Somnia: Sleep Actigraphy Approximation

Medha Potluri, Vidhart Bhatia, Udit Ranasaria
Carnegie Mellon University, Pittsburgh, PA, USA
mpotluri@andrew.cmu.edu, vnbhatia@andrew.cmu.edu, uar@andrew.cmu.edu

***Abstract* -- Many sleep sensing apps on the market are able to classify sleep using traditional signal analysis. We train an ML Model to detect light or deep sleep. These classifications are used by an app to implement features based on when a subject has fallen asleep or is awakening.**

***Index Terms* -- Actigraphy, Android Sensing, IoT, Weka**

***Github Repository:* github.com/vidhartbhatia/somnia**

I. INTRODUCTION

Vision

There are countless apps on the market that analyze and classify your sleep, but we wanted to make our own that catered to our specific needs. A common problem in our daily lives is the large disruption in sleep when one person has an alarm before the other in a shared bed. One of our primary goals was to build a system that would be able to carefully track individual sleep in a shared bed scenario and then carefully attempt to wake up only that person. Furthermore, we wanted to use smart-light APIs to ensure that the lights in the bedroom turn-off whenever participants in a room have fallen asleep.

While achieving our original vision became quite difficult due to time constraints and hardware access difficulties, we focused on the general idea of classifying sleep data and building basic falling asleep and awakening functionality.

Sleep Phases

Traditionally sleep phases are tracked using an EEG, in a study called Polysomnography (PSG). The scientific community has settled on classifying sleep into 4 distinct phases:

- N1 – a short transitional stage between sleep and wakefulness.
- N2 – mostly light dreamless sleep that occupies about 60% of the night.
- N3 – the deepest sleep phase. EEG displays slow regular waves. This stage is believed to play a crucial role in the regenerative processes. The body is most relaxed in this phase.
- REM – high likelihood of vivid dreams, muscle paralysis, bursts of rapid eye movements.

The phases alternate in a typical sequence called the sleep cycle (Fig. 1) – from a short N1, via N2, to the deep N3, then a shorter transitory N2, followed by REM, interrupted by brief awake.

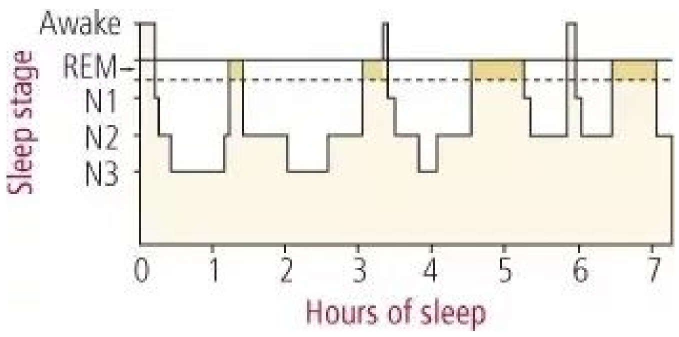


Fig. 1: Example Sleep Cycle Graph

The cycle repeats several times throughout the night. Since the goal of our project is to wake up people at the right time, being able to tell when someone is in a particular phase is part of our goal. Instead of using an EEG we wanted to make use of activity data to approximate sleep phase tracking, as is done by many popular sleep cycle applications.[1]

Actigraphy

This leads us to actigraphy, which is the act of monitoring human activity in a non intrusive manner. There is a lot of prior research showing a link between relative body movement and sleep phases[1][3] and many popular applications make use of accelerometer data to approximate sleep phase. There are also papers that made use of accelerometer data to build predictive models for sleep phase tracking[2]. Due to this existing research we deemed it sufficient to use accelerometer data for our project. We also decided to use Sleep as Android (SaA) as our main reference application for tagging data and to compare results, as explained later. One thing to note is that for the purposes of our project, we did not need to be able to classify all 4 phases of sleep, instead we needed to be able to determine when someone was in "Light" or "Deep" sleep, as that was the most important distinction needed to be able to implement our features. The main idea was that people feel more refreshed if they are woken up from light sleep over deep sleep, so we wanted to focus on being able to distinguish between those two phases first.

II. DEVELOPMENT PROCESS

Initial Approach

Our original approach was going to involve building a dedicated sleep device by combining various sensors with an IoT Arduino or Particle system. That said, after the

[1]https://sleep.urbandroid.org/how-do-we-measure-your-dreams

COVID-19 pandemic, we wanted to minimize efforts and time-loss due to shipping and attempting to build hardware in our homes. Thus, we ended up choosing the smartphone as our ready all-in-one solution for this project.

Now our problem was scoped as classification of sleep phases based on sensor signals collected from Android phones. This problem seemed very approachable from the 2 methods learned through Project 1 and Project 3. One approach would be using traditional signal analysis via plotting, noise reduction, peak detection and the likes while another approach would use Weka to build an ML model.

The goal was that we would split up the approaches while sharing updates and findings between each and hopefully end up with a final solution that integrated elements of both.

App and Data Collection

Before jumping into signal analysis, we needed a basic system to collect our sensor data. Since we are asleep while collecting data, we can’t actually use “live” plotting data in any meaningful way. We had to figure out a way to record the data in volume and export them all in the morning

We decided to use an application we found on GitHub that was built for a research lab[2]. We just didn't want to spend too much time learning about how to build a background service and use SQL databases in the android API, so making use of existing code helped us get started. With modifications, the initial version of our application was pretty simple - when directed it would launch a background service that would collect sensor data and store it in a SQL database. The data could then be exported to a CSV through the app so we could analyse and work with the data. We settled on a sample rate of 10 Hz as we wanted to collect a lot of data over the course of a night, but we couldn't have too high of a sample rate as it would make the exported files extremely large and hard to work with. Our goal was to have the phone lying face up anywhere on the same mattress as the user sleeps. In order to be able to account for this we collected accelerometer data for all three axes. Once we collected raw sensor data for a few nights of sleep we began the task of graphing and analysing the data, as is explained in the Traditional Approach section.

