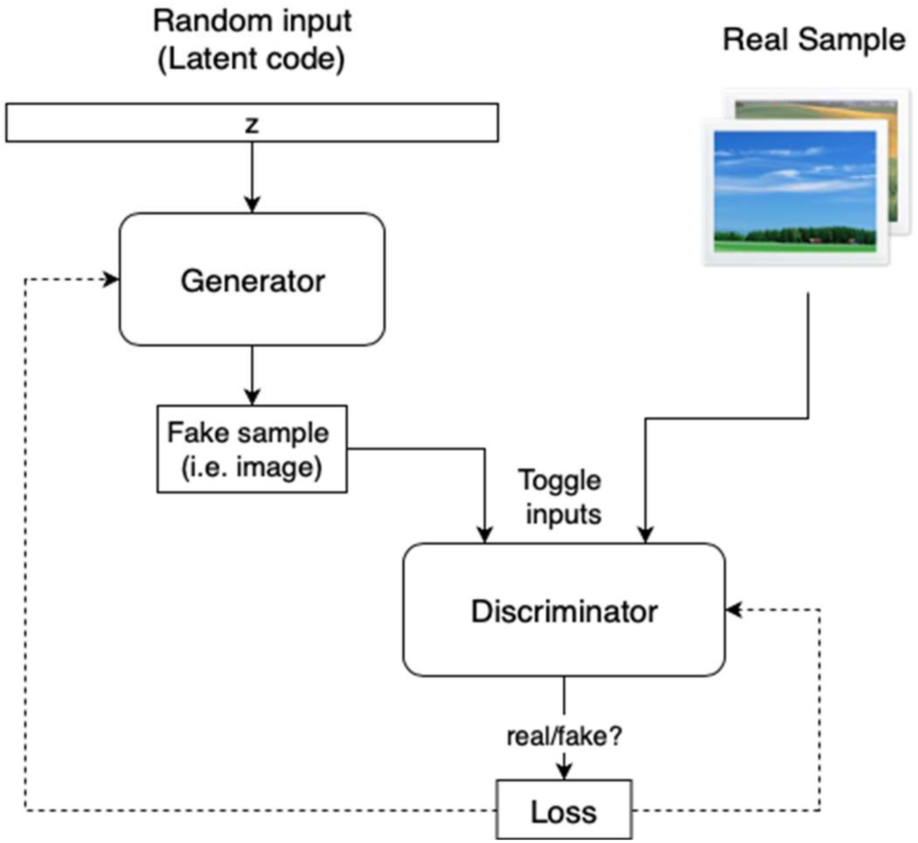
GAN

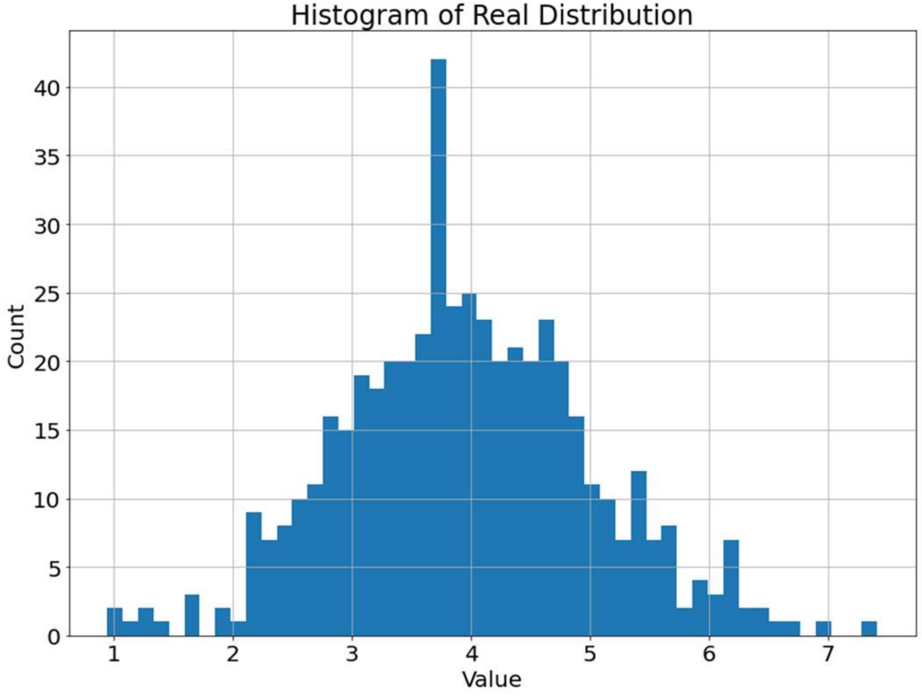
GAN overview

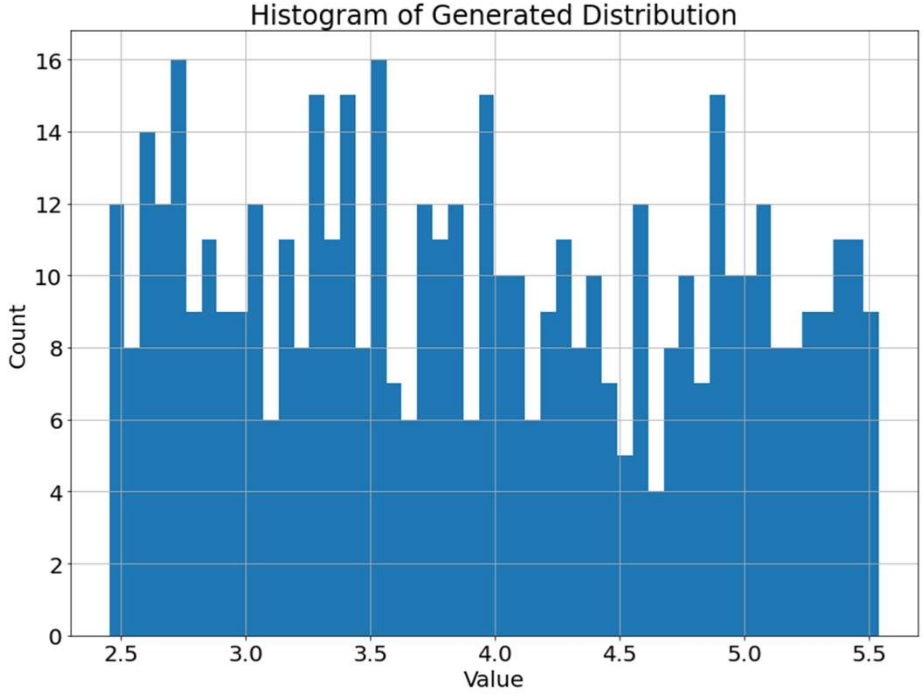


Generator의 분포 학습

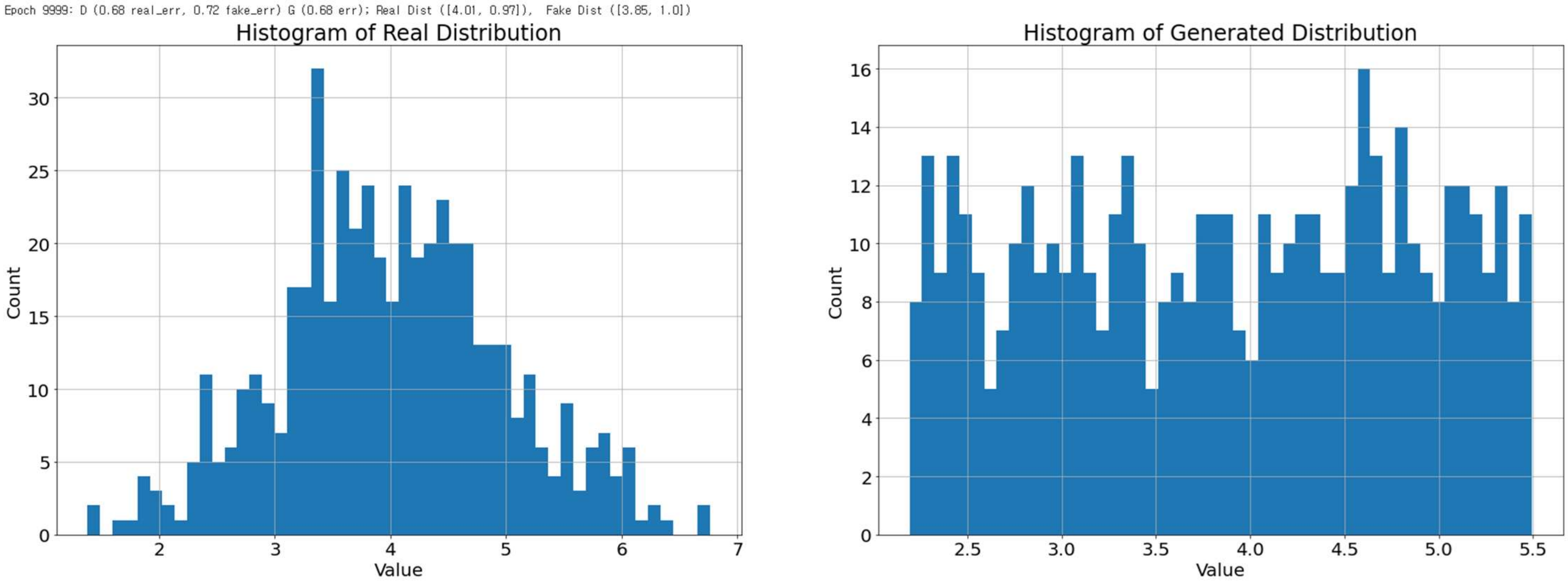
- 평균과 표준편차만을 이용한 방법

Epoch 4999: D (0.7 real_err, 0.7 fake_err) G (0.69 err); Real Dist ([3.97, 1.04]), Fake Dist ([3.98, 0.87])

