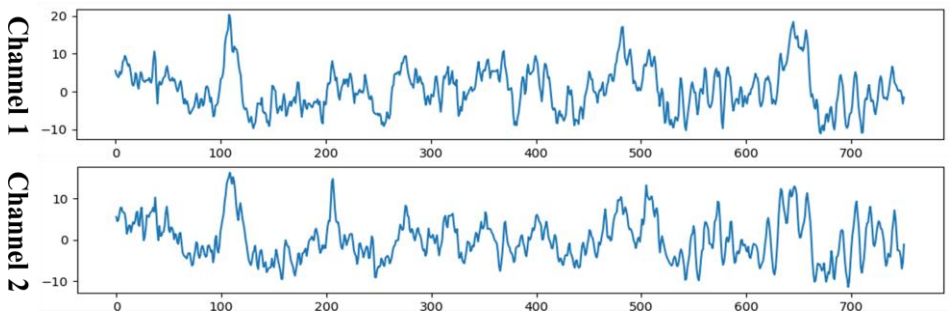
Lab4-1 : EEG classification

310605005 王映勻

1. Introduction

在這次 lab 中，需要利用 pytorch 實作出 EEGNet 和 DeepConvNet 兩種分類模型，並應用於分類 BCI (Brain-Computer Interface) competition 資料集。其資料集是基於視覺誘發的左右手運動想像，分為兩個 channel，各有 750 個資料點，標記為左手和右手兩種類別。

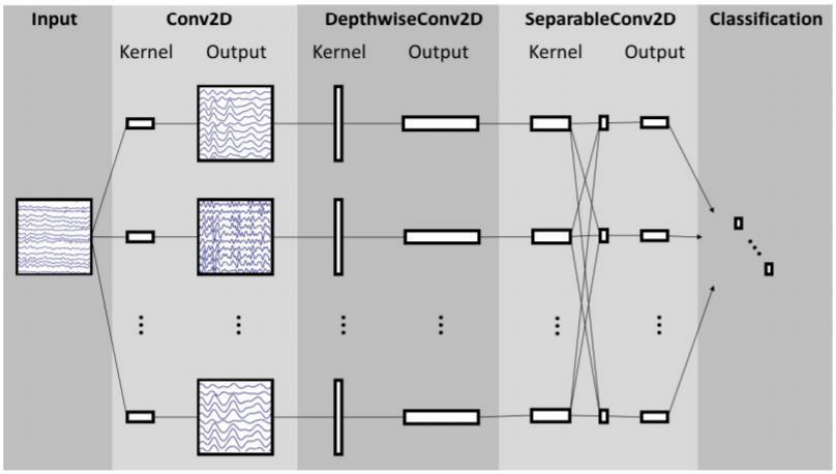


另外，還需要嘗試三種不同的 activation function(ReLU, Leaky ReLU, ELU)，利用視覺化的方法討論並比較實驗結果。

2. Experiment set up

A. The detail of your model

- ✓ **EEGNet**



EEGNet 是一種輕量化的 CNN 架構，大幅降低使用的參數數量，有效提昇訓練速度，且能在數據資源有限的情況下仍得到不錯的結果。如上圖所示，其架構大致可分成三個卷積層，由 1 個普通卷積 Conv2D + 1 個 DepthwiseConv2D + 1 個 SeparableConv2D 組成，參數設定如下：

```
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)
  )
)
```

(1) Conv2D 時間卷積層

這一層的目的是提取時間維度上的信息，學習 frequency filters。

Conv2D 的參數依序為輸入通道數、輸出通道數、卷積核大小、卷積核移動步長、Padding 層數和是否學習 bias。

BatchNorm2D 的參數則是特徵數(輸入通道數)、為分數值穩定而添加到分母的值、均值和方差的估計參數、是否有可學習的仿射變換參數以及是否跟蹤當前 batch 的統計特性。

(2) DepthwiseConv2D 深度卷積層

與第一層輸出的 feature map 相接，針對每個通道學習 frequency specific spatial filters。這種方法雖然能減少卷積層參數數量，但沒辦法利用不同通道在相同空間位置上的 feature 信息。

這邊的 Conv2D 多了一個 groups 的參數，值與輸入通道數設定一樣(16)，意思就是每個輸入通道各自為一組，也就是一組輸出的通道對應一組輸入的通道，不像一般卷積每個通道經過卷積核後會相加。

接著的 ELU 為 activation function，可換成 ReLU, Leaky ReLU 等。

而池化層這邊是使用 AvgPool2D 做平均池化，池化層的作用是對提取到的特徵信息進行降維，簡化網絡計算複雜度，且藉由特徵壓縮提取主要特徵。平均池化是減少資料兩個軸上的維度，並且以其的平均值來代替被減少的部分。

最後的 Dropout 是為了防止 overfitting 的問題，其 p=0.25 的意思是該層的神經元在每次迭代訓練時會隨機有 25% 的可能性被丟棄。

(3) SeparableConv2D 深度可分離卷積層

由一個 Depthwise Conv 和一個 Pointwise Conv 所組成，分別總結每個特徵映射，再最佳化合併輸出。Depthwise Conv 即為前面所提的深度卷積，而 Pointwise Conv 為卷積核大小 1*1 的普通卷積，其作用是要擴展 feature map 的數目，且將上一步生成的 feature map 在空間維度進行加權組合，彌補 Depthwise Conv 的缺點。

