

2023

MJU Data Analytics

CRNN Project



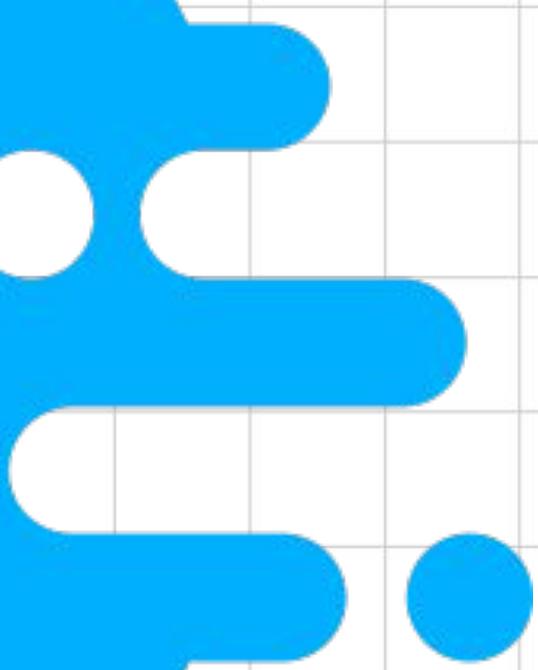
조원 : 권혁민 60192467
장원준 60211451
이현서 60202531

목차

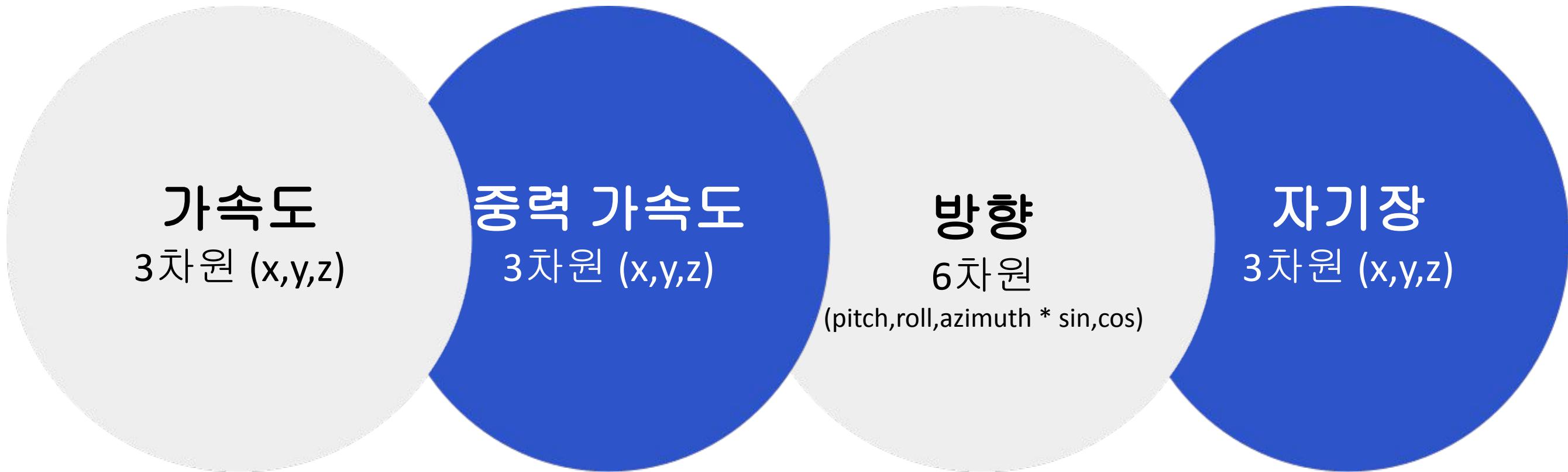
- 01 Dataset
- 02 EDA
- 03 Preprocessing
- 04 Feature Engineering
- 05 Modeling & Tuning

01

Dataset



- Sensor data



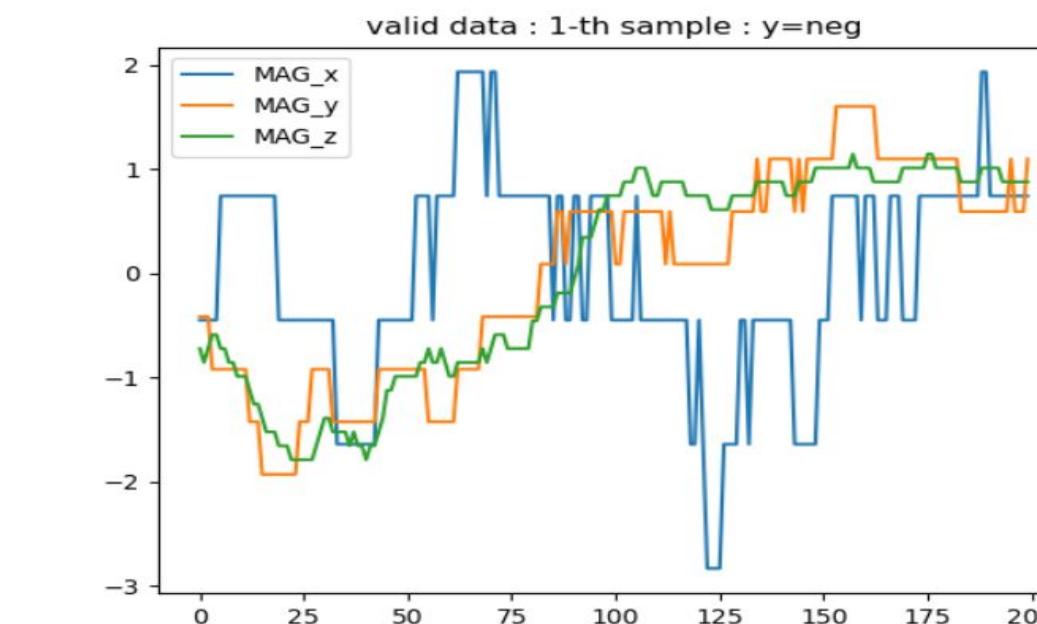
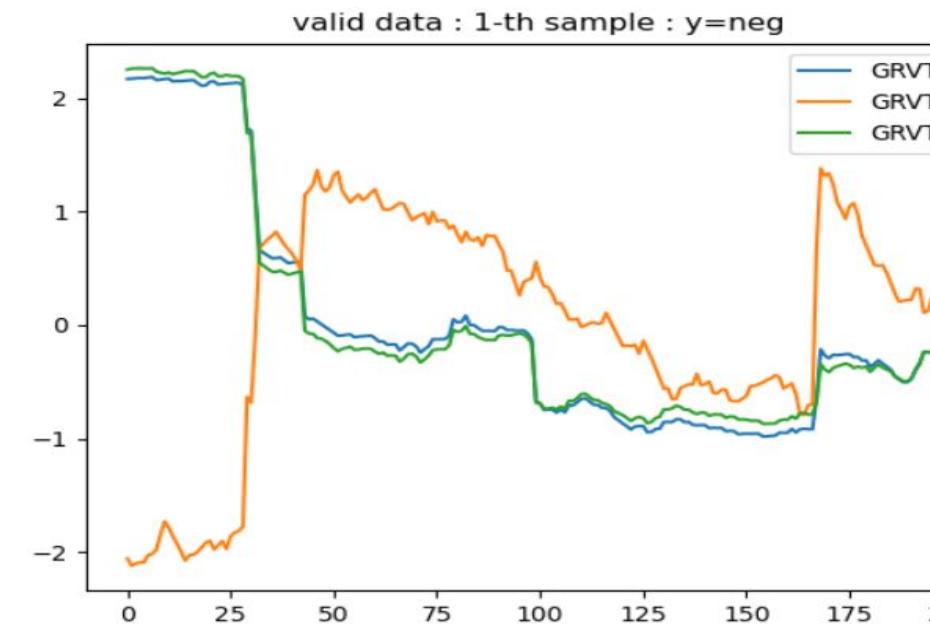
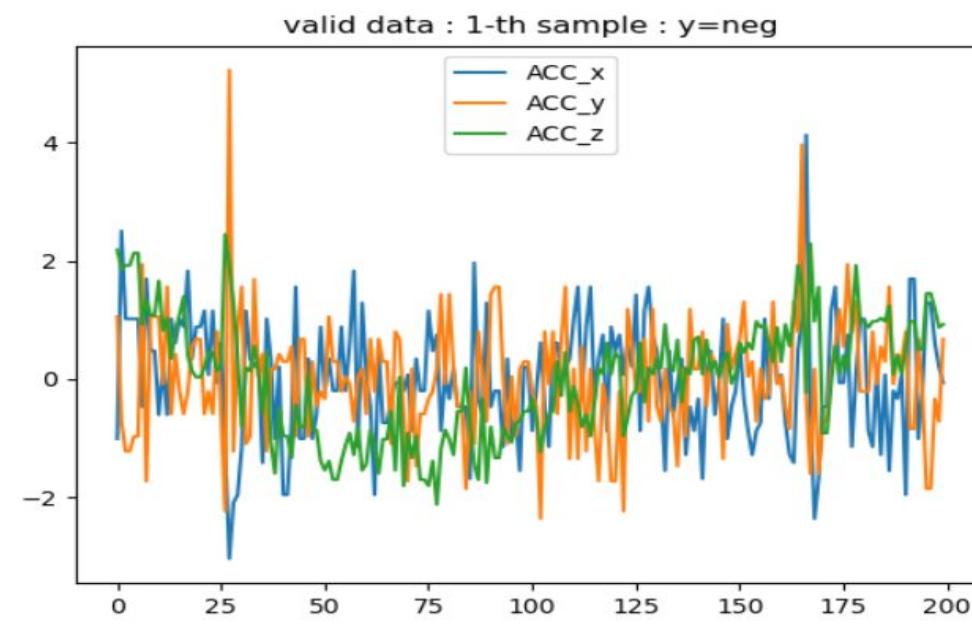
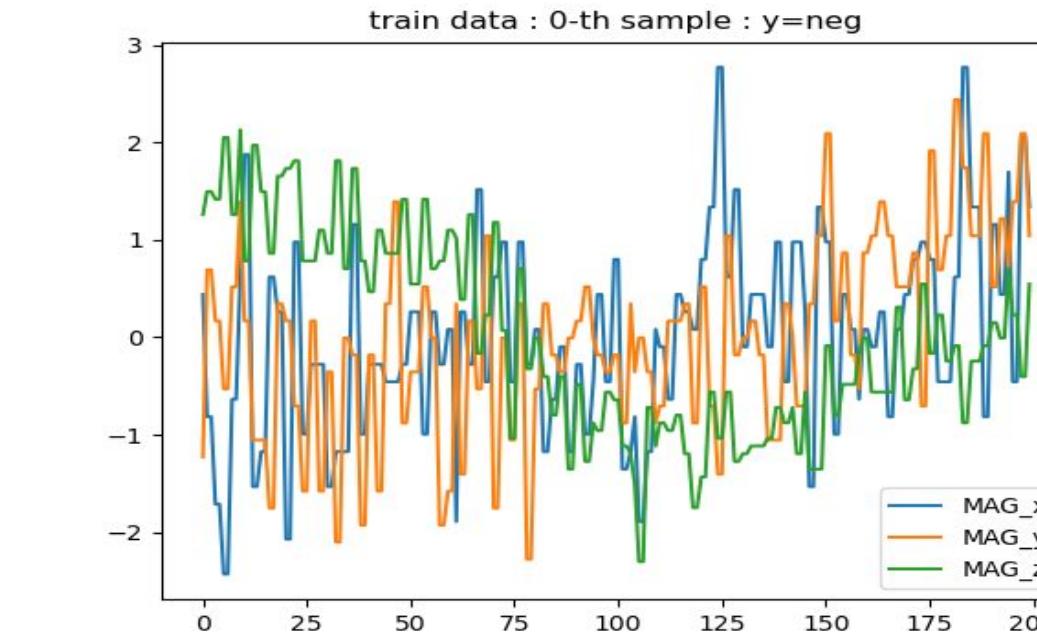
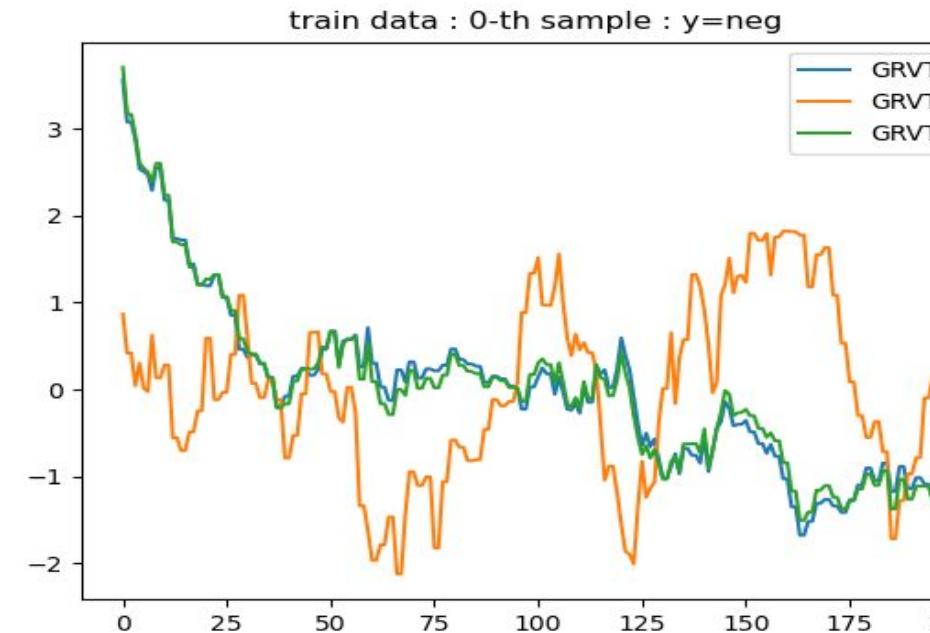
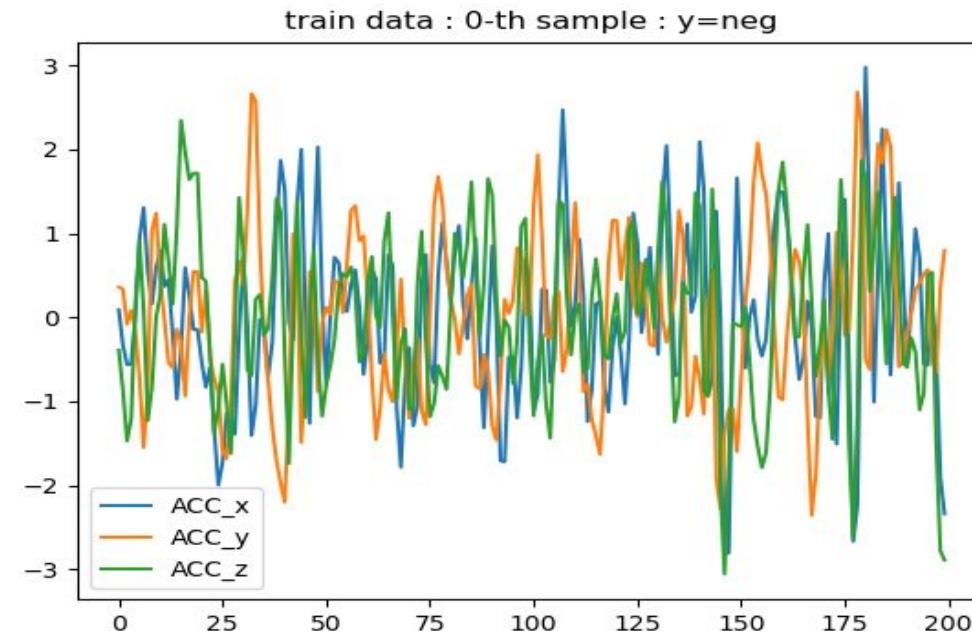
- 핸드폰 내부의 측정 센서를 통해 핸드폰의 회전 및 움직임을 측정한 데이터 셋
 - 방향 센서 : pitch, roll, azimuth 총 3 종류의 방향 센서 (sin,cos 별로 존재)
 - pitch : 위 아래 각도 (평평 : 0도, 지구 중심에서 아래로 : +90도, 위로 : -90도)
 - roll : 왼쪽 오른쪽 각도 (왼쪽으로 롤링 : 0~+90도, 오른쪽으로 롤링 : 0~-90도)
 - azimuth : 동서남북을 나타냄 (북 : 0도, 동 : +90도, 남 : +180도, 서 : 270도)

