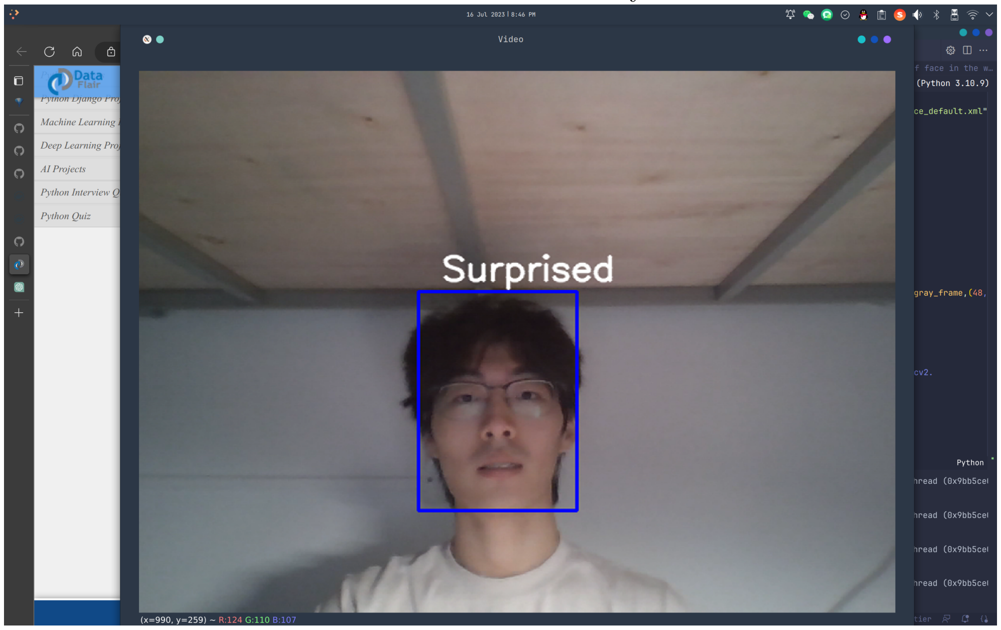
2_emojify

吴清柳

July 24, 2023

1 Facial Emotion Recognition using CNN

面 部 表 情 识 别, CNN 网 络, Pytorch 搭 建, GPU 训 练



[2]:
```
import numpy as np
import cv2

from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D
from keras.optimizers import Adam
from keras.layers import MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
```

```
2023-07-17 09:49:02.159971: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
```

[3]:
```python
train_dir = "../dataset/train"
val_dir = "../dataset/test"
train_datagen = ImageDataGenerator(rescale=1.0 / 255)
val_datagen = ImageDataGenerator(rescale=1.0 / 255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(48, 48),
    batch_size=64,
    color_mode="grayscale",
    class_mode="categorical",
)
validation_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=(48, 48),
    batch_size=64,
    color_mode="grayscale",
    class_mode='categorical'
)
```

```
Found 28709 images belonging to 7 classes.
Found 7178 images belonging to 7 classes.
```

