**#Importing the libraries**

import tensorflow as tf

import matplotlib.pyplot as plt

import numpy as np

import os

image\_size = 128

batch\_size = 32

channels = 3

epochs = 20

**#Path to the dataset, loading it and the total amount of classes and size of the dataset**

video\_dir = "C:/Users/aditi/Downloads/dynamic\_gestures"

image\_size = 128

batch\_size = 32

num\_frames = 30

def load\_video(video\_path, num\_frames, image\_size):

cap = cv2.VideoCapture(video\_path)

frames = []

total\_frames = int(cap.get(cv2.CAP\_PROP\_FRAME\_COUNT))

frame\_indices = np.linspace(0, total\_frames-1, num\_frames).astype(int)

for idx in frame\_indices:

cap.set(cv2.CAP\_PROP\_POS\_FRAMES, idx)

ret, frame = cap.read()

if not ret:

break

frame = cv2.resize(frame, (image\_size, image\_size))

frame = frame / 255.0 # Normalize

frames.append(frame)

cap.release()

if len(frames) < num\_frames:

frames.extend([frames[-1]] \* (num\_frames - len(frames))) # Pad if needed

return np.array(frames)

def load\_video\_dataset(video\_dir, num\_frames, image\_size):

videos = []

labels = []

class\_names = sorted(os.listdir(video\_dir)) # Get class labels

for class\_index, class\_name in enumerate(class\_names):

class\_path = os.path.join(video\_dir, class\_name)

for video\_file in os.listdir(class\_path):

video\_path = os.path.join(class\_path, video\_file)

video\_data = load\_video(video\_path, num\_frames, image\_size)

videos.append(video\_data)

labels.append(class\_index)

return np.array(videos), np.array(labels), class\_names

X, y, class\_names = load\_video\_dataset(video\_dir, num\_frames, image\_size)

dataset = tf.data.Dataset.from\_tensor\_slices((X, y))

dataset = dataset.shuffle(len(X)).batch(batch\_size).prefetch(tf.data.AUTOTUNE)

print(f"Dataset size: {len(X)} videos")

print(f"Number of classes: {len(class\_names)}")

**#CNN code using adam optimizer**

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv3D, MaxPooling3D, Flatten, Dense, Dropout, BatchNormalization

num\_classes = len(class\_names)

input\_shape = (num\_frames, image\_size, image\_size, 3)

model = Sequential([

Conv3D(32, (3, 3, 3), activation='relu', padding='same', input\_shape=input\_shape),

MaxPooling3D((2, 2, 2)),

Conv3D(64, (3, 3, 3), activation='relu', padding='same'),

MaxPooling3D((2, 2, 2)),

Conv3D(128, (3, 3, 3), activation='relu', padding='same'),

MaxPooling3D((2, 2, 2)),

Flatten(),

Dense(512, activation='relu'),

Dropout(0.5),

Dense(num\_classes, activation='softmax')

])

model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

model.summary()

**#Model training using 20 epochs**

epochs = 20

batch\_size = 8

history = model.fit(dataset, epochs=epochs, verbose=1)

model.save("isl\_video\_model.h5")

**#Test accuracy of the model**

loss, accuracy = model.evaluate(dataset)

print(f"Test Accuracy: {accuracy:.2f}")

**#Loading the trained model**

from tensorflow.keras.models import load\_model

model = load\_model("C:/Users/aditi/Indian Sign Language/isl\_video\_model.h5")

model.compile(optimizer="adam", loss="categorical\_crossentropy", metrics=["accuracy"])

**#Pre processing the frames which match the dataset video frame size**

def preprocess\_frame(frame):

frame = cv2.resize(frame, (128, 128))

frame = frame / 255.0

frame = np.expand\_dims(frame, axis=0)

frame = np.expand\_dims(frame, axis=1

return frame

**#Code for identifying the gesture “Call”**

import cv2

import numpy as np

from tensorflow.keras.models import load\_model

from googletrans import Translator

model = load\_model("isl\_video\_model.h5")

gesture\_dict = {

0: "Call",

1: "Doctor",

2: "Help"

