Lab4: Model Pruning

TA: 林玦菩

Contact: course.aislab@gmail.com

Outline

- Introduction to model compression techniques
- Case study – Network Slimming
- Lab4 introduction

Model Compression Techniques

Why we need model compression

- Size of Model become larger nowadays. To put model on resource constrained device, the need of model compression is rising.

- For instance, size of the pre-trained VGG16 model is more than 500 MB. Such model size is large for the utilizing on resource constrained device

Model Compression Techniques

- **Knowledge Distillation**: Distill the knowledge from a large deep neural network into a small network

- **Quantization**: Reduce the number of bits required to represent each weight. For example, the weights can be quantized to lower bits (e.g. from float32 to int8)

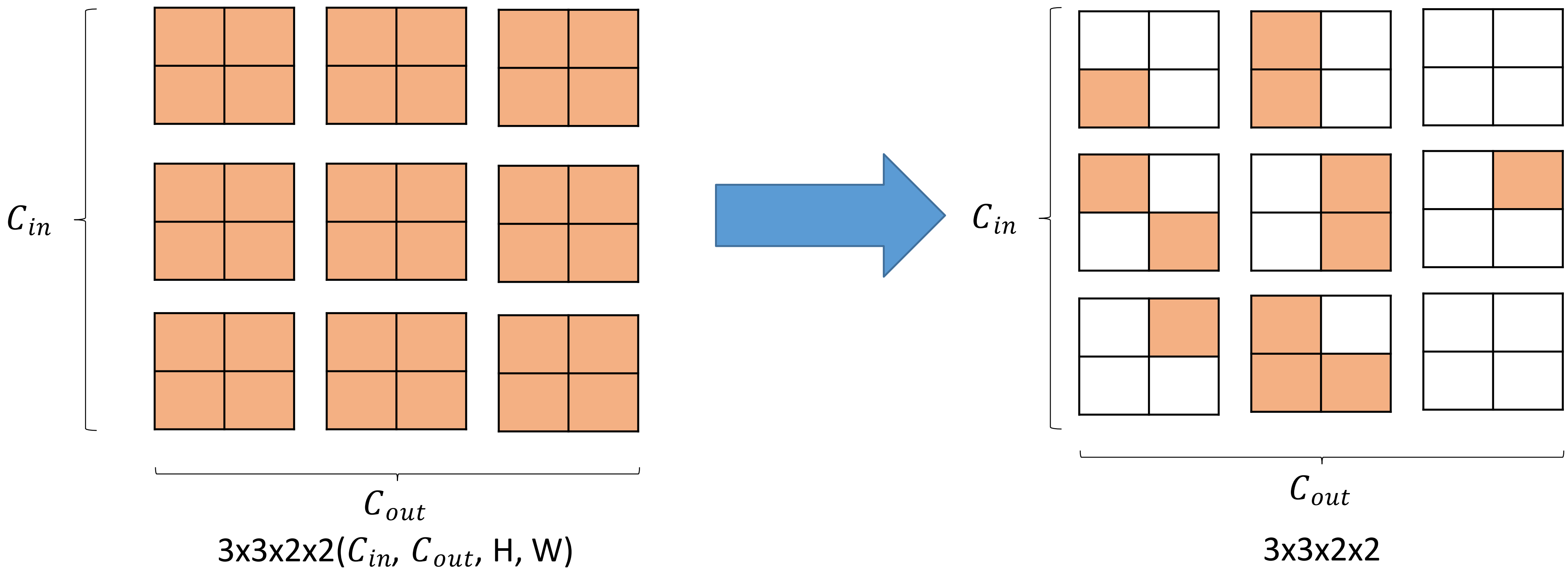
- **Pruning**: Removing inessential parameters from deep neural networks without significant effect on the performance

Categories of Pruning

- Unstructured pruning
- Structured pruning

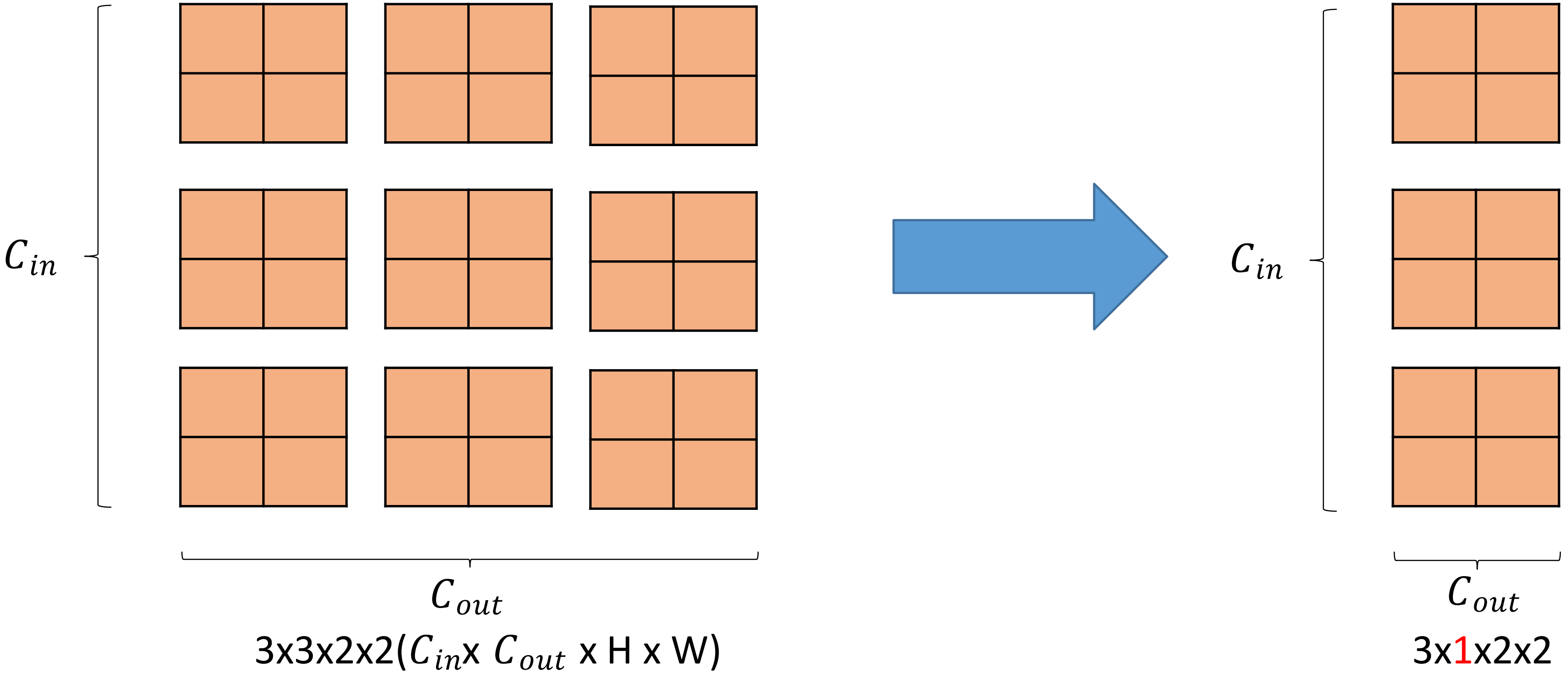
Unstructured Pruning

- Pros: Could achieve higher sparsity rate
- Cons: Need special library support to reshape filter for sparsity operation



Structured Pruning

- Pros: Easy to implement
- Cons: Not easy to achieve higher sparsity rate



Case Study – Network Slimming

Network Slimming

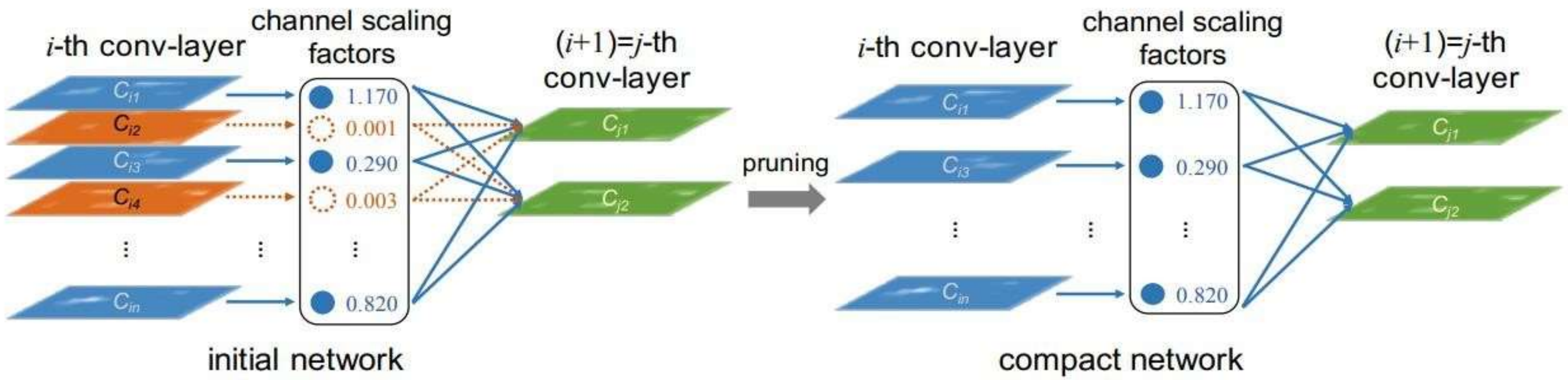
- Title: Learning Efficient Convolutional Networks through Network Slimming
- Authors: Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, Changshui Zhang1
- Year: 2017 ICLR

- Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan and C. Zhang, "Learning Efficient Convolutional Networks through Network Slimming," 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2755-2763, doi: 10.1109/ICCV.2017.298.

- Reference: https://ieeexplore.ieee.org/document/8237560

Network Slimming

- Channel-wise pruning method, which utilize batch normalization layer statistic (**scaling factor**) to help evaluate what to prune
- Sparsity regularization is imposed on these scaling factors to identify unimportant channels



Batch Normalization Layer

- By normalizing the inputs to each layer, batch normalization reduces the sensitivity of the network to the initial weight values and the learning rate.

- The network can utilize higher learning rate without exploding or vanishing gradients, and converge faster to a good solution.

Batch Normalization Layer

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \text{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

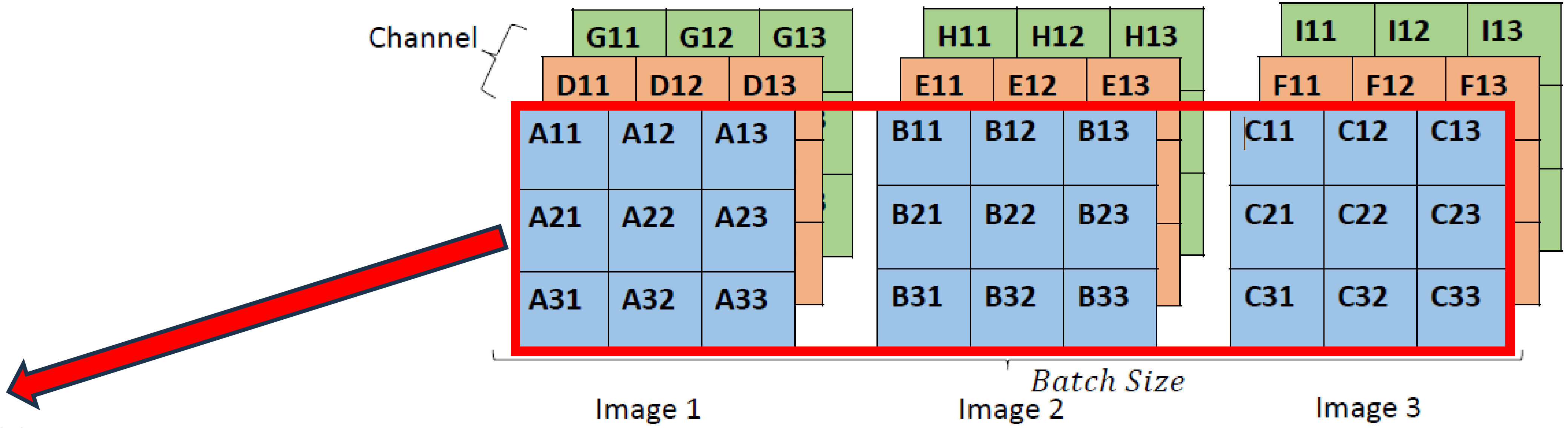
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \text{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Batch Normalization Layer – Training

Suppose batch size = 3, we have 3 images after convolution layer output



Calculate Mean:

$$\mu_1 = (A11 + \ldots. + A33 + B11 + \cdots + B33 + C11 + \ldots + C33) \,/\, (9*3)$$

Calculate Variance:

$$\sigma_1^2 = [(A11 - \mu_1)^2 + \ldots + (A33 - \mu_1)^2 + (\mathrm{B}11 - \mu_1)^2 + \ldots + (\mathrm{B}33 - \mu_1)^2 + (\mathrm{C}11 - \mu_1)^2 + \ldots + (\mathrm{C}33 - \mu_1)^2] \,/\, (9*3)$$

