Deep Learning Practice Lab 2

0856056 Yu-Wun Tseng

April 2020

1 Introduction

The goal of the lab is to implement two models, the first one is EEGNet and the other is DeepConvNet, to classify brain signals. The dataset comes from BCI competition and one data contains two channels and 750 time steps of brain signals. Each model need to be trained with three activation functions: ReLU, ELU, LeakyReLU. Finally, find the model with the highest accuracy.

2 Experiment Setups

2.1 Model Architecture

We use EEGNet and DeepConvNet for brain signal classify, both of them are implemented using Pytorch.

2.1.1 EEGNet

The EEGNet contains three convolution submodels followed by a linear classifier. The first layer contains a convolution layer followed by a batch normalize layer. And the second layer and the third layer additionally contain a activation function, a average pooling and a dropout layer. The layer parameters are set to the specifications provided by TA. The architecture is as Figure 1.

The models with different activation functions has the same epochs, optimizer and loss function. The epochs is 300, the optimizer is Adam, and the loss function is cross entropy. For model using ReLU, the batch size is set to 64 and the learning rate is set to 1e-3. For model using leaky ReLU, the batch size is set to 12 and the learning rate is set to 4e-4. For model using ELU, the batch size is set to 12 and the learning rate is set to 5e-4.

2.1.2 DeepConvNet

The DeepConvNet contains four convolution submodels followed by a linear classifier. The first layer contains two convolution layers, followed by a batch normalize layer, a activation function, a maximum pooling and a dropout layer. The remaining layers are similar to the first layer, except for the convolution layer. There's only one convolution layer in the second to fourth layer. The parameters are set to the specifications provided by TA. The architecture is as Figure 2.

The three models have the same setting, except for the learning rate of the model using ReLU. The optimizer is Adam, and the loss function is cross entropy. The batch size is 64, the learning rate is 3e-4, and the epochs is 300. Only the model using ReLU sets the learning rate to 1e-3.

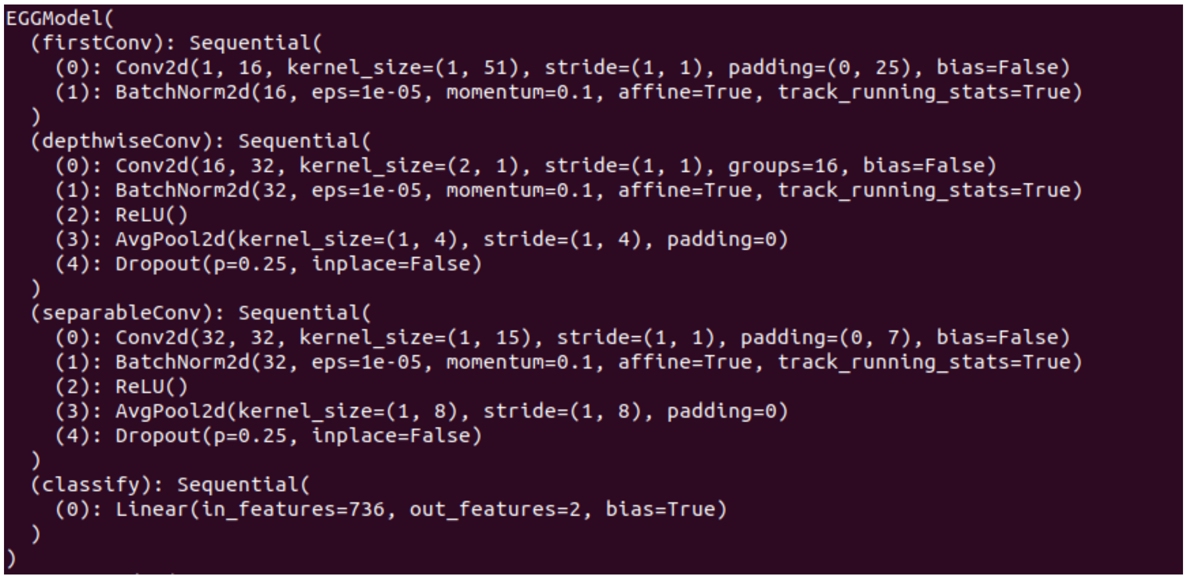
2.2 Activation Functions

In this lab, we use three different activation functions which are commonly used in CNNs, including ReLU, leakyReLU and ELU.

