From Bag of Categories to Tree of Object Recognition

Haijing Wang Tianwen Zhang
School of Computer Science and Technology
Harbin Institute of Technology, Harbin, China
haijing.wang@yahoo.com

Peihua Li
College of Computer Science and Technology
Heilongjiang University, Harbin, China
peihualj@hotmail.com

ABSTRACT

To recognize different category of objects, multiclass categorization problem is often reduced to multiple binary problems. Traditional approaches require training different classifiers for each category. This can be slow and the performance of learned single classifier is poor for limited training samples. We present a multiclass object recognition tree, in which the leaf node and the non-leaf node correspond to one category and a bag of categories, respectively. Each non-leaf node captures the shared features of a bag of categories. Each node also holds a group of classifiers trained by AdaBoost, to discriminate the categories locating at its left and right child node. Recognition is then a process to find a path from the root to a leaf, which represents a unique category. The very promising result on Caltech 101 dataset shows the robustness of the proposed approach.

Keywords
object recognition, bag of categories, multiclass object recognition tree, AdaBoost

1. INTRODUCTION

Object recognition[1, 2, 3, 4, 5] is a fundamental vision problem: put simply, what's in the image, and where? To recognize different categories of objects in images, people usually reduce multiclass categorization problem to binary problem [6]. The approaches include that each category is compared against all others, or all pairs of categories are compared to each other, or error correcting output codes (ECOC) are used. They all ignore one fact that the subset of categories may exist some common features. And these traditional approaches require training different classifiers for each category. This can be slow and the performance of learned single classifier is poor for limited training samples.

We propose a novel concept "*bag of categories*" to show how to share the common features among different categories. Just like in the context of document analysis, *bag-of-words* [7] assumes that the order of words in a document can be neglected. In computer vision, the *bag of keypoints* [8, 9] method is based on vector quantization of affine invariant descriptors of image patches. *Bag-of-features* [10, 11] methods represent an image as an orderless collection of local features. *Bag of patches* and *bag of codewords* [12] are also appeared in the context of vision analysis.

For "*bag of categories*", two questions should be answered:

(Q1) How to combine these bags of categories?

(Q2) How to describe each bag of category?

For the first question, we build a multiclass object recognition tree, in which the leaf node and the non-leaf node correspond to one category and a subset of all categories, respectively. For each non-leaf node, its left and right children node hold two disjoint category subsets of its parent node. We call the category subset at each non-leaf node of tree as "*a bag of categories*". That is, bags of categories are combined by object recognition tree. For the second question, features with classifier is used to describe a bag of category. To improve the performance of each node, a group of classifiers are trained by AdaBoost [13, 14] based on the same dataset and different feature set. In the proposed framework, object recognition turns into a process to find a path from the root to a leaf, which represents a unique category, instead of the common nearest neighbor algorithm.

Our approach is different with probabilistic boosting tree (PBT) [15]. At each layer of the tree, each

category only exists in one node in our approach, however, each category may appear in more than one nodes in PBT. The construction and training of tree are together done by PBT. They are two different phases in our approach. In addition, we take more complicated features than PBT approach. Each node holds more than one strong classifier in our approach comparing one strong classifier in PBT. The recognition rate is 20% for PBT, however, it is 56.3% in our approach under the same dataset Caltech 101 [16].

The whole paper is scheduled as follows. Binary decision tree for multiclass object recognition is presented in Section 2. Details of bag of categories are introduced in Section 3. In Section 4, experiment design and analysis result are given. Finally, the conclusion of the paper is presented.

2. BINARY DECISION TREE FOR COMBINING BAGS OF CATEGORIES

Decision tree employs a hierarchically structured decision function in a sequential fashion. To combine bags of categories, the binary decision tree is constructed. For each non-leaf node, it corresponds to a binary classification. Thus, multiclass object recognition is transferred into many binary classification problems. It is the basis to construct a tree framework. In this section, we first describe the elements of the recognition tree and the splitting rules. Then we present the algorithm of tree construction for combining bags of categories.

