Shedding Light on Mouse Sleep: Automated Sleep Phase Detection using Machine Learning

Lars Barmettler*, Matthieu Burguburu† and Alexandre Piveteau‡
École Polytechnique Fédérale de Lausanne

*Email: *lars.barmettler@epfl.ch,†matthieu.burguburu@epfl.ch,‡alexandre.piveteau@epfl.ch*

***Abstract*—In this report, we use a dataset recorded by the Center for Integrative Genomics at UNIL to infer the sleep phase of mice from miscellaneous genetic strains and evaluate the performance of our approach in this classification problem. We use Electroencephalogram (EEG) and Electromyogram (EMG) signals to differentiate between wake, NREM and REM sleep phases via random forests and neural networks. Our highest classification accuracy for a single individual, 93%, is obtained using a random forest model, whereas our highest classification precision for a cross-individual model is 95%.**

I. Introduction

Mouse sleep can be decomposed into several phases. *rapid eye movement sleep* (*REM sleep*), also called *paradoxal sleep*, is a phase of sleep characterized by rapid eye movements and is associated with the propensity to dream vividly and with reduced muscle tone. On the other hand, *non-rapid eye movement sleep* (*NREM sleep*), or *quiescent sleep*, includes the phases of sleep characterized by the absence or reduction of eye movements and the absence of muscle paralysis.

In this project, we evaluated and implemented some machine learning methods to help the Institute for Information and Communication Technologies (IICT) of HEIG-VD to automatically classify sleep phases in individuals mostly based on the BXD panel, a population of recombinant inbred lines of mice. This data was recorded by the Center for Integrative Genomics (CIG) at UNIL in the context of studies on sleep regulation in mice [1].

The rest of this report is organized as follows. Section II explores the data set, its characteristics, and the preprocessing steps we applied. We then discuss the feature engineering we performed in section III and candidate models in section IV. We evaluate and quantify the performance of our classification model in section V, and discuss our results in section VI. Finally, section VII concludes our report.

The source code and models to reproduce our experiments is available on GitHub [2].

II. Data exploration

The data provided by the Center for Integrative Genomics consists of a set of sleep phase recordings of mice, *electroencephalogram* (*EEG*) and *electromyogram* (*EMG*) measurements of individuals, available both as `.raw` files (containing raw recorded data) or as `.smo` files (containing pre-processed data). One file corresponds to a different individual in the study.

The overall dataset consists in total of 255 mice. Each individual belongs to one of 42 different breeds. The number of mice in each breed varies between 2 and 12 .

Each file holds a collection of $86'400$ epoch records of 4 seconds. The epoch records are consecutive, and the individuals were followed for exactly 4 days. On day 3 of the experiment, mice were gently disturbed to keep them awake, typically by placing tissues in their cages or changing their litter. Each epoch is composed of the following information (fig. I): the sleep phase labelled by a technician (which may be either *wake*, *NREM sleep*, *REM sleep*, or numerical values if an artefact was observed), the EEG variance, the EMG variance, and the body temperature (always set to $0.0\,°\mathrm{C}$, since it wasn't recorded in the study). In addition, the EEG power density of the full DFT spectrum of the epoch is provided for the range $0.00\,\mathrm{Hz}$ to $100\,\mathrm{Hz}$, in the form of 401 values with a resolution of $0.25\,\mathrm{Hz}$ [3].

In this data set, the different classes are not evenly distributed. For instance, in the `c57bl6` line, the wake class contains 56.4% of the measures, while the NREM class contains 39.3% of the measures and the REM class contains 4.3% of the measures.

III. Feature engineering

Due to the fact that the EEG and EMG measurements are cyclical, we can exploit their frequencies to compute new features that may be useful for the model. Additionally, we can utilize standard techniques such as adding non-

Feature name	Description
Sleep phase	w (wake), n (NREM), r (REM), or digits for artefacts
Temperature	Always $0.0\,°\mathrm{C}$
EMGv	EMG variance over 1 epoch
EEGv	EEG variance over 1 epoch
$\mathrm{Bin}_{i\in(0,400)}$	DFT power density for a frequency of $i \times 0.25\,\mathrm{Hz}$

Table I
FEATURES IN A PROCESSED `.SMO` FILE

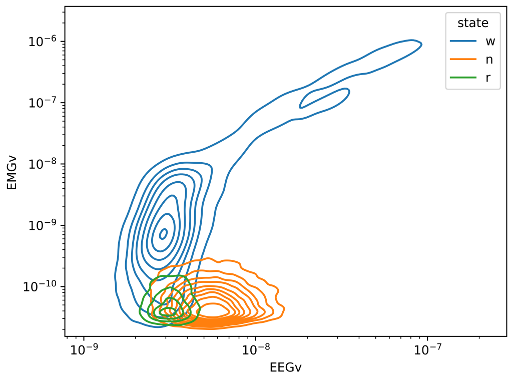

Figure 1. Distribution of sleep phases using `10101`'s EEGv and EMGv over the days 1 to 4

linearity or aggregating historical values to further enhance the feature space and improve the model's performance. The sections III-A to III-F follow the order of the applied feature engineering pipeline.

