基于卷积神经网络的面部表情识别及 emoji 生成项目实验报告

隋建政 2251079 计算机科学与技术

摘要：本项目为一个基于卷积神经网络识别人类面部表情并生成对应表情符号（emoji）的小程序设计，利用 kaggle 上提供的数据集以及深度学习模型库 keras 及进行模型的训练，通过笔记本电脑的前置摄像头实时捕获屏幕中的人类面部表情并利用训练好的模型进行识别，在程序中生成相应的表情符号。

关键词：卷积神经网络；深度学习；人脸识别

一、案例背景介绍

在通信网络飞速发展的今天，人们对于不同通信方式的要求越来越高，电话、短信、视频聊天、社交媒体等新型通信方式层出不穷。在众多的通信方式之间，目前最为新颖、最为天马行空的创意非元宇宙莫属。元宇宙是人类利用数字技术构建，由现实世界映射或超越现实世界，可与现实世界交互的虚拟世界，具备新型社会体系的数字生活空间。

尽管目前元宇宙的很多布局与功能还处于刚刚起步与谋划蓝图的阶段，但依然可以对其中的关键技术进行遐想与逐步的实现。本项目就是一种预期可以应用在元宇宙领域的技术，通过本项目的模型，可以将用户在现实生活中的面部表情，映射为虚拟世界的数字表情符号。通过项目实时捕捉实时生成的特性，无疑可以提高用户在虚拟世界与他人进行通信的真实感。这种人工智能技术可以为用户提供更为细腻的交互体验，使其在虚拟数字世界中也能体验同现实一般时时变化的交互方式。

二、案例相关数据集表述

该项目训练模型所使用到的数据集来自 kaggle 用户 Mana Sambare 创建的开源数据集 FER-2013（https://www.kaggle.com/datasets/msambare/fer2013）。该数据集中的文件全部为像素 48x48 的人脸灰度图，所有的图片按照不同的面部表情总共被分为了七类：愤怒、厌恶、恐惧、高兴、悲伤、惊讶以及面无表情，不同类别的图片被分别放在不同的文件夹中。

数据集中包含训练数据集和测试数据集，其中训练数据集包含 28709 张图片，而测试数据集包含 3589 张图片。数据集中的部分图片如下图所示。

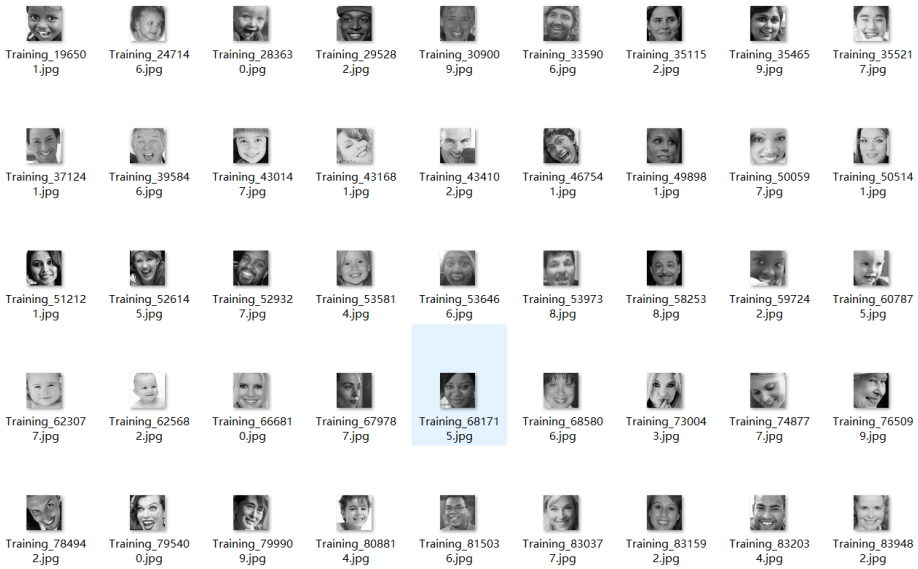

图 1 FER-2013 训练数据集中分类为“高兴”的部分图片

三、案例涉及的模型与方法实现

1、 卷积神经网络

卷积神经网络（Convolutional Neutral Networks）是一种深度学习或类似于人工智能神经网络的多层传感器，常用来分析视觉图像。卷积神经网络的基础架构如下图所示。

