Gesture - Based Language Recognition

Problem Statement:

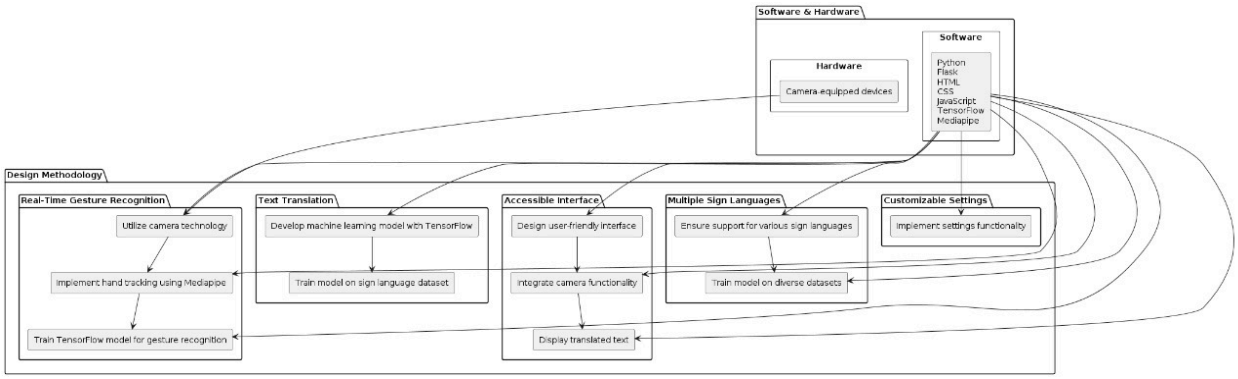
Develop an innovative system that utilizes camera technology in web to translate sign language gestures into text.The primary goal is to enhance communication accessibility for the Deaf and Hard of Hearing community by providing a real-time sign language-to-text translation solution.

Modularization Of Problem Statement:

1. Real-time gesture recognition
2. Text Translation
3. Accesable interface

Design Methodology:

1. **Real-Time Gesture Recognition:**
   Utilize camera technology for capturing hand gestures.
   Implement hand tracking using Mediapipe.
   Train a TensorFlow model for recognizing sign language gestures.
2. **Text Translation:**
   Develop a machine learning model using TensorFlow to translate recognized gestures into text.Train the model on a dataset of sign language gestures and their corresponding text Labels.
3. **Accessible Interface:**
   Design a user-friendly interface using HTML, CSS, and JavaScript.
   Integrate camera functionality for gesture capture.
   Display translated text in a readable format.



Software & Hardware Requirements:

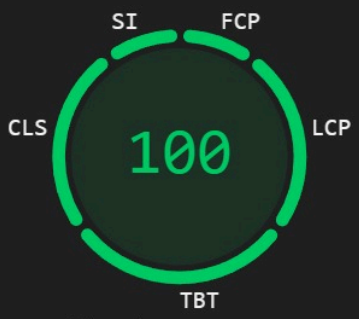
- **Software Requirements:**
    1. Html
    2. Css
    3. Javascript
    4. Python
    5. Flask
    6. Tensorflow
    7. mediapipe

- **Hardware Requirements:**
  Camera - equipped devices (webcams for desktops,front-facing cameras for mobile

Test case Specification:

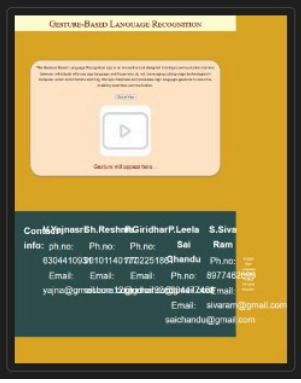
Input: Capture various sign language gestures using the application.
Expected Output: Accurate translation of gestures into corresponding text.
Test Scenarios:Test with various hand positions and movements.
Test with different sign languages.

