빅데이터응용보안 프로젝트

**「ICS 공격 탐지」**



5조

201820685 김동환

201920659 오수빈

201920664 정예은

목차

[**Ⅰ. 개요**](#_uj7xijnnond) **2**

[**Ⅱ. 데이터 설명**](#_giqhk9bruo6k) **2**

[1) P1: Boiler Controllers](#_y85r78p1qnau) 3

[1-1) P1-PC(Pressure controller)](#_ydi0ivx095xy) 4

[1-2) P1-FC(Flow rate controller)](#_okcdvtdj6lj0) 5

[1-3) P1-LC(Level controller)](#_an9zgglahi71) 6

[1-4) P1-TC(Temperature controller)](#_xckz0spiejvm) 7

[1-5) P1-CC(Cooling controller)](#_mfnhcf2l8a85) 8

[2) P2: Turbine Controllers](#_t049bmnwb24p) 8

[2-1) P2-SC(Speed Control)](#_c51ezzllodjk) 9

[2-2) P2-TRIP(Over-speed and over-vibration trips)](#_5nspkn49nlyl) 9

[2-3) P2-SC와 P2-TRIP은 연관성이 커서 하나의 attack scenario로 취급](#_qq5rfu58jt6x) 10

[3) P3 : Water Treatment Controllers](#_mxlg4ntevpop) 11

[3-1) P3-LC : Level Control](#_x9dpq4y5ehhj) 12

[**Ⅲ. 평가 방식**](#_ohapo25dwdlv) **12**

[**Ⅵ. 모델**](#_wlh042md76wc) **13**

[1. Isolation Forest](#_m55w2jfbkn8a) 14

[1) contamination 값 튜닝](#_tphtbskgutx8) 15

[2) PCA 적용](#_xan0qvy67vt1) 16

[3) 오탐 줄이기](#_i45pg08b0vkg) 19

[4) 최종 구조도](#_s8erukadr625) 20

[2. Bidirectional LSTM Autoencoder, Bidirectional GRU Autoencoder](#_u2vd21vvhq9b) 21

[1) 데이터 전처리](#_eaz8hv7ac8kv) 21

[2) Bidirectional LSTM Autoencoder](#_h31qvjdkg3bx) 21

[3) Bidirectional GRU Autoencoder](#_noq1qijcdeqp) 31

[3. Isolation Forest + Bidirectional LSTM/GRU Autoencoder](#_eg2apxxh8xzi) 34

[4. LSTM + GRU](#_2wpzghvyevg3) 34

[**Ⅴ. 결론**](#_aemkqoalxn3z) **35**

[**참고문헌**](#_hg7bzv8guyzh) **36**

# 

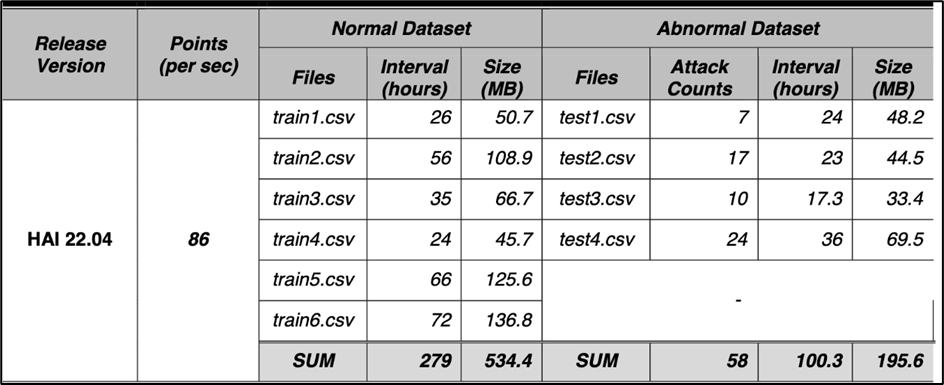
# **Ⅰ. 개요**

최근 ICS(Industrial Control System, 산업제어시스템)에 대한 사이버 보안 위협이 지속적으로 증가하고있으며 세계 각국에서 실제로 피해를 입은 사례들이 존재한다[1-2]. ICS는 가용성이 매우 중요하기 때문에 사이버 공격은 국가와 사회에 돌이킬 수 없는 막대한 피해를 일으킬 수 있어 세계 각국이 이에 대한 보안기술 개발에 전념하고 있다. ICS 보안 문제가 심각해지는 이유는 ICS에 대한 보안 인식이 약하며 전문성도 부족하기 때문이다. 그렇기 때문에 우리는 보안전문가로서 앞으로 ICS에 대한 보안 인식을 강화하고 사이버 공격을 막기 위한 다양한 보안 기술들을 연구할 필요가 있다. 이러한 이유로 “ICS 공격 탐지” 방법에 대한 연구를 진행하고자 한다.

국가보안기술연구소는 GE, Emerson, Siemens 등의 산업용 제어기기, 센서, 액추에이터를 이용한 발전 모사 제어시스템 테스트베드를 2019년에 구축하였다. 이를 기반으로 HAI 를 지속적으로 개발하고 있으며, 우리는 가장 최신인 22년 4월에 공개한 HAI 22.04 버전을 데이터 셋으로 사용한다[3]. HAI 20.07 버전과 21.03 버전은 대회 데이터 셋으로 사용되었지만, 22.04 버전은 공개된지 얼마 되지 않아 관련 연구와 대회가 아직 없다. 또한 기존의 대회에서 사용된 모델을 참고하지 않고, 데이터 분석을 바탕으로 적절하다고 판단되는 모델과 함수들을 사용했기 때문에 차별성이 있다.

# **Ⅱ. 데이터 설명**

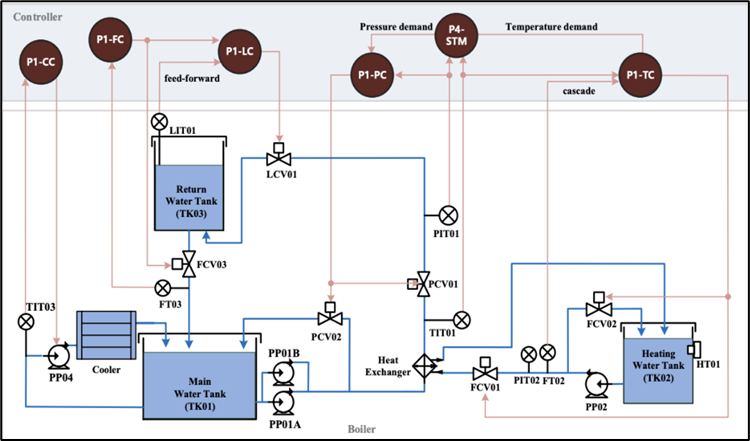
HAI 22.04는 정상 데이터만을 포함하고 있는 총 6개의 training data set과 비정상 데이터가 불균형하게 포함되어있는(약 3%) 총 4개의 test data set으로 구성되어 있다. 총 86개의 특성을 가지고 있고 데이터 조작을 편리성을 위하여 특성의 이름을 C01 ~ C86으로 치환하였다. Attack scenario는 총 37가지로 구성되어 총 58개의 공격이 있다. 이 중 단순 공격은 32개, 결합 공격(2개의 공격을 동시에 수행)은 26개를 차지한다[그림 1].



[그림 1] HAI 22.04 data set

HAI는 Boiler, Turbine, Water Treatment System, HIL(Hardware-in-the-loop) simulation을 기준으로 구성되어 있다. 모든 데이터는 각 Controller 속 set point에서 추출된 숫자 data로 구성된다. 이후에 소개하겠지만 딥러닝 모델의 데이터 전처리 과정에서 데이터를 그룹화를 진행할 때 아래 정보를 활용하였다[4].

## 1) P1: Boiler Controllers



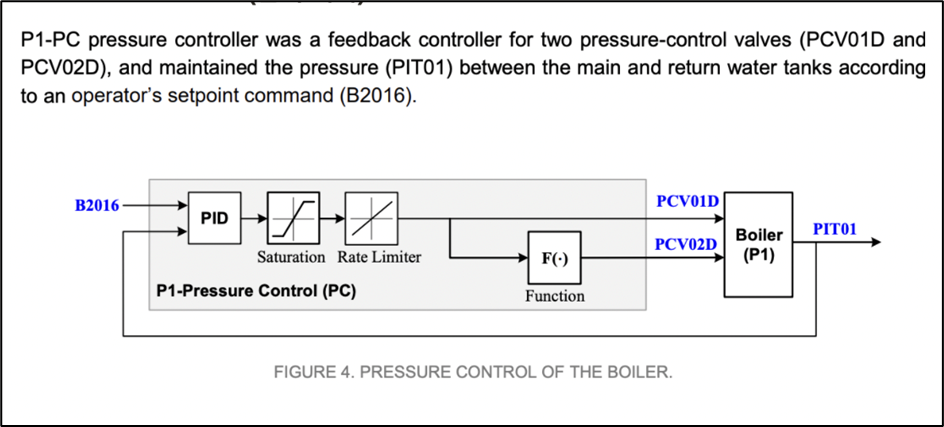
[그림 2] Boiler Controllers

Associate columns : C01 ~ C44, C79~C86(Associate P4-STM)

Boolean columns : C31~C36, C39~C41

ambiguous column : C01, C06, C15, C16, C30, C43

### 1-1) P1-PC(Pressure controller)



[그림 3] Pressure Controller

* P1\_B2016 (C02) : Pressure demand for thermal power output control
* P1\_PCV01D (C24) : Position command for the PCV01 valve
* P1\_PCV01Z (C25) : Current position of the PCV01 valve
* P1\_PCV02D (C26) : Position command for the PCV02 valve
* P1\_PCV02Z (C27) : Current position of the PCV02 valve
* P1\_PIT01 (C28) : Heat-exchanger outlet pressure
* P1\_PIT01\_HH (C29) : Highest outlet pressure of the heat-exchanger

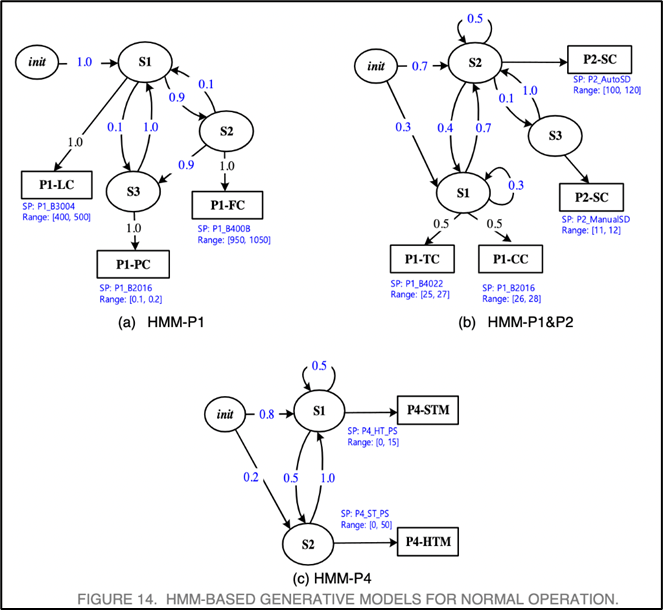
=> Has 6 attack scenarios AP01 ~ AP07(except AP06)

