Deep Learning and Practice Lab 2

311605011

黃品振

1. Introduction:

這次的作業是用 PyTorch 建構出 EEGNET 以及 DeepConvNet，並且用 ELU、ReLU 以及 LeakyReLU 三種 activation function 來觀察結果。

2. Experiment set up:

A. Detail of my model:

- Data processing:

```
#data processing
train_data, train_label, test_data, test_label = read_bci_data()
trainingdataset = TensorDataset(torch.from_numpy(train_data), torch.from_numpy(train_label))
loader_train = DataLoader(dataset = trainingdataset, batch_size = 128, shuffle = True)
testdataset =  TensorDataset(torch.from_numpy(test_data), torch.from_numpy(test_label))
loader_test = DataLoader(dataset = testdataset, batch_size = 128, shuffle = True)
```

利用 TensorDataset 將助教已經處理好的 dataset 讀取進來，以利使用

- EEGNET:

```
#def network
class EEGNET(nn.Module):
    def __init__(self, activation = nn.ELU(), dropout = 0.25):
        super(EEGNET, self).__init__()
        self.firstconv = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size = (1, 51), stride = (1, 1), padding = (0, 25), bias = True),
            nn.BatchNorm2d(16)
        )
        self.depthwiseConv = nn.Sequential(
            nn.Conv2d(16, 32, kernel_size = (2, 1), stride = (1, 1), groups = 16, bias = True),
            nn.BatchNorm2d(32),
```

```
            activation,
            nn.AvgPool2d(kernel_size = (1, 4), stride = (1, 4), padding = 0),
            nn.Dropout(p = dropout)
        )
        self.separableConv = nn.Sequential(
            nn.Conv2d(32, 32, kernel_size = (1, 15), stride = (1, 1), padding =
(0, 7), bias = True),
            nn.BatchNorm2d(32),
            activation,
            nn.AvgPool2d(kernel_size = (1, 8), stride = (1, 8), padding = 0),
            nn.Dropout(p = dropout)
        )
        self.classify = nn.Linear(in_features = 736, out_features = 2, bias =
True)
    def forward(self, x):
        output = self.firstconv(x)
        output = self.depthwiseConv(output)
        output = self.separableConv(output)
        #攤平:
        output = output.view(output.shape[0], -1)
        output = self.classify(output)
        return output
```

上圖為將助教在 PPT 上給的 EEGNET 架構實作出來的結果。先分別建出一段一段，再透過 forward function 將前一個 output 往後送。另外，因為使用 crossentropy 當作 loss，所以最後就不用加 softmax。

- DeepConvNet:

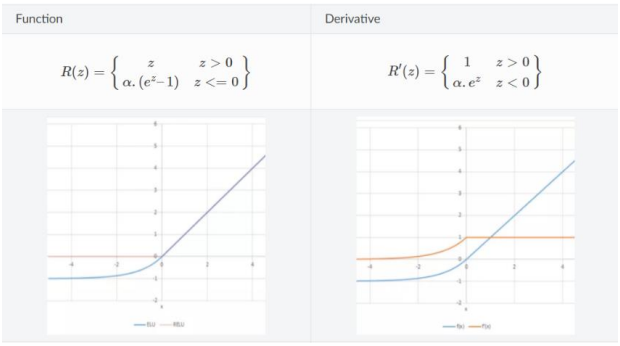
```
class DeepConvNet(nn.Module):
    def __init__(self, activation = nn.ELU(), filters = [25, 50, 100, 200],
dropout = 0.5):
        super(DeepConvNet, self).__init__()
        filters = filters
        self.conv0 = nn.Sequential(
            nn.Conv2d(1, filters[0], kernel_size = (1, 5), stride = (1,
1),padding = (0, 0), bias = True),
            nn.Conv2d(filters[0], filters[0], kernel_size = (2, 1), padding =
(0, 0), bias = True),
            nn.BatchNorm2d(filters[0]),
            activation,
```

```
                nn.MaxPool2d(kernel_size = (1, 2)),
                nn.Dropout(p = dropout)
            )
            for i in range(1, len(filters)):
                setattr(self, 'conv'+str(i), nn.Sequential(
                    nn.Conv2d(filters[i-1], filters[i], kernel_size = (1, 5), stride
= (1, 1), padding = (0, 0), bias = True),
                    nn.BatchNorm2d(filters[i]),
                    activation,
                    nn.MaxPool2d(kernel_size = (1, 2)),
                    nn.Dropout(p = dropout)
                ))
            self.classify = nn.Linear(8600,2)
        def forward(self, x):
            out = self.conv0(x)
            out = self.conv1(out)
            out = self.conv2(out)
            out = self.conv3(out)
            out = out.view(out.shape[0], -1)
            out = self.classify(out)
            return out
```