100	69	96	67	PWA
Performance	Accessibility	Best Practices	SEO	PWA



Performance

Values are estimated and may vary. The performance score



100 69 96 67 PWA

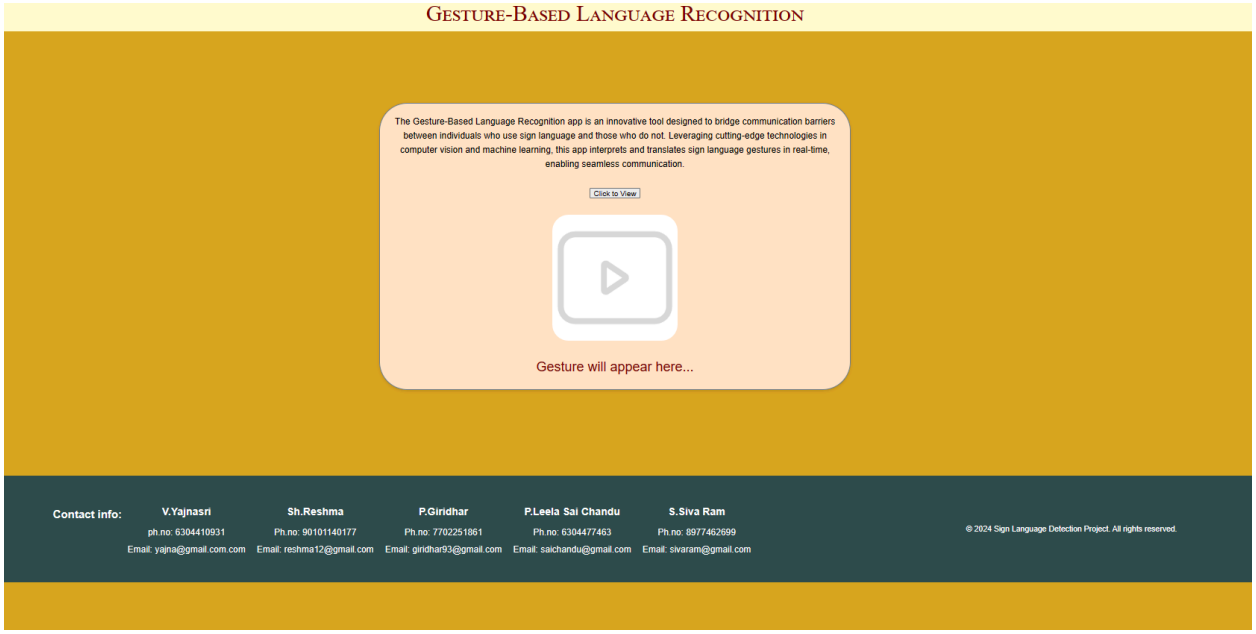
▲ 0–49 ■ 50–89 ● 90–100

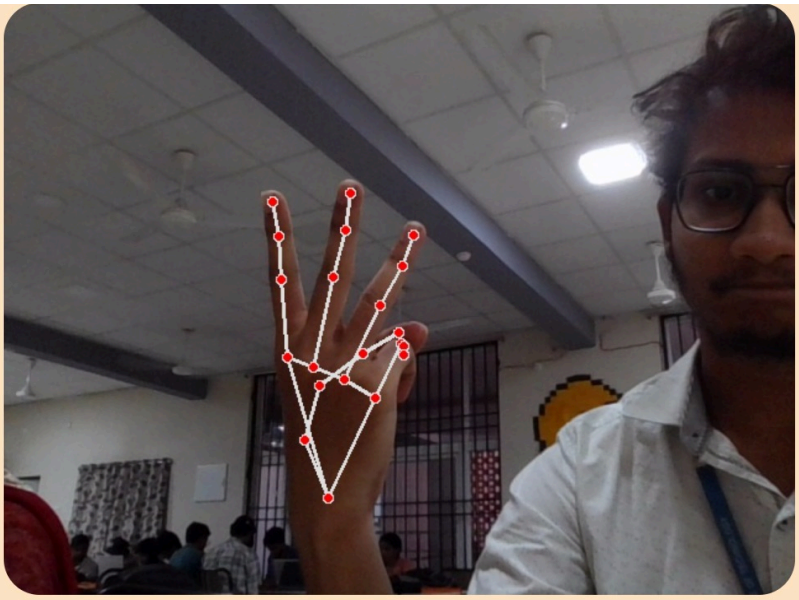
METRICS Expand view

- First Contentful Paint: 0.8 s
- Largest Contentful Paint: 0.8 s
- Total Blocking Time: 10 ms
- Cumulative Layout Shift: 0.028
- Speed Index: 0.8 s

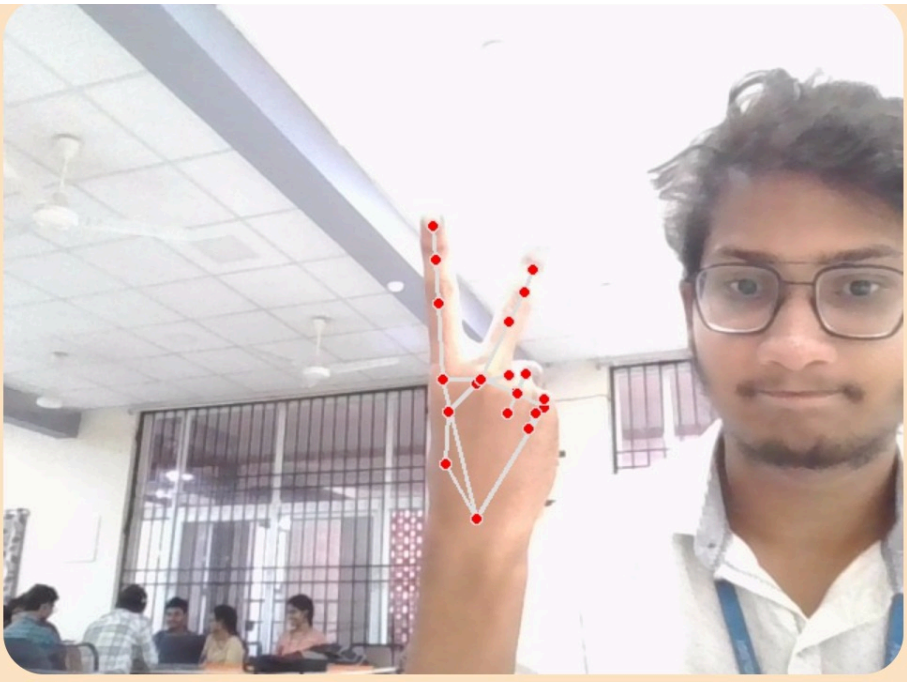
View Treemap

Screenshot Of Output:





W



V

C

Code Screenshots:

index.html app1.py

```python
from flask import Flask, render_template, Response
import cv2
import mediapipe as mp
from model import KeyPointClassifier
import itertools
import copy
import csv

app = Flask(__name__, template_folder='template')

current_gesture = None  # Global variable to store the current gesture

# Load the KeyPointClassifier model
keypoint_classifier = KeyPointClassifier()

# Read labels from a CSV file
with open('model/keypoint_classifier/label.csv', encoding='utf-8-sig') as f:
    keypoint_classifier_labels = [row[0] for row in csv.reader(f)]

# Initialize the video capture
vid = cv2.VideoCapture(0)

# Helper functions
def calc_landmark_list(image, landmarks):
    image_width, image_height = image.shape[1], image.shape[0]
    landmark_point = []
    for _, landmark in enumerate(landmarks.landmark):
        landmark_x = min(int(landmark.x * image_width), image_width - 1)
        landmark_y = min(int(landmark.y * image_height), image_height - 1)
        landmark_point.append([landmark_x, landmark_y])
```

```
    return landmark_point

def pre_process_landmark(landmark_list):
    temp_landmark_list = copy.deepcopy(landmark_list)
    base_x, base_y = 0, 0
    for index, landmark_point in enumerate(temp_landmark_list):
        if index == 0:
            base_x, base_y = landmark_point[0], landmark_point[1]
        temp_landmark_list[index][0] -= base_x
        temp_landmark_list[index][1] -= base_y
    temp_landmark_list = list(itertools.chain.from_iterable(temp_landmark_list))
    max_value = max(list(map(abs, temp_landmark_list)))
    return [x / max_value for x in temp_landmark_list]

def gen_frames():
    global current_gesture  # Access the global variable
    prev = ""
    with mp.solutions.hands.Hands(min_detection_confidence=0.5, min_tracking_confidence=0.5) as hands:
        while True:
            success, frame = vid.read()
            if not success:
                break
            image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            results = hands.process(image)
            if results.multi_hand_landmarks:
                for hand_landmarks in results.multi_hand_landmarks:
                    mp.solutions.drawing_utils.draw_landmarks(frame, hand_landmarks, mp.solutions.hands.HAND_CC
                    landmark_list = calc_landmark_list(frame, hand_landmarks)
                    pre_processed_landmark_list = pre_process_landmark(landmark_list)
```

```
                    mp.solutions.drawing_utils.draw_landmarks(frame, hand_landmarks, mp.solutions.hands.HAN
                    landmark_list = calc_landmark_list(frame, hand_landmarks)
                    pre_processed_landmark_list = pre_process_landmark(landmark_list)
                    hand_sign_id = keypoint_classifier(pre_processed_landmark_list)
                    cur = keypoint_classifier_labels[hand_sign_id]
                    if cur != prev:
                        print(cur)
                        prev = cur
                        current_gesture = cur  # Update the current gesture
            ret, buffer = cv2.imencode('.jpg', frame)
            frame = buffer.tobytes()
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/current_gesture')
def get_current_gesture():
    return current_gesture if current_gesture else "No gesture detected"

@app.route('/')
def index():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(debug=True, threaded=True)
```

```python
import csv

import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split

RANDOM_SEED = 42
```

Alpha cnt

```python
NUM_CLASSES = 27
```

Dataset reading

```python
X_dataset = np.loadtxt(dataset, delimiter=',', dtype='float32', usecols=list(range(1, (21 * 2) + 1)))
```

Python

```python
y_dataset = np.loadtxt(dataset, delimiter=',', dtype='int32', usecols=(0))
```

Python

```python
X_train, X_test, y_train, y_test = train_test_split(X_dataset, y_dataset, train_size=0.75, random_state=RANDOM_SEED)
```

Model building

```python
model = tf.keras.models.Sequential([
    tf.keras.layers.Input((21 * 2, )),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(20, activation='relu'),
    tf.keras.layers.Dropout(0.4),
    tf.keras.layers.Dense(10, activation='relu'),
    tf.keras.layers.Dense(NUM_CLASSES, activation='softmax')
])
```

```
model.summary()  # tf.keras.utils.plot_model(model, show_shape
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dropout (Dropout)           (None, 42)                0

 dense (Dense)               (None, 20)                860

 dropout_1 (Dropout)         (None, 20)                0

 dense_1 (Dense)             (None, 10)                210

 dense_2 (Dense)             (None, 27)                297

=================================================================
Total params: 1367 (5.34 KB)
Trainable params: 1367 (5.34 KB)
```

```
model.summary()  # tf.keras.utils.plot_model(model, show_shapes=True)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dropout (Dropout)           (None, 42)                0

 dense (Dense)               (None, 20)                860

 dropout_1 (Dropout)         (None, 20)                0

 dense_1 (Dense)             (None, 10)                210

 dense_2 (Dense)             (None, 27)                297

=================================================================
Total params: 1367 (5.34 KB)
Trainable params: 1367 (5.34 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# Mo  (variable) model_save_path: str
cp_c                                    lCheckpoint(
    model_save_path, verbose=1, save_weights_only=False)
# Callback for early stopping
es_callback = tf.keras.callbacks.EarlyStopping(patience=25, verbose=1)
```

