

**机器学习课程设计工程报告**



**课 程：** 机器学习基础

**姓 名：** 刘嘉伟

**学 号：** 2018213106

**完成时间：** 2020年12月24日

一． 工程摘要与每人贡献

该工程是一个实时的性别分类与表情识别系统，通过opencv训练好的Haar特征级联检测器从照片中提取出人脸，然后通过Pytorch我自己训练好的性别分类器来检测性别，通过表情分类器来分类表情，所展示的表情只有三种，高兴，正常和难过，但实际训练的表情有7种。该工程通过PyQt整合成了一个可以选择照片或视频或实时检测的GUI图像界面的软件。

该工程由我自己一人独立完成。

二、研究背景与意义

性别分类和表情分类都是机器学习领域的经典问题。两者都需要对输入的人脸图像特征进行提取，所以也是一个机器视觉领域的问题。

性别分类是一个二分类问题，比较基础，许多机器学习的方法都能很好的完成这个任务，如SVM，感知机，卷积神经网络等等，并且识别的准确率一般都可以达到90%以上，所以性别分类在机器学习领域已经不算是一个难题，准确率也相当高。本工程的性别分类器是用Pytorch预训练的resnet18来进行迁移学习，将最后一层的全连接层输出改为2个，分别用来表示男和女的概率大小。Resnet18是一个经典的残差神经网络，其中18表示的是带有参数的网络层数有18层，包括卷积层和全连接层，但不包括池化层和归一化层，因为这两层是不带参数的。

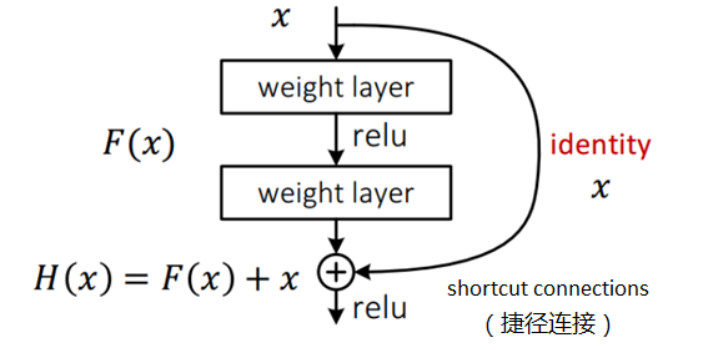
表情分类是一个多分类问题，涉及的方面就不光是机器视觉和机器学习了，由于人的表情是很难划定界限的，所以可能还与心理学等领域相关，即使让一个真正的人来对某个表情进行分类，也有可能分错的。要让机器能正确分类的前提是让一个人来分类，能准确地分类。正因为人的表情是很难分类的，因此目前大多数表情分类算法的准确率都不是很高，数据集也不一定很准确，目前网上看到最好的表情识别论文也只达到了百分之七八十左右。表情分类常用的数据集有CK+，fer2013,JAFFE等等，其中CK+是在实验室采集的人脸数据，所以可能数据集的质量会相对较高一些，fer2013是kaggle的一场表情分类比赛使用的数据集，该数据集包含了七类表情，由35886张人脸表情图片组成，其中，训练图（Training）28708张，公共验证图（PublicTest）和私有验证图（PrivateTest）各3589张，每张图片是由大小固定为48×48的灰度图像组成，共有7种表情。本工程使用的就是fer2013，使用的网络是一个卷积神经网络，网络结构是参考LeNet进行修改的。LeNet是一个用来进行手写识别数字分类的经典网络，由于数字分类和性别分类都是多分类任务，并且数据集里的图像大小也都差不多，所以我就想到对这个网络进行改进来来完成表情分类。

三、模型方法

该工程分为人脸检测和表情识别，因此分两部分讲解模型。

性别分类模型：

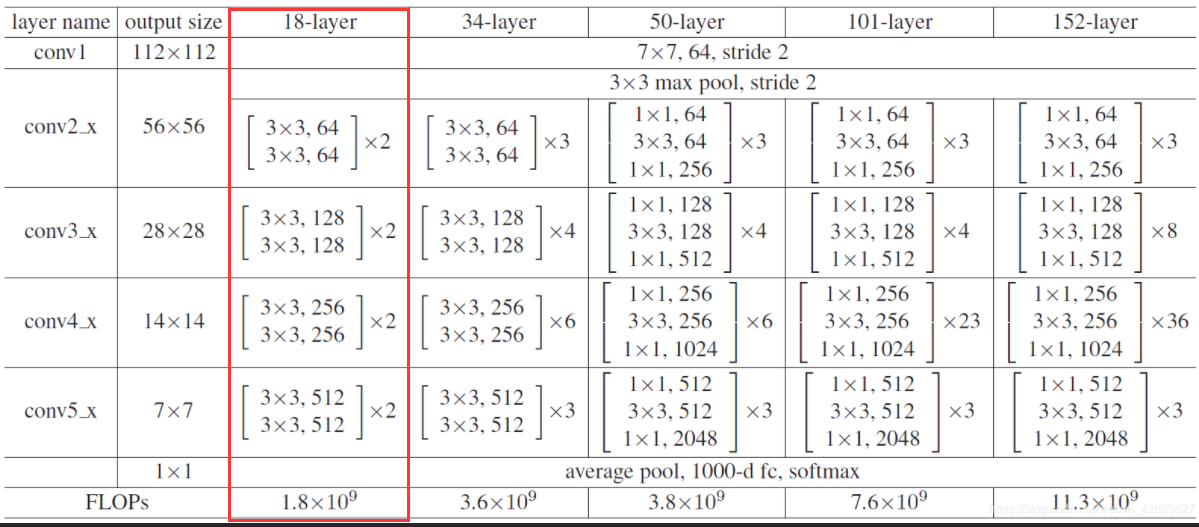
性别分类使用的是resnet18残差神经网络。下面先介绍一下残差神经网络是什么。神经网络层数比较高时，就很难被训练，容易出现梯度爆炸、梯度消失等问题。残差网络就是一种跳跃连接的网络，它通过将前面的激活值跳过中间的网络层而直接传到更后面的网络中，来避免梯度爆炸和梯度消失，训练更深的网络。

**残差块的结构**

由上图可以看到，输入一个然后通过两层带权重的层后，并不直接把结果输出，而是将结果加上输入的，构成来输出。这样做的目的是至少不让深层次的网络输出结果变差，最差情况就是不学习，F(X)=0，不会出现变差的情况。

于是，ResNet相当于将学习目标改变了，不再是学习一个完整的输出，而是目标值H(X)和x的差值，也就是所谓的残差F(x) = H(x)-x，因此，后面的训练目标就是要将残差结果逼近于0，使到随着网络加深，准确率不下降。若F(x)与x维度相同，则直接相加，若维度不相同，可以使H(x)=F(x)+Wx，通过一层卷积来调整x的维度，使得能与F(x)相加。

下面介绍一下resnet18网络结构。



resnet18的结构式上图中用红框画出的那一部分，输入的图像维度是，

其中一个残块的结构如下：

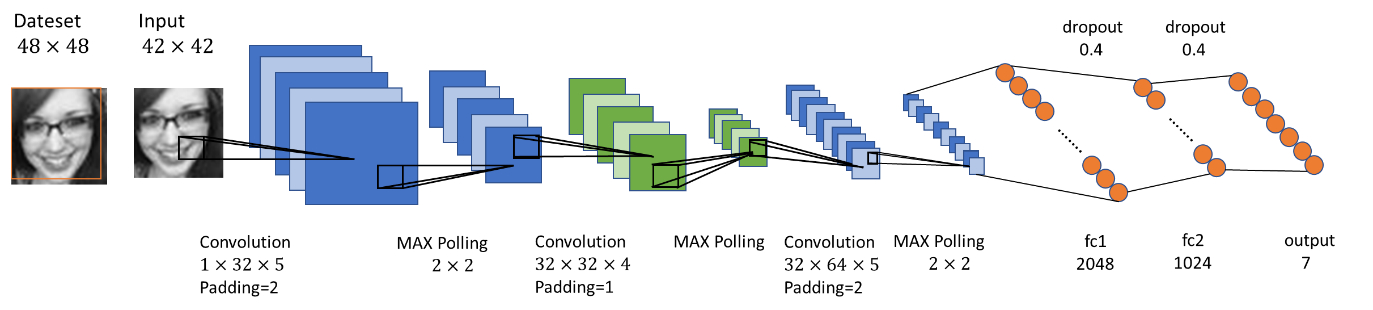
Resnet block1

Basic block

Basic block

经过四个残差块后，使用一个全连接层，将所有输出的特征扁平化，最后输出的维度可由自己决定。在本工程中，使用的是预训练好的resnet18，然后将输出层改为2，就能区分男女。

