

深度學習系統與實現

LAB01 - Basic DL framework & model architecture

Dept. of Computer Science and
Information Engineering
National Chiao Tung University



Outline

LAB 1-1, Train a CNN model via pytorch

Review of some concept

LAB 1-2, Use [thop](#) to count MACs of your model

LAB 1-3, Count the MACs / FLOPs of your model
via the customized forward hook function

Notices & Hints

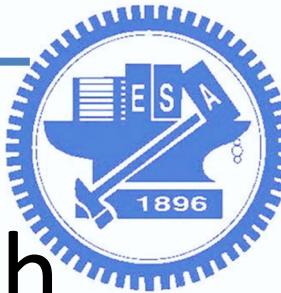
Questions

Grading



Prerequisite

- ❑ Most of Labs for this semester are based on Pytorch ($>=1.2$, 1.2 is best for deployment)
- ❑ The GPU resource requirements
 - ❑ Maxwell (只推薦 980ti, Titan)
 - ❑ Pascal (1060 6G 或以上)
 - ❑ Turing (1660 6G, 或是 RTX 家族) / Volta (Titan V)



LAB 1-1

Train a CNN model via pytorch

We provide a cifar10 example on New E3

[? LAB-1-1-example-code.ipynb](#)

Just need to change the task to the other dataset

You should show total accuracy (test case)

You should show accuracy for each class

We only verify the correctness of your function usage
& workflow (total accuracy > 60% is OK!)



LAB 1-1

Dataset - skewed_food11

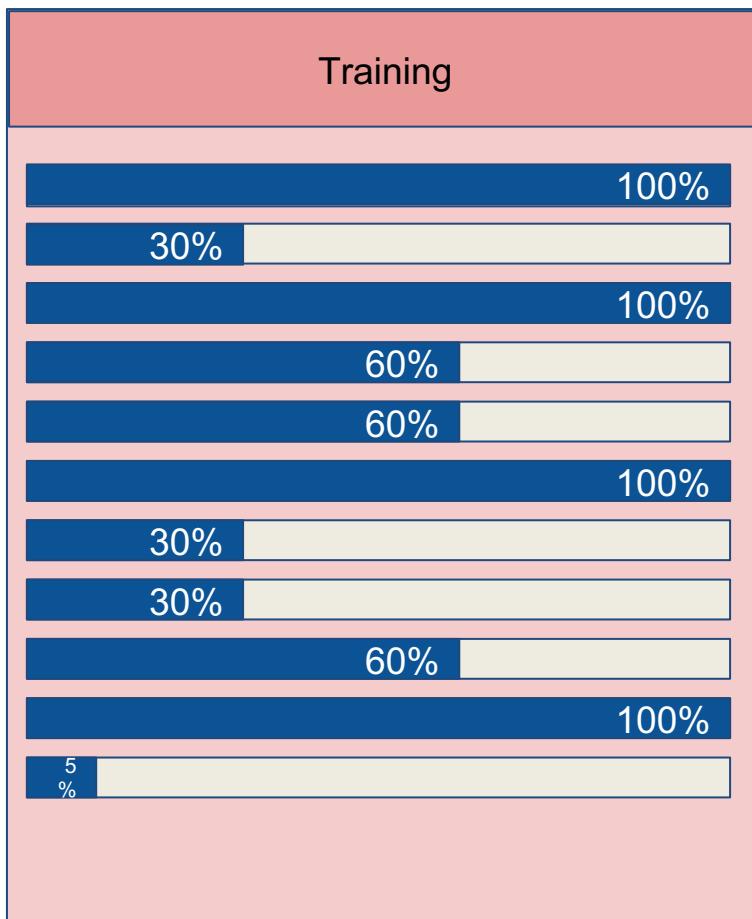
- ❑ Food11 Download link - <https://www.kaggle.com/tohidul/food11>
- ❑ Transfer to skewed_food11 - [build_imbalanced_food11.sh](#)
 - ❑ step 1. copy the script file into the dataset folder /food11
 - ❑ step 2. run this script file (build_imbalanced_food11.sh)

 0	Bread	 1	Dairy products	 2	Dessert	
 3	Egg	 4	Fried food	 5	Meat	
 6	Noodles & Pasta	 7	Rice	 8	Seafood	
 9	Soup	 10	Vegetables & Fruits	 11		



LAB 1-1

Dataset - skewed_food11

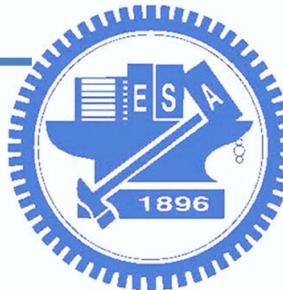


Validation

Fixed!

Evaluation

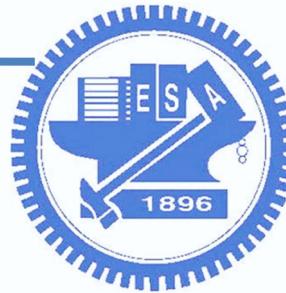
Fixed!



LAB 1-1

Output format (ref.)

