DLP Lab2 EEG classification

0711529 陳冠儒

1. Introduction (20%)

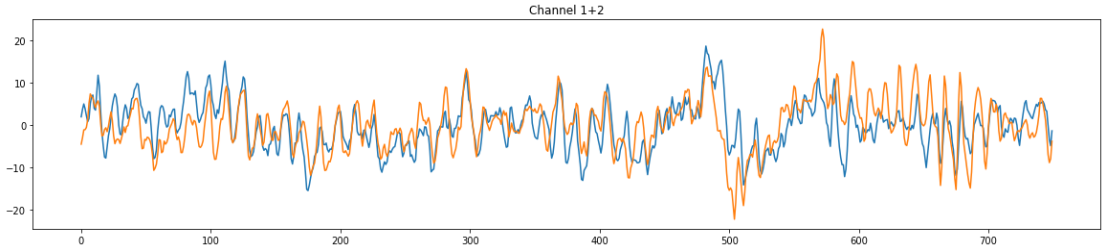
   使用 EEGNet 跟 DeepConvNet 搭配三種不同的 activation function (ReLU、LeakyReLU、ELU)，去比較 acc 的影響。

   A. Dataset：

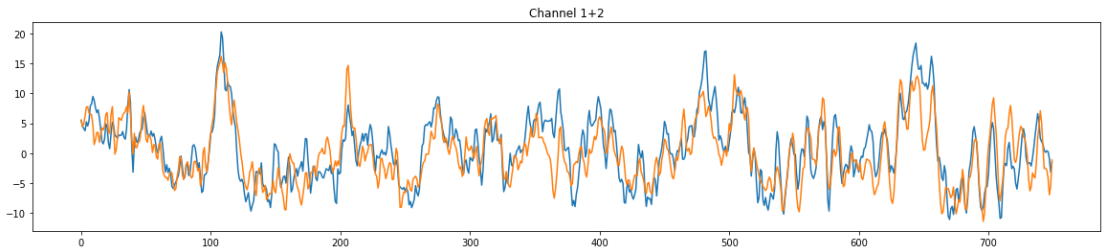
   這次使用的是 BCI Competition III 的 IIIb dataset，為一個用 125 Hz 採樣的 ERP (Event-Related Potential) EEG dataset，他是一個 non-stationary 的 classifier (e.g., time-varying)，並用兩個 channel 的 EEG 圖來預測是左邊還是右邊。

   - Training and Testing data
     - ✓ X_train：shape (1080, 1, 2, 750)，1080 筆資料，兩個 channel，一個 channel 750 時間單位的訊號。
     - ✓ y_train：shape (1080,)，1080 筆資料，每個資料為 0 或是 1 代表左邊或是右邊。
     - ✓ X_test：shape (1080, 1, 2, 750)
     - ✓ y_test：shape(1080,)

   - Data example
     - ✓ Label = 0



     - ✓ Label = 1



   - Reference

[1] Dataset IIIb: Non-stationary 2-class BCI data: http://www.bbci.de/competition/iii/desc_IIIb.pdf

B. EEGNet 的優點

在 EEGNet 的論文中提到了 EEGNet 的三個優點：

(1) 可以應用於多種不同的 BCI 模式

在 paper 中他總共使用了四種不同的 EEG dataset 來進行測試，包含了最主要兩種 BCI 的資料種類：ERP (Event-Related Potential)和 Oscillatory，在這四種上都能達到不錯的成效，且每一個都包含不同數量的數據，使我們能夠探索 EEGNet 在各種訓練數據大小上的功效。

