MLDS HW1

B03901145 郭恆成 B03901101 楊其昇 B03091065 林宣竹

I. HW1-1 Deep v.s. Shallow

1. HW1-1-1 Simulate a function

Bonus (5) Use more than two models. (6) Use more than one function.合併在(1)~(4)中回答

(1) Describe the models you use, including the number of parameters (at least two models) and the function you use.

使用了 shallow, medium 跟 deep 三種深度的 DNN 模型，訓練在 damping function 以及 triangle wave function 上。shallow, medium 與 deep 模型結構分別是 1 層 hidden layer size=288、2 層 hidden layer size=28、4 層 hidden layer size=16。Activation 除了 output layer 為 linear 外，其餘皆為 ReLU。詳細如下圖

Layer (type)	Output Shape	Param #
shallow/hidden/kernel:0	[1, 288]	288
shallow/hidden/bias:0	[288]	288
shallow/outputs/kernel:0	[288, 1]	288
shallow/outputs/bias:0	[1]	1

Total parameters: 865

Layer (type)	Output Shape	Param #
medium/hidden1/kernel:0	[1, 28]	28
medium/hidden1/bias:0	[28]	28
medium/hidden2/kernel:0	[28, 28]	784
medium/hidden2/bias:0	[28]	28
medium/outputs/kernel:0	[28, 1]	28
medium/outputs/bias:0	[1]	1

Total parameters: 897

Layer (type)	Output Shape	Param #
deep/hidden1/kernel:0	[1, 16]	16
deep/hidden1/bias:0	[16]	16
deep/hidden2/kernel:0	[16, 16]	256
deep/hidden2/bias:0	[16]	16
deep/hidden3/kernel:0	[16, 16]	256
deep/hidden3/bias:0	[16]	16
deep/hidden4/kernel:0	[16, 16]	256
deep/hidden4/bias:0	[16]	16
deep/outputs/kernel:0	[16, 1]	16
deep/outputs/bias:0	[1]	1

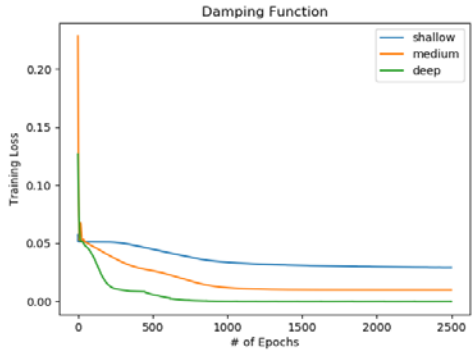
Total parameters: 865

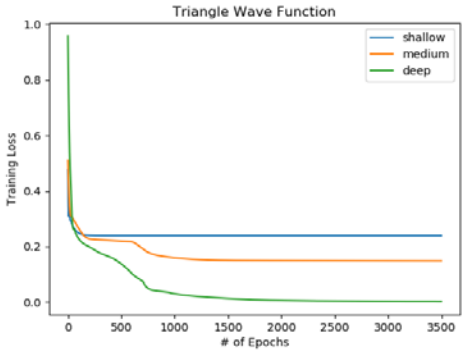
(左上圖 shallow、左下圖 medium、右圖 deep)

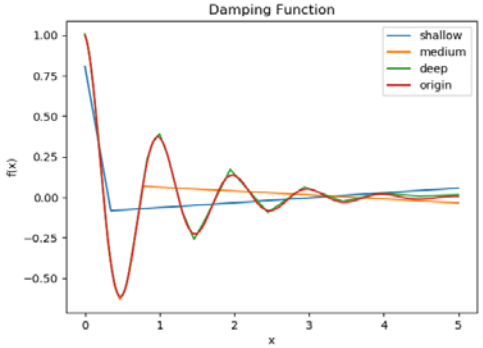
(2) In one chart, plot the training loss of all models.

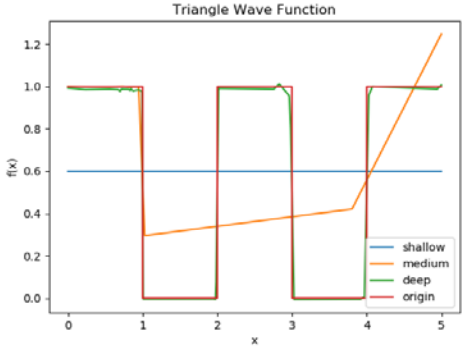
(3) In one graph, plot the predicted function curve of all models and the ground-truth function curve.

