南开大学
Nankai University

南　　开　　大　　学

网 络 空 间 安 全 学 院

深度学习平时作业

GAN 实验报告

穆禹宸　2012026

年级：2020 级

专业：信息安全、法学双学位班

指导教师：侯淇彬

2023 年 6 月 23 日

目录

一、 实验过程

(一) GAN

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

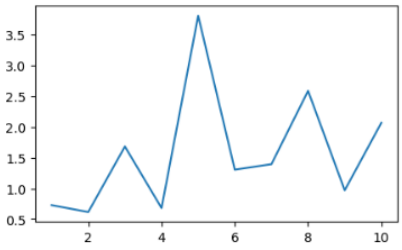

图 1: **生成器的损失**

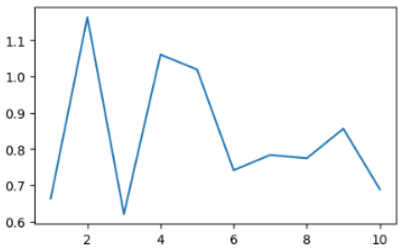

图 2: **判别器的损失**

这里由于训练的轮数比较少，因此没有完全反映出特点。理论上说：在训练初期，生成器会生成很差的假数据，这意味着鉴别器会更容易的检测到假数据，因此生成器的损失会很高，鉴别器的损失会很低。但是，随着时间的推移，生成器会逐渐生成更加真实的数据，鉴别器无法轻易区分真实数据和假数据的差异。此时，生成器的损失会变低，鉴别器的损失会变高。

因此，如果 GAN 网络的训练顺利，应该呈现的变化反应在损失函数上：生成器的损失随着时间的推移而变小，鉴别器的损失随着时间的推移而变大。最终，两个网络的 loss 都会趋于一个稳定的状态，说明 GAN 网络已经学会了生成接近真实数据的假数据。

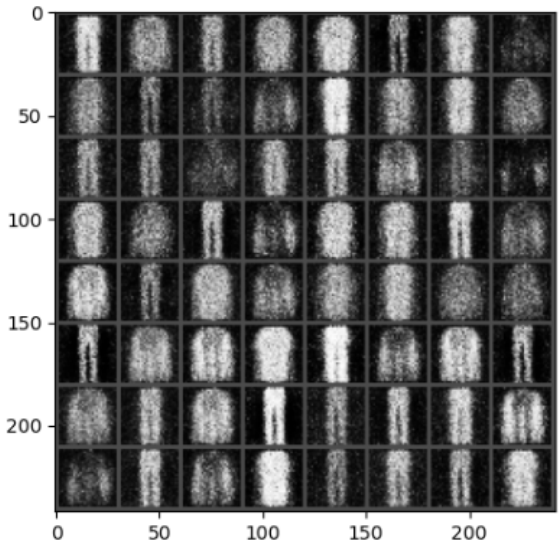

图 3: **某一轮生成的结果**

这是其中某一轮的结果，可见还是可以生成一些图片的。

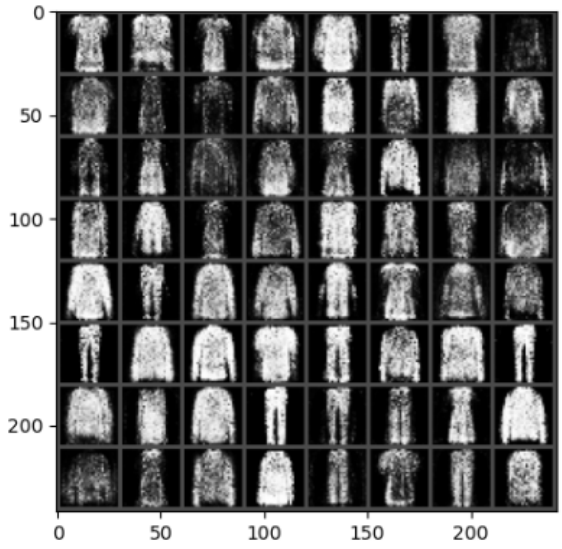

图 4: **生成的结果**

最终生成的结果还是保持了一定的质量的。

二、 解释不同随机数调整对生成结果的影响

自定义随机数，生成 8 张图

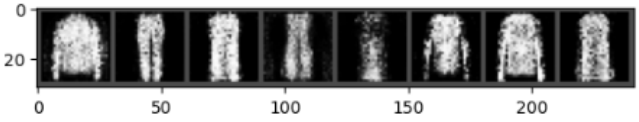

图 5: **生成的 8 张图**

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。

代码如下：

```
random_noise = torch.randn(8, 100, device=device)
x_gen = G(random_noise)
show_imgs(x_gen, new_fig=False)

random_list = [0.1,5,20]
```

这里注意，我们把原来的噪声分别改为 0.1，5，20. 得到如下的效果：

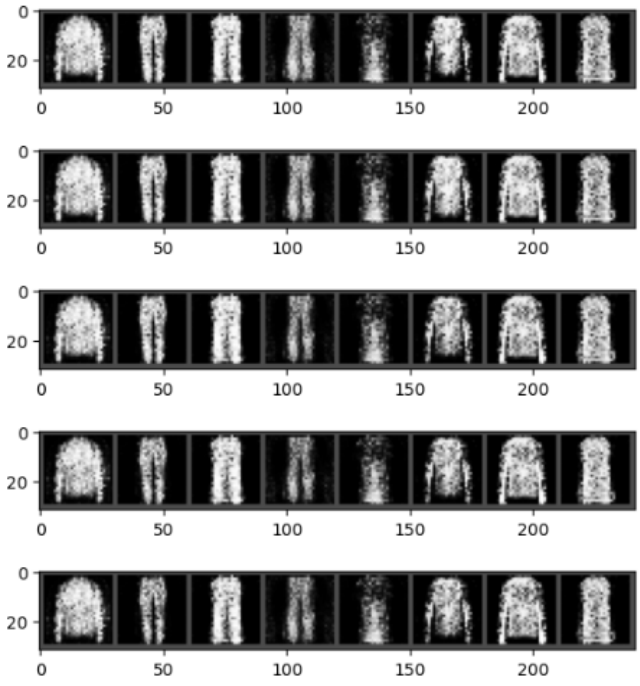

图 6: **噪声改为 0.1**

从上面的图片可以看出，当我们把随机数改为 0.1 时，生成的图像没有巨大的变化，肉眼几乎不可见，只有一些细微的差别。当然，由于我们的随机数生成的是均值为 0，方差为 1 的正态分布，因此从结果上看大部分随机数的绝对值会大于 0.5。当我们把噪声改为 0.1 时，生成的图像可能会变得更加清晰，比如我这 8 张图片之中的“裤子”图片，其两个裤腿的分叉变得比最初大了一些，图像变得更好了一点。**但是，这里对生成图像的影响几乎微不足道，这里之所以取得了一定的效果也是取决于原来的值**。而对于其他图片，我们完全观察不出任何变化。

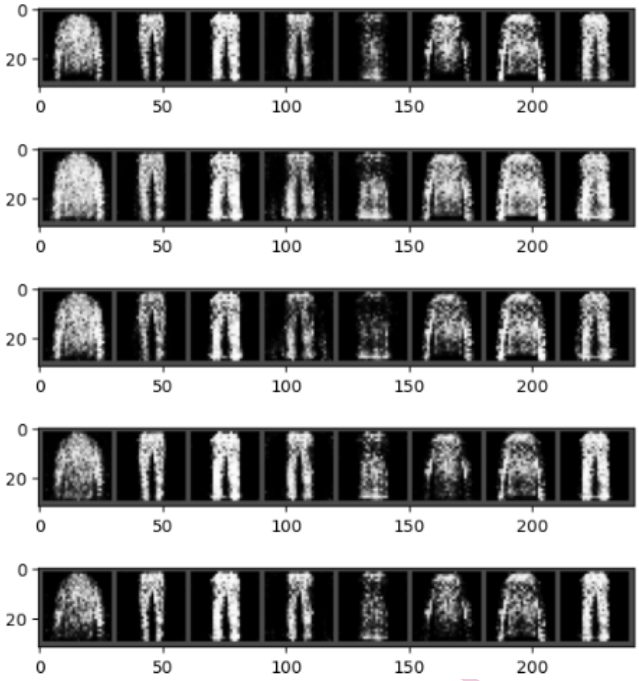

图 7: **噪声改为 5**

当我们把随机数改为 5 时，可以看到这里生成的图像开始出现一定的变化，首先是会使生成的图像变得更加模糊，清晰度开始下降，比如其中“裙子”的图片开始变得失去细节，逐渐和背景融为一体，有向随机噪声转化的趋势。因为这些值会使输入的随机噪声与训练数据的实际分布相差更远，从而产生更多的噪音，使生成器模型难以生成质量更高的图像。

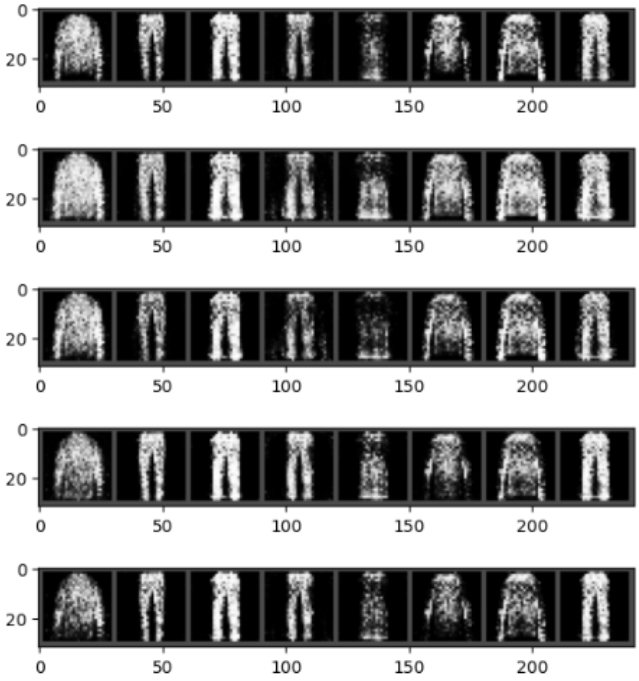

图 8: **噪声改为 20**

而当我们把随机数改为 20 时，生成图片的质量开始严重下滑，首先是清晰度很低，看不到

原始图片那样的清晰度（体现在我们的图片之中没有那样集中的白色）。而且逐渐出现了一些不真实、混乱的图像，几乎无法分辨其是“裤子”还是“裙子”，同时生成的结果开始趋同，几乎大多数肉眼可识别的图片都变成了“裤子”，几乎失去了其他类别。

因此我们可以看到，过大的噪声值会使得生成的图像中存在更多的噪声和不规则性，从而使图像变得更加模糊。生成的图像可能会失去一些细节和质感。太大的噪声值可能会使整个图像更加“扁平”，失去细节和质感。这是因为过大的噪声值会使得生成的图像过多地关注“特别嘈杂”的区域，从而影响了整体的质感和细节表现。

对于较小的噪声来说，也不一定就完全是对的。较小的噪声会导致生成器干扰减少，能迅速拟合出图像，但此时图像多样性会减小，比如“衣服”的“领子”“袖”等位置有一些细节其实在很小的噪声时会消失。因此需要根据应用场景和需求权衡选择合适的噪声大小。**关键是要保证噪声的分布合理，比如与原来的数据保持一致，就是一种合理的分布。**

三、 自己实现的 DCGAN

(一) 网络结构

```
Generator(
  (main): Sequential(
    (0): ConvTranspose2d(100, 256, kernel_size=(4, 4), stride=(1, 1))
    (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): ConvTranspose2d(256, 128, kernel_size=(3, 3), stride=(2, 2), padding
        =(1, 1))
    (4): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding
        =(1, 1))
    (7): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): ConvTranspose2d(64, 1, kernel_size=(4, 4), stride=(2, 2), padding
        =(1, 1))
    (10): Tanh()
  )
)
Discriminator(
  (main): Sequential(
    (0): Conv2d(1, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (5): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
```

```
24      (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
            track_running_stats=True)
25      (7): LeakyReLU(negative_slope=0.2, inplace=True)
26      (8): Conv2d(256, 1, kernel_size=(4, 4), stride=(1, 1))
27      (9): Sigmoid()
28    )
29  )
```

(二) 实验结果

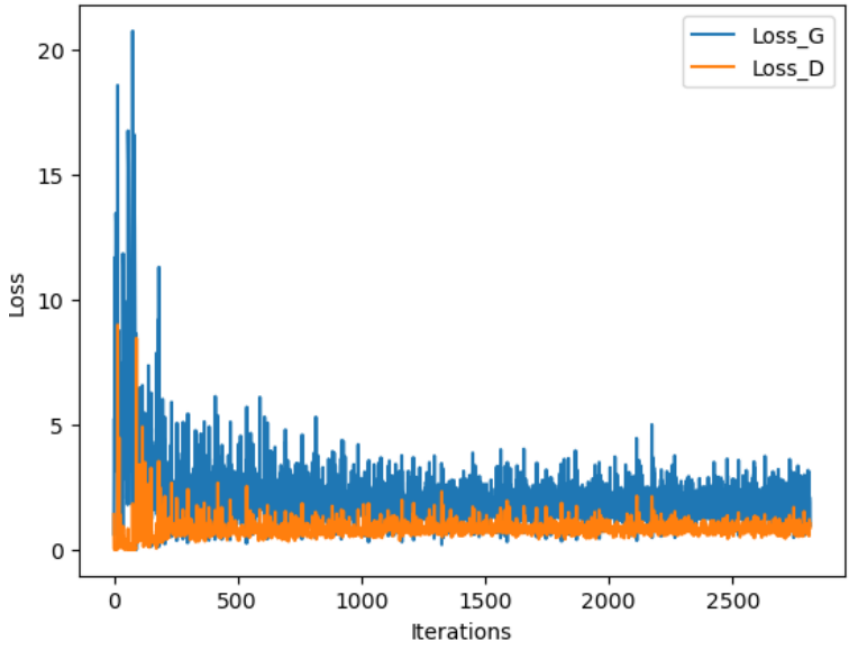

图 9: **判别器和鉴别器的损失**

可以看到损失整体上是下降并且接近相同的趋势。

图 10: **生成的结果**

可以看到效果比原来要好得多。