# 스팀터빈 사전고장감지 인공지능 설계 및 구현

조형배

# 목차

**1. 서론**

**가. 과제의 요약**

**나. 선행 연구 사례**

**2. 과제의 수행**

**가. 데이터 셋 구성**

**나. 데이터 셋 차원 축소**

**다**. Long Short-Term Memory models

**라**. Generative Adversarial Networks

**마**. MAD-GAN

3. **결과 분석**

**가**. **최적** epoch**과 모델 탐색**

**나**. **시기별 이상 감지 결과**

4. **한계점 및 향후 과제**

1. 서론

가. 과제의 요약

본 과제는 기존에 미리 예측하지 못했던 순간적인 스팀터빈의 고장을 사전에 감지할 수 있는 인공지능을 설계하고 구현하는 것을 목적으로 한다. 한국동서발전으로부터 제공 받은 갑작스러운 고장 시점 전 보름의 데이터와 정상 운전 중 3개월의 데이터로 인공지능을 학습하고 테스트하였다. 이 인공지능은 해당 고장 사례에 대해서 고장 3일 전부터 유의미한 이상을 감지하였다.

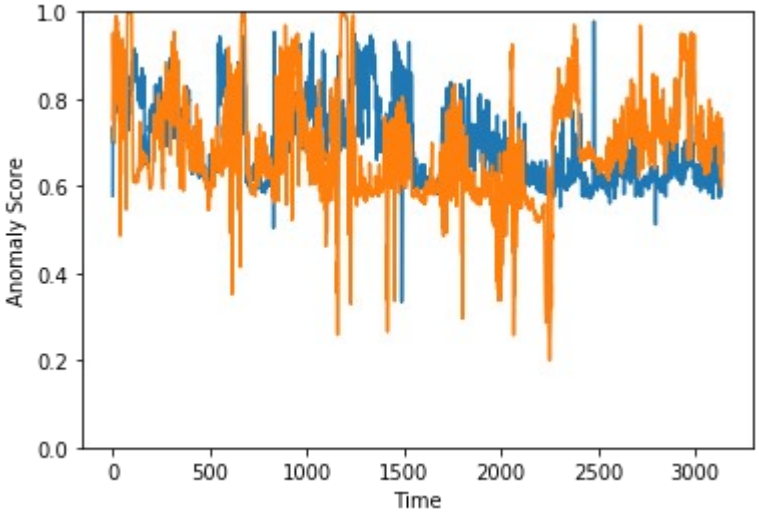



그림1.1 고장 시점 전 11일 동안 정상 데이터와 고장 데이터의 Anomaly Score

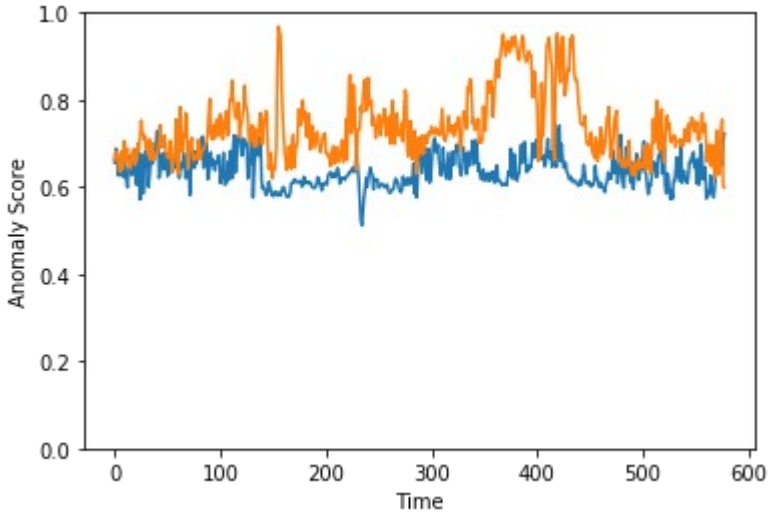



그림1.2 고장 시전 전 2일 동안 정상 데이터와 고장 데이터의 Anomaly Score

그림1.1은 2000년 5월 1일 00시 00분(가로축의 0)부터 갑작스러운 고장이 발생한 2000년 5월 11일 22시 35분(가로축의 3150)의 직전까지 고장이 발생한 스팀터빈과 고장이 발생하지 않은 스팀터빈의 Anomaly Score를 비교한 것이다. 처음 8일 동안은 정상 데이터와 고장 데이터가 비슷한 수준에서 다소 우하향하는 트렌드를 보인다. 그러나 고장 3일 정도 전부터 계속 트렌드를 유지하는 정상 데이터와 달리 고장 데이터에서는 Anomaly Score가 정상 데이터보다 유의미하게 높은 수치를 보인다. 이는 고장 시점 전 2일 동안만 관찰한 그림1.2를 통해 더욱 확실히 파악할 수 있다. 이와 같은 Anomaly Score의 차이를 이용하여 이상을 감지할 수 있다.

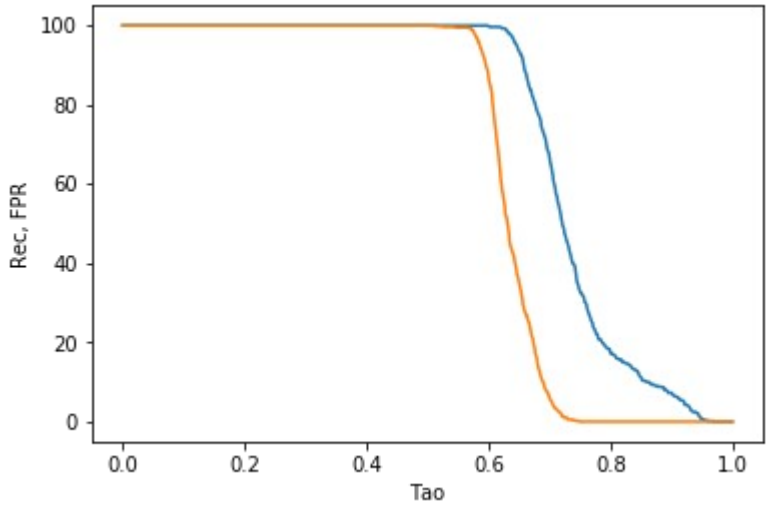



그림1.3 이상 감지 기준(Tao)에 따른 고장 시전 2일 전 데이터의 Rec와 FPR

그림 1.3은 고장 시점 2일 전의 데이터에 대해 0부터 1까지 이상 감지 기준(이하 Tao)을 바꾸어가며 Rec와 FPR을 비교한 것이다. Rec는 실제 고장 데이터를 고장으로 정확하게 예측한 비율이며 FPR은 실제 정상 데이터를 고장으로 잘못 예측한 비율이다. 따라서 Rec는 높고 FPR은 낮을수록 성능이 좋은 것으로 판단할 수 있다. 하지만 이 둘은 Tao에 따른 trade-off가 존재하여 적절한 Tao를 설정하는 것이 중요하다. 해당 고장 사례에 대해서 최적으로 판단되는 0.7의 Tao 하에서 약 66.15%의 Rec와 약 5.70%의 FPR이 관찰되었다.

나. 선행 연구 사례

1) Smart diagnosis of journal bearing rotor systems: Unsupervised feature extraction scheme by deep learning. (2016) [1]

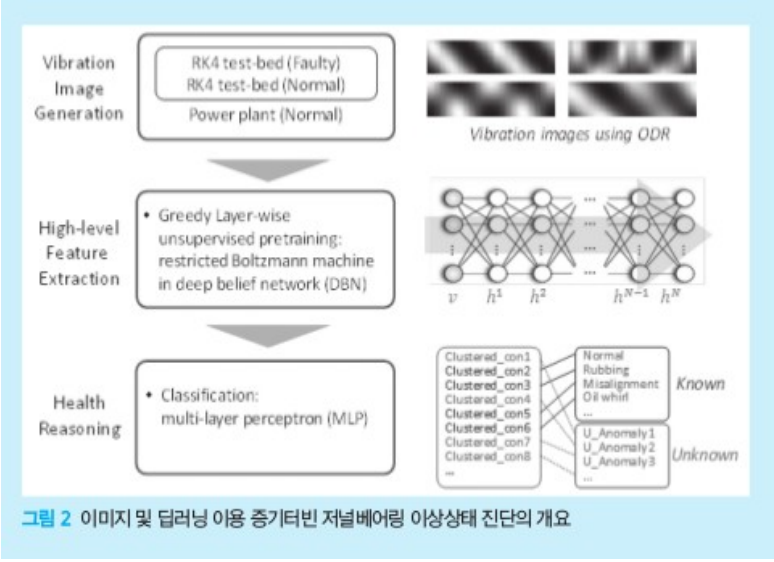



그림1.4 딥러닝 이용 스팀터빈 저널베어링 이상감지 연구의 개요

이 연구는 Oh가 2016년에 수행한 딥러닝을 이용한 스팀터빈 저널 베어링의 이상감지 연구이다. 이 연구에서 Oh는 스팀터빈의 고장 데이터를 획득하는 것이 매우 어렵기 때문에 테스트베드에서 시뮬레이션으로 고장 데이터를 생성하여 실제 정상 데이터와 함께 분류 모델을 학습하였다고 하였다. 이 연구는 스팀터빈의 진동 데이터를 이미지로 변환하여 딥러닝 네트워크의 일종인 CNN을 사용하였다는 점에서 본 과제의 모델과 차이가

있지만 딥러닝으로 스팀 터빈의 이상을 감지했다는 점과 고장 데이터를 시뮬레이션으로 생성하였다는 점에서 참고가 된다.

