CS4186

Assignment-1 Report

Zhumakhan Arman

SID: 56027756

Introduction

In the assignment, I used three methods Scale Invariant Feature Transform (SIFT) and Convolutional Neural Network (CNN), and combined method.

SIFT is a feature detection algorithm that was developed by David Lowe [1]. SIFT can extract keypoints and compute their descriptors; then we can match points of one image with points of another image, using Brute-Force matching.

CNN takes an image as input and computes features through several layers. Large Deep Learning models have the last layer called the “softmax” layer that uses the features of the previous layer and computes the probability of each defined item. By removing the last layer, we can use CNN models to generate feature vectors (embeddings). I used the pre-trained ResNet-50 model to extract embeddings of images, then find similar images using Euclidean distance between embeddings.

Methods

SIFT

SIFT locates keypoints and gives descriptors of these points. Each descriptor consists of 128 values. This method went through all images and calculated descriptors of all keypoints in images. In the dataset, some images have bounding box data. I used the image in the bounding box if it’s available. I found that it improves the algorithm significantly.

In the image search, I first take descriptors of the query image, then match with descriptors of every other image, using the Brute-Force algorithm. In the matching, I used a ratio distance to find a good match. The formula of ratio distance $RD = \frac{\|f1-f2\|}{\|f1-f2\prime\|}$ where, f1 is a descriptor of the query image, f2 is that of the best match, f2’ is the second best match. I considered matches with ratio distances less than 0.7 good matches. It eliminates a lot of wrong matches between similar images.

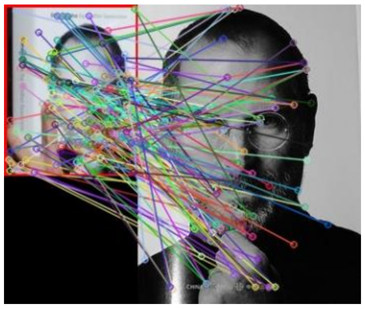
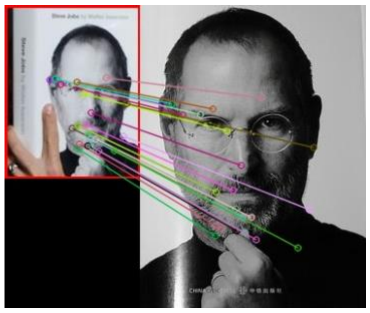
1)  2) 

Figure 1. Matches with SIFT: 1 – all matches, 2 – matches with the ratio distance less than 0.7

Then, for comparing images, I used new metrics that equal the ratio of numbers of good matches to numbers of all matches. I considered similar images as images with a high ratio and ranked similar images based on this ratio. I tried bag-of-words to improve this method, but it didn’t improve the results. Also, it took more time to create histograms of images. Therefore, I didn’t include it in the final method.

CNN

CNN models take ab image as input and calculate different features layer by layer, then classifies the image based on the features of the previous layer. We can use features of the penultimate layer as a feature vector. Images similar to each other have features that their Euclidean distance is close to each other. We use this property of CNN for an image search method.

For this method, I used the model ResNet-50 that was pre-trained on the dataset ImageNet [4]. I used Keras library in Python for accessing pre-trained models, and ResNet-50 was one of the available models in Keras. ResNet50’s top-1 accuracy on ImageNet is 75.9%. Also, after comparing with another nice model InceptionV3, which has 78.8% top-1 accuracy on ImageNet, I found ResNet-50 performs better on the given example queries. Then, I decided to use only ResNet-50.

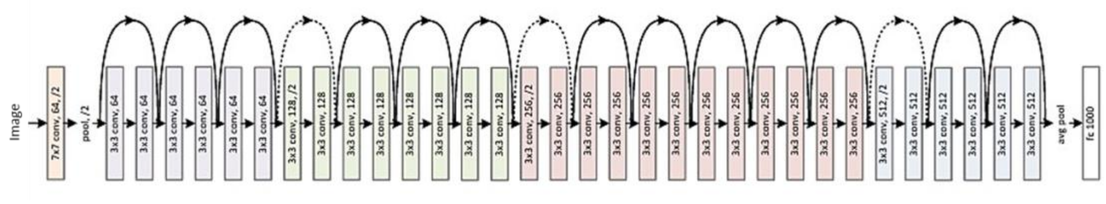
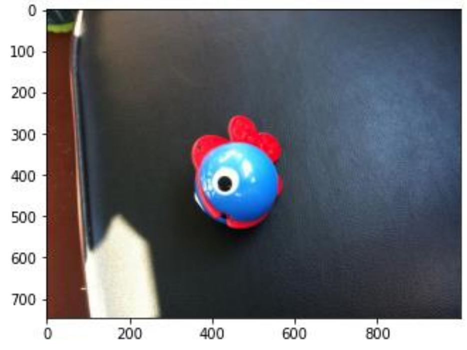


