Journal of Computer Science and Cybernetics, V.32, N.2 (2016), 133–152
DOI no. 10.15625/1813-9663/30/2/7574

USING SUM MATCH KERNEL WITH BALANCED LABEL TREE FOR LARGE-SCALE IMAGE CLASSIFICATION

TIEN-DUNG MAI

University of Information Technology,
Vietnam National University-Ho Chi Minh City
dungmt@uit.edu.vn

Abstract. Large-scale image classification is a fundamental problem in computer vision due to many real applications in various domains. One of the popular methods is to train one-versus-all binary classifiers independently for each class. Although this method is simple, they are impracticable in the case of a large number of classes because their testing complexity grows linearly with the number of classes. Another is to organize classes into a hierarchical tree structure. The number of classifier evaluations of a test sample when traveling from the root to a leaf node is significantly reduced. A challenging issue is how to learn a tree structure which achieves both classification accuracy and computational efficiency. The current methods use a confusion matrix and spectral clustering techniques to group confusing classes into clusters associated with the nodes. However, training one-vs-all classifiers used to calculate the confusion matrix is costly for a large number of classes. Moreover, the output tree might not be balanced because the objective function of spectral clustering penalizes unbalanced partitions. In this paper, we suggested a novel method to learn the tree structure by using a spectral clustering algorithm and a similarity matrix. Here, the similarity between two classes is exactly measured by the sum-match kernel. In addition, a feature map is used to reformulate the sum-match kernel function as a dot product of two mean feature vectors in a mapped-feature space. Furthermore, we proposed an algorithm for learning a balanced tree which gains the computational efficiency in classification. We carried out experiments on benchmark datasets including Caltech-256, SUN-397, and ImageNet-1K. The evaluation results indicated that our method achieves a significant improvement in terms of accuracy and efficiency compared to other methods. In particular, our method achieved 14.52% in accuracy on ImageNet-1K, compared to 6.51% of the Bengio et al.'s method.

Keywords. Large-scale image classification, multi-class classification, label tree-based classification, sum-match kernel.

1. INTRODUCTION

In this paper, we tackle the problem of large-scale image classification - classifying an image belonging to one of a large number of target categories or classes. This is one of the fundamental problems that has been recently receiving significant interest in computer vision. The reason is that there are many real applications such as visual recognition, semantic image retrieval [5, 8, 9], and scene understanding [16, 18, 35], which require the classifier to discriminate multiple categories on large image databases. For example, SUN dataset [35] contains 130,519 images of 899 scene categories and ImageNet [26] contains 14,197,122 images of 21,841 categories.

The One-versus-All (OvA) [1, 25] is one of the most commonly used methods for multi-class classification problem in which each class corresponds to one binary classifier. However, with a large number of class labels, this method is not practical because all classifiers have to be evaluated for every testing image at run-time classification. Hence, the testing complexity grows linearly with the number of class labels.

The error-correcting output codes (ECOC) [2,38] used the embedding of binary classifiers and corresponding codewords for classification. Although the testing complexity of these methods was reduced, it is very difficult to design a coding matrix that satisfies both high accuracy and low computational cost in the case of large-scale datasets.

A recent approach reduces the testing complexity to a sub-linear with the number of class labels by exploiting a hierarchical structure in the label space [3, 10, 19]. The key idea is to organize classes into a label tree structure in which each leaf node is assigned a class label, and each internal node is associated with a set of class labels and an OVA classifier for determining which child node to follow. For classifying, we traverse the tree from the root until a leaf node is reached, the label class of the leaf node is predicted as the class label of the test sample. By using a small number of classifiers along the path, the testing complexity is sub-linear to the number of class labels.

There are two main tasks for a label tree-based classification approach: *learning the tree structure and learning classifiers at internal nodes*. The current methods [3, 10, 12, 19] either separate or combine these two tasks in a joint optimization framework.

Despite the combining methods usually have higher classification accuracy, their learning cost is too costly because the classifiers have to be trained multiple times until the solution is converged. Hence, in this paper, we follow the methods in which these two tasks are learned separately, as [3], and focus on the task of learning the tree structure.

The popular learning methods are to use clustering algorithms (e.g., k-means, spectral clustering) to recursively partition a set of class labels into subsets such that classes which are easily confused or highly similar should be grouped in the same subset. Each subset corresponds to a node of the tree, the root contains all class labels, and the leaf node contains a single class label. For example, the methods (e.g., Bengio et al. [3], Griffin and Perona [14], Wang and Forsyth [34]) used a spectral clustering algorithm and a confusion matrix which measures a confusion among classes. First, training OVA classifiers for all classes, and evaluating these classifiers on a validation set to obtain a confusion matrix. After that, the spectral clustering is used to recursively split the classes into groups. However, these methods have several limitations. First, training all OVA classifiers is costly for a large number of classes. Second, the confusion between two classes is not reliable when the OvA classifiers have poor accuracy due to a small number of available training samples and the curse of dimensionality. Third, the tree structure may be unbalanced because the objective function of spectral clustering does not take into account the size of groups. Consequently, the testing complexity may not achieve the maximum efficiency. Another method is to use k-means clustering algorithm on training samples [20]. In this method, the mean of all feature vectors of the training samples of a class is used as a representative of that class. However, using the mean is not effective for classes with large variations, and the resulting tree is not always balanced.

The above-mentioned problems are addressed as follows. First, we used the spectral clustering and a similarity matrix to learn the tree structure. To measure the similarity between classes, we used a sum-match kernel instead of having to train OvA classifiers as the prior work. Second, by using the feature map [32], we reformulated the sum-match kernel function in the original feature space as a dot product of two mean feature vectors in a mapped-feature space. Finally, we proposed an algorithm for learning a balanced tree which gains the computational efficiency while maintaining the classification accuracy. Experimental results on large-scale datasets, including Caltech-256, SUN-397, and ImageNet-1K, have shown that our method outperforms other state-of-the-art methods in terms of accuracy and efficiency.

Generally, this study introduce a novel approach to learn balanced trees for hierarchical classification. Compared to current state-of-the-art tree-based systems, we propose to use similarity of classes instead of their confusion scores. There are two main benefits of such an approach. First, the similarity of classes can be computed with low computational cost. Second, experimental results have shown that using class similarity results in higher classification accuracy (e.g., our method achieved 42.96% in accuracy using CNN feature on ImageNet-1K, compared to 24.97% of the confusion metric based method of Bengio et al, as shown in Fig.2(a)). This is mainly due the precisely built and balanced tree structure. However, there is a trade-off. By using a balanced tree for classification, we significantly reduce testing cost since the number of evaluation needed for each sample (i.e., new image) decreases, compared to flat-based methods. But, this also causes accuracy drop. For example, we achieve 50 times speed-up with the accuracy drop from 57.09% to 42.96% on ImageNet1K. Such a trade-off should be carefully considered in real-life application.

