计算机科学与技术学院神经网络与深度学习课程实验报告

实验题目： GAN		学号：201600181073
日期：2019.5.17	班级： 智能 16	姓名： 唐超

实验目的：

- 学习使用 pandas 读取数据；
- 学习使用 torchvision 处理数据；
- 练习并使用 pytorch 搭建 GAN.

实验软件和硬件环境：

Jupyter notebook&python3.6

实验原理和方法：

- **GAN**

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。

D 的目标是尽正确估计样本是否来自真实数据集 data：

$$D^* = \arg\max_D V(D,G)$$

其中目标函数（来自交叉熵损失）：

$$V = E_{x\in P_{data}}\left[\log D(x)\right] + E_{x\in P_G}\left[\log(1-D(x))\right]$$

G 的训练程序是将 D 判断正确的概率最小化：

$$G^* = \arg\min_G \max_D V(G,D)$$

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。

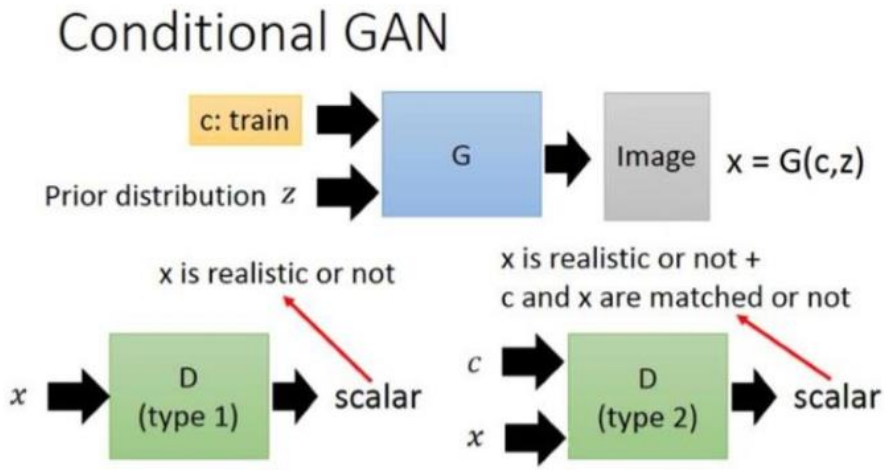
- **cGAN**

conditional GAN 是最早的 GAN 的变种之一，把原生 GAN 中的概率全改成条件

概率:

$$\max D \{\mathbb{E} \sim PdatalogD(G(x|y)) + \mathbb{E}x{\sim}PG \log(1 - (D(x|y)))\}$$

