



국민대학교
전자정보통신대학
컴퓨터공학부

캡스톤 디자인 I

종합설계 프로젝트

프로젝트 명	WiBi
팀 명	MOBICOM
문서 제목	중간보고서

Version	1.2
Date	2018-04-08


팀원	이 경재 (조장)
	양 재영
	박 수찬
	김 태기
지도교수	김 상철 교수

CONFIDENTIALITY/SECURITY WARNING


이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인 수강 학생 중 프로젝트 “WiBi”를 수행하는 팀 “MOBICOM”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “MOBICOM”의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역

Filename	중간보고서-WiBi.doc
원안작성자	양재영, 박수찬, 이경재, 김태기
수정작업자	양재영, 박수찬


 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2018-04-08	양재영	1.0	최초 작성	프로젝트 목표, 연구 내용 작성
2018-04-09	박수찬	1.1	내용 수정	수정된 연구내용 추가
2018-04-09	양재영	1.2	최초 작성	수행 내용 작성

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30


목 차

1	프로젝트 목표	4
2	수행 내용 및 중간결과	5
2.1	계획서 상의 연구내용	5
2.2	수행내용	5
3	수정된 연구내용 및 추진 방향	6
3.1	수정사항	6
4	향후 추진계획	7
4.1	향후 계획의 세부 내용	7
5	고충 및 건의사항	8

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30

1 프로젝트 목표

본 프로젝트에서는 현재에 사용되고 있는 카메라나 빛을 이용한 센서의 대안책을 제시하고자 한다. 그 이유는 현재 사용되고 있는 기술에서는 여러 가지 문제점이 발생할 수 있기 때문이다. 예를 들어, 카메라 설치 위치에 따라 생겨날 수 있는 사각지대가 있다. 이러한 문제점을 극복하고자 Wifi CSI 기술을 이용하여 사람의 움직임을 감지하고 그것을 그래프로 변환, 그래프를 바탕으로 텐서플로우를 활용한 머신러닝 기술을 이용하여 학습을 시키고 학습된 데이터를 이용하여 사람의 움직임이 들어왔을 때 어떤 움직임에 해당하는지 알아내는 것을 목표로 한다. 사람의 움직임을 와이파이를 통해서 파악하는 기술이 더 발전이 되면 홈 IoT에 까지 적용이 될 수도 있을 것이고, 그것을 위하여 따로 장비를 설치하지 않고 이미 설치되어 있는 와이파이를 통해서 기능을 제어할 수 있다면 비용절감이나 사생활 관리, 편의성 향상에도 많은 도움이 될 것이다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30

2 수행 내용 및 중간결과

2.1 계획서 상의 연구내용

2.1.1 행동 샘플 추출 단계

이 프로젝트에서 사람의 움직임을 파악하기 위하여 가장 먼저 해야할 것이 행동 샘플을 추출하는 단계이다. 그 이유는 먼저 텐서 플로우를 사용한 기계학습을 시키기 위해서는 정확한 행동에 해당하는 샘플을 바탕으로 학습을 시켜야 하기 때문이다. 이에 대한 이유는 정확한 행동이 있어야 그 다음에 사람의 행동이 들어오면 정확한 데이터를 가지고 분석해서 결과를 도출해 낼 수 있기 때문이다. 이 단계에서 추출되는 샘플 데이터 값은 그래프로 나타낼 수 있으며, 그 그래프는 timestamp, phase, amplitude 값을 가지고 있다.

2.1.2 샘플을 이용한 학습 단계


정확한 데이터를 가진 샘플을 추출한 이후, 텐서 플로우에 그 샘플을 학습 시킨다. 텐서 플로우에 샘플을 학습 시키면 ckpt 파일로 변환되어서 나오게 된다. 다음 단계에서는 이 ckpt파일을 이용을 하게 된다.

2.1.3 학습을 바탕으로 사람 움직임 검증 단계

검증을 하기 위해서 따로 제한된 공간에서 사람의 움직임을 측정하는 실험을 한다. 현재 진행되고 있는 측정 방법은 두 개의 NIC 사이에서 패킷이 전송되는 과정에서, 송신되는 전파의 사이에 있는 장애물에 따른 전파의 변화를 수치화해서 측정하는 방법이다. 이 사람의 움직임에 대한 변화를 적절한 방법을 통해 유의미한 데이터로 데이터로 변환하면 amplitude(크기), phase(위상), timestamp로 나뉘게 되며 이를 학습시킨 머신에 적용하여 제스처를 인식할 수 있다.

