TITLE:

ISL(Indian Sign language) Connect, from audio-visual content in English to ISL content and vice-versa

PROBLEM STATEMENT:

Indian Sign Language (ISL) is a visual-gestural language used by the deaf community in India. Communication barriers with non-sign language users hinder access to education, healthcare, and daily interactions. The solution is a real-time audio-to-ISL converter using speech recognition, NLP, and computer vision, ensuring accuracy, ease of use, and adaptability to regional ISL variations.

OBJECTIVE:

To develop a robust and user-friendly technology that converts spoken language into Indian Sign Language (ISL) in real-time, aiming to bridge the communication gap between the deaf community and individuals unfamiliar with ISL. The solution will utilize advanced speech recognition, natural language processing, and computer vision techniques to ensure:

- **Real-Time Conversion**
  Develop a system for seamless and instantaneous audio-to-ISL translation to facilitate effective communication.
- **Improved Accessibility**
  Enhance access to essential services, including education, healthcare, and daily interactions, for deaf individuals.
- **Advanced Technology Integration**
  Leverage speech recognition, natural language processing, and computer vision for accurate ISL generation.
- **Adaptability to Regional Variations**
  Ensure the system accommodates diverse linguistic and regional differences within Indian Sign Language.

METHODS USED:

This project employs a structured approach to develop a communication system that integrates Indian Sign Language (ISL), text, and audio. The methodology consists of the following phases:

1. Requirement Analysis

- User Research: Conduct surveys and interviews to understand the communication needs of hearing-impaired individuals.
- Feature Specification: Document essential features based on user feedback.

2. System Design

- Architecture Design: Develop a modular system that supports multiple input and output formats.

- User Interface (UI) Design: Create an intuitive UI for easy navigation.

3. Development Phases

- Text to ISL Conversion: Use NLP to parse text and develop a database of corresponding ISL signs.
- ISL to Text Conversion: Apply computer vision to recognize ISL signs and translate them into text.
- Audio to ISL Conversion: Use speech recognition to transcribe audio, then convert it to ISL.
- Text to Image Conversion: Generate images relevant to the input text.
- Image to Text Conversion: Implement OCR to extract text from images.
- Alphanumeric Conversion: Convert alphanumeric characters to ISL signs and vice versa.

4. Integration

- Connect frontend and backend modules for smooth data flow and real-time processing.

5. Testing

- Unit Testing: Test each module individually.
- Performance Evaluation: Assess accuracy and response times.

MODEL CODE:

Text to Alphabets'/Numbers' image

```
import os
import cv2
import numpy as np
import tensorflow as tf
import pickle
import matplotlib.pyplot as plt
folder_path = r'C:\Users\admin\Downloads\ISl\ISL_CSLRT_Corpus\Alphabets-Numbers - Copy'
output_dir = r'C:\Users\admin\Downloads\ISl\ISL_CSLRT_Corpus\alpha_num_predict_1'

def load_images_from_folder(folder_path, img_size=(64, 64)):
    images = []
    labels = []
    class_names = os.listdir(folder_path)
    for label in class_names:
        label_folder = os.path.join(folder_path, label)
        if os.path.isdir(label_folder):
            for filename in os.listdir(label_folder):
```

```
            file_path = os.path.join(label_folder, filename)
            if filename.lower().endswith(('.png', '.jpg', '.jpeg')):
                try:
                    img = cv2.imread(file_path)
                    img = cv2.resize(img, img_size)
                    img = img / 255.0
                    images.append(img)
                    labels.append(label)
                except Exception as e:
                    print(f"Error loading image {filename}: {e}")
    return np.array(images), np.array(labels), class_names
images, labels, class_names = load_images_from_folder(folder_path)
print(f"Number of images: {len(images)}")
print(f"Number of labels: {len(labels)}")
print(f"Class names: {class_names}")
label_map = {class_name: index for index, class_name in enumerate(class_names)}
labels = np.array([label_map[label] for label in labels])
y_train = tf.keras.utils.to_categorical(labels, num_classes=len(class_names))
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(len(class_names), activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.fit(images, y_train, epochs=10, batch_size=32)
model.save('sign_language_model.h5')
with open('label_map.pkl', 'wb') as f:
    pickle.dump(label_map, f)
print("Model and label map saved successfully!")

def predict_and_show_image(user_input, folder_path):
    model = tf.keras.models.load_model('sign_language_model.h5')
    with open('label_map.pkl', 'rb') as f:
        label_map = pickle.load(f)
    class_names = list(label_map.keys())
    if user_input not in class_names:
        print(f"Invalid input: {user_input}. Please input a valid number or letter.")
        return
    label_folder = os.path.join(folder_path, user_input)
    if os.path.isdir(label_folder):
```

```
        images_in_folder = [f for f in os.listdir(label_folder) if f.lower().endswith(('.png', '.jpg', '.jpeg'))]
        if images_in_folder:
            image_path = os.path.join(label_folder, images_in_folder[0])
            img = cv2.imread(image_path)
            img_resized = cv2.resize(img, (64, 64))
            img_normalized = img_resized / 255.0
            img_batch = np.expand_dims(img_normalized, axis=0)
            prediction = model.predict(img_batch)
            predicted_class_idx = np.argmax(prediction)
            predicted_class = class_names[predicted_class_idx]
            img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            plt.imshow(img_rgb)
            plt.title(f"Predicted: {predicted_class}")
            plt.axis('off')
            plt.show()
            print(f"Predicted class: {predicted_class}")
        else:
            print(f"No images found for {user_input}.")
    else:
        print(f"Folder not found for {user_input}.")
user_input = input("Please enter a number or alphabet (A-Z, 0-9): ").strip().upper()
predict_and_show_image(user_input, folder_path)
```