핸드폰의 회전 방향 및 방법을 알기 위해 pitch, roll, azimuth를 활용하여 분류 진행
→ pitch, roll의 부호들의 조합으로 바라보는 방향 및 회전 방향을 예상

01 Dataset

- dataset.py의 시각화

- ACC, GRVT, MAG

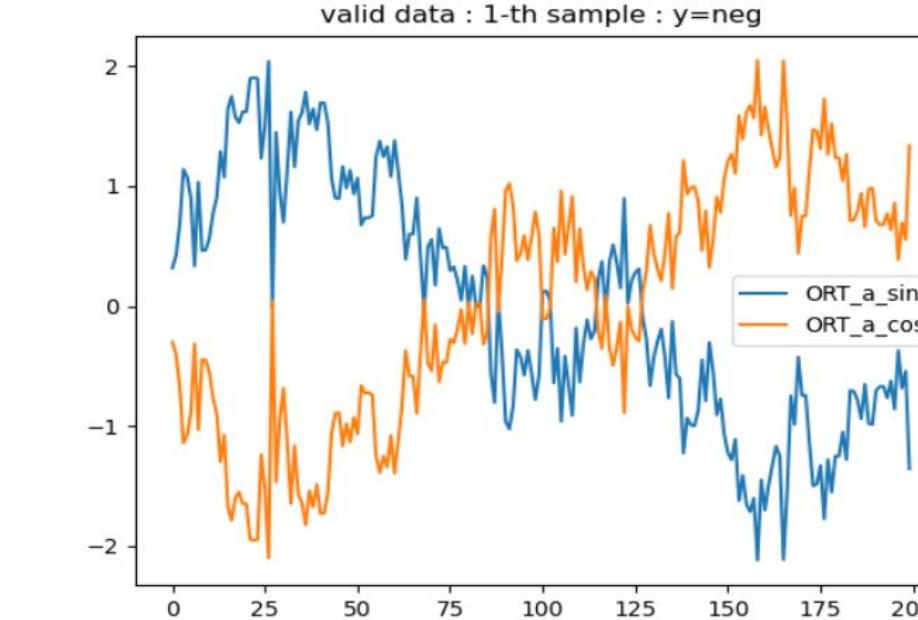
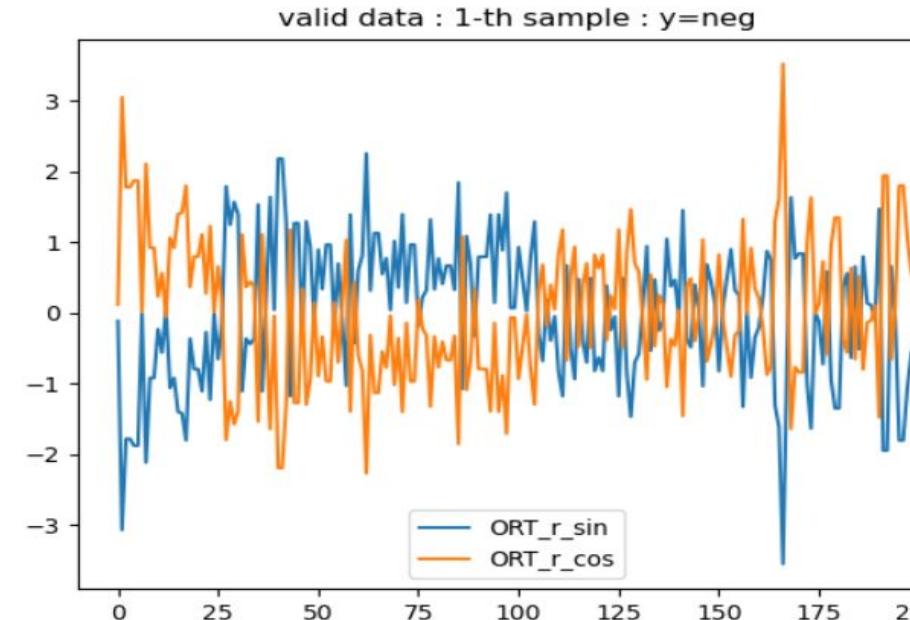
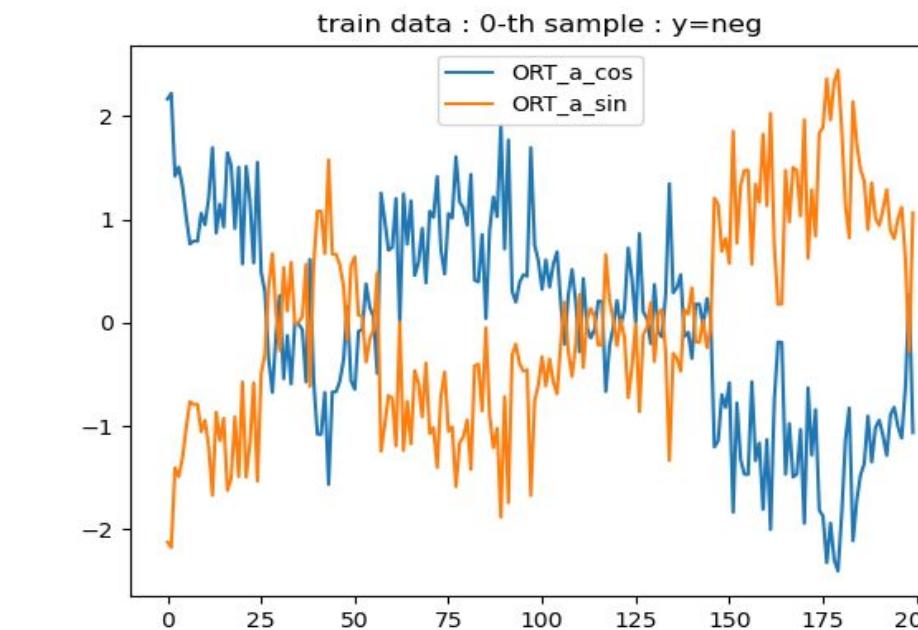
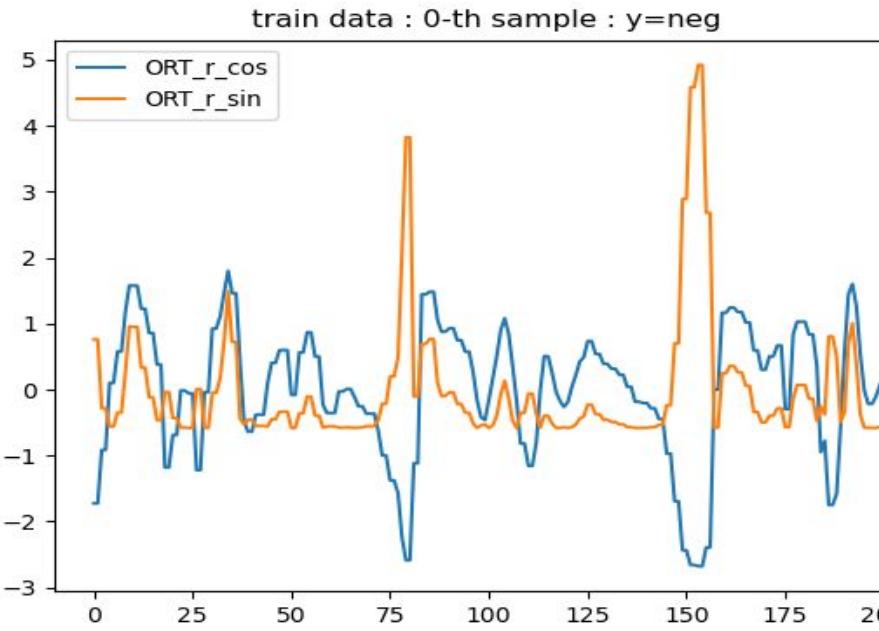
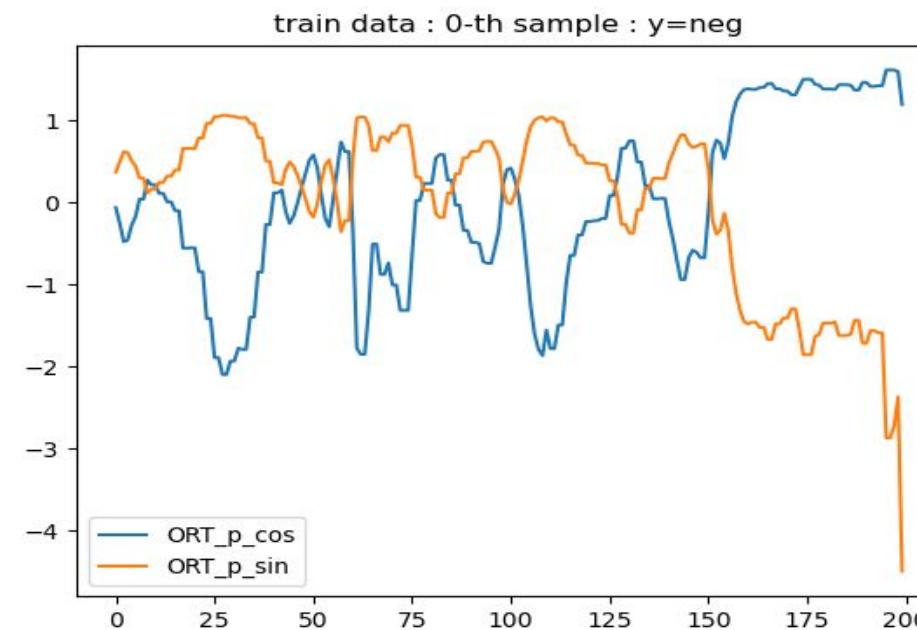


- 데이터 분석을 위해 plot들을 찍어보며 특징성을 찾아보려 했지만 plot의 형태의 복잡성과 해당 plot이 의미하는 바를 자세히 분석하지 못하여 plot을 EDA 부분에서 활용하지 못 하였음
- sample_idx를 다르게 할 때마다 데이터들의 분포가 완전 다르게 나와 유의미한 주어진 데이터셋으로의 plot 분석은 의미가 없어 보임

01 Dataset

- dataset.py의 시각화

- Pitch, Roll, Azimuth



- ORT 데이터에 대한 plot들에서 얻을 수 있는 인사이트는 sin과 cos이 음의 상관 관계를 크게 갖을 것으로 예상 되었음
 - 당연히 수식적으로 관계가 있으므로 둘의 상관 관계는 당연한 결과
- ORT 또한 유의미한 인사이트를 도출하지 못함

02

EDA

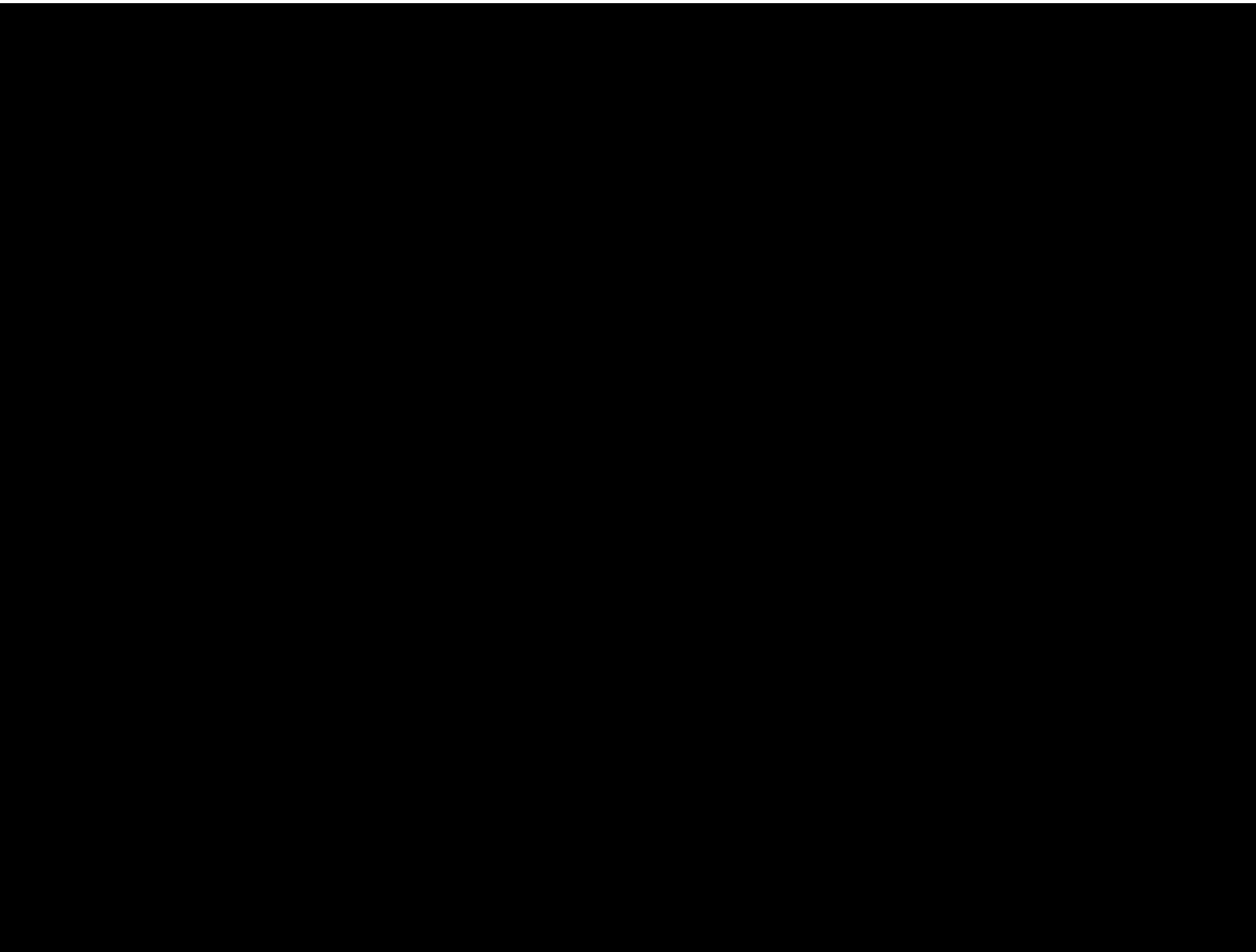


- 해당 plot들로 유의미한 인사이트를 도출해내지 못한 이유

- plot의 복잡성
 - 너무 많은 데이터와 다양성 때문에 선 그래프로 표현을 하더라도 plot의 복잡성이 너무 높음
- 해당 plot이 어떤 행동일 때 나타나는 행동인지 모름
 - plot에 대한 이해를 하더라도 해당 plot이 실험자가 어떤 행동을 했을 때 측정되는 데이터인지 모름
→ 해당 plot이 실험자가 주머니에서 꺼냈을 때의 데이터인지, 책상 위에서 들어올린 데이터인지 등에 대한 정보가 없음
- 센서 데이터들이 무엇을 의미하는지 어느 방향인지 자세히 모름
 - 어떤 행동을 기반으로 만들어진 데이터인지 정보가 없으므로 왜 이 데이터가 증가, 감소하는지에 대한 정보가 없음
→ 각각의 데이터들이 핸드폰의 어느 방향을 설명하는지, 어떤 의미인지를 모름
- 위의 문제점들을 해결하기 위해 "직접" 핸드폰에 센서 데이터 측정 어플을 다운로드하여 여러 상황에서의 센서 데이터를 수집하기로 결정

