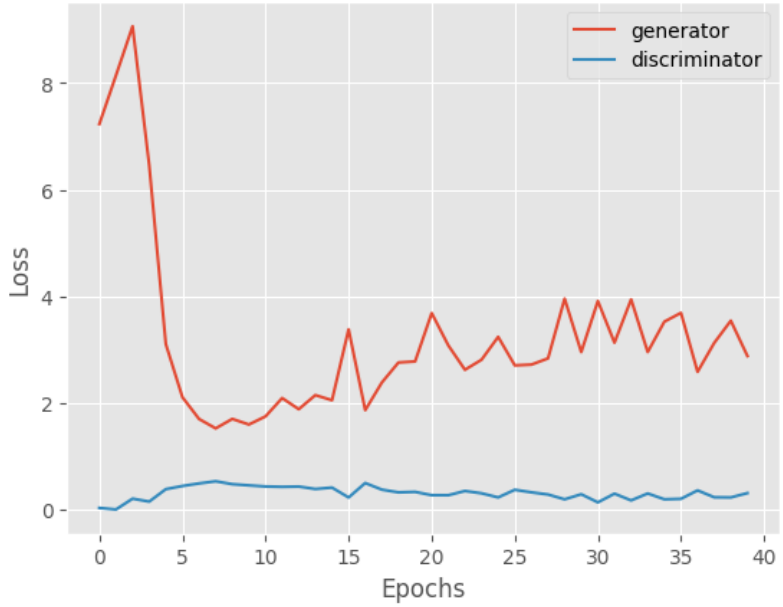
DATA303 HW3: MNIST with GANs

데이터과학과 2021320322 윤민서

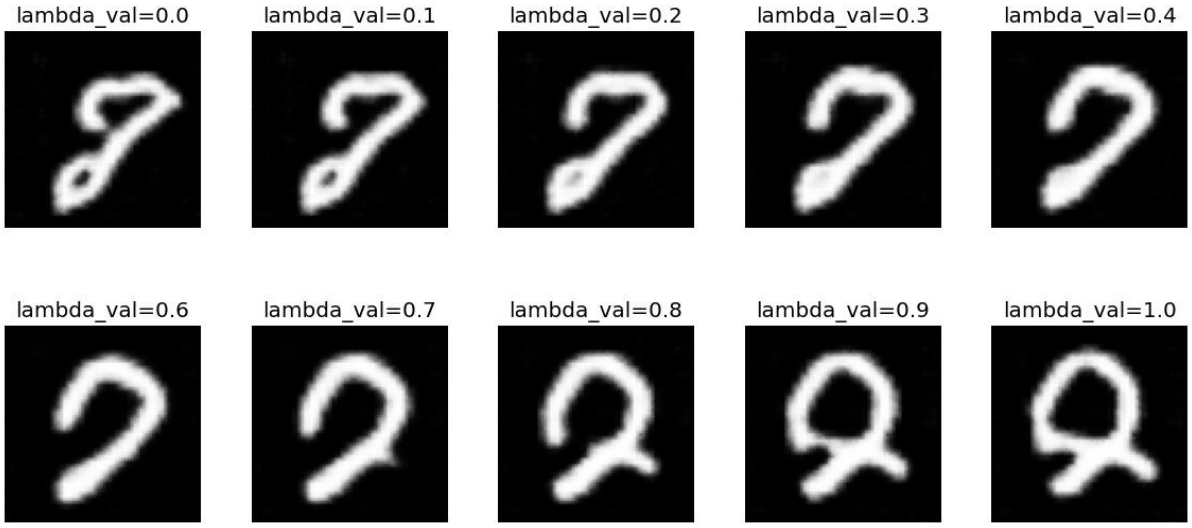
1. Learn a GAN (deconv) with 64x64

a. Learning curve

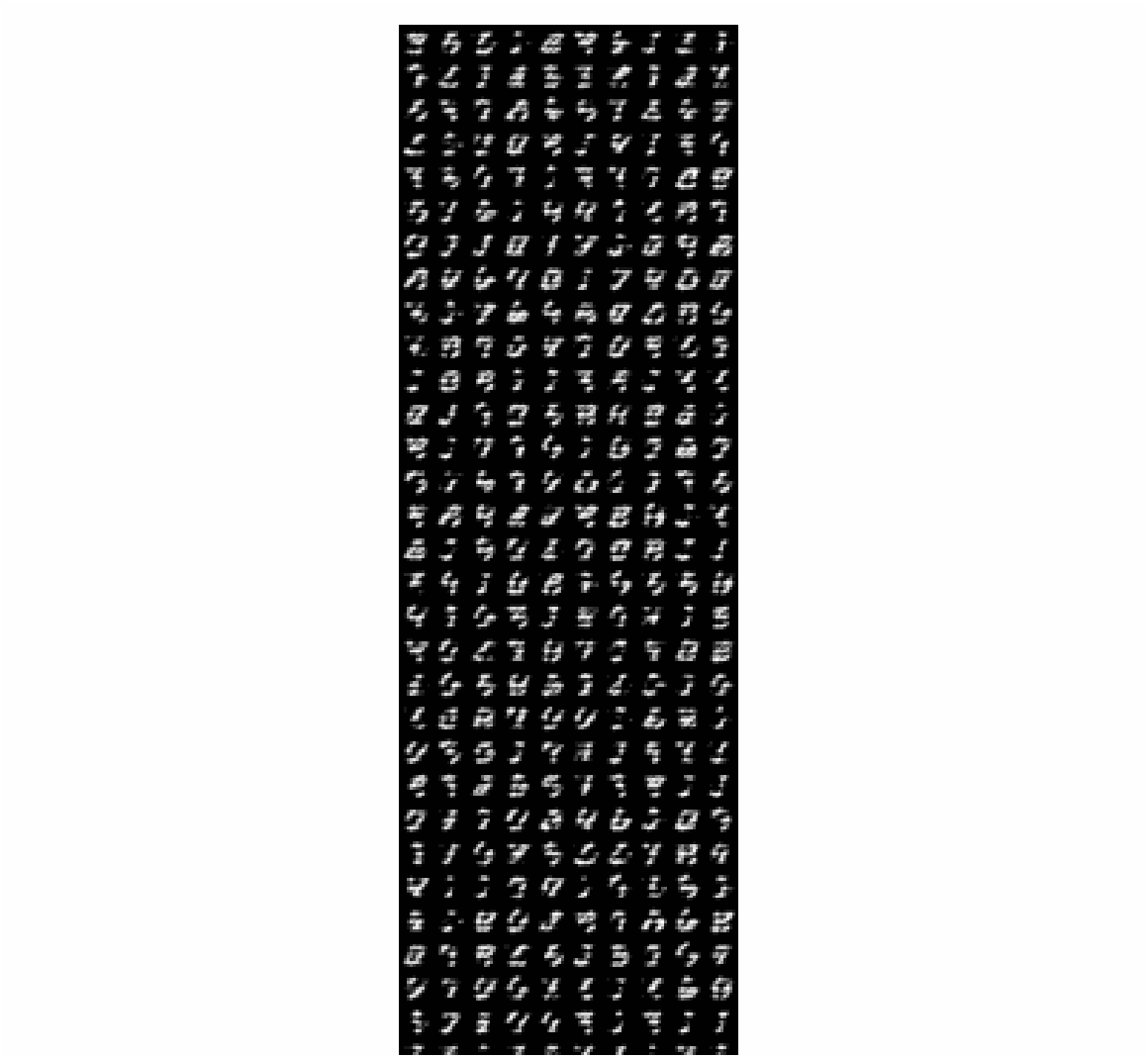


b. Visualize Generated samples

c. Interpolation



d. Visualize generated images using the same latent vectors across epochs

