Introduction

The aim of this project is to develop an automatic sign language transcription system to assist individuals who are deaf or hard of hearing in communicating effectively. By leveraging computer vision and machine learning techniques, the system converts sign language gestures into written text in real-time, facilitating seamless communication.

Methodology

Custom Sign Language:

The chosen sign language includes essential signs like "bring," "coffee," "tea," and "not," selected for their clarity and common usage.

Clarification for each sign:

Bring: The reason behind me choosing bring as shown in the video is because, it felt the most common sign used when implying to bring. Coffee: The sign for coffee was originally extracted from my memory of some sign language, and I believe it is similar to the actual sign language of coffee (which is a C by hand). Not: In the video, you can see making my brother do a finger cross which I made mean 'not' because it kind of resembles an X. Tea: I honestly don't have a reason why such a sign for 'Tea' but it felt kind of like the letter T for Tea I guess?

Dataset Creation:

I've used an original video of my brother making all the signs and cut each sign alone to use for training. I made sure the signs are clear and the video is in good quality to allow easy training and frame captures.

Model Selection and Transfer Learning:

The base image classifier MobileNetV2 was selected for its efficiency in image recognition tasks. Transfer learning was employed to train the model on the custom sign language dataset, enabling it to recognize and differentiate between different signs accurately.

Posture Recognition Implementation:

The system captures frames from videos, preprocesses them for optimal input, and utilizes the trained model to predict the corresponding sign.

Results

The following results include all codes and their outputs showcasing the results of each part. I'll also add the final videos showcasing the functionality of the code.

Clarification for output videos (2 videos):

output_video_all signs 2: This video is where the code does not fully function since it's a new video with a new angle and everything as explained in the discussion below. output_video_all signs trained: Here, in this video, the system works 100% since it is the original it was trained from as explained in the discussion too.

