RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES
(catering the Educational Needs of Gifted Rural Youth of AP)

R.K Valley,Vempalli(M), Kadapa(Dist) – 516330

A Project Report

On

Real Time Sign Language Detection

Submitted To

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE AND TECHNOLOGIES
RK VALLEY

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

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

Submitted by

M CHANDINI(R180009)
M CHARITHA(R180247)

Under the Guidance of

MS. V SRAVANI

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Rajiv Gandhi University of Knowledge Technologies

RK Valley, Kadapa (Dist), Andhra Pradesh, 516330

CERTIFICATE

This is to certify that the project work titled **"Real Time Sign Language Detection"** is a bonafide project work submitted by **M CHANDINI -R180009**,**M CHARITHA – R180247** in the department of **COMPUTER SCIENCE AND ENGINEERING** in partial fulfillment of requirements for the award of degree of Bachelor of Technology in **Computer Science and Engineering** for the year 2023-2024 carried out the work under the supervision.

GUIDE
Ms. V SRAVANI
Guest Faculty
RK Valley RGUKT

HEAD OF THE DEPARTMENT
N. SATYANANDARAM
PGDIT, MSIT
IIIT, Hyderabad.

Submitted for the Practical Examination held on_____

Internal Examiner **External Examiner**

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

A.P.Government Act 18 of 2008)

RGUKT-RK Valley,kadapa, Andhrapradesh-516330

DECLARATION

We,M.charitha(R180247),M.Chandini(R180009),hereby declare that the project report entitled “Real Time Sign Language Detection” done under guidance of Ms.V.Sravani is submitted in partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering during the academic session 2023-2024 at RGUKT-RK Valley. I also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites are mentioned in the references. To the best of my knowledge, the results embodied in this dissertation work have not been submitted to any university or institute for the award of any degree or diploma.

Date : M.Charitha(R180247)

Place: RGUKT RK VALLEY M.Chandini(R180009)

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success.

I am extremely grateful to our respected Director, Prof. A V S S KUMARA SWAMI GUPTA for fostering an excellent academic climate in our institution.

I also express my sincere gratitude to our respected Head of the Department Mr. N. SATHYANANDARAM for his encouragement, overall guidance in viewing this project a good asset and effort in bringing out this project.

I would like to convey thanks to our guide at college Ms V SRAVANI for his guidance, encouragement, co-operation and kindness during the entire duration of the course and academics.

My sincere thanks to all the members who helped me directly and indirectly in the completion of project work. I express my profound gratitude to all our friends and family members for their encouragement.

TABLE OF CONTENTS

Abstract

The Real-Time Sign Language Detection Project addresses the communication challenges faced by the deaf and hard-of-hearing community by developing an advanced system capable of recognizing and interpreting sign language gestures in real-time. Leveraging cutting-edge technologies such as computer vision and machine learning, the project focuses on creating a robust and efficient solution that facilitates seamless communication between sign language users and non-signers.The core components of the project include gesture recognition, hand tracking, and natural language processing. Deep learning models are employed for accurate and real-time recognition of a diverse set of sign language gestures. Computer vision techniques ensure precise hand tracking, adapting to various lighting conditions and hand orientations. The integration of natural language processing enhances the interpretability of the system by converting recognized signs into text or speech.

LIST OF FIGURES

1.INTRODUCTION

The Real-Time Sign Language Detection System project addresses the communication challenges faced by the deaf and hard-of-hearing community by developing a robust system capable of recognizing and interpreting sign language gestures in real-time. This report outlines the project's objectives, methodology, key components, implementation details, and future prospects.

