Analysis of Brain Regions associated with stimuli using Machine Learning

Tarekul, Junchen,Shateesh, Emmanuil
Department of Computer Science- City College of New York

Abstract

What if we could detect what kind of stimuli the mind is processing at any given time from just EEG data? In this paper, we aim to explore this question by developing classification and visualization techniques driven by human brain signals. In particularly we employ EEG data evoked by visual, audio, language, and non-language stimuli combined with binary machine learning classifiers to learn discriminative brain activities. Our experiment uses 128-channel EEG with active electrodes to record brain activity of several subjects while looking at a combination of flanker stimuli followed by target stimuli and another flanker stimuli.

1.Introduction

The human brain fizzles constantly with activity. Even as we sleep, our brain cells communicate through the passage of electric impulses. EEG 's objective is to measure the electrical activity resulting from neuron interactions. Neuroscientists use special devices called EEG caps to record those Electroencephalogram data. Electrodes are the key to the measurement of these electrical charges. Electrodes are flat, metal disks attached to the scalp. EEG caps have multiple electrodes because different skull areas produce different signals. Our experiment uses the *ANT DukeWaveGuard EEG cap* to record the signals from the different regions of the brain. The cap is composed of 128 channels. There are 4 types of stimuli, picture, sound, spoken word, or written word. We worked with a predefined experiment given to us by *Dave Britton, a researcher at the graduate center, City University of New York.* In his dissertation, he explores the semantic effects of modality, lexicality, and semantic content on attention. The experiment is composed of the following: A trial is made up of 9 events which are, the start of the trial, target concepts, flanker concepts, flanker stim-code-1, target stim-code, flanker stim-code-2, button click, congruency, and accuracy. Each trial shows a subject a flanker stimuli than a target stimuli, and then the same flanker stimuli, to see and understand how the flankers affect the subject’s attention to the targets stimulus semantic content. The instructions for the subject is to click left or right to select the appropriate target on the screen. The time it takes to respond and

accuracy is taken into consideration. There were 32 subjects that each experienced 2 full runs of the experiment. Each trial consists of four stages: focus time, flanker, target, flanker. Each stage is presented for 500 milliseconds, resulting in a total of 2-second trials. Each stimuli time window is 500 ms, however not all stimuli are shown for the same time. Each stimulus is shown for a time within 500 ms depending on the sensory input. For example, the word “DOG” is shown for 50ms, then black screen for 450ms. The sample rate to collect data was 512 samples/sec. This experiment produced a total of 9.8GB of data, which includes all of the EEG readings and the triggers for each session. Using the data collected and the experiment conducted by Dave, we came to the conclusion of if we can do something different with the data. We wanted to examine the subtle details we can detect from EEG signals that a binary classifier uses to differentiate between two different types of stimuli. Can we say what kind of stimuli a subject has seen from just their EEG data? Identifying different brain regions active from the EEG data is important to scientists that deal with the brain. Creating a visual outline of brain regions where there is strong electrical activity allows neuroscientists to study if the results are coherent with the research on what they already know about what happens to the brain when a certain stimulus is presented. If the results of the EEG data is not coherent we can say that something went wrong in the experiment or maybe a particular data is not useful. Furthermore, another application is maybe a way to differentiate how the brain of a different type of people are wired. An autistic person brain might have different signals active in the brain than a person with no autism. This is only possible if we have a consistent and accurate method of identifying signals.

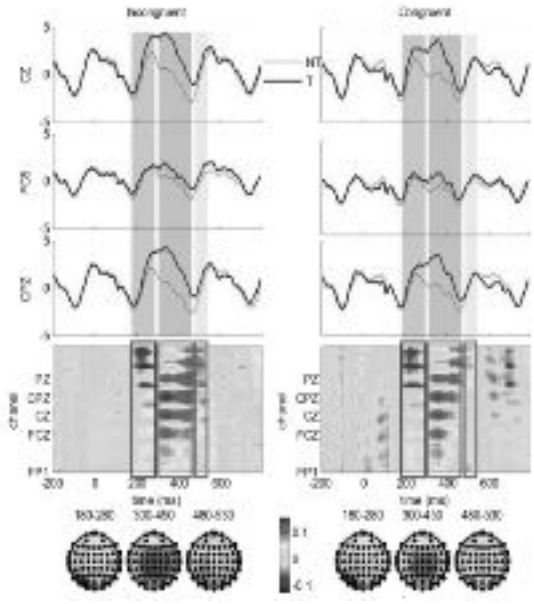
The base problem is can we identify what regions of the brain has strong electrical impulses that distinguish visual trial from an auditory trial. There are many research on visual and auditory regions of the brain. There is sufficient evidence to say that researchers have found that when a person sees a visual stimulus the occipital lobe is very active. The base problem is important because we are able to compare our results with results from previous research.

