MP4 to MP3

```
pip install moviepy
```

```
Requirement already satisfied: moviepy in /usr/local/lib/python3.10/dist-packages (1.0.3)
Requirement already satisfied: decorator<5.0,>=4.0.2 in /usr/local/lib/python3.10/dist-packages (from moviepy) (4.4.2)
Requirement already satisfied: tqdm<5.0,>=4.11.2 in /usr/local/lib/python3.10/dist-packages (from moviepy) (4.66.1)
Requirement already satisfied: requests<3.0,>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from moviepy) (2.31.0)
Requirement already satisfied: proglog<=1.0.0 in /usr/local/lib/python3.10/dist-packages (from moviepy) (0.1.10)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from moviepy) (1.23.5)
Requirement already satisfied: imageio<3.0,>=2.5 in /usr/local/lib/python3.10/dist-packages (from moviepy) (2.31.1)
Requirement already satisfied: imageio-ffmpeg>=0.2.0 in /usr/local/lib/python3.10/dist-packages (from moviepy) (0.4.8)
Requirement already satisfied: pillow>=8.3.2 in /usr/local/lib/python3.10/dist-packages (from imageio<3.0,>=2.5->moviepy) (9.4.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3.0,>=2.8.1->moviepy)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3.0,>=2.8.1->moviepy) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3.0,>=2.8.1->moviepy) (2.0.4
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3.0,>=2.8.1->moviepy) (2023.
```

```
from moviepy.editor import *

def mp4tomp3(mp4file,mp3file):
    videoclip=VideoFileClip("/content/mp4.mp4")
    audioclip=videoclip.audio
    audioclip.write_audiofile(mp3file)
    audioclip.close()
    videoclip.close()

mp4tomp3("video.mp4","audio.mp3")
```

```
MoviePy - Writing audio in audio.mp3
                                                                    MoviePy - Done.
```

```
pip install librosa
```

```
Requirement already satisfied: librosa in /usr/local/lib/python3.10/dist-packages (0.10.1)
Requirement already satisfied: audioread>=2.1.9 in /usr/local/lib/python3.10/dist-packages (from librosa) (3.0.0)
Requirement already satisfied: numpy!=1.22.0,!=1.22.1,!=1.22.2,>=1.20.3 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.23.
Requirement already satisfied: scipy>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.11.2)
Requirement already satisfied: scikit-learn>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.2.2)
Requirement already satisfied: joblib>=0.14 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.3.2)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (4.4.2)
Requirement already satisfied: numba>=0.51.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.56.4)
Requirement already satisfied: soundfile>=0.12.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.12.1)
Requirement already satisfied: pooch>=1.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.7.0)
Requirement already satisfied: soxr>=0.3.2 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.3.6)
Requirement already satisfied: typing-extensions>=4.1.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (4.5.0)
Requirement already satisfied: lazy-loader>=0.1 in /usr/local/lib/python3.10/dist-packages (from librosa) (0.3)
Requirement already satisfied: msgpack>=1.0 in /usr/local/lib/python3.10/dist-packages (from librosa) (1.0.5)
Requirement already satisfied: llvmlite<0.40,>=0.39.0dev0 in /usr/local/lib/python3.10/dist-packages (from numba>=0.51.0->librosa) (0.39
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from numba>=0.51.0->librosa) (67.7.2)
Requirement already satisfied: platformdirs>=2.5.0 in /usr/local/lib/python3.10/dist-packages (from pooch>=1.0->librosa) (3.10.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from pooch>=1.0->librosa) (23.1)
Requirement already satisfied: requests>=2.19.0 in /usr/local/lib/python3.10/dist-packages (from pooch>=1.0->librosa) (2.31.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.20.0->librosa) (3.2
Requirement already satisfied: cffi>=1.0 in /usr/local/lib/python3.10/dist-packages (from soundfile>=0.12.1->librosa) (1.15.1)
Requirement already satisfied: pycparser in /usr/local/lib/python3.10/dist-packages (from cffi>=1.0->soundfile>=0.12.1->librosa) (2.21)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.0->l
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.0->librosa) (3.4
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.0->librosa
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.19.0->pooch>=1.0->librosa
```

res

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

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')
```

```
# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/short.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
previous_frame_time = 0
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients
duration = 1.0  # Duration of audio segment in seconds

