MLDS HW 1-1 Deep v.s. Shallow

A. Simulate a Function:

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

  Function 1: $\frac{\sin(5\pi x)}{5\pi x}$ / Function 2: $\mathbf{sgn}(\sin(5\pi x))$

  Each layer of these three models is **tanh** function.

  Model 1(#parameters: 987)

Layer (type)	Output Shape	Param #
dense_1 (Dense)	(None, 1)	2
dense_2 (Dense)	(None, 5)	10
dense_3 (Dense)	(None, 5)	30
dense_4 (Dense)	(None, 5)	30
dense_5 (Dense)	(None, 10)	60
dense_6 (Dense)	(None, 10)	110
dense_7 (Dense)	(None, 10)	110
dense_8 (Dense)	(None, 5)	55
dense_9 (Dense)	(None, 1)	6
dense_10 (Dense)	(None, 1)	2
dense_11 (Dense)	(None, 10)	20
dense_12 (Dense)	(None, 18)	198
dense_13 (Dense)	(None, 15)	285
dense_14 (Dense)	(None, 4)	64
dense_15 (Dense)	(None, 1)	5

Total params: 987
Trainable params: 987
Non-trainable params: 0

  Model 2 (#parameters: 986)

Layer (type)	Output Shape	Param #
dense_25 (Dense)	(None, 1)	2
dense_26 (Dense)	(None, 5)	10
dense_27 (Dense)	(None, 5)	30
dense_28 (Dense)	(None, 5)	30
dense_29 (Dense)	(None, 10)	60
dense_30 (Dense)	(None, 10)	110
dense_31 (Dense)	(None, 10)	110
dense_32 (Dense)	(None, 5)	55
dense_33 (Dense)	(None, 1)	6
dense_34 (Dense)	(None, 1)	2
dense_35 (Dense)	(None, 10)	20
dense_36 (Dense)	(None, 10)	110
dense_37 (Dense)	(None, 10)	110
dense_38 (Dense)	(None, 4)	44
dense_39 (Dense)	(None, 1)	5
dense_40 (Dense)	(None, 1)	2
dense_41 (Dense)	(None, 93)	186
dense_42 (Dense)	(None, 1)	94

Total params: 986
Trainable params: 986
Non-trainable params: 0

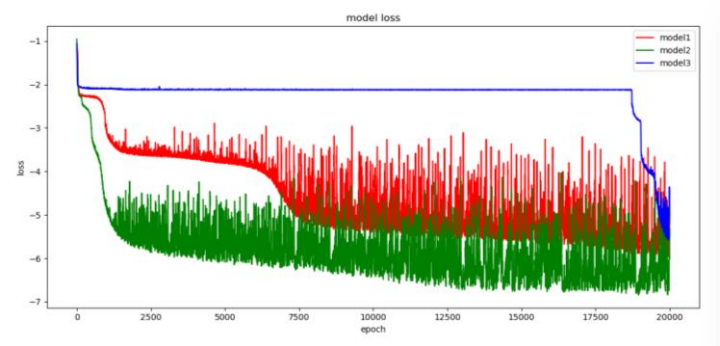
  Model 3 (#parameters:984)

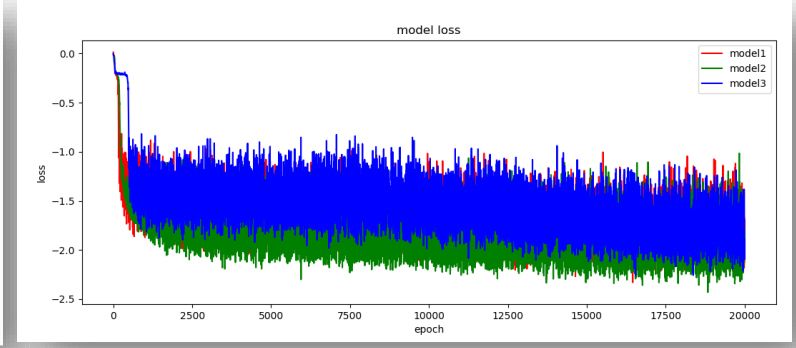
Layer (type)	Output Shape	Param #
dense_16 (Dense)	(None, 1)	2
dense_17 (Dense)	(None, 5)	10
dense_18 (Dense)	(None, 10)	60
dense_19 (Dense)	(None, 10)	110
dense_20 (Dense)	(None, 29)	319
dense_21 (Dense)	(None, 10)	300
dense_22 (Dense)	(None, 10)	110
dense_23 (Dense)	(None, 6)	66
dense_24 (Dense)	(None, 1)	7

