Image Segmentation using Mean Shift Algorithm

Letian Chen, School of Psychological and Cognitive Sciences

1 Introduction

Nowadays, application of computer vision has fulfilled our life: smart photo gallery, camera face auto-detect, robot path planning, etc. The main method behind these application is usually deep neural networks (NN). Since 2011, NN has become the super star for many artificial intelligence applications due to its impressive performance, including the computer vision field. Therefore, in this class project we'll use neural network model to solve a scene classification task.

1.1 Task

Scene classification is the main task of the course homework, i.e. to detect the features of the image, and then recognize which scene is it. Specifically, this homework's database has 55999 training images in 80 classes, and the test set has 5000 images. The training images falling into each class is relatively balanced, with the minimum 212 images in class 55, and maximum 881 images in class 32. Most classes has image number between 500-700. This task has quite a bit difficulty that even people could not find the correct answer since the image is ambiguous. Moreover, the features of a scene is much harder to represent than an object.

1.2 Algorithm Review

The neural network architecture for scene classification is pretty similar with the object classification. As we all know, this area has the architecture from AlexNet, VGGNet to more complicated Inception and Resnet model. AlexNet [1] is the first modern neural network architecture successfully winning the IMAGENET 2011. After that, VGGNet [2] goes deeper to more than 10 layers and achieve better performance. VGGNet is also the first widely-used neural networks for multiple tasks. Google introduced Inception Net using more layers [3], but the core of it is that each layer is a little module and consists of several path of convolution and pooling. More importantly, Google kept updating Inception Model with Inception V2, V3 and the latest V4. On the other hand, **(author?)** introduced ResNet [4] with residual link between layers and achieve better performance with less computation burden. These four architectures are so famous in computer vision domain that almost all the tasks use them as their basic architecture.

1.3 This Report

In this report, we introduced the task and some classic algorithms in Section 1. Then, we will explain the method we use, the difficulty we face and the solution to tackle these problems in Section 2. After that, we will show our algorithm's result in validation set. Finally we will have a discussion in Section 4.

2 Method

We mainly used the Inception-Resnet-V2 Architecture in this course homework, with some slight tricks in the training period.

2.1 Inception-Resnet-V2

Inception-Resnet-V2 [5] is introduced by Google together with Inception-V4. The Inception-Resnet-V2 combined the Inception module (See Fig. 1) in Inception-V4 with the Residual link in Resnet. Because of the page restriction, I will not present the detailed Inception-Resnet-V2 here, but generally it is composed by many layers of residual Inception module.

2.2 Implementation

My code implementation is based on the code of https://github.com/puke3615/SceneClassify with tensorflow and keras. I mainly modified the training process and add validation set while leaving the data augmentation unchanged.

To use my code,

1. edit the "config.py" to adjust the path

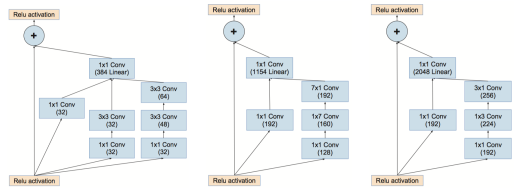

Figure 1. Inception

2. run "DataTransformer.py" to organize the Raw Data to adjust to the requirement by the training process. Specifically, "DataTransformer.py" will organize the images under a directory of their classes. For exmaple, a class 14 image "011586.jpg" will be put under a directory named 14. Thus, after transformation, we will have 80 directories with images inside. Moreover, "DataTransformer.py" also divide the training set into training set (4/5) and validation set (1/5). Each set has 80 directories.

3. run "ClassifierInceptionResnetV2.py" to train the network according to the training and the validation set configured in "config.py". It is worth mentioning that after epoch the model parameters will be automatically saved, and if the training is interrupted, you can just run "ClassifierInceptionResnetV2.py" again and it will restore the model parameters with best validation accuracy.

4. After training, run "predictor.py" to predict the test image classes, and the test image path is also configured in "config.py".

2.3 Difficulty and Solution

Although it seems easy, I encounter many difficulties during the implementation and debugging.