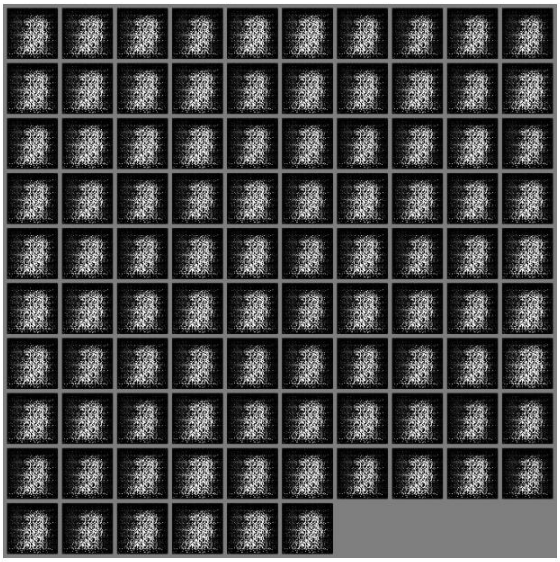


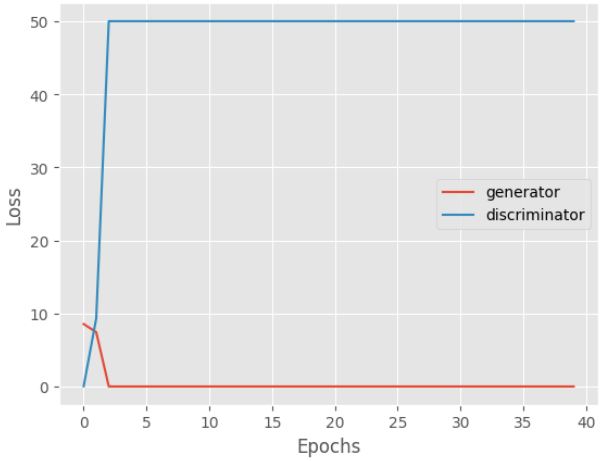
+ 시도해 본 hyperparameters

latent_dims	num_epochs	batch_size	learning_rate	숫자 생성 여부
100	40	512	2e-4	X
100	40	128	2e-4	X
100	40	512	1e-4	O (최종 모델)
100	40	128	1e-4	X
200	40	512	1e-4	X
200	40	512	5e-5	X
200	40	512	1e-5	X
100	40	512	1e-5	X

