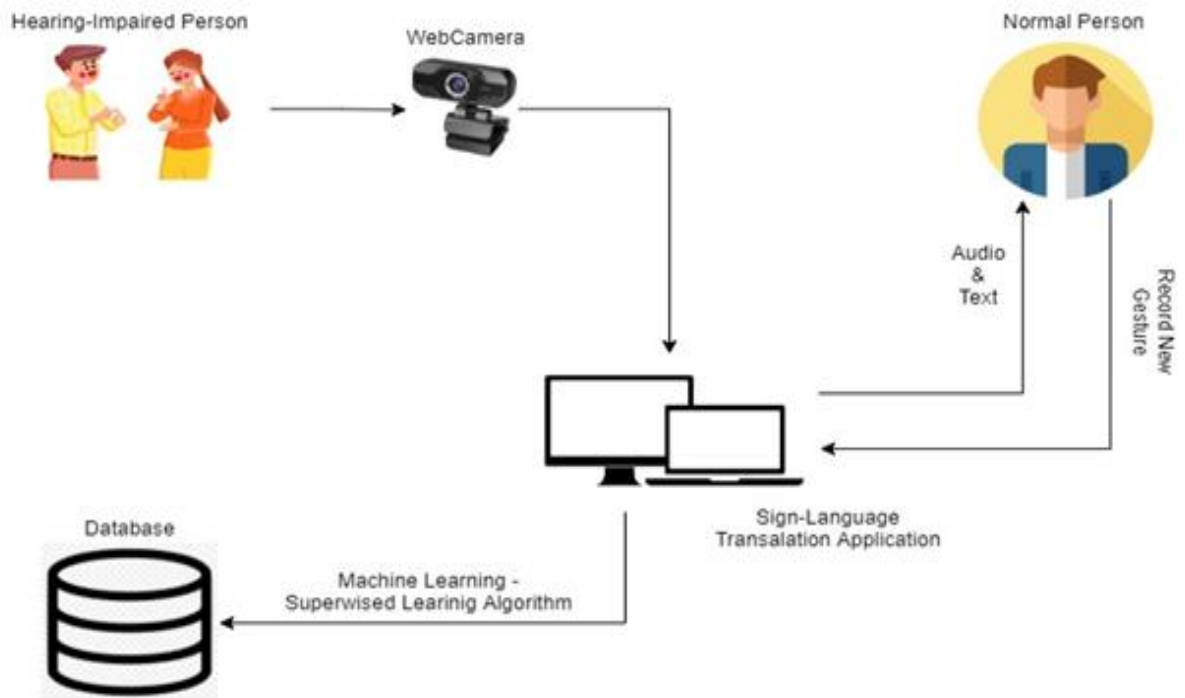


Fig 5.1 Architecture of the model

5.2 ALGORITHM

The term "CNN algorithm" typically refers to Convolutional Neural Networks (CNNs), which are a class of deep neural networks commonly used for image recognition and classification tasks. CNNs are particularly effective in tasks where the input data has a grid-like topology, such as images, audio spectrograms, and time-series data.

There are three layers in a convolutional Neural Network:

Convolutional Layers: CNNs consist of multiple layers, the first being convolutional layers. In these layers, small filters (also known as kernels) slide over the input data,

computing dot products between the filter weights and the input at each position. This operation captures spatial hierarchies by learning patterns such as edges, textures, and more complex features.

Pooling Layers: Pooling layers are typically used to down sample the feature maps obtained from convolutional layers, reducing their spatial dimensions. Common pooling operations include max pooling, which retains the maximum value in each subregion, and average pooling, which computes the average value.

Fully Connected Layers: After several convolutional and pooling layers, the feature maps are flattened into a vector and fed into one or more fully connected (dense) layers. These layers perform classification based on the learned features.

CNNs can automatically learn hierarchical representations of features. Lower layers learn basic features like edges and corners, while higher layers learn complex patterns and shapes. This mimics the hierarchical organization of the human visual system.

There are many advantages on using the Convolutional Neural Networks algorithm:

CNNs use parameter sharing, which means that a single filter is applied across the entire input to produce a feature map. This greatly reduces the number of parameters compared to fully connected networks, making CNNs more efficient and easier to train.

CNNs can detect features regardless of their position in the input image. This property, known as translation invariance, is achieved through the use of shared weights and pooling layers, making CNNs robust to variations in position, scale, and orientation.

By focusing on local connectivity, CNNs are well-suited for capturing spatial patterns in images. Each neuron in a convolutional layer is connected to only a small region of the input volume, allowing CNNs to efficiently capture local dependencies.

CNNs can automatically learn relevant features from raw input data without the need for manual feature extraction. This end-to-end learning approach allows CNNs to adapt to different tasks and datasets, making them highly versatile.

The use of pooling layers and dropout in CNN architectures helps prevent overfitting by reducing the model's sensitivity to small changes in input data. This regularization

technique improves the generalization ability of CNNs, leading to better performance on unseen data.

Pre-trained CNN models, such as those trained on large datasets like ImageNet, can be fine-tuned for specific tasks with smaller datasets. Transfer learning leverages the knowledge learned from one task to improve performance on another task, saving time and computational resources.

CNN operations, especially convolutions and matrix multiplications, can be highly parallelized using modern hardware architectures like GPUs and TPUs. This enables efficient training and inference on large datasets and accelerates computation, making CNNs suitable for real-time applications.

Overall, these advantages make CNNs a powerful tool for a wide range of computer vision tasks, including image classification, object detection, segmentation, and more.

CHAPTER 6

RESULTS AND DISCUSSIONS

6.1 RESULTS

The project is focused on solving the problem of deaf and dumb people. This system will automate the hectic task of recognizing sign language, which is difficult to understand for a normal person, thus it reduces the effort and increases time efficiency and accuracy. Using various concepts and libraries of image processing and fundamental properties of images we trying to develop this system. This paper represented a vision-based system able to interpret hand gestures from the sign language and convert them into text. The proposed system is tested in a real-time scenario, where it was possible to prove that obtained RNN models were able to recognize hand gestures. As future work is to keep improving the system and make experiments with complete language datasets.

