Image Classification - Dog vs. Cat

Team Members:

Yibing Luo： 912491862

Fan Wu: 912538518

Yingjie Li： 912435543

1. Introduction

In this project, the main purpose is to investigate how to identify a cat and a dog from an image. We got 12499 pictures of different cats and the same number of pictures for dogs from kaggle.[1] By using all these pictures, we want to find out a good way to classify remaining unlabeled pictures.

We realize that, unlike number or word, picture is the category, which cannot be easily described in programing languages. Thus we would like to find methods to transform picture into some manipulable form. As a result, we choose HOG and bag-of-word model (with k-means clustering) as the descriptors to describe pictures.

Finally, we feed the obtained descriptors into some recognition system based on supervised learning. Here, we try SVM classifier and KNN method.

2. Exploration the images

Before we started the classification, we decided to explore the difference between dogs and cats. In other words was to extract the features from the images. We want to find something unique and broad enough to distinguish cats and dogs so that all dogs would fall into one category and all cats into the other.
Also, since there is so many approaches that can help us extract the features from the images, we choose two of them to find the edge of the animals.

The first approach is an edge detector algorithm called Canny to find edges in images. Canny algorithm aims to satisfy three main criteria: Low error rate: Meaning a good detection of only existent edges; Good localization: The distance between edge pixels detected and real edge pixels have to be minimized; Minimal response: Only one detector response per edge. What's more, Canny does use two thresholds (upper and lower):

a. If a pixel gradient is higher than the upper threshold, the pixel is accepted as an edge.
b. If a pixel gradient value is below the lower threshold, then it is rejected.
c. If the pixel gradient is between the two thresholds, then it will be accepted only if it is connected to a pixel that is above the upper threshold.

Therefore, we can try different threshold and obtain the best result.

The second approach is Using the OpenCV function findContours and drawContours to finding contours in images. In this way, we can find colorful lines to describe the contours of cats and dogs. It would be much beautiful and clearly.

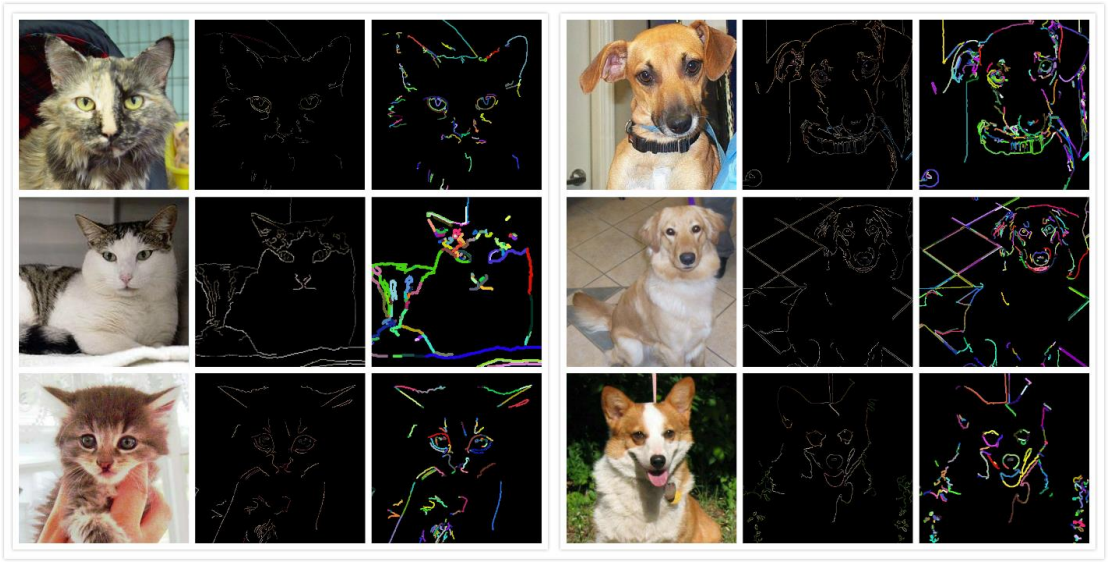

Figure 1 Two approaches of finding edge

From Figure 1, we can clearly find that the two approaches both can find the edge of images accurately and clearly. But if we judging from the visual effects alone, approach two is better than one. Because its colorful lines could see much more clearly. Hence, we choose the second one to find the difference between cats and dogs.

