pytorch 生成对抗网络编程_GAN 的简单实例代码

目录

本文代码来源于《pytorch 生成对抗网络编程》的前半部分，并做了一些小改动。为了方便读者使用，其中代码已经做过 debug（主要是图片显示部分）在 python3+cpu 环境下可一键运行。关于 GAN 的基础理论请读者自己了解。

本文的数据和代码均在同一个压缩包，使用代码文件时，请确保所有文件在同一个文件夹目录下。

Instance01

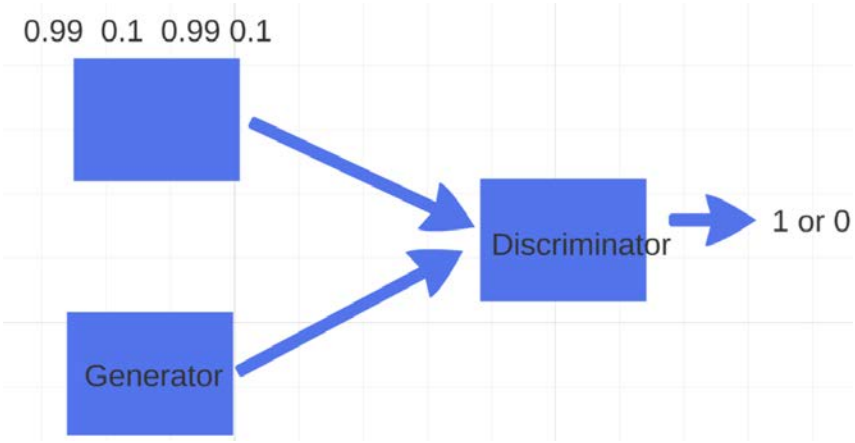
一、1010 格式规律生成

用 GAN 生成 1010 格式规律的数字

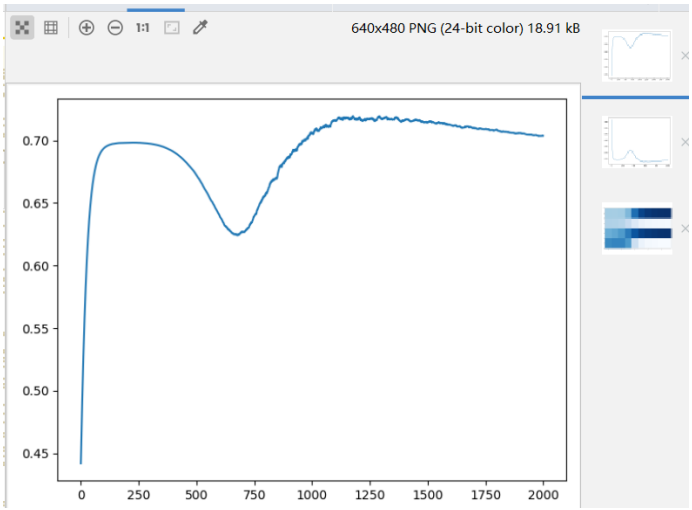
D：能够区别符合 1010 规律的数字，符合为 1，不符合为 0。

G：能够生成 1010 规律的数字。

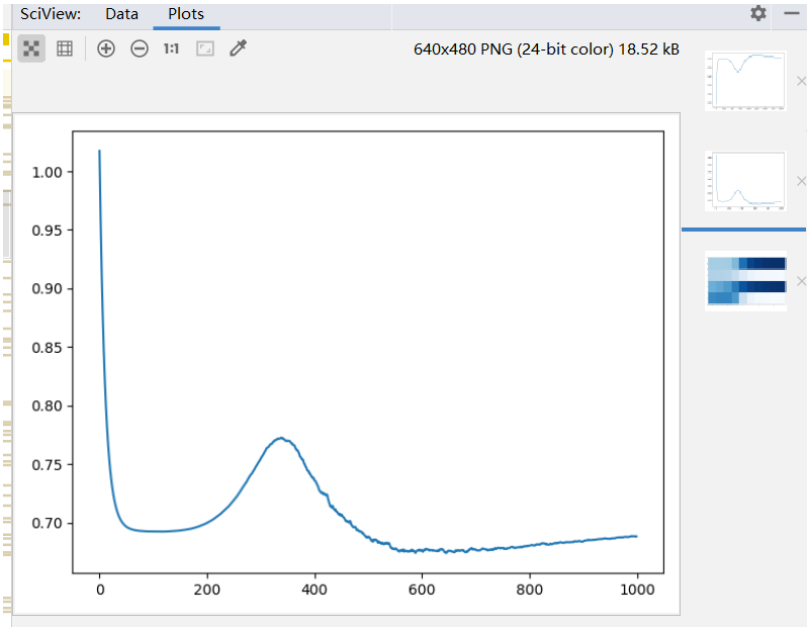