숫자가 생성되지 않은 원인으로는 mode collapse가 대표적이다. 즉, generator가 입력 값을 mode에 치우쳐 변환시키거나 일부 값에 대해서만 학습이 반복되어 전체 분포를 학습하지 못하는 것 등이 원인이 될 수 있다. 또한 hyperparameter 자체가 적절하지 않아서 수렴에 실패하는 경우도 종종 있었다.
더욱 구체적으로, discriminator가 지나치게 학습되어 (Loss < 0.001) generator가 학습 경향을 따라가지 못하는 경우가 있거나 discriminator가 수렴 자체를 실패하는 경우가 있다 (generator Loss = 0, discriminator Loss = 50).

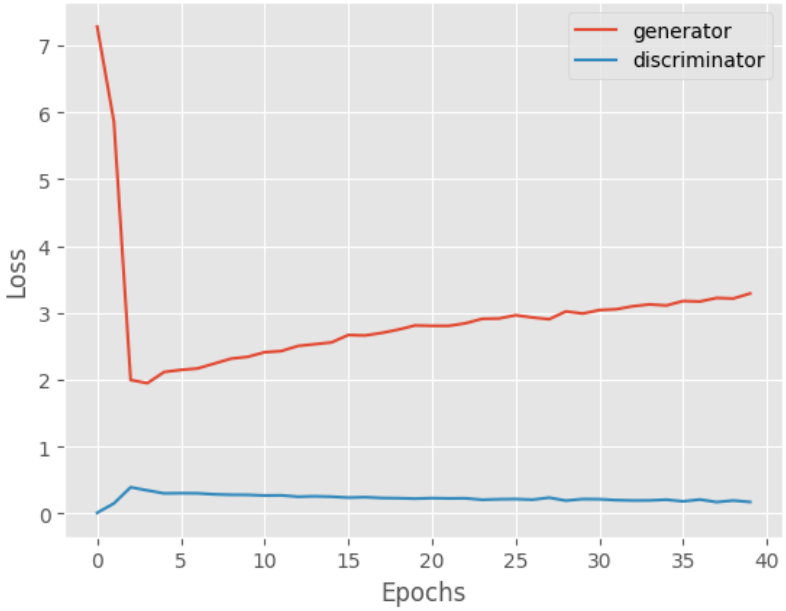
숫자가 생성되지 않은 결과의 예시는 다음과 같다.



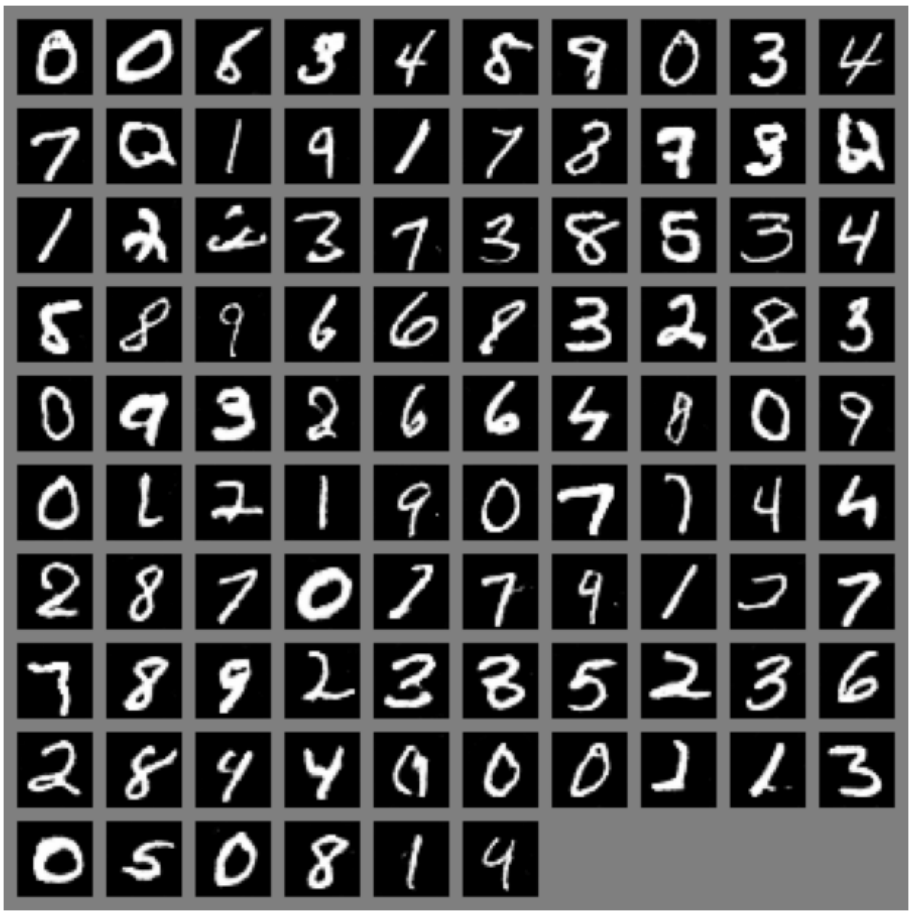


2. Learn a GAN (deconv) with 28x28

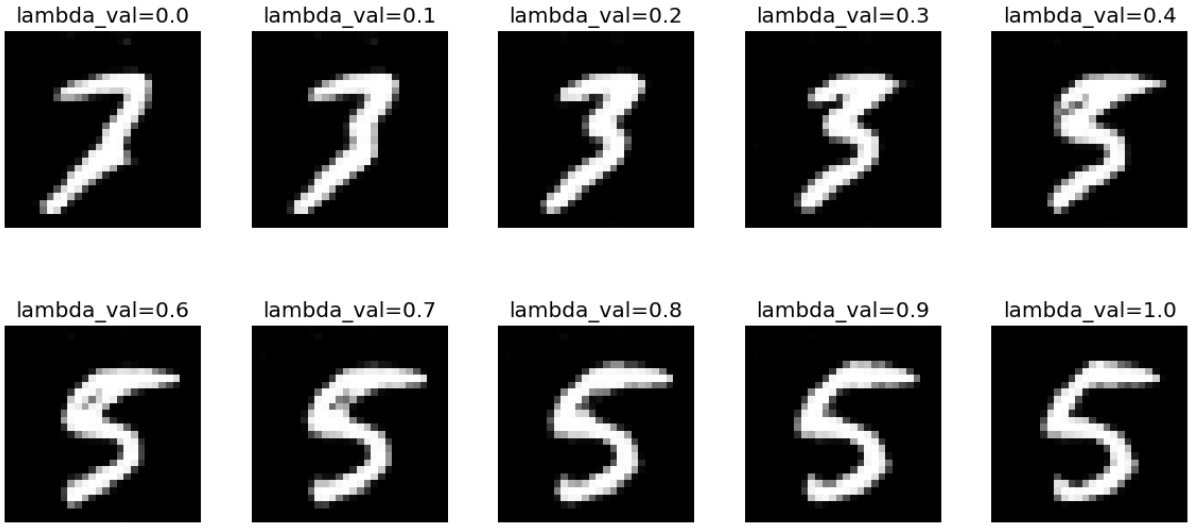
a. Learning curve



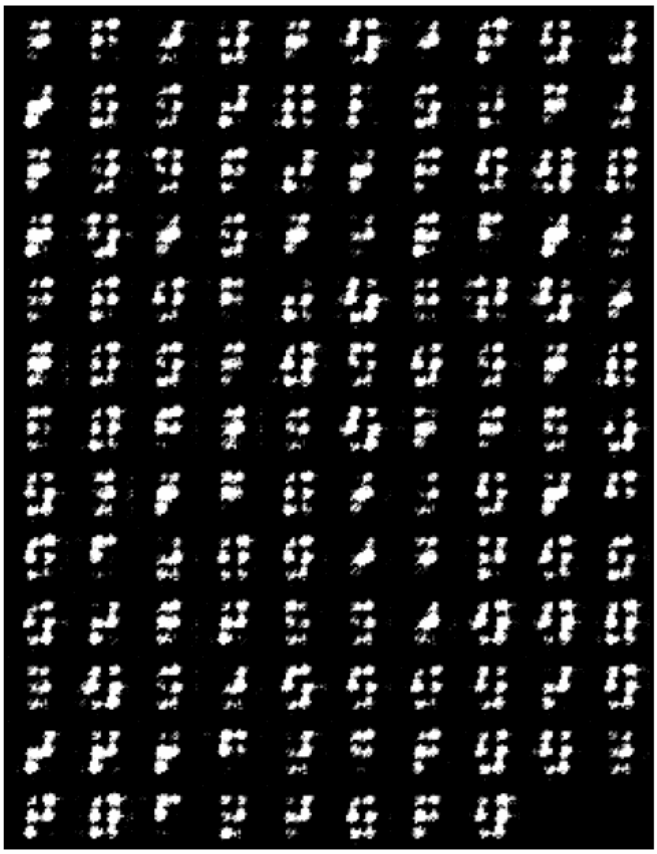
b. Visualize Generated samples



c. Interpolation



