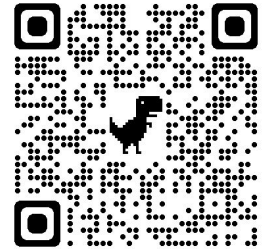


Group 7

EEG Presentation




- Anuj Gupta
- Nutapol Thungpao
- Praewphan Tocharoenkul
- Suphawich Sungkhavorn

Introduction

This project aims to generate a machine learning model that is able to generalize EEG signals no matter the subject. The model should be able to discern between movement of the subject's left and right fists.

DATASET - [EEG Motor Movement/Imagery Dataset](#) (Schalk, G., McFarland, D.J., Hinterberger, T., Birbaumer, N., Wolpaw, J.R. BCI2000: A General-Purpose Brain-Computer Interface (BCI) System. IEEE Transactions on Biomedical Engineering 51(6):1034-1043, 2004.)

Real movement of the left/right fists are used in addition to the motor imagery variants of the same action

- 
1. A target appears on either the left or the right side of the screen. The subject opens and closes the corresponding fist until the target disappears. Then the subject relaxes.
 2. A target appears on either the left or the right side of the screen. The subject imagines opening and closing the corresponding fist until the target disappears. Then the subject relaxes.
 3. A target appears on either the top or the bottom of the screen. The subject opens and closes either both fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears. Then the subject relaxes.
 4. A target appears on either the top or the bottom of the screen. The subject imagines opening and closing either both fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears. Then the subject relaxes.

Preprocessing

Preprocessing of Data

In this project, all subjects' experimental runs were pooled and concatenated together in an effort to get the model to generalize well. This means that every subject's trials were shuffled when converted to DataLoader.

This meant that some models performed better than others. Additionally, different combinations of electrode channels were attempted by each group member to obtain the best possible test accuracy.

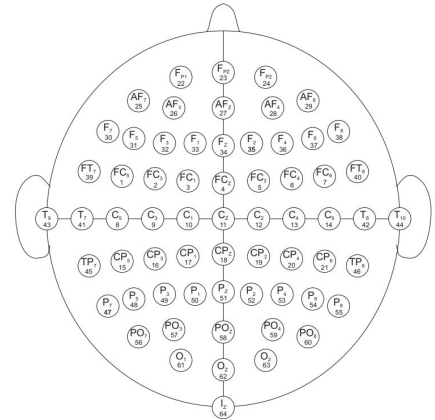
Three distinct events are present in the data (left fist clenching, right fist clenching, and complete rest). Only the first two events from these are extracted to train and test our models.

```
# home directory + datasets folder
path = '' #change to where the dataset is stored
base_url = 'https://physionet.org/files/eegmmidb/1.0.0/'
# subjects = [1]
runs = [3,4, 7,8, 11,12] # only take the trials where left and right fist is involved
subjects = [i for i in range(1, 21)]

eeg = EEG(path, base_url, subjects, runs)

# apply filter
freq = (1., 40.) #bandpass filter values
eeg.filter(freq=freq)

eeg.create_epochs()
```

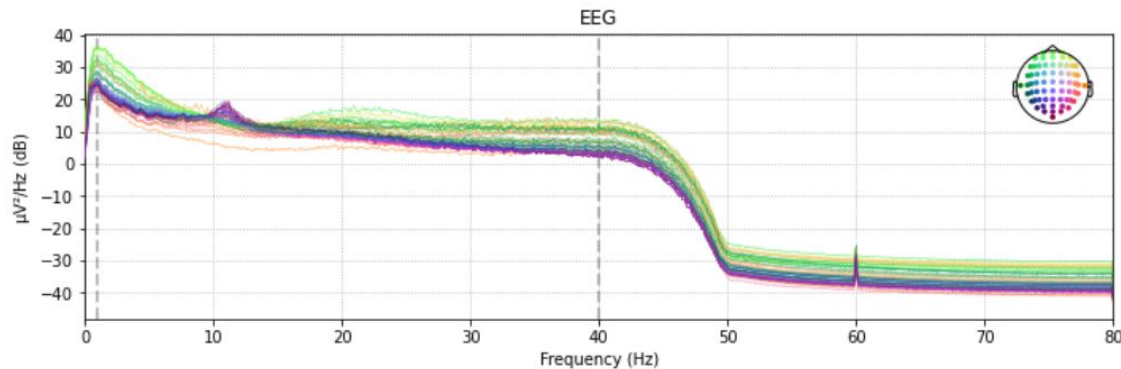


Preprocessing of Data

The dataset contains upwards of 1,500 brain recordings comprised of 109 subjects. Different combinations of these subjects were used during our experiments; 5, 10, 20, etc.

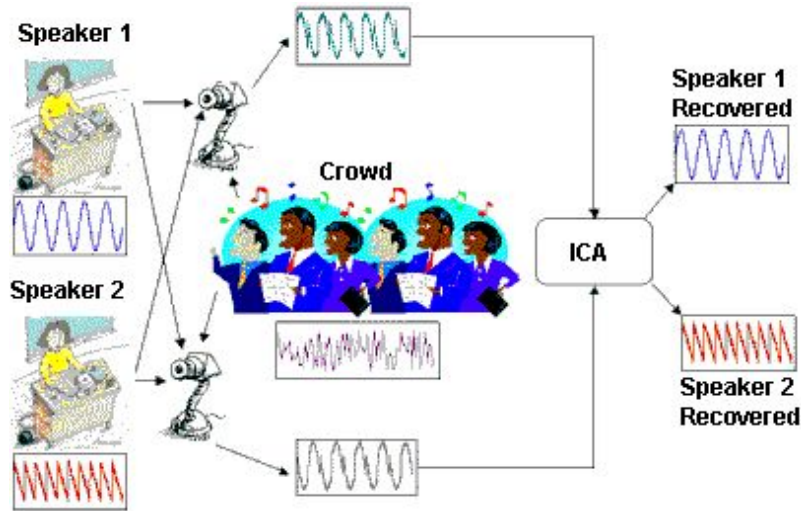
1. Select the number of subjects and the experimental runs to use [3, 4, 7, 8, 11, 12]
2. Conversion to MNE Raw European Data Format
3. Standardize electrode channel names and locations to international 10-05 montage (64 channels)
4. Implement bandpass filter from 1Hz to 40Hz, as seen on papers

```
eeg.raw.plot_psd()
```



EEG signal after bandpass filter

Independent Component Analysis (ICA)



Statistica Independent Component Analysis (ICA), part of the Data Mining suite of analyses, is designed for signal separation using a well established and reliable statistical method known as Independent Component Analysis.