(2), (3)題合併作答，左上到右下依次為 damping function epoch-loss 圖、triangle wave function epoch-loss 圖、damping function x-f(x) 圖以及 triangle wave function x-f(x) 圖。









(4) Comment on your results

由 epoch-loss 圖可以觀察到，deep 的模型 loss 降得最快，而且對相同 epoch 數可以得到更低的 loss，證明 deep 的結構，學得較快且能更好的 fit training data，而這樣的結果跟模型的深度成正相關。而 visualize 各個 model 預測出來的 function，也能發現 deep 模型只有在數值變化較大的轉角處不太理想，其餘皆能有良好的預測。而 medium 模型，只有前面小部分的 function 預測得不錯，其餘只能看出大概趨勢。shallow 模型則完全還未學起來。可見參數量相仿的情況下，deep 模型對 training data 有較佳的學習能力。

(5) Use more than two models in all previous question.

(6) Use more than one function.

2. HW1-1-2 Train on actual task using shallow and deep models

(1) Describe the models you use and the task you chose.

選擇 MNIST dataset，並分別比較 CNN 和 DNN 的 shallow 模型和 deep 模型的差別。實驗模型架構如下：

a. CNN:

實驗中 filter size=(3, 3)，strides=(1, 1)，均會 padding 至原本圖片大小(padding=SAME)，且皆沒有 pooling layer。Activation function 除了 output layer 為 softmax 外，其餘為 ReLU。Shallow 模型的設計為 1 層 hidden layer，filter 數量分別為 8、16、32；Deep 模型的設計依序為 1、2、4 層 hidden layers，filter 數量固定為 8 或 16。

b. DNN:

實驗中 hidden layer 的 activation function 除了 output layer 為 softmax 外，其餘為 ReLU。Shallow 模型的設計為 1 層 hidden layer，hidden units 數量分別為 64、128、256；Deep 模型的設計依序為 1、2、4 層 hidden layers，hidden units 數量固定為 64 或 128。

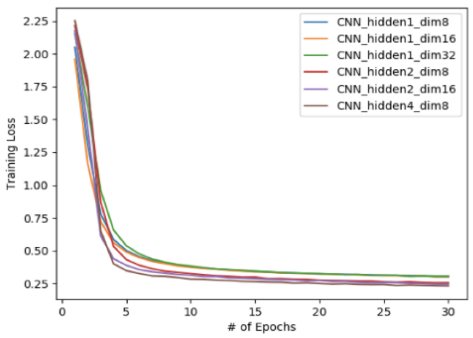
下表為實驗中各模型的參數量：

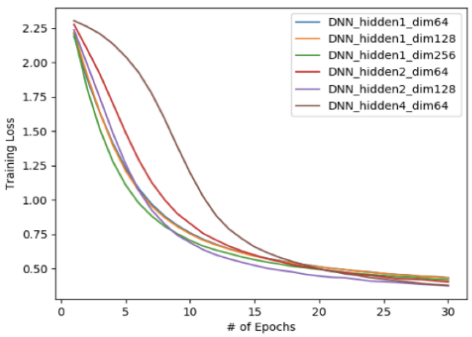
Model Type	# Parameter	Model Type	# Parameter
CNN_hidden1_dim8	62810	DNN_hidden1_dim64	50890
CNN_hidden1_dim16	125610	DNN_hidden1_dim128	101770
CNN_hidden1_dim32	251210	DNN_hidden1_dim256	203530
CNN_hidden2_dim8	63394	DNN_hidden2_dim64	55050
CNN_hidden2_dim16	127930	DNN_hidden2_dim128	118282
CNN_hidden4_dim8	64562	DNN_hidden4_dim64	63370

(2) In one chart, plot the training loss of all models.

訓練過程中 batch size=128，使用的 Gradient Descent Optimizer 作為 Optimizer，learning rate=0.001。

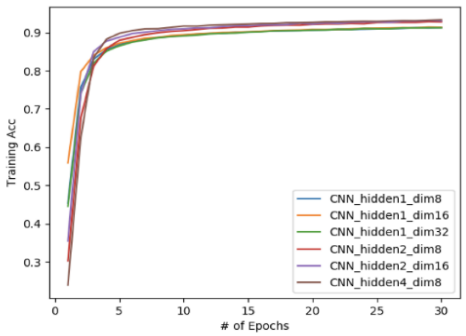
下圖分別為 CNN 及 DNN 的實驗中訓練過程的 loss：

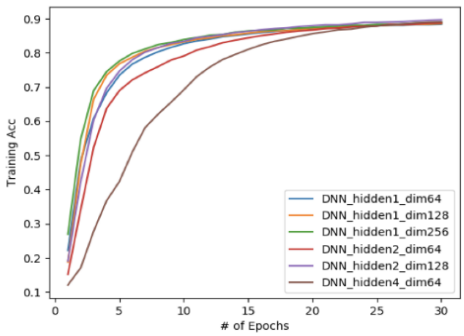




(3) In one chart, plot the training accuracy.

下圖分別為 CNN 及 DNN 的實驗中訓練過程的準確度：



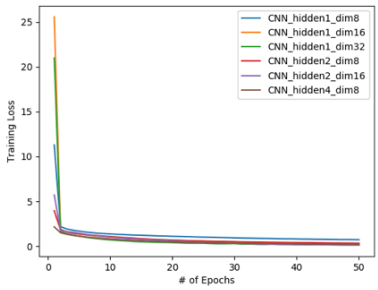
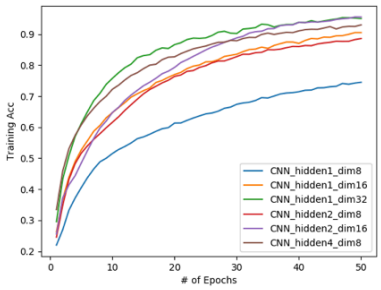
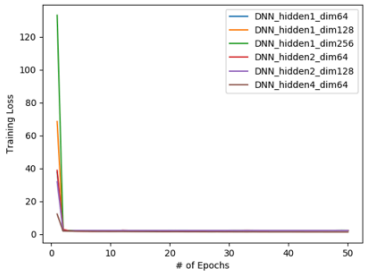
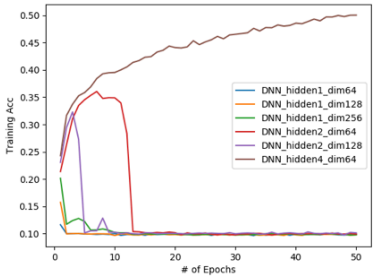


(4) Comment on your results.

由於 MNIST 屬於較簡單的 dataset，在本次實驗中 shallow model 和 deep model 的差別並不是十分巨大。不過仍然可以在上述的 training loss 和 training accuracy 中看出模型表現的好壞基本上跟模型本身的深度呈現明顯的正相關，但和模型本身的寬度並無十分明顯的關聯。以 CNN model 為例，在一層 hidden layer 的情況下，即便增加 hidden layer 裡面的 filter 數目，使其參數量增加為原本（約 6 萬）的兩倍及四倍，基本上模型最後的表現是差不多的。但相對若增加模型的深度，最後的表現卻可以在維持差不多數量級的參數量下獲得較佳的表現。本實驗最後結果也如同上課時所說的，deep model 執行較複雜的任務，確實可以在差不多的參數量下獲得比較好的表現。

(5) Bonus

將上述實驗過程重新在 CIFAR-10 上做一次。大部分模型架構皆和上述相同，只有 DNN 的部分有增加每層 hidden layer 的 neuron 數目以及為了較快得到收斂結果將 Optimizer 換成 Adam Optimizer。以下為實驗結果：

	Training Loss	Accuracy
CNN		
DNN		

可以明顯看出在這個實驗中，CNN 的表現遠比 DNN 要來得好，而且在參數量差不多的情形下(參數量在第一題中有提到)，Deep model 的表現會比 Shallow model 好很多。比較特別的是，或許是 CIFAR-10 是有 RGB channel 的圖片資料庫，所以若直接將將 input image 壓扁(flatten)後去接 DNN，即使每層的 neuron 數目調大，最後基本上還是 train 不起來。

但是隨著層數的增加，慢慢的訓練表現有變好。雖然二層 hidden layer 的 model 在一開始看似表現不錯，但 train 到最後也崩潰，但到了四層 hidden layer 的 model 就可以慢慢 train 起來了。因此在 DNN 的實驗中可以明顯看到，當 model 愈深，model 愈能處理更複雜的任務。

II. HW1-2

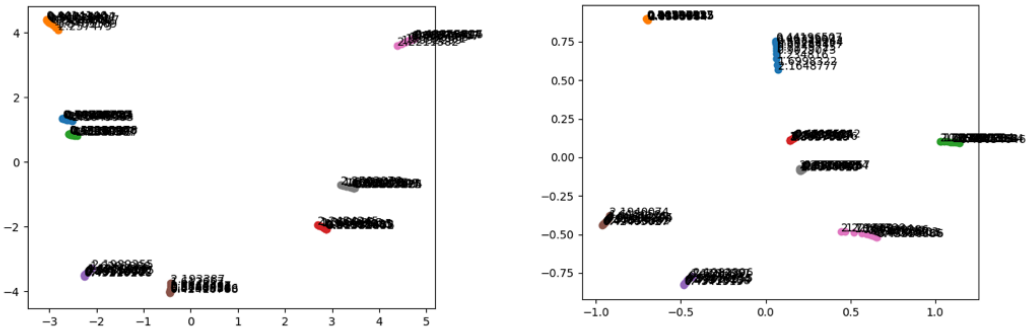
1. HW1-2-1 Visualize the optimization process.

(1) Describe your experiment settings. (The cycle you record the model parameters, optimizer, dimension reduction method, etc.)

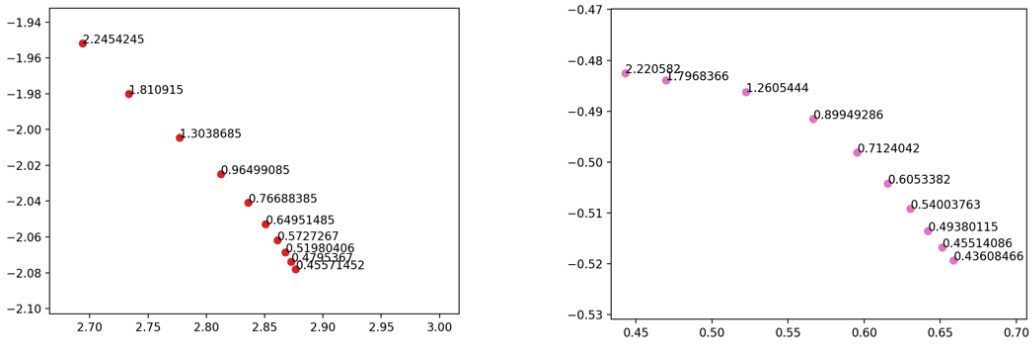
在本次實驗中採用 MNIST dataset，模型則是 2 層 hidden layer 且每層 hidden layer 皆為 32 個 neuron (總參數量約 2 萬 6 千)，activation function 為 ReLU，Optimizer 為 Gradient Descent Optimizer，learning rate=0.001，batch size=128。總共訓練 30 epochs，每三個 epoch 儲存一次模型來記錄參數。訓練完成後，按照作業說明投影片的方式將參數取出整理好後，利用 sklearn 套件進行 PCA，將高維的參數降到二維平面上。

(2) Train the model for 8 times, selecting the parameters of any one layer and whole model and plot them on the figures separately.

下圖分別為將第一層 hidden layer 的參數(下圖左)以及整個模型的所有參數取出來後做 PCA 後所得到的 loss 下降的過程(下圖右)。



分別放大其中一個 event 的軌跡圖(下圖左為只取第一層 hidden layer 的參數，圖右為全部模型的參數)：



(3) Comment on your result.

若如上題放大每個 event 的參數變化軌跡可以發現，不論每個 event 的初始位置在哪邊，軌跡的變化方向(即 loss 下降方向)基本上為朝著遠離座標原點的方向前進，因此可以推測原點的位置是 loss 最大值的位置，愈往外 loss 愈低。由於 MNIST 是相對簡單的 dataset，所以可以發現每個 event 的前面 2～3 點的參數變化軌跡最大和 loss 下降最多，到後面的

階段模型接近飽和，每次參數的更新幅度和 loss 的下降程度都小了許多。
另外，在實驗過程中有發現兩個現象：

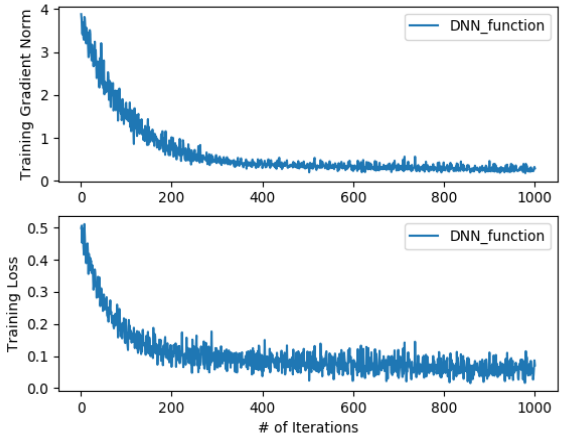
a. 若對整個模型的參數進行降維，其參數變化軌跡比只拿第一層 hidden layer 參數來得明顯。觀察參數的變化後發現，在 Gradient Descent Optimizer 的優化參數過程中，bias 的變化量比 weights 的變化量要大的多，因此拿來降維的參數中，若 bias 的部分愈多，則最後 PCA 將為後的軌跡變化愈明顯。

b. 若模型的參數愈少則壓縮成 2 維後的軌跡變化愈明顯，推測原因可能有兩個，其一為參數量大到某個程度時，已經有一定程度的隨機性，利用 PCA 無法成功有效的將資料降成 2 維；另一個原因可能是當參數量夠多的時候，有些 neurons 在訓練過程中是沒做事的狀態(silent neurons)，但因為其他 neurons 可以支援 silent neurons 的工作，因此 silent neurons 幾乎從頭到尾都沒有被訓練到，因此也導致最後 PCA 在降維的時候只有少數 neurons 參數的變化來驅動整體軌跡的變化，使得軌跡變化不夠明顯。

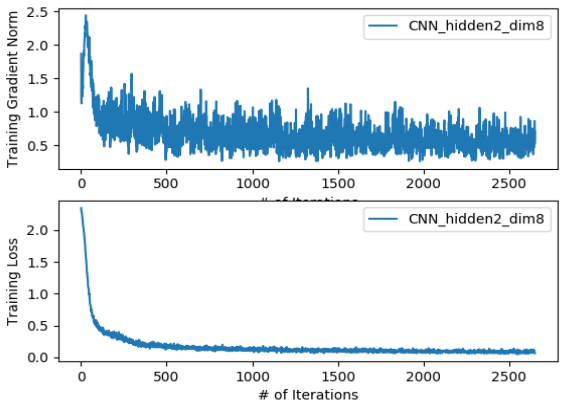
2. HW1-2-2 Observe gradient norm during training.

(1) Plot one figure which contain gradient norm to iterations and the loss to iterations.

a. 以方程式 $y = sinc(5\pi x)$ 從 0 ~ 1 取 1000 個 sample point 做為 dataset：
採用 DNN 模型架構，activation function 均使用 ReLU。因 dataset 並不複雜且量少，所以我們使用較簡單的模型，設計兩層 hidden layers，每層 neuron 數量為 10。Training 過程中，batch size 設為 100，而 gradient norm 和 loss 的變化紀錄如下：



b. 以 MNIST 手寫數字辨識資料集作為 dataset：
採用 CNN 模型架構，使用的 filter size=(3, 3)，strides=(1,1)，沒有 pooling layer。除了 output layer 使用了 softmax，其餘 activation function 皆使用 ReLU。共兩層 hidden layer，而 filter 數量取 2，總共參數量約為 15000。Batch size 取 1024，可以看到 Gradient norm 和 training loss 對 iteration 的關係如下：



(2) Comment your result.

根據作業投影片提供的公式，我們將每個參數的 gradient 平方後相加再開根號，得到 2-norm，然後紀錄每個 iteration 下 gradient norm 的值。

由上述兩圖可以看出 loss 隨著 iteration 增加而下降的同時，gradient norm 也跟著持續下降，因為當 loss function 隨著參數改變，逐漸接近自己的最小值時，參數變化量也會變小，而 gradient norm 就是所有參數變化量的平方總和再開根號，所以自然也會下降。

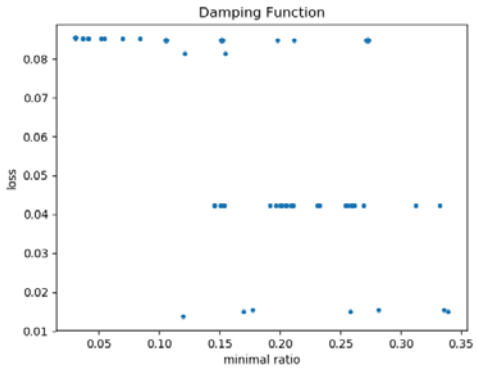
另外，訓練 MNIST dataset 時，gradient norm 的振盪幅度比較大，可能是每個 iteration 都記錄，而每個 iteration 訓練的 training data 不相同，所以根據局部的 data 去 optimize 得到的 loss 差異也就較容易震盪。

3. HW1-2-3 What happens when gradient is almost zero?

(1) State how you get the weight which gradient norm is zero and how you define the minimal ratio.

使用兩層 hidden layer size=4 的 DNN 模型，做在 simulate damping function 上。先 minimize MSE 5000 epochs，再接著 minimize gradient norm，直到 gradient norm < 1e-5。然後利用 loss function (MSE)對參數 θ 的 Hessian Matrix，求出其 eigenvalues，最後計算 eigenvalue 大於 0 的比例，得到 minimal ratio。不過由於 minimize gradient norm 的過程極不穩定，所以部分實驗直接 minimize MSE 直到 gradient norm < 1e-5。

(2) Train the model for 100 times. Plot the figure of minimal ratio to the loss.



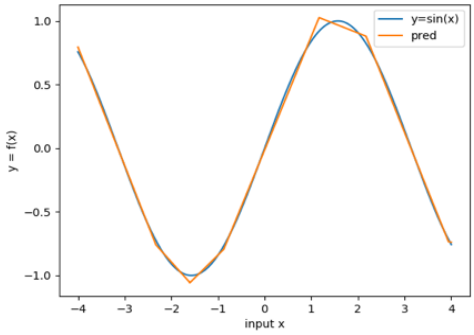
(3) Comment your result.

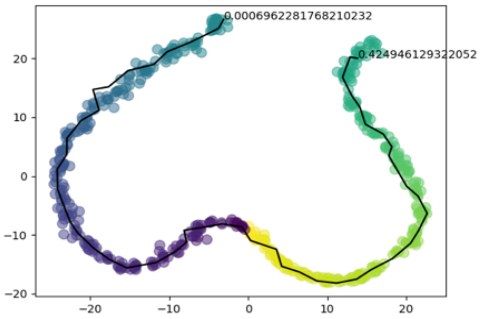
可以看出大概有三個 gradient almost 趨近於 0 的點，分別是 loss=0.085, 0.042, 0.015 附近。而 loss=0.085 左右的實驗，大部分的 minimal ratio 都小於 0.15，loss=0.042 附近的實驗則大多大於 0.15。另外 loss 落在 0.015 的樣本數較少，參考度較低，不過也可以發現大致上的 minimal ratio 還是高於 loss=0.085。理論上 minimal ratio 越高代表對參數 θ 小幅度的變化，得到的 loss 有較高的機率是上升的，越像一個 minimal point，所以預期 loss 也相對較低，這樣的理論與實驗結果相符。而實驗結果相同的 loss 橫跨的 minimal ratio 範圍很廣，應該是因為 gradient 很難真正 training 到 gradient=0 的點，只要一偏差就會根據周圍 loss function 的趨勢而有變化，較難穩定精確地描述。

4. **Bonus**

(1) Use any method to visualize the error surface.

(2) Concretely describe your method and comment your result.





上圖左為本實驗中的 simulate function $y = sin(x)$，模型訓練到最後的結果。模型架構本身為三層 hidden layer 的 DNN，每層 3 個 neurons，共有 34 個參數，訓練過程使用 Adam Optimizer (learning rate=0.001)。上圖右為將訓練過程(共 50 epochs)的參數更新過程，利用 TSNE 壓縮至二維。0.4249 標示著訓練過程中第一個 epoch 的 loss，0.000696 則是第 50 個 epoch。黑色折線旁邊的點則為每個 epoch 隨機選擇 10 個 step 的參數用同樣的方式將其壓縮至二維所繪製出來的結果。

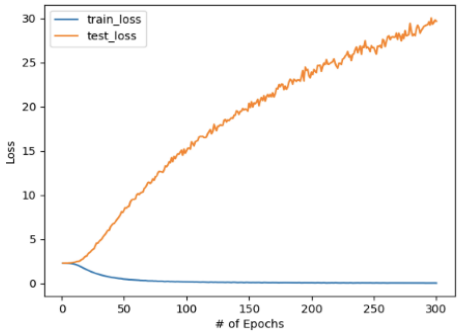
III. HW1-3

1. **HW1-3-1 Can network fit random labels?**

(1) Describe your settings of the experiments. (e.g. which task, learning rate, optimizer)

此實驗使用 MNIST database，model 為三層 hidden layers 且每層有 256 neurons 的 DNN，每層 layer 的 activation 除了 output layer 為 softmax 之外，其餘皆為 ReLU。為了較快看到收斂的結果，這裡使用 Adam Optimizer，learning rate = 0.001。

(2) Plot the figure of the relationship between training and testing, loss and epochs.



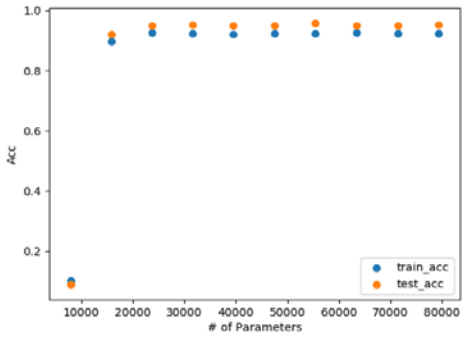
2. **HW1-3-2 Number of parameters v.s. Generalization**

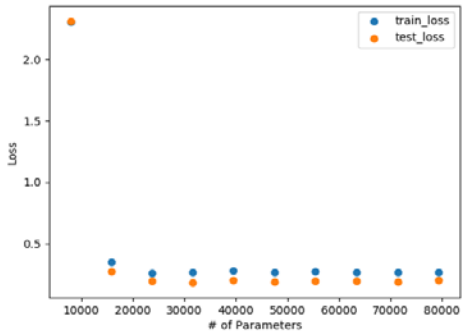
(1) Describe your settings of the experiments. (e.g. which task, the structures you choose)

以 CNN 為模型主要架構，一開始先採用 MNIST 作為 dataset，統一使用兩層 hidden layer，然後將各層的 hidden unit 從 1 變化到 10，紀錄不同參數總量下的 training 和 testing 結果，但是發現兩者差異並不顯著，所以改用 cifar10 作為 dataset。

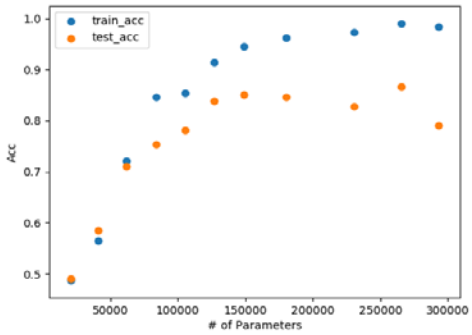
同樣使用 CNN 模型架構，建立 11 種不同參數數量的模型，其 hidden layer 和各層的 hidden units 數量分別(2,2), (2,4), (2,6), (4,8), (4,10), (4,12), (4,14), (8,16), (8,20), (10,22), (10,24)，參數數量範圍從兩萬變化到三十萬。訓練過程中，採用 Adam Optimizer，並將 learning rate 設為 0.001，batch size 設定在 100，觀察不同參數數量會對 training 和 testing 的結果造成什麼影響。

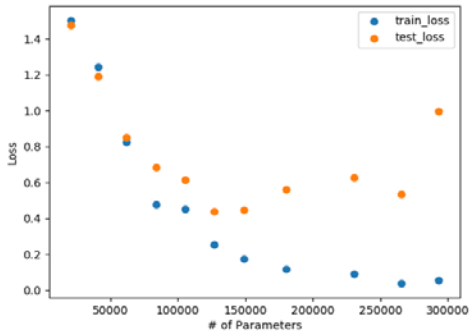
(2) Plot the figures of both training and testing, loss and accuracy to the number of parameters.
MNIST dataset





cifar10 dataset





(3) Comment your result.

在 MNIST 作為 dataset 的情況下，看不出 training 和 testing 的明顯差異，可能是因為它本身屬於較簡單的 dataset，很容易被 train 起來，只需要少量的參數就能使 training 和 testing data 達到 90%以上的準確率，所以看不出兩者差異。

反觀以 cifar10 為 dataset 時，當參數數量偏低時，testing 和 training 的 loss 與 accuracy 都不高而且很接近，隨著參數數量的提升，training 的 accuracy 可以被提高到將近 100%，然而 testing 的 accuracy 始終停留在 90%以下，兩者的正確率可以看出明顯的差異。

觀察這樣的數據，我們發現不斷加大參數數量，使之能更加 fit training data 的同時，testing data 所得到的結果也不會 overfitting，仍然維持一定的正確率，這也是 deep learning 特別的地方。雖然有許多人對此提出自己的看法與推論，然而現在對於這樣的現象仍舊沒有準確且受大家認同的解釋。

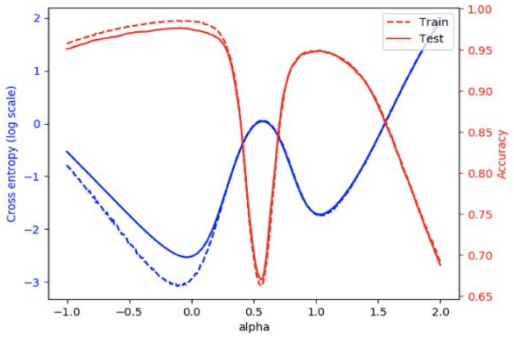
3. HW1-3-3 Flatness v.s. Generalization

Part 1

(1) Describe the settings of the experiments (e.g. which task, what training approaches)

此實驗同樣使用 MNIST database，model 架構採用 2 層 hidden layers 每層 128 個 hidden neurons，activation function 除了 output layer 為 softmax 外其餘皆為 ReLU。由於實驗後半部畫圖很花時間，因此採用 Adam Optimizer (learning rate = 0.001)，以求快速達到訓練完成的 model。總共訓練了兩個不同的 model，兩者的設計全部都一樣，只差在訓練時的 batch size 分別為 64（以下稱 model 1）和 1024（以下稱 model 2）。

(2) Plot the figures of both training and testing, loss and accuracy to the number of interpolation ratio.



(3) Comment your result.

根據作業投影片提供的公式：$\theta_\alpha = (1-\alpha)\theta_1 + \alpha\theta_2$，以 model 1 和 model 2 的參數做線性組合所得到的新的參數，分別去計算 training 和 testing data 的 loss 和準確度。以下幾個簡單的觀察。首先大概可以發現，即使兩個 train 的好的 model，兩者之間並沒有直接的線性關係，a 的組成來源愈單一，則其表現愈好。其次是 batch size 較大的 model 在 training 和 testing data 上的表現幾乎一樣，但可能效果比較差；但 batch size 較小的 model 雖然 testing 的表現比 training data 的表現要差，可能發生 overfitting，但最後達到的效果確實會比較大 batch size 所訓練出來的 model 要來得好。最後一點，可以觀察＝-1 和 ＝2 時，a 的表現落差。雖然直觀上都是兩倍的某一個 model 參數減去另一個 model 的參數，但卻可以發現，＝-1 時的表現遠比＝2 時的表現要好得多。推測原因可能是因為 batch size 較小所訓練出來的 model 處於谷底較寬的 local minimum，即使被另外處於另一個 local minimum 的 model 參數影響位置，最後還是能盡可能的處在 local minimum 的低谷，而有較佳的表現。此實驗基本上驗證了 batch size 比較小所訓練出來的 model 可能會有比較好的效果（不僅止於表面上衡量的數據還有其處於 local minimum 的位置品質）。

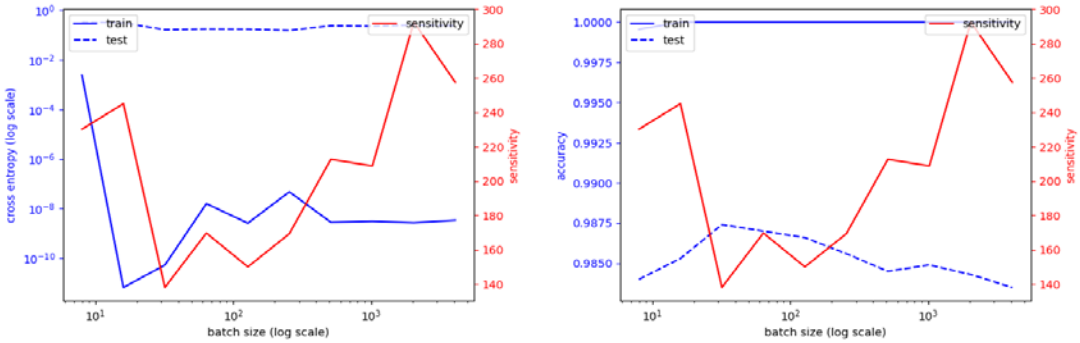
Part 2

(4) Describe the settings of the experiments (e.g. which task, what training approaches)

使用 MNIST dataset，模型架構為兩層的 CNN 當作 hidden layer，一層的 DNN 當作 output layer，其中 CNN 的 filter size=(3, 3), stride=(1, 1)。Activation 除 output layer 為 softmax 外，其餘皆為 ReLU。實驗採用 10 種不同的 batch size 當作變數，來評估不同 batch size 下的 generalization。每個實驗皆訓練到連續 10 個 epoch，training loss 不再下降才停止。

另外這邊使用 sensitivity 的定義為 outputs 對 inputs 微分的 Jacobian Matrix，再取 2-norm。

(5) Plot the figures of both training and testing, loss and accuracy, sensitivity to your chosen variable.



(6) Comment your result.

首先除了 batch size=8 的第一個模型的 training accuracy 略低外，其餘 batch size 的模型皆達到 100%，所以可以視為每個模型皆能幾乎完美的 fit training data。而由 testing accuracy 可以觀察到，除了 batch size=8, 16 的兩個點外，大致上呈現，batch size 越小，testing accuracy 越高，generalization 的能力越好。這邊定義的 sensitivity 代表 x 變化對 y 的影響，所以 sensitivity 越高，代表 y 越容易受到 x 小量的變化影響，導致預測錯誤，generalization 的能力越差。右上圖的 sensitivity 與 testing accuracy 也成負相關，與理論相符。

而左上 loss-batch size 圖，雖然 testing loss 在 log scale 下不易觀察，但是也可發現 sensitivity 越小的地方，testing loss 多半也較小，符合上段落 accuracy 變化的推論。而 training loss 也可以得知，在 model 對 training data 幾乎有百分之百的預測能力的情況下，越低的 training loss 反而會造成 testing loss 上升、testing accuracy 下降、sensitivity 提高，都意味者 generalization 的能力變差，類似 overfit 的情況產生。

而關於 batch size=8, 16 的兩個點 generalization 的能力較差，有可能是因為當 batch size 較小的時候，訓練過程容易震盪，training loss 的變化大，所以 early stop 的 period 設為 10 還不能保證 training 到了一個穩定的參數，所以與後面 8 個模型前提不相同，出現誤差。不過也有可能是因為 batch size 不是絕對且唯一影響 generalization 的因素，只能提供一個大致的參考方向，所以為合理誤差。

Bonus

(7) Use other metrics or methods to evaluate a model's ability to generalize and concretely describe it and comment your results.

IV. 分工表

B03901145 郭恆成: 1-1.1, 1-1 bonus, 1-2.3, 1-3.3 part2

B03901101 楊其昇: 1-1.2, 1-1 bonus, 1-2.1, 1-2 bonus, 1-3.1, 1-3.3 part1

B03091065 林宣竹: 1-1 bonus, 1-2.2, 1-3.2