This ICCV2013 paper is the Open Access version, provided by the Computer Vision Foundation.
The authoritative version of this paper is available in IEEE Xplore.

Find the Best Path: an Efficient and Accurate Classifier for Image Hierarchies

Min Sun[1] Wan Huang[2] Silvio Savarese[3]
[1]Dept. of Computer Science and Engineering, University of Washington, USA
[2]Dept. of Electrical and Computer Engineering, University of Michigan at Ann Arbor, USA
[3]Dept. of Computer Science, Stanford University, USA
sunmin@cs.washington.edu wanhuang@umich.edu ssilvio@stanford.edu

Abstract

Many methods have been proposed to solve the image classification problem for a large number of categories. Among them, methods based on tree-based representations achieve good trade-off between accuracy and test time efficiency. While focusing on learning a tree-shaped hierarchy and the corresponding set of classifiers, most of them [11, 2, 14] use a greedy prediction algorithm for test time efficiency. We argue that the dramatic decrease in accuracy at high efficiency is caused by the specific design choice of the learning and greedy prediction algorithms. In this work, we propose a classifier which achieves a better trade-off between efficiency and accuracy with a given tree-shaped hierarchy. First, we convert the classification problem as finding the best path in the hierarchy, and a novel branch-and-bound-like algorithm is introduced to efficiently search for the best path. Second, we jointly train the classifiers using a novel Structured SVM (SSVM) formulation with additional bound constraints. As a result, our method achieves a significant 4.65%, 5.43%, and 4.07% (relative 24.82%, 41.64%, and 109.79%) improvement in accuracy at high efficiency compared to state-of-the-art greedy "tree-based" methods [14] on Caltech-256 [15], SUN [32] and ImageNet 1K [9] dataset, respectively. Finally, we show that our branch-and-bound-like algorithm naturally ranks the paths in the hierarchy (Fig. 8) so that users can further process them.

1. Introduction

Large-scale visual classification is one of the core problem in computer vision. Thanks to the ongoing efforts on collecting large scale dataset such as ImageNet [9] (15,589 non-empty synsets) and SUN dataset [32] (899 scene categories), researchers have been strongly motivated to design methods that can scale up to large number of classes.

Among them, methods based on tree-based representations ("tree-based" methods) [11, 4, 2, 3, 16, 14, 6, 30, 23, 35] achieve good trade-off between accuracy and test time efficiency. These methods organize the large number of classes into a tree-shaped hierarchy. At each internal node, a classifier separates the classes into smaller subsets to the nodes at a lower level (Fig. 1(a)). In order to achieve test time efficiency, a greedy prediction algorithm is used to progressively reduce the number of classes by applying the classifiers from the top to the bottom level in the hierarchy. Thus, the algorithm achieves a desired "sublinear complexity" (in the number of classes) during testing. In order to achieve high accuracy, most methods focus on learning the tree-shaped hierarchy and the corresponding set of classifiers. However, we argue that the trade-off achieved are sub-optimal mainly due to the following reasons:

- **Greedy algorithm.** The prediction algorithm only explores one single path in the tree-shaped hierarchy; and the path is *greedily* selected according to the order of the classifiers applied (Fig. 1(b)-Left). This implies that errors made at a higher level in the hierarchy cannot be corrected later.
- **Inefficient usage of training data.** Rather than learning all classifiers jointly, most "tree-based" methods learn one classifier at a time from the top level to the bottom level gradually using fewer images corresponding to a subset of classes. Thus, the classifiers at a lower level are trained using less data than the ones trained at a higher level.

In this paper, we propose an efficient and accurate classifier for image hierarchies which achieves a better trade-off with a given tree-shaped hierarchy. Our contributions in model representation, efficient inference, and learning algorithm are summarized below.

- **Best path model.** We convert the *greedy* prediction model into a "Best Path Model", where the objective is to find the best path in the tree-shaped hierarchy corresponding to the maximum sum of the classifiers responses. We formulate model learning as a Structured SVM (SSVM) [29] problem (Eq. 4) to jointly train all classifiers jointly using all training data.

- **Novel branch-and-bound-like search.** We further propose a novel *branch-and-bound-like* searching approach to efficiently find the best path without exploring the whole tree nor greedily pruning out classes (Fig. 1(b)-Right).
- **Learning Bounds.** Both the bounds and the model parameters can be jointly learned using an extended SSVM formulation with additional bound constraints (Eq. 3) which allows us to search for a better trade-off between efficiency and accuracy.

Our method is closely related to the one-versus-all and "tree-based" methods as discussed in Sec. 3.2. On one hand, our method is equivalent to an one-versus-all method encoded with a hierarchical structure. On the other hand, the greedy "tree-based" methods are also a special case of our method. By preserving both the joint training strategy of the one-versus-all methods, and the hierarchical organization of classes in "tree-based" methods, we show that i) our best accuracy outperforms the accuracy of the one-versus-all method [33] (Sec. 6); ii) our method achieves better trade-off between efficiency and accuracy than the state-of-the-art greedy tree-based method [14]. Most importantly, we achieve a significant 4.65%, 5.43%, and 4.07% (relative 24.82%, 41.64%, and 109.79%) improvement in accuracy compared to [14] on Caltech-256, SUN, and ImageNet 1K dataset, respectively, at high efficiency. Finally, our branch-and-bound-like algorithm naturally ranks the paths in the hierarchy to maintain a diverse set of class predictions without significantly increasing the complexity (Fig. 5).