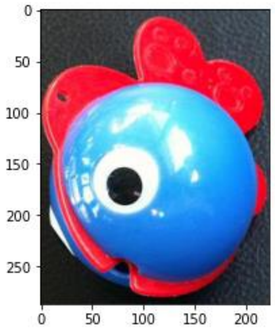
Figure 2. ResNet model architechture [2]

ResNet-50 takes only 224x224 pixel images. Before loading an image to the model, I crop the image in a bounding box if the bounding box is given, then resize the image to the proper dimension. If the bounding box isn’t given, the whole image is just resized. An example of this process is given below:

1)



2)



3)

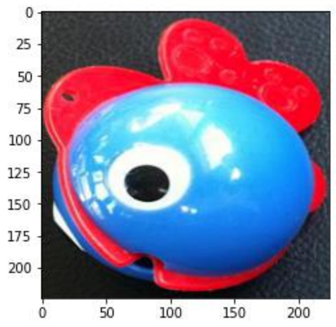


Figure 3. Process of taking an image into CNN model: 1 - an original image, 2 - image in a bounding box, 3 - image after resizing to 224 x 224

I use the pre-trained ResNet-50 without the last softmax layer. The model had fully connected layer with 2048 neurons before the softmax layer. So, after taking an image, model generates a feature vector of length 2048. I went through all the given images and extracted their features.

Each number in the feature is a float of 32-bit. By multiplying 2048, we get that each image has 8 KB feature vector. In the assignment's dataset, there are 5000 images and features of all images are equal to 40 MB, which is not very large. Therefore, I decided that it's not necessary to try to decrease a feature size.

To find similar images, the method first extracts the features of query image using the same model, then finds the Euclidean distance from the query feature to features of other images in 2048-dimensional space. We consider the image with the less distance similar image and rank image similarity by Euclidean distance.

Combined method

In CNN, as values are small, images are more similar. In SIFT, as values are large, images are more similar. So, the values of SIFT are in the opposite direction with CNN. I combined two methods by giving weight 1 on CNN values and weight -5 on SIFT values. In other words, given the query image, the combined model finds values in CNN and SIFT methods, then subtracts 5 times the value of SIFT from CNN. Finally, it takes images with less values as similar images. In a simple way, we can write as: Combined method = CNN method – 5 * SIFT method.

Results

Table 1. MAP values of 3 methods on example queries

Values	**SIFT**	**CNN**	**Combined**
Average Precision of Q1:	0.0990	0.8466	0.8467

Average Precision of Q2:	0.8602	0.6004	0.9317
Average Precision of Q3:	0.9451	0.2550	0.8785
Average Precision of Q4:	0.0110	0.7784	0.7966
Average Precision of Q5:	0.0498	0.9127	0.9214
Average Precision of Q6:	0.8451	0.5109	0.8706
Average Precision of Q7:	0.0069	0.6425	0.5509
Average Precision of Q8:	0.3279	0.1693	0.4025
Average Precision of Q9:	0.0206	0.8961	0.7791
Average Precision of Q10:	0.1475	0.5666	0.5922
Mean Average Precision:	0.331311	0.617845	0.757035

SIFT

The first method with SIFT had relatively low results. It failed with complex images like the Eiffel Tower or Statue of Liberty but did really well on simple images like a can of Pepsi, which have the same logo of Pepsi in all images.

