2024 년 전기

# 초소형 MCU 와 IMU 센서를 활용한 온 디바이스 Anomaly Detection 구현

# 목차

## 5. 보고 시점까지의 과제 수행 내용 및 중간 결과

5-1. Anomaly Dataset 에 대한 분석

5-2. LSTM 모델 실행 결과

5-3. GRU 모델 실행 결과

5-4. Transformer 모델 실행 결과

5-5. 낮은 성능을 기록한 데이터에 대한 분석 및 모델 개선

5-6. Tensorflow Lite 를 사용한 포팅

## 6. 피드백 반영사항

## 7. 결론

7-1. 진행사항 요약

7-2. 향후 진행 계획

# 1. 요구조건 및 제약 사항 분석에 대한 수정사항

## 1-1. 과제 배경

현대 사회에서는 산업 현장에서 데이터 과학 및 인공지능 기술의 도입이 활발하게 이루어지고 있다.

산업 현장에서의 생산, 제조 설비에서 이상이 발생할 경우, 공정이 멈추거나 불량이 발생하는 등 심각한 경제적 손실이 발생할 수 있다. 이러한 상황을 방지하기 위해 이상 데이터를 수집, 분석하여 일반적인 상황과 이상 상황을 구분하는 것이 중요 해졌다.

이상 감지 기술을 통해 설비 이상을 실시간으로 탐지할 수 있다면 설비 이상으로 인한 경제적 손실을 최소화할 수 있기 때문에 이상 감지 기술에 대한 수요가 높은 상황이다.

## 1-2. 목표

- *정상 데이터와 이상 데이터를 구분할 수 있는 Dataset 수집*
- *공장의 베어링이나 회전기계에 붙여 IMU 센서를 통해 데이터 수집 후 이상치를 판단할 수 있는 모델 및 임베디드 보드 시스템 설계 및 구현*
- *저전력 MCU 에서 구동 가능한 경량화 된 딥러닝 라이브러리를 사용하여 모델 학습*
- *학습된 모델을 임베디드 보드에 포팅 하여 이상 상태를 판별*

데이터셋을 이용해 공정 시스템의 Anomaly 를 감지하는 딥러닝 모델을 설계한다. 임베디드 보드의 상대적으로 작은 KB 대의 메모리 환경에서 모델을 실행하기 위해서, Tensorflow Lite for Microcontrollers 를 사용하여 모델을 경량화 한다.

베어링이나 회전 팬과 같은 회전설비, 기계에 부착된 IMU 센서에서 데이터를 수집 후, 임베디드 보드를 통해 Anomaly 가 탐지된 경우, 블루투스 통신을 통하여 외부 기기에 알림을 보내, 공정의 시스템에 문제가 발생함을 알리게 한다.

## 1-3. 제약 사항 및 대책

● Anomaly Dataset 부족 문제

- 산업 현장의 사용되는 장비나 운영 방식에 따른 다양성에 의해 어떤 경우를 anomaly 로 라벨링 할지에 대한 기준이 명확하지 않다.
- 컴퓨터 비전, nlp 의 식별 및 판별 문제 등에 비해 산업 현장에서는 anomaly 를 예방하려고 하기 때문에, anomaly 에 대한 label 이 적다.

√ 대책

- 데이터 증강, 증식 혹은 전이학습을 이용하여 데이터셋을 늘린다.
- 실제 공정과 비슷한 환경을 시뮬레이션 하여 직접 데이터셋을 만든다.

● 초소형 MCU 의 제한된 딥러닝 구현 환경

- 일반적으로 KB 단위 메모리 크기를 가진 MCU 에서 머신러닝 모델을 구동할 수 없는 경우가 많고, 구동하는 경우에도 PC 등과 같은 환경에 비해 프로세스의 성능이 떨어진다.
- MCU 에 탑재할 수 있는 TensorFlow, PyTorch 라이브러리를 지원하는 경우가 많지 않다.

√ 대책

- MCU 중 딥러닝 탑재가 용이한 ESP32, nRF, STM32 등을 사용하여 시스템을 구현한다.
- Tensorflow Lite for Microcontrollers, CMSIS-NN 과 같은 MCU 에 최적화된 딥러닝 라이브러리를 사용하여 모델을 개발한다.

## ● MCU 구동 환경에 따른 이상치 측정

- IMU 센서는 외부 진동, 충격 등으로 인한 센서 데이터의 변동이 있을 수 있다.
- 외부 환경에 따른 변수를 제외하더라도 IMU 센서가 항상 정확한 데이터를 수집하지 못할 수 있다.

### √ 대책

- 데이터 전처리를 통해 이상치를 제외하고 학습한다.
- 데이터를 수집할 센서를 여러 개 설치하여 얻은 값 중 중간 값을 활용한다.

# 2. 설계 상세화 및 변경 내역

## 2-1. Anomaly Dataset 수집

https://github.com/SmartManuAD/Smart-Manufacturing-AD/tree/main?tab=readme-ov-file

해당 사이트는 스마트 제조 공정에 대한 데이터셋이 있다. 여러 분야의 공정과정에서 수집된 이상 탐지 데이터셋이 있고, 이 데이터 셋을 통해 모델을 학습시킬 수 있다.

그림 1 과 같이 여러 분야에 대한 Anomaly Dataset 을 얻을 수 있다.

| ID | Transformed Name | Link to source | Description | Original_Format | Economic Activity | I |
|---|---|---|---|---|---|---|
| 5 | cfrp | Anonymous | CFRP (Carbon Fiber Reinforcement Polymer): Automatic Fiberplacement fabrication priocess for aerostructures. | CSV | Aerospace | aer mar |
| 14 | PlantMonitoring | link | Production Plant Data for Condition Monitoring | CSV | Machinery & Equipment | Pla fa |
| 15 | Multistage | link | Multi-stage continuous-flow manufacturing process | CSV | Machinery & Equipment | M mar |
| 18 | Robotfail | link | Robot Execution Failures Data Set This dataset contains force and torque measurements on a robot after failure detection. Each failure is characterized by 15 force/torque samples collected at regular time intervals. | CSV | Machinery & Equipment | F |
| 21 | Engine | link | Public D3M datasets: Engine (Simple engine data): | CSV | Automotive components | Au |

그림 1. Smart Manufacturing Anomaly Detection 데이터셋

## 2-2. 모델 개발 및 학습

이번 과제에 이용할 모델은 LSTM, GRU, Transformer 이다.

LSTM(Long Short-Term Memory)은 순환 신경망(RNN, Recurrent Neural Network)의 일종으로, 긴 시퀀스 데이터를 처리할 수 있도록 고안된 모델이다. LSTM 은 기존의 순환 신경망의 단점인 장기 종속성 학습 문제를 해결하여, 장기적인 의존성을 잘 학습할 수 있다.

LSTM 의 핵심요소는 Cell state, Gate 이다.

- Cell state : 시퀀스 데이터를 따라 정보를 전달하며, 필요 없는 정보는 제거하고 중요한 정보는 유지하는 역할을 한다.
- Input Gate : 새로운 정보가 셀 상태에 얼마나 중요한지를 결정한다.
- Forget Gate : 이전 셀 상태의 정보를 얼마나 버릴지를 결정한다.
- Output Gate : 셀 상태에서 어떤 정보를 출력으로 보낼 지를 결정한다.

