Water Tracking System:
Using Machine Learning Method to Recognize Water Drinking Movement Patterns

Hanye Wei
Computer Science
Illinois Institute of Technology
IL, USA
hwei15@hawk.iit.edu

Xin Su
Computer Science
Illinois Institute of Technology
IL, USA
xsu11@hawk.iit.edu

***Abstract*—the quantity of drinking water is an important factor in maintaining a healthy life style. Till today, there is no practical personalized, wristband application for tracking drinking water quantity. Current water drinking application are mostly on the drinking bottle, or need users to input the quantity each time or one time a day. Our research show that with proper times of training, there is a way to automatically track the quantity of the water. One basic way to implement is to use the sensor of smart bracelet to tracking the wrist moving. But due to the openness of the bracelet API, we use Android phone's sensor to retrieve the raw data and using Weka algorithms to recognize specified water drinking movement of individual's behavior. We use sensors in the smart phone to train with a user's water drinking movement. Then we use the training data set to build a personal model to recognize and track the user's water drinking movement and reach a high accuracy. Our result show that our system achieves consistently about 90% accuracy for event detection. By providing automatically detection of water system we can give user a good profile to see the daily water drinking quantity without input manually.**

Keywords—Machine Learning, Decision Tree, Android Phone, pattern recognition

I. INTRODUCTION

Drinking water is an important part of daily life. Having 8 cups of water daily is one of the healthy life style. People are getting lazy but trying to get health life these days. So there is growing need to automatic detection application which monitor user's health. Dehydrate of a person can lead some seriously issues[1].

There are similar work of tracking drinking water quantity application, such as Drinking Water[2], Aqualert[3] and etc. However they all just remind you drinking water any certain time and you need to input the quantity to do this work or just tell it the daily quantity water you need to drink based on your weight and activity which are all input manually.

This paper presents two stage of our research. First stage, we use Weka[4] to analysis the accuracy of classifier machine learning algorithm accuracy. Then, we develop an Android[5] application to demo our research result and have the full function to detect the right behavior after training properly.

II. RELATED WORK

There is one implementation of automatically detecting smoking gesture [6]. It uses an additional device combined with the smart phone. The smoking gesture is quite similar to drinking movement, however there is a difference: smoking is a sequential movement, but drinking can be either a single or sequential ones. They have the same property that the hand starts from still, raise to close to mouth and then back to still position. Also, different users may have different habits. Therefore, we treat it as a personal gesture to detect and recognize as well.

III. SYSTEM REQUIREMENT

Our system is designed to automatically detect the movement of drinking water in real-time on smart phone. Thus, the following requirement need to be addressed:

(1) The algorithm should be applied to the smart phone and the algorithm should be lightweight;
(2) The algorithm should be suitable for the personalized gesture detection;
(3) The obtaining data type and feature should has its own pattern for individual.

To meet the requirement, there are four major challenges of the application of our system. The first is to choose the accelerometer of Android phone. There are thirteen accelerometers [7] in Android phone for developers. We choose gyroscope and linear acceleration for us to test.