[9]

```
# Model compilation
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

[10]

```python
# Model checkpoint callback
cp_callback = tf.keras.callbacks.ModelCheckpoint(
    model_save_path, verbose=1, save_weights_only=False)
# Callback for early stopping
es_callback = tf.keras.callbacks.EarlyStopping(patience=25, verbose=1)
```

```python
# Model compilation
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

Model train

```python
model.fit(
    X_train,
    y_train,
    epochs=1000,
    batch_size=128,
    validation_data=(X_test, y_test),
    callbacks=[cp_callback, es_callback]
)
```

Python

```
Epoch 1/1000
1/7 [===>.....................] - ETA: 8s - loss: 3.4040 - accuracy: 0.0234
Epoch 1: saving model to model/keypoint_classifier\keypoint_classifier.hdf5
7/7 [==============================] - 2s 72ms/step - loss: 3.4305 - accuracy: 0.0203 - val_loss: 3.3812 - val_accuracy: 0
Epoch 2/1000
1/7 [===>.....................] - ETA: 0s - loss: 3.3745 - accuracy: 0.0156
Epoch 2: saving model to model/keypoint_classifier\keypoint_classifier.hdf5
7/7 [==============================] - 0s 16ms/step - loss: 3.3907 - accuracy: 0.0203 - val_loss: 3.3500 - val_accuracy: 0
Epoch 3/1000
1/7 [===>.....................] - ETA: 0s - loss: 3.4176 - accuracy: 0.0234
c:\Users\Tirth Kothari\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\training.py:3079: UserWa
  saving_api.save_model(
```

```
# Model evaluation
val_loss, val_acc = model.evaluate(X_test, y_test, batch_size=128)
```

```
3/3 [==============================] - 0s 6ms/step - loss: 0.9892 - accuracy: 0.6996
```

```
# Loading the saved model
model = tf.keras.models.load_model(model_save_path)
```

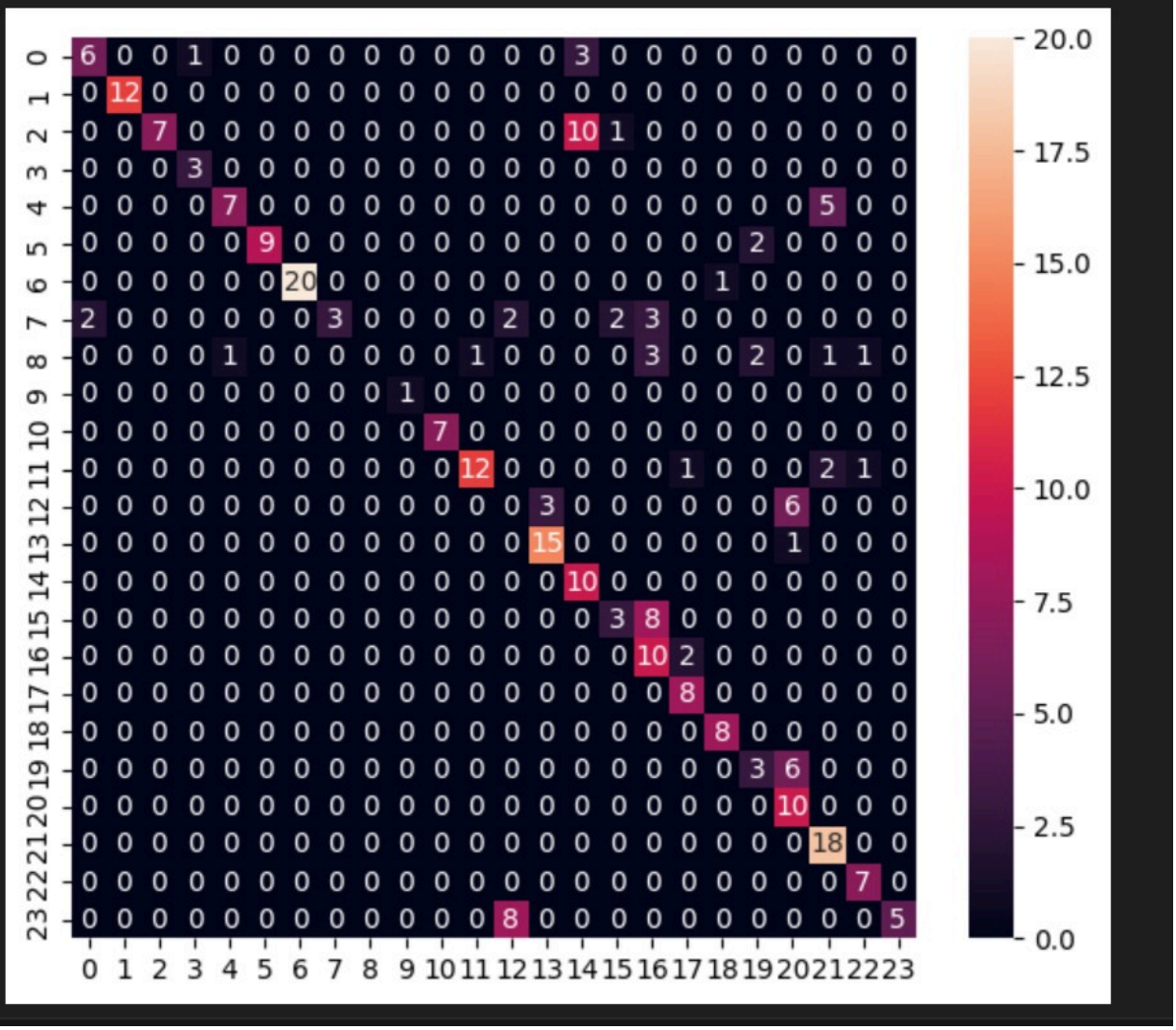
```
# Inference test
predict_result = model.predict(np.array([X_test[0]]))
print(np.squeeze(predict_result))
print(np.argmax(np.squeeze(predict_result)))
```

```
1/1 [==============================] - 0s 154ms/step
[7.3152198e-04 1.2456423e-10 2.2885220e-02 3.3734581e-03 3.0199146e-08
 1.3165612e-05 3.9126060e-04 8.1487961e-02 1.2231432e-03 4.7883476e-15
 6.4331852e-11 3.9766374e-07 5.3641636e-02 5.7961706e-02 3.2268139e-03
 3.0663386e-01 3.7988904e-01 6.7972427e-04 1.4341011e-03 2.4908180e-08
 5.1495289e-07 6.3423640e-09 5.5545904e-02 3.0880481e-02 2.9974046e-14
 1.5767528e-09 1.0445328e-08]
16
```