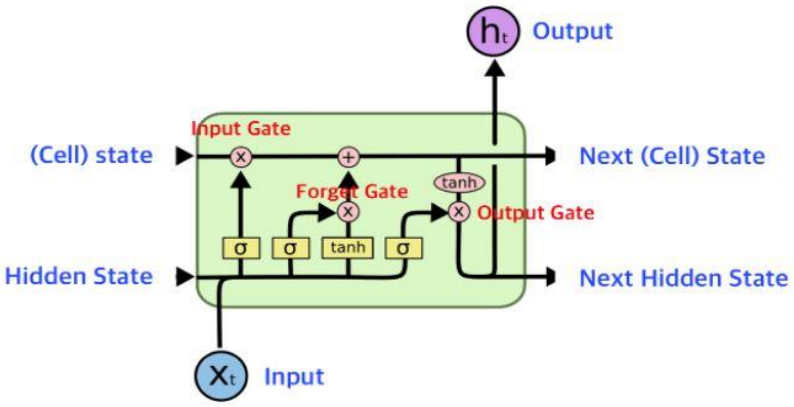



그림 2. LSTM 모델 아키텍처

GRU(Gated Recurrent Unit)은 LSTM 의 변형으로, 구조가 더 간단하면서도 비슷한 성능을 내기 위해 고안된 모델이다. LSTM 과 달리 Cell State 가 없으며, hidden state 를 통해 정보를 유지하고 전달한다. 그리고 Update Gate, Reset Gate 두 개의 게이트만을 사용한다.

- Update Gate : 현재 입력과 이전 상태를 바탕으로 새로운 상태를 업데이트할 지를 결정한다.
- Reset Gate : 얼마나 이전 상태를 무시하고 새로운 정보를 받아들일지를 결정한다.

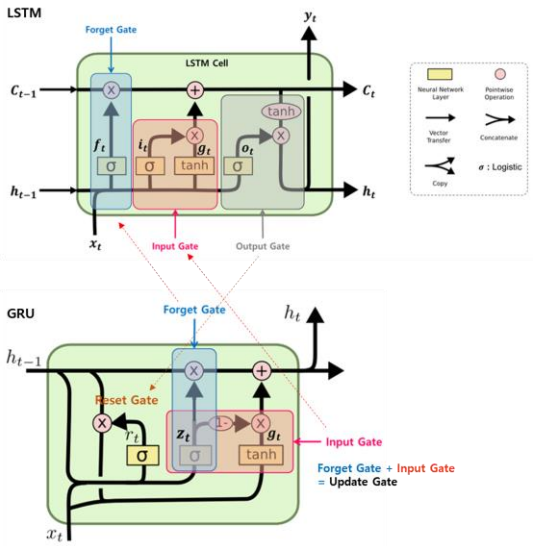



그림 3. LSTM 과 GRU 의 비교

Transformer 는 RNN 구조를 사용하지 않고 시퀀스 데이터를 병렬로 처리할 수 있는 모델로, Self-Attention 메커니즘을 사용하여 시퀀스 내 모든 위치의 요소들 간의 관계를 학습한다. 데이터를 병렬 처리한다는 것은, 모델 학습 속도를 크게 향상시키며, 이는 실시간으로 이상 탐지를 하는 데 매우 유용할 수 있다.

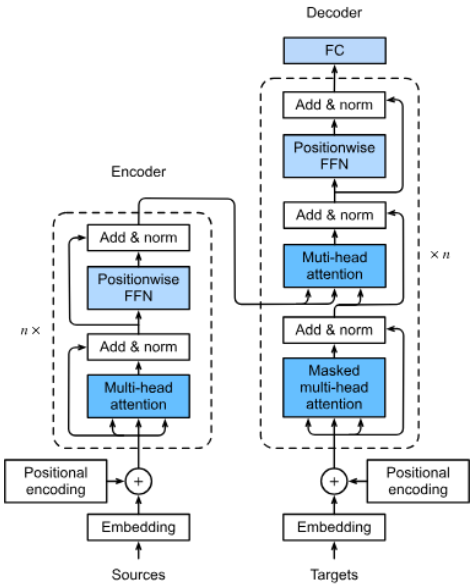



그림 4. Transformer 아키텍처

위 3 개의 모델로 데이터셋을 학습시켜 정확도와 Loss 를 획득하여 이상 탐지에 적합한 모델을 선정한다.

## 2-3. 라즈베리 파이 딥러닝 환경 구축

라즈베리파이에 모델을 포팅 하기 위해 라즈베리파이 환경을 구축해야 한다.

우선, 라즈베리파이 공식 홈페이지에서 운영체제를 다운로드하여 SD 카드에 저장한다.


그림 5. 라즈베리파이 공식 홈페이지

SD 카드에 운영체제를 설치한 후, 라즈베리파이에 SD 카드를 삽입한다. 그리고 라즈베리파이 내에서 파이썬 가상환경을 만든 후, 가상환경 내에서 필요한 모듈과 라이브러리를 다운로드 한다. 필요한 라이브러리는 Tensorflow, keras, Numpy 등이 있다.

이후 MNIST 예제를 다운로드 하여 학습 후, Tensorflow Lite 모델로 변경하여 정상적으로 딥러닝 환경이 구축되었는지 확인한다.

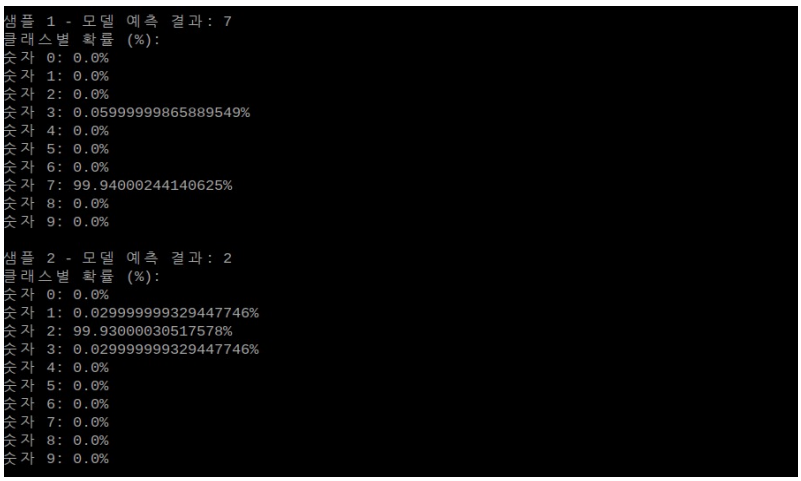



그림 6. MNIST 예제 실행 결과

그림 6 과 같이 Tensorflow Lite 모델이 정상적으로 실행되었다. 이로써 라즈베리파이 환경에서의 모델 테스트 환경이 구축되었다.

## 2-4. ESP32 환경 구축

이번 과제에서 사용할 ESP32 모듈은 ESP32-DevKitM-1 이다.

