

模式识别与机器学习

Final Project : Face Recognition

2020011083 自 03 赵振禹*

2023 年 1 月 14 日

【摘要】本项目采用 torch1.4.0 环境，并且采取深度神经网络方案。项目采取 *face_recognition* 进行人脸和特征点检测，并且使用眼睛两点进行 alignment。神经网络的主干特征提取网络采用 mobilenet，整体的网络设计基本还原 Facenet 的网络设计。本人第 2 节将阐述整体思路，第 3 节将剖析代码结构，第 4、5 节将展现最终结果，以及对比超参数对于结果的影响，第 6 节进行总结。

1 任务描述

- **【目标】**建立一个无约束人脸图像识别系统，输入两张图片，输出判断二者是否为同一个人。
- **【要求】**可调用 *face_recognition* 等库进行特征识别，人脸转正等部分需要自行实现。
- **【训练集】** lfw 数据集，来自 5354 个人的 13101 张图像
- **【测试集】** 600 个 test pairs
- **【输出格式】** txt 文件，共 600 行，每行用 1/0 表示相同/不同

2 实现思路

整体过程分为 3 个部分：

- 人脸识别、对齐、伸缩等预处理
- 深度神经网络训练
- 测试集测试

我们接下来分别从这 3 个方面分别展开。

*zhaozhen20@mails.tsinghua.edu.cn

2.1 预处理

在本项目中，本人采取 *face_recognition* 进行人脸和特征点检测，并且使用眼睛两点进行 alignment。

先使用 *face_recognition* 识别出两只眼睛的特征点，而后取平均获得两只眼睛的中心 $(x_1, y_1), (x_2, y_2)$ 。则照片的倾角是：

$$\alpha = \frac{y_2 - y_1}{x_2 - x_1}$$



图 1：原图

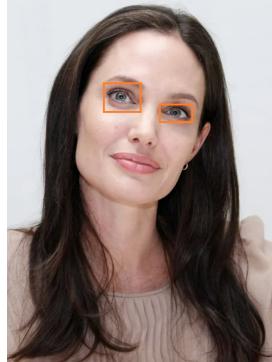


图 2：识别眼睛特征点（实际上
是特征点而非框）

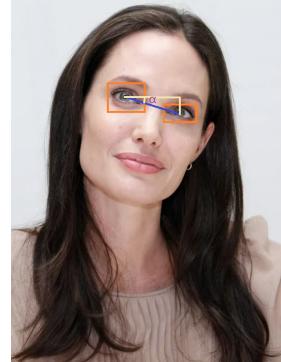


图 3：判断角度

我们旋转 α 后，再次使用 *face_recognition* 识别人脸并且进行裁剪。而后通过自己定义的 *img_processing* 函数进行无差的伸缩变换为 $(160, 160, 3)$ ，冗余的部分使用灰边补齐。（也可以采用直接伸缩变换，函数也支持）

2.2 深度神经网络

整体网络结构

我们网络的整体架构是这样的：

- 使用主干网络 backbone 进行初步特征向量提取：我们将在下一部分仔细说明
- 全局平局池化：将高维向量降维
- Dropout 层：防止过拟合
- 全连接网络层 Bottleneck：将维度转换为最终特征向量的维度
- L2 标准化/归一化：为距离的计算提供基础
- 全连接网络层 classifier：将维度转换为所有人数

代码上基本的结构如下，其中 x 可用于距离的计算， cls 可用于 label 的判别。实际上， cls 可用于单张图像的人脸判断，而 x 在比对多张图像时具有更好的效果。

```
1 x = self.backbone(x)
2 x = self.avg(x)
3 x = x.view(x.size(0), -1)
4 x = self.Dropout(x)
```

```
5     x = self.Bottleneck(x)
6     x = F.normalize(x, p=2, dim=1)
7     cls = self.classifier(x)
```

主干网络

主干网络是主要用于对图片进行特征提取。Facenet 中采取了 ResNetV1 进行特征提取。由于本人看到网络上有人使用轻量级深层神经网络 MobilenetV1 作为主干网络，故而选择其进行尝试。