The remainder of the paper is organized as follows. Sec. 2 describes the related work. Detail of our model representation, efficient branch-and-bound-like algorithm, and learning for a better trade-off are elaborated in Sec. 3, 4, and 5, respectively. Finally, experimental results are shown in Sec. 6.

2. Related Work

Recently many methods for large scale classification have been proposed. Despite their differences, they can be divided into two groups according to how they convert the multi-class classification problem into multiple binary classification problems.

The first group of methods is generic and do not assume that classes are organized into a hierarchy. It includes methods based on "one-versus-all" and "one-versus-one" strategies, which further assume classes are unrelated (e.g., do not share features). It also includes *error correcting output codes* [12, 1], *output-coding based* methods [26, 21], and [28] which utilize the relationship between classes (e.g., sharing features) to build more compact and robust models. These methods typically show good classification accuracy. However, the time complexity for evaluating the classifiers are "linearly" proportional to the number of classes.

The second group of methods aims at reducing the time complexity utilizing the hierarchical structure of classes. [11, 4, 2, 3, 16, 14, 6, 30, 23, 35] propose different methods to automatically build the hierarchy of classes. Other methods [37, 5] rely on a given hierarchy. Notice that, no matter how the hierarchies are obtained, most of them rely on a greedy algorithm to explore one single path in the hierarchy for class prediction and train the classifier at each node of the hierarchy separately (except [5]). Hence, the accuracy is typically sacrificed in order to achieve "sublinear" time complexity. In Sec. 3.1, we will describe their methods in details and summarize their issues.

Researchers also have looked at different aspects of the large-scale classification problem. [22, 31, 36, 25] focus on developing efficient and effective feature representations. Similarly, [18, 20] learn discriminative feature representations using sophisticated deep networks and achieve state-of-the-art accuracy. Notice that their contributions are orthogonal to ours since our method can use other feature representations easily to explore the trade-off between accuracy and efficiency. Hence, we do not optimize the performance over different features in this paper.

[10] propose the first method connecting class selective rejection with hierarchical visual classification so that they can generate object class labels at different levels in the hierarchy while guaranteeing an arbitrarily high accuracy. It is worth mentioning that our branch-and-bound-like algorithm possesses similar property. It ranks hypotheses at different levels and naturally maintains a diverse set of class predictions (properties described in Sec. 4.2).

Recently, [14] propose to learn a relaxed hierarchy which explores the trade-off between efficiency and accuracy. In the relaxed hierarchy, a unique class is allowed to appear at more than one nodes at the same level which means some classes are ignored when learning the hierarchy at a certain level. In this way, classes which are hard to separate are handled at a lower level to improve the accuracy. At the same time, the level of relaxation also controls the efficiency of the algorithm (i.e., the more relax the less efficient). In this paper, we explore the trade-off by finding the best path in the hierarchy (Sec. 3.2) using an efficient branch-and-bound-like algorithm (Sec. 4.1). Notice that our method is orthogonal to the relaxed hierarchical model [14]. In the experiments, we achieve a better trade-off between efficiency and accuracy by combining the relax hierarchy with our method (Fig. 4, 6, and 7).

Our method is also related to decision forest [7] which uses an ensemble of tree-based classifiers to improve accuracy. However, the complexity increases linearly to the number of trees. In contrast, our method increases the accuracy with the same complexity. Moreover, our method can become a forest by training a set of models using bagging.

Finally, our best path model is similar to [5]. However, they focus on learning the model efficiently by first training a set of local classifiers then combining them into a global classifier. In contrast, we focus on efficient prediction, and we show that efficient prediction can be used to make training efficient as well (Sec. 5).

3. Our Model

Recall that our goal is to design a classifier for image hierarchies which achieves a better trade-off between efficiency and accuracy. In this section, we first review and define the "tree-based" model, and then we show how to convert the model into a "Best Path Model".

3.1. Classical Tree-based Model.

The model organizes the set of K classes $\mathcal{K} = \{1, ..., K\}$ into a tree-shaped hierarchy $\mathcal{T} = \{\mathcal{V}, \mathcal{E}\}$ with a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$. Each node $v \in \mathcal{V}$ is associated with a set of classes $\mathcal{K}_v \subset \mathcal{K}$. The edges connect each node $v \in \mathcal{V}$ to N child nodes $\mathcal{C}_v = \{c_v^e\}_{e \in \{1,...,N\}}$ and are associated to N linear classifiers $W_v = \{w_v^e \in \mathbb{R}^D\}_{e \in \{1,...,N\}}$. The classifier associated to the edge e generates a score $S_v^e(x) = x^T w_v^e$ as a function of an input $x \in \mathbb{R}^D$. The model requires that the set of classes $\mathcal{K}_c$ at every child node is a subset of the classes $\mathcal{K}_v$ at its parent node (i.e., $\mathcal{K}_c \subset \mathcal{K}_v; \forall c \in \mathcal{C}_v$). The notations are illustrated in Fig. 1(c).

Greedy Algorithm. Given the model, a greedy algorithm is typically used to predict the class of the input $x \in \mathbb{R}^D$. The algorithm explores a single path in the tree from the root node until a leaf node is reached as follows. Starting from the root node (with index $v = 1$), the edge e corresponding to the largest classifier score is selected (i.e., $e = \arg\max_j S_v^j(x)$), and the algorithm proceeds to the child node c_v^e. The same selection procedure is iteratively applied to all the child nodes visited until a leaf node is reached. Thus, the predicted path P can be represented by the starting node $v = 1$ concatenated with the sequence of selected edges $[e_1, ..., e_L]$ (i.e., $P = [v; e_1, e_2, ..., e_L]$), where L is the length of the path. This implies that the algorithm only needs to evaluate the classifiers L times. Hence, this algorithm clearly has sublinear complexity, since in the best case, $L = \log K$. However, this also implies that errors made at a higher level in the hierarchy cannot be corrected later (Fig. 1(b)-Left).

