DL Seminar

AutoEncoder

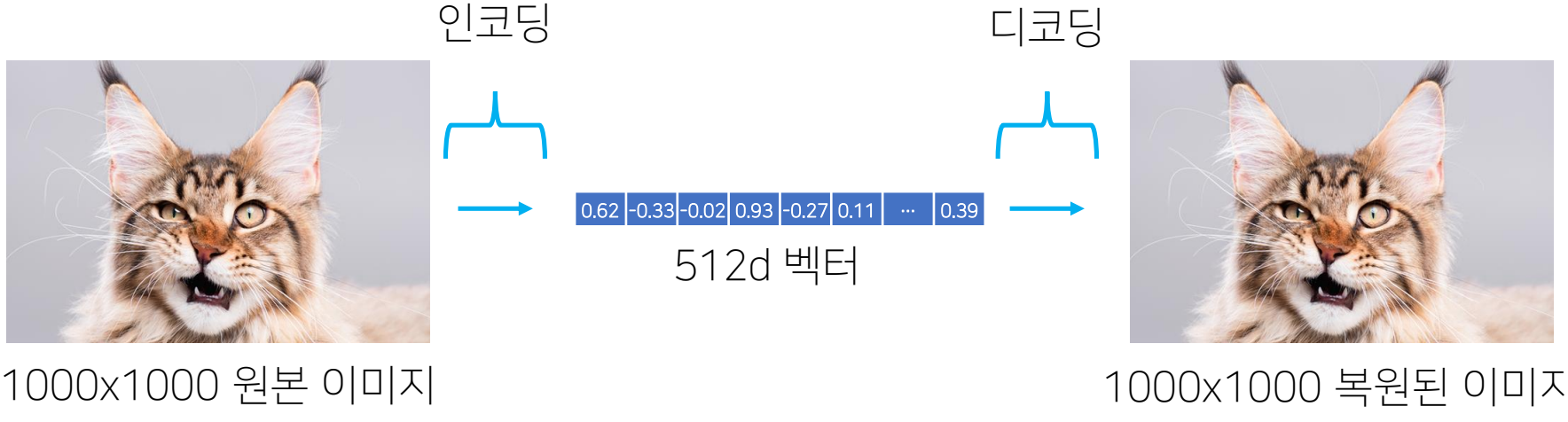
A way for Unsupervised Learning of Nonlinear Manifold

한양대학교 HANYANG UNIVERSITY

인공지능 연구실
김지성

Introduction

AutoEncoder



Introduction

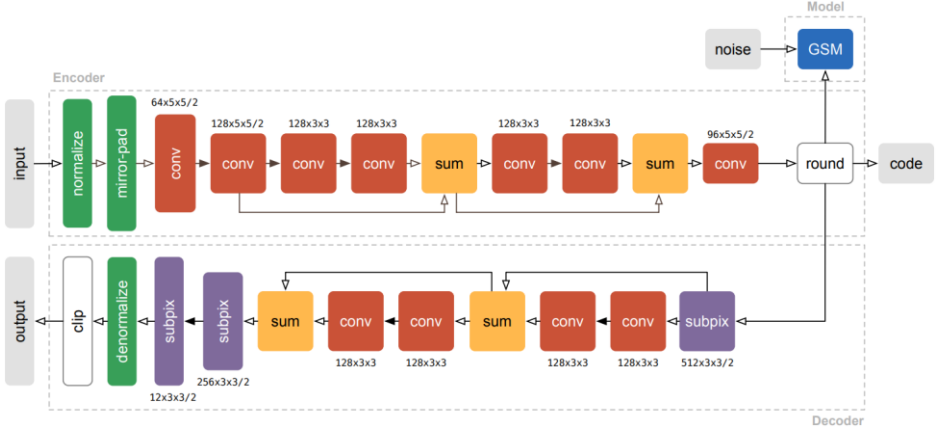
Manifold Hypothesis

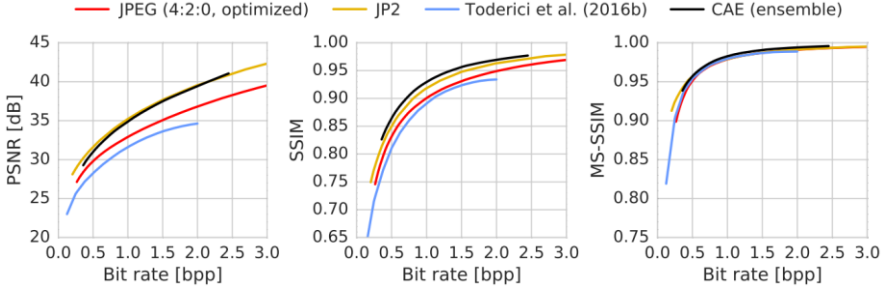
- Data Compression
  데이터 압축
- Data Visualization
  데이터 가시화
- Curse of dimensionality
  차원의 저주 해결
- Discovering most important features
  가장 중요한 피쳐 찾기

