Neural Networks & Fuzzy Logic - BITS F312

Under the guidance of:

Dr. Surekha Bhanot

Dr. Bijoy Krishna Mukherjee

Mentor:

Jivat Neet Kaur(2017A7PS0050P)

Design project done by:

Venkata Sai Kiran Piratla - 2016A3PS0178P

Kalimi Venkata Yashwanth Kumar Reddy - 2017A3PS0221P

Yaswanth Kumar Rayapati - 2017A7PS0038P

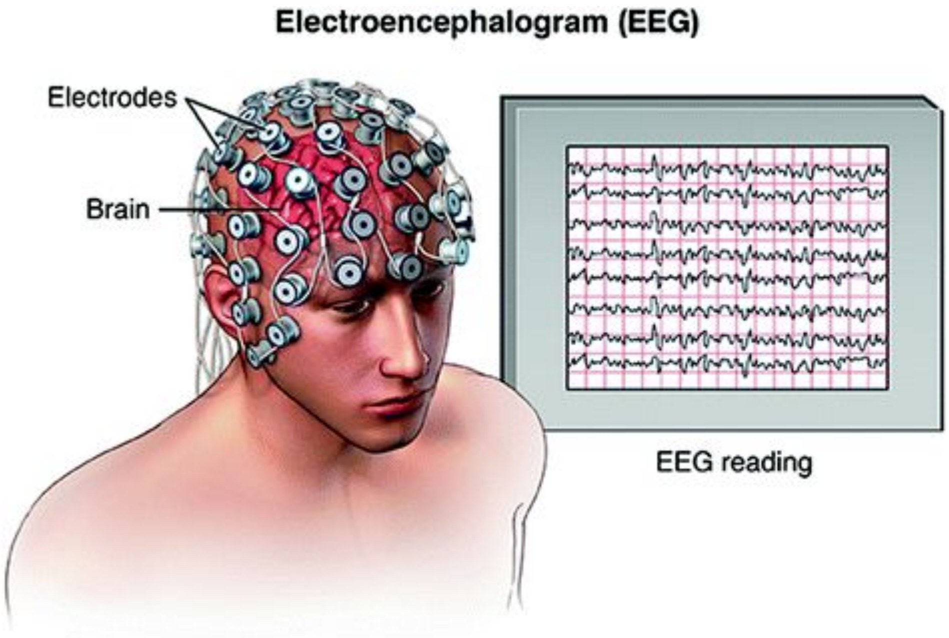
Introduction

- Title : Deep Learning EEG Response Representation for Brain Computer Interface
- Authors: LIU Jingwei, CHENG Yin, ZHANG Weidong
- Implemented Neural Networks to classify different imagined motor tasks

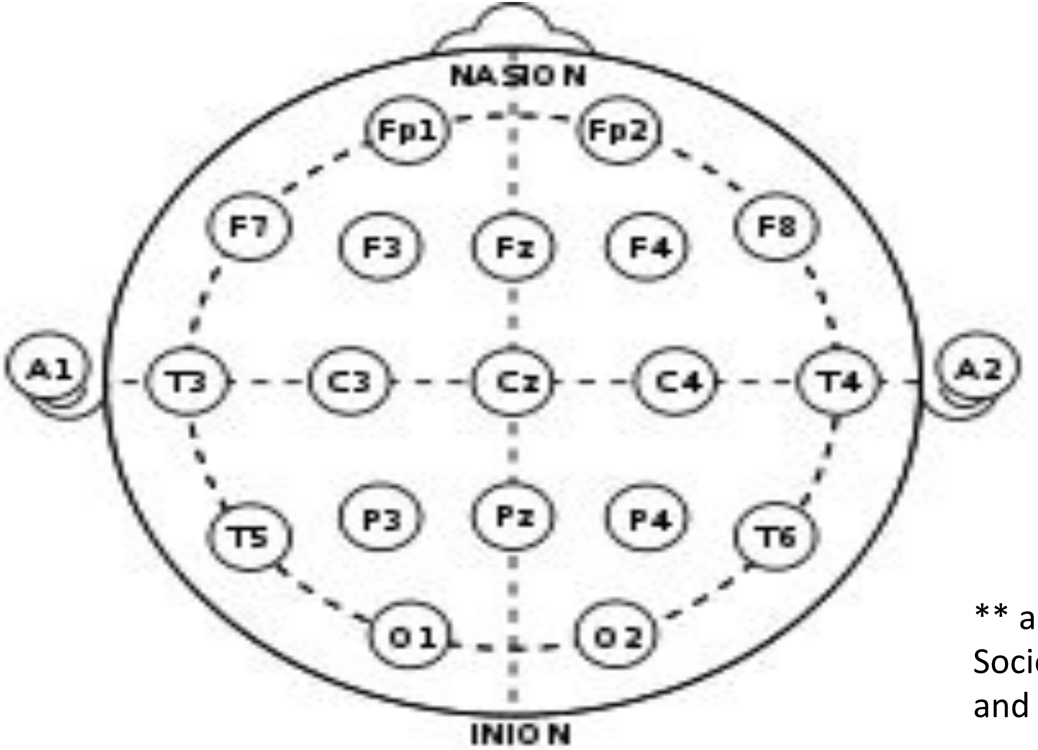
Electroencephalography (EEG)