6.2 FUTURE ENHANCEMENTS

In the future, the dataset can be enlarged so that the system can recognize more gestures. The TensorFlow model that has been used can be interchanged with another model as well. The system can be implemented for different sign languages by changing the dataset. In this model we are using the Indian sign language meaning, So in the future, we can use some other language such as American or Spanish.

CHAPTER 7

CONCLUSION

In conclusion, the development of a real-time sign language detection to text language conversion system is a technologically challenging but highly impactful endeavor. This system has the potential to facilitate seamless communication for individuals who use sign language, enhancing their accessibility to the broader community. Here are the key takeaways from this project:

- **Complex Multidisciplinary Approach:** Creating this system requires expertise in computer vision, machine learning, natural language processing, and real-time processing. Collaboration between experts in these fields is crucial for success.
- **Data is fundamental.** A diverse and representative dataset of sign language gestures is essential for training the machine learning models. Collaboration with sign language experts and native users is crucial for data collection.
- **Challenges in Variability and Noise:** The system must account for the variability in sign language gestures and mitigate the impact of background noise in real-world environments. Robust algorithms and noise reduction techniques are necessary.
- **Real-Time Processing is Critical:** Achieving low-latency processing is imperative to enable effective, real-time communication. Hardware and software optimizations are essential for real-time operation.
- **User-Centric Design:** The system should be user-friendly, with a well-designed interface. Customization options and accessibility features should be incorporated to cater to a broad user base.
- **Continuous Improvement:** The development of this system should be viewed as an ongoing process. Collecting more data, improving machine learning models, and taking user feedback into account are key for continuous enhancement.

Building a real-time sign language detection to text language conversion system is a noble endeavor with the potential to significantly improve the quality of life for individuals in the Deaf and hard of hearing community.

REFERENCES

- [1] By great learning team, “Real-Time Object Detection Using TensorFlow”, December 25, 2021, <https://www.mygreatlearning.com/blog/object-detection-usingtensorflow/>
- [2] “Computer Vision” <https://www.ibm.com/in-en/topics/computer-vision>
- [3] Dutta, K.K., Bellary, S.A.S.: Machine Learning Techniques for Indian Sign Language Recognition. Int. Conf. Curr. Trends Comput. Electr. Electron. Commun. CTCEEC 2017. 333–336 (2018). <https://doi.org/10.1109/CTCEEC.2017.8454988>
- [4] <https://github.com/HumanSignal/labelImg>
- [5] <https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-practice/?sh=60a27b1b1018>
- [6] <https://github.com/nicknochnack/RealTimeObjectDetection>
- [7] Jamie Berke, James Lacy March 01, 2021 “Hearing loss/deafness| Sign Language” <https://www.verywellhealth.com/sign-language-nonverbal-users-1046848>
- [8] Jeffrey Dean, minute 0:47 / 2:17 from YouTube clip “TensorFlow: Open source machine learning”. Google. 2015. Archived from the original on November 11, 2021. “It is machine learning software being used for various kinds of perceptual and language understanding tasks”
- [9] Joseph Nelson “LabelImg for Labeling Object Detection Data” march 16, 2020 <https://blog.roboflow.com/labelimg>
- [10] Juhi Ekbote, M. Joshi Published 1 March 2017 Computer Science 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS) | DOI:10.1109/ICIIECS.2017.8276111 Corpus ID: 24740741 | Indian sign language recognition using ANN and SVM classifiers
- [11] Kapur, R.: The Types of Communication. MIJ. 6, (2020).

- [12] Konstantinidis, D., Dimitropoulos, K., Daras, P.: Sign language recognition based on hand and body skeletal data. 3DTV-Conference. 2018-June, (2018). <https://doi.org/10.1109/3DTV.2018.8478467>
- [13] “National Health Mission -report of deaf people in India”, nhm.gov.in. 21-12-2021.
- [14] Stephanie Thurrott|November 22 ,2021 “The Best Ways to Communicate with Someone Who Doesn’t Hear Well”
- [15] Suharjito, Anderson, R., Wiryana, F., Ariesta, M.C., Kusuma, G.P.: Sign Language Recognition Application Systems for Deaf-Mute People: A Review Based on InputProcess-Output. Procedia Comput. Sci. 116, 441–448 (2017). <https://doi.org/10.1016/J.PROCS.2017.10.028>

APPENDIX

A. SOURCE CODE:

```
WORKSPACE_PATH = r'Tensorflow\\workspace'
SCRIPTS_PATH = r'Tensorflow\scripts'
APIMODEL_PATH = r'Tensorflow\\models'
ANNOTATION_PATH = WORKSPACE_PATH+r'\\annotations'
IMAGE_PATH = WORKSPACE_PATH+r'\\images'
MODEL_PATH = WORKSPACE_PATH+r'\\models'
PRETRAINED_MODEL_PATH = WORKSPACE_PATH+r'\\pre-trained-models'
CONFIG_PATH = MODEL_PATH+r'\\my_ssd_mobnet\\pipeline.config'
CHECKPOINT_PATH = MODEL_PATH+r'\\my_ssd_mobnet'

labels = [
    {'name':'Hello', 'id':1},
    {'name':'Yes', 'id':2},
    {'name':'No', 'id':3},
    {'name':'Thank you', 'id':4},
    {'name':'I Like you', 'id':5},
]

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

CUSTOM_MODEL_NAME = 'my_ssd_mobnet'

import os
```



```

CUSTOM_MODEL_NAME = "my_ssd_mobnet"

PRETRAINED_MODEL_PATH =
"C:\\Users\\harri\\RealTimeObjectDetection\\Tensorflow\\workspace\\pre-trained-
models\\ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8"

MODEL_PATH =
os.path.join('C:\\Users\\harri\\RealTimeObjectDetection\\Tensorflow\\workspace\\mod
els', CUSTOM_MODEL_NAME)

#Create the directory
os.makedirs(MODEL_PATH, exist_ok=True)

#Define the source and destination file paths
source_config_file = os.path.join(PRETRAINED_MODEL_PATH, 'pipeline.config')
destination_config_file = os.path.join(MODEL_PATH, 'pipeline.config')

#Debugging: Print the paths
print(f'Source: {source_config_file}')
print(f'Destination: {destination_config_file}')

# Copy the file
os.system(f'copy "{source_config_file}" "{destination_config_file}"')

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 = os.path.join(MODEL_PATH, 'pipeline.config')
print(f'Config Path: {CONFIG_PATH}')

try:
    config = config_util.get_configs_from_pipeline_file(CONFIG_PATH)