해당 모듈에서의 딥러닝, 블루투스 통신 등의 환경을 구축하기 위해서 ESP-IDF 가 필요하다.

ESP-IDF 는 ESP 마이크로컨트롤러를 위한 소프트웨어 개발을 지원하며, 네트워킹, 보안, 스토리지 등에 필요한 라이브러리와 툴체인을 포함하고 있다. 해당 프로그램은 Visual Studio Code 의 확장프로그램으로 설치가 가능하여 Visual Studio Code 에서의 개발이 가능하다.

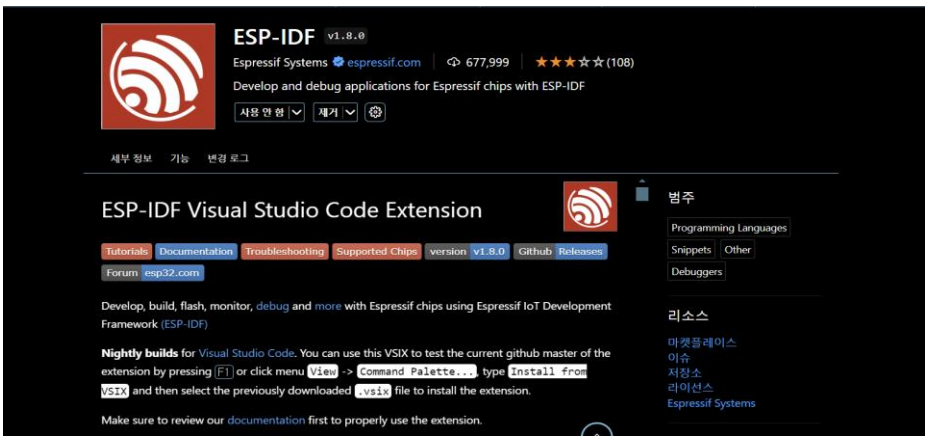



그림 7. VSC 의 확장프로그램인 ESP-IDF

## 3. 갱신된 과제 추진 계획

| 개발 구분 | 세부 항목 | 8월 | | | 9월 | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 5 |
| 인공지능 모델 개발 | 모델 성능 테스트 (상위 모듈) | | | | | | | | |
| 경량화 및 무선 통신 환경 구축 | 모델 경량화 | | | | | | | | |
| | MCU 펌웨어 개발 | | | | | | | | |
| | MCU 보드 포팅 | | | | | | | | |
| | 모델 성능 테스트 (저전력 MCU 모듈) | | | | | | | | |
| Application 개발 | 연동 앱 개발 | | | | | | | | |
| | 앱 연동 테스트 및 최종 보완 | | | | | | | | |

## 4. 구성원별 진척도

| 이름 | 역할 |
|---|---|
| 김민재 | - Data Preprocessing<br>- 머신러닝 모델 설계<br>- 클라이언트 앱 설계 |
| 최세영 | - 데이터셋 수집, 분석<br>- 머신러닝 모델 설계<br>- 머신러닝 모델 포팅 |
| 김경준 | - 라즈베리파이 모델 테스트 환경 구축<br>- 라즈베리파이 환경에서의 모델 포팅 |
| 공통 | - 기초 지식 학습<br>- 보고서 작성<br>- 딥러닝 모델 테스트 |

# 5. 보고 시점까지의 과제 수행 내용 및 중간 결과

## 5-1. Anomaly Dataset 에 대한 분석

Anomaly Dataset 은 주로 시간에 따라 나열된 Time Series 데이터이다.
따라서 RNN, LSTM, GRU 등과 같은 timestep 에 따라 이전 학습결과를 반영할 수 있는 모델을 고려하여 설계한다.

## 5-2. LSTM 모델 구축과 실행 결과

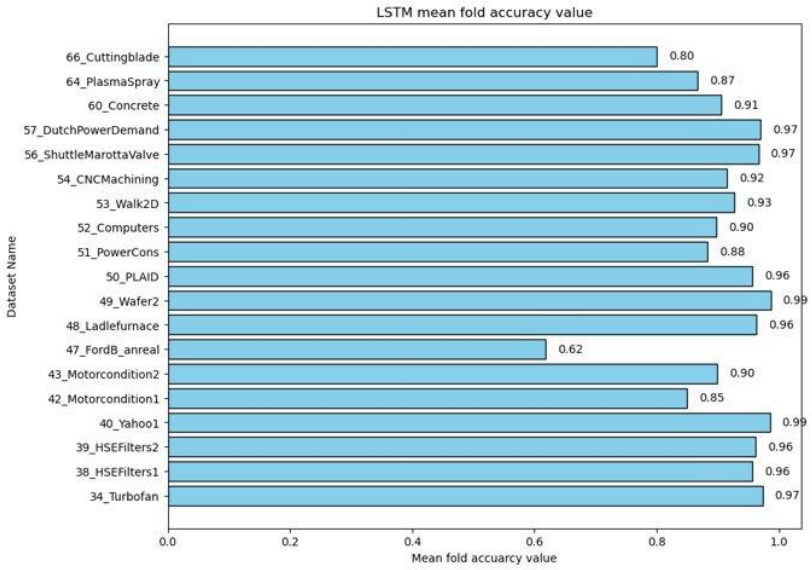



그림 8. 기존 LSTM 모델 학습에 따른 정확도

기존 LSTM 기반의 모델의 경우는 평균 90.6%의 정확도를 가진다. 그러나 데이터의 불균형 문제를 고려해 모델을 구축해야 한다. LSTM 과 GRU 모델은 1D Convolution Layer 를 두 개 추가하여 모델의 복잡성을 줄이면서도 성능을 높일 수 있었다.

또한, 아래 그림 9 ~ 15 는 개선된 모델의 Accuracy, Loss, Precision, Recall, F1-score, Validation Accuracy, Validation Loss 이다.

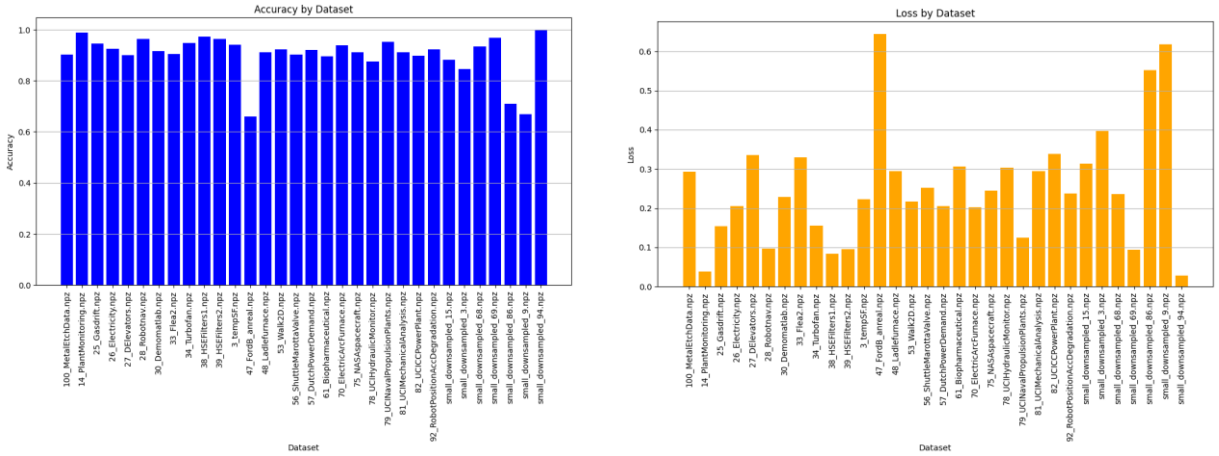



그림 9, 10. 개선된 LSTM 모델 학습에 따른 Accuracy, Loss

Loss 그래프의 경우, 일부 데이터 셋을 제외하면 Loss 가 낮게 측정되었다.

