第一章　数人数

输出图像中的人数，实质上是目标检测问题，目标类别仅“人脸”一个类别。一般目标检测网络进行目标检测的过程中，分两阶段进行：①提取潜在的候选框；②用分类器逐一筛选每个候选框。暴力遍历的存在使得检测实时性差，同时两阶段需要单独训练，难以优化。通过吴恩达机深度学习网课的介绍，以及资料的查阅，我了解到了 YOLO，YOLO 直接将整张图片作为网络输入，仅通过一次向前推断，即可在输出层同时得到边界框定位和相关的分类，从而速度更快；YOLO 直接分析全图，能更好地编码上下文信息，如物体与物体的关系，而不仅仅局限于物体的外观；YOLO 的泛化能力也很好，即便应用在与训练集分布不一致的情况下也能取得良好表现。综上，我选择 YOLO 作为我的人脸识别方法。

自 YOLOv1 被提出以来，不断发展如今已经到了 YOLOv5，纵观其发展历程，我最终选择 YOLOv3。因为 YOLOv1 检测小物体的能力较差，检测数量有限，而教室合照人脸一般小而密集；YOLOv2 在速度和精度方面都做了极大改善，YOLOv3 相较于 YOLOv2 做了进一步优化，介绍 YOLOv3 的论文实则是篇技术报告，读起来并不费劲；YOLOv4、YOLOv5 就过于高级，杀鸡焉用宰牛刀。权衡之下我选用 YOLOv3 去实现人脸识别。原理的介绍，也仅仅介绍 YOLOv3 相关的原理，不会涉及其从 YOLOv1 与 YOLOv2 的差别与不足。

1.1　YOLOv3 原理

1.1.1　网络输出参数与解码过程

将一幅图像分成 $S\times S$ 个网格(grid cell)，每个目标(object)会对应一个边界框(Bounding Box)，边界框的中心必然落在 $S\times S$ 个网格之一，那么则称包含该目标边界框中心的网格负责预测这个目标。每个网格预测 B 个边界框，每个边界框包含 $5+C$ 个特征：4 个位置参数和 1 个置信度参数和 C 个类别概率。由此可得，一幅图片最多可以检测出 $S\times S\times B$ 个目标（每个网格预测的每个边界框都检测出一个目标），最终的输出维度是 $S\times S\times B\times(5+C)$。

为方便说明问题，我们提前明确称呼的含义：网络预测的目标边界框为预测框，数据集提前标注的目标真实边界框为真实框（即 GT）。接下来引入先验框。

每个网格预测的边界框虽然中心位于该边框内，但是其长宽和大小未定，训练过程中完全处于野蛮生长的状态，这不利于神经网络进行学习。解决办法是，在每个网格中先预定好一组不同大小和宽高比的先验框，覆盖整个图像的不同位

置和多种尺度。先验框帮助我们定好了常见目标的宽和高，在进行预测的时候，我们就不直接预测边界框的坐标与长宽（b_x、b_y、b_w、b_h），而是预测边界框相对先验框的位置参数（t_x、t_y、t_w、t_h），这样可以使模型更加稳定。具体解码过程如图 1.1，其中(b_x,b_y)表示目标中心点坐标，(b_w,b_h)表示目标边界框的长宽，(c_x,c_y)表示该预测框所在网格左上角距离整张图片左上角相差格子数，(p_w,p_h)表示先验框的宽和高，σ是激活函数，论文中用的是 sigmoid 函数，将数值压缩到了区间[0,1]中。我们将网格的长宽归一化了，c_x和c_y都是从 0 开始的整数，那么预测框相对先验框的偏移量被限定在了[0,1]内，说明该预测框只能与先验框同属一个网格，不会跑到其他网格，也限制了其位置也不会野蛮生长。

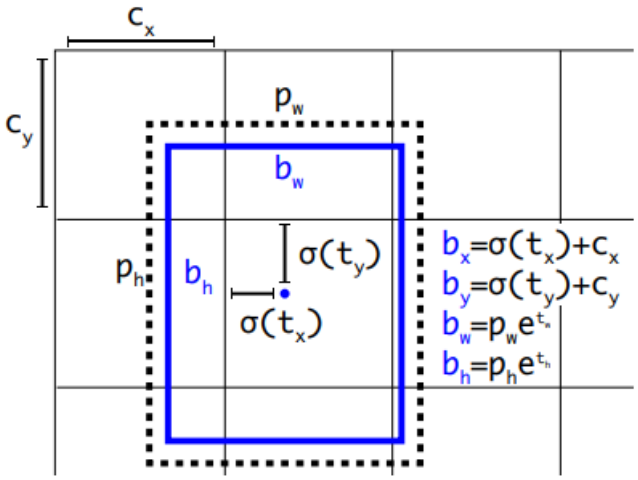

图 1.1 网络输出参数与边界框位置参数转换示意图

除了 4 个位置参数外，还有 1 个置信度和 C 个类别概率，置信度的含义有两重：①当前预测框内包含物体的概率$\Pr(Object)$，他只能说明框内是目标还是背景，不能用来说明框内目标属于哪一类。②表示当前框内有物体时，网络自己预测出的边界框与物体真实的边界框可能的交并比，即IOU_{pred}^{truth}。所以第 i 个网格中第 j 个预测框的置信度可以用公式表示为$C_i^j=\Pr(Object)*IOU_{pred}^{truth}$。实际网络输出的参数是$t_0$，置信度$C_i^j=\sigma(t_0)$。类别概率的含义是在该预测框包含物体的条件下，包含$Class_i(i=0,1...C)$的概率，即$\Pr(Class_i|Object)$。[1]

