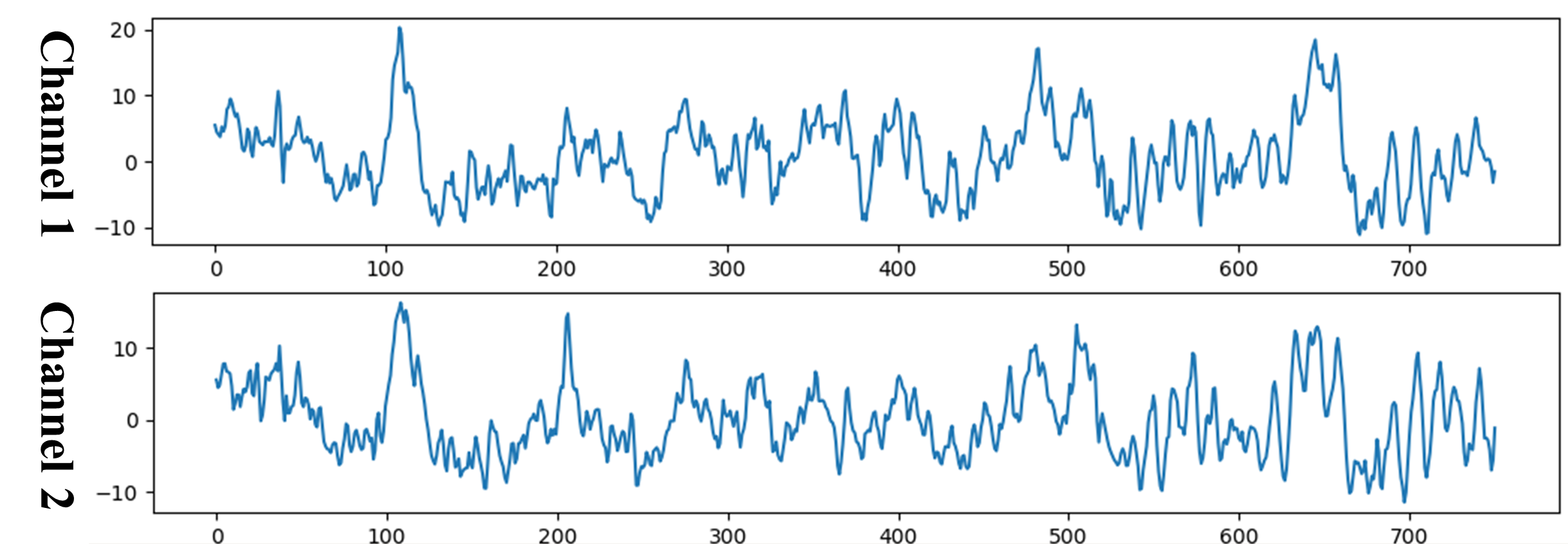
Lab2 : EEG classifications

311511043 李承翰

1. Introduction

在這次Lab，需要我們使用pytorch去實作EEGNet還有DeepConvNet兩種分類模型，用來分類BCI資料集，而這個資料集會有兩個channel，分別代表著左手與右手，而總共會分別有1080筆的訓練資料及測試資料

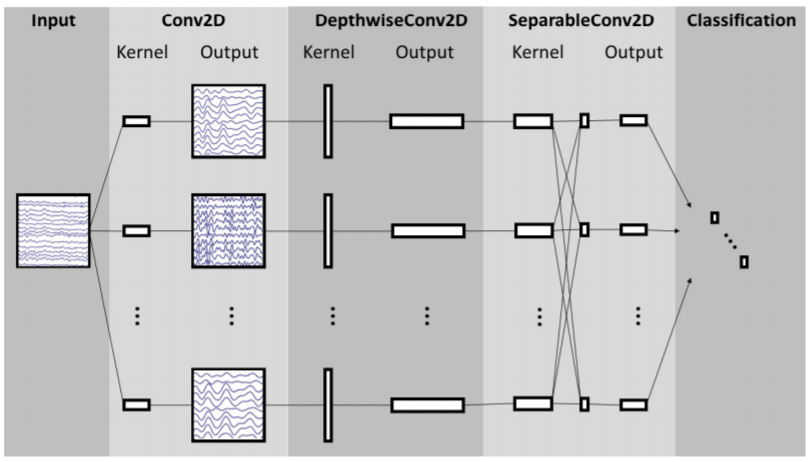


上圖為資料範例

此外，我們需要分別嘗試三種的activation function(ReLU, Leaky\_ReLU, ELU)，並且探討、比較實驗結果。

1. Experiment setup
2. The detail of models:

* EEGNet



EEGNet是一個小型的卷積神經網路，常用於腦電波類的資料，因為大幅降低了模型當中的參數數量，所以可以在有限的硬體當中還是有不錯的訓練速度。

而整個模型的架構如上圖所示，大致上可以分成三個convolutional layer，其設定如下:

一張含有 文字 的圖片

自動產生的描述

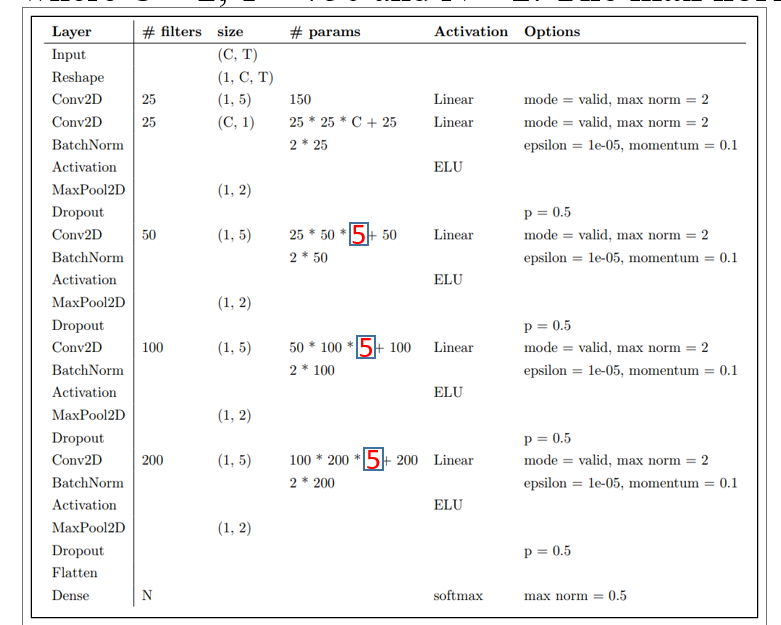
第一層的FistConv layer 負責的是將input 的資料以頻率的方式分成16個channel

第二層的 Depthwise layer 負責將第一層經過frequency filter的各個通道再經過spatial filter，而因為我們是直接將第一層的output分別當作第二層的input，讓使用上減少了參數數量，不過後果就是我們沒有辦法獲取不同頻率(通道)下同個空間的資料

而最後一層則是總結了第二層過來的資料，彌補了第二層的時候看不見不同頻率下相同空間的資料，將資料特徵都完整取出，交由下一層的linear classification來做判斷。

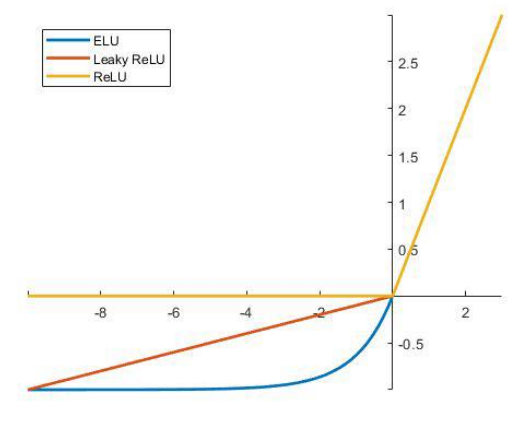
* DeepConvNet

DeepConvNet是傳統的深度學習架構，由一層捲積層，在後面再接上convolution、batch normalization、activation function、pooling、dropout，以這樣的順序重複4次，最後再用一個fully connected配合softmax進行分類，而每一層的參數設定如下圖: C = 2, T = 750, N = 2



1. Explain the activation function:

激勵函數的存在是為了讓神經網路模型可以判斷非線性的資料，如果沒有加上激勵函數，那麼整個神經網路的模型在判斷資料時就會出現困難。而這次我們分別要使用ReLU,Leaky\_ReLU,ELU三個激勵函數



* ReLU

f(x) = max(0,x), f’(x) =

ReLU函數是現在常用的激勵函數，因為他運算速度快，而且不會有梯度消失的問題存在。不過因為她在x < 0的時候梯度為0，因此在某些情況下面有可能完全不激發，而這種問題被稱為dead ReLU問題。

* Leaky\_ReLU

f(x) = max(0,x) + min(0.001x,0) , f’(x) =

Leaky\_ReLU對ReLU函數進行了改良，針對x < 0的部分提供了解決方案，在本來梯度為零的部分增加一個0.001的項，讓模型可能未被激活的情況得到改善，不過Leaky\_ReLU依然有在x = 0的點無法微分的情況。

* ELU

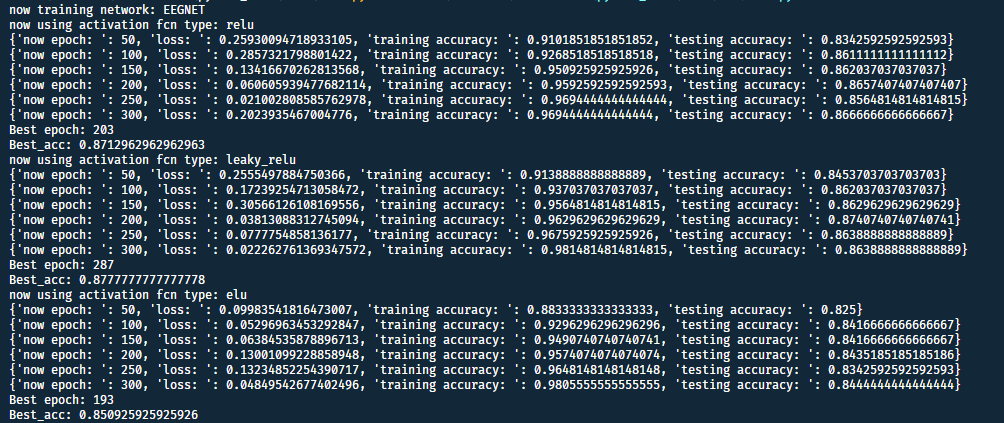
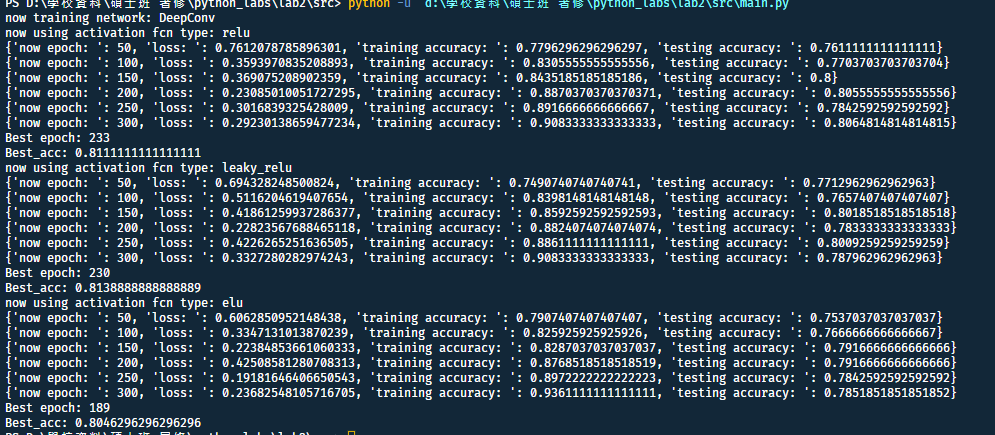
f(x) = max(0,x) + min(𝛼 ∗ (𝑒𝑥𝑝(𝑥) − 1),0), f’(x) =

ELU也是ReLU函數的改良版本，他解決了上述兩種函數遇到的各種問題，但是因為在函數運算當中牽涉了指數函數的運算，因此有了運算量較大的問題。

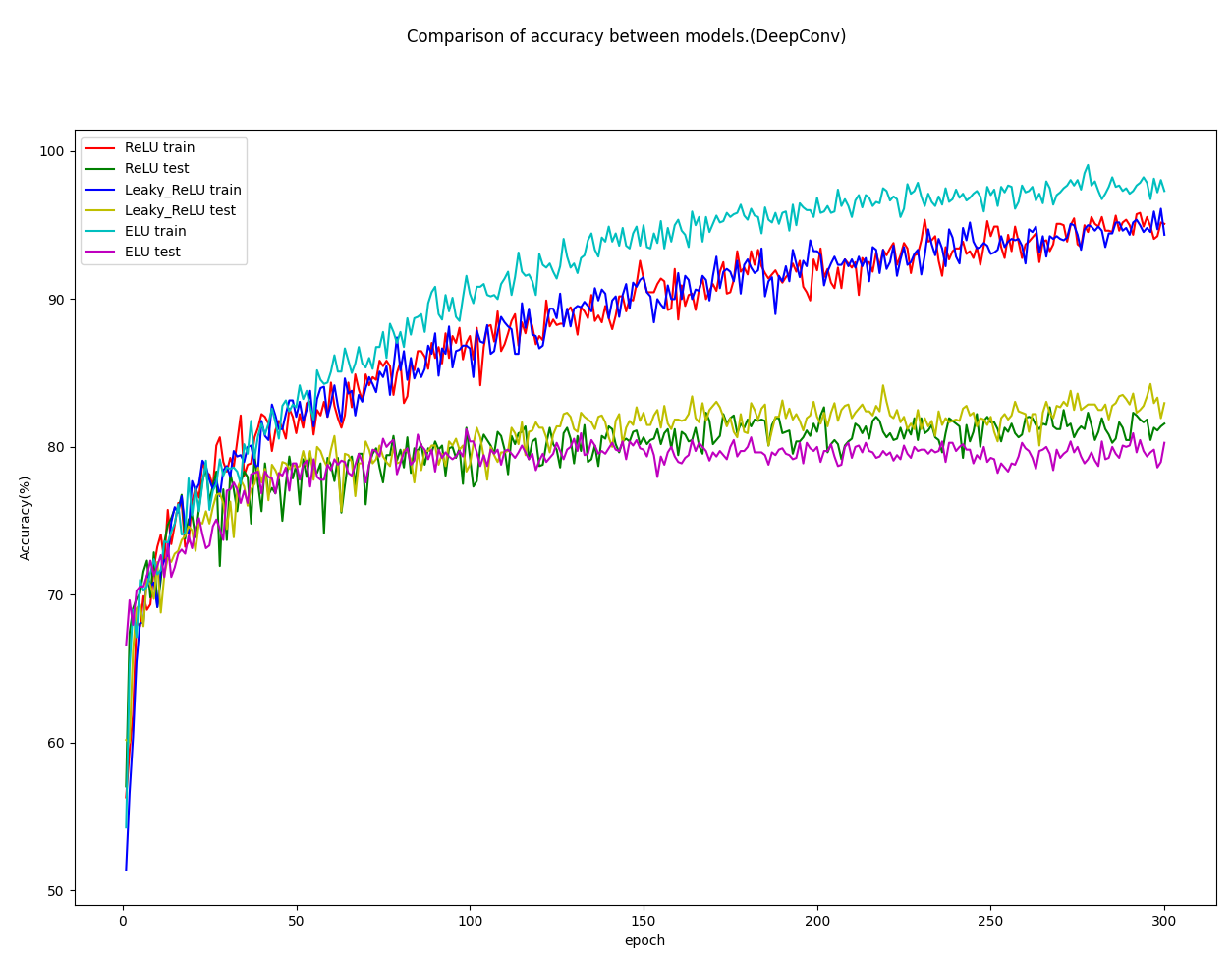
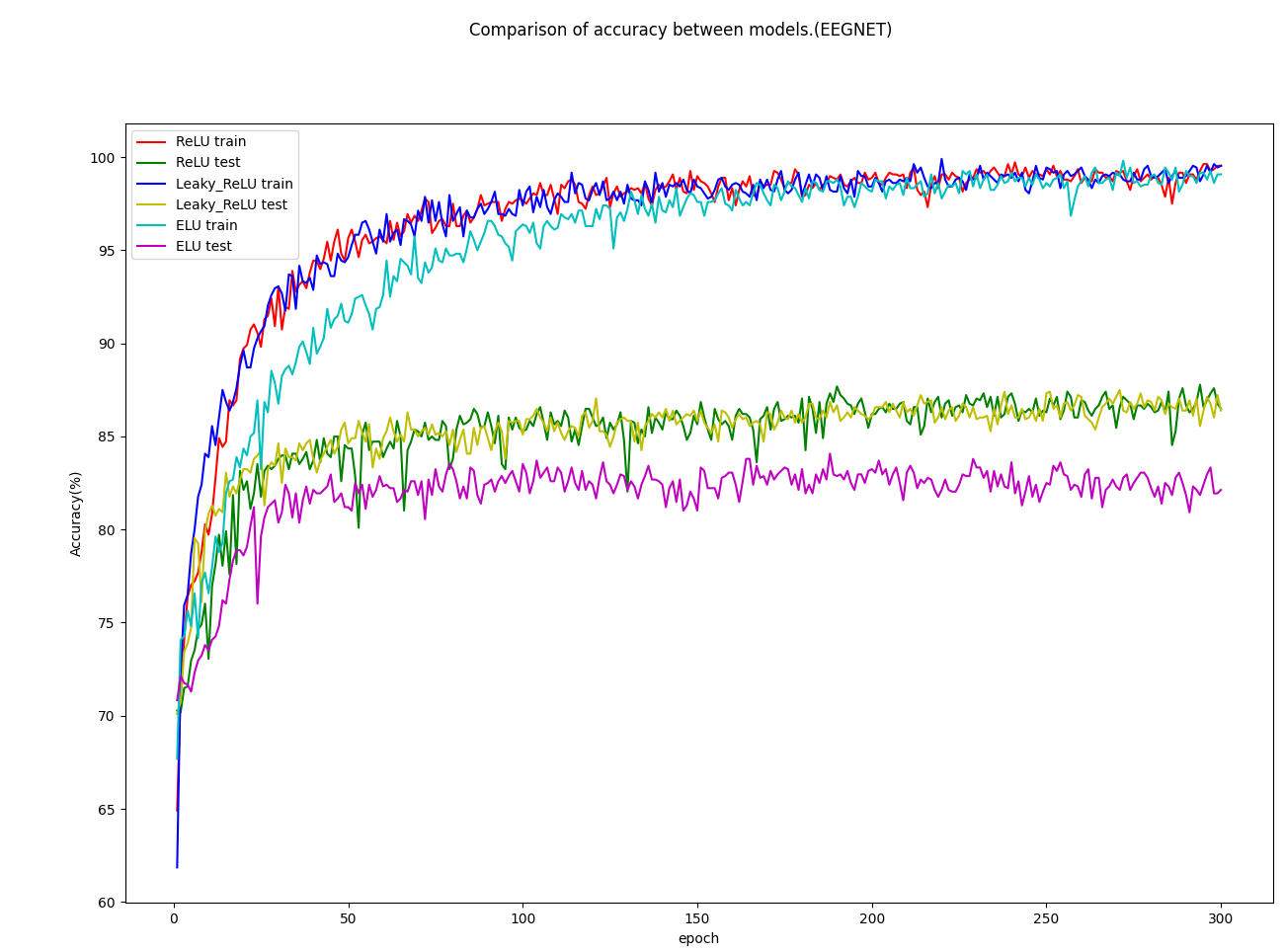
1. Experimental results
2. The highest testing accuracy:

表格中與截圖數據有些微不同，因為是不同次實驗當中擷取的

|  |  |  |  |
| --- | --- | --- | --- |
|  | ReLU | Leaky\_ReLU | ELU |
| EEGNet | 88.98% | 89.07% | 83.61% |
| DeepConvNet | 81.11% | 81.38% | 80.46% |

1. Comparison figures



1. Discussion

這一次的實驗當中有幾個主要的可以供我們調整的超參數:

* 1. Learning\_rate
  2. Momentum
  3. Epoch
  4. Dropout rate
  5. Batch size

雖然在模型當中還有很多我們可以調整的超參數，但我主要有修改到的超參數就是這幾個，而learning rate、momentum、epoch會有什麼樣的影響在上一次lab當中都有提及，這次我新學到了dropout rate，以及batch size。

在這次的模型當中我們在許多地方加了一層dropout layer，這一層相當的神奇，她會以機率 = dropout rate的方式，在訓練過程當中適當的使一些神經元不被觸發，以這樣的方式來防止overfit。而在EEGNet這種比較沒那麼深的模型當中，跟在DeepConvNet這種深度的模型比起來，兩者就需要不一樣的dropout rate，在DeepConvNet當中需要較大的dropout rate才可以有效地防止overfit。

而batch size的部分，雖然在lab1當中有學到，但當初的資料我並沒有特別針對batch size進行調整。這一個參數代表著一個epoch當中，要分幾次將所有的訓練資料丟到模型當中去訓練。如果將batch size調大，那麼因為矩陣的乘法次數會減少，所以整體來說運算效率也會提升，但因為在進行gradient運算的時候，將各個訓練資料所產生的gradient進行了平均的動作，因此每一次error造成的gradient特徵會在這一個過程當中被犧牲，在模型學習上會相對而言不直觀。因為理論上我們對每一個權重的修正量，應該是看此次error的gradient，但這個一平均下去就會稍微變得不同。所以在學習效率上我們需要稍微取捨。