

졸업프로젝트 결과 보고서

작품 제목	보행 패턴을 이용한 개인식별
작품 기능	사람마다 보행 시 팔 움직임이 다르다. 팔 동작으로 개인을 식별하게 되면 보안과 편의성을 높일 수 있다. 개인의 정보를 통해 빅데이터의 개념과 규모 역시 익힐 수 있다.
작품 계획 기간	2016년 02월 01일 ~ 2016년 02월 25일
작품 제작 기간	2016년 03월 01일 ~ 2016년 10월 21일
개발 환경	OS: Windows 7 언어: JAVA, python Tools: Android Studio, Eclipse, Docker container

이름	구분	학교	학과	학년	전화번호
한중수	PL	한양대학교	컴퓨터전공	4	010-2079-5292

한양대학교 공과대학 컴퓨터공학부

졸업프로젝트 자가 평가서

항 목	평가 점수
1. 성실성 (개발목표 대비 완료와 정리: 20점 만점) 개발 규모와 범위가 넓었기 때문에 계획이 중요했다. 구성은 안드로이드 부분, 윈도우 서버 부분, 그리고 딥러닝 학습 서버 부분으로 되어 있다. 딥러닝과 파이썬 노트북 활용과 같이 새로운 분야에 대해 공부할 시간을 가졌고, 시기별 계획에 큰 차질 없이 모든 개발을 완료하였기 때문에 높은 점수를 주었다.	19점
2. 기술성 (우수한 개발 능력과 기술 구현: 25점 만점) 학부 과정에서 배우지 못한 지식들을 공부하게 되었다. 딥러닝과 안드로이드 프로그램, 파이썬 서버 프로그램등을 사용하였다. 과정에서 Docker 라는 새로운 가상머신을 활용하여 개발하였다. 추가로 딥러닝 과정에서 보행 패턴을 학습하는 방법을 독자적으로 개발하여 적용하였기 때문에 높은 점수를 주었다.	24점
3. 창의성 (구현의 창의성과 차별성: 25점 만점) 개인이 일상생활에서 경험할 수 있는 데이터를 대규모로 뽑아내고, 딥러닝 학습을 한다는 것이 새로운 시도라고 생각했다. 또한 보행 데이터들을 학습 데이터로 가공하는 새로운 방법을 생각해야 했다. 이를 완성시켜 성공적으로 구현했다.	23점
4. 응용성 (개발 결과물에 대한 현업 적용: 20점 만점) 개발을 하던 중, 애플에서 후세대 IOS에 보행인식을 도입한다는 뉴스를 보았다. 개인으로 진행하였기 때문에 정확성에 대한 오차가 있었지만, 보정이 된다면 보안과 편리성 측면에서 활용도가 높을 것이다. 뿐만 아니라 보행 패턴과 같은 추상적이고 연속적인 데이터를 가공하는 방법 역시 학습 방법에 응용될 수 있을 것이다.	16점
5. 사업성 (향후 활용성 및 사업화: 10점 만점) 인식에서 중요한 것은 인식 정확성과 악성인식 거부에 있다. 이러한 체계는 정확도와 높은 완성도가 중요하다. 이는 거대한 데이터로 최적화해야 만족스러운 결과를 얻을 수 있을 것이다. 하지만 내가 진행했던 보행 학습은 규모가 작고 데이터가 불충분하여 바로 활용하고 사업화를 하지는 못할 것이라 생각했다.	5점