2.1.4 와이파이와의 연동 단계

지금까지는 두 개의 NIC만 상호통신을 하지만, 이후 애플리케이션 연동, 홈 IoT 등의 활용을 위해 다른 Wi-fi도 받아들여 사용할 수 있게 할 수 있게 연구중이다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30

2.2 수행내용

2.2.1 개요


```

12     xx = np.empty([0,window_size,90],float)
13     yy = np.empty([0,8],float)
14
15     ###Input data###
16     #data import from csv
17     input_csv_files = sorted(glob.glob(path1))
18     for f in input_csv_files:
19         print("input_file_name=",f)
20         data = [[ float(elm) for elm in v] for v in csv.reader(open(f, "r"))]
21         tmp1 = np.array(data)
22         x2 =np.empty([0,window_size,90],float)
23
24         #data import by slide window
25         k = 0
26         while k <= (len(tmp1) + 1 - 2 * window_size):
27             x = np.dstack(np.array(tmp1[k:k+window_size, 1:91]).T) #매번 시작점을
28             x2 = np.concatenate((x2, x),axis=0)
29             k += slide_size

```

먼저 이 부분에서 정확한 정보를 바탕으로 한 데이터를 전처리하는 과정이다.

전처리 하는 과정은 데이터의 정보를 보면 어떤 것을 제거하고 어떤 것을 추가하는 지를 알 수 있다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30

tmp1

	0	1	2	3	4	5
0	14.247	18.366	20.718	20.363	20.592	21.403
1	14.248	4.7672	3.8654	9.7157	9.7157	11.359
2	14.249	4.7672	3.8654	9.7157	9.7157	11.359
3	14.25	6.2604	6.2604	6.9049	9.9152	10.41
4	14.251	4.3495	7.6231	10.882	10.437	10.437
5	14.252	4.3495	7.6231	10.882	10.437	10.437
6	14.253	7.0083	9.0495	8.3437	10.189	12.323
7	14.254	3.8986	6.9089	9.7489	10.002	9.7489
8	14.255	3.8986	6.9089	9.7489	10.002	9.7489
9	14.256	6.8765	6.7061	10.856	11.052	11.36
10	14.257	4.6726	9.5281	6.9921	11.911	10.221
11	14.258	4.6726	9.5281	6.9921	11.911	10.221
12	14.259	4.1312	6.9712	9.5239	10.626	12.13
13	14.26	6.9626	9.3283	10.468	11.633	11.881
14	14.261	6.9626	9.3283	10.468	11.633	11.881
15	14.262	6.8353	9.2177	9.6753	9.8456	11.011
16	14.263	5.3195	7.624	9.2111	11.34	11.852
17	14.264	5.3195	7.624	9.2111	11.34	11.852
18	14.265	6.8501	3.8398	10.554	9.2325	8.8282

tmp1

Format: %.5f

☒ Colored cells
☒ Resize Automatically

Close

x

	0	1	2	3	4	5
0	18.366	20.718	20.363	20.592	21.403	21.377
1	4.7672	3.8654	9.7157	9.7157	11.359	10.855
2	4.7672	3.8654	9.7157	9.7157	11.359	10.855
3	6.2604	6.2604	6.9049	9.9152	10.41	10.606
4	4.3495	7.6231	10.882	10.437	10.437	11.802
5	4.3495	7.6231	10.882	10.437	10.437	11.802
6	7.0083	9.0495	8.3437	10.189	12.323	13.029
7	3.8986	6.9089	9.7489	10.002	9.7489	11.084
8	3.8986	6.9089	9.7489	10.002	9.7489	11.084
9	6.8765	6.7061	10.856	11.052	11.36	10.856
10	4.6726	9.5281	6.9921	11.911	10.221	13.013
11	4.6726	9.5281	6.9921	11.911	10.221	13.013
12	4.1312	6.9712	9.5239	10.626	12.13	11.121
13	6.9626	9.3283	10.468	11.633	11.881	11.833
14	6.9626	9.3283	10.468	11.633	11.881	11.833
15	6.8353	9.2177	9.6753	9.8456	11.011	11.259
16	5.3195	7.624	9.2111	11.34	11.852	10.438
17	5.3195	7.624	9.2111	11.34	11.852	10.438
18	6.8501	3.8398	10.554	9.2325	8.8282	4.7415

x[0]

Format: %.5f

☒ Colored cells
☒ Resize Automatically

Close

x2


	0	1	2	3	4	5
0	18.366	20.718	20.363	20.592	21.403	21.377
1	4.7672	3.8654	9.7157	9.7157	11.359	10.855
2	4.7672	3.8654	9.7157	9.7157	11.359	10.855
3	6.2604	6.2604	6.9049	9.9152	10.41	10.606
4	4.3495	7.6231	10.882	10.437	10.437	11.802
5	4.3495	7.6231	10.882	10.437	10.437	11.802
6	7.0083	9.0495	8.3437	10.189	12.323	13.029
7	3.8986	6.9089	9.7489	10.002	9.7489	11.084
8	3.8986	6.9089	9.7489	10.002	9.7489	11.084
9	6.8765	6.7061	10.856	11.052	11.36	10.856
10	4.6726	9.5281	6.9921	11.911	10.221	13.013
11	4.6726	9.5281	6.9921	11.911	10.221	13.013
12	4.1312	6.9712	9.5239	10.626	12.13	11.121
13	6.9626	9.3283	10.468	11.633	11.881	11.833
14	6.9626	9.3283	10.468	11.633	11.881	11.833
15	6.8353	9.2177	9.6753	9.8456	11.011	11.259
16	5.3195	7.624	9.2111	11.34	11.852	10.438
17	5.3195	7.624	9.2111	11.34	11.852	10.438
18	6.8501	3.8398	10.554	9.2325	8.8282	4.7415

x2[0]

Format: %.5f

☒ Colored cells
☒ Resize Automatically

Close

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30

위 그래프는 tmp1, x, x2에 대한 시각적인 자료를 보여준다. 처음 첫 열은 timestamp에 해당한다.

먼저 어떠한 데이터를 추가하고 제거하는지 설명을 하면, 이 데이터는 timestamp, phase, amplitude의 세가지로 구성되어있다. 그 중에 timestamp 부분은 잘려 나가게 된다. 학습을 시키는 과정에서 필요하지 않은 정보이기 때문이다. timestamp를 자르는 동시에 amplitude에 해당하는 데이터를 x2라는 배열을 이용하여 차례대로 저장을 시킨다.

현재 이 부분에서 실행하는 과정은 한 가지의 행동에서 여러개의 샘플에 대한 각각의 amplitude만을

```

31             xx = np.concatenate((xx,x2),axis=0)
32         xx = xx.reshape(len(xx),-1)

```

모으는 과정이다.

다음에 실행하는 부분은 이 부분인데, 이 부분에서는 지금까지 각각의 샘플에 대한 amplitude를 하나의 배열로 만들어 주는 부분이다. 즉 하나의 행동에 대하여 관하여 모아놓은 amplitude들을 하나의 파일로 만들기 위하여 하는 과정이라고 볼 수 있다.


```

for i, label in enumerate(["bed", "fall", "pickup", "run", "sitdown", "standup", "walk"]):
    filepath1 = "./Dataset/input_" + str(label) + "*.csv"
    filepath2 = "./Dataset/annotation_" + str(label) + "*.csv"
    outputfilename1 = "./input_files/xx_" + str(window_size) + "_" + str(threshold) + "_" + label + ".csv"
    outputfilename2 = "./input_files/yy_" + str(window_size) + "_" + str(threshold) + "_" + label + ".csv"

    x, y = dataimport(filepath1, filepath2)
    with open(outputfilename1, "w") as f:
        writer = csv.writer(f, lineterminator="\n")
        writer.writerows(x)
    with open(outputfilename2, "w") as f:
        writer = csv.writer(f, lineterminator="\n")
        writer.writerows(y)
    print(label + "finish!")

```

마지막 메인 부분에서 각각의 파일에 대하여 이름을 만들어 주는 과정을 하고 있다. 현재 사용하는 정확한 샘플은 침대에 누워있을 때, 사람이 넘어질 때, 무언가를 잡는 행동을 할 때, 달리는 행동, 앉는 행동, 일어서는 행동과 마지막으로 걷는 행동을 사용하고 있다.

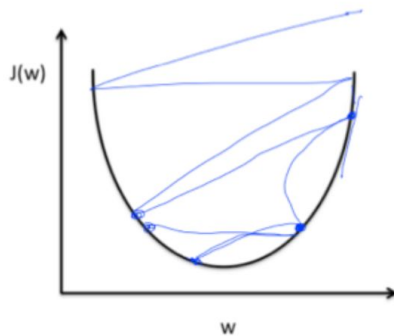
 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30

다음은 샘플을 이용하여 텐서 플로우에 학습을 시키는 과정이다.

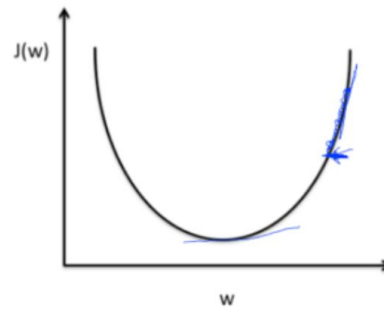
```
# Parameters
```

```
learning_rate = 0.0001
```


Large learning rate: overshooting



Small learning rate:
takes too long, stops at local minimum



이 그래프는 gradient decent로 불리는 학습을 시키는 과정에서 사용하는 알고리즘을 시각적으로 보여준다. 이 안에서 learning rate 가 너무 커지게 되면 이 알고리즘의 궁극적인 목표인 최저점을 찾는 과정을 하지 못할 수도 있고 그래프의 밖으로 빠져나가는 현상도 벌어질 수 있다. 하지만 반대로 learning rate가 너무 작게 되면 이 그래프의 최저점을 찾아가는 과정에서 시간이 너무 많이 걸릴 수가 있다. 그 현상을 오른쪽 그래프에서 보여진다. 파란점의 움직임이 너무 조금씩 움직이는 바람에 언제 이 알고리즘이 언제 끝날지 모르는 것이다. 그래서 적절한 learning rate를 찾아서 입력해 주는것이 중요하다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30


```
# Keep training until reach max iterations
while step < training_iters:
    batch_x, batch_y = wifi_train.next_batch(batch_size) #wifi_train에 저장된 x와 y를 배
    x_vali = wifi_validation.images[:]
    y_vali = wifi_validation.labels[:]
    # Reshape data to get 28 seq of 28 elements
    batch_x = batch_x.reshape((batch_size, n_steps, n_input)) #batch_x를 500행 90열 ba
    x_vali = x_vali.reshape((-1, n_steps, n_input)) #x_vail은 500행, 90열로 만들고 남은걸
    # Run optimization op (backprop)
    sess.run(optimizer, feed_dict={x: batch_x, y: batch_y})


    # Calculate batch accuracy
    acc = sess.run(accuracy, feed_dict={x: batch_x, y: batch_y})
    #x에 batch_x, y에 batch_y를 입력하여 accuracy 실행하여 결과값을 acc에 저장
    acc_vali = sess.run(accuracy, feed_dict={x: x_vali, y: y_vali})
    # Calculate batch loss
    loss = sess.run(cost, feed_dict={x: batch_x, y: batch_y})
    loss_vali = sess.run(cost, feed_dict={x: x_vali, y: y_vali})

    # Store the accuracy and loss
    train_acc.append(acc)
    train_loss.append(loss)
    validation_acc.append(acc_vali)
    validation_loss.append(loss_vali)

    if step % display_step == 0:
        print("Iter " + str(step) + ", Minibatch Training Loss= " + \
              "{:.6f}".format(loss) + ", Training Accuracy= " + \
              "{:.5f}".format(acc) + ", Minibatch Validation Loss= " + \
              "{:.6f}".format(loss_vali) + ", Validation Accuracy= " + \
              "{:.5f}".format(acc_vali) )
        step += 1
```

이 과정은 텐서 플로우에 학습을 시키는 과정이다.


 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30

3 수정된 연구내용 및 추진 방향

3.1 수정사항

이전에는 Channel State Information을 활용한 Home IoT의 여러 가지 기능을 구현함을 개발의 마지막 단계로 계획하였으나, 본 프로젝트의 필수 과제들(기계 학습 모델 제작, 그에 따른 제스처의 구현 등..)과 IoT를 모두 기간 내에 수행하기 어렵다는 판단 하에 Home IoT 구현 수행 과제를 CSI 제스처 상태에 따른 정보와 인식 그래프를 보여주는 안드로이드/IOS 애플리케이션으로 대체하기로 결정하였다.


 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30

4 향후 추진계획

4.1 향후 계획의 세부 내용

4.1.1 인터넷을 통한 패킷 전송

현재는 다른 Wi-fi 네트워크가 연결되지 않은 두 NIC간의 상호 전송으로만 통신이 진행되고 있다. 이후에 애플리케이션과의 연동을 고려하여 향후에는 다른 Wi-fi와의 연결이 동시에 가능하도록 하여 애플리케이션에서 정보를 받을 수 있게 구현할 계획이다.

 국민대학교 컴퓨터공학부 캡스톤 디자인 I	중간보고서		
	프로젝트 명		
	팀 명		
	Confidential Restricted	Version 1.2	20xx-APR-30

5 고충 및 건의사항