LAB2. EEG classification

陽明 大一大二不分系二年級 10612012 張凱博

1. Introduction

Convolutionary Neuron Network 簡稱CNN，中文名稱為「卷積類神經網絡」，是受到人類大腦視覺皮層在識別物體時是如何工作的方法，所引發的方法，所引發的一系列模型。

CNN這類的類神經網絡被視為，被視為特徵搜尋引擎，藉由輸入hyperparameter: filter size、padding size和stride後，進行卷積(convolution)，前面幾層擷取低階特徵，最後會層層結合形成高階特徵，來建立所謂的feature hierarchy。 在每一次convolution完後就會接一個pooling layer，對一塊塊數值化過後的data不重疊地取他們的最大或是平均運算，最後進行dropout避免model在training時過度依賴某一個特定的neuron。

這次lab的主題是以Electroencephalogram(腦電圖)為data建構CNN model進行EEG的分類，屬於影像識別的一種。

1. Experiment set up
2. The detail of model
3. EEGNet

在我的EEGNet model裡，共有3層convolutional layer，每層model利用torch.nn.sequential()函數建置。

1. 第一層layer (firstconv)

用nn.Conv2d()進行convolution，input channel為1，output channel為16，kernel\_size長寬為1, 51，填補模式為same padding，接下來用BatchNorm2d進行normalization。

1. 第二層layer (depthwiseConv)

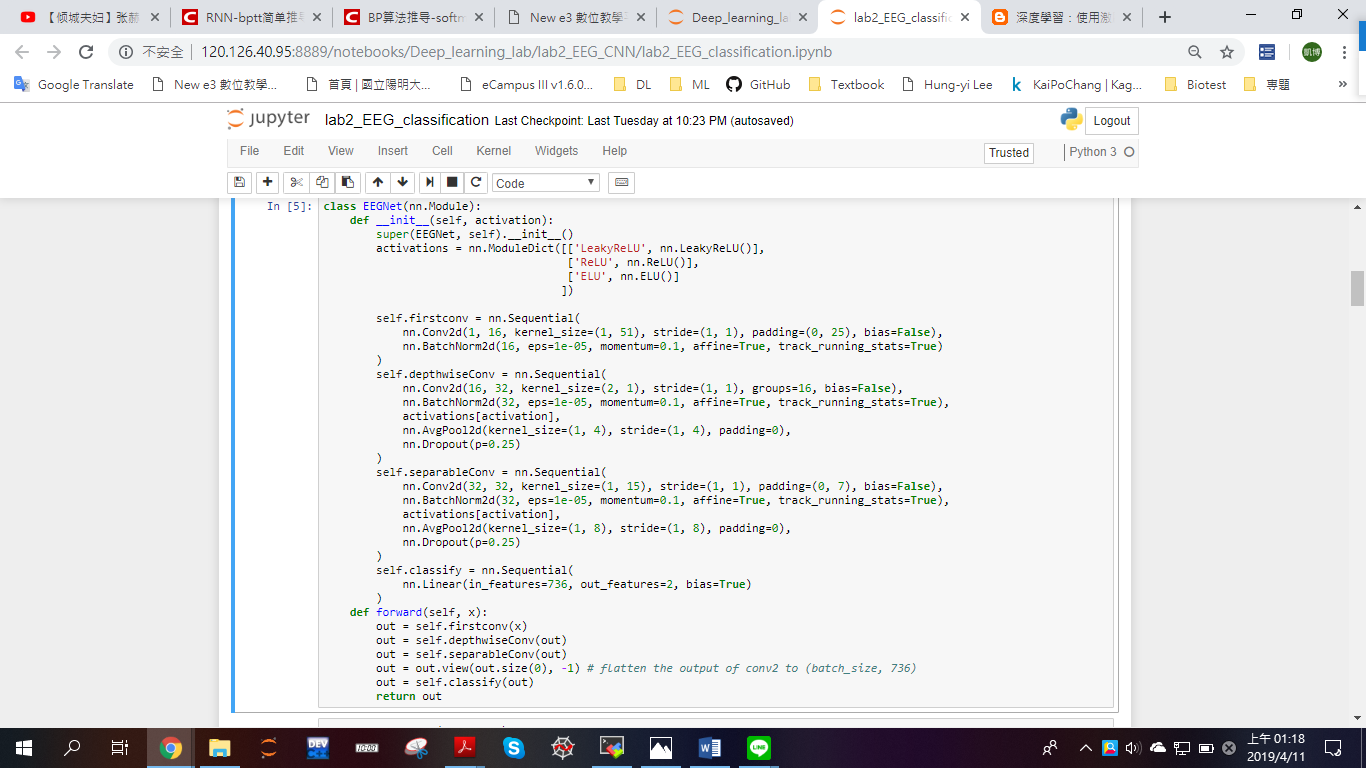
用nn.Conv2d()進行convolution，input channel為16，output channel為32，kernel\_size長寬為2, 1，填補模式為same padding，接下來用BatchNorm2d進行normalization，再用activation function引入非線性，再用average pooling layer做subsampling，最後在用0.25的機率隨機dropout neuron。

