DL LAB2 Report - EEG classfication

匯出pdf之後排版變得有點醜 😥😥
助教不介意的話可以到HackMD網址閱讀 謝謝～～～
https://hackmd.io/ygl3id4VS3qBE1VSMkVL5g (https://hackmd.io/ygl3id4VS3qBE1VSMkVL5g)

Introduction

In this lab, we need to implement simple EEG classification models.

Requirments

1. Implement the EEGNet, DeepConvNet with three kinds of activation function including ReLU, Leaky ReLU, ELU.
2. In the experiment results, show the highest accuracy of two architectures with three kinds of activation functions.
3. To visualize the accuracy trend, plot each epoch accuracy during training phase and testing phase.

Dataset

- BCI Competition III - IIIb
- [2 classes, 2 bipolar EEG channels]

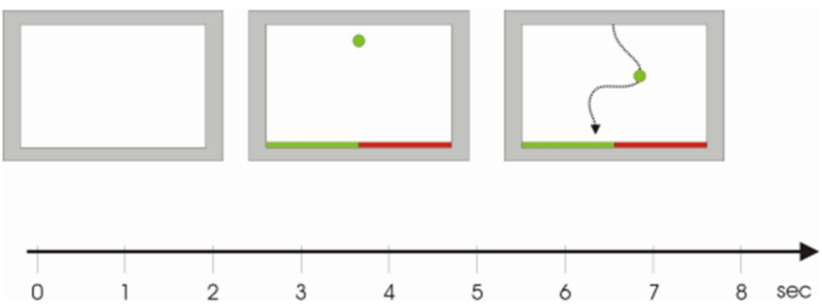
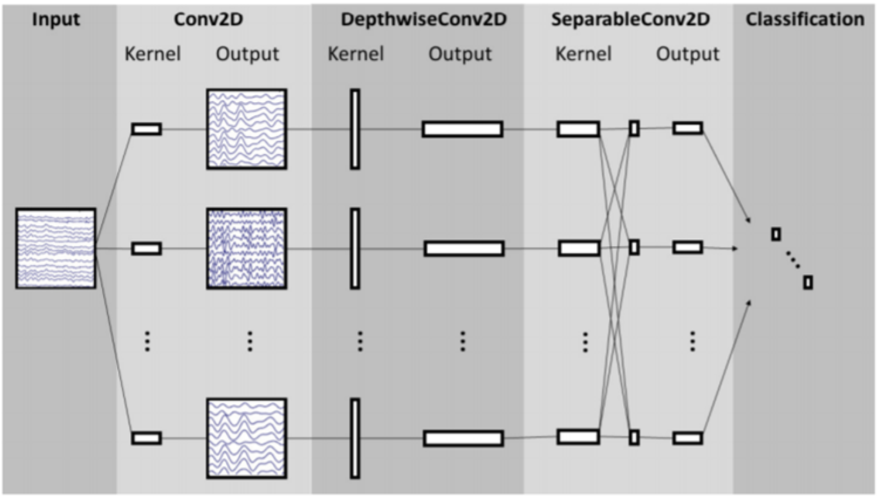


Figure 3: Basket paradigm used for S4 and X11 [3].

Experiment set up

The detail of your model

- EEGNet

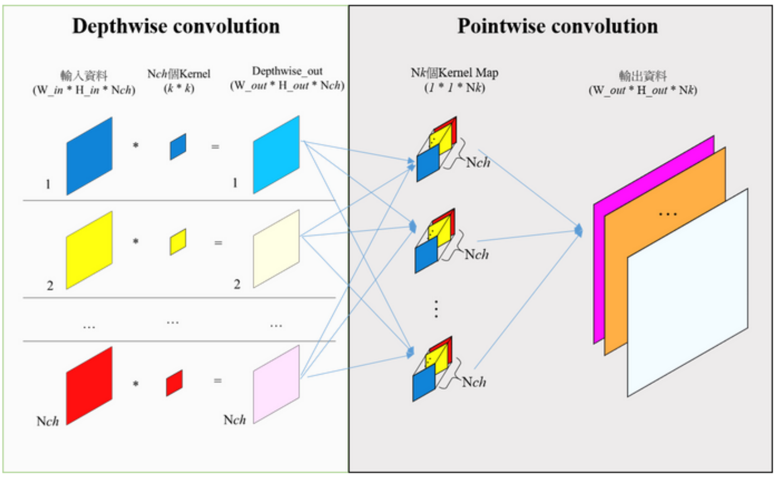



```
EEGNet(
  (firstconv): Sequential(
    (0): Conv2d(1, 16, kernel_size=(1, 51), stride=(1, 1), padding=(0, 25), bias=False)
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (depthwiseConv): Sequential(
    (0): Conv2d(16, 32, kernel_size=(2, 1), stride=(1, 1), groups=16, bias=False)
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): AvgPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0)
    (4): Dropout(p=0.25)
  )
  (separableConv): Sequential(
    (0): Conv2d(32, 32, kernel_size=(1, 15), stride=(1, 1), padding=(0, 7), bias=False)
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ELU(alpha=1.0)
    (3): AvgPool2d(kernel_size=(1, 8), stride=(1, 8), padding=0)
    (4): Dropout(p=0.25)
  )
  (classify): Sequential(
    (0): Linear(in_features=736, out_features=2, bias=True)
  )
)
```

較特別的部分是使用了Depthwise separable convolution，以下簡單介紹

- Depthwise separable convolution
  是Google在2017年提出的MobileNet模型，主要是因為隨著deep learning的快速發展，model不但深又巨大，使得整體參數很可觀。在巨大的網路模型中，大部分的計算量都發生在convolution計算中，因此Depthwise separable convolution就是為了在不減低太多performance狀態下，盡量去減少原始convolution的計算量，基本上可以拆成兩部分Depthwise convolution和pointwise convolution。

1. Depthwise convolution
   針對輸入資料的每一個channel都建立一個k * k的kernel，然後每一個channel針對對應的kernel都各自分開做convolution。
   這步驟和一般convolution不太一樣，一般的convolution計算是每個kernel map都要和所有channel都去做convolution，這邊是分開獨立去做。
   輸入資料 : W_in * H_in * Nch
   做每個輸入channel相對應的kernel map(k * k)的convolution
   輸出大小為W_out * H_out * N_ch
2. Pointwise convolution
   在輸入資料的每個channel做完depthwise convolution後，針對每個點的所有channel做pointwise convolution。
   實際做法是建立Nk個1 * 1 * Nch的kernel map，將depthwise convolution的輸出做一般1*1的convolution計算



$$\frac{\text{Depthwise separable convolution 計算量}}{\text{一般卷積計算量}}$$

$$= \frac{W_{in} * H_{in} * Nch * k * k + Nch * Nk * W\_in * H\_in}{W\_in * H\_in * Nch * k * k * Nk}$$

$$= \frac{1}{Nk} + \frac{1}{k * k}$$

所以當kernel map越大和數量越多，Depthwise separable convolution可以節省越多計算量。

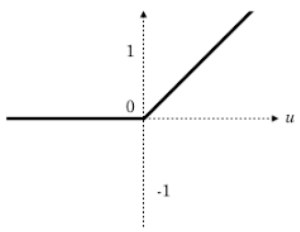
- DeepConvNet
  C = 2, T = 750, N = 2

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

使用最傳統的CNN CBAPD架構
C(Conv2D) -> B(BatchNormaliztion) -> A(Activation) -> P(MaxPool2D) -> D(Dropout)

Explain the activation function

- ReLU

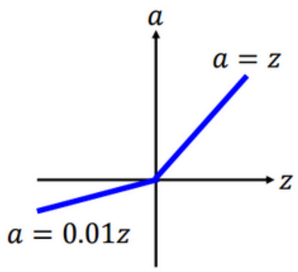


$$\mathrm{ReLU}(x) = \max(0, x)$$

優點
1. ReLU的SGD收斂速度比sigmoid和tanh快
2. 在x>0的區域上，不會出現梯度飽和、梯度消失的問題
3. 計算複雜度低，不需要進行指數運算，只要一個 threshold

缺點
1. ReLU的輸出不是zero mean
2. Dead ReLU Problem：ReLU在負數區域被kill的現象叫做dead ReLU。在x<0時，梯度為0，這個neuron及之後的neurons梯度永遠為0，不再對任何數據有所響應，導致相應參數永遠不會被更新

- Leaky ReLU

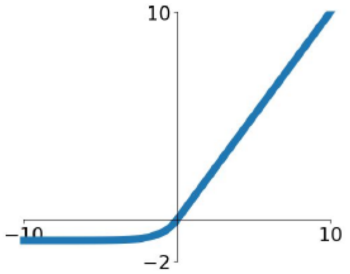


$$\text{LeakyRELU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \text{negative\_slope} \times x, & \text{otherwise} \end{cases}$$

By default, the negative slope = 0.01

為了解決dead ReLU現象，用一個小值來初始化neuron，使得ReLU在負數區域更偏向於激活而不是死掉。
LeakyReLU是ReLU的演變版本，主要就是為了解決ReLU輸出為0的問題。如圖所示，在輸入小於0時，雖然輸出值很小但是值不為0。
它有一個缺點就是它有點近似線性，導致在複雜分類中效果不好。

- ELU



$$\text{ELU}(x) = \max(0, x) + \min(0, \alpha * (\exp(x) - 1))$$

具有ReLU的優勢，且沒有Dead ReLU問題，輸出的mean接近0。
有負數飽和區域，所以對noise有一些robustness，可以看作是介於ReLU和Leaky ReLU之間的一個function。
但它需要計算指數，所以計算量上更大一點。

Experiment results

The highest testing accuarcy

- Screenshot with two models

EEG

```
ReLU_train max acc: 98.2%
ReLU_test max acc: 87.5%
LeakyReLU_train max acc: 97.8%
LeakyReLU_test max acc: 88.0%
ELU_train max acc: 92.1%
ELU_test max acc: 83.7%
```

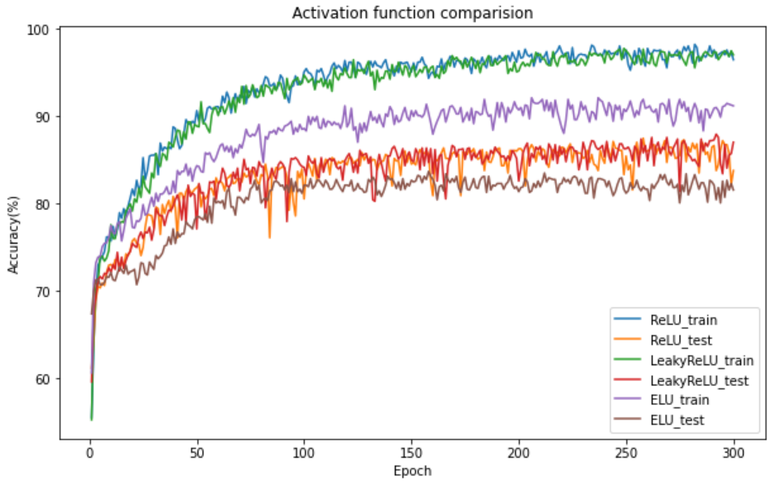
DeepConv

```
ReLU_train max acc: 95.0%
ReLU_test max acc: 83.6%
LeakyReLU_train max acc: 94.7%
LeakyReLU_test max acc: 85.6%
ELU_train max acc: 98.4%
ELU_test max acc: 83.1%
```

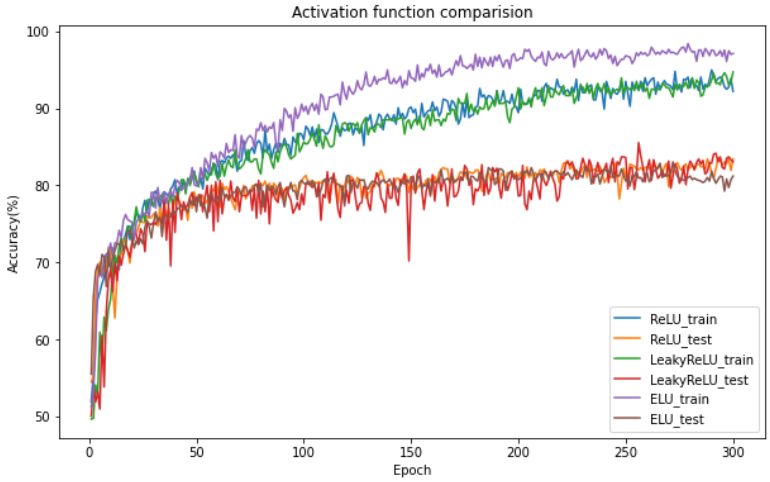
	ReLU	**Leaky ReLU**	**ELU**
EEGNet	**87.5%**	**88.0%**	**83.7%**
DeepConvNet	83.6%	85.6%	83.1%

Comparison figures

- EEGNet



- DeepConvNet



Discussion

有嘗試過以下各種組合，highest accuracy大約都在84~87%之間

```
Batch size = 64, 128, 256
Learning rate = 0.01, 0.001
Epochs = 150, 300
random seed = 1, 2, 3
```

最後選定以下組合，得到highest accuary = 88%的結果

```
Batch size = 256
Learning rate = 0.001
Epochs = 300
random seed = 3
```

還有發現，大部分組合的三種activation function都是在EEGNet的accuracy比較高。

> 另外，在計算nn.Linear的參數時，有在網路上查到
> 如果數字沒辦法整除，CNN取ceiling，pooling取floor