为了减少参数，实现网络的轻量化，Mobilenet 采用了深度可分离卷积块作为网络构建的核心。可以这样理解，普通卷积在输入层和输出层之间采用全连接的方式，每个连接对应一个卷积核，即对应 a^2 个参数；深度可分离卷积先对输入层进行一次卷积，再在输入层和输出层之间采用全连接的方式，每个连接只对应一个参数。代码层面上可如下实现：(这一实现基本上借鉴成熟的代码)

```
1 def conv_dw(inp, oup, stride = 1):
2     return nn.Sequential(
3         nn.Conv2d(inp, inp, 3, stride, 1, groups=inp, bias=False),
4         nn.BatchNorm2d(inp),
5         nn.ReLU6(),
6
7         nn.Conv2d(inp, oup, 1, 1, 0, bias=False),
8         nn.BatchNorm2d(oup),
9         nn.ReLU6(),
10    )
```

以深度可分离卷积块为基础，Mobilenet 由这样的模块堆叠而成：

```
1 self.stage1 = nn.Sequential(
2     # 160,160,3 -> 80,80,32
3     conv_bn(3, 32, 2),
4     # 80,80,32 -> 80,80,64
5     conv_dw(32, 64, 1),
6
7     # 80,80,64 -> 40,40,128
8     conv_dw(64, 128, 2),
9     conv_dw(128, 128, 1),
10
11    # 40,40,128 -> 20,20,256
12    conv_dw(128, 256, 2),
13    conv_dw(256, 256, 1),
14 )
15 self.stage2 = nn.Sequential(
16     # 20,20,256 -> 10,10,512
17     conv_dw(256, 512, 2),
18     conv_dw(512, 512, 1),
19     conv_dw(512, 512, 1),
20     conv_dw(512, 512, 1),
21     conv_dw(512, 512, 1),
```

```

22     conv_dw(512, 512, 1),
23 )
24 self.stage3 = nn.Sequential(
25     # 10,10,512 -> 5,5,1024
26     conv_dw(512, 1024, 2),
27     conv_dw(1024, 1024, 1),
28 )

```

损失函数选取

我们的训练数据可以组建三元组 Triplet，这些 Triplet 包括主样本 anchor、正样本 positive 和负样本 negative 组成。我们最终希望达到的效果是 anchor 与 positive 的距离比与 negative 的距离要小一些，即：

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2$$

对于那些已经满足上式的三元组（简单样本），它们并不具有训练网络的意义。因此对于每一个 batch，我们需要在训练之前挑选出那些违反上式的样本（困难样本），这样就能够减少无意义的计算，更高效地训练网络。因此我们需要最小化的目标为：

$$\sum_{hard sample} \|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2$$

从理论上来讲，Triplet Loss 已经可以实现模型的训练了。但是参考网络上一些文章所实现的网络和自己的实践尝试，发现 Triplet Loss 一直比较小，仅仅使用 Triplet Loss 会使得整个网络难以收敛，因此结合 Cross-Entropy Loss 和 Triplet Loss 作为总体的 loss。

Triplet Loss 用于进行不同人的人脸特征向量欧几里得距离的扩张，同一个人的不同状态的人脸特征向量欧几里得距离的缩小。Cross-Entropy Loss 用于人脸分类，具体作用是辅助 Triplet Loss 收敛。

为了利用 Cross-Entropy Loss，我们需要在原本网络的基础之上构建分类器，即一个从某个中间向量到人脸类别维度向量的全连接层。这也是我们在整体代码结构中有两个全连接层的原因。

2.3 测试集测试

正如前文所言，可以使用两个图像的最终特征向量计算欧式距离，而后和阈值进行比较；也可以使用最后 classifier 全连接层输出的 5354 个神经元中最大的 label 进行比较而后判断。但明显前者在比较中更具有优势。故而我们在最终的 predict.py，即测试集测试中选择了距离作为判断依据。

3 代码结构说明

在这一节，我们将主要通过图像了解整体的代码逻辑与变量关系。我们在第 1 部分将说明整体文件结构，在第 2 部分将说明训练过程的 main.py 文件的过程以及变量关系。