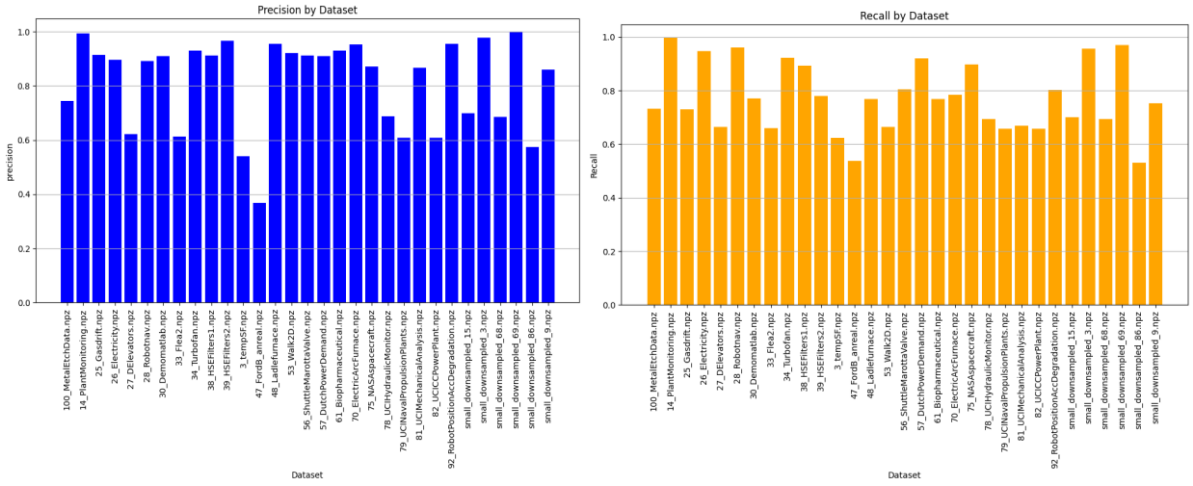



그림 11, 12. 개선된 LSTM 모델 학습에 따른 Precision, Recall

Recall 값의 경우, 이전의 모델과 비교해서 크게 개선된 모습을 확인할 수 있었다.

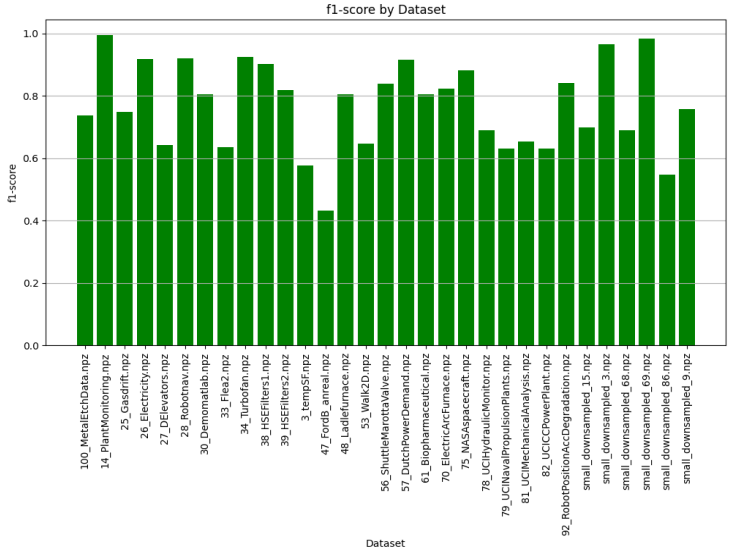



그림 13. 개선된 LSTM 모델 학습에 따른 F1-score

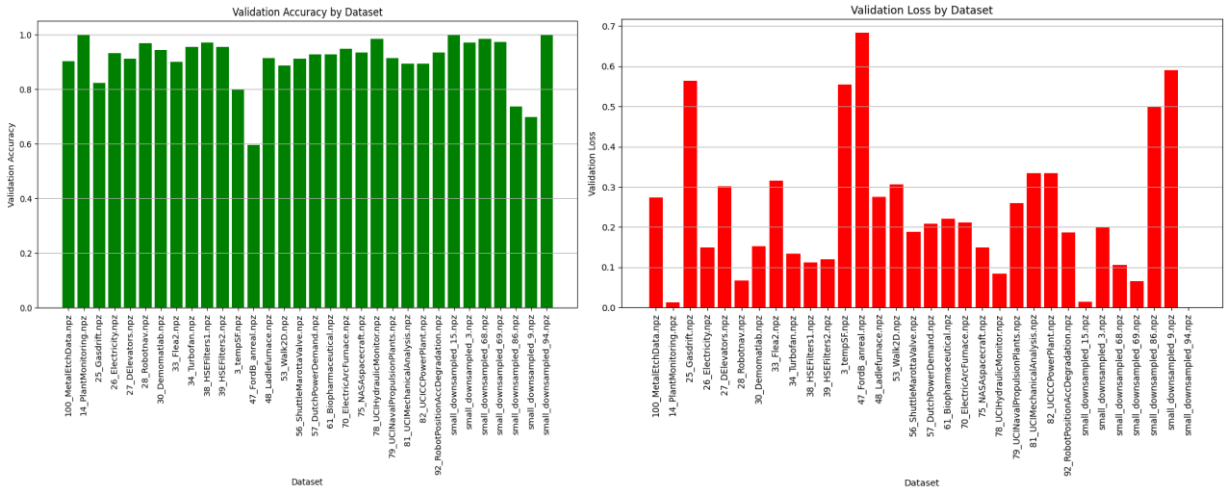



그림 14, 15. 개선된 LSTM 모델 학습에 따른 Validation Accuracy, Validation Loss

# 5-3. GRU 모델 실행 결과

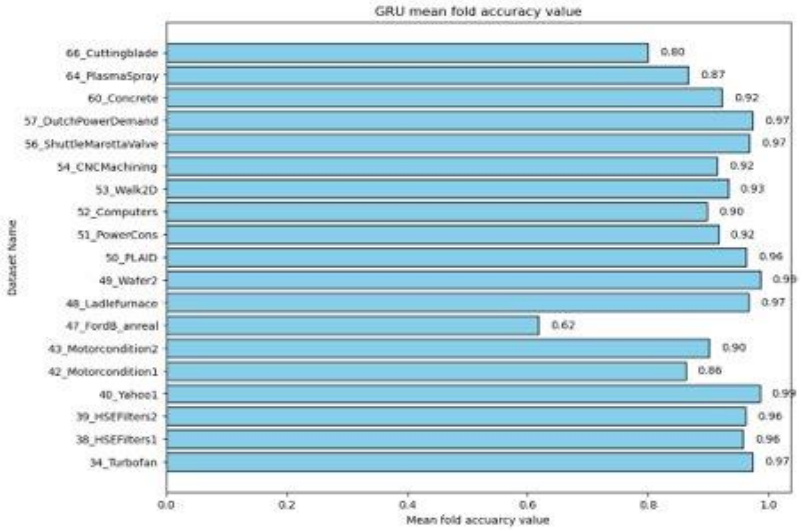



그림 16. GRU 모델 학습에 따른 정확도

기존 GRU 기반의 모델의 경우는 평균 91.5%의 정확도를 가진다. GRU 모델 역시 LSTM 과 같이 1D Convolution Layer 를 두 개 추가하여 모델의 복잡성을 줄이면서도 성능을 높일 수 있었다.

