

Image Generation with GAN

沙之洲 2020012408

1 Different Latent dimension and Hidden dimension

本次实验的 latent dimension 和 hidden dimension 均从 [16, 64, 100] 中选择。总共进行了 $3 * 3 = 9$ 组实验。这里，为了节省时间，generator hidden dimension 和 discriminator hidden dimension 同时改变。

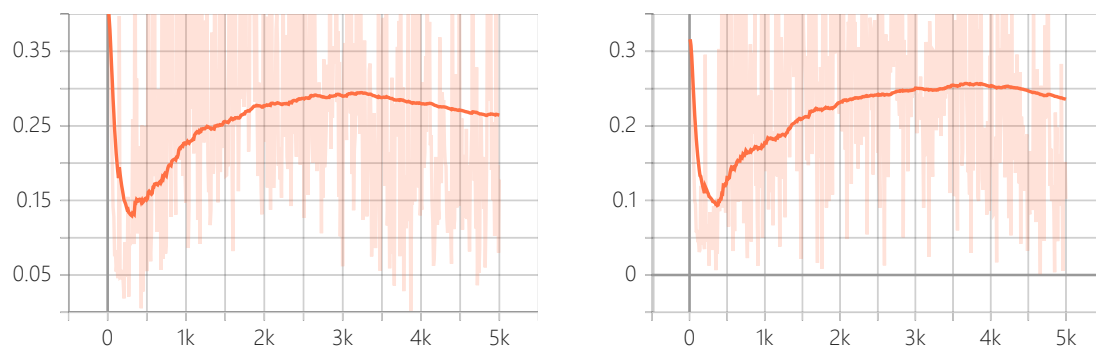
其余的参数均为实验默认参数。也即 batch size 为 64，training step 为 5000。

下图中的 D_x 是 discriminator 判断 real image 为真实图片的概率。

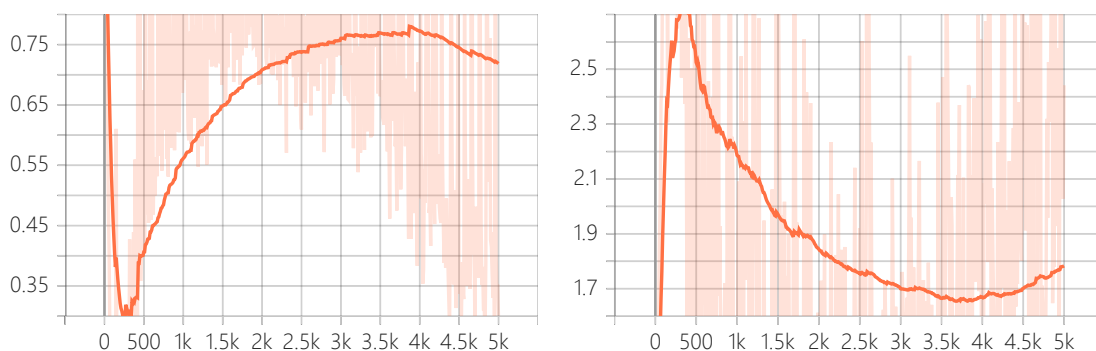
D_G_z1 是 discriminator 参数更新之前判断 generator 产生的 image 不是真实图片的概率，D_G_z2 是 discriminator 在参数更新之后判断 generator 产生的 image 不是真实图片的概率。

