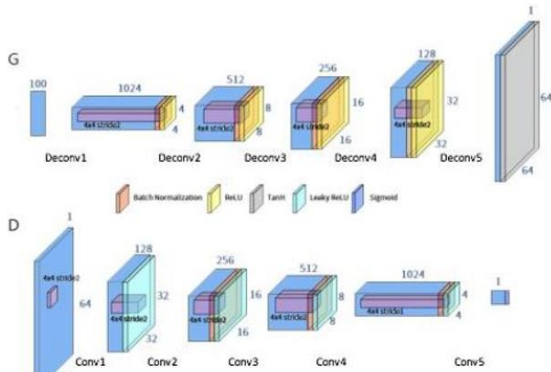


认知科学与类脑计算课程实验报告

实验题目：生成对抗网络的实现		学号：201600181073
日期：2019.6.6	班级：智能 16	姓名：唐超
实验目的： 在 MNIST 数据集上训练 GAN，生成“假的”手写数字。		
实验环境： Jupyter notebook & python3.7 & Pytorch		
主要算法： <p>生成式对抗网络（GAN, Generative Adversarial Networks）是一种深度学习模型，模型通过框架中两个模块：框架中同时训练两个模型：捕获数据分布的生成模型 G，和估计样本来自训练数据的概率的判别模型 D。</p> <p>D 的目标是尽正确估计样本是否来自真实数据集 data:</p> $D^* = \arg \max_D V(D, G)$ <p>其中目标函数（来自交叉熵损失）:</p> $V = E_{x \in P_{data}} [\log D(x)] + E_{x \in P_G} [\log(1 - D(x))]$ <p>G 的训练目的是将 D 判断正确的概率最小化:</p> $G^* = \arg \min_G \max_D V(G, D)$ <p>在训练 GAN 的过程中，固定住其中一模型，用反向传播算法更新另一模型，如此交替进行，构成一个动态的“博弈过程”，最终使得 G 完全模拟了真实数据的分布，D 难以判别。</p> <p>另外，对于生成手写数字这一任务，需用 CNN 作 GAN 中的 G 和 D，即 DCGAN，网络结构如下：</p>		
		

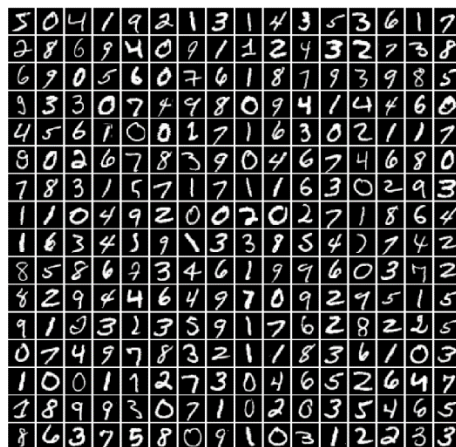
程序设计：

- 下载并处理 MNIST 数据集

采用 torchvision 下的 datasets 类直接导入 MNIST 数据集，用 torch.utils.data 下的 DataLoader 函数组织数据集，并作归一化处理，作为真实手写数字数据库（训练集）。

```
train_set = MNIST('./mnist', train=True, download=True, transform=preprocess_img)
train_data = DataLoader(train_set, batch_size=batch_size, sampler=ChunkSampler(NUM_TRAIN, 0))
```

可视化 train_data 中部分图像如下：



- 搭建 GAN，生成 MNIST 假数据

对于生成器 G，输入为随机的 noise=96 维噪声向量，经过上一栏 DCGAN 原理图中的一系列反卷积等操作后输出一个 28*28 维的假图像。

对于判别器 D，输入为 28*28 维的图像，经过上一栏 DCGAN 原理图中的一系列卷积等操作，得到该图像为真的概率。网络结构如下：

```
def discriminator():
    net = nn.Sequential(
        nn.Linear(784, 256),
        nn.LeakyReLU(0.2),
        nn.Linear(256, 256),
        nn.LeakyReLU(0.2),
        nn.Linear(256, 1)
    )
    return net
def generator(noise_dim=NOISE_DIM):
    net = nn.Sequential(
        nn.Linear(noise_dim, 1024),
        nn.ReLU(True),
        nn.Linear(1024, 1024),
        nn.ReLU(True),
        nn.Linear(1024, 784),
        nn.Tanh()
    )
    return net
```