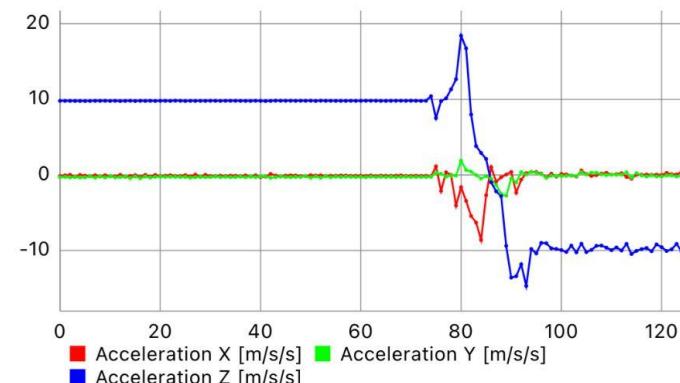
- 해당 plot들로 유의미한 인사이트를 도출해내지 못한 이유

- 눈물 젓은 데이터 수집

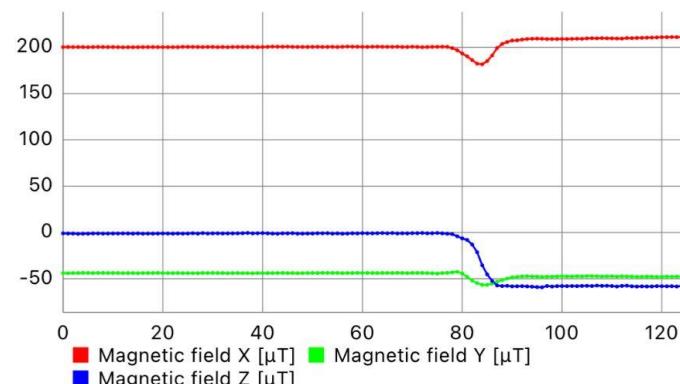
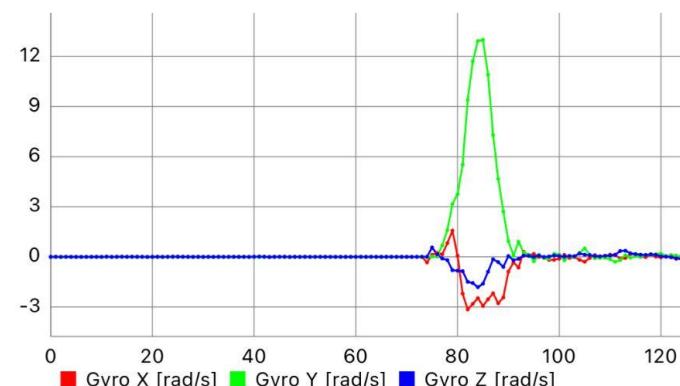


● Sensor Data를 이해하기 위해 직접 센서 데이터 실험 진행

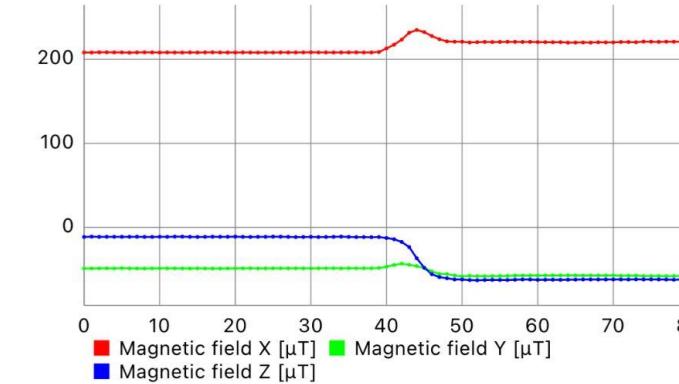
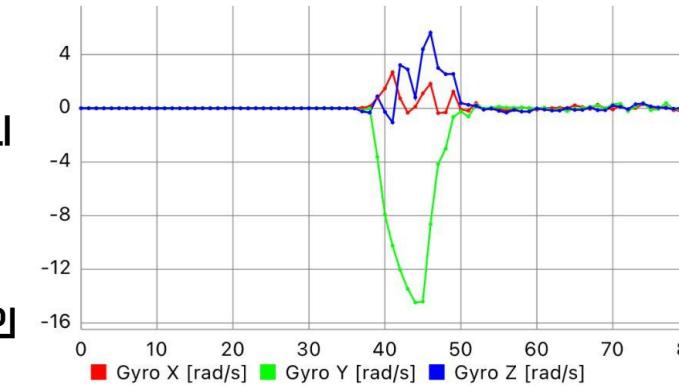
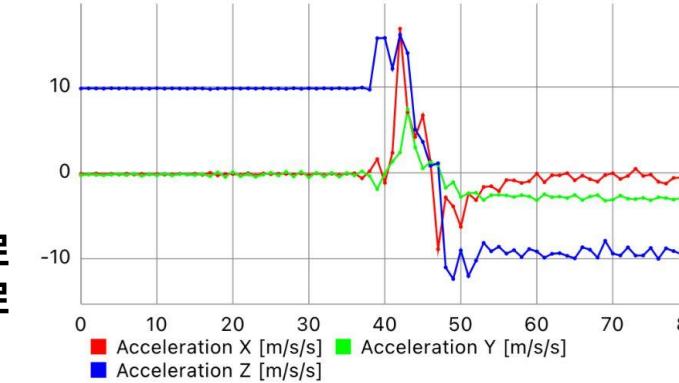
핸드폰 뒤집기 (화면을 뒤집은 상태에서 x축 시계 방향으로)



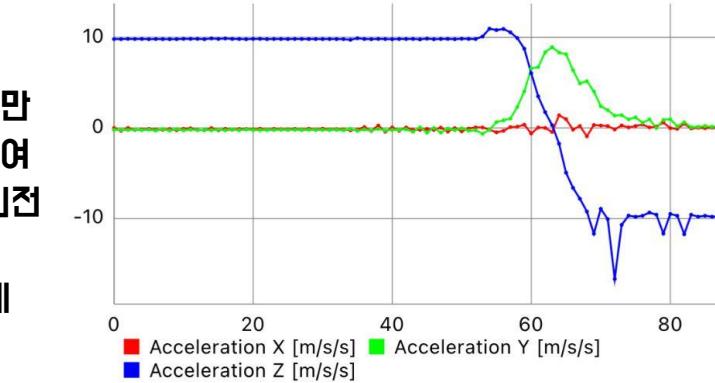
- 가장 기본적인 핸드폰 뒤집기의 센서를 얻기 위해 진행
- 핸드폰의 화면을 아래로 가도록 뒤집어둔 상태로 시작하여 z축 방향으로 중력가속도가 9.8m/s/s 로 작용하는 것을 알 수 있음
→ 핸드폰을 뒤집었더니 중력가속도의 방향이 z축에서 -z축으로 바뀜
- (휴대폰 전면기준) x축의 음의 방향으로 회전하므로 x에 대한 가속도는 (-) → (+) → 0으로 변화
- y축을 주축으로 삼아 뒤집어서 Gyro Y값이 크게 움직인 것을 볼 수 있음



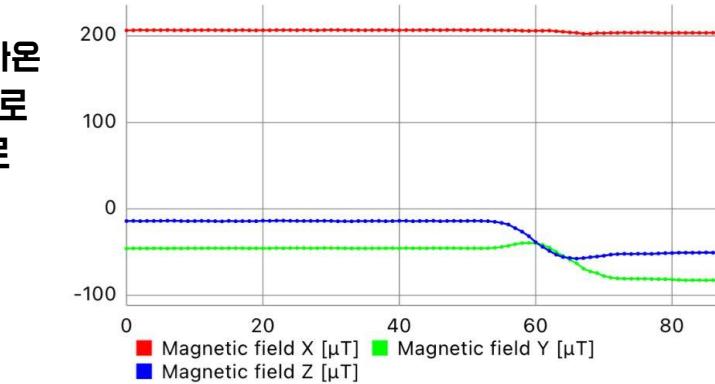
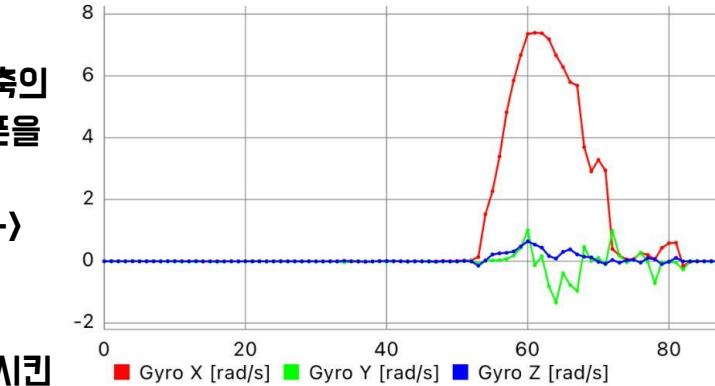
핸드폰 뒤집기 (화면을 뒤집은 상태에서 x축 반시계 방향으로)



핸드폰 뒤집기 (화면을 뒤집은 상태에서 y축 시계 방향으로)



- 앞의 뒤집기와 같이 기본적인 뒤집기이지만 원손잡이들을 고려하여 반시계 방향으로의 회전
- 앞의 분석과 동일하게 z축 방향으로 중력가속도가 작용함
- (휴대폰 전면기준) x축의 양의 방향으로 휴대폰을 뒤집어서 x축의 가속도가 (+) → (-) → 0으로 변화
- 시계 방향으로 회전 시킨 것과는 다르게 y의 각각속도가 음수가 나온 이유는 y축을 주축으로 하여 반시계 방향으로 회전 시켰기 때문



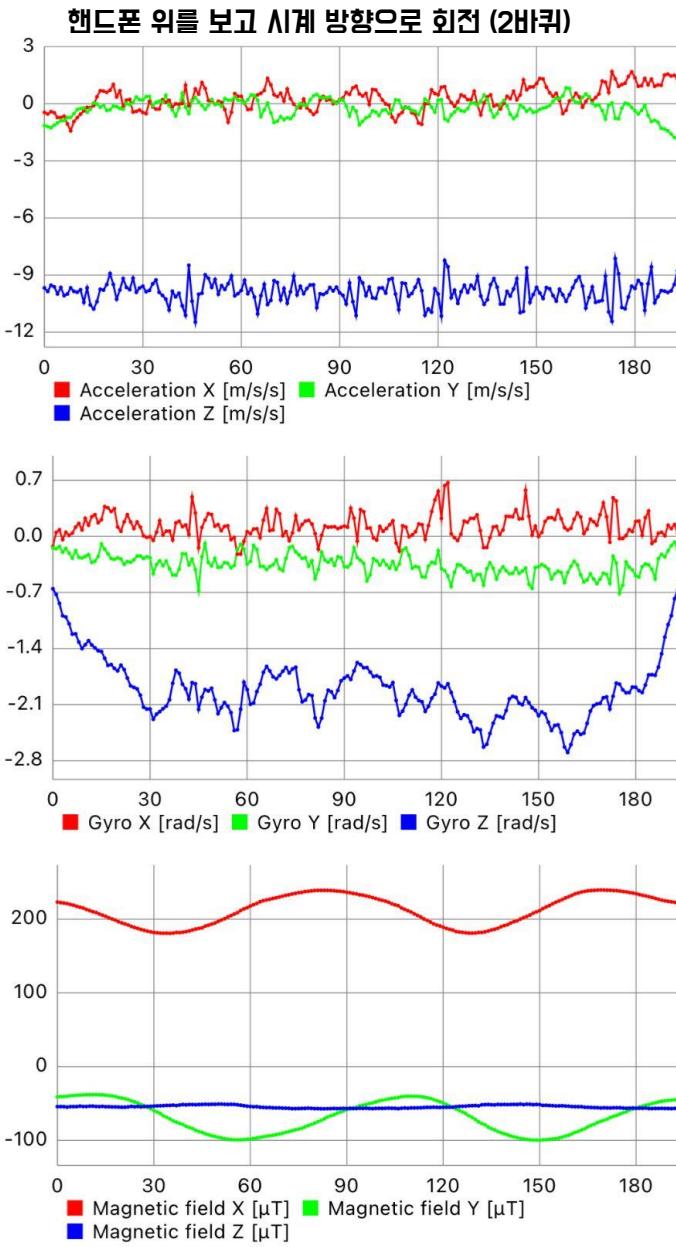
- 일반적인 뒤집기와는 차별점을 두는 세로 방향 (y축의 (+)방향)으로 시계 방향 뒤집기를 진행하는 센서 데이터

- y축 시계 방향으로 회전하므로 기준에는 x 가속도의 변동성이 보였다면 해당 데이터를 y축으로 보다 더 큰 크기의 변동성을 보여줌
→ y축으로 회전하는 것의 회전 반경이 더 커서

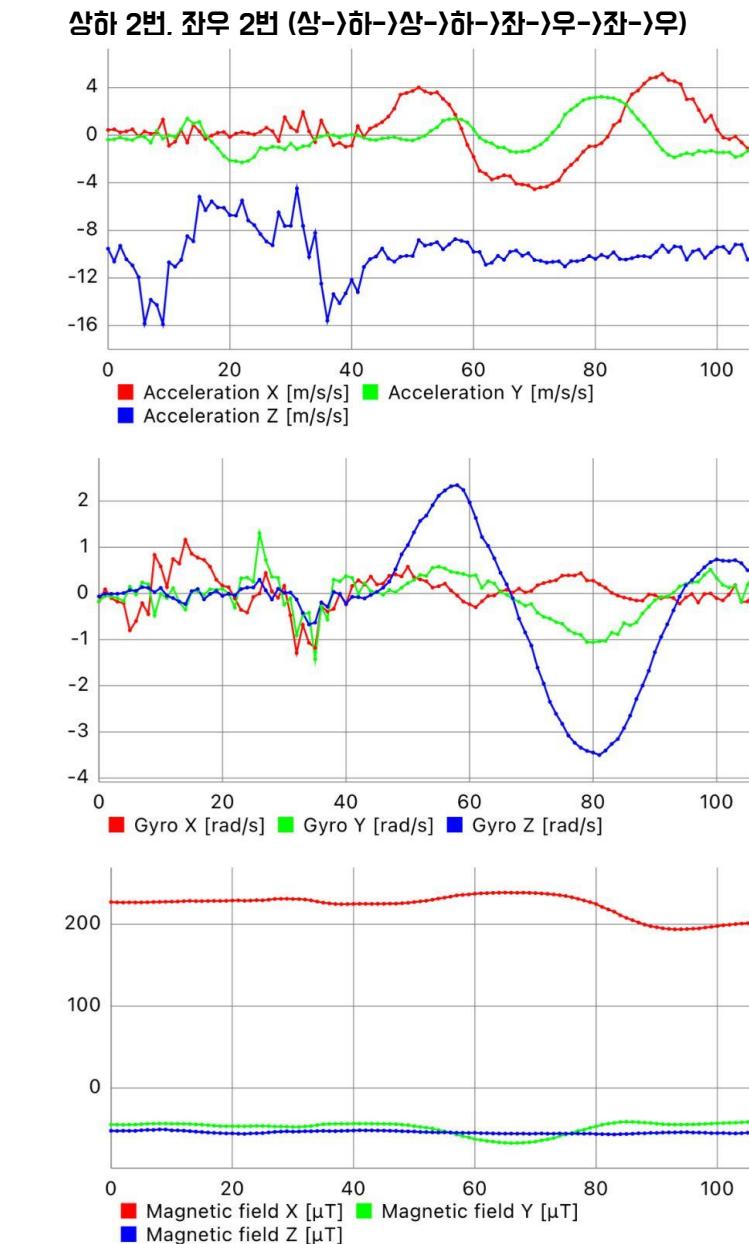
- 각속도는 앞과 비슷하지만 회전 방향이 완전 달라졌기 때문에 기준의 y 각각속도 변동에서 x 각각속도 변동으로 바뀜

