Structure：

在 basic 的 binary classifiers 我使用了三層 hidden layer，activation function 都是使用 relu，而最後的 output layer 因為要做二元分類所以使用的是 sigmoid，總共五層 layer(包含 input layer)的 neural network，每一層的 nodes 數分別為 30(input layer)，20, 10, 5, 1(output layer)。

在 advance 的 multi-class classifiers 裡因為一開始的 units 數比較多，為了避免 units 太快減少，我使用了四層 relu 的 hidden layer，但 output layer 因為要分類十個數字所以要使用 softmax，總共也是六層 layer，但是因為 input layer 是以像素格作為參數，nodes 數即為總共的像素格，高達 784 個，所以在後面 hidden layer 的 nodes 數也會相應變大，我選擇的 hidden layer 的 nodes 數為 512，256，128，64，最後要 output 10 個數字的個別機率所以 output layer 的 node 數要為 10。

Improving model：

一開始對於 neural network 不熟悉開了很多 layer 而且 iteration 很多次，train 到一半的時候發現 cost 不再改變但也不敢停止，等到 train 了好幾個小時後發現 accuracy 不盡人意，後來查了 google 才知道 layer 越多層越可能發生 vanish gradient problem，隨後我先將 iteration 的次數減少以觀察 cost 的變化，然後再將 layer 減少與對 batch size 做調整，到如今便形成這個五層和六層的 neural network。

Difficulty：

- 對於 Neural network 的觀念和符號不熟悉
- Back propagation 的數學微分難以計算
- Dimension 的對應

Summarize：

此次的作業在實作的 coding 量不會比前兩次多，但在觀念上卻比前兩次都還要深，其實感覺 neural network 就是將前兩次的觀念都結合在一起，變成一個更大的 model 去做 learning。而對很多東西的不熟悉當然也是花了不少時間在講義和網路資料中來回才逐漸找到實作方向和鞏固觀念。但是在 back propagation 的微分計算上面，首先遇到的困難是在 code 裡面的符號分別對應哪一項偏微分要弄清楚就花了不少時間，而後微分出來的東西打進 code 裡測出來發現不是因為 dimension 沒有對應好發生 error，就是和 expect answer 不一樣。在很多輪運算和思考 dimension 後，總算逐漸實作好了部分的 code，後來往下滑的時候才發現——其實公式已經寫在下面了。知道的瞬間突然覺得先前的自己很好笑，但是相信經過自己算過想過之後會對這整個運作更加熟悉。

進入了 neural network，也代表正式踏入了 deep learning 的範疇裡，隨著 final project 確定題目，相信後面關於 deep learning 的知識將會更加有趣。