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    # Adjust this threshold to control the frequency of audio segment extraction
    if current_frame_time - previous_frame_time >= 1000:  # 1-second interval
        previous_frame_time = current_frame_time

        # Extract audio from the current second
        audio_start_time = current_frame_time - 1000  # Adjust for desired segment length
        audio_end_time = current_frame_time

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=duration)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)
        mfccs = mfccs[:, :, np.newaxis]  # Add a channel dimension

        # Predict emotion for the audio segment
        emotions = model.predict(np.expand_dims(mfccs, axis=0))[0]
        predicted_emotion = emotion_labels[np.argmax(emotions)]
        print(f"Emotion: {predicted_emotion}")

# Release video capture
cap.release()
cv2.destroyAllWindows()
```

```
<ipython-input-1-37b75083d150>:40: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=duration)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 34ms/step
Emotion: Fear
<ipython-input-1-37b75083d150>:40: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=duration)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 34ms/step
Emotion: Surprise
<ipython-input-1-37b75083d150>:40: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=duration)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 37ms/step
Emotion: Surprise
```

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

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/sad.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 3    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

        # Pad or trim the MFCCs to match the expected shape (40, 44)
        desired_shape = (40, 44)
        if mfccs.shape[1] < desired_shape[1]:
            mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
        else:
            mfccs = mfccs[:, :desired_shape[1]]

        # Add batch dimension
```

```
        # Add batch dimension
        mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

        # Predict emotion for the audio segment
        emotions = model.predict(mfccs)[0]
        predicted_emotion = emotion_labels[np.argmax(emotions)]
        print(f"Emotion: {predicted_emotion}")

# Release video capture
cap.release()
cv2.destroyAllWindows()
```

```
1/1 [==============================] - 0s 43ms/step
Emotion: Sad
<ipython-input-9-b54b65c2faf3>:43: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 67ms/step
Emotion: Fear
<ipython-input-9-b54b65c2faf3>:43: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 32ms/step
Emotion: Sad
1/1 [==============================] - 0s 23ms/step
Emotion: Fear
<ipython-input-9-b54b65c2faf3>:43: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-9-b54b65c2faf3>:43: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 21ms/step
Emotion: Fear
1/1 [==============================] - 0s 26ms/step
Emotion: Fear
<ipython-input-9-b54b65c2faf3>:43: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-9-b54b65c2faf3>:43: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 21ms/step
Emotion: Fear
1/1 [==============================] - 0s 22ms/step
Emotion: Sad
<ipython-input-9-b54b65c2faf3>:43: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')
```

```
# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/sad.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 3    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

        # Pad or trim the MFCCs to match the expected shape (40, 44)
        desired_shape = (40, 44)
        if mfccs.shape[1] < desired_shape[1]:
            mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
        else:
            mfccs = mfccs[:, :desired_shape[1]]

        # Add batch dimension
        mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

        # Predict emotion for the audio segment
        emotions = model.predict(mfccs)[0]
        predicted_emotion = emotion_labels[np.argmax(emotions)]

        # Update emotion counts
        emotion_counts[predicted_emotion] += 1

# Release video capture
cap.release()
cv2.destroyAllWindows()

# Find the dominant emotion based on counts
dominant_emotion = max(emotion_counts, key=emotion_counts.get)
print(f"Dominant Emotion: {dominant_emotion}")
```

```
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
<ipython-input-10-f800760f6394>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-10-f800760f6394>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
<ipython-input-10-f800760f6394>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-10-f800760f6394>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
<ipython-input-10-f800760f6394>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-10-f800760f6394>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 23ms/step
Dominant Emotion: Sad
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/Mad.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 2    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
```

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

        current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

        if start_frame <= current_frame_time <= end_frame:
            # Extract audio from the current second
            audio_start_time = current_frame_time
            audio_end_time = current_frame_time + 1000  # 1-second interval

