Introduction

此次lab利用pytorch實作EEGNet以及DeepConvNet兩種convolutional neural network，兩者均為照著架構圖所刻劃出來，再利用這兩種net去對腦電圖進行分類，並從中替換activation function，共有ELU，ReLU，Leaky ReLU三種，比較三種不同activation function對兩CNN的影響，進而判斷在哪種架構下效果最好。
先利用dataloader.py將腦電圖轉換成list，並將其丟到所需的CNN中，算出其loss，並做back propagation，更新weight，這邊的batch size為64，故每64筆data做一次weight更新，當每筆data都丟完後，為一epoch，共運算150個epoch。
通過CNN後會得到一個二維向量，當第一個數的值大於第二個數的值時，判斷為第一類，小於等於時判斷為第0類。

Experiment setups

A. The detail of model

1. EEGNet

layer	filters	size	Activation	options
Input		(1, C, T)		
Conv2d	16	(1, 51)	Linear	
BatchNorm2d				eps=1e-05, momentum=0.1
Conv2d	32	(C, 1)	Linear	
BatchNorm2d				eps=1e-05, momentum=0.1
Activation			ELU	
AvgPool2d		(1, 4)		
Dropout				p = 0.25
Conv2d	32	(1, 15)	Linear	
BatchNorm2d				eps=1e-05, momentum=0.1
Activation			ELU	
AvgPool2d		(1, 8)		
Dropout				p = 0.25
Flatten				
Dense	2		softmax	

主要是依照助教給的範例code去刻劃，和ppt上圖片基本一致，但因為這裡的loss function是用cross entropy，所以最後一層output不需要通過softmax function，因為已經包含在cross entropy的class裡了。

2. DeepConvNet

layer	filters	size	Activation	options
input		(1, C, T)		
Conv2d	25	(1, 5)	Linear	
Conv2d	25	(C, 1)	Linear	
BatchNorm2d				eps=1e-05, momentum=0.1
Activation			ELU	
MaxPool2d		(1, 2)		
Dropout				p = 0.5
Conv2d	50	(1, 5)	Linear	
BatchNorm2d				eps=1e-05, momentum=0.1
Activation			ELU	
MaxPool2d		(1, 2)		
Dropout				p = 0.5
Conv2d	100	(1, 5)	Linear	
BatchNorm2d				eps=1e-05, momentum=0.1
Activation			ELU	
MaxPool2d		(1, 2)		
Dropout				p = 0.5
Conv2d	200	(1, 5)	Linear	
BatchNorm2d				eps=1e-05, momentum=0.1
Activation			ELU	
MaxPool2d		(1, 2)		
Dropout				p = 0.5
Flatten				
Dense	2		softmax	

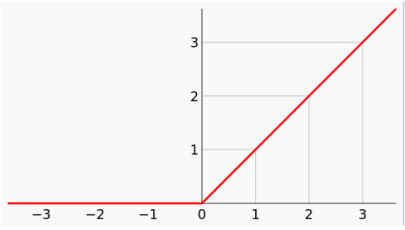
和ppt上所附架構完全相同，照著架構一層一層慢慢通過，這裡最後一層同樣不需要經過softmax function。

B. Explain the activation function

1. ReLU

將小於0的數全部轉為0。

f(x) = max(0, x)



2. Leaky ReLU

ReLU是將所有負數設為0，leaky ReLU則是賦予負數一個非0斜率，可以用來防止遇到負數時，斜率變0，參數無法更新的問題

LeakyReLU(x) = x, if x >= 0
Negative_slope * x, if x < 0

3.ELU

ELU試圖將ReLU的平均值接近0，以加快學習的數度。

ELU(x) = max(0, x) + min(0, α * (exp(x)-1))