### 1-2) P1-FC(Flow rate controller)

### 

[그림 4] Flow rate controller

* P1\_B3005 (C04) : Discharge flowrate setpoint(return water tank)
* P1\_B400B (C07) : Water outflow rate setpoint



[그림 5] HMM-BASED GENERATIVE MODELS FOR NORMAL OPERATION

* P1\_FCV03D (C13) : Position command for the FCV03 valve
* P1\_FCV03Z (C14) : Current position of the FCV03 valve
* P1\_FT03 (C19) : Measured flowrate of the return water tank → P1-LC의 input으로 사용
* P1\_FT03Z (C20) : Water outflow rate conversion from P1\_FT03

=> Has 6 attack scenarios AP08 ~ AP13(6 attack scenarios)

### 1-3) P1-LC(Level controller)

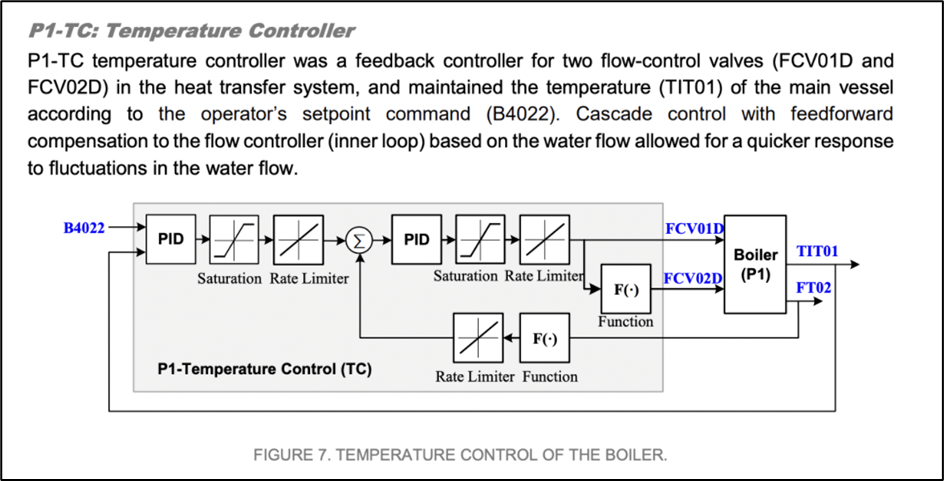
### 

[그림 6] Level controller

* P1\_B3004 (C03) : Water level setpoint(return water tank)
* P1\_FCV03D (C13) : Position command for the FCV03 valve → output of P1-FC
* P1\_FCV03Z (C14) : Current position of the FCV03 valve
* P1\_LCV01D (C21) : Position command for the LCV01 valve
* P1\_LCV01Z (C22) : Current Position of the LCV01 valve
* P1\_LIT01 (C23) : Water level of the return water tank

=> Has 5 attack scenario AP14 ~ AP18

### 1-4) P1-TC(Temperature controller)

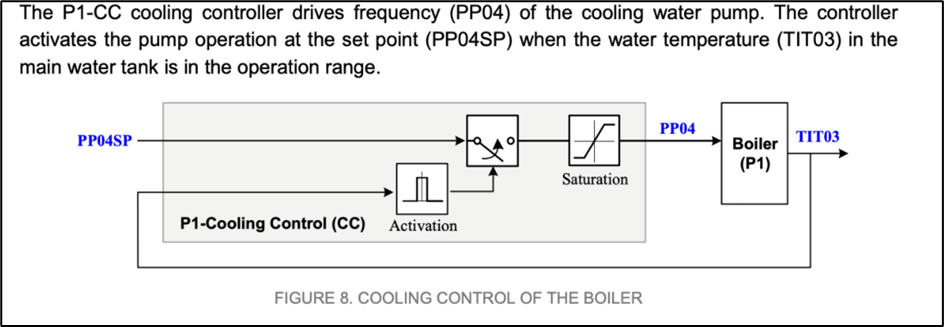


[그림 7] Temperature controller

* P1\_B4022 (C08) : Temperature demand for thermal power output control
* P1\_B4002 (C05) : Heat-exchanger outlet temperature setpoint(표시되어 있지는 않지만 attack에 포함)
* P1\_FCV01D (C09) : Position command for the FCV01 valve
* P1\_FCV01Z (C10) : Current position of the FCV01 valve
* P1\_FCV02D (C11) : Position command for the FCV02 valve
* P1\_FCV02Z (C12) : Current position of the FCV02 valve
* P1\_TIT01 (C42) : Heat-exchanger outlet temperature
* P1\_FT02 (C17) : Measured flow rate of heating water tank
* P1\_FT02Z (C18) : Water outflow rate conversion from P1\_FT02

=> Has 4 attack scenario AP19~AP22

### 1-5) P1-CC(Cooling controller)

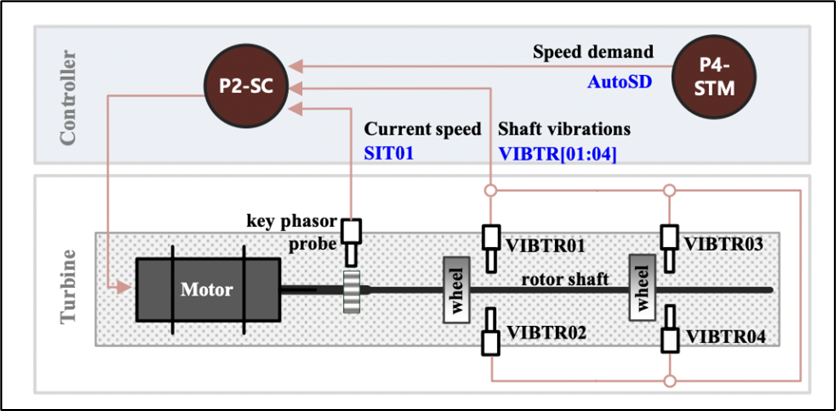


[그림 8] Cooling controller

* P1\_PP04SP (C38) : Cooler temperature setpoint
* P1\_PP04 (C37) : Control out the cooler pump
* P1\_TIT03 (C44) : Temperature of the main water tank

=> Has 3 attack scenario AP23~AP25

## 2) P2: Turbine Controllers

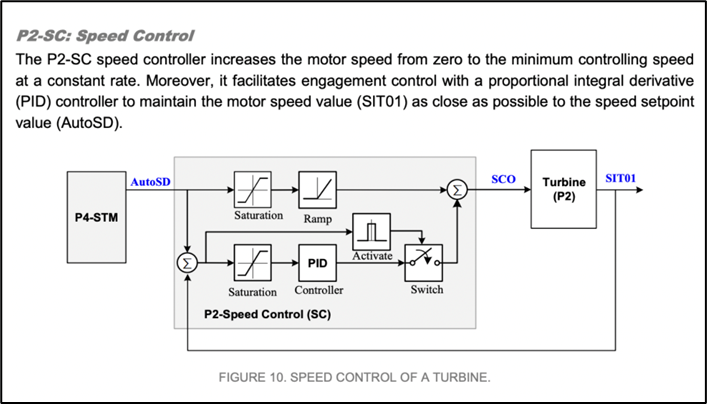


[그림 9] Turbine Controllers

Associate columns : C45~C68, C79~C86(Associate P4-STM)

Boolean columns : C46, C47, C49~C52, C54, C59

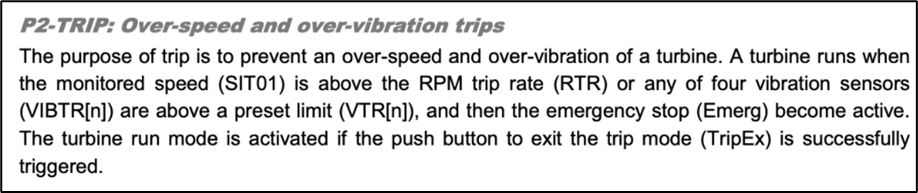
### 2-1) P2-SC(Speed Control)



[그림 10] Speed Control

* P2\_24Vdc : C45 : DCS 24V Input Voltage
* P2\_AutoSD : C48 : Auto speed demand
* P2\_ManualSD : C53 : Manual speed demand
* P2\_SCO : C56 : Control output value of the speed controller
* P2\_SCST (C57) : Speed change proportional to frequency change of the STM
* P2\_SIT01 (C58) : Current turbine RPM measured by speed probe

### 2-2) P2-TRIP(Over-speed and over-vibration trips)



[그림 11] Over-speed and over-vibration trips

* P2\_RTR (C55) : RPM trip rate
* P2\_VIBTR01 (C60) : Shaft-vibration-related Y-axis displacement near the 1st mass wheel
* P2\_VIBTR02 (C61) : Shaft-vibration-related X-axis displacement near the 1st mass wheel
* P2\_VIBTR03 (C62) : Shaft-vibration-related Y-axis displacement near the 2nd mass wheel
* P2\_VIBTR04 (C63) : Shaft-vibration-related X-axis displacement near the 2nd mass wheel
* P2\_VT01 (C64) : Phase lag signal of the key phasor probe
* P2\_VTR01 (C65) : Preset vibration limit for the sensor P2\_VIBTR01
* P2\_VTR02 (C66) : Preset vibration limit for the sensor P2\_VIBTR02
* P2\_VTR03 (C67) : Preset vibration limit for the sensor P2\_VIBTR03
* P2\_VTR04 (C68) : Preset vibration limit for the sensor P2\_VIBTR04

### 2-3) P2-SC와 P2-TRIP은 연관성이 커서 하나의 attack scenario로 취급

- Associate with SC

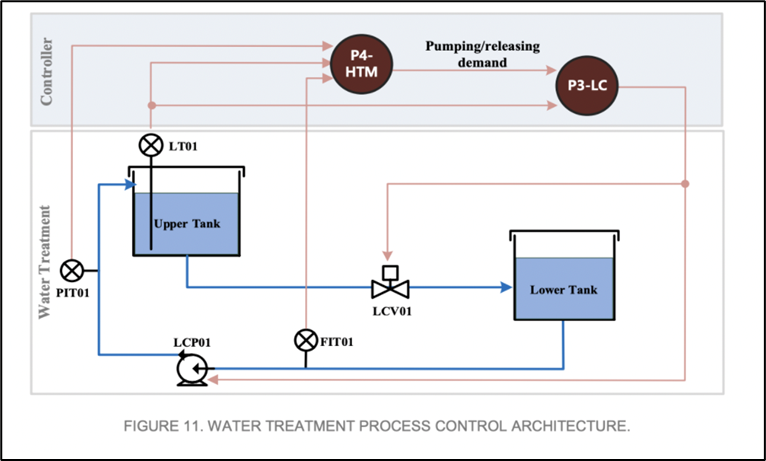
* P2\_24Vdc (C45) : DCS 24V Input Voltage
* P2\_AutoSD (C48) : Auto speed demand
* P2\_ManualSD (C53) : Manual speed demand
* P2\_SCO (C56) : Control output value of the speed controller
* P2\_SCST (C57) : Speed change proportional to frequency change of the STM
* P2\_SIT01 (C58) : Current turbine RPM measured by speed probe

