https://doi.org/10.1016/j.jvcir.2018.09.008

# Can a machine have two systems for recognition, like human beings?

*Jiwei Hu*, Kin-Man Lam†, Ping Lou*, Quan Liu* and Wupeng Deng**
[*]Wuhan University of Technology
[*]Key Laboratory of Fiber Optic Sensing Technology and Information Processing (Wuhan University of Technology), Ministry of Education
E-mail: hujiwei@whut.edu.cn
[†]Department of Electronic and Information, Engineering, The Hong Kong Polytechnic University
E-mail: enkmlam@polyu.edu.hk

## Abstract

Artificial Intelligence has attracted much of researchers’ attention in recent years. A question we always ask is: “Can machines replace human beings to some extent?” This paper aims to explore the knowledge learning for an image-annotation framework, which is an easy task for humans but a tough task for machines. This paper’s research is based on an assumption that machines have two systems of thinking, each of which handles the labels of images at different abstract levels. Based on this, a new hierarchical model for image annotation is introduced. We explore not only the relationships between the labels and the features used, but also the relationships between labels. More specifically, we divide labels into several hierarchies for efficient and accurate labeling, which are constructed using our Associative Memory Sharing method, proposed in this paper.

***Keywords***– image annotation, multi-labeling, hierarchical tree structure, feature-pool selection

## 1. Introduction

Many computer-vision applications, such as scene analysis and image segmentation, are ill-suited for traditional classification, in which each image can only be associated with a single class or label. However, in the real world, an image is usually associated with multiple labels, which are characterized by different regions of the image. Thus, image classification is naturally considered either as a multi-label learning or a multi-instance learning problem. Most of the recent work in multi-label classification task, such as scene recognition and multi-object recognition [1,2,3], has focused on the method of tagging a given image with multiple class labels. A serious problem with most of these existing approaches is that they do not exploit the correlations between the class labels.

For multi-label learning, a straightforward method of achieving the goal of correctly classifying the multiple labels of an image is to consider images with the same multiple labels as a new class, and to build a model for this new, multi-label class. However, the problem with this approach is that the samples belonging to the multi-label classes are usually too sparse to build usable models. To solve this problem, the multi-label samples are used more than once during training. Each sample is considered a positive example of each of the label classes it belongs to. This training method is called 'cross-training' [4]. Another approach [5] to multi-label learning is to perform image segmentation first. As an image is divided into a number of non-overlapping regions, and each region may be described by one label, this can roughly determine the maximum numbers of classes it can fit. Image segmentation is the process of dividing an image into different regions such that each region is nearly homogeneous, whereas the union of any two regions is not. It serves as a key task in image analysis and pattern recognition, and is a fundamental step toward low-level vision, which is significant for object recognition, image retrieval and other computer-vision-related applications [6,7,8]. However, segmentation itself is a difficult, imperfect task. Segmentation always results in the problem of complexity, and unsuccessful segmentation also degrades the performance of the image-annotation task. Nevertheless, a lot of research is still being devoted to achieving a good segmentation performance.

Human beings see image-annotation tasks as an easy problem. The related tags that we assign to an image can be classified into two categories. As shown in Fig. 1, one category includes those basic or obvious tags that we do not need to think about, e.g. apple, sky, dog, etc. The other includes the more complex or abstract tags that we need to think over, e.g. market, African, indoor, etc. The book titled "Thinking, Fast and Slow" [36] surmises that humans have two systems; one is used to solve the problems without requiring thinking, while the other requires some thought. Can a machine have two such systems, like human beings, for the image-annotation tasks? Motivated by this book, we wondered if a machine could have two such systems, like human beings? Therefore, in this paper, we propose a hierarchical framework to mimic the two systems for handling tags, i.e. with solid concepts and abstract concepts, respectively.

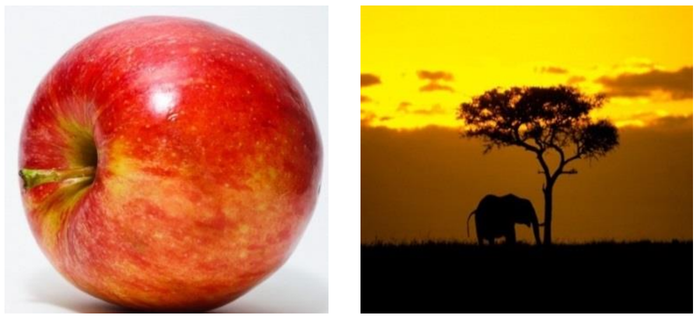


**Fig. 1.** (a) An image with a simple, solid tag, and (b) an image with a confusing, abstract tag.

In order to exploit the correlations between the class labels, we introduce a method called Associative Memory Sharing (AMS), which classifies image labels into different levels of a hierarchy according to their level of abstraction, for the purpose of constructing a tree structure in the learning framework. In other words, the labels or graphs of labels are linked to each other through the tree structure. In pursuit of the ultimate goal of building an intelligent image-annotation system, it is also necessary to incorporate human knowledge into our proposed framework. In the training part, we will use human knowledge interactively to help the system to choose representative images for each label class.

The remainder of this paper is organized as follows. In Section 2, a brief introduction to related works will be given. We present our proposed method in detail in Section 3. The experiment set-up and results, and a conclusion, are given in Sections 4 and 5, respectively.

## 2. Related work

In this section, we will give a brief overview of the different models for solving multi-label learning problems. We will also discuss feature extraction and image representation, which play an important role in image-annotation frameworks.

### 2.1 Literature Review

In recent years, various learning methods have been proposed for automatic image annotation. These methods have in common that they all rely on a set of labeled pictures to learn models, which can then predict the labels for unlabeled data. The literature can be grouped based on three models: generative models, discriminative models, and nearest-neighbor (NN)-based models. Most generative models [9, 10] construct a joint distribution over image contents and the associated keywords while finding a mapping between the image features and the annotation keywords. These generative models aim to learn a single model for all the vocabulary terms, which yields a better modeling in terms of dependencies. Some methods treat the task of image annotation as several binary classification problems. This means that the joint distribution of the unobserved variables and the observed variables is not needed. In this situation, discriminative models [11,12,13] can generally yield a superior performance. Discriminative models learn a separate classifier for each single label, and use the classifier to judge whether the test image belongs to a particular label or not. Although the training process is complicated and time-consuming, this approach can, with a smart design, achieve more promising performances than the generative models. The third model – one of the oldest, simplest, and most effective methods for pattern classification – is the *k*NN-based model [14], which is accurate, especially with an increasing amount of training data. Recently, a NN-based keyword-transfer approach was proposed in [15]. In this method, the labels are transferred from neighbors to a given image after a simple distance calculation. The nearest neighbors are determined using Joint Equal Contribution (JEC) only, which finds the average distance obtained from the differences in image features. The method was extended in [16] to filter out most of the irrelevant labels, with a promising result obtained.

Although the learning stage plays an important role in an image-classification system, the features employed also affect the performance of the whole framework. In [17], a graph structure was proposed to describe the relationship between the features. In this approach, a pair-wise graph is constructed, with each vertex representing a single image that may be labeled or unlabeled. Two similar images are connected by an edge, and the edge weight is calculated as an image-to-image distance. In [18], a new graph-based model was proposed for recognition based on a semi-supervised framework, which can predict both the predefined labels and undefined labels. The concept of a simple graph was extended in [19] to a hypergraph, whose main argument is that the simple graph cannot completely represent the relations between images. Actually, a hypergraph can contribute to a better representation of the relations between images by considering not only the local grouping information, but also the similarities between the hyperedges that involve more than two images. The idea of a hypergraph was used in [20] to determine a suitable feature space for each class. It is a simple and efficient method for finding a good representative image patch for each label class, which can greatly enhance efficiency in the learning stage.

With the ongoing development of consumer electronics equipment, image databases are becoming larger and larger, with a growing number of labels. In [21], millions of photos have been captured as informative reports, and utilized for computer-vision tasks, such as situation recognition. In their work, a visual analytics system was built to understand the information that could be collected from their photo report streams. To learn about thousands of objects from millions of images, a model with a large learning capacity and considerable efficiency is needed. Deep Convolution Neural Networks (CNNs) [22, 23, 24], which have achieved great success on single-label image classification in recent years, constitute this model. Because of their strong capability for learning representative features, CNN models yield breakthrough performance on many other computer-vision tasks, which have attracted attention in the research of image understanding recently. Although many techniques that have been proposed in the last decade can give a reasonable performance, a large number of potential labels causes problems, in terms of decreasing their accuracy and efficiency. More and more researchers are now exploring the relationship between labels; many contributions, which represent a landmark in the research of image annotation. have been made regarding this.

## 2.2 A Structured Framework for Image Understanding

Recent progress on image annotation mainly focused on exploring semantic relations between different labels. Such relations can be modeled by graphical models [25, 26] or recurrent neural networks (RNNS) [27]. Despite the great improvements achieved by exploiting semantic relations, existing approaches cannot capture the spatial relations of labels, for the reason that their spatial locations are not annotated for training. To address the problem of a large number of labels required for an image multi-labeling framework, contextual modeling has become a recent focus. For example, in object-class recognition, the presence of one class may suppress (or promote) the presence of another class that is negatively (or positively) correlated, e.g. [28,29]. In [24], the object-detection task is achieved by modeling the object co-occurrences and spatial relationships using a graphical tree. Enforcing tree-structured dependencies among objects allows for the learning of the model in more than a hundred object categories; this can also be applied efficiently to image annotation. Even though the method does not explicitly impose a hierarchical structure in the learning procedure, the tree organizes objects into a natural hierarchy. In [28], structured prediction models, which explicitly take the dependencies among image labels into account, were proposed for image labeling. In the tree-structured models, the nodes represent image labels, and the edges between the nodes encode the dependency relations. To allow for more complex dependencies, labels are combined in a single node, and mixtures of trees are used.

This paper aims to devise a learning algorithm to handle labels in a human way. Although it is difficult for machines to handle labels in different ways according to each label's nature, we can construct a hierarchical structure among the labels to facilitate this capability. Our proposed method will incorporate the correlations between labels into a learning framework. We will explore the relationship between labels that appear in the same images, and then build the hierarchy of those labels in the whole dictionary. Specifically, a word or label that frequently appears simultaneously with several other words will be located at a higher level in the hierarchy. For example, the label "Market", which often appears with labels such as "People", "Fruit", "Outdoor", and so on, will be higher in the hierarchy. The hierarchy will also be represented using a tree structure. Our proposed tree structure not only considers the hierarchy between the parent nodes to child nodes, but also introduces different levels of hierarchy, which is significantly different from the trees in [28, 29, 30]. In Section 3, we will introduce the details of how to construct the tree hierarchy using our Associative Memory Sharing algorithm.

For image representation, we follow our previous work [31], which uses pools of features. We will learn the mapping between the tag space and the image-representation space. We aim to boost the performance of feature-label selection based on the constructed hierarchical structure. Our proposed approach can achieve a good balance between efficiency and accuracy. Moreover, deep learning features are also employed in our framework on the NUS-Wide dataset.

The concept of Associative Memory Sharing (AMS) proposed in this work is an extension of our previous work [32], where we determined the labels of an image by making use of statistical data and prior knowledge of the dataset, and also by considering the relationships between the labels. In order to accommodate more dependencies between labels in a model, [30] considered the extension of grouping label variables, and then defined a tree covering these groups. A label group can be viewed as a fully connected set of variables in the graphical model. However, a certain structure in this cyclic, graphical model contains local cycles only (i.e. among the neighboring groups in the tree). [30] also extended the tree to a mixture-of-trees, which allows for more label dependencies. The mixtures are defined either to cover trees with different group sizes, or to cover trees with different structures over a fixed set of nodes. A tree model was used to learn the dependencies among object categories, rather than class labels. Before the work in [30], [29] have used a fully-connected conditional random field to model object dependencies, which is a time-consuming process when handling a large number of object categories. [33] distinguishes a scene from objects and from model dependencies among objects using scene-object relationships. In that way, the direct dependencies among objects may be ignored. [30] described a prior model that captures the co-occurrence statistics and spatial relationships between objects, which may provide a richer representation of object dependencies. However, one class or one object represented by a single node may still lead to a complex connection between labels and a high computational cost. In this paper, we employ probability functions in our tree structure. Instead of representing one label using one node, we introduce a totally different form of representation. Our method allows for either assembled labels or a simple label only. We will present the details of our algorithm in next section.

## 3. Associative Memory Sharing

Associative memory in computer organization is the memory unit accessed by the content of data. It is also the memory unit, where the storage locations are identified by their contents. In our work, labels are considered as different contents of data located in a tree. We want to make use of the transfer knowledge from one node to another node. In other words, associated memory can be shared among the nodes.

In this paper, the labels of the same images are assumed to be associated, and have their associations modeled by our AMS algorithm.

### *3.1 Construction of a Tree Hierarchy*

Suppose that the set $\Gamma = \{l_1, l_2, \ldots, l_n\}$ represents the dictionary of the labels for the whole image dataset. The training images are denoted as $\{i_1, i_2, \ldots, i_N\}$, while $l_{Si}$ ($Si \in \{l_1, l_2, \ldots, l_n\}$) represents the labels of the image $I$ in the training set. Then, assuming that each image contains no more than 4 labels, a tree structure can be constructed using the following steps:

Step 1. Only images containing label pairs (i.e. two labels) are considered in this step. We count the frequency of each of these label pairs (e.g. Fruit-Apple, assigned to *Layer* 1 in Fig. 2), which appears in the same training images. Sort these label pairs in descending order according to their frequencies.

Step 2. Only images containing label triplets (i.e. three labels) are considered in this step. We count the frequency of each of these label triplets (e.g. Person-Bottle-Dining table, assigned to *Layer* 3 in Fig. 2), which appears in the same training images. Sort these label triplets in descending order according to their frequencies.

Step 3. We count the frequency of each single label, which appears with a label pair (e.g. Garden-(Fruit, Apple), assigned to *Layer* 2 in Fig. 2, where the label "Garden" appears with the label pair "Fruit, Apple"). Sort these labels in descending order according to their frequencies.

Step 4. We count the frequency of each single label which appears with a label triplet (e.g. Dinner-(Person-Bottle-Dining table), assigned to *Layer* 4 in Fig. 2). Sort these labels in descending order according to their frequencies.

Step 5. We can now construct a 6-layer tree-hierarchical structure. The bottom layer, namely "*Layer* 0", consists of all the individual class labels (*n* classes), and each node in this layer contains a class label. Then, *Layer* 1 to *Layer* 4 are constructed according to Steps 1 to 4, such that each node in each of the layers contains different combinations of the labels. Finally, the top layer, i.e. *Layer* 5, contains all the labels in a single cluster.

The above steps illustrate the construction of a hierarchy for a number of labels. Specifically, the single labels, label pairs, and label triplets will form individual small clusters. Each of these clusters is considered a node in the tree structure. The constructed tree structure represents the relationships between the labels and the corresponding image features. The tree structure can be extended to include images having even more labels. However, the number of these types of images is usually small.

Based on our previous work [20, 31], we can learn image exemplars for each of the label classes (i.e. single label, label pairs, single + label pairs, label triplets, single + label triplets, etc.). These image exemplars can be obtained by incorporating image patches into a hypergraph. The exemplars are good representations of each class label, which are used to represent the nodes in the tree structure in our proposed framework. Then, we can extract the corresponding image features relating to each node in the constructed hierarchical graph.

## *3.2 Tree Learning and Classification*

In this paper, we propose a novel algorithm which incorporates different feature pools for a hierarchical training of node classifiers. Unlike our previous work [31], classifiers are not trained for each label class. We learn a classifier for each node in the tree structure instead of learning a classifier for each single label. As was done in [34], we train a regression model with the use of the tree structure. But unlike [34], we extend the nodes which combine labels in different ways in our proposed framework.

Assume that the correlations among the labels can be represented using a tree structure consisting of a set of vertices or nodes $V$. In the tree structure, each bottom-leaf node corresponds to a single label (i.e. $a$, $b$, $c$, $d$, $e$, $X$, $Y$, and $Z$ in Fig. 2) in the dictionary, while the middle nodes represent the label pairs (e.g. $ab$), label triplets (e.g. $bcd$), and two types of extended nodes, which are formed by combining a single label with either a label pair (e.g. $X$-($ab$)) or a label triplet (e.g. $Y$-($bcd$)), as illustrated in Fig. 2.

To train a node classifier, a feature pool will first be selected from a set of feature pools, and then a specific classifier is trained using the feature pool for the node. In the first step of our algorithm, the transferable knowledge between nodes and the common features among the different nodes are not considered. Only the most suitable feature pool for each node is identified. It should be noted that the same feature pool may be selected and used for a number of nodes. In the second step of our training, we incorporate the multi-task learning algorithm in [35] in our framework to train the node classifiers. Fig. 3 shows the details of our tree-structure-based feature-label selection algorithm, which consists of a training stage and a testing stage. In testing a query input, we aim to find the likelihood of a test image belonging to each node. Then, the final score for each label class is computed according to hierarchical factors (i.e. the weights learned for the nodes). Further details will be given in Sections 3.3 and 3.4. For convenience, the mathematical symbols used in our algorithm are tabulated in Table 1.

*Training Step*:

Step 1: Feature Selection (AdaBoost) – A classifier is trained for each node used to find the feature pool that results in the best performance for the class labels, with the help of the hierarchical factors.

Step 2: Use the multi-task learning algorithm in [35] to learn a classifier for each node based on the selected feature pool.

Step 3: Train a biased classifier, based on each of the feature pools, for labeling novel images.

*Testing Step*:

Step 1: Use the biased classifier trained for each feature pool to judge which of the feature pools is the most suitable for classifying a test image.

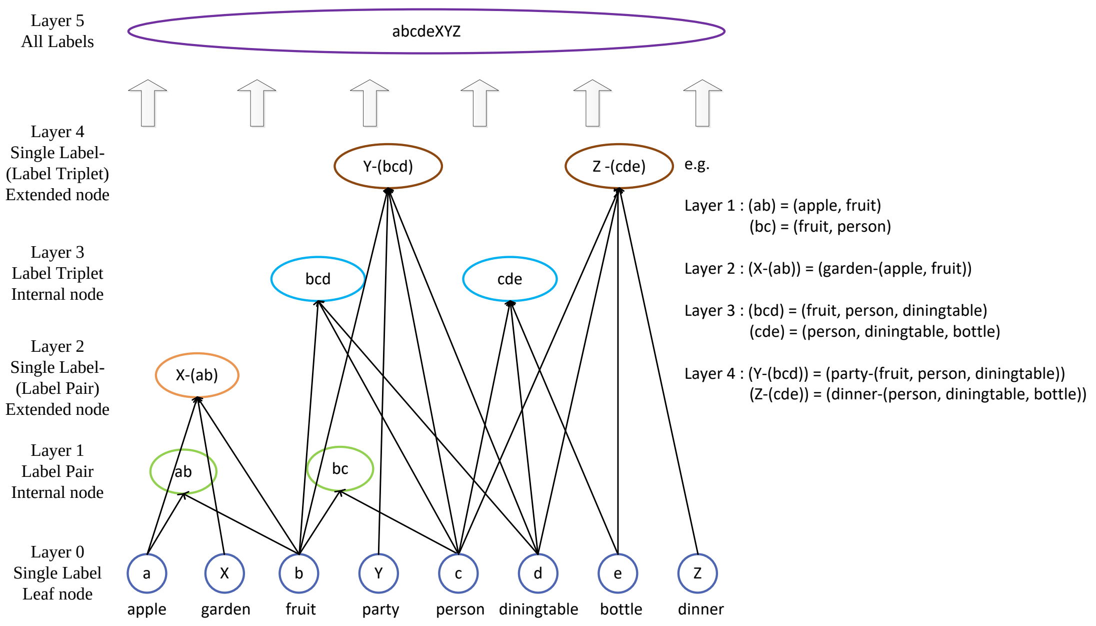
Step 2: Use the node classifiers to output scores (in the form of probabilities), which represent how likely it is that the test image belongs to the corresponding node.

Step 3: Consider the hierarchical factor for each node in computing the final score for each label class, and choose the labels with the highest scores.

**Fig. 3.** The algorithm for training the nodes and for label selection.

**Table 1** Meaning of the mathematical symbols used in our algorithm.

| *Math Symbol* | *meaning* | *Math Symbol* | *meaning* |
|---|---|---|---|
| $I$ | image | $p$ | probability |
| $l$ | label | $K$ | Kernel function |
| $S$ | label set | $N$ | number of training images |
| $t$ | tag | $w, s, g$ | weighting factors |
| $X$ | feature | $W_c$ | common regularization |
| $f$ | feature pool | $V_k$ | specific regularization |
| $b$ | offset | $\alpha$ | optimization factor |
| $c$ | penalty term | $\beta$ | regularization parameter |
| $d$ | dimension | $\lambda$ | tuning parameter |
| $n$ | number of labels | $\rho$ | memory sharing factor |
| $v$ | nodes | $\xi$ | error rate |
| $G_v$ | groups of nodes | $\Phi$ | nodes in layers |



Layer 5 All Labels
abcdeXYZ
Layer 4 Single Label-(Label Triplet) Extended node
Y-(bcd)
Z -(cde)
Layer 3 Label Triplet Internal node
bcd
cde
Layer 2 Single Label-(Label Pair) Extended node
X-(ab)
Layer 1 Label Pair Internal node
ab
bc
Layer 0 Single Label Leaf node
a
X
b
Y
c
d
e
Z
apple
garden
fruit
party
person
diningtable
bottle
dinner
e.g.
Layer 1 : (ab) = (apple, fruit)
(bc) = (fruit, person)
Layer 2 : (X-(ab)) = (garden-(apple, fruit))
Layer 3 : (bcd) = (fruit, person, diningtable)
(cde) = (person, diningtable, bottle)
Layer 4 : (Y-(bcd)) = (party-(fruit, person, diningtable))
(Z-(cde)) = (dinner-(person, diningtable, bottle))

**Fig. 2.** The tree structure constructed using our proposed model, where *Layer* 0 contains eight bottom-leaf nodes corresponding to eight labels (*a*, *b*, *c*, *d*, *e*, *X*, *Y*, and *Z* in the dictionary; *Layer* 1 and *Layer* 3 contain label pairs and label triplets, respectively; *Layer* 2 and *Layer* 4 are the extended nodes, which are those nodes composed of a single label with either a label pair or triplet, respectively; and *Layer* 5 is a single node, which contains all the labels. A leaf node represents a single label, which may be a simple label (e.g. *a*, *b*, *c*, *d*, *e*) or an abstract label (e.g. *X*, *Y*, *Z*). An internal node represents a combination of simple labels, while an extended node represents a combination of abstract and simple labels.

*3.3 Problem Statement and Formulation*

In the image-annotation problem, upon receiving a query or test image $I_q$, the annotation algorithm will output its corresponding labels $l(I_q, t)$, where $l(I_q, t)$ refers to the set of tags $t$ related to the query image $I_q$. Assume that the label set $S$ contains $n$ classes, so $l(I_q, t)$ is a subset of $S$. The aim of annotation is to find the tags $t^*$ that maximize the conditional distribution $p(t|I_q)$. The feature pool used in learning is denoted as $F = \{f_1, ..., f_j, ..., f_M\}$, where $f_j$ represents one type of features (e.g. color histogram, local shape descriptors, etc.). Denote $d_j$ as the dimension of the feature $f_j$. Then, the total dimensionality of all the image features in the feature pool $F$ is $d$, where $d = \sum_{j=1}^{M} d_j$ . We can form a feature matrix $X_j \in \Re^{n' \times d_j}$ to represent the features of $n'$ training images using the jth feature pool. Then, we can learn a corresponding regression coefficient vector $\beta_{kj} \in \Re^{d_j}$ for the jth feature of the kth node. Since we have to output the final scores for each label class in the last step, we have $\beta_k = (\beta_{k1}^T, \dots, \beta_{kM}^T)^T$ . The groups of regression coefficients associated with each node of the tree can be obtained by a method called tree-guided group lasso, which applies group lasso to groups of output variables, defined in terms of a hierarchical clustering tree. The detail is illustrated by Equation (7) to Equation (10) below.

Following our previous work, the weak classifier for the kth node, $T_k$, can be defined as follows:

$$\Phi_{T_k}(X) = W_k^T X + b, \qquad (1)$$

where $W_k = W_c + V_k$, $||W_c||$ is a common regularization term shared by those classifiers using the same feature pool, and $||V_k||$ is the specific regularization term for the individual node class. $X$ is the feature vector of the training samples (which will be described in Section 4), and $b$ is an offset. For the second and third steps of our training algorithm, we aim to learn a classifier for each node using the same feature pool, and then we train a biased classifier for each feature pool. Following are the specific details of our training algorithm.

