lstm-tensorflow-main

November 24, 2024

[2]:
```
import numpy as np
import pandas as pd
```

[ ]:
```
%pip install mediapipe
import cv2
import numpy as np
import mediapipe as mp
```

```
Collecting mediapipe
  Downloading mediapipe-0.10.18-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_
x86_64.whl.metadata (9.7 kB)
Requirement already satisfied: absl-py in /opt/conda/lib/python3.10/site-
packages (from mediapipe) (1.4.0)
Requirement already satisfied: attrs>=19.1.0 in /opt/conda/lib/python3.10/site-
packages (from mediapipe) (23.2.0)
Requirement already satisfied: flatbuffers>=2.0 in
/opt/conda/lib/python3.10/site-packages (from mediapipe) (24.3.25)
Requirement already satisfied: jax in /opt/conda/lib/python3.10/site-packages
(from mediapipe) (0.4.26)
Requirement already satisfied: jaxlib in /opt/conda/lib/python3.10/site-packages
(from mediapipe) (0.4.26.dev20240620)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.10/site-
packages (from mediapipe) (3.7.5)
Requirement already satisfied: numpy<2 in /opt/conda/lib/python3.10/site-
packages (from mediapipe) (1.26.4)
Requirement already satisfied: opencv-contrib-python in
/opt/conda/lib/python3.10/site-packages (from mediapipe) (4.10.0.84)
Collecting protobuf<5,>=4.25.3 (from mediapipe)
  Downloading protobuf-4.25.5-cp37-abi3-manylinux2014_x86_64.whl.metadata (541
bytes)
Collecting sounddevice>=0.4.4 (from mediapipe)
  Downloading sounddevice-0.5.1-py3-none-any.whl.metadata (1.4 kB)
Requirement already satisfied: sentencepiece in /opt/conda/lib/python3.10/site-
packages (from mediapipe) (0.2.0)
Requirement already satisfied: CFFI>=1.0 in /opt/conda/lib/python3.10/site-
packages (from sounddevice>=0.4.4->mediapipe) (1.16.0)
Requirement already satisfied: ml-dtypes>=0.2.0 in
/opt/conda/lib/python3.10/site-packages (from jax->mediapipe) (0.3.2)
```

```
Requirement already satisfied: opt-einsum in /opt/conda/lib/python3.10/site-
packages (from jax->mediapipe) (3.3.0)
Requirement already satisfied: scipy>=1.9 in /opt/conda/lib/python3.10/site-
packages (from jax->mediapipe) (1.14.1)
Requirement already satisfied: contourpy>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->mediapipe) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.10/site-
packages (from matplotlib->mediapipe) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->mediapipe) (4.53.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->mediapipe) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->mediapipe) (21.3)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.10/site-
packages (from matplotlib->mediapipe) (10.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->mediapipe) (3.1.2)
Requirement already satisfied: python-dateutil>=2.7 in
/opt/conda/lib/python3.10/site-packages (from matplotlib->mediapipe)
(2.9.0.post0)
Requirement already satisfied: pycparser in /opt/conda/lib/python3.10/site-
packages (from CFFI>=1.0->sounddevice>=0.4.4->mediapipe) (2.22)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-
packages (from python-dateutil>=2.7->matplotlib->mediapipe) (1.16.0)
Downloading
mediapipe-0.10.18-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(36.1 MB)
                         36.1/36.1 MB
47.1 MB/s eta 0:00:00:00:0100:01
Downloading protobuf-4.25.5-cp37-abi3-manylinux2014_x86_64.whl (294 kB)
                         294.6/294.6 kB
15.2 MB/s eta 0:00:00
Downloading sounddevice-0.5.1-py3-none-any.whl (32 kB)
Installing collected packages: protobuf, sounddevice, mediapipe
  Attempting uninstall: protobuf
    Found existing installation: protobuf 3.20.3
    Uninstalling protobuf-3.20.3:
      Successfully uninstalled protobuf-3.20.3
```

```
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
apache-beam 2.46.0 requires cloudpickle~=2.2.1, but you have cloudpickle 3.0.0
which is incompatible.
apache-beam 2.46.0 requires dill<0.3.2,>=0.3.1.1, but you have dill 0.3.8 which
is incompatible.
apache-beam 2.46.0 requires numpy<1.25.0,>=1.14.3, but you have numpy 1.26.4
which is incompatible.
apache-beam 2.46.0 requires protobuf<4,>3.12.2, but you have protobuf 4.25.5
which is incompatible.
apache-beam 2.46.0 requires pyarrow<10.0.0,>=3.0.0, but you have pyarrow 16.1.0
which is incompatible.
google-cloud-aiplatform 0.6.0a1 requires google-api-
core[grpc]<2.0.0dev,>=1.22.2, but you have google-api-core 2.11.1 which is
incompatible.
google-cloud-automl 1.0.1 requires google-api-core[grpc]<2.0.0dev,>=1.14.0, but
you have google-api-core 2.11.1 which is incompatible.
google-cloud-bigquery 2.34.4 requires protobuf<4.0.0dev,>=3.12.0, but you have
protobuf 4.25.5 which is incompatible.
google-cloud-bigtable 1.7.3 requires protobuf<4.0.0dev, but you have protobuf
4.25.5 which is incompatible.
google-cloud-vision 2.8.0 requires protobuf<4.0.0dev,>=3.19.0, but you have
protobuf 4.25.5 which is incompatible.
kfp 2.5.0 requires google-cloud-storage<3,>=2.2.1, but you have google-cloud-
storage 1.44.0 which is incompatible.
kfp 2.5.0 requires protobuf<4,>=3.13.0, but you have protobuf 4.25.5 which is
incompatible.
kfp-pipeline-spec 0.2.2 requires protobuf<4,>=3.13.0, but you have protobuf
4.25.5 which is incompatible.
tensorflow-metadata 0.14.0 requires protobuf<4,>=3.7, but you have protobuf
4.25.5 which is incompatible.
tensorflow-transform 0.14.0 requires protobuf<4,>=3.7, but you have protobuf
4.25.5 which is incompatible.
Successfully installed mediapipe-0.10.18 protobuf-4.25.3 sounddevice-0.5.1
```

Note: you may need to restart the kernel to use updated packages.

[3]:
```
mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_holistic = mp.solutions.holistic
```

[4]:
```
def extract_holistic_landmarks(frame, holistic):
    rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    results = holistic.process(rgb_frame)

    hand_landmarks = []
    face_landmarks = []
    pose_landmarks = []

    if results.left_hand_landmarks:
        hand_landmarks.extend([(lm.x, lm.y, lm.z) for lm in results.
 ↪left_hand_landmarks.landmark])
    if results.right_hand_landmarks:
        hand_landmarks.extend([(lm.x, lm.y, lm.z) for lm in results.
 ↪right_hand_landmarks.landmark])

    if results.face_landmarks:
        face_landmarks.extend([(lm.x, lm.y, lm.z) for lm in results.
 ↪face_landmarks.landmark])

    if results.pose_landmarks:
        pose_landmarks.extend([(lm.x, lm.y, lm.z) for lm in results.
 ↪pose_landmarks.landmark])

    all_landmarks = {
        'hand_landmarks': np.array(hand_landmarks) if hand_landmarks else None,
        'face_landmarks': np.array(face_landmarks) if face_landmarks else None,
        'pose_landmarks': np.array(pose_landmarks) if pose_landmarks else None,
    }

    return all_landmarks
```

[5]:
```
def process_videos(root_folder):
    holistic = mp_holistic.Holistic(static_image_mode=False,␣
 ↪min_detection_confidence=0.5)

    dataset = []
    labels = []

    sign_folders = os.listdir(root_folder)
    print(sign_folders)
```

```
    for sign_folder in sign_folders:
        sign_path = os.path.join(root_folder, sign_folder)
        print(f'Processing subcategory: {sign_folder}')

        if os.path.isdir(sign_path):
            video_files = [filename for filename in os.listdir(sign_path)
                           if filename.endswith(('.MOV', '.mp4', '.avi', '.
↪mkv', '.wmv', '.flv', '.webm'))]

            current_video_count = len(video_files)
            print(f'Number of videos files in {sign_folder}:␣
↪{current_video_count}')

            for video_file in video_files:
                video_path = os.path.join(sign_path, video_file)
                print(f'Processing video: {video_path}')

                cap = cv2.VideoCapture(video_path)

                while cap.isOpened():
                    ret, frame = cap.read()
                    if not ret:
                        break

                    landmarks = extract_holistic_landmarks(frame, holistic)
                    if (landmarks['hand_landmarks'] is not None or
                        landmarks['face_landmarks'] is not None or
                        landmarks['pose_landmarks'] is not None):
                        combined_landmarks = []
                        if landmarks['hand_landmarks'] is not None:
                            combined_landmarks.
↪extend(landmarks['hand_landmarks'])
                        if landmarks['face_landmarks'] is not None:
                            combined_landmarks.
↪extend(landmarks['face_landmarks'])
                        if landmarks['pose_landmarks'] is not None:
                            combined_landmarks.
↪extend(landmarks['pose_landmarks'])

                        dataset.append(combined_landmarks)
                        labels.append(sign_folder)

                cap.release()


    dataset = np.array(dataset, dtype=object)
    labels = np.array(labels)
```

```
    np.save('holistic_landmarks_Greetings_1of2.npy', dataset)
    np.save('holistic_landmarks_labels_Greetings_1of2', labels)

    holistic.close()
```

[6]:
```
process_videos('/kaggle/input/include/Greetings_1of2/Greetings')
```

```
['51. Good Morning', '50. Alright', '52. Good afternoon', '49. How are you',
'48. Hello']
Processing subcategory: 51. Good Morning
Number of videos files in 51. Good Morning: 21
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_0047.MOV

INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
WARNING: All log messages before absl::InitializeLog() is called are written to
STDERR
W0000 00:00:1731229625.636685     106 inference_feedback_manager.cc:114]
Feedback manager requires a model with a single signature inference. Disabling
support for feedback tensors.
W0000 00:00:1731229625.696535     108 inference_feedback_manager.cc:114]
Feedback manager requires a model with a single signature inference. Disabling
support for feedback tensors.
W0000 00:00:1731229625.703755     109 inference_feedback_manager.cc:114]
Feedback manager requires a model with a single signature inference. Disabling
support for feedback tensors.
W0000 00:00:1731229625.703755     108 inference_feedback_manager.cc:114]
Feedback manager requires a model with a single signature inference. Disabling
support for feedback tensors.
W0000 00:00:1731229625.705284     106 inference_feedback_manager.cc:114]
Feedback manager requires a model with a single signature inference. Disabling
support for feedback tensors.
W0000 00:00:1731229625.722053     109 inference_feedback_manager.cc:114]
Feedback manager requires a model with a single signature inference. Disabling
support for feedback tensors.
W0000 00:00:1731229625.739019     107 landmark_projection_calculator.cc:186]
Using NORM_RECT without IMAGE_DIMENSIONS is only supported for the square ROI.
Provide IMAGE_DIMENSIONS or use PROJECTION_MATRIX.
W0000 00:00:1731229625.739350     108 inference_feedback_manager.cc:114]
Feedback manager requires a model with a single signature inference. Disabling
support for feedback tensors.
W0000 00:00:1731229625.747833     106 inference_feedback_manager.cc:114]
Feedback manager requires a model with a single signature inference. Disabling
support for feedback tensors.

Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_9934.MOV
```

```
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_9935.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_0048.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_0099.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_9970.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_9993.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_0045.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_0098.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_9991.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_0100.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_0043.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_0046.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_9932.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_9968.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_9992.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_9933.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_0042.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_0044.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_9971.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/51. Good
Morning/MVI_9969.MOV
Processing subcategory: 50. Alright
Number of videos files in 50. Alright: 21
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_9988.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_0097.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_9990.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_0037.MOV
```

```
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_0096.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_0040.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_0045.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_9925.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_9923.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_9963.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_0043.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_9965.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_9989.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_0038.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_0095.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_9966.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_9926.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_0044.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_0039.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_9924.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/50.
Alright/MVI_9964.MOV
Processing subcategory: 52. Good afternoon
Number of videos files in 52. Good afternoon: 22
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_9938.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_0049.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_0047.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_0050.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_0048.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_0050_ (2).MOV
```

```
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_0102.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_0101.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_9936.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_9973.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_0046.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_9975.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_9937.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_0050_ (1).MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_9974.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_0051.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_9994.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_0103.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_9995.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_9972.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_9996.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/52. Good
afternoon/MVI_9939.MOV
Processing subcategory: 49. How are you
Number of videos files in 49. How are you: 21
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_0041.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_9986.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_0033.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_9921.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_0094.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_9960.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_0040.MOV
```

```
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_9985.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_9919.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_9987.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_0034.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_9959.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_0035.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_9962.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_9918.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_0093.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_0042.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_0092.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_9920.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_9961.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/49. How are
you/MVI_0036.MOV
Processing subcategory: 48. Hello
Number of videos files in 48. Hello: 21
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_9954.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_9957.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_0031.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_0091.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_0029.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_9915.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_0037.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_9982.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_9917.MOV
```

```
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_0030.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_0038.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_9916.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_0032.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_0090.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_9956.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_0039.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_0089.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_9983.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_9984.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_9955.MOV
Processing video: /kaggle/input/include/Greetings_1of2/Greetings/48.
Hello/MVI_9914.MOV
```

1 Training Dataset Importing

[ ]:
```python
landmarks_file = 'holistic_landmarks_Greetings_1of2.npy'
labels_file = 'holistic_landmarks_labels_Greetings_1of2.npy'

# Loading the numpy arrays
x = np.load(landmarks_file, allow_pickle=True)
y = np.load(labels_file)

print(f'Landmarks shape: {x.shape}')
print(f'Labels shape: {y.shape}')
```

```
Landmarks shape: (7278,)
Labels shape: (7278,)
```

