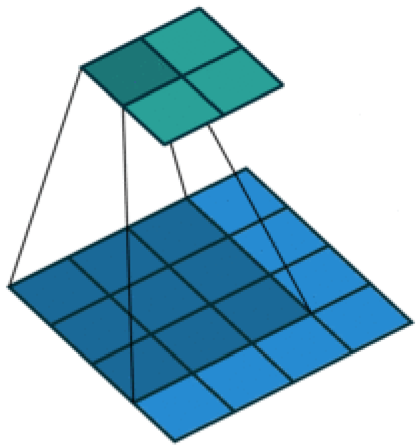
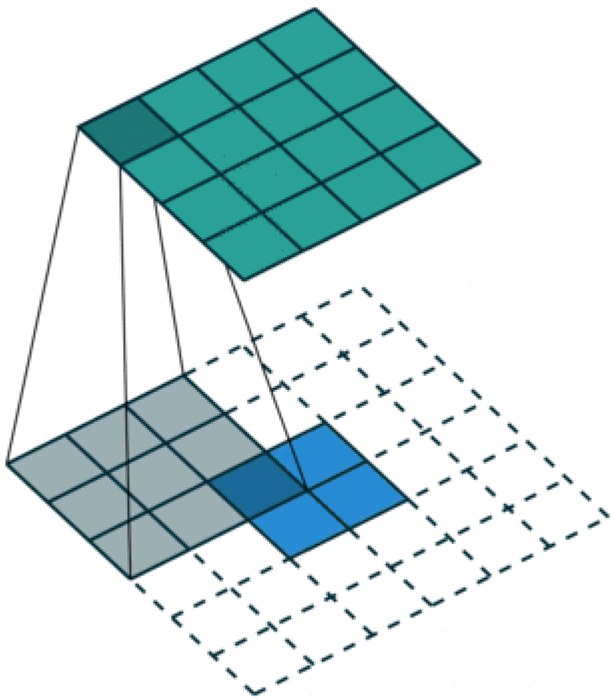
InfoGAN

