An Smartphone-based Algorithm to Measure and Model Quantity of Sleep

Alvika Gautam
IIIT-Delhi
New Delhi, India
alvika1261@iiitd.ac.in

Vinayak S. Naik
IIIT-Delhi
New Delhi, India
naik@iiitd.ac.in

Archie Gupta
IIIT-Delhi
New Delhi, India
archie12023@iiitd.ac.in

S. K. Sharma
AIIMS
New Delhi, India
sksharma.aiims@gmail.com

***Abstract*—Sleep quantity affects an individual's personal health. The gold standard of measuring sleep and diagnosing sleep disorders is Polysomnography (PSG). Although PSG is accurate, it is expensive and it lacks portability. A number of wearable devices with embedded sensors have emerged in the recent past as an alternative to PSG for regular sleep monitoring directly by the user. These devices are intrusive and cause discomfort besides being expensive. In this work, we present an algorithm to detect sleep using a smartphone with the help of its inbuilt accelerometer sensor. We present three different approaches to classify raw acceleration data into two states - Sleep and Wake. In the first approach, we take an equation from Kushida's algorithm to process accelerometer data. Henceforth, we call it Kushida's equation. While the second is based on statistical functions, the third is based on Hidden Markov Model (HMM) training. Although all the three approaches are suitable for a phone's resources, each approach demands different amount of resources. While Kushida's equation-based approach demands the least, the HMM training-based approach demands the maximum.**

We collected data from mobile phone's accelerometer for four subjects for twelve days each. We compare accuracy of sleep detection using each of the three approaches with that of Zeo sensor, which is based on Electroencephalogram (EEG) sensor to detect sleep. EEG is an important modality in PSG. We find that HMM training-based approach is as much as 84% accurate. It is 15% more accurate as compared to Kushida's equation-based approach and 10% more accurate as compared to statistical method-based approach. In order to concisely represent the sleep quality of people, we model their sleep data using HMM. We present an analysis to find out a tradeoff between the amount of training data and the accuracy provided in the modeling of sleep. We find that six days of sleep data is sufficient for accurate modeling. We compare accuracy of our HMM training-based algorithm with a representative third party app SleepTime available from Google Play Store for Android. We find that the detection done using HMM approach is closer to that done by Zeo by 13% as compared to the third party Android application SleepTime. We show that our HMM training-based approach is efficient as it takes less than ten seconds to get executed on Moto G Android phone.

Keywords–Mobile Sensing, Smart Healthcare, and Physical Analytics

I. Introduction

Quantity of sleep play an important role in physical, mental, and emotional health aspects of a person. Various sleep disorders, such as sleep apnea, insomnia, and hypersomnia [1] can cause abnormal sleep quantity. Poor sleeping habits may result in cardiovascular diseases and mental problems, such as depression, stress and anxiety. The study of sleep is essential in order to analyze any of the sleep disorders. PSG is the most the gold standard to diagnose sleep disorders. PSG involves comprehensive recording of the bio-physiological changes that occur during sleep. The PSG monitors many body functions including limb movement, brain activity via EEG, eye movements via EOG, muscle activity, skeletal muscle activation via EMG, and heart rhythm via ECG [2]. Previous research [3] has established that limb movements and EEG as vital parameters in measuring sleep quantity. Collection of sleep disorder related data using the PSG tests requires a patient to be admitted in a hospital. These tests severely limit regular long-term monitoring owing to the cost and complexity constraints associated with hospital admission. Further, the data collected in these tests may not be representative of the patient's sleep pattern in his or her home under regular conditions.

Hypothesis *The state of Sleep and Wake can be inferred from the amount of body movement during sleep. Deeper the sleep, lesser the body movement. The body movement leads to movement of the mattress. This movement is captured by mobile phone's accelerometer sensor, which in turn is indicative of Sleep or Wake state of the person.*

In this paper, we use this hypothesis and use off-the-shelf smartphone, to detect sleep. Although there exists smartphone apps, which claim to measure sleep quantity, to the best of our knowledge ours is a first work that presents algorithms and compares accuracy with Zeo sensor that is derivative of EEG.

Main Contributions

- We propose a simple algorithm, suitable for a smartphone, to classify its accelerometer data into Sleep/Wake states
- We verify the results using Zeo sensor that accuracy of detecting sleep of the proposed algorithm is as much as 84%
- We use HMM to model sleep of an individual in as less as six days of training
- We do an experimental analysis to show that sleep detection done using HMM training-based approach is closer to that done by Zeo by 13% as compared to the third party Android application SleepTime
- We show that our algorithm takes less than ten seconds for executing on an off-the-shelf Android phone.

Organization of this Paper The rest of the paper is organized as follows. In section II, we discuss literature related to the approaches adopted to recognize user activity, sleep in specific, from a mobile device. Section III includes problem statement and our approach for a solution. Section IV gives details of

modeling of sleep. Section V focuses on our experimental setup and the data collection. In section VI, we discuss in detail our three approaches to detect sleep. Section VII presents comparison of the three approaches and a third party Android application in terms of accuracy. It also presents an analysis of our technique to model sleep. The paper is concluded in section VIII and section IX gives future work.