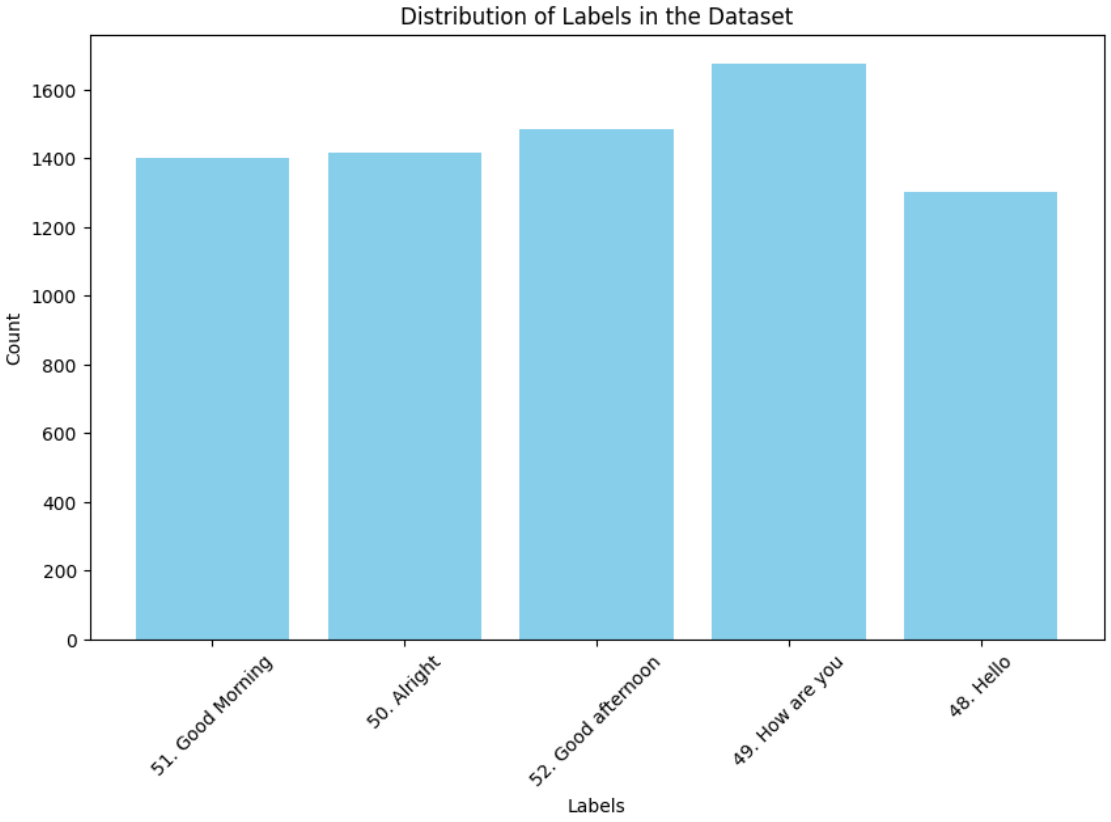
[4]:
```python
import matplotlib.pyplot as plt
from collections import Counter

# Count the occurrences of each label
label_counts = Counter(y)

labels, counts = zip(*label_counts.items())
```

```
plt.figure(figsize=(10, 6))
plt.bar(labels, counts, color='skyblue')
plt.xlabel('Labels')
plt.ylabel('Count')
plt.title('Distribution of Labels in the Dataset')
plt.xticks(rotation=45)
plt.show()

print("Label Distribution:")
for label, count in label_counts.items():
    print(f'Label: {label}, Count: {count}')
```




```
Label Distribution:
Label: 51. Good Morning, Count: 1400
Label: 50. Alright, Count: 1417
Label: 52. Good afternoon, Count: 1485
Label: 49. How are you, Count: 1676
Label: 48. Hello, Count: 1300
```

[5]:
```
# Check for missing items in landmarks
missing_landmarks = []
for i, landmark in enumerate(x):
    if isinstance(landmark, np.ndarray):
        if np.any(np.isnan(landmark)):
            missing_landmarks.append(i)

missing_labels = [i for i, label in enumerate(y) if label is None or label ==␣
 ↪'']

num_missing_landmarks = len(missing_landmarks)
num_missing_labels = len(missing_labels)

print(f'Total missing landmarks: {num_missing_landmarks}')
print(f'Total missing labels: {num_missing_labels}')

print(f'Indices of missing landmarks: {missing_landmarks}')
print(f'Indices of missing labels: {missing_labels}')
```

```
Total missing landmarks: 0
Total missing labels: 0
Indices of missing landmarks: []
Indices of missing labels: []
```

2 Data Processing

[6]:
```
# Checking the shape of each landmark in x to find inconsistencies
landmark_shapes = [np.array(landmark).shape for landmark in x if␣
 ↪isinstance(landmark, list)]

# Identify the unique shapes
unique_shapes = set(landmark_shapes)
print(f"Unique shapes found: {unique_shapes}")

# Fixing inconsistent landmarks by padding or truncating
fixed_landmarks = []
for landmark in x:
    if isinstance(landmark, list):
        landmark_array = np.array(landmark)
        if landmark_array.shape[0] < 543:
            # Pad with zeros if less than 543
            padded = np.pad(landmark_array, ((0, 543 - landmark_array.
 ↪shape[0]), (0, 0)), mode='constant')
            fixed_landmarks.append(padded)
        elif landmark_array.shape[0] > 543:
            # Truncate if greater than 543
```

```
            truncated = landmark_array[:543]
            fixed_landmarks.append(truncated)
        else:
            # Append as is if it has the correct shape
            fixed_landmarks.append(landmark_array)

x_fixed = np.array(fixed_landmarks)
print(f"Fixed landmarks shape: {x_fixed.shape}")
```

```
Unique shapes found: {(543, 3), (522, 3), (501, 3)}
Fixed landmarks shape: (7278, 543, 3)
```

[7]:
```
from sklearn.preprocessing import StandardScaler

# Normalizing the data
scaler = StandardScaler()
landmarks_data_scaled = scaler.fit_transform(x_fixed.reshape(-1, x_fixed.
 ↪shape[-1]))  # Flatten for scaling
landmarks_data_scaled = landmarks_data_scaled.reshape(x_fixed.shape)  # Reshape␣
 ↪back to original
print(f'Scaled landmarks shape: {landmarks_data_scaled.shape}')
```

```
Scaled landmarks shape: (7278, 543, 3)
```

[8]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(landmarks_data_scaled, y,␣
 ↪test_size=0.2, random_state=42)

print(f"x_train shape: {x_train.shape}")
print(f"x_test shape: {x_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
x_train shape: (5822, 543, 3)
x_test shape: (1456, 543, 3)
y_train shape: (5822,)
y_test shape: (1456,)
```

3 Artifically Creating Data using SMOTE and then Train and Test Split

[9]:
```
from imblearn.over_sampling import SMOTE

# Oversample the minority classes using SMOTE
smote = SMOTE()
```

```
x_train_resampled, y_train_resampled = smote.fit_resample(x_train.
 ↪reshape((x_train.shape[0], -1)), y_train)
x_train_resampled = x_train_resampled.reshape((x_train_resampled.shape[0],␣
 ↪x_train.shape[1], x_train.shape[2]))

print(f"x_train shape: {x_train_resampled.shape}")
print(f"x_test shape: {x_test.shape}")
print(f"y_train shape: {y_train_resampled.shape}")
print(f"y_test shape: {y_test.shape}")
```