2.1. From Multiclass Problem to Binary Classification Problem

Suppose we have m samples, each of which corresponds to an image,

$$\{\mathbf{x}_i | \mathbf{x}_i \in \mathbf{X}, \mathbf{X} \subset \mathbb{R}^n\}_{i=1}^m$$

where n is equal to the number of image pixels. The corresponding class label of each sample is

$$\{y_i | y_i \in \mathbf{Y}, \mathbf{Y} = \{1, \ldots, C\}\}_{i=1}^m$$

where C is the total number of categories.

There are many ways to reduce a multiclass problem to multiple binary classification problems [6], such as one-against-all approach, all-pairs approach, and error correcting output codes (ECOC).

- One-against-all approach is the simplest approach to create one binary problem for each of the C categories. That is, for any category label $r \in \mathbf{Y}$, all samples labeled $y_i = r$ are considered as one class and all other samples are considered as the other class.
- In all-pairs approach, for each distinct label pair $r, s \in \mathbf{Y}$, samples labeled $y_i = r$ are considered as one class, and those labeled $y_i = s$ belong to the other class. All other samples left are simply ignored.
- Error correcting output codes (ECOC) is to associate each class $r \in \mathbf{Y}$ with a row of a "coding matrix" $\mathbf{M} \in \{-1, 0, +1\}^{C \times l}$ for some l, where l is number of classifiers. The binary classification problem is then run once for each column of the matrix.

Unlike the one-against-all, all-pairs, and ECOC approach, we construct the multiclass object recognition tree by subdividing the whole category set into two subsets, as the two sides of the binary classification tree, and recursively to apply this kind of subdivision until all category is distinguished each other. To construct the decision tree, some elements should be considered first.

2.2. Elements of Decision Tree

Binary decision tree is known as a class of nonlinear classifiers. The root of the tree $\mathbf{T}$ is associated with the training set $\mathbf{X}$, which corresponds to all categories to be recognized. Each node, t, represents a specific subset $\mathbf{X}_t$ of the training set $\mathbf{X}$, which corresponds to a subset of all categories. We call categories in one subset as "a bag of categories", since we describe these categories in the subset using the same classifiers with sharing features. Splitting of a node is equivalent to split the subset $\mathbf{X}_t$ into two disjoint descendant subsets, $\mathbf{X}_{tL}, \mathbf{X}_{tR}$. For every split, the following equation (1) and (2) need to be satisfied:

$$\mathbf{X}_{tL} \bigcap \mathbf{X}_{tR} = \emptyset \tag{1}$$

$$\mathbf{X}_{tL} \bigcup \mathbf{X}_{tR} = \mathbf{X}_t \tag{2}$$

Here $\mathbf{X}_{tL}$ and $\mathbf{X}_{tR}$ correspond to two bags of categories, and locate at the left and right children node, respectively.

In order to develop a binary decision tree, the following design elements have to be considered in the constructing phase:

(E1) A splitting criterion should be adopted according to which the best split from the set of candidate one is chosen.

(E2) A stop splitting rule is required to control the growth of the tree and to declare a node as a terminal leaf.

(E3) A rule is required that assigns each leaf to a specific category.

A leaf node is formed when the subset $\mathbf{X}_t$ only includes the samples which belong to a single category. That is, a leaf node corresponds to a single category. And non-leaf node is a bag of categories. The code word from the root to each leaf node represents a specific category. Details on splitting criterion are described in the next subsection.

2.3. Splitting Criterion

The descendant nodes are associated with two new subsets, i.e., $\mathbf{X}_{tL}, \mathbf{X}_{tR}$, respectively. For the tree growing methodology, from the root down to the leaves, every split generates new subsets, in which elements are are more similar each other compared to the ancestor's subset $\mathbf{X}_t$. This means that the training feature vectors in each one of the new subsets show a higher preference for specific categories. The splitting algorithm is described as follows.

Algorithm 1. Splitting algorithm

Input:

- Matrix of feature values $\mathbf{G_t}$ of samples $\mathbf{X_t}$ with corresponding label set $\mathbf{Y_t}$.
- Desired number of clusters L, here $L = 2$ for left and right node of the binary tree.
- Iteration times N of splitting.

