

강의계획표

주	해당 장	주제
1	1장	머신러닝이란
2	2장, 3장	머신러닝을 위한 기초지식, 구현을 위한 도구
3	4장	선형 회귀로 이해하는 지도학습
4	5장	분류와 군집화로 이해하는 지도 학습과 비지도 학습
5	6장	다양한 머신러닝 기법들
6		- 다항 회귀, Logistic Regression
7		- 정보이론, 결정트리 - SVM, Ensemble
8		중간고사 (04-20)
9	7장	인공 신경망 기초 - 문제와 돌파구
10	8장	고급 인공 신경망 구현
11	9장	신경망 부흥의 시작, 합성곱 신경망
12	10장	순환 신경망
13	11장	차원축소와 매니폴드 학습
14	12장 보강주	오토인코더와 잠재표현 학습 AI의 현재와 미래
15		
16		기말고사

12장 오토인코더와 잠재 표현 학습

14 주차

- 신경망을 통해 데이터의 특징을 찾아내는 방법
- 데이터의 특징을 유지한 잠재표현으로 바꾸고 원 데이터를 복원하는 모델
- 오토인코드를 통한 특징 추출
- 잠재표현과 오토인코드를 통한 차원축소와 복원
- 잠재표현을 이용한 새로운 데이터 생성 방법

- 참고문헌
 - [\[정리노트\] \[AutoEncoder의 모든것\] Chap3. AutoEncoder란 ...](#)
 - [08. 오토인코더 \(AutoEncoder\)](#)
 - [AutoEncoder\(오토인코더\) - MNIST dataset 사용 - 별준 코딩](#)
 - [케라스로 이해하는 Autoencoder | Keras for Everyone](#)

특징, 그리고 잠재 표현 학습 (1)

❖ 차원 축소 dimensionality reduction

- 데이터를 표현하기 위해 사용하는 공간을 더 낮은 차원으로 바꾸는 것
- 차원의 저주를 피함
- CNN은 합성곱 계층과 풀링 계층을 거치면서 불필요한 정보는 제거불필요한 정보는 제거하고 필요한 **특징을 추출**하여 낮은 차원에 표현

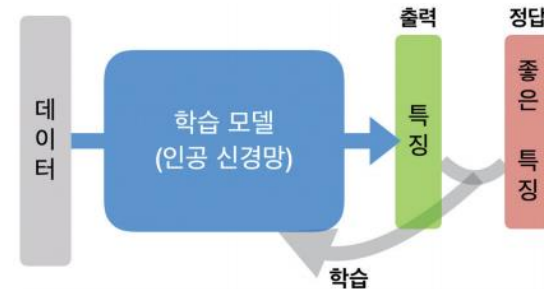
❖ 특징 추출

- 차원 축소와 원래 데이터를 다른 공간으로 옮기는 것
- 데이터를 다른 방식으로 표현(예, SVD: 데이터의 압축과 복원)
 - 데이터를 압축해서 얻은 정보는 원래 이미지에서 발견할 수 있는 특성이 전혀 없어 보임
 - 복원 연산을 통해 정보가 다 사라진 것처럼 보였던 압축 데이터에서 원래 이미지의 모습을 다시 생성
 - 압축 데이터는 우리가 눈으로 관찰하는 것과는 전혀 다른 방식으로 데이터를 표현하는 것으로, 원래의 데이터가 가진 특징을 잃어버리지 않아 복원 과정에서 원래의 데이터를 더 잘 재현
- LLE, Isomap, t-SNE