使用 BCEloss，理论值为 0.69



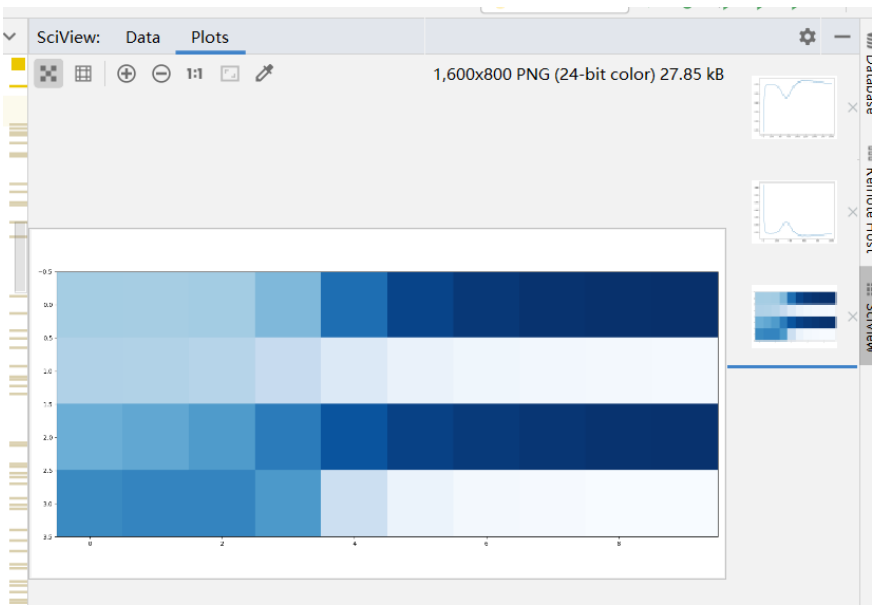
1010GAN



D loss



G loss



1010GAN 训练结果

Instance02

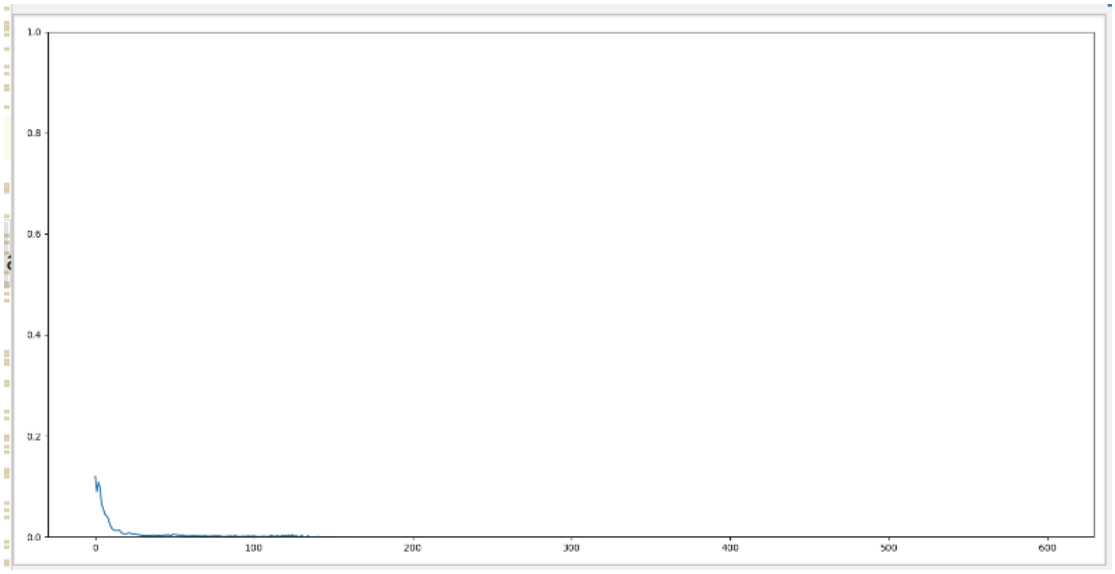
一、mnist 手写数字分类器

这里实现了一个最简单的 mnist 手写数字图片分类器。仅有线性层和 sigmod。使用 MSEloss。其中代码作为 mnistGAN 的参考。

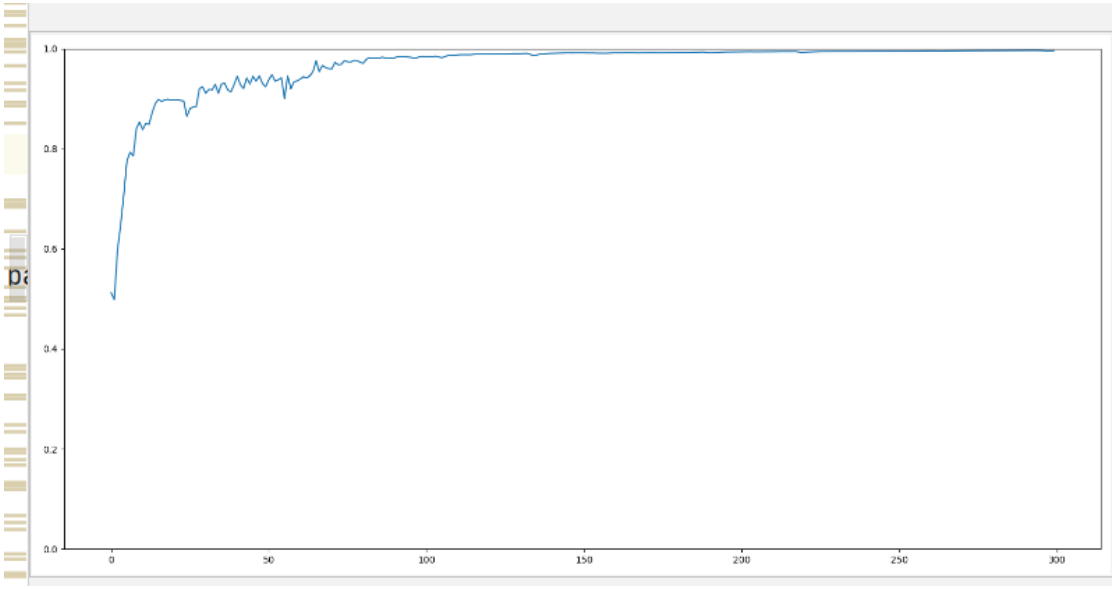
Instance03

一、会产生模型崩溃的 mnistGAN

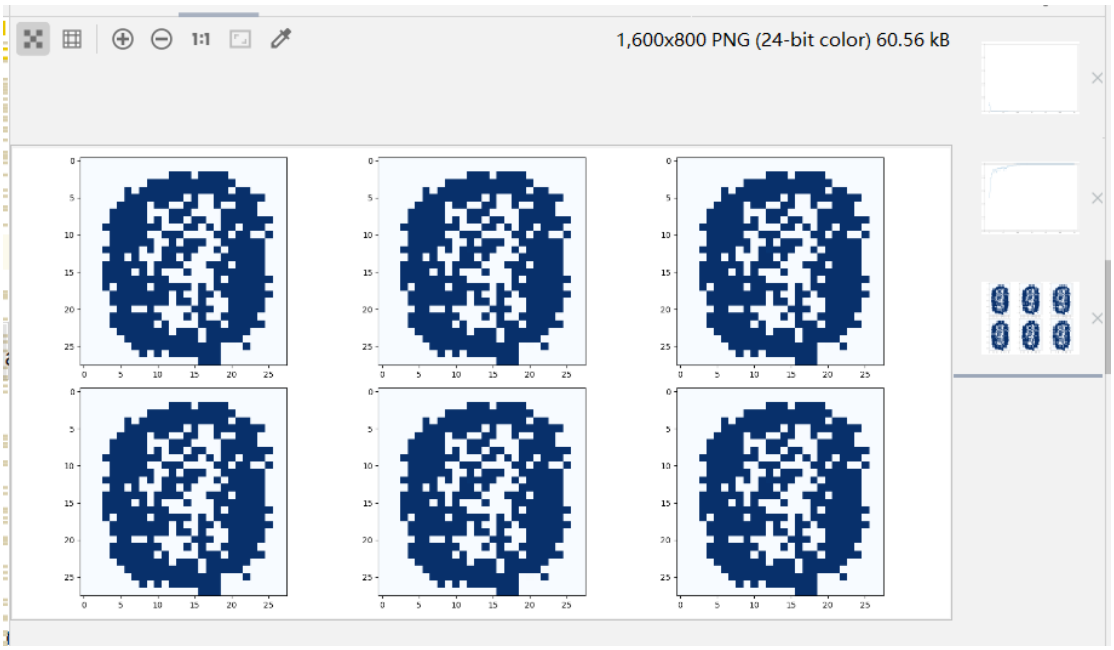
GAN 在拟合的 PG 和 Pdata 相差较大的时候，会发生 mode collapse。导致 G 实际上只学习到部分信息。运行结果展示如下。


