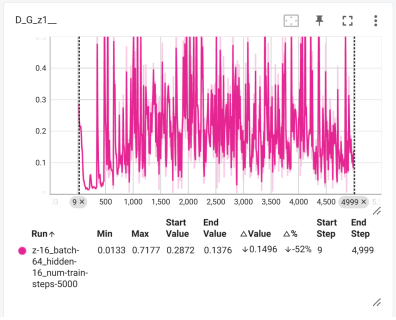
HW4 Report

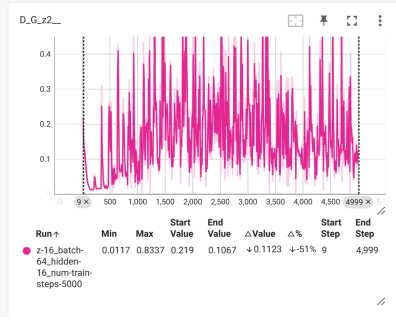
计11班 周韧平 2021010699

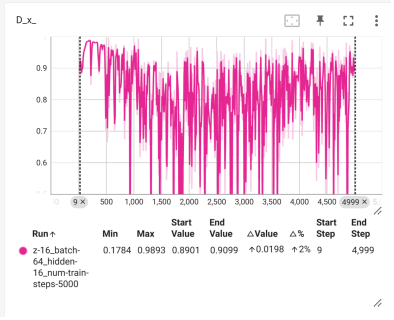
latent_dim 和 hidden_dim 在 16，100下的结果

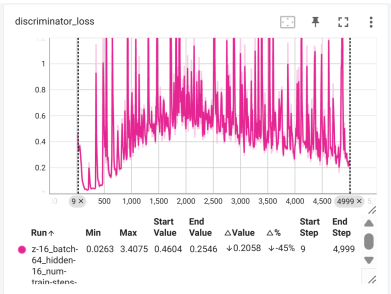
本节中，batch_size 和 num_train_steps 均为默认值 64，5000