- Research indicates that it is primarily the synchronized activity of pyramidal neurons in cortical brain regions which can be measured from the outside

- Applied first to humans in the 1920s by German neurologist Hans Berger



Electrode placement



** as per International Federation of Societies for Electroencephalography and Clinical Neurophysiology

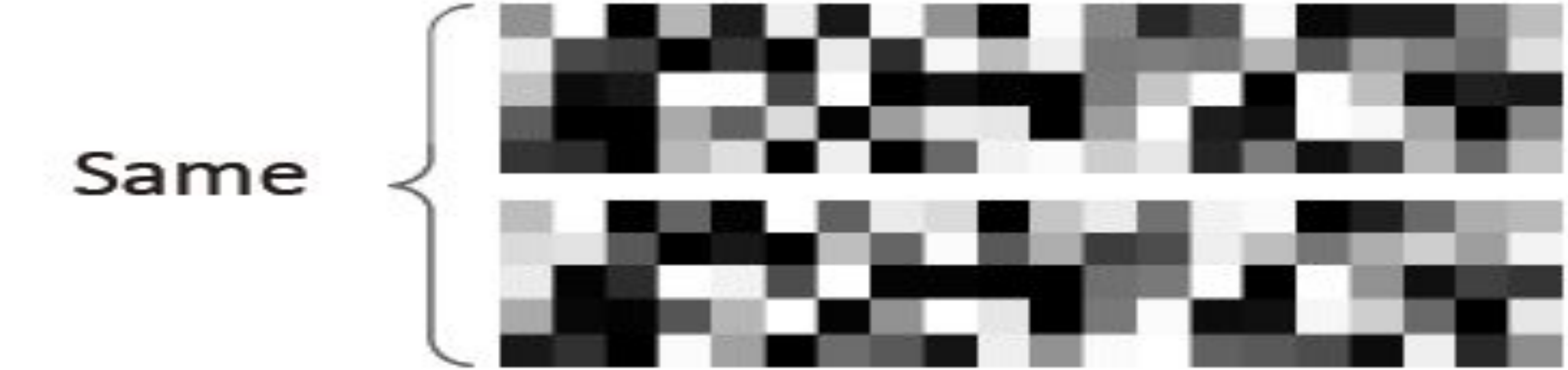
The subject was asked to do the following movements, and the corresponding EEG signals of the subject was recorded

(i) Imagined left hand backward movement (Fig.(a))
(ii) Imagined left hand forward movement (Fig.(b))
(iii) Imagined right hand backward movement (Fig.(c))
(iv) Imagined right hand forward movement (Fig.(d))

(a) Task 1. (b) Task 2. (c) Task 3. (d) Task 4.

Comparision among tasks

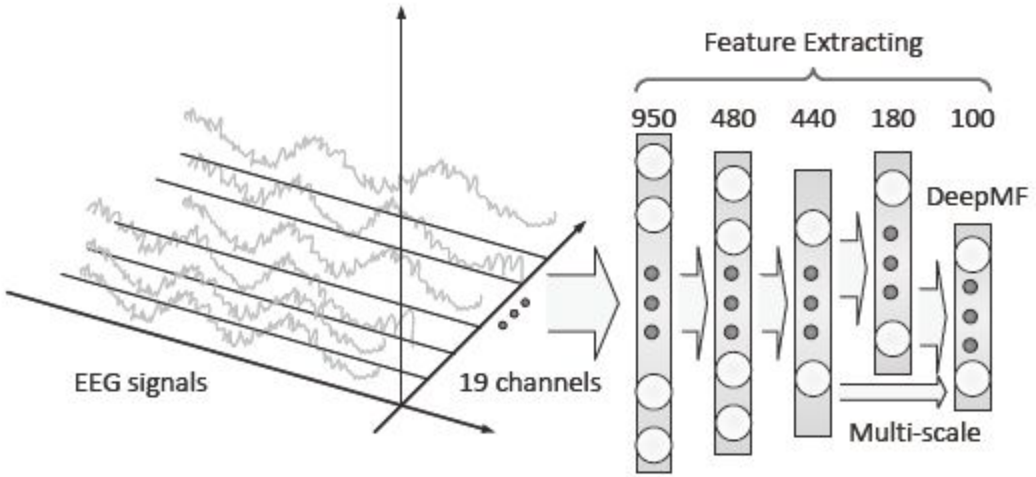
- The feature values are non-negative since they are processed by ReLUs, the brighter squares indicates higher value of activation.
- The extracted features are different between diverse tasks, but they are much alike within the same task even for different random trials.



(a) Both dissimilar random trails for task1.

(b) Random trails for task3 and task4.

- Time series signals are extracted from previously mentioned 19 channels and forwarded to ConvNets in this fashion.



Data Extraction

- Data taken from https://sites.google.com/site/projectbci/
- Data is in .csv format containing signals from 19 electrodes
- Each csv contains signals for a particular action among the following:
  - Imagined left hand backward movement
  - Imagined left hand forward movement
  - Imagined right hand backward movement
  - Imagined right hand forward movement
- The columns in csv correspond to electrodes FP1, FP2, F3, F4, C3, C4, P3, P4, O1, O2, F7, F8, T3, T4, T5, T6, Fz, Cz, and Pz respectively.

Pre-Processing

Mainly consists two steps

Scaling

Single trial extraction

Scaling

- Samples from each column were normalized as z-scores

	A
1	***Region***
2	*East*
3	*West*
4	*North*
5	*South*
6	*MidWest*
7	*Far Off*
8	*Central*

A	B
Sales (May)	Sales (June)
45	61
21	21
25	45
52	81
22	52
54	24
62	11

A	B
Sales (July)	Sales (August)
24	44
55	21
31	38
71	91
6	14
12	41
51	3

z-score

Data point

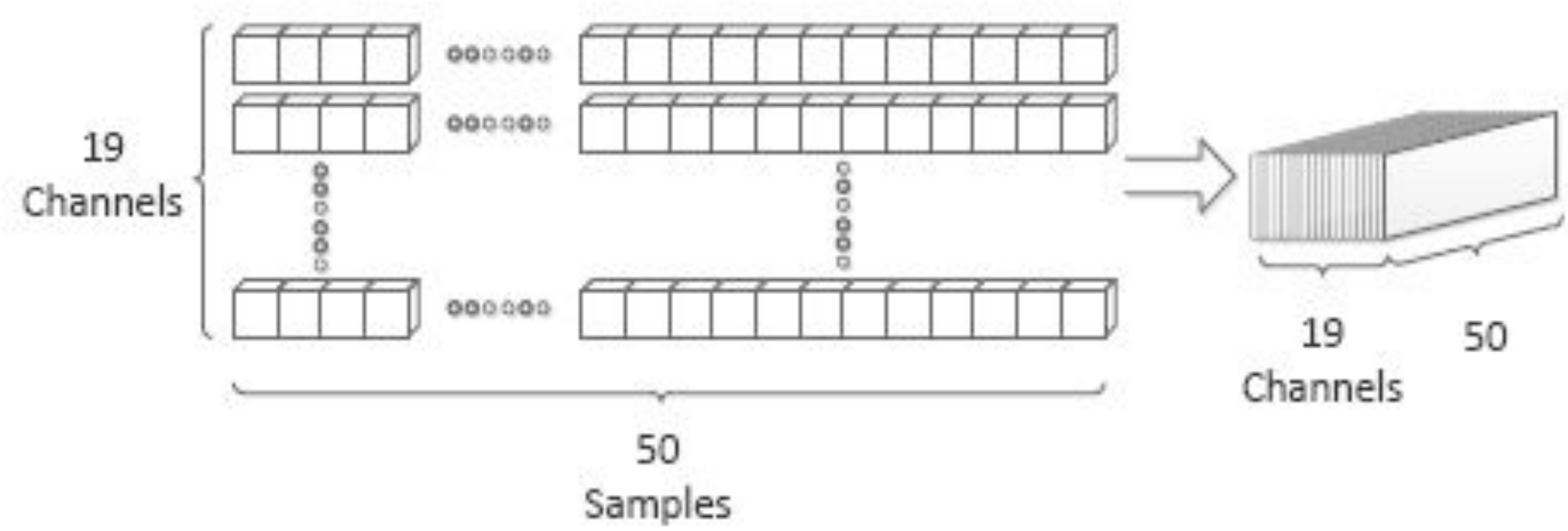
Mean

$$z = \frac{(x - \mu)}{\sigma}$$

Standard deviation

Single trial extraction

- In .csv files each column represents a channel whereas row represents a single trail point
- 50 such single trail points are bundled together to form a trial
- We bundled 5 trials as a minibatch for input, and one epoch stands for input the whole training dataset for a round

