**Lab2: EEG classification**

Author: 311553007應耀德

1. **Introduction**

**Lab Objective:**

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

**Requirements:**

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

**Data:**

BCI Competition III - IIIb Cued motor imagery with online feedback (non-stationary classifier) with 2 classes (left hand, right hand) from 3 subjects.

[2 classes, 2 bipolar EEG channels]

*Reference:* [*http://www.bbci.de/competition/iii/desc\_IIIb.pdf*](http://www.bbci.de/competition/iii/desc_IIIb.pdf)

* Train data: S4b\_train.npz、X11b\_train.npz

Test data: S4b\_test.npz、X11b\_test.npz

一張含有 文字 的圖片

自動產生的描述

Figure 1. Label distribution for data

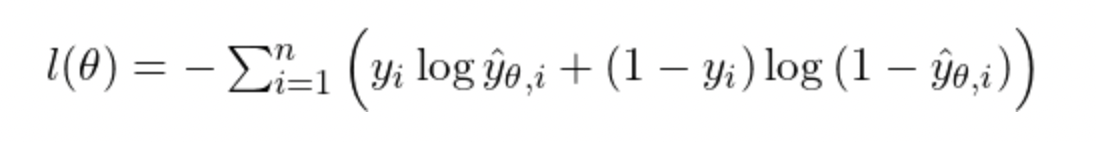
* BCIDataset



Figure 2. Implementation of BCIDataset

**Loss functions:**

* Cross-Entropy Loss



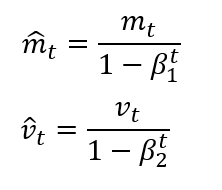
Where

**Optimizer:**

* Adam (Adaptive Moment Estimation)

一張含有 文字 的圖片

自動產生的描述

 一張含有 文字 的圖片

自動產生的描述

Where

1. **Experiment set up**

**Shared settings:**

* Batch Size: 512
* Epoch Size: 500
* Optimizer: Adam, learning rate = 0.001, weight decay = 0.1
* Loss function: Cross-Entropy Loss

**Base class:**

To share the way to select the activation function, extract the same parts to base class.

self.activation is a property of lambda function to generate selected activation function.

一張含有 文字 的圖片

自動產生的描述

Figure 3. Implementation of base class

**EEGNet:**

一張含有 文字 的圖片

自動產生的描述

Figure 4. Implementation of EEGNet

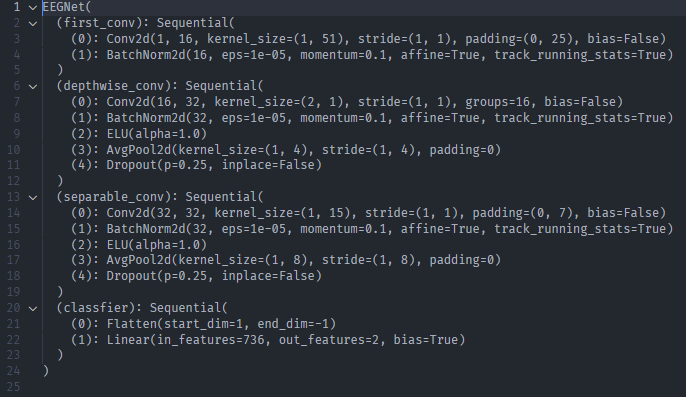


Figure 5. An example architecture of EEGNet with ELU activation function

**DeepConvNet:**



Figure 6. Implementation of DeepConvNet

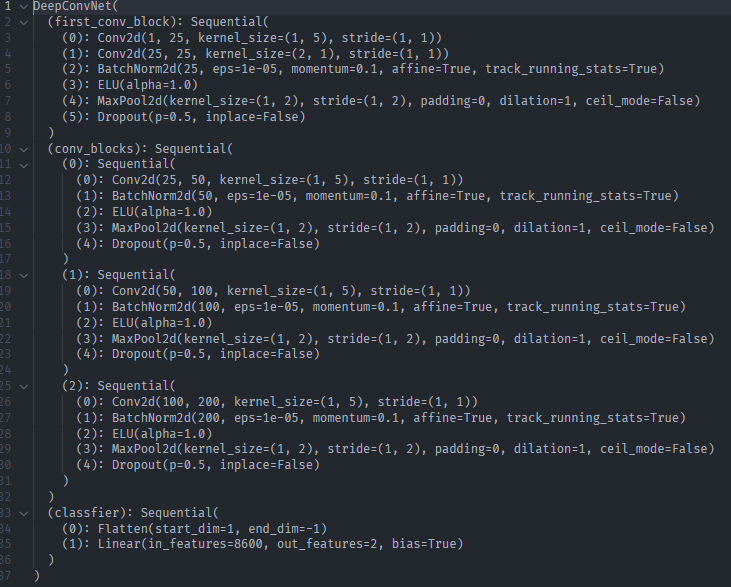


Figure 7. An example architecture of DeepConvNet with ELU activation function

**Activation functions (ReLU, Leaky ReLU, ELU):**

* ReLU

ReLU is a good activation function for hidden units. It diminishes the vanishing gradient problem happened from Sigmoid and Tanh. But it still has a chance to cause this problem when the input is negative due to too many dead neurons.

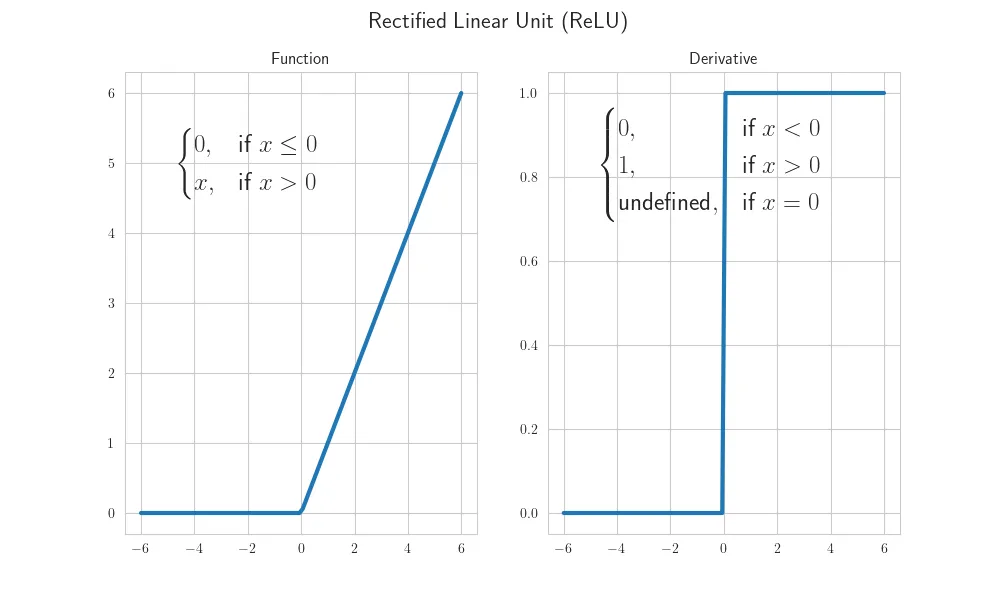
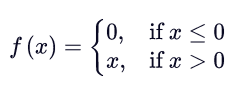


Figure 8. Visualization of ReLU function

* + The function of the forward pass





* + The function of the backward pass (derivative function)

一張含有 文字 的圖片

自動產生的描述

* Leaky ReLU

Leaky ReLU is an improved version of ReLU.

Although Leaky ReLU keeps the negative value in limited scope to diminish the number of dead neurons, sometimes the value 0.01 is still too small to update parameters.

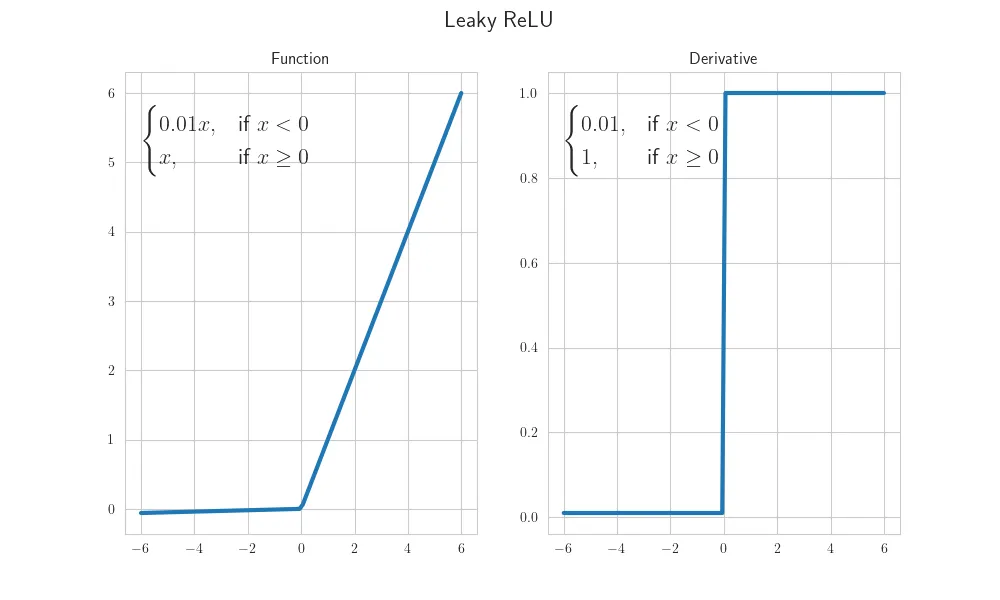


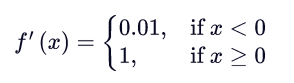
Figure 9. Visualization of Leaky ReLU function

* + The function of the forward pass

一張含有 文字 的圖片

自動產生的描述

* + The function of the backward pass (derivative function)



* ELU

The default value of hyperparameter is 1.0.

ELU has all advantages of ReLU and succeed to solve the vanishing gradient problem.

But there is no explicit evidence to prove that it is better than ReLU and Leaky ReLU.

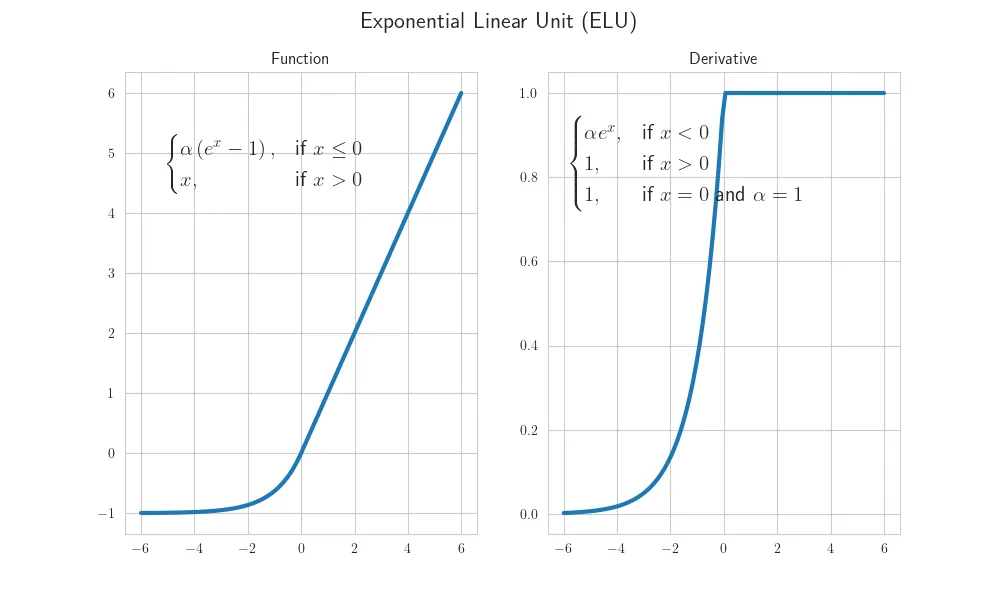


Figure 10. Visualization of ELU function

* + The function of the forward pass

一張含有 文字 的圖片

自動產生的描述

* + The function of the backward pass (derivative function)

一張含有 文字 的圖片

自動產生的描述

1. **Experimental results**

**Result:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Network** | **ReLU** | **Leaky ReLU** | **ELU** |
| EEGNet | 73.76% | 74.15% | **75.53%** |
| DeepConvNet | **84.14%** | 83.35% | 80.59% |

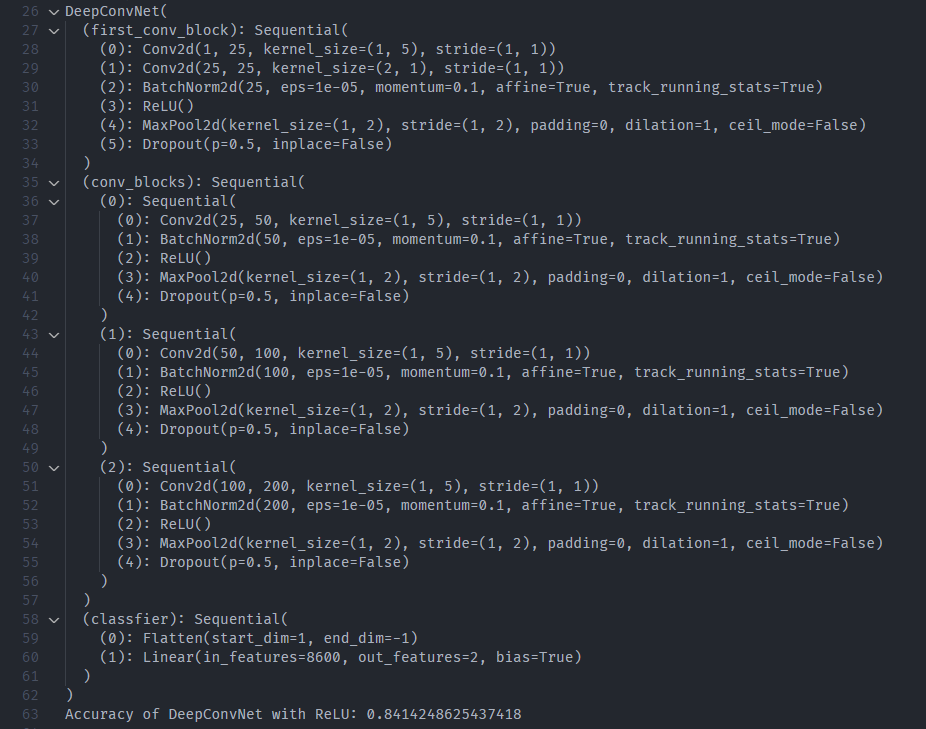
**Best model (highest testing accuracy):**

* **EEGNet (at epoch 325):**

一張含有 文字 的圖片

自動產生的描述

* **DeepConvNet (at epoch 221):**



**Comparison figures:**

Note: To use the same hyperparameters, I choose an appropriate set of hyperparameters to fit the data although the model is still overfitting. (Increasing the weight decay can solve this problem but decreasing accuracy.)

I think we need more data to train the model.

* **EEGNet with 3 activation functions:**

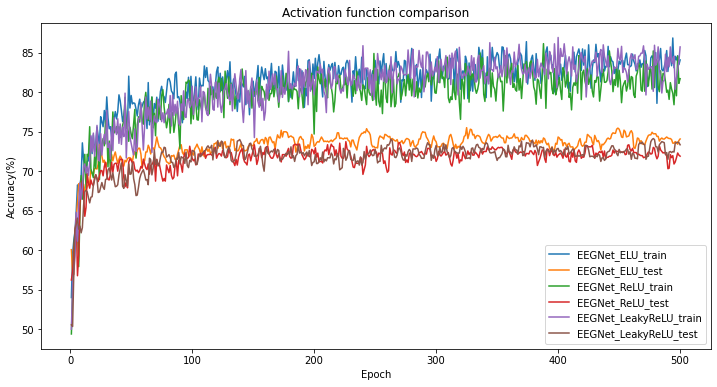


Figure 11. Accuracy comparison for EEGNet with 3 activation functions

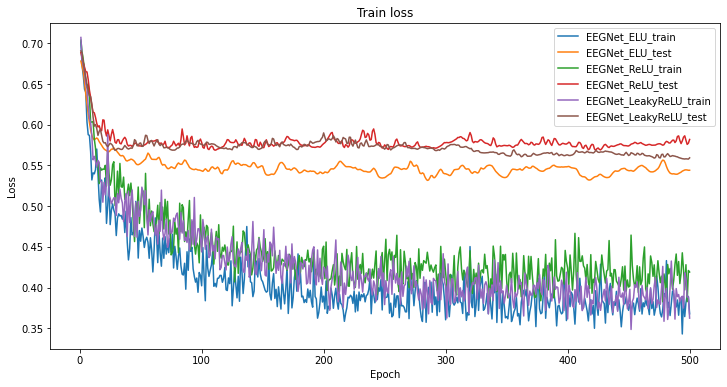


Figure 12. Loss comparison for EEGNet with 3 activation functions

* **DeepConvNet with 3 activation functions:**

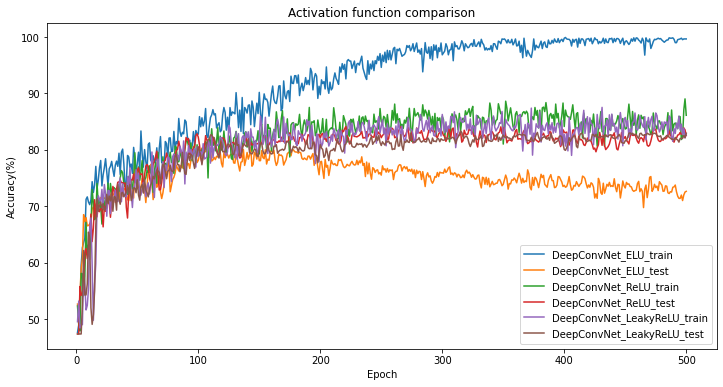


Figure 13. Accuracy comparison for DeepConvNet with 3 activation functions

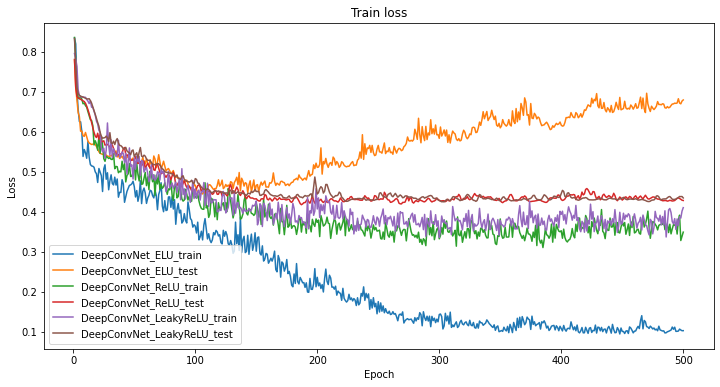


Figure 14. Loss comparison for DeepConvNet with 3 activation functions

1. **Discussion**

**What happened if I increase weight decay to diminish overfitting?**

Use the same experiment set up except for changing the weight decay.

No matter the more weight decay I add, and the more accuracy is decreasing. The test accuracy is totally the same.

It seems that the model cannot generalize in a limited number of data (1080 samples).

Possible solutions:

1. Try adding more data (**higher probability)**

Because the comparison figures in the experiment result part indicate the model can learn somethings from the training data but it cannot predict the testing data as well as the training data.

1. Study the EEGNet paper and find what the EEG signal means.
2. Change the model architecture
3. Finetune the pretrained model

* EEGNet, weight decay = 0.2

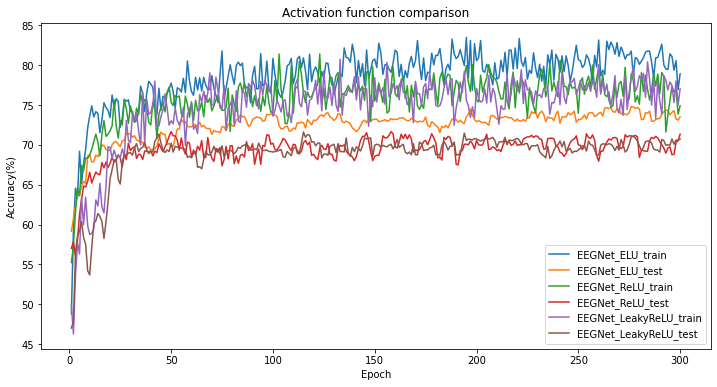


Figure 15. Accuracy comparison for EEGNet with weight decay = 0.2 with 3 activation functions

* EEGNet, weight decay = 0.3

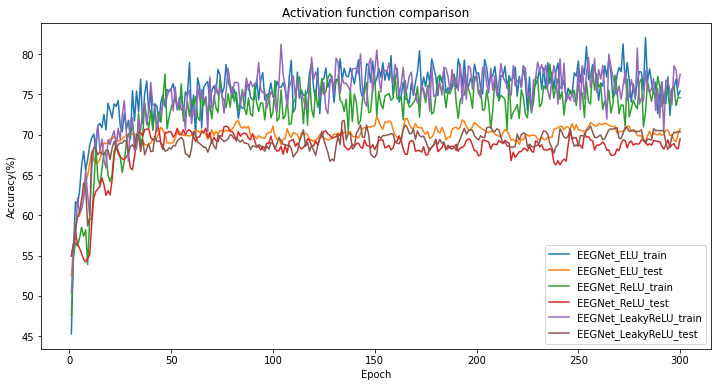


Figure 15. Accuracy comparison for EEGNet with weight decay = 0.3 with 3 activation functions

* DeepConvNet, weight decay = 0.2

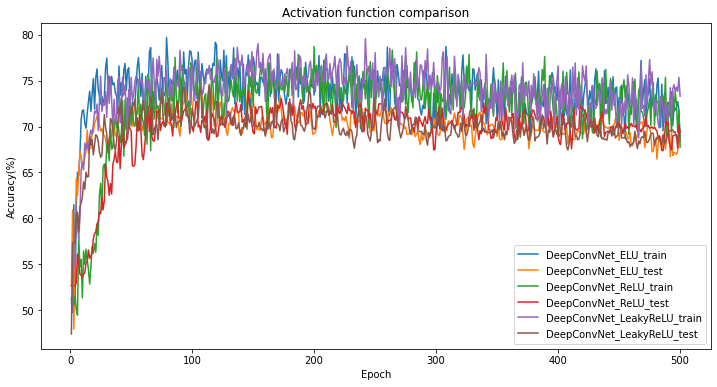


Figure 15. Accuracy comparison for DeepConvNet with weight decay = 0.2 with 3 activation functions

* DeepConvNet, weight decay = 0.3

一張含有 文字, 鉛筆 的圖片

自動產生的描述

Figure 16. Accuracy comparison for DeepConvNet with weight decay = 0.3 with 3 activation functions

1. **Reference**
   1. Activation functions: <https://ml-explained.com/blog/activation-functions-explained>
   2. PyTorch document: <https://pytorch.org/docs/stable/index.html>
   3. Lab 2 materials