```

```

    print(f'Config: {config}')
except FileNotFoundError as e:
    print(f"File not found: {e}")
except Exception as e:
    print(f"An error occurred: {e}")

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)

print(r"""python {}\\research\\object_detection\\model_main_tf2.py --model_dir={} --
pipeline_config_path={}\\pipeline.config
--num_train_steps=20000""".format(APIMODEL_PATH,
MODEL_PATH,MODEL_PATH))

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-51')).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

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

```

B. OUTPUT SCREENSHOTS

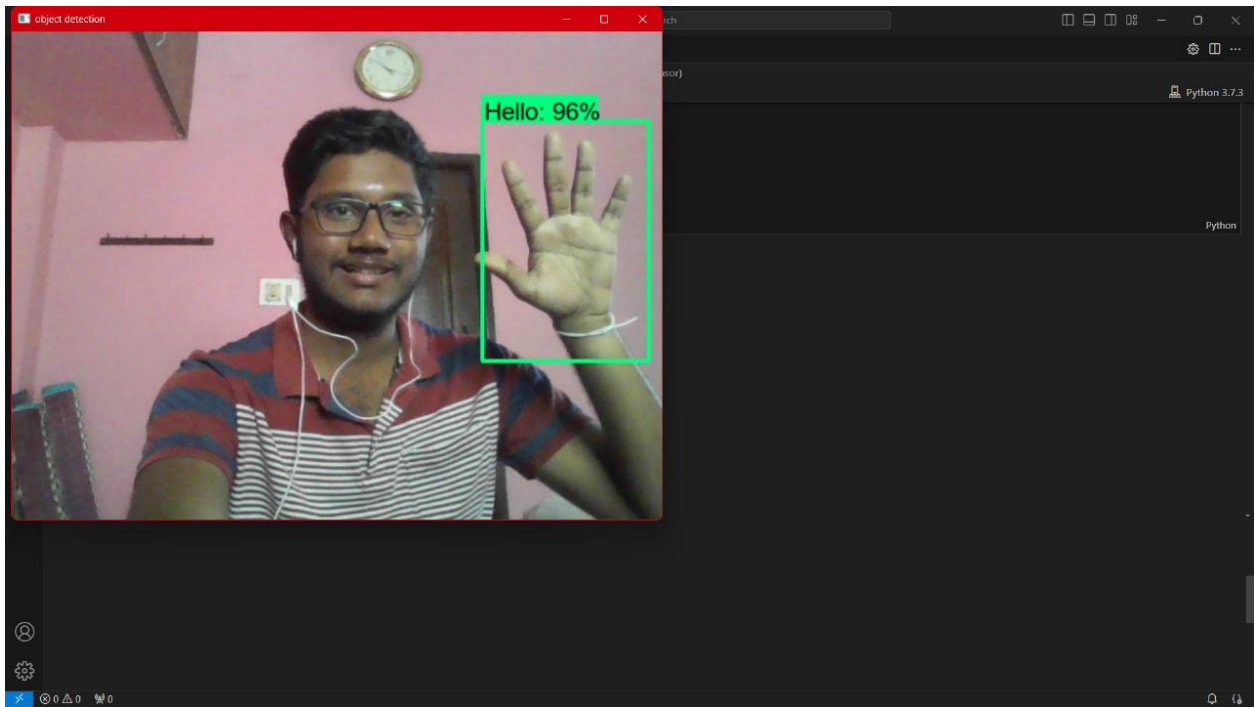


Fig B.1 Hello command

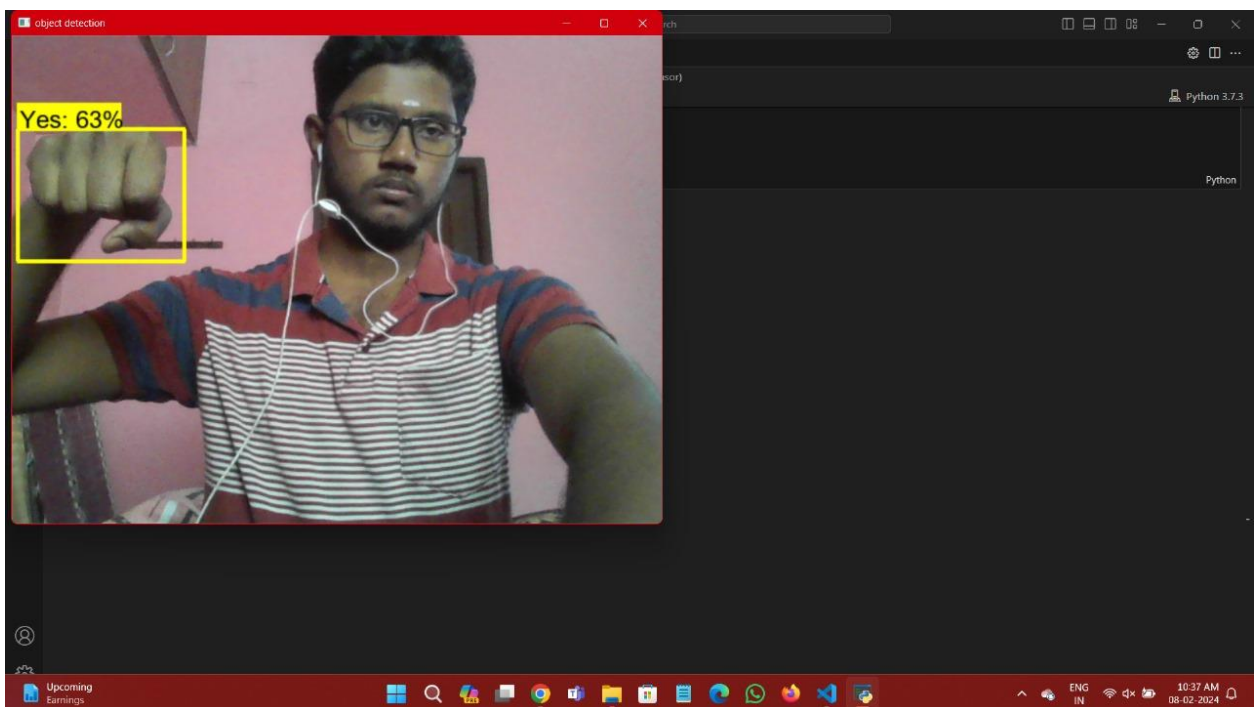


Fig B.2 Yes command

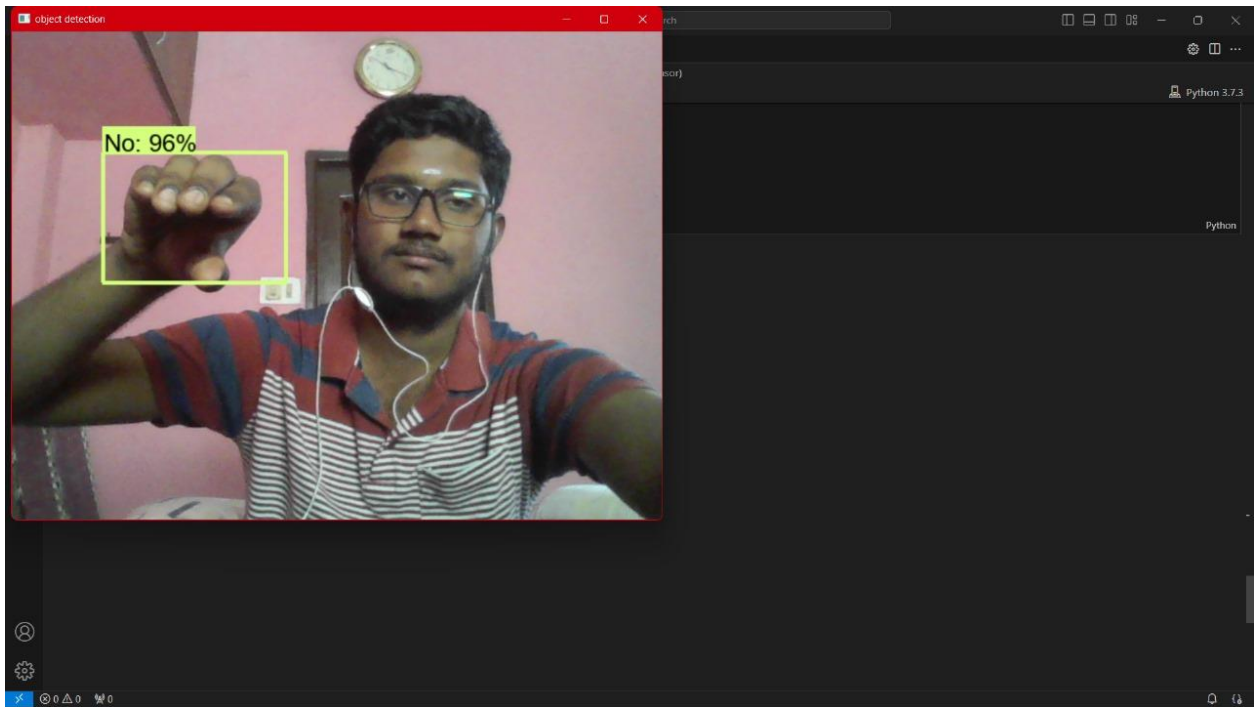


Fig B.3 No command

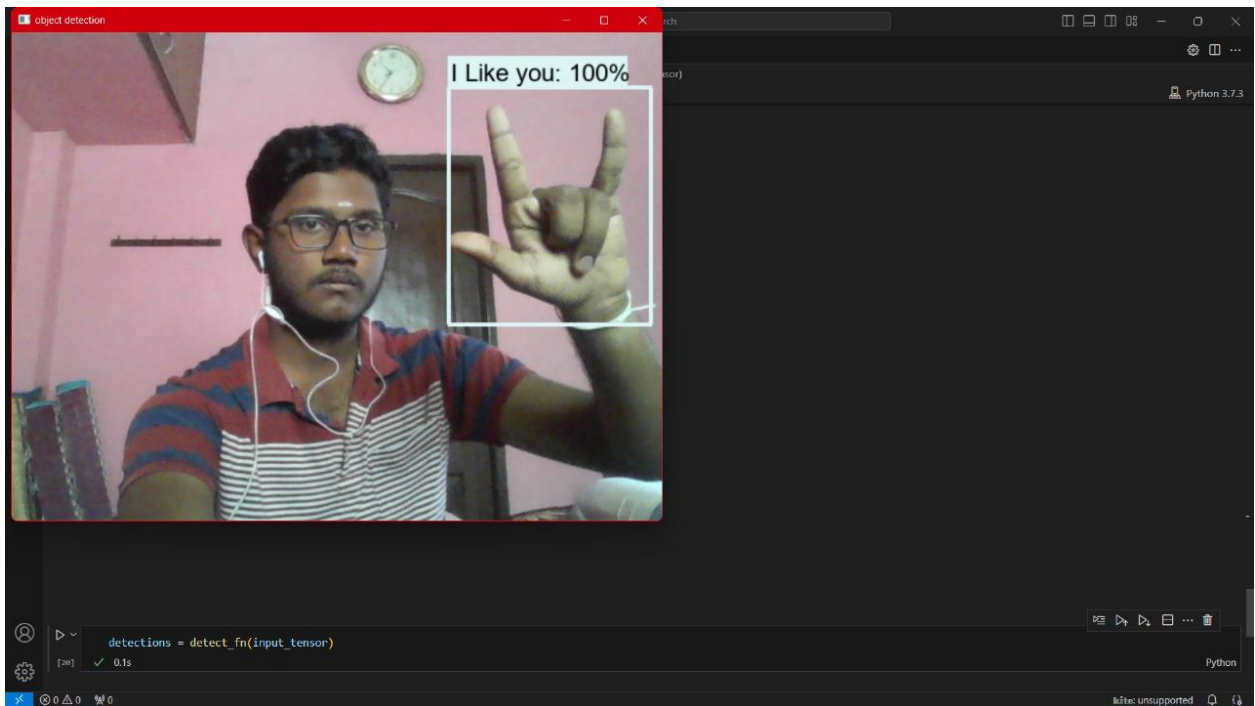


Fig B.4 I like you command

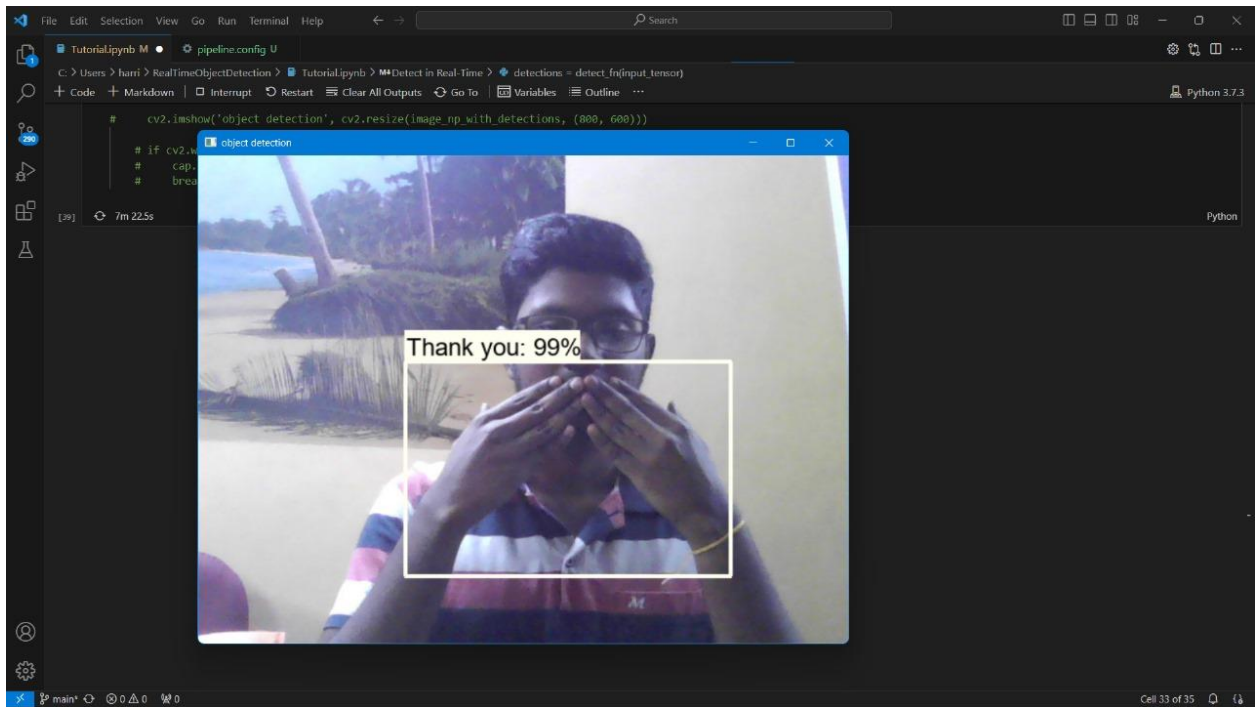


Fig B.5 Thank you command

C. RESEARCH PAPER

REAL-TIME SIGN DETECTION TO TEXT CONVERSION

HarrishKumar L
Department of CSE
Sathyabama Institute of Science
and Technology
Chennai, India
Harrishkumar12020@gmail.com

Hashwanth Kumar H
Department of CSE
Sathyabama Institute of Science
and Technology
