Brown University Division of Applied Mathematics

GENERATIVE APPROACHES TO THE INFERENCE OF SLEEP PATTERNS FROM PHYSICAL ACTIVITY

January 13, 2022
Submission for Problem 1 of BMCM, 2021.

Dichuan Gao
Shreyas Rao
Alexander Young

Contents

1 Introduction

1.1 The Sleep Detection Problem

Brown University is developing a new application named "BrownWell", which helps students track their health habits. In particular, we are interested in helping students track their sleep time and sleep patterns.

BrownWell takes inputs from various pre-existing sensors, including those built into students' phones, smartwatches, fitbits, and so on. These sensors count the number of steps a student takes within every 15-minute period, starting at midnight each day, and updating every 15 minutes.

Our task is the following: given a time-series of a student's step-counts measured at 15-minute intervals, infer when the student is awake, and when they are asleep. This will allow BrownWell to track the sleep patterns of a student, thereby providing crucial health data that can otherwise only be obtained via intrusive or highly unreliable methods (for example, by polysomnography, or by voluntary sleep diary).

We are given a "labelled data set" consisting of a month's worth of step-count data for 100 students, together with (possibly faulty) labels indicating whether the student is sleeping or awake at each 15 minute interval. We are also given an "unlabelled data set" consisting of a month's worth of step-count data for 400 students, with no corresponding labels. Using these data sets, we implement two models for sleep-pattern inference.

We then proceed to draw some conclusions on the general sleep patterns of the Brown student body, based on the inferences made by our models using the unlabelled data set.

1.2 Review of Classical Approaches

The most commonly used model today for detecting sleep at a low cost from physical activity is the Cole-Kripke Algorithm [2]. Instead of step-counts, this algorithm requires the acceleration measured by an accelerometer worn on the subject's wrist (which provides substantially more information than the data we have). Let A_t be the measured acceleration of the subject's wrist at time t. The model consists of parameters $P, W_{t-4}, ..., W_{t+2}$, and computes

$$D_t = P(W_{t-4}A_{t-4} + ... + W_{t+2}A_{t+2})$$

where D_t indicates the likelihood that the subject is awake - usually the higher D_t is, the more likely the subject is awake at time t.

A boundary value D^* is then fixed (usually $D^* = 1$), such that the algorithm predicts that the subject is awake at time t if $D_t > D^*$:

$$Y_t = \begin{cases} 1 & D_t > D^* \\ 0 & D_t \leq D^* \end{cases}$$

Finally, several hand-crafted rescoring rules are applied to the series Y_t, to correct the algorithm's tendency to misscore actual wakefulness as sleep, which occurs in part because subjects falling asleep stop moving a few minutes before actual sleep occurs:

1. If $Y_{t-5} = \cdots = Y_{t-1} = 1$, then Y_t is manually rescored to 1 regardless of its predicted value;
2. If $Y_{t-11} = \cdots = Y_{t-1} = 1$, then $Y_t, \ldots, Y_{t+2}$ are all rescored to 1 regardless of its predicted value;
3. If $Y_{t-16} = \cdots = Y_{t-1} = 1$, then $Y_t, \ldots, Y_{t+3}$ are all rescored to 1;
4. Any period of 6 minutes or less predicted as sleep, surrounded by at least 10 minutes before and after predicted as wake, are rescored as wake. That is, if $Y_{t-11} = \cdots = Y_{t-1} = 1$, and $Y_t = \cdots = Y_{t+\tau-1} = 0$, and $Y_{t+\tau} = \cdots = Y_{t+\tau+9} = 1$, for some $\tau \leq 6$, then $Y_t, \ldots, Y_{t+\tau}$ are rescored to 1;
5. Any period of 10 minutes or less predicted as sleep, surrounded by at least 20 minutes before and after predicted as wake, are rescored as wake. That is, if $Y_{t-21} = \cdots = Y_{t-1} = 1$, and $Y_t = \cdots = Y_{t+\tau-1} = 0$, and $Y_{t+\tau} = \cdots = Y_{t+\tau+19} = 1$, for some $\tau \leq 10$, then $Y_t, \ldots, Y_{t+\tau}$ are rescored to 1.

1.3 Our Approach

The rest of this paper is organized as follows: we begin by introducing the models we implemented (section 2); then proceed to evaluate the results from our successful models (section 3), and discuss the implications of these results in section 4. Finally, we conclude with a brief summary in section 5.

Overall, our models followed three iterations. First, inspired by the Cole-Kripke approach, we implemented a simple logistic regression model as a baseline. We then extended the logistic regression model to a Convolutional Neural Network, achieving a much higher accuracy. Finally, in order to leverage the full set of data provided, we develop an unsupervised Hidden Markov Model.

In our results section, we evaluate our CNN-based and HMM-based models using three strategies. First, we apply quantitative performance metrics such as accuracy, ROC, and PRC. Second, we inspect the optimized parameters in each model to give an intuitive understanding of the behavioural claims that our models make. Finally, we analyze the sleep patterns that our models claim to infer from the unlabelled data set, and discuss whether these claims are intuitive, and whether they match the sleep patterns exhibited in the labelled data set.

Finally, in the discussion section, we compare the CNN-based and HMM-based approaches holistically, describing the scenarios under which users should prefer one over the other. We draw overall conclusions on the sleep patterns of Brown students using the (relatively more accurate) inferences made by our CNN model. We also discuss limitations of our models, and directions for future work.

2 Model

In this section, we begin by inspecting the data sets provided to us, summarizing some essential features in the sleep behaviour and step-count behaviour of the students who participated in data collection. This is detailed in section 2.1.

We then develop, as a first-iteration, a logistic regression model equipped with Cole-Kripke-like specifications. This model gives us a baseline accuracy of 84.6% when inferring sleep/wakefulness on the labelled data set. However, this model has a major shortcoming: it fails to characterize the sleep *patterns* - that is, frequency and duration of sleep - with any accuracy at all. This is discussed in section 2.2.

Departing from this baseline, we articulate two distinct approaches to modelling the relationship between sleep/wakefulness and step-counts, both based on the insights we obtain in section 2.1.

The first is a Convolutional Neural Network (CNN) model, which is nothing but a natural extension of the logistic regression model (as we explain in section 2.3). This model is trained entirely on the labelled data set, and achieves an accuracy of 92.97% in validation time. This is the highest accuracy obtained in any of our models.

The second is a Hidden Markov Model (HMM), which captures the intuition that sleep/wakeful states causally underlie the process through which step-counts are generated, and that this underlying processes of transitioning between wakefulness and sleep is a way to segment the time series of step-counts. This approach is detailed in section 2.4. As HMM in general is a class of powerful time-series segmentation techniques that encode underlying unobserved processes, this second model provides an intuitive understanding of the patterns with which students transition between sleep and wakefulness, and the way in which these inferred patterns explain the observed step-count data in both the labelled and unlabelled sets. This model is trained entirely on the unlabelled data set, and the predictive accuracy of this model, when validated against the labelled data set, is 89.6%.

In all our models, we make a few universal assumptions. First, we assume that the data provided to us, both labelled and unlabelled, does not contain any systemic errors or irregularities. This assumption is crucial in training our models, for it is impossible to account for such errors without extensive field-specific expertise in the general relation between sleep patterns and physical activity. Secondly, we assume that the majority of step-counts measured is accurate to a reasonable degree; while the models we create can handle the possibility of a few vastly inaccurate data points, they will likely produce erroneous results should a large portion of the data be sampled incorrectly. We do not define here numerically what it means for a "majority of the data" to be sampled accurately, however.

2.1 Data Inspection

We plot a sample section of time series of step-counts from the labelled dataset in figure 1. We notice immediately that the time series has clear qualitative differences between sleeping and wakeful stretches of time: the step-counts are more volatile during wakeful hours, and peak at higher values.

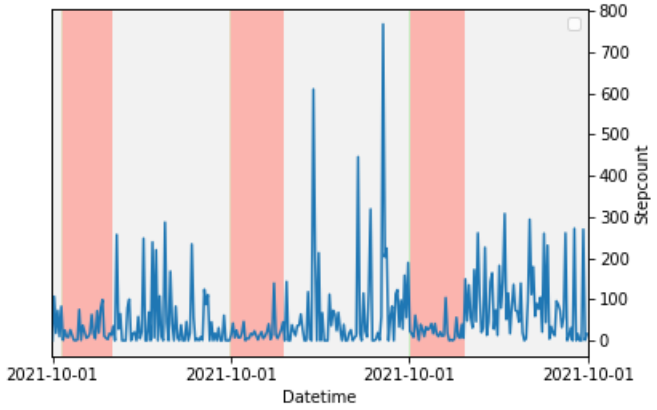
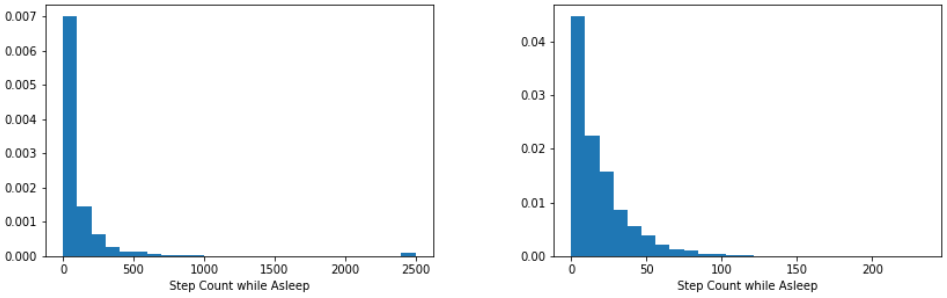


Figure 1: Time Series of Step-Counts
Note: Time asleep is colored pink.

However, the time series does not have hard boundaries: i.e. there does not exist any clear bound K below which a student's step count is unlikely to fall during wakeful hours. This matches our intuition: Brown students do spend long stretches of wakeful hours in sedentary states.

To understand the qualitative difference between the step-count time series during wakeful and sleeping hours, we plot the histogram distribution of step-counts in the labelled data set, stratifying for wakeful and sleeping hours, in figure 2.