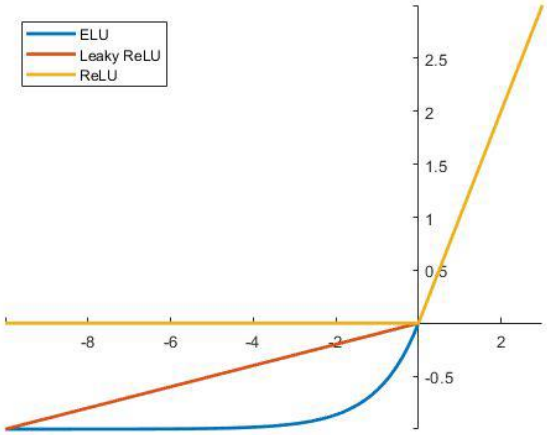
✓ **DeepConvNet**

DeepConvNet 即為傳統的 CNN 架構，由一個卷積層再加上四組 (Convolution, Batch Normalization, Activation function, Pooling, Dropout)，最後全連接層利用 softmax 函數進行分類。此次使用的模型架構參數如下 (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)

激活函數 Activation function 的目的是為了讓神經網路能實現非線性輸出，下面為 ReLU, Leaky ReLU, ELU 三種激活函數的圖形：



(1) ReLU （Rectified Linear Unit）

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

ReLU 只保留正值，負值則設為 0。

優點

- ✓ 非負區間的梯度為常數，不存在梯度消失問題
- ✓ 計算簡單，運算速度快

缺點

- ✓ 輸出不是以 0 為中心
- ✓ Dead ReLU Problem：由於在負區間梯度為 0，造成某些神經元可能永遠不會被激活，導致相應參數不會被更新

(2) Leaky ReLU

$$f(x) = max(0, x) + \alpha * min(0, x) \ (\alpha \ is \ a \ small \ value)$$

LeakyReLU 的目的是為了避免激活函數不處理負值，乘上一個微小的參數 α，使網絡可以傳遞負值部分的梯度。

優點

- ✓ 解決 Dead ReLU Problem

缺點

✓ 不是單側飽和的函數，可能會較難收斂

(3) ELU （Exponential Linear Units）

$$f(x) = max(0, x) + min\big(0, \alpha * (exp(x) - 1)\big)\ (\alpha\ is\ a\ small\ value)$$

ELU 是 ReLU 的另一種改良方法，解決了以上兩種激活函數的缺點。

優點

✓ 解決 Dead ReLU Problem

✓ 輸出均值為 0

✓ 函數全部區間為一階可導

缺點

✓ 包含指數運算，存在運算量較大的問題

3. Experimental results

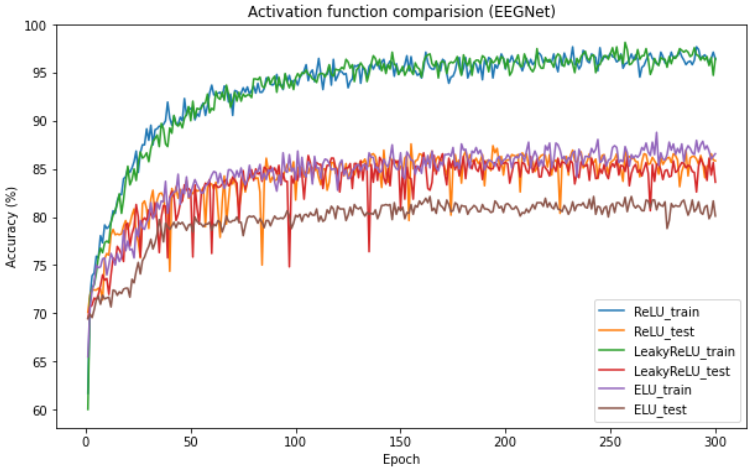
A. The highest testing accuracy

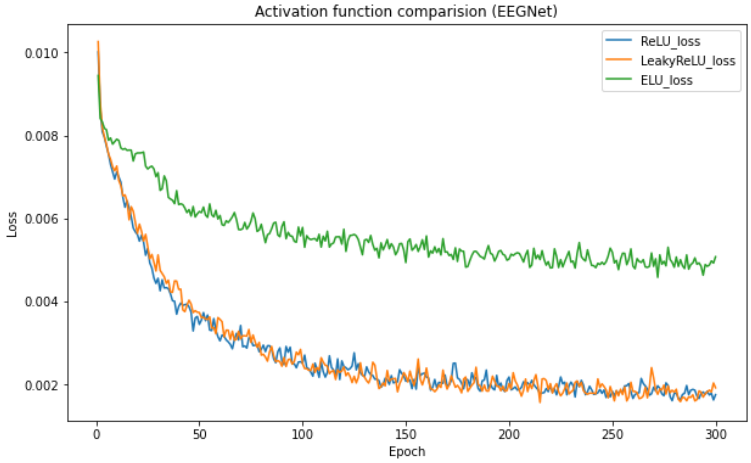
	ReLU	**Leaky ReLU**	**ELU**
EEGNet	87.59%	86.67%	82.13%
DeepConvNet	85.46%	85.19%	83.15%

