Cdiscount’s Image Classification Challenge

1.Business/Real-world Problem

1.2. Problem Statement

kaggle has begun a competiton Cdiscount’s Image Classification Challenge

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. This is up from 10 million products only 2 years ago. Ensuring that so many products are well classified is a challenging task. 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 are required to build a model that automatically classifies the products based on their images. As a quick tour of Cdiscount.com's website can confirm, one product can have one or several images. The data set Cdiscount.com is making available is unique and characterized by superlative numbers in several ways:

- Almost 9 million products: half of the current catalogue
- More than 15 million images at 180x180 resolution
- More than 5000 categories: yes this is quite an extreme multi-class classification!

1.3 Source/Useful Links

Competition Page: https://www.kaggle.com/competitions/cdiscount-image-classification-challenge/overview

2. Machine Learning Problem

2.1. Data

2.1.1. Data Overview

The datasets consists of -

BSON Files

BSON, short for Binary JSON, is a binary-encoded serialization of JSON-like documents, used with MongoDB.

File Descriptions

- Please Note: The train and test files are very large!

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. This kernel provides an example of how to process the data.

2. train_example.bson - Contains the first 100 records of train.bson so you can start exploring the data before downloading the entire set.
3. 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.
4. 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 may be useful, but it is not necessary for building models and making predictions. All the absolutely necessary information is found in train.bson.
5. sample_submission.csv - Shows the correct format for submission. It is highly recommended that you zip your submission file before uploading for scoring.

2.1.2. Example Data Point

data point contains dictionaries, one per product. Each dictionary contains :

- product id (key: _id)
- the category id of the product (key: category_id),
- 1-4 images, stored in a list (key: imgs).


```
{'_id': 0, 'imgs': [{'picture': b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00\x00\x01\x00\
```


2.2. Mapping the real-world problem to an ML problem

2.2.1. Type of Machine Learning Problem

```
there are 5270 different category_id
The goal of the competition is to predict category_id by image. We need to predict a number
This is an image classifiaction task but The variable number of images (1-4) for each produ
```

2.2.2. Performance Matrix

Accuracy scores:

Top-1 scores are used to measure training and validation split classification performance on training instances, therefore product images. The score was used to be the same as one used in the competition . Score is defined below in the formula, where T1 is Top-1 score, c1 is number of training instances where most probable class is the same as the target label and n is the number of training instances in the split.

$$T1 = c1/n$$

Kaggle score:

Kaggle score is a metric assigned by an online system that is part of Kaggle competition framework. In order to obtain the score, one must upload a CSV file containing product identifiers from unlabeled test dataset portion. The score cannot be computed offline since the correct labels of the test dataset are not publicly available and not available to the author of this work. According to the definition on Kaggle, the score is the number of correctly classified products over all of the products, meaning that it's Top-1 score for products, which is written in form of the equation, where K1 is the Kaggle score, p is a number of correctly classified products and m is the number of products in the test dataset.

```
K1 = p/m
```

Loss function:

All the models are performing classification task into multiple categories and so cross-categorical entropy can be used.

3 Importing Libraries and Data

```
import os
import sys
import numpy as np
import pandas as pd
import bson
import cv2
import matplotlib.pyplot as plt
import seaborn as sns
```

```
INPUT_PATH = os.path.join('/content/drive/Shareddrives/datascience/', 'Case_study_2')
CATEGORY_NAMES_DF = pd.read_csv(os.path.join(INPUT_PATH, 'category_names.csv'))
TRAIN_DB = bson.decode_file_iter(open(os.path.join(INPUT_PATH, 'train.bson'), 'rb'))
#TEST_DB = bson.decode_file_iter(open(os.path.join(INPUT_PATH, 'test.bson'), 'rb'))
```

```
for item in TRAIN_DB:
    break
print(type(item), list(item.keys()))
print(item['_id'], len(item['imgs']), item['category_id'],)
```

```
<class 'dict'> ['_id', 'imgs', 'category_id']
4 1 1000015539
```

CATEGORY_NAMES_DF (category_names.csv)

```
# display dataframe
CATEGORY_NAMES_DF.head()
```

	category_id	category_level1	category_level2	category_level3
0	1000021794	ABONNEMENT / SERVICES	CARTE PREPAYEE	CARTE PREPAYEE MULTIMEDIA
1	1000012764	AMENAGEMENT URBAIN - VOIRIE	AMENAGEMENT URBAIN	ABRI FUMEUR
2	1000012776	AMENAGEMENT URBAIN - VOIRIE	AMENAGEMENT URBAIN	ABRI VELO - ABRI MOTO
[illegible]	[illegible]	AMENAGEMENT URBAIN -	AMENAGEMENT	[illegible]

```
level_tags = CATEGORY_NAMES_DF.columns[1:]
CATEGORY_NAMES_DF[CATEGORY_NAMES_DF['category_id'] == item['category_id']][level_tags]
```

	category_level1	category_level2	category_level3
1492	BRICOLAGE - OUTILLAGE - QUINCAILLERIE	SECURITE MAISON	ALARME AUTONOME

```
print("Unique categories: ", len(CATEGORY_NAMES_DF['category_id'].unique()))
print("Unique level 1 categories: ", len(CATEGORY_NAMES_DF['category_level1'].unique()))
print("Unique level 2 categories: ", len(CATEGORY_NAMES_DF['category_level2'].unique()))
print("Unique level 3 categories: ", len(CATEGORY_NAMES_DF['category_level3'].unique()))
```

```
Unique categories:  5270
Unique level 1 categories:  49
Unique level 2 categories:  483
Unique level 3 categories:  5263
```

Observations

- Table CATEGORY_NAMES_DF shows the hierarchy of product classification.
- category_id has 3 category tags of different levels
- Using category_id field we can associate images to 3 levels of category tags, labels.