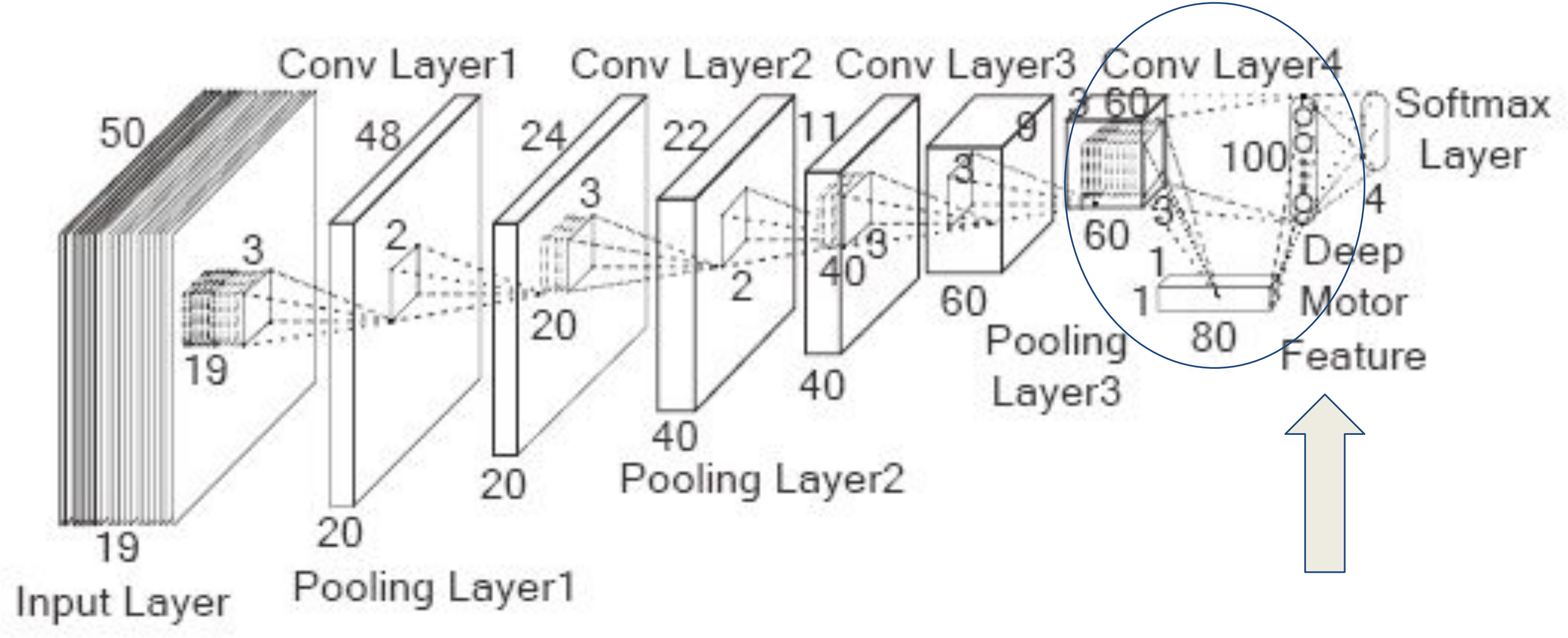

19
Channels
50
Samples
19
Channels
50

Model Architecture

Project consists of 3 CNN models

1. Multi scale CNN
2. Single scale CNN
3. Shallow CNN

Multi-scale CNN



- The DeepMF layer is fully connected to both the 3rd pooling layer and 4th convolutional layer.

- The ConvNets will be able to learn the multi-scale features through this double fully connected structure

- This is crucial to learn more effective features since this design provides different scales of receptive fields to the last softmax layer for identification.

configuration

Layer	Layer type	Kernel shape	Output shape
0	Input	-	[5, 19, 1, 50]
1	Convolutional	[20, 19, 1, 3]	[5, 20, 1, 48]
2	Pooling	[1, 2]	[5, 20, 1, 24]
3	Convolutional	[40, 20, 1, 3]	[5, 40, 1, 22]
4	Pooling	[1, 2]	[5, 40, 1, 11]
5	Convolutional	[60, 40, 1, 3]	[5, 60, 1, 9]
6	Pooling	[1, 3]	[5, 60, 1, 3]
7	Convolutional	[80, 60, 1, 3]	[100, 80, 1, 1]
8	DeepMF	-	[100]
9	Softmax	-	[4]

Activation functions, weights & loss functions

- It is mentioned in paper to use Relu and tanh activation functions alternatively
- Initialized biases to be 0
- Weights at each layer with commonly used heuristic: glorot uniform
- The loss is computed through stochastic gradient descent(SGD) algorithm

Glorot uniform function and pooling

- Weights are initialized with the following:

$$W_{ij} \sim U\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$$

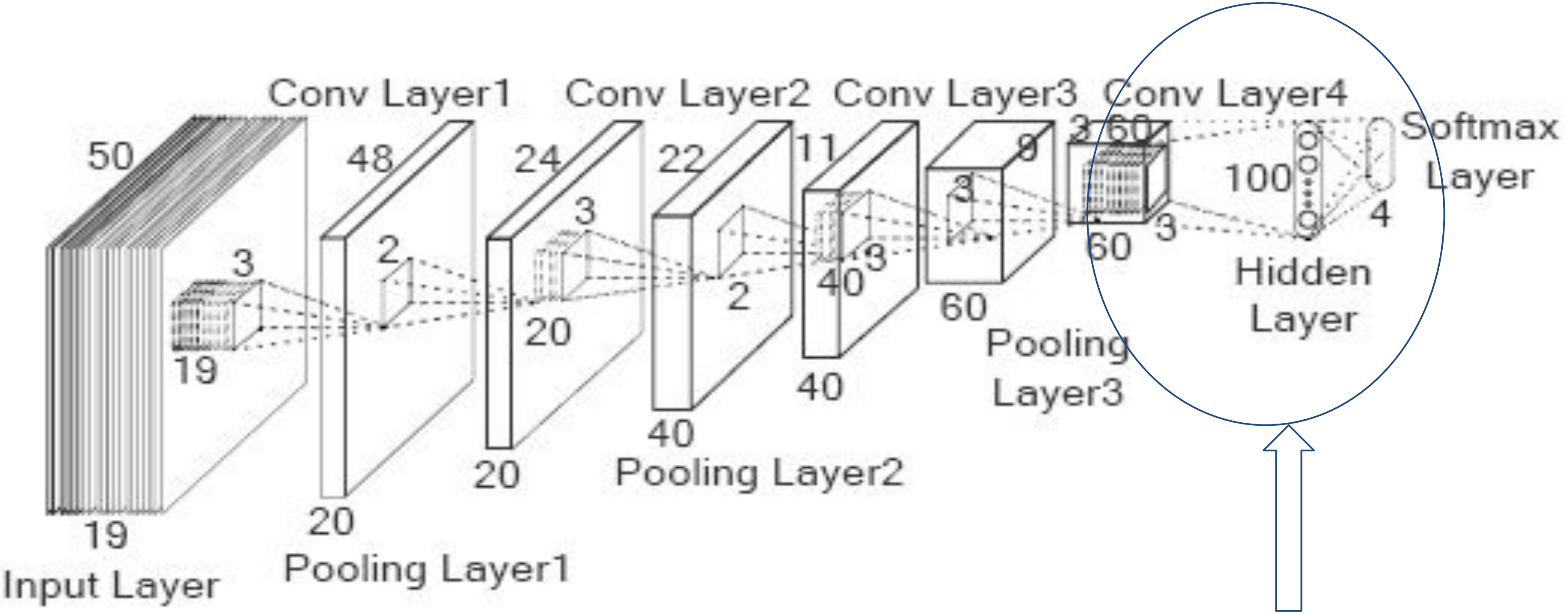
  - where U [−a, a] is the uniform distribution in the interval (−a, a) and n is the size of previous layer.

- Used max pool layers for pooling

$$y^i_{j,k} = \max_{0\leq m,n\leq s} \left\{x^i_{j\cdot s+m,k\cdot s+n}\right\}$$

Single scale CNN

- Everything is same as multi-scale CNN except at DeepMF layer
- Here, output of the third layer is directly connected to hidden layer(DeepMF).
- In this case, the single stage will not be able to provide the diverse scales of receptive fields to the softmax classifier behind.
- This configuration will reduce the ability of networks to learn more effective features through the information that skips the layer.


Conv Layer1
Conv Layer2
Conv Layer3
Conv Layer4
50
48
24
22
11
9
3 60
Softmax
Layer
100
4
3
2
3
2
3
3
3
60
19
20
40
60
Hidden
Layer
Pooling
Layer3
40
20
40
Pooling Layer2
19
20
Input Layer
Pooling Layer1

Single-scale CNN architecture

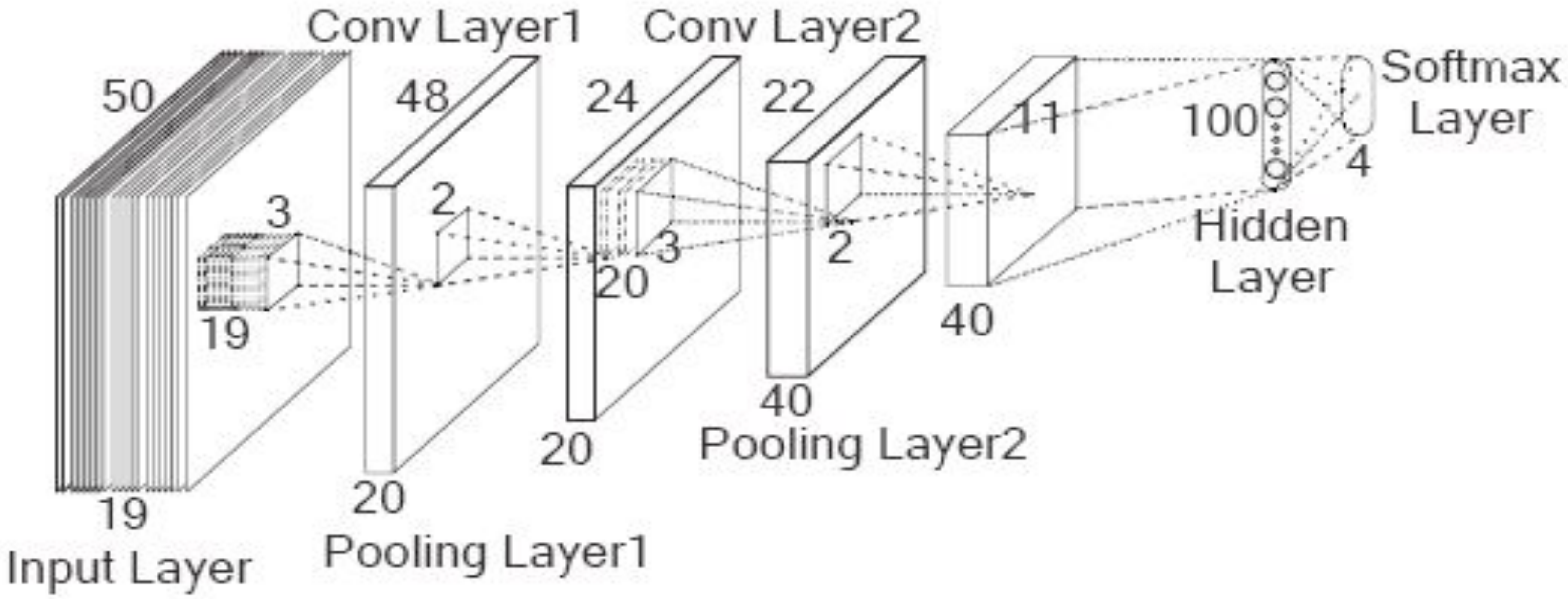
Table 2: Configuration of single-scale deep CNNs