}

translator = Translator()

def preprocess\_frame(frame, target\_size=(128, 128)):

resized\_frame = cv2.resize(frame, target\_size)

normalized\_frame = resized\_frame / 255.0

return normalized\_frame

def preprocess\_video(video\_path, target\_frames=30, target\_size=(128, 128)):

cap = cv2.VideoCapture(video\_path)

frames = []

while cap.isOpened() and len(frames) < target\_frames:

ret, frame = cap.read()

if not ret:

break

frames.append(preprocess\_frame(frame, target\_size))

cap.release()

while len(frames) < target\_frames:

frames.append(frames[-1])

return np.expand\_dims(np.array(frames), axis=0)

def process\_video\_with\_translation(video\_path, model, gesture\_dict):

input\_data = preprocess\_video(video\_path)

predictions = model.predict(input\_data)

confidence = np.max(predictions)

class\_index = np.argmax(predictions)

THRESHOLD = 0.60

if confidence >= THRESHOLD:

recognized\_text = gesture\_dict.get(class\_index, "Unknown Gesture")

print(f"Recognized Gesture: {recognized\_text} (Confidence: {confidence:.2f})")

try:

translation\_kn = translator.translate(recognized\_text, src="en", dest="kn").text

translation\_hi = translator.translate(recognized\_text, src="en", dest="hi").text

print(f"Kannada: {translation\_kn}")

print(f"Hindi: {translation\_hi}")

except Exception as e:

print(f"Translation Error: {e}")

else:

print("Confidence too low for reliable prediction.")

video\_path = "C:/Users/aditi/Videos/call\_gesture.mp4"

process\_video\_with\_translation(video\_path, model, gesture\_dict)

By changing the path to other gestures such as “Doctor” and “Help”, the model will accurately predict the gesture and convert it into English, Kannada and Hindi. The accuracy is 100%.

**#Front-end code using Streamlit**

import streamlit as st

import cv2

import numpy as np

from tensorflow.keras.models import load\_model

from googletrans import Translator

import tempfile

import os

model = load\_model("isl\_video\_model.h5")

gesture\_dict = {0: "Call", 1: "Doctor", 2: "Help"}

translator = Translator()

def preprocess\_frame(frame, target\_size=(128, 128)):

    resized = cv2.resize(frame, target\_size)

    return resized / 255.0

def preprocess\_video(path, target\_frames=30):

    cap = cv2.VideoCapture(path)

    frames = []

    while cap.isOpened() and len(frames) < target\_frames:

        ret, frame = cap.read()

        if not ret:

            break

        frames.append(preprocess\_frame(frame))

    cap.release()

    while len(frames) < target\_frames:

        frames.append(frames[-1])  # Padding

    return np.expand\_dims(np.array(frames), axis=0)

def predict\_gesture(video\_path):

    input\_data = preprocess\_video(video\_path)

    predictions = model.predict(input\_data)

    confidence = np.max(predictions)

    class\_index = np.argmax(predictions)

    if confidence >= 0.60:

        label = gesture\_dict.get(class\_index, "Unknown")

        trans\_kn = translator.translate(label, src="en", dest="kn").text

        trans\_hi = translator.translate(label, src="en", dest="hi").text

        return label, trans\_kn, trans\_hi, confidence

    else:

        return "Low Confidence", "", "", confidence

st.title("Indian Sign Language To Multilingual Texts")

uploaded\_video = st.file\_uploader("Upload a gesture video", type=["mp4", "avi", "mov"])

if uploaded\_video:

    with tempfile.NamedTemporaryFile(delete=False) as tmp\_file:

        tmp\_file.write(uploaded\_video.read())

        video\_path = tmp\_file.name

    st.video(video\_path)

    if st.button("Predict Gesture"):

        label, kn, hi, conf = predict\_gesture(video\_path)

        st.success(f"Predicted Gesture: {label} (Confidence: {conf:.2f})")

        if kn:

            st.write(f"Kannada: {kn}")

            st.write(f"Hindi: {hi}")

