IMAGE DESCRIPTION PLATFORM

Xing Hao
EECS, University of California, Irvine
xingh2@uci.edu

1. Introduction

Today, image recognition becomes a very important technique and is used in many applications. Such as, face recognition for security, online shopping where customers can find similar products, disease diagnosis by comparing Magnetic resonance imaging (MRI) images, or test spotting which can be used to translate the text in an image.

Since image recognition is so important, many researchers have devoted much effort to find the best solution and done a lot of great works in the past few decades. Some of them are proved to be very successful and used in many areas. These technologies can be summarized to two categories. The first one is local feature learning including Scale-invariant feature transform (SIFT) [1], Speeded-up robust features (SURF) [2], and so on. The second one is deep learning which has drawn a lot of attention since 2012 and is widely used today. There are many deep learning architectures such as Convolutional neural networks (CNN) [3], Recurrent neural network (RNN) [4], and Deep belief network (DBN) [5]. Among them, CNN is the most popular one. Table I shows the comparison of SIFT and CNN.

Table 1. Comparison of SIFT and CNN

	SCALE-INVARIANT FEATURE TRANSFORM (SIFT)	**DEEP NEURAL NETWORKS (CNN)**
Design complexity	Simpler deign and less parameters to set	Needs experience to make design decisions
Power and memory demands	Less processing power needed, memory needed for storing features for each image	High demand for processing during the training phase, memory needed to store the weights of the network
Training set	Smaller training set	The bigger the training set the better
Speed of output	Faster	Slower
Applications	More relevant for identification tasks; used in vide range of vision tasks; Can be used for real time scenarios	More relevant for classification and categorization tasks, has very good generalization abilities; image and video tasks

In this project, we build a platform to find description and similar images for a picture. After uploading an image, the platform will extract features of it, use them to classify images, and predict the descriptions for the images. The description of an image includes the top 5 labels which match the images with higher *confidence*. The most similar images will also be shown. The platform uses SIFT based image recognition algorithm, combined with Bag of words (BOW) [6] model and Support vector machine (SVM) [7].

In the rest of this report, we will review the work related to the project. As the first step, we will talk about the dataset in section 2 and architecture of the platform in section 3. Following them, we will introduce the preprocessing and training phase in section 4, which is then followed by a discussion of the experiment result in section 5. We will conclude the project in section 6.

2. Dataset

In this project, we used the images of Caltech 101[8] which includes objects belonging to 101 categories. About 40 to 800 images per category. Most categories have about 50 images. The size of each image is roughly 300 x 200 pixels. The categories includes accordion, anchor, bonsai, brain, butterfly, and more. Figure 1 shows some example images in the data set.

accordion anchor bonsai butterfly

Figure 1. Example images

Most images have little or no clutter. The objects tend to be centered in each image. Most objects are presented in a stereotypical pose. We will use those images as both training set and testing set to calculate the accuracy of our algorithm.

The dataset includes 8677 images. Figure 2 shows the number of images in different categories. The smallest category 'inline_skate' includes 31 images and the largest one 'airplanes' includes 800 images. So the accuracy of a model that always predict the majority class is 9.2%, which will be the benchmark for our algorithm and used to evaluate our algorithm.

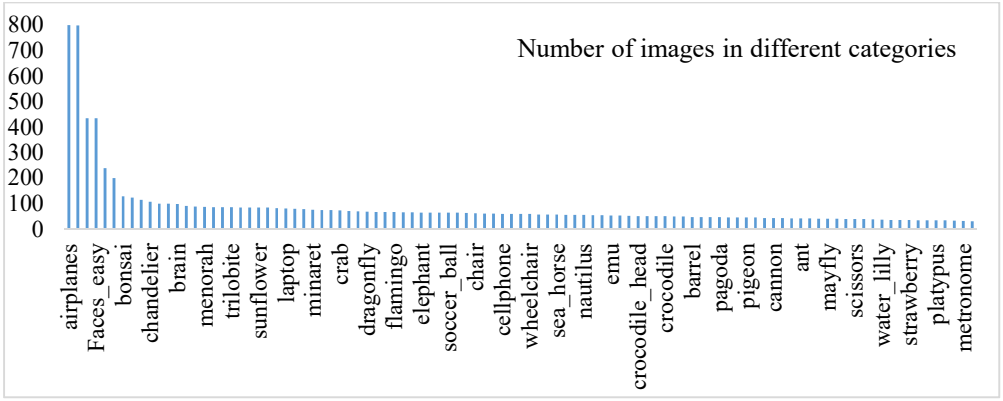

Figure 2. Number of images in different categories

3. Platform architecture

The architecture of the platform is shown in Figure 3. It includes two main steps. In the first step, after uploading the image, the SVM classifier of each category will calculate the confidence of the image belonging to this category. The top 5 categories will be returned as the description of the image. In the second step, the distance between the uploaded image and all the images in the top five categories will be calculated, and the top 6 closest images will be shown on the platform as the similar images. Next, we will introduce these two steps in detail.

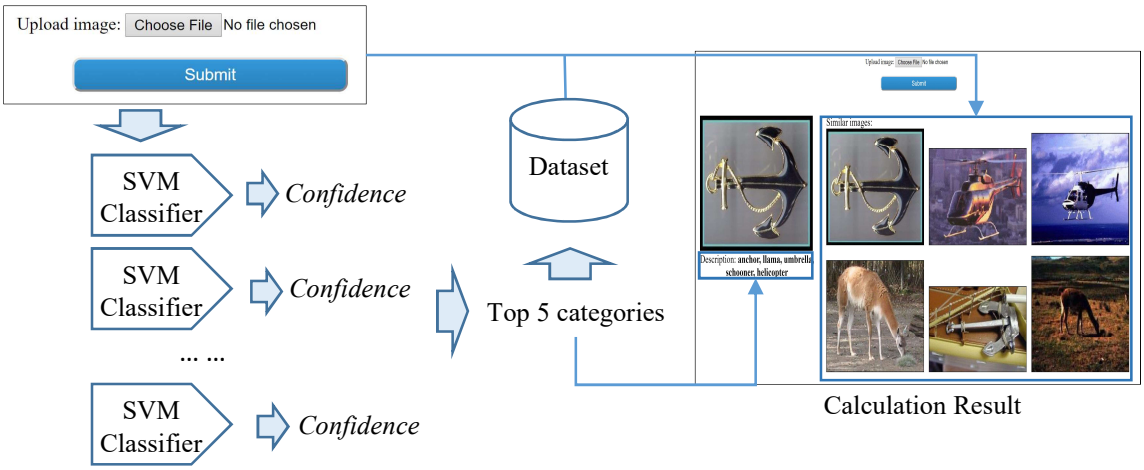

Figure 3. Architecture of the platform

3.1. *Calculate confidence*

In the training phase, we calculate the dictionary for each category which includes 50 centering features of the images in this category. After an image is uploaded to the platform, it will calculated the distances from all the features of this image and the centering features, and count the most close centers as the bag of words (BOW) of this image. For example, suppose the feature is an integer (it's a vector containing 128 values in our algorithm, which is calculated using SIFT). If the dictionary is [2, 6, 21], and the features of an image is [1, 5, 20, 21, 30], then the BOW of this image is [1, 1, 3], because the nearest centers for the image features is [2, 6, 21, 21, 21].

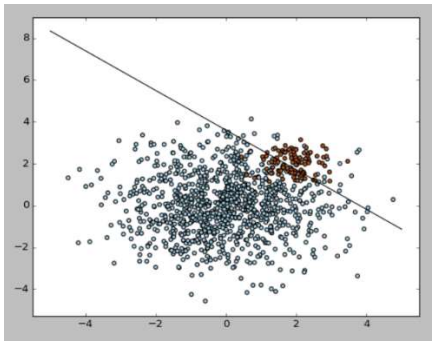

Figure 4. SVM

The SVM classifier uses an optimal separating hyperplane to classify the images, as shown in Figure 4. The input of the SVM in our algorithm is the BOW of the image. The SVM classifier of one category will return two values – *label* and *distance*. The *label* (0 or 1) denotes whether the image belongs to this category, and the *distance* is the distance from the image to the hyperplane. For each category, the *confidence* of the category is calculated as (*label*-0.5)*2**distance*, which is equal to

$$\begin{cases} distance, & if\ label = 1 \\ -distance, & if\ lable = -1 \end{cases} \tag{1}$$

So, *confidence*(*label*=1, larger *distance*) > *confidence*(*label*=1, smaller *distance*) > *confidence*(*label*=0, smaller *distance*) > *confidence*(*label*=1, larger *distance*). After calculating the *confidence* of the image in different categories, top 5 categories will be returned as the description of the image.