- Associate with TRIP

* P2\_RTR (C55) : RPM trip rate
* P2\_VIBTR01 (C60) : Shaft-vibration-related Y-axis displacement near
* the 1st mass wheel
* P2\_VIBTR02 :(C61) : Shaft-vibration-related X-axis displacement near
* the 1st mass wheel
* P2\_VIBTR03 (C62) : Shaft-vibration-related Y-axis displacement near
* the 2nd mass wheel
* P2\_VIBTR04 (C63) : Shaft-vibration-related X-axis displacement near
* the 2nd mass wheel
* P2\_VT01 (C64) : Phase lag signal of the key phasor probe
* P2\_VTR01 (C65) : Preset vibration limit for the sensor P2\_VIBTR01
* P2\_VTR02 (C66) : Preset vibration limit for the sensor P2\_VIBTR02
* P2\_VTR03 (C67) : Preset vibration limit for the sensor P2\_VIBTR03
* P2\_VTR04 (C68) : Preset vibration limit for the sensor P2\_VIBTR04
* C79~C86 (Associate P4-STM)

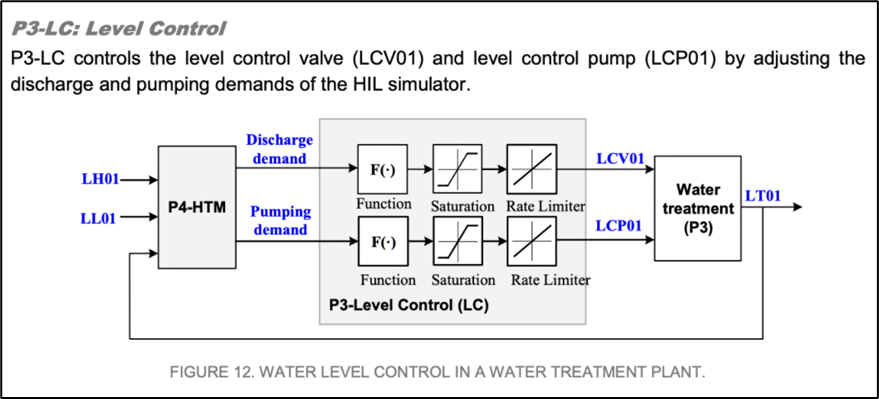
=> Has 8 attack scenario AP26 ~ AP34(except AP32)

## 3) P3 : Water Treatment Controllers



[그림 12] Water Treatment Controllers

### 3-1) P3-LC : Level Control



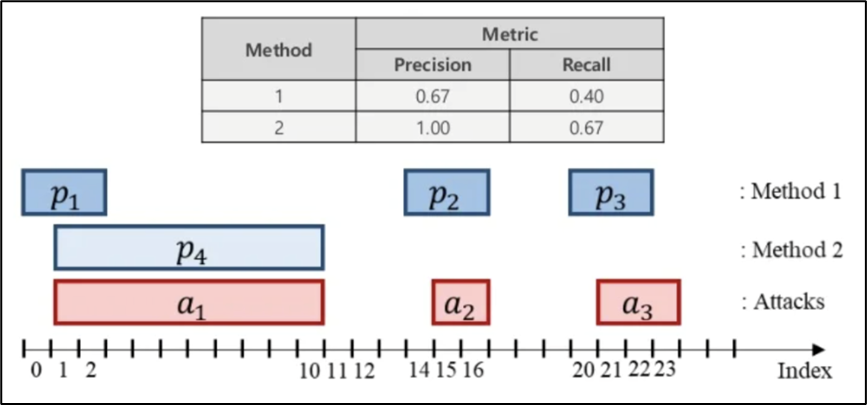
[그림 13] Level control

* P3\_LH01 (C72) : High water level set-point
* P3\_LL01 (C74) : Low water level set-point
* P3\_LCV01D (C71) : Position command for the valve LCV01
* P3\_LCP01D (C70) : Speed command for pump LCP01
* P3\_FIT01 (C69) : Flow rate of water flowing into the upper water tank
* P3\_LIT01 (C73) : Water level of the upper water tank
* C76~C79(Associate with P4-HTM)
* ambiguous columns : C75

=> Has 5 attack scenario AP35 ~ AP39

# **Ⅲ. 평가 방식**

모델 평가 방식은 eTaPR(Enhanced Time-series Aware Precision and Recall)을 사용한다. 기존의 TaPR보다 다양한 탐지 시나리오를 고려할 수 있는 평가 방식이다. 일반적인 Precision, Recall을 사용하지 않는 이유는 아래의 예시로 설명이 가능하다.



[그림 14] 시계열 데이터 이상탐지 F1 score 성능 평가

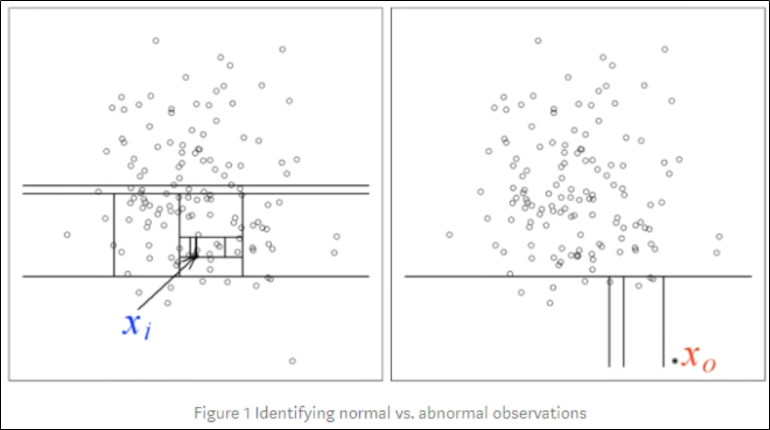
[그림 14]를 보면 method1(a1, a2, a3 탐지)가 method 2(a1만을 탐지)보다 더 좋은 예측을 하였다고 판단할 수 있지만, method 2가 Precision, Recall값이 더 높게 측정된다. 그렇기 때문에 eTaP(Enhanced Time-series aware Precision)와 eTaR(Enhanced Time-series aware Recall)을 사용하였다. eTaP는 탐지 알고리즘의 모든 예측이 평균적으로 옳을 가능성을 의미하는 지표이고 eTaR은 평가 데이터의 모든 anomaly가 평균적으로 탐지될 가능성을 의미하는 지표이다. 결론적으로 eTaP는 이상구간의 탐지 정확성을 나타내고 eTaR은 공격패턴의 탐지 다양성을 나타낸다. 우리는 eTaP, eTaR의 조화평균 값인 eTaPR을 사용하여 성능평가를 진행하였다[5-6]. 이후 편의를 위해 e를 생략하고 TaP, TaR, TaPR로 표현한다.

# **Ⅵ. 모델**

모델에 데이터를 학습시키기 전에 공통적으로 데이터 전처리를 해주었다. 우선 여러 파일로 분리되어 있는 Data Set을 하나의 Train과 Test Data set으로 합쳤다. 이후 Column별로 Data의 scale의 차이가 있어 모델의 학습이 정상적으로 이루어지지 않을 것 이라고 판단되어 정규화를 진행하였다. Z-score 정규화는 데이터가 정규분포를 이룰 때 사용하는 것이 유용하지만, HAI 데이터는 데이터가 정규분포를 이루지 않고, 데이터의 분포에 대하여 알 수 없으므로 Min-Max 정규화를 사용하였다. Test Data set의 정규화는 Train Data set의 정규화 기준과 같게 사용하여 정규화를 적용하였다[7].

## 1. Isolation Forest

Isolation Forest는 Decision Tree를 이용한 이상탐지 비지도 학습 알고리즘으로, 재귀 이진분할이기 때문에 tree의 깊이가 짧을수록 비정상 데이터라는 아이디어를 사용하여 outlier를 탐지한다[그림 15]. Isolation Forest로 이상탐지를 하는 선례들도 존재하고[8-9], train 데이터에 이상치가 포함되지 않아도 score 판정 자체가 tree의 깊이로 판단하기 때문에 iForest 성능에는 학습 데이터에 이상치 존재 여부가 중요하지 않다는 특징도 있어서 이 모델을 사용하게 되었다[10].



[그림 15] Isolation Forest 원리

Isolation Forest는 sklearn 제공하고 있으며 여러 parameter가 존재하지만, 실험 결과 contamination parameter를 제외하고는 결과에 큰 영향을 미치지 않아 나머지 parameter들은 모두 default 값으로 설정한다. 성능 비교를 위해 random\_state 값만 42로 고정했다. contamination은 data set에서의 outlier 비율인데, 실제 데이터 분석을 기반한 수치를 사용하였을 때 결과가 좋지 않아 이 값을 튜닝해가는 방향으로 프로젝트를 진행했다. 또한 데이터 전처리 이후에, 모델의 성능이 좋지 않아 여러 방법으로 학습을 시켜보았고 그 과정을 순서대로 기록하였다.

### 1) contamination 값 튜닝

먼저 모든 특성을 가지고 contamination 값을 튜닝한 결과이다[표 1]. 여러 값 중에서 ‘0.059’와 ‘0.06’일 때 비교적 좋은 성능을 내는 것으로 판단되어 두 값을 중심으로 모델 학습을 진행하였다.