1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600

D loss


1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300

G loss



随机生成 12 张手写图片

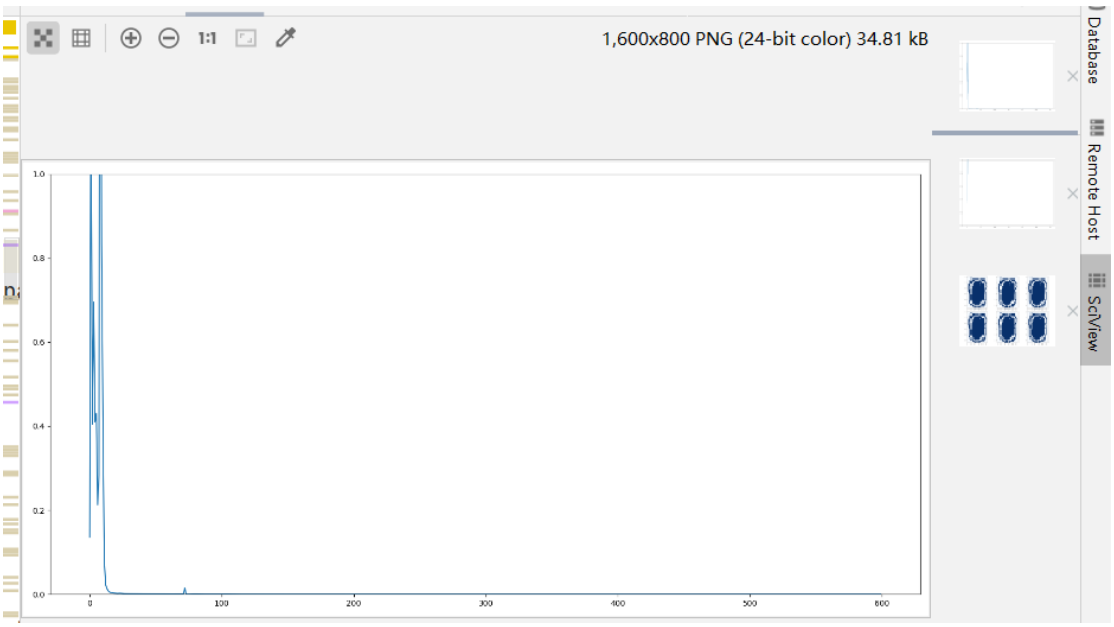
Running time: 72.11255264282227 Seconds

Instance04

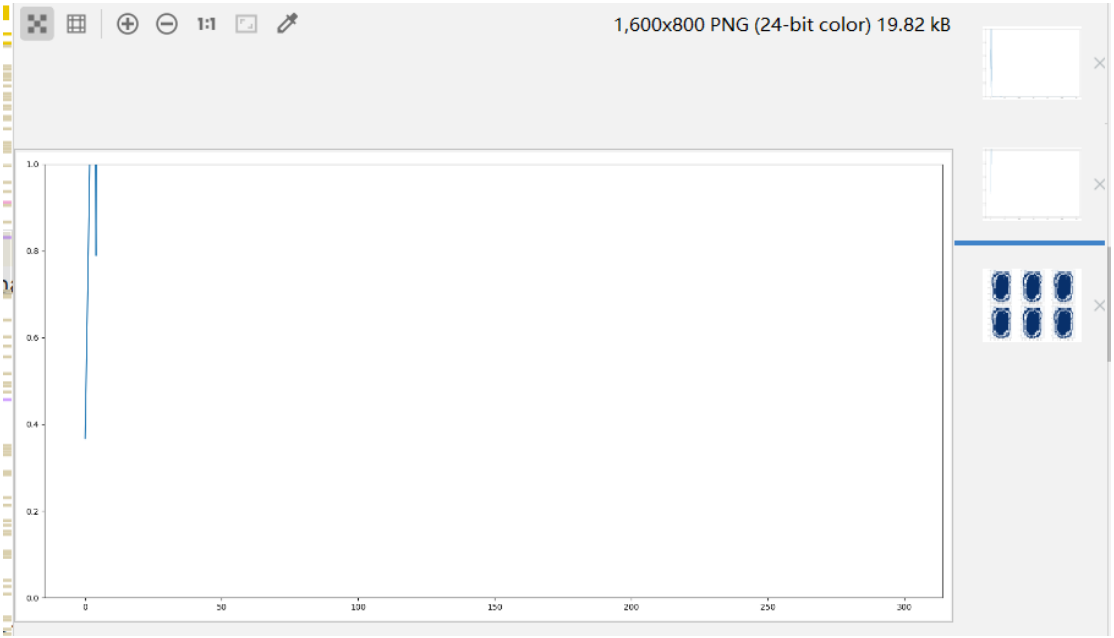
一、对崩溃 mnist_GAN 的改进

改进：

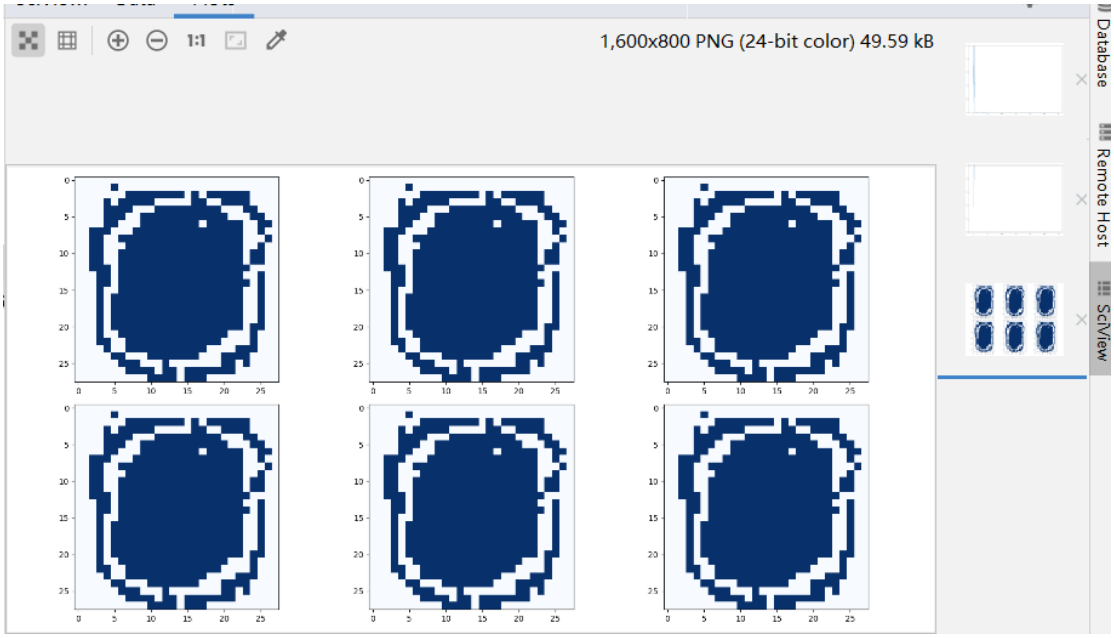
1、在 G,D 中替换一个 sigmod 为 LeakyRelu，以增大梯度。其后添加一个 Layernorm 层，以使得数据均一化。优化器用 Adam，该优化器可以解决局部最小问题和梯度减小问题。损失用 BCEloss，使其能够惩罚网络输出。把 G 的输入由一个数字改成 100 个数字，以提高输入的随机性和空间大小。


1,600x800 PNG (24-bit color) 34.81 kB
1:1
Database
Remote Host
SciView
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600

D loss


1,600x800 PNG (24-bit color) 19.82 kB
1:1
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300