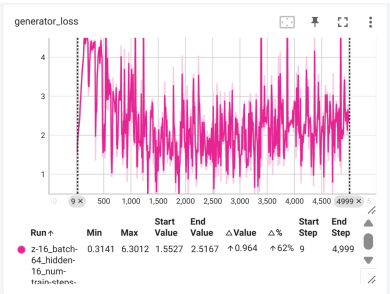
latent_dim=16, hidden_dim=16

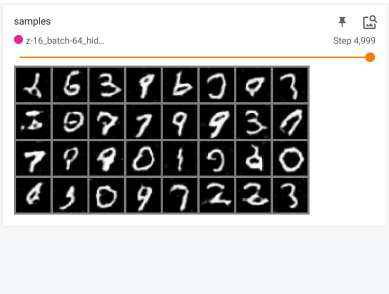




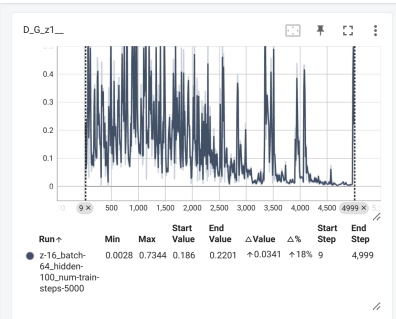


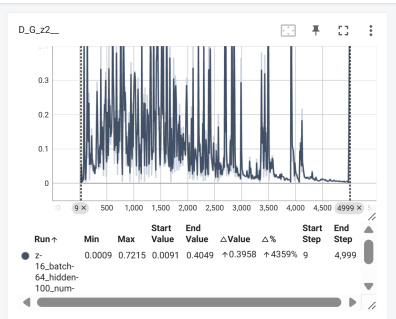


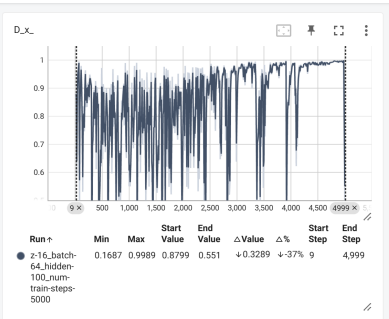


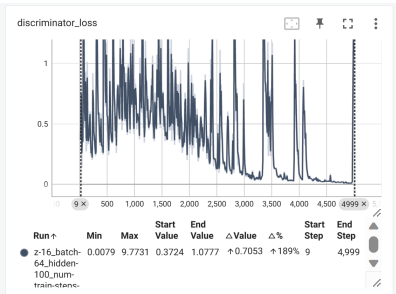


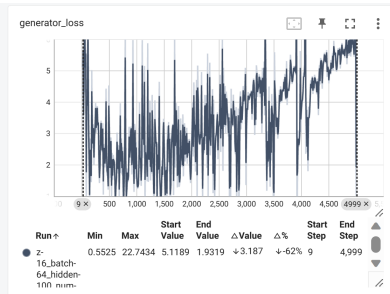
latent_dim=16, hidden_dim=100



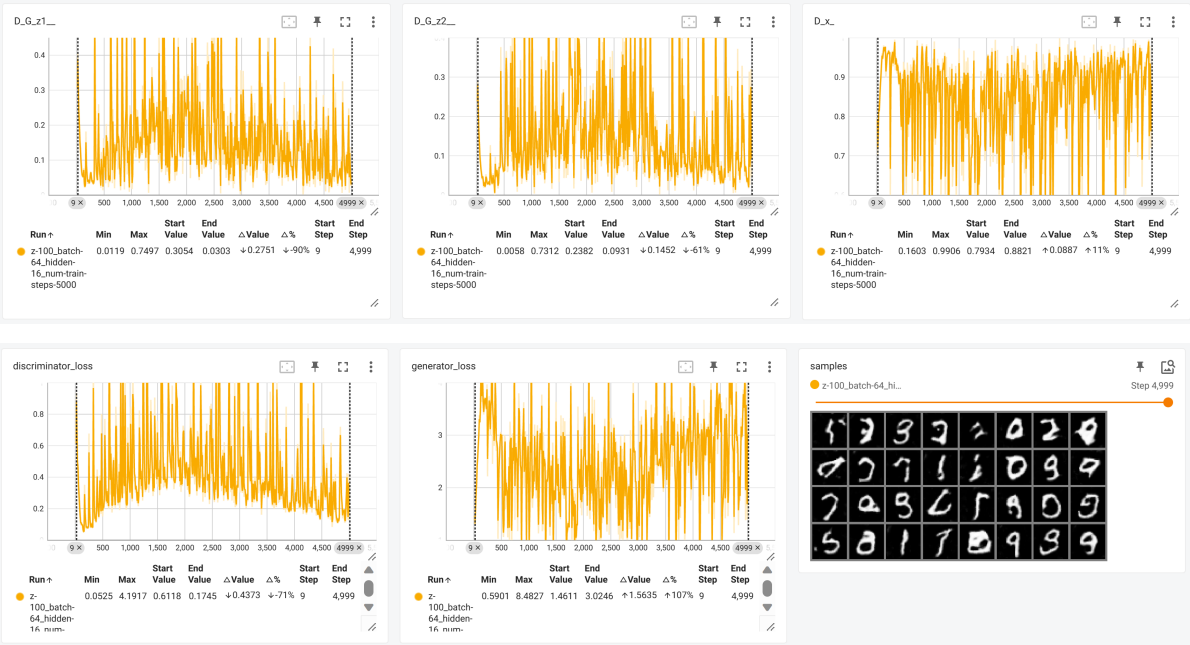




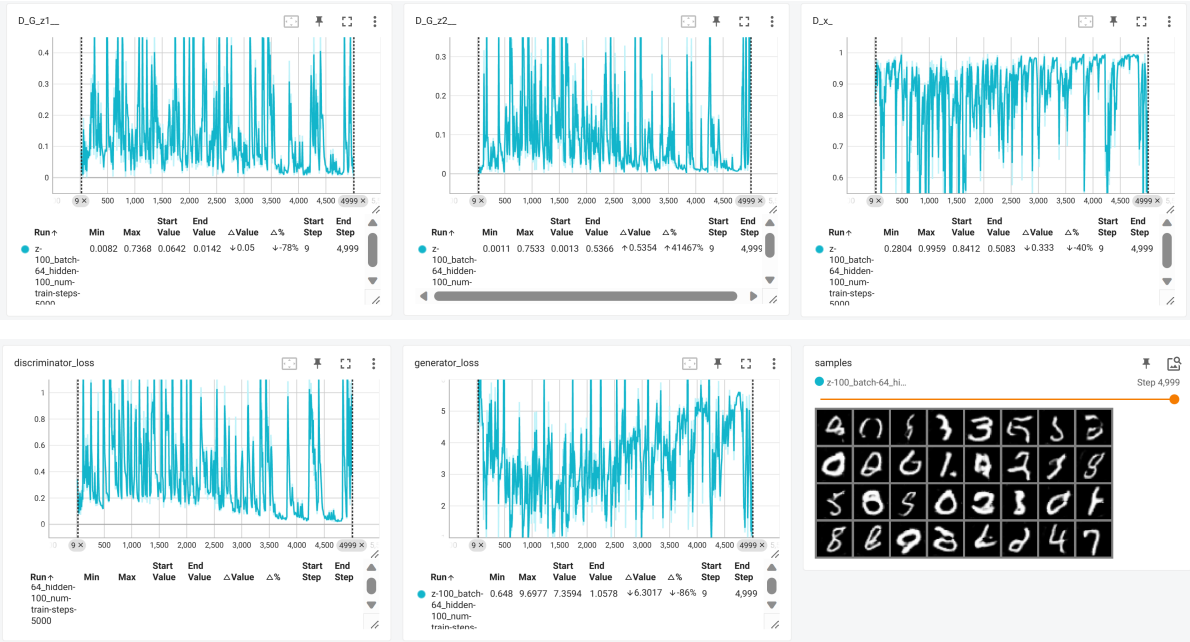




latent_dim=100, hidden_dim=16



latent_dim=100, hidden_dim=100



汇报FID结果

上述四个模型的FID结果为

	hidden_dim=16	hidden_dim=100
latent_dim=16	93.82	54.46
latent_dim=100	69.91	34.67

讨论 hidden_dim 和 latent_dim 对 GAN 模型的影响

实验结果

在其它超参数和第一节保持一致的情况下，进一步调整 hidden_dim 和 latent_dim 得到如下结果

	hd=16	**hd=32**	**hd=50**	**hd=64**	**hd=75**	**hd=100**
ld=16	93.82	83.71	81.72	88.35	79.45	**54.46**
ld=100	69.91	39.16	44.88	40.04	**30.36**	34.67

	ld=16	**ld=32**	**ld=50**	**ld=64**	**ld=75**	**ld=100**
hd=16	93.82	63.70	72.99	**45.38**	61.86	69.91
hd=100	54.46	46.23	40.46	33.35	**32.61**	34.67

实验分析

hidden_dim

- 对 `hidden_dim` 来说，**整体来看，模型表现会随着 `hidden_dim` 的增加而提升**，在 `ld=16` 和 `ld=100` 两组设置下，模型分别在 `hidden_dim=100` 和 `hidden_dim=75` 时效果最佳。这是因为 `hidden_dim` 的增加使得模型的参数量提升，这让无论是Generator还是Discriminator都能学到更多的特征，也就提升了模型的表现能力
- 对比 `ld=16` 和 `ld=100` 和两组设定，可以发现，当 `ld` 较大时，`hidden_dim` 对模型效果的提升不如 `ld` 较小时明显，通过观察 `ld=100` 时二者的loss曲线可以发现，Discriminator 的loss曲线稳定下降，而 Generator 的loss函数曲线则在一开始下降后迅速上升，**这说明 Discriminator 的学习速度快于 Generator**，因此此时增加 `hidden_dim` 尽管还可以提升模型表现能力，但已经打破了二者相互学习的平衡，也就不能实现整体模型性能的提升

latent_dim

- 总体来说，**在 `latent_dim` 介于 16 到 64 之间时，提高 `latent_dim` 对于模型性能提升有所帮助**，这是由于 `latent_dim` 的增加会提高输入模型的随机噪声的维度，使得 Generator 能够依赖更复杂的信号去生成图片，这样设计有利于提升生成图片整体的准确率和多样性，正如上一组实验所观察的结果，generator的学习能力不足是限制模型整体能力的因素，因此提升 `latent_dim` 可以有效提高 generator 的学习能力，进而提高模型整体的性能
- **更大的 `latent_dim` （>64）对于模型性能提升效果不明显**，甚至在 `hd=16` 时会起到相反的作用，这可能是因为在 `hd` 本身不大的情况下，Generator的学习能力不足，过高的 `latent_dim` 并不能让模型生成更好的结果。Generator 的网络结构中，第一层为输入输出通道分别为 (`latent_dim`, `4*hidden_dim`) 的反卷积网络，当 $latent_{dim} > 4 \times hidden_{dim}$ 时，第一层网络反而会使得通道数的降低，这在一定程度上有悖于 Generator 的设计思路。

推导判别器最优解，并判断是否达到纳什均衡

对 Discriminator 来说，训练即最大化

$$P_{data}(x)logD(x) + P_G(x)log(1 - D(x))$$

其中 $P_{data}(x), P_G(x), D(x)$ 分别表示真实图像的分布，Generator生成图像的分布和 Discriminator 判别为真的结果，令 $P_{data}(x) = a, P_G(x) = b$ 对D求导可得

$$f(D) = alog(D) + blog(D)$$
$$\frac{f(D)}{dD} = 0 \to D^* = \frac{a}{a+b}$$
$$i.e. D^*(x) = \frac{P_{data}(x)}{P_{data}(x) + P_G(x)}$$

将D的优化结果代入 Min-Max 游戏的 Value Function 中去求G的优化

$$min_G max_D V(G, D) = min_G V(G, D^*)$$

$$\begin{aligned} V &= E_{x \sim P_{data}}[logD(x)] + E_{x \sim P_G}[log(1 - D(x))] \\ &= -2log2 + KL(P_{data}(x) || \frac{P_{data}(x) + P_G(x)}{2}) + KL(P_G(x) || \frac{P_{data}(x) + P_G(x)}{2}) \\ &= -2log2 + 2JS(P_{data}(x) || P_G(x)) \end{aligned}$$

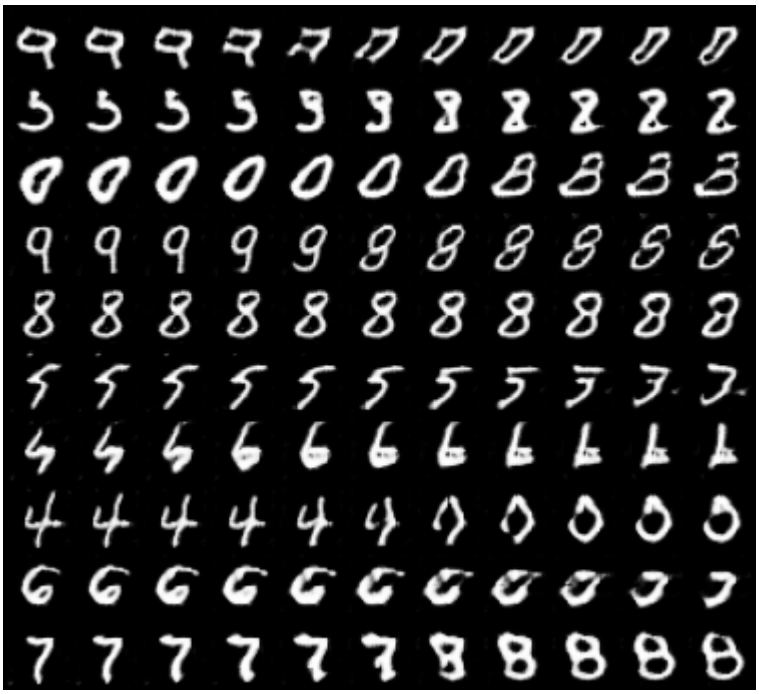
当 $P_{data}(x) = P_G(x)$ 时， $V(G, D^*)$ 取到最小值，因此对于 Discriminator 来说，**纳什均衡点时应该为对于一张图片将其预测为真实和虚假的概率相等，难以分辨真假**

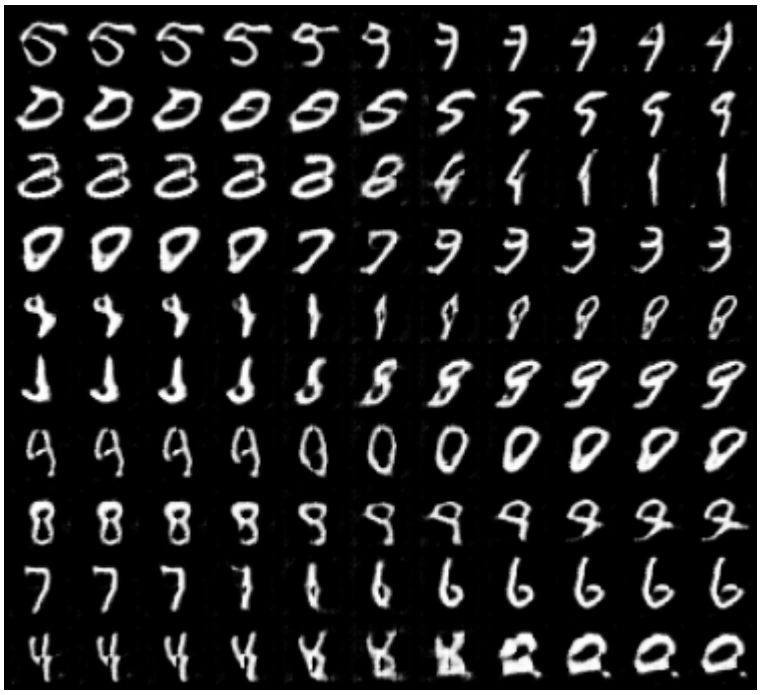
然而在前面的实验中，**模型并未达到纳什均衡**，Discriminator 对于真实图像判定得分过高（`D_x`>0.9）而对于虚假图像判定得分过低（`D_G`<0.1），也就是说它能很准确的判别出图片真假，并没有达到纳什均衡的状态。

观察线性插值结果

本节中我选取了FID值最低的 ld=75，hd=100，作为观测线性插值效果的模型

下图分别为scale在(0,1)之间内插的结果和(-1,2)进行一定外插的的结果，从左到右，scale逐渐增大





内插结果显示，一些行的数字都经历了一个连续的转变，如第二行从3到8的变化，第三行从0到8的变化，第四行从9到8的变化，最后一行从7到8的变化等，这表明模型能够生成中间状态，这些状态在视觉上与两端的数字是连贯的。同时，大部分实例又能保留其基本的数字特征，这表明模型在保持数字可识别性的同时也具有一定的多样性。然而，在其他行中，如第九行从“6”到“7”的过渡，中间的数字有时看起来并不像任何标准数字，这可能表明对于某些插值路径，模型可能不如其他路径那么有效。

外插结果显示，模型对于某些数字的外插效果，如第四行从0到3的变化，倒是第二行从1到6的变化等，模型显示出了较好的性能，特别是倒数第二行中，模型外插生成了更接近于“7”的生成效果图，这表明模型在创造新样式上也具有一定的潜力，也在一定程度上说明模型具有较好的泛化性

BONUS

Mode Collapse

下图为使用 ld=75，hd=100参数生成的100个采样结果

为其手动打上标签，结果如下

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

其中“-”表示难以辨别，各个数字出现的次数为

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
12	8	5	11	7	8	5	10	11	12

- 实验结果表明，模型具有一定的 mode collapse，在可辨别的数字中，0，3，7，8，9占比超过60%，而数字2和6则只有5%的出现频率。推测可能的原因，可能存在训练数据不平衡（2，6数据量较少等），模型容量不足导致无法捕捉到数据多样性，导致偏好生成最容易学习的模型等问题，另外，还可以从生成数据的难度上解释，观察发现模型十分倾向于生成一个圆，而数字2、4等数字却又恰好缺乏这种数字特征，可能也是导致其出现频率小于其它数字的原因