d. Visualize generated images using the same latent vectors across epochs



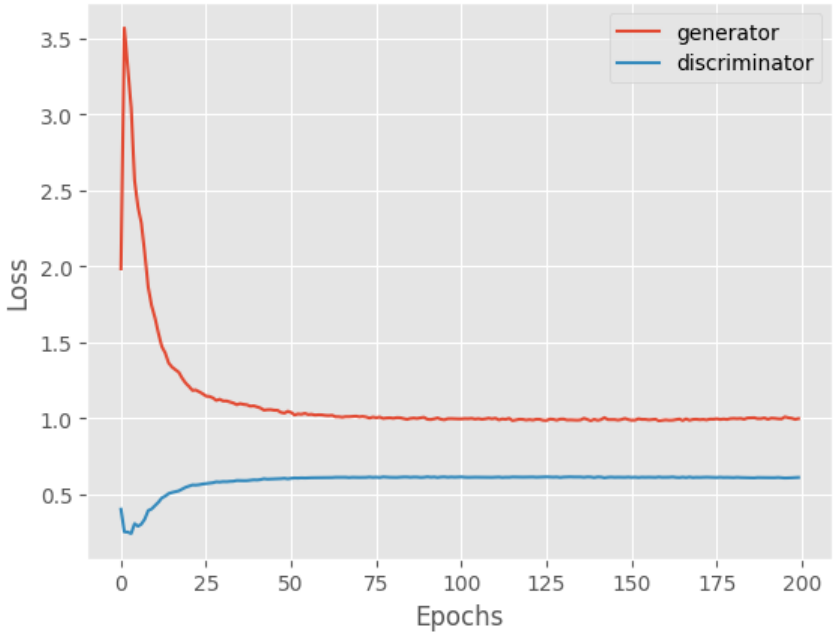
+ 시도해 본 hyperparameters

latent_dims	num_epochs	batch_size	learning_rate	숫자 생성 여부
100	40	512	1e-4	O
200	40	512	1e-4	O
100	40	128	1e-4	O (최종 모델)
100	40	128	1e-5	X

