Homework 2

April 17, 2020

[KSE527/AI502] Deep Learning
TA Jongwoo Ko, Younggyo Seo

HW Description

In this homework, you will study about convolution neural networks by using CIFAR100 dataset.

The contents of this homework consist of:

- **Task1** : Implement 18-layer CNNs for CIFAR100 [3]
- **Task2** : Implement ResNet18 [1] for CIFAR100
- **Task3** : Implement MobileNet [2] for CIFAR100

After you make a code for each task, you should write a report according to the each problem.

Source Code
Download homework2.zip from KLMS that contains the source code for three tasks.

Before tasks : GPU setting

Because this homework takes a lot of time if you do on CPU, we recommend you to use GPU. If you do not have a GPU machine, please use Google Colab. The manual about how to use GPU on Colab is included in homework2.zip file. (Also, the provided code is GPU-based code.)

Before tasks : Change the indent size (For Colab User)

For python, the indent size is very important. If indent is not proper, program will not run because of syntax error. Anyway, the indent size of our code is 4. When you use Colab first, this size might be set 2. Therefore, you have to change the indent size from 2 to 4 on Colab. In our Colab manual file, there is the way to change this size.

Changes

(2020.04.17 18:33) The number of channels for conv0 is changed to 64 (from 32) in Figure 1 and Figure 3.

(2020.04.20 14:00) We clarified the architecture for shortcut connection in Figure 4.

(2020.04.20 15:00) We fixed the typo (M $\leftrightarrow$ N) in Figure 5.

(2020.04.22 22:00) We fixed the typo (stride 1 $\rightarrow$ 2 in layer14) in Figure 1 and 3.

Homework 2

April 17, 2020

Task1 : Implement 18-layer CNNs for CIFAR100

Convolutional neural networks (CNNs) have been widely used for many computer vision tasks, i.e., image classification, segmentation, and detection. The goal of this task is to implement and train 18-layer CNNs for CIFAR100 dataset. You have to complete code corresponding to Figure 1.

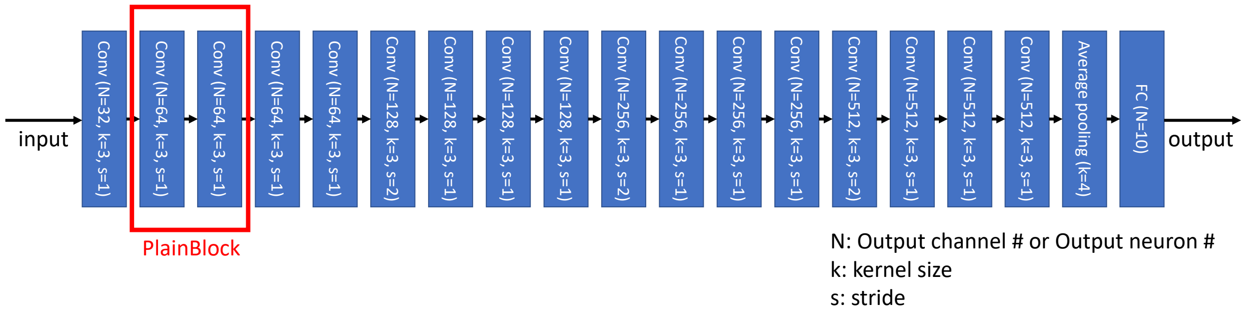


Figure 1: PlainNet18

Notion 1: For all layers, you should use batch normalization and relu activation function after using 3×3 convolution layer as Figure 2.

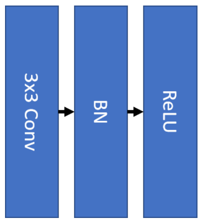


Figure 2: Standard convolutional layer with batchnorm and ReLU.

Notion 2: Also, please read the description for Task2 before you implement the PlainBlock.

Task2: Implement ResNet18 for CIFAR100

As discussed in class, ResNet have achieved strong results on image classification and been widely used in various fields. The crucial ingredient for acheiving such performance is the shortcut connections. **Throughout this task, we will implement ResNet18 using PyTorch and analyze the effect of having shortcuts in network architecture.** You have to complete code corresponding to Figure 3 and 4. For more detailed information about the architecture, see [1].

