ANN HW4

计02 刘明道 2020011156

Basic

1. 训练过程

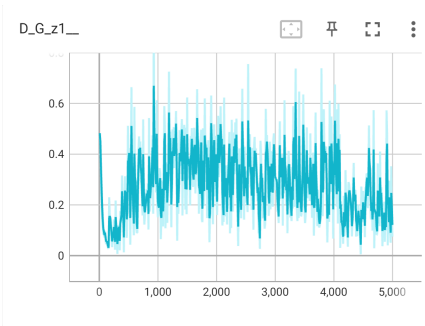
实验中选用的参数为

```
batch_size=64
num_train_steps=5000
seed=2022
```

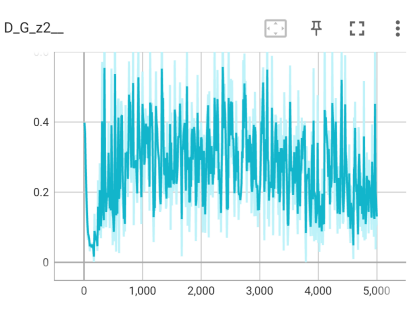
下面列举 `latent_dim={16, 100}` ， `hidden_dim={16, 100}` 共 4 组结果。很多指标的振荡相当剧烈，为了更清晰地观察指标的变化趋势，在 TensorBoard 中使用 Smoothing=0.4 。