The remainder of the paper is organized as follows: Section 2. introduces related work. Section 3. describes the proposed method for generating a similarity matrix using sum-match kernel and learning a balanced tree. Section 4. shows the experimental results. In Section 5., the advantages and disadvantages of the proposed method are under discussion. Finally, Section 6. presents the conclusion and further research.

2. RELATED WORK

Multi-class classification problem in large-scale datasets is a challenging and interesting problem. One of the popular approaches is to decompose multi-class to binary classification problems such as One-versus-All (OvA) [25]; or conversely to combine binary classifiers toward a multi-class problem such as Error-Correcting Output Coding (ECOC) [2].

In OvA method, one binary classifier is independently trained for each class. For class i^{th} it assumes i-labels as positive while the rest is negative. In classification, all classifiers have to be evaluated on a test sample, the class label corresponding the highest score is assigned to the test sample. Although this method shows good classification accuracy [1], its testing complexity scale linearly up the number of classes. Therefore, it becomes impractical with a large number of classes.

In contrast to OvA, ECOC combines several binary classifiers to classify a test sample. In this method, the main task is to design an optimal coding matrix $N \times L$, where N is the number of class labels and L is the desired number of binary classifiers. Each row corresponds to a unique codeword which is associated with a class. Each column corresponds to a binary classifier. In classification, all L binary classifiers are evaluated to obtain an output codeword. The class whose codeword is closest to the output codeword is assigned to the test sample. In Spectral ECOC [37], the coding matrix is designed basing on the eigenvectors of the normalized Laplacians of the similarity graph of the classes. In Sparse Output Coding [38], the optimal coding matrix and binary classifiers is learned separately basing on semantic similarity between classes using training data and class taxonomy. However, when the number of class labels is large, it is extremely difficult to design a coding matrix that ensures two properties: each row is a unique codeword for robustness and the number of binary classifiers L is minimized for computational efficiency.

The hierarchical classification is one of the efficient approaches to reduce the testing complexity to a sub-linear with the number of class labels while maintaining reasonable classification accuracy. The main idea is to exploit a hierarchical structure in the label space for organizing classes into a tree [3, 10, 12, 19, 39]. The root contains all classes, each internal is associated with a subset of classes and each leaf node is associated with a single class. For classifying with a given tree, test examples traverse from the root until a leaf node is reached. Therefore, for a well balanced tree, the number of classifier evaluations on the path is logarithmic with the number of classes.

Bengio et al. [3] used a confusion matrix to measure confusion and a spectral clustering to recursively partition a set of class labels into disjoint groups. Each group corresponds to a child node of the tree. The confusion matrix is generated by applying OvA binary classifiers to a validation set. Due to the objective function of spectral clustering penalizes unbalanced partitions, the result is implicitly a balanced tree. However, this method is not reliable for a large number of classes. The reasons are as follows: training OvA classifiers is too costly, and the confusion among classes is not exactly estimated when the corresponding OvA classifiers have poor accuracy due to the curse of dimensionality.

Deng et al. [10] proposed a method that combines class partitioning and classifier learning for each child node in an optimization problem. The problem is solved by alternating between two optimization steps. However, learning cost is high because the classifiers have to be trained multiple times until the solution is converged. Moreover, by allowing overlapping of classes among child nodes to reduce false navigation, it increases the testing cost.

The relaxed hierarchy method proposed by Gao and Koller [12] is an alternative based on max-margin optimization in which a subset of confusing classes is allowed to be ignored at each node. This method shares the idea of the method proposed by Marszalek and Schmid [22], but has significant improvements over it. However, learning complexity increases if there are more than two branches at each node.

Sun et al. [29] considered the classification problem as finding the best path in the label tree and proposed a branch-and-bound-like algorithm. The bounds and the classifiers are jointly learned using structured SVM formulation with additional bound constraints for a trade-off between efficiency and accuracy.

Wang and Forsyth in [34] proposed a method to aggregate the probability distribution associated with a leaf node of trees in a label forest i.e. an ensemble of label trees. The $(i+1)$-th label tree is constructed by applying a method of Bengio et al. [3] with a confusion matrix computed for the i-th label tree on a validation set. Although this method improved classification accuracy, computational cost can be significantly increased if a large number of label trees are used.

Liu et al. [19] proposed a probabilistic approach for learning a tree structure. Each node of the probabilistic label tree is associated with a categorical probability distribution and a maximum likelihood classifier defined as a multinomial logistic regression model. The training process at each node is formulated as a maximum optimization of a log likelihood function, which is then solved by using alternating convex optimization.

Recently there are some interesting results in which the features and classifiers are jointly learned using a deep learning architecture [7, 15, 28]. Although these methods archived the excellent results in large-scale classification, they require the modest computational resources and GPU programming skills.

In this work, we follow the methods in which learning the tree structure and learning node classifiers are learned separately. Specifically, the spectral clustering algorithm and similarity matrix are used to build a tree structure. Moreover, we proposed an algorithm for learning a balanced tree aiming to gain the computational efficiency while maintaining reasonable classification accuracy.

3. OUR APPROACH

In this section, we first represent a technique for generating a similarity matrix among classes using the sum-match kernel in Section 3.1. We then describe an algorithm for learning a balanced label tree structure in Section 3.2..

3.1. Generating the similarity matrix

A similarity matrix among classes can be used in a spectral clustering algorithm to partition these classes into groups so that the classes in the same group are more highly similar and the classes in different groups are less similar. Thus, the similarity between two classes is measured more exactly, the clustering algorithm achieves higher accuracy. In this study, the sum-match kernel is used to measure the similarity. The reason is this measure recently has been achieved the effective results in evaluating the similarity between sets of local features [4, 36, 38].

Given a set of class labels $L = \{c_1, ..., c_N\}$, a similarity matrix $S_{N\times N}$ is a symmetric matrix whose element $S_{i,j}$ represents the similarity measurement between two class labels c_i and c_j. Let $f_{i,p}$ and $f_{j,q}$ be the feature vectors of corresponding images of class c_i and class c_j. Then, the similarity between classes c_i and c_j is defined by summing the local kernels between every pair of feature vectors of class c_i and class c_j:

$$S_{i,j} = \frac{1}{n_i}\frac{1}{n_j}\sum_{p=1}^{n_i}\sum_{q=1}^{n_j}\hbar(f_{i,p}, f_{j,q}), \tag{1}$$

where $\hbar(.)$ is a Mercer kernel function; n_i and n_j are the total number of images in class c_i and c_j, respectively. By this way, the similarity matrix $S_{N\times N}$ can be built without the need of OvA classifiers training as the prior work. Eq.(1) requires to compute the kernel function $\hbar(.,.)$ for all feature vectors, so the computational cost is too expensive when the number of images of classes is large or the number of class labels can be in the thousands of classes. This is also a limitation of our previous work [21]. In this work, we introduce an approach to deal with this problem by applying recently developed feature map [32] that is described in Section 3.1.1. and 3.1.2..