II. Related Work

Smartphones have been used in the recent past for user activity recognition. Such activities are usually defined in the context of the intended application. For instance, an activity recognition system inside a car would try to decipher whether the car is accelerating, decelerating, or stopped. Similarly, in the context of a home, previous studies have used smartphones for energy apportionment tasks. In our intended application, we aim to characterize human sleep activity using the accelerometer sensor of a smartphone. This sensor returns a real valued estimate of acceleration along the X, Y, and Z axes, from which velocity and displacement can be estimated. Accelerometers have also been used as motion detectors [4], body-position, and posture sensing [5]. Apple's iLife fall detection sensor, which embeds an accelerometer and a microcomputer to detect falls, shocks, or jerky movements, is a good example. Active research is being carried out in exploiting this property for determining user context [6]. Activity recognition is mainly a classification problem and the complexity of recognition is activity dependent. For example, detecting running is simpler than that of limb movements during sleep because the difference of acceleration is greater in walk or run as compared to sleep or wake.

Advances permit accelerometers to be embedded within wristbands, bracelets, and belts and to wirelessly send data to a mobile computing device that can use the signals to make inferences. A number of commercial wearable devices have emerged in recent times, which enable users to monitor their sleep on a regular basis. Most of these devices use a combination of inbuilt sensors. One such example is the Zeo headband, which uses EEG to measure sleep data [7]. The Zeo headband interacts over Bluetooth with a mobile device for data visualization. However, most of these devices though accurate are obtrusive and cumbersome to use, since the user has to wear them while sleeping. Logistics such as maintaining Bluetooth connection and battery constraints make these sensors challenging to use. Such headbands might also cause minor discomforts, such as numbness, headaches, and skin irritation to the user. Ren et al. [8] used earphone of an smartphone to monitor breathing rate of a subject during sleep. Their proposed algorithm involves removal of noise to extract signal using a band pass, followed by noise subtraction using Fourier transform analysis in frequency domain, and pattern recognition of the extracted signal to detect periodic breathing cycles. They used NEULOG [9] respiration motion sensor attached to the ribcage as the ground truth to analyze accuracy of their approach. While their work does not translate breathing rate into Sleep and Wake states, our work translates movements detected by accelerometer into Sleep and Wake states.

There are a number of Android applications in the market for sleep monitoring but there hasn't been much technical analysis regarding their accuracy [10], [11]. Some of these applications use a number of sensors, e.g. microphone and light sensors, in addition to accelerometer to achieve better results [12], [13]. Using multiple sensors in addition to accelerometer results in higher power consumption and a substantial increase in the task complexity. We develop an Android application to collect data using only accelerometer sensor. We validate our approach empirically by comparing the results obtained with those from Zeo sensor and a third party Android application Sleep Time [14] from Google Play, which also uses only accelerometer.

III. Problem Statement and Proposed Approach to Solution

Problem Statement To develop mathematical model to detect and characterize an individual's sleep pattern by using smartphone accelerometer data.

In this paper, we present a novel approach to measure sleep using a mobile device and its inbuilt accelerometer sensor. The accelerometer is able to accurately measure limb movement, which is an important vital to measure quantity of sleep. To test the validity of the approach, we compare our results with that obtained from Zeo sensor. Although a smartphone doesn't have same accuracy as that of a PSG test but the obtained accuracy is sufficient enough to enable its use as a screening device. Since the smartphone is placed on the mattress, its accelerometer can detect movements of the mattress, and in effect limb movements.

We present the implementation of our approach, where data collection is implemented as an Android mobile application and further analysis of the collected data is done in order to measure sleep. We conducted an experimental study with different subjects using different mobile devices in order to test the robustness of our application. In the experiment, each user recorded the data for eleven consecutive days to maintain uniformity in results. The obtained results are compared with those from Zeo, which serves as the ground truth for our approach.

IV. Modeling of Sleep Pattern

We model the sleep pattern of users as a stochastic process.Probabilistic modeling of processes helps us to achieve a compact representation of the system characteristics, thus making it easier to compare two instances of a process. Sleep of a person is an example of such a process because it is different from one subject to another subject and shows random day-to-day variations. In order to do a simple qualitative analysis of sleep pattern of a person across a number of days, we use HMM. HMM permits analysis of non-stationary multivariate time series by modeling the state transition probabilities and state observation probabilities. Modeling of sleep should consider the relationship between the previous sleep stage and the next sleep stage. During the HMM process, result of the previous state will influence the state recognition result of the next state. As it possesses the properties of successive stage transition, HMM is a promising model for sleep modeling [15]. The model derived using this can also be used to compare the sleeping pattern of multiple subjects since the aim of modeling is to characterize an individual's sleep.

