Mini Project: Image Classification

Jianyong Yuan

Update: October 7, 2019

Abstract

This is the summary report of my mini-project, which is about building image classification models and testing them using cifar-10.

1 Environment

I use pytorch as the tool to build the models and tensorboardX to visualize them. Both can be installed by *conda* or *pip*.

2 My Alex Model

2.1 Original Alex Model

Alexnet was a pioneer network that proposed many new ideas and got a high accuracy in the competition. Generally, it uses five layers of CNN and ReLU and then followed by some layers of full connection network. And it also uses local response normalization (LRN) and drop out to avoid overfitting. But as it used two gpu to calculate and its structure got a little complex because of the limitation of hardware at that time, I try to follow its basic principle but simplify it.

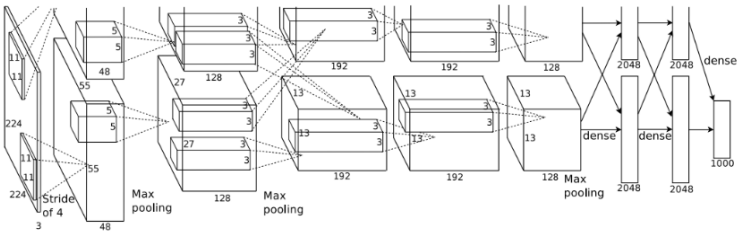

Figure 1: Original Alexnet

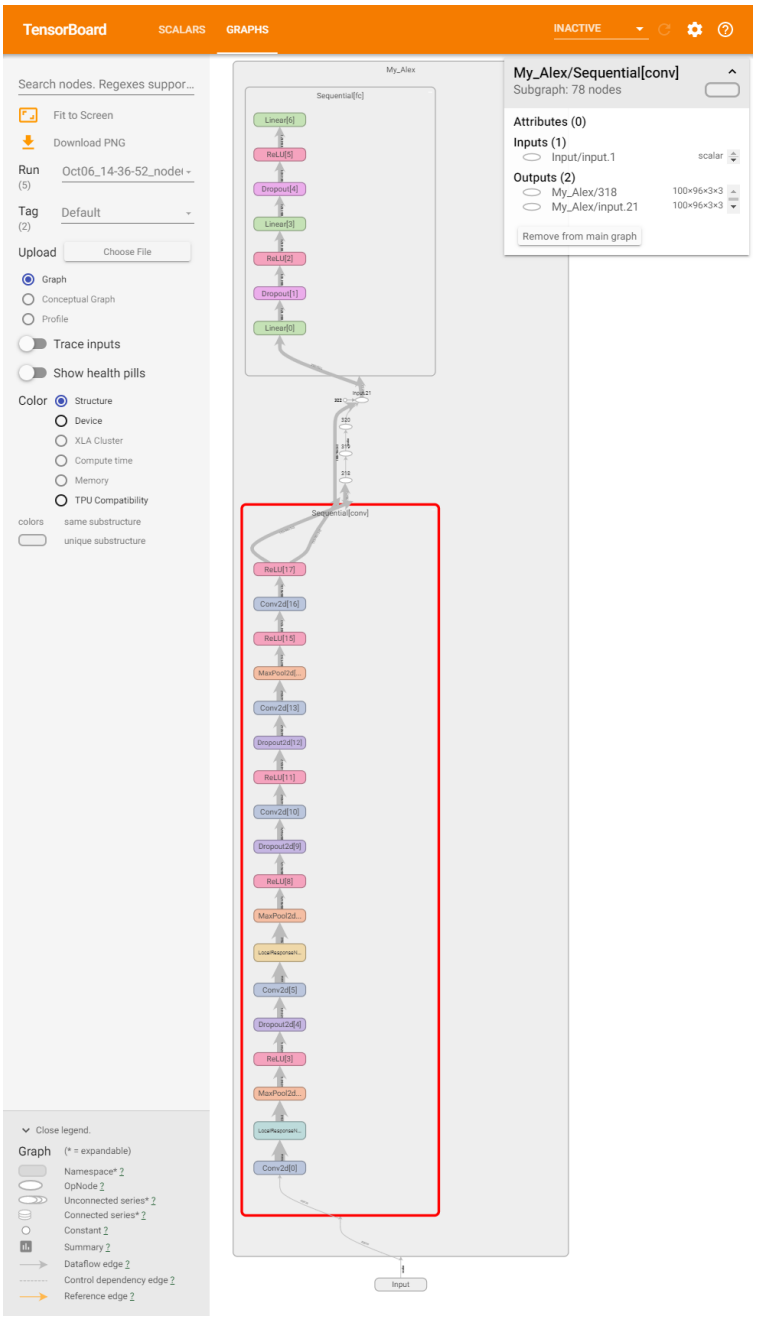

Figure 2: My first Alexnet

2.2 Simplify the two gpu structure

At first, I build a model nearly the same as the Alexnet, but not implement his tow gpu structure. (Another difference is that Alex's model dealt with the 227 * 227 image but cifar-10 is 32 * 32). See **Figure ??**.

And then I am just curious about the real effect of every layer – the LRN, the maxpool, the drop out and the kernel size of the CNN. So I do some changes to the model and compare them with my first model.

2.3 Training trick–variable learing rate

I use a trick in the training process – decrease the learing rate as the epoch number grows. Here I use 0.1 as the learing rate during the first 30 epochs, then 0.01, then 0.001, and finally 0.0005. It is really useful and the accuracy grows rappidly in every period!