Algorithm:

(S1) For $j = 1$ to L

- Initialize arbitrary clustering center $\theta_j(0)$ for the j-th cluster center θ_j.

(S2) Repeat N iterations to find L cluster centers.

(S2a) For $i = 1$ to m_t (m_t is the number of samples to be split.)

- Determine the closest representative, say θ_j, for $\mathbf{g}_i \in \mathbf{G_t}$. Here $\mathbf{g}_i$ is the feature value of sample $\mathbf{x}_i$. The closest representative means that $d(\mathbf{g}_i, \theta_j) < d(\mathbf{g}_i, \theta_k), k = 1, \ldots, L,$ and $k \neq j$, where $d(\cdot, \cdot)$ is a distance measure.
- Set $b(i) = j$ for sample $\mathbf{x}_i$. It shows sample $\mathbf{x}_i$ is allocated to the j-th node of the tree.

(S2b) For $j = 1$ to L

- Determine θ_j as the mean of the vectors $\mathbf{g}_i \in \mathbf{G}$ with $b(i) = j$ to update parameter. That is, the original θ_j is replaced by the mean of the feature values of samples which are allocated to the j-th node of the tree in step (S2a).

(S3) For each category, count the number of samples located in each child node.
For $j = 1$ to L
For $k = 1$ to C_t (C_t is the number of categories to be split.)

- Compute

$$\lambda_{jk} = \sum_{i=1}^{m} \delta(\mathbf{x}_i) \tag{3}$$

where

$$\delta(\mathbf{x}_i) = \begin{cases} 1 & \text{if } \mathbf{y}_i = k \text{ and } b(i) = j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

(S4) Decide the final splitting
For $i = 1$ to m_t

- Set $b(i) = j$, if $\lambda_{jy_i} \geq \lambda_{ky_i}, k = 1, \ldots, L$, $k \neq j$. It means sample $\mathbf{x}_i$ is split to the j-th node of the tree.■

For binary tree ($L = 2$), the descendant nodes are derived with two new subsets, i.e., $\mathbf{X}_{tL}$ and $\mathbf{X}_{tR}$.

$$\mathbf{X}_{tL} = \{\mathbf{x}_i | b(i) = 1\}, \mathbf{X}_{tR} = \{\mathbf{x}_i | b(i) = 2\}$$

Therefore, the splitting algorithm is also the process to transfer each bag of categories into a smaller bag of categories. The process is continued until the stop rule is satisfied.

2.4. Algorithm for Constructing Binary Decision Tree

After the discussion on the major elements needed for the growth of a decision tree, we are now ready to summarize the basic algorithmic steps for constructing a binary decision tree for bag of categories.

Algorithm 2. Tree constructing algorithm

(S1) Begin with the root, i.e., $\mathbf{X}_t = \mathbf{X}$

(S2) For each new node t

- Learn a group of classifiers with AdaBoost based on sample set $\mathbf{X}_{tL}$ and $\mathbf{X}_{tR}$.

(S3) Do all samples of the subset $\mathbf{X}_t$ belong to a single category?

- If yes, stop splitting and declare node t as a leaf and designate it with a category label. End.
- If not, generate two descendant nodes t_L and t_R with associated subsets $\mathbf{X}_{tL}$ and $\mathbf{X}_{tR}$. Return to (S2).■

In the algorithm, we first classify the whole sample set into two subsets. In the first stages of binary tree, the root includes all categories to be classified. The existed classifiers in the node of tree are used to estimate the image's category label for its children nodes. Next, we classify each subset into two new subsets following Equ.(1) and Equ.(2). Each subset corresponds to a bag of categories. We end up the classification until each category is a sole part. This is a binary decision tree model for multiclass object recognition.

3. BAG OF CATEGORIES FOR VISUAL CATEGORY VOCABULARY

The classifier with selected features in each node of the tree corresponds a visual category vocabulary. To make the visual category vocabulary describe the bag of categories more clearly, we learn a group of classifiers for each node. Since these classifiers are independent, we propose a two-stage voting strategy to help the final decision of each bag of categories. After describing the bag of categories with visual category vocabulary and combing the bag of categories with binary decision tree, the recognition process for a new sample can be finished by the traversal of the tree from root to leaf.