In [2]:
```
import cv2
import os

def extract_frames_from_video(video_path, output_folder, start_time=0, en
    if not os.path.exists(output_folder):
        os.makedirs(output_folder)  # Create the folder if it doesn't exi
    video = cv2.VideoCapture(video_path)
    if not video.isOpened():
        print(f"Failed to open video: {video_path}")
        return

    fps = video.get(cv2.CAP_PROP_FPS)
    if frame_rate <= 0 or fps <= 0:
        print("Error: Invalid frame rate or FPS.")
        return

    video.set(cv2.CAP_PROP_POS_MSEC, start_time * 1000)  # Set start time

    frame_count = 0
    frame_interval = int(fps / frame_rate) if fps > frame_rate else 1  #
    while True:
        success, frame = video.read()
        if not success:
            break

        if frame_count % frame_interval == 0:
            frame_path = os.path.join(output_folder, f"frame_{frame_count
            cv2.imwrite(frame_path, frame)
            print(f"Saved {frame_path}")

        frame_count += 1

        if end_time and frame_count >= fps * end_time:
            break

    video.release()
    print(f"Done extracting frames from {video_path}.")

# Update paths as necessary
```

```python
video_files = {
    'bring': '/Users/m201903466/cv final/bring.MOV',
    'coffee': '/Users/m201903466/cv final/coffee.MOV',
    'tea': '/Users/m201903466/cv final/tea.MOV',
    'not': '/Users/m201903466/cv final/not.MOV'
}
output_base_folder = 'cv final'

for sign, video_path in video_files.items():
    output_folder = os.path.join(output_base_folder, sign)
    extract_frames_from_video(video_path, output_folder, frame_rate=100)
```

```
Saved cv final/bring/frame_0.jpg
Saved cv final/bring/frame_1.jpg
Saved cv final/bring/frame_2.jpg
Saved cv final/bring/frame_3.jpg
Saved cv final/bring/frame_4.jpg
Saved cv final/bring/frame_5.jpg
Saved cv final/bring/frame_6.jpg
Saved cv final/bring/frame_7.jpg
Saved cv final/bring/frame_8.jpg
Saved cv final/bring/frame_9.jpg
Saved cv final/bring/frame_10.jpg
Saved cv final/bring/frame_11.jpg
Saved cv final/bring/frame_12.jpg
Saved cv final/bring/frame_13.jpg
Saved cv final/bring/frame_14.jpg
Saved cv final/bring/frame_15.jpg
Saved cv final/bring/frame_16.jpg
Saved cv final/bring/frame_17.jpg
Saved cv final/bring/frame_18.jpg
Saved cv final/bring/frame_19.jpg
Saved cv final/bring/frame_20.jpg
Saved cv final/bring/frame_21.jpg
Saved cv final/bring/frame_22.jpg
Saved cv final/bring/frame_23.jpg
Saved cv final/bring/frame_24.jpg
Done extracting frames from /Users/m201903466/cv final/bring.MOV.
Saved cv final/coffee/frame_0.jpg
Saved cv final/coffee/frame_1.jpg
Saved cv final/coffee/frame_2.jpg
Saved cv final/coffee/frame_3.jpg
Saved cv final/coffee/frame_4.jpg
Saved cv final/coffee/frame_5.jpg
Saved cv final/coffee/frame_6.jpg
Saved cv final/coffee/frame_7.jpg
Saved cv final/coffee/frame_8.jpg
Saved cv final/coffee/frame_9.jpg
Saved cv final/coffee/frame_10.jpg
Saved cv final/coffee/frame_11.jpg
Saved cv final/coffee/frame_12.jpg
Saved cv final/coffee/frame_13.jpg
Saved cv final/coffee/frame_14.jpg
Saved cv final/coffee/frame_15.jpg
Saved cv final/coffee/frame_16.jpg
Saved cv final/coffee/frame_17.jpg
Saved cv final/coffee/frame_18.jpg
Saved cv final/coffee/frame_19.jpg
Saved cv final/coffee/frame_20.jpg
Saved cv final/coffee/frame_21.jpg
Saved cv final/coffee/frame_22.jpg
Saved cv final/coffee/frame_23.jpg
Saved cv final/coffee/frame_24.jpg
Saved cv final/coffee/frame_25.jpg
Saved cv final/coffee/frame_26.jpg
Done extracting frames from /Users/m201903466/cv final/coffee.MOV.
Saved cv final/tea/frame_0.jpg
Saved cv final/tea/frame_1.jpg
Saved cv final/tea/frame_2.jpg
Saved cv final/tea/frame_3.jpg
Saved cv final/tea/frame_4.jpg
Saved cv final/tea/frame_5.jpg
Saved cv final/tea/frame_6.jpg
Saved cv final/tea/frame_7.jpg
Saved cv final/tea/frame_8.jpg
Saved cv final/tea/frame_9.jpg
Saved cv final/tea/frame_10.jpg
Saved cv final/tea/frame_11.jpg
Saved cv final/tea/frame_12.jpg
Saved cv final/tea/frame_13.jpg
Saved cv final/tea/frame_14.jpg
Saved cv final/tea/frame_15.jpg
Saved cv final/tea/frame_16.jpg
Saved cv final/tea/frame_17.jpg
Saved cv final/tea/frame_18.jpg
Saved cv final/tea/frame_19.jpg
Saved cv final/tea/frame_20.jpg
Saved cv final/tea/frame_21.jpg
Saved cv final/tea/frame_22.jpg
Saved cv final/tea/frame_23.jpg
Saved cv final/tea/frame_24.jpg
Saved cv final/tea/frame_25.jpg
Saved cv final/tea/frame_26.jpg
Saved cv final/tea/frame_27.jpg
Saved cv final/tea/frame_28.jpg
Saved cv final/tea/frame_29.jpg
Saved cv final/tea/frame_30.jpg
Saved cv final/tea/frame_31.jpg
Saved cv final/tea/frame_32.jpg
Saved cv final/tea/frame_33.jpg
Saved cv final/tea/frame_34.jpg
Saved cv final/tea/frame_35.jpg
Saved cv final/tea/frame_36.jpg
Saved cv final/tea/frame_37.jpg
Saved cv final/tea/frame_38.jpg
Saved cv final/tea/frame_39.jpg
Saved cv final/tea/frame_40.jpg
Saved cv final/tea/frame_41.jpg
Saved cv final/tea/frame_42.jpg
Saved cv final/tea/frame_43.jpg
Saved cv final/tea/frame_44.jpg
Saved cv final/tea/frame_45.jpg
Done extracting frames from /Users/m201903466/cv final/tea.MOV.
Saved cv final/not/frame_0.jpg
Saved cv final/not/frame_1.jpg
Saved cv final/not/frame_2.jpg
Saved cv final/not/frame_3.jpg
Saved cv final/not/frame_4.jpg
Saved cv final/not/frame_5.jpg
Saved cv final/not/frame_6.jpg
Saved cv final/not/frame_7.jpg
Saved cv final/not/frame_8.jpg
Saved cv final/not/frame_9.jpg
Saved cv final/not/frame_10.jpg
Saved cv final/not/frame_11.jpg
Saved cv final/not/frame_12.jpg
Saved cv final/not/frame_13.jpg
Saved cv final/not/frame_14.jpg
Saved cv final/not/frame_15.jpg
Saved cv final/not/frame_16.jpg
Saved cv final/not/frame_17.jpg
Saved cv final/not/frame_18.jpg
Saved cv final/not/frame_19.jpg
Saved cv final/not/frame_20.jpg
Saved cv final/not/frame_21.jpg
Saved cv final/not/frame_22.jpg
Saved cv final/not/frame_23.jpg
Saved cv final/not/frame_24.jpg
Saved cv final/not/frame_25.jpg
Saved cv final/not/frame_26.jpg
Done extracting frames from /Users/m201903466/cv final/not.MOV.
```