2.4 Replace LRN with BN

The difference between LRN and BN is that LRN do normalization on different layers but BN on different samples in the same batch. And I find that LRN doesn't work as well as BN. See **Figure ??** , you can see that the network using BN gets a high accuracy on both train and test process but the network using lrn looks not so good.

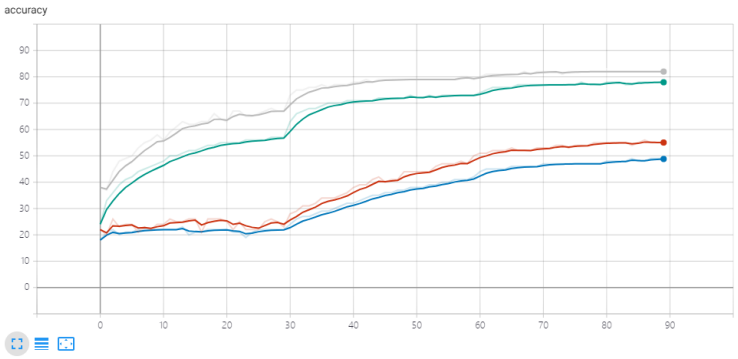

Figure 3: LRN VS BN

2.5 Drop out and Maxlpool

Then I build three models to test the affect of the drop out and the maxpool layer.

Table 1: Five Alexnets

	drop out	maxpool
alex1	yes	yes
alex2	no	yes
alex3	yes	no

The alex1 and alex2 use the same padding on conv2d which keeps the feature map size and use maxpool to decrease the feature map size, while the alex3 use conv2d to decrease the feature map size without the maxpool layer.

$$new\_size = (old\_size - kernel\_size + 2 * padding)/stride + 1 \quad (1)$$

As you can see in **Figure ??**, **Figure ??** and **Figure ??**, alex2(no drop out) 's loss decreases more quickly than the other two's and it gets higher accuracy in both train and test phase. But its train accuracy is 8% higher than its test accuracy, which means it gets a little overfitting. What an amazing result! Network without drop out work better that those with drop out? Then I realize that maybe I drop out too much. So I tune down the drop out probability to 0.2 - 0.4, and the result supports my conjecture! And I find that alex3(no maxpool) works as well as (even a little better than) alex1.