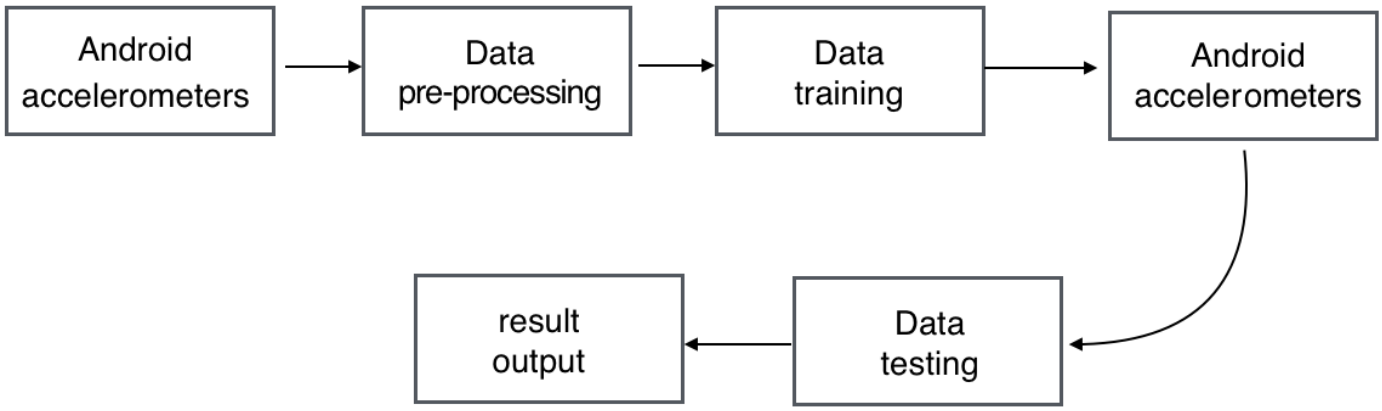

FIGURE 1. SYSTEM OVERVIEW

The second is to choose the data range for our application. There are four sensors change speed to choose, such as fastest, game, UI and normal mode[8]. Choosing one best for application refresh and algorithm need to be tested.

The third is how to do the feature extraction. We do not know how to do with raw data. There are many way to extract from the raw data, such as mean, standard deviation, and change rate.

The last one is the better algorithm which can be applied to the Android phone and have proper time and energy consume. There are also machine learning algorithms to do these kind of unsupervised algorithm to detect the specific feature.

IV. SYSTEM OVERVIEW

First, the system obtains the raw training data from android accelerometers. Then pre-process the data to find if it qualify to be training data or not. After that the system use the raw data of training data to one row of basic input to the algorithm to train. After training, the application obtain one row of testing data from android accelerometers. Then manipulate the raw test data with the same way of training data. After this, algorithm such as J48 test the data and output the result(Figure 2).

V. SYSTEM DESIGN

A. Model

There are mainly three models to choose:

- All-user model: this model will reduce the time of training period, we can collect data set ahead and have it trained provide the result as the base when a user starts to use the device, but it usually has the worst accuracy, because even based on general or average training cases, the water drinking movement is a personal specific character a user has, therefore there are many trivial difference between users.
- Personal model: As we describe that the gesture is treated as a personal one, this personal model has the best accuracy. The drawback is it will need a long period of time to get enough training data so as to reach a high accuracy.
- Hybrid model: hybrid model combines the all-user model and the personal model. It bootstraps from a general all-user model and grows with a personal model by using personal data. This model will reach an acceptable accuracy more quick than a personal model. But it would take longer than the personal model if it wants to reach a high accuracy.

For our to-be tracked movement is mainly personalized one, we choose to use a personal mode[9].

B. Feature

We collect 190 sample points for one drinking movement for our test using the linear accelerator in the smart phone. Each point has three values: x, y, z which stands for the acceleration force along the three axes (excluding gravity). We use the following formula to reduce the 3 dimensions to one dimension: $x^2 + y^2 + z^2$. After this operation, pass the data through a low pass to exclude the noise.

Then we compute the change value: for the i^{th} point's value samplePoint[i] and the $(i+1)^{th}$ point's value samplePoint[i+1] , the change value changeValue[i] is samplePoint[i+1] - samplePoint[i] . There are totally 189 change values. We also compute the mean value standard deviation of the 190 sample points and add these two values into the list of the features. In the end we have 191 features for one drinking movement[10].

C. *Algorithm*

We pick three algorithms that may fit to our requirement of recognizing the drinking movement: J48, RBFNetwork and MultiplayerPerceptron.

J48: J48 is a classifier in Weka. It is an implementation of C4.5 algorithm and use decision tree. A decision tree is a decision support system that implements tree-like graph decisions and their possible after-effect, including chance event results, resource costs, and utility. It is also used to learn a classification function which concludes the value of a dependent attribute (variable) given the values of the independent (input) attributes (variables). This verifies a problem known as supervised classification because the dependent attribute and the counting of classes (values) are given [8].

RBFNetwork: RBFNetwork is a class that implements a normalized Gaussian radial basis function network. It uses the k-means clustering algorithm to provide the basis functions and learns either a logistic regression (discrete class problems) or linear regression (numeric class problems) on top of that. Symmetric multivariate Gaussians are fit to the data from each cluster. If the class is nominal it uses the given number of clusters per class. It standardizes all numeric attributes to zero mean and unit variance [3].

MultiplayerPerceptron: MultiplayerPerceptron is a classifier that uses back propagation to classify instances. This network can be built by hand, created by an algorithm or both. The network can also be monitored and modified during training time. The nodes in this network are all sigmoid (except for when the class is numeric in which case the output nodes become unthresholded linear units) [3].

