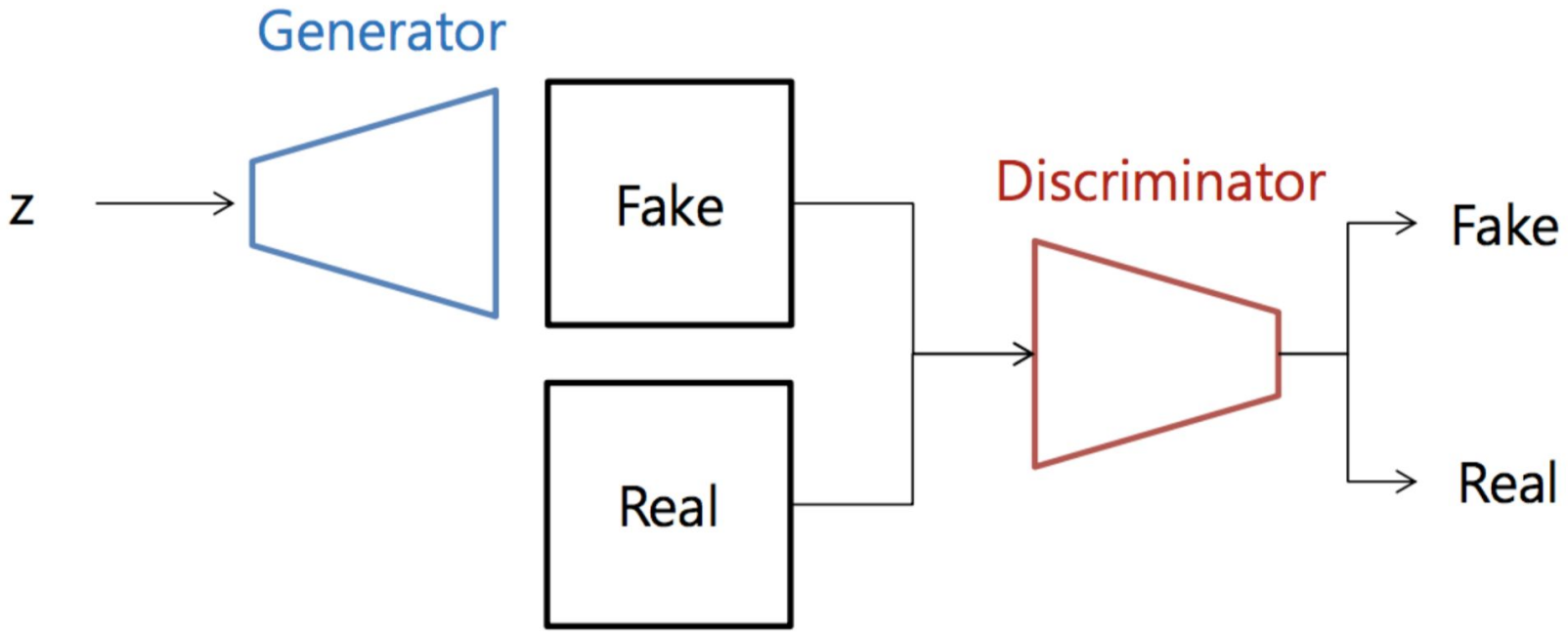
CHAPTER 7

[GAN 때문이야] 3주차
발제자료

손으로 쓴 숫자 훈련

18기 분석 박규연


Generator
z
Fake
Real
Discriminator
Fake
Real

CHAPTER 7

[GAN 때문이야] 3주차
발제자료

1. **판별기**를 **실제 데이터**로 훈련한다.
2. **판별기**를 **생성된 데이터**로 훈련한다.
3. 판별기를 속일 수 있도록 **생성기**를 훈련한다.

```
# 1단계: 참에 대해 판별기 훈련
D.train(generate_real(), torch.FloatTensor([1.0]))

# 2단계: 거짓에 대해 판별기 훈련
# G의 기울기가 계산되지 않도록 detach() 함수를 이용
D.train(G.forward(torch.FloatTensor([0.5])).detach(), torch.FloatTensor([0.0]))

# 3단계: 생성기 훈련
G.train(D, torch.FloatTensor([0.5]), torch.FloatTensor([1.0]))
```

