National Tsing Hua University
Fall 2023 11210IPT 553000
Deep Learning in Biomedical Optical Imaging
Homework 4

SHIU-FENG CHENG[1]

[1] *Brain Research Center, National Tsing Hua University, Hsinchu 30013, Taiwan.*

Student ID:111080527

1. Introduction

Existing pre-trained models are convenient tools to use. This report aims to learn to use two of the models in pytorch vision and compare their performance.

2. Result

2.1 TaskA: The pre-trained Convolution Neural Model to be estimated.

The models I chose to test are Resnet18 and Resnet34 which are both based on the Deep Residual Learning for Image Recognition paper [1]. The Resnet models are composed of 4 convolution layers which are called “basic blocks”. A basic block of Resnet is built by 3*3 convolutions followed by 2D batch normalization and ReLU activation function additionally there is a shortcut that forwards (additive) the basic block input to the front of the last ReLU activation function (Fig.1a). The advantage of adding a shortcut to introduce residual term is to reserves the information of early layers. The main difference between Resnet18 and Resnet34 is the number of basic blocks in each layer, Resnet has 2 basic blocks for each layer and Resnet34 has respectively 3, 4, 6, and 3 basic blocks for layer 1, 2, 3, 4.

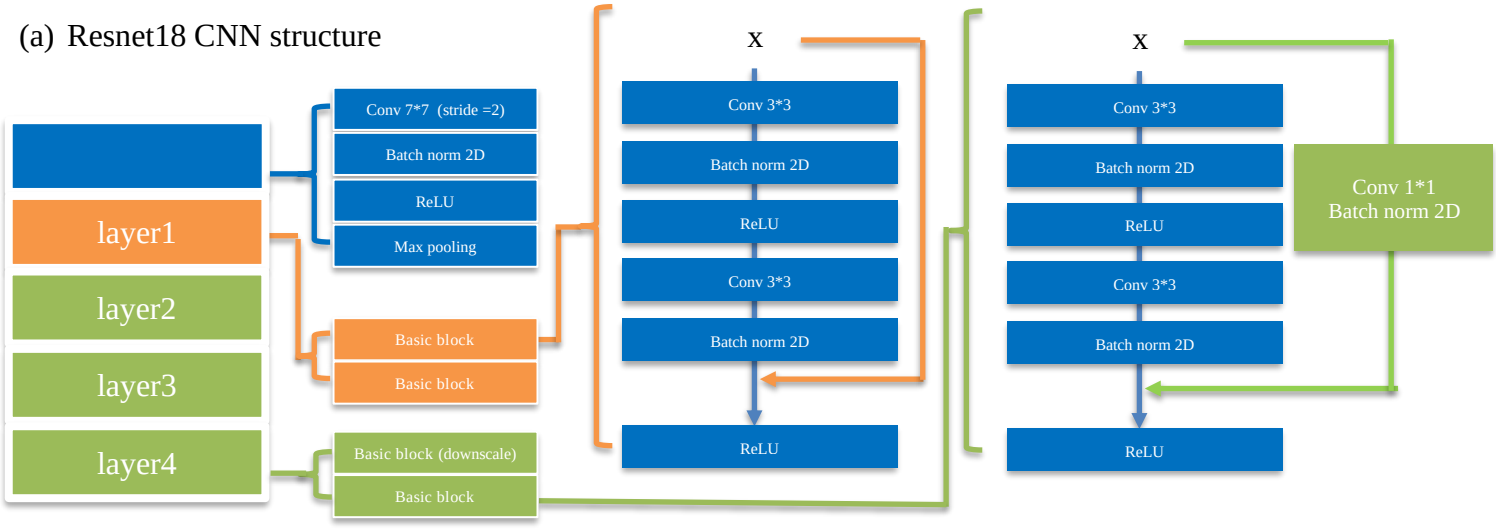

Fig. 2. The Resnet18 model structure. Resnet models shares same basic block structures but having different numbers of building block for each layer.

