September 6, 2019 By Abhisek Jana — Leave a Comment (Edit)

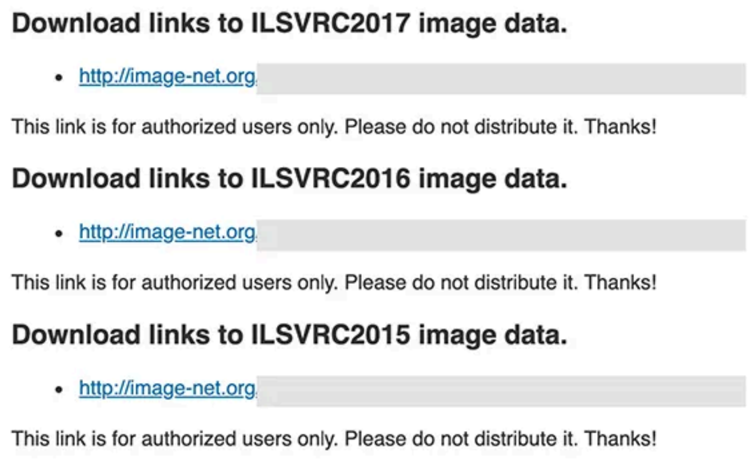
How to prepare Imagenet dataset for Image Classification

Imagenet is one of the most widely used large scale dataset for benchmarking Image Classification algorithms. In case you are starting with Deep Learning and want to test your model against the imagine dataset or just trying out to implement existing publications, you can download the dataset from the imagine website. The downloaded dataset is not human readable, hence In this How to prepare Imagenet dataset for Image Classification tutorial I will explain how you can use this dataset.

How to download imagenet dataset?

Option 1:

You need to have an .edu email address to download directly from the imagenet website. Click on the below website, and login using your .edu email id. You need to register in case you don't have a profile created.
http://image-net.org/download-images
The Dataset has not changed since 2012, I recommend to download from 2017-2015 links. Click on any of the following links:
crane_bird



Download links to ILSVRC2017 image data.

- http://image-net.org[redacted]

This link is for authorized users only. Please do not distribute it. Thanks!

Download links to ILSVRC2016 image data.

- http://image-net.org[redacted]

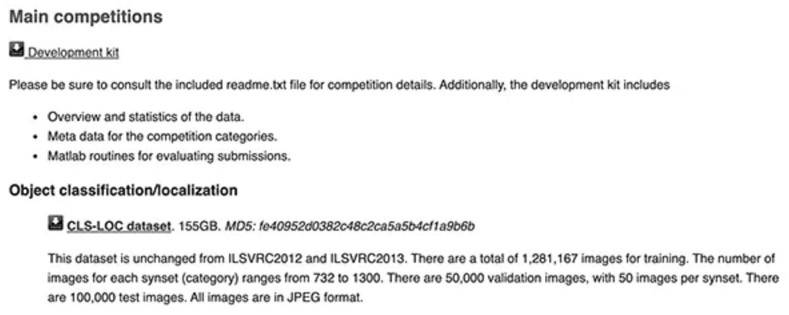
This link is for authorized users only. Please do not distribute it. Thanks!

Download links to ILSVRC2015 image data.

- http://image-net.org[redacted]

This link is for authorized users only. Please do not distribute it. Thanks!

Then download the Development Kit (it has the labels) and the CLS-LOC dataset which is 155GB.



Main competitions

Development kit

Please be sure to consult the included readme.txt file for competition details. Additionally, the development kit includes

- Overview and statistics of the data.
- Meta data for the competition categories.
- Matlab routines for evaluating submissions.

Object classification/localization

CLS-LOC dataset. 155GB. *MD5: fe40952d0382c48c2ca5a5b4cf1a9b6b*

This dataset is unchanged from ILSVRC2012 and ILSVRC2013. There are a total of 1,281,167 images for training. The number of images for each synset (category) ranges from 732 to 1300. There are 50,000 validation images, with 50 images per synset. There are 100,000 test images. All images are in JPEG format.

I strongly recommend using a download manager where you can pause or resume the downloads since its going to take a while based on your internet connection speed.

Option 2:

There might be other ways to download the dataset, without having the .edu email address. My suggestion is to google for finding those options.

Downloaded folder structure:

Once the zip files are downloaded, extract them. If you are using SSD, then extraction will be much faster than HDD.

