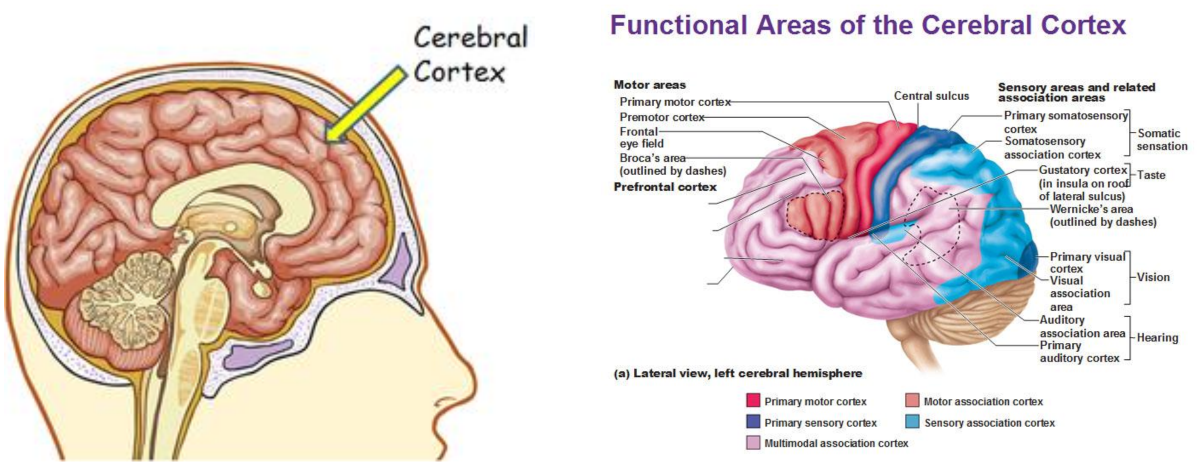
Using Deep Learning to Map Electroencephalogram Readings to Activation Regions in the Brain

Varderes Barsegyan
California State University, Long Beach

Introduction & Background

The Cerebral Cortex

The cerebral cortex is the outermost layer of the mammalian brain. It is responsible for key cognitive roles such as language, memory, perception, and consciousness. Composed of gray matter, it has a folded structure that is about 3 millimeters thick covering the entirety of the brain's surface. Three main areas perform sensory, motor, and associative functions persistent in complex and intelligence beings. Sensory areas receive and process information pertaining to the senses: visual cortex, auditory cortex, and somatosensory cortex. Motor areas govern voluntary movement of limbs. Associative areas, perhaps the most complex and most distinguishable, provide abstract thinking and language.


Cerebral Cortex
Functional Areas of the Cerebral Cortex
Motor areas
Primary motor cortex
Premotor cortex
Frontal eye field
Broca's area (outlined by dashes)
Prefrontal cortex
Central sulcus
Sensory areas and related association areas
Primary somatosensory cortex
Somatosensory association cortex
Somatic sensation
Gustatory cortex (in insula on roof of lateral sulcus)
Taste
Wernicke's area (outlined by dashes)
Primary visual cortex
Visual association area
Vision
Auditory association area
Primary auditory cortex
Hearing
(a) Lateral view, left cerebral hemisphere
Primary motor cortex
Motor association cortex
Primary sensory cortex
Sensory association cortex
Multimodal association cortex
Figure 1: The cerebral cortex and its functionalities

At the molecular level, brain activity involves the movement of ions within interconnected neurons. Although the brain is enclosed by the skull, these ionic movements induce voltage fluctuations and so if the brain is active in any way an electric field is present outside of the skull. Electroencephalograms (EEGs) are machines designed to monitor the brain's electric activity and use this data to better understand cognitive and emotional states, create brain-computer interfaces (BCIs) and brain-brain interfaces (BBIs), and diagnose patients with brain disorders such as epilepsy and concussion. EEG recordings are usually done using noninvasive electrodes places strategically along the scalp. Each electrode tries to pinpoint a frequency range that represents a specific type of activity. Noninvasive methods, alongside a few invasive methods, are used by cognitive scientists, neuroscientists, and biomedical engineers to further research and establish a thorough understand of the complex workings of the brain.

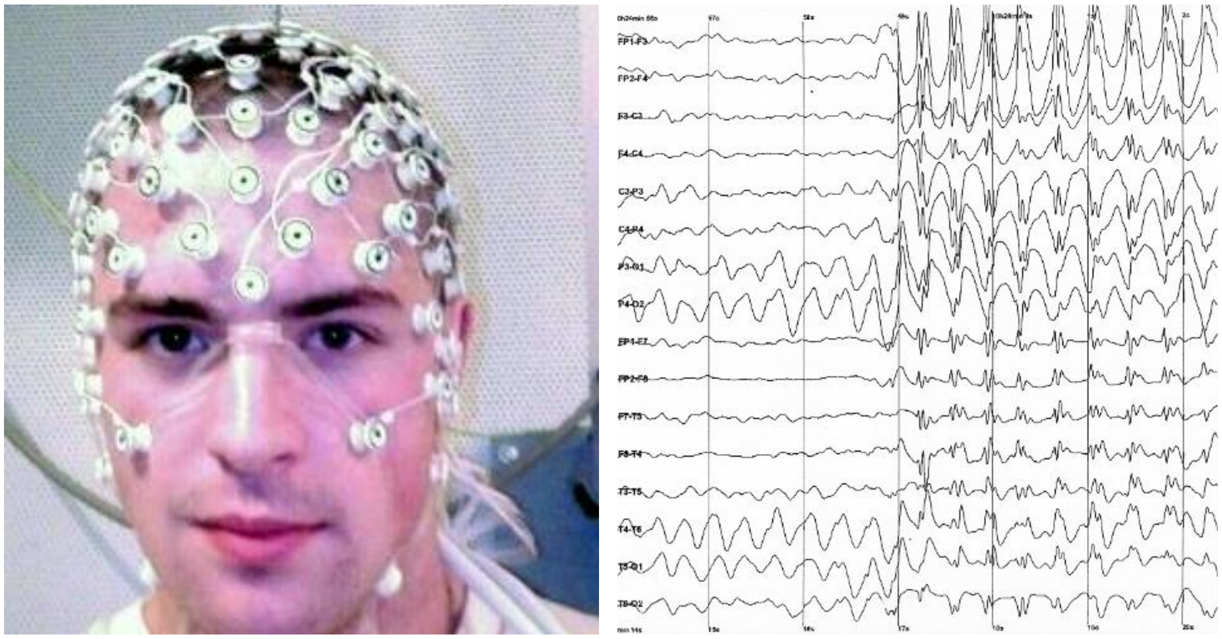


Figure 2: A gentleman wearing electrodes around his head. It generates data like what is shown on the right.

