DLP HW2

310551053 Chieh-Ming Jiang

July 23th 2021

1 Introduction

In this assignment, I am going to implement EEGNet and DeepConvNet with pytorch. Moreover, I am going to try 3 kinds of different activation functions, which are ReLU, Leaky ReLU and ELU.
The dataset is from BCI Competition III - IIIb Cued motor imagery with online feedback (nonstationary classifier) with 2 classes (left hand, right hand) from 3 subjects. There are two channels, and 750 data points for each channel.

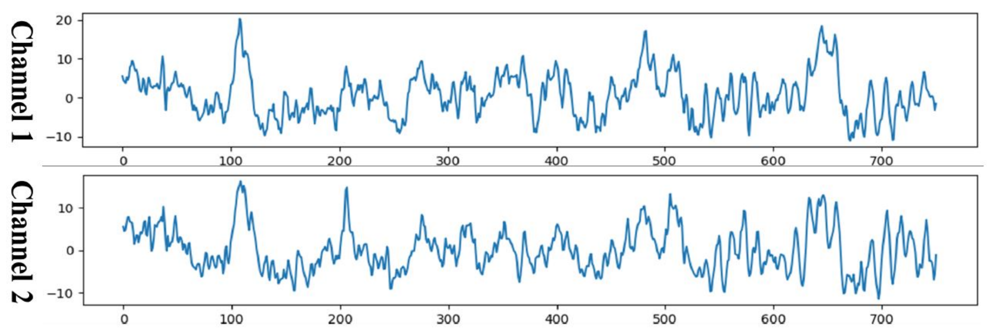

Figure 1: Dataset

2 Experiment setups

2.1 EEGNet

```
EEGNET(
  (activation): ReLU()
  (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=0.15, 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): ReLU()
    (3): AvgPool2d(kernel_size=(1, 8), stride=(1, 8), padding=0)
    (4): Dropout(p=0.15, inplace=False)
  )
  (classify): Sequential(
    (0): Linear(in_features=736, out_features=2, bias=True)
  )
)
```

Figure 2: EEGNet

2.2 DeepConvNet

```
DeepConvNet(
  (activation): ReLU()
  (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): ReLU()
    (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): ReLU()
    (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): ReLU()
    (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): ReLU()
    (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)
  )
)
```

Figure 3: DeepConvNet

2.3 Activation function

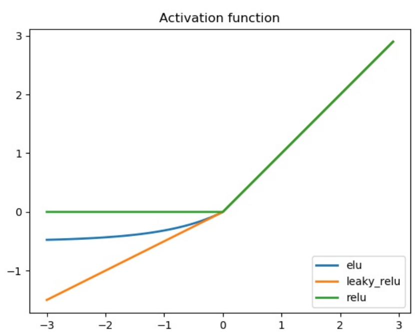

Figure 4: Activation

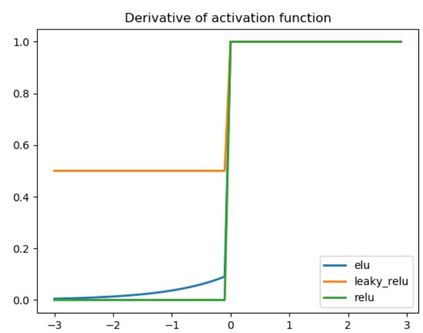

Figure 5: Derivative of Activation

1. ReLU

$$ReLU = max(0, x)$$

2. ELU

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

3. LeakyRELU

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

Observing from the Figure4, the difference of 3 kinds of activation functions are the points when x is smaller than 0. Both first derivative of ReLU and ELU will converge to nearly 0 when x is much more smaller than 0. That is, the gradient vanishing problem may happen in ReLU and ELU function. On the contrary, LeakyRELU avoids the gradient vanishing problem.

3 Results of your testing

1. EEGNet

Hyper parameters of EEGNet	
learning rate	0.001
epoch	500
dropout	0.15
weight decay	0.012
criterion	CrossEntropyLoss
optimizer	Adam

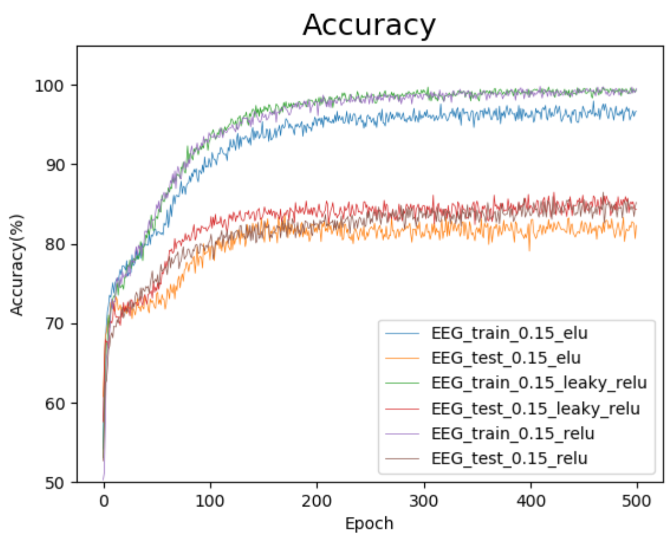

Figure 6: EEGNet

2. DeepConvNet

Hyper parameters of DeepConvNet	
learning rate	0.001
epoch	500
dropout	0.25
weight decay	0.012
criterion	CrossEntropyLoss
optimizer	Adam

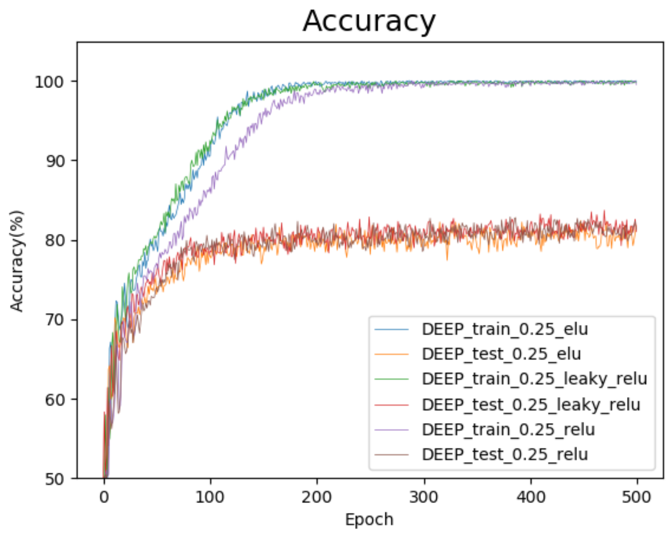

Figure 7: DeepConvNet

3. Comparison with different activation function

Best testing accuracy		
	EEGNet	DeepConvNet
ReLU	**0.875**	0.824
Leaky ReLU	0.865	0.837
ELU	0.844	0.829

4 Discussion

Without weight decay

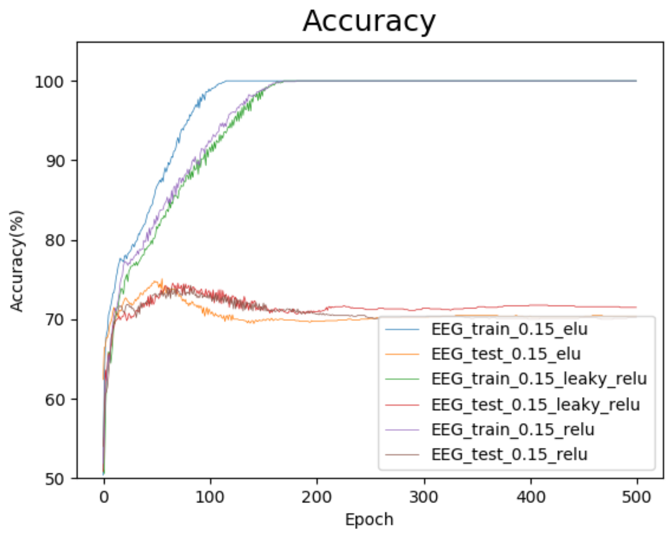

Figure 8: EEGNet without weight decay

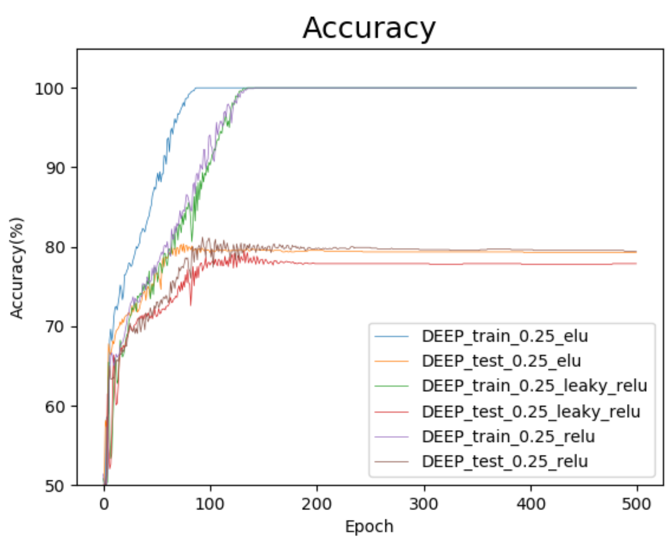

Figure 9: DeepConvNet without weight decay

Best testing accuracy		
	EEGNet	DeepConvNet
ReLU	0.743	**0.812**
Leaky ReLU	0.745	0.795
ELU	0.751	0.804

If I were not to use weight decay as a normalization term for the model, the model seems to be overfitted. Both of models almost stop updating parameters since 100 epochs, whose training accuracy are almost 100%. Observing the testing accuracy of EEGNet, the accuracy falls about almost 10% without weight decay. As for DeepConvNet, the testing accuracy falls about only 3%. As a consequence, adding the weight decay term helps the model learn better.