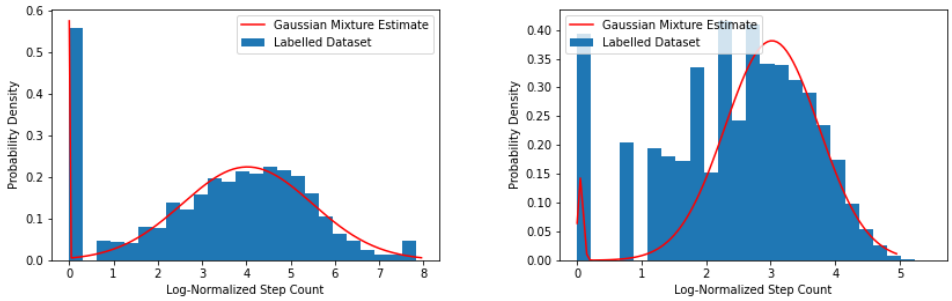
(a) Awake (b) Asleep

Figure 2: Histograms of Step Counts

These distributions are heavily skewed and difficult to work with. In keeping with the tradition discussed by Li et.al. [4], we log-normalize step-counts using the map

$$x \mapsto \log(x+1)$$

yielding the normalized histogram distributions shown in figure 3.



(a) Awake (b) Asleep

Figure 3: Log-Normalized Step Counts and their Gaussian Mixture Curves of Fit

We notice that the normalized step count while asleep seems to be bimodal, with one peak around 0, and another around 3. In their 2020 paper, Li et.al. models this phenomenon by claiming that physical activity levels during sleep follows a truncated normal distribution, with a blowup point at 0. That is, if O is the random variable representing the physical activity observed, then its probability density function is proportional to some normal distribution everywhere except at 0, where there is a blowup.

$$\lim_{\epsilon \to 0^+} \frac{1}{\epsilon} \mathbb{P}\{0 \leq O < \epsilon \,|\, \text{Asleep}\} > 0$$

On the other hand, the normalized step count while awake seems to be trimodal, with one peak around 0, a roughly bell-shaped cluster centered around 4, and yet another peak around 7.6. We believe this is explained by partitioning students' behaviour during wakeful hours into three types: sedentary, relaxed movements, and physical activities. This trimodal nature of step-counts during wakeful hours seems to be peculiar to our data set, as the existing literature doesn't seem to take account of this phenomenon [2, 4].

Considering the time limitation of this project, our decision is to provide some simplicity and uniformity to the situation, by explaining both step-count distributions as generated by Gaussian mixtures, with each mixture containing two Gaussians. Intuitively, the two Gaussians during waking hours correspond to sitting and moving, and the two Gaussians during sleeping hours correspond to motionless sleep and moving during sleep. This is admittedly a strong simplifying assumption, and the limitation associated with this assumption is discussed more in detail in section 4.3.

The Gaussian Mixture is fitted to our labelled data set using the sklearn.mixture package in Python. This results in the following Gaussian mixture distributions plotted as red curves in figure 3, with location, scale, and weight parameters given in table 1.

Parameter	GMM Fitted Value
μ_1	4.023
σ_1^2	2.157
ω_1	0.826
μ_2	0
σ_2^2	10^{-4}
ω_2	0.174

(a) Awake

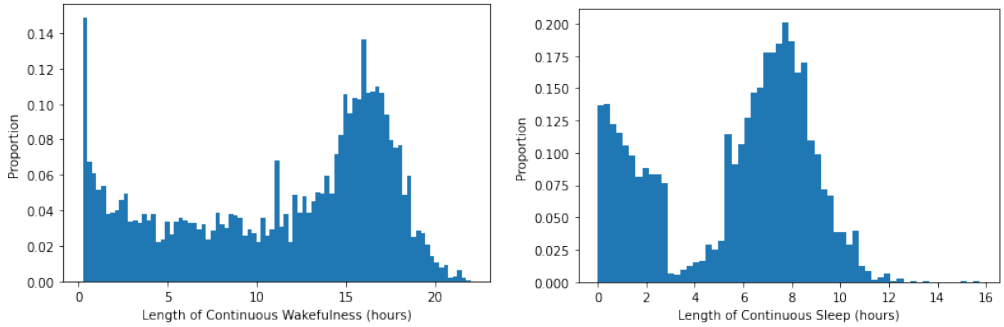
Parameter	GMM Fitted Value
μ_1	1.291
σ_1^2	0.879
ω_1	0.376
μ_2	3.097
σ_2^2	0.485
ω_2	0.630

(b) Asleep

Table 1: Gaussian Mixture Parameters for Log-Normalized Step Counts in Labelled Data

We are also interested in the lengths of continuous blocks of sleep/wakefulness, since this quantity is important for characterizing the patterns of sleep behaviour in students (we all know that sleeping for a continuous stretch of 8 hours is very different than taking four 2-hour naps). Again using the labelled data set, we obtain distributions of lengths of continuous sleep/wakefulness in figure 4.

We notice that both of the distributions obtained are bimodal. The lengths of continuous blocks of sleep are expected to be bimodal - this corresponds to the fact that blocks of sleep are either coming from naps, or from sustained sleep. As we will see in the results section (section 3), our CNN-based model will implicitly capture this fact, while our HMM-based model will require some future enhancement to capture this bimodality (These possible future enhancements are described in section 4.3). On the other hand, the bimodality of lengths of continuous wakefulness is more difficult to explain. We believe the cluster of shorter blocks of wakefulness correspond to short durations of wakefulness at night, or to durations of wakefulness on days when the subject takes naps. We did not have enough time to account for this bimodality in our model, although as shown in section 3, the CNN model seems to implicitly account for this as well. We acknowledge this limitation again in section 4.3.



(a) Distribution of Length of Wakefulness (b) Distribution of Length of Sleep

Figure 4: Histograms of Continuous Activity

2.2 Logistic Regression

Inspired by the Cole-Kripke model as discussed in section 1.2, we start with a regression based analysis of the labelled data set as the baseline model. Following Cole-Kripke, we define our features to be

$$\mathbf{x}_t = \begin{pmatrix} A_{t-w_b} \\ \vdots \\ A_{t+w_f} \end{pmatrix}$$

where A_t denotes the number of steps recorded in the 15-minute period ending at t, and w_b, w_f are our backward and forward window-sizes (Cole-Kripke used $w_b = 4$ and $w_f = 2$). The total window size is $w = w_b + 1 + w_f$. Our labels are

$$y_t = \begin{cases} 1 & \text{If subject is labelled asleep at time } t \\ 0 & \text{otherwise} \end{cases}$$

Notice, importantly, we mark sleeping time as 1; rather than waking time as 1.

We inspect the relation between the step count and wakefulness to determine the method of regression to use. For example, here we display a scatter plot of the step count of time period t against sleep at time period t:

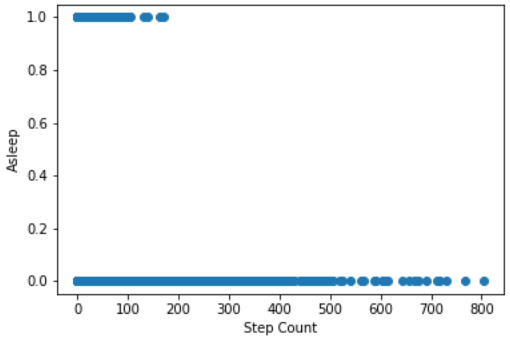


Figure 5: Step Count vs. Sleep

Unsurprisingly, since we are in a situation where the dependent variable is binary, a logistic curve is desirable. So we parametrize our model by $\theta \in \mathbb{R}^w$ and $b \in \mathbb{R}$, and define the model to be the computation

$$h_\theta(\mathbf{x}_t) = \frac{1}{1 + \exp(-\theta^T \mathbf{x}_t - b)}$$

where $h_\theta(\mathbf{x}_t)$ is interpreted as the likelihood that the subject is asleep at time t, given that the step-count behaviour in the window of w measurements is $\mathbf{x}_t$. During test-time, the model is interpreted as predicting the state "asleep" if $h_\theta(\mathbf{x}_t) > 0.5$, and as predicting the state "awake" otherwise.

The parameters θ, b are optimized by maximizing the log-likelihood function

$$l_\theta(\mathbf{y}; \mathbf{x}, \theta) = \log \prod_t h_\theta(\mathbf{x}_t)^{y_t} (1 - h_\theta(\mathbf{x}_t))^{1-y_t} = \sum_t \left[y_t \log h_\theta(\mathbf{x}_t) + (1 - y_t) \log(1 - h_\theta(\mathbf{x}_t)) \right]$$

by gradient descent.

This optimization is done with respect to the labelled dataset consisting of 100 students' behaviour over a month, using the package sklearn in python. We set $w_b = 4$ and $w_f = 2$ in line with Cole-Kripke. The resulting optimized parameters are

$$\theta^* = \begin{pmatrix} -0.002 \\ -0.000 \\ -0.005 \\ -0.003 \\ -0.011 \\ -0.011 \\ -0.036 \end{pmatrix} \qquad b^* = 1.940$$

The resulting mean accuracy with respect to the labelled dataset is 0.846. For intuition, a slice of the resulting logistic curve is displayed in figure 6.

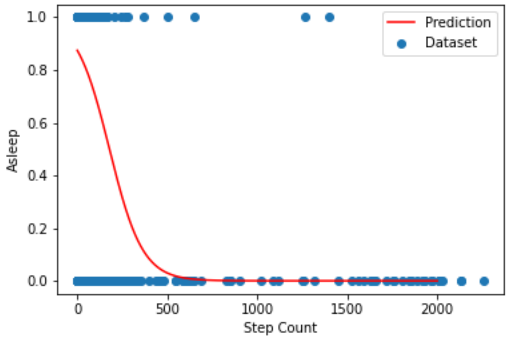


Figure 6: Slice of Logistic Curve

This result will serve as a baseline for our later iterations. The main advantage of the logistic Regression model is its simplicity - each entry of θ^* is interpretable as a correlation between a step-count and the log-odds of being asleep. However, the major drawback is that, while having a reasonable accuracy (0.846), the predicted sleep-behaviour does not correspond to actual behaviour. In figure 7, we see a three-day sample of the logistic regression model's predictions, plotted against the corresponding labels given to us in the labelled data set. While the predictions generally follow the trend of the labels, there is far too frequent switching between predicted sleep and predicted wakefulness. Thus, while the logistic regression model gives us a way to fairly accurately infer the sleep *time* of students, it does not allow us to infer sleep *patterns*. To address this issue, we develop two improved models.