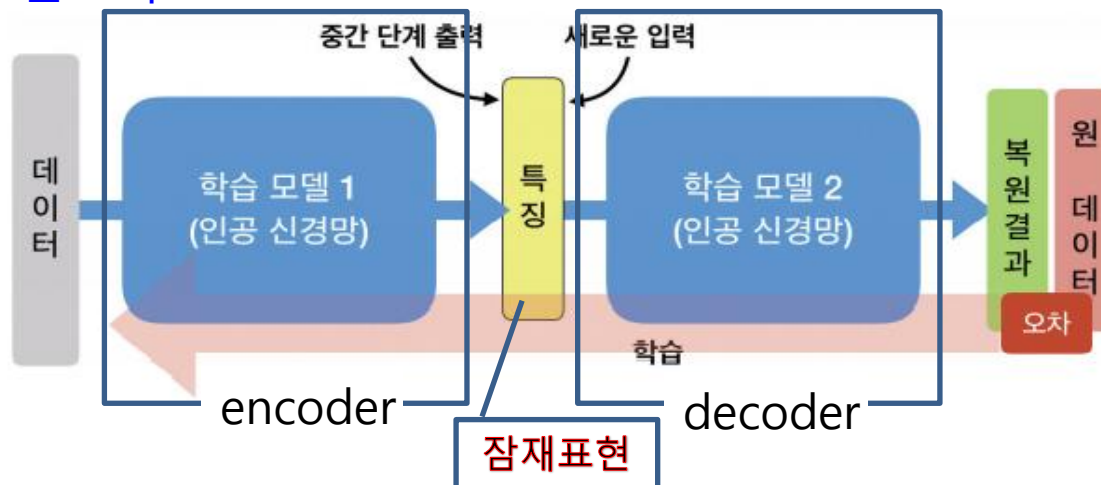
특징, 그리고 잠재 표현 학습 (2)

❖ 인공지능망을 이용한 특징 추출

- 정답으로 **좋은 특징(?)**을 제공
- 출력과 비교하여 학습



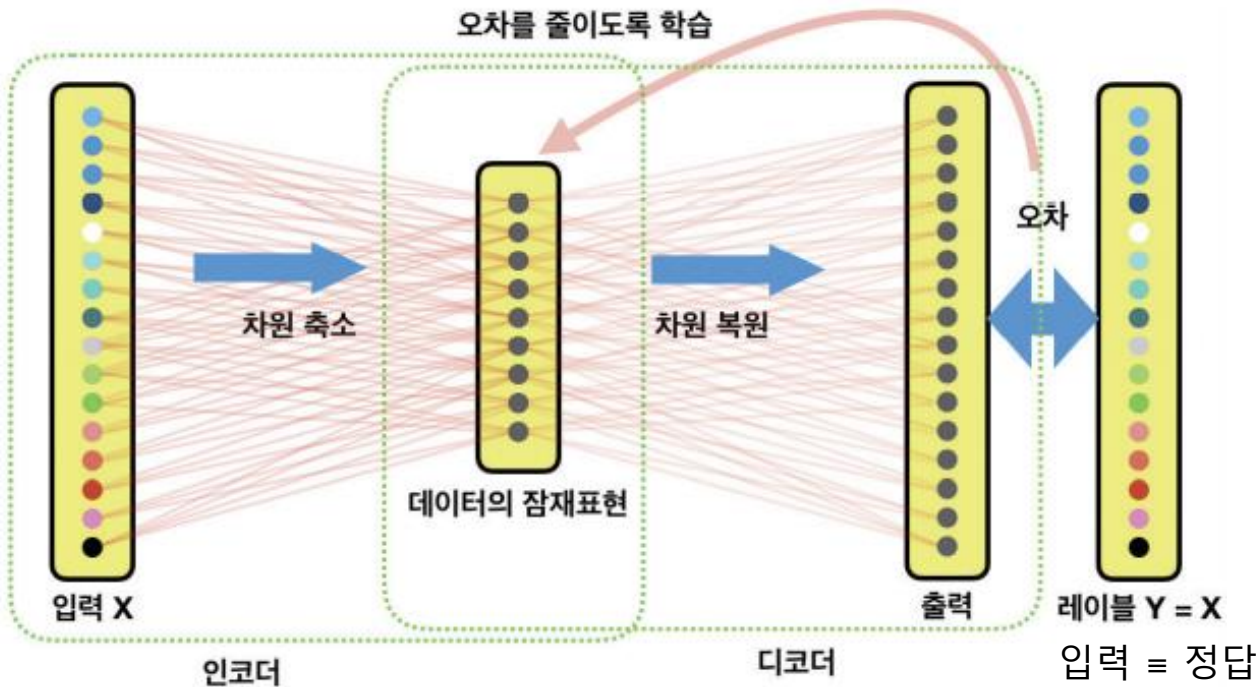
❖ 오토인코더 (autoencoder)



- 입력 데이터를 우리가 관심을 가지는 데이터 공간의 일부만을 제한적으로 드러내는 표본이라고 할 때, 이 데이터를 포함하는 미지의 큰 데이터 집단은 어떤 특징을 가지는지 파악하기 위해 사용할 수도 있을 것
- **잠재 표현** (latent representation) : 데이터를 잠재공간(중간 단계의 특징표현 공간)에 표현
- **오토인코더**: 입력 데이터의 **분포를 학습**하여 얻은 잠재 표현을 이용하여 다양한 분야, 특히 데이터를 생성하는 모델