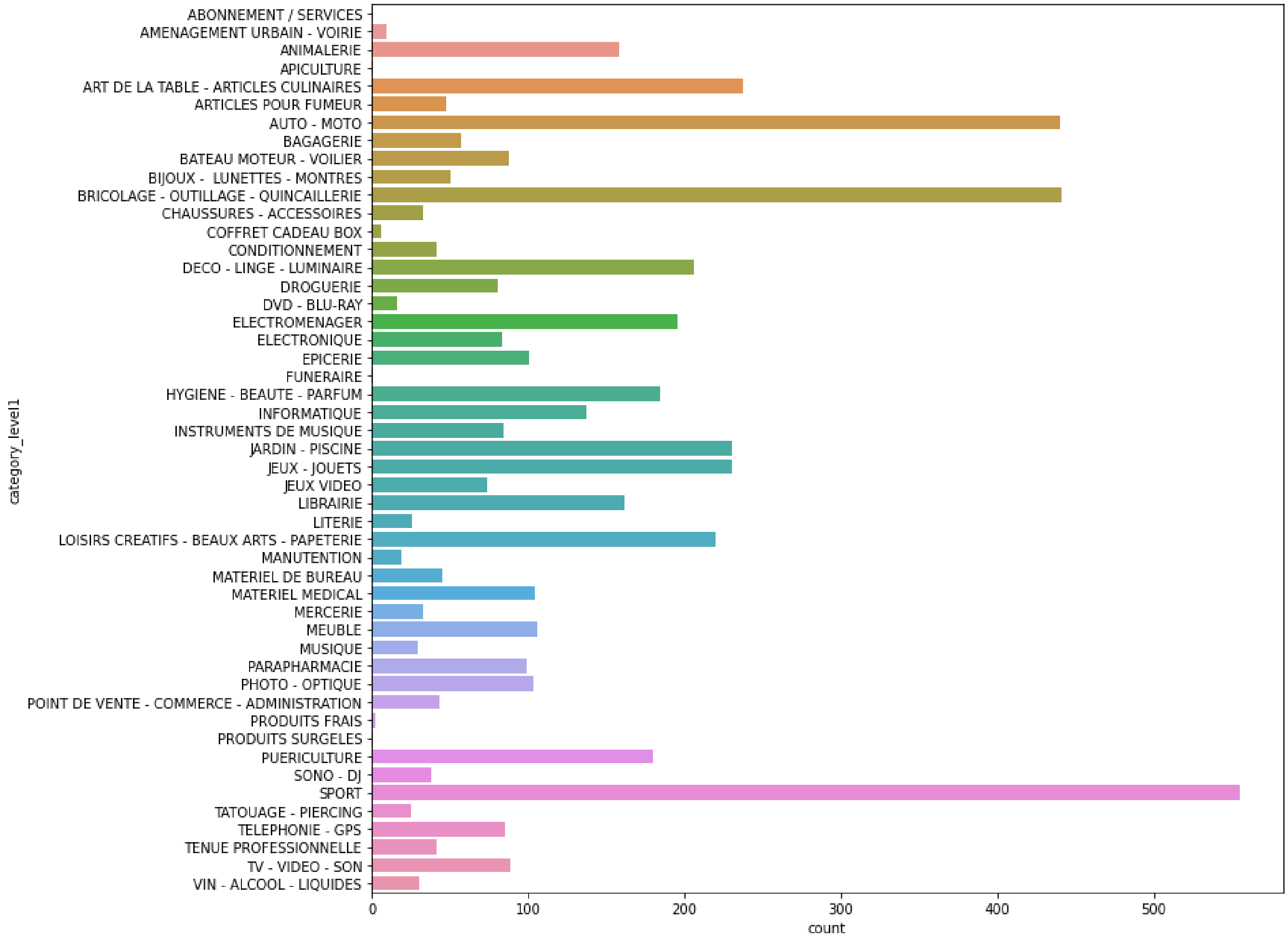
The histogram of level 1 categories

```
# value counts
CATEGORY_NAMES_DF[['category_level1']].value_counts()
```

```
category_level1
SPORT                                       555
BRICOLAGE - OUTILLAGE - QUINCAILLERIE       441
AUTO - MOTO                                 440
ART DE LA TABLE - ARTICLES CULINAIRES       237
JARDIN - PISCINE                            230
```

```
JEUX - JOUETS                                230
LOISIRS CREATIFS - BEAUX ARTS - PAPETERIE    220
DECO - LINGE - LUMINAIRE                     206
ELECTROMENAGER                               196
HYGIENE - BEAUTE - PARFUM                    184
PUERICULTURE                                 180
LIBRAIRIE                                    162
ANIMALERIE                                   158
INFORMATIQUE                                 137
MEUBLE                                       106
MATERIEL MEDICAL                             104
PHOTO - OPTIQUE                              103
EPICERIE                                     101
PARAPHARMACIE                                 99
TV - VIDEO - SON                              89
BATEAU MOTEUR - VOILIER                       88
TELEPHONIE - GPS                              85
INSTRUMENTS DE MUSIQUE                        84
ELECTRONIQUE                                  83
DROGUERIE                                     81
JEUX VIDEO                                    74
BAGAGERIE                                     57
BIJOUX -  LUNETTES - MONTRES                  50
ARTICLES POUR FUMEUR                          48
MATERIEL DE BUREAU                            45
POINT DE VENTE - COMMERCE - ADMINISTRATION    43
TENUE PROFESSIONNELLE                         42
CONDITIONNEMENT                               42
SONO - DJ                                     38
CHAUSSURES - ACCESSOIRES                      33
MERCERIE                                      33
VIN - ALCOOL - LIQUIDES                       30
MUSIQUE                                       29
LITERIE                                       26
TATOUAGE - PIERCING                           25
MANUTENTION                                   19
DVD - BLU-RAY                                 16
AMENAGEMENT URBAIN - VOIRIE                    9
COFFRET CADEAU BOX                             6
PRODUITS FRAIS                                 2
PRODUITS SURGELES                              1
FUNERAIRE                                      1
APICULTURE                                     1
ABONNEMENT / SERVICES                          1
dtype: int64
```

```
plt.figure(figsize=(12,12))
_ = sns.countplot(y=CATEGORY_NAMES_DF['category_level1'])
```