목 차

1. 과제 개요
 - 1.1 개발 목적 및 동기
 - 1.2 개발 목표
2. 과제 내용
 - 2.1 안드로이드 어플리케이션
 - 2.1.1 가속도 데이터 전송
 - 2.1.2 서버로부터 승인
 - 2.1.3 통신의 액티비티 구성
 - 2.1.4 안드로이드 UI
 - 2.2 윈도우 서버
 - 2.2.1 안드로이드와 블루투스 통신
 - 2.2.2 기기별 ID값으로 데이터 저장, 관리
 - 2.2.3 딥러닝 서버로 parameter 전달
 - 2.2.4 서버 UI
 - 2.3 딥러닝 서버
 - 2.3.1 딥러닝 환경 구축
 - 2.3.2 딥러닝 학습 모델
 - 2.3.3 모델 저장 및 불러오기
 - 2.3.4 오차계산
3. 기술 내용
 - 3.1 안드로이드 어플리케이션
 - 3.1.1 싱글톤 액티비티 구성
 - 3.1.2 통신 데이터 패딩
 - 3.2 윈도우 서버
 - 3.2.1 주피터 서버 파라미터 전달
 - 3.3 딥러닝 서버
 - 3.3.1 컨테이너 생성
 - 3.3.2 RNN 학습 데이터와 학습 라벨 만들기
 - 3.3.3 RNN 학습 모델 저장하기, 불러오기
 - 3.3.4 오차 계산하기
4. 세부 역할
 - 4.1 개별 임무 분담
 - 4.2 개발 일정
5. 결론

1. 과제 개요

1.1 개발 목적 및 동기

현재 널리 구현되어 있는 개인 식별 기술엔 모방성이 존재한다. 지문 인식의 경우 실리콘 지문으로 생각보다 간단하게 모방할 수 있으며 가장 간단한 형태인 개인 ID 카드의 경우 대여를 통해 간단하게 무력화 된다. 나는 모방 불가능한 개인 식별을 목표로 하여 보안적인 면에 초점을 맞췄다.

또한 체계를 의도적으로 악용할 수 없도록 하고 싶었다. 철없을 때 대학교 대리출석을 한 경험이 있었다. 이는 편의성을 통해 만들어진 일종의 보안 시스템을 악용한 경우다. 이러한 의도적인 악용을 막고자 하는 과정에서 편리성 확보의 측면을 생각했다. 만약 고속도로의 하이패스처럼 개인이 편의를 얻을 수 있다면 굳이 악용할 이유가 없을 것이다.

1.2 개발 목표

이번 과제의 목표는 보행패턴을 통한 개인 식별이다. 환승게이트나 도서대출 입구를 예로 들어보자. 일정 거리 전에 개인 식별이 가능하다면 하이패스와 같이 카드를 꺼내서 찍지 않아도 '그대로 걸어감'으로서 인증이 완료된다.

스마트폰은 대다수가 가지고 있는 물건이다. 본 과제는 개인이 일상적으로 사용하는 안드로이드의 센서를 활용하여 보행패턴 인식을 구현하는데 있다. 추구하는 목표는 다음 세 가지이다.

1. 모방 불가능한 개인 식별을 통한 보안성 확보
2. 분별 가능한 개인 식별을 통한 편리성 확보
3. 의도적으로 악용할 수 없는 보안성 확보

프로젝트의 구체적인 목표를 세웠다. 열 걸음 걷기 전에 누군지 식별해 보기.

2. 과제 내용

입력 데이터는 팔 움직임에 따른 안드로이드의 방위 값, 가속도 값의 빅데이터이다. 보행자는 연속적인 센서 수치들을 반복 입력하게 된다. 머신 러닝을 통해 모션 패턴을 분석한 후, 개인마다의 가중치 값을 결과로 얻게 되어 최종적으로 개인 식별을 하게 된다.

키 차이에 의한 팔 높이차, 걷는 속도에 따른 가속도 차이, 팔 벌리는 각도 혹은 팔 굽는 각도 차이는 구분 지어질 수 있는 차이를 만들 것이다. 전체 코드와 실행파일은 CD에 첨부하였다.

2.1 안드로이드 어플리케이션

2.1.1 가속도 데이터 전송

안드로이드는 실시간 가속도 값을 블루투스 통신을 통해 윈도우 서버로 전송한다.

2.1.2 서버로부터 승인

서버로부터 승인 신호를 대기한다.

서버로부터 받은 식별값을 저장한다.

2.1.3 통신의 액티비티 구성

액티비티간, 그리고 서버와의 주고받는 데이터를 공유 액티비티를 설정하여 관리한다.

2.1.4 안드로이드 UI

다중 프래그먼트를 사용하여 UI를 보여준다.

2.2 윈도우 서버

2.2.1 안드로이드와 블루투스 통신

동시에 여러 안드로이드와 통신 채널을 구성한다.