(2) 可以使用非常有限的數據進行訓練

使用 Depthwise 和 Separable Convolutions 來構建，讓可訓練參數大大的減少，故只要小的 dataset 就可以訓練起來。

(3) 可以產生神經生理學上可解釋的特徵

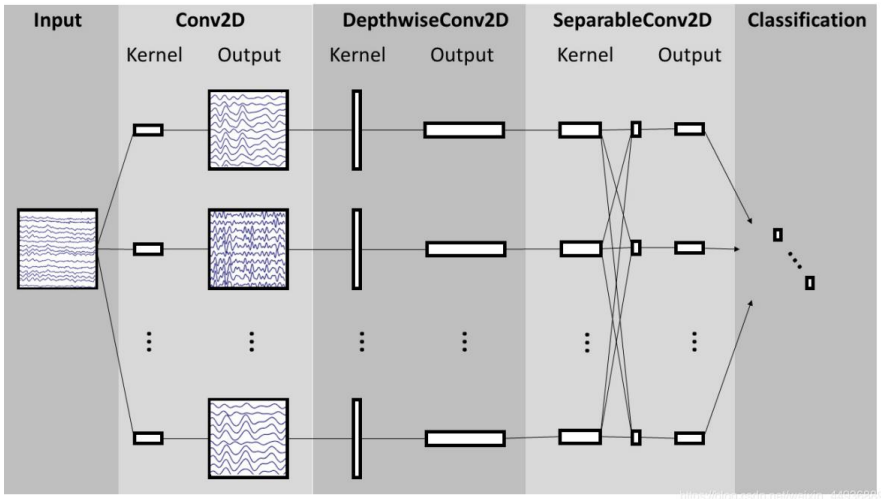
儘管 CNN 具有強大的自動特徵提取能力，但通常會產生難以解釋的特徵。對於神經科學家來說，深入了解 CNN 衍生的神經生理現象的能力可能與實現良好的分類性能同樣重要。paper 中驗證了 EEGNet 架構在幾個經過充分研究的 BCI 模式上提取神經生理學可解釋信號的能力，以表明網絡性能不是由數據中的 noies 或人工信號驅動。

2. Experiment setups(30%):

A. The detail of your model

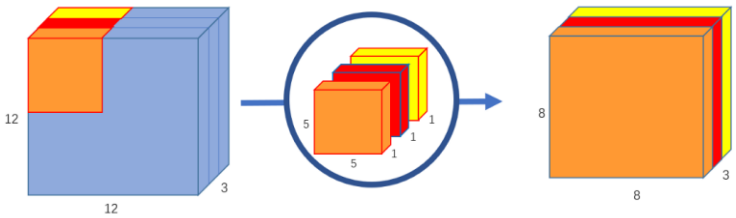
◆ EEGNet

EEGNet 可以主要可以分成三層：Conv2D、DepthwiseConv2D、SeparableConv2D，下面將會對這三層提取的 feature 進行分析。



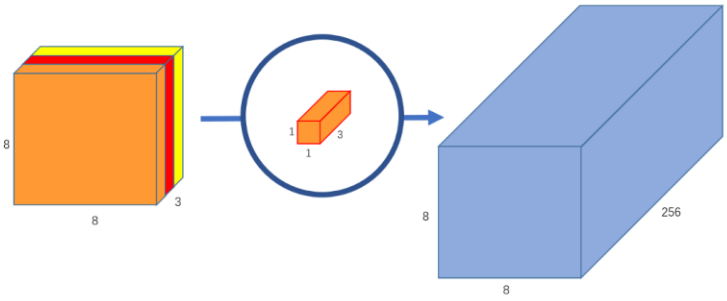
- Conv2D：利用時間卷積(temporal convolution)學習頻率過濾器(frequency filters)，得到的各個 feature map 對應到各特定頻率。
- DepthwiseConv2D：使用深度卷積(depthwise convolution)，單獨連接到每個 feature map，以學習特定於頻率的空間過濾器(frequency-specific spatial filters)，而這裡的 depth parameter D 是表要為每個 feature map 學習的 spatial filters 數。