Imagine being in a room with a crowd of people and two speakers giving presentations at the same time. The crowd is making comments and noises in the background. We are interested in what the speakers say and not the comments emanating from the crowd. There are two microphones at different locations, recording the speakers' voices as well as the noise coming from the crowd. Our task is to separate the voice of each speaker while ignoring the background noise.



Modeling and Training

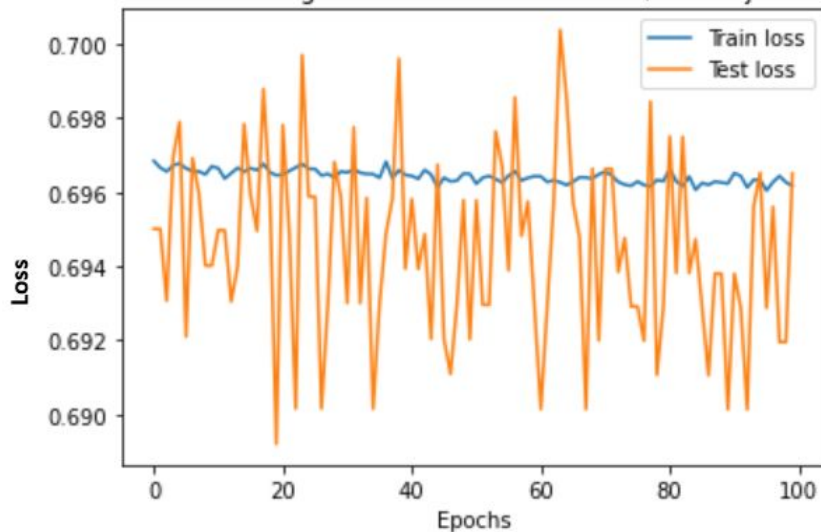
Bi-Directional LSTM

Modeling - Bi-LSTM (64 channels)

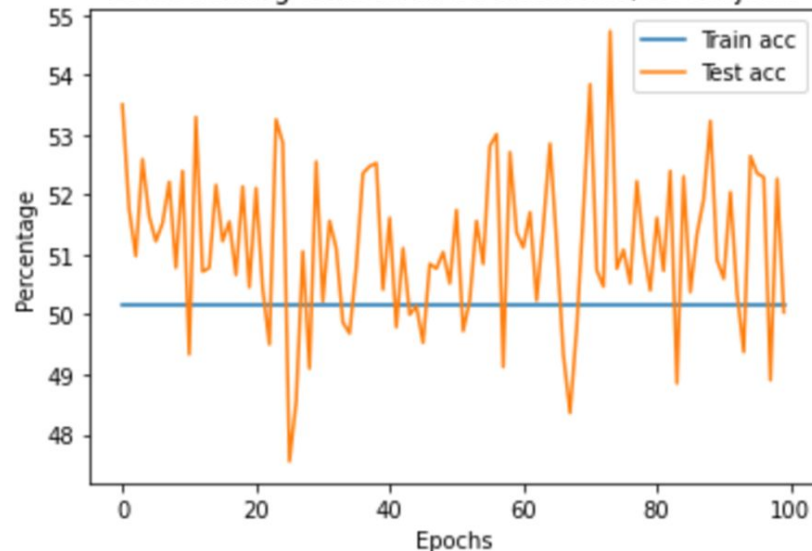
```
model_lstm
```

```
LSTM(  
  (lstm): LSTM(64, 128, num_layers=2, batch_first=True, dropout=0.25, bidirectional=True)  
  (fc): Linear(in_features=256, out_features=2, bias=True)  
  (softmax): LogSoftmax(dim=1)  
)
```

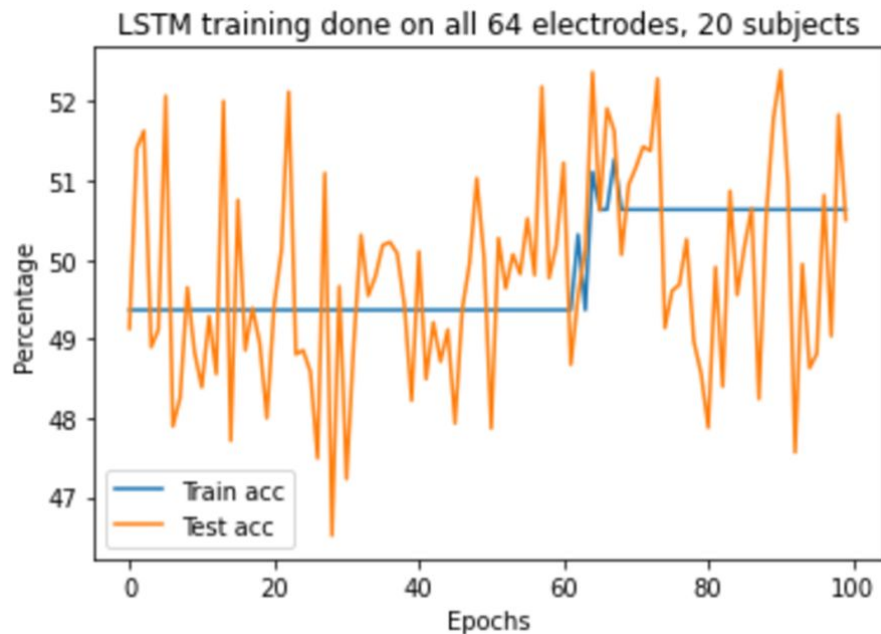
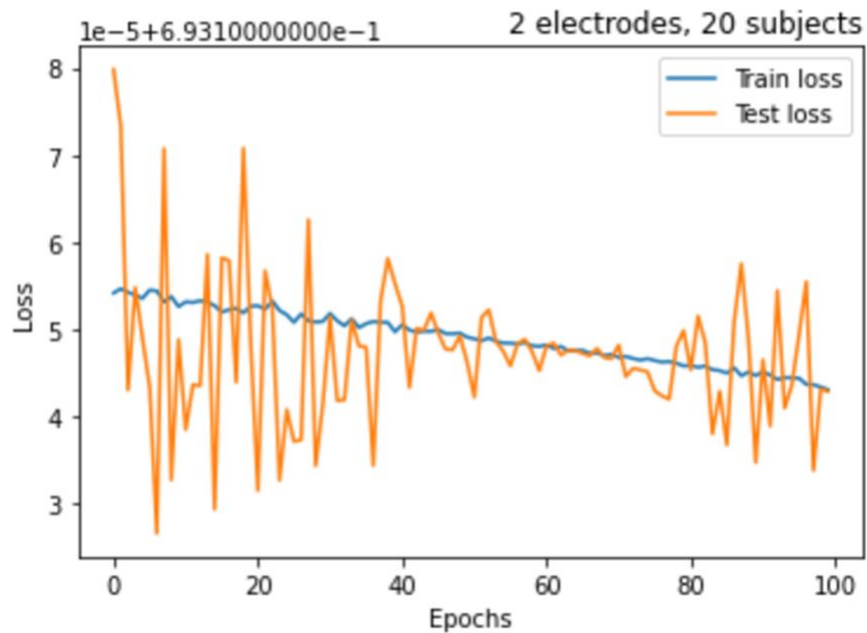
LSTM training done on all 64 electrodes, 20 subjects



