THE USE OF DEEP LEARNING FEATURES IN A HIERARCHICAL CLASSIFIER LEARNED WITH THE MINIMIZATION OF A NON-GREEDY LOSS FUNCTION THAT DELAYS GRATIFICATION

Zhibin Liao and Gustavo Carneiro

ARC Centre of Excellence for Robotic Vision, University of Adelaide, Adelaide, Australia

ABSTRACT

Recently, we have observed the traditional feature representations are being rapidly replaced by the deep learning representations, which produce significantly more accurate classification results when used together with the linear classifiers. However, it is widely known that non-linear classifiers can generally provide more accurate classification but at a higher computational cost involved in their training and testing procedures. In this paper, we propose a new efficient and accurate non-linear hierarchical classification method that uses the aforementioned deep learning representations. In essence, our classifier is based on a binary tree, where each node is represented by a linear classifier trained using a loss function that minimizes the classification error in a non-greedy way, in addition to postponing hard classification problems to further down the tree. In comparison with linear classifiers, our training process increases only marginally the training and testing time complexities, while showing competitive classification accuracy results. In addition, our method is shown to generalize better than shallow non-linear classifiers. Empirical validation shows that the proposed classifier produces more accurate classification results when compared to several linear and non-linear classifiers on Pascal VOC07 database.

1. INTRODUCTION

Ever since Krizhevsky and Hinton [1] published the outstanding classification results on ImageNet [2], the attention of the computer vision community has shifted from the bag of features [3] and Fisher vector [4] representations to the deep learning representation provided by convolutional neural networks [5, 6]. Recent results [7, 8, 9, 10, 11, 12] show that convolutional neural networks (CNN), a high-capacity multi-layer non-linear classification framework, can produce the most accurate classification results in several databases in the field. It is generally believed the convolutional layers of CNN are responsible for generating well-separated image representations. For this reason, simple linear support vector machine (SVM) and softmax [1] classifier are used to classify the CNN image representations. More complex classifiers are with higher capacity, such as non-linear SVM [13, 14] or boosting [15], but the main limitations are the high training and testing time complexities and the risk of overfitting the training data. Hierarchical models can achieve a good trade off between generalization and training and testing complexities, and for this reason it has been explored in computer vision in the past, such as with the probabilistic boosting tree (PBT) [16] and the discriminative learning of relaxed hierarchy (DLRH) [17]. In this paper, we propose a new hierarchical classifier to be used with high dimensional feature vectors (e.g., CNN features). The main novelty of our proposal is the loss function to learn this hierarchical classifier, which minimizes the classification error in a non-greedy way and at the same time delays hard classification problems to nodes further down the tree. Our main objective with this new training process of hierarchical classifiers is to reach a good trade-off between generalization and complexities, particularly when compared with linear [1, 7] , shallow non-linear [13, 15, 14] and hierarchical classifiers [17, 16] previoulsly proposed in the field. We test our method on the Pascal VOC07 [18, 19] database, and show that we produce better classification results and have less training test complexities, compared to other linear and non-linear classifiers using the deep learning features [7, 12].

This research was supported by the Australian Research Council Centre of Excellence for Robotic Vision (project number CE140100016)

2. LITERATURE REVIEW

In this section, we briefly describe the related work in the field, by first introducing the mid-level features extracted from convolutional neural nets, then by discussing the classifiers that are (or can) be used with such feature vectors.

Deep learning methods have been present in the field for several years [20, 21], but their use as feature generators has been considered only after the outstanding result obtained on ImageNet [1]. Using millions of training images, a large number of nodes and hidden layers and an effective implementation, Krizhevsky and Hinton [1] produced a CNN model that integrates feature extraction and classification functionalities into an easy-to-use framework. In addition to this, Razavian et al. [11] have found that the CNN mid-layer activations trained with ImageNet provide powerful off-the-shelf image representations for other databases. This results have been confirmed by Chatfield et al. [3], who showed that much deeper CNN mid-layer activations produce much more accurate classification results than the previous bag of features [3] and Fisher vector [4] representations.

