Final Project Report

Qingxiao Yuan

(1) Problem statement

In this final project, I work on the scene binary classification problem. Specifically, if we are given an image of the scene (256 x 256), there is a coast or forest in it with a complex background. What I need to do is to determine whether it is coast or forest by using what we learnt in this course.

The main challenge is that the small number of the dataset may lead to overfitting. The reason is that I used a relatively small training and test set. Moreover, some scenes may be pretty similar or even contain each other. For example, there may be a lake in a forest, or a lake with forest shadow, which increases the test burden.

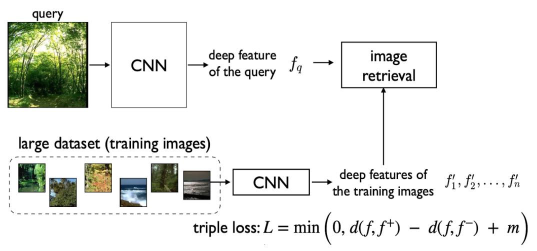
This project is important. It exists everywhere since there are always multiple models performing different functions and exchanging information back and forth in the real-world. In the computer vision related area, there are many applications. It is widely used in the cutting edge medical area. For example, by distinguishing the X ray photos taken for the patient, it can be monitored whether the patient's tumor is positive or negative. Another application is to detect the spam email through the picture content, which uses keypoint detection.

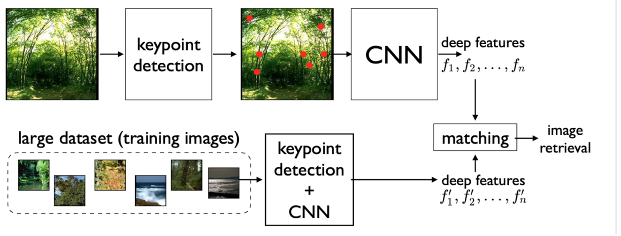
This problem is not only related to the medical research area, but also interdisciplinary with the basic application of machine learning and deep learning.

(2) Approach

First, partition the image into patches, apply CNN to each patch to compute deep features of the patches; then match the patches. Second, to each pair (query, image) apply a Siamese CNN to the entire images and estimate their similarity, then train the Siamese CNN using the triplet loss;

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


query
CNN
deep feature of the query
f_q
image retrieval
large dataset (training images)
CNN
deep features of the training images
$f'_1, f'_2, \dots, f'_n$
triple loss: $L = \min\left(0, d(f,f^+) - d(f,f^-) + m\right)$



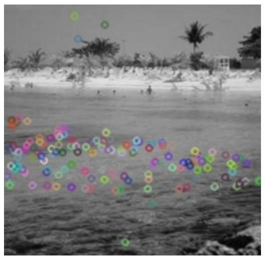
Detail:

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

Input format: scene image (256 x 256).

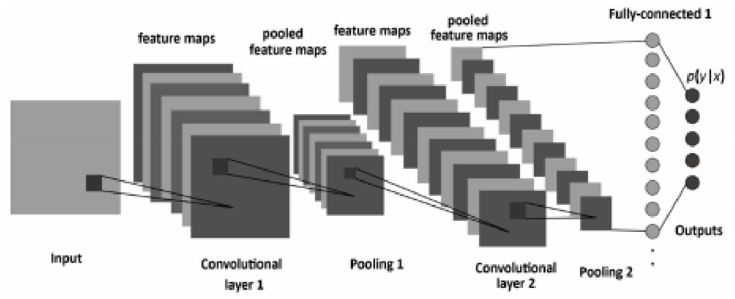
The expected output format is: “coast” or “forest” .

First, detect the SIFT keypoints of images. Meanwhile, save the relative keypoints of the coordinate (x, y).



Second, use CNN to compute a deep feature

Specifically, input keypoints and extract patches. Input image patches to CNN and compute the patches’ deep feature.



Third, use one-one matching to do the feature matching.

Specifically, the test image feature and train image feature was given. Train image data = { f1 , f2 , …, fn } and the test image data = { f’1 , f’2 , ..., f’n }. Then, minimize the matching total cost.