3.2. *Find similar images*

Using the BOW of the images, we calculate the distance from the uploaded image to all the images in the top 5 categories, and return the 6 nearest ones as the similar images. Figure 5 shows the 6 similar images (right side) for an image (left side). As mentioned in section 3.1, the BOW is a vector. We use 2-norm to calculate the distance of two vectors $X=(x_1, x_2, \ldots, x_n)$ and $Y=(y_1, y_2, \ldots, y_n)$:

$$[\sum_{i=1}^{n}(x_i - y_i)^2]^{1/2} \quad (2)$$

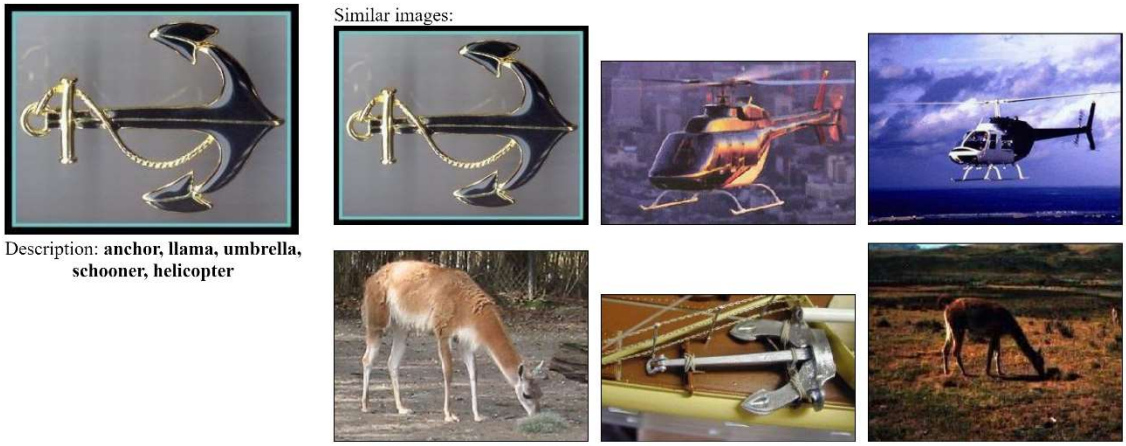

Figure 5. Calculation result

4. Preprocessing and training

In this project, we use Bag of Visual Words to train the classifier. Bag of Visual Words is an extension to the NLP algorithm Bag of Words. It was developed by CSurka et. al [9] essentially creates a vocabulary that can best describe the image in terms of features.

As we mentioned in section 3, for each category, we will calculate 50 features as the dictionary and train a SVM classifier. Figure 6 shows the work flow of the preprocessing and training. We will introduce the details of the 4 steps in this section.

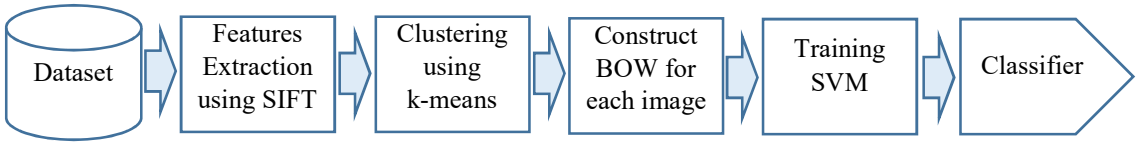

Figure 6. Steps of training

4.1. *Features extraction using SIFT*

SIFT algorithm extracts local features of objects from a set of reference images and store the local features. It contains 3 steps:

- **Scale-space creation**
  To create the scale space of an image, the image is first convolved with Gaussian-blurs at different scale δ. The larger of δ, the blurrier of the image. And then the blurry and resized images will be put together to build a tower. In this tower, there are many images in each level.

Images with same size but different blurry scales are put in the same level. Then the Difference-of-Gaussian (DoG) images are taken from adjacent Gaussian-blurred images per level.