[4]:
```python
# Build the CNN model
# Layers in Sequential are stacked on top of each other
emotion_model = Sequential()

# 2D Conv layer with 32 filters of size 3x3. Input shape is (48,48,1), for
# grayscale images of size 48x48.
emotion_model.add(
    Conv2D(32, kernel_size=(3, 3), activation="relu", input_shape=(48, 48, 1))
)
emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation="relu"))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation="relu"))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation="relu"))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Flatten())
emotion_model.add(Dense(1024, activation="relu"))
emotion_model.add(Dropout(0.25))
emotion_model.add(Dense(7, activation="softmax"))
```

```
2023-07-17 09:49:07.924673: I
```

```
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:981]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2023-07-17 09:49:07.949355: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:981]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2023-07-17 09:49:07.949664: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:981]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2023-07-17 09:49:07.950856: I tensorflow/core/platform/cpu_feature_guard.cc:193]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2023-07-17 09:49:07.951713: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:981]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2023-07-17 09:49:07.951954: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:981]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2023-07-17 09:49:07.952102: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:981]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2023-07-17 09:49:07.997469: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:981]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2023-07-17 09:49:07.997661: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:981]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2023-07-17 09:49:07.997796: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_gpu_executor.cc:981]
successful NUMA node read from SysFS had negative value (-1), but there must be
at least one NUMA node, so returning NUMA node zero
2023-07-17 09:49:07.997888: I
tensorflow/core/common_runtime/gpu/gpu_device.cc:1613] Created device
/job:localhost/replica:0/task:0/device:GPU:0 with 4279 MB memory:  -> device: 0,
name: NVIDIA GeForce RTX 3060 Laptop GPU, pci bus id: 0000:01:00.0, compute
capability: 8.6
```

