Detection of Traumatic Brain Injury Using Single Channel Electroencephalogram in Mice

A. Sutandi[1], N. Dhillon[1], M. Lim[2,3], H. Cao[4,5] and D. Si[1]

1. Computing and Software Systems, University of Washington, Bothell, WA, USA
2. VA Portland Health Care System, Portland, OR, USA
3. Dept. of Neurology, Oregon Health & Science University, Portland, OR, USA
4. Dept. of Electrical Engineering and Computer Science, University of California, Irvine, CA, USA
5. Dept. of Biomedical Engineering, University of California, Irvine, California, USA
{sutandia, navjodh, dongsi}@uw.edu, lmir@ohsu.edu, hungcao@uci.edu

Abstract— Preclinical studies of traumatic brain injury (TBI) are often performed using a murine model of mild traumatic brain injury (mTBI) due to highly controlled settings and high reproducibility in this experimental model, compared to studies of human TBI. We have previously demonstrated persistent changes in the sleep wake cycle using a widely accepted mouse model of mTBI. The gold standard of sleep wake assessment is achieved by recording brain electroencephalogram (EEG), which not only allows for standard sleep staging but also allows further signal processing through quantitative EEG methods. Conventional methods of sleep staging require manual scoring by a trained expert. Here, a 1-D deep convolutional neural network (Deep CNN) is proposed to automatically score sleep stages and identify mTBI from a single-channel EEG signal with duration of 64 seconds by classifying the EEG signal into one of four classes: sham (control) wake, sham (control) sleep, mTBI wake, and mTBI sleep. We demonstrated that the proposed Deep CNN has the ability to learn features to classify the target classes. Deployment of the trained model on Raspberry Pi further indicates the capacity to perform classification in real time and mobile applications. Thus, the proposed system has the potential to provide a low-cost and fast method for detection of TBI in individuals.

I. Introduction

Traumatic brain injury (TBI) contributes significantly to death and disability in the United States (U.S.). According to the U.S. Centers for Disease Control and Prevention (CDC), 155 people die per day from injuries that include TBI [1]. One of the major sequelae of TBI is persistent sleep-wake dysfunction, which can significantly contribute to cognitive impairment and disability. The exact mechanisms underlying persistent changes in sleep are still unclear, but our group and others have implicated long-term changes in neurotransmission [2].

A well-established method to study TBI is to create mouse models using the fluid percussion injury (FPI) method [3]. This particular method shows high reliability and reproducibility, can be graded in severity, and recapitulates important features of human mild TBI including both neuropathology and behavioral deficits [4]. To more closely examine quantitative sleep EEG after FPI in mice, electroencephalogram (EEG) signals from adult male mTBI and control mice who were subjected to sham surgery procedures (e.g., anesthesia, craniotomy, but not the fluid pressure pulse) underwent secondary analysis from a previously published study [3]. Recently, machine-learning algorithms have been investigated to classify mTBI from mice using signal processing of EEG of varying epoch duration between 1 – 4 minutes with promising accuracy up to 92% [5][6]. The investigation reviewed several classification models and concluded that convolutional neural network (CNN) was the most reliable with the lowest average of variances among all cross-validation experiments for each classifier model. This study provided the motivation herein to further explore CNN-based models.

In this work, we propose and implement a CNN-based deep learning system to 1) stage sleep, and 2) detect the presence of mTBI in mice using a single-channel EEG that is deployable on a Raspberry Pi. The system enables real-time analysis of EEG data using a relatively inexpensive and portable hardware. The system takes short EEG epochs of 64 s in duration and assigns a class from four classes: sham wake, sham sleep, mTBI wake, and mTBI sleep. An epoch size of 64 s is optimal as it incorporates a sufficient number of distinguishing patterns for accurate and reliable classification, and at the same time is small enough to allow for fast prediction and deployment on an embedded system such as a Raspberry Pi. Using supervised learning from labeled data, the proposed system was able to learn the patterns required to classify EEG epochs into the target classes. We also successfully demonstrated that our system was able to perform a prediction of a 64 s EEG epoch in the order of seconds on a Raspberry Pi with results identical to those obtained on a regular computer, which may eventually enable TBI detection in daily life.

II. Related Works

Study of brain activity using electroencephalogram (EEG) typically involves extracting information from very noisy signals associated with certain activities, which is commonly referred to as quantitative EEG (qEEG). We and others have applied qEEG methods to the study of TBI and related disorders [2], [7]-[9]. In summary, various methods can be categorized into

spectral analysis, functional connectivity measures, discriminant functions, event-related potentials (ERPs), and advanced signal processing. One of the main conclusions of the survey was that current qEEG methods provide an imperfect assessment of mTBI. The survey authors also recommended that more advanced mathematical methods should be explored.