```
x_train shape: (6755, 543, 3)
x_test shape: (1456, 543, 3)
y_train shape: (6755,)
y_test shape: (1456,)
```

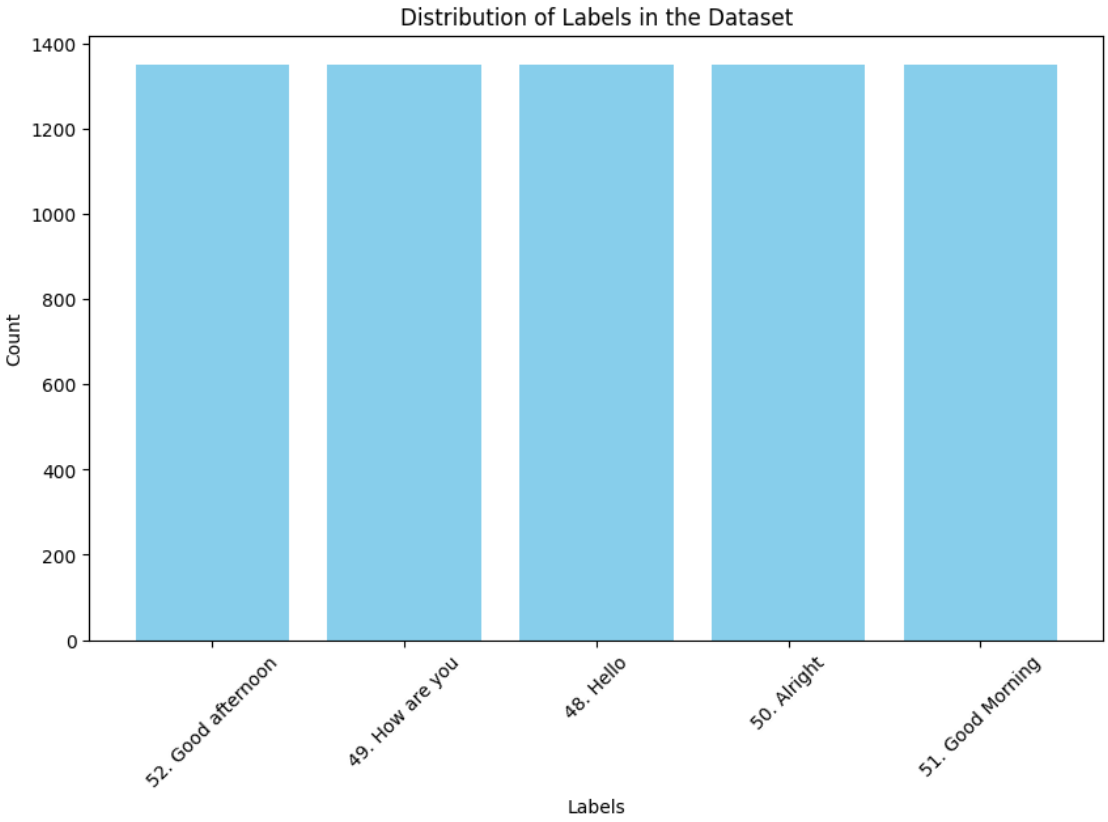
[10]:
```
import matplotlib.pyplot as plt
from collections import Counter

# The occurrences of each label
label_counts = Counter(y_train_resampled)

# The unique labels and their counts
labels, counts = zip(*label_counts.items())

plt.figure(figsize=(10, 6))
plt.bar(labels, counts, color='skyblue')
plt.xlabel('Labels')
plt.ylabel('Count')
plt.title('Distribution of Labels in the Dataset')
plt.xticks(rotation=45)
plt.show()

# The counts
print("Label Distribution of Y Training:")
for label, count in label_counts.items():
    print(f'Label: {label}, Count: {count}')
```




```
Label Distribution of Y Training:
Label: 52. Good afternoon, Count: 1351
Label: 49. How are you, Count: 1351
Label: 48. Hello, Count: 1351
Label: 50. Alright, Count: 1351
Label: 51. Good Morning, Count: 1351
```

4 Data Preparing (For Y as X has already been prepared)

[11]:
```python
#Label encoding
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()

y_train_encoded = label_encoder.fit_transform(y_train_resampled)
y_test_encoded = label_encoder.transform(y_test)

print(f"Encoded y_train: {y_train_encoded[:5]}")
print(f"Encoded y_test: {y_test_encoded[:5]}")
```

```
label_mapping = dict(zip(label_encoder.classes_, label_encoder.
  ↪transform(label_encoder.classes_)))
print(f"Label Mapping: {label_mapping}")
```

```
Encoded y_train: [4 1 0 1 1]
Encoded y_test: [0 0 0 4 1]
Label Mapping: {'48. Hello': 0, '49. How are you': 1, '50. Alright': 2, '51.
Good Morning': 3, '52. Good afternoon': 4}
```

[12]:
```
from collections import Counter

print(f"y_train_encoded distribution: {Counter(y_train_encoded)}")
print(f"y_test_encoded distribution: {Counter(y_test_encoded)}")
```

```
y_train_encoded distribution: Counter({4: 1351, 1: 1351, 0: 1351, 2: 1351, 3:
1351})
y_test_encoded distribution: Counter({1: 325, 2: 294, 3: 289, 4: 286, 0: 262})
```

[13]:
```
print(f"x_train shape: {x_train_resampled.shape}")
print(f"x_test shape: {x_test.shape}")
print(f"y_train shape: {y_train_encoded.shape}")
print(f"y_test shape: {y_test_encoded.shape}")
```

```
x_train shape: (6755, 543, 3)
x_test shape: (1456, 543, 3)
y_train shape: (6755,)
y_test shape: (1456,)
```

5 Model Arch

[22]:
```
from keras.models import Model
from keras.layers import LSTM, Dense, Attention, Flatten, Input, Bidirectional,␣
  ↪Dropout, Conv1D, MaxPooling1D
from keras.optimizers import Adam

# Defining input shape (length, features)
input_shape = (x_train_resampled.shape[1], x_train_resampled.shape[2])

# CNN + LSTM Model Architecture
input_layer = Input(shape=input_shape)
conv1 = Conv1D(64, kernel_size=3, activation='relu')(input_layer)
pool1 = MaxPooling1D(pool_size=2)(conv1)

# LSTM Layers
lstm_out = Bidirectional(LSTM(128, return_sequences=True))(pool1)

# Attention Layer
```

```
attention_out = Attention()([lstm_out, lstm_out])

# Flatten and Dense Layers
flattened = Flatten()(attention_out)
dense1 = Dense(64, activation='relu')(flattened)
output_layer = Dense(len(np.unique(y_train_encoded)),␣
 ↪activation='softmax')(dense1)

model = Model(inputs=input_layer, outputs=output_layer)
model.compile(optimizer=Adam(), loss='sparse_categorical_crossentropy',␣
 ↪metrics=['accuracy'])

#model.summary()
```

6 Training the model on Covo + Bi Direction LSTM + Attention

[20]:
```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
early_stopping = EarlyStopping(monitor='val_loss', patience=5,␣
 ↪restore_best_weights=True)

# Training the model for 120 epochs
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3,␣
 ↪min_lr=1e-6)
history = model.fit(
    x_train_resampled,
    y_train_encoded,
    validation_data=(x_test, y_test_encoded),
    epochs=120,
    batch_size=64,
    callbacks = [early_stopping, reduce_lr]
)

if early_stopping.stopped_epoch > 0:
    print(f"Training stopped early at epoch: {early_stopping.stopped_epoch +␣
 ↪1}")
    print(f"Weights restored from epoch: {early_stopping.best_epoch + 1}")
else:
    print("Training completed without early stopping.")
```