助教：來俊聖

Conv Layer v.s. ConvTranspose Layer



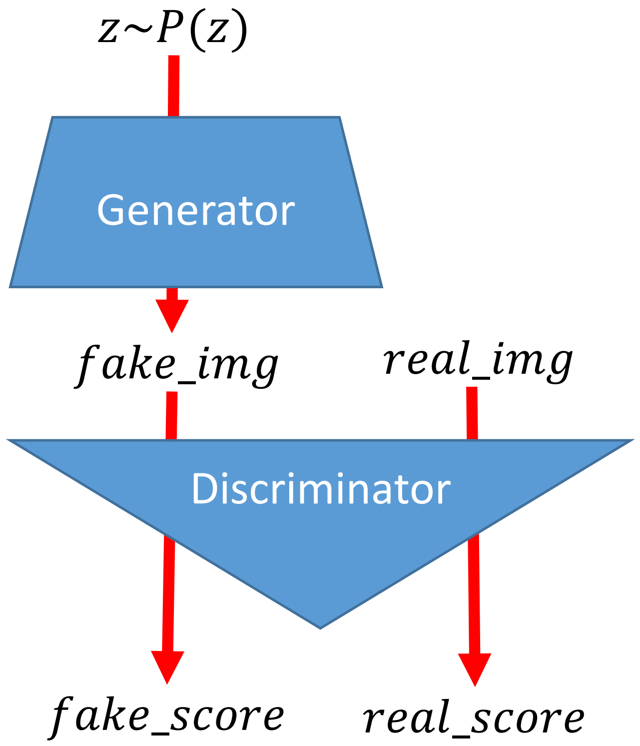
Conv Layer

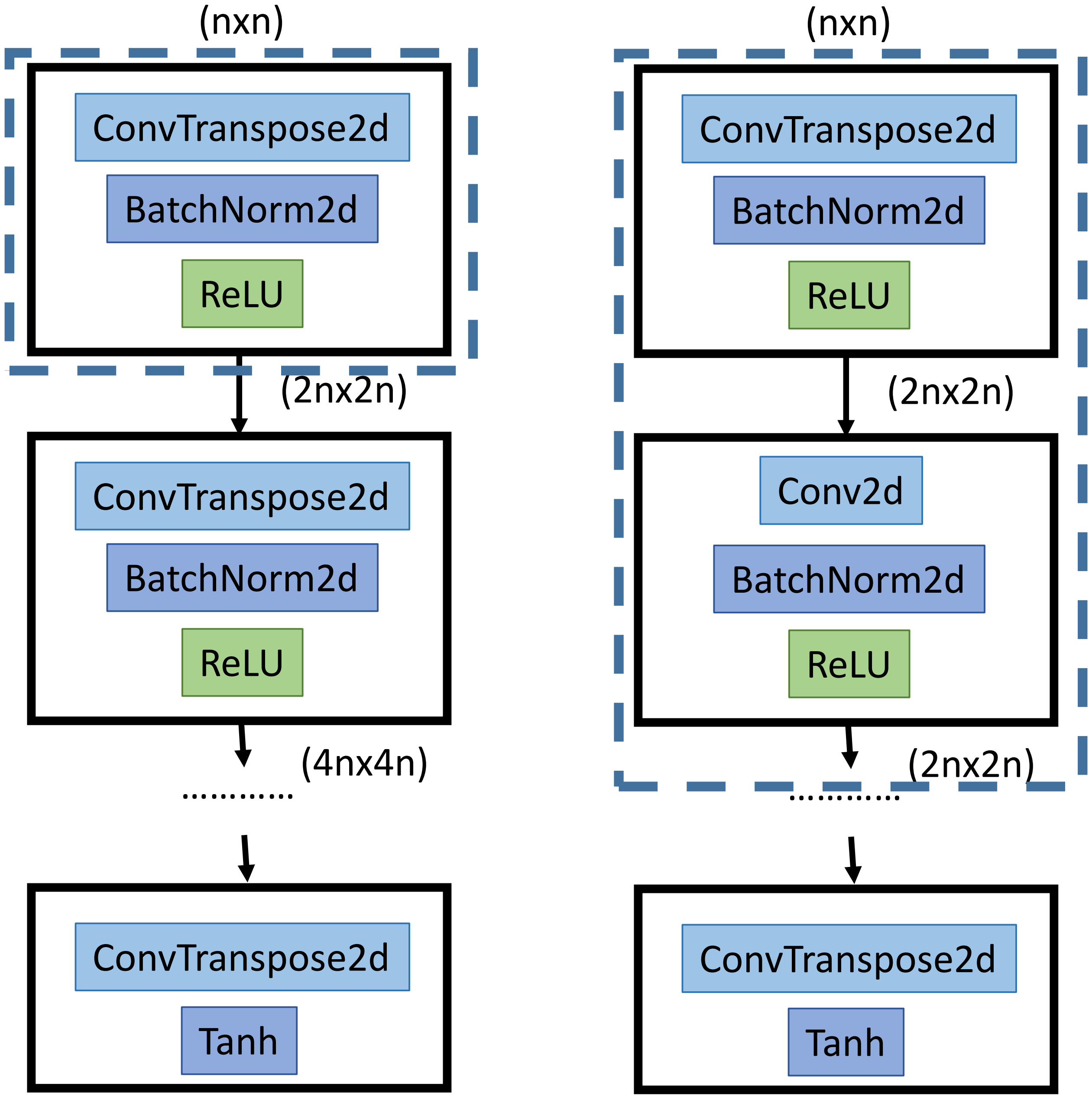


ConvTranspose Layer

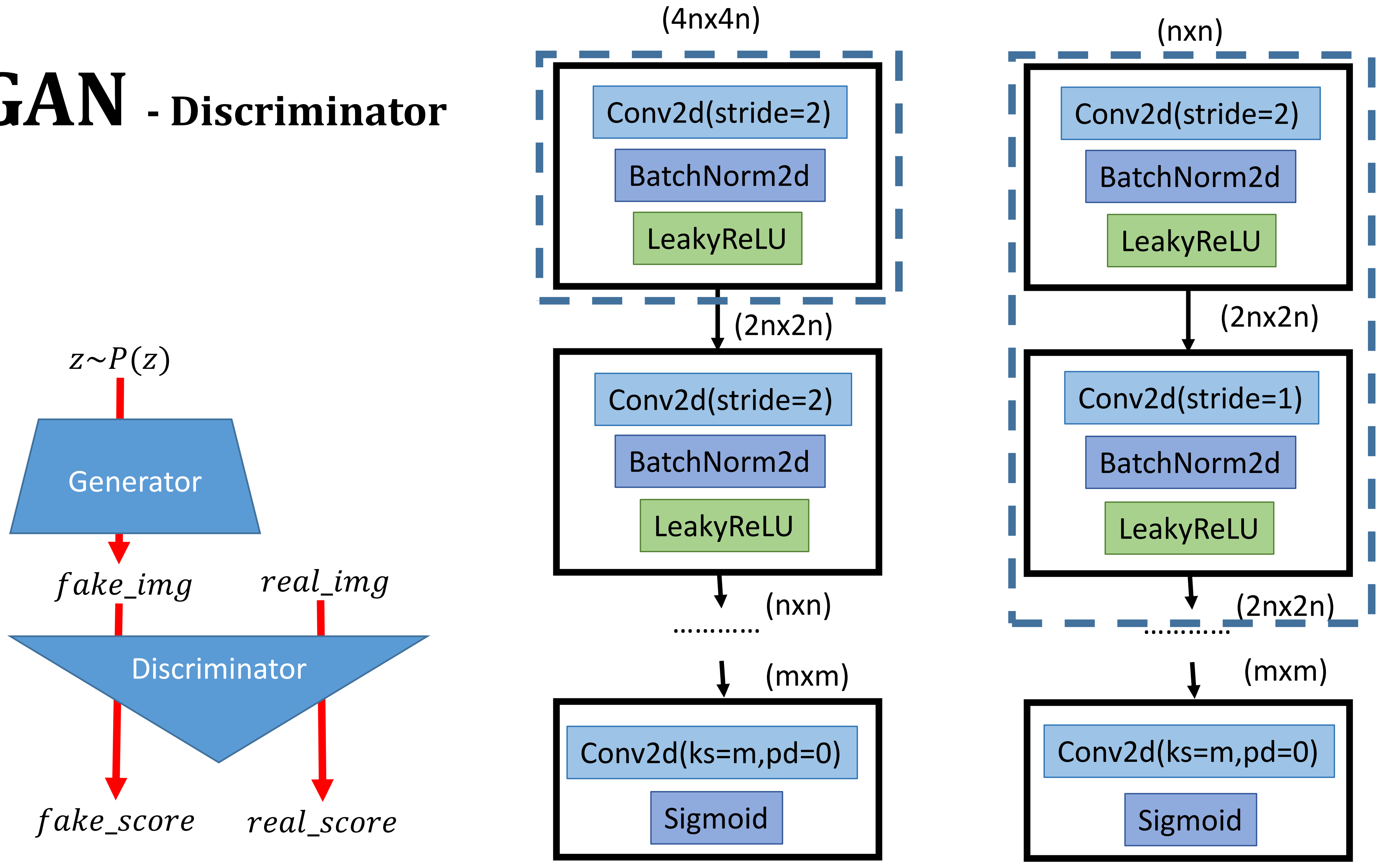
DCGAN - Generator

example code:
https://github.com/pytorch/examples/blob/master/dcgan/main.py





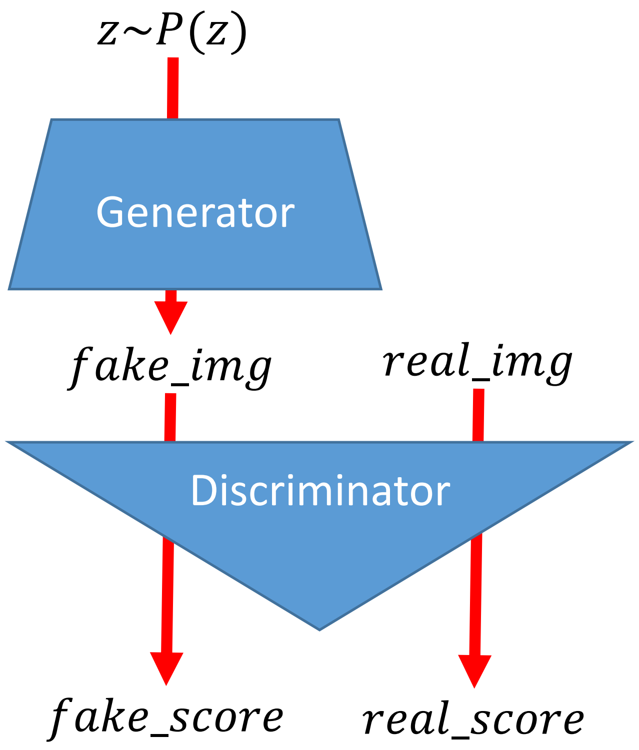
DCGAN - Discriminator



Standard GAN

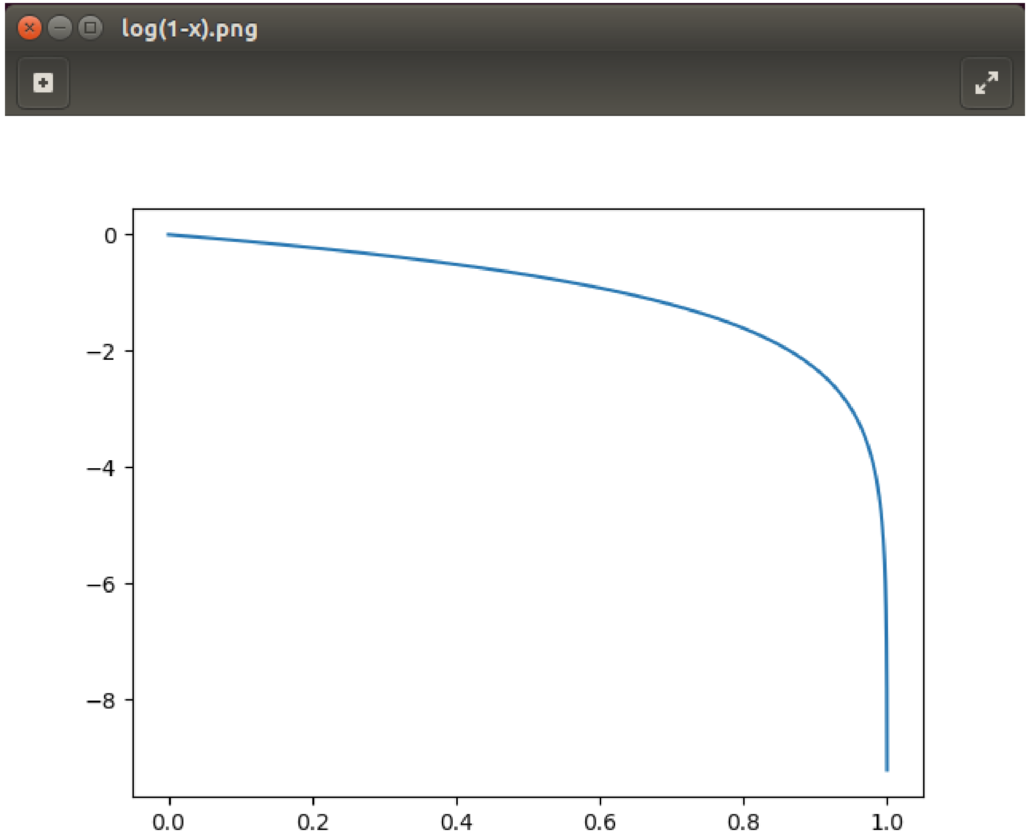
$$L_D = -\log(\mathrm{D}(I_{real})) - \log\left(1 - D\left(G(z)\right)\right)$$

