EOG Artifact Rejection

Edward Chan

University of California Santa Cruz

[edcchan@ucsc.edu](mailto:edcchan@ucsc.edu)

Wentao Xu

University of California

Santa Cruz

[wxu22@ucsc.edu](mailto:wxu22@ucsc.edu)

Scott Liu

University of California Santa Cruz

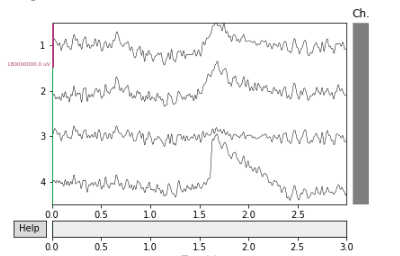
[sxliu@ucsc.edu](mailto:sxliu@ucsc.edu)

# 

# **ABSTRACT**

Electroencephalography (EEG) evaluates electrical activity in the brain and is conducted to diagnose various mental disorders, such as seizures or behavior. However, electrooculogram (EOG) artifacts (electrical “noise” generated by eye movements) distort the desired pure EEG data, thus hinder proper interpretation of EEG signals. Our goal is to train a model that can recognize the presence of EOG artifacts. The dataset being used is pure EEG readings manually contaminated with EOG artifacts. Eighty percent of these samples were used for training and twenty percent for testing. The dataset is stored as MatLab files and undergoes preprocessing, which will then be fed into our model. In total, 486 samples were used after augmentation, collected from 54 different EEG readings. We propose a model that will classify if a segment of data three seconds long contains artifacts or not. The overall accuracy in training achieved close to one-hundred percent, but achieved only sixty-five percent accuracy in testing. Many factors contributed to such poor testing accuracy, but, undoubtedly, our inexpertise in classifying segments of data as contaminated by EOG artifacts or not was the biggest factor. Our model was trained on labels that we manually inputted, subject to our inexpertise.

# **MOTIVATION AND OBJECTIVE**

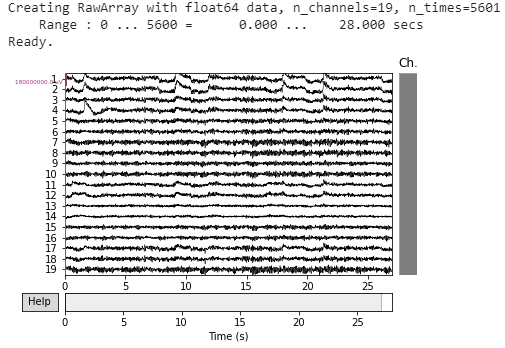
The primary objective of this project was to build and train a deep learning model that would read different segments of EEG signals and discover which segments of data were contaminated by EOG artifacts. 

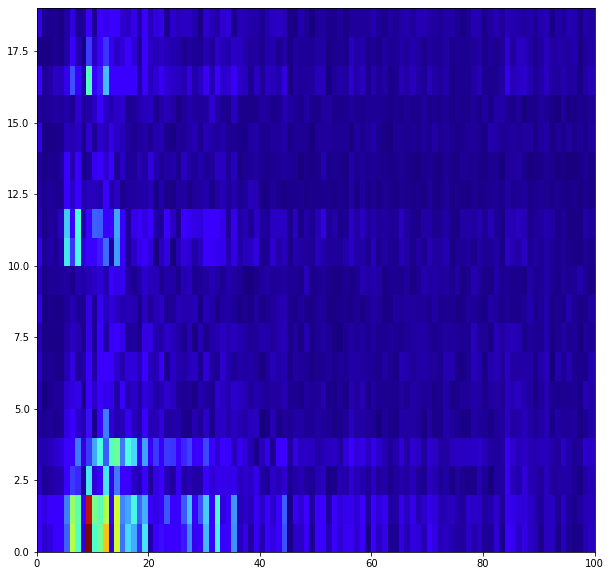
The peak in this segment of the EEG reading represents an EOG artifact. Thus, if we were to feed this segment of data into our model, it should identify the presence of an artifact.

Currently, there already exists multiple different research groups that have developed algorithms for this purpose, such as using an ICA Regression Method, or also by using Wavelet Enhanced Independent Component Analysis. While our algorithm to detect EOG artifacts is not novel, the primary objective of training our deep learning model to detect and automatically remove EOG artifacts has not been accomplished as of yet, and as such is our primary motivation for this project.

# **DATASET**

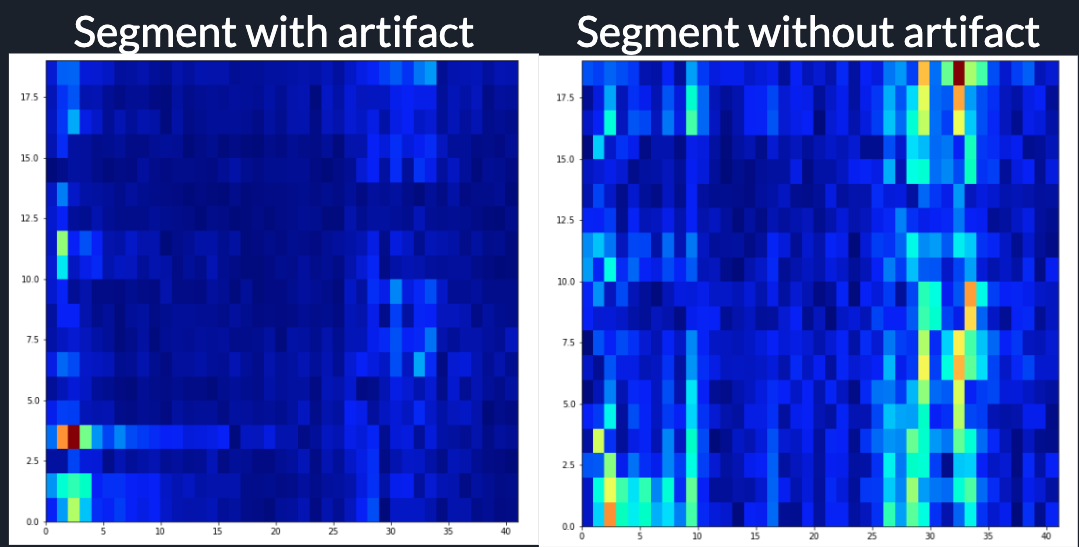
The dataset worked on will be a semi-simulated EEG dataset taken from Mendeley Data, a free open-source data repository. It contains 54 clean samples taken from 27 subjects of both male and female. Each sample includes readings from 19 electrodes attached to various parts of the subject’s head, taken at frequency of 200Hz for 30 seconds. These samples are then artificially contaminated with ocular artifacts by adding readings of horizontal and vertical eye movements to each corresponding clean data.

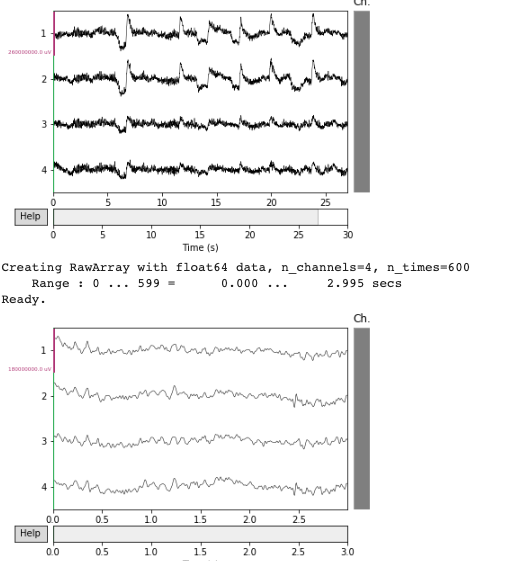


The data is stored in four matlab files: clean samples, contaminated samples, horizontal electrooculogram artifact and vertical electrooculogram artifact. We used scipy to import the data into python as dictionaries. Then key-value pairs containing meta data will then be removed. We divide each contaminated sample into three second intervals, which contains one thousand elements, for training. Since the samples range from 5601 readings to 8401 readings, we take the minimum of all the readings and cut off the excess to keep the data uniform. Each three second interval of data will be converted into a tensor.

Once we convert our data into a tensor, we apply the Fast-Fourier Transformation (FFT) to it. This transforms the electrical signals from the time domain into the frequency domain to be plotted in a frequency power spectrum. Once the transformation is performed, we will convert the data back into a numpy array. We perform this process of reshaping the data to essentially turn our input data, which is lengthy and time-dependent (not useful to our model), into an array which will represent our input layer. When we plot the data post-FFT, the graph is composed of different colored rectangles, similar to pixels in images.

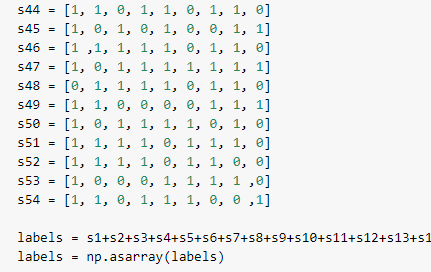
This is the plot of our data after applying the fast fourier transform. The different colors represent different signal intensity, with the red representing higher microvolts and blue representing close to none. Since there’s little variance after 40 hertz, restrict all transformed data to just 40 hertz.

We wanted to see if the plots between data segments containing artifacts differed from those without artifacts. The left graph depicts the first the fast fourier transformed data where there is an artifact and the right depicts the transformed data where there is no artifact. We see that there exists a lot more variance in signal intensity from 25 hertz to 35 hertz in the artifact free data. This trend is repeated when looking at different segments of data, confirming our suspicions. Seeing there is a visible difference in our data when plotting it, our model should be able to also recognize a difference in the data, correctly identifying if a segment of data contains an artifact or not.

We decided that our model would take in 3 seconds worth of data, and since we had 54 different EEG readings each about 27 seconds long, we had to manually input 486 labels, where 1s represent the existence of artifacts and 0s represent nonexistence of them. 

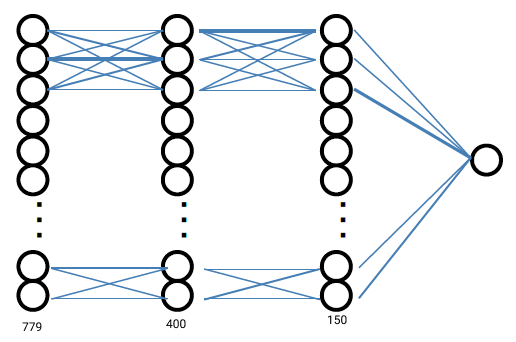
We did this by viewing two graphs, with the top showing the 27 seconds of one EEG reading and the bottom showing the 3 second interval we wanted to examine. Therefore, each 27-second EEG reading created an array of 9 elements, each element representing a 3 second interval. We only used the first four channels because they best reveal the existence of EOG artifacts.

The labels are binary arrays, with 1 representing the interval containing the artifact, and 0 represents pure data.



We tried using the raw data as an input with a shape of 19 x 5601, and the clean data as labels. But the model we have requires a binary classification of the output: either it contains artifacts or it doesn’t. So we ended up creating the labels manually.

# **MODELS AND ALGORITHMS**

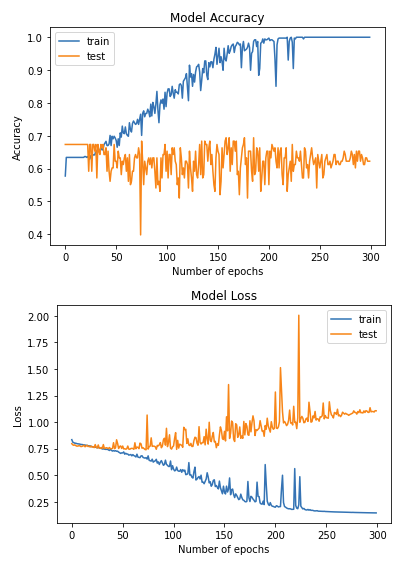
Our input layer was composed of 779 neurons because that was the shape of the data after we applied the fast fourier transform on a single tensor. The deep learning model that yielded the best results was composed of two hidden layers, with the first containing 400 neurons and the second containing 150 neurons. The output layer consisted of a single neuron. Relu activation was applied to every layer except for the output layer, where sigmoid activation was applied. We used binary cross entropy as the loss function because the task is a binary classification problem. This model achieved around 65% accuracy on testing and close to 100% in training, so in attempts to reduce overfitting, we created other models.

One model utilized 20% Dropout at each layer, resulting in testing accuracy around 60%. Another model utilized L2 regularization, achieving around the same accuracy as the Dropout model. We also attempted only one hidden layer with 500 neurons, still resulting in similar, poor accuracy. Adding a hidden layer, so a total of three hidden layers, yielded the worst accuracy out of any other model. We did not attempt a model that utilized early stoppage, because there was never an epoch in the model history where validation accuracy showed a general negative trend - validation accuracy increased in the first epochs and just plateaued.

# **RESULTS AND ANALYSIS**

We attained close to 100% accuracy on the data classification over the training data. But there is a significant overfitting problem as the test data only showed roughly 65% accuracy. We attempted many different models with the intent to reduce overfitting to no avail.

The reason for such poor accuracy has to do with incorrect labelling. There were many instances where we viewed segments of data and were unsure whether the segment absolutely contained an artifact or absolutely did not. Thus, the poor performance of our model should be expected, because if we are teaching the model with labels that we are not completely sure are correct, then the model will undoubtedly experience the same difficulty.



# **CONTRIBUTIONS**

All three members did extensive research on how to use MNE to help with data visualization and how to work with EEG datasets. In addition, each member helped to manually make the 486 labels. Edward Chan wrote all the code for data preprocessing and the different models.

Wentao Xu went to all the meetings with the teaching assistants to ask for help when the team members were working on issues with preprocessing data especially.

Scott Liu made and edited the video presentation.

**FUTURE WORK**

We can further improve the model by developing a more advanced model to attain better test accuracy. For instance, we can work towards transforming the data so that it can be fed into a CNN, and we need to reduce the currently overfitting problem. Once we obtain a higher accuracy, we can then work towards developing an algorithm that, given a contaminated EEG reading, will adjust the elements in each channel such that the output will be similar to its corresponding pure readings.

The time needed to further increase the accuracy to an acceptable degree would require at least another quarter of work. After our accuracy has been increased and the overfitting problem has been solved, we would then be able to proceed with the objective of automatically removing EOG artifacts from the data, this will likely take a similar amount of time.

This project is worth the time investment for future work, if all the objectives were to be accomplished, the efficiency at which EEG signals can be read and analyzed will increased significantly