为了识别提升可识别物体的数量以及识别小物体的精度，YOLOv3 从三个不同的尺度进行预测，S 分别取 13，26 和 52，每个尺度设定三种尺寸的先验框，9 种先验框尺寸分别是(116,90)，(156,198)，(373,326)，(30,61)，(62,45)，(59,119)，(10,13)，(16,30)，(33,23)。比较大的物体则大概率归为 S=13 的特征层来检测，比较小的物体则大概率归为 S=52 的特征层来检测，这些先验框的尺寸是论文作

者基于 COCO 数据集，用 k-means 聚类的方法提取得到。

1.1.2 得分排序与非极大抑制筛选

如果仅仅依据上面的分析去编写代码，得到的结果如图 1.2。

图 1.2 原图（左）及其人脸识别结果（右）

网络识别出了一共 10647 张脸，标注每张脸都需要一个红色的边框，共 10647 个红色边框使得图像呈现出一片红。观察边界框参数，如图 1.3 所示，大部分置信度都很小，也就是说，大部分预测框其实内部是没有目标的。我们需要把那些预测框去除，故设置阈值 conf_thres，将得到的预测框置信度大于 conf_thres 的保留，其余的舍去。

```
face 3.585255e-06 1051 1026 1080 1155
face 9.240431e-07 1054 1044 1080 1194
face 6.6849447e-07 1053 1073 1080 1221
face 3.2316146e-07 1057 1094 1080 1257
face 2.629834e-07 1060 1117 1080 1292
face 3.8894012e-07 1061 1141 1079 1322
```

图 1.3 部分人脸识别结果，第 1 列是类别，第 2 列是置信度，3~6 列为边界框位置参数

上述操作一定程度上可以保证只保留存在目标的预测框，但是仍然存在问题，如图 1.4 所示。

图 1.4 原图（左）及置信度阈值筛选后的人脸识别结果（右）

该图共五个人，显然网络已经识别出了五张人脸的位置，但是每张人脸都标记了数个边界框。这是因为，为了保证识别准确率与不遗漏，每个网格设置了数个先验框，他们有可能检测同一个目标,他们无法通过置信度阈值筛选去除。可以发现，若两预测框预测的是同一个目标，则这两个预测框的 IoU 会比较大，此时只需要保留置信度较大的预测框，这种策略叫做非极大抑制，具体流程如下：

①选取这一类预测框中 score 最大的那一个，记为 max_detections，并保留它。score 为预测框得分，是综合了置信度和类别概率后的得分，公式如式(1.1)

$$score = \Pr(Object) * IOU_{pred}^{truth} * \Pr(Class_i \mid Object) = \Pr(Class_i) * IOU_{pred}^{truth} \quad \text{式(1.1)}$$

②计算 max_detections 与其余的预测框的 IoU

③如果 IoU>nms_thres（设置好的阈值），则意味着这两个预测框可能表示同一目标，那么就舍弃 score 较低的预测框。

④从最后剩余的预测框中，再找出最大 scores 的哪一个，跳转到①，循环往复。

最后可以得出概率最大的预测框，即最后显示出的预测框。

1.1.3 YOLOv3 的损失函数

YOLOv3 实现的是端到端的目标检测，仅仅一个损失函数便搞定分类和目标提取。损失函数具体表达式如式(1.2)。

$$Loss = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} [(b_x - \hat{b}_x)^2 + (b_y - \hat{b}_y)^2]$$

$$+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} I_{ij}^{obj} [(\sqrt{b_w} - \sqrt{\hat{b}_w})^2 + (\sqrt{b_h} - \sqrt{\hat{b}_h})^2]$$

$$-\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[\hat{C}_i^j \log(C_i^j)+(1-\hat{C}_i^j)\log(1-C_i^j)]$$

$$-\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{noobj}[\hat{C}_i^j \log(C_i^j)+(1-\hat{C}_i^j)\log(1-C_i^j)]$$

$$-\sum_{i=0}^{S^2} I_{ij}^{obj}\sum_{c\in classes}[\hat{P}_i^j \log(P_i^j)+(1-\hat{P}_i^j)\log(1-P_i^j)] \quad \text{式(1.2)}$$

我们来逐项解释损失函数。$\sum_{i=0}^{S^2}\sum_{j=0}^{B}$ 是遍历S*S个网格中的所有的预测框。I_{ij}^{obj} 表示第i个网格的第j个先验框是否负责该目标（正样本），如果负责 $I_{ij}^{obj}=1$，同理 I_{ij}^{noobj} 表示第 i 个网格的第 j 个先验框不负责该目标（负样本）。只有正样本对分类和定位产生贡献，负样本对置信度学习产生贡献。

