Deep Learning Lab3 – EEG classification Report

智能所 311581006 林子淩

1. Introduction

本次 lab 主題為實作 EEG signal 分類任務。考量到 Convolution Network 在影像特徵提取(Feature extraction)和影像分類(Classification)任務上的優秀表現，本次 lab 要求建構兩個簡單的 CNN-based model 用來分類 EEG signal，分別為 EEGNet 和 DeepConvNet。兩個模型皆使用 BCI competition dataset 來進行訓練和測試，並比較在三種不同的 activation function (ReLU, Leaky ReLU, ELU)設定之下，模型的 train/test accuracy 數據和 learning curve 的變化。

2. Experiment Setup

A. The detail of your model

本次 lab 的 model 實作細節皆按照標準架構，考量到程式碼截圖比較冗雜，這裡只介紹模型搭建內容，詳細的 CNN 實作參數則記錄於 Table 1。程式碼請另外參考 source code。

- **EEGNet**

Figure 1 為 EEGNet 的整體實作架構，將提取特徵的過程分成三個部分。第一部分 Conv2D 使用普通的 2D CNN 架構從 Input 生成多種不同的 feature maps。接著，第二部分 DepthwiseConv2D 的 CNN kernel 連接到上層生成的每一個 feature map，用來學習 frequency-specific spatial filters (kernels)。第三部分 SeparableConv2D 則是先學習上層輸出 feature maps 的時間摘要(time summary)，再通過 point-wise convolution 學習如何將這些特徵混合到一起。最後，模型再根據 output feature maps 進行分類。EEGNet 的 convolution modules 與對應的 input, output channel 和 kernel size 記錄於後面 Table 1 中。

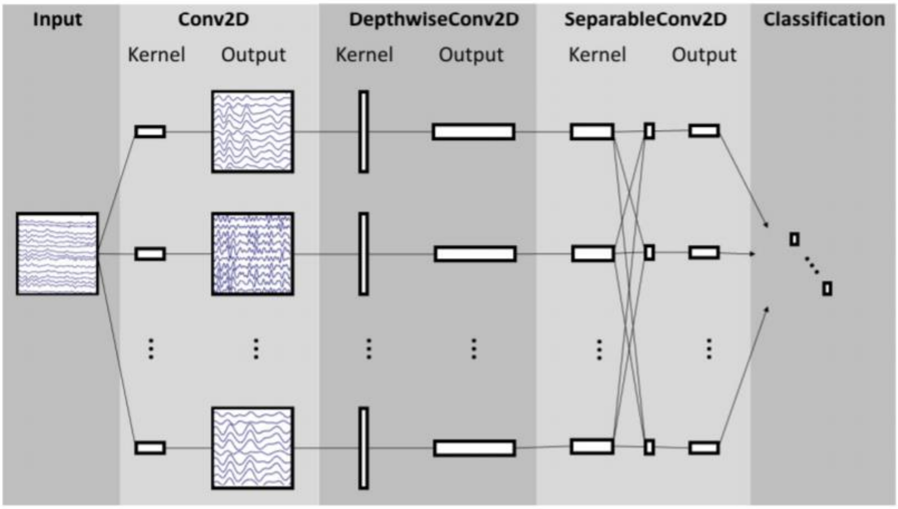

Figure 1 EEGNet 整體架構

- **DeepConvNet**

Figure 2 為 DeepConvNet 的整體實作架構，其中參數 C = 2、T = 750、N = 2。根據圖中描述建構出的模型可以分成六個部分。第一部分為單一層 2D CNN，初步從 Input 中提取特徵。第二到第五部分為四個重複的 convolution modules，每一個 module 按照順序由 2D CNN, Batch Normalization, activation function ,2D Max Pooling, Dropout layer 組成，每一部分的 input, output channel 和 kernel size 略有差異。最後第六部分為一層 fully-connected layer，用於輸出分類結果。

DeepConvNet 的 convolution modules 與對應的 input, output channel 和 kernel size 同樣記錄於後面 Table 1 中。

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

Figure 2　DeepConvNet 整體架構

底下 Table 1 記錄 EEGNet 和 DeepConvNet 的實作參數細節，因為本次模型實作重點在 CNN 架構上，因此 Module 只記錄 CNN 相關部分，其他 layer 不特別記錄在表格中。

