Grasp-and-Lift EEG Detection

--- Identify Hand Motions From EEG Recordings

Team EEGer

- Xinmeng Li
- Yalin Shi

Background

- Identify hand-motion using EEG is critical to developing BCI devices that would give patients with neurological disabilities the ability to autonomy . Restoring a patient's ability to perform these basic activities of daily life with a brain-computer interface (BCI) prosthetic device would greatly increase their independence and quality of life.
- The detailed signal data is provided by a system with more electrodes and thus higher resolution than the International 10–20 system that we have learned in class.

Abstract

- Analyze the relationship between EEG channels and hand motions
- Attempting to build some models to predict the series of hand motions
    - SVM, Random Forest, Logistic Regression, etc

Introduction

Dataset:

- 12 subjects in total
  - Each subject contains 8 series of trails
- In our project: only use series from subject 1
  - Because of the long running time and limited use of GPU
  - Split the data: 75% (6 series)--training set, 25% (2 series)--testing,
  - Figure 1 show the dimension of the training set.
- Six hand motion events: HandStart, FirstDigitTouch, BothStartLoadPhase, LiftOff, Replace, BothReleased

```
datalist= []
for i in range(1,7):
    file2 = 'train/subj1_series'+str(i)+'_data.csv'
    with open(file2, newline='') as csvfile:
        data = list(csv.reader(csvfile))
    data = data[1:]
    for i in range(0,len(data)):
            datalist.append([data[i][j] for j in range(1,len(data[i]))])
```

```
datalist = np.array(datalist)
print(datalist.shape)
```

```
(1185498, 32)
```

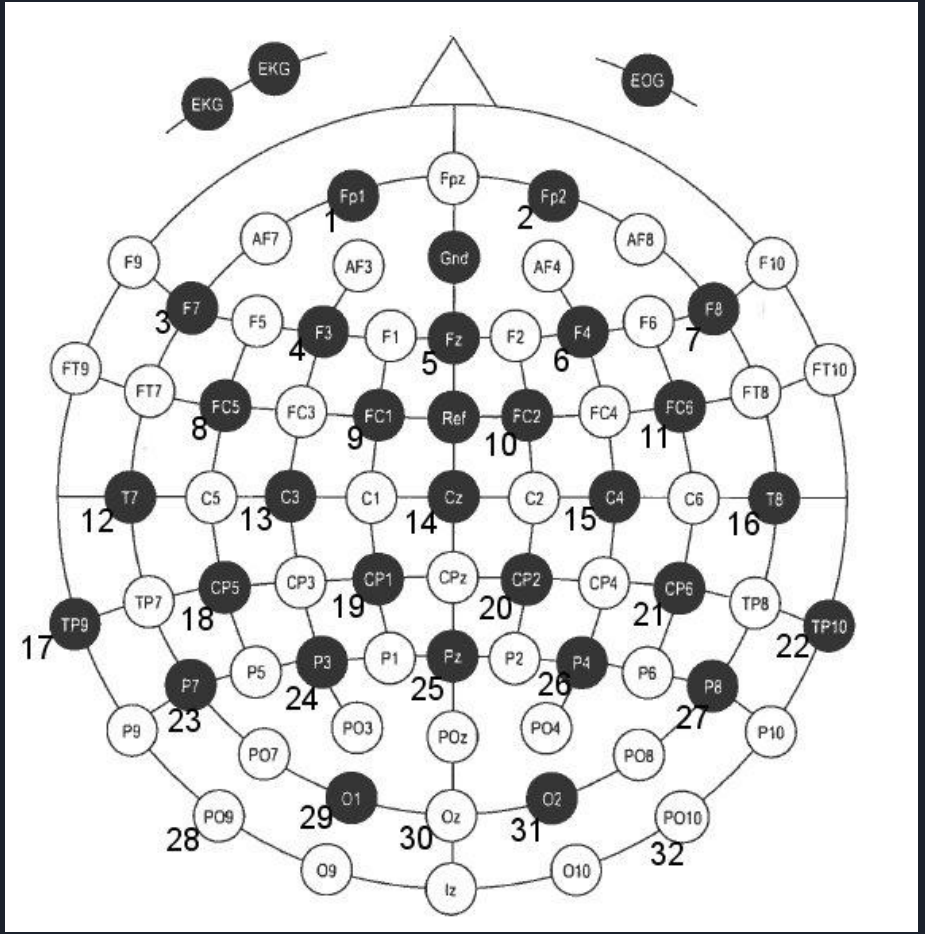
Figure 1

```
import csv
import numpy as np
resultlist = []
for i in range(1,7):
    file1 = 'train/subj1_series'+str(i)+'_events.csv'
    with open(file1, newline='') as csvfile:
        result = list(csv.reader(csvfile))
    result = result[1:]
    for i in range(0,len(result)):
        resultlist.append( [result[i][j] for j in range(1,len(result[i]))])
print(np.array(resultlist).shape)
```

```
(1185498, 6)
```

EEG Signals

- 32 Channels
- Sampling rate 50hz
- Timeframe is ±150ms (±75frames) for each row of data
- Spatial relationship between the electrode locations: All of the electrodes distribute uniformly on scalp.
- P - Parietal
- F - Frontal
- O - Occipital
- C - Central
- T - Temporal
- Fp - Prefrontal



Data Analysis --- Fz Signal in one series

- Position: Forehead. (Here we use subject1_series1 as an example)