Layer	Layer type	Kernel shape	Output shape
0	Input	-	[5, 19, 1, 50]
1	Convolutional	[20, 19, 1, 3]	[5, 20, 1, 48]
2	Pooling	[1, 2]	[5, 20, 1, 24]
3	Convolutional	[40, 20, 1, 3]	[5, 40, 1, 22]
4	Pooling	[1, 2]	[5, 40, 1, 11]
5	Convolutional	[60, 40, 1, 3]	[5, 60, 1, 9]
6	Pooling	[1, 3]	[5, 60, 1, 3]
7	Hidden	-	[100]
8	Softmax	-	[4]

Shallow CNN

- In the configuration of convolutional neural networks, the depth also play an important role in better accuracy performance

- We build a shallow CNN containing only two convolutional and pooling layers within network.

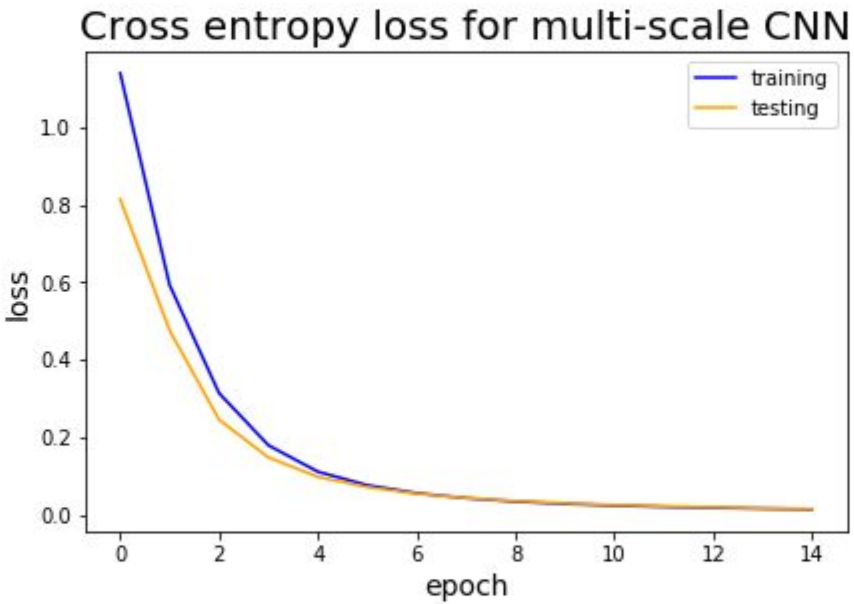

Conv Layer1
Conv Layer2
50
48
24
22
11
100
Softmax
Layer
4
3
2
3
2
20
19
Hidden
Layer
40
40
20
20
19
Input Layer
Pooling Layer1
Pooling Layer2