Interestingly, the brain can also be stimulated in specific regions to create a two-communication between an interface and the brain. In 2013, researchers at the University of Washington created the first ever noninvasive human-to-human brain interface. For their research, they showed that a person could control the hand motions of another person in a different building on campus.

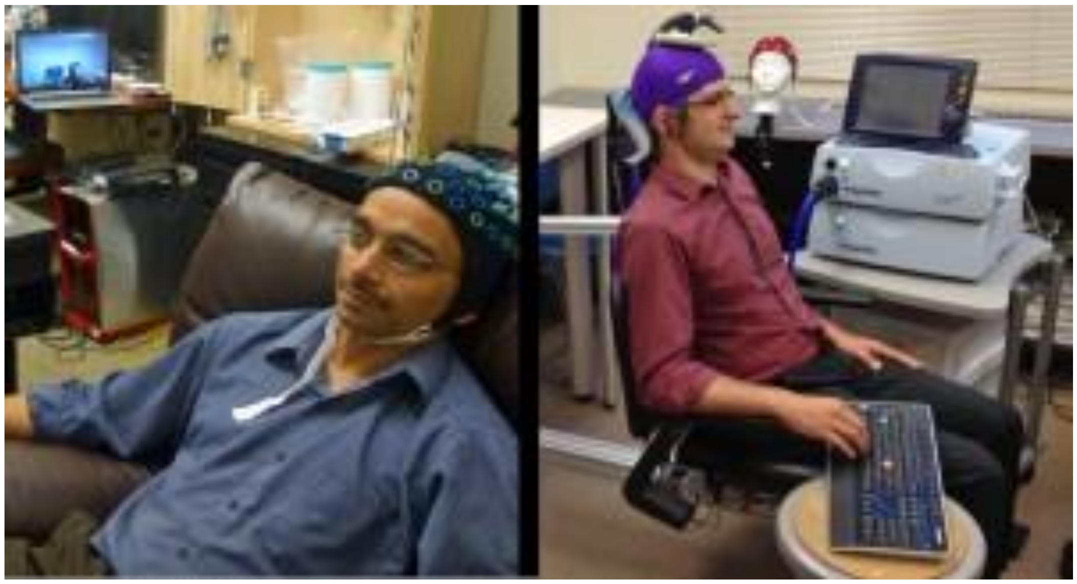


Figure 3: The person on the left is playing a video game via the hand of the person on the right.

EEG Mapping

Surface potential readings of an EEG can be mapped to corresponding activation zones in the brain. Achieving this is important and can lead to a deep understanding of the cognitive and emotional states of the brain using noninvasive electrodes. By correctly mapping EEG readings to voxels (activation zones), one can predict the emotional state of a person. This understanding can be used for many different applications: advanced lie detectors, mind-to-mind communication, and intelligent consumer products to name a few.

Research in EEG mapping is typically done as follows:

- Gather data using EEG headsets. Data should indicate which readings are associated with which cognitive and emotional states. Data can be recorded both in the frequency domain as well as the temporal domain. Finally, interpret the data. This forms the experimental portion of the research.
- Create a model that maps surface potentials to voxels. Associate each voxel with a cognitive or emotional state (f-labels). This forms the theoretical portion of the research.

A very basic mathematical formulation is as follows:

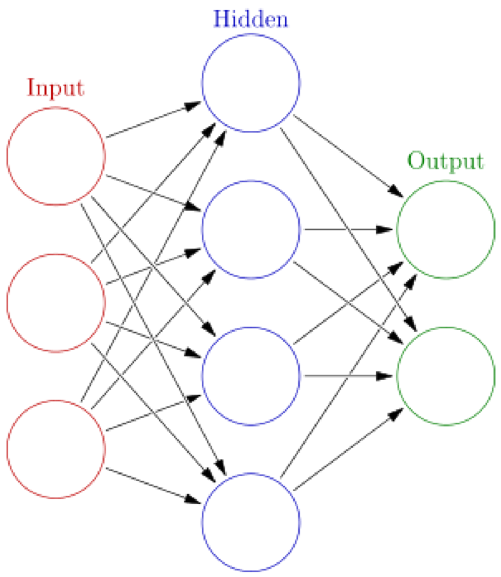
$$V = AX \text{ or } X = A^{-1}V$$

V and X are vectors representing the surface potentials and activation zones, respectively, and A is a tensor mapping X onto V. A is constructed by calculating the inverse distance from a recording node on the surface to an activation site in the brain. These activation sites are typically called **f-labels,** and they are functions associated with various stimuli.

Thus, the goal of this project is to predict X given A and V. Previous attempts to calculate X by inverting A have proven to be computationally difficult and unstable. Instead I will train an artificial neural network to predict X.

Artificial Neural Networks

Artificial neural networks are a set of machine learning tools used to solve a wide variety of problems in the same way a human brain would. The typical architecture of a neural network is based on a primitive model of the brain and how it learns. A brain solves problems by receiving stimulation from its input nodes, which in turn receive their signals from the senses, and traverses those signals through a network of neural units deeper and deeper in the brain's architecture. In the case of artificial neural networks, each node has a summation function that adds all the values of its incoming inputs. A threshold function then checks the resulting sum and determines whether the receiving node will activate. The nodes within this kind of an architecture are also referred to as multilayer perceptrons.



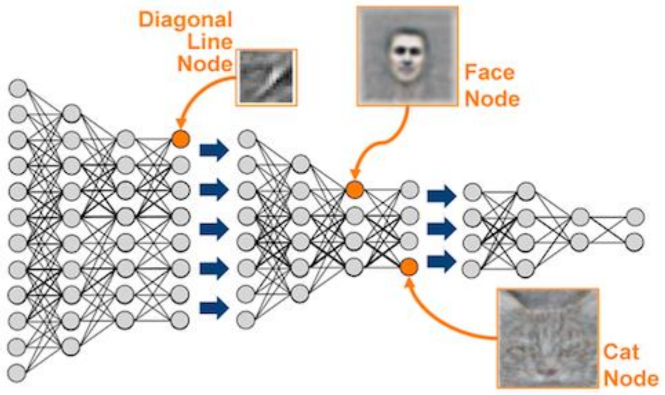


Figure 4: Basic artificial neural network (LEFT). Complex neural network used to detect objects in images (TOP)

This form of learning has been coined "deep learning". "Deep" in the context of machine learning means that multiple "hidden" layers reside between the input layer and the output layer of nodes. The more hidden layers, the higher the order (nonlinearly) of the threshold function that determines whether or not a node becomes activated. This means that features of the system do not have to be linearly separable, and therefore a neural network with multiple hidden nodes can make quite complex distinctions between data points. Consider the following image:

One would agree that this image is formed of various complex objects: birds, buildings, boats, an island, and clouds. However, these objects are all formed of more fundamental shapes such as squares, triangles, circles, and other polygons. These polygons in turn are formed of even more fundamental objects: lines and edges. An artificial neural network that is formed of simply an input layer and an output layer cannot abstract beyond linearly separable data: say for example a binary classification indicating whether a black dot exists in the middle of the image or not. Indicating whether a bird exists in the image, for example, requires a more nonlinear architecture that will be able to isolate the lines and edges that compose a bird from the lines and edges that compose an island. More concretely, convolutional neural networks have been widely adopted for object recognition in images.

Deep learning can be performed in two types of settings: supervised and unsupervised. In the case of supervised learning, the values of the output nodes of the neural network are compared to expected values that input values are labeled with. Thus, this becomes a classification problem.

The question is: given a set of features of a system, how can that system be classified in terms of a discrete set of values? For example, given the pixels of an image, what object (within a set of possible objects) is contained within that image? In the case of this project: given spectral data recorded from an EEG device, what is the cognitive or mental state of a human? On the other hand, unsupervised learning seeks to find structure in unlabeled data. An incredibly vast topic of its own, unsupervised learning will not be covered here.

In terms of statistics, artificial neural networks use logistical regression to predict from a discrete set of values in a range [0, n], n being the number of classifications. Nodes from each layer are mapped to the next layer using weights. These weights are the parameters that form the hypothesis function. The goal of neural networks is to minimize the error between the output of the hypothesis function and the expected results given in the training set. Various minimization algorithms are used to fine tune the weights of the regression model, but the most popular one is gradient descent. Gradient descent is used to backpropagate from the output layer and fine tune the weights accordingly. Weights are tuned up if they contributed to the correct answer, and they are tuned down if they contributed to the incorrect answer. After many iterations, the neural network can be considered "trained".

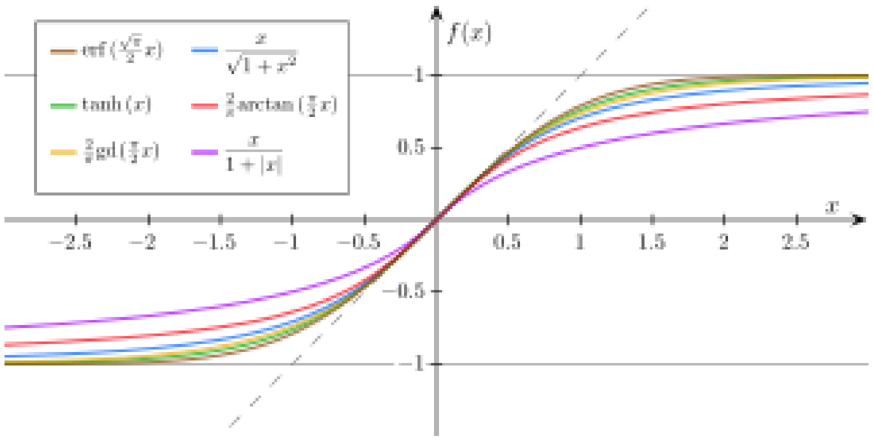


Figure 5: Sigmoid functions used to determine the firing of a node

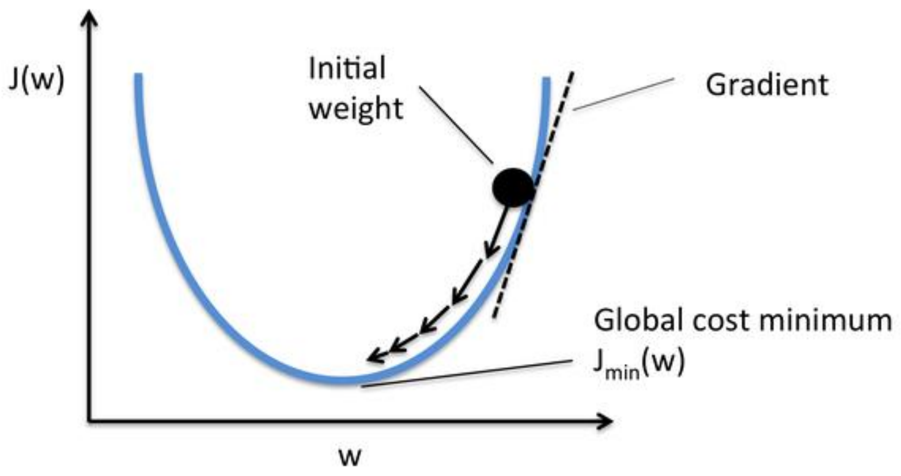


Figure 6: A primitive depiction of gradient descent

Finally, there are some important things to consider when training neural networks. How will the accuracy of the trained network be determined? How biased is the trained network to future data? What will happen if hidden layers are added or removed? And is the size of my dataset sufficient? These questions do not have definite answers. A simple technique is using a portion of the dataset to test the trained network. This is called cross-validation, and it is used in this project as well as many other important applications. Furthermore, the problem of figuring out how many hidden layers to add is a complicated one. If there is too much nonlinearity, as can be the case with too many hidden layers, then the problem of overfitting becomes immediately apparent. If the model cannot be generalized to future dataset, then it becomes impractical. Also, having more data always aids in building a stronger model. Tracking down a sufficiently large dataset for an application is a key task for implementing versatile artificial neural networks.