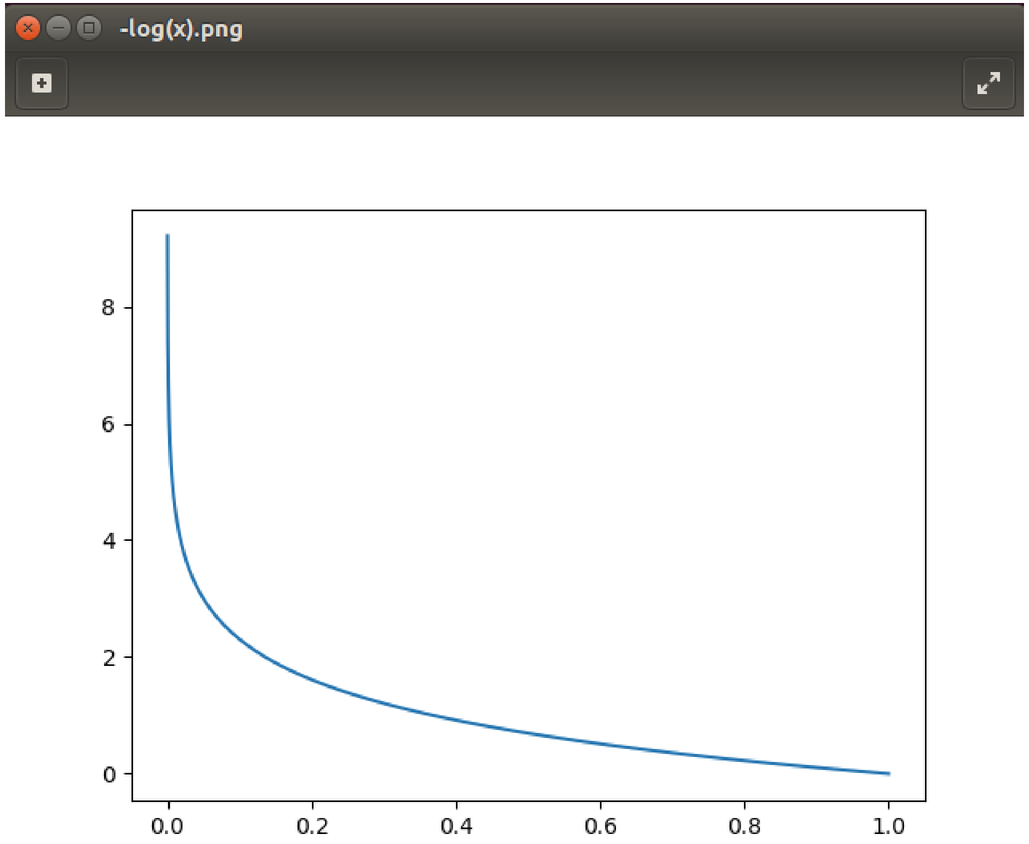
$$\begin{cases} L_G = \log\left(1 - D\left(G(z)\right)\right) \\ L_G = -\log\left(D\left(G(z)\right)\right) \end{cases}$$