In this section, we present background on HMM and HMM Viterbi algorithm. The latter is used to take a particular HMM and determine from an observation sequence the most likely sequence of underlying hidden states that might have generated it.

A. HMM

HMM is a statistical Markov model, in which the system being modeled is assumed to be a Markov process [16] with unobserved, i.e., hidden states. In HMM, state is not directly visible, but output dependent on the state is visible [17]. Each state has a probability distribution over the possible visible states. Therefore, the sequence of outputs generated by HMM gives some information about the sequence of hidden states. Key characteristics of a HMM are as follows.

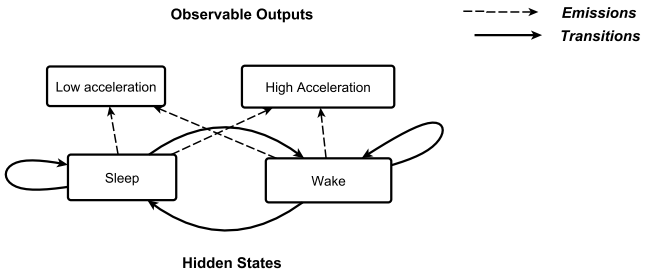

Figure 1: HMM state transition diagram of Sleep-Wake process

- Transition probability matrix $T(i, j)$ represents probability of going from state i to j. Order of matrix = $S * S$ where, S is the number of states.
- Emission probability matrix $E(i, k)$ represents probability that output k is emitted from state i. Order of matrix = $(S * V)$ where S = number of hidden states and V = set of vocabulary for observable outputs.
- Posterior Probability matrix $P(i, j)$ is an array with the same length as observation sequence and one row for each state in the model. The (i, j) element of P gives the probability that the model was in state i at the j^{th} step of sequence.

Figure 1 shows our 2-state HMM. Here, observable outputs are the acceleration magnitude values obtained from mobile's accelerometer, which are grouped into Low and High accelerations. The two hidden states are Sleep and Wake.

B. HMM Viterbi Algorithm

HMM Viterbi algorithm deals with the problem of estimating the state sequence, which can best represent the observed data. P is the probability of being at state n at time i, having come from m, T is the transition probability from m to n, and O is the probability of that specific observation at time i from state n. Using Markov property that the current state only depends on the last state, considering the process from the beginning, probabilities of sequences happening can be calculated by multiplying together the corresponding transition values and observation values with the previous step's probability.

The HMM Viterbi algorithm expands on the idea that at each time step, only the sequence path that has the best probability going into each state needs to be stored. If the model has two states then at the two paths need to be stored and updated at every time step. HMM learning involves the calculation of maximum likelihood estimate of the transition and emission probabilities, given an observation sequence and the corresponding state sequence. Therefore, learning helps in finding the best HMM parameters that can explain a set of observation and state sequences. Once these parameters are learned these can be used to find the hidden state sequence for a new set of observation sequence for the same process. In our paper, this concept is applied to learn the parameters for modeling an individual's sleep.

V. Data Collection

A. Our Android Application for Data Collection

We developed an Android application to sample accelerometer data and provide user with an estimate of his or her sleep quantity. Once the user registers, a folder with the name of the user is created on phone's SD card.The registration and logging in enables multiple users to use the application on the same phone, as different folders get created for different users. The data is collected as a CSV file. The data from accelerometer has timestamp and acceleration along X, Y, and Z axes respectively. We take magnitude of accelerometer sampled data as $\sqrt{x^2 + y^2 + z^2}$.

B. Experimental Setup for Data Collection

Four subjects were asked to place their mobile phones near the pillow, with our application running, in order to collect the data. We had consent from the subjects to do so. The subjects started the application by pressing the start button in the app as soon as they lie on the bed and pressing the stop button after waking up. In addition to our app, the data was collected from another Android application called SleepTime, which is a third-party app to measure sleep quantity, using the same phone. The data was also collected from Zeo sensor with other phone receiving the data recorded, through Bluetooth. .

Both our app and SleepTime use accelerometer for detecting sleep. By virtue of running these apps on the same phone, we ensure that the same hardware was used by both the apps. We can then fairly compare the accuracy of both the apps. Zeo, which uses EEG to detect sleep, does not use accelerometer. It is used as ground truth to estimate the accuracy of our approach for detecting sleep. Figure 3 shows the experimental setup for data collection with the subject wearing the Zeo headband and two different phones near the pillow. We used different Android phones, Samsung, namely HTC-Desire, HTC Wildfire, and Sony Xperia, to establish the generality of our approach.