이를 통해 핸드폰을 뒤집을 때 나타나는 센서 데이터들의 공통점을 발견
→ 매우 큰 z 가속도 변동성 & 축에 따라 다른 매우 큰 y(x) 각속도 변동성

● Sensor Data를 이해하기 위해 직접 센서 데이터 실험 진행



- 핸드폰이 위를 보고 있음 → 가속도에서의 Z는 음수 (주로 -10의 값)을 가짐
- 시계 방향으로 회전 시 핸드폰의 x축의 많은 변동 → x 가속도의 많은 변동성
- 회전시 핸드폰의 수평 방향의 많은 회전력 → z 각속도의 많은 변동성
- 회전 시작 및 회전 종료 시의 마무리가 유사함
- 자기장의 규칙성도 포착이 됨

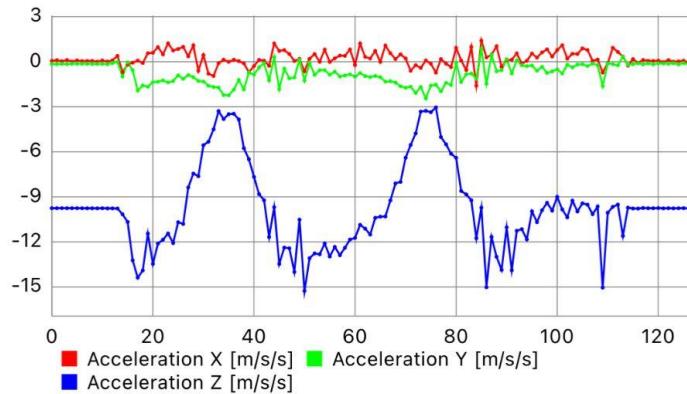


- 핸드폰이 위를 보게 하고 상하좌우로 각각 2번씩 움직임
 - > 상하 2번은 상하의 값을 담당하는 z축의 가속도의 변동성이 크게 측정 됨
 - > 좌우 2번은 좌우의 값을 담당하는 x축의 가속도의 변동성이 크게 측정 됨
 - > y축에 대한 변동성은 사람이 상하 좌우로 움직일 시에 아무리 고정하고 움직인다해도 y축에 대한 변동 (앞뒤로의 움직임)을 배제할 순 없기 때문에 존재함
- 상하로 움직일 땐. 상하의 움직임을 표현하는 x 각속도의 변동성이 크게 나타남
상하 움직임시 앞뒤로의 움직임은 자연스레 나타나기에 y 각속도의 변동성도 나타남
- 좌우로 움직일 땐. 좌우의 움직임을 표현하는 z 각속도의 변동성이 크게 나타남
 - > 주기성이 보이는데 이는 좌우로 움직일 때 좌로 갔다가 다시 돌아오고 다시 우로 움직이는 형태를 표현

해당 실험을 통해 핸드폰의 운동방향(상하/좌우)에 따라 양상이 달라지고 변하는 값들이 확실히 다르다는 것을 파악함

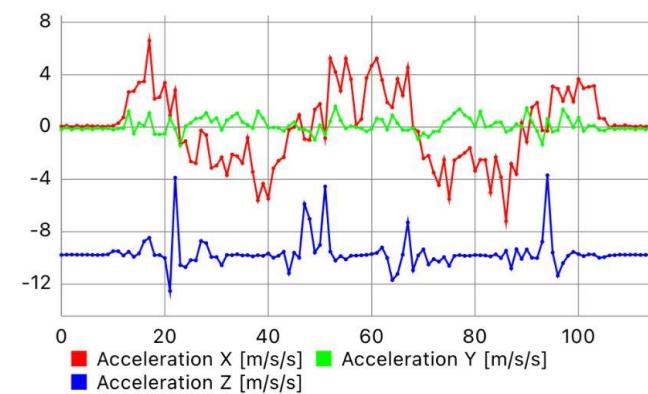
● Sensor Data를 이해하기 위해 직접 센서 데이터 실험 진행

상하 2번 (상 → 하 → 상 → 하)

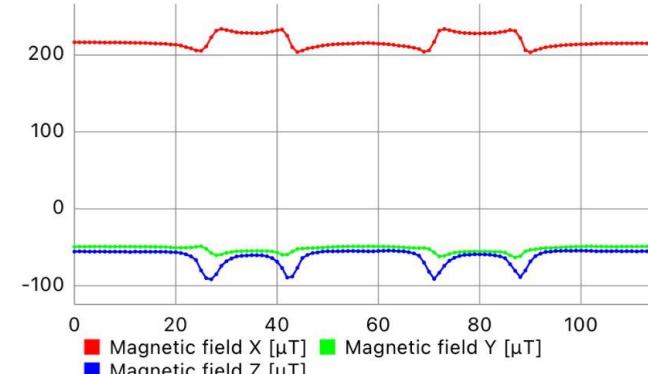
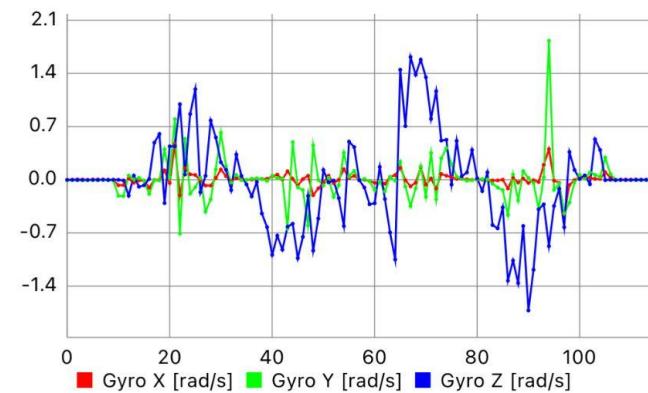


- 핸드폰의 축 방향에 대해 자세히 알기 위해 진행
- 상하 방향은 역시 z
가속도의 변동성이 가장 큼
- 상하 방향을 가장 잘 표현하는 각속도는 x각속도로 상하의 움직임을 가장 잘 포착함
→ 상하의 움직임을 규칙적으로 잘 파악 (하늘색 상자로 표시)
- 상하의 설명축 :
가속도 : z
각속도 : x

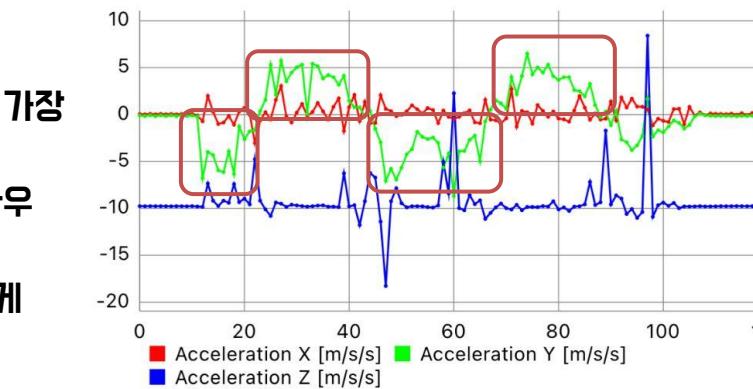
좌우 2번 (좌 → 우 → 좌 → 우)



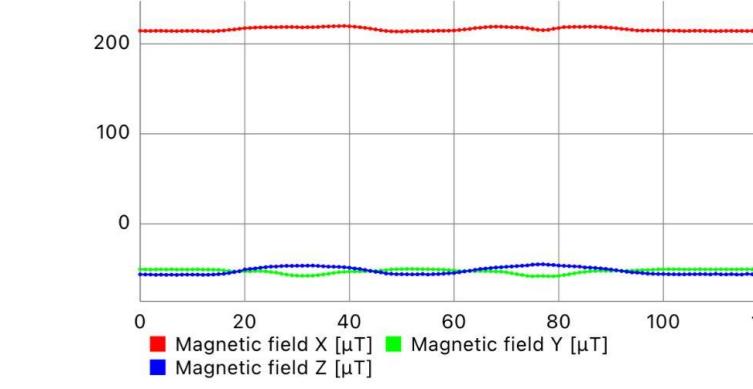
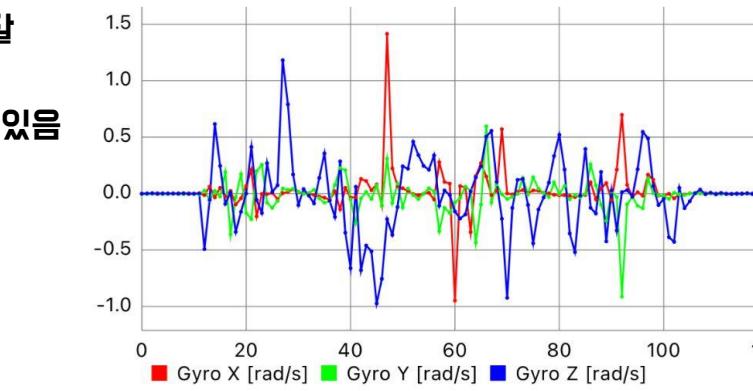
- 좌우 방향 역시 x
가속도의 변동성이 가장 큼
→ z의 변동성은 좌우 움직임 시 상하의 움직임은 불가피하게 발생
- 좌우 방향을 가장 잘 표현하는 각속도는 z각속도임을 알 수 있음
- 좌우의 설명축 :
가속도 : x
각속도 : z



앞뒤 2번 (앞 → 뒤 → 앞 → 뒤)



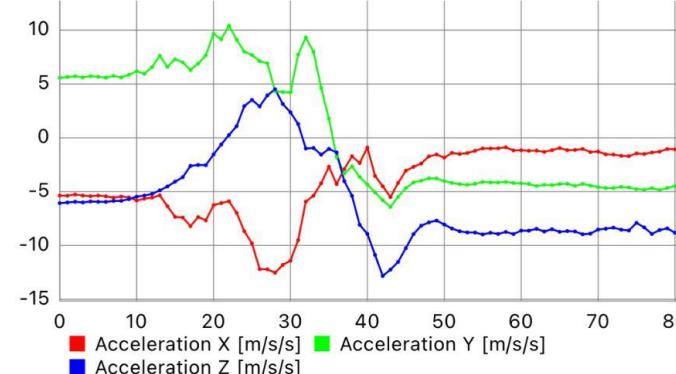
- 앞뒤 방향은 남은 축인 y가속도가 주기성을 가지면서 설명하는 것을 알 수 있음
→ 좌우와 마찬가지로 상하의 움직임을 완전히 통제할 수 없음
- 앞뒤 운동을 가장 잘 표현하는 각속도는 특정하기가 어려움
→ 앞뒤로 움직이는 부분에서 규칙성 및 주기성을 보이는 축이 없음 (가속도 부분의 빨간색 상자로 표시)
- 앞뒤 설명축 :
가속도 : y
각속도 : 무의미



각각의 행동들마다 특징적으로 바뀌는 값들이 존재 → 타겟으로 하는 행동을 잘 설명하는 값을 찾아야함

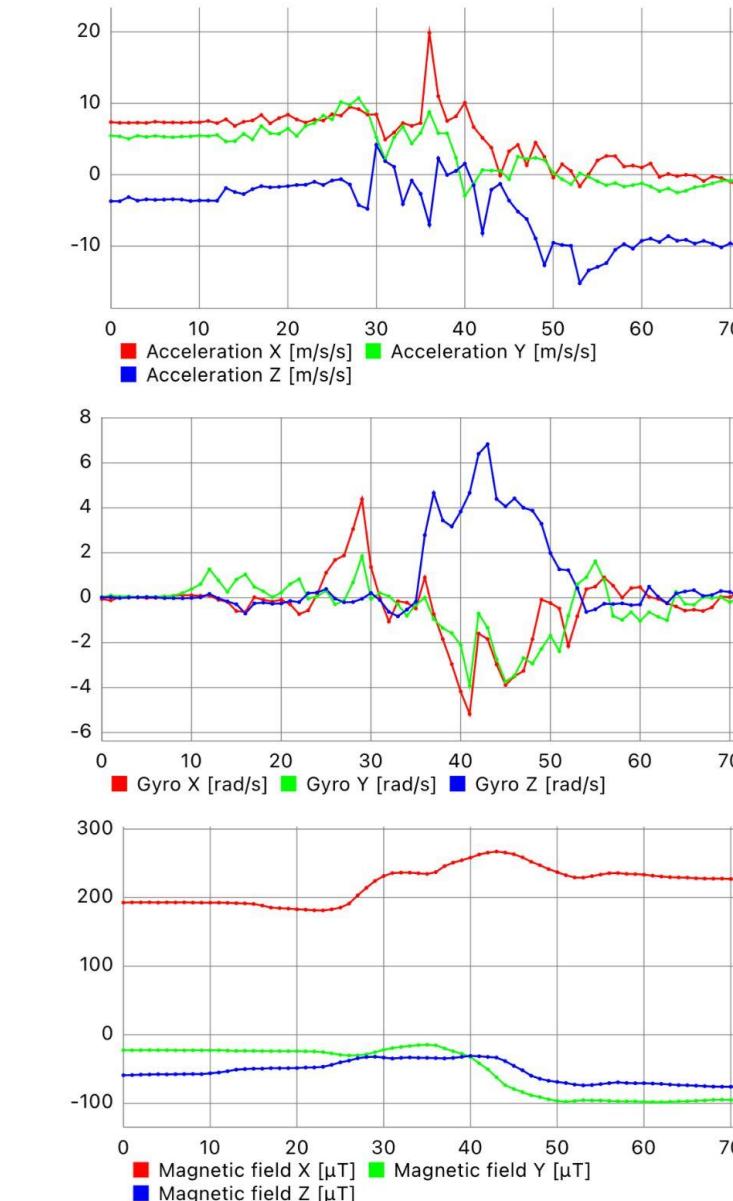
● Sensor Data를 이해하기 위해 직접 센서 데이터 실험 진행

앞에서 패딩 주머니에서 꺼내기 → 오른쪽 주머니



- 사람들이 휴대폰을 주머니에 넣어놓고 꺼내서 사용하는 자주 볼 수 있는 상황
- 휴대폰이 주머니 속에 있을 때는 비스듬히 있으므로 중력가속도가 3축에 모두 작용하는 것을 알 수 있고. 휴대폰을 꺼내는 행동을 하니 3축에 모두 가속도 변화가 큰 것을 알 수 있음
- Gyro 값을 보면 3축을 주축으로 각속도의 변화가 큰 것을 알 수 있음
- 사용자에 따라서 주머니에 놓인 휴대폰의 각도 및 위치가 다르기 때문에 3축에 적용되는 중력가속도의 값이 다름
- 사용자마다 휴대폰을 꺼내는 행동(속도와 각도)이 다르기 때문에 3축에 적용되는 가속도와 각속도. 즉 그래프의 형태가 다르게 나타남