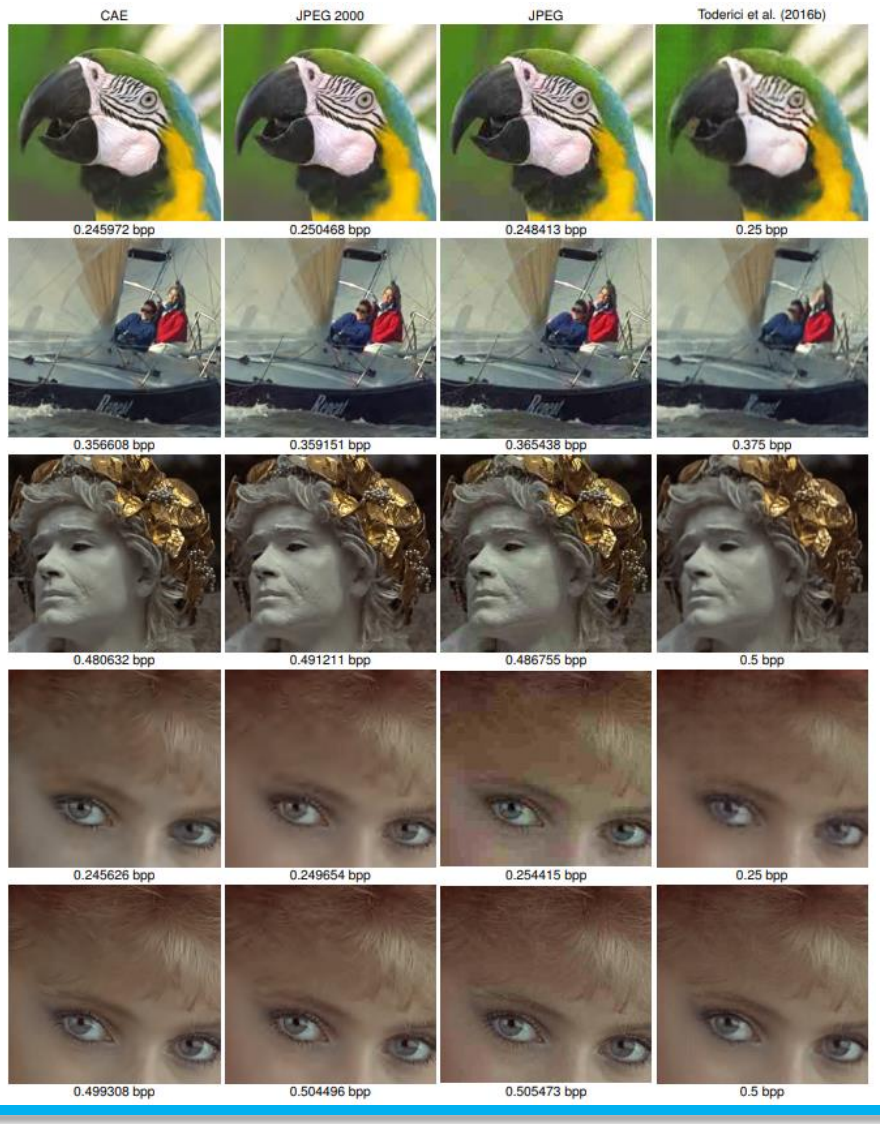
Introduction

Data Compression

(ICLR2017)Lossy Image Compression with compressive Autoencoders



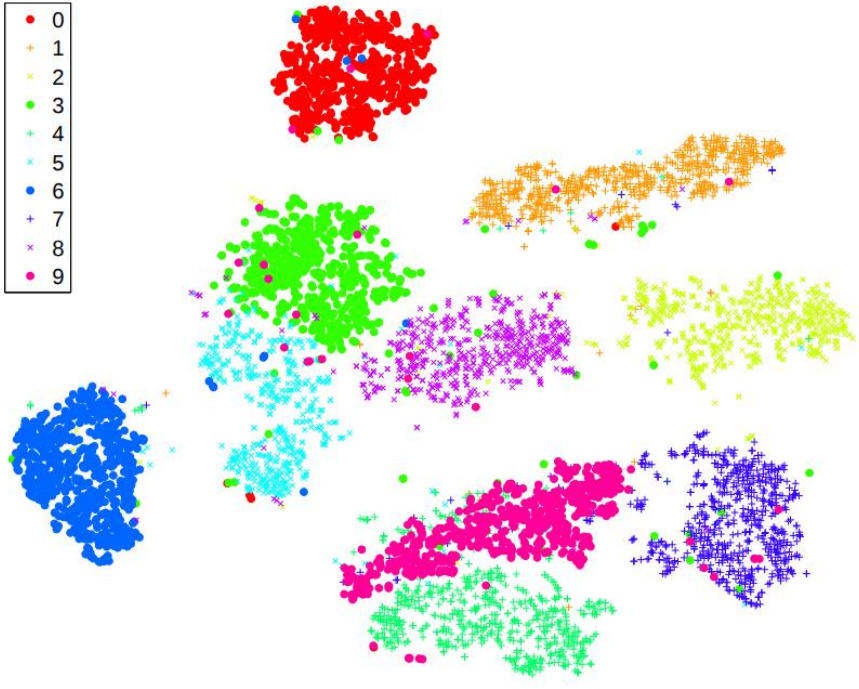




Introduction

Data Visualization

t-distributed stochastic neighbor embedding(t-SNE)



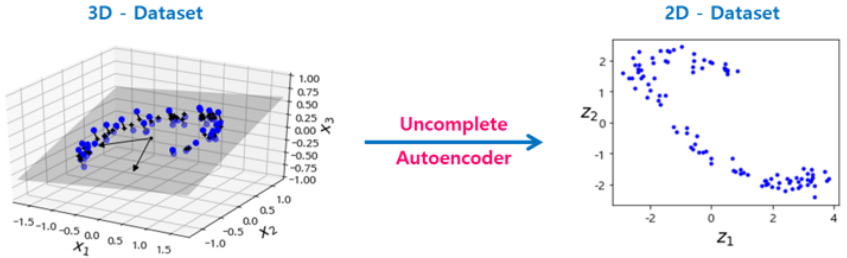
(a) Visualization by t-SNE.

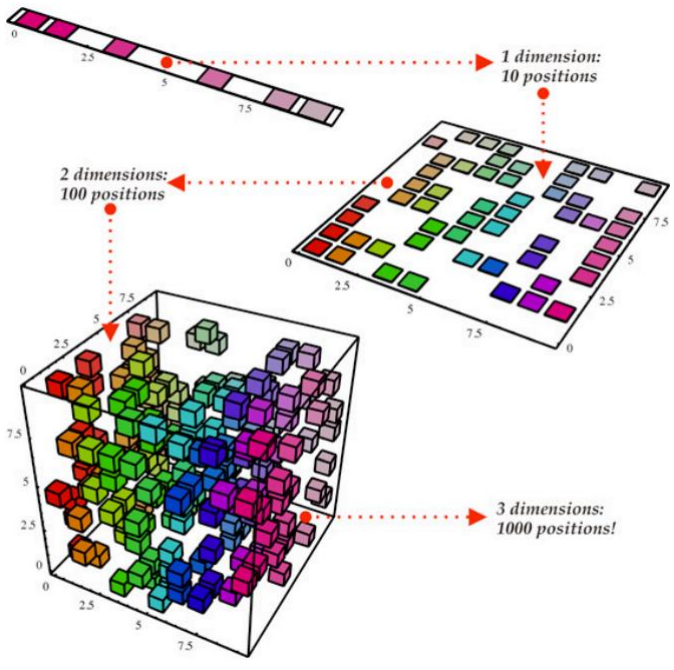
Introduction

Curse of dimensionality