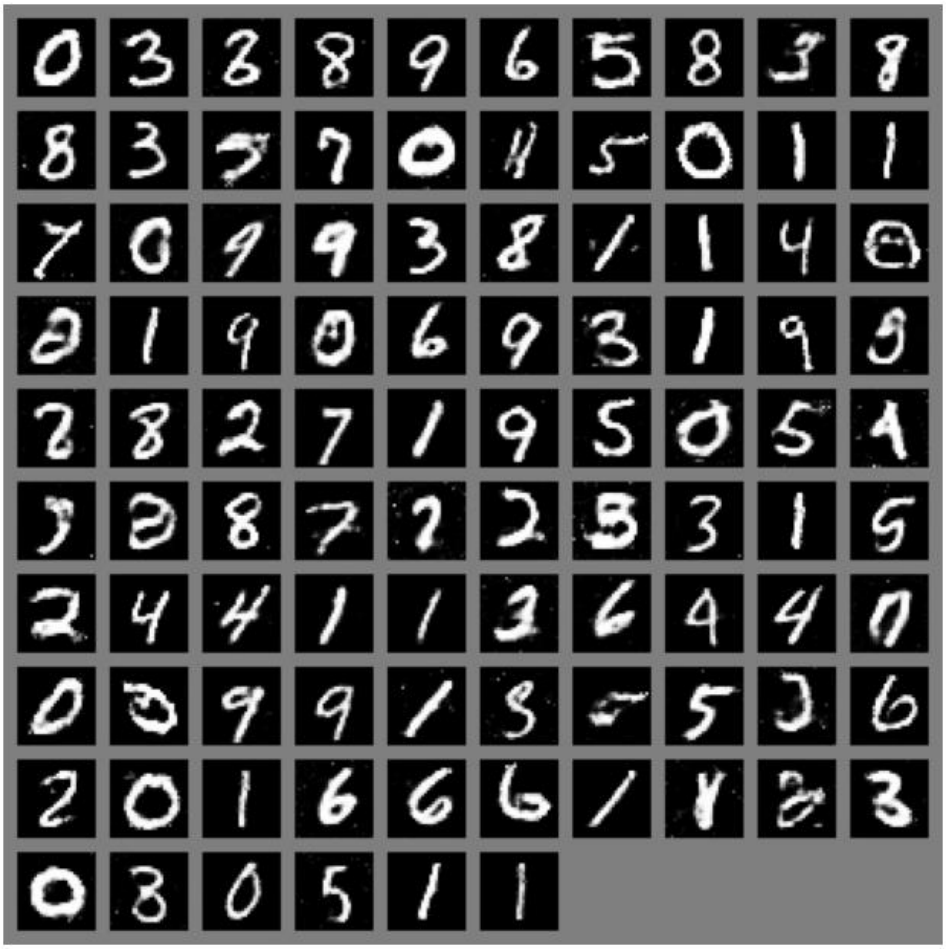
64x64 deconv GAN과는 다르게 28x28의 경우에는 학습이 조금 더 수월한 경향이 있었다. 먼저 64x64에서 최종 모델로 선정된 hyperparameter를 시도해 본 뒤 latent_dims, batch_size, learning_rate를 하나씩만 변경하여 실험을 진행했다. batch_size를 512에서 128로 줄인 실험에서 만족할 만한 결과를 얻었고 최종 모델로 선정하였다. 마지막 실험의 경우에는 learning_rate가 지나치게 작아 수렴하기 전에 40 epoch가 전부 끝난 것으로 생각된다.