2.2 Task B~D: Performance Comparison between Resnet18 and Resnet34

The performance of Resnet18 and Resnet34 is shown in Fig.2 the true and false in the legend represent if the weights of CNN layers are updated during the training process. We can find that the Resnet18 and Resnet34 have similar performance on our task which suggests that this task is simple enough for Resnet18 thus having more layers does not improve the performance. Note that having more layers increases the computation burden and takes more time. In my experiment, Resnet18 took 15 seconds to process 400 validation data and Resnet34 took 23 seconds. Thus, to enhance the efficiency of the task the least complex NN network that can perform well is preferred.

The second thing we observe in this result is that having weight updated while training does improve the training performance since it should help the model adapt to the sample type in our task. The validation accuracies are enhanced from ~95% to ~99% (enhanced to 100% for training data) for both Resnet18 and Resnet34. However, it should also be pointed out that updating more weights indicates that the training process takes more time. In my case, it takes 176% more time (64s→177s) for Resnet18 and 185% (99→ 281s) for Resnet34.

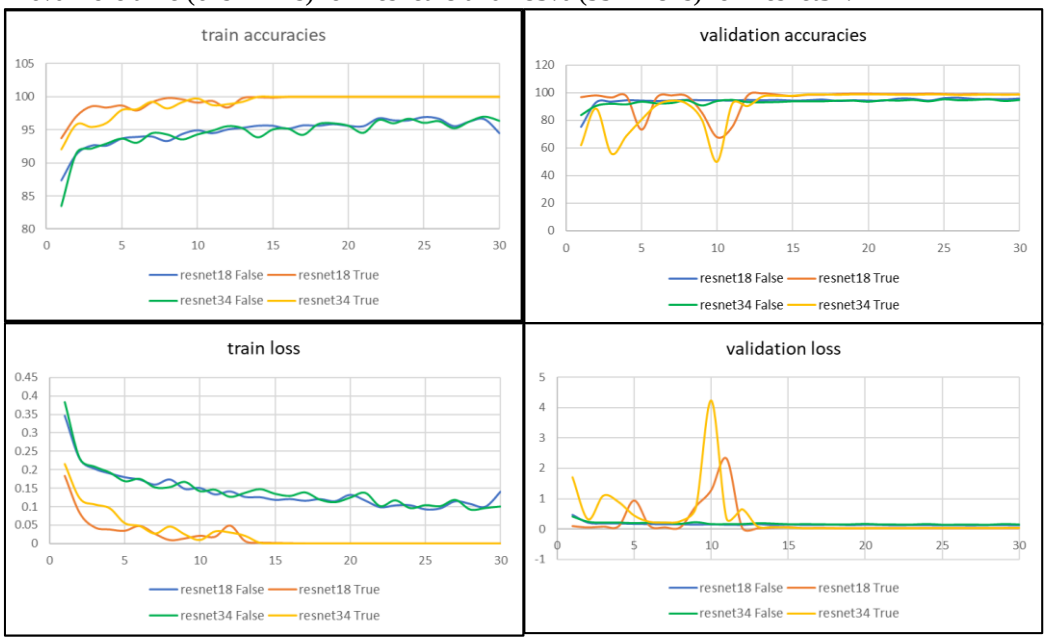

Fig. 2. The performance of the Resnet18 model. Blue/red: Resnet18 model performance when training the model with fixed/updating convolution weight. Green/Yellow: Resnet34 model performance when training the model with fixed/updating convolution weight.

2.3 Task E: Test Dataset Analysis

The testing performance of the fine-tuned Resnet18 is not ideal I tracked the accuracy changing while training and found that it only converges to ~80% (Fig.3b). This is a very low score compared to the validation result which is weird since the validation dataset and test dataset are equivalent to the CNN, they both don't contribute to the update of weight. While guessing the reason for this unusually low score I came up with a hypothesis that the testing dataset might have a large bias compared to the training and validation dataset. To confirm this guess, I shuffled image sets in the training, validation, and testing datasets together and then separated them into new datasets with the same number of image sets. The actual method I use is shown in Fig.3a. I generate a list of indexes from 0~1199 since we have 1200 image

sets in total. Then the index list is segmented into proper length (equal to the demanding length of training, validation, test dataset). Finally, the new training, validation and test dataset are generated according to the assigned index list. By this preprocessing the bias should be eliminated. As I expected, now the testing dataset get similar results with validation dataset (Fig.3c). This experimental result supports my hypothesis since the unusual performance drop on the testing data is eliminated after removing the bias by image shuffling.