In recent years, machine-learning (ML) techniques have been applied for classification of mTBI due to their ability to extract complex and typically non-linear pattern from data. A literature survey [10] showed that ML techniques have great potential to improve prediction performance from traditional qEEG methods. Most of the work surveyed used rule-based techniques, such as k-nearest neighbors, random forest, etc. In our proposed system, we used a deep-learning technique, convolutional neural network (CNN), as its performance was demonstrated to be better than rule-based techniques [5][6].

Machine-learning techniques have been used to successfully classify sleep stages of humans and rodents (rats and mice). In [11], sleep stages of rats were classified from EEG signals using normalized powers of sub-bands and the k-nearest neighbor (k-NN). The classification accuracy of the method was 95.43%. Another study used similar extracted features with Naïve Bayes classifier to classify sleep stages of mice and rats from single-channel EEG signals and achieved 93% accuracy [12]. More recently, researchers have reported accuracy of 87% in 5-class sleep stage prediction using CNN with single-channel EEG epoch with 30 s duration [13]. In our system, we used longer duration of 64 s to perform 2-class sleep stage prediction (wake or sleep) and sham (control) versus mTBI prediction.

The Neuroberry platform [14] uses a Raspberry Pi 2 device to capture EEG signals but the focus is on making the EEG signal readily available in the Internet of Things (IoT) domain, rather than EEG signal classification itself. The Acute Ischemic Stroke Identification System [15] utilizes a Raspberry Pi 3 device with an Analog to Digital Converter (ADC) front-end to capture raw EEG signals but the signal analysis and processing is done on a regular computer using MATLAB. This system does not perform signal classification. [16] describe a system created using Raspberry Pi 3 with deep learning to perform EEG signal classification that is used to control wheel-chair movement, but the focus is on creating a standalone wheelchair control system and not on the variation of training hyper-parameters on the model and its impact on classification metrics. In addition, it does not perform TBI/non-TBI classification.

In this work, we trained a deep-learning model on a high-performance computer (HPC) using GPUs and then deployed the trained model on Raspberry Pi 4. The small size, low cost, and classification accuracy being identical to an HPC makes it feasible for deployment in portable systems that can operate standalone. To our best knowledge, no standalone, portable system has yet been created using Raspberry Pi that classifies TBI/non-TBI and sleep/wake states from EEG signals.

III. Methods

The multi-class classification system follows typical methodology for analyzing EEG recordings, where features are extracted from EEG and then analyzed. The proposed system uses a machine learning model that is trained from data to extract optimum features for differentiating the classes. The following subsections describe the proposed system architecture, dataset used to train the model used by the system, training process, and deployment on Raspberry Pi.

A. System Description

Figure 1 shows the architecture of the proposed system to classify a single-channel EEG epoch. The input is a single-channel EEG epoch of 64 s duration with 256 Hz sampling frequency. The EEG epoch is then filtered and down-sampled by 4 before being sent to a deep convolutional neural network (CNN). This signal preprocessing step reduces the number of deep CNN parameters, resulting in shorter training time and lower hardware memory requirements. The deep CNN extracts features from the preprocessed EEG epoch and predicts a class for the epoch.

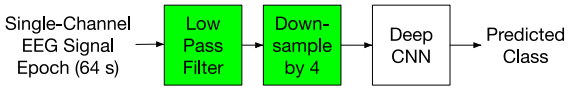

Figure 1. Proposed system architecture.

The proposed Deep CNN approach is based on a deep neural network used in speaker identification system.[1] The architecture is shown in Figure 2. Conv1D is 1-dimensional (1D) convolution layer that convolves its input with multiple filters to produce 1D feature maps. Rectified Linear Unit (ReLU) activation is applied at its outputs. Batch Normalization layer normalizes and scale inputs so that mean is close to 0 and standard deviation is close to 1. The transformation is applied on all inputs in a training batch. Spatial Dropout 1D drops entire 1D feature maps at specified rate to promote independence between feature maps. MaxPool1D down-samples its input by taking the maximum value over the time window defined by pool_size and shifting this time

[1] https://github.com/oscarknagg/voicemap

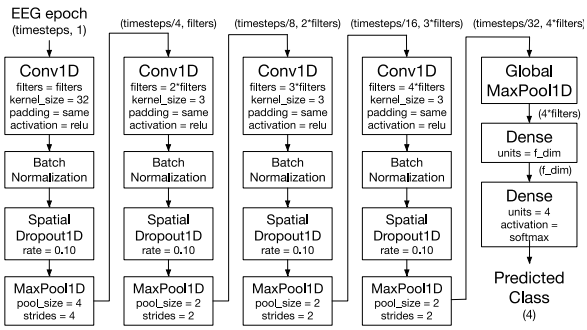

Figure 2. Deep CNN architecture.

window by strides. Global MaxPool1D down-samples its input by taking maximum value over the time dimension. Dense layer following Global MaxPool1D is a regular densely connected neural network layer producing extracted features of dimension f_dim. The final Dense layer performs the class prediction. Its 4-dimensional output goes through a softmax activation function where the most likely class is the dimension with highest value.