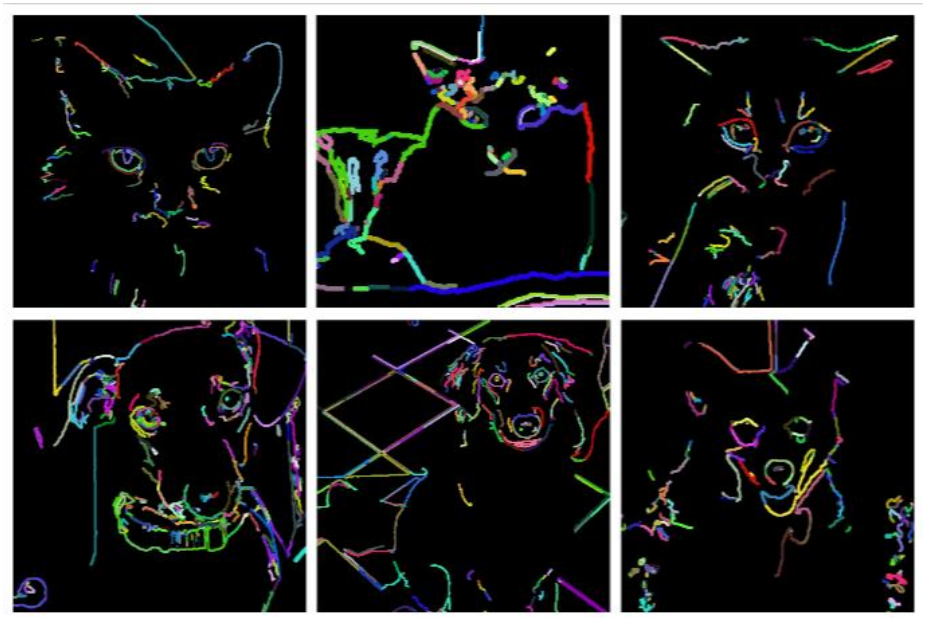

Figure 2 Difference between cats and dogs

From Figure 2, we can distinguish cats and dogs from their ears, eyes and noses. Usually, cats' ears are standing up while dogs' ears are drooping; Cats' eyes are big and elliptical while dogs' eyes are round; Cats' noses are small and short while dogs' noses are big and long. From these three significant differences, we can easily distinguish cats and dogs, but there will be some misjudgment. For example, the dog

in the bottom right picture from Figure 2, its ears is standing up which may be misleading.

3. HOG Descriptor and SVM Classifier

HOG is a feature classifier which can describe an image very well. And SVM is a binary classifier which looks for an optimal hyperplane as a decision function. So we want to get HOG descriptor of each image and train SVM by it.

3.1Filename Lists

At the beginning, we need to collect all the names of pictures which look like '*.JGP'. Obviously, recording the names manually is not a wise way. So we use a batch file which includes followed command:

dir /b/s/p/w *.jpg>trainlist.txt

Run this batch file in one specific fold can get a txt file which contains a list of all the names of files under current fold. Thus we can get a train-list and test-list.

3.2 Compute HOG Descriptor

The main and rough idea for HOG algorithm is described as below.

Firstly, we need to divide the image into small-connected regions called cells. Then for each cells, we will calculate their gradient and then compute a histogram of gradient directions or edge orientations for the pixels in each cells. We call these our cell descriptors. Groups of cells are considered as a block (such as 3×3 cells). And these groups of cells' histograms leads to one block histogram. Finally the set of these block histograms represents the descriptor of the image. And this descriptor is what we can use to do supervised learning.

It's quite convenient when we compute HOG descriptor in OpenCV. Utilizing class HOGDescriptor, we can define our HOG object detector. And by using compute() function, we can get a HOG descriptor of an image.

3.3Train SVM and Do Classification

After the previous part, we get a set of numbers to describe all the images. Then we can use these digital data to do some supervised learning. In this part, we try SVM in OpenCV to do classification.

First of all, we train our support vector machine by HOG descriptor of training images. Then we need to do almost the same action in 2.1 again to get the HOG descriptor of test images. Using the descriptor to evaluate the SVM model, we obtain the prediction result.

3.4Result

Since the limit of PC, we can support millions of images as train dataset. So we just train a small part of image. The result is below:

# of Images	Misclassification Rate
200 cats + 200 dogs	27%
400 cats +400 dogs	26%
600 cats + 600 dogs	24%
800 cats + 800 dogs	22%
1400 cats +1400 dogs	19%

So from the table above we can conclude that if we can train SVM by more train data, we can obtain a better model. But it cannot keep going. By reading some articles we know that there will be a place where the misclassification rate become stable.

Further we try to use the edge images, which are produced by Canny in part 2 to train SVM model. Noticed that the edge images come from the ones we used before. And then we get the following result:

# of Images	Misclassification Rate
200 cats + 200 dogs	26%
400 cats +400 dogs	24%
600 cats + 600 dogs	23%
800 cats + 800 dogs	23%
1400 cats +1400 dogs	17%

The result is a little bit better than previous one, which may implies the edge images give us more precisely information about dogs and cats.

4. BOW Model and KNN classifier.

Bag of words model treats the image as a text documents. In order to implement BOW model, we have following steps to do: feature detection, feature description and generate a word vocabulary. As long as we generate the word vocabulary, we use K-nearest-neighborhood (KNN) to predict our images.

4.1 Feature Detection and Description

We try two different methods to detect key points and present patches as numeric vectors. 3 methods are Scale-Invariant Feature Transform (SIFT) and Oriented Brief (ORB), where in ORB, Features From Accelerated Segment Test (FAST) is used as detector of features and Binary Robust Independent Elementary Features (BRIEF) as descriptor and it is an improvement of BRIEF.

Figure 3 shows SIFT and ORB key points location examples for cat and dog gray image.
We can see that the key points by SIFT is more than ORB for both cat and dog. There are a lot overlapping among key points from SIFT and ORB. In generally, they all detect some edge of the animals especially SIFT. They all detect key points covering the eye of cats. SIFT is much more sensitive to the background of the image than ORB.

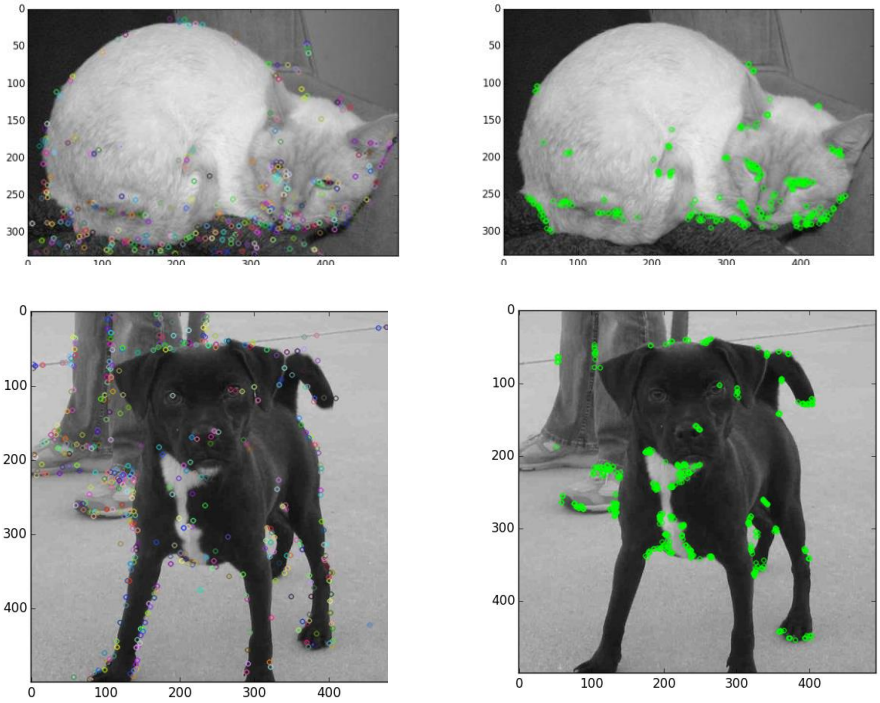

Figure 3 SIFT, ORB key points

Then these 2 methods applied to descript those features, while ORB uses the method as BRIEF to descript the key points. SIFT converts each features to 128-dimentional vector, while BRIEF finds the binary strings directly without finding descriptors. BRIEF provides a shortcut, so it is a faster method feature descriptor.

4.2 Creating a Visual Vocabulary

We perform K-means clustering to apply vector quantization. It divides up data into k (default is 3) clusters so as to minimize the within-cluster sum of squares. We

implement *BOWimgDescriptorExtractor* on *dextractor* and *dmatcher*, while *dextractor* is created by the method SIFT or ORB to compute descriptors for an input image and its features, and *dmatcher* is using L1 norm type for SIFT and HAMMING norm type for ORB by Brute-force descriptor matcher. [2]

4.3 KNN Performance

After we generate visual vocabulary, we conduct KNN classifier to predict image whether it contains cat or dog by setting neighborhood 3. The predictor for KNN classifiers are the clustered features getting from sample and response is whether the image is dog (Y = 1) or cat (Y = 0).

First, we use 100 cat images and 100 dog images as our train data to apply BOW model and KNN classifier. We find that classification rate for ORB method for feature detection and description is 0.24, for SIFT is 0.17. However, it takes ORB 15.161 seconds to complete and 23.687s for SIFT. Although SIFT predict better than ORB, it takes longer to detection and computation.

Since totally, we have 12499 images of dogs and cats, we want to know the influence of sample size on computational time and misclassification rate for both SIFT and ORB method.

The Figure 4 shows the relationship between computational time and sample size for method ORB and SIFT. In general, the computational time for method ORB is shorter than SIFT. The computational time for method SIFT is proportional to the simple size of images, indicating that the performance of SIFT method gets worse when sample size increases. Because the method SIFT needs to detect and descript more images for conducting BOW method. For 1 image, the time to feature detection and description seems to be constant for SIFT. For method ORB, the computational time increases as well when sample size increases.

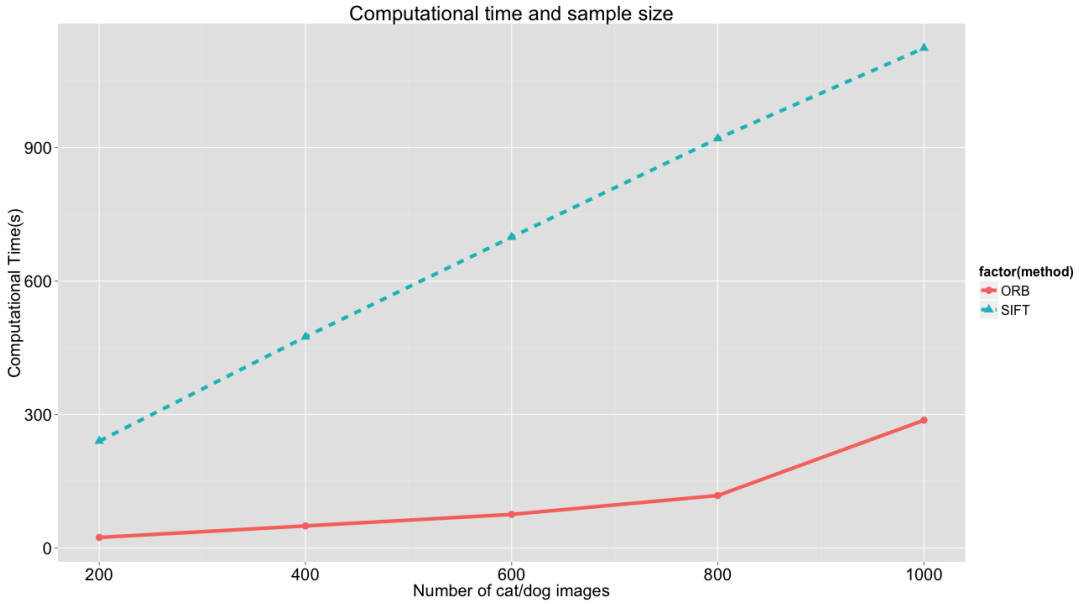

Figure 4

The Figure 5 indicates the relationship between the misclassification rate and sample size. The misclassification rate for method SIFT increases at first, then it reach a stability at misclassification rate 0.21. However, there is no clear association between the misclassification rate and sample size for the method ORB.

