Generative Adversarial Networks

2017.02.28

최건호

INDEX

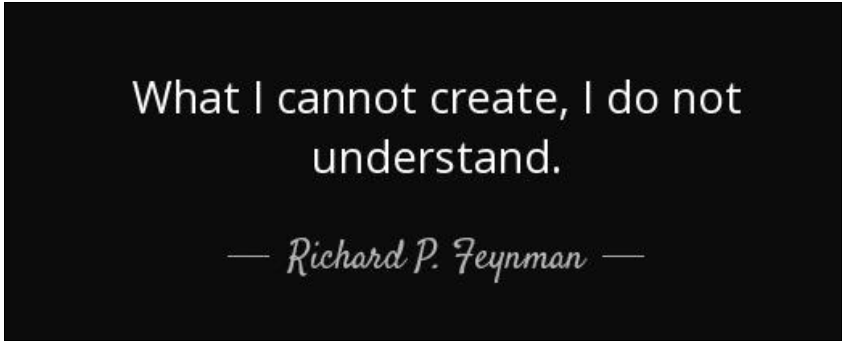
Basic Concept

Basic Concept

Generative Adversarial Networks

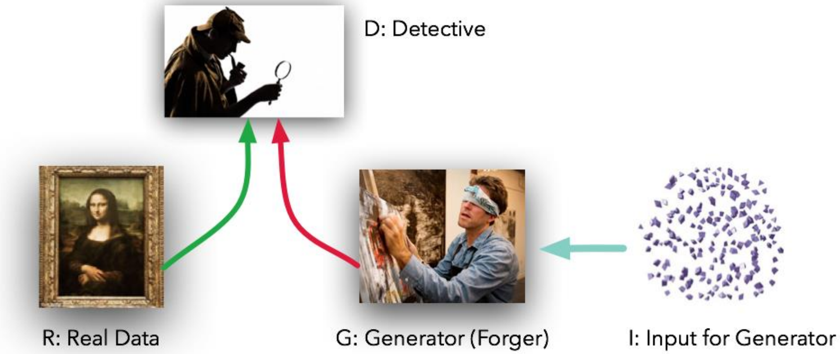
Basic Concept

Generative Adversarial Networks

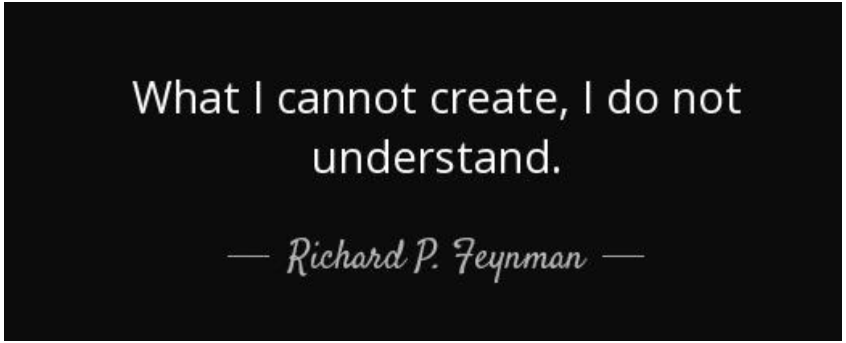


Generative Model은 스스로 데이터를 생성해내는 모델. 어떤 데이터를 보고 분류하는 것 보다 그 데이터를 만들어 -내는 것이 더 어려움.
ex) GAN, VAE

Basic Concept

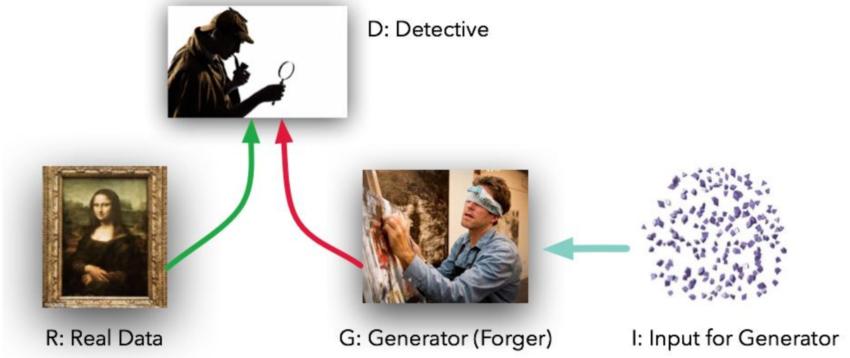


Generative Adversarial Networks



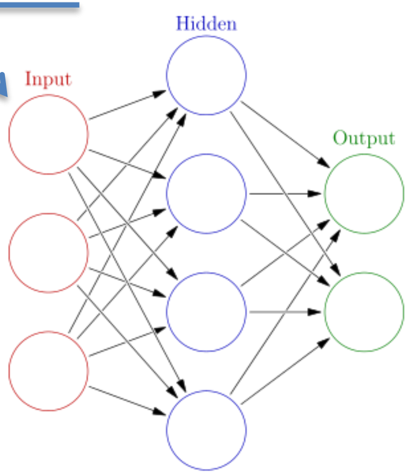
https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f

Basic Concept



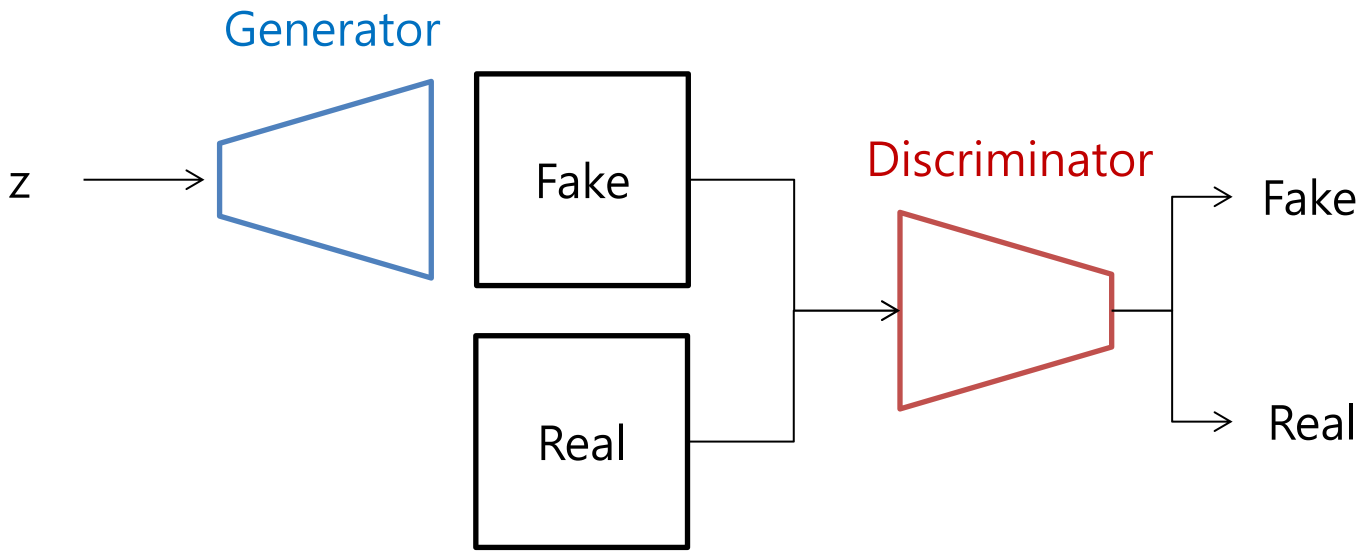
Generative Adversarial Networks

> What I cannot create, I do not understand.
>
> — Richard P. Feynman —

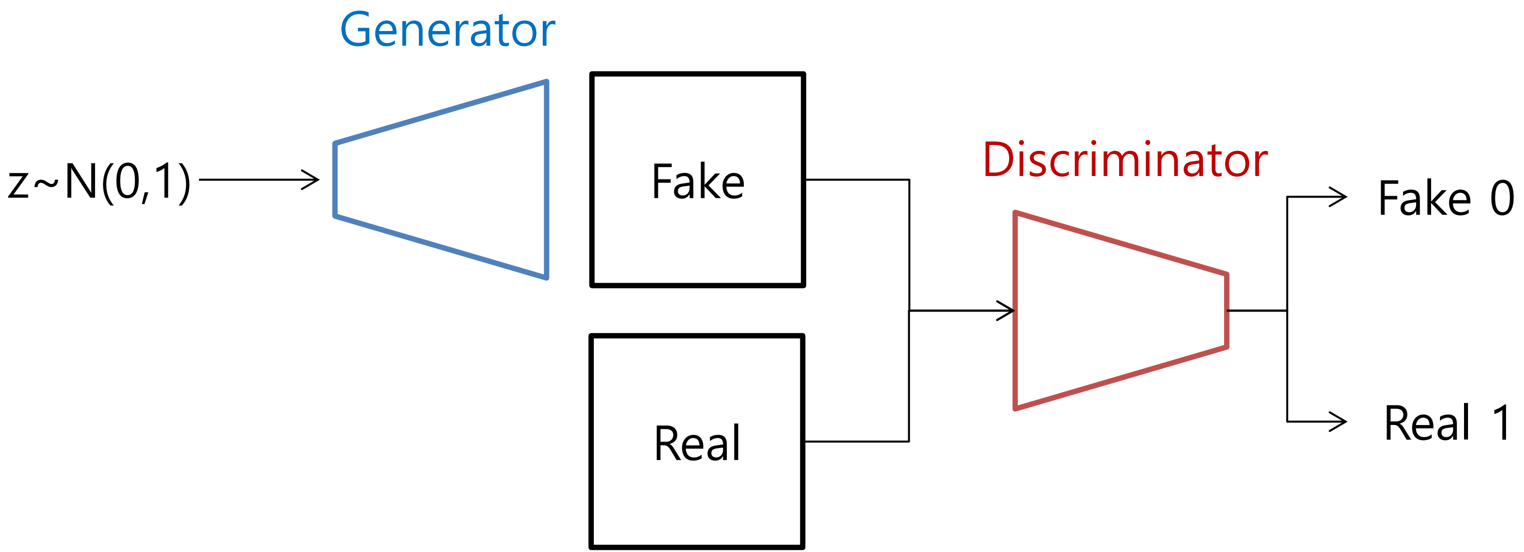


https://medium.com/@devnag/generative-adversarial-networks-gans-in-50-lines-of-code-pytorch-e81b79659e3f

Basic Concept

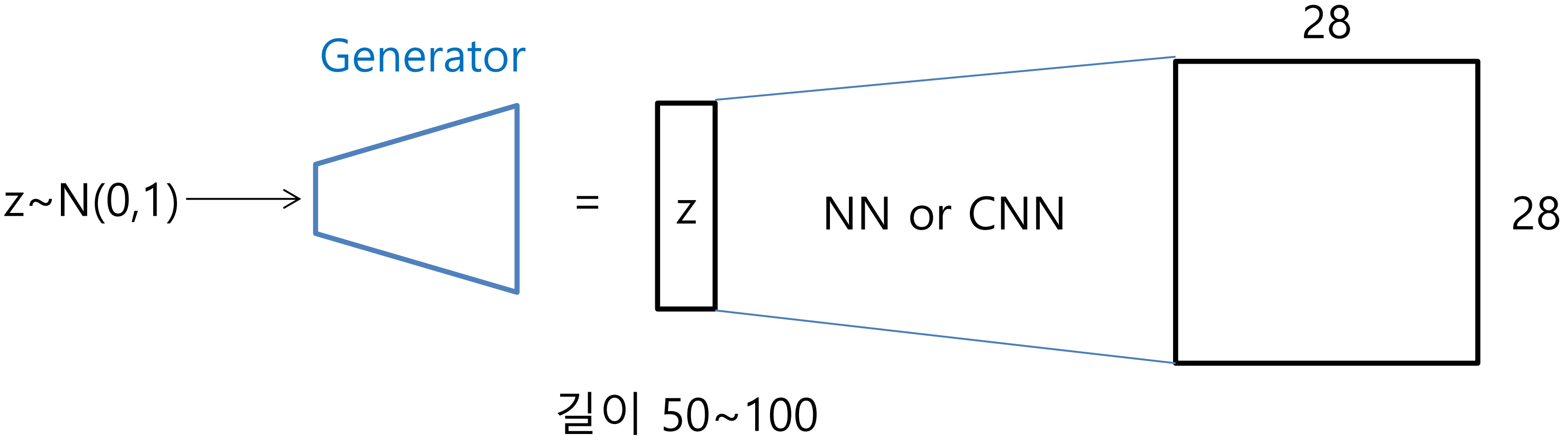


Basic Concept



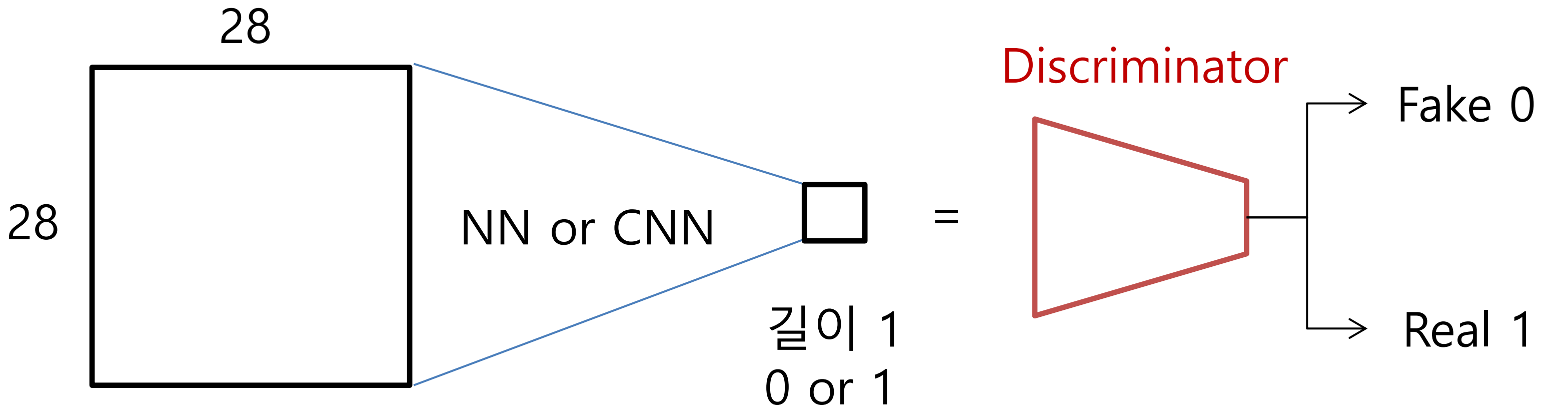
Basic Concept

ex) MNIST



Basic Concept

ex) MNIST



Basic Concept

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Basic Concept

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Generator: 전체 값이 최소가 되는 G를 찾아야 하기 때문에 $1 - D(G(z))$값이 작아야 함. 즉, G가 만들어낸 데이터에 대해 D가 1(real data)라고 판단하도록 학습.