            # Extract audio segment
            audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

            # Preprocess audio data (compute MFCCs)
            mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

            # Pad or trim the MFCCs to match the expected shape (40, 44)
            desired_shape = (40, 44)
            if mfccs.shape[1] < desired_shape[1]:
                mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
            else:
                mfccs = mfccs[:, :desired_shape[1]]

            # Add batch dimension
            mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

            # Predict emotion for the audio segment
            emotions = model.predict(mfccs)[0]
            predicted_emotion = emotion_labels[np.argmax(emotions)]

            # Update emotion counts
            emotion_counts[predicted_emotion] += 1

    # Release video capture
    cap.release()
    cv2.destroyAllWindows()

    # Find the dominant emotion based on counts
    dominant_emotion = max(emotion_counts, key=emotion_counts.get)
    print(f"Dominant Emotion: {dominant_emotion}")
```

```
1/1 [==============================] - 0s 22ms/step
<ipython-input-11-d2812e85ea52>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-11-d2812e85ea52>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 22ms/step
Dominant Emotion: Fear
<ipython-input-11-d2812e85ea52>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/Omg.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 3    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

        # Pad or trim the MFCCs to match the expected shape (40, 44)
        desired_shape = (40, 44)
        if mfccs.shape[1] < desired_shape[1]:
            mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
        else:
            mfccs = mfccs[:, :desired_shape[1]]
```

```
            mfccs = mfccs[:, :desired_shape[1]]

        # Add batch dimension
        mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

        # Predict emotion for the audio segment
        emotions = model.predict(mfccs)[0]
        predicted_emotion = emotion_labels[np.argmax(emotions)]

        # Update emotion counts
        emotion_counts[predicted_emotion] += 1

# Release video capture
cap.release()
cv2.destroyAllWindows()

# Find the dominant emotion based on counts
dominant_emotion = max(emotion_counts, key=emotion_counts.get)
print(f"Dominant Emotion: {dominant_emotion}")
```

```
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 26ms/step
<ipython-input-12-95e2ed634736>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-12-95e2ed634736>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 27ms/step
<ipython-input-12-95e2ed634736>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-12-95e2ed634736>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 35ms/step
<ipython-input-12-95e2ed634736>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-12-95e2ed634736>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 21ms/step
<ipython-input-12-95e2ed634736>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-12-95e2ed634736>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 26ms/step
Dominant Emotion: Fear
```