Ablation Study

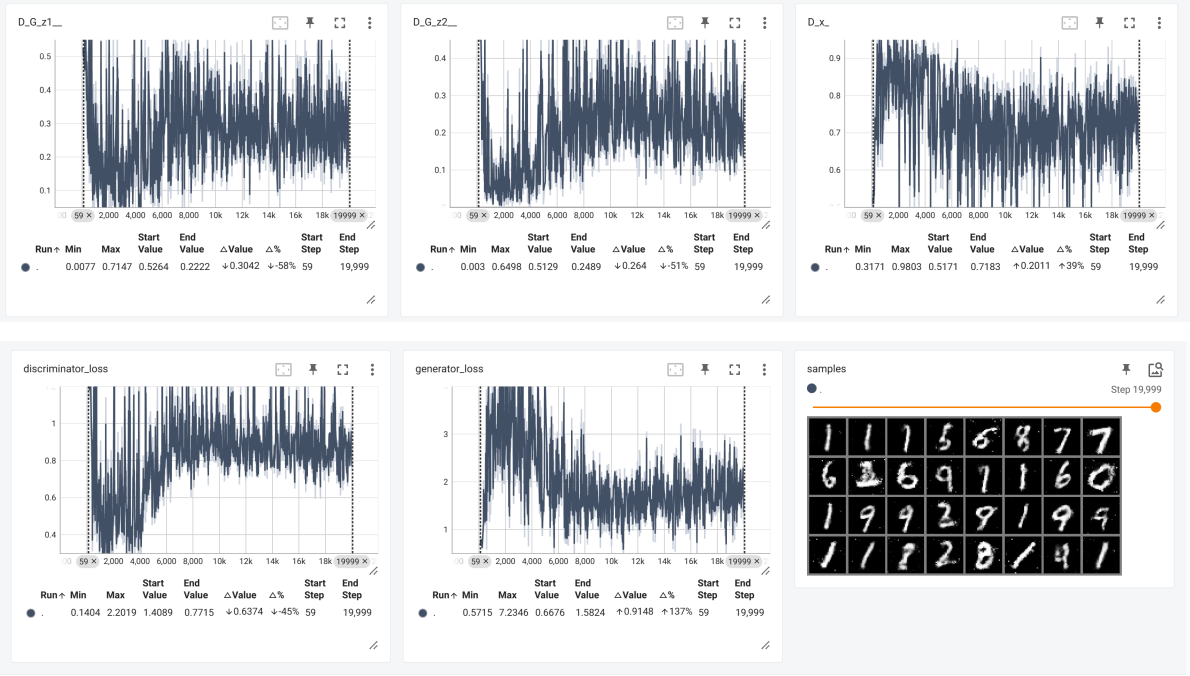
实现MLPGAN

通过多组实验，最后选择的最佳MLPGAN的网络结构实现为

```
Generator{
    Linear(latent_dim, hidden_dim * 8),
    ReLU(),
    Linear(hidden_dim * 8, hidden_dim * 4),
    ReLU(),
    Linear(hidden_dim * 4, hidden_dim * 2),
    ReLU(),
    Linear(hidden_dim * 2, 32 * 32 * num_channels),
    nn.Tanh()
}
Discriminator{
    Linear(32 * 32 * num_channels, hidden_dim * 4),
    LeakyReLU(0.2,inplace=True),
    Linear(hidden_dim * 4, hidden_dim * 2),
    LeakyReLU(0.2,inplace=True),
```

```
    Linear(hidden_dim * 2, hidden_dim),
    LeakyReLU(0.2,inplace=True),
    Linear(hidden_dim, 1),
    Sigmoid()
}
```