데이터의 차원이 증가할수록 해당 공간의 크기가 기하급수적으로 증가하기 때문에 동일한 개수의 데이터의 밀도는 차원이 증가할수록 급속도로 희박해 진다.

따라서, 차원이 증가할 수록 데이터의 분포 분석 또는 모델추정에 필요한 샘플 데이터의 개수가 기하급수적으로 증가하게 된다.





Introduction

Manifold Hypothesis

200x200 RGB Image : $\mathbf{10^{96329}}$의 경우의 수를 가지는 40000dimension Data

나눔고딕 가장 높은 곳에 올라가려면, 가장 낮은 곳 부터 시작하라.

나눔명조 가장 높은 곳에 올라가려면, 가장 낮은 곳 부터 시작하라.

나눔 손글씨 붓체 가장 높은 곳에 올라가려면, 가장 낮은 곳 부터 시작하라.

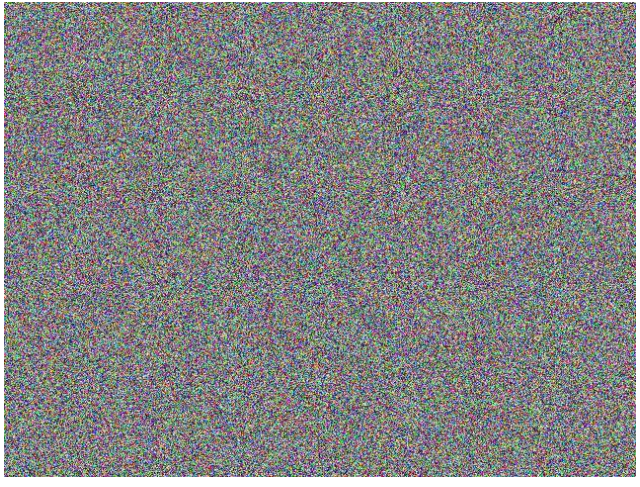
나눔 손글씨 펜체 가장 높은 곳에 올라가려면, 가장 낮은 곳 부터 시작하라.

