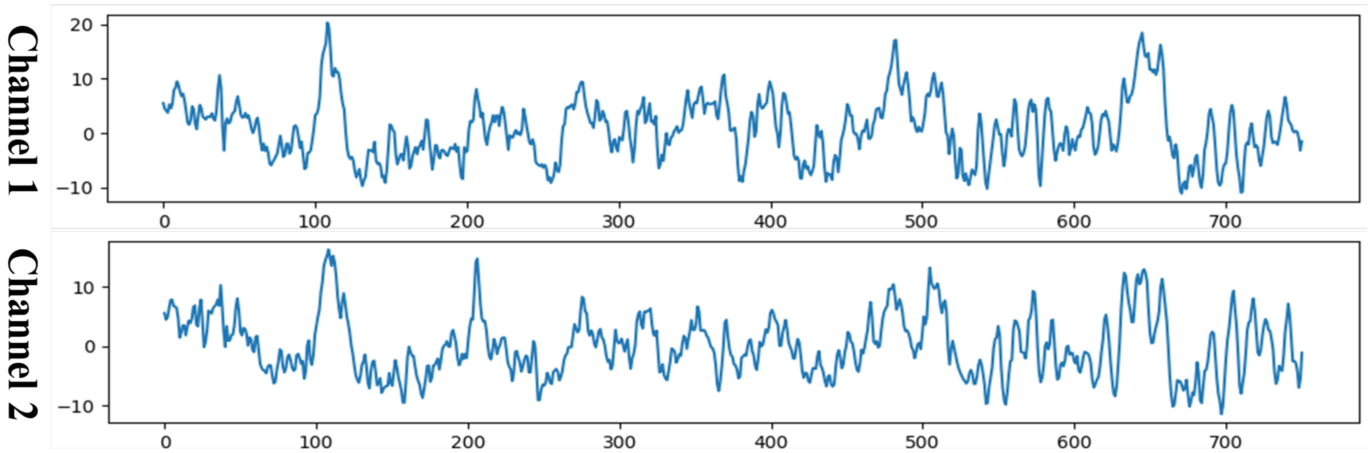
311551180_資工所_楊佳誠

Introduction

The aim of this work is to test whether EEGNet performs better than DeepConvNet. In addition, different activations are used to test the performance.

The dataset is a BCI competition dataset, which consists of EEG signals. The dimensions of the dataset are [B, 1, 2, 750], where B is the batch size, 2 means there are 2 channels, and 750 represents the time duration. The problem we need to deal with is a binary classification problem with two labels.



Experiment set up

A. The detail of your model

The setting of my two models, which are EEGNet and DeepConvNet, are shown below:

EEGNet structure:

```
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)
  )
)
```

DeepConvNet structure:

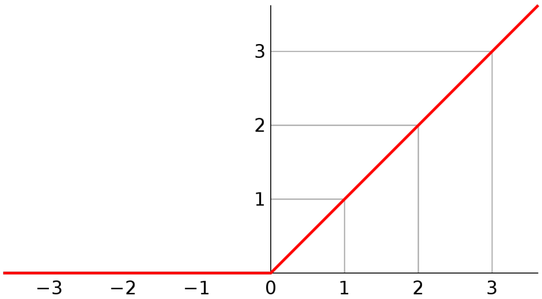
Layer	# filters	size	# params	Activation	Options
Input		(C, T)			
Reshape		(1, C, T)			
Conv2D	25	(1, 5)	150	Linear	mode = valid, max norm = 2
Conv2D	25	(C, 1)	25 * 25 * C + 25	Linear	mode = valid, max norm = 2
BatchNorm			2 * 25		epsilon = 1e-05, momentum = 0.1
Activation				ELU	
MaxPool2D		(1, 2)			
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

Other hyperparameters:

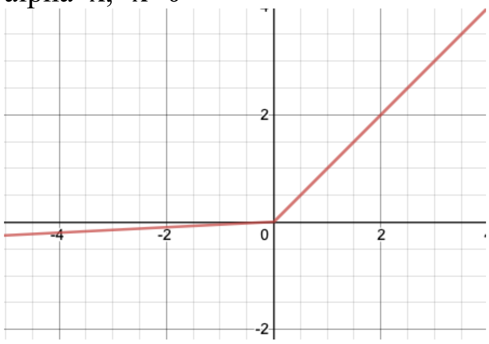
- Epoch : 500
- Batch : 64
- Learning rate : 10^-3
- Loss : nn.CrossEntropyLoss
- Optimizer : Adam
- Dataloader : shuffle = True

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

- ReLU : max(0,x)



- Leaky ReLU: Leaky ReLU is a variant of ReLU that has a small slope within the negative range.
  x,   x>0
  alpha*x, x<0



- ELU:

$$x < 0 : ELU(x) = (e^x - 1)\alpha$$

$$x \geq 0 : ELU(x) = x$$

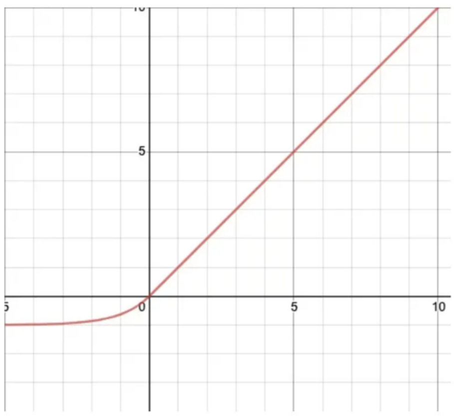


Fig 1. Exponential Linear Unit or ELU activation function

Comment of these activation functions:

- The introduction of the ReLU activation function was to solve the problem of vanishing gradients.
- The "dying ReLU" problem occurs when the ReLU activation function outputs zero for any input value less than or equal to zero. This can cause neurons in a deep neural network to stop learning because the gradient of the loss function becomes zero, leading to a "dead" neuron that does not contribute to the computation.
- Leaky ReLU is often preferred over ReLU because it can help to mitigate the issue of "dying ReLU", which occurs when a large portion of the neurons in the network become inactive and produce zero outputs.
- ELU's design satisfies two characteristics. The output distribution has zero mean, which can speed up the training process. The activation function is unilaterally saturated, which can lead to better convergence.

Experimental results

A. The highest testing accuracy

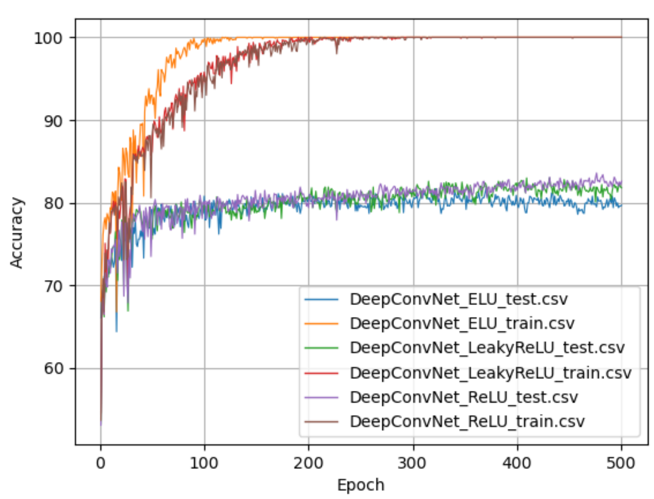
The highest test accuracy found in the record for each model and activation function is shown below:

	LeakyReLU	ReLU	ELU
EEGNet	86.9%	**87.1%**	82.8%
DeepConvNet	83.0%	83.6%	81.7%

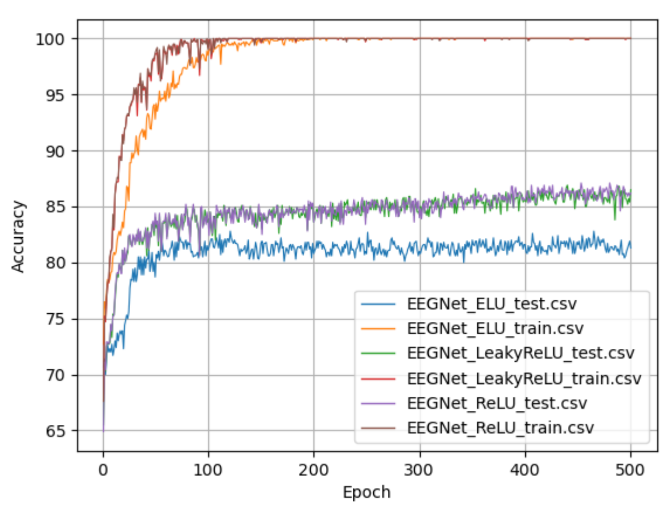
```
The highest accuracy found in each csv:
EEGNet_ELU_test.csv: 82.8
EEGNet_ELU_train.csv: 100.0
EEGNet_LeakyReLU_test.csv: 86.9
EEGNet_LeakyReLU_train.csv: 100.0
EEGNet_ReLU_test.csv: 87.1
EEGNet_ReLU_train.csv: 100.0
cheng@Roman-Yangs-MacBook-Pro ~/Desktop/研究所/深度學習/lab/lab3
```

```
The highest accuracy found in each csv:
DeepConvNet_ELU_test.csv: 81.7
DeepConvNet_ELU_train.csv: 100.0
DeepConvNet_LeakyReLU_test.csv: 83.0
DeepConvNet_LeakyReLU_train.csv: 100.0
DeepConvNet_ReLU_test.csv: 83.6
DeepConvNet_ReLU_train.csv: 100.0
cheng@Roman-Yangs-MacBook-Pro ~/Desktop/研究所/深度學習/lab/lab3
```

B. Comparison figures



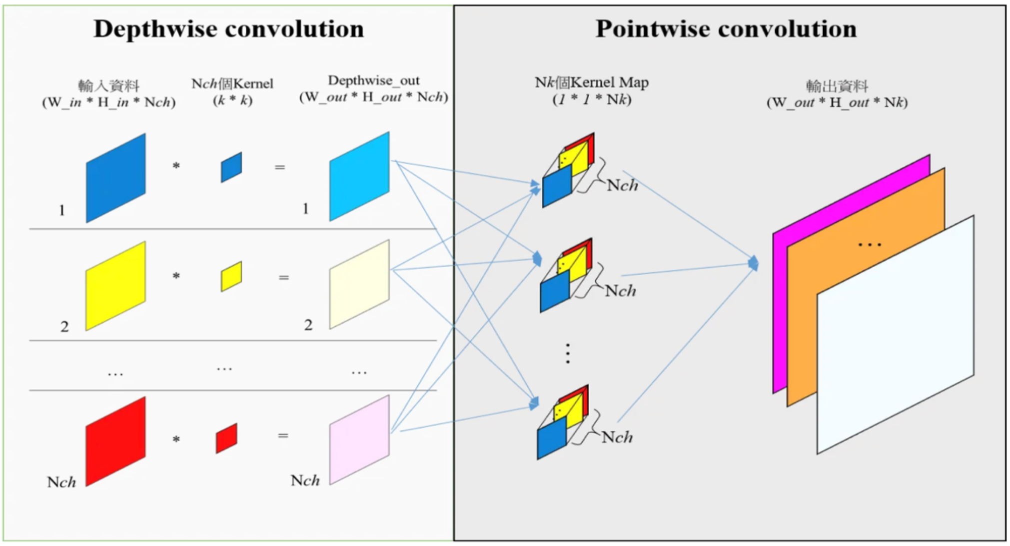
DeepConvNet



EEGNet

Discussion

Why does depthwise convolution perform better in EEG classification?
Answer from **EEGNet: A Compact Convolutional Neural Network for EEG-based Brain-Computer Interfaces**



Depthwise separable convolution

- In CNN applications for computer vision the main benefit of a depthwise convolution is reducing the number of trainable parameters to fit, as these convolutions are not fully-connected to all previous feature maps.

- Importantly, when used in EEG-specific applications, this operation provides a direct way to learn spatial filters for each temporal filter, thus enabling the efficient extraction of frequency-specific spatial filters (see the middle column of Figure 1).

- Conclusion : Depthwise convolution can capturing spatial information:
  EEG signals are inherently spatial in nature, and depthwise convolution has been shown to capture spatial information better than traditional convolution. **By applying separate filters to each input channel, depthwise convolution captures the unique spatial relationships between the input channels.**