DLP HW3

數據所 311554019 宋沛潔

1. Introduction:

在這次的 lab 要將 BCI competition dataset 進行分類，其中使用 EEGNet, DeepConvNet 兩種模型搭配 ReLU, Leaky ReLU, ELU 三種 activation function 訓練。其中每個 sample 包含兩個 channels。輸入大小：(B, 1, 2, 750)，輸出大小：(B, 2)， B 是 batch size。

2. Experiment set up:

A. The detail of your model

1. EEGNet

下方的圖為 EEGNet 的架構。共有四層架構，雖然是分類但是最後不用加上 softmax 的原因是再計算 loss 的時候是用 cross entropy loss。他與一般 classification model 不同的地方是減少了參數量，但最後的結果預測準確度卻不會下降。其中 depthwiseConv 層可以學習各自 channel 而不受干擾；separableConv 會學習如何有效結合 feature。

```
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): ReLU()
    (3): AvgPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0)
    (4): Dropout(p=0.25, inplace=False)
  )
  (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): ReLU()
    (3): AvgPool2d(kernel_size=(1, 8), stride=(1, 8), padding=0)
    (4): Dropout(p=0.25, inplace=False)
  )
  (classify): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=736, out_features=2, bias=True)
  )
)
```

```python
class EEGNet(nn.Module):
    def __init__(self, activation=nn.ELU() ):
        super(EEGNet, self).__init__()

        # Layer 1
        self.firstconv = nn.Sequential(
            nn.Conv2d(1, 16, (1, 51), stride=(1, 1), padding=(0, 25), bias=False),
            nn.BatchNorm2d(16, eps=1e-05, momentum=0.1),
        )

        # Layer 2
        self.depthwiseConv = nn.Sequential(
            nn.Conv2d(16, 32, (2, 1), stride=(1, 1), groups=16, bias=False),
            nn.BatchNorm2d(32, eps=1e-05, momentum=0.1),
            activation,
            nn.AvgPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0),
            nn.Dropout(p=0.25),
        )

        # Layer 3
        self.separableConv = nn.Sequential(
            nn.Conv2d(32, 32, (1, 15), stride=(1, 1), padding=(0, 7), bias=False),
            nn.BatchNorm2d(32, eps=1e-05, momentum=0.1),
            activation,
            nn.AvgPool2d(kernel_size=(1, 8), stride=(1, 8), padding=0),
            nn.Dropout(p=0.25),
        )

        # FC Layer
        self.classify = nn.Sequential(nn.Flatten(), nn.Linear(736, 2, bias=True))
```

```python
def forward(self, x):

    x = self.firstconv(x)
    x = self.depthwiseConv(x)
    x = self.separableConv(x)
    x = self.classify(x)
    return x
```

2. DeepConvNet

下方的圖為 DeepConvNet 的架構是一個 CNN 的模型。包含六層，最後一層為分類層。

```
DeepConvNet(
  (conv1): Conv2d(1, 25, kernel_size=(1, 5), stride=(1, 1))
  (conv2): Sequential(
    (0): Conv2d(25, 25, kernel_size=(2, 1), stride=(1, 1))
    (1): BatchNorm2d(25, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.5, inplace=False)
  )
  (conv3): Sequential(
    (0): Conv2d(25, 50, kernel_size=(1, 5), stride=(1, 1))
    (1): BatchNorm2d(50, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.5, inplace=False)
  )
  (conv4): Sequential(
    (0): Conv2d(50, 100, kernel_size=(1, 5), stride=(1, 1))
    (1): BatchNorm2d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.5, inplace=False)
  )
  (conv5): Sequential(
    (0): Conv2d(100, 200, kernel_size=(1, 5), stride=(1, 1))
    (1): BatchNorm2d(200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.5, inplace=False)
  )
  (out): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=8600, out_features=2, bias=True)
  )
)
```

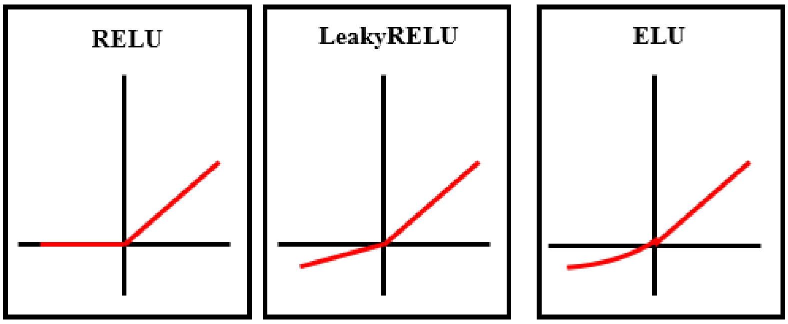
```python
class DeepConvNet(nn.Module):
    def __init__(self, activation=nn.ELU()):
        super(DeepConvNet, self).__init__()

        self.conv1 = nn.Conv2d(1, 25, kernel_size=(1, 5))
        self.conv2 = nn.Sequential(
            nn.Conv2d(25, 25, kernel_size=(2, 1)),
            nn.BatchNorm2d(25, eps=1e-5, momentum=0.1),
            activation,
            nn.MaxPool2d(kernel_size=(1, 2)),
            nn.Dropout(p=0.5),
        )
        self.conv3 = nn.Sequential(
            nn.Conv2d(25, 50, kernel_size=(1, 5)),
            nn.BatchNorm2d(50, eps=1e-5, momentum=0.1),
            activation,
            nn.MaxPool2d(kernel_size=(1, 2)),
            nn.Dropout(p=0.5),
        )
        self.conv4 = nn.Sequential(
            nn.Conv2d(50, 100, kernel_size=(1, 5)),
            nn.BatchNorm2d(100, eps=1e-5, momentum=0.1),
            activation,
            nn.MaxPool2d(kernel_size=(1, 2)),
            nn.Dropout(p=0.5),
        )
        self.conv5 = nn.Sequential(
            nn.Conv2d(100, 200, kernel_size=(1, 5)),
            nn.BatchNorm2d(200, eps=1e-5, momentum=0.1),
            activation,
            nn.MaxPool2d(kernel_size=(1, 2)),
            nn.Dropout(p=0.5),
        )
        self.out = nn.Sequential(nn.Flatten(), nn.Linear(8600, 2))
```

```python
def forward(self, x):

    x = self.conv1(x)
    x = self.conv2(x)
    x = self.conv3(x)
    x = self.conv4(x)
    x = self.conv5(x)
    x = self.out(x)
    return x
```

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



1. ReLU:

   當值為負時，皆為 0，代表小於 0 不會去更新梯度，但是可以去避免梯度消失的問題

   $ReLU(x) = max(0, x)$

2. Leaky ReLU:

   具有 Relu 的特點，當大於 0 時跟 Relu 相同，但小於 0 時不為 0，特別乘一個α，可以解決不會去更新梯度的問題。

   $$LeakyReLU(x) = \begin{cases} x, if\ x \geq 0 \\ \alpha \cdot x, if\ x < 0 \end{cases}$$

3. ELU:

   具有 Relu 的特點，當大於 0 時跟 Relu 相同，但小於 0 時，會呈現指數的樣子，可以更快收斂且產生更準的結果。

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

3. Experiment set up:

3.1. The highest testing accuracy

對於參數的設定為:

- Batch size: EEGNet :256 / DeepConvNet :128
- Epoch: 500
- Learning rate: 5e-4
- Weight decay: 2e-3
- Optimizer: Adam
- Loss function: cross entropy

3.1.1 Screenshot with two models

EEGNet:

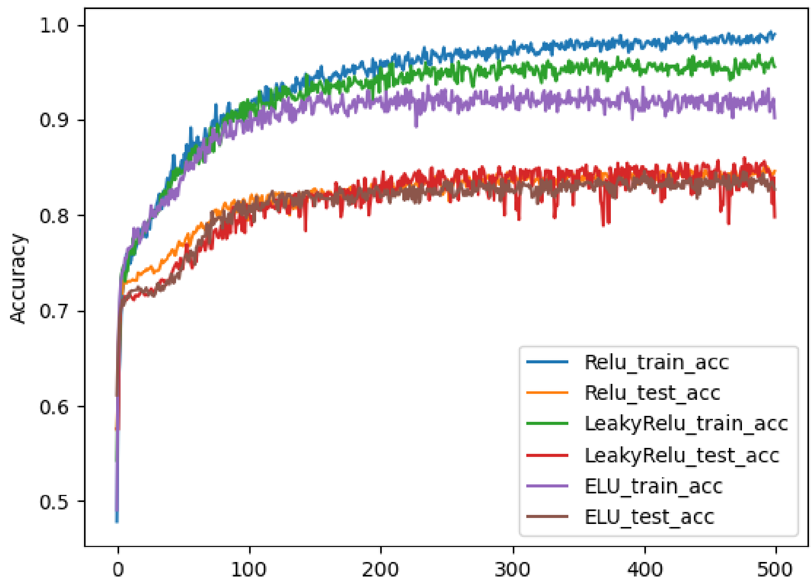
```
Relu_train_max: 0.9925925925925926
Relu_test_max: 0.850925925925926
LeakyRelu_train_max: 0.9685185185185186
LeakyRelu_test_max: 0.8601851851851852
ELU_train_max: 0.9361111111111111
ELU_test_max: 0.8462962962962963
```

DeepConvNet:

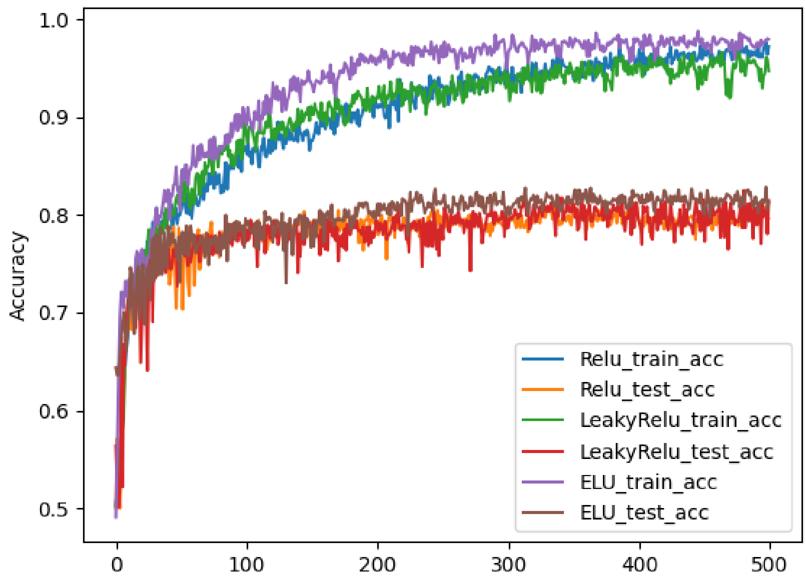
```
Relu_train_max: 0.9768518518518519
Relu_test_max: 0.812037037037037
LeakyRelu_train_max: 0.9666666666666667
LeakyRelu_test_max: 0.825
ELU_train_max: 0.9879629629629629
ELU_test_max: 0.8287037037037037
```

3.2. Comparison figures

3.2.1 EEGNet



3.2.2 DeepConvNet

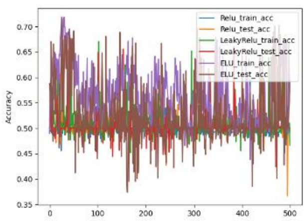


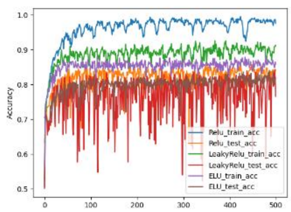
4. Discussion:

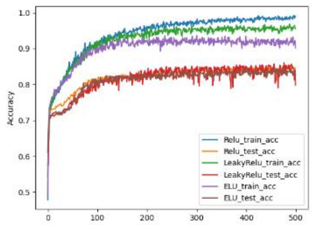
這個作業中，我有嘗試加入多種 scheduler，但是結果好像都沒有比較好。
此外，learning rate 的設定越小越好，所試過的 optimizier 是 adam 最好，且要加入 regularization term，收斂會比較穩定。
而不同的 activation function，EEGNet 是 leakyrelu 表現最好，DeepConvNet 是 ELU。
以下是一些 hyperparameter 的比較：

1. Learning rate :
   不管是哪一種 model，learning rate 的設定越小越好，準確度高且收斂穩定。
   - EEGNet:

     lr = 0.5: lr = 0.005: lr = 0.0005:







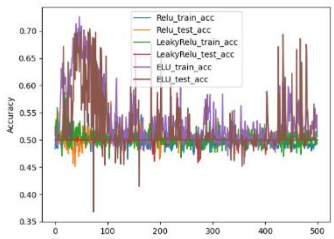

```
Relu_train_max: 0.5944444444444444
Relu_test_max: 0.6064814814814815
LeakyRelu_train_max: 0.6703703703703704
LeakyRelu_test_max: 0.6601851851851852
ELU_train_max: 0.7185185185185186
ELU_test_max: 0.7157407407407408
```

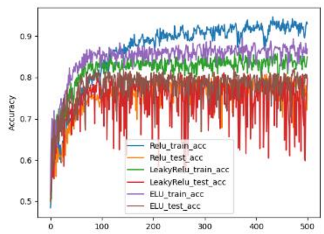
```
Relu_train_max: 0.9953703703703703
Relu_test_max: 0.8574074074074074
LeakyRelu_train_max: 0.9259259259259259
LeakyRelu_test_max: 0.8416666666666667
ELU_train_max: 0.8814814814814815
ELU_test_max: 0.8398148148148148
```

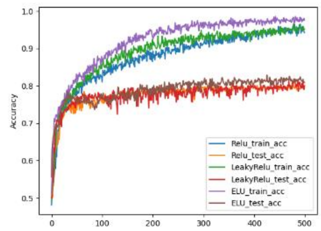
```
Relu_train_max: 0.9925925925925926
Relu_test_max: 0.850925925925926
LeakyRelu_train_max: 0.9685185185185186
LeakyRelu_test_max: 0.8601851851851852
ELU_train_max: 0.9361111111111111
ELU_test_max: 0.8462962962962963
```

   - DeepConvNet:

     lr = 0.5: lr = 0.005: lr = 0.0005:








```
Relu_train_max: 0.5481481481481482
Relu_test_max: 0.5703703703703704
LeakyRelu_train_max: 0.587037037037037
LeakyRelu_test_max: 0.5194444444444445
ELU_train_max: 0.725925925925926
ELU_test_max: 0.7037037037037037
```

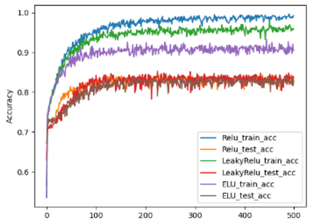
```
Relu_train_max: 0.9462962962962963
Relu_test_max: 0.8074074074074075
LeakyRelu_train_max: 0.862037037037037
LeakyRelu_test_max: 0.812037037037037
ELU_train_max: 0.8953703703703704
ELU_test_max: 0.8212962962962963
```

```
Relu_train_max: 0.9768518518518519
Relu_test_max: 0.812037037037037
LeakyRelu_train_max: 0.9666666666666667
LeakyRelu_test_max: 0.825
ELU_train_max: 0.9879629629629629
ELU_test_max: 0.8287037037037037
```

2. Batch size:
   不管是哪種模型，batch size 越大表一次訓練的 data 比較多，因此越大準確度高，且運算較快。

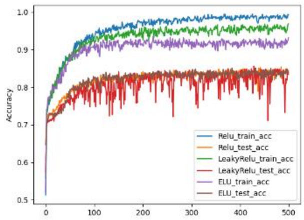
- EEGNet:

Batch size = 64



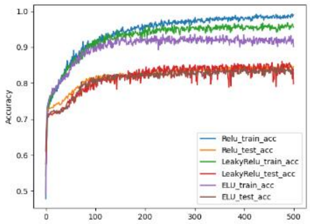

```
Relu_train_max: 0.9962962962962963
Relu_test_max: 0.8435185185185186
LeakyRelu_train_max: 0.9768518518518519
LeakyRelu_test_max: 0.8527777777777777
ELU_train_max: 0.9324074074074075
ELU_test_max: 0.8407407407407408
```

Batch size = 128




```
Relu_train_max: 0.9944444444444445
Relu_test_max: 0.8537037037037037
LeakyRelu_train_max: 0.9685185185185186
LeakyRelu_test_max: 0.8555555555555555
ELU_train_max: 0.9342592592592592
ELU_test_max: 0.850925925925926
```

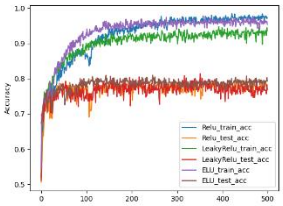
Batch size = 256




```
Relu_train_max: 0.9925925925925926
Relu_test_max: 0.850925925925926
LeakyRelu_train_max: 0.9685185185185186
LeakyRelu_test_max: 0.8601851851851852
ELU_train_max: 0.9361111111111111
ELU_test_max: 0.8462962962962963
```

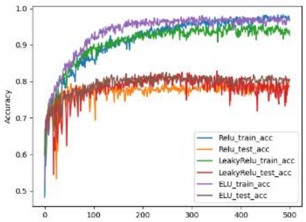
- DeepConvNet:

Batch size = 32



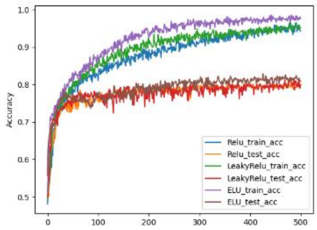

```
Relu_train_max: 0.9842592592592593
Relu_test_max: 0.7935185185185185
LeakyRelu_train_max: 0.9472222222222222
LeakyRelu_test_max: 0.8138888888888889
ELU_train_max: 0.9768518518518519
ELU_test_max: 0.8092592592592592
```

Batch size = 64




```
Relu_train_max: 0.9824074074074074
Relu_test_max: 0.8
LeakyRelu_train_max: 0.9592592592592593
LeakyRelu_test_max: 0.8287037037037037
ELU_train_max: 0.9805555555555555
ELU_test_max: 0.825
```

Batch size = 128




```
Relu_train_max: 0.9768518518518519
Relu_test_max: 0.812037037037037
LeakyRelu_train_max: 0.9666666666666667
LeakyRelu_test_max: 0.825
ELU_train_max: 0.9879629629629629
ELU_test_max: 0.8287037037037037
```