Emotion Recognition

This tutorial is specific to the CM5 version and will not work directly with the CM4 version

1. Experimental Objective

Learn to manually control the robotic arm of a robot dog, thereby enabling control of multiple robot arms.

2. Experimental Procedure

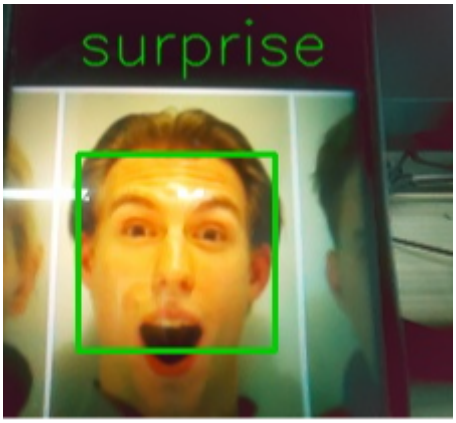
Log in to the robot dog's system, exit the robot dog program, and enter "ip (the IP address of the robot dog):8888" in your browser. Once logged in, enter the password "yahboom."

Then log in.
Go to the directory **cd ~/DOGZILLA_Lite_class/5.AI Visual Recognition Course/19. Emotion_Recognition** and run **emotion_demo.ipynb**.

Finally, run the program.

3. Experimental Results

1. After the program runs, this function will detect a person's facial expression. If an emotion is detected, a box will be drawn on the live screen and the result will be marked.



2. Exit the program by pressing the button in the lower left corner of the screen.

4. Experimental Source Code Analysis

```
while True:
    bgr_image = picam2.capture_array()
    gray_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2GRAY)
    rgb_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2RGB)
    faces = detect_faces(face_detection, gray_image)

    for face_coordinates in faces:
        x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
        gray_face = gray_image[y1:y2, x1:x2]
        try:
            gray_face = cv2.resize(gray_face, (emotion_target_size))
        except:
            continue

        gray_face = preprocess_input(gray_face, True)
        gray_face = np.expand_dims(gray_face, 0)
        gray_face = np.expand_dims(gray_face, -1)
        emotion_prediction = emotion_classifier.predict(gray_face)
        emotion_probability = np.max(emotion_prediction)
        emotion_label_arg = np.argmax(emotion_prediction)
        emotion_text = emotion_labels[emotion_label_arg]
        emotion_window.append(emotion_text)
        if emotion_text != last_emotion:
            Count_num = 0
        else:
            Count_num += 1
        last_emotion = emotion_text

        if len(emotion_window) > frame_window:
            emotion_window.pop(0)
        try:
            emotion_mode = mode(emotion_window)
        except:
            continue

        if emotion_text == 'angry' and Count_num >= Count :
            color = emotion_probability * np.asarray((255, 0, 0))
            Count_num = 0
        elif emotion_text == 'sad' and Count_num >= Count :
            color = emotion_probability * np.asarray((0, 0, 255))
            Count_num = 0
```

```
        elif emotion_text == 'happy' and Count_num >= Count :
            color = emotion_probability * np.asarray((255, 255, 0))
            Count_num = 0
        elif emotion_text == 'surprise' and Count_num >= Count :
            color = emotion_probability * np.asarray((0, 255, 255))
            Count_num = 0
        elif emotion_text == 'neutral' and Count_num >= Count :
            color = emotion_probability * np.asarray((0, 255, 255))
            Count_num = 0
        else:
            color = emotion_probability * np.asarray((0, 255, 0))

        color = color.astype(int)
        color = color.tolist()

        draw_bounding_box(face_coordinates, rgb_image, color)
        draw_text(face_coordinates, rgb_image, emotion_mode,
                  color, 0, -45, 1, 1)

    bgr_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2BGR)

    image_widget.value = bgr8_to_jpeg(bgr_image)


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

detect_faces(): This function's main business logic is to detect faces.
apply_offsets(): This function's business logic is to detect facial expressions.
draw_bounding_box(): Draws the recognition result box.
draw_text(): Draws the recognized emotional text.
bgr8_to_jpeg(): Displays the image.