

# MLDS HW1 Report

組員：b04901060 電機三 黃文璉 分工：1-1-1, 1-1-2, 1-2-1, 1-3-1, 1-3-2, 1-3-Bonus  
b04901003 電機三 許傑盛 1-2-1, 1-2-2, 1-2-3, 1-2-Bonus, 1-3-3-1, 1-3-3-2  
b04901096 電機三 蔡昕宇 1-1-1, 1-1-2, 1-3-Bonus, Report typesetting

## 1-1 Deep vs. Shallow

### 1-1-1 Simulate a function

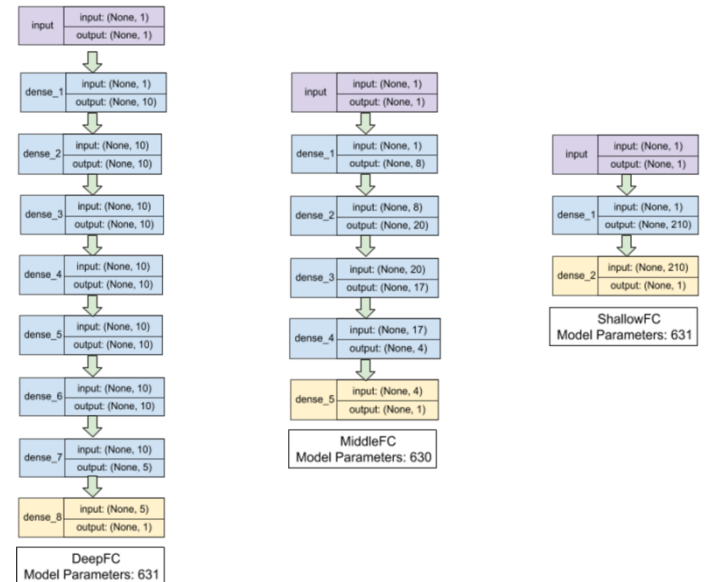
- Describe the models you use, including the number of parameters (at least two models) and the function you use. (0.5%)

本次使用的模型

(DeepFC、MiddleFC、ShallowFC)，

架構和參數量如右圖：

三種模型的參數量幾乎相同。



實驗用到的兩種函數分別為：

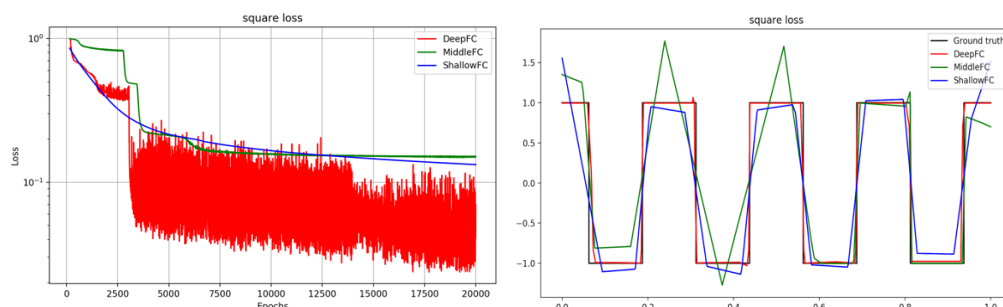
- $square(x) = \text{sgn}(\cos(8\pi x))$
- $damp(x) = 10\sqrt{x}e^{-8x} \sin(6\pi(x - 0.25))$

每個函數 sample 2048 個點。訓練時的 optimizer 為 Adam (lr=1e-3)，batch size 為 128。

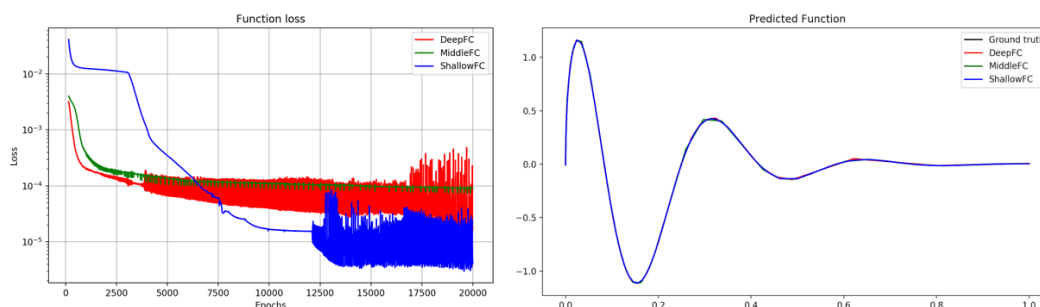
- In one chart, plot the training loss of all models. (0.5%)
- In one graph, plot the predicted function curve of all models and the ground-truth function curve. (0.5%)

2. 3. 的實驗結果如下：

(1)  $square(x)$



(2)  $damp(x)$



#### 4. Comment on your results. (1%)

這次實驗將三種不同深度 (hidden depth = 1, 4, 7) 的模型，應用在兩個函數上。

以 Loss function 來看，實驗 (1) 的  $\text{square}(x)$  方波函數 loss 最低的為 DeepFC，其次為 ShallowFC，再來才是 MiddleFC。可以注意到的是最深的模型 loss 最低，但其中 shallow 卻比 middle 要好一些，故其實沒辦法斷定深度和模型 capacity 之間的關係。而可以再看到 DeepFC 在約第 3000 個 epoch 的時候遇到一個 local minimum，loss 瞬間下降，loss 在那之後震盪變得更加劇烈。

而在實驗 (2) 的  $\text{damp}(x)$  函數可以注意到的是，一開始 MiddleFC 與 DeepFC 的 loss 下降較快，而 ShallowFC 在約 3000 多 epoch 時找到更低的 loss，曲線開始迅速下降，甚至最後比其他兩者還來的低。