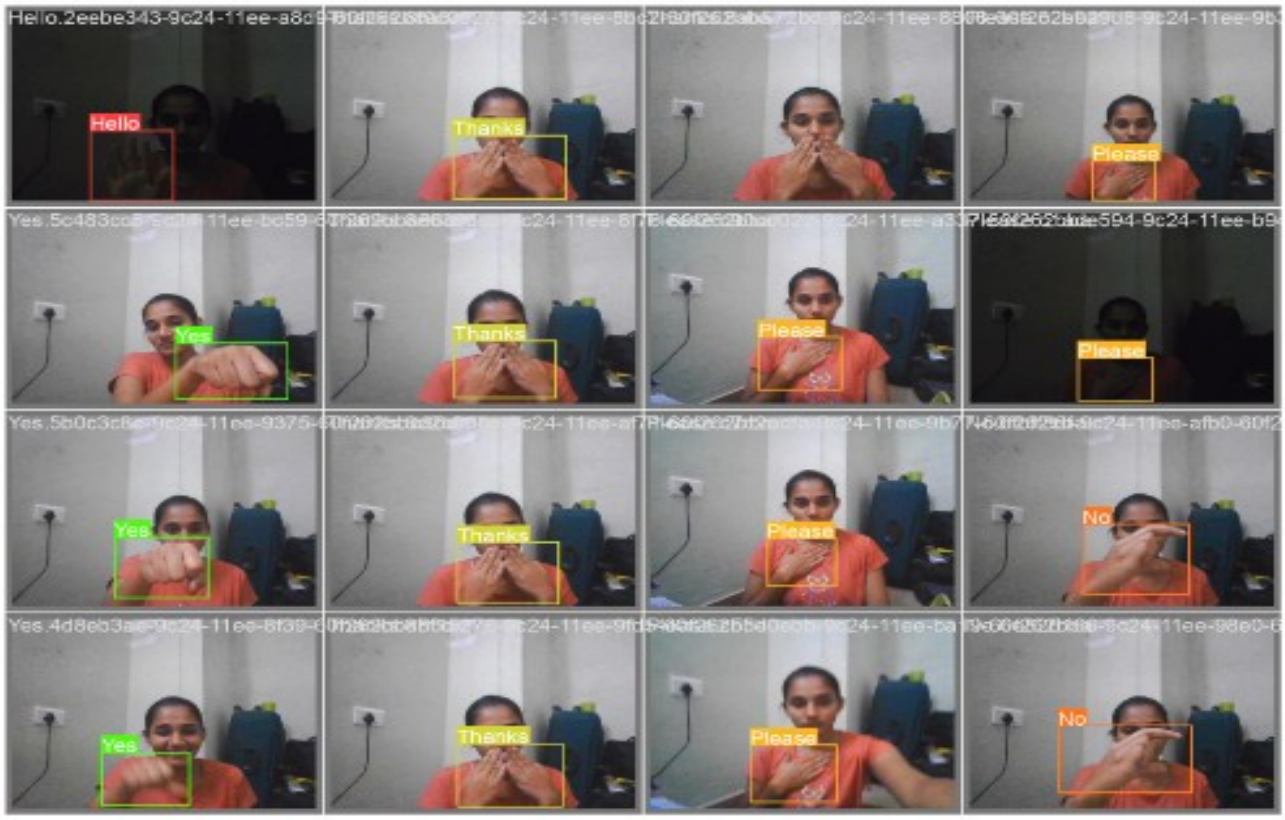

Fig:1.1

1.1.Background:

Sign language serves as a primary means of communication for millions of individuals globally. This project aims to leverage technology to bridge the communication gap between sign language users and non-signers in real-time

1.2.Objectives:

Develop a real-time sign language detection system.

Implement accurate gesture recognition and hand tracking.

Integrate natural language processing for translation.

Create a user-friendly interface for seamless interaction.

1.3.Problem Statement:

Communication is a fundamental aspect of human interaction, and for the deaf and hard-of-hearing community, sign language serves as a primary mode of expressing thoughts and emotions. However, a significant barrier exists between sign language users and individuals who do not understand sign language, hindering effective communication. The Real-Time Sign Language Detection project aims to address the following challenges: Communication Gap, Limited Accessibility, Inclusivity Issues, Technological Constraints , Educational and Professional Disparities etc.