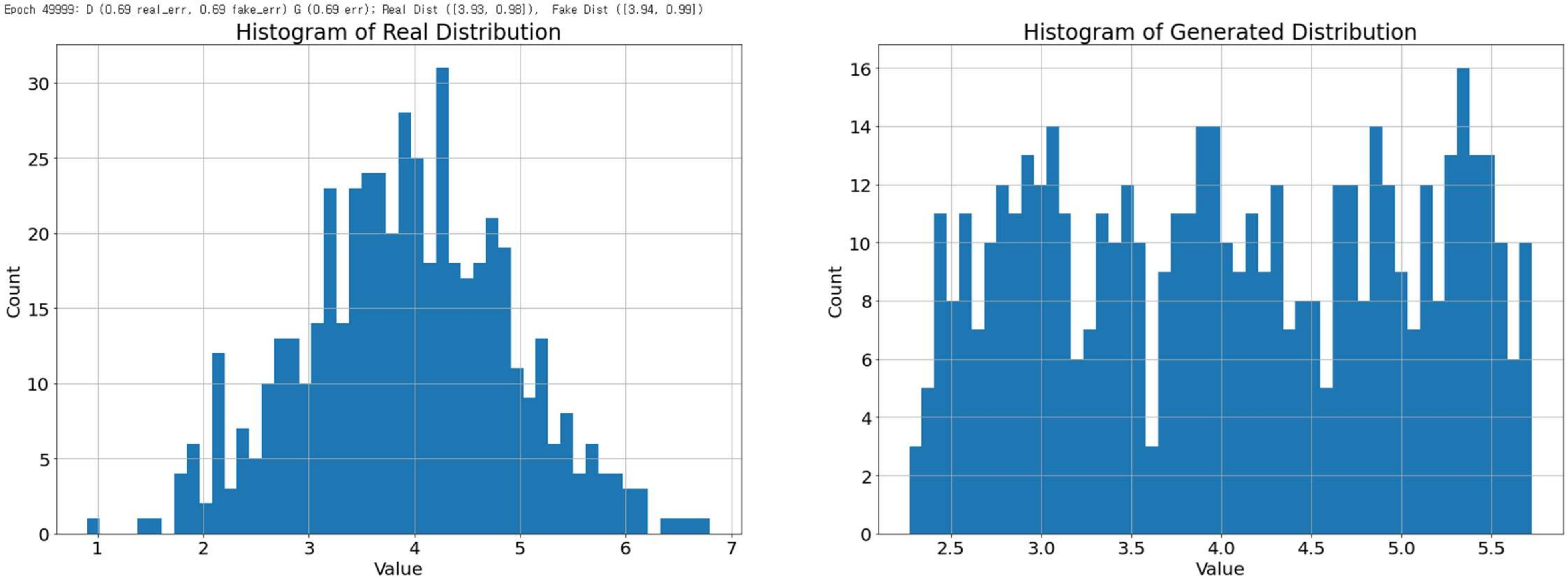




Generator의 분포 학습

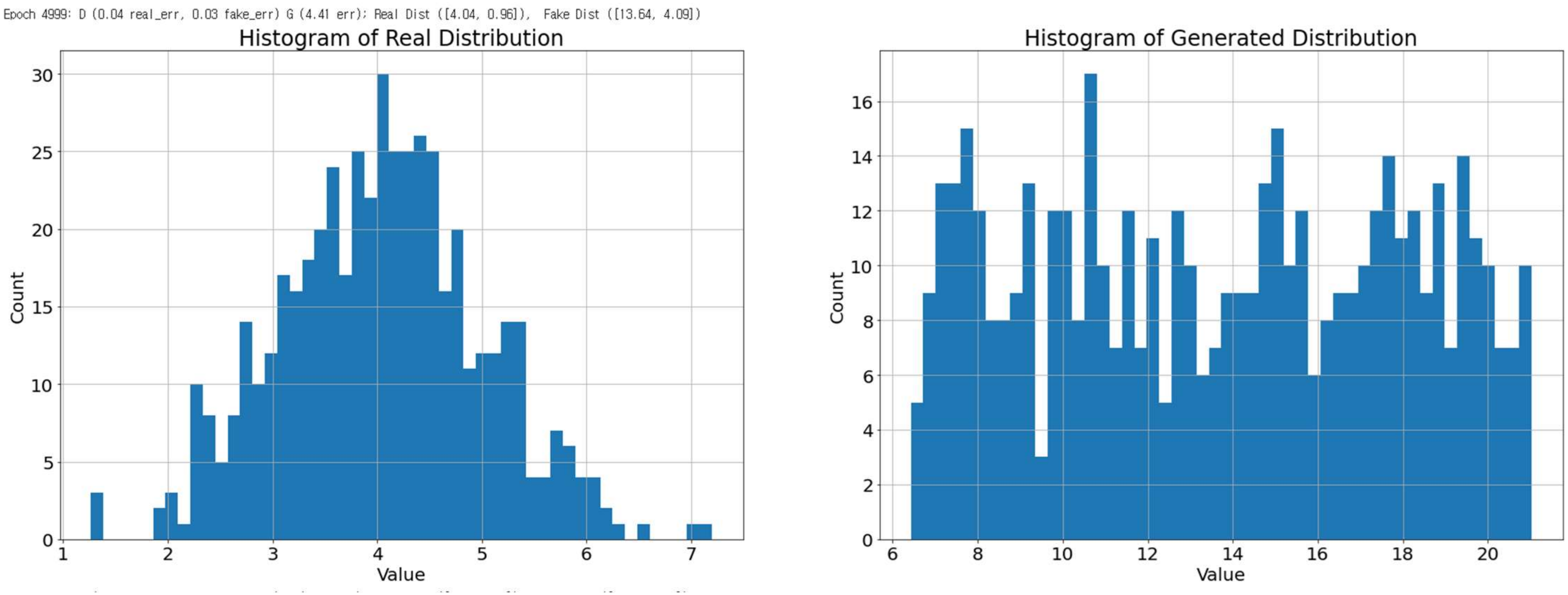


Generator의 분포 학습



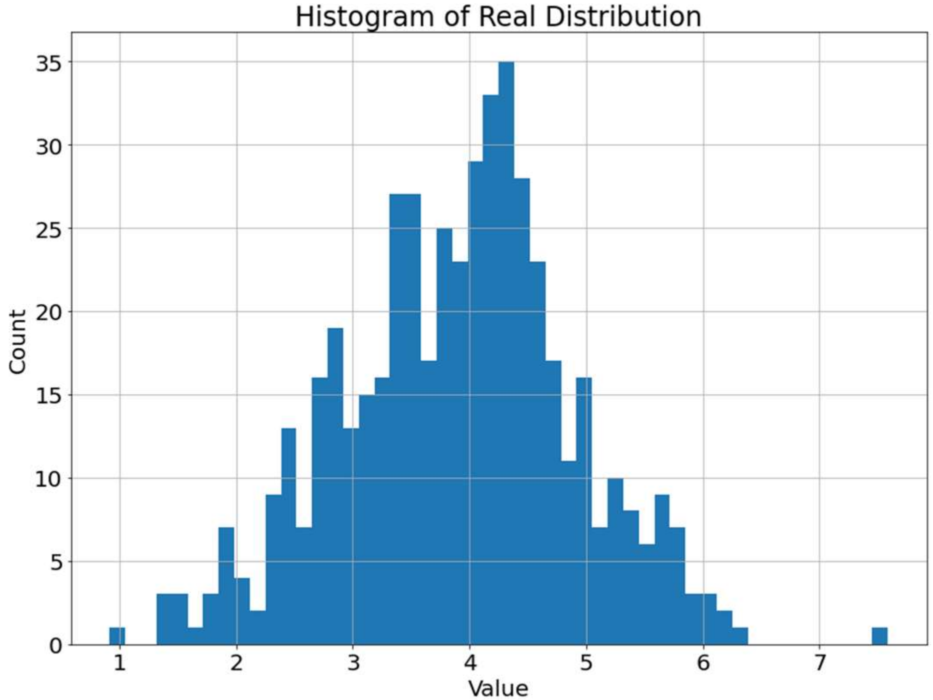
Generator의 분포 학습

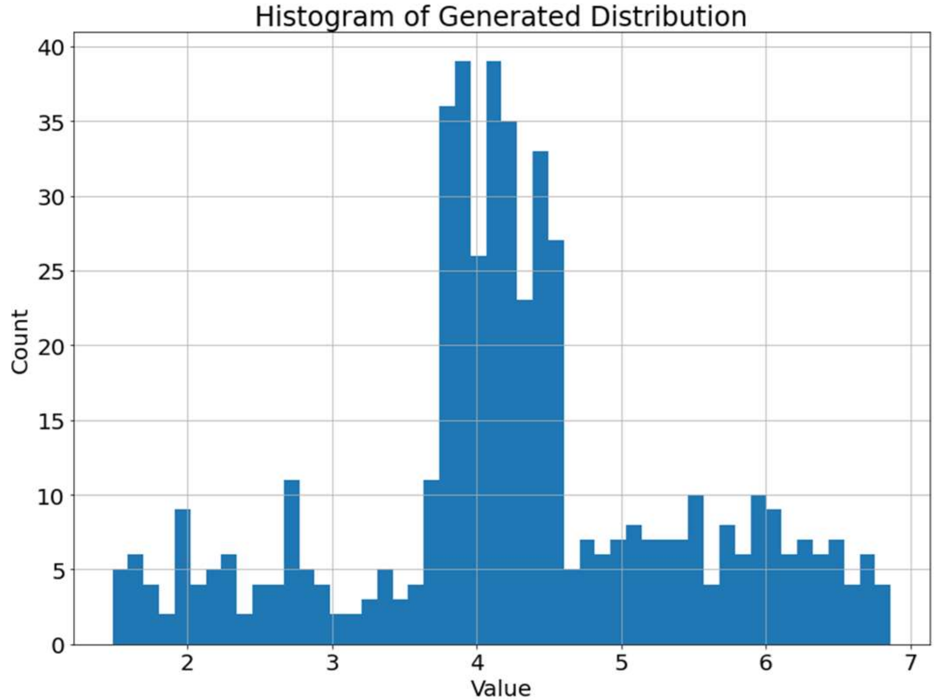
- 평균, 표준편차, 왜도, 첨도를 이용한 방법



Generator의 분포 학습

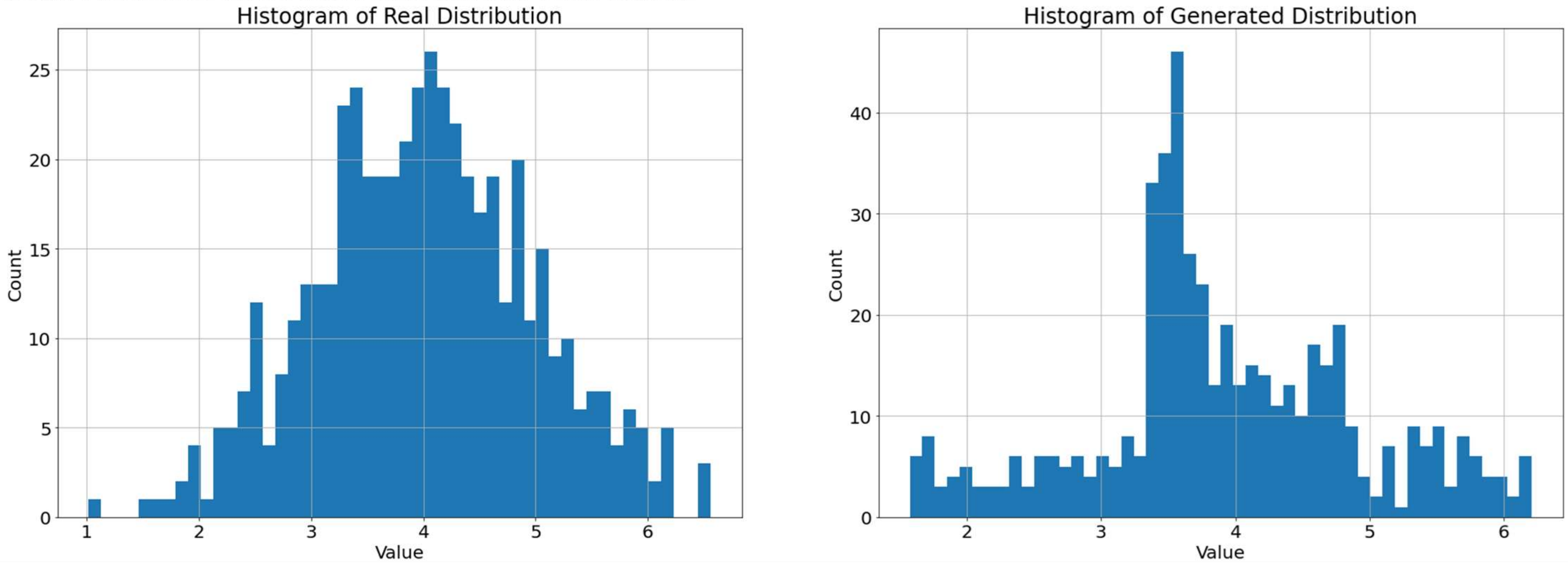
Epoch 9999: D (0.78 real_err, 0.71 fake_err) G (0.72 err); Real Dist ([3.9, 0.99]), Fake Dist ([4.23, 1.18])





Generator의 분포 학습

Epoch 49999: D (0.69 real_err, 0.7 fake_err) G (0.69 err); Real Dist ([4.01, 0.97]), Fake Dist ([4.06, 0.99])