$\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[(b_x-\hat{b}_x)^2+(b_y-\hat{b}_y)^2]$ 是框中心坐标误差，用的是平方和误差。

$\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[(\sqrt{b_w}-\sqrt{\hat{b}_w})^2+(\sqrt{b_h}-\sqrt{\hat{b}_h})^2]$ 是框的宽高误差，这里加了平方根的原因是为了对大框和小框公平，因为在相对误差相同的情况下，大框的绝对误差更大。

$\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}[\hat{C}_i^j \log(C_i^j)+(1-\hat{C}_i^j)\log(1-C_i^j)]$ 是有物体的框的置信度误差，

$\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{noobj}[\hat{C}_i^j \log(C_i^j)+(1-\hat{C}_i^j)\log(1-C_i^j)]$ 是没有物体的框的置信度误差，都是求交叉熵损失，其中没有物体的损失部分增加了权重系数，因为一般一幅图像大部分内容是不包含待检测物体的，所以要减少没有物体计算部分的贡献权重，以免网络倾向于预测单元格不含有物体。

$\sum_{i=0}^{S^2} I_{ij}^{obj}\sum_{c\in classes}[\hat{P}_i^j \log(P_i^j)+(1-\hat{P}_i^j)\log(1-P_i^j)]$ 是分类误差，求分类概率的交叉熵损失。

1.1.4 YOLOv3 的网络结构

YOLOv3 进行主干特征提取部分的网络名为 Darknet53，输入在经过 Darknet53 时，从 Darknet53 不同位置处产生三个特征层，他们的 shape 分别为 (52,52,256)、(26,26,512)、(13,13,1024)，接下来构建 FPN 特征金字塔，将这些不同 shape 的特征层进行特征融合，从而提取出更好的特征，具体操作如下：

(13,13,1024)的特征层进行 5 次卷积后进行后续操作获得预测结果；与此同时，(13,13,1024)的特征层将进行上采样后与(26,26,512)的特征层结合，得到(26,26,768)的结合特征层，对结合特征层进行 5 次卷积后进行后续操作得到预测结果；同理，我们还能得到(52,52,384)的结合特征层，进行 5 次卷积后进行后续操作得到预测结果。

上文特征层与预测结果之间还夹着一个“后续操作”，该后续操作即 1 次 3*3 卷积和 1 次 1*1 卷积，利用 3*3 卷积整合特征，利用 1*1 卷积调整通道数。最后的输出有三个，shape 分别为(52,52,75)、(26,26,75)、(13,13,75)，他们的大小不同，因为分别对应大、中、小不同尺度的待检测目标，通道数均为 75。之所以是 75，是因为 YOLOv3 的每一个特征层的每一个特征点存在三个先验框，每个先验框涵盖 4 个位置参数和 1 个置信度参数，图 xxx 基于 VOC 数据集，共 20 类，所以共 3*（5+20）=75 个通道数，如果是仅仅识别人脸，则类别数为 1，共 3*（5+1）=18 个通道数。