Lab 1-1:		
Test set: Top 1 Accuracy: 2925/3347 (87%) , Top 3 Accuracy: 3264/3347 (98%)		
Class 0 :	302/368	82.07%
Class 1 :	100/148	67.57%
Class 2 :	460/500	92.00%
Class 3 :	268/335	80.00%
Class 4 :	237/287	82.58%
Class 5 :	410/432	94.91%
Class 6 :	141/147	95.92%
Class 7 :	93 /96	96.88%
Class 8 :	263/303	86.80%
Class 9 :	482/500	96.40%
Class 10:	169/231	73.16%

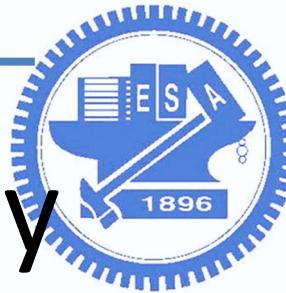


Review of some concept

Metrics of model efficiency

MACs

FLOPs



Metrics of model efficiency

the **increased** model complexity leads to a **higher** computational burden

infeasible to deploy on the edge where the compute capacity is limited

How should we evaluate the efficiency (complexity) of a neural network?



How to evaluate complexity

□ Measurement

- via some profiling tools, to record runtime of each ops
- consider the real cases, ex: memory behavior ...
- most of DL frameworks provide their own profiler
 - [TensorFlow's profiler](#), [PyTorch's profiler](#) (only record timestamp)

□ Theoretical metrics

- MACs
- FLOPs

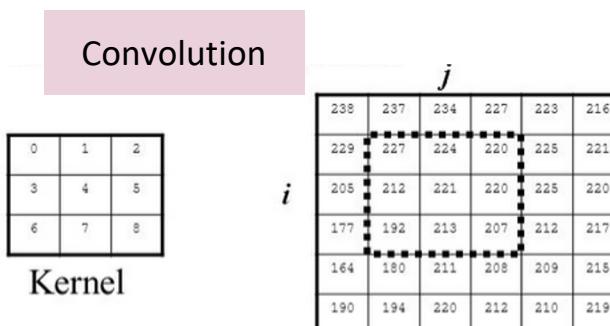


MACs

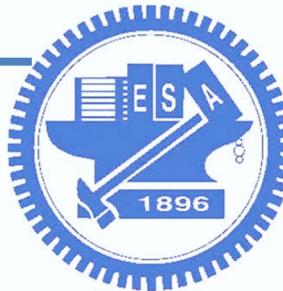
- In computing, especially digital signal processing, the multiply–accumulate(MAC) is a common operation
- Most layers in CNN model can be composed of many MAC operations - like "convolution"
- It's common used to estimate complexity

MAC Op

$$a \leftarrow a + (b \times c)$$

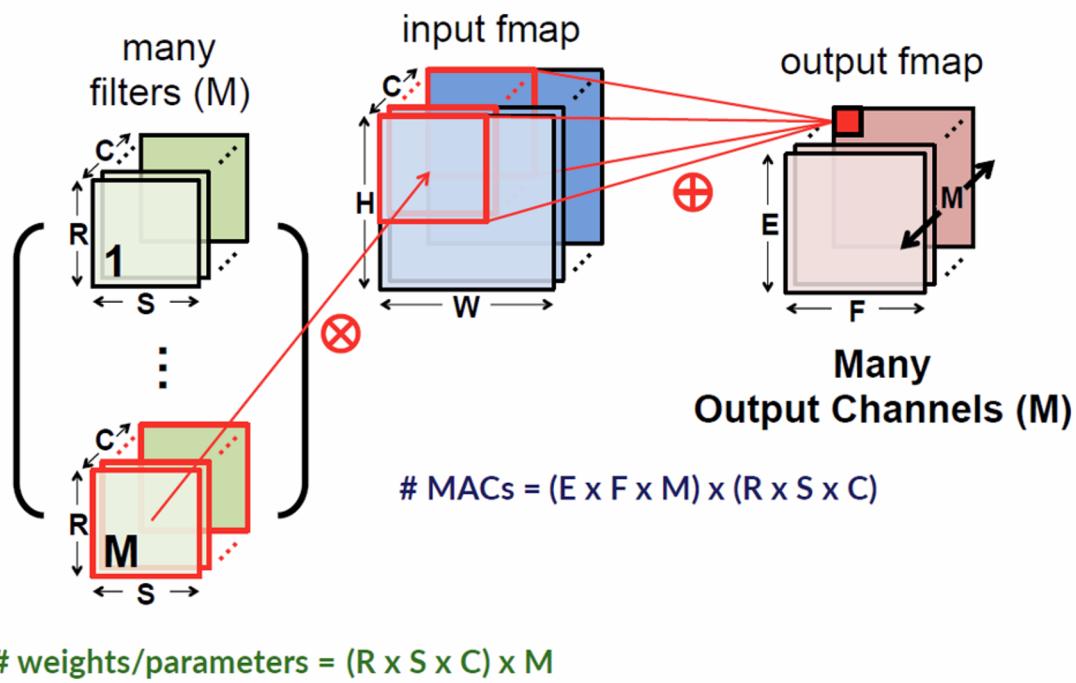


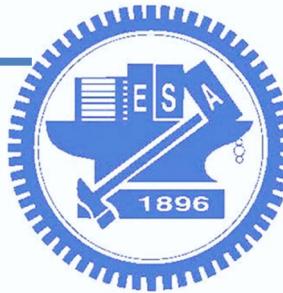
$$R[i][j] = (0 * 227) + (1 * 224) + (2 * 220) + \\ (3 * 212) + (4 * 221) + (5 * 220) + \\ (6 * 192) + (7 * 213) + (8 * 207)$$



