Child Mind Institute
Detect Sleep States

Diya Saha
sahad1@uci.edu
Department of Computer Science
University of California, Irvine

Abhinand Ganesh
aganesh5@uci.edu
Department of Computer Science
University of California, Irvine

1 Introduction and Problem Statement

This project employs AI-based techniques to predict sleep onset and wakeup events using data produced by a wrist-worn accelerometer. The dataset consists of multi-day recordings with annotated sleep events: onset, the beginning of sleep, and wakeup, the end of sleep. This is a sequence labeling/time series event detection problem that will use time series analysis algorithms, with evaluation using Accuracy of the predicted event points. We test out experiments with randomly sampled 30 and 90 subjects, and split them using the 80-20 criterion for training and validation sets. Feature engineering was also carried out to expand the 2-dimensional input set into a 7-dimensional set using rolling standard deviation and differences between two consecutive accelerometer data. We used deep learning classification architectures, from MLP (Multi-Layer Perceptron) models and LSTM (Long Short Term Memory) networks - owing to the latter's ability to capture long term temporal dependencies in the accelerometer data. We get a maximum classification accuracy of 91.15% using the LSTM network on the training set that contains data from 90 randomly sampled subjects.

2 Related Work

This is mainly a time-series change point detection problem where the machine learning model is capable of predicting abrupt changes in the accelerometer data. This has been studied heavily in papers like (3) and (4). The input contains a training set, where the accelerometer data for different series and different nights are annotated mainly using two labels - 'onset' and 'wakeup', where the former denotes sleep onset and the latter denotes wakefulness. Changepoint detection models capture temporal dependencies in the data. As observed historically, the best models are capable of detecting complex temporal patterns and dependencies and LSTMs (Long-Short-Term-Networks). The final output of the model would be able to predict sleep events/changepoints (i.e, onset and wakeup) on newly presented wrist accelerometer data that the model hasn't been exposed to before. Prediction of sleep states from wrist-worn accelerometer data has been attempted before in studies like (1), where there has been emphasis on subjects with disturbed sleep, presenting unique challenges. Our project focuses on predicting sleep states for individuals who are children.

Usage of wrist-worn devices to predict sleep states is also seen in studies like (2), where there has been usage of traditional machine learning classifiers. However, there is a lack of deep architectures in their study. We will be overcoming that challenge by using deep learning models that will be capable of capturing long term temporal dependencies in the data. In our study we are focusing on two events the onset and the wakeup so we would not have to keep track of multiple different features and conduct multiple feature selection. The inputs to our system will be multi-day recordings from a wrist-worn accelerometer, labeled with sleep events such as onset and wakeup. The desired output is an accurate prediction of these sleep events, providing insights into sleep patterns. Previous

approaches in similar domains have. We plan to leverage these techniques, possibly exploring additional models, to enhance the accuracy of sleep event predictions in our dataset.

3 Data

We were provided two input files: the **train_series.parquet** and **train_events.csv**.

3.1 train_series.parquet

The parquet file contains 4 columns: the **series_id** that contains the unique ID of the patients, the **Step** numbers are used as labels by the parquet file to label that particular timestamp. The **timestamp** is to indicate when the measurement was taken by the accelerometer data. The **anglez** is taken from the accelerometer, and it stands for the movement on the z-axis. The **enmo** is also another measurement taken from the accelerometer, and it stands for the magnitude of motion.

	series_id	step	timestamp	anglez	enmo
0	038441c925bb	0	2018-08-14T15:30:00-0400	2.636700	0.0217
1	038441c925bb	1	2018-08-14T15:30:05-0400	2.636800	0.0215
2	038441c925bb	2	2018-08-14T15:30:10-0400	2.637000	0.0216
3	038441c925bb	3	2018-08-14T15:30:15-0400	2.636800	0.0213
4	038441c925bb	4	2018-08-14T15:30:20-0400	2.636800	0.0215
...	...	...	...	...	...
127946335	fe90110788d2	592375	2017-09-08T00:14:35-0400	-27.277500	0.0204
127946336	fe90110788d2	592376	2017-09-08T00:14:40-0400	-27.032499	0.0233
127946337	fe90110788d2	592377	2017-09-08T00:14:45-0400	-26.841200	0.0202
127946338	fe90110788d2	592378	2017-09-08T00:14:50-0400	-26.723900	0.0199
127946339	fe90110788d2	592379	2017-09-08T00:14:55-0400	-31.521601	0.0205

127946340 rows × 5 columns

Figure 1: Dataframe imported from the train_series.parquet

3.2 train_events.csv

This csv contains the columns **series_id**; the **unique_id** for each patient, **night**; the night for which the event is happening, **event**; the label for that particular **step number**, step; corresponding to the step in the parquet file and **timestamp**; the time stamp at which the measurement was taken.

