0516003 李智嘉 lab2 report

1. Introduction

在上次的作業實做了 back propagation，這次將利用讓運算量不要像 fully connected 那麼大的 Convolution Neural Network，建制兩種不同的網絡，EEGNet 與 DeepConvNet，並各使用三種不同的 activation function (ReLU, LeakyReLU, ELU)，套用在助教先處理過的 BCI competition dataset 上，比對彼此結果的差異，並探討怎麼樣調整 model 參數會使正確率較高。

2. Experiment set up

A) Model

a) EEGNet

```
EEGNet(
  (firstConv): Sequential(
    (0): Conv2d(1, 32, kernel_size=(1, 31), stride=(1, 1), padding=(0, 15), bias=False)
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (depthwiseConv): Sequential(
    (0): Conv2d(32, 64, kernel_size=(2, 5), stride=(1, 2), groups=16, bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): AvgPool2d(kernel_size=(1, 6), stride=(1, 6), padding=0)
    (4): Dropout(p=0.5)
  )
  (separableConv): Sequential(
    (0): Conv2d(64, 64, kernel_size=(1, 13), stride=(1, 1), padding=(0, 6), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): AvgPool2d(kernel_size=(1, 5), stride=(1, 5), padding=0)
    (4): Dropout(p=0.15)
  )
  (classify): Sequential(
    (0): Linear(in_features=768, out_features=2, bias=True)
  )
)
```

b) DeepConvNet

```
DeepConvNet(
  (firstConv): Sequential(
    (0): Conv2d(1, 25, kernel_size=(1, 15), stride=(1, 2), padding=(0, 11), bias=False)
    (1): Conv2d(25, 25, kernel_size=(2, 1), stride=(1, 1), bias=False)
    (2): BatchNorm2d(25, eps=1e-05, momentum=0.2, affine=True, track_running_stats=True)
    (3): ELU(alpha=1.0)
    (4): MaxPool2d(kernel_size=(1, 7), stride=(1, 7), padding=0, dilation=1, ceil_mode=False)
    (5): Dropout(p=0.5)
  )
  (secondConv): Sequential(
    (0): Conv2d(25, 50, kernel_size=(1, 11), stride=(1, 1), padding=(0, 5), bias=False)
    (1): BatchNorm2d(50, eps=1e-05, momentum=0.15, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): MaxPool2d(kernel_size=(1, 5), stride=(1, 5), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.5)
  )
  (thirdConv): Sequential(
    (0): Conv2d(50, 100, kernel_size=(1, 7), stride=(1, 1), padding=(0, 3), bias=False)
    (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, 3), stride=(1, 3), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.4)
  )
  (fourthConv): Sequential(
    (0): Conv2d(100, 200, kernel_size=(1, 3), stride=(1, 1), padding=(0, 1), bias=False)
    (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=0.2)
  )
  (lastDense): Sequential(
    (0): Linear(in_features=200, out_features=2, bias=True)
  )
)
```

其中我對 EEGNet 有特別多加調整，最後結果也最好，後面會再顯示成果。

我將 filter 數目調高，並將 kernal size 降低，希望他可以做的再更仔細一點，而前面的 dropout rate 調高到 0.5，後面調低成 0.15，一方面希望他可以減緩 overfitting，一方面又可以學到後面較 high level 的 feature。其他的參數就再各自微調。

而 DeepConvNet 做了一下，稍微調整後發現層數太多，表現很難很好，就沒再更動他了。

B) Activation function

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

ReLU 會將負數變 0，正數維持不變

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

LeakyReLU 跟 ReLU 很像，不過負數不是變 0，而是乘上一個很小的數字，使其還是保留一點影響力，在 pytorch 的 default 是 0.01，這邊我也沒有再額外更動他

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

ELU 在輸入正數時一樣是保留，負數則是趨近於負 alpha 的一個曲線，在 pytorch 的 default 是 1.0，我一樣也沒有在額外動他

3. Experimental results

- EEGNet highest testing accuracy: 0.8713

```
(kk) karljackab@pc3421:~/DL/lab2$ python3 lab2_0516003.py
use cuda : True
(1080, 1, 2, 750) (1080,) (1080, 1, 2, 750) (1080,)
Start EEGNet with ReLU
EEGNet with ReLU highest testing accuracy: 0.8712962962962963
Start EEGNet with LeakyReLU
EEGNet with LeakyReLU highest testing accuracy: 0.8537037037037037
Start EEGNet with ELU
EEGNet with ELU highest testing accuracy: 0.8157407407407408
```

