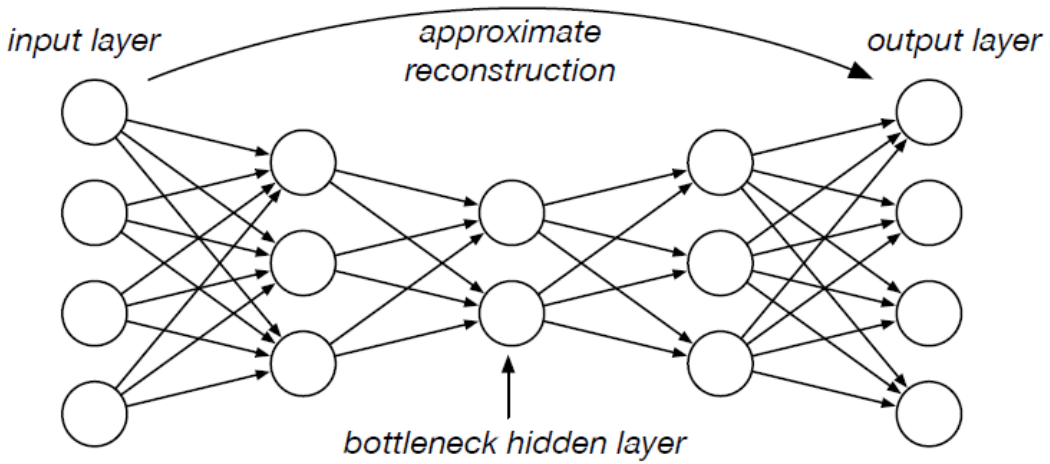
USAD :
Unsupervised Anomaly Detection on Multivariate Time

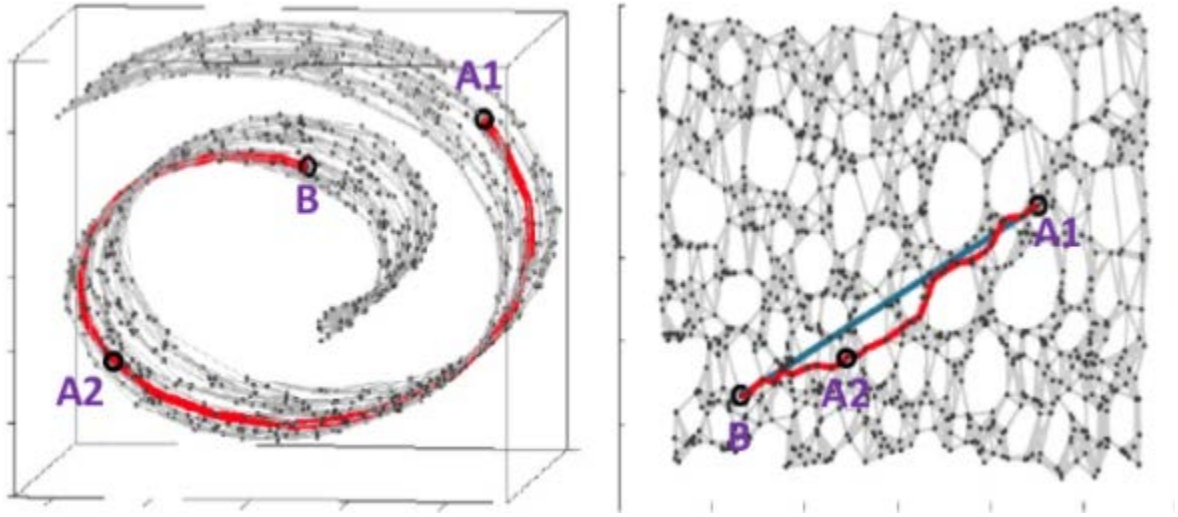
(KDD 2020)