LSTM training done on all 64 electrodes, 20 subjects

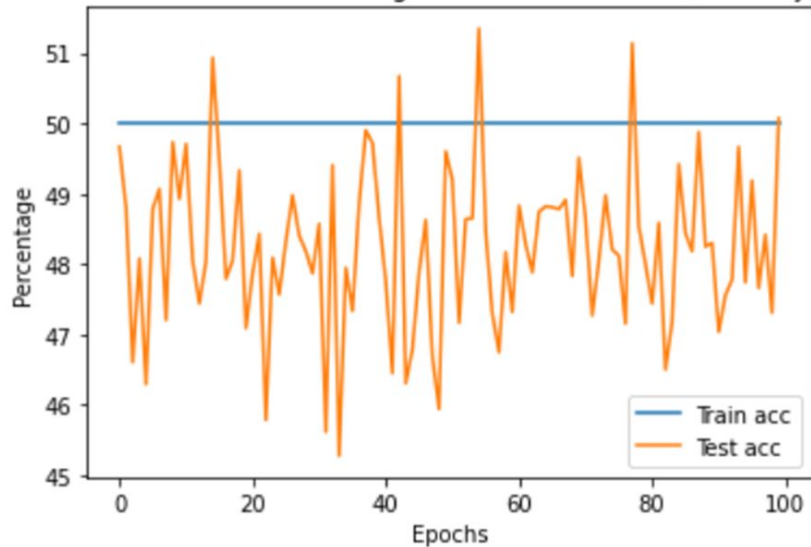


Modeling - Bi-LSTM (2 channels)

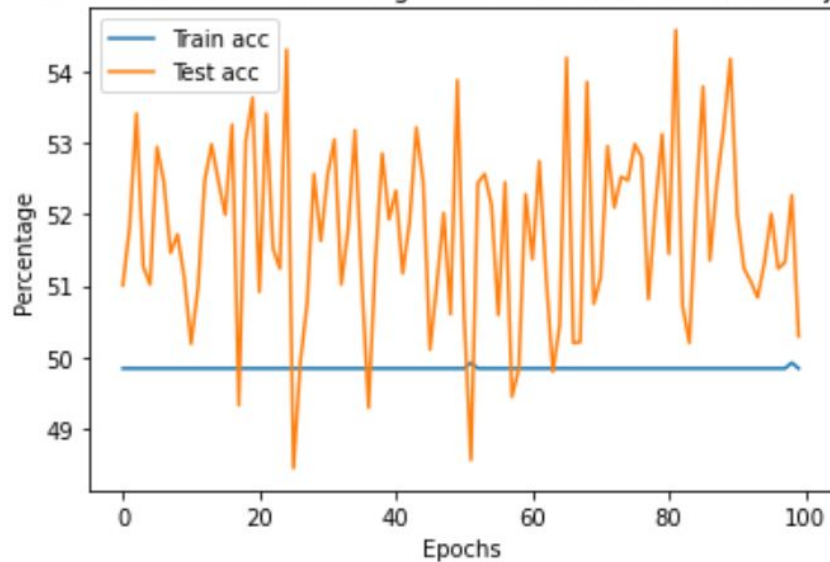


Modeling - Uni-LSTM

Unidirectional LSTM training done on 2 electrodes, 20 subjects



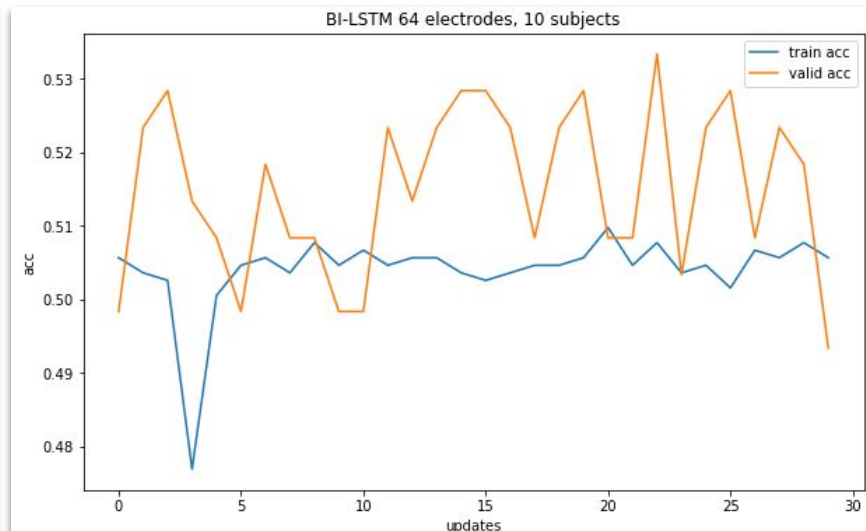
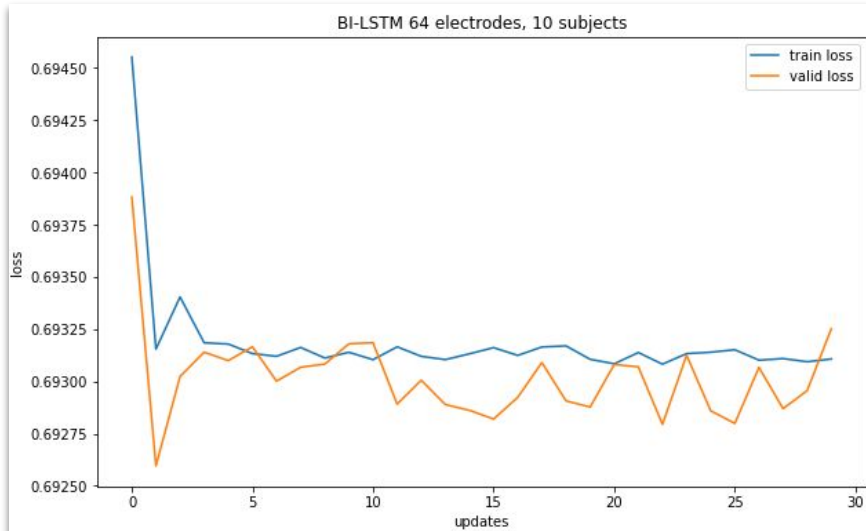
Unidirectional LSTM training done on 64 electrodes, 20 subjects