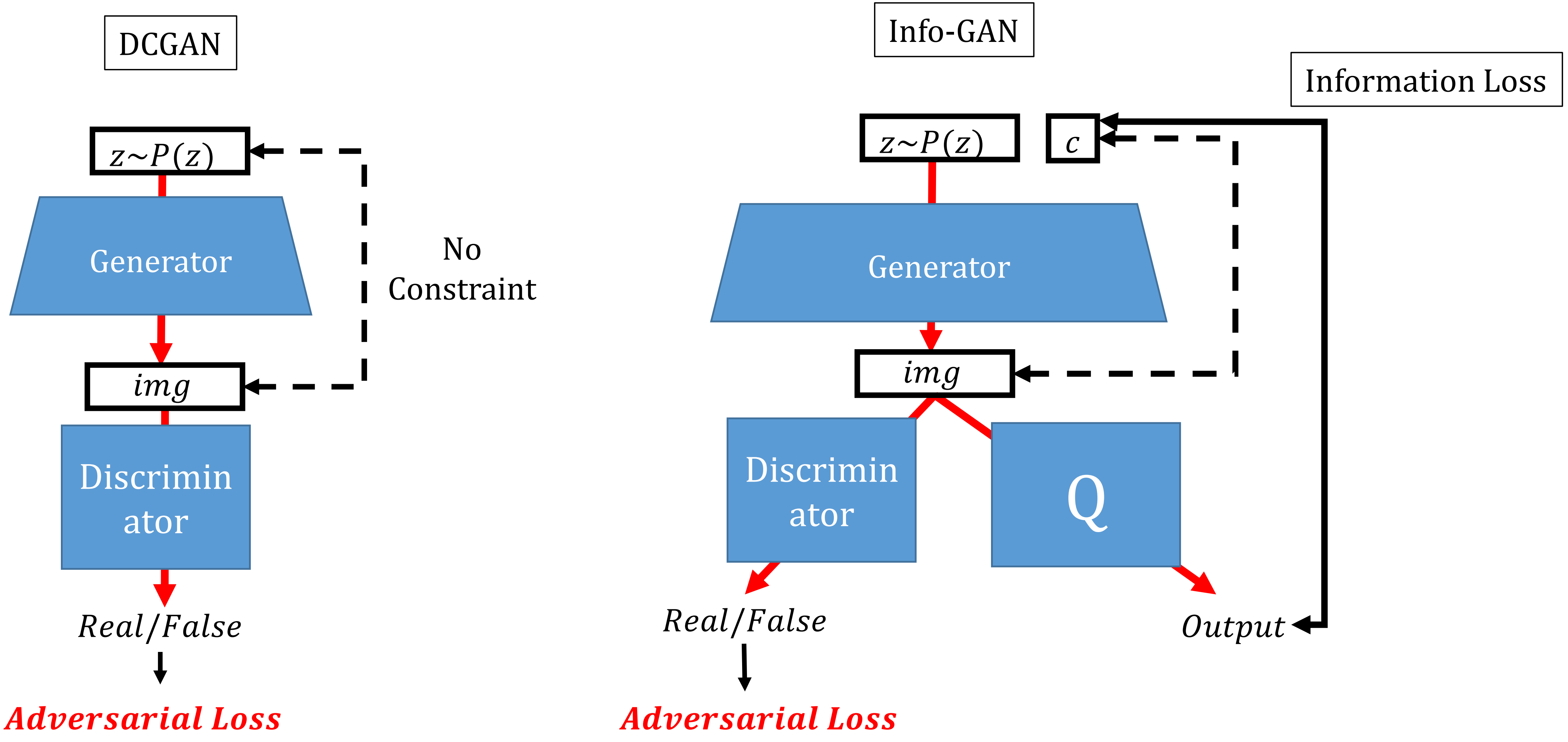
In this lab, you can use either of them, but you should tell me which one you use in your report.

Standard GAN Loss





Info-GAN



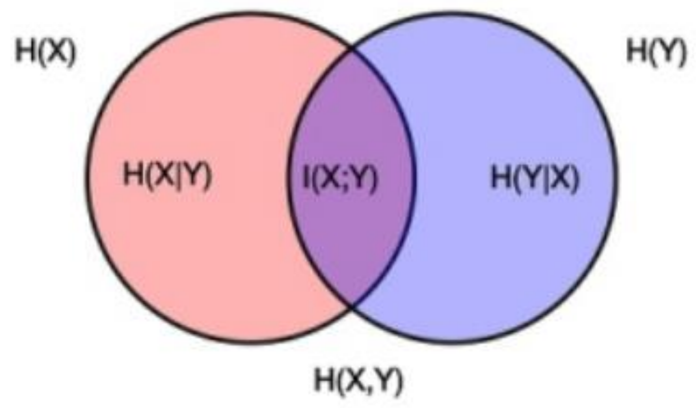
DCGAN
$z\sim P(z)$
Generator
img
Discriminator
$Real/False$
Adversarial Loss
No Constraint
Info-GAN
Information Loss
$z\sim P(z)$
c
Generator
img
Discriminator
Q
$Real/False$
Adversarial Loss
$Output$

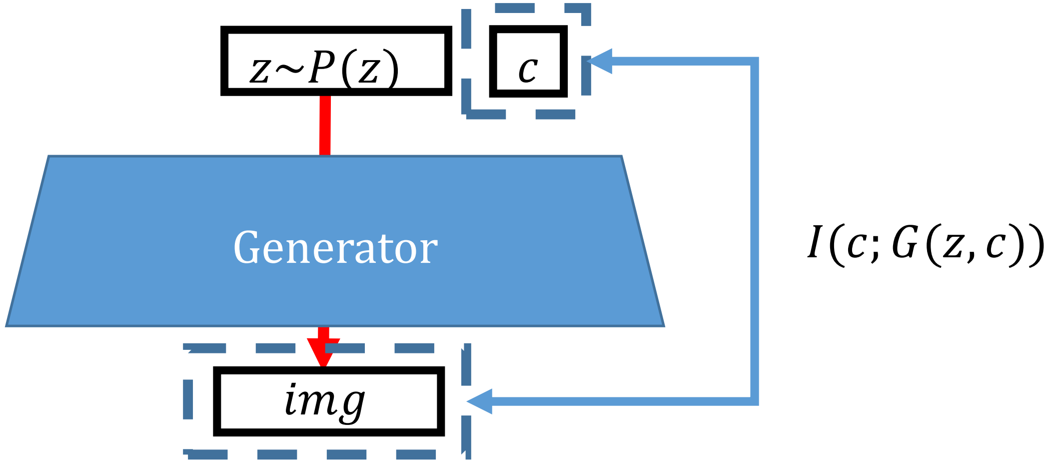
Info-GAN

In Information Theory:
Mutual Information: I(X;Y) = H(X)-H(X|Y) = H(Y)-H(Y|X)