在 mlp 实验中，hidden_dim = latent_dim = 100，num_training_steps = 20000，其余超参数设置与 DCGAN 相同，**最终FID结果为128.28**

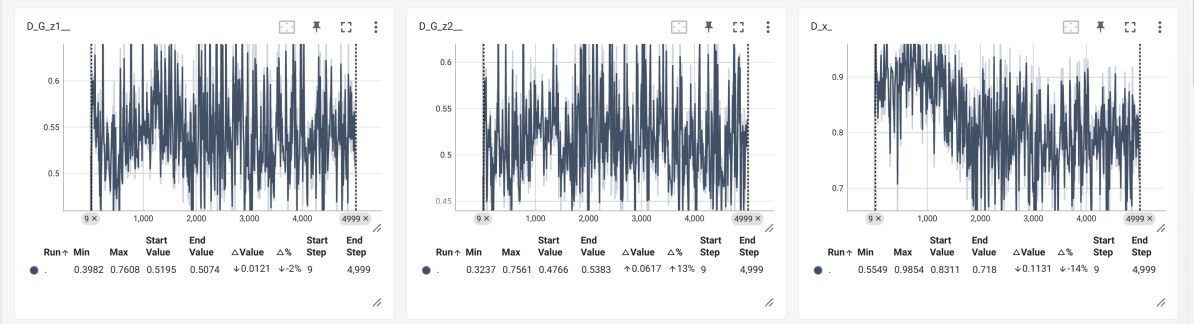


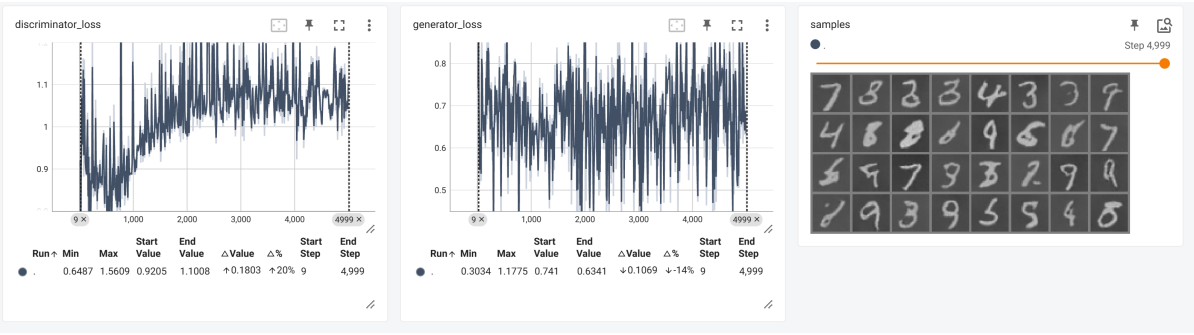
- 基于训练过程的分析显示，MLP 的 Discriminator 分数更接近于0.5，更接近于Nash均衡
- MLP 和 DCGAN 相比需要更多的迭代步数才能让训练结果收敛，且最后的FID结果远不如后者，从samples结果来看，在训练迭代步数不够时，生成的图像周围会存在白色的“噪点”
- 从采样的结果来看，基于MLP模型生成的采样结果存在大量数字“1”，这说明模型存在一定的Mode Collapse

综上， MLP 模型结构并不是该任务的最好选择。

去除 Batchnorm Layer 后的训练效果

下图 hidden_dim = latent_dim = 100 下训练 5000 步的结果，**最终FID值为 49.50**





- 和MLP一样，去除 Normalization 后 Discriminator 的得分更均衡，但其生成效果远不如加入Normalization层，从loss曲线来看，Discriminator过早出现了过拟合的现象，这可能导致Discriminator并不能随着generator能力得提升而提升，也就影响了模型整体的性能。其次，从图片生成效果来看，生成图片在灰度上存在明显差异，这可以解释为由于取出了Normallization层后模型学习到了一些和任务无关的特征（灰度）。这说明，**加入Normallization层后，模型可以更多的关注于那些和任务本身有关的特征，进而提升模型整体的性能**，这也是去除 Normalization 层后FID值下降的原因