Conf mat

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report

def print_confusion_matrix(y_true, y_pred, report=True):
    labels = sorted(list(set(y_true)))
    cmx_data = confusion_matrix(y_true, y_pred, labels=labels)

    df_cmx = pd.DataFrame(cmx_data, index=labels, columns=labels)

    fig, ax = plt.subplots(figsize=(7, 6))
    sns.heatmap(df_cmx, annot=True, fmt='g' ,square=False)
    ax.set_ylim(len(set(y_true)), 0)
    plt.show()

    if report:
        print('Classification Report')
        print(classification_report(y_test, y_pred))

Y_pred = model.predict(X_test)
y_pred = np.argmax(Y_pred, axis=1)

print_confusion_matrix(y_test, y_pred)
```

```
9/9 [==============================] - 0s 2ms/step
```



Classification Report

	precision	recall	f1-score	support
0	0.75	0.60	0.67	10
1	1.00	1.00	1.00	12
2	1.00	0.39	0.56	18
3	0.75	1.00	0.86	3
4	0.88	0.58	0.70	12
5	1.00	0.82	0.90	11
6	1.00	0.95	0.98	21
7	1.00	0.25	0.40	12
8	0.00	0.00	0.00	9
9	1.00	1.00	1.00	1
10	1.00	1.00	1.00	7
11	0.92	0.75	0.83	16
12	0.00	0.00	0.00	9
13	0.83	0.94	0.88	16
14	0.43	1.00	0.61	10
15	0.50	0.27	0.35	11
16	0.42	0.83	0.56	12
17	0.73	1.00	0.84	8
18	0.89	1.00	0.94	8
19	0.43	0.33	0.38	9
20	0.43	1.00	0.61	10
21	0.69	1.00	0.82	18
...				
accuracy			0.70	263
macro avg	0.73	0.71	0.68	263
weighted avg	0.75	0.70	0.68	263

Tensorflow-Lite

```
# Save as a model dedicated to inference
model.save(model_save_path, include_optimizer=False)
```

```
c:\Users\Tirth Kothari\AppData\Local\Programs\Python\Python311\Lib\site-packages\keras\src\engine\training.py:3079: U
  saving_api.save_model(
```

```
# Transform model (quantization)

converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_quantized_model = converter.convert()

open(tflite_save_path, 'wb').write(tflite_quantized_model)
```

```
INFO:tensorflow:Assets written to: C:\Users\Tirth Kothari\AppData\Local\Temp\tmpbqy8bb60\assets
INFO:tensorflow:Assets written to: C:\Users\Tirth Kothari\AppData\Local\Temp\tmpbqy8bb60\assets

7556
```

```
%%time
# Inference implementation
interpreter.invoke()
tflite_results = interpreter.get_tensor(output_details[0]['index'])
```

```
CPU times: total: 0 ns
Wall time: 0 ns
```

```
print(np.squeeze(tflite_results))
print(np.argmax(np.squeeze(tflite_results)))
```

```
[7.3152198e-04 1.2456398e-10 2.2885226e-02 3.3734578e-03 3.0199093e-08
 1.3165598e-05 3.9126043e-04 8.1488028e-02 1.2231419e-03 4.7883471e-15
 6.4331734e-11 3.9766257e-07 5.3641632e-02 5.7961661e-02 3.2268136e-03
 3.0663398e-01 3.7988904e-01 6.7972357e-04 1.4340992e-03 2.4908129e-08
 5.1495283e-07 6.3423391e-09 5.5545837e-02 3.0880507e-02 2.9974046e-14
 1.5767528e-09 1.0445308e-08]
16
```

error solve

