iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 11 **Issue:** I **Month of publication:** January 2023

**DOI:** https://doi.org/10.22214/ijraset.2023.48586

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 11 Issue I Jan 2023- Available at www.ijraset.com*

# Sign Language Recognition

Shriya Dubey[1], Smrithi Suryawanshi[2], Aditya Rachamalla[3], K. Madhu Babu[4]
*Department of Electronics and Computer Engineering, SNIST, Hyderabad, Telangana, India*

***Abstract:*** ***People communicate using sign language by visually conveying sign patterns to portray purpose. One method of communicating with deaf-mute people is to use sign language mechanisms. One of the nonverbal communication strategies used in sign language is the hand gesture. Many manufacturers all over the world have created various sign language systems, but they are neither adaptable nor cost-effective for end users. We present a design that can recognize various American sign language static hand motions in real-time using transfer learning, Python, and OpenCV in this paper. "Hello, Yes, No, Thank You, and I Love You" are all prevalent sign language terms that our system correctly acknowledges. The following are the key steps in system design; we created our own dataset taking prominent gestures of the American Sign Language, captured images with OpenCV and webcam, the images were then labelled for object detection, training and testing of dataset was done with transfer learning using SSD MobileNet, and eventually the gestures were successfully determined in real-time.***
***Keywords: Sign Language, Python, OpenCV, Transfer Learning***

## I. INTRODUCTION

Sign languages are regarded as a systematic compendium of hand gestures with strong connotations used by hearing impaired individuals to socialize in everyday experiences. The impairments effecting hearing and speeches affect over 360 million people worldwide. Non - verbal communication is used by deaf and dumb people to communicate with one another and with others. Because once you can't talk or listen, it's hard to communicate with one another.

They facilitate communique without the usage of words. Even so, there may be one ultimate problem: only a few humans appear to recognize signal language. Despite the truth that deaf and dumb humans can have interaction the use of signal languages, it's far difficult for all of them to communicate with people who have regular listening to and vice versa main to a scarcity of signal linguistic knowledge. This problem may be constant through imposing a technological solution. Using that solution, you'll fast rework signal language gestures into the extensively spoken language, English.

They use a concurrently and particular mixture of gestures, hand forms, and alignment to express detailed info. The American Sign Language (ASL) system is however one widely used set of languages. Speech-impaired individuals learn sign language so they can converse with others and meet their daily needs.

This is an open cv execution of strategic that makes use of an internet digital digicam to take pics of hand gestures. Image labelling is needed after photo capture, and a pre-educated version SSD Mobile internet v2 is used for signal recognition. Like a consequence, a robust communique path may be fashioned amongst deaf and listening to publics. This machine lets in someone to explicit themselves nonverbally through transferring their palms speedy instead of losing time seeking to shape every phonics for each word.

This reduces the probability of misinterpretation and mistranslation. Similarly, hundreds of phrases that we use on a normal foundation can certainly be delegated through hand gestures and recognized.

## II. PROPOSED ALGORITHM

### *A. Methodology*

This system enables individuals to convey themselves nonverbally by quickly moving their hands instead of losing time trying to form every alphabet for each word. It thus reduces the possibility of confusion and mistranslation. Hundreds of words that we use on a daily basis can also be allotted hand movements and readily identified [1]. We took common words like Yes, No, Thank You, I Love You, and Hello and interpreted them into signs that best represented the word.

The goal of our project is to develop a static sign language detection system based on a vision-based SLR architecture. That is, the system can use OpenCV to ascertain the signer's word signs in real time. As a result, it has better precision and is less expensive. The system's weak point is the inordinate use of computing power during model training with the Tensor flow Object Detection API.

Sign language recognition architecture has been divided into four segments, and they are as follows;

*1)* "Creating our own dataset".
*2)* "Labelling captured images".
*3)* "Training the models using tensor flow object detection".
*4)* "Detecting the hand gesture".[1]