In [2]:

```
# Locating files location purposes don't mind please!
# Print contents of the directory after cleaning
print("Directory contents after cleaning:")
for root, dirs, files in os.walk('cv final'):
    for name in dirs:
        print(os.path.join(root, name))
```

```
Directory contents after cleaning:
cv final/not
cv final/bring
cv final/tea
cv final/coffee
```

In [9]:

```
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define the base model
base_model = MobileNetV2(weights='imagenet', include_top=False, input_sha

# Add custom layers on top of the base model
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(4, activation='softmax')(x)  # Assuming 4 output clas

# Create the final model
model = Model(inputs=base_model.input, outputs=predictions)

# Freeze all layers of the base model
for layer in base_model.layers:
    layer.trainable = False

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=

# Define data generator
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

train_generator = train_datagen.flow_from_directory(
    'cv final/',
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical'
)

# Define the number of epochs
epochs = 10

# Loop through epochs and train the model
for epoch in range(epochs):
    print(f"Epoch {epoch+1}/{epochs}")
    for batch_images, batch_labels in train_generator:
        model.fit(batch_images, batch_labels)
        break  # Break after one batch for demonstration purposes

print(f"Found {train_generator.samples} images belonging to {train_genera
```

```
Found 157 images belonging to 4 classes.
Epoch 1/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step - accuracy: 0.0938 - loss: 1.5773
Epoch 2/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 307ms/step - accuracy: 0.4688 - loss: 1.8430
Epoch 3/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 276ms/step - accuracy: 0.5000 - loss: 1.7646
Epoch 4/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 279ms/step - accuracy: 0.6875 - loss: 0.8552
Epoch 5/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 1s/step - accuracy: 0.4483 - loss: 1.3517
Epoch 6/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 355ms/step - accuracy: 0.7812 - loss: 0.5114
Epoch 7/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 277ms/step - accuracy: 0.4688 - loss: 1.1867
Epoch 8/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 265ms/step - accuracy: 0.8125 - loss: 0.3707
Epoch 9/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 261ms/step - accuracy: 0.8125 - loss: 0.5954
Epoch 10/10
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 243ms/step - accuracy: 0.9310 - loss: 0.3580
Found 157 images belonging to 4 classes.
```