表情分类模型：



该模型是参考lenet模型设计的，通过设定一系列参数，将图像通过卷积层和池化层后，变成2048个特征参数，通过全连接层，最后输出7个值，来判断是输出表情是哪类。具体每层输出的图像大小如上图所示。该图是训练7个表情时的流程图，而在工程实际应用中，为了模型精确度的提升，将最后7个输出改为了3个，并且只保留了训练集中的三类，分别是开心、正常和难过。

四、系统设计

一、性别分类部分：

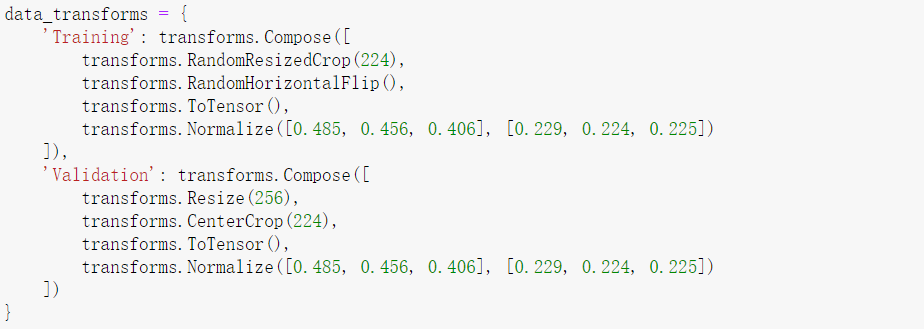
1、本工程所使用的训练集为Kaggle比赛的公开数据集，其中包含了28708张训练数据。我只从原数据集提取了部分数据进行训练，三千张男图片，三千张女图片。



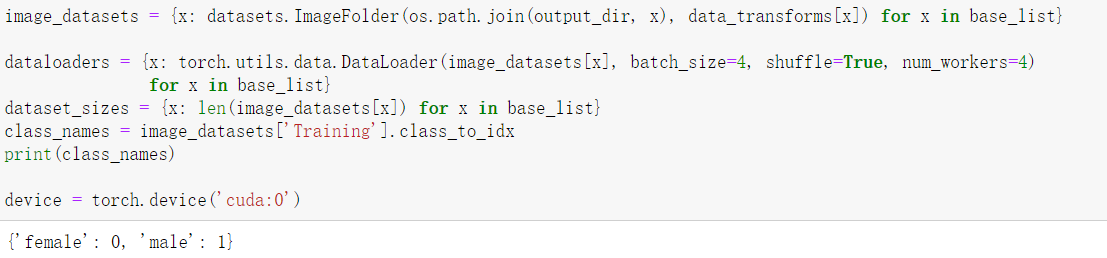
2、分别对训练数据和测试数据进行变换。

对训练集进行的增强变换有：将图片随机裁剪并resize到224\*224的大小，将图片随机水平翻转，然后将图片的ndarray转化为Pytorch的torch格式向量，并进归一化。

对测试集进行的变换有：resize到256大小，中心裁剪到224，然后转化为torch格式向量并归一化。

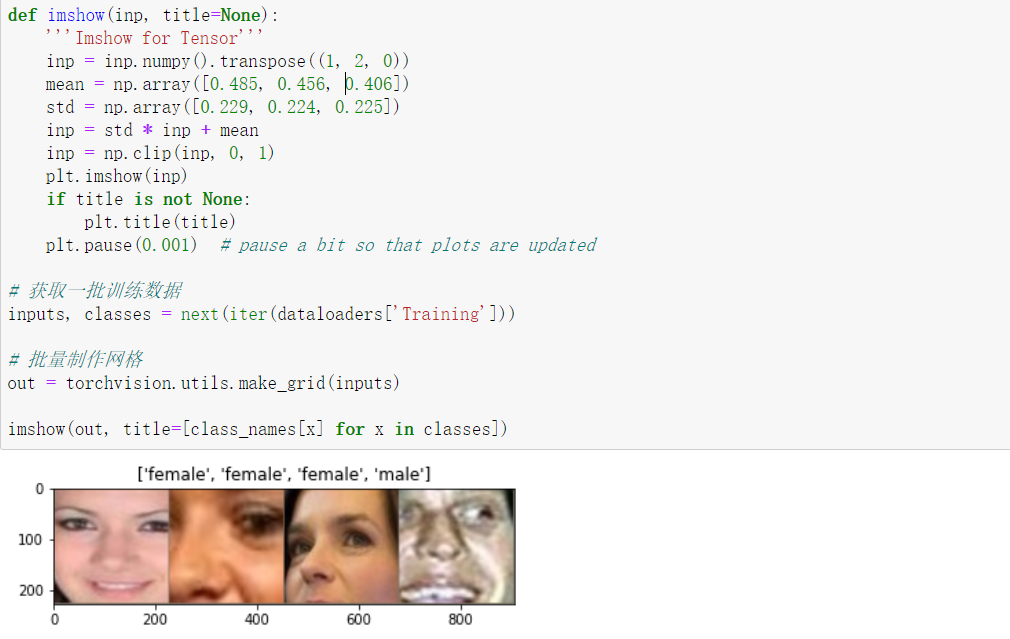


3、通过datasets.ImageFolder读取标准格式的文件夹。Pytorch会自动对数据集各个类别进行分类，每个子文件夹作为一类数据，自动编号，通过.class属性可以查看具体每一类对应哪个编号。

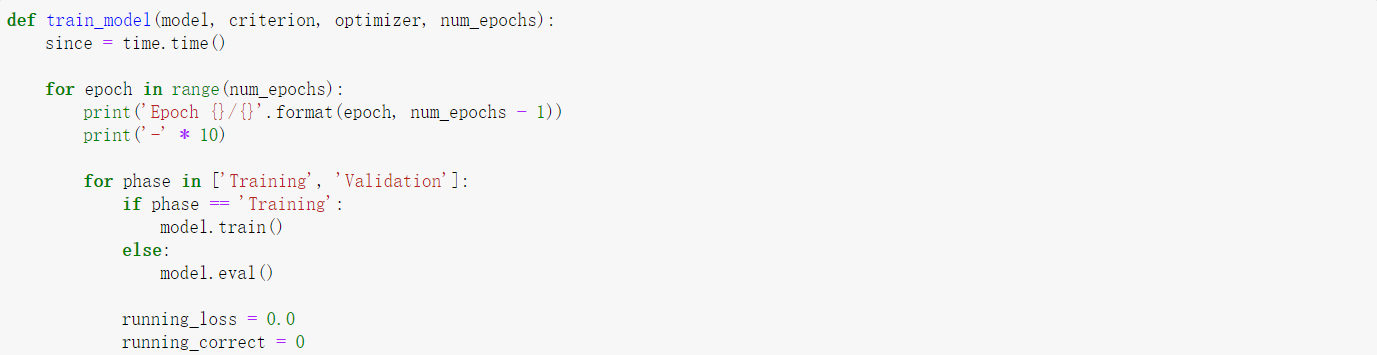


可以看到female对应的时0编号，male对应的是1编号。

4、展示训练数据集的前几张图片。由于Pytorch读取rgb格式图片得到的向量维度是，而matplotlib展示的图片维度是，所以，要将图片向量的维度进行变化，交换1，2维度和0维度。

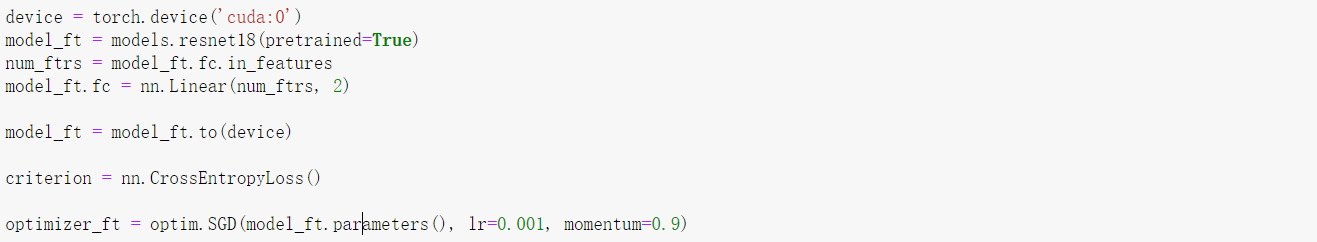


5、训练函数定义。Pytorch的模型有train和eval两种模式，如果是对模型进行训练，需要转换到train模式，如果是对模型进行评估，用eval模式，这两者的区别在于eval模式下，不会计算梯度，增加运算速度。也不会进行dropout和batch normalization，因为如果测试还使用这两者的话会使得结果不一致。 模型的评估在每个epoch中评估一次，也就是每训练完一个epoch，就对该模型在测试集进行评估一次。并在每轮输出误差和迭代时间。





