EEG event Classification using Convolutional Neural Networks and the Temple University Hospital EEG Corpus

Joshua Levitt, M.S..
Joshua_Levitt@alumni.brown.edu

This report is intended only to be exploratory and illustrative of some preliminary work. It is not intended for academic or commercial publication. If you have any questions about this project, or the underlying code, please contact Josh Levitt at Joshua_Levitt@alumni.brown.edu

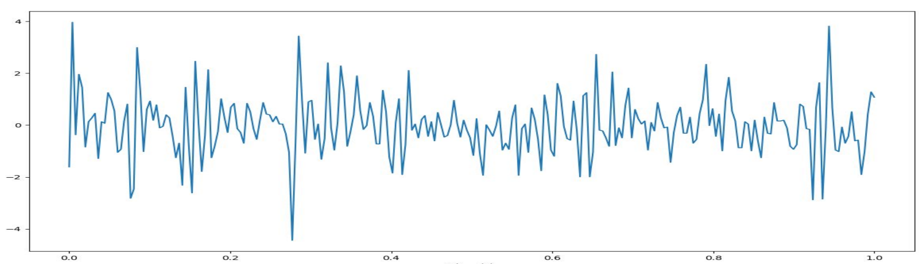
---

Electroencephalography (EEG) is an essential medical imaging tool used to non-invasively examine the electrical activity of the brain. The EEG community is beginning to realize to power of using machine learning methodologies to better analyze and understand this tool.

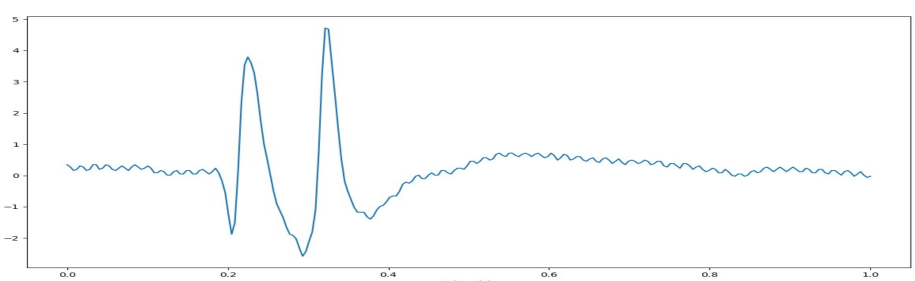
Doing so requires access to powerful to a large set of EEGs. The largest publicly available EEG dataset is the Temple University Hospital EEG Corpus (TUHEEGC). Included in the TUHEEGC is the six-way event classification dataset, which contains EEG files with thousands of events labeled into into six categories. The names of these, and an example of each is shown below. I chose to use these labeled events to train a 1-dimensional Convolutional Neural Network (CNN) to classify these events. I chose this model because CNNs can learn to recognize spatial patterns, like the characteristic shape of a waveform during an eyeblink, which is analogous to how a human might approach this task.

<u>Event Classes</u>

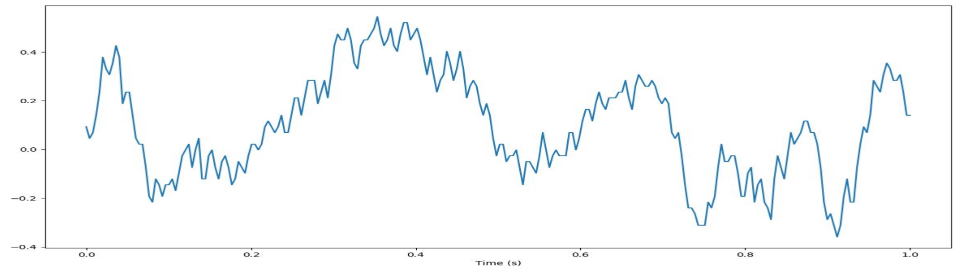
1: Spike and Slow Wave



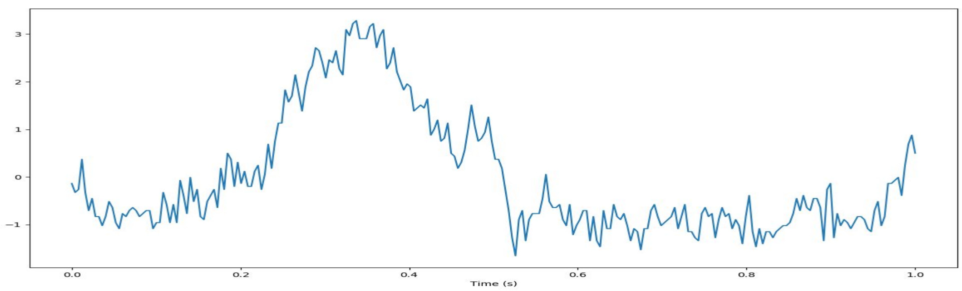
2: Generalized Periodic Epileptiform Discharge