2.2.2 기기별 ID값으로 데이터 저장, 관리

기기별로 가속도 데이터를 관리하는 프로필을 구성한다. 기기 연결 시 프로필과 대조한다.

2.2.3 딥러닝 서버로 parameter 전달

ID값을 동일 딥러닝 서버로 넘겨준다. 프로필을 만들것인지, 식별을 할 것인지 선택한다.

2.2.4 서버 UI

관리자가 명령을 내릴 수 있는 콘솔화면을 보여주고, 상태를 알 수 있는 윈도우를 구성한다.

2.3 딥러닝 서버

2.3.1 딥러닝 환경 구축

딥러닝을 학습하기 위한 리눅스 서버를 구축한다. Docker Container를 사용하였다.

2.3.2 딥러닝 학습 모델

가속도 값들의 행렬을 학습시키기 위해 test set과 test label로 데이터를 가공했다.

2.3.3 모델 저장 및 불러오기

학습된 모델을 시간을 절약하기 위해 저장해 놓고, 기기별로 비교 시 불러온다.

2.3.4 오차계산

식별할 보행 데이터를 입력받은 후 모델과 비교하여 오차를 산출한다.

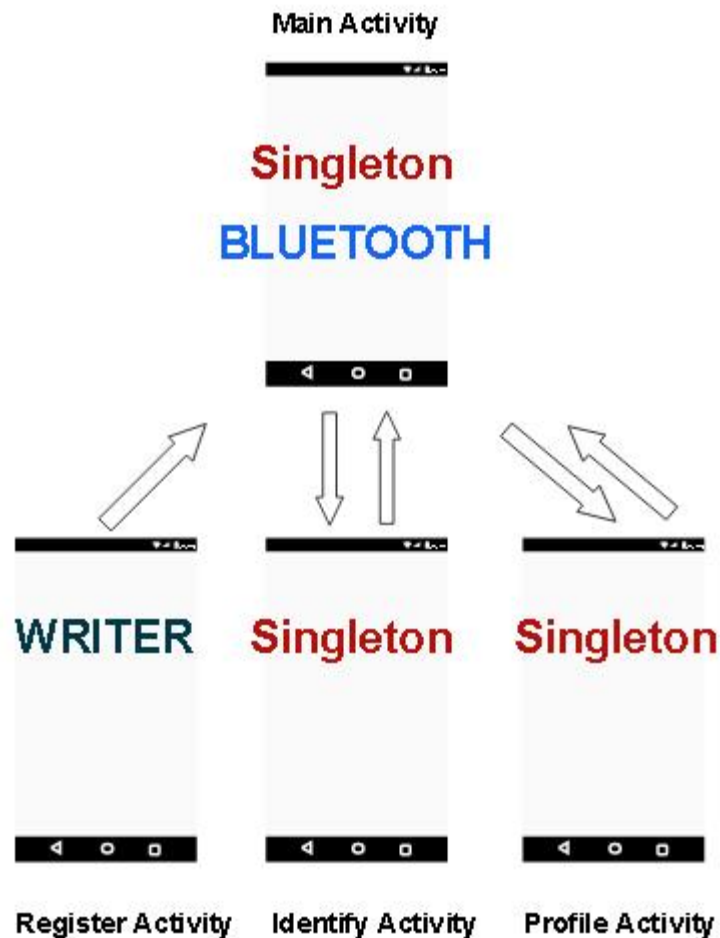
3. 기술 내용

3.1 안드로이드 어플리케이션

3.1.1 싱글톤 액티비티 구성

액티비티끼리 자주 Context가 바뀌고, 반면 통신은 연결되어야 했다. 액티비티를 생성자를 이용해 새로 만들게 된다면 액티비티 수명주기상 onCreate(), onStart(), onResume()으로 이어지는 함수가 자동으로 호출된다. 각 액티비티별로 한 차례만 초기화 된 후 instance를 유지하기 위하여 싱글톤 디자인 패턴을 적용하게 되었다.