As our experiment goes on, the result shows that the accuracy of the MultiplayerPerceptron is not quite stable. For one user it reach as high as 91%, however for another user, it falls to as low as 50%. Additionally. This algorithm has a high time cost issue. Therefore we would use the previous two algorithms and drop the last one.

VI. Implementation

The process of our experiment is:

- Acquire number of sample points
- Training
- Test

First we test to see how many sample points we should get for each movement (Figure 2). We collect over 10 cases to test how many sample points it will record for the entire movement, Figure 3 shows some of them. We conclude the following statistics:

- Short: 150 points (150~170)
- Mid: 170 points (170~190)
- Long: 190 points (>190)

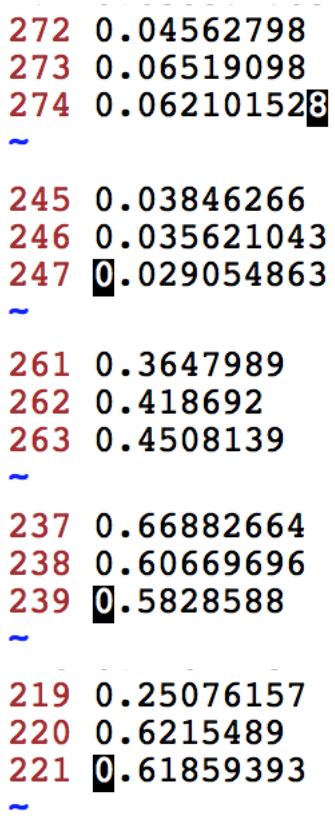

Figure 2 Acquire number of sample points

A. *First Attempt*

The first attempt of our experiment is to have 50 data as training set that are all positive movement, we say it is TRUE. Then each time we have a test case, no matter if it is predicted result is correct, we add it into the training set to let it grow in this way. Finally we have about 20 tests cases and all are included into the training set one by one. However, the accuracy of the predicted results is very bad, only about 40%. The reason is that the test results do not have a regulation or correction. When we have a false positive of false negative result we still include it into the training set, which makes the set less accurate.

