**Overview**

本次作業是要針對已經事先寫好的類神經網路骨幹做填空和修改，資料集是28\*28的圖片對47個class進行判別。這次的作業剛好逢自己去開刀動手術，有點不剛好，所以CNN的部分就來不及鑽研做出來，但也慶幸助教讓我晚交不扣我分，所以在此「感恩助教！讚嘆助教！」，謝謝助教的寬宏大量 😊。

雖然如此，我還是很努力地試圖增進我的performance，我對我的模型作了以下的改變：

**Improvements & Tweaks**

1. **在每個epoch進行之前將訓練資料重新洗牌：**

為了減少temporal data對訓練的影響，我在每個epoch進行之前將x、y training資料綁起來，重新shuffle一次然後再拆開做labels的one hot。當然，這樣的話很多code的寫法就要稍微修正，像是做one hot的地方，以及validation set在一開始就要切好，不能直接在iteration進行當中從train data取subset，否則這樣的話因為shuffle的關係，epoch與epoch間的training data很可能成為下一個epoch的validation data，但這樣就會失去validation的意義，因為等於模型有對validation set進行訓練，loss會不斷下降、accuracy不斷上升，最終導致overfitting。

1. **修改validation set的batch size**原先設定batch size的意義就是怕在training的時候因為資料筆數太多，如果每跑完一筆資料就要更新weights會太沒有效率，所以將一塊一塊的資料綁在一起一次訓練更新，training data是這樣但validation data就不一定了。因為validation data不是我們要訓練拿來更新我們權重的資料所以沒有更新效率的問題，會把validation data分成batch的原因就是怕在validate的時候記憶體一次塞不下，而我這次把validation的batch size另外做設定，讓他是validation size的因數，可以整除validation set，這樣就不會有時因為batch size的整除問題導致一些資料沒有validate到，可以做更完善的validation。
2. **改變網路層級數**本來的層數是2，我在經過測試後將其改為4，基本上，只要超過兩層的網路都可以有效模擬任何有限範圍的函數，在大於兩層就純粹靠試驗了，而我發覺4層的效果不錯。
3. **改變神經元個數**神經數的設定我在查資料的過程當中眾說紛紜，跟網路層一樣沒有確切的答案都要靠試驗以及嘗試，但因為神經數總不可能一個一個累加看模型的表現所以我抓住了幾個要點去設計自己的神經元個數：

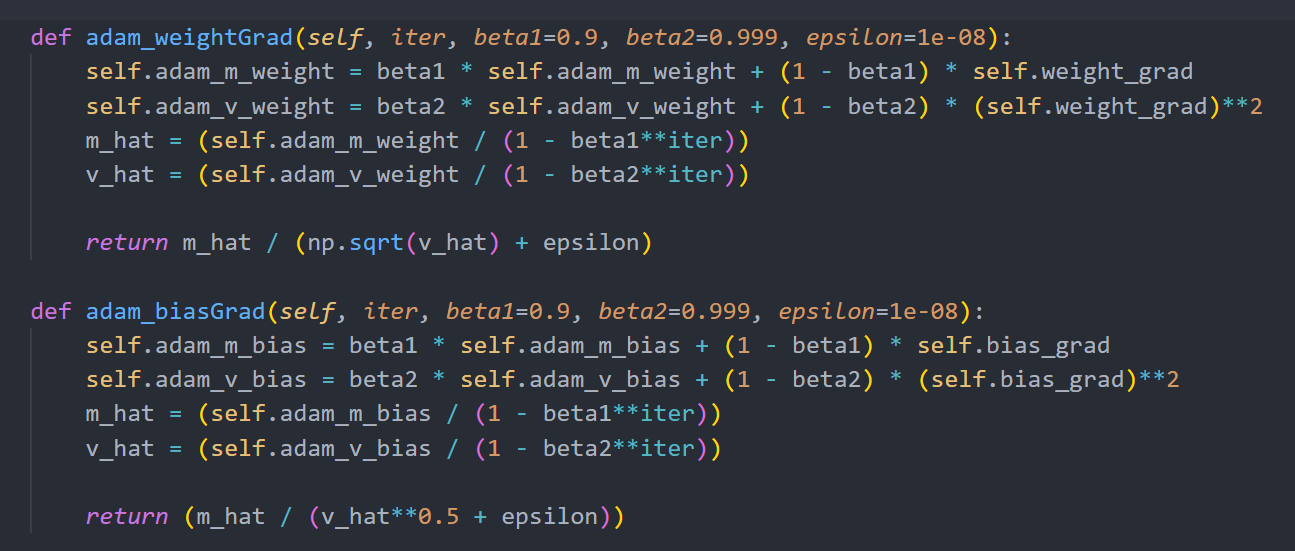
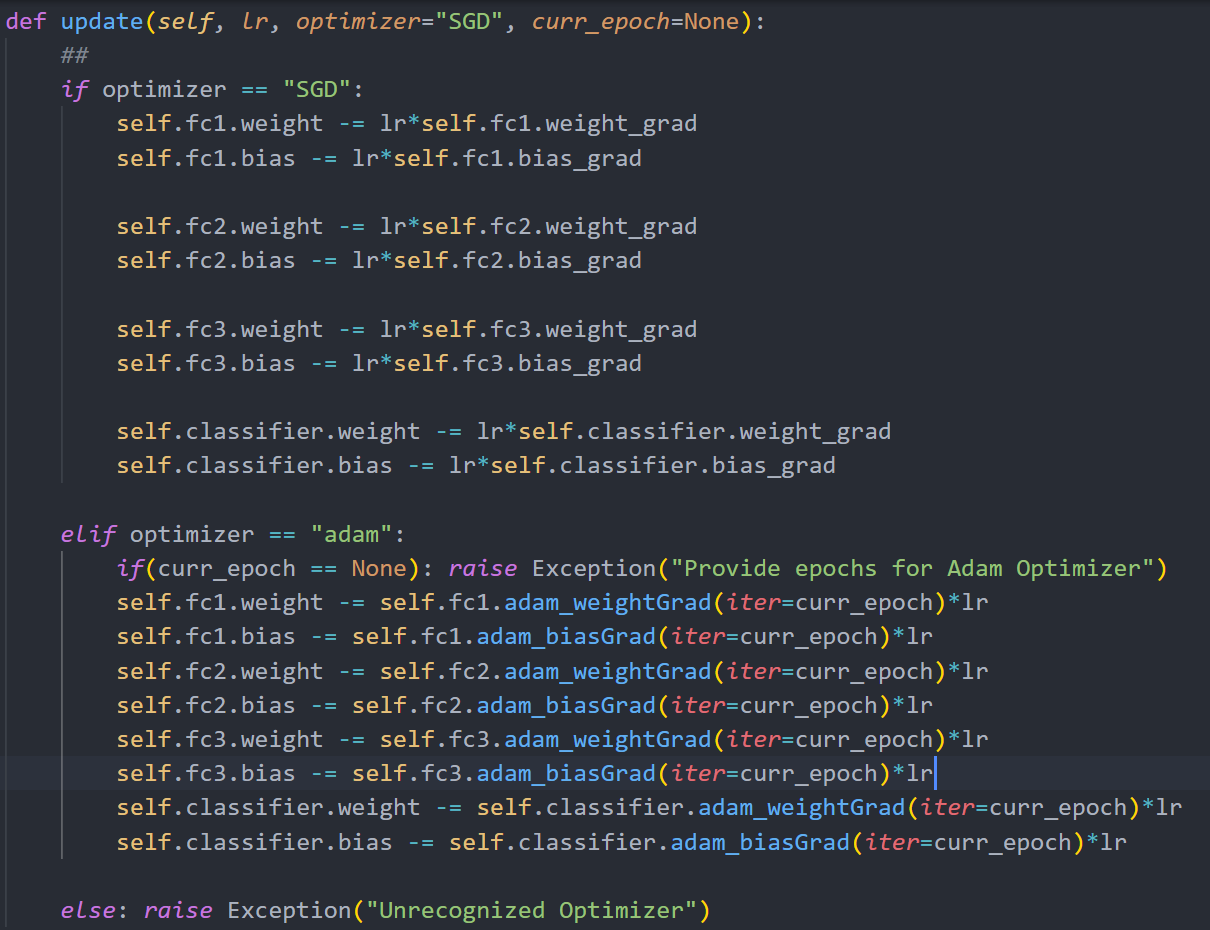
* 神經數藉由輸入層和輸出層大小之間
* 神經數大概是輸入層大小的2/3加上輸出層

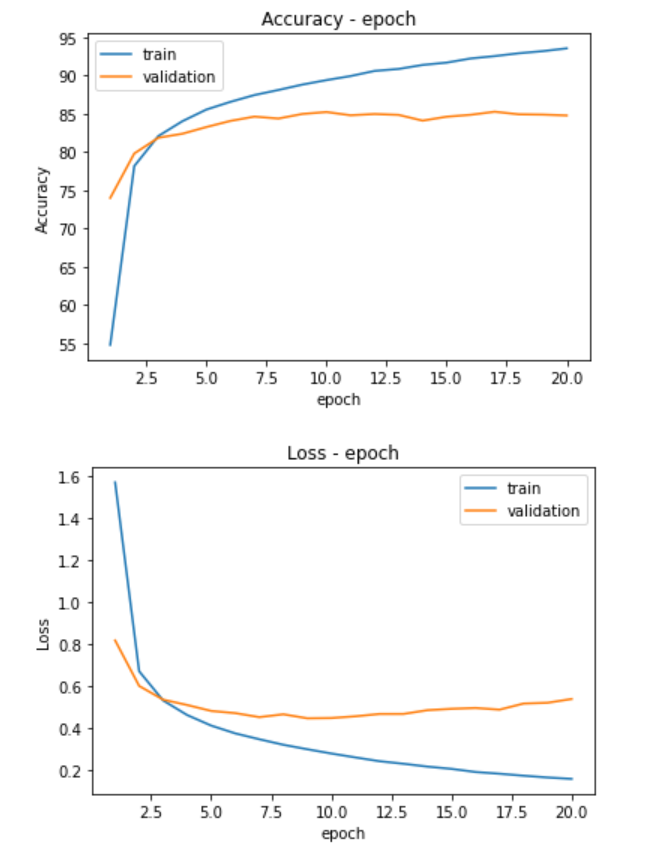
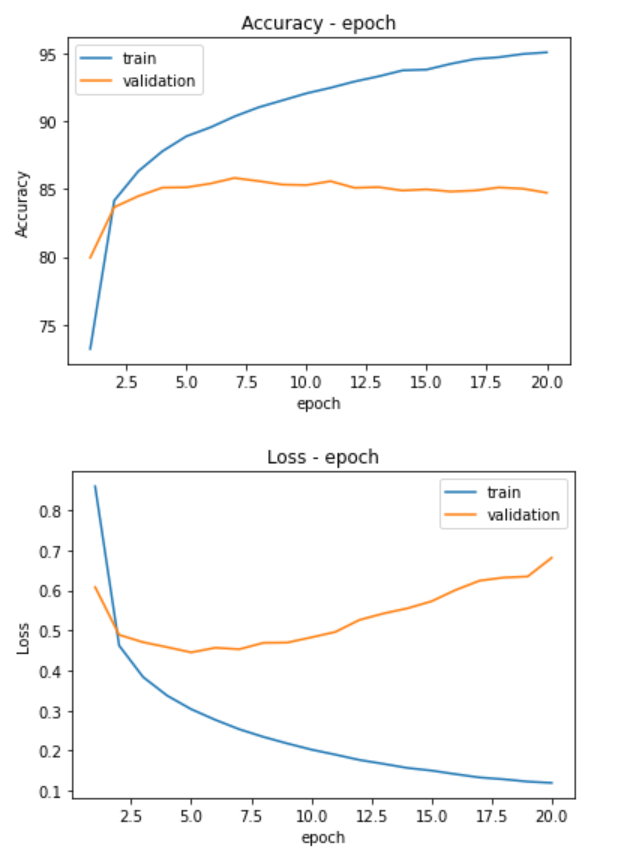
所以在中間夾雜的兩層網路維度我分別取了先前提到的

以及

最後一層的classifier就用原本的60。

1. **調Learning Rate  
     
   5.1 The Traditional Method**  
   LR大小在我所有參數裏頭花最多心思也應該是影響最大的參數，首先我用了不同的learning rate去跑以上已經建出來的模型，結果發現因為learning rate愈小，所需要的epoch就自然愈多，但這樣一個一個訓練太沒有效率了，我就決定進一步增強自己模型的訓練方式，給它加了個Adam optimizer！

**5.2 With Adam Optimizer**  
因為怕動到好不容易建起的模型，所以我先在未調過神經層、神經元個數的預設檔寫入adam optimizer，參考TensorFlow還有當年發表Adam的paper預設的learning rate、beta、epsilon，結果沒想到竟然跑不動，accuracy一直卡在零，後來把adam移到四層的模型中就成功地跑出來了，可見神經網路的深度真的還是對training有影響，而且收斂速度真的很快，又不用調太多參數，很快的讓我train我的model。這次配合先定義好的架構，我將自己的adam optimizer放在Fully Connected的class定義裏頭，這樣好分別存weight和bias的w和v，設計如下：  
  
  
比較傳統的learning rate和加了adam optimizer果然收斂速度不一樣！

   
 traditional gradient descent with adam optimizer  
由上圖中可以看到傳統的SGD到了十幾個epoch才飽和，而adam optimizer在5~7個epoch就差不多飽和了。  
把最終結果上傳到Kaggle上面可以得到84.8%的準確度。