YONSEI UNIVERSITY

Predicting Disease Probabilities Using a Single Gait Data Type Through Deep Learning

2024.07.26 Digital Healthcare Bootcamp
8조(박상연, 배하정, 오해규, 이승준, 정희현)

Severance

목차

1. 연구 배경 – Rationale

- 정상적인 gait는 중추신경계, 말초신경계, 근골격계가 정확히 조절되고, 이들의 조화, 반사, 학습이 총체적으로 작용해야[1] 가능함

- Pathologic Gait는 이 과정의 어떠한 요소에서라도 이상이 생기면 발생할 수 있으며, 구체적으로

통증, 구조 이상, 신경계 이상

등의 원인에 의해 발생할 수 있음 [1-3]

- 이러한 이유로 보행 분석은 이미 general practice [3,4], 신경학적 질환의 감별 [5,6] 등에 사용되고 있으며 노인의학 [7,8]에서도 주목받고 있음.

- 그럼에도 gait data는 실생활에서 쉽게 측정이 가능한 만큼 더 많은 real-life potential을 가지고 있음.

1. 연구 배경 – Rationale

- 여기에 더해, gait disturbance는 QoL(quality of life)에도 악영향을 끼침.
  - Parkinson's disease에서 보이는 FOG(Freezing of Gait)가 QoL을 저해한다는 보고[9,10]
  - Stroke 환자에서 Functional Gait Assessment(FGA) 결과가 AQoL-6D(의 6개 dimension 중 dimension of independent living)와 연관이 있다는 보고[11]

- 더 나아가, gait에 대한 실생활 data에 대한 연구는 실시간으로 잘못된 gait를 탐지하는 등 운동 및 실생활에서의 건강 관리에 있어서도 응용 가능할 것으로 기대됨.

1. 연구 배경 – Rationale

- 제한된 데이터를 사용한 분석이 의료 AI에서 각광받고 있음.
  - Low-dose CT → Full-dose CT로 변환하는 연구[12]
  - 가상 data를 생성하여 AI 모델의 학습에 활용하는 연구[13]

- 기존에도 gait analysis 연구는 이뤄져 왔지만[4,14,15], 병원 밖에서 부족한 data로 condition을 예측하는 AI model에 대한 연구는 이뤄지지 않음.
  - Full body kinematics data를 사용한 deep-learning 연구[4]
  - Multimodal data를 사용한 deep-learning 연구[14]
  - Foot force data를 사용한 deep-learning 연구[15]

- 따라서, wearable device 등으로 얻은 Single point data만을 사용하는 deep-learning 연구를 통해 높은 정확성과 접근성을 지닌 model을 개발하고자 함.
  - 1-3차에 걸친 전 주기적 예방 의료에 큰 도움이 될 것으로 기대됨.

2. 연구 목표 & 목적

Our Question

실생활 device로 보행 패턴을 분석하여 원인 질환(군)을 밝힐 수 있을까?

기대 성과

- 높은 접근성으로 신경/근골격계 질환의 조기 진단
- 별도 검사 없이 신경/근골격계 질환의 회복 수준 판별

2. 연구 목표 & 목적

Our Question

실생활 device로 보행 패턴을 분석하여 원인 질환(군)을 밝힐 수 있을까?

Tasks

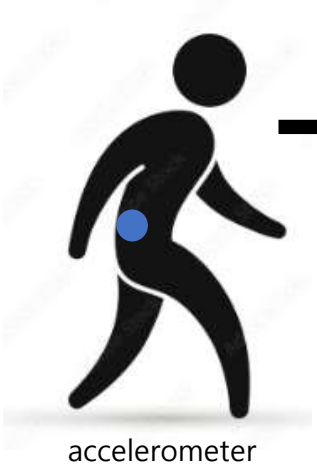
1. 3D 가속도 데이터를 사용한 Parkinson's 환자 vs normal gait 분류
2. 3D 위치 데이터를 사용한 Cerebellar Ataxia vs normal gait 분류

3. 연구 방법

연세대학교 의과대학
YONSEI UNIVERSITY COLLEGE OF MEDICINE

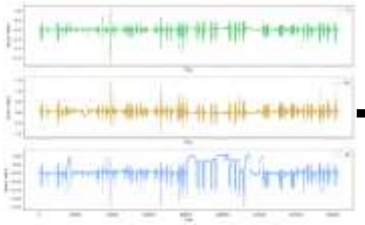
Data Acquisition

- Data from "Kaggle" Platform



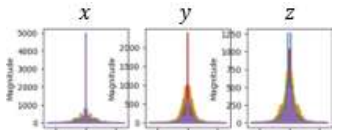
accelerometer

a_x (or x)
a_y (or y)
a_z (or z)



3D dynamics data

FFT



3D dynamics data

Data Preprocessing

Acceleration (3D)

Normal
(n=1906)

Parkinson's
(n=962)

Train/Test Data
80%/20%

Distance (3D)

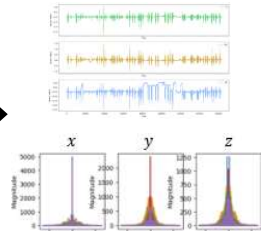
Normal
(n=100)

Cerebellar Ataxia
(n=100)

Train/Test Data
80%/20%

AI Train & Test

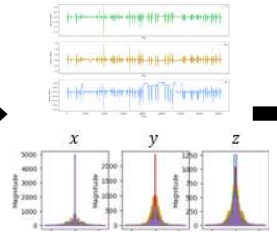
3D Acceleration Model(Normal vs Parkinson's)



LSTM
Nonlinear SVM
Efficient Net

Boosting Stacking Algorithm

3D Distance Model(normal vs cerebellar ataxia)



LSTM
Nonlinear SVM
Efficient Net

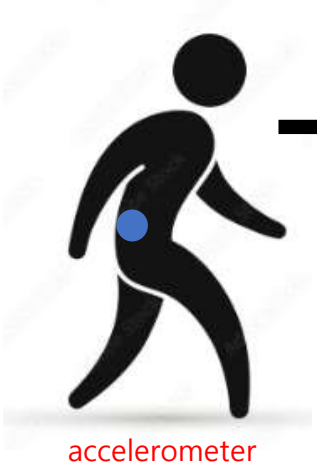
Boosting Stacking Algorithm

3. 연구 방법

연세대학교 의과대학
YONSEI UNIVERSITY COLLEGE OF MEDICINE

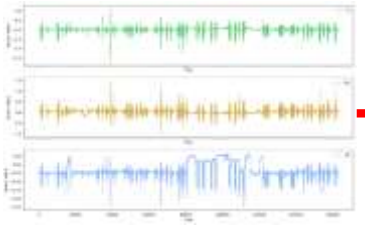
Data Acquisition

- Data from "Kaggle" Platform



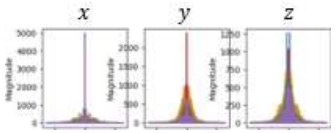
accelerometer

a_x (or x)
a_y (or y)
a_z (or z)



3D dynamics data

FFT



3D dynamics data

Data Preprocessing

Acceleration (3D)

- Normal (n=1906)
- Parkinson's (n=962)

Distance (3D)

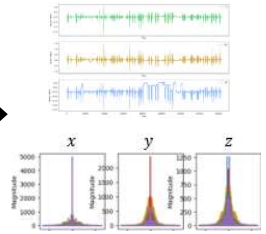
- Normal (n=100)
- Cerebellar Ataxia (n=100)

Train/Test Data
80%/20%

Train/Test Data
80%/20%

AI Train & Test

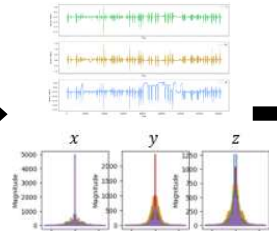
3D Acceleration Model(Normal vs Parkinson's)



LSTM
Nonlinear SVM
Efficient Net

Boosting Stacking Algorithm

3D Distance Model(normal vs cerebellar ataxia)

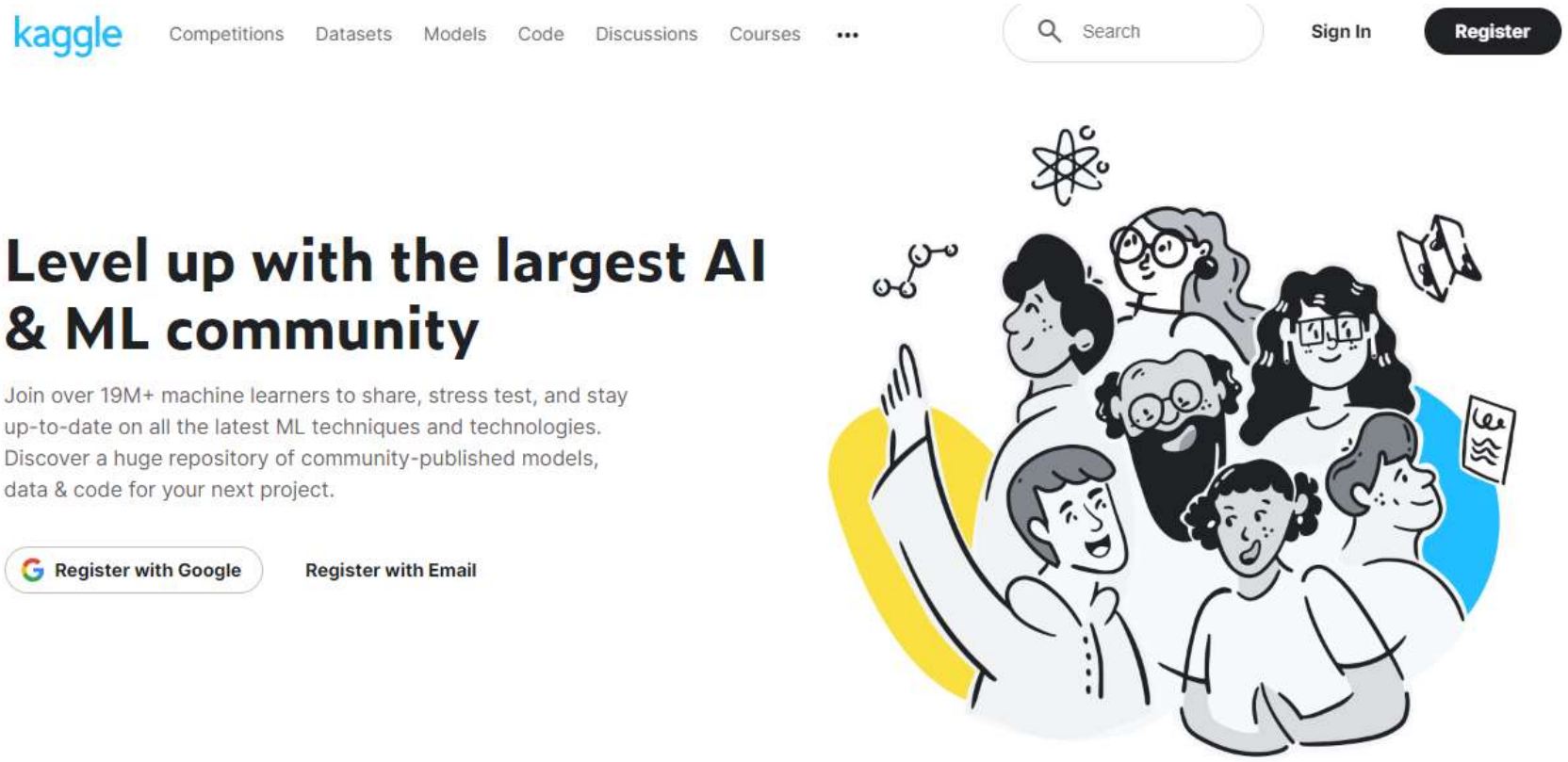


LSTM
Nonlinear SVM
Efficient Net

Boosting Stacking Algorithm

3. 연구 방법

- 학습 Data 및 Model 선택 – Kaggle 플랫폼에서 데이터 취득



3. 연구 방법

- **학습 Data 및 Model 선택 – Kaggle 플랫폼에서 데이터 취득**

데이터 비교	Normal	Parkinson's disease	Cerebellar ataxia (CA + Normal)
Data (location)	3D Acceleration (Lt. waist, Rt. thigh)	3D Acceleration (sensor @ lower back)	3D Distance(coordinate) (Lt. and Rt. shoulder, elbow, wrist, hip, knee, and ankle)
N수	93	437	100/100 (CA/normal)
Time point	1/100s (등간격 X)	1/128s (등간격 O)	1/1000s (등간격 X)
Data Length for model	약16s(2000개) 총 n = 1906	약16s(2000개) n =962	약30s (6000개) n=100/100
비고			정상인이 cerebellar ataxia 모방한 data, 비디오 분석(Sensor 사용하지 않음)

93 Human Gait (walking) Database

Human Gait Database for Normal Walk Collected by SmartPhone Accelerometer

Parkinson's Freezing of Gait Prediction

Event detection from wearable sensor data

Gait Analysis Dataset: Cerebellar Ataxia

University Student Gait Dataset: Simulated Ataxia & Normal Walking

3. 연구 방법

- **학습 Data 및 Model 선택 – Kaggle 플랫폼에서 데이터 취득**

데이터 비교	Normal	Parkinson's disease	Cerebellar ataxia (CA + Normal)
Data (location)	3D Acceleration (**Lt. waist**, Rt. thigh)	3D Acceleration (sensor @ **lower back**)	3D Distance(coordinate) (**Lt.** and Rt. shoulder, elbow, wrist, **hip**, knee, and ankle)

위치 선택 기준

1. **Single point로써 전신의 운동에 대한 종합적인 정보를 가지고 있는 신체의 중심부를 선택**
2. **어쩔 수 없이 데이터를 취합*하는 과정에서 가장 유사한 위치를 선택**

cf1. Timestamp의 길이가 dataset 내/dataset 간에 불균등한 경우 linear interpolation technique을 적용.

cf2. Data augmentation 및 input data의 형식 통일을 위해 동일한 시간 간격으로 truncation 진행.

cf3. FFT를 진행하는 과정에서 중력가속도 등의 영향을 제거하기 위해 linear regression을 사용.

- 동일한 data가 없어서 여러 source의 data를 합칠 수밖에 없었음. 그 과정에서 가장 유사한 위치에 sensor를 부착한 데이터를 취합할 수 있도록 했음.

3. 연구 방법

- **학습 Data 및 Model 선택**

위치 데이터는 suboptimal하지만,
단기적으로 가속 데이터로부터
derivation 가능

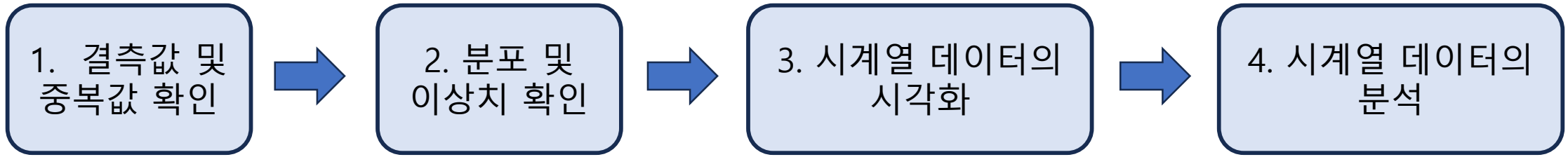
신체 중심부 1개 point의 Euclidean 3축 가속도/위치 시계열 data

Sensor가 적을수록 **실사용에서 유리**
(but, Accuracy는 떨어질 수 있음)

가속 센서는 smart device에
보편적으로 내장되어 **실사용에서 유리**

3. 연구 방법

- **EDA(Explorative Data Analysis) – clipped data**



'전체' 수준에서의 분석

Data_Pos
환자1
환자2
…
환자n

'일부' 수준에서의 분석

Time	X, Y, Z
t1	x1, y1, z1
t2	x2, y2, z2
…	
ti	xi, yi, zi

vs.

'전체' 수준에서의 분석

Data_Neg
정상1
정상2
…
정상m

'일부' 수준에서의 분석

Time	X, Y, Z
t1	x1, y1, z1
t2	x2, y2, z2
…	
tj0	xj, yj, zj

3. 연구 방법

- **EDA(Explorative Data Analysis) – clipped data**

Parkinson's Disease vs. Normal

- **1. 결측값 및 중복값 확인**
  - 결측값 없음
  - (시점 제외) 중복값 있음

Parkinson's Disease(일부) [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 7, 0, 1, 0, 0, 0, 0,

Normal(일부) [144, 112, 131, 129, 129, 129, 135, 144, 158, 0, 125, 132, 137, 242,

- **2. 분포 및 이상치 확인**
  - 기초 통계량 확인

Parkinson's Disease(일부)

	Time	AccX	AccY	AccZ
count	2000.000000	2000.000000	2000.000000	2000.000000
mean	999.500000	3.204363	0.271466	-9.048364
std	577.494589	1.895198	1.392298	1.528158
min	0.000000	-4.633833	-5.366474	-24.863996
25%	499.750000	2.502734	-0.693723	-9.725266
50%	999.500000	3.308928	0.436756	-9.043219
75%	1499.250000	4.021285	1.160875	-8.539737
max	1999.000000	14.257240	4.774288	-4.367093

Normal(일부)

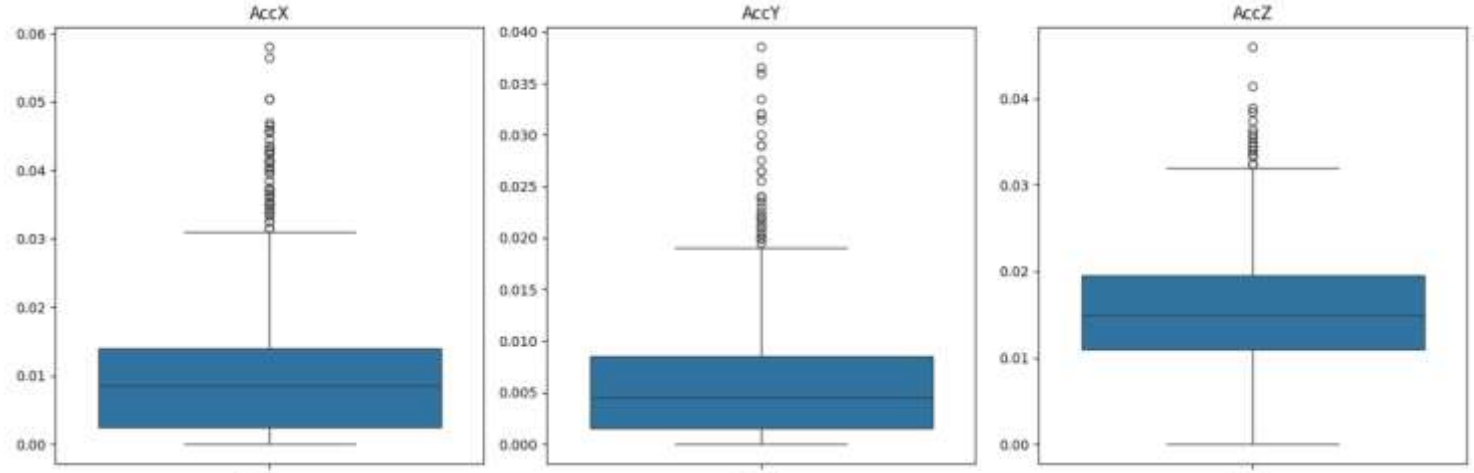
	Time	AccX	AccY	AccZ
count	2000.000000	2000.000000	2000.000000	2000.000000
mean	999.500000	0.024775	0.013695	-0.318089
std	577.494589	0.464237	0.386892	0.285804
min	0.000000	-0.957669	-1.551539	-1.621656
25%	499.750000	-0.333991	-0.162614	-0.467573
50%	999.500000	0.020320	0.049784	-0.273953
75%	1499.250000	0.397109	0.278716	-0.144301
max	1999.000000	1.351709	0.905103	0.634754

3. 연구 방법

- **EDA(Explorative Data Analysis) – clipped data**

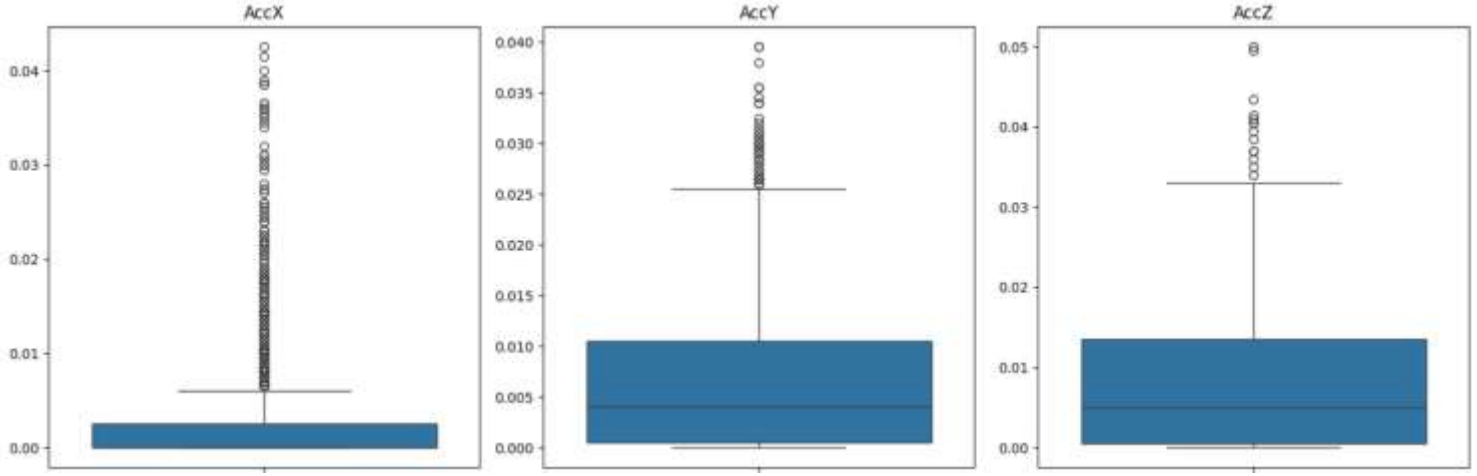
Parkinson's Disease vs. Normal

- **2. 분포 및 이상치 확인**



Parkinson's Disease(전체):

각 데이터마다 z-score를 계산하여, 각 축마다 z-score < -3 또는 z-score > 3인 시점의 비율을 구한 후, boxplot으로 시각화



Normal(전체):

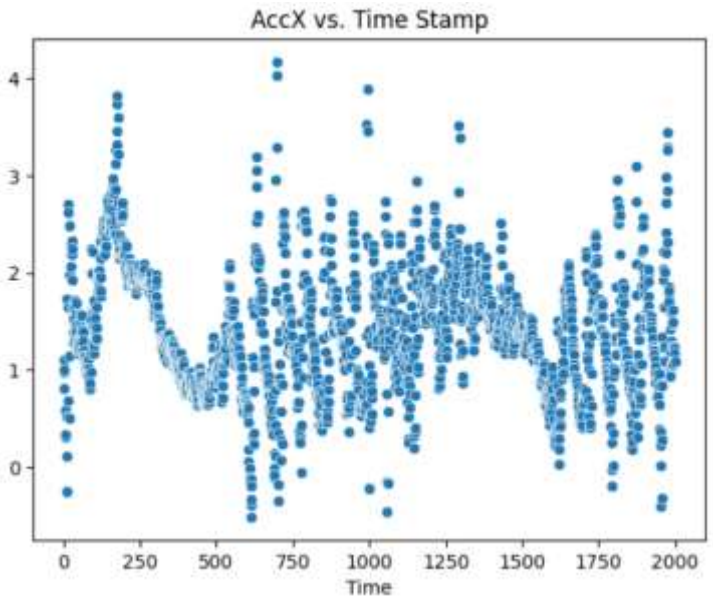
3. 연구 방법

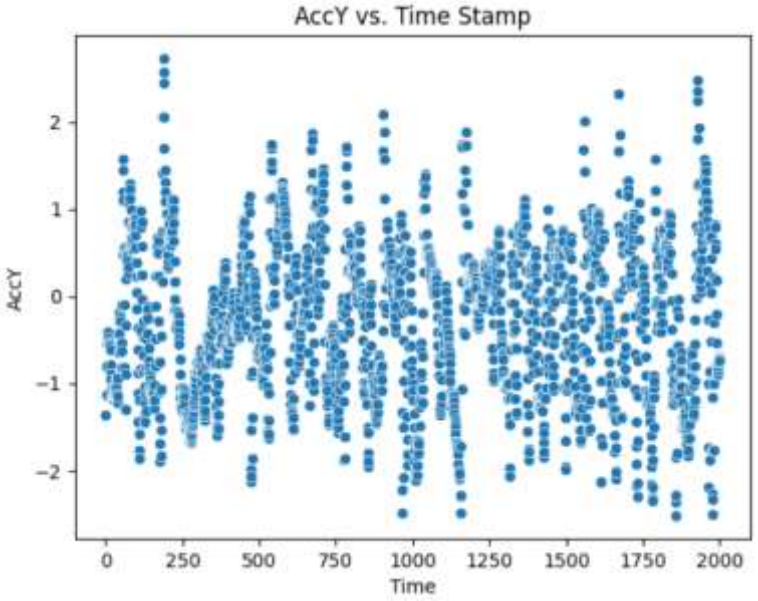
- **EDA(Explorative Data Analysis) – clipped data**

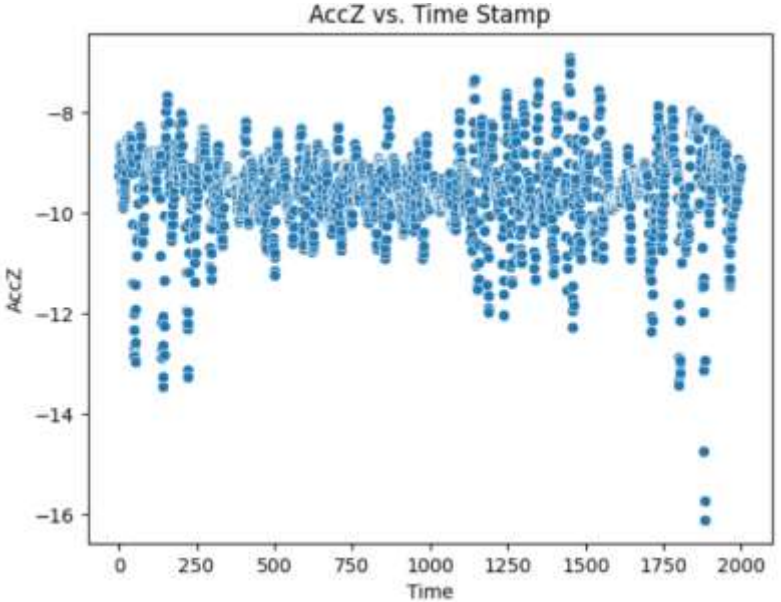
Parkinson's Disease vs. Normal

- **3. 시계열 데이터의 시각화**

Parkinson's Disease(일부):





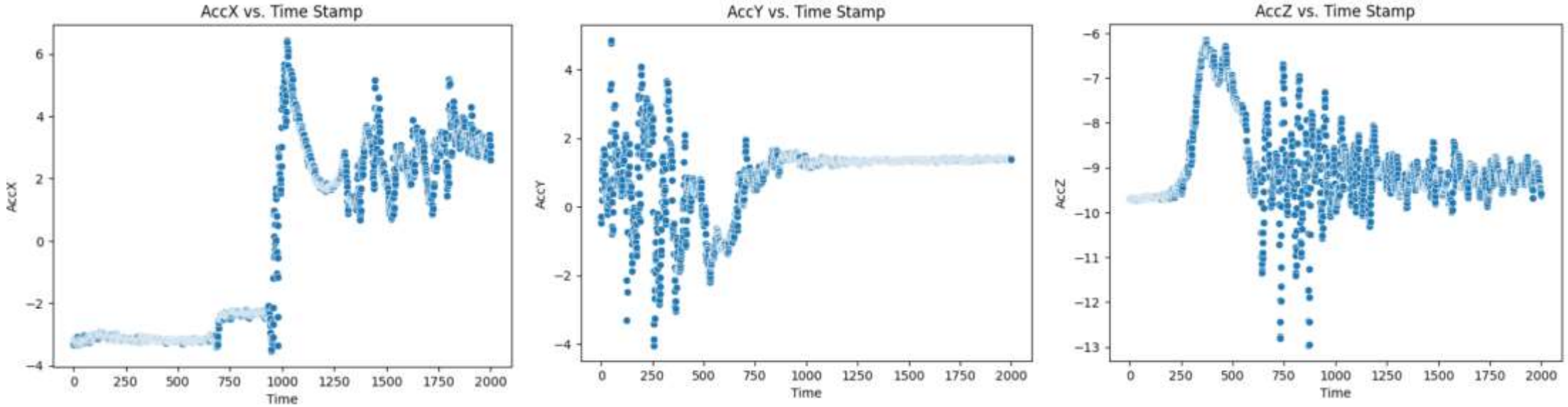


- **EDA(Explorative Data Analysis) – clipped data**

Parkinson's Disease vs. Normal

- **3. 시계열 데이터의 시각화**

Parkinson's Disease(일부):