A. *Exploiting the EEG frequency spectrum*

As shown in figure 1, it is clear that the NREM and REM classes overlap significantly, with the variance of EEG and EMG not being sufficient to distinguish them. However, we can still exploit the regularity of the EEG signal in order to differentiate the classes. Specifically, the REM state is characterized by a much more regular signal than the NREM state. On the other hand, the NREM signals are closer to the awake state than the REM state.

To better differentiate between the NREM and REM states, we can define four features in which we use the 401 Discrete Fourier Transform power spectrum values and evaluate their regularity. These features, which are commonly used in the analysis of EEG signals [4], have roots in audio processing techniques [5].

In equations eqs. (1) to (4), N refers to the total count of frequency bins (which are assumed to be 0-indexed, and has value 401 in the CIG dataset), $x(n)$ to the value of the x-th bin, and $f(x)$ to the frequency of the x-th bin.

1) Spectral flatness measure: The spectral flatness (eq. 1) measures the ratio between the geometric mean of the spectrum and its arithmetic mean and is a good indicator of the noisiness of the signal. A signal with a high peak will tend to generate a lower spectral flatness, whereas a noisy signal will generate a higher spectral flatness value.

$$F = \frac{\sqrt[N]{\prod_{n=0}^{N-1} x(n)}}{\frac{\sum_{n=0}^{N-1} x(n)}{N}} \tag{1}$$

In the context of sleep phase detection, spectral flatness is a good differentiator between REM and NREM sleep, since the EEG during REM sleep is less noisy (fig. 2).

2) Spectral roll-off: Spectral roll-off (eq. 2) indicates the width of a signal, by measuring the number of frequency bins under which a percentage p of the total power exists. A noisy signal will generally produce a homogeneous roll-off.

$$SR(p) = n \text{ such that } \sum_{i=0}^{n} |x(i)| = p \sum_{i=0}^{N-1} |x(i)| \tag{2}$$

In this project, we split our roll-off into 10 equally sized percentile groups.

3) Spectral centroid: The spectral centroid (eq. 3) is the ratio between the weighted sum of the frequencies and their unweighted sum, and can be interpreted as the ”center of mass” of the frequency spectrum.

$$C = \frac{\sum_{n=0}^{N-1} f(x)x(n)}{\sum_{n=0}^{N-1} x(n)} \tag{3}$$

4) Spectral entropy: Similarly to the spectral flatness measure (eq. 1), the spectral entropy (eq. 4) is an indicator of the peakiness and the disorganization of the signal.

$$SE = \sum p_i \log_2(p_i) \tag{4}$$

The spectral entropy is not computed directly using our DFT power spectrum. First, the power spectral density must be normalized in the interval $(0, 1)$ so that it can be treated as a probability density function (PDF). Finally, the entropy can be computed from this density function. Here, p_i refers to the value of the PDF for the i-th bin, which are summed over all the bins.

B. *Aggregating measurements*

To classify sleep phases in mice, it is important to note that while EEG and EMG tend to exhibit similar shapes across individuals, they often do so at different scales.

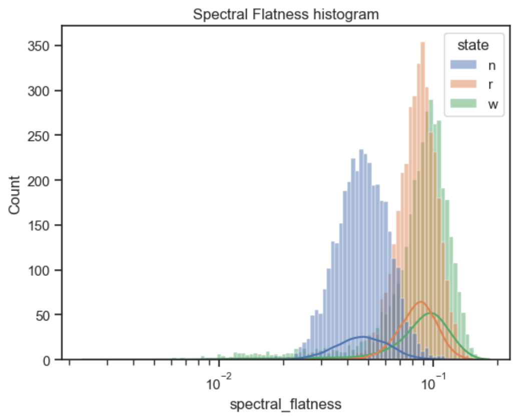

Figure 2. Distribution of the Spectral Flatness Measure of the rebalanced labelled samples from `10101.smo`

Therefore, in order to properly inform the model, we must design features that integrate historical information, allowing the model to not only understand the absolute values of the measurements but also how they relate to the individual.

To do this, we implemented a rolling window scheme to expand each measurement with past and future historical information. Specifically, we took the arithmetic mean, the median, the variance, the maximum and the minimum of both EMGv and EEGv from the past and the future, using windows of various sizes (2, 5, 10, 20, 50, and 100) that represent durations ranging from 8 seconds to 6 minutes and 40 seconds.

C. *Eliminating outliers*

It is essential to address the issue of the interferences and abnormal measurements due to mice moving and shaking the EEG and EMG measuring devices. This potentially creates spikes in the signals, which result in abnormally high EEGv and EMGv values. To address this problem, we implemented a simple outlier elimination scheme: we consider the 99th percentile of the values of the data points and drop any outliers. This helps to ensure the accuracy and reliability of the measurements.

D. *Adding non-linearity*

To ensure the non-linearity of our feature set, we computed the $\log$ values of the EMGv and the EEGv. Additionally, we included polynomial feature expansion of degree 3 for EMGv and EEGv. To further enhance the feature space, we also included a bias term with a value of 1 to each measurement.

E. *Rebalancing*

During the analysis of the data set, we identified several biases that could potentially impact the results. For example, not all of the mice in the data set come from the same line, and some lines are more heavily represented than others. For instance, the line `c57bl6` includes 12 individuals, while the `BXD101` line only includes 2 individuals. In addition to this, the data set is not balanced across the different sleep phase classes, with 56.4% of the epoch samples corresponding to wake periods for the line `c57bl6` , 4.3% corresponding to REM sleep, and 39.3% corresponding to NREM sleep.

After considering the potential impact of mice lines on the model performance, we ultimately decided not to take any action to rebalance the lines. However, we rebalanced the sleep phase classes by using random resampling.

F. *Standardization*

To ensure that the features are properly standardized, we use SciKit's `StandardScaler` [6]. This process is performed after feature expansion and involves subtracting the mean and dividing by the standard deviation. In order to prepare the scaler, we first use the `StandardScaler` to get the mean and the standard deviation from the training data set. Then we apply the rescaling to the training and testing data sets to ensure consistency.

IV. Model

We compared several machine learning models to make our predictions: the Random forest method (sec. IV-A), and a simple neural network (sec. IV-B). We chose these models for their simplicity and effectiveness, as well as low computational requirements.

A. *Random forest*

To evaluate the performance of our model, we employed Sci-Kit Learn's RandomForestClassifier [6], an easy-to-use and efficient implementation of the method. To determine the quality of the splits within the random forest, we used the default Gini impurity as our criterion. Additionally, we allowed for unlimited tree depth and kept the minimum required number of samples for a split at 2.

B. *Neural network*

We used a Sequential neural network from the python library Keras [7]. The network was trained with a single hidden layer consisting of 3 neurons, using the rectified linear unit (ReLU) activation function. The configuration was obtained through manual exploration of the hyperparameters. The output layer applied the softmax activation function, which is commonly used for classification tasks. During training, we employed categorical cross-entropy loss and the Adam optimization algorithm [8] with a learning rate of 1e-4. The batch size was set to 64 and the training process occurred over 150 epochs.

V. Evaluation

The IICT lab at HEIG-VD and the CIG at UNIL identified three main experiments that could be conducted to evaluate the model's performance. The first experiment involves making predictions on a mouse using training data from the same mouse (sec. V-A), with the intended use case being that a technician could annotate a subset of the data points (such as the first day of the study) and the model would then infer the results for the remaining three days. The second experiment involves making predictions on a mouse in a specific genetic line using training data from other mice within the same genetic line (sec. V-B), with the goal of using the current data set for future studies. The third and final experiment involves making predictions across genetic lines (sec. V-C), with the goal of making the model agnostic to the specific mouse line.

A. *Training and testing on the same individual*

For the first experiment, we selected a single mouse from line `c57bl6` with identifier `BL601`. The model was then trained on the samples corresponding to the first day, which totalled 21'600 samples. To evaluate the model's performance, we used the samples corresponding to days 2, 3, and 4, for a total of 63'000 data points.

Table II summarizes the results of this experiment.

Model	Accuracy	Weighted Precision	Weighted F1-Score
Neural Network	0.92	0.95	0.93
Random Forest	0.93	0.94	0.93

Table II
PERFORMANCE ON SINGLE MOUSE `BL601`, (EXP. V-A)

Model	Accuracy	Weighted Precision	Weighted F1-Score
Neural Network	0.93	0.96	0.94
Random Forest	0.95	0.96	0.95

Table III
MODEL PERFORMANCE ON SINGLE LINE `c57bl6` (EXP. V-B)

Model	Accuracy	Weighted Precision	Weighted F1-Score
Neural Network	0.79	0.93	0.83
Random Forest	0.88	0.93	0.89

Table IV
MODEL PERFORMANCE FOR MULTIPLE LINES (EXP. V-C)

B. *Training and testing within a single genetic mouse line*

In the second experiment, we selected the line of mice of the `c57bl6` line. We trained the model on the samples of 11 individuals and tested it on mice `BL6V3`.

