Lab2 EEG classification

Department of Computer Science, NYCU

TA 陳祐平 黃秋蓉

Important Rules

Important Date :

- Report Submission Deadline: 4/13 (Tue) 23:55 p.m.

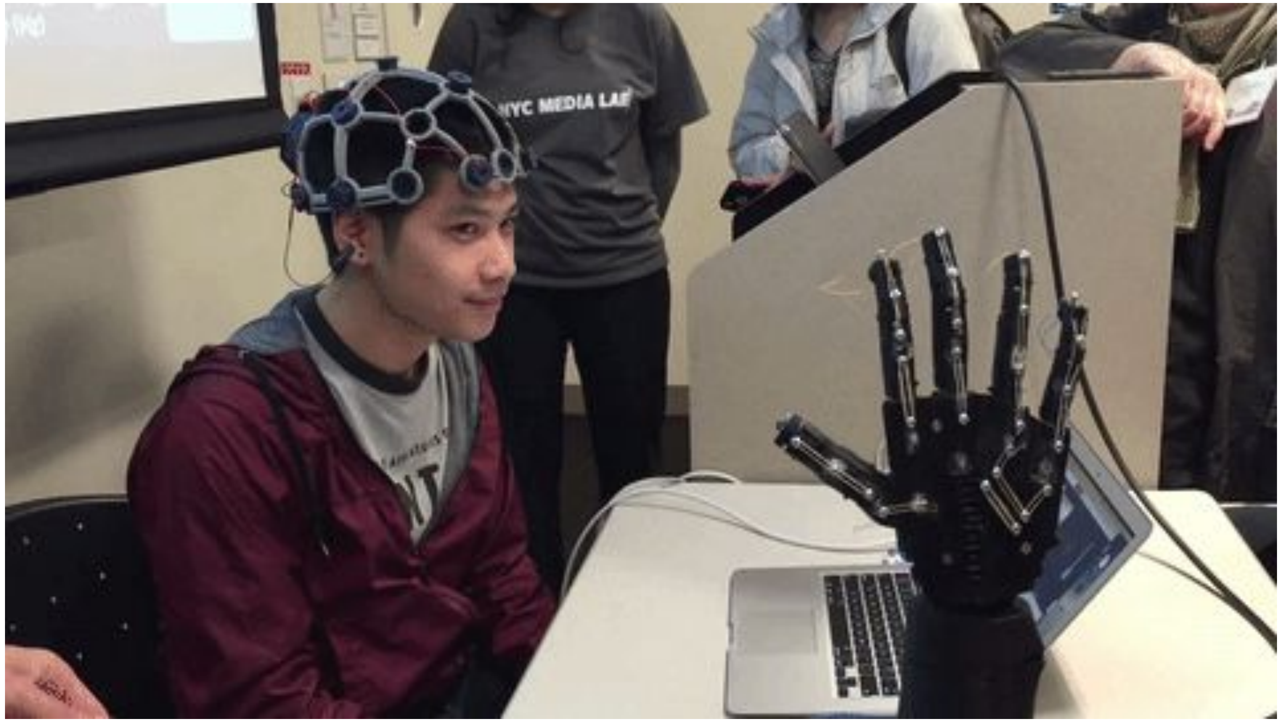
Turn in :

- Experiment Report (.pdf)
- Upload source code (.py) to GitHub

Notice: Name it like「LAB2_your studentID_name.pdf」,
ex:「LAB2_310553043_陳祐平_.pdf」

Lab Objective

- In this lab, you will need to implement simple EEG classification models which are EEGNet[1] with BCI competition dataset.



Requirements

- Implement the EEGNet. And try different alpha value in ELU activation function and different probability in dropout layer.

- In the experiment results, you have to show the highest accuracy and loss of EEGNet with different experiments. (try as more as you can)

- To visualize the accuracy trend, you need to plot each epoch accuracy and loss during training phase and testing phase.