아래 그림 17 ~ 23 은 개선된 모델의 Accuracy, Loss, Precision, Recall, F1-score, Validation Accuracy, Validation Loss 이다.

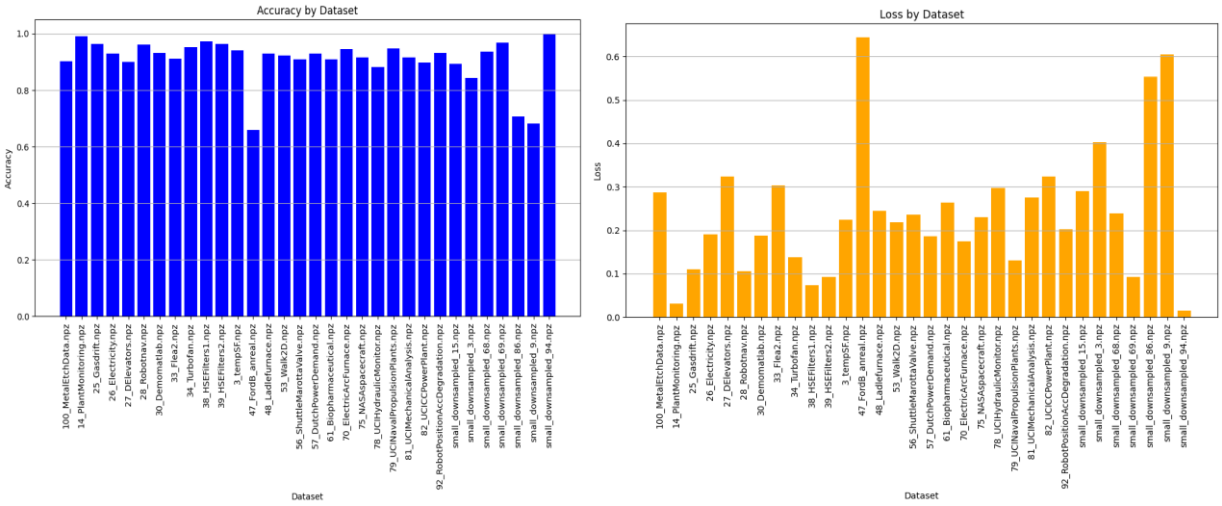



그림 17, 18. 개선된 GRU 모델 학습에 따른 Accuracy, Loss

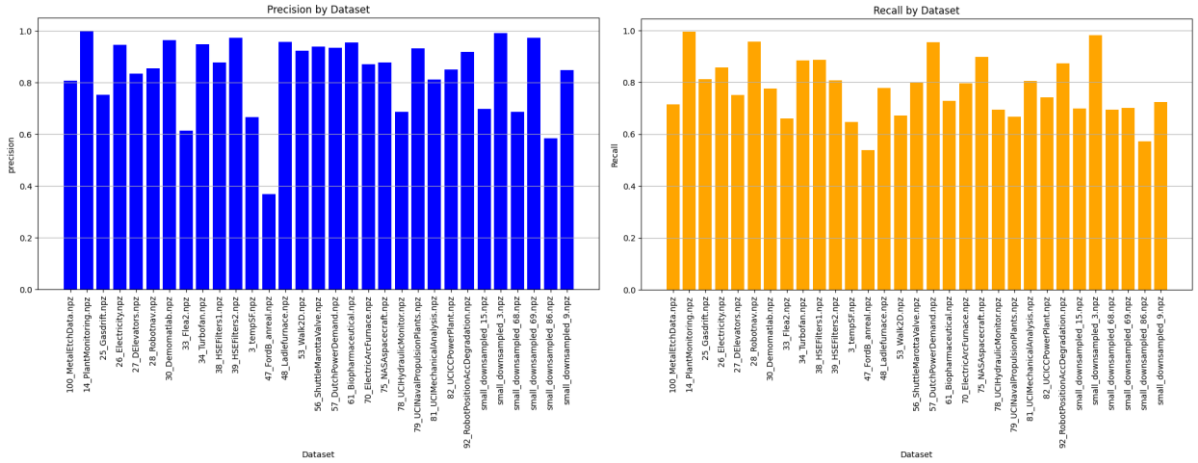



그림 19, 20  개선된 GRU 모델 학습에 따른 Precision, Recall

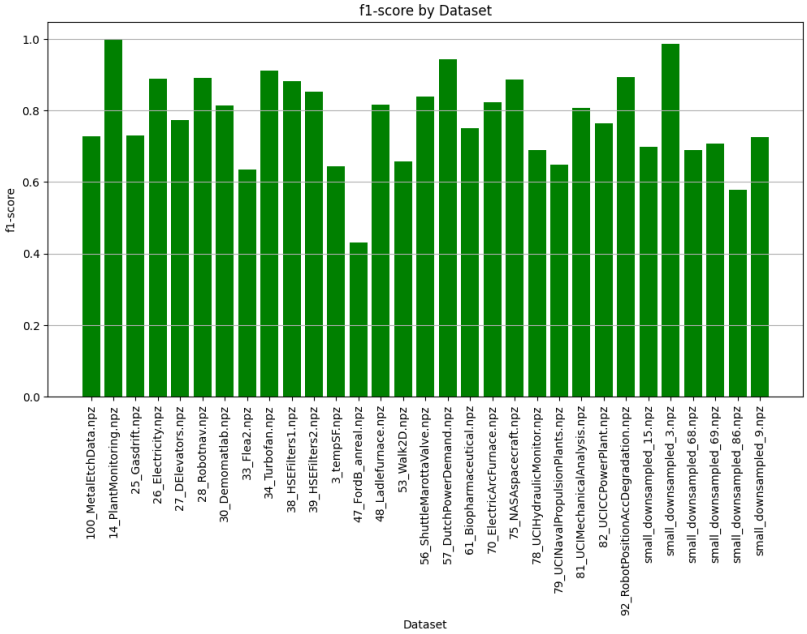



그림 21. 개선된 GRU 모델 학습에 따른 F1-score

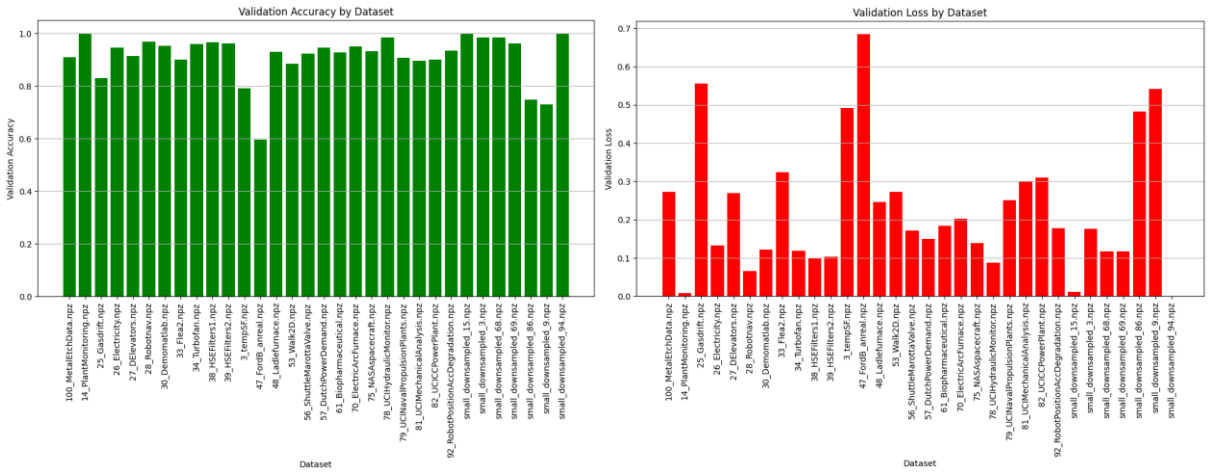



그림 22, 23. 개선된 GRU 모델 학습에 따른 Validation Accuracy, Validation Loss

# 5-4. Transformer 모델 실행 결과

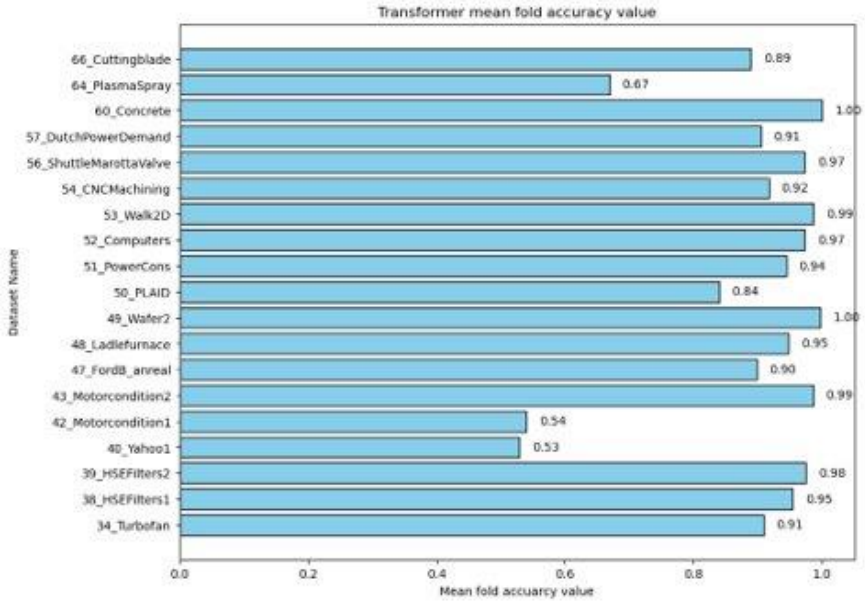



그림 24. Transformer 모델 학습에 따른 정확도

Transformer 기반의 모델의 경우는 평균 88.7%의 정확도를 가진다.

# 5-5. 낮은 성능을 기록한 데이터에 대한 분석 및 모델 개선

기존에 Preprocessing 을 거치지 않은 데이터 셋 중  낮은 모델 성능을 기록한 데이터셋은 그림 25 와 같은 kdd 데이터셋과 같이 모델의 중간 혹은 양단에 큰 시간 간격동안  anomaly 가 발생하지 않거나,

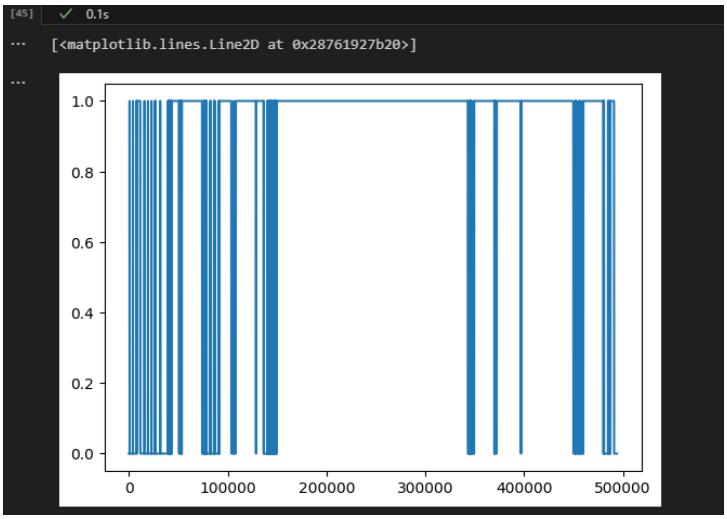



그림 25. 이상 데이터셋의 예시

혹은 그림 26 과 같은 Motorcondition 데이터셋과 같이

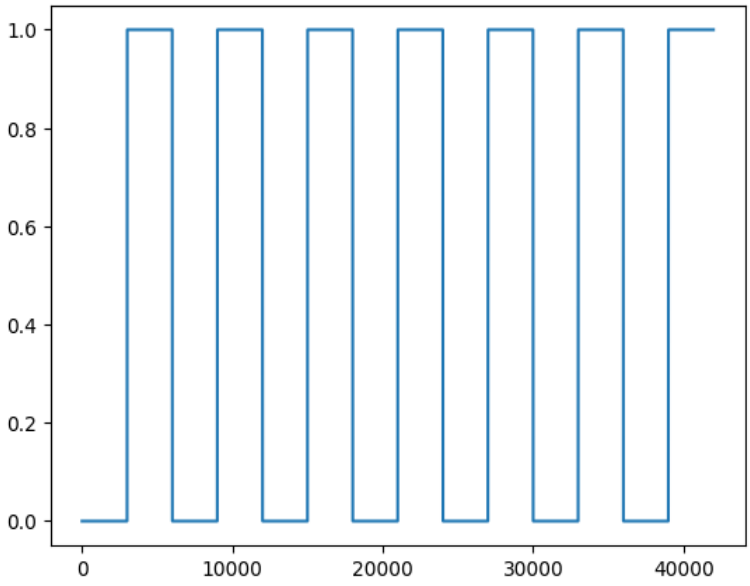
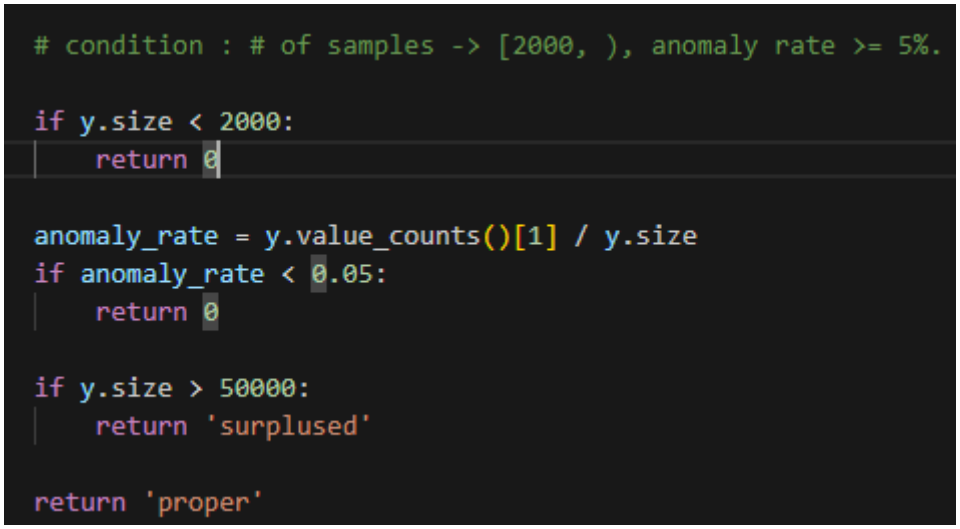