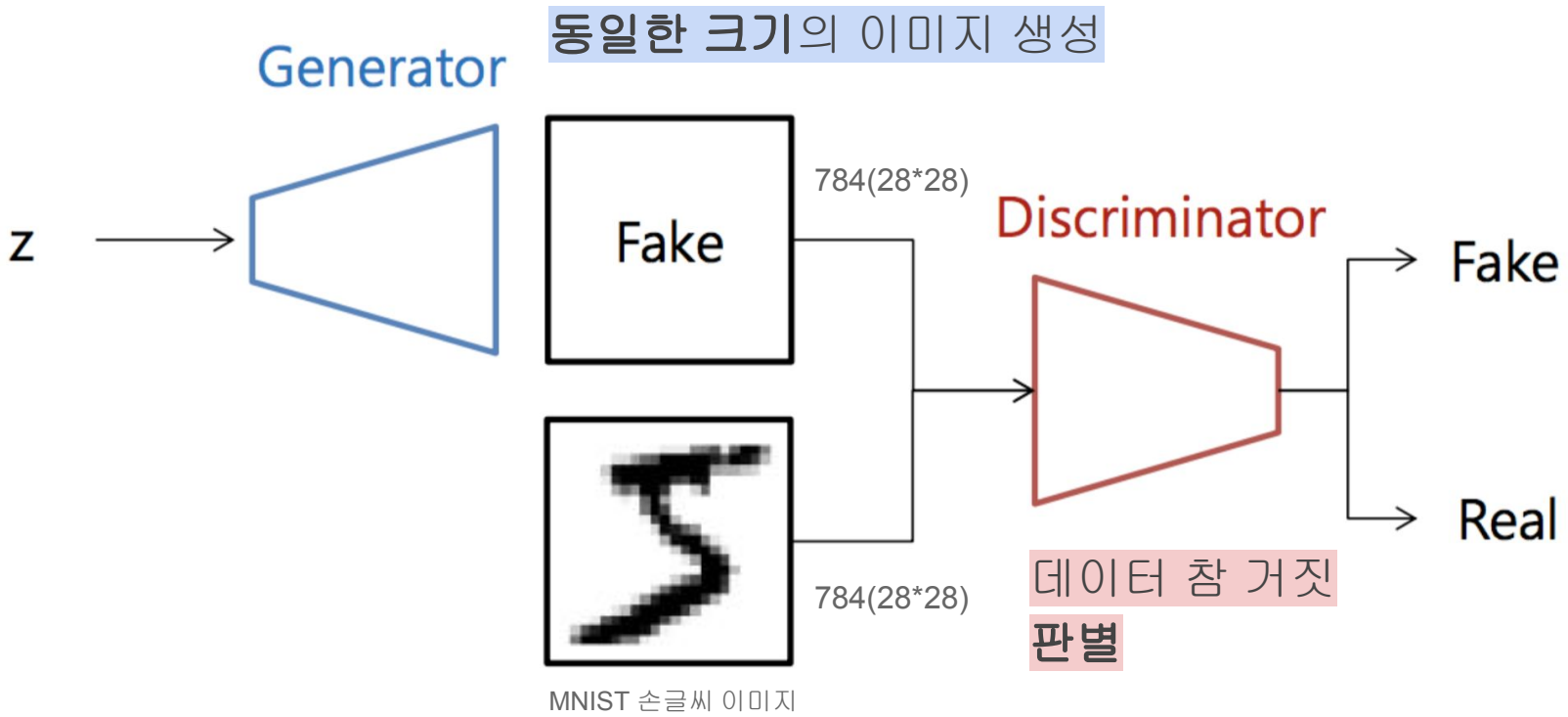
1010 패턴 실습 코드

```
self.model = nn.Sequential(
    nn.Linear(1, 3),
    nn.Sigmoid(),
    nn.Linear(3, 4),
    nn.Sigmoid()
)
```

생성기 신경망

CHAPTER 7

[GAN 때문이야] 3주차
발제자료



CHAPTER 7

[GAN 때문이야] 3주차
발제자료

```
import sklearn.datasets
```

```
mnist = sklearn.datasets.fetch_openml('mnist_784', data_home="mnist_784")
```

mnist.data

	pixel1	pixel2	pixel3	pixel4	pixel5	pixel6	pixel7	pixel8	pixel9	pixel10	...	pixel775	pixel776	pixel777	pixel778	pixel779	pixel780	pixel781	pixel782	pixel783	pixel784
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
69995	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
69996	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
69997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
69998	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
69999	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

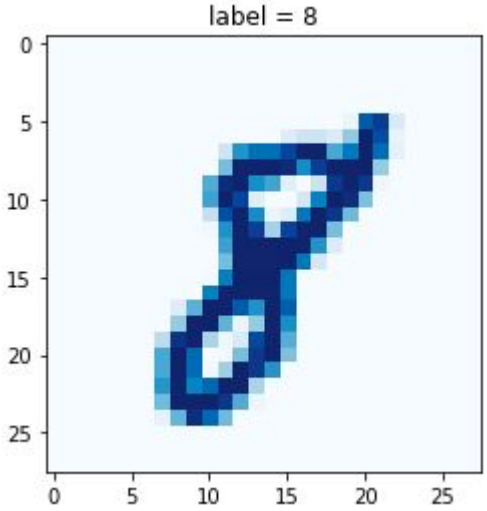
70000 rows × 784 columns

mnist.target

0	5
1	0
2	4
3	1
4	9
	..
69995	2
69996	3
69997	4
69998	5
69999	6

```
Name: class, Length: 70000, dtype: category
Categories (10, object): ['0', '1', '2', '3', ..., '6', '7', '8', '9']
```

```
img = mnist.data.iloc[17].values.reshape(28, 28)
plt.title("label = " + str(mnist.target[17]))
plt.imshow(img, interpolation='none', cmap='Blues')
```