Labeling Data

We realized early on that we would need some sort of reference information for our collected data, otherwise it would be impossible for us to come up with an effective way of analysing it. Without being able to tag out collected data with sleep phases it would be impossible to come up with thresholds or train an ML algorithm. At first we attempted to take a video recording of us sleeping but encountered a lot of difficulty as the file sizes were just too large and getting hardware was not feasible. We decided to use an existing sleep tracking and smart alarm application that made use of sensor data. After doing research and testing several options we settled on Sleep as Android as it had the best sleep phase graphs, seemed to make use of accelerometer data instead of combining it with sonar as many other apps did, and had extensive research to support their methods. According to their documentation SoA used relative activity in 5 minute windows to distinguish between light, deep and REM sleep, which is exactly what we hoped to do. We hoped that labeling our data based on one minute intervals as classified by SaA would give us enough information in order to develop our own approach. After a lot of iteration on models we eventually decided to label all our data as deep or light sleep, as REM is extremely hard to distinguish from light sleep. The SaA documentation also confirmed that REM is almost impossible to classify accurately and their data used a lot of guessing and approximation for REM phases. From our observation, Sleep as Android doesn’t do too well at differentiating between light sleep and awake. We saw several periods where we knew we were awake classified as light sleep by Sleep as Android. Thus we decided to tag periods classified as awake by Sleep as Android as light sleep as well. Additionally, the applications of our analysis only involve differentiating between light and deep sleep for now, and not approximating sleep cycles.

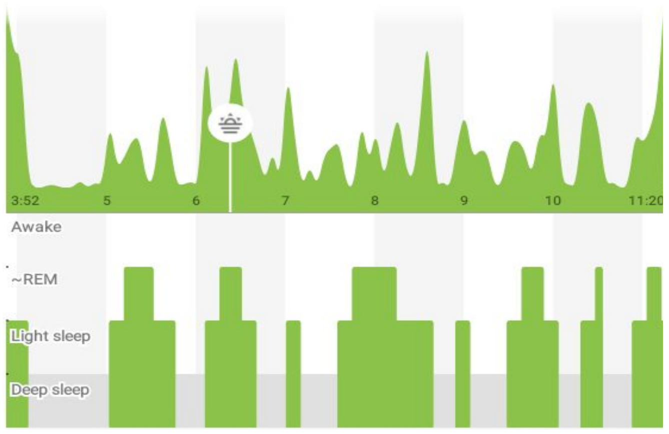


Fig. 2: Sleep as Android sleep graph example showing deep, light and REM

Traditional Approach

Upon reviewing our first night’s worth of phone sensor data, we realized that our data was not only dense but also high in volume. To quantify that, one 10 minute period has 6000 rows of data which means a regular 8-hr sleep night has about ~300000 rows of data. This meant when we tried to graph all the data at once “simply” using Excel or MatPlotLib, we ended up with plots that were incredibly difficult to analyze (Fig 3.).

[2] https://github.com/sho-87/sensor-record

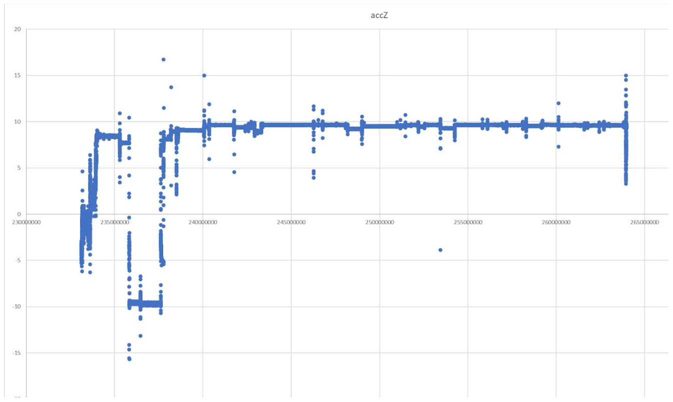


Fig. 3: Acceleration-Z signal over a full night graphed at once

To combat this issue, we felt the need to invest some time and build a "data viewer" application that would provide us a parametrized scrollable zoomed in look on the data. We also knew that in the future this would give us control to compare multiple streams of data using synchronized scrolling subplots. We created this data viewer application using Python, MatPlotLib, and WxPython.

The first thing we quickly noticed was that even during the lowest of movements, there was a lot of noise. For us to get a cleaner signal of movements we employed a smoothing filter to get a cleaner look at each signal (Fig. 4).

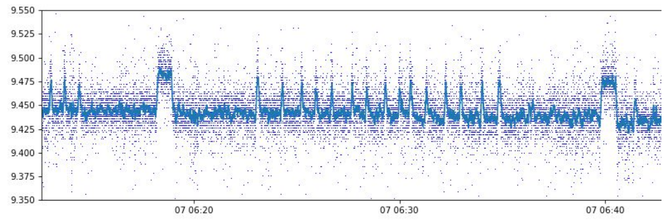


Fig. 4: Example of Acceleration Z signal interval zoomed in using our data viewer application. The strong blue line in the middle is the smoothed signal while the scattered dots are the raw data values

When we started graphing different sensor streams (acceleration, gyroscope, magnetometer), we saw quickly that acceleration was the signal that had the most usable signals. We saw peaks and movement in acceleration corresponding to the times that SaA classified the test subjects sleep as light. We saw much less correlation with the gyroscope or magnetometer and all research and other apps (e.g. SaA) seem to use accelerometer data only. Thus, we decided to pursue an accelerometer only approach.

