합성곱 Autoencoder 실습

임세진

https://youtu.be/obLzUWAcUMs

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB

01. Autoencoder

02. 모델 구조

03. 실습

01. Autoencoder

- **Autoencoder**

- 정답 없이 모델을 학습시키는 비지도 학습(Unsupervised learning) 모델

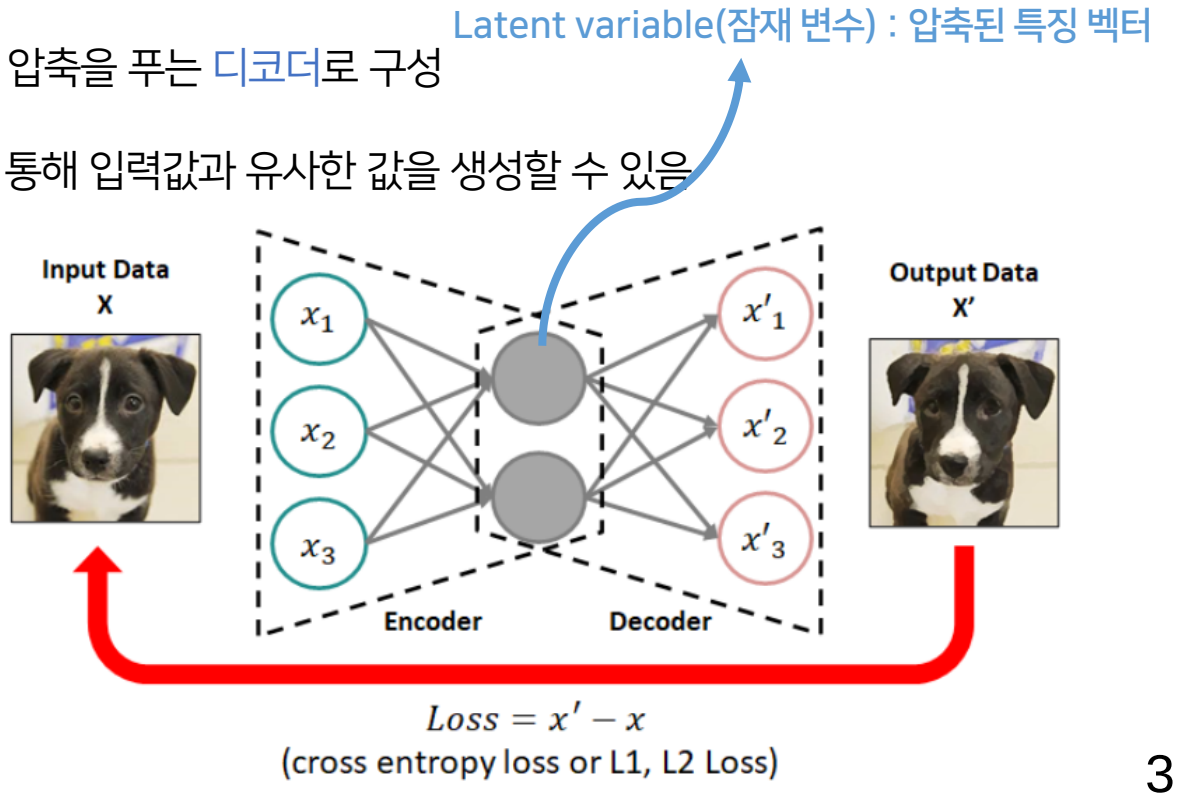
  지도학습 (Supervised learning) : [데이터 : 데이터 라벨] 쌍이 미리 정의된 데이터로 학습하는 경우

