Code :

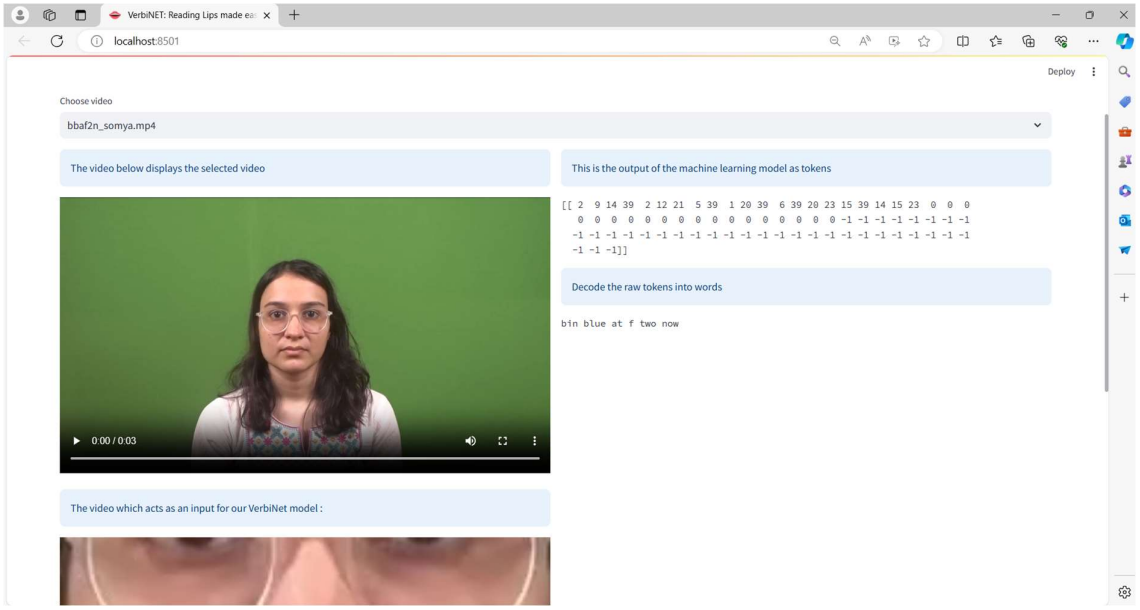
```
import os
import cv2
import tensorflow as tf
import numpy as np
from typing import List
from matplotlib import pyplot as plt

# import dlib
# from imutils import face_utils
def load_video_and_get_bounding_boxes(input_path: str) -> list:
    cap = cv2.VideoCapture(input_path)
    bounding_boxes = []  # Store bounding box coordinates

    detector = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor(datFile)

    while True:
        ret, frame = cap.read()

        if not ret:
            break

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        rects = detector(gray, 1)

        for (i, rect) in enumerate(rects):
            shape = predictor(gray, rect)
            shape = face_utils.shape_to_np(shape)
            jaw_indices = face_utils.FACIAL_LANDMARKS_IDXS['jaw']
```

```
            (i, j) = jaw_indices

            jaw = shape[i:j]
            (x, y, w, h) = cv2.boundingRect(np.array([jaw]))
            bounding_boxes.append((x, y, w, h))  # Append bounding box coordinates

    cap.release()
    return bounding_boxes
def crop_videos_in_directory(input_directory: str, output_directory: str) -> None:
    # Iterate over each file in the input directory
    for filename in os.listdir(input_directory):
        input_file_path = os.path.join(input_directory, filename)
        output_file_path = os.path.join(output_directory, os.path.splitext(filename)[0] + '.mp4')

        # Get bounding box coordinates from the input video
        bounding_boxes = load_video_and_get_bounding_boxes(input_file_path)

        if bounding_boxes:
            x, y, w, h = bounding_boxes[0]  # You may need to adjust the index here
            crop_width = w
            crop_height = h

            cap = cv2.VideoCapture(input_file_path)

            fourcc = cv2.VideoWriter_fourcc(*'mp4v')
            out = cv2.VideoWriter(output_file_path, fourcc, 25.0, (crop_width, crop_height))

            while True:
                ret, frame = cap.read()
                if not ret:
```

```
            break

        cropped_frame = frame[y:y+h, x:x+w]
        out.write(cropped_frame)

    cap.release()
    out.release()
# Get bounding box coordinates from the input video
bounding_boxes = load_video_and_get_bounding_boxes('Abhishek_Data/Research project/trimmed videos/bbbm0n_person7.mp4')

print(bounding_boxes)
import os
import subprocess

# Path to the folder containing MP4 files
input_folder = 'lip_reading_cropped_dataset/s1_cropped'

# Path to the folder where converted MPG files will be saved
output_folder = 'lip_reading_cropped_dataset/s1_cropped_mpg'

# Ensure the output folder exists, create it if not
os.makedirs(output_folder, exist_ok=True)

# Get a list of all MP4 files in the input folder
mp4_files = [f for f in os.listdir(input_folder) if f.endswith('.mp4')]

# Iterate over each MP4 file and convert it to MPG
for mp4_file in mp4_files:
    input_file_path = os.path.join(input_folder, mp4_file)
    output_file_path = os.path.join(output_folder, os.path.splitext(mp4_file)[0] + '.mpg')
```

```
    # FFmpeg command to convert MP4 to MPG
    command = f'ffmpeg -i "{input_file_path}" -c:v mpeg2video -q:v 2 -c:a mp2 -b:a 192k -r 25 -vf "fps=25" "{output_file_path}"'

    # Run the FFmpeg command
    subprocess.run(command, shell=True)

print('Conversion completed.')
model = Sequential()

model.add(Conv3D(128, 3, input_shape=(75, 46,140, 1), padding='same'))
model.add(Activation('relu'))
model.add(MaxPool3D((1, 2, 2)))

model.add(Conv3D(256, 3, padding='same'))
model.add(Activation('relu'))
model.add(MaxPool3D((1, 2, 2)))

model.add(TimeDistributed(Flatten()))

model.add(Bidirectional(LSTM(128, kernel_initializer='Orthogonal', return_sequences=True)))
model.add(Dropout(0.5))

model.add(Dense(char_to_num.vocabulary_size() + 1, kernel_initializer='he_normal', activation='softmax'))
history = model.fit(train, validation_data=test, epochs=30, callbacks=[checkpoint_callback, schedule_callback, example_callback])
```

Interface:


VerbiNET: Reading Lips made ea
localhost:8501
Deploy
Choose video
bbaf2n_somya.mp4
The video below displays the selected video
This is the output of the machine learning model as tokens
[[2 9 14 39 2 12 21 5 39 1 20 39 6 39 20 23 15 39 14 15 23 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
-1 -1 -1]]
Decode the raw tokens into words
bin blue at f two now
0:00 / 0:03
The video which acts as an input for our VerbiNet model :