*B. Architecture*

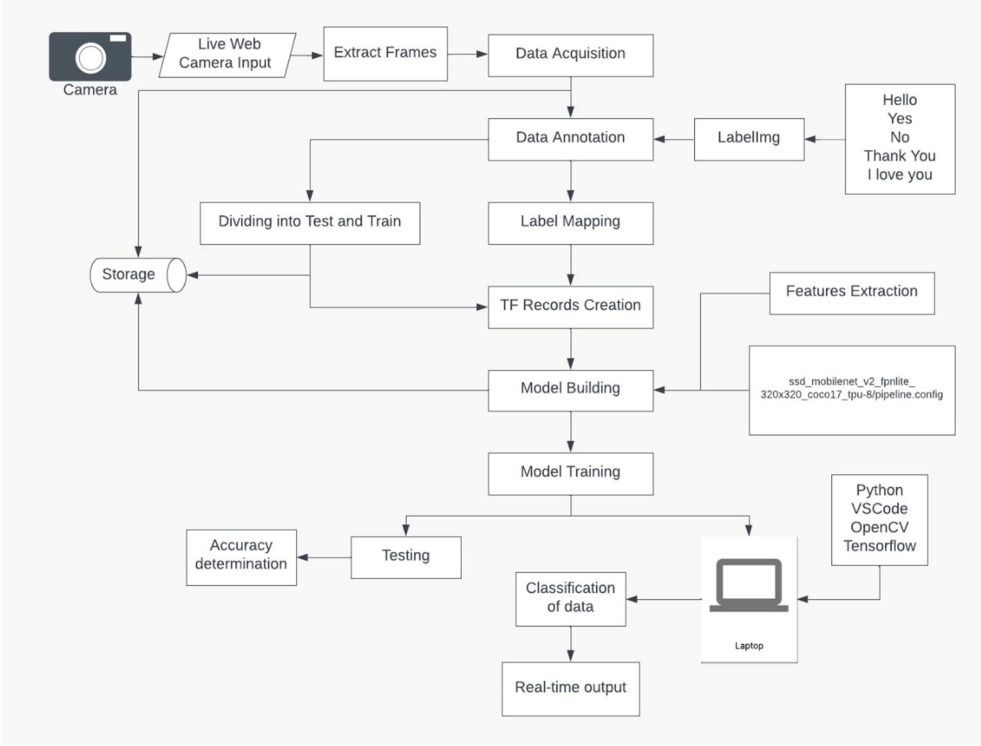



Figure 1. Architecture of Proposed Model

*C. Data Acquisition*

It is necessary to create a proper database of sign language gestures so that the images captured while communicating with this system can be compared. We collected 400 images to create the dataset for this project. This dataset contains five gestures, namely "Hello, Yes, No, I Love You, and Thank You," which are very useful when dealing with real-time applications.

To enhance accuracy, a couple of pix of various signal language gestures have been taken from extraordinary angles and below extraordinary lights conditions. To create our dataset, we used the Open pc vision (OpenCV) library.

Figure 2. Gesture 'Yes'

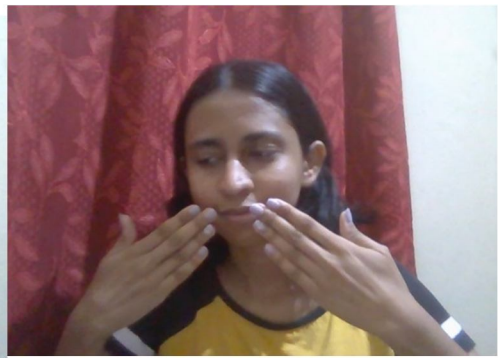


Figure 3. Gesture 'Thank You'

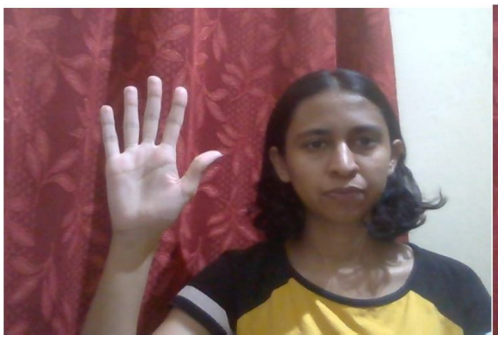


Figure 4. Gesture 'Hello'

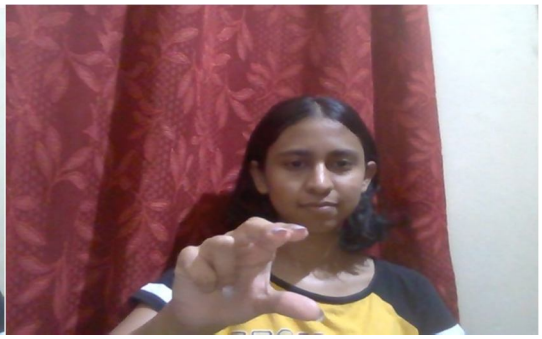


Figure 5. Gesture 'No'


Figure 6. Gesture 'I Love You'

*D. Data Annotation*

Data annotation is the categorization and labelling of data for AI applications. Training data must be properly categorized and annotated for a specific use case. With high-quality, human-powered data annotation, companies can build and improve AI implementations.

