NCTU Spring 2020 Deep Learning and Practice

Lab #2

0616015 劉姿利

Table of Content

1 Introduction

使用 PyTorch 構造 EEGNet 與 DeepConvNet 配上不同的 activation functions（ReLU、LeakyReLU、ELU）來對 BCI competition dataset 進行 classification。

2 Experimental Setup

2.1 The Detail of Model

(presented in PyTorch network structure summary)

EEGNet

```
EEGNet(
  (activate_func): 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=0.6, 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=0.6, inplace=False)
  )
  (classify): Sequential(
    (0): Linear(in_features=736, out_features=2, bias=True)
  )
)
```

DeepConvNet

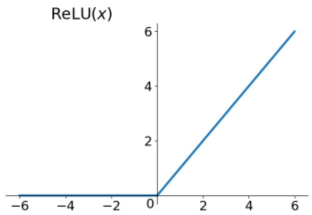
```
DeepConvNet(
  (activate_func): ReLU()
  (Conv1): Sequential(
    (0): Conv2d(1, 25, kernel_size=(1, 5), stride=(1, 1))
  )
  (Conv2): Sequential(
    (0): Conv2d(25, 25, kernel_size=(2, 1), stride=(1, 1))
    (1): BatchNorm2d(25, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): ReLU()
    (3): AvgPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0)
    (4): Dropout(p=0.5, inplace=False)
  )
  (Conv3): 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): AvgPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0)
    (4): Dropout(p=0.5, inplace=False)
  )
  (Conv4): 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): AvgPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0)
    (4): Dropout(p=0.5, inplace=False)
  )
  (Conv5): 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): AvgPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0)
    (4): Dropout(p=0.5, inplace=False)
  )
  (Dense): Sequential(
    (0): Linear(in_features=8600, out_features=2, bias=True)
  )
)
```

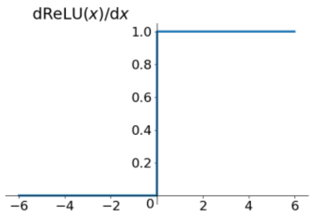
2.2 Activation Functions

(photos from: https://zhuanlan.zhihu.com/p/25110450)

ReLU

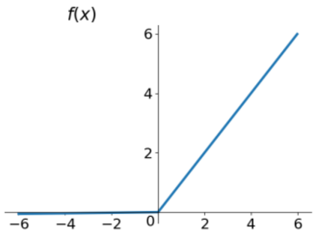
$$f(x) = max(0,\ x)$$

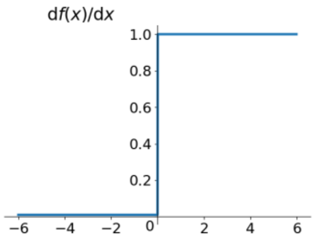




LeakyReLU

$$f(x) = \max(0.01x, x)$$

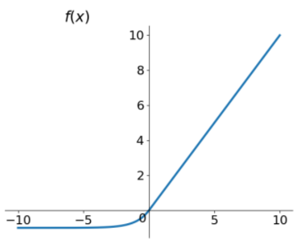


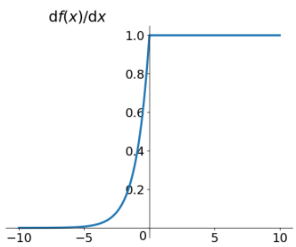


- 若 ReLU 的 inputs 大部份小於 0 會有太多 gradient 為 0 的部份很難 train，LeakyReLU 提供負數也保有 0.01 的 gradient