在训练的每一个 epoch，使用 adam 作为优化器，G 生成 batch_size=256 个图像信息，对 D 输入 G 生成的图像信息的同时输入数据集中真实的图像信息，计算 D 的损失，固定 G 仅对 D 调整参数，计算 G 的损失，固定 D 仅对 G 调整参数。这样不断迭代，使 G 不断拟合真实数据集的信息分布。

```
def train(D_net, G_net, D_optimizer, G_optimizer, discriminator_loss,
          generator_loss, show_every=250, noise_size=96, num_epochs=210):
    iter_count = 0
    for epoch in range(num_epochs):
        for x, _ in train_data:
            bs = x.shape[0]
            # 判别网络
            real_data = Variable(x.view(bs, -1).cuda()) # 真实数据
            logits_real = D_net(real_data) # 判别网络得分
            # -1 ~ 1 的均匀分布
            sample_noise = (torch.rand(bs, noise_size) - 0.5) / 0.5
            g_fake_seed = Variable(sample_noise).cuda()
            fake_images = G_net(g_fake_seed) # 生成的假的数据
            logits_fake = D_net(fake_images) # 判别网络得分
            # 判别器的 loss
            d_total_error = discriminator_loss(logits_real, logits_fake)
            D_optimizer.zero_grad()
            d_total_error.backward()
            D_optimizer.step() # 优化判别网络

            # 生成网络
            g_fake_seed = Variable(sample_noise).cuda()
            fake_images = G_net(g_fake_seed) # 生成的假的数据

            gen_logits_fake = D_net(fake_images)
            g_error = generator_loss(gen_logits_fake) # 生成网络的 loss
            G_optimizer.zero_grad()
            g_error.backward()
            G_optimizer.step() # 优化生成网络

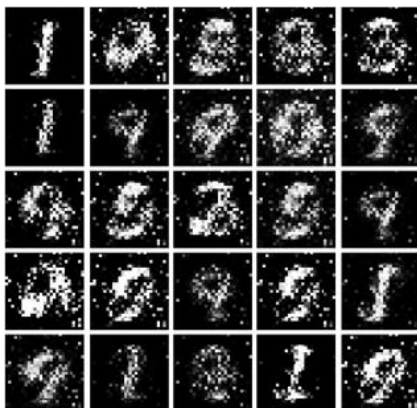
        if (iter_count % show_every == 0):
            print(epoch, iter_count)
            imgs_numpy = deprocess_img(fake_images.data.cpu().numpy())
            show_images(imgs_numpy[0:25])
            plt.show()
            print()
            iter_count += 1
```

调试分析：

在导入 MNIST 数据集时，解压下载后的数据集时遇到了错误 “IOError: CRC check failed”，查阅发现是程序利用系统自带的解压方式出错，改为手动解压缩，解决了这一问题。

测试结果及分析：

共训练 210 个 epoch，部分结果如下：



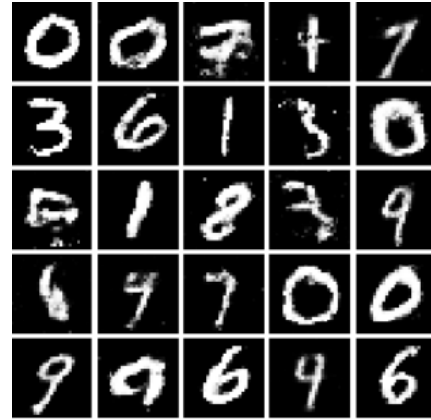
10th epoch



30th epoch



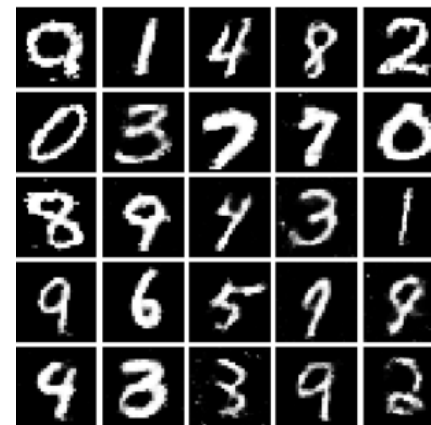
50th epoch



100th epoch



160th epoch



210th epoch

可以明显看出在训练过程中噪声越来越少，最终生成图像与 MNIST 中的手写数字非常相似。