ITCS 6156 - Machine Learning
Dr. Minwoo Jake Lee

Project Proposal
Brain Computer Interface: Meta Learning for ERN Detection

A Project Report
Submitted by:-

Group 10

Abhilash Mandlekar (801136686)
Anushka Tibrewal (801134231)
Jeet Jivrajani (801134130)
Kruti Raval (801135835)

At

University of North Carolina at Charlotte

Introduction

- Brain-Computer Interface (BCI) is a way to connect the brain to an external device in order to send or receive information directly from it. Currently there are several commercial BCI's that are routinely used by thousands of people, such as deep brain stimulators that alleviate Parkinson's Disease symptoms, cochlear implants that restore hearing and and even retinal implants that restore vision. Using BCI techniques it helps sending thoughts and even emotions to your friends and family wirelessly from anywhere in the world.
- There are many applications of using Brain Computer Interface (BCI)[3]
    1. Access information and data from the Internet on the fly.
    2. Upload your memories.
    3. Download knowledge and possibly skill sets.
    4. Have a cognitive AI assistant to aid you in decision making and task management.
    5. Being able to capture the ideas into words, images, or even videos in real time, without recreating them using software.

Problem Statement

Applying meta learning to BCI data. More specifically, we wish to take epoched signal recordings containing error-related negativity (ERN) and classify them as ERN or no ERN. The main goal is to see if a simple task containing ERN can translate to a more complicated task containing ERN.

Steps and Approaches

1. Getting familiar with the concepts of few shot, one shot ,BCI , EEG, ERN and CNN.
2. Understanding the Siamese and Prototypical network
3. Understanding the EEGNet for the data collection purpose
4. Understand and Observe the real time data obtained from EEG
5. Cross task Generalization (Obstacle Avoidance and Observation)
   We apply meta learning to achieve generalization in complex hierarchical task. Ideally we want to complete unseen task with single demonstration. We present metalearning programming to address the task of generalization of complex cross network task. We show that our model is able to generalize across several matrices and is data efficient and work in the setting of various input states.
6. Fine Tuning the model

Literature Survey

1. Siamese neural Network for One-shot Image Recognition[1]

In this era, Deep Convolutional Neural Network has become the prominent in classification of the Image. The major constraint regarding the classification of the images is that it requires huge amount of the labelled data. The situation doesn't guarantees the tons of labelled data for all scenarios. So, an alternative solution is to be designed such that it requires the few samples of the data and provides the accurate classification. This problem can be solved by using One-Shot Learning.

In One shot classification, there is one single image per class for the training set. The model makes a prediction based on the single image per class. One shot learning can also be addressed by developing domain-specific features that have discriminating properties for the test task. The network takes two images as input and generate the similarity score. The Similarity Score denotes how similar the two input images are with each other.

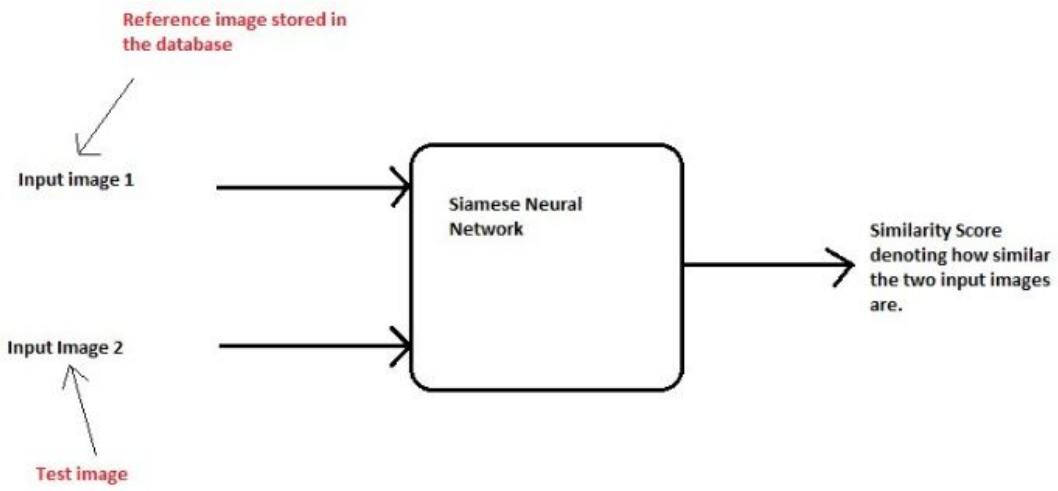


Figure 1. Block diagram of One-shot Classification

A Siamese Neural Network consists of a twin network which takes two distinct or similar input. The standard model is a Siamese convolutional neural network of 'L' layers with Nl units, where h(1,l) represents the hidden vector in layer l for the first twin and h(2,l) denotes the same for second twin.