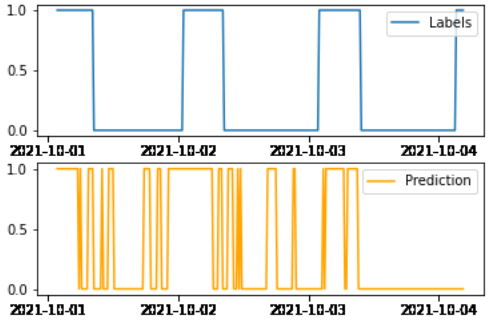


Figure 7: Sample Predictions by Logistic Regression Model
Note: Wakefulness coded as 0; sleep coded as 1.

2.3 Convolutional Neural Network

2.3.1 Background on Convolutional Neural Networks

The natural next step from logistic regression is a convolutional neural network, which takes in an input X, predicts a label $\hat{y}$ - in this case, either 0 for awake and 1 for asleep - and adjusts its trainable parameters based on a comparison with the true label y.

In order to examine the input at a higher resolution, which is necessary to discern, for example, transitions from sleep to wakefulness and vice versa, and to maintain translational invariance, which is necessary considering our input is derived from a sliding window over the time series (see 2.3.3), we utilize convolution. Convolution entails a set of learnable weights called "filters" (or "kernels"), which, analogously to the trainable parameter θ in logistic regression, is overlaid upon and moved over the input, element-wise multiplying

with the input values to produce an output

$$c_i = \sum_n x_{i+n} w_n$$

where W is the filter. Just like in the logistic regression model, the fact that the filter weights are trainable allows them to "pay attention" to different parts of the window of input. This helps to ensure that our neural network is able to output excellent predictions regardless of the time, length, or magnitude of measurement.

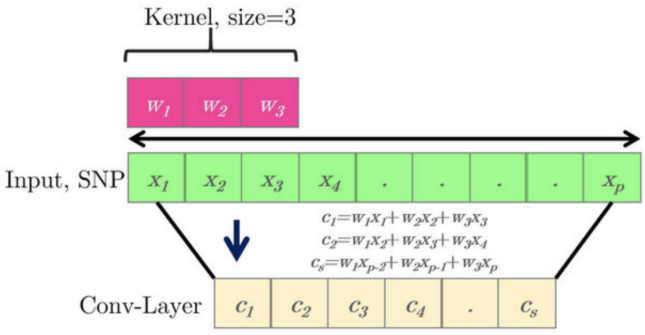


Figure 8: How Convolution Works

What makes a neural network truly different from a mere regression, however, is its multiple layers of *nonlinearity*. We do this by stacking several convolutional layers, and introducing a nonlinear activation function between each. While the dominant inter-layer activation function to avoid vanishing or exploding gradients is the rectifier function, or ReLU, we chose to utilize a leaky ReLU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ ax & \text{otherwise} \end{cases}$$

in order to avoid the dead ReLU problem and therefore the network learning to predict only a single label. We thereby avoid a major potential problem given our sleep data is class-imbalanced (people are asleep only $\tilde{1}/3$ of the time).

For the final output in our CNN model, we make use of the softmax function

$$p_j = \frac{e^{l_j}}{\sum_k e^{l_k}}$$

where l_j are the logits outputted by the final linear layer of the model. This is simply to ensure that the last, dense layer of our neural network outputs predicted classification probabilities p_j which sum to 1.

2.3.2 Other Knickknacks

Two tools implemented in our CNN worth discussing are batch normalization and dropout.

Batch normalization is a normalization of activations for each feature at each layer, leading to more stable inputs and faster training. See 2.3.3 for how batches are set up in our model.

$$x' = \frac{x - \mu_{batch}}{\sigma_{batch}}$$

$$\mu_{batch} = \frac{\sum_{i=0}^{batch\_size} x_i}{batch\_size}$$

$$\sigma_{batch} = \sqrt{\frac{\sum_{i=0}^{batch\_size} (x_i - \mu_{batch})^2}{batch\_size}}$$

Dropout is the process of randomly setting nodes in a layer to zero. By introducing some stochastic source of noise within the network, we force each neuron to learn non-trivial weights and not rely on spurious correlations. In a certain sense we are "smoothing" out the cost landscape. Dropout is particularly important for this paper because the step data and sleep labels may be inaccurate due to device noise or user error. If our model is capable of learning properly with artificially introduced noise in the form of dropout, we know it is capable of dealing with the noise and outliers in the data.

2.3.3 Model Design

As the size and shape of the hidden layers (convolutional and dense layers between the initial input and the final predictive dense layer) are hyperparameters, along with parameters like the Adam optimizer's learning rate, these are things we manually tune. Our final architecture looks like the following.

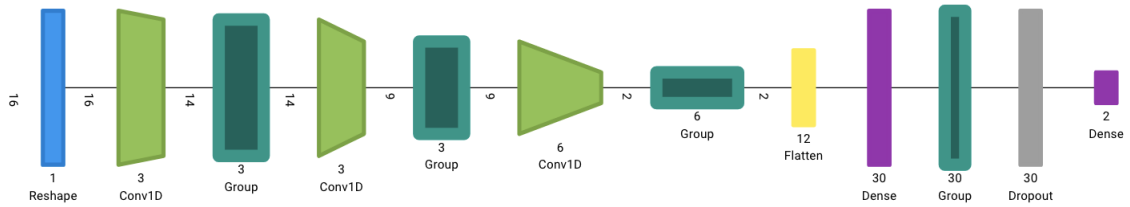


Figure 9: Our CNN Architecture

In short, we have three convolutional layers of increasing size, moving from a "higher" to "lower" resolution view, each with batch normalization and a Leaky ReLU layer, followed by two dense layers.

2.3.4 Model Training

There are a couple steps of preprocessing necessary to ensure our model trains well. Unlike in the logistic regression model, we now select a window size of 16 in our CNN model (this window size was selected by hand-tuning).

We preprocess the 297600 step-count datapoints contained in the labelled data set into a log-normalized 297585×16 matrix, where the rows are every single 16-point window of the original time series. Each input represents the step count at time point t, along with the values at neighboring points $t-8$ to $t-1$ and $t+1$ to $t+7$. The corresponding label is the person's state at time t, encoded as a one-hot matrix.

In other words, the neural network seeks to predict the sleep state of the person at the "center" of each 16-point input. By sliding the window along the entire time series we produce predictions for the state at every time point. *Nota bene* we inevitably miss the 8 and 7 points at the beginning and end of the series, but this is an acceptable loss if we are adamant upon not padding the ends of the input with zeroes.

For training, we randomize the order of our dataset and split 75% of it into a training set, upon which the network learns its weights, and 25% into a validation set, which the network never sees until it is time to test it and check metrics for accuracy.

During the process of training we run the network over the entirety of the training set for 10 epochs, applying the ADAM optimizer to minimize categorical cross-entropy loss.

In order to update the weights multiple times per run-through of the dataset, we train on batches, or small randomly-sampled subsets of the training data, and update our parameters after each batch. This is known as stochastic gradient descent. Since generally the larger batch size the better, we choose a batch size of 200, the most our Colab notebook can reasonably handle.

2.3.5 Model Interpretation

Because neural networks are somewhat more of a black box than many other models, some interpretation can be useful to examine how the network is arriving at its predictions. Due to the low dimensionality of the input and small size of the network, methods like saliency maps and gradient-based approaches are unsuitable, and due to the low dimensionality of the binary output so too is perturbation. However, the

network's small size does allow us to manually examine the convolutional filter weights, which provide insight into which parts of the input window each filter is "paying attention" to.

2.3.6 Model Assumptions

The advantage of the CNN model is that the weights are all randomly initialized, and it relies upon no assumptions apart from the inputs and labels with which it is provided. While this certainly doesn't prevent them from training improperly, it does mean that which biases they acquire are not hard-coded into them by their architect.

2.4 Hidden Markov Model

2.4.1 Background on Hidden Markov Models

The next model we test is based on a time series segmentation approach, namely with the use of Hidden Markov Models, to create a predictive model for sleep. We follow Rabiner's exposition of Hidden Markov Models in this section [7]. An HMM is a discrete time stochastic process consisting of two components:

1. A set of observations (usually denoted by $\mathcal{O} = \{O_1, ..., O_T\}$) drawn for a sample space V, and
2. A Markov chain $\mathcal{Y} = \{Y_1, ..., Y_T\}$ with state space $Q = \{q_1, ..., q_N\}$, transition probability matrix A, and initial probability distribution π.

At any time t, if the Markov chain is in state $Y_t = q_i$, then O_t is distributed according to a probability distribution B_{q_i} (often called the emission distribution). Thus, each observation O_t is only dependent on the process' current state Y_t, rather than relying on any other historical information.

The essential property of HMM is that, while $\mathcal{O}$ can be well observed for the entirety of the process (until some finishing time T), there is limited information, if any at all, about the transition probabilities $A = (a_{q_i q_j})$ of the Markov chain $\mathcal{Y}$, and about the emission distributions B_{q_i} (thus, the Markov process is "hidden"). The quantity of interest is $Y_t \,|\, \mathcal{O}$, that is, the state of the hidden Markov process at time t given all of the observations. We make such calculations using unsupervised machine learning methods, namely Viterbi's algorithm and the Baum-Welch algorithm, which we describe more in depth in section 2.4.3.

2.4.2 Model Design

We apply the HMM paradigm to our problem at hand. To do this, we make two defining assumptions for HMM: 1) that the sequence of behaviour switching between sleep and wakefulness is Markov; and 2) that during a period of continuous sleep or wakefulness, the time series of step-counts is stationary. These assumptions are explored more in-depth in section 2.4.4.

With these assumptions, the observations $\mathcal{O} = (O_t)_{t=1}^T$ will correspond to the step-counts measured by a fitness tracker during the 15-minute interval prior to time t. The state space is $Q = \{W, S\}$, where W denotes the "awake" status of a person and S denotes the sleeping status. We model $\mathcal{Y} = \{Y_t\}_{t=1}^T$, where each Y_t denotes whether the subject is awake (W) or asleep (S), as a Markov chain taking states in Q with transition probabilities

$$A = \begin{pmatrix} a_{WW} & a_{WS} \\ a_{SW} & a_{SS} \end{pmatrix}$$

and with initial probabilities $\pi = (\pi_W, \pi_S)$.

We let B_W and B_S denote emission distributions from each state - that is, B_W and B_S are the probability distributions underlying the measured step-counts, given the subject is awake or asleep, respectively. As discussed in section 2.1, our HMM model assumes that B_W and B_S are Gaussian Mixtures - that is, their density functions are expressible as

$$dB_q(x) = \left(\omega_{1,q} \cdot \phi(x \,|\, \mu_{1,q}, \sigma_{1,q}^2) + \omega_{2,q} \cdot \phi(x \,|\, \mu_{2,q}, \sigma_{2,q}^2)\right) \, dx$$

for $q \in \{W, S\}$, where ϕ denote density functions of normal distributions, and $\omega_{1,q}, \omega_{2,q}$ are weights satisfying $\omega_{1,q} + \omega_{2,q} = 1$. This gives us, in total, 18 trainable scalar parameters: 4 entries in A, 2 entries in π, and the

scalars

$$\omega_{1,W}, \omega_{1,S}, \omega_{2,W}, \omega_{2,S}$$
$$\mu_{1,W}, \mu_{1,S}, \mu_{2,W}, \mu_{2,S}$$
$$\sigma^2_{1,W}, \sigma^2_{1,S}, \sigma^2_{2,W}, \sigma^2_{2,S}$$

appearing in the expressions for B_S and B_N. The above design of our HMM model is summarized in figure 10.

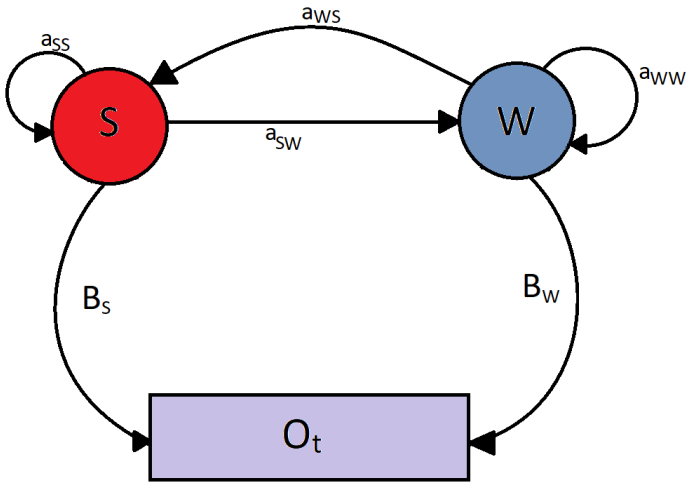


Figure 10: HMM Visualization

In order for HMM model training to be successful, we need to give prior estimate values to all trainable parameters. To ensure that these prior estimates somewhat resemble the behaviour of a reasonable person, we estimate them from the labelled dataset (although the training of the model, described in section 2.4.3, is done entirely on the unlabelled dataset).

Using the labelled data, we compute prior transition probabilities for A and π by counting transitions between sleep status between each 15 minute interval for all students, calling these counts C_{WW}, C_{WS}, C_{SW}, and C_{SS}. For instance, if a student is asleep at time $t = t_i$ but awake at $t = t_{i+1}$, we increase the value of C_{SW} by 1. Then we set the prior transition probabilities A_0 and prior initial probabilities π_0 as

$$A_0 = \begin{pmatrix} a_{WW} & a_{WS} \\ a_{SW} & a_{SS} \end{pmatrix} = \begin{pmatrix} C_{WW}/(C_{WW} + C_{WS}) & C_{WS}/(C_{WW} + C_{WS}) \\ C_{SW}/(C_{SW} + SS) & C_{WW}/(C_{SW} + SS) \end{pmatrix}$$

$$\pi_0 = \begin{pmatrix} \pi_W & \pi_S \end{pmatrix} = \frac{1}{C_{WW} + C_{WS} + C_{SW} + C_{SS}} \begin{pmatrix} C_{WW} + C_{WS} & C_{SW} + C_{SS} \end{pmatrix}$$

Computing this calculation with the provided data, we find that

$$A_0 = \begin{pmatrix} .98 & .02 \\ .04 & .96 \end{pmatrix} \quad \pi_0 = \begin{pmatrix} 0.64 & 0.36 \end{pmatrix}$$

Similarly, we use the labelled data to calculate the initial distributions for B_W and B_S by fitting a Gaussian Mixture Model to the histograms distributions of step-counts in waking/sleeping hours given in the labelled data set (see section 2.1, especially figure 3 and table 1). We reproduce here in table 2 the results of computation in section 2.1 for ease of reference.

Parameter	Prior Value
μ_1	4.023
σ_1^2	2.157
ω_1	0.826
μ_2	0
σ_2^2	10^{-4}
ω_2	0.174

(a) Awake

Parameter	Prior Value
μ_1	1.291
σ_1^2	0.879
ω_1	0.376
μ_2	3.097
σ_2^2	0.485
ω_2	0.630

(b) Asleep

Table 2: Prior Gaussian Mixture Parameters for Emission Functions

This fully describes and initializes our HMM model, and we are ready to train the model.

2.4.3 Model Training using Baum-Welch and Viterbi Algorithms

With the initial parameters calculated above, we use Python's scikit-learn library, specifically the hmmlearn package, to implement the machine learning algorithms necessary to optimize our model. We select the Baum-Welch algorithm (see [7]) for optimization of the trainable parameters A, π, B_S, B_W, and then the Viterbi algorithm (see [3]) for finding the maximum-likelihood estimate for the chain $(Y_1, ..., Y_T)$, given observed step-counts. Both Baum-Welch and Viterbi are unsupervised machine learning methods known to perform well for HMM optimization. We provide a brief description of both algorithms below, though more comprehensive explanations can be found at the sources provided at the end of the paper. A reader familiar with this may prefer to skip ahead to section 2.4.4.

The Baum-Welch algorithm, a type of expectation maximization algorithm, uses gradient descent and dynamic programming to find parameters for A, π, B_W, and B_S which maximize the likelihood of producing the observations $\mathcal{O}$. Let $\theta = (A, \pi, B_W, B_S)$. Baum-Welch looks for the ideal value

$$\theta^* = \arg\max_{\theta} \mathbb{P}\{\mathcal{O} \mid A, \pi, B_W, B_S\}$$

or at least a local maximum serving as an approximate to the above value. The issue, of course, is that the likelihood to be maximized, namely

$$\mathbb{P}\{\mathcal{O} \mid A, \pi, B_W, B_S\} = \int \mathbb{P}\{\mathcal{O} \mid \mathcal{Y},\ A, \pi, B_W, B_S\} \cdot \mathbb{P}\{\mathcal{Y} \mid A, \pi, B_w, B_S\}\, d\mathcal{Y}$$

is unknown to us. Baum-Welch overcomes this issue by iteratively updating A, π, B_W, B_S to the values which maximize the expectation (over $\mathcal{Y}$) of the log of the likelihood, with respect to the latest estimates for A, π, B_W, B_S.

It should be noted that Baum-Welch is sensitive to initial conditions, because it converges to local maxima. It is therefore crucial that we initialize trainable parameters at reasonable points, which is an issue addressed in section 2.4.2.

The Viterbi algorithm is then an algorithm using dynamic programming, to find the sleep/wakeful behaviour path that maximizes the following likelihood:

$$\mathcal{Y}^* = \arg\max_{\mathcal{Y}} \mathbb{P}\{\mathcal{O}, \mathcal{Y} \mid A^*, \pi^*, B_W^*, B_S^*\}$$

where A^*, π^*, B_W^*, B_S^* are the optimal trainable parameters returned by Baum-Welch. The idea is to dynamically compute two 2-row tables T_1 and T_2, such that $T_1[q, j]$ stores the probability of the most likely path so far $(Y_1, ..., Y_j)$ with $Y_j = q$, $q \in \{W, S\}$; and $T_2[q, j]$ stores the second-to-last entry of said most likely path[1]:

[1]Pseudocode obtained from wikipedia entry for Viterbi: https://en.wikipedia.org/wiki/Viterbi_algorithm

```
function VITERBI(O, S, Π, Y, A, B) : X
    for each state i = 1, 2, ..., K do
        T_1[i, 1] ← π_i · B_{i y_1}
        T_2[i, 1] ← 0
    end for
    for each observation j = 2, 3, ..., T do
        for each state i = 1, 2, ..., K do
            T_1[i, j] ← max_k (T_1[k, j − 1] · A_{ki} · B_{i y_j})
            T_2[i, j] ← arg max_k (T_1[k, j − 1] · A_{ki} · B_{i y_j})
        end for
    end for
    z_T ← arg max_k (T_1[k, T])
    x_T ← s_{z_T}
    for j = T, T − 1, ..., 2 do
        z_{j−1} ← T_2[z_j, j]
        x_{j−1} ← s_{z_{j−1}}
    end for
    return X
end function
```

Because both of the algorithms utilized fall under the class of unsupervised learning, we are able to train our model using the 400 unlabelled data points. We use the 100 labelled points instead for validation.

2.4.4 Model Assumptions

The HMM approach to sleep classification is based on a two essential assumptions: 1) that the sequence of behaviour switching between sleep and wakefulness is Markov; and 2) that during a period of continuous sleep or wakefulness, the time series of step-counts is stationary, with a 2-mixed-Gaussian distribution.

The first claim has been reviewed extensively, and numerous studies demonstrate that under suitable conditions, sleep is Markovian in nature. A paper by Perez-Atencio et. al. demonstrates the effectiveness of using Markov chains with 4 states to model sleep in mice [5]; other studies such as one undertaken by Bizzotto et. al. support similar claims for sleep cycles in humans [1]. Both papers consider more nuanced models for sleep, however, using Markov chains with state spaces including four different kinds of sleep: NREM (stage 1-3) sleep, and REM (stage 4) sleep. Though our model is notably simpler, using only two states, results described later in the paper suggest that is still an effective design choice.

One concern of the model for sleep is that there is a statistically significant chance under assumptions of the Markovian property that an individual will spend a very short amount of time asleep, or that they will rapidly alternate between sleep and wakefulness within a short time interval. Li argues in his paper that this is a fixture of Markov chain models for sleep, but that such an issue can be rectified by "smoothing" over short periods of sleep or wakefulness by switching their state [4]. Due to time constraints, we do not attempt this process here. Furthermore, analysis of the stationary distribution p for our transition matrix A demonstrates that

$$p = \begin{pmatrix} p_W & p_S \end{pmatrix} = \begin{pmatrix} .71 & .29 \end{pmatrix}$$

where p_W is the proportion of time spent asleep and p_S is the proportion of time spent asleep in the long run. This agrees closely with the labelled data, in which the proportion of time spent awake among all 100 participants was 0.67 and the proportion of time spent asleep was 0.33. As such, it follows that Markov chains, while not perfect, provide a suitable simplification for modeling sleep cycles.

