Robotic Inference Application on Flower Recognition using GoogLeNet

Peng Xu

Abstract—The project is an attempt to practice deep learning on Nvidia DIGITS and prepare for robotic inference applications. A supplied dataset is given as a benchmark to test the pipeline of training and evaluating a model. The GoogLeNet network is applied and has achieved 75% accuracy and 10 ms inference speed which satisfied the project requirements.

Index Terms—Robotic Inference, Deep Learning, Image Classification.

1 Introduction

THE project aims to apply deep learning methods on robotic inference applications. Imagine there is a garden where several kinds of flowers are planted. The task for the robot is to recognize the flower species. Deep Learning is used to have an end-to-end learning and a state-of-art deep convolutional network model, GoogLeNet is used to train this model.

The training and evaluation process could be greatly simplified by Nvidia DIGITS platform. With only filling the hyperparameters, the data preprocessing and model building could be done using Cloud Computing.

Hopefully, the trained model could be directly applied on a Jetson TX1/TX2.

2 Background

Deep learning has absolutely dominated computer vision over the last few years, achieving top scores on many tasks and their related competitions. The most popular and well known of these computer vision competitions is ImageNet [1]. The ImageNet competition tasks researchers with creating a model that most accurately classifies the given images in the dataset.

Over the past few years, deep learning techniques have enabled rapid progress in this competition, even surpassing human performance.

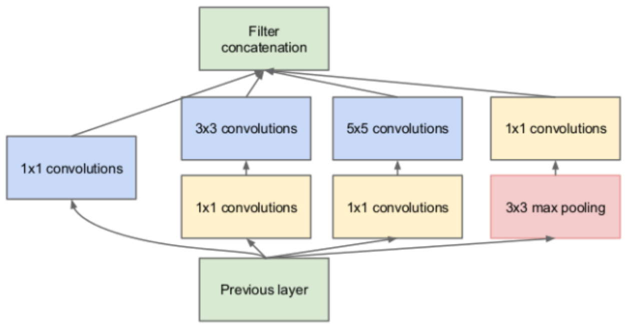


Fig. 1. Inception Module from GoogLeNet.

The GoogLeNet [2] architecture was the first to really address the issue of computational resources along with multi-scale processing in the paper Going Deeper with Convolutions. As we keep making our classification networks deeper and deeper, we get to a point where were using up a lot of memory. Additionally, different computational filter sizes have been proposed in the past: from 1x1 to 11x11; how do you decide which one? The inception module and GoogLeNet tackles all of these problems with the following contributions:

Through the use of 1x1 convolutions before each 3x3 and 5x5, the inception module reduces the number of feature maps passed through each layer, thus reducing computations and memory consumption!

The inception module has 1x1, 3x3, and 5x5 convolutions all in parallel. The idea behind this was to let the network decide, through training what information would be learned and used. It also allows for multi-scale processing: the model can recover both local features via smaller convolutions and high abstracted features with larger convolutions.

GoogLeNet was one of the first models that introduced the idea that CNN layers didnt always have to be stacked up sequentially. The authors of the paper showed that you can also increase network width for better performance and not just depth.

3 Data Acquisition

3.1 Supplied Dataset

Supplied Data from Udacity was having 3 classes (Bottle, Candy Box, Nothing).

TABLE 1
Supplied Data

Bottle	3426	256x256x3
Candy Box	1871	256x256x3
Nothing	2273	256x256x3
Total	7570	

In Fig. 2 some example images are shown as examples picked from the supplied dataset.

(a) Bottle

(b) Candy Box

(c) Nothing

Fig. 2. Example data in Supplied dataset.

(a) Daisy

(b) Rose

(c) Sun-Flower

(d) Tulip

Fig. 3. Example data in Collected dataset.

3.2 Collected Dataset

The dataset, Flowers, contains 4242 images of flowers. The data collection is based on the data flicr, google images, yandex images. The datastet to recognize 5 classes of flowers:: chamomile, tulip, rose, sunflower, dandelion. For each class there are about 800 Images. Images are not high resolution, about 320x240 pixels. Images are not reduced to a single size, they have different proportions.

In Fig. 3 some example images are shown as examples picked from the collected dataset.

4 Results

4.1 On Supplied Dataset

GoogLeNet trained on the supplied dataset converged quickly within 5 epochs. The training accuracy and validation accuracy were almost the same and around 100%.