3. Learn a GAN (deconv) with 64x64

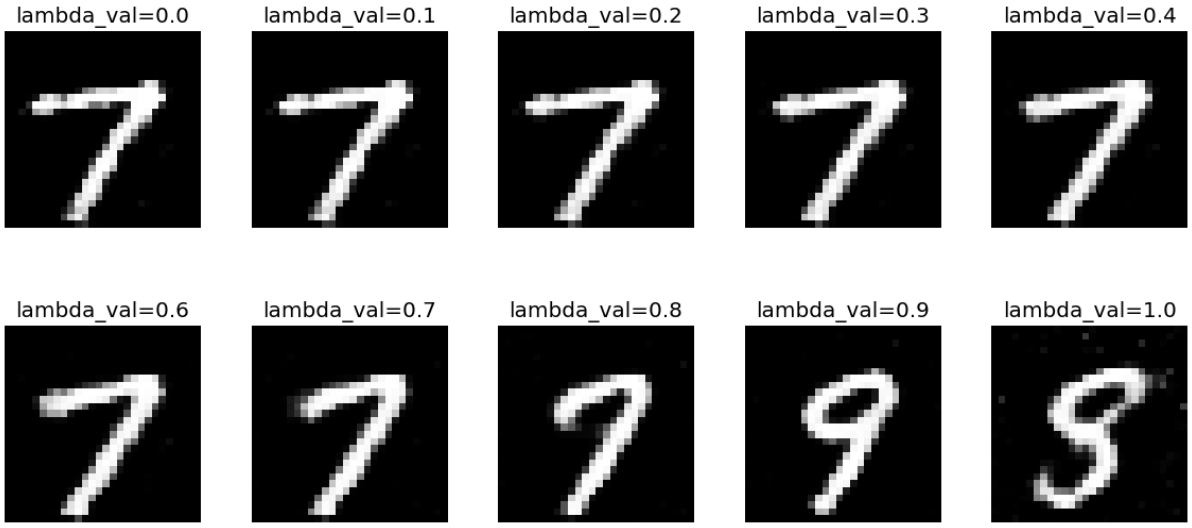
a. Learning curve



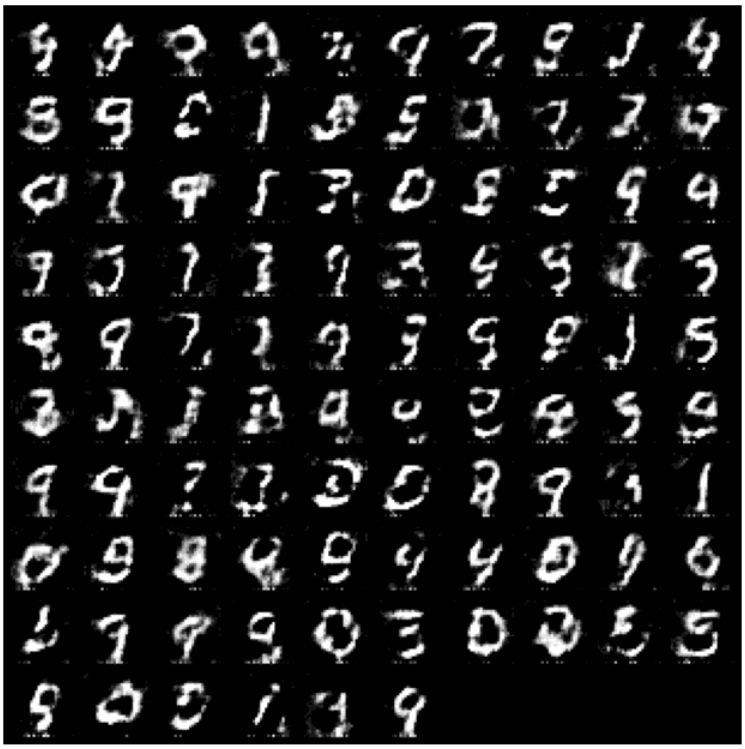
b. Visualize Generated samples



c. Interpolation



d. Visualize generated images using the same latent vectors across epochs

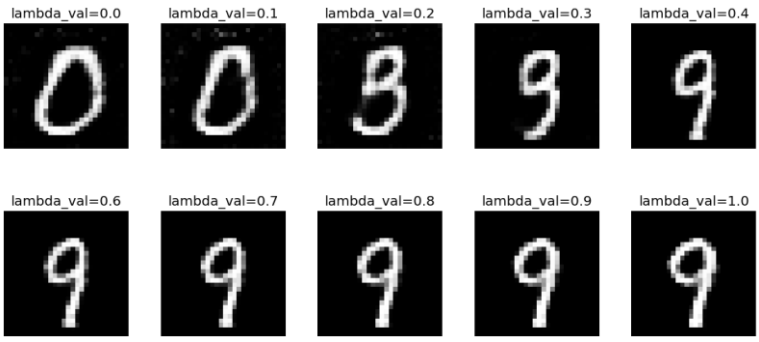


구현 상의 이유로 같은 data 안에서 gif를 생성되지는 않았지만, 경향성을 충분히 확인할 수 있다.

