南開大學
Nankai University

计算机学院

深度学习实验报告

GAN 实验报告

姓名：徐文斌

学号：2010234

专业：计算机科学与技术

2023 年 6 月 24 日

目录

1 原始版本 GAN

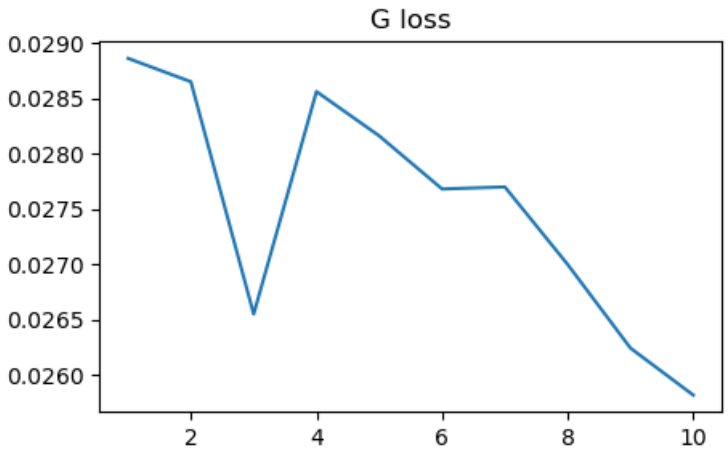
1.1 网络结构

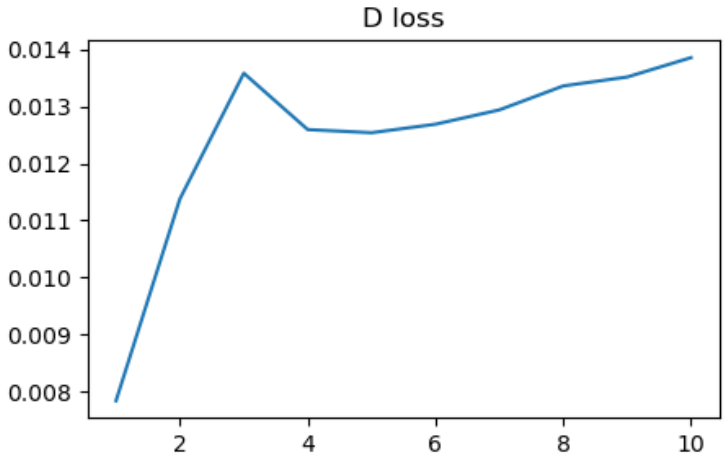
使用 print 函数打印网络结构如下所示：

```
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)
)
```

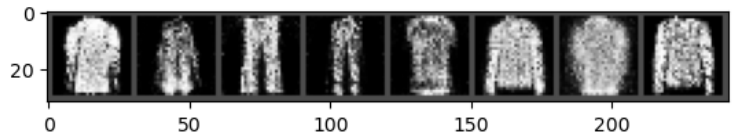
1.2 训练损失曲线

训练损失曲线如下：





1.3 自定义一组随机数，生成 8 张图



1.4 调整随机数，观察图像变化

我们首先生成一组 8x100 维的随机数，用于生成 8 张图片，然后将这组随机数复制五份，得到五组相同的随机数。对于每组随机数，我们将其中的某些随机数修改为一个定值。具体规则为每隔 20 个数修改一次。对第一组随机数修改第 0 个位置的随机数；第二组随机数修改第 20 个位置的随机数，以此类推。

