DLP Lab2 Report

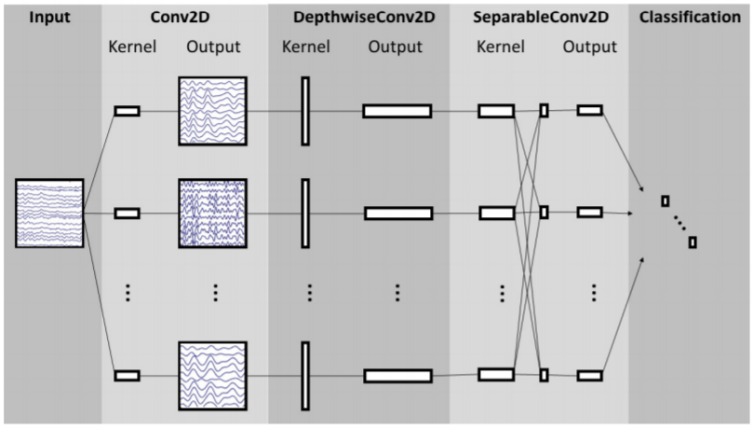
309551124 涂景智

1. Introduction

此次作業用 python 搭配 pytorch 實作 EEGNet 和 DeepConvNet。

以下分別簡單介紹要實作的 EEGNet 和 DeepConvNet：

- EEGNet



EEGNet 是常用於腦電圖識別任務的 Convolution Neural Network，EEGNet 是由三層 Convolution 組成，分別是普通的 Convolution、Depthwise Convolution、和 Separable Convolution，

- DeepConvNet

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

DeepConvNet 的實作與助教提供的圖幾乎完全相同（有試著調整過 Dropout 比例，但發現 Dropout 調高會使模型崩潰）。差別只在最後一層 （FC 層）沒有 Softmax，因為使用的 Loss Function 為 Cross Entropy。

2. Experiment set up

A. The detail of your model

- EEGNet

```
class EEGNet(nn.Module):
    def __init__(self):
        super(EEGNet, self).__init__()

        self.firstconv = nn.Sequential(
            nn.Conv2d(1, 16, (1, 51), stride=(1, 1), padding=(0, 25), bias=False),
            nn.BatchNorm2d(16, affine=True, track_running_stats=True)
        )

        self.depthwiseConv = nn.Sequential(
            nn.Conv2d(16, 32, (2, 1), stride=(1, 1), groups=16, bias=False),
            nn.BatchNorm2d(32, affine=True, track_running_stats=True),
            decide_act(),
            nn.AvgPool2d((1, 4), stride=(1, 4), padding=0),
            nn.Dropout(p=0.25)
        )

        self.separableConv = nn.Sequential(
            nn.Conv2d(32, 32, (1, 15), stride=(1, 1), padding=(0, 7), bias=False),
            nn.BatchNorm2d(32, affine=True, track_running_stats=True),
            decide_act(),
            nn.AvgPool2d((1, 8), stride=(1, 8), padding=0),
            nn.Dropout(p=0.25)
        )
```

```
        self.classify = nn.Sequential(
            nn.Linear(in_features=736, out_features=2, bias=True)
        )

    def forward(self, x):
        conv1_out = self.firstconv(x)
        conv2_out = self.depthwiseConv(conv1_out)
        conv3_out = self.separableConv(conv2_out)
        conv3_out = conv3_out.view(-1, 736)
        out   = self.classify(conv3_out)
        return out
```

上述程式就是 EEGNet 三層的實作，原則上都與助教提供的參數相同，要特別注意的只有 「decide_act()」 function，該 function 的實作如下：

```
def decide_act():
    if _active == "ELU":
        return nn.ELU()
    elif _active == "ReLU":
        return nn.ReLU()
    else:
        return nn.LeakyReLU()
```

因為之後需要測試三種 activate function （ELU、ReLU、Leaky ReLU）在 EGGNet 和 DeepConvNet 的表現，此 function 的目的即為根據不同的情況會在 model 中使用不同的 activate function。