나눔 바른고딕 가장 높은 곳에 올라가려면, 가장 낮은 곳 부터 시작하라.

나눔 바른펜 가장 높은 곳에 올라가려면, 가장 낮은 곳 부터 시작하라.

나눔스퀘어 가장 높은 곳에 올라가려면, 가장 낮은 곳 부터 시작하라.

나눔스퀘어라운드 가장 높은 곳에 올라가려면, 가장 낮은 곳 부터 시작하라.



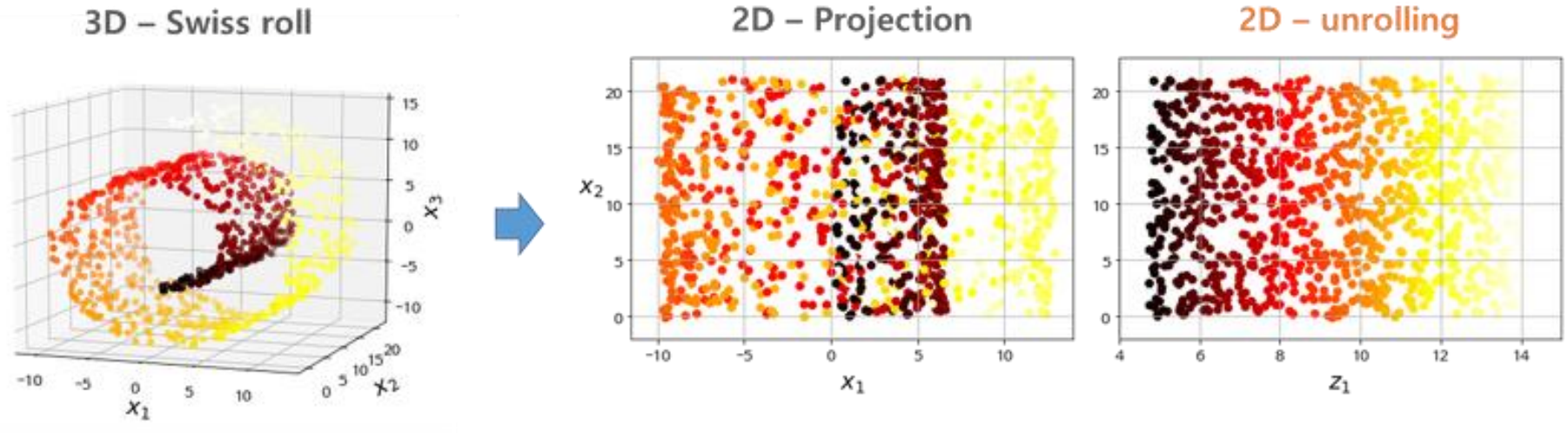
200x200 이미지로 표현 가능한 것들

Introduction

Manifold Hypothesis

가상의 Swiss Roll Dataset을 분석하려면?

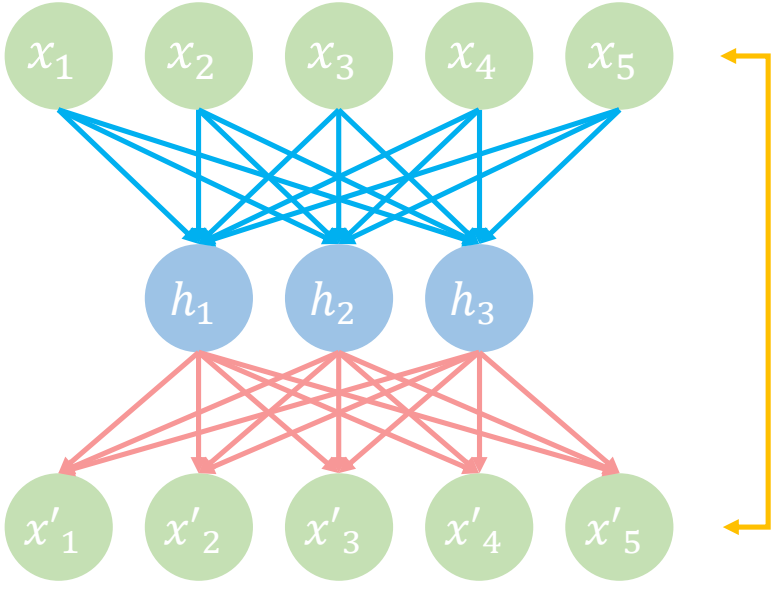
Manifold Hypothesis :
고차원인 실제 데이터셋이 낮은 저차원 매니폴드에 가깝게 놓여 있다고 가정한다.