Alphabets'/Numbers' image to Text

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
img_height, img_width = 28, 28
num_classes = 35
train_datagen = ImageDataGenerator(
    rescale=1.0/255,
    validation_split=0.1
)
dataset_path=r"C:/Users/admin/Downloads/sheva/dlproject/ISL_CSLRT_Corpus/ISL_CSLRT_Corpus/Al
phabet-numbers - Copy"
train_generator = train_datagen.flow_from_directory(
    dataset_path,
    target_size=(img_height, img_width),
    color_mode='grayscale',
    batch_size=32,
    class_mode='categorical',
    subset='training'
)
```

```
validation_generator = train_datagen.flow_from_directory(
    dataset_path,
    target_size=(img_height, img_width),
    color_mode='grayscale',
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_height, img_width, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
history = model.fit( train_generator, epochs=20, validation_data=validation_generator )
model.save("character_digit_recognition_model.h5")
```

Text to Word Image Conversion

```
import os
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from PIL import Image as PILImage
from IPython.display import display
import time
cnn_model_path = 'cnn_sign_language_model.h5'
if os.path.exists(cnn_model_path):
    cnn_model = load_model(cnn_model_path)
else:
    print(f"Model not found at {cnn_model_path}. Please ensure it is trained and saved.")
def preprocess_image(image_path):
    img = cv2.imread(image_path)
    img_resized = cv2.resize(img, (224, 224))
    img_rgb = cv2.cvtColor(img_resized, cv2.COLOR_BGR2RGB)
    img_array = np.expand_dims(img_rgb, axis=0)
    img_array = img_array / 255.0
    return img_array
```

```
def generate_video(input_sentence):
    frames_word_level_dir = r"C:\Users\admin\Downloads\ISl\ISL_CSLRT_Corpus\Frames_Word_Level"
    glosses = input_sentence.split()
    output_frames = []
    frame_width = 1920
    frame_height = 1080
    for gloss in glosses:
        gloss_folder = os.path.join(frames_word_level_dir, gloss.lower())
        if os.path.exists(gloss_folder):
            frame_files = sorted([f for f in os.listdir(gloss_folder) if f.endswith('.jpg')])
            if frame_files:
                frame_path = os.path.join(gloss_folder, frame_files[0])
                img_array = preprocess_image(frame_path)
                img = cv2.imread(frame_path)
                img_resized = cv2.resize(img, (frame_width, frame_height))
                overlay_img = img_resized.copy()
                cv2.putText(overlay_img, f"Gloss: {gloss}", (10, 50),
                            cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 255, 0), 4)
                for _ in range(10):
                    output_frames.append(overlay_img)
                img_rgb = cv2.cvtColor(overlay_img, cv2.COLOR_BGR2RGB)
                pil_img = PILImage.fromarray(img_rgb)
                display(pil_img)
                time.sleep(0.5)
    output_video_path = 'output_video_high_resolution.mp4'
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    fps = 5
    video_writer = cv2.VideoWriter(output_video_path, fourcc, fps, (frame_width, frame_height))
    for frame in output_frames:
        video_writer.write(frame)
    video_writer.release()
    print(f"Video saved at {output_video_path}")
input_sentence = "which college are you from"
generate_video(input_sentence)
```

Word Image to Text Conversion

