图像处理 HW3：DCGAN 图像生成笔记

李天宇　　2200013188　　信息科学技术学院

1　生成对抗网络 GAN

GAN 中有一个生成器（G）和一个判别器（D），两者博弈对抗。D 最大化正确分类概率，G 最小化被抓包概率。损失函数为

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}\big[logD(x)\big] + \mathbb{E}_{z\sim p_z(z)}\big[log(1-D(G(z)))\big] \tag{1}$$

2　DCGAN

DCGAN 是 GAN 的一个具体实现，在判别器（D）、生成器（G）中分别使用了卷积层、卷积转置层。

$D(x) \to p$，其中 x 是 $3\times 64\times 64$ 的图像，$p\in(0,1)$，是输入来自真实数据分布的概率。

$G(z) \to x$，其中 z 是从标准正态分布中提取的潜在向量，输出 x 是 $3\times 64\times 64$ 的 RGB 图像。

2.1　环境配置

先用 conda 配好环境。只需要执行单元格就可以完成环境配置，甚至完成网络生成、训练，还可以自己向其中加代码调试，Jupyter 太厉害啦，只不过重启 vscode 之后似乎执行过的代码就都归零了。

2.2　网络初始化

`weights_init()`函数可以将网络初始化。

2.3　生成器

生成器 G 将 (z) 映射到 $3\times 64\times 64$ 的图像空间，通过一系列转置卷积层实现。

2.4　判别器

判别器 D 是一个二元分类网络，输出图像为真实的概率 p。注意到，G 中用的激活层是`ReLU`函数，而 D 中用的则是`LeakyRelu`。既然`LeakyRelu`很大程度上优于`ReLU`，又不会多计算量，为什么不在 G 中也采用`LeakyRelu`作为激活层呢?

2.5 训练

训练过程需要一个损失函数和一个优化器，代码中使用的`BCELoss`函数定义如下：

$$\ell(x, y) = L = \{l_1, \dots, l_N\}^\top, \quad l_n = -\left[y_n \cdot \log x_n + (1 - y_n) \cdot \log(1 - x_n)\right] \tag{2}$$

优化器则是`Adam`优化器，这个我在 ai 基础的 lab 中也用过，表现很亮眼的一个优化器。

一个好的生成器应该具备从纯噪声中生成数据的能力。因此，生成一批固定来自高斯分布的隐空间向量，也就是代码中的`fixed_noise`，它们用来可视化模型训练成果。

代码中，先对 D 网络进行训练，再用经过训练的 D 网络对 G 网络训练，其中，有一个优化是将最小化 $\log(1 - D(G(z)))$ 转为最大化 $\log(D(G(z))$，这是因为训练开始时，$D(G(z))$ 理论上接近 0，那么 $\log(1 - D(G(z)))$ 就接近 0，梯度可能过小。

3 改进

有如下几个改进这个生成器质量的方式：

在 D 中添加一个`Sigmoid`层，替换对抗损失。

增加训练轮数（大力出奇迹）。

4 集成

虽然不用写，我还是补充一句（因为开始没搞懂）：`netG(noise).detach().cpu()`输出的是张量（的数组），所以我用了`transforms.toPILImage`将其转为图片。

另外，训练了半天，输出的图像还是很抽象，总感觉还不如直接用本来给的 checkpoint。

只做了生成部分，训练代码就不往里放了，毕竟数据集太大。

5 图像编辑功能（选择特征）

我生成了 64 个图像，将其中性别特征明显的做了标注，按照作业中给出的步骤，求出了向量 v_A，并根据此重新生成了图像，可以看出这些图像中，男性特征确实更明显了。

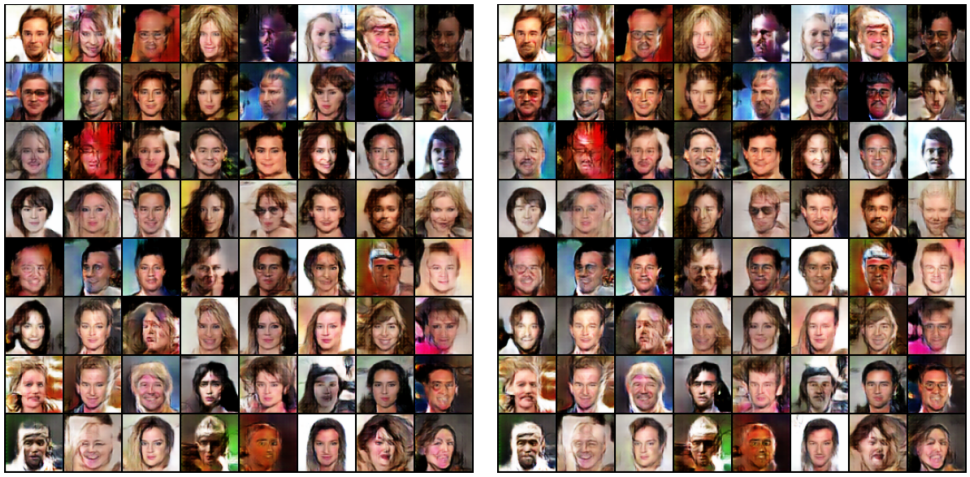

图 1: 添加 v_A 后的对比

用 torch 将该向量保存下来，就可以集成到网页中了。

6 后记

Jupyter 在我做这次作业时发挥了很大作用，唯一美中不足的就是很多插件对 Jupyter 的支持似乎不太好，比如代码提示、代码补全在这里都不是很好用。很多函数不会自动补全，都要到网上搜了才知道怎么写。

另外，不管是单个拿出来，还是集成到网页，G 输出的图像质量似乎都会变低，甚至颜色都不一样了，我没能找出原因。