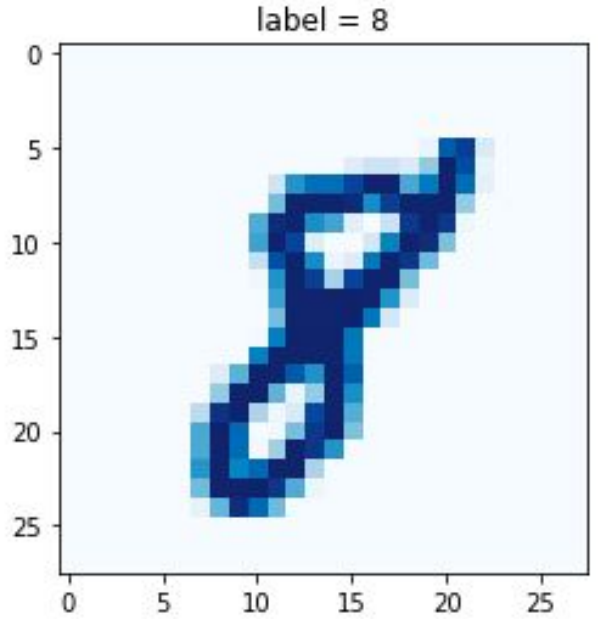


CHAPTER 7

[GAN 때문이야] 3주차
발제자료

```python
class MnistDataset(Dataset):

  def __init__(self, data):
    self.data_set = data
    pass

  def __len__(self):
    return len(self.data_set.data)

  def __getitem__(self, index):
    # 이미지 목표 (레이블)
    label = self.data_set.target[index]
    target = torch.zeros((10))
    target[int(label)] = 1.0   # 길이는 10, 정답만 1로 표시된 텐서, label의 data type이 str이므로 int로 바꿔줘야함

    # 0-255의 이미지를 0-1로 정규화
    image_values = torch.FloatTensor(self.data_set.data.iloc[index].values) / 255.0

    # 레이블, 이미지 데이터 텐서, 목표 텐서 반환
    return label, image_values, target

  def plot_image(self, index):
    img = self.data_set.data.iloc[index].values.reshape(28, 28)
    plt.title("label = " + str(self.data_set.target[index]))
    plt.imshow(img, interpolation='none', cmap='Blues')
    pass

  pass
```

```python
mnist_dataset = MnistDataset(mnist)
mnist_dataset.plot_image(17)
```



CHAPTER 7

[GAN 때문이야] 3주차
발제자료

Custom Dataset

1. 추상 클래스 **상속**
2. 메소드 **오버라이드**

```
from torch.utils.data import Dataset
```

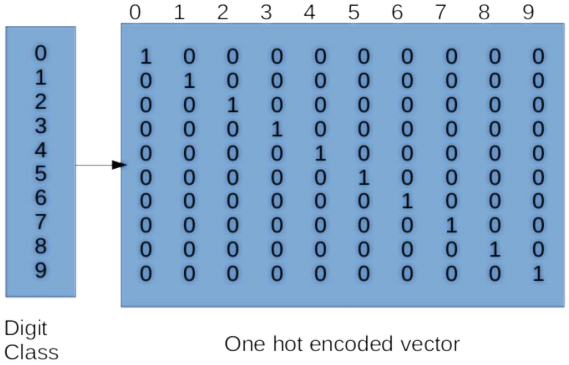
파이토치에서 데이터셋을
제공하는 **추상 클래스**

```
class CustomDataset(torch.utils.data.Dataset):
  def __init__(self):
  데이터셋의 전처리를 해주는 부분

  def __len__(self):
  데이터셋의 길이. 즉, 총 샘플의 수를 적어주는 부분

  def __getitem__(self, idx):
  데이터셋에서 특정 1개의 샘플을 가져오는 함수
```

CHAPTER 7

[GAN 때문이야] 3주차
발제자료

```python
def __init__(self, data):
  self.data_set = data
  pass

def __len__(self):
  return len(self.data_set.data)

def __getitem__(self, index):
  # 이미지 목표 (레이블)
  label = self.data_set.target[index]
  # 길이는 10, 정답만 1.0으로 표시된 텐서
  target = torch.zeros((10))
  target[int(label)] = 1.0

  # 0-255의 이미지를 0-1로 정규화
  image_values = torch.FloatTensor(self.data_set.data.iloc[index].values) / 255.0

  # 레이블, 이미지 데이터 텐서, 목표 텐서 반환
  return label, image_values, target
```




```python
def plot_image(self, index):
  img = self.data_set.data.iloc[index].values.reshape(28, 28)
  plt.title("label = " + str(self.data_set.target[index]))
  plt.imshow(img, interpolation='none', cmap='Blues')
  pass
```

CHAPTER 7

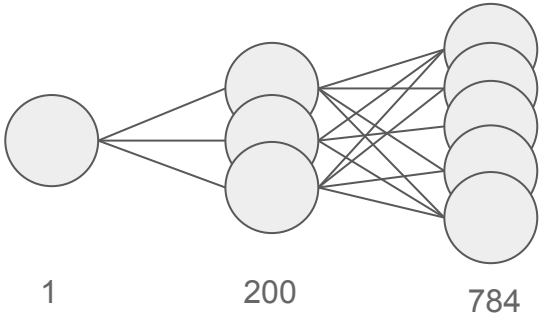
[GAN 때문이야] 3주차
발제자료

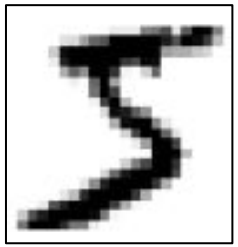
Generator

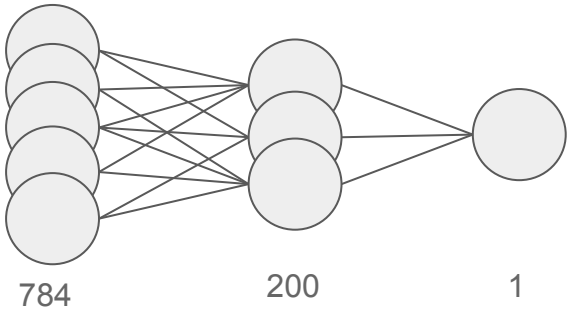
```
self.model = nn.Sequential(
    nn.Linear(1, 200),
    nn.Sigmoid(),
    nn.Linear(200, 784),
    nn.Sigmoid()
)
```

Discriminator

```
self.model = nn.Sequential(
    nn.Linear(784, 200),
    nn.Sigmoid(),
    nn.Linear(200, 1),
    nn.Sigmoid()
)
```







CHAPTER 7

[GAN 때문이야] 3주차 발제자료

1010 패턴 실습 코드

```
# 1단계: 참에 대해 판별기 훈련
D.train(generate_real(), torch.FloatTensor([1.0]))

# 2단계: 거짓에 대해 판별기 훈련
# G의 기울기가 계산되지 않도록 detach() 함수를 이용
D.train(G.forward(torch.FloatTensor([0.5])).detach(), torch.FloatTensor([0.0]))

# 3단계: 생성기 훈련
G.train(D, torch.FloatTensor([0.5]), torch.FloatTensor([1.0]))
```

매 훈련마다 **임의적인 입력**

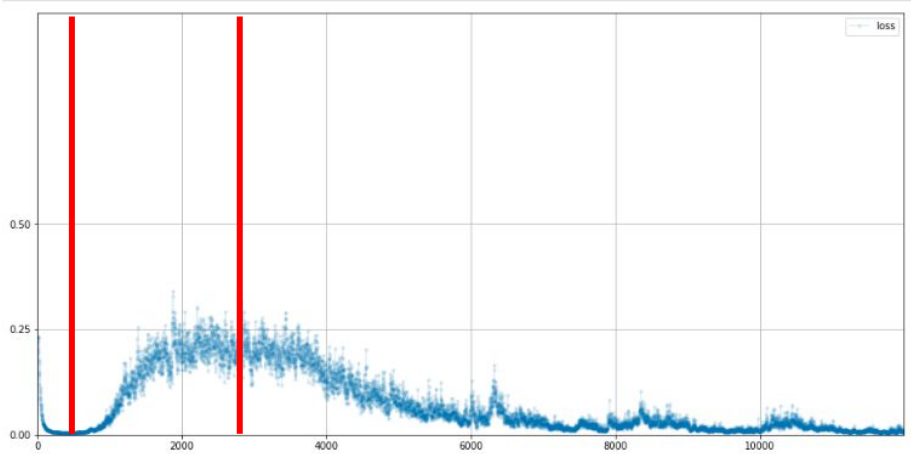
```
def generate_random(size):
  random_data = torch.rand(size) # 0-1사이 랜덤값
  return random_data
```

MNIST 실습 코드

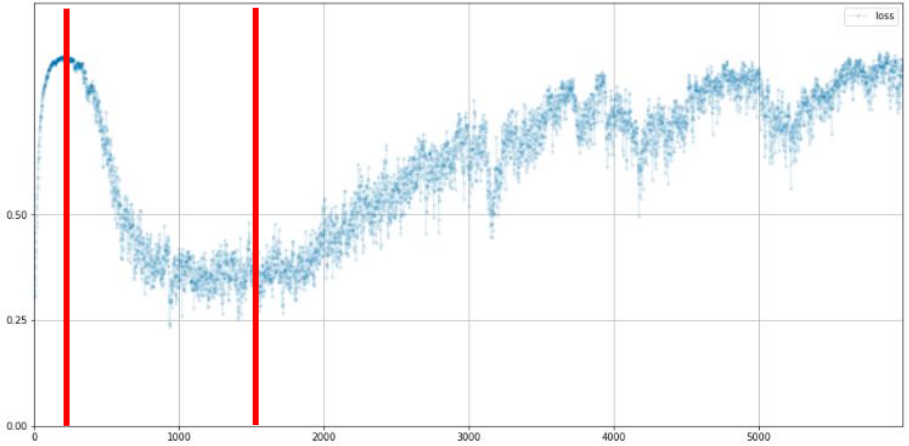
```
# train discriminator on true
D.train(image_data_tensor, torch.FloatTensor([1.0]))

# train discriminator on false
# use detach() so gradients in G are not calculated
D.train(G.forward(generate_random(1)).detach(), torch.FloatTensor([0.0]))

# train generator
G.train(D, generate_random(1), torch.FloatTensor([1.0]))
```

CHAPTER 7

[GAN 때문이야] 3주차
발제자료



판별기 학습곡선



생성기 학습곡선

판별기 우수 → 판별기와 생성기 **균형** → 판별기 우수

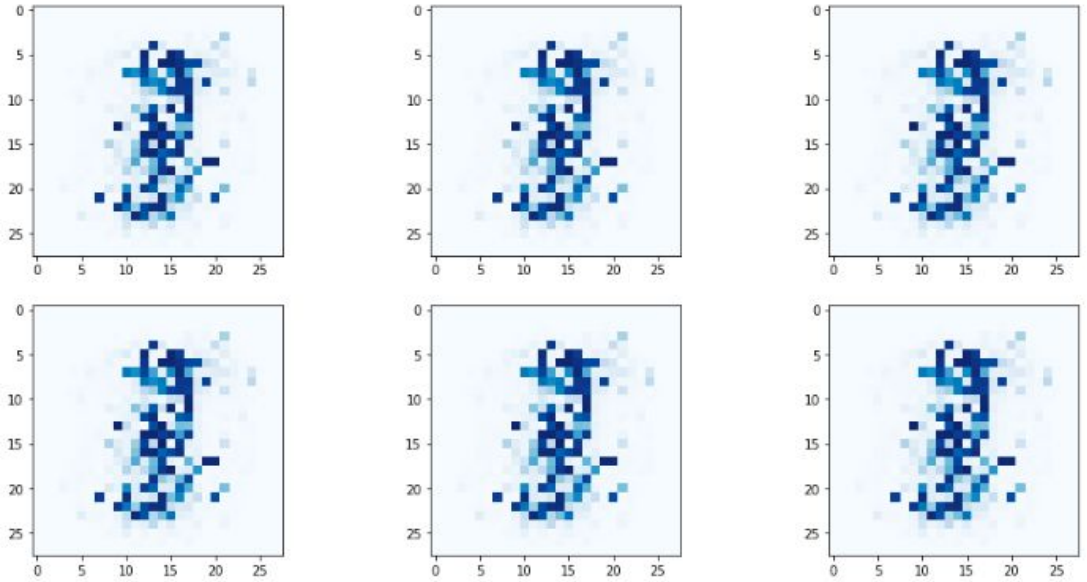
손실 약 0.25

CHAPTER 7

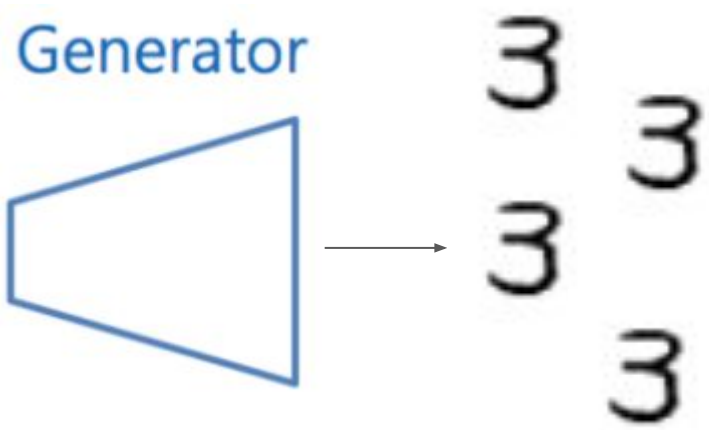
[GAN 때문이야] 3주차
발제자료

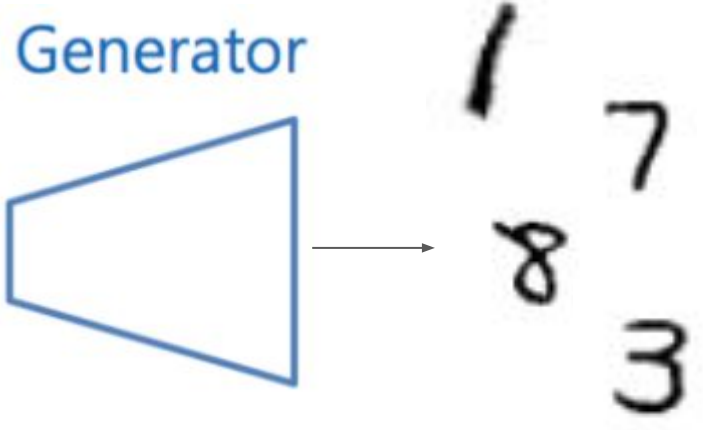
```
f, axarr = plt.subplots(2,3, figsize=(16,8))
for i in range(2):
    for j in range(3):
        output = G.forward(generate_random(1))
        img = output.detach().numpy().reshape(28,28)
        axarr[i,j].imshow(img, interpolation='none', cmap='Blues')
        pass
    pass
```

서로 다른 임의 시드에서
각기 다른 이미지가
생성될 것으로 예상했으나...



Mode collapse





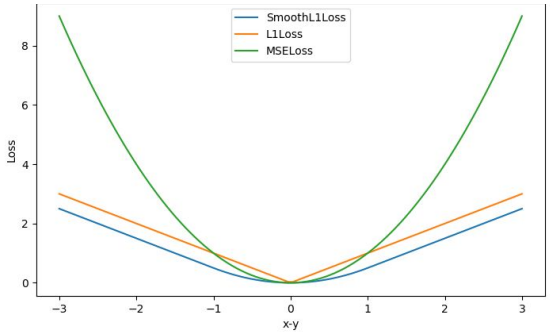
CHAPTER 7

[GAN 때문이야] 3주차
발제자료

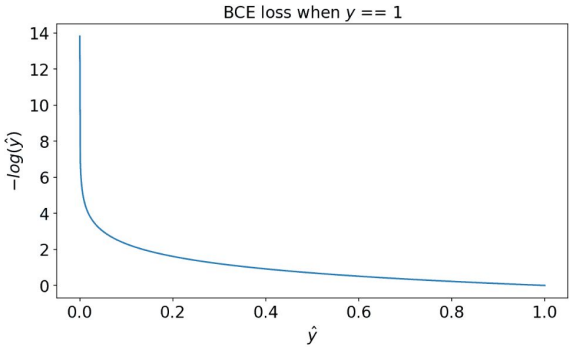
GAN 훈련 성능 향상 - Loss function

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$



MSELoss → 0.25

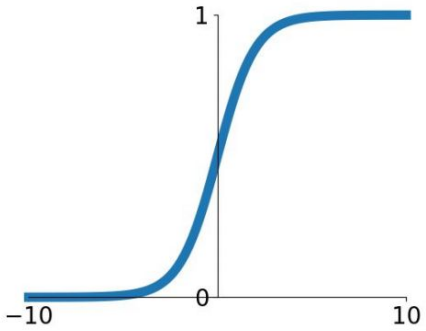


BCELoss → ln(2)

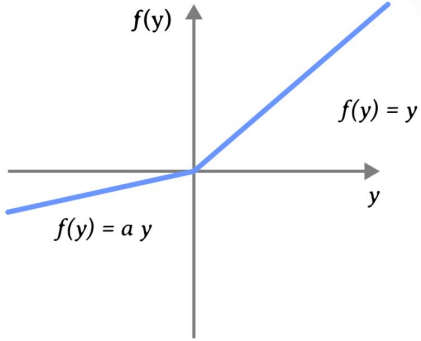
CHAPTER 7

[GAN 때문이야] 3주차
발제자료

GAN 훈련 성능 향상 - Activation function

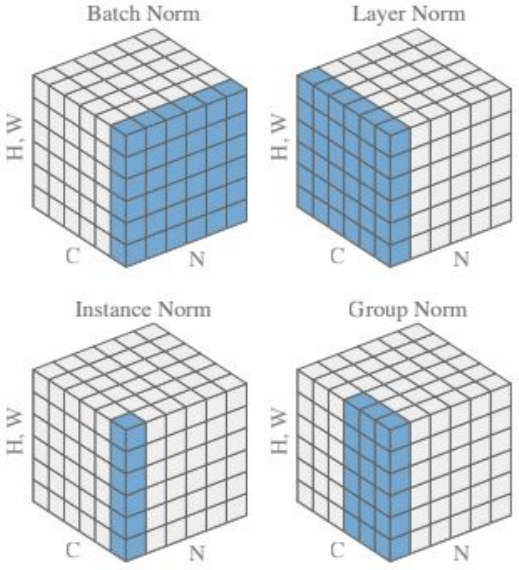


Sigmoid



LeakyReLU

Norm



CHAPTER 7

[GAN 때문이야] 3주차
발제자료

Generator

```
self.model = nn.Sequential(
    nn.Linear(1, 200),
    nn.LeakyReLU(0.02),

    nn.LayerNorm(200),

    nn.Linear(200, 784),
    nn.Sigmoid()
)
```

Discriminator

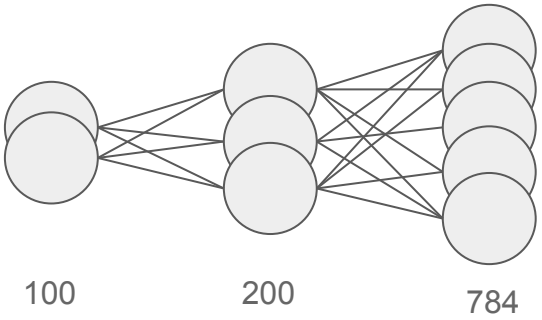
```
self.model = nn.Sequential(
    nn.Linear(784, 200),
    nn.LeakyReLU(0.02),

    nn.LayerNorm(200),

    nn.Linear(200, 1),
    nn.Sigmoid()
)
```

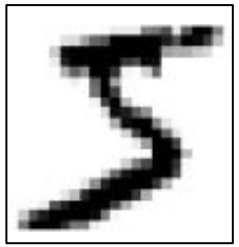
```
self.optimizer = torch.optim.Adam(self.parameters(), lr=0.0001)
```

CHAPTER 7

[GAN 때문이야] 3주차
발제자료

시드 변화 - 시드 늘리기





```
self.model = nn.Sequential(
    nn.Linear(100, 200),
    nn.LeakyReLU(0.02),

    nn.LayerNorm(200),

    nn.Linear(200, 784),
    nn.Sigmoid()
)
```

CHAPTER 7

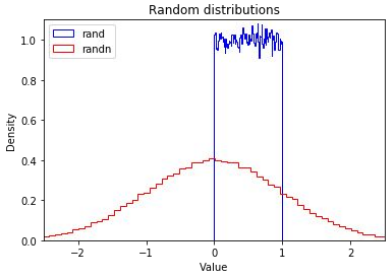
[GAN 때문이야] 3주차
발제자료

시드 변화 - 시드 형태 변경

```
def generate_random(size):
  random_data = torch.rand(size)
  return random_data
```

평균이 0,
분산이 제한된
값들이
학습에 유리

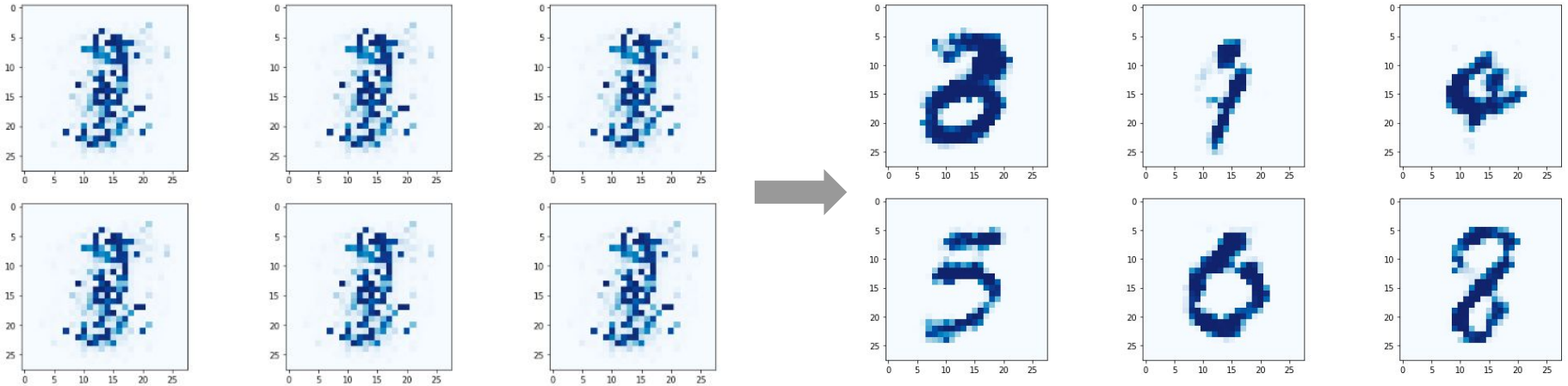
```
def generate_random_seed(size):
  random_data = torch.randn(size)
  return random_data
```




```
# 참에 대해 판별기 훈련
D.train(image_data_tensor, torch.FloatTensor([1.0]))

# 거짓에 대해 판별기 훈련
# G의 기울기가 계산되지 않도록 detach() 함수를 이용
D.train(G.forward(generate_random_seed(100)).detach(), torch.FloatTensor([0.0]))


# 생성기 훈련
G.train(D, generate_random_seed(100), torch.FloatTensor([1.0]))
```

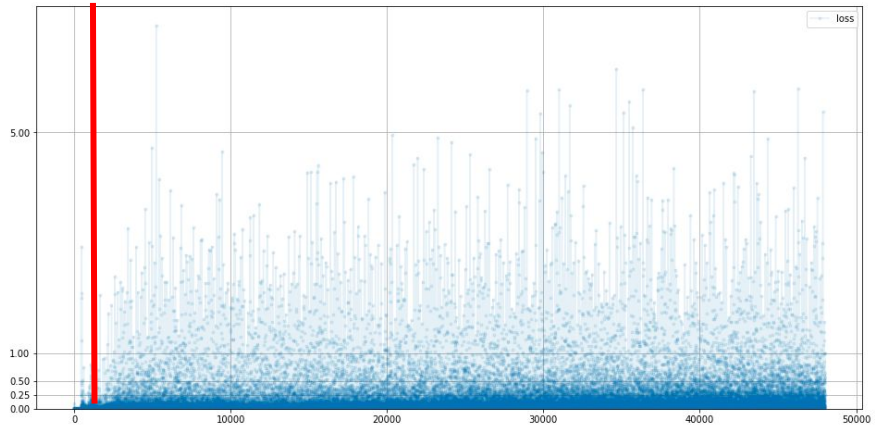
CHAPTER 7

[GAN 때문이야] 3주차
발제자료

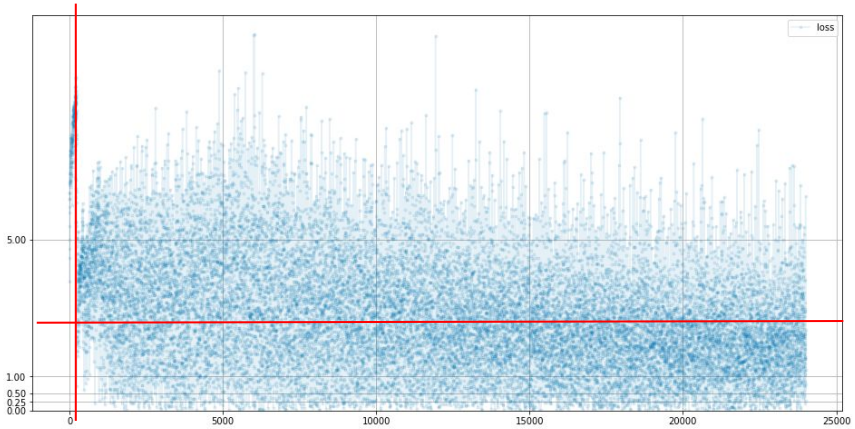


CHAPTER 7

[GAN 때문이야] 3주차
발제자료



판별기 학습곡선



생성기 학습곡선

판별기 우수 → 판별기와 생성기 아직은... **불안정**

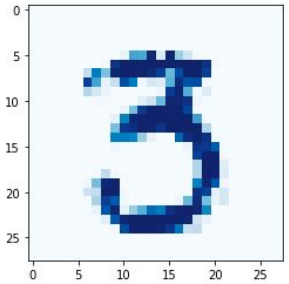
CHAPTER 7

[GAN 때문이야] 3주차
발제자료

시드 실험

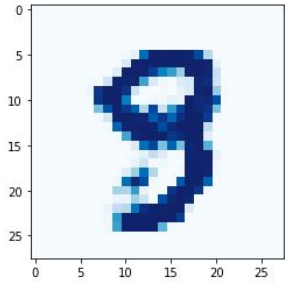
```
seed1 = generate_random_seed(100)
out1 = G.forward(seed1)
img1 = out1.detach().numpy().reshape(28,28)
plt.imshow(img1, interpolation='none', cmap='Blues')
```

seed1



```
seed2 = generate_random_seed(100)
out2 = G.forward(seed2)
img2 = out2.detach().numpy().reshape(28,28)
plt.imshow(img2, interpolation='none', cmap='Blues')
```

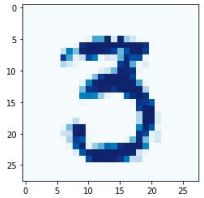
seed2

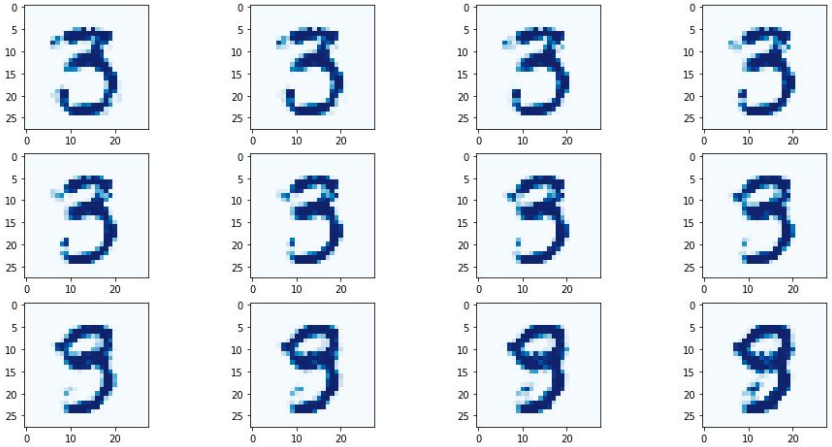


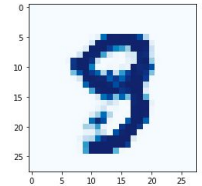
CHAPTER 7

[GAN 때문이야] 3주차
발제자료

시드 실험 - 두 시드 사이값





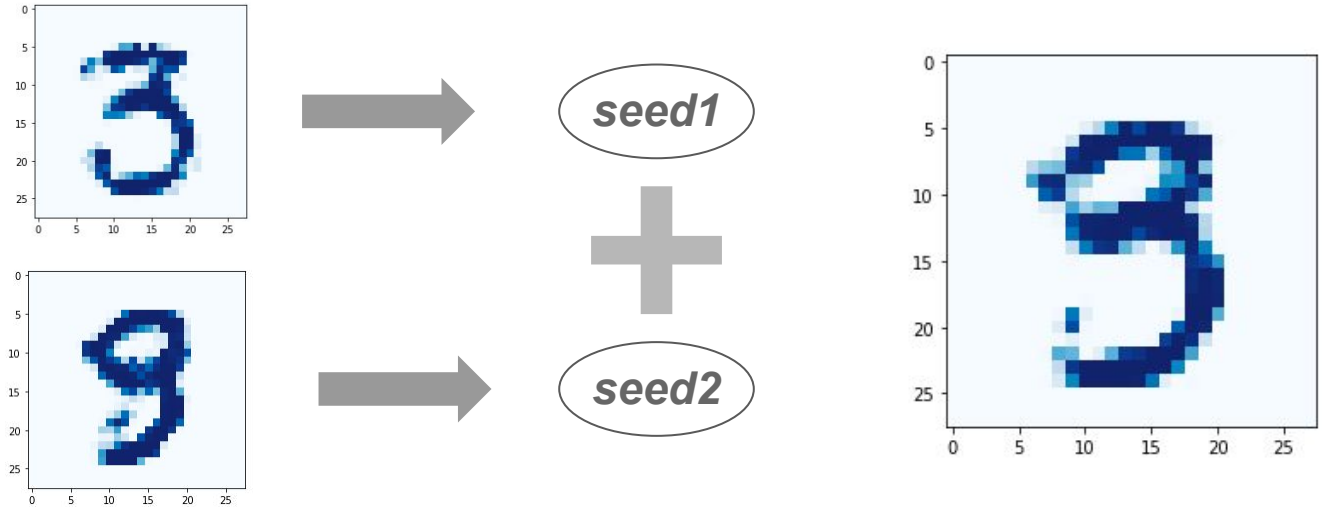


seed1

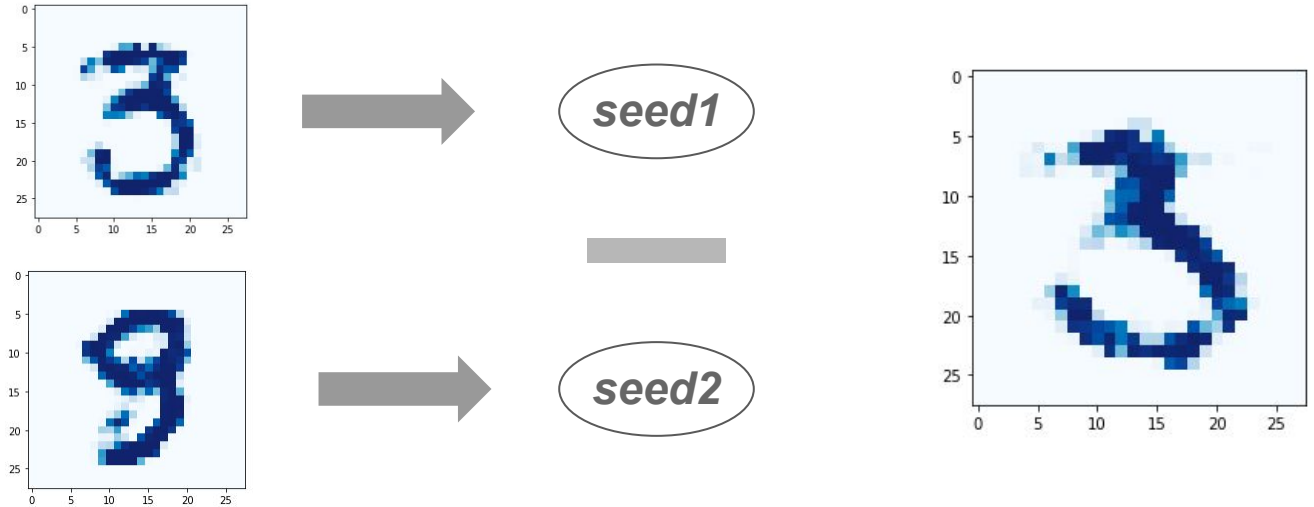
. . . . . .

seed2

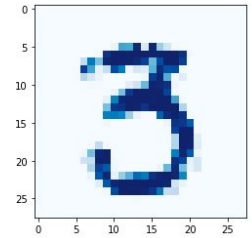
시드 실험 - 두 시드 더하기

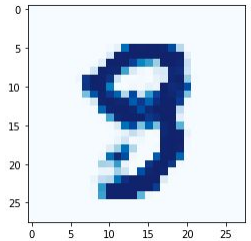


시드 실험 - 두 시드 빼기



시드 실험 - 두 시드 곱하기

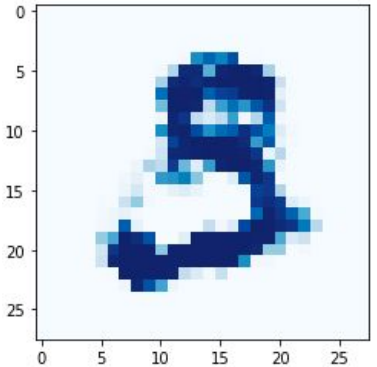




seed1

×

seed2



감사합니다