송민수

AutoEncoder

- Unsupervised learning
- Nonlinear dimensionality reduction
- Feature extraction
- Manifold learning





https://www.slideshare.net/NaverEngineering/ss-96581209

- $min\ (x_i - \hat{x}_i)^2$
- input=output

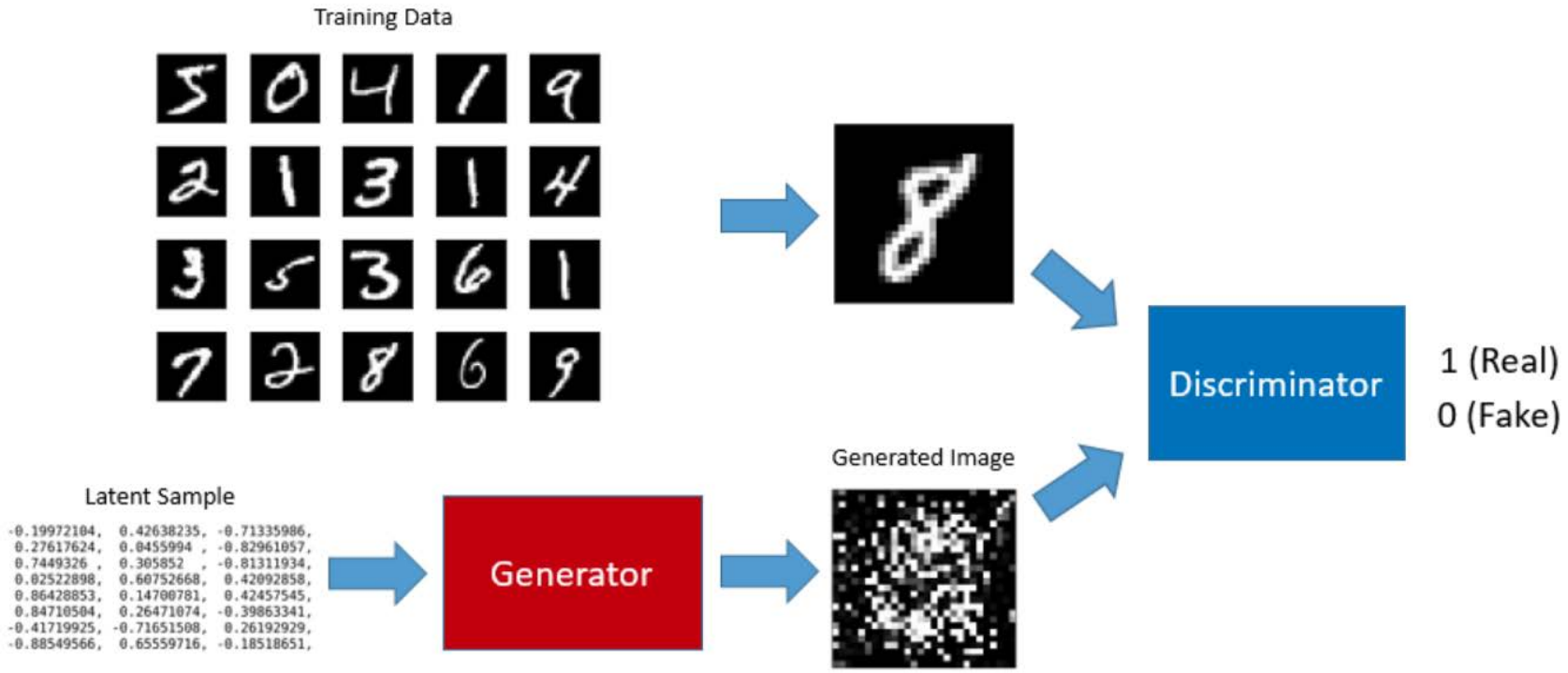
AutoEncoder in Anomaly detection

- Learns the representations of data instances by optimizing a general feature learning objective function that is not primarily designed for anomaly detection but empower the anomaly detection

- 장점 : 다양한 data에 사용가능
- 단점 : feature represetation의 불완전성, overfitting

GAN

- Discriminator
- Generator



https://naokishibuya.medium.com/understanding-generative-adversarial-networks-4dafc963f2ef

Introduction

- 본인들의 회사에서 IT system monitoring을 위해 만들었다.
- 기존의 전통적인 distance-based, clustering, One-class SVM 등의 방법 같은 경우, data가 복잡해지고 차원이 급증하면서 부족함을 느꼈다.
- 또한, RNN 계열은 computationally hungry하고 train시간이 오래걸린다.