  ※ DepthwiseConv2D



  將 input 的各個 channel 單獨學習，一個 channel 會對應到一個(D=1)或多個(D>1)kernel 進行 convolution。
- SeparableConv2D：可分離卷積(separable convolution)是深度卷積與逐點捲積的組合，其中的深度捲積學習每個 feature map 的時間摘要，而最後再由逐點捲積(pointwise convolution)學習如何將 feature map 最佳的組合在一起。

  ※ PointwiseConv2D



使用 1x1 kernel 增加 image 的 depth，即增加 channel 的數量。

- Model Neural Network

```
EEGNet(
  (conv1): Conv2d(1, 16, kernel_size=(1, 51), stride=(1, 1), padding=(0, 25), bias=False)
  (batchnorm1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (depthwise1): DepthwiseConv2d(16, 32, kernel_size=(2, 1), stride=(1, 1), groups=16, bias=False)
  (batchnorm2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu1): ReLU()
  (averagePooling1): AvgPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0)
  (dropout1): Dropout(p=0.2, inplace=False)
  (separableconv1): SeparableConv2d(
    (depthwise): Conv2d(32, 32, kernel_size=(1, 15), stride=(1, 1), padding=(0, 7), groups=32, bias=False)
    (pointwise): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
  )
  (batchnorm3): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu2): ReLU()
  (averagePooling2): AvgPool2d(kernel_size=(1, 8), stride=(1, 8), padding=0)
  (dropout2): Dropout(p=0.2, inplace=False)
  (flatten1): Flatten(start_dim=1, end_dim=-1)
  (dense1): Linear(in_features=736, out_features=2, bias=True)
)
```

- Model layer shape and parameters with input shape (-1, 1, 2, 750)

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 16, 2, 750]             816
       BatchNorm2d-2          [-1, 16, 2, 750]              32
  DepthwiseConv2d-3          [-1, 32, 1, 750]              64
       BatchNorm2d-4          [-1, 32, 1, 750]              64
              ReLU-5          [-1, 32, 1, 750]               0
         AvgPool2d-6          [-1, 32, 1, 187]               0
           Dropout-7          [-1, 32, 1, 187]               0
            Conv2d-8          [-1, 32, 1, 187]             480
            Conv2d-9          [-1, 32, 1, 187]           1,024
 SeparableConv2d-10          [-1, 32, 1, 187]               0
      BatchNorm2d-11          [-1, 32, 1, 187]              64
             ReLU-12          [-1, 32, 1, 187]               0
        AvgPool2d-13           [-1, 32, 1, 23]               0
          Dropout-14           [-1, 32, 1, 23]               0
          Flatten-15                 [-1, 736]               0
           Linear-16                   [-1, 2]           1,474
================================================================
Total params: 4,018
Trainable params: 4,018
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 1.25
Params size (MB): 0.02
Estimated Total Size (MB): 1.27
----------------------------------------------------------------
```

- DeepConvNet

  DeepConvNet 架構由五個卷積層和一個用於分類的 softmax 層組成。

  - Model Neural Network

```
DeepConvNet(
  (conv1): Conv2d(1, 25, kernel_size=(1, 5), stride=(1, 1))
  (conv2): Conv2d(25, 25, kernel_size=(2, 1), stride=(1, 1))
  (batchnorm1): BatchNorm2d(25, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu1): ReLU()
  (maxpool1): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
  (dropout1): Dropout(p=0.5, inplace=False)
  (conv3): Conv2d(25, 50, kernel_size=(1, 5), stride=(1, 1))
  (batchnorm2): BatchNorm2d(50, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu2): ReLU()
  (maxpool2): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
  (dropout2): Dropout(p=0.5, inplace=False)
  (conv4): Conv2d(50, 100, kernel_size=(1, 5), stride=(1, 1))
  (batchnorm3): BatchNorm2d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu3): ReLU()
  (maxpool3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
  (dropout3): Dropout(p=0.5, inplace=False)
  (conv5): Conv2d(100, 200, kernel_size=(1, 5), stride=(1, 1))
  (batchnorm4): BatchNorm2d(200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu4): ReLU()
  (maxpool4): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
  (dropout4): Dropout(p=0.5, inplace=False)
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (dense): Linear(in_features=8600, out_features=2, bias=True)
)
```

  - Model layer shape and parameters with input shape (-1, 1, 2, 750)

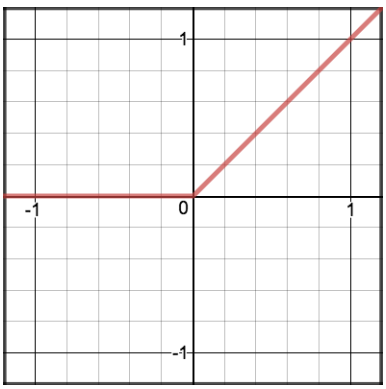
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 25, 2, 746]             150
            Conv2d-2          [-1, 25, 1, 746]           1,275
       BatchNorm2d-3          [-1, 25, 1, 746]              50
              ReLU-4          [-1, 25, 1, 746]               0
         MaxPool2d-5          [-1, 25, 1, 373]               0
           Dropout-6          [-1, 25, 1, 373]               0
            Conv2d-7          [-1, 50, 1, 369]           6,300
       BatchNorm2d-8          [-1, 50, 1, 369]             100
              ReLU-9          [-1, 50, 1, 369]               0
        MaxPool2d-10          [-1, 50, 1, 184]               0
          Dropout-11          [-1, 50, 1, 184]               0
           Conv2d-12         [-1, 100, 1, 180]          25,100
      BatchNorm2d-13         [-1, 100, 1, 180]             200
             ReLU-14         [-1, 100, 1, 180]               0
        MaxPool2d-15          [-1, 100, 1, 90]               0
          Dropout-16          [-1, 100, 1, 90]               0
           Conv2d-17          [-1, 200, 1, 86]         100,200
      BatchNorm2d-18          [-1, 200, 1, 86]             400
             ReLU-19          [-1, 200, 1, 86]               0
        MaxPool2d-20          [-1, 200, 1, 43]               0
          Dropout-21          [-1, 200, 1, 43]               0
          Flatten-22                [-1, 8600]               0
           Linear-23                   [-1, 2]          17,202
================================================================
Total params: 150,977
Trainable params: 150,977
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.01
Forward/backward pass size (MB): 2.56
Params size (MB): 0.58
Estimated Total Size (MB): 3.14
----------------------------------------------------------------
```

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

- ReLU

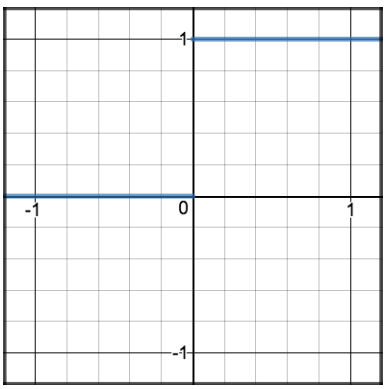
Function

$$\mathrm{ReLU}(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}$$



Derivative

$$\mathrm{ReLU}'(x) = \begin{cases} 1, x > 0 \\ 0, x \leq 0 \end{cases}$$



在 0 處不可微，但可以直接設為 0 或是 1，ReLU 是目前最被廣泛使用的 activation function。

➢ 優

a. ReLU 在$x > 0$的微分值為 1，所以他沒有 gradient vanishing 的問題（gradient vanishing 是因為 gradient ＜ 1，梯度更新後的結果將會越來小，最後趨近於 0）。