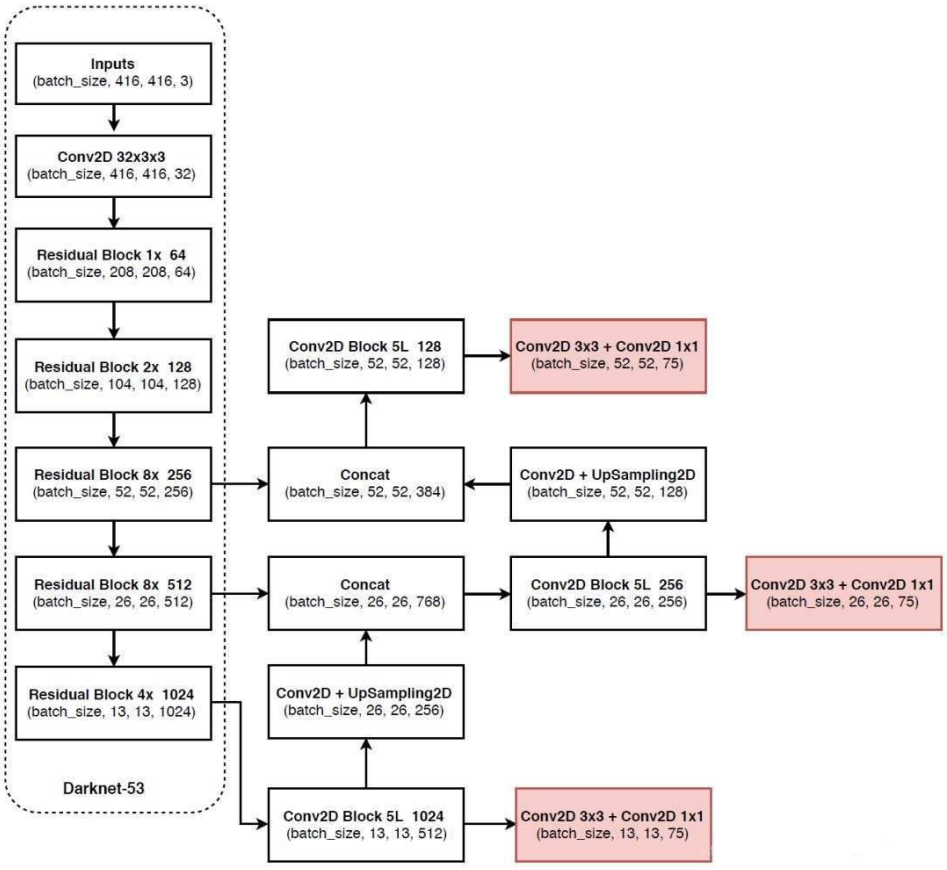

图 1.5 YOLOv3 网络结构示意图

众所周知，随着网络越来越深，训练变得越来越难，网络的优化变得越来越难。理论上，越深的网络，效果应该更好；但实际上，由于训练难度，过深的网络会产生退化问题，效果反而不如相对较浅的网络。而 Darknet53 共 53 层，为

了在加深网络的情况下又解决退化问题，Darknet53 引入残差块(Residual Block)，如图 XXX 所示。残差块的输出 $y=\sigma(F(x)+x)$，由恒等函数 x 和残差函数 $F(x)$ 组成。反向传播的时候，即使这层还没有开始学习，残差函数 $F(x)$ 的梯度接近零，由于恒等函数 x 的导数恒为 1，整个函数的梯度依然接近于 1，根据链式法则，先前的梯度依然可以反向传播，从而避免了“梯度消失”。残差结构可以不通过卷积，直接从前面一个特征层映射到后面的特征层，如果目标函数和恒等函数 x 相当接近，此时网络只需要被迫通过残差学习建模 $F(x)$，那么训练速度会大大加快。

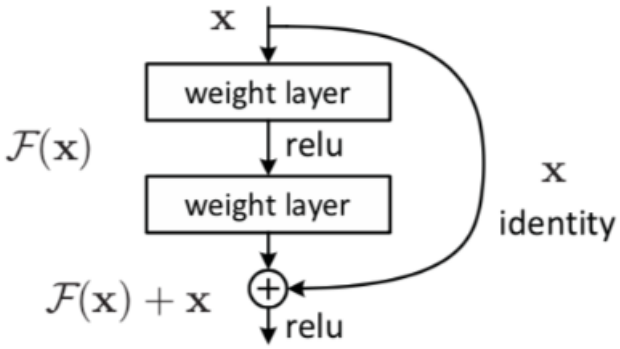

图 1.6 Residual Block 结构示意图

除残差结构外，网络还使用批量归一化(Batch Normalization)去优化网络结构，提高模型训练速度和泛化性能。批量归一化是指将网络每个层间输入的 mini-batch 进行归一化，在用一对可调参数对归一化输入进行缩放和平移，如式 xxx：

$$z_{norm}^{(i)}=\frac{z^{(i)}-\mu}{\sqrt{\sigma^2+\varepsilon}}$$
$$\tilde{z}^{(i)}=\gamma^{(i)}z_{norm}^{(i)}+\beta^{(i)} \qquad \text{式(1.3)}$$

其中 $\mu=\frac{1}{m}\sum_i z^{(i)}$，$\sigma=\frac{1}{m}\sum_i(z^{(i)}-\mu)$，$\gamma^{(i)}$ 和 $\beta^{(i)}$ 是一对可调参数，ε 是防止分母为零而设置的小量。我们知道，当网络的深度很大时，层间相关性增加，前面层参数的改变，逐层积累会导致参数传递到后面某一层时，后面那层的输入值与此次参数更新前的输入值的分布情况有很大的差距，网络要去不断的适应新的数据分布，进而使得训练变得异常困难。当归一化后，减少了内部层间输入分布的改变，增加了网络的独立性，使后面的神经元单元不过分依赖前面的神经元单元，有利于加速网络收敛和提高网络泛化能力。

YOLOv3 中使用 LeakyReLU 函数作为激活函数，LeakyReLU 函数是 ReLU

函数的一个变体，ReLu 函数在自变量小于 0 时恒等于零，对应到神经元上即梯度就变为 0，不会在训练中更新，神经元无“死亡”。而 LeakyReLU 的表达式如式 XXX

$$LeakyReLU(x) = x \bullet \max(x,0) + 0.1x \bullet \max(-x,0) \quad \text{式(1.4)}$$

LeakyReLU 函数在自变量小于 0 时为斜率很小的正比例函数，解决了 ReLU 函数存在的神经元“死亡”的问题。

1.2 实现步骤

1.2.1 数据集制作

原来的 YOLOv3 网络导入网上下载的预训练权重后，目标检测能力极佳，但是存在如下问题：已有的预训练权重对应的 VOC 数据集包含的 person 类，识别出来的是整个人体，而不是人脸。相较于人脸，人体的边界框多余的部分不仅会使得图像因为标注框太大而杂乱无比，多余的背景还无益我后面将要进行的人脸验证。因此我需要制作我的数据集，专门训练网络使之可以识别 face。