```
import cv2
import librosa
```

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/half sad.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 4    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

        # Pad or trim the MFCCs to match the expected shape (40, 44)
        desired_shape = (40, 44)
        if mfccs.shape[1] < desired_shape[1]:
            mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
        else:
            mfccs = mfccs[:, :desired_shape[1]]

        # Add batch dimension
        mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

        # Predict emotion for the audio segment
        emotions = model.predict(mfccs)[0]
        predicted_emotion = emotion_labels[np.argmax(emotions)]

        # Update emotion counts
        emotion_counts[predicted_emotion] += 1

# Release video capture
cap.release()
cv2.destroyAllWindows()

# Find the dominant emotion based on counts
dominant_emotion = max(emotion_counts, key=emotion_counts.get)
print(f"Dominant Emotion: {dominant_emotion}")
```

```
<ipython-input-13-94218120603b>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-13-94218120603b>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 28ms/step
<ipython-input-13-94218120603b>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 50ms/step
<ipython-input-13-94218120603b>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 33ms/step
<ipython-input-13-94218120603b>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 32ms/step
<ipython-input-13-94218120603b>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-13-94218120603b>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 54ms/step
<ipython-input-13-94218120603b>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 41ms/step
Dominant Emotion: Fear
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/full.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 8    # Adjust the end time as needed
```

```
    # Calculate the corresponding frame numbers
    start_frame = int(start_time * 1000)  # Convert to milliseconds
    end_frame = int(end_time * 1000)

    # Initialize a dictionary to count emotions
    emotion_counts = defaultdict(int)

    # Read and process video frames
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

        if start_frame <= current_frame_time <= end_frame:
            # Extract audio from the current second
            audio_start_time = current_frame_time
            audio_end_time = current_frame_time + 1000  # 1-second interval

            # Extract audio segment
            audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

            # Preprocess audio data (compute MFCCs)
            mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

            # Pad or trim the MFCCs to match the expected shape (40, 44)
            desired_shape = (40, 44)
            if mfccs.shape[1] < desired_shape[1]:
                mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
            else:
                mfccs = mfccs[:, :desired_shape[1]]

            # Add batch dimension
            mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

            # Predict emotion for the audio segment
            emotions = model.predict(mfccs)[0]
            predicted_emotion = emotion_labels[np.argmax(emotions)]

            # Update emotion counts
            emotion_counts[predicted_emotion] += 1

    # Release video capture
    cap.release()
    cv2.destroyAllWindows()

    # Find the dominant emotion based on counts
    dominant_emotion = max(emotion_counts, key=emotion_counts.get)
    print(f"Dominant Emotion: {dominant_emotion}")
```

```
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-14-6ecfa32acf22>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 24ms/step
<ipython-input-14-6ecfa32acf22>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-14-6ecfa32acf22>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 23ms/step
Dominant Emotion: Fear
<ipython-input-14-6ecfa32acf22>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/neutral.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 3    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
```

```
            # Preprocess audio data (compute MFCCs)
            mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

            # Pad or trim the MFCCs to match the expected shape (40, 44)
            desired_shape = (40, 44)
            if mfccs.shape[1] < desired_shape[1]:
                mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
            else:
                mfccs = mfccs[:, :desired_shape[1]]

            # Add batch dimension
            mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

            # Predict emotion for the audio segment
            emotions = model.predict(mfccs)[0]
            predicted_emotion = emotion_labels[np.argmax(emotions)]

            # Update emotion counts
            emotion_counts[predicted_emotion] += 1

    # Release video capture
    cap.release()
    cv2.destroyAllWindows()

    # Find the dominant emotion based on counts
    dominant_emotion = max(emotion_counts, key=emotion_counts.get)
    print(f"Dominant Emotion: {dominant_emotion}")
```

```
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
      y, sr_native = __audioread_load(path, offset, duration, dtype)
    1/1 [==============================] - 0s 46ms/step
    Dominant Emotion: Fear
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/angry.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 2    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

        # Pad or trim the MFCCs to match the expected shape (40, 44)
        desired_shape = (40, 44)
        if mfccs.shape[1] < desired_shape[1]:
            mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
        else:
            mfccs = mfccs[:, :desired_shape[1]]

        # Add batch dimension
        mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

        # Predict emotion for the audio segment
        emotions = model.predict(mfccs)[0]
        predicted_emotion = emotion_labels[np.argmax(emotions)]

        # Update emotion counts
        emotion_counts[predicted_emotion] += 1

# Release video capture
cap.release()
cv2.destroyAllWindows()
```

```
# Find the dominant emotion based on counts
dominant_emotion = max(emotion_counts, key=emotion_counts.get)
print(f"Dominant Emotion: {dominant_emotion}")
```

```
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 25ms/step