Deep CNN in the proposed system was trained using supervised learning technique. In supervised learning, the deep CNN was trained using data set with known states, i.e., labeled data to learn the features that identify the classes.

B. Labeled Data Creation

Mice EEG recordings were acquired as part of a previously published dataset [3]. The dataset comprised 11 mice divided into 2 groups: 5 mice in mTBI group and 6 mice in sham group. The mice in mTBI group received a surgery to implant probes and induce mTBI via fluid percussion injury (FPI), while the mice in sham group received an operation to implant probes only. EEG recordings were obtained ~2 weeks after surgery including 5 days of baseline acclimation to the novel recording environment. The EEG data was sampled at 256 Hz, i.e., 22,118,400 timesteps per mouse over a 24-hour recording period. Each 4-second epoch was scored for wake, non-rapid-eye movement (NREM), or rapid-eye-movement (REM) stage by a human expert and the staging was verified by another human expert as described in [3]. Sleep stages were collapsed such that NREM and REM were both labeled as "sleep" and wake remained distinct. We combined REM and NREM sleep stages because the consecutive durations of REM stages in our data set were not significant and the number of REM data samples was too small to significantly improve the model classification accuracy. Further, this allowed us to reduce the number of classes from, 6 to 4, which reduced the overall model complexity.

Each 24-hour EEG data was broken into non-overlapping (in time) epochs with an epoch duration of 64 s made up by consecutive labeled-epochs, i.e., consecutive 4 s epochs with stages scored by human expert. Labels were assigned for 64 s epochs when all 4 s labels for each 64 s epoch were the same. All 64 s epochs with mixed 4 s labels were dropped from the set. This was done to ensure the 64 s labels are valid. For evaluation of the proposed system, epoch durations of 16 s and 32 s were generated as well using same method.

C. Training Method

The Deep CNN was trained with two different data arrangements to evaluate pattern detection capability and trained model generality as explained below. All trainings were done for 50 epochs using Adam optimizer with default parameters as implemented in Keras with TensorFlow backend. Different values of epoch widths, number of filters (filters in Figure 2), and extracted feature dimension (f_dim in Figure 2) were used to investigate their effects on the performance of the system.

1) *Random-Sampling (RS) Data Arrangement:* The goal of training with RS data arrangement was to evaluate the capability of Deep CNN to learn the features to classify the target classes. The RS data arrangement assumed that all mice were identical and that all EEG epochs were independent. In this arrangement, a training dataset was assembled by picking randomly 80% of available epochs for each sleep/wake stage from each mouse. The remaining data not picked for training made up the testing dataset.

2) Species-Aware (SA) Data Arrangement: The goal of training with SA data arrangement was to evaluate the trained model generality in predicting EEG epochs from a new mouse. The SA data arrangement assumed that all mice were not identical and EEG epochs from a mouse had features that were not found in other mice. In this arrangement, a training dataset was assembled by picking all available epochs from 8 mice (4 mTBI and 4 sham). A testing dataset was assembled by picking all available epochs from 2 mice (1 mTBI and 1 sham) not picked for training dataset. Five different sets (commonly referred to as folds) of training and testing data were used to perform cross-validation of the model performance.

D. System Evaluation

The performance of the system was evaluated by metrics typically used for a multi-class classifier system: accuracy, precision for each class, recall for each class, and F1 score (or F-measure) for each class. The definitions are as follows

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$precision = \frac{TP}{TP+FP} \quad (2)$$

$$recall = \frac{TP}{TP+FN} \quad (3)$$

$$F_1 = 2\frac{precision \times recall}{precision+recall} \quad (4)$$

where TP is the number of correct prediction (true positive), TN is the number of correct rejection (true negative), FP is the number of incorrect prediction (false positive), and FN is the number of incorrect rejection (false negative).

E. System Deployment

The previously described model was deployed on Raspberry Pi 4 device. This demonstrates that the model and techniques developed here can be used on a real-life low-cost system for use in practical scenarios. This opens the possibility of creating a highly complex model using computing power available on more capable systems such as workstations with multi-core CPUs or GPUs and deploying it on a compact portable device. We demonstrate that:

1) The model can be trained on a system completely different from Raspberry Pi in terms of computer architecture and processing speed.

2) The model can be run on Raspberry Pi with accuracy within 0.001 (see results section for details) compared to a regular computer.

To deploy the model, we first exported the model to a Hierarchical Data Format version 5 (HDF5) file containing the model and model metadata. We then transferred and loaded the model on Raspberry Pi. We used one of the raw EEG signal data files corresponding to a TBI affected mouse to verify classification accuracy.