但就 model 預測出來的函數值來說，實驗 (1) 的方波函數以 DeepFC fit 的最好，ShallowFC 和 MiddleFC 皆有些地方是以一直線的地放預測一個方波，相對 fit 的不是很好，符合我們對 deep model 能做出較多段的線段的假設。實驗 (2) 則相對較平滑，三個 model 在 20000 epochs 之後 fit 的狀況都相對不錯，但取中間的 model 來預測時我們發現 shallow 確實 fit 的速度較 deep 還要慢，用來 fit 的線段數較 deep 的少。

故綜合以上兩個實驗，我們可以得知 Deep 的 network 有較好的 fit 能力，而以 loss 來看可能這樣的模型稍微看不出結果，但下降的趨勢確實有符合預期。

#### 5. Use more than two models in all previous questions. (bonus 0.25%)

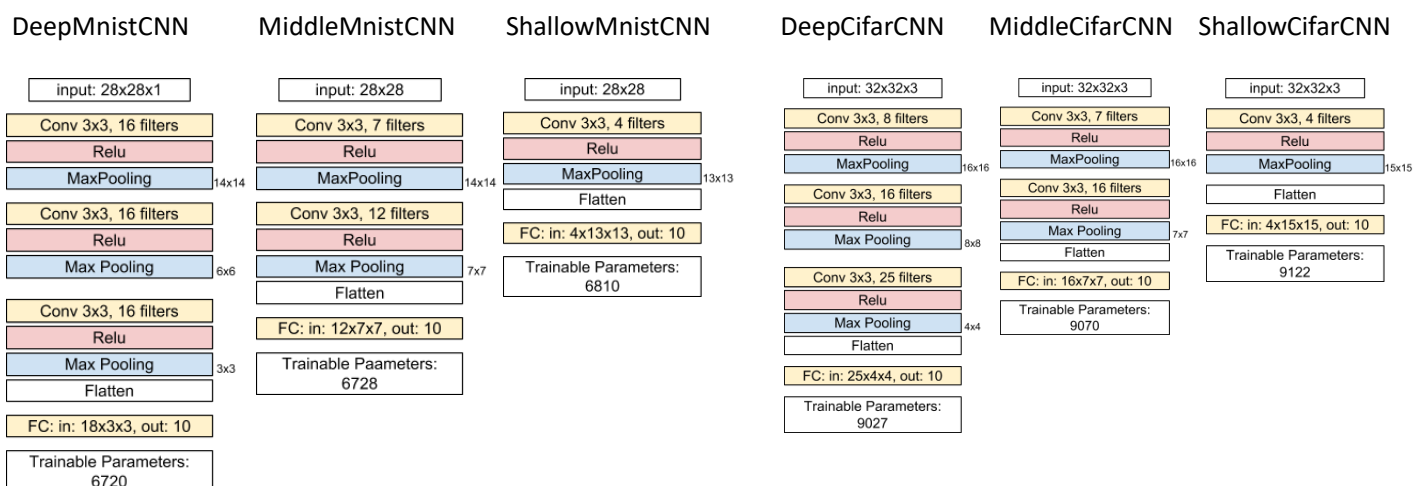
#### 6. Use more than one function. (bonus 0.25%)

我們使用了 3 種不同深度的模型，以及 2 種函數，符合 5. 6. 兩個 bonus 的要求。

### 1-1-2 Train on actual tasks

#### 1. Describe the models you use and the task you chose. (0.5%)

本次訓練在 MNIST 和 CIFAR-10 上，共 6 個模型和參數量如下：



訓練時的 optimizer 為 Adam (lr=1e-3)，batch size 為 128。

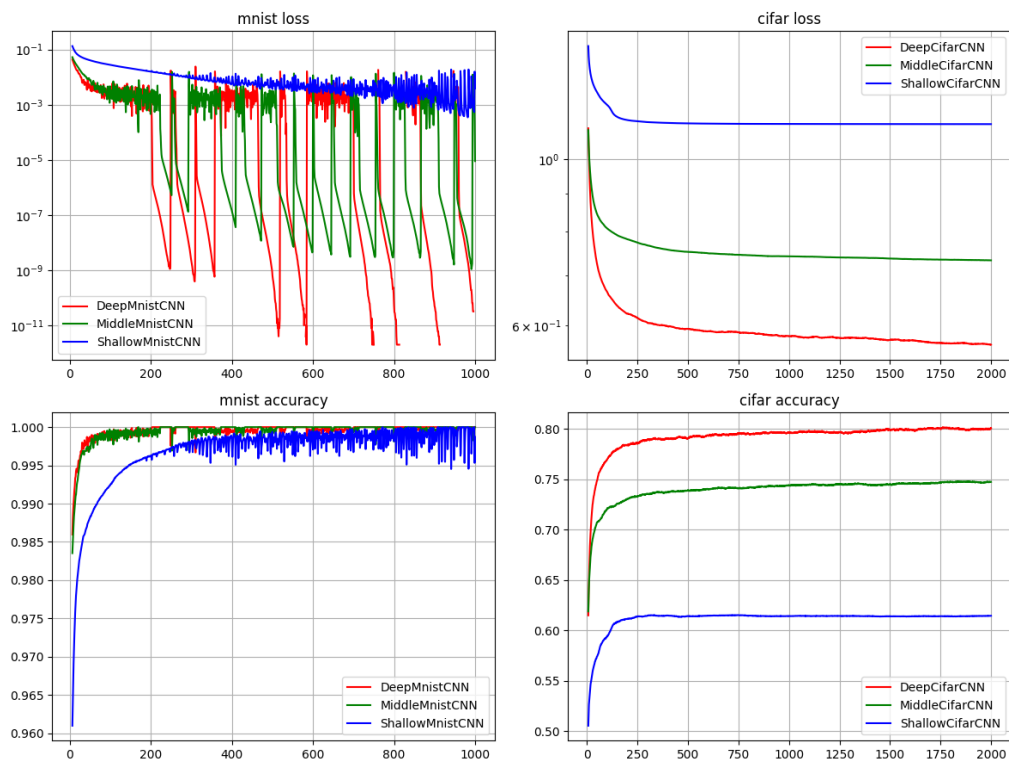
MNIST 訓練 1000 個 epoch，CIFAR-10 訓練 2000 個 epoch。