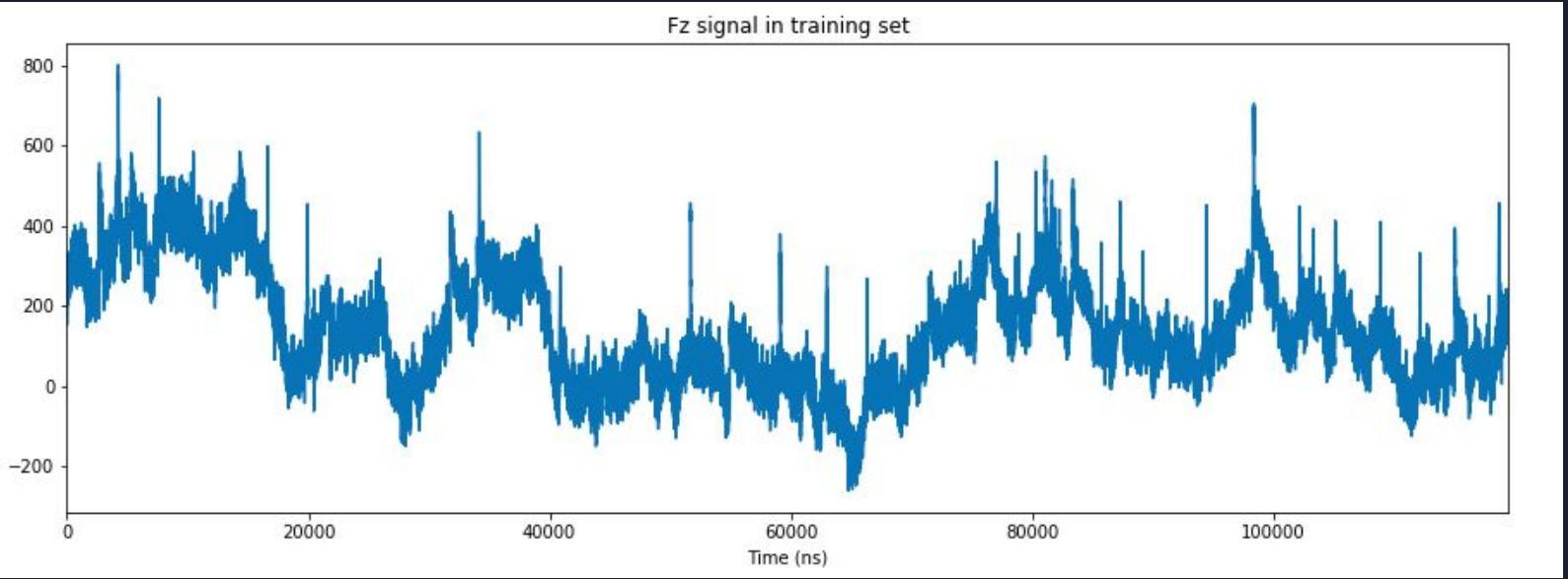


Figure 2

Data Analysis --- Pz Signal

- Position: Back-head

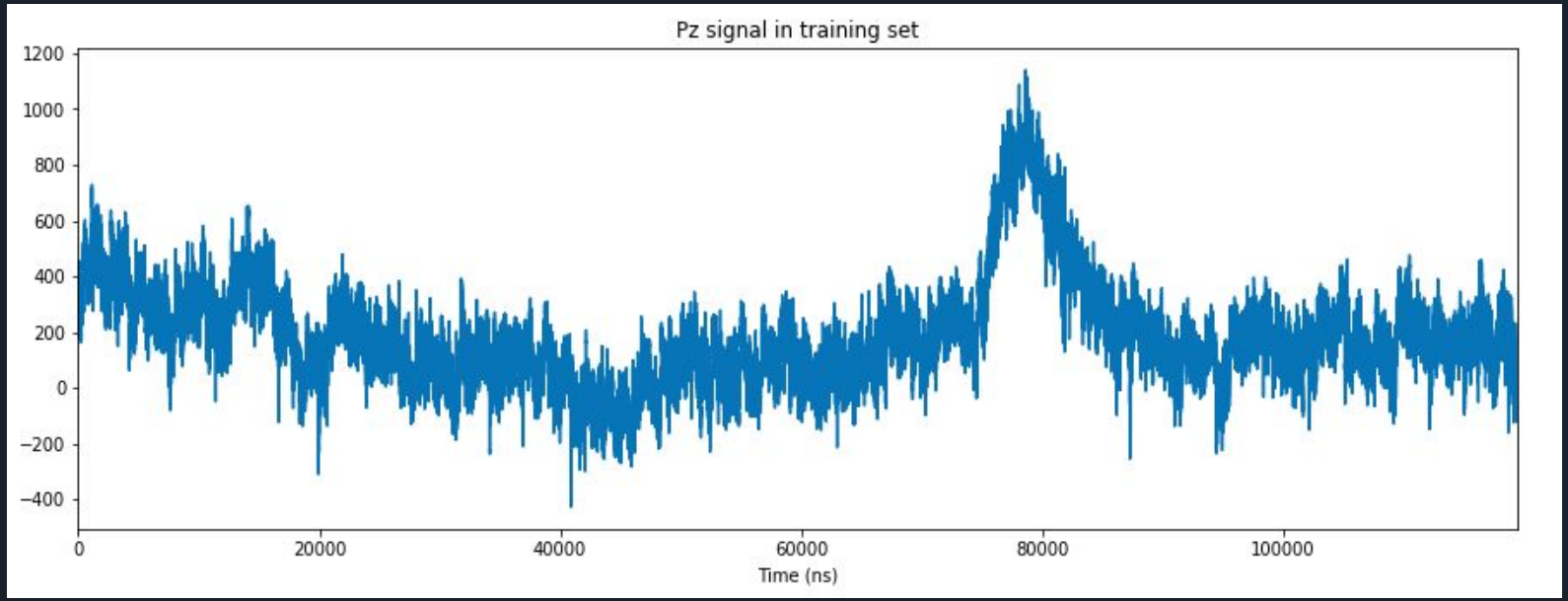


Figure 3

Data Analysis --- C3 Signal

- Position: Left-head

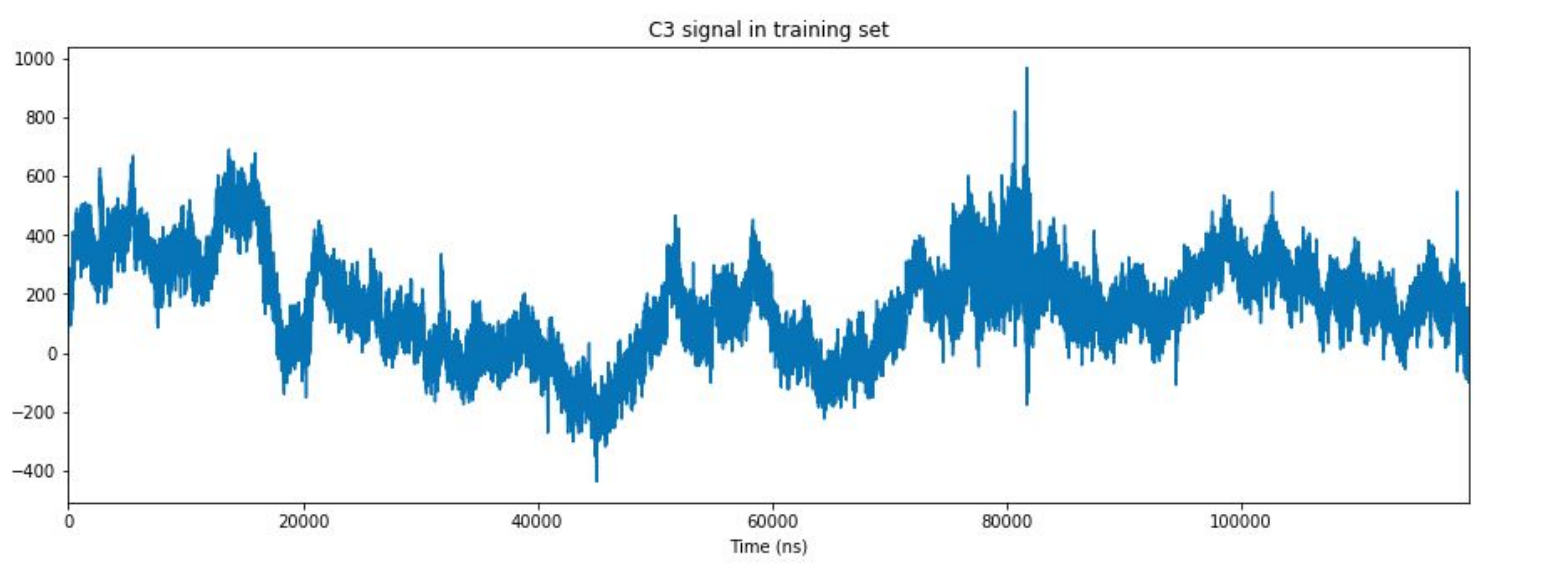


Figure 4

Data Analysis --- C4 Signal

- Position: Right-head

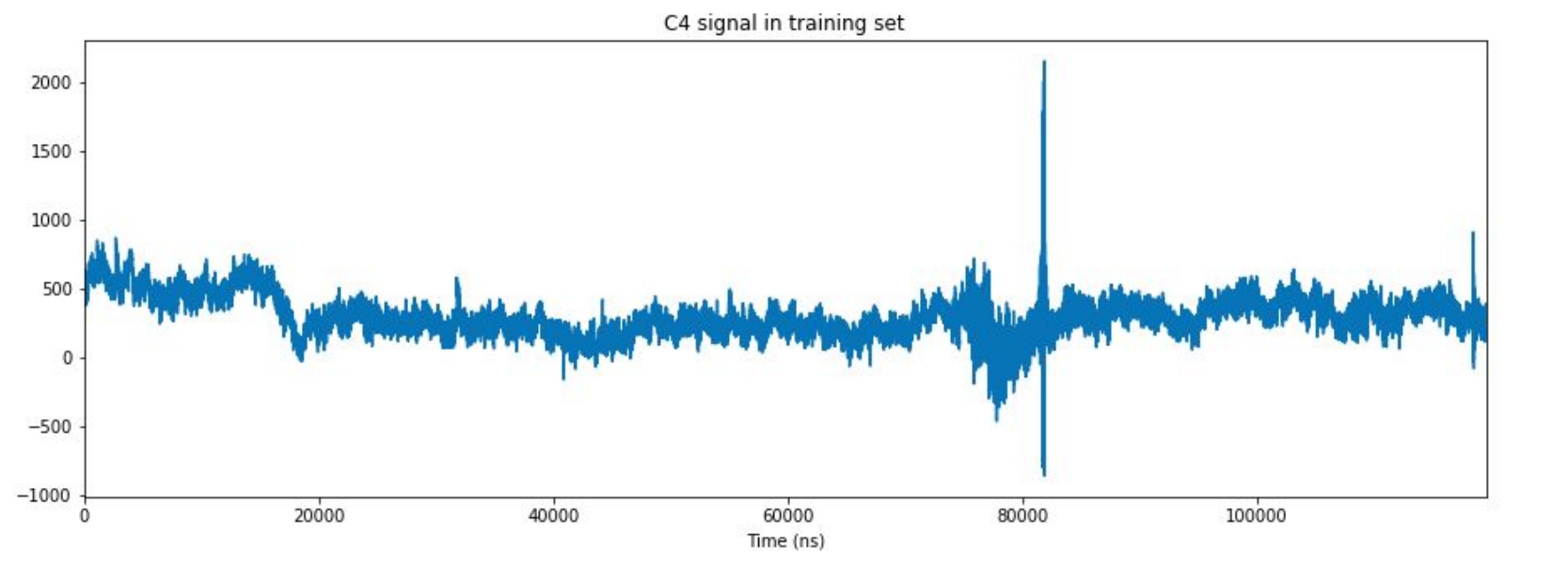


Figure 5

Data analysis -- Hand motions

HandStart, FirstDigitTouch, BothStartLoadPhase, LiftOff, Replace, BothReleased

	HandStart	FirstDigitTouch	BothStartLoadPhase	LiftOff	Replace	Bothreleased
Fz	0.007065336361580768	0.017144912953793223	0.024144879677989207	0.03483140167093763	-0.061990617321090666	-0.056853142790839896
Pz	0.009917277954837943	0.04107831316948137	0.033144842657978416	0.02809829330654008	0.03244634473183917	0.031752864123720276
C3	-0.009204311316436292	0.011428599234671077	0.008999031915078216	0.004629673326964623	0.00190088425587552	-0.014641031487985227
C4	0.006718189866646338	0.05227607613609022	0.05080945691239645	0.05062759612865843	0.017897054913484884	0.019451671290600266

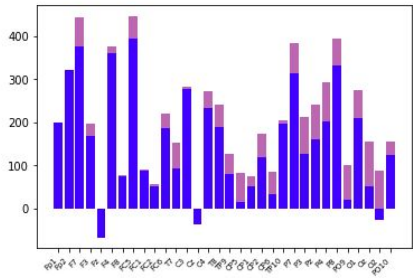
Data Analysis --- Hand Motions

- To find out electrodes that are most associated with each motion. We compare the average signal value in each electrode between two groups (Blue for motion event has occurred and purple for motion event did not occur).

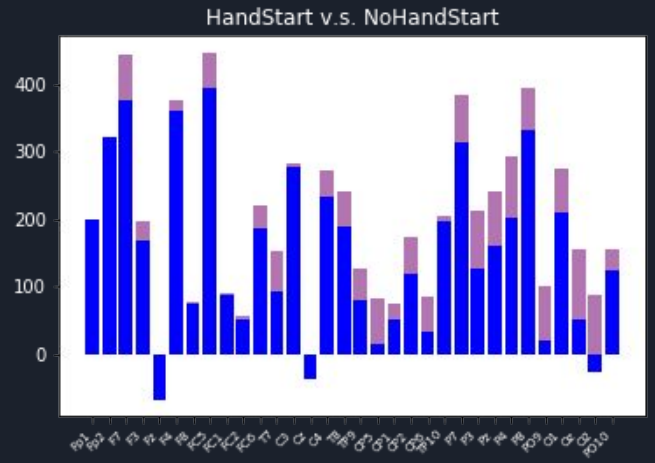
```
HandStart = [0]*32
NoHandStart = [0]*32
count = 0;
for i in range(0,len(data)):
    if(result[i][0] != 0):
        HandStart = HandStart +np.array(data[i])
        count = count +1
    else:
        NoHandStart = NoHandStart+np.array(data[i])
```

