

Generative Deep Learning

# 3 Autoencoder

Yeseul Oh

### autoencoder

- **encoder**와 **decoder**로 구성된 신경망
- 입력데이터가 **encoder**로 들어가서 압축 후 **decoder**로 나오면서 다시 복원(재구성)하는 신경망 구조
- 이미 가지고 있는 입력데이터를 재구성하는게 의미 있음?
  - latent space에서 원본 아이템의 임베딩이 없는 위치에서 샘플링하여 새로운 아이템 생성 가능 -> 생성 모델

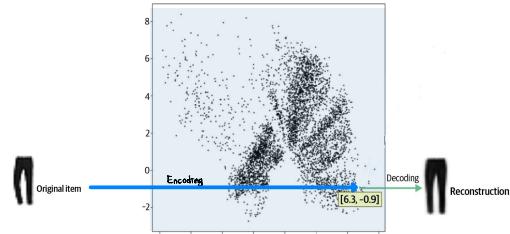


### latent(embedding) space

- 무한 옷장
- 저차원 공간
- 차원  $\propto$  자유도 (실제 2D 이상, ex: 잡음 제거 오토인코더)

### encoder

- 패션스타 나
- encoding
  - 옷장의 특정위치를 알려줌
  - 고차원 입력 데이터를 latent space 안의 저차원 임베딩 벡터에 압축
  - 옷장의 빈곳의 위치를 알려줄 경우



### decoder

- 스타일리스트 브라이언
- decoding
  - 옷장의 특정위치를 받아 해당 아이템을 새로 만듦: 재구성
  - latent space의 포인트(임베딩 벡터)를 유효한 원본 도메인으로 변환하는 방법을 학습하여 원본 도메인으로 압축 해제
  - 완전히 새로운 옷을 만들

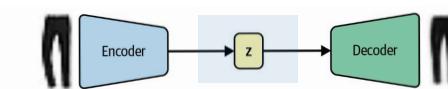


Figure 3-4. Autoencoder architecture diagram

```

# Encoder
class Encoder(nn.Module): # 인코더 클래스 정의
    def __init__(self, latents):
        super().__init__()
        self.latents = latents # 임베딩 크기: 2
        self.model = nn.Sequential( # 순차적 모델 정의
            nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3, stride=2, padding=1), # 첫 번째 conv2d 레이어
            nn.ReLU(),
            nn.Conv2d(in_channels=32, out_channels=64, kernel_size=3, stride=2, padding=1), # 두 번째 conv2d 레이어
            nn.ReLU(),
            nn.Conv2d(in_channels=64, out_channels=128, kernel_size=3, stride=2, padding=1), # 세 번째 conv2d 레이어
            nn.ReLU(),
            nn.Flatten(), # 평탄화
            nn.Linear(in_features=2048, out_features=self.latents) # fully connected layer
        )

    def forward(self, x): # 순전파 함수 정의
        return self.model(x) # 입력 x를 모델에 전달하여 출력 반환

```

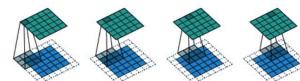


Figure 2-12. A  $3 \times 3 \times 3$  kernel (gray) being passed over a  $5 \times 5 \times 1$  input image (blue), with padding = "same" and strides = 1, to generate the  $5 \times 5 \times 1$  output (green)

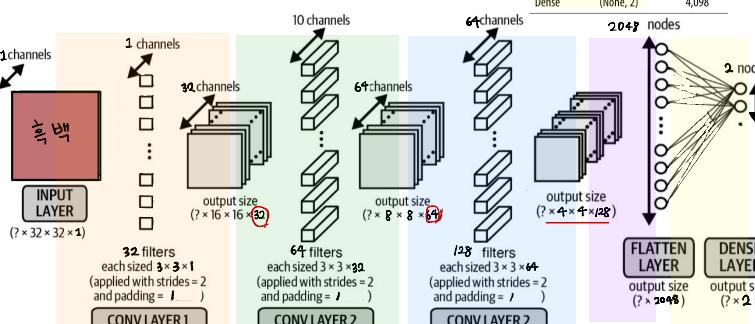


Figure 2-13. A diagram of a convolutional neural network

```

# Decoder
class Decoder(nn.Module): # 디코더 클래스 정의
    def __init__(self, latents):
        super().__init__()
        self.latents = latents # 임베딩 크기: 2
        self.fc = nn.Linear(self.latents, 2048) # fully connected layer

```

```

        self.model = nn.Sequential( # 순차적 모델 정의
            nn.ConvTranspose2d(in_channels=128, out_channels=128,
                kernel_size=3, stride=2, padding=1, output_padding=1), # 첫 번째 convtranspose2d 레이어
            nn.ReLU(),
            nn.ConvTranspose2d(in_channels=128, out_channels=64,
                kernel_size=3, stride=2, padding=1, output_padding=1), # 두 번째 convtranspose2d 레이어
            nn.ReLU(),
            nn.ConvTranspose2d(in_channels=64, out_channels=32,
                kernel_size=3, stride=2, padding=1, output_padding=1), # 세 번째 convtranspose2d 레이어
            nn.ReLU(),
            nn.Conv2d(in_channels=32, out_channels=1, kernel_size=3, stride=1, padding=1) # 출력 conv2d 레이어
        )