-> 그래서 우리는

- Unsupervised
- Anomaly detection
- Multivariate time series
- Based on AE & GAN

Train data

- Train input
  - Only normal points
  - Window별로 anomaly score를 내서 판단
  - $W_t = \{\vec{x}_t, \vec{x}_{t+1}, \cdots, \vec{x}_{t+k}\}$

AutoEncoder in anomaly detection

- AE에서 anomaly score는
  - Reconstruction error
  - 즉, AE가 reconstruction을 잘 못하면 이는 이상치(anomaly)이다.

  - Train할 때, normal data만 이용하면 normal data를 넣었을 때, 복원을 잘 할 것이다. 그런데 anormaly data가 들어오면 복원을 잘 못할 것이다.

  - 하지만 여기서 한계점! Anormaly data가 normal data와 비슷하다면?!

AutoEncoder in anomaly detection

- AE가 normal과 비슷한 anomaly data를 잘 복원하는 경우 발생!
- 따라서 이를 보완하는 장치가 필요하다.

GAN in anomaly detection

- Training을 한 discriminator를 anomaly detector로 사용
  - Normal data로만 train

- Normal과 다른 data가 들어오면 discriminator는 '아 이 data는 generator가 만든 fake data구나' 라고 판단

- 하지만 GAN의 단점이 여전히 존재
  - Mode collapse
  - Non-convergenve
- 이를 보완하는 장치 필요

USAD

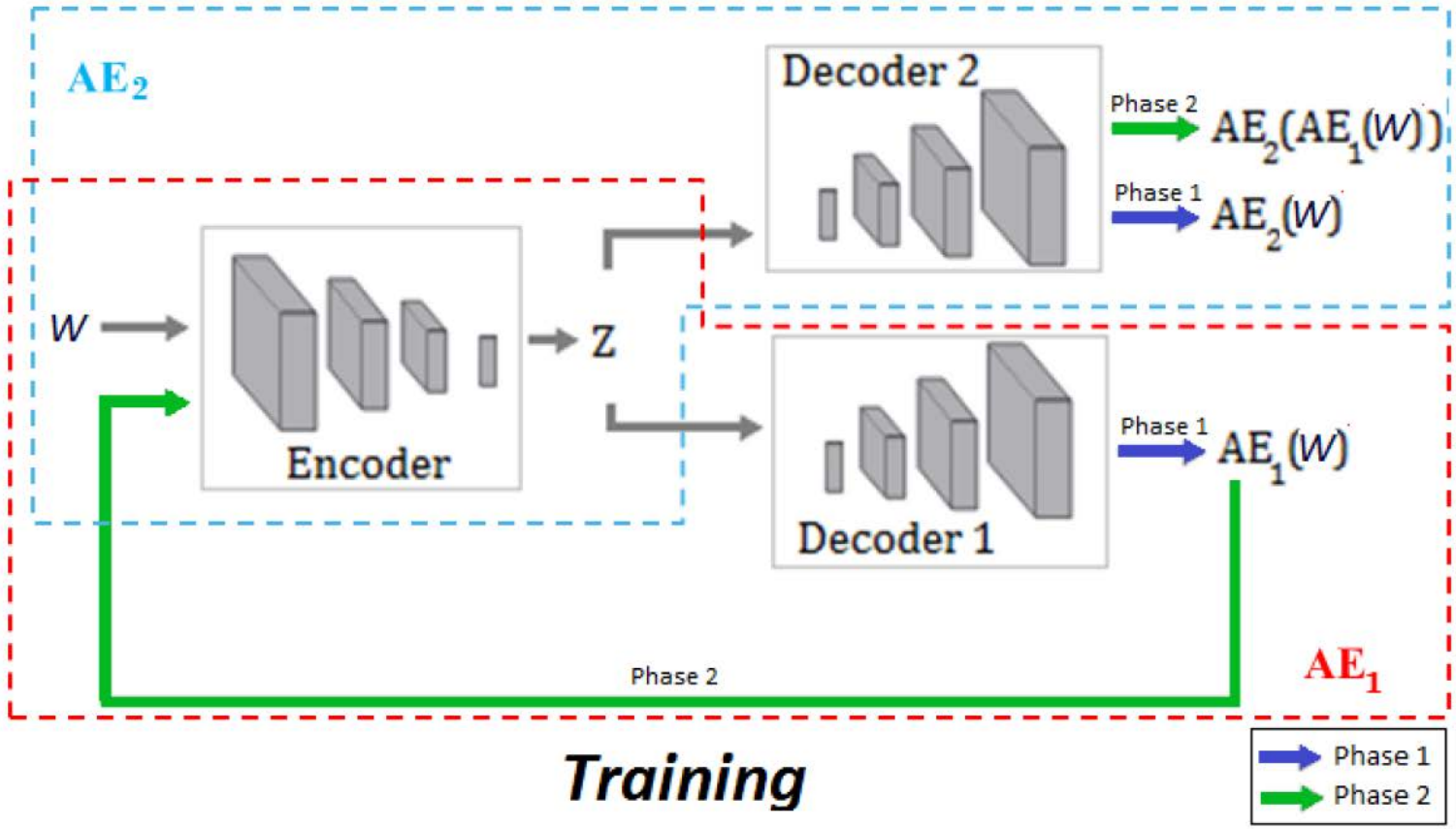
- 그래서 이 두가지 모델을 같이 사용한다.