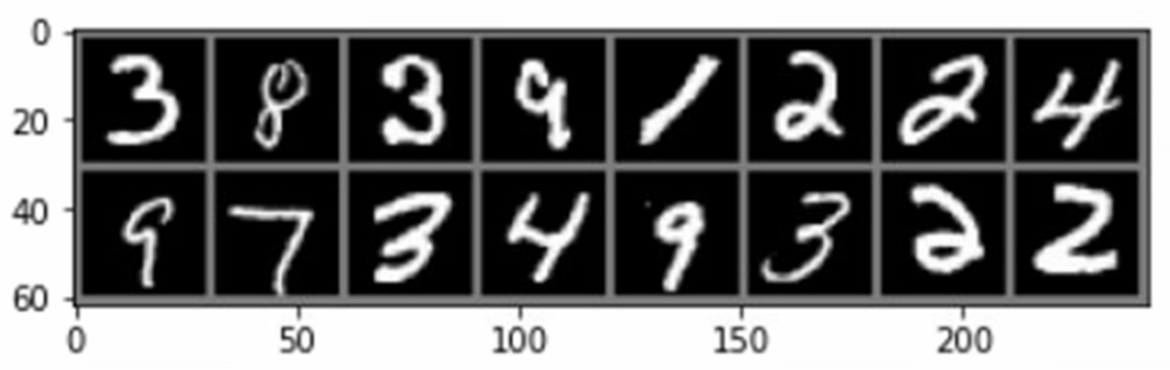


Type of GAN

- **GAN**
- **Deep Convolutional GAN(DCGAN)**
- Conditional GAN
- Info GAN
- Wasserstein GAN(WGAN)
- Attention GAN
- Cycle GAN
- Progressive GAN(PGGAN)
- **Style GAN**

GAN

- MNIST 0~9 까지의 숫자를 필기체 이미지, 라벨 데이터
- 28 x 28 사이즈, 흑백 컬러



GAN

```python
G = nn.Sequential(
    nn.Linear(d_noise, d_hidden),
    nn.ReLU(),
    nn.Dropout(0.1),
    nn.Linear(d_hidden,d_hidden),
    nn.ReLU(),
    nn.Dropout(0.1),