3.3 Combined input files for processing

This csv contains the columns **series_id**; the **unique_id** for each patient, **night**; the night for which the event is happening, **event**; the label for that particular **step number**, step; corresponding to the step in the parquet file and **timestamp**; the time stamp at which the measurement was taken.

3.4 Randomly sampled patients

After printing out the unique patients from the merged file we see that the input file provided us with 277 unique patients which means that there were 277 unique patients.

Since there were about 277 patients, we were not able to process all of them. Therefore, we decided to split the dataset into two smaller sets: one with 30 randomly sampled patients and another with 90 randomly sampled patients. The main purpose of this was to make the 30-patient dataset process faster, allowing for easier testing of different model architectures, while the larger dataset would be used to test our hypothesis.

	series_id	night	event	step	timestamp
0	038441c925bb	1	onset	4992.0	2018-08-14T22:26:00-0400
1	038441c925bb	1	wakeup	10932.0	2018-08-15T06:41:00-0400
2	038441c925bb	2	onset	20244.0	2018-08-15T19:37:00-0400
3	038441c925bb	2	wakeup	27492.0	2018-08-16T05:41:00-0400
4	038441c925bb	3	onset	39996.0	2018-08-16T23:03:00-0400
...	...	...	...	...	...
14503	fe90110788d2	33	wakeup	560604.0	2017-09-06T04:07:00-0400
14504	fe90110788d2	34	onset	574620.0	2017-09-06T23:35:00-0400
14505	fe90110788d2	34	wakeup	581604.0	2017-09-07T09:17:00-0400
14506	fe90110788d2	35	onset	NaN	NaN
14507	fe90110788d2	35	wakeup	NaN	NaN

14508 rows × 5 columns

Figure 2: Dataframe imported from the train_events.csv

	series_id	step	anglez	enmo	y_target
0	1b92be89db4c	0	87.1277	0.0033	0
1	1b92be89db4c	1	88.1777	0.0091	0
2	1b92be89db4c	2	88.2099	0.0030	0
3	1b92be89db4c	3	88.1331	0.0017	0
4	1b92be89db4c	4	87.9545	0.0015	0
...	...	...	...	...	...
40651915	f2c2436cf7b7	519655	-12.3163	0.0767	0
40651916	f2c2436cf7b7	519656	-13.9673	0.0540	0
40651917	f2c2436cf7b7	519657	-11.5024	0.0599	0
40651918	f2c2436cf7b7	519658	-17.8390	0.0682	0
40651919	f2c2436cf7b7	519659	-18.9898	0.0922	0

40651920 rows × 5 columns

Figure 3: Merging the input files and creating a formatted csv for processing

3.5 Description of Technical Approach

The challenge involves two primary data sources: accelerometer data observed through features **anglez** and **enmo**, and a dataset containing annotations for wakeup and onset timestamps denoted by **step**. The initial step is to join these data sources based on steps and series_ids, filling in missing values for the target variable. Labels **1** and **0** are used, where **1** indicates the subject is sleeping, and **0** indicates the subject is awake. We fill **1** between each onset, wakeup pair and **0** for timestamps until the next pair appears. The logic assumes a sleep onset when the target variable switches from 0 to 1 and wakefulness for the latter.