Observation

- Here we can see that SPORT is most frequent category
- ABONNEMENT/SERVICES, APICULTURE, PRODUCITS SURGELES, and FUNERAIRE are less frequent only occuring 1 time

Level 2 catrgories

```
# Level 2
cat_level2_counts = CATEGORY_NAMES_DF.groupby('category_level2')['category_level2'].count()
print(cat_level2_counts.describe())
print()
print("Level 2 the most frequent category: ", cat_level2_counts.argmax())
print("Level 2 the less frequent category: ", cat_level2_counts.argmin())
```

```
count    483.000000
mean      10.910973
std       13.035315
min        1.000000
25%        4.000000
50%        8.000000
75%       13.000000
max      187.000000
Name: category_level2, dtype: float64

Level 2 the most frequent category:  353
Level 2 the less frequent category:  12
```

Obseravtion

- In Level 2 the most frequent category is : 353
- Level 2 the less frequent category is : 12

Level 3 categories

```
cat_level3_counts = CATEGORY_NAMES_DF.groupby('category_level3')['category_level3'].count()
print(cat_level3_counts.describe())
print()
print("Level 3 the most frequent category: ", cat_level3_counts.argmax())
```

```
count    5263.000000
mean        1.001330
std         0.036449
min         1.000000
25%         1.000000
50%         1.000000
75%         1.000000
max         2.000000
Name: category_level3, dtype: float64

Level 3 the most frequent category:  1480
```

```
CATEGORY_NAMES_DF[['category_level3']].value_counts()
```

```
category_level3
FONTAINE A EAU                                   2
VOITURE                                          2
PELUCHE                                          2
CONFORT URINAIRE                                 2
FUSIBLE                                          2
                                                ..
DESODORISANT AUTO - PARFUM AUTO                  1
DESODORISANT - NETTOYANT A LITIERE               1
DESINFECTION DES INSTRUMENTS - LAVE-INSTRUMENTS  1
DESINFECTION DENTAIRE                            1
```

```
ŒILLETS                                            1
Length: 5263, dtype: int64
```

Observation

- level 3 catogeries is at highest granularity level

Train_DF(train.bson)

```
# lets check sample data format

for c,d in enumerate(TRAIN_DB):
  if c == 1:
        break
  print(d)
```


```
{'_id': 0, 'imgs': [{'picture': b'\xff\xd8\xff\xe0\x00\x10JFIF\x00\x01\x01\x00\x00\x01\x
```


First images in train datasets

As it is said in data description page, TRAIN_DB contains a list of 7,069,896 dictionaries, one per product. Each dictionary contains :

- product id (key: _id)
- the category id of the product (key: category_id),
- 1-4 images, stored in a list (key: imgs).