Model	Module	Input Channel	Output Channel	Kernel Size
EEGNet	Conv2D	1	16	(1,51)
	DepthwiseConv2D	16	32	(2,1)
	SeparableConv2D	32	32	(1,15)
DeepConvNet	Conv2D	1	25	(1,5)
	Conv2D Block 1	25	25	(2,1)
	Conv2D Block 2	25	50	(1,5)
	Conv2D Block 3	50	100	(1,5)
	Conv2D Block 4	100	200	(1,5)

Table 1　EEGNet 和 DeepConvNet 的實作參數細節

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

本次 lab 中實作了三種 activation function，即 ReLU、Leaky ReLU、ELU。

- **ReLU**

Rectifed **L**inear **U**nits (ReLU)是一種在深度學習模型中常被使用的 activation function。作為一種映射函數，當 x>0 時，映射結果 y 維持原 x 值不變；當 x≤0 時，y 則為 0。相較於 Sigmoid derivative 在 x=0 附近時才有比較高的數值，容易產生梯度消失的問題。ReLU derivative 在 x>0 時導數值皆為 1，不會導致梯度消失；然而當 x≤0 時，導數值為 0 會使得在更新參數時，輸出為值≤0 的神經元不會被更新到，因為梯度(ReLU derivative)為 0。ReLU 和 ReLU derivative 的公式如下，圖形如 Figure 3：

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

$$\text{Derivative of ReLU function: } ReLU'(x) = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases}$$

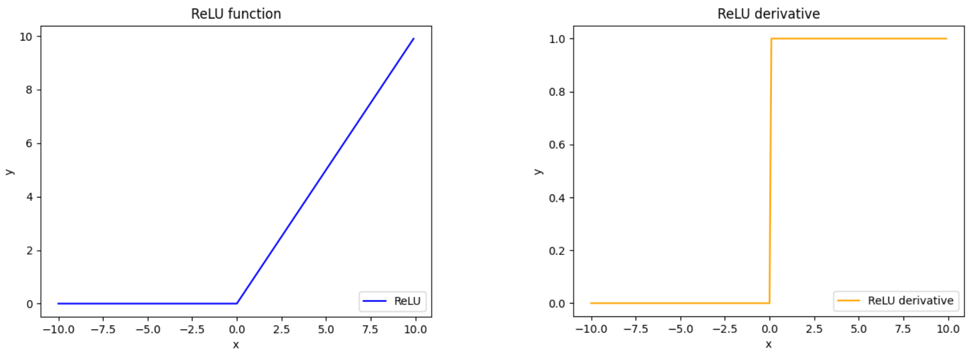

Figure 3　ReLU 和 ReLU derivative 圖形

- **Leaky ReLU**

前面 ReLU 介紹中提到，ReLU derivative 在 x≤0 時導數值為 0，會讓輸出≤0 的神經元梯度為 0 無法再繼續更新，造成所謂的 dead ReLU problem。為了解決以上問題，因此誕生了一種新的 activation function Leaky ReLU。作為改良版的 ReLU function，Leaky ReLU 通過控制參數 α (代表斜率，範圍需介於 0 到 1 之間)，調整映射函數中 x≤0 (左半邊)的斜率使其不為 0。這使得在 x≤0 情況下，Leaky ReLU 的導數值會等於 α (≠0)，以解決 dead ReLU problem。Leaky ReLU 和 Leaky ReLU derivative 的公式如下，圖形如 Figure 4：

$$\text{Leaky ReLU function:}\ \ \mathrm{LeakyReLU(x)} = \begin{cases} \mathrm{x}\ ,\ \mathrm{x} > 0 \\ \alpha\mathrm{x},\ \mathrm{x} \leq 0 \end{cases}，0 \leq \alpha \leq 1$$

$$\text{Derivative of Leaky ReLU function:}\ \ \mathrm{Leaky}ReLU'(x) = \begin{cases} 1,\ \mathrm{x} > 0 \\ \alpha,\ \mathrm{x} < 0 \end{cases}$$

