Sleep Stage Classification using NeuCube on SpiNNaker: a Preliminary Study

Sugam Budhraja
Computer Science & Information Systems
BITS Pilani, Goa Campus
Goa, India
f20160064@goa.bits-pilani.ac.in

Basabdatta Sen Bhattacharya
Computer Science & Information Systems
BITS Pilani, Goa Campus
Goa, India
basabdattab@goa.bits-pilani.ac.in

Simon Durrant
School of Psychology
University of Lincoln
Lincoln, United Kingdom
sidurrant@lincoln.ac.uk

Zohreh Doborjeh
Centre for Brain Research
The University of Auckland
Auckland, New Zealand
zohreh.doborjeh@auckland.ac.nz

Maryam Doborjeh
Information Technology & Software Engineering
Auckland University of Technology
Auckland, New Zealand
maryam.gholami.doborjeh@aut.ac.nz

Nikola Kasabov
KEDRI
Auckland University of Technology
Auckland, New Zealand
and Ulster University
nkasabov@aut.ac.nz

***Abstract*—This paper studies sleep stage classification using NeuCube, a Spiking Neural Network (SNN) architecture, simulated on SpiNNaker, a neuromorphic computer. The sleep electroencephalogram (EEG) time series is converted to spikes and provided as an input to NeuCube. Relevant feature vectors are extracted at different stages of training. We used six standard machine learning classifiers on different combinations of these feature vectors and calculated 5-fold cross-validation accuracy. We observed that the gradient boosted decision trees classifier performed the best by achieving 81.25% accuracy on a combination of two feature vectors. An evaluation of the results using confusion matrices and classification reports showed that the Awake, N2, SWS and REM sleep stages can be classified with $\geq$ 80% F1-score using the gradient boosted decision trees algorithm. Overall, our proof-of-concept work towards autonomous sleep-stage classification using NeuCube shows promise and will form the base for continued research in this direction.**

***Index Terms*—Sleep stage classification, EEG, Spiking Neural Networks, NeuCube, SpiNNaker**

I. Introduction

The traditional method for sleep stage classification is through visual inspection of Electroencephalogram (EEG). A sleep expert looks at 30 second segments of EEG data (called epochs) to identify the sleep stage in accordance with the specifications in the American Academy of Sleep Medicine (AASM) manual [1]. However, this method of scoring is highly time-consuming and dependent on the expert's experience; an automated classification process would be highly desirable for a better, faster and efficient classification system. A recent review [2] looks into several automatic schemes for sleep stage classification based on EEG analysis. A few different approaches that have been taken for sleep stage classification include Maximum Overlap Discrete Wavelet Transform [3], Higher Order Spectra [4], Convolutional Neural Networks [5] and Deep Neural Networks [6]. A relatively recent work provides a proof-of-concept for decoding and classifying time-series signals including EEG using NeuCube, a spiking neural architecture, running on the SpiNNaker machine, a low-power neuromorphic hardware that aims to run in real time [7]. Inspired by this work, we have looked into classifying sleep stages from raw EEG data using NeuCube simulated on SpiNNaker.

Arousal can be divided into three stages namely awake (W), non-rapid eye movement (NREM) and rapid eye movement (REM). NREM can be further divided into light transitional sleep (N1), proper light sleep (N2) and slow wave sleep (SWS). A normal night-time sleep cycle usually follows the sequence: N1, N2, SWS, N2, REM; typically, there are 4 to 6 cycles during a full night's sleep [3]. The gold standard in method of sleep monitoring is polysomnography (PSG). PSG contains multiple measures that are recorded over a person's full night sleep, such as EEG, electromyography (EMG), electrooculogram (EOG), electrocardiography (ECG), blood pressure, heart rate, oxygen saturation and respiration [8]. EEG measures the electrical activity of the brain with excellent temporal resolution across the scalp and is relatively inexpensive. We have used EEG signals of a full night sleep from one person recorded at our laboratory; this is detailed in Sec. II-A.

NeuCube is a spiking neural network (SNN) architecture designed with the motivation of mapping, learning and understanding spatio-temporal brain data (STBD) [9]. Spiking neural networks can incrementally learn from brain dynamics gathered over time in a 3D space and capture meaningful patterns from brain data. The NeuCube architecture consists of three modules viz. Signal to Spike Encoder, SNN Reservoir/Cube and Output Classifier; each of these are described in brief in Sec. II-B.

The SpiNNaker machine is a million-core neuromorphic platform, designed for low power consumption while simulating a massive number of spiking neurons in parallel [10]. It resides at the University of Manchester. In a recent collaboration between the SpiNNaker and the NeuCube groups [7], Behrenbeck et al looked into simulating the NeuCube SNN

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

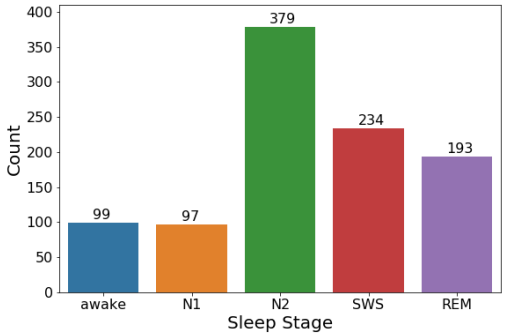

Fig. 1. Number of 30s epochs that belong to each class.

on SpiNNaker. Furthermore, they demonstrated that simulating NeuCube on SpiNNaker rather than Matlab is more efficient when using a larger number of data samples.

The code used in this work is written in Python 3 using the PyNN library [11]. The sPyNNaker package [12] allows code written using PyNN to be simulated on SpiNNaker. Our implementation on SpiNNaker allowed us to use up to 800 samples for training and testing our model. We have used the Leaky-Integrate-Fire (LIF) neuron as our spiking neuron model in the network, same as [7]. Further details on the implementation are specified in Sec. II.

Section III presents our findings from this study. We observe that the best classification accuracy is achieved using the Gradient Boost Decision Tree (GBDT) algorithm, using two feature vectors extracted from different modules of NeuCube, during different stages of training. A combination of confusion matrices and classification reports indicate $\geq$ 84% F1 score for Awake, REM and SWS stages; 81% for N2 stage; 44% for N1 sleep stage when using 800 samples. Thus, the N1 sleep stage could not be classified to a desirable level with our training algorithm. Section IV presents the conclusion from this study.

II. Methodology

In Sec. II-A we describe the EEG dataset which we use for training and testing simulation. In Sec. II-B we make a brief description of the data processing techniques used in NeuCube and the structure of the spiking neural network. In Sec. II-C we discuss the learning rules implemented to train the spiking neural network. In Sec II-D we describe the final module of NeuCube called the Output classifier and the feature vectors extracted from NeuCube. In Sec. II-E we describe the classification methods used after extracting features from NeuCube.

A. EEG data of sleep

The EEG dataset used here has been recorded at the Sleep and Cognition Laboratory at the University of Lincoln and contains data of a healthy adult person's full night sleep [13]. Five EEG scalp electrodes, that have been recommended in the AASM manual, are used in this work viz. F3-M2, F4-M1, C3-M2, C4-M1 and O1-M2. The continuous EEG signal

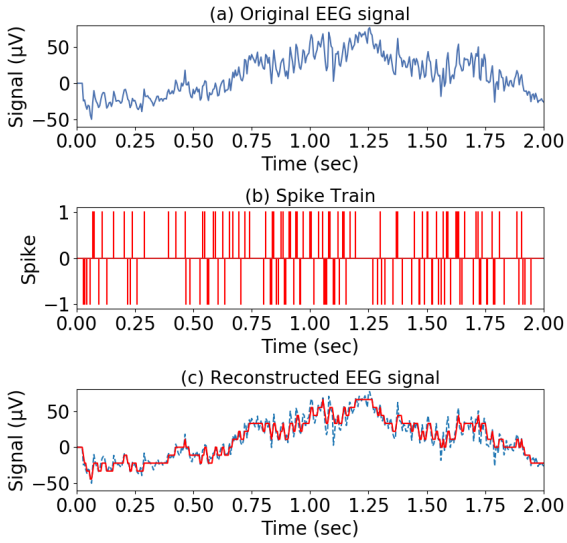

Fig. 2. (a) A 2s EEG signal. (b) Encoded spike train produced using the Step Forward (SF) algorithm. (c) The reconstructed EEG signal superposed with the original EEG signal.

is split into 30s epochs and each epoch is assigned a sleep stage by a sleep expert. As mentioned before, the five sleep stages considered in this work are Awake, N1, N2, SWS and REM. Since the EEG data was recorded for 8 hours 21 minutes, after splitting and labelling we ended up with 1002 30s epochs. Fig. 1 shows the number of epochs in each sleep stage i.e. class. Furthermore, the EEG signals were recorded at a frequency of 200 Hz. Since each 30s epoch has data from 5 channels, it will contain 5$\times$30$\times$200 i.e. 30000 data points.

B. NeuCube Spiking Neural Architecture

NeuCube consist primarily of three modules; here we discuss about the first two.

1) Signal to Spike Encoder: A typical spiking neural network takes spike trains (discrete binary signals that record the times at which a neuron fires) as inputs. Thus, the sleep EEG signals in Sec. II-A, which are to be provided as inputs to the spiking neural network in NeuCube, need to be first converted to spike trains. This is achieved using analog-to-spike encoding algorithms like Threshold Based Representation (TBR), Ben's Spiking Algorithm (BSA), Moving Window (MW) and Step Forward (SF) [14]. We have tested all four aforementioned algorithms on our data, and found that the TBR and MW algorithms generate poor reconstructed signals (not shown here). Thus we focus on the BSA and SF algorithms to find the better of the two in context of our dataset. For ease of reading, we provide a brief overview of the SF and BSA algorithms.

- *Step Forward (SF):* SF is a simple threshold based algorithm. It has two main parameters viz. $base$ and $threshold$. Initial value of $base$ is the value of signal at time $t = 0$, while the $threshold$ parameter is user defined. If the signal value at t is greater than

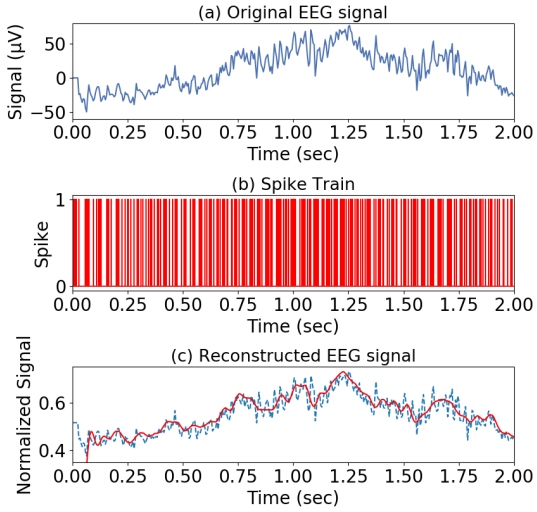

Fig. 3. (a) A 2s EEG signal. (b) Encoded spike train produced using the Ben's Spiker Algorithm (BSA). (c) The reconstructed EEG signal superposed with the original EEG signal.

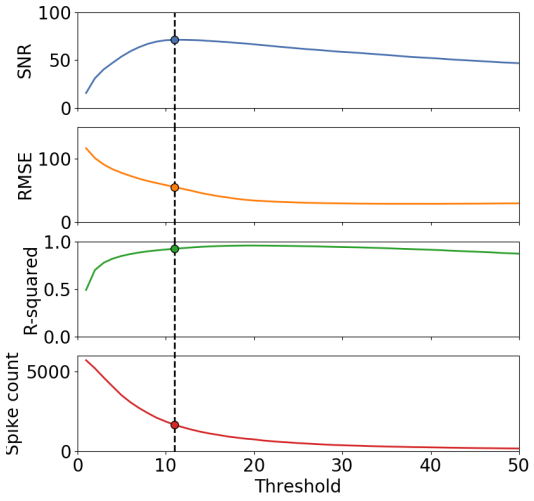

Fig. 4. Threshold optimization in the SF Algorithm. At $threshold$ value 11 the SNR maximizes, RMSE decreases, R-squared score increases to nearly 1 and spike count also decreases significantly.

$base + threshold$ then an excitatory spike is encoded and $base$ is updated to value $base + threshold$. If the signal value at t is less than $base - threshold$ then an inhibitory spike is encoded and $base$ is updated to value $base - threshold$. In other situations, no spike is encoded and value of base remains same [15]. A 2s EEG signal and its spike train, encoded using SF, is shown in Fig. 2.

- *Ben's Spiker Algorithm (BSA):* This algorithm is based on use of FIR filters to encode the signal. At each time instant τ we calculate two errors $\sum_{k=0}^{M} abs(s(k+\tau) - h(k))$ and $\sum_{k=0}^{M} abs(s(k+\tau))$ where s is the original signal and h is the FIR filter of length M. If the first error is smaller than the second minus a threshold, then we encode a spike and subtract the filter from the input. A simple convolution between the spike train and FIR filter can be performed to recover the signal from the spike train. BSA algorithm produces only positive spikes [16]. A 2s EEG signal and its spike train, encoded using BSA, is shown in Fig. 3.

To understand the efficacy of the encoding methods, the spike trains were converted back to the analog signals — the result is referred to as the 'reconstruction' of the original signal. The reconstructed signals are shown in Fig.s 2(c) and 3(c).

2) Optimal Parameters for our simulation: To obtain an objective estimate of how well the signal were reconstructed, we used four metrics as suggested by Petro et al [14]:

- *Signal-to-noise ratio (SNR):* This indicates how much noise is introduced into the signal by the encoding algorithm and is defined in Eqn. (1).

$$\text{SNR} = 20 \cdot \log \frac{\text{Power}(s)}{\text{Power}(\text{s} - \text{r})} [\text{dB}] \tag{1}$$

where s is the original signal and r is the reconstructed signal. Greater the difference between s and r, the lower the SNR will be.

- *Root mean squared error (RMSE):* RMSE is a well known metric used to evaluate the error between two signals. It is calculated as root of the sum of squared differences between the two signals.
- *R-squared score:* R-squared score is a metric typically used to evaluate regression models and is defined in Eqn. 2. Here the original signal can be considered as the required output of a regression problem and the reconstructed signal is our model's prediction.

$$R^2 = 1 - \frac{\sum_{t=1}^{T} (s_t - r_t)^2}{\sum_{t=1}^{T} (s_t - \bar{s})^2} \tag{2}$$

where s is the original signal, r is the reconstructed signal and $\bar{s}$ is the mean of the original signal.

- *Spike Count:* A lesser spike count means we are able to represent the EEG signal with lesser number of spikes.

Optimal parameters for an encoding algorithm would produce a reconstructed signal that maximizes SNR, minimizes RMSE, maximizes R-squared score and has a relatively lower spike count. As mentioned before, we observed that with optimal parameters, only BSA and SF algorithms gave good reconstructed signals. A filter of size 30 and cutoff frequency 0.1 is used for BSA. We found the optimal $threshold$ value for BSA to be 0.9 and for SF to be 11. Out of the two, the SF algorithm gave better reconstructed signals with higher SNR. Hence, we proceeded with using the SF algorithm in the encoder module. Fig. 4 shows the mean values of the four different metrics over the five EEG signals of the first 30s epoch. The number of spikes in these spike trains forms the first of three feature vectors used for the final classification.

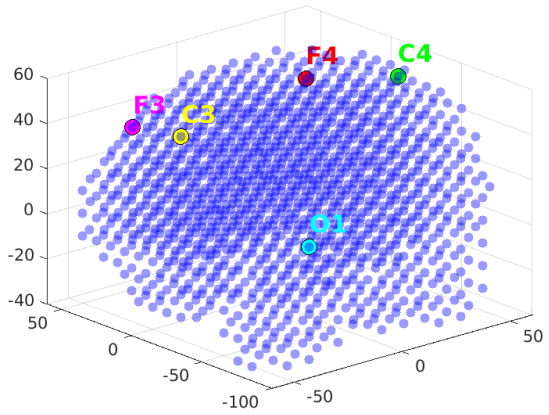

Fig. 5. A pictorial depiction of the SNNr created using Talairach brain atlas containing 1471 neurons with mapped electrode positions. This structure is used for extracting the EEG channel interaction plots.

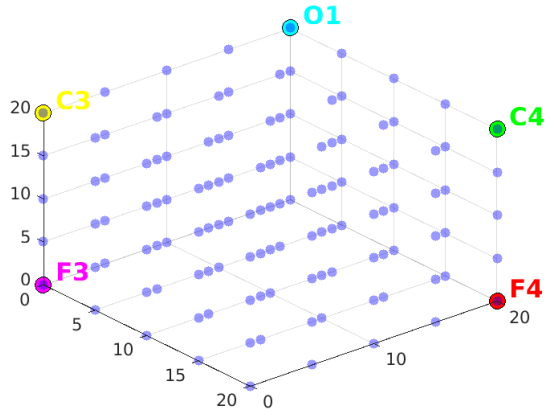

Fig. 6. A pictorial depiction of the 3-D grid consisting of 125 neurons used for the classification task in this work.

3) NeuCube Reservoir: The SNN reservoir (SNNr) consists of spiking neurons whose locations and connections are modelled after the 3D-geometry of the human brain. The Talairach brain atlas is used to place the neurons in a spatially meaningful manner. The 10/20 EEG scalp electrode positions are converted to Talairach coordinates using the mapping given by Koessler et al [17]. The input spike trains are introduced into the cube at these mapped coordinates simultaneously. The distance between two consecutive neurons in one dimension is chosen to be 10mm so that it is easier to map the 10/20 electrode positions. The resulting reservoir contains 1471 LIF neurons where each neuron represents 1 cm^3 of the brain [18]. The SNNr is scalable in size and can support different types of mappings like brain atlas mapping, spatial mapping based on spike correlation and personalized mapping which were explored by Tu et al [19]. We have used the original brain shaped spatial structure for creating the channel interaction plots in Sec. III-A. However, for the classification task, we have adopted a simpler structure containing 125 spiking neurons that are arranged in a 3-D grid ($5 \times 5 \times 5$) with a resolution of 5mm as shown in Fig. 6. This helped in avoiding high dimensionality of feature vectors and also greatly reduced the training time.

The connections between the neurons in the reservoir are initialized using small world connectivity approach. This means that neurons that are closer to each other have a higher chance of being connected. The excitatory and inhibitory synapses within the reservoir are probabilistically determined based on the calculated connection probability $P_{i,j}$ between any two neurons N_i and N_j. :

$$P_{i,j} = \begin{cases} \mathrm{C} * \mathrm{e}^{-\left(d_{i,j}^{\mathrm{norm}}/\lambda\right)^2} & \text{if } d_{i,j}^{\mathrm{norm}} \leq \mathrm{d_{thresh}} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where C is the maximum connection probability which was set to 0.25, λ represents the small world connection radius which was set to 2.5, $d_{i,j}^{norm}$ is the normalized distance between neurons N_i and N_j, d_{thresh} is the maximum connection distance which was set to half of maximum distance between any two reservoir neurons. The values of these parameters are the same as used by Behrenbeck et al [7].

Each connection also has a weight and a delay associated with it. The synaptic weight represents the contribution of a pre-synaptic neuron N_i in the firing of the post-synaptic neuron N_j. Whenever N_i fires, the membrane potential of N_j increases or decreases by the weight according to Spike Timing Dependent Plasticity (STDP) rule (see Sec. II-C). The synaptic delay represents the time taken by a spike generated by N_i to reach N_j. It is proportional to the euclidean distance between the neurons.

C. Learning in NeuCube

During the training phase, the spike trains of the training samples are introduced at the mapped locations of the SNN reservoir and the connection weights are updated using an unsupervised learning rule called Spike-Time Dependent Plasticity (STDP) — where the change in value of a synaptic connection's weight depends on the time of spiking in the pre-synaptic and post-synaptic neurons. If a pre-synaptic neuron spikes just before the post-synaptic neuron, then the synaptic weight of that connection is increased (made stronger) under the assumption of causality between the two neurons. Conversely, if a post synaptic neuron spikes before a pre-synaptic neuron, the assumption is of a lack of causality, and the synaptic weight is decreased (made weaker). Thus, connections in which the pre-synaptic spike causes the post-synaptic spike contribute more in the future (Long Term Potentiation), whereas connections in which the pre-synaptic neurons don't cause the post-synaptic spike contribute less in the future (Long Term Depression). An output spike is only produced when many input spikes occur together in a short period of time. Hence, pair of neurons with high synaptic weights are correlated in time.

The learning weight is defined according to the quintessential STDP learning weight look up table first defined by Bi and Poo [20] from physiological studies .

$$W(s) = \begin{cases} A_+ \exp\left[s/\tau_+\right] & \text{for } s < 0 \\ A_- \exp\left[-s/\tau_-\right] & \text{for } s > 0 \end{cases} \quad (4)$$

s represents the difference between arrival time of the pre-synaptic spike and the firing time of post-synaptic spike. τ_+ represents the pre-synaptic time interval and it was set to 10. τ_- represents the post-synaptic time interval and it was set to 1. A_+ represents amplitude of weight change when pre-synaptic spike arrives before post-synaptic spike. A_- represents amplitude of weight change when post-synaptic spike arrives before pre-synaptic spike. Both A_+ and A_- were set to 0.01. The parameters associated with STDP were optimized for this dataset.

All training samples are passed through the SNNr and the weights of connections between reservoir neurons are updated using STDP unsupervised learning rule before moving on to the final module in NeuCube.

D. Classification Task in NeuCube

The last module in NeuCube is called the Output Classifier. It uses dynamic evolving SNN (deSNN) algorithm for supervised learning, where the association between the class labels of the training samples is learned. In deSNN, a new output neuron (O) is created for each training sample. This output neuron is connected to every reservoir neuron (N). The initial weights for these connections are computed using the Rank-Order learning rule (RO) as described in Sec. II-D1. Then the spike trains of the sample are introduced in the SNNr and the weights of synaptic connection between the reservoir and output neurons are updated using the Spike Driven Synaptic Plasticity (SDSP) algorithm described in Sec. II-D2. The test samples undergo the same processes as the training samples except the STDP weight update.

Three different feature vectors are extracted from NeuCube for every sample. The first feature vector is the number of spikes in the input spike trains and id referred to as Spike Count (SC). The second feature vector is the number of spikes that occur at each reservoir neuron in the SNNr during deSNN and is referred to as Spikes Per Neuron (SPN). The third feature vector is the final weights of connections formed during deSNN and is referred to as Final Weights (FW). Any standard classifier can be used on these feature vectors to predict label for the test samples. Behrenbeck et al [7] used K-Nearest Neighbours (KNN) algorithm but we have also tried other supervised learning algorithms to find the most suitable one for sleep stage classification. These are described briefly in Sec. II-E.

1) Rank Order Learning Rule (RO): Rank-order codes are inspired by fast information processing by sensory neurons. The hypothesis is that the spike arriving first from a reference point in time encode the most salient information about the surrounding environment. Thus, higher synaptic weights are assigned to neurons that fire first, and connection weights are updated based on the order of arrival of spikes.

$$w_{\text{init}}\left(\mathrm{N}_n, \mathrm{O}_m\right) = \alpha \times \mathrm{mod}^{order(\mathrm{N}_n, \mathrm{O}_m)} \tag{5}$$

In our work, the value of α is set to 1 and value of mod is set to 0.9. These values were optimized for our dataset.

2) Spike Driven Synaptic Plasticity (SDSP): SDSP a modified version of STDP, where the pre-synaptic spikes are compared with post-synaptic membrane potential. If at the time of arrival of a pre-synaptic spike, the post-synaptic membrane potential is above some threshold then the synaptic weight is increased (strengthened). Otherwise, if at the time of arrival of a pre-synaptic spike, the post-synaptic membrane potential is below the threshold (usually if it has fired recently) then the synaptic weight is decreased (weakened). In NeuCube, it is assumed that if a pre-synaptic neuron fires, the post-synaptic membrane potential will always be above the threshold. Hence, if there is a pre-synaptic spike, the connection weight is increased, otherwise it is decreased.

$$\begin{aligned} w_{\text{final}}\left(\mathrm{N}_n, \mathrm{O}_m\right) = w_{\text{init}}\left(\mathrm{N}_n, \mathrm{O}_m\right) + \text{drift}_{\text{up}} \times n_{\text{spikes}} \\ - \text{drift}_{\text{down}} \times n_{\text{nospikes}} \end{aligned} \tag{6}$$

where $drift_{up}$ represents the increase in the synaptic weight when a spike is observed at the pre-synaptic neuron and $drift_{down}$ represents the decrease in synaptic weight when no pre-synaptic spike is observed. $drift_{up}$ is set to value 0.08 and $drift_{down}$ is set to value 0.01.

The neurons which fire more will have higher final weights in their connections to the output neuron. The value of $drift_{up}$ is kept more than $drift_{down}$ so that the difference between final weights of neurons that fire more and neurons that fire less is significant.

E. Classifiers used in this work

- *K-Nearest Neighbours (KNN)*: KNN is a simple distance-based algorithm. To classify a new sample, we find the distance between it and every other existing training sample. We find the k closest samples and classify the new sample as majority label from the k selected samples.
- *Logistic Regression (LR)*: LR is in which we try to learn a linear decision boundary that separates the data into different classes. It uses a softmax function on top of the linear regression algorithm that calculates probabilities of a sample belonging to each class. Based on these probabilities we can choose the suitable class for the sample.
- *Support Vector Machines (SVM)*: SVM finds a separating decision boundary that maximizes the margins (gap between the data and the boundary). SVMs also use a kernel function to map the data to a higher dimensional space with the aim of making the data linearly separable.
- *Multilayer Perceptron (MLP)*: MLP is a simple feed forward neural network. It consists of an input layer, few hidden layers and an output layer. Each layer contains a decided number of perceptrons. Each layer is fully connected with the next and the connection weights are updated using back propagation.
- *Random Forest (RF)*: Random Forest is an ensemble learning method that uses multiple decision trees for classification. Each decision tree is given a random subset of training samples and the results of all the trees are averaged to produce the final result. The decision trees

TABLE I
SUMMARY OF 5-FOLD CROSS-VALIDATION ACCURACY OF SLEEP STAGE CLASSIFICATION USING NEUCUBE RUNNING ON SPINNAKER.

Number of epochs	Features	Classifiers					
		KNN	***LR***	***SVM***	***MLP***	***GBDT***	***RF***
200	Spike count (SC)	62 ± 2.45	64 ± 3.39	64 ± 4.36	67.5 ± 8.14	65 ± 5.7	62 ± 4.85
	Spikes per neuron (SPN)	62.5 ± 6.12	65.5 ± 5.79	67.5 ± 4.74	67.5 ± 7.5	71 ± 6.04	69.5 ± 8.86
	Final Weights (FW)	62.5 ± 8.06	66 ± 5.15	66.5 ± 8.46	70.5 ± 7.89	73.5 ± 9.70	73.5 ± 9.82
	SC + FW	68 ± 7.34	76 ± 3.38	73 ± 4.85	78 ± 9.54	77.5 ± 6.12	75.5 ± 11.77
	SPN + FW	62.5 ± 8.06	66.5 ± 5.61	68.5 ± 5.61	68.5 ± 5.39	71.5 ± 11.24	72 ± 9.92
	SC + SPN	68.5 ± 4.64	77.5 ± 5.7	74.5 ± 4	60.5 ± 11.66	**79 ± 5.64**	75 ± 12.14
	SC + SPN + FW	68 ± 6.78	76.5 ± 6.44	74 ± 4.64	58 ± 6.2	76 ± 7.52	75.5 ± 10.3
500	Spike count (SC)	68 ± 2.45	68.4 ± 1.96	70.6 ± 1.02	71.2 ± 3.87	69.2 ± 3.31	69.2 ± 2.93
	Spikes per neuron (SPN)	70.8 ± 4.53	72.6 ± 3.83	75.2 ± 1.17	71.2 ± 6.34	75.4 ± 1.62	74.4 ± 2.58
	Final Weights (FW)	71.6 ± 2.42	71.4 ± 2.94	74.2 ± 0.75	74 ± 2.1	73.8 ± 3.54	74.2 ± 1.72
	SC + FW	72.2 ± 3.54	72.6 ± 3.14	75 ± 1.79	74.6 ± 1.2	74 ± 3.69	75.2 ± 1.6
	SPN + FW	71.6 ± 2.87	71.8 ± 3.92	73.8 ± 1.47	73.4 ± 1.02	74 ± 4.94	74.4 ± 2.33
	SC + SPN	73.8 ± 3.19	73.8 ± 2.93	75.4 ± 2.33	74 ± 1.79	**75.4 ± 2.87**	75.4 ± 2.33
	SC + SPN + FW	72.2 ± 4.92	73 ± 2.68	75 ± 0.89	73.6 ± 1.2	74.6 ± 3.5	74.8 ± 2.48
800	Spike count (SC)	73.5 ± 3.03	71.75 ± 1.27	74.875 ± 2.28	73.5 ± 1.84	73.5 ± 1.46	73.875 ± 2.32
	Spikes per neuron (SPN)	74.625 ± 2.73	79.25 ± 3.22	79.375 ± 2.88	78.125	80.25 ± 3.48	78.875 ± 4.04
	Final Weights (FW)	74.375 ± 3.04	78.25 ± 3.57	78.75 ± 3.11	78.875 ± 2.48	80.375 ± 4.41	78.625 ± 3.22
	SC + FW	74.5 ± 2.51	78.25 ± 4.08	79 ± 3.2	80.375 ± 3.53	81 ± 5.1	78.75 ± 3.24
	SPN + FW	74.5 ± 2.54	78.75 ± 3.77	79.125 ± 2.92	78.625 ± 3.59	**81.25 ± 4.24**	78.75 ± 4.05
	SC + SPN	74.25 ± 2.93	78.75 ± 3.97	79.25 ± 3.27	78.875 ± 2.72	80.875 ± 3.27	78.875 ± 4.58
	SC + SPN + FW	74.875 ± 2.42	78.5 ± 3.37	79.125 ± 3.37	78.875 ± 4.32	81 ± 3.39	78.875 ± 3.76

Results are presented as Mean ± Standard Deviation of the 5 accuracies. Highlighted results were used for evaluation of classification errors.

only use a random subset of the features while splitting nodes.

- *Gradient boosted decision trees (GBDT)*: GBDT is a residual-based ensemble learning method, based on the concept of boosting. In boosting, initially there is a weak learner (just slightly better than random chance). Models are added repetitively that try to learn the errors of the previous model. GBDT uses decision trees as the learning model and gradient descent to minimize the loss when adding trees.

III. RESULTS

Initially, we performed our simulation using NeuCube v1.3 software implemented in Matlab. However, we found that on using more than 500 samples for training and testing, more than 33GB of RAM is required to store the NeuCube output. Matlab deals with this requirement by not loading the complete output in the RAM at once, making the process inefficient and time-consuming.

To overcome the computational constraints, we have based our simulation of NeuCube on the SpiNNaker machine, using upto 800 samples for training and testing. All results presented here are simulated on SpiNNaker.

A. Classification Results

As mentioned above, 3 feature vectors are extracted from NeuCube viz. Spike Count (SC), Spikes Per Neuron (SPN) and Final Weights (FW). Combinations of these feature vectors are tested using the 6 classifiers discussed in Sec. II-E. The number of epochs used for training are gradually increased to observe the effect of increasing data samples on classification accuracy. We performed 5-fold cross validation to evaluate the performance of the classifiers due to limited samples. The results are compiled in Table I in the format Mean ± Standard Deviation of the accuracies obtained during cross validation. For each classifier, the parameters were optimised using grid-search, a standard optimization technique in data science. The best result is an accuracy of 81.25% achieved using the GBDT algorithm on 800 epochs, where the Spikes per neuron and Final Weights feature vectors are used.

B. Evaluation of Classification Errors

To analyse how well each class is being predicted, we computed confusion matrices of the results highlighted in Table I. These are presented in the left column of Fig. 7 for progressively increasing sample size. The sum of each row of the confusion matrix indicates the total number of samples that was tested for a particular epoch size; the total number of epochs available under each class being as shown in Fig. 1. Overall, the most confusion is observed between the N1 and N2 stages where samples from N1 stage get mislabeled as N2.

To have an objective understanding of the confusion matrices, we have presented a classification report for each one in the right column of Fig. 7. There are three measures that define the classification report, and are briefly discussed below in context to our result:

1) Precision: This measures how many of the epochs predicted as a certain class (sleep stage) indeed are of that class, i.e. 'True Positive', as opposed to actually belonging to a different class, but wrongly predicted, i.e. 'False Positive'. Thus, it can be defined as the ratio of the True positives to the sum of both True and False positives for the class. For example, in the Confusion Matrix for 800 epochs in Fig. 7(c), the Awake sleep stage has 58 True Positives and 5 False

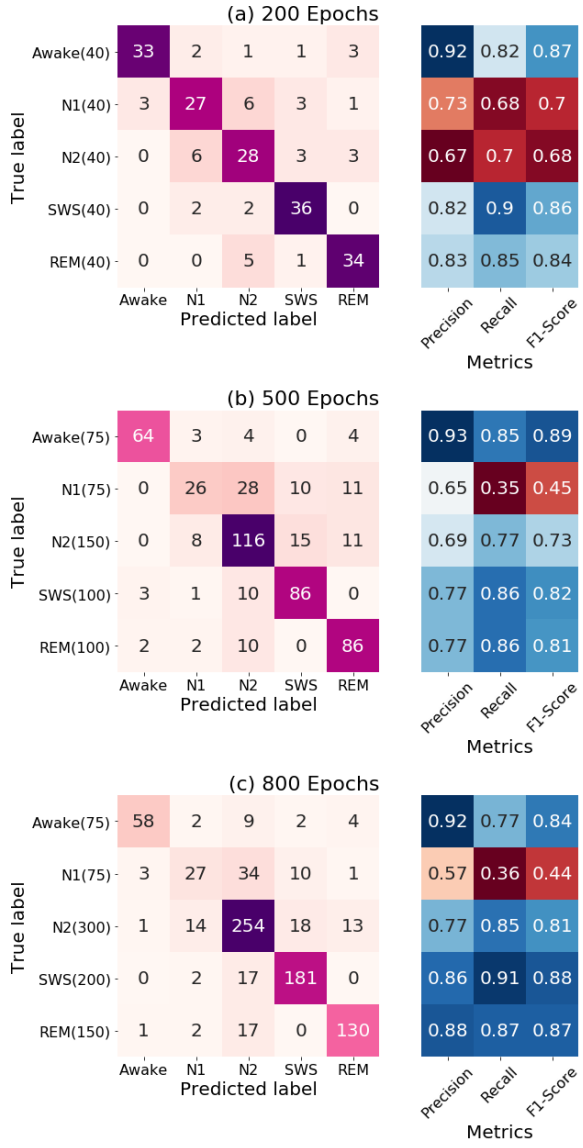

Fig. 7. Confusion matrix and Classification report for prediction by GBDT algorithm on (a) 200 epochs (b) 500 epochs (c) 800 epochs of sleep data

Positives (summing across the column) which results in a high precision of 0.92 in the classification report.

In our work, the Awake state has high precision across all epoch sizes. Conversely, the N1 class has a high number of False Positives with a low Precision of 0.57 in the study with 800 epochs.

2) Recall: This indicates how many of the epochs of a certain class indicated along the rows of the Confusion Matrix are correctly predicted as belonging to the class, i.e. True Positive, as opposed to incorrectly predicted as 'False Negative', i.e. belonging to a different class. Thus, it can be measured as the ratio of the True Positives to the total number of epochs of a class (i.e. the sum of True Positives and False Negatives).

Our results in Fig. 7 show a consistently high Recall for the SWS and REM sleep stages (≥ 0.85), and a consistently low recall (≤ 0.68) for the N1 sleep stage; instances of False Negatives for the N1 stage increase with increasing epoch sizes as shown in the Confusion Matrices of Fig.s 7(b) and (c) and indicated by Recall ≤ 0.36.

3) F1-Score: This is a harmonic mean of the Precision and Recall measures. It is high when both Precision and Recall are high and low when either of them is low. Thus, a high F1-Score implies low values for both False Negatives and False Positives, thus indicating a high predictive accuracy for a certain class.

For our data and simulation with 800 epochs, all sleep stages show a F1-score $\geq 80\%$ except for the N1 sleep stage, which has shown a low F1-score of 44%.

C. Interaction analysis using individual sleep stage data

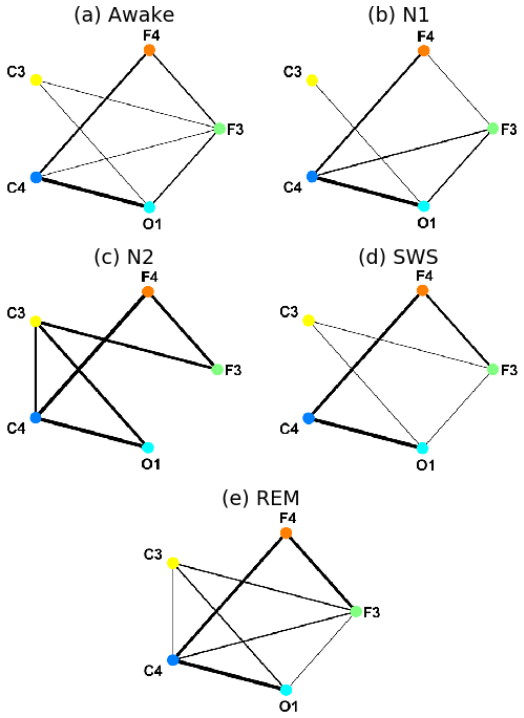

Fig. 8. Interaction between different EEG channels in NeuCube for each sleep stage, based on spike communication during STDP.

We analysed the information interaction between the EEG scalp electrodes across each sleep stage. The total temporal interactions (in terms of spike communication) between the 6 input neurons are depicted in Fig. 8 Thicker lines indicate more interaction between the inputs. These connections were established strongly because of more spikes transmitted between the neurons located in these areas, reflecting more changes in the corresponding EEG signals. The REM and Awake stage sleep have a similar pattern except for additional interaction between C3 and C4 during REM. The time series for both these sleep stages are known to be similar; we speculate that such similarity in the dynamic information being exchanged may indicate similar features extracted from the time-series data upon conversion to spike trains. Overall,

our study indicates plausibility of a more detailed study on relationship between different brain regions during sleep.

IV. Conclusion and Future Work

We have presented a preliminary study towards autonomous classification of sleep stages. We have used NeuCube running on SpiNNaker to train and test our model. Our results show that overall, for one instance of EEG data recorded from one individual, we are able to classify all but one (N1) of five sleep stages with an accuracy of approximately 80%. The misclassification of N1 as N2, which seems to be the main reason for the lower classification accuracy of N1, is justifiable on the grounds that the only difference between N1 and N2 in terms of scoring rules is the presence of spindles or K-complexes in N2. However, once an epoch of N2 has occurred, subsequent epochs are scored as N2 even in the absence of spindles and K-complexes, until a further classification change is made. That means that in practice, many N1 and N2 epochs will be indistinguishable, and so of course they cannot be reliably separated in classification unless the surrounding context (i.e. the fact that N2 was already being scored) is considered. Thus, this misclassification is actually an indicator of the goodness of our classification model. In continuation of the present work, we are looking into using a finite state machine as a means of resolving the observed confusion between N1 and N2 samples.

Overall, the use of the NeuCube framework enables using SNN-based classification of complex time-series signals such as EEG, and with relative ease of computation. The use of SpiNNaker has enabled us to use a larger sample size for training our model, which is indeed important for robustness of the classification study, as indicated with better predictive accuracy of our model.

We note that the $5 \times 5 \times 5$ grid of 125 neurons used in our work is set by trial and error and works better than a grid with 64 neurons. Indeed, we have not checked with larger grid sizes (e.g. 216 neurons) to keep lower computational times on SpiNNaker. Continuing work will look into increasing the size of the training network on NeuCube.

The sleep data used in this work is from one individual over a full night's sleep. To generalise our observation and to comment on what algorithm is best as a generic method for sleep classification, in our future work we will use a larger dataset of sleep.

V. Acknowledgment

The authors acknowledge the availability of KEDRI resources at the University of Auckland, New Zealand, and the SpiNNaker server via the Human Brain Project Platform, with support from the SpiNNaker team at the University of Manchester, UK — the work would not have been possible without the free availability of these resources. The authors would also like to thank the anonymous reviewers for their insightful comments.

References

[1] R. B. Berry, R. Brooks, C. E. Gamaldo, S. M. Harding, C. Marcus, B. V. Vaughn *et al.*, "The aasm manual for the scoring of sleep and associated events," *Rules, Terminology and Technical Specifications, Darien, Illinois, American Academy of Sleep Medicine*, vol. 176, 2012.

[2] R. Boostani, F. Karimzadeh, and M. Nami, "A comparative review on sleep stage classification methods in patients and healthy individuals," *Computer methods and programs in biomedicine*, vol. 140, pp. 77–91, 2017.

[3] S. Khalighi, T. Sousa, G. Pires, and U. Nunes, "Automatic sleep staging: A computer assisted approach for optimal combination of features and polysomnographic channels," *Expert Systems with Applications*, vol. 40, no. 17, pp. 7046–7059, 2013.

[4] U. R. Acharya, E. C.-P. Chua, K. C. Chua, L. C. Min, and T. Tamura, "Analysis and automatic identification of sleep stages using higher order spectra," *International journal of neural systems*, vol. 20, no. 06, pp. 509–521, 2010.

[5] Z. Cui, X. Zheng, X. Shao, and L. Cui, "Automatic sleep stage classification based on convolutional neural network and fine-grained segments," *Complexity*, vol. 2018, 2018.

[6] A. Supratak, H. Dong, C. Wu, and Y. Guo, "Deepsleepnet: A model for automatic sleep stage scoring based on raw single-channel eeg," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 25, no. 11, pp. 1998–2008, 2017.

[7] J. Behrenbeck, Z. Tayeb, C. Bhiri, C. Richter, O. Rhodes, N. Kasabov, J. I. Espinosa-Ramos, S. Furber, G. Cheng, and J. Conradt, "Classification and regression of spatio-temporal signals using neucube and its realization on spinnaker neuromorphic hardware," *Journal of Neural Engineering*, vol. 16, no. 2, p. 026014, feb 2019.

[8] T. Penzel and R. Conradt, "Computer based sleep recording and analysis," *Sleep medicine reviews*, vol. 4, no. 2, pp. 131–148, 2000.

[9] N. K. Kasabov, "Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data," *Neural Networks*, vol. 52, pp. 62–76, 2014.

[10] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The spinnaker project," *Proceedings of the IEEE*, vol. 102, no. 5, pp. 652–665, 2014.

[11] A. P. Davison, D. Brüderle, J. M. Eppler, J. Kremkow, E. Muller, D. Pecevski, L. Perrinet, and P. Yger, "Pynn: a common interface for neuronal network simulators," *Frontiers in neuroinformatics*, vol. 2, p. 11, 2009.

[12] O. Rhodes, P. A. Bogdan, C. Brenninkmeijer, S. Davidson, D. Fellows, A. Gait, D. R. Lester, M. Mikaitis, L. A. Plana, A. G. Rowley *et al.*, "spynnaker: a software package for running pynn simulations on spinnaker," *Frontiers in neuroscience*, vol. 12, p. 816, 2018.

[13] S. J. Durrant, S. A. Cairney, C. McDermott, and P. A. Lewis, "Schema-conformant memories are preferentially consolidated during rem sleep," *Neurobiology of Learning and Memory*, vol. 122, pp. 41–50, 2015.

[14] B. Petro, N. Kasabov, and R. M. Kiss, "Selection and optimization of temporal spike encoding methods for spiking neural networks," *IEEE transactions on neural networks and learning systems*, 2019.

[15] N. Kasabov, N. Scott, E. Tu, S. Marks, N. Sengupta, E. Capecci, M. Othman, M. Doborjeh, N. Murli, R. Hartono *et al.*, "Design methodology and selected applications of evolving spatio-temporal data machines in the neucube neuromorphic framework," *Neural Netw*, vol. 78, pp. 1–14, 2016.

[16] B. Schrauwen and J. Van Campenhout, "Bsa, a fast and accurate spike train encoding scheme," in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, vol. 4. IEEE, 2003, pp. 2825–2830.

[17] L. Koessler, L. Maillard, A. Benhadid, J. P. Vignal, J. Felblinger, H. Vespignani, and M. Braun, "Automated cortical projection of eeg sensors: anatomical correlation via the international 10–10 system," *Neuroimage*, vol. 46, no. 1, pp. 64–72, 2009.

[18] N. K. Kasabov, *Time-space, spiking neural networks and brain-inspired artificial intelligence*. Springer, 2018, vol. 7.

[19] E. Tu, N. Kasabov, and J. Yang, "Mapping temporal variables into the neucube for improved pattern recognition, predictive modeling, and understanding of stream data," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 6, pp. 1305–1317, 2016.

[20] G.-q. Bi and M.-m. Poo, "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type," *Journal of neuroscience*, vol. 18, no. 24, pp. 10 464–10 472, 1998.