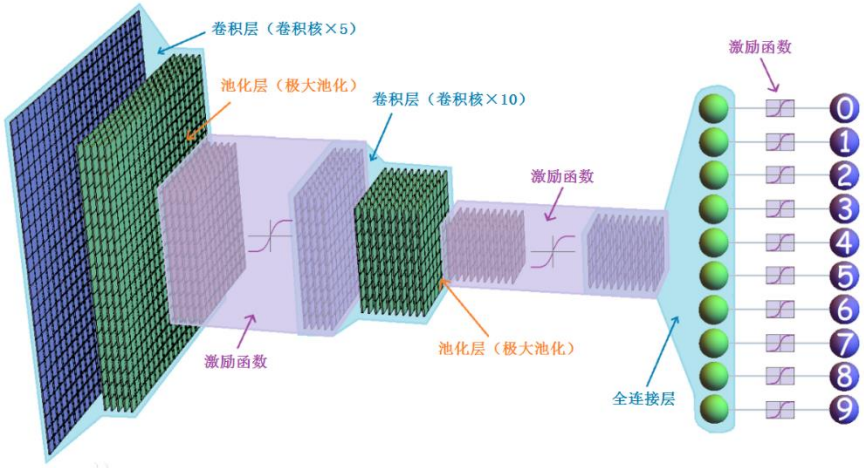

图 2 卷积神经网络的基础架构（图片来源：百度百科）

一个卷积神经网络的主要由以下五层组成：数据输入层（Input layer）、卷积计算层（CONV layer）、ReLU 激励层（ReLU layer）、池化层（Pooling layer）、全连接层（Fully Connected layer）。

在数据输入层中，对原始图像数据进行预处理，包括均值化、归一化、PCA/白化。在卷积层中，对图像进行卷积操作，卷积操作可以捕捉图像中的局部特征而不受其位置的影响。在神经网络中，卷积操作是指将卷积核（滤波器）与原图像矩阵进行逐元素相乘然后相加的操作。其更细节的计算本人曾在《高级程序语言设计》这门课上进行过实现，以下为图片在不同卷积核下进行卷积操作后得到的结果。

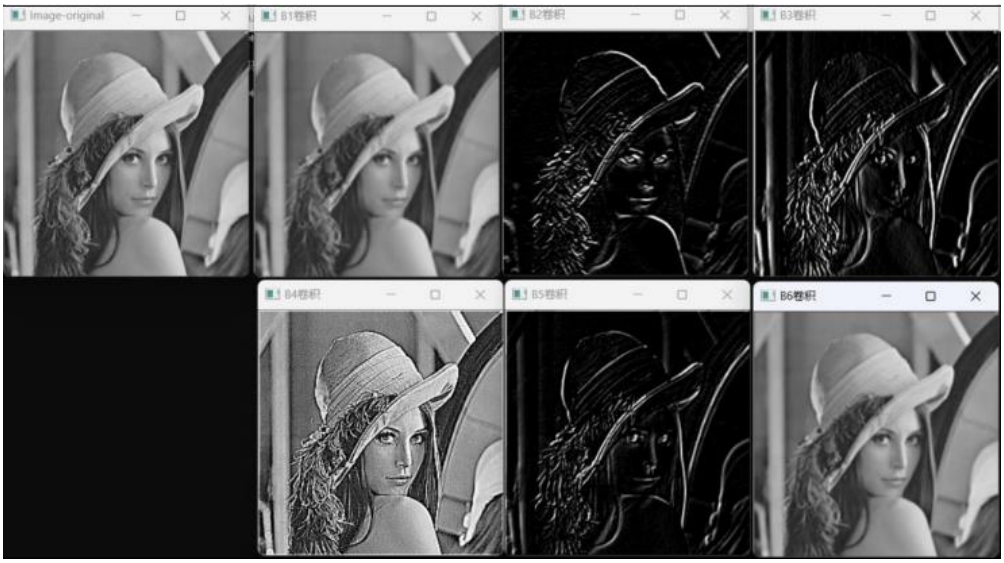

图 3 《高级程序语言设计》大作业矩阵运算中实现的卷积运算的结果展示

在激励层中，我们需要把卷积层的结果进行非线性映射，卷积神经网络采用的激励函数一般为 ReLU（The Rectified Linear Unit/修正线性单元），它的特点是收敛快但比较脆弱，

在该项目中也是使用 ReLU 作为激励函数。池化层夹在连续的卷积层中间，主要是用于压缩数据和参数的量，减少过拟合。在该项目中，池化层的主要作用是压缩图像，使用到的方法是 Max pooling。

为了更好的减少过拟合的现象，在该项目中还使用到了 Dropout 层，池化层的作用是减少特征数量（降维），而 Dropout 层则可以将一部分神经元按照一定概率从神经网络中暂时舍弃，在使用时被舍弃的神经元恢复链接。卷积神经网络中全连接层的连接方式与传统神经网络中的一致，都位于神经网络的尾部，保证两层之间所有神经元都有权重连接。

卷积神经网络的部分在该项目中利用 keras 库中的模型进行实现。

2、 Adam 优化器

Adam 优化器是一种对随机梯度下降（SGD）进行优化的算法工具，其结合了 AdaGrad 和 RMSProp 两种优化算法的优点，通过对梯度的一阶矩估计和二阶矩估计进行综合考虑，计算更新步长。Adam 优化的特点是实现简单、计算高效、对内存的需求小。在很多情况下，Adam 优化器都被默认为工作效率比较优秀的优化器，在该项目中就使用到了 Adam 优化器在模型训练过程中对梯度下降的过程进行优化。

Adam 优化器在该项目中使用 keras.optimizers 中的 legacy.Adam()实现。

3、 分类交叉熵损失函数

交叉熵损失函数是损失函数的一种，经常使用在分类问题中，特别是神经网络实现的分类问题。在该项目中，使用到了交叉熵损失函数的多分类计算，计算方式如下：

$$L = \frac{1}{N}\sum_{i} L_i = -\frac{1}{N}\sum_{i}\sum_{c=1}^{M} y_{ic}\log\left(p_{ic}\right)$$

其中：M表示类别的数量;

y_{ic}表示符号函数（取值为 0 或 1），若样本i真实类别等于 c 则取 1，否则取 0;

p_{ic}表示样本i属于类别 c 的预测概率。

通过上述表达式计算得到损失函数，再利用求导计算全局最优值。

4、 Haar 级联分类器

Haar 级联分类器是 OpenCV 库中用于人脸识别的模型，其理论基础是在 AdaBoost 算法的基础上，使用 Haar-like 小波特征和积分图方法进行人脸检测的检测器。在技术层面，Haar 特征可被笼统的分为三类：边缘特征、线性特征以及中心特征，这些特征利用黑色与白色的矩形表示模板，并可以利用这些特征模板表示人脸的特征。

Haar 级联分类的实现过程可以分为四步：1）使用 Haar 特征进行检测；2）使用积分图对 Haar 特征求值进行加速；3）使用 AdaBoost 算法训练区分人脸和非人脸的强分类器；4）使用筛选式级联把强分类器级联到一起，提高准确率。Haar 级联分类器在该项目中直接使用 OpenCV 包中的 haarcascade_frontalface_default.xml 进行实现。

四、案例实现过程

在该项目中，代码部分总共分为两个文件，分别命名为 train.py 以及 gui.py。其中 train.py 负责模型的训练部分，并将训练结果的数据存入 model.h5 中；gui.py 则负责利用 model.h5 中的训练结果数据快速搭建神经网络，新建一个图形化界面，利用前置摄像头实时捕获视频，通过模型识别表情并实时输出表情符号。

1、导入库

train.py 使用到的库有用于数据处理的 numpy、用于模型搭建的 keras 以及用于捕获图像并处理的 OpenCV（cv2）。

```
# Imports
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.layers import MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import legacy
```

gui.py 中使用到的库除上述几个外，还有搭建图形化界面的 tkinter、用于图像处理的 PIL 以及实现基本的操作系统交互的 os。

```
import tkinter as tk
from tkinter import *
from PIL import Image, ImageTk
import os
```

2、模型训练

运行 train.py 就可以开始模型的训练。在模型训练的过程中，首先对原数据集进行预处理，将图像归一化至[0,1]范围，调整图像大小，并设置图像通道模式为灰度图像，准备了模型所需的输入形式。

数据预处理完成后就可以按照上文中提到的卷积神经网络搭建框架，并利用数据集中的训练数据进行训练。在训练过程中，实时计算损失函数并用测试数据进行检测计算模型准确率，输出在终端里。

```
448/448 [==============================] - 134s 296ms/step - loss: 1.8054 - accuracy: 0.2561 - val_loss: 1.7277 - val_accuracy: 0.3078
Epoch 2/50
448/448 [==============================] - 125s 278ms/step - loss: 1.6312 - accuracy: 0.3642 - val_loss: 1.5197 - val_accuracy: 0.4195
Epoch 3/50
448/448 [==============================] - 123s 274ms/step - loss: 1.5115 - accuracy: 0.4191 - val_loss: 1.4518 - val_accuracy: 0.4418
Epoch 4/50
448/448 [==============================] - 126s 281ms/step - loss: 1.4339 - accuracy: 0.4511 - val_loss: 1.3786 - val_accuracy: 0.4799
Epoch 5/50
448/448 [==============================] - 139s 310ms/step - loss: 1.3774 - accuracy: 0.4769 - val_loss: 1.3418 - val_accuracy: 0.4927
Epoch 6/50
448/448 [==============================] - 128s 286ms/step - loss: 1.3214 - accuracy: 0.5017 - val_loss: 1.2834 - val_accuracy: 0.5148
Epoch 7/50
448/448 [==============================] - 122s 271ms/step - loss: 1.2810 - accuracy: 0.5164 - val_loss: 1.2561 - val_accuracy: 0.5187
Epoch 8/50
448/448 [==============================] - 124s 276ms/step - loss: 1.2475 - accuracy: 0.5295 - val_loss: 1.2311 - val_accuracy: 0.5301
Epoch 9/50
448/448 [==============================] - 125s 279ms/step - loss: 1.2082 - accuracy: 0.5450 - val_loss: 1.1978 - val_accuracy: 0.5407
Epoch 10/50
149/448 [=======>......................] - ETA: 1:18 - loss: 1.1701 - accuracy: 0.5610
```

图 4 模型训练过程中的终端显示

每个训练周期结束后，都会输出损失函数以及模型准确率的数值，可以注意到随着训练的迭代，损失函数的值在不断下降而模型准确率则在不断上升。每个训练周期所需要的时间在 90 秒左右，总共需要进行 50 个周期的训练。

训练结束后，将卷积神经网络中各个神经元之间的连接权重保存在文件 model.h5 中，方便 gui.py 读取。

```
448/448 [==============================] - 148s 330ms/step - loss: 0.5224 - accuracy: 0.8094 - val_loss: 1.1124 - val_accuracy: 0.6217
Epoch 41/50
448/448 [==============================] - 165s 369ms/step - loss: 0.5184 - accuracy: 0.8117 - val_loss: 1.1114 - val_accuracy: 0.6191
Epoch 42/50
448/448 [==============================] - 142s 317ms/step - loss: 0.4912 - accuracy: 0.8221 - val_loss: 1.1272 - val_accuracy: 0.6204
Epoch 43/50
448/448 [==============================] - 175s 391ms/step - loss: 0.4758 - accuracy: 0.8273 - val_loss: 1.1319 - val_accuracy: 0.6207
Epoch 44/50
448/448 [==============================] - 178s 397ms/step - loss: 0.4595 - accuracy: 0.8343 - val_loss: 1.1310 - val_accuracy: 0.6296
Epoch 45/50
448/448 [==============================] - 180s 402ms/step - loss: 0.4466 - accuracy: 0.8390 - val_loss: 1.1457 - val_accuracy: 0.6261
Epoch 46/50
448/448 [==============================] - 184s 410ms/step - loss: 0.4357 - accuracy: 0.8424 - val_loss: 1.1658 - val_accuracy: 0.6217
Epoch 47/50
448/448 [==============================] - 173s 387ms/step - loss: 0.4147 - accuracy: 0.8510 - val_loss: 1.1570 - val_accuracy: 0.6265
Epoch 48/50
448/448 [==============================] - 178s 398ms/step - loss: 0.4008 - accuracy: 0.8545 - val_loss: 1.1657 - val_accuracy: 0.6229
Epoch 49/50
448/448 [==============================] - 147s 328ms/step - loss: 0.3898 - accuracy: 0.8592 - val_loss: 1.1774 - val_accuracy: 0.6253
Epoch 50/50
 99/448 [=====>........................] - ETA: 1:50 - loss: 0.3752 - accuracy: 0.8674
```