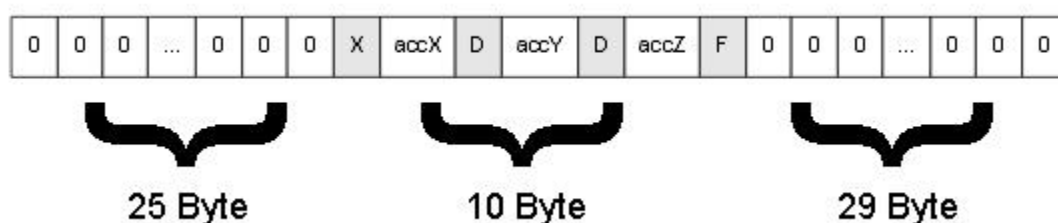
메인 액티비티에서 프로필 정보이나 승인 상태를 저장한 후 각각의 액티비티의 onResume() 시 만들어 두었던 Main instance를 가져오도록 했다.



3.1.2 통신 데이터 패딩

가속도 값이 짧은 간격으로 넘어오는 상황에서 데이터가 자주 깨졌다. 모든 데이터를 활용하기 위해서는 깨짐을 방지해야 했다. 원인은 통신 소켓의 버퍼 길이의 문제였다. 데이터가 자주 쓰이게 되면 버퍼의 길이가 순식간에 차올라서 문자열이 끊기는 것이 문제였다. 버퍼 길이로 인해 문자열이 잘렸기 때문에 문자열의 앞뒤를 0으로 패딩하여 전송하는 방법을 선택했다. 패딩한 문자열은 서버에서 토큰으로 분리하여 전송된다.

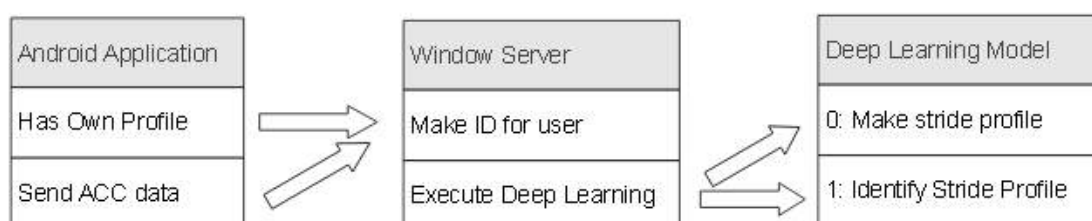
64 Byte정도가 가장 평균적인 길이였기 때문에, 문자열의 패딩 부분을 앞 뒤로 총 54 Byte를 주었다.
데이터 문자열은 총 10바이트로 64 Byte를 패딩하였다. 서버에서 이 패딩을 풀어 데이터를 받는다.



3.2 윈도우 서버

3.2.1 주피터 서버 파라미터 전달

윈도우 서버는 클라이언트로부터 프로필을 생성하며, 이 프로필을 파일화 한다. 그 후 딥러닝 서버에 실행 명령을 파일명과 [0, 1]로 구성된 파라미터를 전달한다.



3.3 딥러닝 서버

3.3.1 컨테이너 생성

Docker for windows라는 가상머신을 통해 딥러닝을 개발하였다. 오픈소스 딥러닝 모델들을 import 하여 볼 일이 많았기 때문에, 운영체제 의존성이 떨어지는 docker container를 사용하여 편리하게 작업했다. Jupyter Notebook은 ipython 작업환경을 지원한다. UI가 보기 쉽고, 간단한 인텔리를 지원했다. ipython은 다중 커널을 지원했기 때문에, notebook 안에서 여러 코드들을 동시에 돌릴 수 있었다.

3.3.2 RNN 학습 데이터와 학습 라벨 만들기

이 부분이 딥러닝 학습의 핵심이었다. 딥러닝으로 학습하기 위해선 데이터를 학습 가능한 SET과 그에 해당하는 Label SET으로 구성해야 한다. 안드로이드로부터 작성된 가속도 행렬로 다음과 같은 학습 데이터를 만들었다.