```

```

    def forward(self, x): # 순전파 함수 정의
        x = self.fc(x) # fully connected layer 통과
        x = x.reshape(x.shape[0], 128, 4, 4) # 텐서 형태 재조정
        x = self.model(x) # 모델에 입력 x 전달하여 출력 반환
        return x # 최종 출력 반환

```

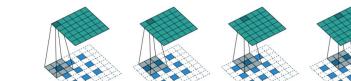


Figure 3-5. A convolutional transpose layer example (source: Dumoulin and Visin, 2018)<sup>2</sup>

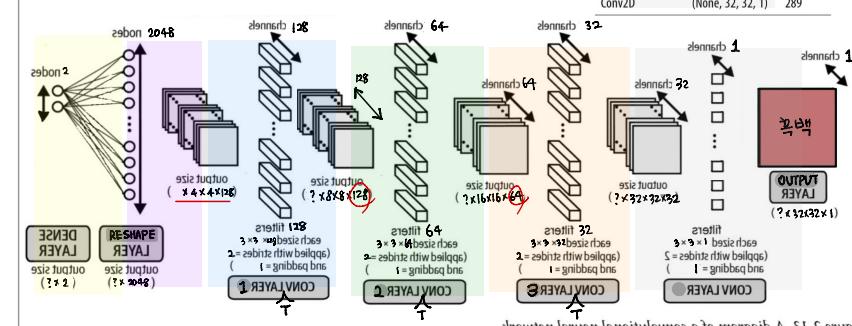


Figure 3-13. A diagram of a convolutional neural network for image generation

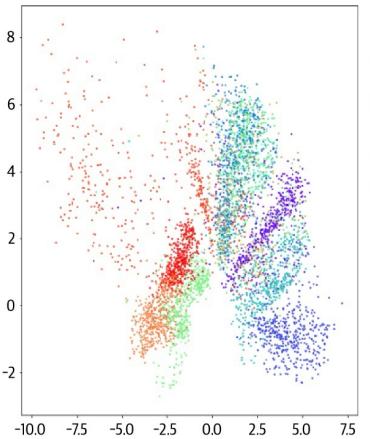


Figure 3-7.

Table 3-3. The Fashion-MNIST labels

ID	Clothing label	Color
0	T-shirt/top	0
1	Trouser	1
2	Pullover	2
3	Dress	3
4	Coat	4
5	Sandal	5
6	Shirt	6
7	Sneaker	7
8	Bag	8
9	Ankle boot	9

- 훈련 중 모델에 의류 레이블 제공 x
- 비슷한 아이템을 latent space의 같은 부분에 그룹화

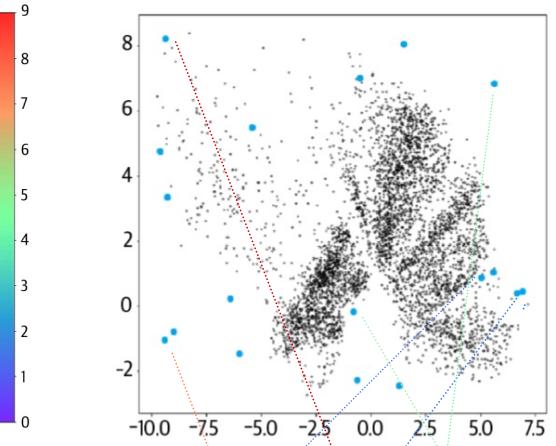


Figure 3-8.

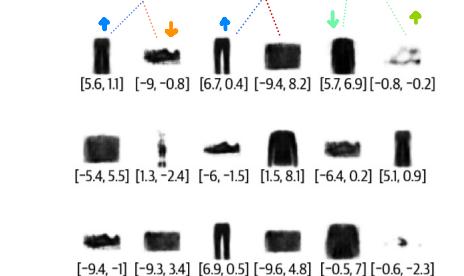


Figure 3-9. • latent space 위에 일정 간격으로 임베딩을 디코딩한 이미지들

- 디코더가 항상 만족스러운 이미지를 생성하지 못하는 이유
  - 임베딩 공간의 크기 문제  
주황색(가방)영역이 빨간색(앵글부츠)영역보다 크기 때문에 무작위로 포인트를 샘플링하면 가방같은 이미지를 디코딩할 가능성이 큼
  - 포인트 선택의 불확실성  
포인트의 분포가 정의되지 않았기 때문에 latent space에서 어떤 포인트를 어떻게 선택해야 할지 명확하지 않음
  - latent space의 빈 구멍 문제  
원본 이미지가 인코딩되지 않은 빈 공간(예: 가장자리)들이 있기 때문에 이 지역의 포인트를 적절히 디코딩 하지 못할 가능성이 높음, 또한 latent space가 연속적이지 않기 때문에 중앙 부근에 있는 점이라 고 하더라도 제대로 디코딩 되지 않을 수 있음  
--> VAE: AE의 세가지 문제 해결

- latent space의 파란색 포인트는 실제 임베팅 벡터로 매핑되지 않은 빈 공간의 포인트
- 디코더가 일부 아이템은 만족스러운 이미지를 생성하지 못함
- 이 latent space의 분포 특징
  - 일부 아이템은 작은 영역, 일부 아이템은 넓은 영역
  - 포인트 (0,0)에 대해 분포가 대칭 x
  - 포인트가 거의 없는 색상 사이에는 간격이 큼  
--> latent space에서 샘플링하는 작업이 어려움