The training samples which result in the best performance with the same feature pool $j$

are denoted as $f_j$: $W = \{X_{jk}, Y_{jk} \mid j = 1,...N; k = 1,...,L\}$, where $L$ is the number of training samples using the $j$th feature pool, $X$ is the feature vector, $Y$ is the label, and $k$ is the node index. Training a multiple number of classifiers for each node using the same feature pool $f_m$ is then transformed into a joint optimization problem as follows:

$$\min\{C\sum_{k=1}^{L}\sum_{j=1}^{N}\xi_{ij} + \beta_1\sum_{k=1}^{L}\|V_k\|^2 + \beta_2\|W_c\|^2\} \tag{2}$$

subject to:

$$\forall_{j=1}^{N}\forall_{k=1}^{L} : Y_{jk}(W_c + V_k)\cdot X_{jk} + b \geq 1 - \xi_{jk}, \xi_{jk} \geq 0 ,$$

where $\xi_{jk} \geq 0$ represents the training error rate, $\beta_1$ and $\beta_2$ are positive regularization parameters, and $C$ is the penalty term. The dual optimization problem for the above equation is to determine the optimal $\alpha_{jk}^{*}$ by:

$$\max\left\{\sum_{k=1}^{L}\sum_{j=1}^{N}\alpha_{ij} - \frac{1}{2}\sum_{k=1}^{L}\sum_{j=1}^{N}\sum_{h=1}^{L}\sum_{l=1}^{N}\alpha_{ih}Y_{ih}\alpha_{kl}Y_{kl}K_{kh}(X_{jh,}X_{kl})\right\} \tag{3}$$

subject to

$$\forall_{j=1}^{N}\forall_{k=1}^{L} : 0 \leq \alpha_{jk} \leq C, \sum_{k=1}^{L}\sum_{j=1}^{N}\alpha_{jk}Y_{jk} = 0 ,$$

where $K_{kh}(.,.)$ is the underlying kernel function. Our multi-task learning algorithm is able to handle the visual similarity among the node classes using the same feature pool.

In our method, a regression model is trained using a tree-structure-based feature-label selection algorithm, which is then used for annotating test images. This means that we have to consider a weight for each node in the tree structure. We can introduce the group lasso [15] into our algorithm, since we also have $M$ subsets which can be considered as $M$ positive definite matrices. The group lasso estimate is defined as the solution to:

$$\min_{\beta_k}\{P\sum_{k=1}^{L}\sum_{j=1}^{N}\xi_{jk} + \beta_1\sum_{k=1}^{L}\|V_k\|^2 + \beta_2\|W_c\|^2\} + \lambda\sum_{j=1}^{M}\sqrt{d_j}\,\|\beta_{kj}\|_2 , \tag{4}$$

where $\lambda > 0$ is the tuning parameter.

*3.4 Weighting the Nodes in the Tree Structure*

Since we have solved the first part of the equation (2) in Sec.3.3, we can now explore the regression part in (4). As we have a tree structure representing the correlations between the labels, we compute the hierarchical relationships between the nodes. With the tree structure constructed, each node will output the probabilities of the prediction labels for a testing image. Fig. 2 illustrated the construction of a 6-layer graph. Suppose that $\{l_1, l_2,\dots, l_n\}$ form the label set of all the training images, where each of the labels forms a leaf node in the tree structure. Putting the leaf nodes in the first layer (i.e. *Layer* 0) of the tree structure, we denote $\Phi_{layer-0} = \{v_1, v_2, \dots, v_n\}$, where $v_t$ represents the $t^{th}$ node in *Layer* 0 of the tree structure. For the *Layer* 1, *Layer* 3, and *Layer* 5, we have:

$$\Phi_{layer-1} = \{v_{n+1}, v_{n+2}, \dots, v_{n+l'}\},$$
$$\Phi_{layer-3} = \{v_{n+l'+1}, v_{n+l'+2}, \dots, v_{n+l'+l''}\}, \text{ and}$$
$$\Phi_{layer-5} = \{v_{n+l'+l''+1}\}. \tag{5}$$

where $l'$ and $l''$ represent the number of nodes in *Layer* 1 and *Layer* 3, respectively.

As *Layer* 2 and *Layer* 4 are treated differently from *Layer* 1, *Layer* 3 and *Layer* 5, we call these two layers the extra branches. The corresponding extended nodes are distributed on these two branches. Similarly, we have

$$\Phi_{layer-2} = \{v_{e_1}, v_{e_2}, \dots, v_{e_{\alpha'}}\},$$
$$\Phi_{layer-4} = \{v_{e'_1}, v_{e'_2}, \dots, v_{e'_{\beta'}}\}. \tag{6}$$

where $\alpha'$ and $\beta'$ represent the number of nodes in *Layer* 2 and *Layer* 5, respectively.

In Fig. 2, it seems that there is no difference between the normal layers and the extra branches. However, in our algorithm, they are located at different levels in the hierarchy. Motivated by the book “thinking, fast and slow” [36], authored by a Nobel Prize winner, we devise a framework to deal with the annotation problem like humans do. This means that we classify the label-dictionary into 2 clusters. One contains simple words which do not require much thinking, while the other one contains complex or abstract words that need more thought. For example, when humans need to solve the annotation work, we can label a picture having an "apple" in it in no time. However, if the ground-truth label of a picture is either "Africa" or "Tropical", we need to think it over before we decide its labels. It is reasonable for our system to treat these two clusters separately: one cluster contains simple, basic words, while the other contains complex, abstract words. Then, different weighting factors are assigned to these nodes, so we can treat the different words unequally.

Following [34], given an arbitrary tree $T_r$, each node $v \in V$ of the tree $T_r$ is associated with the group of children nodes $G_v$, whose members consist of all of the leaf nodes in the subtree rooted at node $v$. We recursively apply the operations below, starting from the root node and moving towards the leaf nodes, as follows:

$$\sum_k \sum_{v \in \Phi} w_v \left\| \beta_{G_v}^k \right\|_2 = \lambda \sum_k W_k(v_{root}) \, , \tag{7}$$

where $W_k(v) = s_v \cdot \sum_{c \in Childen(v)} |W_k(c)| + g_v \cdot \left\| \beta_{G_v}^k \right\|_2$ if $v$ is an internal node, and $W_k(v) = \sum_{m \in G_v} \left| \beta_m^k \right|$ if $v$ is a leaf node. Specifically, $s_v$ and $g_v$ are associated with the internal node and the extended node, respectively, in the tree $T_r$. Moreover, $s_v$ represents the weight for selecting the labels associated with each of the children nodes of $v$ individually, and $g_v$ represents the weight for selecting them jointly. $\beta_{G_v}^k$ is a vector of regression coefficients $\{\beta_\sigma^k : \sigma \in G_v\}$. Each group of regression coefficients $\beta_{G_v}^k$ is weighted with $w_v$, which reflects the strength of the correlation within that group.

Different to the common trees, there are extra branches in our proposed structure. Thus, the weighting factors for these extended nodes are also considered, as follows:

$$W_k(v) = s_v \cdot \sum_{c \in Childen(v)} |W_k(c)| + \rho g_v \cdot \left\| \beta_{G_v}^k \right\|_2 \, , \tag{8}$$

where $\rho$ is the Memory Sharing Factor in our algorithm. This means that the weighting factors of extended nodes are derived from internal nodes. In other words, we share the memory from several simple words to a complex word. Then, we have the relationship between $w_v$ and $(s_v, g_v)$ as follows:

$$w_v = \begin{cases} g_v \prod_{m\in \text{Ancestors}(v)} s_m \text{ if } v \text{ is an internal node} \\ \rho g_v \prod_{m\in \text{Ancestors}(v)} s_m \text{ if } v \text{ is an extended node} \\ \prod_{m\in \text{Ancestors}(v)} s_m \text{ if } v \text{ is a leaf node} \end{cases} \tag{9}$$

As proved in [34], for each of the $k^{\text{th}}$ outputs, the sum of the weights $w_v$ for all the nodes $v \in V$ in $T_r$ whose group $G_v$ contains the $k^{\text{th}}$ output as a member equals one. In other words, we have:

$$\sum_{v:k\in G_v} w_v = \prod_{m\in \text{Ancestors}(v_k)} s_m + \sum_{l\in \text{Ancestors}(v_k)} g_l \prod_{m\in \text{Ancestors}(v_k)} s_m + \sum_{l'\in \text{Ancestors}(v_k)} \rho g_{l'} \prod_{m\in \text{Ancestors}(v_k)} s_m = 1. \tag{10}$$

Equation (10) states that the sum of the weights over all of the groups that contain the given output variable is always one. After selecting the weights for the nodes in the tree, we can solve (4) using a multi-label boosting method [37].

