Final Project: Cdiscount product image data mining
Final report

Haijin He

Introduction:

From Kaggle competition: https://www.kaggle.com/c/cdiscount-image-classification-challenge

The data is from Cdiscount, France's largest online non-food retailer.

As the number of products Cdiscount sell is millions and still rapidly growing, it's a very difficult task to assign them to correct categories manually. Right now, they've already applying machine learning algorithms from text descriptions of the product to predict its category. But it seems the current method is reaching its limits. And in this challenge, they want to try learning from the images associated with the product.

The data contains information of its products, with product id, category, and image information.

Data description:

training.bson. the training data. Size: 59.2 GB. It contains a list of 7,069,896 dictionaries, one per product. Each dictionary contains a product id (key: _id), the category id of the product (key: category_id), and between 1-4 images. Most products have only one image per product. A product has an average of about 1.3 images. The images are of the size: 180 X180 X 3.

The whole Data format is for example:

{id: 11, category: 100, imgs:[{picture: b'xxxx'}, {picture: b'xxxx'},{picture: b'xxxxx'}] }

test.bson the testing data. size: 14.5 GB. It contains 1,768,182 products. Same format as train.bson, except without the category id.

Category_names.csv: contains the 3-level name for each category id. Each category id corresponds to a 3-level category name. Total number of categories is 5271. Number of level1 category is about 50 and number of level2 category is about 350.

Data storage:

The training and testing data are stored in BSON(Binary JSON) format.

To read it, use bson module included in pymongo module. Image are then transformed into numpy.ndarray format

Evaluation metrics:

Popular Image classification challenge ImageNet use top-1 or top-5 accuracy to measure performance. Top 5 accuracy means the model produces 5 most probable predictions, and if one of them matches the label, it is considered correct. State of art ImageNet models can produce a top-5 accuracy to > 96%, which is even better than human eyes, while the top-1 accuracy is still less than 80%. Here the Cdiscount challenge only uses top-1 accuracy.

Part1: classification.

As stated in the Kaggle competition, object 1 is to predict the category id for the test data set.

Since Deep learning is the state of art method for image classification, I plan to use Tensorflow deep learning library and Python. The Keras library turns out to be a more user-friendly library for new users. It is a higher-level library and uses Tensorflow as backend.

While I planned to also use image annotation method to work on the classification problem, I realized that the deep learning is already challenging. So the project did not use image annotation.

After a little reading into image classification with deep learning, it become clear that the Cdiscount is very similar to the famous ImageNet Large Scale Visual Recognition Challenge(ILSVRC). Comparison with ILSVRC Challenge:

1. All have huge number of images. ImageNet has over 1,500 million images. The ILSVRC competition hold every year has 1.2 million images, and about 1000 labels. Cdiscount has around 15 million, much less than ImageNet, but more than the ILSVRC competition. The labels for Cdiscount is around 5000.
2. Resolution. The average image resolution on ImageNet is bigger, but it is common to crop the original image to 256*256 for better speed. Cdiscount images are 180*180. So it is comparable in resolution.
3. Content. The images in ImageNet challenge is usually more complex, often with more objects on the same image, and diverse background. While the Cdiscount images are simpler, usually single object, no background. So if otherwise the same, the Cdiscount challenge might get better Top-1 Accuracy.

So from the comparison, the Cdiscount image classification is very similar to ILSVRC competition, but bigger in image size.

A somewhat different aspect is that Cdiscount has multiple image for one product, but as I checked, it is not frequent. Average pic/product is 1.3. So it might not have a major impact in how we deal with the challenge.

Given the high similarity between Cdiscount and the ImageNet competition, it's natural to consider the several well-known deep learning network structures and their performance in ILSVRC competitions.

Network	Top-5 Accuracy (%)	Top1-accuracy (%)	depth
AlexNet	83.7	57.0	8
VGG16 Net	90.0	70.5	19
Inception Net v3	94.4	78.8	22
ResNet	96.4	77.6	152
Xception	94.5	79.0	126

Table1: CNN networks and ILSVRC performance

Considerations and Implementation:

1. Running Image classification on large datasets requires huge amounts of GPU power. In this project, I used Google Cloud platform, which gives 300 dollars of credit for new subscribers. The Google platform has a Nvidia K80 GPU with 11GB of memory, which is relatively slow compare with current gaming GPUs like GTX 1080.
2. The running time of one epoch of data through a VGG16 network with resolution 180*180*3 on a Nvidia K80 GPU takes about 100 hours. That is too long for me, so I reduced the resolution to 90*90*3. As a result, the running time for one epoch is reduced to around 24 hours. But I imagine the accuracy will suffer to some extent.
3. To get relatively quick results from model and parameter settings like learning rate, I make the training resumable. The procedure is that I run the training for one epoch, the weights for the model will be saved. And then I run a prediction on the saved model. Then I make changes to the parameters, and load the saved weights and resume the training again.
4. Another trick I used to accelerate training is use pretrained network. Pretrained network on large scale image sets like ImageNet already captures a lot of higher level information on how to process images. It's a lot faster to use them as a starting point for your own network. As shown from my results, pretrained network can achieve an accuracy of 0.45 on first epoch of training. While in latter epochs it grows at a much slower speed.
5. Model selection. In this project, I tried pretrained VGG16 and Xception network. VGG network structure is simple, but it has much bigger parameter size compare to Xception. When using Xception, I also freeze the network weights and only train on the final fully connected layer. The running time for one epoch for Xception in this setting is fast, only takes around 8 hours. However, the accuracy struggles to grow compare with VGG16. I finally focused on VGG16 network.