Basic Concept

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Generator: 전체 값이 최소가 되는 G를 찾아야 하기 때문에 $1 - D\big(G(z)\big)$값이 작아야 함. 즉, G가 만들어낸 데이터에 대해 D가 1(real data)라고 판단하도록 학습.

Discriminator: 전체 값이 최대가 되는 D를 찾아야 하기 때문에 진짜 데이터 x -에 대해서는 1(real data)을 출력하고 $D\big(G(z)\big)$에 대해서는 0을 출력하도록 학습.

Basic Concept

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

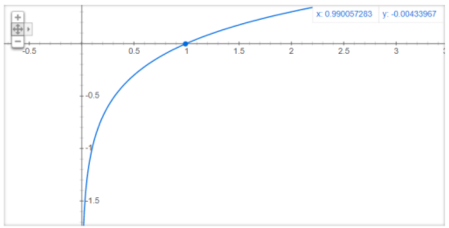
Basic Concept

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

D, G에 대해서 분리

Basic Concept

log(x) 그래프



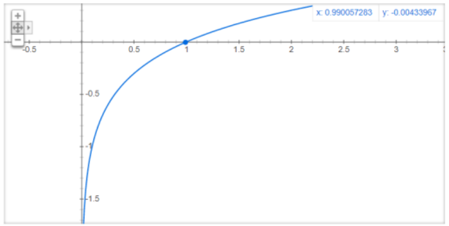
$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

D, G에 대해서 분리

$$\max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Basic Concept

log(x) 그래프



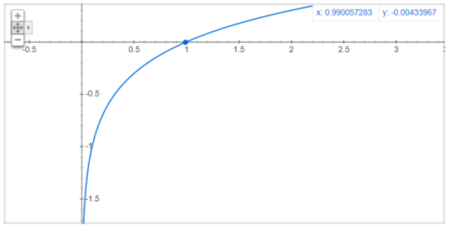
$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

D, G에 대해서 분리

$$\max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Basic Concept

log(x) 그래프



$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

D, G에 대해서 분리

$$\max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log \underline{D(\boldsymbol{x})}] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - \underline{D(G(\boldsymbol{z}))})].$$

1	**0**
0	**1**

Best case: $0+0=0$

Worst case: $-\infty + (-\infty) = -\infty$

Basic Concept

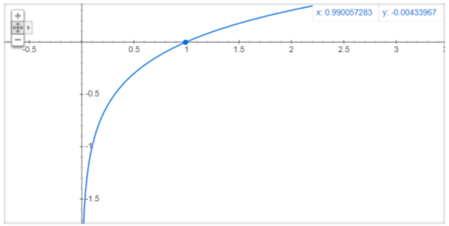
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

D, G에 대해서 분리

$$\min_G V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

Basic Concept

log(x) 그래프



$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

D, G에 대해서 분리

$$\min_G V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - \underline{D(G(\boldsymbol{z}))})].$$

0

1

Best case: -∞

Worst case: 0

Basic Concept

$$\max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

$$\min_G V(D,G) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

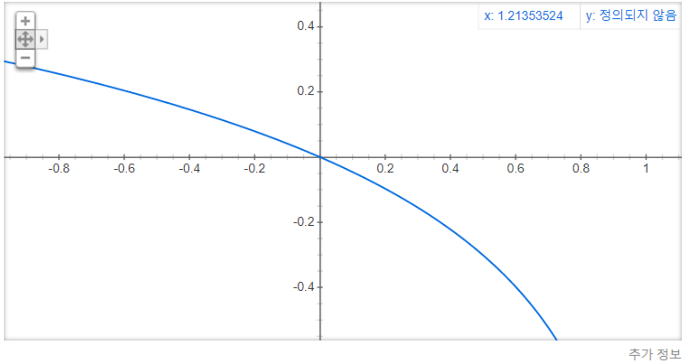
Basic Concept

$$\max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

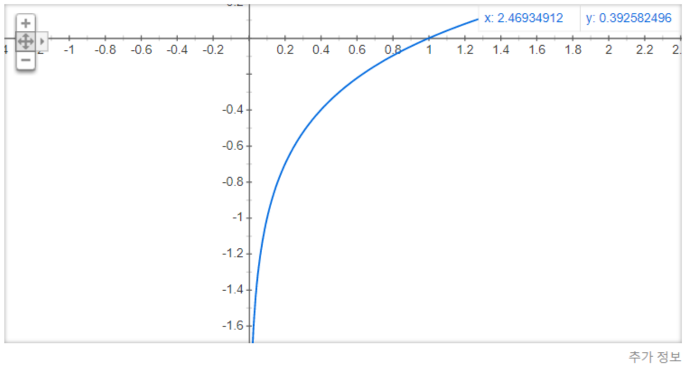
$$\min_G V(D,G) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

사실 $\log\left(1 - \mathrm{D}(\mathrm{G}(\mathrm{z}))\right)$ 를 최소화하는것은 $\log\left(D\left(G(z)\right)\right)$ 를 $maximize$ 하는것과 같음
같은 의미지만 후자의 gradient값들이 더 크기 때문에 학습이 더 잘됨

log(1-x) 그래프



log(x) 그래프



Basic Concept

$$\max_{D} V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log \overset{1}{\underline{D(\boldsymbol{x})}}] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - \overset{0}{\underline{D(G(\boldsymbol{z}))}})].$$

$$\max_{G} V(D,G) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(\underbrace{D(G(\boldsymbol{z}))}_{1})].$$

Basic Concept

$$\max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log \underbrace{D(\boldsymbol{x})}_{}] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - \underbrace{D(G(\boldsymbol{z}))}_{})].$$

$$\overset{1}{D(\boldsymbol{x})} \qquad \overset{0}{D(G(\boldsymbol{z}))}$$

$$\max_G V(D,G) = \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(\underbrace{D(G(\boldsymbol{z}))}_{1})].$$

label이 0 또는 1이기 때문에 구현할 때는 binary cross entropy loss 사용

class **torch.nn.BCELoss**(*weight=None, size_average=True*) [source]

Creates a criterion that measures the Binary Cross Entropy between the target and the output:

$$loss(o,t) = -1/n \sum_i (t[i] * log(o[i]) + (1 - t[i]) * log(1 - o[i]))$$

Basic Concept

학습이 잘 되면 Discriminator의 결과값은 0.5로 수렴하게 됨
결과값이 0.5라는 것은 진짜인지 아닌지 구별이 불가능한 상태

Basic Concept

학습이 잘 되면 Discriminator의 결과값은 0.5로 수렴하게 됨
결과값이 0.5라는 것은 진짜인지 아닌지 구별이 불가능한 상태

하지만 이 값만으로는 Generator가 잘 학습해서 속이는 것인지
아니면 Discriminator가 아직 구분을 잘 못하는 것인지 알 수 없음

Basic Concept

학습이 잘 되면 Discriminator의 결과값은 0.5로 수렴하게 됨
결과값이 0.5라는 것은 진짜인지 아닌지 구별이 불가능한 상태

하지만 이 값만으로는 Generator가 잘 학습해서 속이는 것인지
아니면 Discriminator가 아직 구분을 잘 못하는 것인지 알 수 없음

아직까지는 general metric 이 없어서 눈 discriminator가 사용됨

Basic Concept

```python
class Generator(nn.Module):
    def __init__(self):
        super(Generator,self).__init__()
        self.layer1 = nn.Linear(100,7*7*256)
        self.layer2 = nn.Sequential(OrderedDict([
                ('conv1', nn.ConvTranspose2d(256,128,3,2,1,1)),
                ('relu1', nn.LeakyReLU()),
                ('bn1', nn.BatchNorm2d(128)),
                ('conv2', nn.ConvTranspose2d(128,64,3,1,1)),
                ('relu2', nn.LeakyReLU()),
                ('bn2', nn.BatchNorm2d(64))
            ]))
        self.layer3 = nn.Sequential(OrderedDict([
                ('conv3',nn.ConvTranspose2d(64,16,3,1,1)),
                ('relu3',nn.LeakyReLU()),
                ('bn3',nn.BatchNorm2d(16)),
                ('conv4',nn.ConvTranspose2d(16,1,3,2,1,1)),
                ('relu4',nn.LeakyReLU())
            ]))

    def forward(self,z):
        out = self.layer1(z)
        out = out.view(batch_size//num_gpus,256,7,7)
        out = self.layer2(out)
        out = self.layer3(out)
        return out
```

Basic Concept

size 100인 Noise z에서
7*7*256 크기로 늘림

```python
class Generator(nn.Module):
    def __init__(self):
        super(Generator,self).__init__()
        self.layer1 = nn.Linear(100,7*7*256)
        self.layer2 = nn.Sequential(OrderedDict([
                ('conv1', nn.ConvTranspose2d(256,128,3,2,1,1)),
                ('relu1', nn.LeakyReLU()),
                ('bn1', nn.BatchNorm2d(128)),
                ('conv2', nn.ConvTranspose2d(128,64,3,1,1)),
                ('relu2', nn.LeakyReLU()),
                ('bn2', nn.BatchNorm2d(64))
            ]))
        self.layer3 = nn.Sequential(OrderedDict([
                ('conv3',nn.ConvTranspose2d(64,16,3,1,1)),
                ('relu3',nn.LeakyReLU()),
                ('bn3',nn.BatchNorm2d(16)),
                ('conv4',nn.ConvTranspose2d(16,1,3,2,1,1)),
                ('relu4',nn.LeakyReLU())
            ]))

    def forward(self,z):
        out = self.layer1(z)
        out = out.view(batch_size//num_gpus,256,7,7)
        out = self.layer2(out)
        out = self.layer3(out)
        return out
```

Basic Concept

size 100인 Noise z에서
7*7*256 크기로 늘림

Autoencoder였으면 decoder
부분에 해당하는 generator

```
class Generator(nn.Module):
    def __init__(self):
        super(Generator,self).__init__()
        self.layer1 = nn.Linear(100,7*7*256)
        self.layer2 = nn.Sequential(OrderedDict([
                ('conv1', nn.ConvTranspose2d(256,128,3,2,1,1)),
                ('relu1', nn.LeakyReLU()),
                ('bn1', nn.BatchNorm2d(128)),
                ('conv2', nn.ConvTranspose2d(128,64,3,1,1)),
                ('relu2', nn.LeakyReLU()),
                ('bn2', nn.BatchNorm2d(64))
            ]))
        self.layer3 = nn.Sequential(OrderedDict([
                ('conv3',nn.ConvTranspose2d(64,16,3,1,1)),
                ('relu3',nn.LeakyReLU()),
                ('bn3',nn.BatchNorm2d(16)),
                ('conv4',nn.ConvTranspose2d(16,1,3,2,1,1)),
                ('relu4',nn.LeakyReLU())
            ]))

    def forward(self,z):
        out = self.layer1(z)
        out = out.view(batch_size//num_gpus,256,7,7)
        out = self.layer2(out)
        out = self.layer3(out)
        return out
```

Basic Concept

size 100인 Noise z에서
7*7*256 크기로 늘림

Autoencoder였으면 decoder
부분에 해당하는 generator

nn.Sequential 안에 layer 이름을
지정할 수 있게 OrderedDict를
사용함

```
class Generator(nn.Module):
    def __init__(self):
        super(Generator,self).__init__()
        self.layer1 = nn.Linear(100,7*7*256)
        self.layer2 = nn.Sequential(OrderedDict([
                ('conv1', nn.ConvTranspose2d(256,128,3,2,1,1)),
                ('relu1', nn.LeakyReLU()),
                ('bn1', nn.BatchNorm2d(128)),
                ('conv2', nn.ConvTranspose2d(128,64,3,1,1)),
                ('relu2', nn.LeakyReLU()),
                ('bn2', nn.BatchNorm2d(64))
            ]))
        self.layer3 = nn.Sequential(OrderedDict([
                ('conv3',nn.ConvTranspose2d(64,16,3,1,1)),
                ('relu3',nn.LeakyReLU()),
                ('bn3',nn.BatchNorm2d(16)),
                ('conv4',nn.ConvTranspose2d(16,1,3,2,1,1)),
                ('relu4',nn.LeakyReLU())
            ]))

    def forward(self,z):
        out = self.layer1(z)
        out = out.view(batch_size//num_gpus,256,7,7)
        out = self.layer2(out)
        out = self.layer3(out)
        return out
```

Basic Concept

size 100인 Noise z에서
7*7*256 크기로 늘림

Autoencoder였으면 decoder
부분에 해당하는 generator

nn.Sequential 안에 layer 이름을
지정할 수 있게 OrderedDict를
사용함

형태에 관한 건 뒤에 DCGAN에서
추가적인 설명이 있음

```python
class Generator(nn.Module):
    def __init__(self):
        super(Generator,self).__init__()
        self.layer1 = nn.Linear(100,7*7*256)
        self.layer2 = nn.Sequential(OrderedDict([
                ('conv1', nn.ConvTranspose2d(256,128,3,2,1,1)),
                ('relu1', nn.LeakyReLU()),
                ('bn1', nn.BatchNorm2d(128)),
                ('conv2', nn.ConvTranspose2d(128,64,3,1,1)),
                ('relu2', nn.LeakyReLU()),
                ('bn2', nn.BatchNorm2d(64))
            ]))
        self.layer3 = nn.Sequential(OrderedDict([
                ('conv3',nn.ConvTranspose2d(64,16,3,1,1)),
                ('relu3',nn.LeakyReLU()),
                ('bn3',nn.BatchNorm2d(16)),
                ('conv4',nn.ConvTranspose2d(16,1,3,2,1,1)),
                ('relu4',nn.LeakyReLU())
            ]))

    def forward(self,z):
        out = self.layer1(z)
        out = out.view(batch_size//num_gpus,256,7,7)
        out = self.layer2(out)
        out = self.layer3(out)
        return out
```

Basic Concept

```python
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator,self).__init__()
        self.layer1 = nn.Sequential(OrderedDict([
                ('conv1',nn.Conv2d(1,16,3,padding=1)),   # batch x 16 x 28 x 28
                ('relu1',nn.LeakyReLU()),
                ('bn1',nn.BatchNorm2d(16)),
                ('conv2',nn.Conv2d(16,32,3,padding=1)),  # batch x 32 x 28 x 28
                ('relu2',nn.LeakyReLU()),
                ('bn2',nn.BatchNorm2d(32)),
                ('max1',nn.MaxPool2d(2,2))   # batch x 32 x 14 x 14
        ]))
        self.layer2 = nn.Sequential(OrderedDict([
                ('conv3',nn.Conv2d(32,64,3,padding=1)),  # batch x 64 x 14 x 14
                ('relu3',nn.LeakyReLU()),
                ('bn3',nn.BatchNorm2d(64)),
                ('max2',nn.MaxPool2d(2,2)),
                ('conv4',nn.Conv2d(64,128,3,padding=1)),  # batch x 128 x 7 x 7
                ('relu4',nn.LeakyReLU())
        ]))
        self.fc = nn.Sequential(
                nn.Linear(128*7*7,1),
                nn.Sigmoid()
        )

    def forward(self,x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.view(batch_size//num_gpus, -1)
        out = self.fc(out)
        return out
```

Basic Concept

Discriminator도 그냥 일반적인 형태.

```python
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator,self).__init__()
        self.layer1 = nn.Sequential(OrderedDict([
                ('conv1',nn.Conv2d(1,16,3,padding=1)),   # batch x 16 x 28 x 28
                ('relu1',nn.LeakyReLU()),
                ('bn1',nn.BatchNorm2d(16)),
                ('conv2',nn.Conv2d(16,32,3,padding=1)),  # batch x 32 x 28 x 28
                ('relu2',nn.LeakyReLU()),
                ('bn2',nn.BatchNorm2d(32)),
                ('max1',nn.MaxPool2d(2,2))   # batch x 32 x 14 x 14
        ]))
        self.layer2 = nn.Sequential(OrderedDict([
                ('conv3',nn.Conv2d(32,64,3,padding=1)),  # batch x 64 x 14 x 14
                ('relu3',nn.LeakyReLU()),
                ('bn3',nn.BatchNorm2d(64)),
                ('max2',nn.MaxPool2d(2,2)),
                ('conv4',nn.Conv2d(64,128,3,padding=1)),  # batch x 128 x 7 x 7
                ('relu4',nn.LeakyReLU())
        ]))
        self.fc = nn.Sequential(
                nn.Linear(128*7*7,1),
                nn.Sigmoid()
        )

    def forward(self,x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.view(batch_size//num_gpus, -1)
        out = self.fc(out)
        return out
```

Basic Concept

Discriminator도 그냥 일반적인 형태.

출력 값이 하나이고 0에서 1 값을 가지기 때문에 마지막에 sigmoid를 추가함

```python
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator,self).__init__()
        self.layer1 = nn.Sequential(OrderedDict([
                ('conv1',nn.Conv2d(1,16,3,padding=1)),   # batch x 16 x 28 x 28
                ('relu1',nn.LeakyReLU()),
                ('bn1',nn.BatchNorm2d(16)),
                ('conv2',nn.Conv2d(16,32,3,padding=1)),  # batch x 32 x 28 x 28
                ('relu2',nn.LeakyReLU()),
                ('bn2',nn.BatchNorm2d(32)),
                ('max1',nn.MaxPool2d(2,2))   # batch x 32 x 14 x 14
        ]))
        self.layer2 = nn.Sequential(OrderedDict([
                ('conv3',nn.Conv2d(32,64,3,padding=1)),  # batch x 64 x 14 x 14
                ('relu3',nn.LeakyReLU()),
                ('bn3',nn.BatchNorm2d(64)),
                ('max2',nn.MaxPool2d(2,2)),
                ('conv4',nn.Conv2d(64,128,3,padding=1)),  # batch x 128 x 7 x 7
                ('relu4',nn.LeakyReLU())
        ]))
        self.fc = nn.Sequential(
                nn.Linear(128*7*7,1),
                nn.Sigmoid()
        )

    def forward(self,x):
        out = self.layer1(x)
        out = self.layer2(out)
        out = out.view(batch_size//num_gpus, -1)
        out = self.fc(out)
        return out
```

Basic Concept

```python
generator = nn.DataParallel(Generator()).cuda()
discriminator = nn.DataParallel(Discriminator()).cuda()

loss_func = nn.BCELoss()
gen_optim = torch.optim.Adam(generator.parameters(), lr=learning_rate)
dis_optim = torch.optim.Adam(discriminator.parameters(), lr=learning_rate)

ones_label = Variable(torch.ones(batch_size,1)).cuda()
zeros_label = Variable(torch.zeros(batch_size,1)).cuda()

# model restore

try:
    generator, discriminator = torch.load('./model/vanilla_gan.pkl')
    print("\n--------model restored--------\n")
except:
    print("\n--------model not restored--------\n")
    pass
```

Basic Concept

nn.DataParallel() 함수를 사용해 Multi-GPU에 모델을 올려서 학습

```
generator = nn.DataParallel(Generator()).cuda()
discriminator = nn.DataParallel(Discriminator()).cuda()

loss_func = nn.BCELoss()
gen_optim = torch.optim.Adam(generator.parameters(), lr=learning_rate)
dis_optim = torch.optim.Adam(discriminator.parameters(), lr=learning_rate)

ones_label = Variable(torch.ones(batch_size,1)).cuda()
zeros_label = Variable(torch.zeros(batch_size,1)).cuda()

# model restore

try:
    generator, discriminator = torch.load('./model/vanilla_gan.pkl')
    print("\n--------model restored--------\n")
except:
    print("\n--------model not restored--------\n")
    pass
```

Basic Concept

nn.DataParallel() 함수를 사용해 Multi-GPU에 모델을 올려서 학습

각각 파라미터들을 따로 학습해야 하기 때문에 optimizer도 따로 만듦

```
generator = nn.DataParallel(Generator()).cuda()
discriminator = nn.DataParallel(Discriminator()).cuda()

loss_func = nn.BCELoss()
gen_optim = torch.optim.Adam(generator.parameters(), lr=learning_rate)
dis_optim = torch.optim.Adam(discriminator.parameters(), lr=learning_rate)

ones_label = Variable(torch.ones(batch_size,1)).cuda()
zeros_label = Variable(torch.zeros(batch_size,1)).cuda()

# model restore

try:
    generator, discriminator = torch.load('./model/vanilla_gan.pkl')
    print("\n--------model restored--------\n")
except:
    print("\n--------model not restored--------\n")
    pass
```

Basic Concept

nn.DataParallel() 함수를 사용해 Multi-GPU에 모델을 올려서 학습

각각 파라미터들을 따로 학습해야 하기 때문에 optimizer도 따로 만듦

라벨로 사용할 0과 1값 생성

```python
generator = nn.DataParallel(Generator()).cuda()
discriminator = nn.DataParallel(Discriminator()).cuda()

loss_func = nn.BCELoss()
gen_optim = torch.optim.Adam(generator.parameters(), lr=learning_rate)
dis_optim = torch.optim.Adam(discriminator.parameters(), lr=learning_rate)

ones_label = Variable(torch.ones(batch_size,1)).cuda()
zeros_label = Variable(torch.zeros(batch_size,1)).cuda()

# model restore

try:
    generator, discriminator = torch.load('./model/vanilla_gan.pkl')
    print("\n--------model restored--------\n")
except:
    print("\n--------model not restored--------\n")
    pass
```

Basic Concept

nn.DataParallel() 함수를 사용해 Multi-GPU에 모델을 올려서 학습

```
generator = nn.DataParallel(Generator()).cuda()
discriminator = nn.DataParallel(Discriminator()).cuda()
```

각각 파라미터들을 따로 학습해야 하기 때문에 optimizer도 따로 만듦

```
loss_func = nn.BCELoss()
gen_optim = torch.optim.Adam(generator.parameters(), lr=learning_rate)
dis_optim = torch.optim.Adam(discriminator.parameters(), lr=learning_rate)
```

라벨로 사용할 0과 1값 생성

```
ones_label = Variable(torch.ones(batch_size,1)).cuda()
zeros_label = Variable(torch.zeros(batch_size,1)).cuda()

# model restore
```

학습된 모델 불러오는 부분

```
try:
    generator, discriminator = torch.load('./model/vanilla_gan.pkl')
    print("\n--------model restored--------\n")
except:
    print("\n--------model not restored--------\n")
    pass
```

Basic Concept

```python
# train

for i in range(epoch):
    for j,(image,label) in enumerate(train_loader):
        image = Variable(image).cuda()

        z = Variable(torch.rand(batch_size,100)).cuda()
        gen_optim.zero_grad()
        gen_fake = generator.forward(z)
        dis_fake = discriminator.forward(gen_fake)
        gen_loss = torch.sum(loss_func(dis_fake,ones_label))
        gen_loss.backward(retain_variables=True)
        gen_optim.step()

        dis_optim.zero_grad()
        dis_real = discriminator.forward(image)
        dis_loss = torch.sum(loss_func(dis_fake,zeros_label))+torch.sum(loss_func(dis_real,ones_label))
        dis_loss.backward()
        dis_optim.step()

    if i % 5 == 0:
        torch.save([generator,discriminator],'./model/vanilla_gan.pkl')

    print("{}th iteration gen_loss: {} dis_loss: {}".format(i,gen_loss.data,dis_loss.data))
    v_utils.save_image(gen_fake.data[0:20],"./gan_result/gen_{}.png".format(i), nrow=5)
```

Basic Concept

매번 새로운 noise를 생성해서 generate

```
# train

for i in range(epoch):
    for j,(image,label) in enumerate(train_loader):
        image = Variable(image).cuda()

        z = Variable(torch.rand(batch_size,100)).cuda()
        gen_optim.zero_grad()
        gen_fake = generator.forward(z)
        dis_fake = discriminator.forward(gen_fake)
        gen_loss = torch.sum(loss_func(dis_fake,ones_label))
        gen_loss.backward(retain_variables=True)
        gen_optim.step()

        dis_optim.zero_grad()
        dis_real = discriminator.forward(image)
        dis_loss = torch.sum(loss_func(dis_fake,zeros_label))+torch.sum(loss_func(dis_real,ones_label))
        dis_loss.backward()
        dis_optim.step()

    if i % 5 == 0:
        torch.save([generator,discriminator],'./model/vanilla_gan.pkl')

    print("{}th iteration gen_loss: {} dis_loss: {}".format(i,gen_loss.data,dis_loss.data))
    v_utils.save_image(gen_fake.data[0:20],"./gan_result/gen_{}.png".format(i), nrow=5)
```

Basic Concept

매번 새로운 noise를 생성해서 generate

discriminator 통과해서 1과의 BCE loss
계산 및 업데이트

```
# train

for i in range(epoch):
    for j,(image,label) in enumerate(train_loader):
        image = Variable(image).cuda()

        z = Variable(torch.rand(batch_size,100)).cuda()
        gen_optim.zero_grad()
        gen_fake = generator.forward(z)
        dis_fake = discriminator.forward(gen_fake)
        gen_loss = torch.sum(loss_func(dis_fake,ones_label))
        gen_loss.backward(retain_variables=True)
        gen_optim.step()

        dis_optim.zero_grad()
        dis_real = discriminator.forward(image)
        dis_loss = torch.sum(loss_func(dis_fake,zeros_label))+torch.sum(loss_func(dis_real,ones_label))
        dis_loss.backward()
        dis_optim.step()

    if i % 5 == 0:
        torch.save([generator,discriminator],'./model/vanilla_gan.pkl')

    print("{}th iteration gen_loss: {} dis_loss: {}".format(i,gen_loss.data,dis_loss.data))
    v_utils.save_image(gen_fake.data[0:20],"./gan_result/gen_{}.png".format(i), nrow=5)
```

Basic Concept

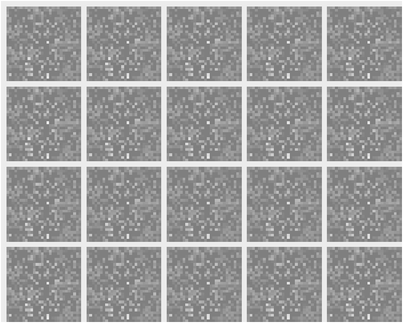
매번 새로운 noise를 생성해서 generate

discriminator 통과해서 1과의 BCE loss
계산 및 업데이트

실제 데이터와 가짜 데이터 가지고 discriminator 통과해서 이번엔 fake는 0, real은 1로
계산 및 업데이트

```
# train

for i in range(epoch):
    for j,(image,label) in enumerate(train_loader):
        image = Variable(image).cuda()

        z = Variable(torch.rand(batch_size,100)).cuda()
        gen_optim.zero_grad()
        gen_fake = generator.forward(z)
        dis_fake = discriminator.forward(gen_fake)
        gen_loss = torch.sum(loss_func(dis_fake,ones_label))
        gen_loss.backward(retain_variables=True)
        gen_optim.step()

        dis_optim.zero_grad()
        dis_real = discriminator.forward(image)
        dis_loss = torch.sum(loss_func(dis_fake,zeros_label))+torch.sum(loss_func(dis_real,ones_label))
        dis_loss.backward()
        dis_optim.step()

    if i % 5 == 0:
        torch.save([generator,discriminator],'./model/vanilla_gan.pkl')

    print("{}th iteration gen_loss: {} dis_loss: {}".format(i,gen_loss.data,dis_loss.data))
    v_utils.save_image(gen_fake.data[0:20],"./gan_result/gen_{}.png".format(i), nrow=5)
```

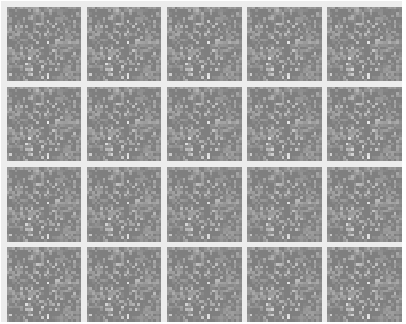
Basic Concept



Mode collapse

Oscillation

Basic Concept



Mode collapse

서로 다른 노이즈 값들로 generate 해도 하나의 mode로 수렴하는 현상

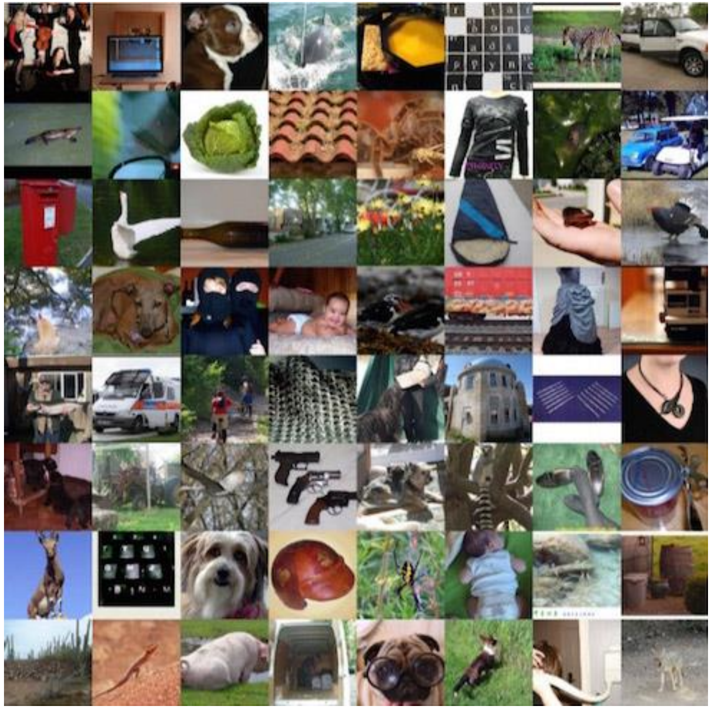
네트워크가 다양한 실제 이미지 같은 결과를 내기를 기대하는데 그럴 싸한 이미지만 만들어냄