The proposed framework aims to construct a node-based classifier in the first stage. Our tree divides labels into different categories by means of internal nodes and extended nodes. Different weights given to different clusters of labels make our system treat simple words and complex words differently. To the best of our knowledge, this work is the first to solve image annotation based on two layers of thinking, with the help of a tree structure. Experiments, analyses and experiment results are given in the following section to show the performance of our proposed annotation framework.

## 4. Experiments and Results

### *4.1. The Databases Used*

Four benchmark image databases are used in our framework. The first database used in the experiments is Corel 5K, which contains 5,000 images comprising 4,500 training samples and 500 testing samples. Each image in the dataset is annotated with about 3.5 keywords on average, and the dictionary has a total of 374 words or labels. Another dataset used is Corel 30K, which is similar to Corel 5K except that it is substantially larger, containing 31,695 images and 5,587 words or labels. The third dataset used is the ESP Game dataset, which contains 18,689 training images and

2,081 testing images. The last database is NUS-Wide [38], which has 269,648 images and the associated tags from Flickr, with a total number of 5,018 unique tags. Since the images and tags are downloaded from websites, NUS-Wide is a real-life image dataset. We choose 1,000 common labels, and randomly sampled about 25,000 related images from the NUS-Wide.

To make a fair comparison with other state-of-the-art methods, we choose the mean precision rate (*P* %), the mean recall rate (*R* %), and the number of total keywords recalled (*N*+) as our evaluation criteria on Corel 5K, Corel 30K and ESP Game. The precision rate and recall rate for each test image are measured by comparing the annotation results to the ground truth, and then the average precision and recall of all the test images are computed to form the final results. Mean Average Precision (MAP) is also employed for performance comparison on the Nus-Wide database.

### *4.2. Methods of Comparisons*

We have evaluated the performance of our proposed framework and compared it to the following methods.

1. Multiple Bernoulli Relevance model (MBRM) [39], a method follows a mixture distribution that models the joint distribution of annotation labels and image features.
2. Supervised Multiclass Labeling (SML) [40], a probabilistic formulation for semantic image annotation.
3. Joint Equal Contribution method (JEC) [15], a method allows each individual basic distance to contribute equally to the total combined cost from different features.
4. L1-Penalized Logistic Regression (Lasso) [15], an approach that combines feature distances by identifying those features that are more relevant for capturing image similarity.
5. Tagprop [12], a discriminatively trained nearest neighbor model, where the keywords of the test image are predicted by a weighted nearest neighbor model with the image similarity metrics learned by a metric learning method.

6. HPM [41], a method making use of prior knowledge (i.e. some labels already known), and completing labels from a probabilistic model.
7. Label filtering algorithm (LFA) [16], an approach firstly removes most of irrelevant labels and then annotate test images using those remaining relevant labels.

## *4.3. Features Pool Used*

Like our previous work in [31], the feature pools used in the following experiments can be divided into three parts: global features, local features, as listed below. The order of the features listed represents the different feature-pool types used in the experiments.

Global features:

(1) Color feature: RGB color moment ($3\times3$ grid, color mean, variance, skewness for R, G, B),

(2) Edge histogram (edge-orientation histogram, Canny edge detector), and

(3) SIFT (based on a regular grid).

Local features:

(4) Gabor wavelets (5 scales and 8 orientations, 3 moments for each sub-image),

(5) Local binary patterns (59-dimensional LBP histogram),

(6) SIFT (based on the interesting points), and

(7) R-CNN (region-based convolutional neural network, only used on the NUS-wide database).

(8) Faster-RCNN (region-based convolutional neural network, only used on the NUS-wide database)

## *4.4 Results Based on the Corel 5K Dataset*

Figure 4 shows the relative performances of the above six types of feature pool used for classifying the class labels. For each feature pool, the number of class labels that can be best classified is measured. It can be observed that the different feature pools have different performances in the various node classes, which means that we can identify and use the most suitable feature pool for each single node in this step. We also believe that the same feature pool can best be used to describe a number of label classes. Combining the features may not always increase the performance of image-classification tasks; in some situations, combining a large number of features may cause the problem of redundancy and lead to a decreased performance. Therefore, rather than combining the features, our algorithm finds the most suitable feature type for each single node.

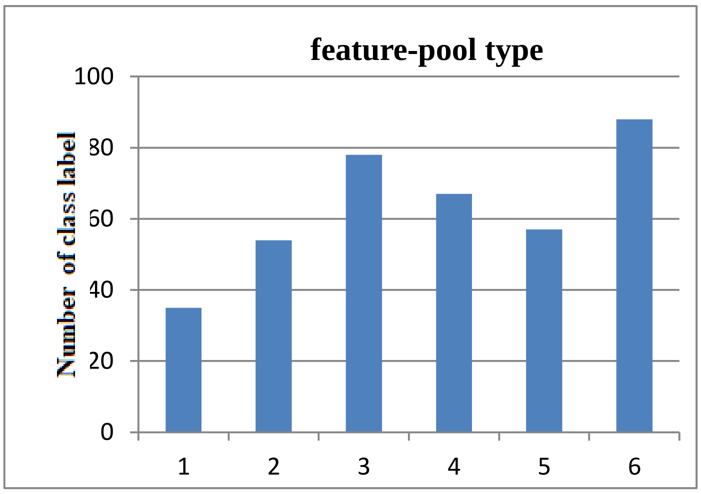


Fig. 4: The number of class labels which results in the best performance based on the same feature-pool type.

After identifying the most suitable feature pool for each single node, we have to explore the tree structure used in the next step for label prediction. Figure 5 shows the performance of our learning framework with and without using the tree structure.

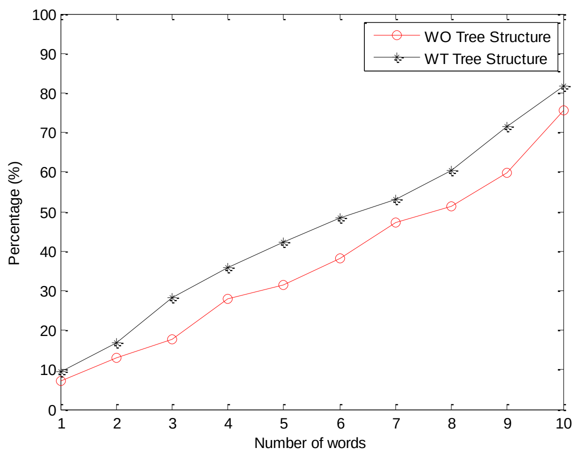


Fig. 5. Percentages of the testing images that can be annotated correctly by at least one word among the top *n* words using our proposed learning framework, with and without using the tree structure.

**Table 2** Performances based on the Corel 5K dataset for some existing methods and our proposed method.

| Methods | $P\%$ | $R\%$ | $N^+$ |
|---|---|---|---|
| MBRM[39] | 24 | 25 | 122 |
| SML [40] | 23 | 29 | 137 |
| JEC [15] | 27 | 32 | 139 |
| LASSO[15] | 24 | 29 | 127 |
| TagProp[12] | 33 | 42 | 160 |
| HPM(with prior knowledge)[41] | 33 | 47 | 162 |
| HPM(without prior knowledge)[41] | 25 | 28 | 136 |
| LFA[16] | 31 | 40 | 151 |
| Proposed Method | 36 | 44 | 165 |

Table 2 shows the performances of our proposed method versus some state-of-the-art methods. Three performance indices are measured: the mean precision rate ($P$ %), the mean recall rate ($R$ %), and the number of total keywords recalled ($N$+), respectively. Our method outperforms all the other methods in terms of the mean precision rate, which is the most important measurement. Our method also achieves a better performance than most of the other methods in terms of the mean recall rate. Although the recall of HPM is slightly better than ours, it is unfair to compare ours directly to those methods using already-known labels. Nevertheless, our method is still better in terms of other performance indices.