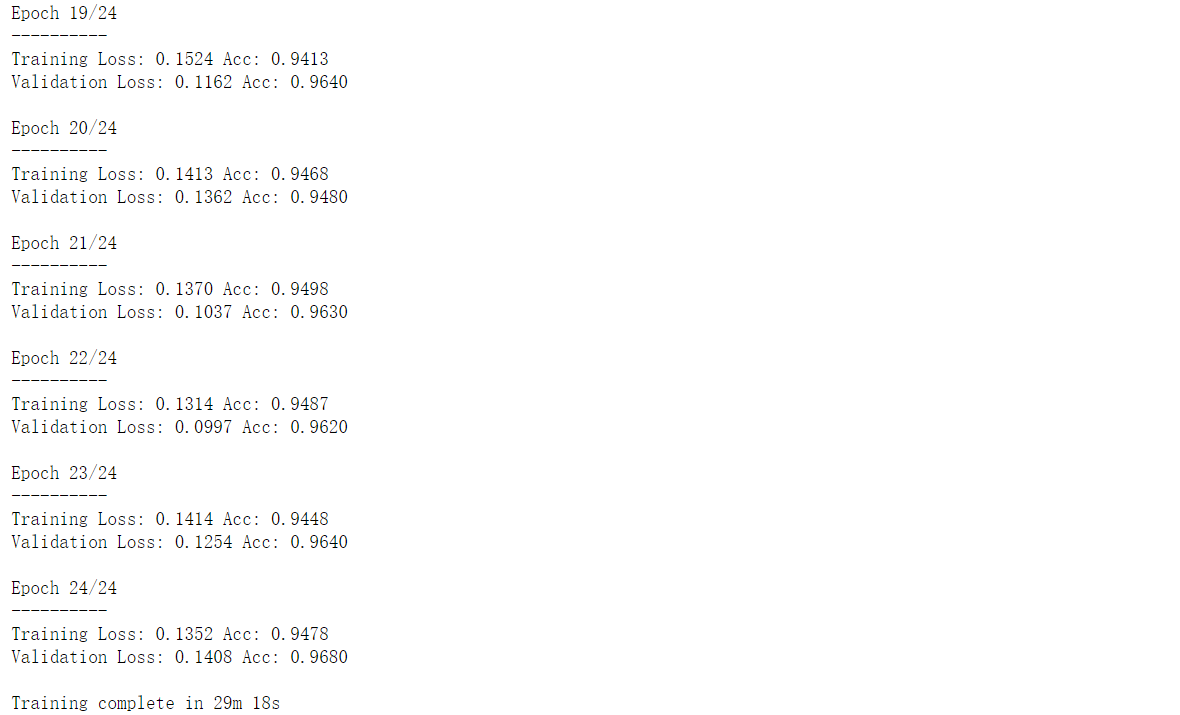
6、定义模型、损失函数、优化器。这里的损失函数使用的是交叉熵损失函数，优化器使用的是SGD梯度下降随机梯度下降优化器。模型的定义只需要调用Pytorch预训练好的resnet网络，然后将它的fc层输出改为2，也就是需要预测的结果类别数，男和女。



7、优化模型。通过第5步定义的优化函数，对第六步定义的模型进行优化。



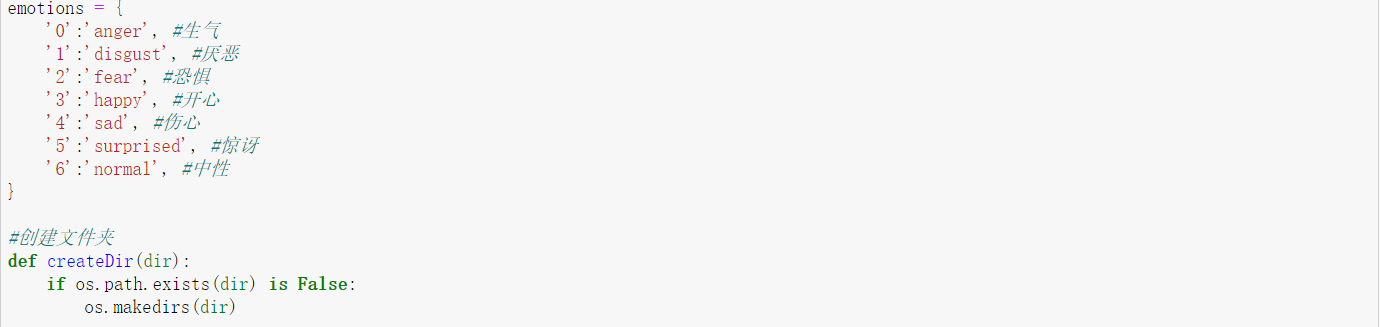
8、训练结果展示



可以看到最后在训练集的精度由94.78%，在测试集的精度有96.80%，可以说是比较高的一个精度了。

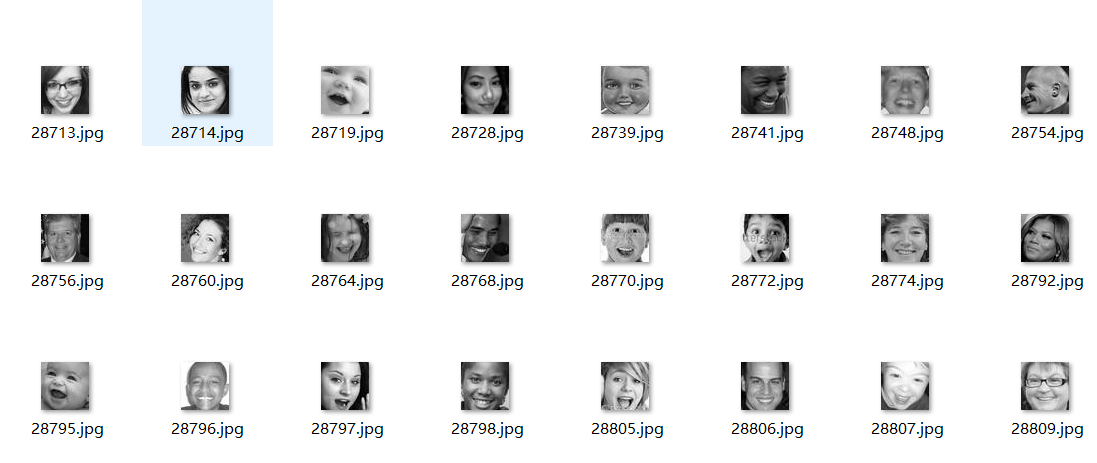
二、表情分类部分：

1、该部分所使用的数据集是Kaggle比赛中的一个公开表情数据集fer2013，其中包含了7类表情。由于一开始参数没有调整好，导致每次训练结果都不收敛，我误以为是类别太多导致的，所以之后我只保留了3类表情进行训练，发现还是不收敛，就在找其他原因。最终发现是学习率调的太高的问题，导致网络不收敛，并不是类别过多的问题。一开始学习率设为0.01导致不收敛，后来将学习率改为0.001才解决了不收敛的问题。不过之后就没有用7类表情进行训练，一直使用的是3类表情。数据集的生成代码如下，要将fer2013的csv文件转化为图像文件。