$$H(Y) = -\sum_{y} \log p(y) p(y)$$

$$I(X;Y) = H(X) - H(X \mid Y) = H(Y) - H(Y \mid X)$$





Info-GAN pf.

I(c;G(z, c)) = H(c)-H(c|G(z,c))

$$=E_{x\sim G(z,c)}\left[E_{c'\sim P(c|x)}[logP(c'|x)]\right] + \mathrm{H(c)}$$

$$=E_{x\sim G(z,c)}\left[D_{KL}(P(\cdot\,|x)||Q(\cdot\,|x)) + E_{c'\sim P(c|x)}[logQ(c'|x)]\right] + \mathrm{H(c)}$$

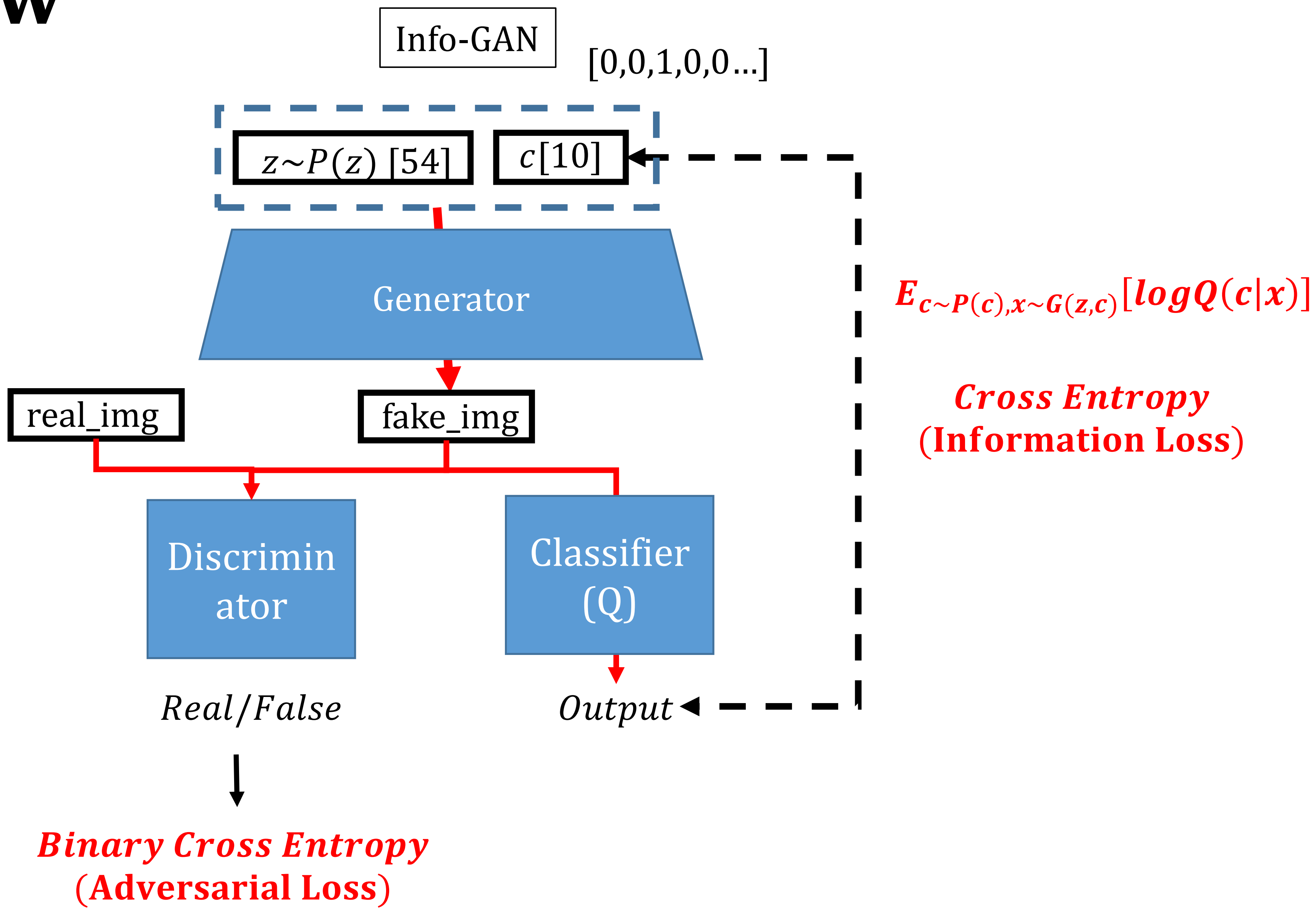
$$\geq E_{x\sim G(z,c)}\left[E_{c'\sim P(c|x)}[logQ(c'|x)]\right] + \mathrm{H(c)}$$

$$\boldsymbol{= E_{c\sim P(c),x\sim G(z,c)}[logQ(c|x)] + \mathrm{H(c)}}$$

We can increase the **lower bound of mutual Information** by other distribution!!

