CS 537 Computer vision Final Project
Yu-Hsiang Liu
June, 2020
Email: liuyuhs@oregonstate.edu

Scene binary classification

Problem statement

Using a given scene dataset (including coast and forest) to build a classifier that can distinguish whether the input image is coast or forest.

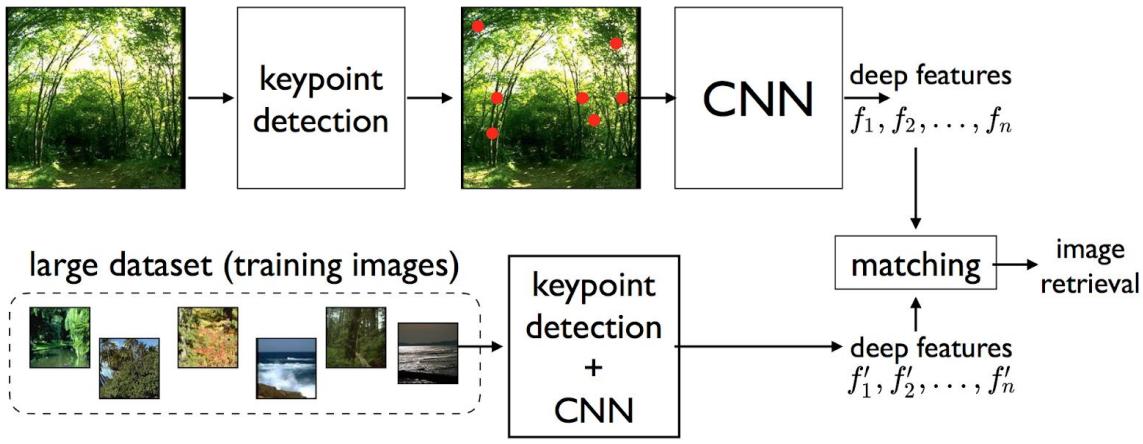
Challenge:
It only has a small dataset for training. The challenge is that a small data set might lead to overfitting. Also, some testing images are similar to others, forest with a lake, for example.
In this experiment, it will also compare the difference size of the dataset.

After finishing scene binary classification, it could also expand to multi-classification. In addition, the technology can be used in many fields. For example, given the CT scan of a patient, the machine can output the disease of the patient. In biology, it can help scientists distinguish different species by the image.

Approach

Overview

1. Keypoint detection
2. For each Keypoint, Compute a deep feature by CNN
3. Feature matching between the query image and training dataset
4. Image retrieval



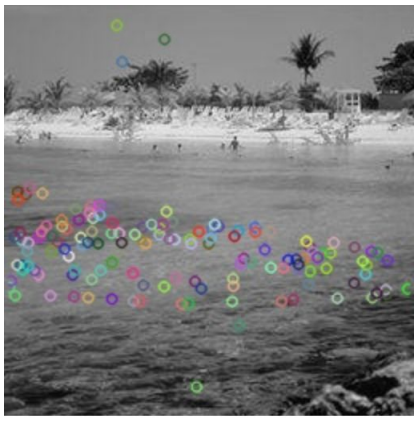
Detail

Given an image of scene (256 x 256), determine whether it is coast or forest.

Input format: scene image (256 x 256)
Output format: “coast” or “forest”

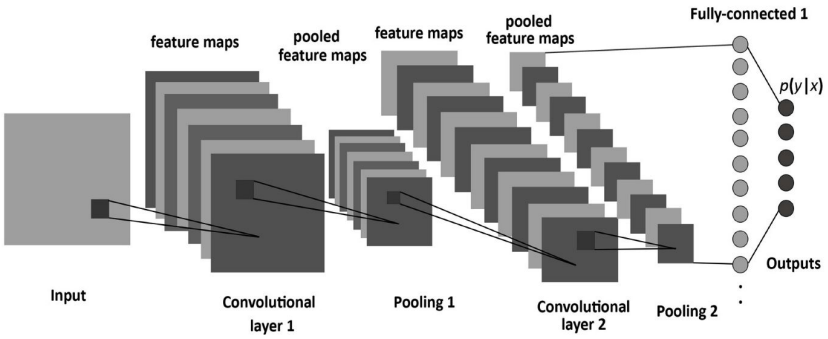
1. Keypoint detection

   In each image, detect SIFT keypoints.
   Save the keypoints as coordinate (x, y).



2. Compute a deep feature by CNN

   Input the keypoints and extract patches around each keypoint.
   Input image patches to CNN for computing the corresponding deep feature of the patches.