An interesting observation about the use of these CNN features is that the classifier is usually based on linear SVM [13] or softmax [1], which are relatively low-capacity models that are efficiently trained and tested, but offer limited accuracy depending on the distribution of the training set samples. In general shallow high-capacity classifiers can handle more difficult classification problems better than linear classifiers, but at a usually higher running time computational cost and the risk of overfitting. On the other hand, hierarchical classifiers that use low capacity classifiers in each node, usually achieve a good trade-off between complexity and accuracy.

Boosting [15] and non-linear SVM [13] are typical examples of high-capacity shallow classifiers that can deal with highly complex classification problems. The former uses a quite large number of features and build simple linear (weak) classifier in each one of these feature spaces, which are then combined to form a strong clas-

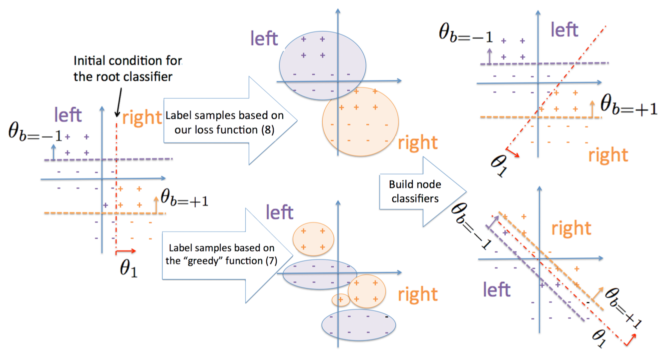

Fig. 1: Comparison between the learning of the root node classifier parameter θ_1 (in a binary classification problem, with labels '+' and '-') using our proposed loss function in (5) in comparison to the "greedy" loss function in (4).

sifier. In general, the training of each weak classifier involves the estimation and comparison of low capacity classifiers in all possible feature spaces, which means that at least this training has complexity $O(DN)$, where D is the size of the feature space and N is the training set size. Considering that potentially, one can use the whole feature space, the complexity of the full training process is around $O(D^2N)$ and the testing complexity is $O(D)$. Non-linear SVM uses the kernel trick to project all training samples in the kernel space and builds a linear classifier there, which is a training process that has complexity around $O(DN^2)$. The testing complexity of non-linear SVM is $O(ND)$. In addition to the complex training and testing processes, both methods can overfit the training data given their large capacity.

The issues presented above have been realized in several works that propose the use of hierarchical classifiers instead of their shallow counterparts. The boosted cascade classifier [22] is a good example of a hierarchical classifier, based on a degenerate binary tree with a boosting classifier on each node, where the training involved in each node minimizes a loss function that greedily estimates the best decision boundary, but delays the decision about hard classification problems. The probabilistic Boosting Tree (PBT), proposed by Tu [16], is a full binary tree structured classifier that has a boosting classifier in each node, which minimizes a similar loss function. One important issue that affects the cascade and PBT classifiers is that each node uses a high capacity boosting classifier that can overfit the training data, hampering their generalization abilities and incrementing the training and testing time complexities. A more relevant work to our proposal is the discriminative learning of relaxed hierarchy (DLRH) [17], which consists of a binary tree, where each interior node is trained using a loss function that delays hard classification problems, but at the same time greedily reduces the classification error. This greedy error reduction can again overfit the training data.

3. METHODOLOGY

Assume that an image is represented by a feature vector $\mathbf{x} \in \mathbb{R}^D$, the image label is represented by the variable $y \in \{-1, 1\}$, and the binary tree classifier has one root node that classifies samples using a hidden variable $b \in \{-1, 1\}$ (which indicates the left child by -1 and right by $+1$).