- **Keypoint localization**
  Once DoG images have been obtained, keypoints are identified as local minima/maxima of the DoG images.
- **Orientation assignment**
  In the last step, each keypoint is assigned one or more orientations based on local image gradient directions.

In our project, we use OpenCV to calculate the SIFT features. A 16x16 neighborhood around the keypoint is taken. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created. So a total of 128 bin values are available, and the dimension of a local feature is 128. For each image, we extract 200 features. Thus, this step will output a 200*128 matrix for each image.

4.2. *Clustering using k-means*

Bag of Visual Words partition similar features that are extracted from the training set of images. These features together help to classify an image. The collection as well as frequency of particular features is what helps in estimating what object does the image contain. In our project, we use k-means to cluster the features of all images in one category and count the frequency of the centers for each image.

To be more specific, for each category, we collect all the features of the images in this category and cluster them using k-means which is one of the widely used algorithms when it comes to unsupervised learning. For example, for a category with n images, we will have $200n$ features, that are to be divided into 50 clusters. The input of k-means is the set of $200n$ features.

$$\arg\min_{S} \sum_{i=1}^{50} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{3}$$

where μ_i is centers for cluster S_i and S denotes set of features partitioned into clusters of $\{S_1, S_2, ..., S_i\}$. The set of 50 centers is the dictionary of the category. Therefore, after this step, we have 101 dictionaries, and each dictionary contains 50 'words'.

4.3. *Construct BOW for each image*

In the second step, we get the 'words' for each category. In this section we will introduce how to calculate the frequency of words for each image.

As discussed previously, each image has 200 features and each feature is a vector whose dimension is 128. For an image $X = \{X_1, X_2, \ldots, X_{200}\}$, and a category dictionary $Y = \{Y_1, Y_2, \ldots, Y_{50}\}$, where X_i and Y_j are vectors whose dimension is 128. We calculate distance between X_i and Y_j d_{ij} using 2-norm as in formula (2). The BOW for X is $\{W_1, W_2, \ldots, W_{50}\}$. $W_k = \sum_{i=1}^{200} \delta\left(\min_{1 \le j \le 50} d_{ij} = d_{ik}\right)$, where $\delta(C)$=1 if C is true and 0 if C is false.

Therefore, each image will have 101 BOW, and each BOW is a vector with 50 integers. And for each category, each image has a corresponding BOW based on the dictionary of this category. The BOW will be the final features of an image and used to classify the image.

4.4. *Training SVM*

In this step, we train a SVM classifier for each category which is used to check if an image belongs to this category.

For a category, we use the BOW of all the images based on the dictionary of this category as the input to train SVM. SVM classify data by finding the hyperplane (line in 2D, plane in 3D and hyperplane in higher dimensions) that best separates two classes of points with the maximum margin, as shown in Figure 4.

In our algorithm, the input data is the BOW of the images. To be more specific, the input data is a set of 8677 vectors, and each vector contains 50 integers. The label is 1 if the image belongs to this category or 0 if it doesn't belong to this category.

After this step, we store the 101 classifiers for all the categories and use them to classify new images.

5. Experiment and result

After train the 101 SVM classifiers, we test the accuracy of them using the same dataset. If the true category of the image is in the description, then the classification is marked as correct. The accuracy of the algorithm is calculated as (correct images)/(all images).

Figure 7 shows the result. The categories are listed from the largest to the smallest from left to right. The dot line denotes the accuracy of the benchmark that always predict the majority class. According to the result, the accuracy of two categories is smaller than the benchmark. The accuracy of all the images is 71.58%.

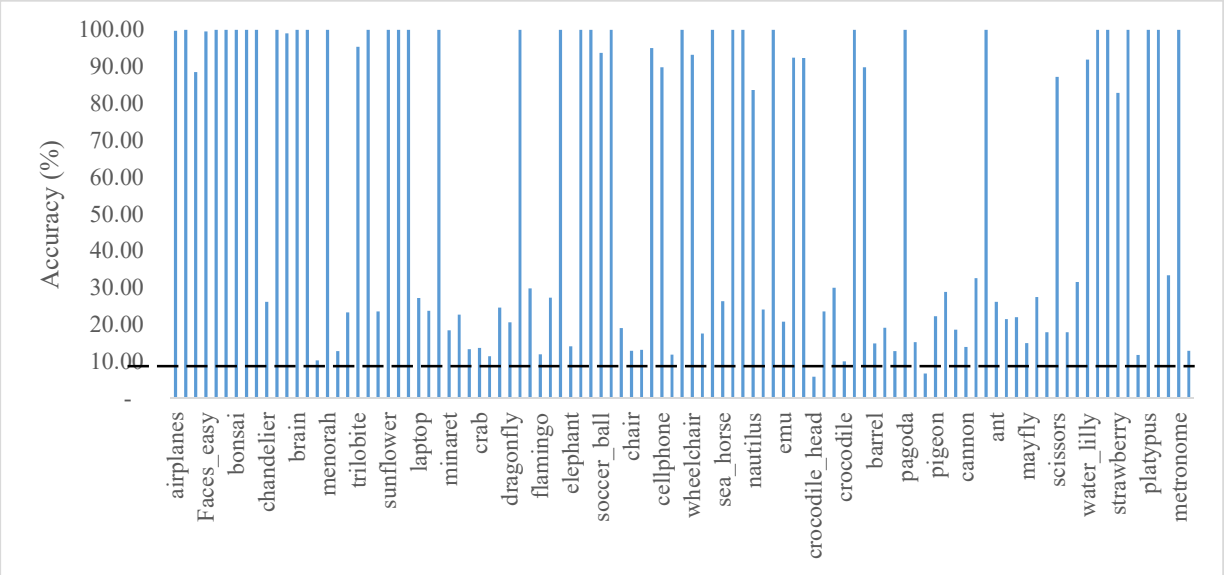

Figure 7. Accuracy of the algorithm

We also calculated the distribution of the accuracy, as shown in Figure 8. 36% of the 101 categories have accuracy larger than 99%, and these categories contain 53% images. 14% of the categories have accuracy larger than 80% which cover 14% images. Half of the categories have low accuracy which is smaller than 80% and these categories cover 34% of all the images. According to this result, we can say that the larger categories tend to have higher accuracy.

Figure 8. Distribution of accuracy

We also test the accuracy for different K from 1 to 5, as shown in Figure 9. The left bar is the accuracy of all images which is calculated as (number of correct prediction)/8677. The right bar is the average accuracy of all categories. From the result, it can be seen that algorithm based on higher K value has higher accuracy. Based on the above analysis, categories with higher accuracy have more images, so the accuracy is larger than the average accuracy of categories.

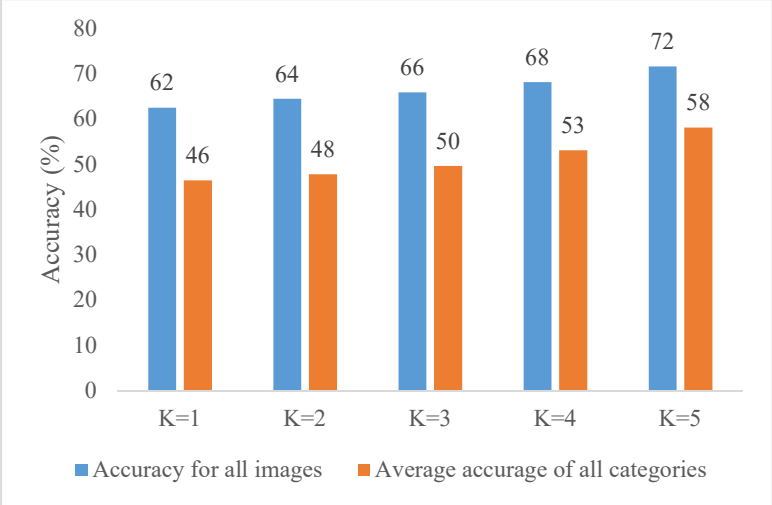

Figure 9. Accuracy for different K

The platform can also recognize multiple objects in the image, as shown in Figure 10, which successfully recognize the bonsai and llama in the image (not in the training dataset), but the similar images didn't show the images of llama. I run the platform in my laptop. The processor is 2.5GHz and the memory is 8GB. The time for calculating the result is 24 seconds to find the description and 101 seconds for the calculation of the nearest images. It is obviously needed to be improved.

.

