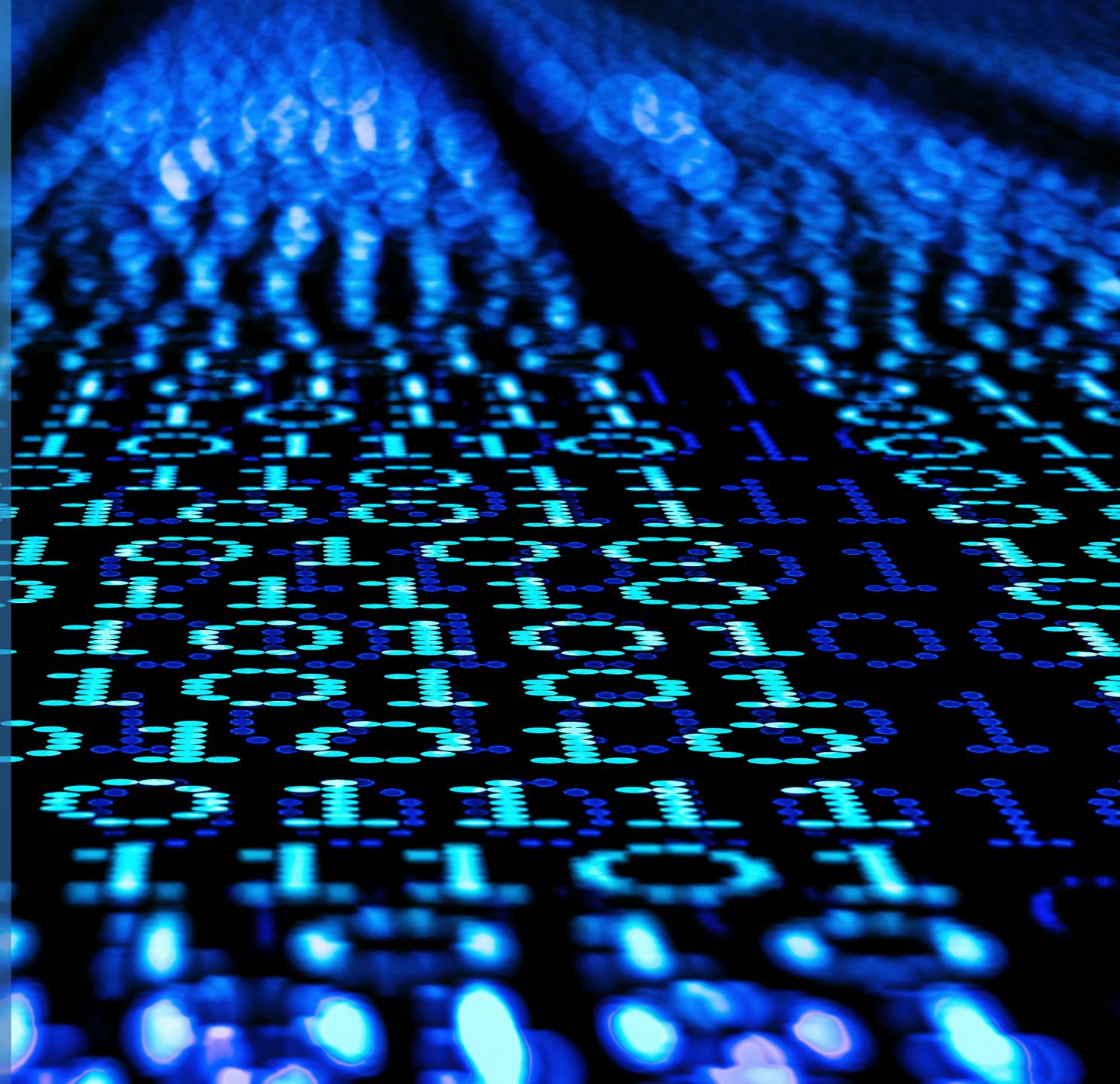


Auto-Encoder

RNN



Auto Encoder

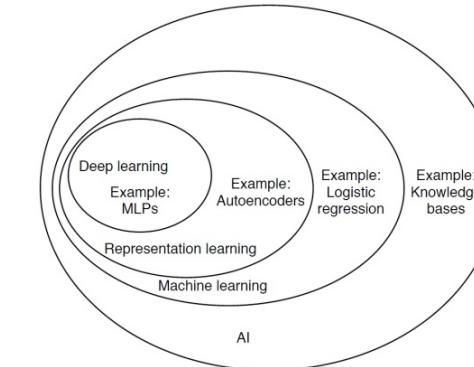
■ Auto Encoder란?