- DeepConvNet

```
class DeepConvNet(nn.Module):
    def __init__(self):
        super(DeepConvNet, self).__init__()

        self.TotalConv = nn.Sequential(
            nn.Conv2d(1, 25, (1, 5), stride=(1, 1), bias=True),
            nn.Conv2d(25, 25, (2, 1), stride=(1, 1), bias=True),
            nn.BatchNorm2d(25, affine=True, track_running_stats=True),
            decide_act(),
            nn.MaxPool2d((1, 2), stride=(1, 2)),
            nn.Dropout(p=0.5),
            nn.Conv2d(25, 50, (1, 5), stride=(1, 1), bias=True),
            nn.BatchNorm2d(50, affine=True, track_running_stats=True),
            decide_act(),
            nn.MaxPool2d((1, 2), stride=(1, 2)),
            nn.Dropout(p=0.5),
            nn.Conv2d(50, 100, (1, 5), stride=(1, 1), bias=True),
            nn.BatchNorm2d(100, affine=True, track_running_stats=True),
            decide_act(),
            nn.MaxPool2d((1, 2), stride=(1, 2)),
            nn.Dropout(p=0.5),
            nn.Conv2d(100, 200, (1, 5), stride=(1, 1), bias=True),
            nn.BatchNorm2d(200, affine=True, track_running_stats=True),
            decide_act(),
            nn.MaxPool2d((1, 2), stride=(1, 2)),
            nn.Dropout(p=0.5),
            nn.Flatten(),
        )

        self.dense = nn.Sequential(
            torch.nn.Linear(8600, 2),
        )

    def forward(self, x):
        x = self.TotalConv(x)
        _out = self.dense(x)
        return _out
```

DeepConvNet 的實作的參數設置，也與助教提供的相同

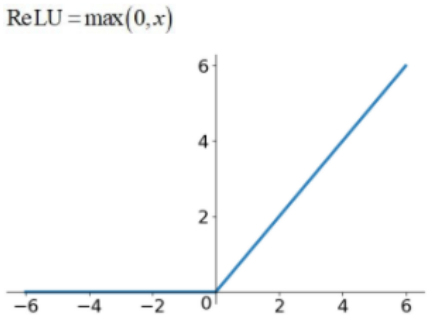
在實際的訓練中，為了使不同的 activate function 和 ML Model 可以公平的比較，每個 model 我們都是訓練 500 Epochs，Batch size 為 ・60・Optimizer 則是使用 Adam，learning rate 參數設為 0.005，weight decade 參數設為 0.0025・

B. Explain the activation function

■ ReLU：

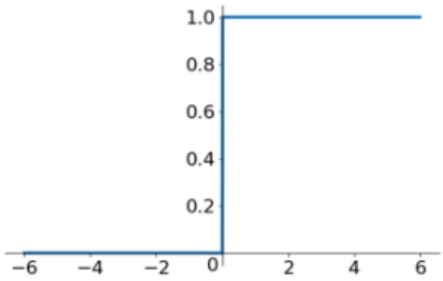
ReLU 是近年來最頻繁被使用的激勵函數，因其可適度解決梯度爆炸問題、且其具備計算和收斂的速度都相當快等特性・

ReLU 圖形



另外，ReLU 函數並不是全區間皆可微分，但是不可微分的部分可以使用 Sub-gradient 進行取代・

ReLU 導數圖形

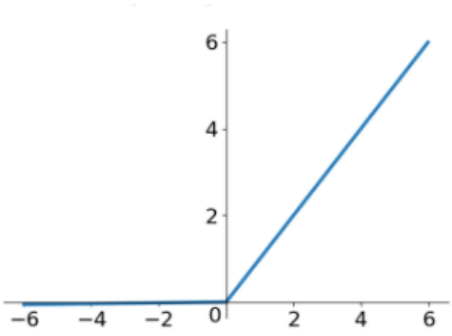


■ LeakyReLU

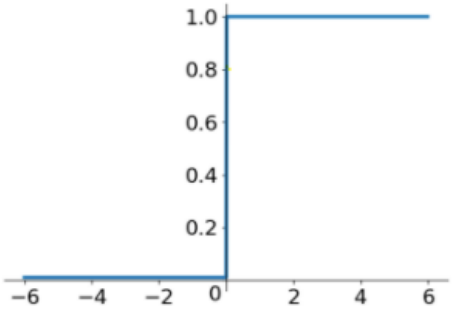
LeakyReLU = max(0.01x, x)