latent: 16 / hidden: 16

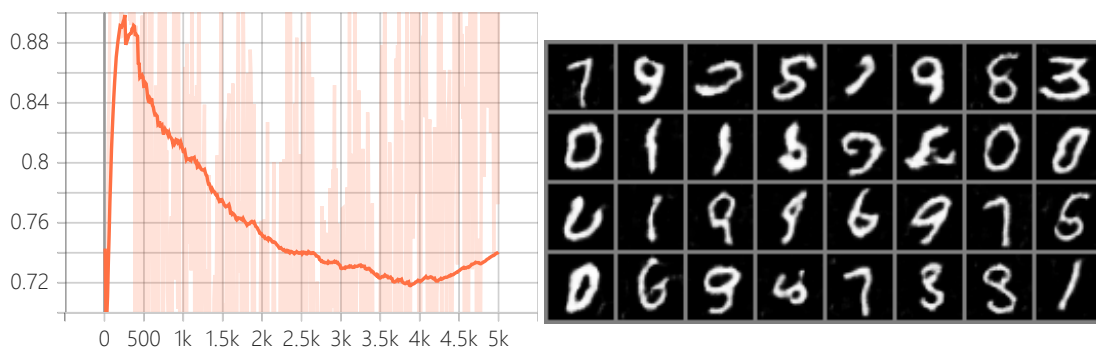
D_G_z1 & D_G_z2



discriminator loss & generator loss

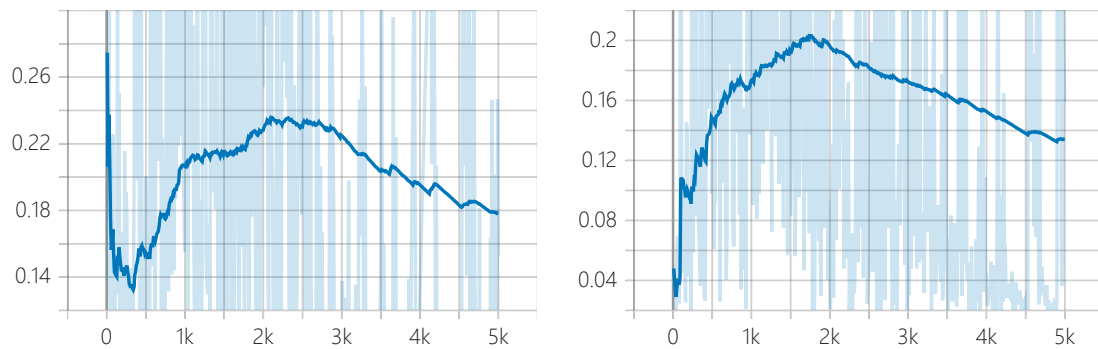


D_x & Final image

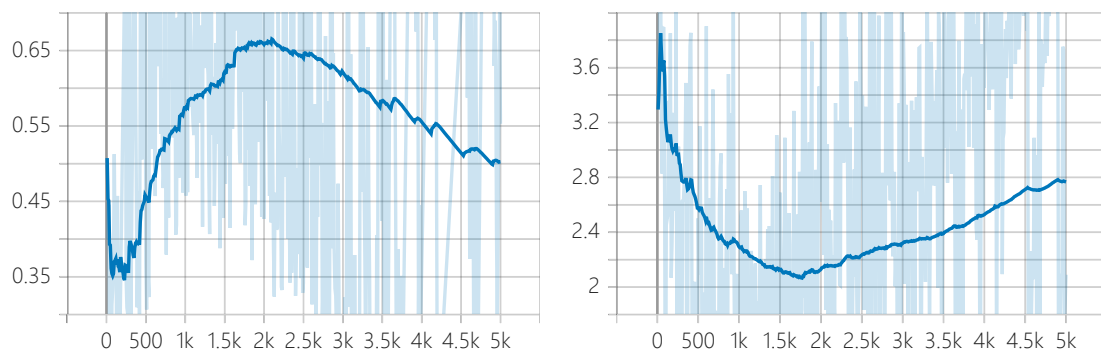


latent: 16 / hidden: 64

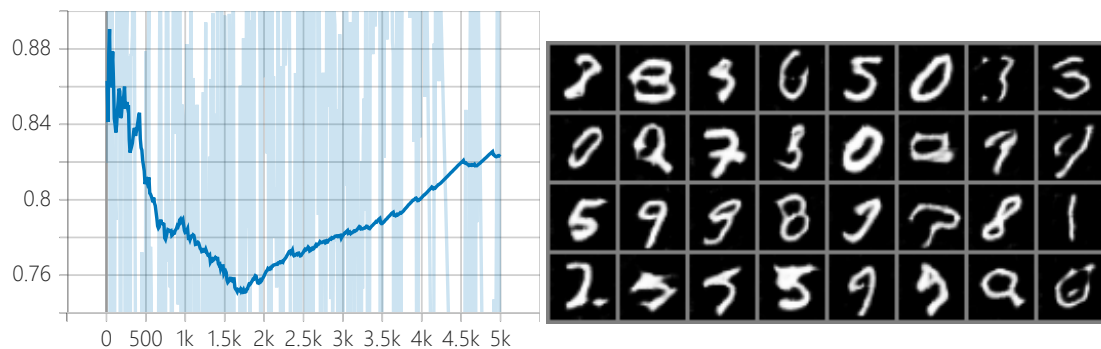
D_G_z1 & D_G_z2



discriminator loss & generator loss

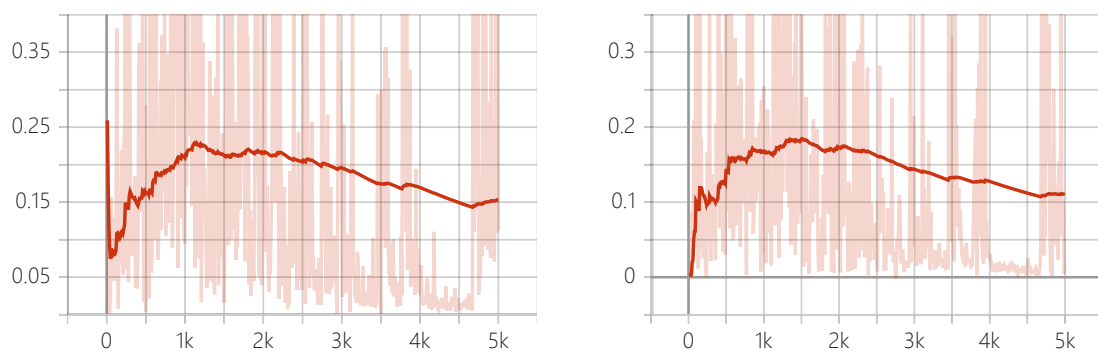


D_x & Final image

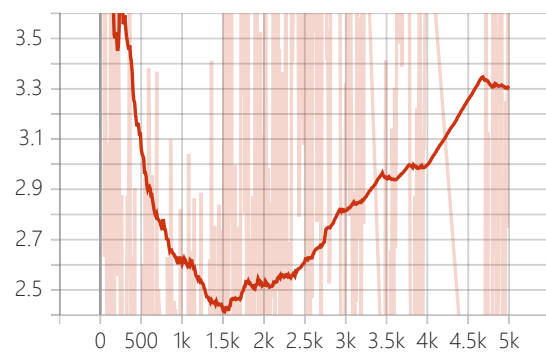
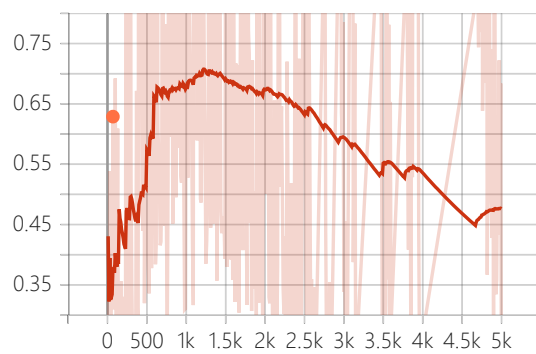