Inefficient usage of training data. Rather than learning all classifiers jointly, most "tree-based" methods learn one classifier at a time from the top level to the bottom level gradually using fewer images corresponding to a subset of classes. Thus, the classifiers at a lower level are trained using less data than the ones trained at a higher level.

Learning the hierarchy. Most of the "tree-based" methods [11, 2, 14] address the two issues described above by learning the hierarchy of classes to avoid i) making mistakes by the greedy algorithms and/or ii) training classifiers using very few data by limiting the depth of the hierarchy.

Instead of focusing on learning hierarchy, our proposed method addresses these two issues with a given hierarchy while maintaining the efficiency of class predictions.

3.2. Best Path Model

Instead of defining a scoring function at each edge (i.e., $S_v^e(x) = x^T w_v^e$), we define a scoring function $S_P(x)$ for each path $P = [v_1; e_1, ..., e_L]$ as the sum of the edge scores (i.e., $S_P(x) = \sum_{j=1}^{L} S_{v_j}^{e_j}(x)$), where $\{v_1, ..., v_{L+1}\}$ is the sequence of node indices visited in the tree according to the path P. We propose to learn a model such that the class prediction of an input x is the class $\mathcal{K}_{v_{L+1}}$ associated to the leaf node v_{L+1} of the path P^* with the highest score.

$$P^* = \arg\max_{P \in \mathcal{P}^1} S_P(x) , \tag{1}$$

where $\mathcal{P}^1$ is the set of all the paths in the tree starting from the root node $v = 1$ to a leaf node.

Properties. The best path model can be considered as a special case of one-versus-all SVM since the accumulated sum of the classifiers parameters along each path can be treated as the classifiers parameters for a class in the one-versus-all SVM. The important difference is that the tree structure enforces the parameters of two paths to be partially shared according to the overlap of the paths. In other words, our proposed model utilizes the knowledge of the hierarchy to share the parameters between classes so that classes closer in the hierarchy have more similar parameters. As we report in Sec. 6, this property helps improving the accuracy over the one-versus-all SVM. However, the time complexity for a brute force class prediction algorithm (i.e., evaluating all paths) is linear to the number of paths which is similar to complexity of the one-versus-all SVM. In the next section, we propose a novel branch-and-bound-like algorithm to predict the class as efficient as the greedy algorithm.

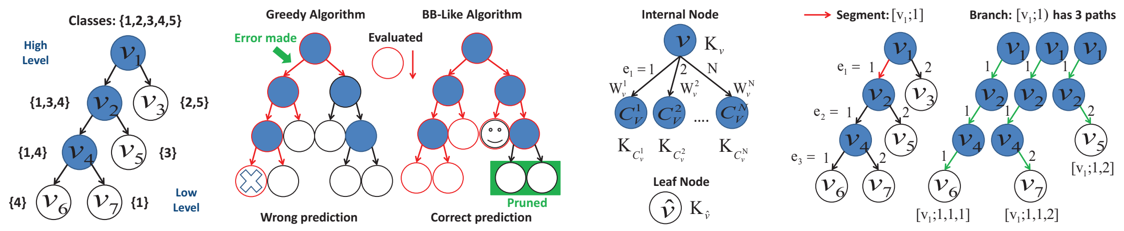
4. Efficient Prediction

Instead of exploring all the paths in the tree, our branch-and-bound-like algorithm explores only a few paths in the tree and typically finishes in "sublinear" time.

4.1. Branch-and-Bound-Like (BB-Like) Search

Although the number of paths is large $|\mathcal{P}^1|$, only a few paths are worth exploring. Hence, the key idea for achieving "sublinear time" is to avoid exploring the whole tree during the search (Fig. 1(b)-Right). The branch-and-bound (BB) framework [19] allows us to do this while guarantee finding the best path if a valid bound is given.

Recall that a path $P = [v; e_1, ..., e_L]$ is defined as a route with increasing levels in the hierarchy from node v to a leaf node following the sequence of selected edges $[e_1, ..., e_L]$. We further define a "branch" $\mathcal{P}^v = [v; e_1, ..., e_F)$ as a set of paths overlapping with the segment $[v; e_1, ..., e_F]$ starting from the node v with length F (not necessarily reaching a leaf node) (Fig. 1(d)). For instance, the path $[v; e_1, ..., e_F, ..., e_L]$ belongs to the branch $[v; e_1, ..., e_F)$.



(a) Tree-shaped Hierarchy (b) Greedy vs. BB-Like (c) Notations (d) Segment and Branch

Figure 1: Panel (a) gives an example of the tree-shaped hierarchy, where the numbers represent object classes. Panel (b) shows an example when the greedy algorithm gives a wrong prediction (node with a "x" symbol) by making an error at a high level. Our BB-Like algorithm gives a correct prediction (node with a smiley face) without evaluating the whole tree (nodes in green plate are pruned). Notice that red nodes and edges denote the ones have been evaluated. Panel (c) illustrates the notation in the tree-based model for internal and leaf nodes. Panel (d) shows the definition of segment (red edges), path (green lines from root to leaf), and branch (a collections of paths).

