Lab2: EEG classification

學號：0756616
系所：多媒體工程學系
姓名：周冠伶

1. Introduction:

程式碼共分 7 個部分：

(1) import：使用的函式庫，包含 numpy、matplotlib、torch 等。
(2) variables：各項參數定義區，包含 Hyper Parameters。
(3) function：各項函式定義區，包含讀取檔案與原始數據繪圖函式。
(4) module(EEG Net)：EEGNet 定義區，包含 ReLU、ELU、Leaky ReLU 三種激勵函式。
(5) module(Deep Conv Net)：DeepConvNet，包含 ReLU、ELU、Leaky ReLU 三種激勵函式。
(6) produce data：讀取原始資料並轉換資料型態，並設定 DataSet 與 DataLoader。
(7) Lab2：主要程式碼，包含 Training phase、Testing phase、結果輸出等。

2. Experiment setups:

A. The detail of model

- EEGNet

  以 ReLU 函式為例，另外兩個則修改此圖的 torch.nn.ReLU()為 torch.nn.ELU(alpha=1.0)與 torch.nn.LeakyReLU()。

```
class EEGNet_LeakyReLU(torch.nn.Module):
    def __init__(self):
        super(EEGNet_LeakyReLU, self).__init__()
        self.firstconv = torch.nn.Sequential(
            torch.nn.Conv2d(
                in_channels=1,
                out_channels=16,
                kernel_size=(1, 51),
                stride=(1, 1),
                padding=(0, 25),
                bias=False),
            torch.nn.BatchNorm2d(
                num_features=16,
                eps=0.00001,
                momentum=0.1,
                affine=True,
                track_running_stats=True))
        self.depthwiseConv = torch.nn.Sequential(
            torch.nn.Conv2d(
                in_channels=16,
                out_channels=32,
                kernel_size=(2, 1),
                stride=(1, 1),
                groups=16,
                bias=False),
            torch.nn.BatchNorm2d(
                num_features=32,
                eps=0.00001,
                momentum=0.1,
                affine=True,
                track_running_stats=True), torch.nn.LeakyReLU(),
            torch.nn.AvgPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0),
            torch.nn.Dropout(p=0.25))
        self.separableConv = torch.nn.Sequential(
            torch.nn.Conv2d(
                in_channels=32,
                out_channels=32,
                kernel_size=(1, 15),
                stride=(1, 1),
                padding=(0, 7),
                bias=False),
            torch.nn.BatchNorm2d(
                num_features=32,
                eps=0.00001,
                momentum=0.1,
                affine=True,
                track_running_stats=True), torch.nn.LeakyReLU(),
            torch.nn.AvgPool2d(kernel_size=(1, 8), stride=(1, 8), padding=0),
            torch.nn.Dropout(p=0.25))
        self.classify = torch.nn.Sequential(
            torch.nn.Linear(in_features=736, out_features=2, bias=True))

    def forward(self, x):
        x = self.firstconv(x)
        x = self.depthwiseConv(x)
        x = self.separableConv(x)
        x = x.view(-1, 736)
        x = self.classify(x)

        return x
```

- DeepConvNet

  以 ReLU 函式為例，另外兩個則修改此圖的 torch.nn.ReLU()為

torch.nn.ELU(alpha=1.0)與 torch.nn.LeakyReLU()。

```
class DeepConvNet_ReLU(torch.nn.Module):
    def __init__(self):
        super(DeepConvNet_ReLU, self).__init__()
        self.Conv2d_1 = torch.nn.Sequential(
            torch.nn.Conv2d(
                in_channels=1, out_channels=25, kernel_size=(1, 5)))
        self.Conv2d_2 = torch.nn.Sequential(
            torch.nn.Conv2d(
                in_channels=25, out_channels=25, kernel_size=(2, 1)),
            torch.nn.BatchNorm2d(num_features=25, eps=0.00001, momentum=0.1),
            torch.nn.ReLU(), torch.nn.MaxPool2d(kernel_size=(1, 2)),
            torch.nn.Dropout(p=0.5))
        self.Conv2d_3 = torch.nn.Sequential(
            torch.nn.Conv2d(
                in_channels=25, out_channels=50, kernel_size=(1, 5)),
            torch.nn.BatchNorm2d(num_features=50, eps=0.00001, momentum=0.1),
            torch.nn.ReLU(), torch.nn.MaxPool2d(kernel_size=(1, 2)),
            torch.nn.Dropout(p=0.5))
        self.Conv2d_4 = torch.nn.Sequential(
            torch.nn.Conv2d(
                in_channels=50, out_channels=100, kernel_size=(1, 5)),
            torch.nn.BatchNorm2d(num_features=100, eps=0.00001, momentum=0.1),
            torch.nn.ReLU(), torch.nn.MaxPool2d(kernel_size=(1, 2)),
            torch.nn.Dropout(p=0.5))
        self.Conv2d_5 = torch.nn.Sequential(
            torch.nn.Conv2d(
                in_channels=100, out_channels=200, kernel_size=(1, 5)),
            torch.nn.BatchNorm2d(num_features=200, eps=0.00001, momentum=0.1),
            torch.nn.ReLU(), torch.nn.MaxPool2d(kernel_size=(1, 2)),
            torch.nn.Dropout(p=0.5))
        # CrossEntropy couldn't add softmax
        self.dense = torch.nn.Sequential(
            torch.nn.Linear(in_features=200 * 1 * 43, out_features=2))

    def forward(self, x):
        x = self.Conv2d_1(x)
        x = self.Conv2d_2(x)
        x = self.Conv2d_3(x)
        x = self.Conv2d_4(x)
        x = self.Conv2d_5(x)
        x = x.view(-1, 200 * 1 * 43)
        x = self.dense(x)

        return x
```

B. Explain the activation function(ReLU, Leaky ReLU, ELU)

激勵函數主要用途為轉換線性函式為非線性函式，使輸入和輸入脫離線性關係。三者激勵函數在正數輸入時，都會直接該正數出入作為輸出；負數輸入時，ReLU 輸出為 0、Leaky ReLU 輸出為該負數輸入乘以一個常數係數、ELU 輸出為該負數輸入的指數函數減去一再乘以一個常數係數。

ReLU 計算快，但是會有死亡節點等問題，Leaky ReLU 和 ELU 都是為了解決 ReLU 的問題而衍生出的激勵函數，其中 ELU 因需計算指

數函數而計算量較大。

3. Results of testing:

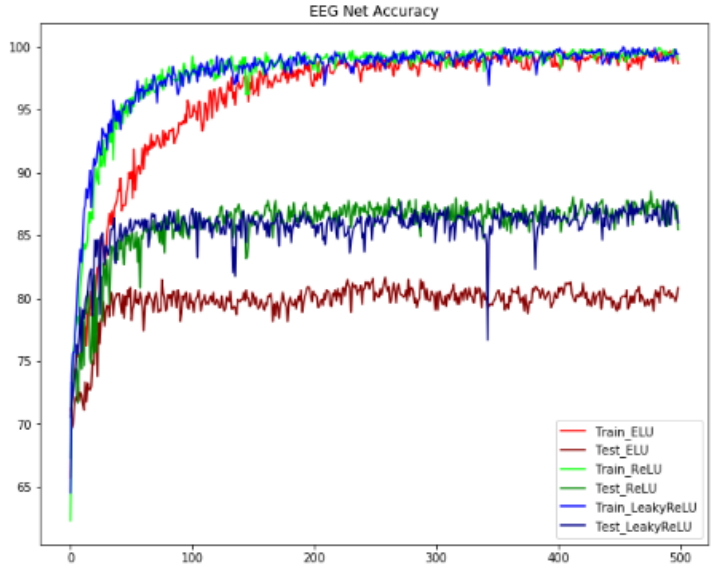
A. The highest testing accuracy

- Screenshot with two models

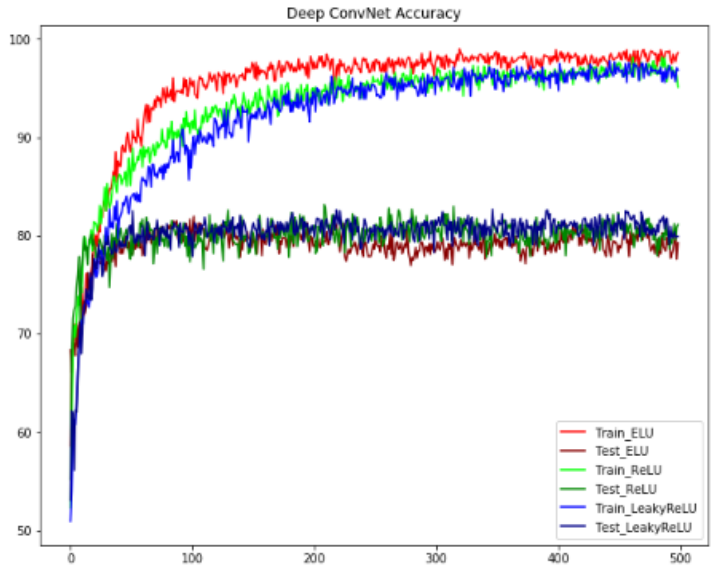
  BATCH_SIZE=64

  LEARNING_RATE=0.004

  EPOCH=500



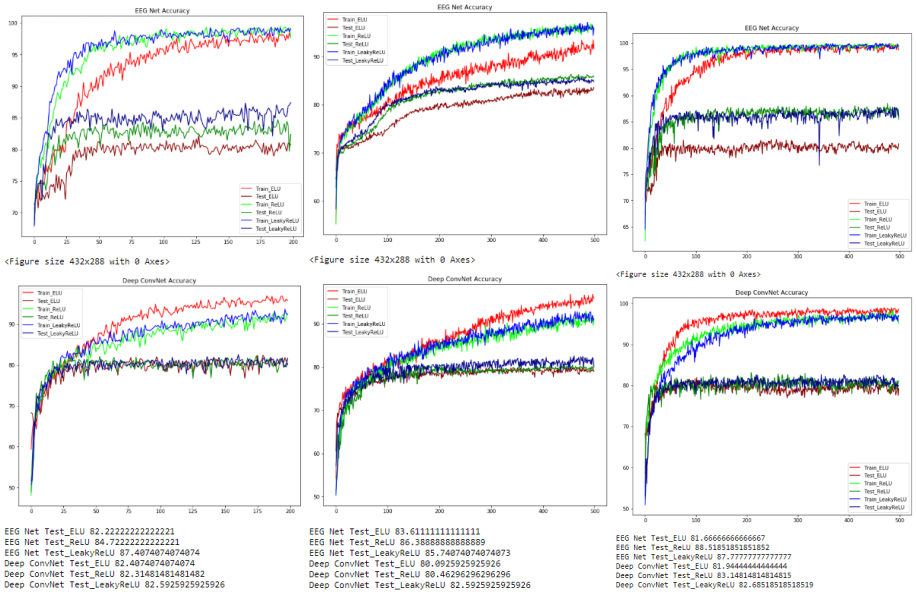

```
<Figure size 432x288 with 0 Axes>
```




```
EEG Net Test_ELU 81.66666666666667
EEG Net Test_ReLU 88.51851851851852
EEG Net Test_LeakyReLU 87.77777777777777
Deep ConvNet Test_ELU 81.94444444444444
Deep ConvNet Test_ReLU 83.14814814814815
Deep ConvNet Test_LeakyReLU 82.68518518518519
```

➢ Anything want to present

嘗試不同的 Hyper Parameters 之部分結果：



B. Comparison figures

Testing accuracy:

	EEG Net	Deep ConvNet
ReLU	**88.51**	83.14
Leaky ReLU	**87.77**	82.68
ELU	81.66	81.94

4. Discussion:

A. Anything want to share

1. EEG Net 表現都比 Deep ConvNet 好，顯示在該類型的資料中，較深層的網路表現反而比較差。
2. Epoch 往上提升，只會讓 Training accuracy 提高、不會讓 Testing accuracy 提高。
3. 三種激勵函數中，表現較好的是 ReLU 和 Leaky ReLU、ELU 表現都明顯較差。

4. Learning rate 會影響 Accuracy 的震盪幅度。
5. 使用 Random seed 可以固定初始化參數，使結果重現(類似於儲存參數)。