latent: 16 / hidden: 100

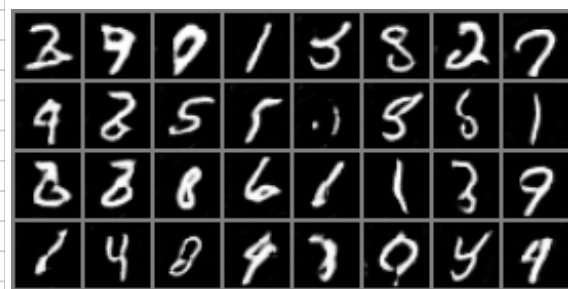
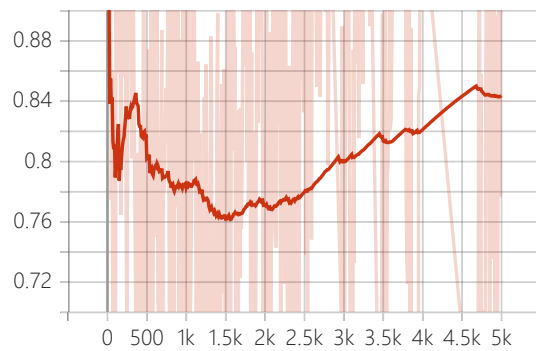
D_G_z1 & D_G_z2



discriminator loss & generator loss

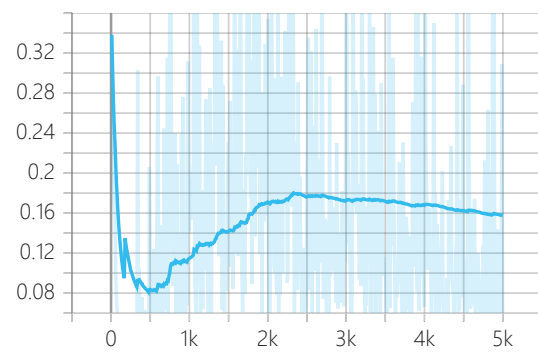
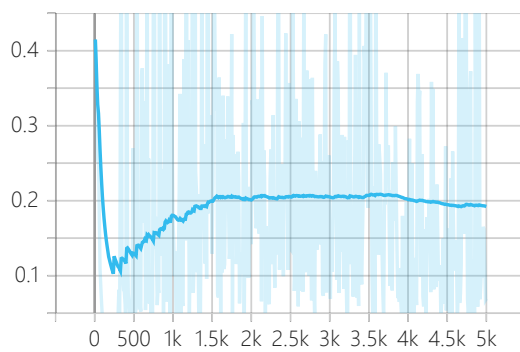


D_x & Final image

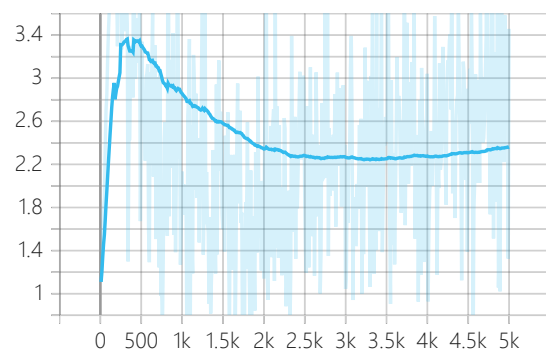
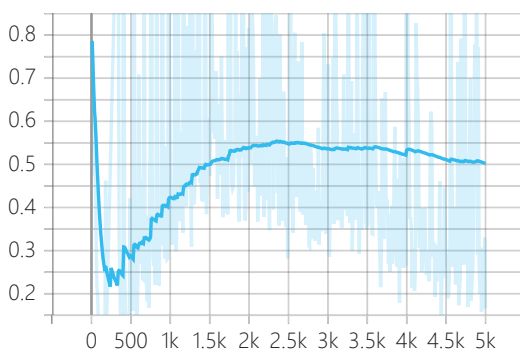


latent: 64 / hidden: 16

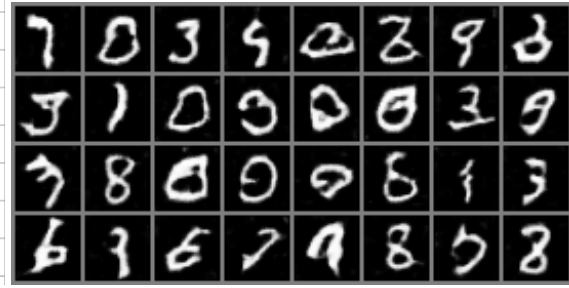
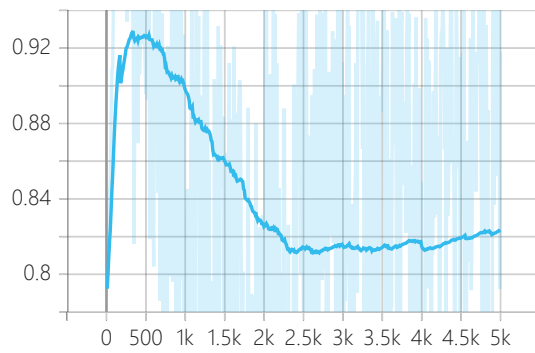
D_G_z1 & D_G_z2