Each trial is a combination of three stim events, flanker-target-flanker. We define a visual trial as a trial that has only visual combinations of stimulus. A visual trial is a combination of picture and word stimuli. An example would be picture, word, picture which represents a visual combination. An auditory trial is a combination of sound and spoken word. The original experiment had 9 events in each trial. We wanted to work with only the three stim code combinations in the trial. We reduced the trial from 9 events to 3. The 3 events we kept in each trial were the three stim code events that

represented flanker stim-code-1, target stim-code, flanker stim-code2.
There are 4 types of stimuli available (a picture, sound the thing makes, thing spelled visually, and the thing name spoken). Using the 4 types of stimuli, 20 combinations are used with either varying modality or lexicality. We were able to distinguish the 4 into single events related to a visual event or audio event. Picture and word are considered visual stimulus and the other two are considered audio. Can we differentiate the 4 stimuli for language or non-language? Word and spoken word are both languages. Picture and sound are both non-languages. How does the brain react differently to language, and how do classifiers used to differentiate language trials and non-language trials decide from the EEG data which is what?

2.Background

Integrated information from various modalities is essential to the processing of information[1]. The convergence of information from multisensory input enhances behavioral performance(e.g. speed and accuracy) by increasing the activation(firing) of certain neurons together. The impact of multisensory input has been investigated in several studies [2,3] and significantly faster times were found in semantically congruent audio-visual pairings compared with unisensory input. Significant longer times were found for mismatched incongruent audio-visual pairings. Two visual-audio bimodal stimuli (VABS) systems were conducted on a study[4]. The study was comparing congruent and non-congruent VABS based model. The paper aimed to see if semantically congruent stimuli can get the same performance as incongruent stimuli. The acquired raw EEG data were preprocessed for artifact removal through ICA. Data was then filtered through Bandpass of 0.5-40Hz. They plotted the ERP of each channel for each condition (congruent and incongruent) The spatial-temporal distribution of target and distractor were shown for both conditions. The scalp map was shown depicting the average sign value for three specific time intervals. A Positive value represents that target ERPs have a larger amplitude than distractor ERPs. SVM was used to do binary classification of target and non-target. The illustration of Fig 1 demonstrates that higher class discrimination values are lying at channels around FCz, Cz, CPz, and Pz with the time interval 200ms - 500ms.



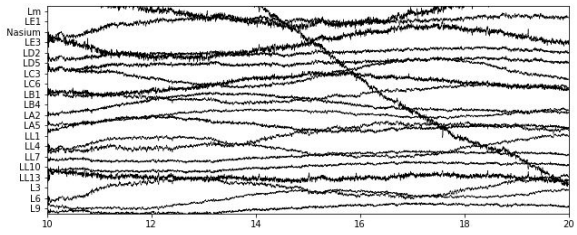
3.Method

We used the MNE python library to read raw data files. The data was in eeglab format and we used the MNE read raw eeglab method to read the data.

raw = mne.io.read_raw_eeglab(set_file, preload=True, verbose = True)

We can see the data has artifacts and noise in it.

raw.plot(n_channels=20,duration=10,start=10)



The y-axis has the channels.

All the event id's each EEG can have:

```
{'TRIALSTART':31,
 'LEFTBUTTON':131,
 'RIGHTBUTTON':132
 'CONGRUENT':133,#
 'INCONGRUENT':134
 'CORRECT':129, #
 'INCORRECT':130,
 #target
 'BABY_T':135,
 'BELL_T':136,
 'BIRD_T':137,
 'BURP_T':138,
 'DOG_T':139,
 'DRUM_T':140,
 'KNOCK_T':141,
 'LAUGH_T':142,
 'PHONE_T':143,
 'TRAIN_T':144,
 'WATER_T':145,
 #flanker
 'BABY_F':155,
 'BELL_F':156,
 'BIRD_F':157,
 'BURP_F':158,
 'DOG_F':159,
 'DRUM_F':160,
 'KNOCK_F':161,
 'LAUGH_F':162,
 'PHONE_F':163,
 'TRAIN_F':164,
 'WATER_F':165,
 }
```

```
{'b-f1-pic': 8,
 'b-f1-snd': 11,
 'b-f1-spk': 5,
 'b-f1-wrd': 2,
 'b-f2-pic': 7,
 'b-f2-snd': 10,
 'b-f2-spk': 4,
 'b-f2-wrd': 1,
 'b-tg-pic': 9,
 'b-tg-snd': 12,
 'b-tg-spk': 6,
 'b-tg-wrd': 3,
 'f-f1-pic': 20,
 'f-f1-snd': 23,
 'f-f1-spk': 17,
 'f-f1-wrd': 14,
 'f-f2-pic': 19,
 'f-f2-snd': 22,
 'f-f2-spk': 16,
 'f-f2-wrd': 13,
 'f-tg-pic': 21,
 'f-tg-snd': 24,
 'f-tg-spk': 18,
 'f-tg-wrd': 15}
```

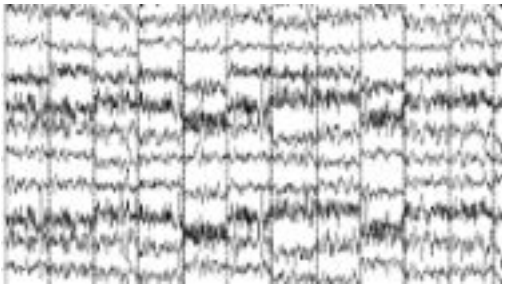
Each trial consisted of 9 events.

The target concept is the name of the target stimulus from the event id list. For example, 144 is *Train_T* The flanker concept is the name of a flanker stimulus from the #flanker event id's. 162 is *laugh_F*. Both concepts tell us the description of the stimulus that is to be presented to the subject. At this point, the subject has not been shown any stimulus. This stage is just for recording information.

The next three events are three stimulus combinations that the subject sees in consecutive order. The flanker stim code is 8 and if we look at the above event id list the key that it refers to we can see that 8 means the flanker is a pic. We know it is a picture of someone laughing because the flanker concept tells us its a laughing stimulus. The target stim code is 3, thus the target stimulus is a word. From the target concept, we know that it is the word 'train'. Lastly, we have flanker stim 7, which is a pic of someone laughing.
In each trial, the same flankers repeat. One is before the target and one is after the target. If we notice carefully we have a combination of a picture, word, and picture which means this trial is a visual one. This will come in handy when we start to filter trials.

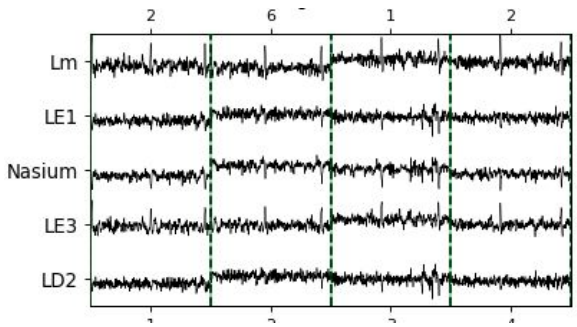
The last three events tell us what button was clicked(left or right) and whether the flanker and target mean the same thing. In this case, we know that flanker is someone laughing and the target is a train, thus they are not semantically similar. The event id 134 if we check above in the events corresponds to incongruent. The last event tells us the accuracy of the click event. In the button click event, the user selects what stimulus he experienced during the target event. If he selects correctly than it is recorded correctly and vice-versa.

For our experiment, we did not need all 9 events. We wanted only 3 events. The three events we were interested in were flanker stim-code-1, target stim-code, and flanker stim-code-2. The data was cleaned using a notch filter of 60 Hz to remove power line noises, data is filtered from 5Hz to 100Hz and ICA is applied to remove the remaining artifacts.
Cleaned data:



```
{'b-f1-pic': 8,
 'b-f1-snd': 11,
 'b-f1-spk': 5,
 'b-f1-wrd': 2,
 'b-f2-pic': 7,
 'b-f2-snd': 10,
 'b-f2-spk': 4,
 'b-f2-wrd': 1,
 'b-tg-pic': 9,
 'b-tg-snd': 12,
 'b-tg-spk': 6,
 'b-tg-wrd': 3,
 'f-f1-pic': 20,
 'f-f1-snd': 23,
 'f-f1-spk': 17,
 'f-f1-wrd': 14,
 'f-f2-pic': 19,
 'f-f2-snd': 22,
 'f-f2-spk': 16,
 'f-f2-wrd': 13,
 'f-tg-pic': 21,
 'f-tg-snd': 24,
 'f-tg-spk': 18,
 'f-tg-wrd': 15}
```

Each trial needed to be 3 events only and all the other events were removed by filtering out any events that were not between 1 and 24. The stim code events were only the ones labeled between 1 and 24.
Each trial would have 3 events that would consist of flanker, target, and flanker.



Above there is the example of the filtered data. The events are only stim events, the others are filtered out. Each event is separated by a line and on the top, we see the event id. On the left, there are 5 channels. The first epoch(event) is a flanker than followed by a target than again a flanker. We also see the start of the next trial. Each epoch is 256 samples of data and 500 ms each.

One additional step was made to combine those three epochs into one and give that one epoch a unique id. We would know that one epoch represented 3 events. The new event id that would be assigned to the new single epoch would depend on the stim code combinations of the three events.

```
stim_combinations = {
    (5,6,4): "AALL",
    (17,18,16) :"AALL",

    (11,8,10):"AALN",
    (23,18,22): AALN",

    (5,12,4) :"AANL",
    (17,24,16): AANL",

    (11,12,10) : "AANN",
    (23,24,22): "AANN",

    (2,6,1):"AVLL",
    (14,18,13): AVLL",

    (8,12,7):"AVNN",
    (20,24,19): AVNN",

    (5,3,4) :"VALL",
    (17,15,16): VALL",

    (11,9,10) : VANN",
    (23,21,22): VANN",

    (2,3,1) :"VVLL",
    (14,15,13): VVLL",

    (8,3,7):"VVLN",
    (20,15,19): VVLN",

    (2,9,1) :"VVNL",
    (14,21,13): VVNL",

    (8,9,7) :"VVNN",
    (20,21,19): VVNN",
```

For example, if the stim code combination of the 3 epochs were a *picture, picture, picture* than the numerical combination would be 20, 21, 19 (from the dictionary pg. 5)
20 = f1-pic (flanker)
21 = tg-pic (target)
19 = f2-pic (flanker)
From the above dictionary and left dictionary, we can see that the combination of (20,21,19) can be combined into one event id called 'VVNN'.
V stands for visual, VN stands for visual non-language, and N stands for non-language. *V-VN-N.* If we combine the first letter V with the last letter N we get VN which means the flankers are visual non-language, which translates to a picture. If we look at the middle two letters we see VN, which means that the target is also a pic. Let's look at another example, AVLL. If we combine the first letter A with the last letter L we get AL. A stands for auditory and L stands for language. AL means auditory languages which translate to a spoken word. We now know that the two flankers for that particular epoch have spoken word. If we look at the middle there is an A-VL-L.

VL stands for visual language which means a written word. The three events associated with AVLL is spoken word, visually written word, and a spoken word. For example, AVLL is either (2,6,1) or (14,18,13) which itself means Spoken word, written word, and spoken word. Epoch objects were then created using the MNE python method called *create_epochs*. The create_epochs method accepts the cleaned epoch data and the dictionary of event_ids associated with the data.
The next step was to classify visual stimulus versus audio stimulus using Logistic Regression and Random Forest Classifier. First, the data was filtered to include only visual trials and audio trials.
Visual trials:

Flanker, Target, Flanker	
Pic, Pic, Pic	VVNN
Word, Word, Word	VVLL
Pic, Word, Pic	VVLN
Word, Pic, Word	VVNL

Audio Trials:

Flanker, Target, Flanker	
Sound, Sound, Sound	AANN
Spkn Wrd, Spkn Wrd, Spkn Wrd	AALL
Sound, Spoken Word, Sound	AALN
Spkn Wrd, Sound, Spkn wrd	AANL

Any epochs that were associated with VVNN, VVLL, VVLN, VVNL, AANN, AALL, AALN, and AANL were kept and the other events were thrown out because they were not strictly visual or auditory epochs. **For example, AVLL stands for a spoken word, word, and spoken word. We can see that the event AVLL is a combination of auditory and visual data. We want to keep those epochs that have a combination of strictly only visual and only audio combinations.*
After the data was filtered, the event id's for all events that started with V was changed to *100*. The event id's for all events that started with A were changed to *200*. The event id is changed to 100 or 200 because the data is fed into a binary classifier and the task is to classify between visual and audio thus we want the machine learning algorithm to understand that all events with 100 are visual and all events with 200 are auditory.

The classification task
Problems:
The combination of all subject data into one file was not possible due to the large of the file exceeding 5GB. We had to split the files into 5 large files so that we don't run into memory errors. CCNY computing node was used to run these large files in Jupyter Notebook environment.

Logistic Regression

```
features = twoEventEpoch.get_data()
target = twoEventEpoch.events[:,-1]
```

We store all the data(filtered) in one variable and the events in one variable. Each event data will correspond to an event of *100* or *200.*

$Shape\ of\ data:\ (10587\ x\ 125\ x\ 256)$

$Shape\ of\ target:\ (10587)$

The data is 3-dimensional data. The first number *10587* refers to the number of epochs or events, 125 refers to the total channels, and 256 is the samples for each channel. For each epoch in the *features variable*, the event_id is in the target variable. The shape of the *target variable* is *10587* because it only holds an array of event_ids.

The pipeline for classifying data

```
clf = make_pipeline( Vectorizer(),
                     StandardScaler(),
                     LogisticRegression( ) )
```

The data is *Vectorized*(python sklearn) into 2D data because the Logistic Regression only accepts 2D data, not 3D data.

$10587\ x\ 125\ x\ 256$ is vectorized to $10587\ x\ 32000$.

Many learning algorithms assume that all features are centered around zero and have the same order of variance. If a feature has a variance greater than other orders of magnitude, it could dominate the objective function and make the estimator unable to learn from other features correctly as expected. The data is fitted and transformed to *Standard Scaler* (Python sklearn library) to avoid that problem.

The last step in the pipeline is to fit the data to Logistic Regression.

StratifiedKFold from Python sklearn library is used to split the data into training and test data. Data is shuffled 10 times and new train and test data is used for classification.

```
cv = StratifiedKFold(n_splits=10,shuffle=True)
scores = []
for train,test in cv.split(features,target):
    X_train = features[train]; y_train = target[train]
    X_test = features[test];   y_test = target[test]

    clf.fit(X_train,y_train)
    scores.append(clf.score(X_test,y_test))
```

The pipeline is 'fitted' on the training data. The machine learning algorithm learns what data relates to visual stimulus and audio stimulus. The machine learning algorithm then predicts the test data to see if it can figure out the correct event that occurred. A good result should have a low standard deviation between the classification scores among the 10 splits, and the mean classification score should be above chance. A good classification score tells us there is a signal in the EEG data that is being used to classify.

After fitting the data we use can get the coefficients of each feature.

```
from mne.decoding import get_coef
coefficients= get_coef(clf,'coef_')
```

$coefficients\ =\ (1\ x\ 32000)$, 32000 because there are 32000 features from vectorized data.

The coefficient data is reshaped to $125\ x\ 256$ by taking 256 coefficients and

putting them on a new row 125 times. Now each channel and time sample has a coefficient. High Positive coefficients and negative coefficients show that the feature or channel associated with the coefficient is important[5]. An estimated coefficient of close to 0 means that the predictor effect is small.

