Lab4–1 EEG classification

Deep Learning and Practice

資科工碩1 游騰德 310551054

Introduction

This lab is to implement simple EEG classification models which are EEGNet and DeepConvNet. The networks to be implemented will be trained and tested on BCI competition dataset. This dataset includes 2 channels of 750 data points per each. The main goal is to take the input data and classify them into 2 classes. Additionally, the networks will be implemented by 3 different activation functions, which are ReLU, Leaky ReLU, and ELU. Please refer to the following of the report for further implementing details, including hyperparameters as well as the introduction to the activation functions.

Experiment set up

Model details

EEGNet

```
EEGNet(
  (activation): ELU(alpha=1.0)
  (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, inplace=False)
  )
  (seperableConv): 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, inplace=False)
  )
  (classify): Sequential(
    (0): Linear(in_features=736, out_features=2, bias=True)
  )
)
```

DeepConvNet

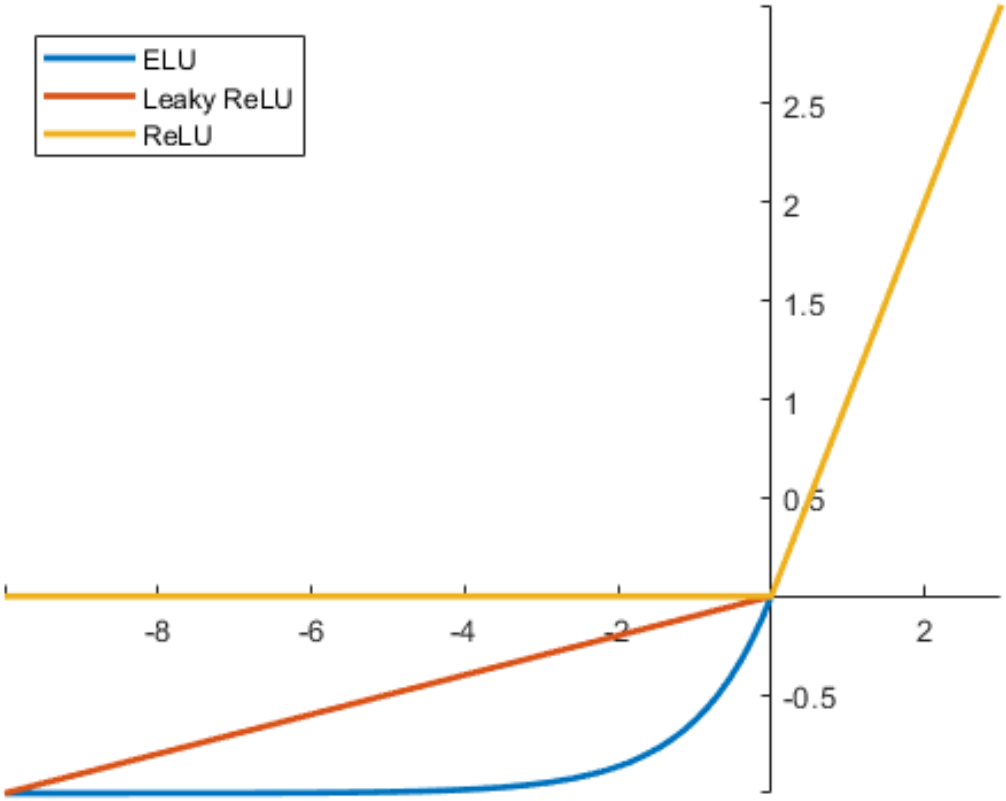
```
DeepConvNet(
  (activation): ELU(alpha=1.0)
  (block0): Conv2d(1, 25, kernel_size=(1, 5), stride=(1, 1))
  (block1): Sequential(
    (0): Conv2d(25, 25, kernel_size=(2, 1), stride=(1, 1))
    (1): BatchNorm2d(25, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.25, inplace=False)
  )
  (block2): 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): ELU(alpha=1.0)
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.25, inplace=False)
  )
  (block3): 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): ELU(alpha=1.0)
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.25, inplace=False)
  )
  (block4): 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): ELU(alpha=1.0)
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.25, inplace=False)
  )
  (classify): Sequential(
    (0): Linear(in_features=8600, out_features=2, bias=True)
  )
)
```