RAW DATA

A0	A1	A2	A3	A4	A5	A6	A7	...	A94	A95	A96	A97	A98	A99	A100
----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	------

DATA UNIT : 5

A0	A1	A2	A3	A4	A5	A6	A7	...	A94	A95	A96	A97	A98	A99	A100
----	----	----	----	----	----	----	----	-----	-----	-----	-----	-----	-----	-----	------



STEP_SIZE 크기만큼 분석할 보폭 단위를 설정한다. 이후에 이 단위들을 전치행렬로 만든 후 복사하여 Test SET을 만들었다. LabelSet은 TestSET에 대한 다음 보행 패턴을 예측하는 것이기 때문에, TestSET 이후 하나의 가속도 값을 추가했다. TestSET에 대한 다음 가속도를 예측하는 것을 의미한다.

		A0	A1	A2	A3	A0	A1	A2	A3	A4	
TestSet [0]	A0	A1	A2	A3	A4	A1	A2	A3	A4	A5	Label Set [0]
TestSet [1]	A1	A2	A3	A4	A5	A2	A3	A4	A5	...	Label Set [1]
TestSet [2]	A2	A3	A4	A5	...	A3	A4	A5	...	A96	Label Set [2]
.	A3	A4	A5	...	A96	A4	A5	...	A96	A97	.
.	A4	A5	...	A96	A97	A5	...	A96	A97	A98	.
.	A96	A97	A98	...	A96	A97	A98	A99	.
TestSet [95]	A95	A96	A97	A98	A99	A96	A97	A98	A99	A100	Label Set [95]
	A96	A97	A98	A99	A100	A97	A98	A99	A100		
	A97	A98	A99	A100		A98	A99	A100			
						A99	A100				

이와 같이 보폭 단위 길이 만큼의 열 수를 가진 TestSet과 LabelSet이 완성됐다.

TestSET [0]



Prediction!



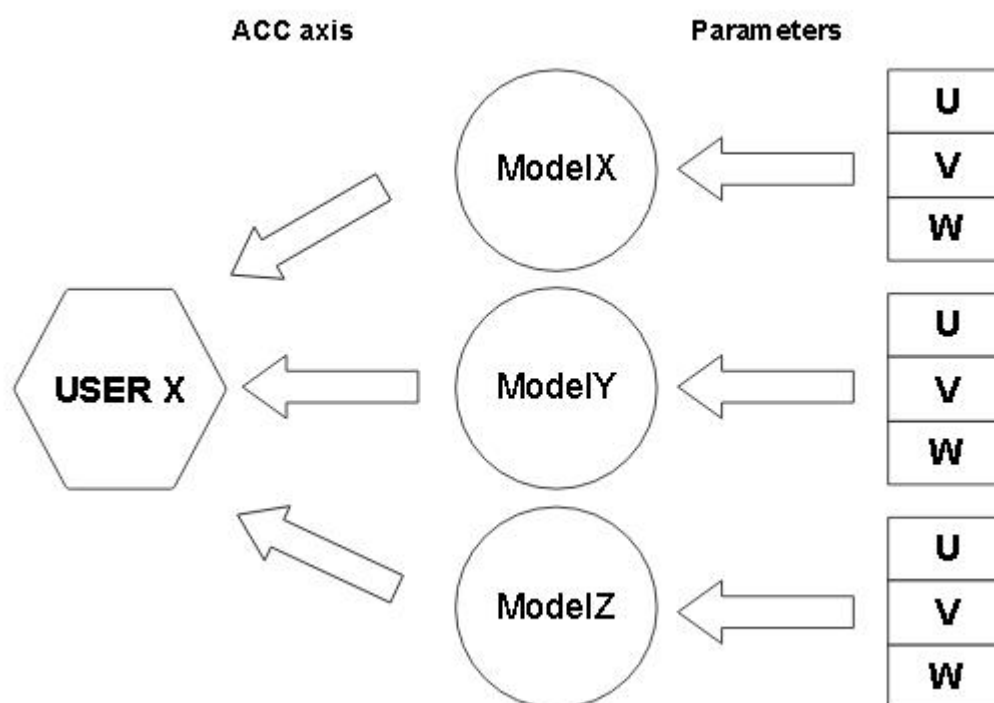
Label SET [0]



최종적으로, LabelSET [N]은 TestSET [N]의 다음 값을 예상하도록 만든 행렬이다. 각각의 index에 대응하는 Test와 Label 행렬들을 학습하여, 모든 Test 값에 대한 Label의 값의 차이를 최소화 하도록 RNN 학습을 하게 된다.