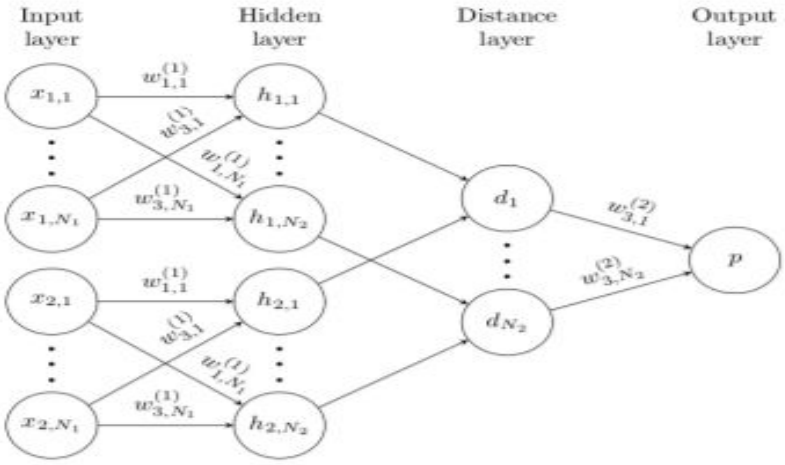


Figure 2. Siamese Network for Binary Classification

The model is a siamese convolutional neural network with L layers each with N1 units, where h(1,l) represents the hidden vector in the layer l of the first twin and h(2,l) denotes the second twin. It uses ReLU units in the first L-2 layers and sigmoidal in the remaining layers. As the ReLU is the non linear function so it can easily back propagate the error and have multiple neurons activated. The disadvantage of the It uses a sequence of convolutional layers each of which uses a single channel with filters of varying size and the fixed stride of 1. The number of convolutional filter is specified as multiples of 16 to optimize performance.(This has been observed that multiples of 16 has increased the performance of the model). Then the network used ReLU activation function to output feature map and it is followed by max pooling with stride of 2. The kth filter map in each layer takes the following form:

$$a^{(k)}{}_{1.m} = max - pool(max(0, W^{(k)}{}_{l-1,l} * h_{1,l-1} + b_l), 2)$$
$$a^{(k)}{}_{1.m} = max - pool(max(0, W^{(k)}{}_{l-1,l} * h_{2,l-1} + b_l), 2)$$

$W_{l-1,l}$ = tensor of feature maps and * signifies the valid convolution.

Siamese ConvNet Model

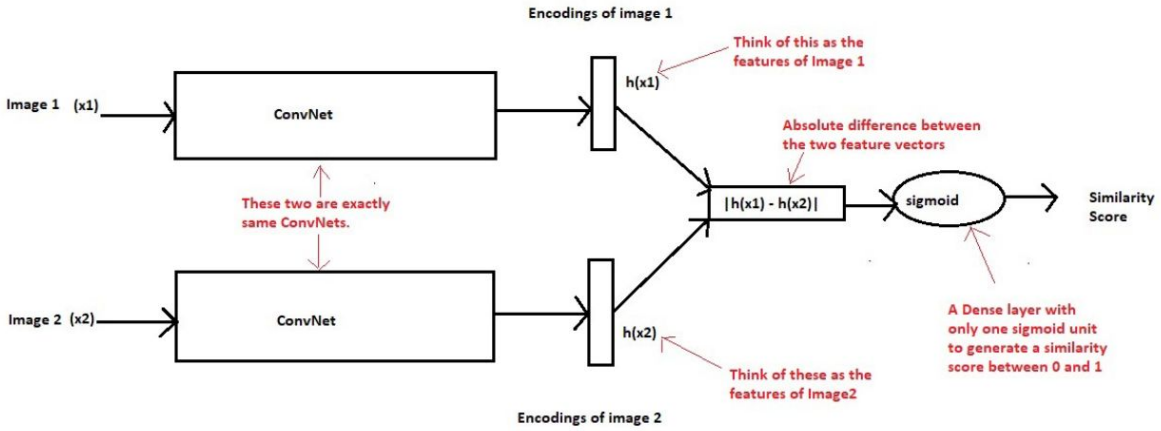


Figure 3. A high level architecture

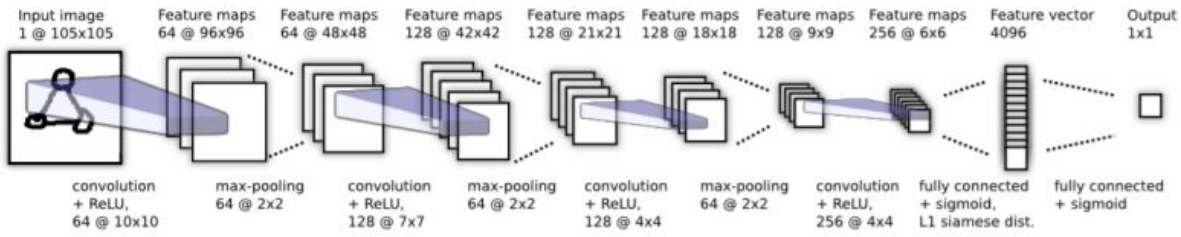


Figure 4. Deep ConvNet Architecture

1. **Loss function-** Regularized Cross Entropy loss is used in the given function. Cross-entropy loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.
2. **Weight initialization-** The model weights are initialized in the convolutional layer from a normal distribution with mean zero and standard deviation 0.01. Biases were initialized with a normal distribution but mean of 0.5 and standard deviation 0.01. But in the fully connected layer the biases are initialized with a normal distribution having mean zero and standard deviation 0.2.
3. **Optimization-** The optimizer that is used is momentum where a mini-batch size of 128 and learning rate nj.
4. **Learning schedule-** The model used different learning rates for each layer, but learning rates are decayed uniformly by 1 percent per epoch. The momentum starts at 0.5 in each layer and linearly increases until it reaches the value uj. The model was trained for 200

epochs but monitored one-shot validation error on a set of 320 one-shot learning tasks generated randomly from validation set.

5. **Hyperparameter optimization-** The model has used beta version of Whetlab, a bayesian optimization framework, to perform hyperparameter selection. The size of the convolutional filters vary from 3*3 to 20*20 while the number of convolutional filters varied from 16 to 256 using multiples of 16. Fully connected layers ranged from 128 to 4096 units.

Results

The one-shot learning performance was evaluated by developing a 20-way within-alphabet classification in which an alphabet is first chosen from among those reserved for the evaluation set, along with twenty characters taken uniformly at random.

Method	Test
Humans	95.5
Hierarchical Bayesian Program Learning	95.2
Affine model	81.8
Hierarchical Deep	65.2
Deep Boltzmann Machine	62.0
Simple Stroke	35.2
1-Nearest Neighbor	21.7
Siamese Neural Net	58.3
Convolutional Siamese Net	92.0

Pros	Cons
Can predict out of training set data	More computationally intensive
Finding similarity or a relationship between two comparable things.	More hyperparameters and fine tuning required

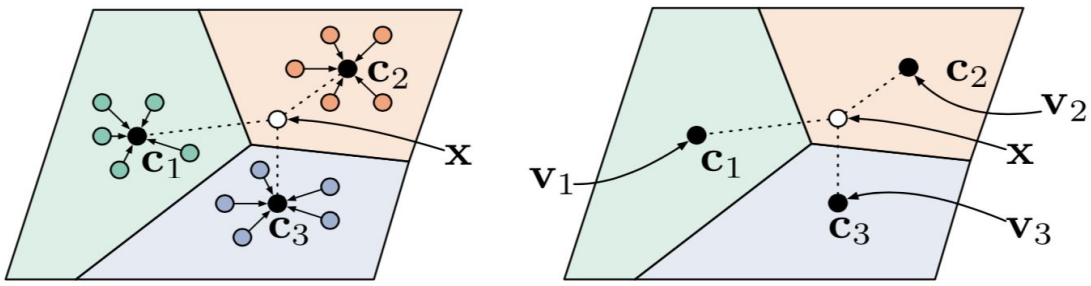
Table : Pros and Cons of Siamese Network

2. Prototypical Networks for Few-shot Learning[5]

Prototypical network is a simpler model and it is kind of a matching network. It is used for the problems of few shot classification. Main features of prototypical network:
1) It is designed for few shot learning.
2) It provides episodic training.
3) Finds a prototype for each class.
4) It uses euclidean distance
5) It uses simpler inductive bias that achieves excellent results.