The bar graph was created by taking the mean of each channel(mean of 256 sample points). There is a clear indicator that some channels are highly negative, some are highly positive, and some are near zero.

The coefficient data is plotted using MNE evoked method called *plot_joint* which allows specificity on what time points we want the *plot_joint method* to head maps showing the signal. In the *plot_joint* function, there is an array of time points that are to be used. During those time points, the coefficient data is taken for each channel and plotted on the head based on how high the coefficients are. The higher positive coefficients are dark red and the higher negative coefficients are dark blue. **The results are discussed in the results section**

Random Forest

Random Forest Classifier, is an extension of decision trees that avoids the problem of that decision trees have to deal with which is overfitting. Random Forest is used to compare the results to the Logistic Regression classifier.

Use GridSeachCv to optimize the Random Forest Parameter

```
param_dist = {'max_depth': [2, 3, 4],
              'bootstrap': [True, False],
              'max_features': ['auto', 'sqrt', 'log2', None],
              'criterion': ['gini', 'entropy']}

cv_rf = GridSearchCV(fit_rf, cv = 10,
                     param_grid=param_dist,
                     n_jobs = 3)

cv_rf.fit(X_train, y_train)
print('Best Parameters using grid search: \n',
      cv_rf.best_params_)
```

The estimator parameters used to use these methods are optimized through cross-validated grid searches over a parameter grid.
The optimized parameters are :
'Bootstrap': False,
'Criterion': 'entropy',
'max_depth': 4, '
Max_features': 'auto'

We want to drop 4 channels from the total channels which are 129 because some channels are located outside the head such as near the eyes which are not reliable data.

```
epoch.drop_channels(ch_names=['LE4', 'LE12'])
epoch.drop_channels(ch_names=['Nasium', 'VEOG'])
```

Now we filter the data for visual and audio events just like before we did Logistic Regression.

```
epoch_copy = epoch.copy()
for event in epoch_copy.events:
  if(event[-1] = VVNN or VVLL \
              or VVNL or VVLN):

        event[-1] = 100

  if(event[-1] = AANN or AALL \
              or AANL or AALN):

        event[-1] = 200


binary_epoch =
  epoch_copy[(epoch_copy.events[:,-1]==100)
          or (epoch_copy.events[:,-1]==200)]
```

Using python array manipulation techniques we changed the event ids of all visual to 100 and audio to 200. Lastly, we filtered the data for only events with 100 or 200 so that we can work with the data we want.

Before Filtering:

$$Shape\ of\ data: (16206\ x\ 125\ x\ 257)$$

After filtering:

$$Shape\ of\ data: (10587\ x\ 125\ x\ 257)$$

We store the 3D data (epochs x channels x samples) into the features variable and the event id's associated with each epoch in the target variable.

```
features = binary_epoch.get_data()
target = binary_epoch.events[:,-1]
```

```
shape of features: (10587, 125, 257)
shape of target: (10587,)
```

We do *train_test_split* from *sklearn.model* library. We set the test size to 25%. Train_test_split returns X_train, X_test, y_train, y_test.

We *Vectorize* X_train data to 2D, (epochs x 125 x 257) ➜ (epochs, 32125).

twoD_X_train = (epochs x 32125)

*epochs refers to # of epochs used in the training set.

We fit the training data to Random Forest forest.*fit(twoD_X_train, y_train)*. The trained model can now be used to test.

We need to *Vectorize* the test data.

twoD_X_test = Vectorizer().fit_transform(X_test)

To see how well our model performs we check how well it can predict the right event id for each epoch.

forest.score(twoD_X_test, y_test)

The random forest has an attribute that returns the feature importance for all the features. Since our classifier was fitted to 2D data the feature importance array would return a 1D array. A 1D array because there are 32125 features. The data that was fitted to Random Forest was (epochs x 32125). 32125 is the features. The features importance attribute gives us the importance of the features relative to the others. The feature importances all add up to 1.

However, we want our feature importance to look like a 2D array because we want the shape to be (125 x 257). We want the importance of each channel for each sample. We use python to manipulate our 1D feature importance array by taking 257 elements each time and putting them on a different row. We do this 125 times to get our desired shape of (125 x 257).

Now three things are done with the feature importance data:

Find top 20 channels:

The feature importance for each channel is summed up and the top 20 highest channels are noted down. There are 5 large files of subject data. We classify each file the same way mentioned above and we take the top 20 for each file and note them. The point is to see which channels are consistently important in the classification training.

Plot Evoked data:

For each file, we have feature importance, and thus we use the evoked.plot_joint() method to plot all the feature importance for all channels across time. We use the same method we used in *Linear Regression*.

```
info = binary_epoch.info
evoked = mne.EvokedArray(feat_importnt_allchannels,
                         info, tmin=binary_epoch.times[0])
joint_kwargs = dict(ts_args=dict(time_unit='s'),
                    topomap_args=dict(time_unit='s'))
evoked.plot_joint(times=[0.125,0.175,0.225,0.275],
                  title='patterns', **joint_kwargs)
```

In the *plot_joint* method, we specify at what time we want the head maps to show. We want to show the times when the signal is strong in the brain and the regions where they are strong.