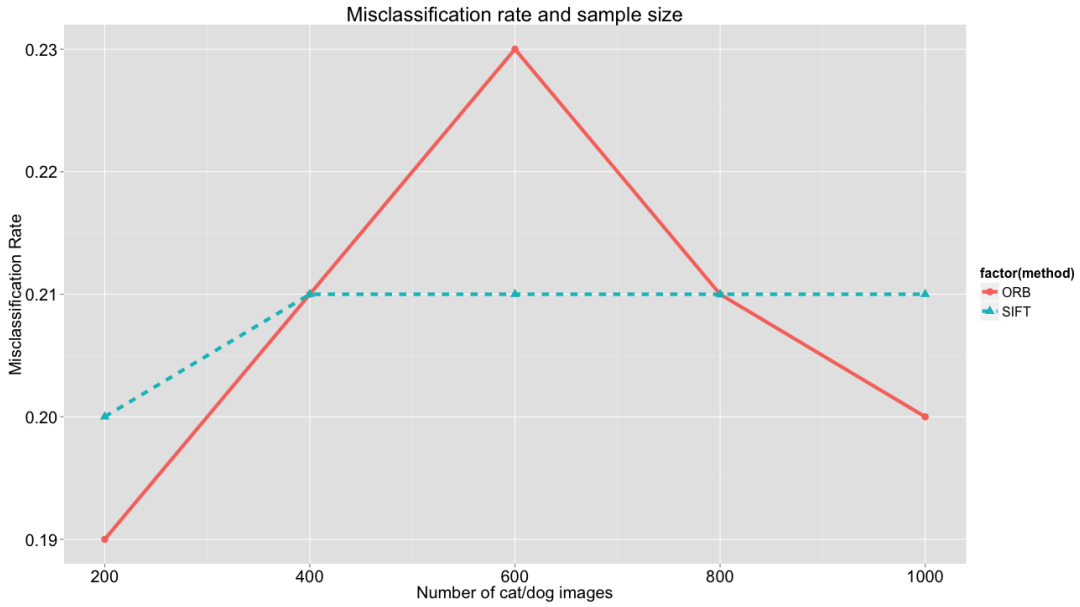

Figure 5

From the Figure 4 and 5, we can get that for small sample size and large sample size such as less than 400 and larger than 800 for our situation, method ORB is not only more efficient but also more accuracy than SIFT. For middle sample size, ORB provides quicker prediction while SIFT provides more precise prediction.

5. Conclusion and Future Work

We implement the HOG as well as BOW method for image classification.

For the method of HOG, we can conclude that as train data size increases, we can obtain a better prediction. Also, through the edge image, we get lower classification rate, indicating that we get a more precise model about dogs and cats at this situation.

For the method of BOW, we can conclude that there seems to be no significant association between sample size and classification rate for ORB for features. It increases at first then decreases. For SIFT, it attains a constant misclassification rate after certain sample size. The computational time for both SIFT and ORB increases as sample sizes increases. If sample sizes double, computational time seems to double as well. The time for the method SIFT is always longer than ORB, indicating ORB to be a faster method to do feature detection and description than SIFT. Because the method ORB utilizes BRIEF that provides a shortcut for binary string as descriptor.

For future work, if applicable, we may try to extract the body and head of cats and

dogs directly to eliminate the influence of background, then conduct HOG and BOW method to further analysis the potential improvement of prediction. For BOW, We may also try larger sample size to provide more evidence for the method SIFT that it will reach a stability and analysis the association between number of neighborhoods and performance of BOW.

Reference

[1] Kaggle, https://www.kaggle.com/c/dogs-vs-cats

[2] Description for Opencv
http://docs.opencv.org/modules/features2d/doc/object_categorization.html