

Lab 3 - EEG classification

Department of Computer Science, NYCU

TA Chao-Yu, Chen

Important Rules

Important Date :

- Report Submission Deadline: 4/27 (Tue) **11:55 a.m.**
- Demo date: 4/27 (Tue)

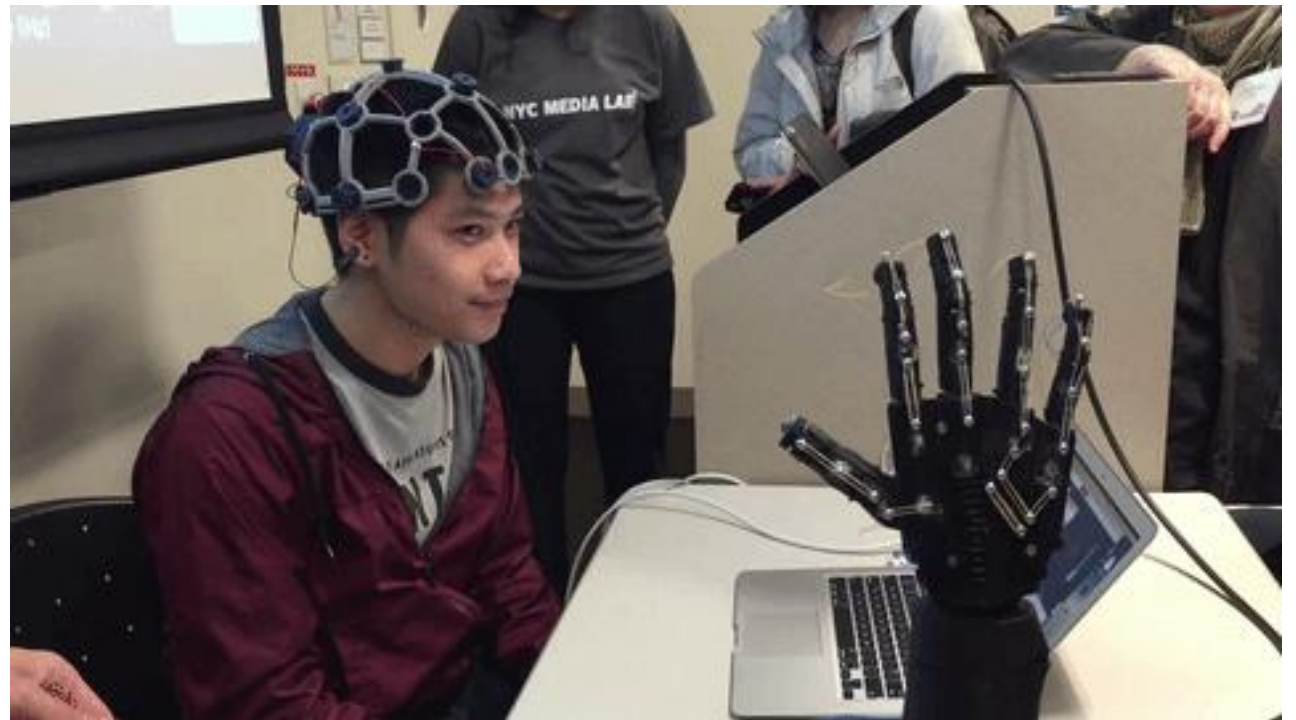
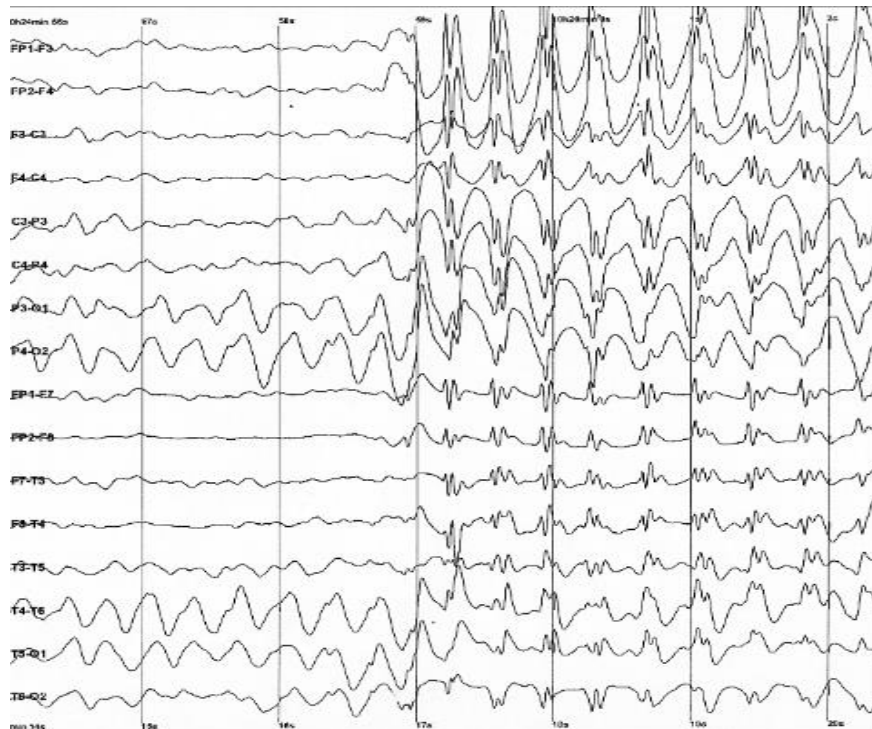
Turn in :

- Experiment Report (.pdf)
- Source code (.py)

Notice: zip all files in one file and name it like 「**DLP_LAB3_your studentID_name.zip**」, ex: 「**DLP_LAB3_0851909_陳昭宇.zip**」

Lab Objective

- In this lab, you will need to implement simple EEG classification models which are **EEGNet**, **DeepConvNet[1]** with BCI competition dataset. Additionally, you need to try different kinds of activation function including 『**ReLU**』 , 『**Leaky ReLU**』 , 『**ELU**』 .



Requirements

- Implement the EEGNet, DeepConvNet with three kinds of activation function including 『ReLU』 , 『Leaky ReLU』 , 『ELU』 .
- In the experiment results, you have to show the highest accuracy (not loss) of two architectures with three kinds of activation functions.
- To visualize the accuracy trend, you need to plot each epoch accuracy (not loss) during training phase and testing phase.

Dataset

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

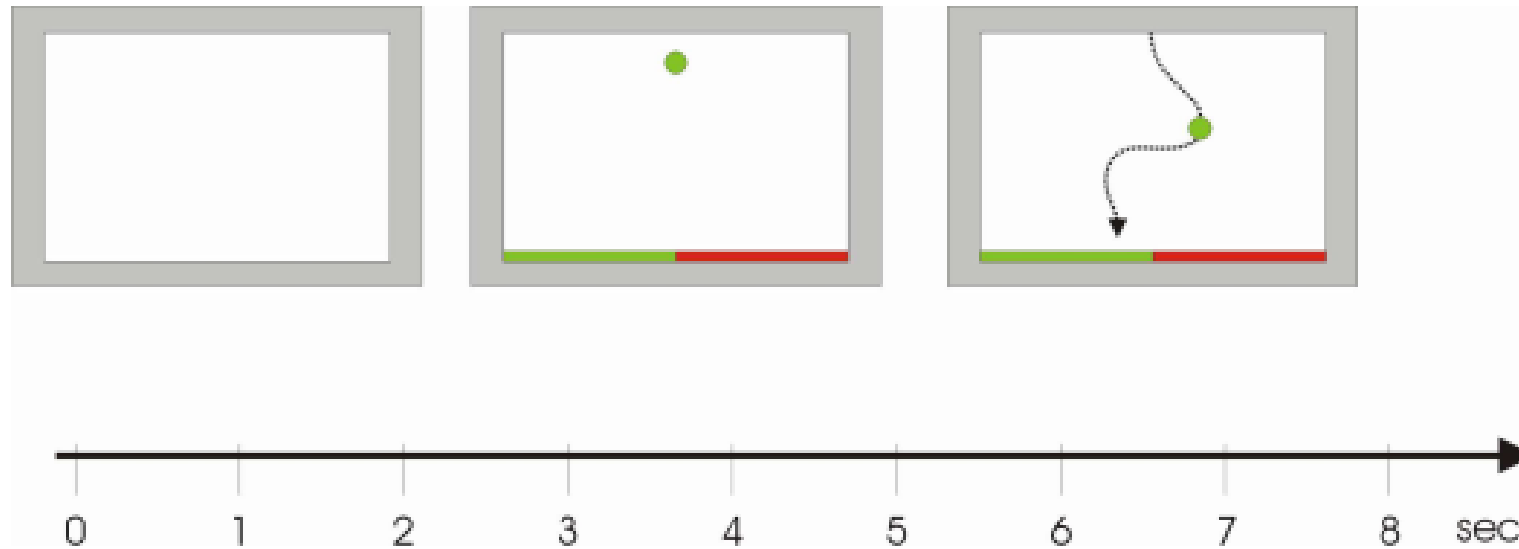


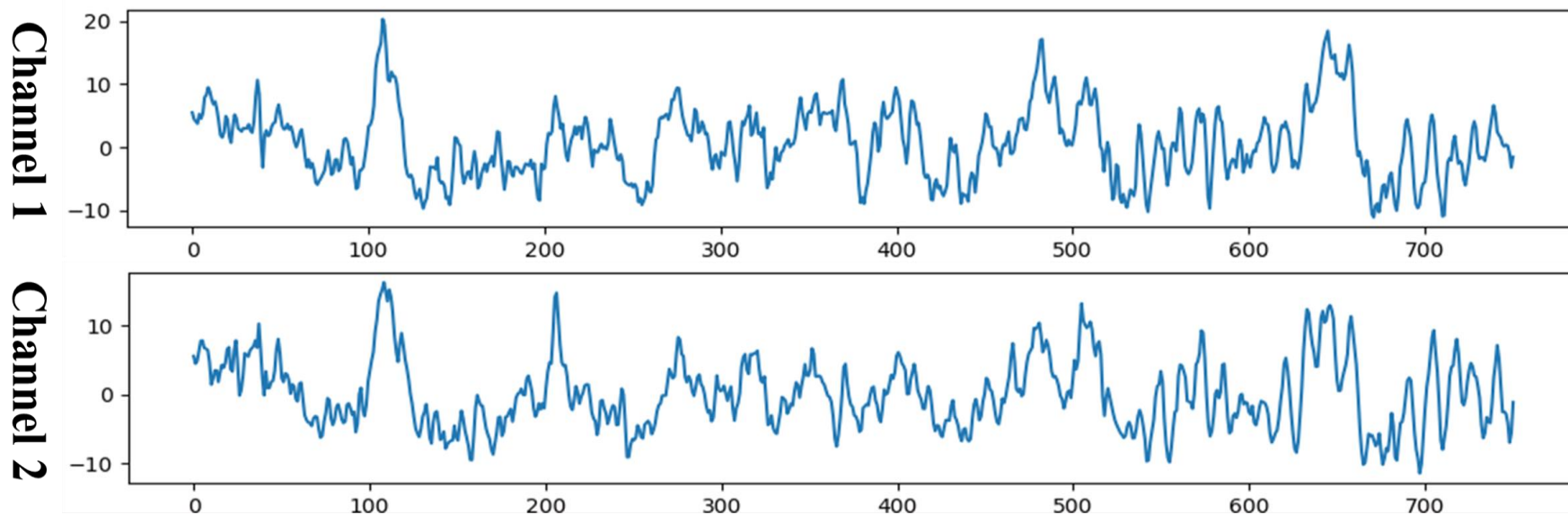
Figure 3: Basket paradigm used for S4 and X11 [3].

Prepare Data

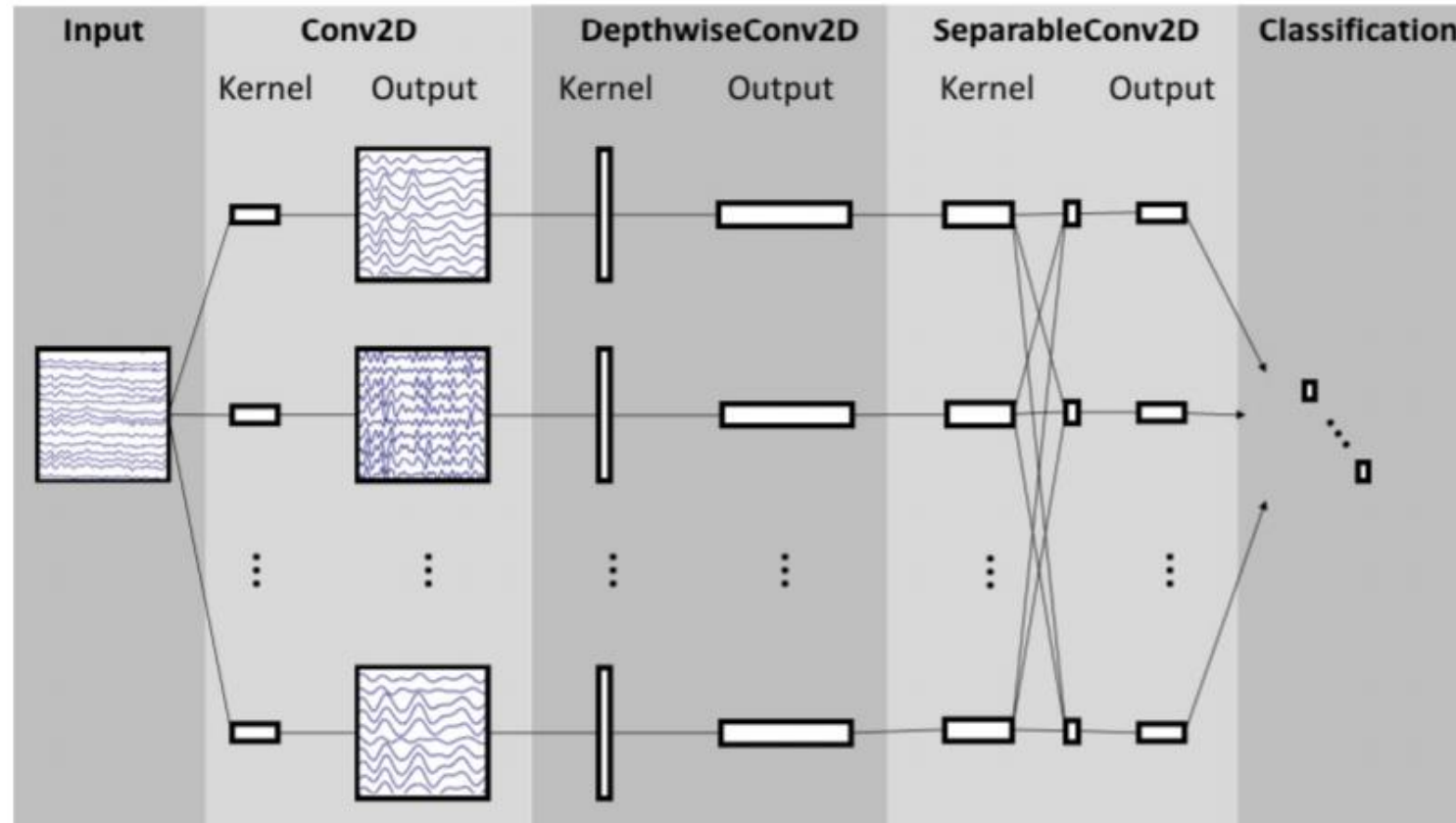
- Training data: S4b_train.npz, X11b_train.npz
- Testing data: S4b_test.npz, X11b_test.npz
- To read the preprocessed data, refer to the “dataloader.py”.

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 - DeepConvNet

- You need to implement the DeepConvNet architecture by using the following table, where $C = 2$, $T = 750$ and $N = 2$. **The max norm term is ignorable.**

Layer	# filters	size	# params	Activation	Options
Input		(C, T)			
Reshape		(1, C, T)			
Conv2D	25	(1, 5)	150	Linear	mode = valid, max norm = 2
Conv2D	25	(C, 1)	$25 * 25 * C + 25$	Linear	mode = valid, max norm = 2
BatchNorm			$2 * 25$		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Conv2D	50	(1, 5)	$25 * 50 * C + 50$	Linear	mode = valid, max norm = 2
BatchNorm			$2 * 50$		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Conv2D	100	(1, 5)	$50 * 100 * C + 100$	Linear	mode = valid, max norm = 2
BatchNorm			$2 * 100$		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Conv2D	200	(1, 5)	$100 * 200 * C + 200$	Linear	mode = valid, max norm = 2
BatchNorm			$2 * 200$		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
Dropout					p = 0.5
Flatten					
Dense	N			softmax	max norm = 0.5

The input data has reshaped to [B, 1, C, T]

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.

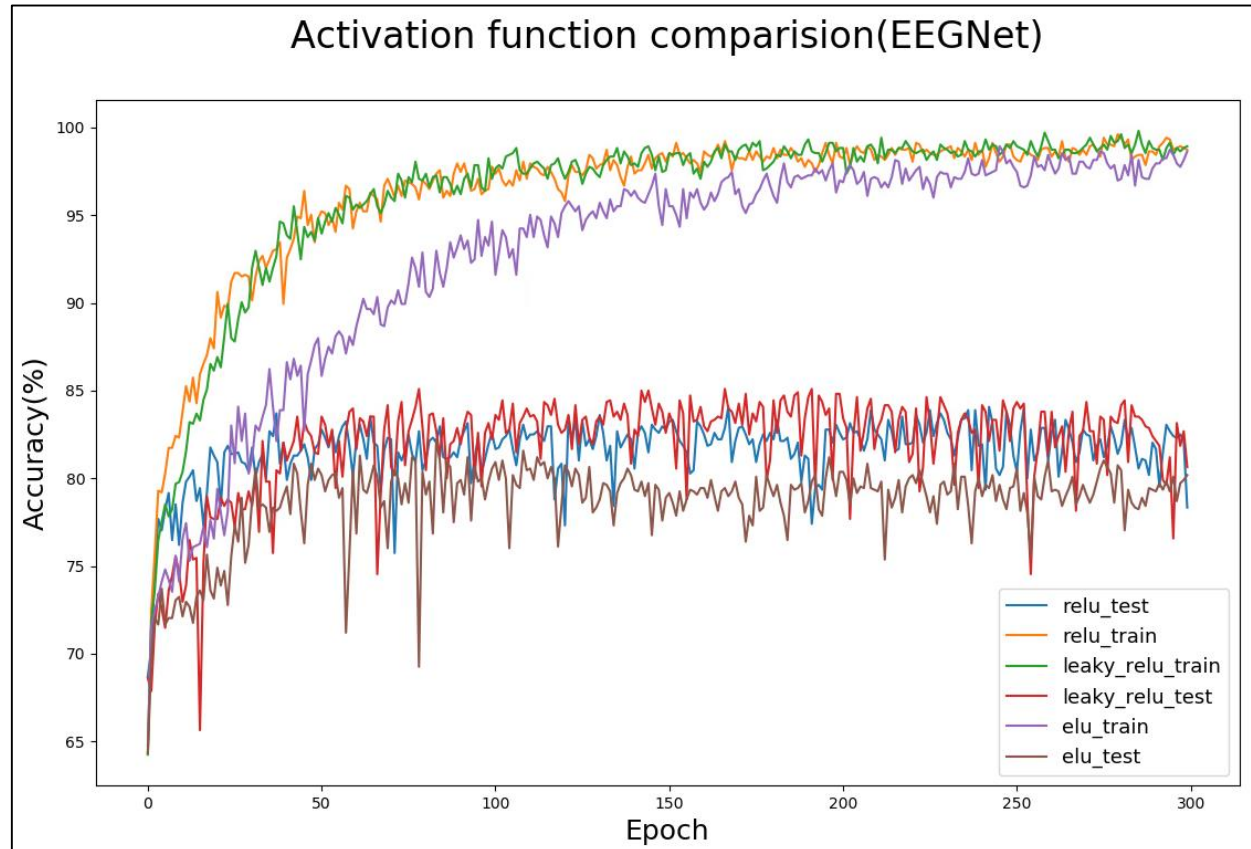
Result Comparison

- You have to show the highest accuracy (not loss) of two architectures with three kinds of activation functions.

	ReLU	Leaky ReLU	ELU
EEGNet	85.33%	84.63%	82.93%
DeepConvNet	82.75%	86.33%	85.73%

Result Comparison

- To visualize the accuracy trend, you need to plot each epoch accuracy (not loss) during training phase and testing phase.
- In this part, you can use the matplotlib library to draw the graph.



Report Spec

1. Introduction (20%)
2. Experiment set up (30%)
 - A. The detail of your model
 - EEGNet
 - DeepConvNet
 - B. Explain the activation function (ReLU, Leaky ReLU, ELU)
3. Experimental results (30%)
 - A. The highest testing accuracy
 - Screenshot with two models
 - anything you want to present
 - B. Comparison figures
 - EEGNet
 - DeepConvNet
4. Discussion (20%)
 - A. Anything you want to share

- ----- Criterion of result (40%) -----
- Accuracy $\geq 87\%$ = 100 pts
- Accuracy 85~87% = 90 pts
- Accuracy 80~85% = 80 pts
- Accuracy 75~80% = 70 pts
- Accuracy $< 75\%$ = 60 pts
- **Score: 40% experimental results + 60% (report+ demo score)**
- **P.S If the zip file name or the report spec have format error, it will be penalty (-5).**

How to Debug?

- *Data shape or type mismatching?*
- *Data shuffle in training?*
- *Wrong on layer setting?*



Christian Wolf
@chriswolfvision

跟隨



How to debug in PyTorch, for Tensorflow coders.

You are welcome.✅

```
# your neural network code here  
  
print (x)  
  
# other code here
```

Reference

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