Normalize:

$$A11' = (A11 - \mu_1)/\sqrt{\sigma_1^2 + \varepsilon}$$

Scale and Shift:

$$A11_{new} = \gamma * A11' + \beta$$

Scaling factor Shift factor

Batch Normalization Layer – Inference

- In inference stage, we may not able to calculate μ and σ. The solution is to adapt running mean and running variance calculated in training stage

Calculate running mean at training stage:

$$\mu_{mov} = \alpha * \mu_{mov} + (1 - \alpha) * \mu_1 , 0 \leq \alpha \leq 1$$

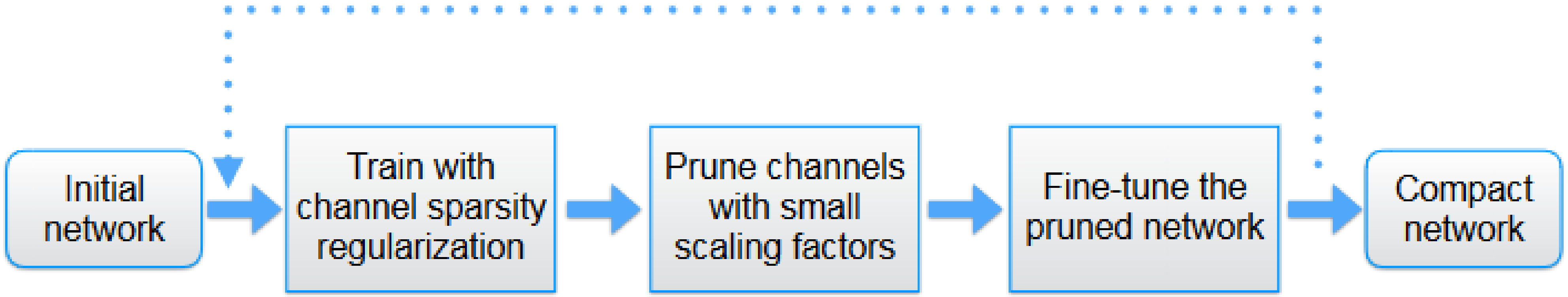
Calculate running variance at training stage:

$$\sigma_{mov} = \alpha * \sigma_{mov} + (1 - \alpha) * \sigma_1 , 0 \leq \alpha \leq 1$$

At inference stage, utilize $\boldsymbol{\mu}_{\mathbf{mov}}$ and $\boldsymbol{\sigma}_{\mathbf{mov}}$

α: momentum (hyperparameter)

Network Slimming Pipeline



Sparsity regularization

- Sparsity regularization penalizes scaling factors

$$L = \underbrace{\boxed{\sum_{(x,y)} l(f(x,W),y)}}_{\text{Classification Loss}} + \lambda \underbrace{\boxed{\sum_{\gamma\in\Gamma} g(\gamma)}}_{\text{Sparsity Regularization Loss}}$$

$x = input, y = label, W = weight, l(\cdot) = loss\ function$

$\lambda = balance\ factor, g(\cdot) = \text{sparsity−induced penalty on scaling factors}, \gamma = scaling\ factor$

Sparsity regularization

- In the view of backward propagation (updating scaling factors)

$$\gamma_{new} = \gamma_{old} - \eta \left(\frac{\partial Loss}{\partial \gamma_{old}} + \boxed{\frac{\partial(\lambda \sum g(\gamma))}{\partial \gamma_{old}}}\right)$$

Classification Loss

In this Lab, you need to figure out how to calculate this term !

- In Network Slimming, $g(s) = |s| \Longrightarrow g(r) = |r| \begin{cases} \gamma & if\ \gamma \geq 0 \\ -\gamma & if\ \gamma < 0 \end{cases}$

L1 norm, widely used to achieve sparsity

Lab4 Introduction

Goal and Grading

- Implement algorithm performed in Network Slimming.
- Architecture: VGG
- Dataset: CIFAR10
- Grading:
  - Fill blanks (total 35%, 7 blanks * 5%)
  - Complete different prune ratio (total 15%)
    - Prune ratio **0.5** (5%)
    - Prune ratio **0.9** (10%)
  - Complete scaling factor distribution visualization (10%)
  - Report (total 40%)
    - Plot **sparsity-training** accuracy of origin model over epochs (5%)
    - Plot scaling factor distribution with **3 different λ value** (5%)
    - Show model test accuracy after pruning **50%** channels (5%)
    - Show model test accuracy after pruning **90%** channels (5%)
    - Plot training (**fine-tuning**) accuracy of pruned **90%** model over epochs (5%)
    - Show what problem you encounter and how you solve it (15%)
- **Due: 2023/11/8 23:59**