2) Recurent-Neural-Network-as-a-Tool-for-Parameter-Anomaly-Detection-in-Thermal-Power-Plant (2015) [2]

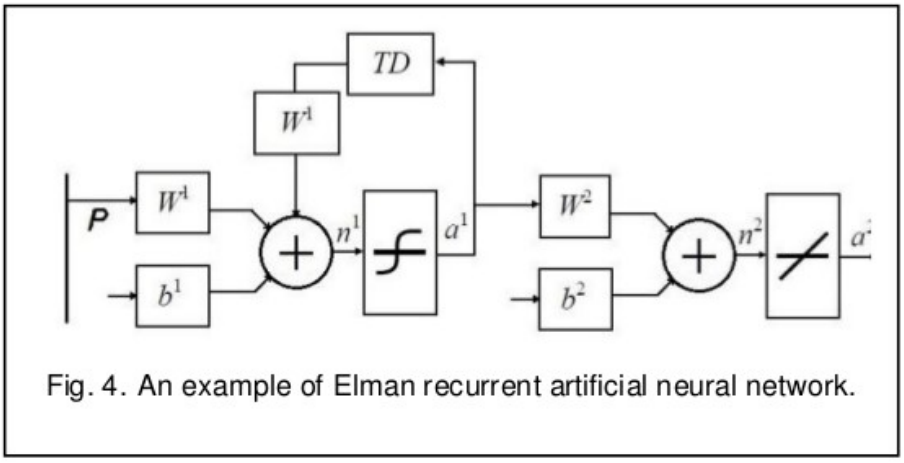


Fig. 4. An example of Elman recurrent artificial neural network.

그림1.5 연구에 사용된 RNN의 구조

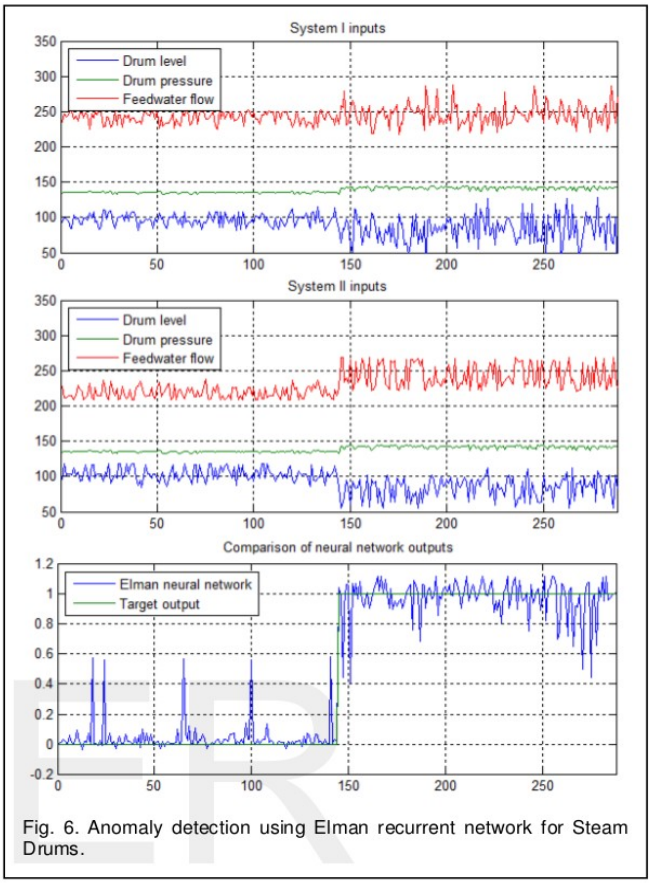


Fig. 6. Anomaly detection using Elman recurrent network for Steam Drums.

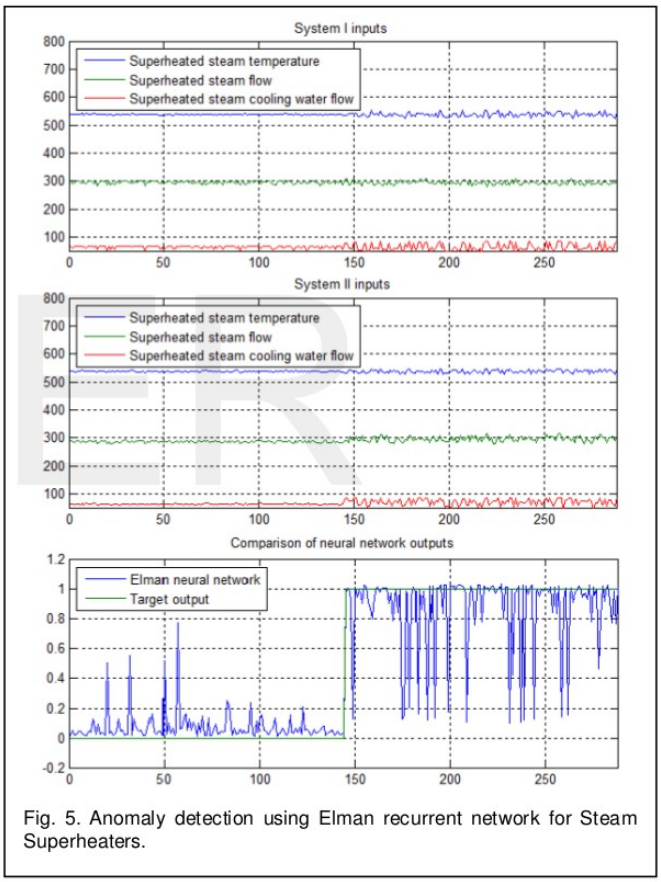


Fig. 5. Anomaly detection using Elman recurrent network for Steam Superheaters.

그림1.6 Steam Drum과 Steam Superheater의 이상 감지 결과

2015년에 수행된 이 연구는 스팀 터빈은 아니지만 Steam Drum과 Steam Superheater에 대한 이상 감지를 수행하였다. 인풋 데이터가 시계열 데이터라는 점에서 순서가 있는 데이터의 처리에 용이한 딥러닝 모델의

일종인 RNN을 사용했다는 점에서 본 과제에 참고가 되었다. 하지만 위의 Oh의 연구와 이 연구 둘 다 순간적인 고장을 '사전에' 감지하는 것을 목표로 하지 않는 것으로 보여 과연 스팀 터빈의 고장을 운전 데이터로 사전에 감지하는 것이 가능한 것인지 문제가 되었다.

3) Prediction of Turbine Failure [3]
이 연구는 웹사이트에 소개된 Dr. Patrick Bangert의 스팀터빈 고장 예측 연구이다.

**Turbine Prediction**

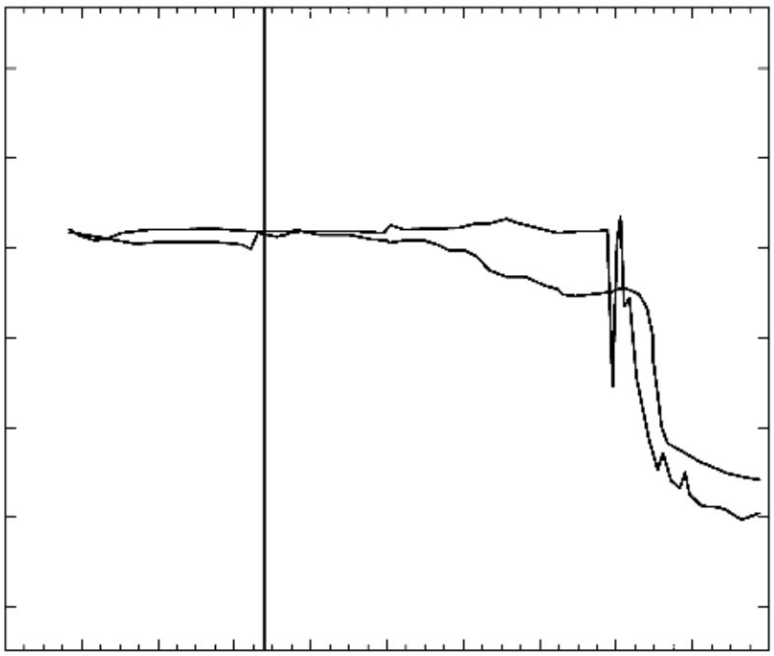


**Figure 1:** Here we see the actual measurement (spiky curve) versus the model output (smooth line) over a little history (left of the vertical line) and for the future three days (right of the vertical line. We observe a close correspondence between the measurement and the model. Particularly the event, the sharp drop, is correctly predicted two days in advance.