图 5 最后几组训练周期的输出

3、 模型测试

在 train.py 中，还附带有一个模型训练完毕的测试模块，该模块也可以打开电脑的前置摄像头，但只能在终端中以文本形式显示识别到的表情并没有更直观的图形化界面。仅仅是做初步的测试功能。

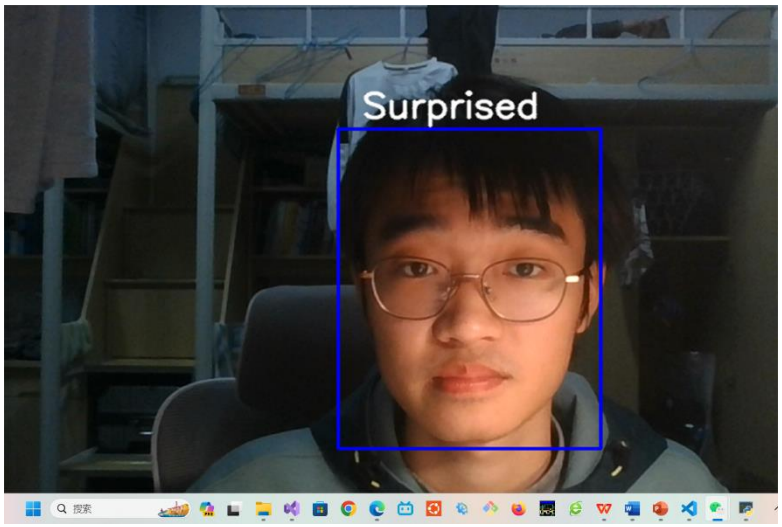

图 6 测试结果：惊讶

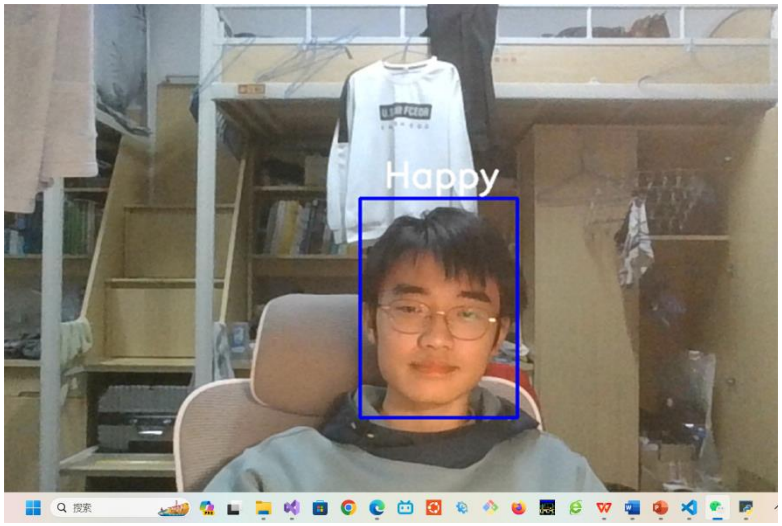

图 7 测试结果：高兴

4、 功能演示

运行 gui.py 打开图形化界面进行结果的演示。

在这个界面中，电脑的前置摄像头将会开启，并在页面左侧显示捕捉到的视频。Haar 级联分类器会对视频中的人脸进行检测，并通过已训练好的模型输出识别到的表情类别，在图形化界面的右侧生成对应的表情符号。