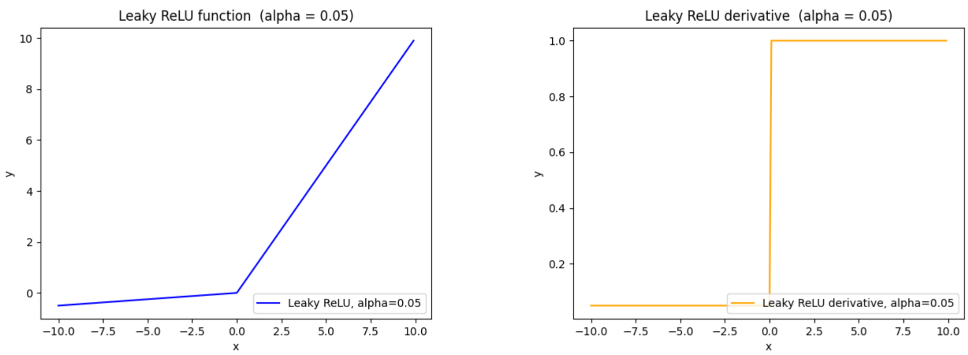

Figure 4　Leaky ReLU 和 Leaky ReLU derivative 圖形(α=0.05)

- **ELU**

Exponential **L**inear **U**nit (ELU)同為 ReLU function 的一種變形。和 Leaky ReLU 一樣，通過控制參數 α (介於 0 到 1 之間)，讓函數中 x≤0 (左半邊)的映射結果 y 不為 0，而是一個接近 0 的負值。與 ReLU 和 Leaky ReLU 不同的是，ELU 在 x=0 附近的變化較平滑，使整個函數可微分(differentiable)。ELU 也提供 non-linearity，提高模型的分類能力。ELU 和 ELU derivative 的公式如下，圖形如 Figure 5：

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

$$\text{Derivative of ELU function:}\ \ \mathrm{E}LU'(x) = \begin{cases} 1 & ,\ \mathrm{x} \geq 0 \\ \alpha\mathrm{e^x}, & \mathrm{x} < 0 \end{cases}$$

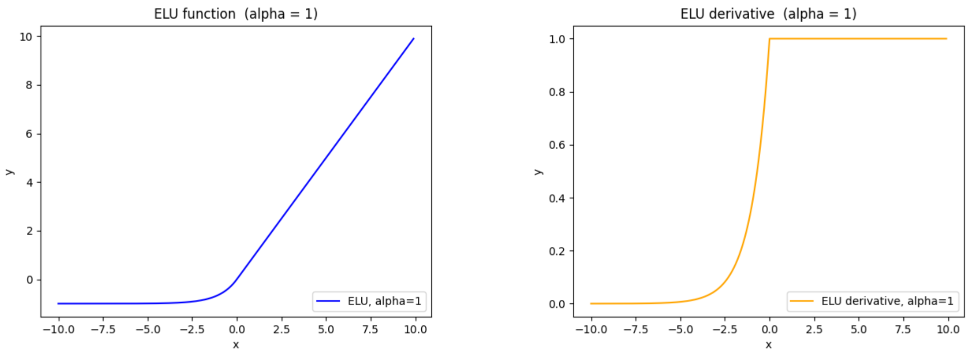

Figure 5　ELU 和 ELU derivative 圖形(α=1)

3. Experimental results

A. The highest testing accuracy

- **Screenshot with two models**

Table 2 中包含了 EEGNet 和 DeepConvNet 兩個模型訓練過程的截圖(選最好的 activation 和 learning rate 組合，完整數據截圖見 Figure 6)，記錄了整個 training 結果，包含 training loss, training accuracy 和 testing accuracy，最高的 testing accuracy 和出現該結果的 epoch 顯示在最後一行。根據第四部份 Discussion 中的 Table 3 數據，兩個模型能夠達到最高 testing accuracy 的實驗設定分別為(EEGNet, activation=RELU, learning rate=2e-3)和(DeepConvNet, activation=Leaky ReLU, learning rate=1e-2)，各達到了 88.70%和 82.78%的 testing accuracy。