3.1. Visual Category Vocabulary

Each category vocabulary is a group of AdaBoost strong classifiers. Once the samples are grouped into a binary tree, we train a group of classifiers for each node of the binary tree. AdaBoost[13, 14] is used to train each classifier, and original haar-like features proposed by viola and Jones[17] are taken as feature set. AdaBoost algorithm, proposed in the Computational Learning Theory literature, is a method to find a highly accurate hypothesis (a strong classifier) by combining many "weak" hypotheses, each of which is based on the reweighted version of the training data in order to emphasize those which are incorrectly classified by previous weak classifiers, and only moderately accurate. The final strong classifier is a weighted combination of weak classifiers followed by a threshold. The decision of a strong classifier is bias. Thus, we learn a group of independent strong classifiers at each node. The final decision is created with the voting rule described as follows.

3.2. Voting Rule

The combined decision is obtained by a majority vote of the individual classifiers. Majority vote does not assume prior knowledge of the behavior of the individual classifiers.

We assume that there are n classifiers, and each classifier produces a unique decision regarding the identity of the sample. In our approach, the following conditions are satisfied:

(C1) The number of voters is odd.

(C2) Each voter has the same probability of voting one way.

(C3) The individual decision is independent, since each classifier is trained independently and is based on different feature set.

We take two-stage voting strategy.

Algorithm 3. Two-stage voting algorithm

(S1) If $|k_t - \frac{n_t+1}{2}| > \tau$, the sample is assigned to the side when k experts are agreed.

(S2) If $|k_t - \frac{n_t+1}{2}| \leq \tau$, consider two descendant nodes t_L and t_R with associated subsets $\mathbf{X}_{tL}$ and $\mathbf{X}_{tR}$.

- If $|k_{tLL} + k_{tLR} - n_{tL}| > |k_{tRL} + k_{tRR} - n_{tR}|$, the sample is assigned to the left node.
- If $|k_{tLL} + k_{tLR} - n_{tL}| < |k_{tRL} + k_{tRR} - n_{tR}|$, the sample is assigned to the right node.
- If $|k_{tLL} + k_{tLR} - n_{tL}| = |k_{tRL} + k_{tRR} - n_{tR}|$, the sample is assigned to the left or right node randomly.

Here τ is voting factor. k_t, k_{tL}, and k_{tR} are the number of experts (strong classifiers) agreeing on the identity in the parent node, left child, and right child node, respectively. k_{tLL} and k_{tLR} are left and right child of the left child node t_L. And k_{tRL} and k_{tRR} are left and right child of the right child node t_R.

3.3. Category Encoding

In the proposed framework, each node is a set of classifiers. The bag of categories refers to a set of classifiers to describe a group of categories. More precisely, the category recognition is a two-step process.

(a) The local regions (patches) corresponding to the features of each classifier are sampled from image.

(b) The sample is assigned to the left or right node of the binary decision tree.

We repeat (b) until we reach a leaf node classifier to make a global decision about the test image.

Decision tree is a multistage system, in which categories are sequentially rejected until we reach a final accept category. To this end, the feature space is split into unique regions, corresponding to each category, in a sequential manner. Recognition is achieved from the root to the leaf node. Each leaf node corresponds to a code book, and each category can be noted by a binary code. This is a deep first traversal of tree.

4. EXPERIMENTS

We use the Caltech 101 object class dataset [16]. The Caltech 101 dataset[1] contains 9,197 images comprising 101 different object categories, plus a background category, collected via Google image search in September 2003 by Fei-Fei Li, Marco Andreetto, and Marc 'Aurelio Ranzato. Each category contains about 40 to 800 images. Most categories have about 50 images.