当我打开自己保存的图片，打算用 labelImg 自己制作数据集，我放弃了，首先我手上图片很少，其次图片人很多，面孔朝向各异且彼此交叠。作为一个能甩活给电脑就绝不自己付出苦力的人，我思考，有没有现成的人脸数据集给我使用呢？一查资料还真有，如 PubFig: Public Figures Face Database，Large-scale CelebFaces Attributes (CelebA) Dataset 和 Colorferet 等等，最后我选用了香港中文大学的 WIDER FACE: A Face Detection Benchmark，原因有二：①图片多，而且标注不复杂，单纯是非限制场景下框出了人脸的位置；②有描述上比它更好的数据集，但是我没能够成功地下载。

YOLOv3 使用的数据集是 VOC2007 格式，与 WIDER FACE 的格式存在差异，因此需要转换，只需要将 WIDER FACE 数据集下 wider_face_split 文件中，根据 wider_face_train_bbx_gt 和 wider_face_val_bbx_gt 中所提供的标注框中的信息，取填补对应图片的 xml 文件中<object>…<object>的内容，即可完成对应的 VOC 格式数据集的制作。具体转换过程可参考代码 wideface_to_voc.py。

VOC 格式数据集由三个文件组成，Annotations、ImageSets 和 JPEGImages，Annotations 中保存各个图片的 xml 文件，xml 文件中存有图片中真实框的信息；ImageSets 下的文件 train.txt 和 val.txt 分别存有用于训练的图片和用于验证的图片的名称，实现训练集与验证集的划分；JPEGImages 中存有原始 JPEG 图片。

之后通过程序 voc_annotation.py 即可综合 train.txt、val.txt 和 xml 的内容，生成新的 train.txt、val.txt 文件，里面存有每个图片的绝对路径和真实框信息，

方便后续将数据输入网络、损失计算等部分代码的书写。

1.2.2 训练过程

由于我没有充足的时间进行超参数调试，便把之前下载的权值文件作为预训练权值。训练分为两个阶段，分别是冻结阶段和解冻阶段。冻结阶段，batchsize=8，lr=1e-3，此时模型的主干被冻结了，特征提取网络不发生改变，此时仅对网络进行微调，可以避免训练初期权值被破坏，还可以加快训练速度。冻结阶段训练 50 个 epoch。Loss 曲线如图 1.7。

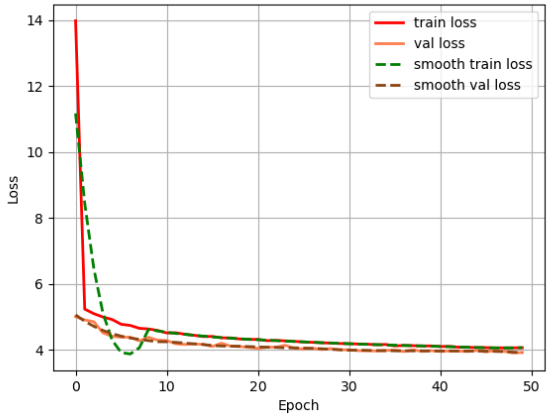

图 1.7 冻结阶段 Loss 曲线

解冻阶段，batchsize=4，lr=1e-4，此时模型的主干不被冻结了，特征提取网络会发生改变，占用的显存较大，网络所有的参数都会发生改变。线程数 num_workers = 4。但程序报错，显示显存不够，即便对代码修改以及时释放显存，也只能训练 1 个 epoch，便借用 26 教 C303 的服务器训练，Loss 曲线如图 1.8。

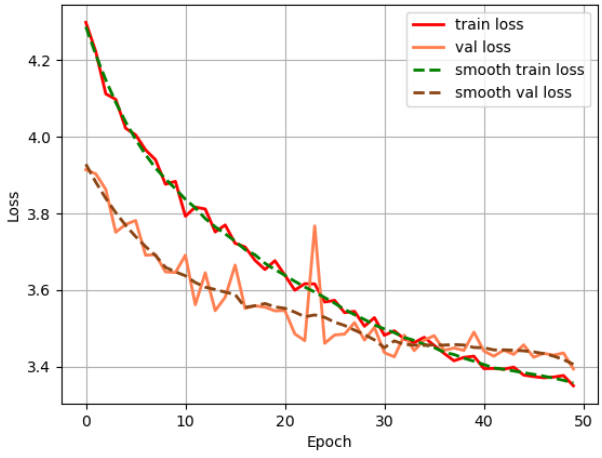

图 1.8 解冻阶段 Loss 曲线

进行解冻训练后的损失，与进行冻结训练相比，进一步降低，但是验证集损失出现大于训练集损失的趋势，我担心可能出现过拟合的情况，测试结果出现如图 1.9 的情况。

图 1.9 载入第 50 轮训练得到的权值文件（左）与载入第 100 轮训练得到的权值文件（右）的预测结果对比

载入第 50 轮训练得到的权值文件的网络和载入第 100 轮训练得到的权值文件的网络都识别出了正中间的大脸，右上角的小脸都未识别出来。但是右图多识别出了两张脸，位于图片中间，是位于绿色背景上边框的两个卡通图像的脸。这一方面训练的多图像的泛化能力更强了，但是，我不需要这么精细，如果有同学杯子上或衣服上有人脸卡通图案却也被识别为人脸并计入总数，是我不希望遇见的情况。故让我觉得选用第 50 轮得到的权值，即冻结训练最后一轮，不采用解冻训练得到的权值。