Chennai, India
hashwanthprince@gmail.com

Karunya K
Assistant professor,
Department of CSE
Sathyabama Institute of Science
and Technology
Chennai, India
karunsai.ifs@gmail.com

Poornima D
Assistant professor, Department
of CSE
Sathyabama Institute of Science
and Technology
Chennai, India
Poornima.d.cse@sathyabama.ac.in

Gowri Manohari V
Assistant Professor,
Department of CSE
Sathyabama Institute of Science
and Technology
Chennai, India
Gowrimanohari.cse@sathyabama.ac.in

Abstract- Sign language is an essential communication aid for the hard of hearing. Rather than being real-time, Machine learning is an essential component of today's Indian Sign Language Recognition systems techniques that consider motions made with both one and two hands. We offer an identification system for speaking with signs in real-time that uses TensorFlow and OpenCV, two cutting-edge technologies, to convert gestures from sign language to text in a quest to narrow the communication gap that exists between the hearing and deaf communities. In this project working, we suggest a technique to use a webcam to generate an Indian Sign Language dataset, which can subsequently be utilized in the TensorFlow model training process during the transferring of learning to build an Real time Sign Language Recognition system. Even with a limited amount of data, this approach can achieve a reasonable degree of accuracy.

Key phrase – Sign-Language Detection, Hand Gestures, Close the Communication Gap, TensorFlow

I. Introduction

One of the most popular uses of computer vision is face detection. It is a basic among the pattern recognition and computer vision. Several techniques for detecting facial features have been introduced in the past ten years. Convolutional neural networks (CNN) and deep learning have demonstrated remarkable progress in recent years, enabling extremely accurate face identification systems. Computer technology known as "face detection" locates and measures a person's face in digital photos. The objective of facial recognition is to identify faces in a picture and to return the bounding box of each face that is identified (see object detection). In the computer image, other items such as bodies, buildings, and trees are omitted. Finding the locations and sizes of every item in an image that belongs to a particular class is the problem of object-class detection, which may be thought of as a special instance of face detection.