The inference is defined by the following problem:

$$y^* = \arg \max_{y \in \{-1,1\}} \sum_{b \in \{-1,1\}} P(y|\mathbf{x}, b, \theta_b) P(b|\mathbf{x}, \theta_1), \quad (1)$$

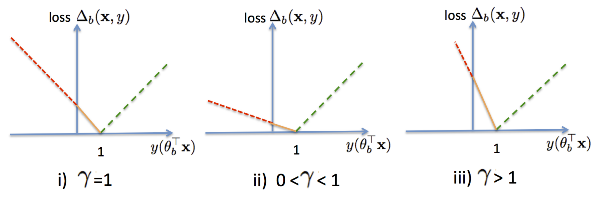

Fig. 2: Shape of the proposed loss function (5), where dotted red shows the region where classification is incorrect, solid orange displays the margin region, and dashed green depicts correct classification.

where $\theta_b, \theta_1 \in \mathbb{R}^D$ denote the classifier parameters of the leaf and root nodes, respectively. This inference is easily extended to a tree with three levels, as follows:

$$\begin{aligned} P(y|\mathbf{x}, \Theta) = & \sum_{b^{(1)}} \sum_{b^{(2)}} \sum_{b^{(3)}} P(y|\mathbf{x}, b^{(1)}, b^{(2)}, \theta_{b^{(1,2)}}) \\ & P(y|\mathbf{x}, b^{(1)}, b^{(3)}, \theta_{b^{(1,3)}}) P(b^{(2)}|\mathbf{x}, b^{(1)}, \theta_2) \\ & P(b^{(3)}|\mathbf{x}, b^{(1)}, \theta_3) P(b^{(1)}|\mathbf{x}, \theta_1), \end{aligned} \quad (2)$$

where $b^{(1)}, b^{(2)}, b^{(3)} \in \{-1, 1\}$, and we assume that the classifier in the root node is represented by the variable $b^{(1)}$, its left child by $b^{(2)}$ and right child by $b^{(3)}$. The extension to higher trees is then trivial.

For the learning procedure, assume the availability of a training set $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^M$ and a validation set $\mathcal{V} = \{(\mathbf{x}_i, y_i)\}_{i=1}^Q$. The description of the learning methodology is clearer if we assume the tree has only one parent node (the extension to larger tree is again trivial), where the ideal learning process uses the training set $\mathcal{T}$ as follows:

$$\{\theta_b^*, \theta_1^*\} = \arg \max_{\{\theta_b, \theta_1\}} \prod_{i=1}^M \sum_{b_i} P(y_i|\mathbf{x}_i, b_i, \theta_b) P(b_i|\mathbf{x}, \theta_1), \quad (3)$$

where we assume that the training samples are i.i.d. and $b_i \in \{-1, 1\}$. Eq. 3 involves the maximization of two functions, which are hard to be optimized jointly, so we break this optimization into the following iterative algorithm containing two alternating stages, where we assume that the parameters $\theta_b^{(t-1)}$ and $\theta_1^{(t-1)}$ are known at stage (t):

$$\begin{aligned} \theta_1^{(t)} &= \arg \max_{\theta_1} \prod_{i=1}^M \sum_{b_i} P(y_i|\mathbf{x}_i, b_i, \theta_b^{(t-1)}) P(b_i|\mathbf{x}, \theta_1), \\ \theta_b^{(t)} &= \arg \max_{\theta_b} \prod_{i=1}^M \sum_{b_i} P(y_i|\mathbf{x}_i, b_i, \theta_b) P(b_i|\mathbf{x}, \theta_1^{(t)}). \end{aligned} \quad (4)$$

Essentially, this learning procedure involves the division of the training samples into two clusters, one representing the left child samples (labeled as $b = -1$) and the other, the right child samples (labeled as $b = +1$). After this division is performed, the parameter θ_1 of the classifier $P(b|\mathbf{x}, \theta_1)$ is estimated, and then, the classifier associated with each child can have its parameter θ_b estimated based on the samples belonging to that child (which is decided based on $P(b|\mathbf{x}, \theta_1)$). We consider this training process to be greedy because we maximize the classification probability $P(y|\mathbf{x}, b, \theta_b)$ even when splitting the training points to the left or right children. Our main contribution in this paper is the proposal of a non-greedy loss function used for the estimation of θ_1 in (4), which also delays hard classification problems.