3.1.1. Explicit feature mapping

Relying on a property of reproducing kernel Hilbert spaces [27], it is guaranteed that there exists a function φ mapping the data x into a Hilbert space $\mathcal{H}$ for any positive definite kernel function $\hbar(x,y)$, such that

$$\hbar(x,y) = \langle \varphi(x), \varphi(y) \rangle, \tag{2}$$

where $\varphi(x)$ and $\varphi(y)$ are the mapped data point of x and y in the Hilbert space, and $\langle \varphi(x), \varphi(y) \rangle$ denotes the inner product between $\varphi(x)$ and $\varphi(y)$.

Moreover, following [32], if $\hbar(x,y)$ is an additive kernel, known as a homogeneous kernel (e.g., the Hellinger's, χ^2, intersection, and Jensen-Shannon), a suitable feature map φ can be explicitly constructed for sufficiently approximating it to a linear kernel. This allows using $\hbar(x_i, y_j) = \langle \varphi(x_i), \varphi(y_j) \rangle$, where $\varphi(x_i)$ and $\varphi(y_j)$ correspond the mapped data point x_i of the feature vector x and y_j of the feature vector y, respectively.

3.1.2. Sum-match linear kernel

Given the explicit feature map, an additive kernel function in the original feature space can be approximated by a linear kernel function in the mapped-feature space. As a result, we have $\hbar(x,y) = \langle \varphi(x), \varphi(y) \rangle = \varphi(x)^T \cdot \varphi(y)$. Then, the value of $S_{i,j}$ in Eq.(1) can be written as follows:

$$\begin{aligned}
S_{i,j} &= \frac{1}{n_i}\frac{1}{n_j}\sum_{p=1}^{n_i}\sum_{q=1}^{n_j}\hbar(f_{i,p}, f_{j,q}) = \frac{1}{n_i}\frac{1}{n_j}\sum_{p=1}^{n_i}\sum_{q=1}^{n_j}(\varphi(f_{i,p})^T \cdot \varphi(f_{j,q})) \\
&= \frac{1}{n_i}\frac{1}{n_j}\Big[(\varphi(f_{i,1})^T \cdot \varphi(f_{j,1}) + \cdots + \varphi(f_{i,1})^T \cdot \varphi(f_{j,n_j})) + \cdots + \\
&\qquad (\varphi(f_{i,n_i})^T \cdot \varphi(f_{j,1}) + \cdots + \varphi(f_{i,n_i})^T \cdot \varphi(f_{j,n_j}))\Big] \\
&= \frac{1}{n_i}(\varphi(f_{i,1}) + \cdots + \varphi(f_{i,n_i}))^T \cdot \frac{1}{n_j}(\varphi(f_{j,1}) + \cdots + \varphi(f_{j,n_j})) \\
&= \tilde{\varphi}_i^T \cdot \tilde{\varphi}_j,
\end{aligned} \tag{3}$$

where $\tilde{\varphi}_i = \frac{1}{n_i}(\varphi(f_{i,1}) + \cdots + \varphi(f_{i,n_i}))$ and $\tilde{\varphi}_j = \frac{1}{n_j}(\varphi(f_{j,1}) + \cdots + \varphi(f_{j,n_j}))$ are mean feature vectors that are computed by averaging the mapped-feature vectors of class c_i and c_j, respectively. Consequently, $S_{i,j}$ can be computed as a dot product of two mean feature vectors.

3.2. Learning a balanced label tree structure

The motivation of the label tree approach is classification efficiency, which is measured in terms of the average number of operations needed to produce a final label for a new sample. The efficiency will be maximized when the tree structure is balanced [10, 19].

The process of learning a tree structure can be presented as a clustering problem which splits a set of N class labels of node v into Q clusters. Each cluster corresponds to a child node of v, and Q is the desired number of children per node. However, the objective function of clustering algorithm (e.g., spectral clustering [23]) does not take into account the size of clusters, and thus the output tree might not be balanced.

Although the constrained clustering algorithms may be applied, the balancing constraints are formulated as a linear programming that is known to be an NP-complete problem in practice [10]. In this work, to overcome this drawback, we propose an algorithm for balancing the number of class labels in each cluster.

According to [23], let $\mathcal{L}$ be the Laplacian matrix of the similarity matrix $S_{N\times N}$ (e.g., $\mathcal{L} = \mathcal{D}^{-1/2}S\mathcal{D}^{-1/2}$, where $\mathcal{D}$ is the diagonal matrix whose (i,i)-element is the sum of S's i-th row). The Q largest eigenvectors $e_1, ..., e_Q$ of $\mathcal{L}$ are formed into the matrix $Y = [e_1 e_2 \quad e_Q] \in \mathbb{R}^{N\times Q}$. Each row $y_i \in Y$ is treated as an eigen-based feature vector of class c_i. The k-means algorithm is applied to partition Y into Q clusters. A class c_i is assigned to a child node j^{th} if and only if an item y_i is assigned to the cluster j^{th}.

To create a balanced tree structure, at each node v, its child node has at most $P_{max} = Q^{H-1}$ class labels, where $H = \log_Q(N)$ is the maximum depth of node v to the root, and N is the number of class labels of node v. So, if the child node j^{th} has more than P_{max} class labels, several its classes have to be moved to others. This is equal to moving several eigen-based feature vectors in the cluster j^{th} to other clusters.

The process for balancing the number of class labels in the child nodes basing on the eigen-based feature vectors at the node v is summarized by Algorithm 1.

To learn the label tree structure, the similarity matrix $S_{N\times N}$ is firstly computed with the given set of N class labels $L = \{c_1, ..., c_N\}$ as described in Section 3.1. For each non-leaf node v, beginning from the root, we rebuild a similarity matrix S_v basing on the matrix S with a set of class labels L_v of the node v. Then, the spectral clustering algorithm [23] - implemented as $[Y, G, A] = SpectralClustering(L_v, S_v, Q, P_{max})$ - is applied to the matrix S_v to partition the set of class labels L_v into Q clusters. The maximum number of classes in each cluster is P_{max}. The outputs of this function are three sets Y, G, and A. Here, Y is a set of $|L_v|$ eigen-based feature vectors, $Y = \{y_1, ..., y_{|L_v|}\}$. G is a set of Q cluster centers, $G = \{g_1, ..., g_Q\}$. And, A is a set of $|L_v|$ items, $A = \{a_1, ..., a_{|L_v|}\}$. $a_i = k$ indicates that the class c_i is assigned to the cluster g_k.

Finally, these outputs are used as the inputs of the Algorithm 1 for balancing the number of class labels in the child nodes of the node v. This learning process is performed repeatedly for each node until the tree structure is completely built. The algorithm for clustering at every node of the label tree is summarized in Algorithm 2.

Following the notation in [10,19], a Q-way balanced label tree is denoted as $T_{Q,H}$, where H is the maximum depth and Q^H approximate the number of class labels. Notice that it is unable to partition into Q child nodes with less than Q class labels.

Algorithm 1 $[A] = Balancing(Y, G, A, P_{max})$: balancing the number of class labels in the child nodes of the node v

```
Input:
1:     • Set $Y = \{y_1, .., y_N\}$ of $N$ items correspond to eigen-based feature vectors of class
         labels at node $v$;
2:     • Set $G = \{g_1, ..., g_Q\}$ of $Q$ cluster centers that were obtained from spectral clustering;
3:     • Set $A = \{a_1, ..., a_N\}$ of $N$ items, each item $a_i = k$ indicates that class $c_i$ is assigned
         to cluster $g_k$;
4:     • $P_{max}$: the maximum number of class labels in a cluster;
Output: Set $A = \{a_1, ..., a_N\}$ contains information about the assignment of class labels into
    $Q$ children. $a_i = k$ means that class label $c_i$ is assigned to child node $k^{th}$. For each child
    node, the size of its set of class labels is at most $P_{max}$.
5:  Step 1:
6:     • Let $R$ be set of clusters whose number of items is greater than $P_{max}$.
7:     • Let $T$ be set of clusters whose number of items is less than $P_{max}$.
8:     • Let $D$ be set of items that will be assigned to clusters in $T$: $D = \emptyset$
9:  Step 2: For each cluster in $R$, we only hold $P_{max}$ items whose distance to its cluster
    center is minimum in the 2-norm. The remaining items are added to $D$.
10: Step 3:
11: while $D \neq \emptyset$ do
12:     $y_i \leftarrow D$
13:     Assign $y_i$ to cluster $t_j \in T$ such that the distance from center $g_j$ to $y_i$ is minimum in
    the 2-norm: $t_j = t_j \cup \{y_i\}$
14:     Update center $g_j$: compute $g_j$ as the mean of all items assigned to cluster $t_j$.
15:     if $|t_j| = P_{max}$ then                    ▷ the number of items of $t_j$ equals $P_{max}$
16:         $T = T \setminus \{t_j\}$
17:     end if
18: end while
```

Algorithm 2 $[A] = Clustering(L_v, S_{N\times N}, Q, P_{max})$: for clustering the set of class labels L_v into Q children.

```
Input:
1:     • $L_v$ : the set of class labels of node $v$;
2:     • $S_{N\times N}$: the similarity matrix among $N$ classes;
3:     • $Q$: the number of children per node;
4:     • $P_{max}$: the maximum number of class labels in a child node;
Output: Set $A = \{a_1, ..., a_N\}$ contain information about the assignment of class label into
    $Q$ children. $a_i = k$ means that class label $c_i$ is assigned to child node $k^{th}$. For each child
    node, the size of its set of class labels is at most $P_{max}$.
5: Step 1: Compute $S_v$ matrix basing on the similarity matrix $S$ and $L_v$.
6: Step 2: Cluster $[Y, G, A] = SpectralClustering(L_v, S_v, Q, P_{max})$
7: Step 3: Balance the number of class labels: $[A] = Balancing(Y, G, A, P_{max})$
```

4. EXPERIMENTS

4.1. Experimental setting

4.1.1. Datasets

The experiments are carried out on benchmark datasets which are usually used to evaluate large-scale image classification approaches.

- **Caltech-256** [13]. There are 29,780 images of 256 object classes. Each image is assigned to a single class label, each class contains at least 80 images. Images are different lighting conditions, poses, sizes, and resolutions.

- **SUN-397** [35]. This is a subset of the SUN dataset. It is selected from 908 scene classes used for a scene recognition. There are 108,754 images of 397 classes, at least 100 images per class. The collection of images covers a large variety of environmental scenes, places, and the objects within.

- **ImageNet-1K** [26]. The dataset contains 1,461,406 images of 1,000 classes (e.g., animal, plant, artifact, events, people). Each class contains at least 668 images.

With SUN-397 and Caltech-256, we randomly picked 50% of the images for training, 25% images for validation and the remaining for testing. Meanwhile, on ImageNet-1K, we used the provided image sets for validation with a total of 50,000 images, each class includes 50 images. For testing, there are 150,000 images, each class includes 150 images. We randomly pick 100 images from each class for training. In our experiments, the validation set is used to compute the confusion matrix, similar to [3]. The training set is used to obtain the similarity matrix represented in Section 3.1., and to train the OvA classifiers as well as the classifiers at the non-leaf nodes.

4.1.2. Image descriptors

Besides standard features for a fair comparison with previous methods, the state-of-the-art feature i.e., deep features is also applied to investigate the influence of feature selection. Particularly, two types of features are applied, including:

- **SIFT+LLC+SPM feature.** In order to compare our method with the others, e.g., those of Bengio et al. [3] and Deng et al. [10], we used the same feature settings as in [10]. Specifically, we extract dense SIFT feature for each image by VLFeat toolbox [30]. These features are then encoded using the Locality-constrained Linear Coding (LLC) approach described in [33]. The code book consists of 10,000 visual words generated by using k-means with images randomly selected from the dataset. Each image is encoded using a two-level Spatial Pyramid Matching (SPM) [17] with 1×1 and 2×2 grids. The results are feature vectors with 50,000 dimensions.

- **CNN feature.** We used the state-of-the-art deep feature for image representation. Following settings that have been widely used in recent work [6, 24, 28], we used MatConvNet toolbox [31] with the VGG-VERYDEEP-16 model. The model is pre-trained on the ILSVRC-2012 dataset with 16 layers (13 convolutional layers and three fully-connected layers) [28]. The output of the network at the fc7 layer with 4,096 channels was widely used as the feature vector of an input image.

4.1.3. The baseline methods

In this section, we briefly describe baseline methods that are used for comparison.

- First is the label embedding tree of Bengio et al. [3]. This method achieved state-of-the-art results in testing with the tree structure learned by using a confusion matrix. N binary OvA classifiers were trained independently, where N is the number of class labels. These classifiers are then evaluated on a validation set to compute a confusion matrix $\bar{C}$ and an affinity matrix $A = \frac{1}{2}(\bar{C} + \bar{C}^T)$. Starting from the root, the spectral clustering algorithm [23] is applied to the affinity matrix A to partition the set of class into subsets. Each subset corresponds to a child node. This process is recursively repeated until the label tree is completely built.

- Second is the SVM tree of Liu et al. [20]. This method used the mean feature vectors of classes to build a binary SVM tree. We re-implemented this method for comparison. In particular, each class is represented by a mean feature vector obtained by averaging all feature vectors in this class. In contrast to [20], the k-means algorithm was used for partitioning the set of mean vectors into Q clusters to build a Q-ways tree instead of the original binary tree.

- Last is the Fast-Balanced Tree of Deng et al. [10]. This method combined tree construction and classifier learning at each node in an optimizing framework. Due to the insufficient description, we cannot re-implement the method, we used the experimental results on ImageNet-1K with the SIFT+LLC+SPM feature for comparison.

To make a fair comparison, the LIBLINEAR library [11] is used to train all OvA classifiers and the classifiers at each node of the tree without parameters tuning.

4.2. Evaluation measurement

To compare the methods, the classification accuracy and test speedup are used. These measures are widely used in the label tree-based classification [10, 19]. In addition, the run-time is recorded for evaluation.

