DL_Lab3 Convolutional Network
309553012 黃建洲

1. Introduction:

這次作業讓我們初步了解pytorch，利用pytorch來寫出有名的EEG以及deep convolution架構。並且讓我們自行修改其中的部分內容來觀察其差異。

2. Experiment Setup:

(1) Detail of the models:

a. EEGNet:

```python
class EEGNet(nn.Module):
    def __init__(self, activation, 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=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(),
            nn.AvgPool2d(kernel_size=(1,4), stride=(1,4), padding=0),
            nn.Dropout(dropout)

        )

        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(),
            nn.AvgPool2d(kernel_size=(1,8), stride=(1,8), padding=0),
            nn.Dropout(dropout)


        )
        self.flatten_size = 736
        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, self.flatten_size)
        x = self.classify(x)
        return x
```

由於PDF之題意非常清楚，EEGNet的部分基本上全部按照PDF給的圖表進行，僅改動hyper parameter。

b. DeepConvNet:

```python
class deepConvNet(nn.Module):
    def __init__(self, activation, dropout=0.5):
        super(deepConvNet, self).__init__()
        self.conv0 = nn.Sequential(
            nn.Conv2d(1, 25, kernel_size=(1,5), stride=(1,1), bias=False)
        )
        self.conv1 = nn.Sequential(
            nn.Conv2d(25, 25, kernel_size=(2,1), stride=(1,1), bias=False),
            nn.BatchNorm2d(25, eps=1e-05, momentum=0.1),
            activation(),
            nn.MaxPool2d(kernel_size=(1,2)),
            nn.Dropout(dropout)
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(25, 50, kernel_size=(1,5), stride=(1,1), bias=False),
            nn.BatchNorm2d(50, eps=1e-05, momentum=0.1),
            activation(),
            nn.MaxPool2d(kernel_size=(1,2)),
            nn.Dropout(dropout)
        )
        self.conv3 = nn.Sequential(
            nn.Conv2d(50, 100, kernel_size=(1,5), stride=(1,1), bias=False),
            nn.BatchNorm2d(100, eps=1e-05, momentum=0.1),
            activation(),
            nn.MaxPool2d(kernel_size=(1,2)),
            nn.Dropout(dropout)
        )
        self.conv4 = nn.Sequential(
            nn.Conv2d(100, 200, kernel_size=(1,5), stride=(1,1), bias=False),
            nn.BatchNorm2d(200, eps=1e-05, momentum=0.1),
            activation(),
            nn.MaxPool2d(kernel_size=(1,2), stride = (1,2)),
            nn.Dropout(dropout)
        )
```

```python
        #flatten size = total input size / batch size = 550400 / 64 = 8600
        self.flatten_size = 8600
        self.fc1 = nn.Sequential(
            nn.Dropout(dropout),
            nn.Linear(self.flatten_size, 100),
            activation()

        )
        self.fc2 = nn.Sequential(
            nn.Dropout(dropout),
            nn.Linear(100, 50),
            activation()
        )
        self.fc3 = nn.Sequential(
            nn.Dropout(dropout),
            nn.Linear(50, 2)
        )
    def forward(self, x):

        x = self.conv0(x)
        x = self.conv1(x)
        x = self.conv2(x)
        x = self.conv3(x)
        x = self.conv4(x)
        x = x.view(-1, self.flatten_size)
        x = self.fc1(x)
        x = self.fc2(x)
        x = self.fc3(x)
        return x
```

Deep Convolution Network的部分在前面幾個卷積層我沒有做改動，但完全不改動的話調整超參數也效果不佳，因此我多加了兩層含有activation和dropout的fc layer。

(2) Explain the activation function

a. ReLU:

ReLU = max(0,x)

優點:

(1)在positive的部分解決了gradient descent的問題

(2)計算速度快

(3)收斂速度快

缺點:

(1) 中心不為0

(2) 有可能會出現dead ReLU problem(有些神經元永遠無法被activation)

b. LeakyReLU:

Leaky ReLU = max(ax,x), a = 0.01

優點: 與ReLU相同，且不會有dead ReLU problem

但不一定在每個case都比ReLU好

c. ELU:

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$$

優點: 也都與ReLU相同，不會有dead ReLU problem，且中心接近於0
但也一樣，無法證明在每個case都會比ReLU好

3. Experiment results:

(1)The highest testing accuracy

(a)EEG:

```
train accuracy: 98.51851851851852, epochs = 299, Loss: 0.0031978923361748457
test accuracy: 82.22222222222223, epochs = 299, max_accuracy = 83.61111111111111
```

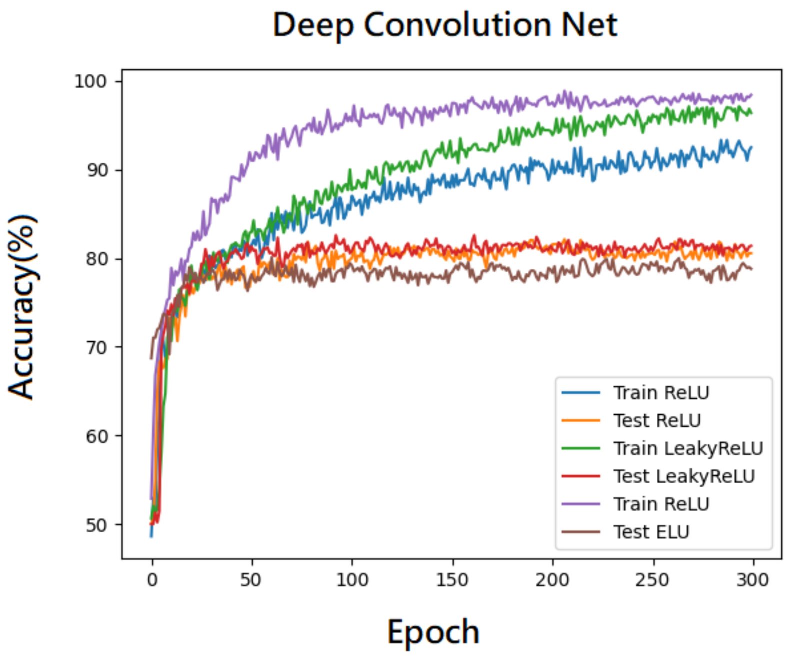
(b)Deep Conv:

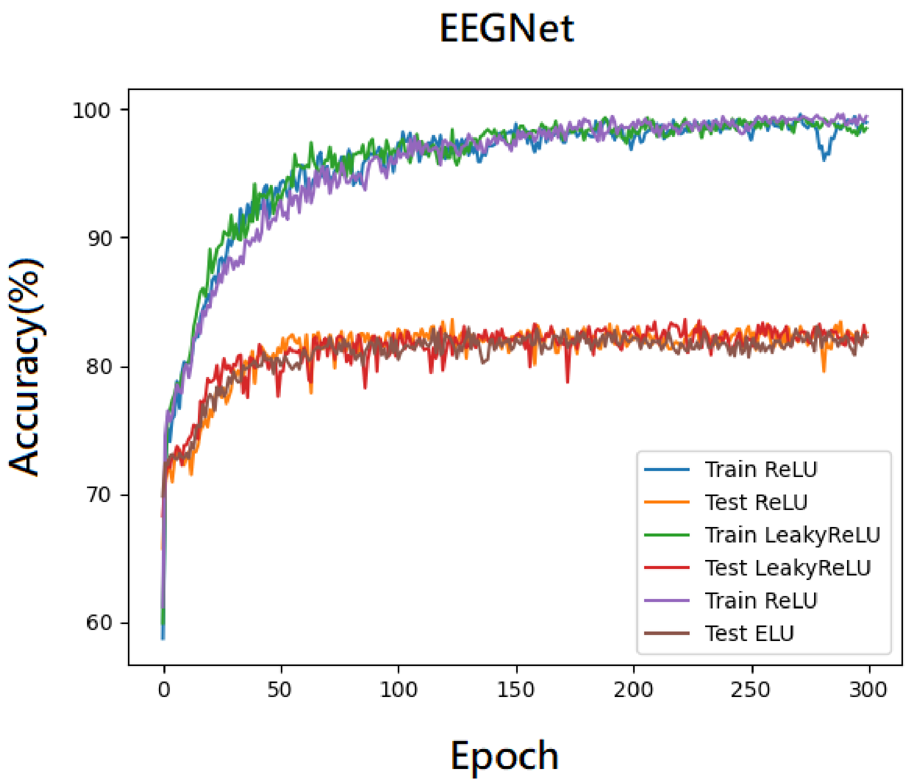
```
train accuracy: 96.38888888888889, epochs = 299, Loss: 0.01443187054246664
test accuracy: 81.38888888888889, epochs = 299, max_accuracy = 82.5925925925926
```

(c) Statistic Chart

epochs = 300 lr = 0.001 optimizer = Adam	ELU	LeakyReLU	ReLU
EEGNet (batch size = 64)	83.05%	83.61%	83.61%
Deep Conv Net (batch size = 32)	80.09%	82.59%	82.12%

(2) Comparison





4. Discussion:

EEGNet由於是針對這個dataset所設計的網路，因此在寫這次作業的時候就能清楚感覺到，在建立model之後僅僅只需要微調一些超參數就可以讓EEG model擁有不錯的準確率。但deep convolution model則比較general一點，在針對learning rate, epochs, optimizer以及scheduler進行實驗後，發現並不會有太大的幫助。因此我認為原因在於網路的分類能力不足，故多加了兩層fc layer以增強其分類能力，同時不會造成overfitting。