Hyperparameters

- Batch size = 64
- Learning rate = 1e–4
- Epochs = 500
- Optimizer: Adam Loss function: Cross Entropy Loss
- Weight decay = 0.01
- Dropout = 0.25

Activation functions

Activation functions are applied before a neuron outputs the value. The 3 activation functions used, ReLU, Leaky ReLU, and ELU, are only different in the region that **x** is less than 0. As shown in the figure below, Leaky ReLU preserves the most gradient, while ReLU and ELU may occur to gradient vanishing problem more easily. Note that the following figure is downloaded from the ResearchGate website. The definition of each activation functions are written below.



ReLU

$$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases}$$

Leaky ReLU

$$f(x) = \begin{cases} \alpha x, & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases}$$

ELU

$$f(x) = \begin{cases} \alpha(e^x - 1), & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases}$$

Experimental results

Highest testing accuracy

Screenshot with two models

```
Inference
EEGNet_ELU Test Acc:            0.8435185185185186
EEGNet_LeakyReLU Test Acc:              0.8777777777777778
EEGNet_ReLU Test Acc:           0.8870370370370371
DeepConvNet_ELU Test Acc:               0.8268518518518518
DeepConvNet_LeakyReLU Test Acc:         0.8444444444444444
DeepConvNet_ReLU Test Acc:              0.8518518518518519
```

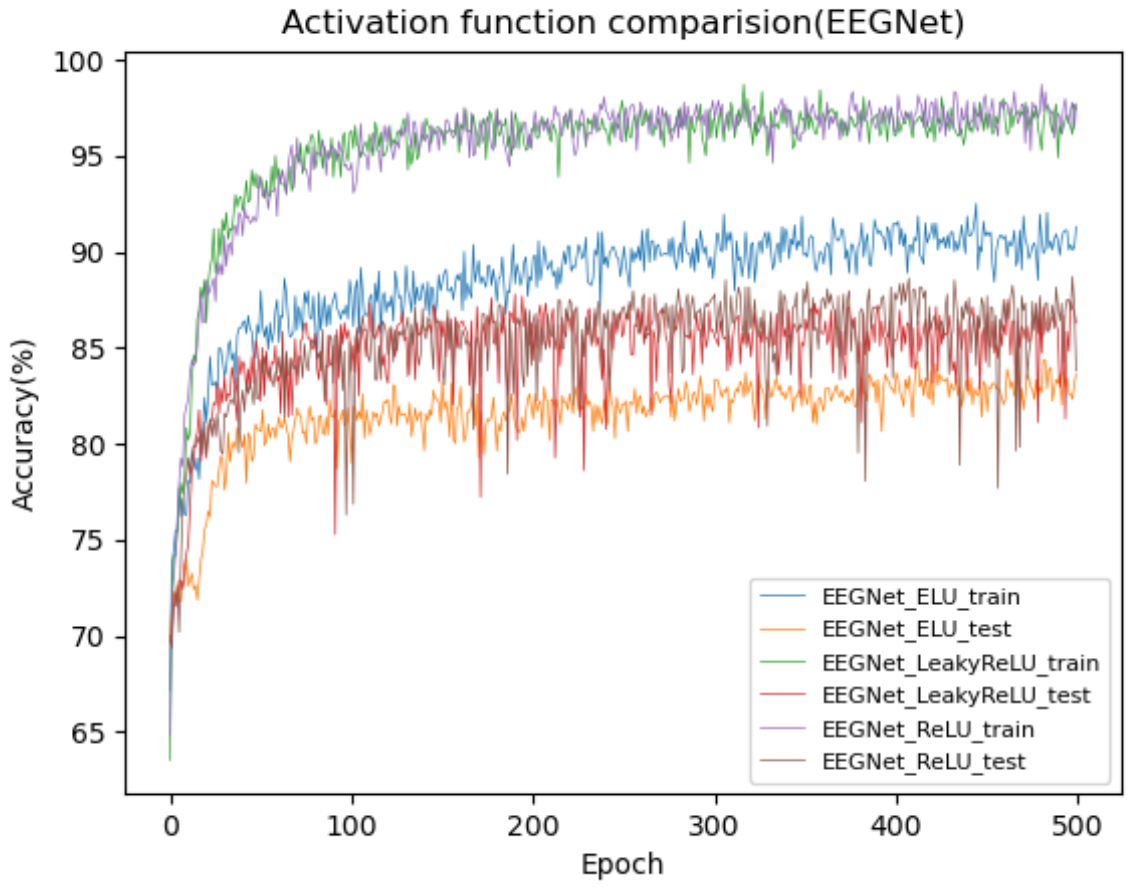
- Best of EEGNet: w/ ReLU activation, **88.70%** accuracy.
- Best of DeepConvNet: w/ ReLU activation, **85.19%** accuracy.