图 1.10 人脸识别结果展示

第二章 找自己

之前“数人数”的问题属于人脸识别，在之前识别的基础上找出自己，则属于人脸验证的问题。人脸验证问题比人脸识别更困难，因为要想要验证人脸，必须先识别出人脸。

起初我想沿用 YOLOv3 的结构，将人脸作为一类，秉承 YOLOv2 中提出的 Hierarchical classification 思想，将 face 类下设置专业同学数目一样多的子类，每个子类代表某一位同学的脸，即使用 multi-label 模型，进行训练，即最后输入我的照片，网络不仅知道照片中存在目标属于 face 类，还知道该目标属于 face_of_LiangQianxi 类。但这么做有一个难点，大三信息专业将近 100 人，我没有每个人的私人照，也不认识所有人，而且我担心不同人脸之间的特征差别未必明显。因此，如何从较小的数据中去学习，如何增添新的类别时不需要重新训练神经网络，是我需要克服的问题。

通过查阅资料我了解到 Siamese Neural Network 可满足要求。本章讲述如何通过 Siamese Neural Networks 实现“找自己”的功能，即借助自己的单人照，在集体照图像中找到自己。

2.1 Siamese Neural Networks 原理

Siamese Neural Networks(SNN)将输入进来的两张图片，利用同一个神经网络进行特征提取，然后比较这两个特征，最终输出一个长度为 1 的一维向量，其数值介于 0-1，用以表征输入的两张图片的相似度。如图 2.1 所示，Network1 与 Network2 网络结构完全相同，且权值共享，分别对 input1 和 input2 进行网络特征提取，这样输出得到的特征属于同一分布域，之后再通过 Loss 的计算，从而更加科学地评判 input1 和 input2 之间的相似性。

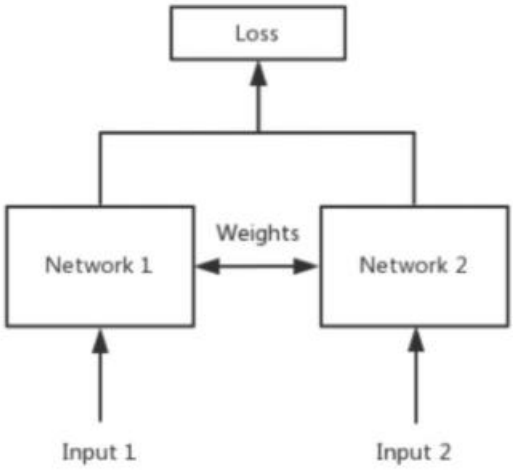

图 2.1 SNN 示意图

Siamese Neural Networks 的主干特征提取网络的功能是进行特征提取，各种神经网络都可以适用，我使用的神经网络是 VGG16。VGG16 的结构如图 2.2 所示。使用 VGG16 的原因如下：

①结构简单，卷积层均采用相同的卷积核参数，池化层均采用相同的池化核参数，代码写起来方便。虽然他的参数量仍然很大，但是相比于 LeNet-15 和 AlexNet 等特征提取网络，只能说小巫见大巫，我的电脑还是跑得动 VGG16 的。

②模型是由若干卷积层和池化层堆叠构成，容易形成较深的网络结构，卷积核较小，既可以保证感受视野，又能够减少卷积层的参数，特征提取能力强。

图 2.2 VGG16 结构示意图

通过 VGG16 网络分别获得 input1 和 input2 的多维特征，将多为特征全连接

展平为一维向量，对两个一维向量取 L1 范数，再将得到的范数全连接到一个神经元上，对该神经元取 sigmoid，使其值在 0-1 之间，代表两个输入图片的相似程度。

2.2 训练步骤

Siamese Neural Networks 有两个输入，当两个输入属于同一类型（同一人）时，此时标签为 1；当两个输入属于不同类型（不同人）时，此时标签为 0。然后将网络的输出结果和真实标签进行交叉熵运算，得到最终的 Loss。

训练集的文件结构如 2.3 所示，dateset 文件夹下 face_verification 中，有多个子文件夹，每个子文件夹存储同一张人脸，人脸截图是根据 YOLOv3 人脸识别结果自动切割得到的。程序会遍历所有子文件的图片，存储在列表中，然后打乱顺序，且标注对应的标签值，程序自动划分数据集和验证集，完成训练。