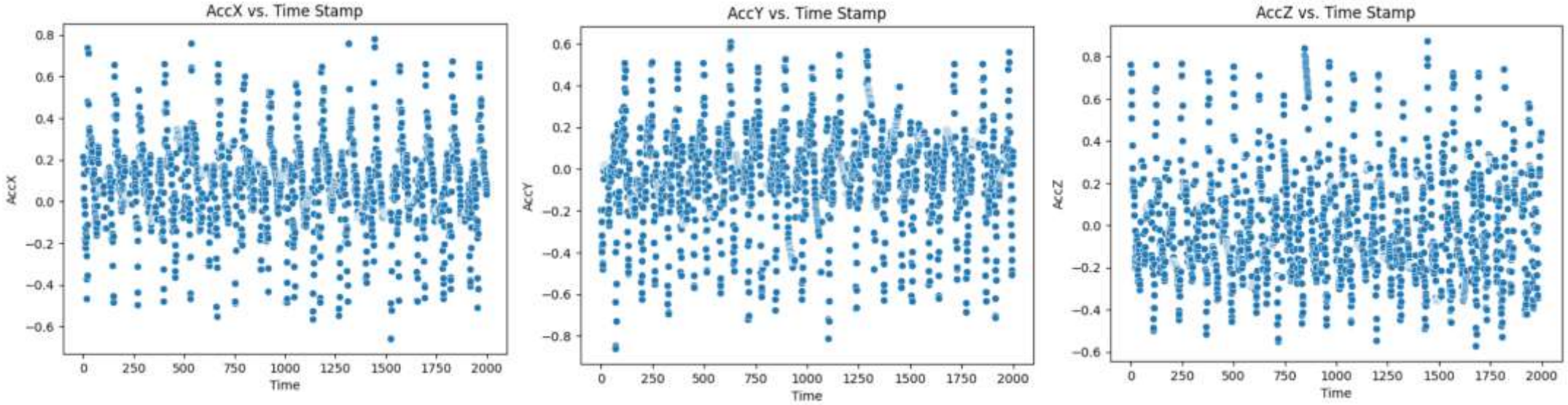


- **EDA(Explorative Data Analysis) – clipped data**

Parkinson's Disease vs. Normal

- **3. 시계열 데이터의 시각화**

Normal(일부):



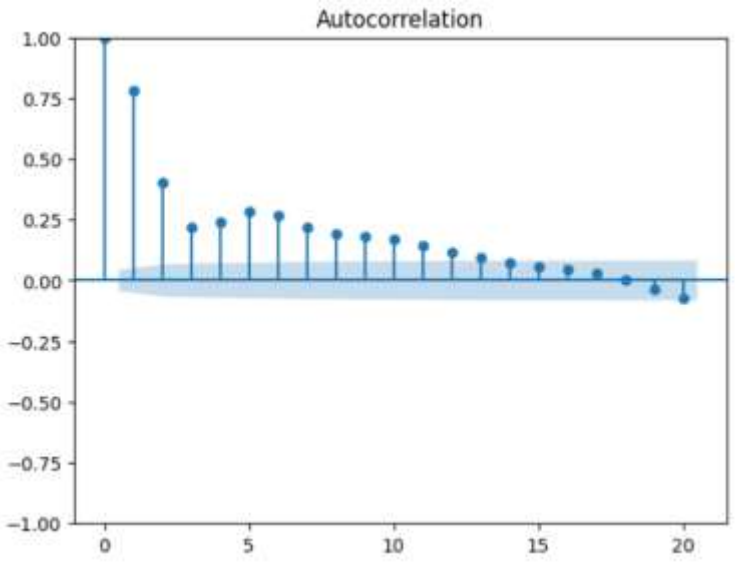
- **EDA(Explorative Data Analysis) – clipped data**

Parkinson's Disease vs. Normal

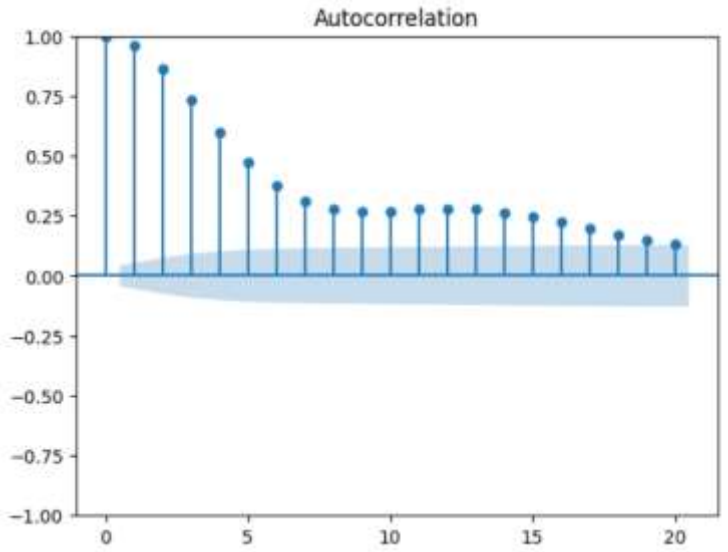
- **4. 시계열 데이터의 분석**
  - **Autocorrelation:** lag

Parkinson's Disease(일부):

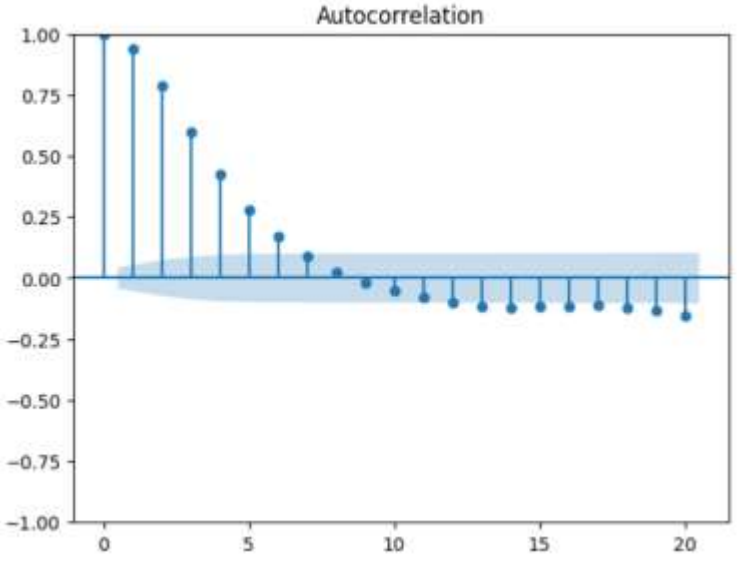
X axis



Y axis



Z axis



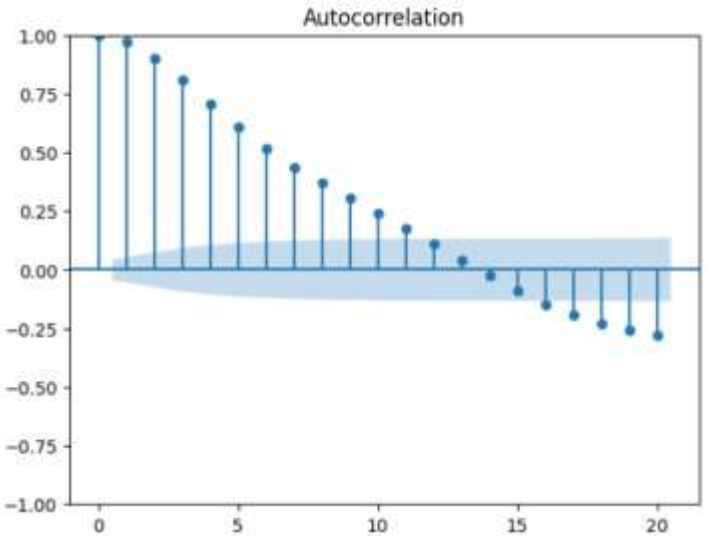
- **EDA(Explorative Data Analysis) – clipped data**

Parkinson's Disease vs. Normal

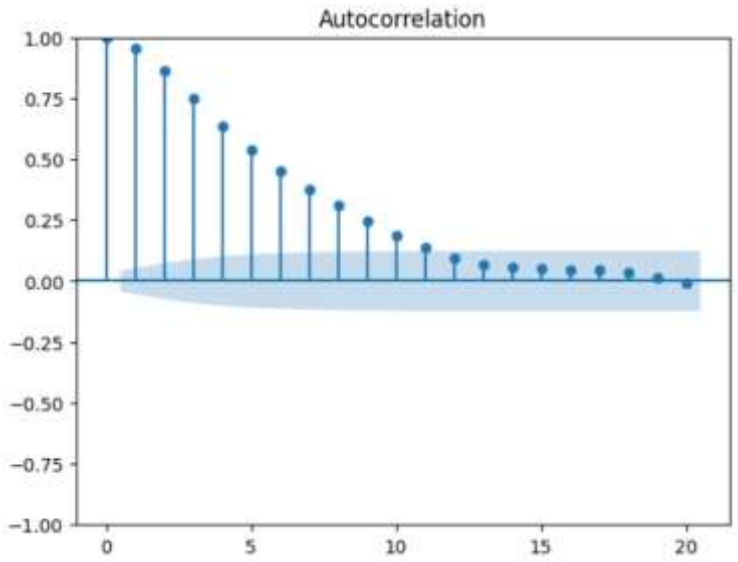
- **4. 시계열 데이터의 분석**
  - **Autocorrelation:** lag

Normal(일부):

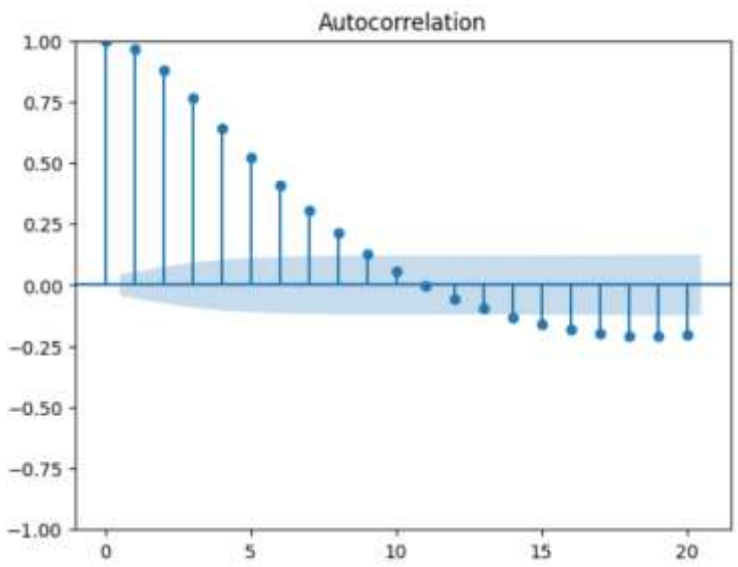
X axis



Y axis



Z axis



3. 연구 방법

- **EDA(Explorative Data Analysis) – clipped data**

Parkinson's Disease vs. Normal

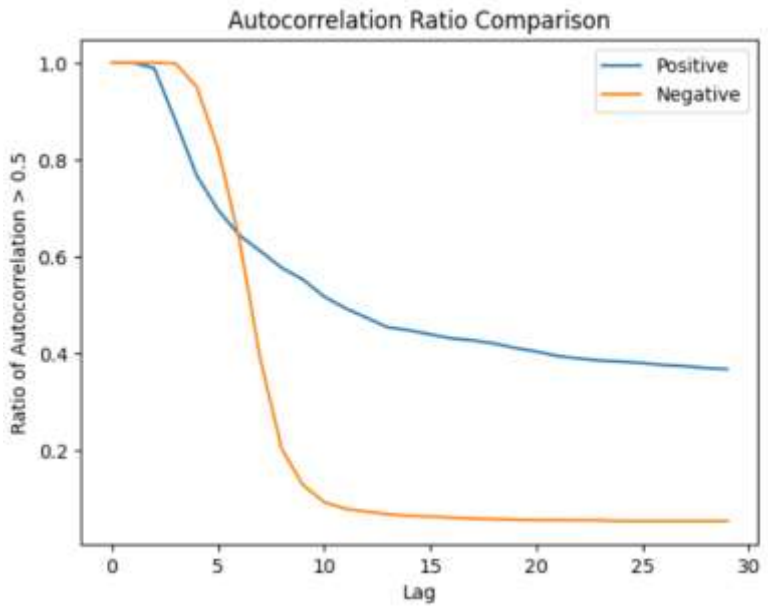
- **4. 시계열 데이터의 분석**
  - **Autocorrelation:** lag

    Parkinson's Disease(전체)

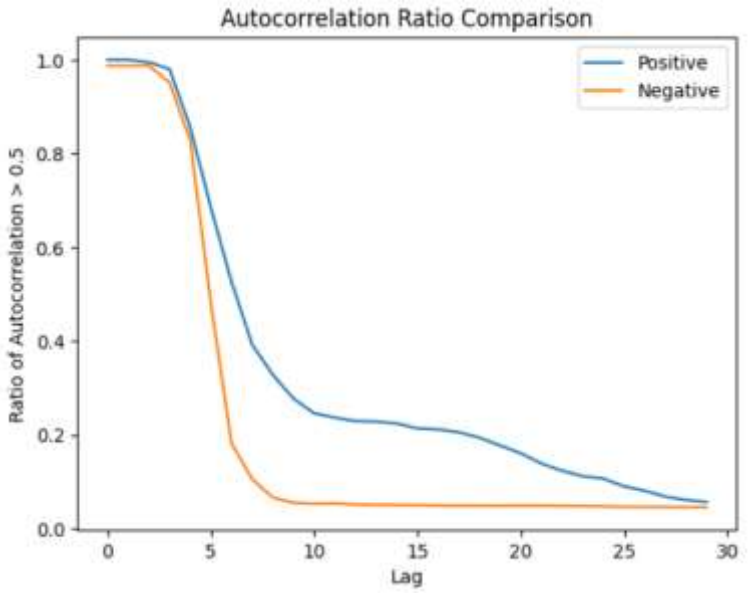
    Normal(전체)

각 데이터의 각 축마다 1~29까지의 lag 값에 대해 autocorrelation을 계산하여, autocorrelation > 0.5인 데이터의 비율을 lineplot으로 시각화