[5]:
```python
# Train the model
emotion_model.compile(
    loss="categorical_crossentropy",
    optimizer=Adam(learning_rate=0.0001, decay=1e-6),
    metrics=["accuracy"],
)
emotion_model_info = emotion_model.fit_generator(
    train_generator,
    steps_per_epoch=28709 // 64,
    epochs=50,
    validation_data=validation_generator,
    validation_steps=7178 // 64,
)
```

```
Epoch 1/50
```

```
/tmp/ipykernel_3283901/729160076.py:7: UserWarning: `Model.fit_generator` is
deprecated and will be removed in a future version. Please use `Model.fit`,
which supports generators.
  emotion_model_info = emotion_model.fit_generator(
2023-07-17 09:49:11.474218: E
tensorflow/core/grappler/optimizers/meta_optimizer.cc:954] layout failed:
INVALID_ARGUMENT: Size of values 0 does not match size of permutation 4 @ fanin
shape insequential/dropout/dropout/SelectV2-2-TransposeNHWCToNCHW-
LayoutOptimizer
2023-07-17 09:49:11.595667: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:428] Loaded cuDNN
version 8800
2023-07-17 09:49:12.058971: I
tensorflow/compiler/xla/stream_executor/cuda/cuda_blas.cc:630] TensorFloat-32
will be used for the matrix multiplication. This will only be logged once.
```

```
448/448 [==============================] - 9s 16ms/step - loss: 1.7829 -
accuracy: 0.2728 - val_loss: 1.6283 - val_accuracy: 0.3758
Epoch 2/50
448/448 [==============================] - 6s 14ms/step - loss: 1.5656 -
accuracy: 0.3932 - val_loss: 1.5067 - val_accuracy: 0.4210
Epoch 3/50
448/448 [==============================] - 6s 14ms/step - loss: 1.4544 -
accuracy: 0.4454 - val_loss: 1.4125 - val_accuracy: 0.4636
Epoch 4/50
448/448 [==============================] - 6s 14ms/step - loss: 1.3752 -
accuracy: 0.4788 - val_loss: 1.3264 - val_accuracy: 0.5008
Epoch 5/50
448/448 [==============================] - 6s 14ms/step - loss: 1.3161 -
accuracy: 0.5021 - val_loss: 1.2807 - val_accuracy: 0.5138
Epoch 6/50
448/448 [==============================] - 6s 14ms/step - loss: 1.2694 -
accuracy: 0.5229 - val_loss: 1.2504 - val_accuracy: 0.5229
```

```
Epoch 7/50
448/448 [==============================] - 6s 14ms/step - loss: 1.2245 -
accuracy: 0.5398 - val_loss: 1.2287 - val_accuracy: 0.5306
Epoch 8/50
448/448 [==============================] - 6s 14ms/step - loss: 1.1906 -
accuracy: 0.5546 - val_loss: 1.1861 - val_accuracy: 0.5467
Epoch 9/50
448/448 [==============================] - 6s 14ms/step - loss: 1.1604 -
accuracy: 0.5652 - val_loss: 1.1785 - val_accuracy: 0.5587
Epoch 10/50
448/448 [==============================] - 6s 14ms/step - loss: 1.1277 -
accuracy: 0.5763 - val_loss: 1.1546 - val_accuracy: 0.5621
Epoch 11/50
448/448 [==============================] - 6s 14ms/step - loss: 1.0971 -
accuracy: 0.5899 - val_loss: 1.1342 - val_accuracy: 0.5692
Epoch 12/50
448/448 [==============================] - 7s 15ms/step - loss: 1.0707 -
accuracy: 0.6007 - val_loss: 1.1243 - val_accuracy: 0.5752
Epoch 13/50
448/448 [==============================] - 7s 15ms/step - loss: 1.0448 -
accuracy: 0.6093 - val_loss: 1.1206 - val_accuracy: 0.5752
Epoch 14/50
448/448 [==============================] - 6s 14ms/step - loss: 1.0148 -
accuracy: 0.6238 - val_loss: 1.1092 - val_accuracy: 0.5845
Epoch 15/50
448/448 [==============================] - 7s 15ms/step - loss: 0.9858 -
accuracy: 0.6344 - val_loss: 1.1015 - val_accuracy: 0.5830