2.LITERATURE REVIEW

There were several research papers and articles on real-time sign language detection. Please note that the field of research is dynamic, and new publications may have emerged since then. Here are some key papers and works that were notable at the time:

1.Title: Real-time American Sign Language Recognition from Video Sequences Using Hidden Markov Models

- Authors: Starner, Thad, and Alex Pentland
- Published in: IEEE Transactions on Pattern Analysis and Machine Intelligence (1998)
- This paper explores the use of Hidden Markov Models (HMMs) for real-time recognition of American Sign Language (ASL) from video sequences.

2.Title: A Real-Time System for Recognizing Unrestricted American Sign Language

- Authors: Starner, Thad, Alex Pentland, and Charles Matthew Grodon
- Published in: IEEE Transactions on Pattern Analysis and Machine Intelligence (1998)
- The authors present a real-time system for recognizing unrestricted ASL using a combination of computer vision techniques and statistical models.

3.Title: Real-Time Sign Language Recognition Using a Range Camera

- Authors: Juan José Alcaraz-Jiménez, José L. Alba-Castro, and Luis A. Guerrero
- Published in: Sensors (2012)
- This paper explores the use of a range camera for real-time sign language recognition, introducing a novel approach to capturing 3D hand movements.

4.Title: Real-time Sign Language Recognition using Convolutional Neural Networks

- Authors: Amit Das, Koushik Sinha, and Ayan Seal
- Published in: Procedia Computer Science (2016)
- The paper discusses the application of Convolutional Neural Networks (CNNs) for real-time sign language recognition, highlighting the advantages of deep learning in this context.

5.Title: Sign Language Recognition with Microsoft Kinect

- Authors: Ahmad Al Hanbali, Awni Hannun, Dan Jurafsky, and Andrew Y. Ng
- Published in: Stanford University Technical Report (2012)
- This report explores the use of Microsoft Kinect for sign language recognition, providing insights into the challenges and opportunities associated with this technology.

6.Title: Real-time Recognition of American Sign Language Alphabets Using Principal Component Analysis and Neural Networks

- Authors: Arun Rajagopalan, Salil Parekh, and Sridhar Krishnan
- Published in: Procedia Computer Science (2016)
- The paper presents a real-time sign language recognition system using Principal Component Analysis (PCA) and Neural Networks.

These references provide a starting point for understanding the evolution of real-time sign language detection techniques. To access these papers, you may use academic databases such as IEEE Xplore, PubMed, or Google Scholar. Keep in mind that the field continues to evolve, and it's advisable to check for the latest research publications for the most up-to-date information.

2.1.Motivation:

The motivation behind a Real-Time Sign Language Detection project stems from a recognition of the communication challenges faced by the deaf and hard-of-hearing community. The project aims to leverage technological advancements to address these challenges and contribute to creating a more inclusive and accessible society.

3. SYSTEM ARCHITECTURE

3.1 Gesture Recognition:

Utilizes deep learning models trained on the sign language dataset for accurate and real-time gesture recognition.

3.2 Hand Tracking:

Employs computer vision techniques to track the movement and position of the user's hands, ensuring precise recognition.

3.3 Natural Language Processing:

Converts recognized signs into text or speech using NLP algorithms, enhancing the interpretability of the system.

4.METHODOLOGY

4.1. Custom Dataset:

A custom dataset of sign language gestures was collected, comprising various signing styles, hand orientations, and lighting conditions. This dataset was used to train and validate the deep learning models.

4.2. Technology Stack:

Computer Vision: OpenCV for hand tracking.

Machine Learning:YOLOV5,Torch and Annotation Tool

Natural Language Processing: Integration of NLP algorithms for translation.