2.2.1 ReLU

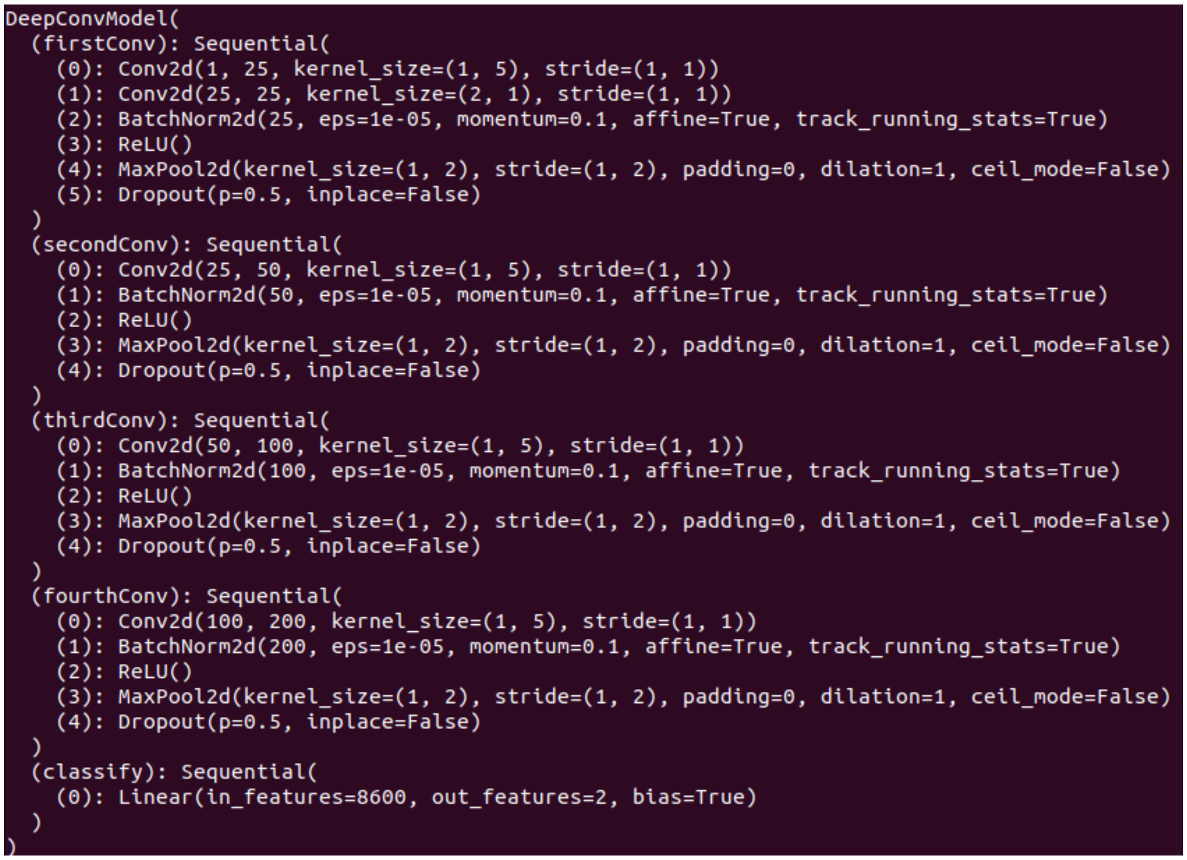
ReLU stands for rectified linear unit. When the input is greater than zero, it's output is equal to input, otherwise, it's set to be zero. The equation of ReLU is as below.

$$y = max(o, x)$$



```
EGGModel(
  (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): ReLU()
    (3): AvgPool2d(kernel_size=(1, 4), stride=(1, 4), padding=0)
    (4): Dropout(p=0.25, inplace=False)
  )
  (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): ReLU()
    (3): AvgPool2d(kernel_size=(1, 8), stride=(1, 8), padding=0)
    (4): Dropout(p=0.25, inplace=False)
  )
  (classify): Sequential(
    (0): Linear(in_features=736, out_features=2, bias=True)
  )
)
```

Figure 1: EEGNet



```
DeepConvModel(
  (firstConv): Sequential(
    (0): Conv2d(1, 25, kernel_size=(1, 5), stride=(1, 1))
    (1): Conv2d(25, 25, kernel_size=(2, 1), stride=(1, 1))
    (2): BatchNorm2d(25, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): ReLU()
    (4): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (5): Dropout(p=0.5, inplace=False)
  )
  (secondConv): Sequential(
    (0): Conv2d(25, 50, kernel_size=(1, 5), stride=(1, 1))
    (1): BatchNorm2d(50, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.5, inplace=False)
  )
  (thirdConv): Sequential(
    (0): Conv2d(50, 100, kernel_size=(1, 5), stride=(1, 1))
    (1): BatchNorm2d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.5, inplace=False)
  )
  (fourthConv): Sequential(
    (0): Conv2d(100, 200, kernel_size=(1, 5), stride=(1, 1))
    (1): BatchNorm2d(200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): MaxPool2d(kernel_size=(1, 2), stride=(1, 2), padding=0, dilation=1, ceil_mode=False)
    (4): Dropout(p=0.5, inplace=False)
  )
  (classify): Sequential(
    (0): Linear(in_features=8600, out_features=2, bias=True)
  )
)
```