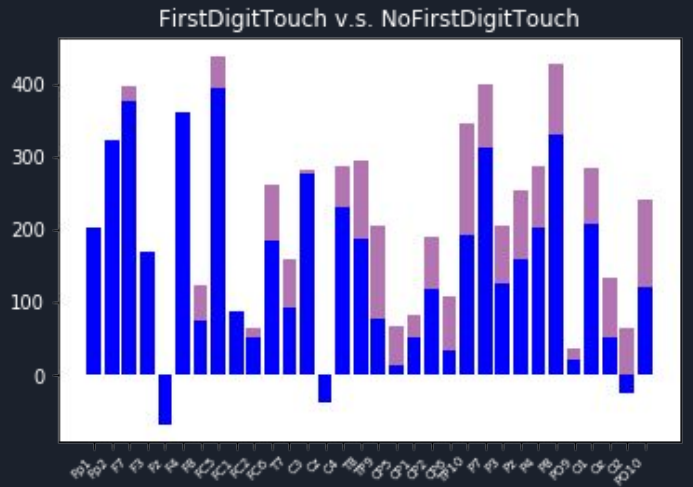
```
avghs = HandStart/count
avgnhs = NoHandStart/(len(data)-count)
```

```
import matplotlib.pyplot as plt
elec = 'Fp1,Fp2,F7,F3,Fz,F4,F8,FC5,FC1,FC2,FC6,T7,C3,Cz,C4,T8,TP9,CP5,CP1,CP2,CP6,TP10,P7,P3,Pz,P4,P8,PO9,O1,Oz,O2,PO10'.split(',')
y_pos = np.arange(len(elec))
plt.bar(y_pos, avghs, color = (0.5,0.1,0.5,0.6))
plt.bar(y_pos, avgnhs, color = 'blue')
plt.xticks(y_pos, elec, color='black', rotation=45,  fontsize='7', horizontalalignment='right')
plt.show()
```



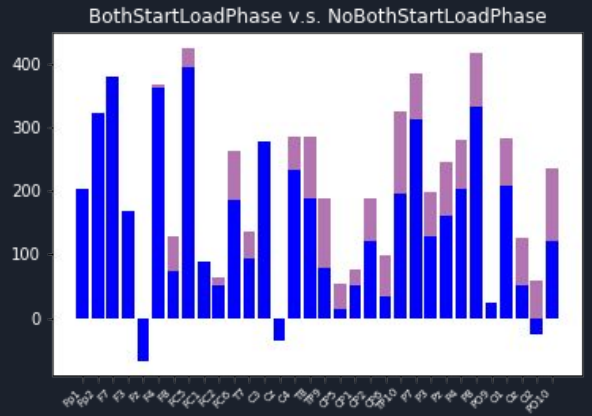
Data Analysis --- Hand Motions

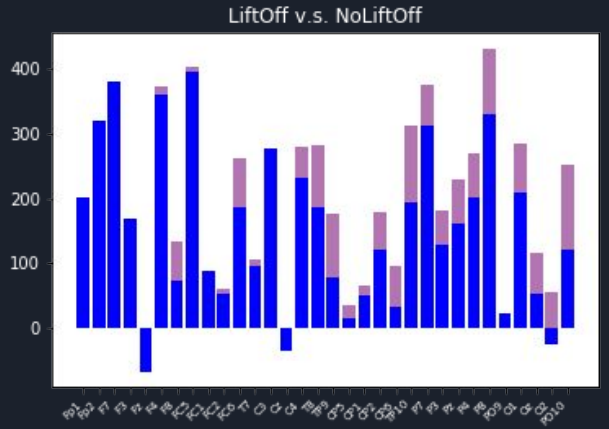


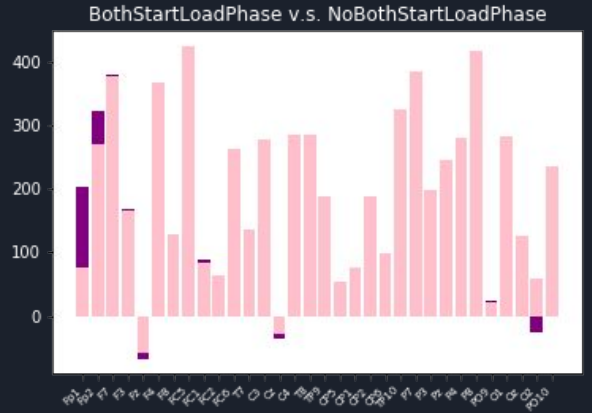


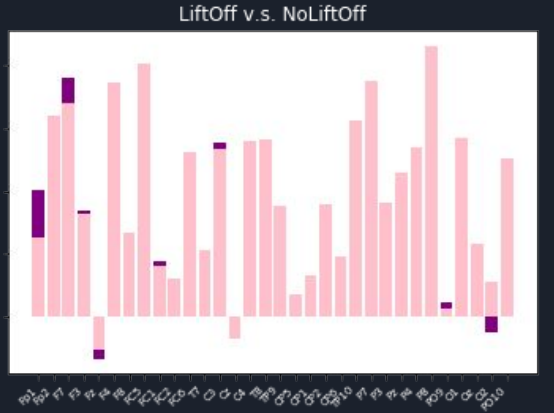
- F7, FC1, C3, CP1, P7, P3, P2, P4, P8, PO9, O1 ,Oz, O2 have large discrepancy between Hand Start and No Hand Start.
- Similar to Hand Start, Parietal and Occipital channels still have large discrepancy between First Digit Touch and No First Digit Touch. Besides, Tp9 and Cp5 are also very different in these two cases.

