ATTRIBUTE-BASED LEARNING FOR LARGE SCALE OBJECT CLASSIFICATION

Worapan Kusakunniran[1,2,5]*, Shin'ichi Satoh*[3]*, Jian Zhang*[4,5]*, Qiang Wu*[4]

[1]The Faculty of ICT−Mahidol University, [2]University of New South Wales, [3]National Institute of Informatics,
[4]University of Technology Sydney, and [5]National ICT Australia

ABSTRACT

Scalability to large numbers of classes is an important challenge for multi-class classification. It can often be computationally infeasible at test phase when class prediction is performed by using every possible classifier trained for each individual class. This paper proposes an attribute-based learning method to overcome this limitation. First is to define attributes and their associations with object classes automatically and simultaneously. Such associations are learned based on greedy strategy under certain conditions. Second is to learn a classifier for each attribute instead of each class. Then, these trained classifiers are used to predict classes based on their attribute representations. The proposed method also allows trade-off between test-time complexity (which grows linearly with the number of attributes) and accuracy. Experiments based on Animals-with-Attributes and ILSVRC2010 datasets have shown that the performance of our method is promising when compared with the state-of-the-art.

Index Terms— Large scale object classification, attribute-based learning, greedy strategy, Bayes' rule, sublinear complexity

1. INTRODUCTION

Classification with a large number of classes becomes more important in the computer vision community. This is because image collections have been significantly growing over time. Meanwhile, the standard one-vs.-all strategy [1] can be inefficient in large scale multi-class classification, because its test-time complexity grows linearly with the number of classes.

Recently, several methods [2][3][4][5][6][7][8] have been proposed to address **sublinear** testing cost for large multi-class tasks. All these methods rely on a tree-based learning. In particular, Bengio et al. [2] proposed an algorithm for learning a label tree of classifiers by optimizing the overall tree loss. Each node of the label tree consisted of a subset of class labels and its corresponding linear classifier. In the classification phase, a given test image started at the root node which contained all class labels, traveled down the tree where a decision was made at each node to determine which branch to follow, and ended at a leaf node which contained a single class label.

For a well balanced tree, the test-time complexity can be reduced from O(DK) to O($D\ log\ K$) where D is the feature dimension and K is the number of classes. The method [2] has been demonstrated to outperform the other tree-based methods including Conditional Probability Tree (CPT) [3] and Filter Tree (FT) [4]. However, it also has several limitations [6]. First, sets of class labels in any two nodes at a same depth of the tree are disjoint. This can sensitively lead to a misclassification when closely related classes are assigned to different nodes at a top layer of the tree. Second, the tree can be unbalanced, which will lead to a suboptimal complexity.

To improve the performance of the method in [2], Deng et al. [6] proposed an algorithm for learning an efficient label tree, which has been shown to be the state-of-the-art. The method [6] simultaneously determined the tree structure and learned the classifiers for the tree nodes. It allowed overlapping of class labels among children of each node. This resulted in an efficient optimization. It also allowed precise control of the trade-off between accuracy and efficiency, which can guarantee balanced trees.

Although the tree-based learning has been reported to be successful in reducing test-time complexity for large scale object classification, it is limited to several practical constraints. First, the tree-based methods cannot be further speeded up by using parallel computing which is a very important and useful methodology to solve a computational problem in real applications. In practice, parallel computing is a way to perform multiple processes simultaneously by using multiple compute resources. Since the tree-based methods rely on a hierarchical structure, multiple classifiers cannot be applied in parallel on a given test image, but in sequence. This is because a selection of classifier used in each level of the tree depends on a decision making of classifier used in previous level of the tree. Second, the tree-based methods consume expensive memory for storing and loading all classifiers at all nodes of the tree in test phase. All classifiers must be presented because it cannot be known beforehand that which classifiers will be used for each given test image. Based on our investigation, the total number of classifiers for each tree is $\geq K$ ($= K$ for a well balanced tree), where K is the total number of classes. This can cause a trouble in memory consumption when K is very large and/or a tree structure is heavily unbalanced.

To avoid these limitations, this paper proposes a new solution from different perspective using an attribute-based learning. It is also shown to achieve comparable performance to the state-of-the-art (i.e. the balanced label tree) [6] regarding sublinear test-time complexity and accuracy. However,

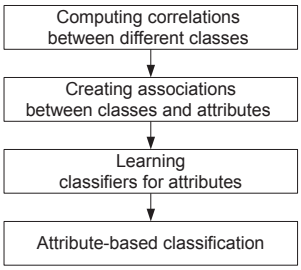

Fig. 1. Attribute-based learning for large scale object classification.

our method can be more flexible and preferable for real applications. This is because its efficiency can be practically enhanced by using parallel computing. Since same classifiers are known beforehand to be used for any test image, multiple classifiers can be applied in parallel on a given test image. Besides, a total number of classifiers used in our attribute-based learning ($\leq K$) is less than used in the tree-based learning ($\geq K$) for all cases. Thus, our method will consume less memory for storing and loading classifiers in test phase.

Fig. 1 shows the framework of the proposed method. A concept of attribute-based classification is adopted for this study to reduce test-time complexity. In previous works [9][10], it was applied for different purpose to detect unseen object classes. Moreover, our method will make significant difference in learning attribute representation by using visual correlations between classes instead of manual human efforts as in [9] or semantic relatedness as in [10]. This makes attributes more related to visual information in images and sequentially leads to the better classification.

Our framework contains three main steps in the training process, as shown in the first three rectangles of Fig. 1. The first step is to compute a similarity matrix (**S**) based on one-vs.-all classification using training and validation datasets. The matrix **S** represents correlations between different object classes. Then, the second step is to build up relations between classes and attributes which are virtually defined based on **S**. Object classes are represented by different sets of attributes and each attribute may belong to several related classes. The third step is to learn a classifier for each attribute instead of each object class. In this way, classes are associated with the trained classifiers by using their attribute representations.

Afterwards, at test phase, the per-attribute classifiers are applied to generate predictions of attribute values for each test sample. The final prediction score of the test sample against each class is calculated from the relevant attribute values.

Based on our proposed method, it can be seen that the test-time complexity grows linearly with the total number of attributes (M) instead of the total number of classes (K). Thus, M can be selected to be much less than K, in order to significantly reduce the complexity.

The rest of this paper is organized as follows. Attribute-based classification is explained in section 2. Constructions of correlations between different classes and associations between classes and attributes are proposed in section 3. Experimental results are shown in section 4 and conclusions are drawn in section 5.

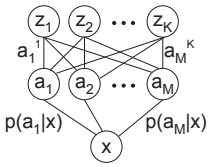

Fig. 2. Direct Attribute Prediction (DAP) for attribute-based classification. $z_1, ..., z_K$ are K object classes. $a_1, ..., a_M$ are M attributes. a_m^k is an association between z_k and a_m. $p(a_m|x)$ is a posterior probability of that a_m being present in an image x, which is estimated from a classifier for a_m.

2. ATTRIBUTE-BASED CLASSIFICATION

Given that there are total M attributes $(a_1, a_2, ..., a_M)$, each class k (z_k) is represented by a set of attributes $A_k = [a_1^k, a_2^k, ..., a_M^k]$ where a_m^k $(1 \leq m \leq M)$ is a binary number. $a_m^k = 1$ means z_k is associated with a_m, otherwise z_k is not associated with a_m. Fig. 2 shows the Direct Attribute Prediction (DAP) suggested by [9]. Attributes are used as in between layer to decouple images from layer of class labels.

In this study, the probabilistic formulation of the DAP in [9] is adopted and revised for our attribute-based classification. It starts by learning a probabilistic classifier β_m for each attribute a_m. β_m is trained by using images of all classes k for which $a_m^k = 1$ as positive training samples and the rest as negative training samples. In this way, β_m can provide a posterior probability $p(a_m|x)$ of that a_m being present in an image x.

β_m models the relation between an image x and an attribute a_m. Next is to consider the relation between an attribute a_m and a class z_k, based on Bayes' rule as:

$$p(z_k|a_m) = \frac{p(a_m|z_k)p(z_k)}{p(a_m)} = \frac{a_m^k}{p(a_m)K} \quad (1)$$

where $p(z_k) = 1/K$ by assuming identical class priors, and $p(a_m|z_k) = a_m^k$ which represents the association between an attribute a_m and a class z_k.