3. Using one-one matching for feature matching

   Given test image feature and train image feature

Test image data = { f_1, f_2, …, f_n }
Train image data = { f'_1, f'_2, ..., f'_n }

Minimizes the total cost of matching

$$\hat{M} = \min_{M \in \mathcal{M}} \sum_{(\mathbf{f},\mathbf{f}') \in M} d(\mathbf{f}, \mathbf{f}')$$

4. Calculating the similarity for image retrieval

   Similarity = 1 - cost

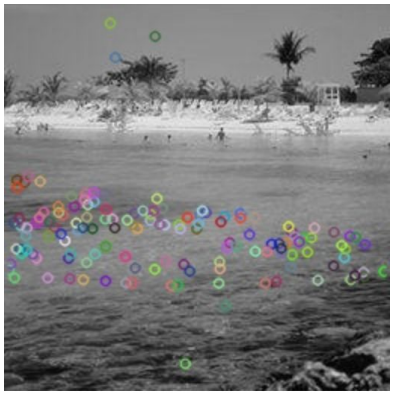
   If $S_{coast} > S_{forest}$,then query image = coast
   If $S_{coast} < S_{forest}$,then query image = forest

Evaluation

Implementation

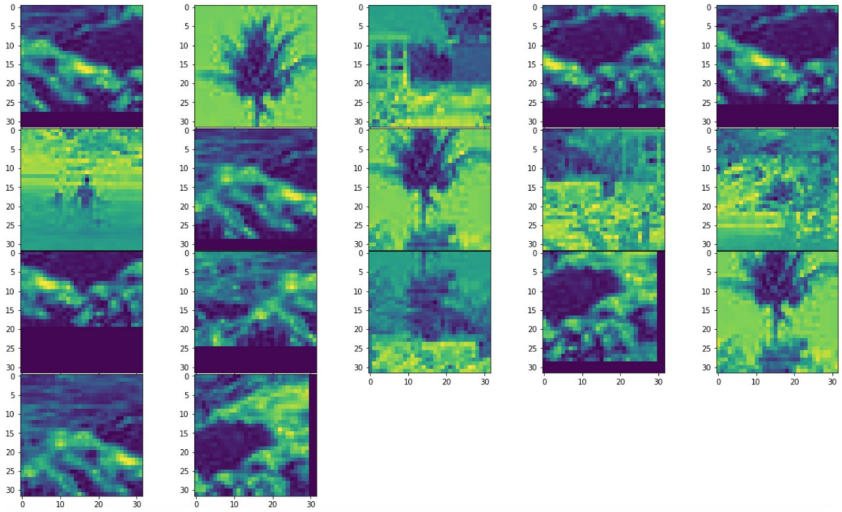
1. In each image, detect SIFT keypoints.

   Keypoints size = torch.Size([100, 17, 2])



2. Extract patches

   Patches size = torch.Size([100, 17,1, 32, 32])



3. Compute a 128-dimensional deep feature by CNN

   The CNN layer DesNet:

```
DesNet(
  (features): Sequential(
    (0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=False, track_running_stats=True)
    (2): ReLU()
    (3): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=False, track_running_stats=True)
    (5): ReLU()
    (6): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (7): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=False, track_running_stats=True)
    (8): ReLU()
    (9): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (10): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=False, track_running_stats=True)
    (11): ReLU()
    (12): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (13): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=False, track_running_stats=True)
    (14): ReLU()
    (15): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (16): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=False, track_running_stats=True)
    (17): ReLU()
    (18): Dropout(p=0.3, inplace=False)
    (19): Conv2d(128, 128, kernel_size=(8, 8), stride=(1, 1), bias=False)
    (20): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=False, track_running_stats=True)
  )
)
```

   Feature size = torch.Size([100, 17, 128])

```
tensor([[ 0.2054,  0.0652, -0.0765,  ..., -0.1591,  0.0253,  0.0039],
        [ 0.1968, -0.0884,  0.1480,  ..., -0.0584,  0.0297, -0.1203],
        [ 0.0289, -0.0344,  0.0172,  ..., -0.0729, -0.0649, -0.0263],
        ...,
        [ 0.1333, -0.0035,  0.2584,  ..., -0.0067, -0.0468, -0.0510],
        [ 0.1174, -0.1252,  0.0924,  ..., -0.0715, -0.0417,  0.0268],
        [ 0.0725, -0.0559, -0.0266,  ..., -0.0559, -0.0382,  0.0372]])
```

4. Using one-one matching for feature matching

   Finding the matching matrix M by hungarian algorithm.

5. Calculating the total similarity for image retrieval

   S_{coast} = 487.49
   S_{forest} = 462.13

   → coast

6. Output the result of classifier compare to groundtruth.txt (0 represent False Positive, 1 represent True Positive)

   [0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Library

Detect the SIFT keypoints using the OpenCV Python library

Training CNN model with PyTorch library, (e.g. torch.nn)

One-One Feature Matching using scipy.optimize.linear_sum_assignment

Set parameters

1. Detect 17 SIFT keypoints in the image.
   (The reason that using 17 is because after deleting all the duplicate keypoints that are found by SIFT, coast_train_44 only has 17 keypoints)
2. Extract 32 x 32 patches from each keypoint.
3. 128-dimensional deep feature
4. CNN parameter:
   Learning rate: 1; Epoch: 20; CNN Layer: 3; Optimizer : sgd

Dataset

1. 2 classes : coast and forest.

2. Number of images:
   50 coast train images 25 coast test images
   50 forest train images 25 forest test images

3. Groudtruth.txt:

```
0 coast
1 coast
2 coast
3 coast
4 coast
5 coast
6 coast
7 coast
8 coast
9 coast
10 coast
```

   Test image ground truth will be combined to a vector, which size = 50.
   Index 0 ~ 24 = coast, 25~49 = forest

Evaluation metrics

In this project, we will use precision to evaluate the binary classifier

	Condition positive (CP)	Condition negative (CN)
Test outcome positive (OP)	True positive	False positive
Test outcome negative (ON)	False negative	True negative

Precise : True Positive / True Positive + False Positive

More versions of approach

In the implementation, we will reduce the number of keypoints and the dataset size to generate different versions of the approach.
The experiment will compare the result with different number of keypoints and different dataset size.
We will test data size = 20, 50, 100, number of keypoints = 5, 10, 17.

Result Table

(Datasize, number of keypoints)

Version	(20, 17)	(50, 17)	(100, 17)	(100, 10)	(100, 5)
Precise	52%	66%	84%	78%	62%

The biggest dataset and number of keypoints will get best performance.
By the way, the dataset size has more impact on the performance than the number of keypoints. 52% seems high, but actually it is bad because the baseline of this experiment is 50% (binary classification)

Training runtime & Test runtime

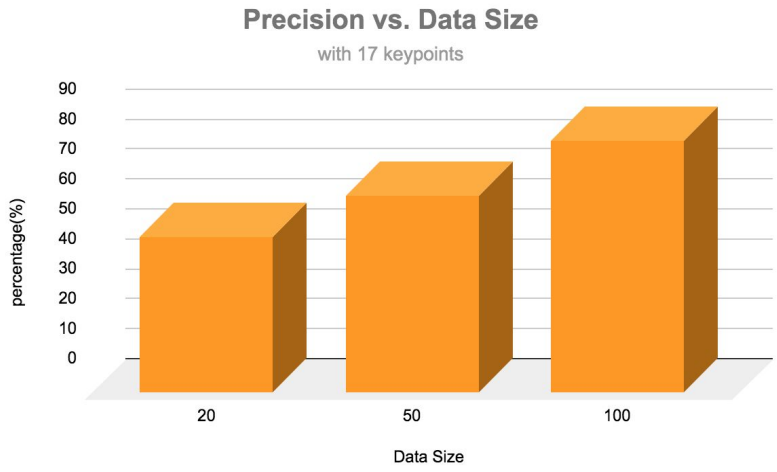
Hardware information:

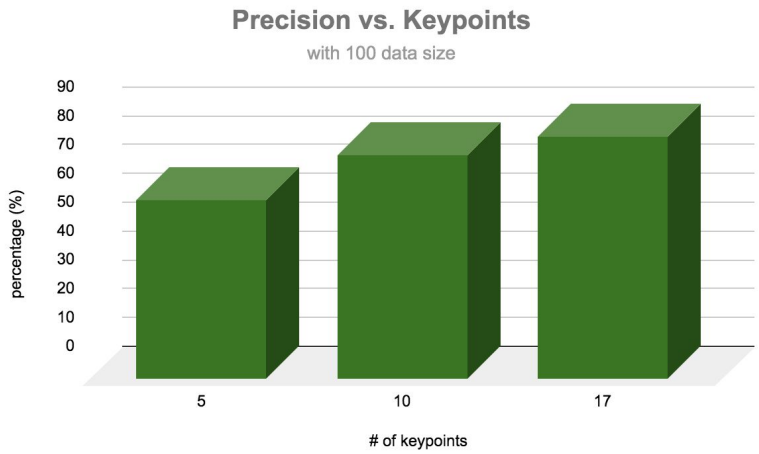
Intel(R) Xeon(R) CPU ES-2640 v3 @ 2.60GHz (pelican Server)

For training CNN model, it took about 1minute per epoch

For testing, it took less than 10 second per image.

Qualitative evaluation





From the result of 100 dataset size and 17 keypoints, the classifier gets 100% for forest, that means all the error is from the coast. It is because the training of CNN models is still not enough to handle the features of the coast. Increasing the coast image might increase the precision.

Labor Distribution

Individual work.
Note: All work is in src.tar.gz