VI. Three Approaches to Detect Sleep/Wake States

We need an approach, whose resource requirements can be satisfied by a mobile phone. We present three approaches to

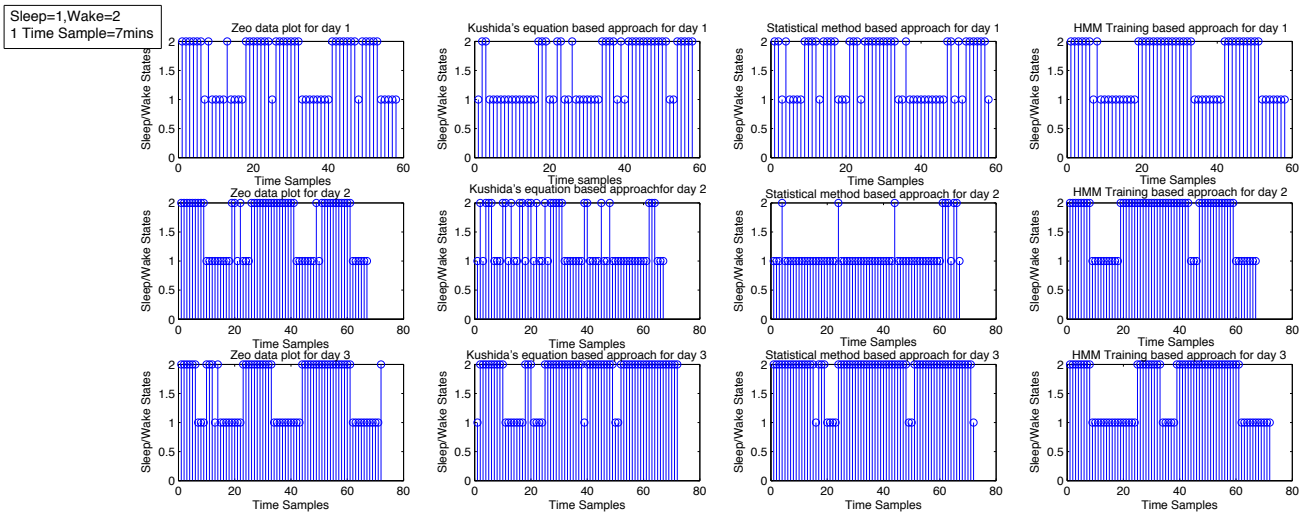

Figure 2: Comparison of classified data plots using Zeo, Kushida's equation-based, Statistical method-based, and HMM training-based approaches for three days. The readings are consistent across the three representative days. Among the three, the plot of HMM training-based approach matches that of Zeo more closely.

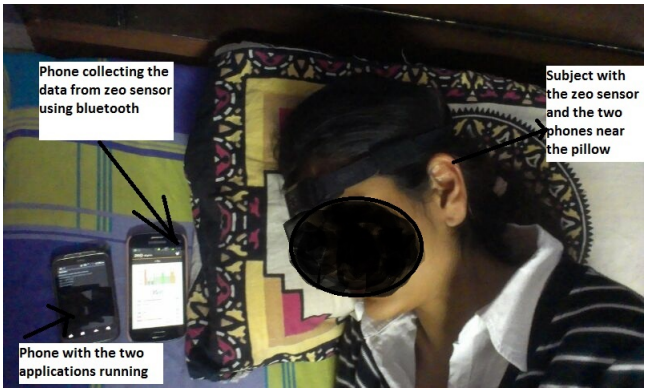

Figure 3: Experimental setup shows data collection on two Android phones and one Zeo sensor. One phone receives the data collected by Zeo using Bluetooth and the other phone has our data data collection app and third party SleepTime mobile app.

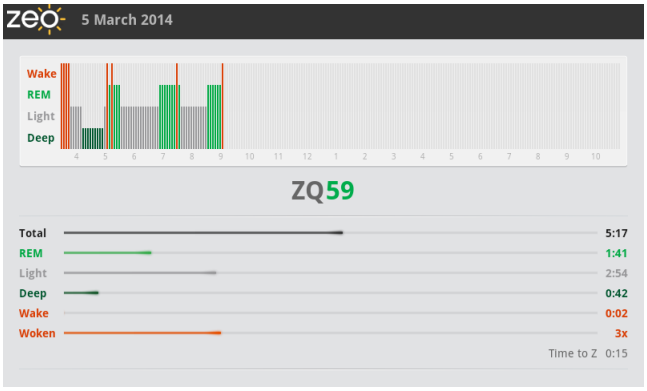

Figure 4: Output graph from Zeo sensor classifying sleep into REM, Light, Deep, and Wake states.

detect sleep, with increasing order of demands on resources. Two of these approaches use fixed thresholding, where as the third approach use a probabilistic modeling approach.