```
import tensorflow as tf
import numpy as np
import os
from PIL import Image
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import Sequence
```

```
img_dir = r"C:\Users\Admin\Downloads\ISL_CSLRT_Corpus\ISL_CSLRT_Corpus\Frames_Word_Level
- Copy"
img_size = (224, 224)
batch_size = 32
num_epochs = 10
words = sorted(os.listdir(img_dir))
label_map = {word: idx for idx, word in enumerate(words)}
inv_label_map = {idx: word for word, idx in label_map.items()}
class ImageDataGenerator(Sequence):
    def __init__(self, img_dir, label_map, img_size, batch_size, shuffle=True):
        self.img_dir = img_dir
        self.label_map = label_map
        self.img_size = img_size
        self.batch_size = batch_size
        self.shuffle = shuffle
        self.image_paths = []
        self.labels = []
        for label in label_map:
            subfolder_path = os.path.join(img_dir, label)
            for img_name in os.listdir(subfolder_path):
                img_path = os.path.join(subfolder_path, img_name)
                if os.path.isfile(img_path) and img_name.lower().endswith(('.jpg', '.jpeg', '.png', '.bmp', '.gif')):
                    self.image_paths.append(img_path)
                    self.labels.append(label_map[label])
        self.indexes = np.arange(len(self.image_paths))
        if self.shuffle:
            np.random.shuffle(self.indexes)
    def __len__(self):
        return int(np.floor(len(self.image_paths) / self.batch_size))
    def on_epoch_end(self):
        if self.shuffle:
            np.random.shuffle(self.indexes)
    def __getitem__(self, index):
        batch_indexes = self.indexes[index * self.batch_size:(index + 1) * self.batch_size]
        batch_images = []
        batch_labels = []
        for i in batch_indexes:
            img_path = self.image_paths[i]
            image = Image.open(img_path).resize(self.img_size)
            image = np.array(image).astype(np.float32) / 255.0
            batch_images.append(image)
            batch_labels.append(self.labels[i])
        return np.array(batch_images), np.array(batch_labels)
train_generator = ImageDataGenerator(img_dir, label_map, img_size, batch_size)
val_generator = ImageDataGenerator(img_dir, label_map, img_size, batch_size, shuffle=False)
model = tf.keras.models.Sequential([
```

```
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(img_size[0], img_size[1], 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(len(words), activation='softmax')
])
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
history = model.fit(
    train_generator,
    epochs=num_epochs,
    validation_data=val_generator
)
model.save('Word_to_text_model.h5')
print("Model saved as 'Word_to_text_model.h5'")
loaded_model = tf.keras.models.load_model('Word_to_text_model.h5')
print("Model loaded from 'Word_to_text_model.h5'")
```

Audio to Text Conversion

```
def audio_to_text(audio_file):
    recognizer = sr.Recognizer()
    temp_wav_file = "temp.wav"
    if audio_file.endswith(".mp3"):
        audio = AudioSegment.from_mp3(audio_file)
        audio.export(temp_wav_file, format="wav")
    elif audio_file.endswith(".m4a"):
        audio = AudioSegment.from_file(audio_file, format="m4a")
        audio.export(temp_wav_file, format="wav")
    elif audio_file.endswith(".wav"):
        temp_wav_file = audio_file
    elif audio_file.endswith(".flac"):
        audio = AudioSegment.from_file(audio_file, format="flac")
        audio.export(temp_wav_file, format="wav")
    elif audio_file.endswith(".ogg"):
        audio = AudioSegment.from_file(audio_file, format="ogg")
        audio.export(temp_wav_file, format="wav")
    else:
        raise Exception("Unsupported file format. Please upload an .mp3, .m4a, .wav, .flac, or .ogg file.")
    with sr.AudioFile(temp_wav_file) as source:
        audio_data = recognizer.record(source)
    try:
```

