Report Spec

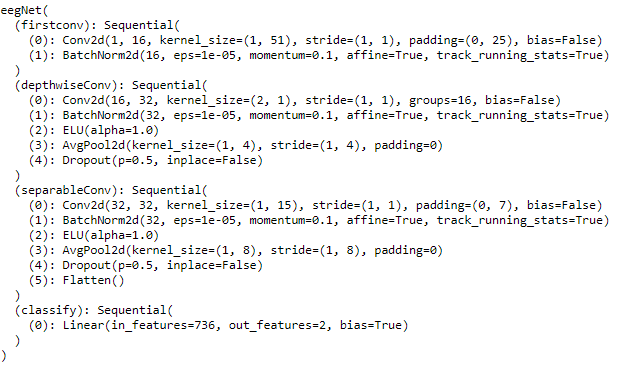
1. **Introduction**

在LAB2中要將初步學掉的CNN用法透過pytorch學以致用，譬如model每層Layer的架設與初值設定，另外必須要充分了解到convolution的作用，是利用Parameter sharing、Local connectivity、Equivalent representation ...等特性來達到CNN最大的效果。

另外於本次實作中，也是更加熟悉了pytorch之使用，並且了解到cuda平行運算的強大，順帶一提一些基本的debug還是需要化解的。另外由於本次作業有根據acc的準確度來判定分數，所以從accuracy與loss中找到最佳的hyperparmeter設定，也是一大挑戰。

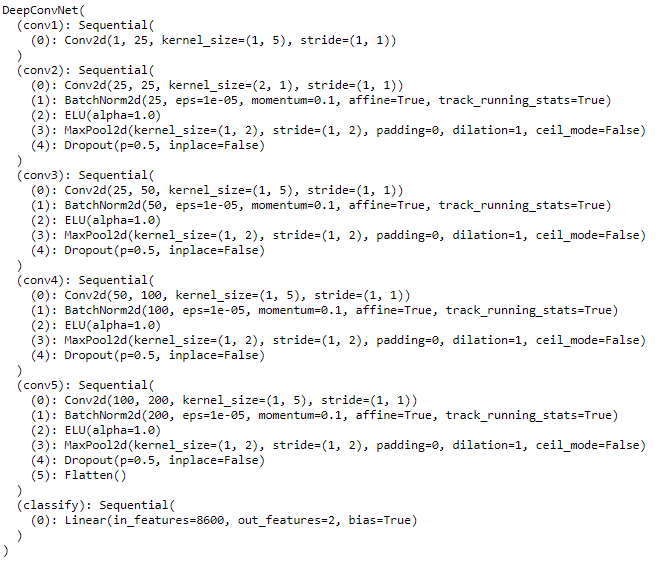
1. **Experiment setups**
   * 1. The detail of your model

* EEGNet



|  |  |
| --- | --- |
|  | 其實Convolution的架構不外乎就是這些   1. 利用Convolution的特性，將原本的input依照kernel size分成對應的feature map。 2. 根據filter(neurons)數量切割出不同的channel，再by channel 做Normalization為的是要使gradient descent更容易 3. 加上activation function為的是突破linear的限制 4. 丟入Pooling根據kernel size取最佳特徵值 |

* DeepConvNet



|  |  |
| --- | --- |
|  | 1. 最後利用flatten() 打平channel、與softmax等之類的activation function，來總結一個推測的結果出來。 2. 補充1 : 另外stride, padding等應用在conv上的作用這邊簡短提個大概，就是kernel size 移動的距離、與空位補0的意思。 3. 還有dropout能降低overfit，另外有趣的是pooling其實不是必要，在某些講求精度的model其實是做pooling會降低performance的。(full convonlution) |

* + 1. Explain the activation function (ReLU, Leaky ReLU, ELU)
* **ReLU**

|  |  |
| --- | --- |
|  |  |

Relu 為 nonlinear function，他非常簡單去設定，不會梯度消失問題。

但也有個重大的缺點；非常脆弱，當x < 0時 梯度為0。導致不對任何值有反應。(非梯度消失問題)

* **Leaky ReLU**

|  |  |
| --- | --- |
|  |  |

為了解決 dead relu現象 使用了一種固定斜率的方法使得 x<0時梯度也不會變為0

* **ELU**

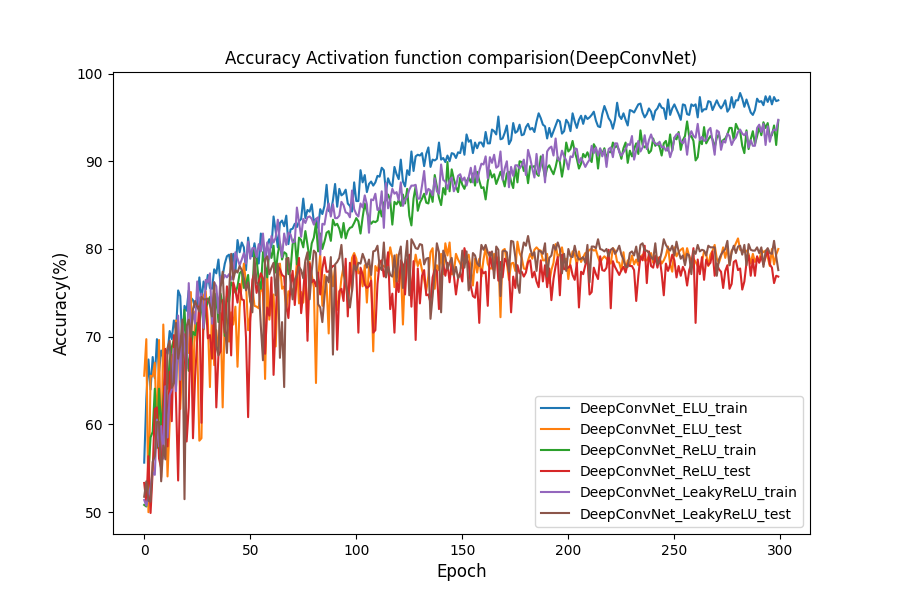
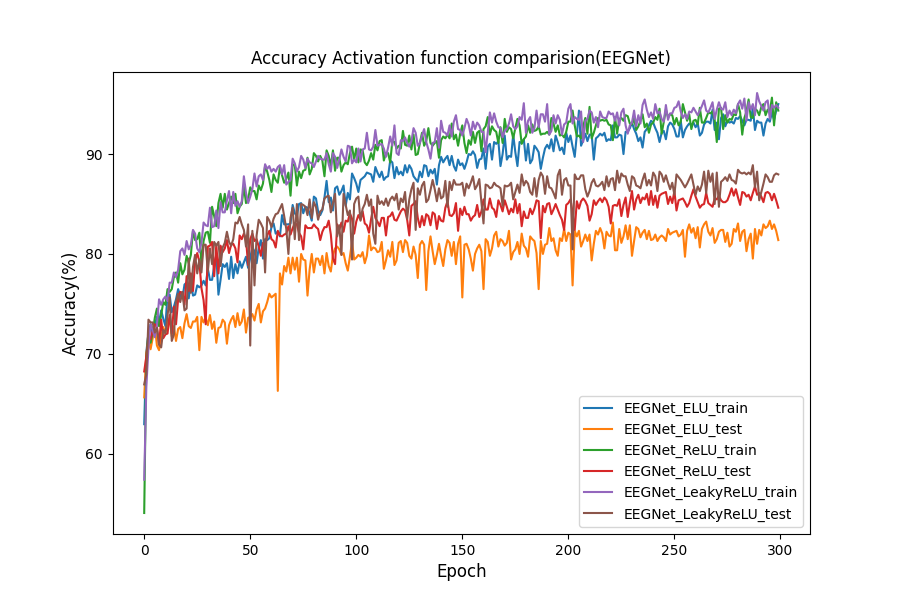
|  |  |
| --- | --- |
|  |  |

也解決的dead relu的現象，利用**指數**來當基準計算

1. **Experimental results**

A. The highest testing accuracy

- Screenshot with two models

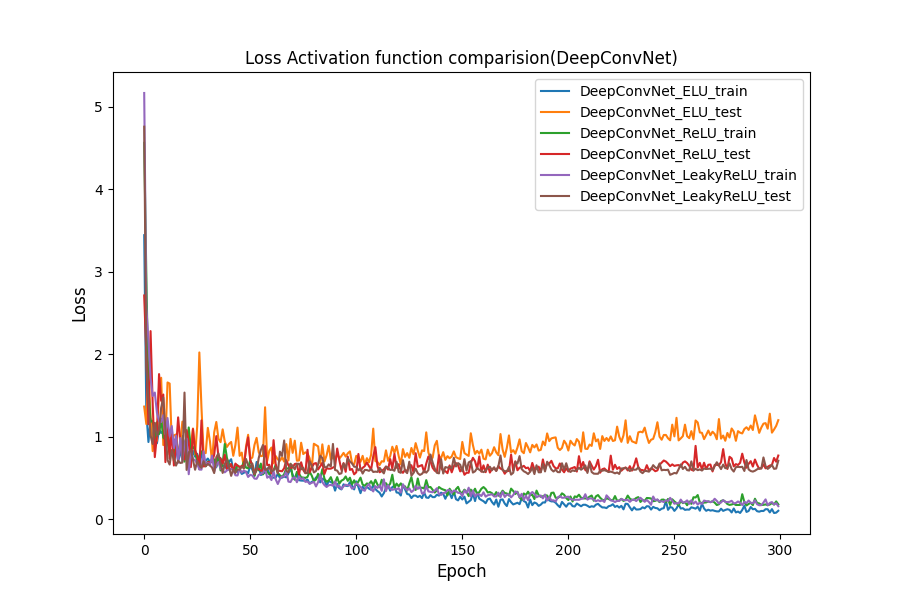
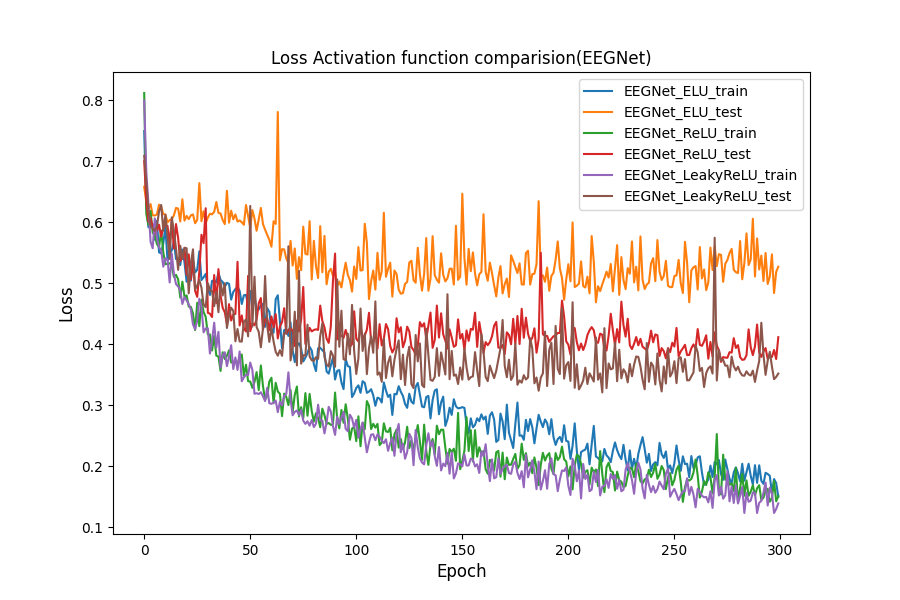


**Best accurancy**



- Anything you want to present

loss value



值得一提的是 :

這邊在算acc時只要除總數1080即可得到accuracy的%

但是loss不一樣，會因為batch size的緣故，需要除不同的值。

B. Comparison Figures

下面是我只調整lr所帶來不同的結果。

|  |  |
| --- | --- |
|  |  |
|  |  |

發現在optimizer 為 Adam時 :

learning rate太大並無好的結果且acc上下浮動劇烈。(無法進入opt區間)

lerrning rate調小果然浮動劇烈消失，且穩定上升

1. **Discussion**

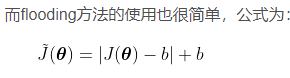
A. Anything you want to share

在測資時有遇到一個有趣的現象，照理來說當loss下降時，通常代表著accuracy的上升。但卻發生了loss上升accuracy也上升的情況 如圖

|  |  |
| --- | --- |
|  |  |

很明顯的由橘、棕色的線可以看出左圖acc為不變且些微上升的情況，右圖loss橘、棕線卻有往上的趨勢。

上網查證後發現可以使用一種flooding的正規化方法來避免此種反常理的情況發生，以下為公式。



其中J代表原始的目標函數，b則是一個hyperparmeter，為了是要設定訓練損失的下限。

下面是我套用後的成果。雖然整體浮動大但是少了loss acc同方向的反常理情況

|  |  |
| --- | --- |
|  |  |

另外由於我b值是按照上上圖抓個0.3附近，相信再摸索一下能得到更漂亮的數值