앞에서 패딩 주머니에서 꺼내기 → 왼쪽 주머니



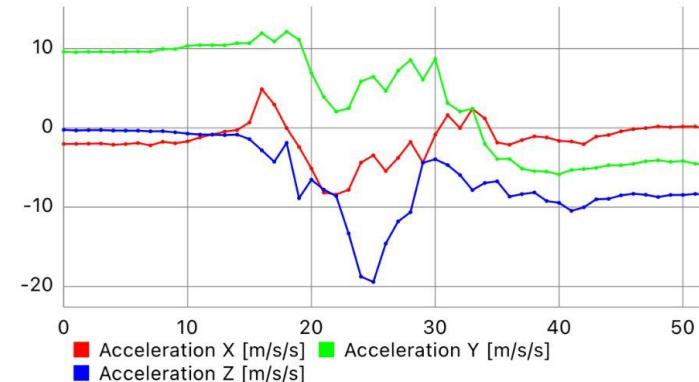
- 휴대폰을 주머니에서 꺼내는 상황은 자주 볼 수 있기에 주머니 위치를 달리하여 측정해봄
- 오른쪽 주머니와 마찬가지로 휴대폰이 주머니 속에 있을 때는 비스듬히 있으므로 중력가속도가 3축에 모두 작용하는 것을 알 수 있음
- 왼쪽 주머니로 측정하니 x축 방향으로 작용하는 중력가속도가 (+)로 됨
- 휴대폰을 꺼내는 행동을 하니 3축에 모두 가속도와 Gyro 값의 변화가 큰 것을 알 수 있음
- 사용자마다 휴대폰을 꺼내는 행동(속도와 각도)이 다르기 때문에 오른쪽 주머니 그래프 형태와 유사한 모양을 찾아볼 수 없음

주머니에서 꺼내는 행동들을 수차례 반복했으나 특징적인 분포나 데이터 형태가 보이지 않았음

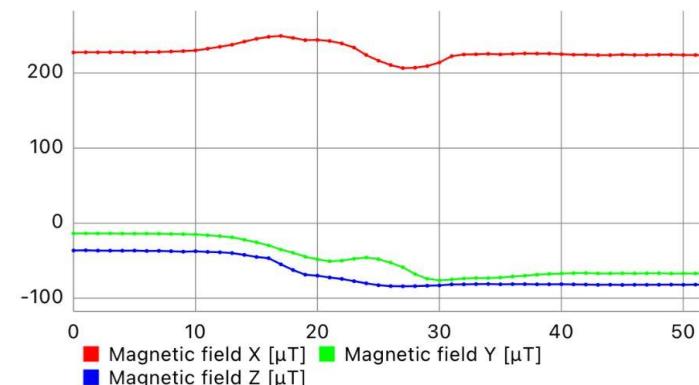
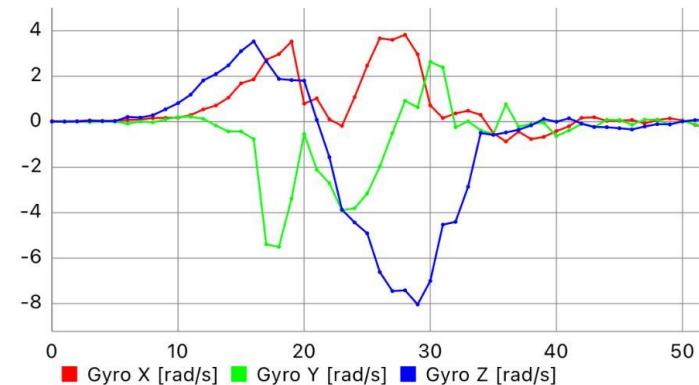
→ 모두 다른 사람들이 다른 위치에서 다른 주머니로 부터 다른 속도로 다른 핸드폰을 꺼낸다는 것을 생각하면 일반화는 어려워 보임

● Sensor Data를 이해하기 위해 직접 센서 데이터 실험 진행

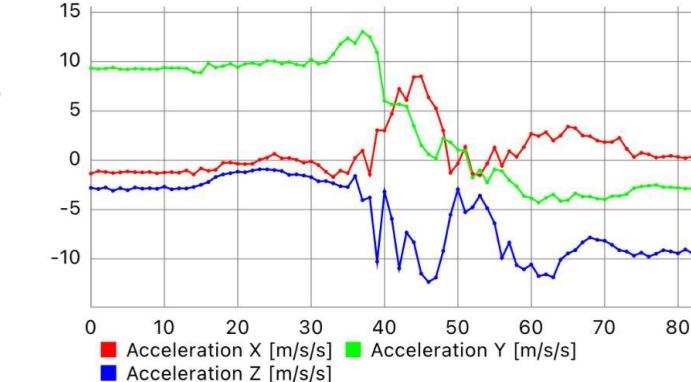
일어나서 바지 뒷주머니에서 꺼내기 -> 오른쪽 주머니



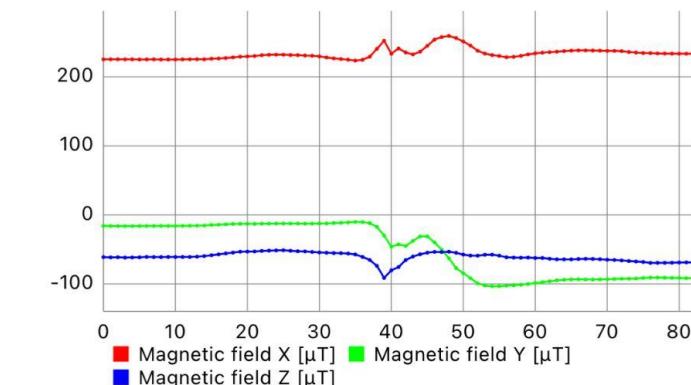
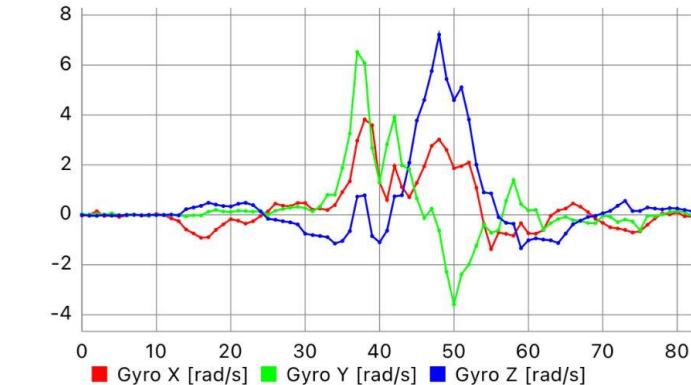
- 하의 주머니에 휴대폰을 넣어두는 경우에는 초기 가속도가 Y축의 방향으로 중력가속도만 작용하는 것을 볼 수 있음
- 휴대폰을 꺼내는 동작을 할 때는 가속도와 Gyro값이 무척 크게 움직이는 것을 볼 수 있음
- 사용자마다. 그리고 같은 사용자여도 행동(속도 및 각도)이 항상 같을 수는 없기 때문에 휴대폰을 꺼내는 행동을 할 때 3축에 적용되는 가속도와 Gyro 그래프 형태는 다를 수 있음



일어나서 바지 뒷주머니에서 꺼내기 -> 왼쪽 주머니



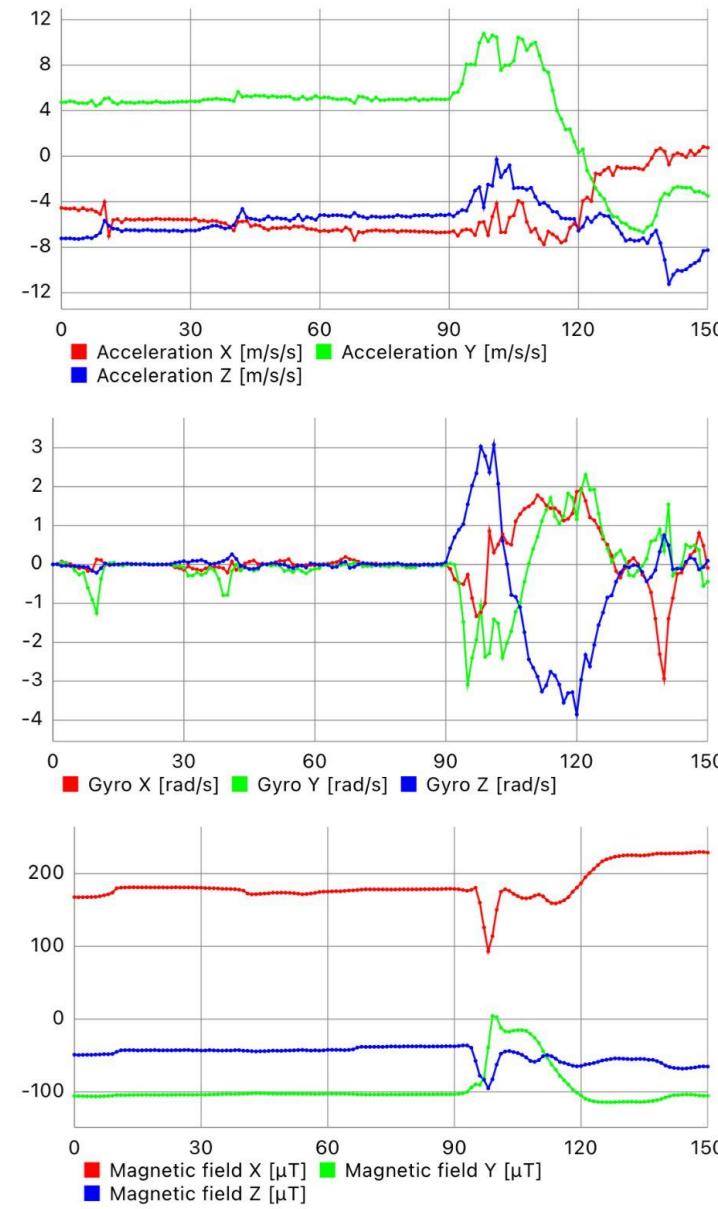
- 오른쪽 주머니와 비교해보니 Gyro Y값의 변화하는 방향이 반대인 것을 알 수 있음
(오른쪽의 경우: (-) >> (+) >> 0)
(왼쪽의 경우: (+) >> (-) >> 0)



앞슬라이드의 주머니에서 꺼내는 행동처럼 일반화가 불가능해 보임

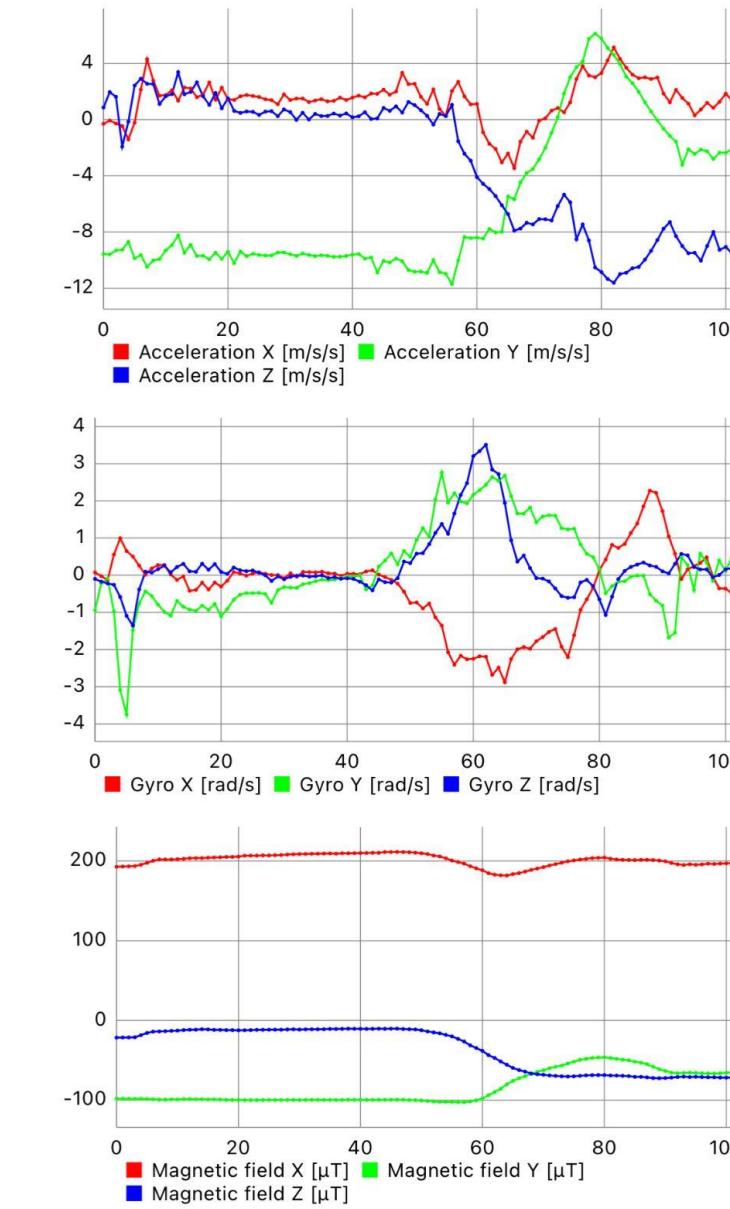
● Sensor Data를 이해하기 위해 직접 센서 데이터 실험 진행

가방에서 꺼내기



- 가방에서 핸드폰을 꺼내기 전에는 가방 속 가만히 있으니 가속도 및 각속도의 변동성이 매우 적음
→ 핸드폰을 가방에서 꺼내는 순간 가속도 및 각속도에서 변동성이 큰 것을 알 수 있음
- 가방의 위치 및 가방의 크기에 따라 센서 데이터의 값들이 달라지기 때문에 일반화 불가능

아우터 안주머니에서 꺼내기 → 원쪽 안주머니

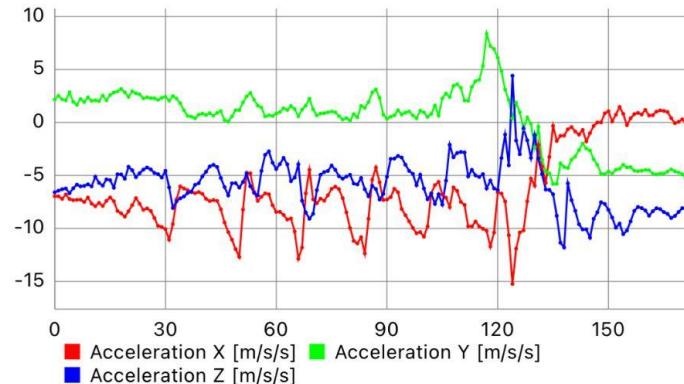


- 아우터 (특히, 패딩)의 안주머니에서 핸드폰을 꺼내는 행동의 센서 데이터 측정
→ 안주머니는 대부분 원쪽에 있기 때문에 원쪽만 측정
- 가방과 유사하게 안주머니 속에 있을 때는 변동성이 매우 적음
→ 가속도 및 각속도에서 변동성이 생기는 부분이 안주머니에서 핸드폰을 꺼내는 순간임을 알 수 있음