```
Epoch 1/120
106/106             9s 47ms/step -
accuracy: 0.3212 - loss: 1.6525 - val_accuracy: 0.4107 - val_loss: 1.1417 -
learning_rate: 0.0010
Epoch 2/120
106/106             5s 43ms/step -
accuracy: 0.4800 - loss: 1.1105 - val_accuracy: 0.5371 - val_loss: 1.0158 -
```

```
learning_rate: 0.0010
Epoch 3/120
106/106             5s 43ms/step -
accuracy: 0.5342 - loss: 1.0280 - val_accuracy: 0.5776 - val_loss: 0.8831 -
learning_rate: 0.0010
Epoch 4/120
106/106             5s 43ms/step -
accuracy: 0.5959 - loss: 0.8917 - val_accuracy: 0.6133 - val_loss: 0.8290 -
learning_rate: 0.0010
Epoch 5/120
106/106             5s 43ms/step -
accuracy: 0.6197 - loss: 0.8356 - val_accuracy: 0.6030 - val_loss: 0.8653 -
learning_rate: 0.0010
Epoch 6/120
106/106             5s 43ms/step -
accuracy: 0.6283 - loss: 0.7909 - val_accuracy: 0.6264 - val_loss: 0.8418 -
learning_rate: 0.0010
Epoch 7/120
106/106             5s 43ms/step -
accuracy: 0.6611 - loss: 0.7426 - val_accuracy: 0.6394 - val_loss: 0.7723 -
learning_rate: 0.0010
Epoch 8/120
106/106             5s 43ms/step -
accuracy: 0.6654 - loss: 0.7369 - val_accuracy: 0.6504 - val_loss: 0.7264 -
learning_rate: 0.0010
Epoch 9/120
106/106             5s 43ms/step -
accuracy: 0.6789 - loss: 0.6933 - val_accuracy: 0.6930 - val_loss: 0.6924 -
learning_rate: 0.0010
Epoch 10/120
106/106             5s 43ms/step -
accuracy: 0.6872 - loss: 0.6813 - val_accuracy: 0.6889 - val_loss: 0.6745 -
learning_rate: 0.0010
Epoch 11/120
106/106             5s 43ms/step -
accuracy: 0.7050 - loss: 0.6539 - val_accuracy: 0.6628 - val_loss: 0.7211 -
learning_rate: 0.0010
Epoch 12/120
106/106             5s 43ms/step -
accuracy: 0.7265 - loss: 0.6286 - val_accuracy: 0.6683 - val_loss: 0.7028 -
learning_rate: 0.0010
Epoch 13/120
106/106             5s 43ms/step -
accuracy: 0.7250 - loss: 0.6111 - val_accuracy: 0.7315 - val_loss: 0.5993 -
learning_rate: 0.0010
Epoch 14/120
106/106             5s 43ms/step -
accuracy: 0.7411 - loss: 0.5863 - val_accuracy: 0.7157 - val_loss: 0.6160 -
```

```
learning_rate: 0.0010
Epoch 15/120
106/106             5s 43ms/step -
accuracy: 0.7455 - loss: 0.5635 - val_accuracy: 0.7102 - val_loss: 0.6738 -
learning_rate: 0.0010
Epoch 16/120
106/106             5s 43ms/step -
accuracy: 0.7453 - loss: 0.5875 - val_accuracy: 0.7404 - val_loss: 0.5842 -
learning_rate: 0.0010
Epoch 17/120
106/106             5s 43ms/step -
accuracy: 0.7624 - loss: 0.5257 - val_accuracy: 0.7699 - val_loss: 0.5376 -
learning_rate: 0.0010
Epoch 18/120
106/106             5s 43ms/step -
accuracy: 0.7710 - loss: 0.5137 - val_accuracy: 0.7768 - val_loss: 0.5244 -
learning_rate: 0.0010
Epoch 19/120
106/106             5s 43ms/step -
accuracy: 0.7877 - loss: 0.4836 - val_accuracy: 0.7548 - val_loss: 0.5569 -
learning_rate: 0.0010
Epoch 20/120
106/106             5s 43ms/step -
accuracy: 0.7875 - loss: 0.4931 - val_accuracy: 0.7967 - val_loss: 0.4811 -
learning_rate: 0.0010
Epoch 21/120
106/106             5s 43ms/step -
accuracy: 0.8008 - loss: 0.4673 - val_accuracy: 0.7734 - val_loss: 0.5484 -
learning_rate: 0.0010
Epoch 22/120
106/106             5s 43ms/step -
accuracy: 0.8121 - loss: 0.4410 - val_accuracy: 0.7727 - val_loss: 0.5191 -
learning_rate: 0.0010
Epoch 23/120
106/106             5s 43ms/step -
accuracy: 0.7911 - loss: 0.4800 - val_accuracy: 0.8111 - val_loss: 0.4456 -
learning_rate: 0.0010
Epoch 24/120
106/106             5s 43ms/step -
accuracy: 0.8206 - loss: 0.4162 - val_accuracy: 0.7637 - val_loss: 0.5265 -
learning_rate: 0.0010
Epoch 25/120
106/106             5s 43ms/step -
accuracy: 0.8161 - loss: 0.4352 - val_accuracy: 0.8077 - val_loss: 0.4478 -
learning_rate: 0.0010
Epoch 26/120
106/106             5s 44ms/step -
accuracy: 0.8258 - loss: 0.4155 - val_accuracy: 0.8180 - val_loss: 0.4261 -
```

```
learning_rate: 0.0010
Epoch 27/120
106/106             5s 43ms/step -
accuracy: 0.8215 - loss: 0.4223 - val_accuracy: 0.8180 - val_loss: 0.4294 -
learning_rate: 0.0010
Epoch 28/120
106/106             5s 43ms/step -
accuracy: 0.8304 - loss: 0.3923 - val_accuracy: 0.8159 - val_loss: 0.4455 -
learning_rate: 0.0010
Epoch 29/120
106/106             5s 43ms/step -
accuracy: 0.8320 - loss: 0.3917 - val_accuracy: 0.7823 - val_loss: 0.5086 -
learning_rate: 0.0010
Epoch 30/120
106/106             5s 43ms/step -
accuracy: 0.8532 - loss: 0.3578 - val_accuracy: 0.8503 - val_loss: 0.3633 -
learning_rate: 5.0000e-04
Epoch 31/120
106/106             5s 43ms/step -
accuracy: 0.8848 - loss: 0.2816 - val_accuracy: 0.8503 - val_loss: 0.3716 -
learning_rate: 5.0000e-04
Epoch 32/120
106/106             5s 43ms/step -
accuracy: 0.8814 - loss: 0.2927 - val_accuracy: 0.8455 - val_loss: 0.3454 -
learning_rate: 5.0000e-04
Epoch 33/120
106/106             5s 43ms/step -
accuracy: 0.8847 - loss: 0.2771 - val_accuracy: 0.8681 - val_loss: 0.3283 -
learning_rate: 5.0000e-04
Epoch 34/120
106/106             5s 43ms/step -
accuracy: 0.8971 - loss: 0.2548 - val_accuracy: 0.8702 - val_loss: 0.3242 -
learning_rate: 5.0000e-04
Epoch 35/120
106/106             5s 43ms/step -
accuracy: 0.9020 - loss: 0.2517 - val_accuracy: 0.8613 - val_loss: 0.3524 -
learning_rate: 5.0000e-04
Epoch 36/120
106/106             5s 43ms/step -
accuracy: 0.8909 - loss: 0.2598 - val_accuracy: 0.8661 - val_loss: 0.3193 -
learning_rate: 5.0000e-04
Epoch 37/120
106/106             5s 43ms/step -
accuracy: 0.9095 - loss: 0.2279 - val_accuracy: 0.8620 - val_loss: 0.3182 -
learning_rate: 5.0000e-04
Epoch 38/120
106/106             5s 43ms/step -
accuracy: 0.8971 - loss: 0.2507 - val_accuracy: 0.8812 - val_loss: 0.2974 -
```