[표 1] 86개 특성을 가지고 contamination 조정

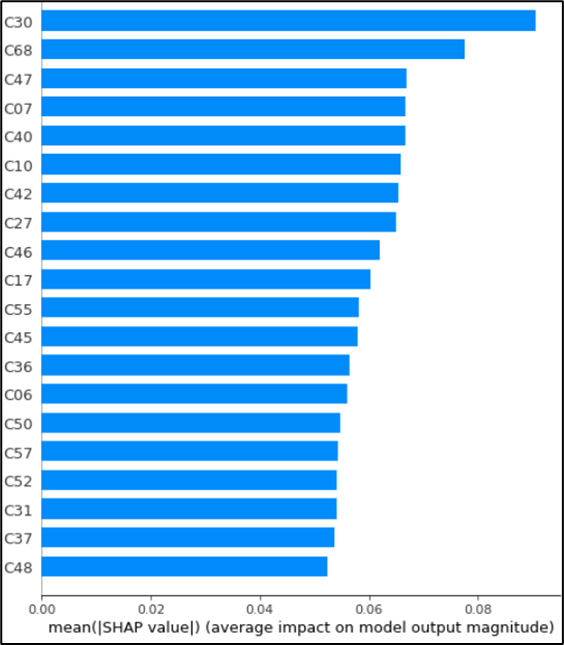
| **contamination** | **TaP** | **TaR** | **TaPR** | **Detected anomalies** |
| --- | --- | --- | --- | --- |
| 0.5 | 0.012 | 0.255 | 0.024 | 13 |
| 0.1 | 0.015 | 0.303 | 0.028 | 27 |
| 0.05 | 0.016 | 0.128 | 0.028 | 10 |
| 0.04 | 0.016 | 0.110 | 0.028 | 9 |
| 0.055 | 0.016 | 0.146 | 0.028 | 12 |
| 0.049 | 0.015 | 0.125 | 0.026 | 10 |
| 0.059 | 0.015 | 0.166 | 0.028 | 15 |
| 0.06 | 0.016 | 0.171 | 0.029 | 16 |
| 0.058 | 0.015 | 0.162 | 0.028 | 15 |

특성이 너무 많아서 데이터 중에서 항상 일정한 값을 가지는 특성(C29, C31, C32, C33, C34, C35, C36, C39, C40, C41, C49, C54, C55, C59, 이후 상수 특성으리고 칭함)은 불필요하다고 판단되어 삭제하고 돌려보았지만 성능이 좋지 않았다[표 2].

[표 2] 항상 일정한 값을 가지는 특성 제외

| **contamination** | **TaP** | **TaR** | **TaPR** | **Detected anomalies** |
| --- | --- | --- | --- | --- |
| 0.06 | 0.015 | 0.278 | 0.028 | 20 |
| 0.059 | 0.015 | 0.276 | 0.028 | 20 |

그래서 상수 특성을 삭제한 이후에, 특성들 사이의 상관관계를 분석하여 특성별로 그룹을 만들었다. 각 그룹에서는 특성 중요도를 고려하여[그림 16] 비교적 중요도가 낮은 특성들을(C8, C9, C16, C18, C19, C20, C22, C23, C25, C39, C41, C43, C49, C53, C62, C63, C65, C66) 삭제하였을 때 성능이 조금 오르긴 했지만 그래도 성능이 좋지 않았다[표 3].



[그림 16] 특성 중요도

[표 3] 상관관계 분석 + 특성 중요도

| **contamination** | **TaP** | **TaR** | **TaPR** | **Detected anomalies** |
| --- | --- | --- | --- | --- |
| 0.06 | 0.017 | 0.224 | 0.032 | 19 |
| 0.059 | 0.018 | 0.221 | 0.033 | 19 |

### 2) PCA 적용

Isolation Forest 모델은 차원이 높을수록 성능이 좋지 않다는 단점을 가지고 있기 때문에 성능이 좋지 않은 것이라고 판단되어서 PCA를 도입하여 세 가지 케이스에서 파라미터 값을 조정하여 성능을 비교해본 결과, PCA를 도입하기 전에 비해 성능이 훨씬 향상된 것을 확인할 수 있었다.

먼저 모든 특성을 유지한채로 PCA에서 n\_components(축소할 차원 수) 값을 조정해본 결과 50에서 가장 좋은 성능이 나오고, 55에서 가장 많은 공격을 탐지하는 것을 확인할 수 있었다. 그래서 n\_components 값을 50과 55를 중심으로 다른 케이스에서도 성능 확인을 진행하였다[표 4].

[표 4] 모든 컬럼(86)에 PCA 적용

| **n\_components** | **contamination** | **TaP** | **TaR** | **TaPR** | **Detected anomalies** |
| --- | --- | --- | --- | --- | --- |
| 10 | 0.059 | 0.065 | 0.098 | 0.078 | 9 |
| 0.06 | 0.063 | 0.099 | 0.079 | 9 |
| 50 | 0.059 | 0.078 | 0.394 | 0.130 | 30 |
| 0.06 | 0.077 | 0.395 | 0.129 | 30 |
| 54 | 0.059 | 0.063 | 0.360 | 0.108 | 28 |
| 0.06 | 0.063 | 0.361 | 0.106 | 28 |
| 55 | 0.059 | 0.061 | 0.498 | 0.108 | 42 |
| 0.06 | 0.060 | 0.500 | 0.106 | 42 |
| 56 | 0.059 | 0.077 | 0.400 | 0.129 | 31 |
| 0.06 | 0.076 | 0.402 | 0.127 | 31 |
| 57 | 0.059 | 0.064 | 0.460 | 0.112 | 37 |
| 0.06 | 0.063 | 0.462 | 0.111 | 37 |
| 60 | 0.059 | 0.045 | 0.480 | 0.082 | 37 |
| 0.06 | 0.045 | 0.482 | 0.081 | 37 |

상수 특성을 삭제하고 PCA를 적용했을 때, 55 차원으로 축소한 경우 성능이 향상되고 더 많은 공격을 탐지한 것을 확인할 수 있었다[표 5].

[표 5] 상수 특성 삭제 후 PCA

| **n\_components** | **contamination** | **TaP** | **TaR** | **TaPR** | **Detected anomalies** |
| --- | --- | --- | --- | --- | --- |
| 50 | 0.059 | 0.078 | 0.394 | 0.130 | 30 |
| 0.06 | 0.077 | 0.395 | 0.129 | 30 |
| 55 | 0.059 | 0.063 | 0.516 | 0.113 | 43 |
| 0.06 | 0.063 | 0.519 | 0.112 | 43 |

상수 특성을 삭제한 이후에 상관관계와 특성 중요도를 고려하여 추가 특성을 삭제하고 PCA를 적용했을 때, 성능이 비교적 떨어진 것을 확인할 수 있었다. 추가로 특성을 삭제한 결과 총 특성 수가 54개여서 50 차원으로 줄인 경우만 확인해봤는데 성능이 좋지 않아서 더 이상 확인하진 않았다.

[표 6] 상관관계 삭제 후 PCA

| **n\_components** | **contamination** | **TaP** | **TaR** | **TaPR** | **Detected anomalies** |
| --- | --- | --- | --- | --- | --- |
| 50 | 0.059 | 0.018 | 0.221 | 0.033 | 19 |
| 0.06 | 0.017 | 0.224 | 0.032 | 19 |

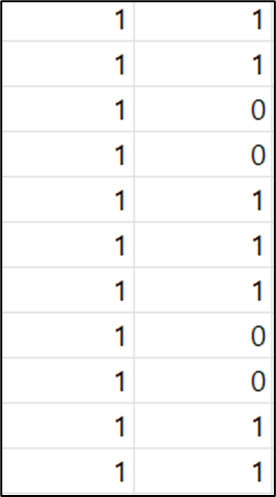
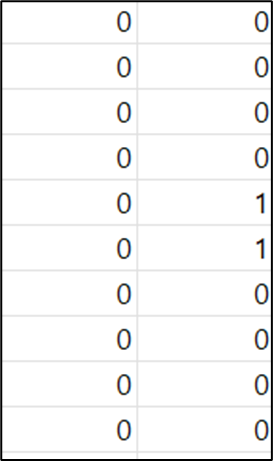
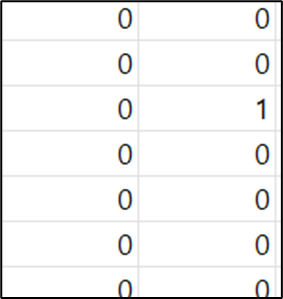
데이터를 추가분석 해본 결과, 삭제된 상수 특성 중에서 공격의 포인트가 되는 특성이 하나 있어서 이를 추가로 보존하고 확인해본 결과 결과가 거의 유사하였다. 이는 모델의 특성 때문에 영향이 크게 없었던 것으로 추측된다.

[표 7] 상수 특성 하나 추가 후 PCA

| **n\_components** | **contamination** | **TaP** | **TaR** | **TaPR** | **Detected anomalies** |
| --- | --- | --- | --- | --- | --- |
| 50 | 0.059 | 0.078 | 0.394 | 0.130 | 30 |
| 0.06 | 0.077 | 0.395 | 0.129 | 30 |
| 55 | 0.059 | 0.063 | 0.516 | 0.113 | 43 |
| 0.06 | 0.063 | 0.519 | 0.112 | 43 |

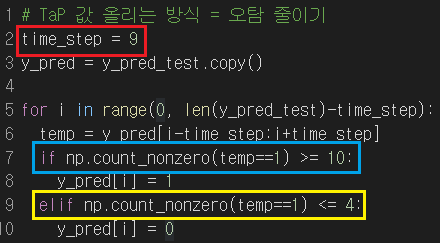
### 3) 오탐 줄이기

[그림 17]은 csv 파일로 확인한 실제 정답과 예측 값의 일부이다. 왼쪽이 실제 정답이고, 오른쪽이 예측한 값이다. 0이 정상을 의미하고 1이 이상치를 의미한다. 왼쪽 두 그림은 공격이 발생하지 않았는데 공격으로 탐지한 일부 예시를 가져온 것이고, 맨 오른쪽의 그림은 공격은 잘 탐지했지만 모든 공격 구간을 정확하게 탐지하지 못한 예시이다. Isolation Forest를 사용한 결과 이러한 케이스가 되게 많아서 오탐 수치가 매우 높고, 이로 인해 성능이 떨어지게 된다고 판단했다. 그래서 위의 언급된 예시들을 핸들링할 수 있는 코드를 추가하여 오탐 수치를 낮춰보았다.



[그림 17] 결과 속 오탐 예시

우리가 생각해낸 방법은 다음과 같다. 한 시점을 기준으로 앞 뒤 time\_step 영역만큼을 확인하여, 해당 영역에 1이 일정 개수 이하로 있으면 첫번째 예시 케이스라고 판단하여 해당 시점의 값을 0으로 변경해주고, 해당 영역에 1이 일정 개수 이상 있으면 두번째 예시 케이스라고 판단하고 해당 시점의 값을 1로 변경해주었다[그림 18]. 그림에서 표시된 부분들의 값을 바꿔가면서 성능을 테스트 해보았고, 이를 정리하면 [표 8]와 같다. 해당 방법을 사용한 결과 오탐이 줄어들고 비교적 성능이 향상됨을 확인할 수 있었다.



