Chapter 7
Fast Automatic Artifact Annotator for EEG Signals Using Deep Learning

Dong Kyu Kim and Sam Keene

7.1 Introduction

The study of the brain, neuroscience, to understand about ourselves better has been a great research area that combines the efforts of scientists and engineers across various disciplines. Due to the brain's complexity, the understanding of the basis of learning, perception, and consciousness is sometimes described as the "ultimate challenge" of biological sciences (Aminoff 2001). Currently, many advances in neuroscience come from analyzing recordings of the brain. However, due to the overwhelming amount of electrochemical activities in the brain, the collection of reliable data is still one of the biggest challenges in neuroscience (Louis et al. 2016).

There are two main branches of brain signal acquisition methods: invasive and non-invasive methods. Invasive methods involve placements of electrodes inside the brain or insertion of needles through the subject's head to collect precise and highly local data. On the other hand, non-invasive methods such as electroencephalogram (EEG) and magnetic resonance imaging (MRI) suffer from noise and various artifacts (Louis et al. 2016). Due to the high interest and potential in this area of research, in addition to relatively cheap and accessible EEG recording machines (DellaBadia et al. 2002), there are a lot of interesting data available for analysis. However, a lot of EEG data suffer from artifacts which are unwanted signals present in the recordings as a result of the procedure of measurements. Artifacts in EEGs are both physiological and technical, and they require well-trained observers to be identified well (Louis et al. 2016). If there is a system that can distinguish

D. K. Kim (✉) · S. Keene
Electrical Engineering Department, The Cooper Union, New York, NY, USA
e-mail: dongkyuk@usc.edu

I. Obeid et al. (eds.), *Biomedical Signal Processing*,
https://doi.org/10.1007/978-3-030-67494-6_7

between artifacts, and cerebral data automatically, neuroscience can advance further as reducing the effect of artifacts will increase the signal-to-noise ratio so that brain activity can be detected more precisely.

To achieve this goal, Temple University has constructed a large dataset of EEG signals from various subjects that are specifically labeled for artifacts (Obeid and Picone 2016) to aid engineers and scientists to build models that detect and remove the artifacts. Previously, Golmohammadi and colleagues developed a model that automatically analyzes EEG signals to help physicians diagnose brain-related disorders such as seizures using hybrid deep learning architectures. This model integrates hidden Markov models, deep learning models, and statistical language models to deliver a composite model that has a true positive rate of 90% while having a false alarm rate of below 5% on events of clinical interests: spike and sharp waves, periodic lateralized epileptiform discharges, and generalized periodic epileptiform discharges (Golmohammadi et al. 2019). This model proves the viability of big data and deep learning methods in detecting events in EEG signals.

The work in Golmohammadi et al. (2019) attempts to classify artifacts as well as the mentioned events of clinical interest, but the model developed was only able to distinguish 14.04% of the artifacts correctly from the data. As the goal of that model was to detect seizures and epilepsy, no further analysis of artifacts was done, but it was noted that transient pulse-like artifacts such as eye movements and muscle movements can significantly degrade the performance. In this chapter, a method that can quickly identify the presence of artifacts and the type of the artifacts during the data acquisition is proposed so that a clinician can resolve the problem immediately and ensure the collected data is cleaner. To achieve this goal, multiple deep learning models with varying model size, inference time, and accuracies were developed and optimized to compare and contrast between advantages and disadvantages of different approaches. The key feature of the models is that all the inferences are done directly on the signals with a minimal preprocessing such as normalizing and aggregating enough samples to be used for predictions by the model. The system aims to be memory efficient, and computationally light, while being fast enough to be implemented on portable systems such as Raspberry Pi. Such portable systems would be able to detect and classify artifacts in real-time, potentially in a clinical setting.

7.2 Related Works

There have been numerous efforts to combat the artifact problems in EEG signals. A lot of research has been done to reduce the effects of artifacts by utilizing prior knowledge such as how some artifacts behave in the signal. Artifact removal and detection tools of this nature tend to examine the statistical characteristics of the signals.

Nolan, Whelan, and Reilly (Nolan et al. 2010) proposed FASTER, Fully Automated Statistical Thresholding for EEG artifact Rejection, which uses independent component analysis (ICA) to separate EEG signals into neural activity and artifacts. ICA works by separating multivariate signals into additive subcomponents by assuming that different subcomponents are statistically independent of each other. The advantage of using ICA is that ICA reduces the statistical dependencies of different components of the signal, by separating the components (Lee et al. 1999). After the separation, the model uses a statistical comparison charts to check for features such as correlation with signal components, mean, variance, spatial, etc. This model was tested on simulated EEGs and real EEGs and had a true positive rate of over 90% in detecting artifacts when the model was given data with more than 64 channels. However, the true positive rate drops to 5.88% when the number of channels provided decreases to 32. Besides, the algorithm takes an hour per 400 s to yield the results using a machine with a 64-bit dual-core machine. Nevertheless, the model not only detects the signal quite accurately but also can remove the artifact, as any separated component of the signal can be extracted. This model can detect eye movements, EMG artifacts, linear trends, and white noise.

Similarly, Singh and Wagatsuma (2017) used Morphological Component Analysis (MCA), which uses a dictionary of multiple bases to guarantee the reconstruction of original signals. MCA is applied to the EEG signal so that the signal is deconstructed into a combination of bases in the dictionary. Singh and Wagatsuma hypothesized that three dictionaries of bases are dominant, and they are undecimated wavelet transform (UDWT), discrete sine transform (DST), and DIRAC (standard unit vector basis). The decomposition was able to show that EEG signals and their artifacts are represented by different dictionaries of bases, indicating that given the decomposition result, artifacts can be distinguished from the signals of interest. Singh and Wagatsuma successfully categorized which dictionary corresponds well with the signal or the artifact. This research demonstrates that an ensemble of different signal processing techniques could work well for artifact classification. The drawback of this method is similar to that of Nolan's. MCA takes about 6 s on 1024 samples of data that are sampled at 173.61 Hz. This corresponds to spending around 1.01 s of computation time per 1 s of a signal. As a result, this computational time is not suitable for fast EEG artifact detection. There are numerous other additional statistical approaches to separate the real EEG signal from the artifacts, such as canonical correlation analysis, which Clercq used to remove muscle artifacts from the EEG signals (Clercq et al. 2006).

All of the statistical approaches of the problem require a deconstruction of EEG signals into multiple components and analyzing each component to determine which components are responsible for artifacts and which are responsible for the real signal. Though they are highly interpretive, the separation procedure takes a lot of computation, and correct prior knowledge, such as the number of artifacts, a set of orthogonal bases that work well with the time-series data, or the general behavior of artifacts, is required. Due to the complex nature of the EEG signals, deep learning with its ability to learn hidden features from the raw data has shown great promises (Goodfellow et al. 2016).

According to the review paper by Roy et al. (2019), among the 156 papers about applying deep learning to EEG signals that the authors reviewed from January 2010 to July 2018, some papers applied data preprocessing techniques and artifact rejection techniques such as the ICA mentioned above to combat the artifacts, while some papers just used the raw EEG signals. Given that the majority of the papers did not use any artifact removal schemes, Roy et al. suggest that using deep learning on EEG signals directly might avoid the artifact removal step without any performance degradation. However, all the papers mentioned in Roy's review paper specifically target certain applications such as detecting epilepsy, monitoring sleep, and making a brain-computer interface, and none of the papers mentioned targets the detection of artifacts specifically. The review paper suggests that convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are the most used networks in this field; 40% of the studies use CNNS and 14% use RNNs.

Other works relating to deep learning and EEG signals or EEG like signals not mentioned in the review paper above include Krishnaveni's work on ocular artifact removal in EEG signals (Krishnaveni et al. 2007) and Hasasneh's work on automatic classification of ocular and cardiac artifacts in magnetoencephalography (MEG) (Hasasneh et al. 2018). Both of these works include some data preprocessing. Hasasneh's work utilizes ICA, and Krishnaveni's work utilizes the Joint Approximation Diagonalisation of Eigen-matrices (JADE) algorithm to separate the real signals from the artifact signals before using neural networks. The detection rates for the test data for both of these works are 94.4% and 92%, respectively. However, both of them only address one or two types of artifacts at the same time, while the model proposed will include four different artifacts to be classified separately with no preprocessing such that the model can be applied directly to the raw data.