b. ReLU 在$x > 0$的微分值為 1，可以降低 gradient explosion

的問題（gradient explosion 是因為 gradient> 1而使得梯度更新後其值越來越大，最後超過可計算範圍）。

c. 沒有複雜的計算，所以計算成本低，很快就能收斂。

➢ 缺

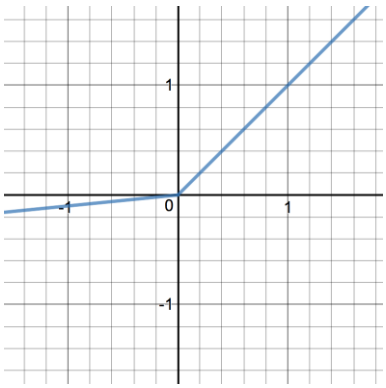
a. 「dying ReLU problem」，在$x \leq 0$處梯度為 0，因此在下降期間不會調整權重，在此狀態的神經元將停止響應錯誤和輸入的變化。

b. ReLU 只是降低 gradient explosion 但因為其範圍是[0, inf)，所以還是可能會發生。

◆ Leaky ReLU

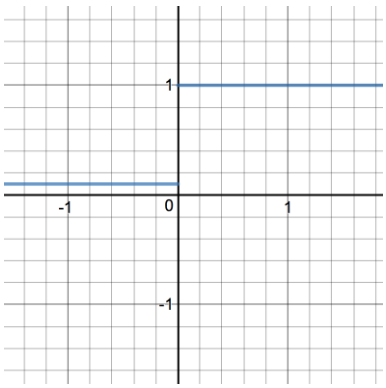
Function

$$\text{LeakyReLU(x)} = \begin{cases} x, x > 0 \\ \alpha x, x \leq 0 \end{cases}$$



Derivative

$$\text{LeakyReLU}'\text{(x)} = \begin{cases} 1, x > 0 \\ \alpha, x \leq 0 \end{cases}$$



在負值時有一由α控制的洩漏量，通常是設為 0.01，並且在 0 的地方也不可微，通常也會直接設為 1 或α。

➢ 優

a. 繼承了 ReLU 的所有優點。

b. Leaky ReLU 在負值時的α洩漏值解決了 dying ReLU problem。

c. 在《Empirical Evaluation of Rectified Activations in Convolution Network》這篇研究中發現 Leaky ReLU 的性能優於 ReLU。此外，洩漏值較高的 Leaky ReLU 的性能優於洩漏值較低的。（但是在不同的 model 上主要還是要看實驗結果來決定）

➢ 缺

a. α 值是在訓練之前定義的，因此在訓練期間無法調整，

因此選擇的α值可能不是最佳值。

- Reference

  [1] Empirical Evaluation of Rectified Activations in Convolution Network: https://arxiv.org/pdf/1505.00853.pdf

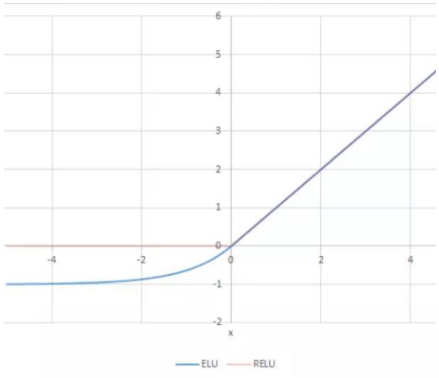
◆ ELU (Exponential Linear Unit)