Info-GAN Flow

Take MNIST for example



Info-GAN Loss

Adversarial Loss

$$L_D = -\log\big(\mathrm{D}(I_{real})\big) - \log(1 - D(G(z)))$$

$$\begin{cases} L_G = \log\Big(1 - D\big(G(z)\big)\Big) \\ L_G = -\log\Big(D\big(G(z)\big)\Big) \end{cases}$$

Information Loss

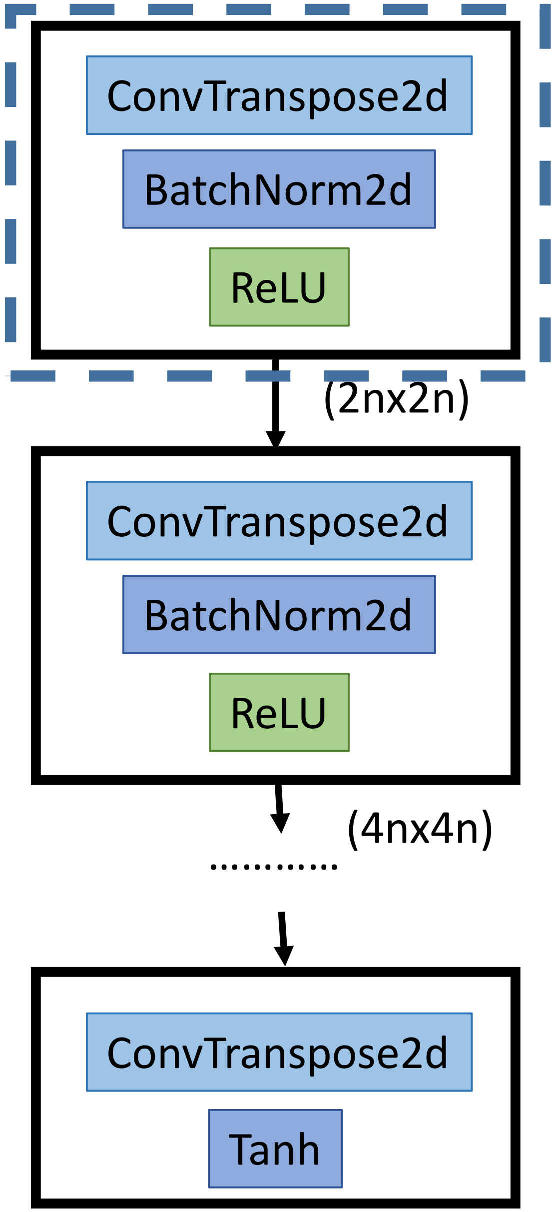
$$L_I(Q, G) = -\log\big(Q(c|G(z, c))\big)$$

Update D $\rightleftharpoons$ Update G/Q

Info-GAN Generator

Example code:
https://github.com/pianomania/infoGAN-pytorch

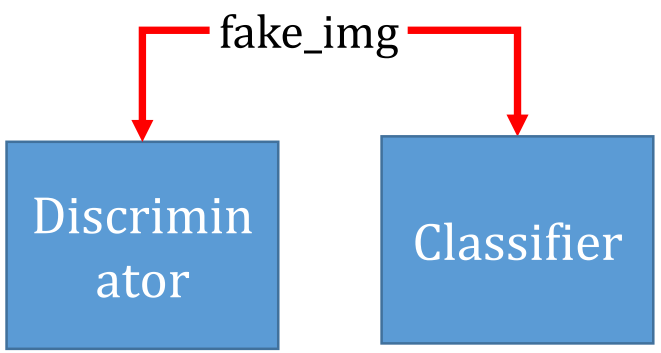
```
_netG(
  (main): Sequential(
    (0): ConvTranspose2d(64, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (2): ReLU(inplace)
    (3): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (5): ReLU(inplace)
    (6): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (8): ReLU(inplace)
    (9): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (10): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True)
    (11): ReLU(inplace)
    (12): ConvTranspose2d(64, 1, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (13): Tanh()
  )
)
```