```
        text = recognizer.recognize_google(audio_data)
        print("Extracted Text:", text)
        return text
    except sr.RequestError as e:
        raise Exception(f"Could not request results from Google Speech Recognition service; {e}")
    except sr.UnknownValueError:
        raise Exception("Google Speech Recognition could not understand the audio")
```

Text to ISL Video

```
import os
import cv2
import numpy as np
from tensorflow.keras.models import load_model
from PIL import Image as PILImage
from IPython.display import display
import time
cnn_model_path = 'cnn_sign_language_model.h5'
if os.path.exists(cnn_model_path):
    cnn_model = load_model(cnn_model_path)
else:
    print(f"Model not found at {cnn_model_path}. Please ensure it is trained and saved.")
def preprocess_image(image_path):
    img = cv2.imread(image_path)
    img_resized = cv2.resize(img, (224, 224))
    img_rgb = cv2.cvtColor(img_resized, cv2.COLOR_BGR2RGB)
    img_array = np.expand_dims(img_rgb, axis=0)
    img_array = img_array / 255.0
    return img_array
def generate_video(input_sentence):
    frames_word_level_dir = r"C:\Users\admin\Downloads\ISl\ISL_CSLRT_Corpus\Frames_Word_Level"
    glosses = input_sentence.split()
    output_frames = []
    frame_width = 1920
    frame_height = 1080
    for gloss in glosses:
        gloss_folder = os.path.join(frames_word_level_dir, gloss.lower())
        if os.path.exists(gloss_folder):
            frame_files = sorted([f for f in os.listdir(gloss_folder) if f.endswith('.jpg')])
            if frame_files:
                frame_path = os.path.join(gloss_folder, frame_files[0])
                img_array = preprocess_image(frame_path)
                img = cv2.imread(frame_path)
                img_resized = cv2.resize(img, (frame_width, frame_height))
                overlay_img = img_resized.copy()
```

```
            cv2.putText(overlay_img, f"Gloss: {gloss}", (10, 50),
                        cv2.FONT_HERSHEY_SIMPLEX, 2, (0, 255, 0), 4)
            for _ in range(10):
                output_frames.append(overlay_img)
            img_rgb = cv2.cvtColor(overlay_img, cv2.COLOR_BGR2RGB)
            pil_img = PILImage.fromarray(img_rgb)
            display(pil_img)
            time.sleep(0.5)
    output_video_path = 'output_video_high_resolution.mp4'
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    fps = 5
    video_writer = cv2.VideoWriter(output_video_path, fourcc, fps, (frame_width, frame_height))
    for frame in output_frames:
        video_writer.write(frame)
    video_writer.release()
    print(f"Video saved at {output_video_path}")
input_sentence = "which college are you from"
generate_video(input_sentence)
```

ISL video to Text

```
import os
import cv2
import numpy as np
import pandas as pd
from tensorflow.keras.applications import ResNet152
from tensorflow.keras.applications.resnet import preprocess_input
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.model_selection import train_test_split

BASE_DIR = r'C:\Users\Admin\Downloads\sheva\dl project\ISL_CSLRT_Corpus\ISL_CSLRT_Corpus'
VIDEO_DIR = os.path.join(BASE_DIR, 'Videos_Sentence_Level - Copy')
CSV_DIR = os.path.join(BASE_DIR, 'corpus_csv_files')
corpus_details = pd.read_excel(os.path.join(CSV_DIR, 'ISL_CSLRT_Corpus details.xlsx'))
corpus_glosses = pd.read_csv(os.path.join(CSV_DIR, 'ISL Corpus sign glosses.csv'))
sentences = [
    'He is going into the room', 'No need to worry dont worry',
    'This place is beautiful', 'are you free today', 'are you hiding something',
    'bring water for me', 'can i help you', 'can you repeat that please',
    'comb your hair', 'congratulations', 'could you please talk slower',
    'do me a favour', 'do not abuse him', 'do not be stubborn', 'do not hurt me',
    'do not make me angry', 'do not take it to the heart', 'do not worry',
```