The second assumption of the model is justifiable along the following lines: that the emission distributions are 2-mixed-Gaussian in nature, is justified through a preliminary exploration of the data set. As plotted in Figures 3a and 3b, observation shows that the log distribution of step count both for wakefulness and sleep states are quite similar to Gaussian mixture distributions - this was discussed above in section 2.1. On the other hand, the assumption that within each continuous stretch of wakefulness and sleep, the time series

of step-counts is stationary, is admittedly a very simplifying assumption. The limitation induced by this assumption is discussed in section 4.3.3.

2.5 Evaluation Metrics

We are given a small set of labelled data against which we can check our predictions. To take advantage of this, the metrics we are using to quantify the performance of our models are as follows:

1. Accuracy ($Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$), a rough measure of systematic errors.
2. The receiving operator characteristic (ROC) curve, or a plot of true positive rate ($TPR = \frac{TP}{TP+FN}$) against the false positive rate ($FPR = \frac{FP}{FP+TN}$), and the area under it (AUROC). It is therefore sensitivity as a function of fallout, and reflects degree of Type I error.
3. The precision-recall curve (PRC), or a plot of recall ($Recall = TPR = \frac{TP}{TP+FN}$) against the precision ($Precision = \frac{TP}{TP+FP}$), and the area under it (AUPRC). It is a measure of relevance which better deals with class imbalances (like in this paper's dataset) due to not using a false positive rate which contains true negatives.

where TP is true positives, FP is false positives, TN is true negatives, and FN is false negatives.

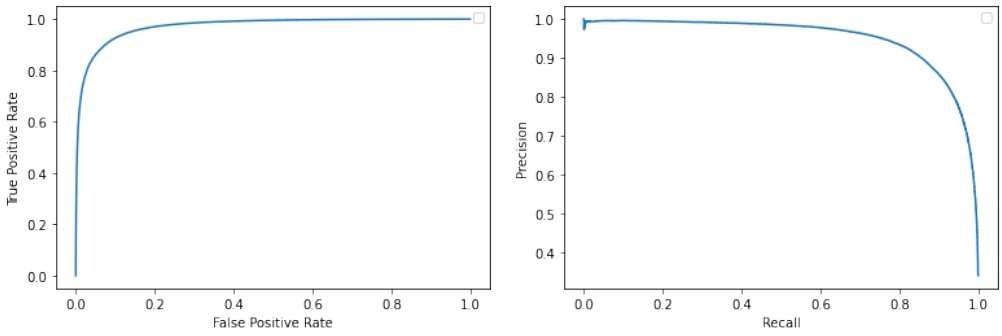
We also plot distributions of various sampled random variables, such as distribution of lengths of continuous sleeping blocks and proportion of time spent sleeping each day. When doing so, we consider only the standard statistics: mean (μ) and variance (σ^2). Although higher order statistics such as skew ($\tilde{\mu}_3$) and kurtosis (Kurt) could also be measured, these values are arguably harder to interpret, and much of the same conclusions can be drawn by simple observation of the distributions.

3 Results

3.1 Convolutional Neural Network Results

3.1.1 Performance

Following training, our CNN's best snapshot is evaluated with the aforementioned metrics on the unseen validation set, producing a validation loss of 0.1750, accuracy of 0.9297, AUROC of 0.9811, and AUPRC of 0.9809 - all far superior to the logistic regression baseline. Clearly, the CNN is properly classifying true positives and negatives as such, and generally avoiding false predictions of either class.



(a) Receiver Operating Characteristic Curve (b) Precision-Recall Curve

Figure 11: Metrics for CNN Performance

Let us inspect an example of the predictions given by our CNN model when validated against labelled data, comparing it to the given labels. Figure 13 plots the CNN model's predictions for student 0 on the stretch of time between Oct. 5 and Oct. 10.

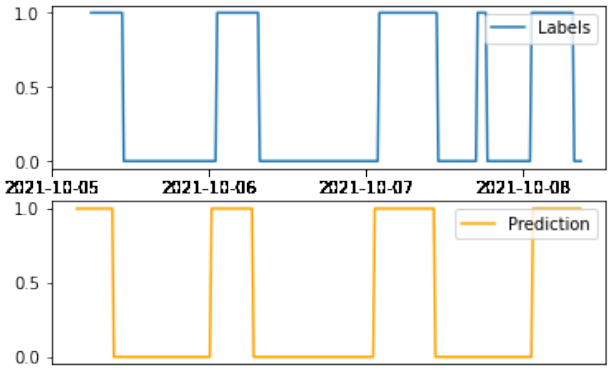


Figure 12: Sample CNN Predictions against Actual Labels

We notice that the CNN model successfully captures many sustained periods of sleep, but has more trouble with very short naps. The first makes sense given the overall high accuracy of our model, and the second is reasonable especially because the moving window from which we are preprocessing our inputs is length 16, or 4 hours. This means inputs containing a nap, labelled as 1 (sleep), may have long periods of high step activity on the flanks, giving the network more trouble distinguishing them from periods of wakefulness. More on this limitation in section 4.3.4.

3.1.2 Interpretation by Filter Weights

The black-box nature of the neural network can be partially remedied by qualitative inspection of the weights in each of the model's 3, 3, and 6 convolutional filters, respectively. The filters are small on account of the small size of the input, but are nonetheless useful. The dark squares indicate small or inhibitory weights and the light squares represent large or excitatory weights.

(a) Filters for Conv Layer 1 (b) Filters for Conv Layer 2 (c) Filters for Conv Layer 3

Figure 13: Convolutional Layer Filter Weights Visualized

Using this intuition, we can see that the filters in the first convolutional layer detect the step values at the center of or immediately following the center of the window. And in the second and third convolutional layers, we can see that many of the filters either focus on detecting the center values of the window, or the two ends of the window. This makes sense: for windows that take place entirely within a period of wakefulness or sleep, the center is representative of the overall classification; for windows that span the transition from wakefulness to sleep or vice versa, it becomes necessary to check the extrema to see what transition is happening.

3.1.3 Sleep Pattern Analysis

Applying our trained CNN to predict the states of unlabelled data, we seek to make observations about student population sleep patterns.

First, we consider the distribution of lengths of continuous sleep blocks over all students. This is important because, as we all know, sleeping for 8 hours continuously is very different than taking four 2-hour naps. We compare the CNN-predicted distribution of continuous sleep duration, with the continuous sleep duration given by the labelled data set, obtaining figure 14.

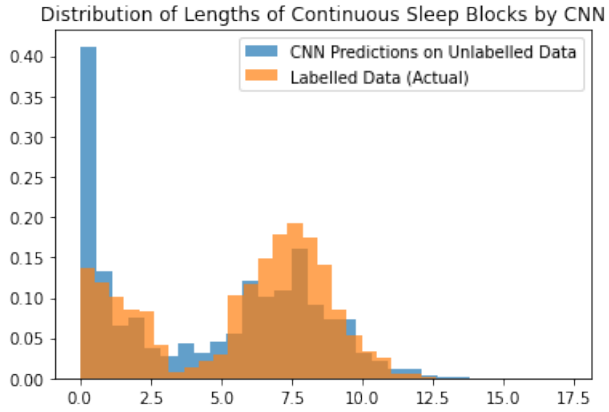


Figure 14: Distribution of Lengths of Continuous Sleep Predicted by CNN

By qualitative inspection, it is clear that both the labelled and unlabelled data display a bimodal distribution, with the first cluster covering the range $t = 0$ to 12 and the second covering the range $t = 12$ to 50. We'll call these the "nap" cluster and the "sustained sleep" cluster, respectively. If we filter for lengths of continuous sleep less than and greater than 12, respectively, we obtain the following means and variances, which are more descriptive of the true distribution.

Statistic	Net	Sustained	Nap
μ	5.65	7.49	1.34
σ^2	9.63	2.22	0.62

(a) Labelled Data

Statistic	Net	Sustained	Nap
μ	4.57	7.30	0.80
σ^2	12.49	3.61	0.49

(b) Unlabelled Data

Table 3: Statistics for Distribution of Lengths of Continuous Sleep

It appears that the means of the labelled and unlabelled distribution are similar, as expected given the assumption that they are both representative of the population of students. However, the CNN's predictions seem to be more conservative, generally imputing fewer periods of continuous sleep for each length; it also has higher variance. In particular, the CNN predicts an outsize number of very short naps, 15-45 minutes long. It makes sense that there are some isolated false positives like this scattered here and there; the original data is quite noisy, and brief periods of waking relaxation and inactivity may well be brief naps, as far as step counts are concerned. We can say that length of continuous sleep is a pattern the CNN captures with fair success. In both the labelled and unlabelled data, therefore, students are mostly napping for $\tilde{7}.5$ hours for sustained sleep, most likely during the night, and about 1 hour for naps.

Next we consider the CNN-inferred distribution of the proportion of the day spent asleep over all students. We compare this to the distribution of proportion of sleeping time in labelled data.

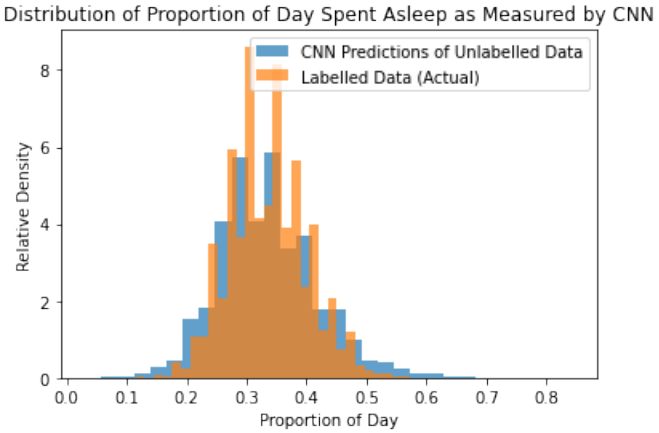


Figure 15: Distribution of Proportion of Day Spent Sleeping

The labelled data has a mean of 0.33 and a variance of 0.0040, and the unlabelled data a mean of 0.33 and a variance of 0.0066. This is particularly impressive: the CNN's predictions almost perfectly match the true values in labelled data. We can say with confidence that the CNN is properly capturing the proportion of students' days spent sleeping in its predictions. In both the labelled and unlabelled data, therefore, students are mostly sleeping about 1/3 of the day.