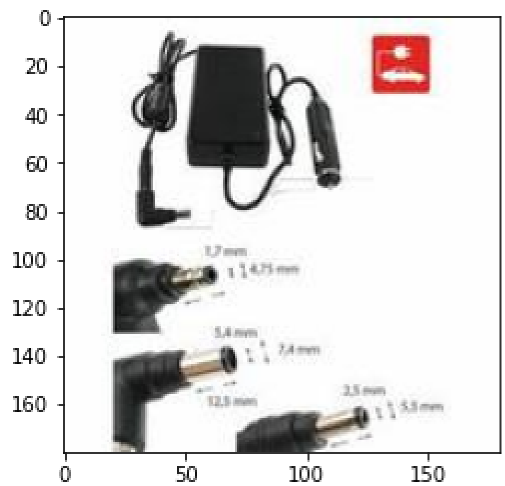
```
for item in TRAIN_DB:
    break
print(type(item), list(item.keys()))
print(item['_id'], len(item['imgs']), item['category_id'],)
```

```
<class 'dict'> ['_id', 'imgs', 'category_id']
2 1 1000004079
```

Images (key: imgs)

```
# https://www.geeksforgeeks.org/python-opencv-imdecode-function/
def decode(data):
    arr = np.asarray(bytearray(data), dtype=np.uint8)
    img = cv2.imdecode(arr, cv2.IMREAD_COLOR)
```

```
        return cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    import io
    from PIL import Image

    def decode_pil(data):
        return Image.open(io.BytesIO(data))

    for img_dict in item['imgs']:
        img = decode(img_dict['picture'])
        plt.figure()
        plt.imshow(img)
```



```
    # this code is refered from https://www.kaggle.com/code/vfdev5/data-visualization-and-analysi
    # Method to compose a single image from 1 - 4 images
    def decode_images(item_imgs):
        nx = 2 if len(item_imgs) > 1 else 1
        ny = 2 if len(item_imgs) > 2 else 1
        composed_img = np.zeros((ny * 180, nx * 180, 3), dtype=np.uint8)
        for i, img_dict in enumerate(item_imgs):
            img = decode(img_dict['picture'])
            h, w, _ = img.shape
            xstart = (i % nx) * 180
            xend = xstart + w
            ystart = (i // nx) * 180
            yend = ystart + h
            composed_img[ystart:yend, xstart:xend] = img
        return composed_img


    max_counter = 15
    counter = 0
    n = 4
    for item in TRAIN_DB:
        if counter % n == 0:
            plt.figure(figsize=(14, 6))
```

```
    mask = CATEGORY_NAMES_DF['category_id'] == item['category_id']
    plt.subplot(1, n, counter % n + 1)
    cat_levels = CATEGORY_NAMES_DF[mask][level_tags].values.tolist()[0]
    cat_levels = [c[:25] for c in cat_levels]
    title = str(item['category_id']) + '\n'
    title += '\n'.join(cat_levels)
    plt.title(title)
    plt.imshow(decode_images(item['imgs']))
    plt.axis('off')

    counter += 1
    if counter == max_counter:
        break
```

1000010653
TELEPHONIE - GPS
ACCESSOIRE TELEPHONE
COQUE TELEPHONE - BUMPER

1000005744
AUTO - MOTO
PIECES
BOBINE D'ALLUMAGE - BOBIN

1000004079
INFORMATIQUE
CONNECTIQUE - ALIMENTATIO
CHARGEUR - ADAPTATEUR SEC

1000010667
TELEPHONIE - GPS
ACCESSOIRE TELEPHONE
HOUSSE - ETUI - CHAUSSETT

1000018290
MUSIQUE
CD
CD MUSIQUE CLASSIQUE

1000010653
TELEPHONIE - GPS
ACCESSOIRE TELEPHONE
COQUE TELEPHONE - BUMPER

1000018306
MUSIQUE
CD
CD VARIETE INTERNATIONALE

1000010961
TV - VIDEO - SON
CASQUE - MICROPHONE - DIC
CASQUE - ECOUTEUR - OREIL

Random item access

Let's make a random access to products by maping each product byte offset and length.

Following code creates a dictionary with key indexing item _id and values (offset, length)

1000015309 1000010653 1000007361 1000018294

```
# https://www.kaggle.com/code/vfdev5/random-item-access

import struct
from tqdm import tqdm_notebook

num_dicts = 7069896 # according to data page
length_size = 4
IDS_MAPPING = {}

with open(os.path.join(INPUT_PATH, 'train.bson'), 'rb') as f, tqdm_notebook(total=num_dicts)
    item_data = []
    offset = 0
    while True:
        bar.update()
        f.seek(offset)

        item_length_bytes = f.read(length_size)
        if len(item_length_bytes) == 0:
            break
        # Decode item length:
        length = struct.unpack("<i", item_length_bytes)[0]

        f.seek(offset)
        item_data = f.read(length)
        assert len(item_data) == length, "%i vs %i" % (len(item_data), length)

        # Check if we can decode
        item = bson.BSON.decode(item_data)
```

```
        IDS_MAPPING[item['_id']] = (offset, length)
        offset += length

def get_item(item_id):
    assert item_id in IDS_MAPPING
    with open(os.path.join(INPUT_PATH, 'train.bson'), 'rb') as f:
        offset, length = IDS_MAPPING[item_id]
        f.seek(offset)
        item_data = f.read(length)
        return bson.BSON.decode(item_data)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:10: TqdmDeprecationWarning
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  # Remove the CWD from sys.path while we load stuff.
```

7069897/? [08:26<00:00, 18830.17it/s]

```
item = get_item(1234)

mask = CATEGORY_NAMES_DF['category_id'] == item['category_id']
cat_levels = CATEGORY_NAMES_DF[mask][level_tags].values.tolist()[0]
cat_levels = [c[:25] for c in cat_levels]
title = str(item['category_id']) + '\n'
title += '\n'.join(cat_levels)
plt.title(title)
plt.imshow(decode_images(item['imgs']))
_ = plt.axis('off')
```