압축과 복원 - 오토인코더로 구현하기 (1)

❖ Autoencoder의 신경망 구현



- 잠재 표현은 입력 데이터를 잘 표현하는 특징
- 사람의 개입 없이 특징을 추출하는 역할
- 차원 축소 기법을 통해 얻는 것이 아니라, 신경망이 자동으로 생성한 모델을 통해 얻음

압축과 복원 - 오토인코더로 구현하기 (2)

❖ Keras API

```
encoding_layers = Sequential(...  
decoding_layers = Sequential(...
```

← decoding_layers는 encoding_layers의
출력을 입력으로 사용

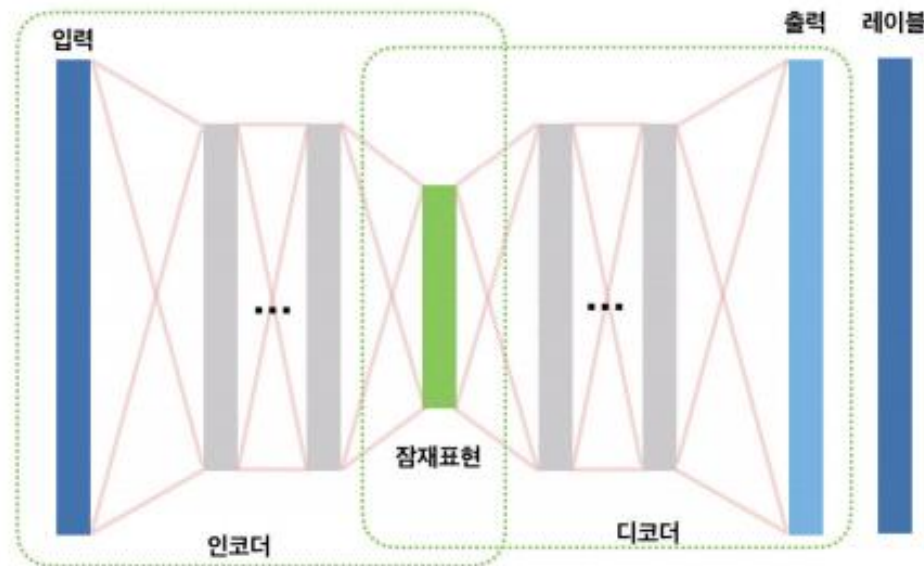
```
AE = Sequential([encoding_layers, decoding_layers])
```

```
AE.fit(X, X, epoch = n_epoch)
```

← 학습: 입력과 출력이 같음

```
embedding = encoding_layers.predict(X)
```

← 잠재표현



Program (1)

❖ LAB¹²⁻¹ 오토인코더로 롤 모양의 데이터응 2차원 공간에 임베딩

- <https://colab.research.google.com/drive/1rCWIRCOELejbUEKIX0YLp8bDWDoBMn8b>
- 비교: 11장 차원 축소 결과

