Lab 2 – CNN Classifier

Department of Computer Science, NCTU

TA Yan-Yu, Tien

Important Rules

Important Date :

- Report Submission Deadline: 9/16 (Wed) 11:59 a.m.
- Demo date: 9/16 (Wed)

Turn in :

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

Notice :

zip all files in one file and name it like 「DLP_LAB2_yourID_name.zip」,
ex: 「DLP_LAB2_0760447_王大明.zip」

Email to :

92242@saes.tc.edu.tw

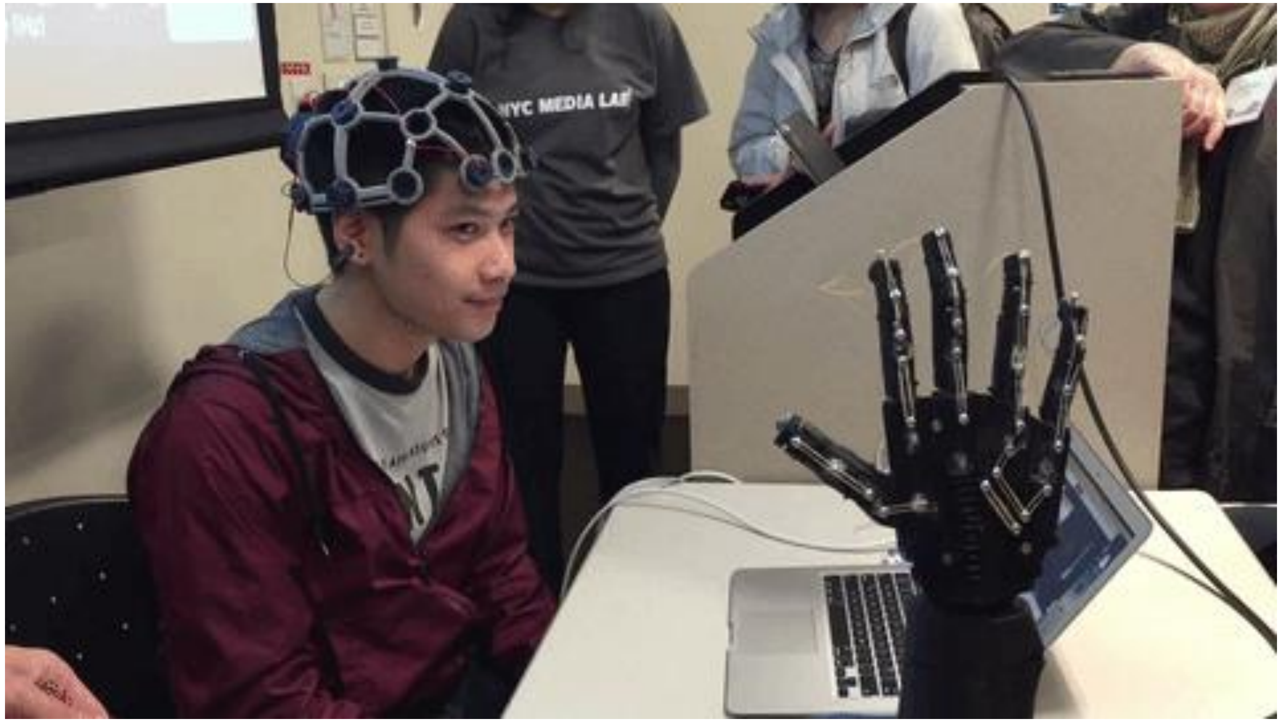
- Subject: MTK_DLP_LAB2_yourID_name

Lab Description

- Familiar with convolutional neural network structure
- Familiar with convolutional layer design by using pytorch
- Understand the difference of activation functions
- Finish the classifier task
- Custom dataloader is not required in this lab

Lab Objective

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



Requirements

- Implement the DeepConvNet, EEGNet 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.bbci.de/competition/iii/desc_IIIb.pdf*