G loss



随机生成

实际上，通过改进网络层不能一下子解决崩溃问题。

Instance05

一、解决模式崩溃

以上改进依然会发生模式崩溃，但是清晰度和手写度有提升。

（1）输入生成器的随机种子和　输入鉴别器的随机种子应该不同。

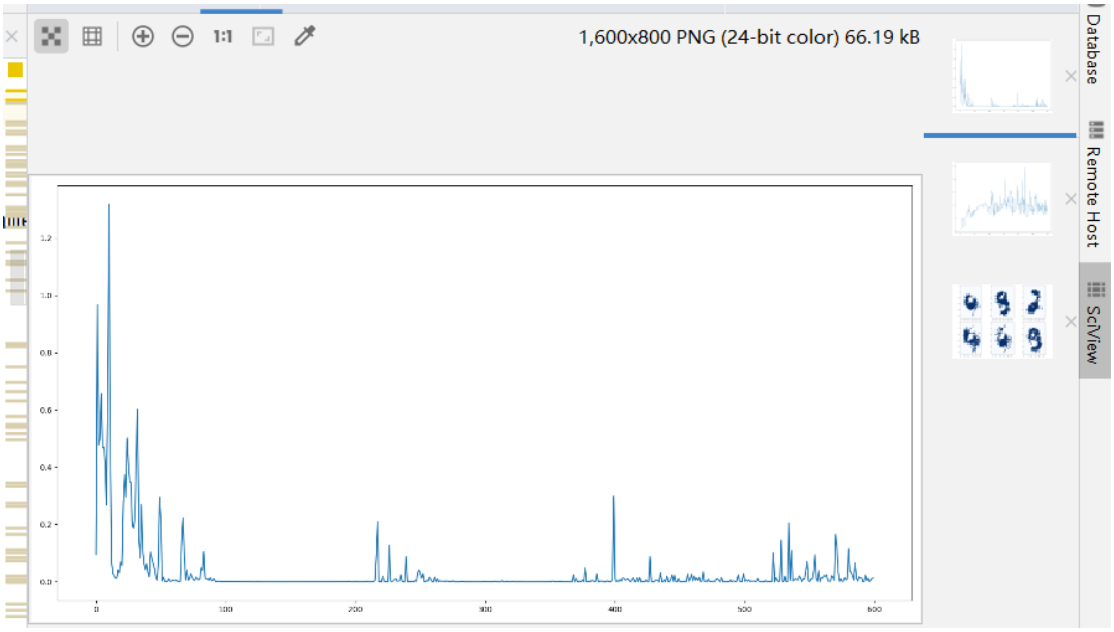
将随机种子函数进行改变。输入鉴别器的图像的像素值应该在 0-1 范围内符合 U（0,1）随机，而输入生成器的随机数应该是 N-（0,1）分布的，其数值不需要在（0,1）之间。注意到，由于 G 的网络最后一层是 layernorm+sigmod，因此输出符合分布 U(0,1)。

将随机数代码重写成：

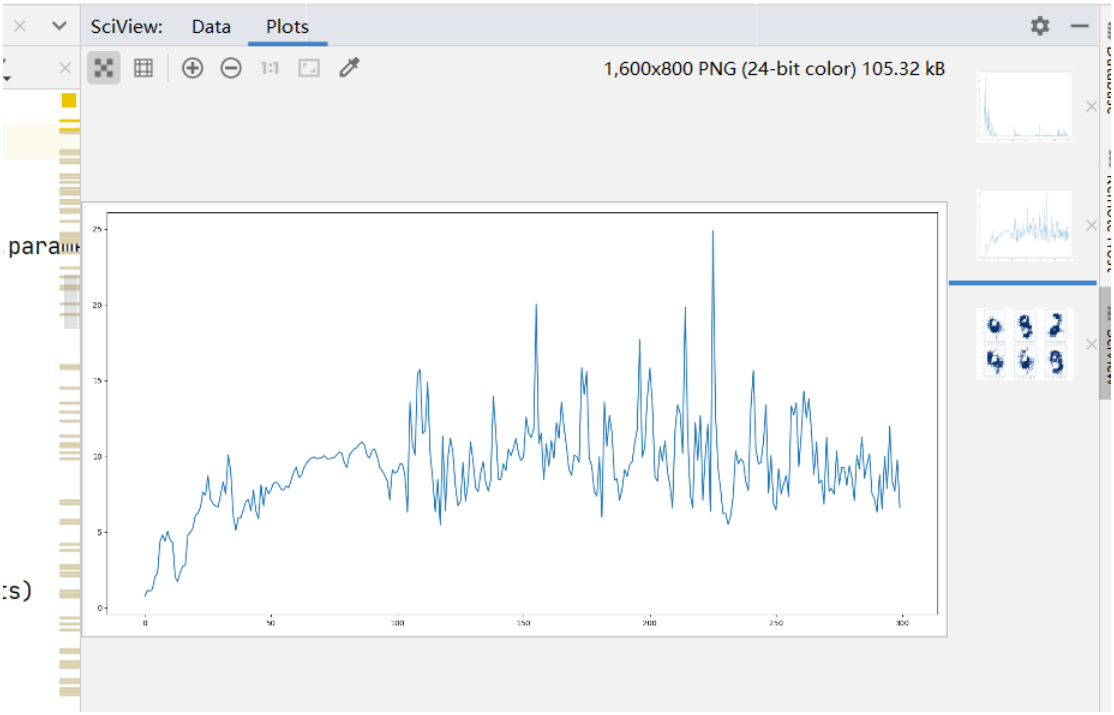
```
def generate_random_seed(size):
    return  torch.randn(size)

def generate_random_image(size):
    return  torch.rand(size)
```