Plot evoked data of top 20 most consistent channels:

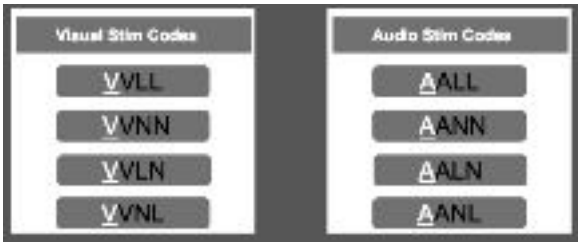
We kept track of consistent channels for each file, now we check which channels appeared in the top 20 at least 3 out of 5 times. We plot a bar graph to show the top 20 consistent channels and use those channels data only to classify and plot evoked data. The point is to see if those important channels are indeed the ones that give us the signal because, in theory, they should give us similar classification scores and similar head maps images. To drop channels we use the drop channel method that MNE python provides.

Language versus non-language classification is done the exact same way as we did with the visual vs audio classification. The filtering for language and non-language data is the only difference. We specify epochs that are language only and epochs that are non-language. For example, VVLL is language and AANN is non-language because VVLL stands for V-VL-V which means visual language. Word, word, word combination. AANN which is a non-language event stands for sound, sound, and sound.

4.Results

Audio vs Visual Random Forest

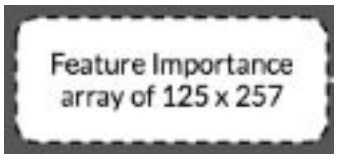
The visual and audio events



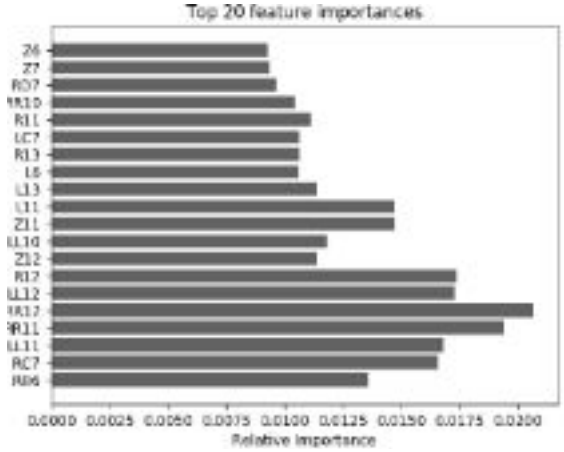
Filter for the above events:

Vectorize the data than fit to algorithm

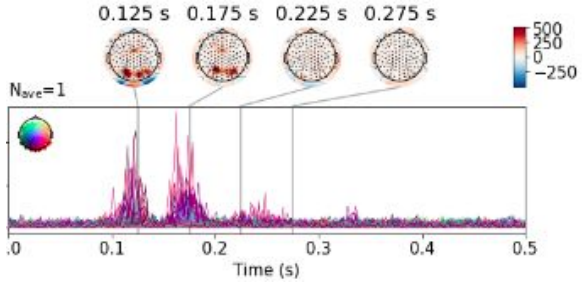
Reshape the feature importance array
From 32125 to (125 x 257)



The first file consisted of 16206 epochs, after filtering for audio and visual the epochs were 10587. The classification score was 69%.

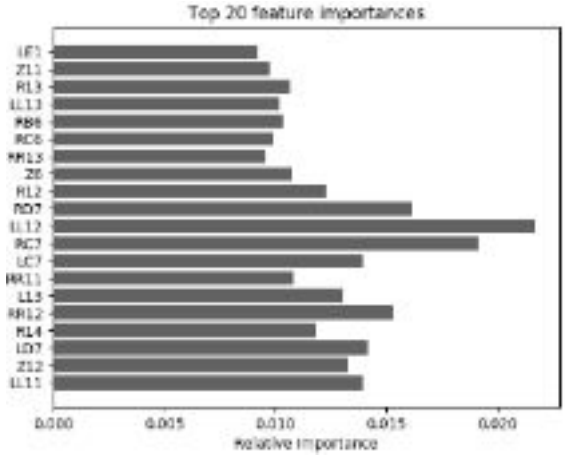


The top 20 channels that are important in the classification algorithm.

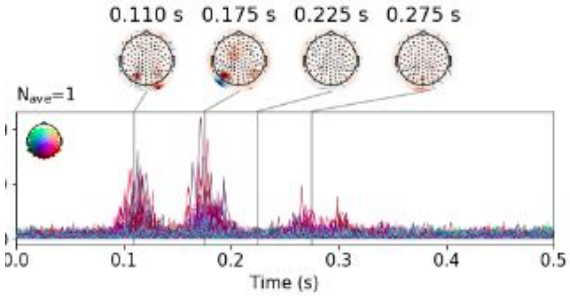


In this plot, we can see the x-axis is from zero seconds to 0.5 seconds. This translates to zero milliseconds to 500 milliseconds. Each epoch or event is 500 ms in our data. We see that in 125 milliseconds and 175 milliseconds we see the most signal on the head. The signal is coming from the back of the head. Can we say that the brain realizes that a visual stimulus was seen between 100 ms and 200 ms? The signal is seen in the back of the head where the visual recognition takes place.