II. Proposed system

A real-time sign language identification system that uses TensorFlow and OpenCV, two cutting-edge technologies, to translate sign language motions into text. In this research, we propose a technique to utilize

a camera to construct an Indian sign-language database, after which a TensorFlow model is trained with transfer learning to develop a real-world system for interpreting language understanding. The algorithm can get a respectable degree of accuracy even with a little dataset. A real-time recognition system based on Indian Sign Language is being designed. Webcam feeds are gathered for gathering data utilizing OpenCV and Python. This project combines computer vision and natural language processing to develop a system that is capable of deciphering and understanding real-world sign language scenarios.

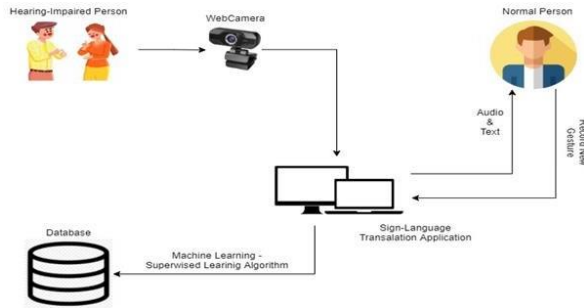


Fig 2.1 System Architecture

A. Data acquisition

There are several methods for obtaining data, but the most popular ones involve the use of sensory devices and vision-based approaches.

1) By the use of the sensory devices

It can offer information on our hand motions with the aid of other IOT devices and electromagnetic sensors. Data gloves will be used to do this. They are not very user-friendly and rather pricey.

2) By vision-based approach

The computer camera is an input device in this strategy. Therefore, additional sensors or data gloves are not needed in order to retrieve the data. As a result, the subject of the camera will become aware that it is an ordinary dialogue with the computer. The primary library utilized in this instance is OpenCV, which stands for Open-source computer vision. The primary difficulty with computer vision is that the system must adjust to the pace of hand motions and ensure that variations in a person's color do not cause confusion in the actions.

B. Data preprocessing

Threshold-based color detection with background removal is required for hand detection using the OpenCV package. OpenCV requires threshold-based color detection with background removal in order to

recognize hands. Every color has a distinct value that is used to set it apart from the others. The OpenCV utilizes a filter called Gaussian blur for this. This filter is used to downscale the image's frames in order to extract as much details as possible. The Gaussian pyramid has three as its default value. We tried utilizing color segmentation algorithms to manually segment each image, but as the study report points out, lighting has an enormous effect on skin tone and hue. As a result, the segmentation we attempted to do did not yield very good results. We also determined that rather than segmenting the hand based on skin color, we would maintain the hand's background as a stable single color. This is because we have plenty of signs to be trained in my project, more of which have similar gestures, such as the "V", "2" symbols. This would enable us to achieve improved outcomes.

C. Gesture classification

A rapid and effective method for identifying static hand motions is used: the Naïve-Bayes classifier. Its foundation is the categorization of various movements based on geometric invariants that are extracted from segmented visual data. The movements are captured with a static background from every frame in the video. The objects of interest must first be divided, labeled, and their geometric invariants must be extracted. After that, a distance weighting strategy and the K nearest neighbor method (KNN) are used to classify the gestures in order to provide relevant information for a selectively loaded naive Bayesian classifier.

III. Methodology

A. Data gathering

This module generates a numerical value for each action done by a person using sign language gesture recognition..

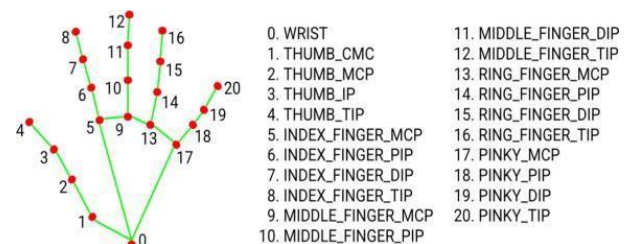


Fig3.1 Module Collecting Hand Gesture

B. Algorithm used

Here, the CNN algorithm is employed. The series of data points is processed by a Convolutional Neural Network algorithm, which then forecasts the sign language motions being made. The CNN system can accurately anticipate the movements being done because a collection of annotated sign language movements served as its training source.

C. Data pre-requisitions

1) Python

Python is popular among programmers due to its versatility, flexibility, and ease of use. Python is the best programming language for machine learning as it is widely used in the programming community and can run on its own platform. The subfield of artificial intelligence called machine learning aims to eliminate the need for precise instructions by enabling computers to learn by making mistakes and repeating tasks. But “machine learning” is the process of teaching computers to detect visual and auditory cues, understand speech, interpret language, and ultimately make important decisions on their own, commonly known as “artificial intelligence” (AI). Accomplishing a process that is difficult to execute without expertise requires further development of expertise because intelligent solutions must be adapted to real situations. Give good answers to the questions the real world needs to evolve in this way. Python is a popular programming language that is considered by many to have the best technology for this type of work. Python is simpler and similar to other programming languages. Additionally, a strong Python community allows employees to easily discuss current projects and offer suggestions for improvements.

2) OpenCV

A collection of projects called OpenCV (Open Source Computer Vision Collection) focuses mainly on real-time computing. Apache 2 is a free and open source cross-platform library. OpenCV has been providing GPU acceleration for real-time tasks since 2011. Its main applications are image processing, video recording and analysis, including object and face recognition. Its main interface is built in C++, but interfaces for Python, Java and MATLAB / OCTAVE are also available.

3) TensorFlow

This open-source AI system uses data streams to build models. Developers can use it to create multi-layered, large-scale neural networks. The main applications of TensorFlow are design, understanding, classification,

detection and prediction. Google created it to perform tasks using machine learning, deep learning, and other statistics and predictions. TensorFlow is a tool that can be used to create models for a range of uses, including partial differential modeling, picture recognition, text recognition, and natural language processing.