Algorithm 1 Efficient Branch and Bound Prediction

```
Require: input $x \in \mathbb{R}^D$, tree $\mathcal{T}$, scoring function $S$, and node-wise upper bound $\{U_v\}_v$
Ensure: $P^* = \arg\max_{P \in \mathcal{P}} S_P(x)$
1:  Initialize $Q$ as an empty priority queue.
2:  Set node index $\hat{v} = 1$, branch as $\hat{\mathcal{P}} = \mathcal{P}^1$ and accumulated score $\hat{S}(x) = 0$ (Starting from root node).
3:  repeat
4:      split $\hat{\mathcal{P}}$ to $[\hat{\mathcal{P}}\ 1), ..., [\hat{\mathcal{P}}\ N)$ ($N$ sub-branches).
5:      for e=1:N do
6:          Update accumulated score $\hat{S}(x) += S^e_{\hat{v}}(x)$.
7:          Update upper bound $\hat{U}(x) = \hat{S}(x) + U_{c^e_{\hat{v}}}$.
8:          push $(c^e_{\hat{v}}, [\hat{\mathcal{P}}\ e), \hat{S}(x), \hat{U}(x))$ into $Q$.
9:      end for
10:     retrieve $(\hat{v}, \hat{\mathcal{P}}, \hat{S}(x))$ from branch with max $\hat{U}(x)$ in $Q$.
11: until $|\hat{\mathcal{P}}| == 1$ (a branch consists of one single path).
12: set $P^* = [\hat{\mathcal{P}}]$.
```

For each branch, we calculate the upper bound $\hat{U}(x)$ of the highest score that a path in the branch could take for input x. At the beginning of the search, we start with a branch $\hat{\mathcal{P}}$ corresponding to the set of paths in the whole tree $\mathcal{P}^1 = [1; \ )$ (See line 2 in Algorithm 1). Then, we continuously split the working branch $\hat{\mathcal{P}}$ to N sub-branches $\{[\hat{\mathcal{P}}\ e)\}_e$ [1] (See line 4 in Algorithm 1), and update the upper bound $\hat{U}(x)$ (See line 7 in Algorithm 1). The BB algorithm organizes the search over candidate branches in a best-first manner. Therefore, the working branch always corresponds to the one with the highest upper bound (See line 10 in Algorithm 1). The search terminates when it has identified a branch consisting of one single path (i.e., $|\hat{\mathcal{P}}| == 1$) with a score that is at least as good as the upper bound of all remaining candidate branches (See line 11 in Algorithm 1). The best-first manner guarantees that the best path has been found. The efficiency of the algorithm usually strongly depends on the tightness of the bound. In our experiment, our efficient prediction algorithm is typically much faster than the worst case complexity.

[1] $[[v; e1, ..., e_F)\ e) = [v; e1, ..., e_F, e)$. [ ) denotes concatenation.

4.2. Bound Calculation

We assume that each node v caches the upper bound U_v of paths in branch $\mathcal{P}^v$ starting from node v to a leaf node can take. Hence, the upper bound $\hat{U}(x)$ of the branch $[1; e_1, ..., e_F)$ equals to the sum of score $\hat{S}(x)$ accumulated from the segment $[1; e_1, ..., e_F]$ and the upper bound U_v cached at the last node of the segment $[1; e_1, ..., e_F]$ (See line 7 in Algorithm 1).

The tightest possible upper bound cached per node is input dependent $U_v(x)$ and can be obtained by evaluating all the classifiers in the tree with the given input x. However, obtaining the bound is as costly as the brute-forced search algorithm to find the best path. Moreover, it needs to be done once for every input. Therefore, we propose an input independent upper bound U_v estimated from the training inputs X defined as,

$$U_v = \max_{x \in X} U_v(x) \ . \tag{2}$$

The upper bounds only need to be estimated once after the model is trained. Notice that U_v is not guaranteed to be a valid bound since U_v is possible to be smaller than the upper bound $U_v(\hat{x})$ of an unseen testing input $\hat{x}$. However, we found that the estimation is very reliable in practice since the classification accuracy is typically not sacrificed (Fig. 2). Therefore, we call it a branch-and-bound-like algorithm.

Properties. Although the BB algorithm terminates when the best path is found, much more information is kept in the queue Q (in Algorithm 1) which ranks the evaluated branches. We show that the rank naturally maintains a diverse set of predictions consisting of both branches sharing similar path segments (e.g., cats and dogs) and branches with very different path segments (e.g., tools and animals) which can be potentially used to facilitate human-computer interaction (Fig. 8). For example, a user would rather see a diverse set of predictions and select the most suitable one, than to see a similar set of predictions where all predictions might not be suitable. In contrast, it is unclear how the greedy tree-based methods can rank hypotheses in a principled way.

In the next section, we describe how to learn the model parameters using a Structured SVM formulation with the option to jointly learn the node-wise upper bound U_v as well.