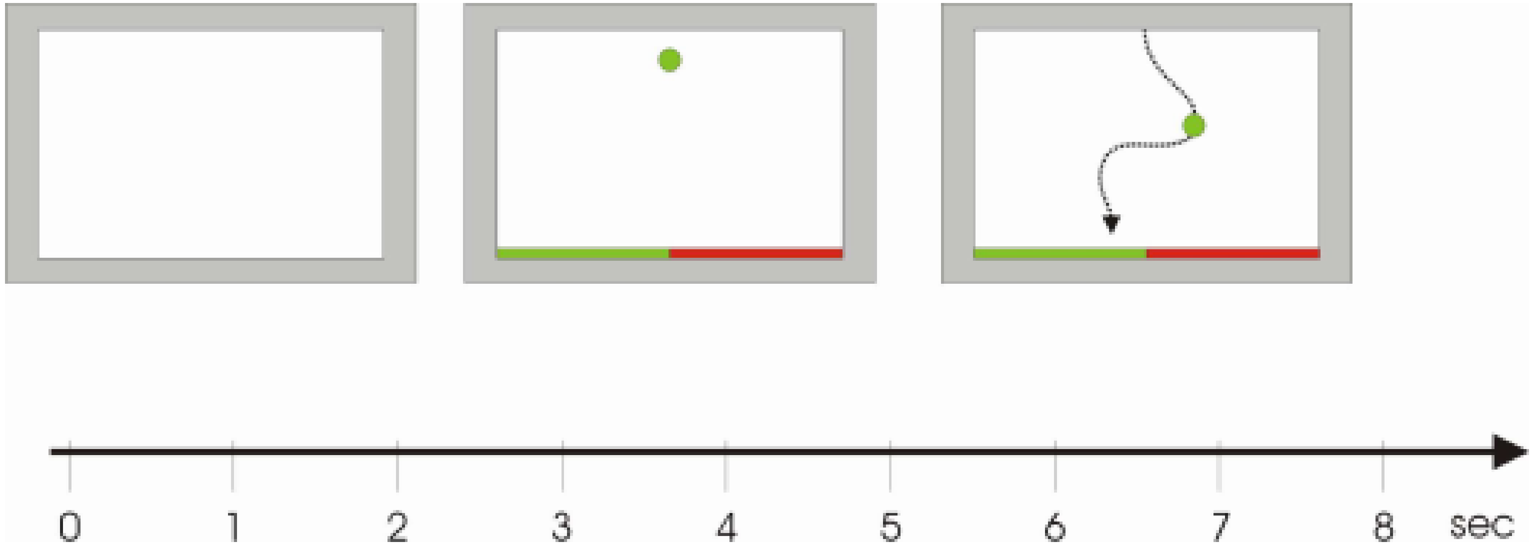
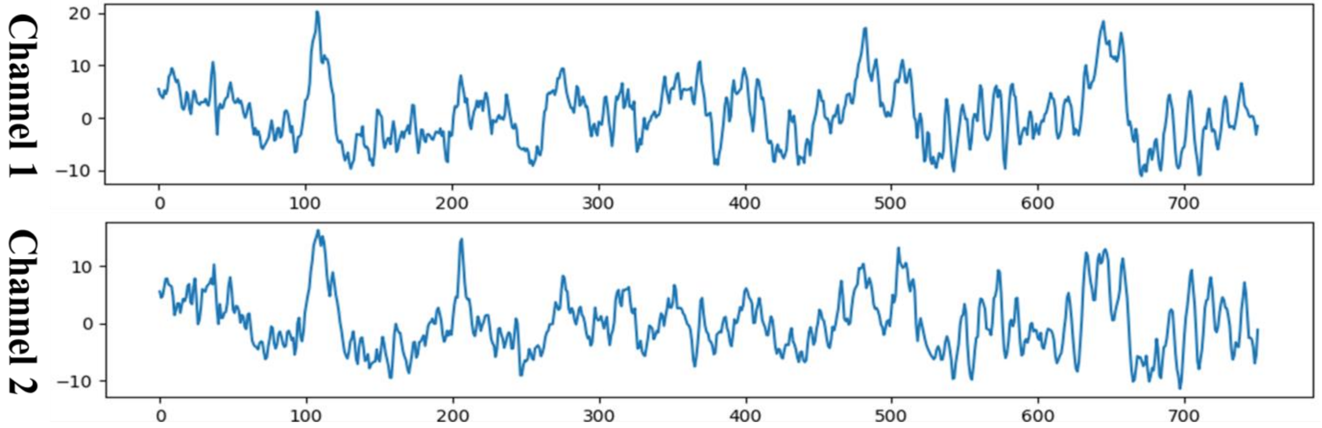


Figure 3: Basket paradigm used for S4 and X11 [3].

Prepare Data

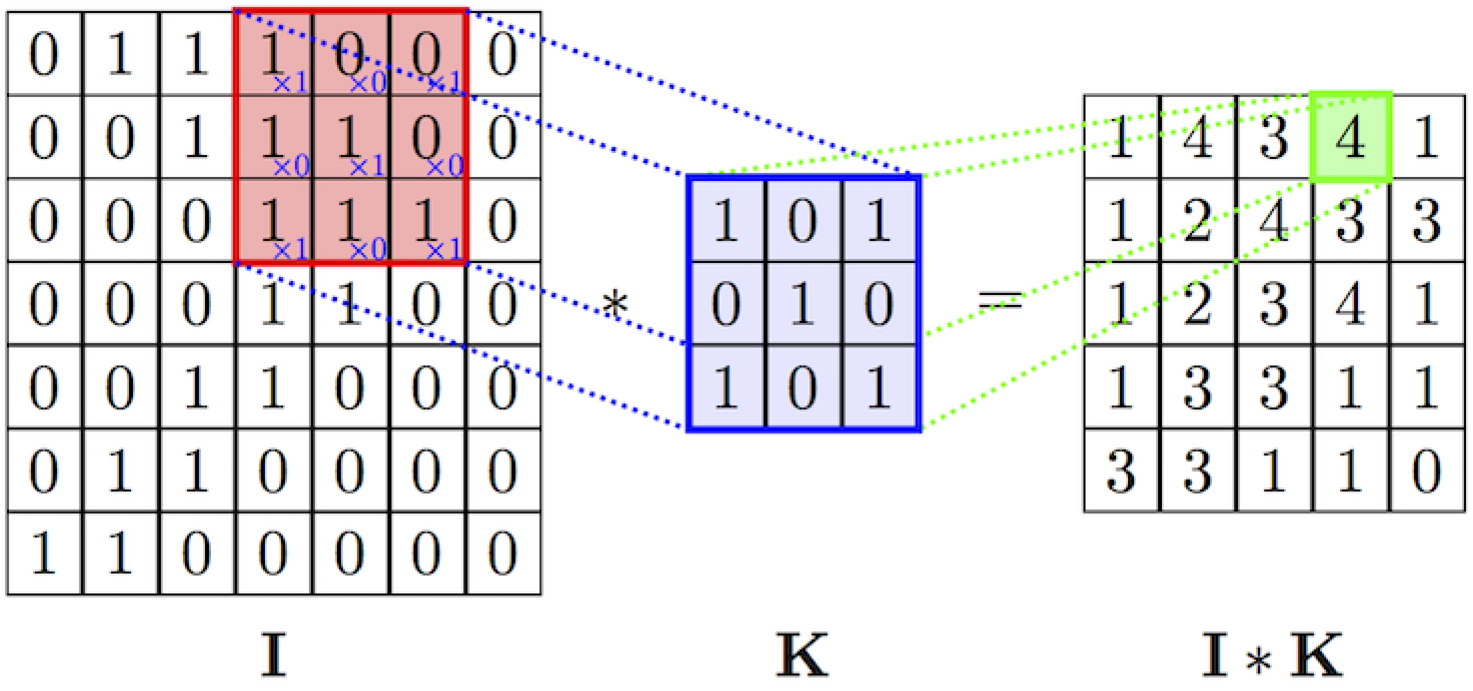
- Training data: S4b_train.npz, X11b_train.npz
- Testing data: S4b_test.npz, X11b_test.npz
- To read the preprocessed data, refer to the "read_bci_data.py".
- Prepared data
  - Train data: [1080, 1, 2, 750]
  - Train label: [1080]
  - Test data: [1080, 1, 2, 750]
  - Test label: [1080]