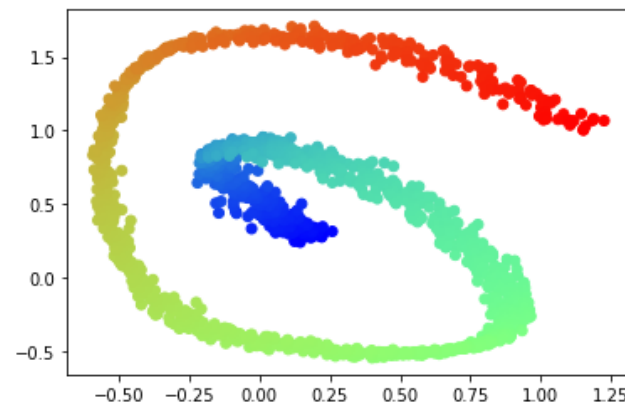
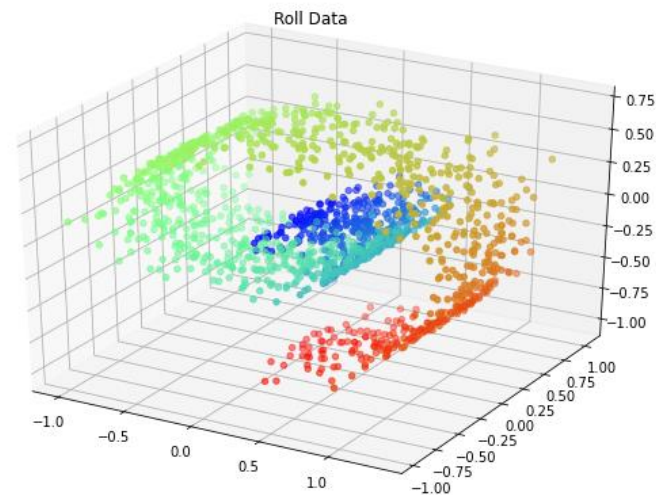
```
from tensorflow.keras import models
from tensorflow.keras import layers
enc = models.Sequential([layers.Dense(2, input_shape=[3],
                                     activation = 'elu')])
dec = models.Sequential([layers.Dense(3, input_shape=[2],
                                     activation = 'elu')])
```

```
AE = models.Sequential([enc, dec])
AE.compile(loss = 'mse')
AE.summary()
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
sequential (Sequential)	(None, 2)	8
sequential_1 (Sequential)	(None, 3)	9

Total params: 17
Trainable params: 17
Non-trainable params: 0



Program (2)

❖ LAB¹²⁻² 다층 구조 오토인코더로 차원 축소/복원

```
from tensorflow.keras import models
from tensorflow.keras import layers

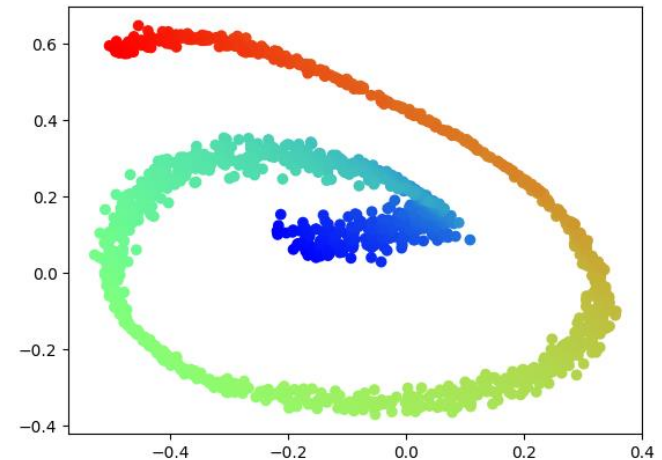
enc = models.Sequential([layers.Dense(2, input_shape=[3],
                                     activation='elu'),
                        layers.Dense(2, activation='elu'),
                        layers.Dense(2, activation='elu')])
dec = models.Sequential([layers.Dense(2, input_shape=[2],
                                     activation='elu'),
                        layers.Dense(2, activation='elu'),
                        layers.Dense(3, activation='elu')])
```

```
AE = models.Sequential([enc, dec])
AE.compile(loss = 'mse')
AE.summary()
```

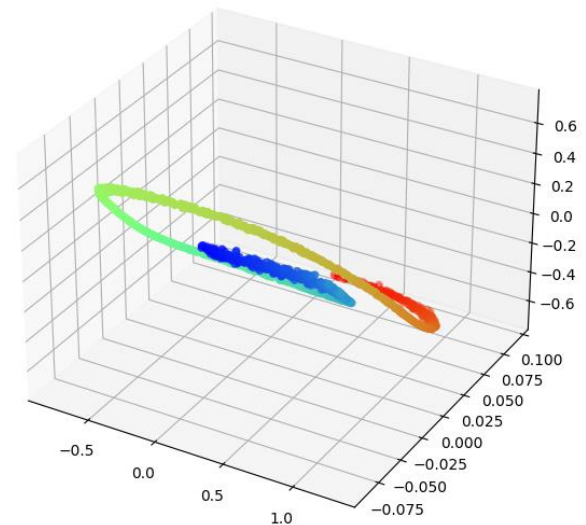
Model: "sequential_4"

Layer (type)	Output Shape	Param #
sequential_2 (Sequential)	(None, 2)	20
sequential_3 (Sequential)	(None, 3)	21

```
=====
Total params: 41
Trainable params: 41
Non-trainable params: 0
=====
```



reduced = enc.predict(X)



rcvrd = dec.predict(reduced)

Program (3)

❖ LAB¹²⁻³ 오토인코더를 이용한 이미지 압축과 복원 (비교 11장 LAB¹¹⁻²)

- MLP