5.MODULES

5.1.Capture Images:

capture_image.py

```python
import os
import cv2
import time
import uuid

IMAGE_PATH='CollectedImages'

labels=['Hello','Yes','No','Thanks','IloveYou','Please']

number_of_images=20

for label in labels:
    img_path = os.path.join(IMAGE_PATH, label)
    os.makedirs(img_path)
    cap=cv2.VideoCapture(0)
    print('Collecting images for {}'.format(label))
    time.sle
    for imgn                    s):
        ret,frame=cap.read()
        imagename=os.path.join(IMAGE_PATH,label,label+'.'+'{}.jpg'.format(str(uuid.uuid1())))
        cv2.imwrite(imagename,frame)
        cv2.imshow('frame',frame)
        time.sleep(2)

        if cv2.waitKey(1) & 0xFF==ord('q'):
            break
    cap.release()
```

(variable) frame: MatLike

It is the first module .In this module we collected the images and create dataset . we use Packages and Libraries like cv2,uuid,time,os etc . After collecting Images we give the labels to the Images by using Annotation Tool.

.

5.2.Install Dependences:

(Remember to choose GPU in Runtime if not already selected. Runtime --> Change Runtime Type --> Hardware accelerator --> GPU)

+ Code + Markdown

```
# clone YOLOv5 repository
!git clone https://github.com/ultralytics/yolov5.git  # clone repo
%cd yolov5
```

Python

```
Cloning into 'yolov5'...
remote: Enumerating objects: 14106, done.
remote: Counting objects: 100% (124/124), done.
remote: Compressing objects: 100% (55/55), done.
remote: Total 14106 (delta 83), reused 106 (delta 69), pack-reused 13982
Receiving objects: 100% (14106/14106), 13.30 MiB | 18.28 MiB/s, done.
Resolving deltas: 100% (9740/9740), done.
/content/yolov5
```

Here we can we clone the YOLOv5 from the Git hub. YOLO (You Only Look Once) is a popular object detection framework, and YOLOv5 is one of its iterations. YOLOv5 has been widely used in various computer vision applications, including real-time object detection.

```
# install dependencies as necessary
!pip install -qr requirements.txt  # install dependencies (ignore errors)
import torch

from IPython.display import Image, clear_output  # to display images
# from utils.google_utils import gdrive_download  # to download models/datasets

# clear_output()
print('Setup complete. Using torch %s %s' % (torch.__version__, torch.cuda.get_device_properties(0) if torch.cu
```

Python

```
|▏                                | 10 kB 30.5 MB/s eta 0:00:01
|▍                                | 20 kB 11.6 MB/s eta 0:00:01
|▋                                | 30 kB 15.9 MB/s eta 0:00:01
|▉                                | 40 kB 14.0 MB/s eta 0:00:01
|█                                | 51 kB 12.2 MB/s eta 0:00:01
|█▎                               | 61 kB 14.1 MB/s eta 0:00:01
|█▌                               | 71 kB 15.1 MB/s eta 0:00:01
|█▊                               | 81 kB 16.2 MB/s eta 0:00:01
|██                               | 92 kB 13.7 MB/s eta 0:00:01
|██▏                              | 102 kB 13.9 MB/s eta 0:00:01
|██▍                              | 112 kB 13.9 MB/s eta 0:00:01
|██▋                              | 122 kB 13.9 MB/s eta 0:00:01
|██▉                              | 133 kB 13.9 MB/s eta 0:00:01
|███                              | 143 kB 13.9 MB/s eta 0:00:01
|███▎                             | 153 kB 13.9 MB/s eta 0:00:01
```

Cell 1 of 33

We can install necessary Packages for sign language detection. YOLOv5 is implemented using Python Torch, a popular deep learning framework. Ensure that you have Python Torch installed on your system. Here we can download the dataset.

5.3. Your Custom Data:

Open Google Colab:

Go to Google Colab in your web browser.

Start a New Notebook:

Create a new document where you can write and run code.

Drag and Drop ZIP File:

Find your ZIP file on your computer.