LabelImg is a data annotation tool that is used for data annotation. LabelImg is indeed a free and open-source image labelling programme. It's implemented in Python and has a graphical interface built with QT. It's a quick and easy way to label some few hundred images. They assist in identifying elements in your data that you want to train your model to determine in unlabeled datasets. Elevated datasets are required for computer vision and the development of a developing competence model. The garbage in, garbage out philosophy is followed when creating computer vision models, which means labelling images carefully and accurately is critical.

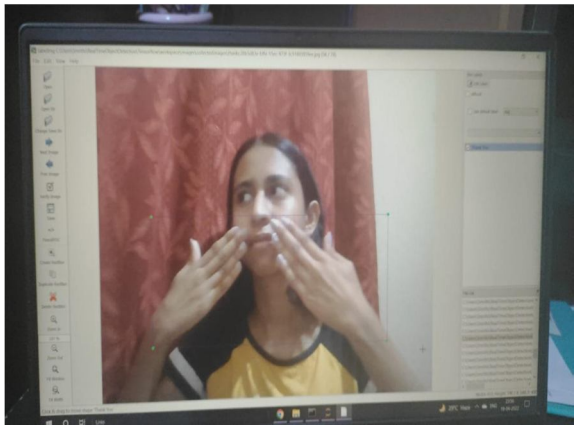


Figure 7. Labelling Gesture 'Thank You'

*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538*
*Volume 11 Issue I Jan 2023- Available at www.ijraset.com*

### *E. Data to Label Mapping*

So, after data collection and annotation, a labelled map is generated as a portrayal of all of the objects in the model, comprising the label of every gesture including its id. The label map contains five labels, each of which represents a gesture. Also, every label has a unique id that ranges from 1 to 5.

Following the completion of the capturing portion, a specific region of the image containing the sign language symbol to be anticipated is chosen. Bounding boxes are enclosed in order for the sign to be detected. These boxes should be tightly packed all around identified part of the image.

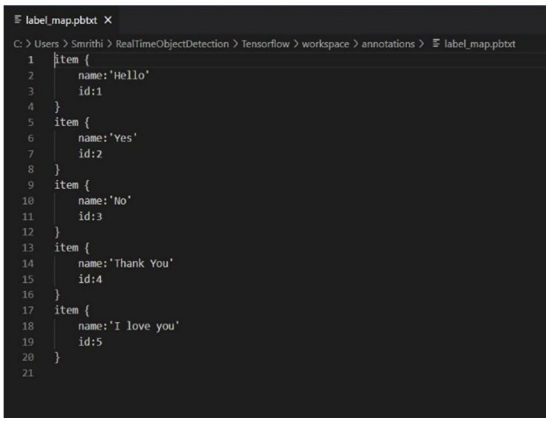



Figure 8. Label Mapping

### *F. Creation of TF Records and Collation of pre-trained Model*

It will be used to take a peek up the class name later. The training and testing data TF records are then generated using generate TFrecord, that would be used to train the TensorFlow object detection API.[33]

The pre-educated TensorFlow version this is getting used is SSD MobileNet v2 320x320. The SSD MobileNet v2 Object detection version is mixed with the FPN-lite function extractor(fig), shared container predictor, and focal loss with schooling photographs scaled to 320x320. Pipeline configuration, i.e., the configuration of the pre-educated version is installation after which up to date for switch gaining knowledge of to teach it via way of means of the created dataset. TensorFlow item detection API, an on hand structure, makes it easy to create, train, and put in force an item detector [2]. The good sized replace that the variety of instances, which became firstly 90, has been decreased to 26, as has the variety of signs (alphabets) on which the version can be educated. After configuring and updating the version, it became educated in ten thousand steps. During education, the hyper-parameter used to set the series of iterations in which the prototype can be obtained education became set up to ten thousand steps.

### *G. Model Building, Updating and Training using Transfer Learning*

The pre-trained model that we have used is named: ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/pipeline.config.

This configuration file helps us build customized pipelines. There is dictionary-like deployment in the file, which is distributed using key-value pairs, and we have to modify these key-value pairs according to how we want the model to perform, the inputs, the outputs and the learning curve we are setting for the model. All of this happens using transfer learning. We transfer all of the properties from a pre-trained model to a new model and make customizations accordingly.