A. *Kushida's Equation-based Approach*

First approach involves classifying the raw accelerometer data using Kushida's equation. This equation modifies data by taking into account the data in the neighboring time windows, both earlier and later. For the raw accelerometer time series data, the acceleration measured at the current time sample was modified according to the accelerometer values of $\pm$ 4 time samples in time as shown in equation1

$$A_{modified} = 0.04 * A_{n-4} + 0.04 * A_{n-3} + 0.20 * A_{n-2} + 0.20 * A_{n-1} + 2 * A_n + 0.20 * A_{n+1} + 0.20 * A_{n+2} + 0.04 * A_{n+3} + 0.04 * A_{n+4} \quad (1)$$

where, $A_{modified}$ is the modified value for the present time sample, A_n is the acceleration value at the present time sample, and $A_{n\pm x}$ are the acceleration values at the surrounding time samples. Kushida's equation-based approach takes the modified time series as given in equation 1 and threshold as input. If the summed acceleration value obtained from equation 1 was above a certain threshold, the epoch was scored as Wake, otherwise as Sleep. The pseudocode is mentioned in Algorithm 1.

The value of threshold was selected empirically, which involved use of trial and error method by using multiple thresholds and asking the subjects which of the threshold resulted

in more accurate classification of sleep. It was observed that a high threshold range gave false sleep epochs and a low threshold gave false wake epochs. Better results were obtained using medium value for threshold and thus further analysis and comparison was done with the help of the medium threshold. The algorithm is computationally least expensive among the three approaches.

Algorithm 1: Kushida's Equation-based Approach

Data: acurr
Result: State
$acurr \leftarrow$ *raw accelerometer series* ;
$amod \leftarrow$ *modified accelerometer series* ;
$len \leftarrow$ *length of acurr* ;
$i \leftarrow$ *1* ;
read current;
while $i \leq len$ **do**
  if $(i > 4)$ *and* $(i < len - 4)$ **then**
    $amod(i) =$
    $0.04 * acurr(i-4) + 0.04 * acurr(i-3)$
    $+0.20 * acurr(i-2) + 0.20 * acurr(i-1) +$
    $2 * acurr(i)$ $+0.20 * acurr(i+1) + 0.04 *$
    $acurr(i+2) + 0.20 * acurr(i+3)$
    $+0.20 * acurr(i+4)$;
  else
    $amod(i) = acurr(i)$;
  end
  $i \leftarrow i + 1$;
end
$thres \leftarrow$ *Selected threshold* ;
$j \leftarrow$ *1*;
$State \leftarrow$ *Sleep/wake state map*;
while $j \leq len$ **do**
  if $amod(j) < thres$ **then**
    $State(j)$ = 1
  else
    $State(j)$=2
  end
  $j \leftarrow j + 1$
end
return *State*

B. Statistical Method-based Approach

Second approach involves a simple statistical technique, in which data is first processed by discarding the noisy data, in the form of peaks of high amplitude. After removal of noise, the data is normalized to an amplitude range of zero to one. Normalization is done in order to compensate for the variations in accelerometers of different phones.

The normalized data is viewed in windows of four minutes each and one of the two states, Sleep or Wake, is assigned to every window. The basis for this detection is that if a certain number of samples in each window, in our case 40% of the samples, have magnitude greater than the threshold, that window is classified as Wake, otherwise Sleep. The value of the threshold was calculated using the following equation.

$$Threshold = \frac{Mean + StandardDeviation}{2} \quad (2)$$

It is a well known statistical equation to decide thresholds [18]. Algorithm 2 gives the pseudocode of the approach. While the first approach selected threshold based on trial and error method, this one uses a statistically sound method. However, this approach uses more amount of data to compute threshold as compared to the first approach.

Algorithm 2: Statistical Method-based Approach

Data: normalizeddata
Result: State
$normalizeddata \leftarrow$ *data after noise removal*;
normalization to range 0 to 1 ;
$len \leftarrow$ *length of normalizeddata*;
$num \leftarrow$ *number of samples in 4 min window*;
$j \leftarrow$ *1*;
while $j \leq len - num$ **do**
  $i \leftarrow$ *1*;
  $k \leftarrow$ *1*;
  $thres \leftarrow (mean + stddev)/2$;
  while $i \leq num$ **do**
    if $acurr(i) < thres$ **then**
      $count = count + 1$
    end
    $i \leftarrow i + 1$;
  end
  if $count < 0.4 * num$ **then**
    $State(k)$ = 1
  else
  end
  $State(k)$ = 2
  $count \leftarrow$ *1*;
  $i \leftarrow$ *1*;
  $j \leftarrow$ *j+num*;
  $k \leftarrow$ *k+1*;
end
return *State*

C. HMM Training-based Approach

Third approach involves the detection of Sleep and Wake states using HMM training. The model is trained using the HMM Viterbi algorithm. HMM Viterbi algorithm takes the observation sequence and the sequence of states corresponding to the observation sequence as inputs and gives emission and transition probabilities as output. In our case, the inputs are raw acceleration values and the Zeo sensor output containing two states; Sleep and Wake after appropriate mapping as explained in section V-B. The acceleration values are down-sampled to match the sampling rate of the Zeo sensor. These transition and emission probabilities are estimated using the state map as given by Zeo, which is the ground truth for Sleep-Wake states. The obtained probabilities, along with a new set of raw accelerometer data of the same subject, are then used to estimate the sequence of Sleep/Wake states corresponding to this newly collected acceleration values, i.e., observation sequence. Algorithm 3 gives the pseudocode.

While the first two approaches have fixed threshold, this approach has probabilistic threshold. Unlike the first two approaches, this approach involves training, which requires more resources in terms of sensor and computation.

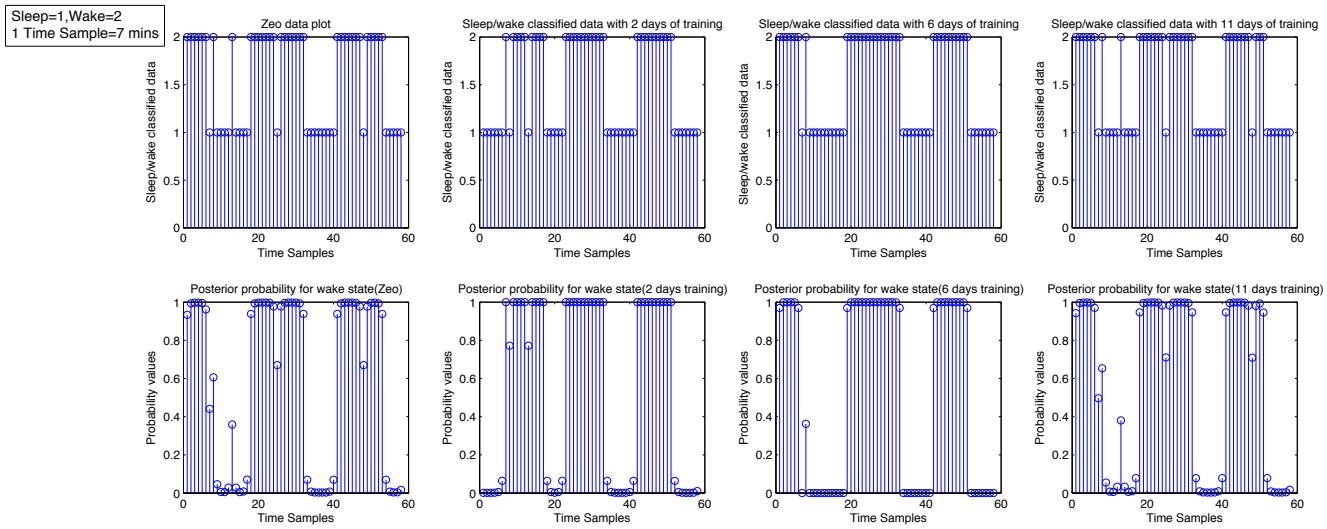

Figure 5: Evaluation of HMM approach showing the detection and posterior probability plots for 2, 6, and 11 days of training data respectively. Detection accuracy was 69%, 84% and 85% for 2, 6 and 11 days of training data respectively. Comparing the detection and posterior probability plots among 2, 6, and 11 days of training data respectively, 6 days of training data is provide sufficient accuracy.

Algorithm 3: HMM Training-based Approach

Data: acurr
Result: State
$amod \leftarrow$ *modified accelerometer series* ;
$len \leftarrow$ *length of acurr*;
$i \leftarrow$ *1*;
$zeodata \leftarrow$ *Extracted data from Zeo sensor* ;
$data \leftarrow$ *after noise removal and normalization*;
$datafin \leftarrow$ *downsampled to match the rate of Zeo sensor*;
(*State*)=$runHMM$(datafin, datagrp,zeogrp)▷ Calculates state map
return *State*

Algorithm 4: *runHMM()* function used in HMM Training-based Approach

Data: datafin,datagrp,zeogrp
Result: State
$dataquant \leftarrow$ *Quantize datafin*;
$datagrp \leftarrow$ *m by n matrix*;
$m \leftarrow$ *no. of days* $n \leftarrow$length of accelerometer data;
$zeogrp \leftarrow$ *m by n matrix*;
$m \leftarrow$ *no. of days* $n \leftarrow$ *length of Zeo data*;
(*trans,emiss*)=$hmmestimate(datagrp, zeogrp)$ ▷ Maximum likelihood estimate
(*State*)=$hmmviterbi(seq, trans, emiss)$ ▷ Calculates state map
return *State*

VII. Analysis and Results

In the following subsections, we will compare results of the three approaches in terms of their accuracy of classification. We will present tradeoff between accuracy of classification and the amount of training data required in the HMM approach. We will compare accuracy of classification with a third party Android application and finally analyze the performance of our algorithm on an Android smartphone.