`latent_dim=16`, `hidden_dim=16`

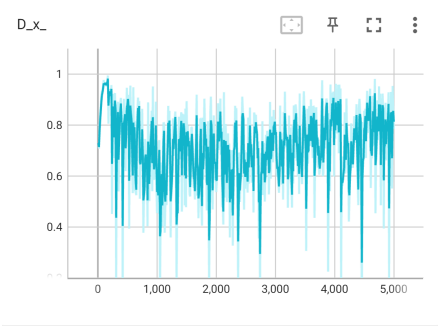
D_G_z1



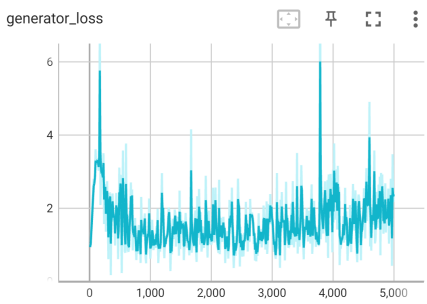
D_G_z2



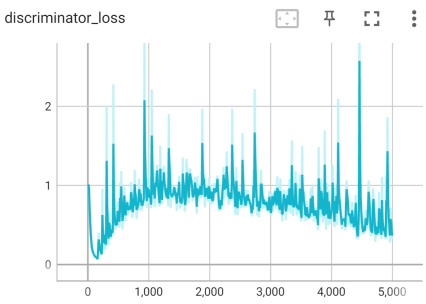
D_x___



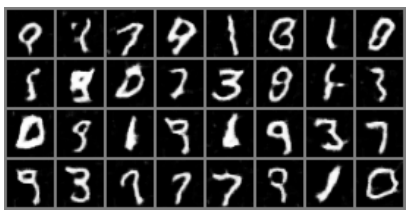
Loss_G



Loss_D

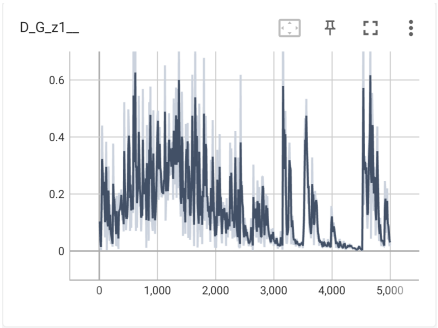


Image@4999

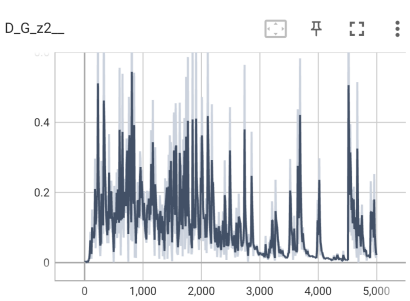


`latent_dim=16`, `hidden_dim=100`

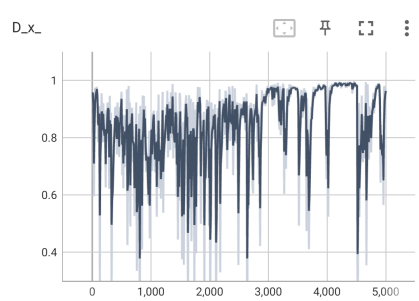
D_G_z1



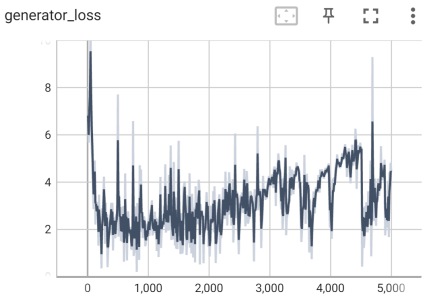
D_G_z2



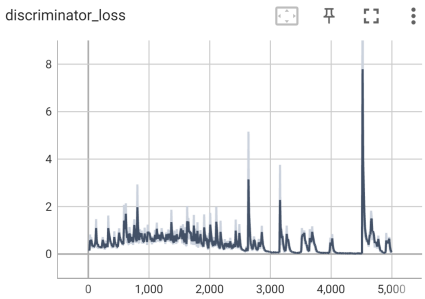
D_x__



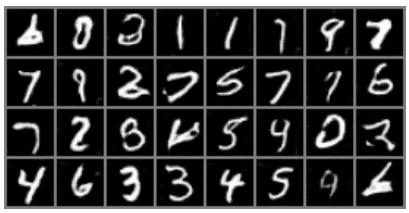
Loss_G



Loss_D

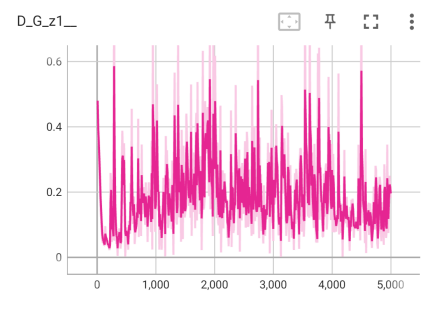


Image@4999

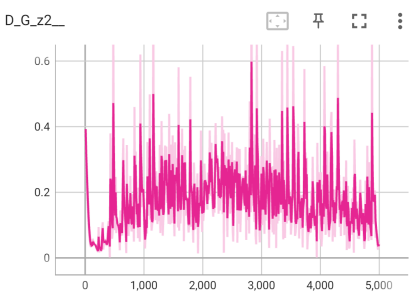


`latent_dim=100`, `hidden_dim=16`

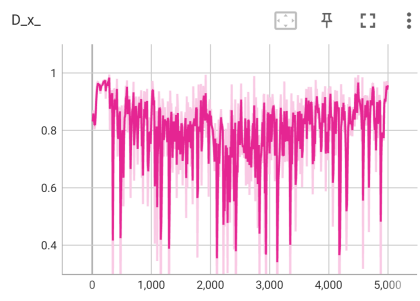
D_G_z1



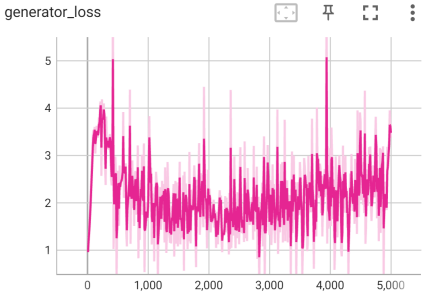
D_G_z2



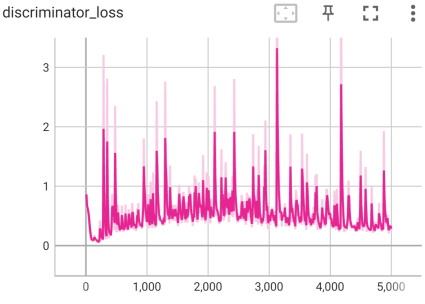
D_x__



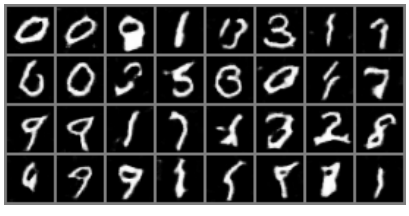
Loss_G



Loss_D

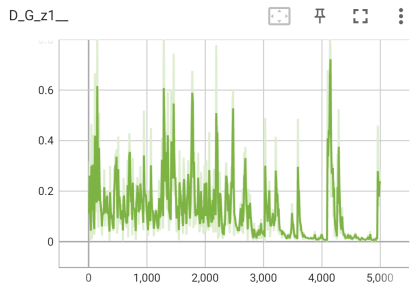


Image@4999

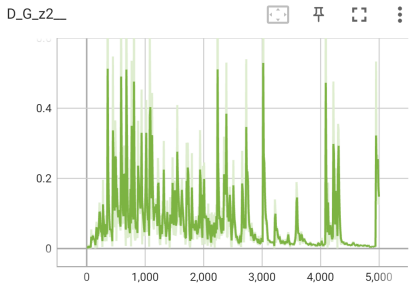


`latent_dim=100`, `hidden_dim=100`

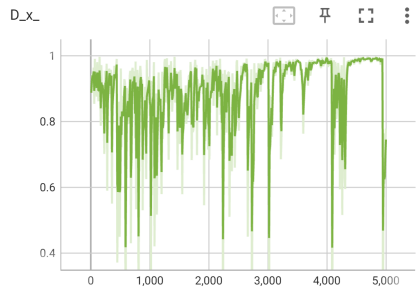
D_G_z1



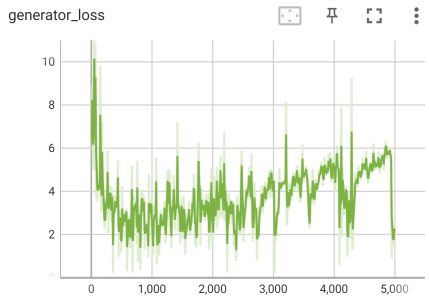
D_G_z2



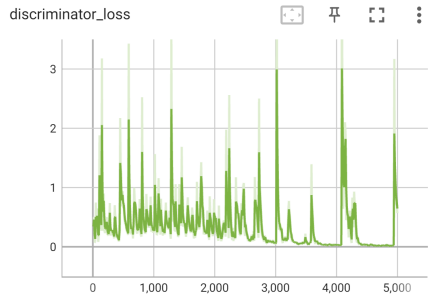
D_x__



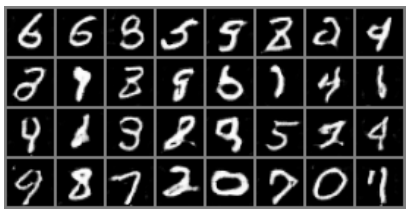
Loss_G



Loss_D



Image@4999



2. FID Score

在这一节，我们给出各个模型的 FID Score。

实验发现，不同随机数种子训练得到的结果差异较大。因此，选取随机数种子 $\text{seed} = \{2022, 2023, 2024, 2025, 2026\}$，也就是每种配置进行了 5 组实验，结果如下。

	hidden_dim = 16	hidden_dim = 32	hidden_dim = 64	hidden_dim = 100
latent_dim = 16	98.8 ± 14.3	67.8 ± 4.7	60.7 ± 7.0	38.5 ± 8.0
latent_dim = 32	75.7 ± 13.6	69.5 ± 13.9	53.7 ± 4.9	37.4 ± **3.8**
latent_dim = 64	83.3 ± 17.5	62.1 ± 11.9	49.6 ± 12.1	**36.8** ± 5.6
latent_dim = 100	82.4 ± 23.9	59.3 ± 9.3	37.0 ± 7.8	37.9 ± 6.6