discriminator loss & generator loss

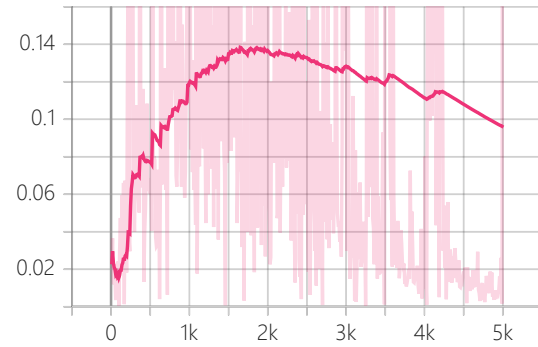
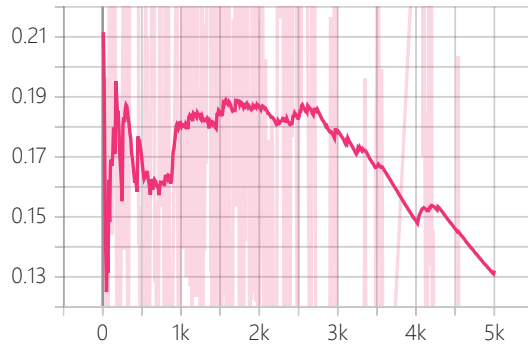


D_x & Final image

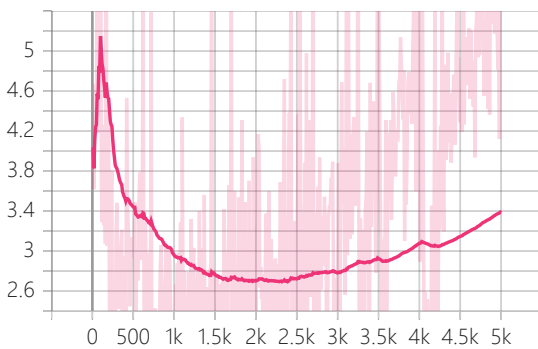
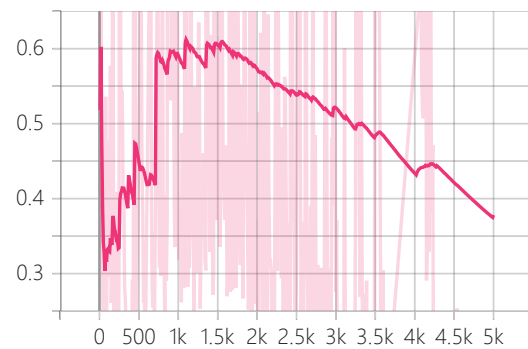


latent: 64 / hidden: 64

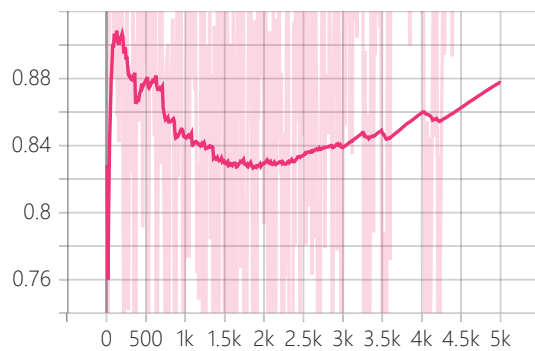
D_G_z1 & D_G_z2



discriminator loss & generator loss

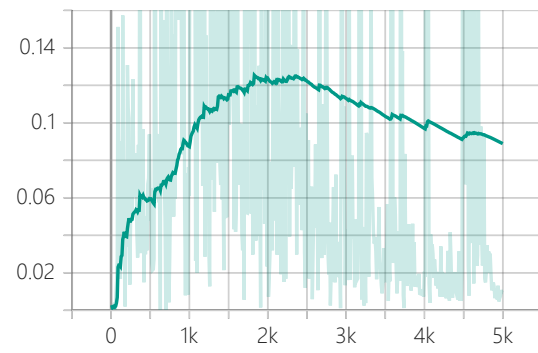
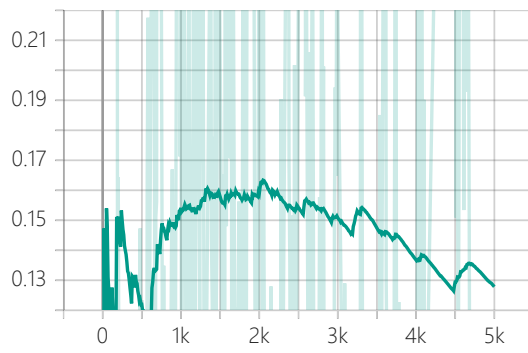


D_x & Final image

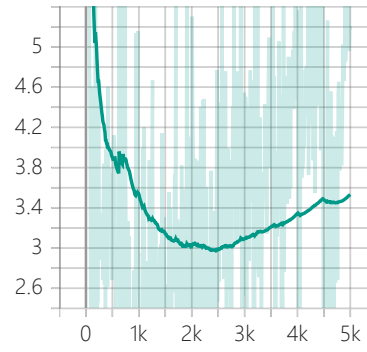
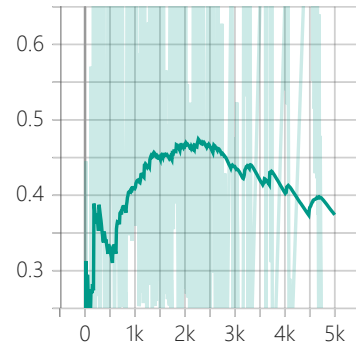


latent: 64 / hidden: 100

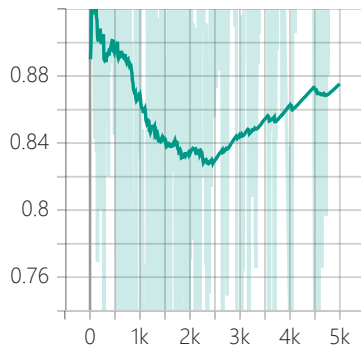
D_G_z1 & D_G_z2



discriminator loss & generator loss

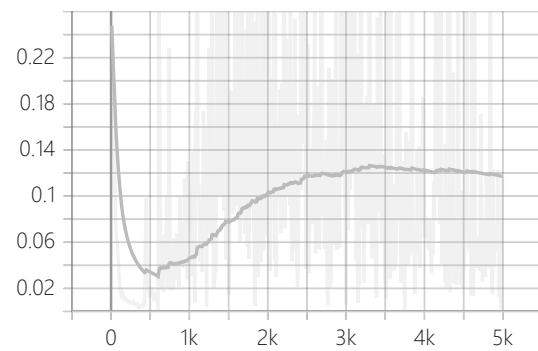
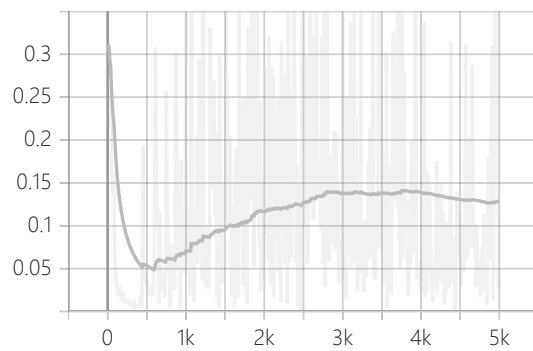


D_x & Final image

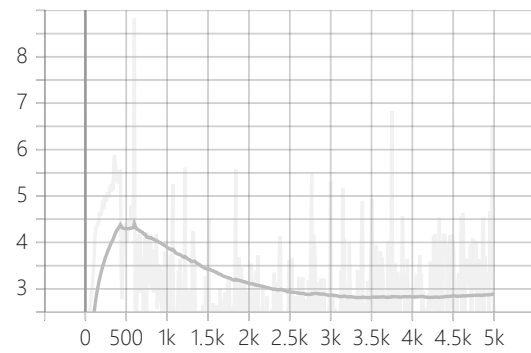
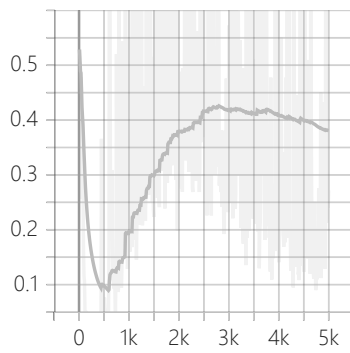


latent: 100 / hidden: 16

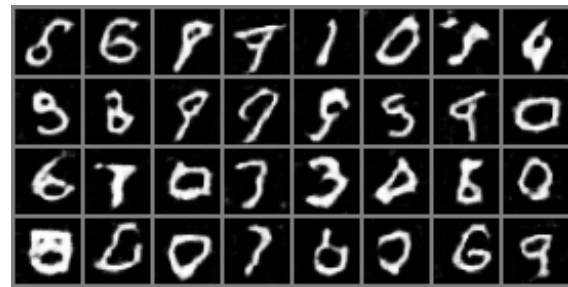
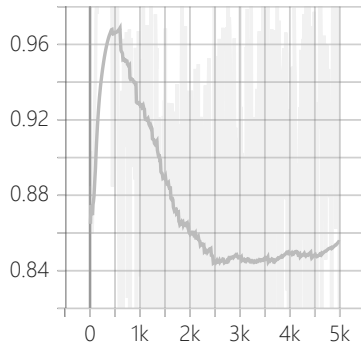
D_G_z1 & D_G_z2



discriminator loss & generator loss

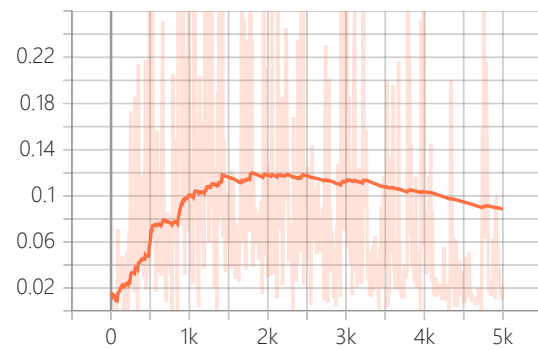
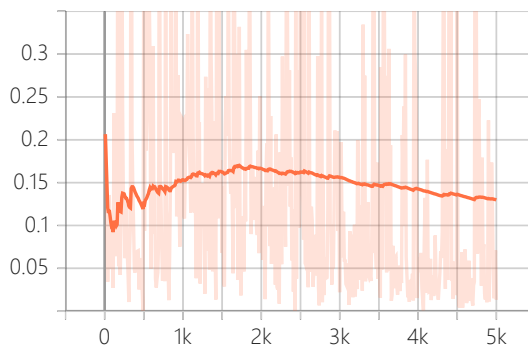


