DLP Lab2 309551067 吳子涵

1. Introduction

在這次的 lab 使用兩種神經網絡 EEGNet 和 DeepConvNet 在 BCI competition dataset 上，實作出 EEG classification。另外，每種神經網路分別搭配 ELU, ReLU, Leaky ReLU 三種不同 activation function，因此共有 6 種 network。

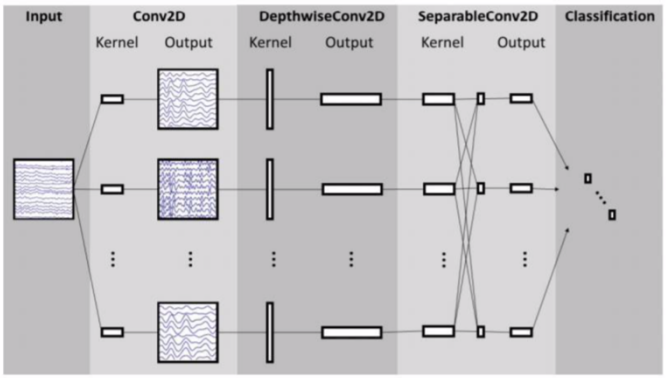
對於 6 種不同的 network，需要透過調整適當的參數，使準確率能夠提升，再使用 matplotlib 畫出 training, testing accuracy 與 epoch 數量之間關係的趨勢圖。

2. Experimental setups

A. The detail of your model

(1) EEGNET

架構如下圖所示，EEGNET 由三層 2D convolution layer 所構成。



第一層 convolution layer 將 input 做 convolution 以及 batchnorm
第二層 Depthwise convolution layer 做 convolution, batchnorm, activation function, pooling, dropout
第三層 Separable convolution layer 做 convolution, batchnorm, activation function, pooling, dropout
最後再經過 classification layer 來得到最後預測的結果。
詳細的 EEGNET 實作架構如下。

```
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): ELU(alpha=1.0)
    (3): AvgPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0)
    (4): Dropout(p=0.25)
  )
  (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): ELU(alpha=1.0)
    (3): AvgPool2d(kernel_size=(1, 8), stride=(1, 8), padding=0)
    (4): Dropout(p=0.25)
  )
  (classify): Sequential(
    (0): Linear(in_features=736, out_features=2, bias=True)
  )
)
```

我發現用原本助教給的參數訓練模型時，會有 overfitting 的現象，training accuracy 接近 99%，而 testing accuracy 在 85%左右一直無法提升。因此為了增加準確率，我嘗試降低 model 的 capacity，像是減少 hidden layer 的 node 數量，以及增加 dropout 的比例。而最後發現將 Depthwise Conv 這一層的 dropout 比例調成 0.5 時能提高準確率，而最高的準確率出現在使用 ReLU 時 (accuracy=0.88)。

(2) DeepConvNet

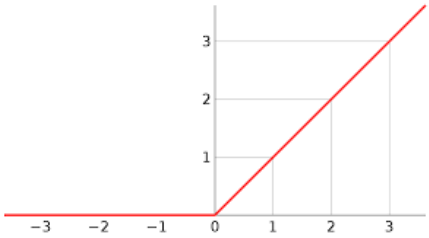
我將 DeepConvNet 分成四層 2D convolution layer。
第一層將 input 做 2 次 convolution, batchnorm, activation function, pooling, dropout。
第二層做 convolution, batchnorm, activation function, pooling, dropout。
第三、四層架構與第二層相似。
最後再經過 classification layer 來得到最後預測的結果。
詳細的 DeepConvNet 實作架構如下。
（C=2, T=750, N=2）

Layer	# filters	size	# params	Activation	Options
Input		(C, T)			
Reshape		(1, C, T)			
Conv2D	25	(1, 5)	150	Linear	mode = valid, max norm = 2
Conv2D	25	(C, 1)	25 * 25 * C + 25	Linear	mode = valid, max norm = 2
BatchNorm			2 * 25		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Conv2D	50	(1, 5)	25 * 50 * C + 50	Linear	mode = valid, max norm = 2
BatchNorm			2 * 50		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Conv2D	100	(1, 5)	50 * 100 * C + 100	Linear	mode = valid, max norm = 2
BatchNorm			2 * 100		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Conv2D	200	(1, 5)	100 * 200 * C + 200	Linear	mode = valid, max norm = 2
BatchNorm			2 * 200		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Flatten					
Dense	N			softmax	max norm = 0.5