앞과 동일하게 변수들이 너무 다양해서 일반화가 불가능해보임

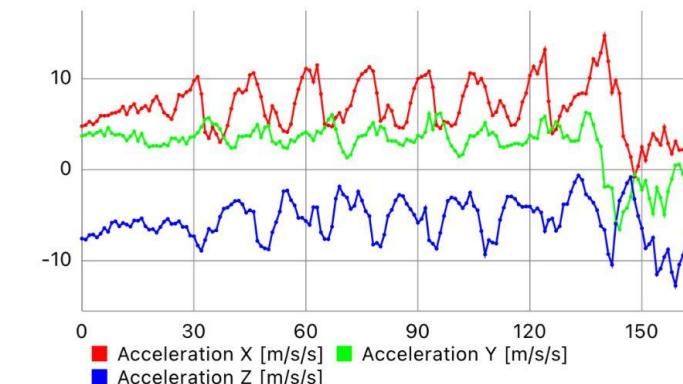
● Sensor Data를 이해하기 위해 직접 센서 데이터 실험 진행

길거리 걸으면서 주머니에서 꺼내기 -> 오른쪽 주머니



- 주머니 속 핸드폰 센서의 움직임 + 주머니에서 핸드폰을 꺼내 화면을 키는 행동의 센서 데이터
- 핸드폰을 꺼내기 전까지는 걷고 있으므로 규칙적인 형태의 그래프
-> 규칙성이 깨지는 순간이 핸드폰을 꺼내는 순간임을 알 수 있음
- 오른쪽 주머니는 화면이 시계 방향으로 회전
-> 좌우의 변동성이 많이 측정
-> z의 각속도의 변동성이 가장 큼
-> 화면을 보기 위해 뒤집으므로 z의 가속도 또한 크게 변동함

길거리 걸으면서 주머니에서 꺼내기 -> 왼쪽 주머니



- 오른쪽 주머니에서 꺼내는 데이터와 유사하지만 전체적인 그래프의 형태가 반대인 것을 알 수 있음
-> 반시계 방향으로 회전하므로 z 각속도가 크기는 비슷하지만 부호가 반대임을 알 수 있음
- 가속도에서는 똑같이 규칙성이 깨지는 순간이 핸드폰을 꺼내는 순간임을 알 수 있음

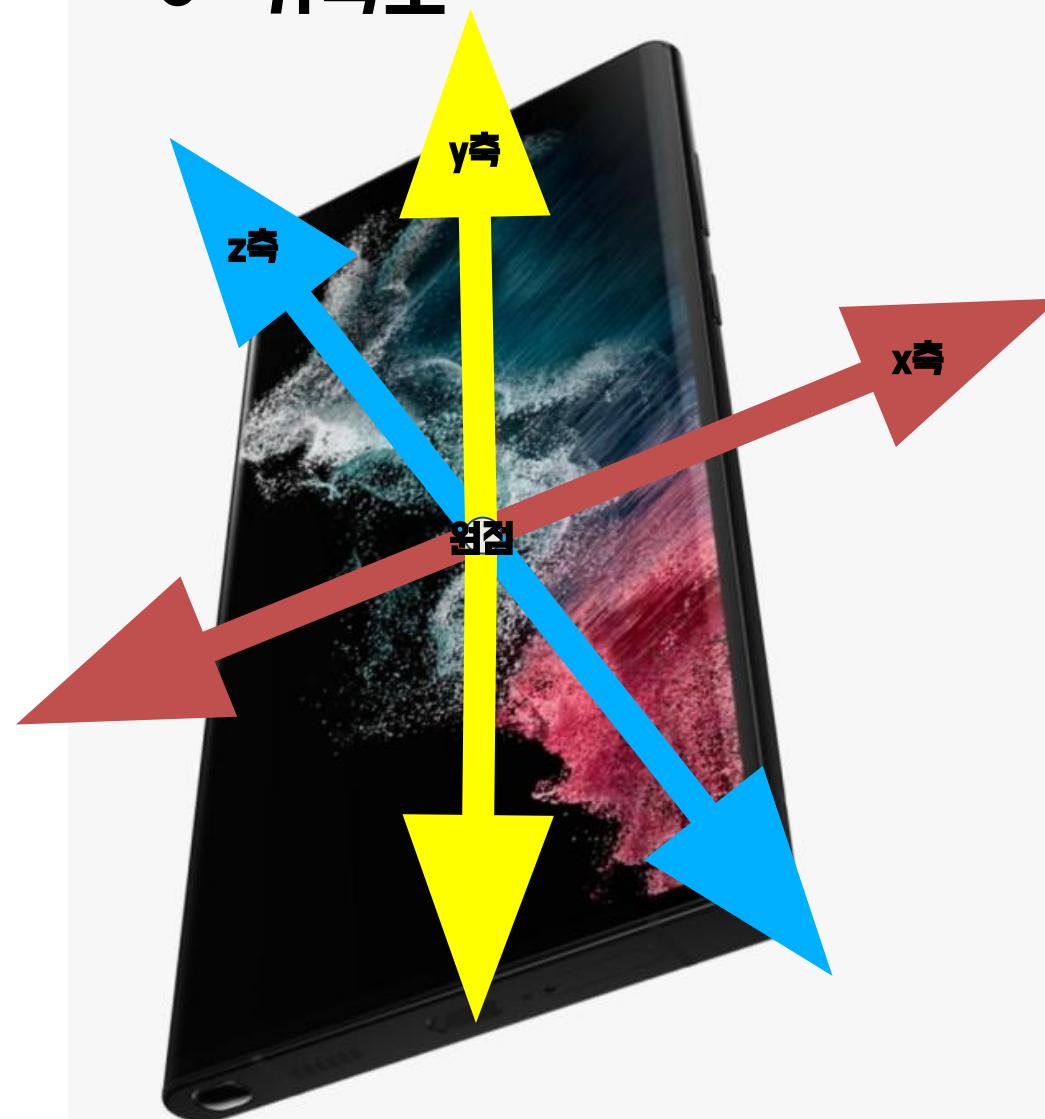
핸드폰을 주머니에서 꺼내기 전/후의 데이터 양상의 차이점이 분명함

- > 핸드폰을 꺼낸 타이밍을 파악하기엔 좋겠지만 측정자들마다 핸드폰을 꺼내는 타이밍이 다르면 모두 이상치로 측정될 수도 있음
- > 일반화가 어려워보임

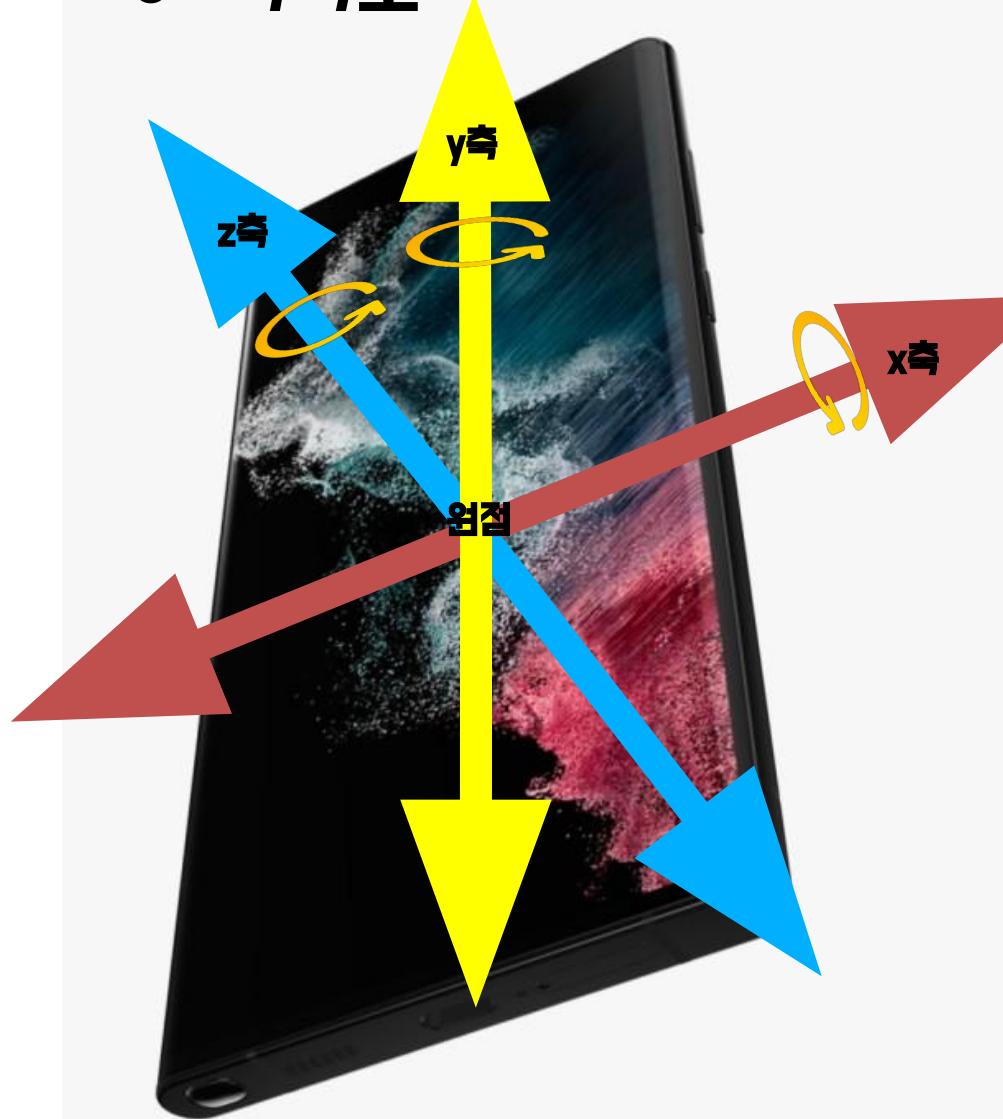
- Sensor Data를 이해하기 위해 직접 센서 데이터 실험 진행

- 실험에 대한 결과

- 가속도



- 각속도



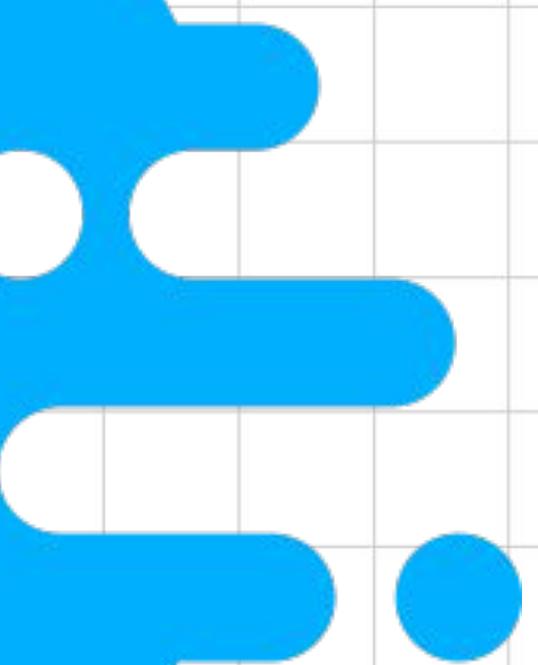
- 핸드폰을 사용자가 보고 있는지 아닌지에 대한 데이터가 중요함
 - 가속도에선 Z가 중요
 - 각속도에선 Y가 중요
 - but. 현재 Dataset에는 각속도가 없음
→ 각속도를 대체 할 다른 변수를 활용해야함
 - 각속도란, 단위 각도에 대해서 얼마나 움직였는지를 파악
→ 비슷한 역할을 하는 방향 (pitch)를 활용
방향 (pitch)은 부호(음양)으로 핸드폰의 회전 유무 및 핸드폰 화면의 방향을 표현

여러 움직임을 센서 데이터로 직접 측정하며 얻은 인사이트는 핸드폰 화면이 회전시에 나타나는 중요한 특징이 있다는 것

→ Z 가속도, pitch 방향 등이 핸드폰 회전에서 높은 설명력을 가질 것으로 생각됨

03

Preprocessing



- Dataset의 문제점에 대한 해결책 고안
- 현재 Dataset의 문제점인 너무 많은 변수들과 타겟 클래스의 불균형을 해결하기 위해 노력

너무 많은 변수들

- PCA 및 Feature Choice를 통한 변수 축소
 - PCA 구현의 어려움으로 인해 시도에서 그침
 - 실제 실험들로 인해 얻은 인사이트를 기반으로 Feature Choice 진행
→ 친적의 Feature로 train 진행

타겟 클래스의 불균형

- 타겟 클래스에 대한 oversampling과 undersampling의 필요성을 느낍
 - oversampling : positive 클래스의 수가 적으니 positive 클래스의 데이터를 oversampling을 통해 불균형을 해소
→ positive를 늘려서 negative와 비슷한 수를 가지도록 변형
 - undersampling : negative 클래스의 수가 너무 많으니 negative 클래스의 데이터를 undersampling을 통해 불균형을 해소
→ negative를 줄여서 positive와 비슷한 수를 가지도록 변형

- Dataset의 class 불균형 처리하기

- Smote와 ADASYN

- SMOTE (Synthetic Minority Over-sampling Technique) :

- SMOTE는 새로운 합성 데이터를 생성하며 oversampling을 진행
 - > 기존의 Negative 클래스 샘플 주변에 가상의 샘플을 생성하여 데이터를 확장
 - > 데이터에 대한 손실을 줄이며 데이터를 합성하면서 생성하기 때문에 과적합의 위험도 적음

- ADSYN (Adaptive Synthetic Sampling) :

- ADASYN은 SMOTE와 유사하지만, 각 샘플에 대해 합성 데이터를 생성할 때 샘플의 밀도(density)를 고려하여 더 많은 가중치를 부여하는 방식

- > 가중치를 사용하므로 SMOTE 보다 불균형 문제를 더 잘 다룰 것이라 생각함

But, 가중치를 부여하기 때문에 train에 overfitting 될 수도 있음

좋은 시도였지만 적용 후 성능 평가 결과 오히려 성능을 저하시켜 최종 아키텍쳐에서는 제외함.

```

if mode == 'train' and balance_data:
    # 데이터를 2차원으로 변환
    num_samples, num_features, num_timesteps = X.shape
    X_2d = X.reshape(num_samples, -1) # 2차원으로 변환

    # 불균형 처리 로직
    over = SMOTE(sampling_strategy=0.5) # 3
    under = RandomUnderSampler(sampling_strategy=0.8)
    pipeline = Pipeline(steps=[('o', over), ('u', under)])
    X_2d, y = pipeline.fit_resample(X_2d, y.ravel())

    # 데이터를 원래의 3차원 형태로 되돌림
    X = X_2d.reshape(-1, num_features, num_timesteps)

self.x = torch.from_numpy(X).float()

if mode == 'train':
    # 훈련 모드에서만 y의 형태를 [batch_size, 1]로 변경
    self.y = torch.from_numpy(y).float().unsqueeze(-1)

else:
    # 검증 및 테스트 모드에서는 y의 원래 형태를 유지
    self.y = torch.from_numpy(y).float()

del data

```

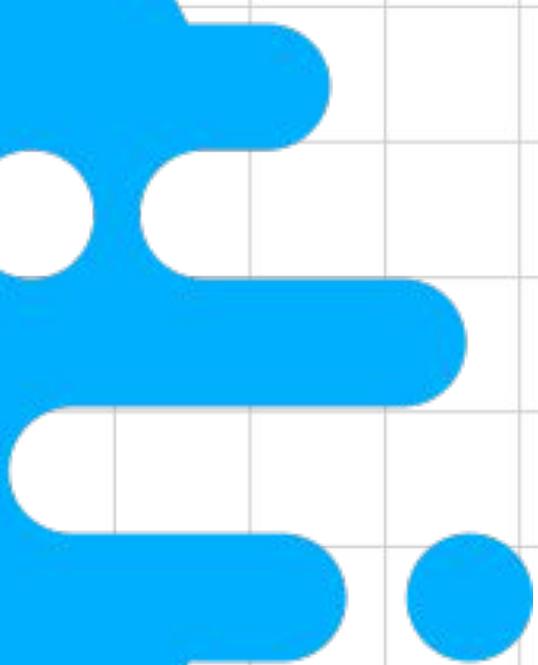
04

Feature Engineering

- ACC_x와 ACC_y의 중요성
 - 가속도 데이터셋에서 실험을 통해 얻은 인사이트인 ACC_z의 중요성 말고도 ACC_x, ACC_y에 대한 중요성도 알게 됨
 - ACC_z에서는 중력 가속도가 중요하기 때문에 ACC_z는 그대로 유지하되, ACC_x,y에서 GRVT_x,y를 뺀 x,y에 대한 순수 가속도를 활용하기로 함
 - $PURE_x = ACC_x - GRVT_x$
 - $PURE_y = ACC_y - ACC_y$
- 순수 가속도 - PURE_x,y를 만들었지만 기존의 주어진 변수들로만 돌리는 것이 성능이 더 좋게 나와서 파생변수는 추가하지 않음

05

Modeling & Tuning



- Model Choice → 여러가지 논문 및 참고 자료를 토대로 모델링 진행

- Normal CNN : 교수님께서 주신 기본 CNN 모델

- 7기존 모델에서 `cnn_hidden_list`를 조정 후 분석 진행
 - `hidden_list`를 `[16, 32] -> [32, 64], [64, 128], [128, 256]`으로 진행

- ※ `hidden_list`를 증가시키는 이유

- 더 많은 필터를 사용하면 더 세밀하고 복잡한 패턴을 파악하는 데 유용함
 - Representation Capacity가 증가함. 위 항목과 유사
 - 단. Overfitting의 Risk나 계산 비용의 증가는 감수해야 함

- Model Choice → 여러가지 논문 및 참고 자료를 토대로 모델링 진행

- Normal LSTM : 교수님께서 주신 기본 LSTM 모델

- 기존 모델에서 `LSTM_hidden_dim`와 `num_layers` 조정 후 분석 진행

※ `LSTM_hidden_dim`를 증가시키는 이유는 CNN과 유사

※ `num_layers`를 조정하는 이유

- 모델이 더욱 깊어지게 되어, 더 추상적이고 고차원적인 데이터 표현을 학습하는 데에 도움이 됨
 - LSTM의 특성 상 layer의 수가 많아지면 장기 의존성 학습에 용이함
 - 단. Overfitting의 Risk나 계산 비용의 증가는 감수해야 함

- Model Choice → 여러가지 논문 및 참고 자료를 토대로 모델링 진행

- GRU Model

- RNN 계열의 모델로 LSTM과 유사한 구조를 가지고 있는 모델

```
class GRU(BaseModel):  
    def __init__(self, input_size, seq_len, hidden_size, num_layers=1, output_size=1, dropout_p=0.0):  
        super(GRU, self).__init__()  
        self.gru = nn.GRU(input_size, hidden_size, num_layers, batch_first=True, dropout=dropout_p)  
        self.fc = nn.Linear(hidden_size, output_size)  
  
    def forward(self, x):  
        x = x.permute(0, 2, 1) # x: (batch_size, seq_len, input_size)  
        x, _ = self.gru(x) # x: (batch_size, seq_len, hidden_size)  
        x = x[:, -1, :] # x: (batch_size, hidden_size) - 마지막 time step의 output만 사용  
        x = self.fc(x)  
        return x
```

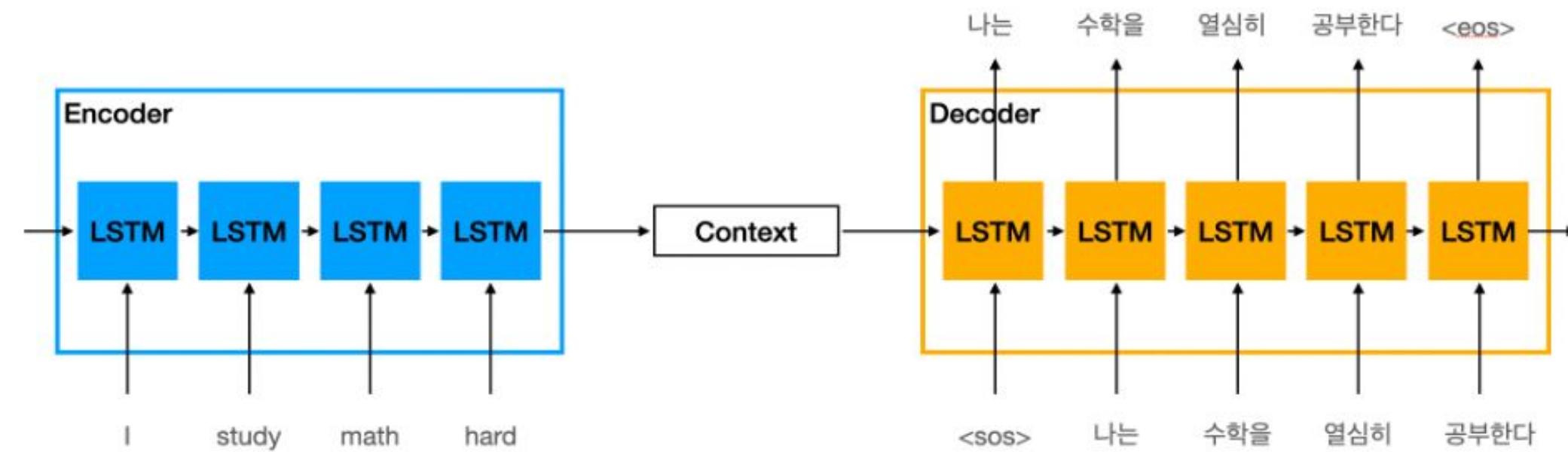
- Model Choice → 여러가지 논문 및 참고 자료를 토대로 모델링 진행

- CR2SeqWA Model

- CR2SeqWA 모델이란 Encoder, Decoder 구조를 가지는 Seq2Seq 모델로, 모델의 Encoder에는 CRNN 구조를 가지고 Decoder에는 Attention Mechanism을 적용한 모델

- Encoder-Decoder 구조란?

- Seq2Seq 와 같은 말임. 쉽게 말해 RNN 두개를 이어붙인 모델이라 생각하면 됨



- Model Choice → 여러가지 논문 및 참고 자료를 토대로 모델링 진행

- CR2SeqWA Model

- 이 모델에서 Encoder를 CNN과 GRU를 통해 구현함
- Convolution → GRU → X 출력

```
2 usages

class CRNN(nn.Module):
    def __init__(self, input_size, hidden_size, pool_size=2, dropout_p=0.2, num_layers=2):
        super(CRNN, self).__init__()
        self.conv = nn.Conv1d(input_size, hidden_size, kernel_size=3, padding=1)
        self.pool = nn.MaxPool1d(pool_size)
        self.dropout = nn.Dropout(p=dropout_p)
        self.gru = nn.GRU(hidden_size, hidden_size, num_layers=num_layers, batch_first=True)

    def forward(self, x):
        x = self.conv(x)
        x = self.dropout(x)
        x = x.permute(0, 2, 1) # 이거 해줘야 돼 원준아
        x, _ = self.gru(x)
        x = self.dropout(x)
        return x
```

- Model Choice → 여러가지 논문 및 참고 자료를 토대로 모델링 진행

- CR2SeqWA Model

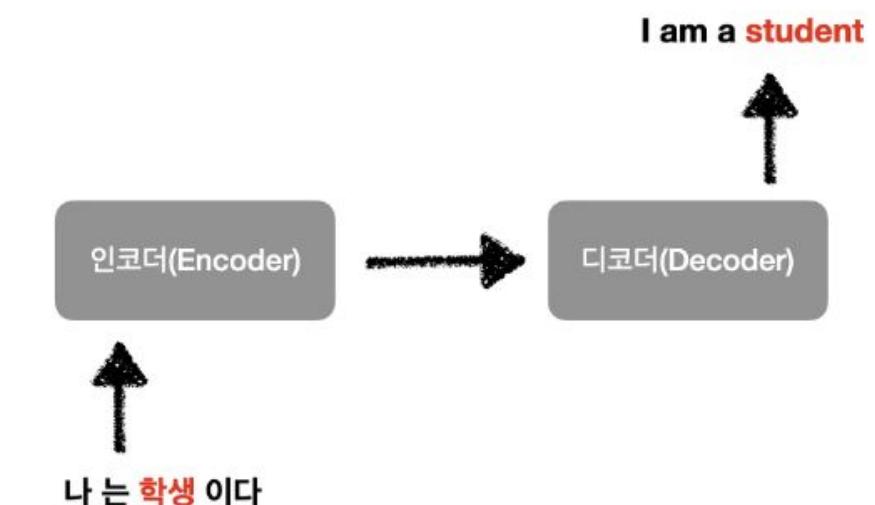
- Decoder + Attention Mechanism

- Decoding 시 encoder의 hidden state 중 어느 부분을 볼 지 학습하여 decoding을 용이하게 하는 attention mechanism을 사용함
 - Decoder에 사용된 RNN 모델은 Encoder와 동일하게 GRU를 사용함

- Attention Mechanism?

- 모델이 전체 데이터 중에서 핵심적인 요소에 '주의를 기울이게' 하여 더 효과적이고 정확한 학습 및 예측을 가능하게 하는 기술

Attention의 기본적인 컨셉은 말 그대로 중요한 부분에 더 '집중(Attention)'하자! 는 것이다.



- Model Choice → 여러가지 논문 및 참고 자료를 토대로 모델링 진행

- CR2SeqWA Model

- Decoder + Attention Mechanism
- 인코더(CRNN) → 어텐션 + 디코더(GRU)
→ 출력 로짓값

```

class CR2SeqWithAttention(BaseModel):
    def __init__(self, input_size, seq_len, hidden_size, output_size=1, gru_decoder_layers=2):
        super(CR2SeqWithAttention, self).__init__()
        self.encoder = CRNN(input_size, hidden_size)
        self.attention = Attention(hidden_size)
        self.gru_decoder = nn.GRU(hidden_size, hidden_size, num_layers=gru_decoder_layers, batch_first=True)
        self.decoder_fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        # 인코더 속삭
        encoder_outputs = self.encoder(x) # [batch size, seq_len, hidden_size]

        # 어텐션 속삭
        hidden = encoder_outputs[:, -1, :] # 마지막 hidden state를 사용
        attention_weights = self.attention(hidden, encoder_outputs)
        attention_applied = torch.bmm(attention_weights.unsqueeze(1), encoder_outputs).squeeze(1)

        # 디코더 야미
        # 디코더에 어텐션 인풋으로 속삭
        decoder_output, _ = self.gru_decoder(attention_applied.unsqueeze(1))

        x = self.decoder_fc(decoder_output.squeeze(1))
        return x

```

- Loss Tuning

- BCEwithLogitsLoss (Binary Cross Entropy with Logits Loss)

- N : 데이터 포인트의 개수
- y : 0 or 1, x : logit 상태의 출력값
- 시그모이드 함수 사용

$$\text{BCEwithLogitsLoss} = -\frac{1}{N} \sum_{i=1}^N (y_i \cdot \log(\sigma(x_i)) + (1 - y_i) \cdot \log(1 - \sigma(x_i)))$$

- 각 데이터 포인트에서의 오차 계산 후 이를 평균하여 전체 손실을 계산
- This loss combines a 'Sigmoid' layer and the 'BCELoss' in one single class.
This version is more numerically stable than using a plain 'Sigmoid' followed by a 'BCELoss' as, by combining the operations into one layer.
- we take advantage of the log-sum-exp trick for numerical stability.
 - > pytorch/torch/nn/modules/loss.py : pytorch Github repository에서 torch 속 모든 loss 함수를 정의 한 .py 파일에서의 BCEwithLogitLoss에 대한 설명
 - > BCEwithLogitLoss는 수치적 안정성을 모델에 보장하게 됨
 - > But, 데이터 클래스의 불균형을 해결하지 못함

● Loss Tuning

● FocalLoss

- BCEwithLogitLoss가 이진 변수 + 데이터셋의 수치적 안정성을 보장해주지만, 현재 데이터셋의 가장 큰 문제점인 데이터 불균형을 해결해주지 못함
- 이를 해결하기 위해 FocalLoss를 사용 (Focus On One Class Loss)
 - FocalLoss : 타겟 클래스의 불균형을 해결하기 위한 Loss
 - > 특히, 드문 클래스 (rare class)에 집중하여 모델을 학습 시킴
 - > Positive에 집중해야하지만 positive class의 양이 적다는 문제점을 해결하기 위해 사용하면 좋아 보임
 - p : 예측 확률
 - α (알파) : 가중치 파라미터 (클래스 불균형을 조절)
 - λ (람다) : Focal Loss의 강도를 조절 (손실 함수의 강도를 조절)

$$FL(p_t) = -\alpha \cdot (1 - p_t)^\gamma \cdot \log(p_t)$$

```
[Epoch 05/60] TRAIN LOSS: 0.1194, ACC: 0.7798, F1: 0.6876, PRECISION: 0.9304, RECALL: 0.5453
[Epoch 10/60] TRAIN LOSS: 0.0989, ACC: 0.7968, F1: 0.7178, PRECISION: 0.9379, RECALL: 0.5814
-----
saved/checkpoints/CNN_model-e10.pt already exists. overwrite? [y/N]y
save model: saved/checkpoints/CNN_model-e10.pt
-----
[Epoch 15/60] TRAIN LOSS: 0.0887, ACC: 0.8169, F1: 0.8113, PRECISION: 0.7485, RECALL: 0.8857
-----
saved/checkpoints/CNN_model-e20.pt already exists. overwrite? [y/N][Epoch 20/60] TRAIN LOSS: 0.0821, ACC: 0.8316, F1: 0.8287, PRECISION: 0.7561, RECALL: 0.9168
```

- > 위의 그림과 같이 precision이 높고 recall이 낮게 나오다 FocalLoss의 가중치로 인해 이후의 학습에선 precision이 낮아지고 recall이 높아짐
- > FocalLoss가 가중치 학습을 통해 precision, recall들이 높아졌다 낮아졌다를 반복하며 최적의 f1-score를 탐색

● Hyper Parameters tuning

- epoch, hidden_list, drop_out, lr, weight_decay, loss를 튜닝하며 최적의 하이퍼 파라미터를 탐색

| Model | Epoch | cnn_hidden_list | fc_hidden_list | dropout_p | lr | weight_decay | loss | F1-Score |
|------------|------------|-----------------|----------------|-----------|--------|--------------|-------------------|----------|
| NoramI_CNN | 60(10~100) | 16,32 | 128,64 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.76 |
| NoramI_CNN | 30(10~30) | 32,64 | 128,64 | 0.2 | 0.0008 | 0.0001 | BCEWithLogitsLoss | 0.7186 |
| NoramI_CNN | 60 | 256,512 | 512,256 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.7863 |
| NoramI_CNN | 30(10~60) | 64,128 | 128,64 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.7985 |
| NoramI_CNN | 30(10~30) | 256,512 | 512,256 | 0.3 | 0.0008 | 0.0001 | BCEWithLogitsLoss | 0.7616 |
| NoramI_CNN | 20(10~30) | 128,256 | 256,128 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.753 |
| NoramI_CNN | 30(10~70) | 32,64 | 128,64 | 0.3 | 0.001 | 0 | BCEWithLogitsLoss | 0.7923 |
| NoramI_CNN | 50(10~60) | 16,32 | 128,64 | 0.2 | 0.001 | 0 | FocalLoss | 0.7704 |

- 여러가지 하이퍼파라미터들을 바꿔가며 실험해 본 결과

- CNN 기본의 model에서는 대부분이 Epoch 40~50에서 가장 좋은 성능을 보임
- 앞선 설명처럼 cnn_hidden_list의 크기를 키워 학습의 양을 늘림
- 과적합이 발생하긴 하였지만 주가 데이터만큼의 과적합은 발생하지 않았음
- cnn_hidden_list를 키우는 것이 나쁘진 않았음
→ RNN에서도 hidden의 크기를 키워보자

- 빨간색 상자들을 보면 모두 같은 하이퍼파라미터에서 loss만을 BCEwithLogitsLoss에서 FocalLoss로 바꿨을 때, F1-Score가 더 좋아졌음
→ loss 튜닝이 성공적

인줄 알았으나...GRU 및 cr2seqwatt 처럼 더 좋은 성능을 내는 RNN계열 모델에서는 BCEwithLogitsLoss가 더 좋았음.....