相应修改训练代码：

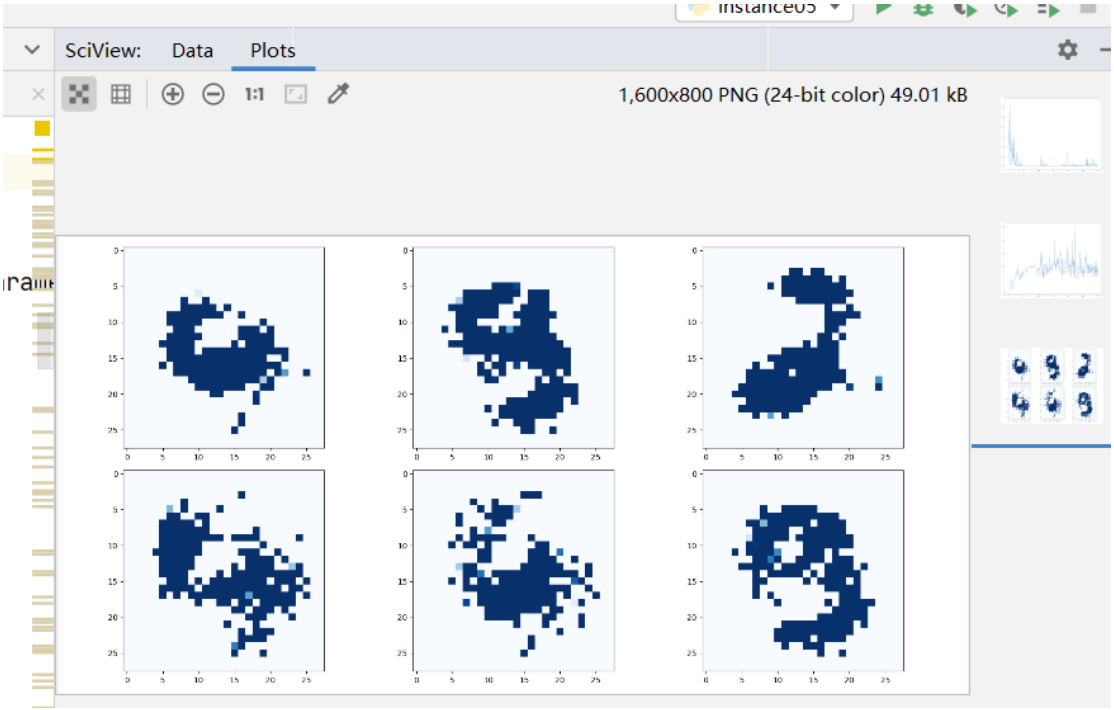
```
for i in range(epoch):
    for batch in loader:
        label, image, target = batch
        # 真实数据训练 D
        D.trainit(image,torch.ones(loader.batch_size,1))  #  label 为 B*1 （B个1）
        # 使用G 的生成样本训练 D
        D.trainit(G.forward(generate_random_seed((loader.batch_size,100))),torch.zeros(loader.batch_size,1))
        # 训练生成器
        G.trainit(D,generate_random_seed((loader.batch_size,100)),   torch.ones(loader.batch_size,1) )

        pass
```



D loss



G loss



随机生成的图片

注意到，这样的结果并不是每次都可重现的。

Instance06

一、种子实验

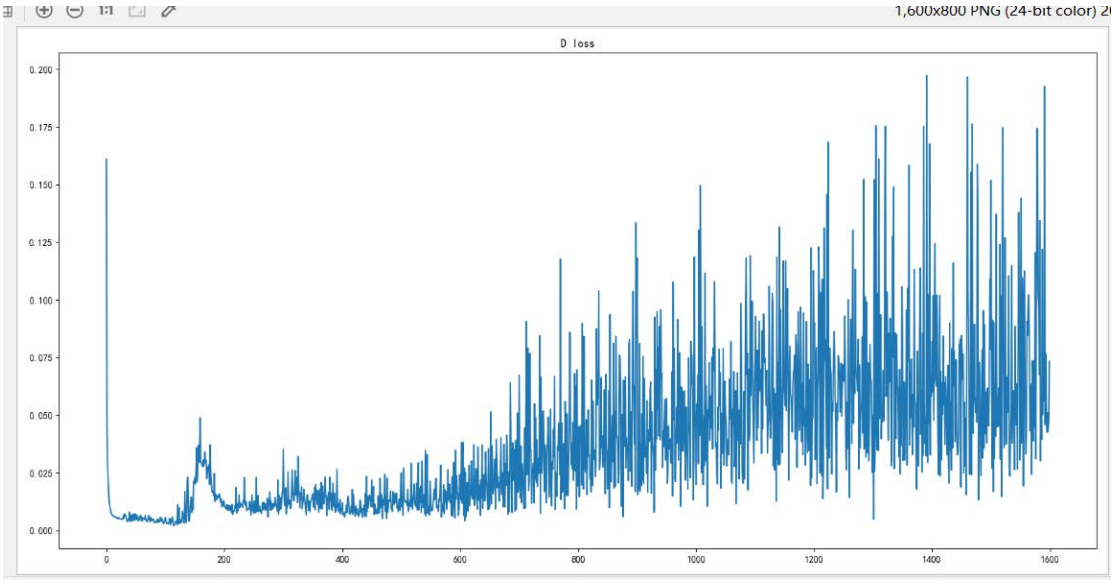
基于 instance05，我们把输入 generator 的输入维度从一个数字改成了 100 个数字。这样其空间表示大大增加了。那么，由不同的随机向量来 生成的图片，与随机向量本身有什么样的联系呢？