Total params: 984
Trainable params: 984
Non-trainable params: 0

- In one chart, plot the training loss of all models.

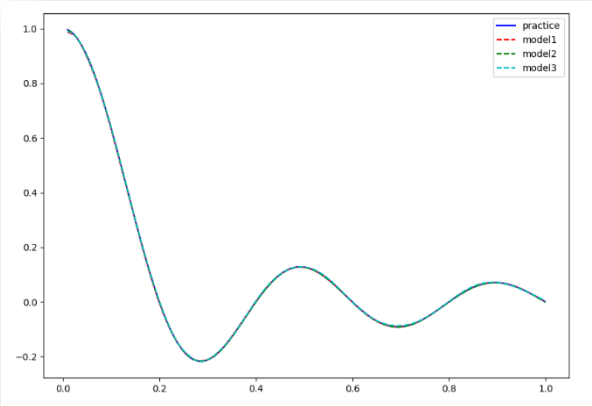
  Function 1: $\frac{\sin(5\pi x)}{5\pi x}$ Function 2: $\mathbf{sgn}(\sin(5\pi x))$

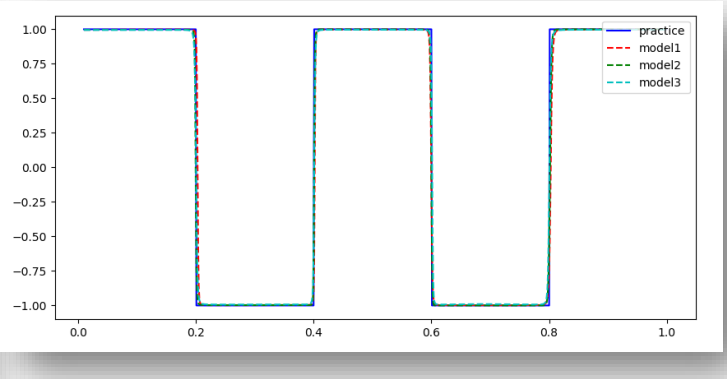




- In one graph, plot the predicted function curve of all models and the ground-truth function curve.

  Function 1: $\frac{\sin(5\pi x)}{5\pi x}$ Function 2: $\mathbf{sgn}(\sin(5\pi x))$





- Comment on your results.
  From the above experiment results, it can be observed that when the model gets deeper and more complicated, the loss of the model cannot always decrease faster than the shallow one. I supposed that one deep model may stuck in the saddle point (local minimum) if the initialization of the model is near around the original point, which may be the reason why Model 3 (the deepest one) has flatter loss curve in Function 1.

B. Train on Actual Tasks:

- Describe the models you use and the task you chose.
  Task 1: **MNIST**

Model 1(#parameters: 751933)

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 28, 28, 16)	160
max_pooling2d_5 (MaxPooling2	(None, 14, 14, 16)	0
conv2d_10 (Conv2D)	(None, 14, 14, 32)	4640
max_pooling2d_6 (MaxPooling2	(None, 7, 7, 32)	0
conv2d_11 (Conv2D)	(None, 7, 7, 64)	18496
max_pooling2d_7 (MaxPooling2	(None, 3, 3, 64)	0
conv2d_12 (Conv2D)	(None, 3, 3, 128)	73856
max_pooling2d_8 (MaxPooling2	(None, 1, 1, 128)	0
dropout_3 (Dropout)	(None, 1, 1, 128)	0
flatten_2 (Flatten)	(None, 128)	0
dense_7 (Dense)	(None, 1024)	132096
dense_8 (Dense)	(None, 505)	517625
dropout_4 (Dropout)	(None, 505)	0
dense_15 (Dense)	(None, 10)	5060