The motivation for our proposed loss function is based on the graphs shown in Fig. 1. Assuming that both $P(y|\mathbf{x}, b, \theta_b)$ and $P(b|\mathbf{x}, \theta_1)$ are represented by linear classifiers (which means that

θ_b and θ_1 are vectors denoting the normal vectors of the learned hyperplane), notice that if we try to maximize $P(b|\mathbf{x},\theta_1)P(y|\mathbf{x},b,\theta_b)$ when estimating θ_1, using the depicted initial guess (graph on left), we will greedily label the '+' points with '+1' (right child) and the '-' points with '-1', which generates a very difficulty learning problem in the next iteration of the algorithm (graph on the bottom of Fig. 1). On the other hand, our loss function has a shape depicted by Fig. 2, which means that

1. if the classifiers in both children generate a correct classification (or both are incorrect), pick the one (with its respective child label) with the closest hyperplane,
2. if the classifier of only one child is correct, pick the correct one as long as it is not farther from the margin than the incorrect one.

This means that in addition to performing a division of the training samples without trying to greedily minimize the classification error (item (1) above), we also delay the hard classification problems to later stages of the binary tree [17, 16], as depicted in top part of Fig. 1. The loss function depicted in Fig. 2 is defined by:

$$f(b_i;\mathbf{x}_i,y_i)=\sum_b \Delta_b(\mathbf{x}_i,y_i)\delta(b_i-b), \quad (5)$$

with $\delta(.)$ denoting the Dirac delta function and

$$\Delta_b(\mathbf{x},y)=\gamma\max(0,1-y(\theta_b^\top\mathbf{x}))+\max(0,y(\theta_b^\top\mathbf{x})-1), \quad (6)$$

where $\gamma\in\mathbb{R}^+$ is a scalar that weighs the relative importance of the two terms in (6), and θ_b denotes the parameter of a linear SVM classifier learned for each child node b using the training samples $\mathbf{x}_i,y_i$ where $b_i=b$.

The proposed learning algorithm iterated steps 1-3 below (until convergence), assuming that we have the estimated value for $\theta_b^{(t-1)}$:

1). For each training sample $i\in\{1,...,M\}$, determine its label b_i with:

$$b_i^{(t)}=\arg\max_{b\in\{-1,+1\}} f(b_i;\mathbf{x}_i,y_i), \quad (7)$$

defined in (5) and using $\theta_b^{(t-1)}$;

2). Estimate $\theta_1^{(t)}$, as follows:

$$\begin{aligned}\theta_1^{(t)}&=\arg\max_{\theta_1}\prod_{i=1}^{M}P(b_i^{(t)}|\mathbf{x},\theta_1),\\ &=\arg\min_{\theta_1}\frac{1}{2}\|\theta_1\|^2+\lambda\sum_{i=1}^{M}\log(1+\exp(-b_i^{(t)}\theta_1^\top\mathbf{x}_i)),\end{aligned} \quad (8)$$

which is a logistic regression classifier;

3). Estimate $\theta_b^{(t)}$, with:

$$\begin{aligned}\theta_b^{(t)}&=\arg\max_{\theta_b}\prod_{i=1}^{M}P(y_i|\mathbf{x}_i,b_i^{(t)},\theta_b)\delta(b_i^{(t)}-b)\\ &=\arg\min_{\theta_b}\frac{1}{2}\|\theta_b\|^2+\lambda\sum_{i=1}^{M}\max(0,1-y_i(\theta_b^\top\mathbf{x}_i)),\end{aligned} \quad (9)$$

for $b\in\{-1,+1\}$, which is the definition for the linear SVM classifier (note that we use the soft margin training that allows for non-separable problems).

The initialization of this algorithm is achieved with the K-means clustering (with two clusters) considering the set $\{\mathbf{x}_i|y_i=+1,(\mathbf{x}_i,y_i)\in\mathcal{T}\bigcup\mathcal{V}\}$. We run this clustering 20 times and pick the labels b_i for the case that minimizes the loss in (5), and then we run from step (2) of the algorithm above. The stopping criterion is also based on the computation of the loss in (5), where when the loss difference between two iterations is smaller than a threshold ϵ, then we stop iterating. Finally, there is also a model selection problem involved in this method concerning the structure of the binary tree, where after convergence, we verify a condition to determine if we will backtrack to the previous structure or keep the current tree structure and thus continue to grow the tree. The initial tree contains only the root node and a linear SVM classifier (i.e., this is the original classifier found in previous works [7]), and the condition that we use for model selection is the classification accuracy (e.g., mean average precision) measured in the validation set $\mathcal{V}$ using the latest trained tree.

3.1. Complexity Analysis

For each node expansion, we train three separate linear classifiers per iteration of our algorithm. Therefore, we need K iterations in a tree with N nodes, where $N\in[2^h-1,2^{h+1}-1]$, with $h=$maximum tree depth, so the training complexity of our method is $O(3KNDM)$ (recall that D is the feature size and M is the training set size), so this means that compared to the linear classifier, our training is $3KN$ slower. Fortunately, we can control the values for K and N by constraining the number of iterations and the tree depth, and in general we have $KN << D,M$, which means that our training is significantly faster than typical shallow non-linear classifiers (see Sec. 2). The testing complexity is essentially based on running h linear classifiers, which means that the running time complexity is $O(hD)$, which is just marginally larger than a single linear classifier given that $h\leq 3$, typically.

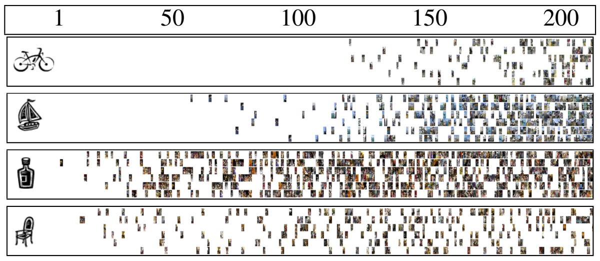

Fig. 3: Sorted Pascal VOC07 [19] first 200 retrieval results by using the VGG features [7]. Note that only the incorrectly classified images are shown, so the sparsity pattern of the correct retrieval can be analyzed by noticing the white regions of the image. For each presented class (see icon on the left [7]), each row is the result of the methods PBT [16], DLRH [17], Non-linear SVM (3-poly), Non-linear SVM (RBF), Linear SVM, and our method, respectively.

4. EXPERIMENTS

We quantitatively compare our proposed hierarchical classifier method with linear and non-linear SVM classifiers (RBF and 3rd-degree polynomial kernels), and also with the following hierarchical classification methods: PBT [16] and DLRH [17]. For the compared methods except the DLRH [17] method, which learns a single model for all classes jointly, we adopt the one-against-all (OVA) strategy for training each class. We re-implemented the PBT method exactly as described by Tu [16], and use the publicly available training method by Gao and Koller [17] available from their web page to train the DLRH. The LIBLINEAR [23] and LIBSVM [24] are used for training linear and non-linear SVMs respectively. The implementation of our method involves setting the values for the maximum number of iterations to $K=1$ and the maximum number of tree nodes to $N=3$ (see Sec. 3.1). Also, the weight γ in (5) and (6)

Table 1: AP results on Pascal VOC07 [19] using the **OverFeat** [12] and **VGG** [7] features. The best results per class are highlighted.

	Classifier	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	MAP
OverFeat features [12]	Non-linear SVM (RBF)	87.59	78.64	**87.72**	78.76	**46.61**	70.11	85.34	**82.44**	**63.81**	**59.61**	66.37	77.27	77.82	**80.92**	89.11	47.06	70.77	61.46	**88.41**	68.66	73.42
	Non-linear SVM (3-poly)	85.26	79.46	82.31	81.30	42.55	72.18	83.31	80.87	59.41	58.84	**67.26**	76.91	79.01	78.22	88.88	54.17	71.13	63.49	87.21	71.08	73.14
	DLRH [17]	86.69	78.33	82.94	82.39	37.00	69.52	85.26	81.21	59.08	52.37	65.54	78.56	79.00	79.28	88.33	53.17	68.72	60.99	86.22	68.35	72.15
	PBT [16]	85.44	78.40	82.86	80.54	40.48	69.38	84.41	78.01	55.68	59.35	63.94	75.46	81.11	77.41	89.36	49.72	**71.96**	56.20	85.85	66.35	71.60
	Linear SVM	**88.89**	79.72	84.35	82.02	44.63	73.26	**85.98**	81.76	61.47	57.49	67.08	77.99	81.20	78.92	90.55	**55.71**	71.43	**63.57**	87.13	**72.10**	74.26
	Our Method	**88.89**	**80.63**	84.32	**82.75**	43.25	**73.55**	85.66	81.76	61.47	59.51	67.20	**79.03**	**81.20**	78.92	**90.72**	**55.71**	71.46	**63.57**	86.96	**72.10**	**74.43**
	Classifier	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	MAP
VGG features [7]	Non-linear SVM (RBF)	90.76	**87.33**	89.79	86.47	51.83	**81.99**	**88.29**	87.89	64.01	**77.47**	76.82	**86.64**	88.13	**85.80**	92.19	60.82	82.36	70.15	90.39	75.38	80.73
	Non-linear SVM (3-poly)	90.39	84.91	88.15	86.06	53.47	79.16	85.40	86.46	62.85	73.91	**78.42**	84.57	84.69	81.16	89.63	59.66	81.82	70.49	90.83	73.50	79.28
	DLRH [17]	**93.13**	85.19	88.99	86.69	51.28	78.28	87.67	87.69	60.72	71.58	72.63	85.92	85.64	84.00	90.19	58.21	80.56	70.80	91.41	71.81	79.12
	PBT [16]	91.16	84.58	87.96	82.70	47.68	77.30	86.02	86.68	57.77	74.75	69.18	82.28	86.89	82.32	91.20	53.90	79.49	66.30	91.19	70.97	77.52
	Linear SVM	91.44	86.31	89.54	85.93	53.16	79.74	87.74	87.93	64.79	75.97	78.05	85.14	**88.20**	83.83	92.33	60.16	**82.52**	71.59	91.84	74.57	80.54
	Our Method	91.90	86.80	**89.88**	**86.59**	**53.59**	80.47	87.87	**88.75**	**66.50**	75.53	77.87	85.88	87.83	84.46	**92.70**	**62.18**	82.22	**72.65**	**91.94**	**75.38**	**81.05**

that controls the shape of loss function is set at $\gamma = 1$ (we reached this value by varying $\gamma \in [0.1, 100]$ and noticing little change in the classification results).

We evaluate the performance of the methods on the Pascal VOC07 [19] database, which contains 9963 images of 20 visual classes. The performance of each class is measured by average precision (AP), which is the standard ranking efficiency measurement. The mean AP (MAP) of all 20 classes is also listed to show the average performance. The parameter C of the SVM training are cross-validated from the standard training-validation split of the database. The running time of the training and testing processes are obtained using a computer with the following configuration: Intel(R) Xeon(R) CPU @ 2.70GHz, with 8GB of memory.

The features are extracted from two publicly available CNN models: OverFeat [12] and VGG [7]. Both models were trained on the ImageNet ILSVRC12 [25] database, which contains 1.2 million images with 1000 classes. For all experiments, the extracted CNN features are $L2$ normalized before being used as the input features for the classifiers. The OverFeat [12] feature is extracted from the first fully connected layer (layer 22) of its 'accurate' architecture, which has 4096 dimensions and is the fully connected layer before the rectification layer. No data augmentations were used to increase the training volume. For the VGG [12], Chatfield et al. offer several pre-trained models and data augmentation choices. We choose the 2048-dimensional feature model with the augmentation choice 'ss' (both training and testing are max-pooled as one sample from the ten samples) option because this combination is reported with the top results in [7]. In our experiments, we noticed the VGG features are extracted after the rectification layer (note that we consider the fact that these features are extracted after the rectification layer as one of the major differences compared to the OverFeat features).

Table 2: Average running times (in seconds) of the training and testing methods using the VGG features of Table 1 on Pascal VOC07 [19]. The results is reported as the average training time (in seconds) per visual class and the testing time per testing image.