    nn.Linear(d_hidden, 28*28),
    nn.Tanh()
).to(device)
```

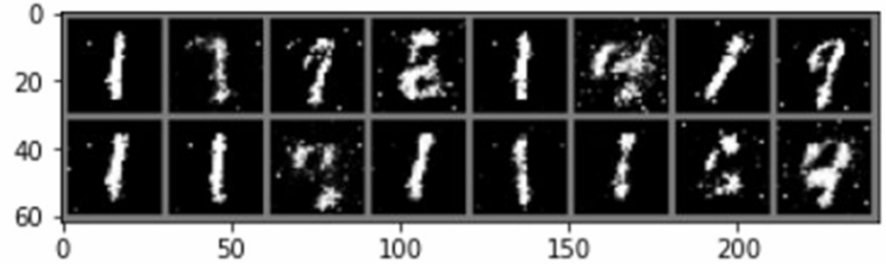
<Generator Modeling>

```python
D = nn.Sequential(
    nn.Linear(28*28, d_hidden),
    nn.LeakyReLU(),
    nn.Dropout(0.1),
    nn.Linear(d_hidden, d_hidden),
    nn.LeakyReLU(),
    nn.Dropout(0.1),
    nn.Linear(d_hidden, 1),
    nn.Sigmoid()
).to(device)
```

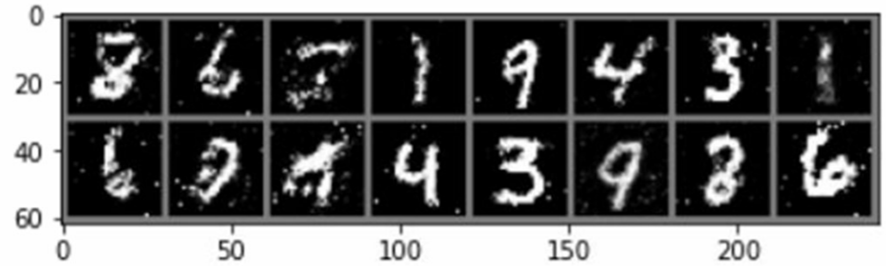
<Discriminator Modeling>

GAN

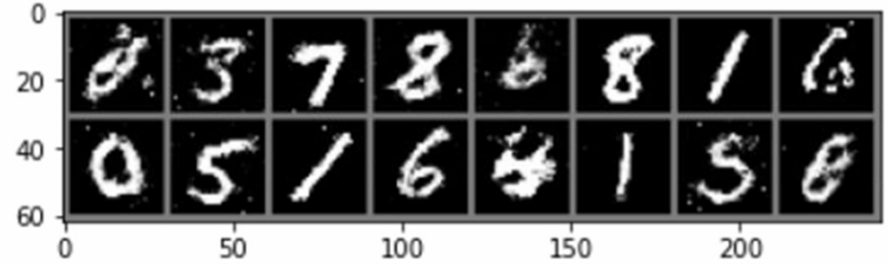
(epoch 50/200) p_real: 0.804420, p_g: 0.284460



(epoch 100/200) p_real: 0.687812, p_g: 0.333965



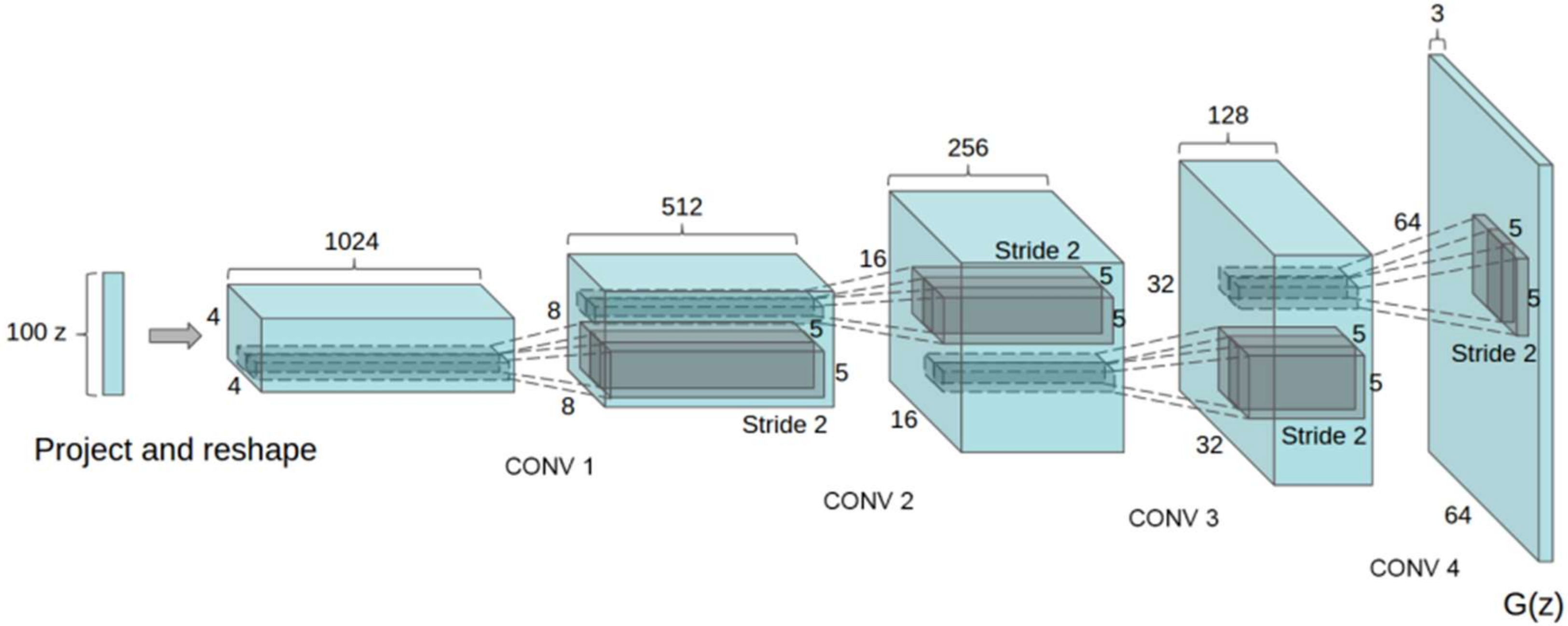
(epoch 150/200) p_real: 0.616051, p_g: 0.316437



(epoch 200/200) p_real: 0.614129, p_g: 0.288346

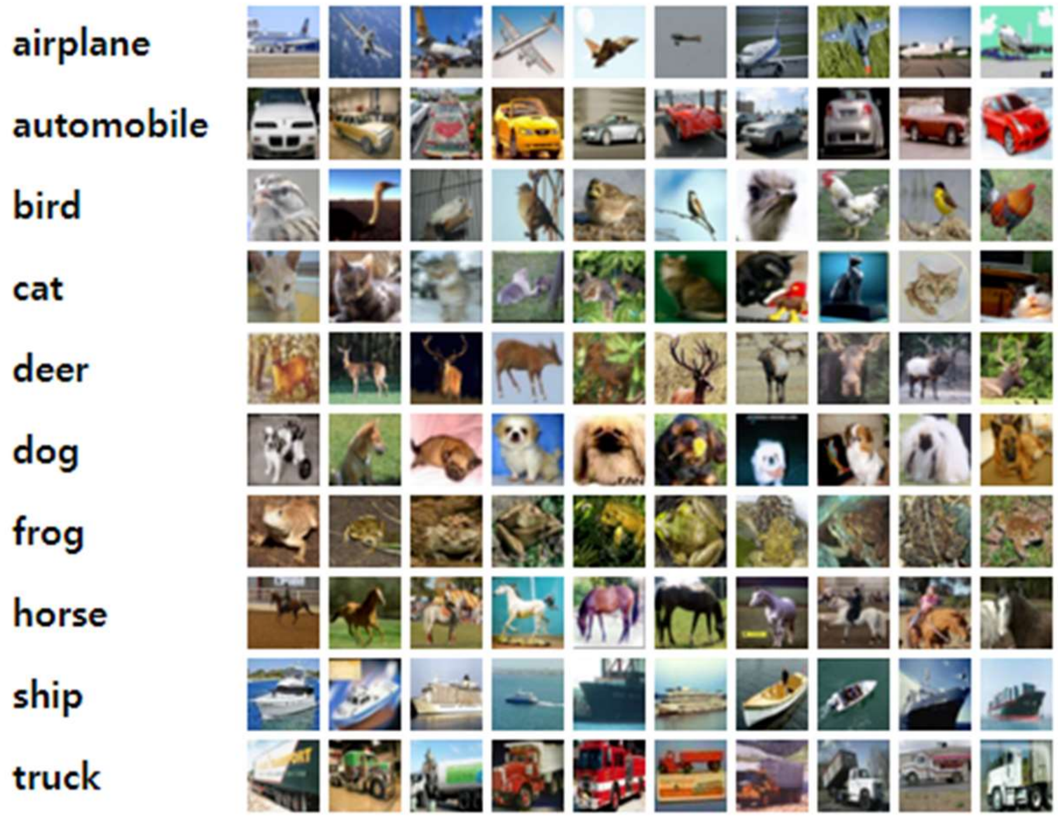
DCGAN

- 생성자와 판별자의 fully-connected에 cnn을 적용
- 조작된 가짜 이미지 생성



DCGAN

- CIFAR10, 10개의 class의 이미지, 라벨 데이터
- 32 x 32 이미지, RGB 컬러



DCGAN

```
_netG(
  (main): Sequential(
    (0): ConvTranspose2d(100, 512, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
    (2): ReLU(inplace=True)
    (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, track_running_stat
s=True)
    (5): ReLU(inplace=True)
    (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, track_running_stat
s=True)
    (8): ReLU(inplace=True)
    (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, track_running_stat
s=True)
    (11): ReLU(inplace=True)
    (12): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1,
1), bias=False)
    (13): Tanh()
  )
)
```