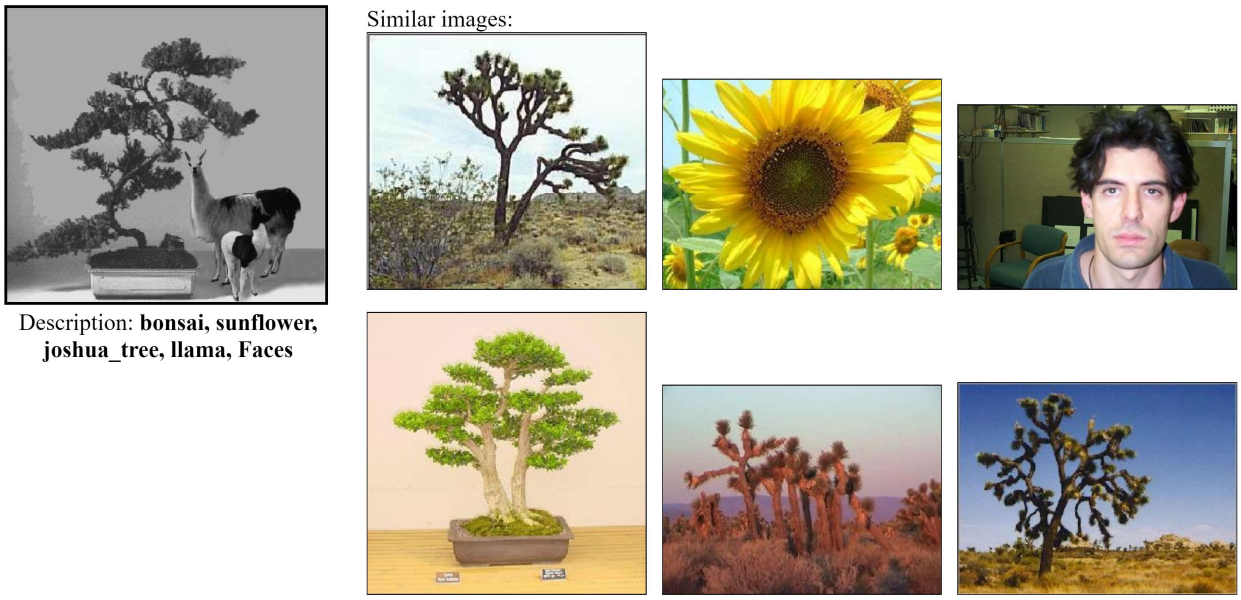

Figure 10. Calculation result

6. Conclusion and future work

In this project, we build a platform to show the description and similar images for a given image. We use the BOW model to train the SVM classifier and SIFT to extract the features from images. The platform can classify the object in an image and recognize multiple objects.

The result shows that the accuracy is much better than the benchmark. But there is still room for improvement:

1. To improve the accuracy, we can use more words for each dictionary;
2. According to the experiment result, the larger categories tend to have higher accuracy. So we can collect more training data for improving the accuracy.
3. To improve the efficiency, we can calculate the *confidence* and nearest neighbors for an image in parallel.

7. References

[1] Lowe, David G. "Object recognition from local scale-invariant features." In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, vol. 2, pp. 1150-1157. Ieee, 1999.

[2] Bay, Herbert, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. "Speeded-up robust features (SURF)." *Computer vision and image understanding* 110, no. 3 (2008): 346-359.

[3] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In *Advances in neural information processing systems*, pp. 1097-1105. 2012.

[4] Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9, no. 8 (1997): 1735-1780.

[5] Hinton, Geoffrey E., Simon Osindero, and Yee-Whye Teh. "A fast learning algorithm for deep belief nets." *Neural computation* 18, no. 7 (2006): 1527-1554.

[6] https://en.wikipedia.org/wiki/Bag-of-words_model
[7] Cortes, Corinna, and Vladimir Vapnik. "Support vector machine." *Machine learning* 20, no. 3 (1995): 273-297.
[8] Caltech 101: http://www.vision.caltech.edu/Image_Datasets/Caltech101/
[9] Csurka, Gabriella, Christopher Dance, Lixin Fan, Jutta Willamowski, and Cédric Bray. "Visual categorization with bags of keypoints." In *Workshop on statistical learning in computer vision, ECCV*, vol. 1, no. 1-22, pp. 1-2. 2004.