  비지도학습 (Unsupervised learning) : [데이터 : 데이터 라벨] 쌍이 정의되지 않은 데이터로 학습하는 경우

- 입력과 출력이 같은 구조 (대칭형 구조) : 입력 데이터를 압축하는 인코더 + 압축을 푸는 디코더로 구성

- 인코더를 통해 차원 축소가 된 잠재 변수로 별도의 계산을 하거나 디코더를 통해 입력값과 유사한 값을 생성할 수 있음

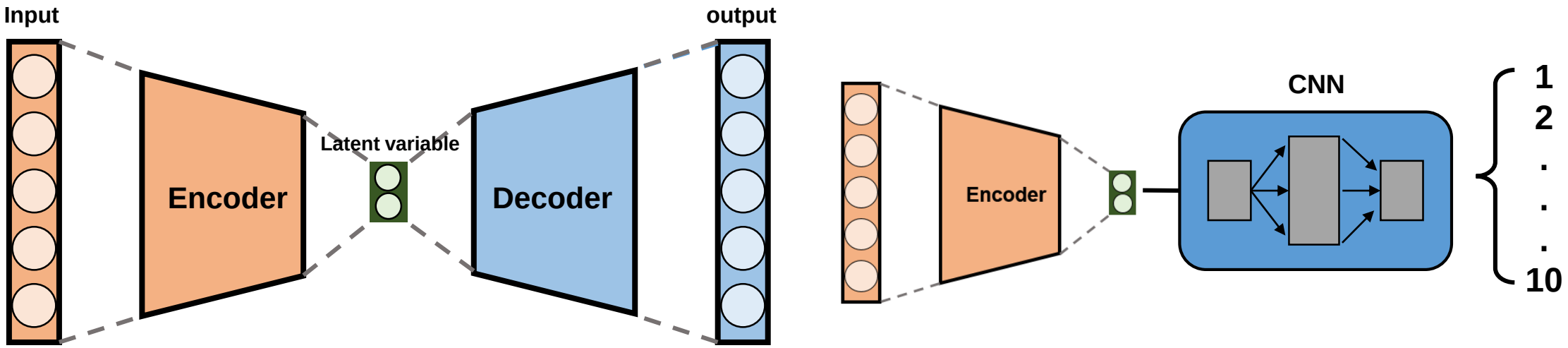
- AE의 종류
  - 스택 AE : 층을 여러 개 쌓았다는 의미 (기본 AE)
  - 디노이징 AE : 노이즈를 제거한 데이터 생성 or 이미지 복원
  - 합성곱 AE : Linear Layer 대신 합성곱 계층을 사용하는 구조



02. 모델 구조

- 데이터셋 : MNIST 데이터

(audio > image, animal 등 시도했는데 차원이 너무 커서 다루기 어려운 경우, loss가 너무 크게 나오는 등.. 문제가 많았음..)



02. 모델 구조

- 해당 구조를 사용함으로써 얻는 이점

  Encoder를 통해 특징 추출한 것을 입력으로 받기 때문에 feature 수를 줄일 수 있음

  ➔ 모바일에 배포 시 일반 CNN 모델 < Encoder + CNN 모델이 효율적임

  ➔ (이미지 input보다 작은 feature로도 학습이 가능하기 때문)

03. 실습

```python
class Encoder(nn.Module):
    def __init__(self):
        super(Encoder, self).__init__()

        k = 16
        self.encoder = nn.Sequential(
                        nn.Conv2d(1, k, 3, stride=2),
                        nn.ReLU(),
                        nn.Conv2d(k, 2*k, 3, stride=2),
                        nn.ReLU(),
                        nn.Conv2d(2*k, 4*k, 3, stride=1),
                        nn.ReLU(),
                        nn.Flatten(),
                        nn.Linear(1024, 10),
                        nn.ReLU()
        )

    def forward(self, x):

        encoded = self.encoder(x)

        return encoded
```

(Batch_size , 1024)

Latent variable

```python
class Decoder(nn.Module):
    def __init__(self):
      super(Decoder, self).__init__()
      k = 16

      self.decoder = nn.Sequential(
                        nn.Linear(10, 1024),
                        nn.ReLU(),
                        nn.Deflatten(4*k),
                        nn.ConvTranspose2d(4*k, 2*k, 3, stride=1),
                        nn.ReLU(),
                        nn.ConvTranspose2d(2*k, k, 3, stride=2),
                        nn.ReLU(),
                        nn.ConvTranspose2d(k, 1, 3, stride=2,output_padding=1),
                        nn.Sigmoid()
                        )

    def forward(self, x):

        decoded = self.decoder(x)

        return decoded
```