1000010667
TELEPHONIE - GPS
ACCESSOIRE TELEPHONE
HOUSSE - ETUI - CHAUSSETT

Product ID (key:_id) and Category ID (key:category_id)

```
#creating dataframe of _id and category_id from train.bson
```

```
from tqdm import tqdm_notebook

num_dicts = 7069896 # according to data page
prod_to_category = [None] * num_dicts

with tqdm_notebook(total=num_dicts) as bar:
    TRAIN_DB = bson.decode_file_iter(open(os.path.join(INPUT_PATH, 'train.bson'), 'rb'))

    for i, item in enumerate(TRAIN_DB):
        bar.update()
        prod_to_category[i] = (item['_id'], item['category_id'])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:7: TqdmDeprecationWarning:
Please use `tqdm.notebook.tqdm` instead of `tqdm.tqdm_notebook`
  import sys
```

100% 7069896/7069896 [09:51<00:00, 6822.83it/s]

```
TRAIN_CATEGORIES_DF = pd.DataFrame(prod_to_category, columns=['_id', 'category_id'])
TRAIN_CATEGORIES_DF.head()
```

	_id	category_id
0	0	1000010653
1	1	1000010653
2	2	1000004079
3	3	1000004141
4	4	1000015539

```
print("Unique categories: %i in %i entries" % (len(TRAIN_CATEGORIES_DF['category_id'].unique(
```

```
Unique categories: 5270 in 7069896 entries
```

```
# Distribution of categories

train_categories_gb = TRAIN_CATEGORIES_DF.groupby('category_id')
train_categories_count = train_categories_gb['category_id'].count()
print(train_categories_count.describe())
```

```
count     5270.000000
mean      1341.536243
std       4941.011223
min         12.000000
25%         69.000000
50%        200.000000
75%        718.500000
```

```
max      79640.000000
```

```
TRAIN_CATEGORIES_DF.head()
```

	_id	category_id
0	0	1000010653
1	1	1000010653
2	2	1000004079
3	3	1000004141
4	4	1000015539

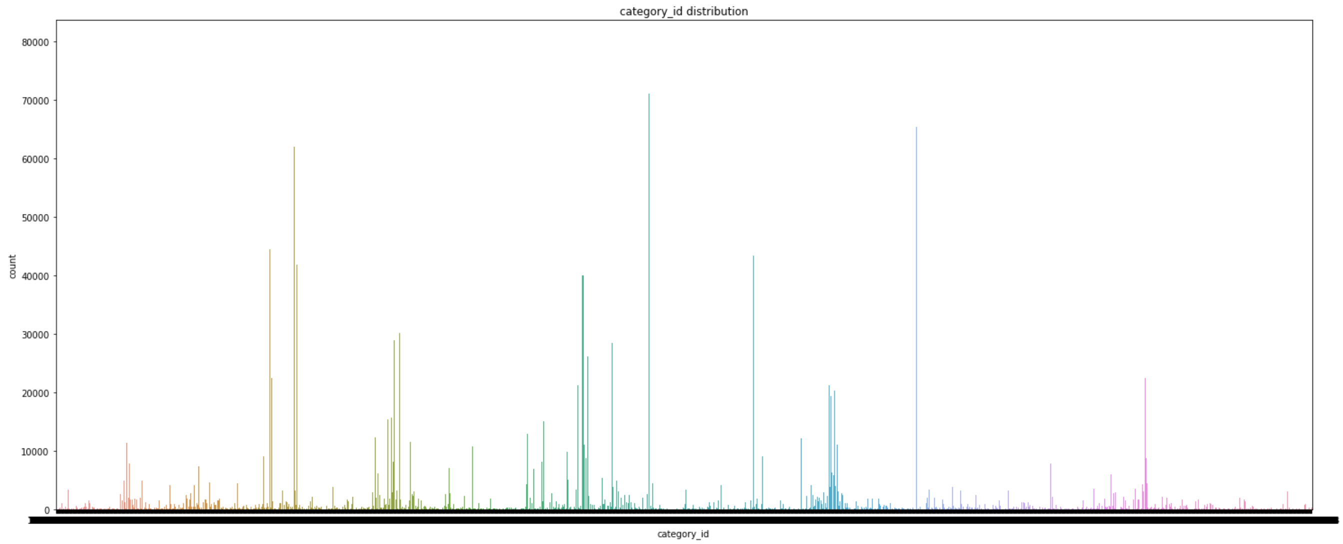
```
TRAIN_CATEGORIES_DF.category_id.value_counts()
```

```
1000018296    79640
1000011423    71116
1000011427    69784
1000014202    65642
1000015309    65435
              ...
1000019608       12
1000012168       12
1000017733       12
1000010893       12
1000019484       12
Name: category_id, Length: 5270, dtype: int64
```

```
TRAIN_CATEGORIES_DF.category_id.unique()
```

```
array([1000010653, 1000004079, 1000004141, ..., 1000012571, 1000020847,
       1000011375])
```

```
# count of category_id
fig, ax = plt.subplots(figsize=(25,10))
ax = sns.countplot(x='category_id', data=TRAIN_CATEGORIES_DF).set_title('category_id distribu
```