```
interpreter = tf.lite.Interpreter(model_path=tflite_save_path)
interpreter.allocate_tensors()
```

```
# Get I / O tensor
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
```

```
interpreter.set_tensor(input_details[0]['index'], np.array([X_test[0]]))
```

```
%%time
# Inference implementation
interpreter.invoke()
tflite_results = interpreter.get_tensor(output_details[0]['index'])
```

```html
<!DOCTYPE html>
<html>
<head>

    <style>

        body {
            background-color: goldenrod;
            height: 100vh;
            text-align: center;
            line-height: 1.6;
            font-family: Arial, sans-serif;
            margin: 0;
            padding: 0;
        }
        header {
            background-color: #fffdd0;
            color: maroon;
            font-size: 40px;
            padding-top: 5px;
            font-family: serif;
            font-variant: small-caps;
        }
        .project-description {
            margin-bottom: 20px;
            padding: 20px;
            max-width: 800px;
            margin: 0 auto;
            background-color: bisque;
            border-radius: 50px;
```

```
        box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
        border: 2px solid slategray;
    }
    .gesture-output {
        color: maroon;
        font-size: 24px;
        margin-top: 20px;
    }
    footer {
        display: flex;
        justify-content: space-around;
        align-items: center;
        padding: 40px;
        background-color: darkslategrey;
        color: #fff;
    }
    .member {
        text-align: center;
        margin: 0 10px;
    }
    .member h3 {
        margin: 10px 0;
    }
    .member p {
        margin: 5px 0;
    }
```

```
.copyright {
    text-align: center;
    font-size: 14px;
}
.contact-info{
    display: flex;
    padding:right 1px;

}
#con{
    font-size: 20px;
}

.material-symbols-outlined {
font-variation-settings:
'FILL' 0,
'wght' 400,
'GRAD' 0,
'opsz' 24
}
img{
    border-radius: 29px;
}

```

```
    </style>
    <title>Hand gestures</title>
</head>
<body>
    <header>Gesture-Based Language Recognition</header><br><br><br><br><br>
    <section class="project-description">
        The Gesture-Based Language Recognition app is an innovative tool designed to bridge communication barr
        <br><br>
        <button onclick="startCamera()">Click to View</button>
        <br><br>
        <img id="videoFeed" src="https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSv5o0rjzUZV8D1UqVE1ukE
        <div id="gestureOutput" class="gesture-output">Gesture will appear here...</div>
    </section><br><br><br><br><br><br>
    <footer>
        <div class="contact-info">
            <h1 id="con">Contact info:</h1>

            <div class="member">
                <h3>V.Yajnasri</h3>
                <p>ph.no: 6304410931</p>
                <p>Email: yajna@gmail.com.com</p>
            </div>
            <div class="member">
                <h3>Sh.Reshma</h3>
                <p>Ph.no: 90101140177</p>
                <p>Email: reshma12@gmail.com</p>
            </div>
```

```
            <div class="member">
                <h3>P.Giridhar</h3>
                <p>Ph.no: 7702251861 </p>
                <p>Email: giridhar93@gmail.com</p>
            </div>
            <div class="member">
                <h3>P.Leela Sai Chandu</h3>
                <p>Ph.no: 6304477463</p>
                <p>Email: saichandu@gmail.com</p>
            </div>
            <div class="member">
                <h3>S.Siva Ram</h3>
                <p>Ph.no: 8977462699</p>
                <p>Email: sivaram@gmail.com</p>
            </div>
        </div>
        <div class="copyright">
            &copy; 2024 Sign Language Detection Project. All rights reserved.
        </div>
    </footer>
    <script>
        function startCamera() {
            document.getElementById('videoFeed').src = "{{ url_for('video_feed') }}";
            setInterval(updateGesture, 1000); // Update gesture every second
        }

```

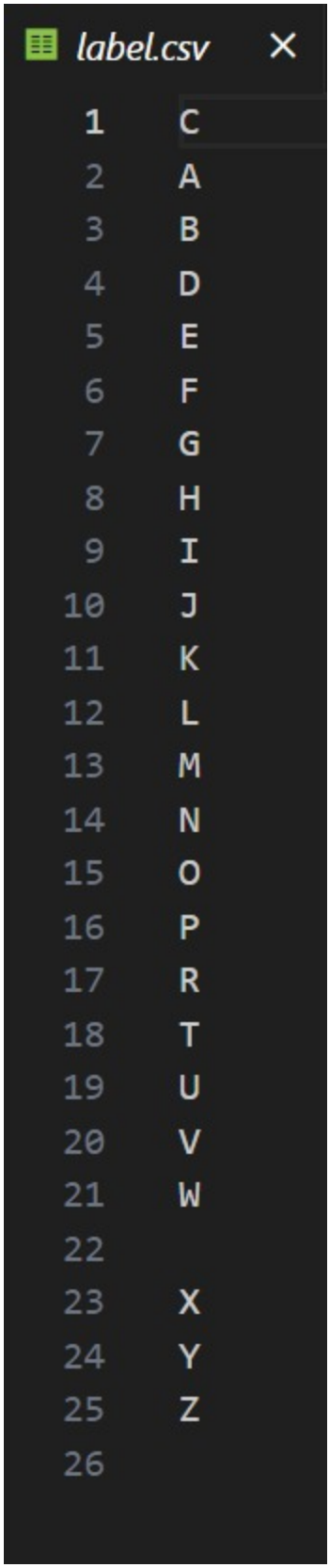
```
            &copy; 2024 Sign Language Detection Project. All rights reserved.
        </div>
    </footer>
    <script>
        function startCamera() {
            document.getElementById('videoFeed').src = "{{ url_for('video_feed') }}";
            setInterval(updateGesture, 1000); // Update gesture every second
        }