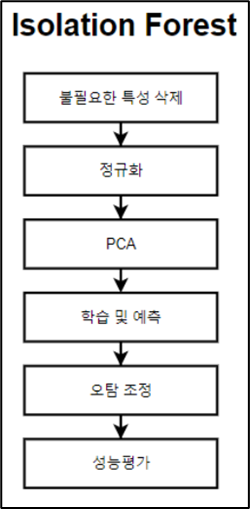
[그림 18] 오탐 조정 코드

[표 8] 상수 특성을 삭제하고 PCA를 삭제한 이후 오탐 조정

| **확인 범위** | **1로갱신** | **0으로 갱신** | **contamination** | **TaP** | **TaR** | **TaPR** | **Detected anomalies** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 8 | 3 | 0.059 | 0.226 | 0.545 | 0.319 | 43 |
| 0.06 | 0.229 | 0.548 | 0.323 | 43 |
| 10 | 11 | 4 | 0.059 | 0.242 | 0.550 | 0.336 | 43 |
| 0.06 | 0.238 | 0.551 | 0.332 | 43 |
| 10 | 11 | 5 | 0.059 | 0.266 | 0.535 | 0.355 | 41 |
| 0.06 | 0.263 | 0.536 | 0.353 | 41 |
| 9 | 10 | 4 | 0.059 | 0.253 | 0.549 | 0.347 | 43 |
| 0.06 | 0.247 | 0.549 | 0.341 | 43 |
| 12 | 13 | 6 | 0.059 | 0.277 | 0.531 | 0.364 | 41 |
| 0.06 | 0.236 | 0.500 | 0.321 | 38 |

### 4) 최종 구조도

Isolation Forest 모델을 사용하면서 다양한 시도를 해보았지만, 가장 좋은 성능이 나왔던 과정의 흐름은 [그림 19]와 같다. 최종적으로 TaPR 점수는 최고점으로 0.364를 얻었고, 가장 많이 탐지한 공격의 개수는 58개 중 43개이다. 최고 성능이 30퍼대인 이유는 모델의 특성상 한계라고 추측된다.



[그림 19] Isolation Forest 최종 프로세스

## 2. Bidirectional LSTM Autoencoder, Bidirectional GRU Autoencoder

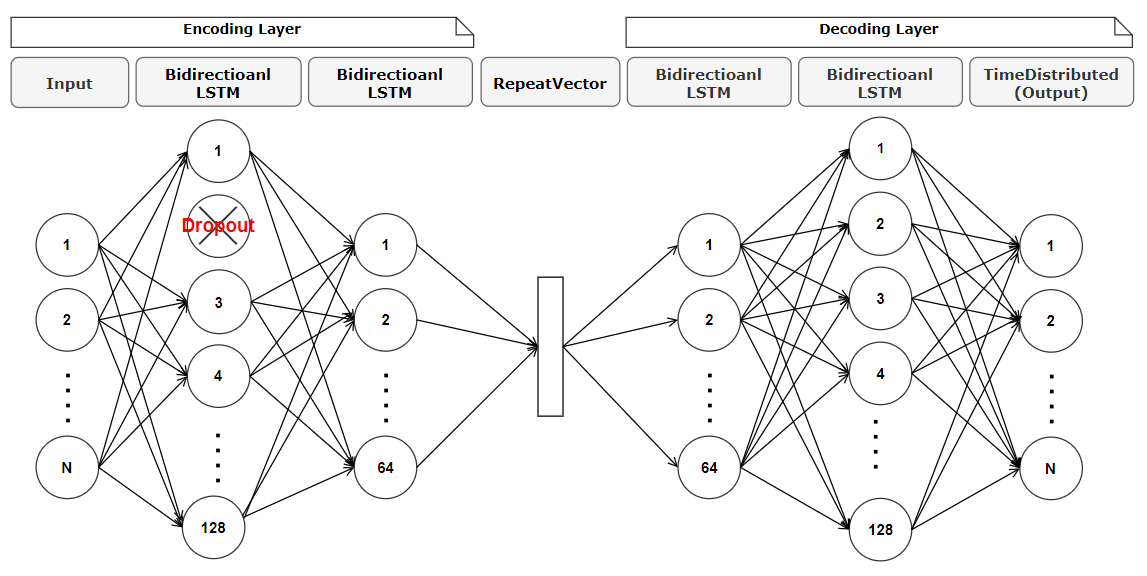
두 모델의 공통적인 부분은 LSTM 모델에서 설명하였다.

### 1) 데이터 전처리

시계열 데이터를 사용할 때 유용한 LSTM과 GRU 모델을 사용하기 위해 ‘reshape’ 함수를 통하여 Train, Test Data set의 구조를 (input\_size, timesteps, input\_dimension)의 3차원 형태로 변환하여 주었다[12].

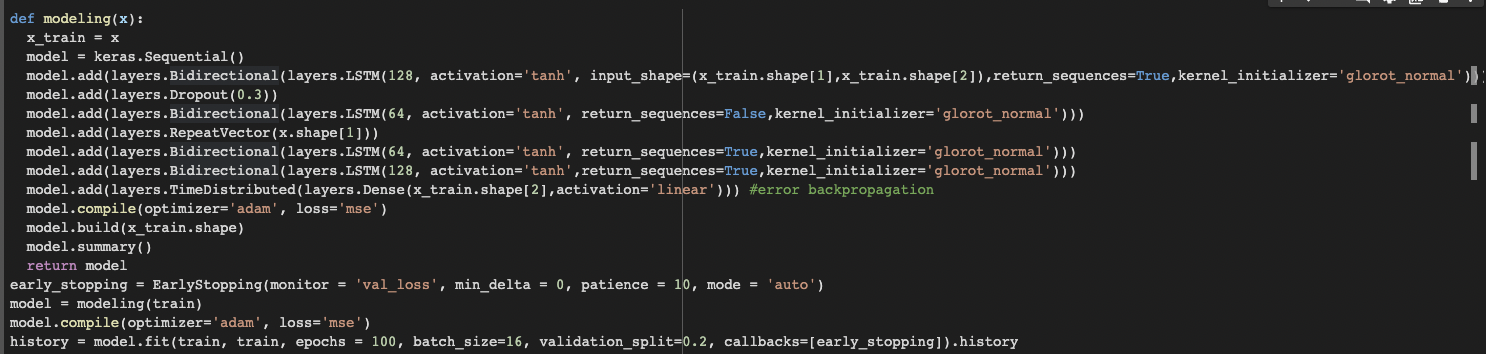
### 2) Bidirectional LSTM Autoencoder

Bidirectional LSTM + AutoEncoder는 AutoEncoder의 Layer에 LSTM을 사용한 모델로, 시계열 데이터의 이상탐지에 자주 사용되는 모델이다. 프로젝트에서 활용한 모델의 전체 구조도는 [그림 20]과 같다. HAI는 86개의 특성을 가진 고차원 데이터이고 이상 데이터 비율이 매우 낮아 데이터 불균형 문제가 심각하였는데, 이를 모두 해결할 방법으로 AutoEncoder를 선택하게 되었다.



[그림 20] Bidirectional LSTM Autoencoder 구조

Encoding Layer는 입력받은 데이터를 압축하는 역할을 한다. Encoding Layer에 Dropout Layer를 추가하여 Overfitting의 가능성을 줄이고 일반성을 높였다. 다음 Layer에는 RepeatVector Layer를 사용하여 다음 Bidirectional LSTM에 넣어줄 데이터의 형식을 맞춰주었다. Decoding Layer로 압축된 데이터를 복원하는 역할을 한다. 모델에 대한 Python code는 [그림 21]과 같다.

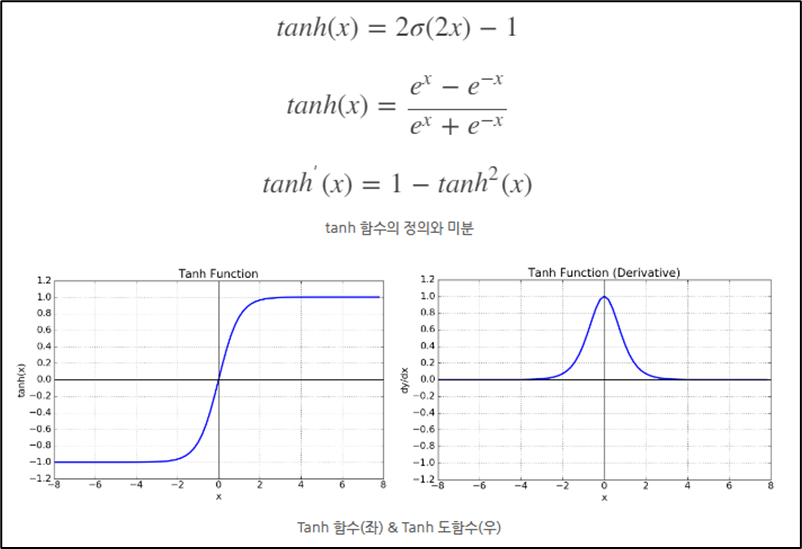


[그림 21] Bidirectional LSTM Autoencoder 구조 python 코드

손실된 데이터로 완전한 데이터를 예측할 때 시간적으로 앞의 데이터 만으로 올바른 데이터를 예측하는 것이 어려울 수 있다. 이 때 양방향 재귀 모델을 사용하면 정방향 뿐만 아니라 역방향의 전파까지도 실행시킨다[13]. 그래서 time\_step의 앞, 뒤 시간을 모두 고려하기 위하여 Bidirectional 모델을 사용하였다. 일부러 데이터를 손상시켜(Dropout Layer) 정상 데이터로 복원시키는 것을 학습시키는 방법을 사용하기 때문에 정확히 복원시키는 것이 중요하므로 ‘Bidirectional’ Layer를 추가하였다.

Activation function(활성화 함수)으로 ‘Hyperbolic Tangent function’ 함수를 사용하였다. ‘tanh’ 함수는 -1에서 1사이의 값을 갖는 S자 곡선으로, 중앙값이 0이기 때문에 시그모이드 함수에서 발생하는 편향 이동이 발생하지 않는다.

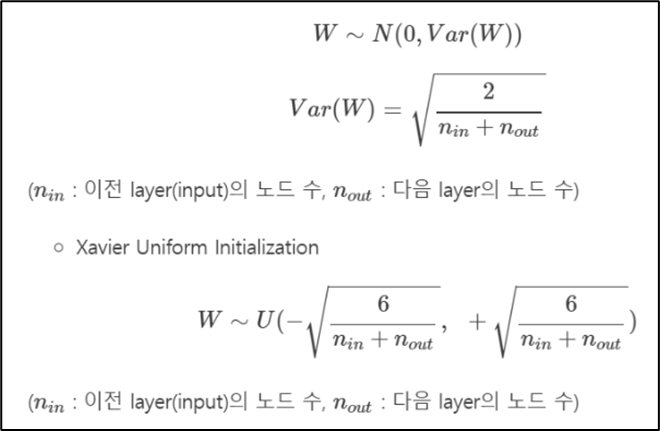
‘tanh’ 함수는 기울기 소실 현상이 발생하지만, 시계열 데이터를 주로 사용하는 순환 신경망 모델에서는 적합한 함수다. 순환 신경망 모델은 이전 step의 값을 가져와서 사용하는 모델이기 때문에 ‘ReLU’와 같은 활성화 함수를 사용하게 되면 이전 값이 커짐에 따라 값이 1보다 크게되었을 때 출력이 발산하는 문제가 생길 수 있기 때문에[14] 순환 신경망 같은 내부 순환하는 구조는 ‘tanh’ 함수를 사용하는 것이 적절하다고 판단되어 ‘tanh’를 사용하였다.



