HW1

R06946009林庭宇、R06946006李筑真、R06946015黃永翰

1-1 Deep vs Shallow

Simulate a Function:

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

shallow_model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_14 (Dense)             (None, 30)                60
_________________________________________________________________
dense_15 (Dense)             (None, 1)                 31
=================================================================
Total params: 91
Trainable params: 91
Non-trainable params: 0
```

medium_model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 4)                 8
_________________________________________________________________
dense_10 (Dense)             (None, 6)                 30
_________________________________________________________________
dense_11 (Dense)             (None, 4)                 28
_________________________________________________________________
dense_12 (Dense)             (None, 4)                 20
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 5
=================================================================
Total params: 91
Trainable params: 91
Non-trainable params: 0
```

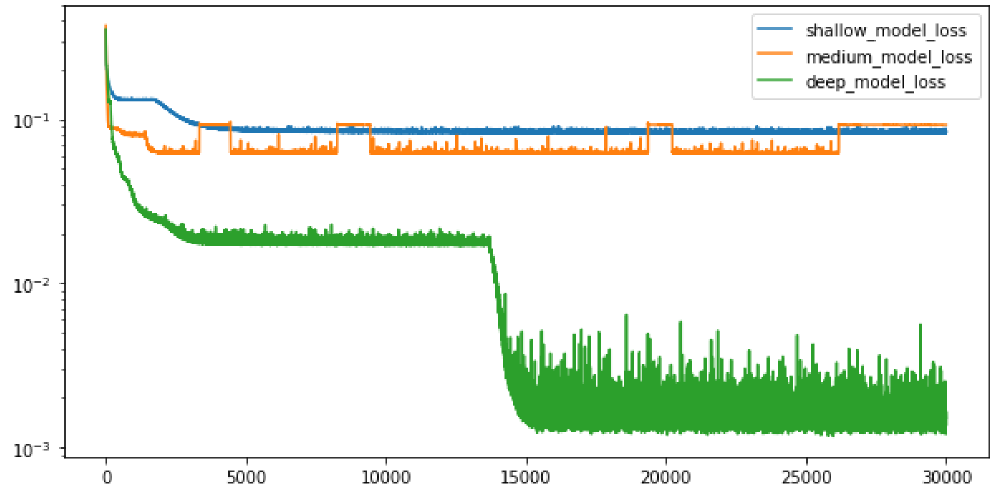
deep_model

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 4)                 8
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 20
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 15
_________________________________________________________________
dense_4 (Dense)              (None, 3)                 12
_________________________________________________________________
dense_5 (Dense)              (None, 3)                 12
_________________________________________________________________
dense_6 (Dense)              (None, 3)                 12
_________________________________________________________________
dense_7 (Dense)              (None, 2)                 8
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 3
=================================================================
Total params: 90
Trainable params: 90
Non-trainable params: 0
```

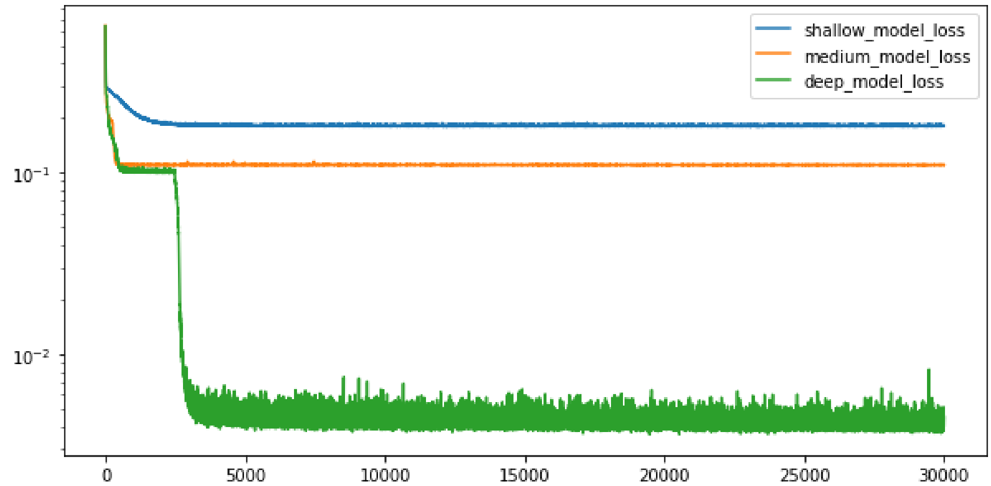
function_1 : $sin(exp(\pi x))$ **function_2 :** $cos(10x^3)$

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

function_1

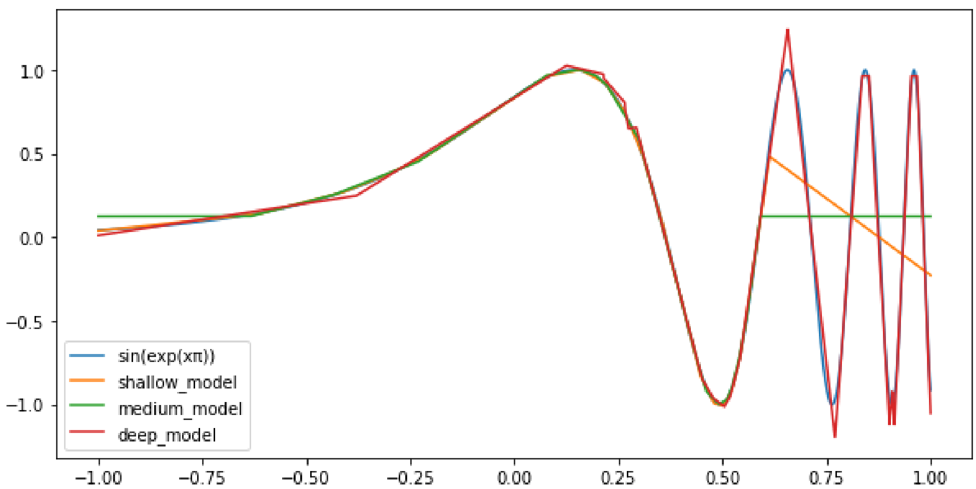


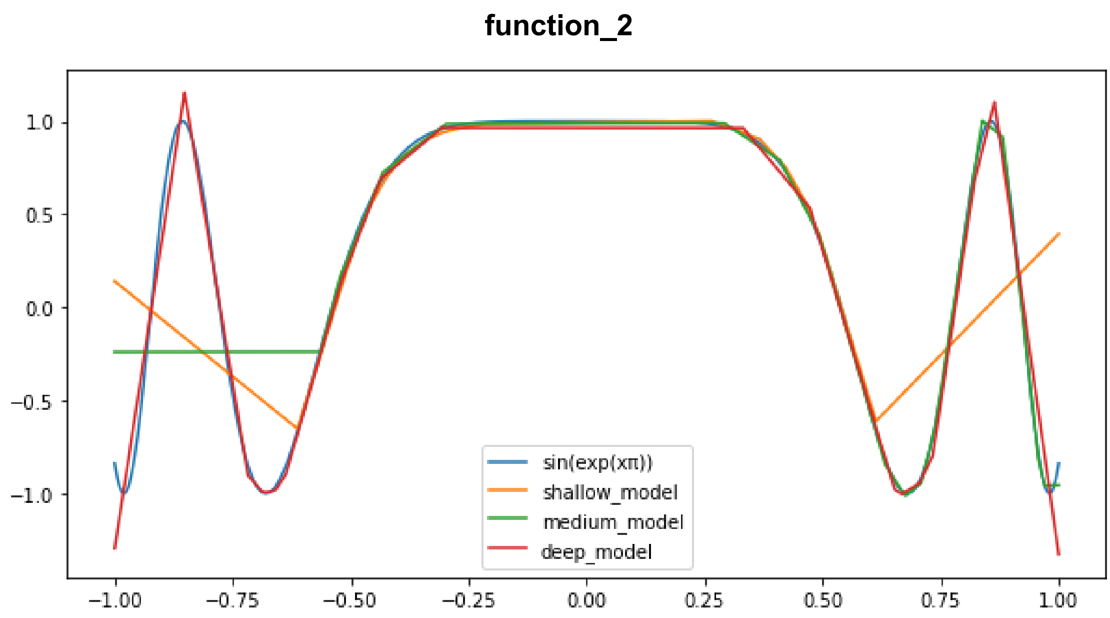
function_2



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

function_1





Comment on your results. (1%)
不管從loss還是function curve來看，都可以發現在參數量一樣的情況下，model越deep所做出來的結果越好

(bonus的第三個model與第二個function已經包含在上面)

Train on Actual Tasks:

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

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 38)	1064
activation_1 (Activation)	(None, 32, 32, 38)	0
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 38)	0
dropout_1 (Dropout)	(None, 16, 16, 38)	0
flatten_1 (Flatten)	(None, 9728)	0
dense_1 (Dense)	(None, 128)	1245312
activation_2 (Activation)	(None, 128)	0
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 10)	1290
activation_3 (Activation)	(None, 10)	0

Total params: 1,247,666
Trainable params: 1,247,666
Non-trainable params: 0

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 16)	448
activation_1 (Activation)	(None, 32, 32, 16)	0
conv2d_2 (Conv2D)	(None, 30, 30, 22)	3190
activation_2 (Activation)	(None, 30, 30, 22)	0
max_pooling2d_1 (MaxPooling2	(None, 15, 15, 22)	0
dropout_1 (Dropout)	(None, 15, 15, 22)	0
flatten_1 (Flatten)	(None, 4950)	0
dense_1 (Dense)	(None, 250)	1237750
activation_3 (Activation)	(None, 250)	0
dropout_2 (Dropout)	(None, 250)	0
dense_2 (Dense)	(None, 10)	2510
activation_4 (Activation)	(None, 10)	0

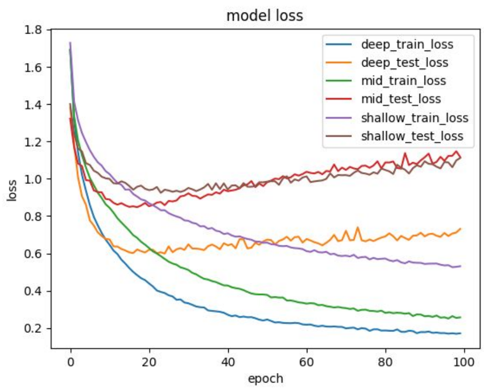
Total params: 1,243,898
Trainable params: 1,243,898
Non-trainable params: 0

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 32)	896
activation_1 (Activation)	(None, 32, 32, 32)	0
conv2d_2 (Conv2D)	(None, 30, 30, 32)	9248
activation_2 (Activation)	(None, 30, 30, 32)	0
max_pooling2d_1 (MaxPooling2	(None, 15, 15, 32)	0
dropout_1 (Dropout)	(None, 15, 15, 32)	0
conv2d_3 (Conv2D)	(None, 15, 15, 64)	18496
activation_3 (Activation)	(None, 15, 15, 64)	0
conv2d_4 (Conv2D)	(None, 13, 13, 64)	36928
activation_4 (Activation)	(None, 13, 13, 64)	0
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 64)	0
dropout_2 (Dropout)	(None, 6, 6, 64)	0
flatten_1 (Flatten)	(None, 2304)	0
dense_1 (Dense)	(None, 512)	1180160
activation_5 (Activation)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130
activation_6 (Activation)	(None, 10)	0

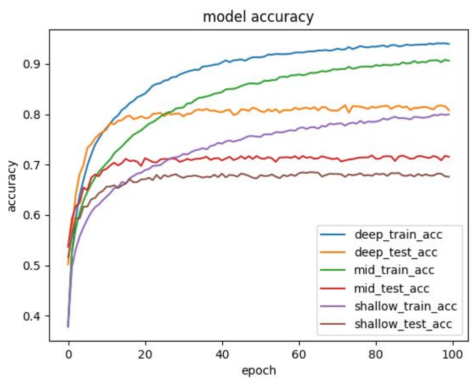
Total params: 1,250,858
Trainable params: 1,250,858
Non-trainable params: 0

為採用CNN模型及cifar10資料集的模型架構，三個模型因應層數由左至右分別為shallow、middle、deep，模型們參數都趨近125萬。

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%)

從cifar 10這個資料集來看，因為他的圖片較為複雜且繽紛，所以相對於MNIST有更高的識別度在deep問題上，由上面兩張圖可以看到從accuracy可以看出deep的模型比其他兩個高，且收斂比較快。在loss上面，deep收斂比較快、比較低。

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

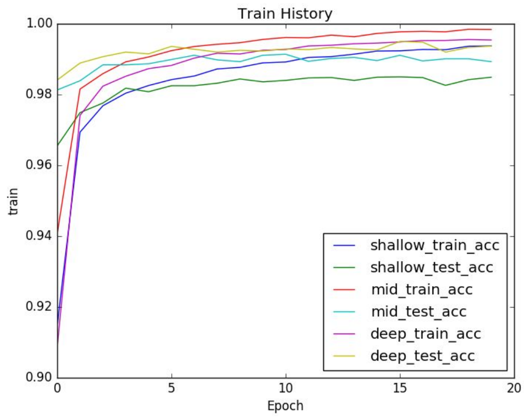
Train on more than one task. (bonus 0.25%)

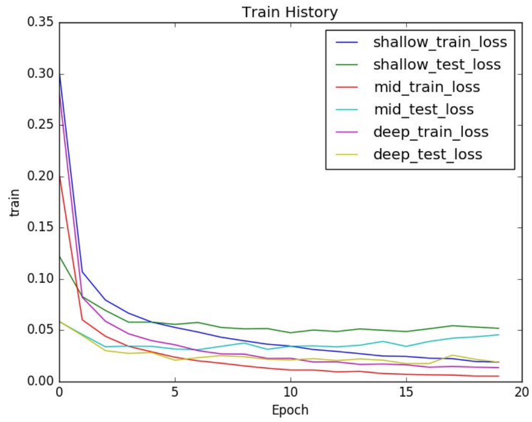
```
Layer (type)                 Output Shape
=================================================
conv2d_1 (Conv2D)            (None, 26, 26, 68)
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 68)
dropout_1 (Dropout)          (None, 13, 13, 68)
flatten_1 (Flatten)          (None, 11492)
dense_1 (Dense)              (None, 10)
=================================================
Total params: 115,610
Trainable params: 115,610
Non-trainable params: 0
```

```
Layer (type)                 Output Shape
=================================================
conv2d_1 (Conv2D)            (None, 26, 26, 64)
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 64)
conv2d_2 (Conv2D)            (None, 11, 11, 64)
dropout_1 (Dropout)          (None, 11, 11, 64)
flatten_1 (Flatten)          (None, 7744)
dense_1 (Dense)              (None, 10)
=================================================
Total params: 115,018
Trainable params: 115,018
Non-trainable params: 0
```

```
Layer (type)                 Output Shape
=================================================
conv2d_1 (Conv2D)            (None, 26, 26, 65)
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 65)
conv2d_2 (Conv2D)            (None, 11, 11, 65)
dropout_1 (Dropout)          (None, 11, 11, 65)
conv2d_3 (Conv2D)            (None, 9, 9, 64)
dropout_2 (Dropout)          (None, 9, 9, 64)
max_pooling2d_2 (MaxPooling2 (None, 4, 4, 64)
conv2d_4 (Conv2D)            (None, 2, 2, 64)
dropout_3 (Dropout)          (None, 2, 2, 64)
flatten_1 (Flatten)          (None, 256)
dense_1 (Dense)              (None, 10)
=================================================
Total params: 115,742
Trainable params: 115,742
Non-trainable params: 0
```

上圖由左而右分別為shallow, middle, deep三個不同model，參數都控制在115,000左右，做在MNIST的dataset上





上圖左邊為各個model的accuracy隨著epoch的變化，右邊則是loss對epoch的變化，由圖中可以發現在training set的部份middle(中等深度)的loss與accuracy均表現最好，而在testing set的部分則均是deep最佳，shallow的model在所有case中都是最差的．

由於參數均控制在115,000左右，因此我們可以發現增加深度確實可以讓model更加powerful，取得更好的結果，而且更深的model對於generation可能是有幫助的，因為最deep的model在testing set上表現是最好的．

1-2 Optimization

Visualize the optimization process.

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

task：Simulate a funtion $sin(exp(\pi x))$

cycle：3

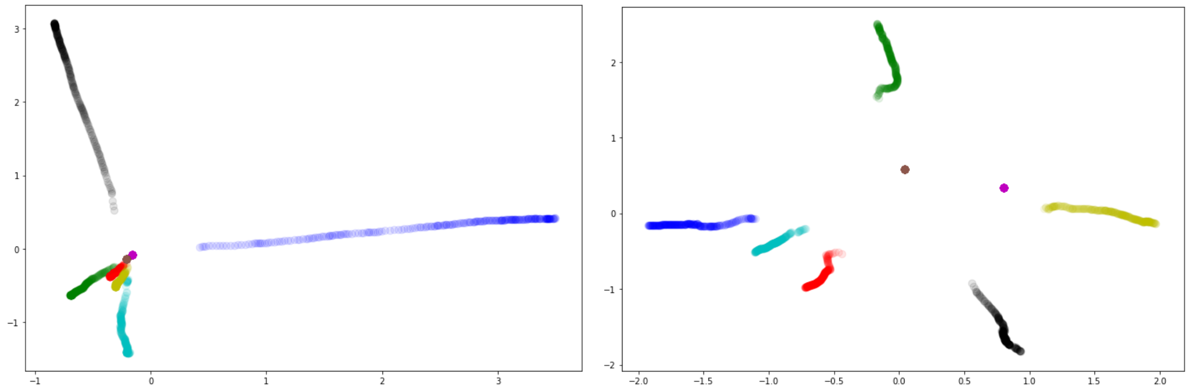
optimizer：adam

dimension reduction method：PCA

model structure：

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 5)                 10
_________________________________________________________________
dense_2 (Dense)              (None, 10)                60
_________________________________________________________________
dense_3 (Dense)              (None, 15)                165
_________________________________________________________________
dense_4 (Dense)              (None, 20)                320
_________________________________________________________________
dense_5 (Dense)              (None, 25)                525
_________________________________________________________________
dense_6 (Dense)              (None, 20)                520
_________________________________________________________________
dense_7 (Dense)              (None, 15)                315
_________________________________________________________________
dense_8 (Dense)              (None, 10)                160
_________________________________________________________________
dense_9 (Dense)              (None, 5)                 55
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 6
=================================================================
Total params: 2,136
Trainable params: 2,136
Non-trainable params: 0
```

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

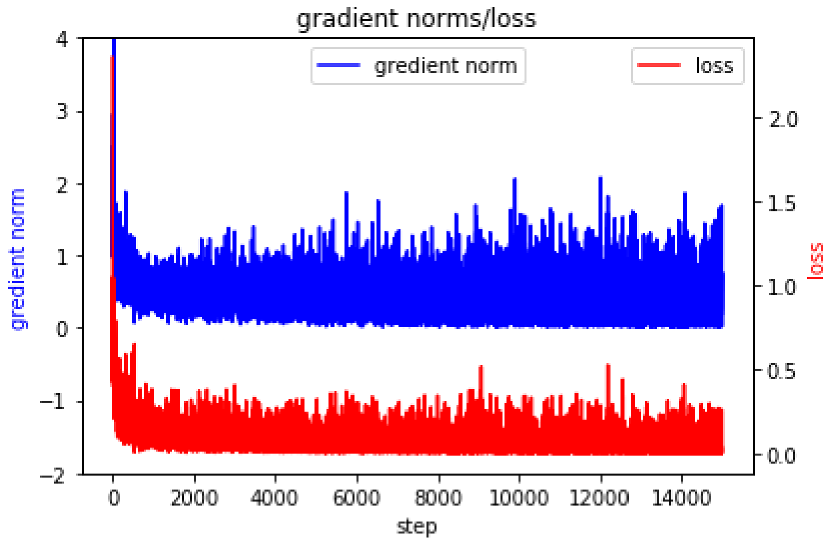


whole model　　　　layer 1

Comment on your result. (1%)
八次訓練都朝不同方向發散，代表每次訓練的參數都收斂到不同地方，而不會每次訓練最終都在同一個minima收斂

Observe gradient norm during training.

Plot one figure which contain gradient norm to iterations and the loss to iterations. (1%)
採用DNN模型和MNIST資料集。

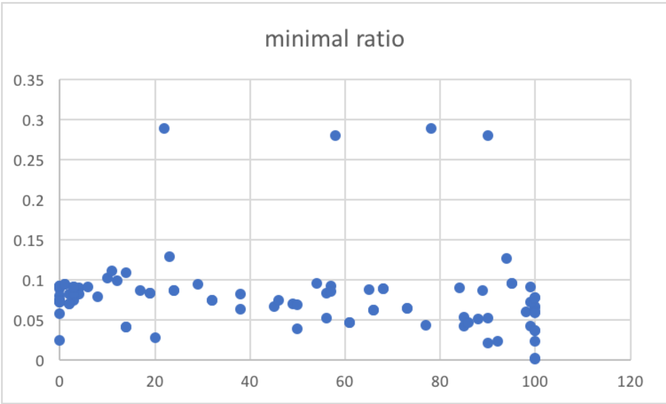


Comment your result. (1%)
從圖片可看到大規模上，norm跟loss是隨著step增加而下降；不過從小範圍觀察，norm值和loss的變動很劇烈，梯度的變化影響著loss變化，，這樣的起伏可能代表訓練中他不斷進入又逃出一個個critical points。另外，loss跟norm的變動其實有相關性（看形狀起伏相似）。

What happens when gradient is almost zero?

State how you get the weight which gradient norm is zero and how you define the minimal ratio. (2%)
首先用一般的loss function作為optimizer最小化的對象，train2500個epoch後改成直接最小化gradient norm，找到gradient norm趨近於零的位置
而這裡所定義的minimal ratio，我們採用sample周邊點的方式．從目前gradient norm趨近於零的附近採樣100個點，分別計算其loss，最後算出這100個點之中有多少比例，其loss大於目前的loss，並以此值作為minimal ratio．
Train the model for 100 times. Plot the figure of minimal ratio to the loss. (2%)

(橫軸為ratio，縱軸為loss)

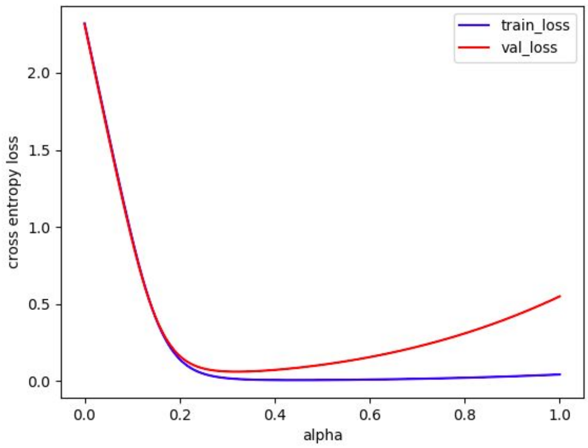


Comment your result. (1%)

由結果來看，採樣周邊點的方式似乎較為不穩定，分布十分凌亂，但整體而言仍然可以看出當loss較低時會有較高的minimal ratio，由此可見在loss較低的時候較有可能是到達local minima，而loss較高的時候則可能是在鞍點．

Bonus (1%)

Use any method to visualize the error surface. Concretely describe your method and comment your result.



方法：採用取起始與終止權重的內插2000個點來畫error surface，針對MNIST資料集訓練DNN模型，遵循原始paper裡頭公式：(1-alpha)*input weights + alpha*input weights。

評論結果：從結果來看，線條挺平滑的，沒辦法看到像助教example裡的抖動的情況，但是我看了Goodfellow論文上mnist的結果，看起來也是較平滑，我認為結果算合理。從論文上提到，如果有個很好的起始點，會使其更容易逃出saddle point和local minima，我想模型應該有一個不錯的initial weights。

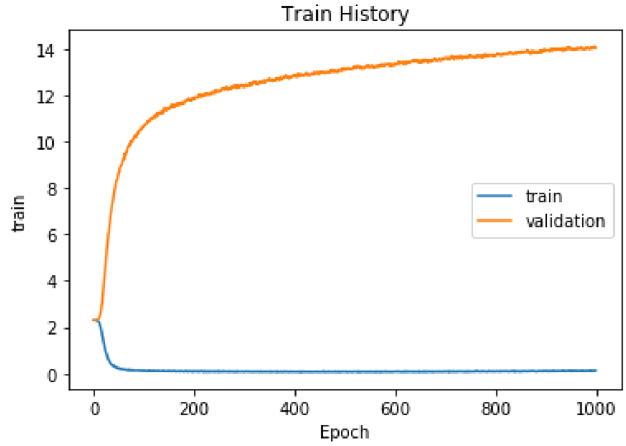
1-3 Generalization

Can network fit random variables?

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

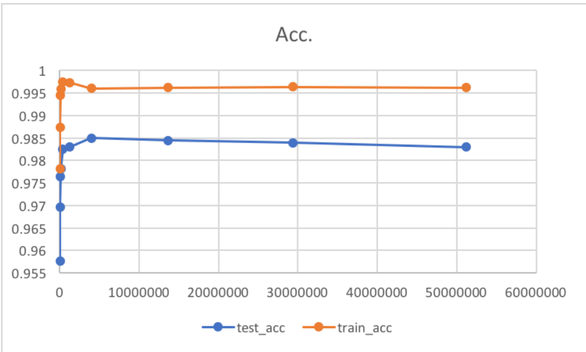
我們的實驗做在MNIST上，將其label打亂後進行訓練，我們使用三層的DNN，每層各有512個neurons，activation function使用relu，optimizer為adam(初始learning rate為0.001)，batch

size為128，大約在530 epochs後就在training set上達到最佳的結果(loss:0.055, acc:0.989)．
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%)
我們的實驗使用MNIST dataset，並使用10個相近結構的model進行訓練，這些model都是四層hidden layer 的DNN，每層的neuron數都相同（而10個model分別採用每層16,32,64,128,256,512,1024,2000,3000,4000個neurons）
Plot the figures of both training and testing, loss and accuracy to the number of parameters. (1%)



Comment your result. (1%)
由數據與圖形可以發現，隨著參數量的增加model不只在training set的表現越來越好，其generalization的能力也有提升(在testing set上有更好的結果)，甚至在training set的表現開始下降的初期，testing set仍有成長，而在一個峰值過後，loss與accuracy都會趨於穩定．由實驗可見參數量的增加對於generalization有一定的幫助．

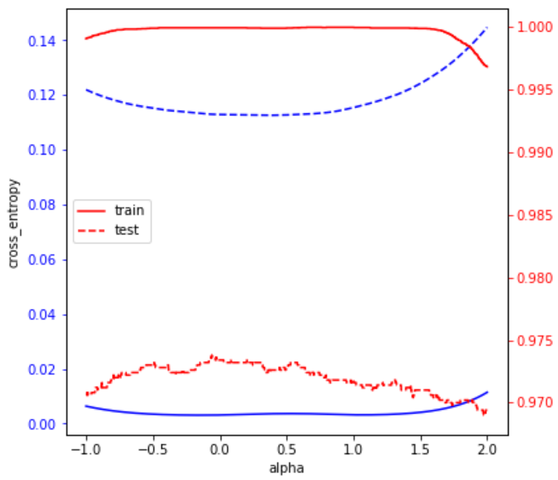
parameters	train_loss	test_loss	test_acc	train_acc
13546	0.0702	0.1741	0.9576	0.9782
28618	0.0398	0.1186	0.9697	0.9874
63370	0.0179	0.1107	0.9764	0.9945
151306	0.0139	0.1061	0.9781	0.9958
400906	0.0081	0.0915	0.9825	0.9975
1195018	0.0091	0.0888	0.983	0.9973
3962890	0.0163	0.0781	0.985	0.996
13596010	0.0149	0.0972	0.9844	0.9962
29394010	0.0149	0.1159	0.9839	0.9963
51192010	0.0154	0.1013	0.983	0.9962

Flatness v.s. Generalization

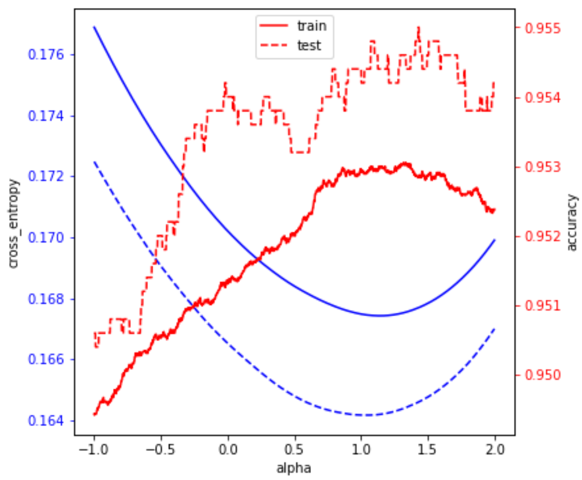
Part 1:

Describe the settings of the experiments (e.g. which task, what training approaches) (0.5%)
task：MNIST
training approaches：DNN模型，兩層64個units的隱藏層，對不同batch size做比較
實驗1. batch size 64 vs. batch size 1024
實驗2. batch size 64 vs. batch size 1024，兩邊epoch數都調小，不要完全收斂
Plot the figures of both training and testing, loss and accuracy to the number of interpolation ratio. (1%)

實驗1

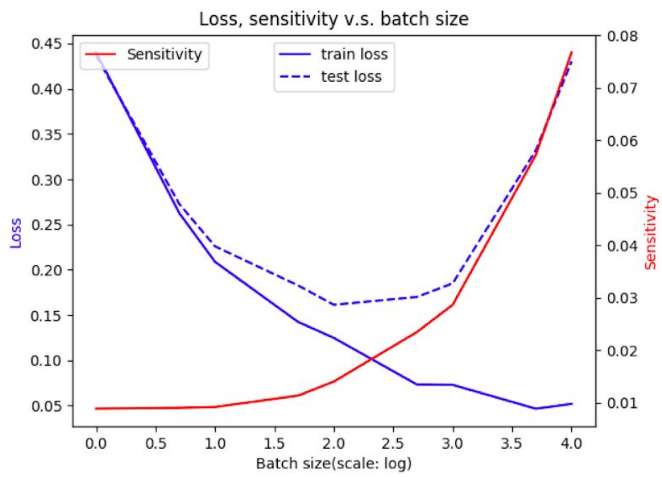


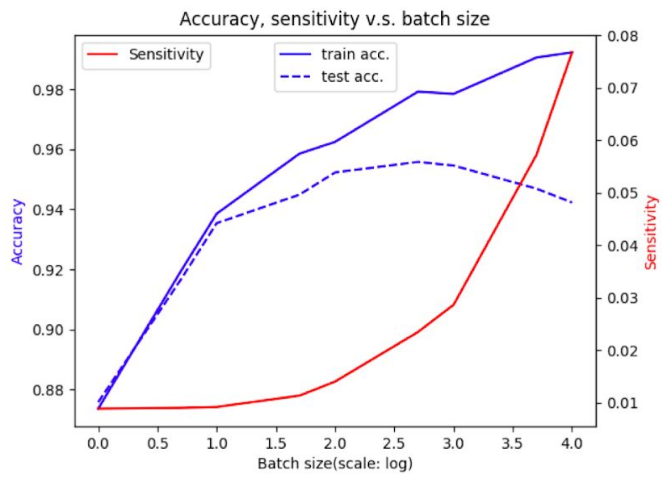
實驗2



Comment your result. (1%)
由實驗1的圖看出兩邊都收斂到一個平滑的平面上，實驗2則在還沒完全收斂時就先畫圖，雖然還沒收斂，但在test_acc的部分可以觀察到兩個寬度不太一樣的高峰，並發現batac size較大（圖中偏右）的模型高峰比較平坦

Part 2 :
Describe the settings of the experiments (e.g. which task, what training approaches) (0.5%)
MNIST資料集，不同approach為batch size為1, 5, 10, 50, 100, 500, 1000, 5000, 10000的DNN模型，各自訓練30000個steps。
Plot the figures of both training and testing, loss and accuracy, sensitivity to your chosen variable. (1%)





Comment your result. (1%)

從sensitivity角度，batch size越大sensitivity越高，在training，則size越大accuracy越高和loss越低，但testing來看，accuracy和loss在batch size為50左右的位置有最好的表現，50之後batch size越大performance越差，在某篇論文(Keskar et al. 2016)中提及，Large-Batch方法的探索性太差，容易在離起始點附近很近的地方停下來，更容易停在sharp minima(這是為什麼sensitivity越高的原因)，而在50之前，我認為主要是因為batch太小看到的sample不夠多，所以performance跟loss才會隨batch數上升(比如batch=1訓練起來相較於64收斂較不穩定)。

Bonus : Use other metrics or methods to evaluate a model's ability to generalize and concretely describe it and comment your results.
我們並沒有執行bonus的部分。

分工表

學號姓名	負責工作	比例(%)
R06946009 林庭宇	hw1-1-1 hw1-2-1 hw1-3-3-1	33.5
R06946006 李筑真	hw1-1-2 hw1-2-2 hw1-2-bouns hw1-3-3-2	33
R06946015 黃永翰	hw1-1-2 hw1-2-3 hw1-3-1 hw1-3-2	33.5