為了解決 Dead ReLU Problem (意即當某些神經元輸出為 0 時，就難以再輸出)，Leaky ReLU 將 ReLU 的前半段輸出設為 0.01x，如此可防止當值為負號時，永遠無法被激活之問題．

LeakyReLU 圖形



LeakyReLU 導數圖形

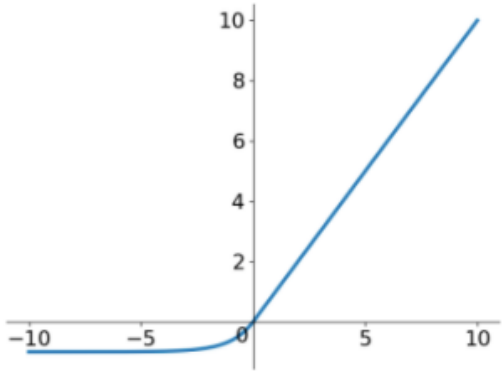


■ ELU

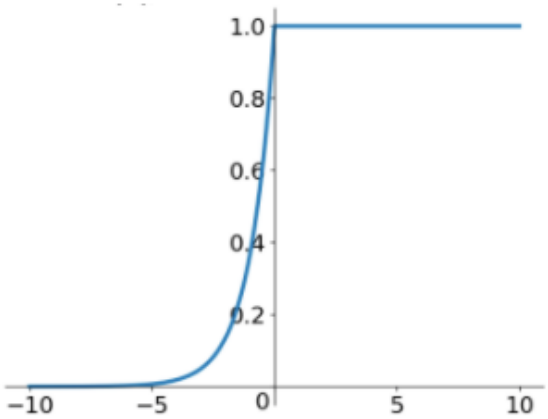
ELU 為 Exponential Linear Units 的簡稱，其目的也是在於解決 Dead ReLU 問題．

ELU 圖形

$$f(x)=\begin{cases} x, & if\ x>0 \\ \alpha\left(e^{x}-1\right), & otherwise \end{cases}$$



ELU 導數圖形



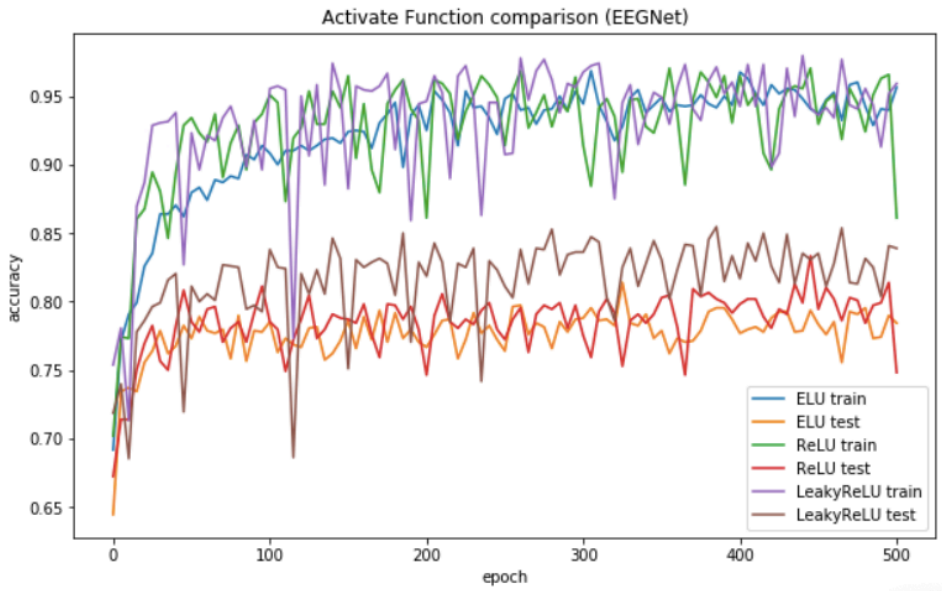
3. Experiment results

A. The highest testing accuracy

	ReLU	Leaky ReLU	ELU
EEGNet	**83.33%**	**85.46%**	**81.38%**
DeepConvNet	82.03%	83.31%	75.27%

B. Comparison figures

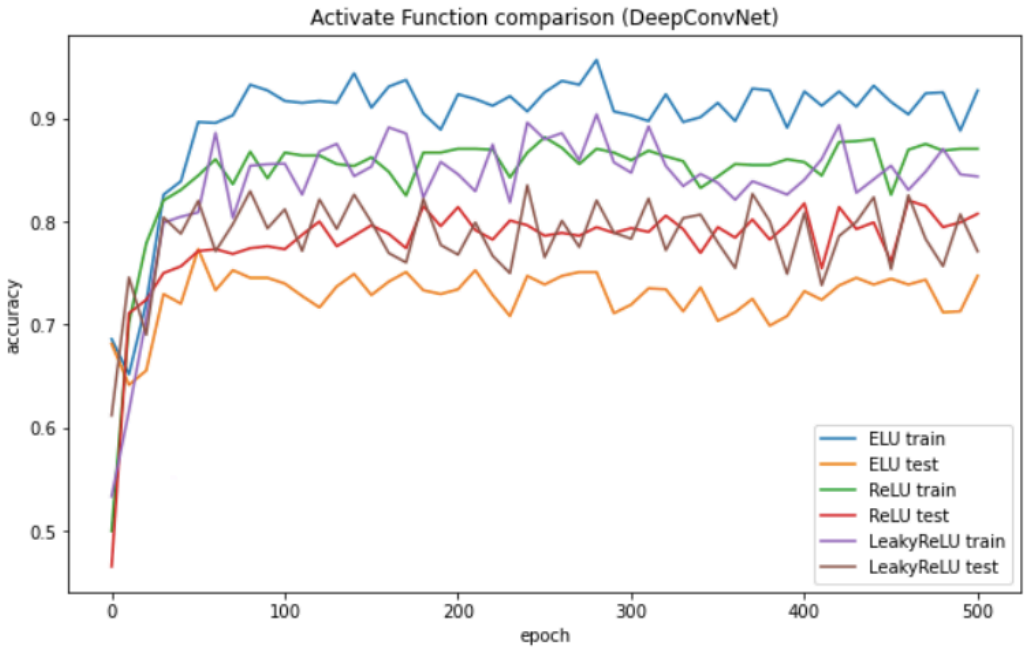
- EEGNet



每 5 epoch 為一點

根據圖表，就 training data 而言，三種 activate function 的準確率都差不多 （在訓練 500 epochs 後，準確率皆約為 95% 左右）. 但就 testing data 而言，LeakyReLU 的結果優於 ReLU

- DeepConvNet



每 10 epoch 為一點

與 EEGNet 相比，DeepConvNet 的結果都較差。值得注意的是若 Activate function 使用 ELU，則 Train data accuracy 最高，但 Test data accuracy 最低。推測可能是 Overfitting 而導致。

4. Discussion

在前面 Model 的訓練中，之所以在 Adam 會在設置 weight decade，是因為若不設置 weight decade，則 Model 可能會有 overfitting 的情形發生

做以下實驗，考慮三組不同的 weight decade：

(1). weight decade = 0
(2). weight decade = 0.001
(3). weight decade = 0.002

每組用 EEGNet 搭配不同 activate function 訓練 150 epochs，且 Learning rate 設為 0.005 （與前面訓練結果相同），訓練結果如下表：

Activate Function	Weight decade	150th Epoch Train Data Accuracy	Highest Test Data Accuracy During Training
ELU	0	98.51%	82.87%
	0.001	96.85%	81.85%
	0.002	93.42%	81.66
ReLU	0	97.12%	81.38%
	0.001	97.12%	83.05%
	0.002	95.55%	82.28%
Leaky ReLU	0	99.16%	82.59%
	0.001	94.72	81.48%
	0.002	91.48%	83.88%

觀察實驗結果，可以發現一如我們所想，當完全沒有 Weight decade 時（Weight decade = 0），無論是何種 Activate Function，都會產生 Overfitting 的問題（Train Data Accuracy 皆超過 97%）。而實驗結果也發現，提高 Weight decade 也的確會使 Overfitting 的情況減少（三種 Activate Function 在調高 Weight decade 後 Train Data Accuracy 皆有下降的趨勢）。

最後，就 Test Data Accuracy 而言，我們也發現稍微調高 Weight decade 並不會使 Model 的準確率受到影響 （不同的 Weight decade 在三種 Activate Function 中的 Test Data Accuracy 都相近）.