그림1.7 고장 시점 전 3일 동안의 실제 값과 모델 값

그림1.7은 스팀터빈의 blade tear가 발생하기 전 3일 동안의 실제 측정 값과 모델의 결과 값을 비교한 것이다. 고장 2일 전부터 두 값은 유의미한 차이를 보이며 이상을 감지할 수 있다. 이 연구를 통해 스팀 터빈의 갑작스러운 고장을 예측하는 것이 가능하다는 것을 확인할 수 있었으나 구체적으로 어떠한 데이터를 사용하였고 어떤 모델로 설계되었는지 파악할 수 없었다. 또한 이 연구의 모델 또한 이상은 감지하지만 그 이상이 구체적으로 blade tear라는 것을 보여주지 못한다는 점에서 본 과제와 유사하다.

2. 과제의 수행

가. 데이터셋 구성

본 과제는 2000년 5월 1일부터 2000년 7월 31일까지 정상 운행한 스팀 터빈으로부터 약 5분 간격으로 생성된 운전 데이터(이하 정상 데이터)와 2000년 5월 1일부터 2000년 5월 15일까지 고장이 발생한 스팀 터빈으로부터 약 5분 간격으로 생성된 운전 데이터(이하 고장 데이터)를 한국동서발전으로부터 제공 받아 수행되었다.

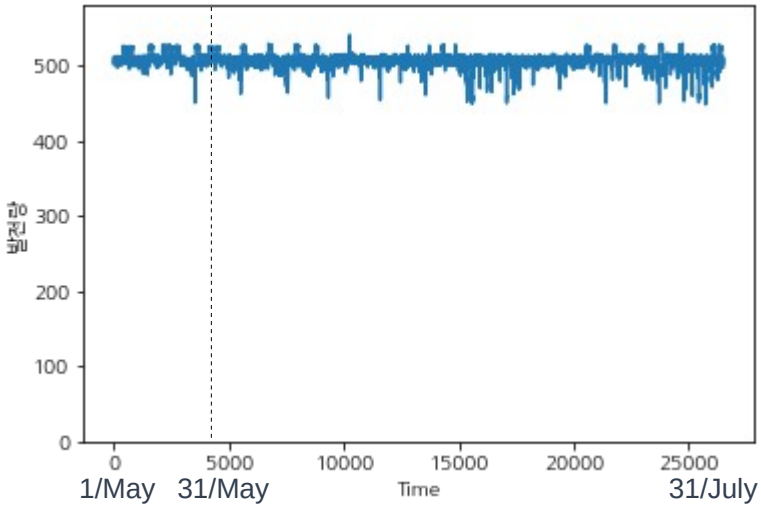



그림2.1 정상 데이터의 시간에 따른 발전량

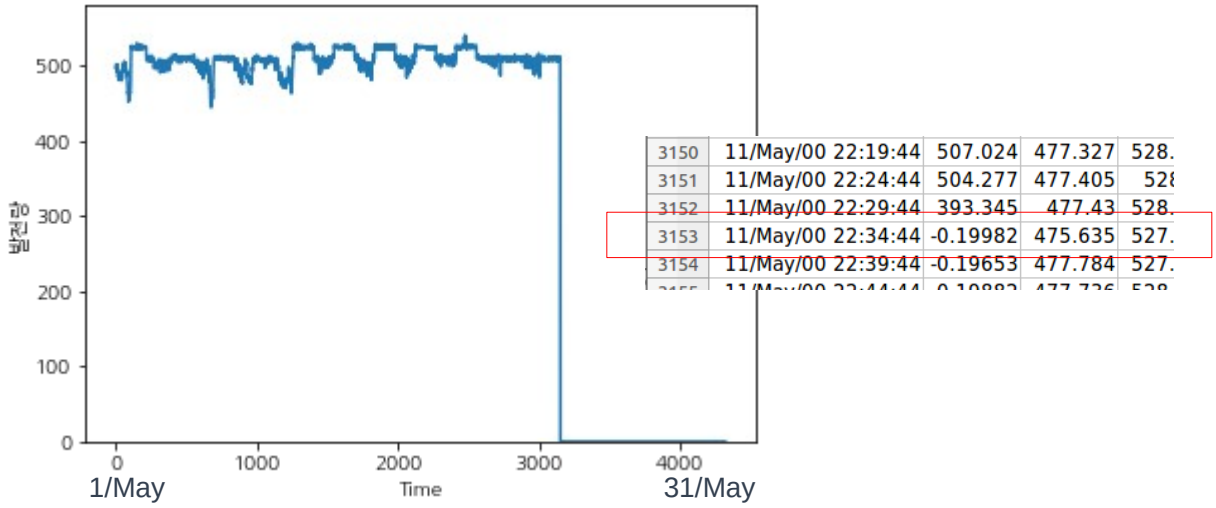



그림2.2 고장 데이터의 시간에 따른 발전량

스팀터빈의 운전데이터는 398개의 변수를 갖는 데이터이다. 이 중에 발전량을 관찰하면 고장 시점을 쉽게 파악할 수 있다. 그림2.1은 정상 데이터의 시간에 따른 발전량이고 그림2.2는 고장 데이터의 시간에 따른 발전량이다. 발전량이 갑자기 0으로 떨어진 5월 11일 22시 35분 경에 고장이 발생하였다. 이미 고장이 발생한 후에 이상을 감지하는 시스템은 현재 한국동서발전에 구축되어있으므로 고장 이후 인공지능이 이상을 감지하는 것은 큰 의미가 없다. 따라서 고장 직전까지의 데이터를 이상에 대한 테스트 데이터로 활용하였다. 또한 성능을 확인하기 위해 같은 시기(5월 1일~ 5월 15일) 정상 데이터를 정상에 대한 테스트 데이터로 활용하였다. 그리고 5월 16일부터 7월 31일까지 데이터를 학습 데이터로 활용하였다.

나. 데이터 셋 차원 축소

본 과제에 활용된 데이터는 비교적 잘 정제되어 있는 데이터였지만 몇 개의 결측치가 있었고 일부 변수의 값이 일정하였다. 이러한 변수들은 모델의 학습에 노이즈를 발생하므로 398개의 변수 중에 37개의 변수를 제거하였다. 또한 이후 PCA 과정에서 단위가 큰 변수는 그만큼 더 영향력이 커지기 때문에 모든 변수의 단위를 -1에서 1로 통일하였다.

본 과제 수행을 위해 사용된 프로세서는 Intel® Core™ i3-6100U CPU로서 연산 능력에 제약이 컸다. 데이터의 차원이 커지면 연산량은 지수적으로 증가한다. 이에 361 차원의 원시 자료의 차원을 축소할 필요가 있었다. 이를 위해 PCA를 수행하였다.

| | | |
|---|---|---|
| 0 | 발전량 | 23DH-MW |
| 1 | HP TBN ROTOR BORE TEMP | 23MK-ROTOR_1_TEMP |
| 2 | LP TBN INLET ROTOR BORE TEMP | 23MK-ROTOR_2_TEMP |
| 3 | LP TBN OUTLET ROTOR BORE TEMP | 23MK-ROTOR_3_TEMP |
| 4 | 1ST STAGE SHELL LOSER INNER TEMP | 23MK-TT_FSS |
| 5 | REHEAT BOWL INNER SURFACE TEMP | 23MK-RT_RBS |
| 6 | CROSSOVER CHAMBER UPPER TEMP | 23MK-TT_XOU |
| 7 | Aux Header Press | 23MK-AP_ASSS |
| 8 | Main Steam Press | 23MK-FP_MSP_PSI |
| . . . | | |
| 391 | Cross Over STM Temp | 23PI_TCO |
| 392 | Cold RH STM Pr | 23PI_PCRS |
| 393 | Cold RH STM Temp | 23PI_TCRS |
| 394 | Hot RH STM Pr | 23PI_PHRS |
| 395 | Hot RH STM Temp | 23PI_THRS |
| 396 | Main STM Pr | 23PI_PMS |
| 397 | Main STM Temp | 23PI_TMS |