그림 26. Motorcondition 데이터셋의 데이터 분포

시계열 데이터가 아닌 경우가 존재하였다.
전자의 경우, 해당하는 시간 간격을 잘라 샘플을 이어 붙이는 방식을 사용하여 preprocessing 을 진행하였다.

후자의 경우 모델이 올바르게 학습할 수 없기 때문에, 학습 데이터에서 제외하였다.

데이터셋의 크기가 너무 작거나, anomaly 데이터의 비율이 너무 낮은 경우, 모델이 적절하게 학습할 수 없기 때문에, 그림 27 과 같이 데이터셋의 샘플 개수가 2000 개보다 적거나, 혹은 anomaly 데이터의 비율이 5%보다 작은 경우 학습에서 제외하였다.



```
# condition : # of samples -> [2000, ), anomaly rate >= 5%.

if y.size < 2000:
    return 0

anomaly_rate = y.value_counts()[1] / y.size
if anomaly_rate < 0.05:
    return 0

if y.size > 50000:
    return 'surplused'

return 'proper'
```

그림 27. 샘플 개수가 2000 개 미만이거나 anomaly 데이터의 비율이 5%보다 작은 데이터셋을 제외하는 코드

반대로, 이진 레이블을 가진 anomaly 데이터셋에 대하여 데이터 샘플의 크기가 너무 큰 경우, 모델이 과대적합 될 수 있고 모델의 학습 속도가 현저히 느려지는 점을 감안하여 일정한 time slice 를 사용하여 모델의 샘플 개수를 10 만개 전후로 downsampling 하는 방식으로 preprocessing 을 진행하였다.
이러한 기준에 따라
총 54 개의 smartManuAD 데이터셋들을 사용하여 모델을 학습하였다.

# 5-6. Tensorflow Lite 를 사용한 포팅

스마트폰이나 라즈베리 파이와 같은 휴대용 기기를 위한 Tensorflow Lite 를 사용하여 라즈베리 파이에 모델을 포팅 할 수 있다.

또한, esp32, arduino nano 등의 KB 단위의 메모리를 가진 전자기기를 위한 Tensorflow Lite for Microcontrollers 를 통해 Tensorflow 로 설계한 머신러닝 모델을 포팅 할 수 있다.


그림 28. Tensorflow Lite for Microcontrollers 가 지원되는 하드웨어 목록

Tensorflow 로 학습된 h5 모델을 Tensorflow Lite 모델로 변환하는 코드는 다음과 같다.

```
import tensorflow as tf

# .h5 모델 로드
h5_model = tf.keras.models.load_model('gru_model_fold_1.h5')

# .tflite 변환기 생성
converter = tf.lite.TFLiteConverter.from_keras_model(h5_model)

# 양자화 적용
converter.optimizations = [tf.lite.Optimize.DEFAULT]

# .tflite 모델로 변환
tflite_model = converter.convert()

# .tflite 모델 저장
with open('gru_model_fold_1.tflite', 'wb') as f:
    f.write(tflite_model)
```

그림 29. Tensorflow Lite 모델로의 변환 코드

Tensorflow Lite 모델을 생성 후 정상적으로 변환되었는지 확인한 결과, 정상적으로 변환되었음을 알 수 있다.

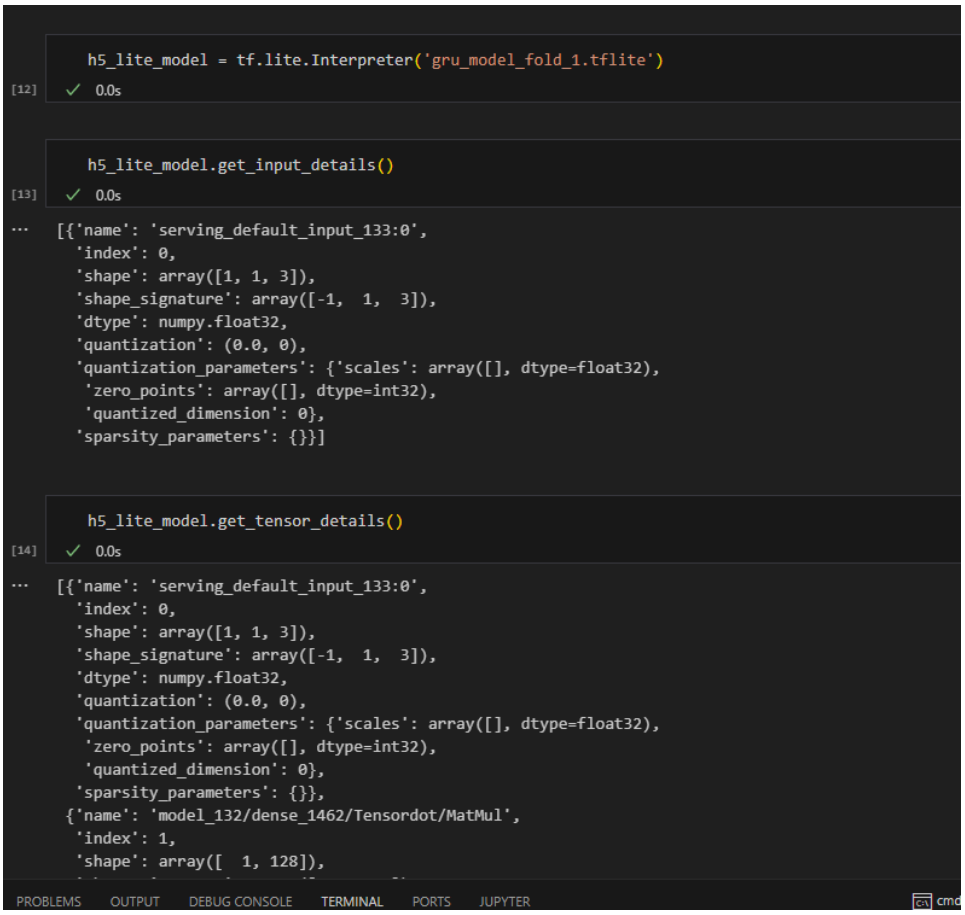



그림 30. 변환된 Tensorflow Lite 모델의 정보

# 6. 피드백 반영사항

## 6-1. 데이터 불균형 문제 해결

데이터셋에 anomaly 데이터의 비율이 너무 낮아 데이터셋이 불균형한 경우, 데이터를 특정 time slice 로 잘라내어 normal 데이터의 개수를 줄여 anomaly 데이터의 비율을 늘려  데이터 불균형 문제를 개선하였다.