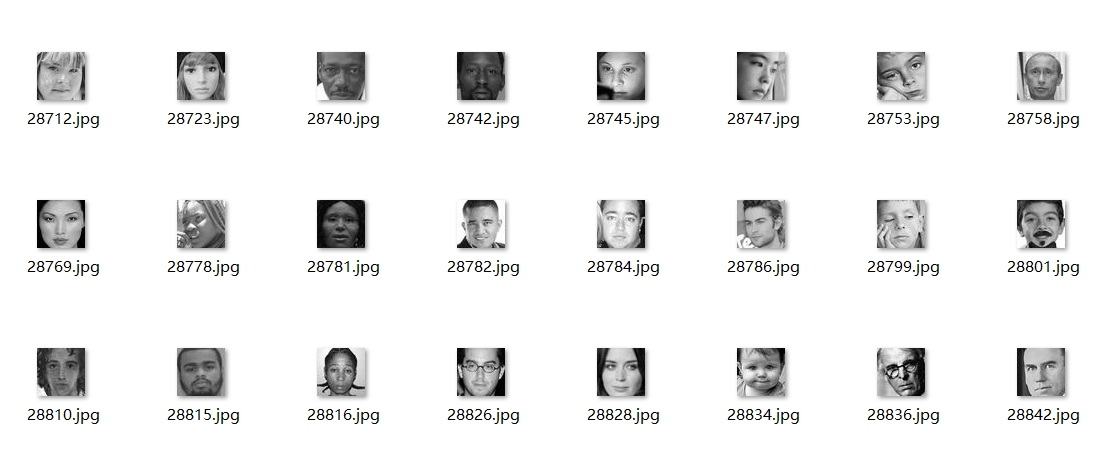




数据集展示：



**happy**

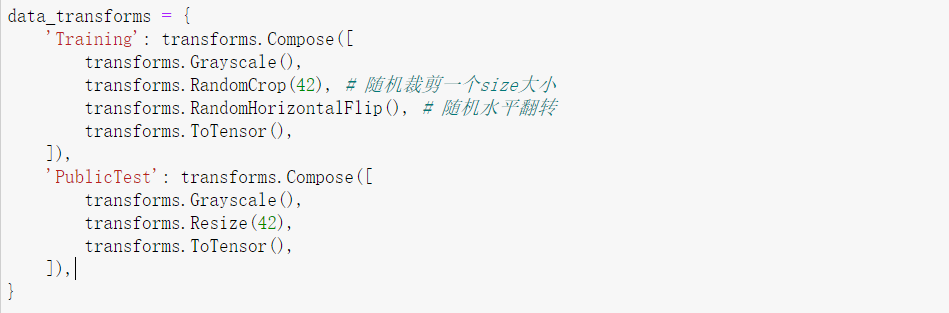


**normal**



**sad**

2、定义数据变换。训练集的数据变换有转化为单通道的灰度图、随机裁剪到42\*42的大小、随机水平翻转和转化为torch的tensor向量，测试集的数据变换有转换为灰度图、resize到42\*42和转化为torch的tensor向量。



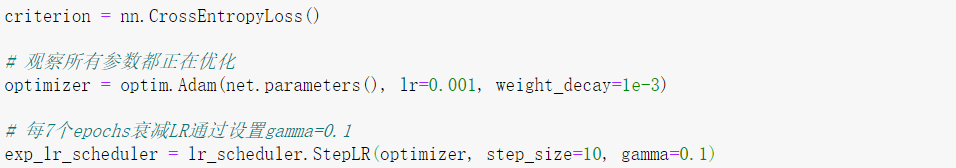
3、数据集的展示，与之前的性别分类流程一样。



4、卷积网络定义。按照之前介绍的模型结构图，通过Pytorch的nn.Model继承，定义一个卷积网络。需要重写前向传播的代码。



5、优化器定义，这里使用的是adam优化器，并且使用了学习率规划器lr\_scheduler.StepLR，每训练十轮就将学习率变为十分之一，防止训练到最后模型不收敛。

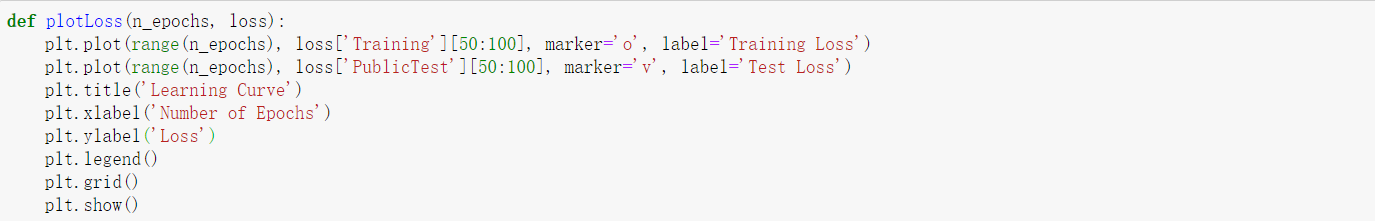


6、训练函数定义，主要流程与之前的性别分类一致，唯一区别在于每迭代完一轮后需要对lr\_scheduler进行更新，每十轮更新一次。在本函数中，还添加了记录损失值的代码，以便训练完成后生成训练图像。