목표 : 고차원의 데이터를 잘 표현할 수 있는 매니폴드를 모델링(러닝) 한다.

Architecture

AutoEncoder Model



Encoder

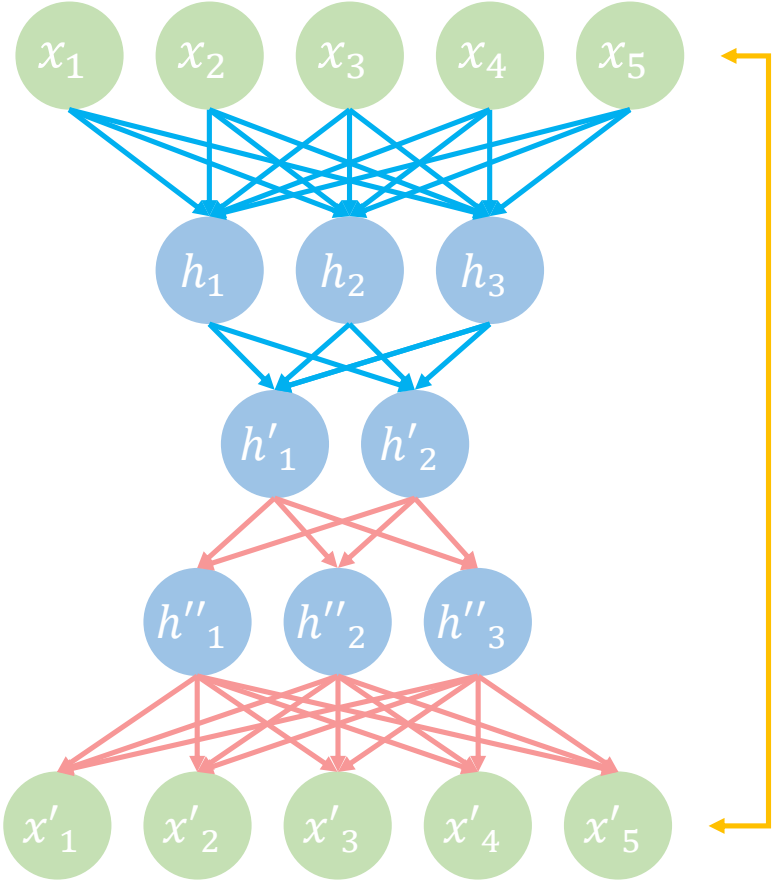
Decoder

Optimizer

X 를 인코딩해 차원을 축소한 뒤 다시
디코딩하여 원본데이터와 유사하도록 학습

Architecture

Stacked AutoEncoder



Encoder

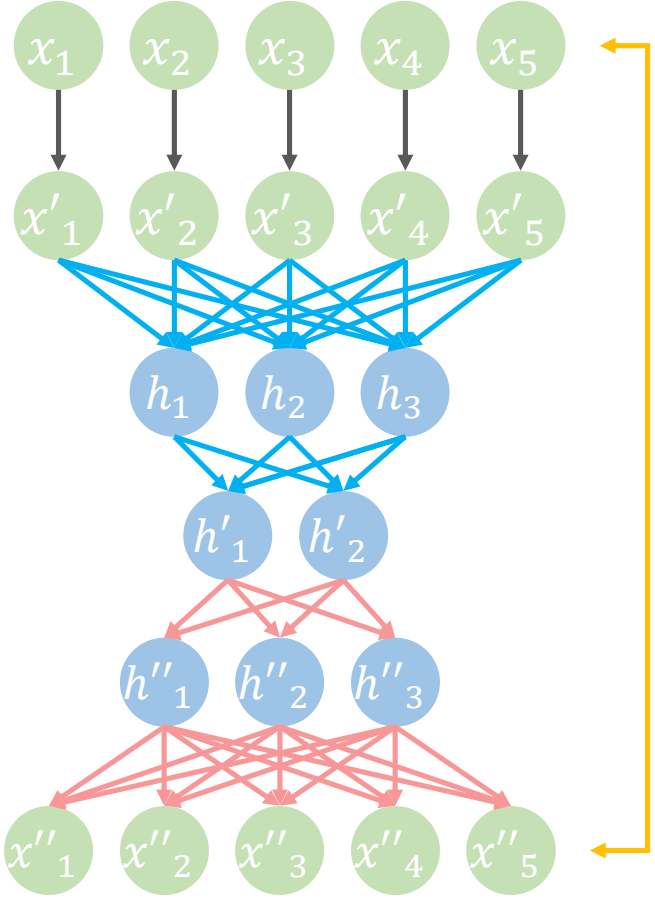
Decoder

Optimizer

Encoder, Decoder가 Deep한 구조
복잡한 구조 학습에 용이

Architecture

Denoising AutoEncoder



Random Noise Adder

Encoder

Decoder

Optimizer

Input Data에 Random Noise를 추가한 뒤