其中 $(\mu \pm \sigma)$ 表示这种配置下的实验均值为 μ 标准差为 σ 。

各组的最佳实验结果为

	hidden_dim = 16	hidden_dim = 32	hidden_dim = 64	hidden_dim = 100
latent_dim = 16	71.1	61.5	47.8	**26.1**
latent_dim = 32	57.9	53.1	49.4	32.3
latent_dim = 64	59.3	45.5	32.6	31.2
latent_dim = 100	47.0	43.5	27.1	29.0

3. `latent_dim` 和 `hidden_dim` 对模型表现的影响

`hidden_dim`

- 从趋势上说，**模型的性能表型随着 `hidden_dim` 的增加而提升，且对于更大的 `hidden_dim` ，不同随机数训练得到的模型差异更小。**这是因为更大的 `hidden_dim` 使模型具有了更大的参数量， Generator / Discriminator 可以学到更多特征，这提升了模型的表现能力。
- 而 `latent_dim = 100` 时， `hidden_dim` 从 64 提升到 100 并没有带来明显的性能提升。从训练过程来看，此时增加参数量对 Discriminator 的的能力提升要大于 Generator，造成 Discriminator远远强于 Generator 的情形。此时 Generator - Discriminator 相互 "进化" 的平衡也被打破，因此通过增加 `hidden_dim` 的方式可能无法显著提升模型的表现。

`latent_dim`

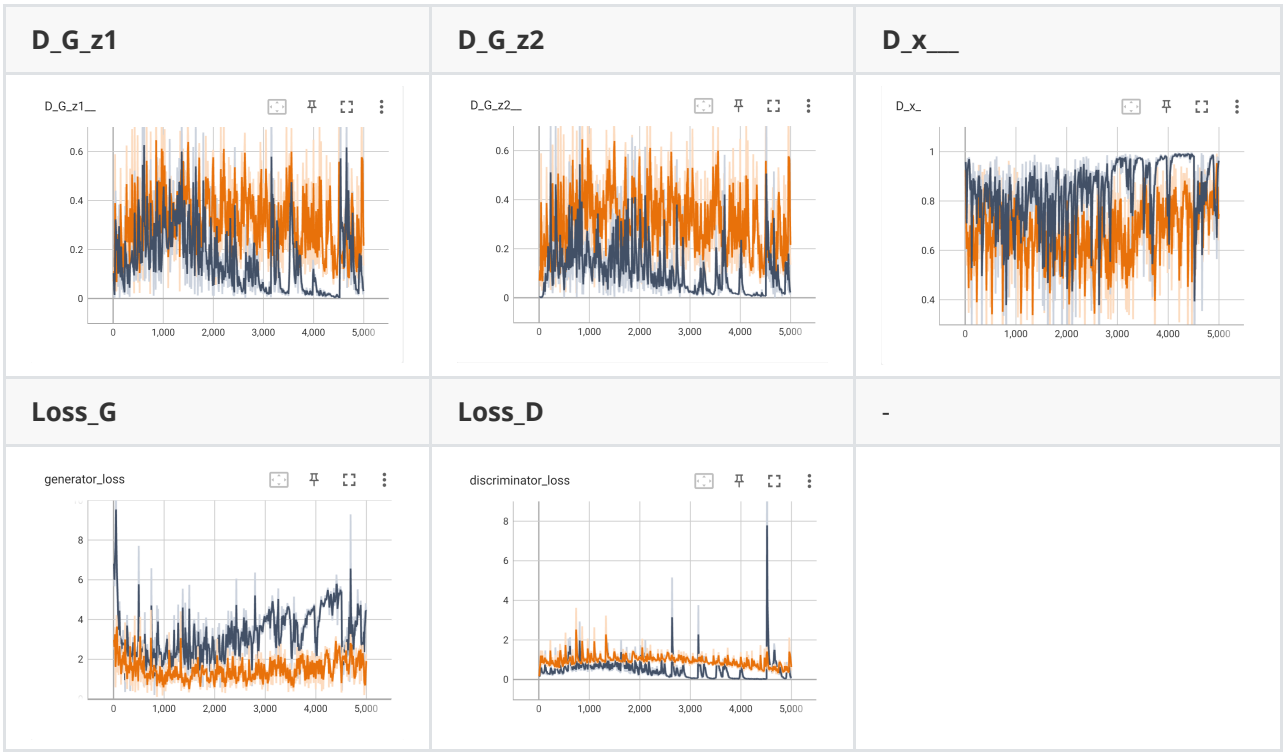
- 就 FID 均值来说，增大 `latent_dim` 会在一定程度上提升模型的表现，但这种提升并不像 `hidden_dim` 那样作用显著。
  - 对于 `hidden_dim=16` ， `hidden_dim=32` 和 `hidden_dim=100` ， `latent_dim` 和 FID 均值 并没有单调关系；
  - 对于和 `hidden_dim=64` ，FID 均值则随着 `latent_dim` 的增大而单调减小。
  - `latent_dim` 较大时，采样到的信号更加复杂，可能有利于模型生成更为多样的结果，但这并不总能降低生成结果和训练集之间的距离（FID）；另外，更大的 `latent_dim` 也对于模型意味着更复杂的输入，因此也可能会更难以学习。
- 就 FID 最优值来说，在参数量较小时，增大 `latent_dim` 可以明显得到更好的结果；而对于较大的 `hidden_dim` ， `latent_dim` 则对最优 FID 影响不大。
  - 这可能是由于对于较小的模型，提升 `latent_dim` 可以显著提升参数量。以 `hidden_dim=16` 来说， `latent_dim` 从 16 提升到 100 使 Generator 参数量从 $57.9\mathrm{k}$ 提升至 $144\mathrm{k}$ ，因此可能让 Generator 在训练过程中有更大概率变得比较强。

4. 模型是否收敛到 Nash 均衡?