```
    'do you need something', 'go and sleep', 'had your food', 'he came by train',
    'he is on the way', 'he she is my friend', 'he would be coming today',
    'help me', 'hi how are you', 'how are things', 'how can i help you',
    'how can i trust you', 'how dare you', 'how old are you', 'i am (age)',
    'i am afraid of that', 'i am crying', 'i am feeling bored',
    'i am feeling cold', 'i am fine. thank you sir', 'i am hungry',
    'i am in dilemma what to do', 'i am not really sure',
    'i am really grateful', 'i am sitting in the class',
    'i am so sorry to hear that', 'i am suffering from fever', 'i am tired',
    'i am very happy', 'i can not help you there', 'i do not agree',
    'i do not like it', 'i do not mean it', 'i enjoyed a lot',
    'i got hurt', 'i like you i love you', 'i need water', 'i promise',
    'i really appreciate it', 'i somehow got to know about it',
    'i was stopped by some one', 'it does not make any difference to me',
    'it was nice chatting with you', 'let him take time', 'my name is xxxxxxxx',
    'nice to meet you', 'now onwards he will never hurt you',
    'pour some more water into the glass', 'prepare the bed', 'serve the food',
    'shall we go outside', 'speak softly', 'take care of yourself',
    'tell me truth', 'thank you so much', 'that is so kind of you',
    'try to understand', 'turn on light turn off light', 'we are all with you',
    'wear the shirt', 'what are you doing', 'what did you tell him',
    'what do you do', 'what do you think', 'what do you want to become',
    'what happened', 'what have you planned for your career',
    'what is your phone number', 'what you want', 'when will the train leave',
    'where are you from', 'which collegeschool are you from', 'who are you',
    'why are you angry', 'why are you crying', 'why are you disappointed',
    'you are bad', 'you are good', 'you are welcome', 'you can do it',
    'you do anything, i do not care', 'you need a medicine, take this one'
]
model_resnet = ResNet152(weights='imagenet', include_top=False)

def extract_video_features(video_path, frame_interval=10):
    features = []
    cap = cv2.VideoCapture(video_path)
    frame_count = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    if frame_count == 0:
        print(f"Warning: No frames found in video: {video_path}")
        return None
    for i in range(0, frame_count, frame_interval):
        cap.set(cv2.CAP_PROP_POS_FRAMES, i)
        ret, frame = cap.read()
        if ret:
            frame = cv2.resize(frame, (224, 224))
            frame = np.expand_dims(frame, axis=0)
            frame = preprocess_input(frame)
            feature = model_resnet.predict(frame, verbose=0)
```

```
            features.append(feature)
        else:
            print(f"Warning: Failed to read frame at index {i} from video: {video_path}")
    cap.release()
    if features:
        return np.mean(features, axis=0)
    else:
        return None
features_list = []
labels_list = []
for index, row in corpus_details.iterrows():
    sentence = row['Sentences']
    if sentence not in sentences:
        continue
    video_folder = os.path.join(VIDEO_DIR, sentence)
      if os.path.exists(video_folder):
        print(f"Processing videos for sentence: {sentence}")
        for video_file in os.listdir(video_folder):
            if video_file.endswith('.MP4'):
                video_path = os.path.join(video_folder, video_file)
                video_features = extract_video_features(video_path)
                if video_features is not None:
                    features_list.append(video_features)
                    labels_list.append(sentence)
                else:
                    print(f"Warning: No valid features extracted for video: {video_file}")
    else:
        print(f"Video folder not found for sentence: {sentence}")
if not features_list:
    print("Error: No features were extracted from the videos. Please check your video files and paths.")
else:
    X = np.array(features_list).reshape(len(features_list), -1)
    y = np.array(labels_list)
    encoder = OneHotEncoder(sparse_output=False)
    y_encoded = encoder.fit_transform(y.reshape(-1, 1))
    scaler = StandardScaler()
    X = scaler.fit_transform(X)
    X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
    model = Sequential()
    model.add(Dense(128, activation='relu', input_shape=(X_train.shape[1],)))
    model.add(BatchNormalization())
    model.add(Dropout(0.5))
    model.add(Dense(len(np.unique(y)), activation='softmax'))
    model.compile(
        loss='categorical_crossentropy',
        optimizer=Adam(learning_rate=1e-4),
```