- *Classification Accuracy* (*Acc*) is measured as the number of correct predictions among the total number of predictions on the testing set. It is calculated as follows:

$$Acc = \frac{1}{M} \sum_{i=1}^{M} \mathbf{1}_{y_i}(\hat{y}_i), \tag{4}$$

  where M is the total number of testing images and $\mathbf{1}_{y_i}(\hat{y}_i)$ is an indicator function. The value of $\mathbf{1}_{y_i}(\hat{y}_i) = 1$ if the predicted class $\hat{y}_i$ and the given assigned class y_i are the same; otherwise, the value of $\mathbf{1}_{y_i}(\hat{y}_i) = 0$.

- *Test speedup* (S_{te}) is used to measure a speedup capacity of the label tree-based classification comparing to OvA classifiers-based classification. That is the one-vs-all test cost divided by the label tree test cost [10]. Since we only used the linear classifiers in experiments, this measure may be computed as follows:

$$S_{te} = \frac{N * M}{P}, \tag{5}$$

  where N is the number of class labels, M is the total number of testing images, and P is the total number of dot products that are performed for classifying M testing images. For example, the S_{te} of the tree $T_{16,2}$ is 8.0, meaning that only 32 linear classifiers are used for classifying a testing image instead of 256 OvA classifiers. The higher the value of S_{te}, the more computationally efficient the method.

- *Run-Time* is used for evaluating the length of time required to classify the testing set. For a fair comparison, all experiments were carried out on the same hardware configuration with the same dataset.

To ensure the stability of the experimental results, each experiment is repeated at least five times for each dataset. The reported performance measures are the average values and their standard deviations.

4.3. Experimental results

In this section, we report the experimental results on benchmark datasets with different tree configurations. For each configuration $T_{Q,H}$, the maximum depth H and the desired number of children per node Q are adjusted such that Q^H approximate the number of class labels. In addition, our method is reported with the popular kernels, including χ^2 (Ours - $kchi2$), $Intersection$ (Ours - $kinters$) and $Jensen - Shannon$ (Ours - kjs) [32].

From the experimental results, some essential conclusions can be drawn as follows. First, the classification accuracy (Acc) and the test speedup (S_{te}) depend on the tree configuration. In particular, when the value of H increases, the value of Q decreases; this leads to decreasing the number of classifiers evaluated at a node. And the test speed up is significantly improved but the accuracy is also dropped. Second, our method outperformed the other tree-based approaches. At the same accuracy level, our approach was more efficient. Moreover, at the same speedup level, our method achieved higher accuracy in most cases. Lastly, our method is faster than others in terms of the run-time in most configurations.

4.3.1. Results on ImageNet-1K

Table 1 lists the experimental results using the SIFT+LLC+SPM feature on ImageNet-1K. The results show that our method significantly outperformed in terms of computational efficiency and classification accuracy. For example, considering the tree $T_{32,2}$, our method achieved the best accuracy $Acc = 14.52 \pm 0.01\%$ and $S_{te} = 15.74 \pm 0.02$ (average $32 * 2$ classifiers evaluated) with the χ^2 kernel. Meanwhile, the accuracy of Bengio et al.'s method [3] was $Acc = 6.51 \pm 0.10\%$ and $Ste = 15.93 \pm 0.03$. Although its value of Ste is slightly greater than, our method is two times more accurate than. In addition, the accuracies of

Table 1. Comparison of the average and standard deviation of our method to the other methods on ImageNet-1K using the SIFT+LLC+SPM feature

Methods	$T_{32,2}$		$T_{10,3}$		$T_{6,4}$		$T_{4,5}$	
	Acc	S_{te}	*Acc*	S_{te}	*Acc*	S_{te}	*Acc*	S_{te}
Bengio et al. [3]	6.51	15.93	4.73	31.49	4.06	38.15	3.73	43.89
	±0.10	±0.03	±0.08	±0.54	±0.12	±0.46	±0.04	±0.41
Deng et al. [10]	11.9	10.3	8.92	18.20	5.62	31.3	NA	NA
Liu et al. [20]	12.51	15.81	10.61	30.91	9.73	38.10	9.08	42.03
	±0.07	±0.05	±0.10	±0.05	±0.09	±0.50	±0.05	±0.40
Ours - *kchi2*	**14.52**	15.74	11.44	33.33	10.28	42.92	**9.79**	50.11
	±0.01	±0.02	±0.20	±0.00	±0.02	±0.01	±0.15	±0.02
Ours - *kinters*	14.47	15.77	11.37	33.33	10.10	42.78	9.69	50.12
	±0.22	±0.02	±0.23	±0.00	±0.29	±0.16	±0.06	±0.02
Ours - *kjs*	14.48	15.75	**11.55**	33.33	**10.35**	43.05	9.52	50.11
	±0.08	±0.01	±0.08	±0.00	±0.02	±0.06	±0.21	±0.01

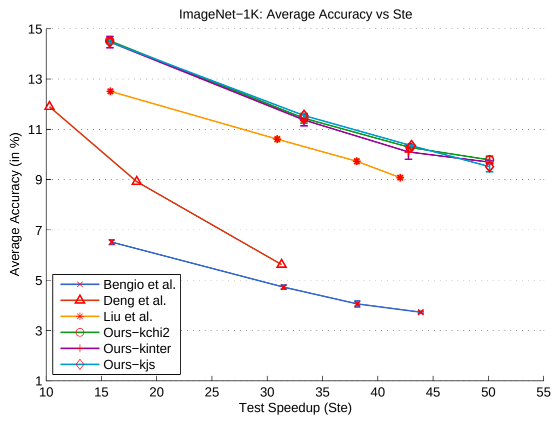
the methods proposed by Deng et al. [10], and Liu et al [20] are 11.9% and $12.51 \pm 0.07\%$, respectively.

The relationship between the accuracy and the test speedup are illustrated in Figure 1(a). As we can see, the accuracy drops when the test speedup increases. The reason is that the average number of classifiers evaluated for classifying a test image decreases. The result is that there is less information to make a correct classification. However, in all of the cases, the accuracy of our method is significantly higher than the accuracies of others at the same test speedup.

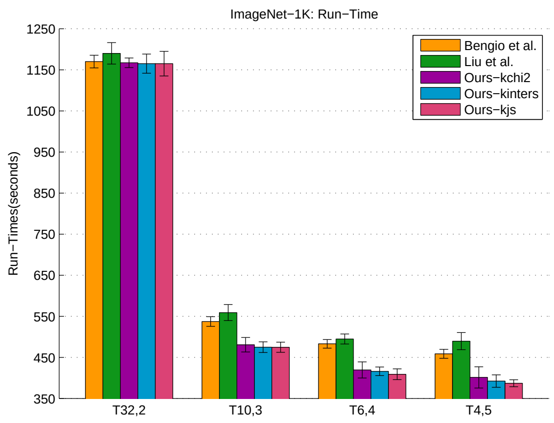
We recorded and reported for each method the length of time they require to classify all images of the testing set in Figure 1(b). The result has shown that our method is faster than the others with the same tree configuration. This is because the lengths of all paths from the root to a leaf node on our balanced tree are mostly equal. They may not exceed $log_Q(N)$, given Q is the number of children per node and N is the number of classes. Unbalanced trees, e.g. those constructed by the methods of Bengio et al. [3] and Liu et al. [20], may have longer paths. As a result, they require more time for classification.

Figure 2 illustrates the results of the methods using the CNN feature. Similar to the results using the SIFT+LLC+SPM feature, the experimental results demonstrate that our method achieved a higher *Acc* than others at the same S_{te} in most cases. Furthermore, the run-times were consistent with those described above.

An interesting result found from the experimental results is that the run-time depend on the number of dimensions of the feature vector and the depth of the tree. Specifically, when the depth of the tree increases, it takes a longer time for making a decision which child node to follow.

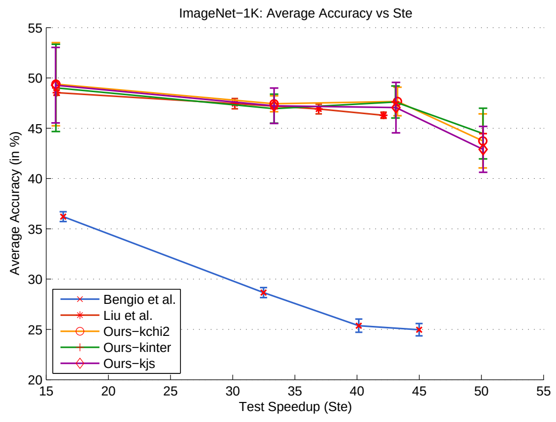


(a) The average accuracy and the test speedup of methods

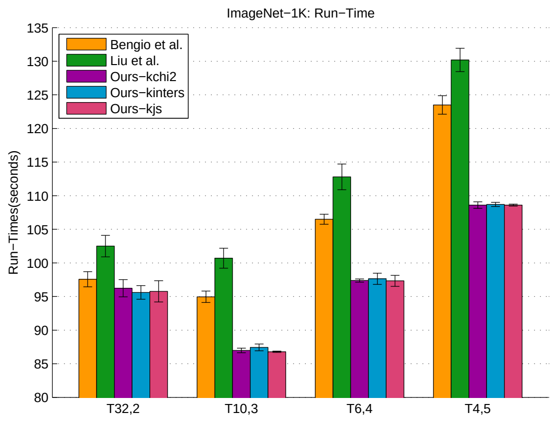


(b) The run-time of methods

Figure 1. Performance of the evaluated methods using the SIFT+LLC+SPM feature on ImageNet-1K



(a) The average accuracy and the test speedup



(b) The run-time of methods

Figure 2. Performance of the evaluated methods using the CNN feature on ImageNet-1K

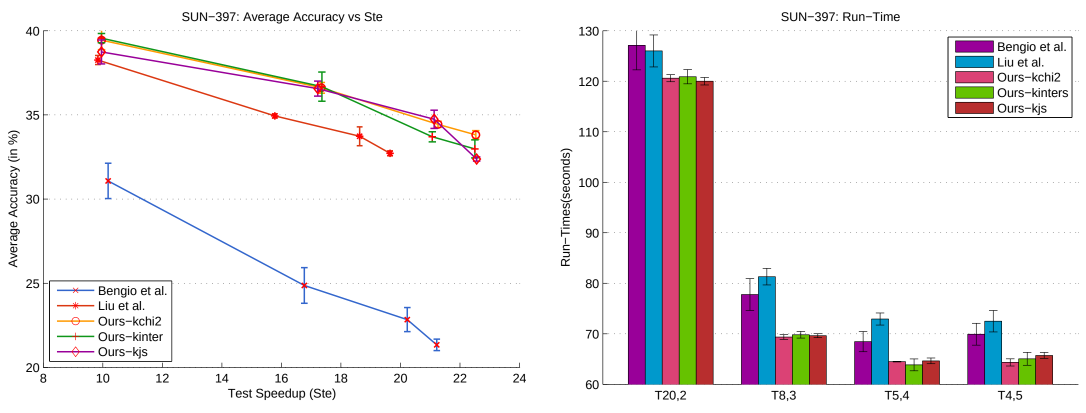
Table 2. Comparison of the average and standard deviation of our method to the other methods on SUN-397 using SIFT+LLC+SPM feature

Methods	$T_{20,2}$		$T_{8,3}$		$T_{5,4}$		$T_{4,5}$	
	Acc	S_{te}	Acc	S_{te}	Acc	S_{te}	Acc	S_{te}
Bengio et al. [3]	31.08	10.18	24.87	16.77	22.84	20.23	21.34	21.22
	±1.05	±0.25	±1.06	±0.37	±0.71	±0.33	±0.34	±0.48
Liu et al. [20]	38.26	9.84	34.94	15.78	33.73	18.63	32.72	19.65
	±0.27	±0.11	±0.12	±0.23	±0.56	±0.11	±0.15	±0.36
Ours - $kchi2$	39.44	9.96	36.61	17.34	34.44	21.26	**33.81**	22.53
	±0.56	±0.01	±0.32	±0.02	±0.18	±0.08	±0.26	±0.16
Ours - $kinters$	**39.56**	9.95	**36.68**	17.36	33.70	21.07	32.98	22.50
	±0.27	±0.00	±0.87	±0.07	±0.30	±0.07	±0.54	±0.16
Ours - kjs	38.74	9.95	36.56	17.22	**34.74**	21.13	32.38	22.56
	±0.71	±0.01	±0.45	±0.08	±0.54	±0.22	±0.13	±0.03

4.3.2. Results on SUN-397

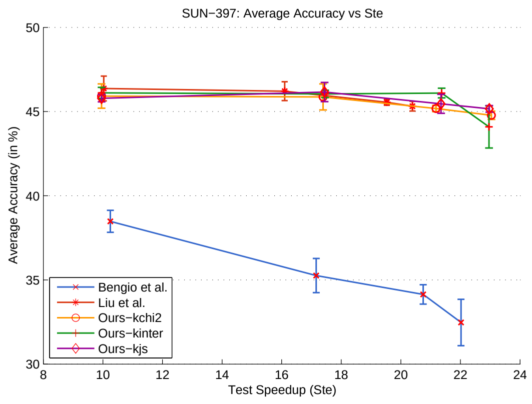
Table 2 and Figure 3 show the experimental results using the SIFT+LLC+SPM feature on SUN-397. The results are consistent with those on ImageNet-1K. At the same level of speedup, we observe that our trees have much better performance than other methods. For example, considering the tree $T_{20,2}$, the best classification accuracy of our method is $Acc = 39.56 \pm 0.27\%$ with $Intersection$ kernel meanwhile of the method proposed by Bengio et al. [3] and Liu et al. [20] are $31.08 \pm 1.05\%$ and $38.26 \pm 0.27\%$, respectively.

