计算机视觉作业 8

信息工程学院 张瀚文 2201212865

1 作业要求

在 GitHub 的/PyTorch-GAN 仓库中选择一个感兴趣的 GAN 程序，下载并运行，写出阅读总结，并对应代码标注出公式以及网络对应的代码。

2 原理分析

2.1 DCGAN 简介

DCGAN 是 GAN 的一个分支，DCGAN 分别在判别器和生成器中使用卷积和反卷积层来代替 GAN 中的多层感知机。它最初由 Radford 等人在论文 Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks 中提出。DCGAN 的判别器由卷积层、batch norm 层以及 Leaky RELU 激活函数组成，输入是 3*64*64 维的图片，输出是判断该图为真实图片的概率。DCGAN 的生成器则是由反卷积层、batch norm 层以及 RELU 函数组成。输入是一个来自正态分布的随机噪声，输出是 3*64*64 的 RGB 彩色图片。反卷积层的目的是，把随机噪声向量转换成维度与真实图像相同的向量。此外，在论文中，作者还介绍了关于设置优化器、计算损失函数以及初始化权重的一些技巧。

2.2 算法推导

DCGAN 的核心思想与 GAN 基本相同。生成器 G 的工作是生成看起来像训练图像的假图。判别器 D 的任务是判别一张图像是真实的训练图像还是来自生成器的伪图像。在训练过程中，生成器通过生成越来越像真实图像的伪图来尝试骗过判别器，而判别器则是努力地想成为更好的侦探，这样才能正确地对真实和伪造的图像进行分类。

D(G(z))是生成器 G 的输出为真实图像的概率值。正如 Goodfellow 的论文所描述的，D 和 G 在玩一个 minimax 的游戏，其中 D 尝试使它能正确分类真图和伪图的概率最大化 log D(x)，而 G 却尝试使 D 预测其输出是伪图的概率最小化 log (1−D(G(x)))。

与 GAN 相似，DCGAN 的损失函数可以表示为：

$$\min_{\mathrm{G}}\max_{\mathrm{D}}\mathrm{V}(\mathrm{D},\mathrm{G})=\mathbb{E}_{\mathrm{x}\sim\mathrm{p}_{\mathrm{data}}(\mathrm{x})}[\log\ \mathrm{D}(\mathrm{x})]+\mathbb{E}_{\mathrm{z}\sim\mathrm{p}_{\mathrm{z}}(\mathrm{z})}[\log\left(1-\mathrm{D}(\mathrm{G}(\mathrm{z}))\right)].$$

DCGAN 的训练方法是：

（1）训练判别器 D：固定 G，D 的损失由两部分组成，包括把真图判定为假和把假图判定为真的总损失。

（2）训练生成器 G：固定 D，G 的损失为真实图片和生成的假图之间的差异。

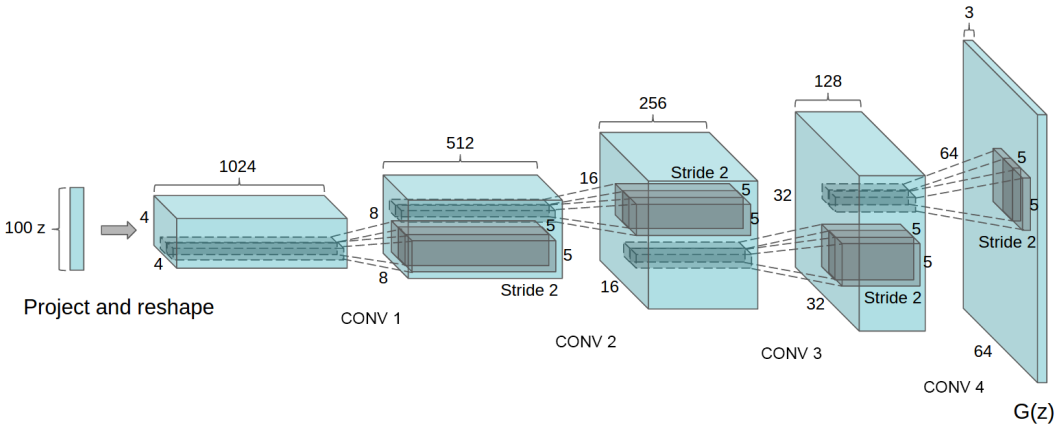
（3）重复上述步骤，直到 G 生成的分布与真实图片的分布非常接近。

2.3 生成器

生成器 G 用于将随机的空间向量映射成 RGB 图像(3*64*64)，实际上，这是通过一系列的二维反卷积层来完成的，每层都配带有批标准化层和 ReLU 激活。实际上通过反卷积操作并不能还原出卷积之前的图片，只能还原出卷积之前图片的尺寸。生成器的输出最终经过 tanh 函数处理，以使其返回到[-1, 1]的输入数据范围。

值得注意的是，在反卷积层之后存在批标准化 batch norm 函数，这是 DCGAN 论文中的关键贡献。这些层有助于训练过程中的梯度流动，DCGAN 论文中展示生成器结构的一

张图片如下。



2.4 判别器

判别器 D 是一个二分类网络，该网络将图像作为输入，并输出该图是真的概率。这里， D 以 3*64*64 的图像作为输入，通过一系列的 Conv2d，BatchNorm2d 和 LeakyReLU 层的处理，然后通过 Sigmoid 激活函数输出最终概率。

3 实现过程

3.1 训练

（1）生成 label 向量：

```
    # Adversarial ground truths label
      valid = Variable(Tensor(imgs.shape[0], 1).fill_(1.0), requires_grad=False)
      fake = Variable(Tensor(imgs.shape[0], 1).fill_(0.0), requires_grad=False)
```

（2）加载真实图片、生成假图并计算生成器 loss：

```
    # Loss measures generator's ability to fool the discriminator
      g_loss = adversarial_loss(discriminator(gen_imgs), valid)
```

（3）计算判别器 loss：

```
    # Measure discriminator's ability to classify real from generated samples
      real_loss = adversarial_loss(discriminator(real_imgs), valid)
      fake_loss = adversarial_loss(discriminator(gen_imgs.detach()), fake)
      d_loss = (real_loss + fake_loss) / 2
```

3.2 结果（在 colab 上训练）

[Epoch 41/200] [Batch 496/938] [D loss: 0.443259] [G loss: 0.795082]

[Epoch 41/200] [Batch 497/938] [D loss: 0.267247] [G loss: 1.534225]

[Epoch 41/200] [Batch 498/938] [D loss: 0.369109] [G loss: 1.347732]

[Epoch 41/200] [Batch 499/938] [D loss: 0.418104] [G loss: 0.828019]

[Epoch 41/200] [Batch 500/938] [D loss: 0.745071] [G loss: 1.446308]

[Epoch 41/200] [Batch 501/938] [D loss: 0.464996] [G loss: 1.021572]

[Epoch 41/200] [Batch 502/938] [D loss: 0.690608] [G loss: 1.056655]

[Epoch 41/200] [Batch 503/938] [D loss: 0.379056] [G loss: 2.159032]

[Epoch 41/200] [Batch 504/938] [D loss: 0.415908] [G loss: 1.079733]