其他實驗設定與對應的 testing accuracy 則紀錄於第四部份 Discussion 的 Table 3 中。

Highest testing accuracy	
EEGNet	**DeepConvNet**
eeg, 0.002, relu	deepconv, 0.01, leaky_relu
start training	start training
epoch 0, total train loss = 11.2328, train acc = 0.5694, test acc = 0.6981	epoch 0, total train loss = 215.8748, train acc = 0.5194, test acc = 0.6130
epoch 20, total train loss = 4.7377, train acc = 0.8907, test acc = 0.8120	epoch 20, total train loss = 9.2282, train acc = 0.7343, test acc = 0.7157
epoch 40, total train loss = 2.5816, train acc = 0.9370, test acc = 0.8398	epoch 40, total train loss = 7.5603, train acc = 0.7731, test acc = 0.7833
epoch 60, total train loss = 1.7321, train acc = 0.9630, test acc = 0.8519	epoch 60, total train loss = 6.5781, train acc = 0.8259, test acc = 0.7963
epoch 80, total train loss = 1.4711, train acc = 0.9630, test acc = 0.8593	epoch 80, total train loss = 5.7918, train acc = 0.8444, test acc = 0.8148
epoch 100, total train loss = 1.3045, train acc = 0.9769, test acc = 0.8546	epoch 100, total train loss = 5.8204, train acc = 0.8528, test acc = 0.8111
epoch 120, total train loss = 1.0017, train acc = 0.9824, test acc = 0.8537	epoch 120, total train loss = 5.1843, train acc = 0.8713, test acc = 0.8148
epoch 140, total train loss = 0.9598, train acc = 0.9806, test acc = 0.8648	epoch 140, total train loss = 5.1074, train acc = 0.8657, test acc = 0.8204
epoch 160, total train loss = 0.7811, train acc = 0.9824, test acc = 0.8639	epoch 160, total train loss = 4.6384, train acc = 0.8870, test acc = 0.8130
epoch 180, total train loss = 0.6405, train acc = 0.9833, test acc = 0.8620	epoch 180, total train loss = 4.2920, train acc = 0.8972, test acc = 0.8102
epoch 200, total train loss = 0.4641, train acc = 0.9898, test acc = 0.8704	epoch 200, total train loss = 3.9492, train acc = 0.9000, test acc = 0.8185
epoch 220, total train loss = 0.5188, train acc = 0.9861, test acc = 0.8565	epoch 220, total train loss = 4.1424, train acc = 0.9000, test acc = 0.7944
epoch 240, total train loss = 0.5425, train acc = 0.9870, test acc = 0.8806	epoch 240, total train loss = 4.6628, train acc = 0.9009, test acc = 0.8120
epoch 260, total train loss = 0.3588, train acc = 0.9935, test acc = 0.8759	epoch 260, total train loss = 4.0604, train acc = 0.8944, test acc = 0.8074
epoch 280, total train loss = 0.4327, train acc = 0.9917, test acc = 0.8685	epoch 280, total train loss = 4.0759, train acc = 0.8991, test acc = 0.8185
epoch 300, total train loss = 0.5652, train acc = 0.9870, test acc = 0.8741	epoch 300, total train loss = 3.6542, train acc = 0.9222, test acc = 0.8111
epoch 320, total train loss = 0.5162, train acc = 0.9898, test acc = 0.8722	epoch 320, total train loss = 3.6037, train acc = 0.9185, test acc = 0.8111
epoch 340, total train loss = 0.3584, train acc = 0.9935, test acc = 0.8583	epoch 340, total train loss = 4.3127, train acc = 0.9083, test acc = 0.8074
epoch 360, total train loss = 0.3841, train acc = 0.9907, test acc = 0.8704	epoch 360, total train loss = 3.3208, train acc = 0.9241, test acc = 0.8269
epoch 380, total train loss = 0.3000, train acc = 0.9954, test acc = 0.8759	epoch 380, total train loss = 3.7548, train acc = 0.9213, test acc = 0.8139
epoch 400, total train loss = 0.2193, train acc = 0.9991, test acc = 0.8648	epoch 400, total train loss = 3.2847, train acc = 0.9231, test acc = 0.8046
epoch 420, total train loss = 0.3492, train acc = 0.9944, test acc = 0.8750	epoch 420, total train loss = 3.2158, train acc = 0.9287, test acc = 0.8083
epoch 440, total train loss = 0.2859, train acc = 0.9954, test acc = 0.8731	epoch 440, total train loss = 3.3987, train acc = 0.9296, test acc = 0.8176
epoch 460, total train loss = 0.3236, train acc = 0.9935, test acc = 0.8741	epoch 460, total train loss = 3.0774, train acc = 0.9278, test acc = 0.8194
epoch 480, total train loss = 0.4364, train acc = 0.9907, test acc = 0.8667	epoch 480, total train loss = 3.2224, train acc = 0.9333, test acc = 0.8093
best epoch 383, test acc = 0.8870	best epoch 424, test acc = 0.8278

Table 2　EEGNet 和 DeepConvNet 最高 Testing accuracy 截圖

```
model = eeg, activation = relu, lr = 0.01, highest accuracy = 87.59%
model = eeg, activation = relu, lr = 0.005, highest accuracy = 85.83%
model = eeg, activation = relu, lr = 0.002, highest accuracy = 88.70%
model = eeg, activation = relu, lr = 0.001, highest accuracy = 86.67%
model = eeg, activation = relu, lr = 0.0005, highest accuracy = 86.67%
model = eeg, activation = relu, lr = 0.0002, highest accuracy = 85.00%
model = eeg, activation = relu, lr = 0.0001, highest accuracy = 83.61%
model = eeg, activation = leaky_relu, lr = 0.01, highest accuracy = 85.83%
model = eeg, activation = leaky_relu, lr = 0.005, highest accuracy = 86.57%
model = eeg, activation = leaky_relu, lr = 0.002, highest accuracy = 86.67%
model = eeg, activation = leaky_relu, lr = 0.001, highest accuracy = 87.69%
model = eeg, activation = leaky_relu, lr = 0.0005, highest accuracy = 85.93%
model = eeg, activation = leaky_relu, lr = 0.0002, highest accuracy = 85.74%
model = eeg, activation = leaky_relu, lr = 0.0001, highest accuracy = 83.43%
model = eeg, activation = elu, lr = 0.01, highest accuracy = 81.11%
model = eeg, activation = elu, lr = 0.005, highest accuracy = 84.07%
model = eeg, activation = elu, lr = 0.002, highest accuracy = 81.48%
model = eeg, activation = elu, lr = 0.001, highest accuracy = 82.50%
model = eeg, activation = elu, lr = 0.0005, highest accuracy = 83.89%
model = eeg, activation = elu, lr = 0.0002, highest accuracy = 82.50%
model = eeg, activation = elu, lr = 0.0001, highest accuracy = 80.28%
model = deepconv, activation = relu, lr = 0.01, highest accuracy = 82.22%
model = deepconv, activation = relu, lr = 0.005, highest accuracy = 81.67%
model = deepconv, activation = relu, lr = 0.002, highest accuracy = 82.50%
model = deepconv, activation = relu, lr = 0.001, highest accuracy = 81.48%
model = deepconv, activation = relu, lr = 0.0005, highest accuracy = 80.74%
model = deepconv, activation = relu, lr = 0.0002, highest accuracy = 81.20%
model = deepconv, activation = relu, lr = 0.0001, highest accuracy = 77.22%
model = deepconv, activation = leaky_relu, lr = 0.01, highest accuracy = 82.78%
model = deepconv, activation = leaky_relu, lr = 0.005, highest accuracy = 80.93%
model = deepconv, activation = leaky_relu, lr = 0.002, highest accuracy = 81.94%
model = deepconv, activation = leaky_relu, lr = 0.001, highest accuracy = 81.67%
model = deepconv, activation = leaky_relu, lr = 0.0005, highest accuracy = 81.11%
model = deepconv, activation = leaky_relu, lr = 0.0002, highest accuracy = 81.02%
model = deepconv, activation = leaky_relu, lr = 0.0001, highest accuracy = 79.63%
model = deepconv, activation = elu, lr = 0.01, highest accuracy = 82.22%
model = deepconv, activation = elu, lr = 0.005, highest accuracy = 80.83%
model = deepconv, activation = elu, lr = 0.002, highest accuracy = 80.83%
model = deepconv, activation = elu, lr = 0.001, highest accuracy = 81.39%
model = deepconv, activation = elu, lr = 0.0005, highest accuracy = 79.63%
model = deepconv, activation = elu, lr = 0.0002, highest accuracy = 80.83%
model = deepconv, activation = elu, lr = 0.0001, highest accuracy = 80.00%
model = vgg, activation = relu, lr = 0.01, highest accuracy = 80.37%
model = vgg, activation = relu, lr = 0.005, highest accuracy = 80.00%
model = vgg, activation = relu, lr = 0.002, highest accuracy = 81.11%
model = vgg, activation = relu, lr = 0.001, highest accuracy = 81.48%
model = vgg, activation = relu, lr = 0.0005, highest accuracy = 79.91%
model = vgg, activation = relu, lr = 0.0002, highest accuracy = 77.13%
model = vgg, activation = relu, lr = 0.0001, highest accuracy = 79.44%
model = vgg, activation = leaky_relu, lr = 0.01, highest accuracy = 79.91%
model = vgg, activation = leaky_relu, lr = 0.005, highest accuracy = 79.17%
model = vgg, activation = leaky_relu, lr = 0.002, highest accuracy = 80.56%
model = vgg, activation = leaky_relu, lr = 0.001, highest accuracy = 79.81%
model = vgg, activation = leaky_relu, lr = 0.0005, highest accuracy = 79.91%
model = vgg, activation = leaky_relu, lr = 0.0002, highest accuracy = 79.63%
model = vgg, activation = leaky_relu, lr = 0.0001, highest accuracy = 78.52%
model = vgg, activation = elu, lr = 0.01, highest accuracy = 82.31%
model = vgg, activation = elu, lr = 0.005, highest accuracy = 83.15%
model = vgg, activation = elu, lr = 0.002, highest accuracy = 81.39%
model = vgg, activation = elu, lr = 0.001, highest accuracy = 80.56%
model = vgg, activation = elu, lr = 0.0005, highest accuracy = 79.26%
model = vgg, activation = elu, lr = 0.0002, highest accuracy = 80.00%
model = vgg, activation = elu, lr = 0.0001, highest accuracy = 81.30%
```

Figure 6　完整數據截圖(三種模型，三種 activation function，七種 learning rate)

B. Comparison figures

- **EEGNet**

Figure 7 為 EEGNet 的 Accuracy learning curve，x 軸為訓練 epoch 數量，y 軸為 accuracy (%)，圖中共有六條曲線，分別表示三種不同 activation function 之下，EEGNet 的 training 和 testing accuracy。從圖中可以看出，當訓練 epoch 達到 50 時，testing accuracy 幾乎就不再增加、趨於穩定，然而此時 training accuracy 仍在不斷增長，代表模型可能有 overfitting training dataset 的問題。此外，使用 ELU 作為 activation function 似乎讓 EEGNet 的分類表現較差，其 training 和 testing 兩條學習曲線的差距比使用 ReLU 和 Leaky ReLU 來的大，得到的 testing accuracy 結果也最差。

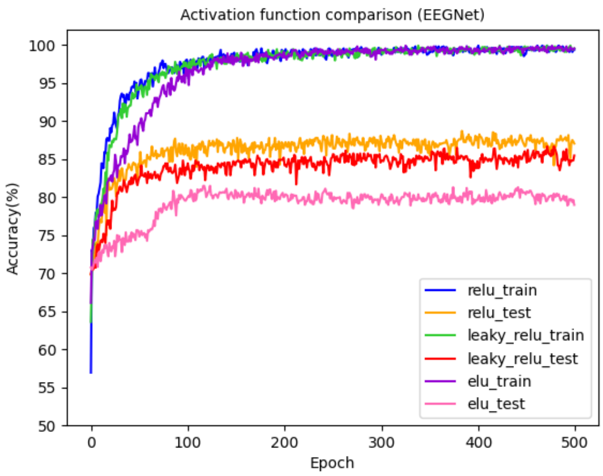

Figure 7　EEGNet 的 Accuracy learning curve

- **DeepConvNet**

Figure 8 為 DeepConvNet 的 Accuracy learning curve，x 軸為訓練 epoch 數量，y 軸為 accuracy (%)，圖中共有六條曲線，分別表示三種不同 activation function 之下，DeepConvNet 的 training 和 testing accuracy。從圖中可以看出，當訓練 epoch 達到 50 時，testing accuracy 幾乎就不再增加、趨於穩定，然而此時 training accuracy 仍在不斷增長，代表模型和 EEGNet 一樣可能有 overfitting training dataset 的問題。不同於 EEGNet 的是，DeepConvNet 使用三種不同的 activation function (ReLU, Leaky ReLU, ELU)的實驗結果沒有特別的差異，三者的 training 和 testing accuracy 學習曲線高度貼合，同組 activation function 設定下的 training 和 testing accuracy 差異幾乎一致。另外，DeepConvNet 的 accuracy 的收斂速度也比 EEGNet 略快一些。

