Weekly report March 10,2019

Rong Ding
dingrong@sjtu.edu.cn

Abstract

This weekly report is a complementary explanation about the previous report , and also records my progress on this task. The previous report can be found here: $https://github.com/Dingrong123/mini\_project\_on_cifar-10/blob/master/Weekly\%20Report\%20\%20Feburary\%2019\%2C2019\%2C\%20\%20Rong\%20Ding.pdf$

1 Answers for some specific questions

In this part I copy the question from email for readers' convenience. All the 'line #num' refer to the line in the previous report.

1. line 75, "It is because when epoch number is 40,60 or 80 , the learning rate will decrease by a half , which prevents loss to be stuck at a high value." **Can you give possible reasons for this situation?**

   Answer. Yes. Because when the learning rate is too high, the 'step' of gradient descent will be too large that it will not go towards the minimal points. However, the loss function may reach a higher value.

2. line 139, "In 1th , 51th , 101th and 151th epoch, the loss is large. But in 50th , 100th , 150th and 200th epoch, the loss is quite small and overfitting exists. So a conclusion is drawn that in each 50 epochs , the network becomes overfitting towards training data in this period." **Can you explain it in details, specifically, for the loss curve in Figure 2?**

   Answer. In this run I change training set by scale jittering for 3 times. So in 1th , 51th , 101th and 151th epoch, new data go into my network . However because I change the data every 50 epochs, which is long enough for my network to get a high performance by overfitting to the existing data. Since it is overfitting rather than really working well, when new data come, that is in 51th , 101th and 151th epoch, the loss becomes huge again. This run makes me to realize that I should update data more frequently in order to get a better performance.

3. **The results you reported in on val dataset (you use val_acc in the text) or testing dataset (you use $test\_acc$ in figure 6)?**

   Answer. Oh, I'm sorry to find myself make a mistake when drawing these figures. All the $test\_acc$ should be val_acc and they are all accuracies on validation dataset.

2 A total complementary explanation about my previous experiments on pretrained model

When I wrote the previous weekly report, I didn't know how to verify VGG16 pretrained by ImageNet to classify cifar-10 cause their fc layers are different. So I use my own fine-tuned model as pretrained models. Here I'd like to give a detailed description about what I do in the previous report.

1. First, I download my fine-tuned pretrained model. As we know that layers of VGG is divided by two parts, CNN layers for extracting features and Linear layers for classifying. I freeze the CNN layers, and then retrain these Linear layers for improvement. However, after 100 epoch, compared to the previous fine-tune model,the network's final accuracy didn't improved. I supposed it is because the CNN layers , identical to the pretrained model , which are not so perfect on their own, thus prevents the classifier layer to perform better.
2. Second, I try a modified approach. I download the fine-tuned pretrained model and retrain its CNN layers and Linear layers using different learning rates. After many trial I set the proper parameters for them. In the CNN layers. I use relatively small initial learning rate($3e-3$). In Linear layers that is bigger ($1e-2$). This approach works better than the former one, However it still can't go beyond over the existing best model I have. I think it because my training policy is not so proper.

2.1 Explanations for benefits of preprocessing data

I used two ways to preprocess data. First is random flip for data augmentation, which prevents overfitting. Second is normalization . This kind of technique make figures obey a overall distribution and become similar in means and variance , which helps loss to converge.

3 Model pretrained with ImageNet

This time I tried the VGG16 model pretrained with ImageNet, which I called $modelA$. Compared with that of cifar-10, the train speed is not improved. Even the best result(91.9%) is happen to be the same as that of cifar-10. However, their curse of validation set accuracy is different. The curse of $modelA$ keeps going higher and stirs less at the epoch near 200(the run lasts for 200 epochs). So I make a run of 300 epochs. However, the best result I get is only 91.3%. Here are the accuracy of validation dataset of epoch 200 and epoch 300.

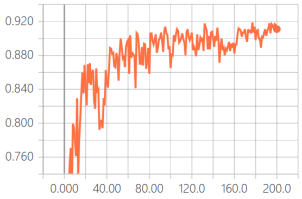

Figure 1: val_acc, epoch=200

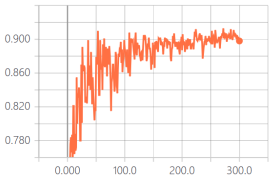

Figure 2: val_acc, rpoch=300

In this these two runs I have a better understanding of scale jittering.

- First, scale jittering itself is an approach with much uncertainty. For instance, although 300 epochs is more than 200 epochs, the later one actually performs over the former one.
- Second, by the figures we can see that the accuracy stirs acutely at first , then as the number of epoch increases, the curve becomes smoother and converges to some value. It's a common case for this approach.

4 Questions and Plans

As for plans,I still don't know is there any project waiting for me later. If any,I will finish it according to my time schedule.