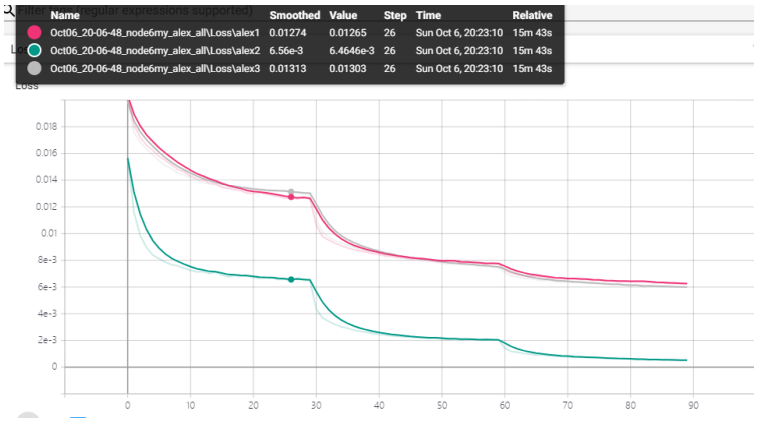

Figure 4: loss

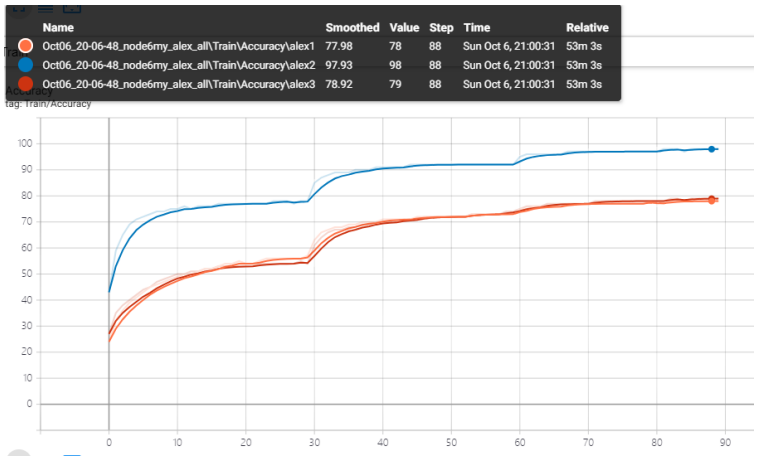

Figure 5: train accuracy

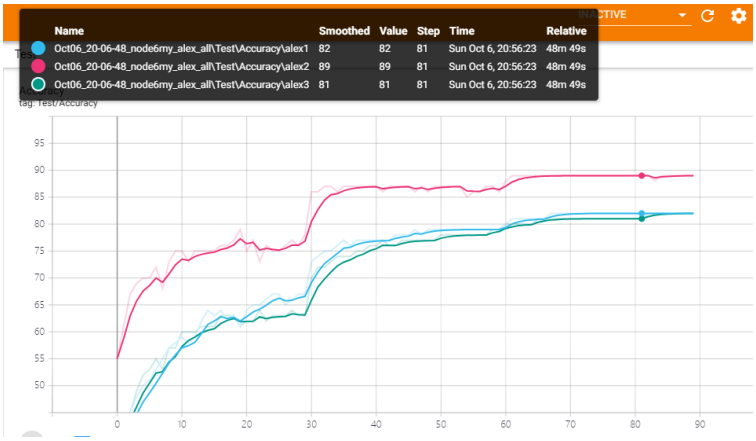

Figure 6: test accuracy

2.6 CNN kernel

I also do some changes to the CNN kernel, such as changing the kernel size, the kernel channel, the stride and the padding. And finaly I come to a conclusion that it doesn't matter a lot, as long as the kernel is not so strange, and it would be better to increase or decrease the channel no more than 2 times between different layers.

2.7 Conclusion

- Batch normalization works better than local response normalization.
- Drop out can avoid overfitting, but too much drop out will affect the performance of the model.
- Maxlpool is not necessary, but it is a good choice to reduce parameters and accelerate model training.
- Ultimately, I achieve my highest accuracy of 89% on the test dataset.

3 Pretrain ResNet-18

3.1 ResNet-18

Compared to Alexnet, ResNet-18 add a residual block to the CNN layer, which can catch the relation between different layer.