B. Comparison figures

✓ **EEGNet**

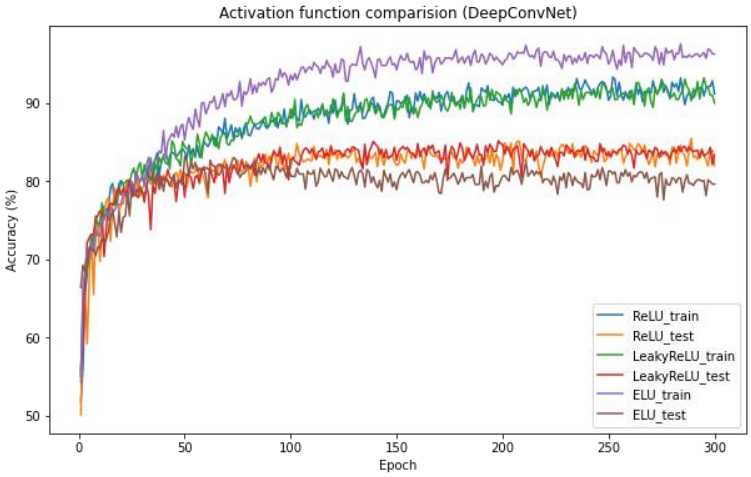
```
ReLU_train          max acc: 97.69 %
ReLU_test           max acc: 87.59 %
LeakyReLU_train     max acc: 98.15 %
LeakyReLU_test      max acc: 86.67 %
ELU_train           max acc: 88.80 %
ELU_test            max acc: 82.13 %
```

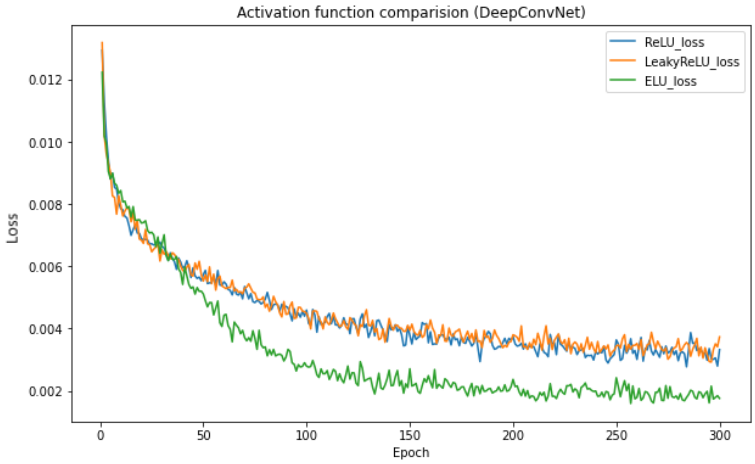




✓ DeepConvNet

```
ReLU_train              max acc: 93.33 %
ReLU_test               max acc: 85.46 %
LeakyReLU_train         max acc: 93.24 %
LeakyReLU_test          max acc: 85.19 %
ELU_train               max acc: 97.59 %
ELU_test                max acc: 83.15 %
```





4. Discussion

(1) Activation function 激活函數效果比較

ReLU 是最常用的激活函數，形式簡單且高度非線性，而 Leaky ReLU 和 ELU 皆為其改善的版本。雖然理論上來說，Leaky ReLU 和 ELU 有 ReLU 的所有優點，也不會有 Dead ReLU Problem，但是在實際操作當中，並沒有完全證明它們總是好於 ReLU。

從這次的實驗結果也可觀察到 ELU 反而是效果最差的，而我認為有可能是因為這次資料並不複雜，所以三者效果差不多，而 Leaky ReLU 和 ELU 因為有負數的輸出，導致其非線性程度沒有 ReLU 強大，且函數中的 α 參數，需要通過先驗知識人工賦值，也可能影響結果。

(2) Softmax 輸出問題

依據 DeepConvNet 的參數設定表，最後應使用 softmax 輸出結果。但因結果比預期的差，仔細看 pytorch 官網關於 CrossEntropyLoss 的介紹，才發現原來 CrossEntropyLoss 已經包含了一個 logsoftmax 的函數，因此 loss 函數的輸入也就是模型的輸出應該要是未做過 softmax 的結果才合理。否則經過兩次 softmax 的運算後，權重可能會被過度壓縮，導致模型更新不易。

(3) Weight decay

在 Adam optimizer 的設定中有一個 weight decay 的參數，其作

用是保持權重在一個較小的值，避免梯度爆炸的問題，類似於 L2 Regularization 的效果，目的是防止模型 overfitting。

從上面的結果圖可以看出大概訓練 100 個 epoch 之後，模型就已經接近飽和，有可能出現 overfitting 的問題，因此引入 weight decay 參數後，透過測試可發現正確率有大幅提升。