CpSc 8430: Deep Learning

Homework 1

Copy Right Notice

- Most slides in this presentation are adopted from slides of text book and various sources. The Copyright belong to the original authors. Thanks!

Three Parts in HW1

- **(1-1) Deep vs Shallow:**
  - Simulate a function.
  - Train on actual task using shallow and deep models.
- **(1-2) Optimization**
- **(1-3) Generalization**

HW1-1: Deep vs Shallow

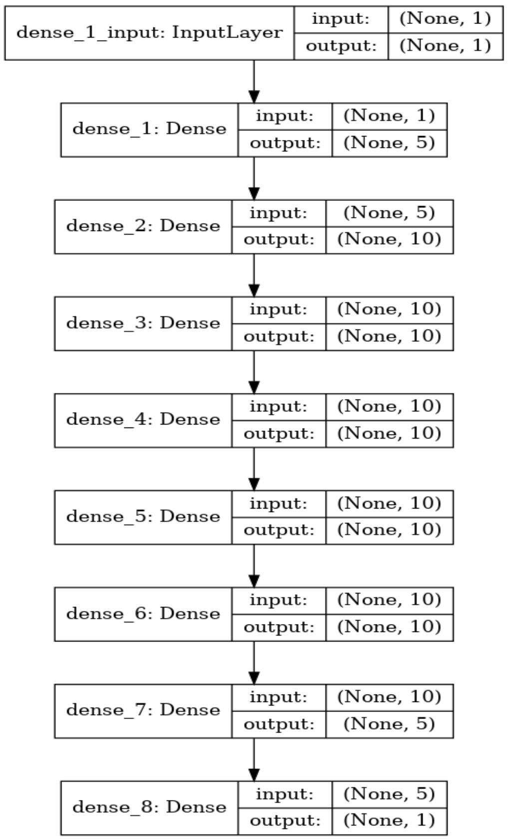
- **Simulate a function:**
  - function need to be single-input, single-output
  - function need to be non-linear
- **Train on actual task:**
  - MNIST or CIFAR-10

HW1-1 Simulate a Function

- **Requirements:**
  - train at least two different DNN models with the same amount of parameters until convergence
  - compare the training process of models by showing the loss in each epoch in a chart
  - visualize the ground-truth and predictions by models in a graph
- **Tips:**
  - constrain the input domain
  - hyper-parameters are important (i.e. tune all models to the best)

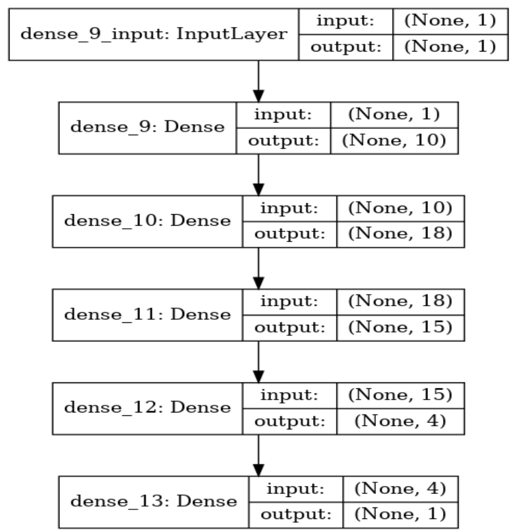
HW1-1 Simulate a Function

Example models:



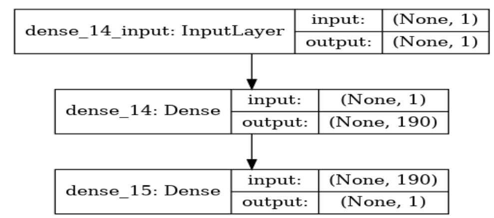
model0

parameters : 571



model1

parameters : 572



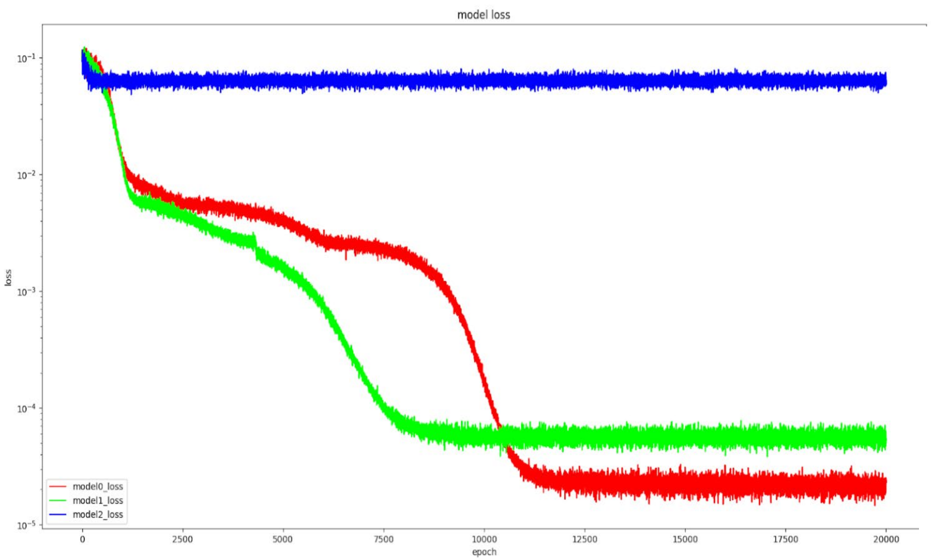
model2

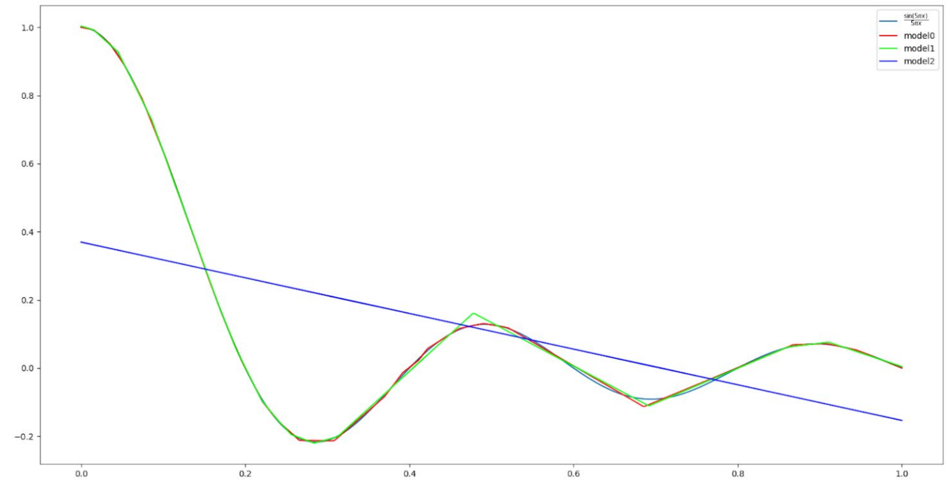
parameters : 571

HW1-1 Simulate a Function

$$\frac{\sin(5\pi x)}{5\pi x}$$

20000 epochs

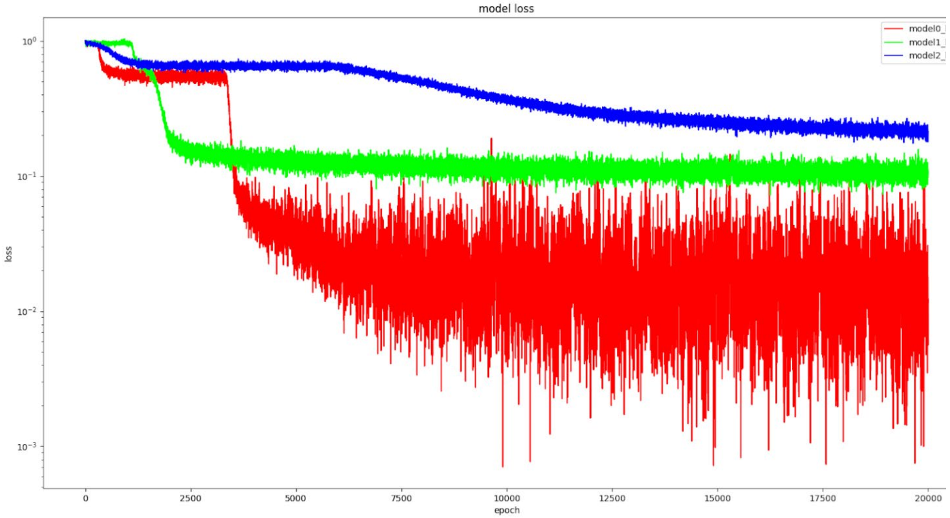


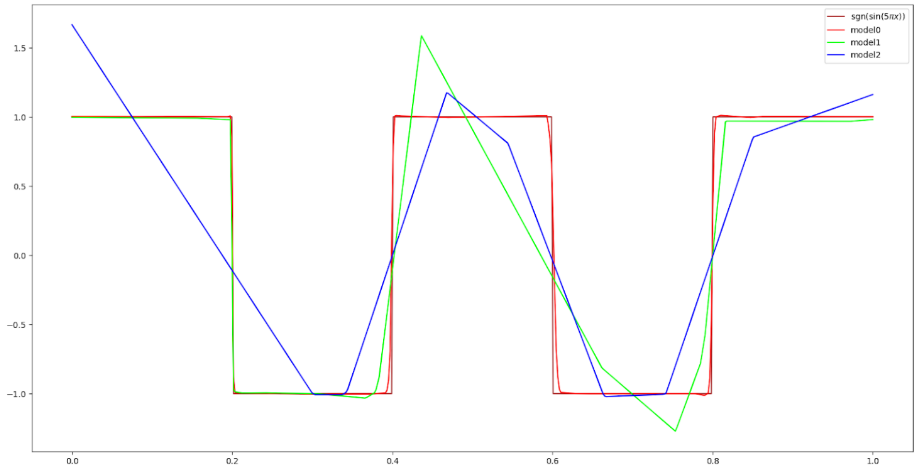


HW1-1 Simulate a Function

$$\operatorname{sgn}(\sin(5\pi x))$$

20000 epochs



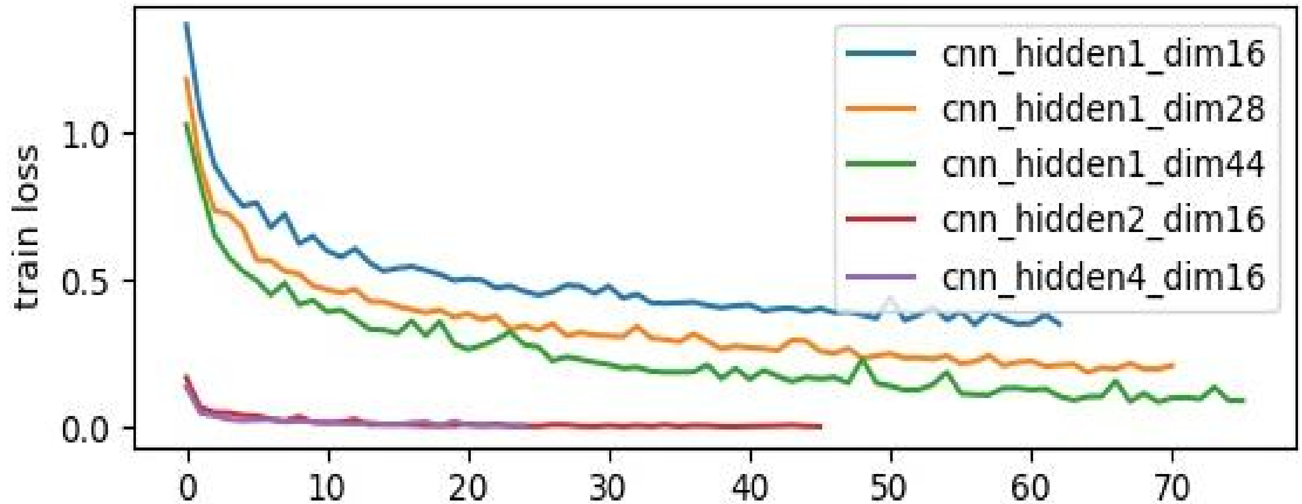


HW1-1 Train on Actual Tasks

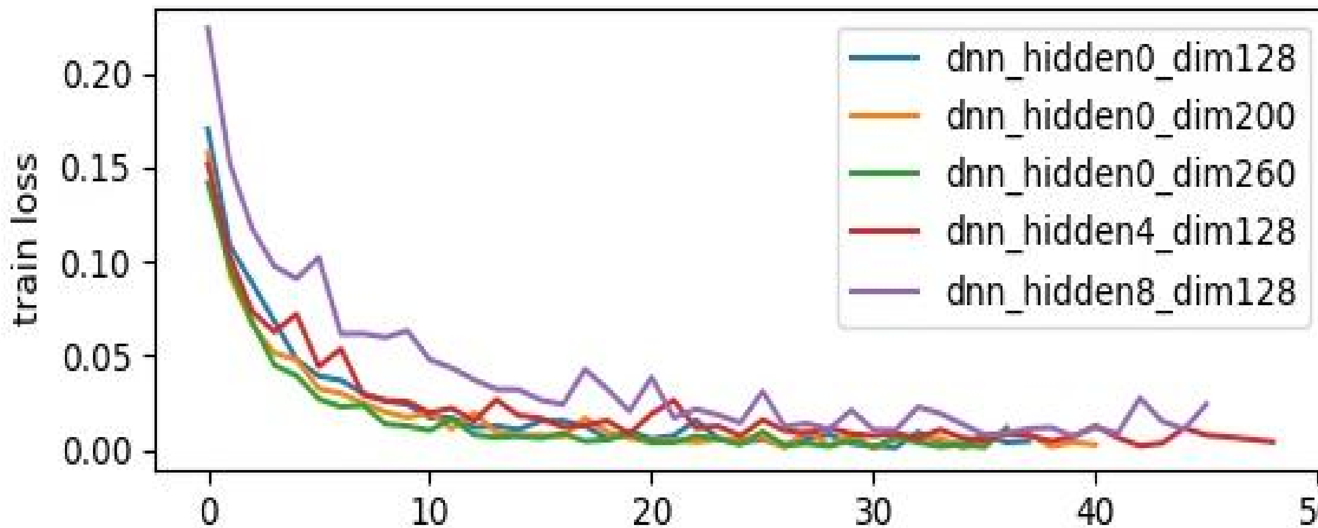
- **Requirements:**
  - use MNIST or CIFAR-10
  - use CNN or DNN
  - visualize the training process by showing both loss and accuracy on two charts
- **Tips:**
  - CNN is easier to see the difference