3.2 Test Result

Here I do some research on the pretrained model and un-pretrained model.

As you can see in **Figure ??** and **Figure ??**, the pretrained model converges more quickly and gets higher accuracy. Both the pretrained and un-pretrained model get a little overfitting, maybe that's because they don't use Drop Out.

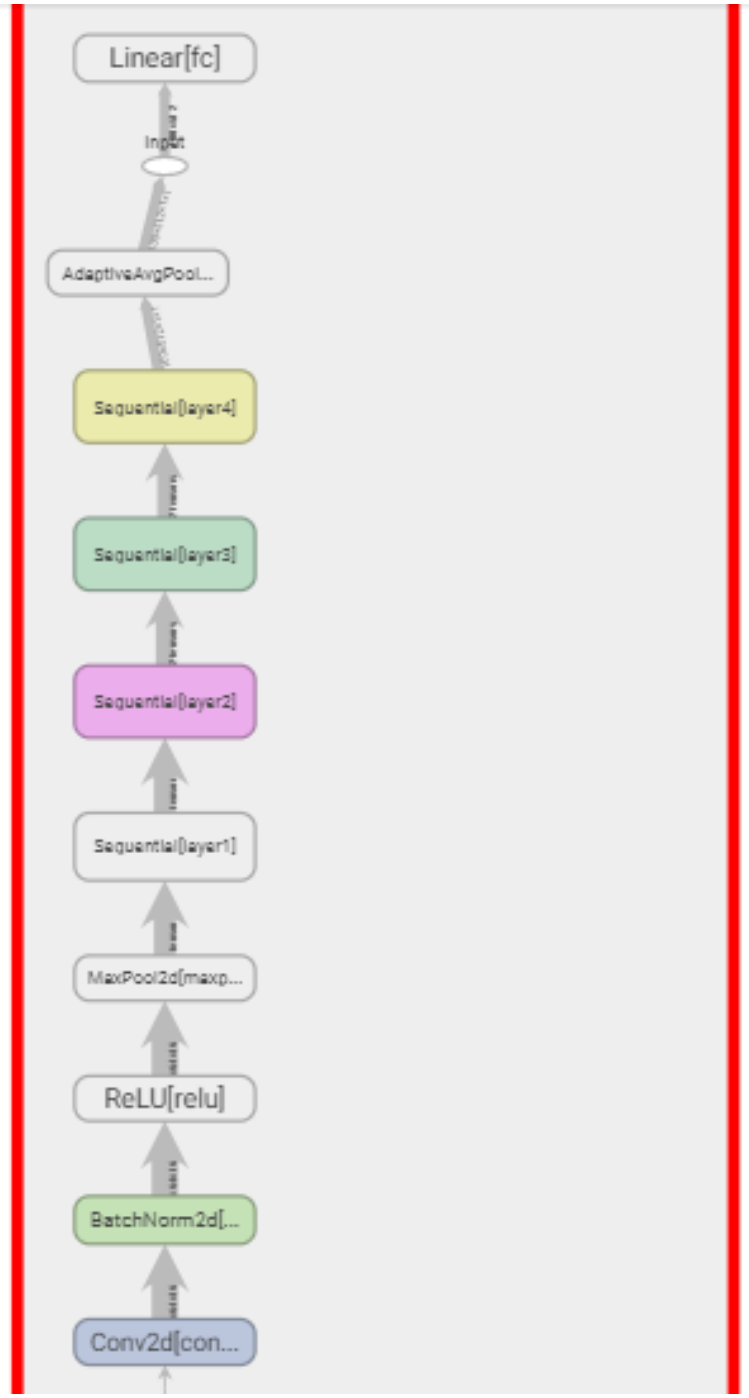

Figure 7: ResNet-18 Structure

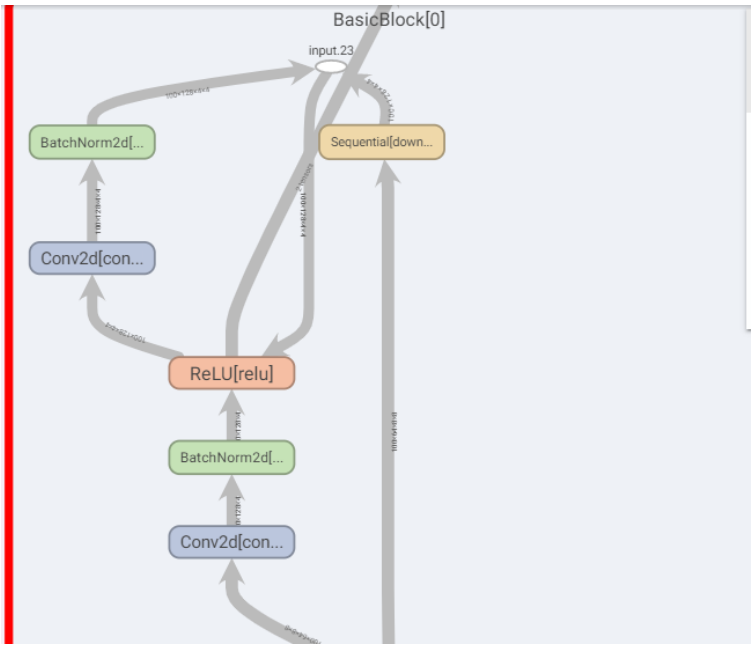

Figure 8: ResNet-18 Structure BasicBlock

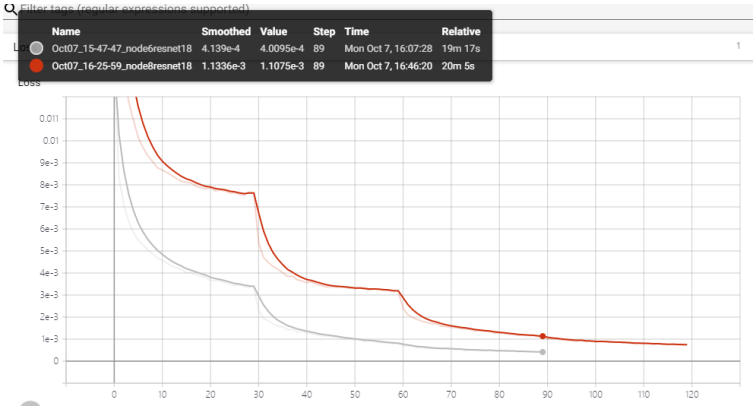

Figure 9: ResNet Loss