Total params: 751,933
Trainable params: 751,933
Non-trainable params: 0

Model 2 (#parameters: 751997)

Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 28, 28, 16)	160
conv2d_14 (Conv2D)	(None, 28, 28, 16)	2320
max_pooling2d_9 (MaxPooling2	(None, 14, 14, 16)	0
conv2d_15 (Conv2D)	(None, 14, 14, 32)	4640
conv2d_16 (Conv2D)	(None, 14, 14, 32)	9248
max_pooling2d_10 (MaxPooling	(None, 7, 7, 32)	0
conv2d_17 (Conv2D)	(None, 7, 7, 64)	18496
conv2d_18 (Conv2D)	(None, 7, 7, 64)	36928
max_pooling2d_11 (MaxPooling	(None, 3, 3, 64)	0
conv2d_19 (Conv2D)	(None, 3, 3, 128)	73856
conv2d_20 (Conv2D)	(None, 3, 3, 128)	147584
max_pooling2d_12 (MaxPooling	(None, 1, 1, 128)	0
dropout_5 (Dropout)	(None, 1, 1, 128)	0
flatten_3 (Flatten)	(None, 128)	0
dense_9 (Dense)	(None, 1024)	132096
dense_10 (Dense)	(None, 275)	281875
dense_11 (Dense)	(None, 128)	35328
dense_12 (Dense)	(None, 64)	8256
dense_13 (Dense)	(None, 16)	1040
dropout_6 (Dropout)	(None, 16)	0
dense_16 (Dense)	(None, 10)	170

Total params: 751,997
Trainable params: 751,997
Non-trainable params: 0

Model 3 (#parameters: 751926)

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 28, 28, 4)	40
conv2d_2 (Conv2D)	(None, 28, 28, 4)	148
max_pooling2d_1 (MaxPooling2	(None, 14, 14, 4)	0
conv2d_3 (Conv2D)	(None, 14, 14, 8)	296
conv2d_4 (Conv2D)	(None, 14, 14, 8)	584
max_pooling2d_2 (MaxPooling2	(None, 7, 7, 8)	0
conv2d_5 (Conv2D)	(None, 7, 7, 16)	1168
conv2d_6 (Conv2D)	(None, 7, 7, 16)	2320
max_pooling2d_3 (MaxPooling2	(None, 3, 3, 16)	0
conv2d_7 (Conv2D)	(None, 3, 3, 32)	4640
conv2d_8 (Conv2D)	(None, 3, 3, 32)	9248
max_pooling2d_4 (MaxPooling2	(None, 1, 1, 32)	0
dropout_1 (Dropout)	(None, 1, 1, 32)	0
flatten_1 (Flatten)	(None, 32)	0
dense_1 (Dense)	(None, 1024)	33792
dense_2 (Dense)	(None, 512)	524800
dense_3 (Dense)	(None, 256)	131328
dense_4 (Dense)	(None, 128)	32896
dense_5 (Dense)	(None, 64)	8256
dense_6 (Dense)	(None, 32)	2080
dropout_2 (Dropout)	(None, 32)	0
dense_14 (Dense)	(None, 10)	330

Total params: 751,926
Trainable params: 751,926
Non-trainable params: 0

Task 2: **CIFAR-10**

Model 1(#parameters:830012)