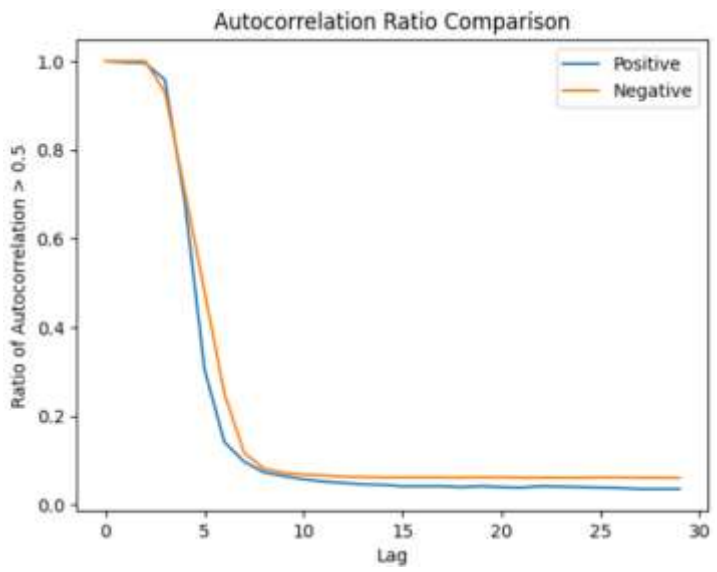
X axis



Y axis



Z axis

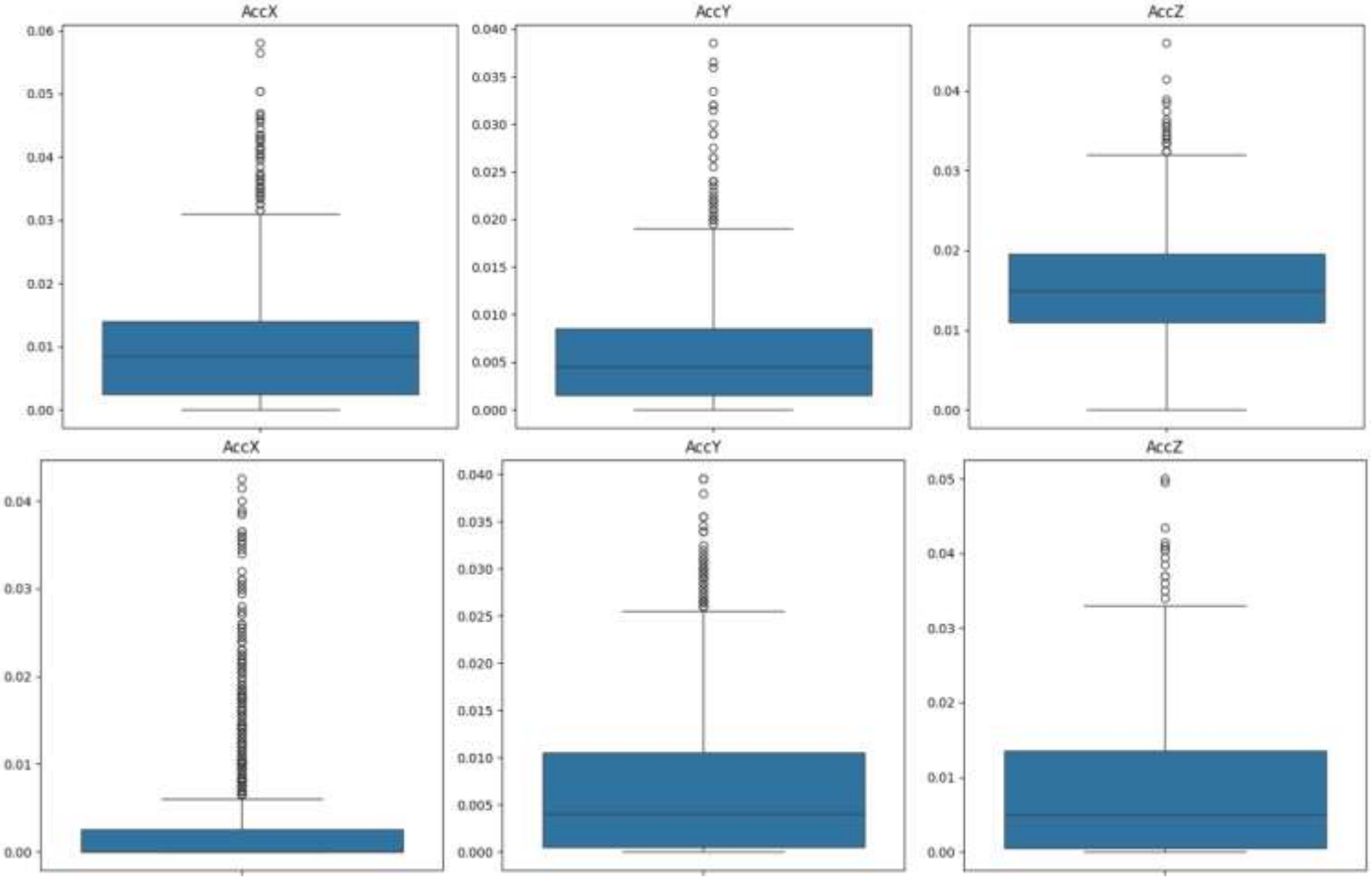


3. 연구 방법

- **EDA(Explorative Data Analysis) – clipped data**

Cerebellar Ataxia vs. Normal

- **2. 분포 및 이상치 확인**



Cerebellar Ataxia(전체):

각 데이터마다 z-score를 계산하여, 각 축마다 z-score < -3 또는 z-score > 3인 시점의 비율을 구한 후, boxplot으로 시각화

Normal(전체):

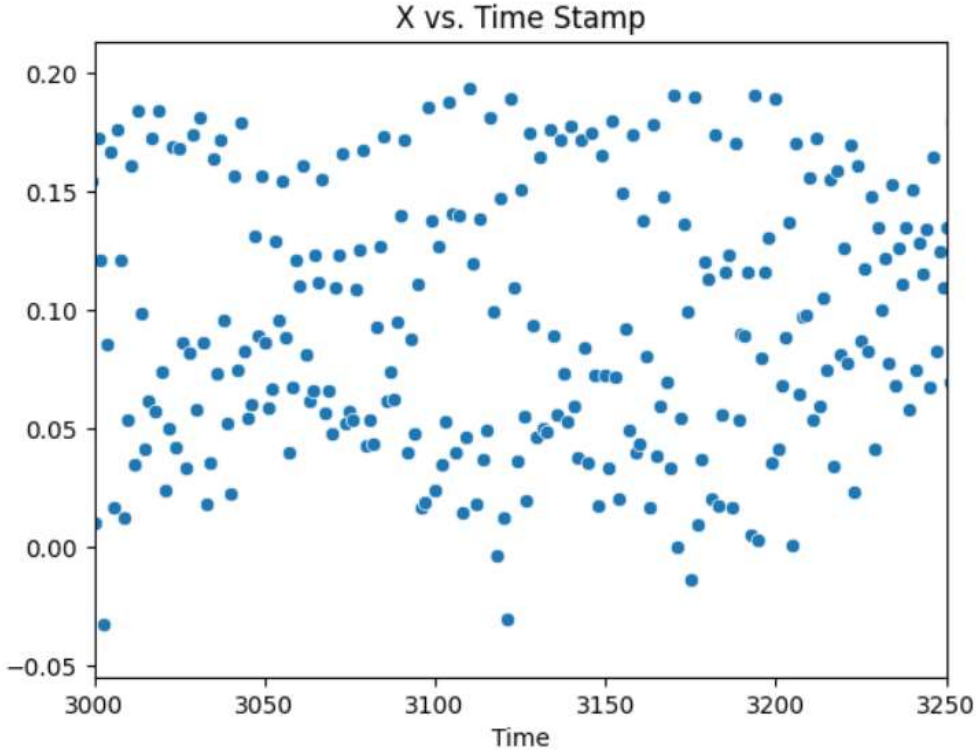
3. 연구 방법

- **EDA(Explorative Data Analysis) – clipped data**

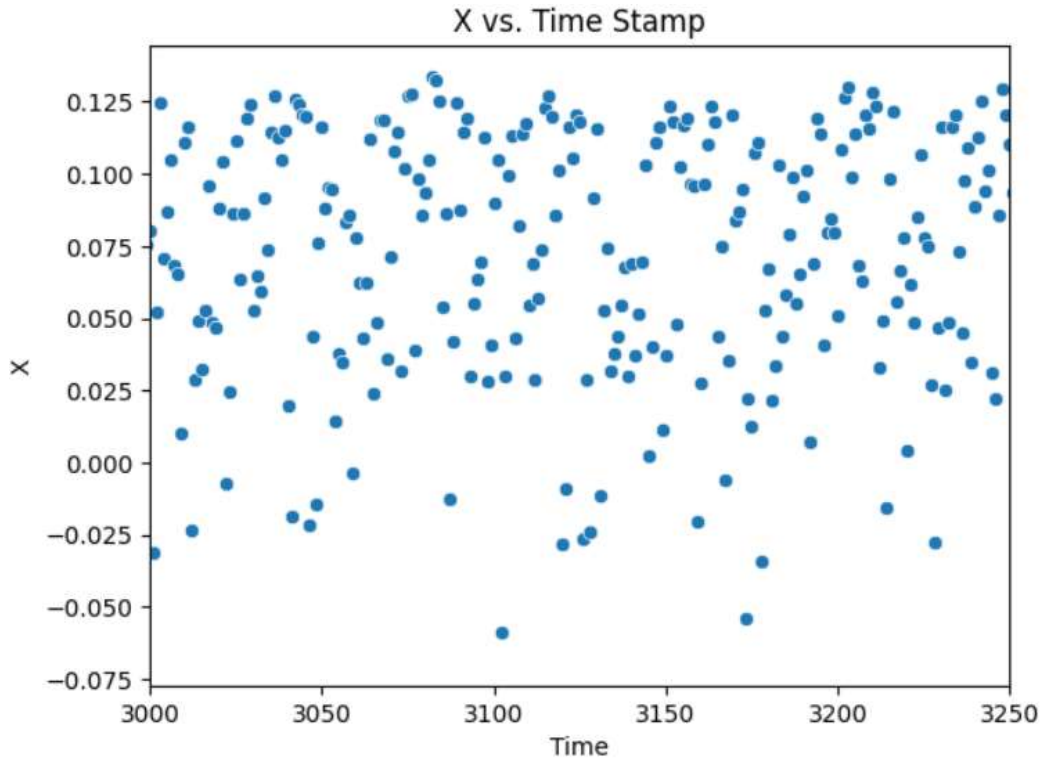
Cerebellar Ataxia vs. Normal

- **3. 시계열 데이터의 시각화**

Cerebellar Ataxia(일부):



Normal(일부):



3. 연구 방법

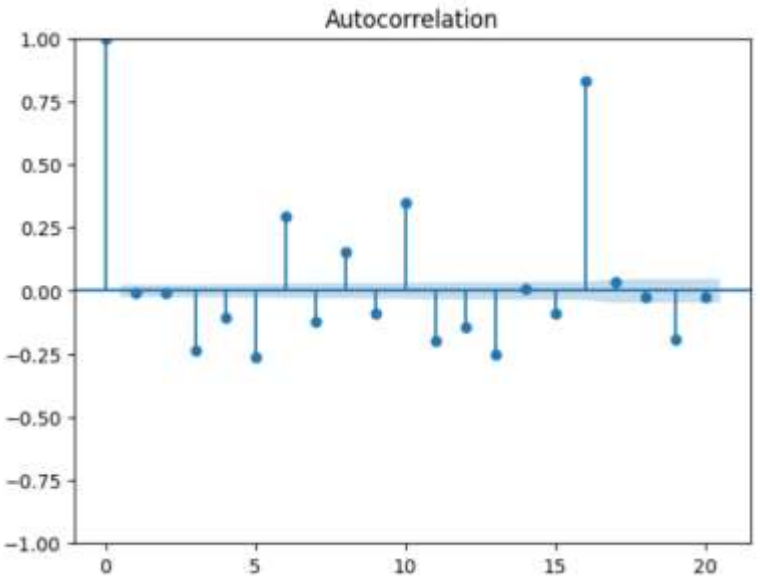
- **EDA(Explorative Data Analysis) – clipped data**

Cerebellar Ataxia vs. Normal

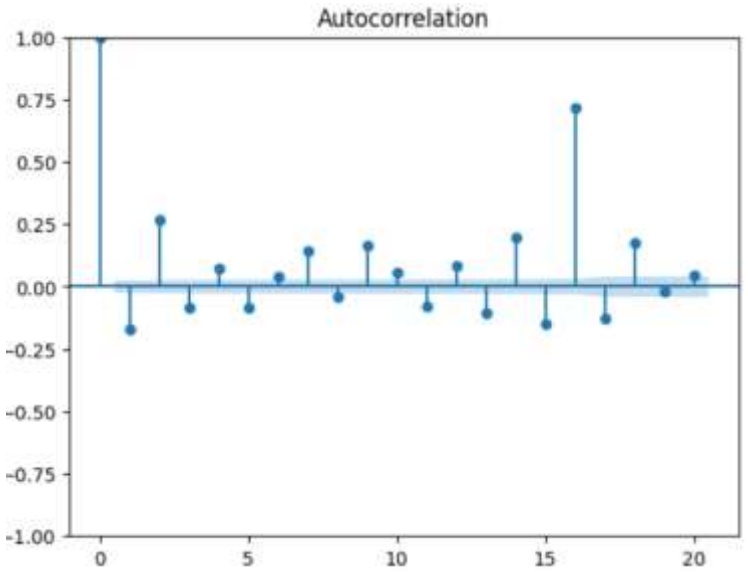
- **4. 시계열 데이터의 분석**
  - **Autocorrelation:** lag

Cerebellar Ataxia(일부):

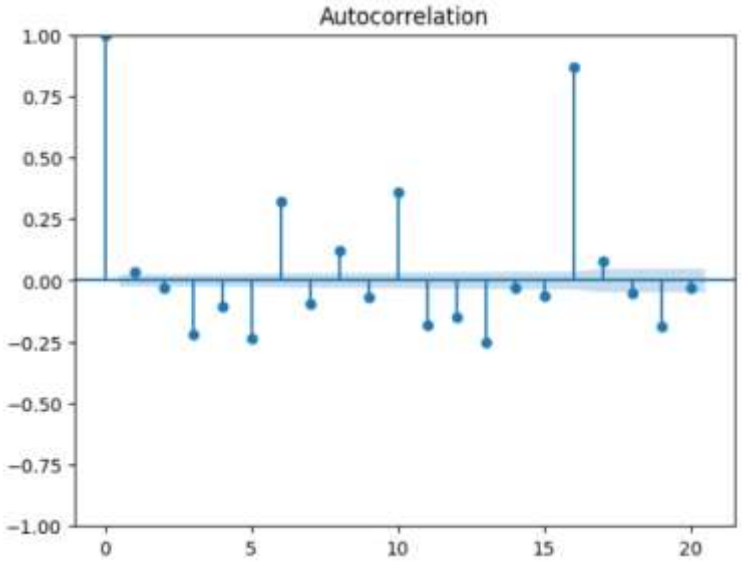
X axis



Y axis



Z axis



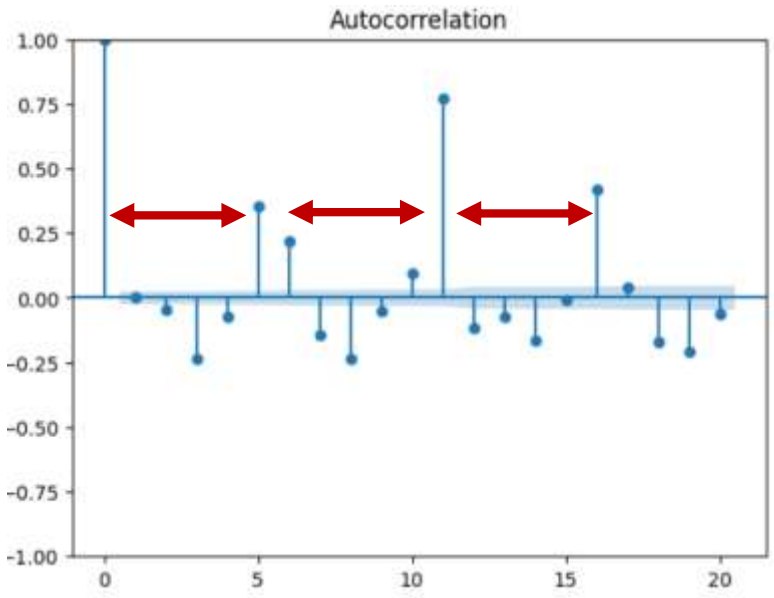
- **EDA(Explorative Data Analysis) – clipped data**

Cerebellar Ataxia vs. Normal

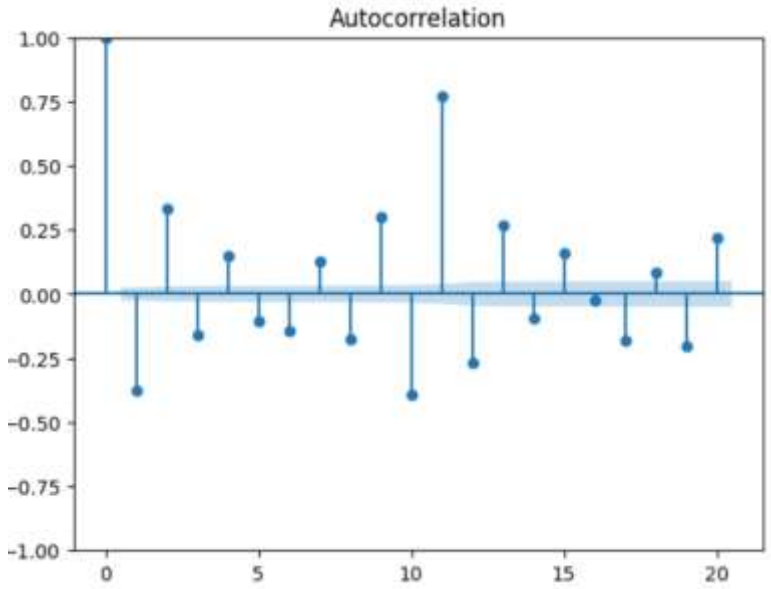
- **4. 시계열 데이터의 분석**
  - **Autocorrelation:** lag

Normal(일부):

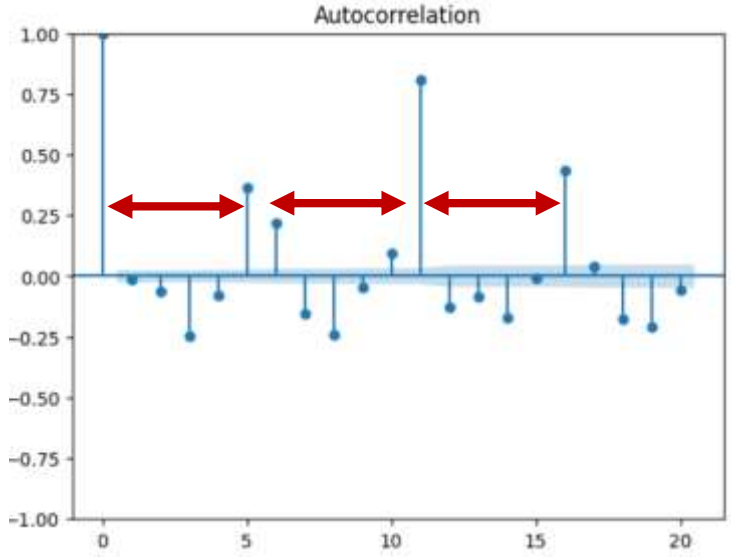
X axis



Y axis



Z axis



- **EDA(Explorative Data Analysis) – clipped data**

Cerebellar Ataxia vs. Normal

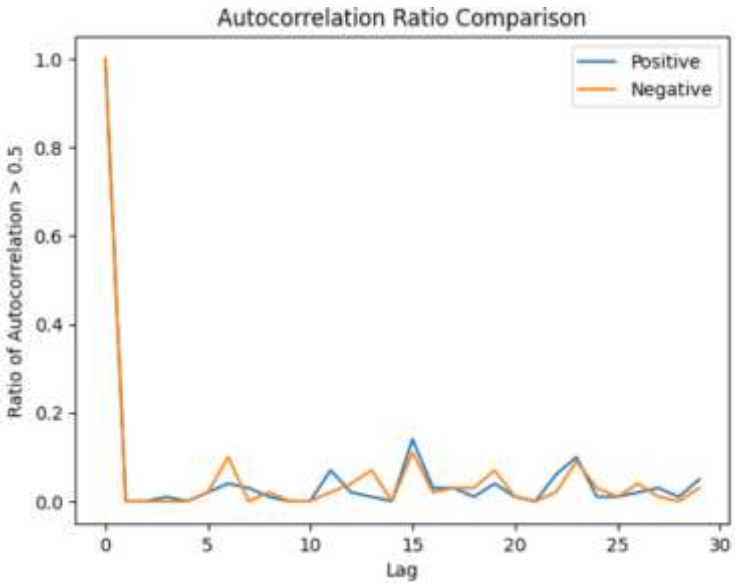
- **4. 시계열 데이터의 분석**
  - **Autocorrelation:** lag

    Cerebellar Ataxia(전체)

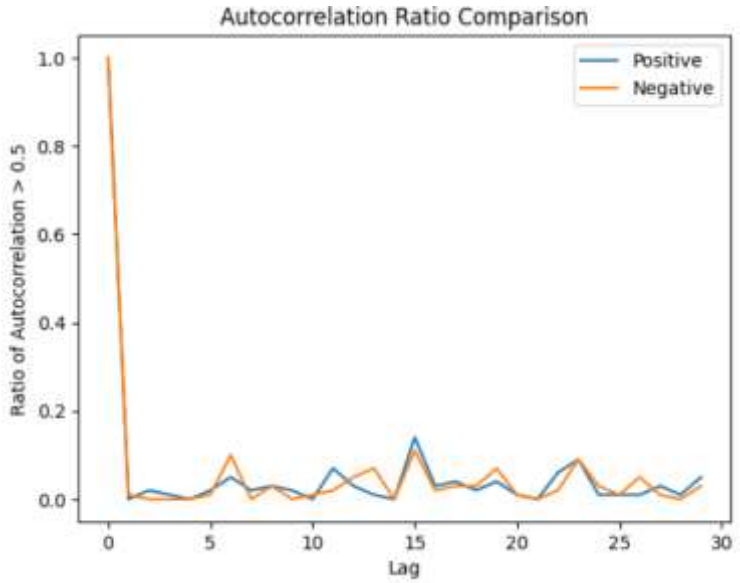
    Normal(전체)

각 데이터의 각 축마다 1~29까지의 lag 값에 대해 autocorrelation을 계산하여, autocorrelation > 0.5인 데이터의 비율을 lineplot으로 시각화

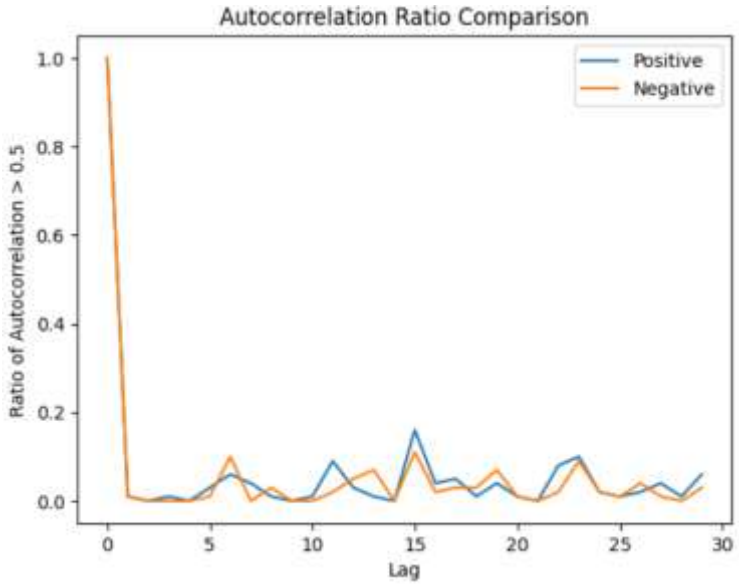
X axis



Y axis



Z axis



3. 연구 방법

- 데이터 전처리 – Fast Fourier Transform(FFT)

- Fourier Transform이란? f(x)를 삼각함수들로 구성된 급수, 더 나아가 적분 식으로 근사하는 변환법

$$f(x) = \int_{-\infty}^{\infty} F(k) e^{2\pi i k x} dk$$
$$F(k) = \int_{-\infty}^{\infty} f(x) e^{-2\pi i k x} dx$$

원리

$$x[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi kn}{N}}$$

Fast Fourier Transform(FFT)

그 계수를 정하기 위해 서로 다른 진동수를 가진 삼각함수의 곱을 정적분하면 값이 0이 된다는 점을 이용하며 위와 같은 식이 나오게 된다. (FFT는 이산적으로 빠르게 Fourier Transform을 수행하는 방식)

3. 연구 방법

- 데이터 전처리 – Fast Fourier Transform(FFT)

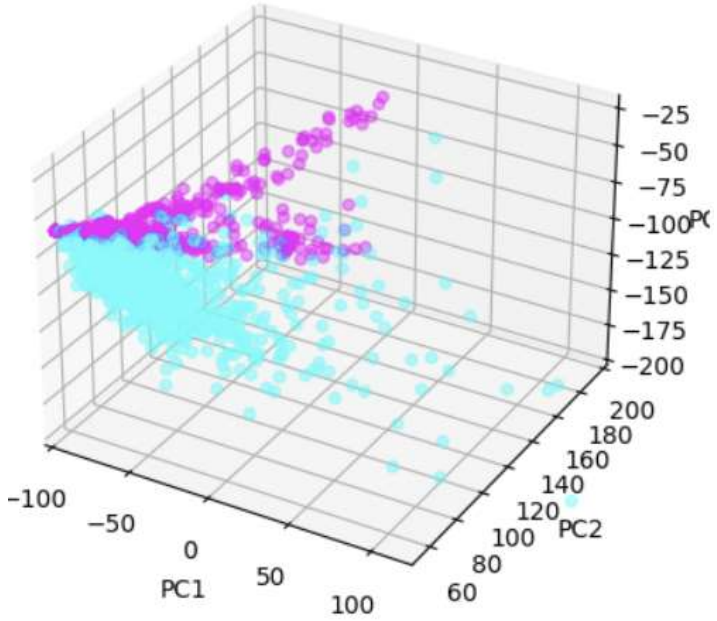
- Gait data는 본질적으로 일정한 주기를 두고 같은 행위가 반복되기에, 이러한 데이터를 처리하기에는 주기함수의 형태로 표현되는 Fourier Transform을 적용하는 것이 유리하겠다 판단함.

- 물론, FFT를 사용함으로써 model 학습 상 불리해질 수 있음을 고려하여 가공되지 않은 data를 사용한 model 역시 만든 뒤 boosting stacking algorithm을 사용하기로 판단함.