5. Structured SVM Learning

The scoring function can be learned using a Structured SVM (SSVM) formulation where the structured output P encodes the path in the tree-shaped hierarchy. Considering that we have a set of training inputs and the ground truth path in the tree $\{x^m, P^m\}_{m=1\sim M}$, we solve the following SSVM problem,

$$\begin{aligned} \min_{\mathbf{W},\xi^m} \quad & \frac{1}{2}\mathbf{W}^T\mathbf{W} + \lambda \sum_m \xi^m \\ \text{s.t.} \quad & S_{P^m}(x^m;\mathbf{W}) - S_P(x^m;\mathbf{W}) \\ & \geq \Delta(P;P^m) - \xi^m, \forall m, \forall P \in \mathcal{P}\,, \end{aligned} \tag{3}$$

where $\mathbf{W} = \{W_v\}_{v\in\mathcal{V}}$ is a vector concatenation of edge-wise model parameters, $\Delta(P;P^m)$ is a loss function measuring incorrectness of the estimated path P, and λ controls the relative weight of the sum of the violation term ($\sum_m \xi^m$) with respect to the regularization term ($\mathbf{W}^T\mathbf{W}$). Notice that, in our implementation, a simple zero-one loss is used (i.e., $\Delta(P;P^m) = 0$ when $P \neq P^m$ and $\Delta(P;P^m) = 1$ otherwise). We solve the problem using dual coordinate-descent solver [34] to achieve fast convergence and less memory usage, where only hard examples corresponding to $P^*(x^m) = \arg\max_{P\in\mathcal{P}} S_P(x^m) + \Delta(P;P^m)$ are added to actively enforce the constraints.

Learning Bounds. Instead of learning the model parameters then estimating the node-wise upper bound $\mathbf{U} = \{U_v\}_v$, we can extend the SSVM problem in Eq. 3 to jointly learn model parameters $\mathbf{W}$ and node-wise upper bounds $\mathbf{U}$:

$$\begin{aligned} \min_{\mathbf{W},\xi^m\geq 0,\mathbf{U}} \quad & \mathbf{W}^T\mathbf{W} + \lambda_1 \sum_m \xi^m + \lambda_2 \sum_v U_v \\ \text{s.t.} \quad & S_{P^m}(x^m;\mathbf{W}) - S_P(x^m;\mathbf{W}) \\ & \geq \Delta(P;P^m) - \xi^m, \forall m, \forall P \in \mathcal{P}^1 \\ \text{i) Valid bound} \quad & S_{\hat{P}}(x^m;\mathbf{W}) \leq U_v, \forall \hat{P} \in \mathcal{P}^v, \forall v \in \mathcal{V} \\ \text{ii) Reduction rate} \quad & U_v \geq \gamma U_{c_v^e}, \forall v \in \mathcal{V}, \forall e \in 1 \sim N\,, \end{aligned} \tag{4}$$

where γ is the rate of reduction of the bound U_v. Notice that two types of additional constraints are ensured.

i) **Valid bound.** The scores $S_{\hat{P}}(x^m;\mathbf{W}); \hat{P} \in \mathcal{P}^v$ of paths starting from node v must be lower than the node-wise upper bound U_v.

ii) **Reduction rate.** The upper bound $U_{c_v^e}$ of a child node should be smaller than the bound U_v of a parent node by a factor of γ.

Similar to solving Eq. 3, we only add violated constraints in i) ii) and solve Eq. 4 using dual coordinate-descent solver [34]. In Fig. 3, we show that by changing the γ value, we explore the trade-off between the accuracy and efficiency.

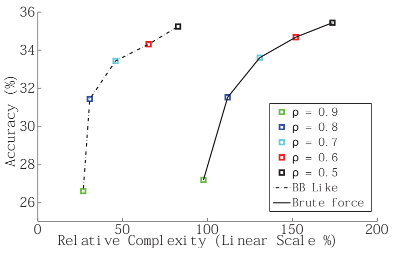


Figure 2: Effectiveness of the branch-and-bound-like algorithm on Caltech-256. Notice that the color-coded nodes denote models with different degree of relaxation (ρ) trained without bound reduction rate constraints.

During training, the BB-Like algorithm (Sec. 4.1) can also be used to speed up the training process by efficiently finding hard examples. However, at the beginning iteration of solving Eq. 4, the estimated upper bound is typically smaller than the real one since there are fewer active set of constraints. Therefore, we increase the upper bound by 10% while applying our BB-Like algorithm in order to find more hard examples.

6. Experiments

We evaluate our method on three publicly available datasets (Caltech-256 [15], SUN-397 [32], and ImageNet 1K [9]). Our goals are to i) verify that our proposed method is more accurate than the one-versus-all methods; ii) achieve a better trade-off between accuracy and efficiency compared to the state-of-the-art "tree-based" methods [14]; iii) show typical examples of ranked hypotheses which maintains a diverse set of class predictions (Fig. 8 and technical report [27]).

6.1. Basic Setup

For accuracy, we report mean of per-class accuracy as [14]. For testing efficiency, since each node in the tree model is a linear classifier with equal complexity, the overall testing efficiency depends on the number of classifier's

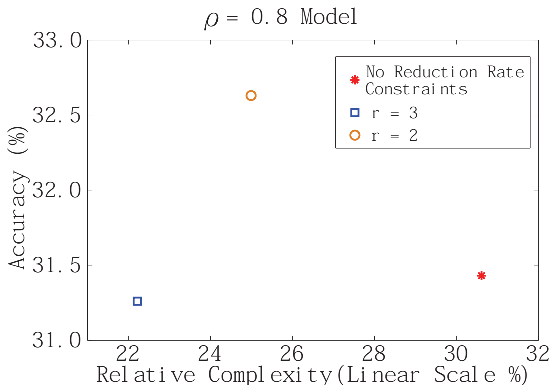


Figure 3: The trade-off between relative complexity (Sec. 6.1 for definition) (x-axis) and accuracy (y-axis) by learning bounds using $\gamma = \{1\,(\text{no reduction rate constraint}), 2, 3\}$ and $\rho = 0.8$ on Caltech-256.

