Fast Automatic Artifact Annotator for EEG Signals Using Deep Learning

D. Kim and S. Keene

Department of Electrical Engineering, The Cooper Union, New York, New York, USA
{kim11, keene}@cooper.edu

Abstract— Electroencephalogram (EEG) is a widely used non-invasive brain signal acquisition technique that measures voltage fluctuations from neuron activities of the brain. EEGs are typically used to diagnose and monitor disorders such as epilepsy, sleep disorders, and brain death and also to help the advancement of various fields of science such as cognitive science, and psychology. EEG signals usually suffer from a variety of artifacts caused by eye movements, chewing, muscle movements, and electrode pops, which disrupts the diagnosis and hinders precise representation of brain activities. This paper proposes a deep learning based model to detect the presence of the artifacts and to classify the kind of the artifact to help clinicians resolve problems regarding artifacts immediately during the signal collection process. The model is optimized to map the 1-second segments of raw EEG signals to detect 4 different kinds of artifacts and the real signal. The model achieves a 5-class classification accuracy of 67.59%, and a true positive rate of 80% with a 25.82% false alarm for binary artifact classification with time-lapse. The model is lightweight and could potentially be deployed in portable machines.

I. Introduction

The study of the brain, neuroscience, to understand humans better has been a great research area that combines scientists and engineers across various disciplines. Much advancement in neuroscience has come from analyzing accurate recordings of the brain. An electroencephalogram (EEG) is a popular non-invasive method for acquiring brain signals. Unfortunately, EEG signals suffer from artifacts that are both physiological and technical, and the artifacts are usually not documented well [1]. Artifacts decrease the signal to noise ratios of signals and disrupt the accurate collection of brain data. A system that detects the presence, and the character of artifacts during the collection of the EEG signals may lead to discoveries on the human brain faster by reducing unnecessary time spent sorting through artifacts.

We present a system that can quickly identify the presence of artifacts and the type if present, during the EEG wave collection. The purpose is so that a clinician can resolve the problem immediately to ensure that the collected data are artifact-free.

We utilize an ensemble system that contains multiple optimized deep learning architectures with a sliding window technique that uses multiple time segments to enhance the accuracy. The system aims to be memory efficient, and computationally light while being fast enough to be both implemented on portable systems, and detect and classify artifacts in real-time, potentially in a clinically setting.

The paper is organized as follows. In Section II, related works on automatic artifact detectors for EEG signals are presented. In Section III, we describe the data, the model, and the experiment. In Section IV, our experimental results are shown and analyzed. Finally, in Section V, conclusion and future works are presented.

II. Related Work

To achieve this goal, Temple University's TUH EEG Corpus (TUEEG) has constructed a large data set of EEG waves from various patients, specifically labeled for artifacts [2]. This data provide engineers and scientists some basis to test their ideas and help advance science and technology.

Golmohammadi et al. utilizes hidden Markov models, deep learning models, and statistical language models to build a model that achieves a true positive rate of 90% and a false alarm rate of below 5% on events of clinical interests, namely spikes, generalized and periodic epileptiform discharges [3]. This work proves the viability of big data and deep learning methods in detecting events in EEG signals. However, this model is only able to distinguish 14.04% of the artifacts correctly from the data, as the study in [3] was not towards detecting artifacts.

Other works for detecting EEG artifacts include FASTER by Nolan [4], which uses independent component analysis (ICA), and Morphological Component Analysis (MCA) by Singh [5]. These signal processing techniques work by separating multivariate signals into subcomponents. Although they tend to have higher accuracies, they are computationally very intensive. Nolan et al. achieves more than 90% sensitivity on data with more than 64 channels, but the sensitivity drops to 5.88% when the number of channels decreases to 32 [4]. This algorithm takes an hour per dataset of around 400 seconds to yield the results using a machine with a 64-bit dual-core machine. Singh et al. only shows the availability of MCA for the signal analysis but mentions that 1024 samples of data that are sampled at 173.61Hz take about 6 seconds, which is around 1.01s computation time per 1 second of a signal [5]. These characteristics are not suitable for a fast EEG detector.

Table 1. Occurrences and Relative Frequencies of Each Label

Label	Occurrences	Percentage (%)
eyem	7471	26.20
chew	2727	9.56
elpp	2663	9.34
musc	4892	17.16
null	10763	37.74
total	**28516**	**100.00**

Table 2. EEG Wave Labels

Label	Description
eyem	Eye movements
chew	Chewing
shiv	Shivering
elpp	Electrode pop
musc	Muscle artifacts
null	No artifact

For the past few years, several attempts have been made to use deep learning framework for EEG signals [6]. Roy et al. indicates 40% of the studies regarding EEG signals using deep learning from 2010 to 2018 use convolutional neural networks and 14% use recurrent neural networks [6]. However, all the studies mentioned in [6] only use deep learning for the detection and classification of clinical events, and use either statistical methods for artifact detection and removal or raw data without artifact handling. Inspired by the success of deep learning for EEG signals, in this paper we investigate using deep learning for artifact detection.

III. Experimental Design

The data used in this study are from the Temple University Hospital's EEG Artifact Corpus. The version of the dataset used is v1.0.0, which is derived from the v1.1.0 of the TUH EEG Corpus [2]. There are 310 observations with 213 patients, and durations and sampling rates differ from observation to observation.

The model is developed in python, and experiments are done using a machine equipped with 16 GB memory, AMD FX(tm)-6300 Six-Core Processor 3.5GHz, and a GeForce GTX 1070 8GB graphics card.

The dataset contains 3 different configurations of EEG: AR (averaged reference), LE (linked ears reference), and AR_A configuration that is a modified version of the AR configuration. All the data contain standard measurements of channel information from the 10-20 International System. For AR, and LE configurations, 22 montages can be derived, and for AR_A configuration only 20 montages can be derived. In addition, only 4 patients are available for AR_A configuration. Since there are too few observations to capture unique features of AR_A compared to the other configurations, AR_A configuration was discarded for all the experiments in this paper. Hence, only 303 observations with 209 patients are used for the experiments.

In order to format all the data to have the same amount of information, the data, which have varying sampling frequencies of 250Hz, 256 Hz, 480Hz, and 500Hz, have been resampled to 250Hz. Also, as each observation has a different duration, all the signals are divided into 1-second segments. The original observations are in 16-bit floating points, but as the model utilizes 32-bit precision floating points, all the observations are converted to 32-bit floating points. The resulting 1-second segment, which will be fed into the model is a 22×250 tensor.

Before any more processing is done, the dataset has been divided into the train set, the validation set, and the test set. The ratio among the three sets is chosen to be 0.75:0.10:0.15 to allow a high probability of the test and the validation sets containing at least one example of each label while leaving enough examples for the training set. The set division is performed on the unique patient ID to ensure that the training and the testing are not performed on the same patient as the goal of this model is to detect artifacts on new patients. The order of the patient IDs has been shuffled before dividing IDs into the 3 sets. This split corresponds to 157 patients in the training set, 21 patients in the validation set, and 31 patients in the test set.

Each 1-second segment belongs to one of the 6 possible labels. The label names and the corresponding descriptions are shown in Table 2. Details of the labels and the dataset are in [2].

There are 6 possible labels in the dataset, but one of the labels, "shiv", is represented by 0.39% of the data, this label is disregarded for the experiment as there is not enough data for the models to train. In addition, there is a high imbalance of data due to a large number of "null". This is because the artifact content in the clinical EEG waves is low. To balance the number of observations for each label more even, "null" is subsampled such that only the 30^{th} observation is kept for the experiment. Break down of occurrences of each label and the relative frequency after the removal of one label, and subsampling is shown in Table 1.

After the training set is formed, all the signals are normalized by subtracting the mean and dividing by the standard deviation of the training set. The mean and the standard deviation used for the normalizing are -1.598, and 219.395, respectively.

Table 3. Network Architecture for RNN

Layer (type)	Output Shape	Param #
Input_1 (InputLayer)	(None, 22, 250)	0
Lstm_1 (LSTM)	(None, 50)	60200
Dense_1 (Dense)	(None, 1024)	52224
Dense_2 (Dense)	(None, 5)	5125

Table 4: Network Architecture for CNN

Layer (type)	Output Shape	Param #
Input_1 (InputLayer)	(None, 22, 250)	0
conv1d_1 (Conv1D)	(None, 16, 250)	1072
batch_normalization_1	(None, 16, 250)	1000
max_pooling1d_1	(None, 16, 125)	0
conv1d_2 (Conv1D)	(None, 32, 125)	1568
batch_normalization_2	(None, 32, 125)	500
max_pooling1d_2	(None, 32, 63)	0
conv1d_3 (Conv1D)	(None, 64, 63)	6208
batch_normalization_3	(None, 64, 63)	252
max_pooling1d_3	(None, 64, 32)	0
conv1d_4 (Conv1D)	(None, 128, 32)	24704
batch_normalization_4	(None, 128, 32)	128
max_pooling1d_4	(None, 128, 16)	0
conv1d_5 (Conv1D)	(None, 256, 16)	98560
batch_normalization_5	(None, 256, 16)	64
max_pooling1d_5	(None, 256, 8)	0
conv1d_6 (Conv1D)	(None, 512, 8)	393728
batch_normalization_6	(None, 512, 8)	32
flatten_1	(None, 4096)	0
dense_1(Dense)	(None, 1024)	4195328
dense_2(Dense)	(None, 5)	5125

Our model is an ensemble model that includes a Recurrent Neural Network (RNN), and two Convolutional Neural Networks (CNNs). Network architectures for all three networks (RNN, CNN, DCNN) are shown in Table 3, Table 4, and Table 5.

All the networks in the ensemble model are trained with Adam optimizer [7] with the default setting. Adam optimizer is a method for gradient-based optimization which is frequently used in deep learning due to its computational efficiency. Each network is trained separately by minimizing the categorical cross-entropy for 5-class classification, or the categorical entropy for binary classification, which is a measure of the error in making a decision of which category the sample belongs to in 5 categories or 2 categories respectively. RNN and CNN are trained to the 100^{th} epochs, and DCNN is trained to 30^{th} epochs. The parameters for the layers in all the networks are chosen through experiments using the validation set. The ensemble model adds all the logits produced at the end of each networks to yield a set of final logits. The label with the highest logit is chosen to be the predicted label.

All the networks have two versions: the first version is for the 5-class classification, and the second version is for the binary classification in which the last layer for all the networks is replaced with a similar layer that has an output shape of (None, 2). As the number of parameters differs for the two versions, all the networks are trained twice with the same settings.

Table 5. Network Architecture for DCNN

Layer (type)	Output Shape	Param #
Input_1 (InputLayer)	(None, 22, 250)	0
conv1d_1 (Conv1D)	(None, 16, 250)	1072
batch_normalization_1	(None, 16, 250)	1000
max_pooling1d_1	(None, 16, 125)	0
conv1d_2 (Conv1D)	(None, 32, 125)	1568
batch_normalization_2	(None, 32, 125)	500
max_pooling1d_2	(None, 32, 63)	0
conv1d_3 (Conv1D)	(None, 64, 63)	6208
batch_normalization_3	(None, 64, 63)	252
max_pooling1d_3	(None, 64, 32)	0
conv1d_4 (Conv1D)	(None, 128, 32)	24704
batch_normalization_4	(None, 128, 32)	128
max_pooling1d_4	(None, 128, 16)	0
conv1d_5 (Conv1D)	(None, 256, 16)	98560
batch_normalization_5	(None, 256, 16)	64
max_pooling1d_5	(None, 256, 8)	0
conv1d_6 (Conv1D)	(None, 512, 8)	393728
batch_normalization_6	(None, 512, 8)	32
max_pooling1d_6	(None, 512, 4)	0
conv1d_7 (Conv1D)	(None, 1024, 4)	1573888
batch_normalization_7	(None, 1024, 4)	16
max_pooling1d_7	(None, 1024, 2)	0
conv1d_8 (Conv1D)	(None, 1024, 2)	3146752
batch_normalization_8	(None, 1024, 2)	8
conv1d_9 (Conv1D)	(None, 1024, 2)	3146752
batch_normalization_9	(None, 1024, 2)	8
flatten_1	(None, 2048)	0
dense_1(Dense)	(None, 1024)	2098176
dense_2(Dense)	(None, 1024)	1049600
dense_3(Dense)	(None, 5)	5125

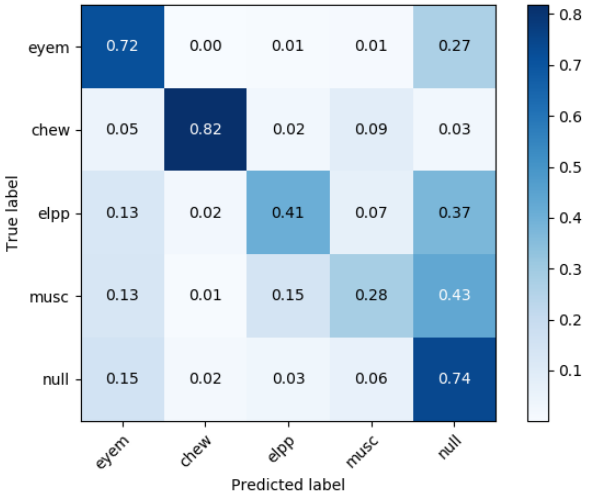

Figure 1. Confusion Matrix

IV. Results and Discussion

The confusion matrix for the model that combines three different networks is shown in Figure 1. This model achieves an overall accuracy of 0.6759. The model performs better on "eyem" and "chew" artifacts and "null". Confusion matrices of individual networks in the ensemble model are similar to that of the ensemble model, but with inferior performance.

In order to evaluate the model's viability as an artifact detector, the binary classification version of the model is tested. The receiver operating characteristic (ROC) curves for the ensemble method and all the networks are shown in Figure 2. Areas under the curve are computed for numerical comparisons.

For the binary classification problem, the main purpose is to accurately detect the artifact events, regardless of their type. A sliding window is utilized to further enhance the performance. The idea is similar to the rationale behind using hidden Markov models as in [3]. Artifacts often come in bursts, the previous segment's label correlates well with the new segment that follows. Hence, we added a sliding window along the time axis for the logits produced. For example, for a sliding window of size 2, the model would add the logits of the first two segments to predict the label of the first segment. After some experimentation, we found that simply adding the logits produced the best results. A sliding window of size 2, which corresponds to using 2 seconds to determine the presence of artifacts, was chosen empirically by comparing accuracies of varying window sizes on the validation set. The new ROC curves for the highest performing window setting are shown in Figure 3.

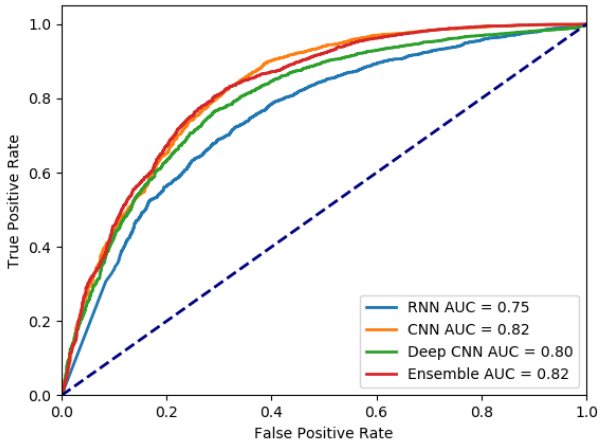

Figure 2. ROC Curves for All Networks

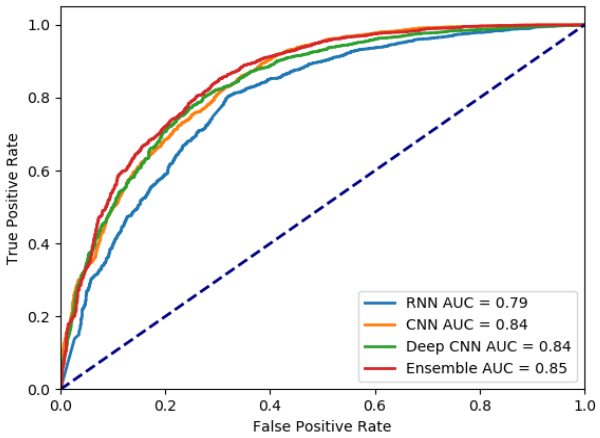

Figure 3. ROC Curves with Time-Lapse

The areas under the curve of the new ROC curves are improved by 0.03, which indicates the sliding window system helps in making a more accurate decision. In addition, this system has a true positive rate of 0.8000 with a false alarm rate of 0.2582 for the binary classification.

The inference time for a 1-second segment is 1.785ms. The times are measured by computing results for 1000 samples 10 times and taking the average. The size of the model is 64191 KB.

V. Conclusion and Future Work

In this paper, we have developed a deep learning based machine learning model that learns to distinguish artifacts from the real signal and classify artifacts for EEG signals. The model achieves a 67.6% 5-class classification accuracy, and a true positive rate of 80% at the false positive rate of 25.8% for the binary classification.

The model is light and fast to be implemented in a portable device such as Raspberry Pi. Evidently, the model contains only 65MB of parameters, and 2ms to perform prediction on a 1-second segment of the signal.

Our model achieves the state of the art performance on detecting and classifying artifacts on the 22 channel EEG data with a much shorter amount of the computation time compared to [3], [4], and [5].

Acknowledgments

We would like to thank the Department of Electrical Engineering at The Cooper Union, and Temple University Hospital for supporting this research by providing us the necessary resources.

REFERENCES

[1] E. K. S. Louis, L. C. Frey, J. W. Britton, J. L. Hopp, P. Korb, M. Z. Koubeissi, W. E. Lievens and E. M. Pestana-Knight, *Electroencephalography (EEG): An Introductory Text and Atlas of Normal and Abnormal Findings in Adults, Children, and Infants*, 2016.

[2] I. Obeid and J. Picone, "The Temple University Hospital EEG Data Corpus," *Front. Neurosci. Sect. Neural Technol.*, vol. 10, p. 196, 2016.

[3] M. Golmohammadi, A. H. H. N. Torbati, S. L. d. Diego, I. Obeid and J. Picone, "Automatic Analysis of EEGs Using Big Data and Hybrid Deep Learning Architectures," *Frontiers in Human Neuroscience,* vol. 13, 2019.

[4] H. Nolan, R. Whelan and R. B. Reilly, "FASTER: Fully Automated Statistical Thresholding for EEG artifact Rejection," *Journal of Neuroscience Methods,* vol. 192, no. 1, pp. 152-162, 2010.

[5] B. Singh and H. Wagatsuma, "A Removal of Eye Movement and Blink Artifacts from EEG Data Using Morphological Component Analysis," *Computational and Mathematical Methods in Medicine,* vol. 2017, pp. 1861645-1861645, 2017.

[6] Y. Roy, H. J. Banville, I. Albuquerque, A. Gramfort, T. H. Falk and J. Faubert, "Deep learning-based electroencephalography analysis: a systematic review.," *Journal of Neural Engineering,* vol. 16, no. 5, p. 51001, 2019.

[7] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations,* 2015.