1. Introduction

   在這個 lab 我實作了 EEGNet 和 DeepConvNet，使用 acitvation functions 像是 Relu, Leaky Relu, ELU，總共六種神經網路來預測 Non-stationary BCI data 到兩種 classes。

2. Experiment set up

   A. The detail of your model

   - EEGNet

```
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): ELU(alpha=1.0)
    (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): ELU(alpha=1.0)
    (3): AvgPool2d(kernel_size=(1, 8), stride=(1, 8), padding=0)
    (4): Dropout(p=0.25, inplace=False)
  )
  (classify): Sequential(
    (0): Linear(in_features=736, out_features=2, bias=True)
  )
)
```

     在 classify 之前使用 view 將維度壓成(batch size, 其他)

   - DeepConvNet

```
DeepConvNet(
  (conv1): Conv2d(1, 25, kernel_size=(1, 5), stride=(1, 1))
  (conv2): Conv2d(25, 25, kernel_size=(2, 1), stride=(1, 1))
  (BN1): BatchNorm2d(25, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (activation1): ELU(alpha=1.0)
  (maxpool1): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
  (drop1): Dropout(p=0.5, inplace=False)
  (conv3): Conv2d(25, 50, kernel_size=(1, 5), stride=(1, 1))
  (BN2): BatchNorm2d(50, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (activation2): ELU(alpha=1.0)
  (maxpool2): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
  (drop2): Dropout(p=0.5, inplace=False)
  (conv4): Conv2d(50, 100, kernel_size=(1, 5), stride=(1, 1))
  (BN3): BatchNorm2d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (activation3): ELU(alpha=1.0)
  (maxpool3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
  (drop3): Dropout(p=0.5, inplace=False)
  (conv5): Conv2d(100, 200, kernel_size=(1, 5), stride=(1, 1))
  (BN4): BatchNorm2d(200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (activation4): ELU(alpha=1.0)
  (maxpool4): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
  (drop4): Dropout(p=0.5, inplace=False)
  (dense): Linear(in_features=8600, out_features=2, bias=True)
)
```

     在 dense 之前使用 view 將維度壓成(batch size, 其他)

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

ReLU：

$$\mathrm{ReLU}(x) = \max(0, x)$$

只有 x 大於 0 才有等於 1 的 gradient，否則 gradient 等於 0

Leaky ReLU:

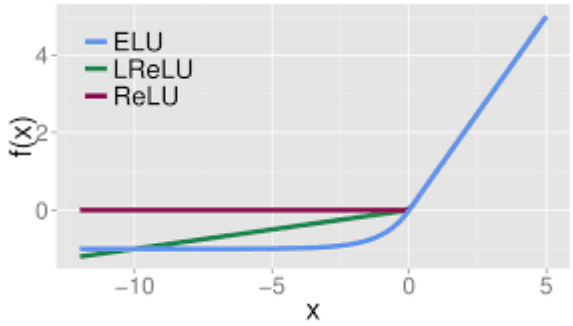
$$\mathrm{LeakyRELU}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \text{negative\_slope} \times x, & \text{otherwise} \end{cases}$$

x 小於 0 也會有一個 negative_slope 的 gradient

ELU:

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

給定一些負值也有 gradient，越負 gradient 越接近於 0



3. Experimental results
   A. A highest testing accuracy
      - Screenshot with two models

      EEGNet – ELU

```
Epoch: 286 | train accuracy: 0.9907 | test accuracy: 0.82
Epoch: 287 | train accuracy: 0.9907 | test accuracy: 0.82
Epoch: 288 | train accuracy: 0.9898 | test accuracy: 0.81
Epoch: 289 | train accuracy: 0.9861 | test accuracy: 0.81
Epoch: 290 | train accuracy: 0.9907 | test accuracy: 0.82
Epoch: 291 | train accuracy: 0.9889 | test accuracy: 0.83
Epoch: 292 | train accuracy: 0.9898 | test accuracy: 0.83
Epoch: 293 | train accuracy: 0.9852 | test accuracy: 0.82
Epoch: 294 | train accuracy: 0.9843 | test accuracy: 0.82
Epoch: 295 | train accuracy: 0.9880 | test accuracy: 0.82
Epoch: 296 | train accuracy: 0.9880 | test accuracy: 0.82
Epoch: 297 | train accuracy: 0.9889 | test accuracy: 0.82
Epoch: 298 | train accuracy: 0.9852 | test accuracy: 0.81
Epoch: 299 | train accuracy: 0.9769 | test accuracy: 0.80
Epoch: 300 | train accuracy: 0.9787 | test accuracy: 0.80
```

EGGNet - ReLU

```
Epoch: 286 | train accuracy: 0.9861 | test accuracy: 0.85
Epoch: 287 | train accuracy: 0.9944 | test accuracy: 0.84
Epoch: 288 | train accuracy: 0.9935 | test accuracy: 0.85
Epoch: 289 | train accuracy: 0.9815 | test accuracy: 0.85
Epoch: 290 | train accuracy: 0.9907 | test accuracy: 0.86
Epoch: 291 | train accuracy: 0.9861 | test accuracy: 0.85
Epoch: 292 | train accuracy: 0.9880 | test accuracy: 0.86
Epoch: 293 | train accuracy: 0.9824 | test accuracy: 0.85
Epoch: 294 | train accuracy: 0.9880 | test accuracy: 0.85
Epoch: 295 | train accuracy: 0.9907 | test accuracy: 0.85
Epoch: 296 | train accuracy: 0.9843 | test accuracy: 0.83
Epoch: 297 | train accuracy: 0.9852 | test accuracy: 0.85
Epoch: 298 | train accuracy: 0.9917 | test accuracy: 0.86
Epoch: 299 | train accuracy: 0.9787 | test accuracy: 0.84
Epoch: 300 | train accuracy: 0.9861 | test accuracy: 0.87
```

EGGNet – LeakyReLU

```
Epoch: 286 | train accuracy: 0.9833 | test accuracy: 0.85
Epoch: 287 | train accuracy: 0.9944 | test accuracy: 0.84
Epoch: 288 | train accuracy: 0.9907 | test accuracy: 0.85
Epoch: 289 | train accuracy: 0.9870 | test accuracy: 0.86
Epoch: 290 | train accuracy: 0.9898 | test accuracy: 0.86
Epoch: 291 | train accuracy: 0.9861 | test accuracy: 0.85
Epoch: 292 | train accuracy: 0.9861 | test accuracy: 0.85
Epoch: 293 | train accuracy: 0.9815 | test accuracy: 0.85
Epoch: 294 | train accuracy: 0.9870 | test accuracy: 0.85
Epoch: 295 | train accuracy: 0.9907 | test accuracy: 0.85
Epoch: 296 | train accuracy: 0.9833 | test accuracy: 0.83
Epoch: 297 | train accuracy: 0.9843 | test accuracy: 0.85
Epoch: 298 | train accuracy: 0.9880 | test accuracy: 0.86
Epoch: 299 | train accuracy: 0.9806 | test accuracy: 0.84
Epoch: 300 | train accuracy: 0.9861 | test accuracy: 0.86
```

DeepConvNet – ELU

```
Epoch: 286 | train accuracy: 0.9481 | test accuracy: 0.77
Epoch: 287 | train accuracy: 0.9500 | test accuracy: 0.76
Epoch: 288 | train accuracy: 0.9324 | test accuracy: 0.72
Epoch: 289 | train accuracy: 0.9602 | test accuracy: 0.77
Epoch: 290 | train accuracy: 0.9361 | test accuracy: 0.76
Epoch: 291 | train accuracy: 0.9398 | test accuracy: 0.75
Epoch: 292 | train accuracy: 0.9463 | test accuracy: 0.77
Epoch: 293 | train accuracy: 0.9435 | test accuracy: 0.75
Epoch: 294 | train accuracy: 0.9528 | test accuracy: 0.73
Epoch: 295 | train accuracy: 0.9380 | test accuracy: 0.76
Epoch: 296 | train accuracy: 0.9463 | test accuracy: 0.75
Epoch: 297 | train accuracy: 0.9426 | test accuracy: 0.73
Epoch: 298 | train accuracy: 0.9426 | test accuracy: 0.75
Epoch: 299 | train accuracy: 0.9491 | test accuracy: 0.76
Epoch: 300 | train accuracy: 0.9435 | test accuracy: 0.75
```

DeepConvNet – ReLU

```
Epoch: 286 | train accuracy: 0.9204 | test accuracy: 0.76
Epoch: 287 | train accuracy: 0.9185 | test accuracy: 0.76
Epoch: 288 | train accuracy: 0.9148 | test accuracy: 0.75
Epoch: 289 | train accuracy: 0.9389 | test accuracy: 0.75
Epoch: 290 | train accuracy: 0.9306 | test accuracy: 0.76
Epoch: 291 | train accuracy: 0.9204 | test accuracy: 0.75
Epoch: 292 | train accuracy: 0.9213 | test accuracy: 0.75
Epoch: 293 | train accuracy: 0.9176 | test accuracy: 0.74
Epoch: 294 | train accuracy: 0.9204 | test accuracy: 0.77
Epoch: 295 | train accuracy: 0.9241 | test accuracy: 0.76
Epoch: 296 | train accuracy: 0.9241 | test accuracy: 0.76
Epoch: 297 | train accuracy: 0.9222 | test accuracy: 0.78
Epoch: 298 | train accuracy: 0.9287 | test accuracy: 0.75
Epoch: 299 | train accuracy: 0.9287 | test accuracy: 0.75
Epoch: 300 | train accuracy: 0.9269 | test accuracy: 0.78
```

DeepConvNet – LeakyReLU

```
Epoch: 286 | train accuracy: 0.9306 | test accuracy: 0.76
Epoch: 287 | train accuracy: 0.9241 | test accuracy: 0.75
Epoch: 288 | train accuracy: 0.9231 | test accuracy: 0.74
Epoch: 289 | train accuracy: 0.9213 | test accuracy: 0.74
Epoch: 290 | train accuracy: 0.9231 | test accuracy: 0.75
Epoch: 291 | train accuracy: 0.9231 | test accuracy: 0.75
Epoch: 292 | train accuracy: 0.9287 | test accuracy: 0.75
Epoch: 293 | train accuracy: 0.9315 | test accuracy: 0.74
Epoch: 294 | train accuracy: 0.9120 | test accuracy: 0.74
Epoch: 295 | train accuracy: 0.9213 | test accuracy: 0.75
Epoch: 296 | train accuracy: 0.9296 | test accuracy: 0.76
Epoch: 297 | train accuracy: 0.9287 | test accuracy: 0.75
Epoch: 298 | train accuracy: 0.9287 | test accuracy: 0.76
Epoch: 299 | train accuracy: 0.9102 | test accuracy: 0.75
Epoch: 300 | train accuracy: 0.9139 | test accuracy: 0.76
```

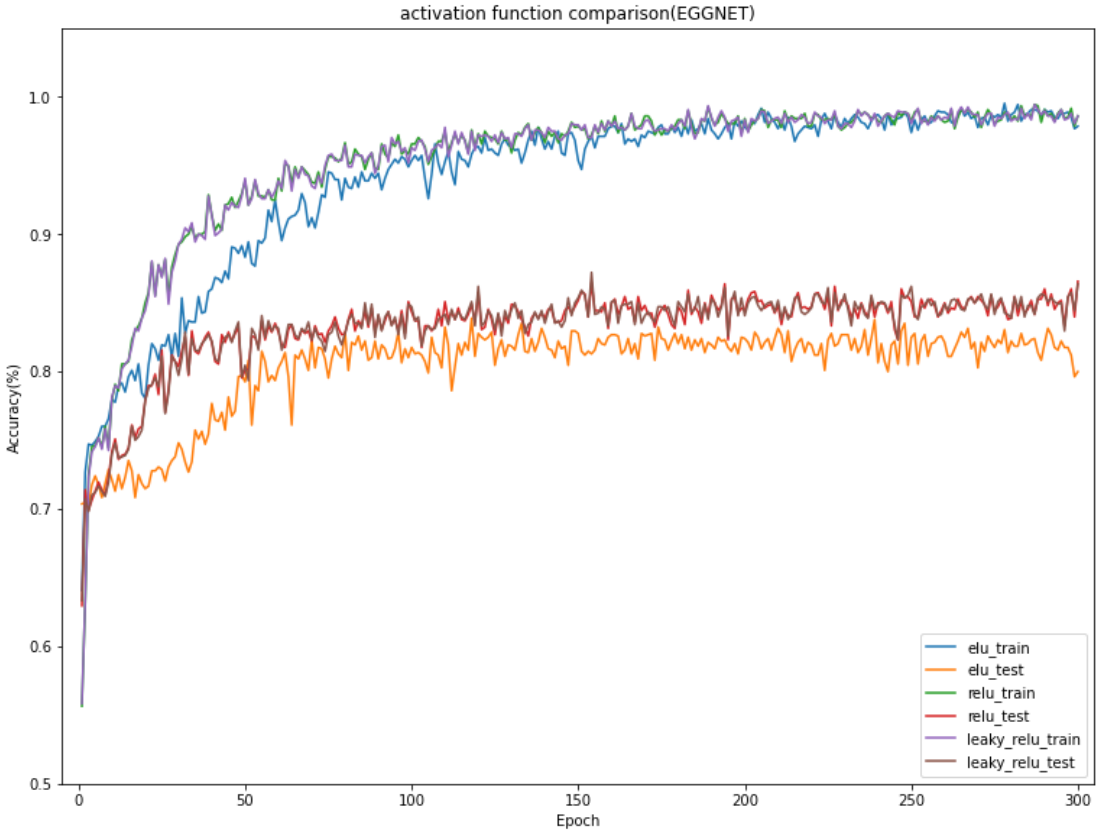
- anything you want to present

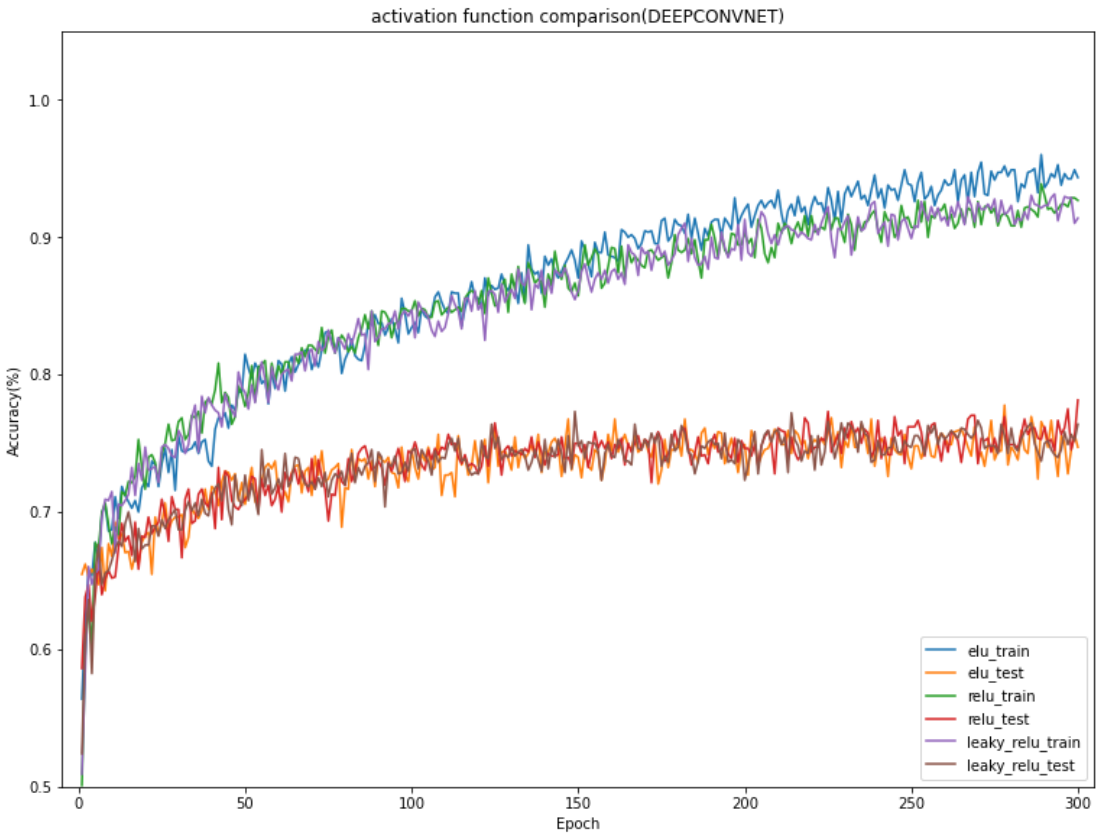
1. 試了不同的 batch size 如 16, 32, 64, 128, 256，最後選擇使用 64。
2. 試了不同的 learning rate 如 0.1, 0.05, 0.01, 0.009~0.005，最後選擇了 0.006。
3. Training 使用的 GPU 是”Tesla V100S-PCIE-32GB”。
4. 由於 DeepConNet 的 train accuracy 還沒有接近 100%，所以給了 1000 個 epoch，結果仍是 ReLU 有最高的 test accuracy，但 test accuracy 仍是只有 0.78 (下面是 DeepConvNet – ReLU，1000 epoch 的結果)。

```
Epoch: 986 | train accuracy: 0.9722 | test accuracy: 0.75
Epoch: 987 | train accuracy: 0.9861 | test accuracy: 0.76
Epoch: 988 | train accuracy: 0.9769 | test accuracy: 0.76
Epoch: 989 | train accuracy: 0.9806 | test accuracy: 0.76
Epoch: 990 | train accuracy: 0.9787 | test accuracy: 0.78

Epoch: 991 | train accuracy: 0.9759 | test accuracy: 0.78
Epoch: 992 | train accuracy: 0.9713 | test accuracy: 0.77
Epoch: 993 | train accuracy: 0.9713 | test accuracy: 0.76
Epoch: 994 | train accuracy: 0.9824 | test accuracy: 0.76
Epoch: 995 | train accuracy: 0.9796 | test accuracy: 0.77
Epoch: 996 | train accuracy: 0.9787 | test accuracy: 0.76
Epoch: 997 | train accuracy: 0.9769 | test accuracy: 0.78
Epoch: 998 | train accuracy: 0.9787 | test accuracy: 0.77
Epoch: 999 | train accuracy: 0.9778 | test accuracy: 0.77
Epoch: 1000 | train accuracy: 0.9759 | test accuracy: 0.78
```

B. Comparison figures





4. Discussion

A. Anything you want to share

為了 reproduce 每一次 training 的結果，寫了如下的 function，只要在每一次 training 之前呼叫該 function 並給予相同數值的 seed，則每一次的訓練結果都會一樣 (但還是 depends on GPU)。除此之外，嘗試了不同的 seed，在 EEGNet 中，ELU 的最終 test accuracy 總是比較低的，但是 ReLU 跟 Leaky ReLU 不分上下，在不同的 seed 中給 300 個 epoch，有時候其中一方的最終 test accuracy 會勝過另一方。

```
def setup_seed(seed):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    torch.manual_seed(seed)
    torch.cuda.manual_seed(seed)
    torch.cuda.manual_seed_all(seed)
    torch.backends.cudnn.benchmark = False
    torch.backends.cudnn.deterministic = True
```