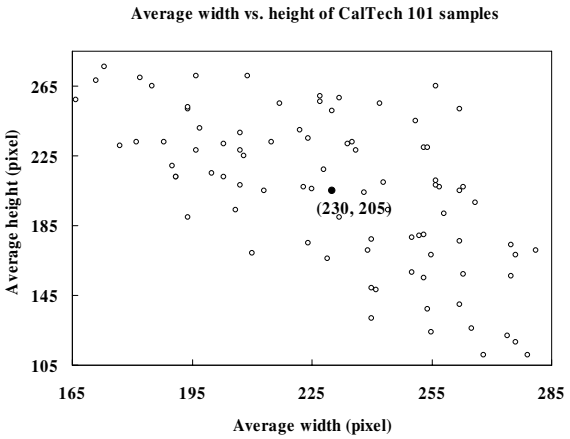

Fig. 1. Average width vs. height of Caltech 101 samples. Each circle represents a pair of width and height of one category. Some small width and height pairs are excluded. The solid circle point with coordinate (230, 205) is the average width and height of samples for all categories.

In our experiment, "Faces" and "Faces_easy" category are combined into one category. Therefore, the total number of categories is 100. Fig. 1 computes the average width and height of Caltech 101 samples. In Fig. 1, fifteen pairs of width and height are excluded because their small values in width, height, or both. In order to accelerate the training speed, we resize the width and height to 100×90 following the scale of average width and height 230×205.

Thirty samples were chosen randomly from each of the 100 object categories, yielding a total 3000 samples. For each class, testing images are ones excluding those used as training samples. Samples are not alignment before training.

Experimental results on Caltech 101 dataset show the robustness of our proposed approach. The results are shown in Table 1 and Fig. 2 for 15, 25 and 30 training samples per category. Results are reported from Fei-Fei, Fergus, & Perona [2], Serre, Wolf, & Poggio [4], Holub, Perona, & Welling [5], Berg, Berg, & Malik [1], Mutch & Lowe [3], Tu [15], and proposed approach.

[1] www.vision.caltech.edu/ImageDatasets/Caltech101/

The final decision tree of our experiment based on Caltech 101 dataset is given in Fig3. Each row corresponds to the coding of each category, and 0 and 1 shows the node locates at the left and right of its father node. For example, the ID of the second row is 23, and the code string "00000010" is the code word of the category 23 named "crab". It exists in the left child node from stage 1 to stage 6, the right child node of the stage 7, and the left child node of the final stage 8. Its substring represents a node of the tree, whose length indicates the number of layer of the tree.

Approach	Samples		
	15	**25**	**30**
Fei-Fei, Fergus, & Perona [2]	16	–	–
Tu [15]	–	20	–
Serre, Wolf, & Poggio [4]	35	–	42
Holub, Perona, & Welling [5]	37	–	43
Berg, Berg, & Malik [1]	48	–	–
Mutch & Lowe [3]	51	–	56
Proposed approach	42	–	56.3

Table 1. Correct rate in percentage with 15, 25, or 30 training samples per category on Caltech 101 dataset.

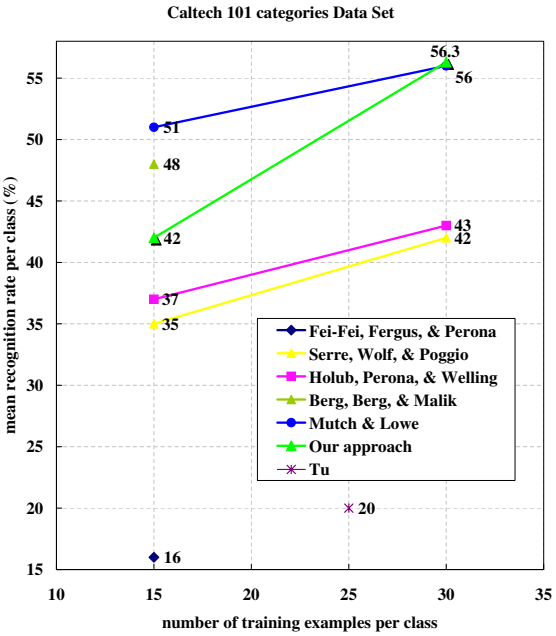

Fig. 2. Mean recognition rate on the Caltech 101 data set. This plot shows the recognition rate using the binary decision tree for multiclass object recognition for 15, 25 and 30 training samples per category. All points on the plot refer to recognition rates that have been normalized according to the number of test samples per category. (The figure is best viewed in color.)