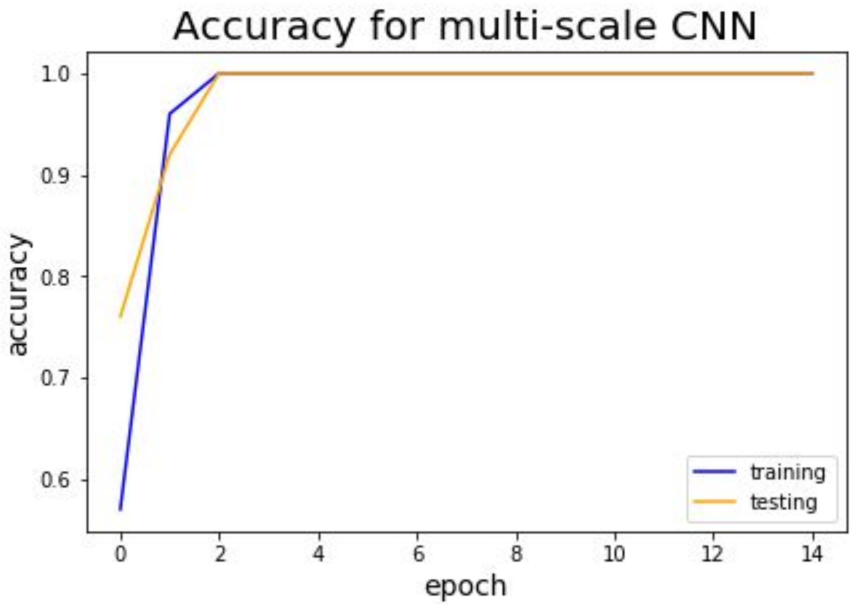
Shallow CNN architecture

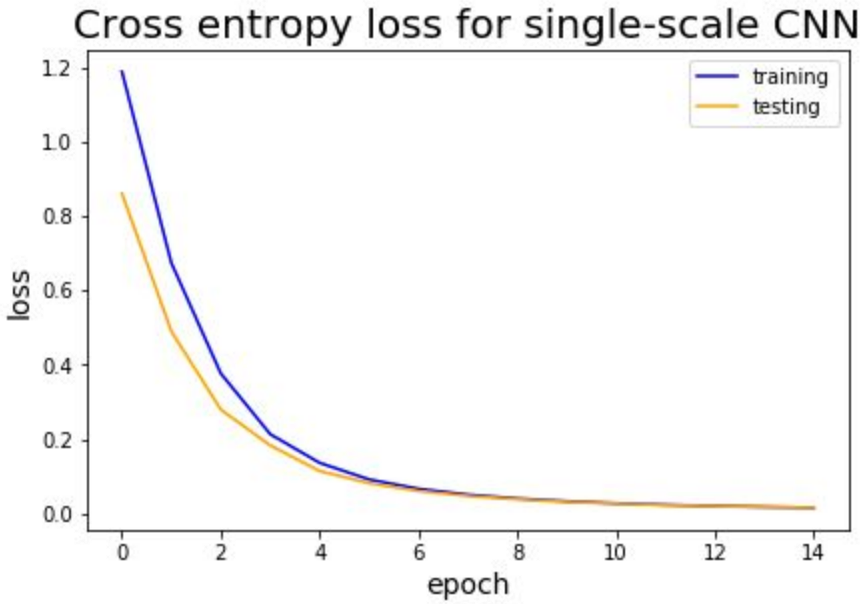
Layer	Layer type	Kernel shape	Output shape
0.	Input	------	[5, 19, 1, 50]
1.	Convolutional	[20, 19, 1, 3]	[5, 20, 1, 48]
2.	Pooling	[1, 2]	[5, 20, 1, 24]
3.	Convolutional	[40, 20, 1, 3]	[5, 40, 1, 22]
4.	Pooling	[1, 2]	[5, 40, 1, 11]
5.	Hidden	------	[100]
6.	Softmax	------	[4]