B. Explain the activation function (ReLU, Leaky ReLU, ELU)

(1) ReLU

ReLU 的全名是 Rectified Linear Unit，是一種在神經網絡中常用的 activation function。ReLU 圖形如下圖所示，若值為正數，則輸出該值大小，若值為負數，則輸出 0。



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

優點：

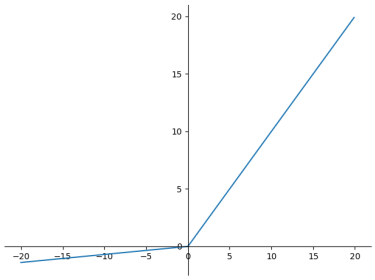
- 解決 vanishing gradient problem
- 計算速度快，因為只需要判斷輸入是否大於 0，不需要指數計算
- 收斂速度較 sigmoid, tanh 快
- 符合生物神經元特徵 (全有全無律): 當刺激未達一定強度時，不會產生神經衝動，當超過臨界值強度時，才會引起神經衝動，而進行訊息傳遞。

缺點：

- Dead ReLU Problem: 指某些神經元可能永遠不會被激活，導致相應的參數永遠不會被更新。

(2) Leaky ReLU

Leaky ReLU 是 ReLU 的變形，當 input 是負值時給予一個非零的斜率。Leaky ReLU 圖形如下圖所示。



$$\mathrm{LReLU}(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{if } x \leq 0 \end{cases}$$ (default : α=0.01)

if 0<α<1,
公式可轉換成

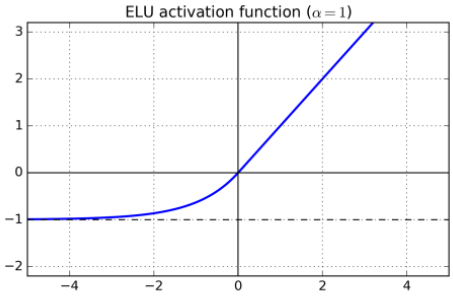
$$f(x) = max(ax, x)$$

優點：

- 解決 Dead ReLU Problem，能防止值為負號時永遠無法被激活的問題。

(3) ELU

ELU 的全名是 Exponential Linear Units，和 ReLU 相似，若值為正數，則輸出該值大小，若值為負數，則輸出指數函數的形式。ELU 圖形如下圖所示。



$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$$ (default : α=1.0)

公式可轉換成

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

優點：

- 與 ReLU 相比，函數輸出的平均值較接近 0，從而加快學習速度。
- 解決 Dead ReLU Problem，能防止值為負號時永遠無法被激活的問題。

缺點：

- 因為有指數運算，計算量稍大。

3. Experimental results

A. The highest testing accuracy

- Screenshot with two models

(1) EEGNET ReLU

```
EEGNET RELU Testing Accuracy: 0.880556
```

(2) EEGNET Leaky ReLU

```
EEGNET LEAKY RELU Testing Accuracy: 0.873148
```

(3) EEGNET ELU

```
EEGNET ELU Testing Accuracy: 0.850000
```

(4) DeepConvNet ReLU

```
DeepConvNet RELU Testing Accuracy: 0.826852
```

(5) DeepConvNet Leaky ReLU

```
DeepConvNet Leaky RELU Testing Accuracy: 0.818519
```

(6) DeepConvNet ELU

```
DeepConvNet ELU Testing Accuracy: 0.811111
```

- Anything you want to present

在合理的範圍下增加 batch size，大矩陣乘法的平行化使得效率提高，且跑完一個 epoch 所需迭代的次數減少，因此處理速度更快。

```
Test Epoch: 0 Accuracy: 0.658333
Test Epoch: 50 Accuracy: 0.819444
Test Epoch: 100 Accuracy: 0.820370
Test Epoch: 150 Accuracy: 0.841667
Test Epoch: 200 Accuracy: 0.851852
Test Epoch: 250 Accuracy: 0.858333
```

Batch Size 過小，花費的時間多，且梯度震盪嚴重，不利於收斂。

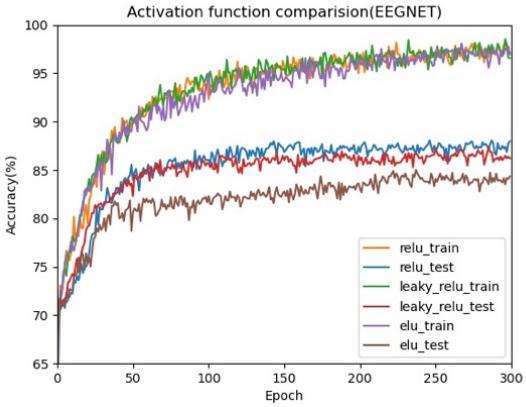
```
Test Epoch: 0 Accuracy: 0.681481
Test Epoch: 50 Accuracy: 0.839815
Test Epoch: 100 Accuracy: 0.858333
Test Epoch: 150 Accuracy: 0.869444
Test Epoch: 200 Accuracy: 0.856481
Test Epoch: 250 Accuracy: 0.854630
```

Batch Size 過大，不同 batch 間的梯度沒有變化，容易陷入區域性最小值。

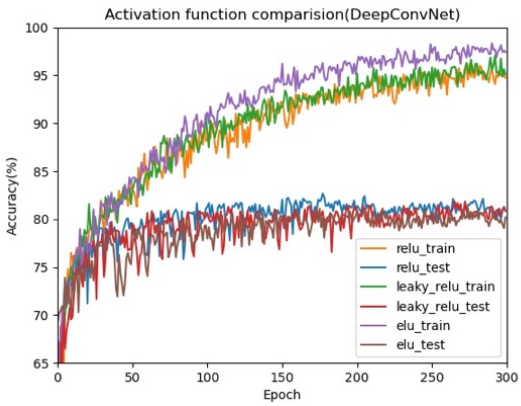
```
Test Epoch: 0 Accuracy: 0.614815
Test Epoch: 50 Accuracy: 0.801852
Test Epoch: 100 Accuracy: 0.825000
Test Epoch: 150 Accuracy: 0.837037
Test Epoch: 200 Accuracy: 0.848148
Test Epoch: 250 Accuracy: 0.846296
```

B. Comparison figures

- EEGNET



- DeepConvNet



4. Discussion

A. Anything you want to share

(1) Dead ReLU Problem

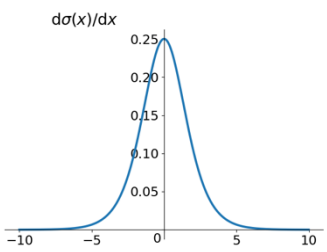
在訓練過程中，由於一次梯度更新的幅度過大，導致某些 ReLU 節點的權重調整的太大，使得後續的訓練對該節點不再起作用，這個節點相當於永久 dead 了。

舉個例子解釋，假設大多數輸入 ReLU 的值 w 是正數，則大多數經過 ReLU 之後都可以得到一個正值，此時稱 ReLU is open，而大多數都能透過反向傳播來得到一個梯度來更新參數。
但若有一個很大的梯度經過 ReLU，由於 ReLU 是打開的，因此會又一個很大的梯度傳給輸入 w，這會引起 w 的分佈產生巨大變化，若分佈改變使大多數輸入 ReLU 的值是負數，則大多數經過 ReLU 之後都會得到 0，此時稱 ReLU is closed，參數無法更新。

(2) Gradient Vanishing Problem

在使用 gradient descent 和 backpropagation 進行神經網路權重更新時，先計算輸出層對應的 loss，然後將 loss 以導數的形式不斷向前一層網絡進行傳遞，並修正相應的權重參數，達到降低 loss 之目的。而某些 activation function 權重更新時，常會因為層數過多，導致導數逐漸變為 0，使得前幾層之權重參數無法順利更新，造成神經網絡無法被優化，而無法找出最佳結果。

Sigmoid: 下圖為 Sigmoid 的微分圖形，最大值為 1/4，且當 input 在 [-4, +4] 之間，為分值趨近於 0，而產生梯度消失的問題，因此在 backpropagation 時，無法有效更新權重。



ReLU: 下圖為 ReLU 的微分圖形，分段線性性質能避免梯度消失的問題。