Drag it into the Colab document.

Wait for Upload:

Let Colab upload your ZIP file.

Use Code to Extract:

Write a little bit of code to open the ZIP file and take out the stuff inside.

Access Your Data:

Your files are now available in the Colab notebook. You can see and use them.

Here we can export the code snippet from git hub and paste it here. After that we can load the data from YAML File into this notebook.

5.4. Define Model Configuration and Architecture:

```python
# define number of classes based on YAML
import yaml
with open("data.yaml", 'r') as stream:
    num_classes = str(yaml.safe_load(stream)['nc'])
```

Python

```python
num_classes
```

Python

```
'6'
```

```python
#this is the model configuration we will use
%cat /content/yolov5/models/yolov5s.yaml
```

Python

```
# YOLOv5 🚀 by Ultralytics, GPL-3.0 license

# Parameters
nc: 80  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 v6.0 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Conv, [64, 6, 2, 2]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, C3, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 6, C3, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, C3, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
```

Cell 1 of 33

```python
#customize iPython writefile so we can write variables
from IPython.core.magic import register_line_cell_magic

@register_line_cell_magic
def writetemplate(line, cell):
    with open(line, 'w') as f:
        f.write(cell.format(**globals()))
```

Python

```
%%writetemplate /content/yolov5/models/custom_yolov5s.yaml

# parameters
nc: {num_classes}  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
```

Cell 1 of 33

By using yolov5.yaml model configuration. Here we use yaml files, IPython.core.magic packages.

5.5. Train Custom YOLOv5 Detector:

```
# Train YOLOv5s on COCO128 for 100 epochs
!python train.py --img 640 --batch 16 --epochs 100 --data custom_data.yaml --weights yolo
```

```
               all         30         29      0.522        0.6      0.678      0.302

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     14/99      4.24G    0.06046    0.01809    0.02622         16        640: 100%|██████
                Class     Images  Instances          P          R      mAP50   mAP50-95:
                  all         30         29      0.598      0.779      0.704      0.299

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     15/99      4.24G    0.06017    0.01699    0.02482         10        640: 100%|██████
                Class     Images  Instances          P          R      mAP50   mAP50-95:
                  all         30         29      0.633        0.5      0.548      0.229

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     16/99      4.24G    0.05765    0.01671    0.01815         12        640: 100%|██████
                Class     Images  Instances          P          R      mAP50   mAP50-95:
                  all         30         29      0.496       0.75      0.779      0.389

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     17/99      4.24G    0.05729    0.01632    0.02269         13        640: 100%|██████
                Class     Images  Instances          P          R      mAP50   mAP50-95:
                  all         30         29      0.497      0.761       0.76       0.34

     Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
     18/99      4.24G    0.06033    0.01644    0.02107         17        640: 100%|██████
```

We can train the yolov5 on custom data.

5.6. Evaluate Custom YOLOv5 Detector Performance:

```
# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
▶ Launch TensorBoard Session
%load_ext tensorboard
▶ Launch TensorBoard Session
%tensorboard --logdir runs
```

Python

```
!nvidia-smi
```

Python

```
# we can also output some older school graphs if the tensor board isn't working for whatever reason...
from utils.plots import plot_results  # plot results.txt as results.png
Image(filename='/content/yolov5/runs/train/yolov5s_results/results.png', width=1000)  # view results.png
```