Scaling factor distribution visualization

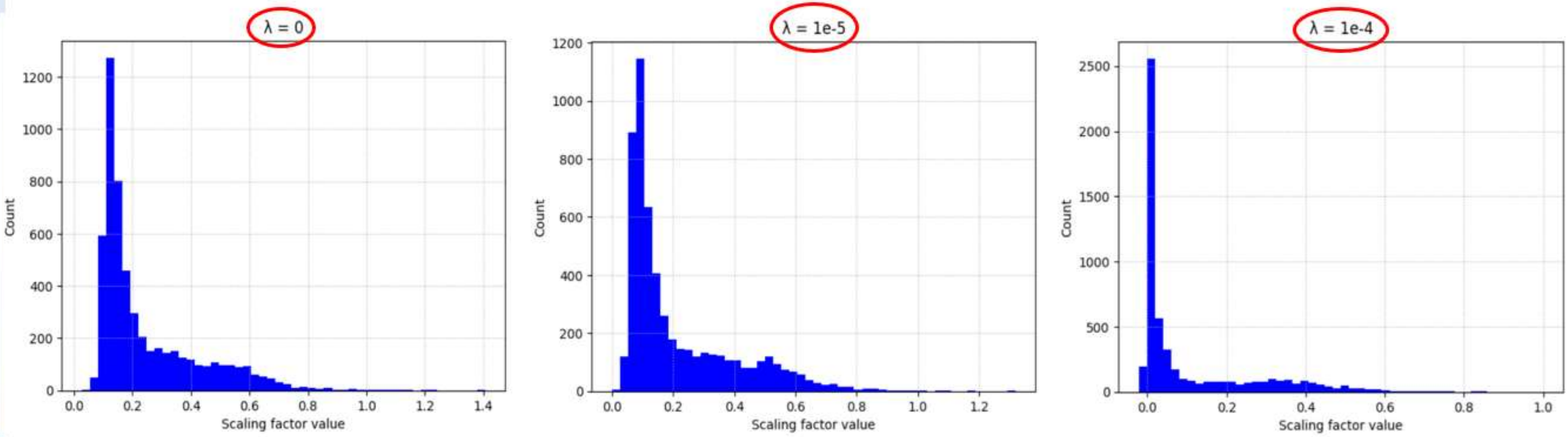
- Sparsity regularization in **Network Slimming** paper

$$L = \overbrace{\boxed{\sum_{(x,y)} l(f(x,W),y)}}^{\text{Classification Loss}} + \underbrace{\lambda}_{\text{balance factor}} \overbrace{\boxed{\sum_{\gamma\in\Gamma} g(\gamma)}}^{\text{Sparsity Regularization Loss}}$$

```
#####  設定λ(balance  factor)  #####
################################################
#                    請填空                    #
################################################
LAMBDA  =1e-4
```

Scaling factor distribution visualization

- Experiment of scaling factor distribution on different λ



With the increase of λ, scaling factors become sparser !

You should run 3 experiments with 3 different λ value and put the results in the report !

Procedure

- 1. Download archive from Moodle
- 2. Extract archive and upload file to Google Drive (Recommend under Colab NoteBooks/)

Procedure

- 3. Double click **sparsity_train.ipynb** would navigate to CoLab UI
- 4. Run **sparsity_train.ipynb** for sparsity regularization training
- 5. Run **vggprune.ipynb** for model pruning
- 6. Run **train_prune_model.ipynb** to train pruned model (fine-tune)

Double Click *.ipynb

Procedure

- 7. Running code in Colab will save running history
- 8. Hand in files as archive

Hand in archive to Moodle with code and report
Hand in code after running on Colab. Colab will save running history.

<<File Hierarchy>>
EAI_Lab4_StudentID.zip
 - vggprune.ipynb
 - train_prune_model.ipynb
 - sparsity_train.ipynb
 - EAI_Lab4 _StudentID_Report.pdf
 - models/

DO NOT ATTACH *.pth and dataset

- **Due: 2023/11/8 23:59**

Thanks for listening

TA: 林鈇菩

Contact: course.aislab@gmail.com