3.1 整体文件结构

我们在一级目录下一共有如下文件/文件夹 (其中 *training_set* 和 *test_pair* 省略)，它们具体的功能定位也在图中说明：

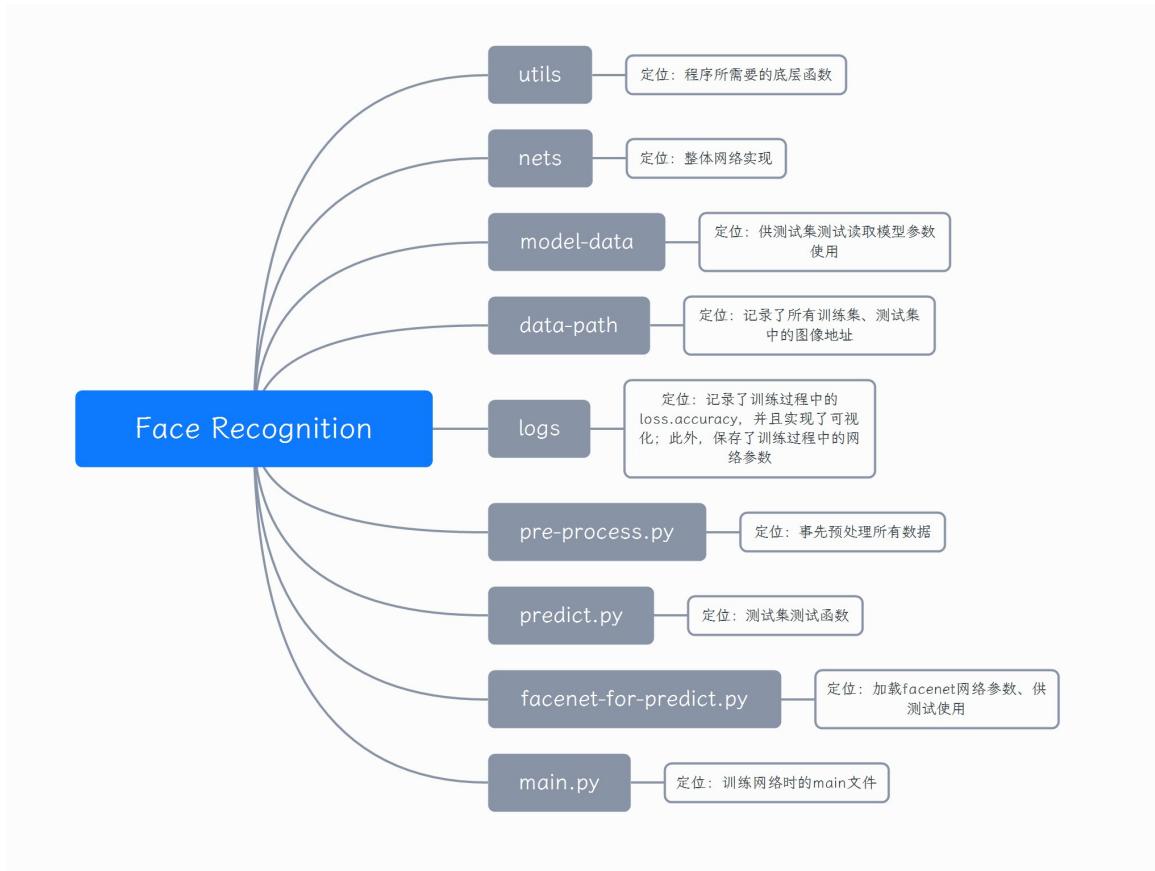


图 4: 一级文件结构

其中 utils 和 nets 是项目的核心部分，它们内部的具体内容如下：

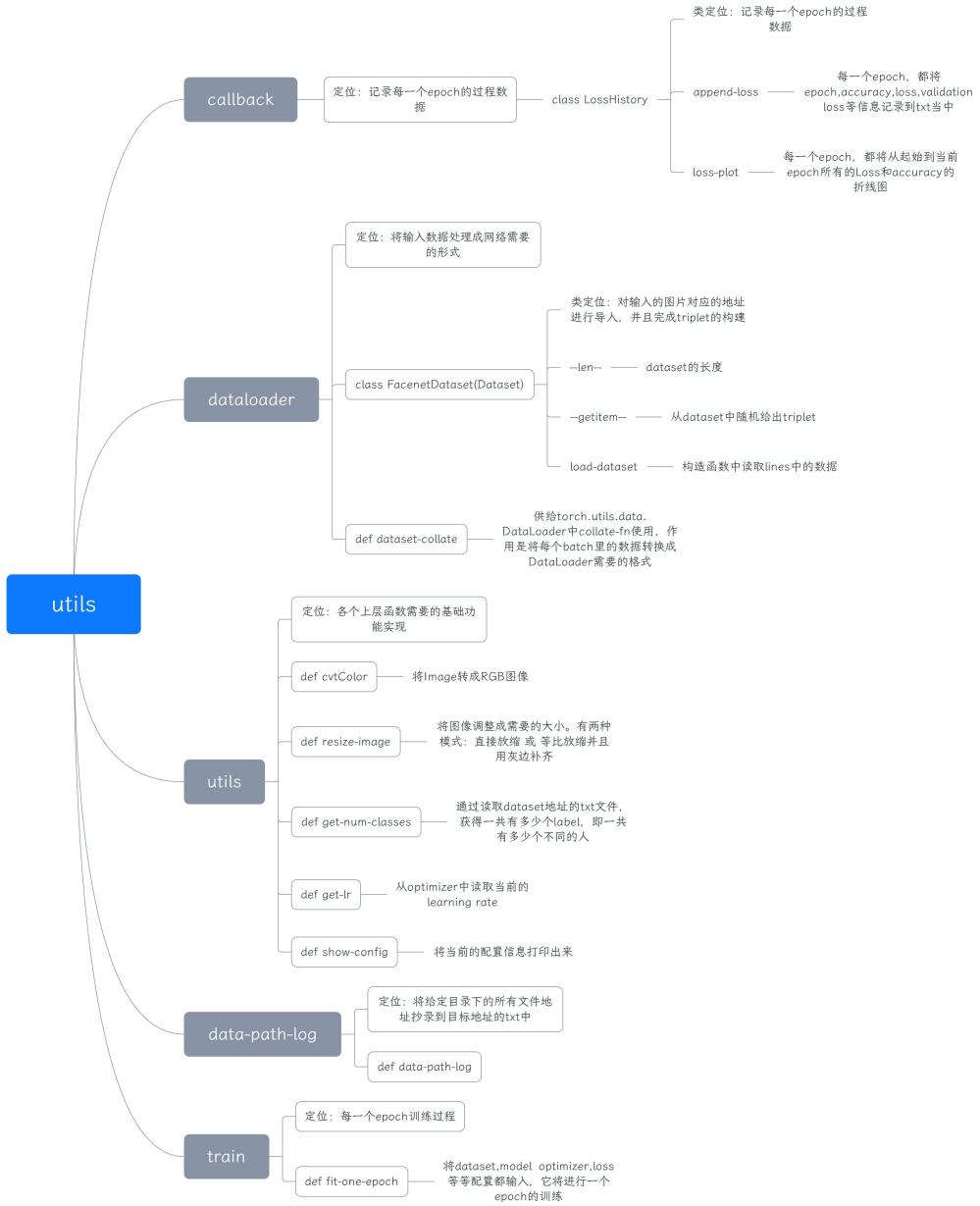


图 5: utils 代码结构