Python

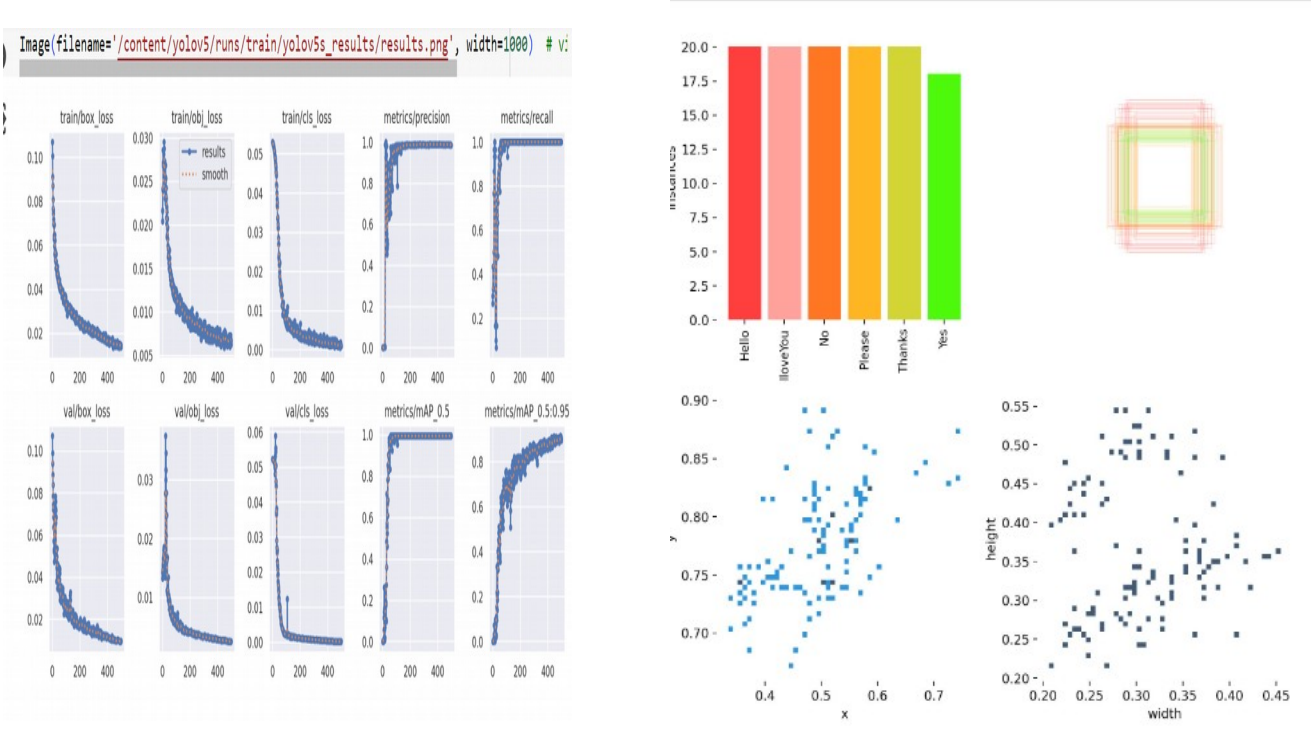

Fig.5.6

Training losses and performance metrics are saved to Tensorboard and also to a logfile defined above with the **--name** flag when we train. In our case, we named this ‘yolov5s_results`. (If given no name, it defaults to `results.txt`.) The results file is plotted as a png after training completes.

Note from Glenn: Partially completed `results.txt` files can be plotted with `from utils.utils import plot_results; plot_results()`.

5.7. Visualize our Training Data with Labels:

```
# first, display our ground truth data
print("GROUND TRUTH TRAINING DATA:")
Image(filename='/content/yolov5/runs/train/yolov5s_results/val_batch0_labels.jpg', width=900)
```