A. Comparison of Accuracy in Classifying Sleep and Wake States

We compared the accuracy obtained by three different approaches with the ground truth obtained from the Zeo sensor. Zeo classifies data into four states Wake, REM sleep, Light sleep, and Deep sleep as shown in Figure 4. Our Sleep and Wake detection is compared to that of Zeo. This is done by mapping Zeo's four states into two states. Zeo's Wake and Light sleep states were mapped to our Wake state, Deep sleep and REM sleep were mapped to our Sleep state. From medical point of view, this generalization of states is sound. Raw accelerometer data was classified into Sleep and Wake states using three different approaches, namely detection using Kushida's equation, Statistical method, and HMM training as discussed in section VI. We used six days of training for HMM training-based approach.

The metric used for quantitative comparison is the *Percentage of Matching Samples*, calculated as follows.

$$Matching\,Samples = \left(\frac{x}{y}\right) * 100 \tag{3}$$

where, x = Number of matching samples and y = Total number of samples in the data

This is a simple and reliable metric for comparison as it captures the accuracy of the detection by direct comparison with the ground truth classified data. This metric comparison can be further extended to give a percentage of false sleep and false wake epochs.

Table 1 shows comparison of accuracy. The classification data obtained from each approach are compared with that from Zeo and the percentage of the matching samples was found out to be maximum for HMM training approach, greater than 80% on average. Figure 2 shows ground truth from Zeo and the output of the three approaches for one subject. It was also observed that HMM training approach identified small wake epochs in between sleep epochs more accurately and more consistently as compared to the other two approaches. Similar results were obtained for other three subjects as well.

Name of the Approach	Accuracy
Kushida's Equation-based Approach	• Max 65% • Avg 59%
Statistical Method-based Approach	• Max 74% • Avg 68%
HMM Training-based Approach	• Max 84% • Avg 79%

TABLE I: Comparison of the three approaches in terms of accuracy in classifying Sleep and Wake states

B. Tradeoff Between Amount of Training Dataset and Accuracy of Detection using HMM Training

In this subsection, we will analyze the HMM training approach to find out what is the tradeoff between the amount of training dataset and resulting accuracy. The HMM training approach enables modeling of the sleep pattern from HMM parameters in terms of transition, emission, and posterior probabilities.

Data is classified using varying amount of training data ranging from two days to eleven days and then modeled. Figure 5 shows the detection as well as posterior probability plot of Wake state obtained using two, six, and eleven days of training, respectively. These plots are for a common day for a fair comparison. The comparison metric is same as that mentioned in section VII-A. Overall, it was observed that the accuracy uniformly increased from two to six days of training data and after that the increase in accuracy was insignificant. However, with more training data, the model became qualitatively strong as it was observed that it was able to detect Wake states of small duration in between larger duration of Sleep states. We conclude that six days of training data was sufficient for accurate detection. Training is a one-time cost and six days of training is not significant.

Training days	Accuracy obtained
2	69%
6	84%
11	85%

TABLE II: Comparison of accuracy based on the amount of training data in HMM Training

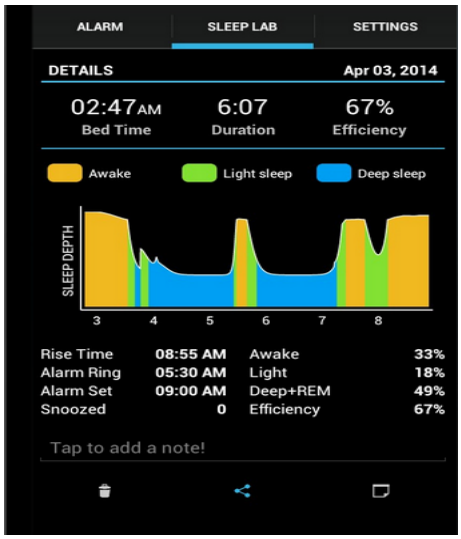

Figure 6: Output of the third party Android application SleepTime

C. Comparison With a Third Party SleepTime Android Application

Although many third party applications are available for detecting sleep using smartphones, it is unclear as to how accuracy of these application compare to that of medically approved devices. We compare accuracy of our approach with a popular third party Android application SleepTime [14] using Zeo as a ground truth. This application was made to run for the same data collection environment. A screenshot of the application is shown in 6.

The percentage of number of Sleep and Wake states after mapping to two states was 49% and 51% respectively. For the same data of the same day, the categorization using Zeo was 68% and 32% respectively. The detection using our HMM training algorithm was 62% and 38% respectively. These comparisons are made using the metric of Matching Samples explained in VII-A. The results using our approach are closer to the ground truth as compared to the third party application. The observations were consistent, when analysis was done for a period of twelve days, thus proving the statistical significance our result.

D. Performance of Our Algorithm

Android provides three different accelerometer sampling rates-fastest, normal, and UI. Fastest being the fastest and UI being the slowest. Our HMM Training-based approach uses normal sampling rate. We performed experiments on an Android phone to find out how long its battery lasts when it is continuously sampling at normal rate. We found that the battery lasts for roughly fourteen hours, which is sufficient enough if we consider average sleep duration to be around eight hours.

We used an Motorola Moto G Android phone with the following specifications.