Oscillation

GAN output들이 목표 카테고리가 여러 가지일 경우 왔다 갔다 하면서 다양한 결과값들을 만들어 내고 하나에 수렴하지 못하는 경우

Basic Concept

이러한 문제들이 있긴 하지만 잘될 땐 또 잘됨



Real Images

GAN Images

https://blog.openai.com/generative-models/

Basic Concept

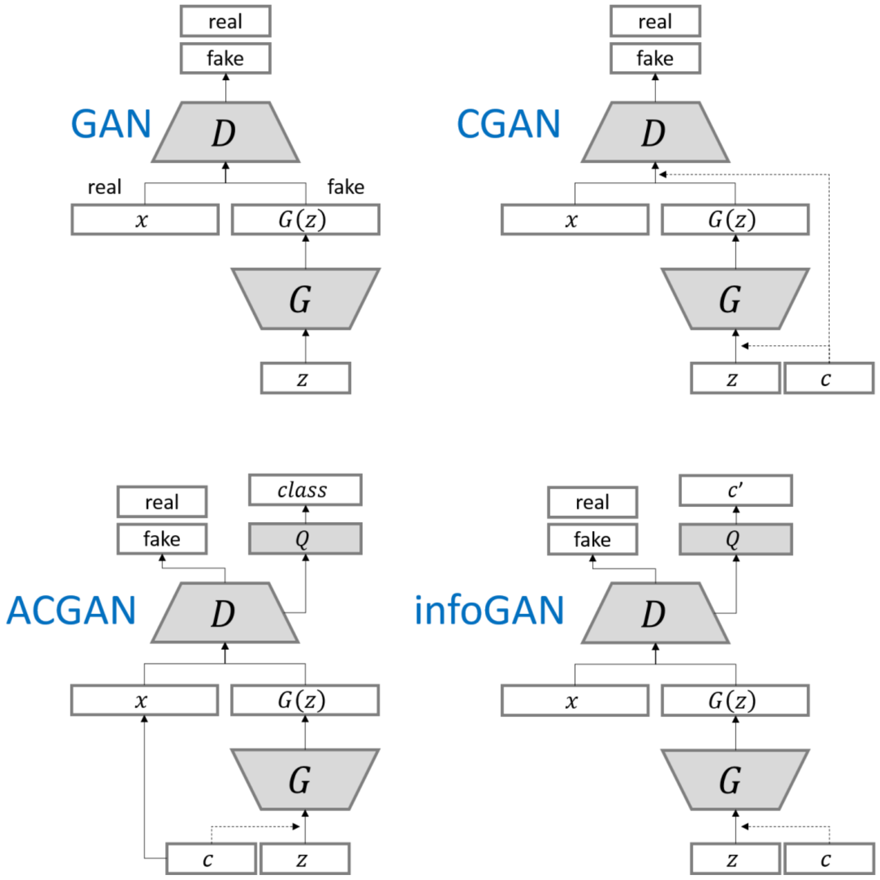
이러한 문제들이 있긴 하지만 잘될 땐 또 잘됨

VAE generated images

GAN Images

https://blog.openai.com/generative-models/

Models

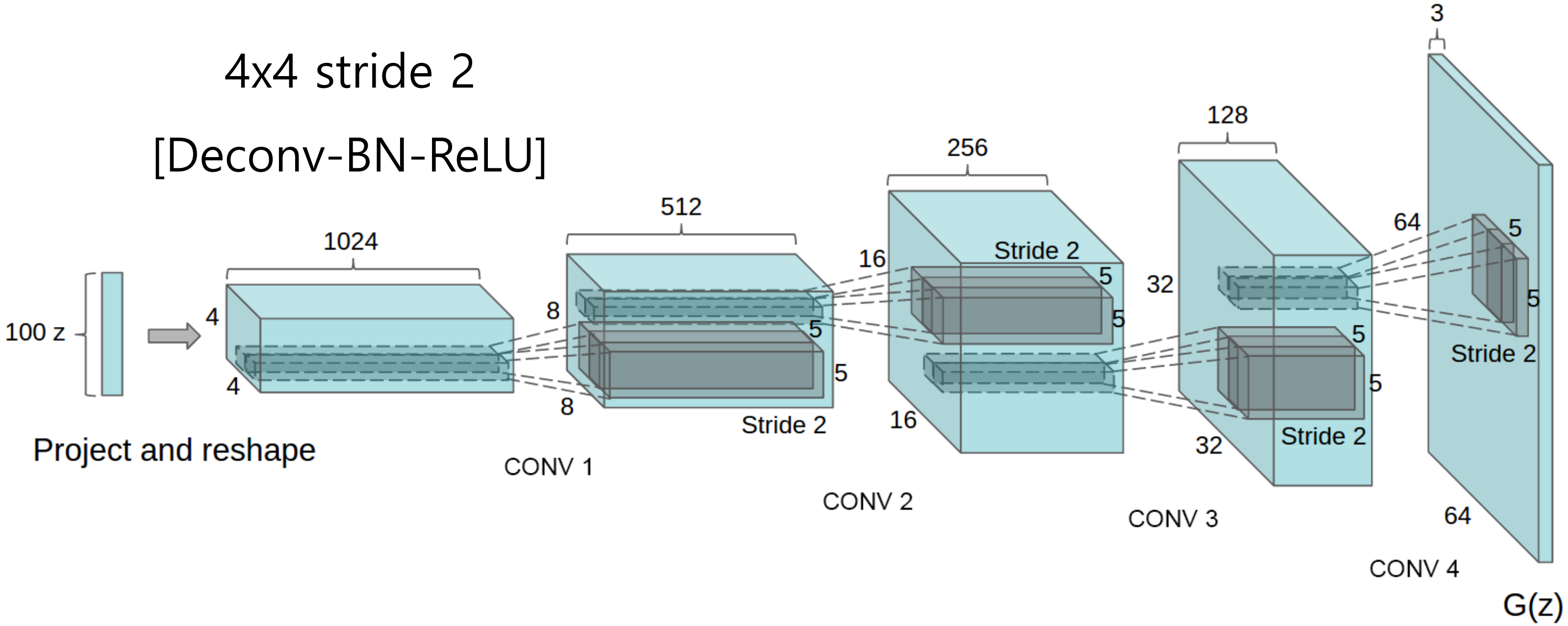


출처: https://github.com/hwalsuklee/tensorflow-generative-model-collections

Models

4x4 stride 2

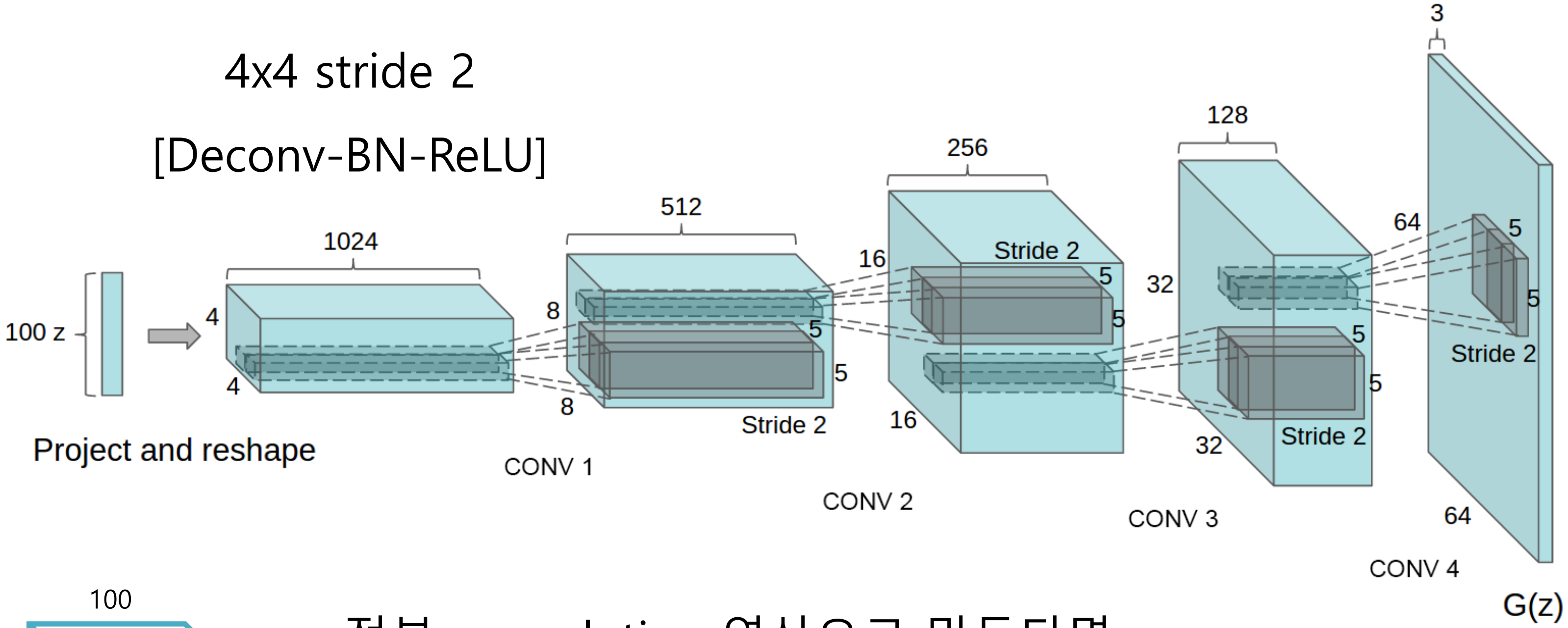
[Deconv-BN-ReLU]



Models

4x4 stride 2

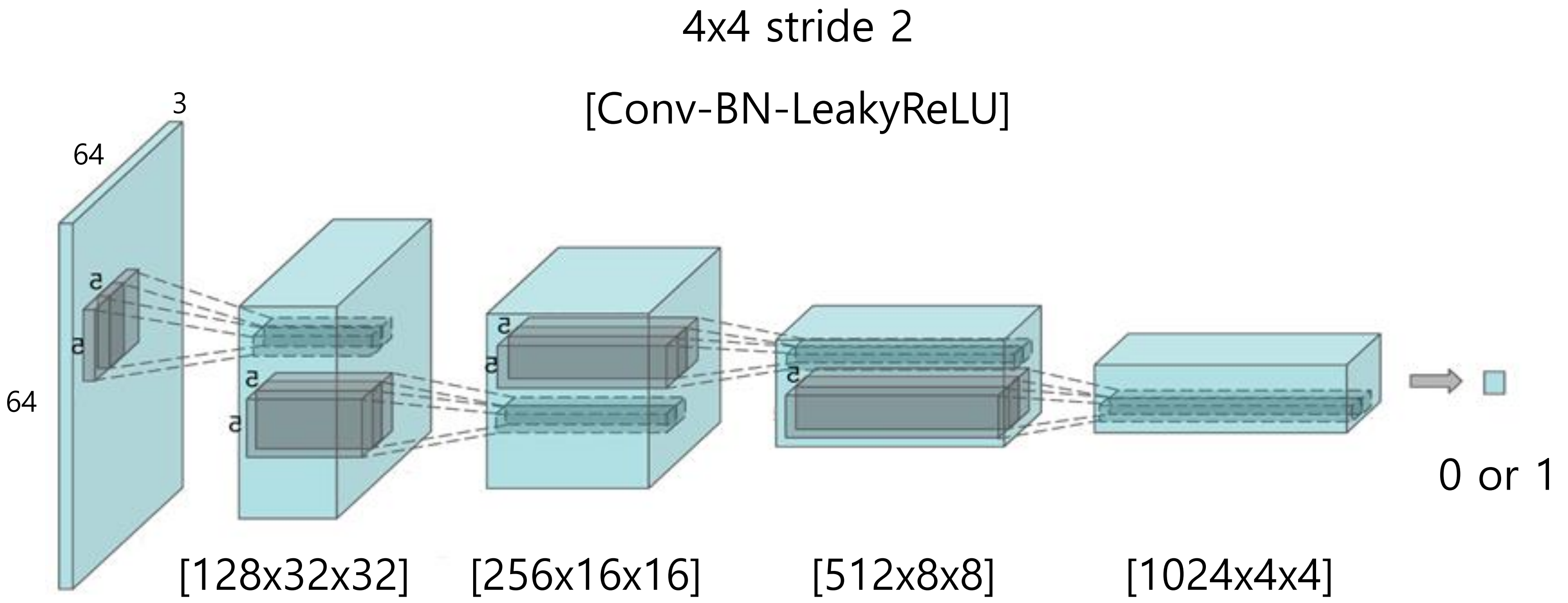
[Deconv-BN-ReLU]



100x1x1

전부 convolution 연산으로 만든다면
z를 길이 100짜리 1차원 벡터가 아닌 [100,1,1]
3차원 벡터화시켜서 여기에 transposed convolution 적용

Models


4x4 stride 2
[Conv-BN-LeakyReLU]
3
64
64
[128x32x32]
[256x16x16]
[512x8x8]
[1024x4x4]
0 or 1

Models

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

- Pooling 제거하고 Strided Convolution으로 대체
- Batch Normalization 사용
- Fully Connected Layer 제거
- Generator에는 ReLU, 마지막단에만 Tanh
- Discriminator에는 LeakyReLU(0.2)

Models

1 epoch 결과

Models

DCGAN

5 epoch 결과

Models

모델이 그냥 외운 것 아닌가?

Models

모델이 그냥 외운 것 아닌가?

⇩

확인을 어떻게 하지?

Models

모델이 그냥 외운 것 아닌가?

⇩

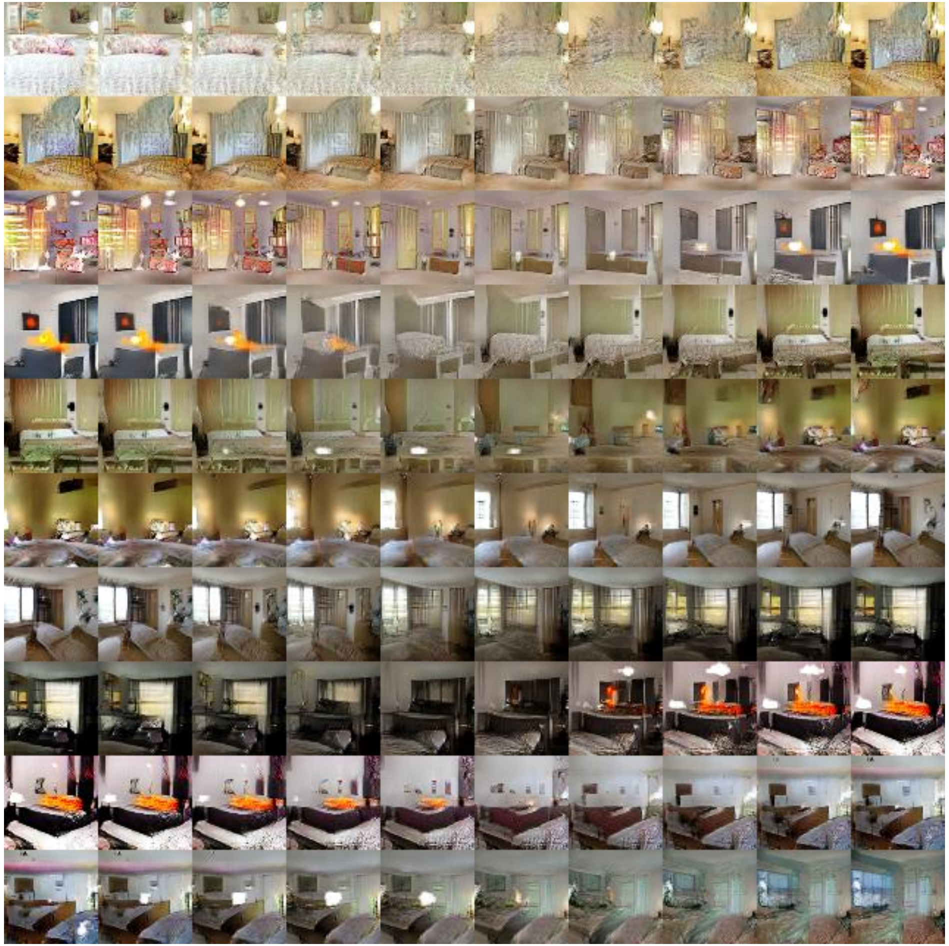
확인을 어떻게 하지?

⇩

z를 interpolation하면서 어떻게 변하나 확인해보자
만약에 급격하게 확 변하면 외운 것이고 아니면 어떤
의미를 이해한 것으로 판단할 수 있다!

Models

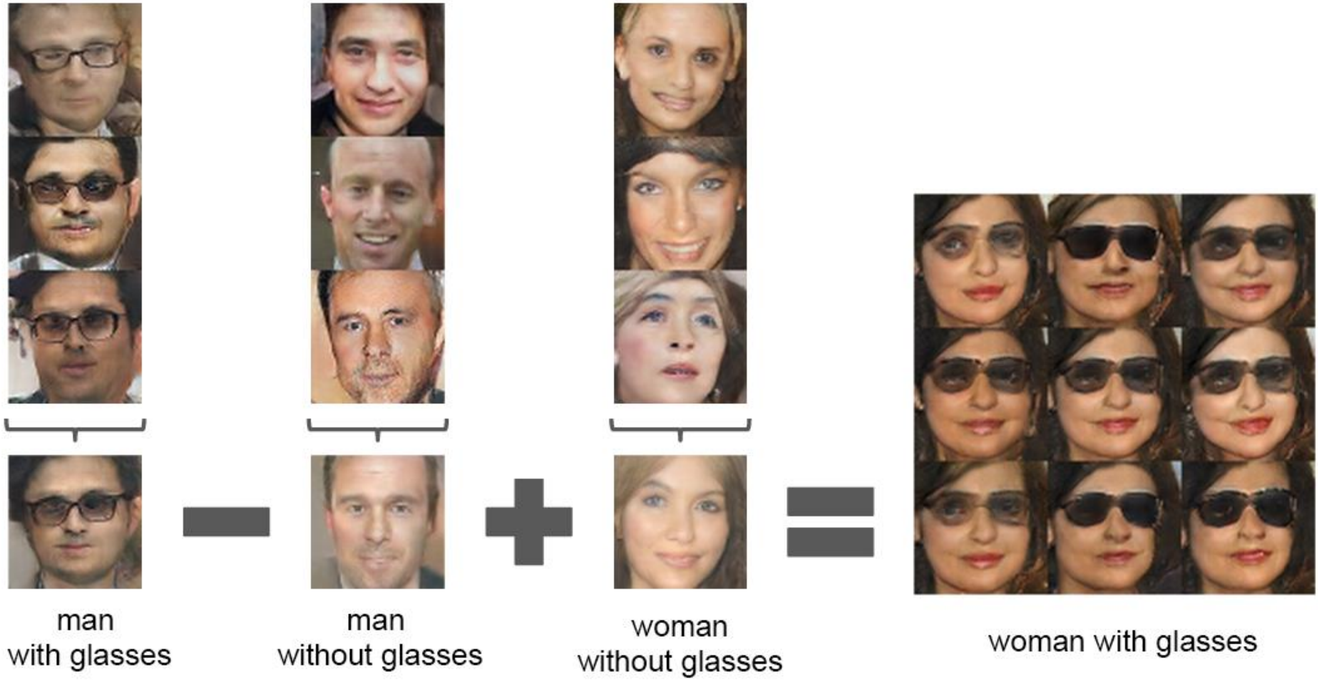
DCGAN



없던 창문이 생겨남

TV가 창문으로 변함

Models



Vector Arithmetic이 가능하다!

Models

DCGAN

이런 Interpolation도 가능하다!

Models

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

출처: https://github.com/hwalsuklee/tensorflow-generative-model-collections

Models

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{z}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

$$\Downarrow$$

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{z}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

출처: https://github.com/hwalsuklee/tensorflow-generative-model-collections

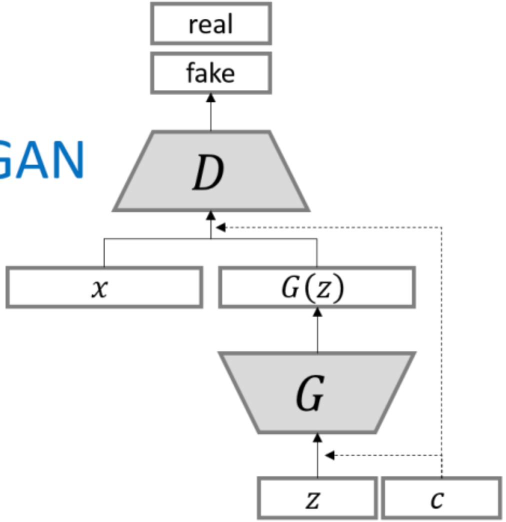
Models

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

$$\Downarrow$$

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{z}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

CGAN



출처: https://github.com/hwalsuklee/tensorflow-generative-model-collections

Models

CGAN

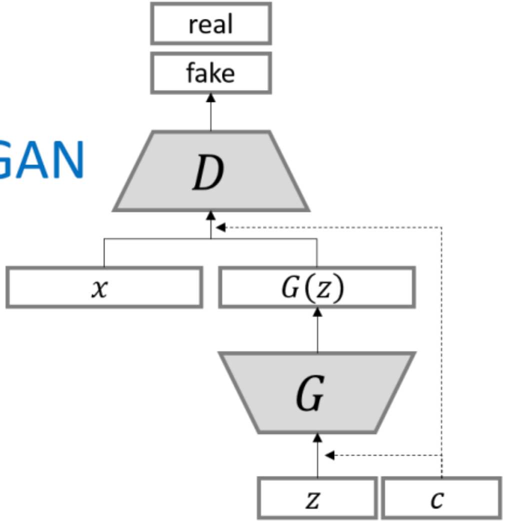
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

$$\Downarrow$$

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

y가 label

CGAN



출처: https://github.com/hwalsuklee/tensorflow-generative-model-collections

Models

CGAN

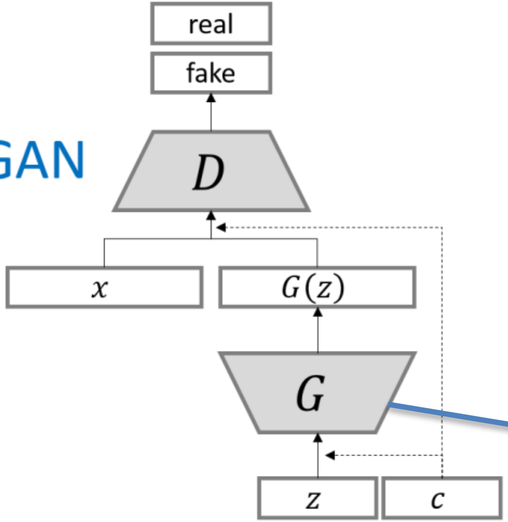
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

$$\Downarrow$$

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

y가 label

CGAN



이미지를 생성할 때 class
라벨을 z와 함께 전달

출처: https://github.com/hwalsuklee/tensorflow-generative-model-collections

Models

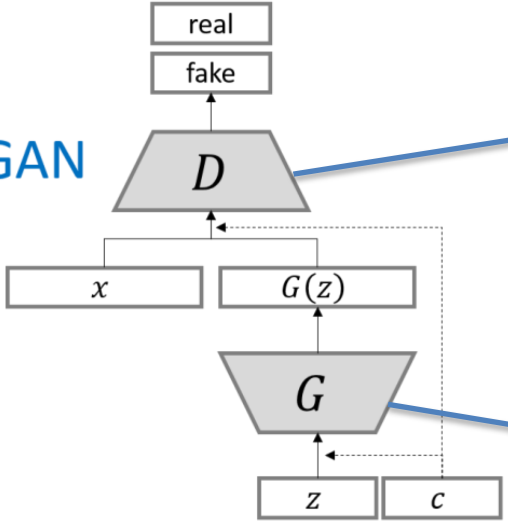
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

$$\Downarrow$$

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

y가 label

CGAN



Discriminator 에도class
라벨을 z와 함께 전달

이미지를 생성할 때 class
라벨을 z와 함께 전달

출처: https://github.com/hwalsuklee/tensorflow-generative-model-collections

Models

CGAN

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

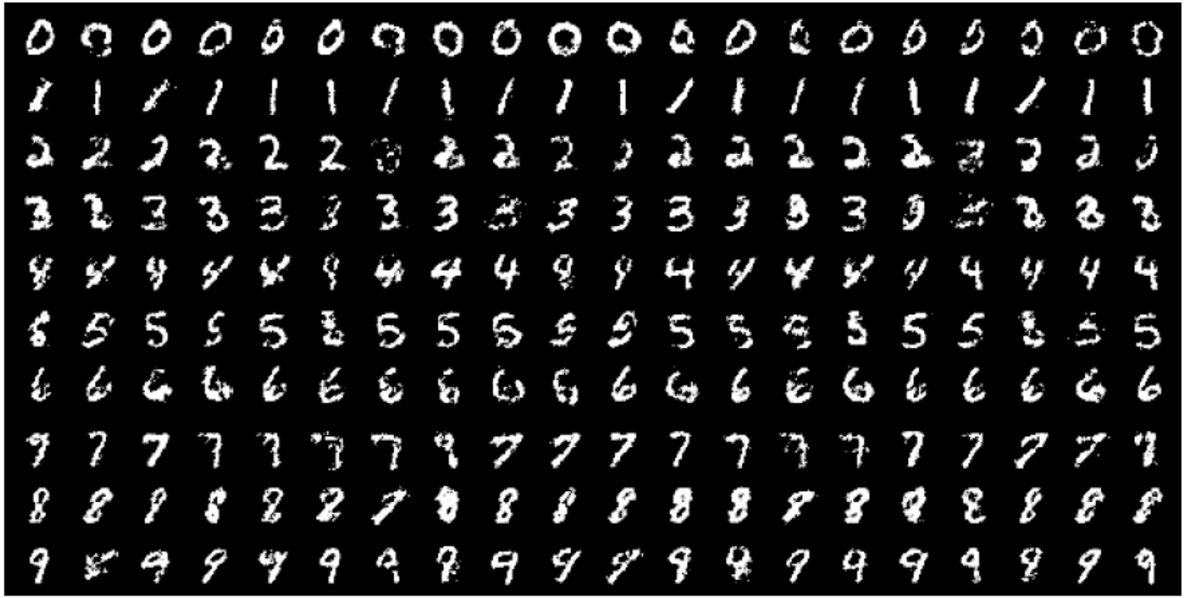


Figure 2: Generated MNIST digits, each row conditioned on one label

Models

CGAN

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

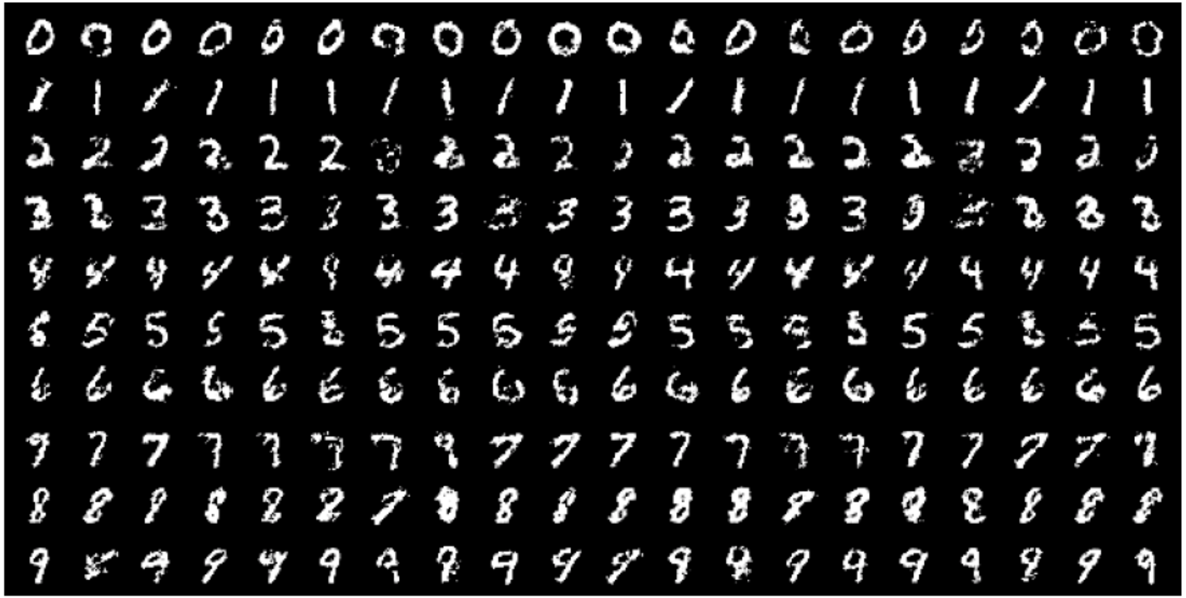
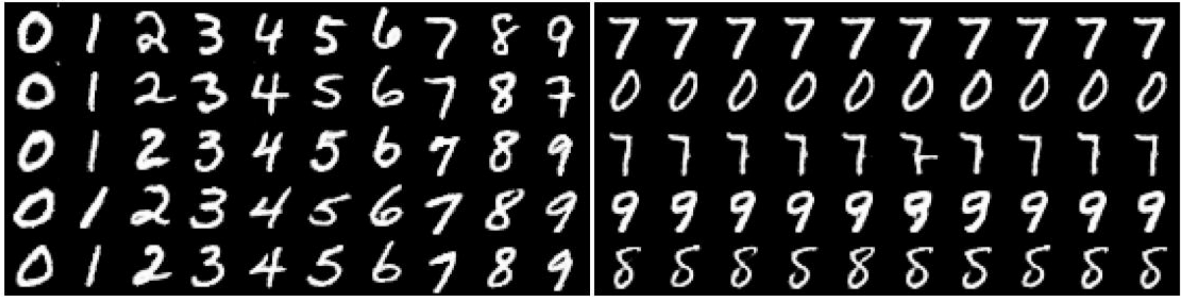


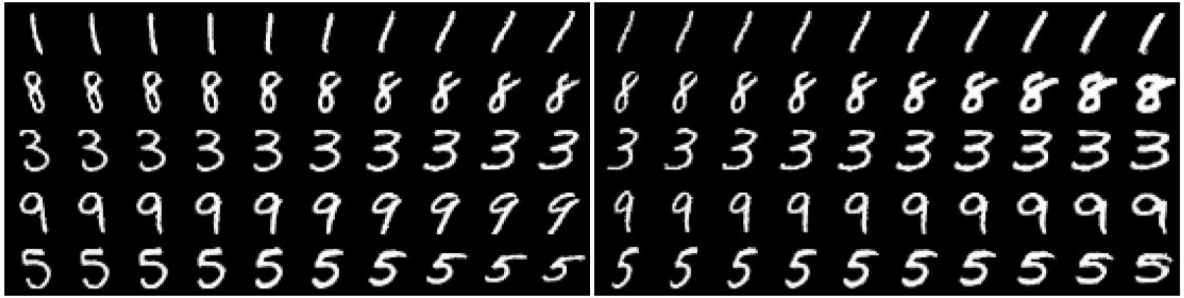
Figure 2: Generated MNIST digits, each row conditioned on one label

꼭 label이 있어야 가능한 건가?