<Generator Modeling>

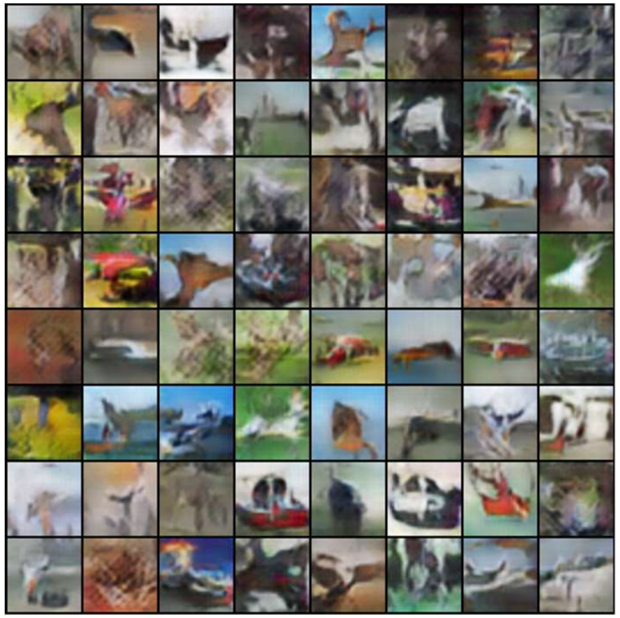
```
_netD(
  (main): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=Fal
se)
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1), bias=F
alse)
    (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stat
s=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (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, track_running_stat
s=True)
    (7): LeakyReLU(negative_slope=0.2, inplace=True)
    (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, track_running_stat
s=True)
    (10): LeakyReLU(negative_slope=0.2, inplace=True)
    (11): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (12): Sigmoid()
  )
)
```

<Discriminator Modeling>

DCGAN

EPOCH = 0

EPOCH = 2

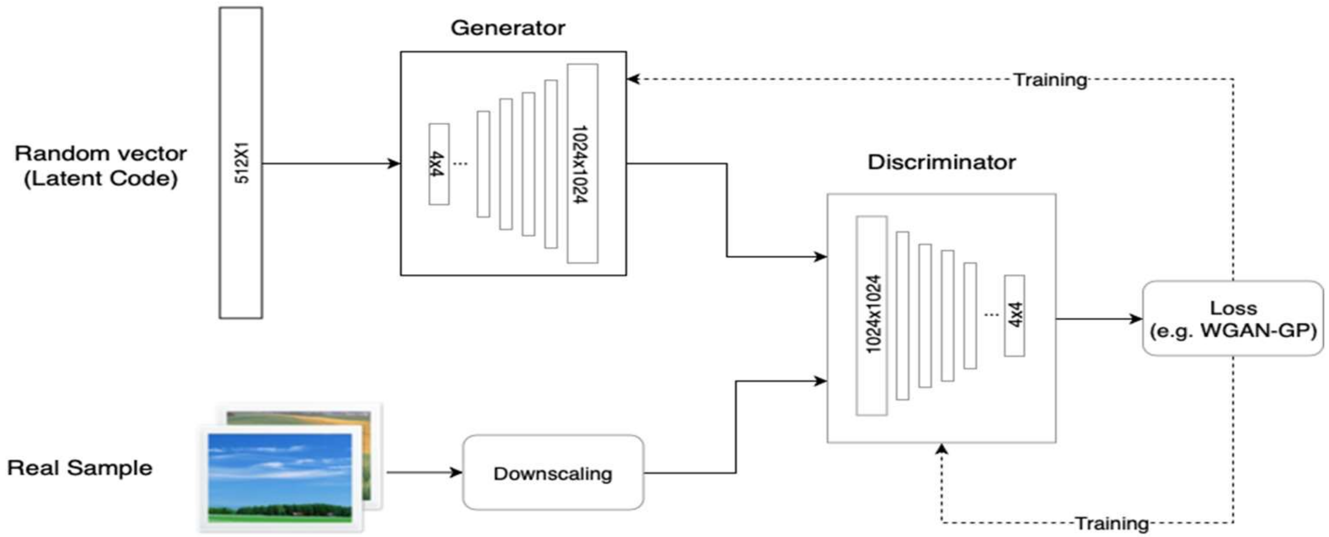


EPOCH = 9

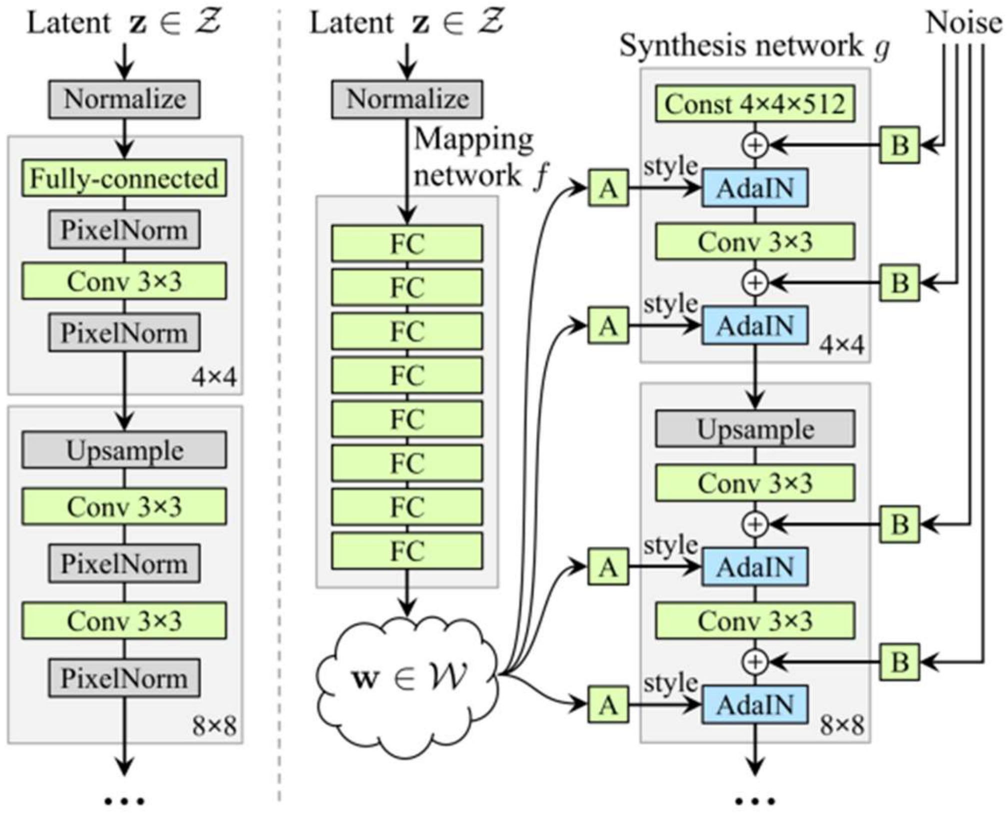
StyleGAN

- A Style-Based Generator Architecture for GANs(StyleGAN)

- Baseline 모델은 PGGAN(Progressive Growing GAN)