3.3.3 RNN 학습 모델 저장하기, 불러오기

RNN의 데이터 모델은 X, Y, Z 축 별로 만들었으며, 각각의 모델엔 세 개의 parameter를 가지도록 했다. epoch, hidden layer와 learning rate를 설정하면 propagation 하며 학습을 시작한다. 모델 도식은 다음과 같다.



3.3.4 오차 계산하기

마지막으로 학습한 모델과 실제 보행 데이터와 오차를 산출해 내는 작업을 하였다. 오차는 보행 데이터를 보행 단위로 나누어 비교했다. 오차 산출을 위한 예측값은 보행 데이터를 (보행 단위 - 1)의 길이만큼 가져온 후, 나머지 길이 1 만큼은 학습으로 예측된 값을 이어붙였다. 그 후 실제 값과 예측 값과의 차이로 오차를 산출했다. 오차를 합산하고, 가속도의 범위로 나누어 백분율로 나타냈다

Input SET [0]



Remove!



Generate Next!



Predicted SET [0]



$$\text{LOSS} = | A4 - [A4] |$$

4. 세부 역할

4.1 개별 임무 분담

혼자 프로젝트를 진행했다. 따라서 위 항목의 내용들을 아래 일정에 맞추어 개발했다.

4.2 개발 일정

SrNo	Months	3	4	5	6	7	8	9
1	Android							
	Use case							
	Aplication Architecture							
	User Interface							
	Data Modeling							
	Application Testing							
2	Backend Server							
	Use case							
	User Interface							
	Server Configuration							
	Message Posting							

초기에 전체적인 구조를 디자인하였고 필요한 구성요소와 공부할 분야에 대해 알아보았다. 그 후 시기별로 구성들을 완성하였다. 일정의 마지막으로 User Interface 및 각 구성요소들 간의 연동을 하였다. 프로젝트에 쓰인 새로운 지식들이 많았다. 적합한 딥러닝 알고리즘 공부, docker 및 jupyter notebook 등을 학습하는 과정에서 많은 시행착오와 시간이 필요했다. 사용할 수 있도록 공부하느라 생각했던 것 이상의 시간을 쏟았다.

일정의 마지막 단계에선 테스트를 진행하려고 하였다. 그러나 결과적으로 테스트가 부족하였다. 딥러닝은 예상보다 많은 데이터를 필요로 했고, hidden layer와 epoch 수를 조금만 늘려도 학습에 소요되는 시간이 크게 늘어났기 때문이었다.

5. 결론




처음 시도에 오차를 산출 과정에서 로그를 사용해 백분위를 줄여보았다. 로그를 썼더니 사용자 간에 차이가 거의 없어졌다. 대부분이 모델과 비교해서 98.5% 가량의 일치도를 보였기 때문이다. 사용자의 식별에 변별이 없었다.

그 후 오차의 제곱을 더하는 방식을 사용했다. 일단 오차를 제곱한 값들을 구한 후, 가속도값이 가질 수 있는 범위만큼 나눴다. 이 방식은 사용자를 좀 더 잘 식별했던 반면, 여전히 사용자 특성에 따라서 데이터를 충분히 특징있게 뽑아내지는 못했다.

마지막으로 쓴 방법은 오차의 제곱들의 합을 구한 후, 가속도값의 범위가 아닌 현재 데이터에 나타나는 값들 중 최소값, 최대값의 차이 만큼으로 나누어 보는 것이었다. 이것으로 일치도를 뽑아냈더니 훨씬 정밀한 값을 도출할 수 있었다.

특징이 뚜렷한 세사람으로 몇 차례 실험해 보았다. 다음 장의 표에는 걸을 때 팔 움직임이 거의 없었던 현령, 동작이 컸던 중수, 중간이었던 영식의 데이터를 첨부하였다.

