Lab2 Report

1. Introduction

Implement simple EEG classification models which are EEGNet, DeepConvNet with BCI competition dataset. And use three different activation function to compare their difference.

2. Experiment set up

A. The detail of model

EEGnet:

```
print(EEGNet('ReLU'))
```

```
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): ReLU()
    (3): AvgPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0)
    (4): Dropout(p=0.25, inplace=False)
  )
  (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): ReLU()
    (3): AvgPool2d(kernel_size=(1, 8), stride=(1, 8), padding=0)
    (4): Dropout(p=0.25, inplace=False)
  )
  (classify): Sequential(
    (0): Linear(in_features=736, out_features=2, bias=True)
  )
)
```

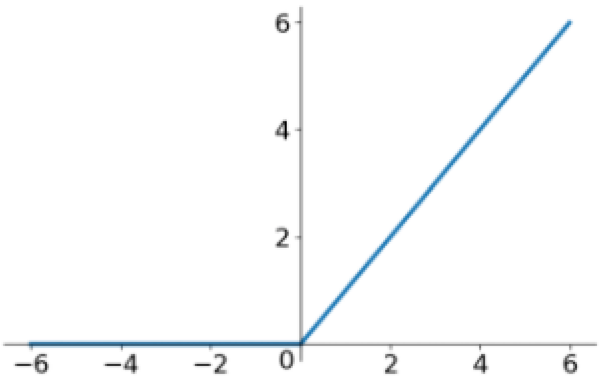
DeepConvNet:

```
print(DeepConvNet('ReLU'))
```

```
DeepConvNet(
  (part1): Sequential(
    (0): Conv2d(1, 25, kernel_size=(1, 5), stride=(1, 1))
    (1): Conv2d(25, 25, kernel_size=(2, 1), stride=(1, 1))
    (2): BatchNorm2d(25, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): ReLU()
    (4): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (5): Dropout(p=0.5, inplace=False)
  )
  (part2): Sequential(
    (0): Conv2d(25, 50, kernel_size=(1, 5), stride=(1, 1))
    (1): BatchNorm2d(50, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.5, inplace=False)
  )
  (part3): Sequential(
    (0): Conv2d(50, 100, kernel_size=(1, 5), stride=(1, 1))
    (1): BatchNorm2d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.5, inplace=False)
  )
  (part4): Sequential(
    (0): Conv2d(100, 200, kernel_size=(1, 5), stride=(1, 1))
    (1): BatchNorm2d(200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.5, inplace=False)
  )
  (classify): Sequential(
    (0): Linear(in_features=8600, out_features=2, bias=True)
  )
)
```

B. Activation function

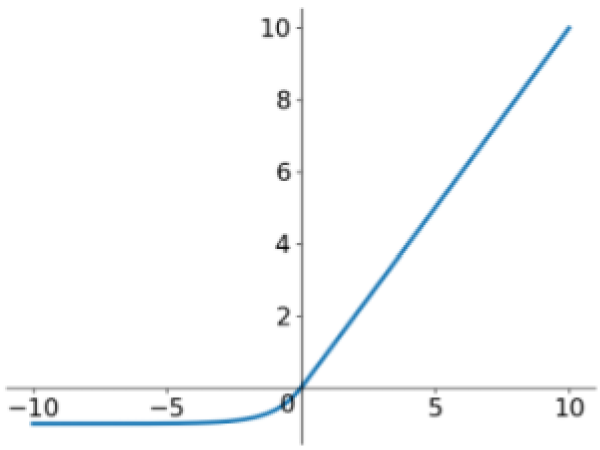
ReLU:



If the value is positive, the value is output, if the value is negative, the output is 0, solve the gradient explosion problem, and the convergence speed is fast.

Leaky ReLU: The ReLU activation function will make the output of the negative part of the neuron to be 0, however when the output of a certain neuron is 0, it is difficult to output again.
 To overcome this problem, when value is negative, the output is "0.01*output".