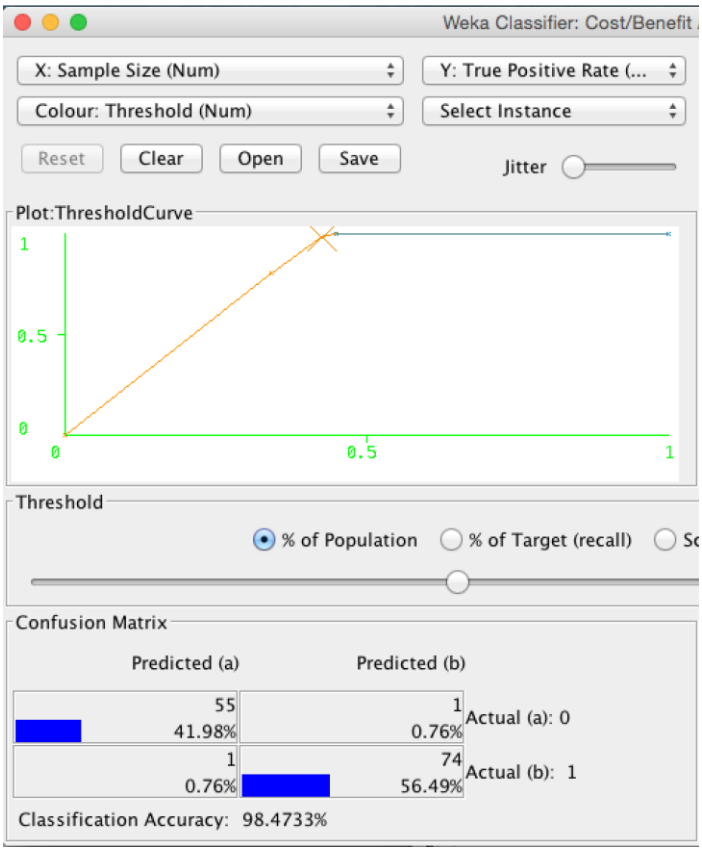

FIGURE 3 J48 PRE-PROCESSING

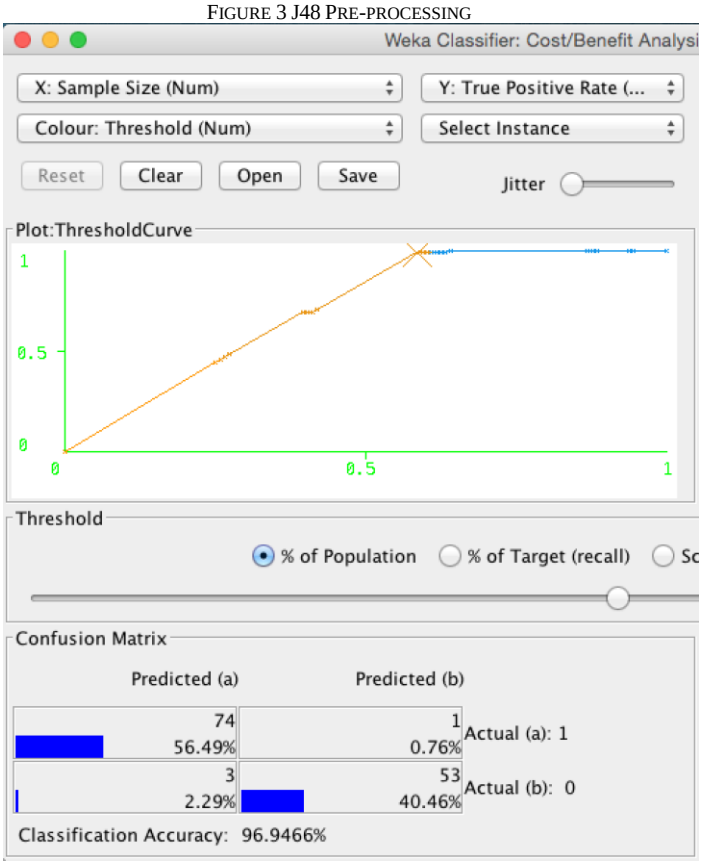

FIGURE 4 RBFNETWORK PRE-PROCESSING

B. *Second Attempt*

The second attempt of our experiment is to use a different accelerator, the gyroscope accelerator. However, since the movement does not require too much rotate on the wrist, the data we get has little changes, then we would keep focusing on using the linear accelerator.

C. *Modification*

For the first attempt, we simply do not add the test cases into the training data set, instead, we could let user to correct the test case first and then let user choose to add it into the training set. This would be a future work of our experiment. Now we will use the fixed 50 data as the training set. Also, we decide to add negative movement data, which is FALSE data, into the training set instead of just all TRUE data. We will use 25 TRUE data and 25 FALSE data to build the training set.

For the second attempt, for now we do not consider using the second accelerator.

D. *Data Pre-processing*

After collecting the data, we use the software with GUI window provided by Weka to have a pre-processing on the data. The Figure 3 shows the pre-processing of J48, and the Figure 4 shows the pre-processing of RBFNetwork:

The graph describes that the classification accuracy of both algorithms is high enough that the data collected is able to be used to have the training. Also from the axis we can tell that when the training uses over 60\% of the amount of the data, its classification accuracy is already close to 100\%. That means the amount of the training data is enough.

E. *Training and Test*

By using the Android app we can collect the data and start training as well as test. We collect the training data set with 130 samples (75 TRUE, 55 FALSE) Test data set: 44 cases (27 TRUE, 17 FALSE). Here is the comprehensive test results of J48 and RBFNetwork for one user (Figure 5 and Figure 6).

The accuracy of J48 comes: accuracy is 84%, false positive is 0, and false negative is 16%. The accuracy of RBFNetwork comes: accuracy is 91%, false positive is 0, and false negative is 9%. Both of the results meet our expectation. We give out some more results of an additional user. The raining data set is 115 samples (75 TRUE, 40 FALSE), test data set is 50 cases (25 TRUE, 25 FALSE). The accuracy of J48 comes: accuracy is 90%, false positive is 8%, and false negative