Similar to matching network it is based on the philosophy that training and test condition should match. Prototypical networks learn a metric space in which classification can be performed by computing distances to prototype representations of each class.

Few Shot Learning
Few shot learning samples few classes from the training set and based on these training classes it takes only few samples for those. For example: Let us say it takes 5 examples per class and it builds a set of those example which is called the support set.
From the below figure we can say that from our training set we sampled 3 class labels and for each of the class labels we sampled 5 examples only then our sample set contains 15 sample and each of them has 3 classes. Then prototypical network learns a function f that embeds those sample sets into space.



(a) Few-shot (b) Zero-shot

Figure 1: Prototypical networks in the few-shot and zero-shot scenarios. **Left**: Few-shot prototypes $\mathbf{c}_k$ are computed as the mean of embedded support examples for each class. **Right**: Zero-shot prototypes $\mathbf{c}_k$ are produced by embedding class meta-data $\mathbf{v}_k$. In either case, embedded query points are classified via a softmax over distances to class prototypes: $p_{\phi}(y = k|\mathbf{x}) \propto \exp(-d(f_{\phi}(\mathbf{x}), \mathbf{c}_k))$.

Each of these examples of the classes tends to cluster and it computes the mean of these points and tells that this mean is the prototype of this class. So this prototype is basically a class representative of the training examples. Hence, when a new example comes up and we want to find it's label we embed the new example X using the same encoding function and find the distance from each of the prototype to see which class it belongs to.

Zero Shot Learning

We use the same approach to tackle zero-shot learning; in zero shot learning each class has metadata giving high level description of the class instead of giving a small number of labeled examples. We therefore learn an embedding of the meta-data into a shared space to serve as the prototype for each class.

In general, we relate prototypical networks to clustering in order to justify the use of class means as prototypes and distances are calculated with a Bregman Divergence , such as squared euclidean distance.

Equations used in the model:

Prototype:

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f_{\boldsymbol{\phi}}(\mathbf{x}_i)$$

Distribution:

$$p_{\boldsymbol{\phi}}(y = k \mid \mathbf{x}) = \frac{\exp(-d(f_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{c}_k))}{\sum_{k'} \exp(-d(f_{\boldsymbol{\phi}}(\mathbf{x}), \mathbf{c}_{k'}))}$$

Loss:

$$J(\Phi) = -log p_{\Phi}(y = K|X)$$

The first equation is used to find the mean of the embedded points of each class which we call the class prototype. And when we try to find the label of the new example X we embed it using the function f and find the euclidean distance from each of these prototypes and do softmax.

Comparison to matching networks

Prototypical Network	**Matching Network**
Uses Euclidean distance	Uses Cosine similarity measure
It is a linear classifier	It is a weighted nearest neighbour classifier
It is a simple model	It is a complex model

3. EEGNet: A Compact Convolutional Network for EEG-based Brain-Computer Interfaces[3]

EEGNet is a compact Convolutional Neural Network for classification and interpretation of EEG-based BCIs. EEGNet achieves good classification performance than the traditional CNN and converges over the less samples as required by Deep and shallow neural networks. The 10-20 system is used to capture the brain signals using 56 channels or the electrodes of EEG device. It collects the data from different parts of the brain such as pre-frontal (Fp), frontal (F), temporal (T), parietal (P), occipital (O) and central (C) .

The BCI has 5 main processing stages which can be listed as follows:

1) Data Collection: Neural data is recorded
2) Signal Processing Stage: Cleaning and processing of recorded data.
3) Feature Extraction Stage: Extracting useful information from the neural data
4) Classification Stage: Some decision is made from the data
5) Feedback Stage: Result of the decision is provided to the user

