­

**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 amount 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 it’s products, with product id, category, and image information.

**Data description:**

Training data: training.bson size: 59.2 GB

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.

Testing data: test.bson size: 14.5 GB

Contains  1,768,182 products. Same format as train.bson, but without the category id.

Category\_names.csv:

Contains the 3 level name for each category id.

**Data storage:**

Bson format

[BSON](https://en.wikipedia.org/wiki/BSON)(Bin­ary JSON) format.

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

**Evaluation metrics:**

Image classification tasks like ImageNet typically 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%, while its top-1 accuracy is in 7x %. Here the Cdiscount challenge only uses top-1 accuracy.

**Preliminary analysis:**

1, Image per product: 1.34

Most product have only 1 image.

2, images:

Size: 180 X180 X 3

Most are color images.

3, Categories:

Each category id corresponds to a 3 level category name.

Total number of categories is 5271

Number of level1 category is about 50.

Number of level2 category is about 350

**Objectives & Execution:**

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

Planned methods:

Deep learning is the state of art method for image classification. Since I plan to use Python environment, I plan to use TensorFlow library with Python, and related libraries like numpy, pandas.

Image annotation method will also be used to enhance prediction accuracy. Image annotation automatically assigns key words to images, and by comparing the key words to category names, it should help classification accuracy.

2: Image clustering

Here the data set contains 3 level of categories. For some categories, I can 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.

To implement the clustering, the method I plan to try several methods include K-means and hierarchical clustering. How to represent distance between images will carefully considered.

**Conclusion**

The classification task is a challenging one for it’s data size, and for it’s big amount of categories. Never the less, it is one that is manageable. The current best score on Kaggle is approaching 0.77 and surely it’ll go up with still 2 month to go. I would be very interested to see the final score.

Data format

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

Similarity:

1. All have huge amount of images. ImageNet has over 1,500 million images. The ILSVRC competition hold every year has 1.2 million images, and about 5000 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.

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 ILSVRC competition, it’s naturally to look into the history of ILSVRC for ideas.

History of ILSVRC:

|  |  |  |  |
| --- | --- | --- | --- |
| Year |  | Top-5 error rate(%) | depth |
| 2012 | AlexNet | 16.3 | 8 |
| 2014 | VGGNet |  | 19 |
| 2014 | Inception Net v1 | 6.67 | 22 |
| 2015 | ResNet | 3.57 | 152 |
| 2016 | CUImage(ensemble) |  |  |

Time estimate:

GTX 1080

Cloud GPU computing resource selection:

Amazon: pricy and slow for on demand GPU instance(0.90/hour for Nvidia K80). 0.27 for Spot Instance.

Paperspace: 0.6 for Nvidia P5000

Floyd: pricy but the interface is really simple, only 2 hour of free GPU time.

Google: $0.74/hour for Nvidia K80. $300 of credit for use.

Use your own machine: a desktop with GTX1080 costs $1000+.

Conclusion: use Google.

Code:

Steps for deep learning

1, use bson.decode\_file\_iter to get a generator for all train/test files.

2, use itertools.islice() to generate batches.

3, output batches of (image,category) data.

Note batch is fixed for sample id, but sample have various number to pictures. So the number of (image,category) data is different.

K80 totalMemory: 11.17GiB freeMemory: 11.09GiB

Cdc-vgg kernel

# without 4th conv

Batch size=16 epoch time =94 hour total batch 695885

Batch size=64 epoch time=69 hour total batch 173971

Batch size=128 OOM error

OOM when allocating tensor with shape[61952,4096]

Number of parameters: 299 767 254

Flatten shape `(?, 61952)`

Output shape `(?, 5270)

#with 4th conv

Batch size 16 epoch time=81 h

Batch size 64 epoch time = 69 h

Batch size 128 epoch time= 73 h

Number of parameters: 105 520 086

Flatten shape `(?, 12800)`

#adding a maxpool [2,2] before the net.

Number of parameters 61479894

Flatten shape (?, 2048)

Batch size 128 epoch time= 19h

Batch size 300 epoch time =18h shows Allocator (GPU\_0\_bfc) ran out of memory

First try.

#adding a maxpool [2,2]. So I am using an image size 90\*90

//resizing to [3,3] is not helping running time.

Learning rate 0.01

Epoch 1

Batch size 128.

Epoch time=18 h.

Prediction speed.

About 800/min.

So about 50000/h

Takes about 2 hours. The count goes up to 9XXXX for batch size of 32.

Exception ignored in: <bound method BaseSession.\_\_del\_\_ of <tensorflow.python.client.session.Session object at 0x7fa8c4c44b70>>hon/client/session.py", line 696, in \_\_del\_\_Traceback (most recent call last):TypeError: 'NoneType' object is not callable

Image clustering through embedding.

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

A recently developed technique called Image embedding can be applied here. I try to use the pre-Trained Convolutional Neural Network to convert a image into a vector containing information representing the image. The penultimate layer of a pretrained network, is used in this case. The hypothesis is that the penultimate layer captures the higher level abstract information of an image, so it can be used to represent a image.