Data Analysis --- Hand Motions

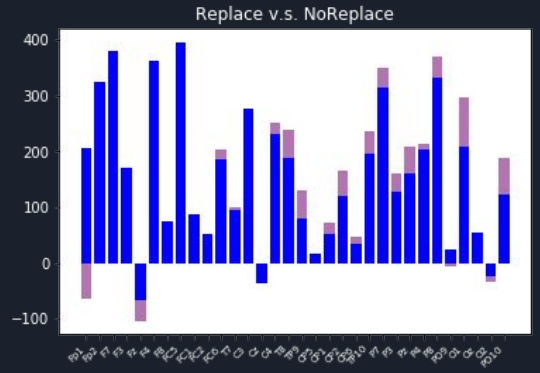


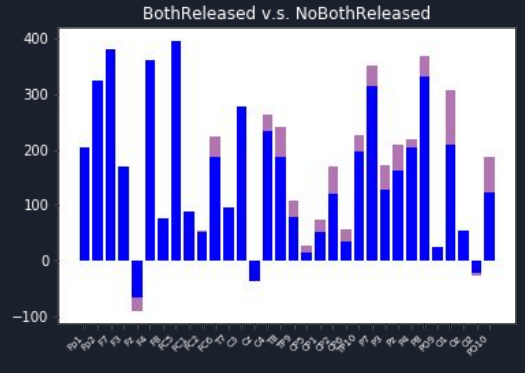


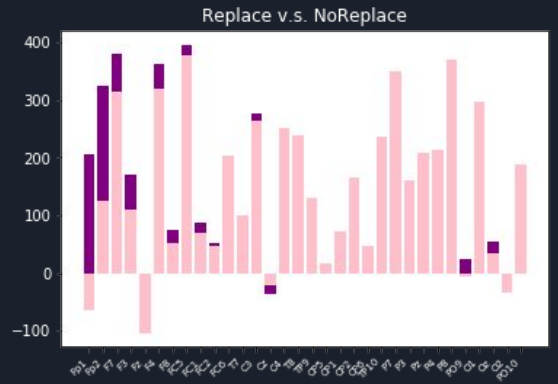


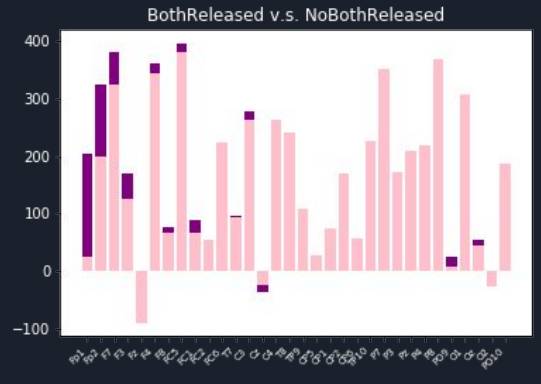


Data Analysis --- Hand Motions









Preprocess

- We did some preprocessing of data including concatenating multiple series together to form training and testing dataset, cleaning data by removing headers and trial names, converting strings into integers.

```
data = np.array([[int(j) for j in datalist[i]] for i in range(0,len(datalist))])
result = np.array([[int(j) for j in resultlist[i]] for i in range(0,len(resultlist))])
datest = np.array([[int(j) for j in datatest[i]] for i in range(0,len(datatest))])
retest = np.array([[int(j) for j in resulttest[i]] for i in range(0,len(resulttest))])
```

- As it will take

Model

- SVM
- Random Forest
- Logistic Regression
- Linear Discriminant Analysis
- MLP
- EEGNET

Models --- SVM

- Use a matrix which contains 32 EEG row signals(as there are 32 electrodes) as feature
- Create 6 clfs to predict 6 hand-motions separately

```
from sklearn.svm import LinearSVC

def train_SVM(X, y):
    clf = LinearSVC()
    clf.fit(X, y)
    return clf
```

```
clf1=train_SVM(X, y1)

clf2=train_SVM(X, y2)
clf3=train_SVM(X, y3)
clf4=train_SVM(X, y4)
clf5=train_SVM(X, y5)
clf6=train_SVM(X, y6)
```

SVM --- Result

- Training accuracy

```
________training acc________
for the hand motion of HandStart, the prediction acc is 0.9572621677713061
for the hand motion of FirstDigitTouch, the prediction acc is 0.8907745865970409
for the hand motion of BothStartLoadPhase, the prediction acc is 0.9082061324228426
for the hand motion of LiftOff, the prediction acc is 0.9330605208542545
for the hand motion of Replace, the prediction acc is 0.7725948985740109
for the hand motion of BothReleased, the prediction acc is 0.9490861618798956
```

- Testing accuracy

```
_____________________in test set_____________________________
for the hand motion of HandStart, the prediction acc is 0.9812063804908183
for the hand motion of FirstDigitTouch, the prediction acc is 0.9592247218279562
for the hand motion of BothStartLoadPhase, the prediction acc is 0.8436978312508733
for the hand motion of LiftOff, the prediction acc is 0.981140192826728
for the hand motion of Replace, the prediction acc is 0.9230421321252859
for the hand motion of BothReleased, the prediction acc is 0.9793862197283364
```

Model --- Random Forest

- Use Random Forest models provided by the Scikit-Learn package
- Choose the best hyperparameter:
  - n_estimators=200, random_state=0

```
In [18]: from sklearn.ensemble import RandomForestClassifier

In [19]: rf1_clf = RandomForestClassifier(n_estimators=200,random_state=0)
         rf1_clf.fit(X,y1)

         rf2_clf = RandomForestClassifier(n_estimators=200,random_state=0)
         rf2_clf.fit(X,y2)

         rf3_clf = RandomForestClassifier(n_estimators=200,random_state=0)
         rf3_clf.fit(X,y3)

         rf4_clf = RandomForestClassifier(n_estimators=200,random_state=0)
         rf4_clf.fit(X,y4)

         rf5_clf = RandomForestClassifier(n_estimators=200,random_state=0)
         rf5_clf.fit(X,y5)

         rf6_clf = RandomForestClassifier(n_estimators=200,random_state=0)
         rf6_clf.fit(X,y6)
```

Random Forest --- Result

- Test accuracy

```
________Random Forest Model, in test set________
for the hand motion of HandStart, the prediction acc is 0.9812468285077623
for the hand motion of FirstDigitTouch, the prediction acc is 0.9811365157342786
for the hand motion of BothStartLoadPhase, the prediction acc is 0.9809379527420078
for the hand motion of LiftOff, the prediction acc is 0.9811585782889753
for the hand motion of Replace, the prediction acc is 0.9812541826926613
for the hand motion of BothReleased, the prediction acc is 0.9812468285077623
```

Model --- Logistic Regression

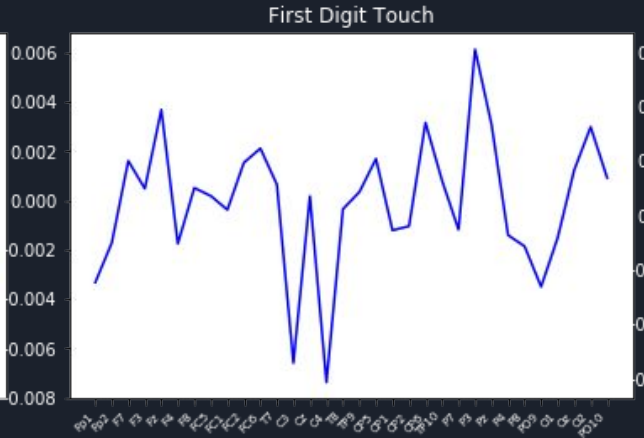
We observe that the accuracy of HandStart, FirstDigitTouch, BothStartLoadPhase, LiftOff, Replace, BothReleased by logistic regression are very close to each other.