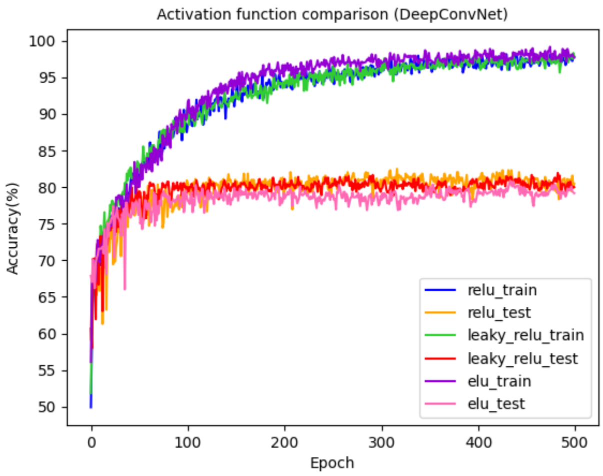

Figure 8　DeepConvNet 的 Accuracy learning curve

4. Discussion

第四部份探討兩種指定模型(EEGNet、DeepConvNet)與額外增加的一種分類模型(VGGNet)，三種模型的分類效果。以 Table 3 記錄三種模型在不同參數設定(Learning Rate, Activation Function)下的 Test Accuracy 實驗結果。額外新增的分類模型(VGGNet)則另外在第五部分 Extra 中，以 Table 4 記錄該模型的架構與實驗參數細節。

A. Anything you want to share

Table 6 為不同 activation function 和 learning rate 設定之下，三種分類模型(EEGNet, DeepConvNet, VGGNet)的 testing accuracy 結果。每個模型皆訓練 500 個 epochs，並回報最好的 testing accuracy 結果。三個模型的 training batch size 設定皆為 64。每個模型表現最好的結果以粗體和螢光底色標示，只有粗體則是標示>87%的 testing accuracy。

從表格中可以看出，EEGNet 在選擇 ReLU 或 Leaky ReLU 作為 activation function 時，表現幾乎一致，testing accuracy 沒有太大差異。然而，在選擇 ELU 當作 activation function 時，testing accuracy 則明顯下滑，表現最差。對於 DeepConvNet 和 VGGNet 兩個模型而言，使用不同 activation function 則對 testing accuracy 沒有太大的差別，差異不大。

另外，從數據中可以明顯發現分類準確度排名 EEGNet > DeepConvNet≒VGGNet，可以由此推測出 EEGNet 中的 depthwise 和 separable convolution modules 設計可以有效提取 EEG 訊號中的關鍵資訊，例如：frequency-specific spatial information 和 time summary。這使得其分類效果(testing accuracy)比一般的 Deep Convolution Network 好很多。而 VGGNet 模型的階層式 2D Convolution layer 架構，可能更適合用於一般有區分細部與整體特徵的圖片，對於 EEG 訊號的特徵提取沒有 EEGNet 好，因此分類準確度和 EEGNet 相較之下差很多。

Learning Rate	EEGNet			DeepConvNet			VGGNet		
	ReLU	Leaky ReLU	ELU	ReLU	Leaky ReLU	ELU	ReLU	Leaky ReLU	ELU
1e-2	**87.59%**	85.83%	81.11%	82.22%	**82.78%**	82.22%	80.65%	81.30%	80.74%
5e-3	85.83%	86.57%	84.07%	81.67%	80.93%	80.83%	79.26%	79.72%	**83.25%**
2e-3	**88.70%**	86.67%	81.48%	82.50%	81.94%	80.83%	79.91%	81.11%	80.56%
1e-3	86.67%	**87.69%**	82.50%	81.48%	81.67%	81.39%	80.93%	81.30%	80.74%
5e-4	86.67%	85.93%	83.89%	80.74%	81.11%	79.63%	80.46%	79.91%	80.28%
2e-4	85.00%	85.74%	82.50%	81.20%	81.02%	80.83%	78.06%	76.11%	81.02%
1e-4	83.61%	83.43%	80.28%	77.22%	79.63%	80.00%	78.70%	78.98%	81.39%

