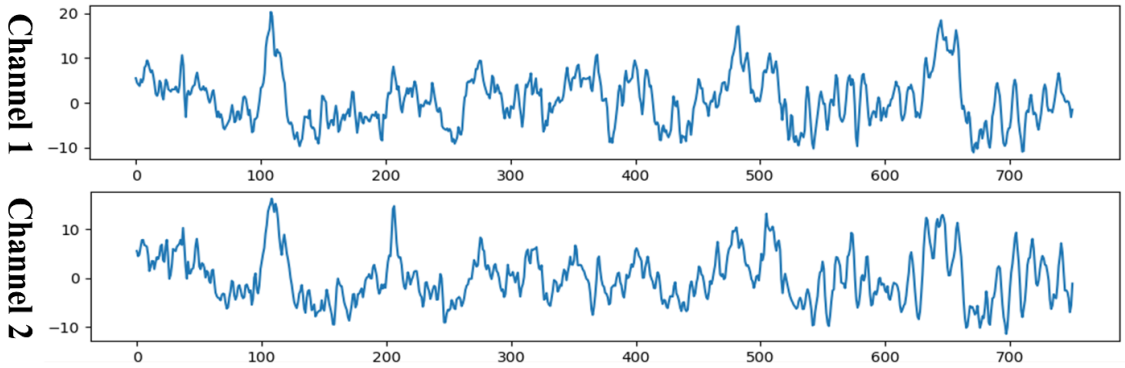
1. Introduction



在此 Lab 當中我們需要透過上述的腦波圖來判斷此人所看到的物體是向左移動還是向右移動，當然如果我們是要自己搜集這類型的資料整個會非常複雜，於是我們此 Lab 的目標就是希望透過 BCI 公開的資料集去進行訓練。
而透過助教給我們的 Data loader 其實只要將資料讀進來，然後硬 train 一發其實就結束了。

2. Experiment setup
   a. The detail of your model
      - Hyperparameters setup:

        Epochs: 500
        Learning rate: 0.001
        Weight decay: 0.01
        Batch size: 256
        Optimizer: Adam
        Loss function: Cross Entropy

      - 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)
  )
)
```

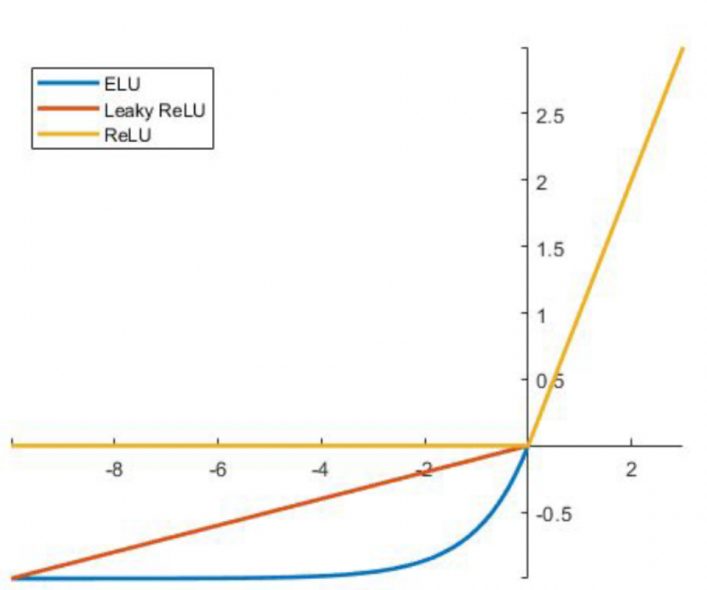
照著 spec 把每一層刻出來，其實就是輕量版本的 CNN，降低參數量減少運算時間的同時不會損失太多的正確率。

- 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

傳統的 CNN 架構，參數量會比 EEG 還要多常數倍，但正確率卻沒有因此攀升特別顯著，因為目前對於神經網路的數學證明還沒有非常完整，我們只能透過不斷的嘗試去調整神經網路要用多少層以及每一層分別要用多少參數量，這也導致一開始的 CNN 比起後來出現的 EEG 有如此多的參數量。

b. Explain the activation function



可以看到 ELU 以及 Leaky ReLU 其實就是平滑版本的 ReLU，因為當我們使用 ReLU 函數去進行 Backpropagation 時，ReLU 會將一切小於 0 的數值直接變成 0，這種情況權重以及參數就不會跟新，這導致我們沒有辦法很好的使用 Gradient descent 來更新參數。

但同時 ReLU 也有一個優點就是函數本身的運算相對簡單且沒有梯度消失的問題，雖然 Leaky ReLU 以及 ELU 也不會有梯度消失的問題而且實際收斂速度會比 Sigmoid/tanh 還要快，但計算量相比最簡單的 ReLU 還是大了一點，尤其 ELU 是有指數運算的。

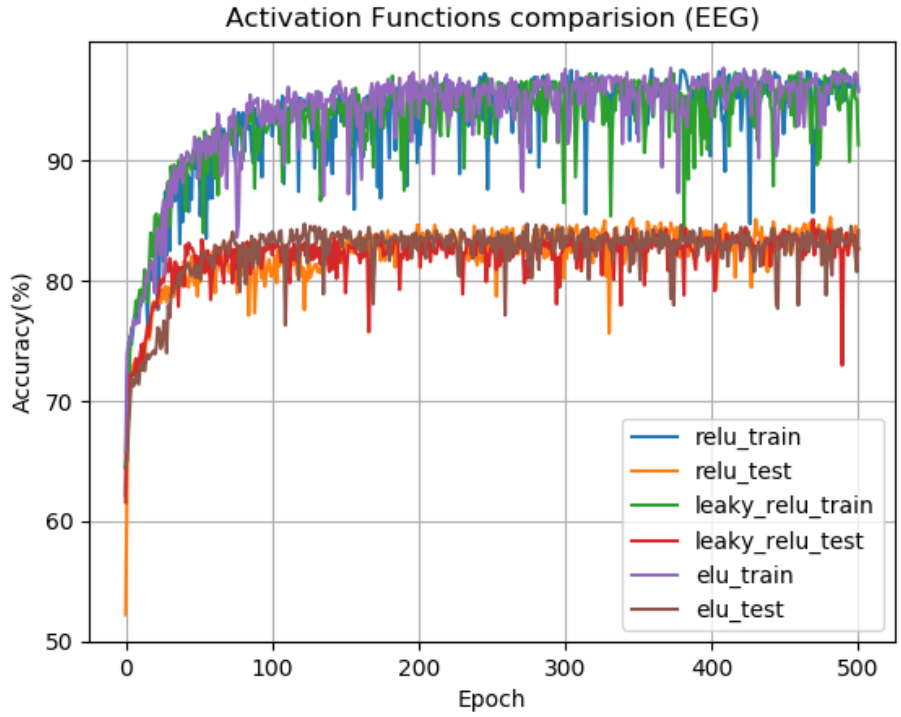
3. Experiment result
   a. The highest testing accuracy

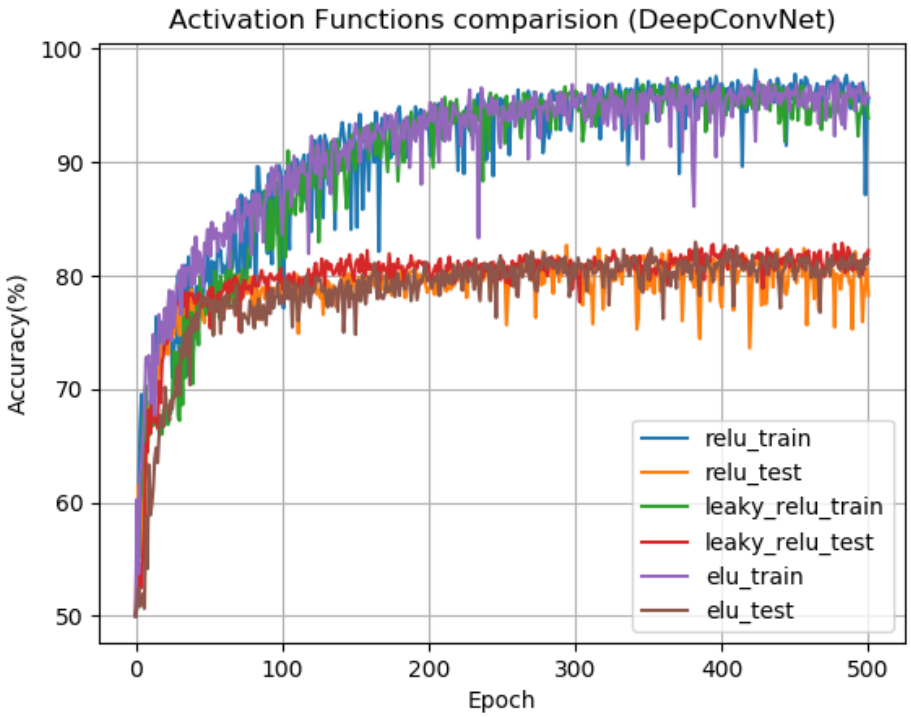
Model/Actication	ReLU	LeakyReLU	ELU
EEGNet	83.88%	83.98%	85.27%
DeepConvNet	80.74%	81.29%	81.48%

```
EEG_ReLU_train
epoch=  0  loss=  0.0006032217983846311  correct=  0.6842592592592592
epoch=  100  loss=  0.00020277792656863178  correct=  0.9305555555555556
epoch=  200  loss=  0.0001494478710271694  correct=  0.9601851851851851
epoch=  300  loss=  0.00020386669094915743  correct=  0.9092592592592592
epoch=  400  loss=  0.00015386664481074723  correct=  0.9490740740740741
epoch=  500  loss=  0.00013962288697560627  correct=  0.9490740740740741
EEG_ReLU_test
epoch=  500  loss=  0.0004325881048485085  correct=  0.8388888888888889
EEG_LeakyReLU_train
epoch=  0  loss=  0.0006119191094681068  correct=  0.625
epoch=  100  loss=  0.00017352163515709065  correct=  0.9425925925925925
epoch=  200  loss=  0.00015296391038982957  correct=  0.9546296296296296
epoch=  300  loss=  0.00016308505502012042  correct=  0.9388888888888889
epoch=  400  loss=  0.00013074636183403157  correct=  0.9574074074074074
epoch=  500  loss=  0.0001283908607783141  correct=  0.9518518518518518
EEG_LeakyReLU_test
epoch=  500  loss=  0.0004097437968960515  correct=  0.8398148148148148
EEG_ELU_train
epoch=  0  loss=  0.0006204882705653156  correct=  0.6833333333333333
epoch=  100  loss=  0.00019857822744934648  correct=  0.9203703703703704
epoch=  200  loss=  0.00014682192769315507  correct=  0.9592592592592593
epoch=  300  loss=  0.00020262975659635332  correct=  0.9037037037037037
epoch=  400  loss=  0.00012908783499841338  correct=  0.9694444444444444
epoch=  500  loss=  0.00013860995294871155  correct=  0.9648148148148148
EEG_ELU_test
epoch=  500  loss=  0.00039897149911633244  correct=  0.8527777777777777


DeepConvNet_ReLU_train
epoch=  0  loss=  0.0006369362274805705  correct=  0.5157407407407407
epoch=  100  loss=  0.00027245534238991916  correct=  0.9018518518518519
epoch=  200  loss=  0.0001835939922818431  correct=  0.9462962962962963
epoch=  300  loss=  0.0001806588912451709  correct=  0.95
epoch=  400  loss=  0.00016120650150157787  correct=  0.9527777777777777
epoch=  500  loss=  0.0001690847592221366  correct=  0.9416666666666667
DeepConvNet_ReLU_test
epoch=  500  loss=  0.0004461049757621906  correct=  0.8074074074074075
DeepConvNet_LeakyReLU_train
epoch=  0  loss=  0.0006736385601538199  correct=  0.5
epoch=  100  loss=  0.00027121879436351636  correct=  0.9
epoch=  200  loss=  0.00020404549108611213  correct=  0.9222222222222223
epoch=  300  loss=  0.0001429709157458058  correct=  0.9555555555555556
epoch=  400  loss=  0.00012362774599481512  correct=  0.9666666666666667
epoch=  500  loss=  0.00014920781056086221  correct=  0.95
DeepConvNet_LeakyReLU_test
epoch=  500  loss=  0.0004435153747046435  correct=  0.812962962962963
DeepConvNet_ELU_train
epoch=  0  loss=  0.0007021352096840187  correct=  0.5
epoch=  100  loss=  0.00029252039061652229  correct=  0.8638888888888889
epoch=  200  loss=  0.00018827295689671127  correct=  0.9444444444444444
epoch=  300  loss=  0.00017317981907614955  correct=  0.9407407407407408
epoch=  400  loss=  0.00015725885276441222  correct=  0.950925925925926
epoch=  500  loss=  0.00011968910694122314  correct=  0.9731481481481481
DeepConvNet_ELU_test
epoch=  500  loss=  0.0003968102512536225 5  correct=  0.8148148148148148
```

b. Comparison figure





4. Discussion
   a. Anything you want to share
      - 增加了 Regularization 之後正確率從 81%上升到 85%。

```
EEG_ReLU
training epoch=  0  loss=  0.0006024818177576419  correct=  0.6129629629629629
testing epoch=  0  loss=  0.0006134245682645727  correct=  0.5824074074074074
training epoch=  100  loss=  0.0001945201583482601  correct=  0.937037037037037
testing epoch=  100  loss=  0.00043831950536480654  correct=  0.8342592592592593
training epoch=  200  loss=  0.0001671256704462899  correct=  0.9435185185185185
testing epoch=  200  loss=  0.0004389563644373858  correct=  0.8268518518518518
training epoch=  300  loss=  0.00014247625238365596  correct=  0.950925925925926
testing epoch=  300  loss=  0.0004200244115458594  correct=  0.85
training epoch=  400  loss=  0.00017008973216568982  correct=  0.9314814814814815
testing epoch=  400  loss=  0.00043739105264345803  correct=  0.8194444444444444
training epoch=  500  loss=  0.00012417871642995764  correct=  0.9648148148148148
testing epoch=  500  loss=  0.0004137987063990699  correct=  0.8351851851851851
```

- 上圖是我在某一次實驗當中所跑出來的結果，可以 EEG_ReLU 看到在第 300 個 epoch 時 testing 的結果已經超過 85%，但隨著 epoch 的增加，testing 的正確率反而降低了，因為 training 的正確率也是下降的，所以這邊看起來也不像是 overfitting，蠻有趣的一個現象，給機器一直看重複的資料正確率居然是下降的。