Robotic Inference

Xueyang Kang

Abstract—The two different neural networks are trained for classification, one is provided with the data containing three classes, bottle, candy, and nothing; the other is trained on the self collected data, including four classes, shoes, trousers, overcoat, the empty showcase in store. The data acquisition method for the latter model training is also introduced, the results of the two different classification are presented at the end of the paper, along with the discussion of reasons leading to the performance, along with the possible applications of these classification models and further needed work.

Index Terms—Robot, inference, class, training, classification, data acquisition.

1 Introduction

ROBOTIC inference has always been an important issue, to facilitate the robot to interact with the environment. Firstly the robot should have an intelligent sensing system, being ale to detect and classify the interests in its view. Traditional methods to solve these problems, is try to extract some robust features from the environment, and use these distinctive features to help the robot identify the object. But these features commonly are derived from the shape and color information of objects, and susceptible to the illumination and other external factors. So the uncertainty makes that the traditional method can not perform complex inference under some harsh constraints. Instead, the alternative solution to this problem, neural network is introduced in the paper, and its application possibility to solve these problems is also explored, because of the current mature techniques in regards to both data and model structure,the neural network can complete the inference more precisely, even superior over the traditional approaches for some purposes. One application example is that the training model can be applied to assistant the clothing shopping staff, to manage the product inventory in the store, the robot integrating the trained model can perform inference to count the item on the showcase and make a statistical analysis on that. Some optimization work pertaining to tensorRT about the inference deployment can be found in [1].

2 Background / Formulation

”AlexNet” is used for both training processes, with some common parameter settings as below,

- ”epoch” is set to 5.
- ”learning rate” 0.01.
- ”batch size” is 1.
- ”Validation data proportion” is 0.25.

the ”AlexaNet” is originally applied to classify the handwritten numbers, so it can be expanded to the classification task such as ”selection of candy on the conveyor”, the targets include only three examples. Through test, the accuracy can reach almost 100% after 3 to 5 epochs, and the batch size is kept with the default number, so that the network can learn more features in each image carefully. As for the data, one quarter of the images are used for training, while the rest used for validation. ”GoogleLeNet” is deeper and it has the inception functionality, so the coarse and refined information can be combined together to extract more unique features have both local and global patterns. In our case, the provided data include only several thousand images per class, this order of magnitude can be done with ”AlexNet” to achieve a good result, instead of use of deeper and wider net like ”GoogleLeNet”. The personal collected data is less than the provided one, so there is no need to use more complex network, just stick to the same model structure is considerable.

3 Data Acquisition

The data collected for my personal designed application is mainly collected both through online search and cellphone, the original acquired pictures are color images. There are three types of commodities in a clothing shop. The sample figures are displayed below. Here the empty store photos are separated as another class.