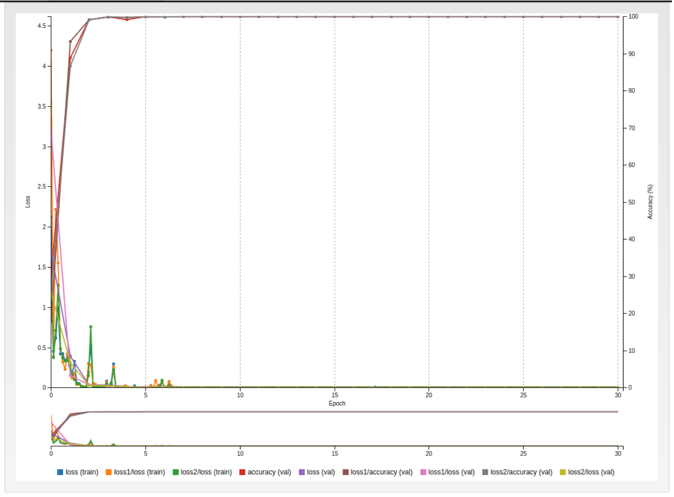
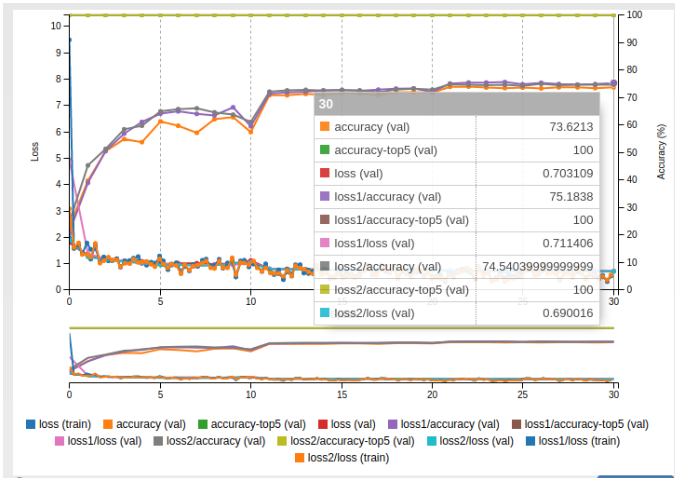


Fig. 4. Training loss and accuracy variance curve on supplied data.

4.2 On Collected Dataset

GoogLeNet trained on the collected dataset converged at around 12th epochs. The training accuracy and validation accuracy were almost the same but only reached 73%.



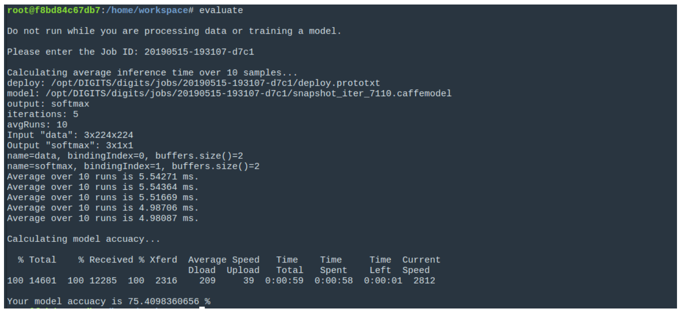
30	
accuracy (val)	73.6213
accuracy-top5 (val)	100
loss (val)	0.703109
loss1/accuracy (val)	75.1838
loss1/accuracy-top5 (val)	100
loss1/loss (val)	0.711406
loss2/accuracy (val)	74.54039999999999
loss2/accuracy-top5 (val)	100
loss2/loss (val)	0.690016

Fig. 5. Training loss and accuracy variance curve on collected data.

5 Discussion

The provided evaluate function could be used on the supplied dataset. After 30 epochs' training, the trained model could reach as high as 75.40% test accuracy on an unseen test dataset. The inference time were a little higher than 5 ms which was within 10 ms which meets the requirement. Both results are shown in Fig. 6.

The test accuracy was much lower than the training accuracy which could happen for several reasons. Firstly, the dataset is small and the training is likely to overfit. The training accuracy is as high as around 100% but it don't mean the similar performance when it comes to an unseen image. The issue could be addressed by data augmenting. Secondly, the model is over complicated for the data which also could lead to the overfit problem. One solution could be introducing more dropout layers. But in general, the performance is good enough for the robotic inference application.



```
root@f8bd84c67db7:/home/workspace# evaluate

Do not run while you are processing data or training a model.

Please enter the Job ID: 20190515-193107-d7c1

Calculating average inference time over 10 samples...
deploy: /opt/DIGITS/digits/jobs/20190515-193107-d7c1/deploy.prototxt
model: /opt/DIGITS/digits/jobs/20190515-193107-d7c1/snapshot_iter_7110.caffemodel
output: softmax
iterations: 5
avgRuns: 10
Input "data": 3x224x224
Output "softmax": 3x1x1
name=data, bindingIndex=0, buffers.size()=2
name=softmax, bindingIndex=1, buffers.size()=2
Average over 10 runs is 5.54271 ms.
Average over 10 runs is 5.54364 ms.
Average over 10 runs is 5.51669 ms.
Average over 10 runs is 4.98706 ms.
Average over 10 runs is 4.98087 ms.

Calculating model accuacy...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14601  100 12285  100  2316    209     39  0:00:59  0:00:58  0:00:01  2812

Your model accuacy is 75.4098360656 %
```

Fig. 6. Model evaluation after training.

6 Future work

In this project, Nvidia DIGITS as Deep Learning platform was used to train models for image classification tasks. Firstly, a supplied dataset with 3 classes was trained with a GoogLeNet network and achieved 75.41% evaluation accuracy and under 10 ms inference speed. While with collected dataset, Flowers, the model trained based on GoogLeNet achieved 73% training accuracy.

One direct improvement for training is to purify the collected data. The images were randomly captured from internet containing a piece of garbage data which might decrease the prediction or recognition level.

The test on the embeded hardware, namely Jetson TX1/TX2, is expected to be done when it is available. Then inference speed and real time recognition could be presented.

References

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[2] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, June 2015.