<ipython-input-17-c149761646eb>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-17-c149761646eb>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
<ipython-input-17-c149761646eb>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-17-c149761646eb>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 32ms/step
<ipython-input-17-c149761646eb>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-17-c149761646eb>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 29ms/step
<ipython-input-17-c149761646eb>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-17-c149761646eb>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 25ms/step
Dominant Emotion: Fear
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/neutral2.mp4'
cap = cv2.VideoCapture(video_path)
```

```
# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 2    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

        # Pad or trim the MFCCs to match the expected shape (40, 44)
        desired_shape = (40, 44)
        if mfccs.shape[1] < desired_shape[1]:
            mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
        else:
            mfccs = mfccs[:, :desired_shape[1]]

        # Add batch dimension
        mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

        # Predict emotion for the audio segment
        emotions = model.predict(mfccs)[0]
        predicted_emotion = emotion_labels[np.argmax(emotions)]

        # Update emotion counts
        emotion_counts[predicted_emotion] += 1

# Release video capture
cap.release()
cv2.destroyAllWindows()

# Find the dominant emotion based on counts
dominant_emotion = max(emotion_counts, key=emotion_counts.get)
print(f"Dominant Emotion: {dominant_emotion}")
```

```
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - ETA: 0s<ipython-input-19-1a0e8cedef99>:47: UserWarning: PySoundFile failed. Trying audioread i
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 39ms/step
<ipython-input-19-1a0e8cedef99>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 46ms/step
<ipython-input-19-1a0e8cedef99>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 38ms/step
1/1 [==============================] - 0s 23ms/step
<ipython-input-19-1a0e8cedef99>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
<ipython-input-19-1a0e8cedef99>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 25ms/step
Dominant Emotion: Fear
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/disgust.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 3    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))
```

```
        if start_frame <= current_frame_time <= end_frame:
            # Extract audio from the current second
            audio_start_time = current_frame_time
            audio_end_time = current_frame_time + 1000  # 1-second interval

            # Extract audio segment
            audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

            # Preprocess audio data (compute MFCCs)
            mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

            # Pad or trim the MFCCs to match the expected shape (40, 44)
            desired_shape = (40, 44)
            if mfccs.shape[1] < desired_shape[1]:
                mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
            else:
                mfccs = mfccs[:, :desired_shape[1]]

            # Add batch dimension
            mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

            # Predict emotion for the audio segment
            emotions = model.predict(mfccs)[0]
            predicted_emotion = emotion_labels[np.argmax(emotions)]

            # Update emotion counts
            emotion_counts[predicted_emotion] += 1

    # Release video capture
    cap.release()
    cv2.destroyAllWindows()

    # Find the dominant emotion based on counts
    dominant_emotion = max(emotion_counts, key=emotion_counts.get)
    print(f"Dominant Emotion: {dominant_emotion}")
```

```
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 26ms/step
Dominant Emotion: Sad
<ipython-input-20-f1d4d2c75358>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
/usr/local/lib/python3.10/dist-packages/librosa/core/spectrum.py:257: UserWarning: n_fft=2048 is too large for input signal of length
  warnings.warn(
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/emotion1.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 4    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

        # Pad or trim the MFCCs to match the expected shape (40, 44)
        desired_shape = (40, 44)
        if mfccs.shape[1] < desired_shape[1]:
            mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
        else:
            mfccs = mfccs[:, :desired_shape[1]]

        # Add batch dimension
        mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

        # Predict emotion for the audio segment
        emotions = model.predict(mfccs)[0]
```

```
        predicted_emotion = emotion_labels[np.argmax(emotions)]

        # Update emotion counts
        emotion_counts[predicted_emotion] += 1

# Release video capture
cap.release()
cv2.destroyAllWindows()

# Find the dominant emotion based on counts
dominant_emotion = max(emotion_counts, key=emotion_counts.get)
print(f"Dominant Emotion: {dominant_emotion}")
```

```
1/1 [==============================] - 0s 45ms/step
<ipython-input-5-bd49989010ee>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 43ms/step
<ipython-input-5-bd49989010ee>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 46ms/step
<ipython-input-5-bd49989010ee>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 38ms/step
<ipython-input-5-bd49989010ee>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 42ms/step
<ipython-input-5-bd49989010ee>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 40ms/step
<ipython-input-5-bd49989010ee>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 52ms/step
<ipython-input-5-bd49989010ee>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 45ms/step
<ipython-input-5-bd49989010ee>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 45ms/step
Dominant Emotion: Sad
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')
```

```
# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/emotion1.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 4    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

        # Pad or trim the MFCCs to match the expected shape (40, 44)