1. 第三層layer (separableConv)

用nn.Conv2d()進行convolution，input channel和output channel都是32，kernel\_size長寬為1, 15，填補模式為same padding，用BatchNorm2d進行normalization，再用activation function引入非線性，再用average pooling layer做subsampling，最後在用0.25的機率隨機dropout neuron。

1. 第四層layer (classify)

將736個input feature用線性轉換成2種類別(out put feature)



1. DeepConvNet

在我的EEGNet model裡，共有3層convolutional layer，每層model利用torch.nn.sequential()函數建置。

(1) 第一層layer (firstconv)

用2層nn.Conv2d()進行convolution，input channel為1，output channel為25，kernel\_size長寬為1, 51，填補模式為valid padding，用BatchNorm2d進行normalization，再用activation function引入非線性，再用maxium pooling layer做subsampling，最後在用0. 5的機率隨機dropout neuron。

(2)第二層layer (secondlayer)

用nn.Conv2d()進行convolution，input channel為25，output channel為50，kernel\_size長寬為1, 5，填補模式為valid padding，接下來用BatchNorm2d進行normalization，再用activation function引入非線性，再用average pooling layer做subsampling，最後在用0. 5的機率隨機dropout neuron。

(3)第三層layer (thirdlayer)

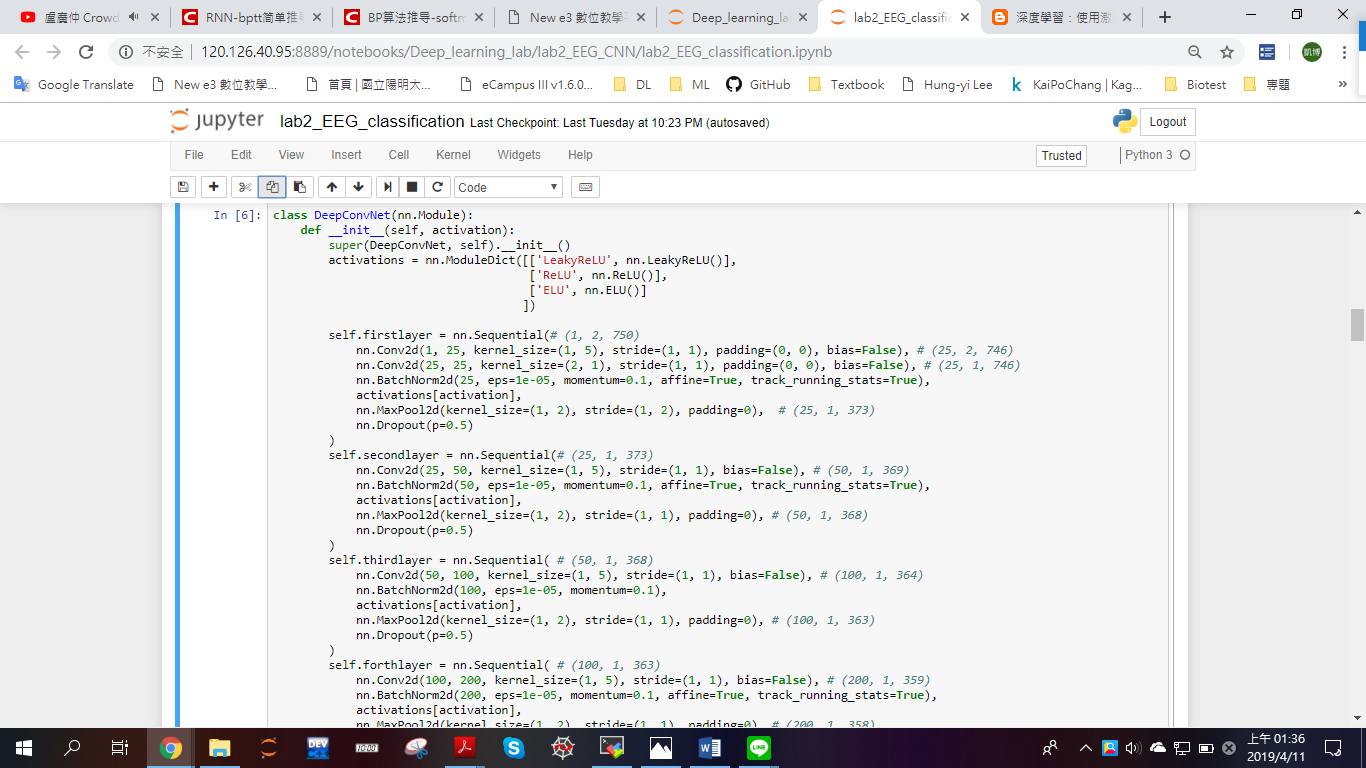
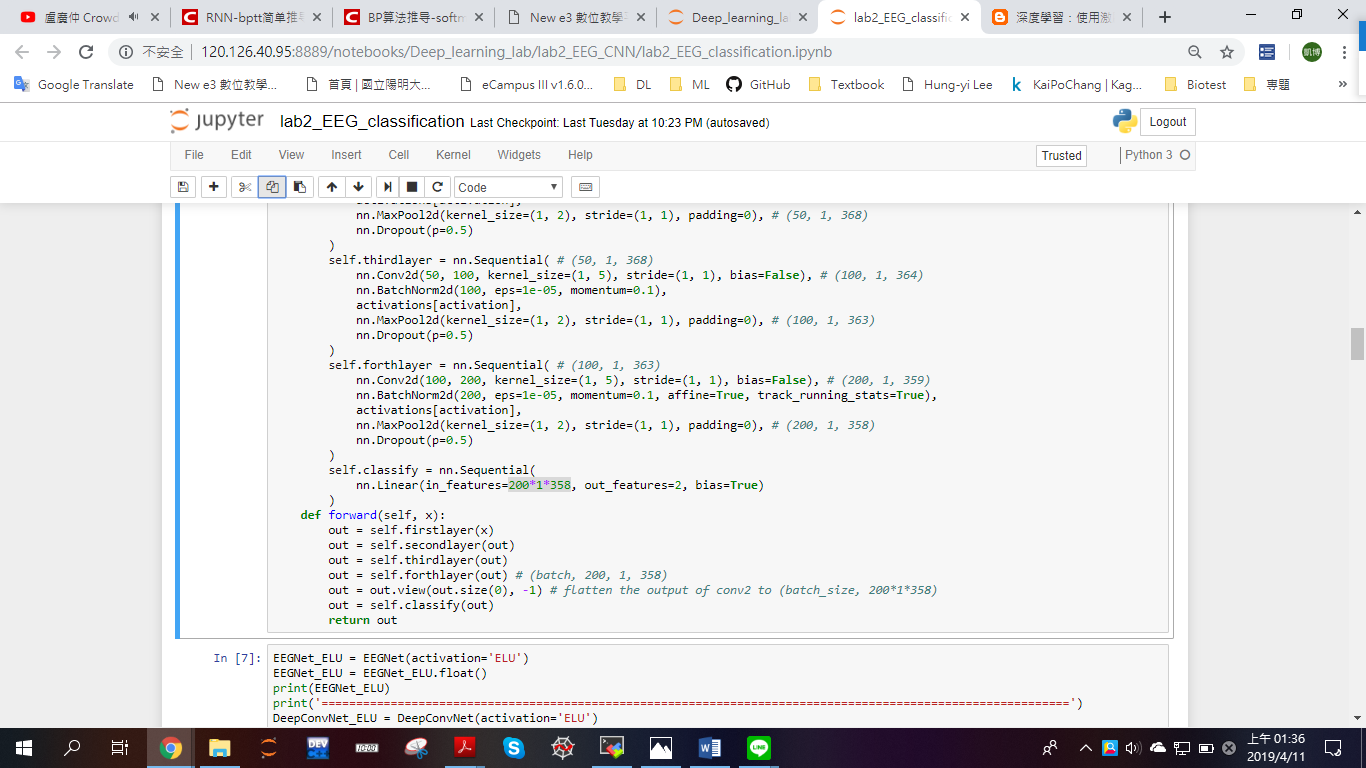
用nn.Conv2d()進行convolution，input channel為50，output channel為100，kernel\_size長寬為1, 5，填補模式為valid padding，接下來用BatchNorm2d進行normalization，再用activation function引入非線性，再用average pooling layer做subsampling，最後在用0.5的機率隨機dropout neuron。

1. 第四層layer (forthlayer)

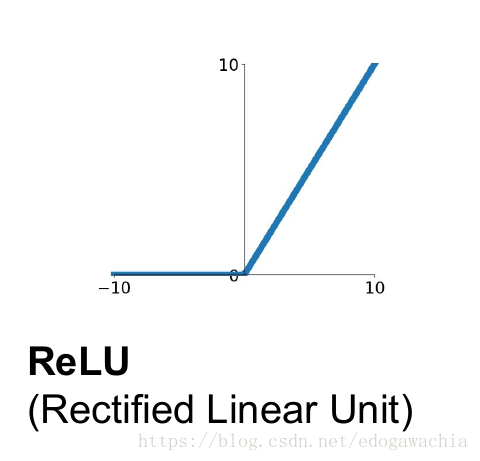
用nn.Conv2d()進行convolution，input channel為100，output channel為200，kernel\_size長寬為1, 5，填補模式為valid padding，接下來用BatchNorm2d進行normalization，再用activation function引入非線性，再用average pooling layer做subsampling，最後在用0.5的機率隨機dropout neuron。

1. 第四層layer (classify)

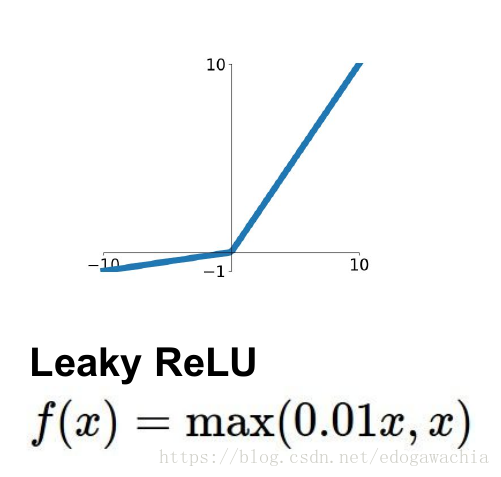
將200\*1\*358個input feature用線性轉換成2種類別(out put feature)

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

在CNN中常用的，對負數直接置為零，對正數利用f(x) = x原樣輸出，在初始化的時候通常會設稍大於零的數字以免用random的參數落在負數區域而不被活化，因不用exponential而training速度較快。

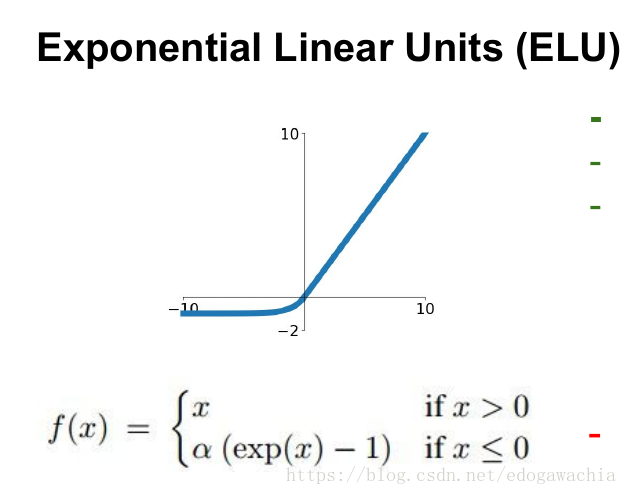
1. Leaky ReLU



F(x) = max(0.01, x)

為了解決一開始參數在初始化而無法被activate的過程，在負數區域使y = 0.01\*x，在正數區域使y = x。

1. (Exponential Linear Units) ELU

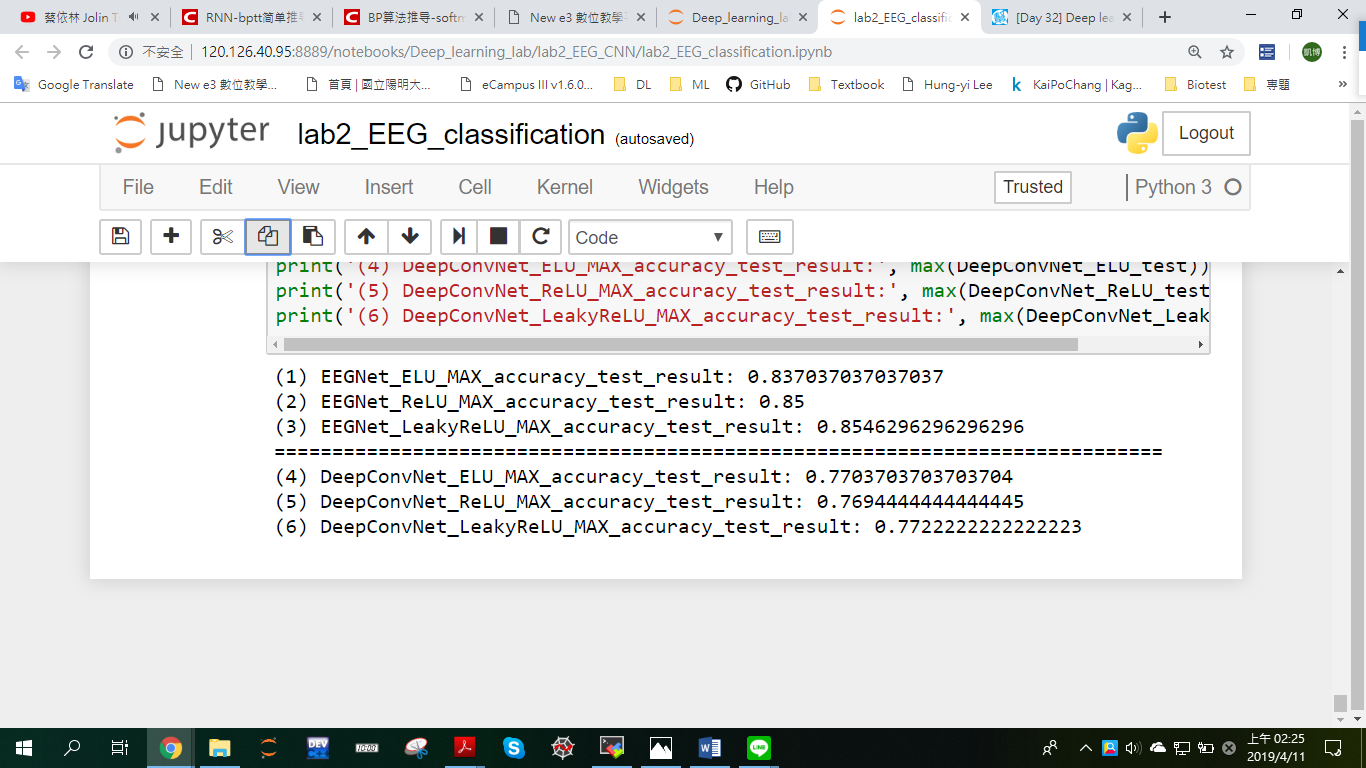
若x>0，f(x) = x，反之當x小於等於0，

f(x) = (alpha)\*(exp(x)-1)。可以說是綜合relu和LeakyReLU兩個activation function的優點，既不會

因為在初始化的時候陷入f(x) = 0的窘境，又因為在負數區域有漸進線，所以可以對Noisy有tolerance。但缺點是因為有exponential，所以計算量較大。

*圖源:* [*https://blog.csdn.net/edogawachia/article/details/80043673*](https://blog.csdn.net/edogawachia/article/details/80043673)

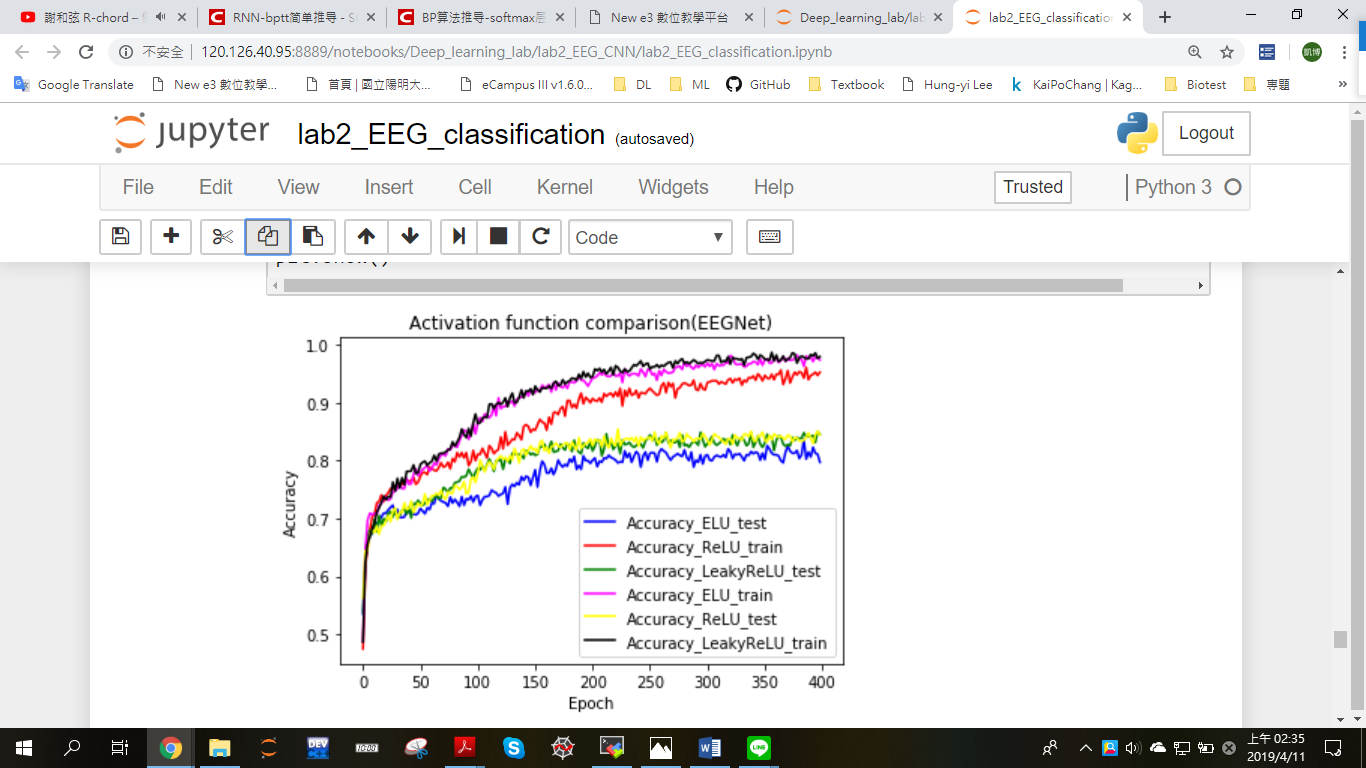
1. Experimental results
2. The highest testing accuracy
3. Screenshot with two models



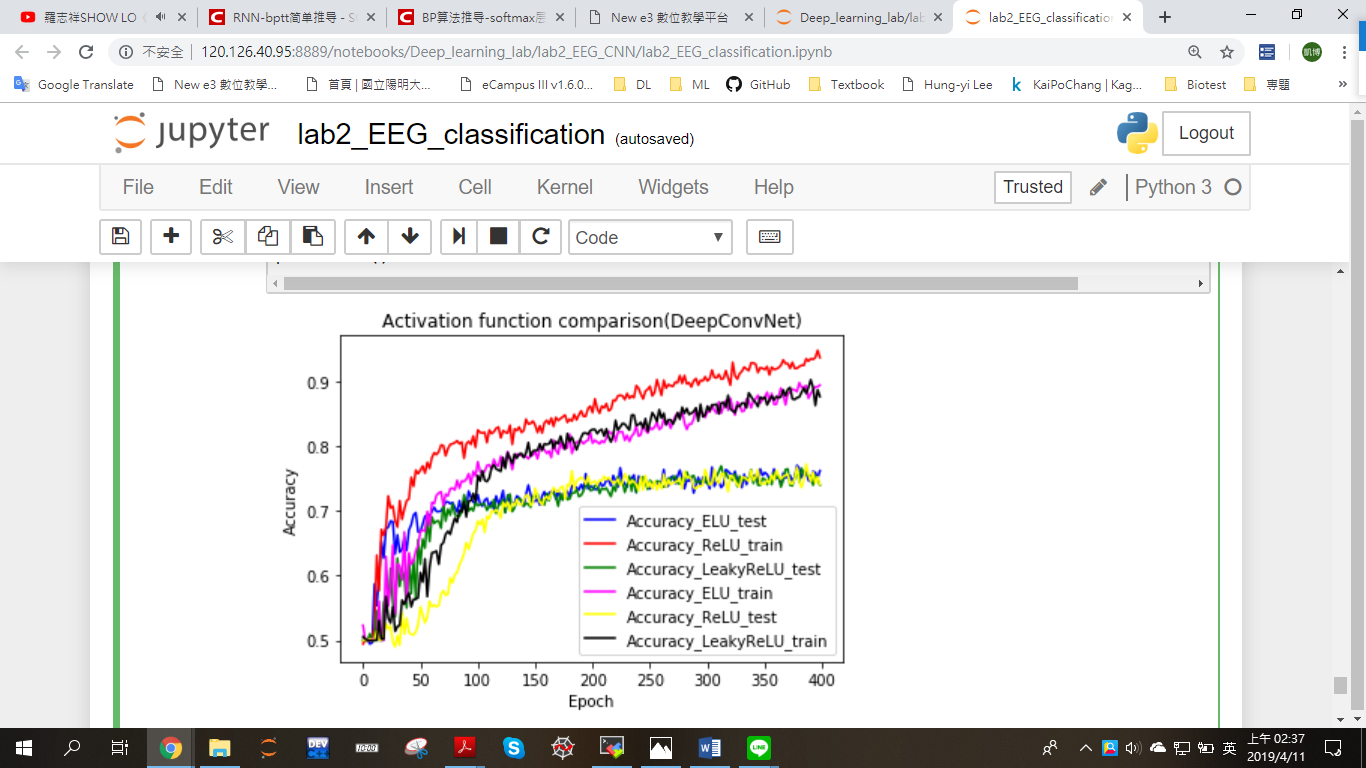
1. Anything I want to present

就預期結果而言，在同一種model底下不同的activation function，如剛剛描素不同的activation function而言，LeakyRelu的accuracy的test result的結果是最好的，而ELU和Relu的不一定可以比較得出來，有可能是model的variance或者是外部假設的hyperparameter不同而有所差別。

1. Comparison figures
2. EEGNet



1. DeepConvNet



1. Discussion

這次Lab，最初令我難以理解的是CNN的架構，因為是第一次實作，所以也和之前一樣做了很久，一開始是從Pytorch實作CNN的tutorial開始一步步看sample做的，到後來了解tutorial開始實作後，發現我對助教給的data完全是一無所知，在打不開npz檔的窘境之下，不得已我又參考了另一篇的sample code，把CNN依次送入的data batch和channel搞清楚，最後在調一下hyperparameter才實做出來。

一開始我按照助教給定的table去實作，發現6個accuracy中，最高的大約是83%，但是我把送進去的batch size改成16、learning rate改成10^(-3)、num epochs改成400之後，發現最高的accuracy有到85.43，但是若將batch size改成8，最高的accuracy卻只有約80%，研判可能是overfitting的問題。