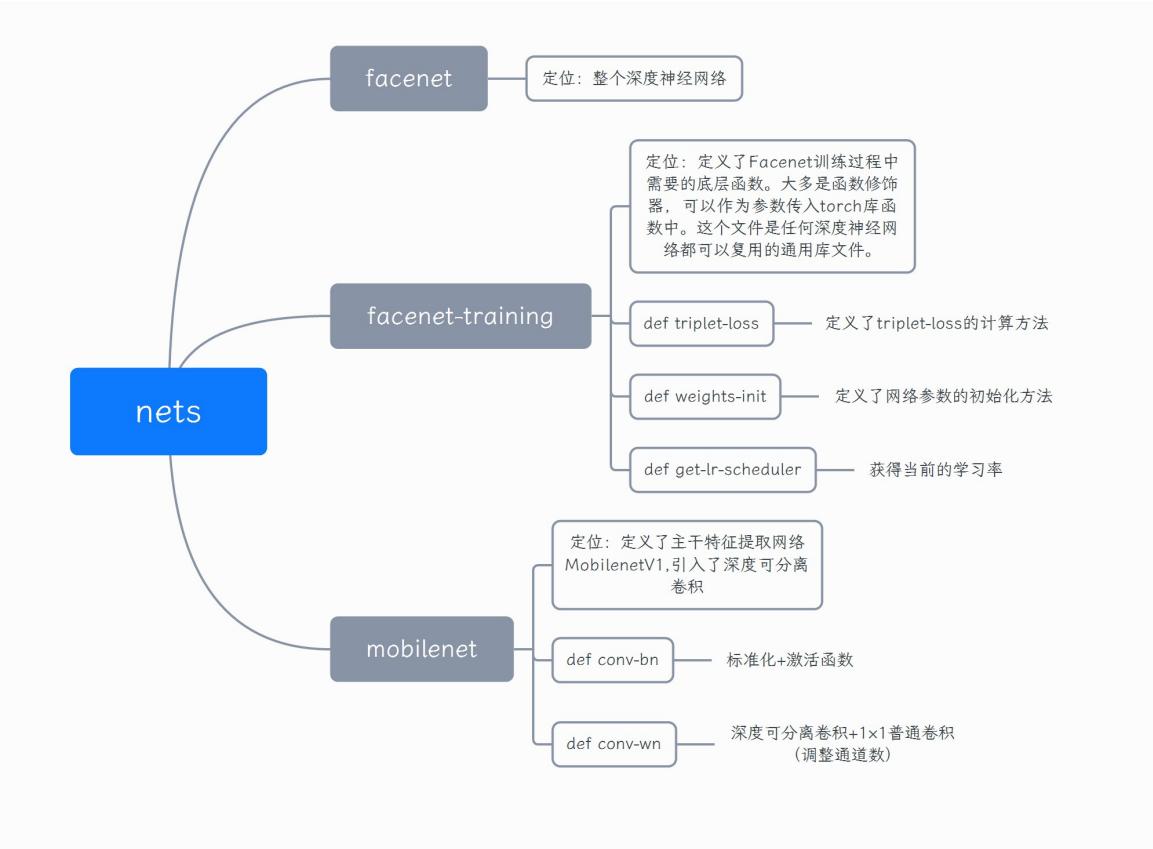


图 6: nets 代码结构

在清楚各个文件的具体定位于内容后，我们横向来看一下各个文件之间的交互：

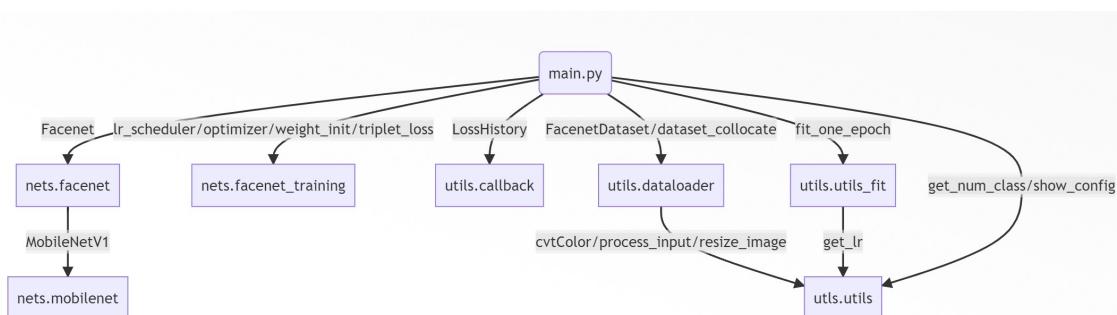


图 7: 文件交互关系

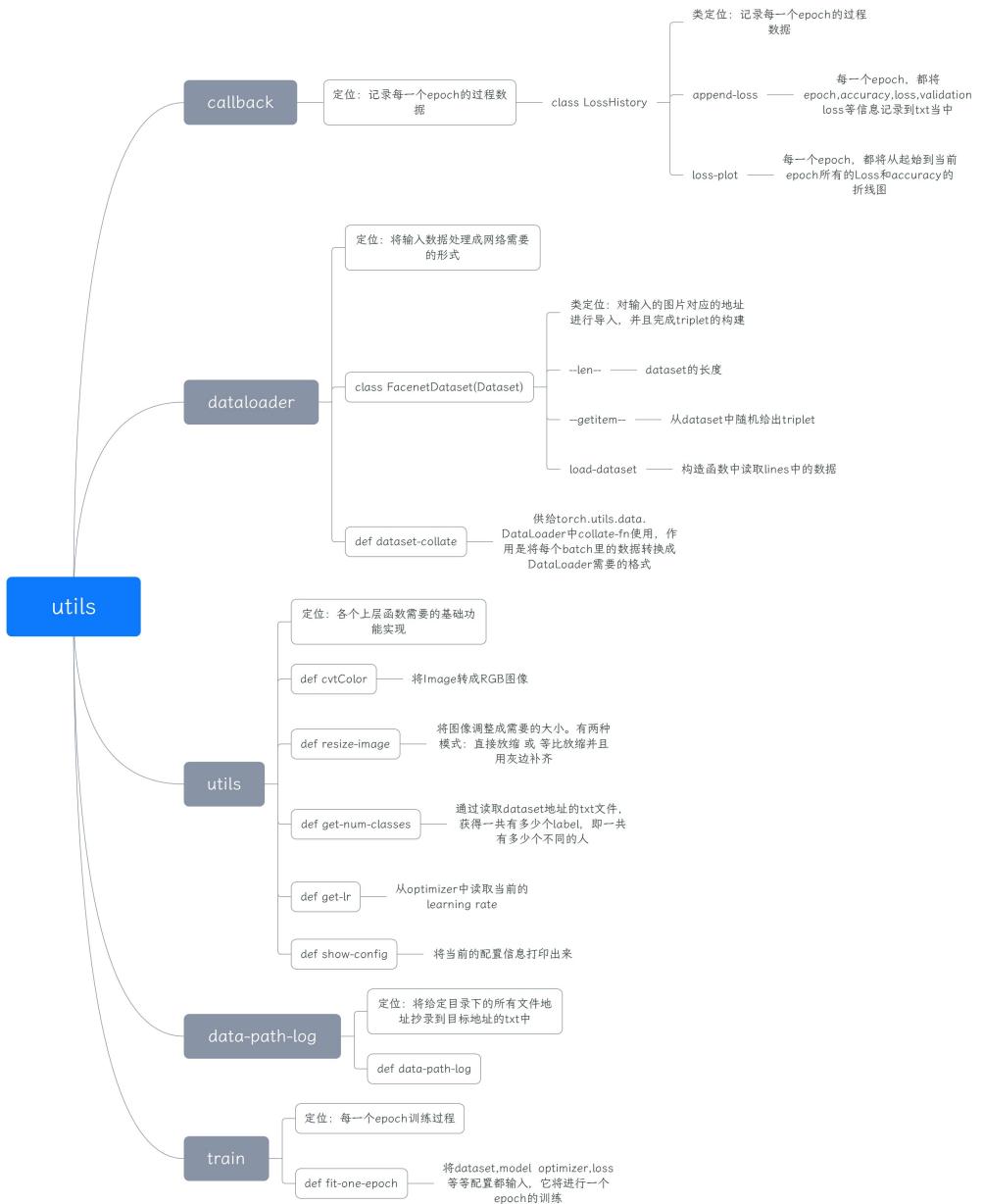


图 8: utils 代码结构

3.2 main 函数过程和变量关系

```

1 # Step 1: 设置各种硬件参数和网络参数
2 #   Step 1.1: 设置各种硬件参数
3 #   Step 1.2: 设置网络参数
4 # Step 2: 载入参数，并且处理成torch网络所需要的参数
5 #   Step 2.1: 载入模型model
6 #   Step 2.2: 设置记录函数logging
7 #   Step 2.3: 设置scaler和训练规则model_train
8 #   Step 2.4: 划分数据集生成lines，并且构造数据加载器train_dataset/val_dataset
9 #   Step 2.5: 处理其它所需要的参数：调整后的学习率Max_lr_fit/Min_lr_fit/优化器optimizer/

```

学习率下降公式 lr_scheduler_func / epoch_step

```
10 # Step 3: 生成 torch 模型所需要的数据加载器 DataLoader  
11 # Step 4: 正式开始训练
```

其中各个中间变量的关系如下，在阅读代码的时候可以对应：

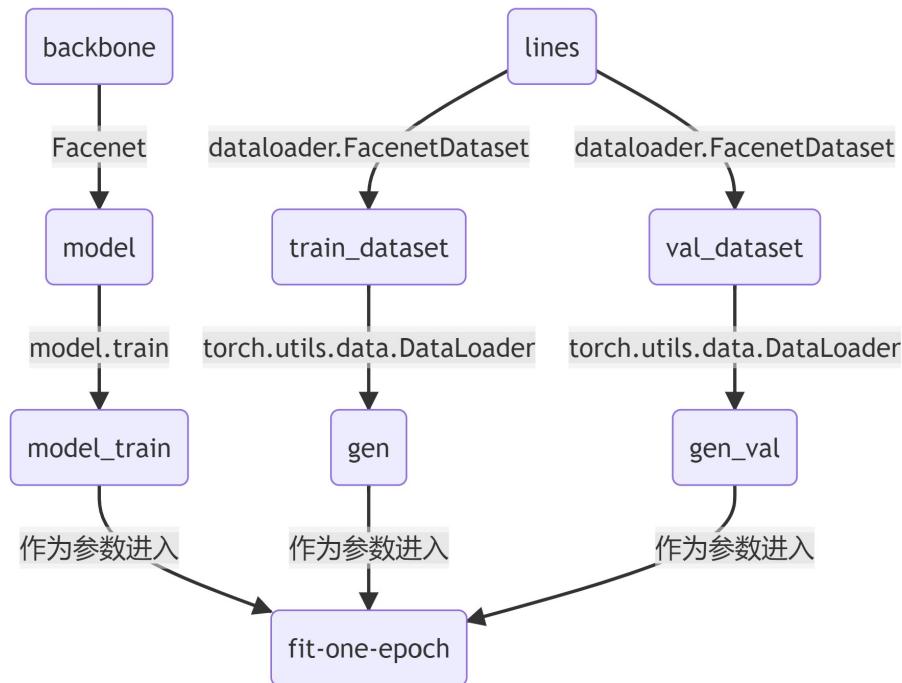


图 9: 变量关系

4 实现效果

我们在训练过程中的 accuracy 选取的是使用最后 classifier 全连接层输出的 5354 个神经元中最大的 label 进行比较而后判断正确率。

在少数情形，我们获得了高达 85% 的 accuracy，但更多情形下我们的 accuracy 在 40 – 50% 的区间。

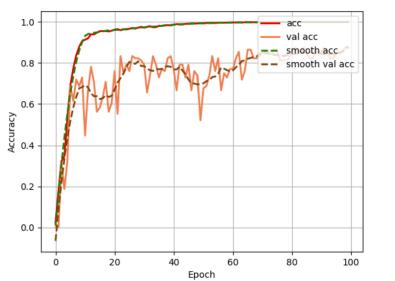


图 10: 特殊情形

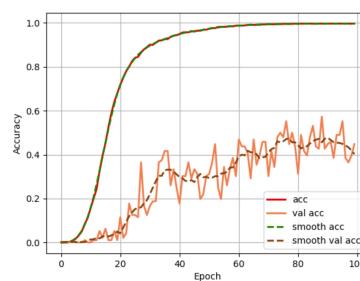


图 11: 大多情形

在测试集上若使用距离和阈值作为判断依据，在本人手动判断的 200 个样本中，正确率高达 95%。具体数据可见 `test_result_1.1.txt`。

5 超参数对比

正常而言，可能会影响网络效果的超参数有：学习率，Dropout 率。此处由于我们的 loss 是 triplet-loss 和 entropy-loss 的加和，它们之间的权值也是我们可以探讨的范畴。

学习率

我们分别选择初始 learning rate=1e-3, 5e-3, 1e-2 进行测试，结果如下，可以看到影响并不大，说明我们目前面临的问题在我们的模型下不太容易陷入到局部极值。

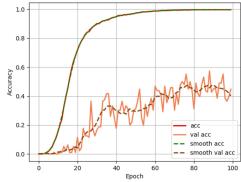


图 12: $init_lr = 0.001$

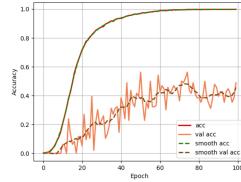


图 13: $init_lr = 0.005$

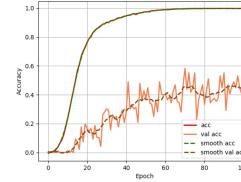


图 14: $init_lr = 0.01$

Dropout 率

我们分别选取 Dropout 率为 0.1, 0.3 和 0.5，得到结果如下。可以观察到 Dropout 的影响也不大。

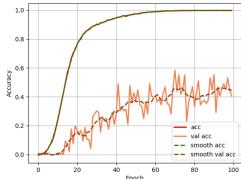


图 15: Dropout=0.1

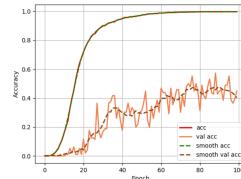


图 16: Dropout=0.3

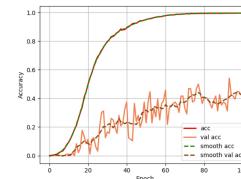


图 17: Dropout=0.5

triplet-loss 和 *entropy-loss* 的权值

我们分别取 triplet-loss: entropy-loss = 1:1, 5:1, 10:1 进行计算，得到如下结果。可以观察到效果还是有所提升，故而可以进一步探究其影响。

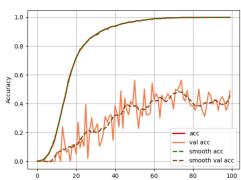


图 18: triplet: entropy = 1:1

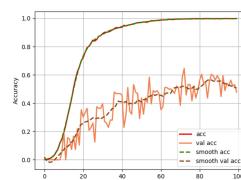


图 19: triplet: entropy= 5:1

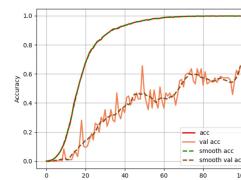


图 20: triplet: entropy = 10:1

6 总结与改进

由于本人之前鲜有搭建网络的经验，很多网络的设计以及实现对于网络上的代码依赖度不低。本文中阐释了整体的思路以及具体实现的思路与结构。整体运行效果出乎本人最开始的预期。

但由于确实精力有限与时间原因，还有很多地方的设计可以改进。比如本人没有充分利用实验数据，没有将单样本翻转新创建一个正样本对，而是直接将仅将数据作为负样本；同时，对于 *face_recognition* 无法识别的图像，代码选择直接标记为“不正确”，这一点也没有充分使用数据。

但是无论如何，通过完全从零开始搭建一个完整的人脸识别项目，从体验配环境的地狱折磨到最后调通了的欣喜若狂，整个过程不仅仅是带来满满的成就感，更是在实践的过程中温习了整个学期的学习内容。在此诚恳地向辛勤了一整个学期的鲁老师还有各位极其 nice 的助教学长们献上一份真诚的感谢！