Epoch 16/50
448/448 [==============================] - 7s 14ms/step - loss: 0.9583 -
accuracy: 0.6470 - val_loss: 1.0874 - val_accuracy: 0.5932
Epoch 17/50
448/448 [==============================] - 7s 14ms/step - loss: 0.9319 -
accuracy: 0.6563 - val_loss: 1.0878 - val_accuracy: 0.5961
Epoch 18/50
448/448 [==============================] - 8s 17ms/step - loss: 0.9062 -
accuracy: 0.6670 - val_loss: 1.0768 - val_accuracy: 0.6014
Epoch 19/50
448/448 [==============================] - 7s 15ms/step - loss: 0.8755 -
accuracy: 0.6793 - val_loss: 1.0733 - val_accuracy: 0.6060
Epoch 20/50
448/448 [==============================] - 7s 15ms/step - loss: 0.8385 -
accuracy: 0.6919 - val_loss: 1.0764 - val_accuracy: 0.6063
Epoch 21/50
448/448 [==============================] - 7s 15ms/step - loss: 0.8162 -
accuracy: 0.6991 - val_loss: 1.0786 - val_accuracy: 0.6098
Epoch 22/50
448/448 [==============================] - 7s 16ms/step - loss: 0.7840 -
accuracy: 0.7159 - val_loss: 1.0843 - val_accuracy: 0.6060
```

```
Epoch 23/50
448/448 [==============================] - 7s 15ms/step - loss: 0.7522 -
accuracy: 0.7267 - val_loss: 1.0808 - val_accuracy: 0.6087
Epoch 24/50
448/448 [==============================] - 6s 14ms/step - loss: 0.7213 -
accuracy: 0.7398 - val_loss: 1.0977 - val_accuracy: 0.6105
Epoch 25/50
448/448 [==============================] - 7s 15ms/step - loss: 0.6940 -
accuracy: 0.7495 - val_loss: 1.0835 - val_accuracy: 0.6140
Epoch 26/50
448/448 [==============================] - 7s 15ms/step - loss: 0.6590 -
accuracy: 0.7638 - val_loss: 1.0951 - val_accuracy: 0.6151
Epoch 27/50
448/448 [==============================] - 6s 14ms/step - loss: 0.6285 -
accuracy: 0.7723 - val_loss: 1.1041 - val_accuracy: 0.6129
Epoch 28/50
448/448 [==============================] - 6s 14ms/step - loss: 0.6040 -
accuracy: 0.7843 - val_loss: 1.1255 - val_accuracy: 0.6187
Epoch 29/50
448/448 [==============================] - 6s 14ms/step - loss: 0.5709 -
accuracy: 0.7946 - val_loss: 1.1197 - val_accuracy: 0.6225
Epoch 30/50
448/448 [==============================] - 7s 14ms/step - loss: 0.5458 -
accuracy: 0.8023 - val_loss: 1.1368 - val_accuracy: 0.6205
Epoch 31/50
448/448 [==============================] - 6s 14ms/step - loss: 0.5127 -
accuracy: 0.8163 - val_loss: 1.1554 - val_accuracy: 0.6200
Epoch 32/50
448/448 [==============================] - 6s 14ms/step - loss: 0.4888 -
accuracy: 0.8250 - val_loss: 1.1744 - val_accuracy: 0.6164
Epoch 33/50
448/448 [==============================] - 7s 15ms/step - loss: 0.4644 -
accuracy: 0.8356 - val_loss: 1.1698 - val_accuracy: 0.6190
Epoch 34/50
448/448 [==============================] - 7s 15ms/step - loss: 0.4423 -
accuracy: 0.8418 - val_loss: 1.1938 - val_accuracy: 0.6207
Epoch 35/50
448/448 [==============================] - 7s 15ms/step - loss: 0.4182 -
accuracy: 0.8520 - val_loss: 1.2137 - val_accuracy: 0.6210
Epoch 36/50
448/448 [==============================] - 7s 15ms/step - loss: 0.3961 -
accuracy: 0.8569 - val_loss: 1.2180 - val_accuracy: 0.6214
Epoch 37/50
448/448 [==============================] - 6s 14ms/step - loss: 0.3731 -
accuracy: 0.8687 - val_loss: 1.2565 - val_accuracy: 0.6198
Epoch 38/50
448/448 [==============================] - 7s 15ms/step - loss: 0.3563 -
accuracy: 0.8725 - val_loss: 1.2841 - val_accuracy: 0.6250
```

