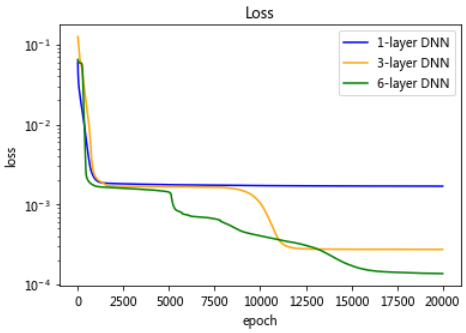
HW1-1 Report Questions:

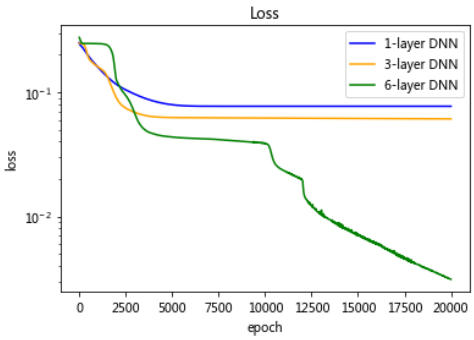
王垣尹 r06921051
許哲瑋 r06921037
李中原 r04921040

Simulate a Function:

- **Describe the models you use, including the number of parameters (at least two models) and the function you use. (0.5%)**
  函數: $y = |\sin(5\pi x)/5\pi x|$，DNN 分別使用了 1 層、3 層和 6 層 hidden layer，參數量分別為 1008、1019、1006。
  函數: $y = sin\ (3\pi x)\ + cos(6\pi x)$ ，DNN 分別使用了 1 層、3 層和 6 層 hidden layer，參數量分別為 1008、1019、1006。
- **In one chart, plot the training loss of all models. (0.5%)**

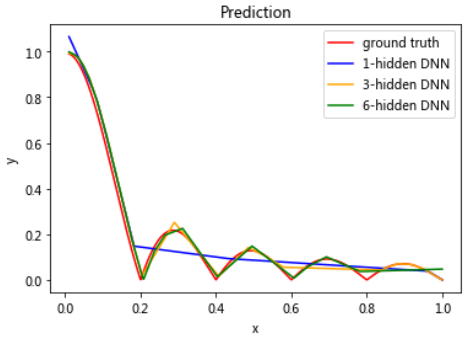


$y = \left|\frac{\sin(5\pi x)}{5\pi x}\right|$

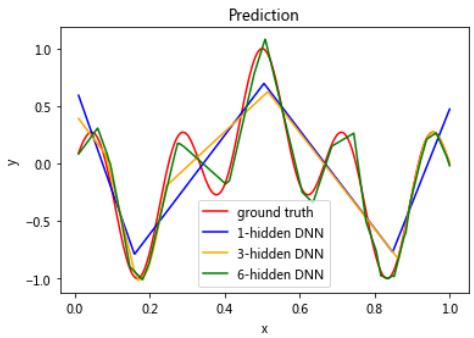


$y = sin\ (3\pi x)\ + cos(6\pi x)$

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



$y = \left|\frac{\sin(5\pi x)}{5\pi x}\right|$



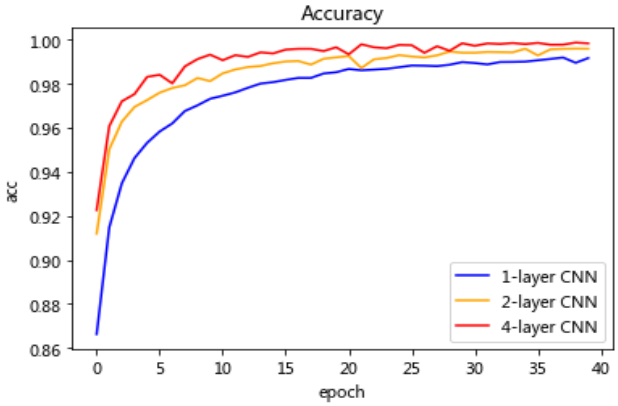
$y = \sin(3\pi x) + \cos(6\pi x)$

Comment on your results. (1%)
可以發現在相同可訓練參數量下，層數越深，loss 的下降可以到越低。而且，深層的神經網路可以更精確的預測函數圖形，不過這樣的前提或許是在Model 表現力已經都充足的情況下,意思是說如果總限制的 trainable variable 量其實很小，或許廣一點的 model 表現會比深一點但每層都只有極少數節點的 model 來得好。

