Lab2 : EEG classification

Lab Objective:

In this lab, 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.

Important Date:

1. Experiment Report Submission Deadline: 4/11 (Thu) 12:00
2. Demo date: 4/11 (Thu)

Turn in:

1. Experiment Report (.pdf)
2. Source code

Notice: zip all files in one file and name it like「DLP_LAB2_your studentID_name.zip」, ex:「DLP_LAB2_0656608_莊祐銓.zip」

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.

Dataset:

BCI Competition III - IIIb

Cued motor imagery with online feedback (non-stationary classifier) with 2 classes (left hand, right hand) from 3 subjects

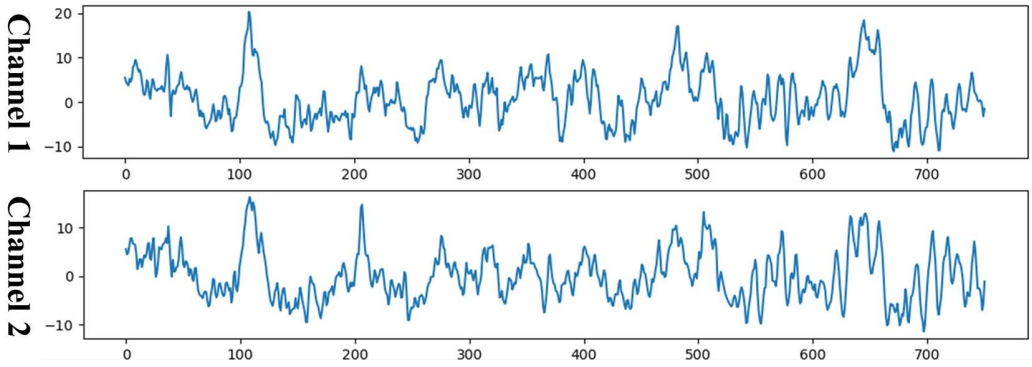
[2 classes, 2 bipolar EEG channels]

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

Implementation Details:

- **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"**.

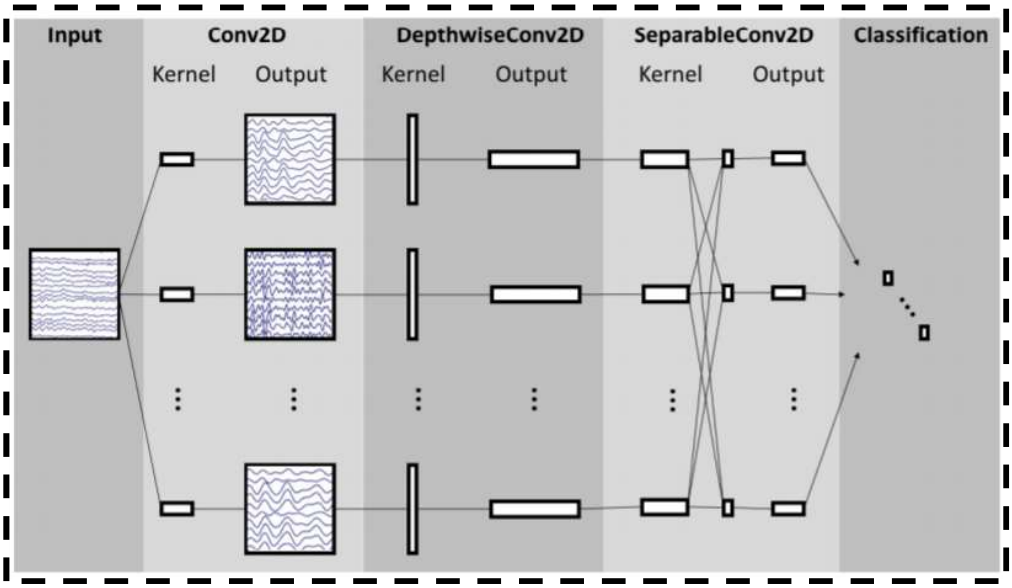


- **Model Architecture**

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

  EEGNet:

  Overall visualization of the EEGNet architecture



  Reference: Depthwise Separable Convolution
  https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

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)
  )
)
```

DeepConvNet:

You need to implement the DeepConvNet architecture by using the following table, where C = 2, T = 750 and N = 2. **The <u>max norm</u> 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

- **Activation Functions**

$$\mathrm{ReLU}(x) = \max(0, x)$$

$$\mathrm{LeakyRELU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \text{negative\_slope} \times x, & \text{otherwise} \end{cases}$$

By default, the negative slope = 0.01

$$\mathrm{ELU}(x) = \max(0, x) + \min(0, \alpha * (\exp(x) - 1))$$

The α value for the ELU formulation. Default: 1.0

Reference:
https://medium.com/tinymind/a-practical-guide-to-relu-b83ca804f1f7
https://pytorch.org/docs/stable/nn.html

In the PyTorch framework, it is easy to implement the activation function.
Just typing the following code!