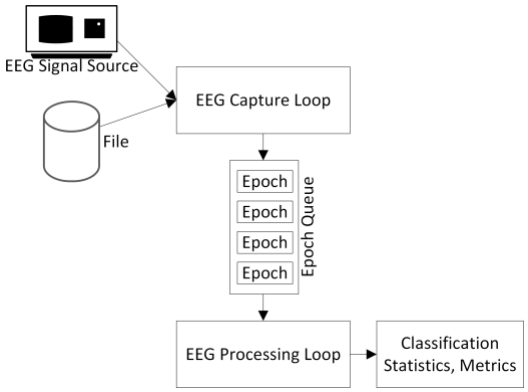

Figure 3. A buffered, queue-based design for capturing and processing EEG epochs.

We simulated a real-life system by repeatedly loading a limited number of epochs from one EEG data file and verified that the classification model provides valid output labels. In terms of timing, the time needed by the Raspberry Pi system to process a fixed number of epochs (processing includes EEG epoch collection, feature extraction and classification tasks) was significantly smaller than the time required to collect EEG epochs at 256 Hz sampling rate using a 64 s epoch window size. Details on timing related to EEG epoch collection and processing are shown in Table 1. This confirms that this system will have sufficient time to process EEG data samples collected at practical EEG signal sampling rates. Further, because we use a queue-based design (Figure 3) with separate threads for epoch collection and processing, incoming epochs are not lost while previously collected epochs are being processed. In the current system, we load epochs from an EEG signal data file, but the system operates identically to a system that captures EEG signal data points from a hardware Analog to Digital Converter (ADC) would. This demonstrates that in future, a complete hardware-based system can be created that can capture a live EEG signal and classify it on the fly. Such a system can be adapted for practical field use given its low cost and compact form factor.

The implemented system was also capable of displaying live, the label count, epoch processing time and distribution of classified labels via a histogram (Figure 4). Having a system that can provide a running view of labels distribution based on EEG signal eliminates the need to capture and store EEG signals separately and enables a quicker conclusion on whether a subject is afflicted with TBI or not.

Table 1. Epoch collection and processing timing details (64 s epoch window size)

Number of Epochs	Epoch collection time (s)	Epoch processing time (s)
1	64	0.14
10	640	0.82
100	6400	7.93

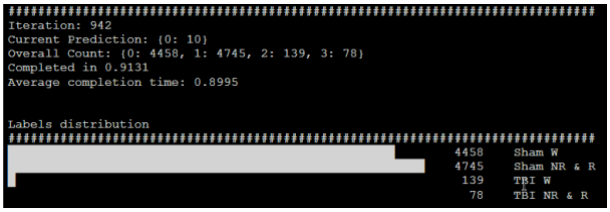

Figure 4. Live classification in progress for the Raspberry Pi based system (epoch batch size: 10).

IV. RESULTS

The following subsections describe the evaluation results of the proposed system. The dependence on system performance on width of EEG epoch, number of filters, and feature dimension is presented, followed by results from Raspberry Pi deployment.

A. *Varying EEG Epoch Width*

One of the design choices made was to limit the EEG epoch width to 64 s or under to ensure that the system can provide quick response when used in a portable device like Raspberry Pi. Using shorter EEG epoch width than 64 s was desired for quicker response, but performance of the system as measured by the metrics degraded as EEG epoch width decreased (see Table 2). For more constrained hardware deployment, EEG epoch width of 32 s may offer a reasonable trade-off between prediction performance and hardware requirements.

Table 2. System performance trained using RS data arrangement using various EEG epoch widths. Parameters filters and f_dim in Figure 2 were set to 128 and 4, respectively. Values in bold are best values.

	Epoch Width		
	16 s	**32 s**	**64 s**
Accuracy	0.742	0.788	**0.815**
Sham Wake			
Precision	0.847	0.889	**0.911**
Recall	0.804	0.821	**0.877**
F_1	0.825	0.853	**0.894**
Sham Sleep			
Precision	0.637	**0.693**	0.670
Recall	0.780	0.787	**0.859**
F_1	0.701	0.737	**0.753**
mTBI Wake			
Precision	0.790	0.776	**0.805**
Recall	0.681	**0.810**	0.805
F_1	0.731	0.793	**0.805**
mTBI Sleep			
Precision	0.679	0.752	**0.827**
Recall	0.666	**0.719**	0.675
F_1	0.672	0.735	**0.743**

B. *Varying Number of Filters and Feature Dimension*

The number of filters (filters in Figure 2) and feature dimension (f_dim in Figure 2) set the complexity and size of the model. We initially set the baseline model to use filters value of 128 and f_dim value of 4. The filters value was chosen based on the hardware memory constraints and training time constraints. The f_dim value was chosen to match the number of target classes. We then explored smaller values for filters (32, 64) and larger values for f_dim (8, 16) to investigate their effects on prediction metrics. The results are summarized in Table 3 and Table 4. The system performance trained in RS data arrangement as summarized in Table 3 was relatively similar across the values of filters and f_dim, as can be seen more clearly in Figure 5. Accuracy and average F1 values were greater than 75% for all models indicating that the models were able to learn features of all target classes very well.