        function updateGesture() {
            fetch('{{ url_for("get_current_gesture") }}')  // Corrected the endpoint name here
                .then(response => response.text())
                .then(data => {
                    document.getElementById('gestureOutput').textContent = data;
                })
                .catch(error => console.error('Error fetching gesture:', error));
        }
    </script>
</body>
</html>
```

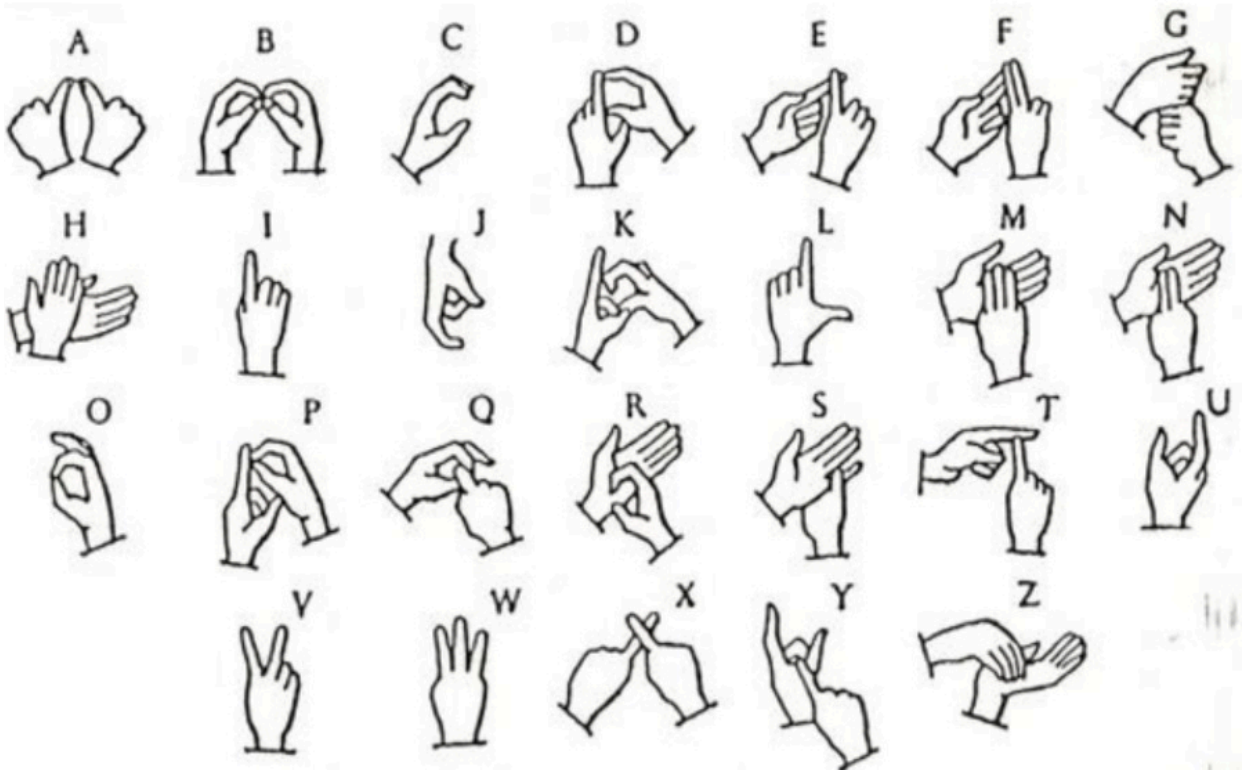
init.py

```
from model.keypoint_classifier.keypoint_classifier import KeyPointClassifier
```

coord.csv