針對同一個 task 的模型其參數量差異在 1% 左右，符合題目要求參數量接近的模型。

2. In one chart, plot the training loss of all models. (0.5%)

3. In one chart, plot the training accuracy. (0.5%)

2. 3. 請參考下圖，左邊為 MNIST 的 training loss 和 accuracy，右邊則為 CIFAR-10。兩個 dataset 都分別訓練在三種不同深度且參數量相同的模型上。



4. Comment on your results. (1%)

上述分別在 MNIST 與 CIFAR-10 作不同深度的 model 的實驗。針對不同 model，其準確率以及 loss 對 epoch 的作圖。

從 loss 來看，可以明顯看到 ShallowCNN 的 model 在兩個 dataset 上的 training loss 下降的比其他兩者慢，下降的幅度也較小。再來就是 DeepCNN 和 MiddleCNN，在 MNIST 上的下降幅度的差異較不大，可是在 CIFAR-10 上可以看到 DeepCNN 下降的最快。這裏 loss 的結果可與前面 function 的實驗作相互比對，得到一致的發現。

而 accuracy 的部分，首先在 MNIST 上，DeepCNN 與 MiddleCNN 的準確率結果差不多，而 ShallowCNN 的結果則較低，且上升接近到 1.0 的速度較慢。在 CIFAR-10 上更是明顯，可以看到 deep 的結果比 shallow 的結果好很多。另外，亦有可能是 ShallowCNN 這次的結果剛好落在 saddle point 上，才導致 loss 較高，accuracy 特別低的結果。

5. Use more than two models in all previous questions. (bonus 0.25%)

6. Train on more than one task. (bonus 0.25%)

我們使用 3 種不同深度的模型，以及 2 種 dataset，且固定三種模型的參數量，符合 5. 6. 兩個 bonus 的要求。

## 1-2 Optimization

### 1-2-1 Visualize the optimization process

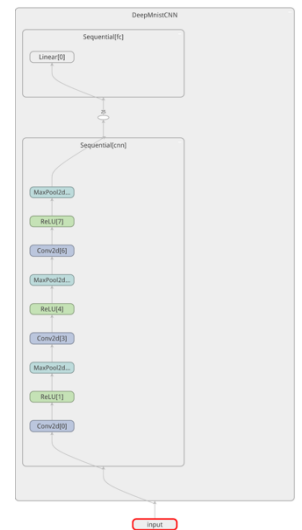
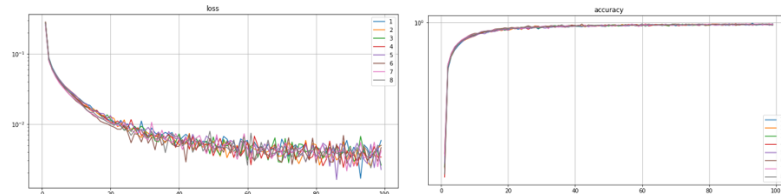
1. Describe your experiment settings. (The cycle you record the model parameters, optimizer, dimension reduction method, etc.) (1%)

我們所使用的模型與 1-1 中訓練在 MNIST 上所所使用的 deep 模型一樣，其參數量為 6720。以此模型反覆訓練 8 次，每 3 個 epoch 紀錄一次模型，每次訓練 99 個 epoch，optimizer 使用 Adam ( $\text{lr}=1\text{e-}3$ )，loss 使用 cross entropy 作為目標函數。

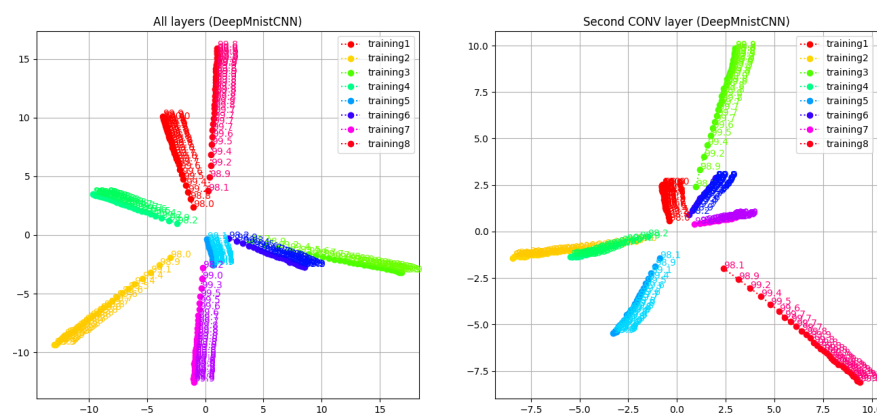
模型架構如右圖，訓練結束後我們分別對各個時間點紀錄的模型的所有 weight 以及第二層的 CONV 作為 vector 展開，並對其作 PCA 降維至 2 個維度，並標上該模型對應的 accuracy 以觀察在 training 過程中 weight 的變化模式，以及其對 accuracy 的影響。

## 2. Train the model for 8 times, selecting the parameters of any one layer and whole model and plot them on the figures separately. (1%)

下圖為 training 過程中 loss 以及 accuracy 的變化：



降維後實驗結果如下圖，左圖為對所有 weight、右圖為對第二層的 CONV 做 PCA 降維至二維，其上標注的數字為該時間點模型的 accuracy：

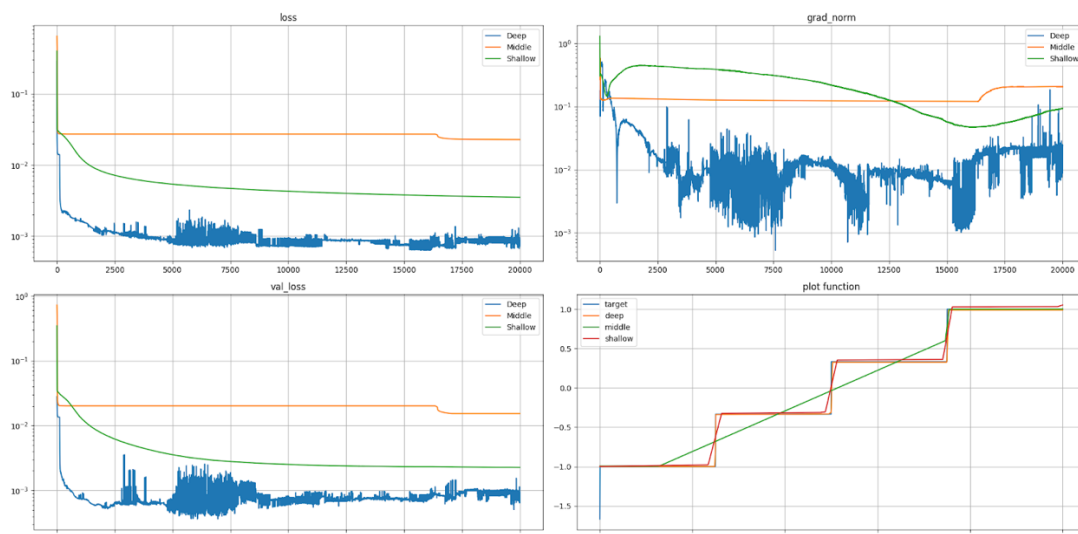


## 3. Comment on your result. (1%)

觀察上面的圖，我們可以發現當 weight 收斂到的地方雖然不一樣，但是卻有差不多的 loss，與老師上課說的不同 local minimum 具有相似的 loss 的情況不謀而合。不過 PCA 的降維方法本來就是最大化不同 vector 之間的歧異程度，這樣的實驗並不太能說明 training 過程時在高維空間中的移動情況為何。

## 1-2-2 Observe gradient norm during training

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

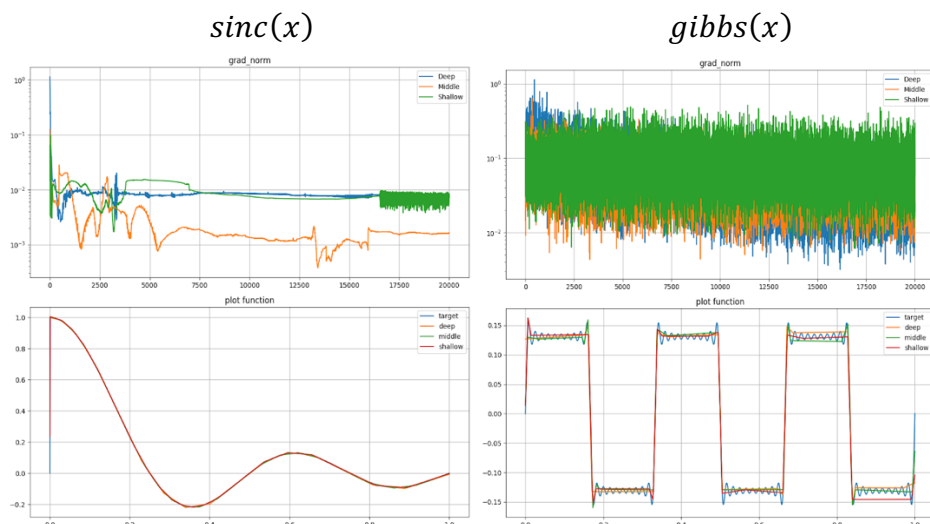


本題使用的函數為階梯函數  $stair(x) = \lfloor 4x - 2.5 \rfloor / 1.5$ 。

圖中左上為 training loss、左下為 validation loss、右上為 gradient norm、右下為 function。

## 2. Comment your result. (1%)

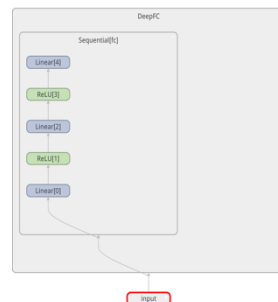
這次的實驗中，我們發現很多的模型在 training 時已經到達一個近似收斂的情況時，gradient norm 卻完全不是那一回事，僅有少數模型能夠收斂到一個比較小的值，如上圖的 stair function，但是大部分的模型卻都比較像右下圖 gibbs 的情況，gradient norm 會在一個很大的值與蠻小的值上下劇震。在 CIFAR-10 和 MNIST 都有這樣的現象，在函數的情況則會根據函數本身的特性有不同的情況。簡單區分的話，若函數越有在小區間內大規模的斜率變化，gradient norm 越會震盪，反之則叫容易收斂，如下圖：



### 1-2-3 What happens when gradient is almost zero?

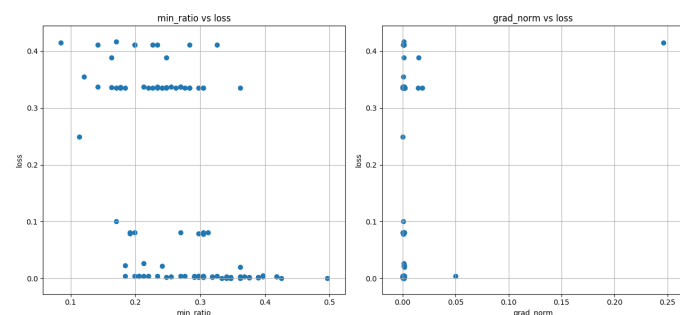
#### 1. State how you get the weight which gradient norm is zero and define the minimal ratio. (2%)

本題使用的模型參數量為 141，詳細架構如右圖，訓練在  $\sin$  函數上，總共 128 筆資料，訓練使用的 optimizer 為 Adam，先訓練 20000 個 epoch，再將 objective function 換為 gradient norm，再 train 20000 次，若 gradient norm 低於設定的 threshold ( $1e-3$ ) 一定次數，則視它為已收斂。至於 minimum ratio 的計算，我們計算該模型的 hessian matrix，並取其 eigenvalue 為正值的次數除以總參數量作為 minimum ratio。Hessian 的計算我們則不如投影片上只取用 Hessian 的對角，而是利用 PyTorch 的 autograd.grad 對模型參數確實進行兩次微分，得到 H。



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

如下圖，左圖為 loss – minimum ratio，右圖為 loss – gradient norm



#### 3. Comment your result. (1%)

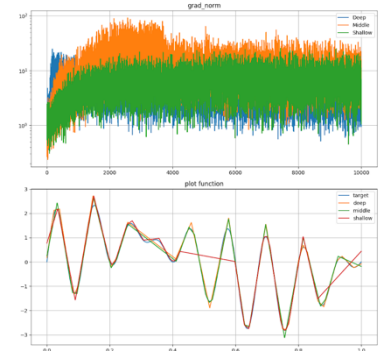
從上面左圖觀察，大致上的趨勢為收斂到的 loss 越低，其 minimum ratio 則越高。雖然在同樣 loss 的情況下，minimum ratio 還是佔了很大一部份的範圍。再者由右圖可以確定，這些模型是停在一個 gradient norm 相當小的情況而這些模型收斂到的地方具有很多能收斂到同樣 loss 的地方，正好驗證了老師上課所說的模型特性，模型可能具有多個能夠收斂到相似 loss 的 local minimum。



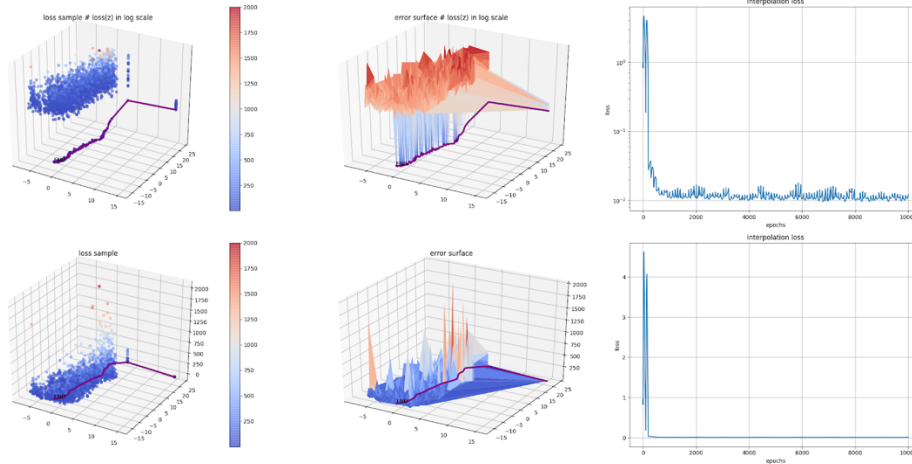
## 1-2-4 Bonus: Error surface

### 1. Use any method to visualize the error surface.

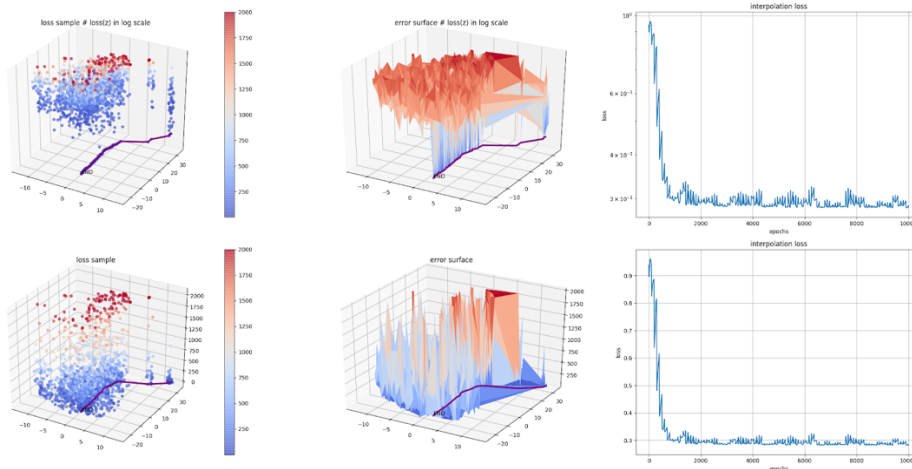
我們將實驗做在參數量較少的 fully connected model 上，總參數量 360，訓練時的 optimizer 為 Adam，batch size 64，實驗對象為右圖的多個正弦函數的加總。由圖我們可以知道此模型在訓練過程中，gradient norm 是比較處於震盪狀態的。稍微比較可以發現，middle 的模型振動幅度較大，而 deep 相對小。下圖中，上排都是取過對數後再作圖。



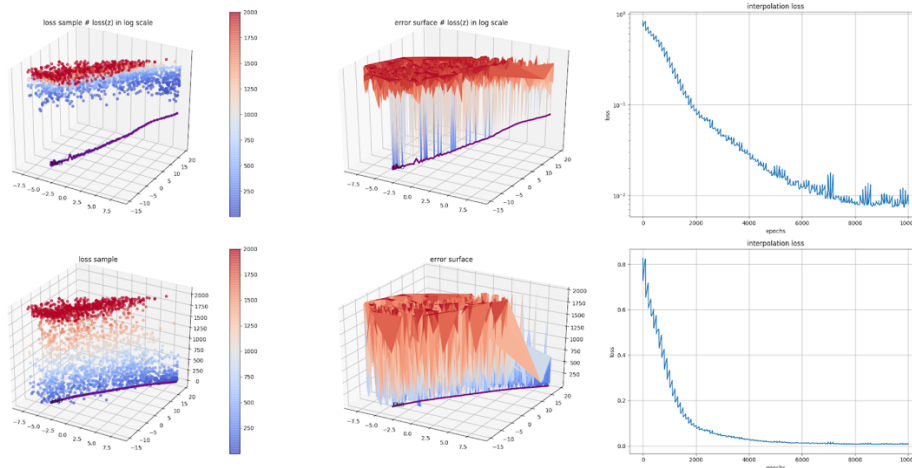
#### (a) DeepFC



#### (b) MiddleFC



#### (c) ShallowFC



## 2. Concretely describe your method and comment your result.

我們使用了兩種方法來視覺化 error surface，一種是計算每次紀錄的 model，用他的 weight 展開成向量並在其附近根據 normal distribution 隨機抽樣，並還原成模型並計算他的 loss。第二種是在每次紀錄的模型做內插。

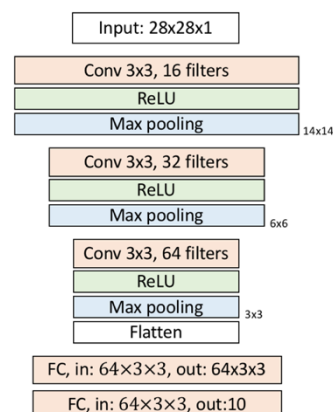
透過這兩種方法，我們可以比較直觀的比較高維空間中 training 過程的情況是如何。幾個結論如下，比較相同參數不同深度的模型，可以發現在結構較深的 error surface 會比較平坦。在先前實驗看到 gradient norm 會劇烈震盪的情況，在高維空間中能看到相當崎嶇平面，而訓練過程彷彿走在海溝中，而 gradient norm 能收斂到較小數值的情況，在高維空間中會是比較平坦的平面。

### 1-3 Generalization

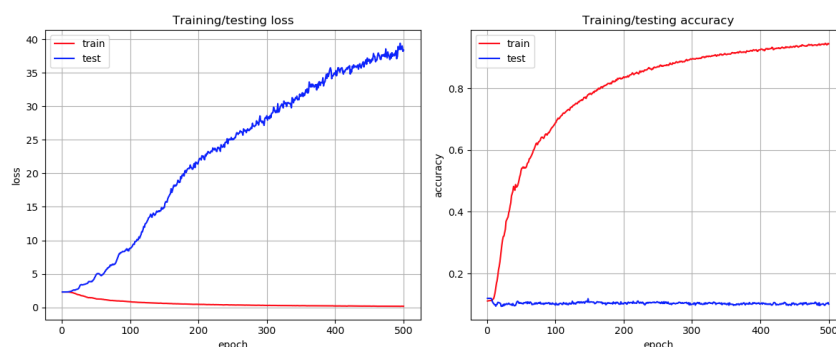
#### 1-3-1 Can network fit random variables?

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

本題使用的資料為 MNIST，模型如右圖，optimizer 為 Adam ( $lr=1e-3$ )，batch size 為 128，總共訓練 500 個 epoch，參數量為 361418，約為 MNIST 訓練資料量的 6 倍。



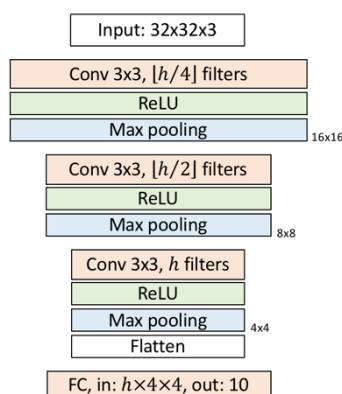
##### 2. Plot the figure of the relationship between training and testing, loss and epochs. (1%)



由上圖可以觀察出，雖然 label 為隨機，但當參數量比資料量大得多時，就有機會讓 network 記住所有 label，也就是 network 的 capacity 夠大，注意到右圖的 training accuracy 接近 1.0。另外由於 label 是隨機，故 testing loss 和 accuracy 顯然不會跟著 training 進步，舉例來說 testing accuracy 基本上維持在 0.1 附近，也就是和隨機猜測的結果是一樣的。

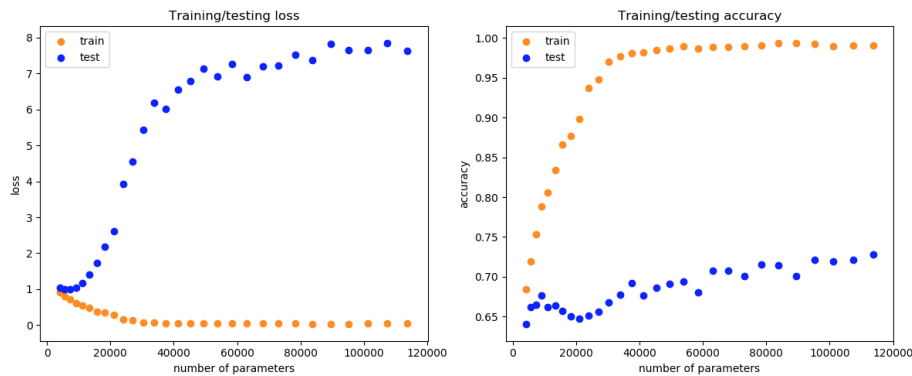
#### 1-3-2 Number of parameters vs. Generalization

##### 1. Describe the settings of the experiments. (e.g. task, 10 or more structures you choose) (1%)



本題使用的資料為 CIFAR-10，模型如左圖，其中  $h$  為變數，範圍為 16~64，batch size 為 128，使用的 optimizer 為 Adam ( $lr=1e-3$ )，每個 model 訓練 200 個 epoch，共 29 個模型。

## 2. Plot the figures of training and testing, loss and accuracy to the number of parameters. (1%)



## 3. Comment your result. (1%)

首先觀察 loss，可以注意到當參數量越多時，training loss 也就隨之變低，但值得注意的是 testing loss 反而變高，也就是參數越多時 overfitting 的情況也就越明顯。

接著觀察 accuracy，顯然當參數量增加時 training accuracy 也隨之增加，值得注意的是雖然 testing loss 不斷增加，但事實上 testing accuracy 卻也逐漸增加，這個現象在許多地方也常被討論<sup>1,2</sup>，簡單來說是由於 accuracy 概念上接近利用 threshold 來判斷類別，舉 binary classification 為例，機率分佈 [0.9, 0.1] 和 [0.55, 0.45] 在 loss 影響很大，在 accuracy 上則沒有影響，故從本實驗也可以得知 accuracy 較不能反映出 model overfitting 的情況。

至於單純從 training loss 和 accuracy 來觀察的話，的確可以得出參數量越多的模型，其 capacity 也就越大的結論。

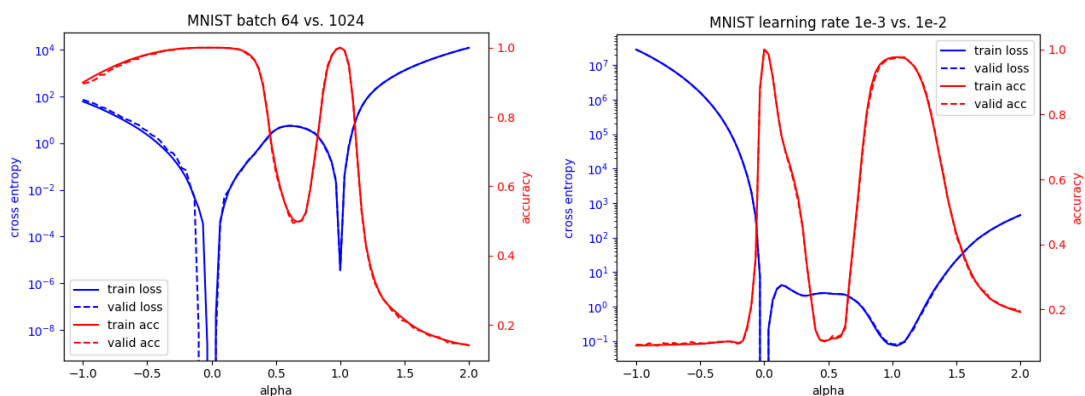
### 1-3-3 Flatness vs. Generalization

#### 1-3-3-1 Visualize the lines between two training approaches

##### 1. Describe the settings of the experiments (e.g. which task, what training approaches) (0.5%)

這次的實驗目標是觀察兩個已經 training 完畢的 model (不同方法)，沿著兩個 model 的方向觀察 error 的變化以及 accuracy 的變化，以探討 model 的 generalize 程度。我們的實驗模型是在 1-1 中使用在 MNIST 上的 DeepMnistCNN 模型，參數量 6720，總共對兩組變量做訓練，一組為 batch size 64 vs. 1024，另一組為 learning rate 1e-3 vs. 1e-2，optimizer 皆使用 Adam (lr=1e-3)。

## 2. Plot the figures of training and testing, loss and accuracy to interpolation ratio. (1%)



## 3. Comment your result. (1%)

實驗結果如上圖，比較 batch size 的部份，如上左圖。我們可以發現在 batch size 比較小的情況下，模型可能更 容易走到一個 loss 比較低的地方，不過在我們實驗結果中，

<sup>1</sup> <https://github.com/keras-team/keras/issues/3755>

<sup>2</sup> <https://stats.stackexchange.com/questions/282160/how-is-it-possible-that-validation-loss-is-increasing-while-validation-accuracy>



batch size 較小的模型反而走到了一個較走感覺上 **generalization** 較差的低點。推測原因為，可能在這方向上的特別陡峭，但其他方向上卻不然。再者由於此繪圖方法是 **log scale**，其實在線性的圖中這兩點看起來都相當的平坦。

接著比較 **learning rate** 的部份，如上右圖。我們一樣可以發現到當 **learning rate** 比較小時，模型一樣走到了一個 **loss** 更低的地方，不過在這個方向上 **learning rate** 比較小的 **model** 反而走到了一個 **sharpness** 較大的地方，推測的原因跟上述的差不多。但是也可能這並不是巧合，還需要更多的實驗來佐證。

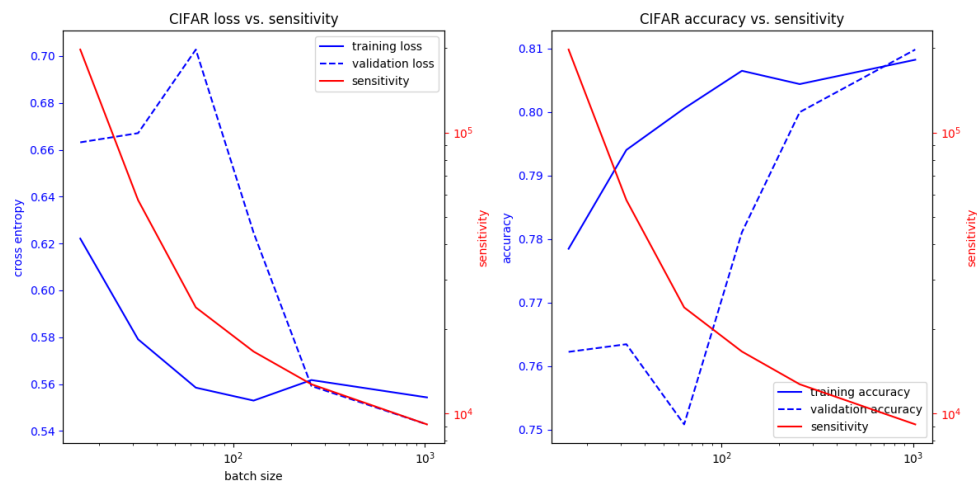
### 1-3-3-2 Visualize the sensitivities of different training approaches

#### 1. Describe the settings of the experiments (e.g. which task, what training approaches) (0.5%)

在本實驗中試著觀察 **sensitivity** 對於 **generalization** 的相關程度，我們將實驗做在 **MNIST** 和 **CIFAR-10** 上，不過由於 **MNIST** 實在太容易 **train** 到準確率都直接封頂，故只展示 **CIFAR-10** 的實驗結果。

實驗模型皆與 1-1 中用於 **CIFAR-10** 的 **DeepCifarCNN** 架構相同，總參數量為 9027，**optimizer** 使用 **Adam** (**lr**=0.001)，總共對六種不同的 **batch size** (16, 32, 64, 128, 256, 1024) 做訓練，每個模型都訓練 2000 個 **epoch**，**sensitivity** 的計算就如投影片上的方法一樣，將所有 **output** 對 **input** 的 **gradient** 平方加總後開根號，即 **Frobenius norm of Jacobian matrix**。

#### 2. Plot the figures of training and testing, loss and accuracy, sensitivity to chosen variable. (1%)



#### 3. Comment your result. (1%)

在實驗結果中，我們可以發現這種方法定義的 **sensitivity** 大致上能夠反應在 **validation** 的表現上，不過細節的部份卻沒辦法充分反應，如上圖中我們可以發現 **validation loss** 在 **batch size = 64** 的地方有上升的現象，不過 **sensitivity** 並沒有辦法表現這邊的轉折趨勢。不過從斜率上觀察的話，可以發現該點的下降斜率有減緩，而且這是用 **log scale** 來繪圖，因此尺度應該會比圖上所示的更大。這也代表著這種定義 **sensitivity** 的方法能夠在一定程度上充分表達 **generalization** 的程度。

### 1-3-3-3 Bonus: Use other metrics or methods to evaluate a model's ability to generalize and concretely describe it and comment your results.

#### 1. Method

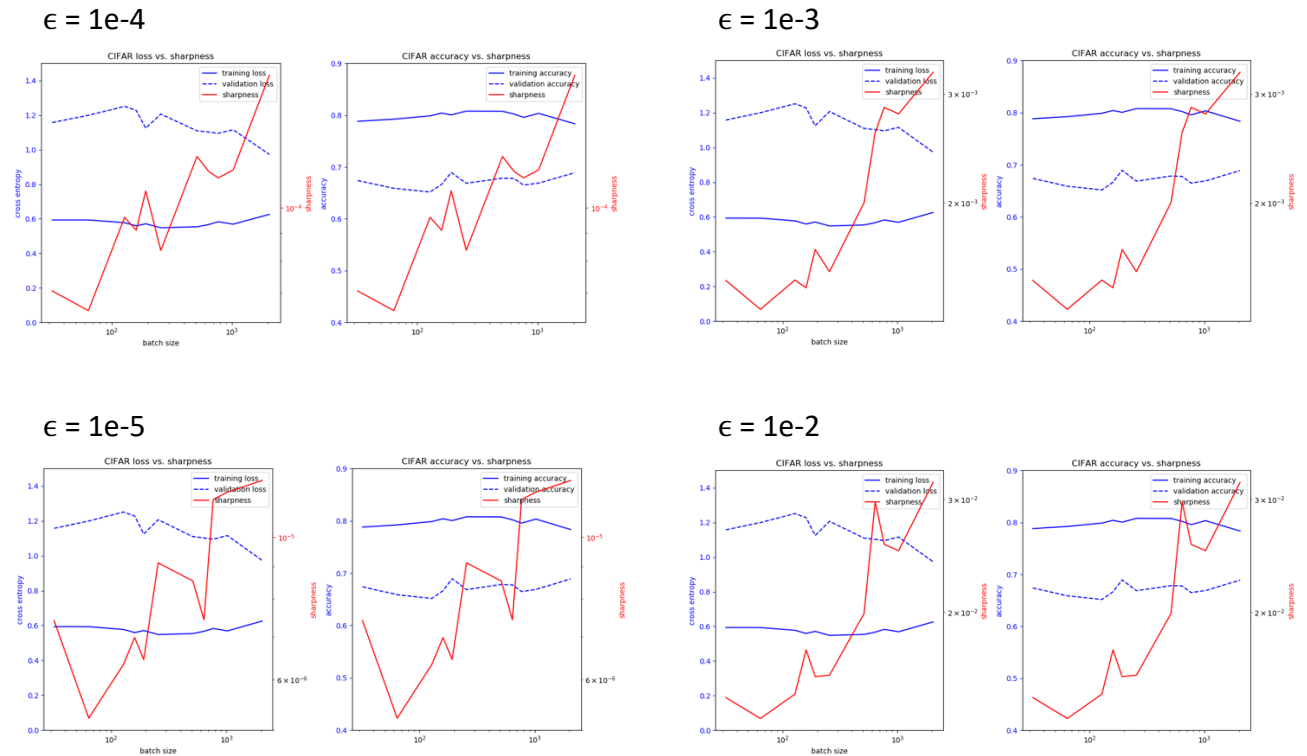
本次使用的方法和投影片上的 sharpness 定義<sup>3</sup>相同，即：

$$\frac{\max_{\theta' \in B_2(\epsilon, \theta)} (L(\theta') - L(\theta))}{1 + L(\theta)}$$

由於此做法需要確實算出 Hessian matrix，雖然根據我們在 1-2-3 中使用的方法可以算出 H，即利用：`torch.autograd.grad(loss, p, create_graph=True)` 但由於 MNIST 和 CIFAR 需要的參數量較大，故計算 H 需要花太長的時間，故我們使用 sample 的方式在半徑為  $\epsilon$  的球中 sample 1000 個點，在其中找出 sharpness 的最大值。

#### 2. Results

本實驗進行在 CIFAR-10 上，模型同 1-1-2 中的 DeepCifarCNN，11 種不同 batch size，每個 batch size 訓練 1000 個 epoch。



#### 3. Comment your result

從實驗結果中我們可以發現在不同 batch size 下，training/testing loss 和 accuracy 大致維持相同，而利用這種方式定義的 sharpness 的趨勢確實有符合在一些論文中提到的，當 batch size 越大則有較大的 sharpness，也就是根據這個 metric，當 batch size 越大則模型 generalize 的能力越差。雖然這次使用 sample 的方式，而不是真正求出極大值，不過可以將實驗結果解讀為，batch size 越大時，模型的 loss 受參數變化的影響傾向大於 batch size 較小的訓練結果。

另外由於  $\epsilon$  的取值可能會影響結果，故我們在這個實驗中採用了四種不同的  $\epsilon$  值來當作半徑，從實驗結果來看，四種  $\epsilon$  都可以觀察出 batch size 越大其 sharpness 越大的趨勢。

<sup>3</sup> <https://arxiv.org/pdf/1609.04836.pdf>