ELU

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





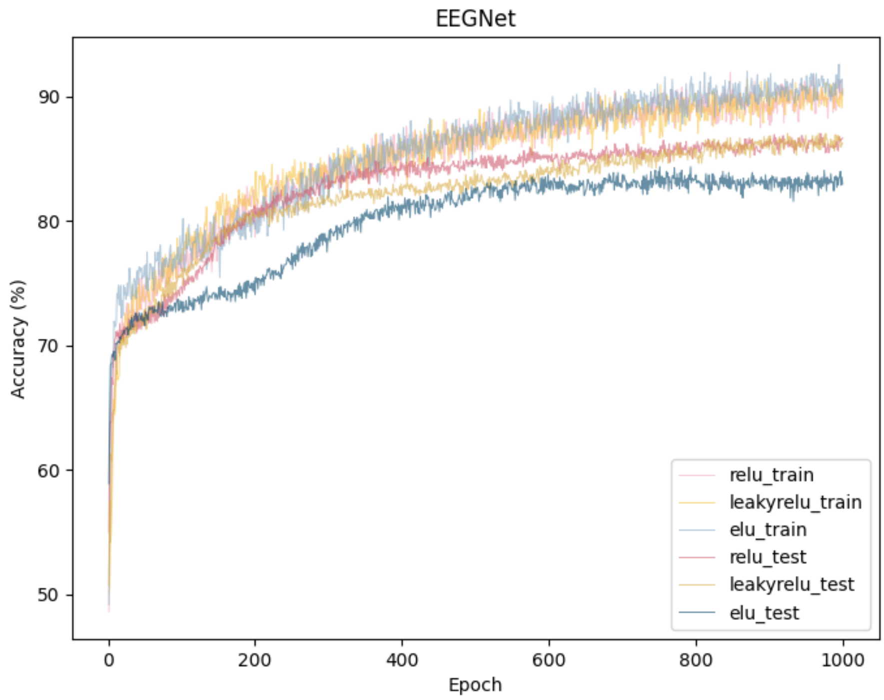
- 與 LeakyReLU 相同也是為了解決 ReLU 的問題，但計算量稍大

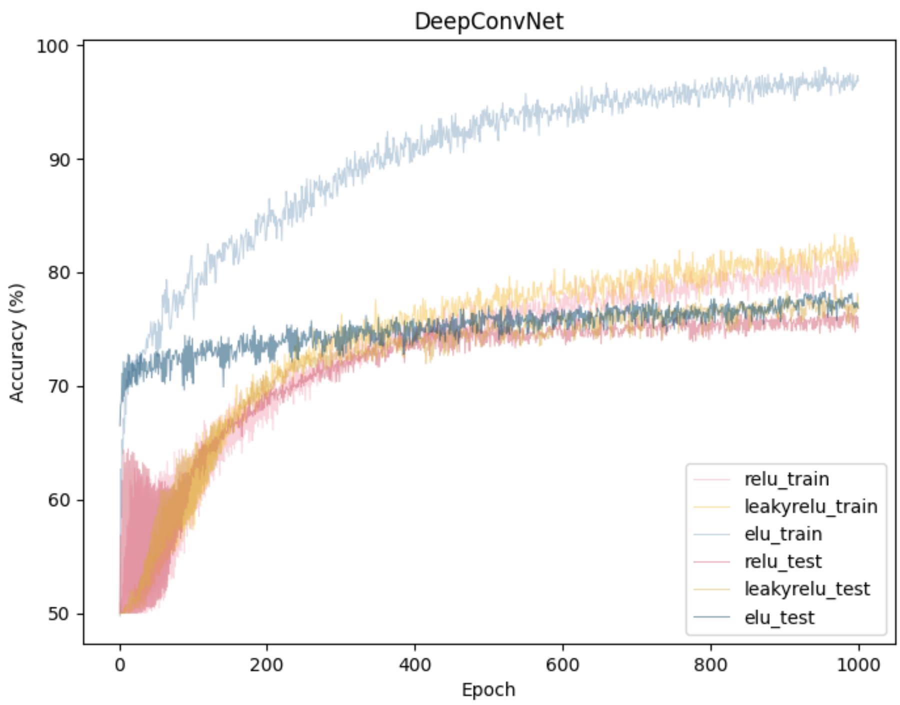
3 Experimental Results

3.1 The Highest Testing Accuracy

	ReLU	LeakyReLU	ELU
EEGNet	88.43%	88.80%	84.63%
DeepConvNet	80.09%	80.46%	79.35%

3.1 Comparison Figures





4 Discussion

這次 lab 一開始 train 的時候結果很不好，EEGNet 的 test accuracy 一直卡在 83% 左右上不去，後來才發現是忘記下 net.eval() 所以在 testing 的時候 dropout 行為跟 training 一樣所以 test accuracy 才會這麼慘，改好之後就順利達到 86% 了，而因為這個，我才想到也許改 dropout 是關鍵，所以將原來的 0.25 改成 0.6，用 train accuracy 來換 test accuracy 數字就比較好看了。

至於兩個 network 的比較，可以看到 EEGNet 的表現比 DeepConvNet 好上很多，而所有 activation functions 中又以 LeakyReLU 表現最好。