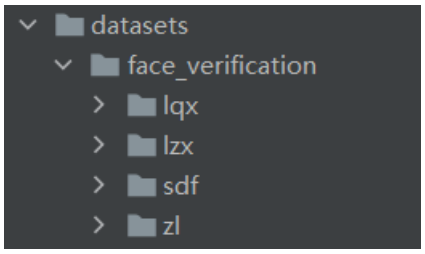

图 2.3 训练集的文件结构

为了防止过拟合，无论是 YOLOv3 还是 SNN，训练过程中都进行了数据扩增，对数据集图像进行了尺寸调整、翻转操作、旋转操作、色域扭曲等等。部分扩增结果如图 2.4 所示。

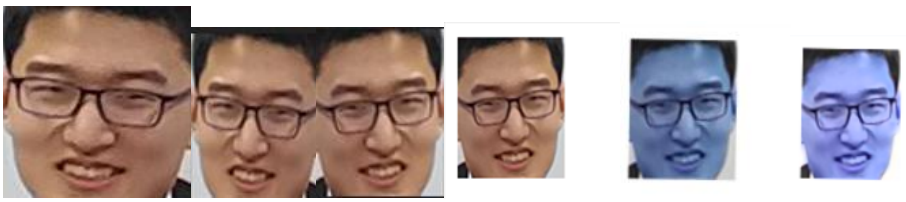

图 2.4 数据扩增示例

第三章 整合

3.1 整合思路

最后人脸识别和人脸验证的功能集合在 face_recognition_and_verification.py 中。YOLOv3 中的 detect_image()与 SNN 中的 compare_image()想结合，完成数人数和找自己。思路如下。

首先选取一张个人照片，作为 SNN 的 input1。然后将班级合照输入 YOLOv3 进行人脸识别，统计出总人数为 x，并截取 x 张人脸，以此将 x 张图片最为 SNN 的 input2，得到一个概率列表，里面存有 input1 分别与 x 张 input2 之间的相似度，x 张人脸中选取相似度最大的作为人脸验证结果，并利用 detect_image()返回的参数，在原图上标注个人处在的位置。

*这里记录一下写代码时候犯的错误，曾一度让我以为自己写的网络有问题，在此记录：

list[0,1,2,3,4,5,6,7,8,9,10]排序后是 list[0,1,2,3,4,5,6,7,8,9,10]。而 list[‘0.jpeg’, ‘1.jpeg’, ‘2.jpeg’, ‘3.jpeg’, ‘4.jpeg’, ‘5.jpeg’, ‘6.jpeg’, ‘7.jpeg’, ‘8.jpeg’, ‘9.jpeg’, ‘10.jpeg’] 排序后是 list[‘0.jpeg’, ‘1.jpeg’, ‘10.jpeg’, ‘2.jpeg’, ‘3.jpeg’, ‘4.jpeg’, ‘5.jpeg’, ‘6.jpeg’, ‘7.jpeg’, ‘8.jpeg’, ‘9.jpeg’]。导致人脸识别时，我对数组下标理解错误，显示的图片不是概率最大的图片。对应的修改代码如下。

```
# !!!这段代码一定要注意，查了很久才发现错误
# list_str 的排序是 1.jpeg 10.jpeg 2.jpeg……
#  需要转换成 1.jpeg   2.jpeg …… 10.jpeg
temp = []
for i in range(len(image2_list)):
      temp.append(image2_list[i][0:-5])
temp.sort(key=int)
image2_list = []
for j in range(len(temp)):
            image2_list.append(temp[j] + '.jpeg')
```

3.2 环境配置

电脑配置：GPU 为 GTX1650；Anaconda3; CUDA V10.0.130; cudnn V7.0.3; pytorch V 1.10.0，Python 3.6.13。编译工具：PyCharm 2021.1.1。

3.3 文件结构说明

相关文件存储在 face_regonition_and_verification 中，下面三个文件：文件夹 siamese_to_verification 存储人脸验证相关文件，文件夹 yolov3_to_recognition 存储人脸识别相关文件，文件 face_recognition_and_verification.py 即整合代码，直接运行即可实现人脸识别和人脸验证。

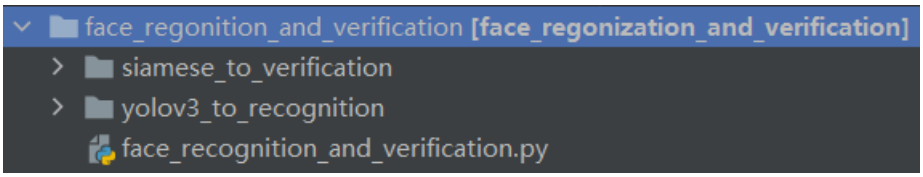

图 3.1 根目录文件结构

文件夹 yolov3_to_recognition 下，文件夹 img 用于存放预测时的输入图片，将图片移入 img 文件夹中，运行 Predict.py 后，直接输入图像名称即可预测。文件夹 img_out 用于保存预测结果。文件夹 logs 保存训练阶段产生的权值文件。model_data 存放相关信息，下存 my_class.txt 文件用于指示类别信息，simhei.ttf 是字体文件，yolo_anchors.txt 存放先验框的尺寸信息。文件夹 nets 和文件夹 utils 都是相关代码。VOC2007_face 存放数据集，上文已经阐述过了，由于数据集文件较大，提交作业时只拷贝部分图片与.xml 文件，仅作示意。Predict.py 用于人脸识别，train.py 用于训练网络。