The system performance trained in SA data arrangement as summarized in Table 4 was significantly worse than the system performance trained in RS data arrangement.

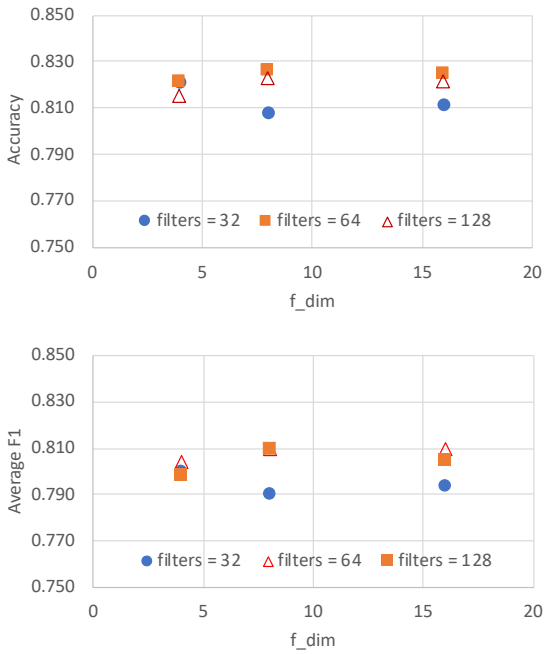

Figure 5. Dependence of accuracy and average F_1 (averaged across target classes) on f_dim and filters for the system trained using RS data arrangement.

Accuracy and average F1 values were less than 60% and 50% for all models as shown in Figure 6. This means that the trained models had poor generalization to predict EEG epochs from previously unseen mice. The system performance was relatively similar across all trained models as seen in RS data arrangement case. The poor generalization was likely due to relatively small number of mice in the dataset. Getting a trained model that generalizes well generally requires a dataset that represents the diversity of the population. In this dataset, the patterns in some excluded mice cannot be learnt from the mice in the training dataset. This can be observed from the system performance across cross-validation folds (not presented for brevity). For example, precision for mTBI wake of the largest model ranged from 0 to 0.906. This wide range of values indicated that the trained model learned mTBI wake features that generalized well to the unseen mTBI mouse for some folds, while trained model for other folds did not generalize well. Similar observations can be made for the other target classes.

C. *Comparison to previous work*

Classification accuracy for our system was compared to a previous work [6] that utilized the same data set as this work (see Table 5). Since the system in previous work performed 2-class classification (sham or mTBI), we post-processed the outputs of our system to combine mTBI classes to mTBI class and Sham classes to Sham class. Performance metrics for 2-class classification were

Table 3. System performance trained using RS data arrangement using 64-s EEG epochs. Values in bold are best values.

filters	**32**	**32**	**32**	**64**	**64**	**64**	**128**	**128**	**128**
f_dim	**4**	**8**	**16**	**4**	**8**	**16**	**4**	**8**	**16**
Accuracy	0.821	0.808	0.811	0.821	**0.826**	0.825	0.815	0.823	0.821
Sham Wake									
Precision	0.881	0.914	0.899	0.919	0.899	0.921	0.911	**0.932**	0.888
Recall	**0.910**	0.869	0.878	0.872	0.886	0.873	0.877	0.854	0.880
F_1	0.895	0.891	0.888	0.895	0.893	**0.896**	0.894	0.892	0.884
Sham Sleep									
Precision	0.742	0.685	0.662	0.725	0.718	0.682	0.670	0.723	**0.765**
Recall	0.758	0.753	0.840	0.787	0.801	**0.902**	0.859	0.811	0.745
F_1	0.750	0.717	0.741	0.755	0.757	**0.777**	0.753	0.764	0.755
mTBI Wake									
Precision	**0.849**	0.819	0.812	0.813	0.838	0.841	0.805	0.830	0.779
Recall	0.701	0.765	0.765	0.756	0.750	0.735	**0.805**	0.779	0.799
F_1	0.768	0.791	0.787	0.783	0.791	0.784	**0.805**	0.804	0.789
mTBI Sleep									
Precision	0.765	0.743	**0.828**	0.756	0.794	0.821	0.827	0.747	0.787
Recall	**0.814**	0.783	0.703	0.811	0.802	0.745	0.675	**0.814**	0.802
F_1	0.789	0.762	0.760	0.783	**0.798**	0.781	0.743	0.779	0.794

Table 4. System performance trained using SA data arrangement using 64-s EEG epochs. Values are average values from cross-validation folds. Values in bold are best values.

filters	**32**	**32**	**32**	**64**	**64**	**64**	**128**	**128**	**128**
f_dim	**4**	**8**	**16**	**4**	**8**	**16**	**4**	**8**	**16**
Accuracy	0.534	0.504	0.525	0.537	0.476	0.542	0.557	0.492	**0.571**
Sham Wake									
Precision	0.712	0.701	**0.717**	0.688	0.622	0.688	0.712	0.502	0.631
Recall	0.765	0.683	0.745	0.722	0.655	**0.802**	0.724	0.631	0.686
F_1	0.697	0.627	0.691	0.700	0.616	**0.726**	0.710	0.555	0.636
Sham Sleep									
Precision	0.386	0.365	0.365	0.302	0.273	**0.403**	0.385	0.354	0.335
Recall	0.551	0.617	**0.631**	0.492	0.412	0.509	0.609	0.543	0.503
F_1	0.419	0.430	0.434	0.324	0.302	0.398	**0.452**	0.380	0.348
mTBI Wake									
Precision	0.265	0.259	0.194	**0.400**	0.302	0.397	0.309	0.284	0.303
Recall	0.165	0.125	0.137	0.247	0.232	0.180	0.234	0.173	**0.259**
F_1	0.186	0.154	0.149	**0.282**	0.208	0.231	0.248	0.214	0.257
mTBI Sleep									
Precision	0.351	0.267	0.319	0.379	**0.406**	0.352	0.326	0.370	0.296
Recall	0.381	0.336	0.339	**0.472**	0.461	0.329	0.357	0.312	0.341
F_1	0.341	0.273	0.305	**0.388**	0.378	0.296	0.291	0.295	0.280

then recalculated. Best accuracies were picked for comparison.

We found that for various scenarios, accuracies for the previous work were comparable to our system for the RS data arrangement and higher than our system for the SA data arrangement. For the SA data arrangement, the nature of neural network and the difference in classification target classes that the networks were trained on could be factors in the difference in accuracy values. We speculate that features extracted by the Feature Extraction layer may be the primary source of better accuracy. Overall, this means that as we increase the number of mTBI mice available to train our network,

Table 5. Comparison of classification accuracy with previous work. For this work, mTBI Wake and mTBI Sleep are labeled as mTBI, and similarly for Sham.

Source	**Reference [6]**			**This work**					
Arrangement	SA	SA	SA	SA	SA	SA	RS	RS	RS
Total mice	9	9	9	10	10	10	11	11	11
Train epochs	Sleep	Wake	All	All	All	All	All	All	All
Test epochs	Sleep	Wake	All	Sleep	Wake	All	Sleep	Wake	All
Network	CNN			CNN					
First layer	Feature Extraction (extracting sub-band average power)			Conv1D					
Accuracy (Sham/mTBI)	0.780	0.854	0.920	0.568	0.684	0.634	0.830	0.902	0.869

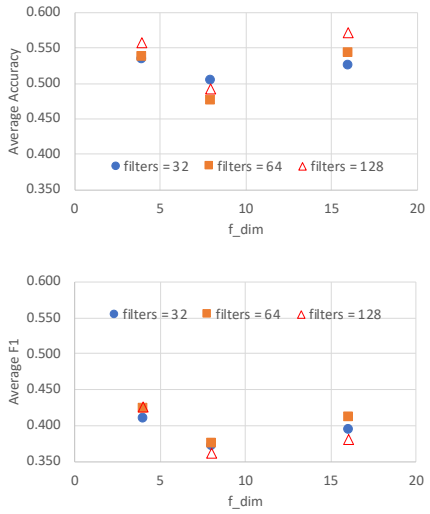

Figure 6. Dependence of accuracy and average F_1 (averaged across target classes) on f_dim and filters for the system trained using SA data arrangement.

we can expect the performance to match the network in the previous work.

D. Deployment of model on Raspberry Pi

The models used in sections 4 (A) and 4 (B) were deployed on Raspberry Pi 4 to retrieve metrics. We tested the smallest (filters = 32, f_dim = 4) and the largest model (filters = 128, f_dim = 16) for both RS and SA. We found that the results on Raspberry Pi were identical to those achieved with a general computer (HPC in this case). Actual results from Raspberry Pi are shown in Table 6, side by side with the results from a general computer. In terms of processing time, we found that the processing time increases with increase in the number of epochs processed in a batch (Figure 7), which is expected. We determined that the processing time is significantly smaller than the time required to collect the EEG epochs.

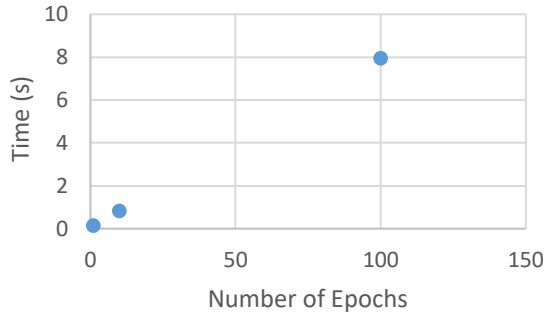

Figure 7. Variation of processing time with epoch batch size

Table 6. System performance comparison of Raspberry Pi (RPi) with a general computer (HPC) using 4 classes and 64 s epochs for each case

Type	**RS**				**SA**			
Device	**RPi**	**HPC**	**RPi**	**HPC**	**RPi**	**HPC**	**RPi**	**HPC**
filters	**32**	**32**	**128**	**128**	**32**	**32**	**128**	**128**
f_dim	**4**	**4**	**16**	**16**	**4**	**4**	**16**	**16**
Accuracy	0.820	0.821	0.821	0.821	0.533	0.534	0.571	0.571
Sham Wake								
Precision	0.880	0.881	0.887	0.888	0.711	0.712	0.631	0.631
Recall	0.910	0.910	0.879	0.880	0.764	0.765	0.686	0.686
F_1	0.895	0.895	0.883	0.884	0.696	0.697	0.635	0.636
Sham Sleep								
Precision	0.742	0.742	0.765	0.765	0.385	0.386	0.334	0.335
Recall	0.757	0.758	0.744	0.745	0.550	0.551	0.502	0.503
F_1	0.750	0.750	0.754	0.755	0.419	0.419	0.347	0.348
mTBI Wake								
Precision	0.848	0.849	0.779	0.779	0.264	0.265	0.302	0.303
Recall	0.700	0.701	0.799	0.799	0.164	0.165	0.259	0.259
F_1	0.767	0.768	0.789	0.789	0.186	0.186	0.257	0.257
mTBI Sleep								
Precision	0.764	0.765	0.787	0.787	0.351	0.351	0.295	0.296
Recall	0.813	0.814	0.801	0.802	0.380	0.381	0.341	0.341
F_1	0.788	0.789	0.794	0.794	0.341	0.341	0.280	0.280

V. Conclusions

The proposed system to classify sleep/wake stages and detect mTBI in mice from single-channel EEG epochs was successfully demonstrated to distinguish the four target classes (sham wake, sham sleep, mTBI wake, mTBI sleep). Using supervised learning methodology, we trained and evaluated several Deep CNN configurations using random-sampling (RS) data arrangement to verify their capabilities to learn features for classification and species-aware (SA) data arrangement to verify the generality of the trained model. From comparison with previous work [6] using same source data, we found that the proposed system achieved similar performance in RS data arrangement with previous work that used handcrafted features. However, generality of the learnt features was not as good, possibly due to relatively low number of mice in the dataset. The results obtained in the current work remain preliminary and should be validated with a separate, larger dataset. With a larger sample size and additional validation cohort, the system should be able to reach performance metrics achieved in the RS data arrangement for the case of SA data arrangement.

Further, we demonstrated that the proposed classification system can be deployed on a portable device (Raspberry Pi) to perform real-time mTBI stage classification and verified that the results on the deployed system were identical to those obtained on a HPC. We also verified that the processing time required by the deployed, buffered, queue-based system to classify EEG epochs in a live configuration (simulated by generating epochs

from EEG recording files) was significantly smaller than the time required to collect EEG epochs from a live signal, making the system feasible for practical EEG classification applications. To our knowledge, this is the first system capable of performing sleep stage classification of mTBI related EEG signals deployed on a portable, low cost system like Raspberry Pi. The techniques developed in this work are general and can be extended to data from individuals. Thus, the proposed system has potential to be used medically to provide a live, low-cost and fast method for detection of mTBI in individuals in the future.

Advantages of the working classification system developed in the current work include replacement of the labor-intensive manual sleep-stage scoring of EEG signals by human experts with an online and automated system with the capability to perform fast sleep staging and mTBI detection. This fast feedback without the need to collect EEG data for a long duration (24 hours for the data set used in our work) may open a way to perform different types of experiments that are not possible with the manual scoring by human experts, which is typically done after long EEG recordings are available.

ACKNOWLEDGMENTS

This research was funded by the Graduate Research Award of Computing and Software Systems division and the startup fund 74-0525 of the University of Washington Bothell. We gratefully acknowledge the support of NVIDIA Corporation (Santa Clara, CA, USA) with the donation of the GPU used for this research.

The data in this work was supported with resources and the use of facilities at the VA Portland Health Care System, VA Career Development Award #1K2 BX002712 to M.M.L. Interpretations and conclusions are those of the authors and do not represent the views of the U.S. Department of Veterans Affairs or the United States Government.

REFERENCES

[1] "TBI: Get the Facts | Concussion | Traumatic Brain Injury | CDC Injury Center," Mar. 11, 2019. (available at: *https://www.cdc.gov/traumaticbraininjury/get_the_facts.html*, accessed Sep. 27, 2020).