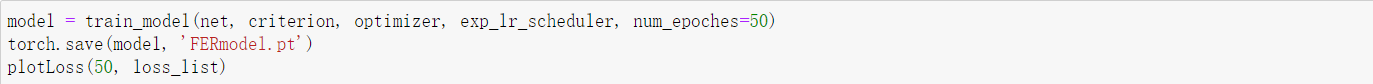




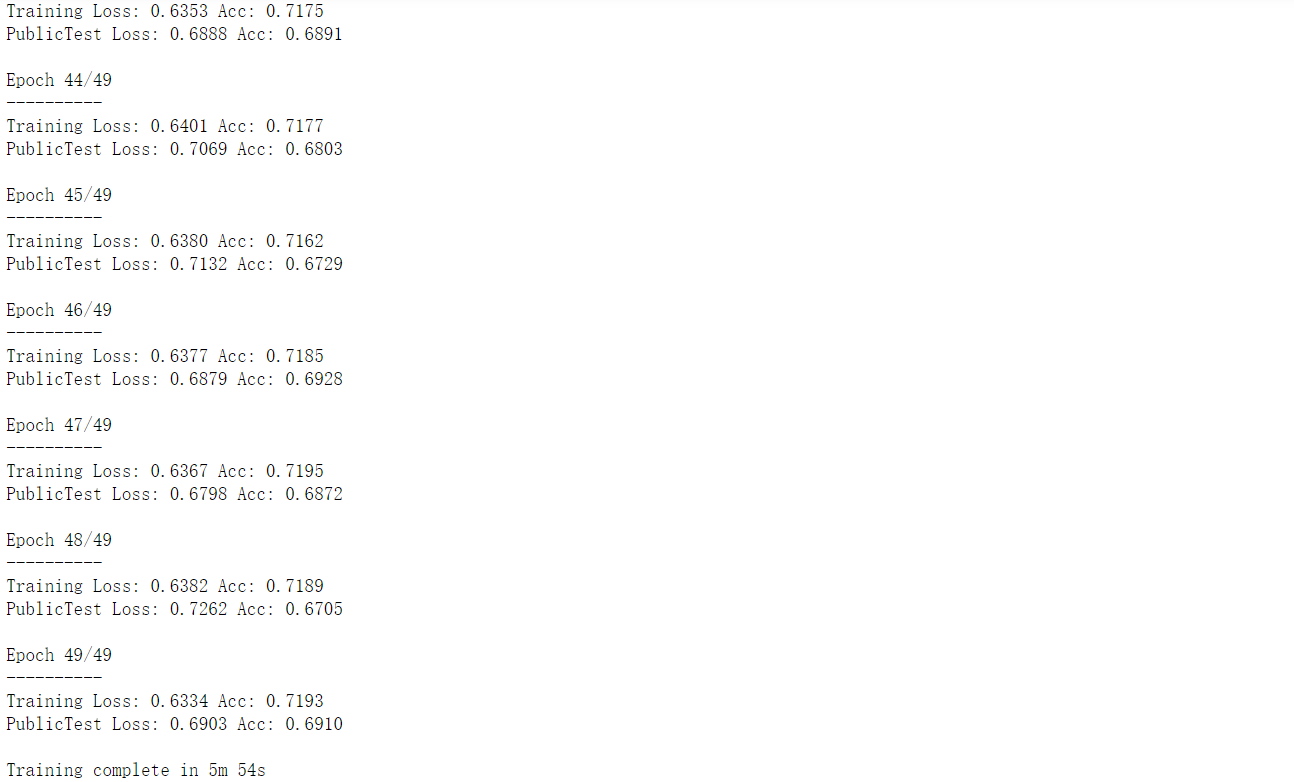
绘制损失loss的图像



7、对模型进行优化并保存，绘图



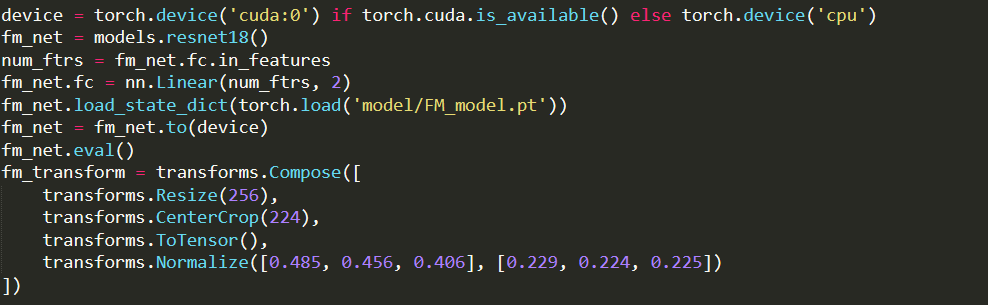
8、训练结果



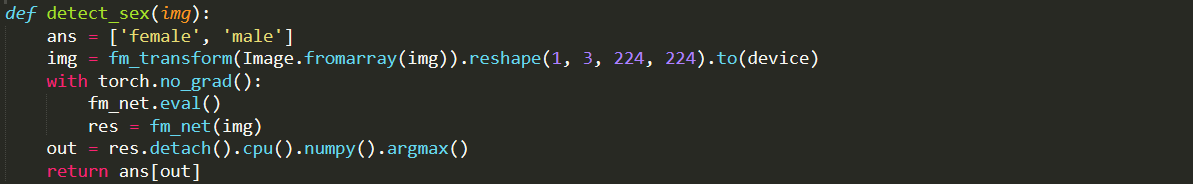
由于训练集太大，自己的电脑跑起来很慢，所以在colab上进行训练，batch\_size设为64，六分钟左右便训练完毕。

三、前端界面PyQt部分

1、图片检测部分。在检测模块中定义之前训练的两个网络，然后加载之前训练好的权重，按照之前定义的transform对检测图像进行变换，然后通过网络来预测，并返回结果给前端展示模块。



