Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 4203 – 4219
© TÜBİTAK
doi:10.3906/elk-1903-186

GACNN SleepTuneNet: a genetic algorithm designing the convolutional neural network architecture for optimal classification of sleep stages from a single EEG channel

Shahnawaz QURESHI[1], **Seppo KARRILA**[2], **Sirirut VANICHAYOBON**[1*]
[1]Information Systems Technology and Applied Research Laboratory (*i*Star), Department of Computer Science, Faculty of Science, Prince of Songkla University, Songkhla, Thailand
[2]Department of Chemistry, Faculty of Science and Industrial Technology, Prince of Songkla University, Surat Thani, Thailand

Received: 30.03.2019 • **Accepted/Published Online:** 02.08.2019 • **Final Version:** 26.11.2019

Abstract: This study presents a method for designing–by a genetic algorithm, without manual intervention–the feature learning architecture for classification of sleep stages from a single EEG channel, when using a convolutional neural network called GACNN SleepTuneNet. Two EEG electrode positions were selected, namely FP2-F4 and FPz-Cz, from two available datasets. Twenty-five generations were involved in diagnosis without hand-crafted features, to learn the architecture for classification of sleep stages based on AASM standard. Based on the results, our model not only achieved the highest classification accuracy, but it also distinguished the sleep stages based on either of the two EEG electrode signals, in both datasets. The results show that our model performed the best with highest overall accuracy rates and kappa statistic (CAP sleep: 95.61% and 0.94; Sleep EDF: 92.51% and 0.90) among other state-of-the-art methods that require no manual intervention. Our model could automatically learn the features for classification of sleep stages, for different raw EEG electrode positions in different datasets, without user-assisted feature extraction.

Key words: Without hand-crafted feature extraction, electroencephalogram (EEG), genetic algorithm, convolutional neural network

1. Introduction

Convolutional neural networks (CNNs) have become a popular machine learning technique that can automatically learn informative features in various applications, including image processing, speech and language processing, and medical image analysis. A CNN relies upon its deep architecture, which enables extracting a set of perceptive features at multiple stages of abstraction. The deep architecture thereby eliminates laborious feature engineering, and instead automatically learns transforms of the inputs that are appropriate for the classification problem by involving so-called "convolutional filters" in an end-to-end learning process. Interest in using CNNs for biosignal-related problems [1–3] has begun to emerge in the context of EEG signals.

In spite of their success, designing a CNN architecture is a challenging task requiring decisions on several design parameters, like the depth of network, the type and parameters of each layer, and the connectivity of the layers. State-of-the-art CNN architectures have become deep and complex, which practically means that a substantial number of design parameters should be tuned for best performance with given type of data. Both trial-and-error explorations and expert knowledge contribute to creating appropriate architectures for an

[*]Correspondence: siriut.v@psu.ac.th

application context. Because of this situation, automatic design methods for CNN architectures can provide significant advantages. In this paper we attempt to design the CNN architecture without human intervention in feature extraction by using a genetic algorithm, in the context of automatic sleep stage scoring from a single EEG channel signal. Efficient diagnosis and treatment of patients with sleep-related diseases is presently an important research topic in the healthcare community. Getting sufficient good quality sleep at night is significantly associated with human mental and physical health. Sleep disturbances may be caused by a variety of disorders, such as sleep apnea, insomnia, or narcolepsy.

Neuroscientists assess sleep by using electrical signals from sensors affixed to different parts of the human body. These electrical signals can fall into several types, such as electroencephalogram (EEG), electrooculogram (EOG), electromyogram (EMG), or electrocardiogram (ECG). Once such data have been collected, sleep scoring is the first step in the diagnosis of a sleep disorder. When EEG is used as an inspection technique to identify the stages of sleep by scoring, this is known as polysomnography (PSG). Traditionally, sleep scoring is done by expert sleep technicians, based on visual subjective perceptions of the signals. The commonly used guidelines in this task are Rechtschaffen and Kales (R&K) and American Academy of Sleep Medicine (AASM) guidelines (https://aasm.org/clinical-resources/practice-standards/practice-guidelines/). According to the R&K guidelines, the recordings should distinguish between six sleep stages, namely wake stage (W), nonrandom eye movement stage 1 (NREM 1), nonrandom eye movement stage 2 (NREM 2), nonrandom eye movement stage 3 (NREM 3), nonrandom eye movement stage 4 (NREM 4), and random eye movement (REM). The AASM guidelines essentially modify these to five stages by combining stages NREM 3 and NREM 4 into N3 stage. These two common nomenclatures for distinct sleep stages are compared side by side in Table 1.

Table 1. Two common standards for sleep stages.

R&K	W	NREM1	NREM2	NREM3	NREM4	REM
AASM	W	N1	N2	N3		REM

Visual sleep state recognition by inspecting the EEG signals is challenging for several reasons. Currently standard sleep stage evaluation is costly, as it requires labor by experts. Furthermore, the massive amounts of data that are analyzed per patient make sleep scoring by human experts exhausting and subject to misclassifications due to tiredness. To overcome these problems with manual sleep scoring, several studies have proposed automating the identification of sleep stages, based on single or multiple signal channels. However, recent studies show promising results in the classification of sleep stages using deep learning, a comparatively novel branch of machine learning that is still evolving rapidly [2, 3].

A systematic literature review covering the period from 1968 to 2018 (Shahnawaz Qureshi, unpublished data) is currently under review. According to it there is a need to identify methods for tuning the structural parameters of a CNN, so as to maximize its performance in classifying sleep stages. We found that most studies did not utilize optimization methods for setting the algorithm parameters to classify sleep stages. Generally, machine learning and deep learning techniques do not yield optimal outcomes without being adequately tuned. To achieve a high classification accuracy, it is not only required to choose an appropriate machine learning algorithm: it can be very time-consuming to set the algorithm parameters properly. To fill this gap, we propose to have a genetic algorithm (GA) to diagnose the convolutional neural network architecture without manual intervention in feature extraction, for the classification of human sleep stages based on AASM standard. In recent years, combining GA with deep learning has attracted considerable attention. Mainly, GA has been employed in the automatic selection of hyperparameters (e.g., learning rate), other parameters (e.g., kernel size), and network structure [4, 5] in different research areas. To the best of our knowledge, this is the first study using a GA for automatically choosing the best-pretrained CNN for sleep stage classification. We tested this approach using two datasets, CAP Sleep and SleepEDF, and two electrode positions from each dataset, namely FP2-F4 and FPz-Cz. The ability to find high-quality network structures by this approach was assessed. Finally, we employed the best CNN network architecture found and verified its effectiveness.

The paper is organized as follows. Section 2 briefly introduces theory and reviews relevant literature on the classification of sleep stages by machine learning techniques, with or without human interventions in feature extraction. Section 3 presents GACNN SleepTuneNet and how the genetic algorithm was used to design the CNN network architecture. Section 4 presents the results and discussion. Finally, the conclusion is given in Section 5.

2. Related work

A number of prior studies have sought to develop automated sleep stage scoring based on multiple signals, such as EEG, EOG, and EMG [6–8], or based on single-channel EEG [9–11]. There are two different ways for classifiaction of sleep stages including with and without human intervention in context of feature extraction. When performing feature extraction with manual intervention, developing an automatic sleep stage classification comprises the following steps: data preprocessing, feature extraction, and classification. Various techniques have been applied in the literature, such as K-nearest neighbors (K-NN) [12], support vector machine (SVM) [13–15], decision tree (DT) [14, 16, 17], and random forest algorithm (RF) [13, 18].

On the other hand, another way to extract the features from EEG signals is by utilizing the deep learning techniques without human intervention. Deep learning can learn the features from EEG signals without human intervention for classification of sleep stages. Recently, deep learning, a branch of machine learning that employs multiple layers of linear and nonlinear processing units to learn hierarchical representations or features from input data has been applied in sleep stage scoring. The authors of [19] have investigated the ability of deep belief nets (DBNs) to learn probabilistic representations from preprocessed raw PSG. They applied the AASM standard classes and obtained F-scores ranging from 31% to 84% in 25-fold leave-one-out cross-validation. CNNs have also been applied to learn multiple filters that are used to convolve with small portions of input data from a single EEG channel [20]. They achieved overall accuracies in the range 71–76% for all subjects in 20 fold cross-validation. Another study [21] proposed a CNN model (DeepSleepNet) consisting of four convolutional layers and two max pooling layers, a dropout layer with probability (0.5), batch size (100), Adam optimizer's parameters 10-4, 0.9 and 0.999, and softmax function. They evaluated their model using two public datasets: Montreal Archive of Sleep Studies (MASS) and Sleep-EDF, achieving F1-scores 86.2–81.7% in the former data and 82.0–76.9% in the latter. Orestis Tsinalis et al. [9] proposed stacked sparse autoencoders (SSA, a special type of neural network) for automatic sleep scoring. They achieved overall 78% accuracy in 20-fold cross-validation. Recurrent neural network (RNN) was proposed in [22] for automatic sleep scoring with energy features of a single EEG channel; a 10-fold cross-validation indicated 87.2% accuracy. Hao Dong et al. [23] applied a mixed neural network approach to the classification of sleep stages. They utilized a rectify neural network and a recurrent neural network with softmax function with single EEG channel inputs, obtaining an accuracy of 85.92%. While the CNN techniques have not been spread rapidly in the biomedical engineering area, they have advantages in the classification of one-dimensional biosignals, such as (EEG) and (ECG). As the literature reviewed above shows, there is growing interest in the use of CNN with EEG signals. In this study we propose a GACNN SleepTuneNet for appropriate selection of parameters in a classifier that performs sleep scoring without human intervention.

3. GACNN SleepTuneNet

Our proposed method, GACNN SleepTuneNet, uses a genetic algorithm to design a competitive network structure for sleep scoring. Its objective function can be set as maximizing the classification accuracy, while controlling the complexity of the created method. The GACNN SleepTuneNet comprises a GA used for selecting the parameters and hyperparameters of a competitive CNN architecture. After obtaining the optimal parameters and hyperparmeters, the CNN is used to classify sleep stages. The details of each process are discussed below. An overview schematic of the GACNN SleepTuneNet is shown in Figure 1.

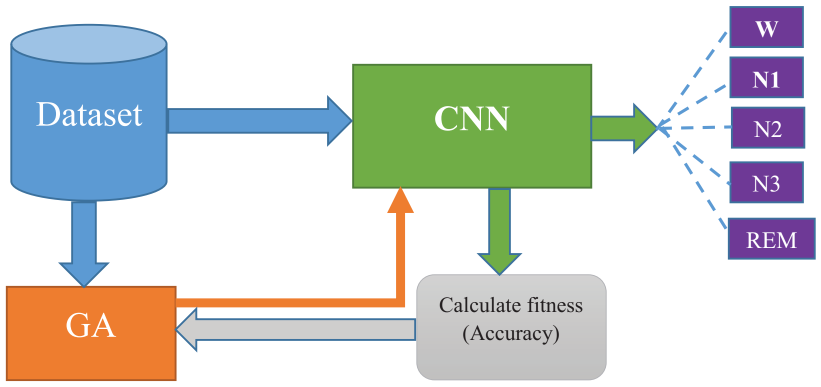

Figure 1. Overview of GACNN SleepTuneNet architecture. The GACNN architecture is trained on a learning task and assigned the validation accuracy of the trained model as its fitness. The evolutionary algorithm searches for the best architecture.

3.1. Dataset

The data used in this study to evaluate our method with different EEG channels came from two public databases available on Physionet repository [24]: CAP sleep and Sleep-EDF.

3.1.1. CAP sleep

This had 108 recordings of 3 EEG, 2 EOG, and EMG channels. The EEG electrodes were set up according to the 10-20 international system. We selected randomly 20 cases that contained FP2-F4 EEG channel (14 females and 6 males, aged 18–72). The sleep stages were annotated into seven classes (W = wake, NREM1–NREM4 = sleep stages, R = REM, MT = body movements) by a sleep expert according to the R&K guidelines [25]. We evaluated our model using the FP2-F4 EEG channel and excluded body movement epochs from our dataset. We also converted the six sleep stages into the five sleep stages of the AASM guideline.

3.1.2. Sleep EDF

These data are from two electrodes (Fpz-Cz and Pz-Cz) and two studies, namely of age effect in healthy subjects (SC) and of Temazepam effects on sleep (ST). In this study we used 20 subjects (10 males and 10 females, aged 25–34) from SC. A sleep expert scored these sleep stages according to the R&K guideline into one of the eight classes (W, NREM1, NREM2, NREM3, NREM4, REM, MOVEMENT, UNKNOWN). In this study, we used the FPz-Cz channel (not Pz-Cz) without any preprocessing. We merged NREM1 and NREM4 stages under N3 stage as recommended by (AASM) [26], as for the EDF dataset, and also excluded a small number of MOVEMENT and UNKNOWN epochs. Single-channel EEG (FPz-CZ) was divided into 30-s length samples (epochs) each representing only a single sleep stage. Table 2 summarizes the epoch counts by AASM standard sleep stage.

Table 2. The epoch counts by AASM standard sleep stage and their total numbers.

Dataset	W	N1	N2	N3	REM	Total
CAP sleep	5760	1097	11,155	7304	3548	28,864
Sleep-EDF	7920	2802	17,781	5700	7697	41,900

3.2. Genetic algorithm (GA)

In other application fields, on developing deep learning algorithms, researchers have studied using GA in the tuning of learning parameters [4, 5] network structure [27], and hyperparameters [28] of deep neural networks. GA uses an adaptive heuristic search algorithm inspired by the process of natural selection and genetics. GA is robust and offers significant benefits in solving some optimization and search problems, and includes reproduction, crossover, and mutation operations. It requires two essential elements: a genetic representation of the solution domain, and a fitness function to evaluate each individual. GA is widely used in business, science, and engineering: a good example is the traveling salesman problem [29]. Generally, evolutionary algorithms have also been applied to hyperparameter optimization problems [30]. The principal concept of a genetic algorithm is to generate a variety of individuals by genetic operations such as mutation and crossover, and apply selection pressure that directs evolution of the population to improve fitness.

In this work the genetic algorithm is only used to propose network architecture, parameters and hyperparameters, while the classification accuracy of each architecture is obtained using individual training from scratch. Our Algorithm 1 has the following steps to evolve CNNs based on GA.

Algorithm 1 GA searching for CNN

```
1: Input: Dataset D
2: Generate a random population of N chromosomes from the reference dataset D.
3: Evaluate the fitness function from CNN.
4: Create a new population of chromosomes by iterating the following steps until the new population is
   complete.
5: Select some chromosomes from the current population based on classification accuracy (fitness) by using
   the roulette wheel process.
6: For crossover, select the most fit chromosomes by elitism and compute the next generation.
7: Apply mutation with probability rate 0.03.
8: Test the classification accuracy; if the perfect classification is achieved then stop the process.
9: Go to step 3.
```

3.2.1. Chromosome evaluation

In this step, we apply the CNN-based fitness function to assess the chromosomes. The highest classification accuracy (fitness) is sought through the design of the fitness function, which is also aware of the number of parameters and hyperparameters in the CNN. Accuracy is here defined as the count of correct classification calls divided by the total number of classification calls. Since we are dealing with multiclass classification, the performance measure is the average accuracy α.

$$\alpha = \frac{\sum_{i=1}^{l} \frac{tp_i+tn_i}{tp_i+tn_i+fp_i+fn_i}}{l} \quad (1)$$

where l is the number of classes, tp_i is true-positive count, fp_i is false-positive count, fn_i is false-negative count, and tn_i is true-negative count for an individual class C_i.

3.2.2. Selection operator

In selection operation, the GA does not pick only the top two but selects an entire subpopulation with a lottery, in which the probability to be chosen is determined by the fitness score. The goodness of each individual depends on its fitness. Fitness is a numeric ”function” characterizing an individual in the population, and is here the accuracy of classification reached by training a CNN matching the individual in the GA population. The selection should prefer better chromosomes that are passed on to the next generation.

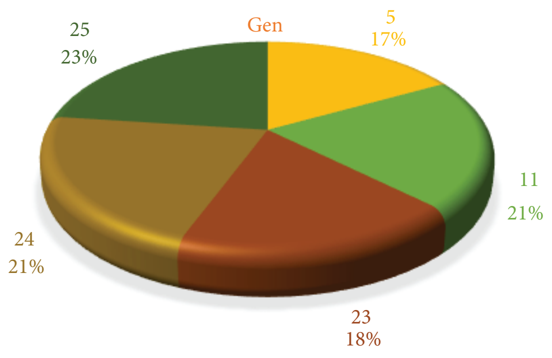

Figure 2. Highest fitness scores randomly generated by Russian roulette wheel.

In practice, various selection methods, such as roulette wheel selection, tournament selection, boltzmann selection, can be used to determine which individuals survive. We performed Russian roulette wheel selection. The higher the fitness is, the more likely an individual survives and can breed, while those with poor performance are more likely to be screened off from the population. The Russian roulette wheel is shown in Figure 2, representing random generation for the highest fitness function (accuracy) in two different datasets.

3.2.3. Crossover

The generation of successors in a GA depends on the two important operators: crossover and mutation [31]. The crossover operation represents the exploitation part of the search algorithm. It takes selected chromosomes for mating to combine the most fit chromosomes in the next generation of offspring. However, if the new population were created only by crossover with mutations, the best chromosomes of the current generation could be lost; there would be no guarantee of consistent improvement. This problem is addressed by elitism [31], preserving the best individuals of each population also unchanged; they deserve not only the right to breed, but also extended longevity for themselves. Therefore, the N most fit individuals were copied straight into the next generation.

3.2.4. Mutation

The mutations used in a genetic algorithm introduce diversity within the population, providing a multitude of ”search directions”, and also helping avoid getting stuck at a local optimum. Mutation represents the exploration part of the search algorithm, and allows discovering unexplored areas with the intent of finding better and better solutions. In the random mutation, a gene is bit flipped, so that 1 becomes 0 or 0 becomes 1. The selection of cases subject to mutations ensures that the best chromosome is never lost in the optimization process. Too high mutation rate gives slow convergence. However, for finding the effects on accuracy, we tested different mutation rates and found that probability 0.03 gave robust results. The rate of mutations used in this study was set at 0.03.

3.3. Convolutional neural network architecture (CNN)

Recently CNN has gained interest and shown promise in 1D biomedical signal problems [32]. We chose CNN as the model type, for architecture selection and hyperparameter optimization, partly due to its current popularity in the machine learning community.

Our CNN architecture comprises layers, namely (1) convolutional layer, (2) pooling layer, (3) normalization layer (using an activation function), and finally (4) a fully connected layer. The hyperparameter optimization approach often tunes predefined hyperparameters, such as the numbers of layers and neurons, and the type of activation functions. However, we propose different layers with different parameters as discussed below. The complete GACNN SleepTuneNet architecture is presented in Figure 3.

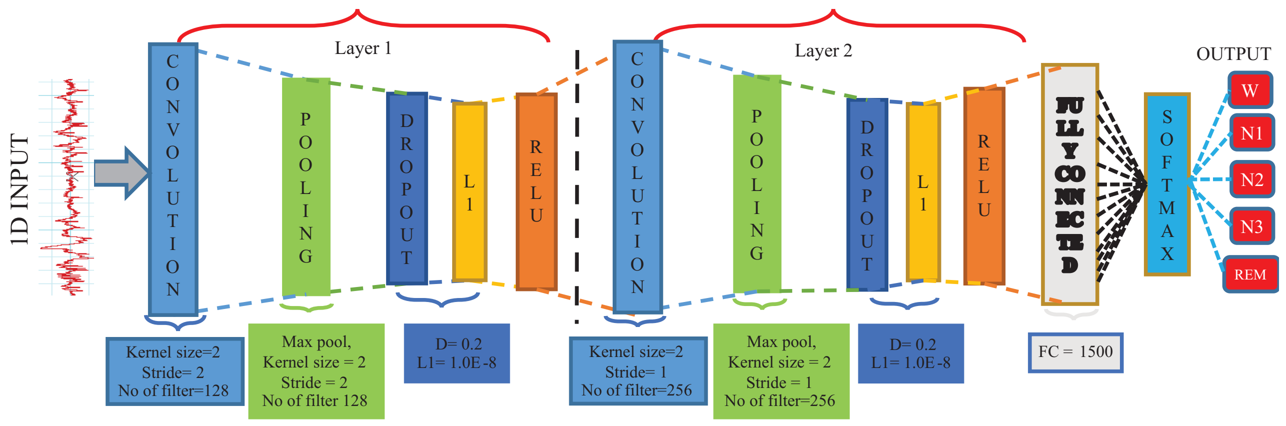
Layer 1
Layer 2
OUTPUT
1D INPUT
CONVOLUTION
POOLING
DROPOUT
L1
RELU
CONVOLUTION
POOLING
DROPOUT
L1
RELU
FULLY CONNECTED
SOFTMAX
W
N1
N2
N3
REM
Kernel size=2 Stride= 2 No of filter=128
Max pool, Kernel size = 2 Stride = 2 No of filter 128
D= 0.2 L1= 1.0E-8
Kernel size=2 Stride= 1 No of filter=256
Max pool, Kernel size = 2 Stride = 1 No of filter=256
D= 0.2 L1= 1.0E-8
FC = 1500

Figure 3. An overview of GACNN SleepTuneNet architecture.

3.3.1. Convolutional layer

Convolution layer is a core building block of the convolutional network [33]. It receives the inputs and consists of a set of learnable filters. Every filter is small relative to the total input dimensionality. When dealing with high-dimensional inputs, we connect each neuron to only a local region of the input array. The spatial extent of this connectivity is a hyperparameter called the receptive field of the neuron (equivalently this is the filter size). The convolution operation is denoted by *. The mathematics behind the CONV layer is to perform the elementwise product between the filter and the input size. Its hyperparameters include the kernel size (K).

Thus, our case of interest is 1D signal and size of the result is;

$$Output = N_H - K + 1 \tag{2}$$

where N_H is the height or input volume size of signals and K is the kernel size of the sliding window that convolves the data. Let us see how the convolution acts on an input. Our interest is 1D signal convolutional layer with 100 feature detectors, input size (N) is 500, and kernel size (K) is 20 applied in each feature detector, the window will slide through the data for 481 steps (500-20+1), giving output size 481*100. There are other parameters involved in the convolutional or pooling layer which can modify the behavior of the layer, such as padding (P) and stride rate (S). The effects of these parameters are described by:

$$Output = \left\lceil \frac{N_H + 2P - K}{S} + 1 \right\rceil \tag{3}$$

where N_H represents height or input size of the 1D signal, P is padding, K is the kernel size, and S is the stride.

Formally, each layer l convolves the set $(X)^{l-1}$ of its input feature map by set of trainable kernels (also known as filters) W^l. With $(H)^{l-1}$ the number of input feature maps and H^l the number of output feature maps, and K^l the height of the kernel, W^l has form (K^l, $(H)^{l-1}$, H^l). As inputs have one channel only, H^0 equals 1. Thus, x^l_j indicates the jth feature map in X^l, and w^l_{ij} the piece of W^l that relates input feature map i to output feature map j. Using such notation, we get:

$$x_j^{(l)} = \sigma\left(\sum_{i=1}^{H^{(l-1)}} x_i^{l-1} * W_{ij}^{(l)}\right) \tag{4}$$

where σ represent nonlinear activation function and * denotes the convolution operator.

3.3.2. Pooling layer (subsampling)

The primary purpose of pooling is to reduce or downsample the input representation, to reduce the number of parameters and computational load in the network, and also to control overfitting. The pooling layer operates independently on every depth slice of the input and resizes it spatially, using the Max operation [21, 34]. In maxpooling there are two hyperparameters: kernel size (K) and stride rate (S). The pooling can be performed with other functions, for example by average or sum pooling, but the max pooling has worked better in practice. Moreover, two variations of max pooling are commonly seen in practice: a pooling layer with K = 2, S = 3 (called overlapping) and K = 2, S = 2.

3.3.3. Regularization

In CNN overfitting is considered a serious problem due to its large number of tunable parameters. The CNN contains multiple nonlinear hidden layers, and these hidden layers create a useful model which can learn the complex relationship between input and output. However, these many relationships can learn noise and negatively impact classification performance. This leads to overfitting, and regularization techniques can help prevent overfitting [33]. The most commonly used regularization techniques are dropout and weight decay. The word ”dropout” refers to dropping out hidden or visible units. This technique is used in CNN to avoid overfitting the training data by dropping out neurons with probability P >0 [33]. The optimal probability is not known generally. On the other hand, another most common kind of regularization technique is known as weight decay (L1) and this uses the sum of the absolute values of the weights, and the related hyperparameter is denoted by lambda or sometimes by alpha [33]. Weight decay makes sure that the weights are not too large and the model is not overfitting the training set. The range of hyperparameter α is from 0.0 to 1.0 and this controls the bias of the model. In this study, we employed two regularization techniques for avoiding overfitting. The first one is dropout [34], and we assessed dropout rates in the range 0.1–0.7 and found that 0.2 gave the best performance. The second technique is weight decay (L1) [33] that prevents exploding gradients. Weight decay (L1) rates were tested in the range from 1.0E-1 to 1.0E-10, and 1.0E-8 showed the best performance.

3.3.4. Normalization

Many types of normalization have been proposed for use in ConvNet architecture, such as tanh, sigmoid, and the ReLu function. Due to its good performance in most situations, we chose to use ReLU [20], which stands for rectified linear unit. ReLU is an element-wise operation (applied per pixel in an image array) and replaces all negative pixel values in the feature map with zeroes. It is computationally efficient and converges much faster than sigmoid/tanh in practice. ReLU is defined as follows:

$$f(z) = max(0, z). \tag{5}$$

3.3.5. Fully connected layer

The last layer connected into the CNN architecture is fully connected and computes the dot product between its weights and the previous layer's outputs to obtain the probability scores for different classes [35]. The convolutional layers and fully connected layer are trained by the so-called backpropagation algorithm, which adjusts the parameters (weights) in these layers so as to minimize the output error. However, the basis of backpropagation is the error in the final answer, which is used to determine how much the network adjusts by updating the weights using gradient descent. The most commonly used type of gradient decent algorithms, in practice, is the stochastic gradient descent (SGD) [33]. It is an iterative optimization technique to minimize the error in the network outputs. The benefit of the SGD is that it is simple to implement and fast for several training problems [33]. Thus, the weight is updated as follows:

$$W = w - \alpha\left(\frac{\partial Error}{\partial w}\right) \tag{6}$$

where W represent the new updated weight, w is the old weight, α is the learning, and $\frac{\partial Error}{\partial w}$ is the derivate of error with respect to this weight.

3.3.6. Softmax approach

Softmax is useful in case there are more than two classes in the data, so we use the softmax [36] function at the end of architecture. It takes as input a vector of scores and produces output based on the probability distribution by using softmax function. Thus, in our case, we have five classes, and we can compute the probability estimate of each class according to the softmax formula:

$$a^{(l)} = \frac{e^{z_l}}{\sum_{j=1}^{5} e^{z_j}} \tag{7}$$

where z^l is a (5,1)-dimensional array (a vector) and a^l is the output vector.

Formally, suppose we have vector size (5) representing different classes such as W, N1, N2, N3, and REM. The indexing we use for these is (0,...,4); from zero to C-1, where C is the number of classes. In this case, we want to build a network where the output layer has C = 5 output units. Thus, $z^l = 5$ is the number of units of the output layer, which is layer l. The units in output layer indicate the probability of each of the five classes: the output is $P(W \mid x), P(N1 \mid x), P(N2 \mid x), P(N3 \mid x), P(REM \mid x)$.

In this study, we consider convolutional neural network architecture design as a model selection problem, or a hyperparameter optimization problem, in the context of classifying sleep stages. In this approach we tune the predefined hyperparameters, such as the numbers of layers and neurons, and the types of activation functions. Our CNN SleepTuneNet is based on the following configuration. Two convolutional layers; the first with kernel size 2 and number of filters 128 and stride rate 2 with valid padding. Max pooling is applied to reduce the computational cost with kernel size 2 and stride rate 2. Then second convlayer has 256 filters with kernel size 2, stride rate 1, and valid padding. Then maxpooling layer was set with kernel size 2 and stride rate 1. ReLU activation function and dropout with 0.2 probability are utilized in our CNN SleepTuneNet architecture. One fully connected layer of size 1500 and finally the softmax layer compute the probability of each sleep stage. The model is trained using mini batch-size = 128 by stochastic gradient descent with learning rate (0.01). The regularization techniques dropout and weight decay (L1) were applied to prevent overfitting, with parameters (0.2) and (1.0E-8). The combination of layers can be summarized as $Input \rightarrow [Conv] \rightarrow [Pool] \rightarrow [Dropout][weightdecay] \rightarrow [ReLU] \rightarrow [Conv] \rightarrow [Pool] \rightarrow [Dropout][weightdecay] \rightarrow [ReLU] \rightarrow FullyConnected \rightarrow [Softmax] \rightarrow Output$. A full network description is shown in Algorithm 2.

3.4. Experimental setup

The numerical experiments were run on a machine with 4.20 GHz CPU, 32.0 GB RAM, equipped with a NVIDIA GeForce GTX 1050 graphical processing unit. The training time was approximately 2 h for each validation fold. To assess the generalizability of our classification system, we evaluated our model with 10-fold cross-validation for the AASM standard. We partition the entire dataset based on the subjects into 70% and 30% for training and validation, respectively. This avoids overfitting over the subjects and supports generalization. In this procedure, we choose three randomly selected subjects as validation data and the remaining 17 subjects for training. Thus, each fold is created with a 85/15 split. With the help of GA, we generated 25 individual populations with different hyperparameters to find the optimal settings for calling AASM standard labels for two datasets. These populations each had several network configurations, created by increasing/decreasing convolutional layer size, and changing stride size and max-pooling size. As regards the regularization parameters, we tried several weight decay parameters ranging from 1.0E-1 to 1.0E-10. Among these 1.0E-8 showed the best performance. In training by backpropagation the learning rate was tested from 0.1 to 0.00001, and 0.01 showed the best performance.

3.5. Sleep scoring performance evaluation

To assess AASM scoring performance we assessed the confusion matrix. However, it was scaled to class balanced form, in order to place an equal weight on each class. The metrics we computed are precision, recall, F1-score, per stage accuracy, overall accuracy, per stage classification error and kappa statistic. However, we have five classes, and each was assessed as a binary classification problem of one vs. all other classes. The formulas for the metrics are provided next.

Algorithm 2 GACNN SleepTuneNet

```
1: Input: 1D EEG Signals, GA
2: Output: Five sleep stages
3: Forward Propagation Layer 1:
4:     Conv1D 1 ← $K_s$ = 2, S = 2, No. of filter = 128;        ▷ $K_s$ =Kernel size, S = Stride
5:     MaxPool 1 ← $K_s$ = 2, S = 2;
6: Regularization:
7:     Dropout ← D = 0.2 & Weight decay ← L1= 1.0E-8;        ▷ Dropout = D & Weight decay = L1
8: Activation Function:
9:     ReLU ($\sigma$) ← max(0,0);
10: Forward Propagation Layer 2:
11:     Conv1D 2 ← $K_s$ = 2, S = 1, No. of filter = 256;
12:     MaxPool 2 ← $K_s$ = 2, S = 1;
13: Regularization:
14:     Dropout ← D = 0.2 & Weight decay ← L1= 1.0E-8;
15: Activation Function:
16:     ReLU ($\sigma$) ← max(0,0);
17: for all subject in data do
18:     $x_j^l = \sigma\left(\sum_{i=1}^{H^{(l-1)}} x_i^{l-1} * w_{ij}^l\right)$
19: end for
20: Fully connected ← $f_c$ = 1500
21: Sleep stages probability ← Softmax ($f_c$)
22: for each class probability do
23:     $a^{(l)} = \frac{e^{z_l}}{\sum_{j=1}^{5} e^{z_j}}$
24: end for
25: Hyperparameters:
26: learning rate ← $L_r$ = 0.01, Max iteration ← $M_i$ = 250
```

3.5.1. Precision

This is the fraction of correct positive calls among all positive calls:

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

3.5.2. Recall

Recall is the fraction of true positives that got called positive by the classifier:

$$Recall = \frac{TP}{TP + FN}. \tag{9}$$

3.5.3. F1-score

F1-score is the harmonic mean of precision and recall:

$$F1 - Score = \frac{2 * Recall * Precision}{Recall + Precision}. \tag{10}$$

3.5.4. Accuracy

This metric describes the correctness of each case, and is the count of correct classification of each case divided by the total number of each case:

$$Accuracy = \frac{No\ of\ each\ case\ correctly\ classified}{Total\ number\ of\ each\ case}. \tag{11}$$

3.5.5. Overall accuracy

This is the number of epochs correctly classified as fraction of the total number of epochs:

$$Overall\ Accuracy = \frac{No\ of\ epochs\ correctly\ classified}{Total\ number\ of\ epochs}. \tag{12}$$

3.5.6. Error rate

The rate of misclassifications for each sleep stage:

$$Error\ rate = 1 - accuaracy. \tag{13}$$

3.5.7. Kappa statistic

This metric is useful with multiclass classification problems and indicates whether a classifier performs better than mere random calls based on the frequency of each class. Landis and Koch (1977) [37] presented an interpretation and naming convention for this performance measure: $<$ 0 for ”no”, 0–0.20 for ”slight”, 0.21–0.40 for ”fair”, 0.41–0.60 for ”moderate”, 0.61–0.80 for ”substantial”, and 0.81–1 for ”almost perfect” agreement.

4. Results

4.1. EEG channel (FP2-F4) from CAP dataset

Table 3 shows the confusion matrix, obtained from 10-fold cross-validation with the single EEG channel (FP2-F4) input and target labelling by AASM standard. The elements in the matrix are counts of 30-s EEG epochs, the actual labels are those decided by a sleep expert, and the classifier calls are from the CNN SleepTuneNet model. The bold numbers on the diagonal highlight the correctly classified epochs.

Table 3. Confusion matrix from cross-validation on using the (FP2-F4) electrode as input and labelling based on AASM standard.

Predicted						
	Stages	W	N1	N2	N3	REM
Actual	W	**5520**	84	112	28	16
	N1	107	**771**	186	4	29
	N2	52	68	**10,810**	152	73
	N3	18	3	187	**7091**	5
	REM	17	17	98	10	**3406**

Table 4 shows separate performance metrics for each sleep stage in the AASM standard. It can be seen that N3 stage was called with the highest accuracy 0.97, and classification error rate around 0.03. N2 and REM ranked second with 0.96 accuracy and error rate 0.04. W also shows good performance with 0.95 accuracy.

Table 4. Performance metrics for each AASM standard sleep stage.

Performance metrics					
Stages	Precision	Recall	F1-score	Accuracy	Error rate
W	0.96	0.95	0.96	0.95	0.05
N1	0.81	0.70	0.75	0.70	0.3
N2	0.94	0.96	0.95	0.96	0.04
N3	0.97	0.97	0.97	0.97	0.03
REM	0.96	0.96	0.96	0.96	0.04
Overall accuracy 95.61%			Kappa statistic 0.94		

The misclassification rate was around 0.05. Lastly, the weakest performance was for the sleep stage N1 with accuracy 0.70, and the highest misclassification rate of 0.3. The precision, recall and F1-score are slightly poorer than for the above discussed stages. The overall accuracy, i.e. correctly classified number of epochs divided by the total number of epochs, was 95.61%. The kappa statistic at 0.94 indicates that the classifier was informative (much better than random guesses) with near perfect agreement to targeted labels.

4.2. EEG channel (FPz-Cz) from Sleep EDF dataset

Table 5 shows the confusion matrix obtained from 10-fold cross-validation on the single EEG channel (FPz-Cz) with AASM labeling. Again, we merged S3 and S4 labels given by an expert into the single N3 stage (slow wave stage). The bold numbers on the diagonal highlight the correctly classified epochs.

Table 5. Confusion matrix from cross-validation using the (FPz-Cz) electrode as input and AASM standard sleep stages.

Predicted						
	Stages	W	N1	N2	N3	REM
Actual	W	**7171**	39	461	92	157
	N1	77	**2457**	181	37	50
	N2	269	53	**17,013**	173	273
	N3	131	18	326	**5115**	110
	REM	136	23	441	88	**7009**

Table 6 gives the performance metrics by AASM sleep stage. In these data, N2 had the highest accuracy of calls, 0.95, with respective error rates 0.05. The stages W, N3, and REM were called with 0.89–0.91 accuracy and respective error rates 0.10–0.11. Finally, the poorest call accuracy was 0.87 for stage N1 having error rate 0.13. The overall accuracy was 92.51% and the kappa 0.90 indicates nearly perfect calls by our SleepTuneNet model.

4.3. Comparison with state-of-the-art approaches

Table 7 presents a comparison between our method and state-of-the-art approaches with or without human intervention in feature extraction, by overall accuracy and Cohen's kappa. It can be noticed that our method performed best among the approaches that utilized the single EEG channel, in terms of overall accuracy. The kappa value confirms good agreement of labelling by a sleep expert with our classification method (0.90 and

Table 6. Performance metrics by AASM sleep stage.

Performance Metrics					
Stages	Precision	Recall	F1-score	Accuracy	Error rate
W	0.92	0.90	0.91	0.90	0.1
N1	0.94	0.87	0.91	0.87	0.13
N2	0.92	0.95	0.93	0.95	0.05
N3	0.92	0.89	0.91	0.89	0.11
REM	0.92	0.91	0.91	0.91	0.09
Overall accuracy 92.51%		Kappa statistic 0.90			

0.94). Besides, our GACNN SleepTuneNet scored the highest accuracy not only among studies using the different dataset, but also overall.

Table 7. Comparison between GACNN SleepTuneNet and other methods that used hand-crafted features and without hand-crafted features by overall accuracy and Cohen's kappa.

Study	EEG Channel	No. of epochs	Scoring standard	Cross-validation	Metrics performance	
					Accuracy	Kappa
[19]	C3-A2	30,000	AASM	25	84	-
[20]	Fpz-Cz	15,000	AASM	20	71-76	-
[21]	Fpz-Cz	41,950	AASM	20	82.0	0.76
[9]	Fpz-Cz	37,022	AASM	20	75-80	-
[22]	FPz-Cz	960	R&K	10	87.2	-
[23]	F4-EOG left	59,066	AASM	31	85.92	-
GACNN SleepTuneNet	FPz-Cz	41,900	AASM	10	92.51	0.90
GACNN SleepTuneNet	FP2-F4	28,864	AASM	10	95.61	0.94

5. Discussion

In this work we presented an automatic sleep scoring method applied to two sleep datasets without hand-crafted features. It was demonstrated that parameter settings are essential for achieving excellent classification accuracy. To tune the hyperparameters, we generated 25 populations for each dataset with classification accuracy as the evolutionary goal. For example, we noticed that kernel size, number of filters, pooling size, and other hyperparameters affected classification accuracy. We also observed that the iteration count in backpropagation training affects the classification accuracy: for instance, too many iterations can give worse accuracy than earlier interrupted training. For designing the GACNN SleepTuneNet architecture we tested various numbers of filters, kernel sizes, pooling sizes, and stride rates, along with other hyperparameters (weight decay, learning rate, and iteration limit) for the classification of sleep stages.

Moreover, we also noticed that there is high variation in the accuracy between two datasets with different electrode positions, or possibly because of having healthy and unhealthy subjects. The selection of EEG electrodes is an essential factor in an automatic system for sleeping scoring. Nonetheless, based on the results our method not only performed well with two different EEG electrodes (FP2-F4 and FPz-Cz) but was able to distinguish the AASM standard sleep stages from each other very well (see Tables 3 and 5). For example, the calls of N2 and N3 stages achieved high performance in both datasets. Overall Table 7 demonstrates that our model had comparatively very good performance among other models with or without hand-crafted features. Our model was evaluated by 10-fold cross-validation. Other fold counts were tested also, ranging from 5- to 31-fold cross-validation. We noticed two problems with too many folds: not only computational expense, but also negative impact on the robustness of the produced model. The appropriate number of cross-validation folds depends on the problem at hand, especially on the total number of training instances available. The larger the dataset is, the fewer folds are needed to produce a robust model.

6. Conclusion

This study applied a genetic algorithm to diagnosing the structure of a convolutional neural network named GACNN SleepTuneNet, intended for automatic sleep scoring by AASM standards, based on a single EEG channel without using hand-crafted features. We used a genetic algorithm to evolve an appropriate CNN architecture for the classification of sleep stages over N = 25 generations, for each dataset. Our results show that the GACNN SleepTuneNet is not only useful without hand-crafted features, but this method's filters are interpretable in the context of AASM standards. Finally, GACNN SleepTuneNet can provide automatic sleep scoring.

Acknowledgments

This work was supported by the Higher Education Research Promotion and the Thailand's Education Hub for Southern Region of ASEAN Countries Project Office of the Higher Education Commission, Prince of Songkla University,Thailand.

References

[1] Cecotti H, Eckstein MP, Giesbrecht B. Single-trial classification of event-related potentials in rapid serial visual presentation tasks using supervised spatial filtering. IEEE Transactions on Neural Networks and Learning Systems 2014; 25(11): 2030-2042.

[2] Kiranyaz S, Ince T, Gabbouj M. Real-time patient-specific ECG classification by 1-D convolutional neural networks. IEEE Transactions on Biomedical Engineering. 2016; 63(3): 664-675.

[3] Zhu X, Zheng WL, Lu BL, Chen X, Chen S, Wang C. EOG-based drowsiness detection using convolutional neural networks. In: International Conference on Neural Network; Beijing, China; 2014. pp. 128-134.

[4] Hossain D, Capi G, Jindai M. Optimizing deep learning parameters using genetic algorithm for object recognition and robot grasping. Journal of Electronic Science and Technology 2018; 16(1): 11-15.

[5] David OE, Greental I. Genetic algorithms for evolving deep neural networks. In: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation; Vancouver, BC, Canada; 2014. pp. 1451-1452.

[6] Lajnef T, Chaibi S, Ruby P, Aguera PE, Eichenlaub JB et al. Learning machines and sleeping brains: automatic sleep stage classification using decision-tree multi-class support vector machines. Journal of Neuroscience Methods 2015; 30 (250): 94-105.

[7] Huang CS, Lin CL, Ko LW, Liu SY, Su TP et al. Knowledge-based identification of sleep stages based on two forehead electroencephalogram channels. Frontiers in Neuroscience 2014; 8: 263.

[8] Güneş S, Polat K, Yosunkaya Ş. Efficient sleep stage recognition system based on EEG signal using k-means clustering based feature weighting. Expert Systems with Applications. 2010; 37(12): 7922-7928.

[9] Tsinalis O, Matthews PM, Guo Y. Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders. Annals of Biomedical Engineering. 2016; 44(5): 1587-1597.

[10] Tsinalis O, Matthews PM, Guo Y. Automatic sleep stage scoring using time-frequency analysis and stacked sparse autoencoders. Annals of Biomedical Engineering. 2016; 44(5): 1587-1597.

[11] Hassan AR, Subasi A. A decision support system for automated identification of sleep stages from single-channel EEG signals. Knowledge-Based Systems 2017; 15(128):115-124.

[12] Qureshi S, Seppo K, Sirirut V. Human sleep scoring based on K-Nearest Neighbors. Turkish Journal of Electrical Engineering and Computer Sciences 2018; 26(6): 2802-2818.

[13] Boostani R, Karimzadeh F, Nami M. A comparative review on sleep stage classification methods in patients and healthy individuals. Computer Methods and Programs in Biomedicine 2017; 140: 77-91.

[14] Şen B, Peker M, Çavuşoğlu A, Çelebi FV. A comparative study on classification of sleep stage based on EEG signals using feature selection and classification algorithms. Journal of Medical Systems. 2014; 38(3): 18.

[15] Diykh M, Li Y. Complex networks approach for EEG signal sleep stages classification. Expert Systems with Applications. 2016; 63: 241-248.

[16] Hassan AR, Bhuiyan MI. Computer-aided sleep staging using complete ensemble empirical mode decomposition with adaptive noise and bootstrap aggregating. Biomedical Signal Processing and Control. 2016; 24: 1-10.

[17] Hassan AR, Bhuiyan MI. Automatic sleep scoring using statistical features in the EMD domain and ensemble methods. Biocybernetics and Biomedical Engineering. 2016; 36(1): 248-255.

[18] da Silveira TL, Kozakevicius AJ, Rodrigues CR. Single-channel EEG sleep stage classification based on a streamlined set of statistical features in wavelet domain. Medical Biological Engineering and Computing. 2017; 55(2): 343-352.

[19] Längkvist M, Karlsson L, Loutfi A. Sleep stage classification using unsupervised feature learning. Advances in Artificial Neural Systems. 2012; 1: 5.

[20] Tsinalis O, Matthews PM, Guo Y, Zafeiriou S. Automatic sleep stage scoring with single-channel EEG using convolutional neural networks. arXiv preprint arXiv:1610.01683. 2016.

[21] Supratak A, Dong H, Wu C, Guo Y. DeepSleepNet: a model for automatic sleep stage scoring based on raw single-channel EEG. IEEE Transactions on Neural Systems and Rehabilitation Engineering. 2017; 25(11): 1998-2008.

[22] Hsu YL, Yang YT, Wang JS, Hsu CY. Automatic sleep stage recurrent neural classifier using energy features of EEG signals. Neurocomputing. 2013; 15 (104): 105-114.

[23] Dong H, Supratak A, Pan W, Wu C, Matthews PM et al. Mixed neural network approach for temporal sleep stage classification. IEEE Transactions on Neural Systems and Rehabilitation Engineering 2018; 26(2): 324-33.

[24] Goldberger AL, Amaral LA, Glass L, Hausdorff JM, Ivanov PC et al. PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. Circulation 2000; 101(23): 215-220.

[25] Rechtschaffen A, Kales A. A Manual for Standardized Terminology, Techniques and Scoring System for Sleep Stages in Human Subjects. Brain Information Service. Los Angeles, CA, USA: National Institutes of Health, 1968.

[26] Iber C, Ancoli-Israel S, Chesson A, Quan SF. The AASM Manual for the Scoring of Sleep and Associated Events: Rules, Terminology and Technical Specifications. Westchester, IL, USA: American Academy of Sleep Medicine; 2007.

[27] Xie AYL. Genetic CNN. In: 2017 IEEE International Conference on Computer Vision (ICCV); Venice, Italy; 2018. pp. 1388-1397.

[28] Young SR, Rose DC, Karnowski TP, Lim SH, Patton RM. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In: Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments; Austin, USA; 2015. pp. 4.

[29] Grefenstette J, Gopal R, Rosmaita B, Van Gucht D. Genetic algorithms for the traveling salesman problem. In: Proceedings of the First International Conference on Genetic Algorithms and Their Applications; Pittsburgh, USA; 1985. pp. 160-168.

[30] Loshchilov I, Hutter F. CMA-ES for hyperparameter optimization of deep neural networks. arXiv preprint arXiv:1604.07269. 2016.

[31] Goldberg DE, Holland JH. Genetic algorithms and machine learning. Machine Learning 1988; 3(2): 95-99.

[32] Faust O, Hagiwara Y, Hong TJ, Lih OS, Acharya UR. Deep learning for healthcare applications based on physiological signals: a review. Computer Methods and Programs in Biomedicine 2018; 161: 1-3.

[33] Goodfellow I, Bengio Y, Courville A, Bengio Y. Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[34] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 2014; 15(1): 1929-58.

[35] Van Doorn J. Analysis of deep convolutional neural network architectures. arXiv preprint 2014: 9-12.

[36] Yulita IN, Fanany MI, Arymurthy AM. Fast convolutional method for automatic sleep stage classification. Healthcare Informatics Research 2018; 24(3): 170-178.

[37] Landis JR, Koch GG. The measurement of observer agreement for categorical data. Biometrics 1977; 33(1): 159-174.