- Qualcomm Snapdragon 400 processor with 1.2 GHz quad-core Cortex-A7 CPU

- GPU Adreno 305
- 1 GB RAM
- Android OS version 4.4.4

Our algorithm took ten seconds to run on the phone, which means our approach is suitable for a phone platform.

VIII. Conclusions

Given the importance of sleep for health, detecting quantity of sleep in an non obtrusive manner is desired. Smartphones with their inbuilt accelerometers have potential for such detection. Although there are many applications for detecting sleep in the smartphone app markets, there is a lack of study about algorithms that these app employ and accuracy which these algorithms provide. Some of these apps also consume significant power.

In this paper, we study candidate approaches to detect sleep using inbuilt accelerometers on smartphones. All of these approaches are suitable for smartphones' resources. We measure and compare accuracy provided by these approaches with that of EEG-based Zeo sensor. The HMM training-based approach provides the maximum accuracy, with only six days of training. The algorithm takes ten seconds on a off-the-shelf Android phone.

IX. Future work

In present work, sleep cycles were identified using a number of approaches and sleep was modeled using a simple first order HMM. This model can be extended to classify sleep into more number of states by increasing the order of the model. In our future work, we will evolve our detection into all four states similar to that of Zeo.

Our system can be extended to detect more complex sleep disorders using additional sensors. For example, sleep apnea, which a sleep disorder, can be detected by using the mobile accelerometer sensor with an additional external pulse oximeter sensor and breathing rate. The pulse oximeter measures the oxygen saturation and pulse rate of a subject. These sensory readings along with the data of body movement can be used to detect the probability that a person might be suffering from sleep apnea using a smartphone.

References

[1] [Online]. Available: http://en.wikipedia.org/wiki/Sleep_disorder

[2] G. Jean-Louis, D. F. Kripke, W. J. Mason, J. A. Elliott, and S. D. Youngstedt, "Sleep estimation from wrist movement quantified by different actigraphic modalities," Journal of neuroscience methods, vol. 105, no. 2, 2001, pp. 185–191.

[3] P. Estévez, C. Held, C. Holzmann, C. Perez, J. Pérez, J. Heiss, M. Garrido, and P. Peirano, "Polysomnographic pattern recognition for automated classification of sleep-waking states in infants," Medical and Biological Engineering and Computing, vol. 40, no. 1, 2002, pp. 105–113.

[4] R. W. DeVaul and S. Dunn, "Real-time motion classification for wearable computing applications," 2001, project paper, http://www. media. mit. edu/wearables/mithril/realtime. pdf, 2001.

[5] F. Foerster, M. Smeja, and J. Fahrenberg, "Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring," Computers in Human Behavior, vol. 15, no. 5, 1999, pp. 571–583.

[6] C. Randell and H. Muller, "Context awareness by analysing accelerometer data," in Wearable Computers, The Fourth International Symposium on. IEEE, 2000, pp. 175–176.

[7] B. Rubin, "Multi-modal sleep system," Sep. 6 2011, uS Patent App. 13/226,121.

[8] Y. Ren, C. Wang, Y. Chen, and J. Yang, "Poster: Hearing your breathing: Fine-grained sleep monitoring using smartphones," in Proceedings of the 20th Annual International Conference on Mobile Computing and Networking, ser. MobiCom '14. New York, NY, USA: ACM, 2014, pp. 409–412. [Online]. Available: http://doi.acm.org/10.1145/2639108.2642898

[9] [Online]. Available: http://neulog.com/

[10] [Online]. Available: http://www.sleepcycle.com/

[11] [Online]. Available: https://sites.google.com/site/sleepasandroid/

[12] Z. Chen, M. Lin, F. Chen, N. D. Lane, G. Cardone, R. Wang, T. Li, Y. Chen, T. Choudhury, and A. T. Campbell, "Unobtrusive sleep monitoring using smartphones," in Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2013 7th International Conference on. IEEE, 2013, pp. 145–152.

[13] T. Hao, G. Xing, and G. Zhou, "isleep: unobtrusive sleep quality monitoring using smartphones," in Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems. ACM, 2013, p. 4.

[14] [Online]. Available: https://play.google.com/store/apps/details?id=com.azumio.android.sleeptime

[15] S.-T. Pan, C.-E. Kuo, J.-H. Zeng, S.-F. Liang et al., "A transition-constrained discrete hidden markov model for automatic sleep staging," Biomed. Eng. Online, vol. 11, 2012, p. 52.

[16] [Online]. Available: http://en.wikipedia.org/wiki/Markov_process

[17] B. G. Leroux, "Maximum-likelihood estimation for hidden markov models," Stochastic processes and their applications, vol. 40, no. 1, 1992, pp. 127–143.

[18] T. L. Alves, C. Ypma, and J. Visser, "Deriving metric thresholds from benchmark data," in Software Maintenance (ICSM), 2010 IEEE International Conference on. IEEE, 2010, pp. 1–10.