DLP Lab3 Report

▼ 310551178 資科工碩 穆冠蓁

1. Introduction

Lab3 is to use Pytorch to build up EEGNet and DeepConvNet to classify EEG signals

2. Experiment set up

A. The detail of your model

- EEGNet

We initialize the FirstConv, DepthwiseConv, and SepearableConv in the `__init__` function and flatten in the last `classify` layer. In the `forward` function, we send the output into the next layer iteratively.

```
class EEGNet(nn.Module):
    def __init__(self, activation) -> None:
        super().__init__()

        # Firstconv
        self.firstconv = nn.Sequential(
            nn.Conv2d(
                in_channels=1,
                out_channels=16,
                kernel_size=(1, 51),
                stride=(1,1),
                padding=(0, 25),
                bias=False
            ),
            nn.BatchNorm2d(16)
        )

        # depthwiseConv
        self.depthwiseConv = nn.Sequential(
            nn.Conv2d(
                in_channels=16,
                out_channels=32,
                kernel_size=(2, 1),
                stride=(1, 1),
                groups=16,
                bias=False
            ),
            nn.BatchNorm2d(32),
            activation(),
            nn.AvgPool2d(kernel_size=(1, 4), stride=(1, 4)),
            nn.Dropout(p=0.25)
        )

        # seperableConv
        self.seperableConv = nn.Sequential(
            nn.Conv2d(
                in_channels=32,
                out_channels=32,
                kernel_size=(1, 15),
                stride=(1, 1),
                padding=(0, 7),
                bias=False
            ),
            nn.BatchNorm2d(32),
            activation(),
            nn.AvgPool2d(kernel_size=(1, 8), stride=(1, 8), padding=0),
            nn.Dropout(p=0.25)
        )

        # classify
        self.classify = nn.Sequential(
            nn.Flatten(),
            nn.Linear(in_features=736, out_features=2)
        )
```

```
    def forward(self, x):
        # Firstconv
        first_conv = self.firstconv(x)
        depth_wise_conv = self.depthwiseConv(first_conv)
        seperable_conv = self.seperableConv(depth_wise_conv)
        return self.classify(seperable_conv)
```

- DeepConvNet

We first initialize parameters of out_channels and kernel_sizes, and initialize every layer.

```
class DeepConvNet(nn.Module):
    def __init__(self, activation) -> None:
        super().__init__()
        # Parameters
        out_channels = [25, 25, 50, 100, 200]
        kernel_sizes = [(2, 1),(1, 5),(1, 5),(1, 5)]

        # DeepconvNet
        self.conv0 = nn.Sequential(
            nn.Conv2d(
                in_channels=1,
                out_channels=out_channels[0],
                kernel_size=(1,5),
                bias=False
            )
        )
        self.convs = nn.ModuleList()

        for idx in range(4):
            conv_i = nn.Sequential(
                nn.Conv2d(out_channels[idx], out_channels[idx + 1], kernel_size=kernel_sizes[idx]),
                nn.BatchNorm2d(out_channels[idx + 1]),
                activation(),
                nn.MaxPool2d(kernel_size=(1, 2)),
                nn.Dropout(p=0.5)
            )
            self.convs.append(conv_i)
        self.classify = nn.Sequential(
            nn.Flatten(),
            nn.Linear(8600, 2)
        )
    def forward(self, x):
        x = self.conv0(x)
        for conv_i in self.convs:
            x = conv_i(x)
        return self.classify(x)
```

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

3. Experimental results

A. The highest testing accuracy

We have the best accuracy and of EEGNet and DeepConvNet, with the following settings:

- EEGNet

```
---------- Model info ----------
Net type =  EEG
device name =  cuda:0
epoch =  300
batch size =  256
learning rate =  0.01
---------- Accuracy ----------
Best accuracy of training(ELU)= 0.9944444444444445
Best accuracy of testing(ELU) = 0.8138888888888889
Best accuracy of training(ReLU)= 0.9962962962962963
Best accuracy of testing(ReLU) = 0.8351851851851851
```