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

Four, calculate the similarity for image retrieval

Similarity = 1 - cost

If $S_{coast} > S_{forest}$,then query image is a coast

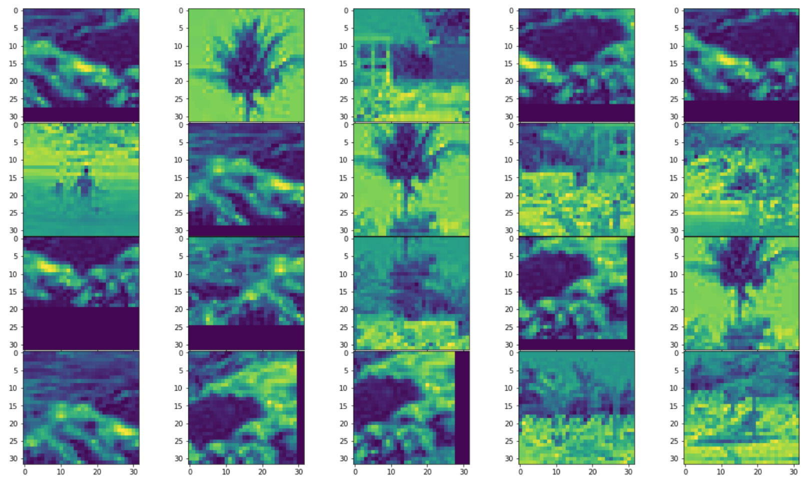
If $S_{coast} < S_{forest}$,then query image is a forest

(3) Evaluation

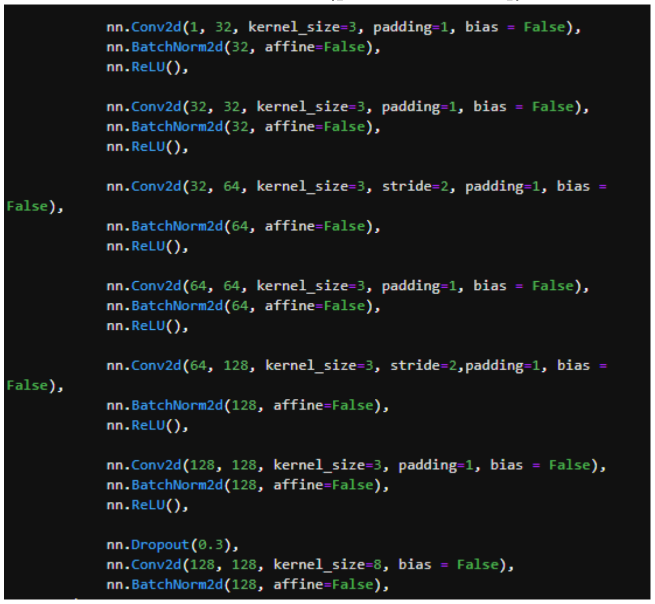
Implementation details

1. In each image, detect SIFT keypoints, size is torch.Size([100, 20, 2])

2. Extract patches, size is torch.Size([100, 20,1, 32, 32])



3. Use the CNN3 to compute the 128-dimensional deep feature
Feature size = torch.Size([100, 20, 128]) and the CNN3 layer is following:



```
            nn.Conv2d(1, 32, kernel_size=3, padding=1, bias = False),
            nn.BatchNorm2d(32, affine=False),
            nn.ReLU(),

            nn.Conv2d(32, 32, kernel_size=3, padding=1, bias = False),
            nn.BatchNorm2d(32, affine=False),
            nn.ReLU(),

            nn.Conv2d(32, 64, kernel_size=3, stride=2, padding=1, bias =
False),
            nn.BatchNorm2d(64, affine=False),
            nn.ReLU(),

            nn.Conv2d(64, 64, kernel_size=3, padding=1, bias = False),
            nn.BatchNorm2d(64, affine=False),
            nn.ReLU(),

            nn.Conv2d(64, 128, kernel_size=3, stride=2,padding=1, bias =
False),
            nn.BatchNorm2d(128, affine=False),
            nn.ReLU(),

            nn.Conv2d(128, 128, kernel_size=3, padding=1, bias = False),
            nn.BatchNorm2d(128, affine=False),
            nn.ReLU(),

            nn.Dropout(0.3),
            nn.Conv2d(128, 128, kernel_size=8, bias = False),
            nn.BatchNorm2d(128, affine=False),
```

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} = 454.57
S_{forest} = 432.65
=>coast

6. Output the result of classifier compare to groundtruth.txt

0: False Positive,
1: True Positive
[1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

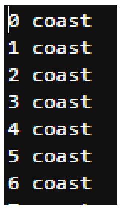
Library:
Detect the SIFT keypoints: OpenCV
Training CNN: PyTorch
One-One Feature Matching: scipy.optimize.linear_sum_assignment

Parameters setting:
1. Detects 20 SIFT keypoints.
Since the number 20 is the maximum value after deduplication.
2. Extract 32 x 32 patches from each keypoint.
3. 128-dimensional deep feature
4. CNN parameter: Learning rate: 0.3; Epoch: 80; CNN Layer: 3; Optimizer : sgd

Datasets

https://drive.google.com/drive/folders/1fjCtMBwbhGsbQAq1QI71rmAe8TinTCTy?usp=sharing

In ground_truth.txt: 0-24 represents the coast, 25-49 stands for the forest.

e.g. 

```
0 coast
1 coast
2 coast
3 coast
4 coast
5 coast
6 coast
```

2 classes : coast and forest.

25 coast test images, 50 coast train images, 25 forest test images, and 50 forest train images.

Evaluation metrics

	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 numbers of keypoints and different dataset sizes. We will test data size = 30, 60, 100, number of keypoints = 5, 10, 20.

Result Table

Datasize, number of keypoints

Version	(30,20)	(60,20)	(100,20)	(100,10)	(100,5)
Precise	64%	68%	81%	74%	66%

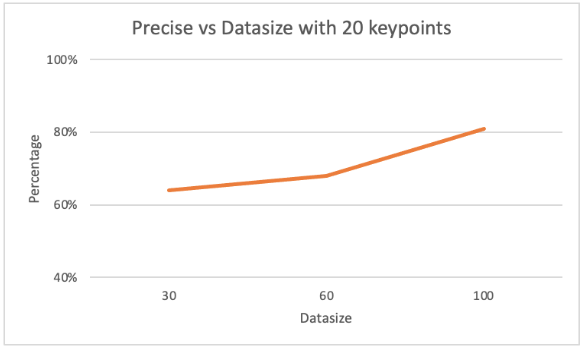
The best performance happens when the "dataset and number of keypoint" is the largest. Since I used the intersection over union to evaluate the result. The higher, the better. Moreover, 64% should not be considered a good result, since the result is just a little bit higher than the baseline, which is 60%.

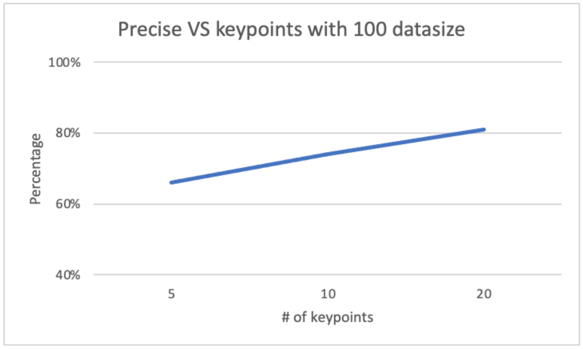
Training runtime & Test runtime Hardware information:

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

Training CNN model: around 50 sec per epoch; Testing runtime: less than 13 sec per image

Qualitative evaluation





From the results of 100 data set sizes and 20 key points, the accuracy of the forest has reached 100%, but the coast has not. This may be because the data set is relatively small, and the training of the CNN model is not enough to handle the characteristics of the coast. In other words, the key to the data lies in the extraction of coastal features. Therefore, adding coastal images to make the model more intelligent may improve the accuracy.

(4) Team work or individual

This project is Individual work, without any team effort. What I refer to is in the following reference list. Specifically, I did the project by referring to what I learnt in computer vision and some part of the homework ([1] & [2]). The program also refers to the python API [3]. Additionally, the topic choice and definition understanding refers to wikipedia [4]. I downloaded the first and coast dataset from the Internet [5] and dealt with them with the picresize website [6].

References

[1] Prof. Sinisa Todorovic. CS537_FinalProjects.pdf

[2] Prof. Sinisa Todorovic. CS537_7.pdf

[3] *programcreek.* https://www.programcreek.com/python/example/97225/scipy.optimize.linear_sum_assignment

[4] Binary classification. *Wikipedia.* https://en.wikipedia.org/wiki/Binary_classification

[5] Forest and coast Dataset. https://unsplash.com/s/photos/forest

[6] *Internich*. https://picresize.com/en/batch