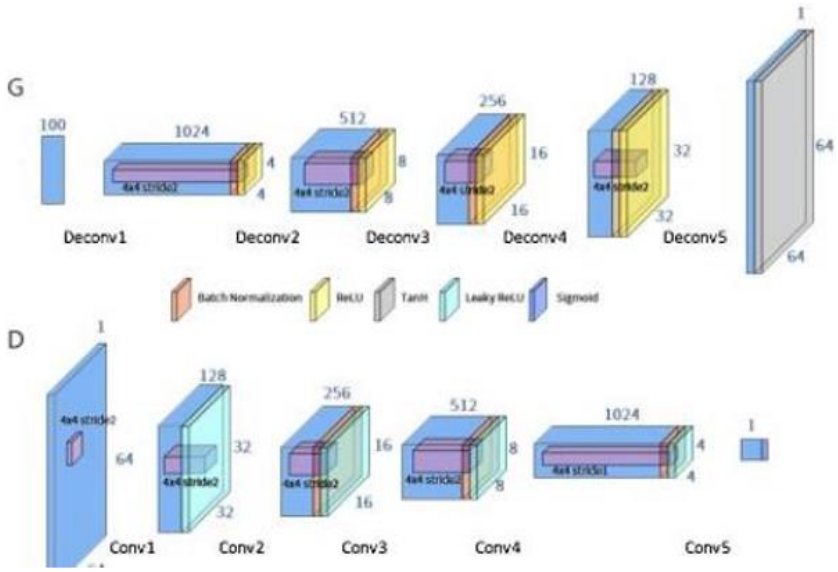
这个条件可以是图片，标注等，结构如下图：



真正在训练的时候，只需将条件直接拼接到原始数据。

- **DCGAN**

用 CNN 作为原始 GAN 中的 G 和 D。结构如下：



实验步骤：（不要求罗列完整源代码）

- **读取数据**

这里使用 df = pandas.read_csv('fashionmnist/fashion-mnist_train.csv'), 该函数有四十多个参数，这里仅需 csv 文件路径即可，返回一个 pandas.core.fram

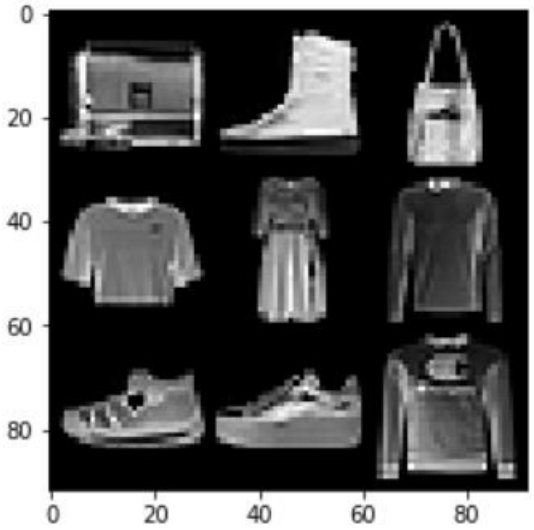
e.DataFrame 类型的数据，通过 df.iloc[:,1:].values， df.label.values 即可访问图像和相应的 label，数据类型是 numpy 数组。

- **处理数据**

用 torchvision.transforms.Compose 函数将图像变化操作组合在一起，如将图片数据 resize、转化为张量、标准化处理等。用 torch.utils.data.DataLoader 将图像数据组织成 batch，由于其第一个参数 dataset 是 torch.utils.data.Dataset 类的对象，因此需要写一个继承自 torch.utils.data.Dataset 类的自定义类对读取的 fashion-mnist 进行处理。详细代码如下：

```
class FashionMNIST(Dataset):
    def __init__(self, transform=None):
        self.transform = transform
        fashion_df = pd.read_csv('fashionmnist/fashion-mnist_train.csv')
        self.labels = fashion_df.label.values
        self.images = fashion_df.iloc[:, 1:].values.astype('uint8').reshape(-1, 28, 28)

    def __len__(self):
        return len(self.images)

    def __getitem__(self, idx):
        label = self.labels[idx]
        img = self.images[idx]
        img = Image.fromarray(self.images[idx])

        if self.transform:
            img = self.transform(img)

        return img[0].resize(1, 28, 28), label

transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Lambda(lambda x: x.repeat(3,1,1)),
        transforms.Normalize(mean=(0.5, 0.5, 0.5), std=(0.5, 0.5, 0.5))
])
dataset = FashionMNIST(transform=transform)
dataloader = torch.utils.data.DataLoader(dataset, batch_size=batch_size, shuffle=True)
```

另外，torchvision.utils.make_grid 函数可以将多幅图像拼成一副图像，方便查看。如：



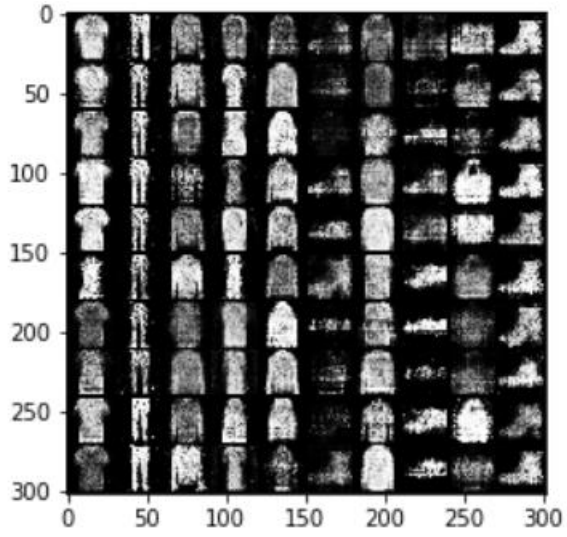
- **搭建 GAN**

图像的 label 应转换为 one hot 向量才能与图像信息拼接输入网络。

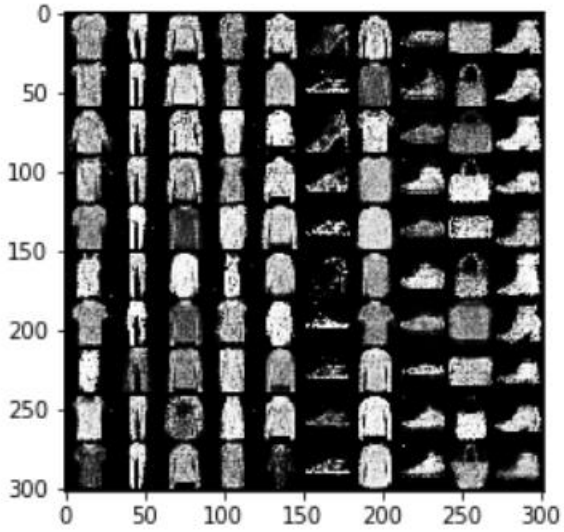
对于生成器 G，输入为随机的 100 维噪声向量加 10 维的 label 向量，经过上一栏 DCGAN 原理图中的一系列反卷积等操作后输出一个 28*28 维的假图像。

对于判别器 D，输入为 28*28 维的图像加 10 维的图像 label 向量，经过上一栏 DCGAN 原理图中的一系列卷积等操作，最后通过 sigmoid 函数得到该信息是真的的概率，即图像是真的且与 label 匹配的概率。

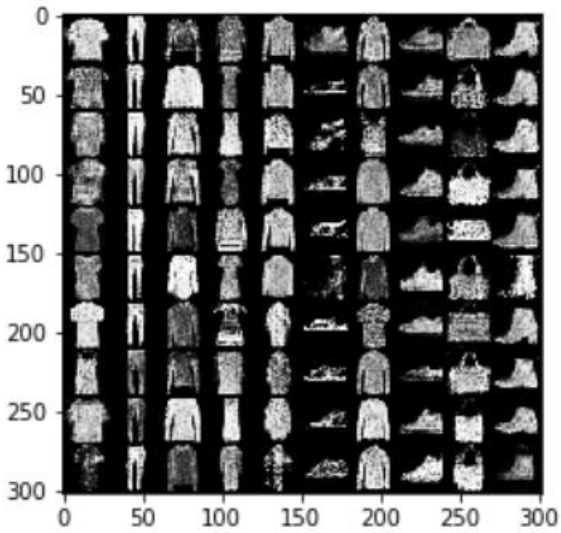
在训练的每一个 epoch，G 生成 batch_size 个图像信息，对 D 输入 G 生成的图像信息的同时输入数据集中真实的图像信息，计算 D 的损失，固定 G 仅对 D 调整参数，计算 G 的损失，固定 D 仅对 G 调整参数。生成图像如下：



5th epoch



50th epoch



100th epoch

结论分析与体会：

最终生成图像与 fashionminst 数据集中的图像较为相似，通过这次实验，学习了用 Gan 生成某一类型图像的完整流程，加深了对 Gan 的认识。

就实验过程中遇到和出现的问题，你是如何解决和处理的，自拟 1－3 道问答题：

在组织数据时，由于 torch.utils.data.DataLoader 的 dataset 参数只接收 torch.utils.data.Dataset 类的对象，不知道应该对 pandas 读取的数据作何操作才能使用这一函数，最后写了一个继承自 torch.utils.data.Dataset 类的自定义类对读取的 fashion-mnist 进行处理。