In [19]:

```
# Save the model in the native Keras format
model.save('custom_model.keras')
```

In [1]:

```
import cv2
from tensorflow.keras.models import load_model
import numpy as np

# Load the saved model
model = load_model('custom_model.keras')  # Replace 'custom_model.keras'

# Open the video file
video_path = 'all signs 2.MOV'  # Replace with the actual path to your vi
video = cv2.VideoCapture(video_path)

# Check if the video opened successfully
if not video.isOpened():
    print(f"Failed to open video: {video_path}")
    exit()
```

```
# Define a dictionary for mapping class indices to class labels
class_labels = {0: 'bring', 1: 'coffee', 2: 'not', 3: 'tea'}

# Get video properties for output video writer
frame_width = int(video.get(3))
frame_height = int(video.get(4))
fps = int(video.get(cv2.CAP_PROP_FPS))

# Define the output video writer
output_video_path = 'output_video_all signs 2.mov'  # Set the path for th
fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # 'mp4v' or 'avc1' should work
output_video = cv2.VideoWriter(output_video_path, fourcc, fps, (frame_wid

# Loop through the frames of the video
while True:
    ret, frame = video.read()
    if not ret:
        break

    # Resize frame for prediction
    frame_for_pred = cv2.resize(frame, (224, 224))  # Make sure to use a
    frame_for_pred = cv2.cvtColor(frame_for_pred, cv2.COLOR_BGR2RGB)  # C
    frame_for_pred = frame_for_pred / 255.0  # Normalize pixel values
    frame_for_pred = np.expand_dims(frame_for_pred, axis=0)

    # Predict
    predictions = model.predict(frame_for_pred)

    # Get the predicted class label
    predicted_class = np.argmax(predictions)
    predicted_label = class_labels[predicted_class]

    # Put the predicted label on the original frame (not the resized one)
    cv2.putText(frame, predicted_label, (10, 50), cv2.FONT_HERSHEY_SIMPLE

    # Display the frame
    cv2.imshow('Video', frame)

    # Write the original frame to the output video (not the resized one)
    output_video.write(frame)

    # Check for key press (press 'q' to exit)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release the video capture and close all windows
video.release()
output_video.release()
cv2.destroyAllWindows()

print(f"Output video saved as: {output_video_path}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 395ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 72ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 69ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 50ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
Output video saved as: output_video_all signs trained.mov
```

Discussion

The system demonstrated robust performance during testing, achieving a high accuracy rate of approximately 100% across various sign language gestures. This success can be attributed to the training dataset, which was carefully curated from a single comprehensive video containing a wide range of sign language expressions. By using the same video for training and testing, the system benefitted from consistent sign presentation and background settings.

However, when testing the system with a new video containing the same signs, certain discrepancies were observed. Specifically, only the signs for "bring" and "not" were consistently recognized, while others showed lower accuracy or were not detected at all. This outcome underscores the system's limitations in handling variations in sign presentation and filming angles beyond the training dataset's scope.

The discrepancy in performance between the original training video and the new test video highlights the need for a more diverse and representative dataset. Future improvements could focus on expanding the dataset to include a broader range of sign language expressions, diverse presentation styles, and varying filming angles. These enhancements aim to improve the system's generalization capabilities and overall accuracy in real-world scenarios.

Despite these challenges, the system's core functionality remains promising, serving as a strong foundation for ongoing development and optimization in automatic sign language transcription applications.

Conclusion

In this project, I developed an automatic sign language transcription system using machine learning techniques. The system processes video input, recognizes signs, and overlays subtitles for enhanced accessibility. Training involved using a custom dataset of sign language videos, preprocessing frames, and applying transfer learning with a pre-trained image classifier. The model achieved good accuracy during testing but faced challenges with variable angle of sign capture. Future work involves refining the model, addressing environmental factors, and expanding the sign language vocabulary for broader utility.

References

MobileNetV2: Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks.

OpenCV Documentation: https://docs.opencv.org/

TensorFlow Documentation: https://www.tensorflow.org/api_docs