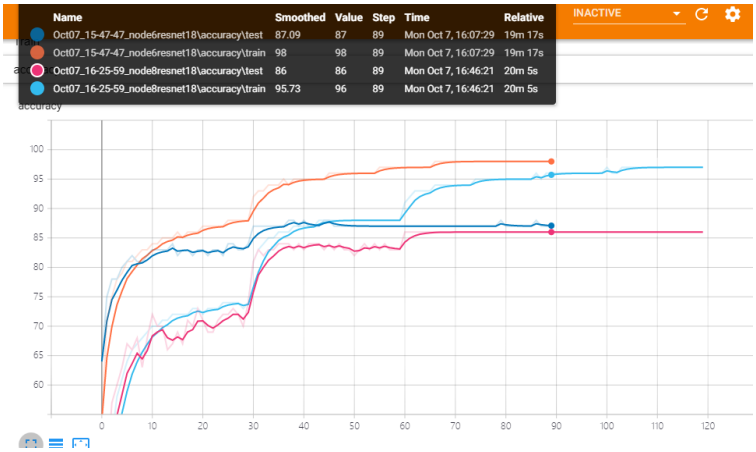

Figure 10: ResNet Accuracy

4 Summary

To draw a conclusion, both alexnet and resnet can work well on cifar dataset. By doing this mini-project, I find that doing research is sometimes very troublesome – to tune the parameters frequently and wait for its feedback for a long time training, and to compare a lot of structure to understand the specific layer's really effect; and sometimes is a little confusing and I don't know what to do because there are so many directions. But it's also very interesting to find out something, and just go ahead.