假设随机一个 种子 seed1 和 seed2，我们观察三种情况：

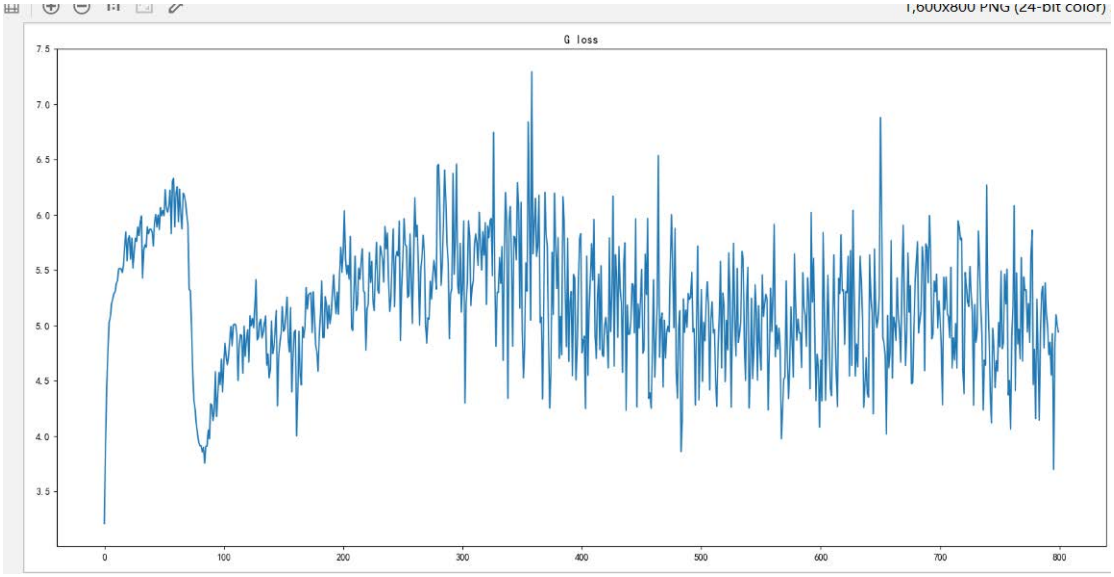
1、seed1 和 seed2 及其之间的过渡向量能生成什么图片。（过渡的意思可以理解为等差数列）。