- 논문에서 딱히 증명은 하지 않았다. 그래서 왜 이 둘을 같이 사용하여 해당하는 문제가 해결되는지 이론적으로 뒷받침하지는 못하지만 뒤에서 모델을 보면 이 둘을 같이 쓰는게 왜 효과적인지 어느 정도는 이해할 수 있다.

- USAD는 크게 3가지 elements:
  - Encoder network E
  - 2개의 Decoder network D_1, D_2

USAD : training

- 2 way train
  - Phase1 : 2개의 AE들은 normal input을 reconstruct하면서 학습
  - Phase2 : AE_1은 reconstruct한 data로 AE_2를 속이려고 한다. AE_2는 real input 과 AE_1의 reconstruct된 값을 구분하도록 학습한다.

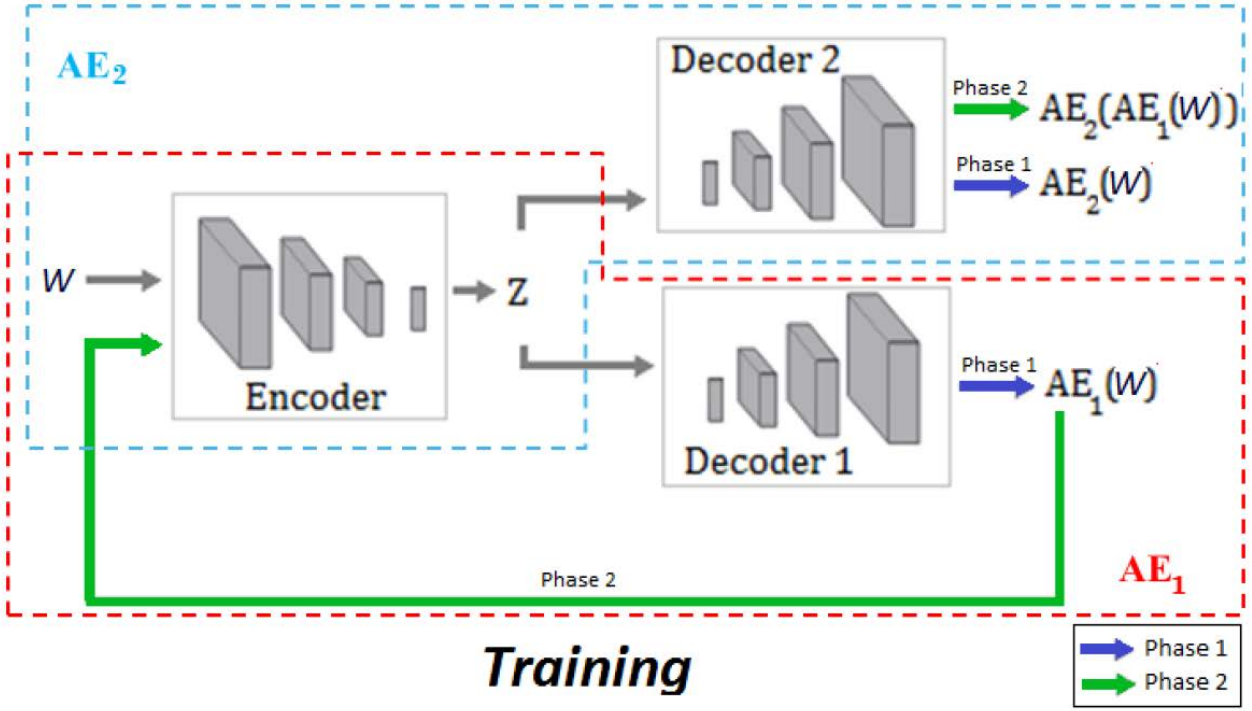


USAD : AE train

- 같은 encoder를 공유하는 2개의 decoder를 학습

< Training objective >

$$L_{AE1} = \left|\left|W - AE_1(W)\right|\right|_2$$
$$L_{AE2} = \left|\left|W - AE_2(W)\right|\right|_2$$