그림2.3 원시 데이터의 398개 변수

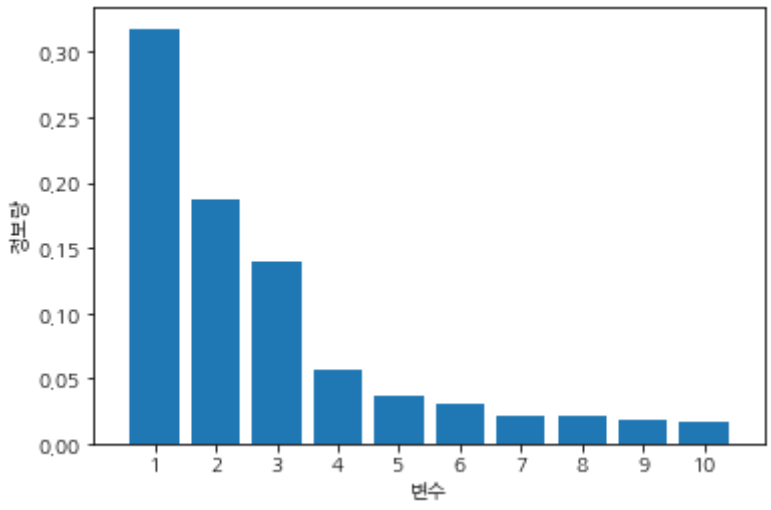



그림2.4 PCA 변수에 따른 정보량

본 과제의 데이터를 PCA를 이용하여 차원을 축소하면 약 3.1% 정도의 정보량을 갖는 6번째 변수 이후 7번째 변수부터 약 2% 내외의 정보량을 가진다. 1번째 변수부터 6번째 변수가 갖는 정보량은 약 76.98%로서 연산의 효율성과 결과의 신뢰성을 고려하였을 때 축소된 차원의 수는 6차원이 적절하다고 판단하였다.

다. Long Short-Term Memory models(이하 LSTM)[4]

LSTM은 Recurrent Neural Networks(이하 RNN)의 일종으로 기본적인 RNN 구조에서 발생하는 vanishing gradient problem을 극복하기 위해 고안되었다.

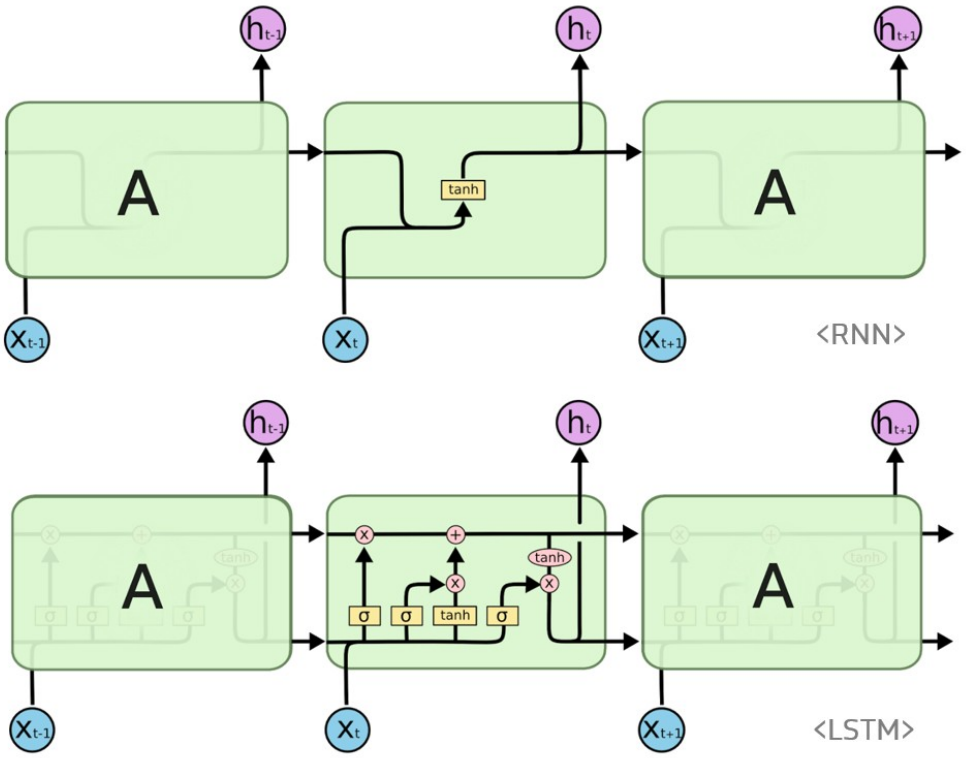



그림2.5 RNN과 LSTM의 기본 구조

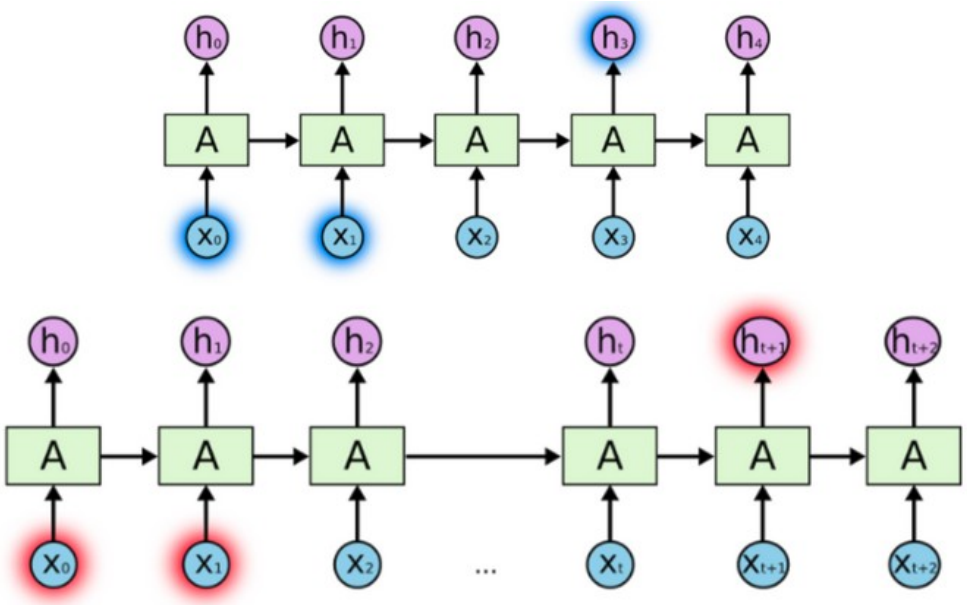



그림2.6 RNN의 vanishing gradient problem

RNN은 그림2.5와 같이 히든 노드가 방향을 가지는 엣지로 연결돼 순환구조를 이루는 인공신경망의 한 종류이다. 이러한 구조 때문에 음성, 문자 등 순차적으로 등장하는 데이터(시퀀스) 처리에 적합한 모델로 알려져 있는데 본 과제도 5분 간격으로 생성되는 데이터가 순차적으로 등장한다는 점에서 RNN은 적합하다. 하지만 RNN은 그림2.6의 아래와 같이 관련 정보와 그 정보를 사용하는 지점 사이 거리가 멀 경우 학습 능력이 저하되는데 이를 vanishing gradient problem이라 한다. LSTM은 RNN의 히든 스테이트에 cell-state를

추가하여 이 문제의 크기를 줄였다. 그림2.4에서 두 노드 사이를 연결하는 방향을 가지는 두 엣지 중에 위의 엣지가 바로 cell-state인데 이 cell-state가 일종의 컨베이어 벨트 역할을 하여 거리가 멀어지더라도 그래디언트가 비교적 잘 전파된다.
본 과제에서는 시퀀스이 길이, 즉 연결된 데이터의 수를 10으로 설정하였다. 따라서 5분 간격의 데이터 10개가 연결되어 50분의 데이터가 하나의 단위를 이룬다. 이렇게 순차적으로 연결된 데이터의 관계와 분포를 적절히 학습하는 한편 vanishing gradient problem을 줄이기 위해 LSTM을 활용한 GAN 모델을 사용하였다.

라. Generative Adversarial Networks(이하 GAN)[5]

Generative Adversarial Networks(이하 GAN)은 2014년 NIPS에서 Ian Goodfellow가 발표한 회귀생성 모델로서 분류를 담당하는 모델(판별자 D)과 회귀생성을 담당하는 두 개의 모델(생성자 G)로 구성되어 있다. 두 모델은 서로의 성능을 개선해 적대적으로 경쟁해 나가는 모델이다.