We also realized that while we were seeing small spikes/peaks in the smoothed data signal for acceleration, writing peak detection code to classify light/deep sleep was difficult. This is because both the amplitude of the peaks and baseline from which peaks rise from vary significantly based on the test subject and test condition (phone location, mattress variation, pillow type). We could have tried to "learn" a baseline and peak amplitude based on the first few minutes but that felt as if it would be unreliable and difficult.

Instead, we looked for a way to normalize the data and one idea we had was to use the variance of each signal over certain sample sizes. Variance would automatically normalize against the mean of that window. To compare the effects of different window sizes, we plotted 1 and 10 minute variance windows against SaA-labeled sleep phases.

For light sleep periods, there is a high correlation in variance peaks with both window sizes. That said, the one minute spikes tend to not span the period of light sleep. This means that if we were to try to classify light sleep based on 1 minute windows, we would have to count the number of 1-minute variance waves over a period of time and classify that whole period as light sleep. In contrast, taking a 10 minute window almost builds that functionality in for us by spreading those multiple 1-minute peaks in a period of time over the variance of that whole period. Thus, doing thresholds and peak detection for the 10 minute or longer windows should make classification easier (Fig. 5).

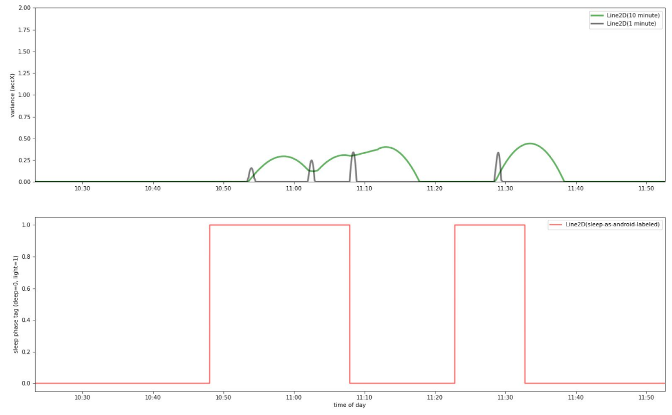


Fig. 5: Example showing variance over 1 and 10 minute windows correlating to light sleep. 10 minute windows in green spread over the whole light sleep better.

One of the next problems to tackle was how to incorporate each of the X, Y, and Z signals. When we graphed the variance of each signal, we were surprised to see that the Z acceleration signal was the smallest in peaks. Our intuition told us that the mattress depressing (i.e. movement in Z direction) would correlate highest with movement on the bed but clearly not. Furthermore, we noticed the "dominant" wave could switch between X and Y throughout the night (Fig. 6). Thus, we realized we should also consider the aggregate of the variances (varX+varY+varZ) as the most reliable signal. This would also hopefully remove reliance on the orientation at which the test subject leaves the phone on their bed.

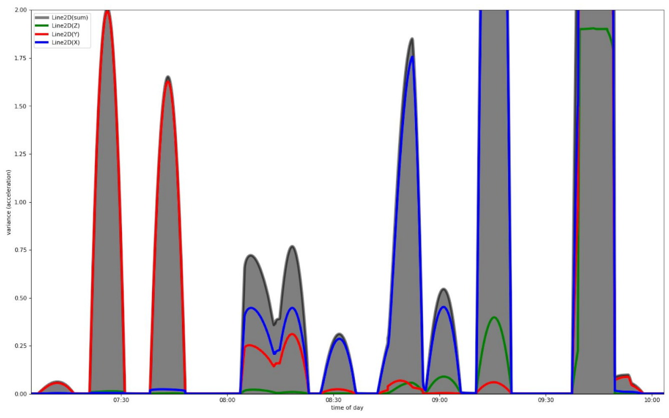


Fig. 6: Graph of Test Subject J on May 7th. Notice how the Y variance was the dominating signal for the waves between 7:00 and 8:00 but the X variance was dominating signal for the waves between 8:30-10:00

While variance does normalize the baseline well, it does not account for the fact that different testing conditions and subjects can result in dramatically different peak heights. Trying to do traditional peak detection to find generalized thresholds to classify these waves as light and deep sleep started to feel like writing a decision tree. See ML Approach section for how we used ML with this data.

ML Approach

For each night of sleep, we had somewhere around 300,000 points of data from each sensor stream. To make useful features out of all of this data, we decided to aggregate data over 1, 5, 10, 20 and 30 minute intervals, by taking the mean, median, minimum, maximum, and variance of the data points in the interval. For each minute of sleep, we considered the aggregates over the past 1, 5, 10, 20 and 30 minutes of sleep. We then tagged each minute of data with a sleep phase (light or deep) using the Sleep as Android graphs. We quickly decided to only use accelerometer data along the three axes, as we saw that the other sensors were not adding any accuracy.

We explored different ML algorithms, including J48, neural nets and more, and found that the Random Forest Classifier gave us the best accuracy. Our model with all 75 features (3 sensor streams, 5 types of aggregates, and 5 interval lengths) had ~95% accuracy when tested using 10-fold cross-validation.

However when we graphed our predicted tags against the Sleep as Android tags and the variance of acceleration for a few nights of data, we were not satisfied with the qualitative results. We will discuss some of the problems we saw and how we addressed them.