**性别分类器的网络定义并加载参数**

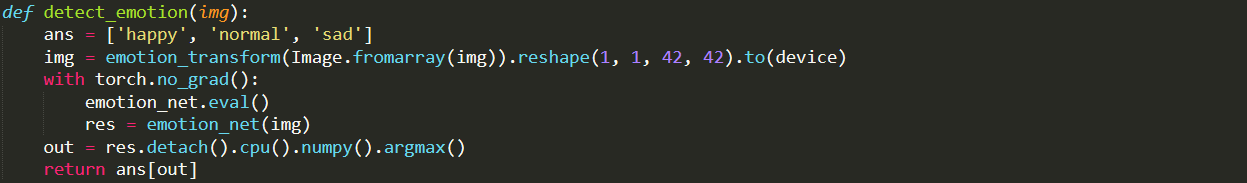


**性别检测函数**

性别检测输入ndarray格式的图片，返回性别。



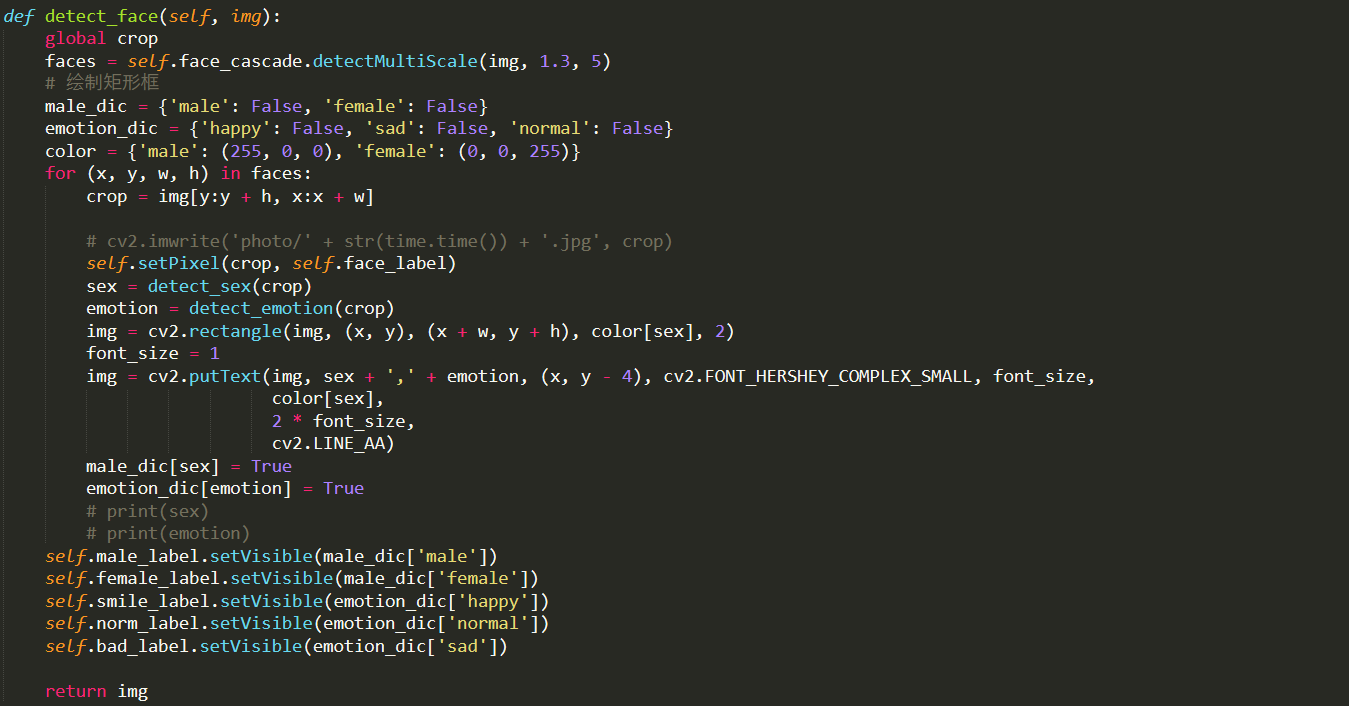
**表情分类网络定义并加载参数**



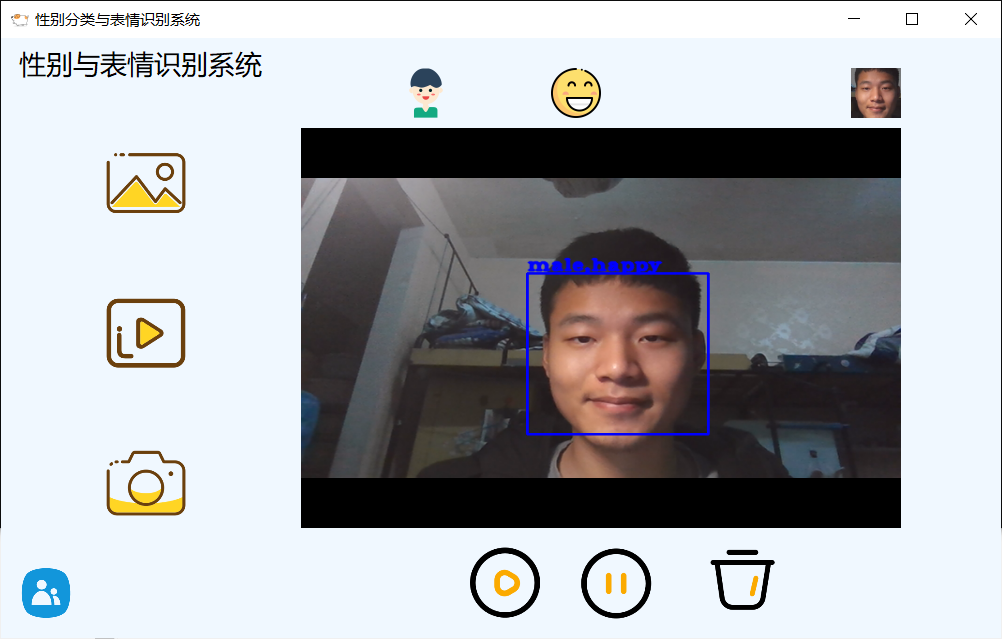
**表情检测函数**

输入一张ndarray格式的表情图片，返回表情。

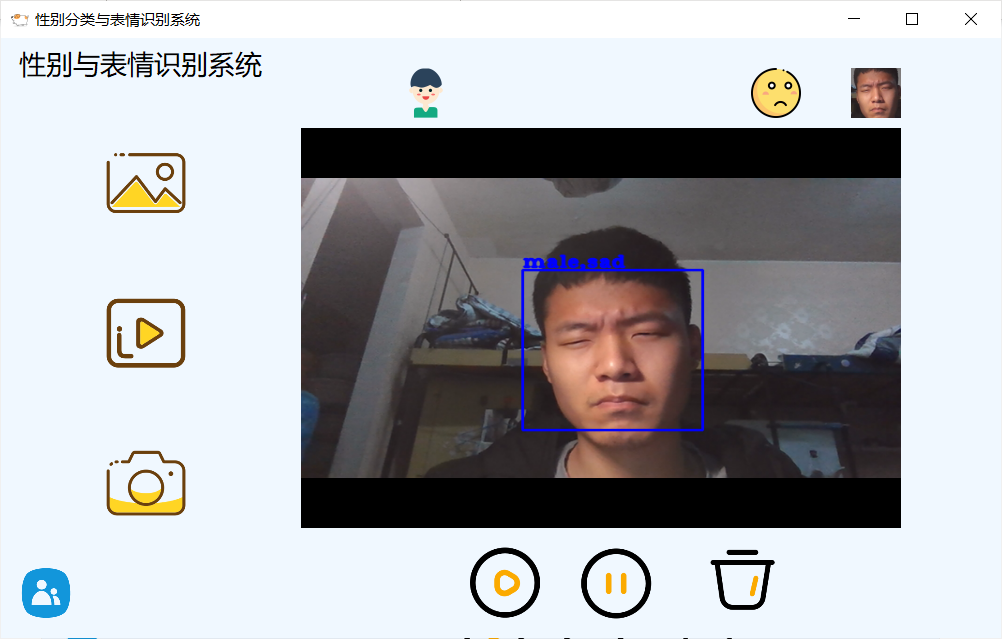
2、前端展示部分。前端展示定义了三种模式，图片检测，视频检测和摄像头检测。三个部分的思路是一样的，如果是图片检测，就直接通过opencv读取选择的图片，然后通过opencv自带的haar级联人脸检测器，检测图片中的人脸，然后将人脸送到上面所述的图片检测模块检测性别和表情。对于视频检测或摄像头检测，需要用opencv打开视频流，读取视频流中的每一帧图像，对每一帧图像用相同的方法进行检测，然后再界面输出显示结果即可。图片检测函数的定义如下：



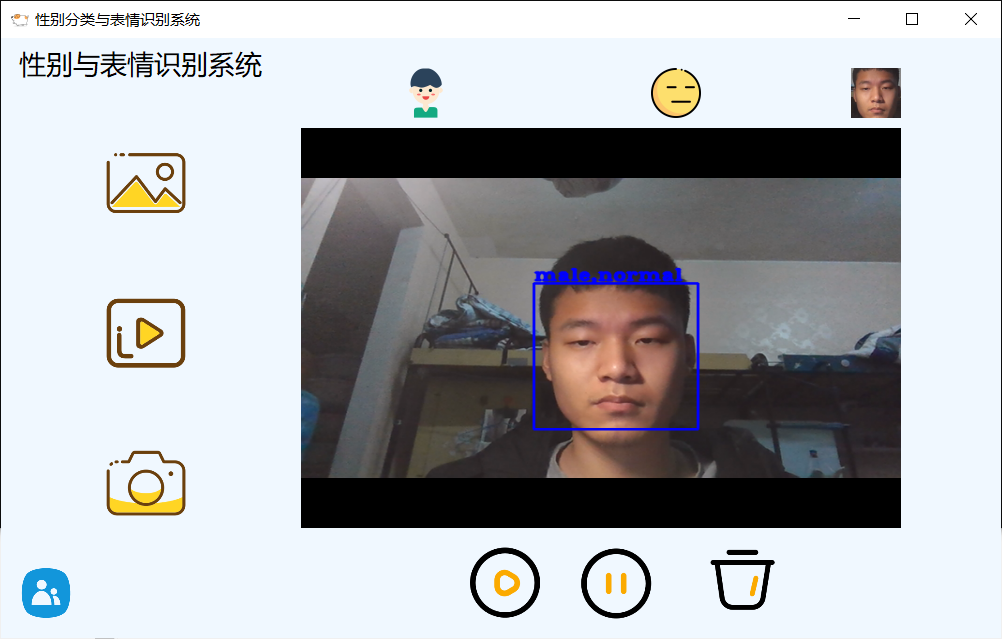
输入一张人脸图像，然后调用face\_cascade的级联分类器，定义绘制的框和文字的颜色，然后将级联分类器获得的图像位置截取下来送到检测函数中，得到结果后绘制人脸框和文字。最后设置检测出来的图标为可见。效果如下：



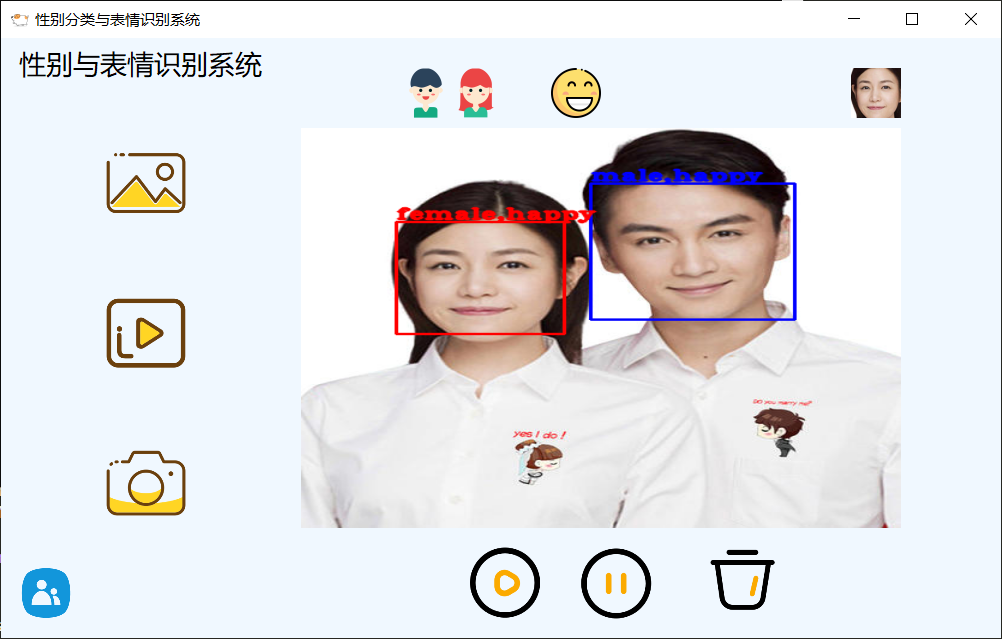
**微笑**



**悲伤**



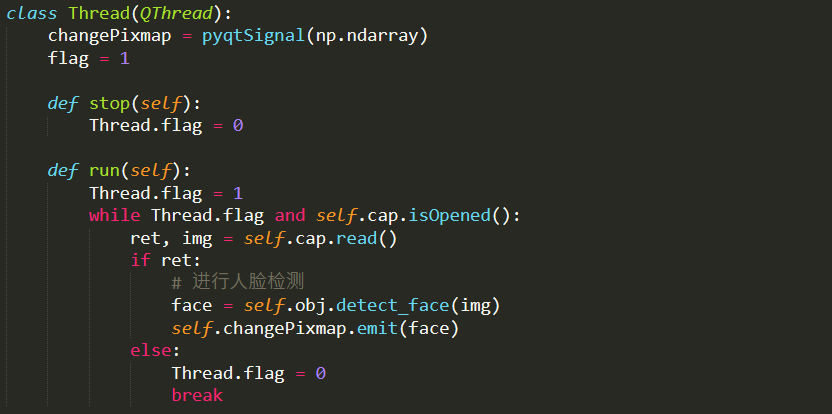
**正常**



**男女，微笑**

上面显示出来的是男女的图标和表情状态，右边小图标是截取下来的人脸。由于只设置了一个label所以只能显示后检测到的人脸。

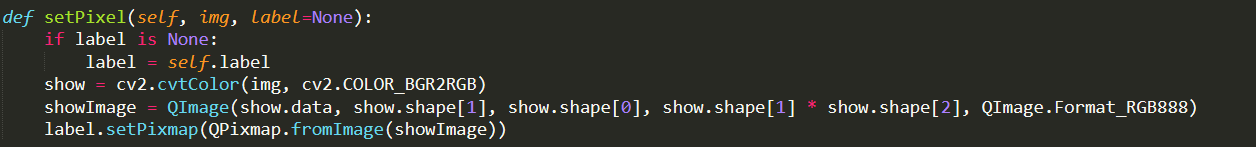
前端的展示使用的是多线程技术，读取视频流后就将图片送入一个线程中进行检测，否则在检测的过程中所有的按钮都不能使用。我还做了暂停和开始的功能，花费了我不少时间来想该怎么实现。由于PyQt的线程类没有杀死线程的功能，所以不知道该如何停下线程，如果直接关闭了由于线程没有关闭，会留在内存内不释放，进行多次检测的话每次会生成一个线程，那么会造成内存泄漏问题。为此，我在线程内定义了一个flag标志，检测视频流的时候都需要判断flag是否为true，如果为真才进行检测，如果用户点击了暂停，那么将flag设置为false，就会由于判断条件失败暂停线程。当用户点击继续按钮的时候，我再将该flag标志设置为true，并重新运行原本的线程，那么这个线程就可以一直被利用，不会造成内存泄漏的问题，如果多次检测，只需要每次切换的时候将flag设置为false，然后开始检测的时候，更换视频流对象，然后继续将flag设置为true，还是能继续利用该线程进行检测，不浪费内存。线程定义如下：