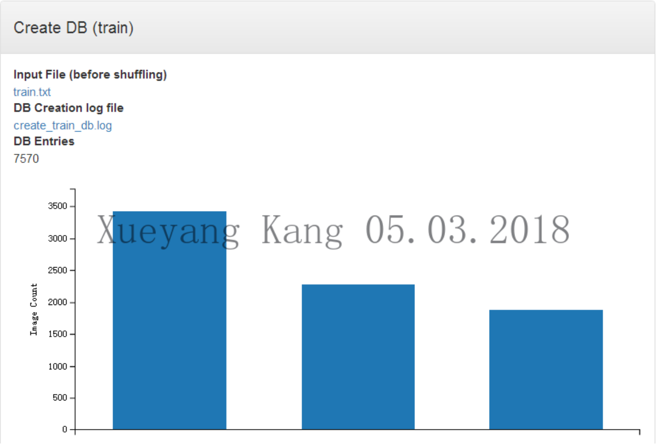
Fig. 1: Shoes pair

Fig. 2: Trousers

Fig. 3: Overcoat

Fig. 4: Empty store

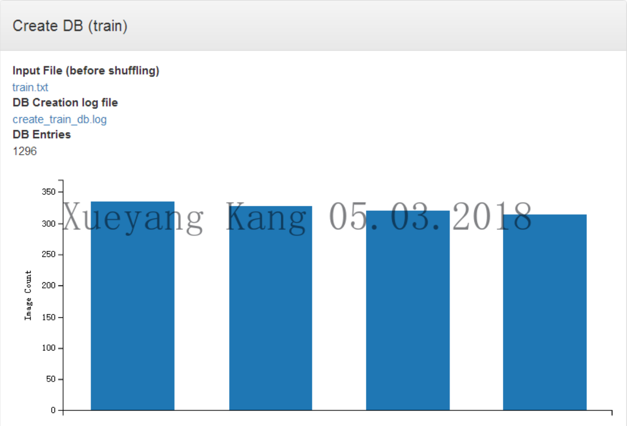
For the commodity class, the original shoes and coat picture collected online or offline are about 60, the photos taken from real commodity can only provide limited number of samples, although the same commodity can be seen at different viewpoints or under different illumination in real world, therefore, the valid number of data can be extended to 6 times of the original with the manual operation on the images. The original 60 images taken online can only be transformed by a software script, the flipped operation and the rotation, scale, brightness adjustment and shift method can be applied onto the original pictures to generate the enough samples for training. Obviously, the complexity of processing is also not negligible. The lat class corresponds to the empty store scenario, and the pictures are taken from real world, an unopened store, the showcase and the cabinet photos are taken at multiple viewpoints. Here continuous capturing mode in camera along fast movement was adopted in acquisition process to speed up sampling process. After all rgb images were collected, the pictures were resized to 256×256 pixels uniformly to meet the model input.


Create DB (train)

Input File (before shuffling)
train.txt
DB Creation log file
create_train_db.log
DB Entries
7570

Fig. 5: Samples distribution

The above is the three classes distribution in training dataset for conveyor sort.


Create DB (train)

Input File (before shuffling)
train.txt
DB Creation log file
create_train_db.log
DB Entries
1296

Fig. 6: Samples distribution

In 6, the collected data on my own is comprising of four types, and each class contains almost identical number of samples.

4 Results

Here "AlexNet" is trained with the provided data. The training result of the model is shown below.

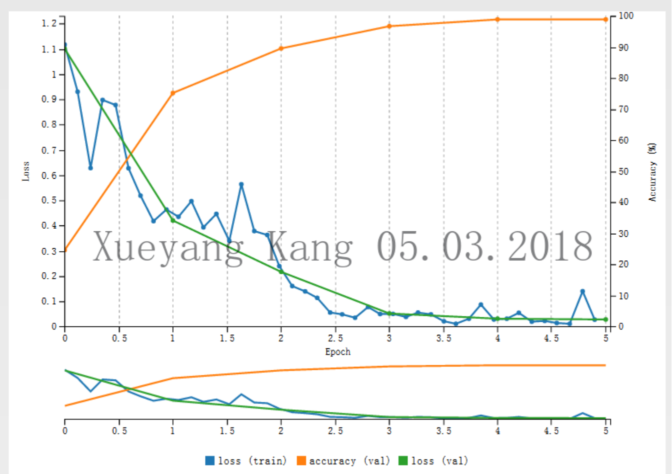
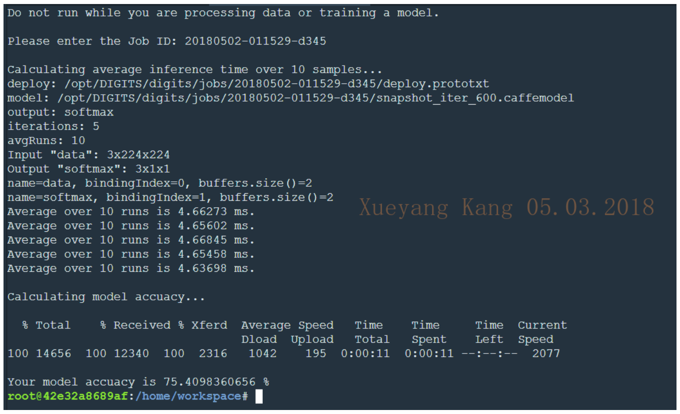


Fig. 7: Plotting of accuracy and loss

The accuracy result reaches almost 100%. Both training loss and validation loss reduce remarkably in the end. The 5 is the evaluation result of accuracy, and inference time of the trained model on provided data, their result meets the criterion well.


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

Please enter the Job ID: 20180502-011529-d345

Calculating average inference time over 10 samples...
deploy: /opt/DIGITS/digits/jobs/20180502-011529-d345/deploy.prototxt
model: /opt/DIGITS/digits/jobs/20180502-011529-d345/snapshot_iter_600.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 4.66273 ms.
Average over 10 runs is 4.65602 ms.
Average over 10 runs is 4.66845 ms.
Average over 10 runs is 4.65458 ms.
Average over 10 runs is 4.63698 ms.