分别将随机数指定为 0.1、10、100，得到如下三张图片：

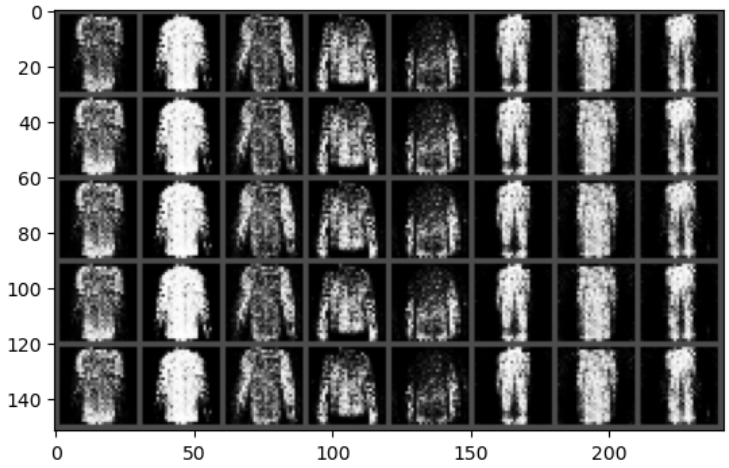


图 1.1: 随机数指定为 0.1

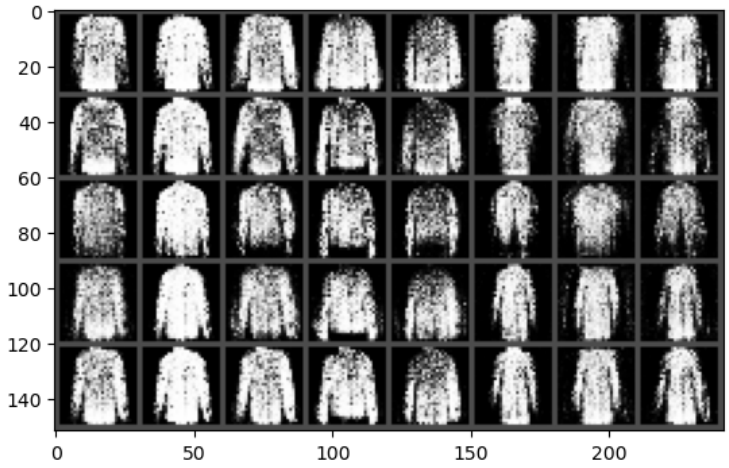


图 1.2: 随机数指定为 10

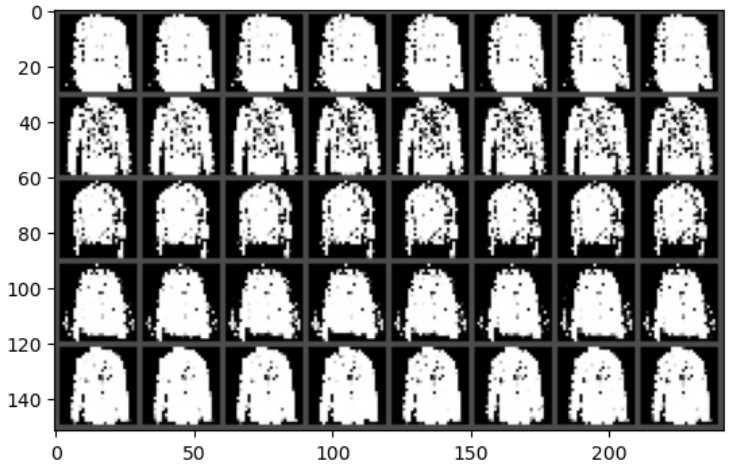


图 1.3: 随机数指定为 100

分析上述图片。显然，第一幅图的每一行都基本相同，并未发生肉眼可见的变化。这应该是因为将随机数设置为 0.1，该数比较小，乘以相应的权重之后对结果的改变也较小，所以图片相差的都不多。

继续观察将随机数指定为 10 时的图片，此幅图片与第一幅图片已经有了明显的区别。且可以发现该幅图片内部同一列内有些图片产生了较为明显的区别，如最后一列的第三行和最后一列的第五行，图片几乎从裤子变成了上衣。因此我们可以得出，不同位置的数的权重不同，控制生成不同类型的图片，随着所指定的数的增大，结果的变化也开始增大。

第三幅图片中我们指定随机数为 100，可以看到同一行图片几乎一样。推测是因为指定随机数的值过大，从而掩盖掉了其他维上的随机数对结果造成的差异，导致最终生成的图像只由该随机数决定，所以一行内的图片几乎完全相同。

2 DCGAN

2.1 网络结构

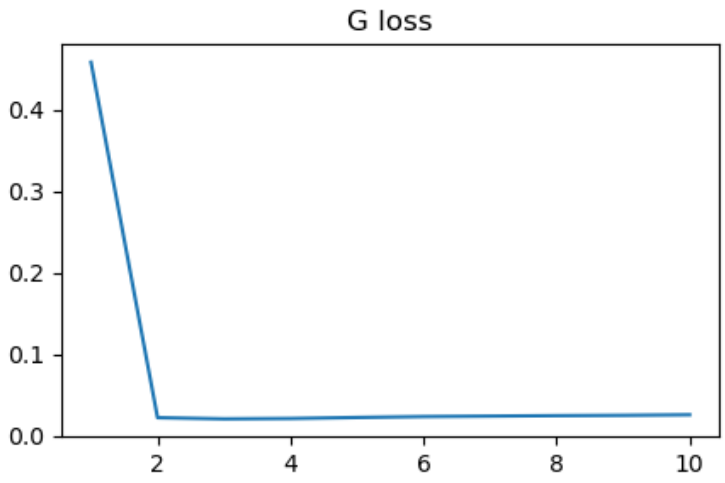
使用卷积实现生成器和判别器，实现 DCGAN 网络，这里对 DCGAN 网络进行了简化，具体结构如下：

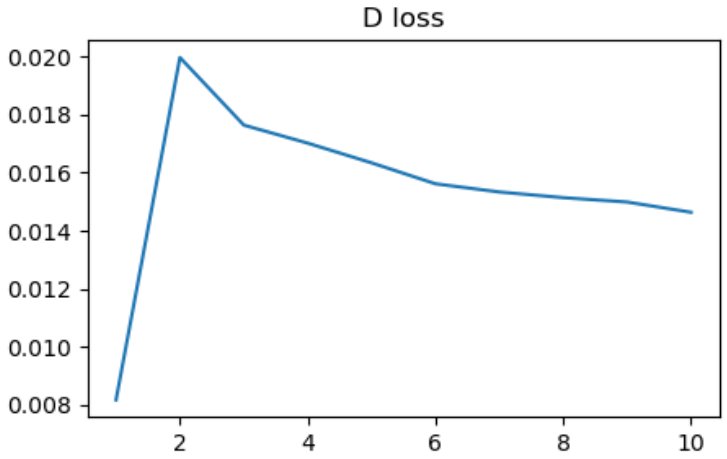
```
Discriminator(
  (model): Sequential(
    (0): Conv2d(1, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): LeakyReLU(negative_slope=0.2)
    (2): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (3): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (4): LeakyReLU(negative_slope=0.2)
    (5): Conv2d(64, 128, kernel_size=(4, 4), stride=(1, 1))
    (6): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (7): LeakyReLU(negative_slope=0.2)
    (8): Conv2d(128, 1, kernel_size=(4, 4), stride=(1, 1))
    (9): Sigmoid()
```

```
  )
)
Generator(
  (model): Sequential(
    (0): ConvTranspose2d(100, 192, kernel_size=(4, 4), stride=(1, 1))
    (1): ReLU()
    (2): ConvTranspose2d(192, 96, kernel_size=(4, 4), stride=(1, 1))
    (3): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (4): ReLU()
    (5): ConvTranspose2d(96, 48, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (6): BatchNorm2d(48, eps=1e-05, momentum=0.1, affine=True,
        track_running_stats=True)
    (7): ReLU()
    (8): ConvTranspose2d(48, 1, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (9): Tanh()
  )
)
```

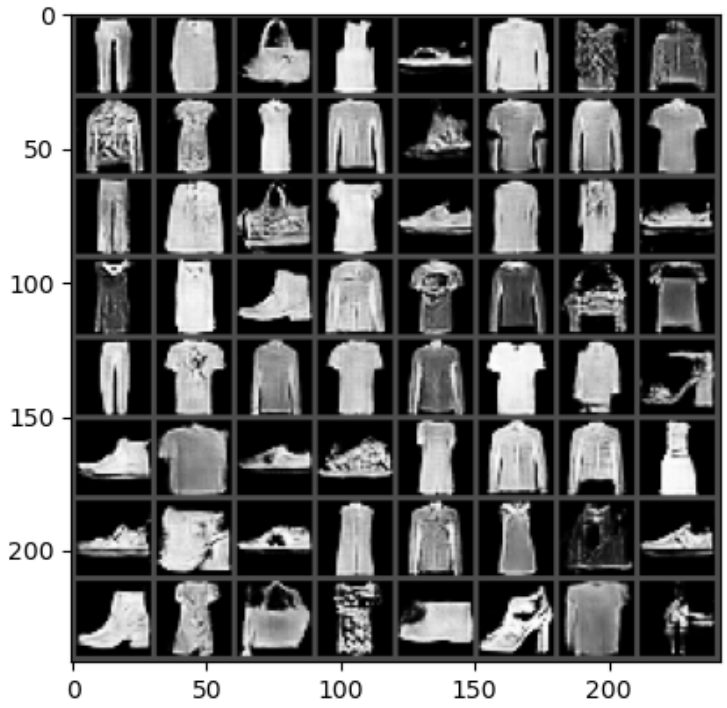
2.2 训练结果

训练损失曲线如下：





训练 10 轮得到生成的图片如下，可以看到相比于普通的 GAN 网络，DCGAN 生成的图片更加清晰易于辨认。



2.3 调整随机数，观察图像变化

DCGAN 生成的图片更加清晰，我们可以更明显地观察到图像地变化。我们按照上文中提到的方法，调整随机数，观察 DCGAN 生成的图像的变化，结果如下所示。

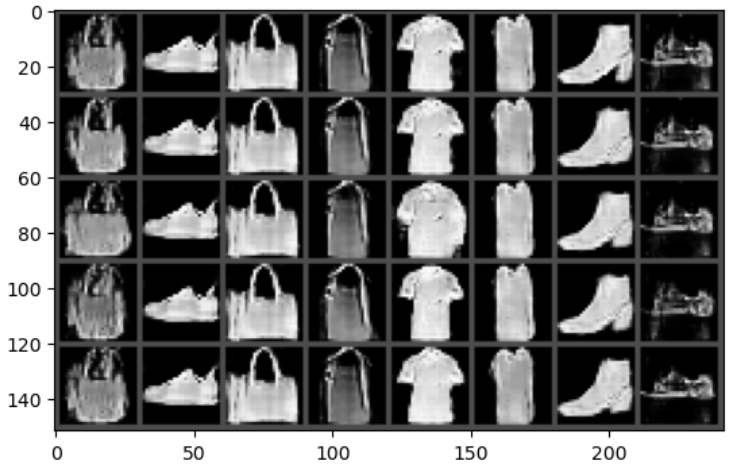


图 2.4: 随机数指定为 0.1

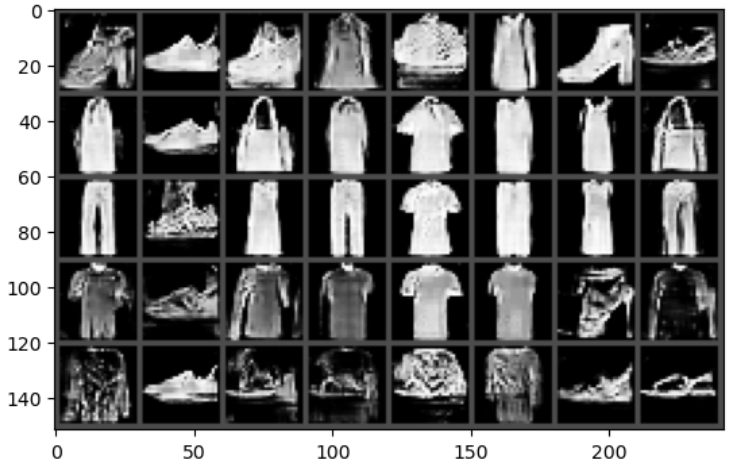


图 2.5: 随机数指定为 10

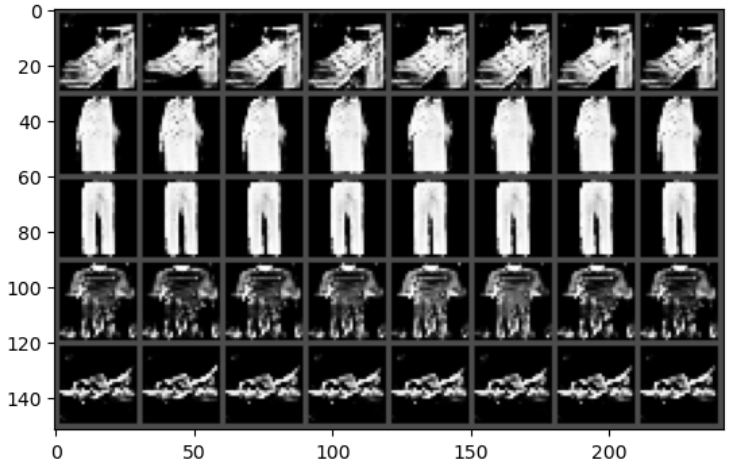


图 2.6: 随机数指定为 100

我们主要观察不同图片中同一行内容的变换，即将相同位置的随机数分别指定为 0.1、10、100。我

们以第一行为例，可以看到第一幅图片的第一行中有多种图像比如提包、鞋子、上衣等等；当我们将随机数指定为 10 后，即第二幅图片，第一行中的一些图片向鞋子发生了一定程度上的转化，比如第一行第一列和第一行第三列的提包已经有了鞋子的大致样子。当我们继续将随机数指定为 100 后，即第三幅图片，第一行所有的图像全部变成了鞋子。观察其他行也可以得到类似的规律，只是最终变换至的图像不同。我们可以理解为，输入给网络的向量中的不同位置处的随机数决定了生成不同的图像，当决定鞋子的那个位置的数值越来越大时，图像也越来越像鞋子。这和我们在观察普通的 GAN 网络时得出的结论是一致的。