在我们的 DCGAN 实现中，模型没有收敛到 Nash 均衡，甚至没有向着 Nash 均衡的方向收敛。

- 从实验的训练曲线来看，Discriminator 给 Generator 生成图像的分数始终保持在较低水平上，在一些情况下（如 `latent_dim=100` 时）甚至收敛到了非常接近 0 的水平；而 Discriminator 给 真实图像的分数则较高，甚至收敛到了 1 附近。这与 Nash 均衡状态下，Discriminator 已经难以分辨 Generator 生成图像 / 真实图像的理想情形差距甚远。
- 造成这种现象的可能原因是，在目前的训练策略之中，Discriminator 能力提升相比 Generator 快很多，Discriminator 总能轻易地分辨出图像的真实性，而 Generator 相对 Discriminator 则太弱。
- 对此，可能的调整策略包括
  - 减少 Discriminator 的训练次数。我们在先前表现最好的一组实验的基础上进行尝试
    - 例如每两个 Batch 训一次 Discriminator，而每个 Batch 都训练 Generator。(图中橙色线为每 2 个 Batch 训一次 Discriminator，灰色线为每个 Batch 都训练 Discriminator)

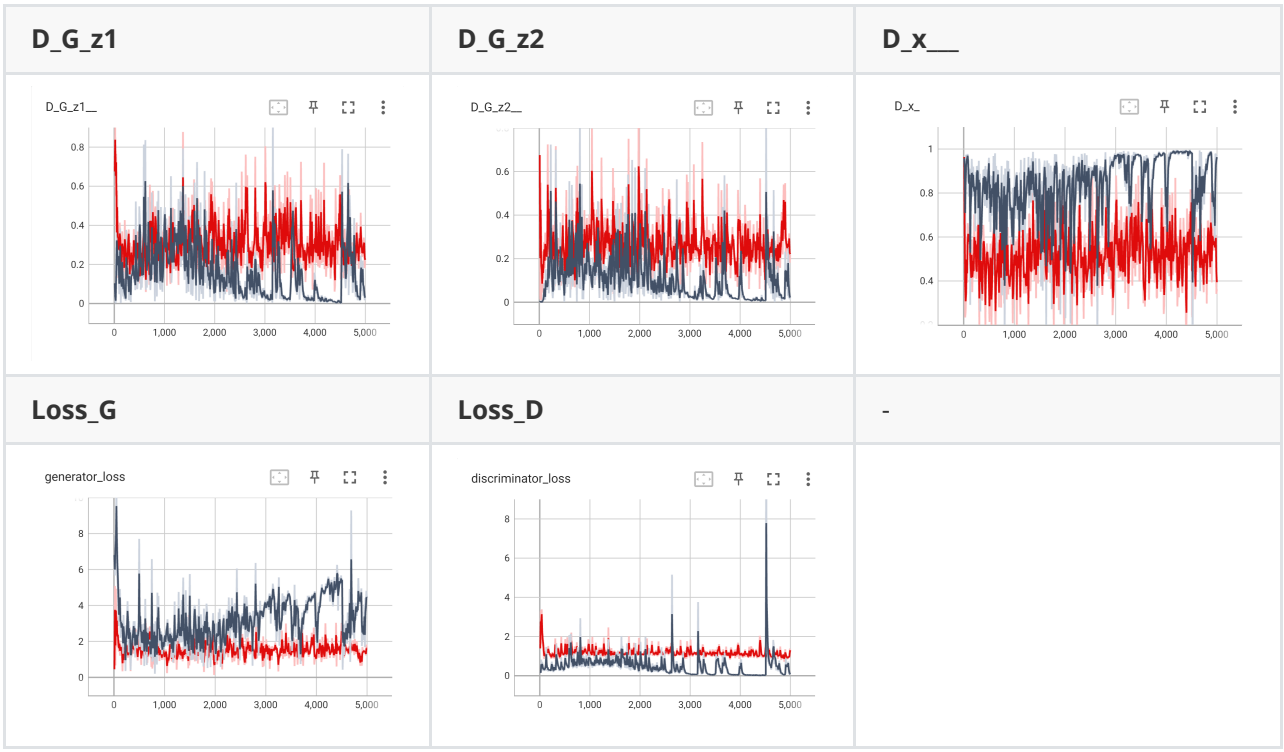
D_G_z1	D_G_z2	D_x___



- 这可以在一定程度上个缓解 D_x → 1 而 D_G_z → 0 的趋势。

- 或者当 $\frac{\text{LossD}}{\text{LossG}}$ 低于某一阈值 λ 时，暂停 Discriminator 的训练，只训 Generator。

  取 $\lambda = 0.8$ 进行实验，(图中红色线为 $\lambda = 0.8$，灰色线是没有使用这一策略的实验)



- 这时收敛的趋势离 Nash 均衡更近一步。可以看到 D_x 已经很接近 0.5 了

- 然而，使用这两种策略都使模型的学习速度显著变慢，此时分数更均衡并不是因为 Generator 变强而是因为 Discriminator 显著变弱。相同的 step 后的 FID 从原先的 26.1 分别增加到 37.1 和 79.1。如果希望使用这种方式提升模型效果，可能还需要进一步调节 num_train_step 等超参。

- 此外，还可以考虑对 Discriminator 和 Generator 使用不同的 learning rate，调整两者测参数量比例等方法来使 Generator 和 Discriminator 的能力更加平衡。