Figure 2: DeepConvNet

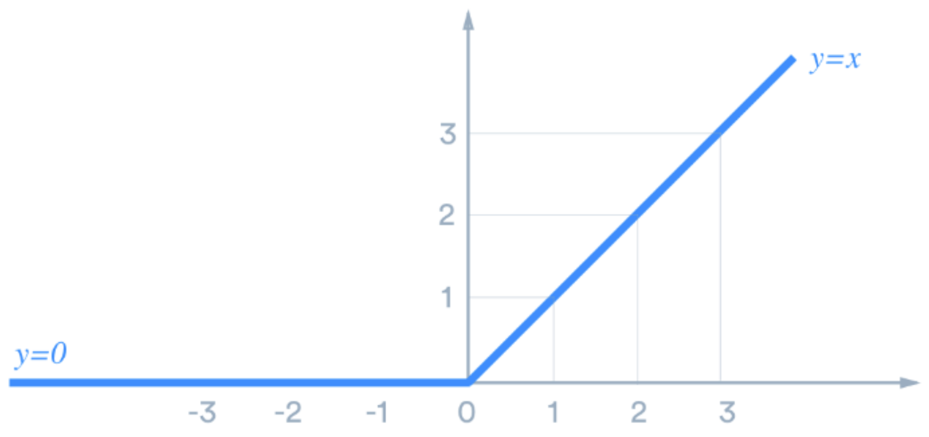


Figure 3: ReLU

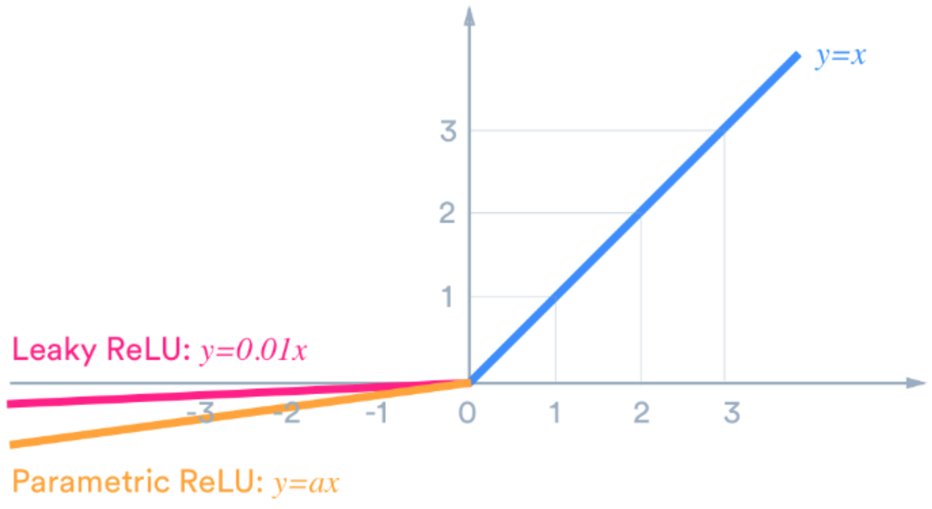


Figure 4: Leaky ReLU

The benefit of using ReLU is sparsity, which means that the output of some neurons are zero due to the negative inputs. It's similar to biological neural network, for instance, the olfactory neuron won't fire while input is a image.

The problem of using ReLU is dying ReLU, because of the zero outputs of all negative inputs. The neuron always outputs zero once the neuron stucks in the negative side. As a result, the neuron has no influence on prediction, it becomes a burdensome neuron.

2.2.2 Leaky ReLU

Leaky ReLU is a variant ReLU which has a small slope for negative inputs like Figure 4. It fixed the dying ReLU problem due to the nonzero values of the negative inputs. In addition, since the mean value of activation is more close to zero, it can speed up training. The equation of Leaky ReLU is as below.

$$y = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{if } x \leq 0 \end{cases} \tag{1}$$

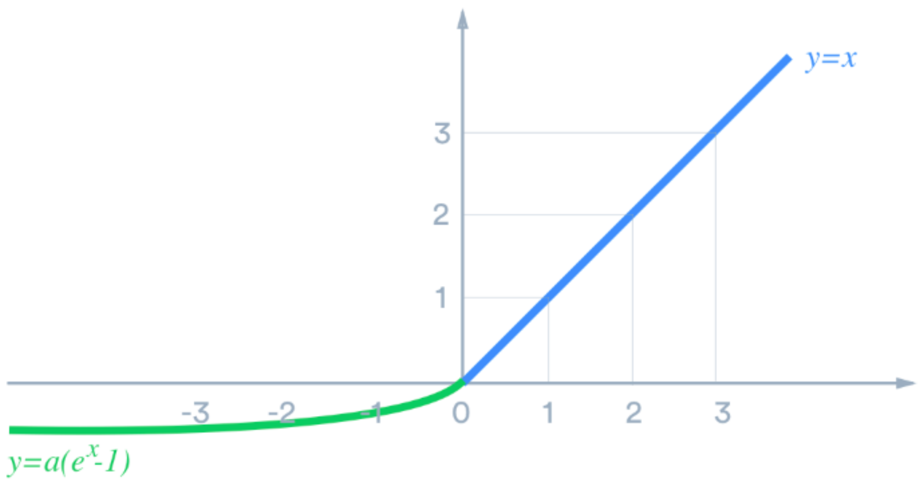


Figure 5: ELU

	ReLU	Leaky ReLU	ELU
EEGNet	**0.85277%**	**0.82962%**	**0.83333%**
DeepConvNet	0.75185%	0.76666%	0.75925%

Table 1: Comparison of each model

2.2.3 ELU

ELU is similar to Leaky ReLU which has a small slope for negative inputs. But it has a log curve like Figure 5, instead of a straight line. The equation of ELU is as below.

$$y = \alpha(e^x - 1) \tag{2}$$

3 Results

3.1 The highest accuracy

The EEG model using leaky ReLU is outperform other models with 0.85% accuracy. The batch size is 12 and the learning rate is 4e-4. As shown in Table 1, the EEG models with three different activation functions have better performance than DeepConv models. The accuracy trends of three models are shown in Figure 6, Figure 7, and Figure 8.

3.2 Comparison Figures

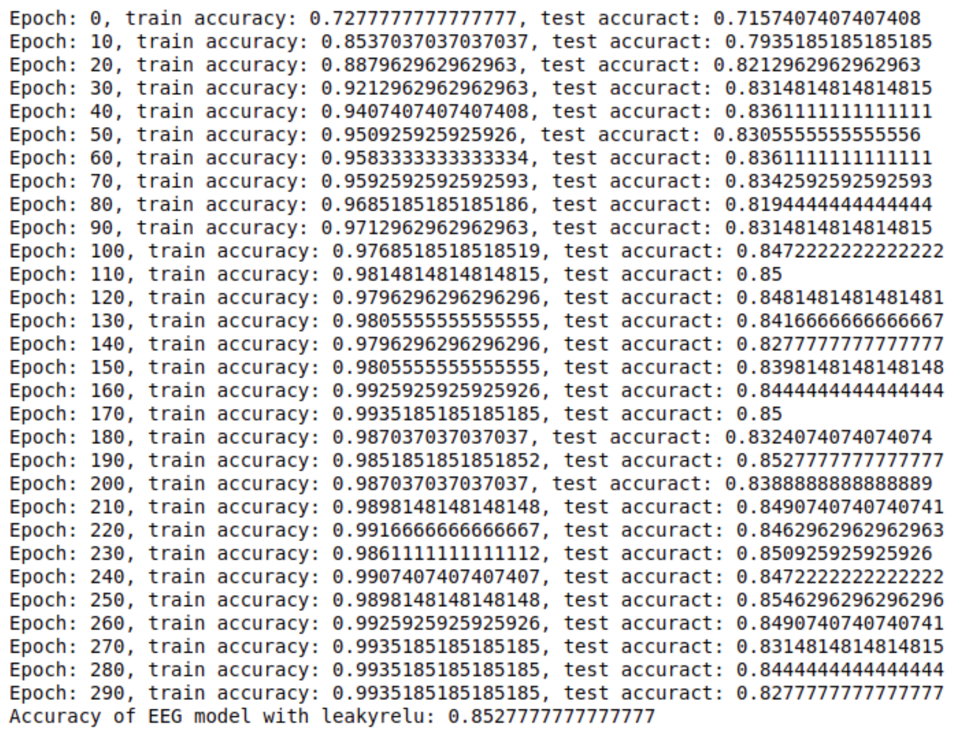
The accuracy trend of EEGNet is shown on Figure 9.

The accuracy trend of DeepConvNet is shown on Figure 10.

4 Discussion

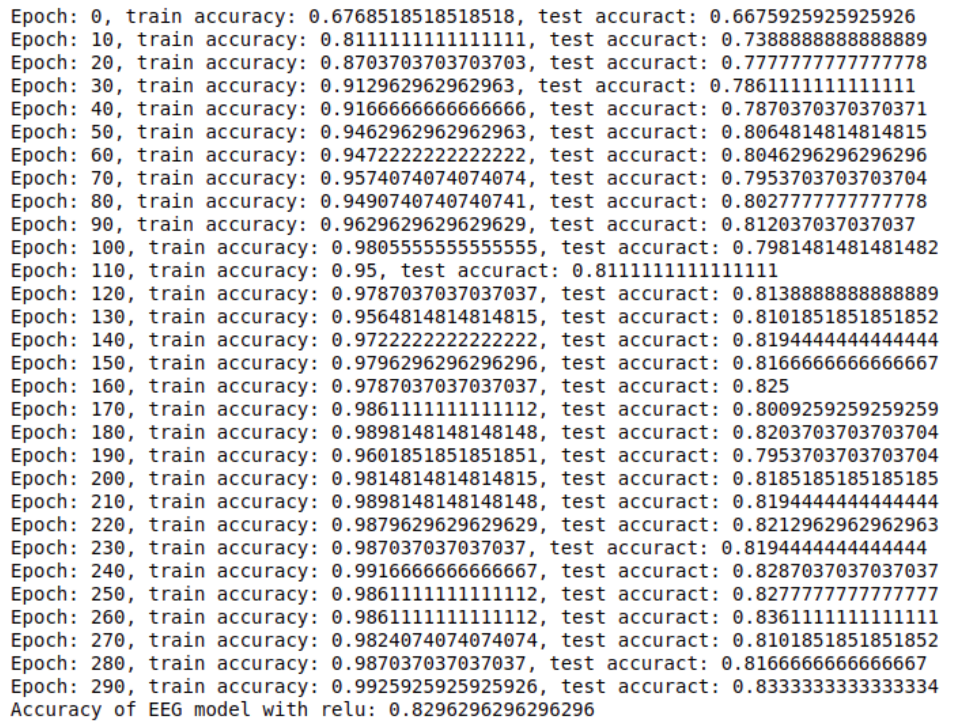
At the beginning, I didn't understand how to implement the second model since the parameters are different from EEG model. After discussing with my classmates, I realized that the parameters such as mode is applicable to Tensorflow, so I can just ignore it. In addition, the input size of the last linear layer is different from the first model because it contains different layers. I need to calculate the output size of the last convolution layer, otherwise, the linear layer will get a mismatch error.

Since the accuracy will impact the score of this lab, after the model was completed, I started to try to improve the accuracy as much as possible. But I don't have much experience of tuning the hyperparameters, thus, I decided to use a brute force method to find out the best hyperparameters.



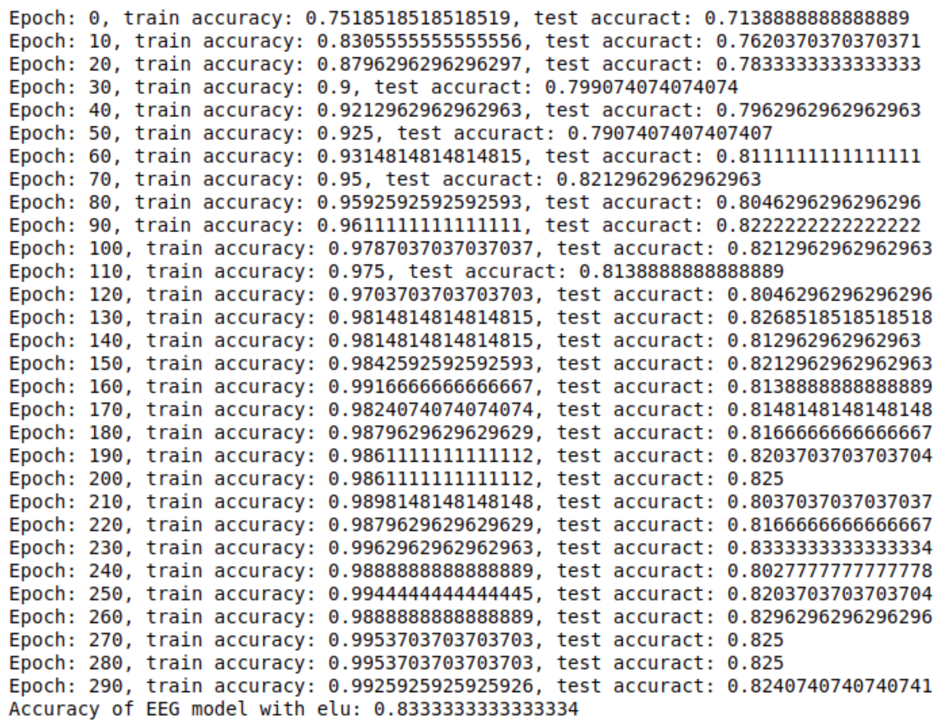
```
Epoch: 0, train accuracy: 0.7277777777777777, test accuract: 0.7157407407407408
Epoch: 10, train accuracy: 0.8537037037037037, test accuract: 0.7935185185185185
Epoch: 20, train accuracy: 0.887962962962963, test accuract: 0.8212962962962963
Epoch: 30, train accuracy: 0.9212962962962963, test accuract: 0.8314814814814815
Epoch: 40, train accuracy: 0.9407407407407408, test accuract: 0.8361111111111111
Epoch: 50, train accuracy: 0.950925925925926, test accuract: 0.8305555555555556
Epoch: 60, train accuracy: 0.9583333333333334, test accuract: 0.8361111111111111
Epoch: 70, train accuracy: 0.9592592592592593, test accuract: 0.8342592592592593
Epoch: 80, train accuracy: 0.9685185185185186, test accuract: 0.8194444444444444
Epoch: 90, train accuracy: 0.9712962962962963, test accuract: 0.8314814814814815
Epoch: 100, train accuracy: 0.9768518518518519, test accuract: 0.8472222222222222
Epoch: 110, train accuracy: 0.9814814814814815, test accuract: 0.85
Epoch: 120, train accuracy: 0.9796296296296296, test accuract: 0.8481481481481481
Epoch: 130, train accuracy: 0.9805555555555555, test accuract: 0.8416666666666667
Epoch: 140, train accuracy: 0.9796296296296296, test accuract: 0.8277777777777777
Epoch: 150, train accuracy: 0.9805555555555555, test accuract: 0.8398148148148148
Epoch: 160, train accuracy: 0.9925925925925926, test accuract: 0.8444444444444444
Epoch: 170, train accuracy: 0.9935185185185185, test accuract: 0.85
Epoch: 180, train accuracy: 0.987037037037037, test accuract: 0.8324074074074074
Epoch: 190, train accuracy: 0.9851851851851852, test accuract: 0.8527777777777777
Epoch: 200, train accuracy: 0.987037037037037, test accuract: 0.8388888888888889
Epoch: 210, train accuracy: 0.9898148148148148, test accuract: 0.8490740740740741
Epoch: 220, train accuracy: 0.9916666666666667, test accuract: 0.8462962962962963
Epoch: 230, train accuracy: 0.9861111111111112, test accuract: 0.850925925925926
Epoch: 240, train accuracy: 0.9907407407407407, test accuract: 0.8472222222222222
Epoch: 250, train accuracy: 0.9898148148148148, test accuract: 0.8546296296296296
Epoch: 260, train accuracy: 0.9925925925925926, test accuract: 0.8490740740740741
Epoch: 270, train accuracy: 0.9935185185185185, test accuract: 0.8314814814814815
Epoch: 280, train accuracy: 0.9935185185185185, test accuract: 0.8444444444444444
Epoch: 290, train accuracy: 0.9935185185185185, test accuract: 0.8277777777777777
Accuracy of EEG model with leakyrelu: 0.8527777777777777
```

Figure 6: EEGNet leaky ReLU accuracy trend



```
Epoch: 0, train accuracy: 0.6768518518518518, test accuract: 0.6675925925925926
Epoch: 10, train accuracy: 0.8111111111111111, test accuract: 0.7388888888888889
Epoch: 20, train accuracy: 0.8703703703703703, test accuract: 0.7777777777777778
Epoch: 30, train accuracy: 0.912962962962963, test accuract: 0.7861111111111111
Epoch: 40, train accuracy: 0.9166666666666666, test accuract: 0.7870370370370371
Epoch: 50, train accuracy: 0.9462962962962963, test accuract: 0.8064814814814815
Epoch: 60, train accuracy: 0.9472222222222222, test accuract: 0.8046296296296296
Epoch: 70, train accuracy: 0.9574074074074074, test accuract: 0.7953703703703704
Epoch: 80, train accuracy: 0.9490740740740741, test accuract: 0.8027777777777778
Epoch: 90, train accuracy: 0.9629629629629629, test accuract: 0.812037037037037
Epoch: 100, train accuracy: 0.9805555555555555, test accuract: 0.7981481481481482
Epoch: 110, train accuracy: 0.95, test accuract: 0.8111111111111111
Epoch: 120, train accuracy: 0.9787037037037037, test accuract: 0.8138888888888889
Epoch: 130, train accuracy: 0.9564814814814815, test accuract: 0.8101851851851852
Epoch: 140, train accuracy: 0.9722222222222222, test accuract: 0.8194444444444444
Epoch: 150, train accuracy: 0.9796296296296296, test accuract: 0.8166666666666667
Epoch: 160, train accuracy: 0.9787037037037037, test accuract: 0.825
Epoch: 170, train accuracy: 0.9861111111111112, test accuract: 0.8009259259259259
Epoch: 180, train accuracy: 0.9898148148148148, test accuract: 0.8203703703703704
Epoch: 190, train accuracy: 0.9601851851851851, test accuract: 0.7953703703703704
Epoch: 200, train accuracy: 0.9814814814814815, test accuract: 0.8185185185185185
Epoch: 210, train accuracy: 0.9898148148148148, test accuract: 0.8194444444444444
Epoch: 220, train accuracy: 0.9879629629629629, test accuract: 0.8212962962962963
Epoch: 230, train accuracy: 0.987037037037037, test accuract: 0.8194444444444444
Epoch: 240, train accuracy: 0.9916666666666667, test accuract: 0.8287037037037037
Epoch: 250, train accuracy: 0.9861111111111112, test accuract: 0.8277777777777777
Epoch: 260, train accuracy: 0.9861111111111112, test accuract: 0.8361111111111111
Epoch: 270, train accuracy: 0.9824074074074074, test accuract: 0.8101851851851852
Epoch: 280, train accuracy: 0.987037037037037, test accuract: 0.8166666666666667
Epoch: 290, train accuracy: 0.9925925925925926, test accuract: 0.8333333333333334
Accuracy of EEG model with relu: 0.8296296296296296
```

Figure 7: EEGNet ReLU traaccuracyining trend



```
Epoch: 0, train accuracy: 0.7518518518518519, test accuract: 0.7138888888888889
Epoch: 10, train accuracy: 0.8305555555555556, test accuract: 0.7620370370370371
Epoch: 20, train accuracy: 0.8796296296296297, test accuract: 0.7833333333333333
Epoch: 30, train accuracy: 0.9, test accuract: 0.799074074074074
Epoch: 40, train accuracy: 0.9212962962962963, test accuract: 0.7962962962962963
Epoch: 50, train accuracy: 0.925, test accuract: 0.7907407407407407
Epoch: 60, train accuracy: 0.9314814814814815, test accuract: 0.8111111111111111
Epoch: 70, train accuracy: 0.95, test accuract: 0.8212962962962963
Epoch: 80, train accuracy: 0.9592592592592593, test accuract: 0.8046296296296296
Epoch: 90, train accuracy: 0.9611111111111111, test accuract: 0.8222222222222222
Epoch: 100, train accuracy: 0.9787037037037037, test accuract: 0.8212962962962963
Epoch: 110, train accuracy: 0.975, test accuract: 0.8138888888888889
Epoch: 120, train accuracy: 0.9703703703703703, test accuract: 0.8046296296296296
Epoch: 130, train accuracy: 0.9814814814814815, test accuract: 0.8268518518518518
Epoch: 140, train accuracy: 0.9814814814814815, test accuract: 0.812962962962963
Epoch: 150, train accuracy: 0.9842592592592593, test accuract: 0.8212962962962963
Epoch: 160, train accuracy: 0.9916666666666667, test accuract: 0.8138888888888889
Epoch: 170, train accuracy: 0.9824074074074074, test accuract: 0.814814814814814 8
Epoch: 180, train accuracy: 0.9879629629629629, test accuract: 0.8166666666666667
Epoch: 190, train accuracy: 0.9861111111111112, test accuract: 0.8203703703703704
Epoch: 200, train accuracy: 0.9861111111111112, test accuract: 0.825
Epoch: 210, train accuracy: 0.9898148148148148, test accuract: 0.8037037037037037
Epoch: 220, train accuracy: 0.9879629629629629, test accuract: 0.8166666666666667
Epoch: 230, train accuracy: 0.9962962962962963, test accuract: 0.8333333333333334
Epoch: 240, train accuracy: 0.9888888888888889, test accuract: 0.8027777777777778
Epoch: 250, train accuracy: 0.9944444444444445, test accuract: 0.8203703703703704
Epoch: 260, train accuracy: 0.9888888888888889, test accuract: 0.8296296296296296
Epoch: 270, train accuracy: 0.9953703703703703, test accuract: 0.825
Epoch: 280, train accuracy: 0.9953703703703703, test accuract: 0.825
Epoch: 290, train accuracy: 0.9925925925925926, test accuract: 0.8240740740740741
Accuracy of EEG model with elu: 0.8333333333333334
```

Figure 8: EEGNet ELU accuracy trend

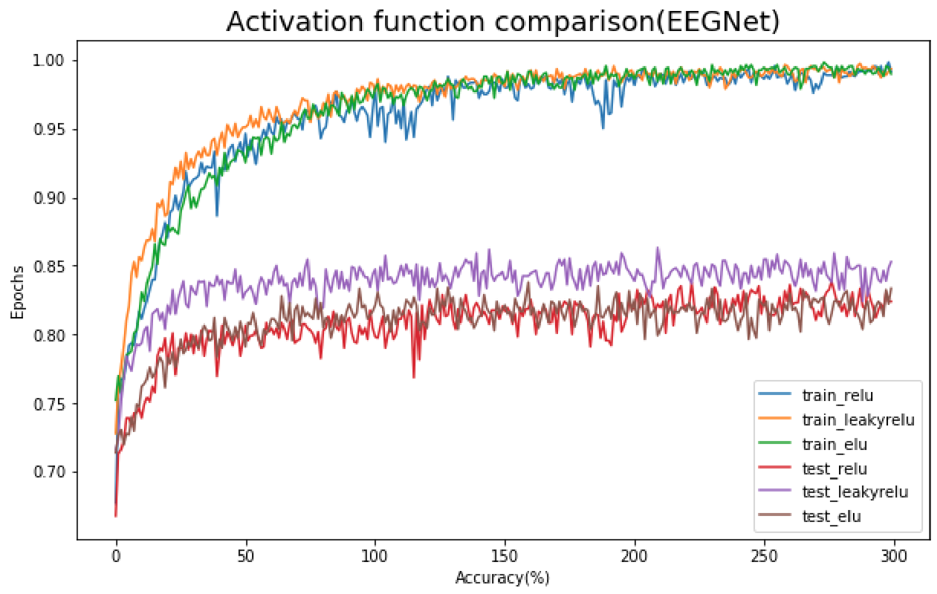


Figure 9: EEGNet accuracy trend

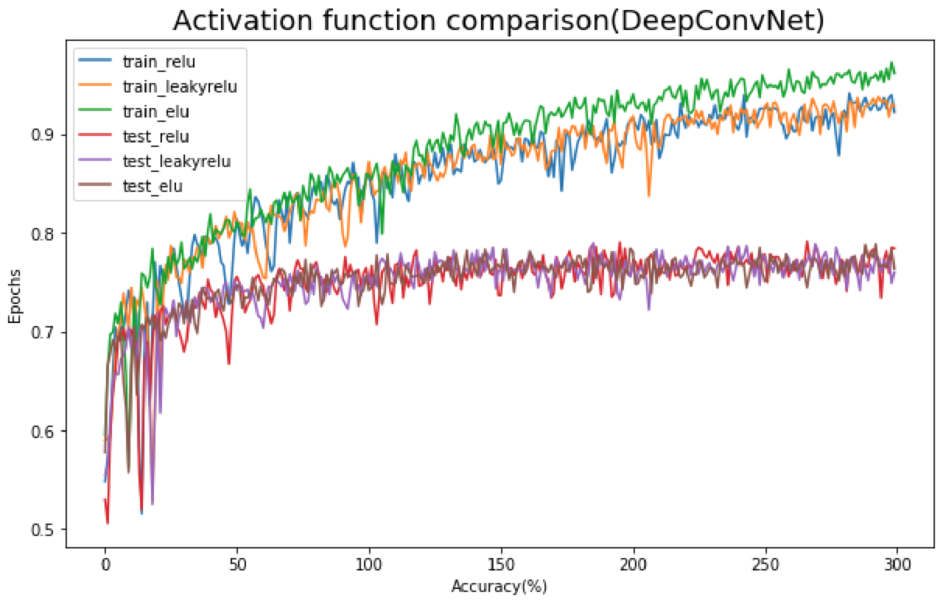


Figure 10: DeepConvNet accuracy trend

As a result, I wrote a nested for loop to find the best hyperparameters. I found that the EEGNet not only ran much faster than DeepConvNet but also had better performance. All in all, I tried many hyperparameters, the accuracy still can't exceed 0.86%.

References

[1] A Practical Guide to ReLU
`https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7`