Due to some technical difficulties and ill discipline, our dataset has a lot more sleep data of test subject V than of the others. We believe this led to our model overfitting to test subject V's movement patterns and magnitudes, and thus giving us inaccurate results when testing on new data from other test subjects. To combat this, we decided to only leave in variance features, and eliminate mean, median, minimum and maximum, since we found that variance values were more consistent among multiple people than the other aggregates. Our accuracy when testing with 10-fold cross-validation dropped to ~91% on removing the other aggregates, but our graphs looked better for other non-V test subjects' data. Fig. 7 compares the results of testing models trained on all features vs. on variance-only features with test subject J's data. We can see that the model with variancy-only features produces more stable results. Additionally, now our predicted sleep phases aligned better with variance in acceleration in our graphs, which we believe is the best measure for amount of movement.

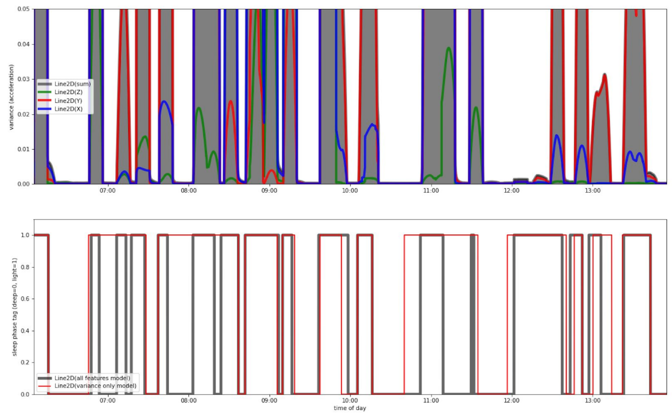
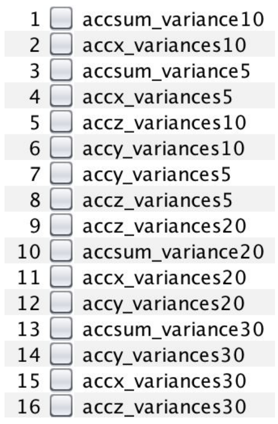


Fig. 7: Subject J data for May 7th. Notice the significantly more stable classifications with the variance-only as features model.

As discussed in the previous section, we also added features that aggregated variance along the three axes, and removed 1 minute windows. The 1 minute intervals were contributing to the short spikes in our graphs and, since sleep phases are traditionally not shorter than about 5 minutes, we wanted to try and get rid of these spikes. Additionally, aggregates over 1 minute intervals are not always predictive of sleep phase, because one is not usually moving every minute when in light sleep, but rather every few minutes.

With this final set of 16 features (3 sensor streams + 1 sum of the sensor streams, and 4 interval lengths), we trained on 25 nights and almost 10,000 minutes of data. Our final model accuracy was 91.31%.



1	accsum_variance10
2	accx_variances10
3	accsum_variance5
4	accx_variances5
5	accz_variances10
6	accy_variances10
7	accy_variances5
8	accz_variances5
9	accz_variances20
10	accsum_variance20
11	accx_variances20
12	accy_variances20
13	accsum_variance30
14	accy_variances30
15	accx_variances30
16	accz_variances30

Fig. 8: Our final features ranked from most information gained to least. This lines up with our traditional analysis findings

Fig. 8 shows our features ranked according to information gain. We can see that the sum of acceleration along the three axes over 10 minute intervals is the most predictive feature. The results based on our ML model are described in section III.

Majority Filtering

When qualitatively analyzing our output, we realized our model is prone to classifying 1 minute of light sleep in a long period of deep sleep or vice versa. This makes sense as there could be a slight amount of movement in that period, but this does not mean you have 1 minute of light sleep.

To combat this, we post-processed our predicted output with a majority filter over a window of 7 minutes. This means that if there is only 1-2 minutes of differently classified data, it will be filtered out. Figure 9 shows the difference before and after filtering.

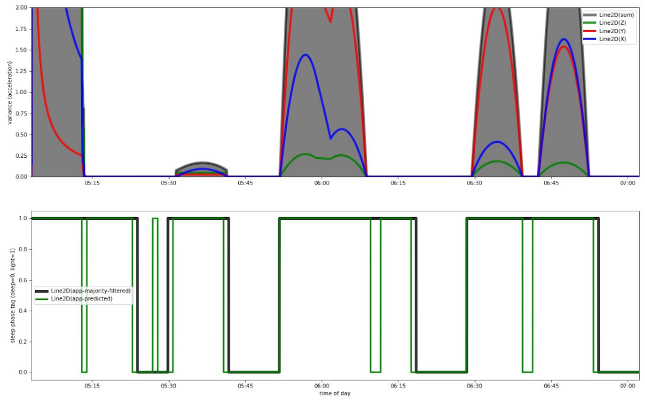


Fig. 9: Graph of predicted sleep phases with majority filtering versus raw.

III. RESULTS

Final Product

We modified our app so that we could use our generated Weka model to classify data in real time in the application. We added data structures to maintain windows of acceleration data samples (5, 10, 20, 30 minutes worth) as they were collected, and every minute we spawned an async thread to calculate the features and classify the data. We kept a window of 7 minutes and used the majority over the window to get rid of noise in our phases. Using this we were able to implement some of our features.

Falling asleep: We detected falling asleep by waiting until the majority of the labeled phases in our window were deep sleep. Essentially when we determined that the user had entered a deep phase of sleep using our model, we marked them as fallen asleep. We integrated a Yeelight java API[3] so that we could control our smart light bulb from the app and using this api we signalled the bulb to turn off when the user was marked as "fallen asleep"

Smart alarm: Although we were unable to add a fledged alarm UI unto our app, we were able to test the features viability by hard coding a test into the system. In the application, once we were within 30 minutes of the set alarm time, we would actively monitor the current sleep phase. Once the user was in light sleep we triggered the alarm. If they remained in deep sleep for the entire 30 minutes we forced an alarm at the pre-set time. In this manor we were able to implement our desired features.