USER \ MODEL	MODEL_hyunryung	MODEL_youngsik	MODEL_joongsoo
⊘ hyunryung	89.764290	91.876841	91.352387
✓ youngsik	83.452458	90.725579	87.459389
✓✓ joongsoo	10.620447	46.972973	89.160987

hyunryung :	
youngsik :	
joongsoo :	

중수: 중수의 경우 자신의 모델과의 오차가 가장 적었다. 팔 동작이 클수록 식별이 뚜렷하게 잘 되었다. 미미했던 동작들과의 차이가 커지며 오차-제공 방식으로 훨씬 눈에 띄게 구분된 것이다.

현령: 팔 동작이 작은 현령은 어느 곳에서나 꾸준한 높은 인식률을 보였다. 값들이 모델 데이터의 평균값 근처에 밀집한 현령은 어느 모델에서나 높은 인식률을 보였다. 구분하는데 실패하였다.

영식: 중간 정도의 영식도 인식에 성공했다. 중수와 마찬가지로 자신의 모델과의 오차가 가장 적었다. 중간 정도의 동작까지는 구분하는데 성공했다.

결과적으로, 팔 동작이 클수록 인식률이 높았다. 이것은 오차-제공 계산 방식이 성공적이었음을 보여준다. 실제 데이터와 예상 데이터와 차이가 클수록 제공된 값이 커지므로 개인의 특징이 잘 반영됐다. 동작이 큰 사람은 잘 구분하는 결과를 얻었다.

하지만, 미세한 데이터를 분석하기엔 안드로이드 센서 값이 충분히 정확하지 못했다고 생각했다. 상황에 따라 다르겠지만, 안드로이드 가속도 센서값을 쓰면서, 가속도가 넘어오는 주기가 비주기적인 것 같았다. 가속도값이 빠르게 변할때는 더 빈번하게 가속도를 보내왔지만 가속도 변화가 적을 때 가속도를 보내는 주기가 길어졌다고 느꼈다. 미세한 걸음을 완벽하게 분석해내기 위해서는 센서의 정확도가 일정하고, 더 정밀한 센서가 필요하다고 생각했다.

보행의 분석은 많은 데이터와 학습 시간을 필요로 한다. 일정의 마지막에 테스트를 했던 것이 아쉬웠다. 한명 한명 모델을 만들고, 데이터를 쌓는 일이 번거로웠기 때문에 오랜 시간적 여유를 두고 테스트 했다면 더 체계적인 특징을 잡을 수 있을 것이다. 딥러닝은 적합한 모델을 찾고 알고리즘을 완성하는 것도 중요하지만, 오랜 시간 학습하여 최적의 값을 찾아내는 것이 같은 비중으로 중요하다는 것을 다시금 느꼈다.

딥러닝에 관한 이론을 스스로 찾아서 공부하고, 실제로 결과를 도출해냈다는 과정에서 보람을 느꼈다. 컴퓨터공학의 분야에서 학부과정 중에 배운 지식을 기초로 딥러닝과 같은 학문의 새로운 지식을 정립했다는 점에서 졸업의 의미를 찾을 수 있었다.

안드로이드 시스템은 스마트워치, 웨어러블 디바이스에도 적용이 가능하다. 정밀한 모션 분석 시스템은 빅데이터를 가공하여 정교한 커스터마이징 할 것이다. Tensor flow, NLTK(Natural Language Toolkit) 등 현존하는 인공지능 오픈소스 라이브러리가 많다. 이를 활용한다면 딥러닝과 데이터 가공에 필요한 수학적 지식을 학습하여 수치적으로 정교한 프로그램을 만들 수 있을 것이다.

나의 목표는 '뚝뚝한' 보안 솔루션 개발이다. 기존에 없던 패턴의 공격이 들어올 시 자동 차단하며 그 패턴들을 다음 차단 시 적용할 수 있도록 학습하는 인텔리 한 침입탐지체계를 개발하고 싶다. 이는 악성코드 분석에도 사용할 수 있을 것 같다. 그 핵심은 기계학습, 그 중 딥 러닝이라고 생각한다. 이번 프로젝트를 통해 전문적인 머신러닝과 인공지능 기술들을 학습하고, 후에 개발의 발판으로 삼고 싶다.