```
learning_rate: 5.0000e-04
Epoch 39/120
106/106             5s 43ms/step -
accuracy: 0.9054 - loss: 0.2434 - val_accuracy: 0.8798 - val_loss: 0.2929 -
learning_rate: 5.0000e-04
Epoch 40/120
106/106             5s 43ms/step -
accuracy: 0.9103 - loss: 0.2183 - val_accuracy: 0.8853 - val_loss: 0.2864 -
learning_rate: 5.0000e-04
Epoch 41/120
106/106             5s 43ms/step -
accuracy: 0.9153 - loss: 0.2144 - val_accuracy: 0.8647 - val_loss: 0.3526 -
learning_rate: 5.0000e-04
Epoch 42/120
106/106             5s 43ms/step -
accuracy: 0.9212 - loss: 0.2058 - val_accuracy: 0.8853 - val_loss: 0.3004 -
learning_rate: 5.0000e-04
Epoch 43/120
106/106             5s 43ms/step -
accuracy: 0.9200 - loss: 0.2150 - val_accuracy: 0.8640 - val_loss: 0.3314 -
learning_rate: 5.0000e-04
Epoch 44/120
106/106             5s 44ms/step -
accuracy: 0.9302 - loss: 0.1811 - val_accuracy: 0.9128 - val_loss: 0.2224 -
learning_rate: 2.5000e-04
Epoch 45/120
106/106             5s 43ms/step -
accuracy: 0.9566 - loss: 0.1370 - val_accuracy: 0.9093 - val_loss: 0.2352 -
learning_rate: 2.5000e-04
Epoch 46/120
106/106             5s 44ms/step -
accuracy: 0.9481 - loss: 0.1396 - val_accuracy: 0.8949 - val_loss: 0.2741 -
learning_rate: 2.5000e-04
Epoch 47/120
106/106             5s 43ms/step -
accuracy: 0.9463 - loss: 0.1448 - val_accuracy: 0.9066 - val_loss: 0.2552 -
learning_rate: 2.5000e-04
Epoch 48/120
106/106             5s 43ms/step -
accuracy: 0.9633 - loss: 0.1185 - val_accuracy: 0.9176 - val_loss: 0.2118 -
learning_rate: 1.2500e-04
Epoch 49/120
106/106             5s 44ms/step -
accuracy: 0.9685 - loss: 0.1088 - val_accuracy: 0.9183 - val_loss: 0.2101 -
learning_rate: 1.2500e-04
Epoch 50/120
106/106             5s 43ms/step -
accuracy: 0.9710 - loss: 0.1016 - val_accuracy: 0.9210 - val_loss: 0.2089 -
```

```
learning_rate: 1.2500e-04
Epoch 51/120
106/106             5s 43ms/step -
accuracy: 0.9690 - loss: 0.0991 - val_accuracy: 0.9190 - val_loss: 0.2129 -
learning_rate: 1.2500e-04
Epoch 52/120
106/106             5s 43ms/step -
accuracy: 0.9684 - loss: 0.1069 - val_accuracy: 0.9238 - val_loss: 0.2066 -
learning_rate: 1.2500e-04
Epoch 53/120
106/106             5s 43ms/step -
accuracy: 0.9689 - loss: 0.0980 - val_accuracy: 0.9224 - val_loss: 0.1974 -
learning_rate: 1.2500e-04
Epoch 54/120
106/106             5s 43ms/step -
accuracy: 0.9697 - loss: 0.0971 - val_accuracy: 0.9286 - val_loss: 0.1976 -
learning_rate: 1.2500e-04
Epoch 55/120
106/106             5s 43ms/step -
accuracy: 0.9770 - loss: 0.0820 - val_accuracy: 0.9176 - val_loss: 0.2128 -
learning_rate: 1.2500e-04
Epoch 56/120
106/106             5s 43ms/step -
accuracy: 0.9705 - loss: 0.0906 - val_accuracy: 0.9169 - val_loss: 0.2091 -
learning_rate: 1.2500e-04
Epoch 57/120
106/106             5s 43ms/step -
accuracy: 0.9733 - loss: 0.0810 - val_accuracy: 0.9382 - val_loss: 0.1861 -
learning_rate: 6.2500e-05
Epoch 58/120
106/106             5s 43ms/step -
accuracy: 0.9764 - loss: 0.0812 - val_accuracy: 0.9348 - val_loss: 0.1903 -
learning_rate: 6.2500e-05
Epoch 59/120
106/106             5s 43ms/step -
accuracy: 0.9782 - loss: 0.0772 - val_accuracy: 0.9389 - val_loss: 0.1829 -
learning_rate: 6.2500e-05
Epoch 60/120
106/106             5s 43ms/step -
accuracy: 0.9781 - loss: 0.0743 - val_accuracy: 0.9334 - val_loss: 0.1865 -
learning_rate: 6.2500e-05
Epoch 61/120
106/106             5s 43ms/step -
accuracy: 0.9786 - loss: 0.0705 - val_accuracy: 0.9306 - val_loss: 0.1898 -
learning_rate: 6.2500e-05
Epoch 62/120
106/106             5s 43ms/step -
accuracy: 0.9794 - loss: 0.0709 - val_accuracy: 0.9354 - val_loss: 0.1853 -
```

```
learning_rate: 6.2500e-05
Epoch 63/120
106/106             5s 43ms/step -
accuracy: 0.9846 - loss: 0.0620 - val_accuracy: 0.9348 - val_loss: 0.1839 -
learning_rate: 3.1250e-05
Epoch 64/120
106/106             5s 43ms/step -
accuracy: 0.9818 - loss: 0.0650 - val_accuracy: 0.9361 - val_loss: 0.1851 -
learning_rate: 3.1250e-05
Training stopped early at epoch: 64
Weights restored from epoch: 59
```

[21]:
```python
training_accuracy = history.history['accuracy']
validation_accuracy = history.history['val_accuracy']

training_loss = history.history['loss']
validation_loss = history.history['val_loss']