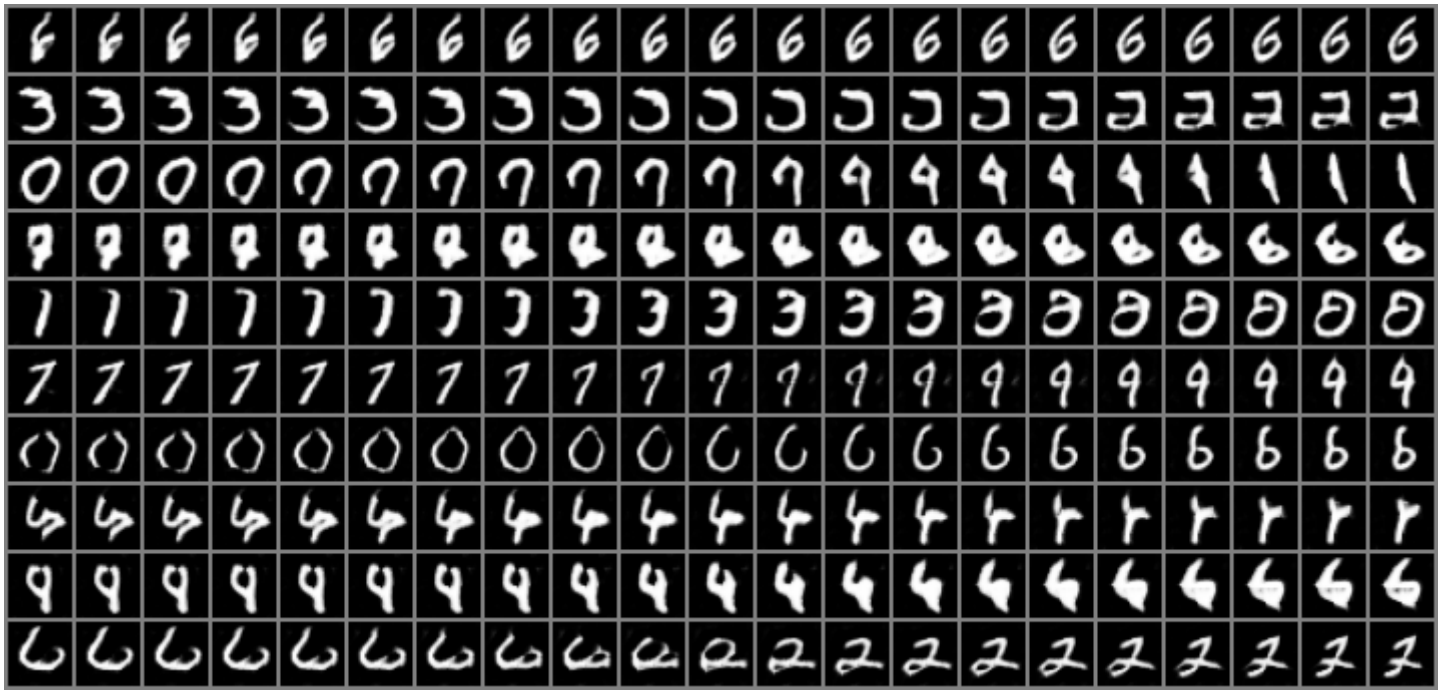
5. 隐空间插值

这一节，我们在隐空间中随机选取两点，对其进行线性插值后使用 Generator 生成图像。

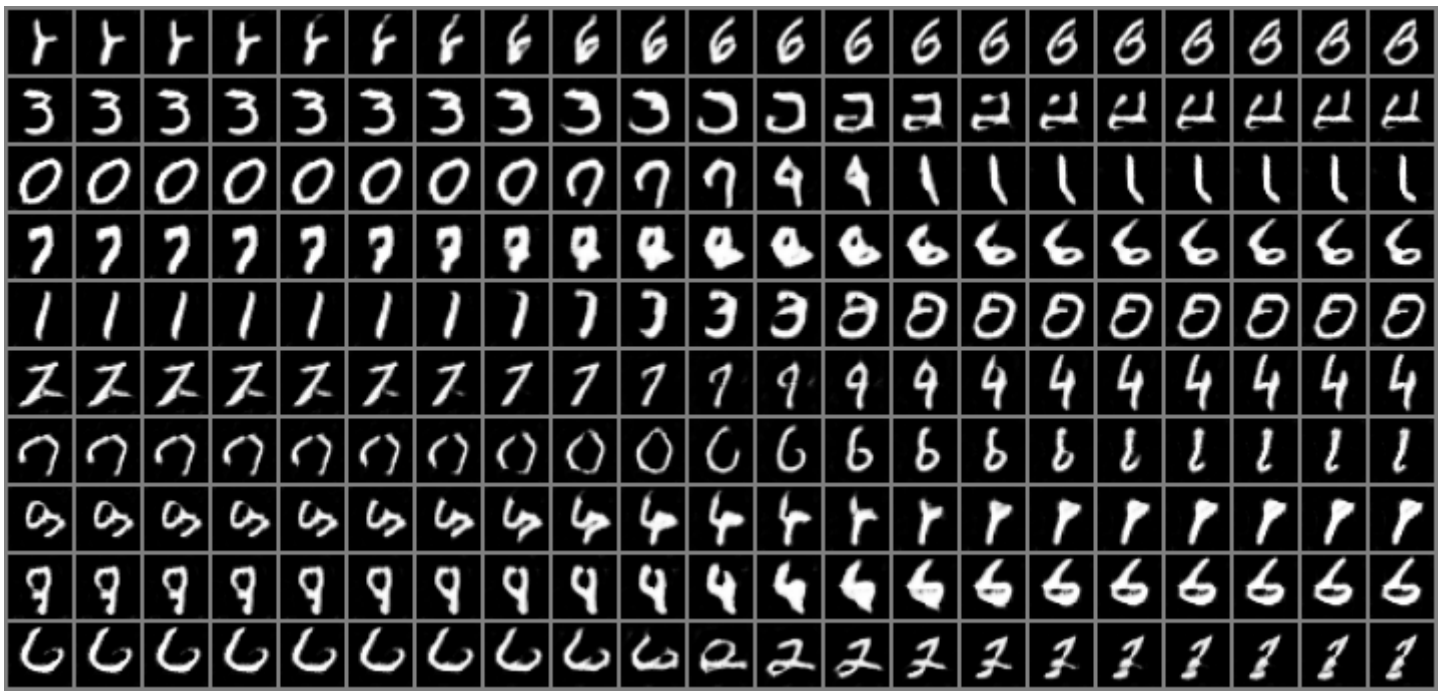
插值的含义约定如下。在区间 $[a, b]$ 上对于样本 z_1, z_2 插值的结果为 $\hat{z}_i = (1 - \lambda_i)z_1 + \lambda_i z_2$，其中 $\lambda_i = \frac{(K-i)a+ib}{K}$ 其中 $i \in \{0, 1, \cdots, K\}$。