LSTM + Attention

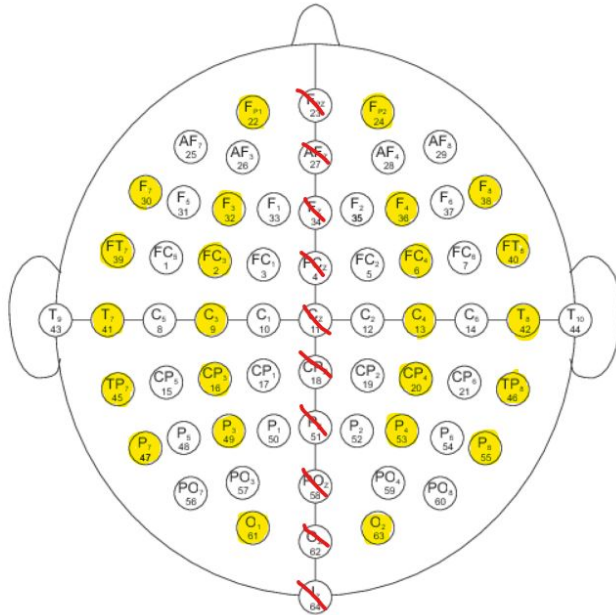
Bi-LSTM+attention(64 channels)

```
LSTM(  
  (lstm): LSTM(64, 128, batch_first=True, dropout=0.2, bidirectional=True)  
  (fc): Linear(in_features=256, out_features=1, bias=True)  
  (dropout1): Dropout(p=0.2, inplace=False)  
  (dropout2): Dropout(p=0.1, inplace=False)  
  (dropout3): Dropout(p=0.2, inplace=False)  
)
```



LSTM+attention(24 channels)

Fp1,Fp2,F7,F3,F4,F8,FT7,Fc3,Fc4,FT8,T7,C3,C4,T8,TP7,CP3,CP4,TP8,P7,P3,P4,P8,O1,O2



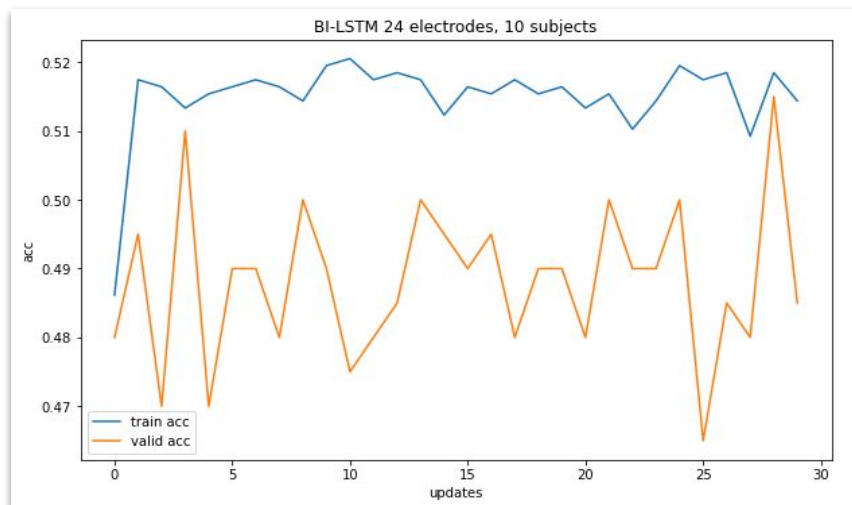
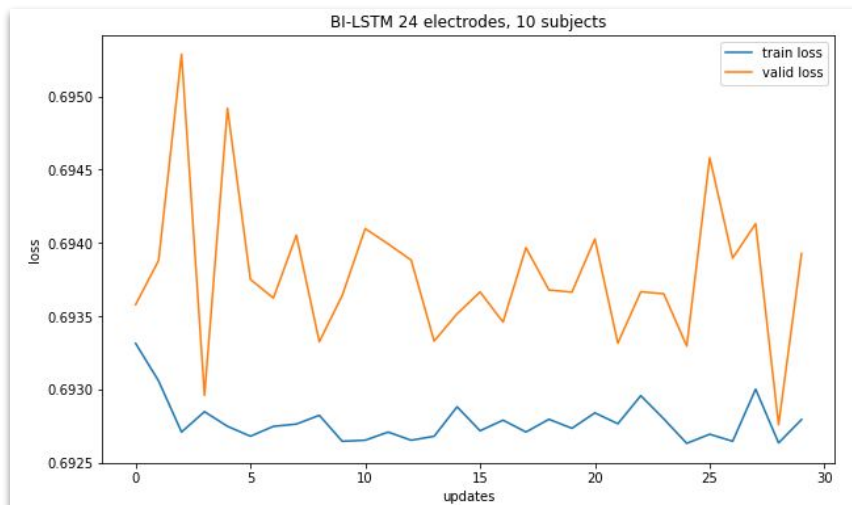
In this context, out of the 64 EEG channels available in the EEG test material, the 10 central sensors were discarded due to their non-symmetric nature [23]. The utilized sensor pairs were selected based on the topology described in [23].

ref : “Classification of Hand Movements from EEG using a Deep Attention-based LSTM Network”

<https://arxiv.org/pdf/1908.02252.pdf>

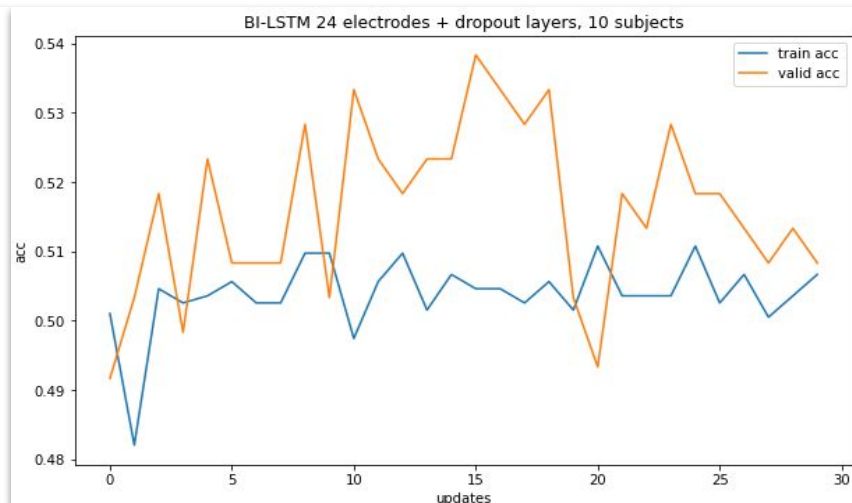
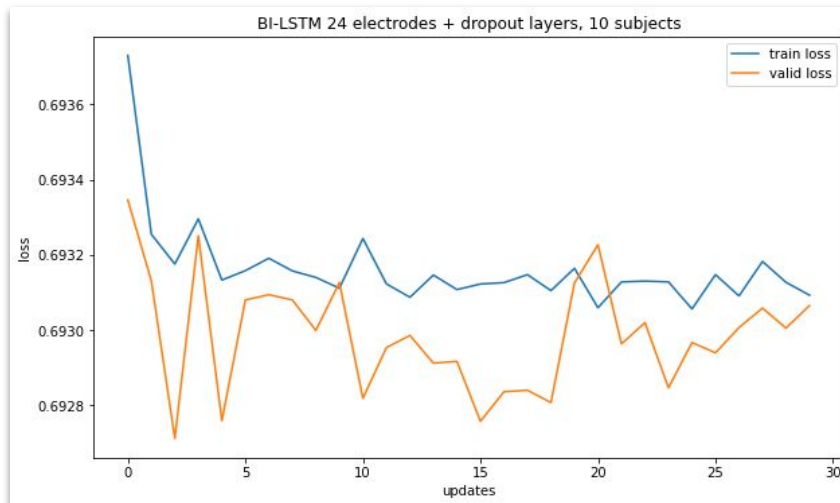
Bi-LSTM+attention(24 channels)

```
LSTM(  
  (lstm): LSTM(24, 128, batch_first=True, dropout=0.2, bidirectional=True)  
  (fc): Linear(in_features=256, out_features=1, bias=True)  
)
```



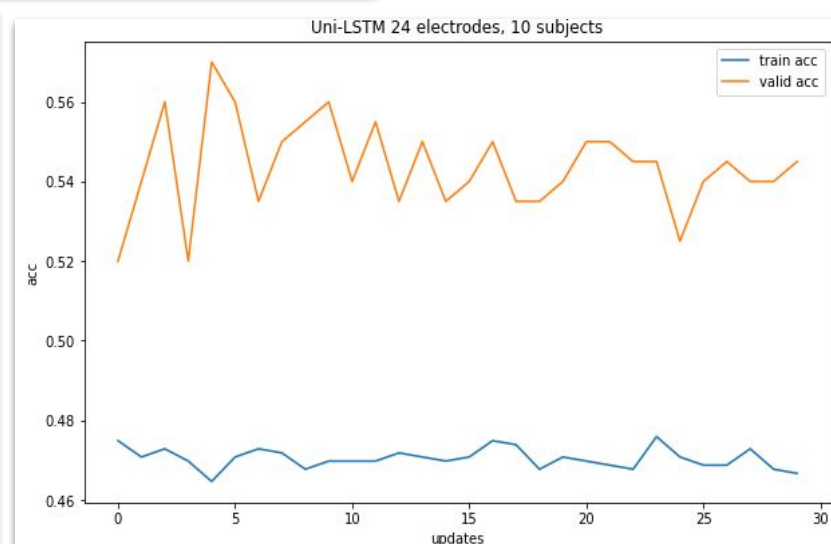
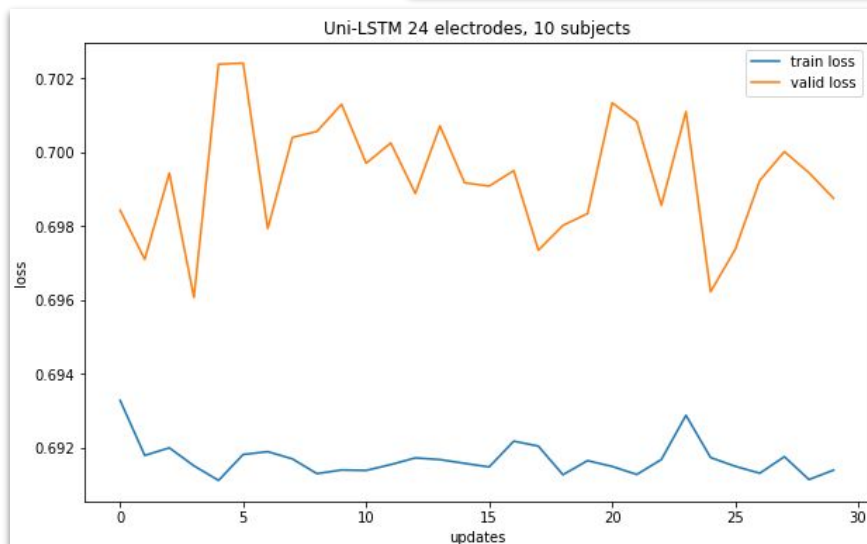
Bi-LSTM+attention(24 channels)

```
LSTM(  
  (lstm): LSTM(24, 128, batch_first=True, bidirectional=True)  
  (fc): Linear(in_features=256, out_features=1, bias=True)  
  (dropout1): Dropout(p=0.2, inplace=False)  
  (dropout2): Dropout(p=0.1, inplace=False)  
  (dropout3): Dropout(p=0.2, inplace=False)  
)
```



Uni-LSTM+attention(24 channels)

```
LSTM(  
  (lstm): LSTM(24, 256, batch_first=True, dropout=0.2)  
  (lstm2): LSTM(256, 256, batch_first=True, dropout=0.1)  
  (lstm3): LSTM(256, 256, batch_first=True, dropout=0.2)  
  (fc): Linear(in_features=256, out_features=1, bias=True)  
  (sigmoid): Sigmoid()  
)
```