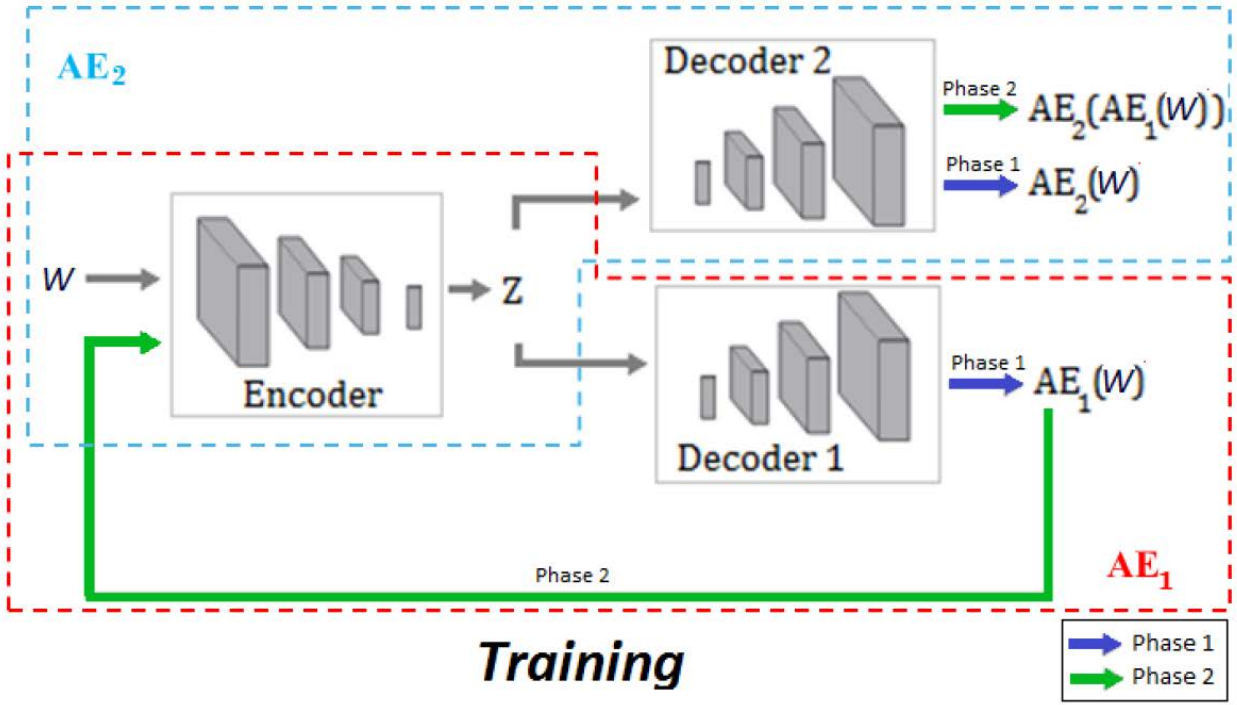


USAD : Adversarial train

- AE_2는 real data 와 AE_1이 만든 data를 distinguish
- AE_1은 fool AE_2 하도록 적대적(adversarial)으로 학습

< Training objective >

$$\min_{AE1} \max_{AE2} \left|\left|W - AE_2\left(AE_1(W)\right)\right|\right|_2$$



USAD : training

- 두 AE의 목적
  - AE_1 : (normal) input을 잘 복원하면서 AE_2를 잘 속이는 모델
  - AE_2 : (normal) input을 잘 복원하면서 AE_1이 복원한 data와 input을 잘 구별하는 모델