Final testing

We tested our final model and processing on two nights' data (5/11 and 5/12) with 4 test subjects (J, M, U, and V). The graphs for 5/11 can be seen in Appendix A.

For 5/12 we also tested our app features. Below are the times marked by our application as when the user fell asleep and when they were woken up by the smart alarm.

Falling asleep time:

Subject	**Time marked as asleep**
V	06:02:57 2020-05-12
U	05:59:20 2020-05-12
M	Invalid data
J	05:26:51 2020-05-12

Smart alarm wake up time for alarm set at 10:00 AM

Subject	**Time marked as asleep**
V	09:30:16 2020-05-12
U	09:30:00 2020-05-12
M	09:30:35 2020-05-12
J	09:38:12 2020-05-12

These results were verified against the phase labels generated by our model.

Results

Once we loaded the model onto the app we noticed a difference in the predictions made by the app and that made by the independently tested Weka model as shown. We believe these to be a result of keeping windows of values in real time and running the classification in threads, or minor differences in how the variances are being calculated. The difference can be seen in Fig. 10.

[3] https://github.com/florian-mollin/yapi

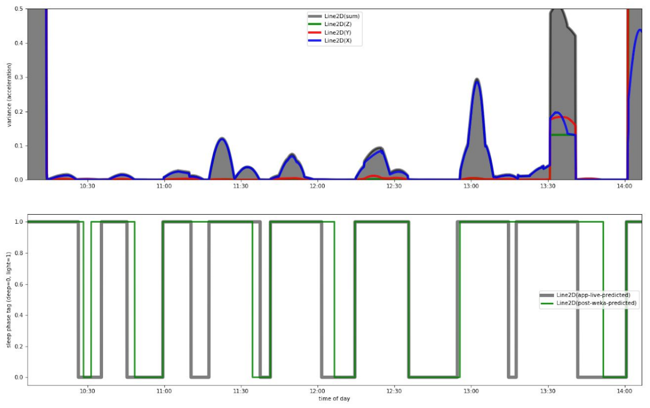


Fig. 10: Graph of sleep phases classified in real time by the app versus that classified in post by our Weka model. There is a minor difference in labels.

When we tested our final model on two night's sleep data of four test subjects, the resulting accuracies ranged from 65%-75%. We are unsure as to why these accuracies were so low, considering our final model's (consisting of 25 nights of data) accuracy is 91% tested using 10-fold cross-validation. However, as can be seen in figs. A.1-A.4 in the appendix, our qualitative results look quite good. Our sleep phase predictions match up well with variance in acceleration, which we believe to be the best indication of amount of movement.

As can be seen in our results, test subjects J and U display a lot more movement in light sleep than test subjects V and M. Our ML model did really well at determining common thresholds for the test subjects despite the varying amounts of movement detected for each of them. This might have been harder to accomplish using a traditional signal analysis approach.

Our predicted sleep phases still have more spikes than the Sleep as Android tags; Sleep as Android classifies longer chunks of time as light/REM sleep than we do. We hypothesize that this is because Sleep as Android considers what a traditional sleep cycle should look like (alternating light and deep sleep phases, with REM sleep in the middle of long light sleep phases) in addition to simply analyzing movement data when making sleep phase decisions. Our model was able to extract the relation between movement and the Sleep as Android tags, but may not have been able to extract information about how a sleep cycle should look. However we wanted to err on the side of classifying sleep as deep rather than light, since waking someone up in deep sleep because it was classified as light sleep is a mistake that would defeat one of the main purposes of our product - to help users wake up feeling refreshed. Thus we are okay with our app predicting shorter light sleep phases than Sleep as Android, since we are not concerned with approximating sleep cycles for our applications.

Additionally, a major problem we realized with our training process was the assumption that REM sleep is similar to light sleep. Upon analyzing our variance graphs against SaA-labeled data more, we are convinced that REM sleep actually usually looks more like deep sleep than light sleep. SaA inserts REM in between periods of light sleep based on knowledge of sleep cycles and not any movement based data. Figure 11 shows how our model can classify deep sleep during REM during this. Luckily, because waking a subject in REM sleep ends up with them groggy, this is not a terrible outcome.

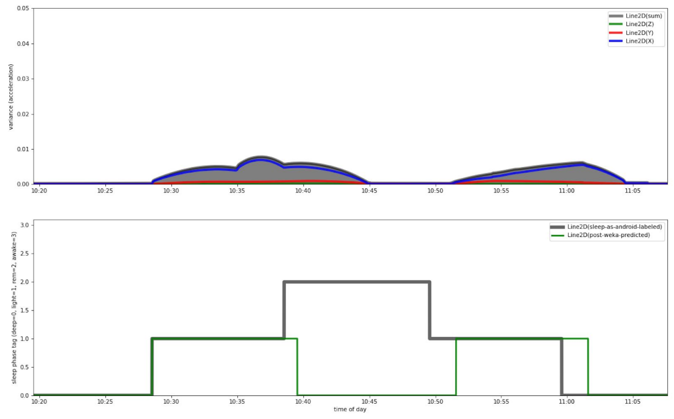


Fig. 11: Test subject V data for 5-11. Notice the deep classification from Weka model during Sleep as Android's REM classification

IV. DIVISION OF LABOR

Vidhart Bhatia: Modified open-source application to serve our needs, handled logging and collection of data and worked on model integration. Slept a lot.