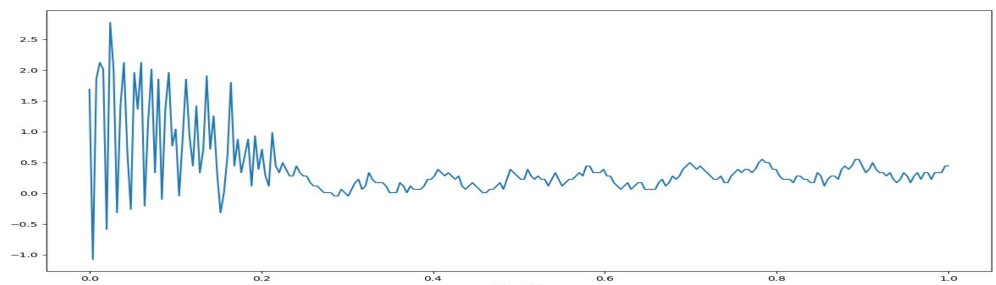
3: Periodic Lateralized Epileptiform Discharge



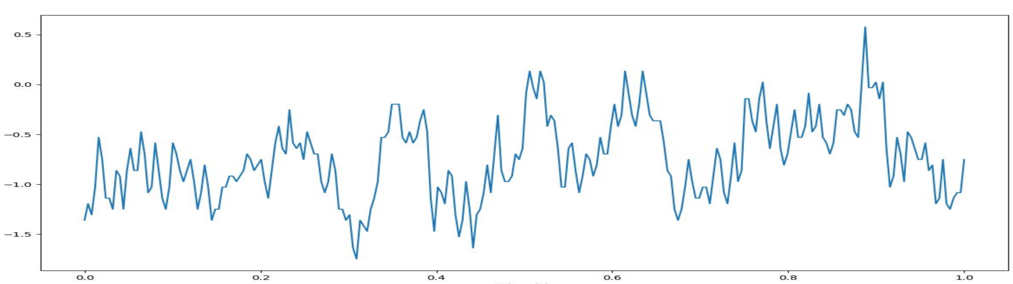
4: Eye Movement



5: Artifact



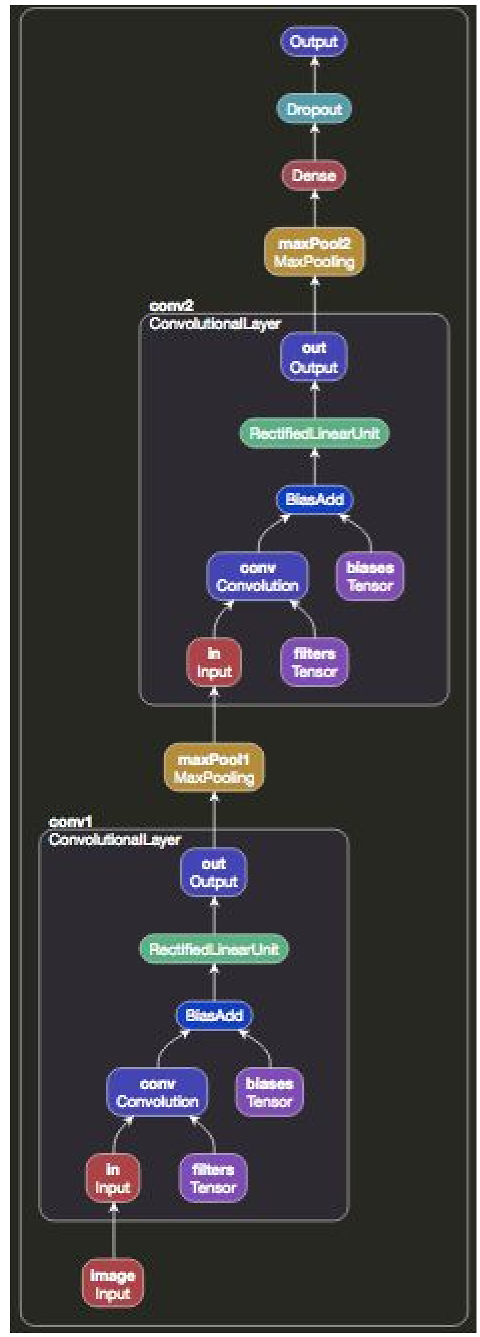
6: Background



Each of these events is one second long, sampled at 250Hz, such that each event contains 250 sample points, and can be represented as an array with dimensions 250 x 1. However, each event has been labeled such that the most critical piece of the event is centered within that one second window. This could pose a problem in accurately classifying off-center events. In order to solve this problem, and increase my sample size, for each event, I created 11 one-second long samples that swept across the event, beginning a half-second before the start of the event, and ending a half-second after its end. This gave me a sample size of over 320,000.

This was partitioned into training and evaluation datasets. The Training set was used to train a convolutional neural network in TensorFlow. The structure of this network was as follows:

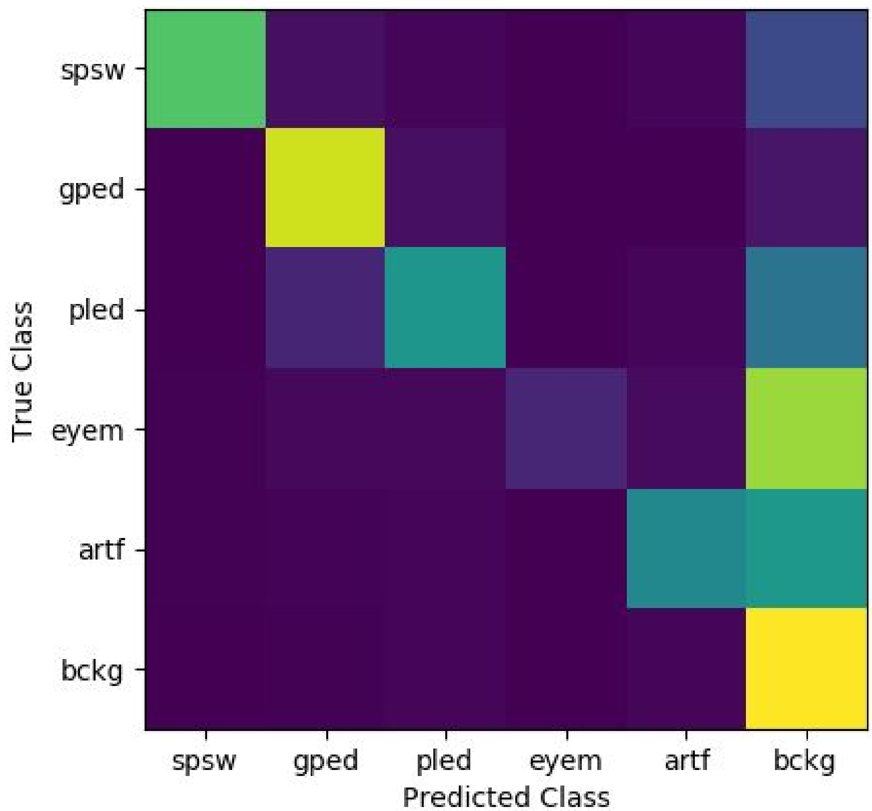
Input Layer {Dimensions: 250 x 1}
First Convolutional Layer {
    Num_filters = 40
    Kernel_size = 10
    Dimensions = 250 x 40}
First Pooling Layer {
    Pool_size = 2
    Stride_size = 2
    Dimensions = 125 x 40}
Second Convolutional Layer {
    Num_filters = 40
    Kernel_size = 10
    Dimensions = 125 x 40}
First Pooling Layer {
    Pool_size = 5
    Stride_size = 5
    Dimensions = 25 x 40}
Fully Connected Layer {
    Dimensions = 600 x 1
    Dropout rate = 0.4}
Output Layer {Dimensions: 6 x 1}



(Network Diagram created using Moniel -- https://github.com/mlajtos/moniel)

This model was trained using a training set size of 200,000, a batch size of 100, a learning rate of 0.005, and momentum of 0.1. After 240000 batches of training, it had an evaluation accuracy of 87.27%, and an ROC-AUC of 0.74. To the best of the author's knowledge, this is the best performance ever achieved on this dataset. It should be noted that this was achieved with minimal optimization of the hyper-parameters (i.e the structure of the network) and there is therefore likely still room for improvement. I predict that a classification accuracy of over 90% is possible using a similar approach.

The evaluation confusion matrix was as follows:



There are several notable things about this dataset, but one of the most important is that the class sizes are very uneven, with the largest (background) representing ~67% of the total, and the smallest (eye movement) only ~1%. This poses challenges for robust learning. Possible solutions include over-sampling the rare classes, or under-sampling the common classes. I have not explored this possibility, but it may be possible to further increase classification accuracy, or selectivity/sensitivity in this way.

This report is intended only to be exploratory and illustrative of some preliminary work it is not intended for academic or commercial publication. If you have any questions about this project, or the underlying code, please contact Josh Levitt at Joshua_Levitt@alumni.brown.edu