2、seed1+seed2 能生成什么。

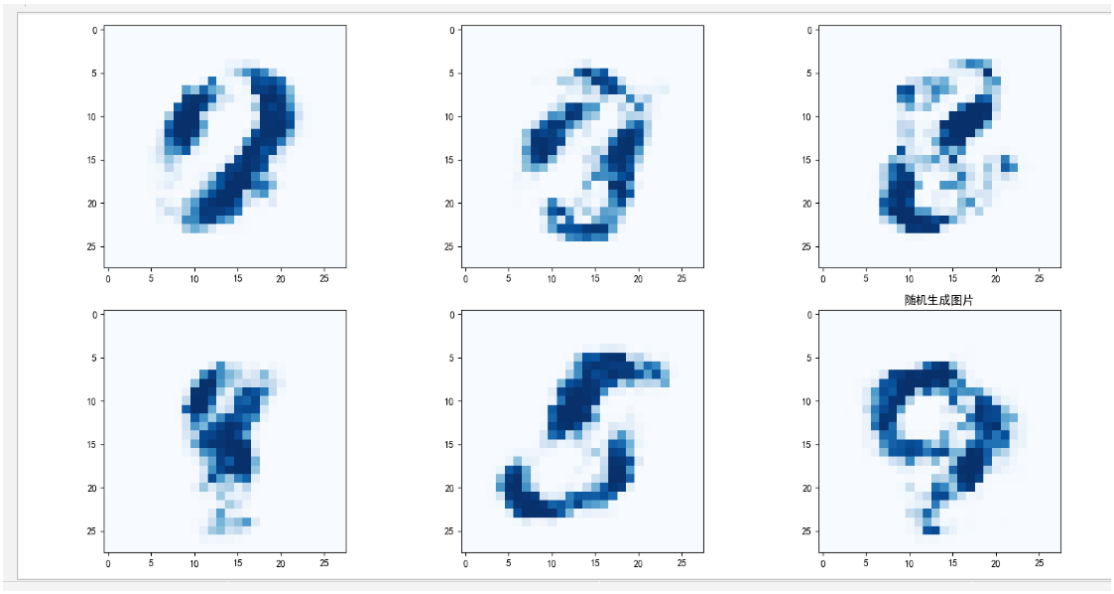
3、seed1-seed2 能生成什么。



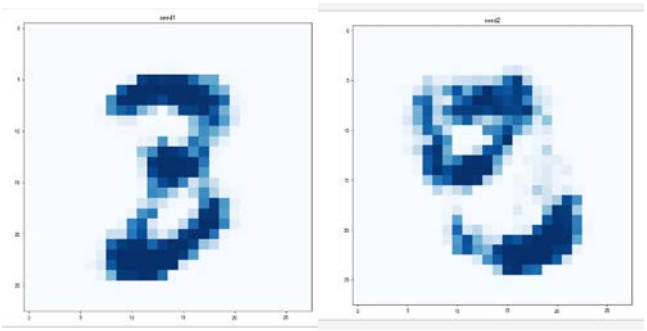
Discriminator Loss



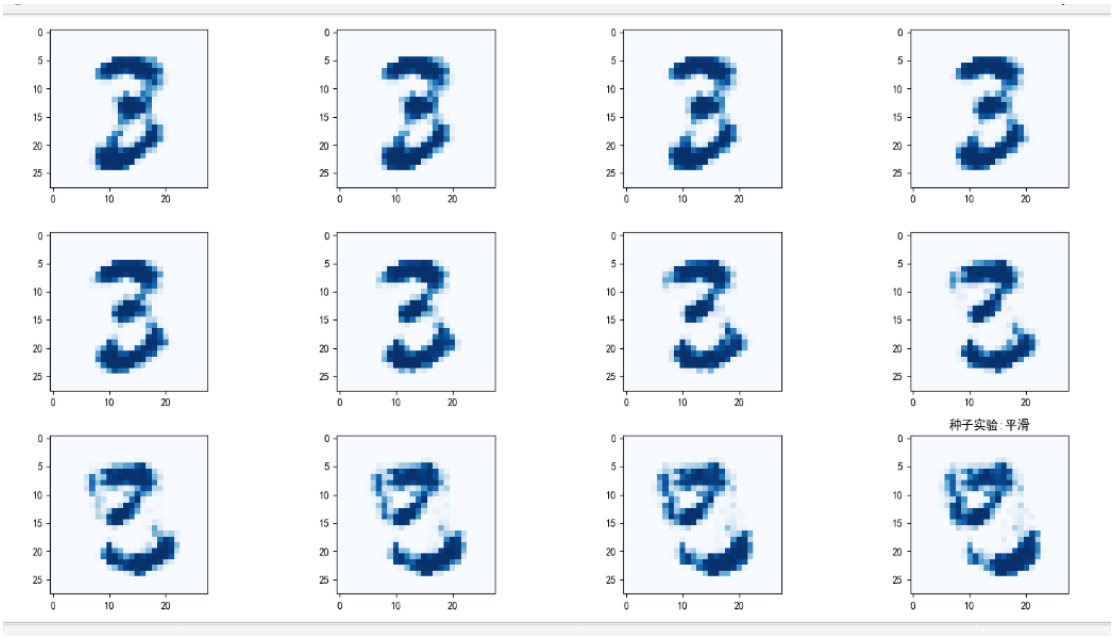
Generator Loss



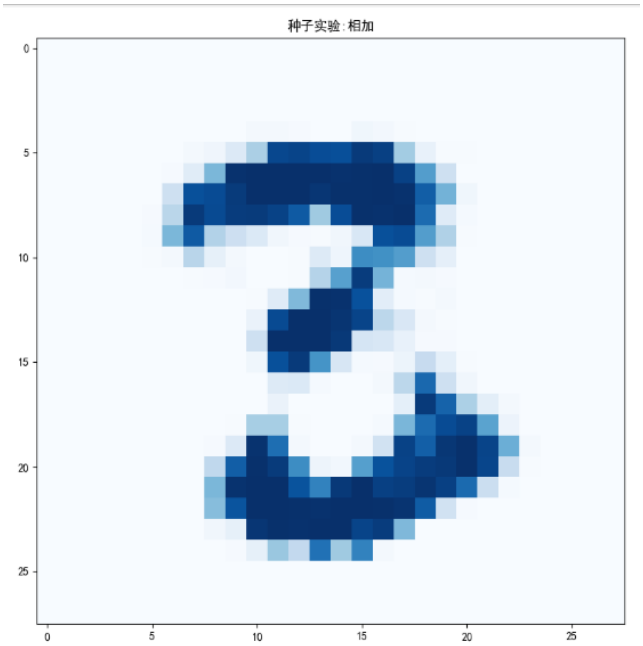
随机生成 12 张手写数字



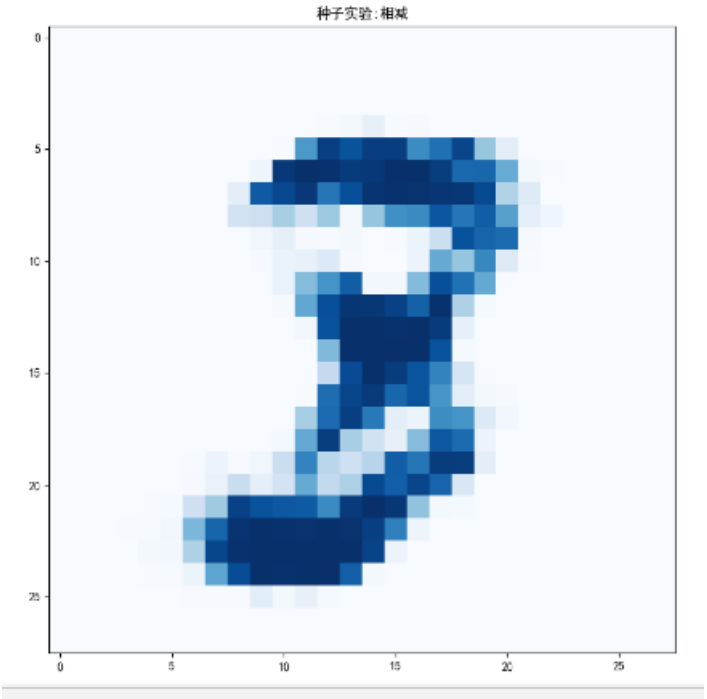
Seed1 & Seed2



由 Seed1 过渡到 Seed2（从左至右，从上至下）


种子实验:相加
0
5
10
15
20
25
0
5
10
15
20
25

种子相加


种子实验:相减
0
5
10
15
20
25
0
5
10
15
20
25

种子相减