上圖為將助教在 PPT 上給的 DeepConvNet 架構實作出來的結果。也是先分別建出一段一段，再透過 forward function 將前一個 output 往後送。最後透過用 torchsummary 得到要輸入全連接層的數量是 8600 個，然後分類是兩個 class，所以輸出是 2 個。也是因為使用 crossentropy 當作 loss，所以最後就不用加 softmax。

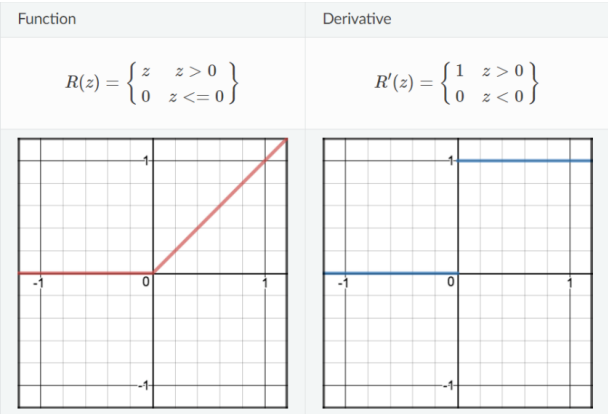
B. Explain the activation function:

- ELU:



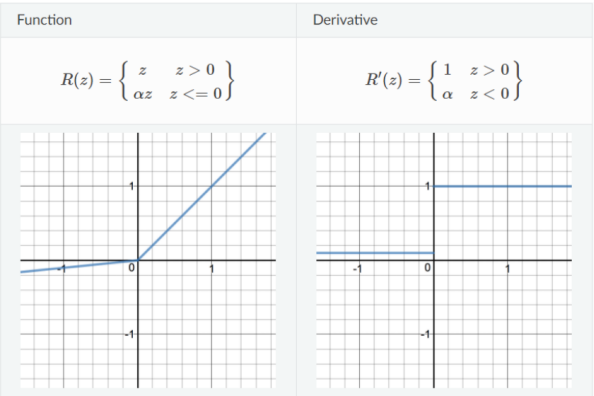
ELU 全名是 Exponential Linear Unit，ELU 的特性是可以有負值，且圖形較為平滑，缺點是正值可能會無限大

- ReLU:



ReLU 全名是 Recitified Linear Units，ReLU 可以有效的解決 gradient vanishing 的問題，ReLU 的運算式簡單，讓訓練可以比較快速，但是 ReLU 也具有顯著的缺點，像是 x<0 的時候，梯度會變成 0，可能使權重無法更新，還有正值可能無限大的問題同樣存在。

- LeakyReLU:



Leaky ReLU 具有 ReLU 的特性，但不同的是，會在負數的地方乘以一個比較小的值，解決權重無法更新的問題。

3. Experimental results:

A. The highest testing accuracy:

- Screenshot with two models:
  a. EEGNET:

```
EEGNET:
MaxELUaccuracy: 83.70
MaxReLUaccuracy: 87.69
MaxLeakyReLUaccuracy: 85.19
Highest accuracy:87.69%
```

以上結果是在 learning rate=0.01，epochs=2000，batch size=128 的時候跑出來的結果，最高準確率為 88.61%。

  b. DeepConvNet:

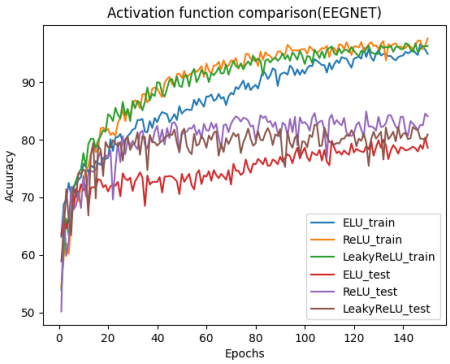
```
DeepConvNet:
MaxELUaccuracy: 81.48
MaxReLUaccuracy: 83.06
MaxLeakyReLUaccuracy: 81.67
Highest accuracy: 83.06%
```

以上結果是在 learning rate =0.01，epochs=1000，batch size=128 的時候跑出來的結果，最高準確率為 83.06%。

-

B. Anything I want to present:

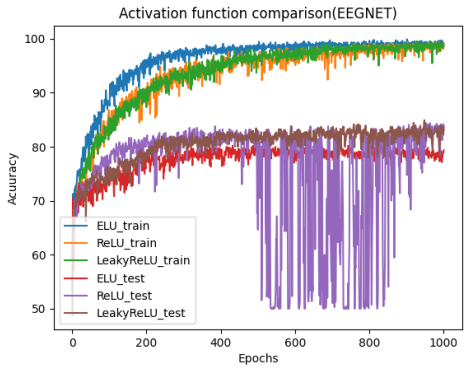
- EEGNET:
  我先將參數設為:lr=0.01、epochs=150、Batchsize=128，得到以下結果:




```
EEGNET:
MaxELUaccuracy: 80.37
MaxReLUaccuracy: 84.91
MaxLeakyReLUaccuracy: 82.59
Highest accuracy:84.91%
```

表現最好的準確率也不超過 85%。

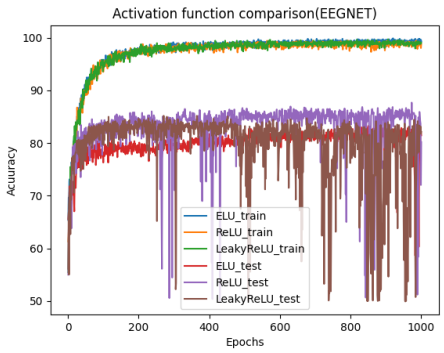
接著我將參數調整成 lr=0.01、epochs=1000、Batchsize=1000，得到以下結果:




```
EEGNET:
MaxELUaccuracy: 80.65
MaxReLUaccuracy: 84.35
MaxLeakyReLUaccuracy: 84.91
Highest accuracy:84.91%
```

最高準確率達到了 84.91，可以發現，不是 Epoch 跟 Batchsize 越大訓練結果就會越好，還是要透過適當的調整才行。

最後我將參數設為 lr=0.01、epochs=500、Batchsize=128，得到以下結果:

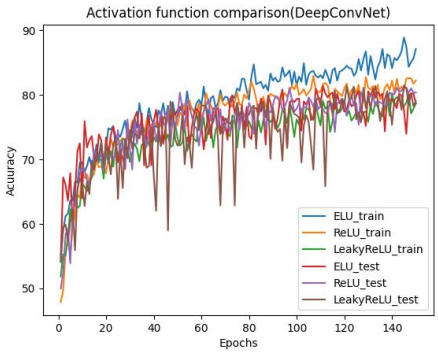



```
EEGNET:
MaxELUaccuracy: 83.70
MaxReLUaccuracy: 87.69
MaxLeakyReLUaccuracy: 85.19
Highest accuracy:87.69%
```

上圖可以發現，最高準確率有達到 87%算是還不錯的結果，雖然最高準確率較高的 ReLU 函數以及 LeakyReLU 函數中間有一點震盪，但最後的結果仍然不錯。

- DeepConvNet:

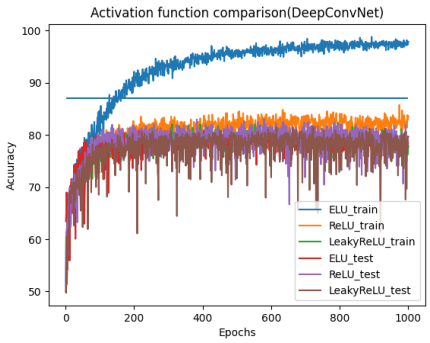
  我先將參數設為 lr=0.01、epochs=150、Batchsize=128，得到以下結果:




```
DeepConvNet:
MaxELUaccuracy: 81.39
MaxReLUaccuracy: 81.20
MaxLeakyReLUaccuracy: 81.48
Highest accuracy: 81.48%
```

最高準確率達到 81.48%，可以發現 ELU 的 activation function 在 Training accuracy 的表現最好。

接著我將參數調整成 lr=0.01、epochs=1000、Batchsize=128，得到以下結果:



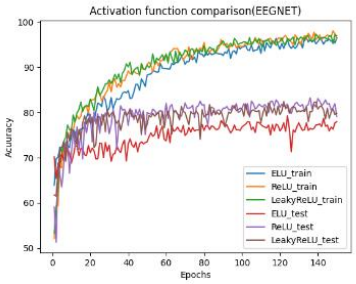
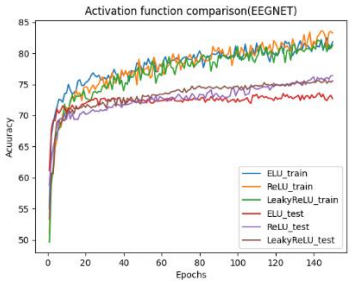
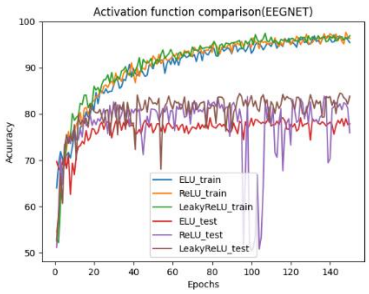

```
DeepConvNet:
MaxELUaccuracy: 81.48
MaxReLUaccuracy: 83.06
MaxLeakyReLUaccuracy: 81.67
Highest accuracy: 83.06%
```

從上圖可以發現，達到的最高準確率變高了，可能是因為 epochs=150 的時候還沒有收斂到 local minimum，所以調高 epoch 後得到的結果比較好。

4. Discussion:

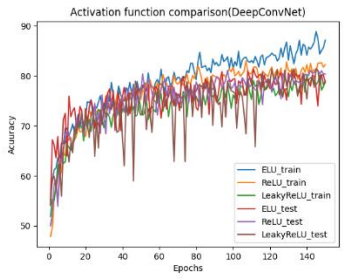
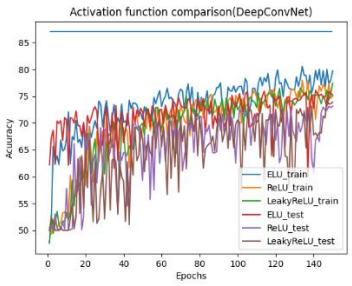
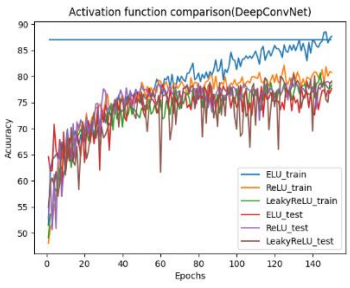
A. Anything I want to present:

我另外使用了兩種 Optimizer，一種是 SGD，另一種是 Adagrad，在 epochs 同樣設 150、batchsize 設 128 在 EEGNET 的表現對比圖如下:

Adam	SGD	Adagrad
		
EEGNET: MaxELUaccuracy: 82.04 MaxReLUaccuracy: 82.78 MaxLeakyReLUaccuracy: 84.26 Highest accuracy:84.26%	EEGNET: MaxELUaccuracy: 75.56 MaxReLUaccuracy: 80.19 MaxLeakyReLUaccuracy: 78.24 Highest accuracy:80.19%	EEGNET: MaxELUaccuracy: 81.39 MaxReLUaccuracy: 86.67 MaxLeakyReLUaccuracy: 83.06 Highest accuracy:86.67%

從上圖可以發現還是 Adam 表現得最好，Adagrad 的最高準確率雖然最高，但是達到最高準確率的 ReLU 圖形並不穩定。

下圖是在同樣的參數之下，在 DeepConvNet 的表現對比圖:

Adam	SGD	Adagrad
		
DeepConvNet: MaxELUaccuracy: 81.39 MaxReLUaccuracy: 81.20 MaxLeakyReLUaccuracy: 81.48 Highest accuracy: 81.48%	DeepConvNet: MaxELUaccuracy: 76.11 MaxReLUaccuracy: 73.15 MaxLeakyReLUaccuracy: 74.54 Highest accuracy: 76.11%	DeepConvNet: MaxELUaccuracy: 78.89 MaxReLUaccuracy: 79.63 MaxLeakyReLUaccuracy: 80.46 Highest accuracy: 80.46%

可以發現到 Adam 表現的一樣是最好的，在 DeepConvNet 中，Adagrad 的震盪就變小了，而 SGD 的震盪反而較大。