生成对抗网络实验报告

朱浩泽 1911530

June 18, 2022

1 网络结构

```
Discriminator(
  (fc1): Linear(in_features=784, out_features=128, bias=True)
  (nonlin1): LeakyReLU(negative_slope=0.2)
  (fc2): Linear(in_features=128, out_features=1, bias=True)
)

Generator(
 (fc1): Linear(in_features=100, out_features=128, bias=True)
 (nonlin1): LeakyReLU(negative_slope=0.2)
 (fc2): Linear(in_features=128, out_features=784, bias=True)
)
```

2 实验结果

2.1 训练损失函数曲线

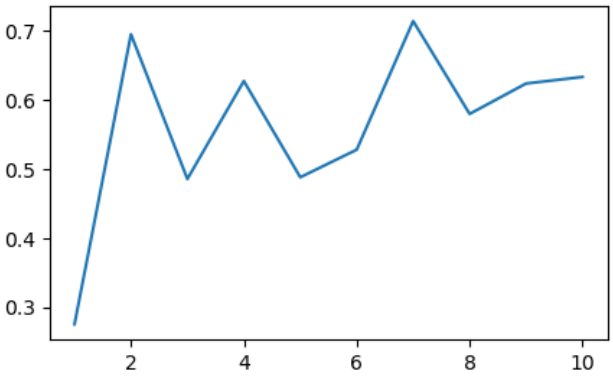

Figure 1: 生成器的损失函数曲线

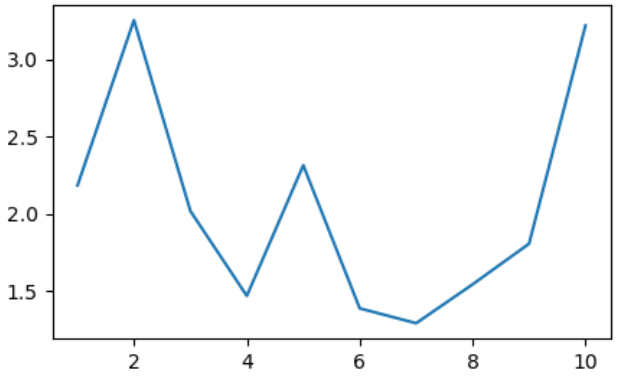

Figure 2: 判别器的损失函数曲线

gan 是个 minimax game，生成器和判别器一直在进行对抗，所以 loss 应该是很杂乱，一会上升一会下降。这就是 gan 难以训练的原因，我们不太能通过查看 loss 来说明 gan 训练得怎么样。

2.2 自定义一组随机数，生成 8 张图

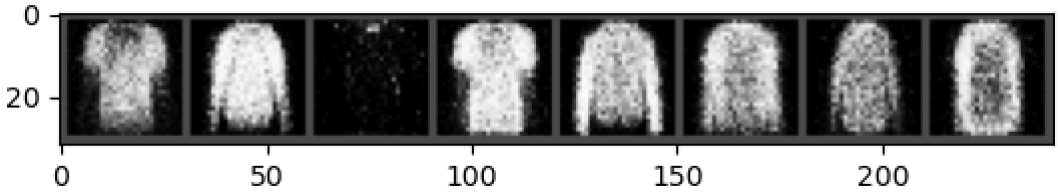

Figure 3: 利用随机数生成的 8 张图片

2.3 针对自定义的 100 个随机数，自由挑选 5 个随机数，查看调整每个随机数时，生成图像的变化（每个随机数调整 3 次，共生成 15x8 张图），总结调整每个随机数时，生成图像发生的变化。

为了方便我们画图，我们进行了如下操作，首先利用 `fixed_noise = torch.randn(8, 100, device=args.device)`生成了八个 100 维的随机数，作为生成图像使用的随机数。然后我们利用 `fixed_noise = fixed_noise.repeat(5, 1)`将这八个随机数分被复制五分，作为我们要更改的每五个位置。我们对于一行中的八个随机数图像，在一次实验中更改的随机数的位置相同，每次实验更改的随机数数值相同。所以一次实验可以画出一张 8×5 的图。每张图的第一行代表的是更改第一个位置的随机数，第二行是代表更改第 20 个位置的随机数，第三行代表的是更改第 40 个位置的随机数，第四行是代表第 60 个位置的随机数，最后一行代表的是更改第 80 个位置的随机数。

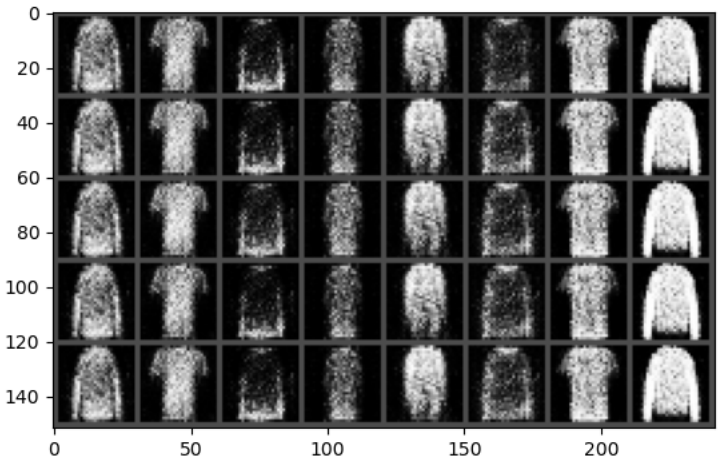

Figure 4: 将随机数指定为 **0.5**

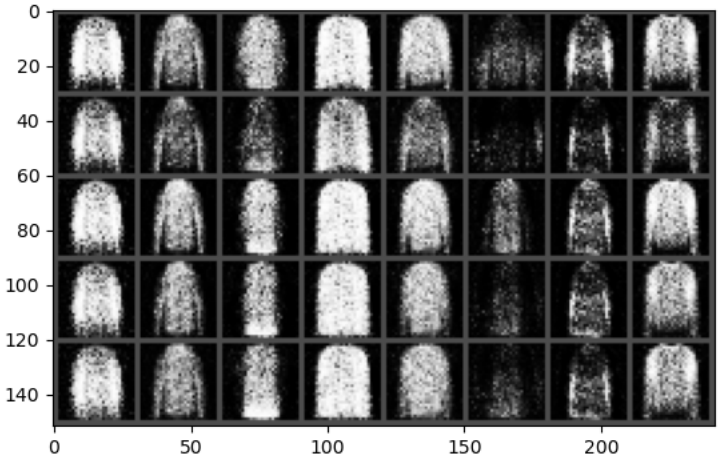

Figure 5: 将随机数指定为 **3**

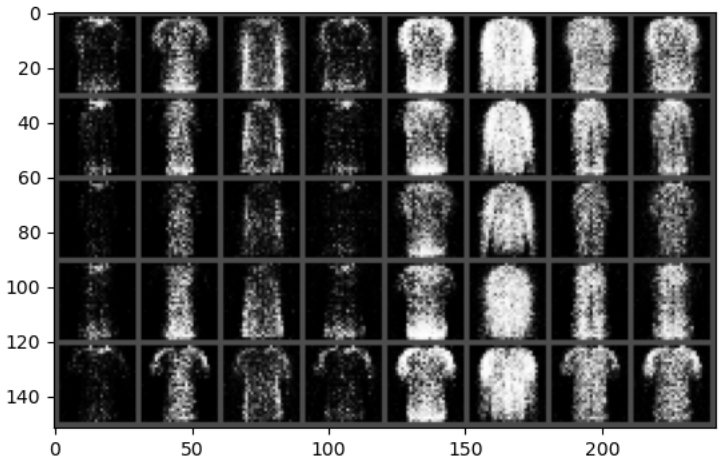

Figure 6: **将随机数指定为 10**

首先我们观察可以看出，当随机数设定的较小的时候（如 0.3），在不同位置改变一个这样的小随机数，收获的效果是比较小的，甚至可以说基本没有什么肉眼可以观察出来的变化；但是，当随机数设置的较大的时候（如 10），如果我们在不同位置改变一个这样的小随机数，收获的效果还是比较明显的，可以很清晰的从最后一张图上看出，相对于位置 60 变为 10，位置 80 的改变直接让裤子变成了两件衣服。当我们在观察相同位置的随机数改变时，我们发现，过小的随机数（0.3）或者过大的随机数（10）都会导致生成的图像的亮度较低，甚至有些图片接近于全黑，而当我们使用一个适中的随机数的时候，我们的模型生成的效果相对来说较为明亮，且生成的图像的效果较好。