Deep learning Lab2

學號:311605015 姓名:張哲源

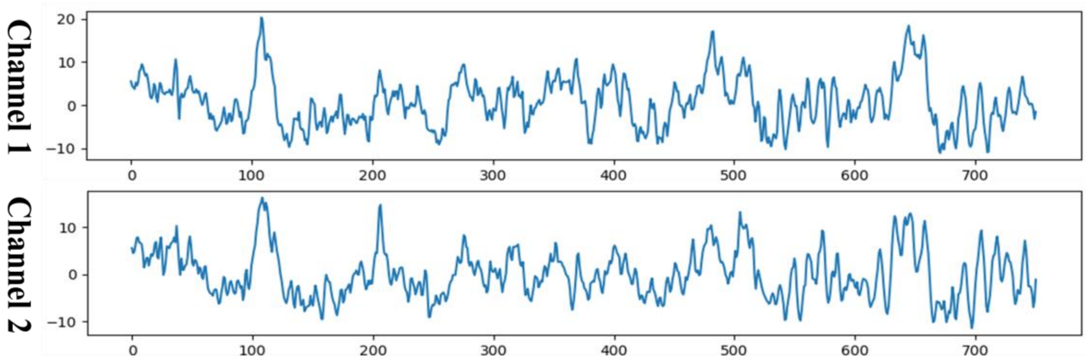
1.Introduction

1.1 Lab Objective

- 建構 EEGNET 以及 DeepConvNet 來實現 EEG 訊號分類模型
- 利用不同的 Activation function (Relu,Leaky Relu,ELY)等並觀察結果
- 將結果可視化並秀出最高準確率之結果(>87%)

1.2 Dataset

- 此資料集有兩個通道，每個通道各有 750 個 data point ，兩個 label 輸出分別代表球掉落到左手邊與右手邊
- 資料集已經被整理成[S4b_train.npz, X11b_train.npz] and [S4b_test.npz, X11b_test.npz]，並透過 dataaloader.py 存取近來



bipolar EEG signal

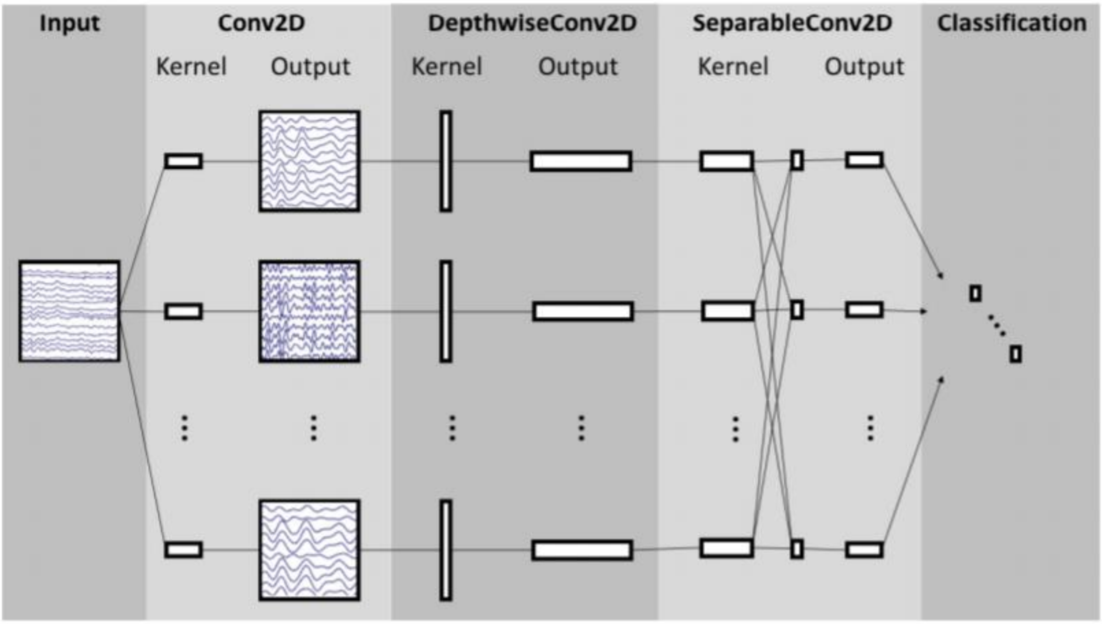
2.Experimental Setup

2.1 Convert Data to Tensor

```
train_data, train_label, test_data, test_label=read_bci_data()
test_dataset = TensorDataset (torch.Tensor(test_data) , torch.Tensor(test_label))
test_loader=DataLoader(test_dataset,batch_size=1080)
trian_dataset=TensorDataset(torch.Tensor(train_data),torch.Tensor(train_label))#資
loader =DataLoader(
    dataset=trian_dataset,
    batch_size=BATCH_SIZE,
    shuffle=True,
)
```

將 train_loader 每次取樣進行 shuffle 能得到更好的準確率

2.2 EEGNET



最初的 Conv2D 用來提取訊號的特徵，DepthwiseConv2D 將 multi channel 整理成一個 channel ，最後 SeperableConv2D 從 DepthwiseConv2D 裡面提取特徵

```
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)
  )
)
```

EEG Implement detail

實作細節如下，依據所給之提示，利用 Sequential 將所有網路連結再一起

```python
class EEGNet(nn.Module):
    def __init__(self, name,activation: nn.modules.activation):
        super().__init__()
        self.name=name
        self.firstconv=nn.Sequential(
            nn.Conv2d(1, 16, kernel_size=(1,51),stride=(1,1),padding=(0,25),bias =True),
            nn.BatchNorm2d(16,eps=1e-05,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=True),
            nn.BatchNorm2d(32,eps=1e-05,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.separableconv=nn.Sequential(
            nn.Conv2d(32,32,kernel_size=(1,15),stride=(1,1),padding=(0,7), bias=True),
            nn.BatchNorm2d(32,eps=1e-05,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.Sequential(
            nn.Flatten(),
            nn.Linear(in_features=736,out_features=2,bias=True),
        )
    def forward(self,input):
        out=self.firstconv(input)
        out=self.depthwiseConv(out)
        out=self.separableconv(out)
        out=self.classify(out)
        return out
```

2.3DeepConvnet

我們要實作 DeepConvnet 透過下表所給之參數,，而 C = 2, T = 750 and N = 2，max norm 可以進行忽略

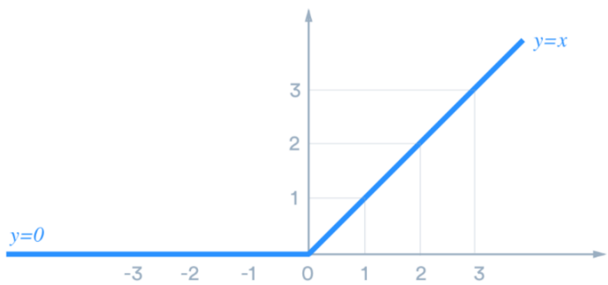
Layer	# filters	size	# params	Activation	Options
Input		(C, T)			
Reshape		(1, C, T)			
Conv2D	25	(1, 5)	150	Linear	mode = valid, max norm = 2
Conv2D	25	(C, 1)	25 * 25 * C + 25	Linear	mode = valid, max norm = 2
BatchNorm			2 * 25		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)	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

實作細節如下

```python
class DeepConvNet(nn.Module):
    def __init__(self,activation):
        super().__init__()
        self.firstconv=nn.Sequential(
            nn.Conv2d(1,25 ,kernel_size=(1,5),stride=(1,1)),
            nn.Conv2d(25,25,kernel_size=(2,1),stride=(1,1)),
            nn.BatchNorm2d(25,eps=1e-05,momentum=0.1,affine=True,track_running_stats=True),
            activation(),
            nn.MaxPool2d(kernel_size=(1,2)),
            nn.Dropout(p=0.5),
            nn.Conv2d(25,50,kernel_size=(1,5),stride=(1,1),bias=True),
            nn.BatchNorm2d(50,eps=1e-05,momentum=0.1),
            activation(),
            nn.MaxPool2d(kernel_size=(1,2)),
            nn.Dropout(p=0.5),
            nn.Conv2d(50,100,kernel_size=(1,5),stride=(1,1)),
            nn.BatchNorm2d(100,eps=1e-05,momentum=0.1),
            activation(),
            nn.MaxPool2d(kernel_size=(1,2)),
            nn.Dropout(0.5),
            nn.Conv2d(100,200,kernel_size=(1,5),stride=(1,1)),
            nn.BatchNorm2d(200,eps=1e-05,momentum=0.1),
            activation(),
            nn.MaxPool2d((1,2)),
            nn.Dropout(0.5),
            nn.Flatten(),
            nn.Linear(8600,2),
        )
    def forward(self,input):
        out=self.firstconv(input)
        return out
```

2.4 Explain the activation function (Relu,Leaky Relu,ELU)

a.Relu:

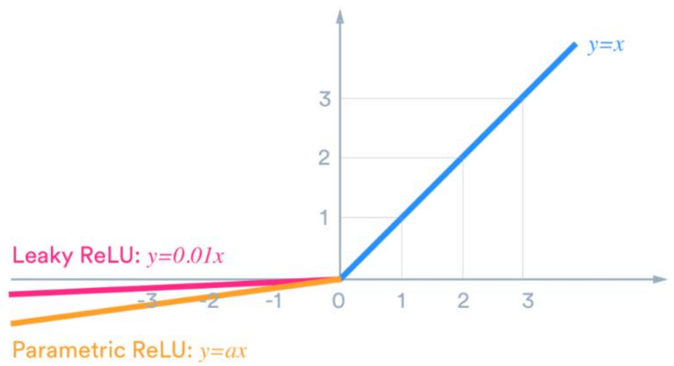


Relu 全名為 Rectified linear Units，會將所有負數變成 0，正數則保持不變，缺點是正值可能會無限大，而且將負數改為 0 會使產生負數的 neuron 不會再對 error 產生反應，當某個神經元輸出為 0 後，就難以再度輸出，這會導致產生 dead relu

優點:

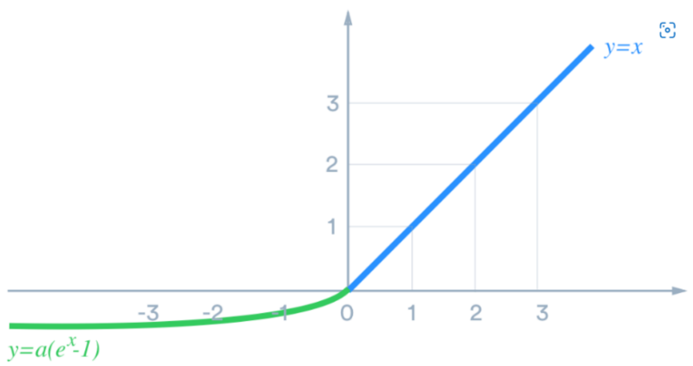
- 計算快速且不像 sigmoid 那樣產生梯度消失的效果。
- sparsely activated，因為負值為 0

b.leaky relu



Leaky relu 修正了一些 relu 的缺點，例如 dying relu 的問題，就能防止輸出為負號時有永遠無法被激活之問題，Leaky relu 同時也繼承 relu 的優點，但實際使用上並沒辦法證明 Leaky Relu 一定優於 Relu ，因次他只做為一個選項而不是預設

c.ELU



ELU 也是為了解決 ReLU 存在的問題而提出，並同時結合了 Leaky Relu 與 Relu 的優點，他也同時解決了 dying relu 的問題，然後在負值很大時也會飽和

3.Experimental results

在此我選用以下 hyper parameters

- optimizer:Adam
- criterion(loss):CrossEntropy loss
- epoch size :1000
- batch size :256
- learning rate for EEGNet:0.01
- learning rate for DeepConvNet:0.01

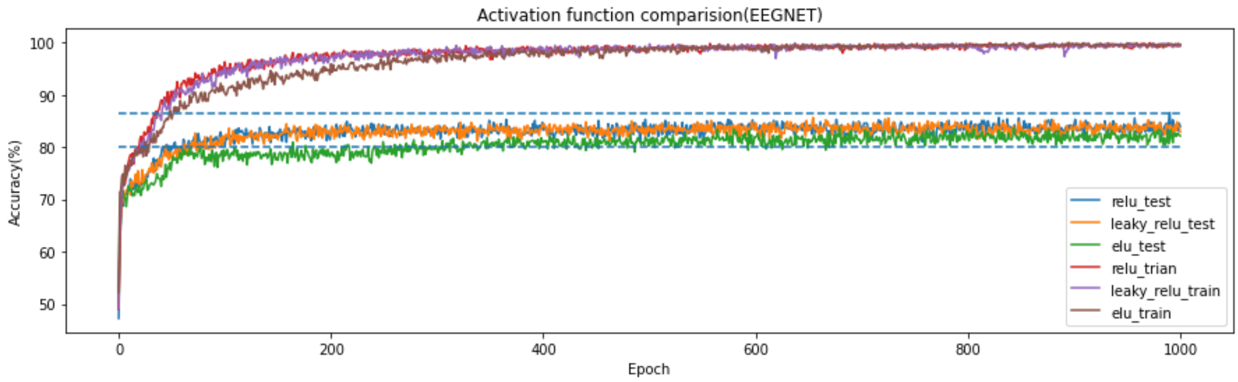
3.1 Highest testing accuracy

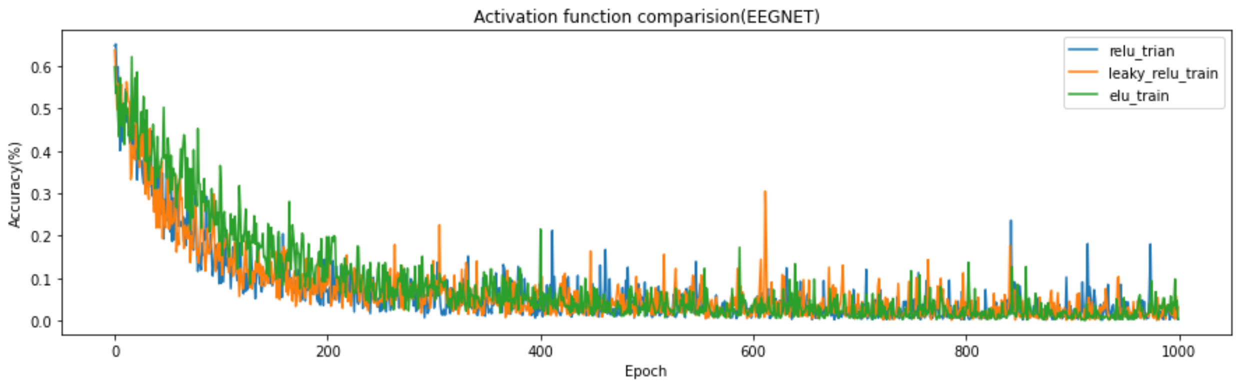
	ReLU	Leaky ReLY	ELU
EEGNet	**87.12%**	86.20%	84.07%
DeepConvNet	80.83%	81.20%	79.81%

3.2 anything you want to present

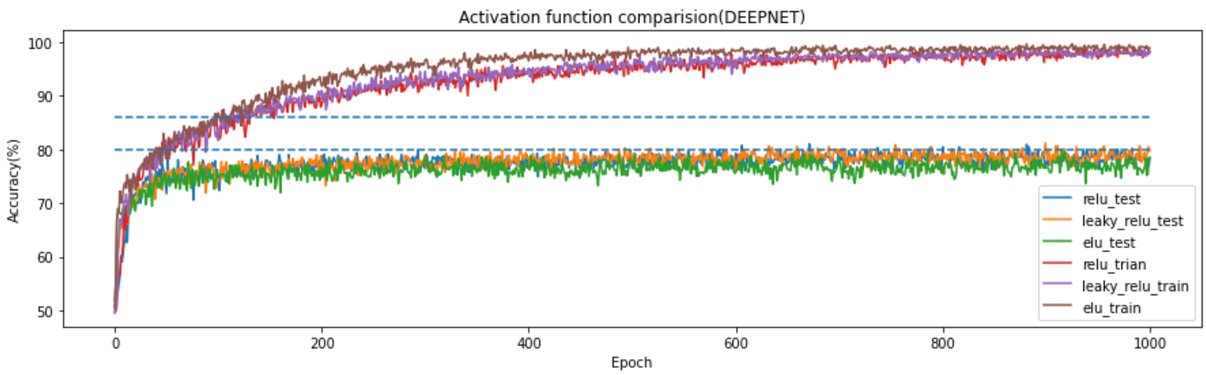
下面為 EEGNet 模型準確率和 Loss 的折線圖，當我們 epoch 調整到 1000，batch size 調整成 256 時，當 activation 為 relu 時，最高準確率可以達到 87.12%

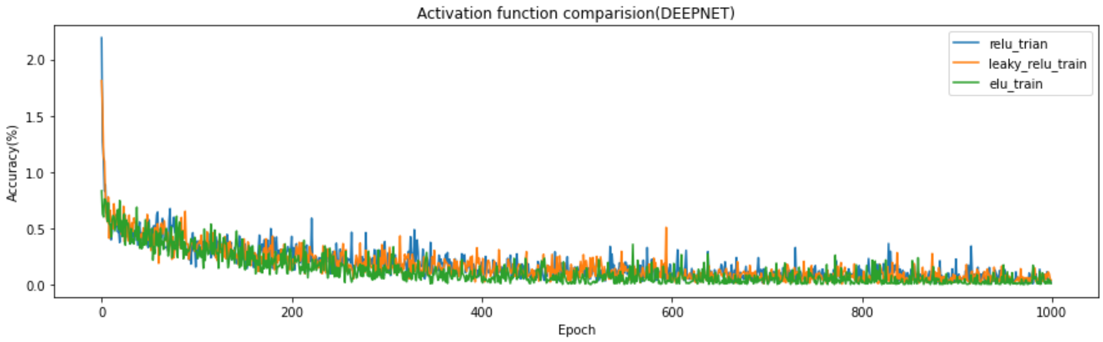
下圖則為 loss 之結果，可以看出 loss 都收斂得不錯





下面則是 DeepConvNet 之 accuracy 和 loss 比較圖，可以看出其準確率相較 EEGNet 是比較低的

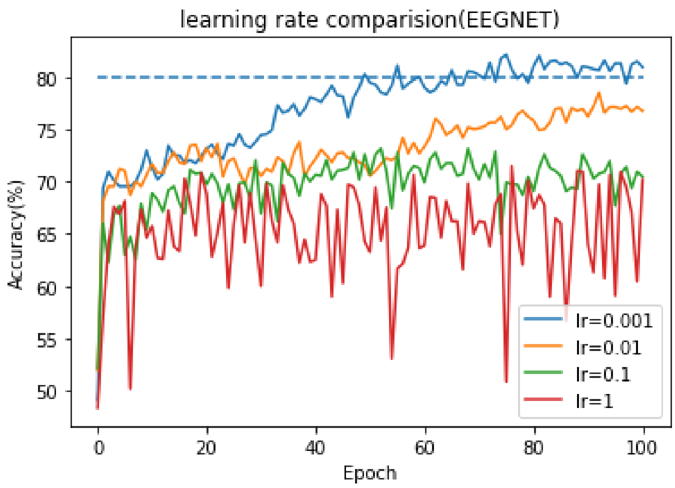


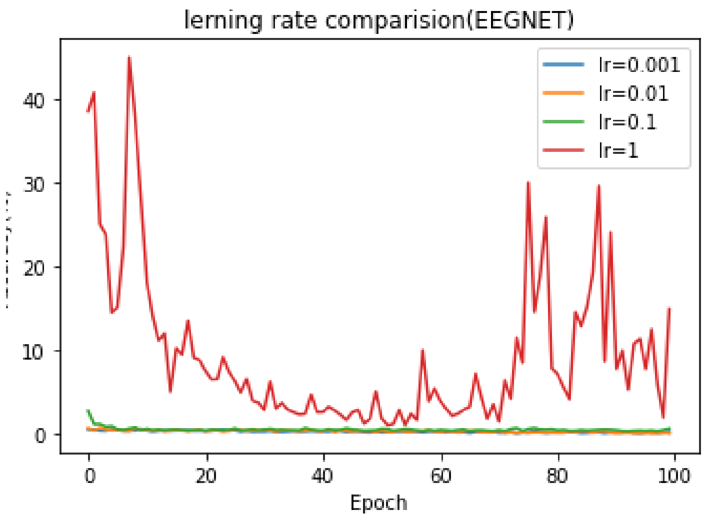


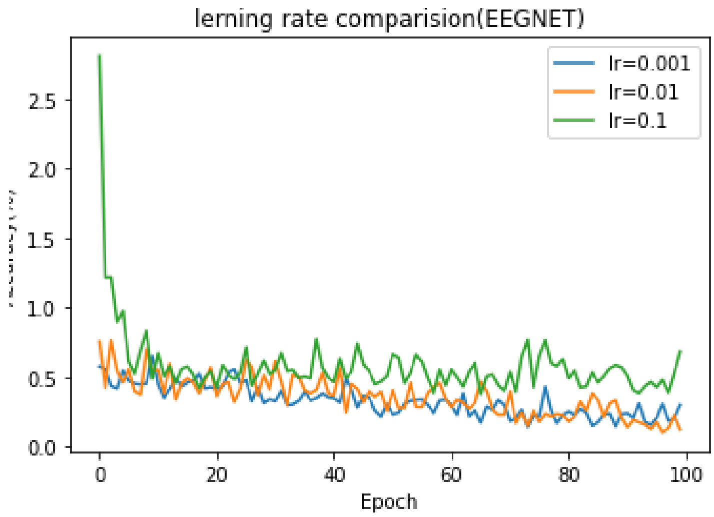
4.Discussion

4.1 使用不同 learning rate 作比較

在其他條件固定下，並將 epoch 設為 100，learning rate 越低其準確率明顯好，loss 的部分除了 lr=1 會產生劇烈震盪以外，當聚焦到細部，lr=0.001 及 0.01 則不會差太多

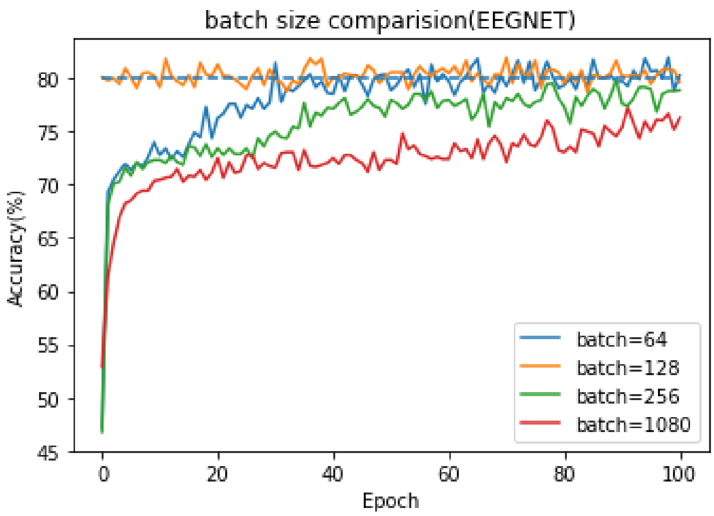


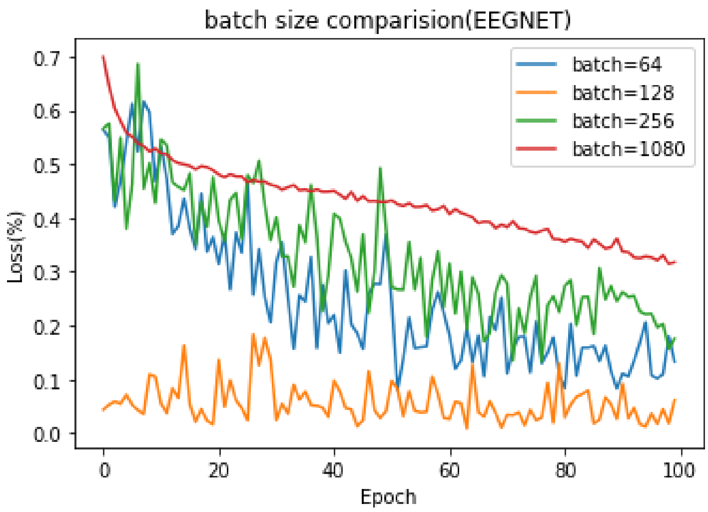




4.2 使用不同 batch size 做比較

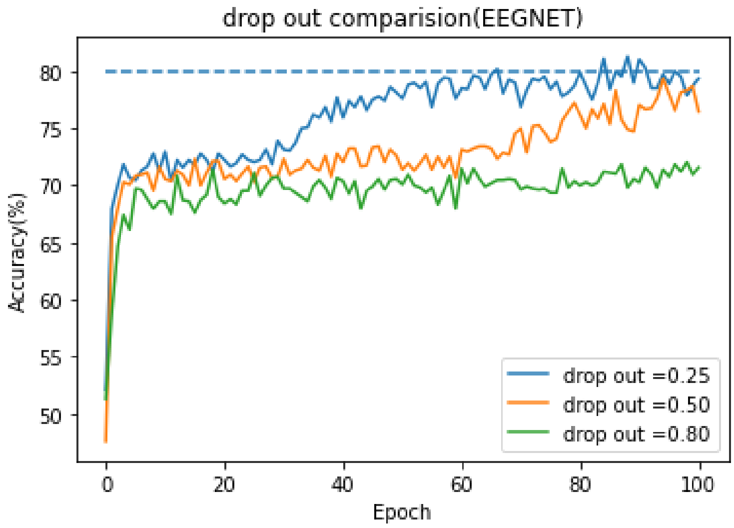
受限於電腦硬體效能調整 epoch 為 100 方便觀看訓練結果並看出準確率，有個令我比較意外的結果為 batch_size 128 和 64 的效果會是更好的，反而 batch_size 越高效果越差，batch size128 在 loss 和 accuracy 效果都相當好，但我認為可能跟 epoch 有關，epoch 調高其準確率可能在後面的結果會發生改變

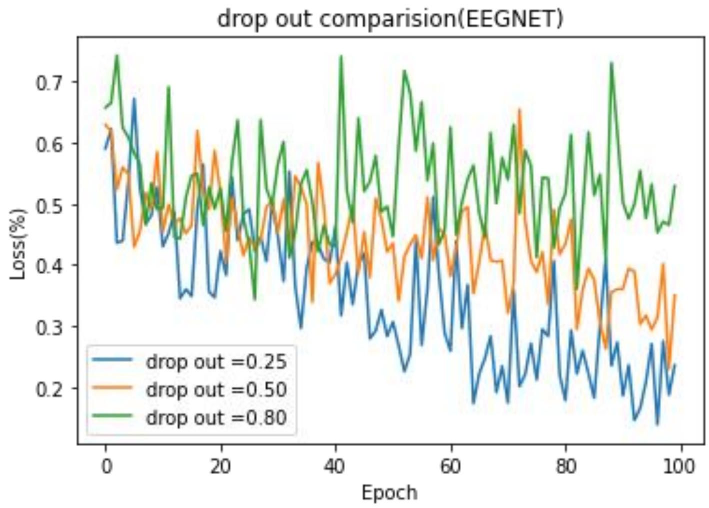




4.3 使用不同 drop out 做比較

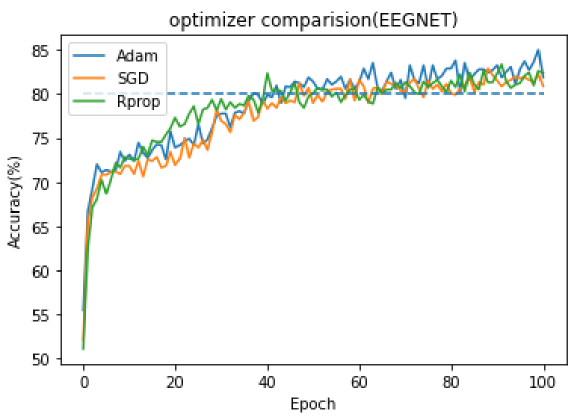
可以看出在預設參數 p=0.25 時效果是最好的，而越高的 drop out 效果越差

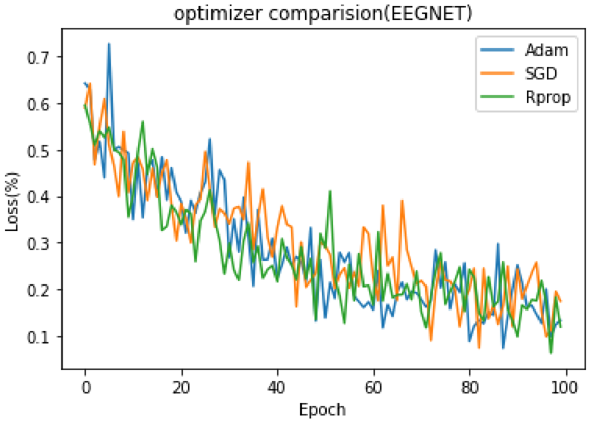




4.4 使用不同優化器進行比較

可以看出在 EPOCH 不大時沒辦法很明顯的看出哪一個優化器最好，因此可以說是各有優劣





optimizer comparision(EEGNET)
Adam
SGD
Rprop
Loss(%)
Epoch
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100