원본 데이터와 유사하도록 학습

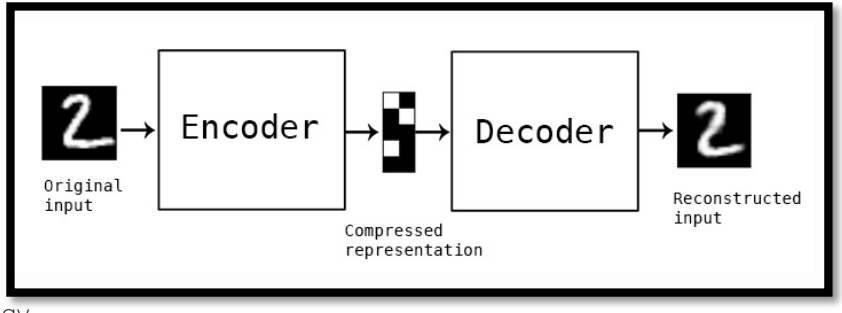
Implement

Implement DAE - Object

MNIST Set

Hand written Digit set
Size : 28 x 28
Training Image : 60000개
Test Image : 10000개

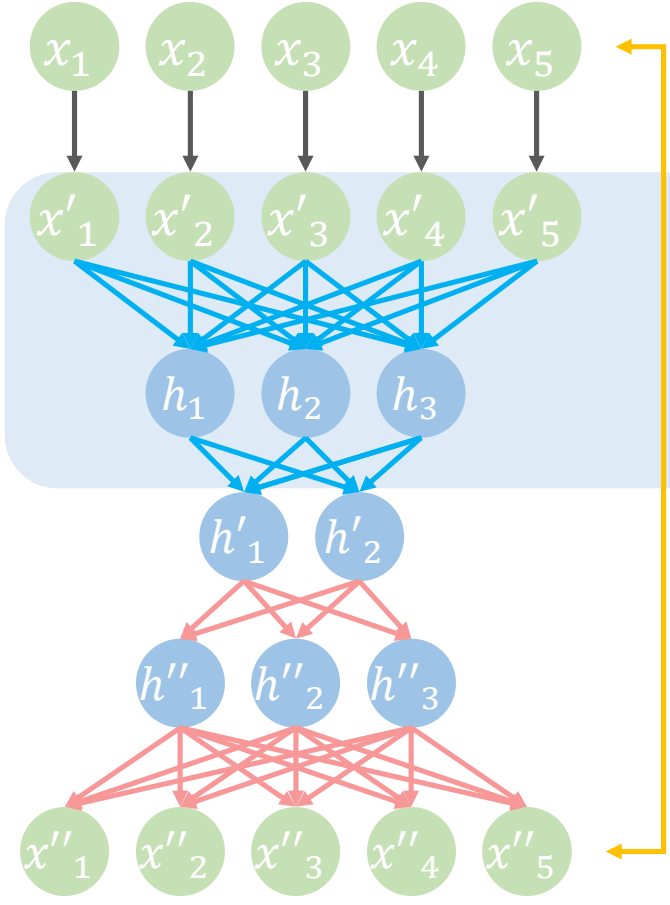
목표 :
MNIST Set에 Noise를 추가한 데이터를 Feeding,
원본 데이터를 목표로 차이를 최소화 하도록 학습



MNIST 데이터베이스 (Modified National Institute of Standards and Technology database)는 손으로 쓴 숫자들로 이루어진 대형 데이터베이스이며, 다양한 화상 처리 시스템을 트레이닝하기 위해 일반적으로 사용된다. (wikipedia)