3. 연구 방법

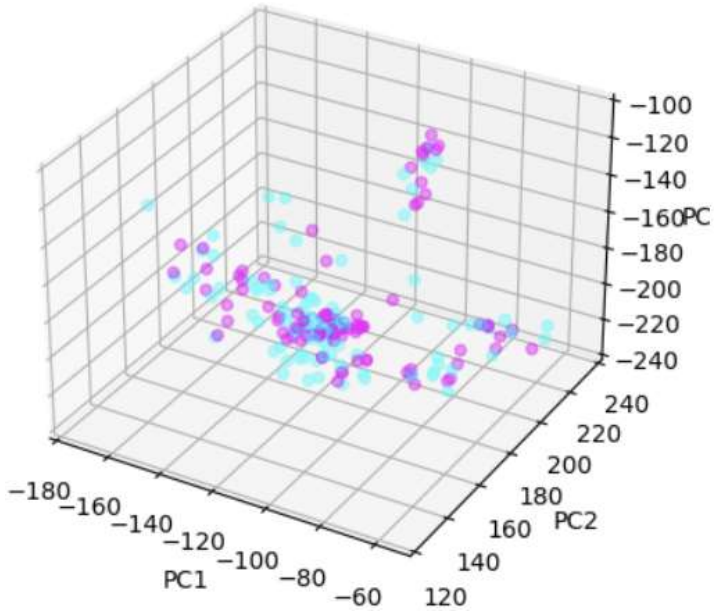
- **EDA(Explorative Data Analysis) – after FFT**

Parkinson's vs. normal



Pink : Parkinson's
Mint : normal

Cerebellar ataxia vs. normal



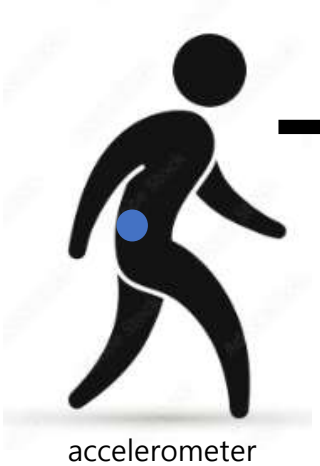
Pink : Cerebellar ataxia
Mint : normal

3. 연구 방법

연세대학교 의과대학
YONSEI UNIVERSITY COLLEGE OF MEDICINE

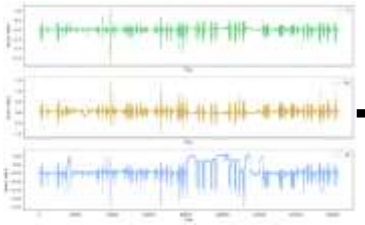
Data Acquisition and Preprocessing

- **Data from "Kaggle" Platform**



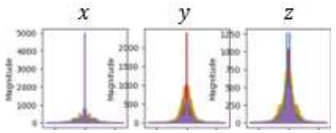
accelerometer

a_x (or x)
a_y (or y)
a_z (or z)



3D dynamics data

FFT



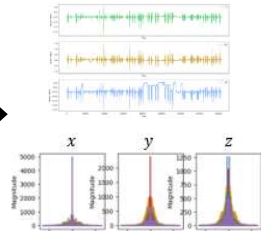
3D dynamics data

Generation of Data Sets

Acceleration (3D)	Distance (3D)
Normal (n=1906)	Normal (n=100)
Parkinson's (n=962)	Cerebellar Ataxia (n=100)
Train/Test Data 80%/20%	Train/Test Data 80%/20%

AI Train & Test

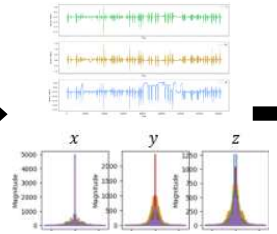
3D Acceleration Model(Normal vs Parkinson's)



LSTM
Nonlinear SVM
Efficient Net

Boosting Stacking Algorithm

3D Distance Model(normal vs cerebellar ataxia)



LSTM
Nonlinear SVM
Efficient Net

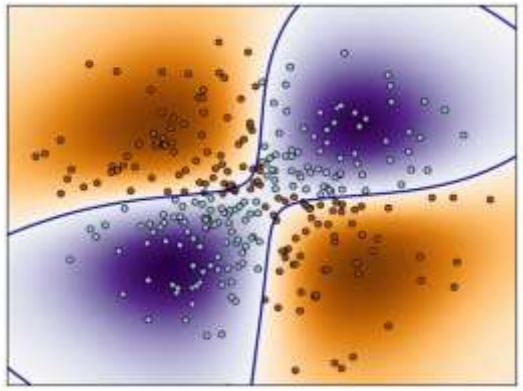
Boosting Stacking Algorithm

3. 연구 방법

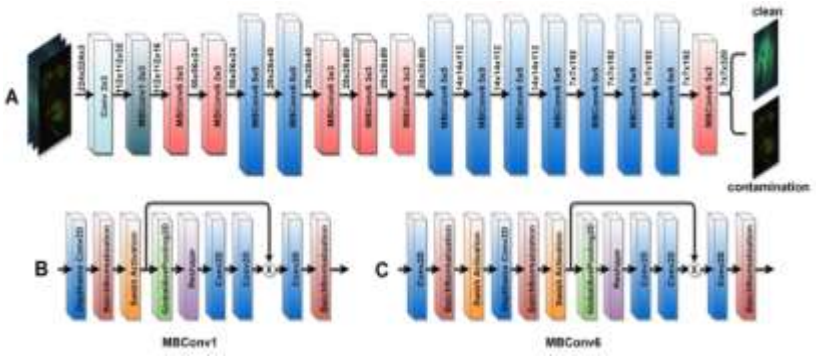
- **학습 Data 및 Model 선택**

기본적으로 model 3개를 사용 후, Stacking Ensemble Algorithm 적용.

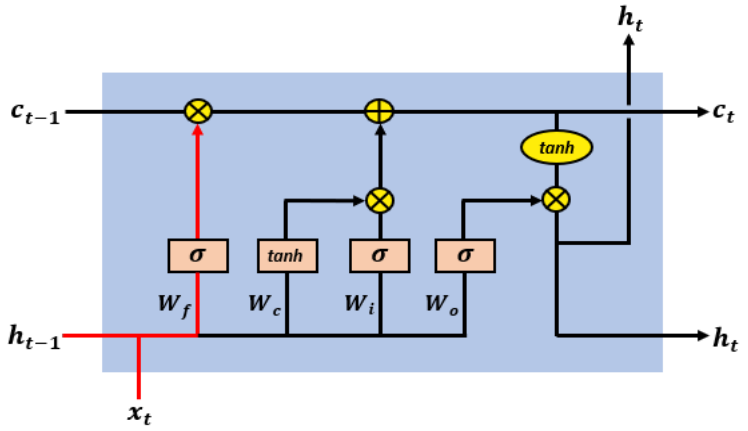
- **Non-linear SVM** : Support Vector Machine의 non-linear form
- **EfficientNet** : 높은 확장성과 효율성을 보이는 고성능 CNN
- **LSTM** : vanishing gradient problem 방지 및 높은 성능의 RNN 모델 (Long Short Term Memory)



nonlinear SVM



EfficientNet



LSTM

3. 연구 방법

- 세 가지 model의 data input은 **다르게** 지정하였습니다.

- **Nonlinear SVM**과 **EfficientNet**은 Fast Fourier Transform를 거친 data를 사용합니다.
- 이는 저희의 modeling pipeline에서 두 model을 주파수 영역의 **Base Wave** 로부터 Disease Pattern을 인식하기 위한 model로 설계하였기 때문입니다.

- LSTM은 기존 시계열 속성을 유지한 전처리된 data를 그대로 사용하였습니다.

3. 연구 방법

EfficientNet과 LSTM의 학습 environment는 다음과 같습니다.

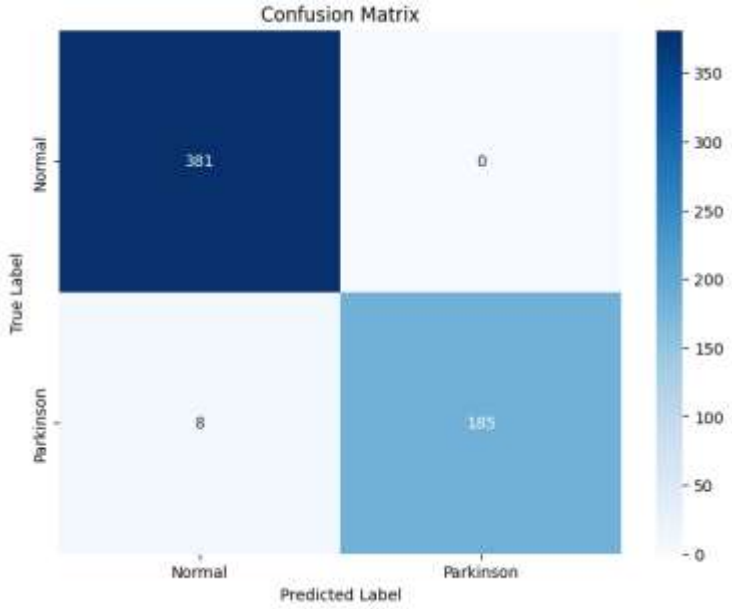
- CPU : 20, Memory : 128 Gi, using GPU
- Base setting: 100 epochs, batch = 4, BCEWithLogitsLoss
- optimization setting : optimizer = Adam, lr = 1e-3 & schedular (step = 10, gamma = 0.5),

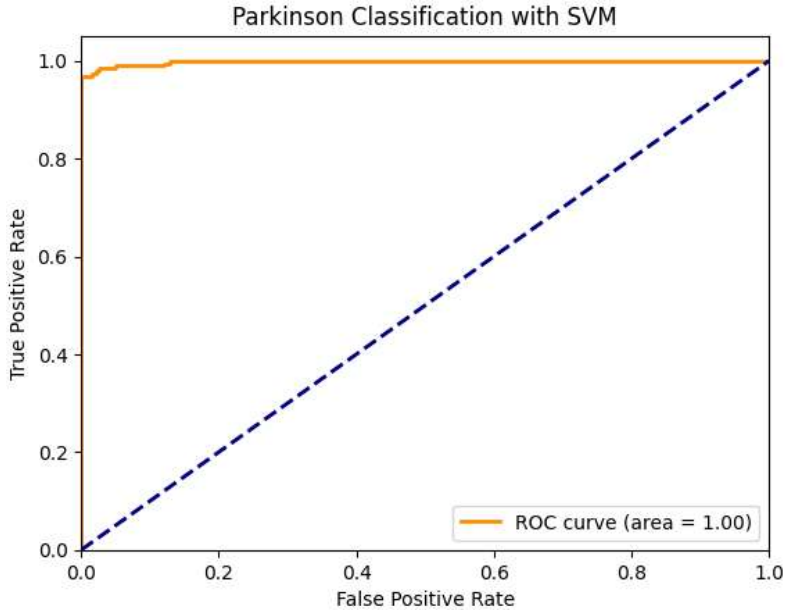
Task 1. 3D 가속도 데이터를 사용한 Parkinson's 환자 vs normal gait 분류

4. 연구 결과

- **Non-linear SVM(with RBF kernel)**

Accuracy: 0.9861



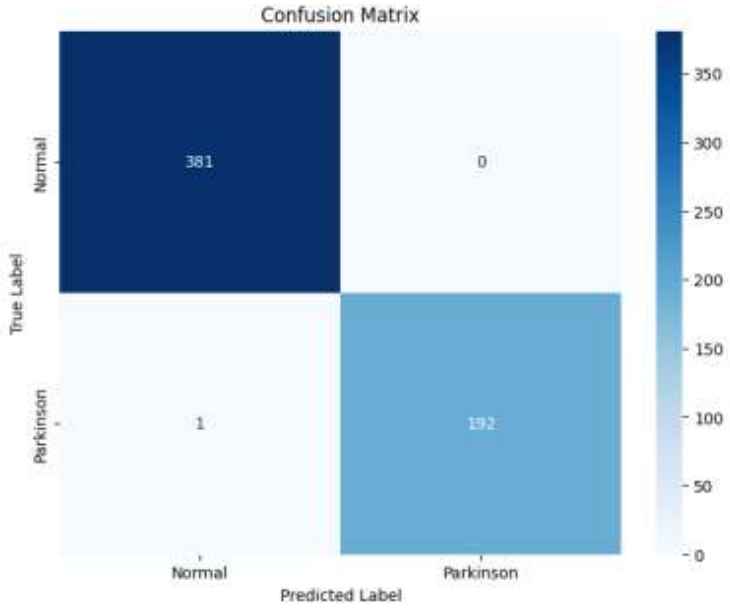


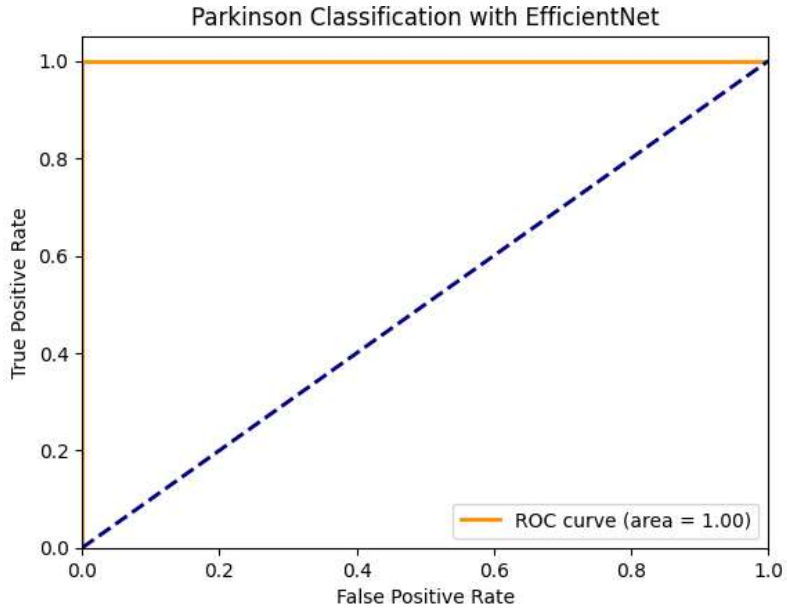
Classification Report:

	precision	recall	f1-score	support
Normal	0.98	1.00	0.99	381
Parkinson	1.00	0.96	0.98	193
accuracy			0.99	574
macro avg	0.99	0.98	0.98	574
weighted avg	0.99	0.99	0.99	574

- **EfficientNet(with pretrained ver. b6)**

Accuracy: 0.9983



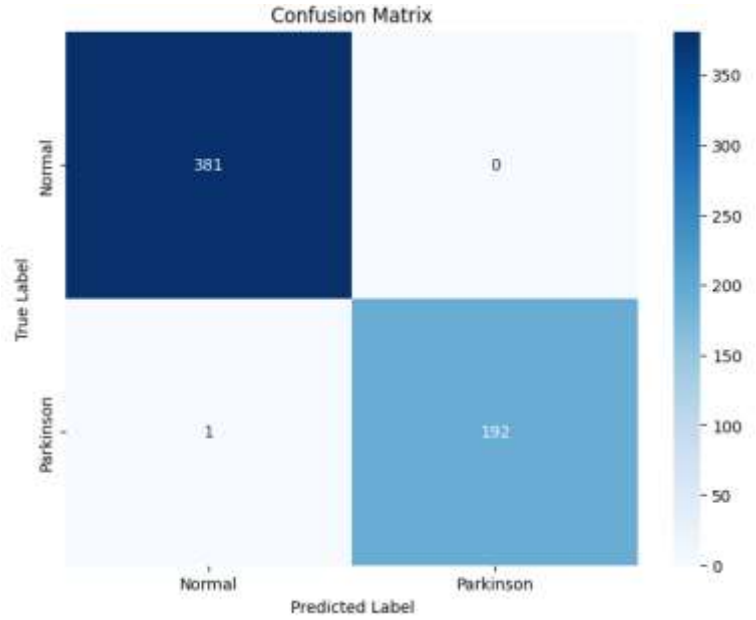


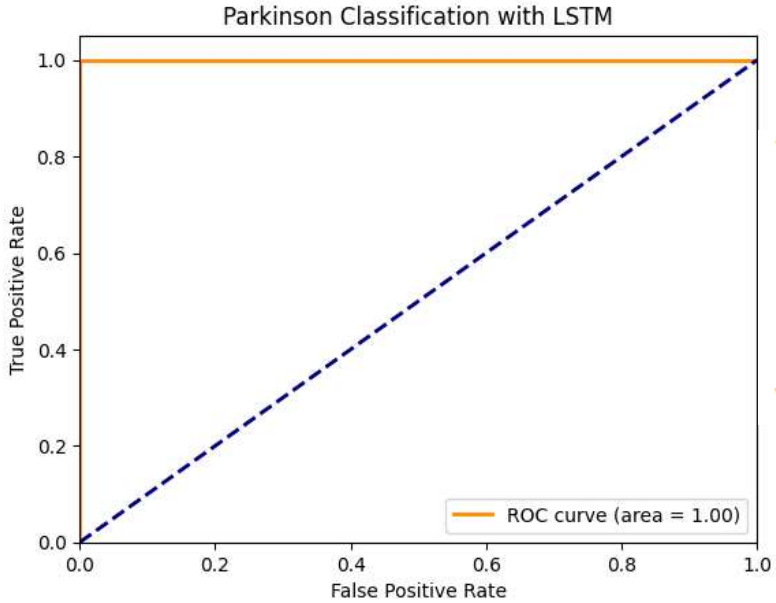
Classification Report:

	precision	recall	f1-score	support
Normal	1.00	1.00	1.00	381
Parkinson	1.00	0.99	1.00	193
accuracy			1.00	574
macro avg	1.00	1.00	1.00	574
weighted avg	1.00	1.00	1.00	574