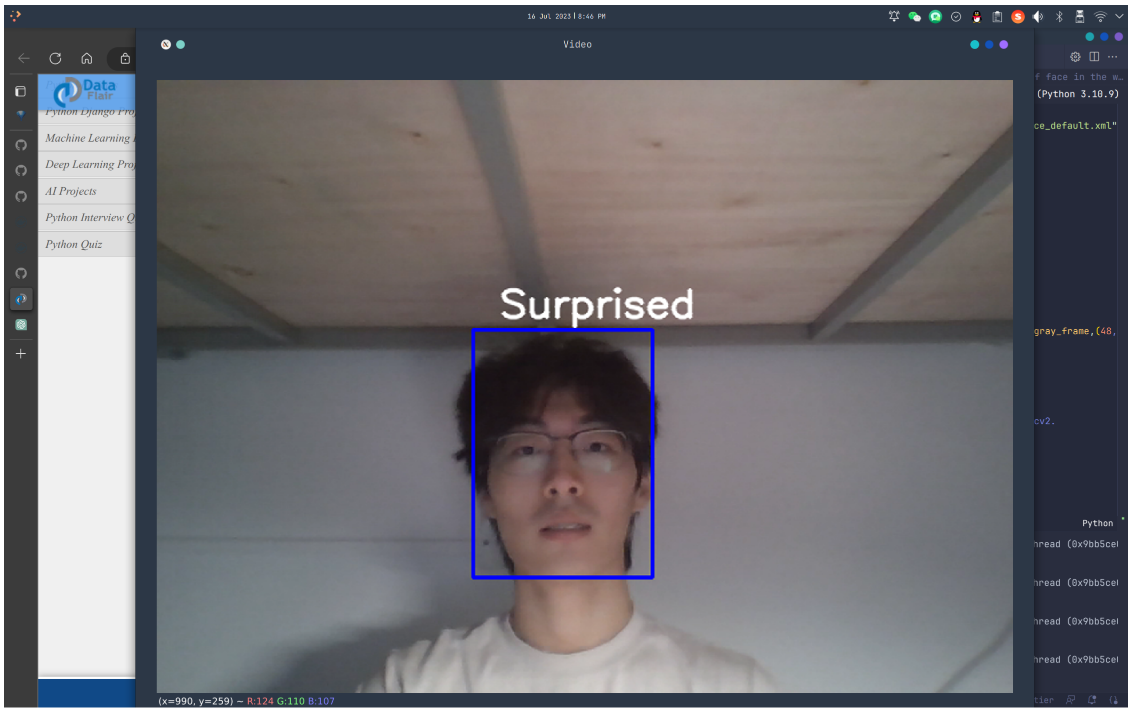
```
Epoch 39/50
448/448 [==============================] - 7s 15ms/step - loss: 0.3360 -
accuracy: 0.8800 - val_loss: 1.2896 - val_accuracy: 0.6208
Epoch 40/50
448/448 [==============================] - 7s 15ms/step - loss: 0.3217 -
accuracy: 0.8863 - val_loss: 1.3039 - val_accuracy: 0.6223
Epoch 41/50
448/448 [==============================] - 7s 15ms/step - loss: 0.3116 -
accuracy: 0.8896 - val_loss: 1.3082 - val_accuracy: 0.6164
Epoch 42/50
448/448 [==============================] - 7s 15ms/step - loss: 0.2943 -
accuracy: 0.8970 - val_loss: 1.3191 - val_accuracy: 0.6256
Epoch 43/50
448/448 [==============================] - 7s 15ms/step - loss: 0.2774 -
accuracy: 0.9020 - val_loss: 1.3495 - val_accuracy: 0.6219
Epoch 44/50
448/448 [==============================] - 7s 15ms/step - loss: 0.2696 -
accuracy: 0.9067 - val_loss: 1.3711 - val_accuracy: 0.6239
Epoch 45/50
448/448 [==============================] - 7s 15ms/step - loss: 0.2595 -
accuracy: 0.9096 - val_loss: 1.4030 - val_accuracy: 0.6184
Epoch 46/50
448/448 [==============================] - 7s 15ms/step - loss: 0.2426 -
accuracy: 0.9166 - val_loss: 1.3908 - val_accuracy: 0.6204
Epoch 47/50
448/448 [==============================] - 7s 16ms/step - loss: 0.2330 -
accuracy: 0.9193 - val_loss: 1.4336 - val_accuracy: 0.6210
Epoch 48/50
448/448 [==============================] - 7s 16ms/step - loss: 0.2309 -
accuracy: 0.9182 - val_loss: 1.4185 - val_accuracy: 0.6189
Epoch 49/50
448/448 [==============================] - 7s 15ms/step - loss: 0.2127 -
accuracy: 0.9279 - val_loss: 1.4498 - val_accuracy: 0.6165
Epoch 50/50
448/448 [==============================] - 7s 15ms/step - loss: 0.2093 -
accuracy: 0.9273 - val_loss: 1.4822 - val_accuracy: 0.6180
```