```
nn.LeakyReLU(),
nn.ReLU(),
nn.ELU(),
```

- **Hyper Parameters**

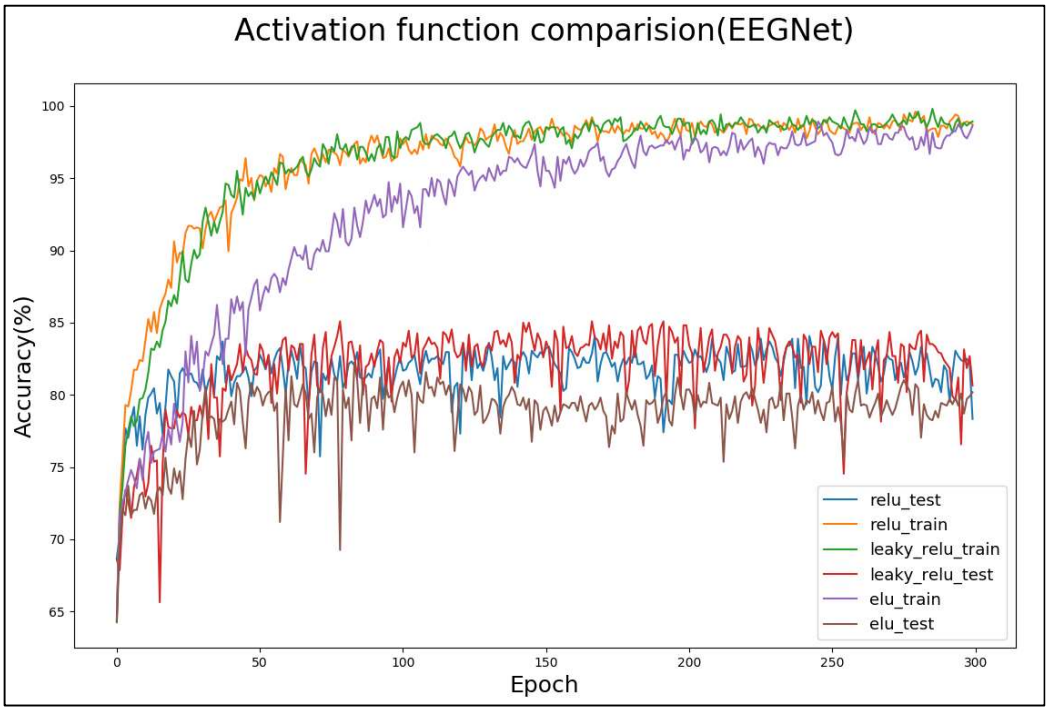
Batch size= 64 Learning rate = 1e-2 Epochs = 300
Optimizer: Adam Loss function: torch.nn.CrossEntropyLoss()

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

- **Result comparison**

  In this part, you can use the matplotlib library to draw the graph.
  Reference : **https://matplotlib.org/**

  The comparison figure should like the example as below. (EEGNet)



Report Spec (60%)

1. Introduction (20%)
2. Experiment set up (30%)
   A. The detail of your model
      - EEGNet
      - DeepConvNet
   B. Explain the activation function (ReLU, Leaky ReLU, ELU)
3. Experimental results (30%)
   A. The highest testing accuracy
      - Screenshot with two models
      - anything you want to present
   B. Comparison figures
      - EEGNet
      - DeepConvNet
4. Discussion (20%)
   A. Anything you want to share

---- Criterion of result (40%) ----
Accuracy > = 87% = 100 pts
Accuracy 85~87% = 90 pts
Accuracy 80~85% = 80 pts
Accuracy 75~80% = 70 pts
Accuracy < 75% = 60 pts

Score: 40% experimental results + 60% (report+ demo score)
P.S If the zip file name or the report spec have format error, it will be penalty (-5).

In the demo phase, you only need to show the highest testing accuracy of the model.

Reference:

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