Medha Potluri: Worked on Weka classification model and wrote a tagging script to help generate a feature dataset.

Udit Ranasaria: Wrote graphing scripts and worked on a traditional signal analysis approach.

All of us collaborated on integration, analysis and iteration of the model.

V. FUTURE WORK

App

We would like to add more features into the application::

- Adding a proper alarm UI and extending the smart alarm feature.
- Linking to smart home devices to possibly adjust humidity and temperature in order to improve sleep conditions.[4]
- Displaying more information to the user such as sleep graphs or insights.

Analysis

There are many realizations we had about improvements to our dataset we could have made that could have had marked improvements on our model but we did not have time to go back and retrain with. The first of these is not using gravity-normalized acceleration. The acceleration we have been using has gravity built in which means that orientation of the phone makes a big impact on the raw acceleration values. We should have subtracted away the gravity sensor values from the acceleration values and used those. We would also remove REM and Awake tagged data from our training dataset to prevent the misclassification that we saw in our results.

Addressing the fact that our training and testing quantitative results were so different could also positively impact our project. It is possible that our Random Forest overfit our dataset by branching too much for specific cases. Adding in depth limits might generalize our system more at the expense of testing accuracy.

Another general issue with our approach to this problem was not having a strong labeling or "truth" oracle. Both using variances to qualitatively compare and Sleep as Android to quantitatively determine accuracy have in-built assumptions on what we want our output to look like. We should have tried to train on available PSG sleep datasets[4]. This would have allowed us to get a more scientific, unbiased labeled dataset.

Similarly, we realized that without a more accurate source of truth, it will be impossible to validate a system of 2 people sharing a bed. Since the whole goal would be to detect sleep individually despite interference in motion data, we would have to use camera or individual biometric data as a truth for 2-person sleep classification.

IX. REFERENCES

[1] WILDE-FRENZ, JOHANNA & Schulz, Hartmut. (1983). *Rate and distribution of body movements during sleep in Humans*. Perceptual and motor skills. 56. 275-83. 10.2466/pms.1983.56.1.275.

[2] S. Chen, R. Perera, M. M. Engelhard, J. R. Lunsford-Avery, S. H. Kollins and B. F. Fuemmeler, *"A Generic Algorithm for Sleep-Wake Cycle Detection using Unlabeled Actigraphy Data,"* 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), Chicago, IL, USA, 2019, pp. 1-4, doi: 10.1109/BHI.2019.8834568.

[3] Girardin Jean-Louis, Daniel F Kripke, Roger J Cole, Joseph D Assmus, Robert D Langer, *Sleep detection with an accelerometer actigraph: comparisons with polysomnography,* Physiology & Behavior, Volume 72, Issues 1–2, 2001, Pages 21-28, ISSN 0031-9384, https://doi.org/10.1016/S0031-9384(00)00355-3.

[4] Matthew Kay, Eun Kyoung Choe, Jesse Shepherd, Benjamin Greenstein, Nathaniel Watson, Sunny. Consolvo, and Julie A. Kientz. 2012. *Lullaby: a capture & access system for understanding the sleep environment.* In Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12). Association for Computing Machinery, New York, NY, USA, 226–234. DOI:https://doi.org/10.1145/2370216.237025