Models



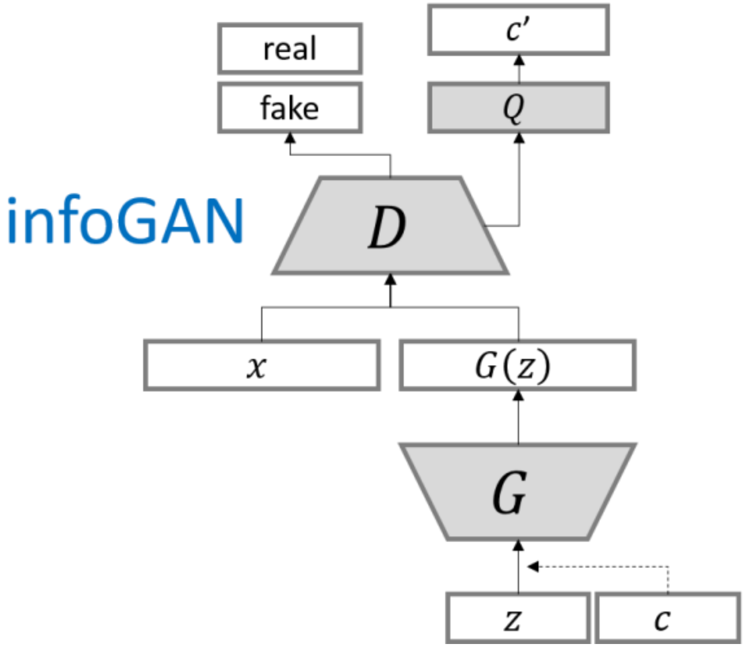
(a) Varying c_1 on InfoGAN (Digit type) (b) Varying c_1 on regular GAN (No clear meaning)



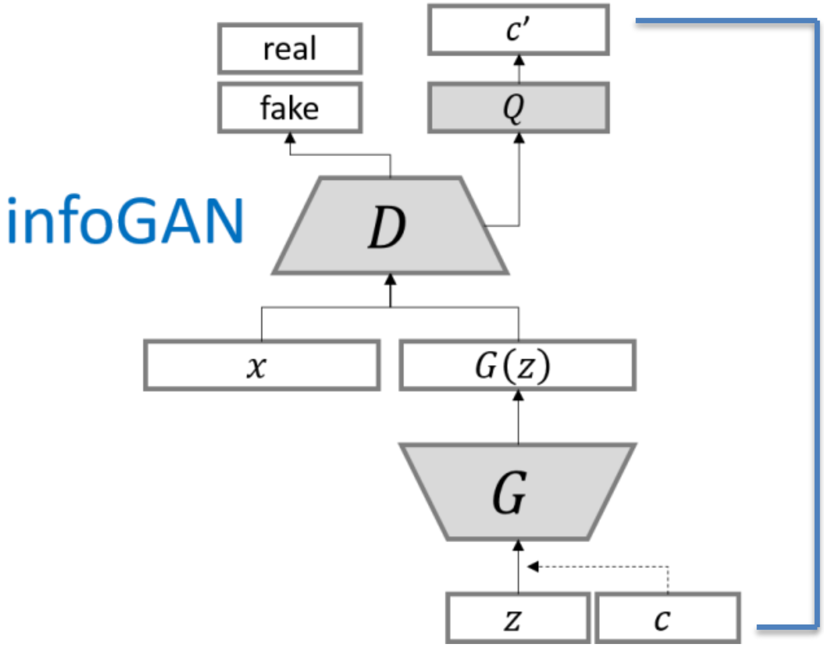
(c) Varying c_2 from -2 to 2 on InfoGAN (Rotation) (d) Varying c_3 from -2 to 2 on InfoGAN (Width)

Unsupervised Learning으로 카테고리나 기울어진 정도, 두께를 찾을 수 있다!

Models



Models

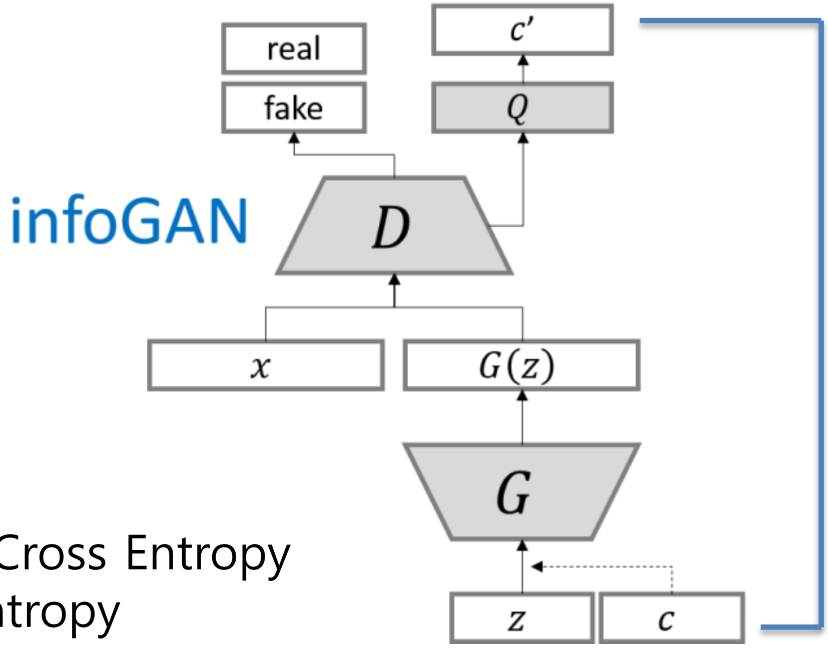


기존의 GAN

+

Generator에 넣은 c가
Discriminator를 통과
-해도 남아있게 만들자

Models



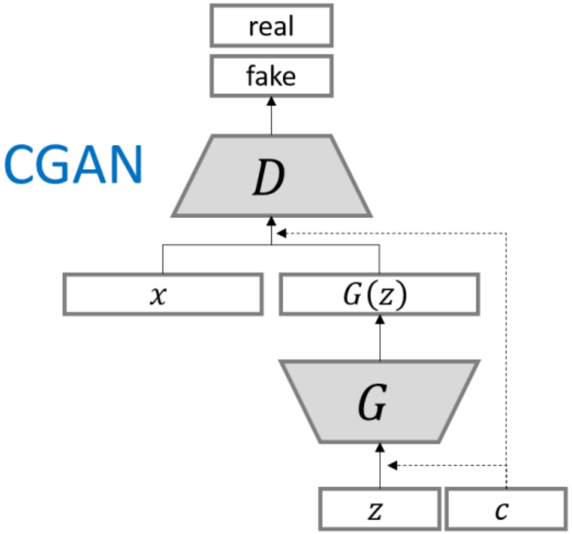
기존의 GAN

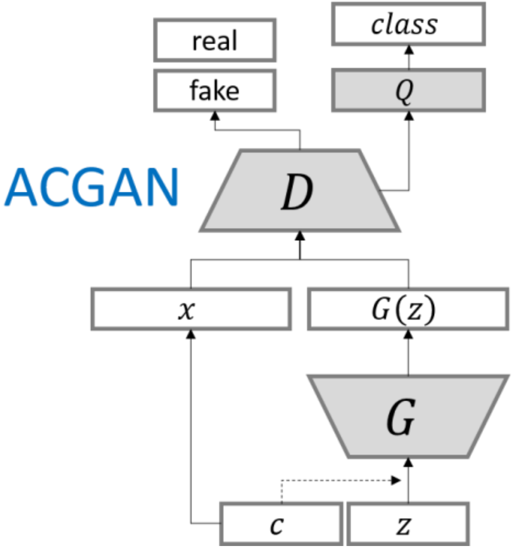
+

Generator에 넣은 c가
Discriminator를 통과
-해도 남아있게 만들자

Real Fake는 Binary Cross Entropy
discrete 은 Cross Entropy
continuous는 MSE

Models





CGAN은 생성 및 구분 시 label을
추가 정보로만 제공했다면

ACGAN은 Auxiliary Classifier를
추가하여 클래스를 구분하게 만듦

Models

SRGAN

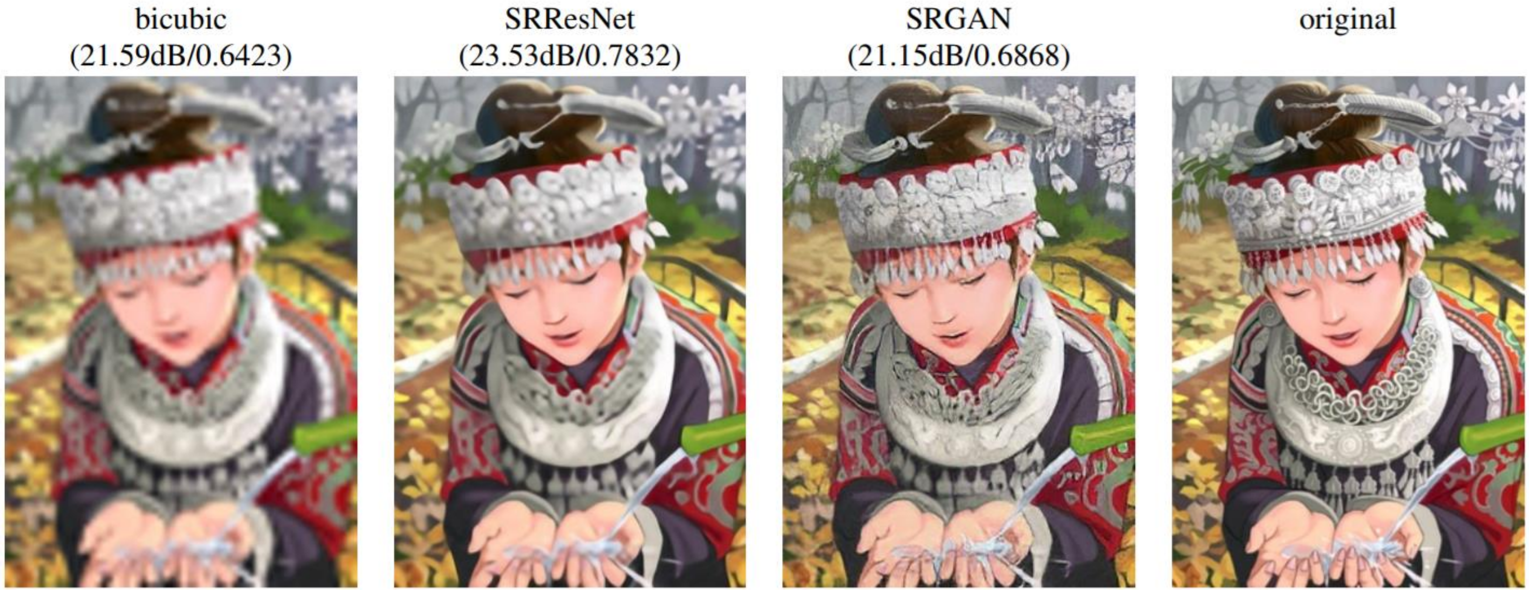
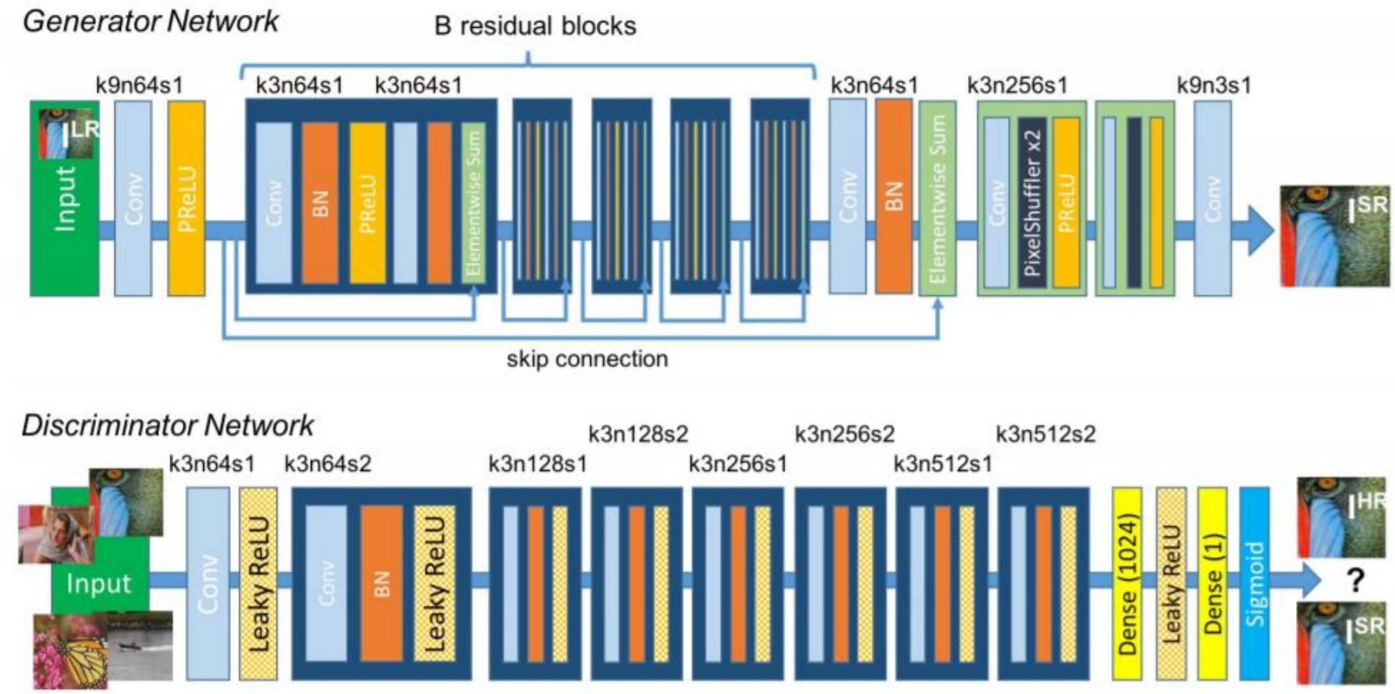


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Models

SRGAN

저해상도



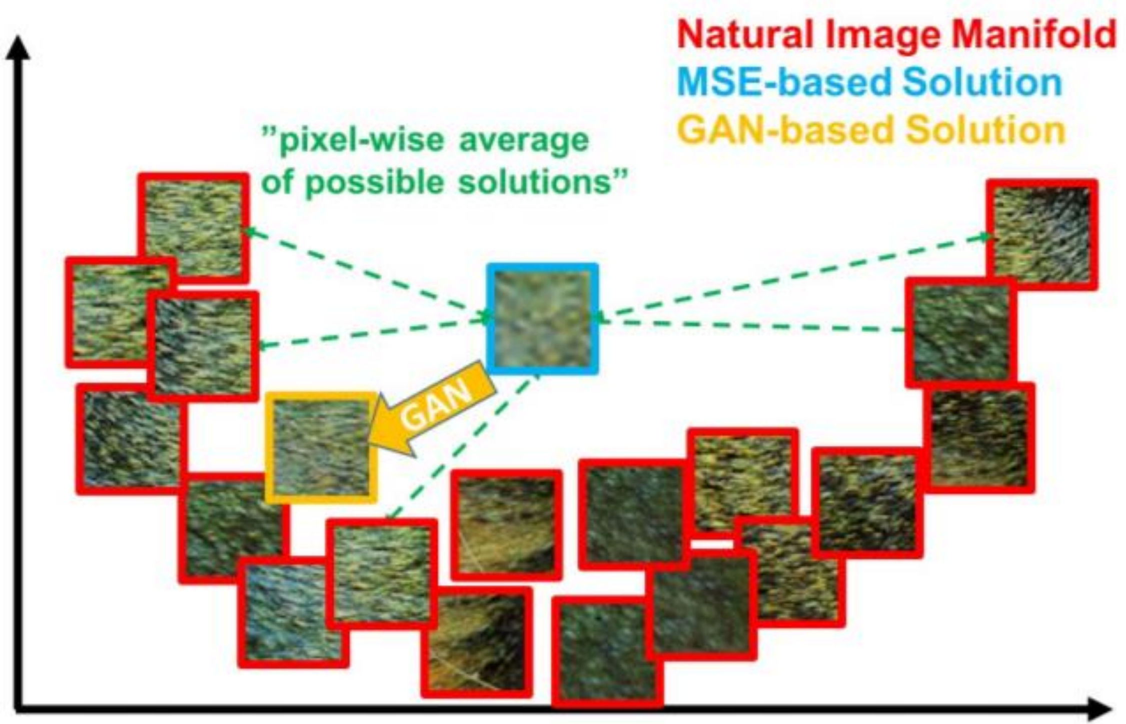
고해상도

Real

Fake

Models

SRGAN



Mode collapse가 일어나는 GAN의 특성이 오히려 장점이 된 케이스

Models

Generative Adversarial Text to Image Synthesis

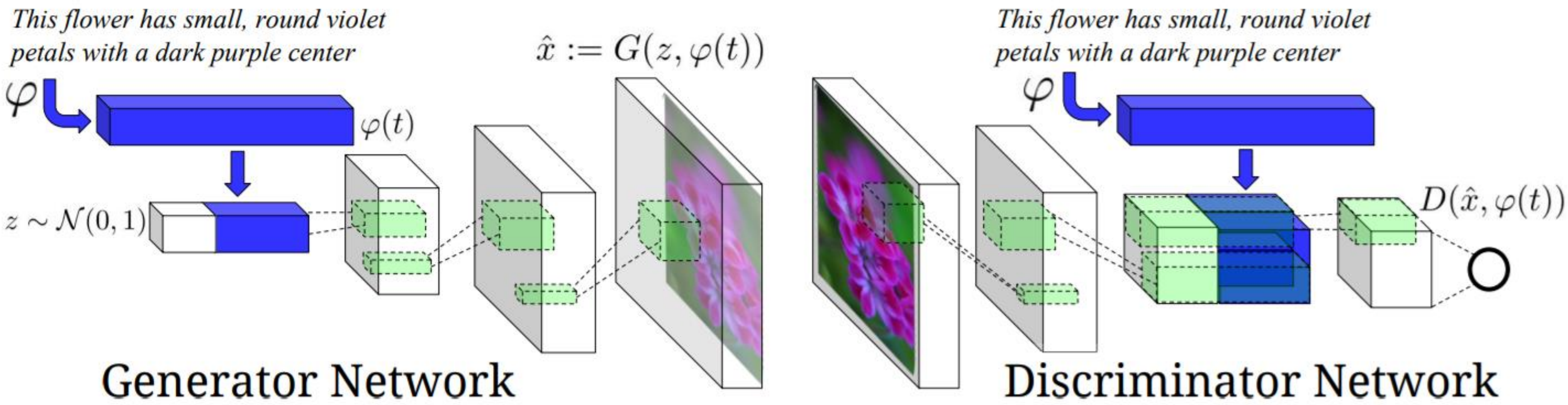
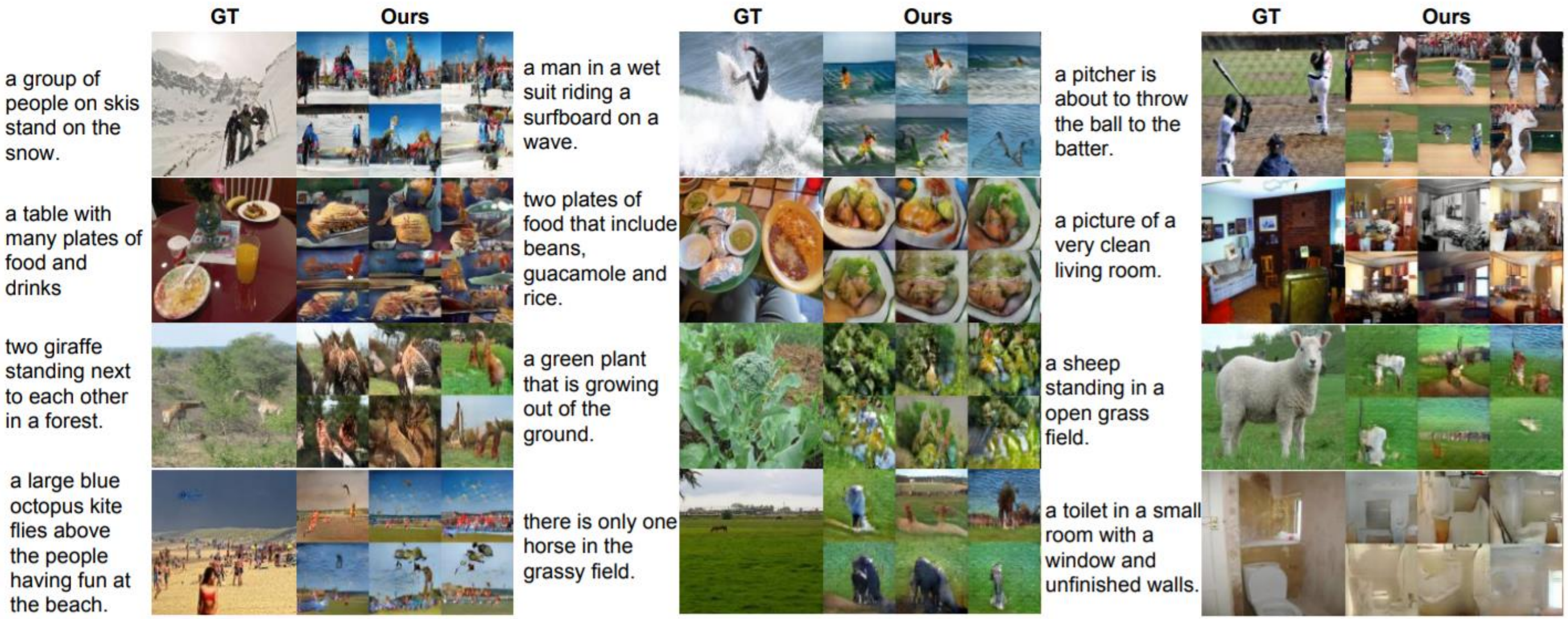


Figure 2. Our text-conditional convolutional GAN architecture. Text encoding $\varphi(t)$ is used by both generator and discriminator. It is projected to a lower-dimensions and depth concatenated with image feature maps for further stages of convolutional processing.

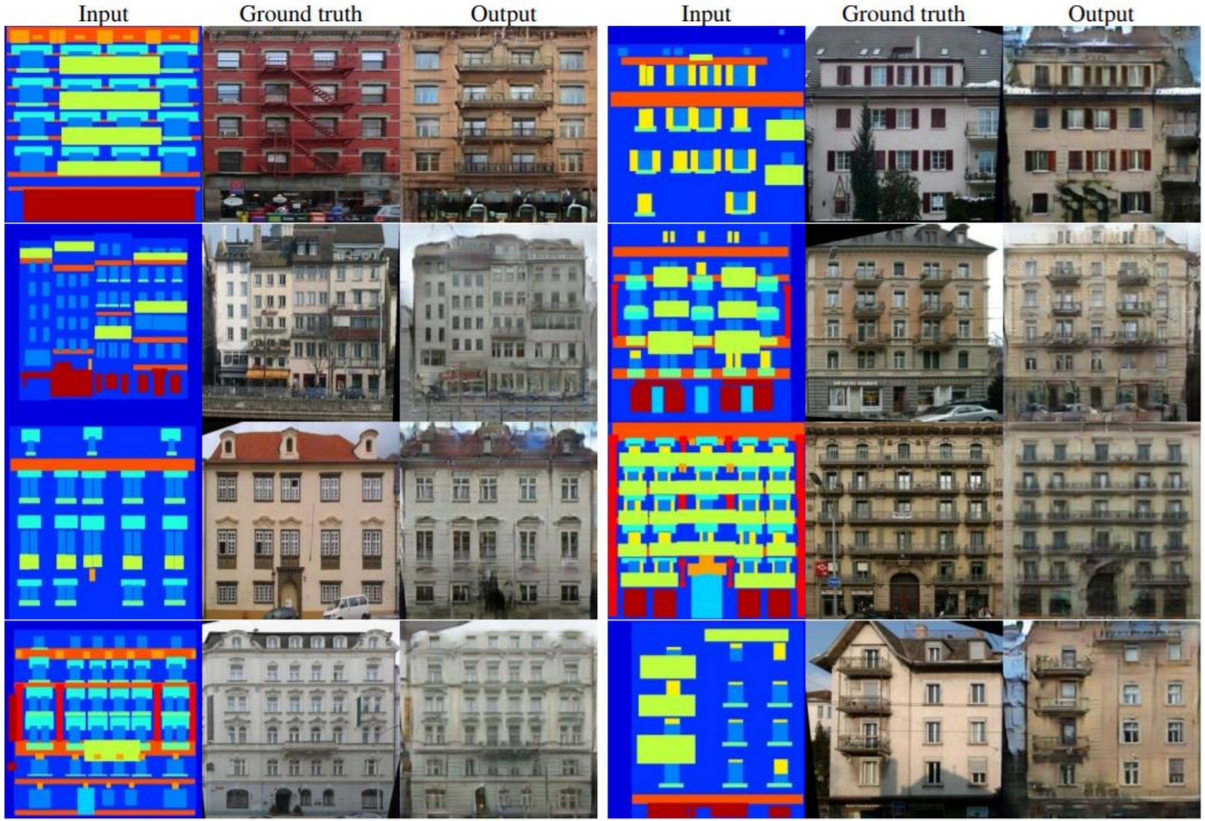
Models



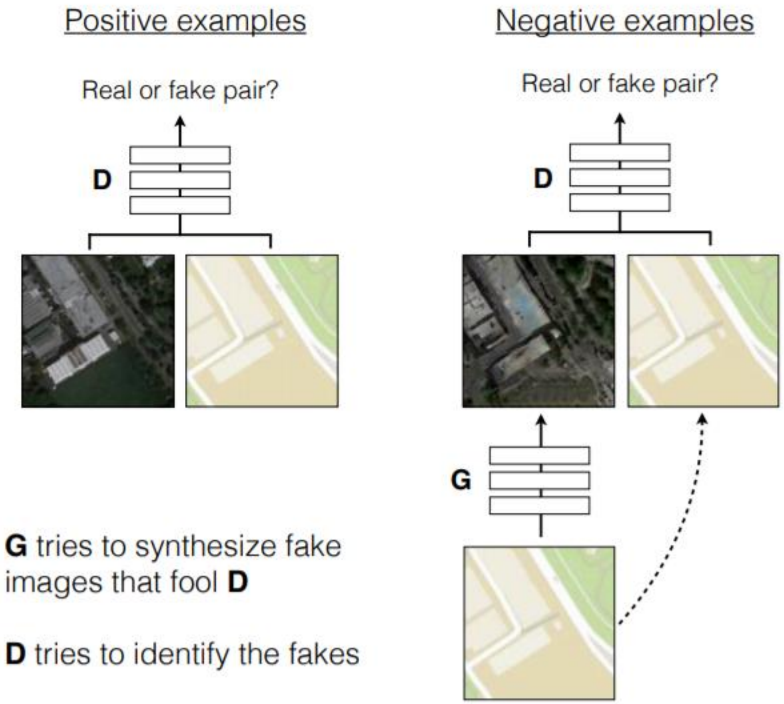
텍스트가 들어오면 해당 문장에 맞는
이미지를 생성해내는 conditional GAN