Results:

The original learning rate is set to be 0.00001. And the submission gets a score of 0.45 in Kaggle's public leaderboard. Same learning rate for another epoch, the score is 0.48. I then increased the learning rate to 0.0001, and the third epoch score is 0.51. I then added a momentum 0.8 for the learning rate. And after forth epoch, the score is 0.55. Same learning rate and the fifth epoch get a score of 0.58. This is my current best score and stands at top 41% of all teams.

So far, the VGG16 network for the Cdiscount seems has not reached its best score yet given the trajectory. I am sure with more training epochs, the score will continue to grow. But due to time and resources, I have to stop here.

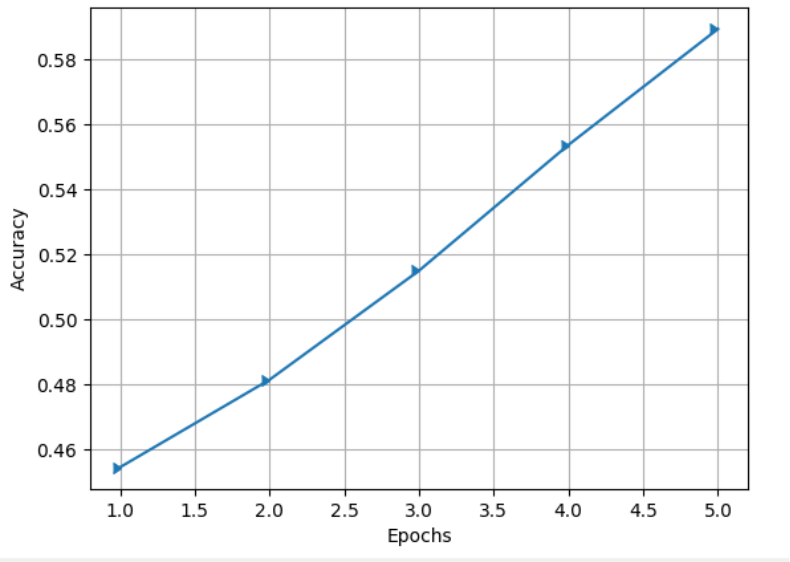

Fig 1: my accuracy score for each epoch.

Fig 2: my best submission score in Kaggle public leaderboard.

Part2: Image clustering

Data set contains 3 level of categories. For some categories, I try to use clustering algorithm to separate them into sub-categories. The result can be compared with the given category label to see if anything interesting is given.

We know that clustering algorithms usually requires distance calculation between samples. But here for images, how can we calculate distance between 2 images? A Euclidean distance between pixels of 2 images would not mean too much and it's costly to compute.

Image embedding based on Deep Learning neural networks can help here. The basic idea is that pretrained networks, in their penultimate layer (or other layers) captures high level of the input image. So the output that layer can be used as a representation of that image. Orange3 has an image embedding widget, from which you can pick which pretrained network to be applied for the algorithm.

For experiment, I picked a leve2 category named “JOUET”, which means “toys” in English, and run through the training data images matches the 12 level3 categories that belongs to JOUET. A total of 14835 images are collected. I than run image embedding on VGG16 model pretrained on ImageNet. It converts 1 data into a length 4096 vector.

K-means clustering algorithm is used with K set to 12. Since Orange does not have a widget that calculates Rand Index, I wrote a customary Python Script to calculate it to check performance.

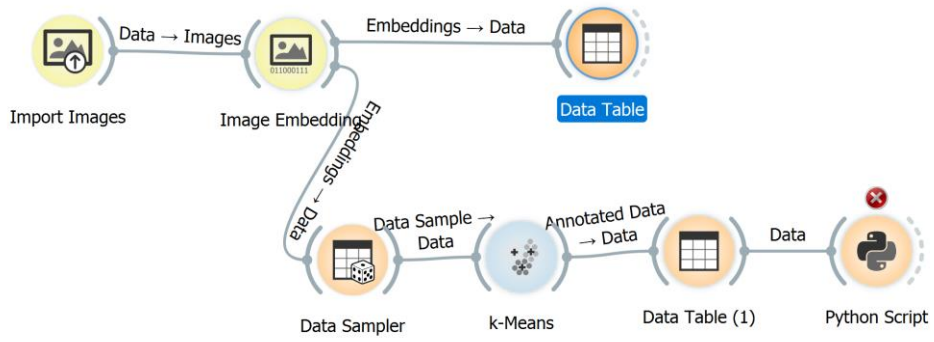

Fig3: flow chart in Orange3 for image embedding based clustering

In this task, it achieved a relatively good number of 0.70.

The results suggest the image embedding technique has good performance with image clustering.

Conclusion

2 month ago, when I started looking at this project, the best score on public leader board is 0.77, and right now it stands at 0.78, suggesting it has reaching its potential given current methods. The score is on par with the best ILSVRC top-1 accuracy score. The Cdiscount challenge is more complex than ILSVRC in that it has more images and more categories, but simpler in image content. So it may not be a surprise that the top-1 accuracy is similar.

Source Files

All codes are written in Python3.

All source file can be found at: https://github.com/hehaijin/cdiscount

For VGG16 classification task:

cdiscountkeras.py: load VGG16 model and save model.

dataset.py: load data set in mini-batches.

train.py: trains with model and saves model weights to *weight.h5*.

predict.py: generates prediction file from *weight.h5*.

plot.py : plots the prediction score

For Xception network:

Xceptionmodel.py : loads the Xception model.

Xceptiontrain.py: trains with Xception model and save weights.

Xceptionpredict.py : predict with xception model.

For clustering:

clusterConvertImage.py reads from training data and save all images related to the specified categories.

Randindex.py calculates Rand Index in Orange3.