```
1,0,0,-0.238938053,-0.230088496,-0.398230088,-0.566371681,-0.566371681,-0.814159292,-0.734513274,-1,0.08849557
1,0,0,0.277777778,-0.211111111,0.488888889,-0.477777778,0.666666667,-0.766666667,0.855555556,-1,0.011111111,-0
1,0,0,-0.23015873,-0.277777778,-0.357142857,-0.603174603,-0.492063492,-0.833333333,-0.642857143,-1,0.087301587
1,0,0,0.1484375,-0.2734375,0.2109375,-0.59375,0.265625,-0.8203125,0.3515625,-1,-0.2265625,-0.765625,0.015625,-
1,0,0,0.145299145,-0.213675214,0.239316239,-0.538461538,0.307692308,-0.794871795,0.41025641,-1,-0.196581197,-0
1,0,0,-0.238095238,-0.277777778,-0.365079365,-0.603174603,-0.492063492,-0.825396825,-0.642857143,-1,0.08730158
1,0,0,-0.264957265,-0.230769231,-0.427350427,-0.564102564,-0.564102564,-0.820512821,-0.709401709,-1,0.03418803
1,0,0,0.181818182,-0.218181818,0.290909091,-0.545454545,0.381818182,-0.809090909,0.518181818,-1,-0.145454545,-
1,0,0,0.170940171,-0.264957265,0.282051282,-0.581196581,0.401709402,-0.811965812,0.521367521,-1,-0.213675214,-
1,0,0,-0.269230769,-0.259615385,-0.432692308,-0.567307692,-0.596153846,-0.817307692,-0.798076923,-1,0.06730769
1,0,0,0.18018018,-0.27027027,0.297297297,-0.594594595,0.414414414,-0.828828829,0.54954955,-1,-0.225225225,-0.8
1,0,0,-0.276785714,-0.258928571,-0.4375,-0.580357143,-0.598214286,-0.8125,-0.794642857,-1,0.026785714,-0.94642
1,0,0,0.184210526,-0.263157895,0.307017544,-0.578947368,0.429824561,-0.815789474,0.561403509,-1,-0.219298246,-
1,0,0,-0.267857143,-0.267857143,-0.419642857,-0.598214286,-0.589285714,-0.830357143,-0.785714286,-1,0.04464285
1,0,0,-0.271929825,-0.254385965,-0.429824561,-0.587719298,-0.578947368,-0.824561404,-0.771929825,-1,0.02631578
1,0,0,0.172413793,-0.275862069,0.267241379,-0.577586207,0.362068966,-0.810344828,0.474137931,-1,-0.25,-0.81034
1,0,0,-0.260162602,-0.260162602,-0.398373984,-0.585365854,-0.520325203,-0.821138211,-0.682926829,-1,0.04065040
1,0,0,0.181818182,-0.26446281,0.26446281,-0.561983471,0.347107438,-0.801652893,0.446280992,-1,-0.223140496,-0.
1,0,0,0.152671756,-0.27480916,0.213740458,-0.572519084,0.267175573,-0.79389313,0.34351145,-1,-0.27480916,-0.74
1,0,0,-0.230769231,-0.276923077,-0.338461538,-0.584615385,-0.438461538,-0.815384615,-0.576923077,-1,0.07692307
1,0,0,-0.286956522,-0.252173913,-0.495652174,-0.573913043,-0.695652174,-0.826086957,-0.895652174,-0.991304348,
1,0,0,0.221153846,-0.211538462,0.384615385,-0.480769231,0.528846154,-0.701923077,0.711538462,-0.855769231,-0.1
1,0,0,-0.307017544,-0.192982456,-0.526315789,-0.5,-0.745614035,-0.719298246,-0.99122807,-0.868421053,-0.078947
1,0,0,0.257425743,-0.198019802,0.465346535,-0.485148515,0.663366337,-0.693069307,0.891089109,-0.851485149,-0.0
1,0,0,-0.290598291,-0.196581197,-0.52991453,-0.495726496,-0.752136752,-0.717948718,-1,-0.854700855,-0.06837606
1,0,0,0.245098039,-0.18627451,0.411764706,-0.431372549,0.578431373,-0.647058824,0.784313725,-0.803921569,-0.03
1,0,0,0.262135922,-0.145631068,0.495145631,-0.368932039,0.757281553,-0.495145631,1,-0.572815534,0.048543689,-0
1,0,0,-0.269230769,-0.146153846,-0.507692308,-0.407692308,-0.746153846,-0.546153846,-1,-0.584615385,-0.16923077
1,0,0,-0.243055556,-0.215277778,-0.354166667,-0.541666667,-0.416666667,-0.798611111,-0.534722222,-1,0.02083333
1,0,0,0.151079137,-0.223021583,0.223021583,-0.539568345,0.251798561,-0.784172662,0.323741007,-1,-0.244604317,-
```


label.csv
1 C
2 A
3 B
4 D
5 E
6 F
7 G
8 H
9 I
10 J
11 K
12 L
13 M
14 N
15 O
16 P
17 R
18 T
19 U
20 V
21 W
22
23 X
24 Y
25 Z
26



Future Scope:

The recognized gestures can be translated into text using Natural Language Processing(NLP). Integrate NLP algorithms to convert recognized sign language gestures into text.

References:

●https://www.researchgate.net/publication/378339145_Developing_an_Offline_and_Real-Time_Indian_Sign_Language_Recognition_System_with_Machine_Learning_and_Deep_Learning
●https://www.researchgate.net/publication/360641283_Sign_Language_Recognition_Application_using_Python_and_OpenCV

Conclusion:

Developing a real-time sign language recognition and translation system can significantly improve communication accessibility for the Deaf and Hard of Hearing community. By leveraging modern technologies such as machine learning and

camera technology, this project aims to bridge communication barriers and provide a seamless experience for users interacting
with sign language. The modular design approach ensures scalability, flexibility, and easy maintenance of the system.

Github-link:https://github.com/PerisettiGiridhar-1293/gesture-recognition/tree/main/gesture-recognition