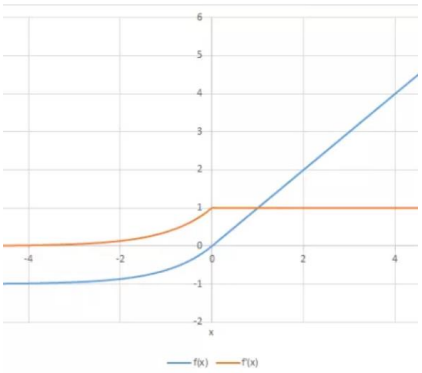
Function

$$ELU(x) = \begin{cases} x, x > 0 \\ \alpha(e^x - 1), x \leq 0 \end{cases}$$

Derivative

$$ELU'(x) = \begin{cases} 1, x > 0 \\ \alpha e^x, x \leq 0 \end{cases}$$





與 Leaky ReLU 在負值都有α參數控制洩漏值大小，但 Leaky ReLU 在 0 的地方不可微，而 ELU 在所有地方都是連續可微的。

- 優
  a. 繼承所有 Leaky ReLU 的優點，且微分後仍為 non-linear function。

- 缺
  a. 在 function 中有 exponential，造成其計算成本比 ReLU 與 Leaky ReLU 還大，因此收斂較慢。
  b. α值也是在訓練之前定義的，因此在訓練期間無法調整，因此選擇的α值可能不是最佳值。

◆ Reference

[1] Activation Functions — ML Glossary documentation: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html

3. Experimental results (30%)
   A. The highest testing accuracy
      ◆ Screenshot with two models

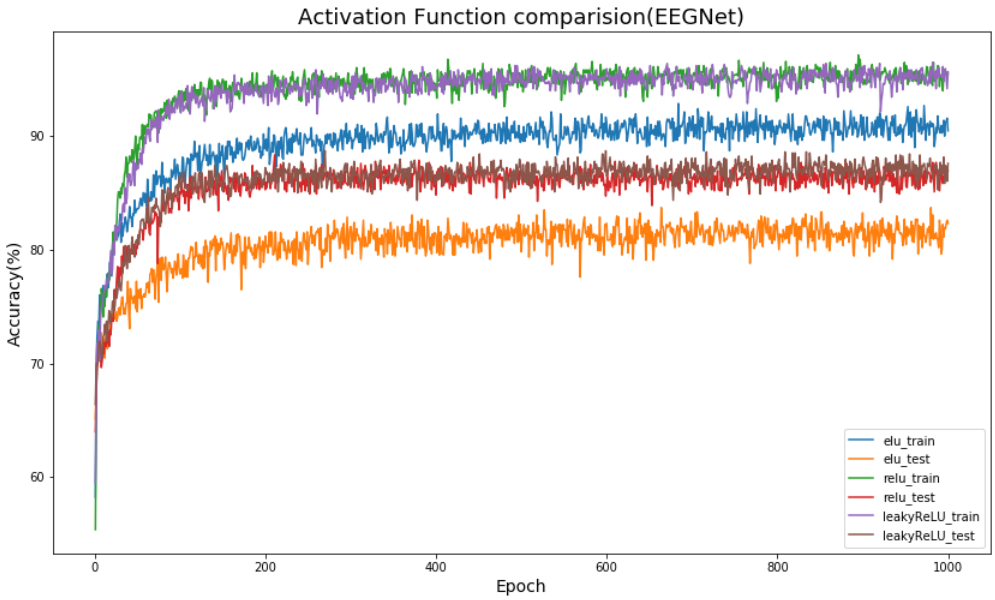
Hyper parameters：

- Optimizer: Adam
- Loss function: torch.nn.CrossEntropyLoss()
- epoch of EEGNet: 1000, epoch of DeepConvNet: 2000
- batch: 128
- learning rate: 1e-3
- weight decay: 0.02

	ReLU	LeakyReLU	ELU
EEGNet	89.074074%	88.240741%	82.685185%
DeepConvNet	83.703704%	83.425926%	82.129630%