[4] https://sleepdata.org/

X. APPENDIX A

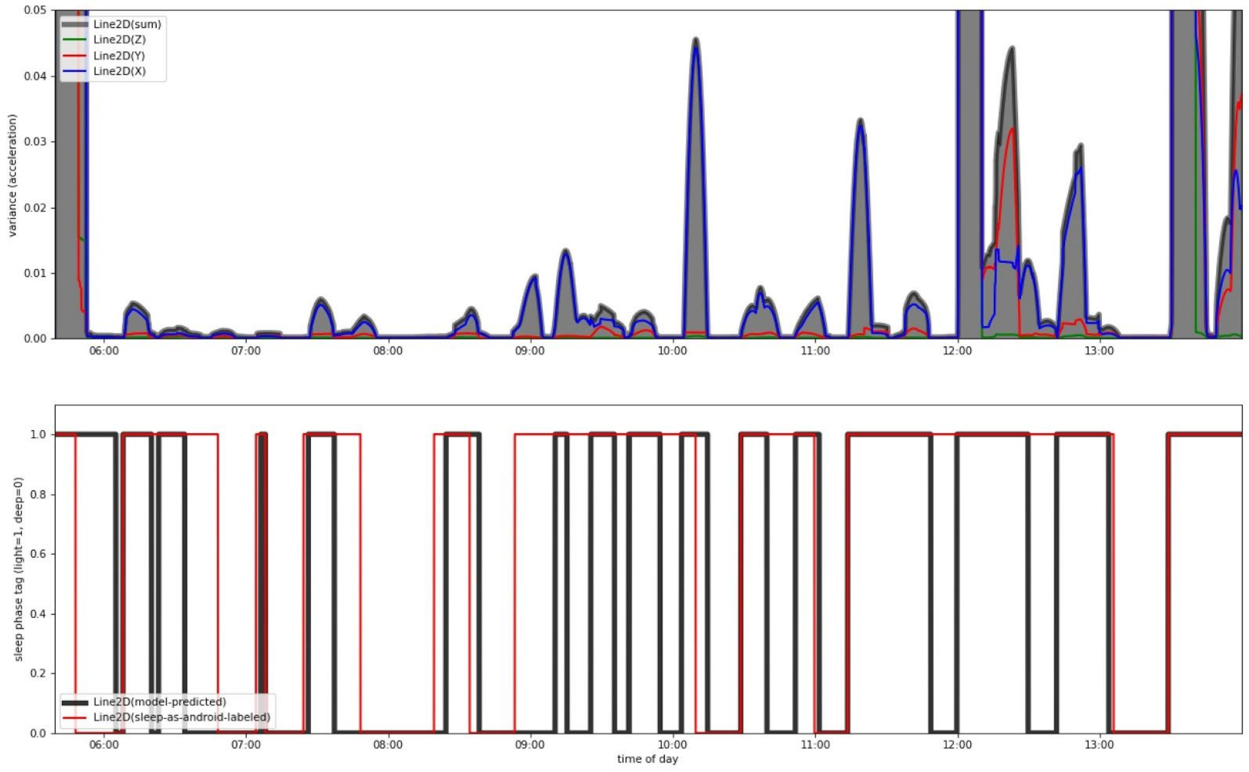


Fig. A.1: Subject V 5/11 sleep phase graph with Sleep as Android labels and our applications predicted labels with movement variance.

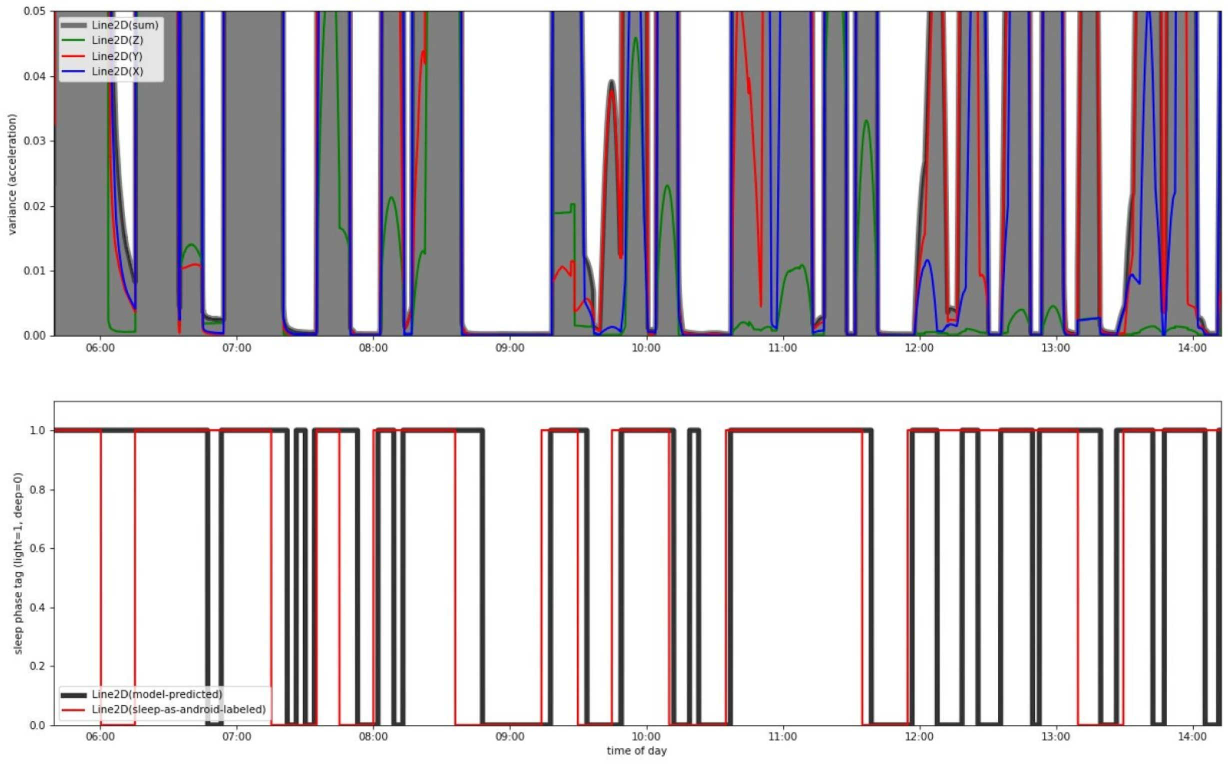


Fig. A.3: Subject U 5/11 sleep phase graph with Sleep as Android labels and our applications predicted labels with movement variance.

