Corrected Code –

```
import cv2
import numpy as np
from tensorflow.keras.models import model_from_json
from tensorflow.python.keras.backend import set_session
import tensorflow as tf
from google.colab.patches import cv2_imshow

# Set up TensorFlow session configuration
config = tf.compat.v1.ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.15
session = tf.compat.v1.Session(config=config)
set_session(session)

# Define the facial expression model class
class FacialExpressionModel(object):
    EMOTIONS_LIST = ["Angry", "Disgust", "Fear", "Happy", "Neutral",
"Sad", "Surprise"]

    def __init__(self, model_json_file, model_weights_file):
        # Load model from JSON file
        try:
            with open(model_json_file, "r") as json_file:
                loaded_model_json = json_file.read()
                self.loaded_model = model_from_json(loaded_model_json)
        except Exception as e:
            print(f"Error loading model JSON file: {e}")
            raise

        # Load weights into the new model
        try:
            self.loaded_model.load_weights(model_weights_file)
        except Exception as e:
            print(f"Error loading model weights file: {e}")
            raise

    def predict_emotion(self, img):
        global session
        set_session(session)
        self.preds = self.loaded_model.predict(img)
        return
FacialExpressionModel.EMOTIONS_LIST[np.argmax(self.preds)]

# Initialize the face detector and the facial expression model
facec = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

try:
```

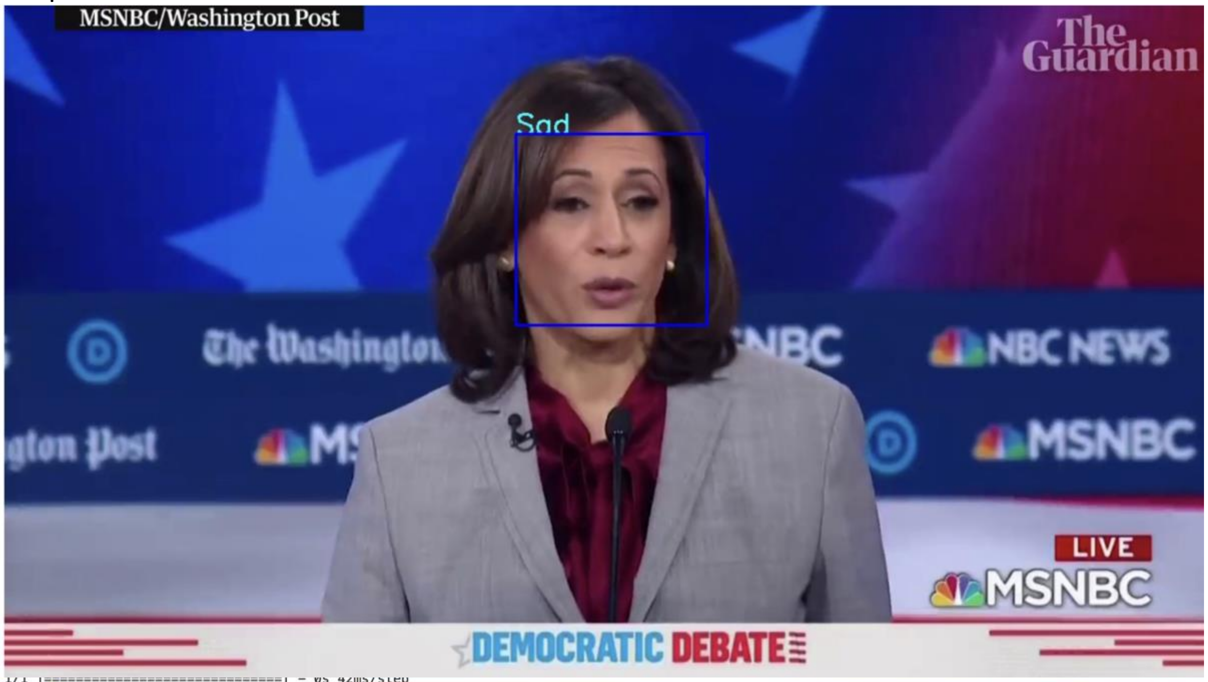
```python
    model = FacialExpressionModel("model.json",
"/content/model_weights.h5")
except Exception as e:
    print(f"Failed to initialize FacialExpressionModel: {e}")
    exit(1)

font = cv2.FONT_HERSHEY_SIMPLEX

# Define the VideoCamera class
class VideoCamera(object):
    def __init__(self):
        # Update the path to your video file
        self.video =
cv2.VideoCapture('/content/presidential_debate.mp4')
        if not self.video.isOpened():
            print("Error: Could not open video.")
            exit(1)

    def __del__(self):
        self.video.release()

    # Returns camera frames along with bounding boxes and predictions
    def get_frame(self):
        ret, fr = self.video.read()
        if not ret:
            print("Error: Could not read frame.")
            return None

        gray_fr = cv2.cvtColor(fr, cv2.COLOR_BGR2GRAY)
        faces = facec.detectMultiScale(gray_fr, 1.3, 5)

        for (x, y, w, h) in faces:
            fc = gray_fr[y:y+h, x:x+w]
            roi = cv2.resize(fc, (48, 48))
            pred = model.predict_emotion(roi[np.newaxis, :, :,
np.newaxis])

            cv2.putText(fr, pred, (x, y), font, 1, (255, 255, 0), 2)
            cv2.rectangle(fr, (x, y), (x+w, y+h), (255, 0, 0), 2)

        _, jpeg = cv2.imencode('.jpg', fr)
        return jpeg.tobytes()

# Example usage
if __name__ == "__main__":
    video_camera = VideoCamera()
    frame_count = 0
    max_frames = 2  # Number of frames to process
```

```
while frame_count < max_frames:
    frame = video_camera.get_frame()
    if frame is None:
        break
    # Process the frame or display it in your application
    # For example, using OpenCV to display the frame
    nparr = np.frombuffer(frame, np.uint8)
    img_np = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
    cv2_imshow(img_np)  # Use cv2_imshow for Google Colab

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    frame_count += 1

cv2.destroyAllWindows()
```