- **LSTM**

Accuracy: 0.9983





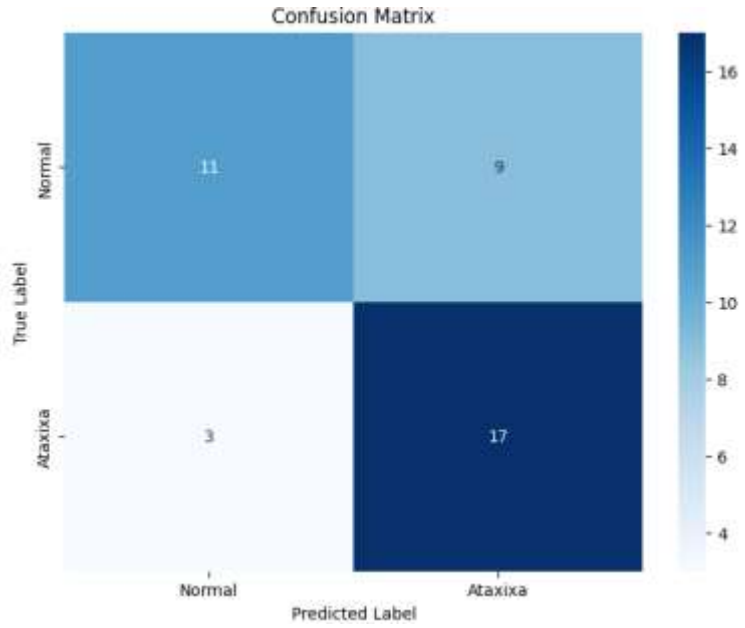
Classification Report:

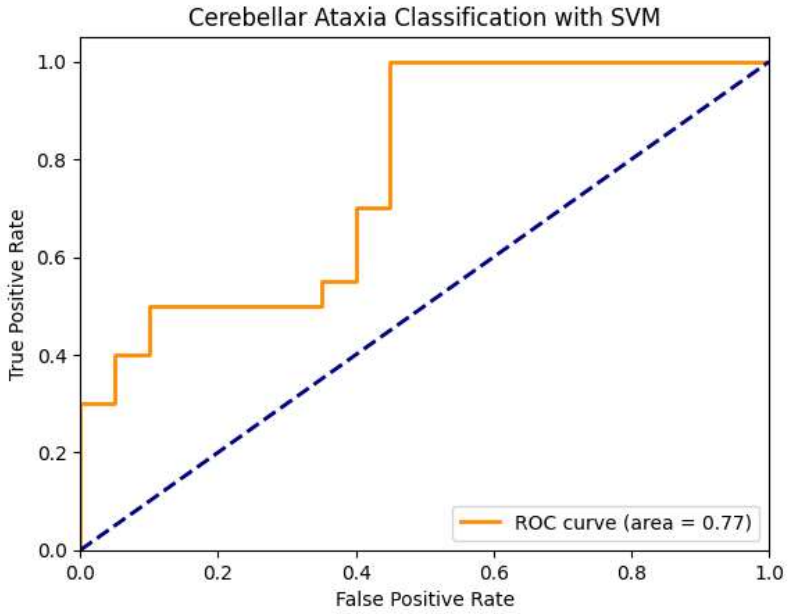
	precision	recall	f1-score	support
Normal	1.00	1.00	1.00	381
Parkinson	1.00	0.99	1.00	193
accuracy			1.00	574
macro avg	1.00	1.00	1.00	574
weighted avg	1.00	1.00	1.00	574

Task 2. 3D 위치 데이터를 사용한 Cerebellar Ataxia vs normal gait 분류

- **Non-linear SVM(with RBF kernel)**

Accuracy: 0.7000





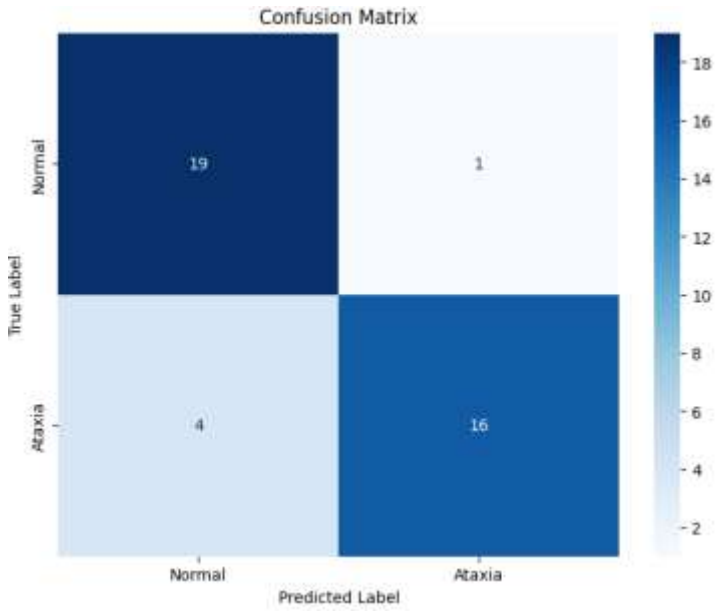
Classification Report:

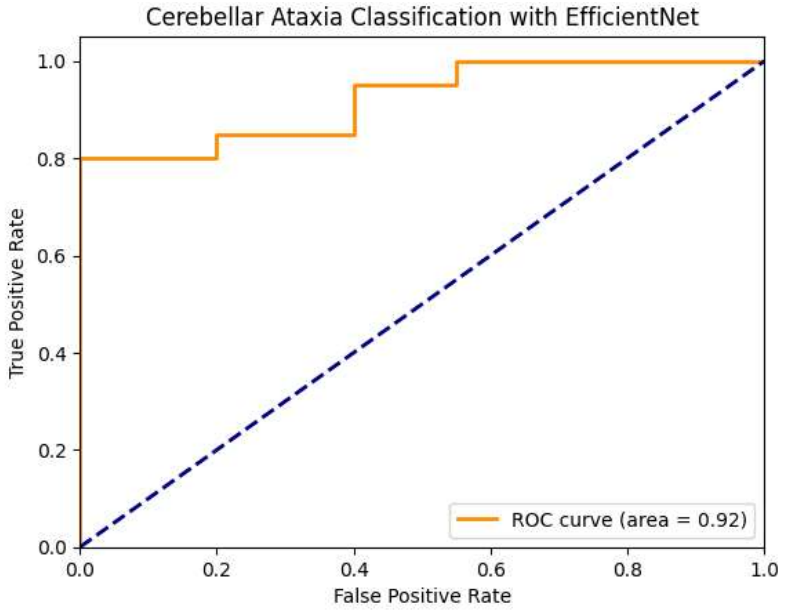
	precision	recall	f1-score	support
Normal	0.79	0.55	0.65	20
Ataxixa	0.65	0.85	0.74	20
accuracy			0.70	40
macro avg	0.72	0.70	0.69	40
weighted avg	0.72	0.70	0.69	40

4. 연구 결과

- **EfficientNet(with pretrained ver. b6)**

Accuracy: 0.8750





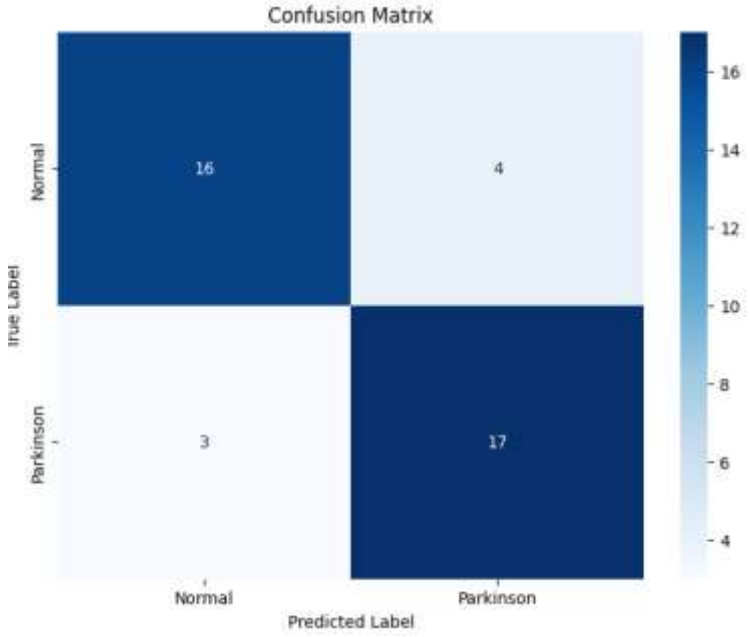
Classification Report:

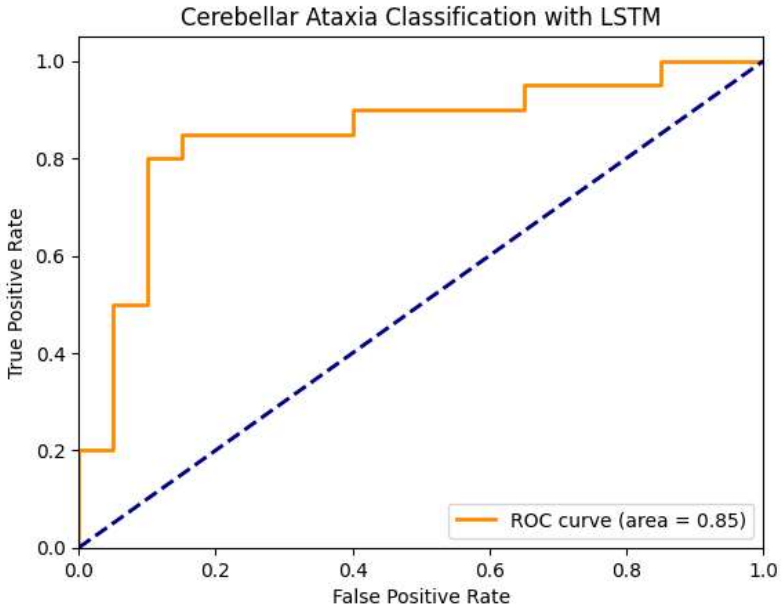
	precision	recall	f1-score	support
Normal	0.83	0.95	0.88	20
Ataxia	0.94	0.80	0.86	20
accuracy			0.88	40
macro avg	0.88	0.88	0.87	40
weighted avg	0.88	0.88	0.87	40

4. 연구 결과

- **LSTM**

Accuracy: 0.8250



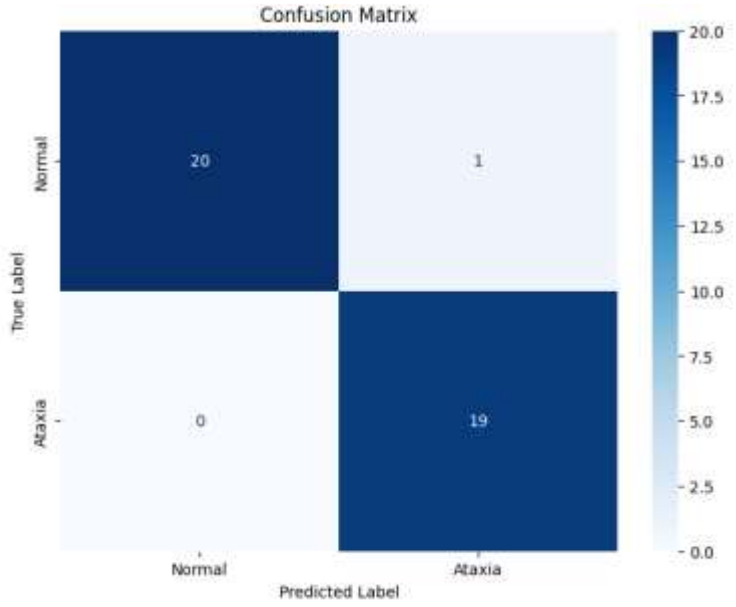


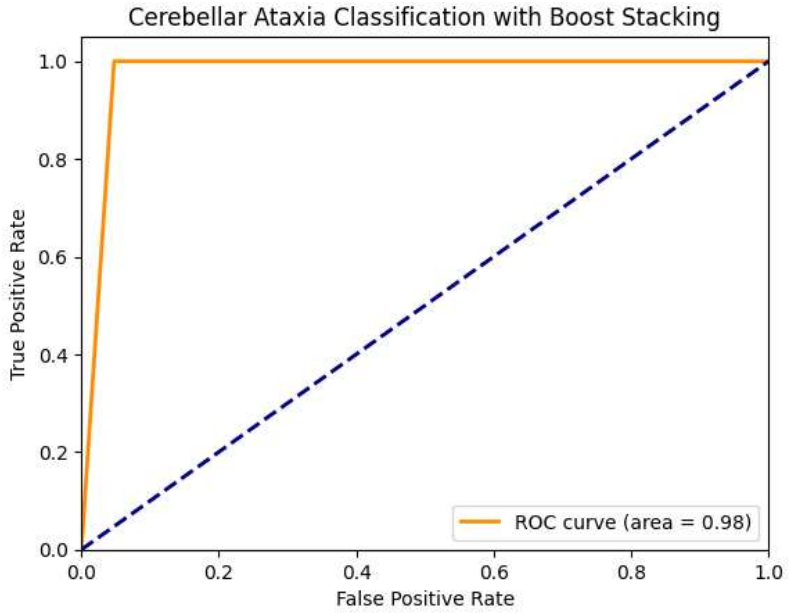
Classification Report:

	precision	recall	f1-score	support
Normal	0.84	0.80	0.82	20
Ataxia	0.81	0.85	0.83	20
accuracy			0.82	40
macro avg	0.83	0.82	0.82	40
weighted avg	0.83	0.82	0.82	40

- **Stacking Ensemble**

Accuracy: 0.9750





Classification Report:

	precision	recall	f1-score	support
Normal	1.00	0.95	0.98	21
Ataxia	0.95	1.00	0.97	19
accuracy			0.97	40
macro avg	0.97	0.98	0.97	40
weighted avg	0.98	0.97	0.98	40

5. 논의 사항

- Task 1에서는, 3가지 model에서 모두 충분히 높은 성능을 보임.
  - ➢ 이는 정상인과 Parkinson's 환자의 gait가 확연히 다른 행태를 보임을 시사함.