The train/val/test data will be in following folder:

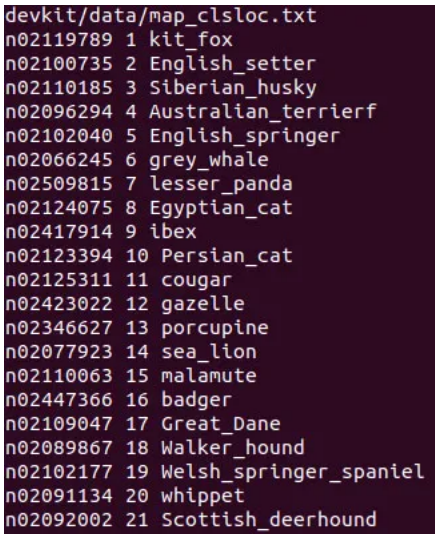
- /ILSVRC2015/Data/CLS-LOC/train
- /ILSVRC2015/Data/CLS-LOC/test
- /ILSVRC2015/Data/CLS-LOC/val

There will be 1000 folders inside the train folder only. However the folder names are not the image labels, for that we need to look into the devkit.

The `/devkit/data/map_clsloc.txt` has the training label and `/devkit/data/ILSVRC2015_clsloc_validation_ground_truth.txt` has the validation labels.

Update the train folders:

What we want to do is, update the folders/dirs inside the train folders with the name of the respective class label. If you view the file, the structure is like following:



```
devkit/data/map_clsloc.txt
n02119789 1 kit_fox
n02100735 2 English_setter
n02110185 3 Siberian_husky
n02096294 4 Australian_terrierf
n02102040 5 English_springer
n02066245 6 grey_whale
n02509815 7 lesser_panda
n02124075 8 Egyptian_cat
n02417914 9 ibex
n02123394 10 Persian_cat
n02125311 11 cougar
n02423022 12 gazelle
n02346627 13 porcupine
n02077923 14 sea_lion
n02110063 15 malamute
n02447366 16 badger
n02109047 17 Great_Dane
n02089867 18 Walker_hound
n02102177 19 Welsh_springer_spaniel
n02091134 20 whippet
n02092002 21 Scottish_deerhound
```

Each row is separated by space, the first column is the name of the train folder and the 3rd one is the mapped label.

We want to make sure the folders name in side the train folders are as per respective images label name. We can easily do that using a simple script.

Read the `map_clsloc.txt` file and create two python map object. The `class_dir_map` will have the current folder name as key and the respective class label as value. The `id_class_map` will have the id for the key and the respective class label as value. We would need the `id_class_map` for validation dataset.

```
import os

MAP_CLASS_LOC = "/media/4TB/datasets/ILSVRC2015/ILSVRC2015/devkit/data/map_clsloc.txt"

class_dir_map = {}
id_class_map = {}

with open(MAP_CLASS_LOC, "rb") as map_class_file:
    rows = map_class_file.readlines()
    for row in rows:
        row = row.strip()
        arr = row.decode("utf-8").split(" ")
        class_dir_map[arr[0]] = arr[2]
        id_class_map[int(arr[1])] = arr[2]

TRAIN_DATA_FOLDER = "/media/4TB/datasets/ILSVRC2015/ILSVRC2015/Data/CLS-LOC_100/train/"
```

Issues with imagenet dataset:

There is an issue with the current dataset, lets fix that before we proceed further.

Issue 1: crane

There are two identical labels named “crane” . One is a bird and another is mechanical crane. We will rename the bird as crane_bird.

```
n02012849 429 crane_bird
```

Issue 2: maillot

maillot has the same problem. We will update them by maillot_1 and maillot_2.

```
n03710637 782 maillot_1
n03710721 977 maillot_2
```

Now we will update the subfolders under train as per their label name.

```
for key in class_dir_map.keys():
    if os.path.isdir(TRAIN_DATA_FOLDER + key):
        os.rename(TRAIN_DATA_FOLDER + key, TRAIN_DATA_FOLDER + class_dir_map[key])
```

Train/Test Split:

Since the labels for the test dataset has not been given, we will 50000 from the train data (50 from each label) and create the test dataset.

Create two list, one containing the path of each image and another their class labels. Then use `sklearn.model_selection.train_test_split()` to create the test dataset. Make sure to set the `stratify=labels`, so that the `train_test_split()` function can distribute the test labels evenly.

```
import glob

files = glob.glob(TRAIN_DATA_FOLDER + "**/*.JPEG")
paths = []
labels = []

for file in files:
    label_str = file.split("/")[-2]
    paths.append(file)
    labels.append(label_str)

from sklearn.model_selection import train_test_split

(trainPaths, testPaths, trainLabels, testLabels) = train_test_split(paths, labels,
test_size=50000, stratify=labels, random_state=42)
```

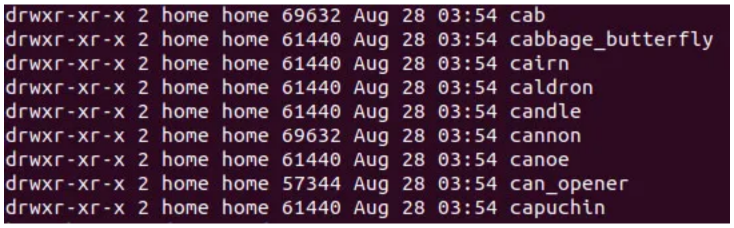
All that is left now is to move the identified test images to a test folder.

```
TEST_DATA_FOLDER = "/media/4TB/datasets/ILSVRC2015/ILSVRC2015/Data/CLS-LOC/test/"

for testPath, testLabel in zip(testPaths, testLabels):

    if not os.path.isdir(TEST_DATA_FOLDER + testLabel):
        os.mkdir(TEST_DATA_FOLDER + testLabel)
    os.rename(testPath, TEST_DATA_FOLDER + testLabel + "/" + testPath.split("/")[-1])
```

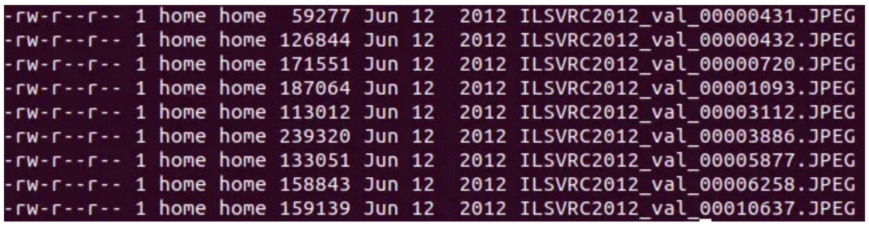
Here is how the train/test folders looks:



```
drwxr-xr-x 2 home home 69632 Aug 28 03:54 cab
drwxr-xr-x 2 home home 61440 Aug 28 03:54 cabbage_butterfly
drwxr-xr-x 2 home home 61440 Aug 28 03:54 cairn
drwxr-xr-x 2 home home 61440 Aug 28 03:54 caldron
drwxr-xr-x 2 home home 61440 Aug 28 03:54 candle
drwxr-xr-x 2 home home 69632 Aug 28 03:54 cannon
drwxr-xr-x 2 home home 61440 Aug 28 03:54 canoe
drwxr-xr-x 2 home home 57344 Aug 28 03:54 can_opener
drwxr-xr-x 2 home home 61440 Aug 28 03:54 capuchin
```

Update the validation folders:

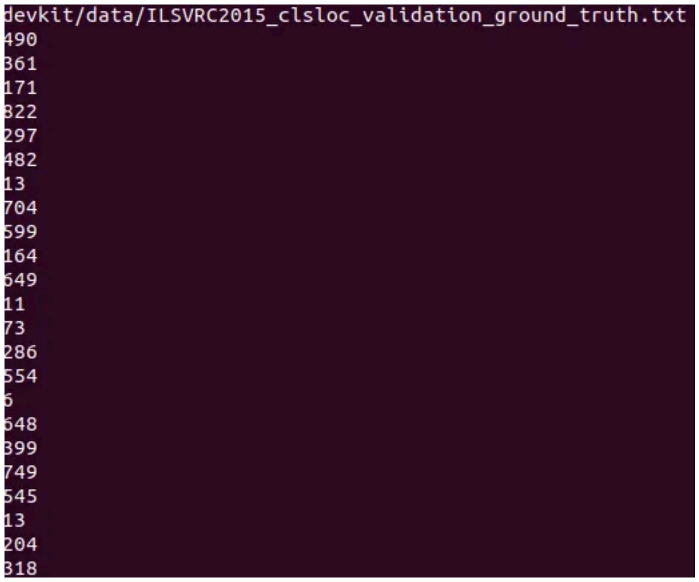
The val folder has only list of images. The name of the files will be like following:



```
-rw-r--r-- 1 home home  59277 Jun 12  2012 ILSVRC2012_val_00000431.JPEG
-rw-r--r-- 1 home home 126844 Jun 12  2012 ILSVRC2012_val_00000432.JPEG
-rw-r--r-- 1 home home 171551 Jun 12  2012 ILSVRC2012_val_00000720.JPEG
-rw-r--r-- 1 home home 187064 Jun 12  2012 ILSVRC2012_val_00001093.JPEG
-rw-r--r-- 1 home home 113012 Jun 12  2012 ILSVRC2012_val_00003112.JPEG
-rw-r--r-- 1 home home 239320 Jun 12  2012 ILSVRC2012_val_00003886.JPEG
-rw-r--r-- 1 home home 133051 Jun 12  2012 ILSVRC2012_val_00005877.JPEG
-rw-r--r-- 1 home home 158843 Jun 12  2012 ILSVRC2012_val_00006258.JPEG
-rw-r--r-- 1 home home 159139 Jun 12  2012 ILSVRC2012_val_00010637.JPEG
```

The last part of the filename is the sequence id of each file. It starts from `00000001`.

The `ILSVRC2015_clsloc_validation_ground_truth.txt` file just has the list of sequence ids for the validation set. So the file with name `ILSVRC2012_val_00000001.JPEG` will have the label by as `490`. The label name of the label id 490 is `sea_snake`. (you can verify by opening the file itself).



```
devkit/data/ILSVRC2015_clsloc_validation_ground_truth.txt
490
361
171
822
297
482
13
704
599
164
649
11
73
286
554
6
648
399
749
545
13
204
318
```

Now, some of the validation images are very difficult to classify, hence they are not used. These files are listed in another file named, `/data/ILSVRC2015_clsloc_validation_blacklist.txt`. The structure of this file is same as the validation ground truth. We need to make sure not to include these in our validation set.

First we need to read the backlist file and store the ids in a list named `black_list`.

```
BLACK_LIST =
"/media/4TB/datasets/ILSVRC2015/ILSVRC2015/devkit/data/ILSVRC2015_clsloc_validation_blacklist.txt
VAL_CLASS_PATH =
"/media/4TB/datasets/ILSVRC2015/ILSVRC2015/devkit/data/ILSVRC2015_clsloc_validation_ground_truth.

VAL_DATA_PATH = "/media/4TB/datasets/ILSVRC2015/ILSVRC2015/Data/CLS-LOC/val/"

VAL_ORI_DATA_PATH = "/media/4TB/datasets/ILSVRC2015/ILSVRC2015/Data/CLS-
LOC/val_original/*.JPEG"

black_list = []

with open(BLACK_LIST) as b_file:
    rows = b_file.readlines()
    for row in rows:
        row = int(row.strip())
        black_list.append(row)
```

Next read the validation ground truth and do the same.

```
val_class = []

with open(VAL_CLASS_PATH) as val_file:
    rows = val_file.readlines()
    for row in rows:
        row = int(row.strip())
        val_class.append(row)
```

Finally, loop through each validation image files,

- Parse the sequence id.
- Make sure its not in the black list.
- Find the class id and class label name.
- Create a folder with the label name in the val directory.
- Move the validation image inside that folder.

```
val_files = glob.glob(VAL_ORI_DATA_PATH)

for file in val_files:
    seq_num = int(file.split("/")[-1].split("_")[-1].split(".")[0])
    if seq_num not in black_list:
        class_id = val_class[seq_num - 1]
        class_name = id_class_map[class_id]

        if not os.path.isdir(VAL_DATA_PATH + class_name):
            os.mkdir(VAL_DATA_PATH + class_name)

        os.rename(file, VAL_DATA_PATH + class_name + "/" + file.split("/")[-1])
```

Create Label Map:

Just for convenience we will store the class label and their NumberEncoder in a JSON file so that we can use that to set the target class. You can also store them as OneHotEncoding here.

```
import json
import glob

label_map = {}

dirs = glob.glob(TRAIN_DATA_FOLDER + "*")
for i, dir in enumerate(dirs):
    label_map[dir.split("/")[-1]] = i

with open("label_map.json", "w") as file:
    file.write(json.dumps(label_map))
```

Here is the part of the json file:

```
{
  "Persian_cat": 0,
  "barracouta": 1,
  "pug": 2,
  "whiskey_jug": 3,
  "pot": 4,
  "cassette": 5,
  "solar_dish": 6,
  "tub": 7,
  "gorilla": 8,
  "microphone": 9,
  "cabbage_butterfly": 10,
  ....
  ....
}
```

Conclusion:

This should get you to the point that you are start your preprocessing. In future I will post another article on how to prepare the imagenet dataset for object detection.

GitHub

Related

Imagenet PreProcessing using TFRecord and Tensorflow 2.0 Data API
In "Computer Vision"

Create a Simple Pie Chart using D3.js
In "D3.js"

Implement Neural Network using PyTorch
In "Data Science"

Filed Under: Computer Vision, Deep Learning | Tagged With: dataset, Deep Learning, image classification, imagenet, Python

Subscribe to stay in loop

* indicates required

Email Address *

Subscribe

Leave a Reply

Logged in as Abhisek Jana. Edit your profile. Log out? Required fields are marked *

Comment *

Post Comment

This site uses Akismet to reduce spam. Learn how your comment data is processed.