ID	Category Name	STAGE(LEFT) 1	2	3	4	5	6	7	8	9	10
1	accordion	0	0	0	0	0	0	0			
23	crab	0	0	0	0	0	0	1	0		
48	hedgehog	0	0	0	0	0	0	1	1		
20	chandelier	0	0	0	0	0	1	0	0		
55	lamp	0	0	0	0	0	1	0	1	0	
80	schooner	0	0	0	0	0	1	0	1	1	
89	stop_sign	0	0	0	0	0	1	1			
13	butterfly	0	0	0	0	1	0	0			
53	kangaroo	0	0	0	0	1	0	1	0		
83	sea_horse	0	0	0	0	1	0	1	1		
73	pizza	0	0	0	0	1	1	0			
94	umbrella	0	0	0	0	1	1	1			
19	chair	0	0	0	1	0	0				
41	flamingo_he	0	0	0	1	0	1	0			
65	minaret	0	0	0	1	0	1	1			
76	revolver	0	0	0	1	1	0				
95	watch	0	0	0	1	1	1				
2	airplanes	0	0	1	0	0	0	0			
39	ferry	0	0	1	0	0	0	1			
6	bass	0	0	1	0	0	1	0			
46	hawksbill	0	0	1	0	0	1	1			
62	mayfly	0	0	1	0	1	0				
67	nautilus	0	0	1	0	1	1				
12	buddha	0	0	1	1	0	0				
50	ibis	0	0	1	1	0	1				
35	euphonium	0	0	1	1	1	0				
98	wild_cat	0	0	1	1	1	1				
3	anchor	0	1	0	0	0					
25	crocodile	0	1	0	0	1	0				
31	dragonfly	0	1	0	0	1	1				
28	dalmatian	0	1	0	1	0	0	0			
57	Leopards	0	1	0	1	0	0	1			
71	panda	0	1	0	1	0	1				
43	gerenuk	0	1	0	1	1	0				
84	snoopy	0	1	0	1	1	1				
5	barrel	0	1	1	0	0	0	0			
37	Faces	0	1	1	0	0	0	1			
10	brain	0	1	1	0	0	1	0			
54	ketch	0	1	1	0	0	1	1	0		
60	lotus	0	1	1	0	0	1	1	1		
16	car_side	0	1	1	0	1	0	0			
47	headphone	0	1	1	0	1	0	1			
96	water_lilly	0	1	1	0	1	1				
26	crocodile_h	0	1	1	1	0					
40	flamingo	0	1	1	1	1	0				
100	wrench	0	1	1	1	1	1				

ID	Category Name	STAGE(RIGHT) 1	2	3	4	5	6	7	8	9	10
4	ant	1	0	0	0	0	0				
14	camera	1	0	0	0	0	1				
17	ceiling_fan	1	0	0	0	1	0	0	0		
34	emu	1	0	0	0	1	0	0	1	0	
63	menorah	1	0	0	0	1	0	0	1	1	
74	platypus	1	0	0	0	1	0	1			
85	soccer_ball	1	0	0	0	1	1				
7	beaver	1	0	0	1	0	0	0	0	0	
36	ewer	1	0	0	1	0	0	0	0	1	0
52	joshua_tree	1	0	0	1	0	0	0	0	1	1
56	laptop	1	0	0	1	0	0	0	1		
61	mandolin	1	0	0	1	0	0	1			
32	electric_guitar	1	0	0	1	0	1	0			
79	saxophone	1	0	0	1	0	1	1			
88	stegosaurus	1	0	0	1	1					
8	binocular	1	0	1	0	0	0				
33	elephant	1	0	1	0	0	1				
18	cellphone	1	0	1	0	1	0	0			
21	cougar_body	1	0	1	0	1	0	1			
93	trilobite	1	0	1	0	1	1				
44	gramophone	1	0	1	1	0					
66	Motorbikes	1	0	1	1	1	0				
72	pigeon	1	0	1	1	1	1	0	0		
77	rhino	1	0	1	1	1	1	0	1		
99	windsor_chair	1	0	1	1	1	1	1			
9	bonsai	1	1	0	0	0	0	0			
22	cougar_face	1	1	0	0	0	0	1			
81	scissors	1	1	0	0	0	1				
29	dollar_bill	1	1	0	0	1	0				
70	pagoda	1	1	0	0	1	1	0	0		
82	scorpion	1	1	0	0	1	1	0	1		
86	stapler	1	1	0	0	1	1	1			
11	brontosaurus	1	1	0	1	0	0	0			
24	crayfish	1	1	0	1	0	0	1	0		
59	lobster	1	1	0	1	0	0	1	1	0	0
69	okapi	1	1	0	1	0	0	1	1	0	1
78	rooster	1	1	0	1	0	0	1	1	1	
15	cannon	1	1	0	1	0	1	0	0	0	
45	grand_piano	1	1	0	1	0	1	0	0	1	
64	metronome	1	1	0	1	0	1	0	1		
91	sunflower	1	1	0	1	0	1	1	0		
92	tick	1	1	0	1	0	1	1	1		
30	dolphin	1	1	0	1	1	0	0	0		
42	garfield	1	1	0	1	1	0	0	1		
68	octopus	1	1	0	1	1	0	1			
97	wheelchair	1	1	0	1	1	1				
27	cup	1	1	1	0	0					
49	helicopter	1	1	1	0	1					
51	inline_skate	1	1	1	1	0	0	0			
75	pyramid	1	1	1	1	0	0	1	0		
87	starfish	1	1	1	1	0	0	1	1		
90	strawberry	1	1	1	1	0	1				
58	llama	1	1	1	1	1	0				
101	yin_yang	1	1	1	1	1	1				