The results of this evaluation can be found in table III, and the associated confusion matrix can be found in figure 3.

C. *Cross-line training and testing*

For the third experiment, we selected 15 individuals to train the model on individuals from different genetic lines. We then tested it on `05101` and obtained the scores from table IV.

VI. DISCUSSION

It may seem intuitive that the three experiments would be in increasing order of difficulty due to the increasing generalization requirements. However, our results showed that this was not the case. Specifically, experiment V-B reached an accuracy of 0.95, while experiment V-A only achieved an accuracy of 0.93. We suspect that this discrepancy is due to the size of the training set in the first experiment, which is limited to a single day for a single mouse. This may not

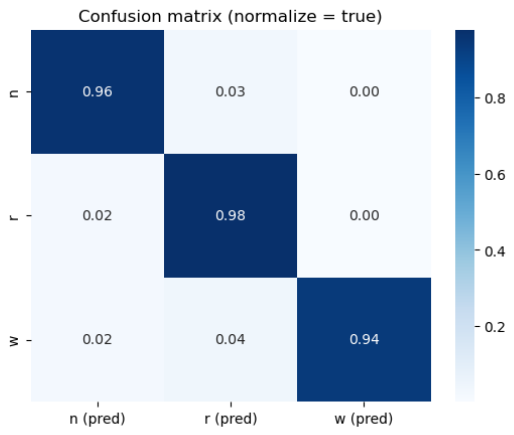

Figure 3. Random forest confusion matrix for experiment V-B

provide enough data for the model to effectively learn and generalize to the testing set.

In experiments V-A, V-B, and V-C, we observed that the random forest method consistently outperformed a simple neural network. Interestingly, this gap seemed to widen when we introduced some EEG signal frequency-related features, such as the spectral roll-off. This suggests that the random forest method may be particularly well-suited for this type of data and may be able to more effectively capture the complex patterns and relationships present in the data.

VII. CONCLUSION

In this work, we used a random forest method and a neural network to classify the sleep phases of mice. When making predictions on a single mouse using training data from the same mouse, we achieved an accuracy of 0.93. When using training data from other mice in the same genetic line, we obtained an accuracy of 0.95. Finally, when using training data from multiple genetic lines, the accuracy was 0.88. These results demonstrate the potential of using machine learning techniques to classify sleep phases in mice and highlight the importance of considering the specific characteristics of the data when selecting a model.

There are several potential avenues for future work in this area. One promising approach would be to use a 1-D convolutional neural network, which has shown to be effective in processing sequential data such as time series. Additionally, there are many other spectral features that could be implemented in the model to potentially improve its performance. Using a tool such as KerasTuner [7] to search for optimal hyperparameters could also be beneficial in finding the best configuration for the model.

References

[1] S. Diessler, M. Jan, Y. Emmenegger, N. Guex, B. Middleton, D. J. Skene, M. Ibberson, F. Burdet, L. Götz, M. Pagni, M. Sankar, R. Liechti, C. N. Hor, I. Xenarios, and P. Franken, "A systems genetics resource and analysis of sleep regulation in the mouse," *PLOS Biology*, vol. 16, no. 8, p. e2005750, Aug. 2018, publisher: Public Library of Science. [Online]. Available: https://journals.plos.org/plosbiology/article?id=10.1371/journal.pbio.2005750

[2] A. Piveteau, M. Burguburu, and L. Barmettler, "Shedding Light on Mouse Sleep - Source code," 2022. [Online]. Available: https://github.com/epfl-ML/project2-code

[3] M. Jan, N. Gobet, S. Diessler, P. Franken, and I. Xenarios, "A multi-omics digital research object for the genetics of sleep regulation," *Scientific Data*, vol. 6, no. 1, p. 258, Oct. 2019, number: 1 Publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41597-019-0171-x

[4] A. Frid and L. M. Manevitz, "Features and Machine Learning for Correlating and Classifying between Brain Areas and Dyslexia," Jan. 2019, arXiv:1812.10622 [cs, q-bio, stat]. [Online]. Available: http://arxiv.org/abs/1812.10622

[5] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the CUIDADO project," Apr. 2004. [Online]. Available: http://recherche.ircam.fr/anasyn/peeters/ARTICLES/Peeters_2003_cuidadoaudiofeatures.pdf

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.org/papers/v12/pedregosa11a.html

[7] T. O'Malley, E. Bursztein, C. Long, F. Chollet, H. Jin, and L. Invernizzi, "Keras Tuner," 2019. [Online]. Available: https://github.com/keras-team/keras-tuner

[8] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017, arXiv:1412.6980 [cs]. [Online]. Available: http://arxiv.org/abs/1412.6980