Models

pix2pix



Models



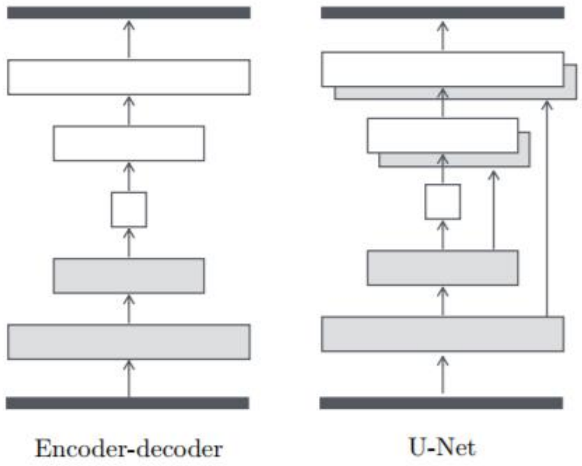
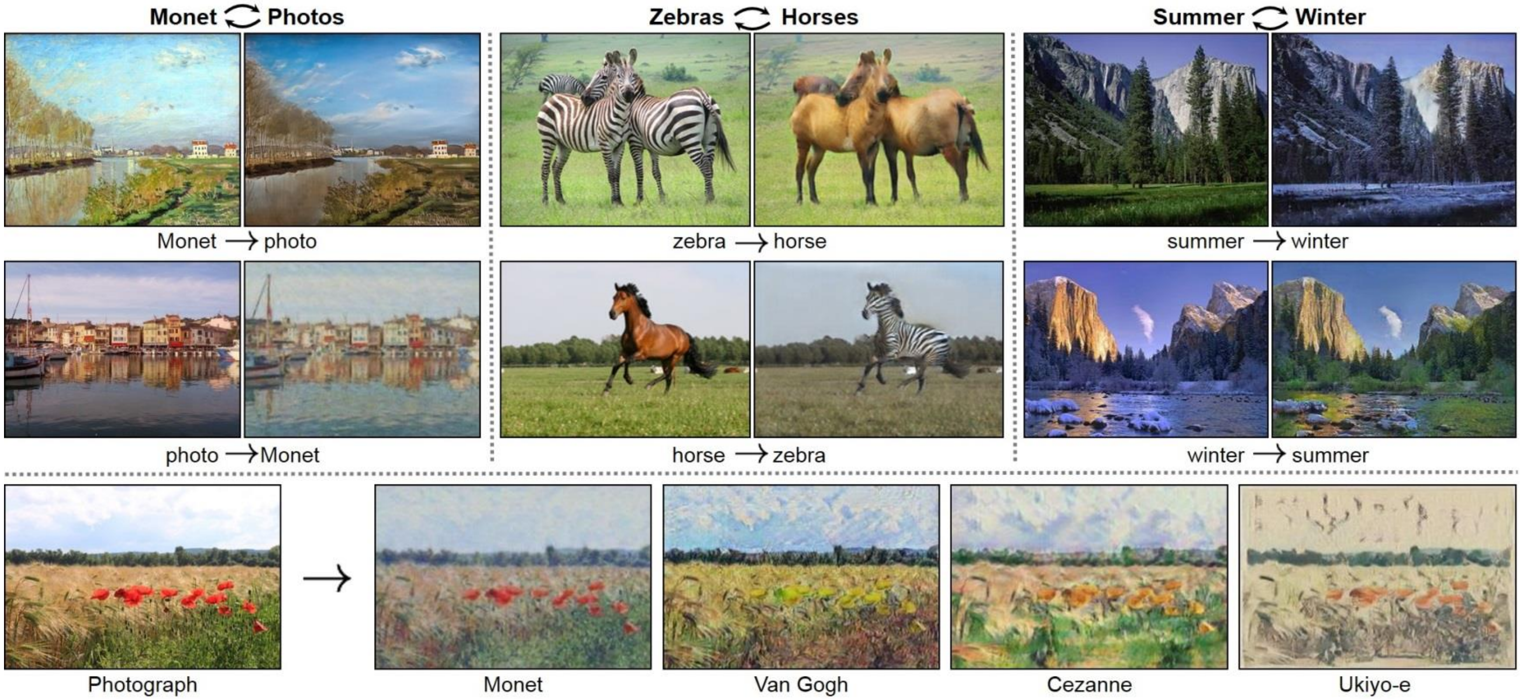


Figure 3: Two choices for the architecture of the generator. The “U-Net” [34] is an encoder-decoder with skip connections between mirrored layers in the encoder and decoder stacks.

입력으로 이미지를 받기 때문에 Generator
-가 Encoder-Decoder 형식을 가짐

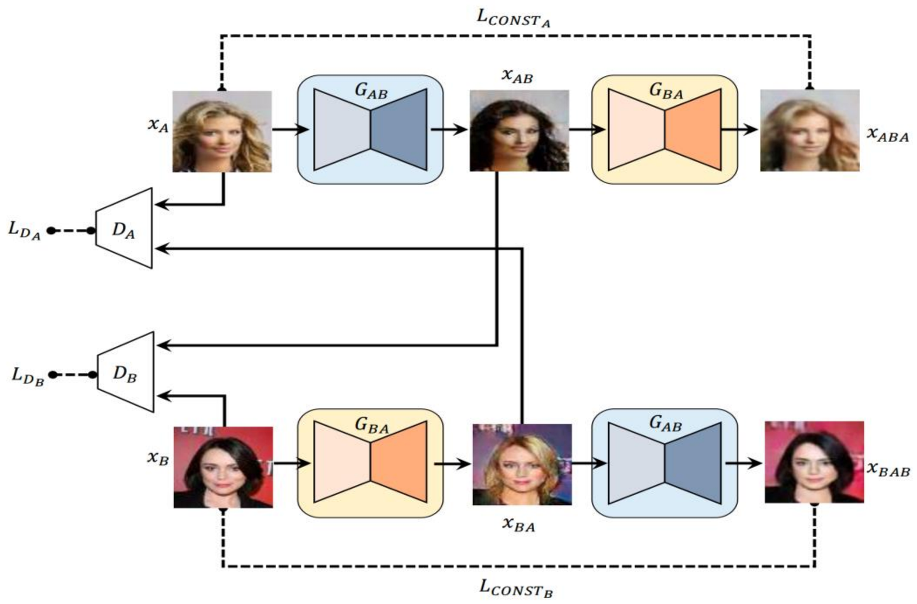
Models

CycleGAN

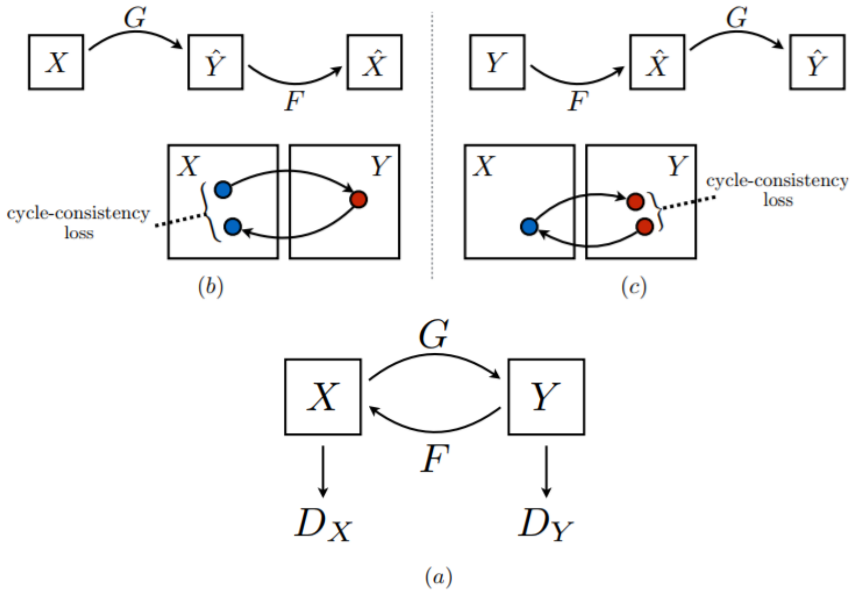


Pair가 있던 pix2pix과 달리 Unpaired Image to Image Translation

Models



DiscoGAN



CycleGAN

Models

Style Transfer

Image to Photograph

Models

Progressive Growing of GANs

https://youtu.be/XOxxPcy5Gr4?t=1m21s

Progressive Growing of GANs

Models

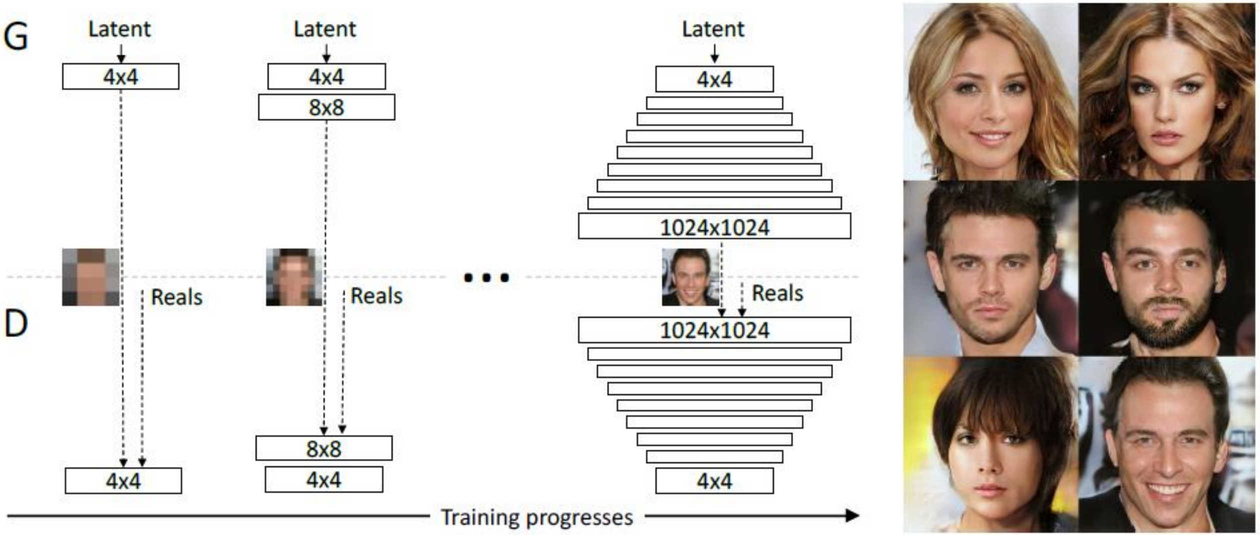


Figure 1: Our training starts with both the generator (G) and discriminator (D) having a low spatial resolution of 4×4 pixels. As the training advances, we incrementally add layers to G and D, thus increasing the spatial resolution of the generated images. All existing layers remain trainable throughout the process. Here $\boxed{N \times N}$ refers to convolutional layers operating on $N \times N$ spatial resolution. This allows stable synthesis in high resolutions and also speeds up training considerably.

Models

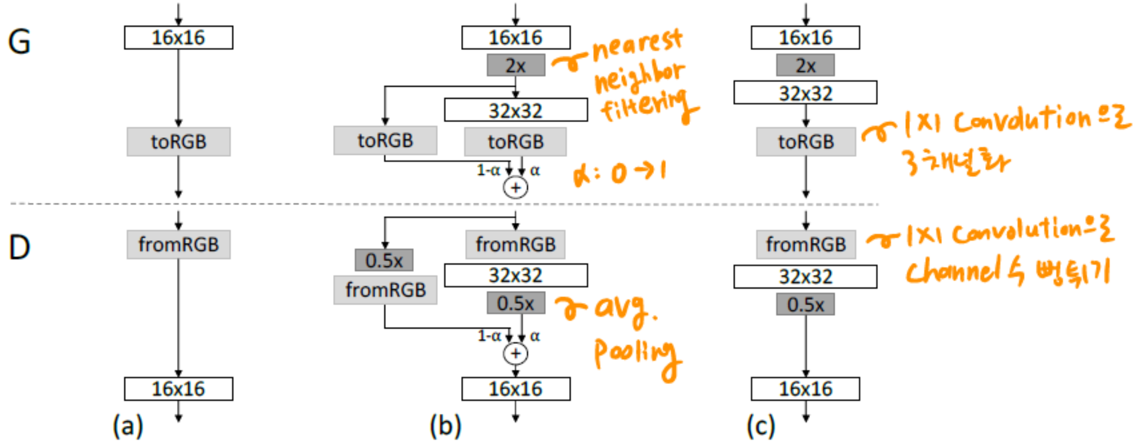


Figure 2: When doubling the resolution of the generator (G) and discriminator (D) we fade in the new layers smoothly. This example illustrates the transition from 16 × 16 images (a) to 32 × 32 images (c). During the transition (b) we treat the layers that operate on the higher resolution like a residual block, whose weight α increases linearly from 0 to 1. Here 2× and 0.5× refer to doubling and halving the image resolution using nearest neighbor filtering and average pooling, respectively. The toRGB represents a layer that projects feature vectors to RGB colors and fromRGB does the reverse; both use 1 × 1 convolutions. When training the discriminator, we feed in real images that are downscaled to match the current resolution of the network. During a resolution transition, we interpolate between two resolutions of the real images, similarly to how the generator output combines two resolutions. 외우

Models

소개한 예시 외에도 엄청 많음

Papers

Theory & Machine Learning

- A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models [arXiv]
- A General Retraining Framework for Scalable Adversarial Classification [Paper]
- Activation Maximization Generative Adversarial Nets [arXiv]
- AdaGAN: Boosting Generative Models [arXiv]
- Adversarial Autoencoders [arXiv]
- Adversarial Discriminative Domain Adaptation [arXiv]
- Adversarial Generator-Encoder Networks [arXiv]
- Adversarial Feature Learning [arXiv] [Code]
- Adversarially Learned Inference [arXiv] [Code]
- AE-GAN: adversarial eliminating with GAN [arXiv]
- An Adversarial Regularisation for Semi-Supervised Training of Structured Output Neural Networks [arXiv]
- Associative Adversarial Networks [arXiv]
- Autoencoding beyond pixels using a learned similarity metric [arXiv]
- Bayesian Conditional Generative Adverserial Networks [arXiv]
- Bayesian GAN [arXiv]
- BEGAN: Boundary Equilibrium Generative Adversarial Networks [Paper] [arXiv] [Code]
- Binary Generative Adversarial Networks for Image Retrieval [arXiv]
- Boundary-Seeking Generative Adversarial Networks [arXiv] [Code]
- CausalGAN: Learning Causal Implicit Generative Models with Adversarial Training [arXiv]
- Comparison of Maximum Likelihood and GAN-based training of Real NVPs [arXiv]
- Conditional CycleGAN for Attribute Guided Face Image Generation [arXiv]
- Conditional Generative Adversarial Nets [arXiv] [Code]
- LR-GAN: Layered Recursive Generative Adversarial Networks for Image Generation [arXiv]
- MAGAN: Margin Adaptation for Generative Adversarial Networks [arXiv] [Code]
- Maximum-Likelihood Augmented Discrete Generative Adversarial Networks [arXiv]
- McGan: Mean and Covariance Feature Matching GAN [arXiv]
- Message Passing Multi-Agent GANs [arXiv]
- Mode Regularized Generative Adversarial Networks [arXiv] [Code]
- Multi-Agent Diverse Generative Adversarial Networks [arXiv]
- Multi-Generator Gernerative Adversarial Nets [arXiv]
- Objective-Reinforced Generative Adversarial Networks (ORGAN) for Sequence Generation Models [arXiv]
- On the effect of Batch Normalization and Weight Normalization in Generative Adversarial Networks [arXiv]
- On the Quantitative Analysis of Decoder-Based Generative Models [arXiv]
- Optimizing the Latent Space of Generative Networks [arXiv]
- PixelGAN Autoencoders [arXiv]
- SegAN: Adversarial Network with Multi-scale L1 Loss for Medical Image Segmentation [arXiv]
- SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient [arXiv]
- Simple Black-Box Adversarial Perturbations for Deep Networks [Paper]
- Softmax GAN [arXiv]
- Stabilizing Training of Generative Adversarial Networks through Regularization [arXiv]
- Stacked Generative Adversarial Networks [arXiv]
- Statistics of Deep Generated Images [arXiv]
- The Cramer Distance as a Solution to Biased Wasserstein Gradients [arXiv]
- Towards Understanding Adversarial Learning for Joint Distribution Matching [arXiv]
- Training generative neural networks via Maximum Mean Discrepancy optimization [arXiv]
- Triple Generative Adversarial Nets [arXiv]
- Unrolled Generative Adversarial Networks [arXiv]
- Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks [arXiv] [Code] [Code] [Code] [Code] [Code]
- Wasserstein GAN [arXiv] [Code] [Code]

https://github.com/nightrome/really-awesome-gan

Q&A