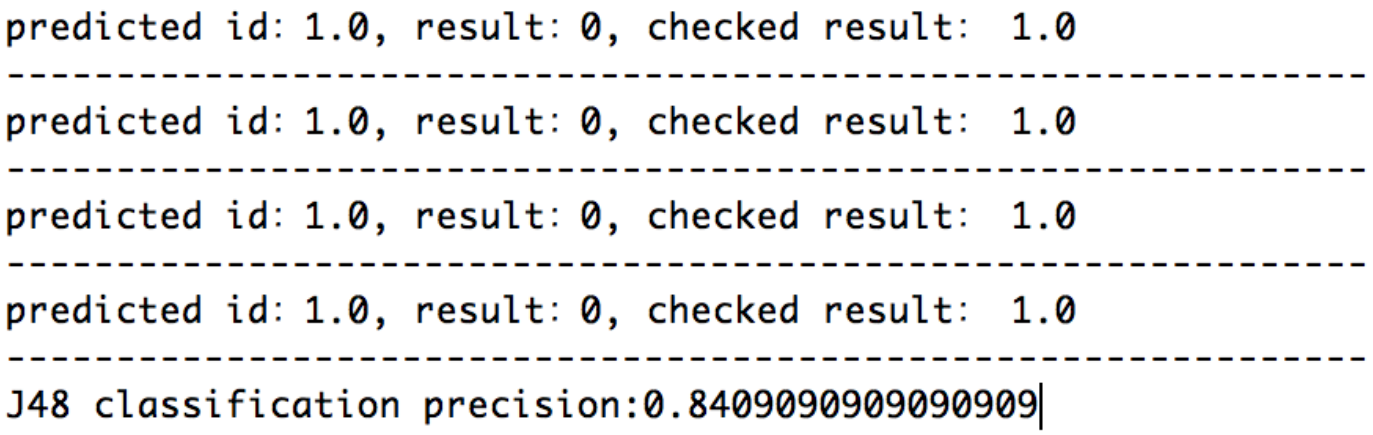

FIGURE 5 J48 RESULT

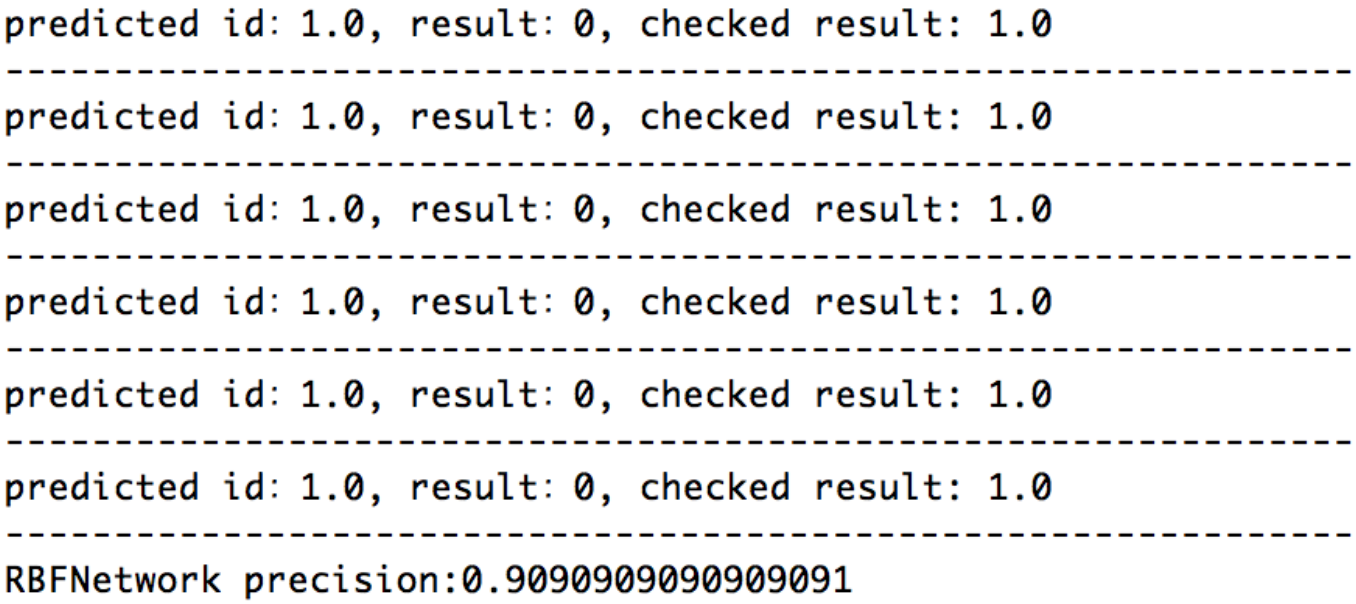

FIGURE 6 RBFNETWORK RESULT

is 2%. The accuracy of RBFNetwork comes: accuracy is 80%, false positive is 20%, and false negative is 0.