Then, the posterior probability of a class k given an image x can be calculated as:

$$p(z_k|x) = \sum_{m=1}^{M} p(z_k|a_m)p(a_m|x) = \frac{1}{K}\sum_{m=1}^{M} \frac{a_m^k \, p(a_m|x)}{p(a_m)} \quad (2)$$

where $p(a_m) = \frac{1}{K}\sum_{k=1}^{K} a_m^k$ by empirical means over all classes. Equivalently, $p(z_k|x)$ can be estimated in the product form as suggested by [9] as:

$$p(z_k|x) = \sum_{m=1}^{M} p(z_k|a_m)p(a_m|x) = \frac{1}{K}\prod_{m=1}^{M} \left(\frac{p(a_m|x)}{p(a_m)}\right)^{a_m^k} \quad (3)$$

where $p(a_m)$ can be approximated by the empirical means or set to 0.5 [9]. Based on our experiments, equations (2) and (3) yield comparable performances.

Therefore, the best output class label $(\widehat{k})$ for an image x is predicted as:

$$\widehat{k} = \underset{1 \leq k \leq K}{\operatorname{argmax}} \; p(z_k|x) \quad (4)$$

where $p(z_k|x)$ is obtained from equation (2) or (3). By contrast, the standard one-vs.-all classification [1] predicts $p(z_k|x)$ by using the trained per-class classifiers.

Challenge. The remaining challenge is to construct the associations between classes and attributes (a_m^k). The method in [9] uses manual human judgments to seek out the relative strength of these associations. The attributes represent high-level descriptions of object classes such as colors and geometric patterns. To reduce this dependency on human labeling effort, the method in [10] performs mining such associations by measuring their semantic relatedness using linguistic knowledge bases including WordNet, World Wide Web and Web Image Search.

The methods in [9][10] estimate the associations without using any visual information from training image samples. This is because they learn to detect unseen object classes, and thus image samples of test classes are not available in the training process. They also use pre-defined attributes, while our method will automatically learn attributes based on correlations between classes. These lead to the following limitations. First, some high-level concepts describing attributes in [9][10] such as 'smelly', 'fast', 'active' and 'strong' are unrelated/weakly related to visual information in images. Second, 85 attributes are used for 50 classes [9][10]. This cannot reduce the test-time complexity in our study because $M > K$.

On the other hand, this paper does not deal with the problem of unseen object classes. Thus, training samples of all classes will be used as a basis to virtually define attributes and their associations with object classes (see section 3).

3. ATTRIBUTE REPRESENTATION

In this paper, attributes are not used to explicitly represent the high-level descriptors as in [9][10]. Instead, they are learned from relations between object classes. We define a matrix $A \in \{0,1\}^{K \times M}$ to represent the associations between M attributes and K classes. The matrix A can be written in three different forms as:

$$A = \begin{bmatrix} a_1^1 \ldots a_M^1 \\ \vdots \\ a_1^K \ldots a_M^K \end{bmatrix} = [A_1, A_2, ..., A_K]^T = [B_1, B_2, ..., B_M] \quad (5)$$

where A_k is a set of attributes representing a class z_k such that $A_k = [a_1^k, a_2^k, ..., a_M^k] \in \{0,1\}^{1 \times M}$, B_m is a set of classes representing an attribute a_m such that $B_m = [a_m^1, a_m^2, ..., a_m^K]^T \in \{0,1\}^{K \times 1}$, and $A(k,m) = a_m^k$ representing the association between z_k and a_m. As mentioned in section 2, z_k is associated with a_m when $a_m^k = 1$, while z_k is not associated with a_m when $a_m^k = 0$.

Moreover, to reduce the test-time complexity, M must be less than K. However, since attributes are in the binary forms, $M \geq \lceil log_2\ K \rceil$ in order to differentiate all K classes. Thus, the optimal complexity of our method grows by $O(log\ K)$, which is equivalent to the complexity of the perfect balanced tree. M can be selected empirically by trading-off between test-time complexity and accuracy.

Basically, classes that share the same attribute should be highly related. Thus, the matrix A will be determined based on the similarities between object classes using the greedy strategy [11] under certain conditions. This section will discuss three issues: 1) measurement of such similarities (see section 3.1); 2) conditions for the matrix A (see section 3.2); 3) construction of the matrix A (see section 3.3).