D_x & Final image

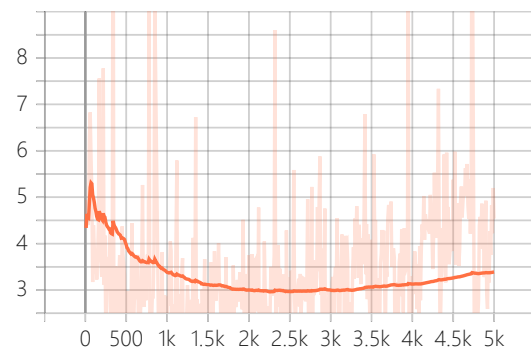
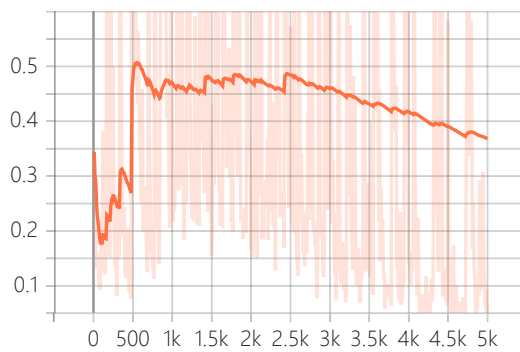


latent: 100 / hidden: 64

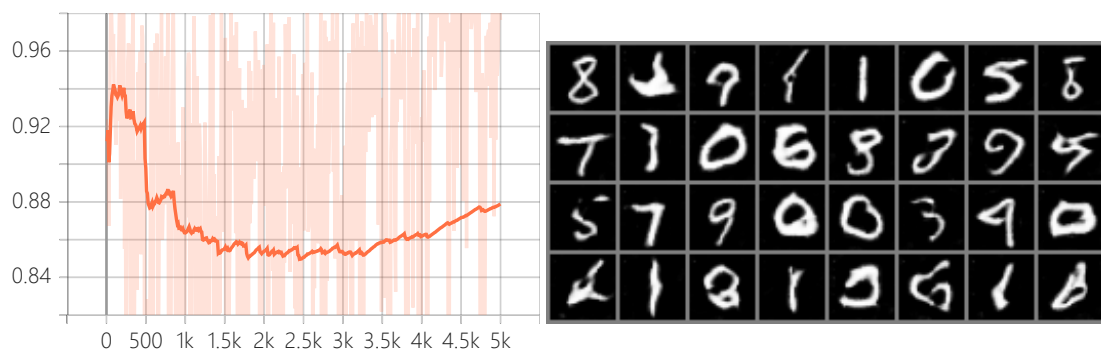
D_G_z1 & D_G_z2



discriminator loss & generator loss

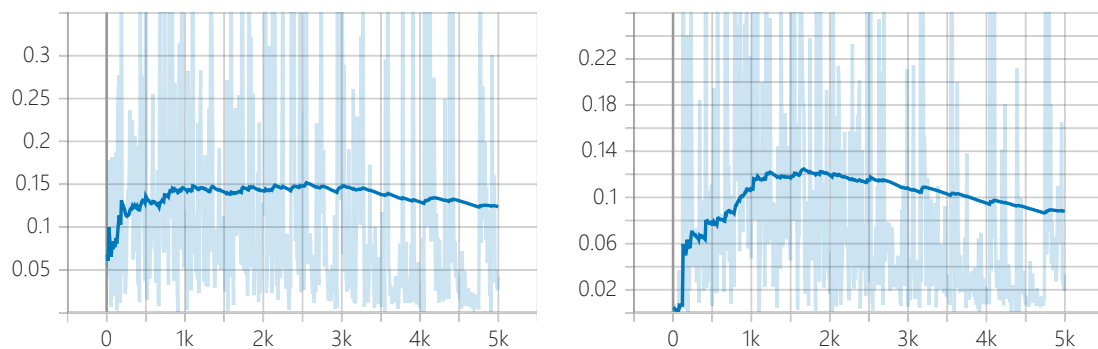


D_x & Final image

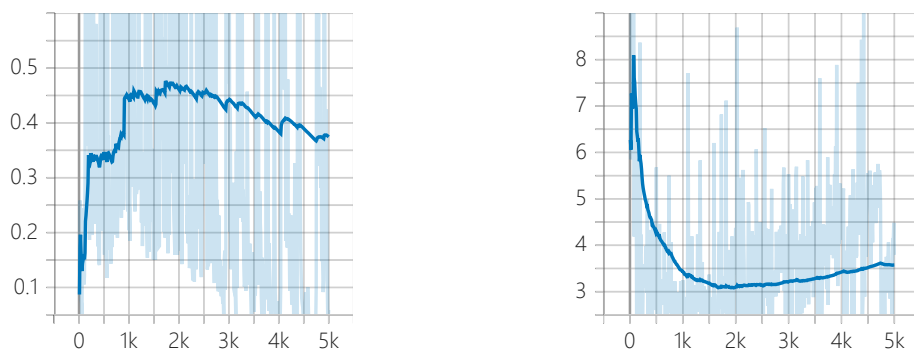


latent: 100 / hidden: 100

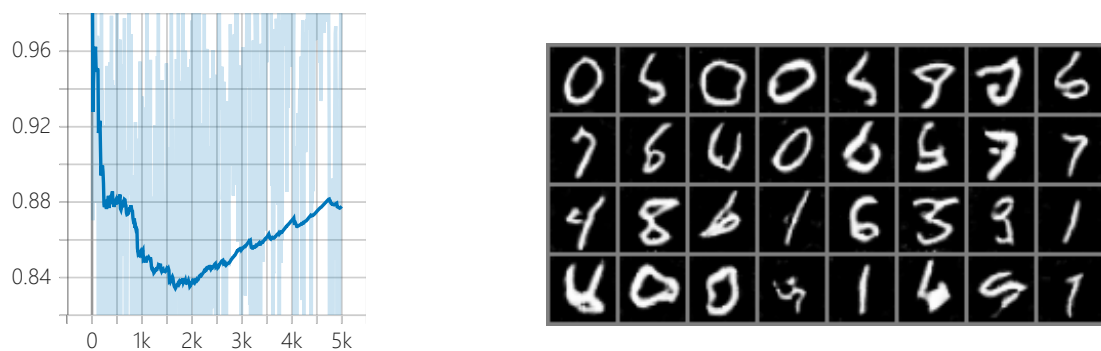
D_G_z1 & D_G_z2



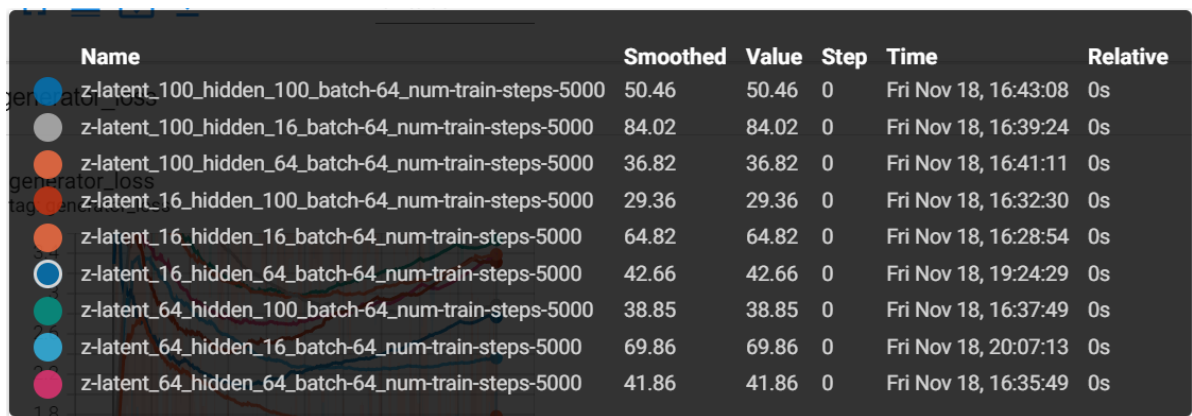
discriminator loss & generator loss



D_x & Final image



2 FID Score



Name	Smoothed	Value	Step	Time	Relative
z-latent_100_hidden_100_batch-64_num-train-steps-5000	50.46	50.46	0	Fri Nov 18, 16:43:08	0s
z-latent_100_hidden_16_batch-64_num-train-steps-5000	84.02	84.02	0	Fri Nov 18, 16:39:24	0s
z-latent_100_hidden_64_batch-64_num-train-steps-5000	36.82	36.82	0	Fri Nov 18, 16:41:11	0s
z-latent_16_hidden_100_batch-64_num-train-steps-5000	29.36	29.36	0	Fri Nov 18, 16:32:30	0s