Obseravtion

- 5270 unique categories in 7069896 entries
- Max is 79640 and Min is 12
- from count plot we can see that we have imbalanced dataset

```
most_freq_cats = train_categories_count[train_categories_count == train_categories_count.max(
less_freq_cats = train_categories_count[train_categories_count == train_categories_count.min(

print("Most frequent category: ", CATEGORY_NAMES_DF[CATEGORY_NAMES_DF['category_id'].isin(mos
print("Less frequent category: ", CATEGORY_NAMES_DF[CATEGORY_NAMES_DF['category_id'].isin(les
```

```
Most frequent category:  [[1000018296 'MUSIQUE' 'CD' 'CD POP ROCK - CD ROCK INDE']]
Less frequent category:  [[1000017266 'APICULTURE' "OUTILS DE L'APICULTEUR"
  'CHASSE-ABEILLES - PIEGE INSECTES - BOUCHON PIEGE']
 [1000022465 'BATEAU MOTEUR - VOILIER' 'ELECTRICITE'
  'CONVERTISSEUR DE TENSION']
 [1000017559 'BATEAU MOTEUR - VOILIER'
  'PIECE MOTEUR DE BATEAU - PIECE MOTEUR DE HORS BORD'
  'ECHAPEMENT - VENTILATION']
 [1000015609 'CHAUSSURES - ACCESSOIRES' 'ACCESSOIRES CHAUSSURES'
  'ESSUIE-BOTTES - LAVE-BOTTES']
 [1000008633 'CONDITIONNEMENT' 'CALAGE - PROTECTION' 'COUSSIN GONFLABLE']
 [1000012168 'DROGUERIE' 'COMBUSTIBLE' 'CHARBON DE BOIS']
 [1000012287 'DROGUERIE' "MATERIEL D'ENTRETIEN" 'PINCE A DECHETS']
 [1000003589 'ELECTROMENAGER' 'ENTRETIEN DES SOLS - MAISON' 'CIREUSE']
 [1000013297 'ELECTRONIQUE' 'CAPTEURS'
  'SYSTEME DE SURVEILLANCE DE NIVEAU']
 [1000000896 'EPICERIE' 'CONSERVE DE LEGUME' 'POIVRON EN CONSERVE']
 [1000016613 'HYGIENE - BEAUTE - PARFUM' 'CAPILLAIRE'
  'PASSE A MECHE - CROCHET A MECHE']
 [1000008894 'JEUX - JOUETS' 'JONGLERIE' 'ASSIETTE CHINOISE']
 [1000014467 'LOISIRS CREATIFS - BEAUX ARTS - PAPETERIE'
  'COLLECTION - PHILATELIE - CARTOPHILIE - NUMISMATIQUE'
  "TROUSSE D'EXPERTISE - PACK MULTI-OUTILS"]
 [1000015046 'MATERIEL DE BUREAU' 'MATERIEL PEDAGOGIQUE'
```

```
 "REGISTRE D'APPEL - CAHIER DE CLASSE"]
[1000011519 'MATERIEL MEDICAL' 'ACUPUNCTURE - MEDECINES PARALELLES'
 'VENTOUSE']
[1000011955 'MATERIEL MEDICAL' 'SOIN' 'CATHETER - OBTURATEUR']
[1000011638 'MATERIEL MEDICAL' 'VETEMENTS MEDICAUX'
 'LUNETTES MEDICALES - LUNETTES DE PROTECTION']
[1000019484 'MEUBLE' 'ACCESSOIRE DE MEUBLE' 'COLONNE SUSPENDUE']
[1000019608 'PHOTO - OPTIQUE' 'PIECES DETACHEES PHOTO - OPTIQUE'
 'LECTEUR CARTE MEMOIRE - HUB']
[1000010893 'PHOTO - OPTIQUE' 'VISIONNAGE PHOTO'
 'SCANNER DE DIAPOSITIVE']
[1000012074 'POINT DE VENTE - COMMERCE - ADMINISTRATION' 'SECURITE'
 'MIROIR DE SURVEILLANCE']
[1000017733 'PUERICULTURE' 'SOIN MAMAN'
 'HOUSSE COUSSIN GROSSESSE - HOUSSE COUSSIN ALLAITEMENT']
[1000000522 'PUERICULTURE' 'SOMMEIL BEBE' 'FLECHE DE LIT BEBE']
[1000007760 'PUERICULTURE' 'TOILETTE BEBE' 'EXTENSION DE ROBINET']
[1000019804 'SPORT' 'BASEBALL' 'BLOUSON DE BASEBALL - VESTE DE BASEBALL']
[1000007168 'SPORT' 'CYCLES' 'TRIPORTEUR']
[1000019423 'SPORT'
 "MATERIEL D'ENTRETIEN DE SPORT - PRODUIT D'ENTRETIEN DE SPORT"
 "PRODUIT D'ENTRETIEN SPORT DE GLISSE - DESINFECTANT"]
[1000020153 'SPORT' 'PECHE'
 'MAILLOT DE PECHE - DEBARDEUR DE PECHE - T-SHIRT DE PECHE - POLO DE PECHE']
[1000010603 'TELEPHONIE - GPS' 'ACCESSOIRE GPS' 'TELECOMMANDE GPS']
[1000022325 'TV - VIDEO - SON' 'LECTEUR MUSIQUE'
 'LECTEUR MP4 RECONDITIONNE - LECTEUR NUMERIQUE MULTIMEDIA RECONDITIONNE']
[1000020847 'TV - VIDEO - SON' 'PROTECTION - ENTRETIEN'
 'HOUSSE POUR ENREGISTEUR - HOUSSE POUR DICTAPHONE - ETUI POUR ENREGISTREUR - ETUI POUR
```