4) Labellmg

The drawing tool used to label images is called Labellmg, which identifies the boundary boxes of the picture's objects. Using graphic imagery, we can define specific areas of the image and ensure that characters are clearly identified.

5) Mobile SSD net V2

The mobile net SSD model is an SSD network that does multi-shot detection attempts to identify and classify objects by identifying pixels falling in the joint box. Unlike the old model, the design of this model is founded on the idea of the reversed residue model, in which the residue's input and output are limited processes. Additionally, nonlinearities in the middle layers were minimized by using deep integration. This model is part of the TensorFlow object recognition API.

D. Proposed model testing and performance evaluation

This is an important step in evaluating how well your machine-learning model is working. Measure your model's performance using the experimental methods used to train your system. Performance indicators for categorization tasks include F1 scores, recall, accuracy, and precision. You should also examine the confusion matrix to determine in which class the model is most confusing. In the classification problem, there are four terms such that,

True positives (TP): Positive predictions that come true.

False positives (FP): Positive predictions that turn out to be negative.

True negatives (TN): Negative predictions that end up turning out to be negative.

False negatives (FN): Positive predictions that end up turning out to be negative.

There will always be confusion with these terms. Therefore, it is preferable to display these in a matrix format for improved visibility, which also improves understanding.

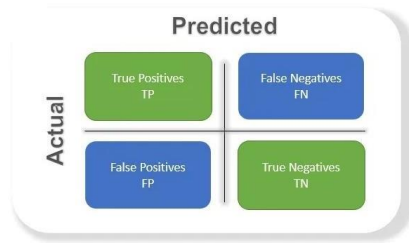


Fig3.2 Confusion Matrix

Accuracy: This tells you the percentage of all samples that the classifier successfully identified, or the overall accuracy of the model. To calculate accuracy, use

$$= (TN + TP) / (FP + TP + FN + TN)$$

Precision: The percentage of positive cases out of every positive outcome that was anticipated to happen. The positive prediction of the model over the entire dataset is used as the denominator in this case. Put it in place of figuring out "the level to which the system is true when it appears to be true."

$$= TP / (FP + TP)$$

Recall: The percentage of successful cases in relation to all actual successful cases. Thus, in this example, the denominator (FN + TP) represents the actual number of positive results in the dataset. Think of it as an effort to figure out "how many more correct ones the model did not display when it displayed the correct ones."

$$= TP / (FN + TP)$$

F1 score: The mean of recall as well as precision is what it is.. Since the F1 score accounts for both contributions, the higher the better. Note that if an F1 score is low due to the product in the numerator, the final score drops significantly. As a result, a model does well in the F1 score if it predicts positives accurately (precision), doesn't miss any positives, and rather predicts negatives (recall).

$$\frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 * precision * recall}{precision + recall}$$

Negative is the opposite and truth is just as important. So depending on our application we will need a higher return and the F1 score will not be a good metric. Therefore, it would be useful to look at the PR or ROC curve based on weighted F1 scores.

IV. Result and conclusion

This module reads the individual's hand gestures and responds in actual time via sign language.

```

Microsoft Windows [Version 10.0.19041.572]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\User>

D:\>cd RealTimeObjectDetection

D:\RealTimeObjectDetection>python Tensorflow/models/research/object_detection/model_main_tf2.py --model_dir=Tensorflow/workspace/models/my_ssd_mobnet --pipeline_config_path=Tensorflow/workspace/models/my_ssd_mobnet/pipeline.config --num_train_steps=10000
2020-11-05 12:10:28.716525: I tensorflow/stream_executor/platform/default/dso_loader.cc:48] Successfully opened dynamic library cudart64_101.dll

```

Fig4.1 Install TensorFlow

```

11185 13:10:02.918599 9044 model_lib_v2.py:652] Step 18500 per-step time 0.127s loss=0.139
INFO:tensorflow:Step 18600 per-step time 0.181s loss=0.113
11186 13:10:14.581546 9044 model_lib_v2.py:652] Step 18600 per-step time 0.181s loss=0.113
INFO:tensorflow:Step 18700 per-step time 0.125s loss=0.115
11187 13:10:26.093042 9044 model_lib_v2.py:652] Step 18700 per-step time 0.125s loss=0.115
INFO:tensorflow:Step 18800 per-step time 0.122s loss=0.093
11188 13:10:37.545224 9044 model_lib_v2.py:652] Step 18800 per-step time 0.122s loss=0.093
INFO:tensorflow:Step 18900 per-step time 0.137s loss=0.103
11189 13:10:49.080930 9044 model_lib_v2.py:652] Step 18900 per-step time 0.137s loss=0.103
INFO:tensorflow:Step 19000 per-step time 0.183s loss=0.112
11190 13:11:00.743386 9044 model_lib_v2.py:652] Step 19000 per-step time 0.183s loss=0.112
INFO:tensorflow:Step 19100 per-step time 0.182s loss=0.139
11191 13:11:12.360541 9044 model_lib_v2.py:652] Step 19100 per-step time 0.182s loss=0.139
INFO:tensorflow:Step 19200 per-step time 0.131s loss=0.221
11192 13:11:24.555228 9044 model_lib_v2.py:652] Step 19200 per-step time 0.131s loss=0.221
INFO:tensorflow:Step 19300 per-step time 0.132s loss=0.184
11193 13:11:36.232512 9044 model_lib_v2.py:652] Step 19300 per-step time 0.132s loss=0.184
INFO:tensorflow:Step 19400 per-step time 0.182s loss=0.128
11194 13:11:47.656452 9044 model_lib_v2.py:652] Step 19400 per-step time 0.182s loss=0.128
INFO:tensorflow:Step 19500 per-step time 0.097s loss=0.117
11195 13:11:59.368521 9044 model_lib_v2.py:652] Step 19500 per-step time 0.097s loss=0.117
INFO:tensorflow:Step 19600 per-step time 0.124s loss=0.117
11196 13:12:10.828460 9044 model_lib_v2.py:652] Step 19600 per-step time 0.124s loss=0.117
INFO:tensorflow:Step 19700 per-step time 0.182s loss=0.119
11197 13:12:22.488073 9044 model_lib_v2.py:652] Step 19700 per-step time 0.182s loss=0.119
INFO:tensorflow:Step 19800 per-step time 0.182s loss=0.118
11198 13:12:34.071634 9044 model_lib_v2.py:652] Step 19800 per-step time 0.182s loss=0.118
INFO:tensorflow:Step 19900 per-step time 0.120s loss=0.113
11199 13:12:45.558648 9044 model_lib_v2.py:652] Step 19900 per-step time 0.120s loss=0.113
INFO:tensorflow:Step 20000 per-step time 0.119s loss=0.099
11200 13:12:57.122153 9044 model_lib_v2.py:652] Step 20000 per-step time 0.119s loss=0.099
D:\RealTimeObjectDetection>