VII. EVALUATION

A. *Decision Tree Compare*

The following Figure 7 is the Decision Tree compare of the first user and the second. From the graph, we can tell that the first user has to use more features to make classification than the second user. They have different characteristic gesture of the drinking movement. For the second user, the reason for such a single decision tree to be generated is that, this user has quite the same track when he/she is do that movement.

B. *Result comparare*

The accuracy of J48 comes: accuracy is 84%, false positive is 0, and false negative is 16%. The accuracy of RBFNetwork comes: accuracy is 91%, false positive is 0, and false negative is 9%. Both of the results meet our expectation. We give out some more results of an additional user. The raining data set is 115 samples (75 TRUE, 40 FALSE), test data set is 50 cases (25 TRUE, 25 FALSE). The accuracy of J48 comes: accuracy is 90%, false positive is 8%, and false negative is 2%. The accuracy of RBFNetwork comes: accuracy is 80%, false positive is 20%, and false negative is 0.

So the best method in theory is the RBFNetwork, not only because the accuracy or time and energy consuming, but also because the stability of the algorithm result.

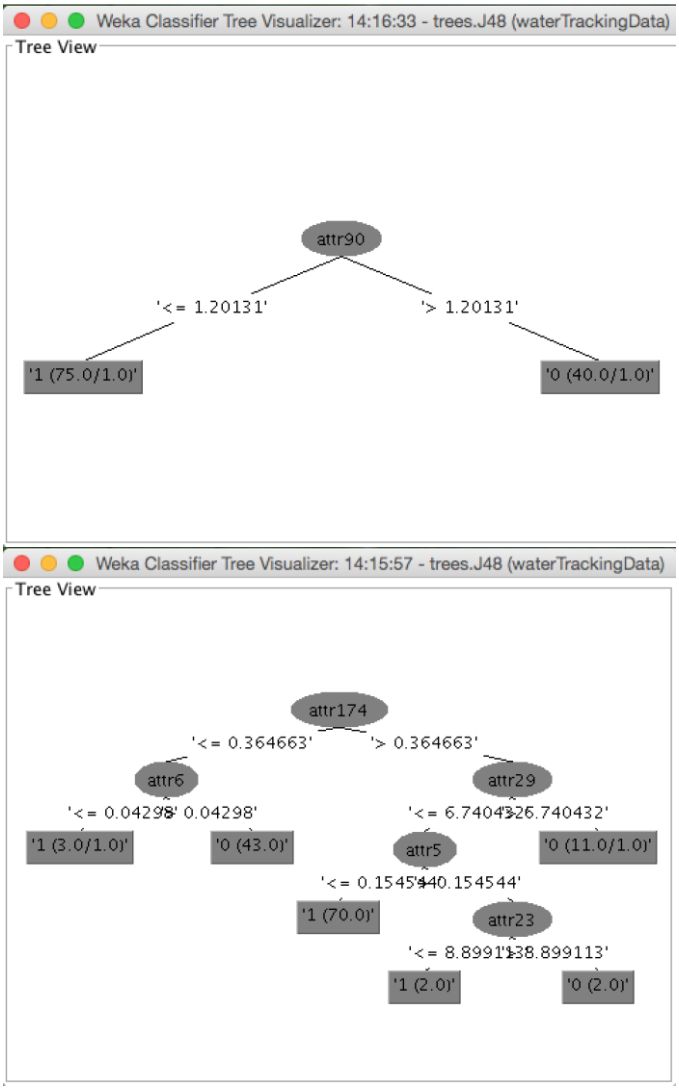

FIGURE 7 DECISION TREE COMPARISON

VIII. ANDROID APPLICATION

With this Android application, we can collect training data into a training file stored in the smart phone and use it for further prediction (Figure 8). After the training phase, the app then starts to make the prediction and show the result to the user (Figure 9). This simple yet practical Android app is built with the J48 algorithm inside. It can simply run the training without any high time cost.

A. *Overview of the Android activity*

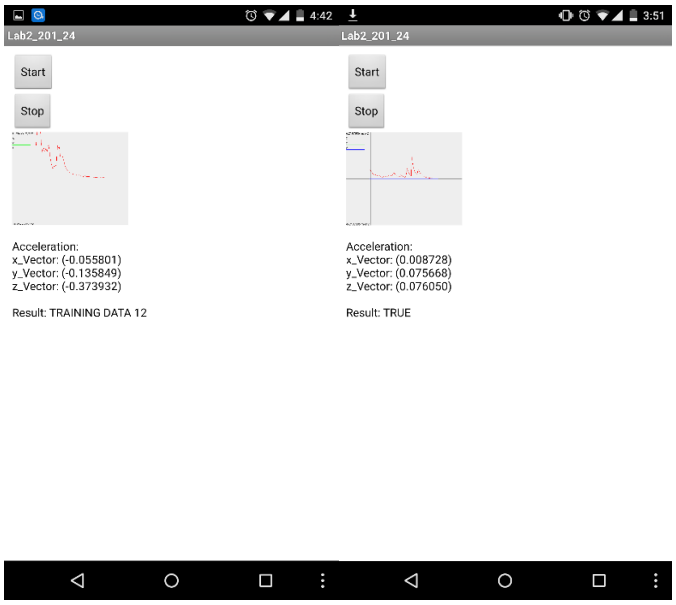

FIGURE 8 ANDROID APPLICATION TRAINING

FIGURE 9 ANDROID APPLICATION TEST

During the Android developing stage, we found very interesting story of Android activity (Figure 10). [11]

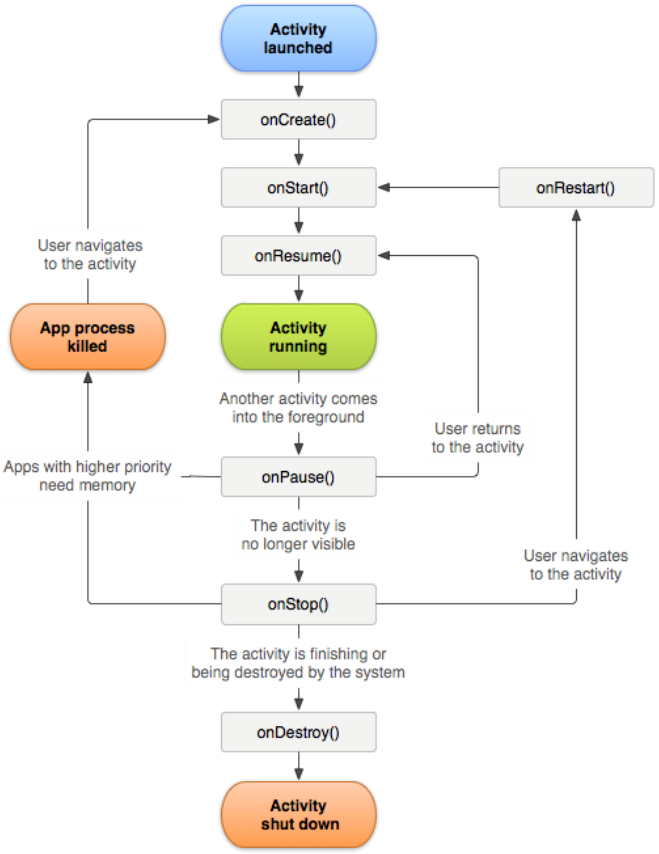

FIGURE 10 ANDROID ACTIVITY

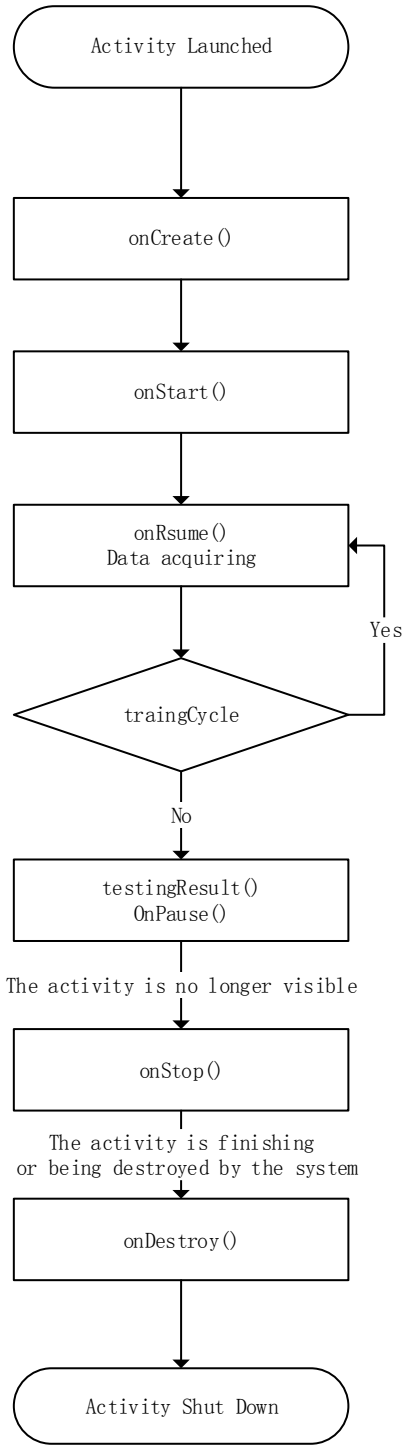

FIGURE 11 OUR SYSTEM ACTIVITY

Lucky for us, there are plenty of material for us to see the background of Android activity. So the real activity chart is Figure 11. The application first require the user to input plenty training data samples. After collecting complete training data samples, the system trains the data which after pre-processing. Then it automatically goes into the testing status. After test, the application output the result to user.

IX. FUTURE WORK

A. *More test data, less training data*

The good algorithm always have high efficiency. It can use less training data and output the more accuracy results. Our experiments have more training data rather than test data which is not proper.

B. *More raw data and proper feature extraction*

Feature extraction by normal machine learning algorithm is also lacking in our experiments. After acquiring more raw data from other Android phone's accelerators, we could use more mathematical and practical way to extract the feature such as slide time window to do the feature extraction.

C. *Energy consuming*

Energy consuming is also an important issue in our paper. One way to solve the problem is to pre-detect the action of phone. Such as in the middle of the obtaining the data. We could have low sampling rate in high-like not drinking action while high sampling rate in more like the drinking water action.

D. *Time sequence rather than movement based data*

Our method just sample the raw data from one action. In the future, we could have one day data and split with actions. After that we could have one day data and process the data thus we could have the drinking time one day one time with better user experience.

X. CONCLUSION

We have present the design, implementation and evaluation of our system. A demo for our movement detection. It use linear accelerator of Android phone to acquire the raw data and use Weka algorithm such as J48 and RBFNetwork to train and test the raw data after rough processing with mathematical way processing. Then the Android phone can output the result in real time. The accuracy of the result is about 90% in our experiment.

REFERENCES

[1] David Thomas. Understanding clinical dehydration and its treatment. Journal of the American Medical Directors Association, page 292301, 2008.

[2] Chickpin. Chickpin drinking water application. https://play.google.com/store/apps/details?id=chickpin.water.

[3] Health, Fitness:body water tracker, and reminder. Aqualert drinking water application. https://play.google.com/store/apps/details?id=com.overseasolutions.waterapp.app.

[4] Weka. Weka. http://www.cs.waikato.ac.nz/~ml/weka/.

[5] Abhinav Parate, Meng-Chieh Chiu, Chaniel Chadowitz, Deepak Ganesan,and Evangelos Kalogerakis. Risq: Recognizing smoking gestures with inertial sensors on a wristband. In Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '14, pages 149{161, New York, NY, USA, 2014. ACM.

[6] Google. Android sensor. http://developer.android.com/reference/android/hardware/SensorManager.html.

[7] Robert LiKamWa, Yunxin Liu, Nicholas D. Lane, and Lin Zhong.Moodscope: Building a mood sensor from smartphone usage patterns. In Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys '13, pages 389{402, New York, NY, USA, 2013. ACM.

[8] A. Jain and D. Zongker. Feature selection: evaluation, application, and small sample performance. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 19(2):153158, Feb 1997.

[9] Thales Korting. C4.5 algorithm and multivariate decision trees. Image Processing Division, National Institute for Space Research, page 292301, 2008.

[10] Google. Android developer. http://developer.android.com/index.html.

[11] Android Activity : http://developer.android.com/reference/android/app/Activity.html.