Deep Learning Lab #3 (Fall 2021)

Lab Objective:

In this assignment, you will need to implement simple EEG classification models which are EEGNet, DeepConvNet [1] with BCI competition dataset. Additionally, you need to try different kinds of activation function including ReLU, Leaky ReLU, ELU.

Rules:

(1) This assignment should be done individually. Plagiarism is strictly prohibited.

(2) You can only use Numpy and other Python standard library. Only PyTorch are allowed in this lab.

(3) You should add comments throughout your implementation for easy understanding.

(4) Write a report in the end of the Jupyter Notebook to detail your procedures and discussions.

Submission:

(1) Please write your code on Jupyter notebook.

(2) Pack the .ipynb to .zip, and submit .zip to E3. Please name as "Lab3_YourStudentID.zip".

Deadline: 2021/12/13 (Mon.) 23:55

Requirements:

(1) Implement the EEGNet, DeepConvNet with three kinds of activation function including ReLU, Leaky ReLU, ELU.

(2) In the experiment results, you have to show the highest accuracy (not loss) of two architectures with three kinds of activation functions.

(3) To visualize the accuracy trend, you need to plot each epoch accuracy (not loss) during training phase and testing phase.

Descriptions:

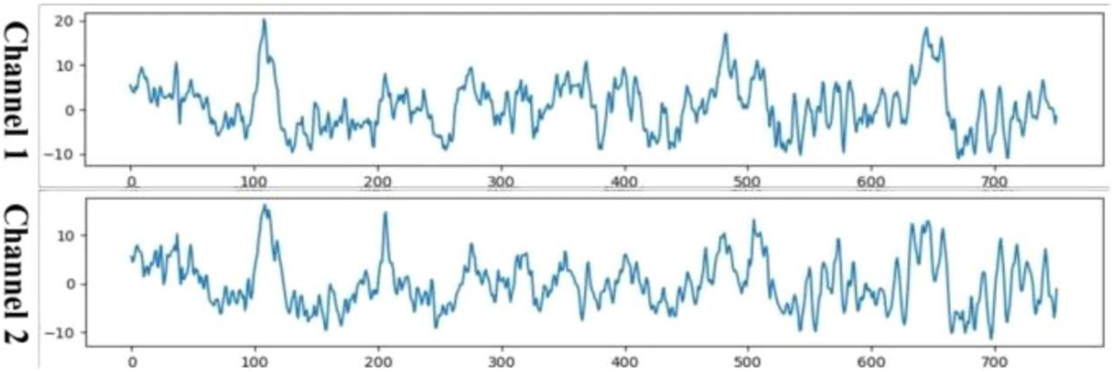
(1) Datasets

BCI Competition III - IIIb Cued motor imagery with online feedback (non-stationary classifier) with 2 classes (left hand, right hand) from 3 subjects [2 classes, 2 bipolar EEG channels]

Reference: http://www.bbci.de/competition/iii/desc_IIIb.pdf

(2) Prepare Data

The training data and testing data have been preprocessed and named [S4b_train.npz, X11b_train.npz] and [S4b_test.npz, X11b_test.npz] respectively. Please download the preprocessed data and put it in the same folder. To read the preprocessed data, refer to the "dataloader.py".