HW1-1 Train on Actual Tasks

- MNIST : CNN

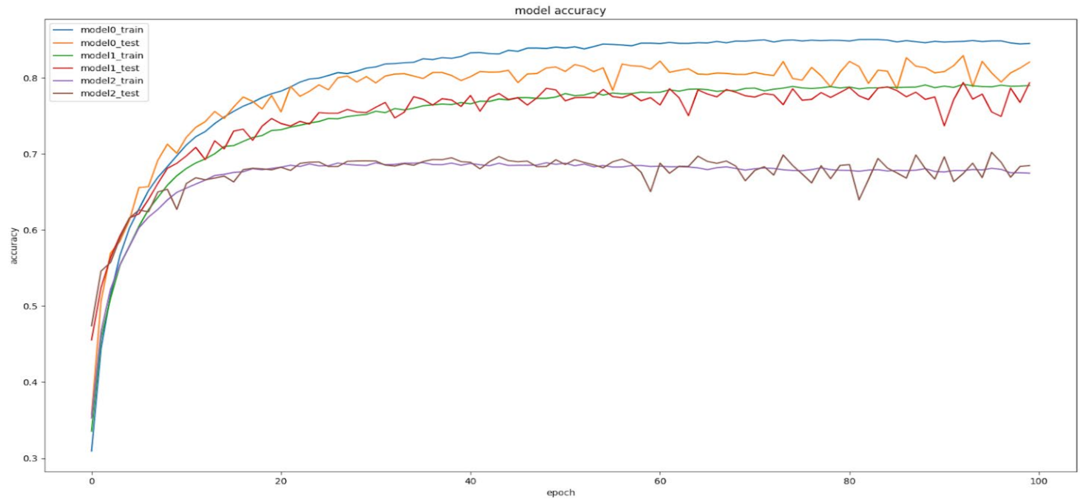


- MNIST : DNN

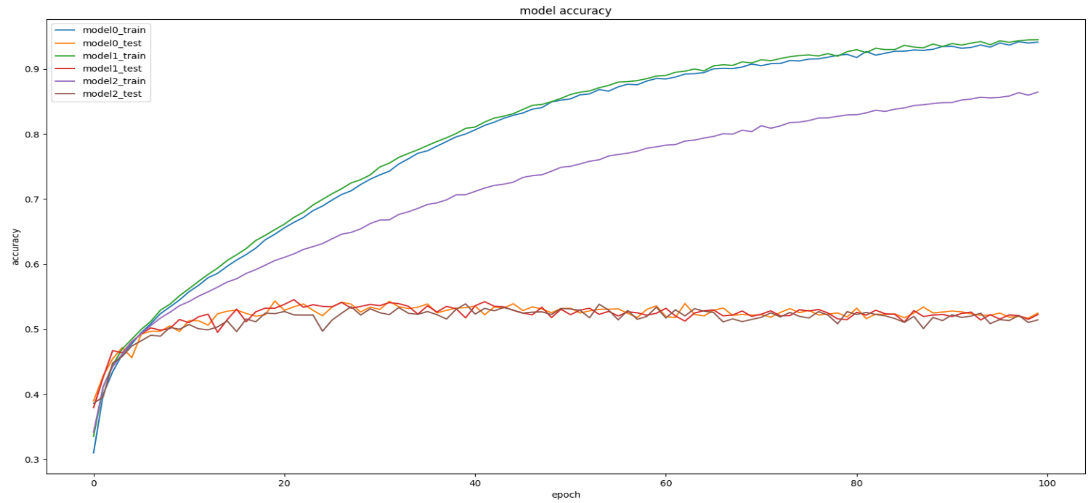


HW1-1 Train on Actual Tasks

- CIFAR-10 : CNN



- CIFAR-10 : DNN



HW1-1 Report Questions

- **Simulate a Function:**
  - Describe the models you use, including the number of parameters (at least two models) and the function you use.
  - In one chart, plot the training loss of all models.
  - In one graph, plot the predicted function curve of all models and the ground-truth function curve.
  - Comment on your results.
  - Use more than two models in all previous questions. (bonus)
  - Use more than one function. (bonus)
- **Train on Actual Tasks:**
  - Describe the models you use and the task you chose.
  - In one chart, plot the training loss of all models.
  - In one chart, plot the training accuracy.
  - Comment on your results.
  - Use more than two models in all previous questions. (bonus)
  - Train on more than one task. (bonus)

HW1-2: Optimization

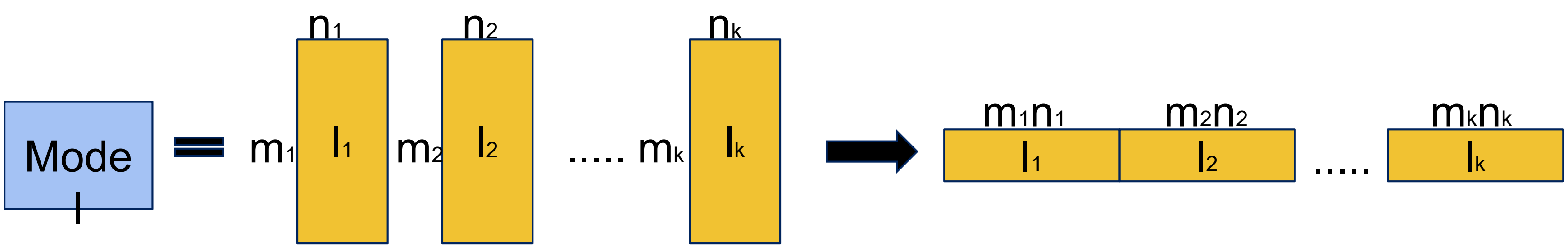
- **Three subtask**
  - Visualize the optimization process.
  - Observe gradient norm during training.
  - What happens when gradient is almost zero?
- **Train on designed function, MNIST or CIFAR-10**

Visualize the Optimization Process

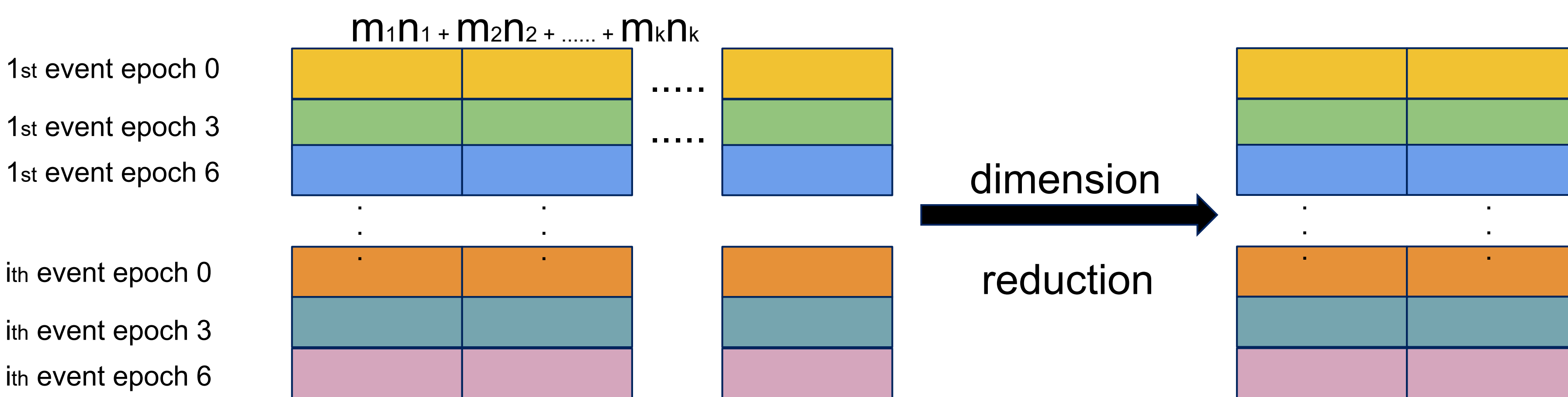
- **Requirement**
  - Collect weights of the model every n epochs.
  - Also collect the weights of the model of different training events.
  - Record the accuracy (loss) corresponding to the collected parameters.
  - Plot the above results on a figure.

Visualize the Optimization Process

- Collect parameters of the model:

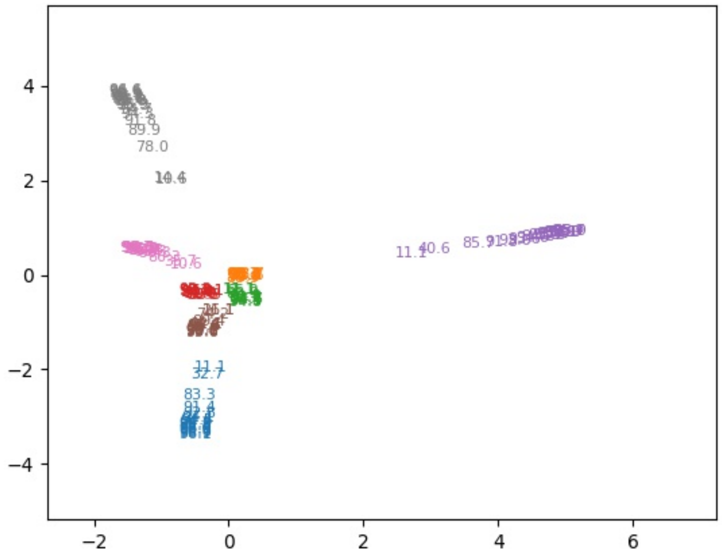


- Reduce the dimension

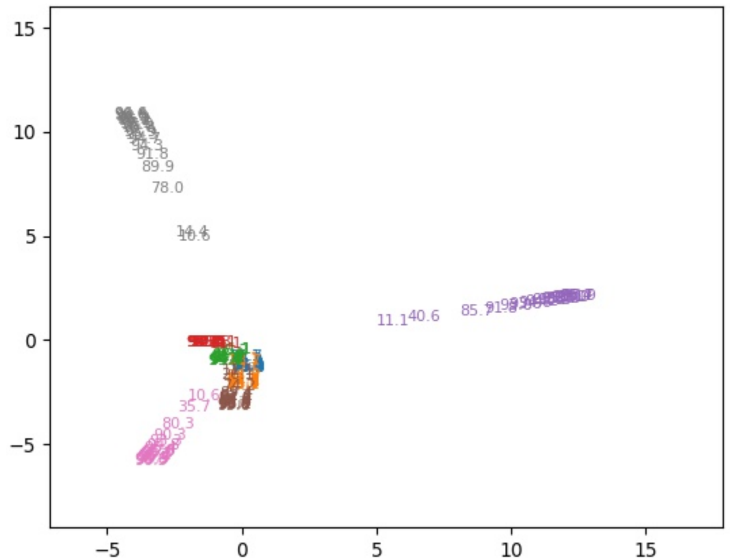


Visualize the Optimization Process

- **DNN train on MNIST**
- **Collect the weights every 3 epochs, and train 8 times. Reduce the dimension of weights to 2 by PCA.**



layer 1



whole model

Observe Gradient Norm During Training

Requirement

- Record the gradient norm and the loss during training.
- Plot them on one figure.