Dataset

- BCI Competition III – IIIb
- [2 classes, 2 bipolar EEG channels]
- *Reference: http://www.bbci.de/competition/iii/desc_IIIb.pdf*

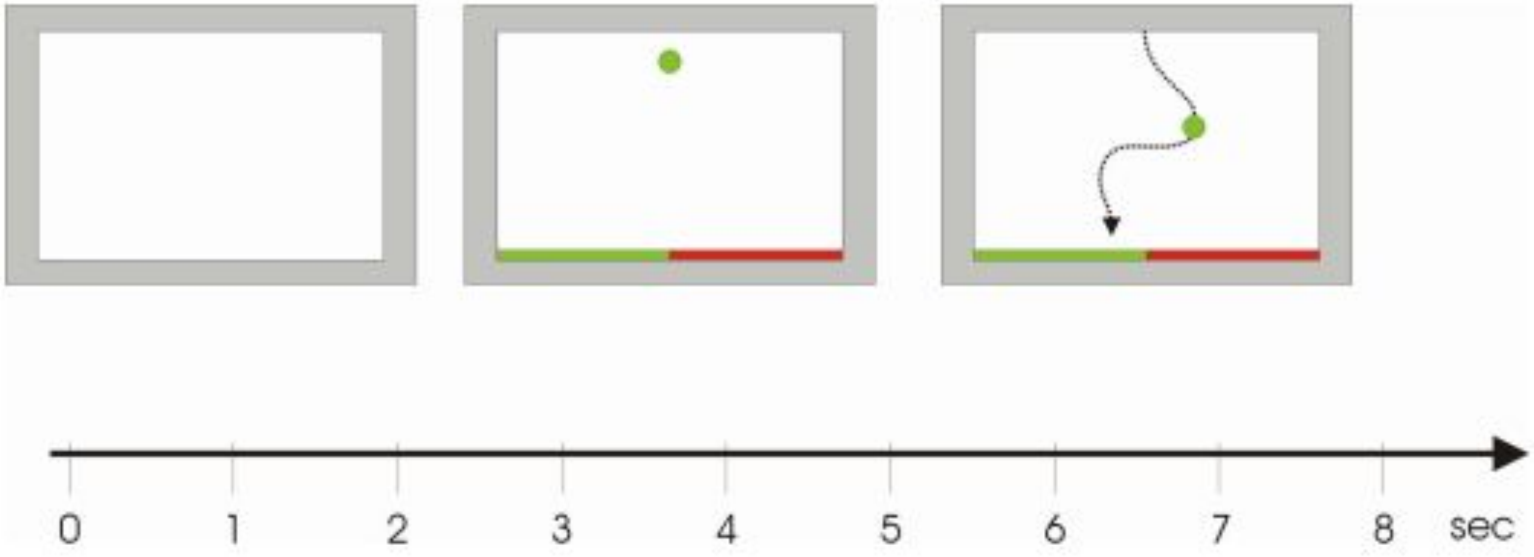


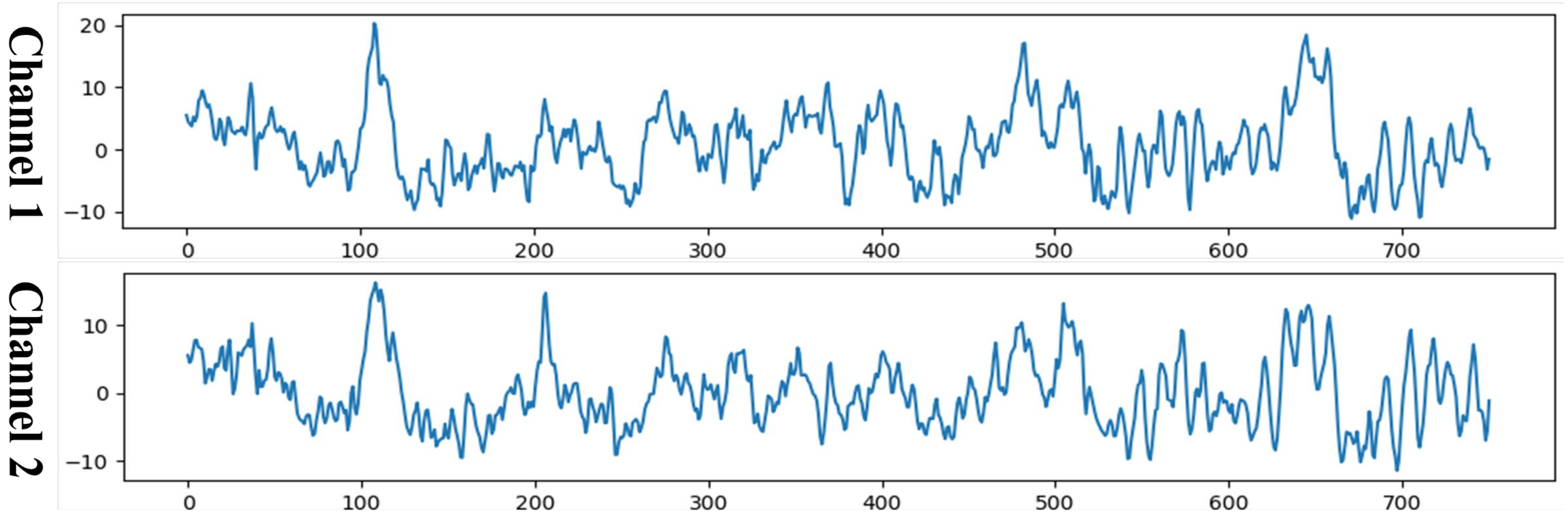
Figure 3: Basket paradigm used for S4 and X11 [3].

Prepare Data

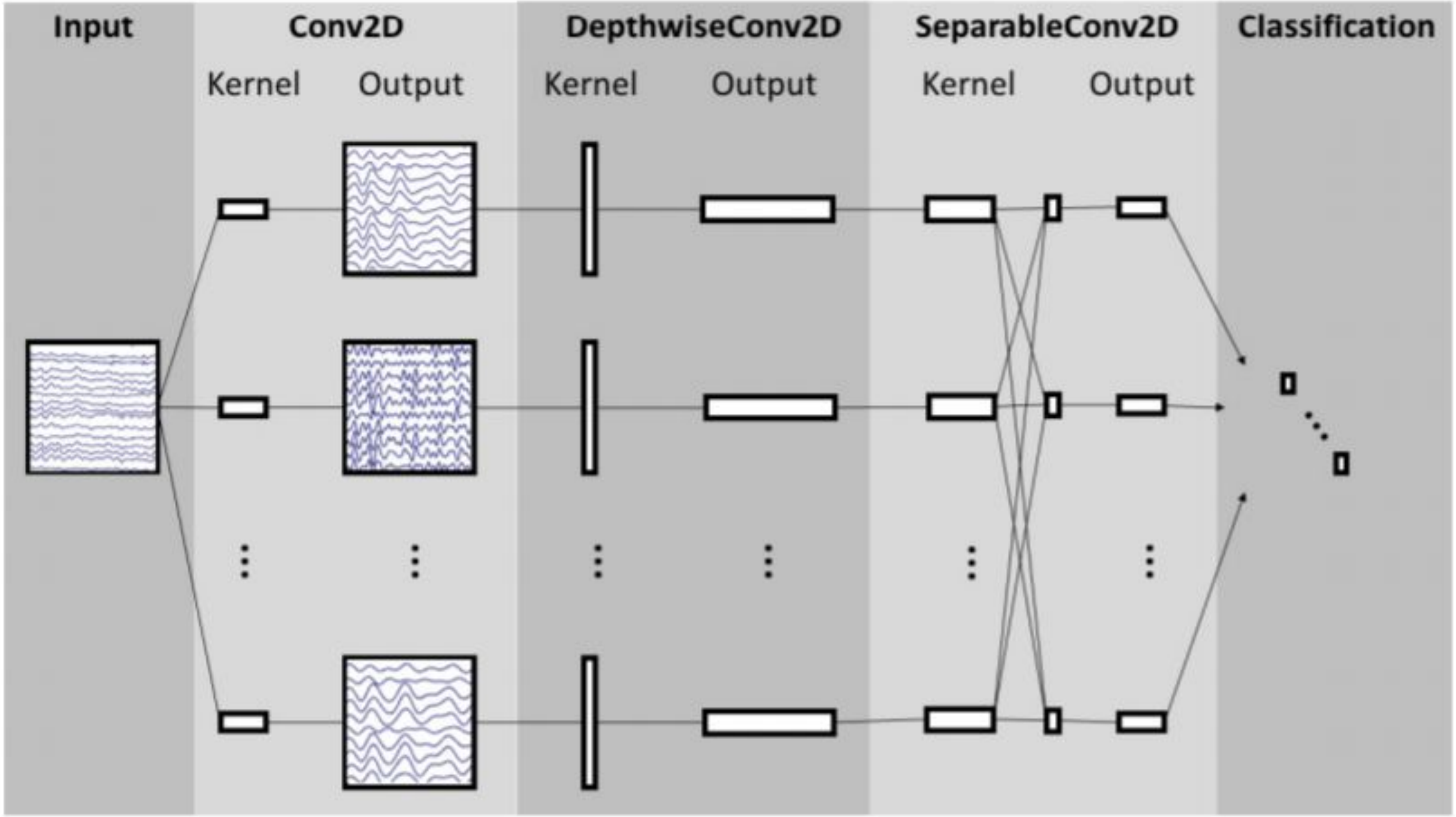
- Training data: S4b_train.npz, X11b_train.npz
- Testing data: S4b_test.npz, X11b_test.npz
- To read the preprocessed data, refer to the "dataloader.py".

B: batch size

- **Input: [B, 1, 2, 750]** **Output: [B, 2]** **Ground truth: [B]**



Create Model - EEGNet



Reference: Depthwise Separable Convolution

https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

Create Model - EEGNet

- EEGNet implementation details

```
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)
  )
  (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)
  )
  (classify): Sequential(
    (0): Linear(in_features=736, out_features=2, bias=True)
  )
)
```

Create Model - Activation Functions

- In the PyTorch framework, it is easy to implement the activation function.

```
nn.LeakyReLU(),
nn.ReLU(),
nn.ELU(),
```

```
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)
  )
  (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)
  )
  (classify): Sequential(
    (0): Linear(in_features=736, out_features=2, bias=True)
  )
)
```

Hyper Parameters

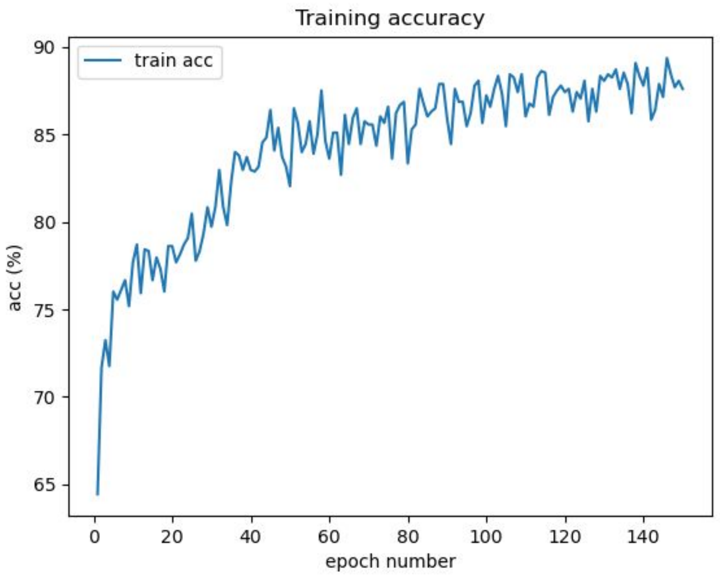
- Batch size= 64
- Learning rate = 1e-2
- Epochs = 150
- Optimizer: Adam
- Loss function: torch.nn.CrossEntropyLoss()

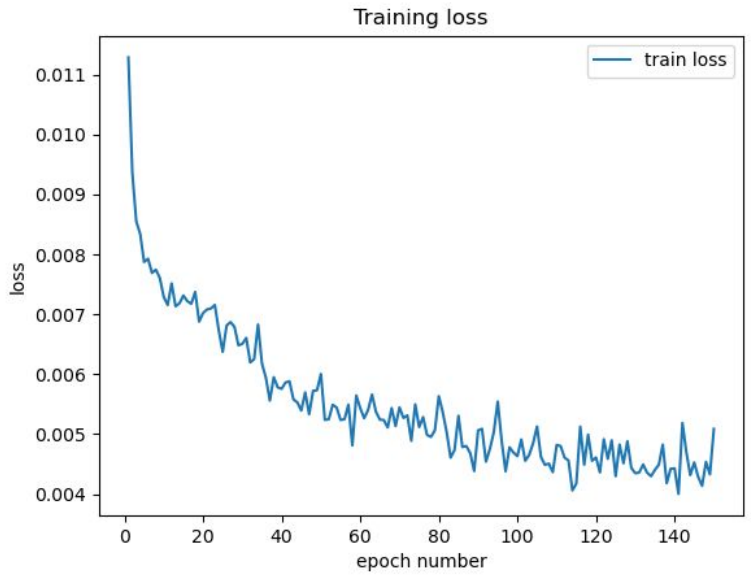
- **You can adjust the hyper-parameters according to your own ideas.**

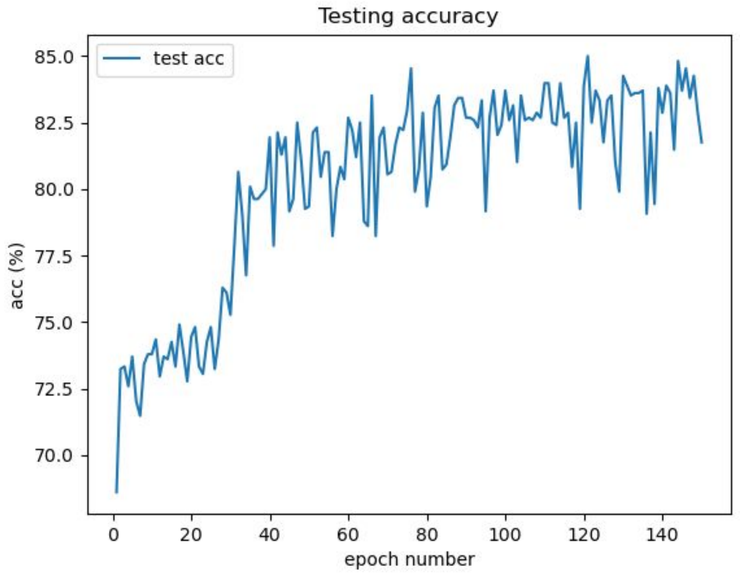
- If you use "nn.CrossEntropyLoss", don't add softmax after final fc layer because this criterion combines LogSoftMax and NLLLoss in one single class.

Result Comparison

- To visualize the accuracy trend, you need to plot each epoch accuracy and loss during training phase and testing phase.
- In this part, you can use the matplotlib library to draw the graph.







Report Spec

1. Introdution (10%)
2. Experiment setup (30%)
    1. The detail of your model
    2. Explain the ELU function
3. Experiment results (40%)
    1. The highest testing accuracy (screenshot)
    2. Figures of accuracy and loss curve
    3. Results of different alpha value in ELU
    4. Results of different dropout probability
    5. Anything you want to present
4. Discussion (20%)
    1. Anything you want to present

- ---- Criterion of result (30%) ----
- Accuracy > = 87% = 100 pts
- Accuracy 85~87% = 90 pts
- Accuracy 80~85% = 80 pts
- Accuracy 75~80% = 70 pts
- Accuracy < 75% = 60 pts

- **Score: 30% accuracy results + 70% report**
- **P.S If the upload file name or the report spec have format error, it will be penalty (-5).**

Reference

[1] EEGNet: A Compact Convolutional Neural Network for EEG-based Brain-Computer Interfaces