Finally, we use the CNN inference to characterize times in the day when students tend to go to sleep: that is, time points when students transition from a state of wakefulness to a state of sleep. We compare this again to the analogous values given by labelled data.

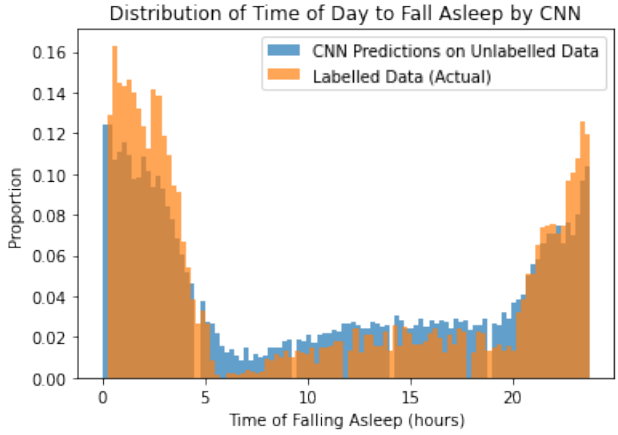


Figure 16: Distribution of Times in the Day when students fall asleep

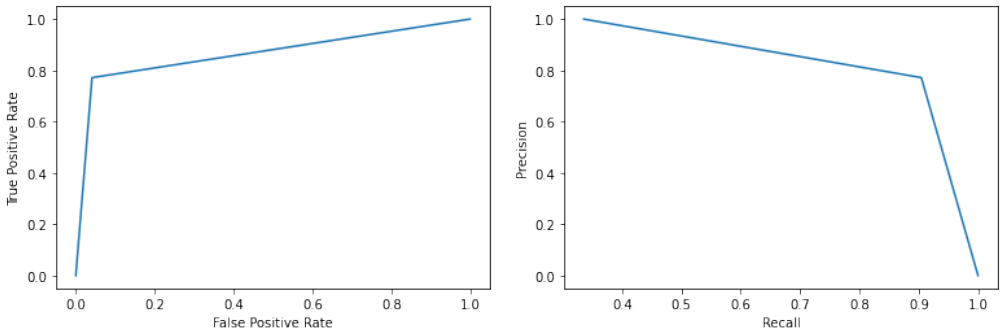
Because the data is cyclic, or categorical, it does not make sense to define a mean or variance. That said, it seems that the CNN does indeed capture when students tend to call asleep. However, the CNN's predictions are slightly less abundant in the early hours of the morning, and more abundant in the middling hours of the day. In both the labelled and unlabelled data, therefore, students are generally falling asleep near midnight, either in the late hours of the evening (e.g. 11pm) or in the early hours of the morning (e.g. 1am), with some sleep scattered elsewhere throughout the day.

3.2 Hidden Markov Model Results

3.2.1 Performance

The average accuracy of the HMM model, when validated against the labelled data set, is 0.896. This is a substantial improvement over the baseline.

Now, one feature of the HMM model is that the predictions are generated by Viterbi algorithm - as discussed in section 2.4.3 - there is no underlying "predicted likelihood" that a subject is asleep, but only the binary prediction maximizing the likelihood of observed emissions. Thus, when computing the ROC and PRC curves, we obtain only three threshold points. Nevertheless, for consistency in comparison, we plot these curves in figure 17.



(a) Receiver Operating Characteristic Curve (b) Precision-Recall Curve

Figure 17: Metrics for HMM Performance

The AUROC and AUPRC are 0.87 and 0.88, respectively. These are substantially worse than the results from the CNN model. Indeed, in the rest of this section, we see that the HMM model underperforms in terms of accuracy with regard to most aspects of sleep-pattern. However, we also see that the HMM model is much more interpretable than the CNN model, and that, by virtue of being an unsupervised algorithm, it is capable of training on larger and lower-cost data sets.

3.2.2 Interpretation of Trained Parameters

In this section, we interpret the trained parameters in our HMM model component-by-component. The first component of the HMM model is the hidden markov chain itself. The trained parameters characterizing the hidden markov chain are:

$$A = \begin{pmatrix} .96 & .04 \\ .10 & .90 \end{pmatrix}$$

$$\pi = \begin{pmatrix} .58 & .42 \end{pmatrix}$$

This tells us the following: first, the matrix A models the general pattern in which Brown students enter periods of sleep/wakefulness. Whenever a student is awake at time t, then overall there is a 0.96 chance that she will continue to be awake 15 minutes after time t, and a 0.04 chance that she will be asleep in 15 minutes; whenever a student is asleep at time t, then overall there is a 0.10 chance that she will awake in 15 minutes, and a 0.90 chance that she will stay asleep.

On the other hand, π is simply the model's estimated distribution for the initial state of a student at the beginning of each stretch of data in time. For our analysis, this corresponds to the probability that a student is asleep/awake at midnight of each day. That is to say, overall, 58% of Brown students are found awake at midnight each day.

Now, the transition probabilities A also allows us to characterize the average amount of time spent asleep by Brown students, as predicted by the HMM model: on average, Brown students spend 28.7% of their time sleeping. However, this seems to be an underprediction: according to the labelled data set, students spend 33.6% of their time sleeping. Indeed, in figure 18, we see that the HMM model consistently underpredicts the

amount of time spent each day sleeping. The amount of time spent sleeping, according to labelled data, is bell-shape distributed with a mean of 0.336 and a variance of 0.004; whereas according to HMM predictions, it has a mean of 0.287 and a variance of 0.007. We discuss possible reasons for this underprediction, together with other related issues, below in section 3.2.3.

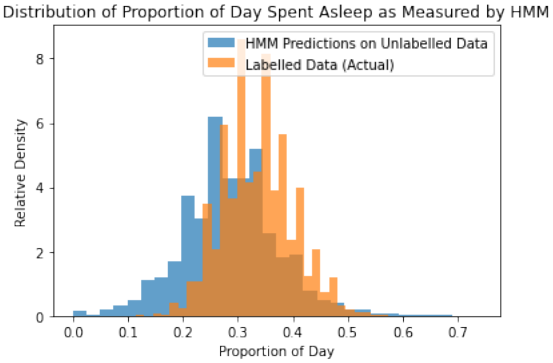


Figure 18: Distribution of Proportion of Day Spent Asleep as Predicted by HMM

Let us now interpret the second component of the HMM model, namely, the emission distributions. The optimized emission parameters are found in table 4:

Parameter	Optimized Value
μ_1	3.97
σ_1^2	2.05
ω_1	0.79
μ_2	0
σ_2^2	0
ω_2	0.21

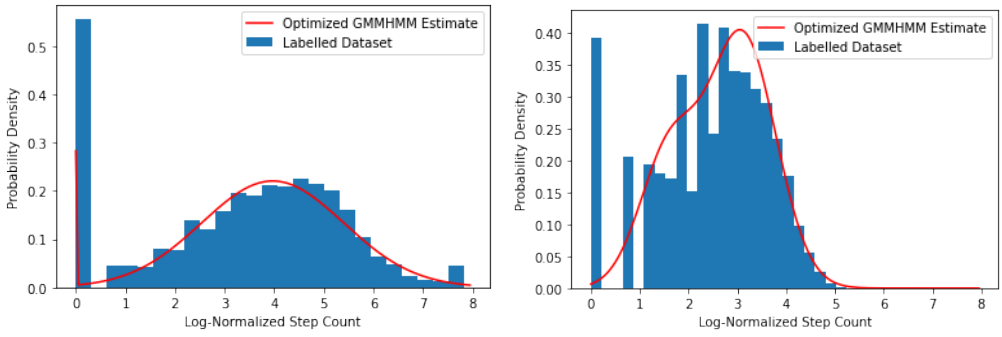
(a) Awake

Parameter	Optimized Value
μ_1	1.61
σ_1^2	0.37
ω_1	0.33
μ_2	3.11
σ_2^2	0.46
ω_2	0.67

(b) Asleep

Table 4: Trained Gaussian Mixture Parameters for Emission Functions

which give emission distribution curves as plotted in figure 19.



(a) Awake

(b) Asleep

Figure 19: Gaussian Mixture Distributions Fitted to Log-Normalized Step Counts

What do these emission distributions tell us? As expected, the emission distribution for wakeful hours roughly follows a normal distribution everywhere, except for at 0, where there is a blow-up of density (corresponding to the spike of the red curve in figure 19a - we apologize for how difficult it is to notice this

spike). This corresponds to periods of time when a student is in lecture, or otherwise in sedentary pose (say, in the SunLab); whereas the rest of the curve corresponds to other day-to-day behaviour: walking around, exercising, and so on. The emission distribution for *sleeping* hours, however, differs quite a lot from our prior estimates. Recall in the data inspection section (section 2.1), we had postulated that the distribution of step-counts during sleeping hours is bimodal, with a spike around 0 and a spike between 3 and 4. It seems that, after training on unlabelled data, the emission distribution is tending towards a unimodal distribution. This suggests that the spike at 0 observed in the labelled data set is not as pronounced in the unlabelled data set - therefore, we should not characterize step-counts during sleeping hours as stratified between motionless sleep and restless sleep. Instead, it is entirely possible that step-count during sleeping hours is unimodal, or that variations derive from different sleep-behaviours (for example, from naps vs. sustained sleep).

This concludes our interpretation of the optimized parameters in the HMM model. Using these parameters for our HMM model, we proceed to use Viterbi's algorithm to yield predicted values for $\mathcal{Y}$ given the observed values $\mathcal{O}$.

3.2.3 Sleep Pattern Analysis

Let us inspect an example of the predictions given by our HMM model when validated against labelled data, comparing it to the given labels. Figure 20 plots the HMM model's predictions for student 0 on the stretch of time between Oct. 5 and Oct. 10.

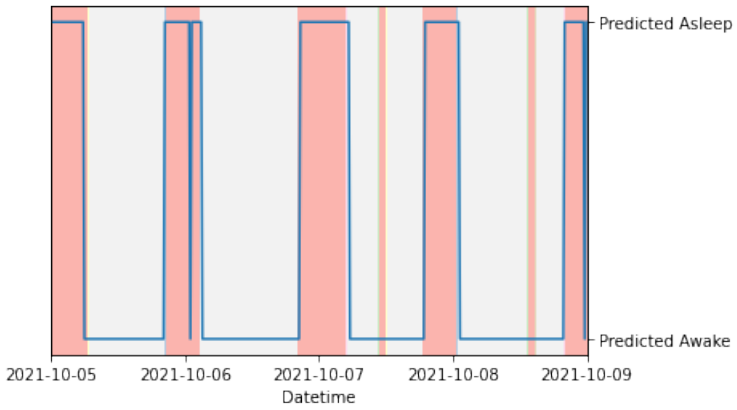


Figure 20: Sample HMM Predictions against Actual Labels
Note: labelled periods of sleep are colored red.

