Week3 Report on Deeping Learning in Mobile

Hansen Wang

***Abstract*—This report is divided into two parts, the first part will show the detailed procedure of running an image recognition based on the caffe model trained by SqueezeNet; and the second part will make a comparison on the app performance between the two models(the CaffeNet one and the SqueezeNet One).**

***Keywords*—SqueezeNet, CaffeNet, Android, Deep Learning**

I. Task

1.Realize the Squeeze-based image recognition app on iOS/Android.

2.Make a comparison on the performance(such as the latency) between two apps which are based on different neural networks.

II. Realized the squeezed based app

The reason I choose to modify the Android App from the templates is, there are more resources on GitHub about android image recognition than iOS, so it will be better for me to generate an new SqueezeNet-Based Android app rather an iOS one. To rewrite the app, we need to figure out the outline of this project. The main purpose of this app, is to realize the image recognition function on a mobile device, which is the same function we can easily achieved on computers and servers. The reference I mainly use for this project is from a personal GitHub project page and the official GitHub page[1] of BVLC(Berkeley Vision and Learning Center).[2]

After reading the readme file and compare each file with the one from SqueezeNet Official site[3], I find it is easy to modify the template into a squeezenet-based app. Because the BVLC caffenet model is also trained by caffe, so another .caffemodel file (which was trained by squeezenet on ImageNet) will need no change. Also, the solver.prototxt and the train_val.prototxt file is already compatible with the caffemodel, so they also don't need change. Therefore, all we have to do is generate a deploy.prototxt file which is compatible with our 'SqueezeNet' model for the new app.

Based on the information from these sites[4][5], the procedure to generate a deploy file is quite easy. All you need is to rewrite the head and the tail of train.prototxt file according template file.(Here I mean the deploy file from [1], and you can see my file from uploaded support material folder.).

After this procedure, I change all the corresponding files' name to be the same as their corresponding template files to avoid unnecessary errors. By the way, I use the official Android Developing Tool, Android Studio Mac Version, and a Nexus6 with Android6.0 Emulator and a SamsungS6 with Android6.0.1 for tests.

Follow the instruction from the readme file from[1], to replace the templates files with the corresponding modified files with the same name. And click run, you will run them successfully.

III. Comparison on the two apps performance

A. Model Comparison

The templates are using caffemodel trained by the BVLC(Berkeley Vision and Learning Center) Group with CaffeNet, and the provided caffemodel is trained by with somebody in Stanford with SqueezeNet. The Basic information of the two models are listed below:

TABLE I. Comparison between models

	Training Resource	Neural Network	Size of model	Number of iterations	Top-1 Accuracy	Top-5 Accuracy
Template Model	ImageNet	CaffeNet (an improved version of AlexNet)[2]	243.9MB	310000	57.4%	80.4%
Squeeze Model	ImageNet	SqueezeNet	4.8MB	800000	61.6%	83.4%

From the table above, we can say that, with the same level accuracy, the model trained by squeezenet is almost 50x smaller than the CaffeNet(AlexNet). Since the model must be stored in the storage space, the more smaller of the model, means it will be more efficient for the device the read data which might leads to a faster feedback.

B. Test Process

I randomy picked up 5 groups of photos, and each groups contains 100 photos. After downloading and taking a look at my test photo from here[6], you will see these 500 pictures are divided into 5 categories: Bus, Dinosaur, Elephant, Flower and Horse. The main three parameters I set to evaluate the app is the **accuarcy**, the real time from clicking button to printing results(which is denoted as **'elapsed wall time'** in the log file) and the CPU time(which is denoted as **forwarding time** in the log time[7]).

Inspired by the top5 accuracy evaluation, I note down every description of each photos, and set points for these descriptions. If the first phrase reaches the answer, this test will achieve full mark 1.0, otherwise, the second phrase is right will receive 0.6 points, the third is 0.4, fourth is 0.2, fifth is 0.1, else is 0.

C. Result

The comparison of two apps is shown below:

Hansen is a visiting undergraduate student with the Department of Electrical and Computer Engineering, UC Berkeley, Berkeley, CA, 94702 USA, $hansen\_wang@berkeley.edu$

And he is also an undergraduate student with KuangYaming Honors School, Nanjing University, Nanjing, 210047 China, $141242081@smail.nju.edu.cn$

TABLE II. THE POINTS SYSTEM OF ACCUARCY

	1st description is right	2nd begin to be right	3rd begin to be right	4th begin to be right	5th begin to be right
Points	1.0	0.6	0.4	0.2	0.1

TABLE III. CAFFENET-BASED APP

	Bus	Dinosaur	Elephant	Flower	Horse
Elapsed Wall Time/ms	922.8	926.4	896.2	925.4	961.0
Forwarding Time/ms	4803.5	5076.8	4705.9	4890.9	4964.6
Accuracy/%	82.2	0	88.0	0	20.0

TABLE IV. SQUEEZENET-BASED APP

	Bus	Dinosaur	Elephant	Flower	Horse
Elapsed Wall Time/ms	948.88	919.7	930.4	N/A	N/A
Forwarding Time/ms	3974.9	3864.6	3881.3	N/A	N/A
Accuracy/%	83.0	0	90.0	N/A	N/A

IV. SHORTAGES

As you might see from my support file named 'Train Results.xls' which noted the down each set record during test, there might not be enough data for the SqueezeNet-Based App. Because this method of modifying app is not professional, so when run the programme on SamsungS6, the console keeps popping up thousands of signals, and the biggest storage for the log data in the consule is 1MB. And my method to collect the time is to use grep command on the log file after i finish all the testing, so for this method, I may not be able to collect all the time data because of thousands of 'Junk Message' popping from the console.

However, the weird thing is, the console stops popping up Junk Message when I am trying to run the system on the emulator NEXUS6 later.

V. SUMMARY

So in general, we can see that there is not much difference between the performance of the 2 apps, however, the SqueezeNet-Based Net app only requires a '50x-smaller' storage than the previous one, which is more competitive from my perspective.

ACKNOWLEDGMENT

The authors would like to thank Chen Yiding for discussion on this task.

REFERENCES

[1] https://github.com/sh1r0/caffe-android-demo

[2] https://github.com/BVLC/caffe/tree/master/models/bvlc_reference_caffenet

[3] https://github.com/DeepScale/SqueezeNet

[4] https://github.com/BVLC/caffe/wiki/Using-a-Trained-Network:-Deploy

[5] http://www.cnblogs.com/denny402/p/5137534.html

[6] https://1drv.ms/u/s!Ag6LSOut6wk5gboCHLATXlEFT-bZzA

[7] https://github.com/sh1r0/caffe-android-lib/issues/27