Pusan National University
Computer Science and Engineering
Technical Report 2023-10

mmwave 센서를 사용한

위험 행동 및 일상 행동 인식 장치

강수빈

박준형

이영인

지도교수 백윤주

목 차

1. 서론

1.1. 연구 배경

고령자들에게 낙상은 흔하다. 가정에 거주하는 고령자 중 1/3은 적어도 1년에 한 번 낙상을 경험하고 요양원에 거주하는 사람들의 절반이 낙상을 경험한다. 미국의 경우 낙상은 대표적인 사고사의 원인이며 65세 이상 고령자의 사망원인 7번째에 해당하기도 한다.

낙상 후 바로 일어나지 못하거나 도움을 부르지 못한다면 낙상은 더 많은 문제를 초래할 수 있다. 이런 문제를 해결하기 위한 CCTV설치는 사생활을 침범할 수 있으므로 mmwave 센서를 통해 낙상을 감지하고 알림이 가는 서비스를 제공하여 사생활을 침범하지 않으면서 고령자들에게 도움이 될 수 있게 주제를 선정하였다.

1.2. 기존 문제점

본 주제를 수행하기에 앞서 mmwave 센서에 대한 사전조사를 통해 다음과 같은 문제를 확인할 수 있었다. mmwave 센서가 정적인 동작의 경우 움직임을 잘 감지하지 못할 수 있다. 정적인 동작뿐만 아니라 격한 동작의 경우에도 사용자의 움직임에 대한 인식이 잘 이루어지지 않을 수 있는 제약사항을 확인하였다. 이에 대한 해결법으로 적절한 거리와 화면 각도를 사용하여 데이터를 수집하면 문제없이 동작들이 잘 인식될 것으로 예측된다. 따라서 데이터를 수집하기에 가장 적절한 거리와 화면 각도를 찾는 것이 필요하다.

또한 데이터 수집에 부족한 부분은 오픈 데이터를 사용하려고 하였으나 이 프로젝트에 사용할 만한 적절한 데이터를 찾기가 쉽지 않았다. 현재 프로젝트에 사용할 만한 적절한 오픈 데이터에 대한 조사가 더욱 필요하다고 생각했고, 동시에 데이터 수집 부분에서는 부족한 부분이 없도록 시간을 투자하여 직접 필요한 만큼 많은 데이터를 수집해야 한다는 결론을 내렸다.

1.3. 연구 목표

노인분들의 안전을 보장하기 위해 mmwave 센서를 이용한 낙상과 같은 위험 행동을 인지하고 일상 행동과 구분하여 위험 행동이 발생했을 때 알림을 울리는 시스템을 개발하는 것이 목표이다. mmwave 기술은 고주파 전자기파를 이용하여 거리, 속도 및 위치 등을 정밀하게 측정할 수 있는 기술로, 거리에 따른 물체의 이동을 감지하는 데에 효과적이다. 이를 이용하여 낙상이 감지되면 즉시 경고가 울리고 상태를 모니터링할 수 있다.

이 시스템은 노인분들의 안전을 증진하고, 긴급 상황에 대한 신속한 대응을 가능하게 함으로써 생명을 구할 수 있다. 또한, 이 시스템은 가족이나 의료 진과 같은 관리자에게 실시간으로 데이터를 전송하여 적시에 적절한 조치를 취할 수 있도록 도와줄 수 있다. 노인분들의 건강과 안전을 중요하게 생각하는 사회적 책임을 충실히 수행하기 위해 이 시스템을 개발하고자 한다.

2. 연구 배경

2.1. 요구조건 및 제약사항

2.1.1. 요구조건

- mmwave 센서를 사용해 구분하고자 하는 행동들에 대한 데이터 수집이 요구된다.
- 수집된 데이터를 사용하여 일상 행동 및 위험 행동을 구분하는 모델 학습을 진행한다. 이후 이렇게 학습된 모델을 사용하여 실시간으로 측정된 사용자의 행동 데이터를 위험 행동 또는 일상 행동으로 분류할 수 있어야 한다.
- 위험 행동이 인지되었을 때 위험 상황임을 알리는 실시간 모니터링 앱이 필요하다.

2.1.2. 제약사항

- 프로젝트의 제목에서처럼 많은 행동(일상 행동 및 위험 행동)을 추론해서 결과를 도출해내기에는 어렵다.

  ⇒ 위험 행동 2가지, 일상 행동 6가지로 선정하여 프로젝트를 진행하였다.
- 학습모델이 디바이스에 탑재 가능한 크기 및 성능이어야 한다.

  ⇒ 디바이스에 탑재하지 않고 서버를 통해 데이터를 전송하고 휴대전화 앱으로 전송 받는 방식을 사용한다.

2.2. 데이터 수집

2.2.1. 데이터 수집 장치

mmwave 센서 보드 중 IWR6843ISK + DCA1000EVM 보드를 사용하여 사용자의 움직임에 대한 데이터를 수집한다. 또한, 모델 학습에는 사용하지 않지만, 행동 구분을 위한 skeleton 데이터를 위해 Kinect 카메라로 동시에 데이터를 수집한다.

2.2.2. 데이터 수집

단순히 위험 행동(낙상)과 일상 행동으로 구분하여 데이터를 수집하는 것이 아닌 위험 행동과 일상 행동을 아래 이미지와 같이 세분화하여 총 8가지의 행동 데이터를 수집한다.

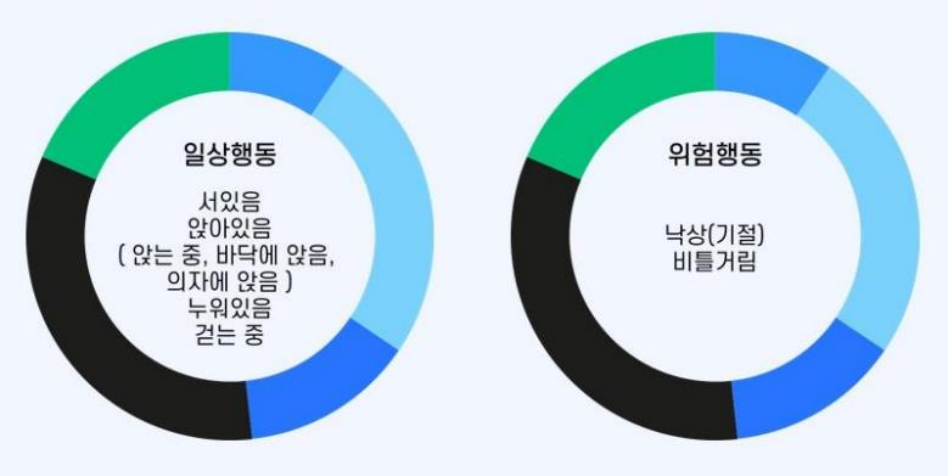


그림 1 수집할 데이터에 대한 구체적인 분류

mmwave 센서를 사용하여 선정한 동작에 대한 timestamp, cloud point 정보를 수집한다.

2.3. 개발환경

2.3.1. 개발 언어

개발 언어로는 Python과 Java를 선택하였다. Python은 데이터 수집, 처리, 서버구축 및 통신에 사용하였고 Java는 Android 앱 개발에 사용하였다.

2.3.2 안드로이드 스튜디오

안드로이드 스튜디오(Android Studio)는 안드로이드 및 안드로이드 전용 앱 제작을 위한 공식 통합 개발 환경(IDE) 이다. mmwave 센서를 이용해 측정한 데이터에 대한 행동 예측 결과를 휴대전화로 보여주기 위해 사용하였다.

2.3.2. Flask

파이썬 기반으로 작성된 마이크로 웹 프레임워크이다. 간단한 웹 사이트나 간단한 API 서버를 만드는데 특화되어 있다. mmwave 데이터를 받아 모델을 적용해 어떤 동작인지 파악하고 서버를 통해 휴대전화로 전달하기 위해 사용하였다.

2.3.3. MySQL

전 세계적으로 가장 널리 사용되고 있는 오픈소스 관계형 데이터베이스 관리 시스템(RDBMS: Relational DBMS)이다. 매우 빠르고, 유연하며, 사용하기 쉬운 특징이 있다. 위험 행동이 일어난 시간 및 현재 행동에 대한 정보를 저장하고 관리하기 위해 사용하였다.

3. 연구 내용

3.1. 데이터 수집

3.1.1. 데이터 수집 방법

앞에서 언급하였듯이 mmwave 센서와 Kinect 카메라로 동시에 데이터를 수집하였다. mmwave 센서의 경우 정적인 동작을 잘 인식하지 못하는 경우가 있어 큰 동작을 위주로 데이터를 수집하였다. 또한, Kincet 카메라의 경우 거리 또는 사물에 의해 사람의 skeleton 구조가 잘 인식되지 않는 경우가 발생할 수 있어 근처에 아무런 사물이 없는 환경에서 데이터 수집을 진행하였다.

조원이 총 3명이므로 3명이서 각각 2m, 3m 거리에 따른 2가지의 위험행동(낙상, 비틀거림)과 6가지의 일상행동(서있음, 바닥에 앉음, 의자에 앉음, 누워있음, 걷는 중, 뛰는 중)으로 분류하여 데이터를 수집하였다.

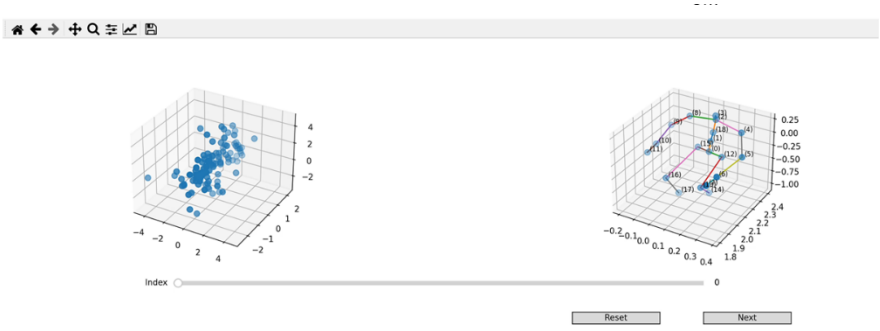


그림 2 수집된 데이터 모양 (왼쪽 - mmwave, 오른쪽 – Kinect)

그림 3 데이터를 수집하는 모습

3.1.2. 데이터 정보

mmwave 센서를 통해 동작에 대한 cloud point 정보 x좌표, y좌표, z좌표, velocity, SNR, Range를 numpy배열형태로 수집하였으며, 각 동작에 대해 약 6~8개 씩 총 약 228개 정도의 데이터를 수집하였다.

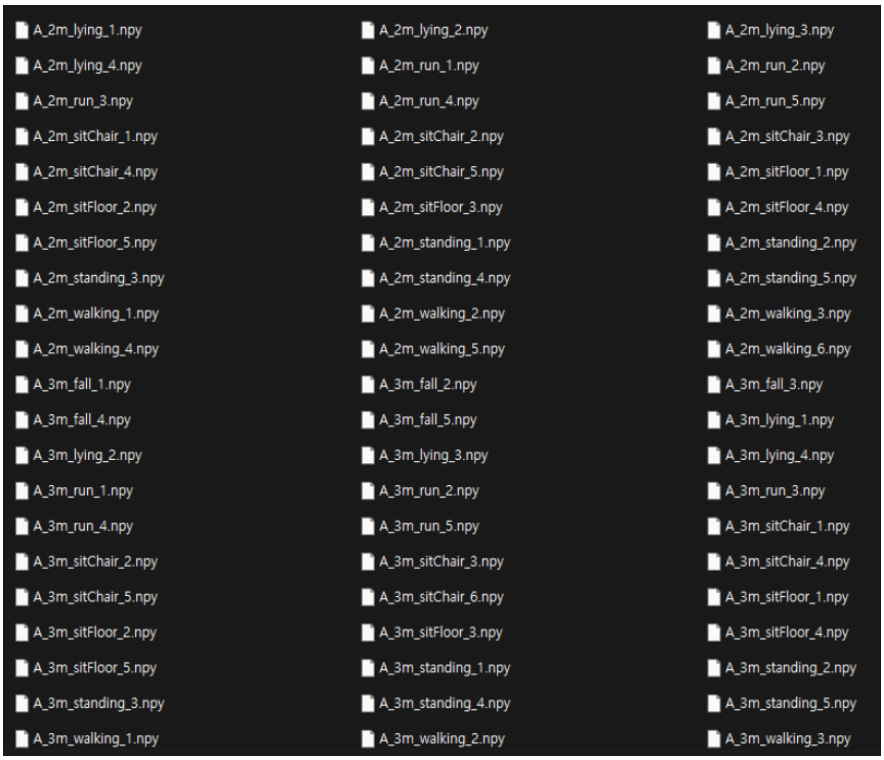


그림 4 수집한 데이터

3.2. 데이터 전처리

3.2.1. 데이터 길이 조절

mmwave 센서로 수집된 데이터는 numpy배열로 모양은 (frame 수, timestamp, (128, 6)) 이다. 이때 (128, 6)은 한 frame 당 무작위로 128개의 point에 대한 x좌표, y좌표, z좌표, 속도, noise, range 정보를 의미한다. 수집된 데이터의 frame 수는 동작마다 다르기 때문에 모델 학습을 진행하기 전에 이에 대한 전처리를 해줘야 한다. 수집된 데이터의 최대 frame 수는 165이고 대부분 120이므로 아래와 같이 전처리를 진행해 주었다.

- Frame 수가 120 미만인 경우
  - 모자란 frame 수만큼 0으로 채워 120으로 맞춰준다.
- Frame 수가 120 이상인 경우

- 120을 넘는 frame부터는 drop 하여 120으로 맞춰준다.

```
def pad_along_axis(array: np.ndarray, target_length: int, axis: int = 0) -> np.ndarray:
    pad_size = target_length - array.shape[axis]
    if pad_size <= 0:
        return array
    npad = [(0, 0)] * array.ndim
    npad[axis] = (0, pad_size)
    return np.pad(array, pad_width=npad, mode='constant', constant_values=0)

# time wise zero padding
X_pad_120 = []
for x in X:
  if x.shape[0] < 120:
    x = pad_along_axis(x, 120, axis=0)
  elif x.shape[0] >= 120:
    x = x[:120]
  X_pad_120.append(x)
X_pad_120 = np.array(X_pad_120)
```

그림 5 데이터 길이를 조절하는 전처리 코드

3.2.2. DBScan

수집된 cloud point 데이터는 맨눈으로 결과를 확인했을 때 동작이 잘 인식되었는지 확인하기 어렵기 때문에 전처리하지 않은 raw data, 정규화만 한 data, DBSCAN 및 정규화를 통해 데이터 전처리한 data를 모두 학습 모델에 적용하여 정확도를 비교하여 최종 모델 학습에 사용할 데이터형식을 선정하기로 하였다.

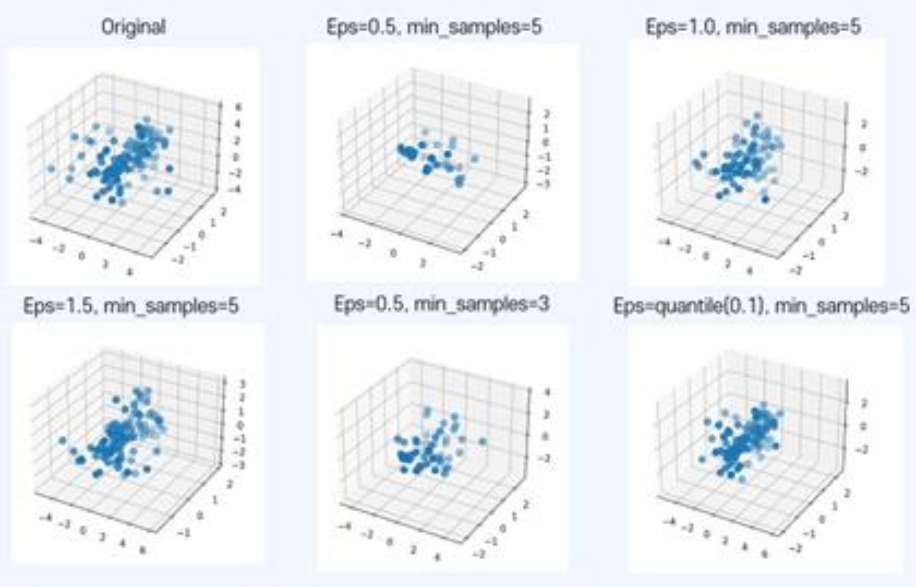


그림 6 DBSCAN 적용 후 출력한 point cloud 데이터

모델 학습에 적합한 데이터를 선정하기 위한 임시 모델로는 아래 구조와 같은 3D CNN 모델을 사용하였다.

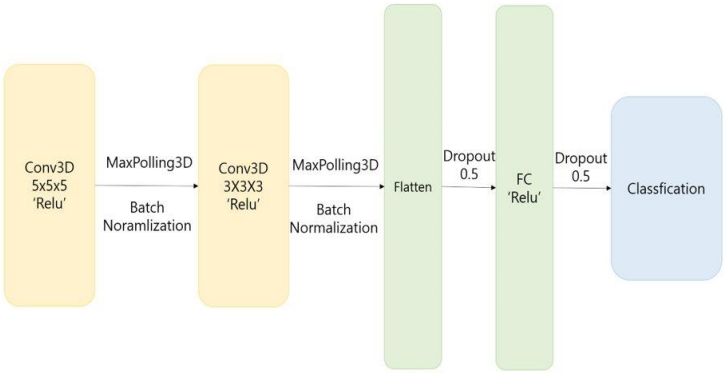


그림 7 전처리 결과 확인을 위한 3D CNN모델 구조

임시 모델에 raw 데이터, 정규화된 데이터, DBSCAN 및 정규화의 데이터를 입력하여 나온 결과는 다음과 같았다. raw 데이터가 정확도 69.56% 정도로 가장 높게 나왔고, 정규화된 데이터의 경우 39.13%로 가장 낮았으며, DBSCAN 및 정규화를 진행한 데이터는 DBSCAN의 파라미터에 따라 58.9%, 54.34%, 52.17%, 54.34%라는 결과가 나왔다. 따라서 아무런 처리를 진행하지 않은 raw 데이터를 사용하여 최종 모델 학습을 진행하기로 하였다.

DATA 종류	LOSS	ACCURACY
RAW DATA 학습	1.3124	0.6957
RAW DATA 정규화	2.2858	0.3914
DBSCAN EPS=0.5, MIN=4	1.8587	0.5870
DBSCAN EPS=0.8, MIN-4	1.3496	0.5435
DBSCAN EPS=QUANTILE(0.2), MIN=4	2.1091	0.5217
DBSCAN EPS=QUANTILE(0.1), MIN=4	2.1547	0.5435

표 1 raw 데이터 및 전처리 후 데이터들의 Loss와 정확도

3.3. 모델 학습

3.3.1. 모델 설계

수집된 데이터 학습을 위해 3D CNN, 2D CNN, LSTM 총 3개의 형태 모델이 고려되었다. 이 모델들의 특징들은 다음과 같으며 가장 좋은 성능의 모델을 찾기 위해 3가지 모델 모두 사용하여 학습을 진행하였다.

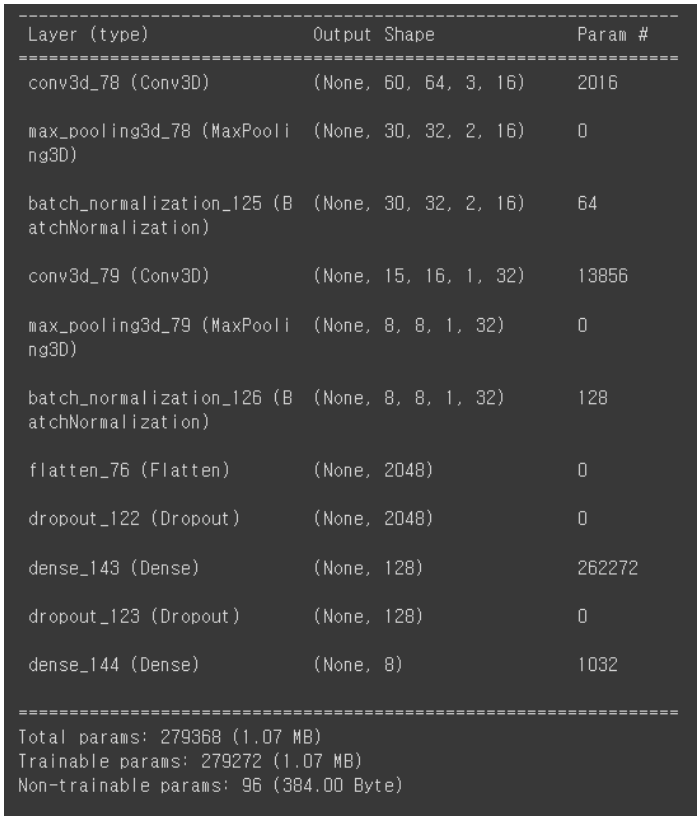
- 3D CNN
  - 3차원 데이터 처리, 시공간 데이터 처리, 시간적인 특성 학습할 수 있다.
- 2D CNN
  - 2차원 데이터 처리, 이미지의 공간적 패턴 학습이 가능하다
- LSTM
  - 순차적인 데이터나 시계열 데이터 등 시간적인 패턴 학습이 가능하다.

또한 수집한 데이터가 각 포인트에 대한 (x, y, z, v SNR, Range) 총 6가지의 정보를 담고 있으므로 3가지 모델에 대해 (x, y, z) / (x, y, z, v) / (x, y, z, v, SNR) / (x, y, z, v, SNR, range) 데이터에 담긴 정보를 다르게 하여 학습하였다.

3.3.2. 모델 학습 및 결과

클래스 수와 데이터 크기에 비해 데이터 개수가 매우 적으므로 validation은 생략하고 train set 80%, test set 20%로 모델 학습 및 평가를 진행하였으며 train set과 test set을 나눌 때 stratify 옵션을 사용하여 클래스들이 골고루 샘플링되도록 하였다.

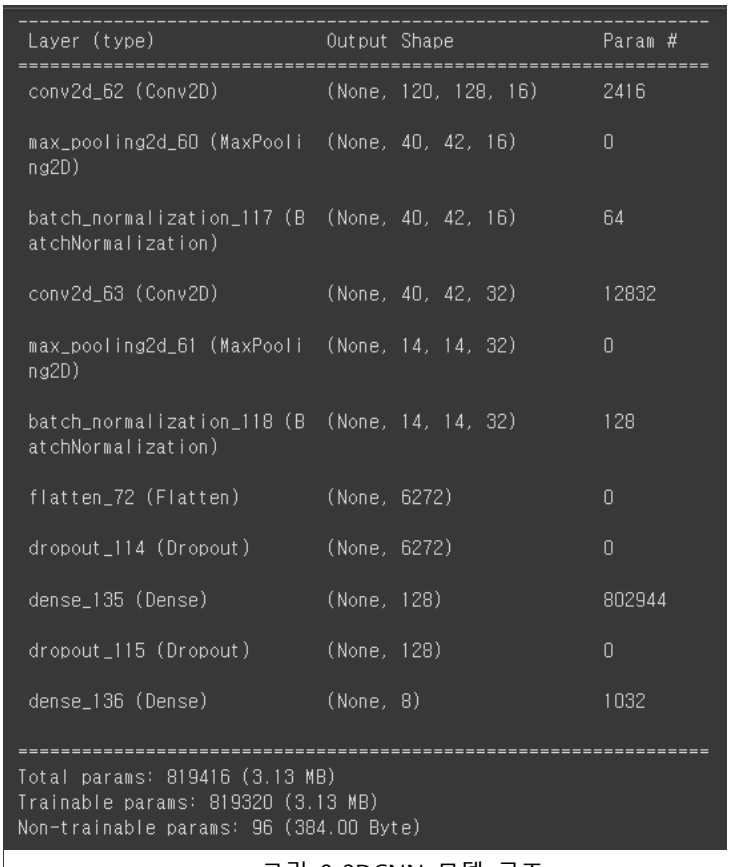
모델 학습은 tensorflow를 사용하여 Google Colab에서 수행하였으며 아래 이미지와 같은 구조의 모델 3가지를 학습시켰다.



Layer (type)	Output Shape	Param #
conv3d_78 (Conv3D)	(None, 60, 64, 3, 16)	2016
max_pooling3d_78 (MaxPooling3D)	(None, 30, 32, 2, 16)	0
batch_normalization_125 (BatchNormalization)	(None, 30, 32, 2, 16)	64
conv3d_79 (Conv3D)	(None, 15, 16, 1, 32)	13856
max_pooling3d_79 (MaxPooling3D)	(None, 8, 8, 1, 32)	0
batch_normalization_126 (BatchNormalization)	(None, 8, 8, 1, 32)	128
flatten_76 (Flatten)	(None, 2048)	0
dropout_122 (Dropout)	(None, 2048)	0
dense_143 (Dense)	(None, 128)	262272
dropout_123 (Dropout)	(None, 128)	0
dense_144 (Dense)	(None, 8)	1032

Total params: 279368 (1.07 MB)
Trainable params: 279272 (1.07 MB)
Non-trainable params: 96 (384.00 Byte)

그림 8 3DCNN 모델 구조



Layer (type)	Output Shape	Param #
conv2d_62 (Conv2D)	(None, 120, 128, 16)	2416
max_pooling2d_60 (MaxPooling2D)	(None, 40, 42, 16)	0
batch_normalization_117 (BatchNormalization)	(None, 40, 42, 16)	64
conv2d_63 (Conv2D)	(None, 40, 42, 32)	12832
max_pooling2d_61 (MaxPooling2D)	(None, 14, 14, 32)	0
batch_normalization_118 (BatchNormalization)	(None, 14, 14, 32)	128
flatten_72 (Flatten)	(None, 6272)	0
dropout_114 (Dropout)	(None, 6272)	0
dense_135 (Dense)	(None, 128)	802944
dropout_115 (Dropout)	(None, 128)	0
dense_136 (Dense)	(None, 8)	1032

Total params: 819416 (3.13 MB)
Trainable params: 819320 (3.13 MB)
Non-trainable params: 96 (384.00 Byte)

그림 9 2DCNN 모델 구조

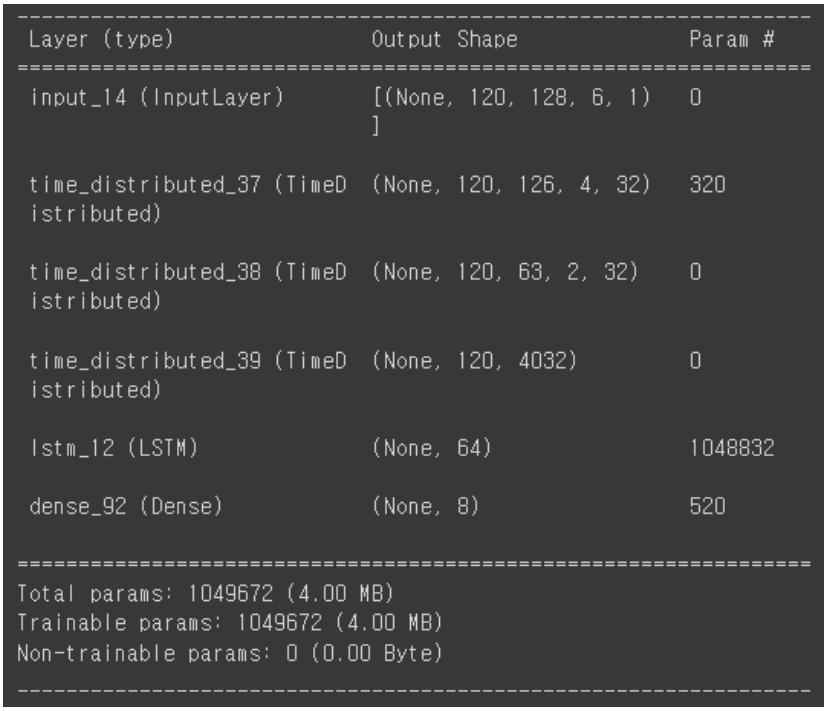


그림 10 LSTM 모델 구조

모델 학습 결과 (x, y, z, v, SNR, range)를 모델의 입력값으로 사용했을 때 다른 데이터들을 입력값으로 사용한 경우보다 3가지 종류의 모델에서 좋은 결과를 얻을 수 있었다.

		X, Y, Z	**X, Y, Z, V**	**X, Y, Z, V, SNR**	**X, Y, Z, V, SNR, RANGE**
3D CNN	Loss	0.9921	1.3122	0.8302	0.6028
	Accuracy	0.7826	0.6957	0.7174	0.8260
2D CNN	Loss	1.4827	1.7461	2.1651	0.7816
	Accuracy	0.7609	0.7609	0.7609	0.8043
LSTM	Loss	1.1979	1.2071	0.9300	0.6746
	Accuracy	0.6739	0.6739	0.6739	0.8043

표 2 3가지 모델에 대한 여러 입력의 결과

(x, y, z, v, SNR, range)를 입력값으로 학습한 모델 중 3D CNN 모델의 결과가 가장 우수하였다. 따라서 본 과제의 예측 모델로 (x, y, z, v, SNR, range)를 입력값으로 하는 3D CNN 모델을 사용하기로 하였다.

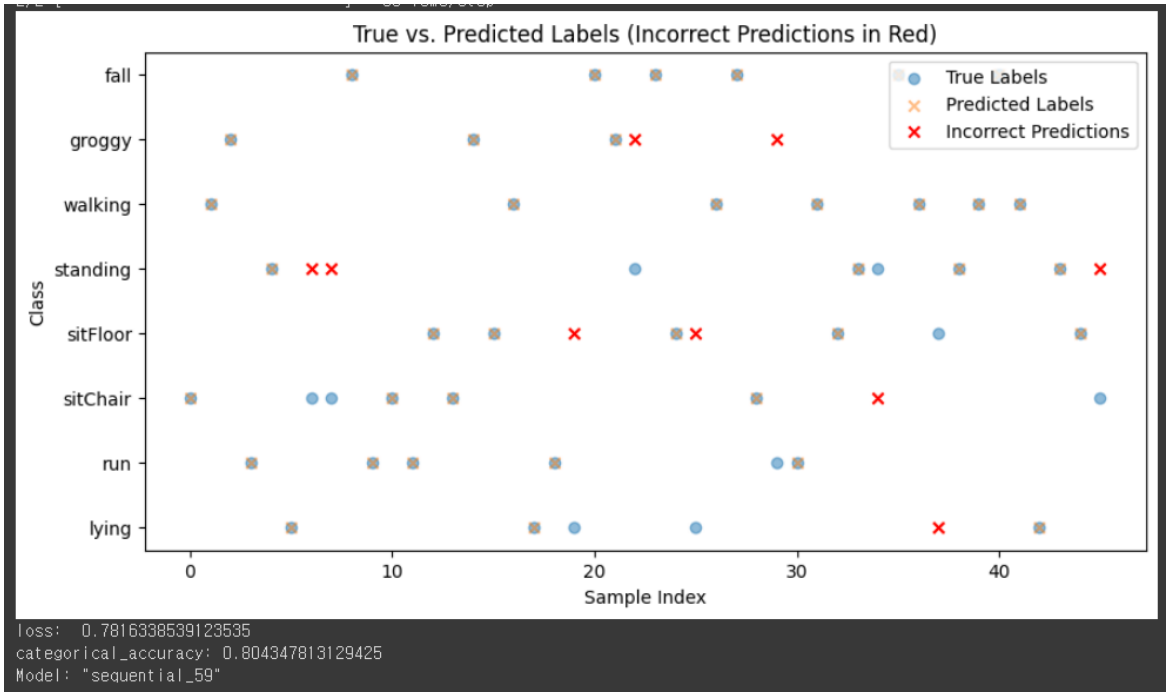


그림 12 (x, y, z, v, SNR, range) 데이터를 사용한 2DCNN 모델 결과

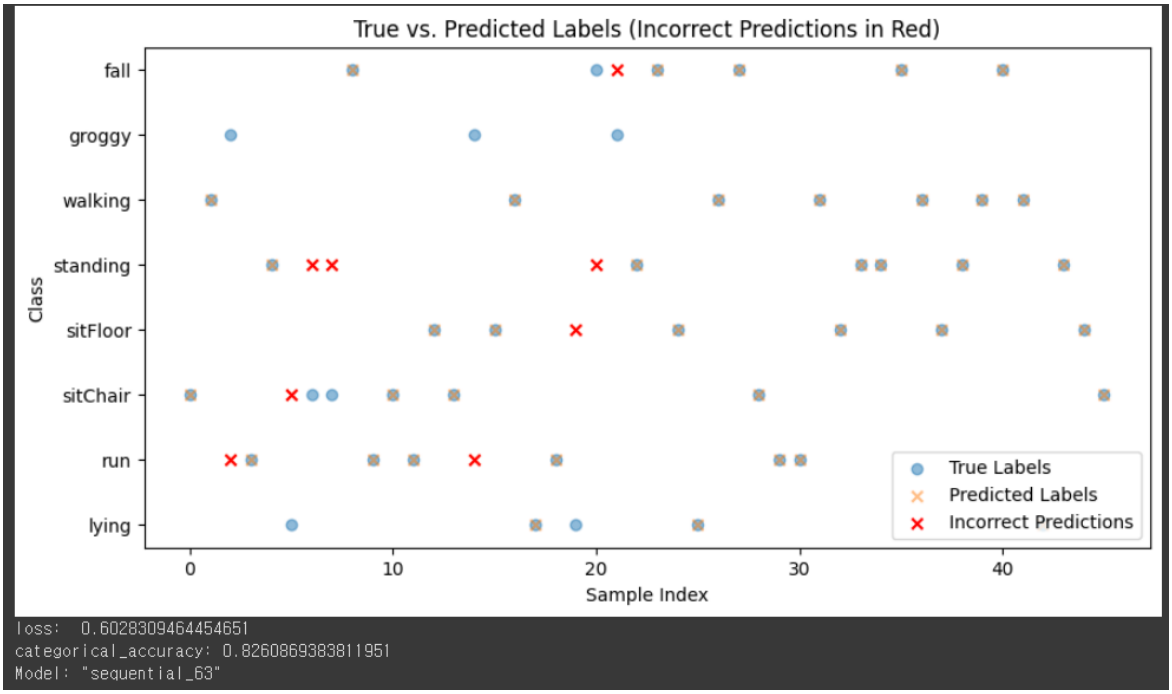


그림 11 (x, y, z, v, SNR, range) 데이터를 사용한 3DCNN 모델 결과

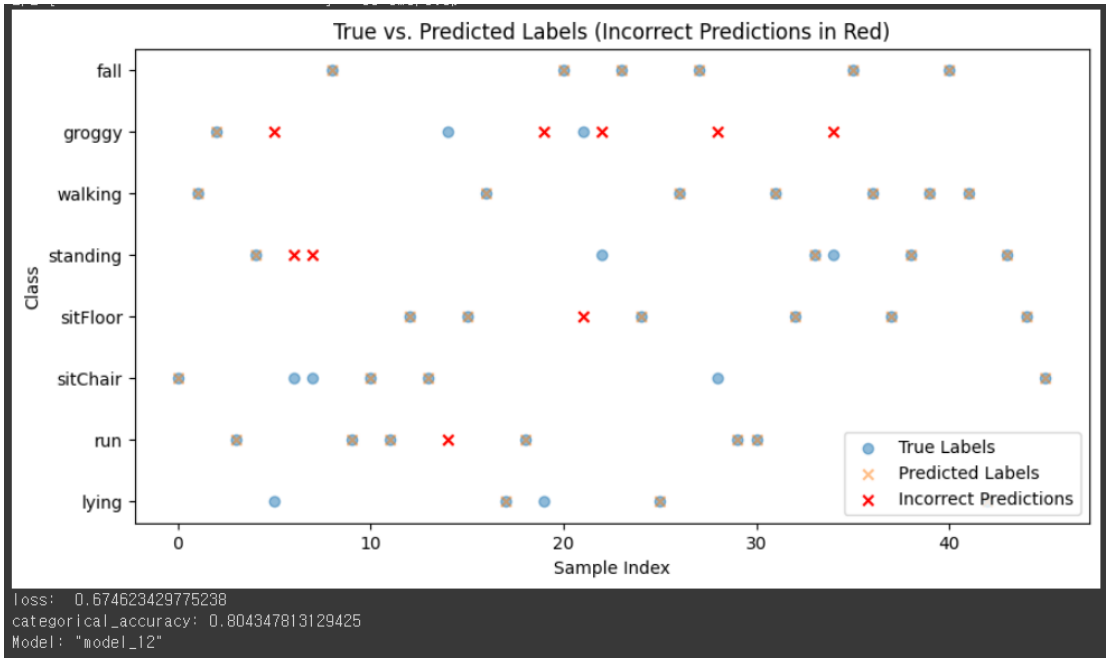


그림 13 (x, y, z, v, SNR, range) 데이터를 사용한 LSTM 모델 결과

3.4. 안드로이드 앱

안드로이드 화면에서는 현재 동작 정보에 대해 보여주는 화면, 위험 행동 시 문자를 전송할 전화번호를 저장할 화면, 그리고 위험 행동 시간 기록을 볼 수 있게 화면을 구성하였다. 사용자 요청에 따라 현재 행동 데이터 또는 위험 행동 기록 데이터 리스트를 서버에게 요청한다. 이때 HTTP 통신 구조를 이용해 앱에서는 GET 요청을 보내고 Flask 서버에서는 요청에 맞는 기능을 수행하여 DB로부터 알맞은 정보를 응답으로 앱에 돌려준

그림 16 낙상기록 화면

그림 15 현재 동작 화면

그림 14 전화번호 등록 화면

다.

위험 행동이 발생하는 경우 핸드폰 내부 알람으로 위험 행동이 발생하였음을 알리고 등록된 번호로 위험 행동이 발생하였다고 문자를 보내준다.

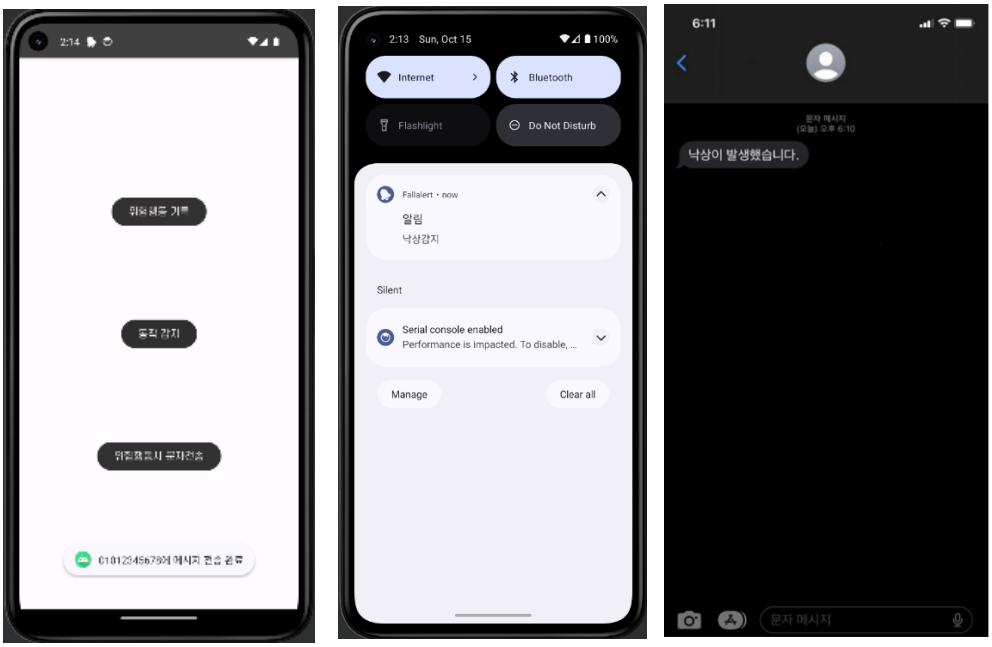


그림 17 위험행동 발생 시 알림 및 메시지가 전송된 화면

3.5. DB 및 서버 구축

3.5.1. DB

MySQL을 사용하여 동작 및 위험 행동 기록을 저장할 데이터베이스를 생성하였다. 현재 동작을 기록하는 action 테이블은 추론 결과에 대한 Label 번호와 동작 이름에 대한 정보를 담는다. 위험 행동을 기록하는 abnormal_action 테이블은 위험 행동에 대한 Id, 위험 행동 종류, 발생 시간에 대한 정보를 담는다.

action_num	action_name
0	Lying

그림 18 action 테이블

abnormal_ID	abnormal_name	time
1	Fall	2023-10-14 23:38:43
2	Groggy	2023-10-14 23:42:39
3	Grogy	2023-10-15 03:21:37
4	Groggy	2023-10-15 04:12:47
5	Fall	2023-10-15 04:13:10

그림 19 abnormal_action 테이블

3.5.2. 서버

사용자 요청에 따라 데이터를 저장하고 관리하기 위한 서버가 필요하다. 데이터 전처리 코드와 딥러닝 모델을 적용하기 편리한 python기반의 웹 프레임워크인 Flask를 사용했다. mmwave 센서를 통해 동작에 대한 데이터가 수집되면, POST 방식으로 전처리된 데이터를 서버로 전송하고 서버에 업로드 되어있는 모델을 통해 어떤 행동인지 추론한다. 이렇게 추론된 결과를 현재 동작에 대한 정보를 담아두는 action 테이블 DB에 저장하고 만약 추론된 결과가 위험 행동(낙상, 비틀거림)이라면 위험 행동 종류 및 발생 시간을 abnormal_action 테이블 DB에 저장해 준다. 이후 앱으로부터 현재 동작 및 위험 행동 기록이 요청되면 알맞은 정보를 응답으로 전달해 준다.

```
<Response 36 bytes [200 OK]>
127.0.0.1 - - [15/Oct/2023 23:17:37] "GET /get_data HTTP/1.1" 200 -
```

그림 20 서버에 앱으로부터 GET 요청이 온 모습

4. 연구 결과 분석 및 평가

처음에 (x, y, z) 세 좌표를 통해 raw data를 가지고 학습을 진행했을 때 약 70%의 정확도를 보이는 것의 성능을 올리기 위해 DBSCAN으로 outlier를 제거 하는 등의 전처리를 진행하였으나 오히려 성능 저하가 되었고 향상시키기 위해 여러 모델을 적용, 수집한 데이터의 velocity, SNR(Signal-to-noise ratio), Range 정보 입력 추가, train set과 test set을 나눌 때 stratify 옵션을 사용하여 클래스들이 골고루 샘플링되도록 하는 등 다양한 방법을 시도하였다. 이를 통해 80% 이상의 정확도까지 올렸으며, 해당 모델을 사용하여 동작을 추론한 뒤 이 결과를 핸드폰 앱을 통해 사용자가 확인할 수 있고 위험 행동 발생 시 알림과 문자가 전송되도록 하는데 성공하였다. 하지만 이를 실생활에 적용하여 사용하기에는 매우 부족한 정확도의 모델이라고 생각한다. 이러한 결과의 원인 중 하나로는 8개의 클래스에 대해 총 228개의 데이터를 사용하여 모델 학습을 진행하기에는 데이터 개수가 부족하였기 때문이라고 생각된다. 이는 프로젝트에 사용할 만한 적절한 오픈 데이터를 수집하기 어려워 직접 데이터를 수집했으나, 더욱 많은 데이터를 수집하기엔 시간적, 공간적 제한으로 인한 발생한 결과라고 생각된다. 또한 데이터 수집을 할 때 모든 동작을 비슷한 시간을 정해두고 측정하여 데이터 전처리 시에 데이터 손실을 막았으면

더 좋은 결과를 얻을 수 있지 않았을까 하는 아쉬움이 있다.

그림 22 앱 동작 화면

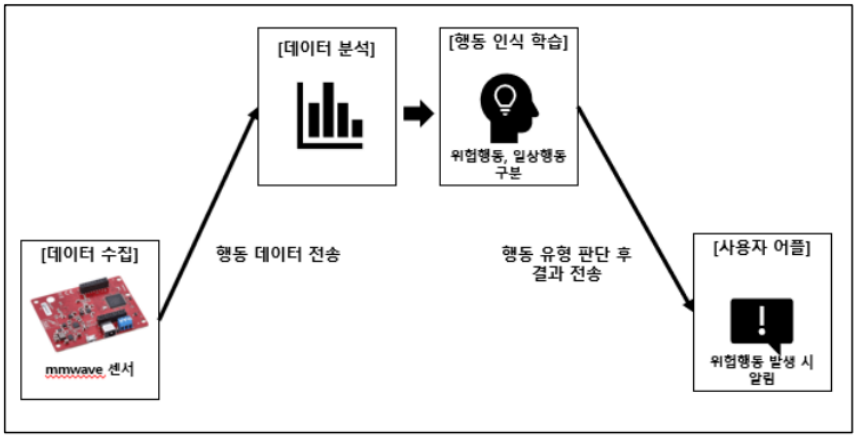


그림 21 시스템 구성도

5. 결론 및 향후 연구 방향

5.1. 결론

본 졸업 과제에서는 mmwave 센서를 사용하여 동작 데이터를 수집하고 수집된 데이터를 서버로 전송하여 학습된 모델을 통해 동작을 추론하였다. 추론 결과는 앱을 통해 사용자가 확인할 수 있으며 위험 행동이 발생하면 발생 시간을 기록하고 핸드폰 알림과 메시지로 알리는 기능까지 구현하였다. 이 과정에서 딥러닝을 위한 데이터 수집부터 전처리, 모델 설계 및 학습까지 경험할 수 있었다. 추론된 정보를 사용자에게 전달하기 위한 서버와 DB를 구축하고 앱을 개발하였고 직접 개발한 앱을 통해 mmwave가 전송한 데이터에 대한 동작 추론이 잘 작동함을 확인할 수 있었다.

5.2. 향후 연구 방향

본 과제에서는 시간과 공간적 문제로 충분한 데이터 수집을 할 수 없어 예측 모델의 정확도가 80% 정도에 그치지만, 추후 충분한 데이터 수집 후 모델 학습을 다시 진행한다면 더 좋은 동작 추론 결과를 얻을 수 있을 것으로 생각된다. 또한 위험 행동 발생 시, 위험 행동이 감지된 위치 정보를 함께 전송하여 신속한 도움을 받을 수 있도록 알림 및 메시지에 위치 정보를 포함한다면 더 좋을 것 같다.

mmwave 센서를 사용한 데이터 수집 부분에서 지금은 실시간으로 동작에 대한 데이터가 수집되는 것이 아닌 직접 기록 시작, 종료를 해주어야 한다. 향후 연구에서는 이를 실시간으로 특정 프레임 길이만큼 기록할 수 있도록 수정할 예정이다. 또한 현재는 서버

에 모델을 올려 동작을 추론하지만, 경량화와 같은 모델 수정을 통해 mmwave 장치에 직접 올려 동작을 추론할 수 있게 하는 것이 목표이다.

6. 구성원별 역할 및 개발 일정

6.1. 구성원별 역할

이름	역할분담
강수빈	데이터 전처리 어플리케이션 UI 개발
박준형	서버 및 DB구축 어플리케이션 서버 연동
이영인	서버 및 DB구축 어플리케이션 서버 연동
공통	mmwave데이터 수집 및 분석 행동인식 머신러닝 학습 학습 모델 개선 및 수정

표 3 구성원별 역할

6.2. 개발 일정

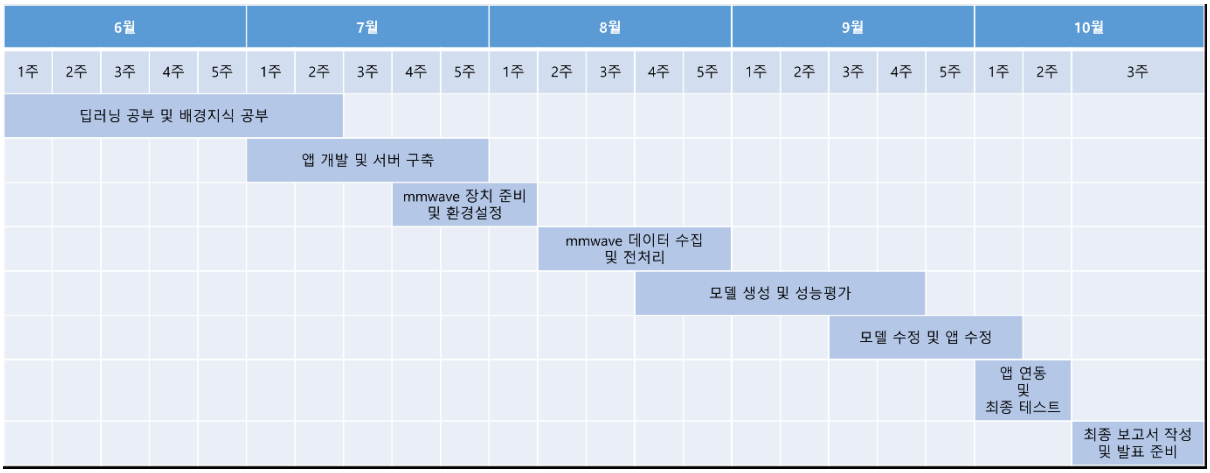


그림 23 개발 일정

7. 참고 문헌

[1] Sizhe An, Yin Li, Umit Ogras. mRI: Multi-model 3D Human Pose Estimation Dataset using mmWave, RGB-D, and Inertial Sensors.
[2] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters
in Large Spatial Databases with Noise.
[3] Shih-Po Lee, Niraj Prakash Kini, Wen-Hsiao Peng, Ching-Wen Ma, Jenq-Neng Hwang. HuPR: A Benchmark for Human Pose Estimation Using Millimeter Wave Radar.
[4] Google. Android Developers [Online]. Available : https://developer.android.com/
[5] Flask. Flask Documentation [Online]. Available : https://flask.palletsprojects.com/en/3.0.x/