```
from tensorflow.keras import models
from tensorflow.keras import layers

enc = models.Sequential([layers.Dense(64, input_shape=(784, ),
                                   activation='elu'),
                        layers.Dense(64, activation='elu'),
                        layers.Dense(64, activation='elu')])
dec = models.Sequential([layers.Dense(64, input_shape=(64, ),
                                   activation='elu'),
                        layers.Dense(64, activation='elu'),
                        layers.Dense(784, activation='elu')])
```

```
AE = models.Sequential([enc, dec])
AE.compile(loss = 'mse')
AE.summary()
```

Model: "sequential_13"

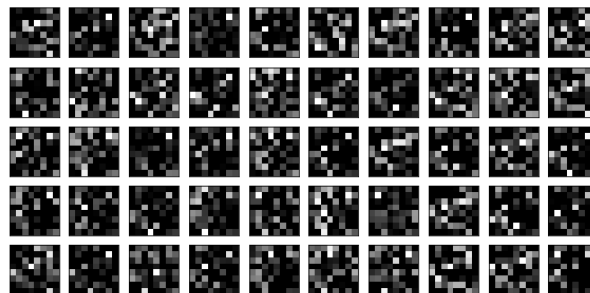
Layer (type)	Output Shape	Param #
sequential_11 (Sequential)	(None, 64)	58560
sequential_12 (Sequential)	(None, 784)	59280

=====

Total params: 117,840
Trainable params: 117,840
Non-trainable params: 0



28x28



잠재표현
8x8



Program (4)

❖ LAB¹²⁻³ 오토인코더를 이용한 이미지 압축과 복원 (비교 11장 LAB¹¹⁻²)

- Con2D 사용

```
enc_cnn = models.Sequential([
    layers.Conv2D(filters=16, kernel_size=3,
                  activation='elu', input_shape=(28,28,1)),
    layers.Conv2D(filters=16, kernel_size=3, activation='elu'),
    layers.Flatten(),
    layers.Dense(64, activation='elu')])

dec_cnn = models.Sequential([
    layers.Dense(9216, input_shape=(64, ), activation='elu'),
    layers.Reshape(target_shape=(24,24,16)),
    layers.Conv2DTranspose(filters=16, kernel_size=3, activation='elu'),
    layers.Conv2DTranspose(filters=1, kernel_size=3, activation='elu')])
```



원본