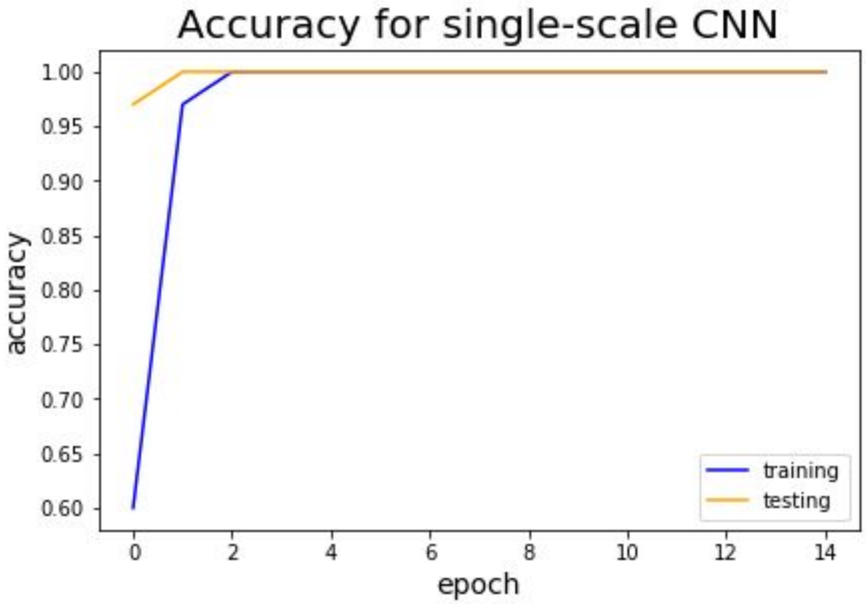
Observations

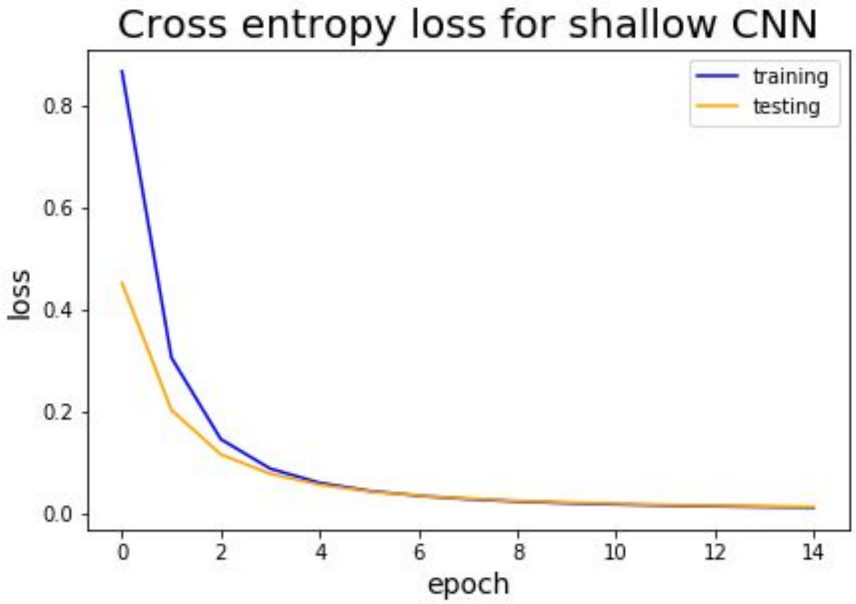
- Cross entropy loss and accuracy are plotted from model history



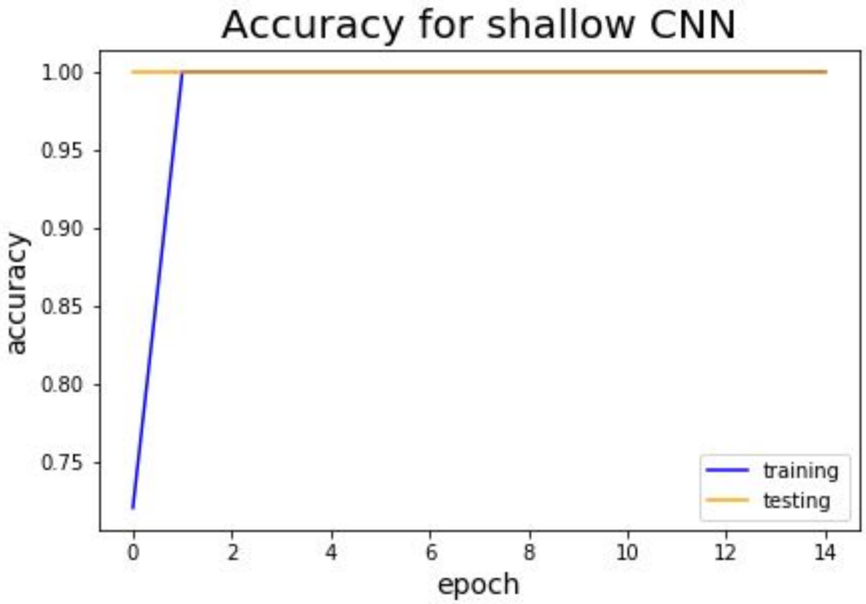



Cross entropy loss for single-scale CNN
loss
epoch
training
testing
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
2
4
6
8
10
12
14


Accuracy for single-scale CNN
accuracy
epoch
training
testing
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0
2
4
6
8
10
12
14



Cross entropy loss for shallow CNN
training
testing
loss
epoch
0.0
0.2
0.4
0.6
0.8
0
2
4
6
8
10
12
14



Accuracy for shallow CNN
training
testing
accuracy
epoch
0.75
0.80
0.85
0.90
0.95
1.00
0
2
4
6
8
10
12
14

	Multi Scale CNN	Single Scale CNN	Shallow CNN
Epochs to reach maximum accuracy	3	3	2
Time taken to reach maximum accuray	5 ms/step	3.923 ms/step	2.817 ms/step
Trainable parameters	51,844	29,364	48,104

```
Epoch 2/15
  5/100 [>.............................] - ETA: 0s - loss: 0.5575 - accuracy: 1.0000Time elapsed: 1.51547
```

multi-scale

```
Epoch 2/15
  5/100 [>.............................] - ETA: 0s - loss: 0.7635 - accuracy: 1.0000Time elapsed: 1.484162
```

single-scale

```
Epoch 2/15
  5/100 [>.............................] - ETA: 0s - loss: 0.3607 - accuracy: 1.0000Time elapsed: 1.206473
```

shallow

Git documentation

- Entire code is uploaded to GitHub.
- https://github.com/white-fusion/EEG_for_BCI
- Follow the instructions given in the repository to run the Jupyter notebooks from your local machine.

Thank you for your attention!

Any Questions?