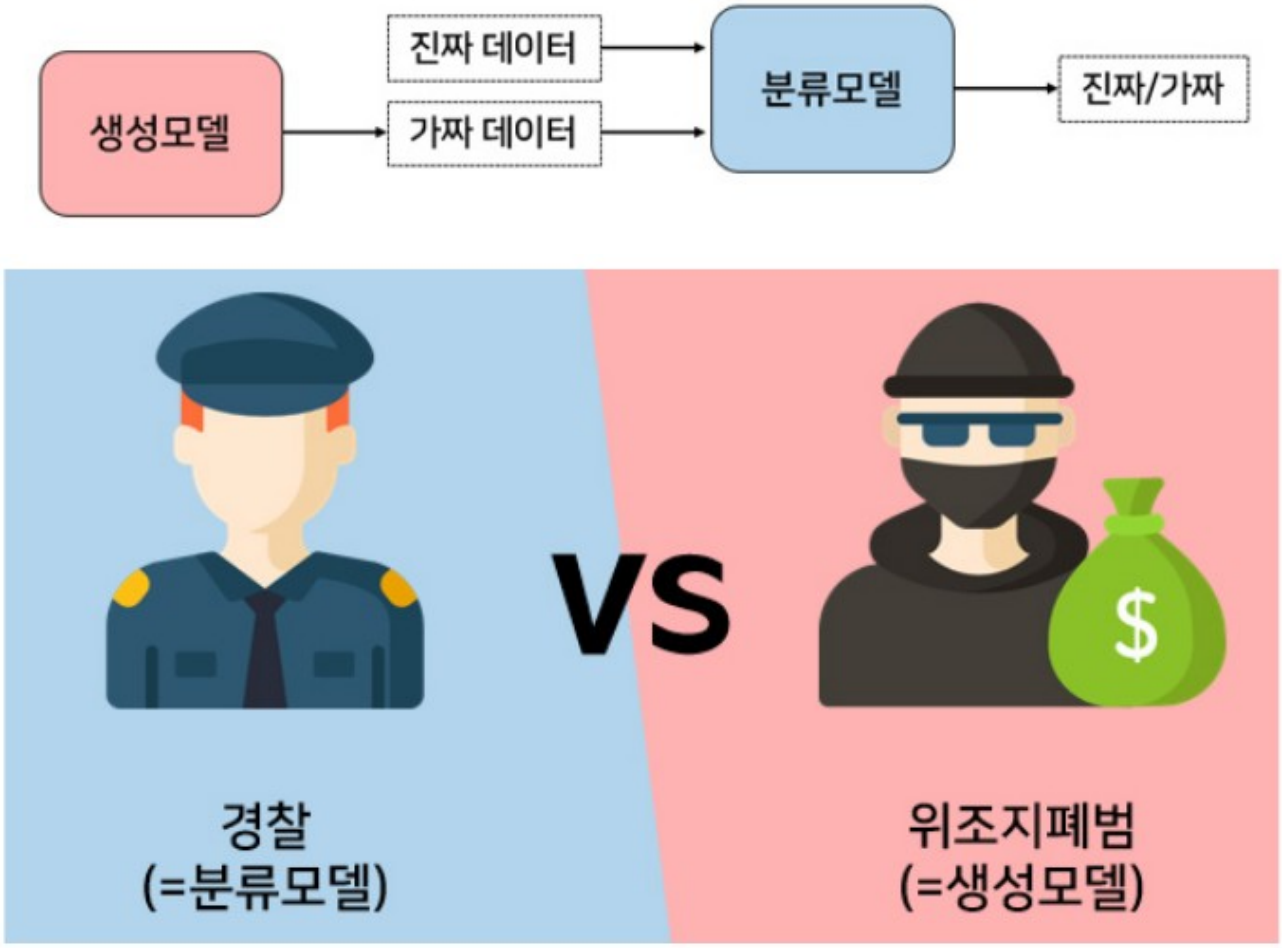



그림2.7 GAN의 생성모델과 분류모델

Ian Goodfellow는 이 두 모델을 경찰과 지폐 위조범의 대립으로 비유하였다. 위조지폐범은 최대한 진짜 같은 화폐를 만들어(생성) 경찰을 속이기 위해 노력하고, 경찰은 진짜 화폐와 가짜 화폐를 완벽히 판별(분류)하여 위조지폐범을 검가하는 것을 목표로 한다. 이러한 경쟁적인 학습이 지속되다 보면 어느 순간 위조지폐범은 진짜와 다를 바 없는 위조지폐를 만들 수 있게 된다.

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[log\ D(x)] + E_{z\sim p_z(z)}[log(1 - D(G(z))]$$

식2.1 GAN의 목적함수

이러한 GAN의 학습내용은 식2.1과 같은 목적함수 V(D,G)에 대한 mini-max problem으로 표현할 수 있다.

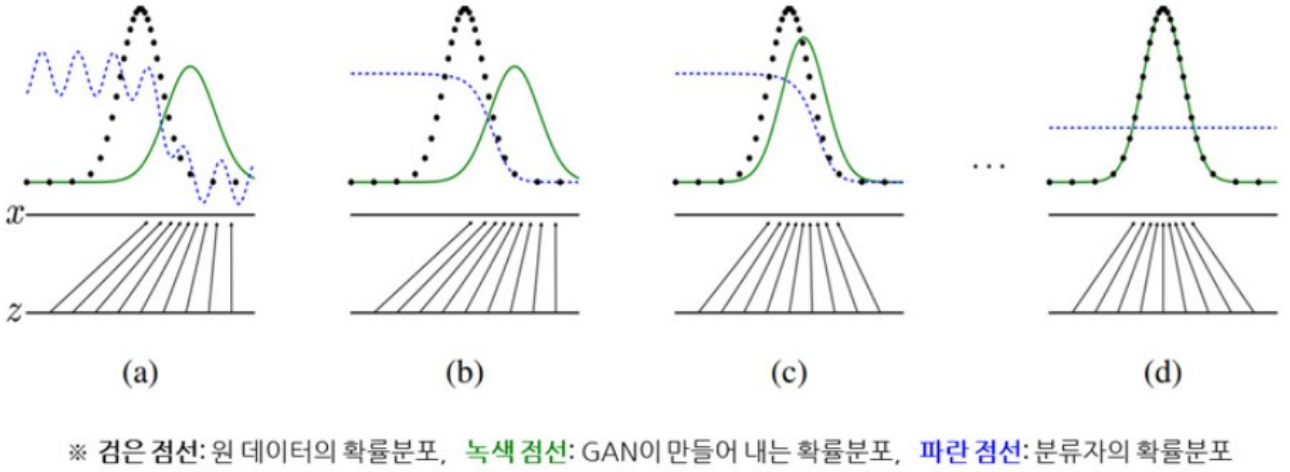



그림2.8 GAN에서 학습을 통해 확률분포를 맞추어 나가는 과정

GAN은 그림2.8과 같이 원 데이터가 가지고 있는 확률분포를 추정하도록 하고, 인공신경망이 그 분포를 만들어 낼 수 있도록 한다는 점에서 군집화 기반의 비지도 학습과 차이가 있다. 머신러닝 알고리즘으로 데이터 대한 확률분포를 모델링 할 수 있게 되면, 원 데이터와 확률분포를 정확히 공유하는 무한히 많은 새로운 데이터를 새로 생성할 수 있음을 의미한다.

마. Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks(이하 MAD-GAN)[6]

MAD-GAN는 2019년 1월 싱가포르 국립대의 Dan Li가 네트워크 시스템으로부터 생성된 다변량의 시계열 데이터를 모니터링하여 사이버 공격의 이상 징후를 발견하기 위해 GAN의 구조를 기반으로 설계한 모델이다. 사이버 공격과 스팀 터빈의 고장은 서로 다른 성격을 가지지만 본 과제의 대상이 되는 스팀 터빈의 운전 데이터도 다변량의 시계열 데이터라는 점에서 MAD-GAN 모델이 적합하다고 판단하였다.