[2] D. K. Sandsmark, J. E. Elliott, and M. M. Lim, "Sleep-Wake Disturbances After Traumatic Brain Injury: Synthesis of Human and Animal Studies," *Sleep*, Mar. 2017, doi: 10.1093/sleep/zsx044.

[3] M. M. Lim *et al.*, "Dietary Therapy Mitigates Persistent Wake Deficits Caused by Mild Traumatic Brain Injury," *Science Translational Medicine,* vol. 5, no. 215, pp. 215ra173-215ra173, Dec. 2013, doi: 10.1126/scitranslmed.3007092.

[4] S. V. Kabadi, G. D. Hilton, B. A. Stoica, D. N. Zapple, and A. I. Faden, "Fluid-percussion–induced traumatic brain injury model in rats," *Nat Protoc*, vol. 5, no. 9, pp. 1552–1563, Sep. 2010, doi: 10.1038/nprot.2010.112.

[5] M. Vishwanath *et al.*, "Investigation of Machine Learning Approaches for Traumatic Brain Injury Classification via EEG Assessment in Mice," *Sensors*, vol. 20, no. 7, p. 2027, Apr. 2020, doi: 10.3390/s20072027.

[6] M. Vishwanath *et al.*, "Classification of Electroencephalogram in a Mouse Model of Traumatic Brain Injury Using Machine Learning Approaches" in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, Montreal, QC, Canada, Jul. 2020, pp. 3335-3338, doi: 10.1109/EMBC44109.2020.9175915.

[7] M. H. Modarres, N. N. Kuzma, T. Kretzmer, A. I. Pack, and M. M. Lim, "EEG slow waves in traumatic brain injury: Convergent findings in mouse and man," *Neurobiology of Sleep and Circadian Rhythms*, vol. 2, pp. 59–70, Jan. 2017, doi: 10.1016/j.nbscr.2016.06.001.

[8] P. E. Rapp *et al.*, "Traumatic Brain Injury Detection Using Electrophysiological Methods," *Front Hum Neurosci*, vol. 9, Feb. 2015, doi: 10.3389/fnhum.2015.00011.

[9] M. H. Modarres, R. A. Opel, K. B. Weymann, and M. M. Lim, "Strong correlation of novel sleep electroencephalography coherence markers with diagnosis and severity of posttraumatic stress disorder," *Sci Rep*, vol. 9, no. 1, p. 4247, Dec. 2019, doi: 10.1038/s41598-018-38102-4.

[10] N. S. E. M. Noor and H. Ibrahim, "Machine Learning Algorithms and Quantitative Electroencephalography Predictors for Outcome Prediction in Traumatic Brain Injury: A Systematic Review," *IEEE Access*, vol. 8, pp. 102075–102092, 2020, doi: 10.1109/ACCESS.2020.2998934.

[11] Z.-E. Yu, C.-C. Kuo, C.-H. Chou, C.-T. Yen, and F. Chang, "A machine learning approach to classify vigilance states in rats," *Expert Systems with Applications*, vol. 38, no. 8, pp. 10153–10160, Aug. 2011, doi: 10.1016/j.eswa.2011.02.076.

[12] K.-M. Rytkönen, J. Zitting, and T. Porkka-Heiskanen, "Automated sleep scoring in rats and mice using the naive Bayes classifier," *Journal of Neuroscience Methods*, vol. 202, no. 1, pp. 60–64, Oct. 2011, doi: 10.1016/j.jneumeth.2011.08.023.

[13] A. Sors, S. Bonnet, S. Mirek, L. Vercueil, and J.-F. Payen, "A convolutional neural network for sleep stage scoring from raw single-channel EEG," *Biomedical Signal Processing and Control*, vol. 42, pp. 107–114, Apr. 2018, doi: 10.1016/j.bspc.2017.12.001.

[14] E. Konstantinidis, N. Conci, G. Bamparopoulos, E. Sidiropoulos, F. De Natale, and P. Bamidis, "Introducing Neuroberry, a platform for pervasive EEG signaling in the IoT domain," presented at the 5th EAI International Conference on Wireless Mobile Communication and Healthcare - "Transforming healthcare through innovations in mobile and wireless technologies," London, Great Britain, 2015, doi: 10.4108/eai.14-10-2015.2261698.

[15] R. Arif, S. K. Wijaya, Prawito, and H. S. Gani, "Design of EEG data acquisition system based on Raspberry Pi 3 for acute ischemic stroke identification," in *2018 International Conference on Signals and Systems (ICSigSys)*, Bali, May 2018, pp. 271–275, doi: 10.1109/ICSIGSYS.2018.8372771.

[16] W. Zgallai *et al.*, "Deep Learning AI Application to an EEG driven BCI Smart Wheelchair," in *2019 Advances in Science and Engineering Technology Inter-national Conferences (ASET)*, Dubai, United Arab Emirates, Mar. 2019, pp. 1–5, doi: 10.1109/ICASET.2019.8714373.