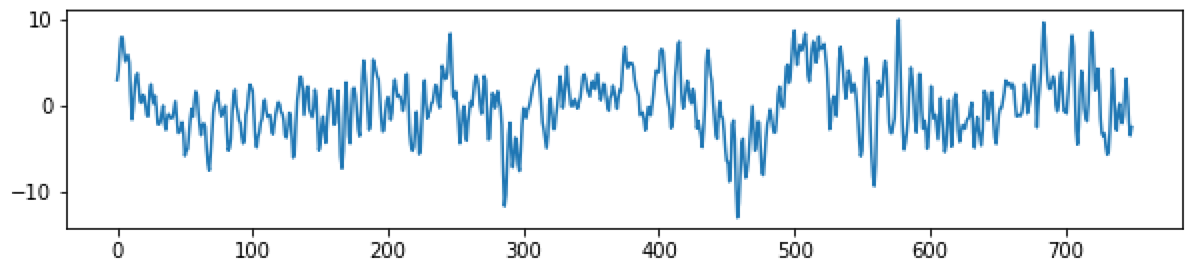
Deep Learning and Practice

Lab 3 309505018 郭俊廷

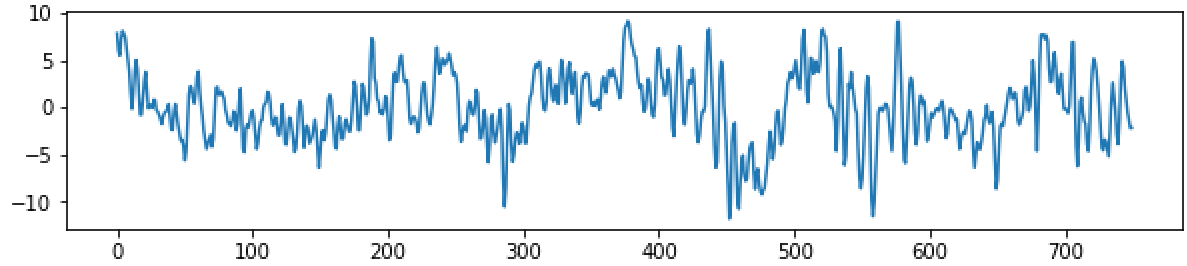
1. Introduction (20%)

實作 EEGNet和 DeepConvNet, 解決2-class classification的問題, (x11b和s4b) 並以BCI competition dataset作為本次的datasets, 舉其中一筆dataset為例:

x11b channel 1:



x11b channel 2:



dataset shape為 (1080, 1, 2, 750) 1080筆data,C=1,H=2,W=750

2. Experiment Setup (30%)

1) **The detail of your model(EEGNet, DeepConvNet)**

EEGNet 架構為:

first Convolution → depthwise Convolution → separable Convolution
EEGNet 所採用的depthwise-separable Convolution設計,比傳統的Convolution 降低更多的參數數量, 提升train和evaluate的速度,但又不太影響accuracy

DeepConvNet 架構為:

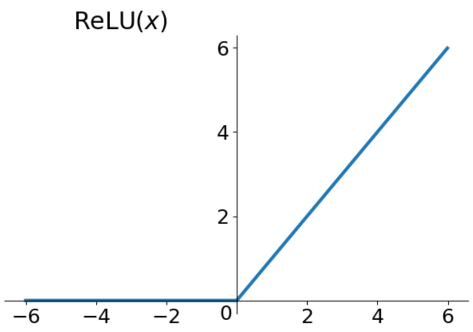
Conv2D
→ (Conv2D + BatchNorm + Activation + MaxPooling + Dropout)
→ (Conv2D + BatchNorm + Activation + MaxPooling + Dropout)
→ (Conv2D + BatchNorm + Activation + MaxPooling + Dropout)
→ (Conv2D + BatchNorm + Activation + MaxPooling + Dropout)
→ Fully connected Layer
跟傳統的Convolution的架構差不多