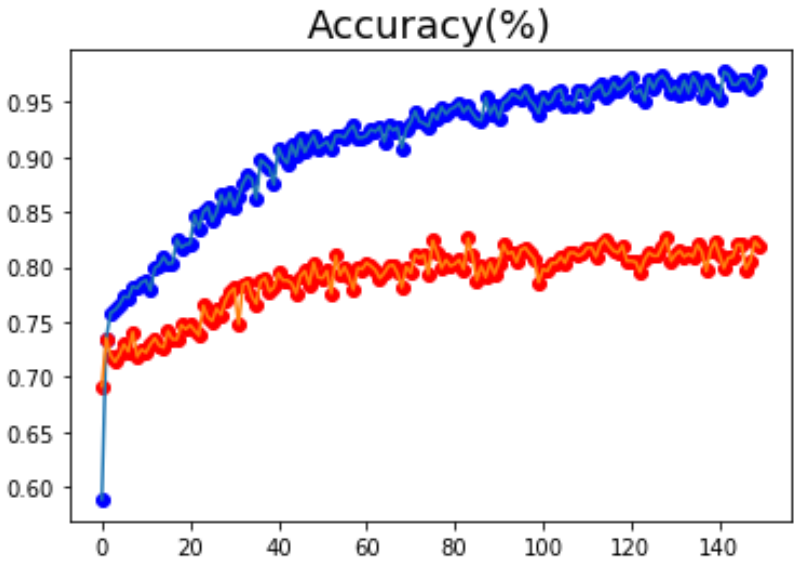
α 的default value為1.0

Experimental results

A. The highest testing accuracy and anything want to present

ELU function:

EEGNet:



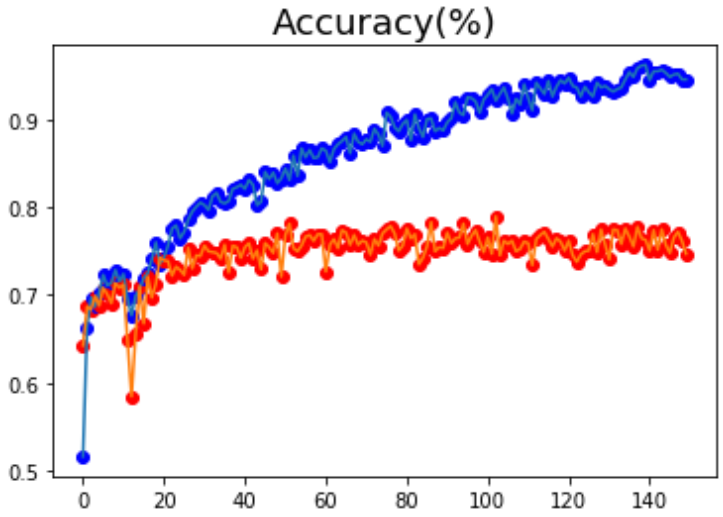
紅線為testing accuracy，藍線為training accuracy。

x軸為epochs，y軸為accuracy。

```
epochs: 148
Train Accuracy:  0.9666666666666667
Test Accuracy:  0.8231481481481482
epochs: 149
Train Accuracy:  0.9777777777777777
Test Accuracy:  0.8185185185185185
0.825925925925926
```

Max accuracy = 0.8259

DeepConvNet:



紅線為testing accuracy，藍線為training accuracy。

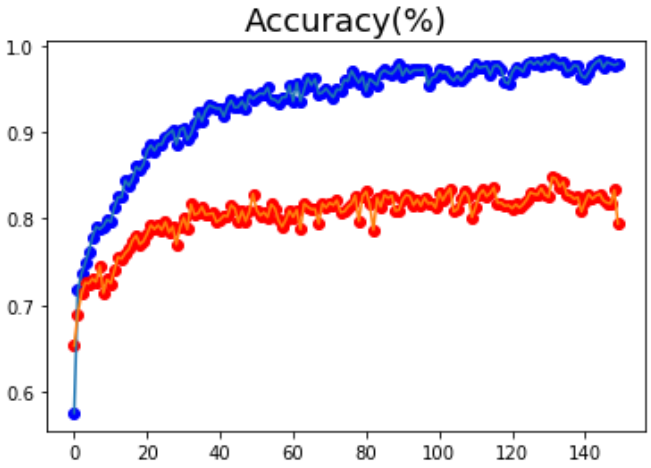
x軸為epochs，y軸為accuracy。

```
epochs: 148
Train Accuracy:  0.9462962962962963
Test Accuracy:  0.7620370370370371
0.7898148148148149
epochs: 149
Train Accuracy:  0.9462962962962963
Test Accuracy:  0.7472222222222222
0.7898148148148149
```

Max accuracy: 0.7898

LeakyReLU function:

EEGNet:



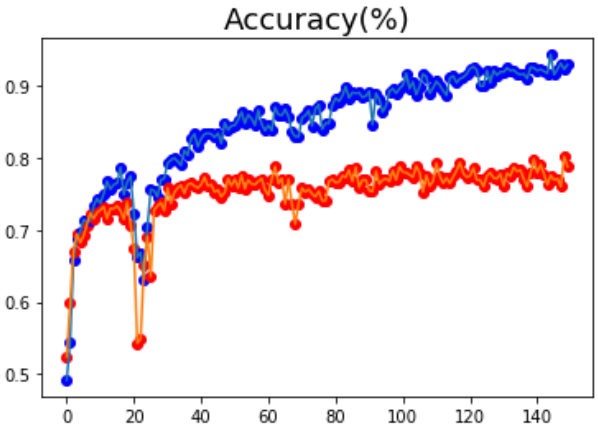
紅線為testing accuracy，藍線為training accuracy。

x軸為epochs，y軸為accuracy。

```
Train Accuracy:  0.9777777777777777
Test Accuracy:  0.8333333333333334
epochs: 149
Train Accuracy:  0.9796296296296296
Test Accuracy:  0.7944444444444444
Max accuracy:  0.8481481481481481
```

Max accuracy: 0.8481

DeepConvNet:



紅線為testing accuracy，藍線為training accuracy。

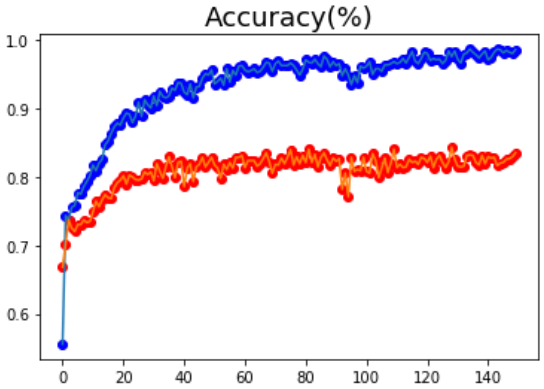
x軸為epochs，y軸為accuracy。

```
epochs: 148
Train Accuracy:  0.9231481481481482
Test Accuracy:  0.8018518518518518
epochs: 149
Train Accuracy:  0.9296296296296296
Test Accuracy:  0.787962962962963
Max accuracy:  0.8018518518518518
```

Max accuracy: 0.8018

ReLU function:

EEGNet:



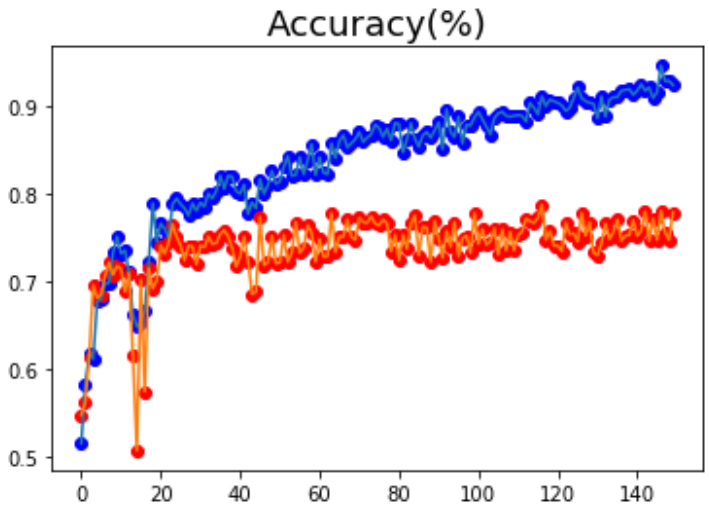
紅線為testing accuracy，藍線為training accuracy。

x軸為epochs，y軸為accuracy。

```
epochs: 148
Train Accuracy:  0.9805555555555555
Test Accuracy:  0.8296296296296296
epochs: 149
Train Accuracy:  0.9861111111111112
Test Accuracy:  0.8351851851851851
Max accuracy:  0.8435185185185186
```

Max accuracy: 0.8435

DeepConvNet:



紅線為testing accuracy，藍線為training accuracy。

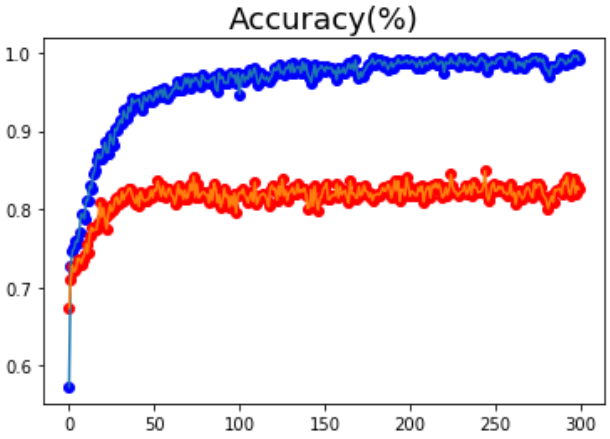
x軸為epochs，y軸為accuracy。

```
epochs: 148
Train Accuracy:  0.9287037037037037
Test Accuracy:  0.7472222222222222
epochs: 149
Train Accuracy:  0.924074074074074
Test Accuracy:  0.7787037037037037
Max accuracy:  0.7870370370370371
```