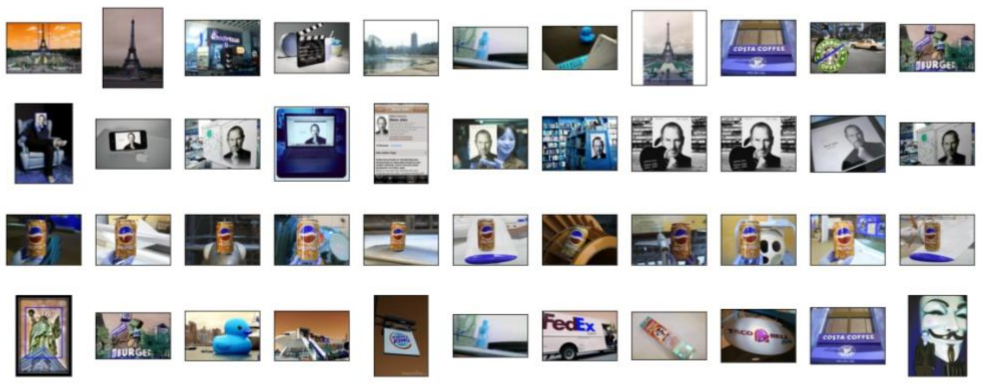


Figure 4. Top-10 results of the first 4 example queries using SIFT

CNN

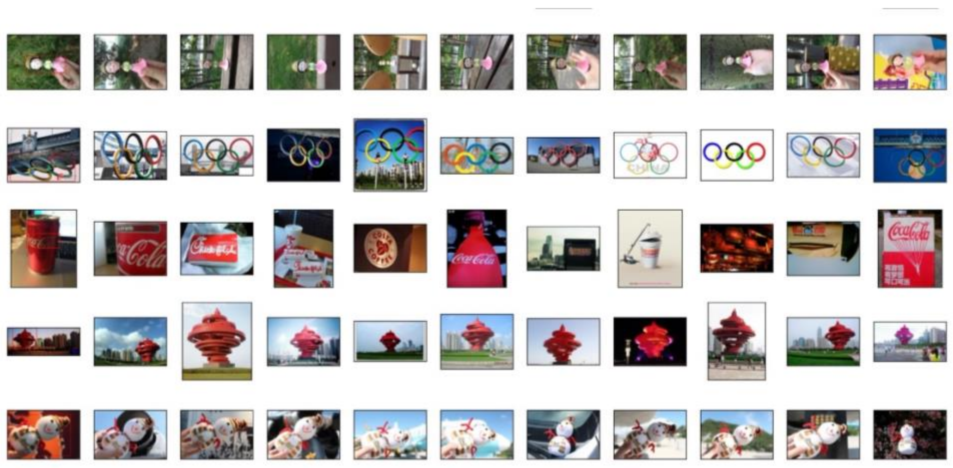


Figure 5. Top-10 results of 4 example queries using CNN

CNN method significantly outperformed SIFT method by doing good on complex queries. Interestingly, it didn't well on queries 2,3,6, in which SIFT performed very well.

In most queries, top-10 results are totally correct. It shows the high accuracy of the method. It performed worst in Q8 where it has to find images with the word "Coca Cola" (MAP=0.1693), but it gave images with different words. So, we can conclude that this method isn't robust in finding images with words.

Combined method

The combined method outperformed 2 previous methods by taking values both models are certain. Top-10 similar images below show how good the method did.

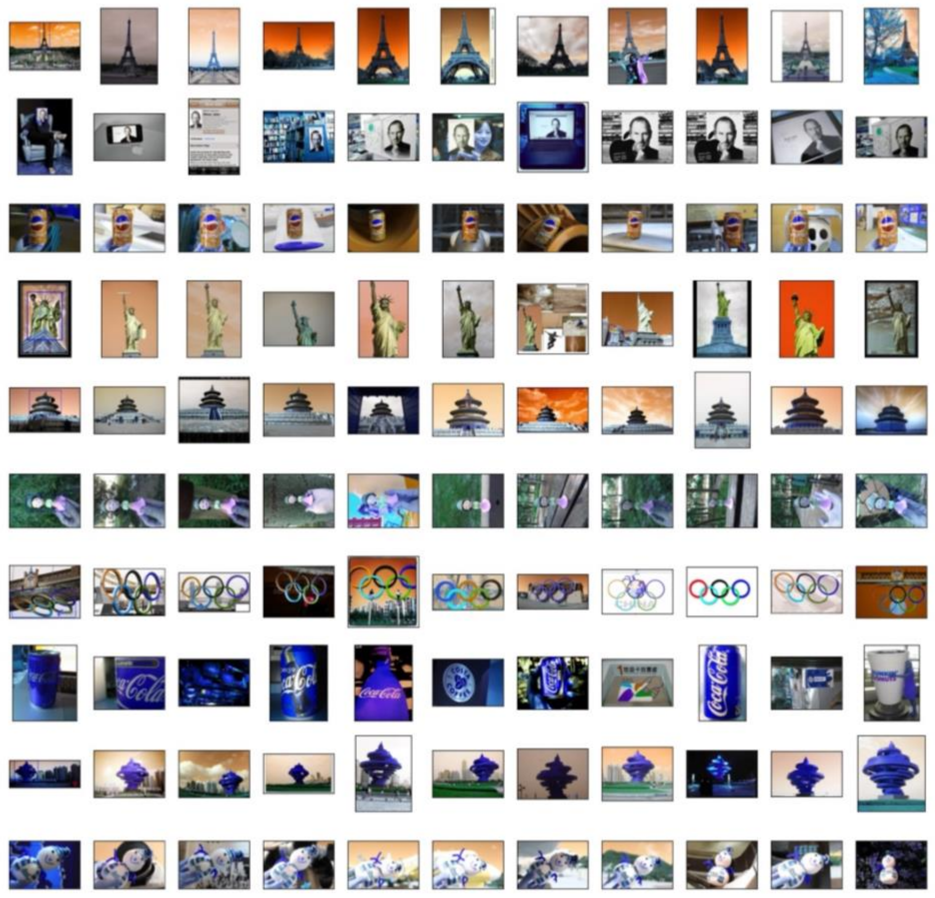


Figure 6. Top-10 results of example queries using the combined method

Remark: When I printed images with matplotlib, red and blue channels of images are swapped

Reference

1. Lowe, David G. (1999). "Object recognition from local scale-invariant features"
2. CNN models. https://datawow.io/blogs/cnn-models
3. A. Koul, S. Ganju, M. Kasam. Practical Deep Learning for Cloud, Mobile, and Edge. O'Reilly Media, 2019.
4. K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition