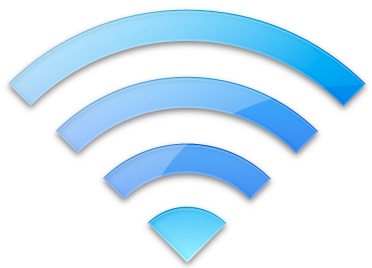
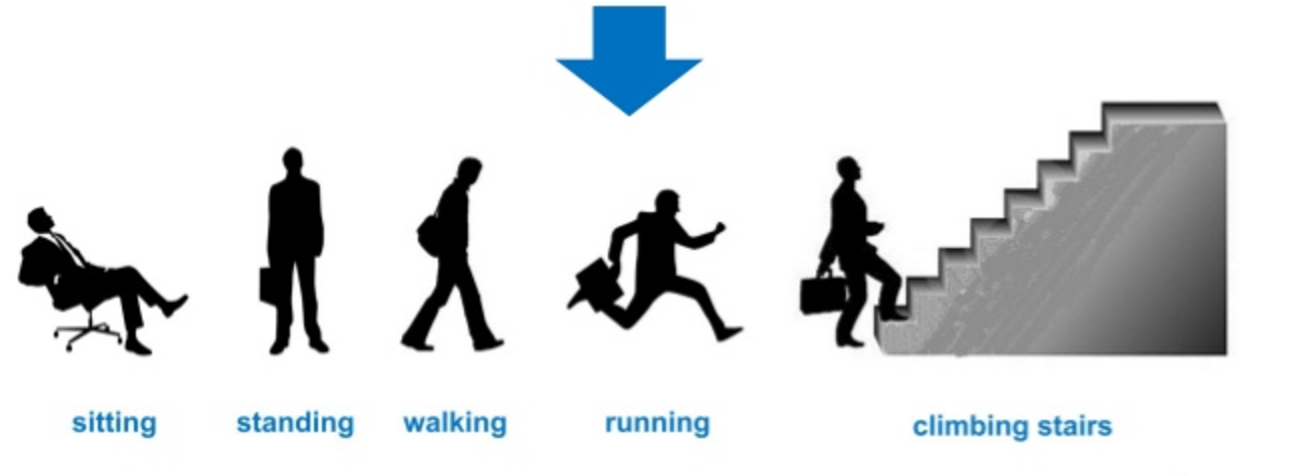
Activity Recognition using B-CSI data

Sanket Dige (110815356) Raunaq Kochar (110944843)



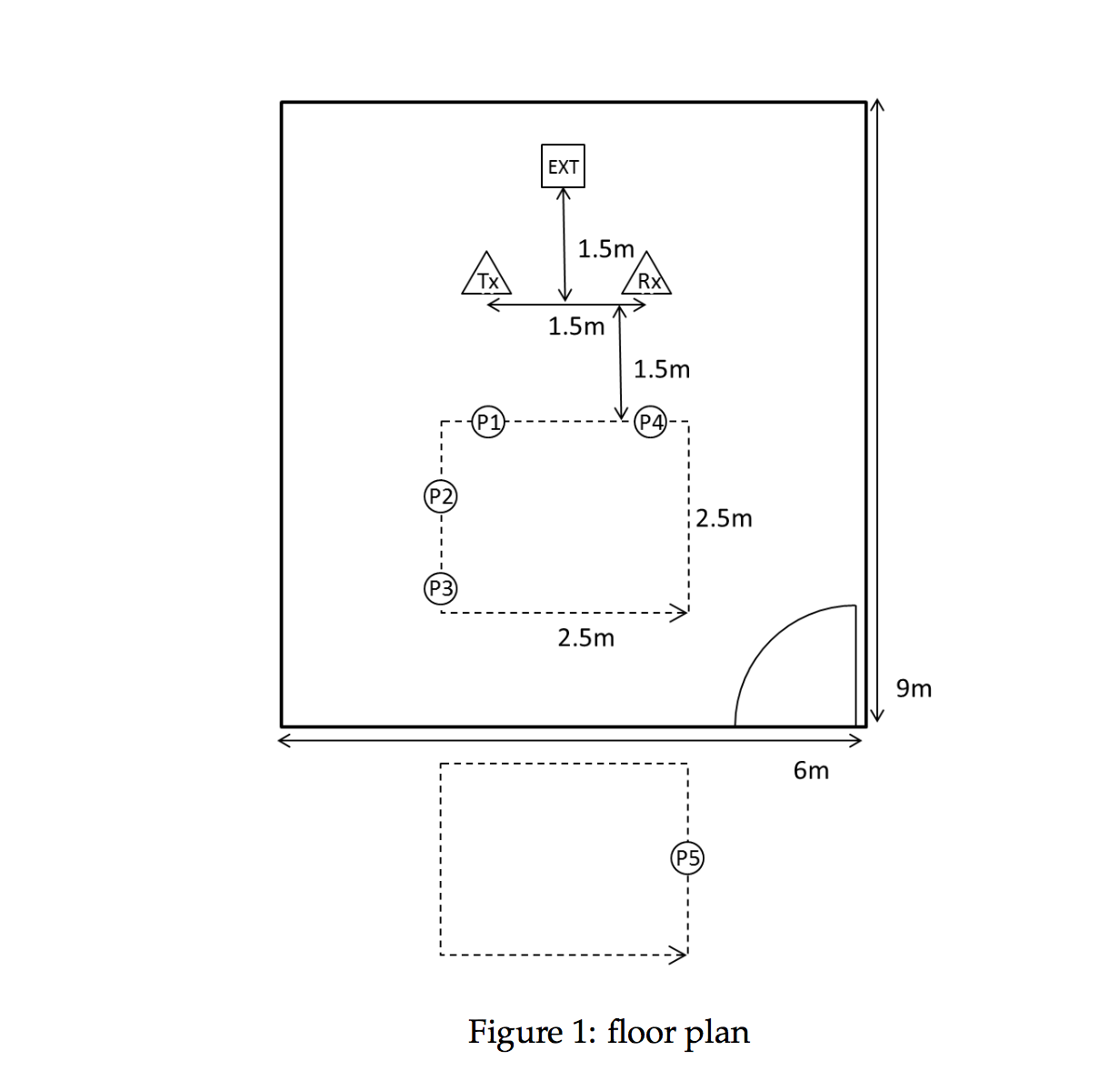


## **Introduction**

This project deals with recognition of different activities performed by humans using Wifi (backscatter-CSI) data in lieu of the traditional methods of wearables, RSSIs or radars. The power of CSI data over plain RSSI values is that it provides us with both RSSI as well as Phase information for all subcarriers in the WiFi channel. “A human body can greatly attenuate WiFi signal and that can act as a "fingerprint" to identify the presence or absence of a person”, is the principle we will use for implementation of this project. Now instead of just using the CSI data onto which lot of research has been done and there has been a dead-end to the effect the technology can be leveraged for activity detection, here we are using back-scatter CSI data also known to be b-CSI data.

* View and edit this document in Word on your computer, tablet, or phone.
* You can edit text; easily insert content such as pictures, shapes, and tables; and seamlessly save the document to the cloud from Word on your Windows, Mac, Android, or iOS device.

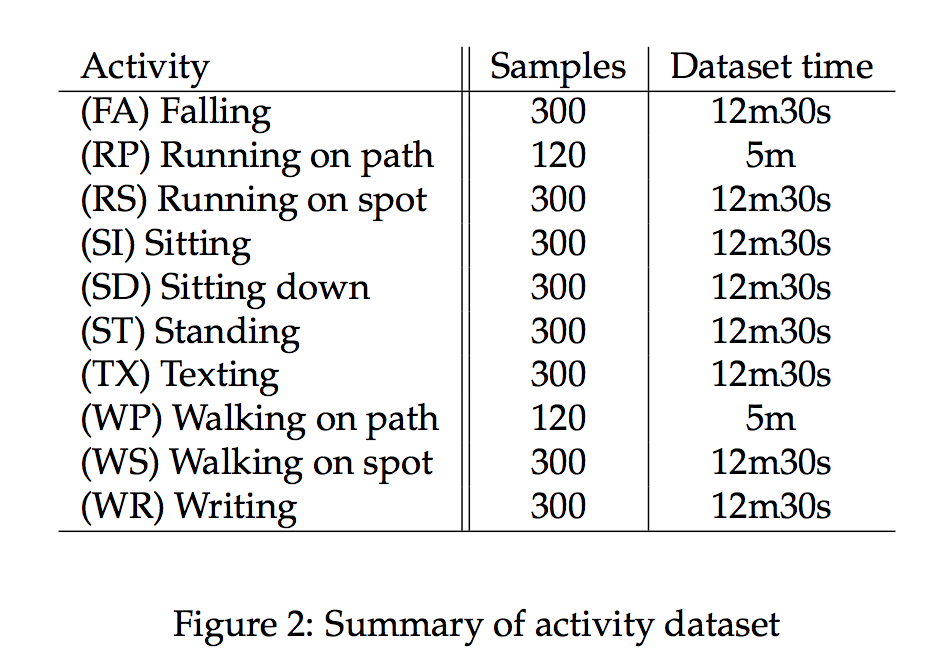
## **Setup**



We collected training samples for 10 different activities in our lab, which is 9m of length and 6m in width, as show in figure 1. We collected total 2640 samples for the testing activities from 6 research participants. The participants included 5 males and 1 female graduate students with ages in the range of 25-34.

We evaluated the recognition accuracy of *B-CSI* through two sets of experiments, one is in the trained environments and the other is in the untrained environments. We use the lab where we collected the training environments.

The activities for which we collected training samples are listed in figure 2, along with abbreviations and number of samples for each activity. We collected samples for each activity. For walking on path and running on path, our participants followed the path with a dashed line at the center of the lab. Other activities were conducted at the five different locations which marked as circle. The location five is exceptionally conducted outside the lab. Tx and Rx represent the location of the tag node and sink node. EXT represent the location of exciter. The total time for our activity dataset with 2640 samples was data collection 1 hour 50 min on a desktop with Intel i5-3470 CPU, as shown in figure 2.



## **Experiment / Implementation**

The Support Vector Machine (SVM) [] with a linear kernel function for mapping input samples into high dimensional space is used. *B-CSI* achieves an across validation accuracy of 98.1% across all activities.

*B-CSI* achieves an accuracy of higher than 98.2% for the research participant who is not included on training set.

## **Run the project**

To predict live activity on incoming data, we have set up an infrastructure with the help of google drive and a command line utility named ‘gdrive’. We have created a client-server architecture where a client will push a sample\_data that needs to be predicted for an activity by server.

On the google drive, we have a specific file ‘sample\_data.csv’ which is specified by a given file\_id once we create a sharable link from it.

Using this file\_id, we override this sample\_data.csv file with a new incoming live data sample using update\_gdrive.py script. At client side, he should have the client folder which includes the gdrive\_data folder and update\_gdrive.py script. Inside the gdrive\_data folder, we have a sample\_data.csv file which needs to have a sample dataset for prediction. One we have that; client PC can push this file using update\_gdrive.py script.

On server side, we have a live prediction system (predict\_activity.py) which is constantly checking if any new data is uploaded on google-drive at that specific file\_id (which is sample\_data.csv pushed to google-drive using the above scenario). If an updated version of file is seen, the prediction system, we download the sample\_data.csv file, predict the activity from it and got to sleep for 10 seconds to check for any new data. If no updated file is seen, the prediction system will simply wait for 10 secs to check back again for any new data.

Now prior to this prediction, we should train our prediction system with some training set and stored the trained model. This is done by the ‘SVM.py’ script which takes in the sample data set from data folder and trains using SVM classifier. This trained model is then exported to ‘activity\_recognizer\_model.pkl’ file. This trained model file is then used by the prediction system to predict for any incoming sample test data-set.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DC1 | DC2 | DC3 | DC4 | DC5 | DC6 | DC7 | DC8 | DC9 | DC10 | DC11 |
| CH1  1 | CH1  2 | CH1  3 | CH1  4 | CH1  5 | CH1  6 | CH1  7 | CH1  8 | CH1  9 | CH1  10 | CH1  11 |
| CH2  1 | CH2  2 | CH2  3 | CH2  4 | CH2  5 | CH2  6 | CH2  7 | CH2  8 | CH2  9 | CH2  10 | CH2  11 |
| CH3  1 | CH3  2 | CH3  3 | CH3  4 | CH3  5 | CH3  6 | CH3  7 | CH3  8 | CH3  9 | CH3  10 | CH3  11 |