Lab2: EEG Motor Imagery Classification

313551104 黃暐洺

1. Overview (total 10%)

   此次 LAB 根據 Spatial Component-wise Convolutional Network SCCNet for Motor-Imagery EEG Classification 這篇論文實作了一個個性化 CNN，又根據不同的訓練方式將 dataset 用不同方式切割去訓練測試，同時調整參數並觀察效能差異。

2. Implementation Details (total 20%)

   ▪ Details of training and testing code (10%)

   trainer.py

```python
SD_trainset = Dataloader.MIBCI2aDataset('train', 'SD')
SD_trainloader = DataLoader(SD_trainset, batch_size=64,shuffle=True)

model1 = SCCNet.SCCNet().to(device)
n_epochs = 3000
# loss_history = []
optimizer = optim.Adam(model1.parameters(), lr=1e-4, weight_decay=1e-4)
criterion = nn.CrossEntropyLoss()
model1.train()
lowest_loss = 2
for epoch in range(1, n_epochs+1):
    for batch_idx, (features, labels) in enumerate(SD_trainloader):
        features, labels = features.to(device), labels.to(device)
        optimizer.zero_grad()
        output = model1(features.to(torch.float32))
        loss = criterion(output, labels.to(torch.int64))
        if loss < lowest_loss:
            lowest_loss = loss
            torch.save(model1.state_dict(), 'SD_best.pth')
            print("loss update : ", loss.cpu().detach().numpy())
        # loss_history.append(loss.cpu().detach().numpy())
        loss.backward()
        optimizer.step()
    if not epoch%1000:
        print("now in epoch : ", epoch)
torch.save(model1.state_dict(), 'SD.pth')
```

tester.py

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
print(f"Using device: {device}")
SD_testset = Dataloader.MIBCI2aDataset('test', 'LOSO')
SD_testloader = DataLoader(SD_testset, batch_size=64, shuffle=False)

model = model.SCCNet.SCCNet().to(device)
model.load_state_dict(torch.load('FT.pth'))
model.eval()

criterion = nn.CrossEntropyLoss()
correct = 0
total = 0
total_loss = 0

with torch.no_grad():
    for batch_idx, (features, labels) in enumerate(SD_testloader):
        features, labels = features.to(device), labels.to(device)
        outputs = model(features.to(torch.float32))
        loss = criterion(outputs, labels.to(torch.int64))
        total_loss += loss.item()
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

avg_loss = total_loss / len(SD_testloader)
accuracy = 100 * correct / total

print(f'Test Loss: {avg_loss:.4f}')
print(f'Test Accuracy: {accuracy:.2f}%')
```

Trainer 使用自定義的 MIBCI2aDataset 加載訓練數據，會根據不同的實驗需求載入不同資料集，並使用 DataLoader 建立 Batch，啟用隨機洗牌。

根據給定的 epoch 做訓練，optimizer 用 Adam，LR 為 1e-4，並使用 L2 regularization，使用 CrossEntropy 作為 Loss Function。

每個批次的訓練步驟：清零梯度→forward →計算 Loss→反向傳播→計算梯度→更新模型參數。

Tester 使用自定義的 MIBCI2aDataset 加載測試數據，會根據不同的實驗需求載入不同資料集，並使用 DataLoader 建立 Batch，不洗牌。

接著加載預訓練的模型權重，使用 CrossEntropy，初始化正確預測數、總樣本數和總損失，使用 torch.no_grad()禁用梯度計算，遍歷測試數

據集的每個批次。每個批次的測試步驟：forward→計算 Loss→累加 Total Loss→得到預測結果（最大概率的 class）→計算正確預測的數量。

▪ Details of the SCCNet (10%)

```python
class SquareLayer(nn.Module):
    def __init__(self):
        super(SquareLayer, self).__init__()

    def forward(self, x):
        return x ** 2

    def back(self, x):
        return 2*x
```

```python
class SCCNet(nn.Module):
    def __init__(self, numClasses=4, timeSample=438, Nu=44, C=22, Nc=20, Nt=8, dropoutRate=0.5):
        super(SCCNet, self).__init__()
        self.conv1 = nn.Conv2d(1, Nu, (C, Nt), padding=0)
        self.bn1 = nn.BatchNorm2d(Nu)
        self.conv2 = nn.Conv2d(1, Nc, (Nu, 12), padding=(0,6))
        self.bn2 = nn.BatchNorm2d(Nc)
        self.square = SquareLayer()
        self.dropout = nn.Dropout(dropoutRate)
        self.pool = nn.AvgPool2d((1, 62), stride=(1, 12))
        PooledSize = (timeSample-Nt+1+1-62)//12+1 #Nt reduce, padding increase
        self.fc = nn.Linear(Nc*PooledSize, numClasses)

    def forward(self, x):
        x = x.unsqueeze(1)
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.dropout(x)
        x = x.permute(0, 2, 1, 3)
        x = self.conv2(x)
        x = self.bn2(x)
        x = self.square(x)
        x = self.dropout(x)
        x = x.permute(0, 2, 1, 3)
        x = self.pool(x)
        x = x.flatten(1)
        x = self.fc(x)
        return x
```

SCCNet 主要包含兩個卷積層、一個自定義的平方層、pooling layer 和 full connected layer，各參數和 paper 相同 ：

- numClasses: 分類的 class 數，默認 4
- timeSample: 時間樣本數，默認 438

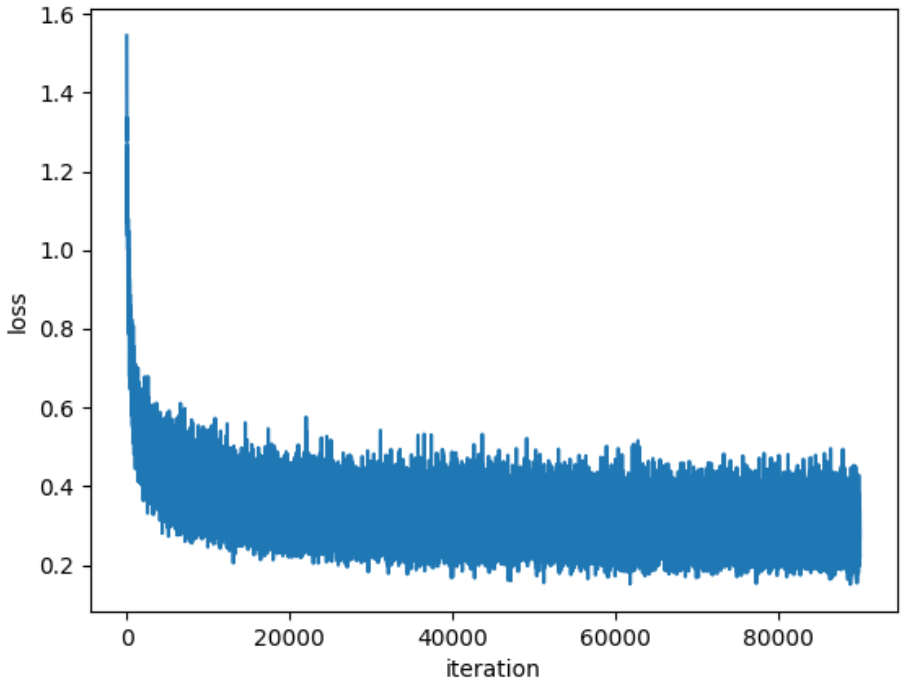
- Nu: 第一個卷積層的輸出 channel 數，默認 22
- C: 輸入 channel 數，默認 22
- Nc: 第二個卷積層的輸出 channel 數，默認 20
- Nt: 第一個 kernel width，默認 16
- dropoutRate: Dropout 率，默認 0.5

另外，卷積層之後會使用 dropout 跟 batch normalization 防止 overfitting 跟加速收斂，也用了 unsqueeze 跟 permute 好讓 data 滿足模型要求。FC layer 之前會有一個 flatten layer，將特徵映射到 numClasses 輸出。

3. Analyze on the experiment results (total 50%)

▪ Discover during the training process (20%)

通常一開始 loss 會下降得很快(epoch<1000)，拿這些時期的 model 做 testing 也會發現準確率快速上升。不過到了 50~60%時速度就會慢下來，在不斷減少 loss 的時候，準確率則不降反升，很可能是 overfitting。到 epoch>3000 之後，loss 會以極緩慢的速度下降，但準確度會上下震盪。



Batch size 可以讓模型訓練得更快，加速收斂，不過對提高準確率沒有明顯影響。

▪ Comparison between the three training methods (30%)

LOSO 準確率最低，主要是因為模型測試所使用資料的受試者未曾出現在訓練資料集，照著 paper 提供的 model 準確率最高能達到 62%。

SD 準確率介於 LOSO、LOSO+FT 中間，訓練集與測試集有高度相關，但涉及不同受測者，個體差異使得 Loss 降不下去，準確率沒辦法提高，我測試下來只能達到 62%。

LOSO+FT 準確率為三者最佳，因為有針對測試集受試者作微調，微調用資料集足夠小，可以快入收斂，Loss 輕鬆達到小數點後五位，測試下來準確最高達到 77%，要達到 80%以上還是很有挑戰。

4. Discussion (total 20%)

▪ What is the reason to make the task hard to achieve high accuracy? (10%)

EEG 信號在不同受試者身上也許有極大不同，也有可能因心情、健康指數、雜訊等因素使訊號失真，如果再提供更大量的數據也許可以有更好表現，也可以說是數據量不足、樣本不夠。

從模型的角度來看可能是 SCCNet 可能不夠複雜，無法捕捉 EEG 信號的全部相關特徵，模型可能在 train 的過程 overfitting，或者卡在 local minimum，可能除了 dropout、batch normalization 和 L2 regularization 需要更多 domain knowledge，但不排除 EEG 訊號本身就是艱難的任務。

▪ What can you do to improve the accuracy of this task? (10%)

應增加模型的複雜程度，可以是增加更多 layer、使用不同 activation function、使用 grid search 搜尋最佳參數、加入注意力機制，抑或是從 dataset 下手，增加樣本數或是做前置處理、讓模型針對單一受試者做優化。