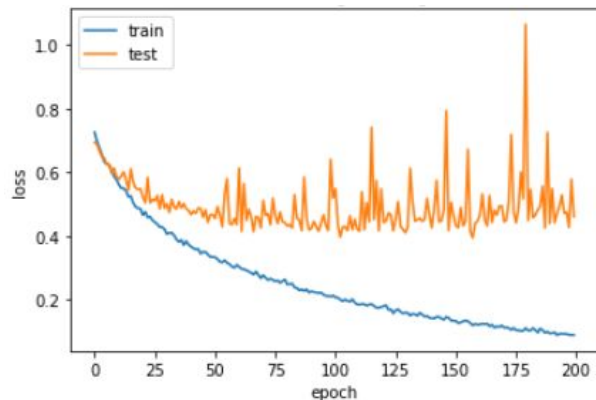
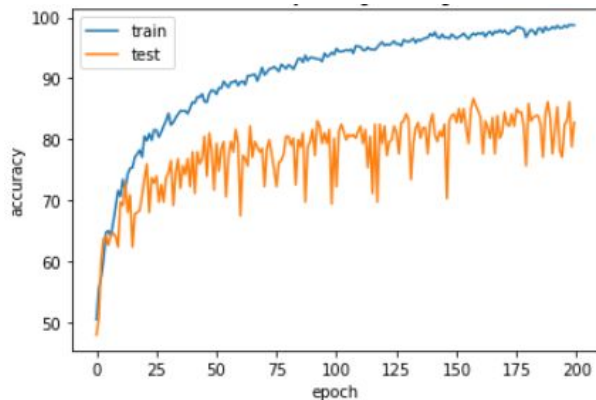
```

torch.Size([50, 1])
down tensor([ 5.5610e-07,  3.4513e-07,  3.3720e-08,  2.1897e-07, -3.0709e-07,
              1.0865e-06, -3.0785e-07,  5.8209e-07, -1.0236e-07,  6.4134e-08,
              5.7815e-09,  7.6890e-07,  5.0260e-08,  9.9454e-07, -4.1602e-07,
              1.5392e-07,  4.5101e-07,  3.7384e-08,  5.9947e-07, -1.5400e-07,
              -1.6691e-08,  2.4602e-07,  9.3956e-07,  1.8831e-07, -1.4813e-07,
              1.4902e-07,  2.2173e-07,  3.4389e-08,  1.3211e-06,  2.3461e-07,
              1.9533e-07,  1.1443e-07,  1.4213e-06,  1.7125e-07,  1.8163e-08,
              1.5846e-07,  2.5248e-07, -2.8212e-07,  1.1850e-06, -7.1664e-07,
              -2.2732e-07, -1.9711e-07, -3.8930e-07, -2.7031e-07,  3.9931e-08,
              -2.7901e-07,  2.1195e-07,  4.8985e-07,  7.1253e-08, -1.0049e-06],
            grad_fn=<SqueezeBackward1>)
Epoch [1/30], Step [1/13], Loss: 0.6931, acc 0.5400torch.Size([50, 1])
down tensor([-0.0181, -0.0182, -0.0182, -0.0182, -0.0182, -0.0181, -0.0181, -0.0182,
              -0.0181, -0.0181, -0.0182, -0.0182, -0.0181, -0.0181, -0.0181, -0.0182,
              -0.0181, -0.0181, -0.0182, -0.0181, -0.0181, -0.0182, -0.0181, -0.0182,
              -0.0181, -0.0182, -0.0181, -0.0181, -0.0181, -0.0181, -0.0182,
              -0.0182, -0.0181, -0.0182, -0.0181, -0.0181, -0.0181, -0.0181, -0.0181,
              -0.0181, -0.0181, -0.0181, -0.0181, -0.0181, -0.0181, -0.0181, -0.0181,
              -0.0181, -0.0181], grad_fn=<SqueezeBackward1>)
Epoch [1/30], Step [2/13], Loss: 0.6921, acc 0.5600torch.Size([50, 1])
down tensor([-0.0355, -0.0359, -0.0357, -0.0359, -0.0356, -0.0358, -0.0358, -0.0358,
              -0.0358, -0.0359, -0.0357, -0.0358, -0.0358, -0.0358, -0.0360, -0.0359, -0.0358,
              -0.0357, -0.0359, -0.0357, -0.0358, -0.0356, -0.0359, -0.0360, -0.0359,
              -0.0358, -0.0357, -0.0358, -0.0358, -0.0358, -0.0359, -0.0358, -0.0358,
              -0.0359, -0.0357, -0.0360, -0.0359, -0.0359, -0.0359, -0.0357, -0.0358,
              -0.0358, -0.0357, -0.0357, -0.0358, -0.0356, -0.0357, -0.0357, -0.0358,
              -0.0358, -0.0359], grad_fn=<SqueezeBackward1>)
Epoch [1/30], Step [3/13], Loss: 0.6919, acc 0.5400torch.Size([50, 1])
down tensor([-0.0526, -0.0524, -0.0524, -0.0526, -0.0526, -0.0524, -0.0523, -0.0525, -0.0524,
              -0.0523, -0.0523, -0.0526, -0.0525, -0.0524, -0.0522, -0.0524, -0.0523,
              -0.0522, -0.0525, -0.0522, -0.0524, -0.0525, -0.0525, -0.0526, -0.0528,
              -0.0524, -0.0524, -0.0527, -0.0524, -0.0526, -0.0524, -0.0523, -0.0525,
              -0.0526, -0.0523, -0.0525, -0.0523, -0.0524, -0.0523, -0.0523, -0.0524,
              -0.0524, -0.0522, -0.0525, -0.0524, -0.0524, -0.0526, -0.0526, -0.0528,
              -0.0527, -0.0525], grad_fn=<SqueezeBackward1>)

```

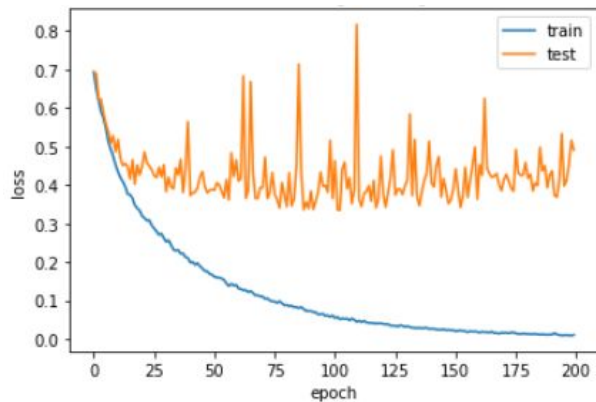
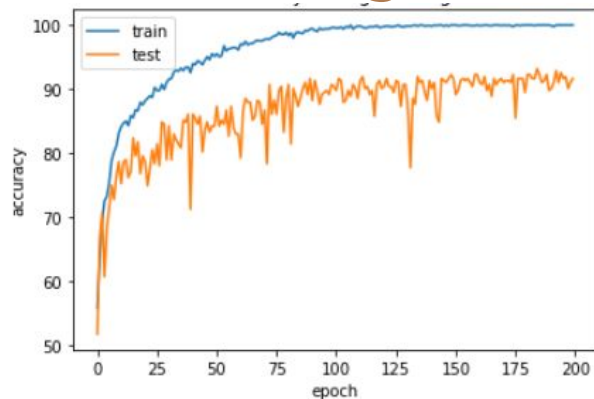
CNN 1D & 2D

Modeling - CNN 1D (2 channels)



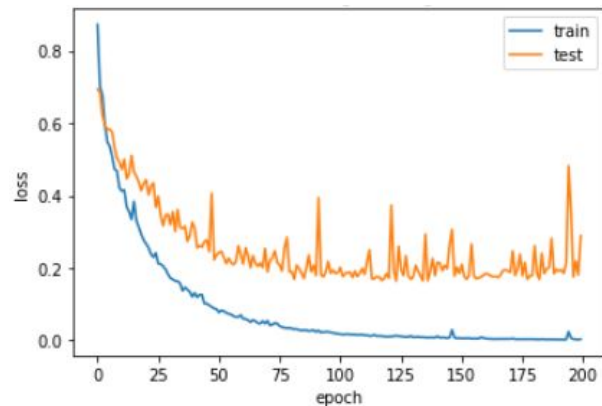
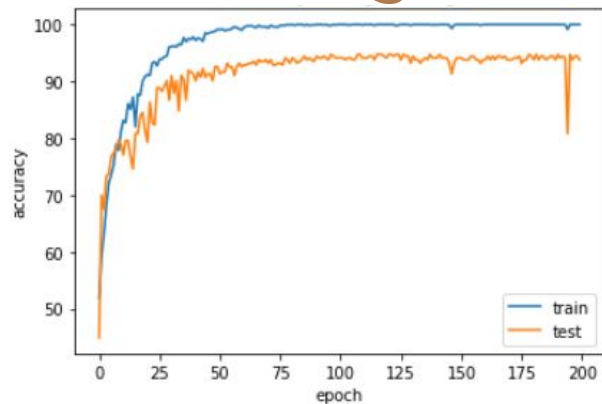
```
ConvNet(  
  (layer1): Sequential(  
    (0): Conv1d(2, 64, kernel_size=(5,), stride=(1,), padding=(2,))  
    (1): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    (2): ReLU()  
  )  
  (layer2): Sequential(  
    (0): Conv1d(64, 128, kernel_size=(5,), stride=(1,), padding=(2,))  
    (1): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    (2): ReLU()  
  )  
  (fc): Linear(in_features=82048, out_features=2, bias=True)  
  (drop_out): Dropout(p=0.5, inplace=False)  
)
```

Modeling - CNN 2D (2 channels)



```
ConvNet(  
  (layer1): Sequential(  
    (0): Conv2d(1, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    (2): ReLU()  
  )  
  (layer2): Sequential(  
    (0): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    (2): ReLU()  
  )  
  (fc): Linear(in_features=164096, out_features=2, bias=True)  
  (drop_out): Dropout(p=0.5, inplace=False)  
)
```

Modeling - CNN 2D (64 channels)

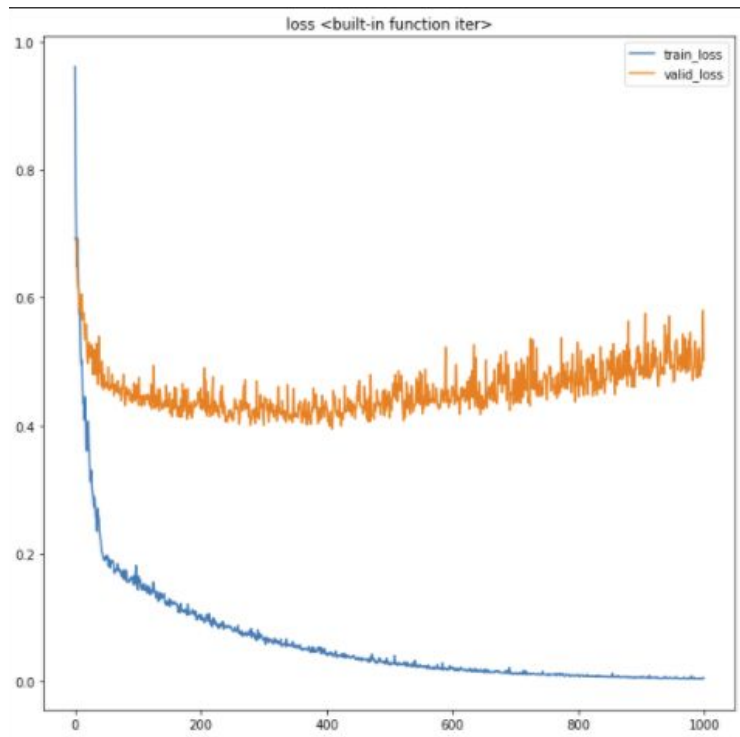


```
ConvNet(  
  (layer1): Sequential(  
    (0): Conv2d(1, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    (2): ReLU()  
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
  )  
  (layer2): Sequential(  
    (0): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    (2): ReLU()  
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
  )  
  (fc): Linear(in_features=327680, out_features=2, bias=True)  
  (drop_out): Dropout(p=0.5, inplace=False)  
)
```

CNN 2D with ICA

Adjust chanal

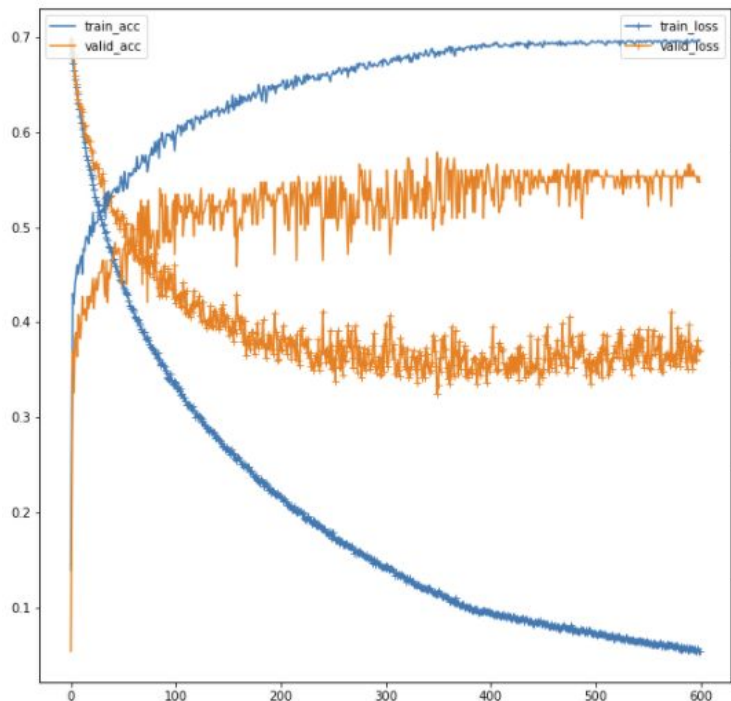
Modeling - CNN 2D (64 channels)



```
ConvNet(  
  (layer1): Sequential(  
    (0): Conv2d(2, 64, kernel size=(5, 5), stride=(1, 1), padding=(2, 2))  
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,  
track running stats=True)  
    (2): ReLU()  
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,  
ceil_mode=False)  
  )  
  (fc): Linear(in features=327680, out features=2, bias=True)  
  (drop_out): Dropout(p=0.5, inplace=False)  
)
```

Epoch 1000/1000, Tr Loss: 0.0043, Tr Acc: 100.0000, Val
Loss: 0.5019, Val Acc: 87.7778

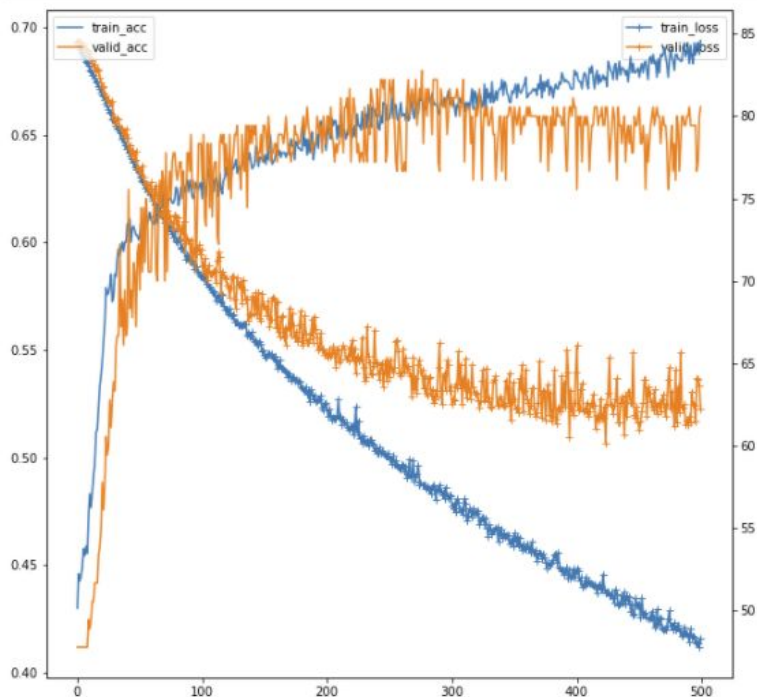
Modeling - CNN 2D (64 channels)



```
ConvNet(  
  (layer1): Sequential(  
    (0): Conv2d(1, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    (2): ReLU()  
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
  )  
  (layer2): Sequential(  
    (0): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    (2): ReLU()  
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
  )  
  (fc): Linear(in_features=655360, out_features=2, bias=True)  
  (drop_out): Dropout(p=0.5, inplace=False)  
)  
  
if loss_fix < 0.1 :  
    optimizer = optim.Adam(net.parameters(), lr=0.000005)
```

Epoch 600/600, Tr Loss: 0.0545, Tr Acc: 99.7222, Val Loss: 0.3712, Val Acc: 86.6667

Modeling - CNN 2D (7 channels)



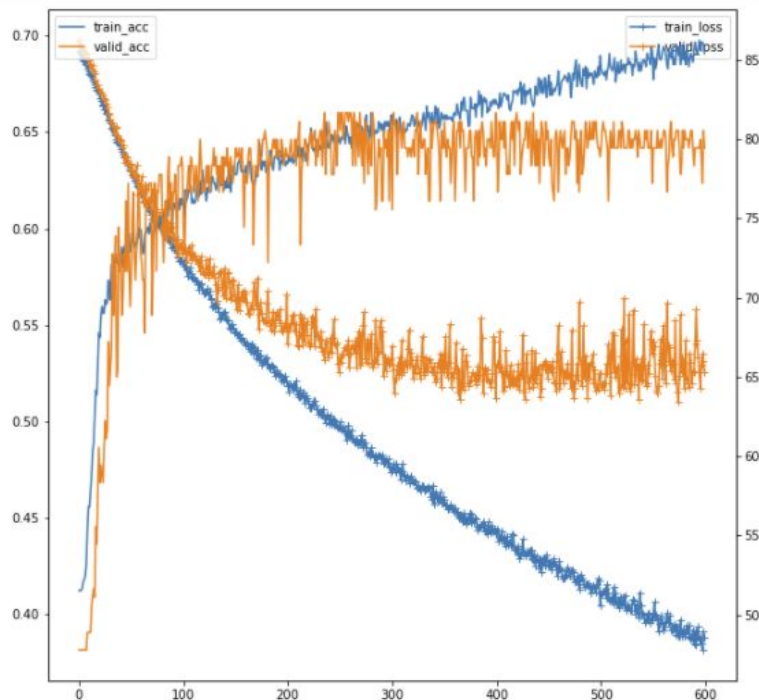
'C5', 'C3', 'C1', 'Cz', 'C2', 'C4', 'C6'

```
ConvNet(  
  (layer1): Sequential(  
    (0): Conv2d(1, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    (2): ReLU()  
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
  )  
  (fc): Linear(in_features=61440, out_features=2, bias=True)  
  (drop_out): Dropout(p=0.5, inplace=False)  
)
```

```
if loss_fix < 0.1 :  
    optimizer = optim.Adam(net.parameters(), lr=0.000005)
```

Epoch 500/500, Tr Loss: 0.4160, Tr Acc: 84.5833, Val Loss: 0.5225, Val Acc: 80.5556

Modeling - CNN 2D (2 channels)



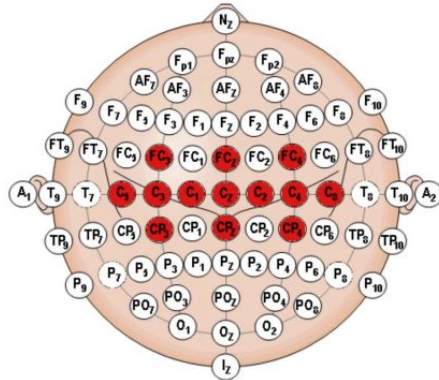
'C5', 'C6'

```
ConvNet(  
  (layer1): Sequential(  
    (0): Conv2d(1, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))  
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
    (2): ReLU()  
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)  
  )  
  (fc): Linear(in_features=61440, out_features=2, bias=True)  
  (drop_out): Dropout(p=0.5, inplace=False)  
)
```

```
if loss_fix < 0.1 :  
    optimizer = optim.Adam(net.parameters(), lr=0.000005)
```

Epoch 600/600, Tr Loss: 0.3881, Tr Acc: 85.4167, Val Loss: 0.5256, Val Acc: 79.4444

Modeling - CNN 2D (7 channels)



'C5', 'C3', 'C1', 'Cz', 'C2', 'C4', 'C6'

```
ConvNet(
  (layer1): Sequential(
    (0): Conv2d(1, 64, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (layer2): Sequential(
    (0): Conv2d(64, 128, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc): Linear(in_features=655360, out_features=2, bias=True)
  (drop_out): Dropout(p=0.5, inplace=False)
)

if loss_fix < 0.1 :
    optimizer = optim.Adam(net.parameters(), lr=0.000005)
```

Epoch 600/600, Tr Loss: 0.0545, Tr Acc: 99.7222, Val Loss: 0.3712, Val Acc: 86.6667

Conclusion