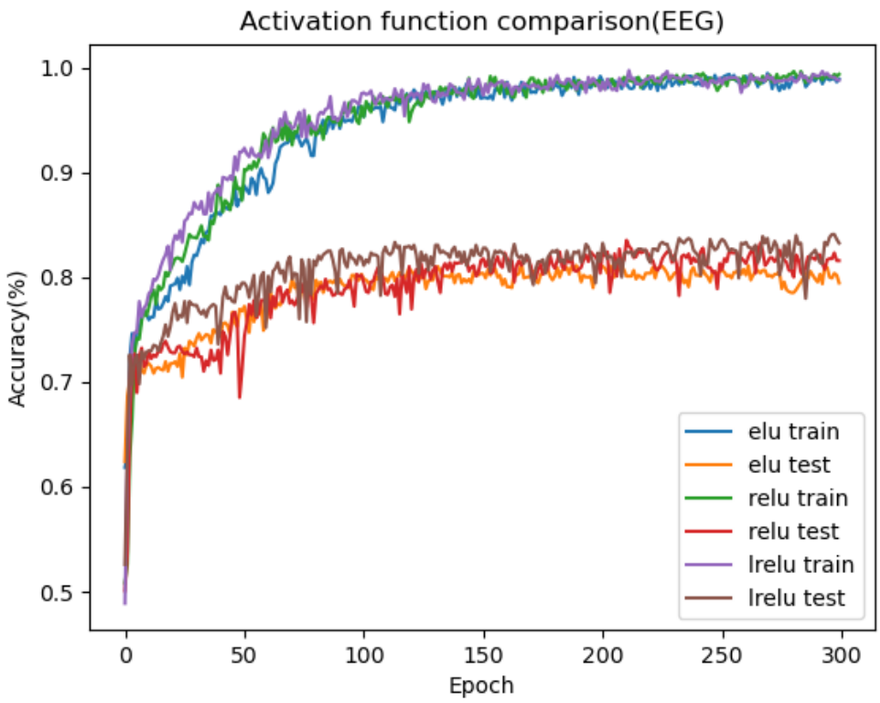
```
Best accuracy of training(Leaky ReLU)= 0.9972222222222222
Best accuracy of testing(Leaky ReLU) = 0.8407407407407408
```

- DeepConvNet

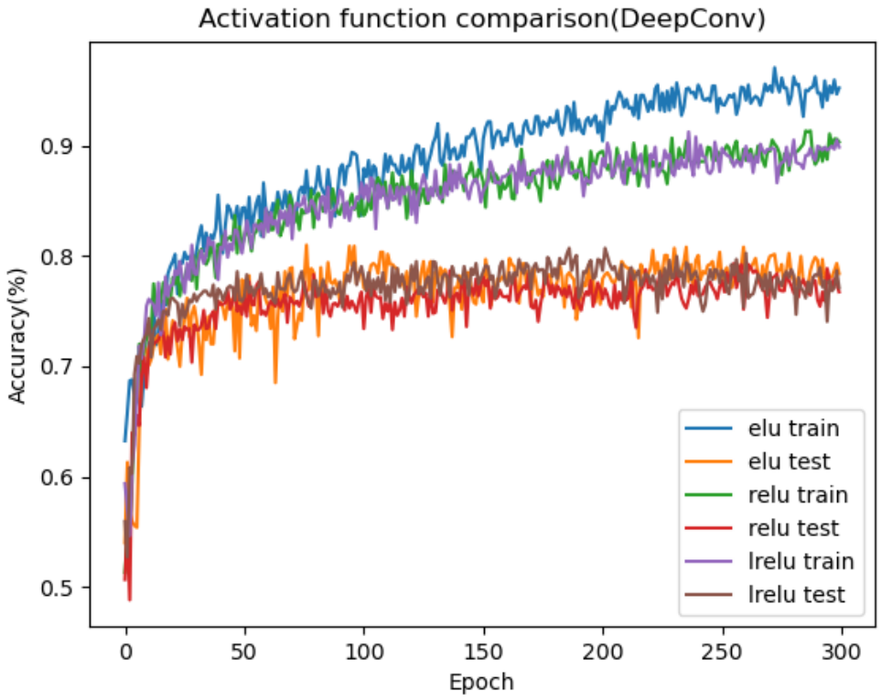
```
---------- Model info ----------
device name =  cuda:1
epoch =  300
batch size =  32
learning rate =  0.01
---------- Accuracy ----------
Best accuracy of training(ELU)= 0.9712962962962963
Best accuracy of testing(ELU) = 0.8101851851851852
Best accuracy of training(ReLU)= 0.9138888888888889
Best accuracy of testing(ReLU) = 0.7935185185185185
Best accuracy of training(Leaky ReLU)= 0.912962962962963
Best accuracy of testing(Leaky ReLU) = 0.8074074074074075
```