p-norm

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}$$

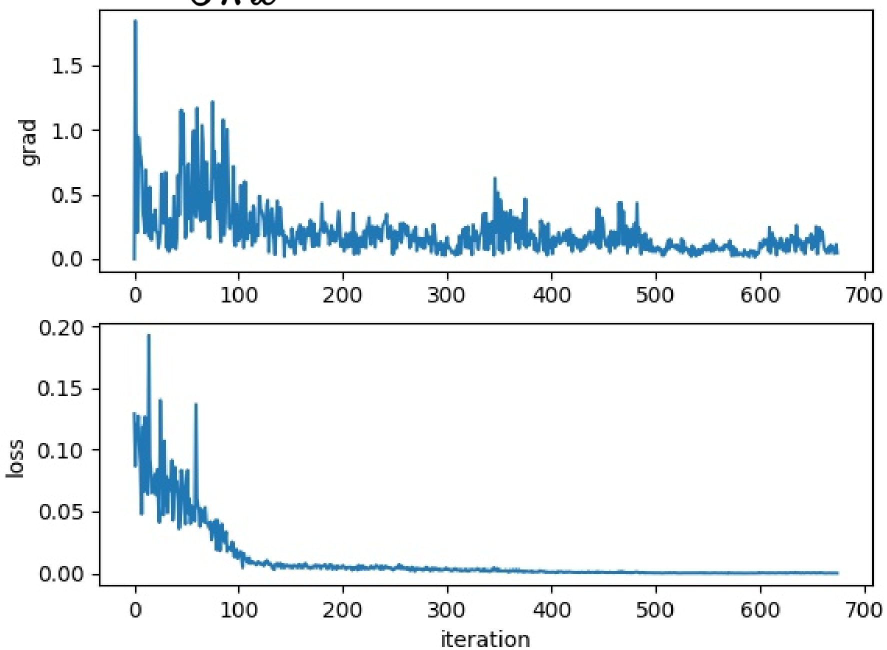
- In PyTorch:

```
grad_all = 0.0

for p in model.parameters():
    grad = 0.0
    if p.grad is not None:
        grad = (p.grad.cpu().data.numpy() ** 2).sum()
    grad_all += grad

grad_norm = grad_all ** 0.5
```

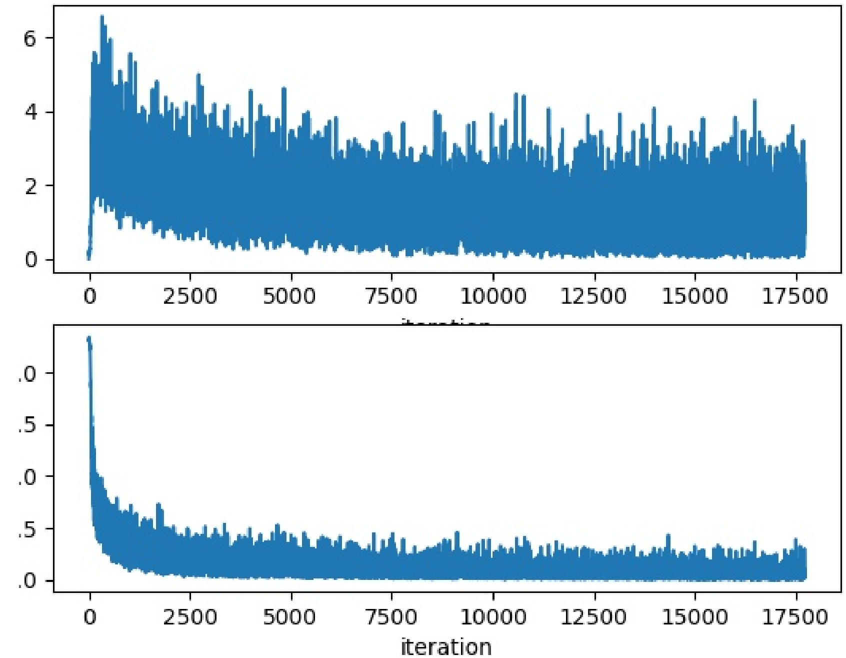
- Other packages: The similar code can be applied.

Observe Gradient Norm During Training

$$\frac{\sin 5\pi x}{5\pi x}$$



MNIST



What Happened When Gradient is Almost Zero

- **Requirement**
  - Try to find the weights of the model when the gradient norm is zero (as small as possible).
  - Compute the "minimal ratio" of the weights: how likely the weights to be a minima.
  - Plot the figure between minimal ratio and the loss when the gradient is almost zero.
- **Tips**
  - Train on a small network.

What Happened When Gradient is Almost Zero

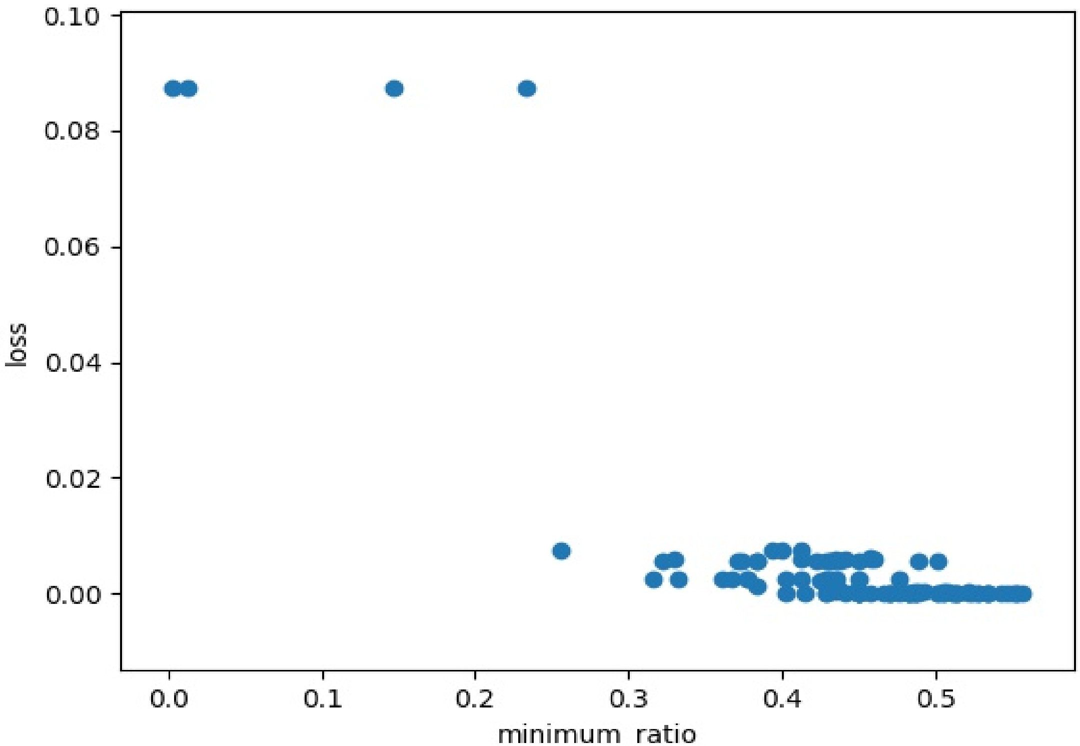
- **How to reach the point where the gradient norm is zero?**
  - First, train the network with original loss function.
    - Change the objective function to gradient norm and keep training.
    - Or use second order optimization method, such as Newton's method or Levenberg-Marquardt algorithm (more stable)

- **How to compute minimal ratio?**
  - Compute () $H(L(\theta_{norm=0}))$ (hessian matrix), and then find its eigenvalues. The proportion of the eigenvalues which are greater than zero is the minimal ratio.
  - Sample lots of weights around $\theta_{norm=0}$, and compute $L(\theta_{sample})$. The minimal ratio is the proportion that $L(\theta_{sample}) > L(\theta_{norm=0})$.