Table 3　不同實驗設定(activation function, learning rate)之下，三種分類模型的 testing accuracy 結果

5. Extra

A. Implement another classification model

在加分項目的部分，除了指定的 EEGNet 和 DeepConvNet 之外，本次 lab 也嘗試建構額外的分類模型。分類模型以階層式的 CNN layer 為主，以著名的 VGG16 架構做為搭建參考。本次 lab 使用的 VGGNet 模型架構與參數如 Table 4 所示，包含四個 convolution blocks，每個 block 裡面包含二或三層 2D CNN layer。考量到這次的分類任務複雜度沒有很高，太多的訓練參數可能導致模型出現 overfitting 問題，或是運算資源無法支援的情況。因此選擇降低原本 VGG16 中的 channel 數量，範圍由 VGG16 的(64→128→256→512)變成 VGGNet (16→32→64→128)，與原始參數不同。

Model	Module	Layer	Input Channel	Output Channel	Kernel Size
VGGNet	Conv2D Block 1	CNN layer 1	1	16	(3,3)
		CNN layer 2	16	16	(3,3)
	Conv2D Block 2	CNN layer 3	16	32	(3,3)
		CNN layer 4	32	32	(3,3)
	Conv2D Block 3	CNN layer 5	32	64	(3,3)
		CNN layer 6	64	64	(3,3)
		CNN layer 7	64	64	(3,3)
	Conv2D Block 4	CNN layer 8	64	128	(3,3)
		CNN layer 9	128	128	(3,3)
		CNN layer 10	128	128	(3,3)

Table 4　本次 lab 設計的 VGGNet 實作參數細節

VGGNet 的分類 accuracy 結果顯示在第四部份的 Table 3 中，由於 VGGNet 模型的階層式 2D CNN layer 架構會逐步抽取細節特徵，相較於這次任務 dataset 的 EEG 訊號而言，可能更適合應用於一般圖片的分類任務上。從數據中也可以看出，VGGNet 在這次任務的表現比 EEGNet 差許多。另外，由於 VGGNet 的參數比其他模型更多，因此需要的運算時間也較長。

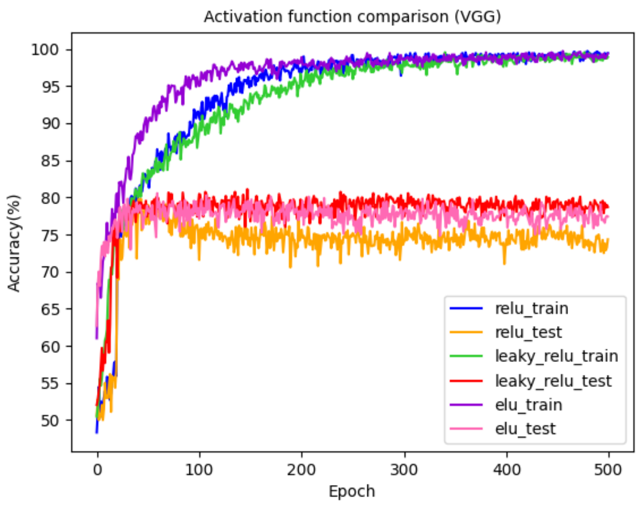

Figure 9　VGGNet 的 Accuracy learning curve

Figure 9 為 VGGNet 的 accuracy learning curve，從圖中可以看出，當訓練 epoch 達到 50 時，testing accuracy 幾乎就不再增加、趨於穩定，然而此時 training accuracy 仍在不斷增長，代表模型一樣有 overfitting training dataset 的問題。不同於 EEGNet 和 DeepConvNet 的是，VGGNet 在使用 ReLU 作為 activation function 時的分類表現較差，其 training 和 testing 兩條學習曲線的差距比使用 Leaky ReLU 和 ELU 來的大，得到的 testing accuracy 結果也最差。