The second file consisted of 16806 epochs before filtering. After filtering for audio and visual the total epochs is 11445. The classification score was 70%.



The top 20 channels that are important in the classification algorithm.

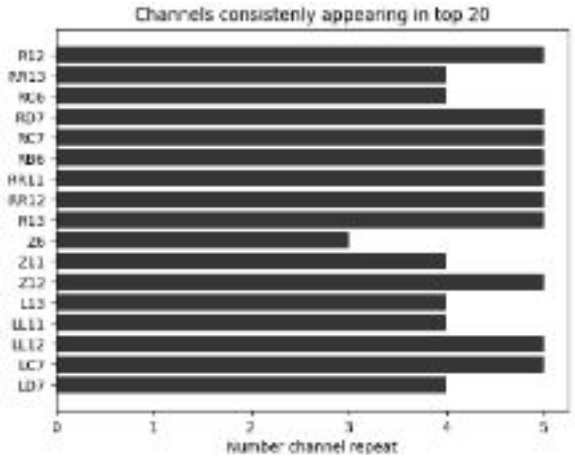


In this plot, the strongest signals are in the back of the head at times 110 milliseconds and 175 milliseconds. The signal is occurring between 100 milliseconds and 200 milliseconds.

The three other files had similar results where the signal was coming from the

back of the head between 100 milliseconds and 200 milliseconds. We kept track of the top 20 channels that consistently important in the classification.

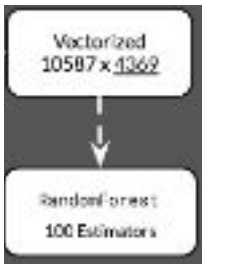
Result: 17 out of the top 20 channels were important across ALL Visual and Audio classifications. The total epochs were *52869* visual/auditory epochs.

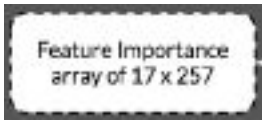


These channels appeared at least 3 out of 5 times in the top 20 feature importance list for each time we did classification. There were 5 large files in total and the total epochs were 52869.

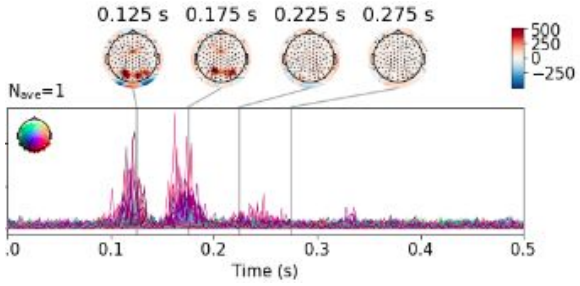
We reduced the original data channels from 125 to 17 than classified using Random Forest. Than plotted the feature importance.

Pipeline for classifying reduced channels



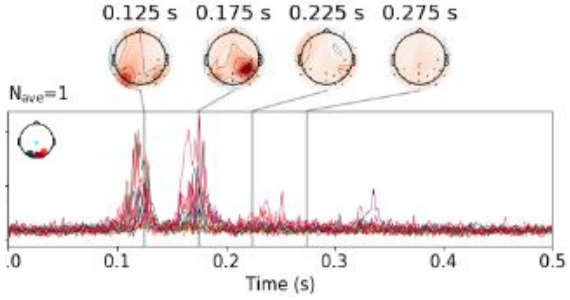


The file data we reduced was the first file. The first file had classification score of 69%. This is how the evoked data plot looked like:



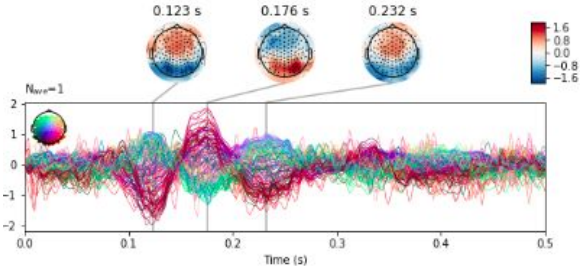
The pipeline for classifying reduced channels was used on the first file to compare how the classification score and the evoked data plot differs. The 17 channels used are listed on the bar graph on the left.

The classification score using reduced channels was 69%, which is exactly the same as before. The plot of the evoked data is below:



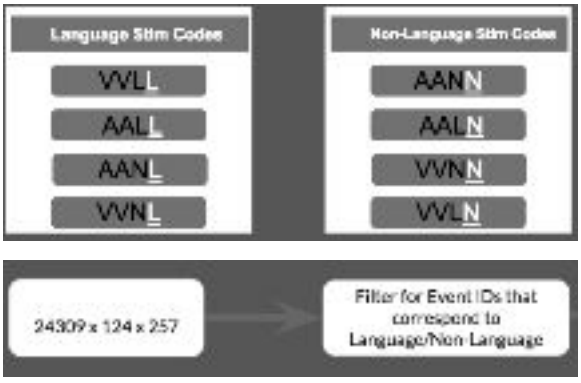
Here we see the time interval between 100 milliseconds and 200 milliseconds gave us the most signal. Also, the signal is primarily coming from the back of the head.