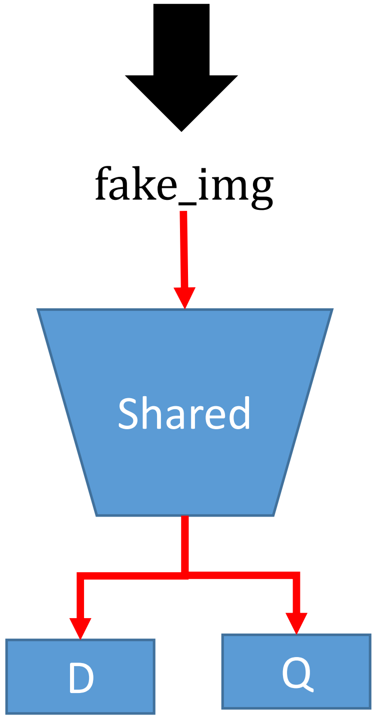


Info-GAN Discriminator

Share weight between D/Q

```
_netD(
  (main): Sequential(
    (0): Conv2d(1, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (1): LeakyReLU(0.2, inplace)
    (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True)
    (4): LeakyReLU(0.2, inplace)
    (5): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True)
    (7): LeakyReLU(0.2, inplace)
    (8): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=False)
    (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True)
    (10): LeakyReLU(0.2, inplace)
  )
  (discriminator): Sequential(
    (0): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): Sigmoid()
  )
  (Q): Sequential(
    (0): Linear(in_features=8192, out_features=100, bias=True)
    (1): ReLU()
    (2): Linear(in_features=100, out_features=10, bias=True)
  )
)
```





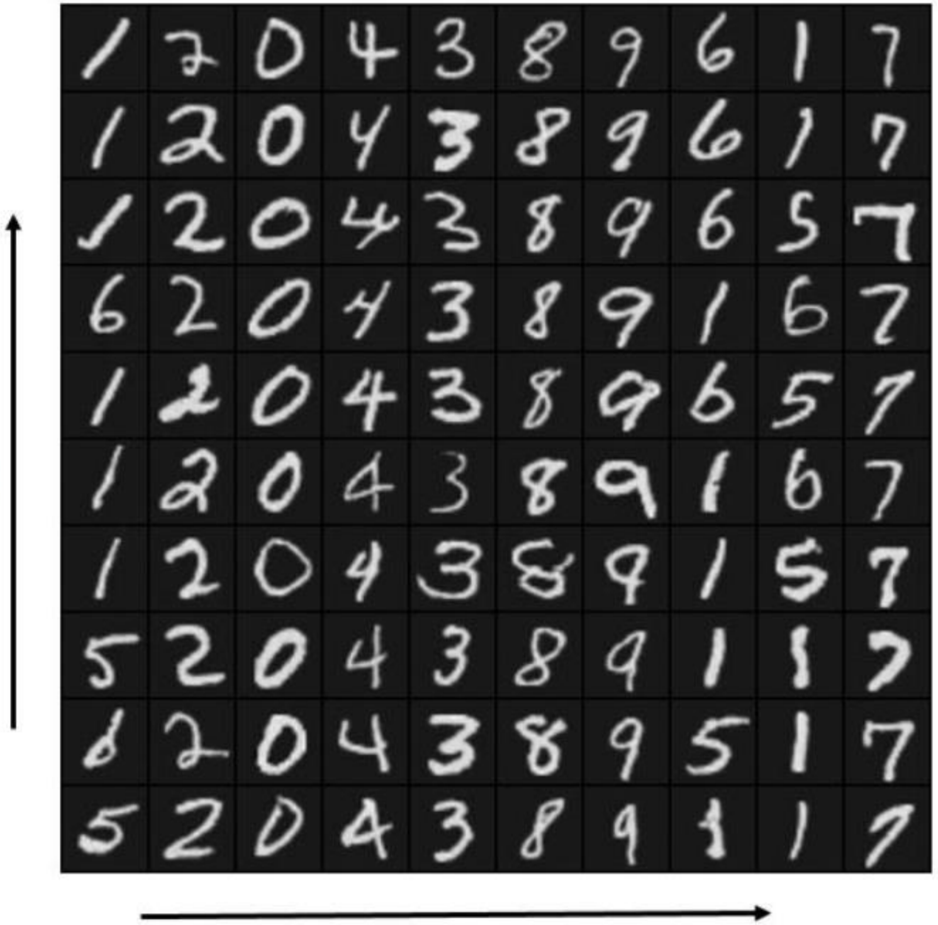
Expected Outputs

Expected Outputs

- Generated images

- More Realistic Output

Different noise with the same one hot vectors



The same noise with different one hot vectors

Hyper-parameters

1. Batch size: 64
2. Learning rate for the discriminator: 2e-4
3. Learning rate for the generator and Q: 1e-3
4. c_size (size of meaningful codes) = 10
5. Total epochs = 80
6. Optimizer: Adam

Info-GAN Report Spec

MNIST

- 1. Introduction (10%)
- 2. Experiment setups: (20%)
  - A. How you implement InfoGAN
    - i. Adversarial loss
    - ii. Maximizing mutual information
  - B. Which loss function of generator you used? What's different?
- 3. Results (30%)
  - A. Results of your samples (shown as in the expected results section)
  - B. Training loss curves
- 4. Discussion (20%)
- 5. Demo (20%)
  - A. Given a label, you have to generate corresponding images
- (*optional*) Bonus: **Facescrub-5** (15%)
  - Results of your samples (10%)
  - Demo (5%)