Python

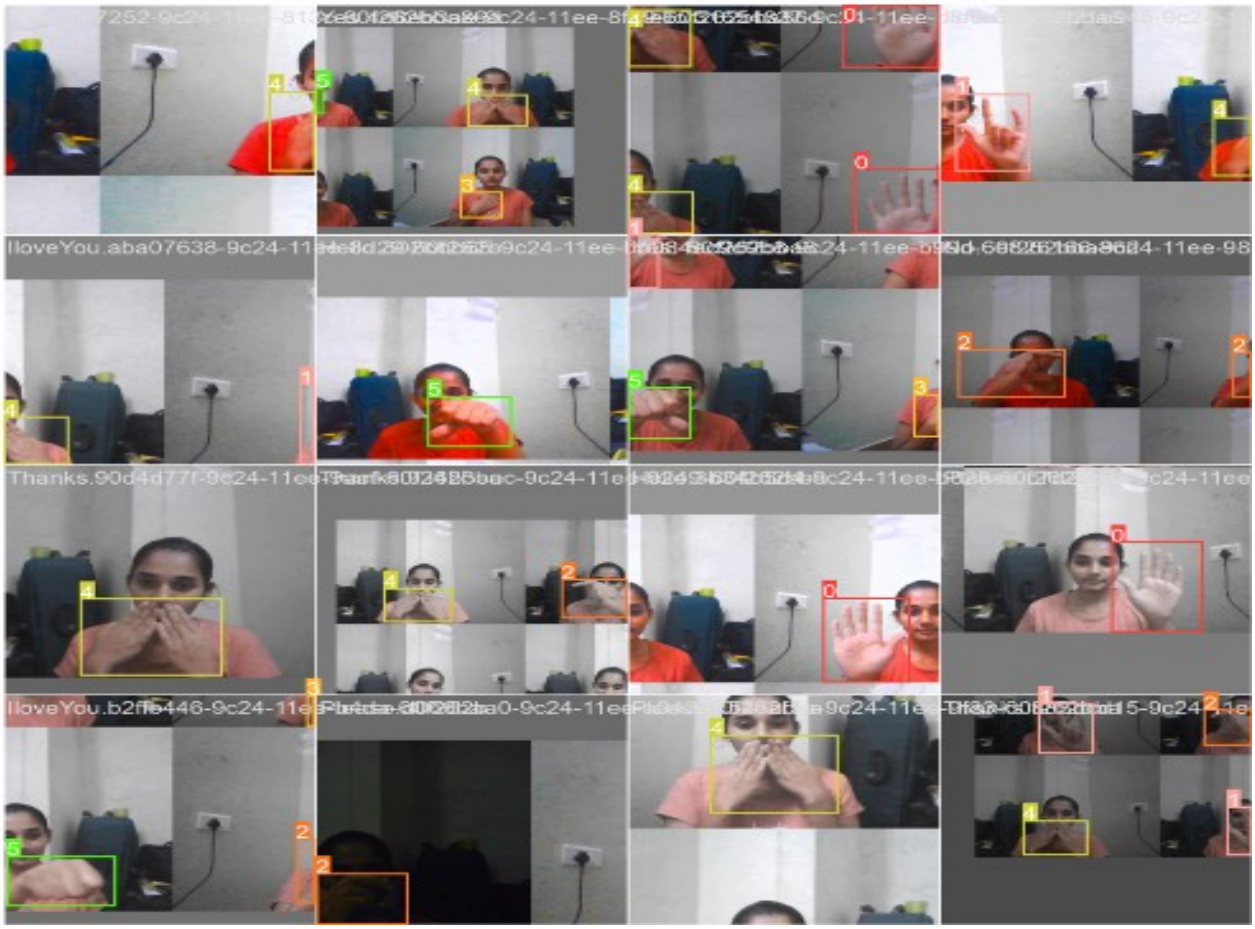

Fig.5.7

```
# print out an augmented training example
print("GROUND TRUTH AUGMENTED TRAINING DATA:")
Image(filename='/content/yolov5/runs/train/yolov5s_results/train_batch0.jpg', width=900)
```

```
#display inference on ALL test images
#this looks much better with longer training above

import glob
from IPython.display import Image, display

for imageName in glob.glob('/content/yolov5/runs/detect/exp*.jpg'): #assuming JPG
    display(Image(filename=imageName))
    print("\n")
```

After training starts, view `train*.jpg` images to see training images, labels and augmentation effects.