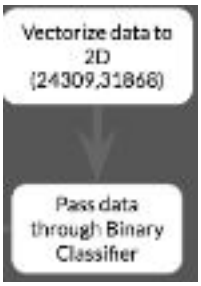
Audio vs Visual Logistic Regression



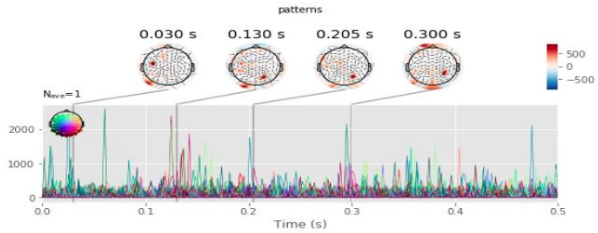
The results from the Logistic Regression had the most signal between 100 milliseconds and 230 milliseconds. We can see due to the high and negative coefficients the head has different colors. The dark blue indicates high negative coefficients and the dark red indicates high positive coefficients. We can clearly see that the darkest areas are the back of the head.

Language vs Non-Language Random Forest



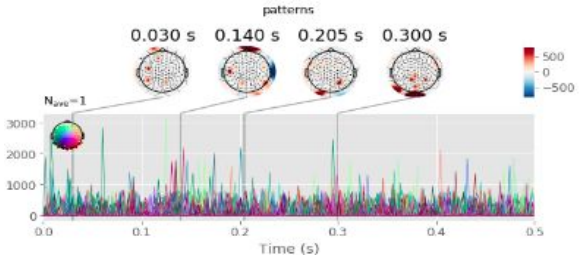


The classification score was 61%.



The plot of the head shows the most electrical impulse signals are coming from the top left of the head around the first 30 milliseconds. Around 130 milliseconds the signal shift to the bottom right and around 205 milliseconds the signal spikes up again around the same region. Around 300 milliseconds the signal arises in the same region as before.

Data file 2 had a classification score of 58%.

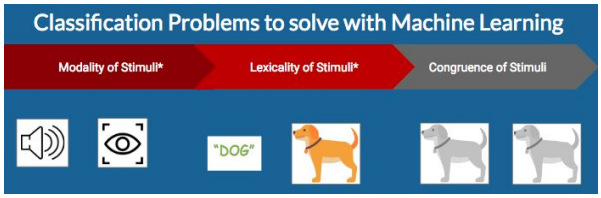


The signal, however, seems similar to the plot in the first file. In the beginning, the signal is in the top left than it shifts to the bottom right later. The Angular gyrus located in the parietal lobe of the brain is responsible for language processing[6]. The angular gyrus is located in the top left of the brain. More information from a neuroscientist is needed to verify this information. Additional this result can be inaccurate due to the low classification score which means it was hard to differentiate the two classes language and non-language.

5.Conclusion

The goal of this research was to eventually reach a point where we can see what parts of the brain are actively firing electrical impulses when a subject is exposed to a certain stimulus. We wanted to learn this from subject's collected EEG data. We also wanted to see if we can detect what modality of stimuli and lexicality of stimuli the subject has seen from their EEG data with great accuracy. Our results with modality(visual vs audio) were consistently giving good classification results and signals were consistently appearing in the same regions and same time periods. Classifying the important channels only showed that those channels were indeed a factor in the decision of the Random Forest algorithm because they returned scores that were the same as the original data with all channels. This meant that the other channels were not giving perhaps not enough useful information. Additionally, the language versus Non-language classification had low classification scores and head maps showed some promise because there was some consistency in the transition of neural firing from one region to the next, however more research is needed on language processing and what is going on with our results.

The future:



The goal is to reach a point where we can say if what the subject saw as flanker and target were semantically congruent. For example, if the flanker is a picture of a baby crying and the target is the sound of a baby crying. Can we detect that the subject saw two things that mean the same thing?

5.References

1. King AJ, Calvert GA. Multisensory integration: perceptual grouping by eye and ear.

2. Laurienti P, Kraft R, Maldjian J, Burdette J, Wallace M. Semantic congruence is a critical factor in multisensory behavioral performance.

3. Laurienti PJ, et al. Cross-modal sensory processing in the anterior cingulate and medial prefrontal cortices.

4. Xingwei An, Yong Cao, Jinwen Wei, Shuang Liu, IEEE Member, Xuejun Jiao, Dong Ming*, Senior IEEE Member. The Effect of Semantic Congruence for Visual-auditory Bimodal Stimuli

5. Tchircoff, Andrew. The Most Complete Chart of Neural Networks, Explained. https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

6.https://support.minitab.com/en-us/minitab-express/1/help-and-how-to/modeling-statistics/regression/how-to/binary-logistic-regression/interpret-the-results/all-statistics-and-graphs/coefficients/

7.http://pandora.cii.wwu.edu/vajda/ling201/test4materials/language_and_the_brain.htm

8.Britton, David. Semantic Attention: Effects of Modality, Lexicality and Semantic Content. 2017.