Feature engineering techniques are then applied to generate features from **anglez** and **enmo** values grouped by **series_id**. The engineered features include **hour** (from timestamp), **anglez_diff** (from anglez), **anglez_rolling_std** (from anglez), **enmo_diff** (from enmo), and **enmo_rolling_std** (from enmo). This transforms the input into a 7-dimensional feature matrix. Three neural network architectures are employed: a basic MLP (Multi-Layer Perceptron) model, an LSTM (Long Short-Term Memory) network, and an improved LSTM model with adjusted hyperparameters. These

```
series_id
78569a801a38    1433880
f564985ab692    1052820
fb223ed2278c     918360
f56824b503a0     846360
cfeb11428dd7     809820
                 ...
c535634d7dcd     136080
1c7c0bad1263     115380
60e51cad2ffb     113940
3a9a9dc2cbd9     103500
349c5562ee2c      37080
Name: count, Length: 277, dtype: int64
```

Figure 4: Unique patient IDs

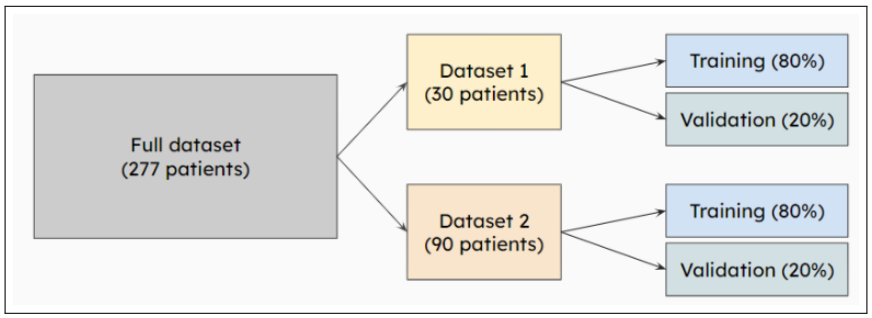


Figure 5: Splitting the data for easier processing

models are applied to two input sets, randomly sampling 30 and 90 subjects. The former contains 14,276,340 timestamps, and the latter contains 40,651,920. The Adam Optimizer with a learning rate of 0.001 is used for optimization, and binary cross-entropy serves as the loss function. The loss function is calculated using the following formula:

$$BinaryCrossEntropy = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{i,j}\log(p_{i,j}) \tag{1}$$

where N stands for the number of timestamps and M stands for the number of class labels - which is 2 in our case because of the binary nature of the target variable. Once all the models were trained on both the input feature matrices (each of which contained 80% of the set and the validation set containing the remaining 20%). The evaluation is carried out using Accuracy as the metric, which is calculated using the formula -

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

Here, **TP** stands for True Positive, **TN** stands for True Negative, **FP** stands for False Positive and **FN** stands for False Negative.

4 Software

4.1 Zipper Function

The zipper function, thoroughly explained in the datasets subsection, was employed to merge the two input files and perform general dataset cleanup.

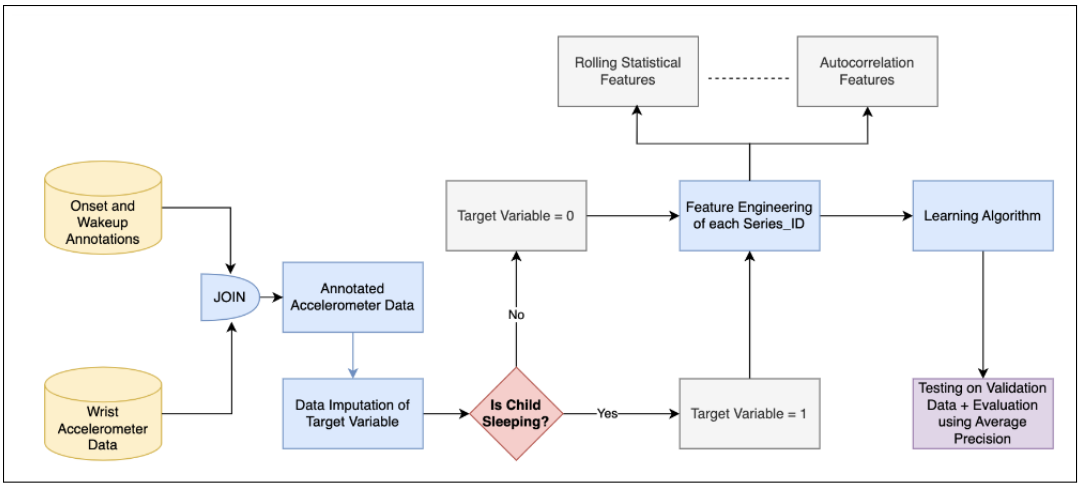


Figure 6: Technical Architecture

4.2 Random Patient Sampler

The random patient sampler, essential due to the dataset's size, divided it into smaller chunks for efficient processing and diverse testing. This code randomly selects **n** patients from a list, executes the zipper function, and generates a .csv file with necessary columns. Importantly, the function ensures distinct patient subsets for each chunk, eliminating redundancy with previously selected subsets.