print(f"Final Training Accuracy: {training_accuracy[-1]:.4f}")
print(f"Final Validation Accuracy: {validation_accuracy[-1]:.4f}")
```

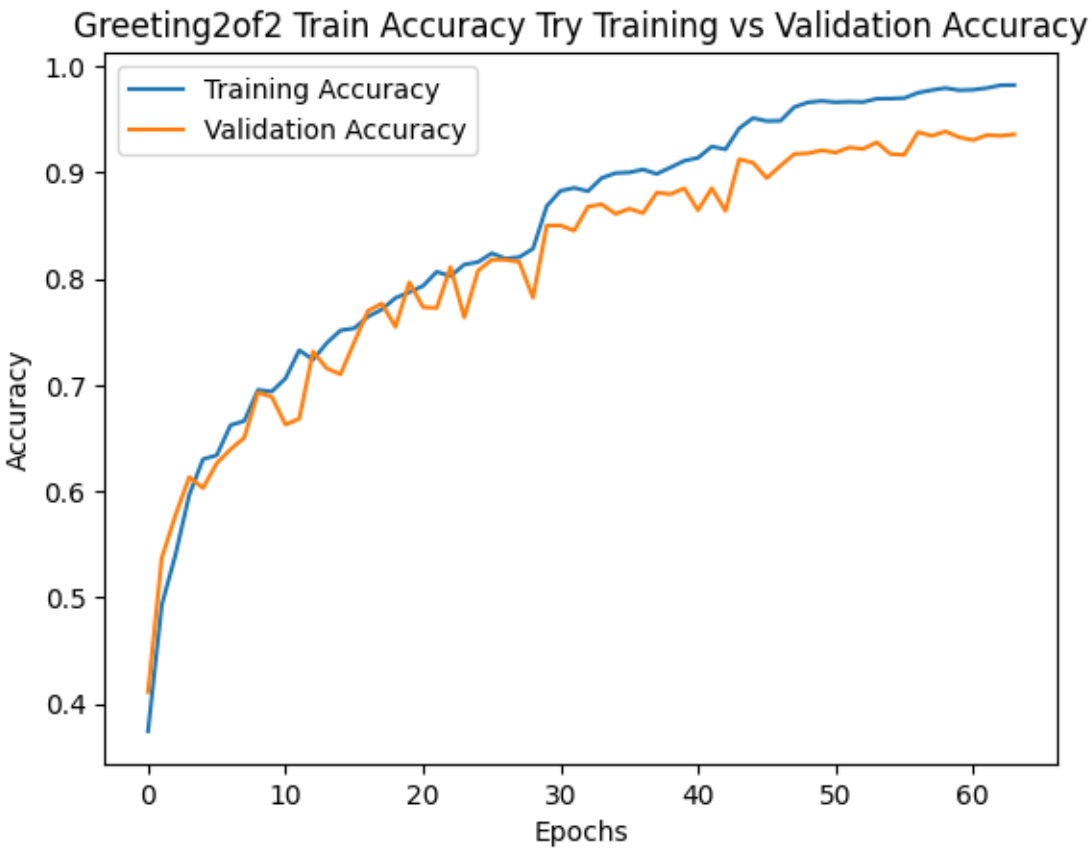
```
Final Training Accuracy: 0.9825
Final Validation Accuracy: 0.9361
```

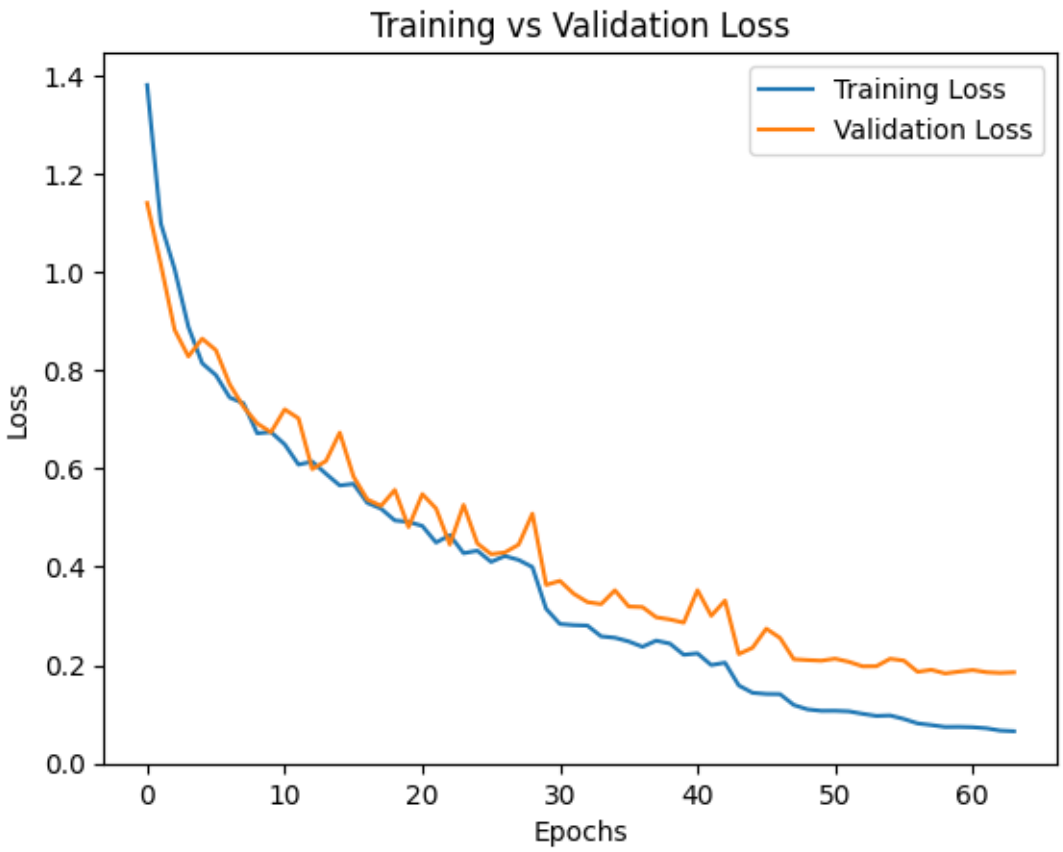
7 Plotting the Model Output

[22]:
```python
import matplotlib.pyplot as plt

plt.plot(training_accuracy, label='Training Accuracy')
plt.plot(validation_accuracy, label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Greeting2of2 Train Accuracy Try Training vs Validation Accuracy')
plt.show()