We immediately notice three qualities of these predictions: 1) the HMM model seems to successfully capture many sustained periods of sleep; 2) it seems to erroneously predict short periods of wakefulness in the middle of long stretches of sleep (as at 2021-10-06 in figure 20), and 3) it seems to miss many periods of naps. The first observation is to be expected, since we do have a mean accuracy of 89.6% for the HMM model overall. The second and third observations are discussed more in depth in the following paragraphs.

Consider the distribution of continuous sleep duration, as predicted by the HMM model, compared to that given by the labelled data set. Again, this is intuitively important because sleeping for 8 hours continuously is very different than taking four 2-hour naps. We plot these distributions in figure 21.

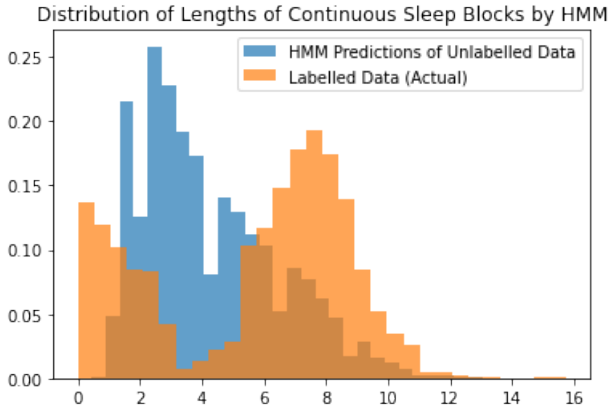


Figure 21: Distribution of Lengths of Continuous Sleep Predicted by HMM

We notice that the two distributions are quite dissimilar; and in particular, the distribution resulting from HMM predictions is unimodal, while the one from labelled data is bimodal. Indeed, inspecting the moments of the distributions as displayed in table 5, we see that the sleep patterns predicted by our HMM model are very different than the ones given by the labelled data.

	Labelled Data	HMM Prediction on Unlabelled Data
$\bar{\mu}$	5.66	4.23
σ^2	9.67	4.80

Table 5: Various Statistics for the Distribution of Lengths of Continuous Sleep

This discrepancy is explained simply as follows: because our HMM model has two hidden states $Q = \{W, S\}$, corresponding simply to wakefulness and sleep, the distribution of the duration of continuous sleep predicted by our model will roughly follow the distribution of time spent in a single state by a Markov chain, which is necessarily unimodal. We must therefore expect that the blue distribution in figure 21 to approximate the orange one as closely as possible, with the constraint that it remains unimodal. This indeed matches what we observe in figure 21. This explains the fact that HMM often erroneously predicts short periods of wakefulness in the middle of long stretches of sleep: only then can it maintain the Markov property of the path of hidden states.

In fact, the third observation, that the HMM model misses many periods of naps, is explained similarly. As the distribution of durations of continuous sleep is unimodal, the likelihood of napping behaviour, according to the model, becomes far too unlikely. Thus, the HMM model makes the sacrifice of labelling naps as wakefulness, in the interest of appropriately labelling sustained sleep.

The solution we propose is to develop an HMM with three hidden states, as described in section 4.3.4. We believe this extra step will be necessary before we can safely draw conclusions about the sleep patterns of the student body using our HMM model, and therefore we focus on the CNN model when we draw general conclusions about sleep patterns in section 4.2. Nevertheless, we believe this proposal is concrete and immediately implementable when given some more development time.

4 Discussion

4.1 So Which Model is Better?

In the previous portions of our paper, we analyzed results and sleep patterns for each model (the CNN and HMM) separately, so we attempt to provide a comparison of the models here. Given that the accuracy of both models far outperform our baseline logistic model (0.896 and .929 accuracy for the HMM and CNN, respectively), the use of either model for detecting sleep patterns is preferable over the logistic regression.

Furthermore, both models have the benefit of being able to implicitly handle the variation of the data collected, a shortcoming of any basic regression model.

Out of the two models, the CNN outperformed the HMM when comparing accuracy of categorization, ROC curves and precision-recall curves. Furthermore, in the CNN, the distributions of both proportion of day spent asleep and length of continuous sleep blocks in the unlabelled data more closely matched the distributions taken from the labelled data in comparison to the HMM. Furthermore, the CNN has the benefit of requiring few prior assumptions; by the nature of neural networks, any patterns picked up during training are a result of only observations in the footstep data rather than any prior bias introduced by a model.

However, the CNN model lacks the benefit of full interpretability. Although the small convolution layers provide some intuition into how one might interpret the decision making process of the CNN, the sheer number of perceptrons in the model introduces a "black box" issue common in many neural networks. Furthermore, the CNN model requires the fine-tuning of hyperparameters and is always prone to overfitting, and issue that had to be consciously considered during the training process. Finally, the CNN model can only be trained on labelled data, making it unable to leverage the larger quantity of unlabelled data available.

Where the CNN model lacks, however, the HMM shines. Although the the HMM had a slightly lower accuracy, as well as distributions which underestimated the distribution of continuous sleeping block lengths, it is simple and easily interpreted. With only 18 trainable parameters, the HMM parameters are each individually interpretable and unlikely to cause any significant overfitting. Because the model has only two states and an explicitly defined relation between footsteps tracked and current sleep state (using the emission distributions B_W and B_S), it is easy to characterize the *causal* relationship between sleep status and activity. Furthermore, analysis of the transition probabilities A allows one to study the sleep state independently of step count. Finally, by virtue of being an unsupervised learning algorithm, it is able to train on unlabelled data, making use of all data available to us. Thus, the HMM model is more interpretable and has more potential for future development, although it is lacking in accuracy compared to the CNN model.

Given the benefits and drawbacks of both the CNN and HMM models, we conclude that the preferable model is highly dependent on the priorities of the model user. If one is looking to use a model whose results can be fully explained at the cost of some accuracy, then our HMM is preferable. However, if one wants optimized results but does not care about model interpretability, then the CNN is ideal.

4.2 Conclusions on Sleep Patterns

Broadly speaking, we can conclude that step counts are highly correlated with sleep state. As expected, low levels of step count activity generally coincide with sleep, while high levels of step count activity generally coincide with wakefulness. This is the fundamental paradigm by which our models are arrived at their predictions

From here we can see by a preliminary examination of conveniently labelled step count data that students tend to spend around a third of their day sleeping. They sleep on average 5.7 hours at a time with extremely high variance, though this is more aptly characterized with one distribution of naps averaging 1.3 hours long, and another distribution of sustained sleep averaging 7.5 hours long. They tend to fall asleep around midnight, with naps scattered throughout the rest of the day.

Both our CNN and HMM closely match true labels when applied to step count input from labelled data, so given the assumptions discussed in 2, we can apply them to step count input from the unlabelled data and say with some confidence that the output closely matches the students' unknown true states at each time point.

The predictions on the unlabelled data display patterns quite similar to the labelled data on all counts. The deviations of note on the part of the CNN are a higher count of short naps, and bedtimes in the early morning; the former in particular is likely a technical artifact, as discussed in 3.1.3. The deviations of note on the part of the HMM are a general under-counting of time spent sleeping, and a unimodal distribution of lengths of sleep centered near 3 hours; both of these are almost certainly artifacts, as discussed in 3.2.3.

Of course, it is difficult to validate our predictions on the unlabelled dataset unless true labels are discovered, but it seems both datasets are quite similar in character.

4.3 Limitations and Future Directions

Though our project produced many positive results, time constraints prevented us from performing an in-depth analysis of the data to a level we originally wanted. As such, we consider some of the limitations present in each of both of our studied models, as well as future directions which would allow us to gain a more nuanced understanding of sleep inference using fitness activity.

4.3.1 Errors in Data Collection

One limitation of our model is that we never explicitly search for/measure potential errors in the provided data. Rather, we assume that the majority of our data collected is "accurate", with only a few outliers. This is quite a large assumption, and one that is not guaranteed to be true. Because sleep patterns are self reported by students, it is quite possible that the fitness data may fall prey to human bias. Additional research into potential sources of bias in self reporting, as well as validation of reported data against more tried and tested methods of sleep detection (such as psychosomnograms) would prove to be beneficial to the work.

Furthermore, our work would likely benefit from additional error analysis in the data exploration stage. Deeper analysis on sleep patterns and fitness patterns, especially in relation to the variance of sleep duration (important for data taken from college students, who are all over the place), would allow us to look for potential outliers in the data, such as people who are moving too much or too little. This could give us a cleaner dataset on which to train both our CNN and HMM models.

4.3.2 Non-Markovian Properties of Sleep

One of the major simplifications made by the HMM model is the assumption that sleep cycles follow a Markovian process, that is, the likelihood of an individual staying asleep or awake is dependent only on their current state and nothing else. This assumption is well known to be false, as a variety of other factors affect sleep such as exhaustion and personal sleeping habits. For instance, a student having already slept 8 hours is much less likely to remain asleep in comparison to a student who has only had 2 hours of sleep. Improving up this assumption would entail considering models which look further into the past rather than only state of the hidden Markov chain. For instance, we could consider HMM using generalized Markov chains, where $P(X_n = x|\,\mathcal{G}_n) = P(X_n = x|\,X_{n-1} = x_{n-1}, ..., X_{n-t} = x_{n-t})$ rather than only considering cases where $P(X_n = x|\,\mathcal{G}_n) = P(X_n = x|\,X_{n-1} = x_{n-1})$. In this way, we allow an underlying Markov process to incorporate more information than just a current state, which might more accurately model sleep cycles.

4.3.3 Non-Stationarity

The HMM model assumes that, within each stretch of continuous wakefulness or sleep, the time series of step-counts is stationary. In other words, if O_t is the step-count at time t, for $t \in \{1, ..., T\}$ where $\{1, ..., T\}$ is a stretch of continuous wakefulness or sleep, then the probability distribution of O_t does not depend on t. This is a strongly simplifying assumption.

Consider, for example, a student with the habit of slowly increasing their activity level throughout the day - say, she attends lectures in the morning, walks around in the afternoon, and goes to the gym in the evening. This is a clear counterexample to the stationarity assumption.