The tree structure has hierarchical relationships among the nodes. Our system treats the different labels (simple or complex) in their own ways. With the help of the tree structure, the normal layers and extra branches are weighted differently. Figure 6 shows the performance with and without using the tree, as a trade-off between precision and recall for this dataset.

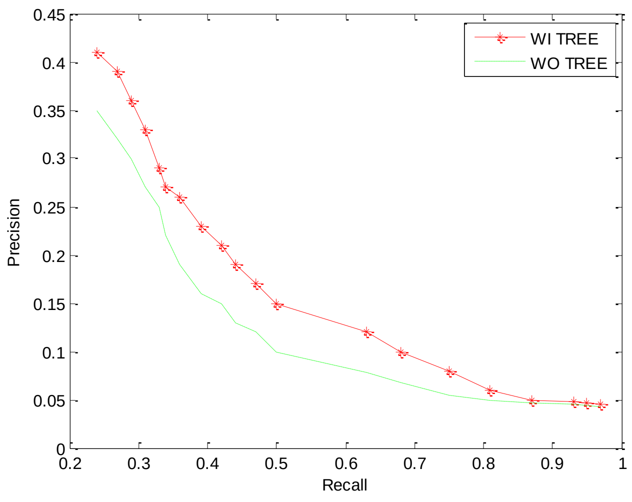


**Fig. 6.** Precision-recall plots generated by varying the number of keywords assigned to an image with and without using the tree structure.

We also control the number of layers in our tree structure, which may affect the performance. Our tree consists of label pairs and label triplets placed at different layers. Table 3 shows the results of our tree structure with (WT) and without (WO) including label triplets. The most important factor of performance, i.e. mean precision rate, is given.

**Table 3** Performance of our algorithm on Corel5K with and without including label-triplets.

| | WO label-triplets | WT label-triplets |
|---|---|---|
| mean precision rate (*P* %) | 32 | 36 |

We also analyze the complexity of our annotation algorithms. We assume that there are $N$ training images, and that each image is represented by an $R$-dimensional visual feature vector. The complexity of MBRM [39] is $O(NR)$, while SML [40] has the complexity of $O(TR)$, where $T$ is the number of semantic classes. TagProp, which can achieve a good performance, has the same time complexity as MBRM, since it is based on the KNN-based model. The framework in [16] filters most of the irrelevant labels, resulting in a reduction of complexity by almost 20% compared to SML. Since the method proposed in this paper is also based on feature-pool selection, with the help of the tree structure, the complexity of our method is about $O(T_N R)$, where $T_N$ is the number of nodes in the tree structure. Figure 7 presents the per-image annotation time required by each of the methods on the Corel dataset as a function of $N$. The time complexity of our method is acceptable, and our performance is the best among all the methods.

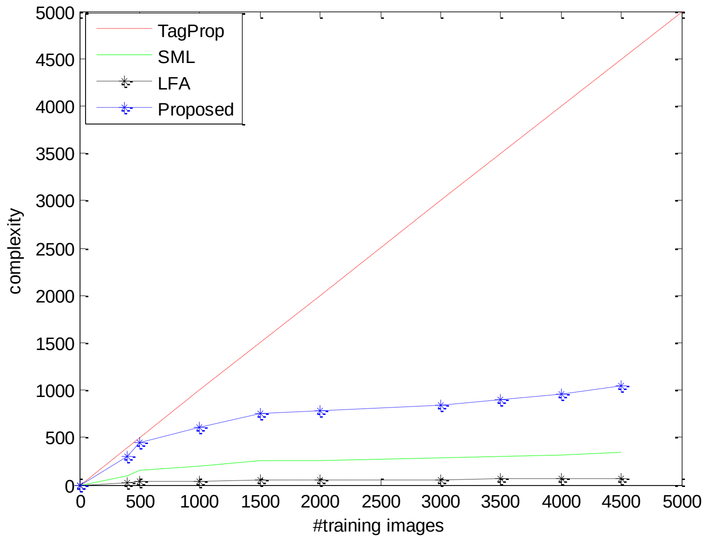


**Fig. 7.** Comparison of the time complexity for annotating a test image on Corel 5K.

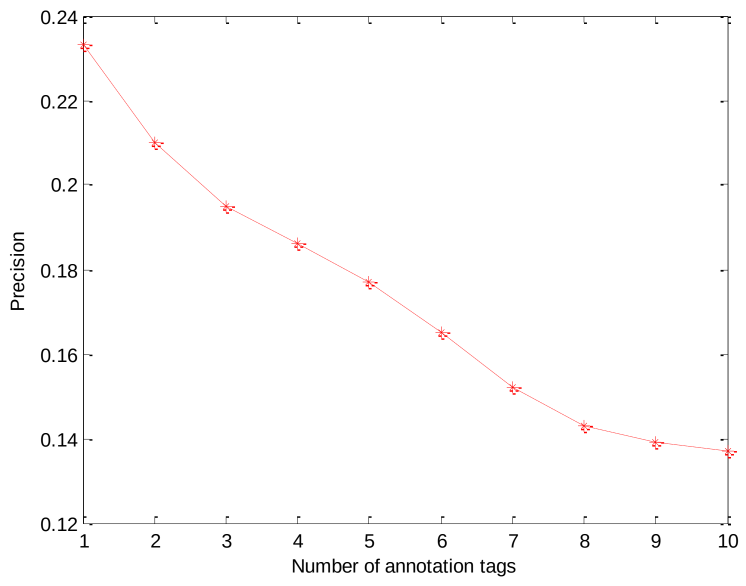
## *4.5 Results on the Corel 30K Dataset and ESP Game Dataset*

In this section, a larger dataset is used to evaluate the performance of our framework. We follow the same procedure as in [41]: we choose only those words that are used as labels for more than 10 images in Corel 30K to form the semantic vocabulary. The average number of labels per image is about 3.6. To evaluate our method for large-scale annotation tasks, we compare the performance of our algorithm with HPM [41], SML [40], and our previous work on the Corel 30K dataset. The results are tabulated in Table 4. We can see that our proposed method is better and that its performance is very promising.

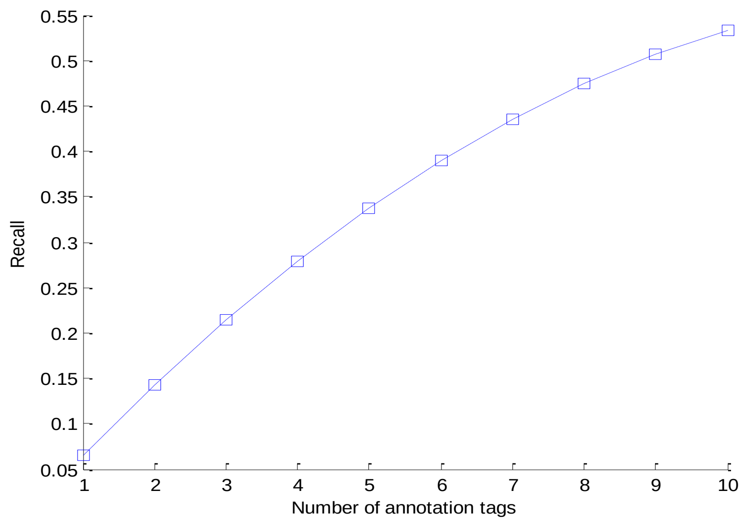
**Table 4** Performance comparison on the Corel 30K dataset.

| Method | SML[40] | HPM, Given 0 | HPM, Given 1 | LFA[16] | Prop. Work |
|---|---|---|---|---|---|
| Avg. Prec. | 0.12 | 0.10 | 0.16 | 0.13 | 0.17 |
| Avg. Recall | 0.21 | 0.19 | 0.31 | 0.24 | 0.33 |
| Rate$^{+}$ | 44.63% | 46.21% | 55.71% | 49.89% | 57.13% |

Fig 8 shows the precision and recall rates when the annotation length changes from 1 to 10 on the Corel 30K dataset. Table 5 shows some predicted labels, as well as the corresponding ground-truth, on the Corel 30K database. Those incorrect predicted labels are in italics. Table 6 compares the performances of the different methods in terms of precision, recall, and number of recalled keywords. The results show that our method always achieves better results on the ESP Game dataset.



(a)



(b)

**Fig. 8.** Performance of our proposed method based on a test set of 1500 tags: (a) Precision, and (b) Recall.

**Table 5** Predicted labels versus ground-truth (the differences are marked in italics) on Corel 30K.

| | | | | |
|---|---|---|---|---|
| Predicted labels | crowed people ceremony *hat book* | tree snow sky *blue clouds* | pizza cuisine food meal *egg* | city trees, *forest building* sky |
| Ground Truth | people wall crowd ceremony | snow trees sun mountain | pizza food meal cuisine | city trees, horizon sky |
| | | | | |
| Predicted labels | giraffe grass *African* trees *animals* | bear polar snow *fur* tundra | building *house* trees *clouds* roof | fireworks night *burst* water *lights* |
| Ground-Truth | giraffe trees grass sky | polar bear snow tundra | building roof trees sky | fireworks night city water |

**Table 6** Performance comparison on the ESP Game dataset.

| Method | JEC [15] | TagProp [12] | AICDM [42] | LFA[16] | Prop. Work |
|---|---|---|---|---|---|
| Avg. Prec. | 0.22 | 0.39 | 0.24 | 0.35 | 0.44 |
| Avg. Recall | 0.25 | 0.27 | 0.26 | 0.25 | 0.30 |
| $N^+$ | 224 | 239 | 231 | 228 | 261 |

## *4.6 Results on the NUS-Wide Dataset*

The NUS-WIDE dataset is a huge dataset, which consists of 269,648 images. In this experiment, we choose 1,000 common labels, and randomly sample about 25,000 related images from NUS-Wide. Three experiments have been done on this database. Besides the traditional features utilized in our first experiment, the R-CNN [43] features were employed in our second experiment. Since many baseline methods cannot be applied to such a large dataset, we show only the results for TagProp, HPM, and our proposed method in the first experiment. The average precision and recall for the first 5 and 10 returned tags, respectively, for the NUS-Wide dataset are shown in Table 7. We can observe that our proposed framework outperforms most of other methods on this huge dataset. Although HPM (given 2) has a slightly better performance than ours, our performance is promising without the help of any prior knowledge (i.e. labels already known).

In the second experiment, a deep feature is utilized in our framework. Specifically, R-CNN is employed to generate the deep feature. We implemented R-CNN as follows. The proposal network (RPN) [44] is trained based on the images for each node in our tree, which is further used to predict the bounding boxes for each ground-truth label. These predicted boxes play an important role of node-independent proposals. Like [45], the highest scored 200 proposals are extracted as training samples to train an R-CNN classifier. In this part, we compare it with state-of-the-art methods on the dataset, including CNN-RNN [23], WARP [46], and R-CNN [43]. The MAP results of all these methods are listed in Table 8, which shows that the proposed framework with deep features substantially outperforms all the deep learning methods compared. Although it is just an attempt to combine deep learning with our framework, it proves that efficient and accurate annotation can be achieved by the use of handcrafted features, while the performance will be further improved with the help of deep features.

In the third experiment, Faster-RCNN [47] is utilized in our framework. In this part, our annotation system is composed of two modules. In the first module, the Faster-RCNN helps predict the simple object label with the bounding box in the output images. The obtained labels in this part are considered as the simple labels in our tree. The tree-learning model is introduced in our second module. The input of our tree-learning module consists of two label categories. The first category is the simple labels output from the Faster-RCNN, and the second category is the other labels collected from the NUS-WIDE dataset, which is called extended label in our framework. The output of the tree-learning model is composed of single labels and complex labels.

As shown in Figure 9, the training process is divided into two steps. The Faster-RCNN network, trained on the NUS-Wide dataset, firstly employs the original images with a single label. Then, the tree-learning model is also trained on NUS-Wide, with the introduction of all the labels including the simple and complex labels. After the training process, Faster-RCNN detects the simple labels in the original images, while our tree-learning model groups the simple labels output from the Faster-RCNN, and the most suitable complex labels together and outputs the final annotations. An example is shown in Figure 9, where an original image is fed into Faster RCNN and simple object labels (car, dog, horse, person) are obtained. Then, the complex label (hunter) is output from our tree-learning model after that.

Table 9 illustrates the predicted labels produced by the Faster-RCNN module and our AMS tree-learning module. Traditional deep-learning models can only detect simple object labels, as shown in the single label row of Table 9 (bus, person, road, sofa, dog, train, fence, sheep, etc.). With the help of our AMS tree-learning module, the complex labels with abstract and confusing meaning can be achieved (cityscape, pets, hunter, apartment, race, farm, etc.). The combination of Faster-RCNN and our AMS tree-learning module shows the success of our annotation system, which also proves the promising performance of our framework.

**Table 7** Average precision and recall on the NUS-Wide dataset.

| | TagProp [12] | HPM(given 0) | HPM(given 2) | Proposed |
|---|---|---|---|---|
| AP@5(%) | $29 \pm 2.5$ | $19 \pm 2.0$ | $33 \pm 1.4$ | $31 \pm 1.2$ |
| AR@5(%) | $38 \pm 1.6$ | $25 \pm 2.4$ | $41 \pm 2.1$ | $39 \pm 1.0$ |
| AP@10(%) | $21 \pm 2.2$ | $13 \pm 1.5$ | $24 \pm 1.7$ | $23 \pm 1.8$ |
| AR@10(%) | $41 \pm 1.8$ | $29 \pm 1.0$ | $44 \pm 1.3$ | $42 \pm 1.3$ |

**Table 8** Mean Average Precision (MAP) on the NUS-Wide dataset.

| Method | MAP(%) |
|---|---|
| CNN-RNN [23] | 70.3 |
| WAPR [46] | 65.7 |
| R-CNN[43] | 71.1 |
| Proposed+R-CNN | 76.2 |

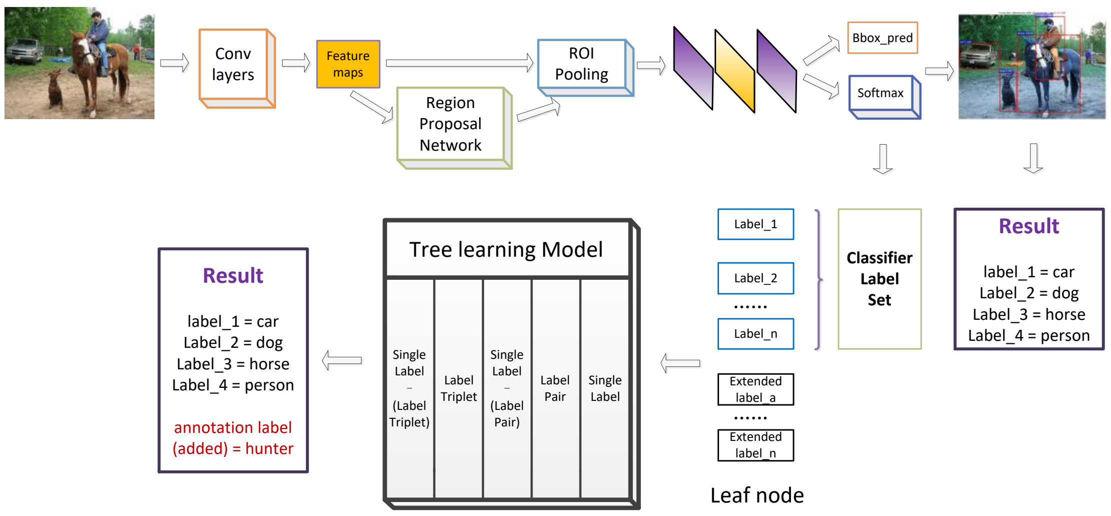
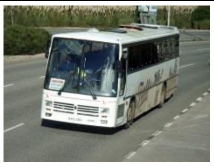


**Fig. 9.** The flow path of our AMS+Faster-RCNN framework on NUS-Wide.

**Table 9** Predicted simple labels by Faster-RCNN, and complex labels added by AMS (marked in italics)

| | | | | |
|---|---|---|---|---|
| Input image |  | | | |
| Single label | bus<br>person<br>road | person<br>sofa<br>dog | car, dog<br>horse<br>person | dining-table<br>potted-plant<br>chair |
| Annotation label (added) | *cityscape* | *pets* | *hunter* | *apartment* |
| Input image | | | | |
| Single label | person<br>horse<br>fence | sheep<br>cow<br>stone | person<br>dining-table<br>bottle | train<br>tv-monitor<br>window |
| Annotation label (added) | *race* | *farm* | *dinner* | *traffic* |

## 5. Conclusions

This paper presents a new hierarchical model for efficient image annotation, which employs an adaptive learning algorithm to select an optimal feature subset for each label. A tree structure is constructed using our proposed Associate Memory Sharing (AMS) algorithm, and a regression model is trained using the tree structure based on our proposed feature-label-selection algorithm. Making use of the tree, the relationships among the labels are considered, which can highly improve the performance of our multi-task learning algorithm. Our model treats simple words and complex words separately by assigning different weights to different clusters of labels. By imitating the thinking of human beings in a simple way, our system can alleviate the semantic-gap problem. Our proposed method works well on all four well-known datasets in terms of annotation benchmarking. The experiment results of our framework have not only shown promising performance, but that it can achieve both efficiency and accuracy in image annotation.

## References

[1] J. Shao, K. Kang, C. Change Loy, and X. Wang. “Deeply learned attributes for crowded scene understanding,” In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4657–4666, 2015.

[2] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan,P. Dollar, and C. L. Zitnick. “Microsoft coco: Common objects in context,” In ECCV, 2014.

[3] K. Kang, W. Ouyang, H. Li, and X. Wang. “Object detection from video tubelets with convolutional neural networks,” In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 817–825, 2016.

[4] Mattew R.Boutell, Jiebo Luo, Xipeng Shen, Christopher M. Brown “Learning multi-label scene classification,” Pattern Recognition 37 (9) (2004) 1757-1771.

[5] Z. Zhang, E. Pasolli, M. Crawford, and J. Tilton, “An active learning framework for hyperspectral image classification using hierarchical segmentation,”IEEE J. Sel. Topics A, 2016

[6] A. Alzu’bi, A. Amira, N. Ramzan, “Semantic content-based image retrieval: A comprehensive study”, Journal of Visual Communication & Image Representation, Vol. 32, pp. 20-54, 2015.

[7] Jie Zhu, Jian Yu, Chaomurilige Wang, Fan-Zhang Li,“Object recognition via contextual color attention,”Journal of Visual Communication & Image Representation, Vol. 27, pp. 44-56, 2015.

[8] L Gomez ，Y Patel ，M Rusinol ，D Karatzas ，CV Jawahar。“Self-supervised learning of visual features through embedding images into text topic spaces, ”IEEE Conference on Computer Vision & Pattern Recognition , 2017

[9] F.Monay, D.Gatica-Perez. PLSA-Based Image Auto-Annotation: Constraining the Latent Space, ACM Multi-media (2004) 348-351.

[10] K.Barnard, P. Duygulu, D Forsyth, N. de Freitas, D. M. Blei, M. I. Jordanetc, Matching Words and Pictures, Journal of Machine Learning Research 3 (2003) 1107-1135.

[11] David Grangier, Samy Bengio. A Discriminative Kernel-Based Model to Rank Images from Text Queries, IEEE Transactions on Pattern Analysis and Machine Intelligence, 30 (8) (2008) 1371-1384.

[12] M. Grubinger, T. Mensink, J. Verbeek, C. Schmid. Tagprop: Discriminative Metric Learning In Nearest Neighbor Models for Image Auto-Annotations, in: Proceedings of the International Conference on Computer Vision, (2009) 309 - 316.

[13] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan. Cnn: Single-label to multi-label. arXiv preprint arXiv:1406.5726, 2014. 1, 2

[14] H. Zhang, A. Berg, M. Maire, J. Mailik. SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2006) 2126–2136.

[15] Ameesh Makadia, Vladimir Pavlovic, Sanjiv Kumar. A New Baseline for Image Annotation, in: Proceedings of the European Conference on Computer Vision, (2008) 316-329.

[16] J. Hu, K.M. Lam, G. Qiu. A Hierarchical Algorithm for Image Multi-labeling, in: Proceedings of the IEEE International Conference on Image Processing (ICIP’2010), (2010) 2349–2352.

[17] S. Zhang, J. Huang, Y. Huang, Y. Yu, H. Li and D. N. Metaxasetc. Automatic Image Annotation Using Group Sparisty, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2010) 3312 - 3319.

[18] Tang, Mengfan, F Nie , S Pongpaichet and R Jain "Semi-supervised learning on large-scale geotagged photos for situation recognition." Journal of Visual Communication and Image Representation 48 (2017): 310-316.

[19] D. Zhou, J. Huang, B. Scholkopf, Learning with Hypergraphs: Clustering, Classification, and Embedding, in: Advances in Neural Information Processing Systems, (2006) 1601-1608.
[20] J. Hu, C. Sun, K.M. Lam, Learning a Discriminative Model for Image Annotation, in: Proceedings of Asia Pacific Information and Signal Processing Association, ASC 2011.
[21] Pongpaichet, Siripen, et al. "Using photos as micro-reports of events." Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval. ACM, 2016.
[22] H. Hu, G.-T. Zhou, Z. Deng, Z. Liao, and G. Mori. Learning structured inference neural networks with label relations. CVPR, 2016.
[23] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu. Cnn-rnn: A unified framework for multi-label image classification. CVPR, 2016.
[24] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In CVPR, 2015.
[25] Q.Li, M.Qiao, W.Bian, and D.Tao. Conditional graphical lasso for multi-label image classification. In CVPR, 2016.
[26] X.Li, F.Zhao, and Y.Guo. Multi-label image classification with a probabilistic label enhancement model. Proc. Uncertainty in Artificial Intell, 2014.
[27] J.Wang, Y.Yang, J.Mao, Z.Huang, C.Huang, and W.Xu. Cnn-rnn: A unified framework for multi-label image classification. CVPR, 2016.
[28] M.Choi, J.Lim, A. Torralba, and A. Willsky, “Exploiting hierarchical context on a large database of object categories,” in CVPR,2010
[29] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie, “Objects in context,” in ICCV, 2007.
[30] Thomas Mensink, Jakob Verbeek, and Gabriela Csurka, “Tree-structured CRF Models for Interactive Image Labeling”, in PAMI 2012.
[31] J. Hu, K.M. Lam, An Efficient Two-stage Framework for Image Annotation. Pattern Recognition 46(3): (2013) 936-947.
[32] J. Hu, K.M. Lam, Ping Lou and Quan Liu, Constructing a hierarchical tree for image annotation. In IEEE international conference on Multimedia and Expo(ICME), 2017.
[33] K. P. Murphy, A. Torralba, and W. T. Freeman. Using the forest to see the trees: a graphical model relating features, objects and scenes. In NIPS, 2003. 2, 4
[34] Seyoung Kim and Eric P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In Proceedings of the 27th International Conference on Machine Learning, (2010) 543-550.
[35] Le, Q. V., Smola, A., Chapelle, O., Teo, C. H., Optimization of Ranking Measures, Journal of Machine Learning Research (2010).
[36] Daniel Kahneman. Thinking, Fast and Slow, Farrar, Straus and Giroux; Reprint edition, (2011).
[37] Fei Wu, Yahong Han, Qi Tian, Yueting Zhuang. Multi-label boosting for image annotation by structural grouping sparsity. ACM Multimedia, (2010) 15-24.
[38] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao Zheng. NUS-WIDE: A Real-World Web Image Database from National University of Singapore, ACM International Conference on Image and Video Retrieval. Greece. Jul.(2009) 8-10.
[39] S.L. Feng, R. Manmatha, V. Lavrenko. Multiple Bernoulli Relevance Models for Image and Video Annotation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2004) 1002-1009.

[40] G. Carneiro, A.B. Chan, P.J. Moreno, N. Vasconcelos. Supervised Learning of Semantic Classes For Image Annotation and Retrieval, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (3) (2007) 394-410.

[41] N. Zhou, W. Cheung, G. Qiu, X. Xue. A Hybrid Probabilistic Model for Unified Collaborative and Content based Image Tagging, IEEE Transactions on Pattern Analysis and Machine Intelligence 33 (7) (2011) 1281-1294.

[42] J.-H. Su, C.-L. Chou, C.-Y. Lin. Effective semantic annotation by image-to-concept distribution model. IEEE Transactions on Multimedia (2011), pp. 530-538.

[43] R. Girshick. Fast r-cnn. In ICCV, 2015.

[44] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[45] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CVPR, 2016.

[46] Y. Gong, Y. Jia, T. Leung, A. Toshev, and S. Ioffe. Deep convolutional ranking for multilabel image annotation. ICLR, 2014.

[47] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," In IEEE Transactions on Pattern Analysis and Machine Intelligence (2017), pp. 1137-1149.