Solution to the Problem

Professor Tankelevich provided me with a dataset that shows EEG readings in the frequency domain. Each data point is labeled with either 0 or 1 indicating that the subject was presented with a dull grey image or a bright flower image. The dataset has a structure that is typical of any dataset used to train an artificial neural network. The first row indicates the column names, where the last column is the classification of the data point.

I decided to develop a graphical software that can be used to do the following:

- Load in a data set and view it
- Build a neural network
- Train the neural network using a selected framework
- Visualize the features of the dataset
- Test the trained network

An important feature of the software is that it is general to any dataset structured in the way mentioned above. It is also important that the backend algorithm used to train and test the neural networks is general as well. That is why I decided to provide the user with multiple software frameworks (namely Google's TensorFlow and SciPy's SKLearn) for building and training an artificial neural network.

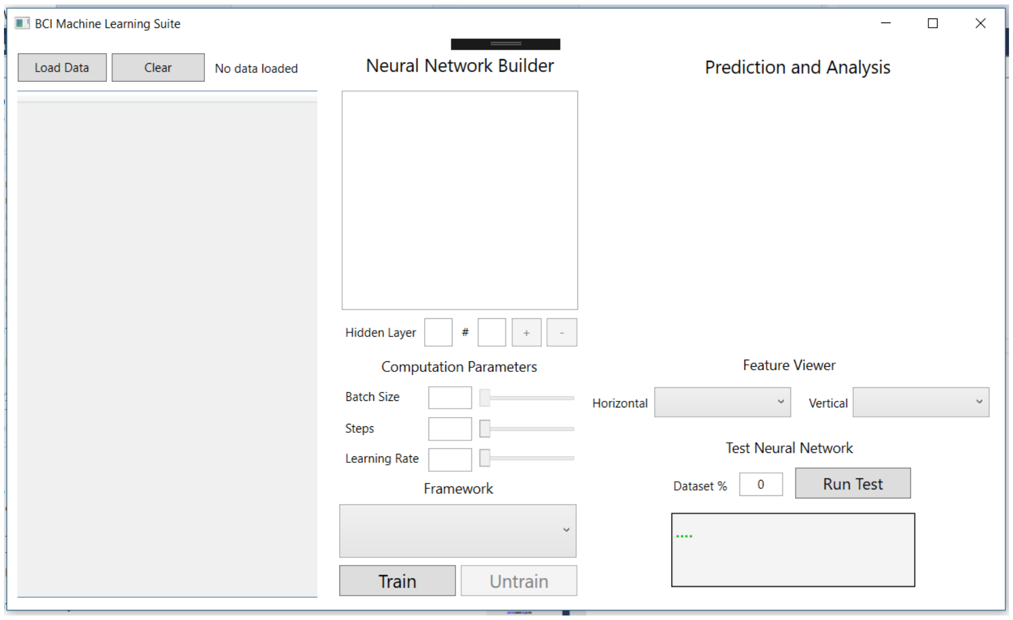
A few additional features are also worth mentioning. When the user loads the dataset, the software detects the number of features and classifications and constructs a hidden-layer-free neural network automatically. The user can also select the batch size, number of steps, and learning rate of the backend algorithm before training the network. Finally, the network can be untrained and trained again. This can be highly useful when fine-tuning the parameters to predict accurate results.

Specifics and Results

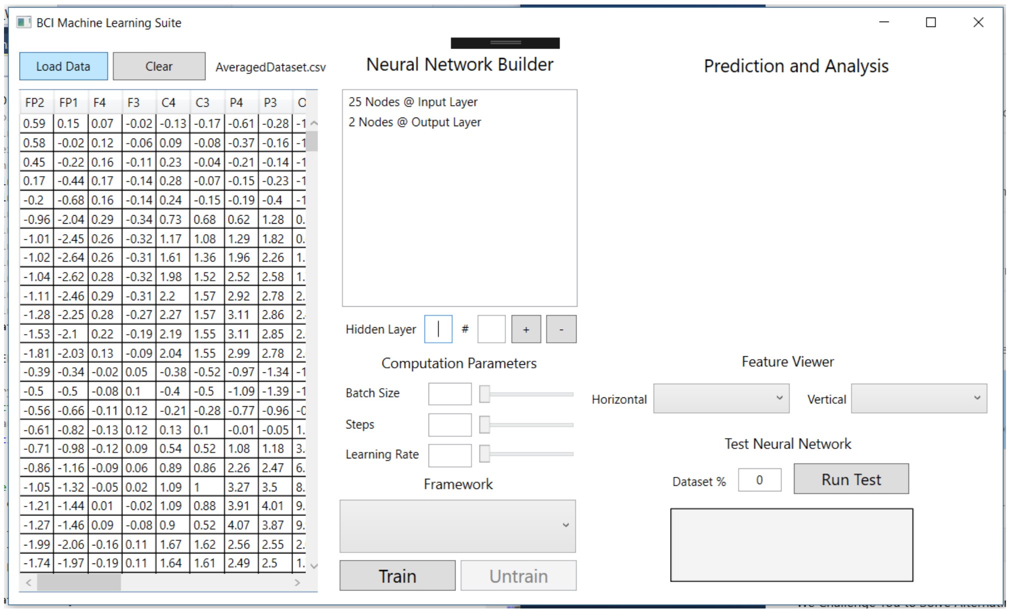
The frontend software was written in C# using the Windows Presentation Foundation graphical user interface framework. The backend was written in Python using Google's TensorFlow machine learning framework. The front-end application interacts with the Python scripts to produce training and testing results. Additionally, the Bokeh visualization library was used in Python to plot features against each other and generate HTML webpages of these plots. The HTML is then generated in an embedded browser on the front-end application.

A sequence of steps in which this application can be used goes as follows:

1. Open the application



2. Load the dataset



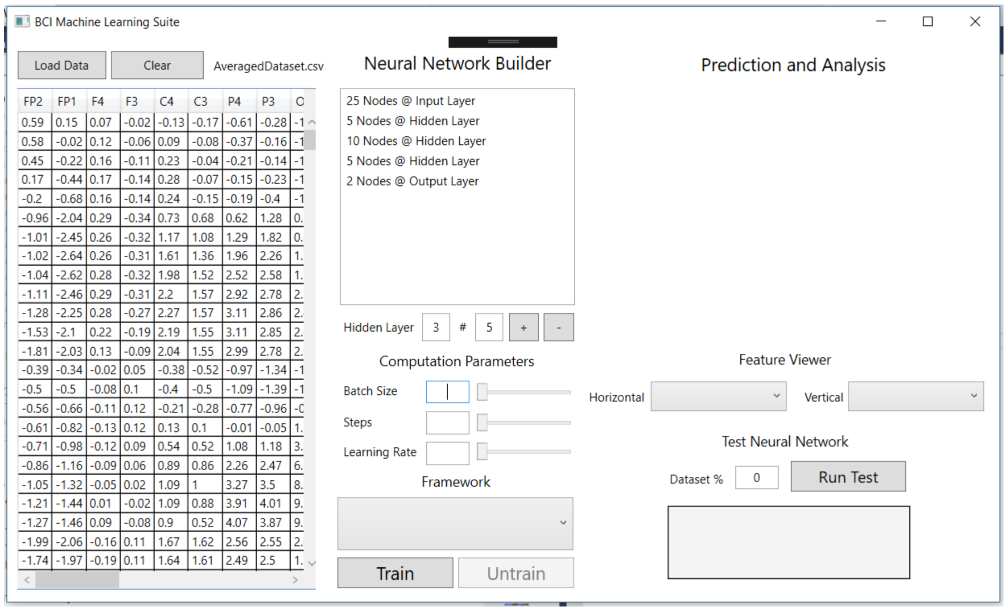
AveragedDataset.csv

FP2	FP1	F4	F3	C4	C3	P4	P3	O
0.59	0.15	0.07	-0.02	-0.13	-0.17	-0.61	-0.28	-1
0.58	-0.02	0.12	-0.06	0.09	-0.08	-0.37	-0.16	-1
0.45	-0.22	0.16	-0.11	0.23	-0.04	-0.21	-0.14	-1
0.17	-0.44	0.17	-0.14	0.28	-0.07	-0.15	-0.23	-1
-0.2	-0.68	0.16	-0.14	0.24	-0.15	-0.19	-0.4	-1
-0.96	-2.04	0.29	-0.34	0.73	0.68	0.62	1.28	0.
-1.01	-2.45	0.26	-0.32	1.17	1.08	1.29	1.82	0.
-1.02	-2.64	0.26	-0.31	1.61	1.36	1.96	2.26	1.
-1.04	-2.62	0.28	-0.32	1.98	1.52	2.52	2.58	1.
-1.11	-2.46	0.29	-0.31	2.2	1.57	2.92	2.78	2.
-1.28	-2.25	0.28	-0.27	2.27	1.57	3.11	2.86	2.
-1.53	-2.1	0.22	-0.19	2.19	1.55	3.11	2.85	2.
-1.81	-2.03	0.13	-0.09	2.04	1.55	2.99	2.78	2.
-0.39	-0.34	-0.02	0.05	-0.38	-0.52	-0.97	-1.34	-1
-0.5	-0.5	-0.08	0.1	-0.4	-0.5	-1.09	-1.39	-1
-0.56	-0.66	-0.11	0.12	-0.21	-0.28	-0.77	-0.96	-0
-0.61	-0.82	-0.13	0.12	0.13	0.1	-0.01	-0.05	1.
-0.71	-0.98	-0.12	0.09	0.54	0.52	1.08	1.18	3.
-0.86	-1.16	-0.09	0.06	0.89	0.86	2.26	2.47	6.
-1.05	-1.32	-0.05	0.02	1.09	1	3.27	3.5	8.
-1.21	-1.44	0.01	-0.02	1.09	0.88	3.91	4.01	9.
-1.27	-1.46	0.09	-0.08	0.9	0.52	4.07	3.87	9.
-1.99	-2.06	-0.16	0.11	1.67	1.62	2.56	2.55	2.
-1.74	-1.97	-0.19	0.11	1.64	1.61	2.49	2.5	1.