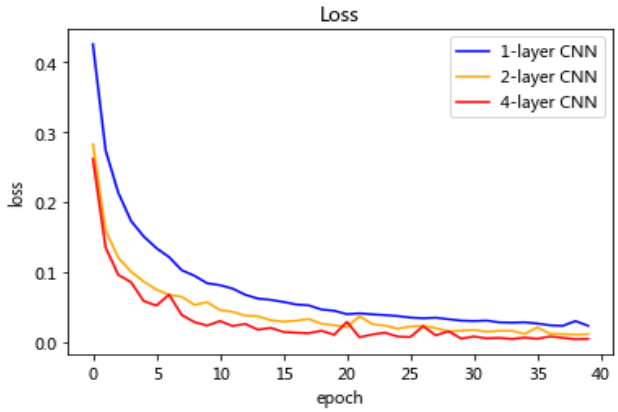
- **Use more than two models in all previous questions. (bonus 0.25%):**
  上面共有有三個 Model。
- **Use more than one function. (bonus 0.25%) :**
  上面共有有兩種 function。

Train on Actual Tasks:

- **Describe the models you use and the task you chose. (0.5%)**
  我們使用 CNN 實現在 MNIST 上，分別疊出 1、2、4 層的 Convolution layer(後面緊接 max-pooling)，Flatten 後接上一層 Fully Connected Layer。
  可訓練參數的數量分別是 90936(1 層)，90956(2 層)，90783(3 層)
- **In one chart, plot the training loss of all models. (0.5%)**



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



- **Comment on your results. (1%)**
  對於 CNN 而言，在可訓練參數差不多的情況下，層數越多(越深)，對於 Loss 的下降或是準確率的提升都能有幫助(我們也嘗試過 DNN，但發現在 DNN 上面，太多層數有時候反而會讓收斂速度比較慢，準確率也會比較差); 當然這樣比較也有些小蹊蹺,第一點是對於 CNN 來說，加入新的 conv layer 實際上增加的參數量並不是線性的而是指數成長(對於每新加一層 conv layer 而言)。但是 Dense Layer 的部分卻是線性成長的，因此對於影像辨識這樣的 task，如果總參數固定，找到一個在 conv layer 以及 dense layer 之間的平衡點十分重要。

- **Use more than two models in all previous questions. (bonus 0.25%) : Three model 3 above.**

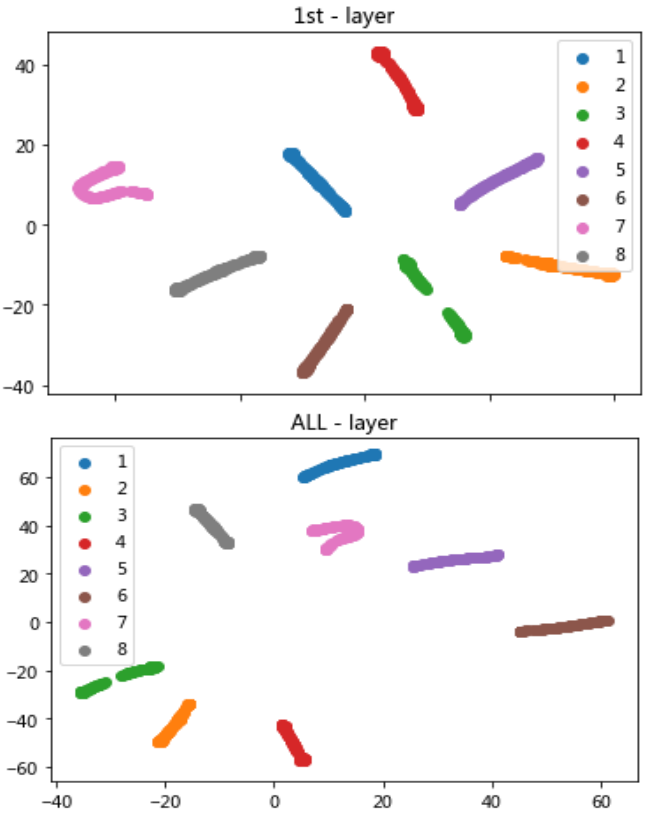
HW1-2 Report Questions:

Visualize the optimization process.

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

- 使用 1-1 小題對方程式 $y = \sin(3\pi x) + \cos(6\pi x)$所做的 DNN model，共有 6 層 hidden layer，每 50 個 epoch 紀錄一次 layer 參數，共訓練 5000 個 epoch，訓練八次(random initialization)。最後使用 t-SNE 降至二維

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



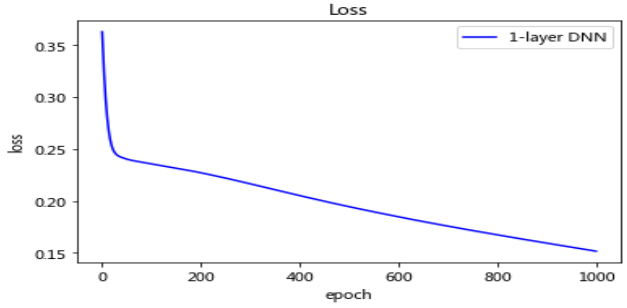
- **Comment on your result. (1%)**

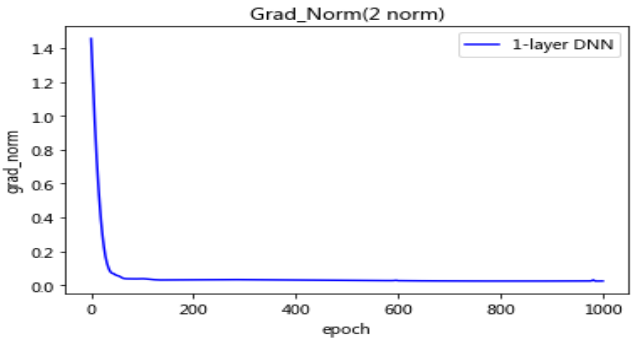
觀察實驗結果可以發現，訓練八次，每次 5000 個 epochs，loss 下降的數量級非常接近，但是參數的更新卻大不相同，隨著每次訓練 initialization 的不同，gradient descent 所走的路徑是非常不同的; 然而，因為 loss 都有降到很低，我們可以推測 loss function 存在許多個 local minimum，對於參數一開始如果起始點設定不同，之後也很有可能會到達不同的 local minimum 位置; 這裡值得討論的議題就是：既然每一個模型都能收斂到不錯的位置，那怎麼樣的初始化(initialization)能使得 loss 較快收斂到理想的 loss。

Observe gradient norm during training.

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

使用方程式$y = sin\ (3\pi x)\ + cos(6\pi x)$，一層 DNN





- **Comment your result. (1%)**

What happens when gradient is almost zero?

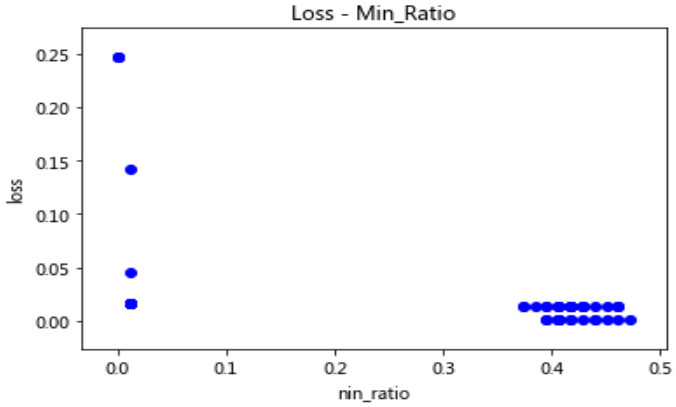
當 梯度范數 (Gradient Norm)接近 0 的時候，我們可以發現 Loss 其實還是不斷的下降，這告訴了我們其實 Loss function 已經到達了相對平緩的地區，如果單純只靠梯度乘上固定的 leaning rate，收斂的速度可能會很慢。又梯度范數本身是 2-Norm 的概念，因此幾乎可以 Imply θ 的1-Norm 也很小；Loss 能持續下降的原因有很多，我們認為比較有可能的是 Loss function 在 minimum 附近接近 Linear(而且 Gradient 也很小)，因此儘管 MSE 是 Convex，還是很可能存在很接近線性的階段使得θ 的絕對值一樣小但是 loss 持續下降。

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

使用函數$y = \left|\frac{sin(5\pi x)}{5\pi x}\right|$作為訓練模型，一層 DNN。因為在 hw1-1-1 中，我們可以發現 loss-epoch 的圖，在約莫 1200 epochs 後 loss 就幾乎收斂到最小值(local min)，因此我們採取訓練 10000 個 epoch，前面 9500 個 epoch 使用 MSE 作為 loss function 下去 train，最後 500 個 epoch 改用 gradient square 作為 loss 訓練(若是使用 gradient norm，可能會因為值接近零開根號而有問題(√0。)

minimum ratio:我們借用上課老師所提到 hessian matrix 的概念，將 loss 對於參數做二階微分算出 hessian matrix，然後計算出 hessian matrix 正的 eigenvalue 所佔的比例，即 minimum ratio。

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

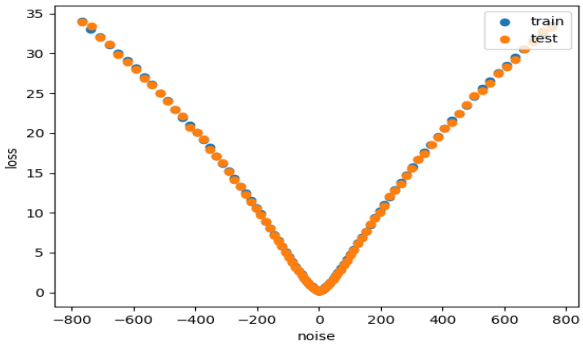


- **Comment your result. (1%)**

在上圖中，我們很容易發現在圖的右下角有一個 cluster，代表 min ratio 高而且 loss 不高的狀況; 另外的一組狀況是在圖左方 min ratio 幾乎成下一條直線部份，這代表了 loss function 實際上是在一個以高頻率震盪的區域(loss 對於θ 來說)，而因為我們要 fit 的函數的確是週期函數，因此這樣的說法比較直觀一點(推導一次 sin 版本的 loss 就會發現)；至於我們比較疑惑的地方在於，既然幾乎收斂，竟然 100 次結果中沒有一次的 minimum ratio 接近 1 的(看起來最後仍然是在鞍點)

Bonus (1%)

- **Use any method to visualize the error surface.**



- **Concretely describe your method and comment your result.**

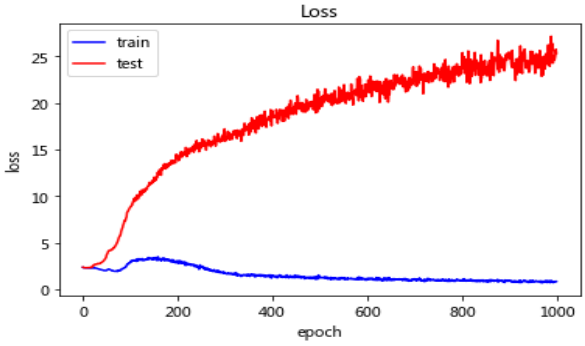
其實我們是把結果降到 1 維，所以只能說是 error curve，但概念上十分類似，就是在對於參數組(θ)加入 Noise，並且在所有維度上取平均；跟助教做的圖最大的不同就是我們的結果而言訓練過程並不是一條線，而是以θ^*為中心，把所有歐氏方向上的等距位置(Hyper-ball)的 Loss 取期望值(E[X]:= $R^M \rightarrow R^1$，即降到 1 維的意思)並對歐氏距離的平方作圖得到的結果；我們認為這樣簡單的做法也能夠看出 Loss Function 的 Sharpness 程度。

HW1-3 Report Questions:

Can network fit random labels?

- **Describe your settings of the experiments. (e.g. which task, learning rate, optimizer) (1%)**
  MNIST, 1 層 CNN + 1 層 DNN(512 neurons)，1000 epochs, learning rate=0.001, Adam Optimizer.

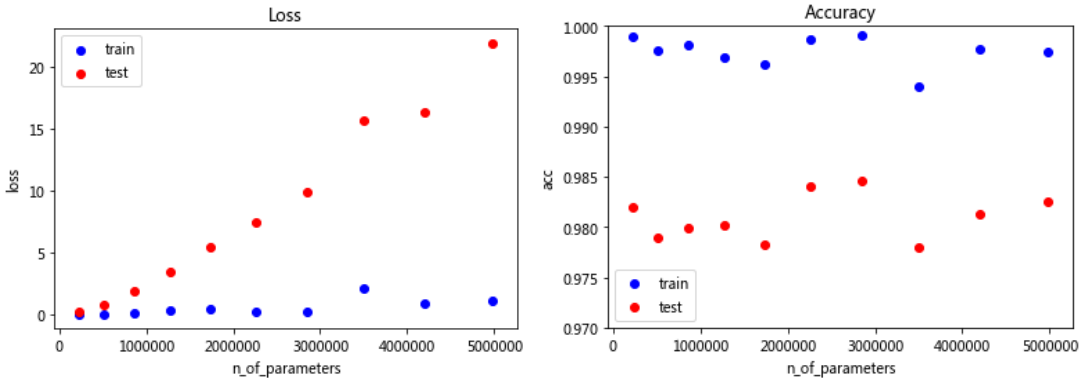
- **Plot the figure of the relationship between training and testing, loss and epochs. (1%)**



Number of parameters v.s. Generalization

- **Describe your settings of the experiments. (e.g. which task, the 10 or more structures you choose) (1%)**
  一樣是 MNIST Classification，4 層 CNN(10,39,39,38 filters) +4 層 DNN(variable)，50 epochs, learning rate=0.001, Adam Optimizer.

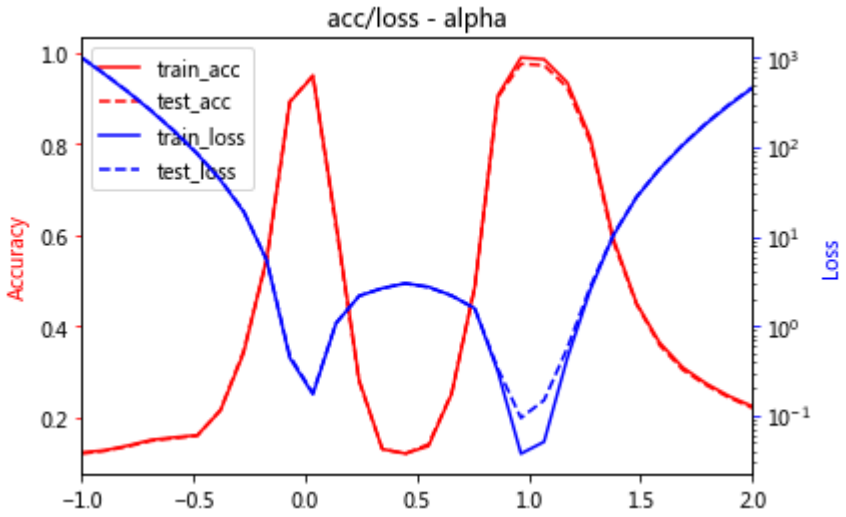
- **Plot the figures of both training and testing, loss and accuracy to the number of parameters. (1%)**



Comment your result. (1%)
我們發現，儘管理論上增加 parameter 容易造成 overfitting，實際對於 accuracy 的影響，至少在我們觀察的範圍內(0->5000000)是不明顯的；然而，值得注意的是，Loss 是不斷上升的，
這告訴我們的是，他的確是在漫漫的 overfit training dataset，只是由於是 classification ，最終只需要選擇一個最大的機率值(argmax)，故或許某個選項在 overfit 的過程已經漸漸不再具有壓倒性，卻仍會被選到不影響 Model 的 accuracy。跟助教結果不太一樣的地方在於，我們 train 出來 testing loss 隨著參數的增加不斷地增加，助教的結果則是仍然有在下降的趨勢。

Flatness v.s. Generalization
Part 1:

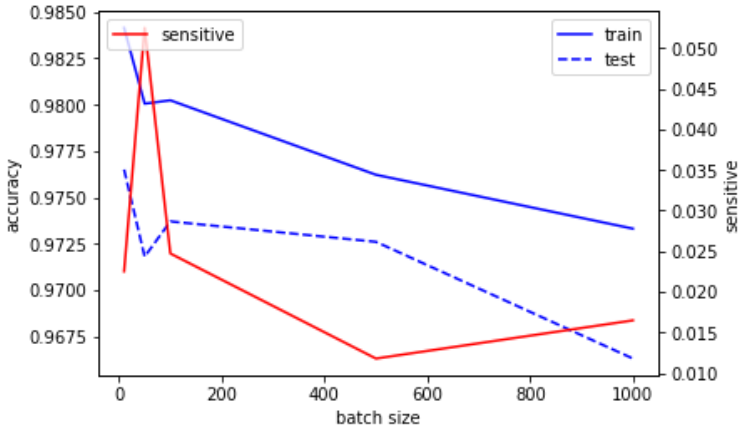
- **Describe the settings of the experiments (e.g. which task, what training approaches) (0.5%)**
  做在 MNIST 上面，一層 CNN+四層 DNN，learning rate=0.001, Batch_Size 分別為 50 和 1000。
- **Plot the figures of both training and testing, loss and accuracy to the number of interpolation ratio. (1%)**

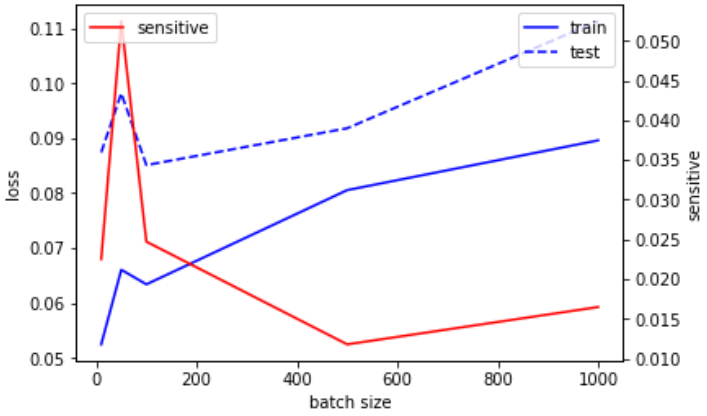


- **Comment your result. (1%)**
- 可以發現對所有參數θ做 Affine Hull [-1,+2]的結果就是，只有在[0] 以及[1]的位置 Accuracy 特別高(Loss 特別低)，這裡我們不難發現，儘管已 Loss 的角度而言，位於 1.0 的參數設定表現比較好，但是在 Accuracy 上沒有太明顯的區別；由此可以呼應上課中的一個猜想:在 NN 中 local minimum 們之間常常十分接近，也很有可能與 Global minimum 很接近。

Part 2 :

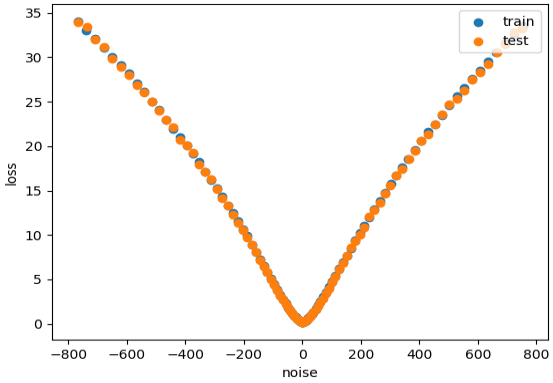
- **Describe the settings of the experiments (e.g. which task, what training approaches) (0.5%)**
  我們 train 在 1-1 做的 function fitting 上，batch size 使用 10、50、100、500 以及 1000；計算 sensitivity 是算 loss 對 input perturbation (Minute shift) 的 gradient 取 f-norm，並挑選最大的值(Loss 在一樣的 Perturbation 情況下變化最多的那一個)。
- **Plot the figures of both training and testing, loss and accuracy, sensitivity to your chosen variable. (1%)**





- **Comment your result. (1%)**
  不難發現，Batch size 大的 model sensitivity 普遍比較低(儘管 loss 不見得比較低)，也就是比較不會受到 input 雜訊的影響而 overfit，理論上的原因或許是當 batch 夠大，各個維度 input 的不穩定被整個 batch 平均稀釋掉了，儘管沒有去做驗證，我們相信如果再另外做 batch normalization，將 input 的資料進一步做一般化，batch 大小的影響將會更加明顯。 這樣類似的問題比較有趣的點就是一般我們算的 gradient 都是在找 loss 對 model 的 parameters 的變化；然而，探討 sensitivity 是針對固定 Model 去觀察它 Generalize 到一般 data 的能力，簡單來說就是固定這些 model 參數來計算 loss 對 input 的 gradient，換個角度想，如果 model 本身是純線性的(假設)，我們就期望每一個參數得值都小一點，這樣微分之後獲得的 gradient 在每個維度上就會比較小(Gradient Norm 也會比要小)，loss 對於受到污染的 data 接受程度也會比較高，可以與在做一些 task(regression…)會加入 Normalization Term 互相呼應，因為有些 train 的方法是沒辦法使用 batch 概念的。

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



Bonus: 上圖表示的是，當我們的 Model 參數θ加入些微的 noise，在找到的 Local minimum 附近

perturb 對 Loss 產生的影響，其中，加入的 noise 是 Normal Distribution 的 Random Noise，我們認為，如果對於每一個$\theta \in \Theta$，加入的 Noise 都遵守 N(0,σ^2)，只要 Sample 的點數 n 夠多，就能一定程度的近似於在θ周圍找 Euclidean ball 並取最大的 Loss Difference (Excessive Loss)，當然這樣的 n 要多大取決於 Model 跟 Task，我們找了 n=20，我們認為這樣能在時間與效果上取得一個不錯的平衡。