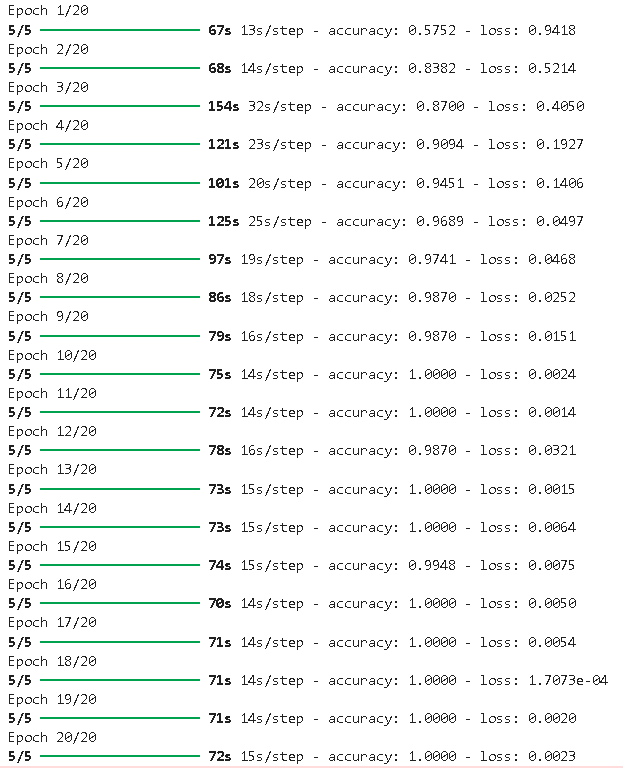
**#Code for running the frontend using streamlit**

