1. Introduction:

The project attempts to identify the movement phases of a human hand during grasping and lifting by detecting brain waves. We want to detect six events: HandStart, FirstDigitTouch, BothStartLoadPhase, LiftOff, Replace and BothReleased. Then we used the 1-dimensonal convolutional neural network module to predict the results.

2. Dataset

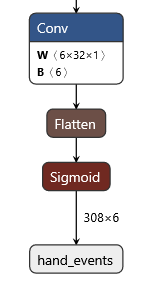
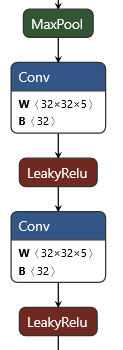
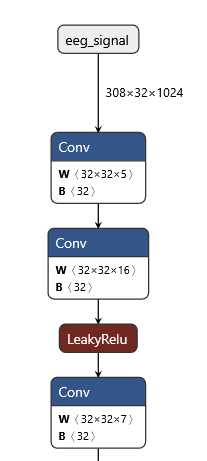
3. Model Framework

First, we preprocess the datasets using a bandpass filter provided by EEGlab. The bandpass filter will preserve the frequency 1-50HZ signals in the data. By doing this, we can get rid of data that is not brain signals.

We take the last two datasets as test data and the remaining datasets are used for training. Then we calculated the training data mean and standard deviation for 32 channels and used them to normalize the training data and the test data.

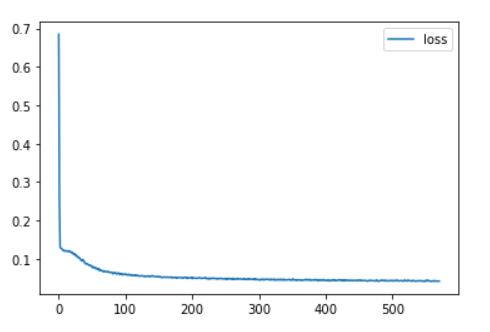
The signal data shape we feed into the module is 1024\*32 tensor and the target data shape is a length 6 tensor. The 32 is the number of channels. The 1024 is the sampled data number, which is also the start of the target data/hand movement. It means we used the pass two second data to predict what happening now. The target data present 6 different event. If one of the events was happening, the relatively data point will set to 1, and 0 otherwise.

The CNN module we used just like following picture. Because the brain signal data is the one-dimensional data, the convolution method we used is 1-dimensional convolution. Doing so, we can extract the features in each channel separately.



X6

When training the module, we used Binary Cross Entropy to calculate the loss and used Adam as the optimizer. The Adam learning rate is 0.001 and betas is (0.5, 0.99).



4. Validation

We used ROC curve (Receiver operator characteristic curve) to validate our results. The ROC curve is a graphical plot that illustrates the diagnostic ability of the binary classifier system. The ROC curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. We also calculated the area under ROC Curve (between 0 to 1). Larger the area means better the result.

5. Usage

First, you need to download the data from Kaggle.

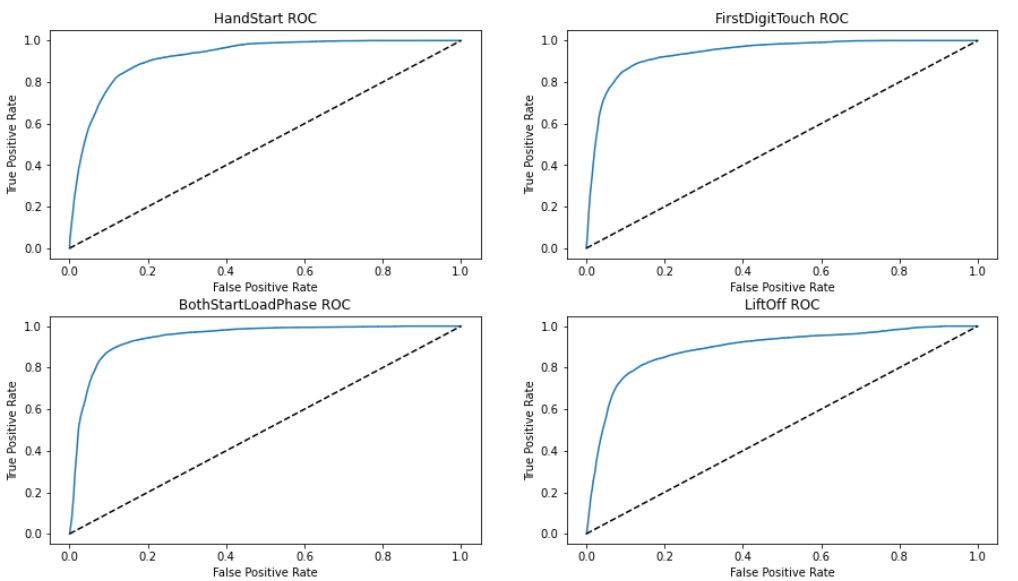
<https://www.kaggle.com/c/grasp-and-lift-eeg-detection/data>

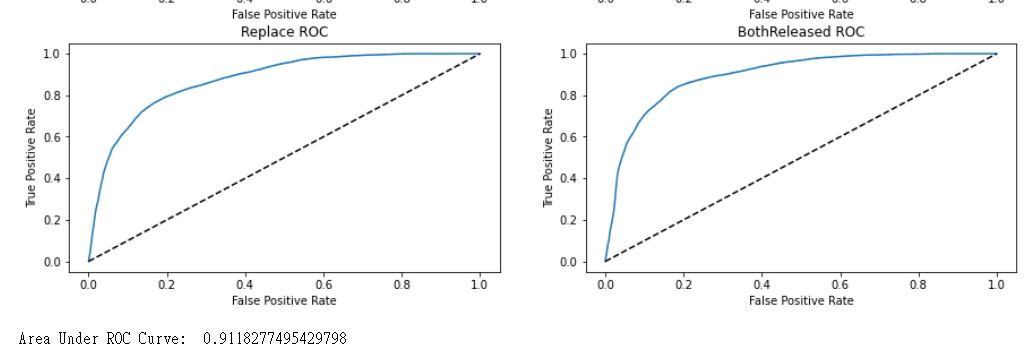
After unzip the data, you will find out there are two folder: ./train and ./test. We only need the train folder. Drag it to the same folder as final\_1.m and final\_cnn-eeg.ipynb.

In order to do data preprocessing, you need to run EEGlab in Matlab first to load the bandpass filter function. Then you can run final\_1.m to do the filtering automatedly. Be sure to create the empty ./newtrain folder first.

After doing bandpass filter, you can open the final\_cnn-eeg.ipynb and test our code.

6. Result:





7. References:

Kaggle:

<https://www.kaggle.com/c/grasp-and-lift-eeg-detection/data>

<https://www.kaggle.com/code/banggiangle/cnn-eeg-pytorch>

EEGlab:

<https://sccn.ucsd.edu/eeglab/index.php>

pytorch:

<https://pytorch.org/docs/stable/generated/torch.nn.Conv1d.html>

Others:

<https://en.wikipedia.org/wiki/Receiver_operating_characteristic>