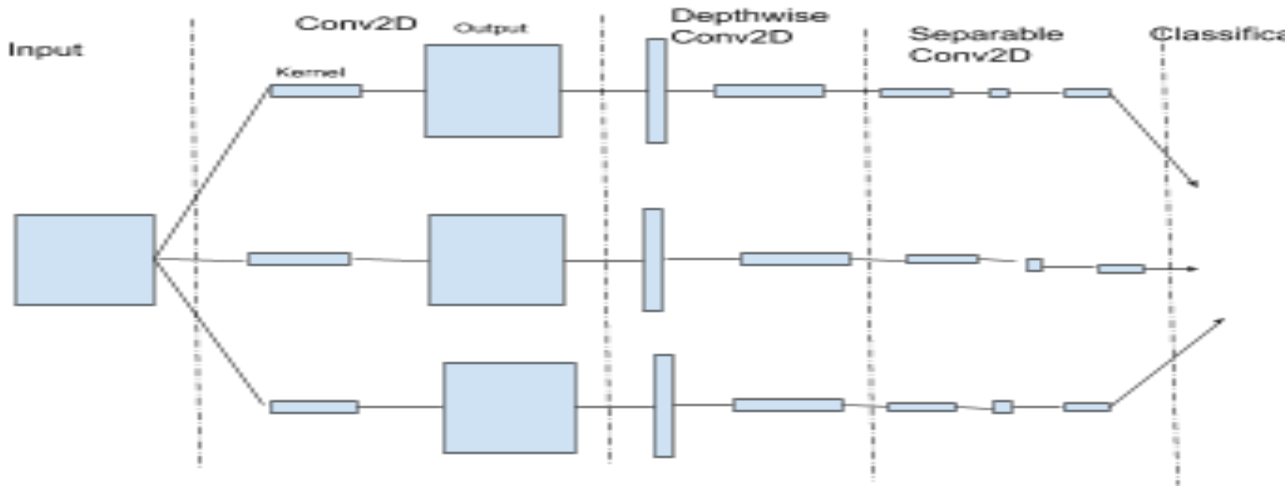


Figure 1. Architecture of EEGNet

Depthwise and Separable convolutions,is used, to construct an EEG-specific network.

The evaluation of the generalizability of EEGNet on EEG datasets is collected from four different BCI paradigms:

1) P300 Event-Related potential (P300)

P300 is the waveform observed over the parietal cortex, and is the large positive deflection of electrical activity. The P300 ERP is one of the strongest neural signatures observable by EEG. The targets are presented to the participants infrequently.
In a particular study, the images of natural scenery had shown to the participants with the infrequent targets of vehicle and a person are shown. When the targets are seen by the participants, they were asked to press a button by their dominant hand. The data is continuously collected by EEG during this time.

2) Error-related negativity (ERN)

If a person makes or perceives an error, an error-related potential can be detected in the EEG due to the person recognizing that error. Depending on the experimental task, different variants of error-related potentials can be measured.
In this, participants were shown flashing letters, arranged in 6 * 6 grid to elicit P300. The goal was to determine whether the feedback of P300 was correct or incorrect.

3) Movement-related cortical potential (MRCP)

It is a combination of both ERP and oscillatory components. It can be produced by the voluntary movements of hands or feet and is observable by EEG.
The data here is participant's finger movements using the left index,left middle, right index or right middle fingers. In this case the data was recorded using the 256 channels of BioSemi Active II system.

4) Sensory motor rhythm (SMR)

SMR is similar to oscillatory component of MRCP. These signals are generally weak and varies between the subjects.
The data used here consists of 4 classes namely the data for right and left hands, feet and tongue.

It is very difficult to interpret and understand the features learned by EEGNet, and hence it is visualized using the three different approaches.
1) analyzing spatial filter outputs, averaged over trials, on the P300 dataset,
2) visualizing the convolutional kernel weights on the SMR dataset and comparing them to the weights learned by FBCSP, and
3) performing single-trial relevance analysis on the MRCP and SMR datasets

Method

Description of idea

The main purpose of the project is to interact with the computer using the real-time data given by the EEG device. The motivation is to design the model that will classify the EEG signals based on the neural activity and will detect the ERN signals from the brain.

Questions that will be answered during the project :

How the EEG signals are classified so that the ERN can be identified
How to generalise the model for different types of BCI paradigms
How does the model perform on cross-task generalization.
How does model perform based on different hyper parameters?
How EEG-based brain-computer interface could provide a new communication and control modality for people with severe motor disabilities.