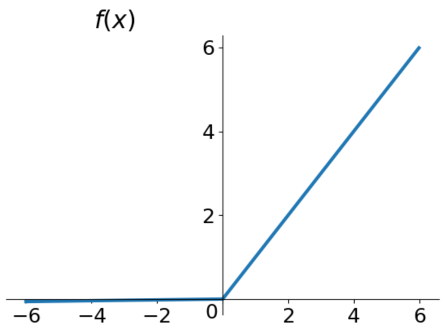
```python
class EEGNet(nn.Module):
    def __init__(self,activation=nn.ELU()):
        super(EEGNet,self).__init__()
        self.firstconv=nn.Sequential(
            nn.Conv2d(1,16,kernel_size=(1,51),stride=(1,1),padding=(0,25),bias=False),
            nn.BatchNorm2d(16,eps=1e-5,momentum=0.1,affine=True,track_running_stats=True)
        )
        self.depthwiseConv=nn.Sequential(
            nn.Conv2d(16,32,kernel_size=(2,1),stride=(1,1),groups=16,bias=False),
            nn.BatchNorm2d(32,eps=1e-5,momentum=0.1,affine=True,track_running_stats=True),
            activation,
            nn.AvgPool2d(kernel_size=(1,4),stride=(1,4),padding=0),
            nn.Dropout(p=0.25)
        )
        self.seperableConv=nn.Sequential(
            nn.Conv2d(32,32,kernel_size=(1,15),stride=(1,1),padding=(0,7),bias=False),
            nn.BatchNorm2d(32,eps=1e-5,momentum=0.1,affine=True,track_running_stats=True),
            activation,
            nn.AvgPool2d(kernel_size=(1,8),stride=(1,8),padding=0),
            nn.Dropout(p=0.25)
        )
        self.classify=nn.Linear(736,2)

    def forward(self,X):
        out=self.firstconv(X)
        out=self.depthwiseConv(out)
        out=self.seperableConv(out)
        out=out.view(out.shape[0],-1)
        out=self.classify(out)
        return out
```

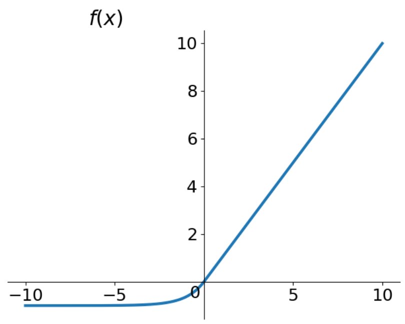
Layer	# filters	size	# params	Activation	Options
Input		(C, T)			
Reshape		(1, C, T)			
Conv2D	25	(1, 5)	150	Linear	mode = valid, max norm = 2
Conv2D	25	(C, 1)	25 * 25 * C + 25	Linear	mode = valid, max norm = 2
BatchNorm			2 * 25		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
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

2) Explain the activation function (ReLU, Leaky ReLU, ELU)

ReLU: $f(x) = max(0, x)$ Leaky ReLU: $f(x) = max(0.01x, x)$ ELU: $f(x) = \alpha(e^{x} - 1),\ if\ x < 0$







ReLU, Leaky ReLU, ELU這三個差別在於x小於0的時候不一樣,
ReLU 會有Dead ReLU Problem, 在training時某些神經元可能永遠不會被activate,導致相應的參數永遠不能被更新, 所以在做backpropagation時, Leaky ReLU和ELU在訓練上會有比較好的結果, 因為當x小於0時還是有值

3. Experiment Results (30%)

1) The highest testing accuracy

EEGNet

```
ReLU_train max acc: 97.96296296296296
ReLU_test max acc: 87.68518518518519
LeakyReLU_train max acc: 97.96296296296296
LeakyReLU_test max acc: 86.66666666666667
ELU_train max acc: 91.48148148148148
ELU_test max acc: 82.87037037037037
```