```
samples = samples.reshape(-1, 28, 28, 1)
reduced = enc_cnn.predict(samples)
recovered = dec_cnn.predict(reduced)
plot_images(5, 10, recovered.reshape(-1, 28, 28))
```

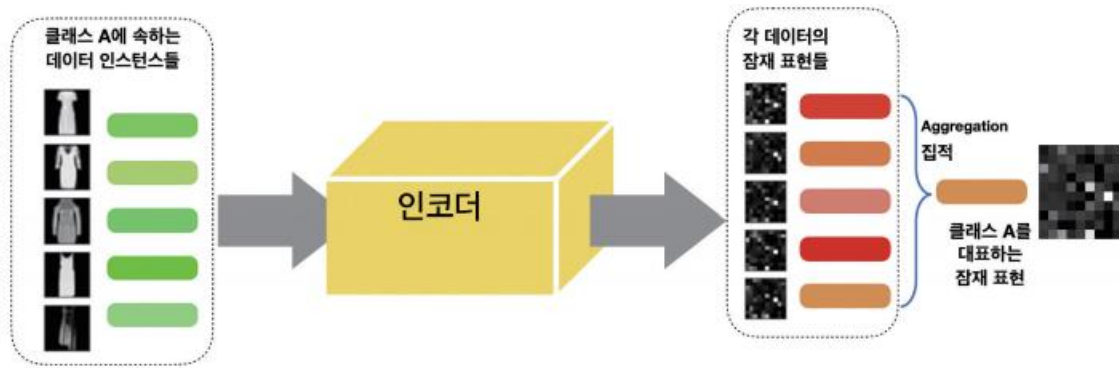
복원



잠재 표현으로 새로운 데이터 만들기 (1)

❖ 클래스 대표 데이터 생성기

- 학습이 끝난 뒤에, 같은 클래스에 속하는 데이터들만 인코더를 통과시키면 해당 클래스에 속한 데이터들의 잠재 표현을 얻을 수 있음
- **집적화**aggregation 연산: 평균값을 구하는 연산



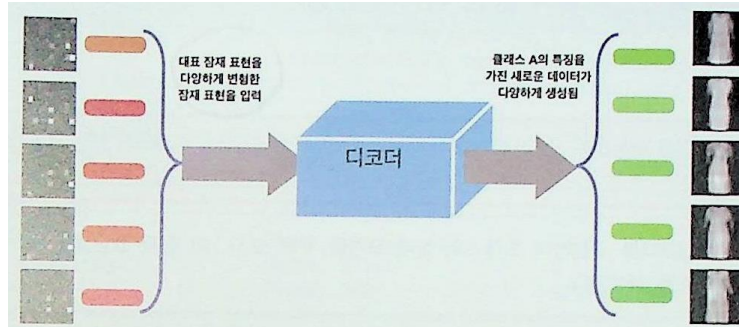
- 대표 잠재 표현을 입력받은 디코더는 해당 클래스의 특징을 가장 잘 드러내는 새로운 이상적인 데이터를 만들어 냄



잠재 표현으로 새로운 데이터 만들기 (2)

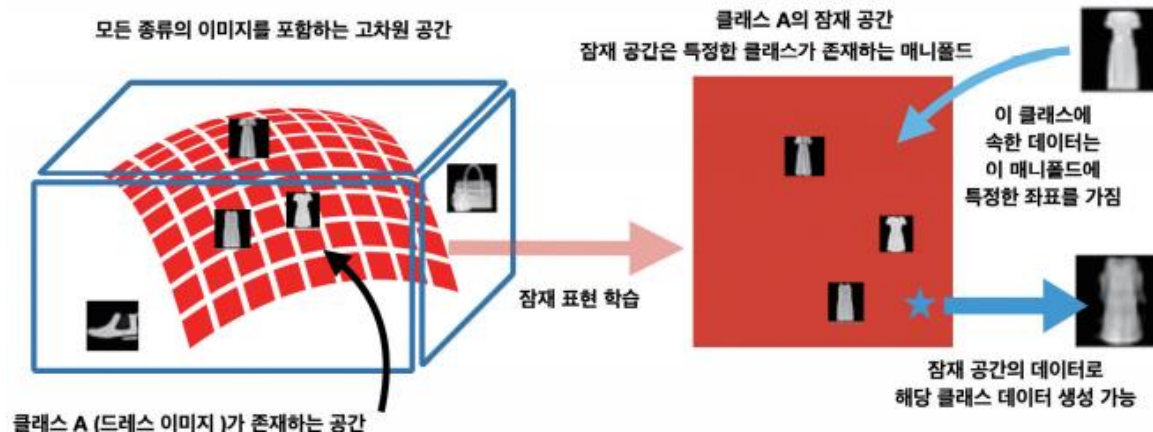
❖ 클래스에 속하는 다양한 데이터 생성기

- 대표 잠재 표현을 조금씩 변형하여 디코더에 입력
- 클래스의 특징은 가지고 있지만 원래의 데이터에서 관찰할 수 없었던 새로운 데이터를 얻을 수 있음



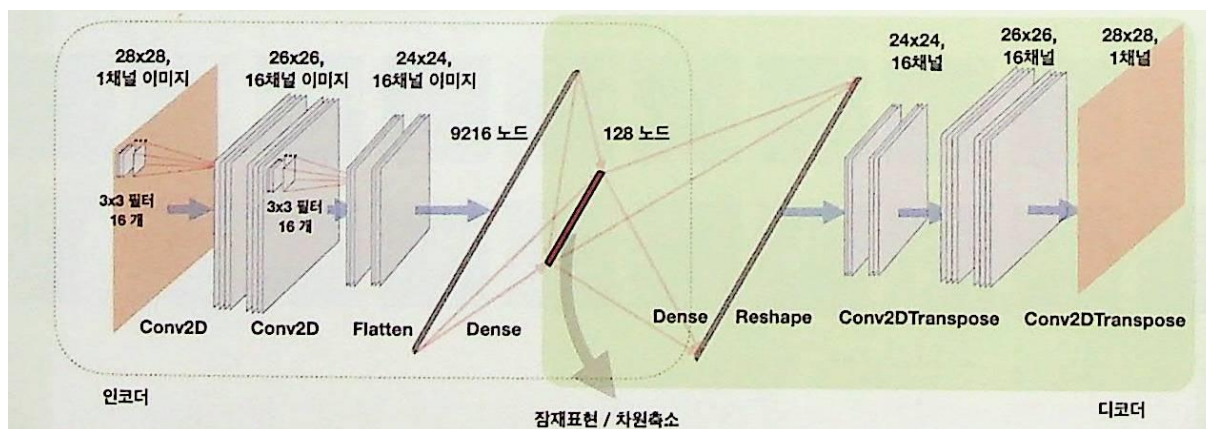
❖ 잠재공간의 학습

- 관찰한 데이터의 본질적 모습을 찾는 작업



Program (5)

❖ LAB¹²⁻⁴ 오토인코더를 이용한 데이터 생성



```
enc_cnn = models.Sequential([
    layers.Conv2D(filters=16, kernel_size=3,
                  activation='elu', input_shape=(28,28,1)),
    layers.Conv2D(filters=16, kernel_size=3, activation='elu'),
    layers.Flatten(),
    layers.Dense(100, activation='elu')])
enc_cnn.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 16)	160
conv2d_1 (Conv2D)	(None, 24, 24, 16)	2320
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 100)	921700

```
dec_cnn = models.Sequential([
    layers.Dense(9216, input_shape=(100, ), activation='elu'),
    layers.Reshape(target_shape=(24,24,16)),
    layers.Conv2DTranspose(filters=16, kernel_size=3, activation='elu'),
    layers.Conv2DTranspose(filters=1, kernel_size=3, activation='elu')])
dec_cnn.summary()
```

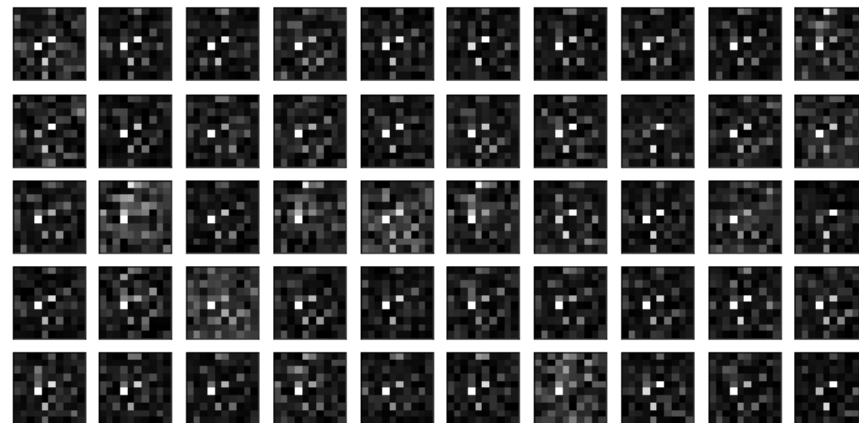
Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 9216)	930816
reshape (Reshape)	(None, 24, 24, 16)	0
conv2d_transpose (Conv2DTran	(None, 26, 26, 16)	2320
conv2d_transpose_1 (Conv2DTr	(None, 28, 28, 1)	145

Program (5)

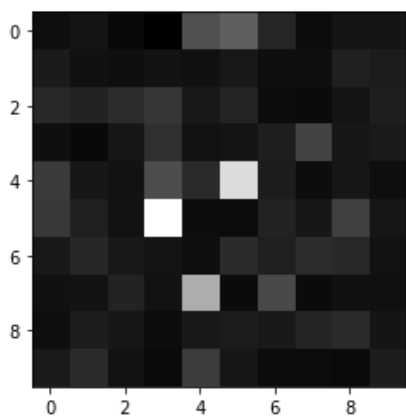


Train_labels = 3



Train_labels 3의 잠재표현

Train_labels 3의
대표 잠재표현



dress_latent =
dress_encoded.mean(axis = 0)



dress_generated = dec_cnn.predict([dress_latent_augmented])

Mini Project (잡음제거)

❖ A2 잡음제거: 오토인코더 활용(비교 A1 k-NN의 활용)

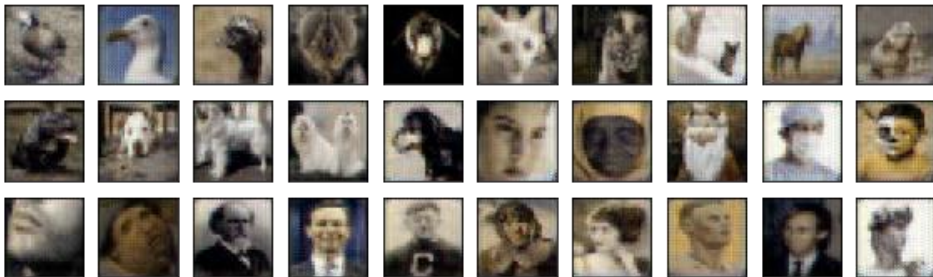
- <https://colab.research.google.com/drive/1cifkeX53mLnYdyv2yeLj02SmtR3GtMkc>
- 잠재표현: 잡음과 같이 중요하지 않는 데이터 제거한 것



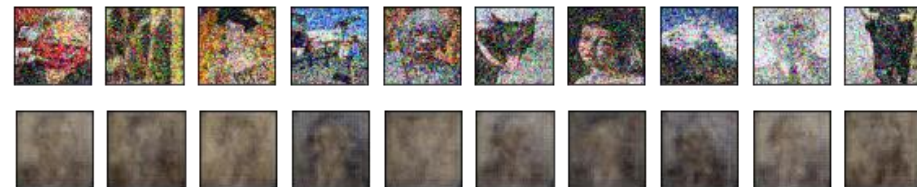
원본에 noise 추가

```
enc_cnn = models.Sequential([
    layers.Conv2D(filters=16, kernel_size=3,
                  strides=(3,3), activation='elu', input_shape=(63,63,3)),
    layers.Conv2D(filters=8, kernel_size=3, activation='elu'),
    layers.Flatten(),
    layers.Dense(1024, activation='elu')])

dec_cnn = models.Sequential([
    layers.Dense(2888, input_shape=(1024, ), activation='elu'),
    layers.Reshape(target_shape=(19,19,8)),
    layers.Conv2DTranspose(filters=16, kernel_size=3, activation='elu'),
    layers.Conv2DTranspose(filters=3, kernel_size=3, strides=(3,3),
                           activation='elu')])
```



Train data의 잠재표현



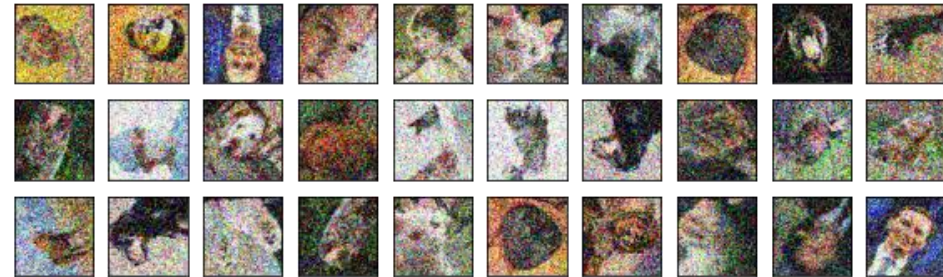
Test data의 잠재표현
Data 부족

Mini Project (잡음제거)

❖ Data 증강

- `keras.preprocessing.image.ImageDataGenerator` class
- 새로운 image 생성: `ImageDataGenerator.flow()`
- 새로운 image에 잡음 추가: `new_y + np.random.randn(nData, imgR, imgC, channel)*0.3`

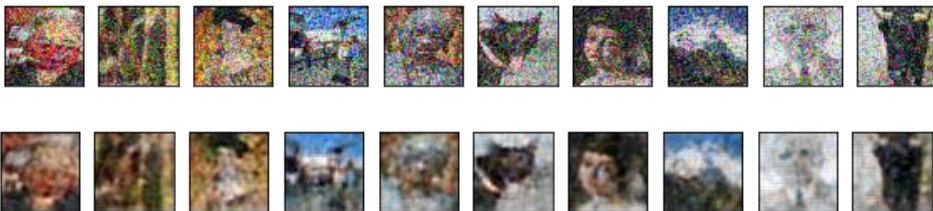
```
from keras.preprocessing.image import ImageDataGenerator
image_generator = ImageDataGenerator(
    rotation_range=360,
    zoom_range=0.1,
    shear_range=0.1,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=True)
```



변형된 image



변형된 image에 noise 추가



Test data의 잠재표현

Mini Project (데이터 분포 visualization)

❖ C1 차원축소: 데이터 분포 가시화

- Autoencoder를 사용한 잠재표현(128차원)
- T-SNE