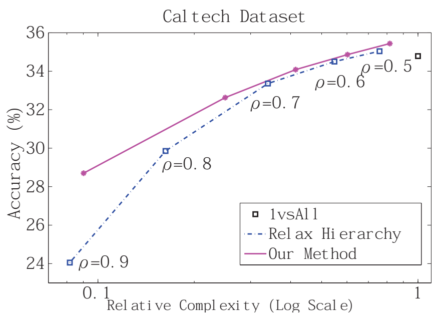


Figure 4: Trade-off between accuracy (y-axis) and relative complexity (Sec. 6.1 for definition) (x-axis) calculated for our method (pink-solid), relax hierarchy [14] (blue-dash), and one-versus-all SVM (black-dot) on Caltech-256.

evaluations. We calculate the mean of per-instance number of classifier's evaluations and report the "relative complexity" which is the mean normalized by the number of classifier's evaluations of the one-versus-all method.

We compare our method to the state-of-the-art "tree-based" method [14] and one-versus-all SVM. All methods use the same linear SVM implementation [13] and Locality-constrained Linear Coding (LLC) feature [31]. The value of parameter λ (Eq. 4) for each method is selected by four-fold cross validation on the training set. For training the tree-shaped hierarchy, we use the released code from [14] and allow the tree to be at most 8 levels. Given the tree-shaped hierarchies, we train our models using SSVM (Sec. 5) to explore a better trade-off. SSVM training in Matlab takes about 10 hours for Caltech and Sun dataset, and about 6 days for ImageNet dataset.

6.2. Caltech-256

Caltech-256 [15] contains 256 object categories. For each category, we randomly sampled 40 images as training data and 40 images as testing data similar to [14]. LLC feature with $21K$ dimension is used in order to recreate the one-versus-all accuracy reported in [14]. The relaxed hierarchy method [14] is a very strong baseline since it outperforms many other "tree-based" method [24, 16]. Therefore, five hierarchies corresponding to relaxed degree $\rho = \{0.5, 0.6, 0.7, 0.8, 0.9\}$ (the bigger value the less relax) are trained for comparison.

Detail Analysis. We first verify the branch-and-bound-like algorithm improves the efficiency without sacrificing much

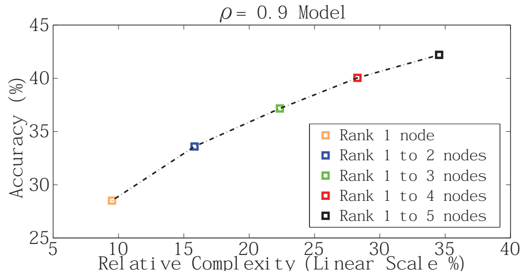


Figure 5: Classification accuracy vs. relative complexity of rank 1 to rank 5 predictions using $\rho = 0.9$ model on Caltech-256.

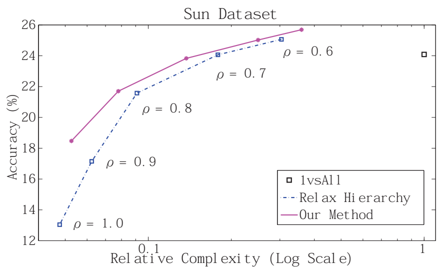


Figure 6: Trade-off between accuracy (y-axis) and relative complexity (x-axis) calculated for our method (pink-solid), relax hierarchy [14] (blue-dash), and one-versus-all SVM (black-dot) on Sun-397.

the accuracy (Fig. 2), given models trained without using bound constraints (Eq. 3). However, without forcing the bounds to reduce at a certain rate in the hierarchy, the efficiency gain is limited. In Fig. 3, we demonstrate that different trade-off between accuracy and efficiency can be explored by training our models with constraints corresponding to $\gamma = \{1, 2, 3\}$ and $\rho = 0.8$ (Eq. 4). We further explore different combination of ρ (degree of relaxation) and γ (bound tightness) and select the models corresponding to the optimal trade-off between efficiency and accuracy on the validation set [2]. The efficiency-versus-accuracy performance on the testing set is shown in Fig. 4. Notice that our method obtains more improvement on accuracy at lower complexity. In particular, we obtain a significant 4.65% (relative 24.82%) improvement at $\sim 9\%$ relative complexity. We also verify that our best accuracy (35.44%) outperforms the one-versus-all method (34.78%) (Fig. 4). Finally, Fig. 5 demonstrates that the accuracy from rank one to rank five predictions increases 10% while the complexity increases sublinearly.

6.3. Sun-397

We evaluate the scene classification task on SUN dataset [32]. Similar to the setting of [32], we use 397 well-sampled categories to evaluate our method. For each category, we randomly sample 50 images as training data and 50 images as testing data. LLC feature with $16K$ dimension is used in order to recreate the one-versus-all accuracy reported in [14]. Five hierarchies corresponding to relaxed degree $\rho = \{0.6, 0.7, 0.8, 0.9, 1.0\}$ are trained for comparison.

Our method achieves better accuracy (25.69%) than the one-versus-all SVM (24.08%). and a better trade-off between efficiency and accuracy than [14] as shown in Fig. 6. Similar to Caltech 256, our method obtains more improvement on accuracy at lower complexity. In particular, we obtain a significant 5.43% (relative 41.64%) improvement at $\sim 5\%$ relative complexity.

6.4. ImageNet

The dataset contains 1K object categories and 1.2 million images. We use the same training and testing split

[2]One fourth of the training images are used for model selection.

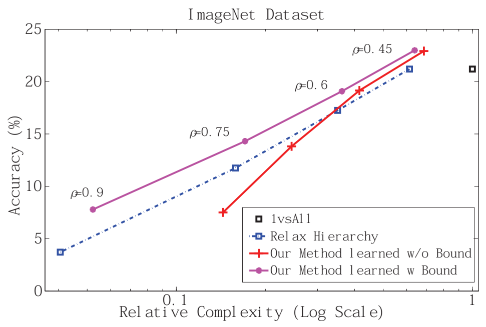


Figure 7: Trade-off between accuracy (y-axis) and relative complexity (x-axis) calculated for our method learned with bound constraints (pink-solid), without bound constraints (red-solid), relax hierarchy [14] (blue-dash), and one-versus-all SVM (black-dot) on ImageNet.

as in [17]. LLC feature with $10K$ dimension is used in order to recreate the baseline one-versus-all accuracy reported in [8]. Four hierarchies corresponding to relaxed degree $\rho = \{0.9, 0.75, 0.6, 0.45\}$ are trained for comparison. Given the tree-shaped hierarchies, we train our models using SSVM (Sec. 5). Our method achieves better accuracy (22.99%) than the one-versus-all SVM (21.2%), and a better trade-off between efficiency and accuracy than [14] as shown in Fig. 7. In particular, we obtain a significant 4.07% (relative 109.79%) improvement at $\sim 5\%$ relative complexity. Moreover, models learned with bound constraints (pink-solid) outperform models learned without bound constraints (red-solid). We also apply our method on hierarchy learned by [2] and achieve similar improvement (See technical report [27]).

7. Conclusion

We propose an efficient and accurate classifier for image hierarchies which achieves a better trade-off between efficiency and accuracy. Our contributions are: i) a novel BB-Like algorithm that utilizes the tree-hierarchy for efficient prediction; and ii) an extended SSVM formulation that allows us to search for a better trade-off between accuracy and efficiency. On Caltech-256 [15], SUN dataset [32], and ImageNet 1K [9], our method outperforms the one-versus-all method in accuracy and achieves a better trade-off compared to the state-of-the-art "tree-based" method [14]. Most importantly, we achieve a significant 4.65%, 5.43%, and 4.07% (relative 24.82%, 41.64%, and 109.79%) improvement, respectively, in accuracy at small values of relative complexity compared to [14]. Finally, our BB-Like algorithm naturally maintains a diverse set of class predictions without significantly increasing the complexity. In the future, we would like to investigate potential human-computer interaction applications which utilize the rich output (e.g., the rank of class predictions) from our method.

acknowledgements

We acknowledge the support of NSF CAREER grant (#1054127), ARO grant (W911NF-09-1-0310), and ONR grant (N00014-13-1-0761).

References

[1] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *J. Mach. Learn. Res.*, 1:113–141, 2001. 2

[2] S. Bengio, J. Weston, and D. Grangier. Label embedding trees for large multi-class tasks. In *NIPS*, 2010. 1, 2, 3, 7

[3] A. Beygelzimer, J. Langford, Y. Lifshits, G. Sorkin, and A. Strehl. Conditional probability tree estimation analysis and algorithms. In *UAI*, 2009. 1, 2

[4] A. Beygelzimer, J. Langford, and P. Ravikumar. Error-correcting tournaments. In *ALT*, 2009. 1, 2

[5] A. Binder, M. Kawanabe, and U. Brefeld. Efficient classification of images with taxonomies. In *ACCV*, 2009. 2

[6] Y. Chen, M. Crawford, and J. Ghosh. Integrating support vector machines in a hierarchical output space decomposition framework. In *IGARSS*, 2004. 1, 2

[7] A. Criminisi and J. Shotton. Decision forests for computer vision and medical image analysis. *Springer*, 2013. 2

[8] J. Deng, A. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10,000 image categories tell us? In *ECCV*, 2010. 7

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 1, 5, 7

[10] J. Deng, J. Krause, A. Berg, and L. Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *CVPR*, 2012. 2

[11] J. Deng, S. Satheesh, A. Berg, and L. Fei-Fei. Fast and balanced: Efficient label tree learning for large scale object recognition. In *NIPS*, 2011. 1, 2, 3

[12] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. Artif. Int. Res.*, 2(1):263–286, 1995. 2

[13] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008. 6

[14] T. Gao and D. Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *ICCV*, 2011. 1, 2, 3, 5, 6, 7

[15] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical report, Cal. Tech., 2007. 1, 5, 6, 7

[16] G. Griffin and P. Perona. Learning and using taxonomies for fast visual categorization. In *CVPR*, 2008. 1, 2, 6

[17] http://www.image net.org/challenges/LSVRC/2010/. 7

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2

[19] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 1960. 3

[20] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML*, pages 609–616, 2009. 2

[21] L. Li. Multiclass boosting with repartitioning. In *ICML*, 2006. 2

[22] Y. Lin, F. Lv, S. Zhu, M. Yang, T. Cour, K. Yu, L. Cao, and T. Huang. Large-scale image classification: Fast feature extraction and svm training. In *CVPR*, 2011. 2

[23] S. Liu, H. Yi, L.-T. Chia, and D. Rajan. Adaptive hierarchical multi-class SVM classifier for texture-based image classification. In *ICME*, 2005. 1, 2

[24] M. Marszalek and C. Schmid. Constructing category hierarchies for visual recognition. In *ECCV*, 2008. 6