- Keywords
 - : Unsupervised learning
 - : Representation Learning
 - : Dimensionality reduction
 - : Generative model learning
- ML과 DNN의 중간 위치에 있는
Representation Learning의 한 종류

FOUR KEYWORDS | Representation learning

INTRODUCTION

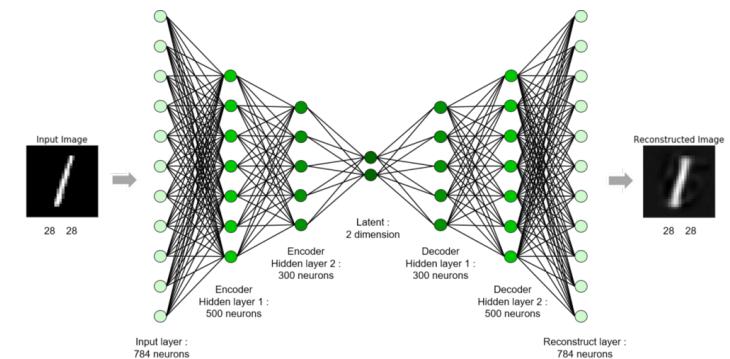
3 / 5



- [KEYWORDS]
- ❖ Unsupervised learning
 - ❖ Nonlinear Dimensionality reduction
 - = **Representation learning**
 - = Efficient coding learning
 - = Feature extraction
 - = Manifold learning
 - ❖ Generative model learning

http://videolectures.net/kdd2014_bengio_deep_learning/

- 입력데이터를 압축시켜 축소한 후 다시 확장하여 결과 데이터를 입력데이터와 동일하도록 하는 DNN
- 우측 그림의 입력데이터는 784개의 뉴런을 500, 300, 2개로 축소한 후 동일한 사이즈인 784로 출력한 경우
- Encoder : 그림의 왼쪽, 즉 압축시키는 부분
- Decoder : 그림의 오른쪽, 즉 확장시키는 부분
- **Unsupervised Learning 혹은 Self-Supervised Learning**



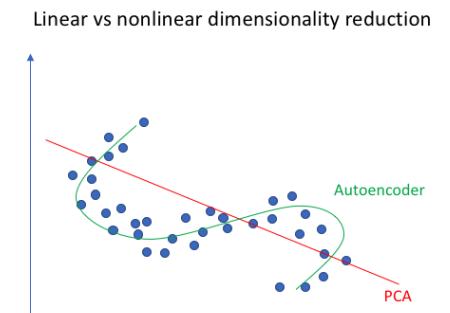
<https://mc.ai/auto-encoder-in-biology/>

Auto Encoder

■ PCA vs Encoder

- **유사성**
: 차원을 축소하여 Feature를 축출한다
- **차이점**
: PCA \rightarrow 선형적으로 데이터 차원을 축소
(예외, Kernel PCA)

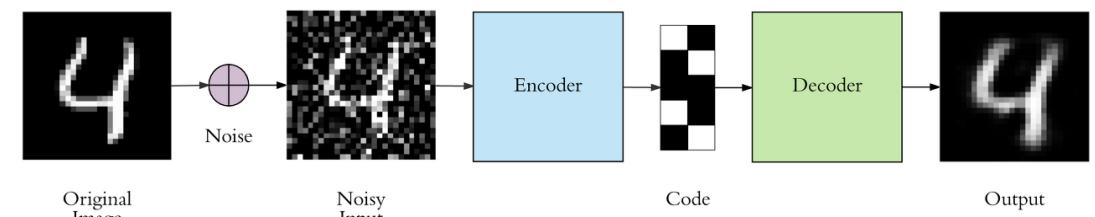
: Encoder \rightarrow 비선형적으로 데이터 차원을 축소 할 수 있다



<https://www.jeremyjordan.me/autoencoders/>

■ Denoising Auto Encoder(DAE)

- 임의의 노이즈 데이터를
의도적으로 추가 혹은 제거
- 노이즈 데이터는
Original Data에서
특정 분포를 이용하여
random sample 추출
- Dropout과 유사한 학습 방법
: dropout은 정의된 확률로 노드의 일부를 삭제, Why \rightarrow Overfit 방지, **Regularization**