25 Nodes @ Input Layer
2 Nodes @ Output Layer

Dataset % 0

3. Build the neural network



AveragedDataset.csv

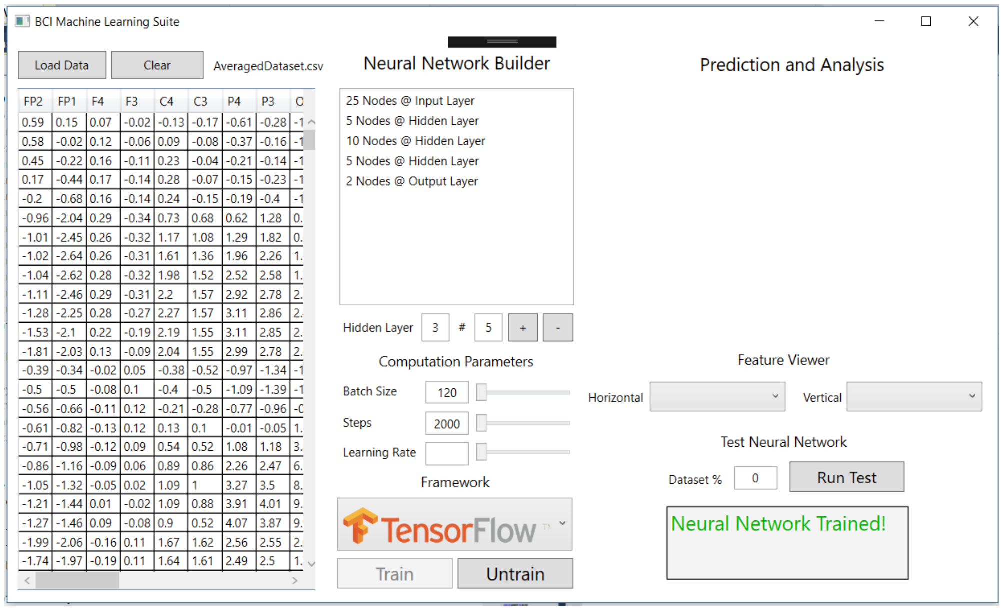
FP2	FP1	F4	F3	C4	C3	P4	P3	O
0.59	0.15	0.07	-0.02	-0.13	-0.17	-0.61	-0.28	-1
0.58	-0.02	0.12	-0.06	0.09	-0.08	-0.37	-0.16	-1
0.45	-0.22	0.16	-0.11	0.23	-0.04	-0.21	-0.14	-1
0.17	-0.44	0.17	-0.14	0.28	-0.07	-0.15	-0.23	-1
-0.2	-0.68	0.16	-0.14	0.24	-0.15	-0.19	-0.4	-1
-0.96	-2.04	0.29	-0.34	0.73	0.68	0.62	1.28	0.
-1.01	-2.45	0.26	-0.32	1.17	1.08	1.29	1.82	0.
-1.02	-2.64	0.26	-0.31	1.61	1.36	1.96	2.26	1.
-1.04	-2.62	0.28	-0.32	1.98	1.52	2.52	2.58	1.
-1.11	-2.46	0.29	-0.31	2.2	1.57	2.92	2.78	2.
-1.28	-2.25	0.28	-0.27	2.27	1.57	3.11	2.86	2.
-1.53	-2.1	0.22	-0.19	2.19	1.55	3.11	2.85	2.
-1.81	-2.03	0.13	-0.09	2.04	1.55	2.99	2.78	2.
-0.39	-0.34	-0.02	0.05	-0.38	-0.52	-0.97	-1.34	-1
-0.5	-0.5	-0.08	0.1	-0.4	-0.5	-1.09	-1.39	-1
-0.56	-0.66	-0.11	0.12	-0.21	-0.28	-0.77	-0.96	-0
-0.61	-0.82	-0.13	0.12	0.13	0.1	-0.01	-0.05	1.
-0.71	-0.98	-0.12	0.09	0.54	0.52	1.08	1.18	3.
-0.86	-1.16	-0.09	0.06	0.89	0.86	2.26	2.47	6.
-1.05	-1.32	-0.05	0.02	1.09	1	3.27	3.5	8.
-1.21	-1.44	0.01	-0.02	1.09	0.88	3.91	4.01	9.
-1.27	-1.46	0.09	-0.08	0.9	0.52	4.07	3.87	9.
-1.99	-2.06	-0.16	0.11	1.67	1.62	2.56	2.55	2.
-1.74	-1.97	-0.19	0.11	1.64	1.61	2.49	2.5	1.

25 Nodes @ Input Layer
5 Nodes @ Hidden Layer
10 Nodes @ Hidden Layer
5 Nodes @ Hidden Layer
2 Nodes @ Output Layer

Hidden Layer 3 # 5

Dataset % 0

4. Select running parameters and a machine learning framework. Then train the neural network


BCI Machine Learning Suite

AveragedDataset.csv

FP2	FP1	F4	F3	C4	C3	P4	P3	O
0.59	0.15	0.07	-0.02	-0.13	-0.17	-0.61	-0.28	-1
0.58	-0.02	0.12	-0.06	0.09	-0.08	-0.37	-0.16	-1
0.45	-0.22	0.16	-0.11	0.23	-0.04	-0.21	-0.14	-1
0.17	-0.44	0.17	-0.14	0.28	-0.07	-0.15	-0.23	-1
-0.2	-0.68	0.16	-0.14	0.24	-0.15	-0.19	-0.4	-1
-0.96	-2.04	0.29	-0.34	0.73	0.68	0.62	1.28	0.
-1.01	-2.45	0.26	-0.32	1.17	1.08	1.29	1.82	0.
-1.02	-2.64	0.26	-0.31	1.61	1.36	1.96	2.26	1.
-1.04	-2.62	0.28	-0.32	1.98	1.52	2.52	2.58	1.
-1.11	-2.46	0.29	-0.31	2.2	1.57	2.92	2.78	2.
-1.28	-2.25	0.28	-0.27	2.27	1.57	3.11	2.86	2.
-1.53	-2.1	0.22	-0.19	2.19	1.55	3.11	2.85	2.
-1.81	-2.03	0.13	-0.09	2.04	1.55	2.99	2.78	2.
-0.39	-0.34	-0.02	0.05	-0.38	-0.52	-0.97	-1.34	-1
-0.5	-0.5	-0.08	0.1	-0.4	-0.5	-1.09	-1.39	-1
-0.56	-0.66	-0.11	0.12	-0.21	-0.28	-0.77	-0.96	-C
-0.61	-0.82	-0.13	0.12	0.13	0.1	-0.01	-0.05	1.
-0.71	-0.98	-0.12	0.09	0.54	0.52	1.08	1.18	3.
-0.86	-1.16	-0.09	0.06	0.89	0.86	2.26	2.47	6.
-1.05	-1.32	-0.05	0.02	1.09	1	3.27	3.5	8.
-1.21	-1.44	0.01	-0.02	1.09	0.88	3.91	4.01	9.
-1.27	-1.46	0.09	-0.08	0.9	0.52	4.07	3.87	9.
-1.99	-2.06	-0.16	0.11	1.67	1.62	2.56	2.55	2.
-1.74	-1.97	-0.19	0.11	1.64	1.61	2.49	2.5	1.