Classifier	Training time/class	Testing time/image
Non-linear SVM (RBF)	36.7	36.4
Non-linear SVM (3-poly)	40.1	34.1
DLRH [17]	16.6	2.5
PBT [16]	1459.3	0.5
Linear SVM	1.4	0.6
Our method	11.7	0.1

4.1. Results

In Table 1, we show the AP results obtained with the OverFeat features [12] and VGG features [7] on Pascal VOC07 [19] using the studied classifiers. The running time of the training and testing processes of these classifiers listed are shown in Table 2. We show in Fig. 3 the sorted retrieval results, where only the wrong cases are shown in order to assess the sparsity pattern of the detection results for each classifier (i.e., the white regions represent correctly retrieved samples).

5. DISCUSSION AND CONCLUSIONS

By analyzing the results, our methodology provides the best MAP among all considered classifiers in Tables 1. We also notice to notice the non-linear methods provide competitive results for some individual classes. This demonstrates that non-linear SVM methods are relevant, particularly in cases where the database is not large (such as the case with Pascal VOC07). On the other hand, the linear SVM always produce robust average classification results, but rarely presents the best per-class result. This shows that linear SVM can generalize well but may have difficulties dealing with some of the more complicated classification problems. Compared to the other hierarchical methods [17, 16], our approach shows superior accuracy, which, we believe, is related to the use of our proposed loss function for training the hierarchical classifier and the fact that the node classifiers are linear methods (which provides good generalization properties). Analyzing the sparsity pattern of the retrieval results in Fig. 3, we see that our approach tends to have one of the most sparse results among the studied methods. It also shows that our approach takes longer to retrieve the first incorrectly classified images. These two facts confirm the AP results from Tables 1.

In addition, our binary trees can grow up to $N = 3$ nodes limits the number of vector multiplications ($\mathbf{x}^\top \theta$) to 2 times per test image which brings the fastest testing time. For linear SVM, the testing time is longer than our method mostly because of the overheads involved in running LIBLINEAR [23]. In terms of training time, we are significantly faster than the non-linear SVM methods and PBT [16], but comparable to DLRH [17], while the linear SVM classifier has the fastest training time. We believe that our training time presents competitive results because we limit the number of iterations to $K = 1$ and the size of the tree in $N = 3$ nodes (see Sec. 3.1), which make the theoretical training complexity only slightly larger than linear SVM.

We plan to apply this method to other databases [2, 26, 27] and use other CNN features that will become available in the near future. We also plan to increase the difficulty of the current database and verify if the impact of the non-linear methods is more remarkable.

6. REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 248–255.

[3] Josef Sivic and Andrew Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 1470–1477.

[4] Florent Perronnin, Jorge Sánchez, and Thomas Mensink, "Improving the fisher kernel for large-scale image classification," in *Computer Vision–ECCV 2010*, pp. 143–156. Springer, 2010.

[5] Dan Ciresan, Ueli Meier, and Jürgen Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649.

[6] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[7] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *arXiv preprint arXiv:1311.2524*, 2013.

[9] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[10] Maxime Oquab, Leon Bottou, Ivan Laptev, Josef Sivic, et al., "Learning and transferring mid-level image representations using convolutional neural networks," 2013.

[11] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," *arXiv preprint arXiv:1403.6382*, 2014.

[12] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv preprint arXiv:1312.6229*, 2013.

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

[14] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[15] Yoav Freund and Robert E Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*. Springer, 1995, pp. 23–37.

[16] Zhuowen Tu, "Probabilistic boosting-tree: Learning discriminative models for classification, recognition, and clustering," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. IEEE, 2005, vol. 2, pp. 1589–1596.

[17] Tianshi Gao and Daphne Koller, "Discriminative learning of relaxed hierarchy for large-scale visual recognition," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2072–2079.

[18] Mark Everingham, LV Gool, Chris Williams, and Andrew Zisserman, "Pascal visual object classes challenge results," *Available from www. pascal-network. org*, 2005.

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[20] Yann LeCun and Yoshua Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, 1995.

[21] Geoffrey E Hinton and Ruslan R Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[22] Paul Viola and Michael Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. IEEE, 2001, vol. 1, pp. I–511.

[23] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

[24] Chih-Chung Chang and Chih-Jen Lin, "Libsvm: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, pp. 27, 2011.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei, "Imagenet large scale visual recognition challenge," 2014.

[26] Ariadna Quattoni and Antonio Torralba, "Recognizing indoor scenes," 2009.

[27] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio 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.