        desired_shape = (40, 44)
        if mfccs.shape[1] < desired_shape[1]:
            mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
        else:
            mfccs = mfccs[:, :desired_shape[1]]

        # Add batch dimension
        mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

        # Predict emotion for the audio segment
        emotions = model.predict(mfccs)[0]
        predicted_emotion = emotion_labels[np.argmax(emotions)]

        # Update emotion counts
        emotion_counts[predicted_emotion] += 1

# Release video capture
cap.release()
cv2.destroyAllWindows()

# Find the dominant emotion based on counts
dominant_emotion = max(emotion_counts, key=emotion_counts.get)
print(f"Dominant Emotion: {dominant_emotion}")



import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
```

```
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/emotion2.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 1    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

        # Pad or trim the MFCCs to match the expected shape (40, 44)
        desired_shape = (40, 44)
        if mfccs.shape[1] < desired_shape[1]:
            mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
        else:
            mfccs = mfccs[:, :desired_shape[1]]

        # Add batch dimension
        mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

        # Predict emotion for the audio segment
        emotions = model.predict(mfccs)[0]
        predicted_emotion = emotion_labels[np.argmax(emotions)]

        # Update emotion counts
        emotion_counts[predicted_emotion] += 1

# Release video capture
cap.release()
cv2.destroyAllWindows()

# Find the dominant emotion based on counts
dominant_emotion = max(emotion_counts, key=emotion_counts.get)
print(f"Dominant Emotion: {dominant_emotion}")
```

```
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 58ms/step
<ipython-input-7-825d333e8ac1>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 46ms/step
<ipython-input-7-825d333e8ac1>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 45ms/step
<ipython-input-7-825d333e8ac1>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 79ms/step
<ipython-input-7-825d333e8ac1>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 88ms/step
<ipython-input-7-825d333e8ac1>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 128ms/step
<ipython-input-7-825d333e8ac1>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 85ms/step
Dominant Emotion: Fear
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/emotion3.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 2    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)
```

```
    # Read and process video frames
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

        if start_frame <= current_frame_time <= end_frame:
            # Extract audio from the current second
            audio_start_time = current_frame_time
            audio_end_time = current_frame_time + 1000  # 1-second interval

            # Extract audio segment
            audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

            # Preprocess audio data (compute MFCCs)
            mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

            # Pad or trim the MFCCs to match the expected shape (40, 44)
            desired_shape = (40, 44)
            if mfccs.shape[1] < desired_shape[1]:
                mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
            else:
                mfccs = mfccs[:, :desired_shape[1]]

            # Add batch dimension
            mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

            # Predict emotion for the audio segment
            emotions = model.predict(mfccs)[0]
            predicted_emotion = emotion_labels[np.argmax(emotions)]

            # Update emotion counts
            emotion_counts[predicted_emotion] += 1

    # Release video capture
    cap.release()
    cv2.destroyAllWindows()

    # Find the dominant emotion based on counts
    dominant_emotion = max(emotion_counts, key=emotion_counts.get)
    print(f"Dominant Emotion: {dominant_emotion}")
```

```
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 56ms/step
<ipython-input-8-2523f96be5f3>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 47ms/step
<ipython-input-8-2523f96be5f3>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 122ms/step
<ipython-input-8-2523f96be5f3>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 89ms/step
Dominant Emotion: Sad
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/emotion4.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 5    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

        # Pad or trim the MFCCs to match the expected shape (40, 44)
        desired_shape = (40, 44)
        if mfccs.shape[1] < desired_shape[1]:
```

```
            mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
        else:
            mfccs = mfccs[:, :desired_shape[1]]

        # Add batch dimension
        mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

        # Predict emotion for the audio segment
        emotions = model.predict(mfccs)[0]
        predicted_emotion = emotion_labels[np.argmax(emotions)]

        # Update emotion counts
        emotion_counts[predicted_emotion] += 1

# Release video capture
cap.release()
cv2.destroyAllWindows()

# Find the dominant emotion based on counts
dominant_emotion = max(emotion_counts, key=emotion_counts.get)
print(f"Dominant Emotion: {dominant_emotion}")
```

```
<ipython-input-9-9f1721abc0d2>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 837ms/step
<ipython-input-9-9f1721abc0d2>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 39ms/step
<ipython-input-9-9f1721abc0d2>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 53ms/step
<ipython-input-9-9f1721abc0d2>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 71ms/step
<ipython-input-9-9f1721abc0d2>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 64ms/step
<ipython-input-9-9f1721abc0d2>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 73ms/step
<ipython-input-9-9f1721abc0d2>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 37ms/step
<ipython-input-9-9f1721abc0d2>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 38ms/step
<ipython-input-9-9f1721abc0d2>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/emotion5.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
start_time = 0  # Adjust the start time as needed