The results are shown in Figure 4 using the CNN feature. The trees generated by our method achieves comparable or significant accuracy while achieving better speedup.

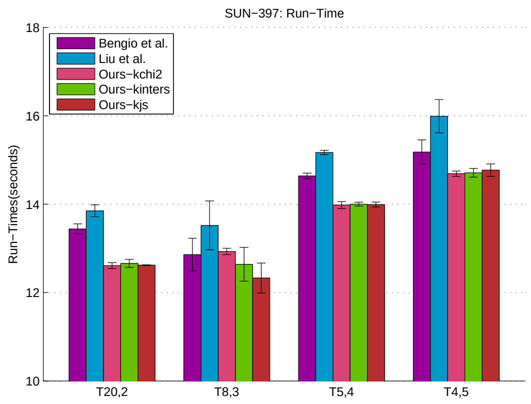


(a) The average accuracy and the test speedup (b) The run-time of methods

Figure 3. Performance of the evaluated methods using the SIFT+LLC+SPM feature on SUN-397



(a) The average accuracy and the test speedup



(b) The run-time of methods

Figure 4. Performance of the evaluated methods using the CNN feature on SUN-397

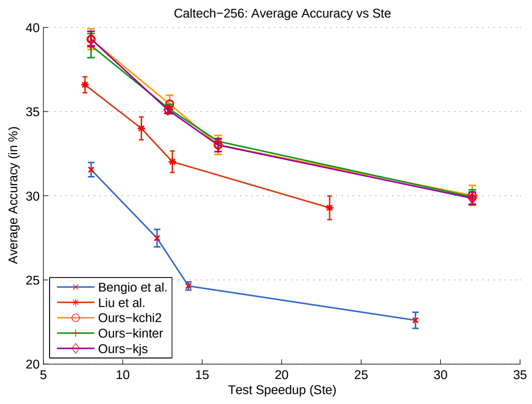
4.3.3. Results on Caltech-256

The experimental results using the SIFT+LLC+SPM feature on Caltech-256 dataset are reported in Table 3 and Figure 5. As shown in this table, with the trees built by our method, the classification accuracy and the test speed for different kernels is significantly higher. For example, we consider the tree $T_{16,2}$, the best classification accuracy of our method is $39.31 \pm 0.46\%$ with $Jensen - Shannon$ kernel meanwhile of the Bengio et al.'s method [3] and Liu et al.'s method [20] are $31.55 \pm 0.43\%$ and $36.59 \pm 0.48\%$, respectively.

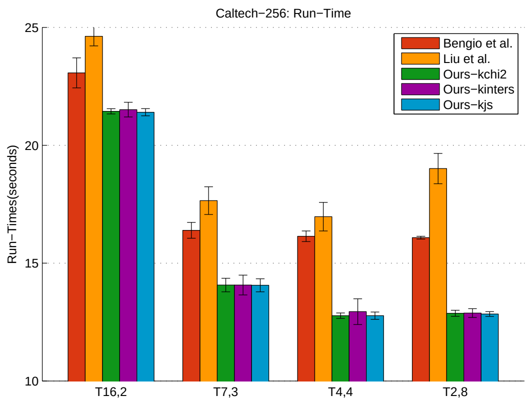
Figure 6 shows the results of methods using the CNN feature. As can be seen, at the same level of speedup, the performance of our method is better than of the other methods. An interesting result with the tree $T_{16,2}$, the accuracy of using 256 binary classifiers is 79%, meanwhile, we achieved 73% but our method is 8 times faster than OvA method.

Table 3. Comparison of the average and standard deviation of our method to the other methods on Caltech-256 using SIFT+LLC+SPM feature

Methods	$T_{16,2}$		$T_{7,3}$		$T_{4,4}$		$T_{2,8}$	
	Acc	S_{te}	Acc	S_{te}	Acc	S_{te}	Acc	S_{te}
Bengio et al. [3]	31.55 ±0.43	8.01 ±0.13	27.48 ±0.52	12.16 ±0.19	24.64 ±0.24	14.13 ±0.27	22.60 ±0.48	28.42 ±0.29
Liu et al. [20]	36.59 ±0.48	7.62 ±0.10	34.00 ±0.68	11.17 ±0.37	32.02 ±0.64	13.12 ±0.59	29.28 ±0.70	23.02 ±0.93
Ours - $kchi2$	39.30 ±0.61	8.00 ±0.00	**35.45** ±0.52	12.95 ±0.09	33.02 ±0.57	16.00 ±0.00	**30.02** ±0.60	32.00 ±0.00
Ours - $kinters$	38.92 ±0.72	8.00 ±0.00	35.15 ±0.28	12.93 ±0.06	**33.23** ±0.12	16.00 ±0.00	29.93 ±0.43	32.00 ±0.00
Ours - kjs	**39.31** ±0.46	8.00 ±0.00	35.10 ±0.20	12.84 ±0.05	33.01 ±0.39	16.00 ±0.00	29.85 ±0.37	32.00 ±0.00

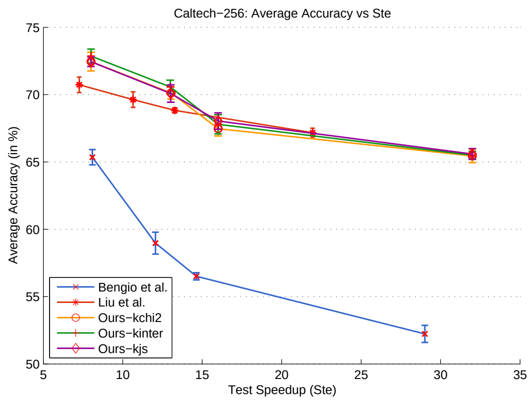


(a) The average accuracy and the test speedup

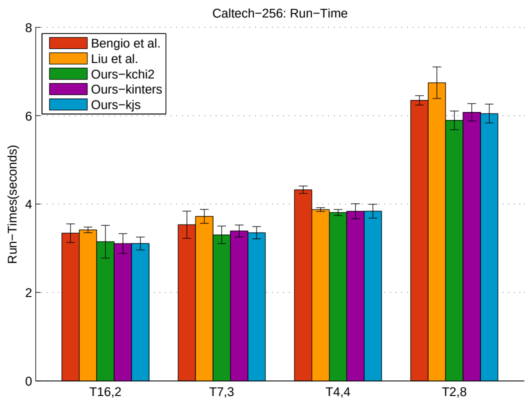


(b) The run-time of methods

Figure 5. Performance of methods using the SIFT+LLC+SPM feature on Caltech-256



(a) The average accuracy and the test speedup



(b) The run-time of methods

Figure 6. Performance of the evaluated methods using the CNN feature on Caltech-256

5. DISCUSSIONS

In this paper, we follow the methods that learn the tree structure by the spectral clustering algorithm with the similarity matrix among classes. Our method has some advantages as follows. First, using sum-match kernel helps to improve the classification accuracy since similar classes are precisely grouped into clusters. Based on a feature mapping technique, the sum match kernel function can be computed as the distance between two the mean feature vectors of two classes in the mapped space. Second, by building a balanced tree, the proposed method is more efficient in classification than other methods which employ unbalanced trees. Here, the balanced tree is constructed without solving a NP-hard optimization problem as in other methods (i.e., Deng et al. [10]). But, the disadvantage of our method is the features vectors in the original feature space are needed to be mapped into a new feature space.