## 6-2. Confusion matrix 를 사용한 점수 평가

기존의 모델은 정확도를 기준으로 모델의 성능을 평가했다. 정확도는 정답 레이블에 대하여 모델이 예측한 결과가 옳은 경우를 모든 경우의 수에 대하여 나눈 값이다. 일반적인 분류 데이터셋과 달리, Anomaly Detection 의 경우에는  Anomaly 데이터의 개수가 매우 적고, 때로는 1% 정도밖에 안 되는 경우가 있다.
이러한 경우는, 모델이 모든 데이터셋에 대하여 Normal 로 판단한 경우에도 99%가 된다. 이는 정확도는 높지만 모델이 올바르게 데이터를 판단하지 못하는 상황일 수 있다.
따라서, 학습에 정확도를 Recall(TP / (TP + FN) )와  Precision 을 조화 평균하여 낸 점수인 F1-score 를 지표로 사용하였다.

다음은 CNC Machine data set 에 대해 점수 성능 평가를 진행한 내용이다.

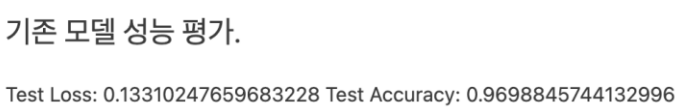



그림 31. 기존 모델 성능 평가

기존 모델의 경우 정확도가 96%에 달한다. 그러나 상위 10 개의 테스트 데이터에 대해 성능 평가를 진행하면 아래와 같은 양상을 띈다.

```
[[0.02759254]
 [0.02759254]
 [0.02759254]
 [0.02759254]
 [0.02759254]
 [0.02759254]
 [0.02759254]
 [0.02759254]
 [0.02759254]
 [0.02759254]]
[[0]
 [0]
 [0]
 [1]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]]
```

그림 32. 상위 10 개 테스트 데이터 평가

모델이 모든 테스트 데이터에 대해 동일한 확률로 평가하고 있기에 차별적 예측을 하지 못하고 있다. 이는 이 모델이 다수 클래스에 대해 치우친 예측을 수행하고 있다는 점을 알 수 있다. 자주 발생하는 0 번 클래스에 대해 높은 성능을 보이지만, 상대적으로 적게 발생하는 클래스에 대해 예측 성능이 떨어진다. 이는 False Negative 같은 중요한 이벤트를 놓칠 수 있다. 그렇기에 Confusion matrix 를 사용해 다양한 지표로 평가하는 것이 중요하다.

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98   2594650
           1       0.71      0.07      0.13     84274

    accuracy                           0.97   2678924
   macro avg       0.84      0.54      0.56   2678924
weighted avg       0.96      0.97      0.96   2678924

[[2592201    2449]
 [  78228    6046]]
```

그림 33. CNC Machine data set Confusion matrix

그림 33 에서 데이터의 주된 0 class 에 대해 높은 precision 값을 띈다. 그러나 Anomaly label 인 1 class 에 대해 낮은 recall 값을 가지는 것을 확인할 수 있다.

모델의 목표가 드물게 발생하는 이상탐지를 정확하게 해내는 것이기 때문에 Recall, F1-score 값이 모델의 성능을 나타내는 중요한 평가지표로 볼 수 있다.

모델은 발생하는 모든 이상치 데이터를 탐지하는 것이기에 False Negative 를 최소화하는 것이 중요하다.

```
Test Loss: 0.25558552145957947
Test Accuracy: 0.8961313962936401
```

그림 34. 개선된 모델의 정확도

```
[[0.86605346]
 [0.31688705]
 [0.9465973 ]
 [0.997297  ]
 [0.07958007]
 [0.09201054]
 [0.7215259 ]
 [0.98327386]
 [0.06335972]
 [0.05657313]]
[1 0 1 1 0 0 1 1 0 0]
```

그림 35. 상위 10 개 테스트 데이터 평가

```
              precision    recall  f1-score   support

           0       0.86      0.94      0.90    165331
           1       0.94      0.85      0.89    168745

    accuracy                           0.90    334076
   macro avg       0.90      0.90      0.90    334076
weighted avg       0.90      0.90      0.90    334076

[[155977   9354]
 [ 25346 143399]]
```

그림 36. CNC Machine data set Confusion matrix

이후 개선한 모델의 경우 정확도 값은 이전 모델보다 떨어졌지만, 모델이 데이터에 대해 정확한 평가를 시도하는 것을 확인할 수 있다. 특히 Recall 값이 0.07 에서 0.85 로 유의미하게 증가한 모습을 보여주고 있다.

# 7. 결론

## 7-1. 진행사항 요약

Anomaly Detection 시스템을 개발하기 위하여 정상 데이터와 이상 데이터를 구분할 수 있는 SmartManuAD 데이터셋들을 이용하였다.
데이터셋 중 일부는 anomaly 데이터 비율이 매우 낮거나, 혹은 샘플의 크기가 충분하지 않는 등 모델 학습을 시키기에 적절하지 않다.
따라서, 이러한 데이터셋에 대하여 시간 간격에 따라 데이터를 간소화하거나, 모델 학습에 적절하지 않을 것으로 판단되는 부분은 잘라내고, 데이터 가중치를 곱하는 전처리를 진행하였다.

특히 기존의 모델에 1D CNN 레이어 두 개를 추가하여 Recall 과 F1-score 평가치에서 유의미한 성능의 개선을 보였다.

## 7-2. 향후 진행 계획

- **모델을 마이크로 컨트롤러 및 라즈베리파이에 포팅**

  지금까지 사용한 데이터셋을 통해 얻은 가장 일반화된 모델을 Tensorflow lite micro 를 사용하여 라즈베리 파이와 esp32 에 포팅 한다.

- **마이크로 컨트롤러 포팅을 위한 펌웨어 개발**

  만든 모델을 실행시키고 실행시킨 결과를 웹에 전송하기 위한 인터페이스를 개발한다.

- **Anomaly 를 판단할 수 있는 웹 개발**

  만들어진 모델이 새로 들어온 값에 대하여 Anomaly 로 판단할 경우, 마이크로 컨트롤러의 송신기를 통하여 웹 서버가 수신하여 이를 화면에 출력할 수 있는 웹 서비스를 개발한다.