Observation

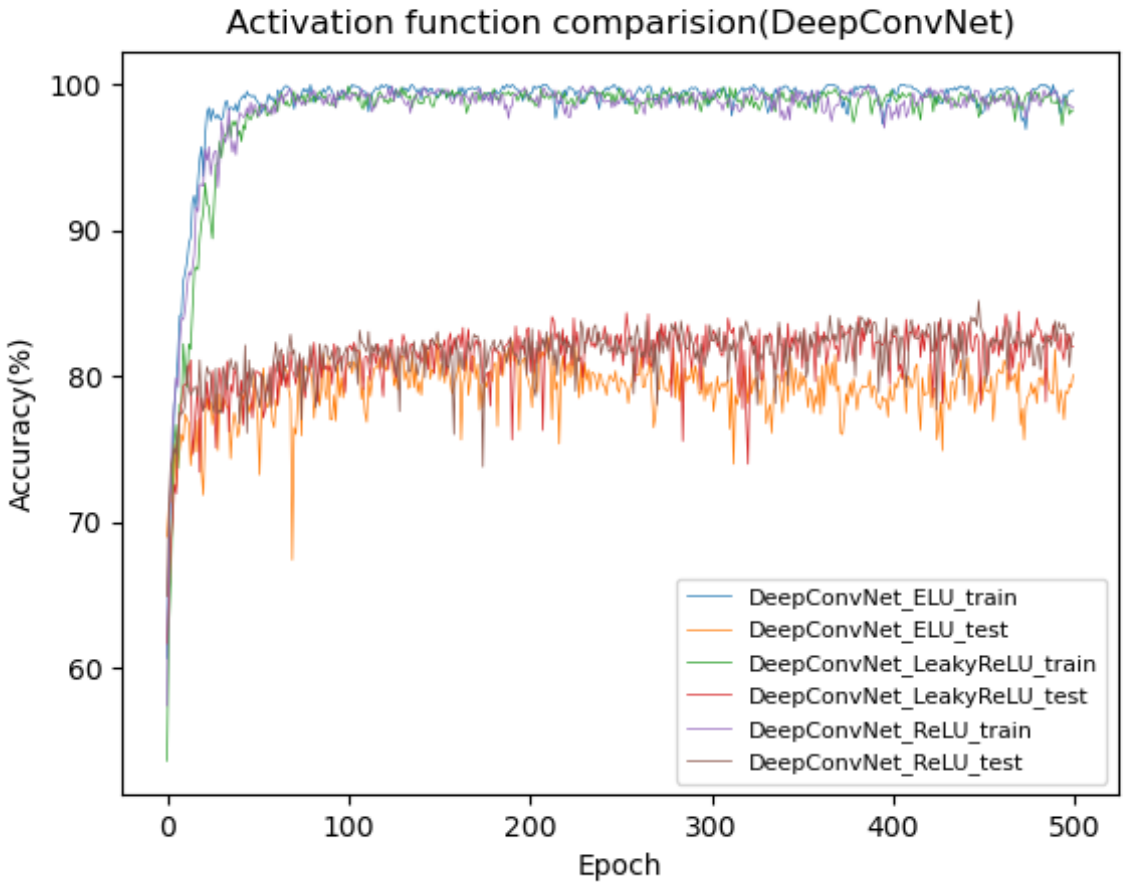
DeepConvNet converged earlier and results in worse accuracy while it is deeper.

