```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from deepface import DeepFace
from IPython.display import display, Javascript, Image
from google.colab.output import eval_js
import base64
from tensorflow.keras.preprocessing import image
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input, decode_predictions


# 1. Load the pre-trained MobileNetV2 model with ImageNet weights
model = MobileNetV2(weights="imagenet")


# Function to capture a photo using webcam
def take_photo(filename='photo.jpg', quality=0.8):
    js = Javascript('''
        async function takePhoto(quality) {
          const div = document.createElement('div');
          const capture = document.createElement('button');
          capture.textContent = '📸 Capture';
          div.appendChild(capture);

          const video = document.createElement('video');
          video.style.display = 'block';
          const stream = await navigator.mediaDevices.getUserMedia({video: true});
          document.body.appendChild(div);
          div.appendChild(video);
          video.srcObject = stream;
          await video.play();
          google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);
          await new Promise((resolve) => capture.onclick = resolve);
          const canvas = document.createElement('canvas');
          canvas.width = video.videoWidth;
          canvas.height = video.videoHeight;
          canvas.getContext('2d').drawImage(video, 0, 0);
          stream.getTracks().forEach(track => track.stop());
          div.remove();
          return canvas.toDataURL('image/jpeg', quality);
        }
    ''')
    display(js)
    data = eval_js('takePhoto({})'.format(quality))
    binary = base64.b64decode(data.split(',')[1])
    with open(filename, 'wb') as f:
        f.write(binary)
    return filename


# Function to upload an image
from google.colab import files
def upload_image():
    uploaded = files.upload()
    for filename in uploaded.keys():
        return filename


# Function to detect emotions using DeepFace
def detect_emotions(img_path):
    result = DeepFace.analyze(img_path=img_path, actions=['emotion'], enforce_detection=False)
    img = cv2.imread(img_path)

    for face in result:
        x, y, w, h = face['region']['x'], face['region']['y'], face['region']['w'], face['region']['h']
        emotion = face['dominant_emotion']
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
        cv2.putText(img, emotion, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)

    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.imshow(img_rgb)
    plt.axis('off')
    plt.title("Detected Emotion(s)")
    plt.show()
```

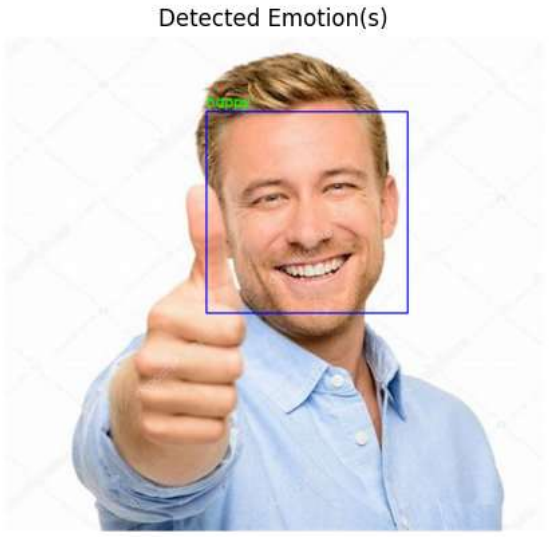
```
    print("Top Emotion:", result[0]['dominant_emotion'])
    print("All Emotions:")
    for emotion, score in result[0]['emotion'].items():
        print(f"{emotion}: {score:.2f}")


# Function to get MobileNetV2 predictions from an image
def get_mobilenet_predictions(img_path):
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)  # Convert the image to a NumPy array
    img_array = np.expand_dims(img_array, axis=0)
    img_array = preprocess_input(img_array)  # Preprocess the image (normalization)

    predictions = model.predict(img_array)

    decoded_predictions = decode_predictions(predictions, top=5)[0]
    print("Predictions from MobileNetV2:")
    for i, (imagenet_id, label, score) in enumerate(decoded_predictions):
        print(f"{i+1}: {label} ({score:.2f})")


# Main function to choose between webcam capture or image upload
def main():
    choice = input("Do you want to use your webcam (w) or upload an image (u)? (w/u): ").lower()

    if choice == 'w':
        # Capture image from webcam
        filename = take_photo()
        print(f"Photo saved as {filename}")
        display(Image(filename=filename))

        # Get predictions from MobileNetV2
        get_mobilenet_predictions(filename)

        # Detect emotions
        detect_emotions(filename)

    elif choice == 'u':
        # Upload an image
        filename = upload_image()
        print(f"Uploaded image: {filename}")

        # Get predictions from MobileNetV2
        get_mobilenet_predictions(filename)

        # Detect emotions
        detect_emotions(filename)

    else:
        print("Invalid choice. Please choose either 'w' for webcam or 'u' for upload.")

# Run the main function
main()
```

```
25-04-22 14:04:02 - Directory /root/.deepface has been created
25-04-22 14:04:02 - Directory /root/.deepface/weights has been created
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_
14536120/14536120 [==============================] - 0s 0us/step
Do you want to use your webcam (w) or upload an image (u)? (w/u): u
```

Choose Files OIP (1).jpeg

- **OIP (1).jpeg**(image/jpeg) - 58730 bytes, last modified: 22/4/2025 - 100% done

```
Saving OIP (1).jpeg to OIP (1).jpeg
Uploaded image: OIP (1).jpeg
1/1 [==============================] - 1s 1s/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
35363/35363 [==============================] - 0s 0us/step
Predictions from MobileNetV2:
1: stethoscope (0.20)
2: lab_coat (0.12)
3: sweatshirt (0.06)
4: Band_Aid (0.04)
5: sunscreen (0.03)
Downloading...
From: https://github.com/serengil/deepface_models/releases/download/v1.0/facial_expression_model_weights.h5
To: /root/.deepface/weights/facial_expression_model_weights.h5
25-04-22 14:05:15 - facial_expression_model_weights.h5 will be downloaded...
100%|██████████| 5.98M/5.98M [00:00<00:00, 240MB/s]
```




```
Top Emotion: happy
All Emotions:
angry: 0.00
disgust: 0.00
fear: 0.00
happy: 99.97
sad: 0.00
surprise: 0.00
```