4.3 Feature Engineering

We wrote our own scripts to expand the dimensionality of the set from (n,2) to (n,7). For the purpose of this, we took advantage of the pandas.DataFrame.groupby() function offered by Python. To calcualte the rolling standard deviations of the anglez and enmo features, we use a window size of 3.

4.4 MLP

To understand our data better and perform a baseline performance test, we utilized a simple Multi-Layer Perceptron with the Adam optimizer, binary cross-entropy loss, and a batch size of 32.

4.5 LSTM

Given the temporal nature of our data, we chose an LSTM model. We started with the basic LSTM model, conducted hyperparameter tuning, and added more layers to improve overall performance. Later, we expanded this model with feature-engineered data to assess the performance of our theory.

5 Experiments and Evaluation

5.1 MLP

We initiated the experimentation with an MLP for a baseline performance test to gauge the model's initial performance on our 30-patient dataset, which showed a performance around the 82% mark. However, we observed a consistent imbalance issue as the validation accuracy was higher than the training accuracy, stemming from the imbalanced sleep and wake durations in the provided data. To address this, we opted for a larger dataset of 90 randomly sampled patients, aiming for a more accurate representation of the complete dataset. While the imbalance issue persisted, the 90-patient dataset was considered closer to the complete dataset.

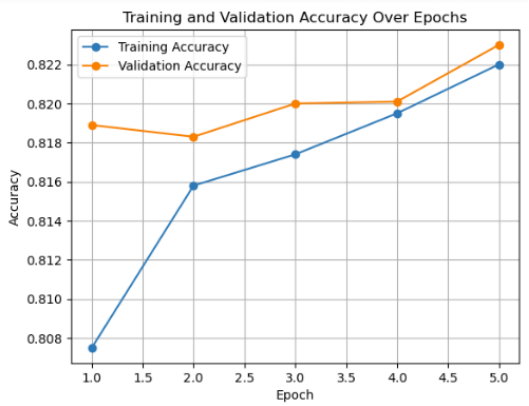
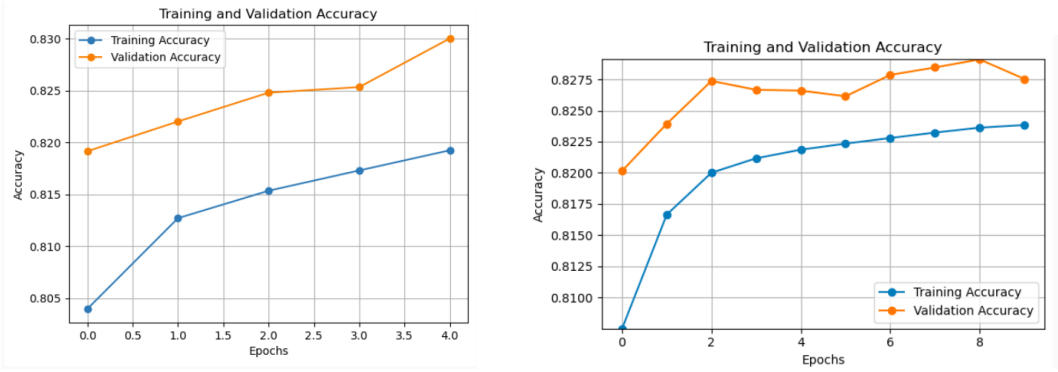


Figure 7: Running the basic MLP model on the 30 randomly sampled patients

5.2 Basic LSTM

Our hypothesis was that selecting a model that performed well on the data would lead to the 90-patient dataset's validation accuracy still being higher than the training accuracy but with a reduced difference. After implementing the basic LSTM model, we observed an improvement in performance, aligning with our theory as the gap between training and validation accuracy narrowed. Subsequently, we decided to enhance the LSTM model further, beginning with hyper-parameter tuning to optimize its performance.