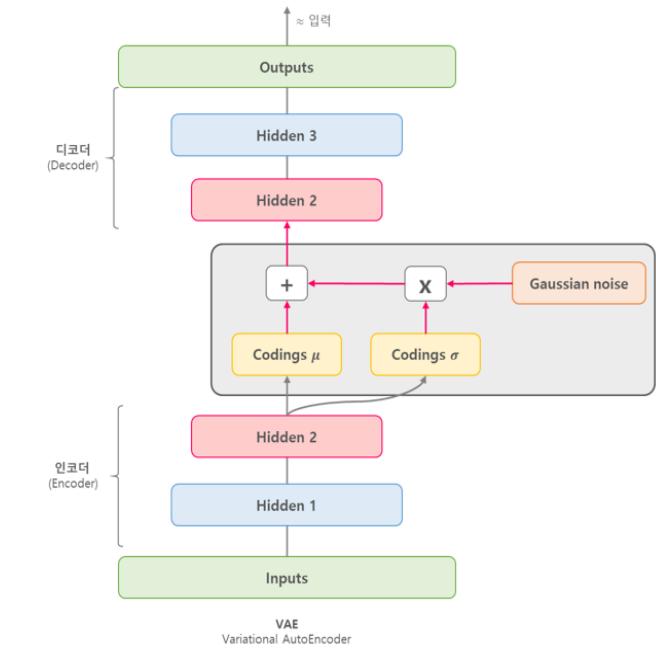


<https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798?gi=b93263c1ab82>

Auto Encoder

■ Variational Auto Encoder(VAE)

- DAE : 노이즈 제거를 통해 Feature Extraction
- VAE : 노이즈를 추가하여 Feature를 추가
- 가우시안 분포에 따르는 값을 Sampling하여 추가
이 과정을 반복적으로 수행하여 근사



<https://engineer-mole.tistory.com/30>

■ Auto Encoder 사례



RNN

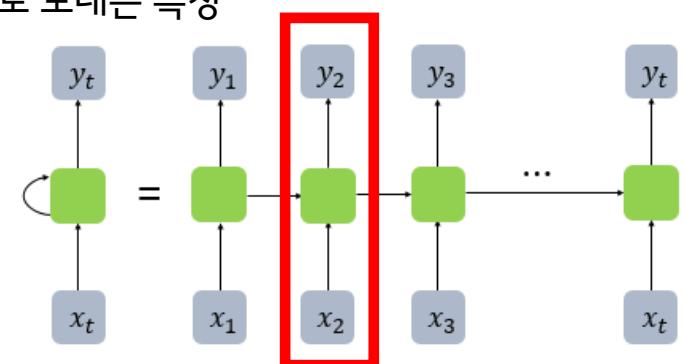
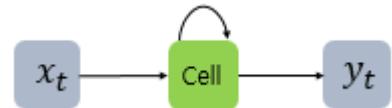
■ Sequence 모델

- 입력과 출력을 sequence 단위로 처리 하는 data를 처리하기 위한 목적의 Model
 - : Text streams, audio clips, video clips, time-series data
 - : 순서가 매우 중요, 이것이 결국 단점, 병렬처리 불가
 - : Sequence는 순서를 거치기 때문에 순서가 오래 될 수록 앞의 정보가 희미 해진다
 - … Transformer에서 self-attention으로 병렬처리
 - … Transformer에서 각 token이 모든 token과 연산을 수행함으로 direct로 관계를 구현 함
- Time Series Data : 생성 시점이 매우 중요, ex) 주식, 센서 데이터

■ 중요개념

- : Hidden State
- : Time Step

- Feed Forward Neural Network
 - : 입력층 \rightarrow 은닉층 \rightarrow 활성화 함수 \rightarrow 출력층
 - : 한 방향
- RNN의 특징
 - : 은닉층의 결과값이 출력층과 다음 은닉층의 입력으로 보내는 특징
- Hidden State
 - : 은닉층의 결과 값
- Time Step
 - : 우측 그림의 붉은 색 상자
 - : 일반적으로 $t-1, t, t+1$ 으로 표현



RNN

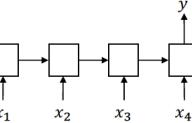
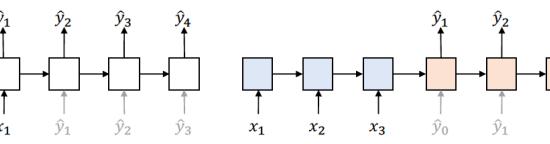
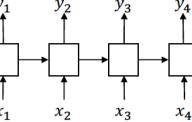
■ RNN 설계> 유연성

- RNN의 셀의 단위

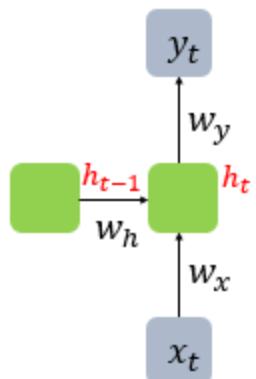
: NLP에서는 일반적으로
단어 벡터

- 다양하게 설계가 가능

: Many to One
: One to Many
: Many to Many

Type	Architecture	Applications
Many to One		Text Classification
One to Many		NLG, Machine Translation
Many to Many		POS Tagging, MRC