Fig. 3. Final decision tree for multiclass object recognition of Caltech 101 dataset. The first two columns are category ID and name. From Column three to Column twelve is the code of category.

5. CONCLUSION

We construct the multiclass object recognition tree by subdividing the whole category set into two subsets, as two sides of the binary classification tree, and recursively to apply the subdivision until all category is distinguished each other. A subset corresponds to a bag of categories. Recognition is a process to find a path from root to a leaf, which represents a unique category. Multiclass object recognition is transferred into many binary classification problems by the binary tree.

We answer three issues for object recognition.

1. How do we represent the object? We represent object by code book and bag of categories.
2. Using the representation, how do we learn a particular object category? The binary decision tree is constructed for multiclass object recognition.
3. Finally how do we use the model we have learnt to find further instances in query images? This is a deep first traversal of tree.

More complicated features will be considered in the future work to improve the performance.

6. REFERENCES

[1] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondence," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[2] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *CVPR Workshop of Generative Model Based Vision*, 2004.

[3] J. Mutch and D. Lowe, "Multiclass object recognition with sparse, localized features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp. 11–18.

[4] T. Serre, L. Wolf, and T. Poggio, "object recognition with features inspired by visual cortex," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[5] A. D. Holub, M. Welling, and P. Perona, "Combining generative models and fisher kernels for object recognition," in *Proceedings of IEEE International Conference on Computer Vision*, 2005.

[6] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: a unifying approach for margin classifiers," *The Journal of Machine Learning Research*, vol. 1, pp. 113–141, 2001.

[7] D. Blei, A. Ng, and M. Jordan, "Latent dirichlet allocation," *The Journal of Machine Learning Research*, vol. 3, pp. 993–1022, January 2003.

[8] G. Csurka, C. Bray, C. Dance, and L. Fan, "Visual categorization with bags of keypoints," in *In Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22.

[9] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local features and kernels for classification of texture and object categories: A comprehensive study," *International Journal of Computer Vision*, vol. 73, no. 2, pp. 213–238, 2007.

[10] M. Marszalek and C. Schmid, "Spatial weighting for bag-of-features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 2118–2125.

[11] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," in *Proceedings of European Conference on Computer Vision*, 2006, pp. 490–503.

[12] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 524–531.

[13] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proceedings of European Conference on Computational Learning Theory*, 1995, pp. 23–37.

[14] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of International Conference on Machine Learning*, 1996, pp. 148–156.

[15] Z. Tu, "Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering," in *Proceedings of IEEE International Conference on Computer Vision*, vol. 2, Beijing, China, Oct 2005, pp. 1589–1596.

[16] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[17] P. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.