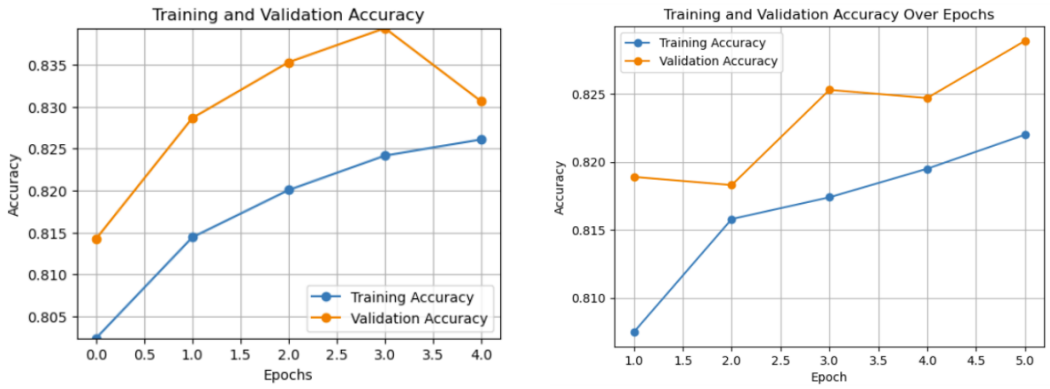
(a) 30 randomly sampled patients

(b) 90 randomly sampled patients

Figure 8: Performance of a basic LSTM Architecture

5.3 Improved LSTM

After conducting the initial experiment, our focus shifted to improving the LSTM model. The most straightforward approach was to perform hyperparameter tuning. Initially, our model had a batch size of 32, a learning rate of 0.001, the Adam optimizer, and ran for 10 epochs with a single layer. To enhance performance, we experimented with different optimizer functions and tested various learning rates. Our observations indicated that the Adam optimizer yielded the best results for our model, and we opted for a learning rate of 0.01 to optimize processing time. We also increased the batch size, resulting in a slight improvement in accuracy. Additionally, recognizing the potential benefits of increasing model complexity, we decided to add more hidden layers. This adjustment aimed to allow the model to better capture the intricacies of the data, potentially leading to quicker training and overfitting on the provided data.

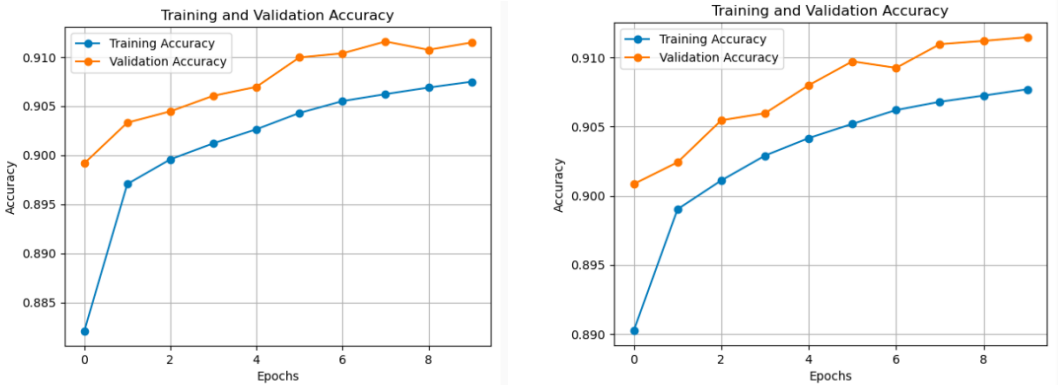


(a) 30 randomly sampled patients (b) 90 randomly sampled patients

Figure 9: Performance of the Improved LSTM Architecture

5.4 Imporved LSTM with additional feature engineering

Initially, we assumed that we were provided with only two features and aimed to explore the possibility of adding more features to enhance our model's understanding of the data. As a result, we extracted an hour feature from the provided timestamp feature. Additionally, we computed the rolling standard deviation of the given features, namely anglez and enmo. Given the temporal nature of our data, we opted to create difference features for both provided features, resulting in enmo_diff and anglez_diff. In summary, starting with a dataset featuring only 2 input features, we constructed an input vector of 7, including anglez, enmo, hour, anglez_rolling_std, enmo_rolling_std, anglez_diff, and enmo_diff.