3.1. Correlations between different classes

We define a matrix $S \in \Re^{K \times K}$ to represent similarities between object classes. $S(i,j)$, $1 \leq i,j \leq K$, is the similarity of class z_j to class z_i, which can be measured from probability scores of that samples of class z_j being predicted as belonging to class z_i. In this way, the matrix S will be obtained based on one-vs.-all classification in the training process, which can be done offline beforehand.

Given a training dataset, a classifier α_k is learned for each class z_k. Thus, α_k can provide a posterior probability $p(z_k|x)$ of that an image x being predicted as belonging to class z_k. Then, the trained per-class classifiers are applied on a validation dataset containing image samples of all classes. $S(i,j)$ is computed as:

$$S(i,j) = \frac{\sum_{n=1}^{N} p(z_i|x_{n,j})}{N} \quad (6)$$

where $x_{n,j}$ is image sample n of z_j on the validation dataset.

3.2. Conditions for attribute representation

In this study, the attribute representation must be constructed under the following five conditions.

C1: No identical sets of attributes representing different classes (i.e. A_i and A_j are not identical for any $1 \leq i,j \leq K$ and $i \neq j$). In order to differentiate all classes, their attribute representations must be different.

C2: No identical sets of classes describing different attributes (i.e. B_m and B_n are not identical for any $1 \leq m,n \leq M$ and $m \neq n$). Otherwise, it will lead to duplicated classifiers which will increase test-time complexity without additional benefit.

C3: No subset among sets of attributes representing different classes (i.e. A_i should not be a subset of A_j for any $1 \leq i,j \leq K$ and $i \neq j$). This is to avoid a bias classification based on equations (2) and (4). A_i is a subset of A_j if and only if there is at least one attribute a_m ($1 \leq m \leq M$) such that $a_m^j = 1$ and $a_m^i = 0$ and there is not any attribute a_n ($1 \leq n \leq M$) such that $a_n^i = 1$ and $a_n^j = 0$.

C4: Each class must be associated with at least one attribute. That is, for any A_i ($1 \leq i \leq K$), there is at least one attribute a_m ($1 \leq m \leq M$) such that $a_m^i = 1$.

Algorithm 1 Construction of the associations between K classes and M attributes

Input: The similarity matrix $S \in \Re^{K \times K}$
Output: The between-class attribute associations $A \in \{0,1\}^{K \times M}$

Initialize all elements in A to be 1
$while$ (max number of loops is not reached AND there is a change in A)
{
    $A = prune(A, S)$, $A = subset(A, S)$
}
return A

Algorithm 2 $prune(...)$

Input: The matrices A and S
Output: The updated version of A

$while$ (there is a change in A)
{
    for (each z_k)
    {
        $min.sim = 2$
        for (each a_m such that $A(k,m) = 1$)
        {
            $s_m^k = similarity(A, S, k, m)$
            if (set $a_m^k = 0$ then *C1,C2,C4,C5* are held AND $s_m^k < min.sim$)
            {
                $min.sim = s_m^k, min.attr = m, min.class = k$
            }
        }
        $if(min.sim < 2)$
            $A(min.class, min.attr) = 0$
    }
}
return A

Algorithm 3 $subset(...)$

Input: The matrices A and S
Output: The updated version of A

$while$ (there is a change in A)
{
    for (each A_k is a subset of others A_i)
    {
        $max.sim = -1$
        for (each a_m such that $A(i,m) = 0$)
        {
            $s_m^k = similarity(A, S, k, m)$
            if (set $a_m^k = 1$ then *C1,C2,C4,C5* are held AND $s_m^k > max.sim$)
            {
                $max.sim = s_m^k, max.attr = m, max.class = k$
            }
        }
        $if(max.sim > -1)$
            $A(max.class, max.attr) = 1$
    }
}
return A

Algorithm 4 $similarity(...)$

Input: The matrices A and S, the class label k, and the attribute label m
Output: The relative strength s_m^k of the association between class z_k and attribute a_m

$B = \{b \mid A(b,m) = 1 \text{ AND } b \neq k\}$
$s_m^k = \sum_{i=1}^{N_B}[S(k, B(i)) + S(B(i), k)]/[2 \times N_B]$
    where N_B is the number of class labels in B
return s_m^k

C5: Each attribute must be associated with at least one class. That is, for any B_m ($1 \leq m \leq M$), there is at least one class z_k ($1 \leq k \leq K$) such that $a_m^k = 1$.

3.3. Associations between classes and attributes

At this stage, the associations between classes and attributes (i.e. A) are constructed using the similarity matrix S based on the greedy strategy [11] under the certain conditions (i.e. C1-C5), as shown in algorithms 1, 2, 3 and 4. Given K classes, the number of attributes M can be selected by trading-off between accuracy and efficiency (i.e. test-time complexity). The larger M will lead to higher accuracy but lower efficiency. This will be verified in our experiments.

Algorithm 1 presents the main function. It performs iteration on two key steps including $prune(...)$ and $subset(...)$. First, all elements in the matrix A are initialized to be 1. That is, each class is initially associated with all attributes. Then, unnecessary associations are removed in $prune(...)$, and subsets among A_i are eliminated in $subset(...)$. The $while$ loop is performed to seek out the optimum.

Algorithm 2 presents the function $prune(...)$ to iteratively remove weak associations under the conditions *C1, C2, C4* and *C5*. In the function, $min.sim$ represents relative strength of the weakest association between attribute $min.attr$ and class $min.class$. For each existing association ($A(k,m) = 1$), the function $similarity(...)$ is used to calculate the relative strength (s_m^k) between class z_k and all other classes sharing the same attribute a_m, where $s_m^k \in [0,1]$. By greedy technique, the weakest association of each class is eliminated ($A(min.class, min.attr) = 0$), under which the matrix A still can hold the conditions *C1, C2, C4* and *C5*. The $while$ loop is repeated till there is no more unnecessary associations between classes and attributes.

Algorithm 3 presents the function $subset(...)$ to reduce subsets among A_i as described in the condition *C3*, by adding more associations as needed under the conditions *C1, C2, C4* and *C5*. In the function, $max.sim$ represents relative strength of the strongest association between attribute $max.attr$ and class $max.class$ that can help in removing a subset. To cut off any subset (i.e. A_k is a subset of others A_i), we need to find attribute a_m which is not associated with any class z_i (i.e. $a_m^i = 0$) and then link it to class z_k by setting $a_m^k = 1$. By greedy technique, one of such associations that has the maximum relative strength is added ($A(max.class, max.attr) = 1$). The $while$ loop is repeated till there is no more subset that can be removed.

Algorithm 4 presents the function $similarity(...)$ to estimate the relative strength (s_m^k) of the association between

class z_k and attribute a_m. In this study, attribute a_m is presented by the set of classes. Thus, s_m^k can be equivalently calculated from correlations between class z_k and all other classes that share the same attribute a_m.

4. EXPERIMENTS

In our experiments, two datasets are used to evaluate the performance of the proposed method, which include 1) Animals-with-Attributes (AwA) [9] and 2) ILSVRC2010 [12]. Besides, classifiers are trained based on RBF-SVM [13].

4.1. AwA

AwA is used to: 1) roughly expose implicit high-level concepts of attributes; and 2) clearly demonstrate trade-off between accuracy and efficiency of the proposed method. It consists of 30475 images from 50 animal classes. We use 20 % for training, 10 % for validation, and the rest 70 % for testing. The provided pre-computed features are used in our experiments, which include RGB color histograms, local self-similarity histograms, SIFT, rgSIFT, PHOG, and SURF. Thus, the total feature dimension is 10940.

4.1.1. Implicit high-level concepts of attributes

In this paper, attributes are represented by using sets of classes, without any pre-defined explicit high-level concept as used in [9][10]. Instead, they are obtained through the learning processes as explained in sections 3.1, 3.2 and 3.3. However, to reveal high-level meanings of our learned attributes, they can be implicitly matched against high-level concepts by investigating common visual features among classes sharing the same attribute.

As an example, we discuss three attributes (a_1, a_2, a_3) learned by using our method on AwA dataset where a total number of attributes is 20. a_1 is associated with classes 'dalmatian', 'siamese cat', 'skunk', 'chihuahua', 'weasel'. a_2 is associated with classes 'persian cat', 'siamese cat', 'hamster', 'chihuahua', 'rat', 'weasel', 'mouse'. a_3 is associated with classes 'gorilla', 'chimpanzee', 'giant panda'.

a_1 can be linked to concepts of 'black' and 'white'. However, 'panda' and 'zebra' also contain black and white colors but do not share a_1 because global shape structure of 'panda' and stripe pattern of 'zebra' are too different from other classes sharing a_1. a_2 can be linked to concepts of 'whisker', 'triangular ears', 'big eyes', and 'small'. Other classes such as 'lion', 'tiger' and 'bobcat' do not share a_2, although they also have whisker, triangular ears, and big eyes. This is because their size is big. a_3 can be linked to concepts of 'black', 'tree', and 'big'. The class 'spider monkey' is also one type of monkeys as similar to 'gorilla' and 'chimpanzee', but it does not share a_3. This is because its main colors are brown and grey and its size is small.

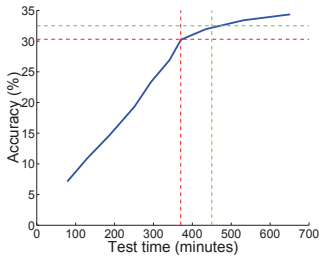

Fig. 3. The trade-off between the test-time and the accuracy using the proposed method.

Many concepts mentioned above such as 'black', 'white', 'small', 'big', 'tree' are also used in [9][10]. Moreover, as discussed in this section, it can be seen that each attribute can be well linked to multiple high-level concepts. In contrast, each attribute in [9][10] is represented by a single high-level concept. This is the main reason why the methods in [9][10] use $M(=85) > K(=50)$. In our study, $M < K$ in order to reduce test-time complexity. Thus, each attribute must be implicitly linked to multiple high-level concepts.

4.1.2. Trade-off between accuracy and efficiency

Since this dataset contains $K = 50$ classes, the minimum number of attributes is $\lceil log_2\ 50 \rceil = 6$. In this section, experiments have been carried out based on 10 different numbers of attributes (i.e. $M = 6$, 10, 15, 20, 25, 30, 35, 40, 45, 50). Figure 3 illustrates the trade-off between the test-time and the accuracy. It can be seen that the accuracy grows faster than linear (i.e. approximately polynomial) with the test-time which only grows linearly with the number of attributes.

When $M = K$ (i.e. 50 attributes for 50 classes), the proposed method is equivalent to the one-vs.-all classification [1] regarding both accuracy and test-time complexity. The one-vs.-all scheme has been confirmed to be as accurate as any other more complicated schemes [1]. So, it is used as a baseline in this study.

In our method, a proper M can be selected to significantly reduce test-time complexity but only slightly reduce accuracy when compared with the one-vs.-all strategy. Two examples are shown in Figure 3. In the first example shown as the green dashed lines, when $M \approx 35$, the test-time complexity is reduced by 30 % but the accuacy is reduced by 2% only. In the second example of a better efficiency shown as the red dashed lines, when $M \approx 30$, the test-time complexity is reduced by almost 50 % but the accuacy is reduced by just 4 %. However, selecting the number of attributes can be properly decided by programmers based on natures of applications and datasets.

4.2. ILSVRC2010

ILSVRC2010 is used for a fair comparison with the state-of-the-art [6] on large scale object classification. It includes a large number of classes i.e. 1k classes, which contains 1.2M images for training, 50k images for validation, and 150k im-

Table 1. Large scale object classification on ILSVRC2010.

Method	Case 1		Case 2		Case 3	
	Accuracy (%)	Test speedup	Accuracy (%)	Test speedup	Accuracy (%)	Test speedup
Label tree [2]	8.33	10.3	5.99	15.2	5.88	9.3
Balanced label tree [6]	11.90	10.3	8.92	18.2	5.62	31.3
The proposed method	12.13	10.3	8.07	18.2	5.81	31.3

ages for testing. We use the pre-computed SIFT on a 10k entry codebook and use a two level spatial pyramid (1x1 + 2x2) to obtain a 50k dimensional feature vector.

As shown in Table 1, the proposed method is compared with the efficient label tree-based methods [2][6]. The method in [6] has been shown to achieve state-of-the-art performance in testing for large scale object classification. We follow the same experiments as in [6]. There are three different cases where test speedups are 10.3, 18.2, and 31.3 respectively. The test speedup is defined as the speedup of the test cost compared to the baseline (i.e. one-vs.-all classification). In our method, given a test speedup (t), $M = K/t$. Thus, our experiments have been carried out based on the three cases by using three different numbers of attributes which are 97, 55, and 32 respectively.