$$\min_{AE1} \frac{1}{n} \left|\left|W - AE_1(W)\right|\right|_2 + \left(1 - \frac{1}{n}\right) \left|\left|W - AE_2\left(AE_1(W)\right)\right|\right|_2$$

$$\min_{AE2} \frac{1}{n} \left|\left|W - AE_2(W)\right|\right|_2 - \left(1 - \frac{1}{n}\right) \left|\left|W - AE_2\left(AE_1(W)\right)\right|\right|_2$$

USAD : training

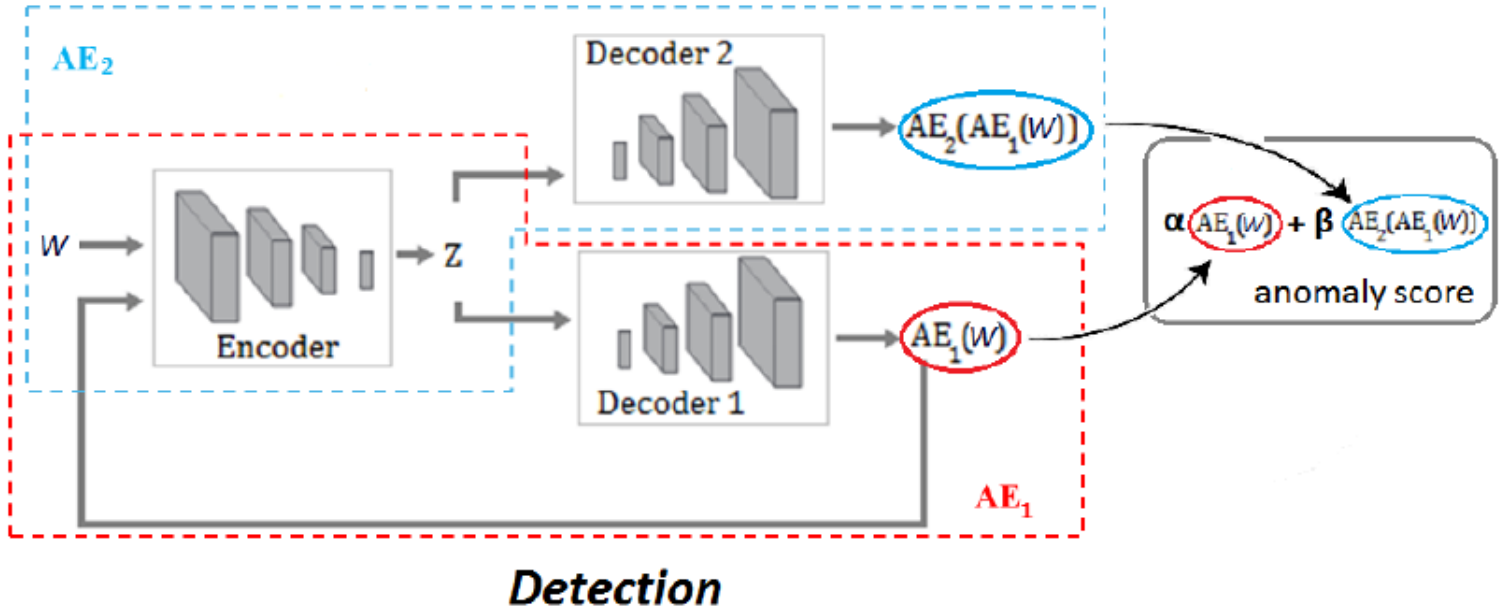
- 두 AE의 목적
  - AE_1 : (normal) input을 잘 복원하면서 AE_2를 잘 속이는 모델
  - AE_2 : (normal) input을 잘 복원하면서 AE_1이 복원한 data와 input을 잘 구별하는 모델