ELU:



$f(x)$

Similar to Leaky ReLU, the formula is: $f(x)=\begin{cases} x, & \text{if } x>0 \\ \alpha\left(e^{x}-1\right), & \text{otherwise} \end{cases}$

3. Experimental result

A. The highest testing accuracy

	ReLU	Leaky ReLU	ELU
EEGNet	**83%**	83%	79%
DeepConvNet	**74%**	73%	73%

EEGNet_ReLU:

```
Epoch: 147 | train accuracy: 0.9769 | test accuracy: 0.82
Epoch: 148 | train accuracy: 0.9898 | test accuracy: 0.84
Epoch: 149 | train accuracy: 0.9833 | test accuracy: 0.84
Epoch: 150 | train accuracy: 0.9759 | test accuracy: 0.83
```

EEGNet_Leaky ReLU:

```
Epoch: 147 | train accuracy: 0.9880 | test accuracy: 0.83
Epoch: 148 | train accuracy: 0.9833 | test accuracy: 0.83
Epoch: 149 | train accuracy: 0.9824 | test accuracy: 0.84
Epoch: 150 | train accuracy: 0.9787 | test accuracy: 0.83
```

EEGNet_ELU:

```
Epoch: 147 | train accuracy: 0.9731 | test accuracy: 0.80
Epoch: 148 | train accuracy: 0.9759 | test accuracy: 0.78
Epoch: 149 | train accuracy: 0.9731 | test accuracy: 0.79
Epoch: 150 | train accuracy: 0.9676 | test accuracy: 0.79
```

DeepConvNet_ReLU:

```
Epoch: 146 | train accuracy: 0.8574 | test accuracy: 0.74
Epoch: 147 | train accuracy: 0.8602 | test accuracy: 0.76
Epoch: 148 | train accuracy: 0.8694 | test accuracy: 0.74
Epoch: 149 | train accuracy: 0.8620 | test accuracy: 0.73
Epoch: 150 | train accuracy: 0.8704 | test accuracy: 0.74
```

DeepConvNet_Leaky ReLU:

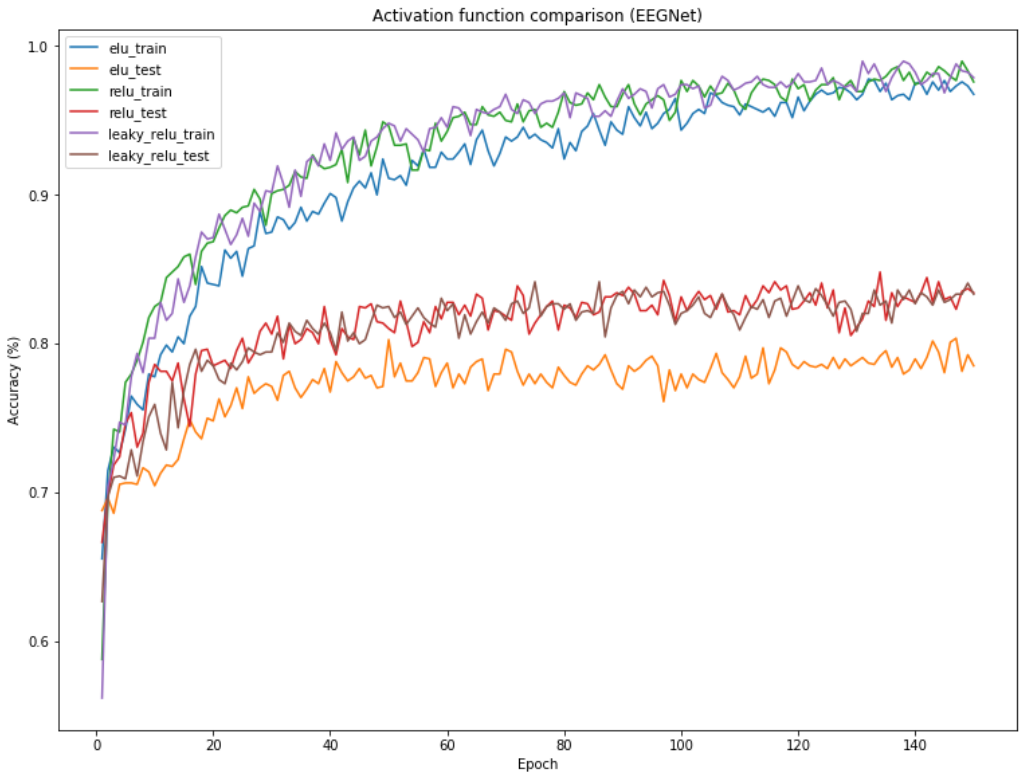
```
Epoch: 147 | train accuracy: 0.8472 | test accuracy: 0.74
Epoch: 148 | train accuracy: 0.8556 | test accuracy: 0.76
Epoch: 149 | train accuracy: 0.8620 | test accuracy: 0.74
Epoch: 150 | train accuracy: 0.8611 | test accuracy: 0.73
```