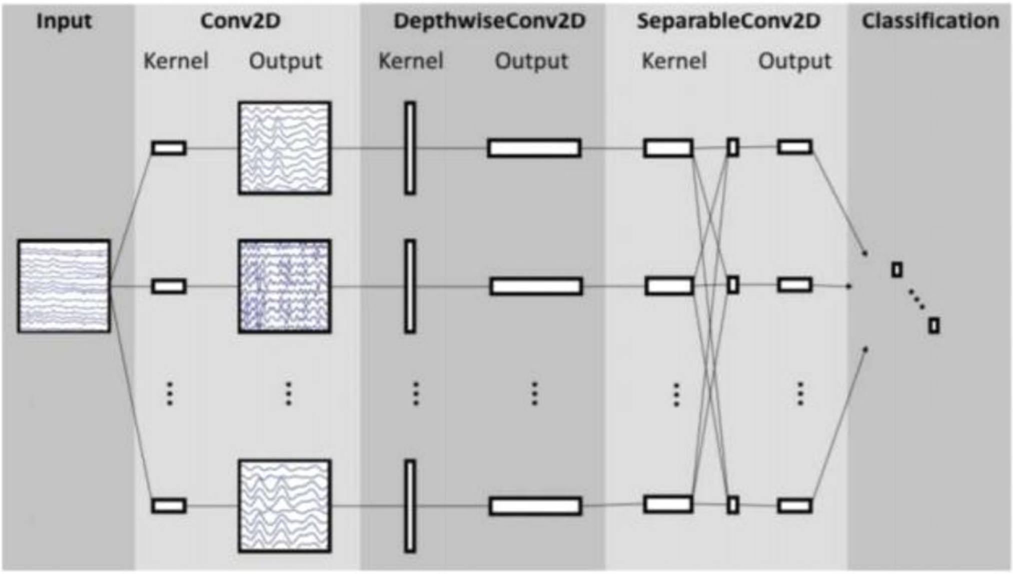
```
from dataloader import read_bci_data
train_x, train_y, test_x, test_y = read_bci_data()
```

(3) Model Architecture

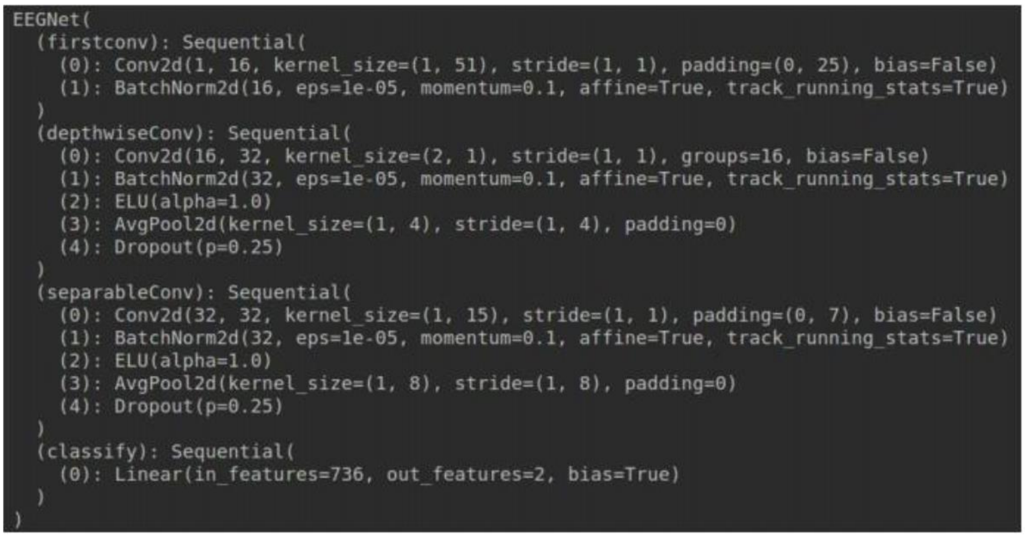
You need to implement simple EEG classification models which are EEGNet and DeepConvNet.