z-latent_16_hidden_16_batch-64_num-train-steps-5000	64.82	64.82	0	Fri Nov 18, 16:28:54	0s
z-latent_16_hidden_64_batch-64_num-train-steps-5000	42.66	42.66	0	Fri Nov 18, 19:24:29	0s
z-latent_64_hidden_100_batch-64_num-train-steps-5000	38.85	38.85	0	Fri Nov 18, 16:37:49	0s
z-latent_64_hidden_16_batch-64_num-train-steps-5000	69.86	69.86	0	Fri Nov 18, 20:07:13	0s
z-latent_64_hidden_64_batch-64_num-train-steps-5000	41.86	41.86	0	Fri Nov 18, 16:35:49	0s

Section 1 中模型对应的 FID 如上图所示。

3 How Latent dimension and Hidden dimension influence GAN

3.1 Latent dimension

latent dimension 是我们用 generator 生成图片之前采样的样本空间。换句话说，**latent dimension 的维度代表着 generator 生成图片中的某一种特征。**

可以看到，Section 2 中的数据来看，随着 latent dimension 的增加，模型最终的 FID 会降低。这说明更大的 latent dimension 的维数会导致更多需要 generator 学习更多从 latent space 到最终生成图片空间的技巧。由于训练步数相同，更低的 latent dimension 能够使得 generator 学的更好，从而得到更低的 FID score。

3.2 Hidden dimension

hidden dimension 是对 generator 生成图片的整个 pipeline 大小的描述。直观上，越大的 hidden dimension 能够**后续提供给 generator 更大的选择空间**，从而得到更好的选择。