6. CONCLUSION AND FUTURE WORK

In this paper, we proposed a method for learning an effective and balanced label tree for efficient multi-class classification. Based on the explicit feature map, we reformulated the sum-match kernel as a distance function between mean feature vectors of classes in the mapped-feature space. Thus, the cost of building label tree is significantly reduced. In addition, the balanced tree structure built by our proposed algorithm gains the computational efficiency in classification. The experimental results on the benchmark datasets indicated the advantage of our method on large-scale classification in terms of accuracy and computational efficiency.

For further research, we are going to exploit the relationship, such as semantic, correlation, exclude among classes in order to learn a label tree structure.

ACKNOWLEDGMENT

This research is funded by Vietnam National University Ho Chi Minh City VNU-HCM under grant number B2015-26-01.

REFERENCES

[1] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Good practice in large-scale learning for image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 507–520, Mar. 2014.

[2] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *Journal of Machine Learning Research*, vol. 1, pp. 113–141, 2000.

[3] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," in *Advances in Neural Information Processing Systems 23, NIPS 2010 . Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada.*, 2010, pp. 163–171.

[4] L. Bo and C. Sminchisescu, "Efficient match kernel between sets of features for visual recognition," in *Advances in Neural Information Processing Systems 22, NIPS 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada.*, 2009, pp. 135–143.

[5] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos, "Supervised learning of semantic classes for image annotation and retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 3, pp. 394–410, 2007.

[6] K. Chatfield, R. Arandjelovic, O. M. Parkhi, and A. Zisserman, "On-the-fly learning for visual search of large-scale image and video datasets," *International Journal of Multimedia Information Retrieval*, vol. 4, no. 2, pp. 75–93, 2015.

[7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *British Machine Vision Conference, BMVC 2014, Nottingham, UK, September 1-5, 2014*, 2014.

[8] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Comput. Surv.*, vol. 40, no. 2, pp. 5:1–5:60, 2008.

[9] J. Deng, A. C. Berg, and F. Li, "Hierarchical semantic indexing for large scale image retrieval," in *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, 2011, pp. 785–792.

[10] J. Deng, S. Satheesh, A. C. Berg, and F. Li, "Fast and balanced: Efficient label tree learning for large scale object recognition," in *Advances in Neural Information Processing Systems 24, NIPS 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain.*, 2011, pp. 567–575.

[11] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[12] T. Gao and D. Koller, "Discriminative learning of relaxed hierarchy for large-scale visual recognition," in *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, 2011, pp. 2072–2079.

[13] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep. 7694, 2007. [Online]. Available: http://authors.library.caltech.edu/7694

[14] G. Griffin and P. Perona, "Learning and using taxonomies for fast visual categorization," in *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*, 2008, pp. 1–8.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25, NIPS 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, 2012, pp. 1106–1114.

[16] M. Lapin, B. Schiele, and M. Hein, "Scalable multitask representation learning for scene classification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, 2014, pp. 1434–1441.

[17] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, 2006, pp. 2169–2178.

[18] L. Li, R. Socher, and F. Li, "Towards total scene understanding: Classification, annotation and segmentation in an automatic framework," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, 2009, pp. 2036–2043.

[19] B. Liu, F. Sadeghi, M. F. Tappen, O. Shamir, and C. Liu, "Probabilistic label trees for efficient large scale image classification," in *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, 2013, pp. 843–850.

[20] S. Liu, H. Yi, L. Chia, and D. Rajan, "Adaptive hierarchical multi-class SVM classifier for texture-based image classification," in *Proceedings of the 2005 IEEE International Conference on Multimedia and Expo, ICME 2005, July 6-9, 2005, Amsterdam, The Netherlands*, 2005, pp. 1190–1193.

[21] T. Mai and K. Hoang, "Label tree based image classification using sum match kernel," in *The 2015 International Conference on Advanced Technologies for Communications (ATC), Ho Chi Minh, Vietnam, October 14-16, 2015*, 2015, pp. 468–472.

[22] M. Marszalek and C. Schmid, “Constructing category hierarchies for visual recognition,” in *Computer Vision - ECCV 2008, 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part IV*, 2008, pp. 479–491.

[23] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in *Advances in Neural Information Processing Systems 14, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, 2001, pp. 849–856.

[24] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “CNN features off-the-shelf: An astounding baseline for recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2014, Columbus, OH, USA, June 23-28, 2014*, 2014, pp. 512–519.

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

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, pp. 1–42, April 2015.

[27] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Cambridge, MA, USA: MIT Press, 2001.

[28] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, vol. abs/1409.1556, 2014.

[29] M. Sun, W. Huang, and S. Savarese, “Find the best path: An efficient and accurate classifier for image hierarchies,” in *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, 2013, pp. 265–272.

[30] A. Vedaldi and B. Fulkerson, “Vlfeat: an open and portable library of computer vision algorithms,” in *Proceedings of the 18th International Conference on Multimedia 2010, Firenze, Italy, October 25-29, 2010*, 2010, pp. 1469–1472.

[31] A. Vedaldi and K. Lenc, “Matconvnet: Convolutional neural networks for matlab,” in *Proceedings of the 23rd Annual ACM Conference on Multimedia Conference, MM ’15*, 2015, pp. 689–692.

[32] A. Vedaldi and A. Zisserman, “Efficient additive kernels via explicit feature maps,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 3, pp. 480–492, Mar. 2012.

[33] J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong, “Locality-constrained linear coding for image classification,” in *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, 2010, pp. 3360–3367.

[34] Y. Wang and D. A. Forsyth, “Large multi-class image categorization with ensembles of label trees,” in *Proceedings of the 2013 IEEE International Conference on Multimedia and Expo, ICME 2013, San Jose, CA, USA, July 15-19, 2013*, 2013, pp. 1–6.

[35] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, “SUN database: Large-scale scene recognition from abbey to zoo,” in *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*, 2010, pp. 3485–3492.

[36] D. Xu and S. Chang, “Video event recognition using kernel methods with multilevel temporal alignment,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1985–1997, 2008.

[37] X. Zhang, L. Liang, and H. Shum, “Spectral error correcting output codes for efficient multiclass recognition,” in *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*, 2009, pp. 1111–1118.

[38] B. Zhao and E. P. Xing, “Sparse output coding for large-scale visual recognition,” in *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*, 2013, pp. 3350–3357.

[39] S. Zhu, X. Wei, and C. Ngo, “Collaborative error reduction for hierarchical classification,” *Computer Vision and Image Understanding*, vol. 124, pp. 79–90, 2014.

Received on December 24 - 2015
Revised on September 21 - 2016