[그림 22] Hyperbolic Tangent function

Kernel initialization(가중치 초기화) 방법으로는 ‘Xavier Glorot’ 초기화 방법을 사용하였다. Xavier는 초기화시키는 파라미터가 존재하는 Layer들의 입력/출력 노드 개수에 따라 동적으로 파라미터 값을 초기화시키는 방법이다. Normal과 Uniform 방법으로 나눠지는데, LSTM모델에서 사용한 방법은 Normal 방법이고, GRU모델에서 사용한 방법은 Uniform 방법이다. Xavier Normal 방법은 초기화 시킬 파라미터 값들의 범위에서 평균은 0으로 유지하되 분산값을 같이 조정하는 방법이고, Xavier Uniform 방법은 파라미터 값들의 범위를 하한값, 상한값을 지정해서 그 범위 안에서 파라미터를 초기화시키는 방법이다[15].

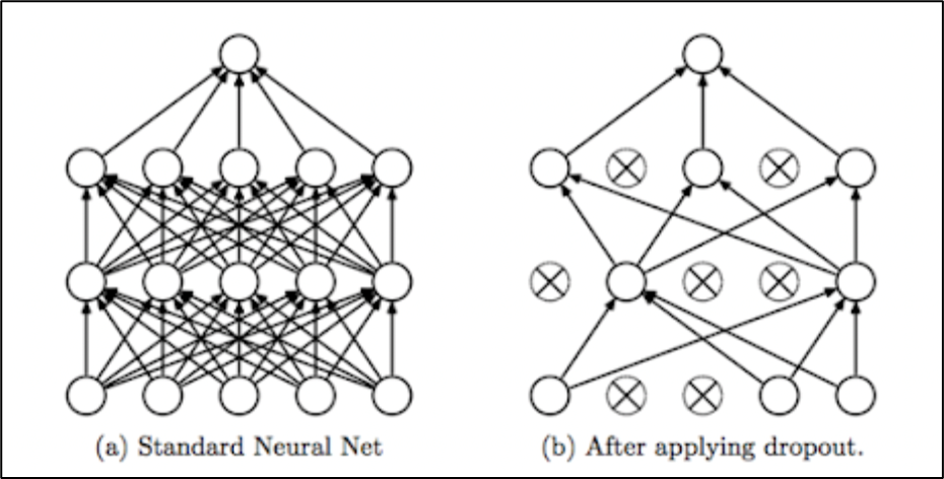
Xavier 초기값은 활성화 함수가 선형인 것을 전제로 이끌어낸 경우이며, ‘tanh’ 함수가 좌우 대칭이라 중앙 부근이 선형인 함수라고 볼 수 있기 때문에[16] ‘tanh’ 함수를 활성화 함수로 사용할 경우 ‘Xavier Glorot’ 초기화 방법을 사용하는 것이 적합하다고 판단되어 사용하였다.



[그림 23] Xavier Glorot

Dropout은 신경망의 노드 중 일부만 동작하고 일부는 동작하지 않도록 하는 방법이다. 딥러닝에서 파라미터가 많다는 것은 모델의 복잡도가 높다는 것을 의미하고, 이는 overfitting이 발생할 수 있다는 것을 의미한다[17]. Keras의 “Layers.Dropout” 함수를 사용하여 LSTM-Autoencoder에서는 노드의 삭제 확률을 0.3으로 지정하고, GRU-Autoencoder에서는 노드의 삭제확률을 0.4으로 지정하여 모델을 구성하였다. 다른 Dropout의 종류로는 AlphaDropout, GaussianDropout이 존재한다.

활성화 함수로 ‘tanh’를 사용하기 전에 딥러닝에서 많이 사용되는 relu의 변형인 elu function을 사용하기 위해 이에 더 효과적인 성능을 보이는 AlphaDropout을 사용하였으나 활성화 함수에서 설명한 출력값의 발산 문제로 최종적인 모델에서 Tangent hyperbolic function을 사용하게 되어 마지막에는 일반적인 Dropout을 사용하였다.

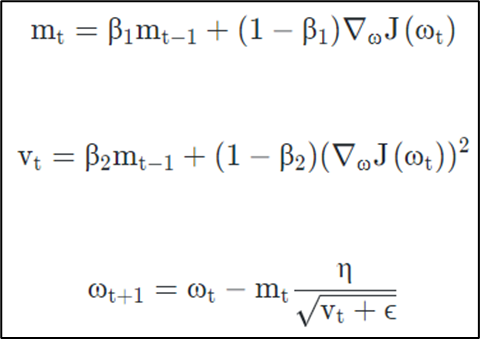


[그림 24] dropout 적용 차이

마지막 layer에 ‘TimeDistributed layer’를 사용하였다. ‘TimeDistributed layer’는 layer의 각 timestep에서 계산된 cost를 해당 지점에서 오류를 전파시킨다[18].

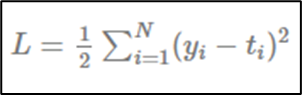
Optimizer function(최적화 함수)으로는 ‘Adam optimizer’를 사용하였다. RMSProp과 Momentum 기법을 합친 최적화 방법으로, 관성계수 m과 함께 계산된 V로 parameter를 업데이트하는 Momentum과 다르게 Adam에서는 기울기 값과 기울기의 제곱값의 지수이동평균을 활용하여 step 변화량을 조절한다[19].

데이터 및 모델 파라미터가 많이 필요한 문제에 적합하며, 간단한 구현으로 효율적인 연산을 해 메모리 요구 사항이 거의 없기 때문에 ‘Adam optimizer’를 사용하였다.



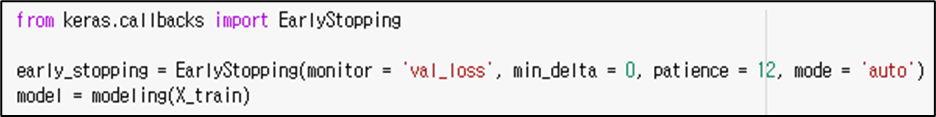
[그림 25] Adam optimizer

Loss function(손실 함수)로는 평균 제곱 오차(MSE)를 사용하였다. 원래 데이터를 복원시키는 회귀 용도로 LSTM과 GRU 모델을 사용하였기 때문에[20] 회귀 용도의 딥러닝 모델을 훈련시킬 때 많이 사용되는 손실 함수인 MSE가 적합하다고 판단되어 MSE를 사용하였다.



[그림 26] MSE

EarlyStopping은 무조건 Epoch을 많이 돌리는 것이 아닌 특정 시점에서 멈추도록 만들어 준다. mode를 auto로 설정해 Keras가 자동으로 min, max를 선택하게 한 후 Validation set의 loss를 monitoring하도록 설정하였고, 성능이 증가하지 않은 epoch을 몇 번이나 허용할 것인가를 patience를 통해 정의하였다[21]. Bidirectional LSTM Autoencoder는 patience = 10으로 고정을 시키고 학습을 진행하였고, Bidirectionall GRU Autoencoder patience = 12로 고정을 시키고 학습을 진행하였다.



[그림 27] EarlyStopping

데이터가 방대한 양이라면 전체 데이터에 대해서 학습을 수행할 경우 속도가 매우 느릴 뿐만 아니라 많은 계산량이 필요하다. 만약 전체 데이터를 더 작은 단위로 나눠 해당 단위로 학습한다면 학습률을 높일 수 있는데 이를 Mini batch라 한다[22]. 보통 Mini batch size는 2의 제곱수를 사용하기 때문에 Bidirectional GRU Autoencoder에서는 32, Bidirectional LSTM Autoencoder에서는 16으로 정해 진행하였다. epoch은 최대 100 epoch를 잡았지만 Early Stopping으로 일정 epoch이 지나면 종료된다.



[그림 28] model fit with mini-batch

Moving average는 이동평균이라는 의미로 특정 간격 안에서의 데이터의 평균을 의미한다. Autoencoder로 복원한 Test 데이터의 MSE값을 구하고, Pandas의 “rolling().mean” 함수를 사용하여 이동평균 값을 구해주었다. 이 때 다양한 공격 패턴(TaR)을 찾아내되 정확도(TaP)가 높은 방향으로 최대의 TaPR값이 나오도록 threshold값을 각 그룹별로 수정하였다. 그 후 이동평균 값을 사용하여 구한 복원 오차가 앞서 구한 threshold값 이상이라면 공격이라고 판정하였다.

시계열 데이터의 시간의 특성을 잘 반영하기 위해 MSE로 구한 복원오차를 바로 이상탐지 기준으로 삼지 않고 Moving average로 전처리하는 error smoothing 작업을 통해 시간의 특성을 잘 반영할 수 있게 하며, 공격구간을 잘 탐지할 수 있게 하였다.



[그림 29] error smoothing

전체 데이터를 연관있는 특성끼리 그룹을 나눠서 학습을 시켰다. 특성 사이의 관계가 복잡하여 명확히 그룹을 나눌 수 없어 Group1(6개로 구성), Group2(4개로 구성), Group3(5개로 구성) Group4(5개로 구성), Group5(5개로 구성) 총 5개로 나누어 학습시키고 성능을 비교하였다. 각각의 그룹의 데이터들을 사용하여 개별적으로 모델을 학습시킨 후 각 그룹에서 모델의 예측결과 중 좋은 결과가 나온 모델의 예측결과를 합치는 방법으로 예측을 수행하였다. Boolean column들은 데이터를 복원하는데에 악영향을 끼칠 것이라고 판단되어 모두 제거 후 그룹핑을 하였다.

각 그룹에 대한 구성은 [표 9-13]과 같고, 그룹별 취합 결과는 [표 14]와 같다. 전반적으로 다른 단일 모델들에 비해 성능이 높으며, 특히 Group1이 TaPR 0.605로 가장 성능이 우수하게 나온 것을 확인할 수 있다.

[표 9] Group 1

| **Group** | **Controller** | **Columns** |
| --- | --- | --- |
| Group1-1 | P1-PC/P4-STM | ‘C02’, ‘C24’, ‘C25’, ‘C26’, ‘C27’, ‘C28’, ‘C29’, ‘C79’, ‘C80’, ‘C81’, ‘C82’, ‘C83’, ‘C84’, ‘C85’, ‘C86’ |
| Group1-2 | P1-FC/P1-LC/P4-STM | ‘C04’, ‘C07’, ‘C13’, ‘C14’, ‘C19’, ‘C20’, ‘C23’ ‘C03’, ‘C21’, ‘C22’, ‘C79’, ‘C80’, ‘C81’, ‘C82’, ‘C83’, ‘C84’, ‘C85’, ‘C86’ |
| Group1-3 | P1-CC | ‘C37’, ‘C38’, ‘C44’ |
| Group1-4 | P1-TC/P4-STM | ‘C05’, ‘C08’, ‘C09’, ‘C10’, ‘C11’, ‘C12’, ‘C17’, ‘C18’, ‘C42’ |
| Group1-5 | P2-SC/P2-TRIP | ‘C45’, ‘C48’, ‘C53’, ‘C56’, ‘C57’, ‘C58’, ‘C55’, ‘C60’, ‘C61’, ‘C62’, ‘C63’, ‘C64’, ‘C65’, ‘C66’, ‘C67’, ‘C68 |
| Group1-6 | P3-LC/P4-HTM | ‘C69’, ‘C70’, ‘C71’, ‘C72’, ‘C73’, ‘C74’ ‘C75’, ‘C76’, ‘C77’, ‘C78’, ‘C79’ |

[표 10] Group 2

| **Group** | **Controller** | **Columns** |
| --- | --- | --- |
| Group2-1 | P1-PC/P1-TC/P4-STM | “C02”, “C24”, “C25”, “C26”, “C27”, “C28”, “C29, “C05”, “C08”, “C09”, “C10”, “C11”, “C12”, “C17”, “C18”, “C42”, “C79”, “C80”, “C81”, “C82”, “C83”, “C84”, “C85”, “C86” |
| Group2-2 | P1-FC/P1-LC/P1-CC/P4-STM | “C04”, “C07”, “C13”, “C14”, “C19”, “C20”, “C23” “C03”, “C21”, “C22”, “C37”, “C38”, “C44”, “C79”, “C80”, “C81”, “C82”, “C83”, “C84”, “C85”, “C86” |
| Group2-3 | P2-SC/P2-TRIP | ‘C45’, ‘C48’, ‘C53’, ‘C56’, ‘C57’, ‘C58’, ‘C55’, ‘C60’, ‘C61’, ‘C62’, ‘C63’, ‘C64’, ‘C65’, ‘C66’, ‘C67’, ‘C68’ |
| Group2-4 | P3-LC/P4-HTM | ‘C69’, ‘C70’, ‘C71’, ‘C72’, ‘C73’, ‘C74’ ‘C75’, ‘C76’, ‘C77’, ‘C78’, ‘C79’ |

[표 11] Group 3

| **Group** | **Controller** | **Columns** |
| --- | --- | --- |
| Group3-1 | P1-PC/P1-TC/P4-STM | “C02”, “C24”, “C25”, “C26”, “C27”, “C28”, “C29, “C05”, “C08”, “C09”, “C10”, “C11”, “C12”, “C17”, “C18”, “C42”, “C79”, “C80”, “C81”, “C82”, “C83”, “C84”, “C85”, “C86” |
| Group3-2 | P1-FC/P1-LC/P4-STM | ‘C04’, ‘C07’, ‘C13’, ‘C14’, ‘C19’, ‘C20’, ‘C23’ ‘C03’, ‘C21’, ‘C22’, ‘C79’, ‘C80’, ‘C81’, ‘C82’, ‘C83’, ‘C84’, ‘C85’, ‘C86’ |
| Group3-3 | P1-CC | ‘C37’, ‘C38’, ‘C44’ |
| Group3-4 | P2-SC/P2-TC | ‘C45’, ‘C48’, ‘C53’, ‘C56’, ‘C57’, ‘C58’, ‘C55’, ‘C60’, ‘C61’, ‘C62’, ‘C63’, ‘C64’, ‘C65’, ‘C66’, ‘C67’, ‘C68’ |
| Group3-5 | P3-LC/P4-HTM | ‘C69’, ‘C70’, ‘C71’, ‘C72’, ‘C73’, ‘C74’ ‘C75’, ‘C76’, ‘C77’, ‘C78’, ‘C79’ |

[표 12] Group 4

| **Group** | **Controller** | **Columns** |
| --- | --- | --- |
| Group4-1 | P1-PC/P4-STM | ‘C02’, ‘C24’, ‘C25’, ‘C26’, ‘C27’, ‘C28’, ‘C29’, ‘C79’, ‘C80’, ‘C81’, ‘C82’, ‘C83’, ‘C84’, ‘C85’, ‘C86’ |
| Group4-2 | P1-FC/P1-LC/P1-CC/P4-STM | “C04”, “C07”, “C13”, “C14”, “C19”, “C20”, “C23” “C03”, “C21”, “C22”, “C37”, “C38”, “C44”, “C79”, “C80”, “C81”, “C82”, “C83”, “C84”, “C85”, “C86” |
| Group4-3 | P1-TC/P4-STM | ‘C05’, ‘C08’, ‘C09’, ‘C10’, ‘C11’, ‘C12’, ‘C17’, ‘C18’, ‘C42’ |
| Group4-4 | P2-SC/P2-TC | ‘C45’, ‘C48’, ‘C53’, ‘C56’, ‘C57’, ‘C58’, ‘C55’, ‘C60’, ‘C61’, ‘C62’, ‘C63’, ‘C64’, ‘C65’, ‘C66’, ‘C67’, ‘C68’ |
| Group4-5 | P3-LC/P4-HTM | ‘C69’, ‘C70’, ‘C71’, ‘C72’, ‘C73’, ‘C74’ ‘C75’, ‘C76’, ‘C77’, ‘C78’, ‘C79’ |

[표 13] Group 5

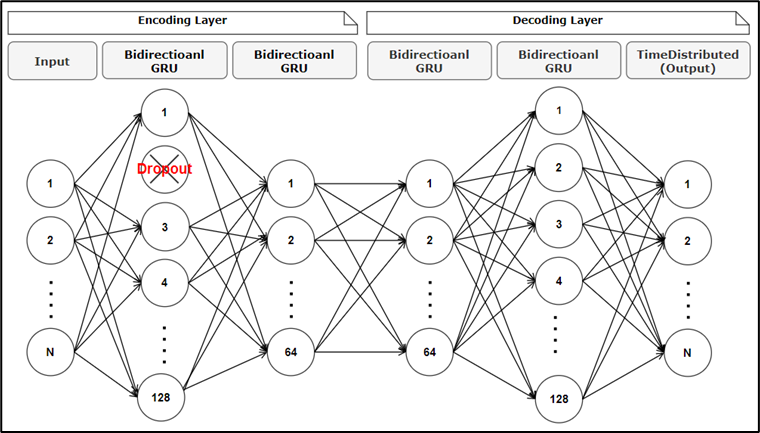
| **Group** | **Controller** | **Columns** |
| --- | --- | --- |
| Group5-1 | P1-PC/P4-STM | ‘C02’, ‘C24’, ‘C25’, ‘C26’, ‘C27’, ‘C28’, ‘C29’, ‘C79’, ‘C80’, ‘C81’, ‘C82’, ‘C83’, ‘C84’, ‘C85’, ‘C86’ |
| Group5-2 | P1-FC/P1-LC/P4-STM | ‘C04’, ‘C07’, ‘C13’, ‘C14’, ‘C19’, ‘C20’, ‘C23’ ‘C03’, ‘C21’, ‘C22’, ‘C79’, ‘C80’, ‘C81’, ‘C82’, ‘C83’, ‘C84’, ‘C85’, ‘C86’ |
| Group5-3 | P1-CC | ‘C37’, ‘C38’, ‘C44’ |
| Group5-4 | P1-TC/P4-STM | ‘C05’, ‘C08’, ‘C09’, ‘C10’, ‘C11’, ‘C12’, ‘C17’, ‘C18’, ‘C42’ |
| Group5-5 | P2-SC,P2-TC,P1-LC | 'C03', 'C21', 'C23', 'C24', 'C48', 'C53', 'C55', 'C56', 'C58', 'C65', 'C66’ |
| Group5-6 | P3-LC/P4-HTM | ‘C69’, ‘C70’, ‘C71’, ‘C72’, ‘C73’, ‘C74’ ‘C75’, ‘C76’, ‘C77’, ‘C78’, ‘C79’ |

[표 14] 그룹별 취합 결과

| **Group** | **TaP** | **TaR** | **TaPR** | **Detected anomalies** |
| --- | --- | --- | --- | --- |
| Group1 | 0.633 | 0.580 | 0.605 | 38 |
| Group2 | 0.466 | 0.639 | 0.539 | 43 |
| Group3 | 0.496 | 0.624 | 0.553 | 41 |
| Group4 | 0.468 | 0.722 | 0.568 | 46 |
| Group5 | 0.635 | 0.573 | 0.603 | 37 |

### 3) Bidirectional GRU Autoencoder

GRU 모델은 LSTM의 장기 의존성 문제에 대한 해결책을 유지하면서 구조를 간단화 시킨 모델로, 시계열 데이터에 적합하다고 판단되었다. 프로젝트에서 사용한 모델의 구조도와 코드는 [그림 30-31]과 같다.



[그림 30] Bidirectional GRU AutoEncoder 구조



[그림 31] Bidirectional GRU AutoEncoder python code

Controller별로 그룹을 나눠 학습을 진행 후 그룹을 결합하여 가장 좋은 성능을 가진 최종 취합 결과를 얻었다[표 15-16].