Implement

Implement DAE

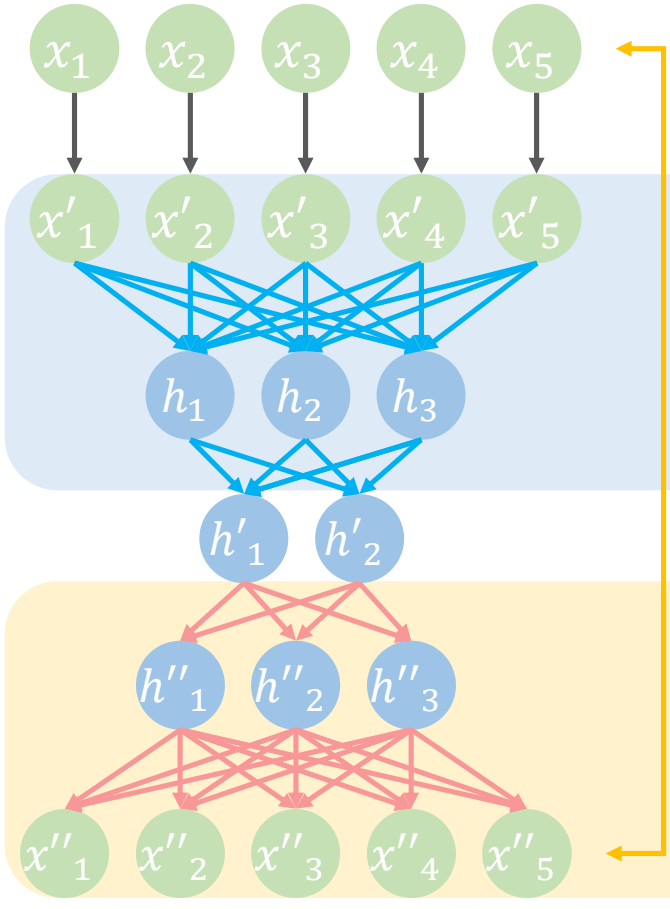


InputSize : 784(28x28)
HiddenLayer Encoder1, Decoder2 : 512
HiddenLayer Encoder2, Decoder1 : 256
OutputSIze : 784(28x28)

```
x = tf.placeholder("float", [None, 784])
y = tf.placeholder("float", [None, 784])

# 인코더
encoder_1_weight = tf.Variable(tf.random_normal([784, 512]))
encoder_1_bias = tf.Variable(tf.random_normal([512]))
encoder_1 = tf.add(tf.matmul(x, encoder_1_weight), encoder_1_bias)
encoder_1 = tf.nn.sigmoid(encoder_1)

encoder_2_weight = tf.Variable(tf.random_normal([512, 256]))
encoder_2_bias = tf.Variable(tf.random_normal([256]))
encoder_2 = tf.add(tf.matmul(encoder_1, encoder_2_weight), encoder_2_bias)
encoder_2 = tf.nn.sigmoid(encoder_2)
```

Implement

Implement DAE



InputSize : 784(28x28)
HiddenLayer Encoder1, Decoder2 : 512
HiddenLayer Encoder2, Decoder1 : 256
OutputSIze : 784(28x28)

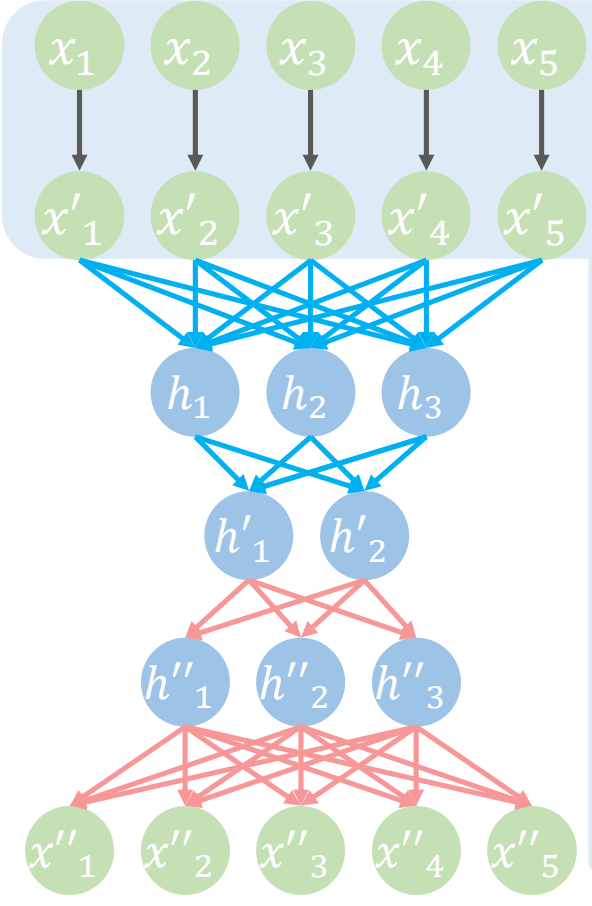
```
x = tf.placeholder("float", [None, 784])
y = tf.placeholder("float", [None, 784])

# 인코더
encoder_1_weight = tf.Variable(tf.random_normal([784, 512]))
encoder_1_bias = tf.Variable(tf.random_normal([512]))
encoder_1 = tf.add(tf.matmul(x, encoder_1_weight), encoder_1_bias)
encoder_1 = tf.nn.sigmoid(encoder_1)

encoder_2_weight = tf.Variable(tf.random_normal([512, 256]))
encoder_2_bias = tf.Variable(tf.random_normal([256]))
encoder_2 = tf.add(tf.matmul(encoder_1, encoder_2_weight), encoder_2_bias)
encoder_2 = tf.nn.sigmoid(encoder_2)
```

```
# 디코더
decoder_1_weight = tf.Variable(tf.random_normal([256, 512]))
decoder_1_bias = tf.Variable(tf.random_normal([512]))
decoder_1 = tf.add(tf.matmul(encoder_2, decoder_1_weight), decoder_1_bias)
decoder_1 = tf.nn.sigmoid(decoder_1)

decoder_2_weight = tf.Variable(tf.random_normal([512, 784]))
decoder_2_bias = tf.Variable(tf.random_normal([784]))
decoder_2 = tf.add(tf.matmul(decoder_1, decoder_2_weight), decoder_2_bias)
decoder_2 = tf.nn.sigmoid(decoder_2)
model = decoder_2
```