There have been many attempts and there have been successful attempts in detecting artifacts and classifying them using statistical machine learning and inferences, but there are not much done using deep learning. Deep learning approaches are particularly adept at optimizing an arbitrary large model and recognizing complex patterns (Goodfellow et al. 2016). The previous methods require mathematical models for artifact events or seizure events to classify the signals accurately; hence the performance of the models depends highly on the accuracy of the proposed mathematical models. However, the usage of deep learning models can alleviate the incorrect modeling error as no accurate mathematical model is needed to classify different events. In addition, the statistical analysis of large temporal data is computationally heavy and takes a long time. While training a deep learning model to optimize the parameters may take a long time, the inference time for the completed model is relatively short compared to that of statistical models. To use these advantages, many works have attempted to classify different aspects of the EEG signals for monitoring purposes for seizure and sleep disorders using deep learning. However, not a lot of works have been done in detecting and classifying artifacts using deep learning, especially classifying multiple artifacts instead of detecting a small number of artifacts.

7.3 Method

7.3.1 Resources

The dataset used to develop our model is the Temple University Hospital's EEG Artifact Corpus. This dataset was developed to help researchers build models to reduce the harmful effects of artifacts for EEG event classification algorithms such as seizure detection algorithms. The version of the dataset is v1.0.0, and the dataset is derived from the v1.1.0 of the TUH EEG Corpus (Obeid and Picone 2016). The TUH EEG Corpus is the largest publicly available database of clinical EEG data that contains over 30,000 EEGs spanning from 2002 to present, and the Artifact Corpus is a subset of the original corpus that has been specifically labeled to enhance artifact related research. There are 310 observations with 213 subjects with varying durations and sampling rates.

The experiments to build our model were done using Python. Specifically, the version of the python that was used is 3.6.8. Additional libraries used are matplotlib v3.0.2, numpy v1.16.0, tqdm v4.31.1, scipy v1.2.0, tensorflow v1.12.0, and keras v2.2.4. matplotlib and tqdm library were used for making plots and monitoring progress, and numpy, scipy, tensorflow, keras libraries were used to build a deep learning model and test. All the experiments were done using a machine equipped with 16GB memory, AMD FX(tm)-6300 Six-Core Processor 3.5GHz, and a Geforce GTX 1070 8GB graphics card. The data drive in which the corpus was in was a standard hard drive with 7200RPM. Finally, the environment was a Windows 10 operating system with a virtual environment with all the above libraries created using conda for the Anaconda Python distribution.

7.3.2 Data Preprocessing

The data corpus contains three different configurations of EEG. The first is the AR (averaged reference) where the average of a finite number of electrodes is used as a reference. This means that the average is subtracted from the signals of each electrode for every time point to account for the common noise. The second configuration is the LE (linked ears reference) which is based on the assumption that ears do not have any electrical activity so that ears can be used as reference points (Lopez et al. 2016). The third configuration is the AR_A which is a modified version of the AR configuration, where A1_REF and A2_REF are not used. All the data contain standard measurements that one could expect from the 10–20 International System. For the AR, and the LE configurations, 22 channels can be derived from the available channel information, while for the AR_A configuration, only 20 channels can be derived. This is because the AR_A configuration lacks the EEG A1_REF and the EEG A2_REF channels. The computations necessary to derive the channels

Table 7.1 List of channels with appropriate computation

Channel number	Computation
1	FP1-F7
2	F7-T3
3	T3-T5
4	T5-01
5	FP2-F8
6	F8-T4
7	T4-T6
8	T6-02
9	A1-T3
10	T3-C3
11	C3-CZ
12	CZ-C4
13	C4-T4
14	T4-A2
15	FP1-F3
16	F3-C3
17	C3-P3
18	P3-01
19	FP2-F4
20	F4-C4
21	C4-P4
22	P4-02

are tabulated in Table 7.1. The AR_A configuration lacks channel number 9 and 14 from Table 7.1.

There are only seven occurrences of the AR_A configuration with four subjects, and as this configuration lacks similarity to other configurations, this configuration was discarded for the experiments. Too few examples of different data would hinder the model from learning the important aspects of the artifacts, and for deep learning models, consistent data size is important. The tradeoff is either to give up 2 channels across all 303 observations or to give up 7 observations, and we have decided to give up these 7 observations. Hence, for the experiment, there are 303 observations with 209 subjects available.

Another way to alleviate this problem is to fill in the missing channels. Nolan's work describes a method to fill in any missing channels using adjacent channels (Nolan et al. 2010). However, since the missing A1 and A2 electrodes in the AR_A configuration are reference points that are placed on ears, they cannot be interpolated from other electrodes, as all the other electrodes are on the head. We decided that guessing the signals on ears based on signals from the brain would not be accurate at all. As a result, we decided not to use the interpolation method and thus discard this configuration.

The original data are in the European Data Format (EDF), which is a standard file format designed for the storage of medical time series data. All the EDF

Table 7.2 Possible labels and corresponding descriptions

Label	Description
eyem	Eye movements
chew	Chewing
shiv	Shivering
elpp	Electrode related artifacts such as electrode pops, static electrodes, lead artifacts
musc	Muscle related artifacts
bckg	Background noise
null	Undefined annotation

files provided have all the electrode information so that channels defined in the instruction can be derived easily using the computations tabulated in Table 7.1. In addition, the corresponding label files contain the artifact class labels for the whole EEG session and also for each channel.

There are seven possible labels and the labels and the corresponding descriptions are tabulated in Table 7.2. The label files provide the start time and the stop time of the existing artifacts in seconds. The files have the confidence level of the label, which indicates the probability that the artifact is what the label says it is. All the labels in this data corpus have the confidence levels of 1, and the background noise label, "bckg," is not available for this dataset. This is because the corpus is still in the beginning stage of the development so it does not have a lot of data available so the "bckg" label seems to be lacking in this version. As a result, the model is developed to classify five artifacts and a "null" label. The "null" label is defined to be any undefined annotation, in this corpus; this label is given to signals that do not seem to have artifacts.

The EEG signals in the dataset have varying sampling frequencies of 250 Hz, 256 Hz, 480 Hz, and 500 Hz. As deep learning models require input features to be consistent that is input features need to be of the same size and having different sampling rates for temporal data can harm the performance of the model. For example, if we were to optimize the model to infer using 500 time points, this is equivalent to using 2 s if the sampling frequency is 250 Hz, and 1 s if the sampling frequency is 500 Hz. Then the two samples have different kinds of information available, as the former sample will have more seconds of information, while the latter sample will have more detailed information on a smaller time window.

To alleviate this problem, all the signals were resampled to 250 Hz, which is the lowest sampling rate using a Fourier method. Then the signals were separated into 1-second segments without overlaps. The separation is done so that the input signal to the model is kept small. The deep learning model size depends on the number of layer parameters, which depends on the complexity of the layer and the input size. Also, the separation allows the model to be able to infer on any instance, which means we can determine whether the segment of the signal is affected by artifacts at any time using the small accumulated data around the specific time. The 1-second segment was chosen as the lowest frequency of brain waves is around

Table 7.3 Occurrences and the percentage of original 1-second segment samples of each label

Label	Occurrences	Percentage (%)
eyem	7471	2.16
chew	2727	0.79
shiv	1338	0.39
elpp	2663	0.77
musc	4892	1.41
null	327,222	94.49
Total:	346,313	100

3 Hz, which allows the segment to have at least three occurrences of the smallest wave. In addition, all the observations end at a whole second, so that there is no loss of information when the time window for segments is 1 s.

