NYCU DL Lab2 – EEG classification

312551068 張紀睿

1. Introduction

在本次作業中，利用 PyTorch framework 建立 EEGNet 與 DeepConvNet 模型，針對 BCI Competition 的 EEG 資料集進行模型訓練，並嘗試 ELU, ReLU, LeakyReLU 三種不同的 Activation Function，觀察不同 function 對模型訓練的影響，另外也將透過調整訓練的各項 parameter 如 lr, batchsize 以觀察結果變化。

2. Experimental set up

A. The detail of your model

EEGNet

根據簡報中的 Implementation details 建立 EEGNet，並可傳入參數 activation 來更改使用的 activation function。

```
class EEGNet(nn.Module):
    def __init__(self, activation = "ELU"):
        super().__init__()
        if activation == "ELU":
            activation_funct = nn.ELU(alpha = 1.0)
        elif activation == "ReLU":
            activation_funct = nn.ReLU()
        else:
            activation_funct = nn.LeakyReLU()

        self.firstconv = nn.Sequential(
            nn.Conv2d(1, 16, kernel_size = (1,51), stride=(1, 1), padding =(0, 25), bias = False),
            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 = False),
            nn.BatchNorm2d(32, eps=1e-05, momentum = 0.1, affine = True, track_running_stats = True),
            activation_funct,
            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 = False),
            nn.BatchNorm2d(32, eps=1e-05, momentum = 0.1, affine = True, track_running_stats = True),
            activation_funct,
            nn.AvgPool2d(kernel_size = (1, 8), stride = (1, 8), padding = 0),
            nn.Dropout(p = 0.25)
        )
        self.classify = nn.Sequential(
            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

根據簡報中的 table 建立 DeepConvNet，並可傳入參數 activation 來更改使用的 activation function。

```python
class DeepConvNet(nn.Module):
    def __init__(self, activation = "ELU"):
        super().__init__()
        if activation == "ELU":
            activation_funct = nn.ELU(alpha = 1.0)
        elif activation == "ReLU":
            activation_funct = nn.ReLU()
        else:
            activation_funct = nn.LeakyReLU()
        self.layers = nn.Sequential(
            nn.Conv2d(1, 25, kernel_size = (1,5), padding = "valid"),
            nn.Conv2d(25,25, kernel_size = (2,1), padding = "valid"),
            nn.BatchNorm2d(25, eps = 1e-05, momentum =0.1),
            activation_funct,
            nn.MaxPool2d(kernel_size = (1,2)),
            nn.Dropout(p = 0.5),
            nn.Conv2d(25,50, kernel_size= (1,5), padding = "valid"),
            nn.BatchNorm2d(50, eps = 1e-05, momentum =0.1),
            activation_funct,
            nn.MaxPool2d(kernel_size = (1,2)),
            nn.Dropout(p = 0.5),
            nn.Conv2d(50,100, kernel_size= (1,5), padding = "valid"),
            nn.BatchNorm2d(100, eps = 1e-05, momentum =0.1),
            activation_funct,
            nn.MaxPool2d(kernel_size = (1,2)),
            nn.Dropout(p = 0.5),
            nn.Conv2d(100,200, kernel_size= (1,5), padding = "valid"),
            nn.BatchNorm2d(200, eps = 1e-05, momentum =0.1),
            activation_funct,
            nn.MaxPool2d(kernel_size = (1,2)),
            nn.Dropout(p = 0.5)
        )
        self.classify = nn.Sequential(
            nn.Linear(in_features = 200*43, out_features = 2, bias = True)
        )
    def forward(self, x):
        x = self.layers(x)
        x = x.view(-1,200*43)
        x = self.classify(x)
        return x
```

B. Explain the activation function

ReLU: 將輸入 x 經過 max(0,x)的 function，比起 sigmoid 能防止梯度消失，但由於 x<0 時梯度為 0，因此可能會遇到 dying ReLU 的問題。

Leaky ReLU: 若 x > 0，則輸出為 x，若 x < 0 則變為 negative_slope * x，透過 negative slope 來防止 dying ReLU 的問題。

ELU: 若 x > 0，則輸出為 x，若 x < 0 則變為 alpha*(exp(x) - 1)，即使 x < 0 也能微分，以防止 dying ReLU 的問題。

3. Experimental results

A. The highest testing accuracy

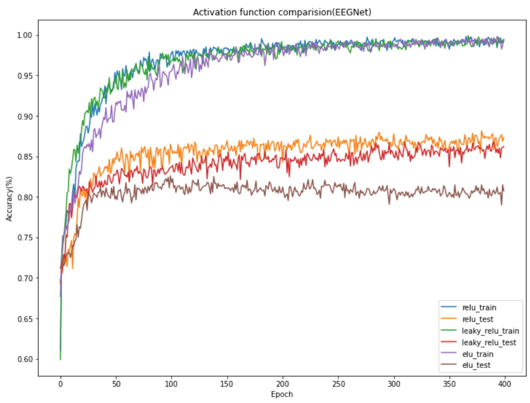
LR = 0.001, Batchsize = 64, Optimizer = Adam, Loss = CrossEntropy

	ReLU	Leaky ReLU	ELU
EEGNet	**88.15%**	**86.76%**	**82.59%**
DeepConvNet	82.04%	82.41%	81.57%

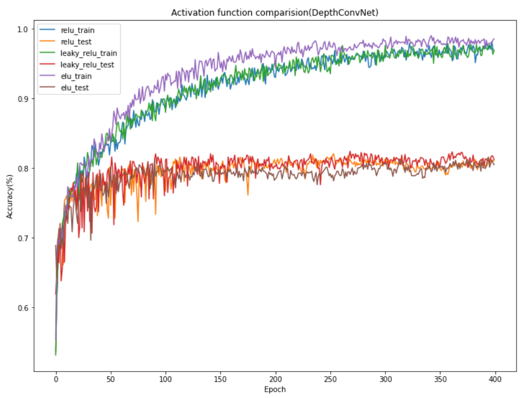
```
Training EEG_ELU model...
Best Accuracy:  0.825925925925926  on epoch  99
Training EEG_ReLU model...
Best Accuracy:  0.8814814814814815  on epoch  379
Training EEG_Leaky_ReLU model...
Best Accuracy:  0.8675925925925926  on epoch  310
Training DeepConvNet_ELU model...
Best Accuracy:  0.8157407407407408  on epoch  394
Training DeepConvNet_ReLU model...
Best Accuracy:  0.8203703703703704  on epoch  253
Training DeepConvNet_LeakyReLU model...
Best Accuracy:  0.8240740740740741  on epoch  268
```

B. Comparison figures

EEGNet



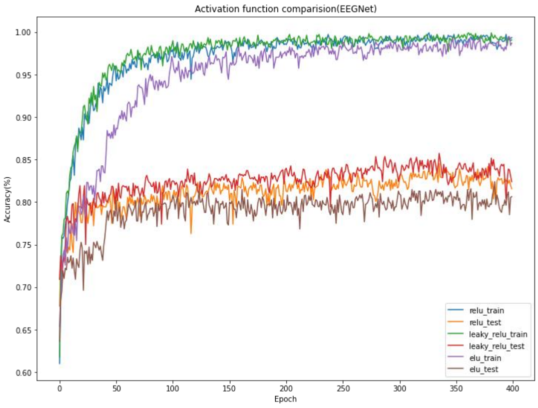
DeepConvNet

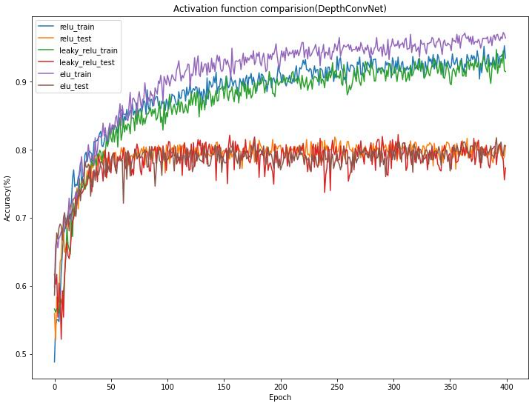


4. Discussion

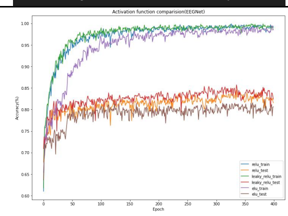
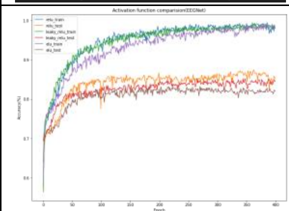
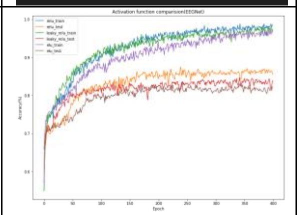
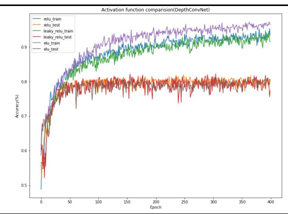
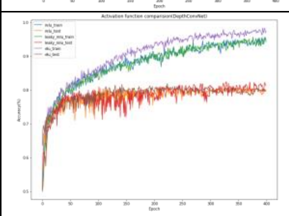
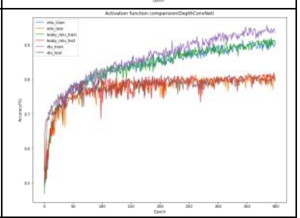
最初使用簡報中的參數 Batch Size = 64, Learning Rate = 1e-2，結果如下:

```
Training EEG_ELU model...
Best Accuracy:  0.8240740740740741  on epoch  384
Training EEG_ReLU model...
Best Accuracy:  0.8453703703703703  on epoch  335
Training EEG_Leaky_ReLU model...
Best Accuracy:  0.8574074074074074  on epoch  335
Training DeepConvNet_ELU model...
Best Accuracy:  0.8175925925925925  on epoch  391
Training DeepConvNet_ReLU model...
Best Accuracy:  0.8185185185185185  on epoch  248
Training DeepConvNet_LeakyReLU model...
Best Accuracy:  0.8222222222222222  on epoch  304
```





為了提升 Accuracy，我調低了 Learning Rate 以讓模型訓練更加穩定，並嘗試不同的 Batch size 觀察訓練結果。再將學習率降為 1e-3 後，EEGNet 搭配 ReLU 的組合能穩定的突破 87%的 Accuracy，對於 EEGNet 搭配 Leaky ReLU 的模型也能有微幅的進步，但在 DeepConvNet 以及 EEGNet 搭配 ELU 的模型表現則沒有太明顯的幫助。以下為 lr = 1e-3 時各 Batch size 之訓練結果:

	Batchsize = 64	Batchsize = 256	Batchsize = 512
Accuracy	Training EEG_ELU model... Best Accuracy: 0.8240740740740741 on epoch 384 Training EEG_ReLU model... Best Accuracy: 0.8453703703703703 on epoch 335 Training EEG_Leaky_ReLU model... Best Accuracy: 0.8574074074074074 on epoch 335 Training DeepConvNet_ELU model... Best Accuracy: 0.8175925925925925 on epoch 391 Training DeepConvNet_ReLU model... Best Accuracy: 0.8185185185185185 on epoch 248 Training DeepConvNet_LeakyReLU model... Best Accuracy: 0.8222222222222222 on epoch 304	Training EEG_ELU model... Best Accuracy: 0.8324074074074074 on epoch 270 Training EEG_ReLU model... Best Accuracy: 0.8740740740740741 on epoch 362 Training EEG_Leaky_ReLU model... Best Accuracy: 0.8527777777777777 on epoch 265 Training DeepConvNet_ELU model... Best Accuracy: 0.8185185185185185 on epoch 223 Training DeepConvNet_ReLU model... Best Accuracy: 0.8101851851851852 on epoch 369 Training DeepConvNet_LeakyReLU model... Best Accuracy: 0.8240740740740741 on epoch 328	Training EEG_ELU model... Best Accuracy: 0.8314814814814815 on epoch 241 Training EEG_ReLU model... Best Accuracy: 0.8740740740740741 on epoch 238 Training EEG_Leaky_ReLU model... Best Accuracy: 0.8462962962962963 on epoch 341 Training DeepConvNet_ELU model... Best Accuracy: 0.8138888888888889 on epoch 364 Training DeepConvNet_ReLU model... Best Accuracy: 0.8148148148148148 on epoch 386 Training DeepConvNet_LeakyReLU model... Best Accuracy: 0.8194444444444444 on epoch 397
EEGNet			
DeepConvNet			

另外，從 Comparison Figure 也可發現使用不同 Activation Function 對於 EEGNet 的表現影響較為明顯，DeepConvNet 的表現則差異不大。使用較大的 Batch size 訓練時，模型 Training Accuracy 上升的趨勢也會較為平緩。

5. Extra

A. Implement another classification model

RNN

由於 EEG Data 中應有時間序列上的關聯，上週正好學習了 RNN 的模型，因此嘗試使用 PyTorch 建立 RNN 模型來進行 classification，模型架構如下:

```
class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers):
        super().__init__()
        self.hidden_size = hidden_size
        self.num_layers = num_layers
        self.rnn = nn.RNN(input_size, hidden_size, num_layers, batch_first=True)
        self.classify = nn.Linear(hidden_size, 2)

    def forward(self, x):
        h0 = torch.zeros(self.num_layers, x.size(0), self.hidden_size).to(x.device)
        x, _ = self.rnn(x, h0)
        x = self.classify(x[:, -1, :])
        return x
```

然而，或許是模型不合適，抑或是因為對於模型特性不熟悉，也可能與資料的處理方式有關，在多次嘗試後模型表現仍然不理想，以下為訓練結果:

```
Model: RNN
Optimizer: Adam, LR: 0.001, Batchsize: 256
Epoch 10: train accuracy = 0.5425925925925926, test accuracy = 0.5231481481481481
Epoch 20: train accuracy = 0.5879629629629629, test accuracy = 0.5314814814814814
Epoch 40: train accuracy = 0.6537037037037037, test accuracy = 0.4722222222222222
Epoch 50: train accuracy = 0.7194444444444444, test accuracy = 0.4842592592592593
Epoch 60: train accuracy = 0.7712962962962963, test accuracy = 0.4962962962962963
Epoch 70: train accuracy = 0.8379629629629629, test accuracy = 0.5203703703703704
Epoch 80: train accuracy = 0.9324074074074075, test accuracy = 0.5138888888888888
Epoch 90: train accuracy = 0.9592592592592593, test accuracy = 0.5148148148148148
Epoch 100: train accuracy = 0.9962962962962963, test accuracy = 0.525
Best test accuracy 0.5342592592592592 in epoch 21.
```