DeepConvNet

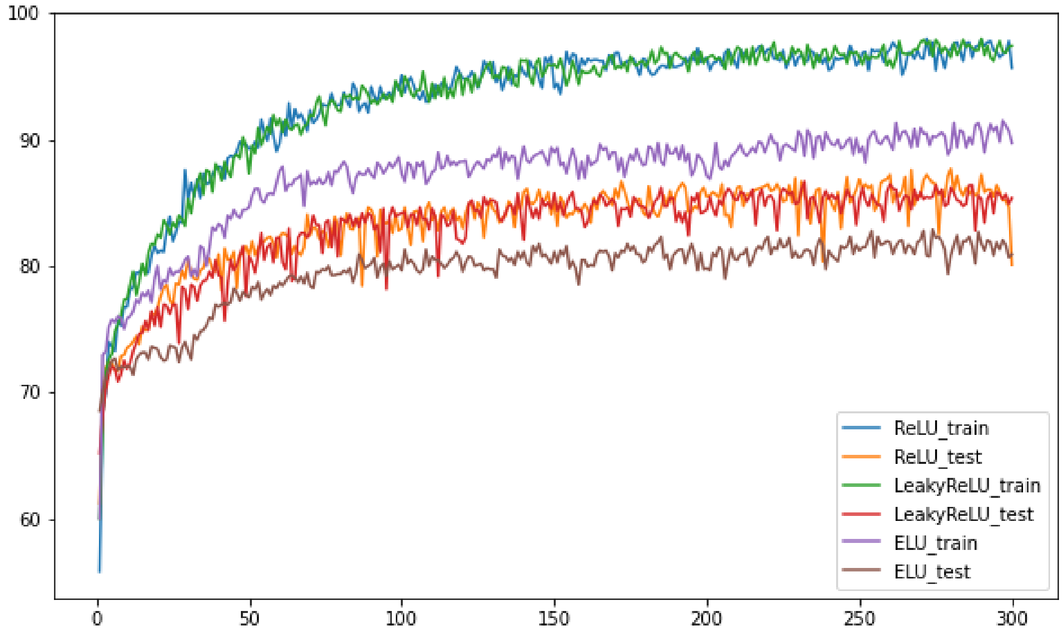
```
ReLU_train max acc: 94.35185185185185
ReLU_test max acc: 84.44444444444444
LeakyReLU_train max acc: 94.9074074074074
LeakyReLU_test max acc: 84.35185185185185
ELU_train max acc: 98.24074074074075
ELU_test max acc: 82.31481481481481
```

Testing data accuracy:

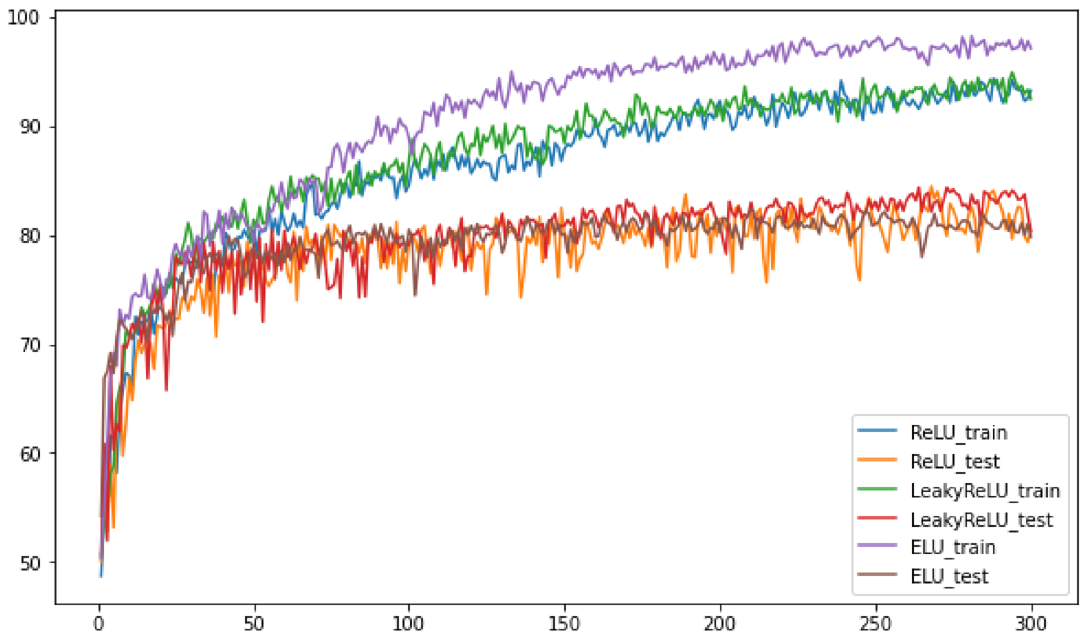
	ReLU	Leaky ReLU	ELU
EEGNet	87.69%	86.67%	82.87%
DeepConvNet	84.44%	84.35%	82.31%

2) Comparison figure

EEGNet



DeepConvNet



4. Discussion (20%)

1. 一開始以為data是一張張的圖片, 後來發現他的shape是(1080, 1, 2, 750), 所以我想很久為什麼一張圖片會是四維的

2.一開始用read_bci_data讀完data後, 不知道要怎麼讓data input到 pytorch training,後來才知道要用torch.utils.data import DataLoader()跟 Tensordataset(),所以流程是用read_bci_data讀完data後, 放到Tensordataset(), 再放到DataLoader(), 算是pytorch的一個語法

3.x的type是torch.float, y的type是torch.long