+ 시도해 본 hyperparameters

latent_dims	num_epochs	batch_size	learning_rate	숫자 생성 여부
100	40	128	1e-4	X
100	40	128	1e-5	X
100	40	512	1e-4	X
100	40	512	1e-5	X
100	200	128	1e-4	O
100	200	128	2e-4	O (최종 모델)
100	200	64	2e-4	O
200	200	64	2e-4	O

28x28 해상도의 이미지를 생성하는 것을 목표로 실험을 진행하였다. deconv 기반 GAN과는 달리, MLP 기반 GAN은 훨씬 더 많은 epoch를 요구했다. 또한 이전 실험들과는 달리 learning_rate가 1e-4일 때에 비해 2e-4일 때 learning curve가 훨씬 안정적인 경향을 보였다. deconv 기반 GAN에 비하여 parameter의 개수는 반 정도밖에 되지 않지만 epoch 당 학습 시간이 비슷한 점을 발견했다. 여기에서 MLP 기반 GAN이 상대적으로 FLOP이 크다, 즉 상대적으로 dense하다고 생각할 수 있다.
추가로 마지막 실험의 결과 또한 상당히 좋은 결과라고 생각하여 아래에 generated sample과 interpolation을 첨부하였다.



4. InfoGAN

a. Add mutual information/information loss to a GAN (deconv) with 64x64

Image_Generation_by_GAN.ipynb에서 관련된 부분 중 일부를 첨부하였다.

```
crit1 = nn.BCELoss()

crit2 = nn.CrossEntropyLoss()

crit3 = NormalNLLLoss()

loss_real = crit1(probs_real, label_real)

loss_fake = crit1(probs_fake, label_fake)

disc_loss = loss_real + loss_fake

gen_loss = crit1(probs_fake, label_real)

gen_loss += crit2(q_logits[:, 0:10], target[0])

gen_loss += crit3(noise[:, latent_dims + 10 :].view(-1, 2), q_mu, q_var)
* 0.1
```

b. Reproduce Figure 2 (a), (c), (d), in infoGAN with latent codes c1, c2, c3

InfoGAN Figure 2 - (a) Varying c1 on InfoGAN (Digit type)

InfoGAN Figure 2 - (c) Varying c2 from −10 to 10 on InfoGAN (Rotation)

InfoGAN Figure 2 - (d) Varying c3 from −10 to 10 on InfoGAN (Width)

28x28 기준으로 구현된 구조를 64x64로 고치는 동안 일어난 구조의 변형이 어느 정도 있었기 때문에 완전히 좋은 결과를 도출해내지는 못했지만, varying의 경향성이 어느 정도 보인다고 판단할 수 있다. InfoGAN 논문은 c2와 c3을 -2부터 2까지 변화시키며 plot 했지만 본 실험에서는 경향성이 잘 보이지 않는다고 생각하여 -10부터 10까지 변화시키며

+ 시도해 본 hyperparameters

latent_dims	num_epochs	batch_size	learning_rate	비고
30	22 (early stopping)	64	G: 1e-3, D: 5e-5	최종 모델

latent_dims: 30, 62, 100

num_epochs: 30, 40, 100

batch_size: 32, 64, 128, 512

learning_rate: 1e-5, 2e-5, 5e-5, 1e-4, 2e-4, 1e-3

실험 도중 discriminator의 loss가 빠르게 하락하며 generator가 이를 따라가지 못해 generator의 loss가 빠르게 상승하는 경향을 발견했다. 이에 근거하였을 때, latent_dims를 작게 하는 경우에 성능이 좋았다.

learning_rate는 generator와 discriminator가 같은 것을 사용하는 경우와 다른 것을 사용하는 경우가 있다. 위와 같은 이유로 generator에 비해 discriminator가 학습을 더 빨리 하는 경향이 있었기 때문에 서로 다른 learning_rate를 사용하며 discriminator의 learning_rate를 작게 하는 경우에 성능이 좋았다.