What Happened When Gradient is Almost Zero

$$\frac{\sin 5\pi x}{5\pi x}$$

- Train 100 times.
  - Find gradient norm equal to zero by change objective function.
  - Minimal ratio is defined as the proportion of eigenvalues greater than zero.



HW1-2 Report Questions

- **Visualize the optimization process.**
  - Describe your experiment settings. (The cycle you record the model parameters, optimizer, dimension reduction method, etc)
  - 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.
- **Observe gradient norm during training.**
  - Plot one figure which contain gradient norm to iterations and the loss to iterations.
  - Comment your result.
- **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.
  - Train the model for 100 times. Plot the figure of minimal ratio to the loss.
  - Comment your result.
- **Bonus**
  - Use any method to visualize the error surface.
  - Concretely describe your method and comment your result.

HW1-3: Generalization

- **Three subtask**
  - Can network fit random labels?
  - Number of parameters v.s. Generalization
  - Flatness v.s. Generalization
- **Train on MNIST or CIFAR-10**

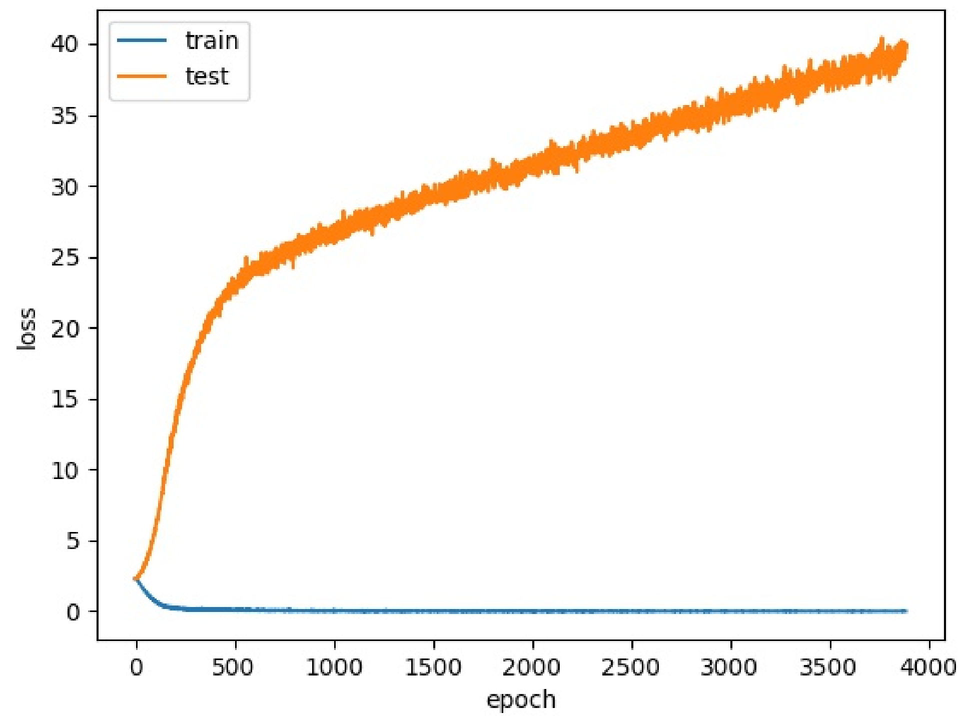
Can network fit random labels?

- **Requirement**
  - Train on MNIST or CIFAR-10
  - Randomly shuffle the label before training.
  - Try to fit the network with these random labels.

Can network fit random labels?

```
[0, 7, 0, 1, 3, 5, 9, 5]
[3, 3, 1, 8, 4, 9, 2, 2]
[8, 9, 2, 9, 7, 6, 4, 8]
[4, 5, 6, 7, 1, 0, 5, 5]
[7, 5, 7, 8, 9, 7, 0, 5]
[3, 0, 6, 8, 0, 1, 8, 9]
[5, 5, 6, 6, 1, 0, 1, 3]
[5, 8, 8, 6, 6, 5, 1, 0]
```

- MNIST
  - 3 hidden layers with 256 nodes.

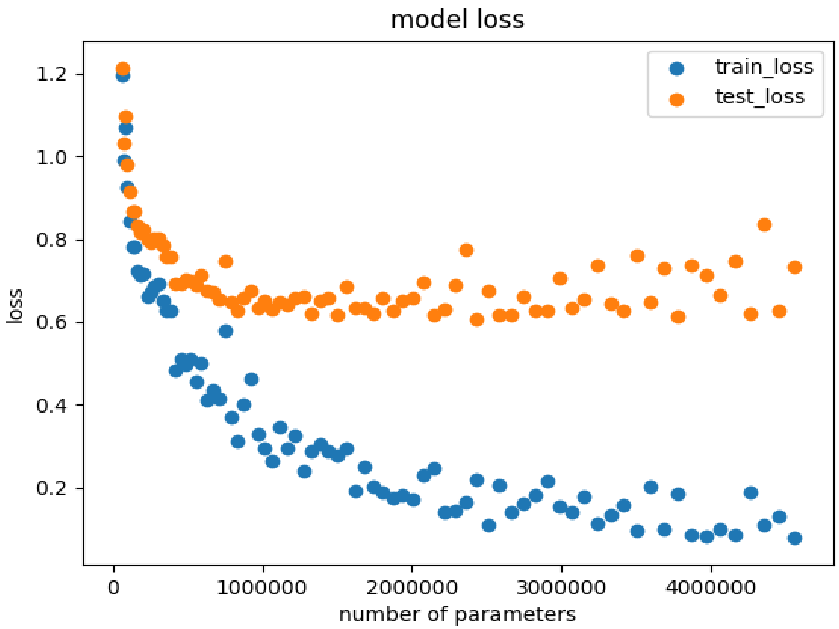


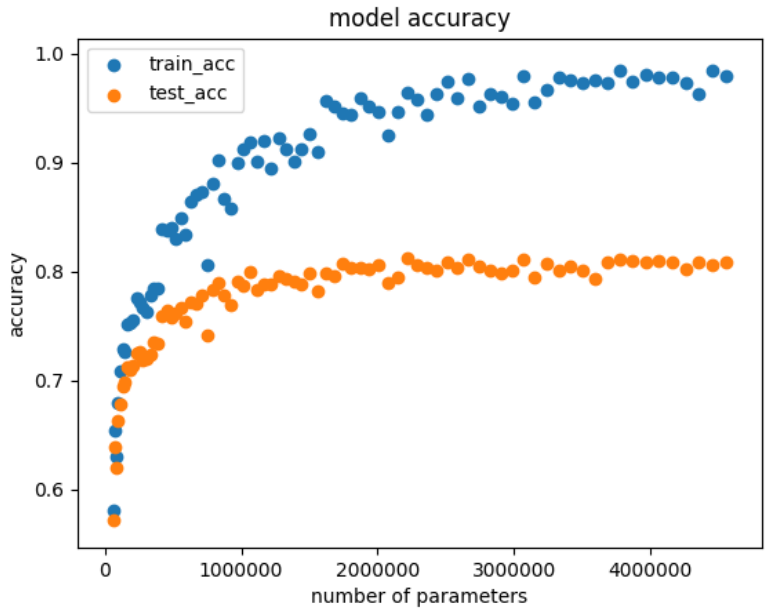
Number of parameters v.s. Generalization

- **Requirement**
  - Train on MNIST or CIFAR-10
  - At least 10 similar-structured models with different amount of parameters
  - Record both training and testing, loss and accuracy