B. Comparison figures

- EEGNet



- DeepConvNet

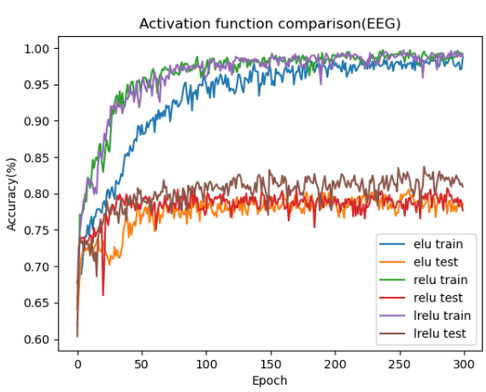


4. Discussion

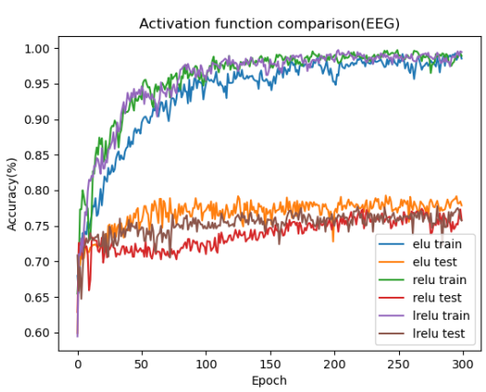
A. Batch size comparison

- EEGNet(optimizer=adam, epoch=300)

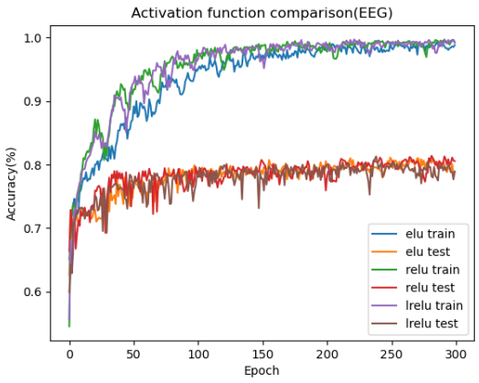
Batch size	32	64	128	256	512
training accuracy(ELU)	0.99	0.99	0.99	0.99	0.99
test accuracy(ELU)	0.81	0.79	0.81	0.81	**0.82**
training accuracy(ReLU)	0.99	0.99	0.99	0.99	0.99
test accuracy(ReLU)	0.81	0.77	0.81	**0.83**	**0.83**
training accuracy(Leaky ReLU)	0.99	0.99	0.99	0.99	0.99
test accuracy(Leaky ReLU)	**0.84**	0.78	0.81	**0.84**	0.82



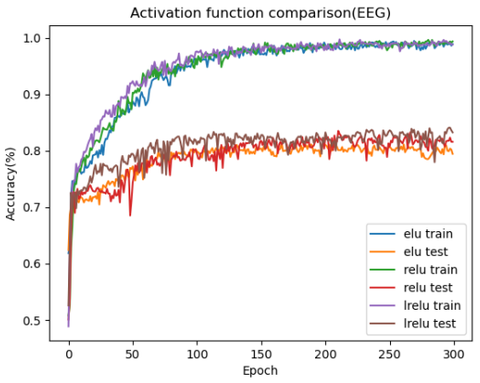
batch_size = 32



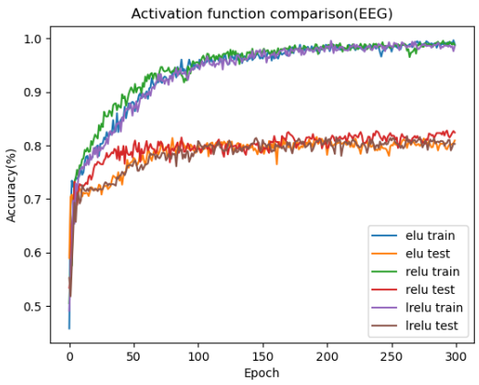
batch_size = 64



batch_size = 128



batch_size = 256

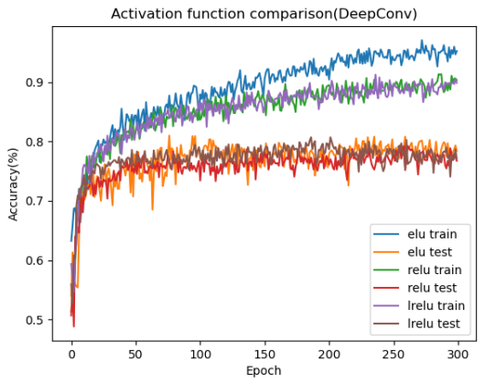


batch_size = 512

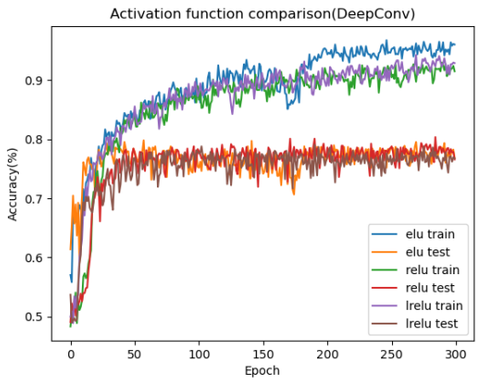
From the accuracy comparison table, we can see that we get two highest testing score when the batch size is 256 and 512. In the activation function comparison figure, we can see while we increase our batch size the gap between testing and training accuracy is smaller.

- DeepConvNet(optimizer=adam, epoch=300)

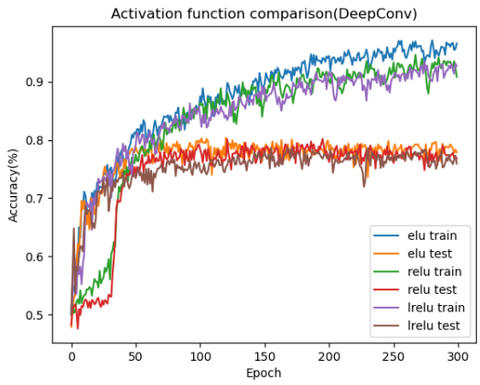
Batch size	32	64	128	256	512
training accuracy(ELU)	0.97	0.97	0.97	0.97	0.96
test accuracy(ELU)	**0.81**	0.80	0.80	0.79	0.80
training accuracy(ReLU)	0.91	0.93	0.95	0.94	0.91
test accuracy(ReLU)	0.79	**0.80**	**0.80**	0.76	0.75
training accuracy(Leaky ReLU)	0.91	0.94	0.94	0.94	0.94
test accuracy(Leaky ReLU)	**0.81**	0.79	0.79	0.78	0.79



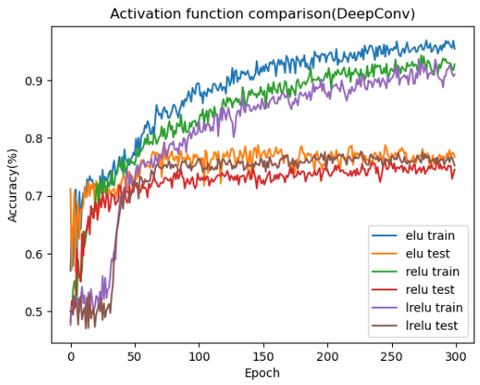
batch_size = 32



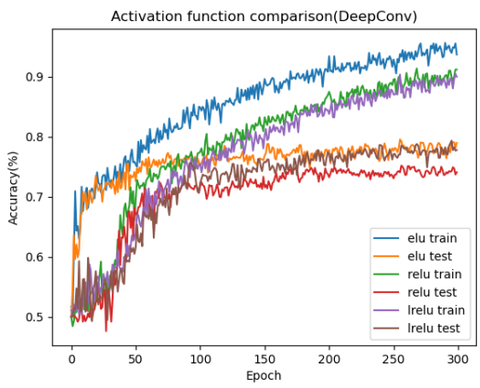
batch_size = 64



batch_size = 128



batch_size = 256



batch_size = 512

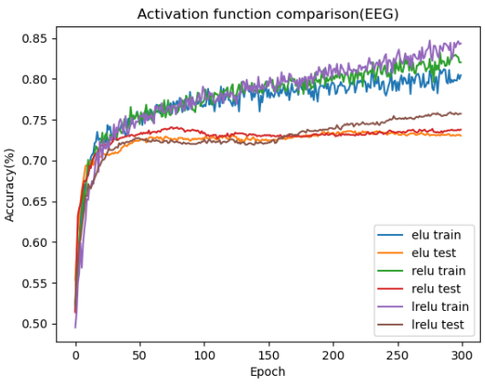
From the accuracy comparison table, we can a different phenomenon from EEGNet, it is while the batch size is low to 32, we get two best accuracy in the experiment. And from the activation comparison figure, while the batch size is larger than 128, the accuracy is lower to 50% in the first 50 epoch than the small batch size.

B. Optimizer comparison

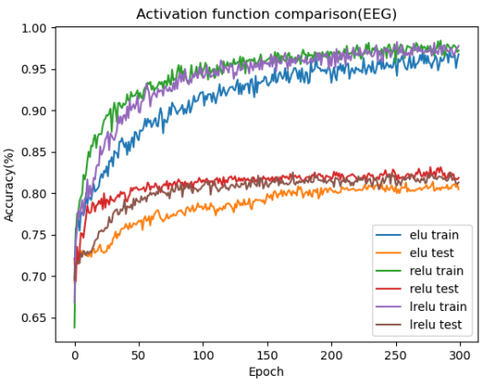
- EEGNet(batch_size = 64, epoch=300)

Optimizer	adadelta	adagrad	adam	adamax	adamw
training accuracy(ELU)	0.81	0.97	0.99	1.0	0.99
test accuracy(ELU)	0.74	0.81	0.80	**0.82**	0.79
training accuracy(ReLU)	0.83	0.98	1.0	1.0	1.0

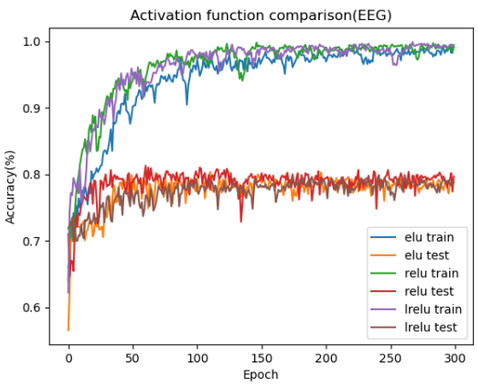
test accuracy(ReLU)	0.74	0.83	0.81	**0.84**	0.81
training accuracy(Leaky ReLU)	0.85	0.98	1.0	1.0	1.0
test accuracy(Leaky ReLU)	0.76	**0.83**	0.8	0.82	0.81



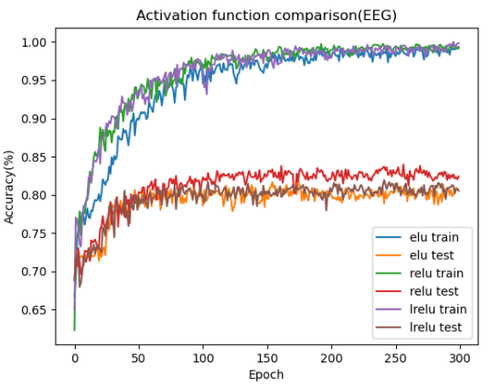
activation=adadelta



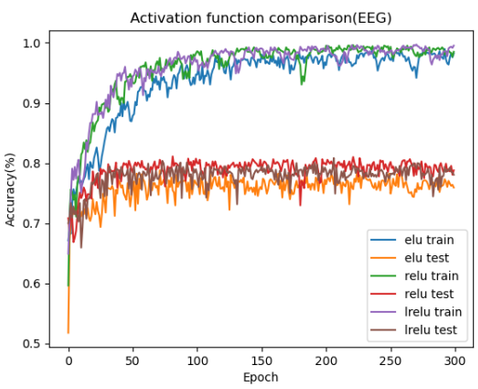
activation=adagrad



activation=adam



activation=adamax

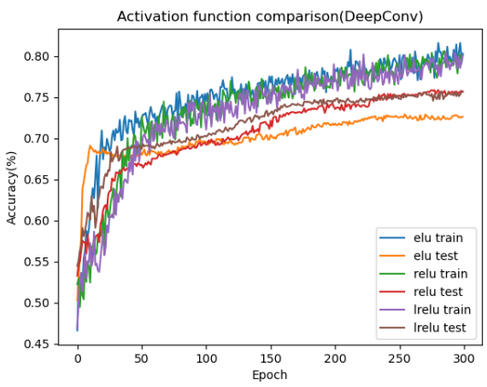


activation=adamw

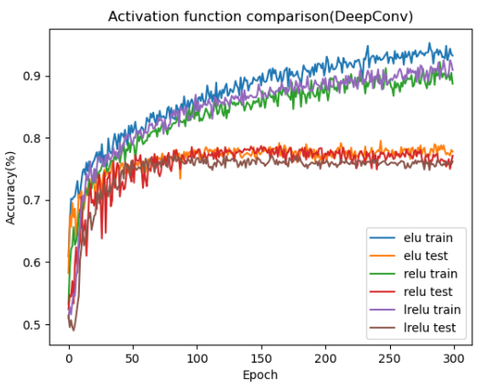
From the accuracy comparison table, we can see that adamax and adagrad perform the best in EEGNet. In the activation function comparison figure, we can see that adadelta has the smallest gap between training and testing but other optimizer has a large gap compare to the curve in DeepConvnet.

- DeepConvNet(optimizer=adam, epoch=300)

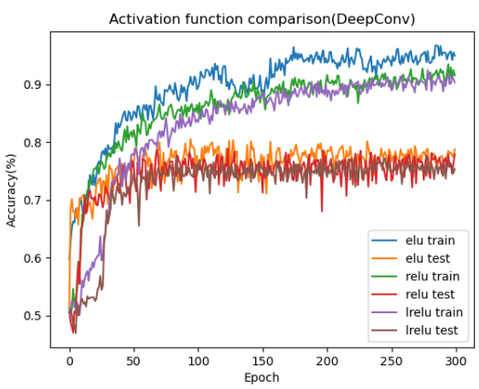
Optimizer	adadelta	adagrad	adam	adamax	adamw
training accuracy(ELU)	0.82	0.95	0.97	0.98	0.97
test accuracy(ELU)	0.73	0.80	**0.81**	0.80	**0.81**
training accuracy(ReLU)	0.81	0.92	0.94	0.97	0.94
test accuracy(ReLU)	0.76	0.78	0.79	0.79	**0.80**
training accuracy(Leaky ReLU)	0.80	0.93	0.93	0.97	0.93
test accuracy(Leaky ReLU)	0.76	0.77	0.78	0.78	**0.80**



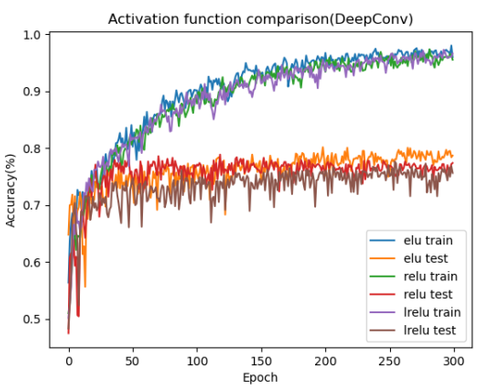
activation=adadelta



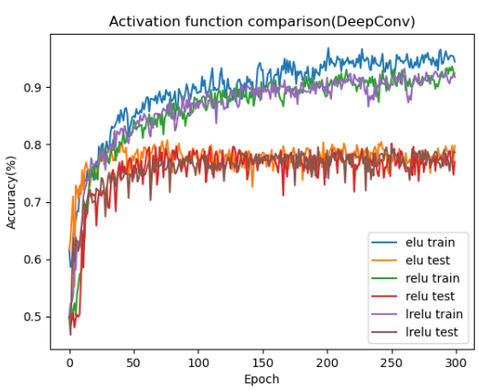
activation=adagrad



activation=adam



activation=adamax



activation=adamw

From the accuracy comparison table, we can see that **adamw** perform the best. And from the activation comparison figure, we can see that the testing curve of **adadelta** optimizer has the closest performance to the training curve.