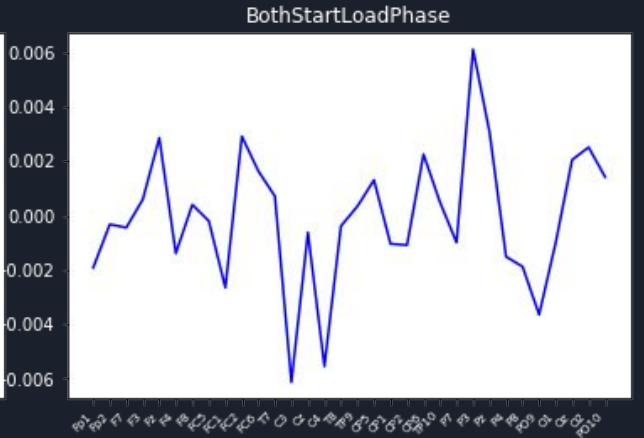
After we got the convergence warning in the beginning, we changed the maximum iterations from 100 to 200, but the warning still appeared. The accuracies under these two max_iter values are very close, but the speed with the larger max_iter value is much slower. Thus, we stopped to try larger max_iter value, as it is not worthwhile to do that with our limitation of time and computing resources.

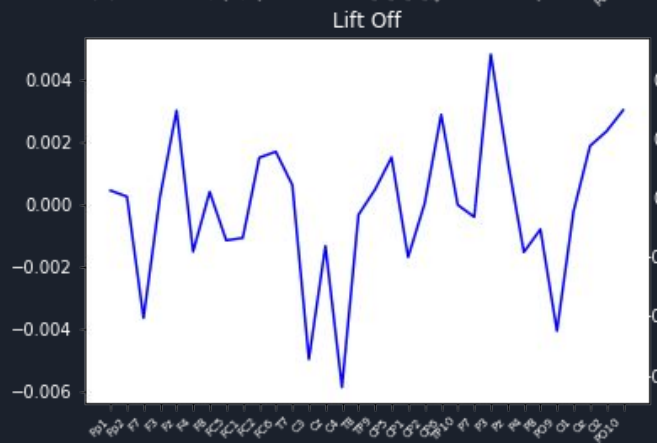
```
from sklearn.linear_model import LogisticRegression
for i in range(0,6):
    clf = LogisticRegression(solver ='lbfgs' , max_iter = 200).fit(data, result[:,i])
    print(clf.score(datest,retest[:,i]))
```

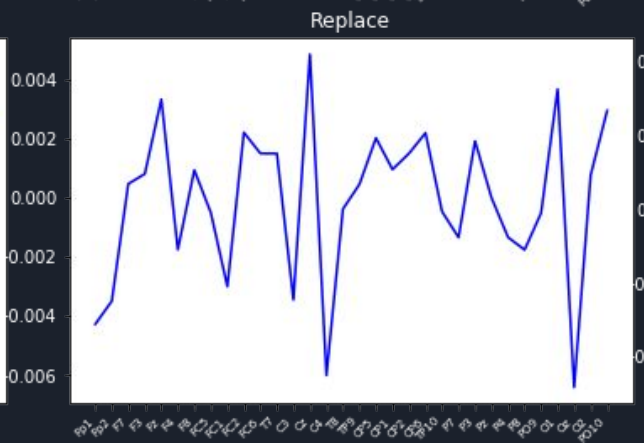
```
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:758: ConvergenceW
er of iterations.
  "of iterations.", ConvergenceWarning)
0.9569469889486437
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:758: ConvergenceW
er of iterations.
  "of iterations.", ConvergenceWarning)
0.9551065033306035
0.9568878907865965
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:758: ConvergenceW
er of iterations.
  "of iterations.", ConvergenceWarning)
0.9569427676513547
/opt/conda/lib/python3.6/site-packages/sklearn/linear_model/logistic.py:758: ConvergenceW
er of iterations.
  "of iterations.", ConvergenceWarning)
0.948559271235236
0.9568330139218385
```

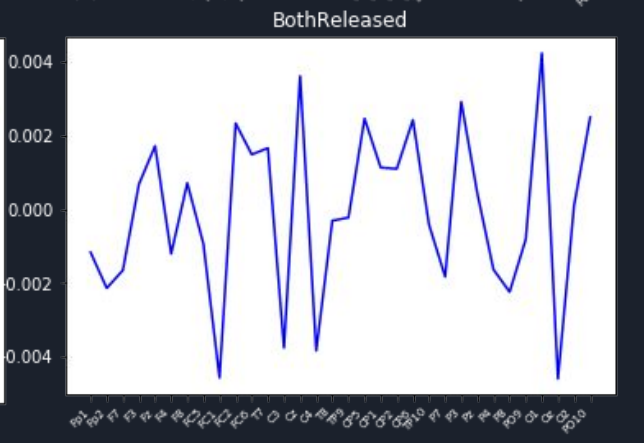
Model --- Logistic Regression Coefficient











Model --- LDA

The accuracies for each of the event are very close, and high.

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
for i in range(0,6):
    clf = LinearDiscriminantAnalysis()
    clf.fit(data, result[:, i])
    print(clf.score(datest,retest[:,i]))
```

```
0.9570609639754489
0.9566852685167205
0.9569005546784638
0.9569469889486437
0.9568794481920184
0.9569427676513547
```

Model-MLP

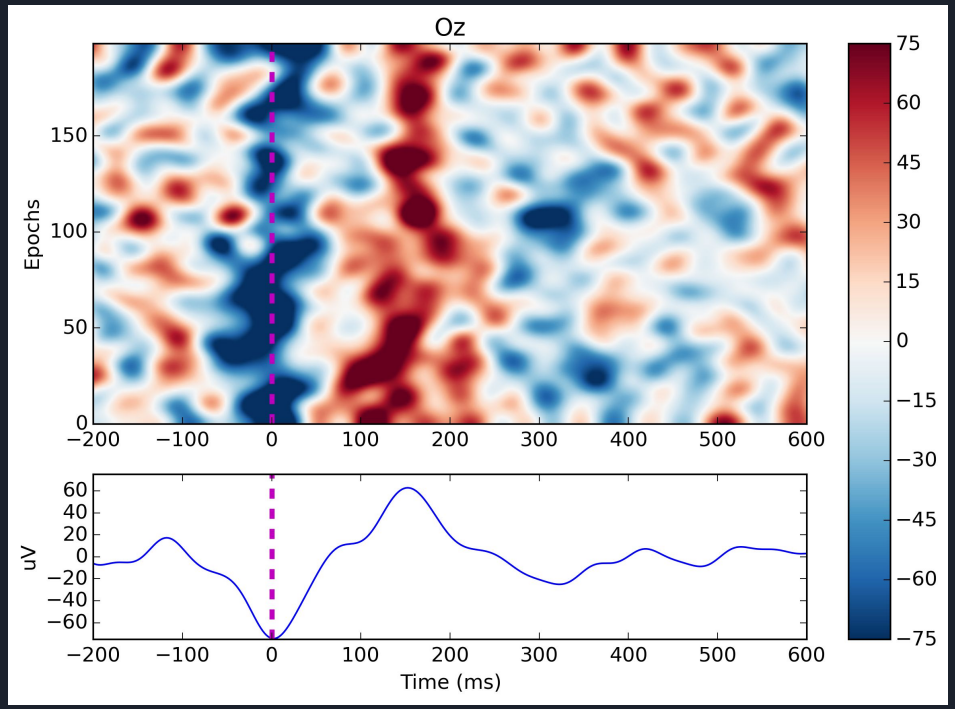
```
from keras.models import Sequential
from keras.layers import Dense
# create model
model = Sequential()
model.add(Dense(12, input_dim=32, activation='sigmoid'))
model.add(Dense(6, activation='sigmoid'))
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Fit the model
model.fit(data, result, epochs=10, batch_size=10)
# evaluate the model
scores = model.evaluate(datest, retest)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
Epoch 1/10
1185498/1185498 [==============================] - 606s 512us/step - loss: 0.1058 - acc: 0.9755
Epoch 2/10
1185498/1185498 [==============================] - 613s 517us/step - loss: 0.1039 - acc: 0.9757
Epoch 3/10
1185498/1185498 [==============================] - 578s 487us/step - loss: 0.1046 - acc: 0.9757
Epoch 4/10
```

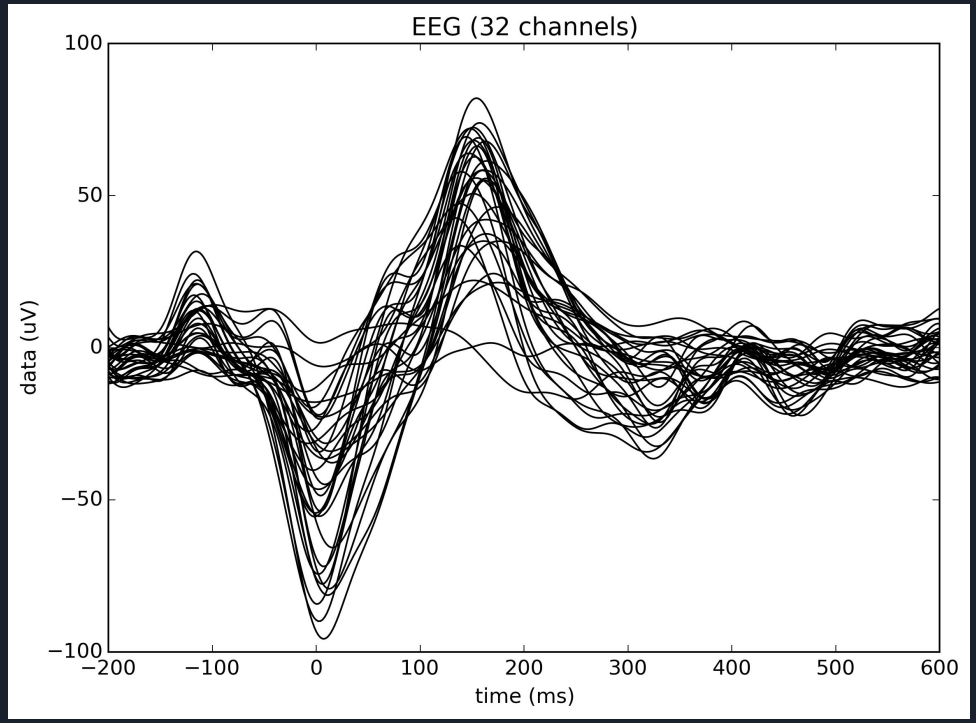
Conclusion

- From our analysis of EEG signals, we discover that the correlation between the hand motions and each single signal is not very high
- However, from the high accuracy of our prediction models, the pattern of raw data collected from the 32 signals could clearly indicate a certain hand motion.