取 $K = 20$，每个区间上选取 10 组不同样本 $\{z_1, z_2\}$。对于不同区间，样本的选取保持一致。结果如下

线性内插 [0, 1]



线性外插 [-1, 2]



实验结果表明，Generator 对隐空间的刻画较好，结果变化稳定，许多结果都明显呈现出从一个数字到另一数字的逐步变化。具体来说，线性内插的结果表明 Generator 对隐空间结构有一定程度的掌握，对于不同位置的采样容易产生合理的输出；而线性外插的结果则说明其学到的隐空间分布也能在一定程度上泛化到隐空间之外的样本点。

Bonus

6. 检测 Mode Collapse

从表现最佳的模型 (`z-L16_G100_D100_B64_S5000_SD2026`) 中随机取 100 个样本，结果如下。

人工将生成结果转换为数字（其中有些数字难以辨认，这里选取我认为最接近的数字）

8	3	0	9	1	7	0	5	9	6
6	2	1	6	0	9	6	7	6	2
9	6	6	8	0	7	9	3	6	9
9	6	8	8	6	6	9	8	4	8
6	6	9	4	8	1	8	9	1	8
9	4	3	7	1	8	4	8	8	9
1	5	9	4	3	8	8	4	5	0
6	2	0	1	6	8	2	8	0	9
8	6	4	8	3	7	8	0	4	7
2	0	7	4	8	1	1	6	8	1

统计各个数字出现的次数

数字	0	1	2	3	4	5	6	7	8	9
出现次数	9	9	5	5	9	4	17	7	21	14

- 实验结果表明，我们训练的 GAN 有明显的 mode collapse。例如，对 $6, 8, 9$ 的 生成频率就显著多于训练集 10% 的分布水平，而 $2, 3, 5$ 则显著低于训练集中 10% 的分布。
- 可能的原因是， $6, 8, 9$ 等数字中都存在一个小而闭合的环。可能 Discriminator 将这一点作为判别的标准，并使 Generator 倾向于产生小圆环，这导致生成出来的图像看起来更容易像 $6, 8, 9$ 。而 $2, 3, 5$ 的特征较为复杂，Generator 可能始终没能有效掌握，在 Discriminator 梯度的作用下，也就逐渐倾向于少生成这些数字以降低 Loss。

7. 使用 MLP 实现 GAN

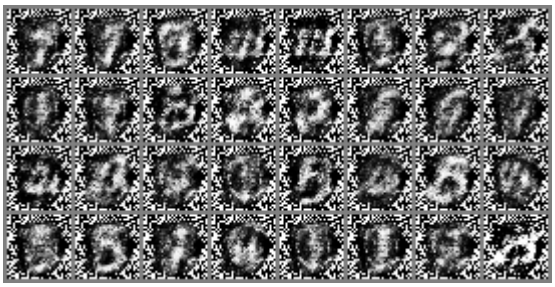
在这一节，我们使用 MLP 来实现 GAN，并与 DCGAN 进行比较。

模型实现

用 MLP 实现 GAN 的过程较为曲折。

一开始，我在 Generator 和 Discriminator 中都使用了 BatchNorm ，然而生成出来的图和 FID Score 表明模型并没有学到东西。（而且更神奇的是周围的的这部分噪声在训练的后半段从未发生任何变化）

Generation Results (MLPDiscriminator with BatchNorm)



后来去掉了 Discriminator 的 BatchNorm，模型大概是学会了某些东西，当然也远不如 DCGAN。

最终采用的模型实现为

```
MLPGenerator(
  (decoder): Sequential(
    (0): Linear(in_features=latent_dim, out_features=128, bias=True)
    (1): BatchNorm1d(128)
    (2): ReLU()
    (3): Linear(in_features=128, out_features=256, bias=True)
    (4): BatchNorm1d(256)
    (5): ReLU()
    (6): Linear(in_features=256, out_features=512, bias=True)
    (7): BatchNorm1d(512)
    (8): ReLU()
    (9): Linear(in_features=512, out_features=1024, bias=True)
    (10): Tanh()
  )
)
MLPDiscriminator(
  (encoder): Sequential(
    (0): Linear(in_features=1024, out_features=512, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): Linear(in_features=512, out_features=256, bias=True)
    (3): LeakyReLU(negative_slope=0.2)
    (4): Linear(in_features=256, out_features=128, bias=True)
    (5): LeakyReLU(negative_slope=0.2)
    (6): Linear(in_features=128, out_features=1, bias=True)
    (7): Sigmoid()
  )
)
```

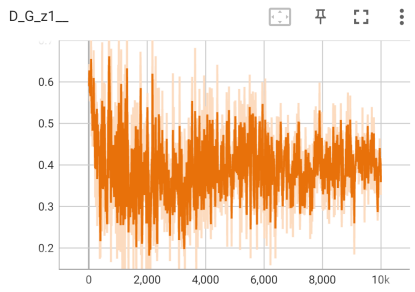
采用的参数为

```
batch_size=64
num_train_steps=10000
```

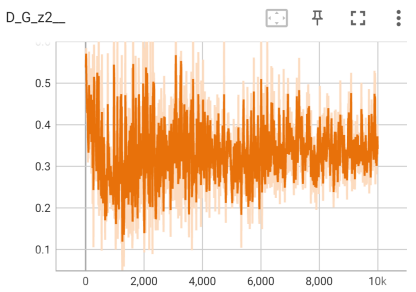
训练过程

这里选取 `latent_dim=100, seed=2022` 这一组实验的训练过程。

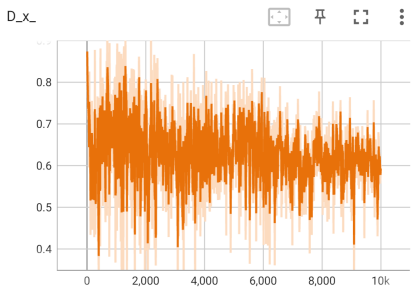
D_G_z1



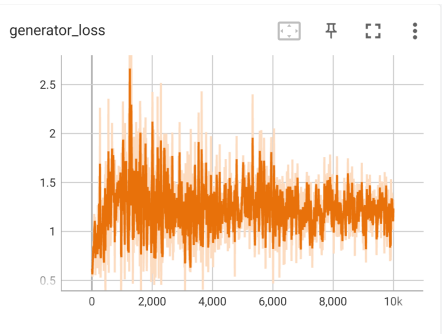
D_G_z2



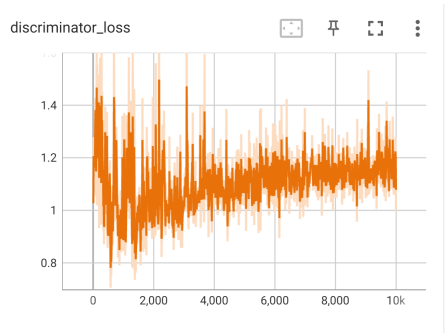
D_x__



Loss_G



Loss_D



Image@9999

- 从训练过程来看，MLP 的 Discriminator 分数和 Loss 并不像 DCGAN 这样极端，分数的收敛趋势更倾向于 Nash 均衡。
- 值得注意的是，MLP 的 Discriminator Loss 在后期不断上升，这可能是由于 MLP 相比 CNN 对图像的特征提取能力较弱。

FID Score

这里同样使用选取随机数种子 $\text{seed} = \{2022, 2023, 2024, 2025, 2026\}$，每种配置进行了 5 组实验，结果如下。Min FID 表示每种配置下 5 次实验的最好结果

latent_dim	FID	Min FID
latent_dim = 16	**112.5** ± 3.5	106.8
latent_dim = 32	113.4 ± 2.4	108.8
latent_dim = 64	114.1 ± **2.3**	111.9
latent_dim = 100	114.4 ± 4.9	**106.2**

$(\mu \pm \sigma)$ 表示实验均值为 μ 标准差为 σ 。

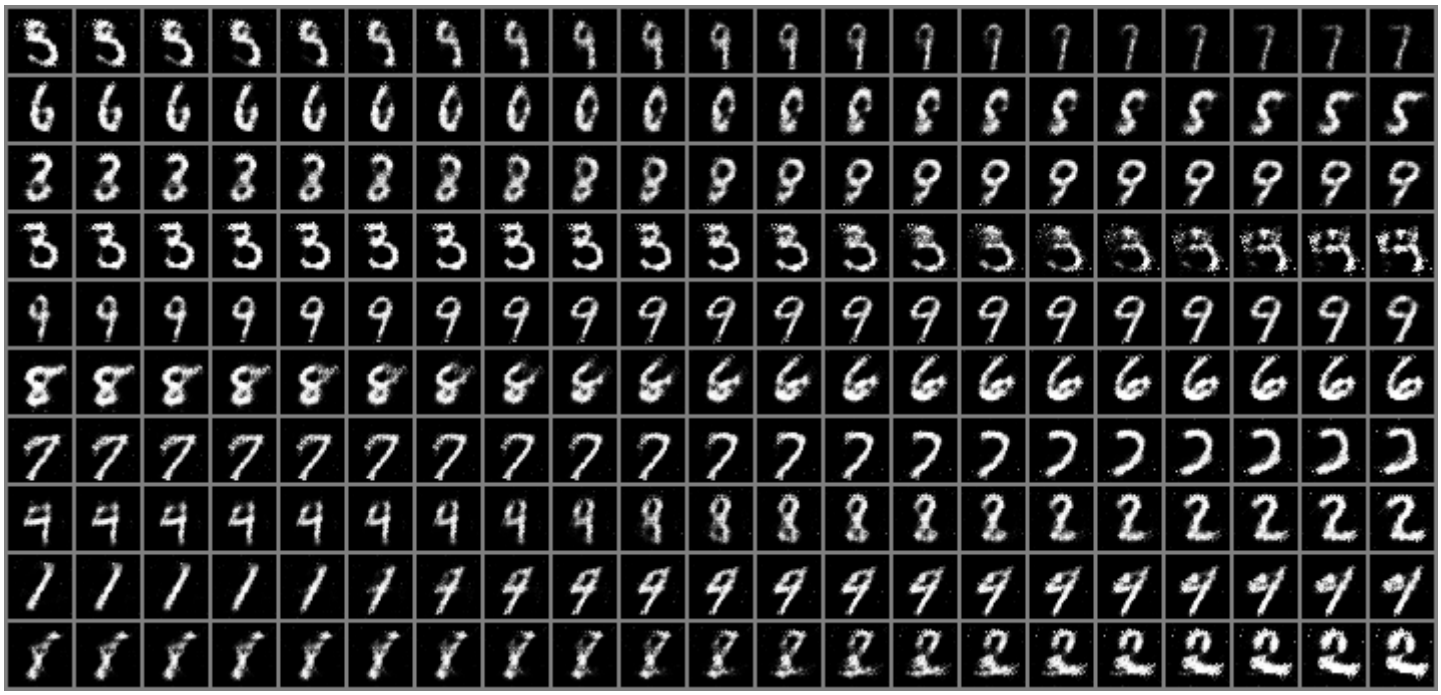
- MLP GAN 的生成能力显著弱于 DCGAN。从生成结果的样本来看，MLP 生成的结果中含有大量噪声，而且数字本身也较为模糊。
- 这可能是因为，MLP 将图像拉直为一维向量进行处理，使得生成 / 鉴别时图像在空间上缺乏连续性。