(a) 30 randomly sampled patients (b) 90 randomly sampled patients

Figure 10: Performance of the Improved LSTM Architecture with feature engineering

6 Future Improvements

The main challenge encountered during this project was the inability to process the entire provided dataset in its entirety. As a workaround, we divided the data into smaller datasets and treated the complete dataset as if it consisted of only 90 patients. Looking ahead, we aspire to train our model on the entirety of the dataset, encompassing all 277 unique patients. During the model training phase, we identified a significant imbalance, and the reduced dataset accentuated this imbalance in our model. In the future, we plan to conduct error analysis to understand specific challenges, such as false positives and false negatives, and focus on improving those aspects of our model. To assess the performance of our model, we initially split the data into 80% training and 20% validation. Going forward, we aim to implement cross-validation to provide a more robust evaluation of our model's performance.

For model improvement, we intend to perform a grid search on all possible parameters of our LSTM model, ensuring that we utilize the best hyper parameters available. Recognizing the significant contribution of feature engineering to our model's performance, we plan to leverage generative models to engineer better and more informative features that can enhance our model's understanding of the data. We could possibly use models like Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs) to generate deep featuees that will aid better for classification of sleep states.

7 Individual Contributions

7.1 Abhinand

I worked on performing data imputation on the target variable to fill up missing values (i.e, 0 if a person is awake and 1 if the person is sleeping). I also randomly sampled 90 patients from the entire dataset and ran the Multilayer Perceptron (MLP) and Long Short-Term Memory (LSTM) networks on it and plotted the accuracy plots. I also coded and ran the feature engineering code for expansion of the dimensionality of the input set from (n,2) to (n,7) - after which, I ran the improved LSTM code over it to get the accuracy plot and confusion matrix. I also played a crucial role in contributing to the project report and conducting a literature survey.

7.2 Diya

In the beginning, I played a crucial role in data processing by assisting in the cleanup of the provided dataset. I contributed to writing code to automate repetitive tasks, such as handling missing values, dropping unnecessary columns, and randomly selecting a subset of patients (n) for training the model. Subsequently, I took charge of executing various models on a smaller dataset, consisting of a randomly sampled subset of 30 patients. This decision was driven by hardware limitations, as processing the larger dataset would have been significantly more time-consuming. This approach allowed us to efficiently explore a range of models and conduct hyperparameter tuning to establish baseline tests on our data. For example, we experimented with different configurations, such as changing the optimizer function to SGD and adjusting the learning rate to 0.1. If we observed that the performance of a modified model was inferior to our original model on the smaller dataset, we discarded that configuration and tried another combination. This iterative process ensured that our efforts were focused on configurations likely to perform well when applied to the larger dataset.

References

[1] Cakmak, A. S., Da Poian, G., Willats, A., Haffar, A., Abdulbaki, R., Ko, Y.-A., Shah, A. J., Vaccarino, V., Bliwise, D. L., Rozell, C., & Clifford, G. D. (2020, August 12). An unbiased, efficient sleep-wake detection algorithm for a population with sleep disorders: Change point decoder. *Sleep*. Retrieved from `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7420526/`

[2] Liang, Zilu Martell, Mario. (2019). Predicting Medical-Grade Sleep-Wake Classification from Fitbit Data Using Tree-Based Machine Learning. 2019-UBI-62. 1-8.

[3] Stuburić, Klara, Maksym Gaiduk, and Ralf Seepold. "A deep learning approach to detect sleep stages." Procedia Computer Science, vol. 176, 2020, pp. 2764-2772, `https://doi.org/10.1016/j.procs.2020.09.280.`

[4] Yoo, Young-Keun, Chae-Won Jung, and Hyun-Chool Shin. 2023. "Unsupervised Detection of Multiple Sleep Stages Using a Single FMCW Radar" Applied Sciences 13, no. 7: 4468. `https://doi.org/10.3390/app13074468`