- 주목할 만한 점은, Parkinson's 환자를 정상인으로 분류한 case가 1건도 없다는 것임.

- Task 2에서는, Parkinson's 환자에 비해 cerebellar atrophy 환자는 정상과 비교하여 덜 확연한 차이를 보였으며 세 모델 모두 Task 1에 비해 낮은 성능을 보임.
- Stacking Ensemble algorithm을 통해 세 모델에 비해 확연히 높은 성능을 얻어냄. (with. logistic regression)

6. 향후 계획

- 더 체계적이고 일관된 data 취득을 통해 모델의 정확도와 신뢰도 향상
  - 다른 방식으로 취합된 data들을 합치는 과정에서 group간 바람직하지 않은 차이가 발생
  - 보행 이상의 원인이 될 수 있는 다양한 질환 군에 대한 데이터 취득, 이후 학습 진행
  - **비중증 or 호전(+/- 치료)된 Parkinson's disease data를 추가하여 더 자세한 분류**를 지원

- FFT 대신 STFT(Short Time Fourier Transformation)과 같이 더 발전된 Fourier Transform 기법 사용
- PCA 상 splitting을 보이는 subgroup에 대한 subgroup analysis
  - Parkinson's disease EDA 상에서 splitting 관찰됨

- 워치, 링, 폰 등에 기능을 도입하기 위해 **다양한 부위의 data에 적용할 수 있는 AI모델** 개발

- 날씨, 식사 전후 등 **오차 요인에 대응하기 위한 방안 마련**

7. 주요 코드

Non-linear SVM(RBF kernel)

```python
class Gait_Dataset(Dataset):
    def __init__(self, root_dir, transform=None):
        self.root_dir = root_dir
        self.transform = transform

        self.data = []
        self.labels = []

        normal_dir = os.path.join(root_dir, 'normal')
        parkinson_dir = os.path.join(root_dir, 'parkinson')

        # Load normal data
        for file in os.listdir(normal_dir):
            if file.endswith('.csv'):
                file_path = os.path.join(normal_dir, file)
                data = np.loadtxt(file_path, delimiter=',')
                self.data.append(data)
                self.labels.append(0)  # Label for normal

        # Load parkinson data
        for file in os.listdir(parkinson_dir):
            if file.endswith('.csv'):
                file_path = os.path.join(parkinson_dir, file)
                data = np.loadtxt(file_path, delimiter=',')
                self.data.append(data)
                self.labels.append(1)  # Label for parkinson

        # Convert data and labels to numpy arrays
        self.data = np.array(self.data)
        self.labels = np.array(self.labels)

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        sample = self.data[idx]
        label = self.labels[idx]

        if self.transform:
            sample = self.transform(sample)

        return sample, label
```

```python
root_dir = 'Task1/ffT'
dataset = Gait_Dataset(root_dir=root_dir)
data = dataset.data
labels = dataset.labels
data = np.array([sample.flatten() for sample in dataset.data])
labels = np.array(dataset.labels)

X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, stratify=labels, random_state=42)
```

```python
svm_model = SVC(kernel='rbf', probability=True)
svm_model.fit(X_train, y_train)
joblib.dump(svm_model, 'svm_model.joblib')
print('Model saved to svm_model.joblib')

y_pred = svm_model.predict(X_test)
y_prob = svm_model.predict_proba(X_test)[:, 1]
```

```
Model saved to svm_model.joblib
```

7. 주요 코드

EfficientNet(with pretrained ver. b6)

```python
class CustomEfficientNet(nn.Module):
    def __init__(self, model_name='efficientnet_b6'):
        super(CustomEfficientNet, self).__init__()
        if model_name == 'efficientnet_b0':
            self.base_model = efficientnet_b0(weights=EfficientNet_B0_Weights.IMAGENET1K_V1)
        elif model_name == 'efficientnet_b1':
            self.base_model = efficientnet_b1(weights=EfficientNet_B1_Weights.IMAGENET1K_V1)
        elif model_name == 'efficientnet_b2':
            self.base_model = efficientnet_b2(weights=EfficientNet_B2_Weights.IMAGENET1K_V1)
        elif model_name == 'efficientnet_b3':
            self.base_model = efficientnet_b3(weights=EfficientNet_B3_Weights.IMAGENET1K_V1)
        elif model_name == 'efficientnet_b4':
            self.base_model = efficientnet_b4(weights=EfficientNet_B4_Weights.IMAGENET1K_V1)
        elif model_name == 'efficientnet_b5':
            self.base_model = efficientnet_b5(weights=EfficientNet_B5_Weights.IMAGENET1K_V1)
        elif model_name == 'efficientnet_b6':
            self.base_model = efficientnet_b6(weights=EfficientNet_B6_Weights.IMAGENET1K_V1)
        elif model_name == 'efficientnet_b7':
            self.base_model = efficientnet_b7(weights=EfficientNet_B7_Weights.IMAGENET1K_V1)
        else:
            raise ValueError("Invalid model name. Choose from 'efficientnet_b0' to 'efficientnet_b7'.")

        self.base_model.features[0][0] = nn.Conv2d(1, self.base_model.features[0][0].out_channels,
                                                   kernel_size=self.base_model.features[0][0].kernel_size,
                                                   stride=self.base_model.features[0][0].stride,
                                                   padding=self.base_model.features[0][0].padding,
                                                   bias=False)

        self.fc = nn.Linear(self.base_model.classifier[1].in_features, 1)
        self.base_model.classifier = nn.Identity()

    def forward(self, x):
        x = self.base_model(x)
        x = self.fc(x)
        return x

    def predict(self, x):
        logits = self.forward(x)
        probs = torch.sigmoid(logits)
        preds = (probs > 0.5).float()  # Threshold at 0.5 for binary classification
        return preds, probs
```

```python
def train_with_eval(model, criterion, optimizer, scheduler, train_loader, val_loader, num_epochs=10):
    best_loss = float('inf')
    for epoch in range(num_epochs):
        model.train()
        running_loss = 0.0
        for inputs, targets in train_loader:
            inputs = inputs.to(device)
            targets = targets.to(device)
            optimizer.zero_grad()
            outputs = model(inputs)
            targets = targets.unsqueeze(1)
            loss = criterion(outputs, targets)
            loss.backward()
            optimizer.step()
            running_loss += loss.item() * inputs.size(0)
        epoch_loss = running_loss / len(train_loader.dataset)
        print(f"Epoch {epoch+1}/{num_epochs}, Loss: {epoch_loss:.4f}")

        scheduler.step()
        model.eval()
        running_loss = 0.0
        with torch.no_grad():
            for inputs, targets in val_loader:
                inputs = inputs.to(device)
                targets = targets.to(device)
                outputs = model(inputs)
                targets = targets.unsqueeze(1)
                loss = criterion(outputs, targets)
                running_loss += loss.item() * inputs.size(0)
        val_loss = running_loss / len(val_loader.dataset)
        print(f"Validation Loss: {val_loss:.4f}")

        if val_loss < best_loss:
            best_loss = val_loss
            torch.save(model.state_dict(), 'lstm_model.pth')
            print(f"Model saved with loss: {best_loss:.4f}")
```

```python
model = CustomEfficientNet('efficientnet_b6').to(device)
criterion = nn.BCEWithLogitsLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
scheduler = optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.5)
```

7. 주요 코드

LSTM

```python
class Gait_Dataset(Dataset):
    def __init__(self, root_dir, transform=None):
        self.root_dir = root_dir
        self.transform = transform

        self.data = []
        self.labels = []

        normal_dir = os.path.join(root_dir, 'normal')
        parkinson_dir = os.path.join(root_dir, 'parkinson')

        # Load normal data
        for file in os.listdir(normal_dir):
            if file.endswith('.csv'):
                file_path = os.path.join(normal_dir, file)
                data = np.loadtxt(file_path, delimiter=',')
                self.data.append(data)
                self.labels.append(0)  # Label for normal

        # Load parkinson data
        for file in os.listdir(parkinson_dir):
            if file.endswith('.csv'):
                file_path = os.path.join(parkinson_dir, file)
                data = np.loadtxt(file_path, delimiter=',')
                self.data.append(data)
                self.labels.append(1)  # Label for parkinson

        # Convert data and labels to tensors
        self.data = [torch.tensor(d, dtype=torch.float32) for d in self.data]
        self.labels = torch.tensor(self.labels, dtype=torch.float32)

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        sample = self.data[idx]
        label = self.labels[idx]

        if self.transform:
            sample = self.transform(sample)

        return sample, label
```

```python
# Dataset usage
root_dir = 'Task1/ffT'
dataset = Gait_Dataset(root_dir=root_dir)

# Stratified split of dataset into training and test sets
labels = np.array(dataset.labels)
stratified_split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
train_idx, test_idx = next(stratified_split.split(np.zeros(len(labels)), labels))

train_dataset = Subset(dataset, train_idx)
test_dataset = Subset(dataset, test_idx)

train_loader = DataLoader(train_dataset, batch_size=4, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=4, shuffle=False)
```

```python
class LSTMClassifier(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, output_size, dropout=0.3):
        super(LSTMClassifier, self).__init__()
        self.hidden_size = hidden_size
        self.num_layers = num_layers

        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True, dropout=dropout)
        self.dropout = nn.Dropout(dropout)
        self.fc = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).to(device)
        c0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).to(device)

        out, _ = self.lstm(x, (h0, c0))
        out = out[:, -1, :]
        out = self.dropout(out)
        out = self.fc(out)
        return out
```

7. 주요 코드

시각화

```
test_loader = DataLoader(test_dataset, batch_size=4, shuffle=False)

all_labels = []
all_preds = []
all_probs = []

with torch.no_grad():
    for inputs, labels in test_loader:
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = model(inputs)
        probs = torch.sigmoid(outputs)
        preds = (probs > 0.5).float()

        all_labels.extend(labels.cpu().numpy())
        all_preds.extend(preds.cpu().numpy())
        all_probs.extend(probs.cpu().numpy())

accuracy = accuracy_score(all_labels, all_preds)
print(f'Accuracy: {accuracy:.4f}')
```

```
classification_rep = classification_report(all_labels, all_preds, target_names=['Normal', 'Parkinson'])
print(f'Classification Report:\n{classification_rep}')
```

```
conf_matrix = confusion_matrix(all_labels, all_preds)
print(f'Confusion Matrix:\n{conf_matrix}')

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Normal', 'Parkinson'], yticklabels=['Normal', 'Parkinson'])
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
```

```
fpr, tpr, _ = roc_curve(all_labels, all_probs)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Parkinson Classification with EfficientNet')
plt.legend(loc="lower right")
plt.show()
```

참고문헌

1. Center AM. Abnormalities of Gait and Mobility [cited 2024 0725]
2. Medicine S. Gait Abnormalities: Standford Medicine; [cited 2024 0725].
3. Appukutty Manickam MDG. Gait Assessment in General Practice. Australian Journal of General Practice. 2021;50(11):801-6.
4. Joshi S, Khanal S, Bista R. Spatiotemporal Gait Analysis for Cardiovascular Disease. 2022 14th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)2022. p. 181-6.
5. Mc Ardle R MR, Wilson J, Galna B, Thomas AJ, Rochester L. What Can Quantitative Gait Analysis Tell Us about Dementia and Its Subtypes? A Structured Review. J Alzheimers Dis. 2017;60(4):1295-312.
6. Bahureksa L, Najafi B, Saleh A, Sabbagh M, Coon D, Mohler MJ, et al. The Impact of Mild Cognitive Impairment on Gait and Balance: A Systematic Review and Meta-Analysis of Studies Using Instrumented Assessment. Gerontology. 2017;63(1):67-83.
7. Kikkert L. Gait characteristics as indicators of cognitive impairment in geriatric patients. Groningen: Rijksuniversiteit te Groningen; 2018.
8. Montero-Odasso M, Verghese J, Beauchet O, Hausdorff JM. Gait and cognition: a complementary approach to understanding brain function and the risk of falling. J Am Geriatr Soc. 2012;60(11):2127-36.
9. Cronin P, Collins LM, Sullivan AM. Impacts of gait freeze on quality of life in Parkinson's disease, from the perspectives of patients and their carers. Ir J Med Sci. 2024.
10. Walton CC, Shine JM, Hall JM, O'Callaghan C, Mowszowski L, Gilat M, et al. The major impact of freezing of gait on quality of life in Parkinson's disease. J Neurol. 2015;262(1):108-15.
11. Price R, Choy NL. Investigating the Relationship of the Functional Gait Assessment to Spatiotemporal Parameters of Gait and Quality of Life in Individuals With Stroke. J Geriatr Phys Ther. 2019;42(4):256-64.

참고문헌

12. Chitale V, Playne D, Liang H-N, Baghaei N. Virtual Reality Data for Predicting Mental Health Conditions. 2022 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)2022. p. 6-8.

13. Chen H, Zhang, Y., Kalra, M. K., Lin, F., Chen, Y., Liao, P., ... & Wang, G. Low-Dose CT with a Residual Encoder-Decoder Convolutional Neural Network. IEEE transactions on medical imaging. 2017;36(12):2524-35.

14. Van Criekinge T, Saeys W, Truijen S, Vereeck L, Sloot LH, Hallemans A. A full-body motion capture gait dataset of 138 able-bodied adults across the life span and 50 stroke survivors. Sci Data. 2023;10(1):852.

15. Noella RN, Gupta, D., & Priyadarshini, J. Diagnosis of Parkinson's disease using Gait Dynamics and Images. Procedia Computer Science. 2019;165:428-34.