I. EEGNet:

Overall visualization of the EEGNet architecture



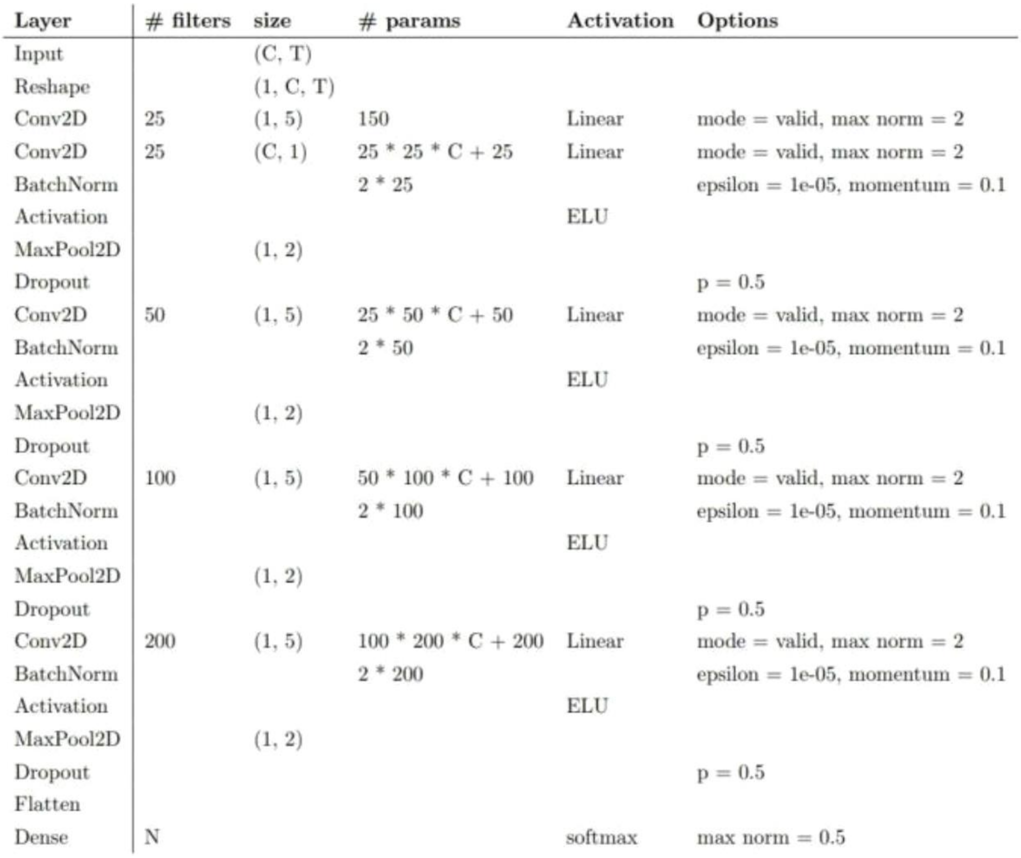
EEGNet implementation details



```
EEGNet(
  (firstconv): Sequential(
    (0): Conv2d(1, 16, kernel_size=(1, 51), stride=(1, 1), padding=(0, 25), bias=False)
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (depthwiseConv): Sequential(
    (0): Conv2d(16, 32, kernel_size=(2, 1), stride=(1, 1), groups=16, bias=False)
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): AvgPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0)
    (4): Dropout(p=0.25)
  )
  (separableConv): Sequential(
    (0): Conv2d(32, 32, kernel_size=(1, 15), stride=(1, 1), padding=(0, 7), bias=False)
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): AvgPool2d(kernel_size=(1, 8), stride=(1, 8), padding=0)
    (4): Dropout(p=0.25)
  )
  (classify): Sequential(
    (0): Linear(in_features=736, out_features=2, bias=True)
  )
)
```

II. DeepConvNet:

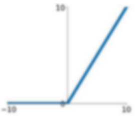
You need to implement the DeepConvNet architecture by using the following table, where $C = 2,\ T = 750$ and $N = 2$. **The max norm term is ignorable.**