Calculating model accuacy...

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 14656  100 12340  100  2316   1042    195  0:00:11  0:00:11 --:--:--  2077

Your model accuacy is 75.4098360656 %
root@42e32a8689af:/home/workspace#
```

Fig. 8: Evaluation result

As for the model trained on personal acquired data, the training process reached an accuracy around 80%, this may be resulted by the insufficient sample number or the homogeneity of features in images.

The evaluation script can't be used this time. So only the individual image is tested with the trained structure manually, and some unseen images, like T-shirt, or other images are misclassified as wrong thing, the probability of classification failure is very big.

5 Discussion

The trained model for the former one can still perform inference well, but as for the latter, the performance is very

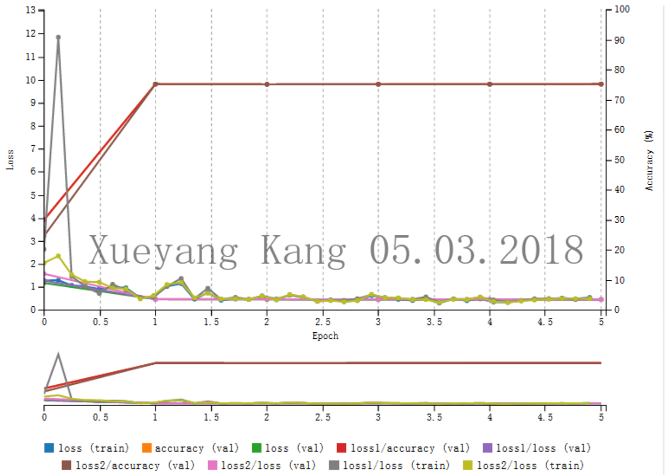


Fig. 9: Plotting of accuracy and loss

TABLE 1: Tests with trained model

Sandal	wrong
T shirt	wrong
Boot	wrong
Short pants	wrong
nothing	correct

awful.the root may reside in the uniformity of samples, because of constraints in reality and hardware, the data volume is expanded with the manually processing way, and each class in reality includes many subclasses, but the training data doesn't take all of them into account. The complex information like some shoes may be colored with a lot of colorful stripes, instead of the monotonous coloration on the appearance. Another concern is the placement of the shoes, in reality, each shoe may be placed randomly, instead of the formal layout of a pair. The training model may consider a pair of shoes together, so that the individual shoe may not be recognized, although this is not tested in my experiment, but is should have some negative impact on the classification. Last but not the least, if the size of samples can be further increased, maybe the 10 orders of magnitude could be enough to generate a good result for ”AlexNet” model.

The deployment on hardware is possible, when the inference time can be implemented in less than 10 milliseconds, then the normal 60fps of capturing rate can be applicable. The computational time is proportional to the operation time, the newly released ”Jetpack” can realize the very fast inference for a complex network in this case, but to the accuracy, the performance may have a subtle discount, because of the hardware constraints and other factors in real environment, like the illumination variant and scale distortion, so there is still a long way to go from the true application. So this idea may be better considered as a prototype, some further improvement work in terms of network structure is necessary.

6 Conclusion / Future work

The trained model performance for first conveyor task is quite acceptable, however, it is a little disappointing that the second model can not implement accurate inference. The prototype still has a lot of shortcomings. To make a mature commercial product, for this store assistant, the robustness and accuracy is an priority in product design. In real world, apart from a variety of clothing types and styles, the obstruction, i.e., some customer may not put back the product to the original place, even randomly place it somewhere, messed with other things, this will definitely lead to some errors during inference. So some further work can be carried on the expansion of samples, and refining the network structure, maybe some configuration parameters within the layers should be changed. Lastly the optimization work is put on the connection between hardware and network, the application input captured from a live camera needs to be calibrated and resized to the input size of the network, and the output of the network should also be converted to the application's expected output, in this case, the cumulative count should be incrementally added as the camera moving around manually or mounted onto some mobile robot. The report list of the inventory can then help the staff to be aware of the best on-sale and the worst, in order to adjust their business strategy accordingly.

References

[1] R. O. By Allison Gray, Chris Gottbrath and S. Prasanna, “Deploying deep neural networks with nvidia tensorrt.” https://devblogs.nvidia.com/deploying-deep-learning-nvidia-tensorrt/, April, 04 2017.