[표 15] 테스트 진행한 그룹 종류

| **Controller** | **Columns** |
| --- | --- |
| P1-FC, P1-LC, P4-STM | 'C03', 'C04', 'C07', 'C13', 'C14', 'C19', 'C20', 'C21', 'C22', 'C23', 'C79', 'C80', 'C81', 'C82', 'C83', 'C84', 'C85', 'C86' |
| P1-FC, P1-LC | 'C03', 'C04', 'C07', 'C13', 'C14', 'C19', 'C20', 'C21', 'C22', 'C23' |
| P4-STM, P1-SC, P2-TC | 'C45', 'C48', 'C53', 'C55', 'C56', 'C57', 'C58', 'C60', 'C61', 'C62', 'C63', 'C64', 'C65', 'C66', 'C67', 'C68', 'C79', 'C80', 'C81', 'C82', 'C83', 'C84', 'C85', 'C86' |
| P4-HTM, P3-LC | 'C21', 'C23', 'C69', 'C70', 'C71', 'C72', 'C73', 'C74', 'C76', 'C77', 'C78', 'C79', 'C84' |
| P4-STM, P1-PC, P1-TC | 'C02', 'C05', 'C08', 'C09', 'C10', 'C11', 'C12', 'C17', 'C18', 'C21', 'C24', 'C25', 'C26', 'C27', 'C28', 'C29', 'C42', 'C57', 'C79', 'C80', 'C81', 'C82', 'C83', 'C84', 'C85', 'C86' |
| P1-PC, P1-TC | 'C02', 'C05', 'C08', 'C09', 'C10', 'C11', 'C12', 'C17', 'C18', 'C21', 'C24', 'C25', 'C26', 'C27', 'C28', 'C29', 'C42', 'C57' |
| P1-CC | 'C37', 'C38', 'C44' |
| P3-LC, P1-FC | 'C02', 'C04', 'C07', 'C13', 'C19', 'C21', 'C23', 'C69', 'C70', 'C71', 'C72', 'C73', 'C74' |
| P2-SC, P3-LC, P2-TC | 'C02', 'C04', 'C21', 'C23', 'C24', 'C48', 'C53', 'C55', 'C56', 'C58', 'C65', 'C66', 'C69', 'C70', 'C71', 'C72', 'C73', 'C74' |
| P2-SC, P2-TC, P1-LC | 'C03', 'C21', 'C23', 'C24', 'C48', 'C53', 'C55', 'C56', 'C58', 'C65', 'C66' |
| P2-SC, P2-TC, P1-FC | 'C04', 'C07', 'C13', 'C19', 'C21', 'C23', 'C24', 'C48', 'C53', 'C55', 'C56', 'C58', 'C65', 'C66' |
| P2-SC, P2-TC, P1-PC | 'C21', 'C23', 'C24', 'C28', 'C48', 'C53', 'C55', 'C56', 'C58', 'C65', 'C66', 'C70', 'C72' |
| P2-SC, P2-TC, P1-TC | 'C05', 'C08', 'C09', 'C10', 'C11', 'C12', 'C17', 'C18', 'C21', 'C23', 'C24', 'C42', 'C48', 'C53', 'C55', 'C56', 'C58', 'C65', 'C66' |
| P2-SC, P2-TC, P1-CC | 'C21', 'C23', 'C24', 'C37', 'C38', 'C44', 'C48', 'C53', 'C55', 'C56', 'C58', 'C65', 'C66' |

[표 16] GRU 최종 취합 결과

| **Controller** | **TaP** | **TaR** | **TaPR** | **Detected anomalies** |
| --- | --- | --- | --- | --- |
| P1-FC, P1-LC | 0.489 | 0.143 | 0.222 | 9 |
| P4-STM, P1-SC, P2-TC | 0.754 | 0.115 | 0.200 | 7 |
| P4-HTM, P3-LC | 0.434 | 0.128 | 0.198 | 8 |
| P1-PC, P1-TC | 0.316 | 0.162 | 0.214 | 14 |
| P1-CC | 0.860 | 0.035 | 0.067 | 3 |
| P2SC-P2TC-P1LC | 0.736 | 0.294 | 0.420 | 19 |
| **Combine** | **0.418** | **0.451** | **0.434** | **32** |

### 

## 3. Isolation Forest + Bidirectional LSTM/GRU Autoencoder

실제로 현업에서 하이브리드 모델을 사용하는 경우가 많고, 성능이 향상될 수 있을 것이라고 판단되어 여러 모델의 결과를 취합하여 성능 평가를 해보았다. 먼저 Bidirectional LSTM/GRU Autoencoder와 Isolation Forest을 합쳐보았다. 그러나 여러 버전을 시도해보고 오탐 줄이기, 파라미터값 조정을 해보아도 성능이 기존보다 떨어졌는데, 이는 비교적 오탐 수치가 높은 Isolation Forest의 영향 때문에 성능이 떨어진 것으로 추측된다.

## 4. LSTM + GRU

두번째로 LSTM과 GRU의 결과를 취합해보았다. 두 모델 모두 데이터 그룹을 나누어 학습시켰기 때문에 각 모델의 경우의 수를 여러 가지 조합해본 결과 아래 [표 17]에서 가장 좋은 성능이 나왔다. LSTM Group1에서 사용한 5개의 그룹과, GRU에서 사용한 2개의 그룹 결과를 합친 결과 단일 모델보다 성능이 향상된 것을 확인할 수 있다. 그 이유는 LSTM과 GRU의 결과에서 서로 다른 Anomalies를 탐지하되 TaP가 적절히 높은 값을 가지는 그룹들을 선별하여 취합하였더니, TaR값이 높아져 전체적인 TaPR값 또한 상승된 것으로 추측된다.

[표 17] LSTM과 GRU의 취합 최종 결과

| **Model** | **Controller** | **TaP** | **TaR** | **TaPR** | **Detected anomalies** |
| --- | --- | --- | --- | --- | --- |
| LSTM | P1-PC, P4-STM | 0.560 | 0.257 | 0.352 | 17 |
| P1-FC, P1-LC, P4-STM | 0.590 | 0.252 | 0.353 | 18 |
| P1-TC, P4-STM | 0.921 | 0.018 | 0.035 | 1 |
| P2-SC, P2-TRIP | 0.657 | 0.181 | 0.284 | 12 |
| P3-LC, P4-HTM | 0.745 | 0.097 | 0.172 | 7 |
| GRU | P1-CC | 0.860 | 0.035 | 0.067 | 3 |
| P2SC-P2TC-P1LC | 0.736 | 0.294 | 0.420 | 19 |
|  | **Combine(최종)** | **0.653** | **0.608** | **0.630** | **40** |

# 

# **Ⅴ. 결론**

처음에 Isolation Forest, LSTM Autoencoder, GRU Autoencoder를 사용하여 학습을 진행하였지만 성능이 매우 좋지 않게 나와 문제점은 모델이라고 판단하였다. 하지만 OneClassSVM, conv1d-LSTM-autoencoder(Haicon2021 수상작 모델) 등의 새로운 모델들을 사용하여 학습을 진행하여도 여전히 TaPR 지표가 0.3을 넘기지 못하였다. 그리하여 여러 고민을 해보고 교수님과의 질의응답시간을 가진 결과, 성능이 좋지 않게 나온 이유는 모델이 문제가 아닌 데이터 전처리의 문제임을 깨닫고 데이터 전처리를 집중적으로 다시 점검해보았다. 이 과정에서 데이터 수집 장소를 고려해보자는 의견이 나오게 되어 같은 Controller 안에 있는 특성들을 묶어 데이터들을 그룹으로 나눠 각각 학습 시킨 뒤 결과를 취합하는 방식을 선택하게 되었다.

결론적으로 국가보안기술연구소에서 공개한 산업제어시스템 보안 데이터 HAI 22.04를 Isolation Forest, Bidirectional LSTM Autoencoder, Bidirectional GRU Autoencoder 총 세 개의 모델에 학습시켰으며, 추가로 LSTM의 결과와 GRU의 결과를 취합해보았다. 각 경우에 대한 최고 성능을 정리하면 [표 18]과 같다.

[표 18] 최종 결과

| **Model** | **TaP** | **TaR** | **TaPR** | **Detected anomalies** |
| --- | --- | --- | --- | --- |
| Isolation Forest | 0.277 | 0.531 | 0.364 | 41 |
| Bidirectional LSTM Autoencoder | 0.633 | 0.580 | 0.605 | 38 |
| Bidirectional GRU Autoencoder | 0.418 | 0.451 | 0.434 | 32 |
| LSTM + GRU | 0.653 | 0.608 | 0.630 | 40 |

2020년과 2021년에 HAI 데이터 다른 버전으로 대회가 진행되었는데, private 기준 성능 최대치가 각각 ‘0.93793’, ‘0.68455’이다. 2020년 HAI 보다 2021년 HAI 에서 새로운 공격패턴이 증가하고 데이터의 크기가 커져서 성능 점수가 낮아진 점을 미루어 보았을 때, 2021년 HAI보다 공격 패턴이 더 다양해진 2022년의 HAI로 프로젝트를 진행한 결과인 ‘0.630’은 꽤 준수한 성적이라고 할 수 있다.

데이터 전처리의 방향성을 잡았으니 기존에 실패했던 CNN, OneClassSVM이나 추가적인 모델들을 활용해보면 좋을 것 같다. 그리고 전체적으로 TaPR 값이 높게 나오는 방향으로 프로젝트를 진행하였지만, TaPR 값이 좀 낮아지더라도 최대한 많은 공격을 탐지하는 모델을 만들어보는 방향도 좋을 것 같은데 못해봐서 아쉬움이 남는다.

# **참고문헌**

[1] 조 타이디, “미국 송유관 해킹 사건: 범죄 조직의 몸값 요구가 딜레마인 이유, BBC, ”<https://www.bbc.com/korean/international-57135880>, 2021년 5월 17일.

[2] 문가용, “혼다 공격한 스네이크, ICS와 SCADA 집중적으로 노리는 랜섬웨어”, 보안뉴스, <https://www.boannews.com/media/view.asp?idx=88892&kind=14>, 2020년 6월 11일.

[3] HAI Data Set, https://github.com/icsdataset/hai

[4] Data Set info, <https://github.com/icsdataset/hai/blob/master/hai_dataset_technical_details_v3.0.pdf>

[5] eTaPR, <https://dacon.io/competitions/official/235757/talkboard/404109?page=1&dtype=recent>

[6] Hwang, Won-Seok, et al. "Time-series aware precision and recall for anomaly detection: considering variety of detection result and addressing ambiguous labeling." *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019.

[7] Normalization, https://heeya-stupidbutstudying.tistory.com/32

[8] Xu, Dong, et al. "An improved data anomaly detection method based on isolation forest." *2017 10th international symposium on computational intelligence and design (ISCID)*. Vol. 2. IEEE, 2017.

[9] Anomaly Detection with Isolation Forest & Visualization,

https://towardsdatascience.com/anomaly-detection-with-isolation-forest-visualization-23cd75c281e2

[10] Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation forest." *2008 eighth ieee international conference on data mining*. IEEE, 2008.

[11] Explain Your Model with the SHAP Values,

<https://towardsdatascience.com/explain-your-model-with-the-shap-values-bc36aac4de3d>

[12] How to reshape data, <https://towardsdatascience.com/how-to-reshape-data-and-do-regression-for-time-series-using-lstm-133dad96cd00>

[13] Bidirectional class, <https://wiserloner.tistory.com/1276>

[14] Reasons for using tanh activation function, <https://coding-yoon.tistory.com/132>

[15] Xavier initialization, <https://techblog-history-younghunjo1.tistory.com/236>

[16] Reasons for using Xavier initialization, <https://velog.io/@cha-suyeon/DL-%EA%B0%80%EC%A4%91%EC%B9%98-%EC%B4%88%EA%B8%B0%ED%99%94Weight-Initialization->

[17] Dropout, <https://sonsnotation.blogspot.com/2020/11/7-dropout.html>

[18] TimeDistributed layer, k

[19] Adam optimizer, <https://onevision.tistory.com/entry/Optimizer-%EC%9D%98-%EC%A2%85%EB%A5%98%EC%99%80-%ED%8A%B9%EC%84%B1-Momentum-RMSProp-Adam>

[20] MSE loss function, <https://bskyvision.com/822>

[21] EarlyStopping, <https://3months.tistory.com/424>

[22] Mini batch, <https://jalbin.tistory.com/39>