```

Fig4.2 Install Object Detection

In summary, creating a system that converts instant recognition into text is a complex task that can have significant consequences. For language users, this approach leads to better communication and allows them to connect with the larger community. The information will be expanded in the future to keep the system aware of new aspects. It is also possible to use a different model instead of the TensorFlow model. Language translation of the system is possible by modifying the data collection. We are employing the meaning of Indian sign language in our model; however, we might use another language, such as Spanish or French, in the future.

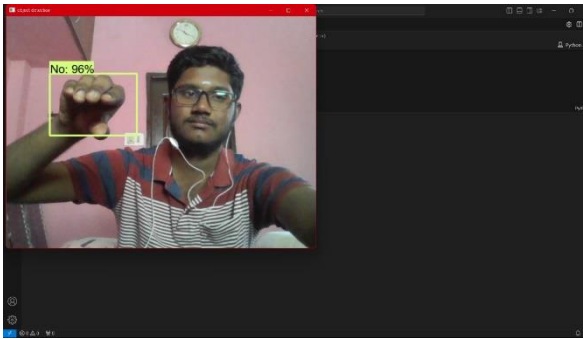


Fig4.3 “No” Sign

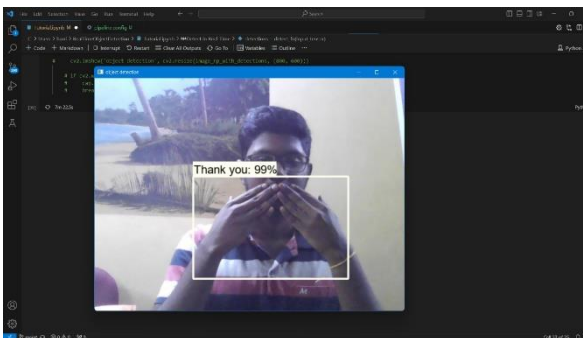


Fig4.4 “Thank You” Sign

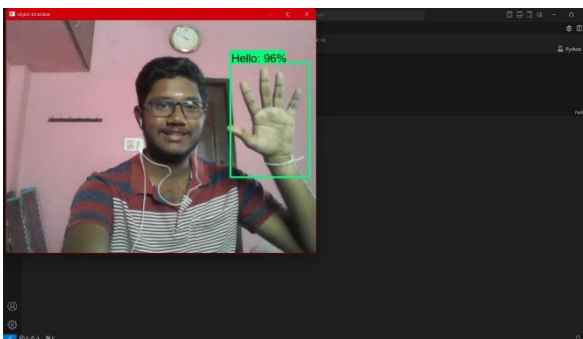


Fig4.5 “Hello” Sign

Reference

- [1] Kapur, R.: The Types of Communication. MIJ. 6, (2020).
- [2] Suharjito, Anderson, R., Wiryana, F., Ariesta, M.C., Kusuma, G.P.: Sign Language Recognition Application Systems for Deaf-Mute People: A Review Based on InputProcess-Output. Procedia Comput. Sci. 116, 441–448 (2017).
<https://doi.org/10.1016/J.PROCS.2017.10.028>.
- [3] Konstantinidis, D., Dimitropoulos, K., Daras, P.: Sign language recognition based on hand and body skeletal data. 3DTV-Conference. 2018-June, (2018).
<https://doi.org/10.1109/3DTV.2018.8478467>.
- [4] Dutta, K.K., Bellary, S.A.S.: Machine Learning Techniques for Indian Sign Language Recognition. Int. Conf. Curr. Trends Comput. Electr. Electron. Commun. CTCEEC 2017. 333–336 (2018).
<https://doi.org/10.1109/CTCEEC.2017.8454988>.
- [5] <https://github.com/nicknochnack/RealTimeObjectDetection>
- [6] <https://github.com/HumanSignal/labelImg>
- [7] Jamie Berke , James Lacy March 01, 2021 “Hearing loss/deafness| Sign Language”
<https://www.verywellhealth.com/sign-language-nonverbal-users-1046848>
- [8] “National Health Mission -report of deaf people in India”, nhm.gov.in. 21-12-2021.
- [9] “Computer Vision” <https://www.ibm.com/en/topics/computer-vision>
- [10] Stephanie Thurrott November 22,2021 “The Best Ways to Communicate with Someone Who Doesn’t Hear Well”
- [11] <https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-practice/?sh=60a27b1b1018>
- [12] Juhi Ekbote, M. Joshi Published 1 March 2017Computer Science 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS) DOI: 10.1109/ICIIECS.2017.8276111 Corpus ID: 24740741 Indian sign language recognition using ANN and SVM classifiers

[13] By great learning team, "Real-Time Object Detection Using TensorFlow", December 25, 2021, <https://www.mygreatlearning.com/blog/object-detection-usingtensorflow/>

[14] Jeffery Dean, minute 0:47/2:17 from YouTube clip "TensorFlow: Open source machine learning". Google 2015. Archived from original on November 11, 2021. "It is a machine learning software being used for various kinds of perceptual and language understanding tasks"

[15] Joseph Nelson "Labelling for labeling object detection data" March 16, 2020
<https://blog.roboflow.com/labelimg>

D. CERTIFICATE