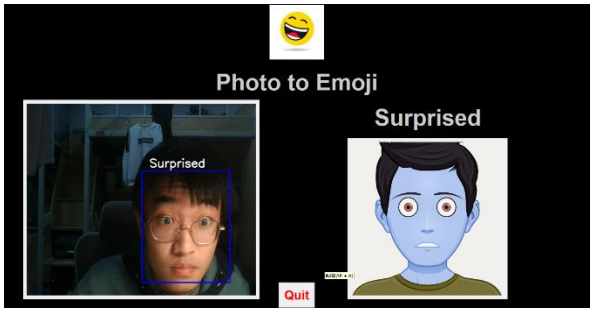

图 8 功能演示：惊讶表情识别

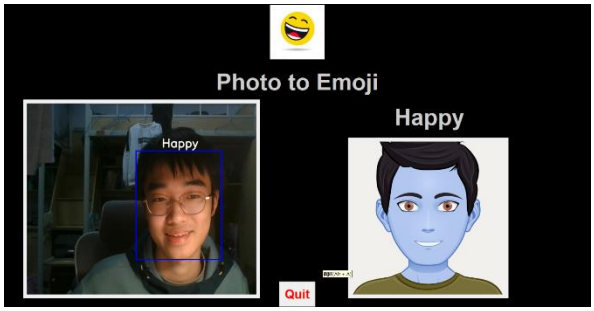

图 9 功能演示：高兴表情识别

图 10 功能演示：愤怒表情识别

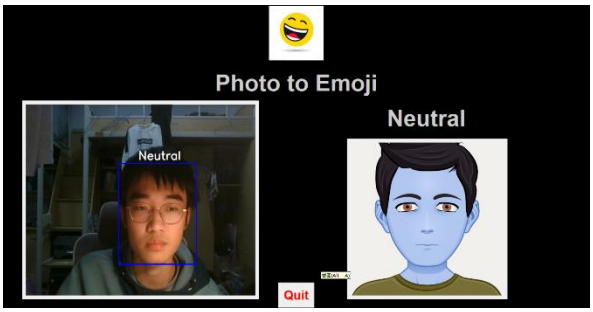

图 11 功能演示：面无表情识别

五、改进思路

1、 表情符号库的扩展

在该项目中表情符号库为提前配置好的图片库，选择较为单一。可以考虑增加多种不同风格的表情符号库，供用户自行选择，提高程序的自由度以及娱乐性。

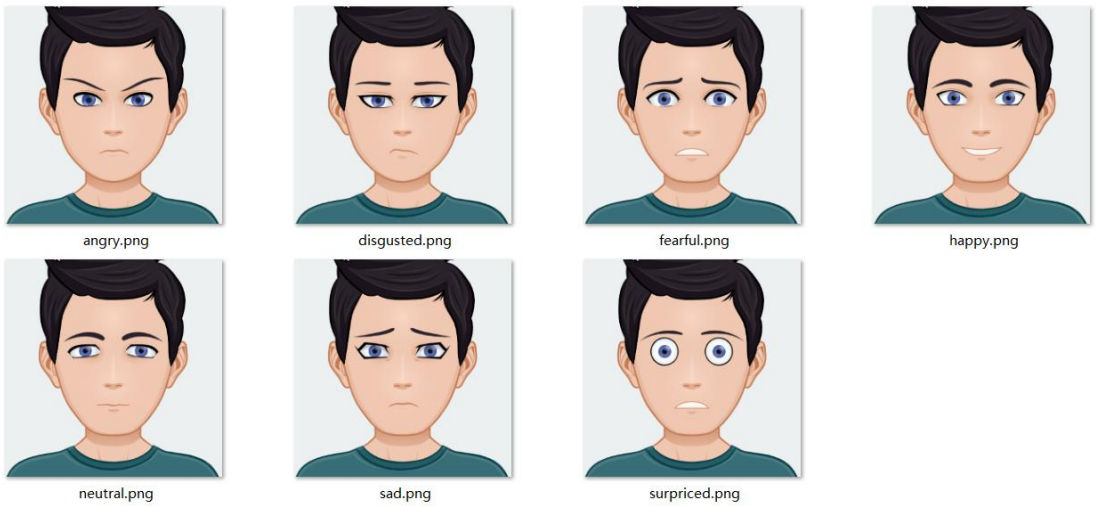

图 12 原本的表情符号图库

用户在使用时可以使用我们提供的其他表情包或自定义一套表情包，例如下面全部由黄豆构成的表情包。

图 13 一套自定义的“黄豆”表情包

2、 更多面部表情的扩展

尽管在该项目中所使用到的数据集已经包含了七种不同类型的表情数据，但这些在人类日常生活交往中还是远远不够的。为使项目提供的表情更为丰富，可以搭建更为丰富的数据集，囊括更多不同类型的表情数据，如尴尬、得意、搞怪等等，但这其中很多表情之间会有所重合，例如得意与高兴的表情，似乎存在很大的重合区域。可以考虑引入对话情景的功能，例如利用收音模块分析用户前段时间的发言情况，利用发言预测当前的情感状况，对纯面部识别难以区分的表情做更细微的区别。当然，这些就都是更为复杂的功能了，目前还没有实现。