plt.plot(training_loss, label='Training Loss')
plt.plot(validation_loss, label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training vs Validation Loss')
plt.show()
```


Greeting2of2 Train Accuracy Try Training vs Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
Epochs
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
10
20
30
40
50
60

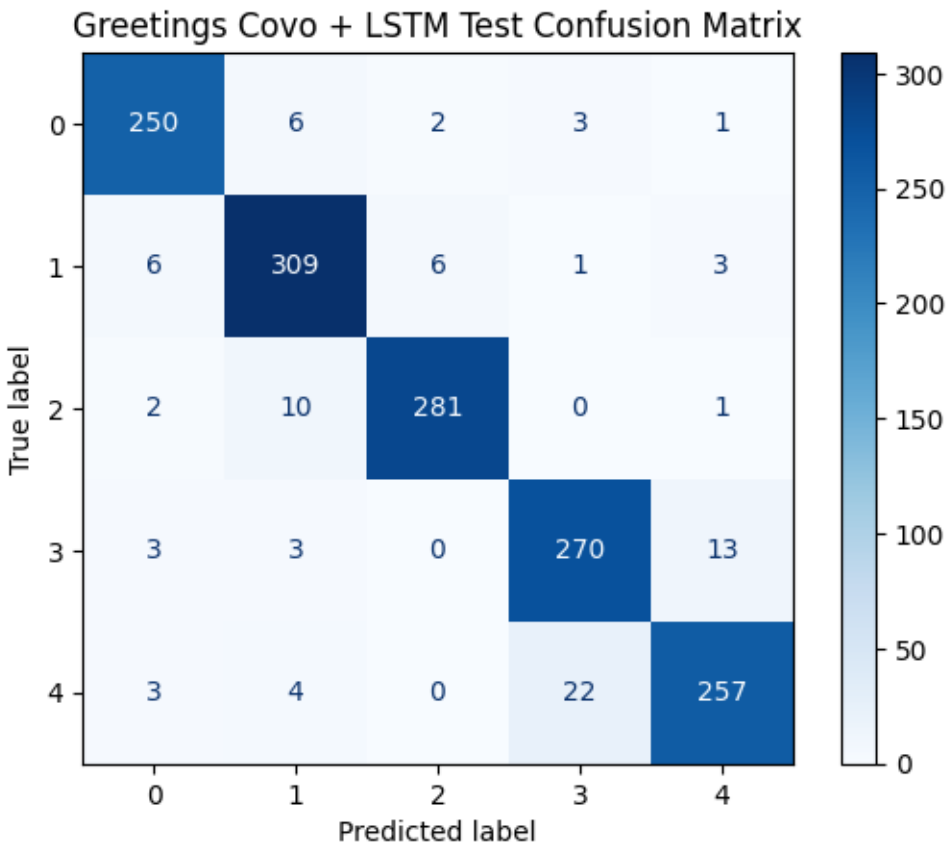


[23]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

y_pred = loaded_model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)
cm = confusion_matrix(y_test_encoded, y_pred_classes)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=np.
 ↪unique(y_train_encoded))
disp.plot(cmap=plt.cm.Blues)
plt.title("Greetings Covo + LSTM Test Confusion Matrix")
plt.show()
```

```
46/46             4s 84ms/step
```

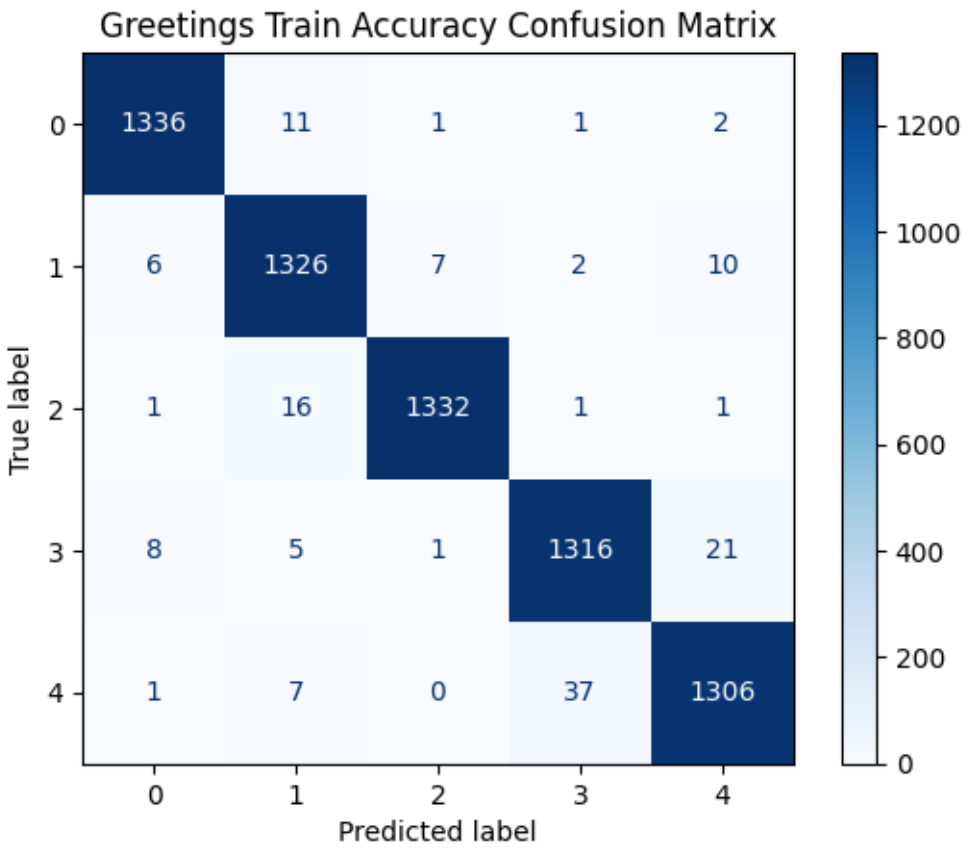


[15]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

y_pred = loaded_model.predict(x_train_resampled)
y_pred_classes = np.argmax(y_pred, axis=1)  # Convert probabilities to class␣
 ↪labels
cm = confusion_matrix(y_train_encoded, y_pred_classes)

disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=np.
 ↪unique(y_train_encoded))
disp.plot(cmap=plt.cm.Blues)
plt.title("Greetings Train Accuracy Confusion Matrix")
plt.show()
```

```
212/212             18s 86ms/step
```



8 Save the Model

[25]:
```
model_name = 'Greeting1of2' + f"{validation_accuracy[-1]:.4f}"
model.save(f"{model_name}.h5")
print(f"Model {model_name} was saved.")
```

```
Model Greeting1of20.9361 was saved.
```

[1]:
```
from tensorflow.keras.models import load_model

loaded_model = load_model('Greeting1of20.9361.h5')
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be
built. `model.compile_metrics` will be empty until you train or evaluate the
model.
```

[28]:
```
loss, accuracy = loaded_model.evaluate(x_test, y_test_encoded, verbose=0)

print(f'Loss: {loss:.4f}')
```

```
print(f'Accuracy: {accuracy:.4f}')
```

```
Loss: 0.1829
Accuracy: 0.9389
```

[25]:
```
from sklearn.metrics import classification_report
import numpy as np

# Predicting on the test data
y_pred = loaded_model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)  # Convert probabilities to class␣
 ↪indices

# Convert y_test_encoded from one-hot encoding to class indices (if necessary)
if len(y_test_encoded.shape) > 1:  # Check if one-hot encoded
    y_test_classes = np.argmax(y_test_encoded, axis=1)
else:
    y_test_classes = y_test_encoded  # Already in class index format

# Class names for the report
class_names = ["good morning", "alright", "good afternoon", "how are you",␣
 ↪"hello"]

# Generate the classification report
report = classification_report(y_test_classes, y_pred_classes,␣
 ↪target_names=class_names)

# Print the classification report
print("Classification Report:")
print(report)
```

```
46/46             4s 81ms/step
Classification Report:
                precision    recall  f1-score   support

  good morning       0.95      0.95      0.95       262
       alright       0.93      0.95      0.94       325
good afternoon       0.97      0.96      0.96       294
   how are you       0.91      0.93      0.92       289
         hello       0.93      0.90      0.92       286

      accuracy                           0.94      1456
     macro avg       0.94      0.94      0.94      1456
  weighted avg       0.94      0.94      0.94      1456
```