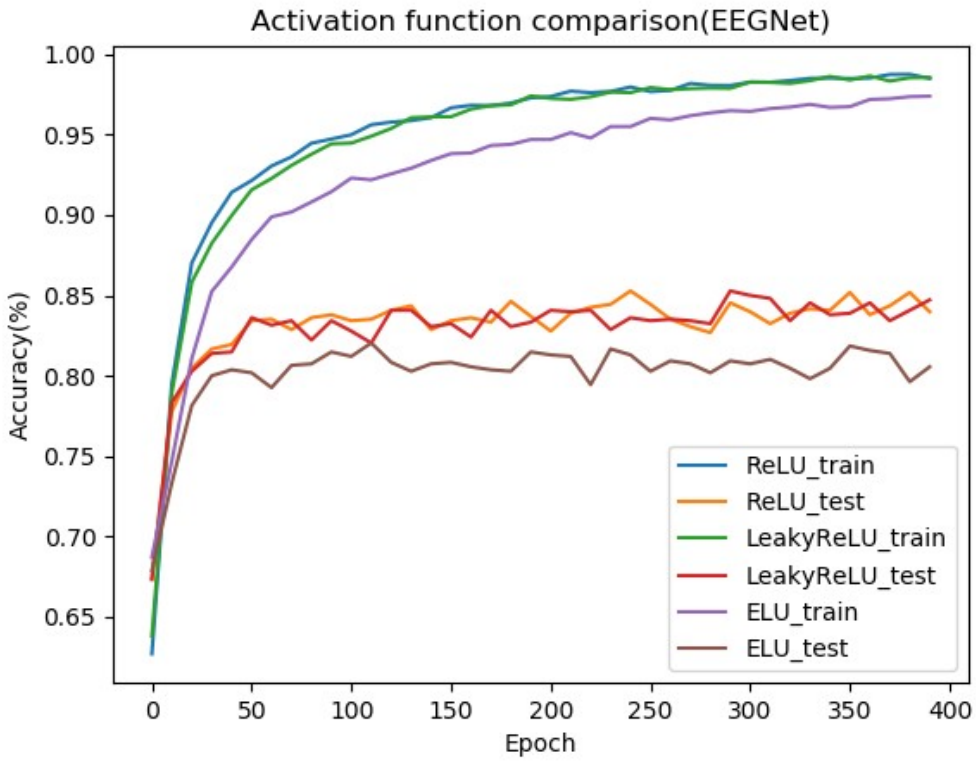
- DeepConvNet highest testing accuracy: 0.8287

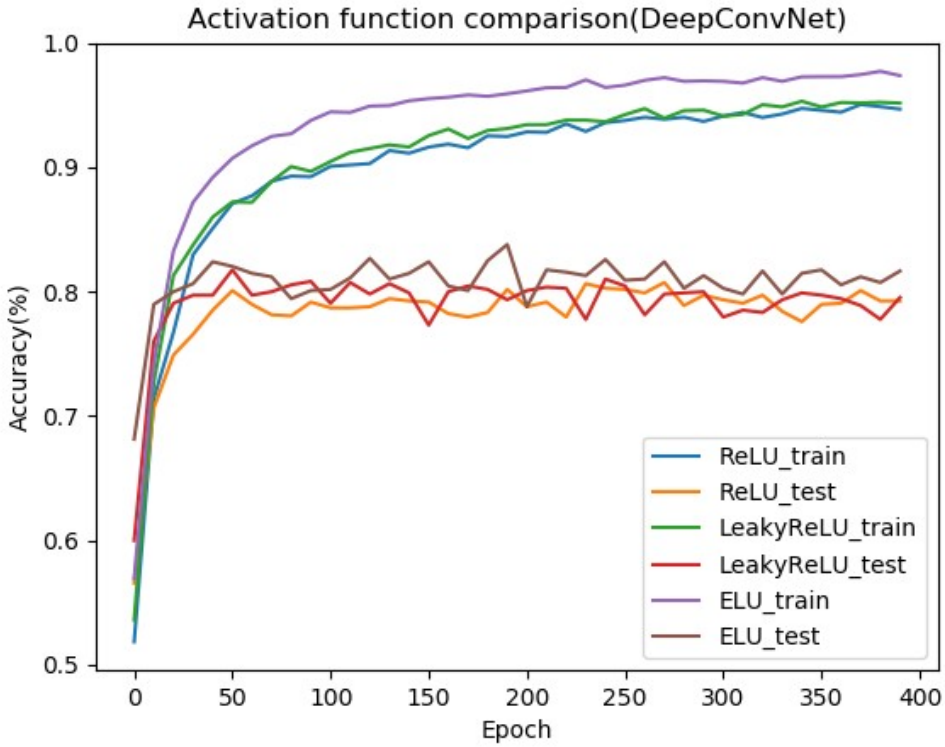
```
Start DeepConvNet with ReLU
DeepConvNet with ReLU highest testing accuracy: 0.8240740740740741
Start DeepConvNet with LeakyReLU
DeepConvNet with LeakyReLU highest testing accuracy: 0.8231481481481482
Start DeepConvNet with ELU
DeepConvNet with ELU highest testing accuracy: 0.8287037037037037
```

- 可以看到 EEGNet 比 DeepConvNet 表現的好很多，主要原因可能是 DeepConvNet 太多層了，對於 training data 過度 fitting，而導致 modle 不夠 generalize，在 testing 也就表現不好
- 針對 EEGNet 來看，三種 activation function 中，ReLU 表現最好，最差的則是 ELU，可能是因為 ReLU 對於負數較嚴格，直接歸零讓 output 不會受到干擾

- 相比下來在 DeepConvNet 上，不同的 activation functionru 就沒有差別那麼大了，而且反而是 ELU 表現比較好，大概也是 case by case 吧

- Comparison figures





- 可以看到不管是在 EEGNet 還是 DeepConvNet，在 Epoch 到 50 多時，testing accuracy 就幾乎穩定了，只剩下些微的起伏
- overfitting 的情況，因為這邊的 Epoch 總數也不高，其實並不是很明顯，唯一就是 DeepConvNet 的 LeakyReLU 有點些微往下掉的趨勢而已，但差異也不大

4. Discussion

這次的作業其實還滿開心的，資料都已經事前準備好了，只需要裝起來再建出已經規劃好的 model，沒有遇到太多困難，查一下 pytorch cuda 的用法、Dataloader 的用法和 Model 怎麼建構，很快就可以 run 了。

花最多時間的是調整參數讓他可以正確率最高，中間試了好多種可能，甚至還有調整 batch size 和 learning rate 的大小，而每次改完都要讓他跑一段時間，真的很費心力，最後是有一次不小心正確率到 87% 了，才終於告一段落。

這次作業寫得很開心，看到 model 跑起來，然後 accuracy 慢慢上升真的有股爽快感，下次作業是 ResNet 和其他東西，真是期待。