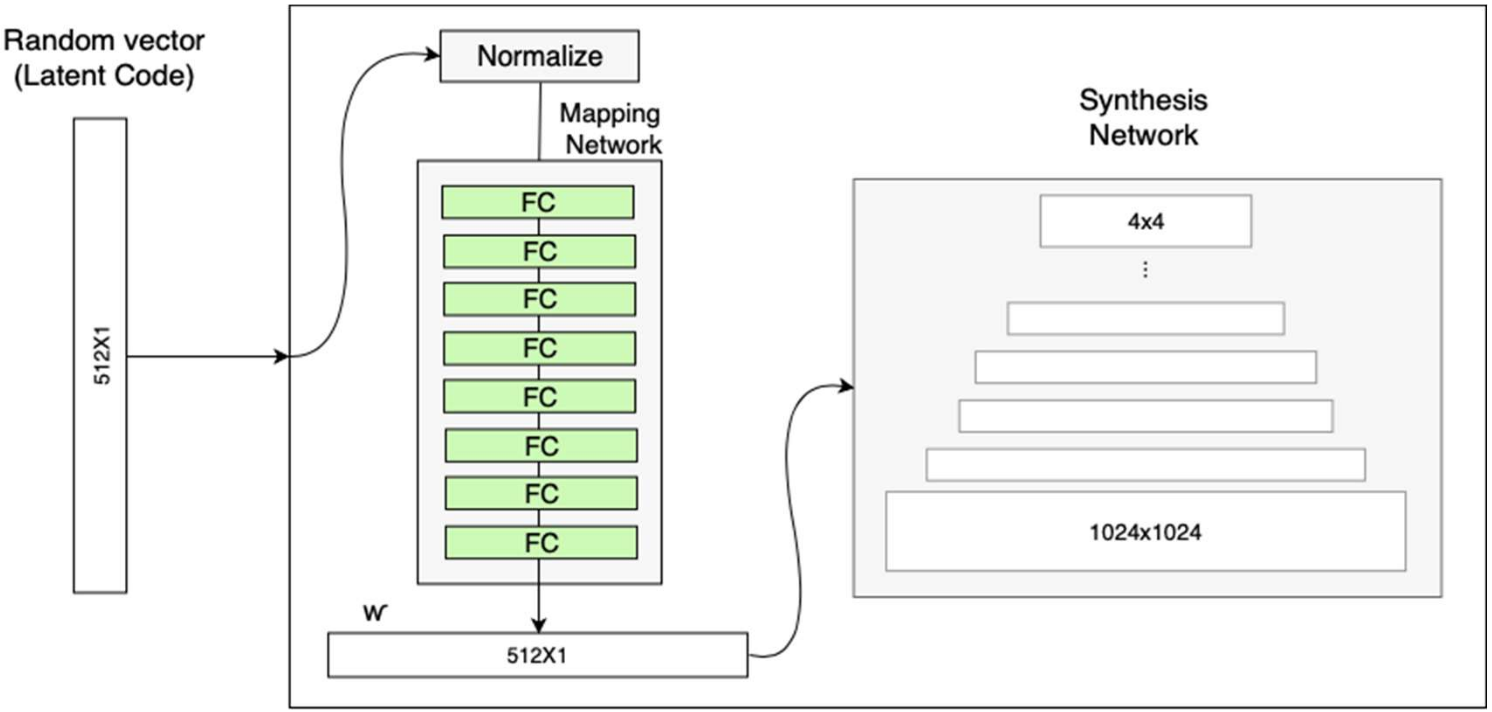


StyleGAN



(a) Traditional (b) Style-based generator

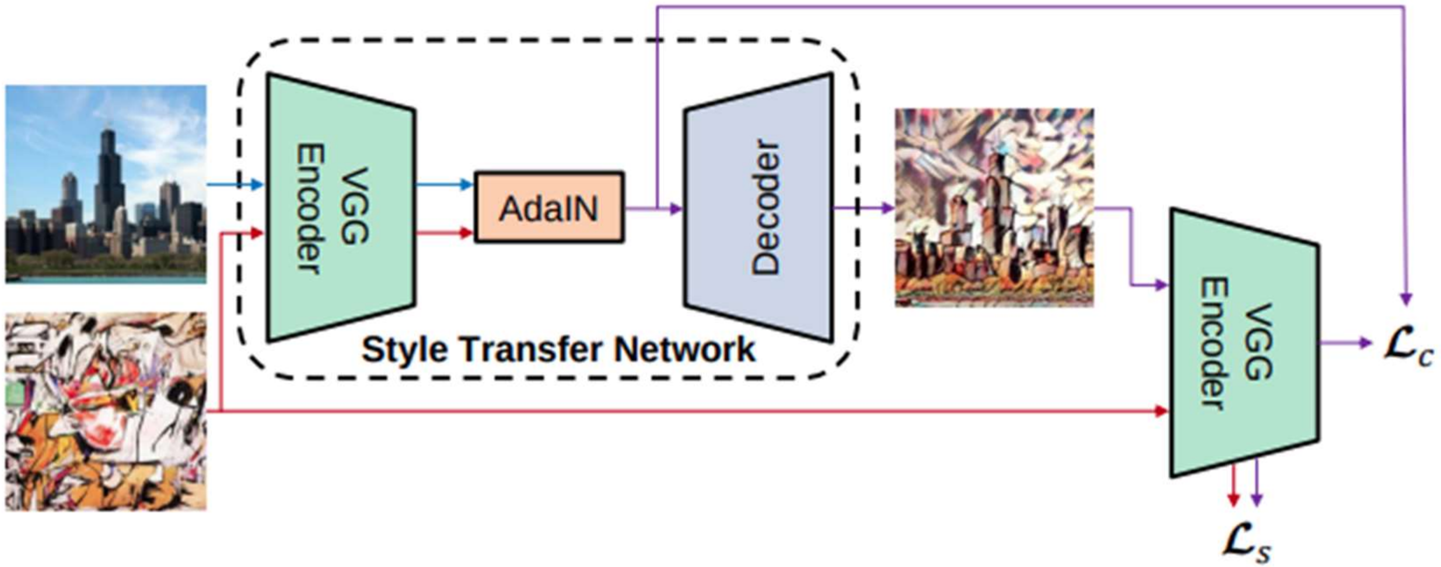
Mapping Network



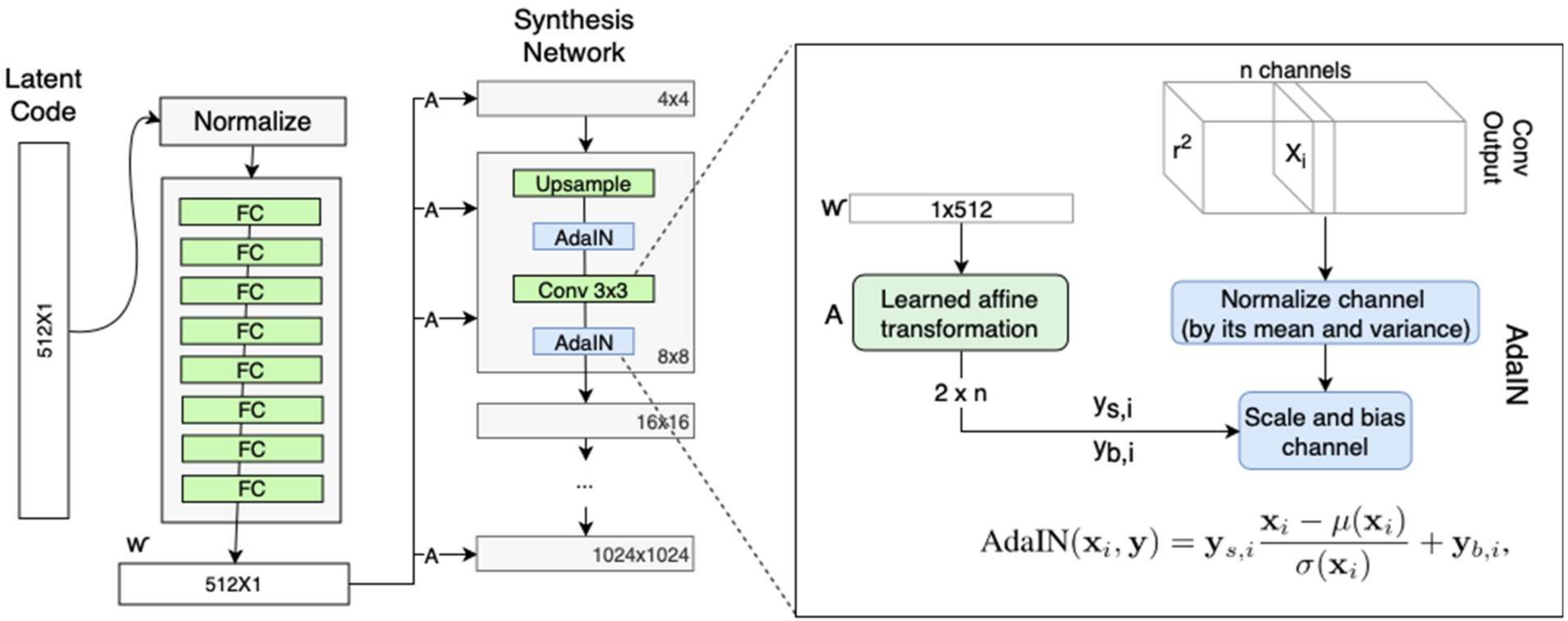
Style Modules

- AdaIN

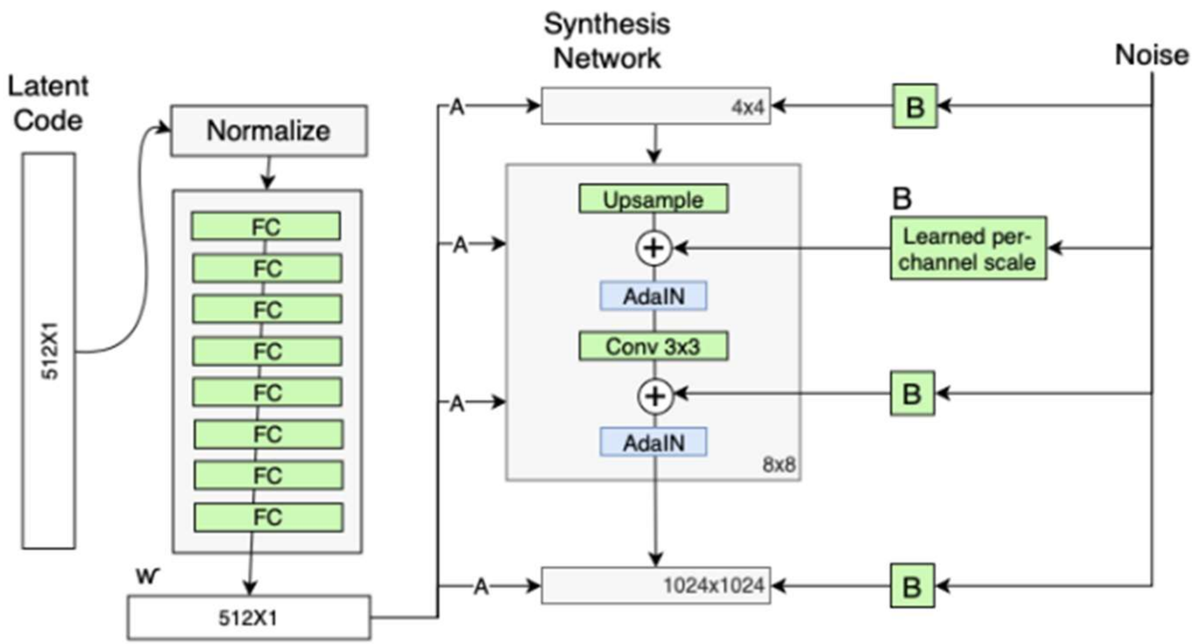
: 다른 데이터로부터 style정보를 가져올 수 있다.