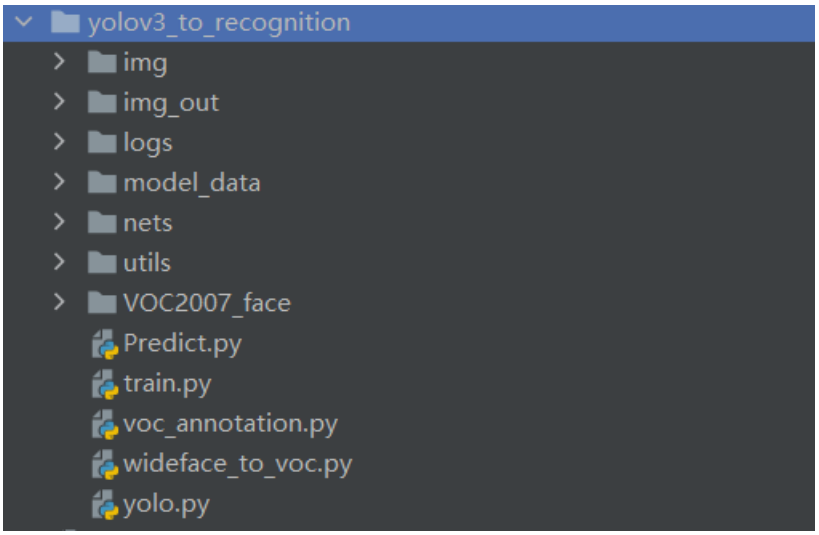

图 3.2 yolov3_to_recognition 文件结构

文件夹 yolov3_to_recognition 下文件夹功能与 yolov3_to_recognition 类似，数据集存在文件夹 datasets 中。

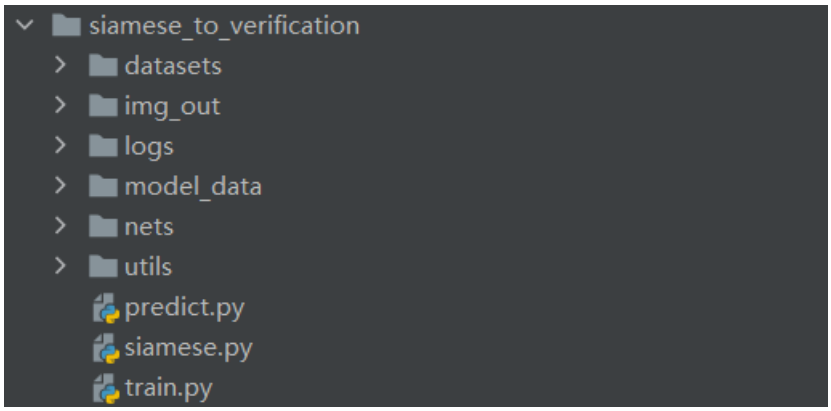

图 3.3 yolov3_to_recognition 文件结构

3.4 结果展示

图 3.4 示例图 1（上）数人数（下）找自己

图 3.5 示例图 2（上）数人数（下）找自己

图 3.6 示例图 3（上）数人数（下）找自己

3.5 反思提升

①如果有更多的时间，我将试着将 YOLOv3 每层网络的情况输出，观察其学期情况，思考能否根据具体场景删除一些不必要的卷积层，简化网络结构，提高运行速率。

②人脸识别的准确率低，可以通过如下方式尝试提升：

a.增加训练集数量。

b.修改损失函数为三元组的形式。

c. 我使用的权值文件仅仅是训练低 15 轮的，然后输入的两张合照正好都完成了识别。数据集不要只是 lqx 和 nonlqx 两类（之所以这么设计是因为时间紧迫，来不及细分了），nonlqx 应该根据不同人再细分类别。

参考文献

[1] Redmon, Joseph et al. “You Only Look Once: Unified, Real-Time Object Detection.”[J] IEEE, 2016 Conference on Computer Vision and Pattern Recognition (CVPR)(2016): 779-788.

[2] Redmon, Joseph and Ali Farhadi. “YOLO9000: Better, Faster, Stronger.” 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)(2017): 6517-6525.

[3] Redmon, Joseph and Ali Farhadi. “YOLOv3: An Incremental Improvement.”[J], arXiv preprint arXiv:1804.02767 , 2018.

[4] Ioffe, Sergey and Christian Szegedy. “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.” [J], arXiv preprint arXiv:1502.03167 (2015).

[5] He, Kaiming et al. “Deep Residual Learning for Image Recognition.”2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)(2016): 770-778.

[6] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov. “Siamese neural networks for one-shot image recognition.” ICML deep learning workshop, vol. 2. 2015. 2015.

[7] Simonyan, Karen and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition.” arXiv preprint arXiv:1409.1556 (2015).