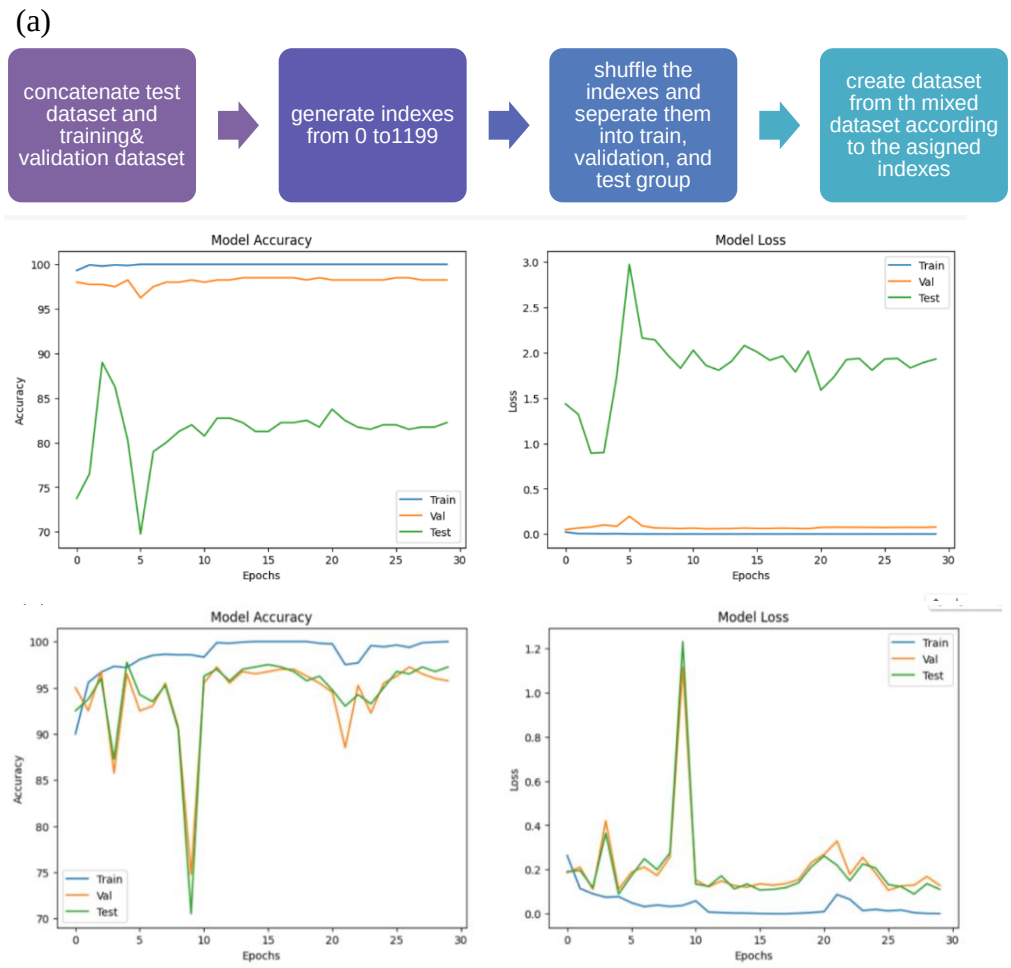

Fig. 3. Test Dataset Analysis. (a) image set shuffling method I adopted to reduce bias. (b) the performance on each dataset without shuffling. (c) the performance on each dataset after pre-shuffling the image sets.

3. Material and Method

3.1 Resnet18 & Resnet34 model comparison

Modified from lab4 code. The network is modified from Resnet18 and Resnet34 to match the demanding output dimension.
The code can be found at:
https://github.com/Frank497/NTHU_2023_DLBOI_HW/blob/main/hw4/2023_hw4.ipynb

3.2 neuron numbers & dropout layer

Modified from lab4 code. The network is modified from Resnet18 to match the demanding output dimension. Also, an image set shuffling preprocessing is added.

The code can be found at:
https://github.com/Frank497/NTHU_2023_DLBOI_HW/blob/main/hw4/2023_hw4_shuffle_dataset.ipynb

4. References

1. He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.