在该函数中先调用检测人脸函数，得到图像的人脸，然后调用识别函数，在图像上绘制对于的方框和文字，并设置界面的展动态变化，就实现了实时检测。其中的cap是利用Python的在类外部定义类成员的机制实现的。下面是对于视频流的检测函数：



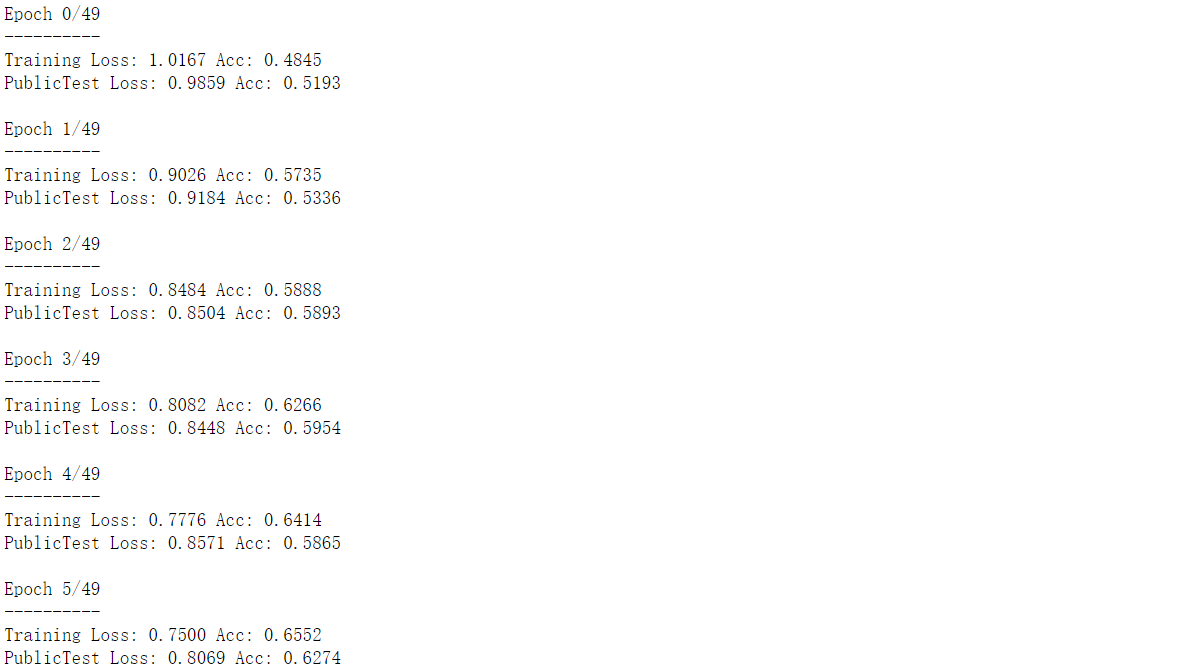
如果当前正有打开的视频，先将视频流关闭，然后重新打开视频流，thd就是上面定义的线程的一个实例，将cap视频流传入，级联分类器传入，还有自身对象传入以便进行性别分类和表情识别。最后开启线程进行检测。



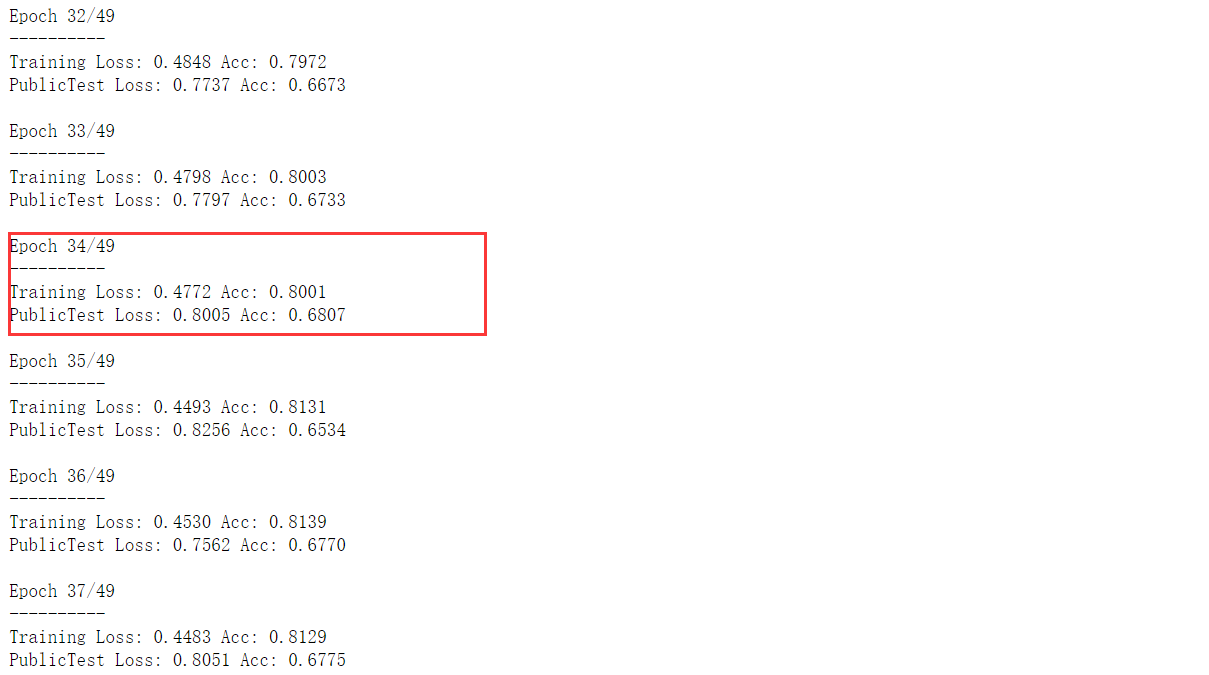
这是设置界面上检测结果的函数，将输出的图像送进来，然后把图像设置到一个QLabel上.

五．实验结果分析、对比和讨论

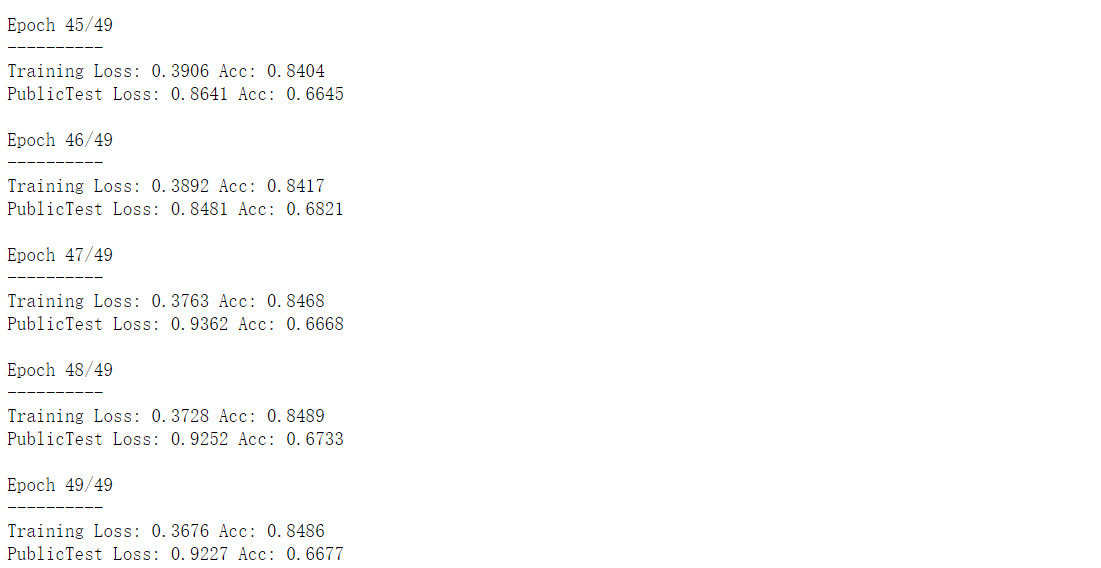
在对模型训练的时候，一开始使用的学习率是0.01，然后模型的精度一直不上升，一直没找到问题，后来将学习率调整为0.001的时候，一轮迭代精度就提升了不少，由于当时没有将结果记录下来，所以没有不收敛的截图，所以在此只展示收敛后的截图：