Implement

Implement DAE

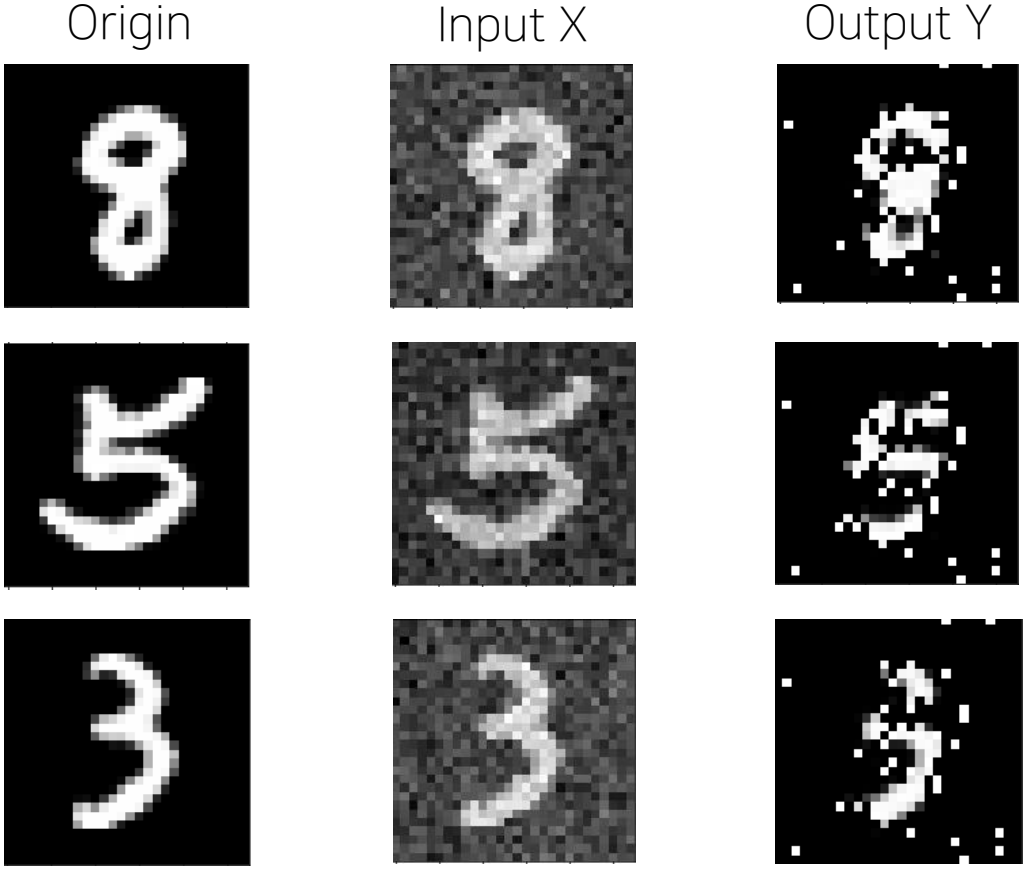
Loss Function : MeanSqureError
Optimizer : ADAM
LearningRate : 0.01
Batch Size : 300
Epoch : 2000




```
# Loss 정의
loss = tf.reduce_mean(tf.pow(model-y, 2))

# optimizer 정의
Optm = tf.train.AdamOptimizer(0.01).minimize(loss)

# tensorflow 서버 생성
sess = tf.Session()
sess.run(tf.initialize_all_variables())

# 학습
while(True)
    # Batch 그룹 생성(GRAM 크기에 따라 설정, Trainset 크기에 나누어 떨어져야 함)
    num_batch  = int(mnist.train.num_examples/300)

    # BatchSize 만큼 잘라 반복
    for i in range(num_batch):
        batch_xs, batch_ys = mnist.train.next_batch(300)

        # Random Noise 생성
        batch_xs_noisy = batch_xs + 0.3 * np.random.randn(300, 784)

        # feed forward
        feeds = {x: batch_xs_noisy, y: batch_xs}
        sess.run(optm, feed_dict=feeds)
```

Implement

Implement DAE - Inference



Conclusion

- Data Compression
  데이터 압축
- Data Visualization
  데이터 가시화
- Curse of dimensionality
  차원의 저주 해결
- Discovering most important features
  가장 중요한 피쳐 찾기


감사합니다.