MACs - example

Convolution





FLOPs

- ❑ Count number of floating-point operations
- ❑ Rely on your instruction set architecture & implement method
- ❑ In most paper, just use MACs to represent FLOPs, but they are totally different
- ❑ If implement the convolution with "dot product instruction" rather than "MAC" ?



FLOPs - example

A.1 FLOPs COMPUTATION

To compute the number of floating-point operations (FLOPs), we assume convolution is implemented as a sliding window and that the nonlinearity function is computed for free. For convolutional kernels we have:

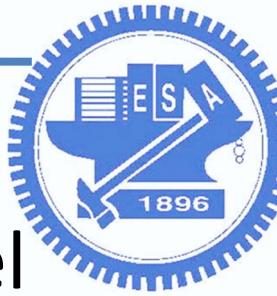
$$\text{FLOPs} = 2HW(C_{in}K^2 + 1)C_{out}, \quad (11)$$

where H , W and C_{in} are height, width and number of channels of the input feature map, K is the kernel width (assumed to be symmetric), and C_{out} is the number of output channels.

For fully connected layers we compute FLOPs as:

$$\text{FLOPs} = (2I - 1)O, \quad (12)$$

where I is the input dimensionality and O is the output dimensionality.



LAB 1-2

Use [thop](#) to count MACs of your model

- ❑ THOP: PyTorch-OpCounter
- ❑ Choose some models from [torchvision](#)
 - ❑ at least 2 kinds (recommend - ResNet, MobileNetV2)
- ❑ How to use

```
from torchvision.models import resnet50
from thop import profile
model = resnet50()
input = torch.randn(1, 3, 224, 224)
macs, params = profile(model, inputs=(input, ))
```

- ❑ Output format

Lab 1-2:	
Total params:	2.238M
Total MACs:	312.879M



LAB 1-3

Count the MACs / FLOPs of your model via the customized forward hook function

- you also need to **count the FLOPs** in LAB 1-3
 - should be **different with your MACs** results
- need to show **layer-wise MACs** information
- **customize** your own MACs calculator for each layer
- hook function example - [pytorch_hook_sample.py](#)

```
def my_hook_function(self, input, output):
    print("Op:{}".format(str(self.__class__.__name__)))
    for param in self.parameters():
        print("params shape: {}".format(list(param.size())))

def main():
    model = SampleNet()
    model.conv1.register_forward_hook(my_hook_function)
    input_data = torch.randn(1, 3, 224, 224)
    out = model(input_data)
```



LAB 1-3 Output Format

- ❑ You don't need to follow the format, but you need to provide each layer's information with the following items
 - ❑ op_type
 - ❑ input_shape
 - ❑ output_shape
 - ❑ params
 - ❑ MACs

Lab 1-3:				
op_type	input_shape	output_shape	params	MACs
Conv2d	[1, 3, 224, 224]	[1, 64, 112, 112]	9408	118013952
...
...
Linear	[1, 512]	[1, 11]	5643	5632
<hr/>				
Total params: 11.182 M				
Total MACs: 1.816G				



Notices & Hints

Please carefully confirm which layers need to count MACs (hint: compare with the works from thop)

How many FLOPs for the non-linear layer ?
Do we need to count MACs for pooling layer ?



Questions

You can try to answer the following questions in your report (**TAs will ask you during the DEMO**)

Do you get the same result from 'thop' & your own functions ? If not, please explain.

How could you get a difference result between FLOPs and MACs ?

Is MACs/FLOPs a good metric to estimate the real inference latency ? If not, please explain.



Grading

LAB 1-1 (60%)

LAB 1-2 (15%)

LAB 1-3 (25%)

Bonus (10%)

Total:

110

- In LAB 1-3, if your selected model contains the following special layers, TAs will give extra points
 - Dilated (Atrous) Convolution, Deconvolution, Others(ask TAs)

Submission: source code + report (.ipynb is accepted)(E3)

- zip format (ex: DLSR_lab1_{student id}.zip)
- 未依照上述命名格式者，扣該Lab成績5分

Deadline : 2020/03/23, 23:59 (Mon)(2 week)

Demo : (TAs will announce date on New-E3 later)



Report Spec.

EX:

- Introduction
- Experiment setup
- Result
- Discussion
- Other ...



Reference

Stanford Course : CS231n

- <http://cs231n.stanford.edu/>

Pytorch Document:

- <https://pytorch.org/docs/stable/index.html>

pytorch_OpCounter:

- <https://github.com/Lyken17/pytorch-OpCounter>