Max accuracy: 0.7870

當將epoch調至300，可以發現到100後test accuracy幾乎沒再繼續上升。(此以LeakyReLU作為activation function)

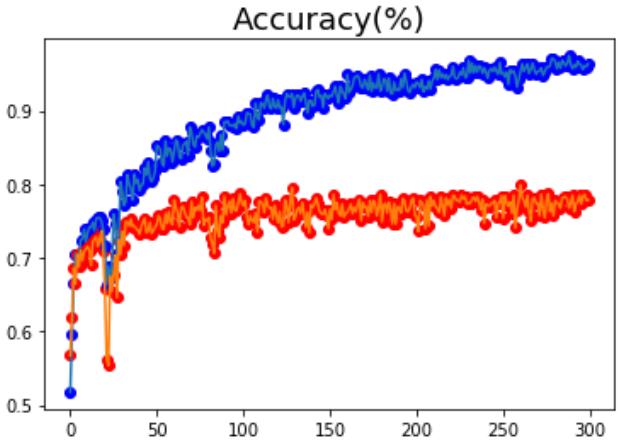
EEGNet:




```
epochs: 298
Train Accuracy:  0.9962962962962963
Test Accuracy:  0.8296296296296296
epochs: 299
Train Accuracy:  0.9907407407407407
Test Accuracy:  0.825925925926
Max accuracy:  0.8490740740740741
```

Max arruracy: 0.8490

DeepConvNet:




```
epochs: 298
Train Accuracy:  0.9601851851851851
Test Accuracy:  0.7777777777777778
epochs: 299
Train Accuracy:  0.9638888888888889
Test Accuracy:  0.7787037037037037
Max accuracy:  0.799074074074074
```

Max accuracy: 0.7990

B. Comparison figures

	ReLU	LeakyReLU	ELU
EEGNet	0.8435	0.8481	0.8259
DeepConvNet	0.7870	0.8018	0.7898

Discussion

A. Anything you want to share

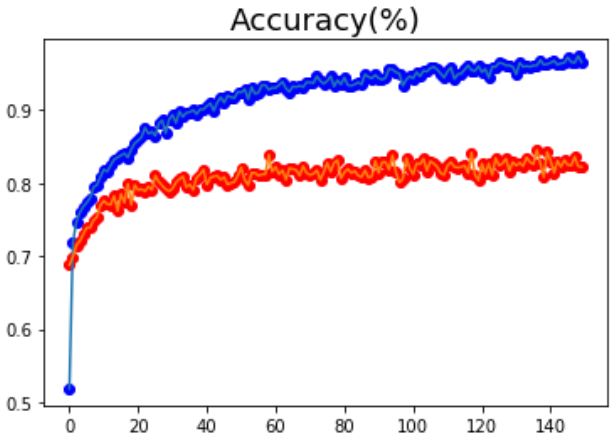
在建立EEGNet和DeepConvNet時，必須注意各層input和output的數量，雖然這次對於CNN的架構都已規範好，但在寫code時難免會不小心打錯，在debug時花了不少的時間。

另外由結果可以觀察到，training到後來，以training data來看，accuracy可以提升到近99%，但testing data的accuracy卻只在80%上下，沒法再往上提升，這邊推測雖然已經利用pooling的手法來降低overfitting的機率，但依然可能有發生overfitting的現象，或是圖片的某些特殊特徵去影響到了整個neural network的計算。

更換loss function(以EEGNet為例，epoch = 150)
(negative log likelihood loss)
需在CNN最後一層加上softmax function

```
y = torch.softmax(self.Linear_4(y), 1)
```

```
Loss = nn.KLDivLoss()
```




```
epochs: 148
Train Accuracy:  0.9740740740740741
Test Accuracy:  0.8231481481481482
epochs: 149
Train Accuracy:  0.9657407407407408
Test Accuracy:  0.8222222222222222
Max accuracy:  0.8453703703703703
```