After the resampling and the separation, 303 observations of varying lengths turn into 346,313 1-second segments. The breakdown of the number and the corresponding percentage of samples available for each label are tabulated in Table 7.3. There is a high imbalance of data due to many examples with the label "null." This is due to the nature of the signal as the artifact content in the clinical EEG waves collected should be ideally low. There are only 1338 observations of "shiv," which consists of 0.39% of all the data available. Due to the relatively small number of occurrences, this label caused problems in developing models. When a preliminary study was done to investigate possible research directions, the "shiv" label caused problems by not being able to separate into three sets required for the development of the models. Since there are too few samples of "shiv" available, and most of the samples are from the same subject, when the dataset is separated into the train, the test, and the validation sets, depending on the random state of the machine, samples with "shiv" label are only found in one or two of the three sets. To illustrate this problem, a recurrent neural network model was trained for 100 epochs on the dataset with the "shiv" label. The confusion matrix for this model is shown in Fig. 7.1. The model completely fails to classify the "shiv" label and predicts all the "shiv" events to be either a "musc" event, an "elpp" event, or a "null" event. In fact, the model does not predict anything to be the "shiv" event. The reason why the model failed to do so was because there was no "shiv" label available in the training set, which caused the model to never be exposed to the label. As a result, we decided to leave out the "shiv" label from the experiments. The updated numbers and the updated percentages of samples available for each label are tabulated in Table 7.4.

The data were divided into a train set, a validation set, and a test set. The ratio among the three was 0.75:0.10:0.15. The ratio was determined arbitrarily while making sure a good amount of data was available for each of the sets. The data division was done on the unique patient ID that was provided in the EEG corpus. The reason why the division was done on the IDs rather than the sessions is that we wanted to ensure that the training and the testing were not performed on the same patient as the goal of the models is to generalize to detect artifacts on new subjects. Out of the 209 subjects, 157 subjects were allocated to the training set, 21 subjects were allocated to the validation set, and 31 patients were allocated to the test set.

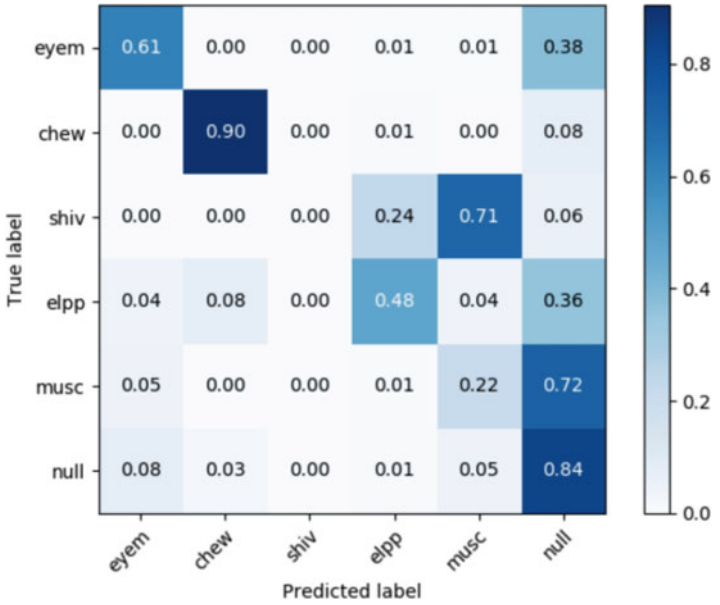

Fig. 7.1 Confusion matrix of the RNN-based model with all the labels

Table 7.4 Occurrences and the percentage of 1-second segment samples of each label after the removal of "shiv"

Label	Occurrences	Percentage (%)
eyem	7471	2.17
chew	2727	0.79
elpp	2663	0.77
musc	4892	1.42
null	327,222	94.85
Total	344,975	100

This translates to 224 sessions in the train set, 23 sessions in the validation set, and 56 sessions in the test set. The order of the patient ID has been shuffled before the division to remove any lingering pattern.

In addition to the sampling rate change, the signals are normalized. As the neural network models generally perform better when the data are in the $(-1,1)$ range, the dynamic range of the EEG signals is modified. All the signals were normalized to have a 0 mean, and a standard deviation of 1. The statistics of the whole training set were used for the normalization, and these statistics are used for all the sets as statistics of the unseen data are assumed to be not available. The mean of the training set was 1.5977595, and the standard deviation was 219.39517. In order to normalize, the mean was subtracted from all the signals and the resulting values were divided by the standard deviation.

All the EDF files are in the 16-bit floating-point format; however as the Tensorflow library does not work with the 16-bit floating-point format, all the

data after the preprocessing were all converted to the 64-bit floating-point format. Converting all the data to the 64-bit floating-point format and saving the data as numpy array objects increased the size of the whole dataset from 5.39GB to 14.2GB. From our initial experiments, it was evident that the extra precision degraded the performance of the training process as the speed of the hard drive reading the data could not keep up with the speed at which the model was training. In order to combat this problem, all the data were converted to the 32-bit floating-point format, which decreased the size of the whole dataset to 7.1GB.

As the goal is to have a fast, online automatic annotator for artifacts, no further signal processing or artifact removal currently available was applied. All the data preprocessing steps were done in python.

7.3.3 Preliminary Studies

In order to examine the dataset to learn the general characteristics and the general behavior, a deep learning model with two fully connected layers was built. The input layer was flattened to reduce the dimension so that the fully connected layer that follows can access all the data. Each fully connected layer had 1024 nodes and was activated by a ReLu (Rectified Linear Unit). The ReLu was chosen as the activation function as it tends to have a good convergence and is computationally light compared to other activations such as the sigmoid function. The Adam optimizer (Kingma and Ba 2015) was used, with the default setting. The default setting is that the learning rate is 0.001, the beta-1 value is 0.9, and the beta-2 value is 0.999 with no decay. The Adam optimizer was used for all the experiments as it is computationally efficient and has a small memory requirement. This fully connected model was trained using the training set for 10 epochs with the batch size of 32. The model was validated using the validation set created, and this model was never tested with the test set. The loss function that was used is "categorical_crossentropy," which is defined as below:

$$L_i = -\sum_{n=1}^{N} \left(y_{i,n} \log \left(\hat{y}_{i,n} \right) \right). \tag{7.1}$$

i denotes the index of the observation, and n denotes the class label. y and $\hat{y}$ denote the true label and the estimated probability of the label, respectively. This is a categorical cross-entropy for N number of classes. The model minimizes this loss function by maximizing the estimated probability of the class when the true label for the class matches the estimation. The model trains completely with an accuracy of 94.4%, which is around the accuracy that one will get with a baseline classifier that guesses all the signals as "null" that yields an accuracy of around 94.9%. The relative frequencies of labels other than "null" were so insignificant as shown in Table 7.4 that the model never attempted to optimize the parameters to account for

Table 7.5 Occurrences and the percentage of 1-second segment samples of each label after the subsampling of "null"

Label	Occurrences	Percentage (%)
eyem	7471	26.20
chew	2727	9.56
elpp	2663	9.34
musc	4892	17.16
null	10,763	37.74
Total	28,516	100

artifact labels. This was evident in the behavior of the test and validation losses and accuracies which just fluctuated a bit without making a meaningful movement over the 10 epochs.

In order to combat the label-imbalance problem, another dataset was prepared. In this dataset, the "null" label is sampled such that every 30th "null" observation is included in the dataset. The number 30 was chosen with one purpose of making the "null" label to be not dominating the dataset, but still be the most frequently occurring label. This effectively reduces the number of "null" observations to around 10,000, which still lets this label to be the most dominant, but not overwhelming. After the sampling, the breakdown of the occurrences and the percentage of each label are tabulated in Table 7.5.

Using the newly created dataset, the model was retrained for 10 epochs. During the first two epochs, the validation accuracy increased to 33%, and the accuracy fluctuated around 33% for the rest of the eight epochs. This indicates that the model's complexity is not high enough for this task.

7.3.4 Version 1: Recurrent Neural Network Approach

Using the prior knowledge that EEG signals are temporal, and previous works on detecting artifacts relied on statistical significances of various signal features such as mean and standard deviation, the recurrent neural network (RNN) seems to be a logical choice for the replacement of a network of 2 fully connected layers. The rationale is that since RNNs have access to the previous outputs as well as the current inputs, they would be adept at capturing patterns spread across time. After trying out different combinations of recurrent layers, long short-term memory (LSTM) layer was found out to be the most successful.

LSTM is a specific architecture of an RNN that was proposed by Hochreiter and Schmidhuber in 1997 to combat the vanishing or exploding gradient problems that are common among RNNs (Hochreiter and Schmidhuber 1997). These problems occur as having access to all the previous outputs essentially leads to a large chain of connections between the error and the input. Hence the gradient information could be vanishing or exploding depending on the situation as the information is propagated back to update the weights.

Table 7.6 Model structure for the RNN-based classifier

Layer (type)	Output shape	Number of parameters
Input_1 (InputLayer)	(None, 22, 250)	0
LSTM_1 (LSTM)	(None, 50)	60,200
Dense_1 (Dense)	(None, 1024)	52,224
Dense_2 (Dense)	(None, 5)	5125

The success of a model was determined by predicting the behavior of the training the model from just observing the first few epochs. The different models have been compared by how much training loss was reduced in three epochs and how much validation loss was reduced as a result of those three epochs. For the cases in which the loss function for this dataset did not decrease significantly (by 0.1 or more), the losses never decreased in a reasonable time, and the model tended to overfit to the training data. The final model that was decided is organized in Table 7.6. The total number of trainable parameters is 117,549, and this translates to 225 KB of weights when the weights are saved.

The LSTM layer is to extract the temporal information embedded in the signal. The final dense layers are to do the classification tasks at the end. The parameters on each layer were chosen such that the model is as light as possible without sacrificing significant performance degradation. For the number of cells in the LSTM layer, a varying number of cells was tried such as 5, 10, 25, 50, 100, 200, and 250, and increasing the number of cells decreased the performance by overfitting. However, having too few cells resulted in degraded performance as well. Hence, the number of cells in the LSTM layer was chosen to be 50. The "None" is the placeholder for the batch size. Changing the number of the batch size does not change the number of parameters.

The model was trained on the training data using categorical cross-entropy as the loss function. The model was optimized using the Adam optimizer with the default learning rate and the beta values. The batch size was 32, and the model was trained for 100 epochs. Each epoch takes about 40 s, and the training roughly took about half an hour. The result of this model will be given in the section.

7.3.5 Version 2: Convolutional Neural Network Approach

Another approach that we investigate is using convolutional neural networks (CNNs). As all the channels are available and ordered such that the arrangement reflects the actual spatial closeness of the electrodes roughly, we hypothesize that there will be certain localities across different channels that will be visible in certain

channels. As EEG measures net neural activity, if an area of the brain gets triggered, all the electrodes that are near that area will be triggered, which makes channels that are close in the ordered list to have similar activity. As CNNs are known to work well with image data by using the fact that pixels that are related are close together in images, it seems possible that convolutional layers will also work well with this task. As there is only one-dimensional information available per time frame, 1-D convolutional layers were used instead of 2-D convolutional layers.

While the convolutional layers capture the spatial information, we have added the max-pooling layers to capture the temporal information by grouping up time frames together. Extracting spatial information and temporal information is done multiple times so that any hidden information can be extracted.

Before the max-pooling layers, batch normalization layers are added so that the values of the latent space representation of the input signals are normalized and scaled. Parameter changes in layers during the training cause the layers to yield different outputs each iteration. This forces all the layers to readjust to the new distribution of the outputs every iteration, which delays the training. The batch normalization layer normalizes the activations to reduce these internal covariate shifts to make the training process to be faster, and more stable, especially for deep and large neural networks (Ioffe and Szegedy 2015). Finally, the model has a flattening layer to prepare the data shape to be usable by fully connected layers, and the model uses fully connected layers to do the classification task.

Two versions of the deep convolutional neural network models have been constructed. One version is "deeper" than the other one to see whether adding more layers helped with the classification or not. The structures of both versions are organized in Tables 7.7 and 7.8.

Both versions were optimized using the Adam optimizer with the default setting. The batch size was 32, and the model was trained for 30 and 100 epochs, respectively. The first CNN model was highly overfitting to the train set at around epochs 40, as the validation loss went up by 10 times. The source of this behavior could not be tracked, so the number of epochs that the shallow CNN model was trained for was decreased to 30 epochs. The shallow CNN model took about 20 s per epochs, and the deeper model took about 40 s per epochs.

The hyperparameters used in the model, such as the filter sizes and the output sizes, for the convolutional layers were optimized based on observations of the first few epochs during the training phase just as we did in the development of the RNN based model.

7.3.6 Ensemble Method

In addition to all the methods with different approaches, the final method that incorporates all the models was created. This model takes in the logit outputs of

Table 7.7 Model structure for the shallow CNN-based classifier

Layer (type)	Output shape	Number of parameters
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)	24,704
batch_normalization_4	(None, 128, 32)	128
max_pooling1d_4	(None, 128, 16)	0
conv1d_5 (Conv1D)	(None, 256, 16)	98,560
batch_normalization_5	(None, 256, 16)	64
max_pooling1d_5	(None, 256, 8)	0
conv1d_6 (Conv1D)	(None, 512, 8)	393,728
batch_normalization_6	(None, 512, 8)	32
flatten_1	(None, 4096)	0
dense_1(Dense)	(None, 1024)	4,195,328
dense_2(Dense)	(None, 5)	5125

each of the three models and simply adds the logits to do the decision-making by choosing the label with the highest logit. Different methods of adding up the logits were tested such as weighing one of the three models higher than the other two or excluding one of the models, but weighing all the models equally without exclusion had the highest validation accuracy.

For all the models, binary classification versions were constructed and trained using the same settings to examine how well models detect artifacts. The binary classification task for this problem is determining whether a 1-second segment contains an artifact or not, which will be denoted as either "artifact" or "null." The only deviation for these new models from the original models is the last dense layer. Instead of returning a label of length 5, the binary classification versions return the output label of length 2 (artifact, null). This causes the parameter numbers to be multiplied by 2/5 on the last dense layer. The number of total trainable parameters for the shallow CNN classifier is 4728269, and for the deeper CNN classifier is 11,548,141. When weights are saved, the shallow CNN classifier requires 18.0 MB, while the deep CNN classifier requires 44.1 MB. The results for both versions are given in the following chapter. All the construction of the models and the pipelines for the input and the output for the EEG signals are done in python.

Table 7.8 Model structure for the deep CNN-based classifier

Layer (type)	Output shape	Number of parameters
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)	24,704
batch_normalization_4	(None, 128, 32)	128
max_pooling1d_4	(None, 128, 16)	0
conv1d_5 (Conv1D)	(None, 256, 16)	98,560
batch_normalization_5	(None, 256, 16)	64
max_pooling1d_5	(None, 256, 8)	0
conv1d_6 (Conv1D)	(None, 512, 8)	393,728
batch_normalization_6	(None, 512, 8)	32
max_pooling1d_6	(None, 512, 4)	0
conv1d_7 (Conv1D)	(None, 1024, 4)	1,573,888
batch_normalization_7	(None, 1024, 4)	16
max_pooling1d_7	(None, 1024, 2)	0
conv1d_8 (Conv1D)	(None, 1024, 2)	3,146,752
batch_normalization_8	(None, 1024, 2)	8
conv1d_9 (Conv1D)	(None, 1024, 2)	3,146,752
batch_normalization_9	(None, 1024, 2)	8
flatten_1	(None, 2048)	0
dense_1(Dense)	(None, 1024)	2,098,176
dense_2(Dense)	(None, 1024)	1,049,600
dense_3(Dense)	(None, 5)	5125

7.4 Results and Discussion

After optimizing hyperparameters, and model structures using validation set accuracy, each model was tested using the test set. We find in all the models that there are limitations in precisely predicting labels, and we were interested in whether the models can act as indicators for artifact presence. So, in addition to being trained to do multi-class classification, the models were retrained to do binary classification with the same number of epochs and optimizer settings.

One thing to note for the binary classification is that the evaluation of the binary classification based on the accuracies depends highly on the threshold that is set for the detection. For example, when there are many examples of "null," or no artifacts,

high accuracy could be achieved by intentionally raising the threshold of detection for artifacts high so that most of the examples are classified as "null." Then the system will have high accuracy while failing to act as a respectable classifier for artifacts.

To evaluate the performance of the detection systems receiver operating characteristic (ROC) curves are used, which illustrate the ability of the systems to diagnose with different thresholds. The ROC curve plots the probability of detection versus the probability of false alarm (Richards 2005). The probability of detection which is also known as the true positive rate (TPR), sensitivity, or recall denotes the proportion of actual positives that are correctly identified. Using the problem of this chapter as an example, the true positive rate is the proportion of segments that contain the artifacts that are correctly classified by the model among all the segments that contain the artifacts. The probability of false alarm, which is often referred to as the fall-out, the Type I error, or the false-positive rate (FPR), denotes the proportion of negatives that are misidentified as positives. Using this task as an example again, the false-positive rate would be the proportion of segments that do not contain artifacts that are classified as containing artifacts by the model.

A perfect classifier has a true positive rate of 1.0 and a false positive rate of 0.0, which makes the ROC curve to pass the upper left corner. Hence, a ROC curve that closely approaches the upper left corner indicates a system that discriminates well (Zweig and Campbell 1993). To numerically compare the performance of different ROC curves, the area under the curve (AUC) is computed to indicate how close the ROC curve is to the upper left corner. For example, AUC ranges from 0 to 1, and AUC value of 1 corresponds to the perfect separation case where the true positive rate is 1.0 and the false-positive rate is 0.0 (Hand and Till 2001).

For all the ROC curves provided in this chapter, the area under the curve is also computed and provided.

7.4.1 Recurrent Neural Network-Based Classifier

The recurrent neural network model was trained for 100 epochs. At the end of the training, the train set accuracy was 0.7168, and the validation accuracy was 0.4262. However surprisingly, the test set accuracy was 0.5801, and the confusion matrix is shown in Fig. 7.2.

The model does well on predicting "eyem" and predicting "null." However, the model cannot predict the electrode popping "elpp" label and the muscle movement "musc" label that well. Unfortunately, this pattern persists in all the results. Our conjecture of the behavior of the model is that eye movement and chewing labels have certain localities. For example, we expect electrodes located near the mouth to be more affected by chewing, and electrodes that are far away from mouth to be less affected. This causes specific channels to be affected while leaving other channels to be like "null." As there is a distinguishing feature to be extracted consistently across all the patients, the model does well on the "eyem" and the "chew" labels.

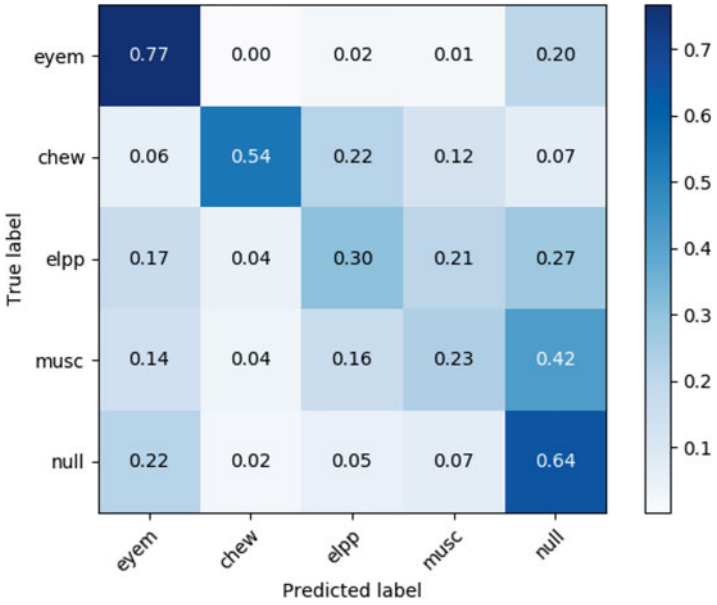

Fig. 7.2 Confusion matrix of the RNN-based model on the test

However, for the cases of "elpp," and "musc", the region of the channels, which are affected, is ambiguous. "elpp" causes similar noise pattern to occur when it occurs, but this can be anywhere, and similar observation could be made regarding "musc."

To see if the RNN-based model is at least powerful enough to indicate the presence of artifacts, the model was retrained to do the binary classification. The RNN-based model trained to the train set accuracy of 0.9885, with the validation accuracy of 0.6254. When tested on the test set, the highest accuracy was 0.7126. In Fig. 7.3, the ROC curve for the RNN based model is shown to visualize the performance of the system. The orange line is the ROC curve, and the dotted blue line is the straight line connecting the (0,0), and (1,1) points. The straight line indicates the worst possible detection system. At around the false-positive rate of 0.424, the true positive rate is 0.800. This indicates that the model would work in a system roughly but would not be recommended in any device that requires high accuracy. The area under the curve is 0.75.

7.4.2 Convolutional Neural Network-Based Classifier

Similar evaluations were done on the shallow CNN model and the deeper CNN model. The confusion matrices are shown in Figs. 7.4 and 7.5, and ROC curves are shown in Figs. 7.6 and 7.7.

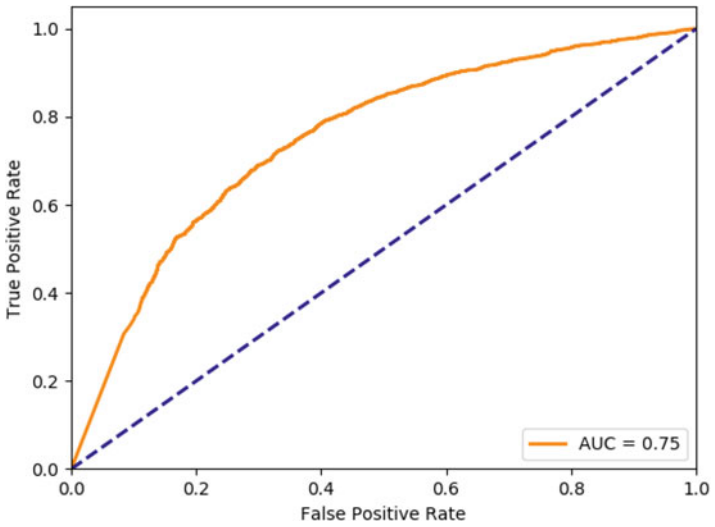

Fig. 7.3 ROC curve for the RNN-based model

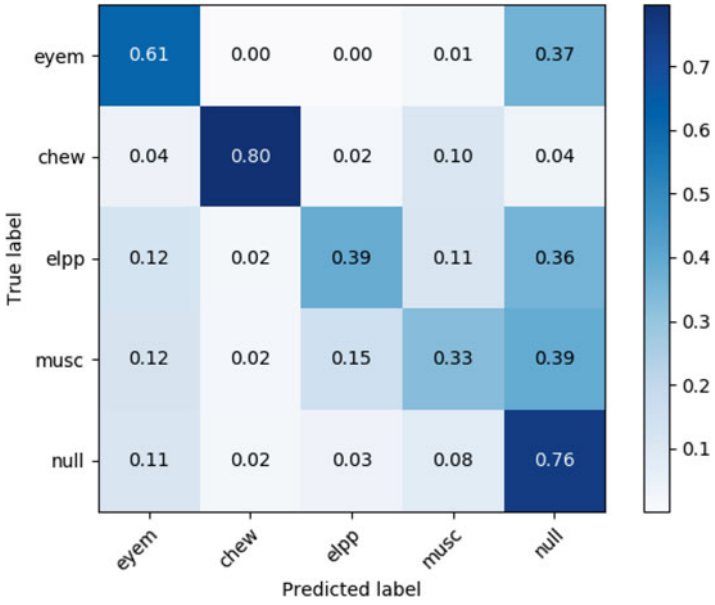

Fig. 7.4 Confusion matrix of the shallow CNN-based model

The shallow CNN-based model was trained for 30 epochs, due to its tendency to overfit when it was trained for more than 40 epochs. The model was trained until the train accuracy of 0.7409 and the validation accuracy of 0.4203. The final test accuracy was 0.6515. Given that both the RNN-based model and the CNN-based

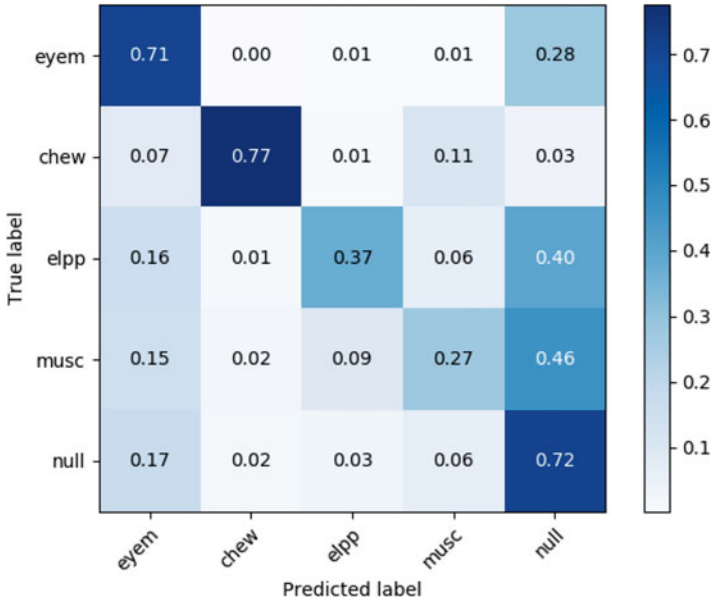

Fig. 7.5 Confusion matrix of the deep CNN-based model

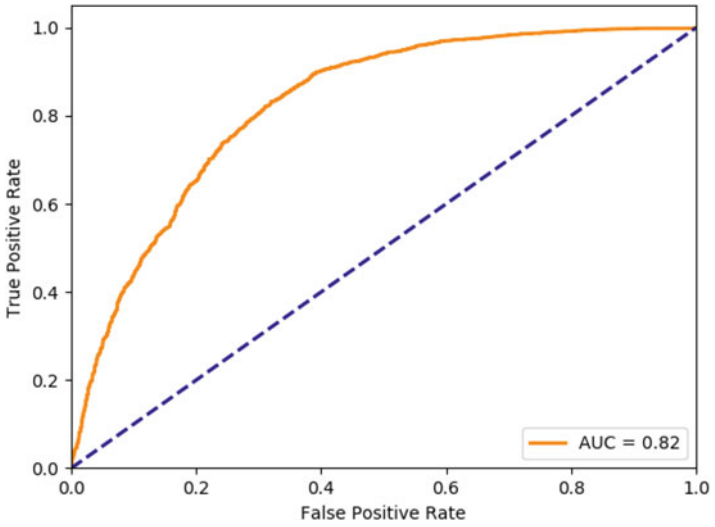

Fig. 7.6 ROC curve for the shallow CNN-based model

model trained until the validation accuracy was around 0.42, the fact that CNN-based model did about 7% better in predicting the 5-class classification problem was interesting. One possibility is that the difference in the complexities of both models causes the difference. Comparing the number of trainable parameters, the CNN-

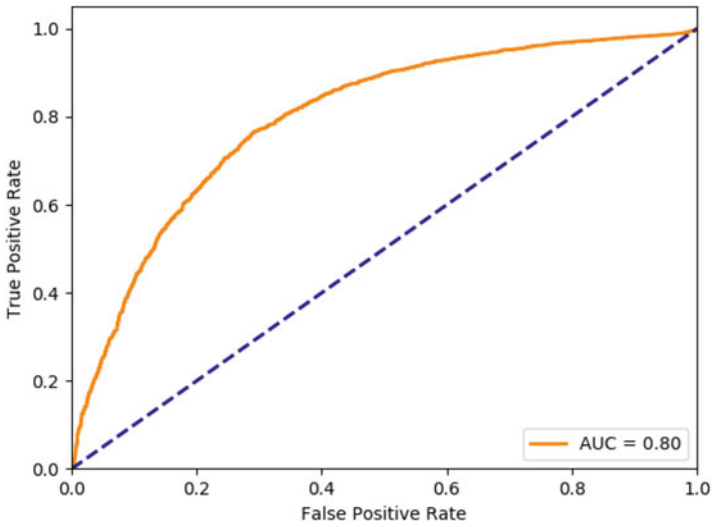

Fig. 7.7 ROC curve for the deep CNN-based model

based model is 4 times bigger, and this may have helped the model to generalize better. However, as evident in Fig. 7.4, this model does significantly better in predicting "eyem" and "chew" than "elpp," and "musc", which is similar to what we observed for the RNN-based model.

The result for the deep CNN-based model is similar. The model was trained to the 100th epochs, the train accuracy of 0.9472 was reached, and the validation accuracy at this epoch was 0.4430. This validation accuracy is slightly higher than that of the shallow CNN-based model. The final test accuracy was 0.6517, which is 0.0002 higher than that of the shallow CNN model. This is likely to be from just noise. The confusion matrix shown in Fig. 7.5 indicates a similar behavior compared to the other models. Hence, we can conclude that CNN-based models work better in multi-class models, but RNN-based model is much lighter, and simply making CNN-based models more complex does not improve the performance of the model significantly.

The more interesting findings are ROC curves. The same analytic method that converts a five-class classification task into a binary classification task was applied to both versions of the CNN-based models just as in the RNN-based model. The shallow CNN model was retrained for 30 epochs, and the deep CNN based model was retrained for 100 epochs. The train set accuracies were 0.8108 and 0.9684, the validation accuracies were 0.5227 0.6008, and the test accuracies were 0.6958 and 0.7499 for the shallow and the deep CNN-based models, respectively. Although these numbers might be misleading as the accuracy depends on the threshold of the binary classifier, for the binary classification problem, the more complex and deeper model has a performance improvement of about 0.05. The receiver operating characteristic curves of CNN-based models are shown in Figs. 7.6 and 7.7.

These ROC curves, compared to that of the RNN based model, have a significantly higher area under the curve, indicating that CNN-based models perform better. Numerically, the areas under the curve for the shallow CNN-based model and the deep CNN-based model are 0.82 and 0.80, respectively, which are larger than that of the RNN-based model which is 0.75. At the true positive rate of 0.800, the false-positive rates are 0.424, 0.295, and 0.339 for the RNN, the shallow CNN, and the deep CNN-based models, respectively. This indicates that CNN-based models can predict the presence of artifact correctly, with fewer false alarms compared to the RNN-based model.

7.4.3 Ensemble Method

Lastly, the ensemble method was examined in the same procedure. The ensemble method incorporates all the other methods by adding the logits produced at the output layers of the other methods. The confusion matrix is shown in Fig. 7.8. The ensemble method's accuracy measures are higher compared to all the other methods, except for the "musc" label. The shallow CNN-based model achieves the accuracy of 0.33 on the "musc" label, while the ensemble method achieves 0.28. Regardless, the ensemble method achieves the overall accuracy of 0.6759, which is the highest among all the methods. In addition to the confusion matrix, the ROC curve for the binary classification version of the model is produced. The ROC curve is shown in Fig. 7.9, with all the ROC curves from other models for better comparison.

Interestingly the ROC curve for the shallow CNN-based model has a similar area under the curve as the ensemble method. The shallow CNN-based model has higher true positive rates in certain regions than the ensemble method, and the ensemble method performs superior to the shallow CNN-based model in the regions of lower thresholds.

For the binary classification problem, as the main purpose is to accurately point out the artifact events, the time-lapse system was proposed to further enhance the performance. The idea comes from the fact that artifacts often come in bursts, such that the previous segment's label correlates well with the new segment that follows. This method does not change any of the models but rather works directly on the logits produced by the models. A sliding window adds all the logits in the window to produce a new logit that the classifier uses. Different methods of producing the new logit were tried such as taking the maximum or doing a weighted sum of the logits, but simply adding all the logits worked the best. Different sizes of sliding windows were tried, ranging from 1 to 10, but a 2-second window produced the best result. The ROC curves for the highest performing window setting are shown in Fig. 7.10.

The time-lapse method improves all the ROC curves, especially lifting the regions in the lower false positive rates. At the true positive rate of 0.800, the new time-lapse method yields the false-positive rates of 0.310, 0.288, 0.268, and 0.258 for the RNN-based, the shallow CNN-based, the deep CNN-based, and ensemble

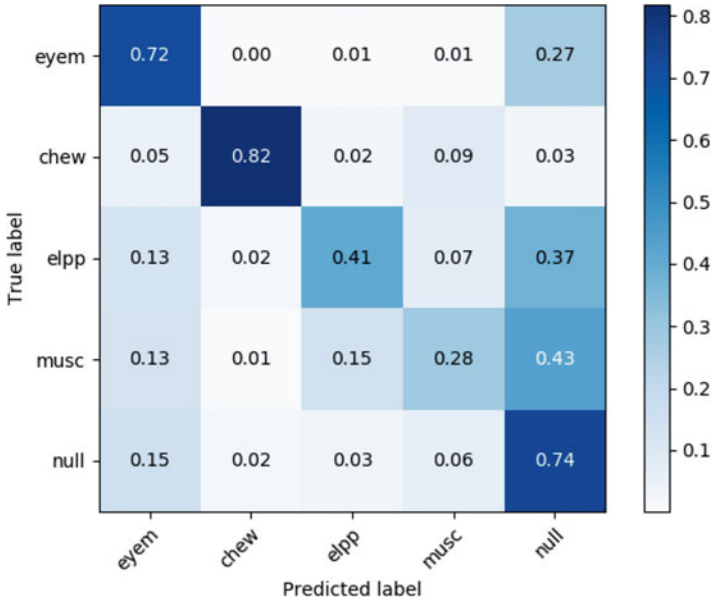

Fig. 7.8 Confusion matrix of the ensemble method

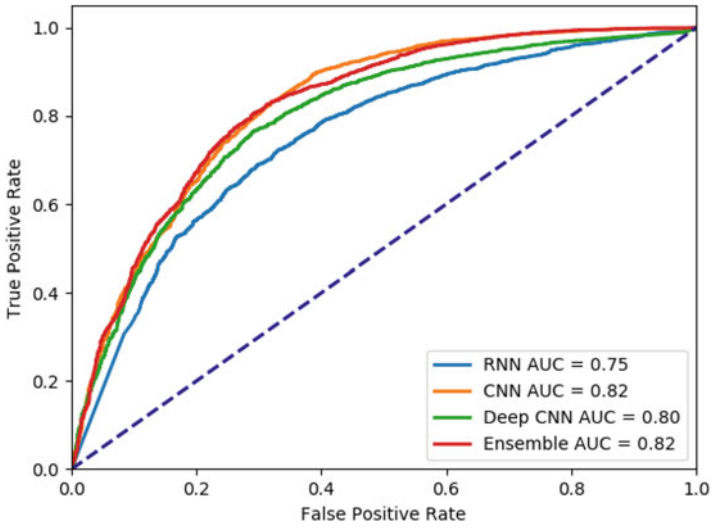

Fig. 7.9 ROC curves for all the models

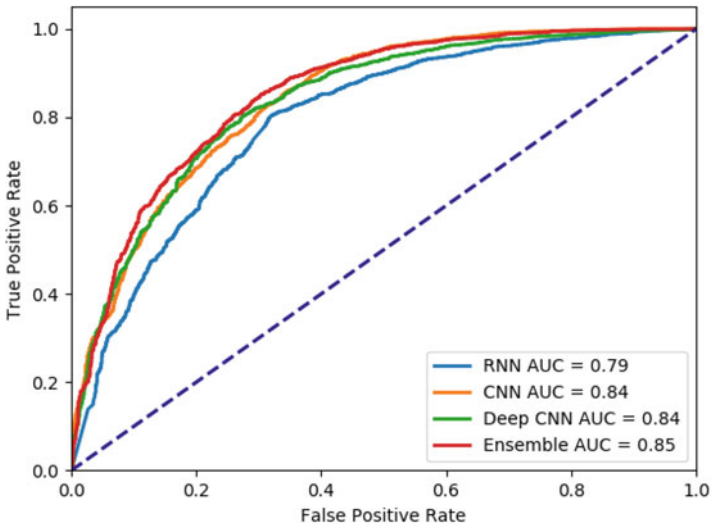

Fig. 7.10 ROC curves for all the models with the time-lapse method

methods, respectively. This is a slight improvement from the false-positive rate of 0.295 from the shallow CNN model without the sliding window. The ensemble method does the best for this method proposed.

In order to see the viability of the model in real-life settings, all the binary classification models were tested on a test set that contains all the "null" information without the sampling procedure. The five-class classification accuracies of the models are 0.7234, 0.7612, 0.7534, and 0.7808, for the RNN based, the shallow CNN-based, the deep CNN-based, and ensemble methods, respectively. Only one confusion matrix from the best result is shown as all the confusion matrices behave similarly. The resulting confusion matrix is shown in Fig. 7.11. The increase in the accuracy comes from the fact that there are more "null" labels in the dataset; hence the accuracy converges to the accuracy of predicting the "null" label which is around 0.78 for the ensemble method.

The ROC curves for the binary classification problem using all the models on the original data are shown in Figs. 7.12 and 7.13. Figure 7.12 shows the ROC curves of the models without the time-lapse method, and Fig. 7.13 shows the ROC curves of the models with the time-lapse method. The areas under the curves are significantly higher than those of the sampled data cases. These curves indicate the viability of the models in a real clinical setting.

Lastly, the average time elapsed in processing one example was computed for each model, for each classification problem to see whether the model is feasible for doing an on-line signal processing task of indicating whether the artifact exists or not. For the reference, there are 5797 observations in the test set. In addition, the time elapsed while loading the Tensorflow module and the libraries as well as loading the data was not accounted for. The results for the accuracy with default

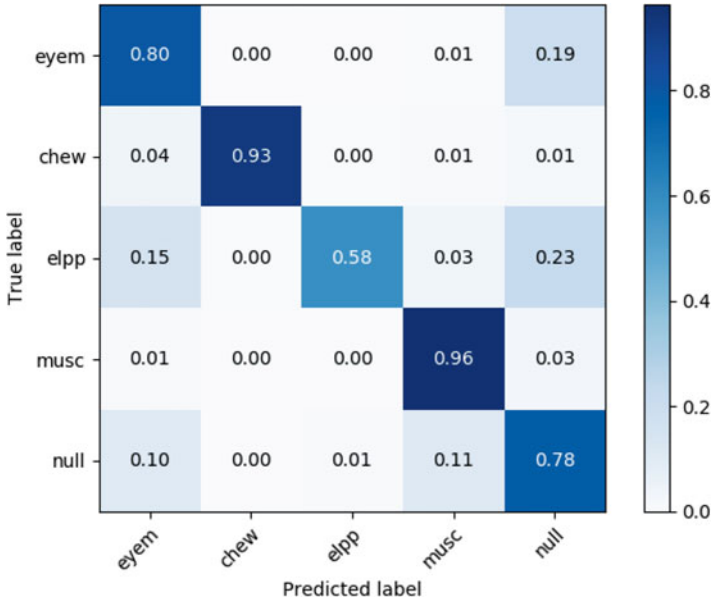

Fig. 7.11 Confusion matrix of the ensemble method on the original data

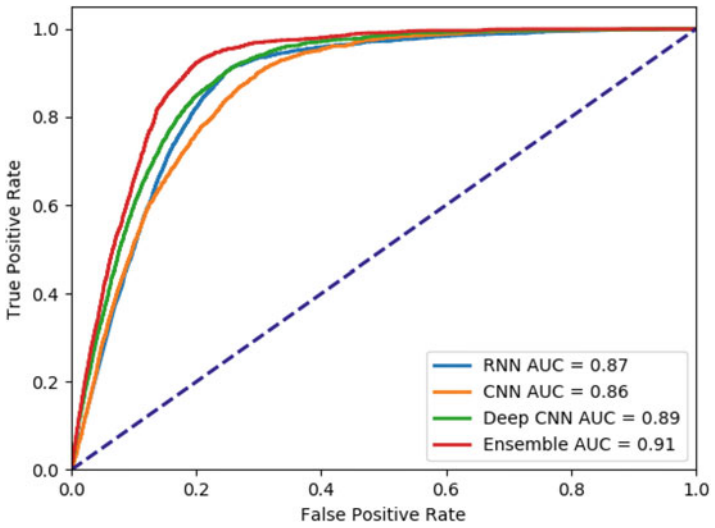

Fig. 7.12 ROC curves for all the models on the original data

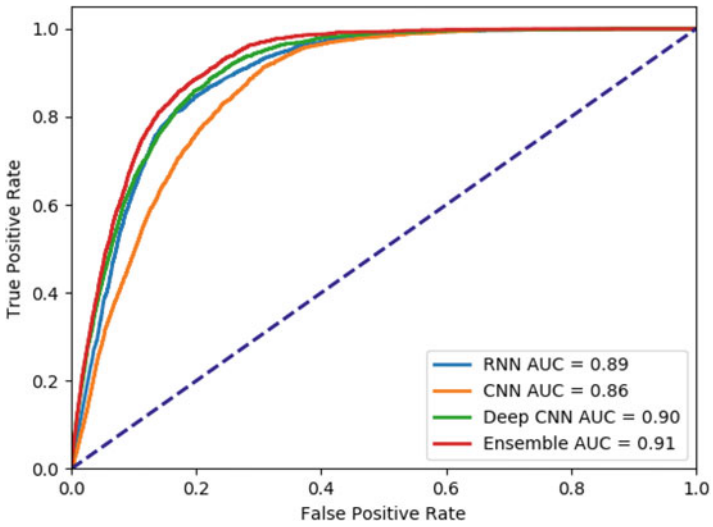

Fig. 7.13 ROC curves for all the models with the time-lapse method on the original data

Table 7.9 The Time elapsed, the accuracy, and the size of each model

Model	Average time elapsed (ms/sample)	Test set accuracy (%)	Size of the model (KB)
RNN	0.707	58.01	476
RNN-binary	0.677	71.26	464
CNN	0.483	65.15	18,526
CNN-binary	0.468	69.58	18,514
DeepCNN	0.595	65.17	45,189
DeepCNN-binary	0.568	74.99	45,177
Ensemble	N/A	67.59	64,191

thresholds, which looks at the maximum confidence level of each label, the average time elapsed, and the size of each model are tabulated in Table 7.9. All the test results on this table are from the sampled test data.

All the average time elapsed for inference for all the models is less than 1 ms, for each of the 1-second segment. This indicates that the model is able to predict the presence and the kind of artifact almost instantaneously. Also, the sizes of the models are small enough to be implemented in a Raspberry Pi, which could make this model highly portable. Since the original EEG signals were expressed in 16-bit floating-point values, the model can be further compressed if all the parameters are converted to 16-bit floating-points instead of 32-bit floating-points. This compression is approximately half the size of the model, further improving the portability. All the evaluations were done in python.

7.5 Conclusion

The chapter proposes three types of deep learning-based machine learning model that learns to distinguish artifacts from the real signal and classify artifacts. Three models, the RNN-based model and the two CNN based models of different depth, have been constructed and evaluated. In addition, the ensemble model was created that utilizes all the other methods. The ensemble model, which has the best overall performance, achieves a 67.59% five-class classification accuracy, and a true positive rate of 80% at the false positive rate of 25.82% for the binary classification problem. The models are light and fast enough to be implemented in a portable device, such as Raspberry Pi. The largest model only has 65 MB of trainable parameters, and the slowest model only takes about 0.7 ms to predict on a 1-second long EEG signal. The speed of the ensemble model has not been tested but given that the slowest component in the model occupies less than 0.1% of the segment implies we expect it to be fast enough for the goal. We expect the time elapsed to be slightly more than the three models combined. As this model can successfully detect whether artifacts are present in the collected signals quickly, and can tell what type of artifacts they are, physicians can use this device while collecting data to check whether the data that are being collected are free of artifacts or not. If the data are being affected by any artifacts, physicians can quickly check which artifact is present and act in response to that artifact. This work is significant to the research community as it adds deep learning as one of the tools that the community can use in recognizing artifacts in EEG signals and potentially removing them also.

Clinicians indicate that a sensitivity, which is the true positive rate, of 95%, and specificity, the false positive rate, of below 5% to be the minimum requirement for clinical acceptance (Golmohammadi et al. 2019). As none of the models achieve that guideline yet, there are many more investigations needed in optimizing the models. Hence for future works, an investigation into incorporating different features that can be extracted quickly, and larger and more complex models to reach the recommended guideline can be done. In addition, since the models were trained, validated, and tested on the first version of the EEG artifact corpus which only consists of observations from 310 patients, in the future when there are more data available, the model could be trained again to see whether the lack of data was part of the inadequate performance. Also, since classification within the artifacts, excluding the "null" label, seems to work at high accuracies evident from the confusion matrix, and the binary classification of artifacts can have arbitrarily high true positive rate, an investigation on a two-step system seems to be another interesting path to take on. This research envisioned to have a portable device that can be used during data acquisition. Building a portable machine that runs these models to predict the presence of artifacts and to classify the artifacts should be the next step. Finally, testing this machine in a real-life setting will be beneficial to see if the machine works and to see if there are additional adjustments and improvements to make.

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

M. Aminoff, Principles of neural science. 4th edition. Muscle Nerve **24**(6), 839–839 (2001)
W. Clercq, A. Vergult, B. Vanrumste, W. Paesschen, S. Huffel, Canonical correlation analysis applied to remove muscle artifacts from the electroencephalogram. IEEE Trans. Biomed. Eng. **53**(12), 2583–2587 (2006)
J. DellaBadia, W. Bell, J. Keyes, V. Mathews, S. Glazier, Assessment and cost comparison of sleep-deprived EEG, MRI and PET in the prediction of surgical treatment for epilepsy. Seizure-Eur. J. Epilepsy **11**(5), 303–309 (2002)
M. Golmohammadi, A. Torbati, S. Diego, I. Obeid, J. Picone, Automatic analysis of EEGs using big data and hybrid deep learning architectures. Front. Hum. Neurosci. **13** (2019)
I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning* (MIT Press, 2016)
D. Hand, R. Till, A simple generalisation of the area under the ROC curve for multiple class classification problems. Mach. Learn. **45**(2), 171–186 (2001) Retrieved from https://academic.microsoft.com/paper/1912982817
A. Hasasneh, N. Kampel, P. Sripad, N. Shah, J. Dammers, Deep learning approach for automatic classification of ocular and cardiac artifacts in MEG data. J. Eng. **2018**, 1–10 (2018)
S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift. International conference on machine learning, 448–456 (2015)
D. Kingma, J. Ba, *Adam: A Method for Stochastic Optimization* (International conference on learning representations, 2015)
V. Krishnaveni, S. Jayaraman, A. Gunasekaran, K. Ramadoss, Automatic removal of ocular artifacts using JADE algorithm and neural network. *International Journal of Computer and Information*. Engineering **2**(4), 1330–1341 (2007)
T.-W. Lee, M. Girolami, T. Sejnowski, Independent component analysis using an extended infomax algorithm for mixed subgaussian and supergaussian sources. Neural Comput. **11**(2), 417–441 (1999)
S. Lopez, A. Gross, S. Yang, M. Golmohammadi, I. Obeid, J. Picone, An analysis of two common reference points for EEGS, in *2016 IEEE Signal Processing in Medicine and Biology Symposium (SPMB), 2016*, (2016), pp. 1–5. Retrieved from https://academic.microsoft.com/paper/2586273059
E. Louis, Frey, L., Britton, J., Hopp, J., Korb, P., Koubeissi, M., ... Pestana-Knight, E. Electroencephalography (EEG): An Introductory Text and Atlas of Normal and Abnormal Findings in Adults, Children, and Infants (2016)
H. Nolan, R. Whelan, R. Reilly, FASTER: Fully automated statistical thresholding for EEG artifact rejection. J. Neurosci. Methods **192**(1), 152–162 (2010)
I. Obeid, J. Picone, The Temple University Hospital EEG data corpus. Front. Neurosci. **10**, 196 (2016). https://doi.org/10.3389/fnins.2016.00196
M. Richards. Fundamentals of Radar Signal Processing (2005)
Y. Roy, H. Banville, I. Albuquerque, A. Gramfort, T. Falk, J. Faubert, Deep learning-based electroencephalography analysis: a systematic review. arXiv preprint arXiv **1901.05498** (2019)
B. Singh, H. Wagatsuma, A removal of eye movement and blink artifacts from EEG data using morphological component analysis. Comput. Math. Methods Med. **2017**, 1861645 (2017)
M. Zweig, G. Campbell, Receiver-operating characteristic (ROC) plots: A fundamental evaluation tool in clinical medicine. Clin. Chem. **39**(4), 561–577 (1993)