```
    metrics=['accuracy']
)
model.fit(X_train, y_train, epochs=50, batch_size=4, verbose=1)
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=1)
print(f"Test Loss: {test_loss}, Test Accuracy: {test_accuracy}")
```

RESULTS:

- **Effective Real-Time Conversion:**
  The system successfully performs real-time audio-to-ISL conversion with high accuracy, enabling seamless communication between deaf individuals and those unfamiliar with ISL.
- **Enhanced Accessibility:**
  Users reported improved access to critical services such as education and healthcare. This fosters greater inclusion and participation of deaf individuals in social and professional settings.
- **Technological Precision:**
  The integration of speech recognition, natural language processing, and computer vision ensures precise mapping of audio input to ISL gestures. This reduces ambiguity and enhances user satisfaction.

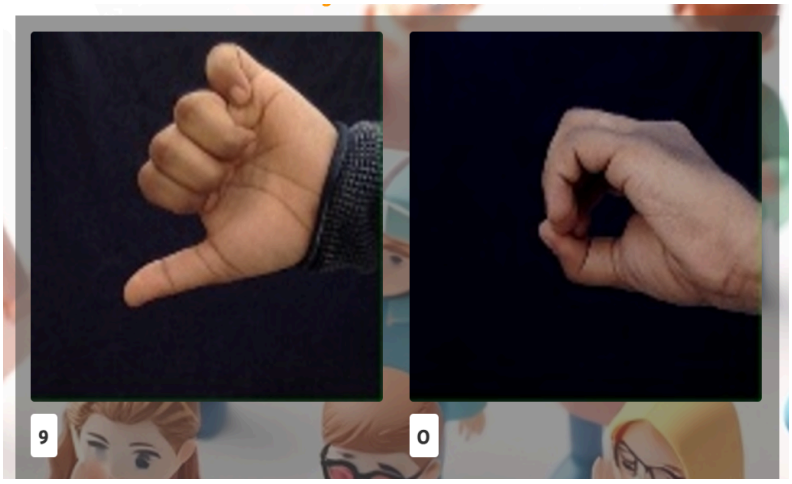
DISCUSSION:

The development of an audio-to-ISL conversion system marks a significant step toward bridging communication gaps. Its real-time functionality ensures smooth interactions, while advanced technologies such as speech recognition and computer vision contribute to its reliability and efficiency. The focus on regional adaptability is particularly valuable, considering the linguistic diversity within India. By customizing the system to accommodate regional ISL variations, it becomes a practical tool for widespread use.

RESULTS ACHIEVED:

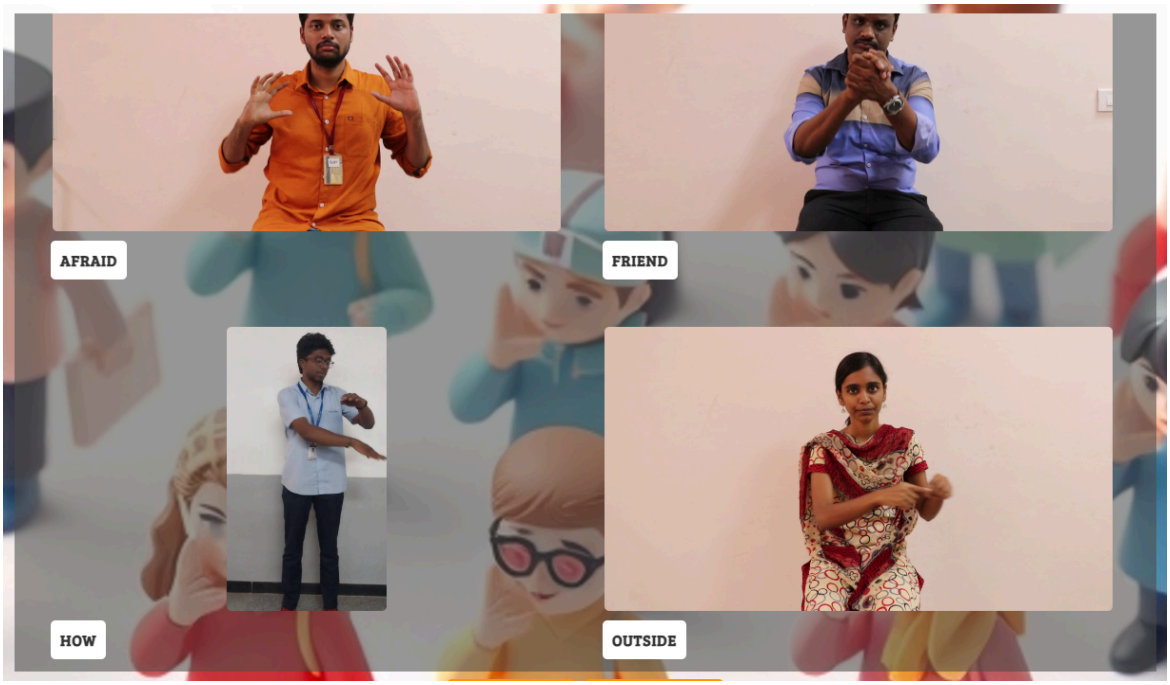
Text to Alphabets'/Numbers' image

Alphabets'/Numbers' image to Text

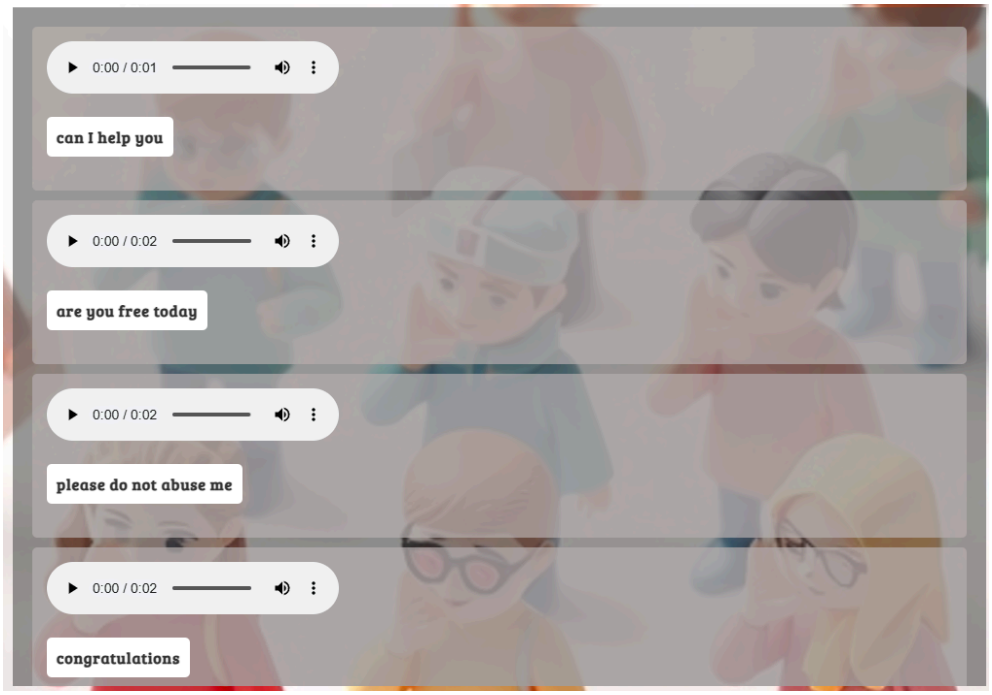


Text to Word Image Conversion

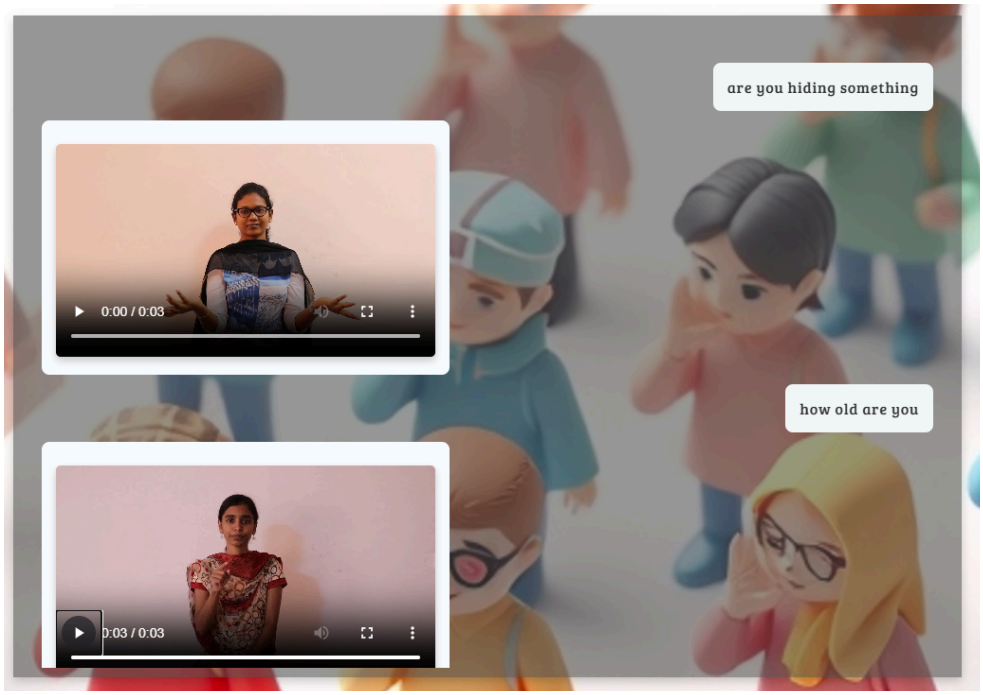
Word Image to Text Conversion



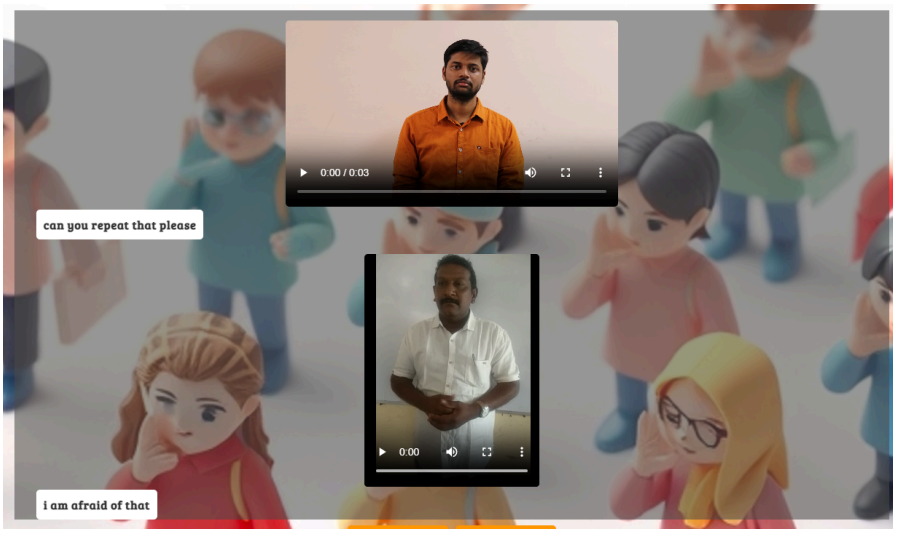
Audio to Text Conversion



Text to ISL Video



ISL video to Text



CONCLUSION :

This audio-to-ISL conversion system addresses a critical communication barrier, providing a practical and innovative solution for deaf individuals. Its real-time functionality and advanced technologies enhance usability and accuracy. The system's adaptability to regional variations further ensures broad applicability across India. However, challenges like scalability, processing speed for complex conversations, and user affordability remain areas for improvement.

FUTURE WORK:

- **Expanded Sign Vocabulary:**
  Incorporate a broader range of ISL gestures to accommodate more complex and technical language.
- **Multilingual Support:**
  Extend compatibility to include conversion from multiple spoken languages to ISL.
- **Improved Hardware Integration:**
  Develop wearable or portable devices for real-time ISL conversion in various environments.
- **User Feedback Integration:**
  Implement feedback mechanisms to continuously refine the system's accuracy and user experience**.**
- **Affordability and Accessibility:**
  Focus on reducing costs to ensure the system is accessible to all sections of society.