→ 다시 한번 교수님의 위대함을 느끼며 BCEwithLogitsLoss로 최종 결정

● Hyper Parameters tuning

| Model | Epoch | cnn_hidden_list | fc_hidden_list | dropout_p | lr | weight_decay | loss | F1-Score |
|------------|------------|-----------------|----------------|-----------|--------|--------------|-------------------|----------|
| NoramI_CNN | 60(10~100) | 16,32 | 128,64 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.76 |
| NoramI_CNN | 30(10~30) | 32,64 | 128,64 | 0.2 | 0.0008 | 0.0001 | BCEWithLogitsLoss | 0.7186 |
| NoramI_CNN | 60 | 256,512 | 512,256 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.7863 |
| NoramI_CNN | 30(10~60) | 64,128 | 128,64 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.7985 |
| NoramI_CNN | 30(10~30) | 256,512 | 512,256 | 0.3 | 0.0008 | 0.0001 | BCEWithLogitsLoss | 0.7616 |
| NoramI_CNN | 20(10~30) | 128,256 | 256,128 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.753 |
| NoramI_CNN | 30(10~70) | 32,64 | 128,64 | 0.3 | 0.001 | 0 | BCEWithLogitsLoss | 0.7923 |
| NoramI_CNN | 50(10~60) | 16,32 | 128,64 | 0.2 | 0.001 | 0 | FocalLoss | 0.7704 |

| model | epoch | hidden_size | num_layers | dropout_p | lr | weight_decay | loss | F1-score |
|---------------------|-------|-------------------|----------------|-----------|--------|--------------|------------------------------------|----------|
| GRU | 50 | 32 | 2 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.7937 |
| GRU | 50 | 64 | 3 | 0.3 | 0.0007 | 0 | BCEWithLogitsLoss | 0.7125 |
| GRU | 65 | 16 | 3 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.8077 |
| GRU (Original) | 125 | 16 | 3 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.8092 |
| GRU (SOMTE + Focal) | 150 | 16 | 3 | 0.2 | 0.001 | 0 | FocalLoss | 0.7139 |
| GRU (SOMTE) | 150 | 16 | 3 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | |
| GRU (SOMTE) | 150 | 16 | 3 | 0.2 | 0.001 | 0 | FocalLoss | |
| CR2SeqWithAtt | 50 | 64 all (C,R,Att) | enc:1 / dec: 3 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.8276 |
| CR2SeqWithAtt | 60 | 64 all (C,R,Att) | enc:1 / dec: 1 | 0.2 | 0.001 | 0 | FocalLoss | 0.7928 |
| | | | | | | | -> 포카리가 더 안좋음 | |
| CR2SeqWithAtt | 55 | 64 all (C,R,Att) | enc:1 / dec: 2 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.8074 |
| CR2SeqWithAtt | 60 | 128 all (C,R,Att) | enc:1 / dec: 3 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.8078 |
| | | | | | | | -> 하든 늘렸는데 더 안좋아짐 | |
| CR2SeqWithAtt | 50 | 128 all (C,R,Att) | enc:1 / dec: 3 | 0.3 | 0.0011 | 0 | BCEWithLogitsLoss | 0.8203 |
| | | | | | | | -> 드랍 아웃이랑 lr 조절해서 오버피팅 방지하기 | |
| CR2SeqWithAtt | 55 | 128 all (C,R,Att) | enc:1 / dec: 4 | 0.35 | 0.001 | 0 | BCEWithLogitsLoss | 0.8203 |
| | | | | | | | 디코더를 한 층 쌓기 = 5에폭 추가 + 드랍 아웃 +0.05 | |
| CR2SeqWithAtt | 55 | 128 all (C,R,Att) | enc:1 / dec: 5 | 0.3 | 0.001 | 0 | BCEWithLogitsLoss | 0.8078 |
| CR2SeqWithAtt | 50 | 64/128/64 | enc:1 / dec: 3 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.8276 |
| CR2SeqWithAtt | 55 | 128/64/64 | enc:1 / dec: 3 | 0.2 | 0.001 | 0 | BCEWithLogitsLoss | 0.8358 |

| Feature Choicing | Epoch | loss | F1-Score | model은 노란색으로 표시한 model을 default로 설정하고 둘림 | original | After - Before |
|------------------|-----------|-------------------|----------|---|----------|----------------|
| NoramI_CNN | 40(10~60) | FocalLoss | 0.7136 | ACC_x, ACC_y, ACC_z, GRVT_x, GRVT_y, GRVT_z, ORT_p_cos, ORT_p_sin | 0.7704 | -0.0568 |
| NoramI_CNN | 40(10~60) | FocalLoss | 0.6436 | ACC_y, GRVT_y, ORT_p_cos, ORT_p_sin | 0.7704 | -0.1268 |
| NoramI_CNN | 40(10~60) | BCEWithLogitsLoss | 0.731 | ACC_x, ACC_y, ACC_z, GRVT_z, ORT_p_cos, ORT_p_sin | 0.7704 | -0.0394 |
| NoramI_CNN | 40(10~60) | BCEWithLogitsLoss | 0.7329 | ACC_x, ACC_y, ACC_z, GRVT_z, ORT_p_cos, ORT_r_cos | 0.7704 | -0.0375 |
| NoramI_CNN | 50(10~60) | BCEWithLogitsLoss | 0.7336 | ACC_x, ACC_y, ACC_z, GRVT_z, ORT_p_cos, ORT_p_sin, ORT_r_cos, ORT_r_sin, ORT_a_cos, ORT_a_sin | 0.7704 | -0.0368 |
| NoramI_CNN | 50(10~60) | BCEWithLogitsLoss | 0.7622 | ACC_x, ACC_y, ACC_z, GRVT_z | 0.7704 | -0.0082 |
| NoramI_CNN | 50(10~60) | BCEWithLogitsLoss | 0.6854 | ACC_x, ACC_y, ACC_z, ORT_p_sin, ORT_r_sin, ORT_a_sin | 0.7704 | -0.085 |
| NoramI_CNN | 50(10~60) | BCEWithLogitsLoss | 0.7034 | ACC_x, ACC_y, ACC_z, ORT_p_cos, ORT_r_cos, ORT_a_cos | 0.7704 | -0.067 |
| GRU | 90 | BCEWithLogitsLoss | 0.7791 | 기준 + Sin 변수들만 | 0.8092 | -0.0301 |
| GRU | 55 | BCEWithLogitsLoss | 0.7755 | 가속도만 | 0.8092 | -0.0337 |

- 여러가지 논문 및 참고 자료를 통해 다양한 모델링 진행

- 처음 접하는 센서 데이터 분석을 위해 기준 관련 논문 Benchmarking 시도

Attention-Based Convolutional and Recurrent Neural Networks for Driving Behavior Recognition Using Smartphone Sensor Data

<https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8784284&tag=1>

운전자의 smart phone sensor 데이터를 활용하여 운전자들의 운전 스타일을 예측

- Model Architecture :
 - Input : **sensor data** → CNN → **extract features map**
 - RNN, Attention → time information mining → **attention unit learn weight**
 - Recurrent layer → predict classification
- Model Architecture :
 - Min-Max normalization - First
 - Z-Score standardization - Second
- CNN
 - Convolutional layer :
 - input : sensor data segments / output : feature map → abstract transformations of image
 - padding, stride → **파로 정함**
 - Activation function :
 - ReLU
 - Pooling layer
 - max-pooling
- Attention based recurrent layer
 - RNN units : **LSTM's forget gate, input gate, GRU's update gate** → purpose-build memory cells to store information

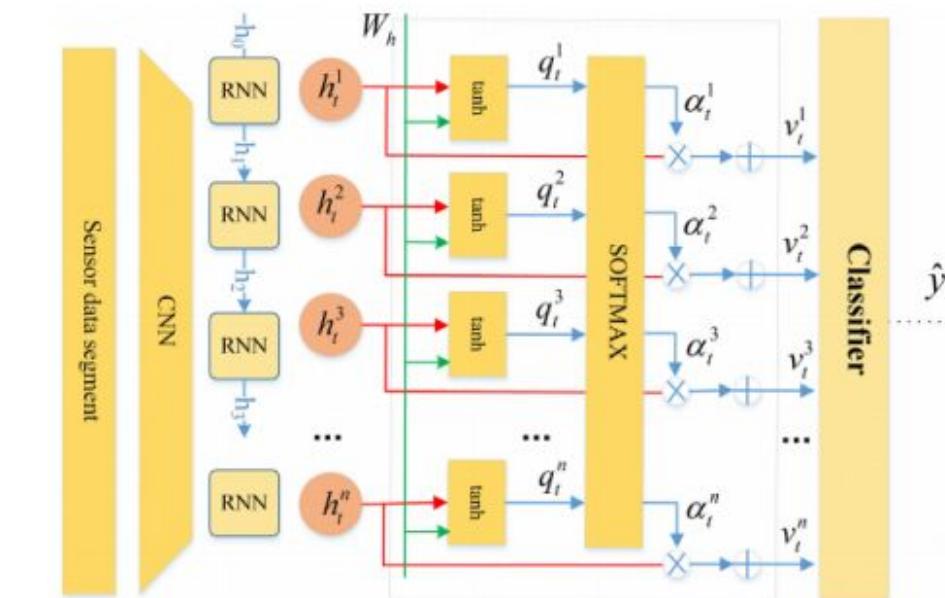


FIGURE 9. Attention-based Recurrent Neural Network.

- Optimization
 - objective : cross-entropy cost function
 - optimizer : SGD (mini-batches → size : 100), Adam
 - L2-norm, dropout
 - lr : 0.001
- model setting :
 - DeepConvGRU(LSTM)-Attention
 - CNN : 2 depth Concolutional layers, pooling layer
 - GRU(LSTM) : 2 attention-based layers
- Evaluation metrics :
 - weighted F1-score

- 여러가지 논문 및 참고 자료를 통해 다양한 모델링 진행
 - 처음 접하는 센서 데이터 분석을 위해 기존 관련 논문 Benchmarking 시도

■ DANA: Dimension-Adaptive Neural Architecture for Multivariate Sensor Data

<https://arxiv.org/pdf/2008.02397.pdf>

모바일 및 웨어러블 기기의 내장된 모션 센서를 활용하여 사용자의 신체 움직임과 생리적 상태의 시간적 변화를 추론

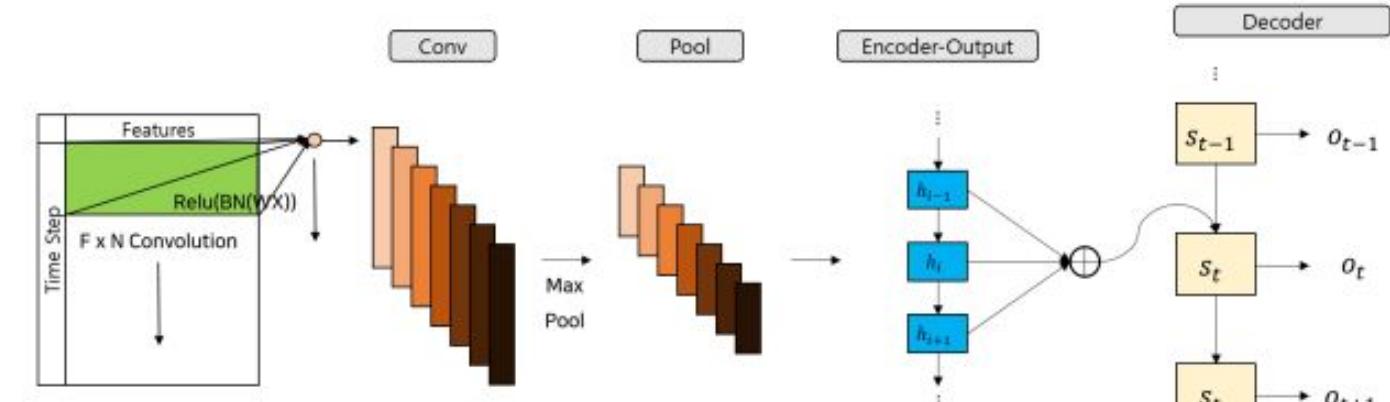
- > DANA (Dimension-Adaptive Neural Architecture)을 고안
- > 동적인 상황에서 신뢰성을 보장하기 위해 **DAP(Dimension-Adaptive Pooling)**, **DAT(Dimension-Adaptive Training)** 절차 도입
- > DAP : 차원 변화에 유연, pooling 필터의 크기를 사각형으로 제한하지 않고 다양한 모양의 필터를 사용
- DAT : 다양한 차원의 무작위 데이터 사용, 누적된 그래디언트를 활용한 최적화 -> 최적화의 효율성 증가 및 센서 데이터의 변화에 대응

- 여러가지 논문 및 참고 자료를 통해 다양한 모델링 진행
 - 처음 접하는 센서 데이터 분석을 위해 기존 관련 논문 Benchmarking 시도

■ Prediction of Solar Energy Generation Sequence Using Convolutional Recurrent Neural Network Encoder-Decoder

<https://s-space.snu.ac.kr/bitstream/10371/141434/1/000000149367.pdf>

- > 태양광 에너지 관련 데이터와 기상 예보 데이터를 input으로 하여 T 시점 이후 K 스텝 동안의 태양광 발전량 시퀀스를 예측하는 모델 - "cr2seqwatt"
- > CRNN (CNN + RNN) + encoder-decoder 모델을 제안



1. Convolutional Recurrent Neural Network (CRNN) Encoder :

입력 데이터로부터 필요한 특징을 추출하고 순서 정보를 유지하기 위해 설계

Convolutional layer와 max pooling layer를 사용하여 데이터의 특징을 추출

-> 그 결과를 Gated Recurrent Unit (GRU)의 입력으로 전달하여 순서 정보를 유지하며 학습

-> 이번 프로젝트의 핸드폰 센서 데이터의 순서와 특징들을 유지하면서 encoder 내용을 생성할 수 있음

2. Decoder & Attention Mechanism :

예측 성능을 높이기 위해 CRNN Encoder의 hidden state 중 어느 부분을 주목할지를 학습하여 decoding을 용이하게 함

-> Decoder는 attention mechanism을 기반으로 설계 (집중을 위해)

-> Encoder의 hidden state를 사용하여 context vector를 생성 이를 이용해 decoding을 수행

-> decoding 단계에서 주목해야 할 encoder의 부분을 선택하는 attention weights(α)를 학습

Q & A

감사합니다.

