HW2_part2_get_similarity

February 20, 2019

0.1 HW2 Solution Part 2

step1: get keypoints for each images
step2: get patches for each keypoints
step3: get descriptions for each keypoints (by forward pass patches to the network)
step4: caculate the similarity matrices
step5: get topK similar images for each query
step6: draw recall vs precision curves

```
In [1]: import os
        os.environ["CUDA_VISIBLE_DEVICES"] = "1"
```

```
In [2]: # load packages
        from __future__ import division, print_function
        import glob
        import os
        import cv2
        import PIL
        import random
        import numpy as np
        # import pandas as pd
        import matplotlib.pyplot as plt
        import matplotlib.image as mpimg
        import torch
        import torch.nn.init
        import torch.nn as nn
        import torch.optim as optim
        import torch.backends.cudnn as cudnn
        import torch.nn.functional as F
        import torchvision.datasets as dset
        import torchvision.transforms as transforms
        from tqdm import tqdm
        from torch.autograd import Variable
        from copy import deepcopy, copy
        from Utils import cv2_scale36, cv2_scale, np_reshape, np_reshape64
        from scipy.optimize import linear_sum_assignment
```

In [3]:
```
# parameters setting
query_num = 35
image_num = 140
kps_num = 30
```

In [4]:
```
des_dir = "des.pt"
[des1, des2] = torch.load(des_dir)
# torch.save([des1, des2], des_dir)
```

In [5]:
```
def getCost_one2one(des1, des2):
    # des1 = 30 x 128
    # des2 = 30 x 128
    cost = torch.zeros(30, 30)
    simi = torch.zeros(30, 30)
    for i in range(30):
        for j in range(30):
            cost[i, j] = torch.dist(des1[i], des2[j], 2)
            simi[i, j] = torch.exp(-1 * cost[i, j])
    row_ind, col_ind = linear_sum_assignment(cost.cpu().numpy())

    return simi[row_ind, col_ind].sum()

def getCost_many2many(des1, des2, threshold=0.053):
    # des1 = 30 x 128
    # des2 = 30 x 128
    simi = torch.zeros(30, 30)
    for i in range(30):
        for j in range(30):
            simi[i, j] = torch.exp(-1*torch.dist(des1[i], des2[j], 2))
    simiNorm = torch.sqrt(torch.sum(simi * simi))
    simi_w = simi / simiNorm
    threshold = max(threshold, (simi_w.max() - simi_w.min())/2)
    simi_w = simi_w.gt(threshold).float()

    return (simi*simi_w).sum()
```

In [6]:
```
# generate similarity matrices
similarity_one2one = torch.zeros(query_num, image_num)
similarity_many2many = torch.zeros(query_num, image_num)
for idx in range(query_num):
    print("finished No.{} query".format(idx+1))
    for jdx in range(image_num):
        similarity_one2one[idx,jdx] = getCost_one2one(des1[idx], des2[jdx])
        similarity_many2many[idx, jdx] = getCost_many2many(des1[idx], des2[jdx])
```

```
finished No.1 query
finished No.2 query
finished No.3 query
finished No.4 query
```

```
finished No.5 query
finished No.6 query
finished No.7 query
finished No.8 query
finished No.9 query
finished No.10 query
finished No.11 query
finished No.12 query
finished No.13 query
finished No.14 query
finished No.15 query
finished No.16 query
finished No.17 query
finished No.18 query
finished No.19 query
finished No.20 query
finished No.21 query
finished No.22 query
finished No.23 query
finished No.24 query
finished No.25 query
finished No.26 query
finished No.27 query
finished No.28 query
finished No.29 query
finished No.30 query
finished No.31 query
finished No.32 query
finished No.33 query
finished No.34 query
finished No.35 query
```

In [7]:
```
# save similarity matrices
similarity_one2one_dir = "similarity_one2one.pt"
similarity_many2many_dir = "similarity_many2many.pt"
torch.save(similarity_one2one, similarity_one2one_dir)
torch.save(similarity_many2many, similarity_many2many_dir)
```