3、 不同程度面部表情的扩展

此外，在实际情况中，同一个表情也可能存在不同的程度。为使表情生成更为有趣和具有连贯性，我们可以对同一类表情的不同程度也进行分类。例如下图中的三个表情符号分别代表了不同程度的愤怒情绪。

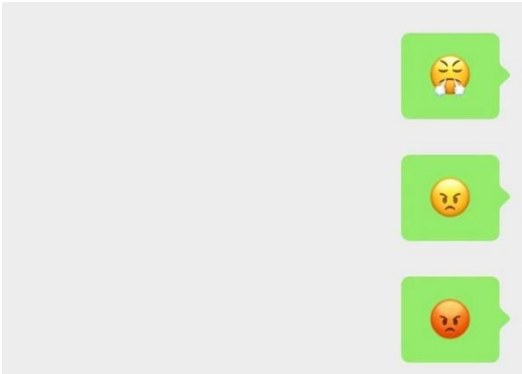

图 14 三种不同程度的愤怒表情

但如此一来对于数据集和模型的要求就更为高了，因此目前也是没有进行实现的。

六、结论

该项目实现了一个基于卷积神经网络的人脸表情识别及对应表情符号生成的小程序，利用 kaggle 网站上的开源数据集作为训练和测试数据，通过 python 及相应的库实现相应功能。该项目的实现可以看作当今虚拟主播虚拟形象的技术蓝图，也为未来元宇宙的功能进行了展望。

在完成该项目的过程中，本人初步学习了卷积神经网络的相关知识，了解了多种算法优化以及数据处理的方法。同时还学习了相应代码库的操作方式，用 python 代码成功搭建了卷积神经网络并对摄像头捕获的视频帧进行了识别，完成了预期的功能。

现如今人工智能已经越来越深入我们生活的方方面面，而在未来越发强大与智能的 AI 一定会让我们的生活更加便利，更富有乐趣与创造力，让所有人都能享受到科技的便利。尽管本项目只是深度学习领域一个小小的应用，但其也无疑是本人迈入人工智能大门的小小纪念，希望在未来还可以继续在领域内深耕，学习更多算法与 AI 模型并实现自己曾经天马行空的想象。

七、参考文献

[1] 科技日报 最近大火的元宇宙到底是什么？最近大火的元宇宙到底是什么？ (qq.com) 2021-09-13

[2] Charmve computer-vision-in-action: A computer vision closed-loop learning platform where code can be run interactively online https://github.com/Charmve/computer-vision-in-action 2021-04-29

[3] hjimce 深度学习（二十二） Dropout 浅层理解与实现 http://blog.csdn.net/hjimce/article/details/50413257 2015-12-27

[4] Emerson 简单认识 Adam 优化器 简单认识 Adam 优化器-知乎 (zhihu.com) 2018-01-07

[5] 飞鱼 Talk 损失函数|交叉熵损失函数 损失函数|交叉熵损失函数(zhihu.com) 2022-03-28

[6] lowkeyway （四十八）Haar 级联检测器 （四十八）Haar 级联检测器 - 知乎 (zhihu.com) 2020-12-31

八、附录

项目打包存放在阿里云盘中 名称为 2251079_隋建政.zip

云盘链接为 https://www.alipan.com/s/YWvgND4rKbS

注：在上传该文件时，阿里云提示.zip 类型文件无法分享，因此采用网上解决策略创建自解压格式压缩文件并将后缀名改为.zip，修改过后成功上传。但不确定是否能下载并解压缩成功，如老师遇到问题请与本人联系，本人手机号 15522839103，微信已与老师添加，微信名 Witness the miracle。