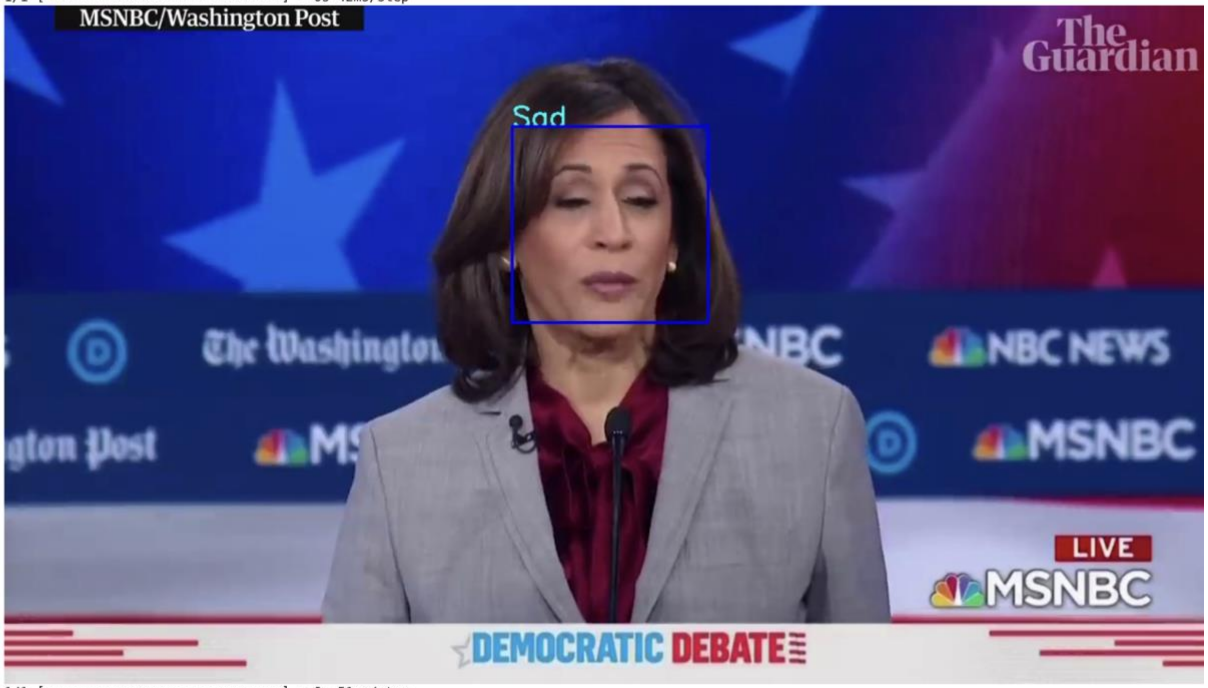
Changes made –

- **TensorFlow Session Configuration:**
  - Added TensorFlow session configuration to manage GPU memory usage.
- **Facial Expression Model Class:**
  - Combined the class definition into a single script.
  - Handled exceptions for loading the model JSON and weights files.
  - Ensured the session is set before predicting emotions.
- **VideoCamera Class:**
  - Updated the video file path to `/content/presidential_debate.mp4`.
  - Added a check to ensure the video file can be opened successfully.
- **Processing Frames:**
  - Added a frame count limit to process only the first 1 or 2 frames of the video.
  - Used `cv2_imshow` for displaying frames in Google Colab.
-

Output –



```
1/1 [==============================] - 0s 42ms/step
```



```
1/1 [==============================] - 0s 51ms/step
```