Number of parameters v.s. Generalization

- **CIFAR-10**



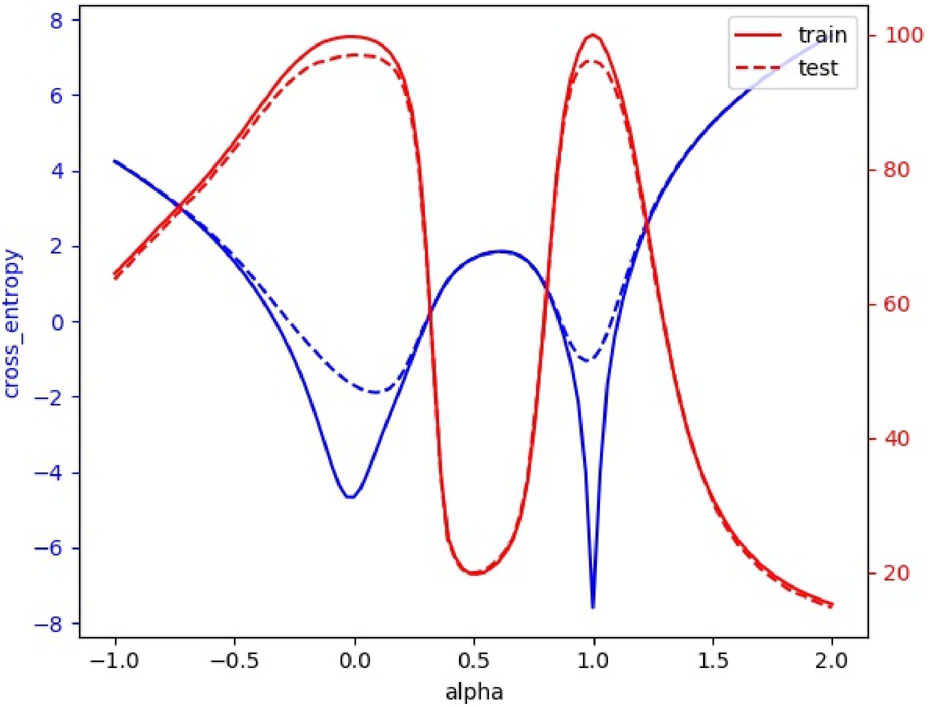


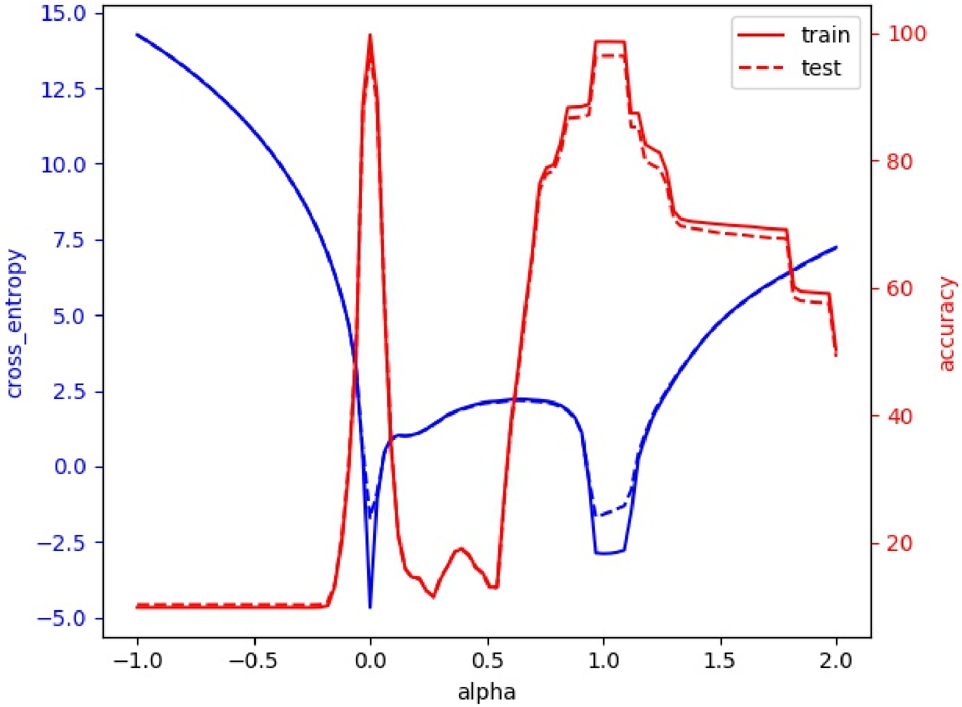
Flatness v.s. Generalization - part1

- **Visualize the line between two trained models**
- **Requirement:**
  - Train two models (m1 and m2) with different training approach. (e.g. batch size 64 and 1024)
  - Record the loss and accuracy of the model which is linear interpolation between m1 and m2.
  - $\boldsymbol{\theta}_{\boldsymbol{\alpha}} = (1 - \alpha)\boldsymbol{\theta}_{\mathbf{1}} + \alpha\boldsymbol{\theta}_{\mathbf{2}}$, where α is the interpolation ratio, $\boldsymbol{\theta}$ is the parameter of the model.

Flatness v.s. Generalization - part1

- **MNIST (The cross_entropy is log scale)**
  - batch size 64 vs. batch size 1024
  - learning rate 1e-3 vs. 1e-2





Flatness v.s. Generalization - part2

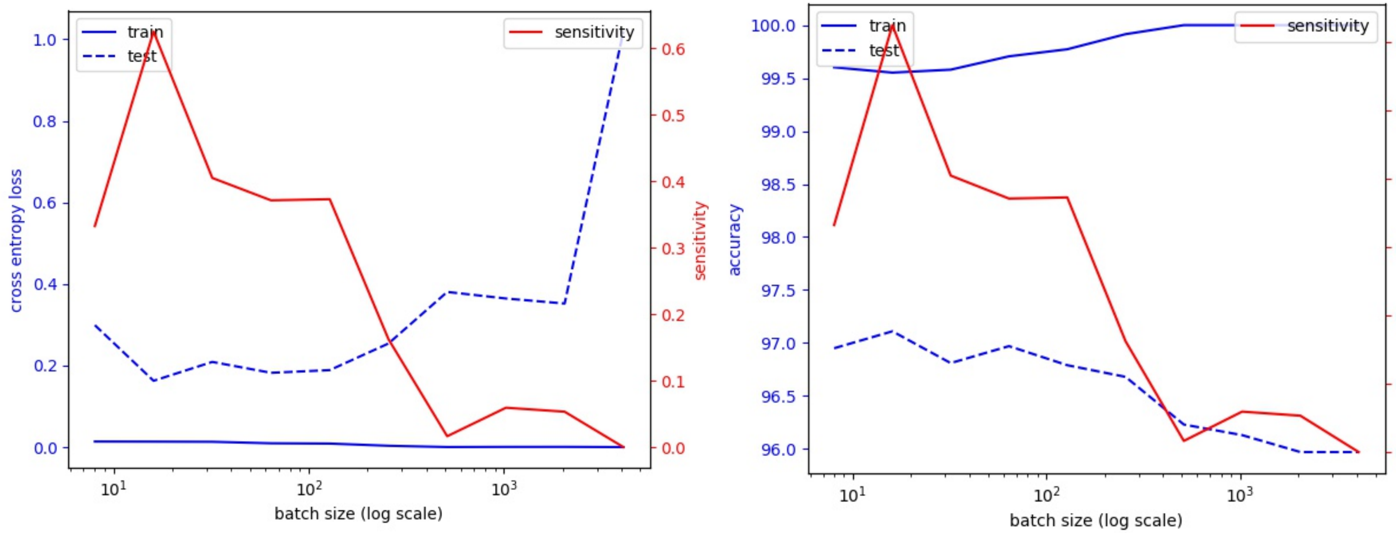
- **Requirement:**
  - Train at least 5 models with different training approach.
  - Record the loss and accuracy of all models.
  - Record the sensitivity of all models.