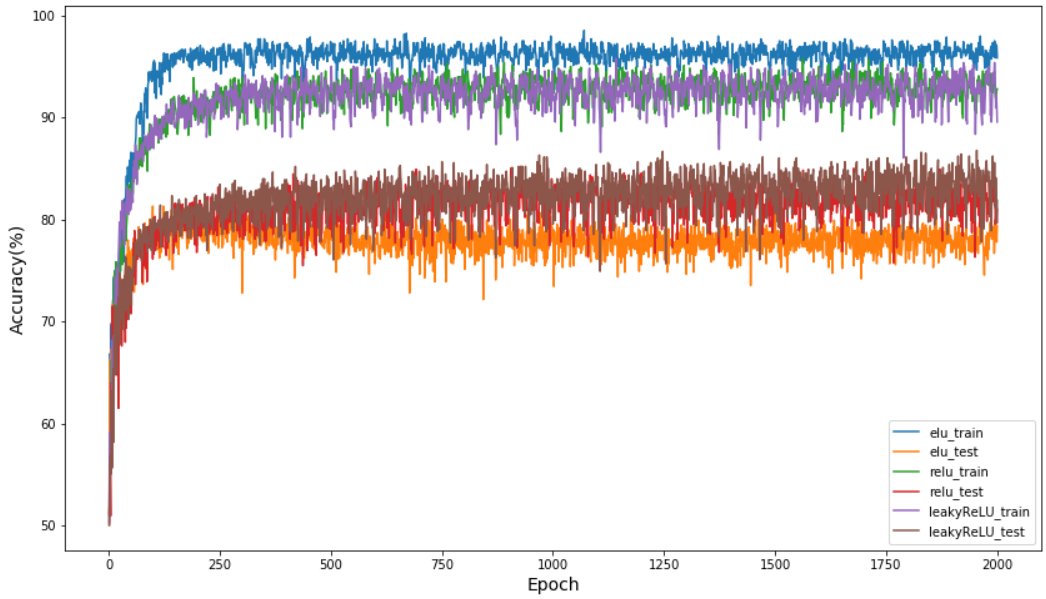
B. Comparison figures

◆ EEGNet



Testing acc: ReLU > Leaky ReLU > ELU

◆ DeepConvNet



Testing acc: ReLU > Leaky ReLU > ELU

4. Discussion (20%)
   A. Anything you want to share
      - 為什麼在這個 Lab 使用 DeepConvNet 與 EEGNet 進行比較而不是 ShallowConvNet？
        因為 DeepConvNet 是一種通用架構，不限於特定的特徵類型，因此它可以作為與 EEGNet 的更有效的比較；而 ShallowConvNet 架構是專門為振盪信號分類而設計的（通過提取與對數帶功率相關的特徵）；因此，它可能不適用於基於 ERP (此 dataset)的分類任務。

      - DeepConvNet 與 EEGNet 比較結果
        論文[1]中說 DeepConvNet 和 EEGNet 的分類性能在所有 cross-subject 分析中都相似，而 DeepConvNet 在幾乎所有 within-subject 分析中的性能都較低。這種差異的一種可能解釋是用於訓練模型的訓練數據量；在 cross-subject 分析中，訓練集大小大約是 within-subject 分析的 10-15 倍。這表明與 EEGNet 相比，Deep-ConvNet 的數據密集程度更高，鑑於 DeepConvNet 的模型大小比 EEGNet 大兩個數量級，這一結果並不令人意外。

        從本文第二點的 Detail of Model 可以得知本次實驗用的 EEGNet 和 DeepConvNet 的可訓練參數分別為 4018 跟 150977，可以看出這兩個網路大小相差很大，且這次的 dataset 數據量也不算大，因此得到 EEGNet 都優於 DeepConvNet 的結果是不意外的。

- ReLU、Leaky ReLU 與 ELU 結果比較

  實驗結果不論是在 EEGNet 或是 DeepConvNet 的 testing accuracy 都是 ReLU >= LeakyReLU > ELU，我覺得是因為 parameter 與資料集大小的關係，這個結果大小順序正好與 trainable parameter 量相反，因為這次的資料集大小不算大，所以越多的 parameter 會越難訓練起來。

- Reference

[1] EEGNet: A Compact Convolutional Neural Network for EEG-based Brain-Computer Interfaces: https://arxiv.org/pdf/1611.08024.pdf

[2] Deep Learning With Convolutional NeuralNetworks for EEG Decoding and Visualization: https://onlinelibrary.wiley.com/doi/epdf/10.1002/hbm.23730