Neural Networks project

Kaggle: Cdiscount's Image Classification Challenge

University of Tartu
Institute of computer science
by A.Potapchuk, V.Fediukov, M.Semikin, V.Mysko

INTRODUCTION

Cdiscount.com generated nearly 3 billion euros last year, making it France's largest non-food e-commerce company. While the company already sells everything from TVs to trampolines, the list of products is still rapidly growing. By the end of this year, Cdiscount.com will have over 30 million products up for sale.

Currently, Cdiscount.com applies machine learning algorithms to the text description of the products in order to automatically predict their category. As these methods now seem close to their maximum potential, Cdiscount.com believes that the next quantitative improvement will be driven by the application of data science techniques to images.

In this challenge, we will be building a model that automatically classifies the products based on their images. Cdiscount.com's website can confirm, one product can have one or several images. The data set is unique and characterized by superlative numbers in several ways:

- Almost 9 million products: half of the current catalog
- More than 15 million images at 180x180 resolution
- 5270 categories

BACKGROUND

The goal of this competition is to predict the category of a product based on its image(s). For every product _id in the test set, we should predict the correct category_id. This competition is evaluated on the categorization accuracy of predictions (measured in percentages).

In this competition we have given data:

train.bson - (Size: 58.2 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, stored in a list (key: imgs). Each image list contains a single dictionary per image, which uses the format: {'picture': b'...binary string...'}. The binary string corresponds to a binary representation of the image in JPEG format.

test.bson - (Size: 14.5 GB) Contains a list of 1,768,182 products in the same format as train.bson, except there is no category_id included. The objective of the competition is to predict the correct category_id from the picture(s) of each product id (_id). The category_ids that are present in Private Test split are also all present in the Public Test split.

category_names.csv - Shows the hierarchy of product classification. Each category_id has a corresponding level1, level2, and level3 name, in French. The category_id corresponds to the category tree down to its lowest level. This hierarchical data is useful, but it is not necessary for building models and making predictions. All the absolutely necessary information can be found in train.bson.

sample_submission.csv - Shows the correct format for submission.

BSON files, that we are using in this project is a binary-encoded serialization of JSON-like documents, used with MongoDB.

Also, we discovered the relation between id and category (Figure 1). As can be seen from the picture, blue color response for how strong is the relation. So, dark blue - relation goes to 1, light blue - there is, probably, no relation between category and id number.

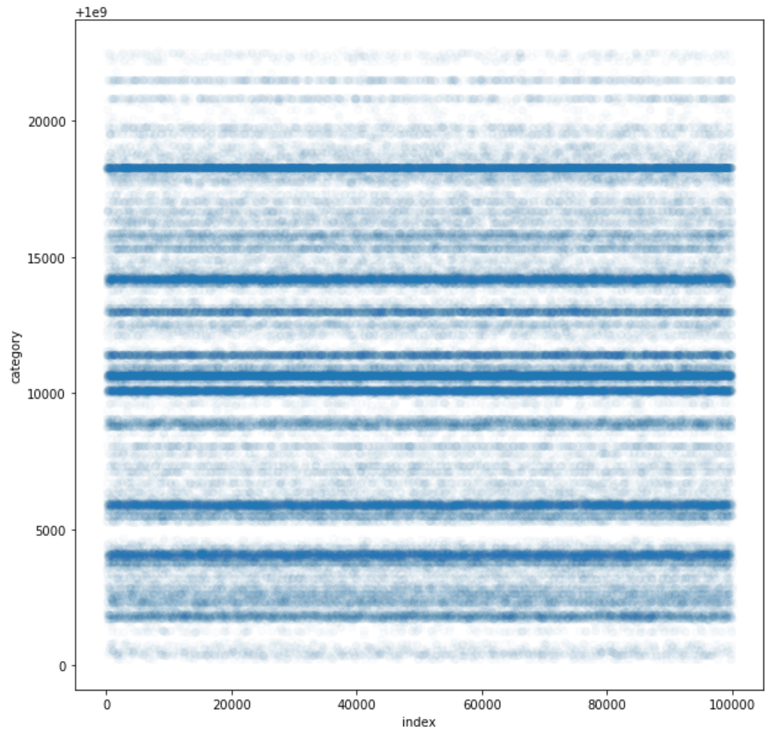

Figure 1

For the better understanding of the data, we should know the most and the least common categories. Below, there are the most common and the least common categories by first 3 levels:

	category	category_level1	category_level2	category_level3
1000018296	79640	MUSIQUE	CD	CD POP ROCK - CD ROCK INDE
1000011423	71116	INFORMATIQUE	IMPRESSION - SCANNER	TONER - RECUPERATEUR DE TONER
1000011427	69784	INFORMATIQUE	IMPRESSION - SCANNER	CARTOUCHE IMPRIMANTE
1000014202	65642	LIBRAIRIE	LITTERATURE	LITTERATURE FRANCAISE
1000015309	65435	LIBRAIRIE	AUTRES LIVRES	AUTRES LIVRES
1000004085	61942	INFORMATIQUE	CONNECTIQUE - ALIMENTATION	BATTERIE D'ALIMENTATION INFORMATIQUE
1000010653	61688	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	COQUE TELEPHONE - BUMPER TELEPHONE
1000018290	60332	MUSIQUE	CD	CD MUSIQUE CLASSIQUE
1000018294	57748	MUSIQUE	CD	CD MUSIQUE DU MONDE
1000008094	56192	INFORMATIQUE	COMPOSANT - PIECE DETACHEE	DALLE D'ECRAN
1000004079	55656	INFORMATIQUE	CONNECTIQUE - ALIMENTATION	CHARGEUR - ADAPTATEUR SECTEUR - ALLUME CIGARE ...
1000005509	51332	AUTO - MOTO	CONFORT CONDUCTEUR ET PASSAGER	PERSONNALISATION VEHICULE - DECORATION VEHICULE
1000015912	49780	INFORMATIQUE	COMPOSANT - PIECE DETACHEE	CLAVIER (PIECE DETACHEE)
1000010635	48488	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	BATTERIE TELEPHONE
1000011349	47691	TELEPHONIE - GPS	ACCESSOIRE TELEPHONE	PACK ACCESSOIRES

Figure 2. The most common categories

1000011519	12	MATERIEL MEDICAL	ACUPUNCTURE - MEDECINES PARALELLES	VENTOUSE
1000000896	12	EPICERIE	CONSERVE DE LEGUME	POIVRON EN CONSERVE
1000015609	12	CHAUSSURES - ACCESSOIRES	ACCESSOIRES CHAUSSURES	ESSUIE-BOTTES - LAVE-BOTTES
1000019484	12	MEUBLE	ACCESSOIRE DE MEUBLE	COLONNE SUSPENDUE
1000019804	12	SPORT	BASEBALL	BLOUSON DE BASEBALL - VESTE DE BASEBALL
1000007168	12	SPORT	CYCLES	TRIPORTEUR
1000022325	12	TV - VIDEO - SON	LECTEUR MUSIQUE	LECTEUR MP4 RECONDITIONNE - LECTEUR NUMERIQUE ...
1000015046	12	MATERIEL DE BUREAU	MATERIEL PEDAGOGIQUE	REGISTRE D'APPEL - CAHIER DE CLASSE
1000011955	12	MATERIEL MEDICAL	SOIN	CATHETER - OBTURATEUR
1000007760	12	PUERICULTURE	TOILETTE BEBE	EXTENSION DE ROBINET
1000010893	12	PHOTO - OPTIQUE	VISIONNAGE PHOTO	SCANNER DE DIAPOSITIVE

Figure 3. The least common categories

There was used public kernel [2] as a base of image generator with some changes that improve a performance.

Architecture

We choose next architectures for this competition: ResNet50, ResNet101, InceptionV3, InceptionResNetV2, Inception3. Our decision was based mostly on collecting information from public kernels and relevant to competition chats. Our stack was Keras with TensorFlow backend, and mostly all of this models are present in Keras. First our steps was using pretrained models from public kernels (ResNet101) also was added 4 fully connected layers on top and made additional training, without augmentation. Then we trained ResNet101 and ResNet50 from beginning (20 epochs with batch 512 for ResNet50 and 128 for ResNet101) with additional dense, dropout and average pooling layers (the best combination). Also we gradually froze layers while training. On first 10 epochs was used only first residual block, on every next 5 epochs the new residual block has been added. This approaches gives us 70.5% and 71.2%, 67,5% respectively on leaderboard. For this part were responsible Maxim Semikin and Vladislav Fediukov.

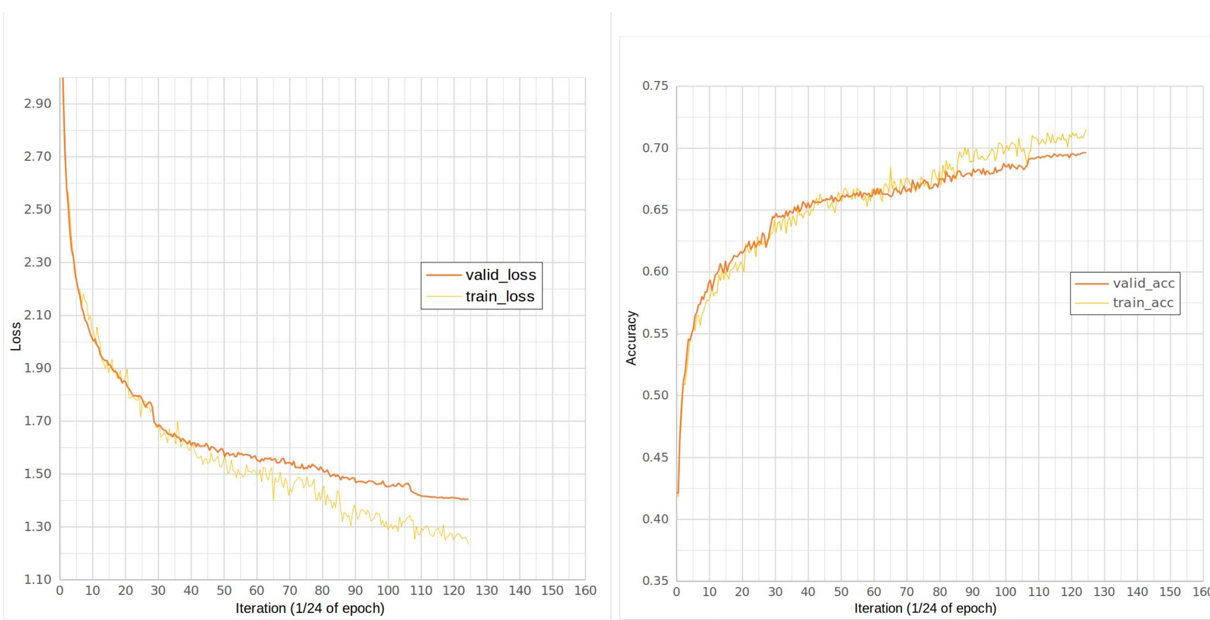

Figure 4. Loss of ResNet50

Maxim Semikin also have trained XCeption with additional fully connected layer, BatchNorm, Dropout and Softmax layers on top. This approach gives us 69% on leaderboard.

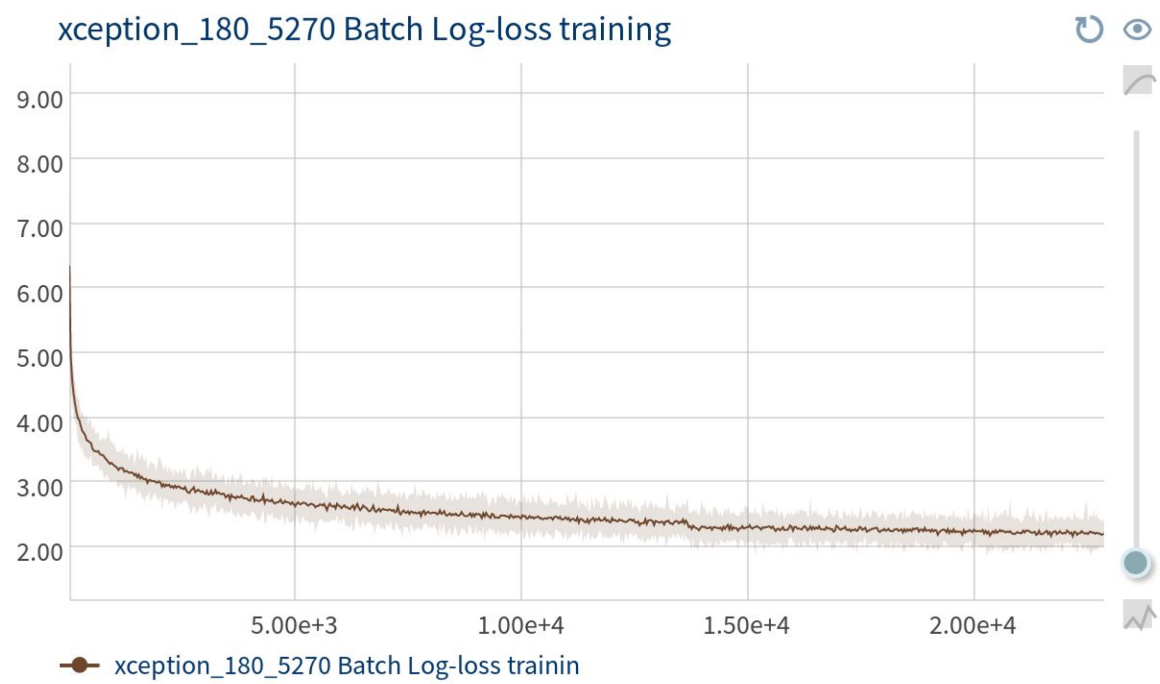

Figure 5. Loss of Xception

Next architecture was InceptionV3 with additional dropout and average pooling layers (15 epochs, batch 256). But this architecture showed very slow speed of learning and not high result - 69%. This was implemented by Viktor Mysko.

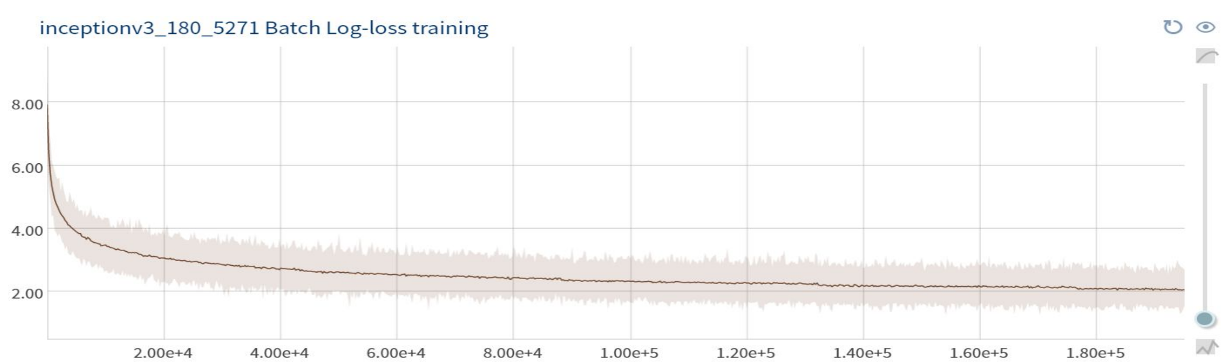

Figure 6. InceptionV3 loss

Anton Potapchuk trained InceptionResNetV2[1] (Figure 7). There is no a split to validation and test data, because some classes have a very small number of samples (12). So, splitting to test and validation sets reduce the number of samples more.

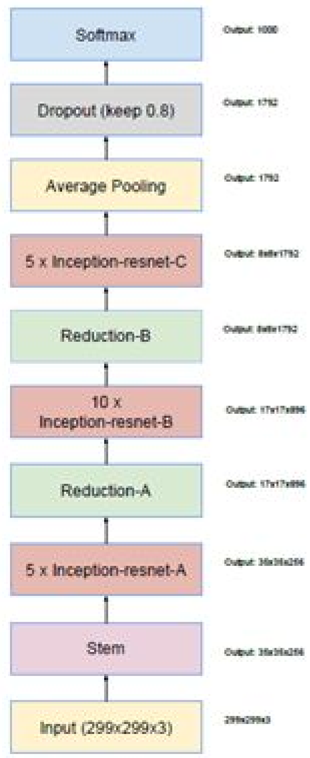

Figure 7. Schema for Inception-ResNet-v2 network

Top layers are GlobalAveragePooling, Dropout with 0.1 rate and dense layer with softmax activation function. During the first epoch, only the top layer was trained. During the next three epochs, whole Inception-resnet-C block (top 164 layers) was trained. On the last epochs, Inception-resnet-B and Inception-resnet-C block was trained (top 507 layers).

Two InceptionResNetV2 models were trained. In the first model, more layers were trainable (first 2 Inception-resnet-A blocks). Figure 2 shows the learning process of two models. The whole dataset is 24 epochs, so we can see, that after 24 epochs the accuracy increased.

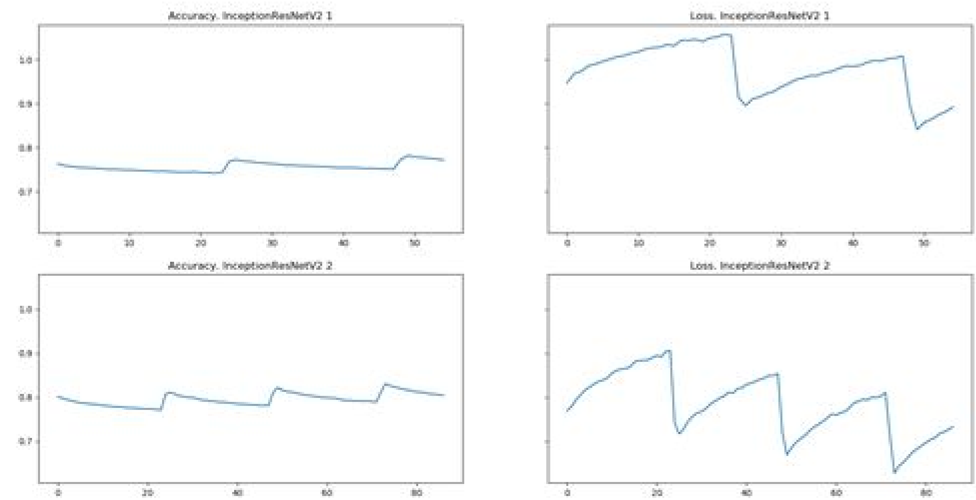

Figure 8. Learning progress of the model.

The performance of the single model on the leaderboard ranges from 0.70 to 0.72 and this two models was best from our single.

Ensembles

Main idea on ensembling was extraction of probabilities of every picture for every good and built simple classifier on top of them - feed forward network with 2 layers of random forest. But this approaches showed big overfitting, so we decide take average and geometric mean of prediction of every picture of two models InteceptionResNetV2 models. This gave us the best, 0.73083, accuracy on the leaderboard.

References

[1] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi
Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning
[2]
https://www.kaggle.com/humananalog/keras-generator-for-reading-directly-from-bson