1. Fine-tune: since I used pretrained model (Imagenet pretrain), I need to fine-tune the parameters to adjust this problem. However, if I froze the last 50 layer parameters, I could only gain a training accuracy of about 50% and the validation accuracy is only 10%. It is an obvious under-training. Thus, I opened the limit to last 20 layers. In this way, I could gain a training accuracy of about 70% and the validation accuracy is about 50%. I want to unfreeze more layers, but my GTX 1080 card won't allow me since it has only 8G memory. Thus, I used another server with 4 TITAN X, whose memory is 12G each. I am able to unfreeze all the layers of the network to train on this new machine, and it ended up with much better results.

2. Learning-rate: learning rate is a very sensitive parameters. If it is set too large, the training loss won't go down anyway. But if it is set too small, the training loss decreases too slow. Thus, I used a slowly-decreased pattern of the learning rate. The initial training was on the learning rate of $1e-3$, and after 4 epoches, the training and validation accuracy stops increasing at 75% and 65%. Then I turned the learning rate to $1e-4$, and after 8 epoches, the training and validation accuracy stops at 85% and 75%. Finally I turned the learning rate to $1e-5$, and after another 1 epoch, the training and validation accuracy stops at 90% and 79%.

3. Architecture error: When reading keras's code for Inception-Resnet-V2 model, I found that although it is claimed as V2, the stem module uses the Inception-Resnet-V1 version (the Inception modules are correct). Maybe it is a bug in keras's code and I haven't had the time to fix it manually.

4. Image Preprocessing: Since GPU is busy with neural network computing, we could use CPU's multiple cores to do some preprocessing for the image, such as resize, image augmentation. This parallelism is critical, since the preprocessing time is almost exactly the same as a batch learning for neural network. Thus, if it is not parallelized, the execution time will double.

3 Result

Using the slowly-decreased learning rate and unfreezing all pretrained layers, I gained

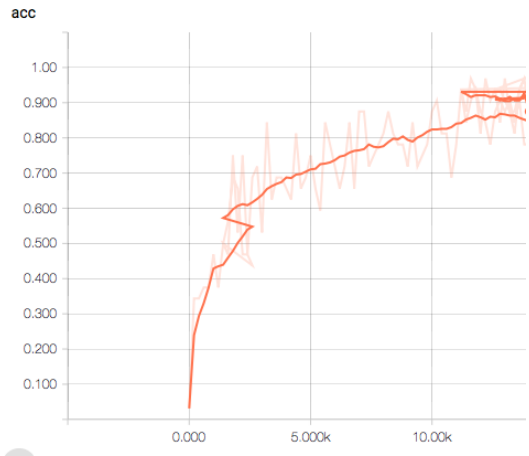

Figure 2. Learning Curve. There is several lines after 10k steps because the training of $1e-4$ learning rate continues, but I restore the params from 8th epoches with $1e-5$ learning rate to start another training. So, the bottom line is $1e-4$ learning rate, and the top line is $1e-5$ learning rate using the parameters from 8th epoches.

a training accuracy of 90.63% and validation accuracy of 79.05% 2. I also tried other training method and listed the performance in Table 1.

Table 1. Comparing result (lr = learning rate)

Training Method	Validate Acc
50 layers freeze, lr=1e-4	10.57%
20 layers freeze, lr=1e-4	56.16%
no freeze, lr=1e-4	75.12%
no freeze, lr=1e-3, 1e-4, 1e-5	79.05%

4 Discussion

In this project, we learned basic concepts of scene classification, read several CV-related papers, implemented our own version of scene classification, using the architecture of Inception-Resnet-V2. I also gained a lot of experiences fine-tuning a pretrained model: If the GPU allows, unfreeze layers as more as possible and take special caution with learning rate. So the rule of thumb for choosing learning rate is: Start with larger learning rate, if it doesn't work, decrease it, and kept trying.

It is worth mentioning that scene classification is much harder than object classification. I look at some test pictures and even I don't know which class does it belong. For example, for the class of "basketball gym", an image is a normal basketball gym with higher viewpoint, including the whole gym. Another image, however, is three people defending one attacking people in near viewpoint, and almost the whole image consists of four people. How can an algorithm classify these two kinds of images into one class? Therefore, in my opinion, scene classification may not share the same feature of object classification, and the best neural network architecture for scene classification may be different.

5 Reference

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, *et al.*, "Going deeper with convolutions," Cvpr, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[5] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning.," in *AAAI*, vol. 4, p. 12, 2017.