streamlit run app\_isl.py

SCREENSHOTS FOR THE PROJECT



**Fig: The dataset size and number of classes**



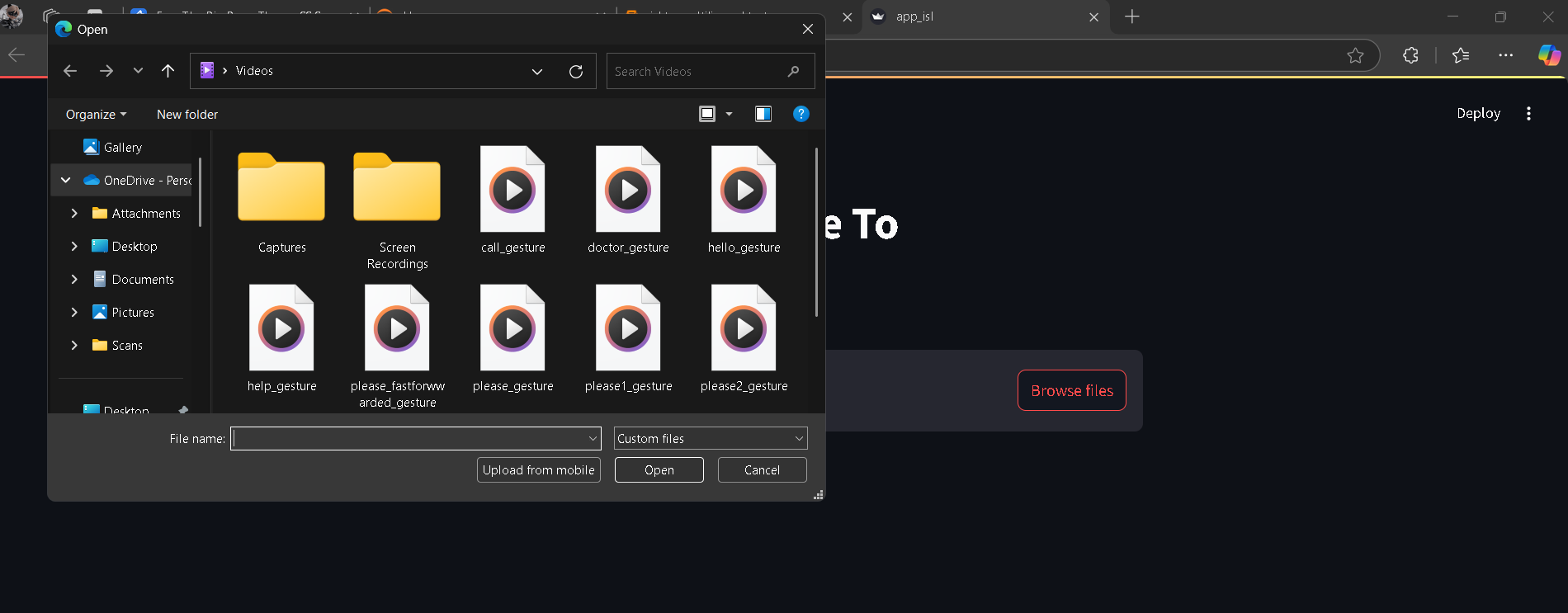
**Fig: Model training over 20 epochs**



**Fig: Test accuracy from the trained model**



**Fig: Front end using streamlit**



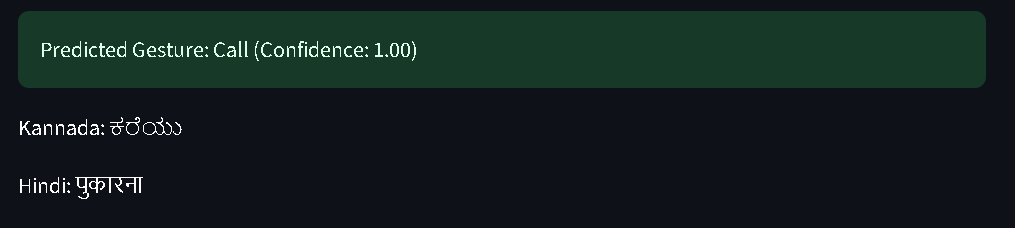
**Fig: Selecting an mp4 file from the web page**



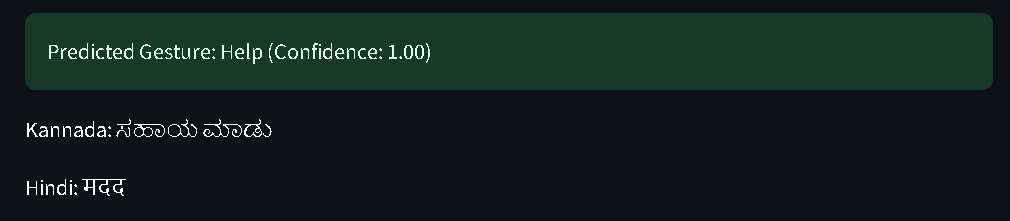
**Fig: Gesture which is to be predicted**



**Fig: The gesture is predicted and then converted into multilingual texts**

****

**Fig: The predicted output and its related multilingual texts**

****

**Fig: The predicted output and its related multilingual texts**



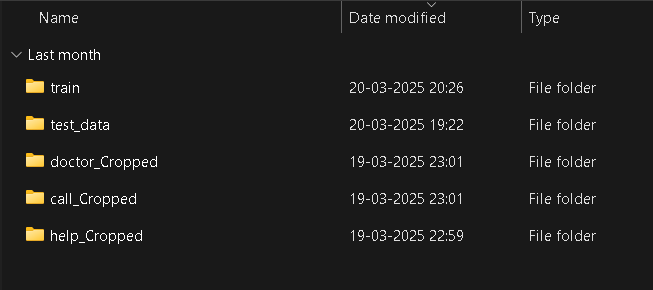
**Fig: Gesture for “Help”**



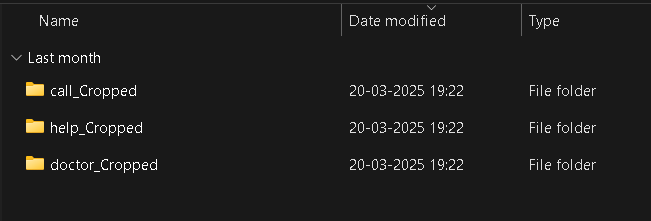
**Fig: Gesture for “Doctor”**



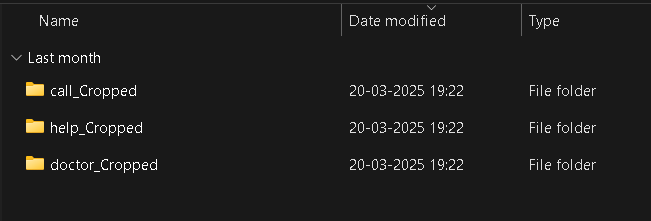
**Fig: Gesture for “Call”**

****

**Fig: The gestures, train and test data**

****

**Fig: The data for the train data**

****

**Fig: The data for the test data**