From Table 1, it can be seen that the performance of the proposed method is comparable to the state-of-the-art [6]. Thus, our method can be considered as an efficient alternative to the tree-based methods, and it can be more preferable in real applications due to practical advantages as mentioned in the introduction. That is, the test speedup of our method can be further improved by using parallel computing. Also, our method uses much smaller number of classifiers in test phase. This leads to smaller memory consumption for storing and loading the classifiers. In the proposed method, the number of classifiers (C) equals to the number of attributes. Thus, we use 97, 55, and 32 classifiers in cases 1, 2, and 3 respectively.

On the other hand, in the tree-based methods [2][6], C equals to the number of nodes in the tree which varies on the number of children (Q) for each node and the maximum depth (H) of the tree. For a well balanced tree, $C \approx \sum_{h=1}^{H} Q^h$. Two parameters Q and H are fixed in the experiments [2][6] as case 1: $Q = 32, H = 2$, case 2: $Q = 10, H = 3$, and case 3: $Q = 6, H = 4$. Thus, they use approximately 1056, 1110, and 1554 classifiers in cases 1, 2, and 3 respectively.

5. CONCLUSION

This paper has proposed a new framework for large scale object classification by using attribute-based learning. Object classes are represented by sets of attributes which are virtually defined in a middle layer between class labels and images. Associations between attributes and classes are learned by using classical greedy strategy based on correlations between classes. Then, per-attribute classifiers are trained instead of per-class classifiers as in standard one-vs.-all scheme. Finally, the trained classifiers are applied on a test image to predict its class label based on the learned attribute representations. In order to reduce test-time complexity, a number of attributes can be selected to be less than a number of classes. That is, trade-off between test-time complexity and accuracy can be precisely carried out. Experimental results have demonstrated efficiency of the proposed method compared to the existing methods based on tree structure.

6. REFERENCES

[1] R. Rifkin and A. Klautau, "In defense of one-vs-all classification," *Journal of Machine Learning Research*, vol. 5, pp. 101–141, January 2004.

[2] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," Canada, December 2010, Advances in Neural Information Processing Systems (NIPS), pp. 163–171.

[3] A. Beygelzimer, J. Langford, Y. Lifshits, G. Sorkin, and A. Strehl, "Conditional probability tree estimation analysis and algorithms," Canada, June 2009, Conf. on Uncertainty in Artificial Intelligence, pp. 51–58.

[4] A. Beygelzimer, J. Langford, and P. Ravikumar, "Multiclass classification with filter trees," Preprint (June 2007).

[5] J. Deng, A. Berg, K. Li, and L. Fei-Fei, "What does classifying more than 10,000 image categories tell us?," Greece, September 2010, European Conf. on Computer Vision, pp. 71–84.

[6] J. Deng, S. Satheesh, A. Berg, and L. Fei-Fei, "Fast and balanced: efficient label tree learning for large scale object recognition," Spain, December 2011, Advances in Neural Information Processing Systems (NIPS), pp. 567–575.

[7] G. Griffin and P. Perona, "Learning and using taxonomies for fast visual categorization," United States of America, June 2008, IEEE Int. Conf. on Computer Vision and Pattern Recognition, pp. 1–8.

[8] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: a large data set for non-parametric object and scene recognition," *IEEE Tran. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, November 2008.

[9] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," United States of America, June 2009, IEEE Int. Conf. on Computer Vision and Pattern Recognition, pp. 951–958.

[10] M. Rohrbach, M. Stark, G. Szarvas, I. Gurevych, and B. Schiele, "What helps where - and why? semantic relatedness for knowledge transfer," United States of America, June 2010, IEEE Int. Conf. on Computer Vision and Pattern Recognition, pp. 910–917.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to algorithms*, chapter 16, MIT Press, 3^{rd} edition, July 2009.

[12] A. Berg, J. Deng, and L. Fei-Fei, "Imagenet large scale visual recognition challenge 2010," `http://www.image-net.org/challenges/LSVRC/2010/`.

[13] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 27:1–27:27, 2011, Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.