DeepConvNet_ELU:

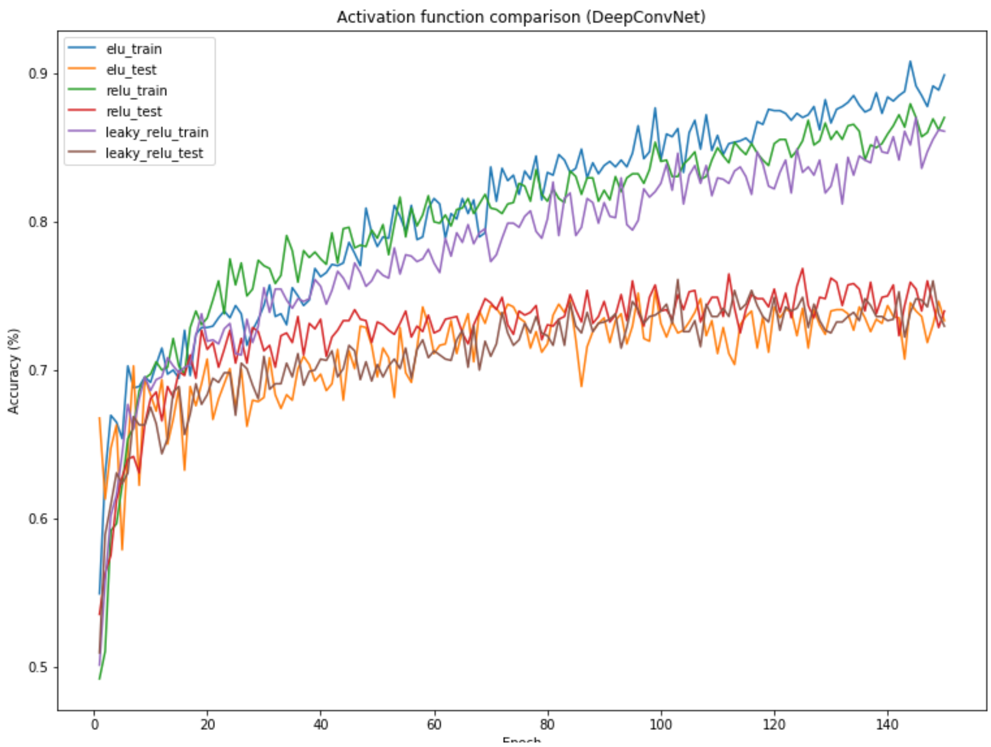
```
Epoch: 146 | train accuracy: 0.8852 | test accuracy: 0.74
Epoch: 147 | train accuracy: 0.8778 | test accuracy: 0.72
Epoch: 148 | train accuracy: 0.8917 | test accuracy: 0.73
Epoch: 149 | train accuracy: 0.8889 | test accuracy: 0.75
Epoch: 150 | train accuracy: 0.8991 | test accuracy: 0.73
```

B. Comparison figures

EEGNet:



DeepConvNet:



4. Discussion

I found that the train accuracy of the two models will be higher than the test accuracy, which may be an overfitting problem.

Then among the 3 different functions, ReLU is the best performing one, and EEGNet has better performance than DeepConvNet.