- MLPGenerator 生成的两个相邻像素点是由两次向量内积独立生成，没有利用空间信息；而 CNNGenerator 进行 ConvTranspose 时，相邻输出的采样来源于相邻的输入，因此在空间上更加连续，这就避免了像素尺度的噪声等问题。
- MLPDiscriminator 在处理图像时同样没有利用相邻像素之间的空间关系，因此也更难提取噪声等空间上的高频变化特征。这也限制了 Generator 的生成能力。

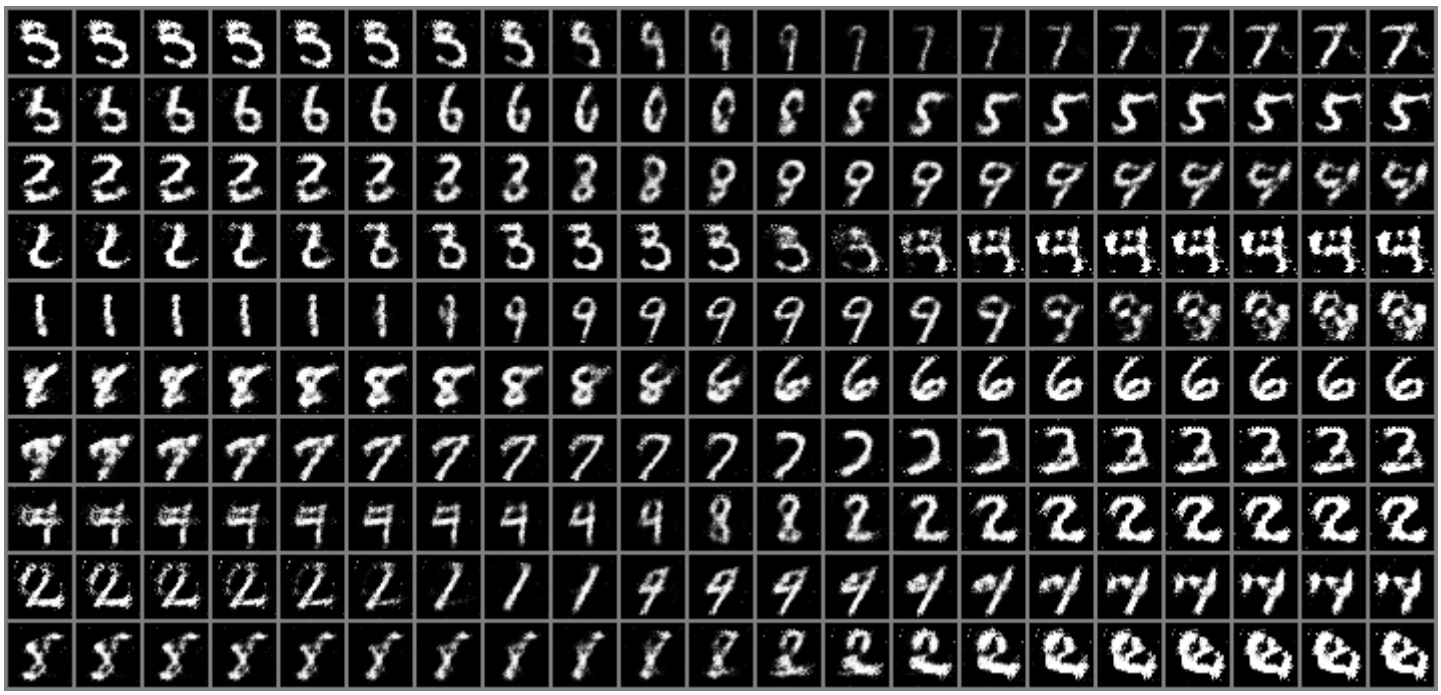
隐空间插值

我们对 MLP 也进行类似的线性插值，结果如下。

线性内插 [0, 1]

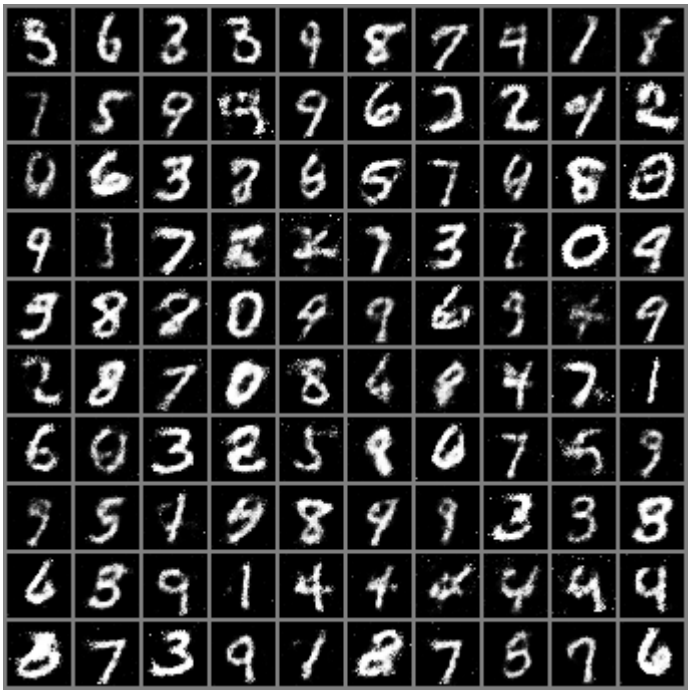


线性外插 [-1, 2]



- MLP 对隐空间的理解也展现出较高的连续性。
- **相比 DCGAN 来说，MLP GAN 插值的过程中出现了更多在像素尺度上 "破碎" 的情况，而 DCGAN 的变化则大概是在笔画尺度上的。**这一现象也可以归因于 MLP 在训练时没有利用图像的空间连续性。

Mode Collapse



我们同样统计 MLP Generator 随机生成的 100 个 Sample，统计结果如下

数字	0	1	2	3	4	5	6	7	8	9
出现次数	7	8	7	10	12	7	9	13	13	14

相比 DCGAN，MLP 生成数字的频率分布于训练集更为接近。这大概也是由于 MLP 对空间信息的提取较弱，因此也不容易认为某个空间特征特别属于真实图像样本。