- 뭐가 좋지?
  - AE_1 에서 복원한 data에는 input에 noise가 추가되었다고 할 수 있고
  - 이를 AE_2가 (normal) input과 구별하는 학습과정을 통해
  - normal과 유사한 anomaly data를 detection할 수 있다!!
  - 또한 AE구조를 통해 adversarial training을 안정적으로 할 수 있다.

USAD : inference

- Anomaly score :

$$\alpha\left|\left|W - AE_1(W)\right|\right|_2 + \beta\left|\left|W - AE_2(AE_1(W))\right|\right|_2$$



USAD : inference

- Anomaly score :

$$\alpha\left||W - AE_1(W)\right||_2 + \beta\left||W - AE_2(AE_1(W))\right||_2$$

- $\alpha + \beta = 1$ 이고 이들은 FP와 TP의 trade-off 관계를 결정짓는 parameters
- $\alpha > \beta$: TF, FP가 낮아진다
  -> positive라고 판단하는 경우가 적어짐 (low detection sensitivity)
- $\alpha < \beta$: TF, FP가 높아진다
  -> positive라고 판단하는 경우가 많아짐 (high detection sensitivity)

Implementation

1. Data preprocessing (normalize, split into window)

2. Offline training (taking care that include abnormal)

3. Online using the model (if anomaly score of a window is higher than threshold, declare as abnormal)

Conclusion

- 우리가 속도도 빠르고 성능도 좋다.
- Hyperprameter를 통해 sensitivity를 조절할 수 있다.
- Anomaly detection을 모델을 통해 자동화하니 팀원들이 편해졌다.
- 딥러닝모델을 production하기 위해서는 지속적인 관심이 필요하다.
- Normal data를 input으로 넣기 위해 data를 만드는 과정이 어려웠다.

My Conclusion

- 딥러닝 모델을 Online (streaming) task에서 활용했다는 점이 흥미롭다.
- Academic이나 Research가 아닌 실제 회사의 이야기라는게 흥미롭다.
- 모델의 성능과 속도를 같이 고려해야 한다는 점을 배웠다.
- Transformer같은 최근 기술이나 전통적인 방법론을 추가로 사용하면 더 좋은 모델을 만들 수 있지 않을까? (hybrid)

- 내 석사 논문과 관련성
  - Unsupervised
  - Anomaly detection
  - Tabular, Time series, Network data ??
  - Generative model (Deep learning) + traditional method
  - Real Industry paper
  - 이런 논문을 토대로 보완하는 접근