Style Modules(AdaIN)



Stochastic Variation

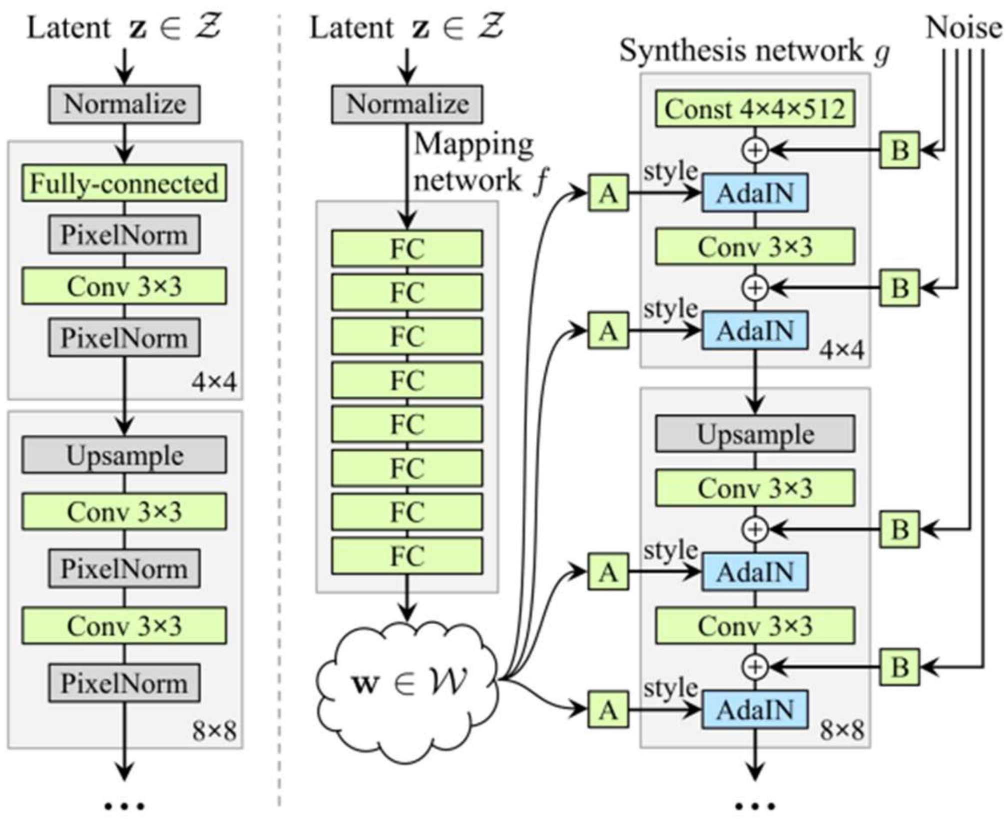


Style Modules와 Stochastic Variation의 차이

• Style Modules
: 얼굴형, 안경의 유무와 같이 high-level 특징을 생성

• Stochastic Variation
: 주근깨, 모공, 곱슬거림 등 low-level 특징 생성

StyleGAN



(a) Traditional (b) Style-based generator

Style Mixing