Comparison figures

EEGNet



DeepConvNet



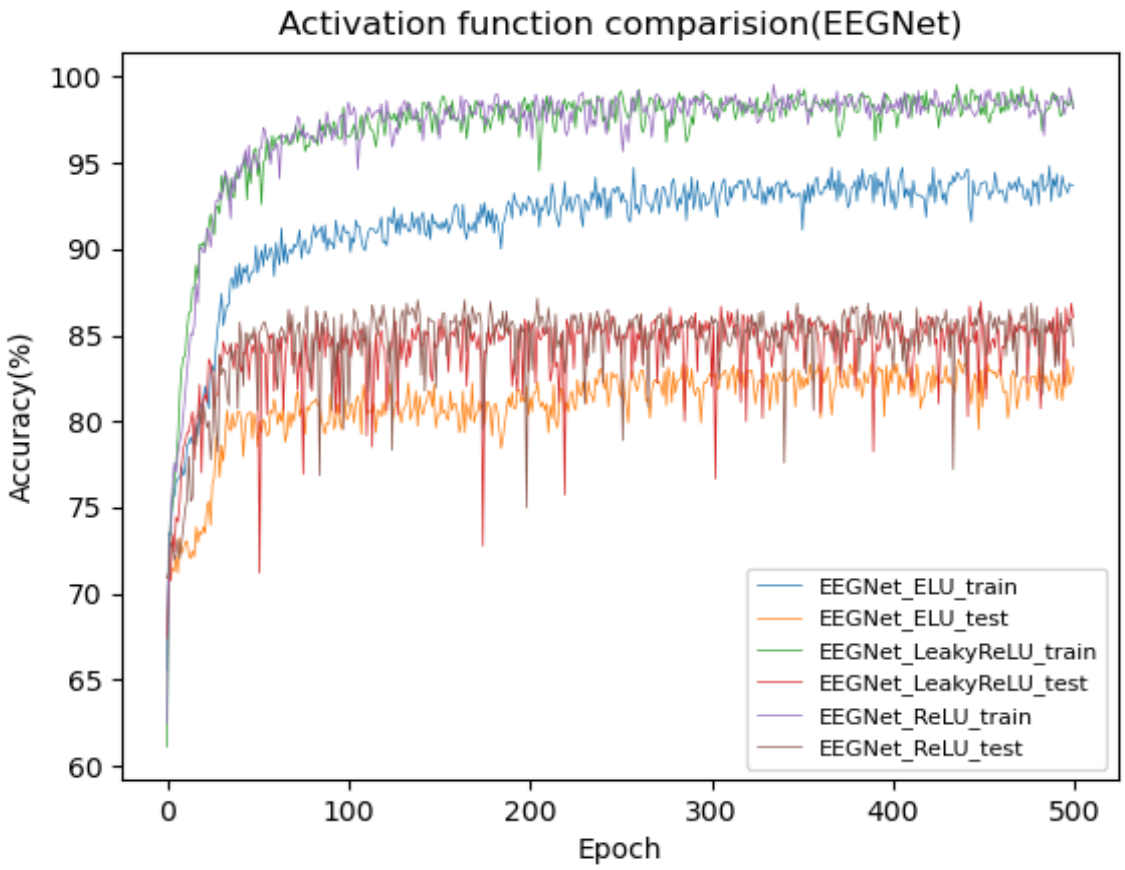
Discussion

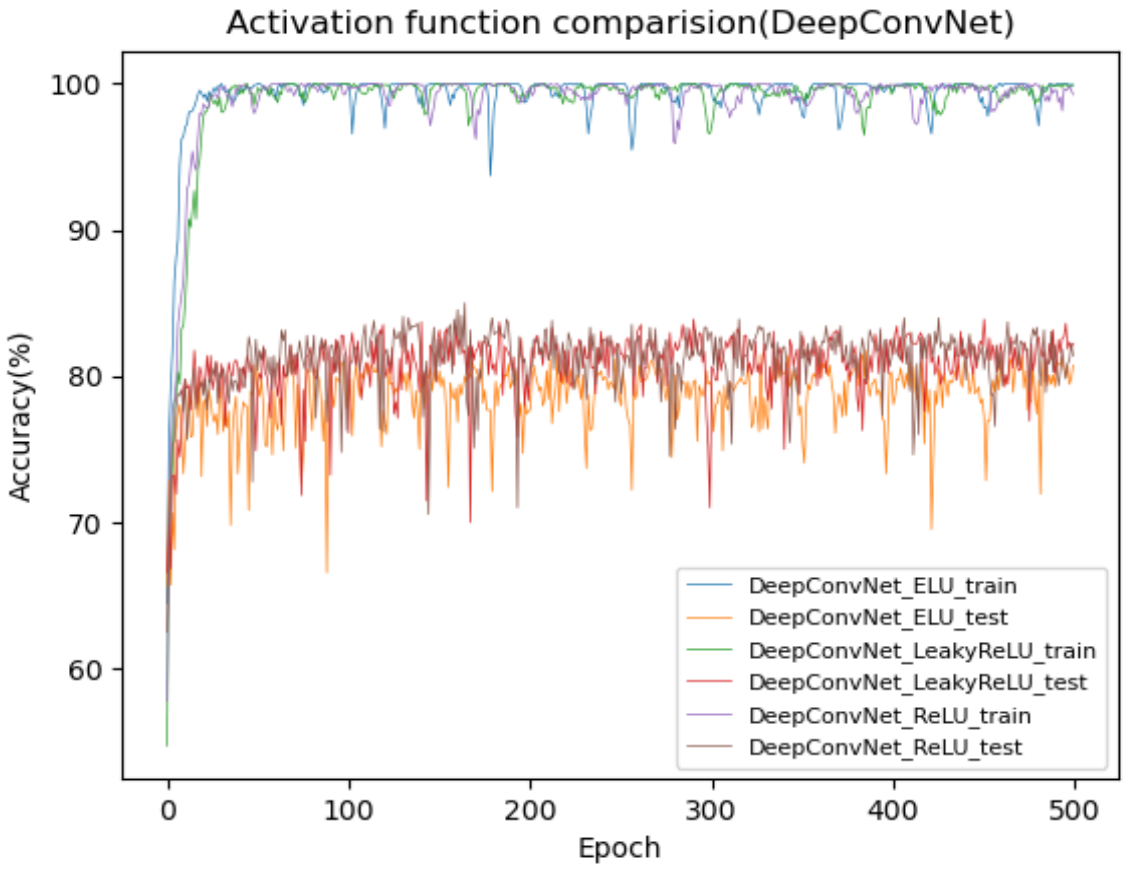
Dropout rate is a special hyperparameter to be tuned during training. It controls the amount of data to be randomly dropped out, and the purpose is to lower the influences caused by bias. In this lab, I originally set the dropout rate to 0.25, which means one–forth of the data may be randomly dropped after passing through a dropout layer.

Lower the dropout rate

As I decreased the dropout rate to 0.15, the highest accuracy of both EEGNet and DeepConvNet dropped by approximately 1%. This phenomena roughly shows that random dropping data is beneficial to training. The overall results and compassion figures with dropout rate set to 0.15 is shown below for your reference.

```
Inference
EEGNet_ELU Test Acc:            0.837037037037037
EEGNet_LeakyReLU Test Acc:              0.8694444444444445
EEGNet_ReLU Test Acc:           0.8712962962962963
DeepConvNet_ELU Test Acc:               0.8175925925925925
DeepConvNet_LeakyReLU Test Acc:         0.8388888888888889
DeepConvNet_ReLU Test Acc:              0.85
```


Activation function comparision(EEGNet)
Accuracy(%)
Epoch
EEGNet_ELU_train
EEGNet_ELU_test
EEGNet_LeakyReLU_train
EEGNet_LeakyReLU_test
EEGNet_ReLU_train
EEGNet_ReLU_test


Activation function comparision(DeepConvNet)
Accuracy(%)
Epoch
DeepConvNet_ELU_train
DeepConvNet_ELU_test
DeepConvNet_LeakyReLU_train
DeepConvNet_LeakyReLU_test
DeepConvNet_ReLU_train
DeepConvNet_ReLU_test