Nevertheless, a past study by Li et.al. seems to be successful even with this stationarity assumption [4], indicating that it is a reasonable simplification to make. We note that the CNN does not make this simplifying assumption, and the HMM model, in any future variant, must necessarily make this assumption.

4.3.4 Naps vs. Sustained Sleep, and Sleep Stages

One of the observations made during the data exploration state of our project which was never fully explored was the bimodal nature of the distribution in sleep duration. We conjecture that the two peaks in the data correspond to two different types so sleeping: napping and sustained sleep. Accounting for these differences provides valuable insight into broader sleeping habits of college students, especially in relation to measurements via fitness trackers. Furthermore, accounting for these different types of sleep separately would allow us to incorporate more nuance into both our CNN and HMM; rather than creating a binary

classifier between sleep and wakefullness based on footstep data, we create a ternary classifier to additionally track type of sleep.

This would provide an especially interesting extension of our HMM; rather than having a state space of simply $Q = \{W, S\}$, we could extend this further to $Q = \{W, S_1, S_2\}$ or even to $Q = \{W_1, ..., W_n, S_1, ..., S_m\}$ where $\{W_i\}$ corresponds to different states of wakefullness and $\{S_i\}$ corresponds to different stages of sleep (e.g. NREM, REM sleep). By increasing the complexity of the HMM, especially by adding more sleep states, we could likely reach much higher levels of accuracy, potentially even beating the CNN.

4.3.5 Levels of Activity

Although both our CNN and HMM handle the fact the activity rates (for example, high intensity workouts versus stationary activity) are highly variable implicitly, it would benefit both models to consider this data feature in a more explicit context. Fitness trackers already track high levels of activity to measure duration of exercise, but it is far less common to study this data in relation to sleeping habits. By further using time series segmentation approaches to learn to differentiate between different levels of activity, we may explicitly look for physical activity patterns that might be predictive of sleep habits in college students. For instance, it is possible that after high levels of activity, the average student will not sleep right away, but rather first engage in low levels of activity. By decoding such patterns explicitly, we are able to demystify the calculations undergone by our CNN and HMM and give greater justification for the models' actions.

4.3.6 A Better CNN

For any neural network, there is always room for improvement, and ours is no different. Beyond the limited data to train our CNN on, our network's structure is still relatively basic. In his paper, Perslev argues that adding encoder and decoder layers to the neural network would create a more robust model by giving the CNN tools to more accurately filter our potential error or white noise introduced into data [6]. Given that labelled data fed into our CNN is known to contain some level of error, the additional structure provided by encoders and decoders would make the CNN more resilient against such noise.

Beyond this, our CNN would benefit from the additional tuning of hyperparameters to increase the accuracy of the model. One method to approach hyperparameter tuning would be to implement a grid searching algorithm, a systematic approach to traversing the state space of network hyperparameters to find the optimal set.

5 Conclusion

The purpose of this paper is to infer the sleep patterns of students at Brown University from only a dataset of step counts, recorded by phones, smartwatches, and fitness trackers. Using two different machine learning approaches, namely, Convolution Neural Networks and Hidden Markov Models, we are capable of predicting the state of a student - awake or asleep - at every time point, significantly outperforming baseline classical and regression models on every selected metric of accuracy. Using these trained models, along with an unlabelled dataset of step counts, we derive key probability distributions to characterize the sleeping habits of college students, including how much of the day they sleep, how long they sleep at a time, and what time in the day they fall asleep. With these models, the BrownWell application can non-invasively extrapolate even more labelled sleep data from step counts *en masse*. We hope these results will give President Paxson and Brown Wellness Center insight into the health of students at Brown University.

References

[1] Roberto Bizzotto, Stefano Zamuner, Giuseppe De Nicolao, Mats O Karlsson, and Roberto Gomeni. Multinomial logistic estimation of markov-chain models for modeling sleep architecture in primary insomnia patients. *Journal of pharmacokinetics and pharmacodynamics*, 37(2):137–155, 2010.

[2] Roger J. Cole, Daniel F. Kripke, William Gruen, Daniel J. Mullaney, and J. Christian Gillin. Automatic Sleep/Wake Identification From Wrist Activity. *Sleep*, 15(5):461–469, 09 1992.

[3] G David Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[4] Xinyue Li, Yunting Zhang, Fan Jiang, and Hongyu Zhao. A novel machine learning unsupervised algorithm for sleep/wake identification using actigraphy. *Chronobiology international*, 37(7):1002–1015, 2020.

[5] Leonel Perez-Atencio, Nicolas Garcia-Aracil, Eduardo Fernandez, Luis C. Barrio, and Juan A. Barios. A four-state markov model of sleep-wakefulness dynamics along light/dark cycle in mice. *PLOS ONE*, 13(1):1–15, 01 2018.

[6] Mathias Perslev, Michael Hejselbak Jensen, Sune Darkner, Poul Jørgen Jennum, and Christian Igel. U-time: A fully convolutional network for time series segmentation applied to sleep staging. *arXiv preprint arXiv:1910.11162*, 2019.

[7] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

Appendix: Code Snippets

Data Grabbing and Processing

```
def grab_labelled_data(data_path, label_begins = 98):
  labelled = pd.read_csv(data_path)
  label_begins = 98

  features = labelled.iloc[:, 2:label_begins]
  labels = labelled.iloc[:, label_begins:]
  ids = labelled.iloc[:, :2]

  labels.columns = [x[10:] for x in labels.columns]

  features = features.join(ids)
  labels = labels.join(ids)

  features.columns.name = 'Time'
  labels.columns.name = 'Time'

  features = features.pivot_table(index = ['Student ID', 'Date'])
  labels = labels.pivot_table(index = ['Student ID', 'Date'])

  features = pd.DataFrame(features.stack()).reset_index()
  labels = pd.DataFrame(labels.stack()).reset_index()

  time_stamps = features['Date'] + " " + features['Time']
  time_stamps = pd.to_datetime(time_stamps)

  features['Datetime'] = time_stamps
  labels['Datetime'] = time_stamps

  features = features.drop(columns = ['Date', 'Time'] )
  labels = labels.drop(columns = ['Date', 'Time'])

  features = features.pivot_table(index = ['Student ID', 'Datetime'])
  labels = labels.pivot_table(index = ['Student ID', 'Datetime'])

  features.columns = ['Steps']
  labels.columns = ['Asleep']

  labelled_data = features.join(labels)

  return labelled_data

def grab_unlabelled_data(data_path):
  unlabelled = pd.read_csv(data_path)
  unlabelled = unlabelled.pivot_table(index = ['Student ID', 'Date'])
  unlabelled.columns.name = 'Time'
  unlabelled = pd.DataFrame(unlabelled.stack())
  unlabelled = unlabelled.reset_index()
  unlabelled['Datetime'] = pd.to_datetime(unlabelled['Date'] + ' ' + unlabelled['Time'])
  unlabelled = unlabelled.drop(columns = ['Date', 'Time'])
  unlabelled = unlabelled.pivot_table(index = ['Student ID', 'Datetime'])
  unlabelled.columns = ['Steps']

  return unlabelled
```

Gaussian Mixture Fitting using Sklearn

```
from sklearn import mixture

clf_sleep = mixture.GaussianMixture(n_components = 2)

# asleep_log_steps is an array containing the log-normalized
# step counts in the labelled dataset, filtered for times
# when the subject is asleep.
clf_sleep.fit(np.array(asleep_log_steps).reshape(-1,1))
```

Logistic Regression using sklearn

```
from sklearn.linear_model import LogisticRegression

labelled_data = grab_labelled_data('./BMCM_steps_sleep.csv')

train_data = pd.DataFrame()

for student_id in student_ids:
  labelled_data_for_student = labelled_data.loc[[student_id]]

  for dt in range(-backward_window, forward_window+1):
    labelled_data_for_student.insert(dt+backward_window, "Steps_{}".format(dt), labelled_data_for_

  labelled_data_for_student = labelled_data_for_student.drop(columns = ['Steps'])
  train_data = train_data.append(labelled_data_for_student)

train_data = train_data.dropna()
X = np.array(train_data.iloc[:, :-1])
y = np.array(train_data.iloc[:, -1])
reg = LogisticRegression().fit(X, y)
```

CNN Model using Keras

```
model = tf.keras.Sequential(
  [
      tf.keras.layers.Reshape(input_shape=(win, ), target_shape=(win, 1)),

      tf.keras.layers.Conv1D(kernel_size=3, filters=3, padding='valid'),
      tf.keras.layers.BatchNormalization(center=True, scale=False),
      tf.keras.layers.LeakyReLU(alpha=0.05),
      #tf.keras.layers.Activation('relu'),

      tf.keras.layers.Conv1D(kernel_size=6, filters=3, padding='valid'),
      tf.keras.layers.BatchNormalization(center=True, scale=False),
      tf.keras.layers.LeakyReLU(alpha=0.05),
      #tf.keras.layers.Activation('relu'),

      #tf.keras.layers.Conv1D(kernel_size=8, filters=24, padding='same'),
      #tf.keras.layers.BatchNormalization(center=True, scale=False),
      #tf.keras.layers.Activation('relu'),

      tf.keras.layers.Flatten(),
```

```
        #tf.keras.layers.Dense(5000, use_bias=False),
        #tf.keras.layers.BatchNormalization(center=True, scale=False),
        #tf.keras.layers.Activation('relu'),

        tf.keras.layers.Dropout(0.1),
        tf.keras.layers.Dense(2, activation='softmax')
    ])

model.compile(optimizer=tf.keras.optimizers.Adam(lr=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy', tf.keras.metrics.AUC(), tf.keras.metrics.AUC(curve='PR')])

BATCH_SIZE = 200
EPOCHS = 10
history = model.fit(x=inputs, y=labels, epochs=EPOCHS, steps_per_epoch=297585//BATCH_SIZE,
                    validation_split=0.25)
```

HMM Model using HMMLearn

```
model = hmm.GMMHMM(n_components = 2, n_mix = 2, init_params = '', params = 'stmcw')

model.startprob_ = np.array([0.64, 0.36])
model.transmat_ = np.array([[0.98, 0.02],
                            [0.04, 0.96]])
model.means_ = np.array([[[4.023],
                          [0]],
                         [[1.291],
                          [3.097]]])
model.covars_ = np.array([[[2.157],
                           [0.0001]],
                          [[0.879],
                           [0.485]]])
model.weights_ = np.array([
                           [0.826, 0.174],
                           [0.370, 0.630]
])
```