end_time = 11    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

        # Pad or trim the MFCCs to match the expected shape (40, 44)
        desired_shape = (40, 44)
        if mfccs.shape[1] < desired_shape[1]:
            mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
        else:
            mfccs = mfccs[:, :desired_shape[1]]

        # Add batch dimension
        mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

        # Predict emotion for the audio segment
        emotions = model.predict(mfccs)[0]
        predicted_emotion = emotion_labels[np.argmax(emotions)]

        # Update emotion counts
        emotion_counts[predicted_emotion] += 1

# Release video capture
cap.release()
cv2.destroyAllWindows()

# Find the dominant emotion based on counts
dominant_emotion = max(emotion_counts, key=emotion_counts.get)
print(f"Dominant Emotion: {dominant_emotion}")
```

```
1/1 [==============================] - 0s 43ms/step
<ipython-input-10-8b216e15728f>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 39ms/step
<ipython-input-10-8b216e15728f>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 38ms/step
<ipython-input-10-8b216e15728f>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 44ms/step
<ipython-input-10-8b216e15728f>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 34ms/step
<ipython-input-10-8b216e15728f>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 36ms/step
<ipython-input-10-8b216e15728f>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 46ms/step
<ipython-input-10-8b216e15728f>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 35ms/step
<ipython-input-10-8b216e15728f>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 34ms/step
Dominant Emotion: Fear
```

```
import cv2
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from collections import defaultdict

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
video_path = '/content/wednesday!!.mp4'
cap = cv2.VideoCapture(video_path)

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Define the start and end times for the short video clip (in seconds)
```

```
start_time = 0  # Adjust the start time as needed
end_time = 5    # Adjust the end time as needed

# Calculate the corresponding frame numbers
start_frame = int(start_time * 1000)  # Convert to milliseconds
end_frame = int(end_time * 1000)

# Initialize a dictionary to count emotions
emotion_counts = defaultdict(int)

# Read and process video frames
while cap.isOpened():
    ret, frame = cap.read()
    if not ret:
        break

    current_frame_time = int(cap.get(cv2.CAP_PROP_POS_MSEC))

    if start_frame <= current_frame_time <= end_frame:
        # Extract audio from the current second
        audio_start_time = current_frame_time
        audio_end_time = current_frame_time + 1000  # 1-second interval

        # Extract audio segment
        audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)

        # Preprocess audio data (compute MFCCs)
        mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

        # Pad or trim the MFCCs to match the expected shape (40, 44)
        desired_shape = (40, 44)
        if mfccs.shape[1] < desired_shape[1]:
            mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
        else:
            mfccs = mfccs[:, :desired_shape[1]]

        # Add batch dimension
        mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

        # Predict emotion for the audio segment
        emotions = model.predict(mfccs)[0]
        predicted_emotion = emotion_labels[np.argmax(emotions)]

        # Update emotion counts
        emotion_counts[predicted_emotion] += 1

# Release video capture
cap.release()
cv2.destroyAllWindows()

# Find the dominant emotion based on counts
dominant_emotion = max(emotion_counts, key=emotion_counts.get)
print(f"Dominant Emotion: {dominant_emotion}")
```

```
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 40ms/step
<ipython-input-2-10a1e9dbeba7>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 36ms/step
<ipython-input-2-10a1e9dbeba7>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 38ms/step
<ipython-input-2-10a1e9dbeba7>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 43ms/step
<ipython-input-2-10a1e9dbeba7>:47: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(video_path, sr=sample_rate, offset=audio_start_time/1000, duration=1.0)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 67ms/step
Dominant Emotion: Fear
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/sad.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-11-3ddcb757123e>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
```

```
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 830ms/step
Emotion Prediction: Sad
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/Omg.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-12-4a0b80d7f1ca>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 1s/step
Emotion Prediction: Sad
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '//content/Mad.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)
```

```
# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-13-47f7fda3fc2a>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 1s/step
Emotion Prediction: Surprise
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/angry.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-14-4911830a3f7b>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
```

```
        It will be removed in librosa version 1.0.
      y, sr_native = __audioread_load(path, offset, duration, dtype)
WARNING:tensorflow:5 out of the last 335 calls to <function Model.make_predict_function.<locals>.predict_function at 0x79ea2cb35090> tri
1/1 [==============================] - 1s 795ms/step
Emotion Prediction: Fear
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/disgust.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-15-08be28aa978c>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
WARNING:tensorflow:6 out of the last 336 calls to <function Model.make_predict_function.<locals>.predict_function at 0x79ea450a72e0> tri
1/1 [==============================] - 1s 1s/step
Emotion Prediction: Sad
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/happy.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients
```

```
# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-16-047f23c39244>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 740ms/step
Emotion Prediction: Sad
```

WEDNESDAY 10 EMOTIONS

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/Mad.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
```

```
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-1-b01fcab67621>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 1s/step
Emotion Prediction: Surprise
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/Omg.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-2-4a0b80d7f1ca>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 723ms/step
Emotion Prediction: Sad
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/angry.mp4'
```

```
# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-3-4911830a3f7b>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 1s/step
Emotion Prediction: Fear
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/disgust.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
```

```
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-4-08be28aa978c>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 759ms/step
Emotion Prediction: Sad
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/full happy.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-5-df6424949340>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7b322cea3400> trigg
1/1 [==============================] - 1s 764ms/step
Emotion Prediction: Sad
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
```

```
audio_path = '/content/happy.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-6-047f23c39244>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<locals>.predict_function at 0x7b322ca90d30> trigg
1/1 [==============================] - 1s 804ms/step
Emotion Prediction: Sad
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/neutral.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
```

```
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-7-46adfb91c926>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 801ms/step
Emotion Prediction: Surprise
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/neutral2.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-8-af32a7c45d33>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 1s/step
Emotion Prediction: Surprise
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']
```

```
# Load the video file
audio_path = '/content/sad.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-9-3ddcb757123e>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 770ms/step
Emotion Prediction: Sad
```

END

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = ''

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)
```

```
# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-11-a65849d2f06d>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 0s 493ms/step
Emotion Prediction: Fear
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = ''

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/_ANGRY.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients
```

```
# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-18-f69ac79074d3>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 837ms/step
Emotion Prediction: Fear
```

```
import librosa
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model

# Load your trained emotion recognition model
model = load_model('/content/audioemotion.h5')

# Define emotion labels
emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Neutral', 'Sad', 'Surprise']

# Load the video file
audio_path = '/content/fire!!.mp4'

# Initialize variables for audio processing
sample_rate = 22050  # Adjust according to your model's audio preprocessing
n_mfcc = 40  # Number of MFCC coefficients

# Extract audio from the entire video
audio, _ = librosa.load(audio_path, sr=sample_rate)

# Preprocess audio data (compute MFCCs)
mfccs = librosa.feature.mfcc(y=audio, sr=sample_rate, n_mfcc=n_mfcc)

# Pad or trim the MFCCs to match the expected shape (40, 44)
desired_shape = (40, 44)
if mfccs.shape[1] < desired_shape[1]:
    mfccs = np.pad(mfccs, ((0, 0), (0, desired_shape[1] - mfccs.shape[1])))
else:
    mfccs = mfccs[:, :desired_shape[1]]

# Add batch dimension
mfccs = mfccs[np.newaxis, :, :]  # Shape will be (1, 40, 44)

# Predict emotion for the entire audio clip
emotions = model.predict(mfccs)[0]
predicted_emotion = emotion_labels[np.argmax(emotions)]

# Print the single emotion prediction for the entire audio clip
print(f"Emotion Prediction: {predicted_emotion}")
```

```
<ipython-input-17-c3d6e63bb9ba>:20: UserWarning: PySoundFile failed. Trying audioread instead.
  audio, _ = librosa.load(audio_path, sr=sample_rate)
/usr/local/lib/python3.10/dist-packages/librosa/core/audio.py:183: FutureWarning: librosa.core.audio.__audioread_load
        Deprecated as of librosa version 0.10.0.
        It will be removed in librosa version 1.0.
  y, sr_native = __audioread_load(path, offset, duration, dtype)
1/1 [==============================] - 1s 972ms/step
Emotion Prediction: Fear
```