Observation

- 1 most frequent category (found 79640 times) : MUSIQUE (en.: music)
- 31 less frequent categories (found 12 times) : PUERICULTURE (en.: childcare), APICULTURE (en.: beekeeping), SPORT/BASEBALL/BLOUSON DE BASEBALL - VESTE DE BASEBALL, ...

```
#display most freq. categories
most_freq_cat = most_freq_cats.index[0]

plt.figure(figsize=(16, 4))
mask = CATEGORY_NAMES_DF['category_id'] == most_freq_cat
cat_levels = CATEGORY_NAMES_DF[mask][level_tags].values.tolist()[0]
title = str(most_freq_cat) + '\n'
title += '\n'.join(cat_levels)
plt.suptitle(title)

most_freq_cat_ids = train_categories_gb.get_group(most_freq_cat)['_id']
max_counter = 50
counter = 0
n = 10
```

```
for item_id in most_freq_cat_ids.values[:max_counter]:
    if counter > 0 and counter % n == 0:
        plt.figure(figsize=(14, 6))

    item = get_item(item_id)

    mask = CATEGORY_NAMES_DF['category_id'] == item['category_id']
    plt.subplot(1, n, counter % n + 1)
    plt.imshow(decode_images(item['imgs']))
    plt.axis('off')

    counter += 1
    if counter == max_counter:
        break
```

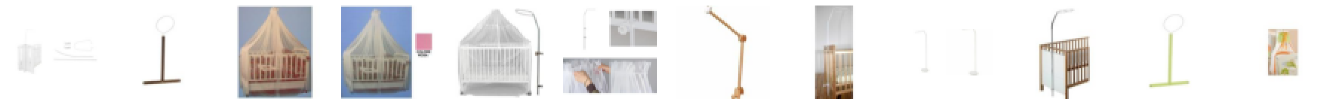
1000018296
MUSIQUE
CD
CD POP ROCK - CD ROCK INDE

```
#display less freq. category
for less_freq_cat in less_freq_cats.index:
    less_freq_cat_ids = train_categories_gb.get_group(less_freq_cat)['_id']
    counter = 0
    n = 12

    plt.figure(figsize=(16, 4))
    mask = CATEGORY_NAMES_DF['category_id'] == less_freq_cat
```

```
    cat_levels = CATEGORY_NAMES_DF[mask][level_tags].values.tolist()[0]
    title = str(less_freq_cat) + '\n'
    title += '\n'.join(cat_levels)
    plt.suptitle(title)

    for item_id in less_freq_cat_ids.values:
        if counter > 0 and counter % n == 0:
            plt.figure(figsize=(16, 4))

        item = get_item(item_id)

        mask = CATEGORY_NAMES_DF['category_id'] == item['category_id']
        plt.subplot(1, n, counter % n + 1)
        plt.imshow(decode_images(item['imgs']))
        plt.axis('off')

        counter += 1
```

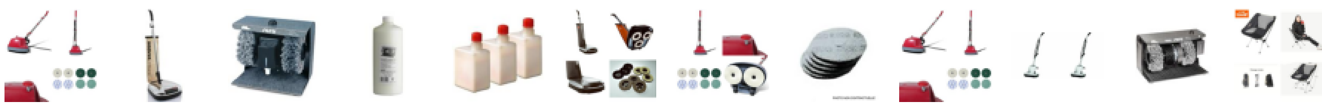
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: RuntimeWarning: More tha
```

1000000522
PUERICULTURE
SOMMEIL BEBE
FLECHE DE LIT BEBE

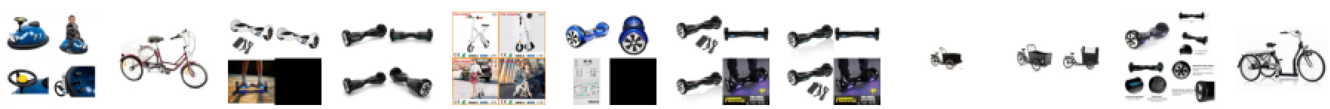


1000000896
EPICERIE
CONSERVE DE LEGUME
POIVRON EN CONSERVE

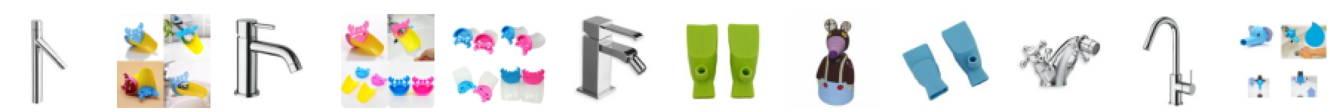
1000003589
ELECTROMENAGER
ENTRETIEN DES SOLS - MAISON
CIREUSE



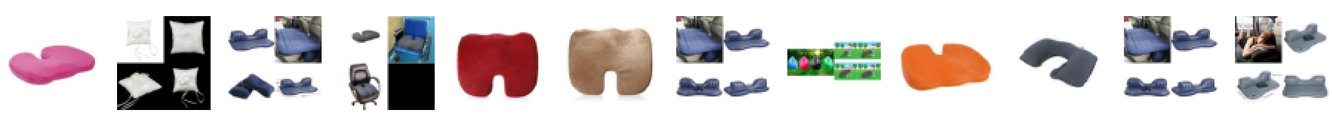
1000007168
SPORT
CYCLES
TRIPORTEUR



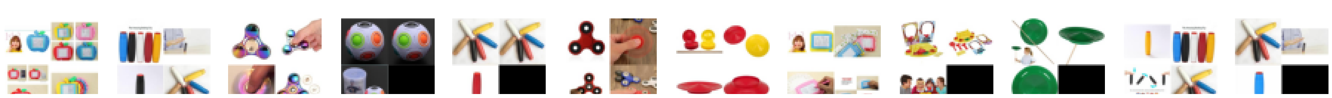
1000007760
PUERICULTURE
TOILETTE BEBE
EXTENSION DE ROBINET



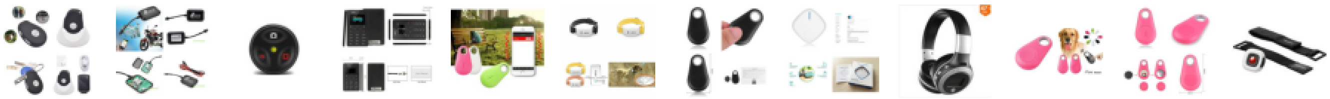
1000008633
CONDITIONNEMENT
CALAGE - PROTECTION
COUSSIN GONFLABLE



1000008894
JEUX - JOUETS
JONGLERIE
ASSIETTE CHINOISE

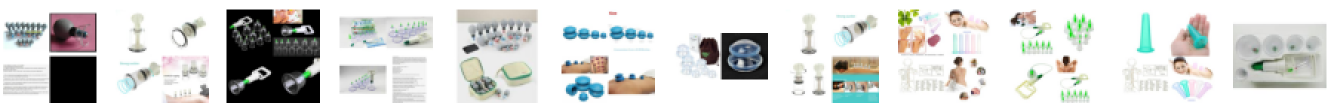


1000010603
TELEPHONIE - GPS
ACCESSOIRE GPS
TELECOMMANDE GPS

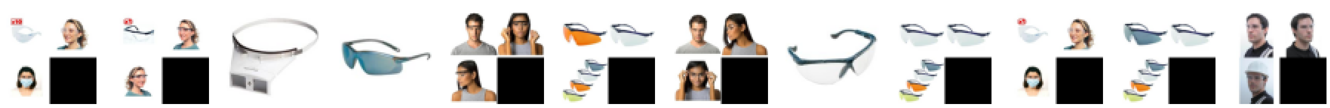


1000010893
PHOTO - OPTIQUE
VISIONNAGE PHOTO
SCANNER DE DIAPOSITIVE

1000011519
MATERIEL MEDICAL
ACUPUNCTURE - MEDECINES PARALELLES
VENTOUSE

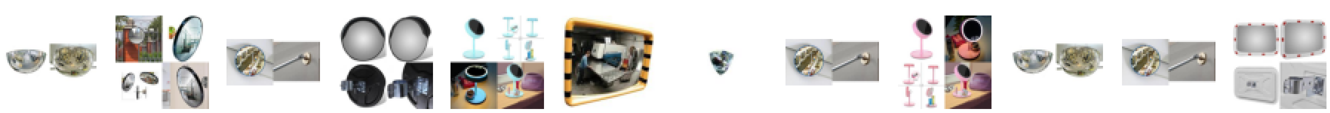


1000011638
MATERIEL MEDICAL
VETEMENTS MEDICAUX
LUNETTES MEDICALES - LUNETTES DE PROTECTION

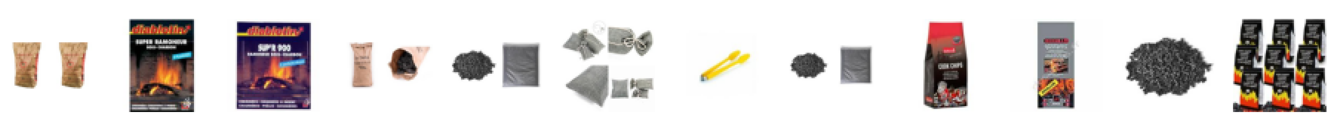


1000011955
MATERIEL MEDICAL
SOIN
CATHETER - OBTURATEUR

1000012074
POINT DE VENTE - COMMERCE - ADMINISTRATION
SECURITE
MIROIR DE SURVEILLANCE



1000012168
DROGUERIE
COMBUSTIBLE
CHARBON DE BOIS



1000012287