### *H. Testing of the Model*

The training dataset containing 300 images was fed to the model. The test data set consists of 100 images. For these 100 images, we calculate the accuracy with which the model determines a correct class and the accuracy with which the model detects the class correctly.

Class-wise accuracy is the amount by which a class is determined accurately within the image. We calculate the class-wise accuracy by taking the mean of all the accuracies of a particular class, for each class respectively.

*Accuracy = Total number of individual accuracies/ Total number of images*

Individual accuracies are obtained from the ‘det’ tensor in our code. This tensor contains all the information about an image after it is passed through the model and has undergone various kinds of detections.

After a model detects the accuracy of a class in an image, it is stored in the form of a NumPy array in the 'det' tensor. That key of the array is named 'detection_scores' here. The first element of the named array contains the highest accuracy element. This accuracy is plotted against the class in the same tensor. This information is stored as another NumPy array inside another key-value pair, with a key named 'detection_classes.' We take this information to determine the highest accuracy and the consequent mapped class.

When we had a closer look at the data and the prediction scores of our model related to every class, we made an insightful observation. The system captures images with the right label when it has a prediction score of more than 85% against the label for the image. Below which, it is either confused about the label or the image is faulty, either way the detection is not accurate.

Hence, the assumed accuracy of the model determines all the times the model got a prediction score of more than 85% for a particular image. The percentage of times the model has satisfied this condition, is the assumed accuracy of the system. This is mapped by using a truth table and assigning positive-negative values against each prediction score. It helps in determining accuracy that can be safely assumed to be the detection accuracy of the model.

### *I. Real-Time Detection*

This model purposes a pipeline that takes through a web camera from a client who is marking a motion and afterward by extricating various casings of video, it creates communication through signing opportunities for each signal.

The ongoing recognition is finished utilizing OpenCV and webcam once more. For ongoing identification, cv2, and NumPy conditions are utilized. The framework recognizes signs continuously and deciphers what each motion implies into English.

## III. OUTPUTS

### *A. Class-wise Accuracy Detection*

```
Category Index:
{1: {'id': 1, 'name': 'Hello'}, 2: {'id': 2, 'name': 'Yes'}, 3: {'id': 3, 'name': 'No'}, 4: {'id': 4, 'name': 'Thank You'}, 5:
{'id': 5, 'name': 'I love you'}}


Class-wise accuracy of the model

Accuracy of class - hello
0.9751931726932526
Accuracy of class - yes
0.9348790645599365
Accuracy of class - no
0.994537353515625
Accuracy of class - thank you
0.9913764595985413
Accuracy of class - i love you
0.9838629961013794
```

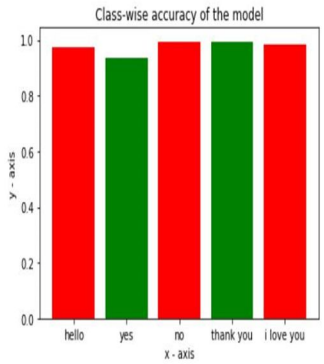



Figure 9. Class-wise Accuracy Result

Matplotlib is used to plot the above result. For better visualization, colors red and green have been chosen. The x-axis represents the 5 classes and the y-axis represents. Highest accuracy achieved is of the class 'Thank You' i.e., 99.13 percent.

*B. Assumed Accuracy of the Model*

In the below graph of assumed accuracy distribution, it depicts that class with truth value more than 85% were considered to give correct identification of label. Hence, for truth value more than 0.85, there is a prediction score of 1.0.

```
Assumed accuracy of the model:
94.73684210526315
```

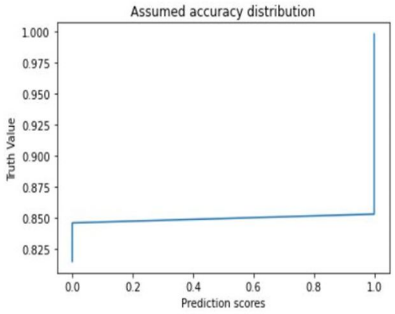



Figure 10. Assumed Accuracy Distribution

*C. Real-time Detection*

The constant location is finished utilizing OpenCV and webcam once more. For, continuous discovery, cv2, and NumPy conditions are utilized. The framework recognizes signs progressively and deciphers what each motion implies into English as displayed:

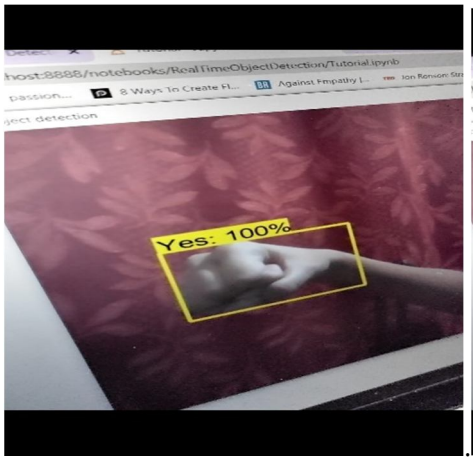



Figure 11. Detection 'Yes'

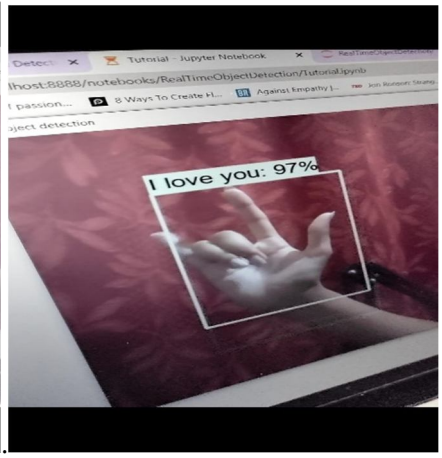



Figure 12. Detection 'I Love You'

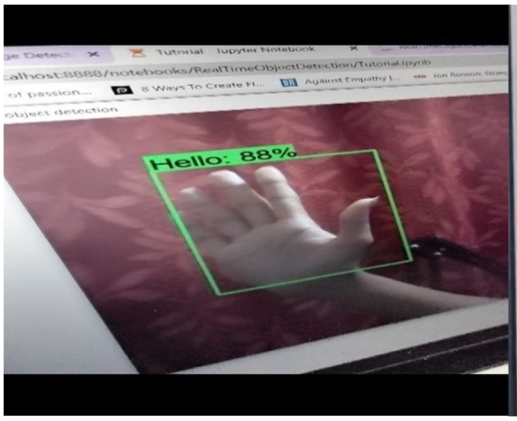



Figure 13. Detection 'Hello'

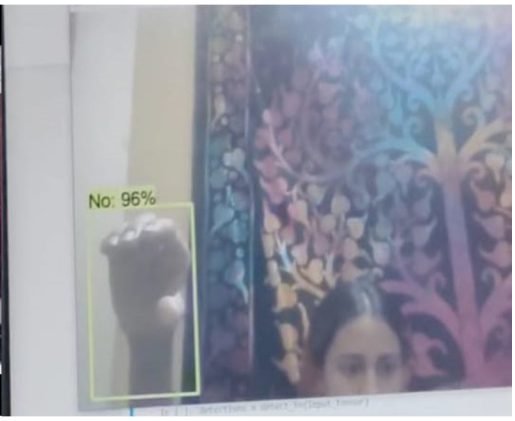



Figure 14. Detection 'No'

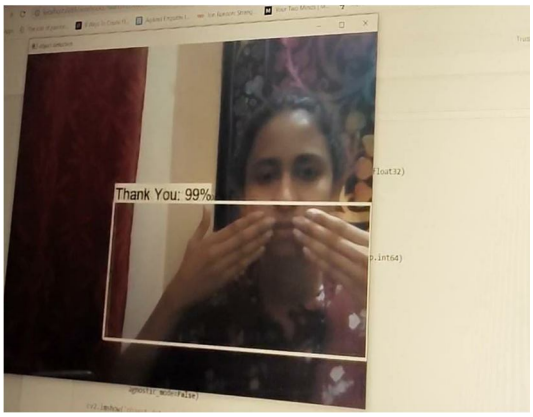



Figure 15. Detection 'Thank You'

For each gesture, the recognition rate is recorded and tabulated in the result as shown:

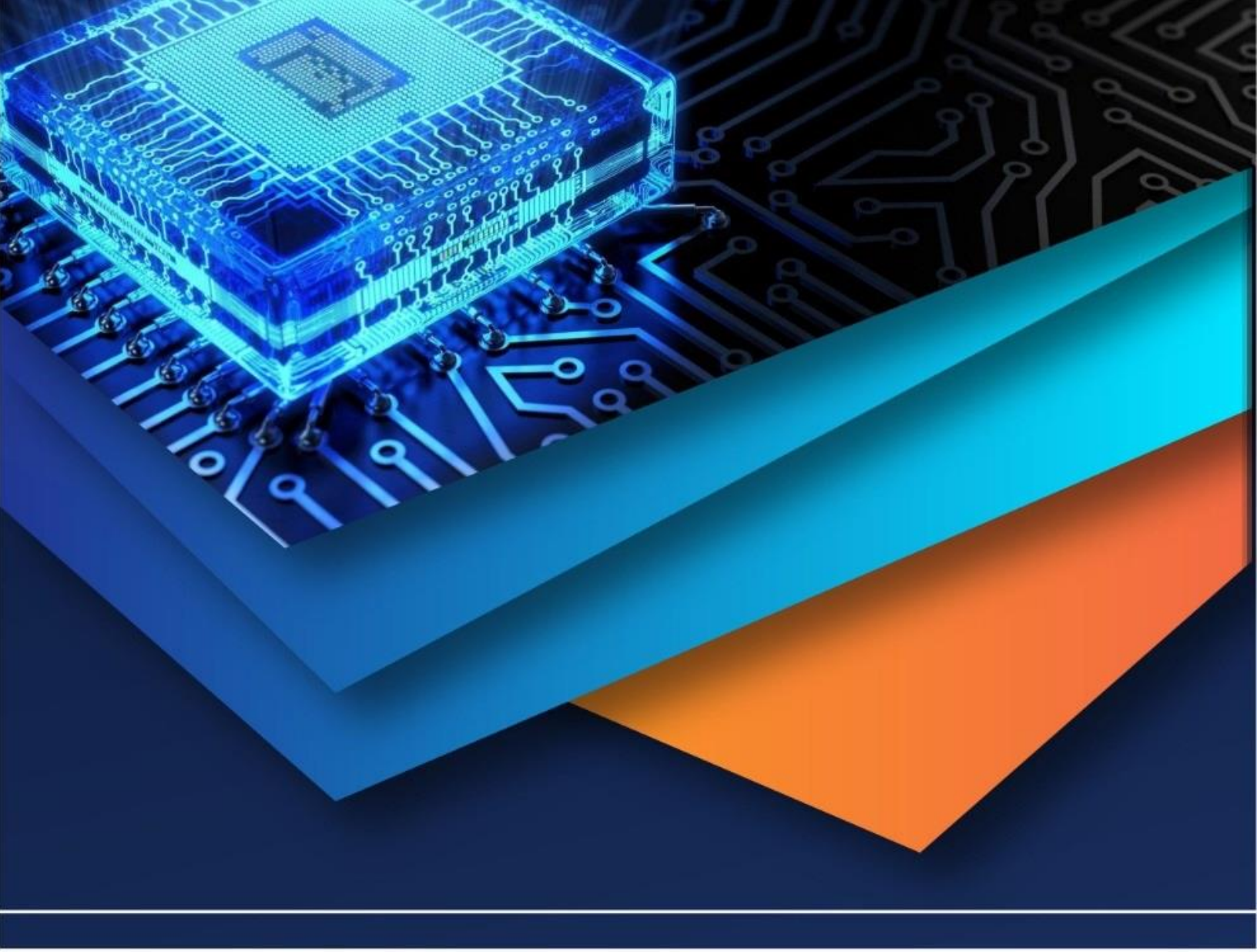
Table 1. Recognition Rate

| S.NO | Labels assigned to hand signs | Recognition rate % |
|---|---|---|
| 1 | Yes | 100 |
| 2 | I Love You | 97 |
| 3 | Hello | 88 |
| 4 | No | 96 |
| 5 | Thank You | 99 |

The 'Yes' sign has the highest recognition of 100 percent. Because of its resemblance to the 'I Love You' hand gesture, the estimation of the 'hello' sign has the least recognition performance of 88 percent.
The project has yielded acceptable results, and the word signs in static position are recognized. The overall output rate was 94.7368 percent.

## REFERENCES

[1] Shivani Y, Jarul R, Vandana N, "Sign Language Recognition System using Transfer Learning Technique", Artificial & Computational Intelligence, December-2021.
[2] Sharvani Srivastava, Amisha Gangwar, Richa Mishra & Sudhakar Singh, "Sign Language Recognition System Using TensorFlow Object Detection API", CCIS Vol.1534, Feburary-2022.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY