Semantic driven Hierarchical Learning for Energy-Efficient Image Classification

Priyadarshini Panda, and Kaushik Roy
School of Electrical and Computer Engineering, Purdue University
Email:[pandap, kaushik]@purdue.edu

Abstract*—Machine-learning algorithms have shown outstanding image recognition performance for computer vision applications. While these algorithms are modeled to mimic brain-like cognitive abilities, they lack the remarkable energy-efficient processing capability of the brain. Recent studies in neuroscience reveal that the brain resolves the competition among multiple visual stimuli presented simultaneously with several mechanisms of visual attention that are key to the brain's ability to perform cognition efficiently. One such mechanism known as saliency based selective attention simplifies complex visual tasks into characteristic features and then selectively activates particular areas of the brain based on the feature (or semantic) information in the input. Interestingly, we note that there is a significant similarity among underlying characteristic semantics (like color or texture) of images across multiple objects in real world applications. This presents us with an opportunity to decompose a large classification problem into simpler tasks based on semantic or feature similarity. In this paper, we propose semantic driven hierarchical learning to construct a tree-based classifier inspired by the biological visual attention mechanism for optimizing energy-efficiency of machine-learning classifiers. We exploit the inherent feature similarity across images to identify the input variability and use recursive optimization procedure, to determine data partitioning at each tree node, thereby, learning the feature hierarchy. A set of binary classifiers is organized on top of the learnt hierarchy to minimize the overall test-time complexity. The feature based-learning allows selective activation of only those branches and nodes of the classification tree that are relevant to the input while keeping the remaining nodes idle. The proposed framework has been evaluated on Caltech-256 dataset and achieves ∼3.7x reduction in test complexity for 1.2% accuracy improvement over state-of-the-art one-vs-all tree-based method, and even higher improvements in test-time (of ∼5.5x) when some loss in output accuracy (up to 2.5%) is acceptable.

***Index Terms*—Efficient Classification, Feature Similarity, Feature Hierachy, SVM Tree, Selective Activation.**

I. Introduction

With the massive growth of digital image data due to social media, surveillance camera, among others, there is a growing demand for computing platforms to perform cognitive tasks. Most of these computing platforms have limited resources in terms of processing power and battery life. Hence, researchers have been strongly motivated to design efficient large-scale image recognition methods to enable resource constrained IoT (Internet of Things) devices with cognitive intelligence [1], [2]. Several machine-learning models including Support Vector Machines (SVM) [3], random forest [4], and Adaboost [5] have proven to be very successful for image classification. However, these classifiers do not scale well with increasing number of image categories. Deep Learning Networks like ConvNets [6] have achieved state-of-the-art accuracies, even surpassing human performance for recognition applications [7]. However, they have been criticized for their enormous training cost and computational complexity. Similarly, the one-versus-all linear SVM, one of the most popular classifiers for large-scale classification, is computationally inefficient as its complexity increases linearly with the number of categories. While these classifiers are modeled to mimic the brain-like cognitive abilities, they lack the remarkable energy-efficient processing capability of the brain. Seeking to attain the brain's efficiency, we draw inspiration from its underlying processing mechanisms to design a multi-class classification method that is both accurate and computationally efficient.

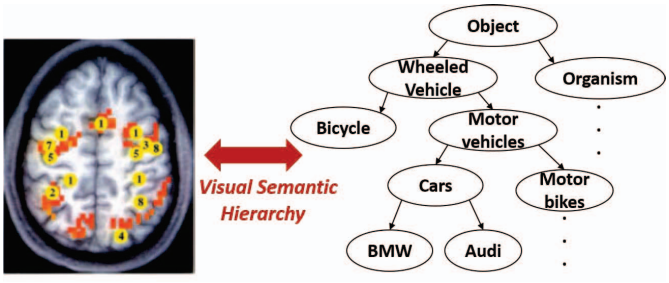

Fig. 1. (Left) Selective attention mechanism observed in the frontal and parietal cortex involved in the generation and control of salient attentional signals [8] (Right) Toy example of hierarchical tree formed on the basis of different semantic categories and inter-class relationships of object types.

One such mechanism known as "saliency based selective attention" shown in Fig. 1 (left) simplifies complex visual tasks into characteristic features and then selectively activates particular areas of the brain based on the feature information in the input [9]. When presented with new visual images, the brain associates the already learnt features to the visual appearance of the new object types to perform recognition [8]. This facilitates the brain to learn a host of new information with limited capacity and also speeds up the recognition process. Interestingly, we note that there is significant similarity among underlying characteristic features (like color or texture) of images across multiple objects in real world applications. This presents us with an opportunity to build an efficient visual recognition system incorporating inter-class feature similarities and relationships.

In this work, we propose a computationally efficient classification method that exploits the feature similarity among multiple classes in the dataset to build a hierarchical tree structure composed of binary classifiers (or SVMs). We use a variant of the boosted tree algorithm [10] that combines Adaboost with a SVM based decision tree. The resultant structure learns a hierarchy of features that transition from general to specific as we go deeper into the tree in a top-down manner. This is similar to the state-of-the-art Deep Learning convolutional Networks (DLNs) where the convolutional layers exhibit a generic-to-specific transition in the learnt features [11]. In case of DLNs, the entire network is utilized for the recognition of a particular test input. In contrast, the construction of the feature hierarchy incorporates effective and active pruning of the dataset during training of the individual tree nodes resulting in an efficient instance-specific classification path. In addition, as we will see in later sections, our model captures both inter and intra class feature similarity to build a tree hierarchy with decision paths of varying lengths even for the same class. This provides substantial benefits in test speed and

978-3-9815370-8-6/17/$31.00 ©2017 IEEE

computational efficiency for large-scale problems while maintaining competitive classification accuracy.

Fig. 1 (right) shows a toy example of our proposed tree based on real-world broad semantic categories for different object classes. For example, to recognise a car, it is not sensible to learn all the specific appearance details. Instead, first we learn the general vehicle-type features (wheels, shape etc) and then learn more discriminative details (brand symbol). Thus, we learn a hierarchy of features generalizing over object instances like: Wheeled vehicle→Motor vehicles→Cars→BMW. If presented with new motorbike object types, the hierarchy now associates this new category of objects to the already learnt "Wheeled vehicle" features and then learns more discriminative details corresponding to the motorbike types. Each node of the tree is then associated with different features based on inter-class relationships. It is evident from Fig. 1 that the proposed tree method bears resemblance to the selective attention mechanism of the brain (Fig. 1 left) by exploiting feature similarity and the implicit relationships among different visual data to learn a meaningful hierarchy for recognition. Please note, the words 'semantic' and 'feature' are used interchangeably in the remainder of the paper.

II. Related Work

In recent years, there has been substantial work that propose different ways to construct a hierarchical classification tree [12]–[14]. However, most of these methods rely on a greedy prediction algorithm for class prediction through a single path of the tree. While these algorithms achieve sublinear complexity, the accuracy is typically sacrificed as errors made at higher nodes of the hierarchy cannot be corrected later. Researchers have also looked at developing efficient and effective feature representations for large-scale classification problems [15]–[17]. [6], [7] learn discriminative features using deep convolutional networks to achieve state-of-the-art accuracy. Please note that our proposed tree is orthogonal to such models since our method can use various feature respresentations to explore the accuracy vs. efficiency tradeoff. Hence, we do not optimize over different features in this work, rather compare the efficiency benefits of our approach with existing hierarchical methods.

While our proposed model draws inspiration from other tree-based methods such as [10], [14], [18], we have a different focus, design and evaluation strategy. As mentioned, most of these methods use a greedy prediction algorithm to achieve a good tradeoff between output quality (accuracy) and complexity. The novelty of our work is that we use the recursive Adaboost training [5] as a unified and principled optimization procedure to determine data partitioning (or learning feature hierarchy) based on feature similarity. This in turn enables the binary SVM to construct a maximum-margin hyperplane for optimal decision boundary modelling (with lower generalization error) leading to better performance. In addition, organizing the binary classifiers in a hierarchical tree structure on top of the feature hierarchy further reduces complexity.

III. Adaboost: Key Principles and Limitations

The Adaboost algorithm combines a set of simple or weak classifiers ($h_t(x)$) [19] to form the final classifier ($H(x)$) given by $H(x) = \sum_{t=1}^{T} \alpha_t * h_t(x)$, where α_t is the assigned weight for each weak classifier. The output of the final or strong classifier is $f(x) = sign(H(x))$. The weak classifiers can be thought of as feature or basis vectors. Given a set of training samples, Adaboost maintains a probability distribution, W (uniform in the first iteration), of the samples. Then, Adaboost calls a WeakLearn algorithm [19] that trains the weak learner or classifier (h_t) on the weighted sample in a series of iterations ($t = 1, 2, ..T$). The distribution W is updated in each iteration to minimize the overall error (ϵ). Finally, Adaboost uses a weighted linear combination of the weak learners (or features) to obtain the final output f. The Adaboost and WeakLearn algorithm have been explained in detail in [19].

One of the key principles of Adaboost is that "easy" samples that are correctly classified by the weak classifiers get low weights while those misclassified ("hard" samples) get higher weights. The weight distribution W captures all the information about selected "features" in a given iteration. However, due to the weight update rule and normalization of W in each iteration, the information about previously selected features might be lost. This will result in misclassification of correctly classified ("easy") samples from earlier iterations in the present epoch. Thus, the algorithm does not maintain a generic-to-specific transition while learning the weak classifiers (or features) that proves to be ineffective after a few iterations. To address this, we build a tree of strong classifiers, instead of constructing a single strong classifier from a linear combination of weak learners. The tree utilizes the features learnt from the previous nodes to construct the subsequent nodes. As we traverse down the tree, the classifiers learn more specific features that are useful for classifying the "hard" inputs correctly while preserving the feature information learnt at the early nodes for "easy" input samples.

IV. Proposed Feature based Hierarchical Tree: Learning and Implementation

A. Training for a 2-class problem

Now, we discuss the procedure for training the tree for a two-class problem. A tree is recursively trained to learn the feature transition hierarchy and determine the data partitioning at each node as shown in Algorithm 1. It learns and preserves a hierarchy of features essential for understanding the underlying image representations and for efficient classification. At each node, a classifier is learnt using the Adaboost algorithm described in [19] that identifies the optimal feature (available from the training dataset) to separate the training inputs at a particular node into the corresponding sub-branches. It is shown in [20] that Adaboost is essentially approximating a logistic regression. For convenience in notation, we denote the output computed by each classifier at the tree node as

$$p(+1|x) = \frac{1}{1 + exp(-H(x))} \quad (1)$$

$$p(-1|x) = \frac{1}{1 + exp(H(x))} \quad (2)$$

Depending upon the probabilities computed by the classifier node, the training set (D) is divided as D_{left} and D_{right} that are then passed to the sub-branches for training the following nodes of the tree. As the tree expands, only a subset of the input samples are passed to the subsequent nodes. Thus, the final nodes or leaves of the tree will consist of input samples belonging to one particular class. Please refer to Fig. 2 for an overview of the tree structure and input sub-sampling obtained with the learning model. Later, in section IV(D), we give a detailed explanation about the input sub-sampling and the hierarchical feature learning achieved with our model.

After learning the feature hierarchy, a binary SVM (with any suitable kernel) is trained at each node of the tree using the D_{right} and D_{left} training sub-samples obtained from *Algorithm 1*. The training labels (+ for D_{right}, - for D_{left} and * for instances that are passed to both D_{left}/D_{right}) are assigned to each input x_i in the corresponding subsets for training the binary SVM. As the training set size decreases (owing to the input partitioning) at the

Algorithm 1 Learning feature hierarchy

Input: Training dataset D=$\{(x_1, y_1, w_1), \ldots., (x_n, y_n, w_n)\}$; $y_i \in \{+1, -1\}$, $\Sigma_i w_i = 1$

Output: Tree with feature hierarchy of depth L

1: **initialize** $tree_{depth}$=1
2: **while** ($tree_{depth} \leq L$)
3: Using Adaboost, Train a strong classifier on D combining T weak classifiers. EXIT Adaboost if $\epsilon_t > \gamma$ (user-defined, γ=0.48 in our experiments).
4: **for** $i = 1 : n$ *//n=# of samples*
5: Compute $p(+1|x_i)$ and $p(-1|x_i)$ using Eqn 1 and 2 for the strong classifier learnt in Step 3.
6: **if** $(p(+1|x_i) > \Delta)$ **then** $D_{right} = (x_i, y_i, 1)$, assign $y_i = \{+\}$
7: **elseif** $(p(-1|x_i) > \Delta)$ **then** $D_{left} = (x_i, y_i, 1)$, assign $y_i = \{-\}$
8: **else** $D_{right} = (x_i, y_i, p(+1|x_i))$ and $D_{left} = (x_i, y_i, p(-1|x_i))$, assign $y_i = \{*\}$
9: **end if**
10: **end for**
11: $tree_{depth}++$
12: Normalize weights in D_{left} subset and goto Step 2.
13: Normalize weights in D_{right} subset and goto Step 2.
//Recursively repeat until $tree_{depth}$ is reached
14: **end while**
15: SVM optimization on the learnt feature hierarchy using D_{left} and D_{right} at a given node ignoring all samples with $y_i = \{*\}$ with standard regularized hinge loss minimization [21].

successive nodes as we traverse down the tree, the complexity of the problem and hence that of the SVM also reduces. This in turn enables better decision boundary modelling with low computational complexity in the subsequent nodes (SVMs) for improved classification performance. Adding SVMs at the nodes on top of the learnt feature hierarchy (*Algorithm 1*) enables the tree model to achieve state-of-the-art accuracies on challenging benchmark databases with significantly lower cost.

The threshold value, Δ in *Algorithm 1*, determines the fraction of training samples separated as positive (+) and negative (-) subsets. If Δ=1, then all training samples are passed to both branches (or sub-trees) of a tree node. The weights for both sub-trees are re-computed based on the node classifier's output. In that case, the tree based Adaboost training converges to a standard boosting algorithm wherein the feature hierarchy (general-to-specific) is not learnt. For all our experiments discussed in section V, we set the Δ value to be > 0.5. For $\Delta < 0.5$, easy inputs that can be correctly classified with general features at the top nodes will be unnecessarily passed down to bottom nodes for classification. This will result in computational inefficiency, defeating the purpose of the propsed model. If $\Delta = 0.5$, then, each training sample is either passed to the right or left sub-tree which leads to a *constrained* partition. In this case, the hard or confusing classes will be assigned to one of the sub-trees causing overfitting of data in the subsequent nodes. This will lead to a decline in accuracy. However, the test complexity will be low since the depth of the tree will be short leading to a quicker decision at the cost of degraded performance.

Those samples whose output probability lies in the range $[1-\Delta, \Delta]$ when $0.5 < \Delta < 1$ can be considered as hard or confusing ones. For $0.5 < \Delta < 1$, the hard samples are passed to both the left and the right sub-trees for training (*). The hard or confusing inputs/classes are ignored while training the SVM at the corresponding node. This is adopted from the *relaxed hierarchy* structure in [14], [22]. This is done to enhance the accuracy of our model. It is understood that the decision boundary becomes progressively non-linear to model the hard or confusing classes in a dataset as we traverse down the tree. The hard or confusing instances are ignored and passed to the bottom nodes that construct better decision boundary models, thereby, decreasing the overall error. In case the hard classes are not passed to bottom nodes, the SVMs at the top will construct overfitted models for the complex data instances, thereby, decreasing the accuracy considerably. In section V, we vary the threshold Δ to build *constrained* and *relaxed* hierarchical tree models and analyze the tradeoff between computational efficiency and accuracy for both approaches.

B. Training for multi-class problem

To conserve the feature transition in the tree, we use a simple method for extending the two-class training model into a multi-class one. We use the minimum entropy measure to select a feature that can be used to categorize the multiple category of objects into two broad classes. The entropy calculation is done as follows:

- For each feature f_j at value v_j (given in the training dataset with multiple labels $y_i \in \{1, ..., n\}$), compute histogram $Hist_{left}(k) = \Sigma_i \delta(k = y_i) w_i$ for $y_i < v_j$ and $Hist_{right}(k) = \Sigma_i \delta(k = y_i) w_i$ for $y_i \geq v_j$.
- Find optimal f_j and v_j that have minimum entropy $Entropy(Hist_{left}) + Entropy(Hist_{right})$.
- **if** $Hist_{left}(y_i) \geq Hist_{right}(y_i)$ **then** assign ${y_i}' = \{-1\}$ **else** assign ${y_i}' = \{+1\}$. *Now the multi-class is reduced to a 2-class problem*

The optimal feature (available from the training dataset) is selected from the entropy calculation across multiple classes that separates the input patterns into 2-classes and then the 2-class training procedure (*Algorithm 1*) is used to learn the subsequent classifier nodes of the tree. In our experiments, we observed that the feature chosen for transforming the multi-class to 2-class problem is often the feature selected by Algorithm 1 to construct the top node of the tree. Intuitively, after the first selection, the features selected at the subsequent nodes help in making a stronger and more accurate decision. Thus, similar objects (with similar features) of different classes are clustered together in the initial nodes of the hierarchy. As the tree expands, these classes are gradually set apart. The tree is terminated when the algorithm does not find any common feature to partition the inputs (at the leaves of the tree). Thus, each leaf of the tree corresponds to a particular class. After the feature hierarchy is learned, SVMs at each node (excluding the leaves) of the hierarchy are trained using regularized hinge loss minimization [21], as discussed earlier.

Traditionally, boosting algorithms use multi-class weak learners to construct a multi-class final strong classifier [20], [23]. However, for large number of classes, constructing reasonably accurate multi-class weak learners turns out to be highly computationally expensive. Thus, we use the minimum entropy measure to transform the multi-class into a 2-class problem and then learn the feature hierarchy.

C. Testing

The tree composed of SVM nodes is then used for testing. Those instances (easy) that can be easily distinguished with general features are identified with SVMs at the top nodes. The SVMs at the bottom nodes perform more accurate classification on the hard instances in the dataset. When an input instance is presented at the root node, the branch with higher output probability at the SVM node is activated.

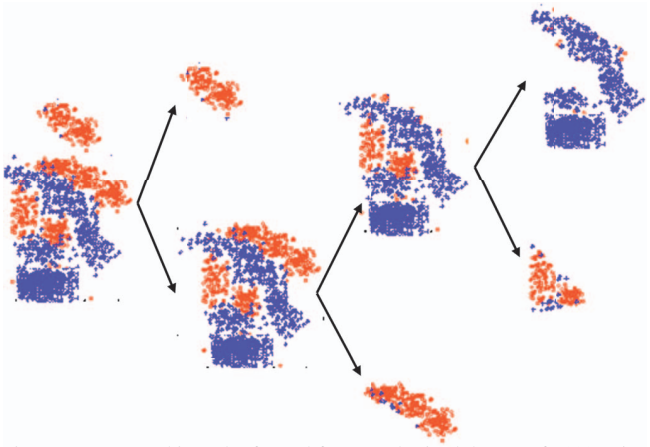

Fig. 2. Learnt tree hierarchy formed for a synthesized dataset of 3000 points. Features for weak classifiers are position or distance to some specific 2D lines

Based on the path activated by the output of SVM nodes, the instance then traverses the hierarchy until a leaf node where a final decision (or class assignment) is made. Note that a subset of classes are eliminated at each tree node as the tree is traversed. The feature based hierarchy, thus, scales sub-linearly *O(log(n))* with respect to the number of classes. In the current era of "data deluge" that presents vision problems with a hefty task of recognizing from hundreds or thousands of classes, the sub-linearly growing tree model can be very useful.

D. Understanding the feature hierarchy

The training algorithm naturally divides the samples into left and right sub-groups based on the configuration of features. Fig. 2 shows an example of how the tree learns and divides the samples on a synthesized dataset of 3000 points. The dataset consists of inputs belonging to two classes (denoted as orange and blue). The samples that are clustered together can be termed as hard inputs. Such samples are passed down the sub-branches of the tree forming the successive nodes. The top node of the tree partitions the inputs into two subsets. This division is intuitive as the right set of orange points are distant from the remaining inputs that are clustered together. The tree then expands on the hard inputs where the two sets are clustered together. If these data points are assumed to be features (like texture or color components) corresponding to two image classes, it is clearly seen that the hierarchy formed is coherent with the basic generic-to-specific feature transition theory of the proposed model. Note, our model automatically learns this feature hierarchy without any need to pre-specify the feature clusters. The pruning of the input data as we traverse down the tree reduces the complexity of the original multi-class problem. This in turn enables better decision boundary modelling at the bottom SVM nodes as compared to the top node resulting in improved classification performance.

A noteworthy observation here is that the model comprises of multiple decision paths of different lengths. In Fig. 2, the tree consists of leaf nodes (for orange data points) at every level. For a given input, the decision can be reached at an earlier leaf node yielding a more optimal speedup during testing. This imbalanced decision tree structure is what separates our model from other decision tree methods where one has to traverse the entire tree to reach a decision [14], [18]. Even within a particular class, all inputs are not equal. Ideally, algorithms should spend effort proportional to the difficulty of the inputs irrespective of whether they belong to the same class or not [24]. Most existing works [14], [22] focus on optimizing the computational complexity based on inter-class feature variability. In contrast, our imbalanced method captures both inter and intra class feature variability while expanding the tree thus yielding more computational benefits.

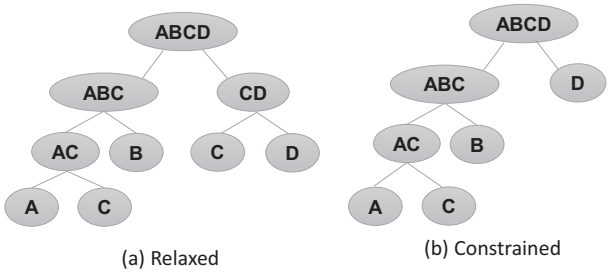

Fig. 3. Sample illustration of tree with Relaxed and Constrained hierarchy

E. Constrained vs Relaxed Hierarchy

Previously, we discussed that the threshold, Δ, serves as a useful control parameter to construct either relaxed or constrained models of the tree. Fig. 3 demonstrates the sample relaxed/constrained hierarchy for a 4-class problem. The instances from class C are the hard inputs in the dataset. In the constrained hierarchy, it is clearly seen that instances from C are forced to the left sub-node. In this case, it is very likely that the SVM at the root node will misclassify a test instance from class C due to overfitting. However, the decision path for recognizing class D is short. So, we will observe an improvement in efficiency (or test speed) at the cost of accuracy. With relaxed hierarchy, we see that there is an extra SVM classifier evaluation required to recognize class D that increases the computational cost. However, the accuracy in this case will be better as the addition of an extra classifier node (Node CD in Fig. 3(a)) minimizes overfitting for complex distribution of data. In addition, the relaxed hierarchy captures the intra-class feature variability for class C which is not seen in the constrained model. In the relaxed model, instances of class C that are relatively easy can be classified at the 2nd level and those that are hard are only passed to the 3rd level for accurate classification. In contrast, with constrained model all instances of class C are passed to the 3rd level for classification. Fig. 4 shows the sample demonstration of the efficiency vs. accuracy tradeoff obtained for a given dataset with relaxed/ constrained hierarchy. It is clearly evident that Δ can be modulated to control the accuracy and efficiency of the proposed model.

V. Experiments

In this section, we evaluate our proposed framework on an object recognition task for the benchmark dataset, Caltech-256 [25]. We use Caltech as it consists of images corresponding to a vast range of classes. It also has been used in contemporary tree-based methods that form a baseline in our analysis for comparison against our model. In this work, for simplicity, we used SVMs with linear kernels as our binary classifiers at the tree nodes.

We use the evaluation metrics: classification accuracy and test speed (or test complexity) to discuss the benefits of our approach. For classification accuracy, we use the mean of per-class accuracy that is reported as a standard way for estimating multi-class classification performance. Since we use linear SVM kernels, the overall test complexity is proportional to the number of evaluated classifiers. So, we report the mean of the number of classifier evaluations for all test instances for test speed. We compare our method to various existing approaches: Gao [14], one-vs-all, one-vs-one, DAGSVM, tree-based

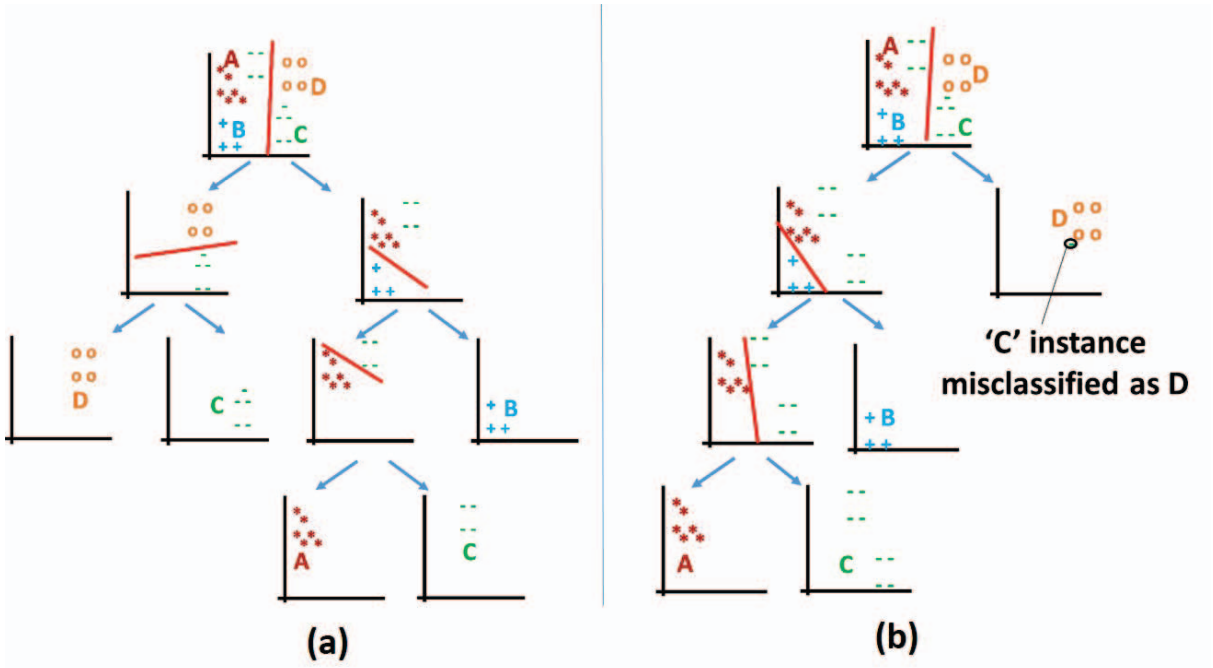

Fig. 4. (a) Relaxed Hierarchy without misclassification error at the cost of longer decision path for 'D' (b) Constrained Hierarchy with misclassification error but shorter decision path for 'D' (Tree structure similar to Fig. 3)

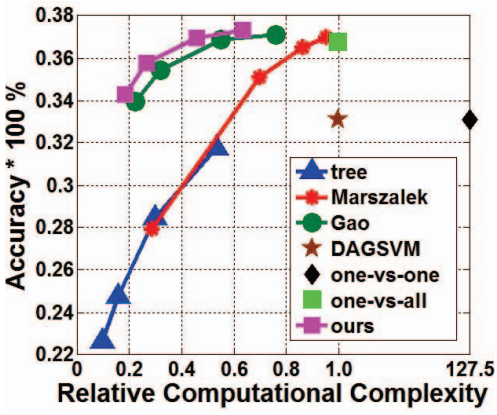

Fig. 5. Comparison of the tradeoff between accuracy and Relative Complexity on Caltech 256. The computational complexity is normalized by the complexity of one-vs-all. Note that for one-vs-one the relative complexity with linear kernel is 127.5.

hierarchy [13] and Marszalek [22]. Note, the regularization parameter of SVM is chosen by cross validation on the training set.

A. Caltech-256

With 256 categories and at least 80 images per class, this is a standard muti-class object recognition dataset. We randomly sampled 80 images for each class, and used half (40 per class) for training and the remaining half for testing. For features, we used the standard spatial histograms of visual words based on dense SIFT [16]. We varied computational parameters for tree [13] (2 to 5 levels), Marszalek [22] ($\alpha \in \{0.2, 0.5, 0.6, 0.8\}$), Gao [14] ($\rho \in \{0.5$ to 0.8 with step size of 0.1$\}$) and our method ($\Delta \in \{0.5$ to 0.9 in steps of 0.1$\}$) to obtain a tradeoff between accuracy and speed. Here, α and ρ are the computational parameters defined in [22] and [14] respectively that are varied to achieve the complexity vs. accuracy tradeoff.

Fig. 5 shows the results. It is clearly seen that our tree performs better (faster at same accuracy and more accurate at the same Relative Complexity (RC)). For instance, our model achieves one of the best accuracy ($\sim$37.3%) with around 27% of the complexity of one-vs-all with a *relaxed hierarchical* model (where $0.5 < \Delta < 1$) while achieving a speedup of 3.7x. Also, for $\Delta = 0.5$, when the tree is modelled as a *constrained hierarchy*, it achieves a higher speed up of 5.5x for $\sim$2.5% accuracy degradation with respect to one-vs-all. However, to achieve a similar 5x speed up other methods: Gao [14], Marszalek [22], tree [13] have to suffer 3.2%, 8%, 10% accuracy degradation. Please note that our model achieves consistently better accuracy performance than the best result reported in [14] for linear kernel SVM.

B. Hierarchy of visual features

Our model builds a feature hierarchy in the label space automatically. Fig. 6 shows the tree formed for a subset of some sampled images from the Caltech-256 dataset. We observe that the images that have similar features are clustered together in an initial node and are gradually set apart as the tree is traversed. Conforming to the imbalanced hierarchical feature model, we observe that for certain classes: zebra, car tire, the classification is done at earlier nodes while more confusing classes are passed down. In addition, we also observe intra-class variability for the camel class in which certain instances are evaluated earlier than others.

VI. Conclusion

We proposed a novel neuro-inspired visual feature learning to construct an efficient and accurate tree-based classifier for large-scale image classification. Our learning algorithm is based on the biological attention mechanism observed in the brain that selects specific

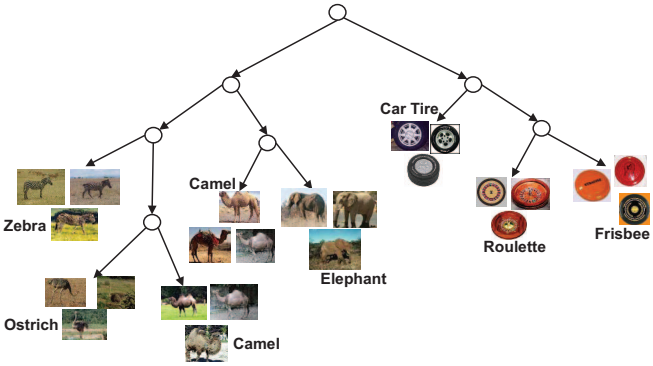

Fig. 6. Tree hierarchy formed for a sub-sample of selected images from Caltech-256

features for greater neural representations. The tree uses a principled optimization procedure (recursive Adaboost training) to extract knowledge about the relationships between object types and integrates that into the visual appearance learning. We evaluated our method on the Caltech-256 dataset and obtained significant improvement in accuracy and efficiency. In fact, our model outperforms the one-vs-all method in accuracy and yields lower computational complexity compared to the state-of-the-art "tree-based"methods [14], [22]. The proposed framework intrinsically embeds clustering in the learning procedure and identifies both inter and intra class variability. Most importantly, our proposed tree learns the hierarchy in a systematic and less greedy way that grows sublinearly with the number of classes and hence proves to be very effective for large-scale classification problems. It is noteworthy to mention that the current framework suffers from overfitting when the training dataset is small. The overfitting behaviour is checked by modulating the depth of the tree and also adopting the relaxed hierarchy structure where confusing or "hard" inputs are passed to both the right and the left sub-nodes. Additionally, tree pruning methods [26] can be used to control overfitting . Further research can be done to explore the overfitting problem. Future work includes using non-linear SVM kernels at the tree nodes and analyzing other complex and larger datasets [2] to analyse the efficiency-accuracy tradeoff with the proposed approach.

Acknowledgment

This work was supported in part by C-SPIN, one of the six centers of StarNet, a Semiconductor Research Corporation Program, sponsored by MARCO and DARPA, by the Semiconductor Research Corporation, the National Science Foundation, Intel Corporation and by the Vannevar Bush Faculty Fellowship.

References

[1] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[2] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "Sun database: Large-scale scene recognition from abbey to zoo," in *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*. IEEE, 2010, pp. 3485–3492.

[3] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[4] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[5] S. E. Yuksel, J. N. Wilson, and P. D. Gader, "Twenty years of mixture of experts," *IEEE transactions on neural networks and learning systems*, vol. 23, no. 8, pp. 1177–1193, 2012.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[8] S. K. Ungerleider and L. G, "Mechanisms of visual attention in the human cortex," *Annual review of neuroscience*, vol. 23, no. 1, pp. 315–341, 2000.

[9] D. Whitney, "Neuroscience: toward unbinding the binding problem," *Current Biology*, vol. 19, no. 6, pp. R251–R253, 2009.

[10] E. Grossmann, "Adatree: boosting a weak classifier into a decision tree," in *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*. IEEE, 2004, pp. 105–105.

[11] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[12] J. Deng, S. Satheesh, A. C. Berg, and F. Li, "Fast and balanced: Efficient label tree learning for large scale object recognition," in *Advances in Neural Information Processing Systems*, 2011, pp. 567–575.

[13] G. Griffin and P. Perona, "Learning and using taxonomies for fast visual categorization," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

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

[15] 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 *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1689–1696.

[16] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[17] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[18] Z. Tu, "Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 2. IEEE, 2005, pp. 1589–1596.

[19] R. E. Schapire, "Explaining adaboost," in *Empirical inference*. Springer, 2013, pp. 37–52.

[20] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[21] X. Li, L. Wang, and E. Sung, "A study of adaboost with svm based weak learners," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 1. IEEE, 2005, pp. 196–201.

[22] M. Marszałek and C. Schmid, "Constructing category hierarchies for visual recognition," in *European Conference on Computer Vision*. Springer, 2008, pp. 479–491.

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

[24] P. Panda, A. Sengupta, and K. Roy, "Conditional deep learning for energy-efficient and enhanced pattern recognition," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2016, pp. 475–480.

[25] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.

[26] O. T. Yıldız, "Vc-dimension of univariate decision trees," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 2, pp. 378–387, 2015.