[6]:
```python
# save the model weigh
import datetime
emotion_model.save_weights(f'model-{datetime.date.today()}.h5')
```

[22]:
```python
# using OpenCV haarcascade xml detect the bounding boxes of face in the webcam␣
↪and predict the emotions
cv2.ocl.setUseOpenCL(False)

emotion_dict = {
    0: "Angry",
    1: "Disgusted",
    2: "Fearful",
    3: "Happy",
    4: "Neutral",
    5: "Sad",
    6: "Surprised",
}

cap = cv2.VideoCapture(0)

# load the cascade classifier before loop
haarcascade_path = cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
bounding_box = cv2.CascadeClassifier(haarcascade_path)

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

    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    num_faces = bounding_box.detectMultiScale(
        gray_frame, scaleFactor=1.3, minNeighbors=5
    )

    for x, y, w, h in num_faces:
        cv2.rectangle(frame, (x, y - 50), (x + w, y + h + 10), (255, 0, 0), 2)
        roi_gray_frame = gray_frame[y : y + h, x : x + w]
        cropped_img = np.expand_dims(
            np.expand_dims(cv2.resize(roi_gray_frame, (48, 48)), -1), 0
        )
        emotion_prediction = emotion_model.predict(cropped_img)
        maxindex = int(np.argmax(emotion_prediction))
        cv2.putText(
            frame,
            emotion_dict[maxindex],
            (x + 20, y - 60),
            cv2.FONT_HERSHEY_SIMPLEX,
            1,
            (255, 255, 255),
            2,
            cv2.LINE_AA,
        )

    cv2.imshow(
        "Video", cv2.resize(frame, (1200, 860), interpolation=cv2.INTER_CUBIC)
    )
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

cap.release()
cv2.destroyAllWindows()
```

```
[ WARN:0@2201.998] global cap_v4l.cpp:982 open VIDEOIO(V4L2:/dev/video0): can't
open camera by index
[ERROR:0@2201.998] global obsensor_uvc_stream_channel.cpp:156
getStreamChannelGroup Camera index out of range
[ERROR:0@2201.998] global persistence.cpp:512 open Can't open file:
'haarcascade_frontalface_default.xmlhaarcascade_frontalface_default.xml' in read
mode
```

1.0.1 Emotion detection result



1.1 Code for GUI and mapping with emojis

save the emojis corresponding to each of the seven enotions in the dataset.