6.RESULTS AND DISCUSSIONS

The Real-Time Sign Language Detection System demonstrated high accuracy in recognizing a wide range of sign language gestures. Real-time processing capabilities were validated, with low-latency performance ensuring a seamless user experience.

OUTPUT:

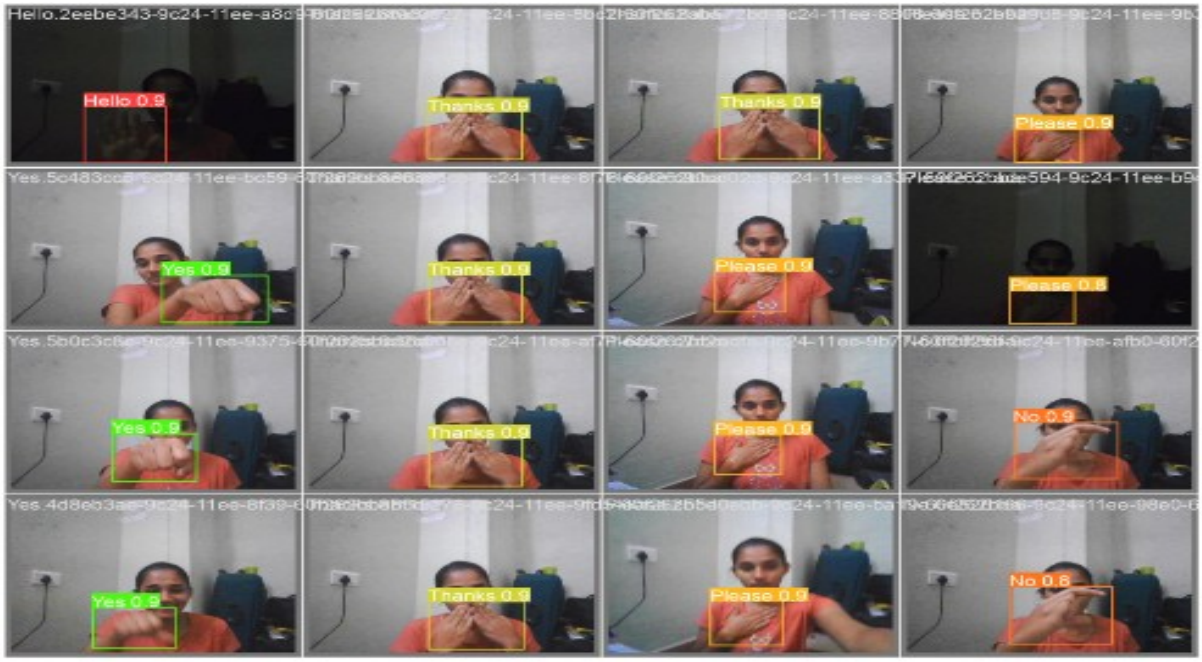

Fig.6.1

7.CONCLUSION AND FUTURE ENHANCEMENT

7.1. Conclusion:

The Real-Time Sign Language Detection System represents a significant step towards breaking down communication barriers for the deaf and hard-of-hearing community. Through the integration of computer vision, machine learning, and natural language processing, this system contributes to fostering inclusivity and equal access to communication

7.2 . Future Enhancement:

- Continuous improvement of gesture recognition through additional training on diverse datasets.
- Integration of more advanced hand tracking algorithms for improved accuracy.
- Expansion of the system to recognize and interpret facial expressions for enhanced communication.

8.REFERENCES

here is a list of reference links that we have used for our project:

1.Google (General research and information):

- https://www.google.com

2.Github (Datasets, machine learning resources):

- https://www.github.com

3.Geeks for Geeks (Programming tutorials and resources):

- https://www.geeksforgeeks.org

4.Javatpoint (Programming tutorials and resources):

- https://www.javatpoint.com

5.YouTube (Video tutorials and explanations):

- https://www.youtube.com