但是，由于我们这里是将 generator 和 discriminator 的 hidden dimension 同时更新，hidden dimension 增大对于 generator 有利的同时，**也会给 discriminator 提供更多判断图片是否为真的信息**，从而有利于 discriminator 的判断。

可以看到，Section2 中的实验数据基本上符合随着 hidden dimension 的增加，FID score 降低的趋势，可以支持上述的推理。

4 Nash Equilibrium

纳什均衡本质的含义是博弈的双方都是理性的，而且在双方都不改变策略的条件下，双方都没有办法通过改变自己的策略使得己方占优。

在本次任务中，纳什均衡代表着 discriminator 有 50% 的概率将 generator 生成的图片判断为真，同样有 50% 的概率将真实图片判断为假。

但是，**上述均衡只是一种理想的状态**。在实际情况中，discriminator 和 generator 的进步是相互的，双方实际上都是不断从“非理性”到“理性”进步。而**只有双方同时达到最“理性”的状态才能达到纳什均衡**。

在本次实验中，实际上 discriminator 和 generator 没有达到纳什均衡。这是因为，在优化自己的过程中，需要一个**“最优秀”的对方**来帮助自己提升，但是这个时候实际上**“对方”也不是最优秀的**。因此，**实践中几乎很难达到理论上的纳什均衡点。**

5 Interpolation in Latent Space

在 latent dimension = 100, hidden dimension = 100 的模型中, 生成了下列样本。



可以看到, 当我们在 latent space 实现“线性插值”之后, 生成的图片确实遵循着一定的规律变化。这从侧面证明了 latent space 的每一个维度实际上对应着生成图片中的某一种特点。平滑地改变这种维度, 在生成的图片上, 也是平滑的特征改变。

Bonus Part

6 Mode Collapse

在 latent dimension = 100, hidden dimension = 100 的条件下, 生成了下列样本。



经过人工识别, 上述图片中生成的数字为:

7697984116376176059398706050177043167478836919958911866467672360180680790817141
917330182382039660777

经过统计, 结果如下:

数字	0	1	2	3	4	5	6	7	8	9
次数	12	14	3	9	5	3	15	17	11	11

可以看到，generator 生成数字 2, 4, 5 的次数尤其少，生成数字 1,6,7 的次数尤其多。

原因分析：

根据老师在课上的推导，当 Generator 和 Discriminator 接近于纳什均衡的时候。目标函数在很大程度上受到**逆向 KL 散度** $D_{KL}(p_{\theta}||p_{data})$ 的影响。根据 KL 散度的公式：

$$\int p_{\theta}(x) \log\left(\frac{p_{\theta}(x)}{p_{data}(x)}\right) dx$$

- 当 $p_{\theta} > 0, p_{data} = 0$ 的时候，也就模型在一个不是样本点的地方产生值的时候，会使得逆向 KL 散度很大，也就是会导致很大的惩罚。
- 当 $p_{\theta} = 0, p_{data} > 0$ 的时候，相当于模型在本来应该有样本点的地方没有产生值，但是这个时候逆向 KL 散度很小，对模型的惩罚也很小。

通过上述的分析，模型会**变倾向于变得更加“谨慎”**，只在有样本点的地方生成。这样虽然能够使得模型的生成效果相对较高，但是在很大程度上损害了 diversity，让 generator **更倾向于在某一个已知的样本点上生成**。这便是 mode collapse 本质上的原因。

7 GAN with MLP-based

在 latent dimension = 100, hidden dimension = 100 的条件下，构造 mlp base GAN

具体来说，在进行 mlp 实验的过程中，generation 的 backbone 被换成了

```
nn.Linear(latent_dim, 4*hidden_dim),
nn.BatchNorm1d(4*hidden_dim),
nn.ReLU(),

nn.Linear(4*hidden_dim, 2*hidden_dim),
nn.BatchNorm1d(2*hidden_dim),
nn.ReLU(),

nn.Linear(2*hidden_dim, hidden_dim),
nn.BatchNorm1d(hidden_dim),
nn.ReLU(),

nn.Linear(hidden_dim, 32*32),
nn.Tanh()
```

discriminator 的 backbone 被替换成

```
nn.Linear(32*32, hid_2),
nn.LeakyReLU(),

nn.Linear(hid_2, 2*hid_2),
nn.LeakyReLU(),

nn.Linear(2*hid_2, 4*hid_2),
nn.LeakyReLU(),

nn.Linear(4*hid_2, 1),
nn.Sigmoid()
```

考虑到 mlp 的参数多于 cnn，这里一组实验训练 5000 steps，一组训练了 10000 steps

5000 steps

FID score : 138.5



10000 steps

FID score: 118



结果分析:

可以看到，无论是从 FID 指标还是生成结果来评判，MLP 的生成效果要差于 CNN。除了生成的图片中的数字更加无法辨认之外，**MLP 产生的图像中还掺杂着许多 CNN 没有的噪点。**

原因分析:

我认为上述现象出现的主要原因在于，MLP 不具有 CNN 卷积操作带来的局部性。

MLP 本质上在 generator 的每一层中实现的是全连接。也就是**一个神经元可以影响上一层图像的所有部分**。这就很有可能导致一个神经元的错误会给整个图像带来很大的影响。因此 MLP 的生成结果中会有更多的噪点。

同时，由于 CNN 的参数全部集中在卷积核上，因此 CNN 作为 backbone 可以更快的学习到 data 的特征。

因此，无论是从效率和效果上来看，CNN 都比 MLP 更适合作为 GAN 的 backbone。