■ RNN 수식



- 은닉층 : $h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$
- 출력층 : $y_t = f(W_y h_t + b)$

$$x_t : (d \times 1) \quad W_x : (D_h \times d) \quad W_h : (D_h \times D_h) \quad h_{t-1} : (D_h \times 1) \quad b : (D_h \times 1)$$

$$\tanh \left(\begin{array}{c} W_h \\ \times \\ D_h \times D_h \end{array} \right. + \begin{array}{c} W_x \\ \times \\ D_h \times d \end{array} \right. \times \begin{array}{c} x_t \\ d \times 1 \end{array} + \begin{array}{c} b \\ D_h \times 1 \end{array} \left. \right) = \begin{array}{c} h_t \\ D_h \times 1 \end{array}$$

xt : t 의 입력, d : 단어 벡터의 차원, Dh : 은닉상태의 크기

RNN

■ RNN in Keras

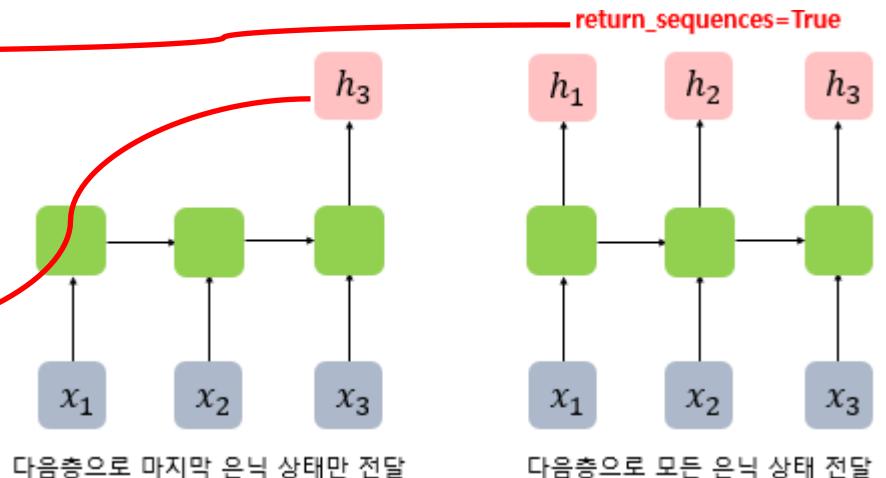
....> `return_sequences=True`

- 은닉층에서 다음 은닉층으로

Hidden State를 전달하기 위해서는
`return_sequences=True`를 사용

- 마지막 은닉 상태에만

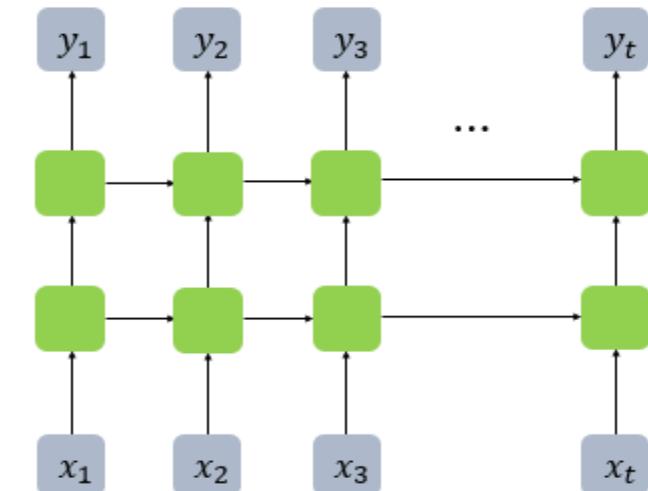
Hidden State를 전달하려면
`return_sequences=False`



■ Deep RNN

- 다수의 은닉층

```
model = Sequential()  
model.add(SimpleRNN(hidden_units,  
                     input_length=10,  
                     input_dim=5,  
                     return_sequences=True))  
model.add(SimpleRNN(hidden_units,  
                     return_sequences=True))
```



■ Bidirectional RNN

```
from tensorflow.keras.layers import Bidirectional  
  
timesteps = 10  
input_dim = 5  
  
model = Sequential()  
model.add(Bidirectional(SimpleRNN(hidden_units,  
                                return_sequences=True),  
                                input_shape=(timesteps, input_dim)))
```