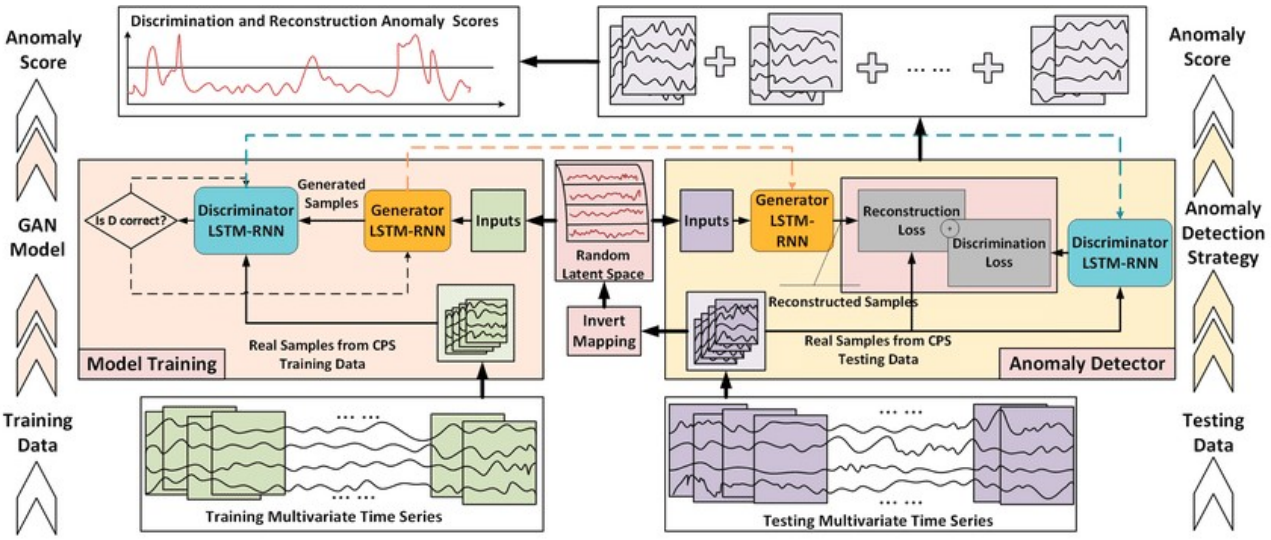



그림2.9 MAD-GAN Architecture

**Algorithm 1** LSTM-RNN-GAN-based Anomaly Detection Strategy

**loop**
  **if** epoch within number of training iterations **then**
    **for** the $k^{th}$ epoch **do**
      Generate samples from the random space:
      $Z = \{z_i, i = 1, ..., m\} \Rightarrow G_{rnn}(Z)$
      Conduct discrimination:
      $X = \{x_i, i = 1, ..., m\} \Rightarrow D_{rnn}(X)$
      $G_{rnn}(Z) \Rightarrow D_{rnn}(G_{rnn}(Z))$
      Update discriminator parameters by minimizing(descending) $D_{loss}$:
      $\min \frac{1}{m} \sum_{i=1}^{m} [-\log D_{rnn}(x_i) - \log(1 - D_{rnn}(G_{rnn}(z_i)))]$
      Update discriminator parameters by minimizing(descending) $G_loss$ :
      $\min \sum_{i=1}^{m} \log(-D_{rnn}(G_{rnn}(z_i)))$
      Record parameters of the discriminator and generator in the current iteration.
    **end for**
  **end if**
  **for** the lth iteration **do**
    Mapping testing data back to latent space:
    $Z^k = \min_Z Er(X^{tes}, G_{rnn}(Z^i))$
  **end for**
  Calculate the residuals:
  $Res = \mid X^{tes} - G_{rnn}(Z^k) \mid$
  Calculate the discrimination results:
  $Dis = D_{rnn}(X^{tes})$
  Obtain the combined anomaly score:
  **for** k,j and s in ranges **do**
    **if** j+s=k **then**
      $R = \lambda Res + (1 - \lambda) Dis$
      $DRS_k = \frac{\sum R_{j,s}}{L_k}$
    **end if**
  **end for**
**end loop**

알고리즘2.1 MAD-GAN의 알고리즘

그림2.9의 왼쪽은 MAD-GAN의 두 모델(생성자 G와 판별자 D)의 학습이 이루어지는 영역이고 오른쪽은 그 모델을 이용하여 이상을 감지하는 영역이다. 생성자 G와 판별자 D는 시계열 데이터를 다루기 위해 2.다.에서 살펴본 LSTM 셀로 구성된다. 생성자 G와 판별자 D가 학습되는 과정은 일반적인 GAN모델과 크게 다르지 않다. 판별자 D는 정상 데이터로만 구성된 실제 학습 데이터와 생성자 G가 생성한 가짜 데이터를 정확히 구분하도록 학습된다. 반면에 생성자 G는 판별자 D를 속일 수 있을 만큼 실제 학습 데이터(즉 정상 데이터)와 비슷한 가짜 데이터를 생성하도록 학습된다.

여러번의 반복 학습을 마친 판별자 D는 매우 민감하게 정상 데이터를 구분할 수 있으며 생성자 G는 정상 데이터의 확률 분포를 학습하여 매우 유사한 데이터를 생성해낼 수 있다. MAD-GAN은 이상 감지를 위해 이 두 가지 모델을 모두 사용한다. 판별자 D가 이상을 감지하는 방식은 직관적으로 이해될 수 있다. 판별자 D는 정상 데이터와 다른 고장 데이터에 대해 민감하게 반응할 것이다. 반면, 생성자 G가 이상을 감지하는 방식을 이해하려면 다음의 이론이 필요하다.(본 과제에서는 이상을 검출하는데 판별자 D만이 활용되었기 때문에 이 부분은 넘어가도 좋습니다.)

앞서 서술된 식2.1 GAN의 목적함수를 다시 보면 생성자 G에 인풋 Z가 있는 것을 볼 수 있다. 판별자 D가 인풋을 갖는 것은 당연하지만 생성자 G가 인풋을 갖는 것은 직관적이지 않다.

$$\min_G \max_D V(D,G) = E_{x \sim p_{data}(x)}[log\ D(x)] + E_{z \sim p_z(z)}[log(1 - D(G(z))]$$

식2.1 GAN의 목적함수

생성자 G의 인풋이 되는 Z는 latent space라고도 부르는데 일반적으로 가우시안분포(정규분포)로부터 샘플링한 데이터를 의미한다. 생성자 G는 사실 가우시안분포를 따르는 임의의 값들을 어떤 데이터로 대응시켜주는 함수인 것이다(The trained generator G is a mapping from the latent space to real data space.). 인풋이 되는 Z가 가우시안분포라는 특정 확률 분포를 따르기 때문에 생성자 G의 아웃풋도 특정 확률 분포를 따르게 되고 이 확률 분포는 학습을 반복하면서 점점 실제 데이터의 확률 분포와 유사해진다. 여기에 놀라운 GAN의 특성이 있다. 생성자 G가 학습한 딥러닝 매핑(Z->G(Z))이 단순히 불연속적인 1:1 매칭이 아니라 실제 데이터의 확률분포를 정확히 표현하고 있어서, 입력에서의 약간의 변화는 출력에서도 부드러운 변화로 표현이 가능하다는 것이 증명되었다(the smooth transitions of latent space)[7]. 그림2.10는 이러한 GAN의 특성을 이용하여, 오바마의 얼굴을 만들어낸 latent space에서 입술에 해당하는 부분만 벡터 연산을 통해 산술적으로 대체하여 만들어낸 가짜 영상의 일부이다.

그림2.10 GAN의 the smooth transitions of latent space 특성을 이용하여 합성한 오바마 전 미국 대통령의 연설 영상

GAN의 이러한 특성에 의해 MAD-GAN의 생성자 G도 마찬가지로 인풋이 latent space 상에서 가까이 위치한다면 비슷한 샘플을 생성한다. 따라서 정상 테스트 데이터라면 이를 invert mapping을 통해 해당 데이터에 대응하는 Z를 찾고 이를 다시 생성자 G에 입력하여 샘플을 생성하면 처음의 정상 데이터와 생성된 샘플은 유사할 것이다. 반면에 고장 테스트 데이터라면 이 데이터에 대응하는 Z를 찾고 이를 다시 생성자 G에 입력하면 정상 데이터의 확률 분포를 학습한 생성자 G는 처음의 고장 테스트 데이터를 생성하지 못하고 차이가 큰 샘플을 생성할 것이다. 이 차이는 식2.2와 같이 계산되고 Reconstruction Loss라고 한다.

$$Res(X_t^{tes}) = \sum_{i=1}^{n} | x_t^{tes,i} - G_{rnn}(Z_t^{k,i}) |$$

식2.2 Reconstruction Loss

판별자 D와 생성자 G로부터 loss는 식2.3과 같이 계산된다. 여기에서 λ는 하이퍼파라미터로서 0과 1사이의 값으로 설정할 수 있다.

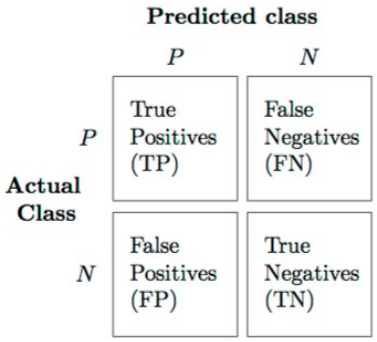
$$L_t^{tes} = \lambda Res(X_t^{tes}) + (1-\lambda) D_{rnn}(X_t^{tes})$$

식2.3 the anomaly detection loss

3. 결과 분석

가. 최적 epoch과 모델 탐색

최적의 epoch을 탐색하기 위해 epoch99, 299, 499, 599, 699에 대해서 결과를 비교 분석하였다. 분석에 가장 주요하게 사용된 평가 기준은 Rec와 FPR이다.




Rec = TP / ( TP + FN )

FPR = FP / ( FP + TN)

그림 3.1 모델 평가 기준

여기에서 TP는 실제 고장 데이터를 고장으로 분류한 것이고 FN은 실제 고장데이터를 정상으로 분류한 것이다. 또한 TN은 실제 정상 데이터를 정상으로 분류한 것이고 FN은 실제 정상데이터를 고장으로 분류한 것이다. 따라서 Rec는 실제 고장 데이터 중에 고장으로 맞게 분류된 비율로서 높을수록 좋고 FPR은 실제 정상 데이터 중에서 고장으로 잘못 분류된 비율로서 낮을수록 좋다.

이 실험은 각각의 경우에 대해 학습과 테스트를 거쳐야 하기 때문에 많은 연산 시간이 소요되었다. 따라서 테스트 데이터의 일부에 대해서 실행되었다. 이 실험에 사용된 테스트 데이터는 정상 테스트 데이터와 고장 테스트 데이터 각각의 마지막 250개를 합친 500개의 데이터이다. 이 부분에서 착각에 의한 실수가 있었는데 정상 테스트 데이터의 마지막은 5월 15일 23시 55분 경이지만 고장 테스트 데이터의 마지막은 고장 직전인 5월 11일 23시 30분 경이다. 같은 시기에 대해 실험을 수행해야 더 정확한 평가가 가능하지만 이 실수가 최적의 epoch을 판단하는데 큰 영향을 미치지는 않았을 것으로 생각한다.

1) 판별자 D만 사용한 고장감지결과

| Epoch | N(정답 개수) | Accu(%) | Rec(%) | FPR(%) |
|---|---|---|---|---|
| 99 | 161 | 31.6 | 27.09 | 62.55 |
| 299 | 340 | 66.8 | 37.85 | 1.992 |
| 499 | 490 | 96.27 | 99.2 | 3.586 |
| 599 | 363 | 71.62 | 54.98 | 9.96 |
| 699 | 411 | 80.75 | 96.41 | 32.27 |

2) 생성자 G만 사용한 고장감지결과

| Epoch | N(정답 개수) | Accu(%) | Rec(%) | FPR(%) |
|---|---|---|---|---|
| 99 | 238 | 46.76 | 6.773 | 11.55 |
| 299 | 258 | 50.69 | 39.04 | 35.86 |
| 499 | 325 | 63.85 | 49.8 | 19.92 |
| 599 | 248 | 48.72 | 11.55 | 12.35 |

| | | | | |
|---|---|---|---|---|
| 699 | 313 | 61.49 | 64.14 | 39.04 |

3) 판별자 D와 생성자 G를 함께 사용한 고장감지결과

| Epoch | N(정답 개수) | Accu(%) | Rec(%) | FPR(%) |
|---|---|---|---|---|
| 99 | 256 | 50.29 | 2.39 | 0.0 |
| 299 | 260 | 51.08 | 3.984 | 0.0 |
| 499 | 475 | 93.32 | 92.83 | 3.187 |
| 599 | 332 | 65.23 | 38.65 | 5.976 |
| 699 | 361 | 70.92 | 54.58 | 10.36 |

표3.1 epoch과 사용 모델에 따른 고장감지결과 비교

표3.1의 결과를 보면 epoch499에서 판별자 D만 사용한 결과가 확연히 우수한 것을 확인할 수 있다. 생성자 G도 epoch499에서는 Rec 49.8%, FPR 19.92%로 유의미한 이상감지 성능을 보였지만 판별자 D의 Rec 99.2%, FPR 3.586%에는 미치지 못하였고 이 둘을 함께 사용하면 오히려 판별자 D를 단독으로 사용할 때보다 성능이 감소하여 생성자 G는 판별자 D를 보완하지도 못하는 것으로 보인다. 따라서 본 과제에서는 epoch 499에서 판별자 D만을 사용하여 스팀 터빈의 고장을 감지한다. 이하의 결과는 이 조건에서 얻은 것이다.

나. 시기별 이상 감지 결과

고장이 발생하기 전에 최대한 일찍 이상을 감지하는 것이 좋다. 고장이 나타나기 얼마 전부터 징후를 감지할 수 있는지 관찰하기 위해 실험을 수행하였다.

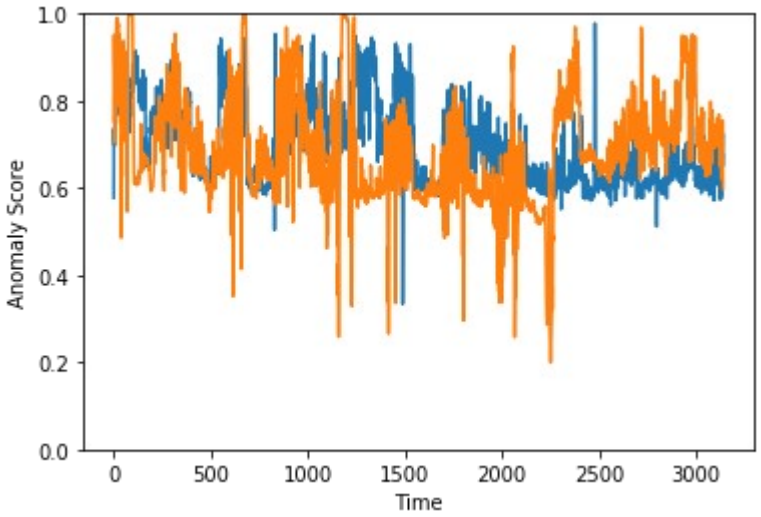



그림3.2 5월1일부터 고장 직전(5월 11일 23시 30분 경)까지 시간에 따른 이상 값 비교

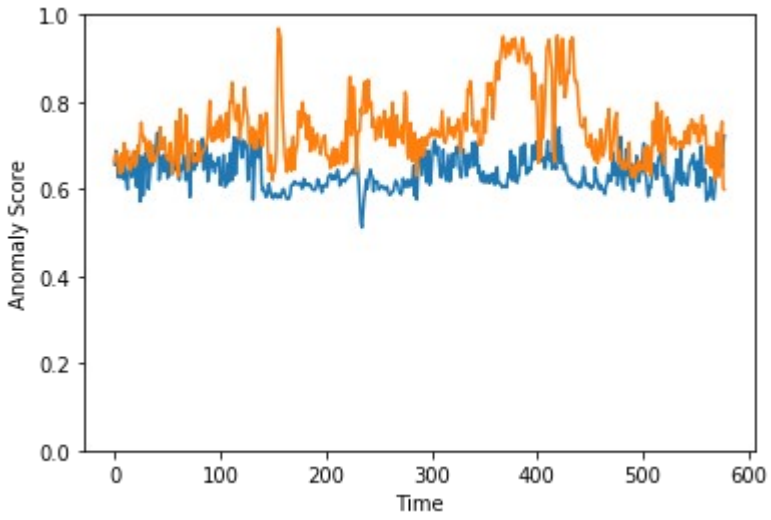


그림3.3 고장 직전 2일 간 이상 값 비교

그림3.2는 테스트 데이터 전체 기간에 대해 정상 데이터와 고장 데이터의 시기별 이상 값( Anomaly Score) 비교이다. 주황색이 고장 데이터이고 파란색이 정상 데이터에 해당한다. 두 데이터는 비슷한 수준에서 우하향하는 경향을 보이다 가로축의 약 2270 지점부터 고장 데이터의 이상 값이 상승하여 정상 데이터의 위에 위치한다. 이 지점은 고장의 약 3일 전이다. 이 시점부터 고장의 징후가 모델에 의해 감지된 것으로 볼 수 있다. 고장의 징후는 고장 직전 2일 간의 이상 값을 비교한 그림3.3을 보면 더욱 분명히 나타난다.

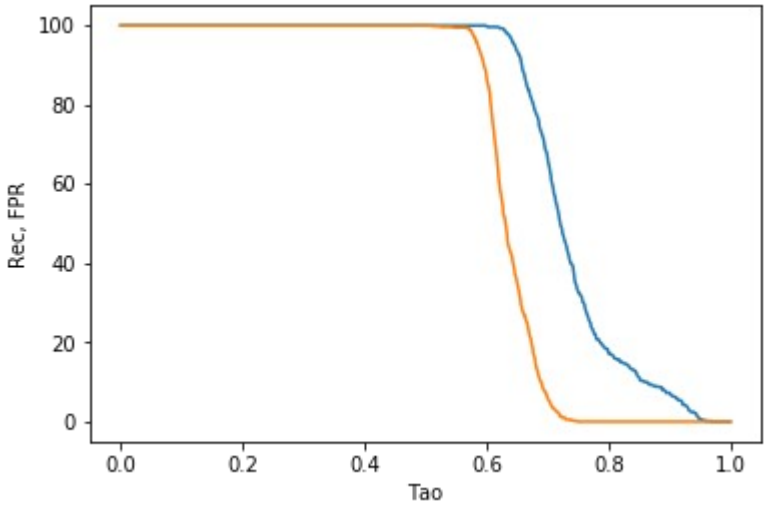



그림3.4 고장 직전 2일 간 데이터에 대한 Tao에 따른 Rec와 FPR의 변화

한편, 이상 값에 대해 이상인지 정상인지 판단하는 기준이 필요하다. 이를 Tao라고 한다. 이 기준은 가설검정에서의 유의수준과 유사한 역할을 한다고 볼 수 있다. 그림3.4는 고장 직전 2일 간 데이터에 대해서 이 기준에 따른 Rec와 FPR 변화를 나타낸다. 주황색이 FPR이고 파란색이 Rec이다. Tao가 너무 낮으면 모든 데이터를 고장으로 감지하여 Rec와 FPR이 모두 100%에 가까워지고 반대로 Tao가 너무 높으면 모든 데이터를 정상으로 감지하여 Rec와 FPR이 모두 0%에 가까워진다. 본 과제에서는 약 66.15%의 Rec와 약 5.70%의 FPR이 관찰되는 0.7의 Tao가 최적이라고 판단하였다. 표3.2는 이 기준에 의한 날짜별 이상 감지 결과이다.

| D-day | Accu(%) | Rec(%) | FPR(%) |
|---|---|---|---|
| 1 | 82.28 | 72.76 | 5.862 |
| 2 | 75.81 | 59.31 | 5.517 |
| 3 | 68.99 | 46.55 | 6.552 |

표3.2 고장 직전 3일 간 날짜별 이상 감지 결과

표3.2를 통해 고장의 징후는 고장에 가까워질수록 강하게 감지되는 것을 확인할 수 있다.

4. 한계점 및 향후 과제

가. 고장에 대한 구체적인 정보 부재

본 과제는 고장의 징후를 미리 감지해서 알려줄 뿐 어디에서 어떤 문제가 발생할 수 있는지는 전혀 알려주지 않는다. 본 과제의 인공지능을 통해 고장의 징후를 감지했을때 구체적인 진단을 내릴 수 있는 인공지능을 설계하는 것은 향후의 과제가 될 것이다.

나. 계절성의 존재

본 과제는 학습에 사용된 데이터가 생성된 시기와 테스트에 사용된 데이터가 생성된 시기가 다르다. 이상값이 시간에 따라 우하향하는 경향을 보이는 것도 이러한 원인이 있지 않을까 추측한다. 학습 데이터가 생성된 시기에서 멀어질수록 인공지능은 정상에서 멀어진다고 판단하는 것이다. 이 추측과는 별개로 계절성을 고려하여 데이터셋을 구축하고 인공지능을 설계한다면 성능을 더욱 향상시킬 여지가 있는 것은 분명하다.

다. 특정 고장 사례에 과적합

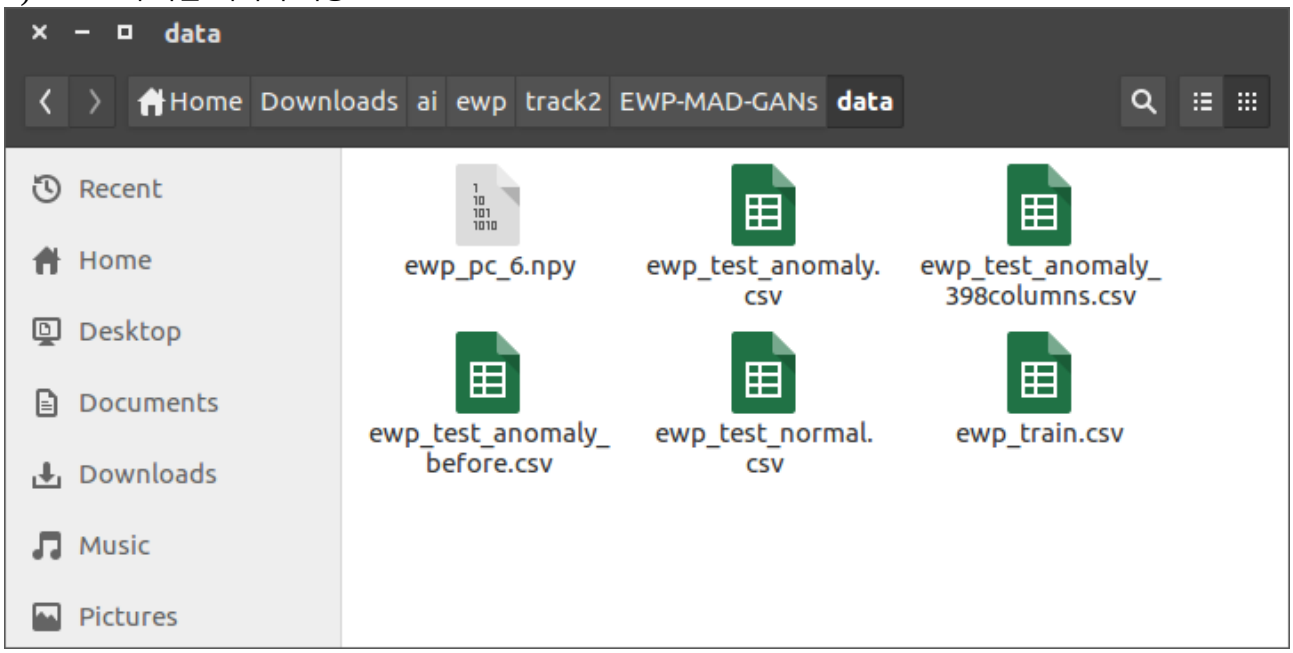
본 과제의 인공지능의 검증과 세팅에 사용한 고장 사례는 하나이다. 본 과제에서는 Tao, epoch 등 여러 세팅 값이 존재한다. 이 값은 하나의 고장 사례를 보며 최적화를 하였기 때문에 일반적인 스팀터빈의 고장 사례에서는 최적이 아닐 수 있다. 이 문제는 여러 고장 사례에 대해서 검증함으로써 극복할 수 있다.

## 5. 프로그램 메뉴얼

### 가. 학습

EWP-MAD-GAN 디렉토리에서

1)./data/ 에 학습 데이터 저장




2)./experiments/settings/ewp.txt 작성


2)RGAN.py 실행
>>python RGAN.py --settings_file ewp

실행 시 현재 인공지능의 학습된 내용은 덮어쓰게 된다.

나. 테스트

1)./data/에 테스트 데이터 저장

2)./experiments/settings/ewp_test.txt 작성

3)AD.py 실
>>python AD.py --settings_file ewp_test

6. 참고문헌

[1] Oh, Hyunseok, et al. “Smart diagnosis of journal bearing rotor systems: Unsupervised feature extraction scheme by deep learning.” (2016).

[2] A. Hajdarevic, et al. ”Recurent-Neural-Network-as-a-Tool-for-Parameter-Anomaly-Detection-in-Thermal-Power-Plant” (2015).

[3] Dr. Patrick Bangert, “Prediction of Turbine Failure”.

[4] “RNN과 LSTM을 이해해보자!”, https://ratsgo.github.io/natural%20language%20processing/2017/03/09/rnnlstm/

[5] 장준혁, “GAN의 활용 사례와 발전방향”, https://www.samsungsds.com/global/ko/support/insights/Generative-adversarial-network-AI-3.html

[6] Dan Li, et al. “MAD-GAN: Multivariate Anomaly Detection for Time Series Data with Generative Adversarial Networks”(2019).

[7] R. Alec, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” (2015).