可以看到，从第0轮开始，模型的训练集精度和测试集精度一直上升的很快，训练集精度0.4845->0.5735->0.5888->0.6266->0.6414，测试集的精度0.5193->0.5336->0.5893->0.5954->0.5865->0.6274。一般训练的时候刚开始的损失下降的是很快的，从图中也可以看出，精度提升的很快，损失下降的也很快每一轮都下降了0.5左右。但是这样训练最后最高的精度是0.6807，最后是稳定在0.66左右：

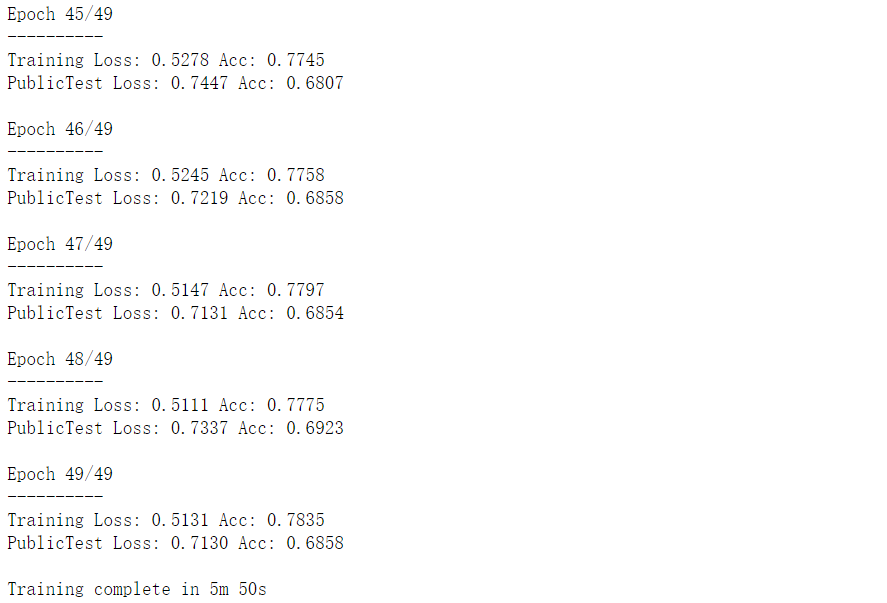


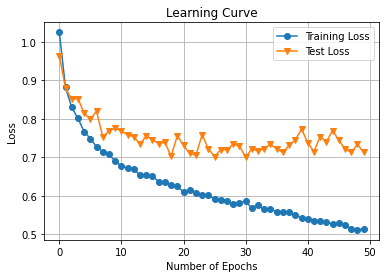
而且之后几轮精度一直在下降，loss一直在上升：



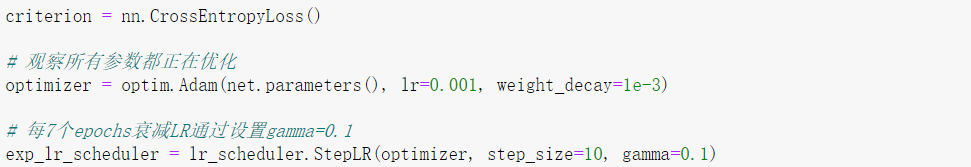
由此可知，数据集过拟合了，Loss重新回到了0.9227（这次没想到画损失图，只保留了训练代码）。

因此，继续修改模型，在之前的基础上，我又增加了一个dropout层，之前只在第二个fc层使用了dropout，现在我在两个fc层都使用了dropout，最终的结果如下：



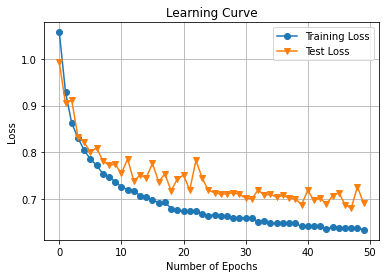


与之前相比，训练到最后没有训练集依然过拟合，但是测试集没有出现loss上升的情况，并且最终精度稳定在了0.68左右，只有一次上了0.69。按照之前课堂上讲的理论知识判断，通过增大正则项的系数可以缓解过拟合的问题，因此，又做了以下改进：



在optimizer中增加了weight\_decay，相当于增添了一个正则项。下面看看最后训练的结果：



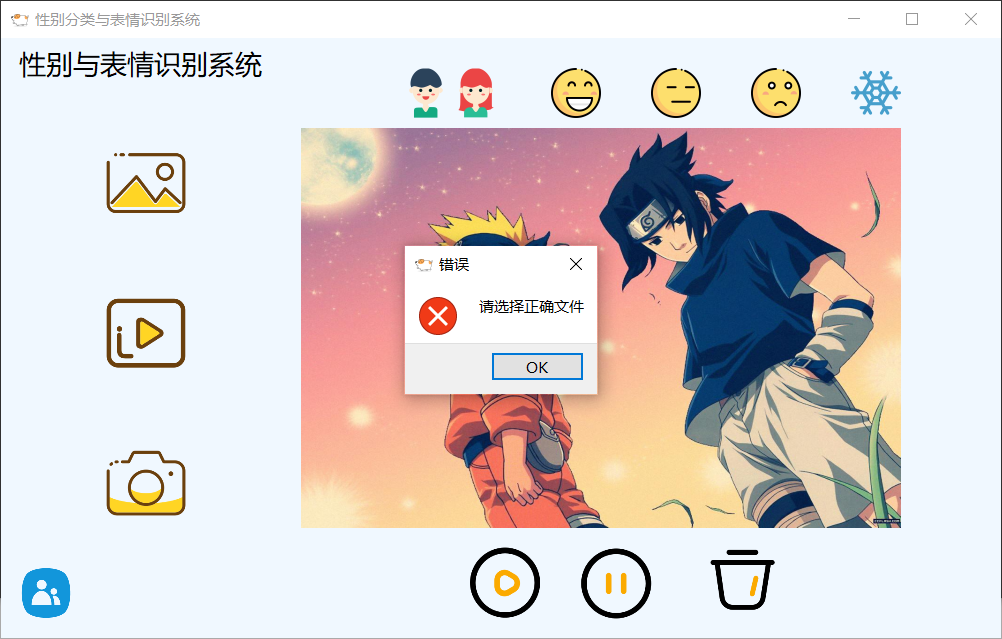


从训练结果中可以看到，最终测试集的精度稳定在了0.69左右，相比之前只提高了0.01，但还是有所提升。再看损失函数图像，相比于之前，训练的损失函数和测试的损失函数没有相差很大，所以可以判断没有产生过拟合问题，但是损失的值没有下降很多。

该网络的设计过程也花了很多时间，才找出能将图像进行两次最大池化的方案，所以总的来说体会挺深的。十一月多的时候刚学完Pytorch的基本使用，然后十二月就正好拿这个机器学习课设练练手，收获也不少。并且在两个星期内学习完了PyQt，刚学完之后就来做了这么个图形界面，巩固了一遍知识。

六．工程细节展示

文件选择错误：



**文件错误报错**

视频文件选择界面，只设置了mp4格式，就会自动屏蔽除了mp4拓展名之外的所有文件：



**视频选择**

运行环境与关于我的介绍：



**关于**

七．对本门课的感想、意见和建议

这门课是很好的实践课程，可能对于很多早就开始做深度学习的同学来说，是个那些题目都非常简单，但是对于我这种刚学完一些深度学习框架的同学来说，是个很好的练手的机会。并且为了这个课程设计，学习了PyQt图像界面设计，所以一门课或者说一个工程项目能让一个人学到很多的知识，比课堂上讲的内容多多了。课堂上只是一些理论内容，做项目时实践，只有理论基础打好才能很好的写出工程项目。这门课设的开设时间也很合理，在机器学习之后的学期开设，能留出一个学期时间来学习一些框架工具，为实践做好准备。