[25] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV*, 2012. 2

[26] R. E. Schapire. Using output codes to boost multiclass learning problems. In *ICML*, 1997. 2

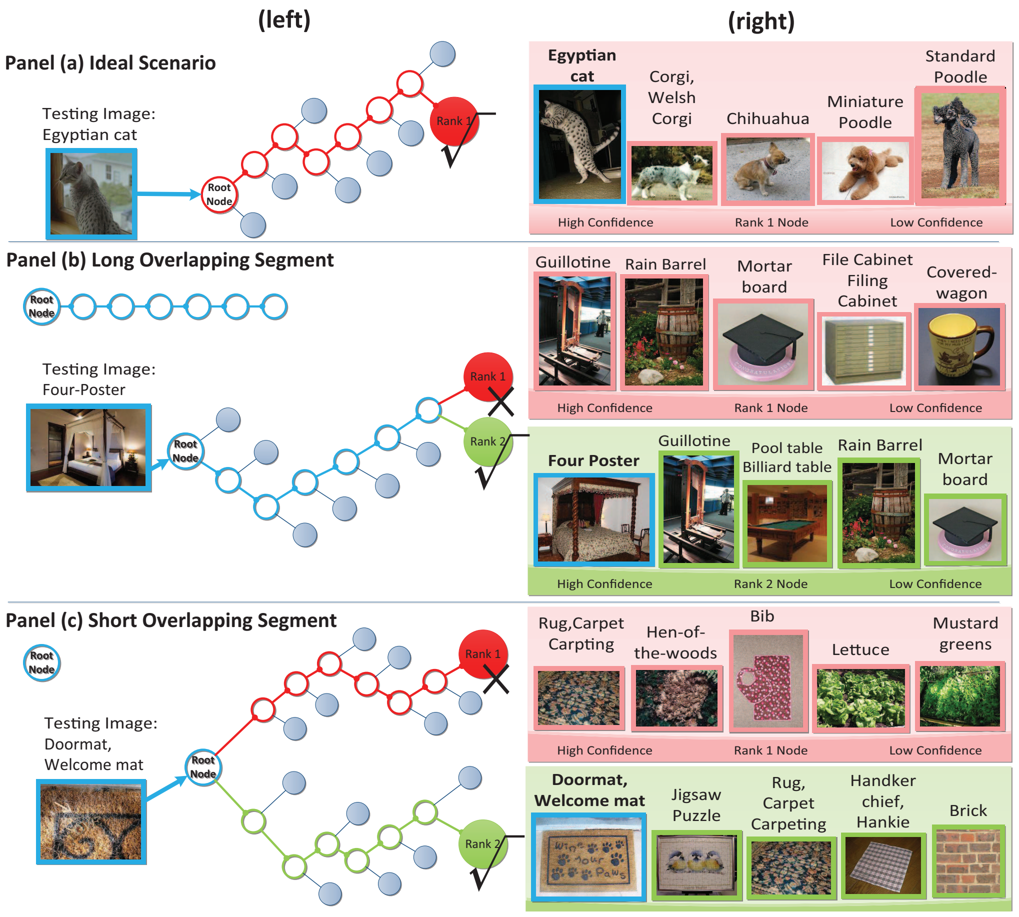


Figure 8: Typical examples of our BB-Like search results on ImageNet. In each panel, we show the testing image and the rank one or two paths on the left. Notice that we draw the tree in a horizontal direction and do not show all the evaluated nodes as shown in Fig. 1(b)-Right for the purpose of a compact visualization. On the right, we order the classes from left to right according to the prediction confidence in each leaf node since a leaf node contains multiple classes in the relaxed hierarchy. Panel (a) shows the ideal case when the rank one path (red) reaches the leaf node containing the correct class (Egyptian cat) and the class prediction in the leaf node is correct. Panel (b,c) demonstrate that our BB-Like algorithm naturally ranks the paths to keep a diverse set of class predictions. Notice that prediction diversity is essential for human-computer-interaction so that an user have the freedom to process the predictions. Panel (b) gives an example that the rank one (red) and the rank two (green) paths are nearby in the hierarchy (long overlapping segment). In this case, the rank two (green) path reaches the leaf node which successfully predicts the correct class (Four Poster). Panel (c) gives an example that the rank one (red) and the rank two (green) paths are far from each other in the hierarchy (short overlapping segment). In this case, the rank two (green) path reaches the leaf node which successfully predicts the correct class (Doormat).

[27] M. Sun. Technical report of find the best path. `homes.cs.washington.edu/~sunmin/`. 5, 7

[28] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, 2004. 2

[29] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector learning for interdependent and structured output spaces. In *ICML*, 2004. 1

[30] V. Vural and J. G. Dy. A hierarchical method for multi-class support vector machines. In *ICML*, 2004. 1, 2

[31] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 2, 6

[32] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 1, 5, 6, 7

[33] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2010. 2

[34] Y. Yang and D. Ramanan. Articulated pose estimation using flexible mixtures of parts. In *CVPR*, 2011. 5

[35] X. Yuan, W. Lai, T. Mei, X. Hua, X. Wu, and S. Li. Automatic video genre categorization using hierarchical SVM. In *ICIP*, 2006. 1, 2

[36] X. Zhou, K. Yu, T. Zhang, and T. S. Huang. Image classification using super-vector coding of local image descriptors. In *ECCV*, 2010. 2

[37] A. Zweig and D. Weinshall. Exploiting object hierarchy: Combining models from different category levels. In *ICCV*, 2007. 2