[23]:
```python
import tkinter as tk
from tkinter import *
import cv2
from PIL import Image, ImageTk
import os
import numpy as np
import cv2
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D
from keras.optimizers import Adam
from keras.layers import MaxPool2D
from keras.preprocessing.image import ImageDataGenerator

emotion_model = Sequential()

emotion_model.add(
    Conv2D(32, kernel_size=(3, 3), activation="relu", input_shape=(48, 48, 1))
)
emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation="relu"))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation="relu"))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation="relu"))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
emotion_model.add(Flatten())
emotion_model.add(Dense(1024, activation="relu"))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation="softmax"))
emotion_model.load_weights("./model-2023-07-17.h5")

cv2.ocl.setUseOpenCL(False)
emotion_dict = {
    0: "   Angry   ",
    1: "Disgusted",
    2: "  Fearful  ",
    3: "   Happy   ",
    4: "  Neutral  ",
    5: "    Sad    ",
    6: "Surprised",
}
emoji_dist = {
    0: "./emojis/angry.png",
    2: "./emojis/disgusted.png",
    2: "./emojis/fearful.png",
    3: "./emojis/happy.png",
    4: "./emojis/neutral.png",
    5: "./emojis/sad.png",
    6: "./emojis/surpriced.png",
}
global last_frame1
last_frame1 = np.zeros((480, 640, 3), dtype=np.uint8)
global cap1
show_text = [0]
```

```python
def show_vid():
    cap1 = cv2.VideoCapture(0)
    if not cap1.isOpened():
        print("Cannot open the camera(0)")
    flag1, frame1 = cap1.read()
    frame1 = cv2.resize(frame1, (600, 500))

    haarcascade_path = (
        cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
    )
    bounding_box = cv2.CascadeClassifier(haarcascade_path)
    gray_frame = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
    num_faces = bounding_box.detectMultiScale(
        gray_frame, scaleFactor=1.3, minNeighbors=5
    )
    for x, y, w, h in num_faces:
        cv2.rectangle(frame1, (x, y - 50), (x + w, y + h + 10), (255, 0, 0), 2)
        roi_gray_frame = gray_frame[y : y + h, x : x + w]
        cropped_img = np.expand_dims(
            np.expand_dims(cv2.resize(roi_gray_frame, (48, 48)), -1), 0
        )
        prediction = emotion_model.predict(cropped_img)

        maxindex = int(np.argmax(prediction))
        cv2.putText(
            frame1,
            emotion_dict[maxindex],
            (x + 20, y - 60),
            cv2.FONT_HERSHEY_SIMPLEX,
            1,
            (255, 255, 255),
            2,
            cv2.LINE_AA,
        )
        show_text[0] = maxindex
    if flag1 is None:
        print("Major error!")
    elif flag1:
        global last_frame1
        last_frame1 = frame1.copy()
        pic = cv2.cvtColor(last_frame1, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(pic)
        imgtk = ImageTk.PhotoImage(image=img)
        lmain.imgtk = imgtk
        lmain.configure(image=imgtk)
        lmain.after(10, show_vid)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        exit()


def show_vid2():
    frame2 = cv2.imread(emoji_dist[show_text[0]])
    pic2 = cv2.cvtColor(frame2, cv2.COLOR_BGR2RGB)
    img2 = Image.fromarray(frame2)
    imgtk2 = ImageTk.PhotoImage(image=img2)
    lmain2.imgtk2 = imgtk2
    lmain3.configure(
        text=emotion_dict[show_text[0]], font=("arial", 45, "bold")
    )

    lmain2.configure(image=imgtk2)
    lmain2.after(10, show_vid2)


if __name__ == "__main__":
    root = tk.Tk()
    # img = ImageTk.PhotoImage(Image.open("../dataset/test/happy/
 ↪PrivateTest_10077120.jpg"))
    # heading = Label(root, image=img, bg="black")
    # heading = Label(root, image=img, bg="black")

    # heading.pack()
    heading2 = Label(
        root,
        text="Photo to Emoji",
        pady=20,
        font=("arial", 45, "bold"),
        bg="black",
        fg="#CDCDCD",
    )

    heading2.pack()
    lmain = tk.Label(master=root, padx=50, bd=10)
    lmain2 = tk.Label(master=root, bd=10)
    lmain3 = tk.Label(master=root, bd=10, fg="#CDCDCD", bg="black")
    lmain.pack(side=LEFT)
    lmain.place(x=50, y=250)
    lmain3.pack()
    lmain3.place(x=960, y=250)
    lmain2.pack(side=RIGHT)
    lmain2.place(x=900, y=350)
    root.title("Photo To Emoji")
    root.geometry("1400x900+100+10")
    root["bg"] = "black"
    exitbutton = Button(
        root,
        text="Quit",
        fg="red",
        command=root.destroy,
        font=("arial", 25, "bold"),
    ).pack(side=BOTTOM)
    show_vid()
    show_vid2()
    root.mainloop()
```

```
Cannot open the camera(0)
```

```
[ WARN:0@43796.046] global cap_v4l.cpp:982 open VIDEOIO(V4L2:/dev/video0): can't
open camera by index
[ERROR:0@43796.047] global obsensor_uvc_stream_channel.cpp:156
getStreamChannelGroup Camera index out of range
```

```
---------------------------------------------------------------------------
error                                     Traceback (most recent call last)
Cell In[23], line 160
    152 root["bg"] = "black"
    153 exitbutton = Button(
    154     root,
    155     text="Quit",
   (…)
    158     font=("arial", 25, "bold"),
    159 ).pack(side=BOTTOM)
--> 160 show_vid()
    161 show_vid2()
    162 root.mainloop()

Cell In[23], line 64, in show_vid()
     62     print("Cannot open the camera(0)")
     63 flag1, frame1 = cap1.read()
---> 64 frame1 = cv2.resize(frame1, (600, 500))
     66 haarcascade_path = (
     67     cv2.data.haarcascades + "haarcascade_frontalface_default.xml"
     68 )
     69 bounding_box = cv2.CascadeClassifier(haarcascade_path)

error: OpenCV(4.8.0) /io/opencv/modules/imgproc/src/resize.cpp:4062: error:␣
 ↪(-215:Assertion failed) !ssize.empty() in function 'resize'
```