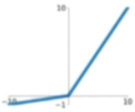
Layer	# filters	size	# params	Activation	Options
Input		(C, T)			
Reshape		(1, C, T)			
Conv2D	25	(1, 5)	150	Linear	mode = valid, max norm = 2
Conv2D	25	(C, 1)	25 * 25 * C + 25	Linear	mode = valid, max norm = 2
BatchNorm			2 * 25		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Conv2D	50	(1, 5)	25 * 50 * C + 50	Linear	mode = valid, max norm = 2
BatchNorm			2 * 50		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Conv2D	100	(1, 5)	50 * 100 * C + 100	Linear	mode = valid, max norm = 2
BatchNorm			2 * 100		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Conv2D	200	(1, 5)	100 * 200 * C + 200	Linear	mode = valid, max norm = 2
BatchNorm			2 * 200		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Flatten					
Dense	N			softmax	max norm = 0.5

(4) Activation Functions

ReLU
$\max(0, x)$

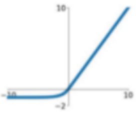


Leaky ReLU
$\max(0.1x, x)$



, By default, the negative slope = 0.01

ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



, The α value for the ELU formulation. Default: 1.0

(5) Hyper Parameters

I. Batch size= 64

II. Learning rate = 1e-2

III. Epochs = 300

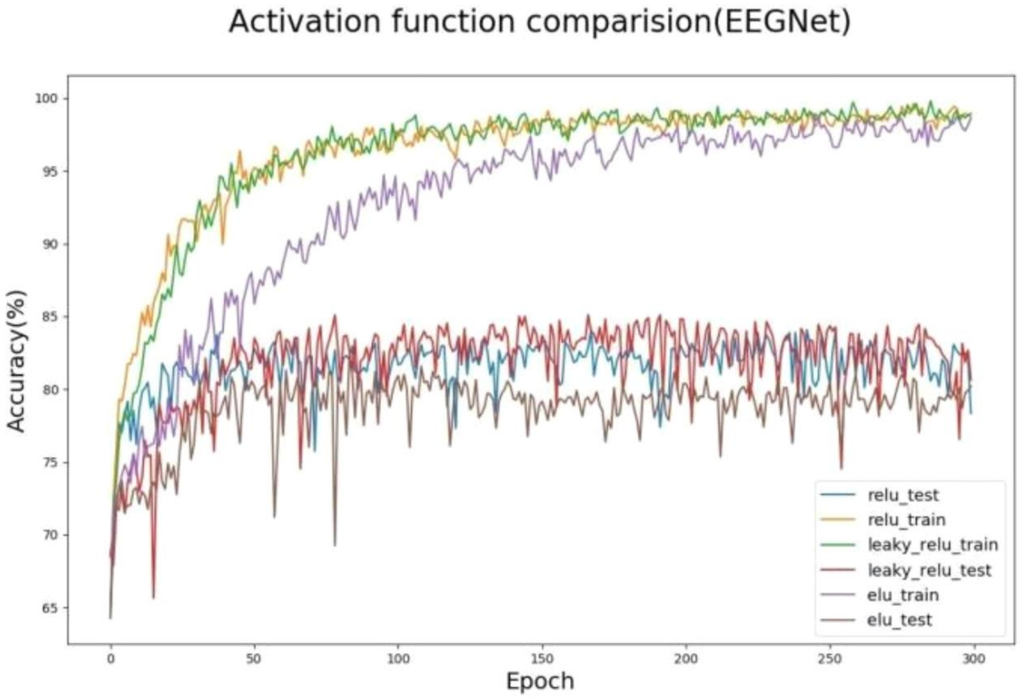
IV. Optimizer: Adam

V. Loss function: torch.nn.CrossEntropyLoss()

You can adjust the hyper-parameters according to your own ideas.

(6) Result comparison

In this part, you can use the matplotlib library to draw the graph. The comparison figure should like the example as below (EEGNet).



Assignment Evaluation:

(1) Code & model performances (60%)

(2) Report (40%)

Reference:

[1] EEGNet: A Compact Convolutional Neural Network for EEG-based Brain-Computer Interfaces

Please contact TA if you have any questions.