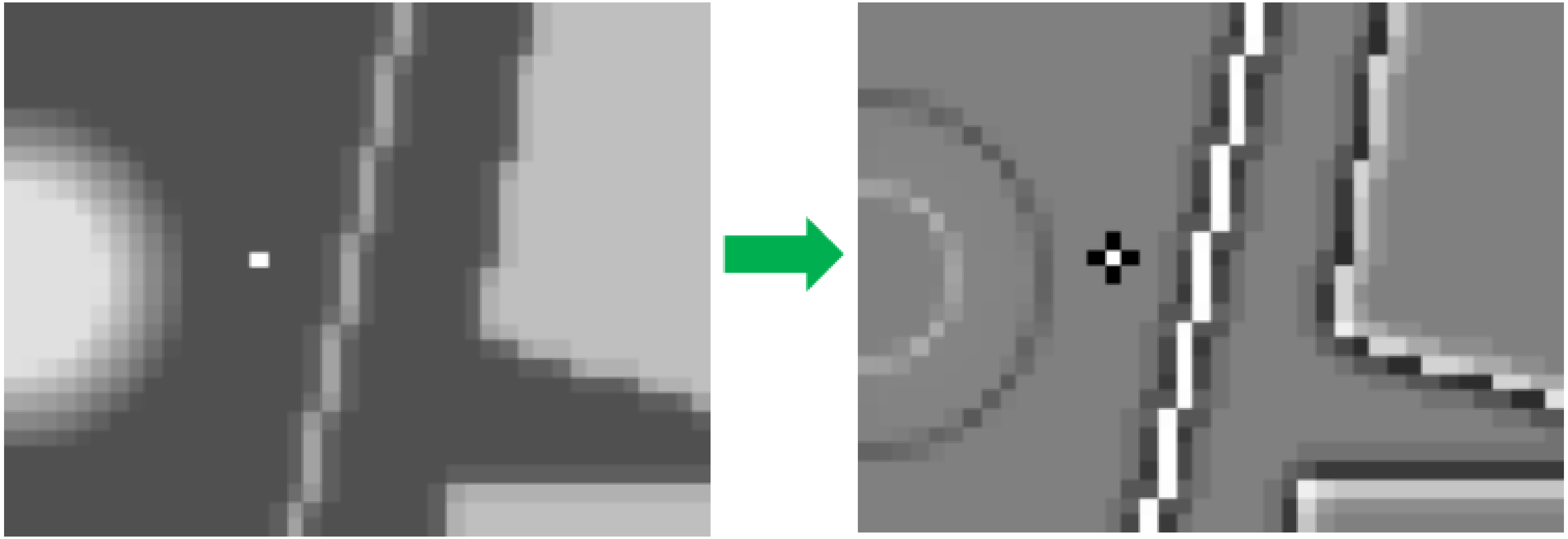
- **Input: [B, 1, 2, 750]**

  B: batch size

Convolution layer

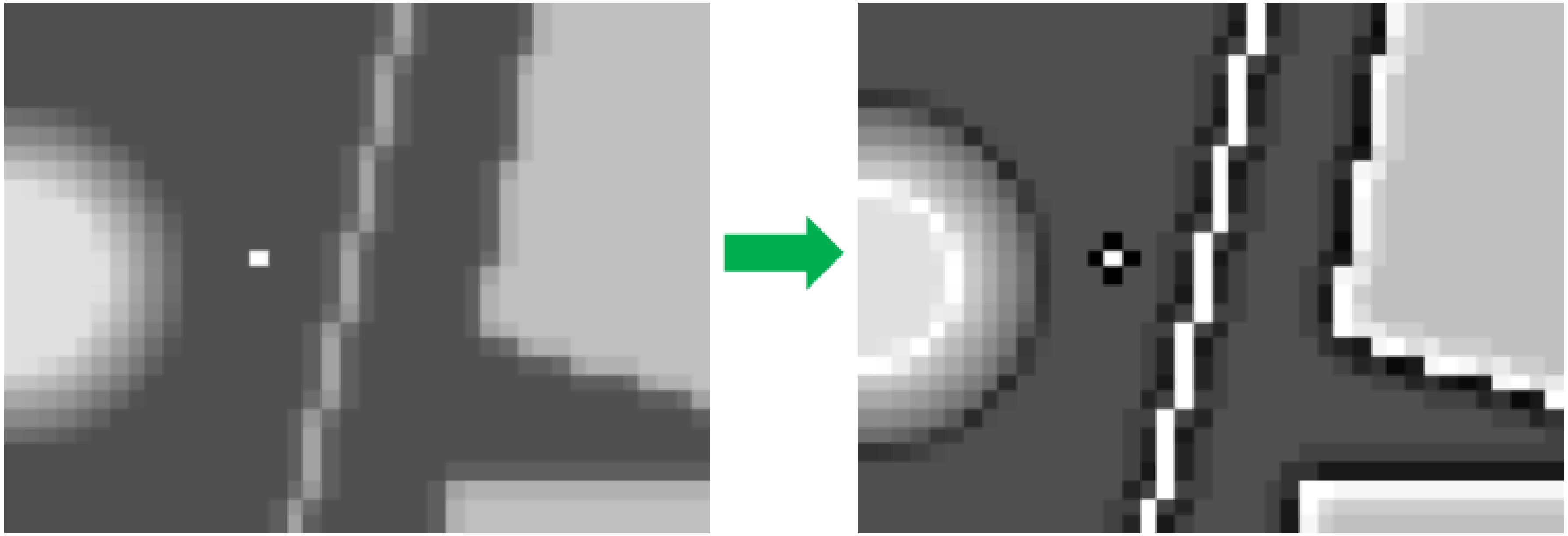


Convolution layer



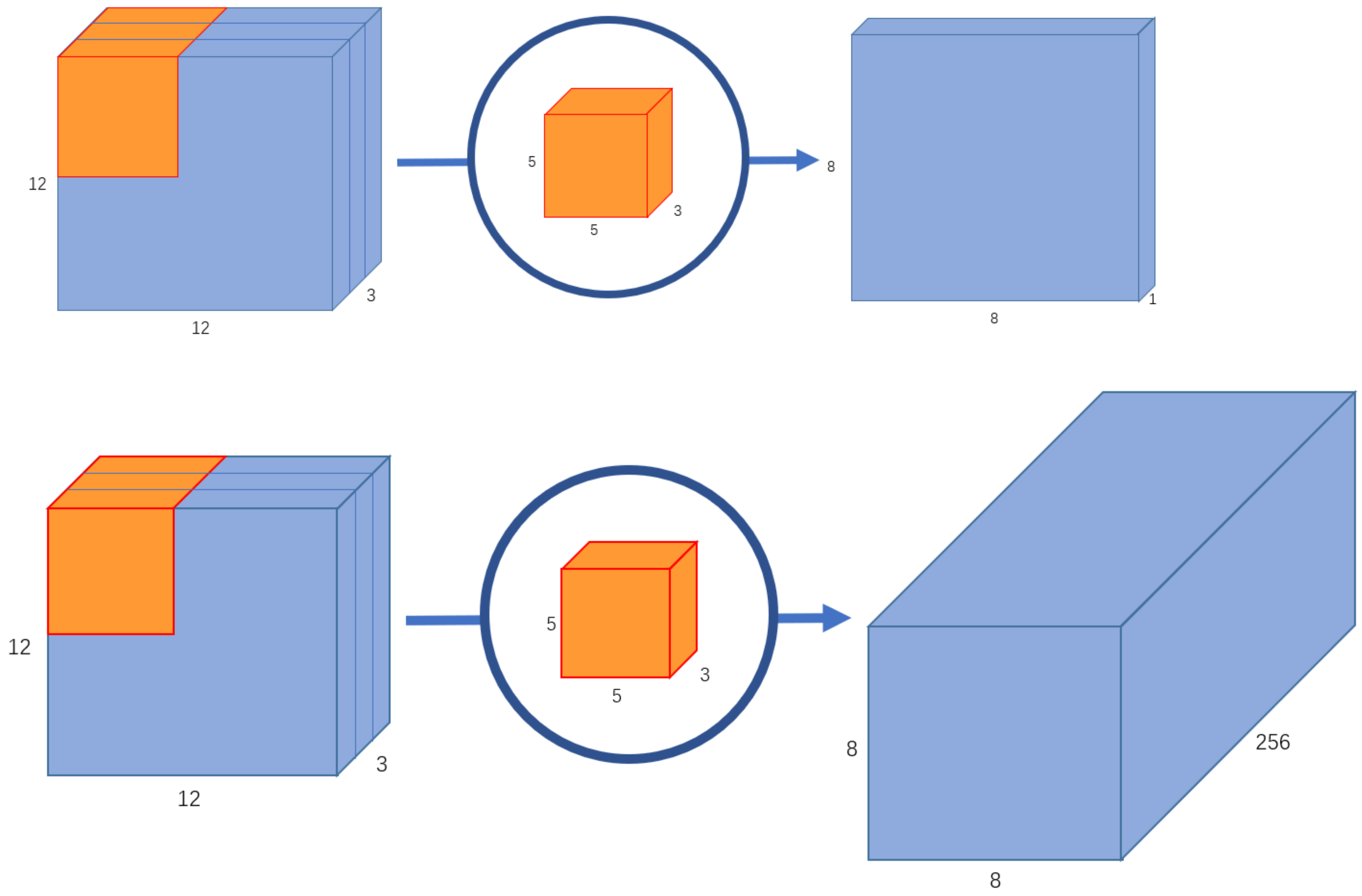
0	-1	0
-1	4	-1
0	-1	0

Convolution layer



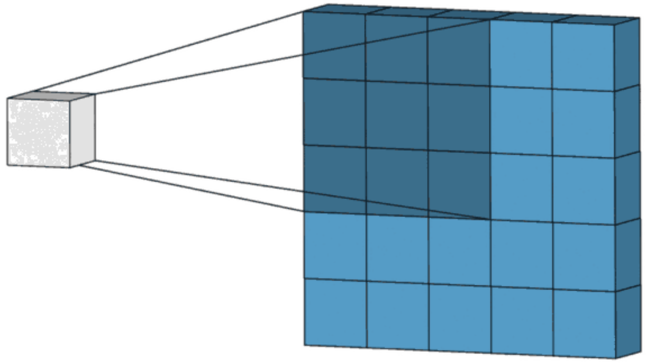
0	-1	0
-1	5	-1
0	-1	0

Convolution layer

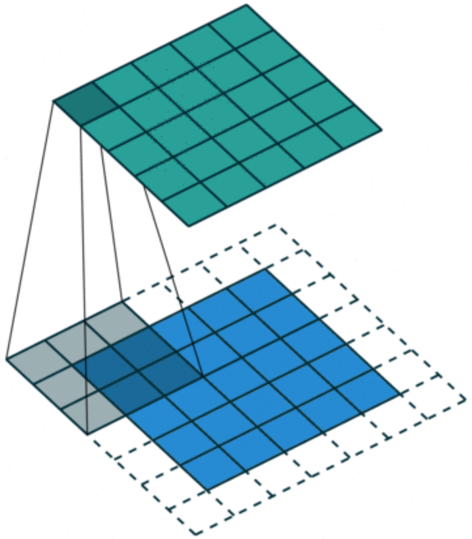


Reference: https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

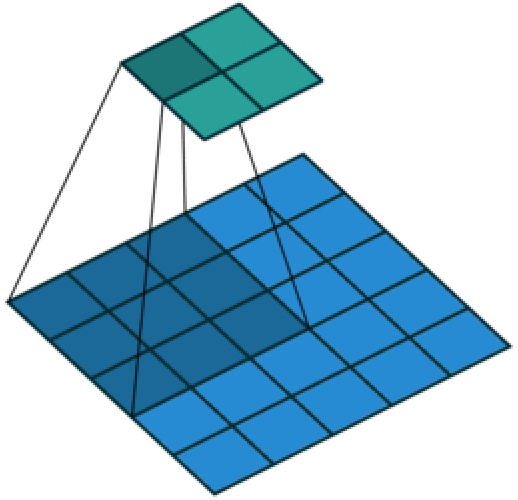
Convolution layer



kernal_size=3 in convolution



padding=1 in convolution



stride=2 in convolution

Reference: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

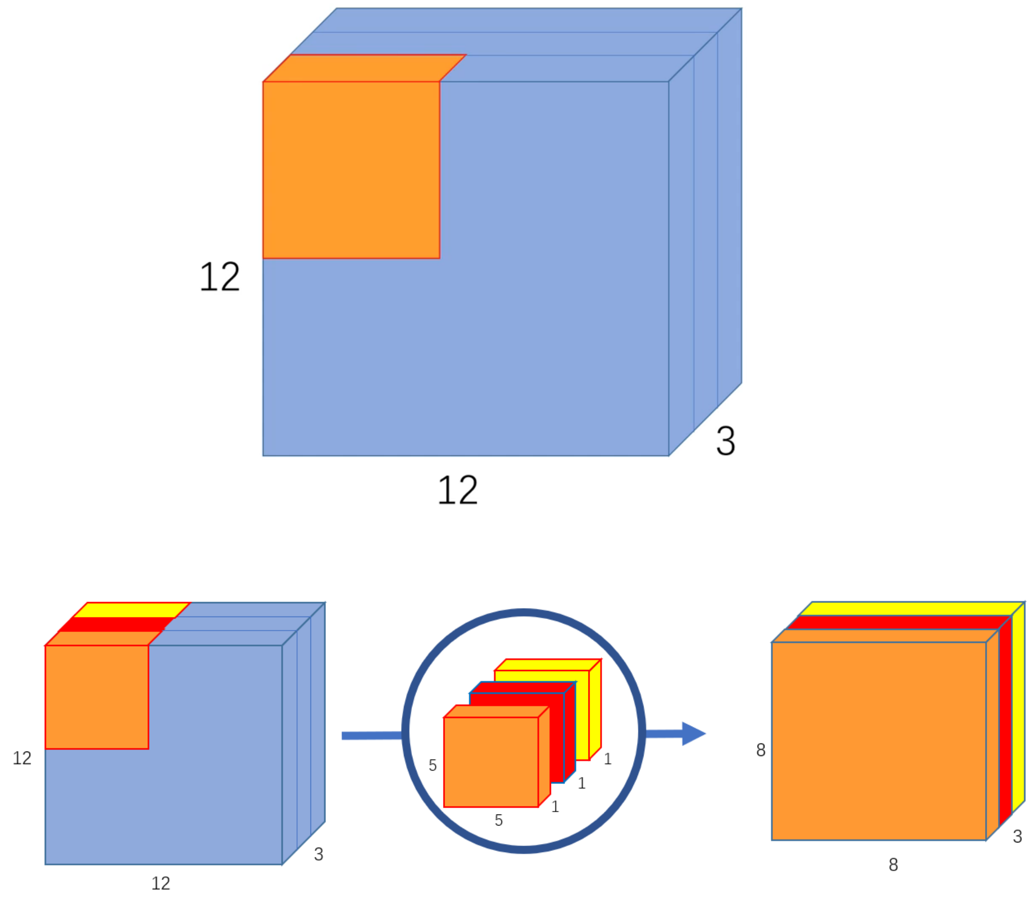
DeepConvNet

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

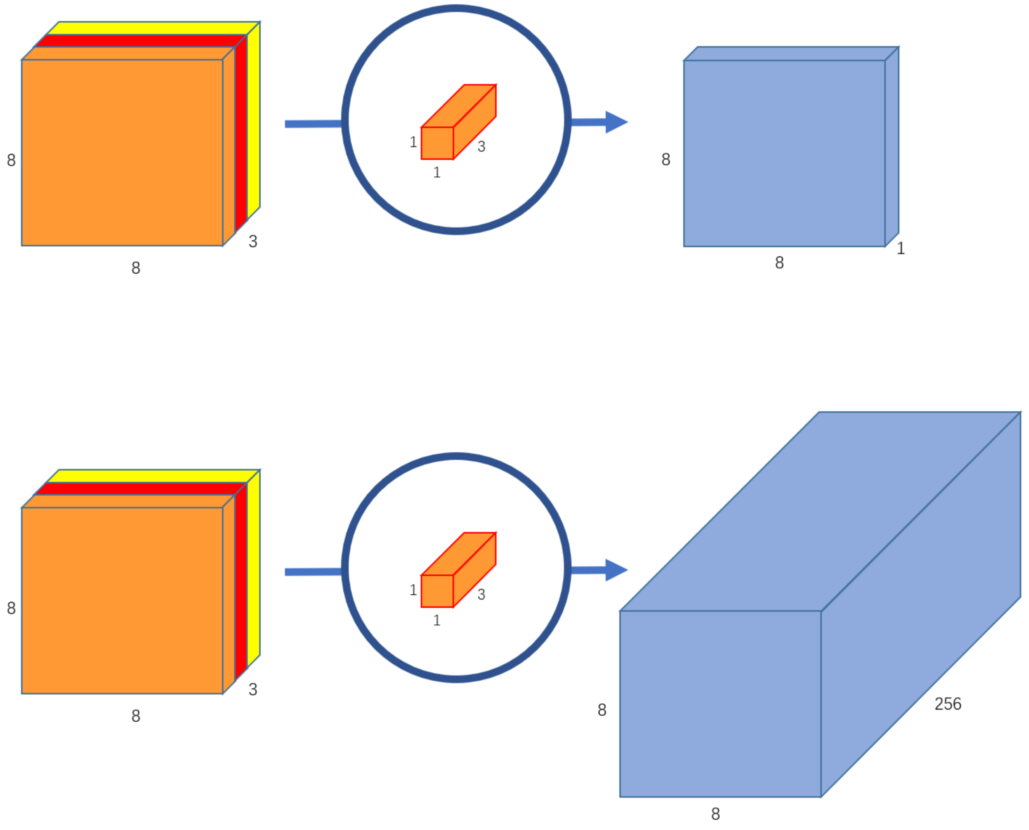
- You need to implement the DeepConvNet architecture by using the following table, where C = 2 and T = 750.
- The input data has reshaped to [B, 1, C, T]

Depthwise separable convolution

1- Depthwise Convolution

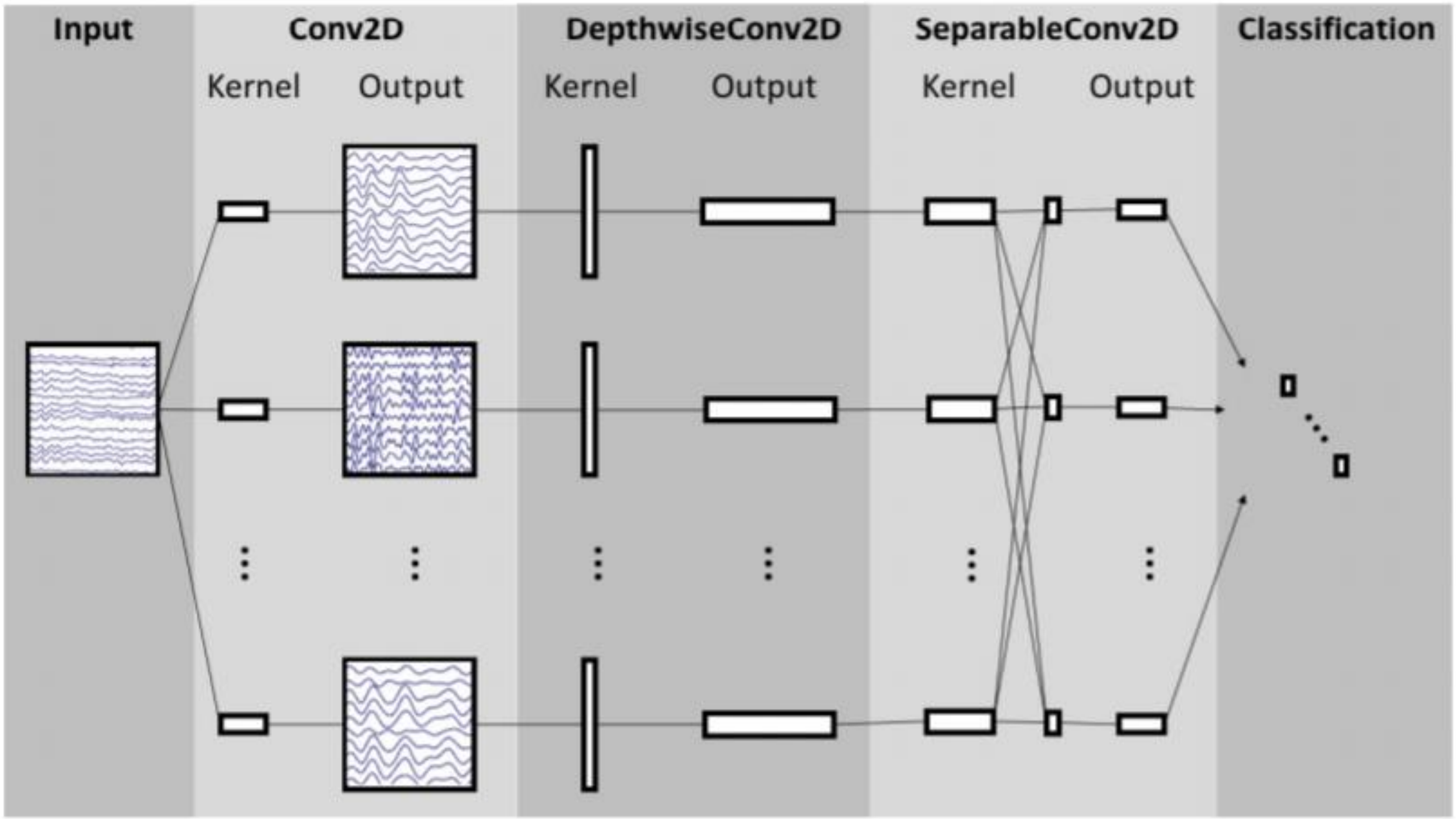


2- Pointwise convolutions



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

EEGNet



Reference: Depthwise Separable Convolution

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

EEGNet

- EEGNet implementation details

Block	Layer	# filters	size	Output	Activation	Options
1	Input			(C, T)		
	Reshape			(1, C, T)		
	Conv2D	F_1	(1, 51)	(F_1, C, T)		mode = same
	BatchNorm			(F_1, C, T)		
	DepthwiseConv2D	D * F_1	(C, 1)	(D * F_1, 1, T)		mode = valid, depth = D
	BatchNorm			(D * F_1, 1, T)		
	Activation			(D * F_1, 1, T)	ELU	
	AveragePool2D		(1, 4)	(D * F_1, 1, T // 4)		
	Dropout*			(D * F_1, 1, T // 4)		$p = 0.25$
2	SeparableConv2D	F_2	(1, 15)	(F_2, 1, T // 4)		mode = same
	BatchNorm			(F_2, 1, T // 4)		
	Activation			(F_2, 1, T // 4)	ELU	
	AveragePool2D		(1, 8)	(F_2, 1, T // 32)		
	Dropout*			(F_2, 1, T // 32)		$p = 0.25$
	Flatten			(F_2 * (T // 32))		
Classifier	Dense			N	?	

EEGNet architecture
C = number of channels
T = number of time points

F_1 = number of temporal filters (recommend F_1=16)

D = number of spatial filters (recommend D=2)

F_2 = number of pointwise filters (recommend F_2=32)

Classifier

- Classifier category
  - Binary classifier
  - Multi class classifier
  - Multi label classifier
- Output layer activation function
  - Softmax ex. [-0.5, 1.2, -0.1, 2.4] → [0.04, 0.21, 0.05, 0.70] (sum=1)
  - Sigmoid ex. [-0.5, 1.2, -0.1, 2.4] → [0.37, 0.77, 0.48, 0.91]
- Loss function in pytorch (cross entropy)
  - nn.CrossEntropyLoss = softmax + cross entropy
  - nn.BCELoss = Binary Cross Entropy
  - nn.BCEWithLogitsLoss = Sigmoid + BCELoss

Hyper Parameters

- Batch size = 64
- Learning rate = 1e-2
- Epochs = 150
- Optimizer: Adam

- nn.Conv2d Doc
  - https://pytorch.org/docs/stable/generated/torch.nn.Conv2d.html
  - Hint: stride, padding, **groups**

- **You can adjust the hyper-parameters according to your own ideas.**
- You **cannot** modify the architecture of DeepConvNet

Report Spec

1. Introduction (10%)
2. Experiment set up (35%)
   A. The detail of your model
      - DeepConvNet
      - EEGNet
   B. Explain the activation function (ReLU, LeakyReLU, ELU)
   C. Explain the output layer activation function and loss function
3. Experiment result (30%)
   A. The highest testing accuracy
      - Two models with three activation functions
      - Anything you want to present
   B. Comparison figures
      - Accuracy curve for two models
4. Discussion (25%)
   A. Depthwise separable convolution improve what issue in normal convolution
   B. Your training strategy
   C. Anything you want to share

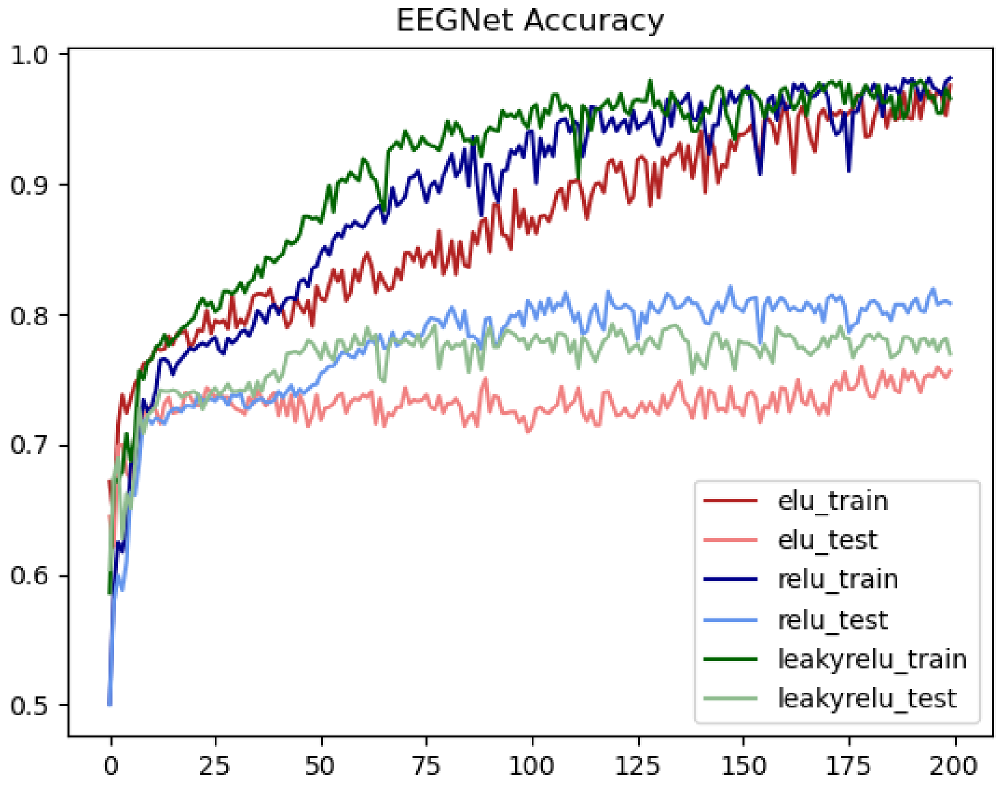
Result Comparison

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

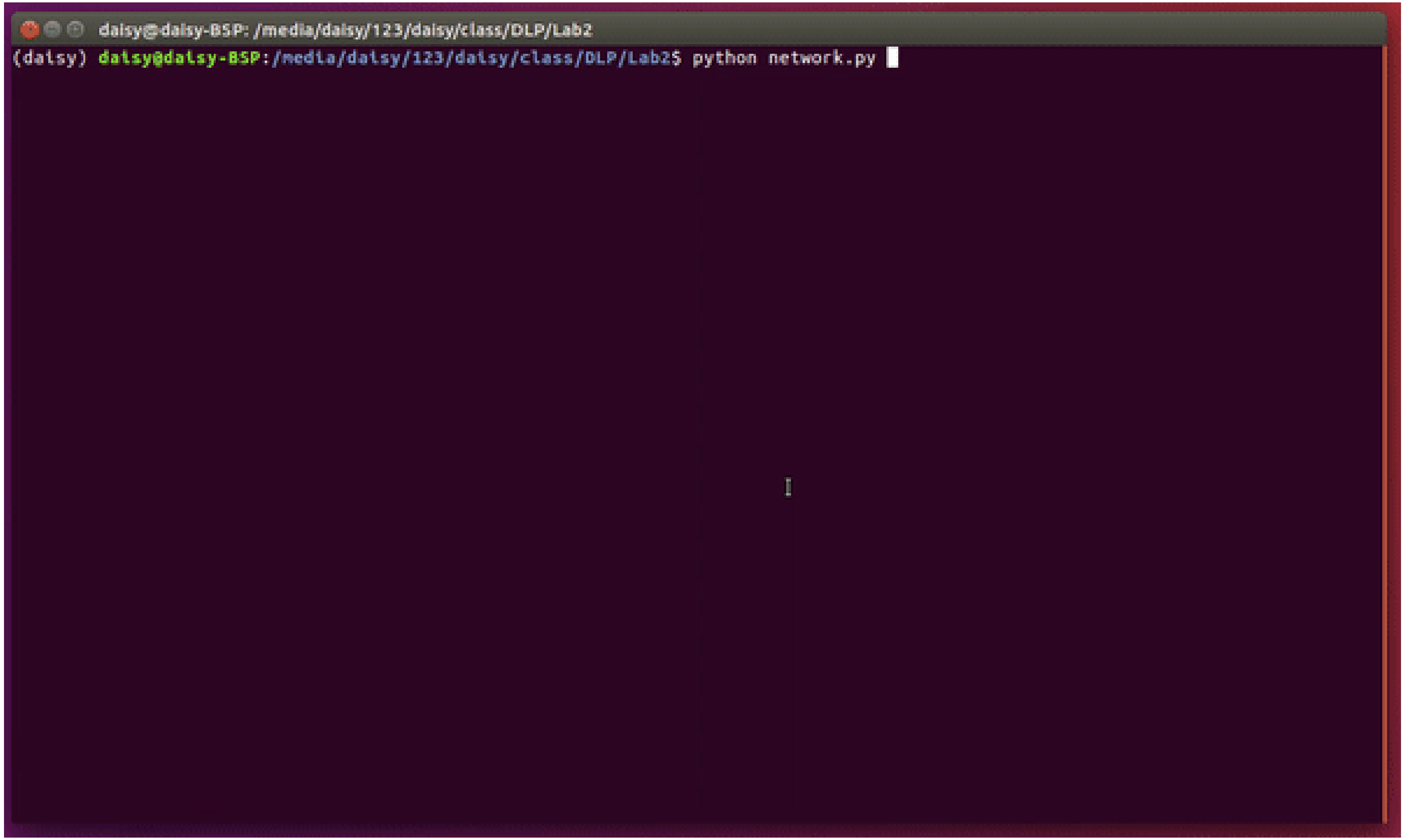
acc	ELU	ReLU	Leaky ReLU
EEGNet	84.90%	87.04%	**87.68%**
DeepConvNet	81.75%	82.75%	81.48%

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.



Example


daisy@daisy-BSP: /media/daisy/123/daisy/class/DLP/Lab2
(daisy) daisy@daisy-BSP:/media/daisy/123/daisy/class/DLP/Lab2$ python network.py

Example


(daisy) daisy@daisy-BSP:/media/daisy/123/daisy/class/DLP/Lab2$ python network.py

- ---- Criterion of result (40%) ----
- Accuracy > = 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).**

Reference

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

Custom Dataloader Practice Lab

Lab Objective

- Classifier for diabetic retinopathy (糖尿病所引發視網膜病變) analysis with ResNet architecture
- This dataset provided with a large set of high-resolution retina images taken under a variety of imaging conditions. **Format: .jpeg**

<u>Class</u>
0 - No DR
1 - Mild
2 - Moderate
3 - Severe
4 - Proliferative DR

Reference : https://www.kaggle.com/c/diabetic-retinopathy-detection#description

Prepare Data

- Download data
  - **28,099** images for training
  - **7025** for testing
- The image resolution is 512x512 and has been preprocessed.

- **Input: [B, 3, 512, 512]   Output: [B, 5]   Ground truth: [B]**

512 x 512

Prepare Data

- test_img.csv
- test_label.csv
- train_img.csv
- train_label.csv

```python
def getData(mode):
    if mode == 'train':
        img = pd.read_csv('train_img.csv')
        label = pd.read_csv('train_label.csv')
        return np.squeeze(img.values), np.squeeze(label.values)
    else:
        img = pd.read_csv('test_img.csv')
        label = pd.read_csv('test_label.csv')
        return np.squeeze(img.values), np.squeeze(label.values)
```

3798_left	0
9317_right	0
1991_right	0
2086_left	0
34952_left	0
18072_right	0
9958_left	0
32121_left	0
29612_left	0
21978_left	1
26746_left	0
21469_right	2
40812_right	0
22575_right	2

Image Format: .jpeg

Please do not sort !!!

Dataloader

- Implement your own custom DataLoader
- Below is the skeleton that you have to fill to have a custom dataset, refer to “dataloader_practice.py”

```python
class RetinopathyLoader(data.Dataset):
    def __init__(self, mode):

    def __len__(self):
        return ...

    def __getitem__(self, index):
        return ...
```

Dataloader

```python
def __init__(self, mode):
    """
    Args:
        mode : Indicate procedure status(train or test)

        self.root (str): Root path of the dataset.
        self.img_name (str list): String list that store all image names.
        self.label (int or float list): Numerical list that store all ground truth label values.
    """

def __len__(self):
    """'return the size of dataset"""
    return ...
```

Dataloader

```python
def __getitem__(self, index):
    """
       step1. load the image file
              hint : path = root + self.img_name[index] + '.jpeg'

       step2. Get the ground truth label from self.label

       step3. (optional)
              Transform the .jpeg rgb images during the training phase,
              such as resizing, random flipping, rotation, cropping, normalization etc.

              In the testing phase, if you have a normalization process during the training phase,
              you only need to normalize the data.

              hints: Convert the pixel value to [0, 1]
                     Transpose the image shape from [H, W, C] to [C, H, W]

        step4. Return processed image and label
    """
    return ...
```

Result

```
daisy@daisy-BSP: /media/daisy/123/daisy/class/DLP/Lab3
(daisy) daisy@daisy-BSP:/media/daisy/123/daisy/class/DLP/Lab3$ python dataloader_practice.py
> Found 28099 images...
> Found 7025 images...
Start training ...
Epoch 1 ---------------- 403 sec
        training loss: 6451.315356582403, train acc: 0.73507954019716, test acc: 0.7335231316725979
```