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 32, 32, 16)	448
conv2d_10 (Conv2D)	(None, 32, 32, 16)	2320
max_pooling2d_5 (MaxPooling2	(None, 16, 16, 16)	0
conv2d_11 (Conv2D)	(None, 16, 16, 32)	4640
max_pooling2d_6 (MaxPooling2	(None, 8, 8, 32)	0
conv2d_12 (Conv2D)	(None, 8, 8, 64)	18496
max_pooling2d_7 (MaxPooling2	(None, 4, 4, 64)	0
conv2d_13 (Conv2D)	(None, 4, 4, 128)	73856
max_pooling2d_8 (MaxPooling2	(None, 2, 2, 128)	0
flatten_2 (Flatten)	(None, 512)	0
dropout_7 (Dropout)	(None, 512)	0
dense_3 (Dense)	(None, 1024)	525312
dense_4 (Dense)	(None, 198)	202950
dropout_8 (Dropout)	(None, 198)	0
dense_5 (Dense)	(None, 10)	1990

Total params: 830,012
Trainable params: 830,012
Non-trainable params: 0

Model 2 (#parameters: 829082)

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 32, 32, 16)	448
dropout_1 (Dropout)	(None, 32, 32, 16)	0
conv2d_2 (Conv2D)	(None, 32, 32, 16)	2320
max_pooling2d_1 (MaxPooling2	(None, 16, 16, 16)	0
conv2d_3 (Conv2D)	(None, 16, 16, 32)	4640
dropout_2 (Dropout)	(None, 16, 16, 32)	0
conv2d_4 (Conv2D)	(None, 16, 16, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 8, 8, 32)	0
conv2d_5 (Conv2D)	(None, 8, 8, 64)	18496
dropout_3 (Dropout)	(None, 8, 8, 64)	0
conv2d_6 (Conv2D)	(None, 8, 8, 64)	36928
max_pooling2d_3 (MaxPooling2	(None, 4, 4, 64)	0
conv2d_7 (Conv2D)	(None, 4, 4, 128)	73856
dropout_4 (Dropout)	(None, 4, 4, 128)	0
conv2d_8 (Conv2D)	(None, 4, 4, 128)	147584
max_pooling2d_4 (MaxPooling2	(None, 2, 2, 128)	0
flatten_1 (Flatten)	(None, 512)	0
dropout_5 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 1024)	525312
dropout_6 (Dropout)	(None, 1024)	0
dense_2 (Dense)	(None, 10)	10250

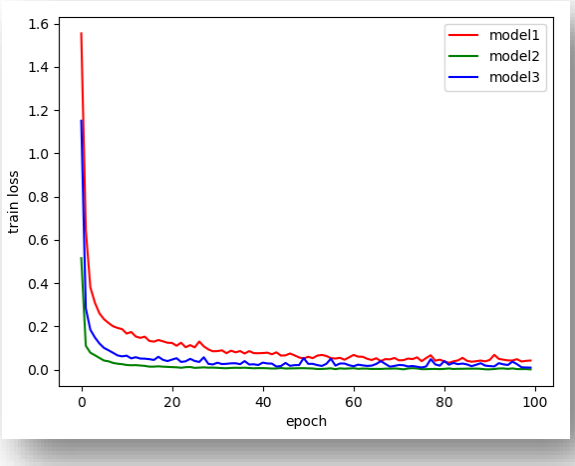
Total params: 829,082
Trainable params: 829,082
Non-trainable params: 0

Model 3 (#parameters: 829430)

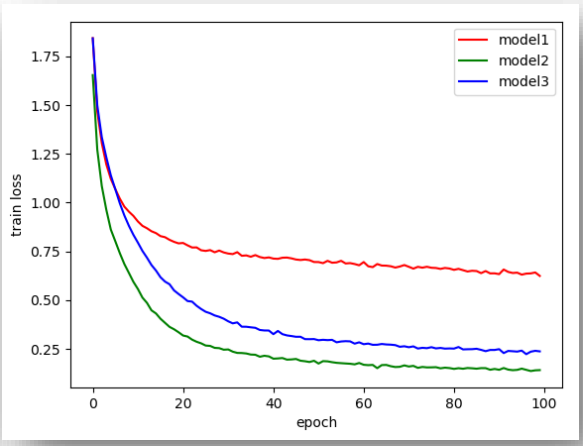
Layer (type)	Output Shape	Param #
conv2d_14 (Conv2D)	(None, 32, 32, 8)	224
max_pooling2d_9 (MaxPooling2	(None, 16, 16, 8)	0
conv2d_15 (Conv2D)	(None, 16, 16, 16)	1168
max_pooling2d_10 (MaxPooling	(None, 8, 8, 16)	0
conv2d_16 (Conv2D)	(None, 8, 8, 32)	4640
max_pooling2d_11 (MaxPooling	(None, 4, 4, 32)	0
flatten_3 (Flatten)	(None, 512)	0
dropout_9 (Dropout)	(None, 512)	0
dense_6 (Dense)	(None, 1024)	525312
dense_7 (Dense)	(None, 256)	262400
dense_8 (Dense)	(None, 128)	32896
dense_9 (Dense)	(None, 20)	2580
dropout_10 (Dropout)	(None, 20)	0
dense_10 (Dense)	(None, 10)	210

Total params: 829,430
Trainable params: 829,430
Non-trainable params: 0

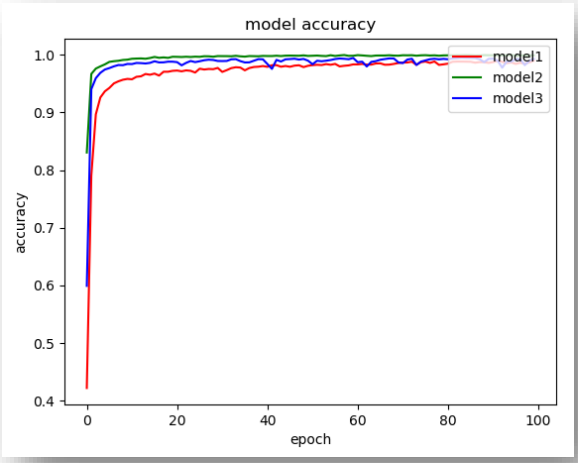
- In one chart, plot the training loss of all models.
  Task 1: **MNIST**



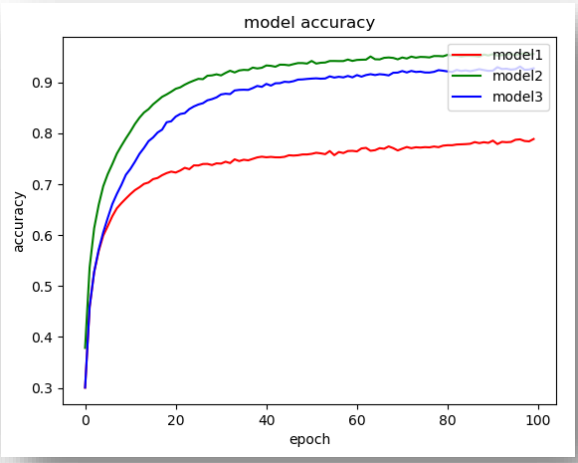
  Task 2: **CIFAR-10**



- In one chart, plot the training accuracy.
  Task 1: **MNIST**



  Task 2: **CIFAR-10**

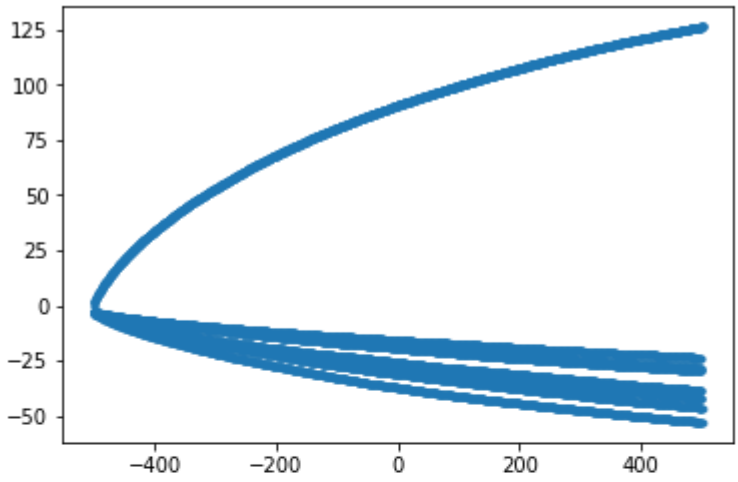


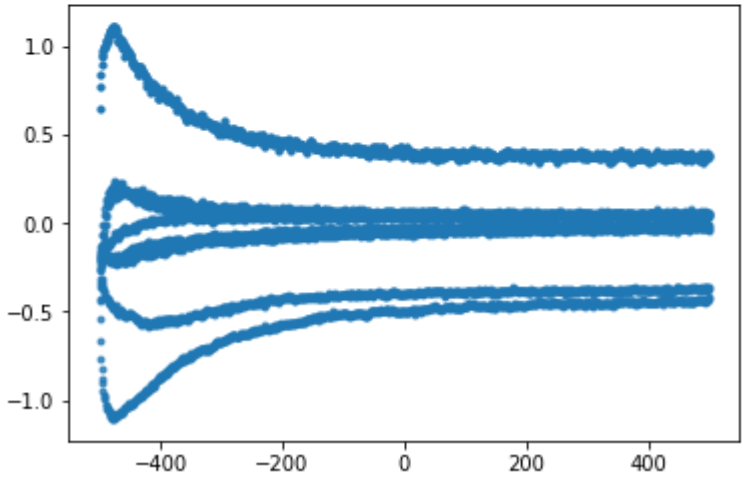
- Comment on your results.
  From the above experiment results, it can be observed that when the model gets deeper and more complicated, the loss value decreases faster and the accuracy value increases faster than the shallower ones. Since the task of MNIST is simpler, the final performance of the there models are similar; however, it can be observed that in the task of CIFAR-10, the final performances of the three models are different among one another.

MLDS HW 1-2 Optimization

C. Visualize the optimization process

- Describe your experiment settings. (The cycle you record the model parameters, optimizer, dimension reduction method, etc)
  Cycle we record the model parameters: 3
  Optimizer: Adam
  Dimension reduction method: PCA
- Train the model for 8 times, selecting the parameters of any one layer and whole model and plot them on the figures separately.

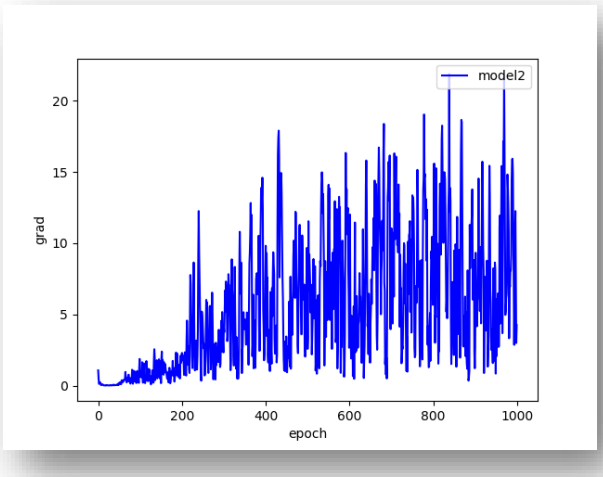


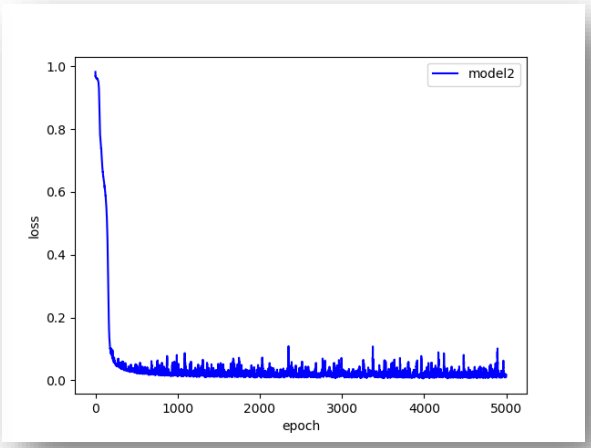


- Comment on your result.
  It can be observed from the above graph that each training process will led to different optimization solution. But the final performances are similar(around 98%~99%).

D. Observe gradient norm during training.

- Plot one figure which contain gradient norm to iterations and the loss to iterations.





- Comment your result.
  From the above result, it can be observed that the gradient norm changes slowly at the beginning, but shows substantial changing when the epoch is up to 200. I think it may because task MNIST has a flatten plan near the original point in the loss surface.

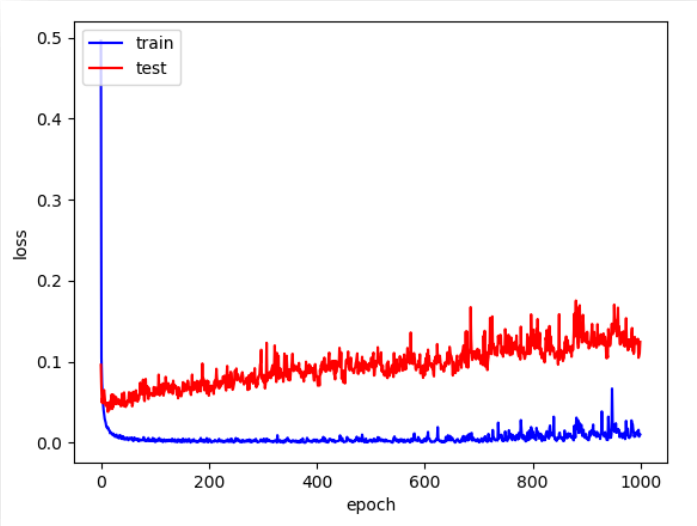
E. What happens when gradient is almost zero?

- State how you get the weight which gradient norm is zero and how you define the minimal ratio. (2%)
- Train the model for 100 times. Plot the figure of minimal ratio to the loss. (2%)
- Comment your result. (1%)

MLDS HW 1-3 Generalization

F. Can network fit random variables?

- Describe your settings of the experiments. (e.g. which task, learning rate, optimizer)
  Task: MNIST
  Learning rate = 0.005
  Optimizer: Adam
- Plot the figure of the relationship between training and testing, loss and epochs.



G. Number of parameters v.s. Generalization

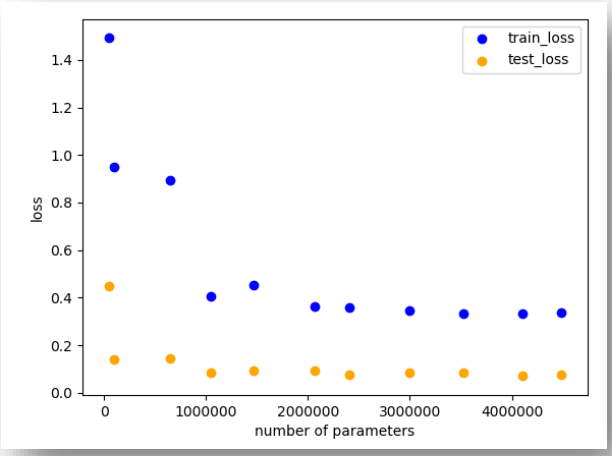
- Describe your settings of the experiments. (e.g. which task, the 10 or more structures you choose)
  Task: MNIST
  Structure: The basic model is listed on the below:

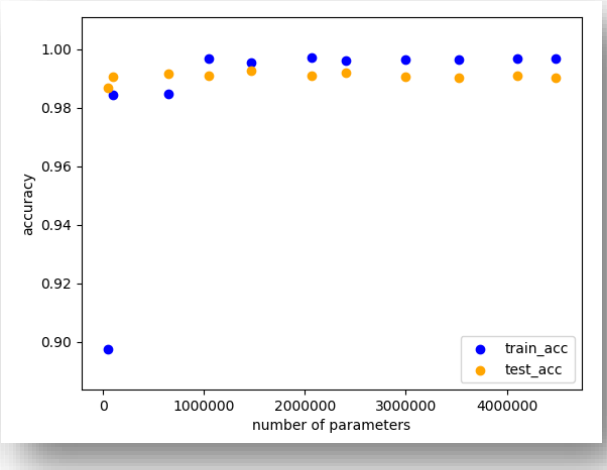
Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 28, 28, 16)	160
max_pooling2d_5 (MaxPooling2	(None, 14, 14, 16)	0
conv2d_10 (Conv2D)	(None, 14, 14, 32)	4640
max_pooling2d_6 (MaxPooling2	(None, 7, 7, 32)	0
conv2d_11 (Conv2D)	(None, 7, 7, 64)	18496
max_pooling2d_7 (MaxPooling2	(None, 3, 3, 64)	0
conv2d_12 (Conv2D)	(None, 3, 3, 128)	73856
max_pooling2d_8 (MaxPooling2	(None, 1, 1, 128)	0
dropout_3 (Dropout)	(None, 1, 1, 128)	0
flatten_2 (Flatten)	(None, 128)	0
dense_7 (Dense)	(None, 1024)	132096
dense_8 (Dense)	(None, 505)	517625
dropout_4 (Dropout)	(None, 505)	0
dense_15 (Dense)	(None, 10)	5060

Total params: 751,933
Trainable params: 751,933
Non-trainable params: 0

Setting: Change the basic model to the ones with different amounts of parameters

- Plot the figures of both training and testing, loss and accuracy to the number of parameters.

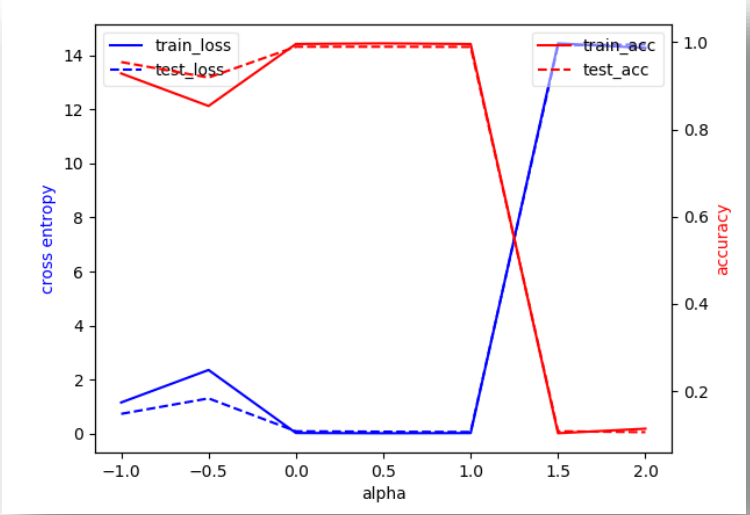




- Comment your result.
  From the above results, it can be observed that when the number of parameters increases, the loss value indeed decreases the same as our expected thoughts; however, the accuracy value seems to be less different when the number of parameters are more than around 750000.

H. Flatness v.s. Generalization

- Part 1:
  - Describe the settings of the experiments (e.g. which task, what training approaches)
    Task: MNIST
    Training approaches: batch size 64 v.s. batch size 1024
  - Plot the figures of both training and testing, loss and accuracy to the number of interpolation ratio.



  - Comment your result.
    It can be observed from the above result that when the alpha value increases, which has indicated that the batch size is larger, the accuracy value is decreasing and the loss value is increasing, and vise versa.
- Part 2 :
  - Describe the settings of the experiments (e.g. which task, what training approaches) (0.5%)
  - Plot the figures of both training and testing, loss and accuracy, sensitivity to your chosen variable. (1%)
  - Comment your result. (1%)

分工:

r06944046	r06922141	b03901156
33.3%	33.3%	33.3%