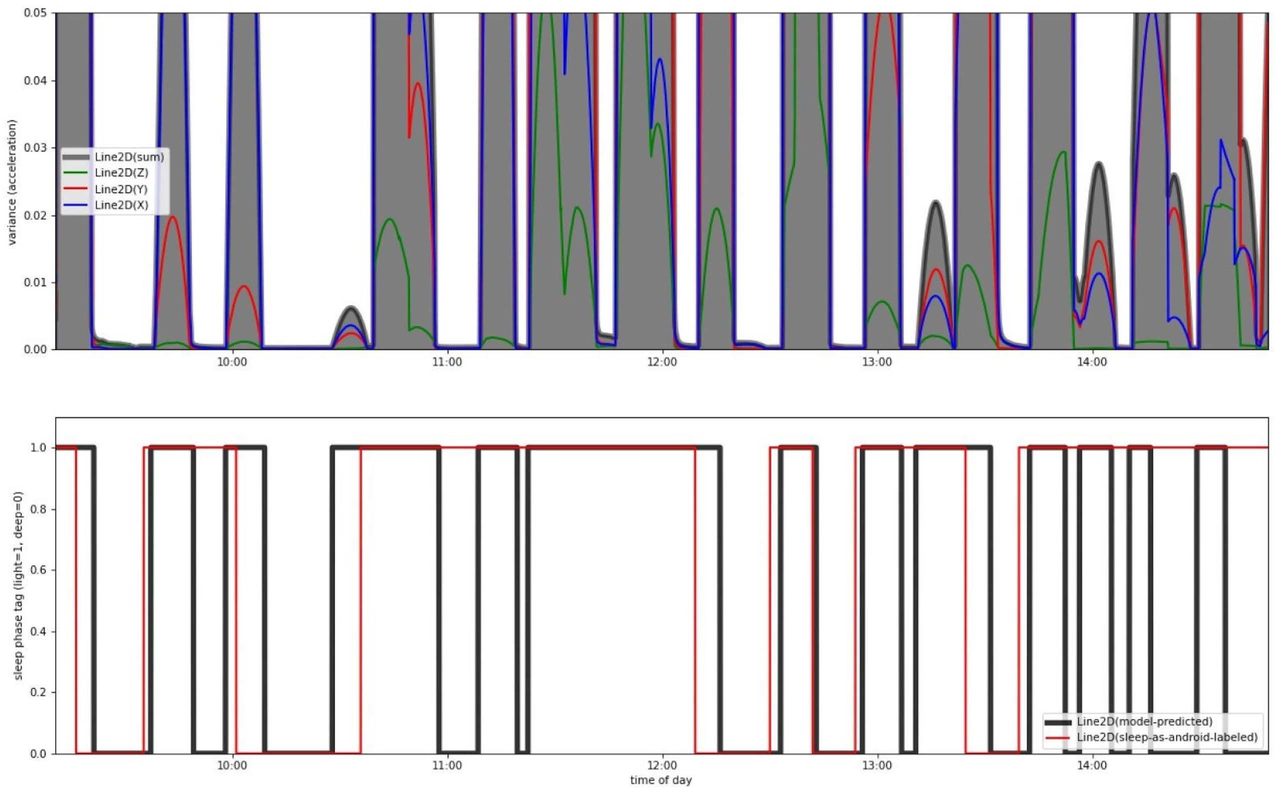


Fig. A.4: Subject J 5/12 sleep phase graph with Sleep as Android labels and our applications predicted labels with movement variance.

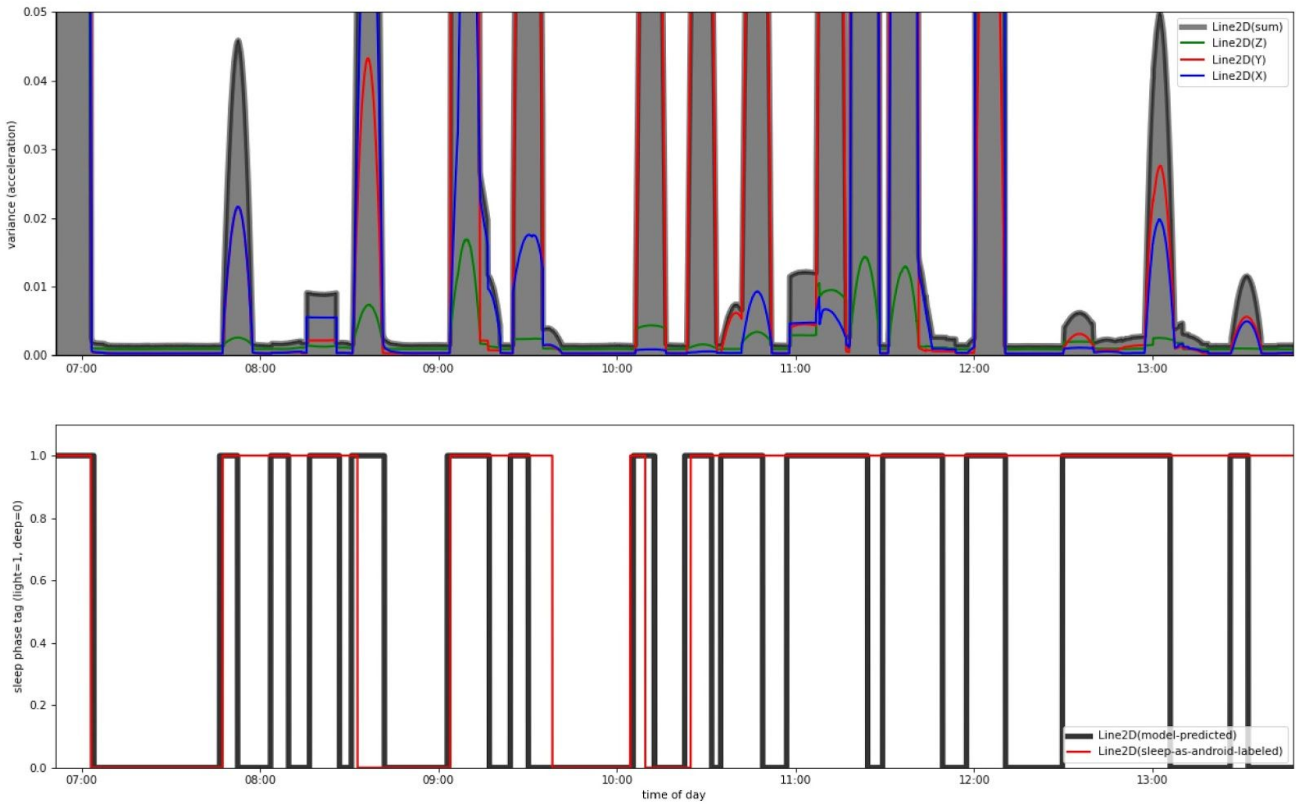


Fig. A.2: Subject M 5/11 sleep phase graph with Sleep as Android labels and our applications predicted labels with movement variance. After 11:00AM, Sleep as Android was turned off but our app was still recording, so the Sleep as Android tags can be ignored.