Flatness v.s. Generalization - part2

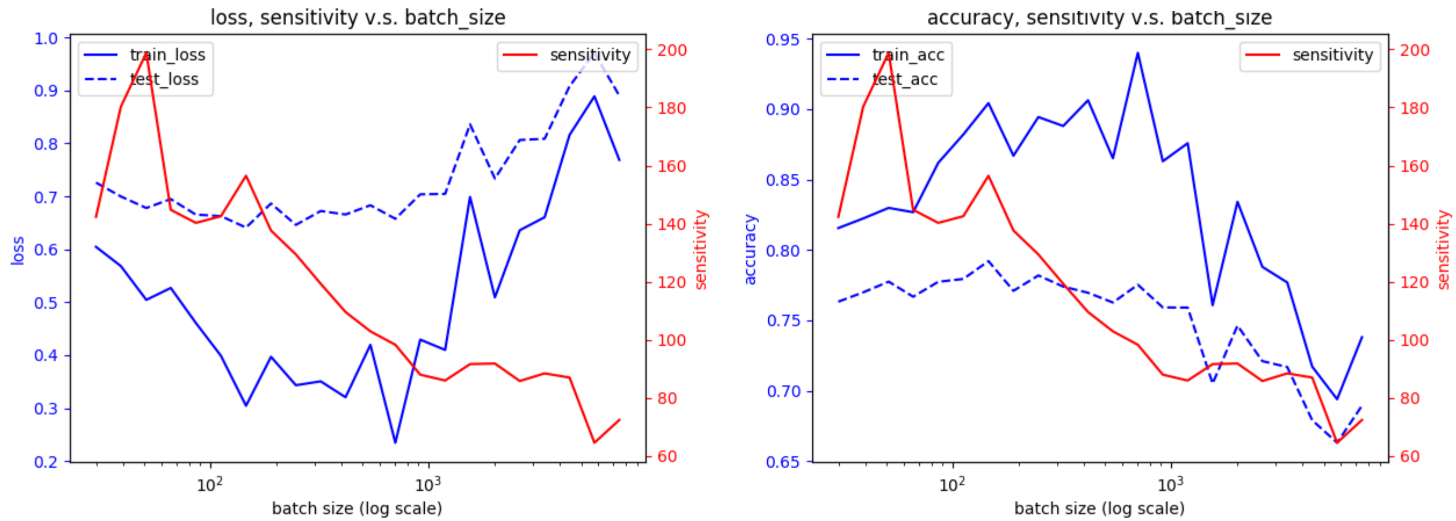
- **What is sensitivity:**
  - Reference: https://arxiv.org/pdf/1802.08760.pdf
  - Original definition:
    - Frobenius norm of Jacobian matrix of model output (class probability) to input
    - Computationally expensive for us
  - Our definition:
    - Frobenius norm of gradients of loss to input

Flatness v.s. Generalization - part2

MNIST :



CIFAR10 :



HW1-3 Report Questions

- **Can network fit random labels?**
  - Describe your settings of the experiments. (e.g. which task, learning rate, optimizer)
  - Plot the figure of the relationship between training and testing, loss and epochs.
- **Number of parameters v.s. Generalization**
  - Describe your settings of the experiments. (e.g. which task, the 10 or more structures you choose)
  - Plot the figures of both training and testing, loss and accuracy to the number of parameters.
  - Comment your result.
- **Flatness v.s. Generalization**
  - Part 1:
    - Describe the settings of the experiments (e.g. which task, what training approaches)
    - Plot the figures of both training and testing, loss and accuracy to the number of interpolation ratio.
    - Comment your result.
  - Part 2 :
    - Describe the settings of the experiments (e.g. which task, what training approaches)
    - Plot the figures of both training and testing, loss and accuracy, sensitivity to your chosen variable.
    - Comment your result.
  - Bonus : Use other metrics or methods to evaluate a model's ability to generalize and concretely describe it and comment your results.

Submission

- **Deadline: Feb. 11th 23:59**
- **Allow package：**
  - python 3
  - **TensorFlow/pytorch ONLY** for CS and ECE student
  - For non-CS/ECE students, Keras is allowed.
- **Write one report**
- **Share your github with TA**
  - Code
  - In your Readme, state clearly how to run your program to generate the results in your report.
  - Files for training is required.

Flatness v.s. Generalization - bonus example

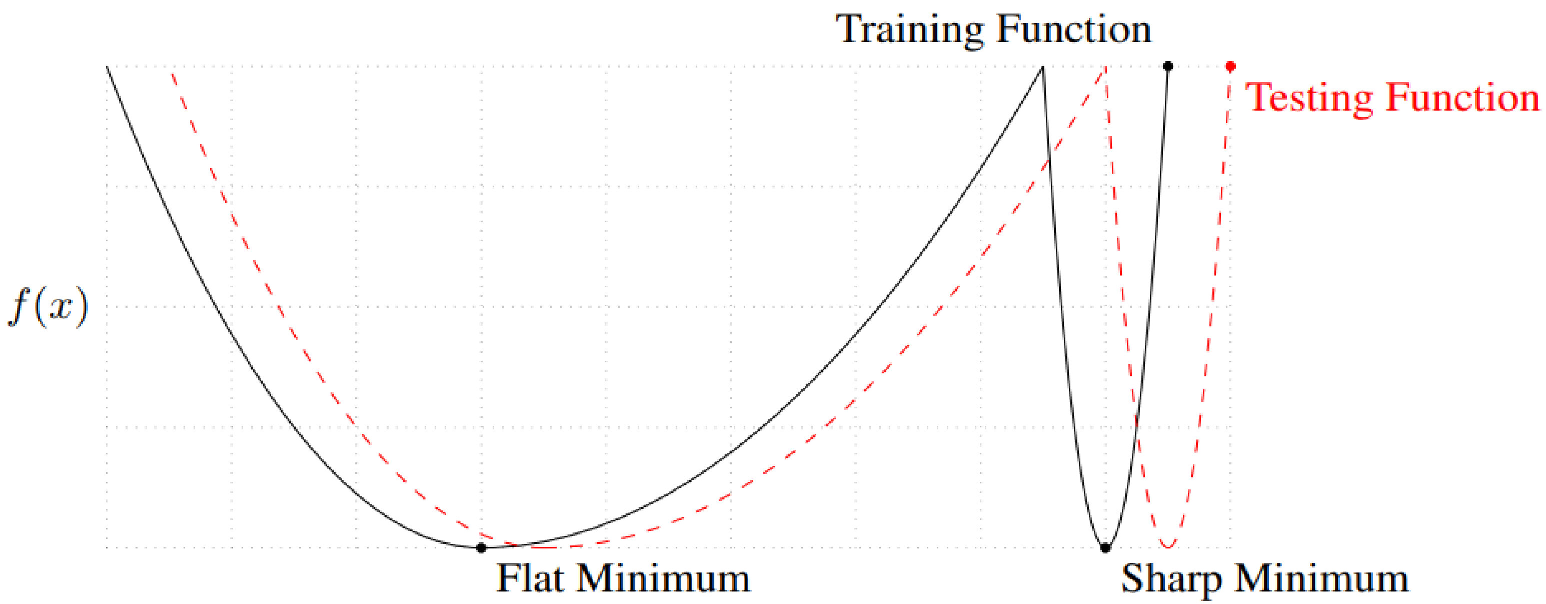
- Reference : https://arxiv.org/pdf/1609.04836.pdf
- Requirement
  - Train on MNIST or CIFAR-10
  - Use at least ten different approaches to train the same model
  - Calculate the sharpness of trained models
- Tips
  - Train on MNIST
  - Train with different batch size

Flatness v.s. Generalization - bonus example

- There is a generalization gap when using large-batch (LB) methods (instead of small-batch (SB) methods) for training deep learning models.
- The reasons (maybe more than these):
  - LB methods lack the explorative properties of SB methods and tend to zoom-in on the minimizer closest to the initial point.
  - SB and LB methods converge to qualitatively different minimizers with differing generalization properties (i.e. SB converges to flat minimizer, LB converges to sharp minimizer)
- We will focus on the second reason.

Flatness v.s. Generalization - bonus example

- Visually, it means that :



- How to measure sharpness (or flatness) ?