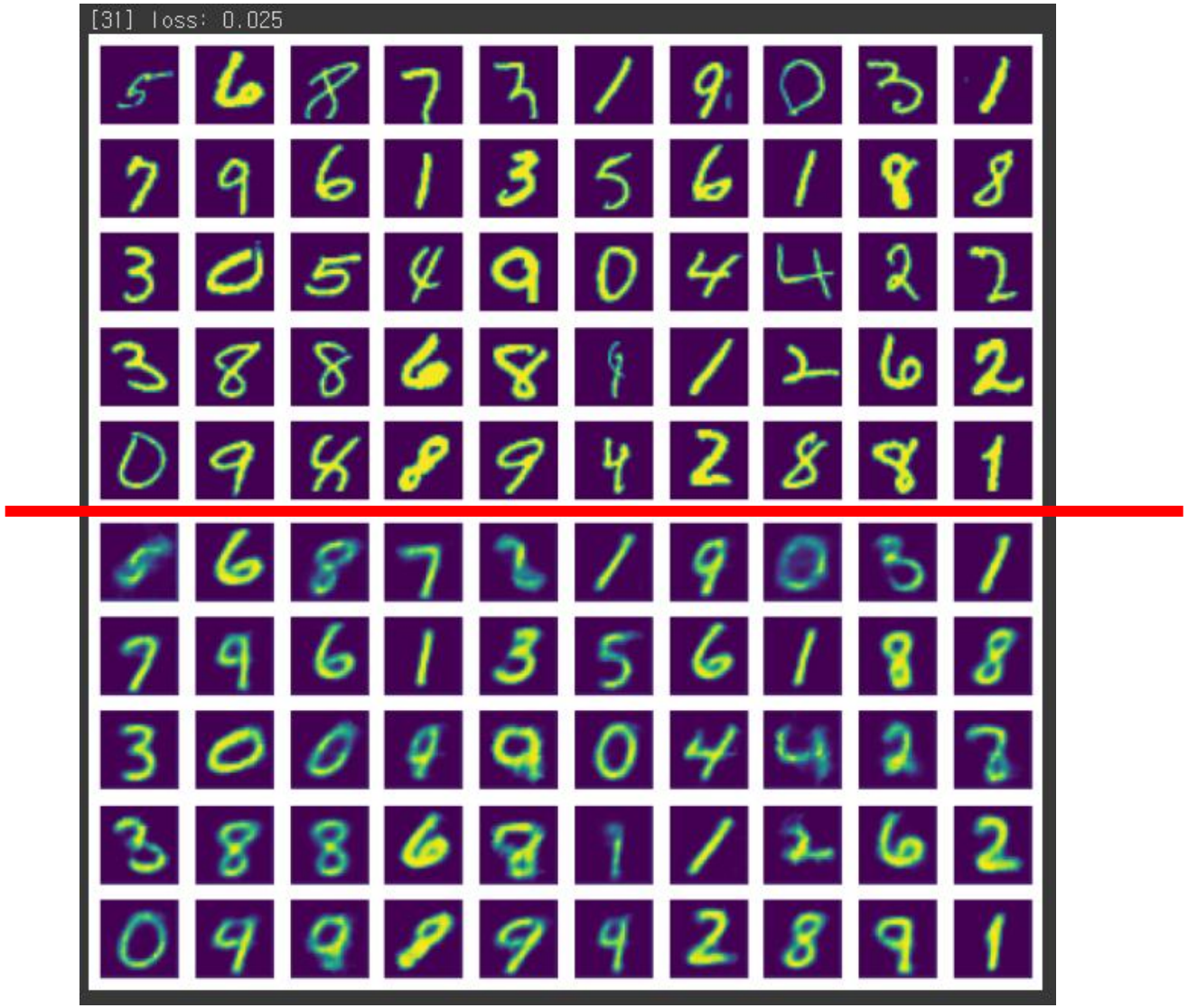
2D -> 4D

업샘플링시 사용하는 함수

03. 실습

```
class Autoencoder(nn.Module):
    def __init__(self, encoder, decoder):
      super(Autoencoder, self).__init__()
      self.encoder = encoder
      self.decoder = decoder

    def forward(self, x):
        encoded = self.encoder(x)
        decoded = self.decoder(encoded)

        return decoded

encoder = Encoder().to(device)
decoder = Decoder().to(device)
model = Autoencoder(encoder, decoder).to(device)
```

03. 실습

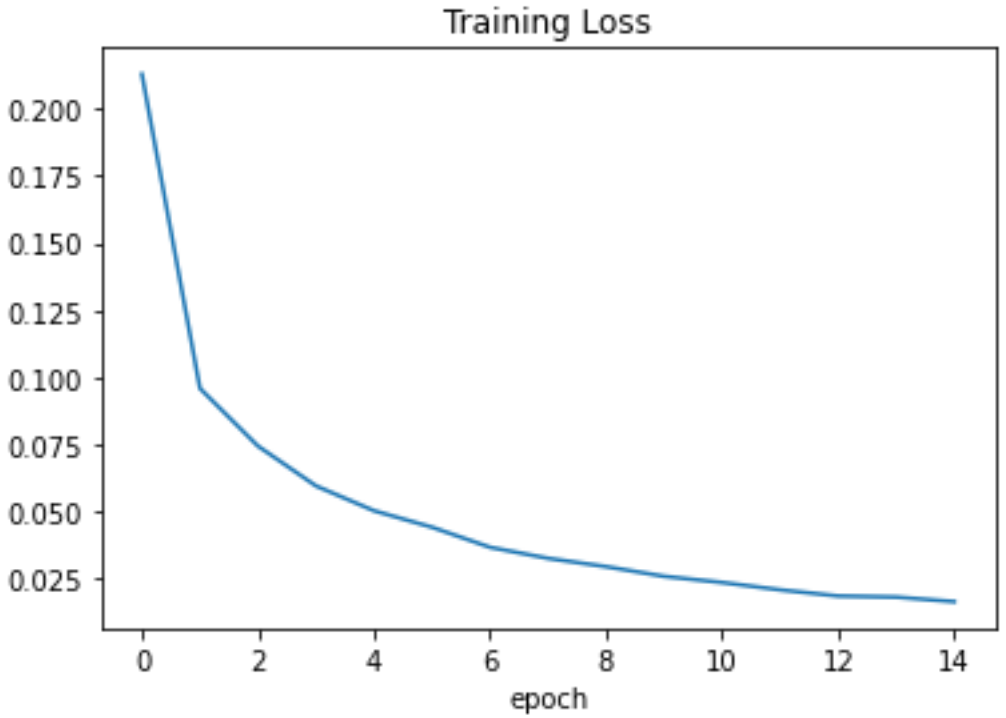


03. 실습

```python
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()

        # batch_size = 50
        # feature = 원래크기 - 필터크기 +1
        self.cnn = nn.Sequential(
                        nn.Linear(10, 1024),
                        nn.ReLU(),
                        Deflatten(64),
                        nn.Conv2d(64, 128, 1, 1),
                        nn.ReLU(),
                        nn.MaxPool2d(2, 2),
                        nn.Conv2d(128, 256, 1, 1),
                        nn.ReLU(),
                        Flatten(),
                        nn.Linear(1024, 10)
        )

    def forward(self, x):
        x = self.cnn(x)

        return x
```

```python
class AutoCNN(nn.Module):
    def __init__(self, encoder, cnn):
        super(AutoCNN, self).__init__()
        self.encoder = encoder
        self.cnn = cnn

    def forward(self, x):
        latent = self.encoder(x)
        output = self.cnn(latent)
        return output
```

03. 실습



분류 정확도 : 98%

Q & A