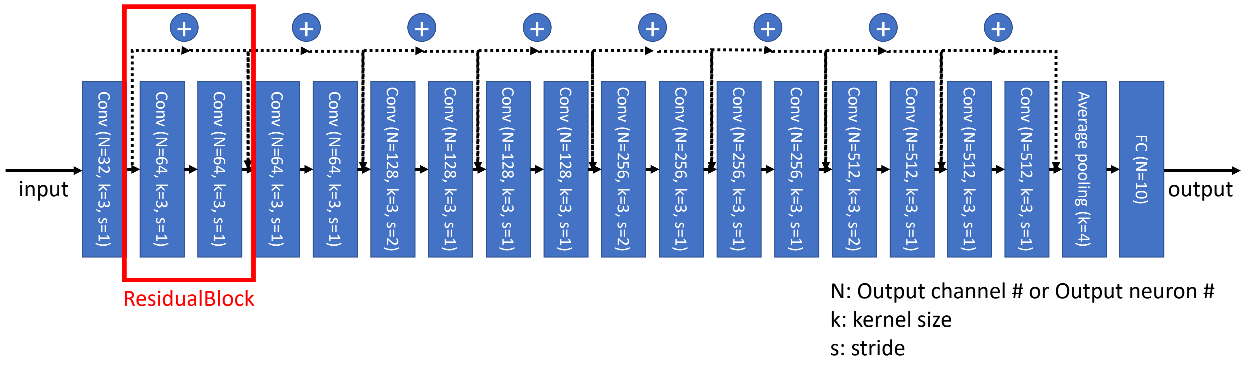


Figure 3: ResNet18

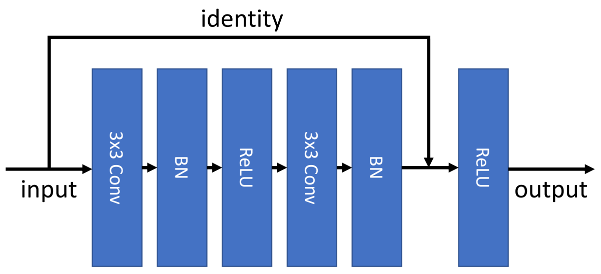


Figure 4: Shortcut connection with batchnorm and ReLU

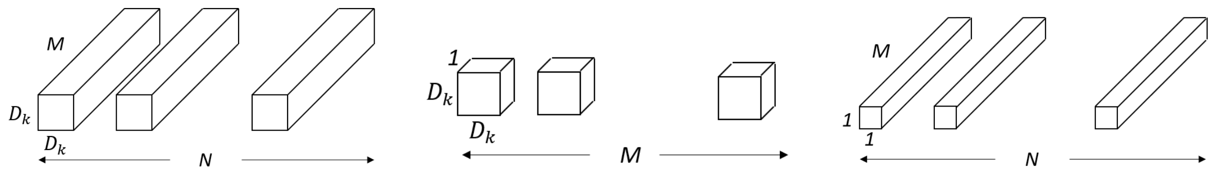
Notion : The dimension of the inputs and outputs connected by shortcut connection in ResNet must have the same dimension. There are two types of shortcut connections.

- Identity connection : Zero padding should be used to increase the dimension.
- Projection shortcut connection : Use 1×1 convolution to match the dimension. (projection shortcut connection is only used when the stride for first convolution layer of block is not equal to 1.)

ResNet18 and PlainNet18 have same structure except for shortcut connection. Use zero padding to maintain the size of the tensor within a block, even for PlainNet18 (and MobileNet18).

Task3: Implement MobileNet for CIFAR100

The MobileNet model [2] is based on depthwise separable convolutions which is a form of factorized convolutions which factorize a standard convolution into a depthwise convolution and a 1×1 convolution called pointwise convolution.



(a) Standard Conv Filters (b) Depthwise Conv Filters (c) Pointwise Conv Filters

Figure 5: The standard convolutional filters in (a) are replaced by two layers: depthwise convolutional in (b) and pointwise convolutional in (c) to build a depthwise separable filter. Note that the M is number of input channels, N is number of output channels, and D_k is kernel size.

Depthwise separable convolution are made up of two layers: depthwise convolutions and pointwise convolutions.We use depthwise convolutions to apply a single filter per each input channel (input depth). Pointwise convolution, a simple 1×1 convolution, is then used to create a linear combination of the output of the depthwise layer. MobileNets use both BatchNorm and ReLU nonlinearities for both layers.

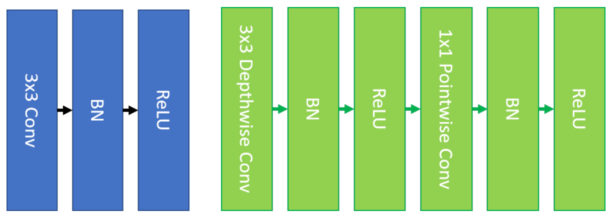


Figure 6: Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by BatchNorm and ReLU.

Note that depthwise convolutions have same kernel size, stride, padding as standard convolutions, and pointwise convolutions have kernel size as 1, stride as 1, and no zero padding. Additionally, set the "groups" in nn.Conv2d as number of input channels when you define the 3×3 depthwise convolutions.

- **Task3-1** : Construct the **conv_dw** function which is used instead of nn.Conv2d function in **MobileBlock** class. The **MobileBlock** is all the same as PlainBlock except that is uses **conv_dw** instead of nn.Conv2d.
- **Task3-2** : Define the **MobileNets** and train the network with CIFAR100.

Homework 2

April 17, 2020

Problem 1 (4pt)

- Plot the training loss and test accuracy of 18-layer CNNs for each epoch. (4pt)

Problem 2 (4pt)

- Plot the training loss and test accuracy of ResNet with plotted result of plain network in Problem 1. (2pt)
- Is there any difference between ResNet and plain network? Are shortcut connections useful? Answer this question and justify your answer based on your experimental results. (2pt)

Problem 3 (6pt)

- Please express the number of parameter for one standard convolutional layer and one depthwise separable convolutions in terms of N, M, and D_k. (2pt)
- Plot the training loss, test accuracy, number of parameters, and number of FLOPs of MobileNet and PlainNet in Problem 1. *https://github.com/sovrasov/flops-counter.pytorch* might be helpful. (2pt)
- While you compare the performance of PlainNet and MobileNet, do you think that the MobileNets are really useful? Describe your opinion and reason in detail. (2pt)

About the Submission

- The deadline for submission is **23:59 on 1 May (Fri)**, and late submission is not permitted.
- The report should be no more than **3 pages** excluding code.
- You have to submit **.zip file** including both **.ipynb file** and **.pdf file**.
  (Please convert .doc file to **.pdf file**)
- File name should be **[hw2]student_ID.zip** (e.g., [hw2]20201234.zip)
  (If you do not keep this naming, there will be a disadvantage.)

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[3] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.