- Our experiment results of training and testing models are consistent with our pre-analysis of data.

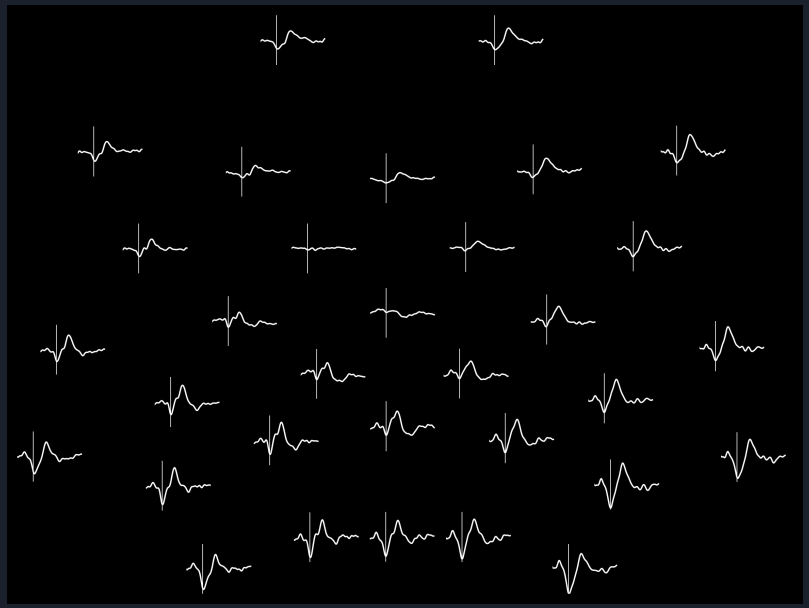
Other Reference [2]

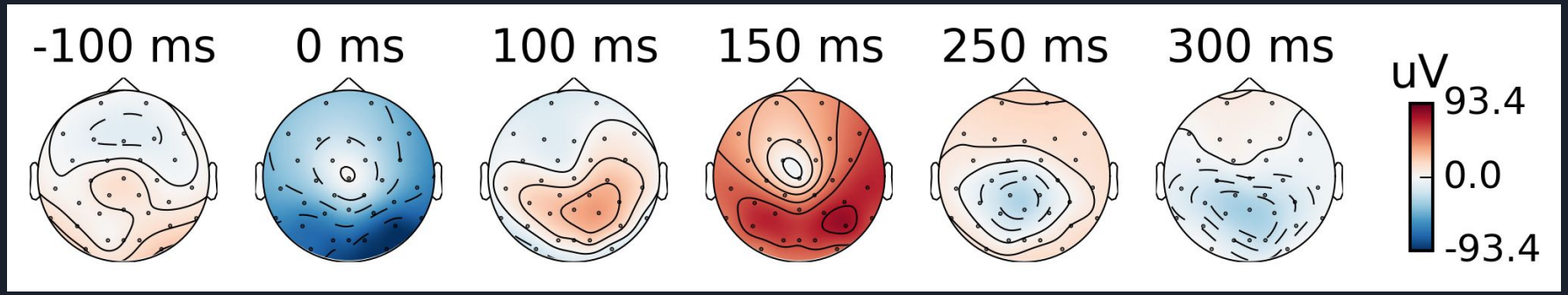


Other Reference [2] --- Evoke Time



Other Reference [2]





Further ...

- Separate different channels of EEG signals that fall into different frequency bands (alpha, beta, gamma waves) [3]

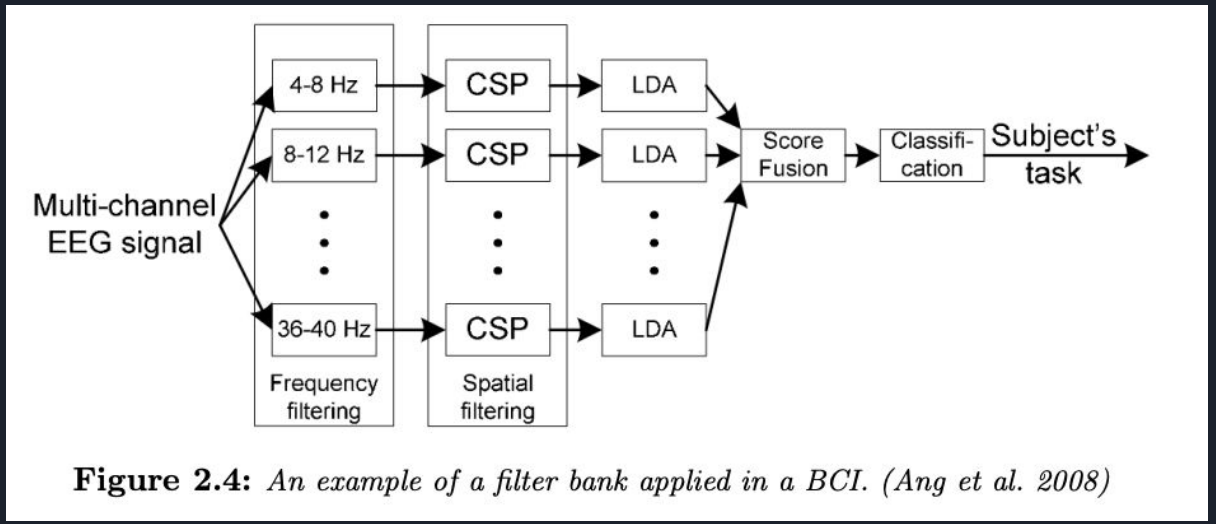


Figure 2.4: *An example of a filter bank applied in a BCI. (Ang et al. 2008)*

Further...

Artificial Neural

Network (ANN)

and CSP

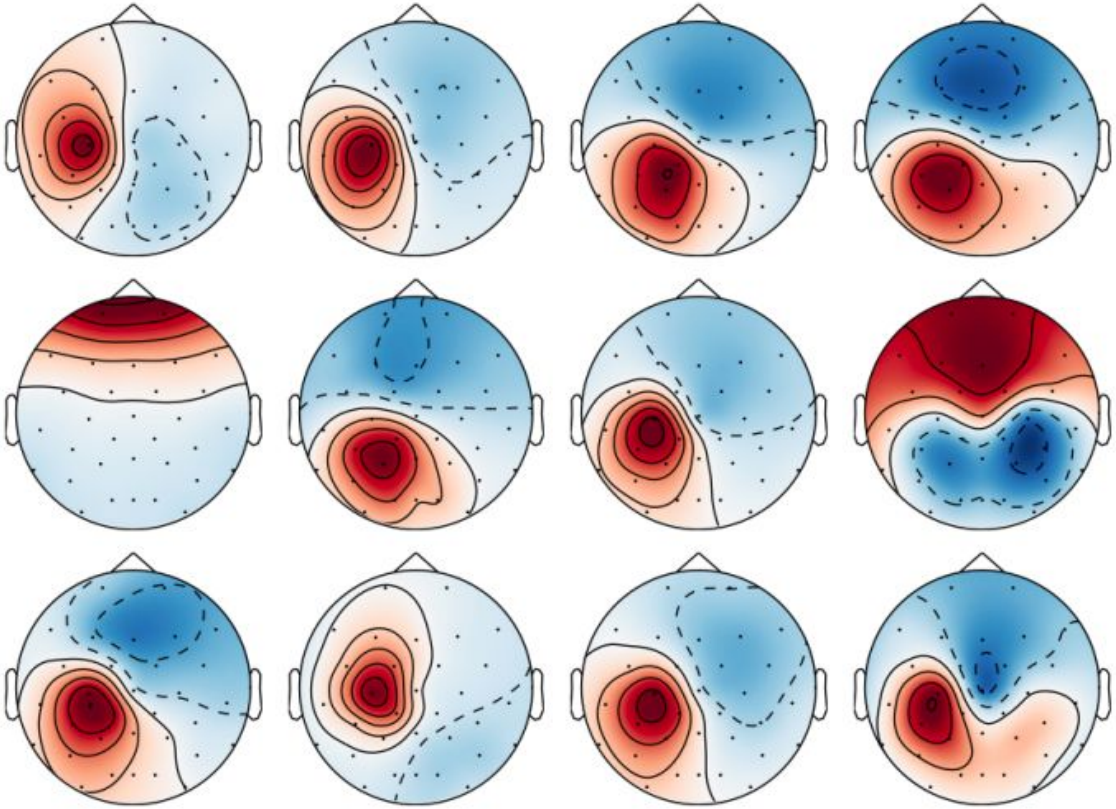


Figure 4.2: *CSP patterns of the subjects. (Barachant 2015)*

Further....

EGGNet:

- spatial filtering and temporal convolution
- Our model was not successful --- shadow and deep network convolution
- View some github references [4][5]

Reference

[1] https://www.kaggle.com/c/grasp-and-lift-eeg-detection/data

[2] https://www.kaggle.com/alexandrebarachant/visual-evoked-potential-vep/data

[3] https://tdk.bme.hu/VIK/DownloadPaper/Kezmozdulatok-detektalasa-EEG-jel-alapjan

[4] https://github.com/vlawhern/arl-eegmodels/blob/master/EEGModels.py

[5] https://github.com/vlawhern/arl-eegmodels/blob/master/examples/ERP.py

Thank you for Watching !