2017

# DATASET PREPARATION

NRUPESH PATEL
nrupesh.patel@sjsu.edu
SJSU ID: 011425271

SAN JOSE STATE UNIVERSITY

# REQUIREMENT

Preparing a dataset becomes inevitable because of following reasons:

- Training a CNN for detecting garbage requires a large image dataset.
- GINI Dataset mentioned in current solution is not open dataset.
- There are no open garbage image datasets currently available.

# IMAGE GATHERING

Images were downloaded using BING Image Search API for the following search queries:

| Garbage Queries | Non-Garbage Queries |
|---|---|
| garbage | pattern |
| trash | clean road |
| litter | suburb |
| street litter | city street |
| rubbish | food |
| waste | rural area |
| trash on roads | sky |
| roadside garbage | face |
| medical waste | vehicles |
| california dirty city | crowd |
| market waste | objects |
| footpath waste | chaos |
| garbage n forest | earth dust |
| railway garbage | building |
| street garbage | night sky |
| california dirty streets | countryside |
| city garbage | environment |
| san jose garbage | people |
| kitchen waste | places |
| park litter | clothes |
|  | pattern background |
|  | nature texture |
|  | landscape texture |
|  | california roads |
|  | lights backdrop |
|  | sound waves |
|  | topography |
|  | railway tracks |
|  | chaos cable |
|  | waves |

**Note:** Total of 14,567 images were downloaded using the above queries.

**CODE (Download Images):**

```
import httplib
import urllib
import base64
import json
import sys
import os

queryString = sys.argv[1]

headers = {
    'Ocp-Apim-Subscription-Key': 'ae6e19b20251426681ee1c5a65a21bad',
}

offset = 0
if not os.path.exists(queryString):
    os.makedirs(queryString)

while(True):
    params = urllib.urlencode({
        # Request parameters
        'q': queryString,
        'count': '150',
        'offset': offset,
        'mkt': 'en-us',
    })

    try:
        conn = 
httplib.HTTPSConnection('api.cognitive.microsoft.com')
        conn.request("GET", "/bing/v5.0/images/search?%s" % params, 
"{body}", headers)
        response = conn.getresponse()
        data = response.read()
        data_parse = json.loads(data)
        for i in range(0, 150):
            print "Downloading Image " + str(offset)
            offset = offset + 1
            urllib.urlretrieve(data_parse['value'][i]['contentUrl'], 
queryString + "/" + data_parse['value'][i]['imageId']+ "." + 
data_parse['value'][i]['encodingFormat'])
        conn.close()
    except IndexError:
        break
    except Exception as e:
        print (e)
        offset = offset + 1
```

# IMAGE ANNOTATION

Images downloaded above could not be used directly to train CNN because:

- Regions of the image containing garbage had to be extracted for training the CNN. This is because using the entire image as an example of garbage, when only a portion of it corresponds to garbage, can affect the training process.

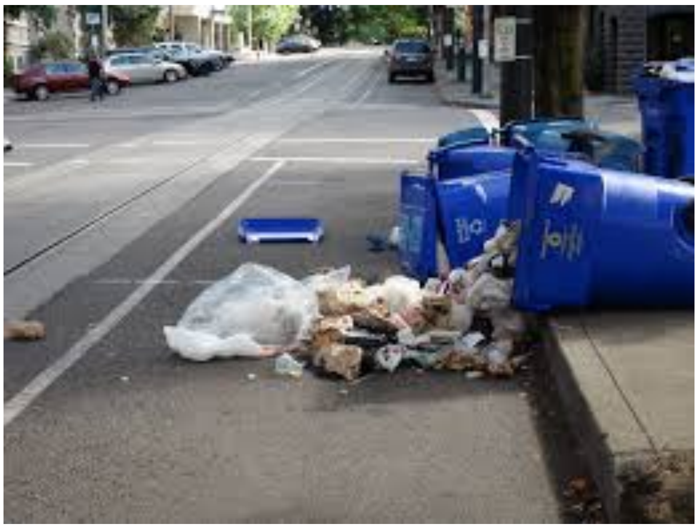

**Figure:** Image with only portion of garbage.

- Some of the garbage related queries resulted in images that did not contain garbage, but were 'figuratively' related to garbage. These images cannot be used for training the CNN with the label as garbage.

Less
Litter.
Live
Better.
EPA

**Figure:** Image which only related to garbage and is not actually garbage.

So, user annotations had to be taken for the downloaded images whether they were garbage or not.

Fast Image Data Annotation Tool is used to give user annotations to various images.

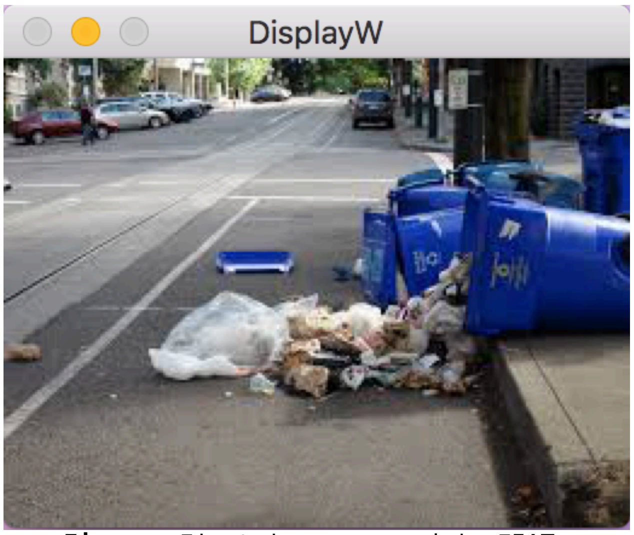

**Figure:** First image opened in FIAT.

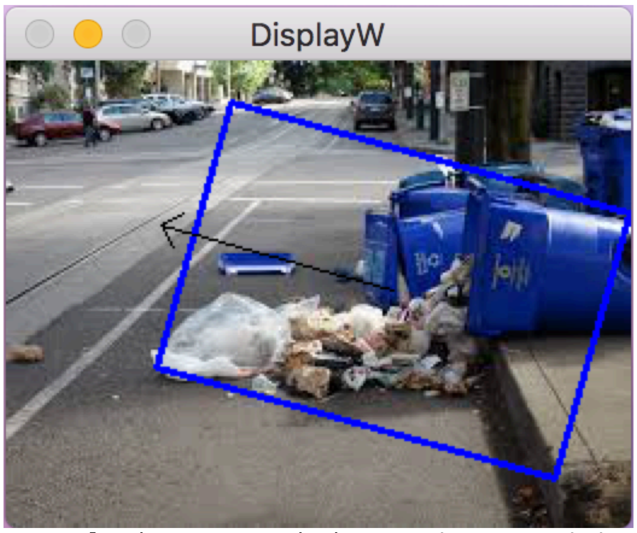

**Figure:** Rectangle drawn around the portion containing garbage.

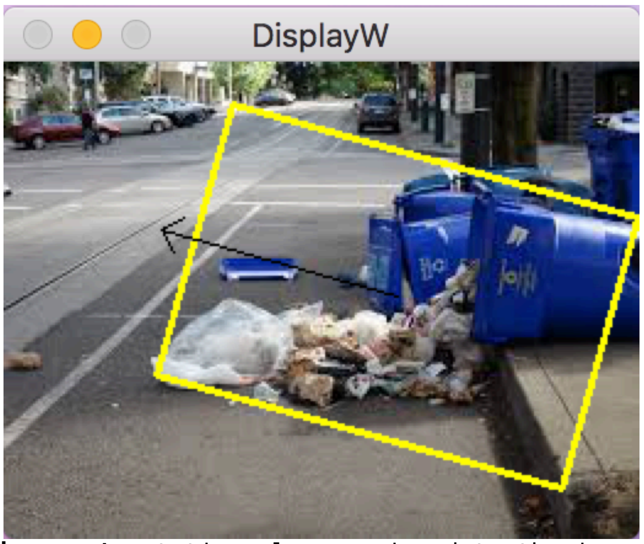

**Figure:** Annotation class assigned to the image.

# IMAGE CLASSIFICATION

Images here are classified into two categories:

- GARBAGE
- NOT GARBAGE

For further Caffe dataset creation we will need two files: **train.txt** and **val.txt.** They will contain paths to images and class number from train and test data respectively.

Part of file can look like following:

```
...
0_garbage31.jpg 0
1_no-garbage45.jpg 1
1_no-garbage98.jpg 1
0_garbage12.jpg 0
0_garbage671.jpg 0
0_garbage92.jpg 0
1_no-garbage111.jpg 1
....
```

**CODE (Generate train.txt and val.txt):**

```
import os
import numpy as np
import time
import shutil
import cPickle
import random

from PIL import Image, ImageOps


def chunkIt(seq, num):
    avg = len(seq) / float(num)
    out = []
    last = 0.0

    while last < len(seq):
        out.append(seq[int(last):int(last + avg)])
        last += avg

    return out


def shuffle_in_unison(a, b):
    assert len(a) == len(b)
    shuffled_a = np.empty(a.shape, dtype=a.dtype)
    shuffled_b = np.empty(b.shape, dtype=b.dtype)
    permutation = np.random.permutation(len(a))
    for old_index, new_index in enumerate(permutation):
        shuffled_a[new_index] = a[old_index]
        shuffled_b[new_index] = b[old_index]
    return shuffled_a, shuffled_b


def move_files(input, output):
    '''
        Input: folder with dataset, where every class is in separate
folder
        Output: all images, in format class_number.jpg; output path
should be absolute
    '''
    index = -1
    for root, dirs, files in os.walk(input):
        path = root.split('/')
        print 'Working with path ', path
        print 'Path index ', index
        filenum = 0
        for file in files:
            fileName, fileExtension = os.path.splitext(file)
            if fileExtension == '.jpg' or fileExtension == '.JPG' or
fileExtension == '.jpeg':
                full_path = path[0] + '/' + path[1] + '/' + file
                print full_path
                if (os.path.isfile(full_path)):
                    file = str(index) + '_' + path[1] + str(filenum)
+ fileExtension
                    print output + '/' + file
                    shutil.copy(full_path, output + '/' + file)
                filenum += 1
        index += 1


def create_text_file(input_path, outpath, percentage):
    '''
        Creating train.txt and val.txt for feeding Caffe
    '''

    images, labels = [], []
    os.chdir(input_path)

    for item in os.listdir('.'):
        if not os.path.isfile(os.path.join('.', item)):
            continue
        try:
            label = int(item.split('_')[0])
            images.append(item)
            labels.append(label)
        except:
            continue
```

```python
    images = np.array(images)
    labels = np.array(labels)
    images, labels = shuffle_in_unison(images, labels)

    X_train = images[0:len(images) * percentage]
    y_train = labels[0:len(labels) * percentage]

    X_test = images[len(images) * percentage:]
    y_test = labels[len(labels) * percentage:]

    os.chdir(outpath)

    trainfile = open("train.txt", "w")
    for i, l in zip(X_train, y_train):
        trainfile.write(i + " " + str(l) + "\n")

    testfile = open("val.txt", "w")
    for i, l in zip(X_test, y_test):
        testfile.write(i + " " + str(l) + "\n")

    trainfile.close()
    testfile.close()

def main():
    caffe_path = 'input'
    new_path = 'output'
    move_files(caffe_path, new_path)
    create_text_file(new_path, './', 0.85)

if __name__ == "__main__":
    main()
```

# LMDB CREATION

To feed Caffe with large images dataset it's good choice to use LMDB format for our dataset.

```
#!/usr/bin/env sh
# Create the imagenet lmdb inputs
# N.B. set the path to the imagenet train + val data dirs
set -e

EXAMPLE=<path_to_store_lmdb>
DATA=<folder_with_train.txt_val.txt>
TOOLS=build/tools

TRAIN_DATA_ROOT=/path/to/train/
VAL_DATA_ROOT=/path/to/val/

# Set RESIZE=true to resize the images to 256x256. Leave as false if
images have
# already been resized using another tool.
RESIZE=true
if $RESIZE; then
  RESIZE_HEIGHT=256
  RESIZE_WIDTH=256
else
  RESIZE_HEIGHT=0
  RESIZE_WIDTH=0
fi

if [ ! -d "$TRAIN_DATA_ROOT" ]; then
  echo "Error: TRAIN_DATA_ROOT is not a path to a directory:
$TRAIN_DATA_ROOT"
  echo "Set the TRAIN_DATA_ROOT variable in create_imagenet.sh to
the path" \
       "where the ImageNet training data is stored."
  exit 1
fi

if [ ! -d "$VAL_DATA_ROOT" ]; then
  echo "Error: VAL_DATA_ROOT is not a path to a directory:
$VAL_DATA_ROOT"
  echo "Set the VAL_DATA_ROOT variable in create_imagenet.sh to the
path" \
       "where the ImageNet validation data is stored."
  exit 1
fi

echo "Creating train lmdb..."

GLOG_logtostderr=1 $TOOLS/convert_imageset \
    --resize_height=$RESIZE_HEIGHT \
    --resize_width=$RESIZE_WIDTH \
    --shuffle \
    $TRAIN_DATA_ROOT \
    $DATA/train.txt \
    $EXAMPLE/ilsvrc12_train_lmdb

echo "Creating val lmdb..."

GLOG_logtostderr=1 $TOOLS/convert_imageset \
    --resize_height=$RESIZE_HEIGHT \
    --resize_width=$RESIZE_WIDTH \
    --shuffle \
    $VAL_DATA_ROOT \
    $DATA/val.txt \
    $EXAMPLE/ilsvrc12_val_lmdb

echo "Done."
```

## References

[1] Mittal, Gaurav, et al. "SpotGarbage: smartphone app to detect garbage using deep learning." *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 2016.

[2] https://medium.com/@alexrachnog/using-caffe-with-your-own-dataset-b0ade5d71233#.m03nhhn5b

[3] https://github.com/christopher5106/FastAnnotationTool

[4] http://www.pyimagesearch.com/2016/11/28/macos-install-opencv-3-and-python-2-7/

[5] https://www.quora.com/How-do-I-fine-tune-a-Caffe-pre-trained-model-to-do-image-classification-on-my-own-dataset

[6] https://frankzliu.com/experimenting-with-different-penultimate-layers-in-caffe/

[7] http://rodriguezandres.github.io/2016/04/28/caffe/#dataset-preparation