Expectations of Knowledge gain from the project:

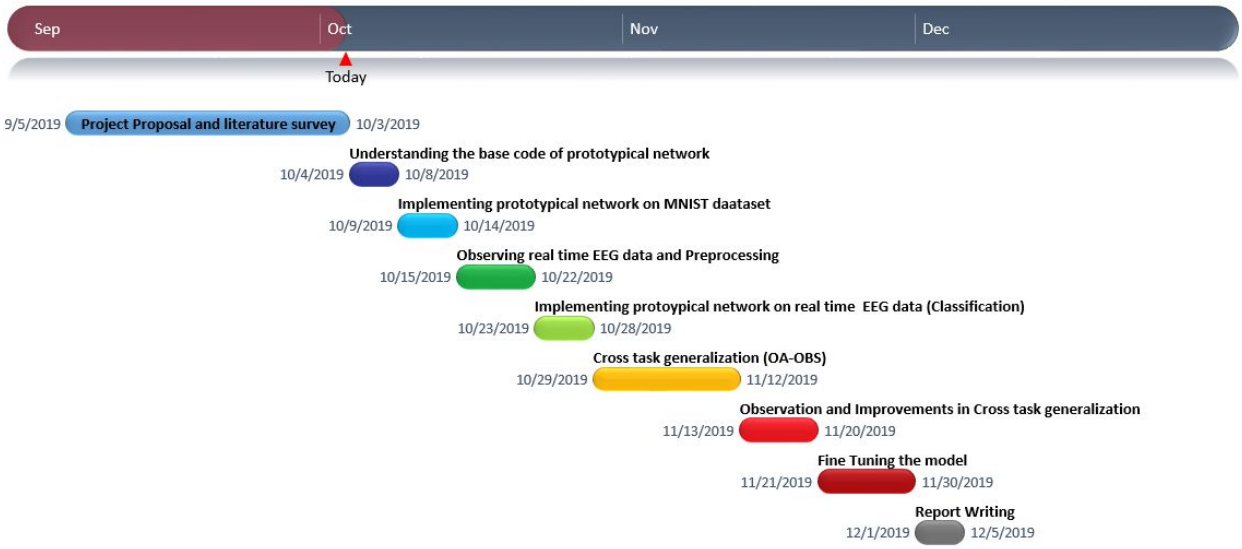
We will learn about the different algorithms (like Siamese Network, EEGNet and Prototypical networks) that can be applied in BCI paradigms. By classifying the EEG signals we can distinguish between the different tasks. We will also learn how to interact with the machine using brain signals.

Novel Approach

1. Fine tuning the neural network. It includes:
- Changing the learning rate
- Separate learning rate for each individual weight
- Depth and Width of Neural network
- Changing the loss function
- Tweaking the filter and stride size
- Using different methods for max pooling
- Analyzing different optimizers like Adam, Rms prop, Ada Delta, and using the best optimizer from it to apply it to our models.

2. Implementing Cross person Generalization(Obstacle avoidance and Observation)

Timeline


Sep
Oct
Nov
Dec
Today
9/5/2019
Project Proposal and literature survey
10/3/2019
Understanding the base code of prototypical network
10/4/2019
10/8/2019
Implementing prototypical network on MNIST daataset
10/9/2019
10/14/2019
Observing real time EEG data and Preprocessing
10/15/2019
10/22/2019
Implementing protoypical network on real time EEG data (Classification)
10/23/2019
10/28/2019
Cross task generalization (OA-OBS)
10/29/2019
11/12/2019
Observation and Improvements in Cross task generalization
11/13/2019
11/20/2019
Fine Tuning the model
11/21/2019
11/30/2019
Report Writing
12/1/2019
12/5/2019
2019

References

1. Lamba, Harshall. “One Shot Learning with Siamese Networks Using Keras.” *Medium*, Towards Data Science, 17 Feb. 2019, https://towardsdatascience.com/one-shot-learning-with-siamese-networks-using-keras-17f34e75bb3d.
2. “10–20 System (EEG).” *Wikipedia*, Wikimedia Foundation, 20 June 2019, https://en.wikipedia.org/wiki/10–20_system_(EEG).
3. Spüler, Martin, and Christian Niethammer. “Error-Related Potentials during Continuous Feedback: Using EEG to Detect Errors of Different Type and Severity.” *Frontiers in Human Neuroscience*, Frontiers Media S.A., 26 Mar. 2015, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4374466/.
4. Svilen. “The Complete Guide to Brain-Computer Interfaces.” *Medium*, Svilen's Portfolio, 16 Mar. 2019, https://medium.com/svilenk/bciguide-246a9ca76fcd.
5. Snell, et al. “Prototypical Networks for Few-Shot Learning.” *ArXiv.org*, 19 June 2017, https://arxiv.org/abs/1703.05175.