Flatness v.s. Generalization - bonus example

- Many methods can measure sharpness, but we will only utilize one in this assignment.
- Notations:
  - θ : vector of all parameters
  - $L(\theta)$: loss of the model given the paramters
  - $B_2(\epsilon, \theta)$: a Euclidean ball centers at θ with radius ϵ
- Sharpness:

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

Flatness v.s. Generalization - bonus example

- How to calculate this : $\frac{\max_{\theta' \in B_2(\epsilon,\theta)}(L(\theta')-L(\theta))}{1+L(\theta)}$
- Original paper : Use L-BFGS-B to maximize $L(\theta')$
- Around a critical point :

$$L(\theta') = L(\theta) + \frac{1}{2}(\theta' - \theta)^T(\nabla^2 L)(\theta)(\theta' - \theta) + o(\|\theta' - \theta\|_2^2)$$

- Then :

$$\frac{\max_{\theta' \in B_2(\epsilon,\theta)}(L(\theta')-L(\theta))}{1+L(\theta)} \longrightarrow \frac{\|(\nabla^2 L)(\theta)\|_2 \epsilon^2}{2(1+L(\theta))}$$

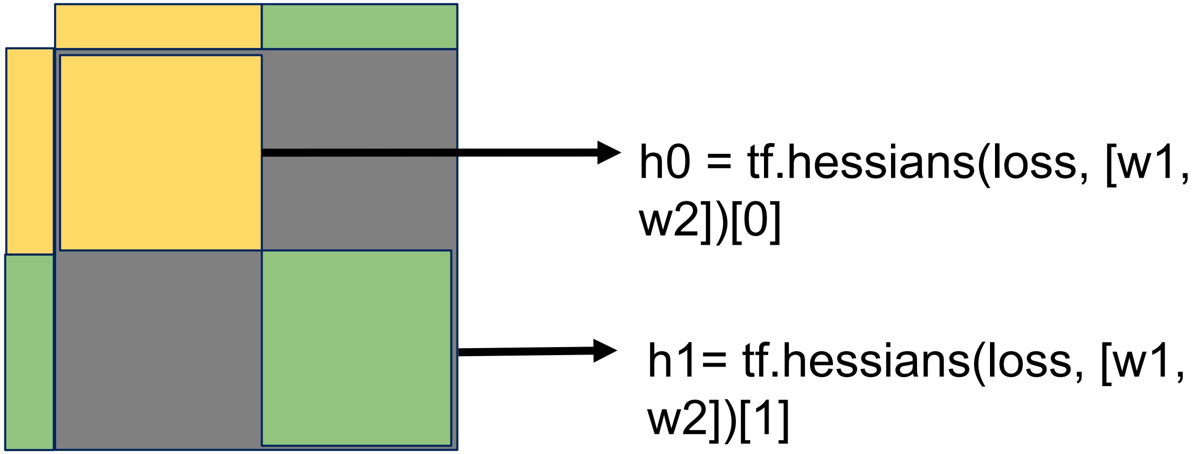
- Since 2-norm of a matrix is defined as :

$$\|A\|_2 = \max\{\|Ax\|_2 : x \in R^n \text{with} \|x\|_2 = 1\} = \sqrt{\lambda_{\max}(A^T A)}$$

Flatness v.s. Generalization - bonus example

- How to calculate Hessian matrices efficiently:
  - Use GPU : tf.hessians
  - Calculate only 500 out of 60000 examples in MNIST
- But tf.hessians only return block-diagonal part:
  - vector of all paramters : 

  - Hessian matrix :

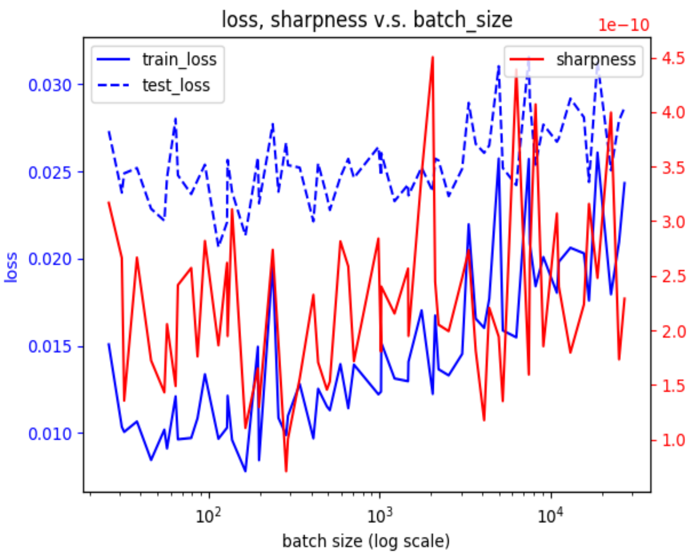


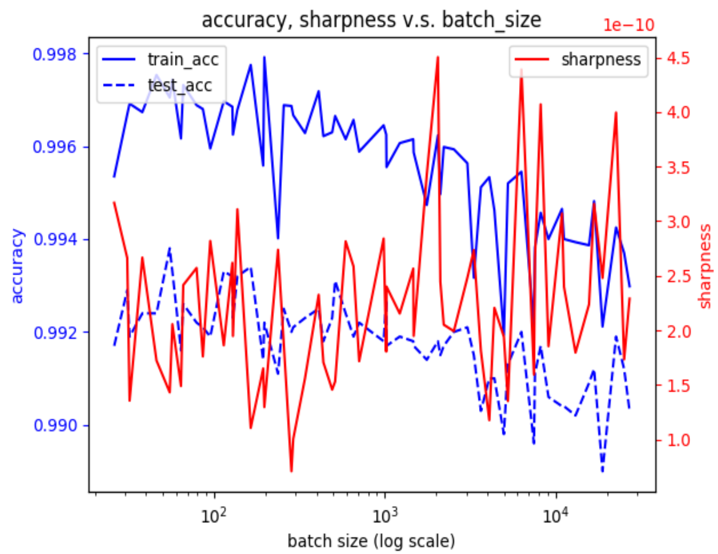
Flatness v.s. Generalization - bonus example

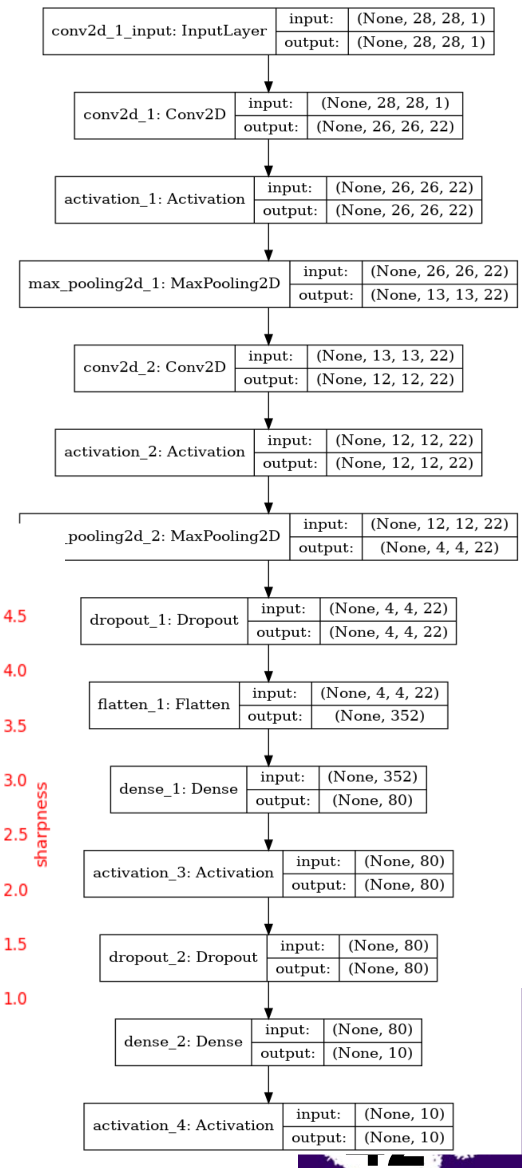
- If we assume off-block-diagonal elements is negligable:
  - Square of block-diagonal matrix is also block-diagonal.
  - Eigenvalues of a block-diagonal matrix is the list of all eigenvalues of each block submatrix.
  - Since we only want the largest eigenvalue, we can conclude that the 2-norm of a block-diagonal matrix is the 2-norm of block submatrix that contains the largest eigenvalue itself.
  - 2-norm of matrix A in tensorflow : tf.norm(A, 2)
  - 2-norm of matrix A in numpy : np.linalg.norm(A, 2)

Flatness v.s. Generalization - bonus example

- MNIST :
  - 20000~30000 parameters (in order to calculate hessian matrices while maintaining enough model capacity)
  - Calculate hessian matrices as mentioned in previous slide
  - epsilon : 1e-4







Flatness v.s. Generalization - more possible bonus

- Reference: https://arxiv.org/pdf/1703.04933.pdf
- This paper shows that several metrics (including sharpness) do not indicates ability of generalization for any RELU-based deep models.
- Reparametrize:
  - $\mathrm{relu}(x \cdot (\alpha\theta_1)) \cdot \theta_2 = \mathrm{relu}(x \cdot \theta_1) \cdot (\alpha\theta_2)$

    if $\alpha > 0$