Neural Network Builder

25 Nodes @ Input Layer
5 Nodes @ Hidden Layer
10 Nodes @ Hidden Layer
5 Nodes @ Hidden Layer
2 Nodes @ Output Layer

Hidden Layer 3 # 5

Computation Parameters

Batch Size 120
Steps 2000
Learning Rate

Framework

Prediction and Analysis

Feature Viewer

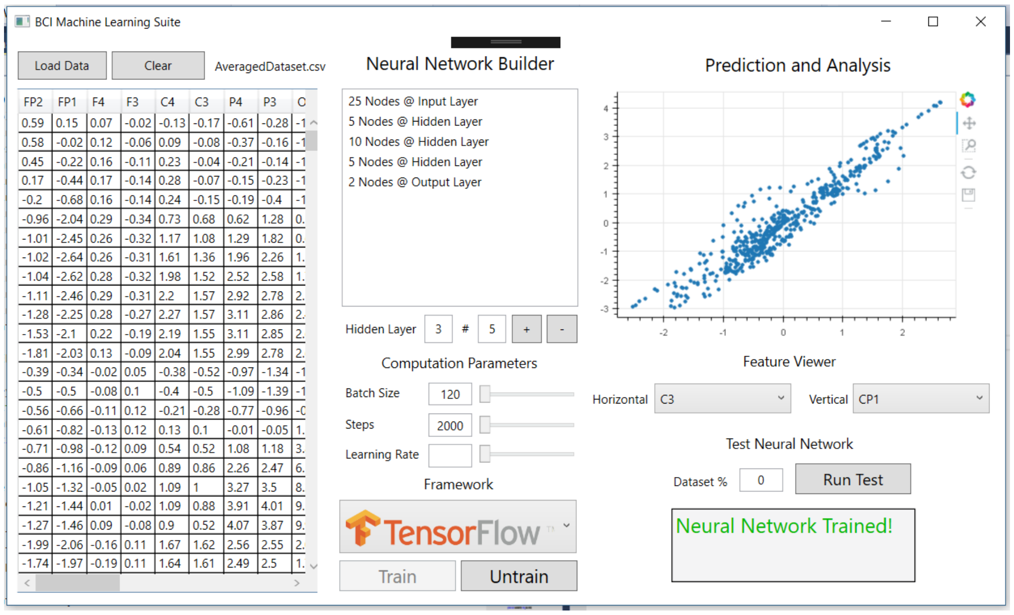
Horizontal
Vertical

Test Neural Network

Dataset % 0

Neural Network Trained!

5. Visualize the features


BCI Machine Learning Suite

AveragedDataset.csv

FP2	FP1	F4	F3	C4	C3	P4	P3	O
0.59	0.15	0.07	-0.02	-0.13	-0.17	-0.61	-0.28	-1
0.58	-0.02	0.12	-0.06	0.09	-0.08	-0.37	-0.16	-1
0.45	-0.22	0.16	-0.11	0.23	-0.04	-0.21	-0.14	-1
0.17	-0.44	0.17	-0.14	0.28	-0.07	-0.15	-0.23	-1
-0.2	-0.68	0.16	-0.14	0.24	-0.15	-0.19	-0.4	-1
-0.96	-2.04	0.29	-0.34	0.73	0.68	0.62	1.28	0.
-1.01	-2.45	0.26	-0.32	1.17	1.08	1.29	1.82	0.
-1.02	-2.64	0.26	-0.31	1.61	1.36	1.96	2.26	1.
-1.04	-2.62	0.28	-0.32	1.98	1.52	2.52	2.58	1.
-1.11	-2.46	0.29	-0.31	2.2	1.57	2.92	2.78	2.
-1.28	-2.25	0.28	-0.27	2.27	1.57	3.11	2.86	2.
-1.53	-2.1	0.22	-0.19	2.19	1.55	3.11	2.85	2.
-1.81	-2.03	0.13	-0.09	2.04	1.55	2.99	2.78	2.
-0.39	-0.34	-0.02	0.05	-0.38	-0.52	-0.97	-1.34	-1
-0.5	-0.5	-0.08	0.1	-0.4	-0.5	-1.09	-1.39	-1
-0.56	-0.66	-0.11	0.12	-0.21	-0.28	-0.77	-0.96	-C
-0.61	-0.82	-0.13	0.12	0.13	0.1	-0.01	-0.05	1.
-0.71	-0.98	-0.12	0.09	0.54	0.52	1.08	1.18	3.
-0.86	-1.16	-0.09	0.06	0.89	0.86	2.26	2.47	6.
-1.05	-1.32	-0.05	0.02	1.09	1	3.27	3.5	8.
-1.21	-1.44	0.01	-0.02	1.09	0.88	3.91	4.01	9.
-1.27	-1.46	0.09	-0.08	0.9	0.52	4.07	3.87	9.
-1.99	-2.06	-0.16	0.11	1.67	1.62	2.56	2.55	2.
-1.74	-1.97	-0.19	0.11	1.64	1.61	2.49	2.5	1.

Neural Network Builder

25 Nodes @ Input Layer
5 Nodes @ Hidden Layer
10 Nodes @ Hidden Layer
5 Nodes @ Hidden Layer
2 Nodes @ Output Layer

Hidden Layer 3 # 5

Computation Parameters

Batch Size 120
Steps 2000
Learning Rate

Framework

Prediction and Analysis

Feature Viewer

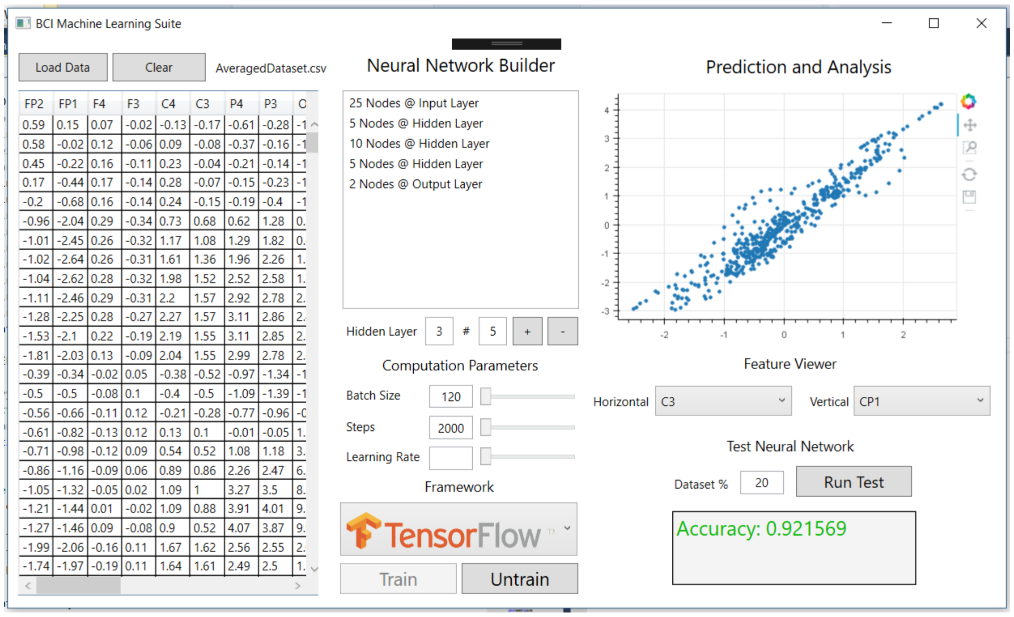
Horizontal C3
Vertical CP1

Test Neural Network

Dataset % 0

Neural Network Trained!

6. Test the accuracy



AveragedDataset.csv

FP2	FP1	F4	F3	C4	C3	P4	P3	O
0.59	0.15	0.07	-0.02	-0.13	-0.17	-0.61	-0.28	-1
0.58	-0.02	0.12	-0.06	0.09	-0.08	-0.37	-0.16	-1
0.45	-0.22	0.16	-0.11	0.23	-0.04	-0.21	-0.14	-1
0.17	-0.44	0.17	-0.14	0.28	-0.07	-0.15	-0.23	-1
-0.2	-0.68	0.16	-0.14	0.24	-0.15	-0.19	-0.4	-1
-0.96	-2.04	0.29	-0.34	0.73	0.68	0.62	1.28	0.
-1.01	-2.45	0.26	-0.32	1.17	1.08	1.29	1.82	0.
-1.02	-2.64	0.26	-0.31	1.61	1.36	1.96	2.26	1.
-1.04	-2.62	0.28	-0.32	1.98	1.52	2.52	2.58	1.
-1.11	-2.46	0.29	-0.31	2.2	1.57	2.92	2.78	2.
-1.28	-2.25	0.28	-0.27	2.27	1.57	3.11	2.86	2.
-1.53	-2.1	0.22	-0.19	2.19	1.55	3.11	2.85	2.
-1.81	-2.03	0.13	-0.09	2.04	1.55	2.99	2.78	2.
-0.39	-0.34	-0.02	0.05	-0.38	-0.52	-0.97	-1.34	-1
-0.5	-0.5	-0.08	0.1	-0.4	-0.5	-1.09	-1.39	-1
-0.56	-0.66	-0.11	0.12	-0.21	-0.28	-0.77	-0.96	-0
-0.61	-0.82	-0.13	0.12	0.13	0.1	-0.01	-0.05	1.
-0.71	-0.98	-0.12	0.09	0.54	0.52	1.08	1.18	3.
-0.86	-1.16	-0.09	0.06	0.89	0.86	2.26	2.47	6.
-1.05	-1.32	-0.05	0.02	1.09	1	3.27	3.5	8.
-1.21	-1.44	0.01	-0.02	1.09	0.88	3.91	4.01	9.
-1.27	-1.46	0.09	-0.08	0.9	0.52	4.07	3.87	9.
-1.99	-2.06	-0.16	0.11	1.67	1.62	2.56	2.55	2.
-1.74	-1.97	-0.19	0.11	1.64	1.61	2.49	2.5	1.

25 Nodes @ Input Layer
5 Nodes @ Hidden Layer
10 Nodes @ Hidden Layer
5 Nodes @ Hidden Layer
2 Nodes @ Output Layer

Hidden Layer 3 # 5

Batch Size 120
Steps 2000
Learning Rate

Horizontal C3
Vertical CP1

Dataset % 20

Accuracy: 0.921569

Summary and conclusion

The human brain is a complex, multilayered, cosmologically vast organ that performs calculations at an alarming rate. The cerebral cortex, containing around 100 billion cells, serves as the center of high-level mental functionality. As it processes this information (emotions, conceptual thinking, planning, etc.) it produces an electric field with a surface potential. Scientists, for decades, have used electroencephalograms (EEGs) to record electrical impulses around the brain. An EEG recording corresponds to a stimulus associated with cognitive states and emotional responses, and with the advent of deep learning, models can learn to find the mapping of surface potentials to activation zones in the brain. I have attempted to do that here, and I have been highly successful. The neural network always predicts with at least 90% accuracy, and graphical interface makes it easy to build and train using a dataset of choice.

References

[1] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach. Saddle River, NJ: Pearson Education, 2011

[2] LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521, 436-444, 2015

[3] Rao RPN, Stocco A, Bryan M, Sarma D, Youngquist TM, et al. (2014) A Direct Brain-to Brain Interface in Humans. PLoS ONE 9(11)

[4] "TensorFlow Is an Open Source Software Library for Machine Intelligence." *TensorFlow - an Open Source Software Library for Machine Intelligence*. N.p., n.d. Web. 12 Dec. 2016.

[5] "Windows Presentation Foundation." *Windows Presentation Foundation*. N.p., n.d. Web. 12 Dec. 2016.

Appendix

The source code to this project can be found here.

The following languages and frameworks were used to build this application:

- C# for the front-end logic
- Windows Presentation Foundation for the GUI design
- Google's TensorFlow for the machine learning algorithms
- SK Learn for the machine learning algorithms