Exploring Features for Classification with Accuracy Guarantees

Jonathan Krause
Computer Science Department, Stanford University
Stanford, CA
jkrause@cs.stanford.edu

Abstract

In this work we explore the choice of features in the task of classification with accuracy guarantees. That is, given a semantic hierarchy and the ability to predict non-leaf nodes, what is the most informative set of features to use? We examine dense SIFT, LBP, BRIEF, and Object Bank features and investigate how to combine them in order to obtain a classifier that does as well as possible on this non-traditional classification task. We also investigate the advantages of using color and, in the end, beat a strong dense SIFT with LLC pooling and SPM baseline. Our features are evaluated on a subset of ImageNet consisting of 57 categories with hundreds of images per class.

1. Introduction

Object classification is one of the central problems in computer vision. To date, slow but steady progress has been made on the problem. However, in order to actually make a useful system out of today's classifiers, a different approach must be taken. Indeed, it is too much to ask for perfect classification accuracy at a detailed level from any system currently available today. However, what would happen if we were to relax the constraint that classifiers give detailed information about the type of an object – is it possible to create a classification system that has arbitrarily high accuracy at the expense of reducing the amount of information contained in a classification response?

The answer to that question is a resounding "yes". The key insight to make is that real-world objects implicitly form a semantic hierarchy and that, for any given class, it is also correct to say that it is a member of any of its ancestors in the hierarchy. For example, a Persian Cat is also a Cat, a Mammal, an Animal, and an Entity. In fact, everything is an Entity! This leads us to a better question: is it possible to maximize the *amount of information* contained in a prediction subject to an *arbitrary* accuracy constraint?

An informative and accurate classification system would have major implications in the field of large-scale classification. Current state-of-the-art at 10,000-way classification is a paltry 16.7% [17]. However, consider the following: can a random human remember or even be aware of all 10,000 classes? This is doubtful, when such classes can include such obscurities as "trimaran", "Egyptian cat", "Rhodesian ridgeback", and "jelly fungus", as is the case in the most prominent large-scale dataset, ImageNet [5]. In the absense of super-human accuracy, then, a reasonable compromise would be to provide *correct* information, even if it be not as detailed as other *wrong* information.

Fortunately, the answer as to whether we can maximize information while still maintaining accuracy guarantees is also a resounding "yes". The previously developed DARTS (Dual Accuracy Reward Tradeoff Search) algorithm does just that. However, DARTS is somewhat of a meta-algorithm – its inputs are posterior probabilities obtained in a completely free way, and in return it maximizes expected information gain subject to an arbitrary accuracy requirement. When viewed in this light, there is clear room for improvement in obtaining posterior probabilities. More specifically, we can decompose the problem of finding posteriors into two steps: (1) obtaining discriminative features, and (2) converting features into probabilities. Problem (2) can be reasonably solved using *e.g* Platt scaling [16]. (1) is harder. Note that we demand that features be *discriminative*; it is easy to generate probabilties that are perfectly accurate by simply giving all images the exact same feature representation! Such an approach is of no use. To this end, in this work we push forward in *obtaining discriminitive features for the problem of classification with accuracy guarantees*.

In total, our contributions in this paper are an examination of different baseline features, including the dense SIFT [12] that DARTS used, LBP [15], BRIEF [2], and Object Bank [11], an evaluation on the effects of color information on a subset of these baselines, and three different boosting schemes, with the end result that we beat the dense SIFT baseline both in terms of flat accuracy and also when considering information gain in the classification task with accuracy guarantees.

2. Related Work

The task of classification with accuracy guarantees has its roots in hierarchical classification [18, 8, 9] and classification with a reject option [1]. The major distinction is that no prior work optimizes the information/accuracy tradeoff, which is what the DARTS algorithm does. It is also related to large-scale classification [4] and fine-grained classification [13] in that as the hierarchy gets very semantically dense leaf classes become harder to classify but optimizing the tradeoff remains feasible. In fact, with a very dense hierarchy the internal (non-leaf) nodes become more semantically meaningful, in a sense, which only helps in optimizing information gain.

Choosing discriminative features is an active area of research in computer vision. The yearly PASCAL challenge [7] regularly showcases state-of-the-art features and boosting techniques. Many modern systems include one of either SIFT [12] or HOG [3] features. There also exists a variety of texture-base features [14] which are complementary to SIFT and HOG in that they do not explicitly consider gradients, thus making them promising candidates for boosting.

3. Problem Statement

Formally, let $G = (V, E)$ be a directed acyclic graph (DAG) representing a semantic hierarchy on a given set of classes. One can think of E as describing all of the 'is-a' relationships in the graph – i.e. $x_1 \to x_2 \in E$ if class x_2 is a subclass of x_1. Although G need not be represented using this 'is-a' relationship, it is perhaps the most natural choice of relation to choose in a semantic hierarchy.

Let $X \sim \mathcal{X}$ be a random variable correponding to the feature representation of an image and the corresponding distribution over such representations. Let $Y \sim \mathcal{Y}$ be the ground truth leaf-node (i.e. most specific) label of X. Define $\pi(Y)$ as the path of nodes going up the DAG from the leaf node Y. For example, if Y is "German Shepherd", then $\pi(Y)$ might include "dog", "mammal", "animal", and "entity". In this formulation, only the nodes in $\pi(Y)$ are considered to be a correct classification – i.e. only *correct* information is deemed valuable; no matter how close the node of an inaccurate prediction is to any node in $\pi(Y)$, if it is not actually *in* $\pi(Y)$, it is not correct.

Let $f : \mathcal{X} \to V$ be an arbitrary classifier. Define

$$\Phi(f) = \mathbb{E}[f(X) \in \pi(Y)] \tag{1}$$

to be the accuracy of f. Let $r : V \to \mathbb{R}^+ \cup \{0\}$ be an arbitrary non-negative reward function, and denote $r_v = r(v)$ for convenience. For our reward function, we shall use information gain (the decrease in entropy between the prior and posterior distributions), the it should be emphasized that $r(v)$ is *completely arbitrary*. The expected reward is

$$R(f) = \mathbb{E}\left(r_{f(X)}[f(X) \in \pi(Y)]\right) \tag{2}$$

Let $\epsilon \in [0, 1]$ be a specified maximum allowed error rate, so that we must have accuracy at least $1 - \epsilon$. The optimization problem, then, is

$$\begin{array}{ll} \underset{f}{\text{maximize}} & R(f) \\ \text{subject to} & \Phi(f) \geq 1 - \epsilon. \end{array} \tag{OP1}$$

Let f^*_ϵ be an (optimal) solution to OP1. Then it is easy to show that $R(f^*_\epsilon)$ is non-decreasing in ϵ, simply by noting that, for $\epsilon > \epsilon'$, $f^*_{\epsilon'}$ is a (not necessarily optimal) solution to OP1. This also makes intuitive sense: the stricter the accuracy guarantee, the less flexibility a classifier has in optimizing reward.

This optimization is solved under practical conditions by the DARTS algorithm. However, there is a less-obvious problem lurking within OP1 – namely, the entire problem setup assumes that X is already given in terms of features, *not* in terms of a raw image. OP1 optimizes over the classifier, but leaves the choice of feature representation completely open. With that in mind, the problem addressed by this paper is this: *What is the most informative feature representation X for an image?*

4. DARTS Algorithm

For completeness, we give a brief description of the DARTS (Dual Accuracy Reward Trade-off Search) algorithm, which, under practical conditions, is optimal for OP1. DARTS is a primal dual algorithm based on the generalized Lagrange multiplier method [6]. The Lagrangian is

$$L(f, \lambda) = \mathbb{E}(r_{f(X)} + \lambda)[f(X) \in \pi(Y)] + \lambda(\epsilon - 1) \tag{3}$$

For a particular λ, f_λ maximizing the Lagrangian is given by

$$f_\lambda(x) = \arg\max_{v \in V} (r_v + \lambda) p_{Y|X}(v|x) \tag{4}$$

with $p_{Y|X}(v|x) = \Pr(v \in \pi(Y)|X = x)$. Also define $r_{max} = \max_v r_v$ and let $\hat{v}$ be the root of G. Let $\tilde{\epsilon}$ be some tolerance for how close $\Phi(f)$ is allowed to get to $1 - \epsilon$ upon completion; i.e. $\Phi(f) - (1 - \epsilon) \geq \tilde{\epsilon}$ upon termination.

With all definitions in place, the DARTS algorithm is given in Algorithm 1.

5. Features

The choice of image representation is absolutely critical for any classification task. Given a sufficiently discriminative representation, classification can be trivial, and given a poor representation, a classifier can be reduced to random guessing. Current state of the art computer vision features are not perfect, which means that considerable

Algorithm 1 DARTS

1. Obtain $p_{Y|X}(y|x), y \in \mathcal{Y}$.
2. $p_{Y|X}(v|x) \leftarrow \sum_{y \in \mathcal{Y}} [v \in \pi(y)] p_{Y|X}(y|x), \forall v \in V$.
3. $f_0 \leftarrow \arg\max_{v \in V} r_v p_{Y|X}(v|x)$.
4. If $\Phi(f_0) \geq 1 - \epsilon$, return f_0.
5. $\bar{\lambda} \leftarrow (r_{max}(1 - \epsilon) - r_{\hat{v}})/\epsilon$.
6. Binary search for a $\lambda \in (0, \bar{\lambda}]$ until $0 \leq \Phi(f_\lambda) - 1 + \epsilon \leq \tilde{\epsilon}$, for a maximum of T iterations. Return f_λ.

thought should be put into choosing features which maximize performance. The original DARTS algorithm used densely-sampled SIFT features, combined with Locality-constrained Linear Coding (LLC) [21] without exploring other options. Although these features are both strong and efficient, they are not the be-all and end-all representation of images, which is a sufficient reason to consider other features. In addition, since OP1 is *not* the traditional flat (in the sense of occupying only the bottom/leaf level of G, which includes only nodes of a single height) classification challenge, it may be the case that the most discriminative features in the flat classification challenge are not also the reward-maximizing features for OP1. Intuition suggests that the two problems go hand-in-hand, but it is worth it to question this assumption.

5.1. Dense SIFT

Dense SIFT features, as their name suggests, consist of sampling SIFT features [12] in a regular grid over an image, optionally at multiple scales or resolutions. This is the technique used in the original DARTS algorithm. They use dense SIFT features as implemented by [20], as do we.

5.2. Dense SIFT w/Colors

Regular dense SIFT features only operate on a grayscale image. This trivial extension operates on each of the RGB levels separately, and merely concatenates the features at each point in the dense grid. Using color representations other than RGB is certainly possible, but we shall focus on RGB here for simplicity.

5.3. Local Binary Patterns

Local binary patterns (LBP) have shown to be a robust descriptor for texture classification [15]. Given a 3×3 grayscale image patch, they work by comparing the intensity of the center pixel in the patch to each of its surrounding pixels, forming a binary vector of length 8, as illustrated in Figure 1. This can be generalized to patches larger than

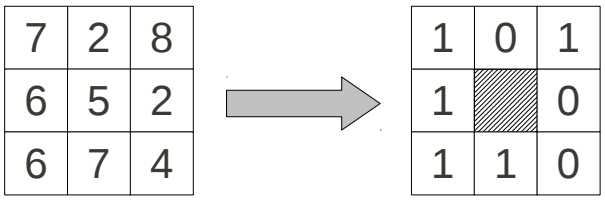

Figure 1. An illustration of the conversion of a 3×3 image patch into LBP features.

3×3, either by doing comparisons to more than just the immediate surrounding pixels or by aggregating these local binary vectors into a larger vector. The latter approach is the one taken by [20], which is the implementation we use.

Like dense SIFT features, LBP can also be trivially adapted to operate on each of the RGB layers separately.

5.4. BRIEF

Binary Robust Independent Elementary Features (BRIEF) [2] are a more modern generalization of LBP. Like LBP, BRIEF descriptors are essentially comprised of many comparisons, concatenated into a single binary vector (which occupy very little memory, hence the name BRIEF). Unlike LBP, they do not simply use the surrounding pixels for comparisons. Rather, BRIEF descriptors pick arbitrary pairs of pixels within each patch (illustrated in Figure 2), using the same locations between patches. [2] compares several different methods for choosing these pairs of points. Interestingly, the method that most closely resembles LBP performs worst! We use the second method in [2], in which points within a patch are choosen by

$$(\mathbf{X}, \mathbf{Y}) \sim \text{i.i.d.} N(0, \frac{1}{25} S^2) \tag{5}$$

where S is the sidelength of an image patch, each patch is centered at 0, and points falling outside the patch are clamped to the edges. As usual, incorporating color is also easy. We use our own implementation of BRIEF descriptors.

5.5. Object Bank

Object Bank (OB) [11] is a recognition system targeting scene classification. It consists of a collection of pre-trained object detectors, then uses SPM and a variety of sparsity-inducing regularization techniques to combine the outputs of each of the detectors. This allows it to obtain state-of-the-art performance on a number of scene recognition datasets. Although we do not consider scene classification in our work here, it is an interesting experiment to see whether their performance in scene classification carries over into object classification and whether OB features behave in any significantly different way when working with

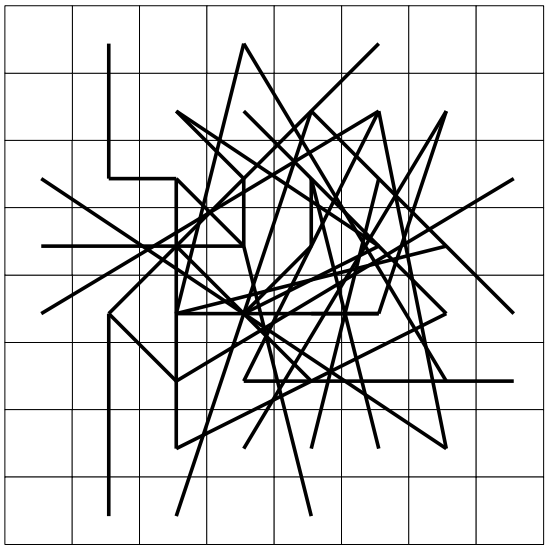

Figure 2. An example of pairs of points selected for comparisons when making BRIEF features.

accuracy guarantees. In any case, testing OB is relatively simple due to the implementation provided by the authors.

5.6. Boosting

Boosting is an incredibly generic and powerful class of meta-algorithms that combine some amount of (weak) classifiers into a single strong classifier. Boosted classifiers routinely win the PASCAL challenge [7]. Thus, if we desire to improve upon the SIFT baseline, boosting is a very natural technique to use. We must be careful with how we approach the matter, though. We can combine features at potentially any phase of the classification pipeline – from before we even make a codebook, including using separate codebooks for two different features, to simply voting with classifiers at the end.

We explore three such options here: combining features (SIFT and BRIEF) from the very beginning when doing a dense sampling, using a variation of multi-class AdaBoost [22] with all four of our classifiers, and training an SVM using the classification values of the other classifiers as features. The slight twist to AdaBoost that we use is, rather than combining the classifiers in a way that uses only the actual class output, we combine them using their (normalized) prediction values for each class. This lets us obtain overall prediction values for all the classes on each example, which is necessary for the DARTS algorithm to run, although it is somewhat of a hack because the weights used on the classifiers are determined (via AdaBoost) solely by the class they predict, *not* their prediction values.

The rationale behind using SVM outputs as inputs into another SVM is that not only allows us to take into account *what* the output of the individual classifiers was, it also takes into account the *confidence* of each of them, as represented by the classification values themselves (i.e. the distances from the margin). Contrast this with AdaBoost, which only accounts for the binary output of each of its input classifiers. Since our classifiers are relatively strong (compared to e.g. decision stumps), we expect that traditional AdaBoost would not perform as well as explicitly taking into account the actual decision values. Note that, in any case, appropriate normalization must be done so that undue influence is not given to any set of features without just cause.

5.7. LLC

Locality-constrained Linear Coding (LLC) [21] is a state-of-the-art coding scheme which not only allows for descriptors to be represented as a linear combination of multiple codewords, unlike hard vector quantization, but also explicitly encourages the selected codewords for a particular descriptor to be close to the descriptor in feature-space. The optimization problem solved by LLC coding is:

$$\begin{aligned} &\min_{\mathbf{C}} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{B}\mathbf{c}_i\|^2 + \lambda \|\mathbf{d}_i \odot \mathbf{c}_i\|^2 \\ &s.t.\, \mathbf{1}^T \mathbf{c}_i = 1, \forall i \end{aligned} \tag{6}$$

where $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_N]$ is the coded version of each all of the descriptors, $\mathbf{x}_i$ is a descriptor feature, $\mathbf{B}$ is a codebook, $\mathbf{d}_i$ describes the distance between $\mathbf{x}_i$ and each of the codewords, and $\odot$ is element-wise multiplication. The term $\mathbf{d}_i$ is useful as it directly forces the encoded descriptors to be represented with nearby codewords. Otherwise, it might be the case that we could end up with a representation consisting of a deceptive mixture of codewords. For example, if a we were trying to encode the number 10 using the codebook $\{-100, 5, 20, 120\}$, it would be much better to use a mixture $\frac{2}{3} \cdot 5 + \frac{1}{3} \cdot 20$ than to use $\frac{1}{2} \cdot -100 + \frac{1}{2} \cdot 120$, since 5 and 20 are much more representative of the number 10 than either -100 or 120. This example is very much artificial, being one-dimensional, but the idea easily extends to higher dimensions.

6. Dataset

We use the ILSCVRC65 dataset, illustrated in Figure 3. It consists of 57 leaf classes and 8 internal nodes. For each class, there are 100 training images, 50 validation images, and 150 test images, for a total of 17,100 images in the entire dataset. All images were drawn from ImageNet [5]. Note that the Dog subtree was intentionally made large in order to mimic types of skewness present in the full ImageNet dataset.

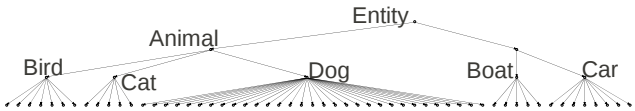

Figure 3. The tree structure of ILSVRC65.

7. Experiments

In all cases in our experiments (except for OB, which has its feature-producing code provided with it), we densely sample the descriptors at multiple resolutions, use LLC for coding with a codebook size of 10,000, and use two levels of spatial pyramid (1×1 and 3×3) [10], which results in features of size $10,000 \cdot (1 + 3^2) = 100,000$. Fortunately, LLC pooling gives us the benefit of sparsity, which, combined with judicious regularization, prevents overfitting. In all cases except for OB and the boosted methods, we use ℓ_2-normalized linear 1-vs-all SVMs as our classifiers. We use Platt scaling [16] to obtain posterior probabilities, as required by the DARTS algorithm. Platt scaling is done on a set of cross-validated decision values from the training set, using 10 folds. The validation set is used to determine $\lambda^\dagger$, the optimal value of the dual variable λ. For all non-OB features, we also get features in each component of the RGB space separately.

For OB, following the lead of [11], we use ℓ_1-normalized logistic regression. This shifts the gains of sparsity from the features to the classifier. This is also the best-performing method for BOOST-SVM.

The features are summarized in Table 1. Unfortunately, there seems to have been an as-yet unidentified issue in the grayscale BRIEF features or testing, which has not been examined due to time – creating one set of features for the dataset takes several hours on a cluster, and training takes anywhere from half a day to several days. A large part of this is due to the cross-validation done on the training set, which increases training time by roughly a factor of 10, but is necessary for the DARTS algorithm to work well.

Also plain just from examining the flat accuracies is that the dense SIFT baseline is remarkably strong, particularly when compared to the other non-boosted classifiers.

7.1. Effects of Color

First we examine how much performance we can gain from incorporating color information. Note first that our treatment of color is incredibly simplistic – much more complex descriptors can be made which incorporate color information [19]. Note second that, regardless of this fact, in both the SIFT and LBP cases including RGB information boosts flat accuracy by 1-3%. When transferred into the setting with accuracy guarantees, this relation holds, as illustrated by Figures 4 and 5. Plotted there is test accuracy vs information gain, normalized to be in $[0, 1]$, where each data point is determined by choosing the parameter set that performed best on the validation set and is evaluated on the test set. Also note that we determine $\tilde{\epsilon}$ by computing a 95% confidence interval around the validation accuracy in order to ensure that the test accuracy is above the guarantee $1 - \epsilon$ with high probability.

Feature	Size	Density	Flat Acc.
SIFT	100k	.2869	.3871
SIFT-RGB	100k	.2770	.4027
LBP	100k	.3014	.2765
LBP-RGB	100k	.2632	.3164
BRIEF	100k	.3259	.0182
BRIEF-RGB	100k	.3360	.2531
OB	44,604	1.0000	.2608
SIFT-BRIEF	100k	.3627	.3896
BOOST-ADA	n/a	n/a	.4178
BOOST-SVM	228	1.0000	.4229

Table 1. Summary of information about features. Size is the total dimension of a feature vector for a single image, Density is the average fraction of non-zero elements, and Flat Acc. is the accuracy when restricted to the leaf nodes of the hierarchy.

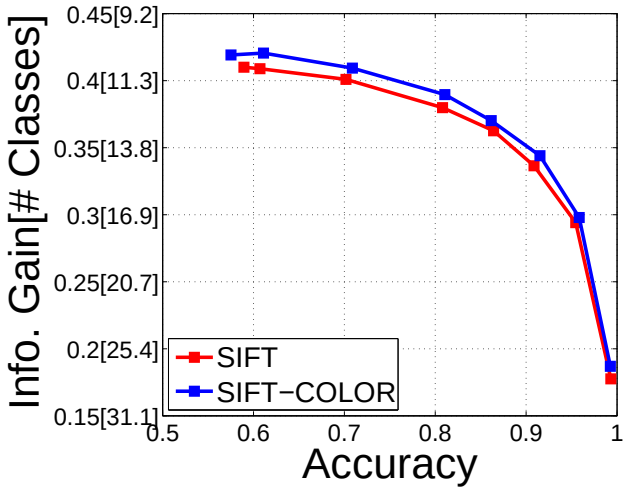

Figure 4. The effects of color on SIFT features. In brackets on the Y-axis is the equivalent number of classes that an example has been narrowed down to in the sense of expected information gain.

We can also see the effects of color by looking at how the distribution of predictions over the various node heights, displayed in Figure 6. Focusing on SIFT, there are very few differences between SIFT and SIFT-COLOR. This can partially be attributed to the only modest gains of 1.5% in flat accuracy. However, we can also see that SIFT-COLOR predicts very slightly (.65 %) more examples at the leaf level. When we turn to LBP, the differences are far more drastic. Adding color information to LBP allows it to be more confident about a significantly larger number of examples, boosting the portion predicted at the leaf level up by 5.6%.

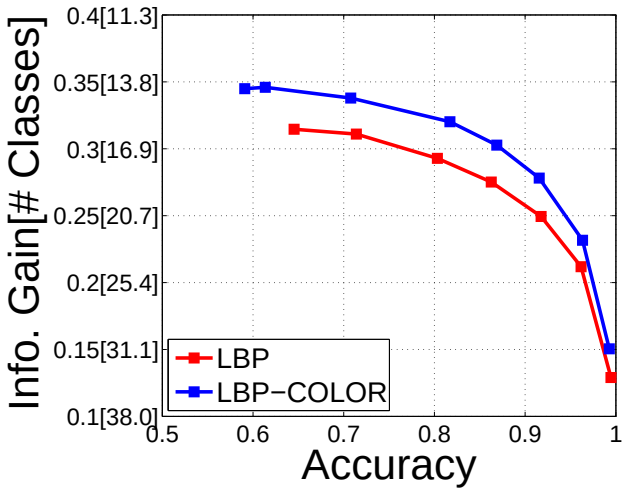

Figure 5. The effects of color on LBP features.

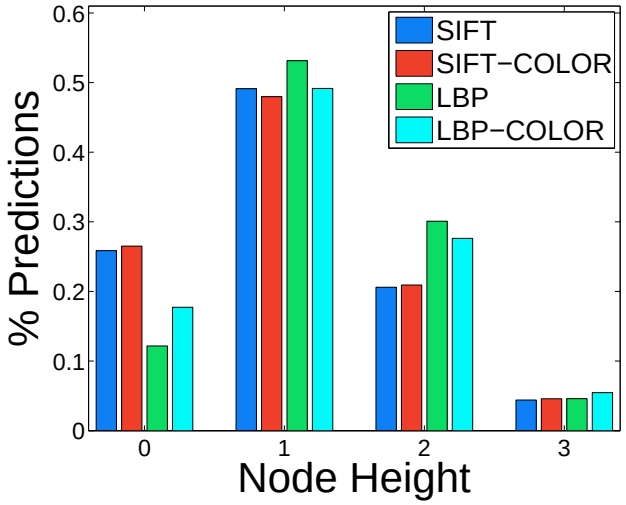

Figure 6. Histogram of node height vs portion of test examples predicted at that height, with an accuracy guarantee of .9

7.2. Comparison of Methods

Figure 7 plots all of the methods which have not been shown to be completely outclassed already (e.g. LBP when compared to LBP-COLOR). There are many interesting points we can make from this figure:

- The dense SIFT (DSIFT) baseline remains quite good. It is consistently much better than all of the other baselines and is very close to the boosted methods, indicating that boosting is serving only to add a comparatively small amount of performance.

- The comparison between BRIEF descriptors and SURF and an LBP-like desriptor in [2] does not hold here. It is possible that the implementation of BRIEF we used was too simplistic, or perhaps its parameters needed more tuning. For example, the image patches used in [2] were much larger than those used in our dense sampling. It might also be the case that the task of multiple-viewpoint matching considered in [2] is not comparable with our object recognition task. To make conclusive statements about BRIEF descriptors would require further experiments.

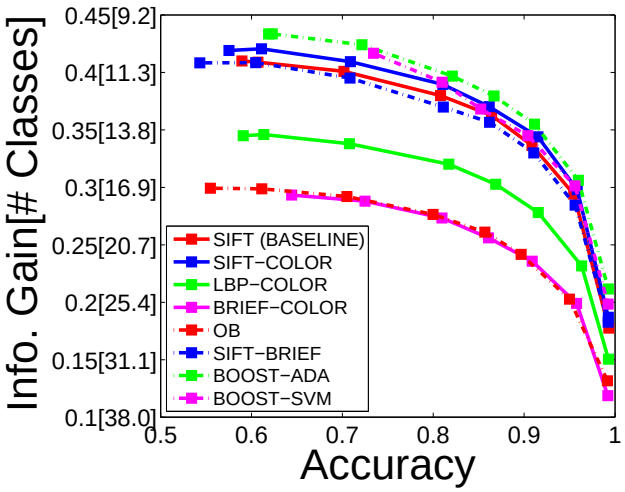

Figure 7. Accuracy vs. normalized information gain for all reasonable methods.

- Object Bank does not fair as well when considered in for object classification. To be fair, OB does have a feature vector more than a factor of two smaller than the other baselines, and adding more objects detectors to its bank may help. Also, anecdotally, OB took longer to train, likely due to the density of its features.

- Not all methods of boosting work! This is evident in the plot for SIFT-BRIEF, which is on-par to actually *worse* than plain old SIFT. It might be that the higher dimension of the raw features for SIFT-BRIEF necessitates a larger codebook or more training examples. It also could be the case that, since BRIEF-COLOR is so much worse than SIFT, BRIEF-COLOR is almost acting as noise (one hopes this is not the case, though).

- Differences in flat accuracy do not always hold when considered in the classification with accuracy guarantee framework. Case in point: BOOST-SVM has a higher flat accuracy than BOOST-ADA, but BOOST-ADA consistently has slightly higher amounts of information gain. On a similar note, this indicates that there is more work to be done in developing boosting algorithms, since determining the weights in BOOST-ADA did *not* take into account the numerical prediction values of the individual classifiers used – it only considered the final predicted classes.

Now, for completeness, we compare the distribution of predictions of our best classifier, BOOST-ADA, and the initial baseline, SIFT, in Figure 8. Some interesting trends:

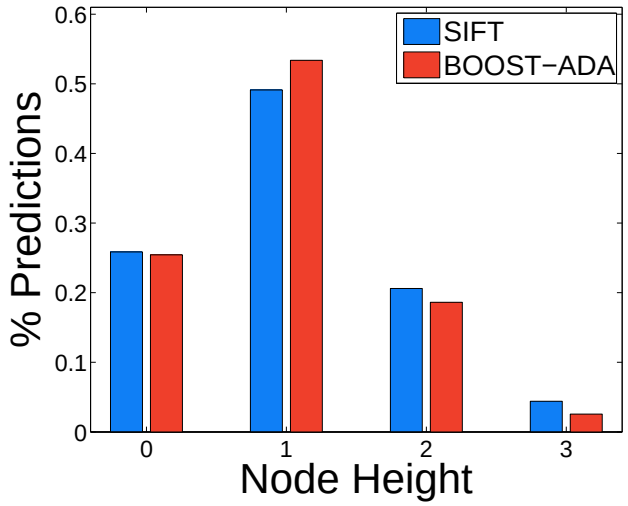

Figure 8. Histogram of node height vs portion of test examples predicted at that height, with an accuracy guarantee of .9, for the SIFT baseline and BOOST-ADA.

SIFT actually has *more* predictions at the leaf level than BOOST-ADA, despite being worse in terms of information gain. Despite this, it seems that combining several classifiers has allowed BOOST-ADA to become more confident about the examples that SIFT was not very confident at all about, as indicated by the larger portion of predictions BOOST-ADA makes at height 1 and the smaller number of predictions made at heights 2 and 3. This also serves to illustrate the importance of using a meaningful semantic hierarchy – without one, or with a hierarchy with very weak relations, BOOST-ADA would not have been able to take advantage of the information that can be gained at the internal nodes.

8. Future Work

Work on more informative visual features for classification is neverending, and there are a number of points in this work which warrant further exploration. Out treatment of color has been incredibly simple, and further gains in performance are expected with a more sophisticated approach. Furthermore, there is the unenviable task of further fine-tuning parameters, which we were unable to do to our satisfaction in this work due to the size of the dataset and the extensive training time. In particular, one simple experiment would be to vary the codebook size over a given range, comparing the types of curves obtained for the baselines with e.g. SIFT-BRIEF. Another simple method to investigate is using a kernelized SVM, especially for the approaches which use LLC features. Although this further boosts the training time and memory costs, there is reason to believe that explicitly using a histogram-interesection or χ^2 kernel can help.

Tangentially, even the choice of classifier used in the DARTS algorithm could be modified to improve performance. More specifically, SVMs use a hinge loss, but we suspect that other loss functions may be more suited for the task of classification with accuracy guarantees. For example, it could be worth it to structure a loss function such that we trade off examples in which we have only weak confidence in order to get more confidence on other examples. The key idea here is that in a setting where information gain matters, losing small bits of information on examples where we were not going to get much in the way of information gain anyway can be offset by pushing some examples down the tree to the higher-reward nodes.

More directly related to our work, we have only examined the fact *that* incorporating color and more features aids in classification, but we have not examined *where* it helps. Doing so, besides being interesting in its own right, could point to directions for further work and improvements.

9. Conclusions

In this work we have examined the problem of chosing features in the unique task of object classification with accuracy guarantees. We have looked at SIFT, LBP, BRIEF, and Object Bank features, as well as 3 very different boosting strategies and the addition of color, and improved upon the dense SIFT baseline both in terms of flat accuracy and in terms of information gain across a range of accuracy guarantees.

References

[1] P. L. Bartlett and M. H. Wegkamp. Classification with a reject option using a hinge loss. Technical report, U.C. Berkeley, 2006.

[2] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF : Binary Robust Independent Elementary Features. pages 778–792, 2010.

[3] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. Ieee, 2005.

[4] J. Deng, A. C. Berg, K. Li, and L. Fei-fei. What Does Classifying More Than 10,000 Image Categories Tell Us ? pages 71–84, 2010.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[6] H. Everett III. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research*, pages 399–417, 1963.

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.

[8] T. Gao and D. Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *ICCV*, 2011.

[9] 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*, pages 1–8. IEEE, 2008.

[10] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, 2:2169–2178, 2005.

[11] L.-j. Li, H. Su, E. P. Xing, and L. Fei-Fei. Object Bank : A High-Level Image Representation for Scene Classification & Semantic Feature Sparsification. *Machine Learning*, pages 1–9, 2010.

[12] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov. 2004.

[13] G. Martinez-Munoz, N. Larios, E. Mortensen, W. Zhang, A. Yamamuro, R. Paasch, N. Payet, D. Lytle, L. Shapiro, S. Todorovic, et al. Dictionary-free categorization of very similar objects via stacked evidence trees. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 549–556. IEEE, 2009.

[14] T. Ojala, M. Pietikäinen, and D. Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29(1):51–59, 1996.

[15] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, July 2002.

[16] J. Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[17] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1665–1672. IEEE, 2011.

[18] A.-M. Tousch, S. Herbin, and J.-Y. Audibert. Semantic hierarchies for image annotation: A survey. *Pattern Recogn.*, 45:333–345, January 2012.

[19] K. Van de Sande, T. Gevers, and C. Snoek. Evaluation of color descriptors for object and scene recognition. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. Ieee, 2008.

[20] A. Vedaldi and B. Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the international conference on Multimedia*, pages 1469–1472. ACM, 2010.

[21] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained Linear Coding for Image Classification. *CVPR*, 2010.

[22] J. Zhu, S. Rosset, H. Zou, and T. Hastie. Multi-class adaboost. *Ann Arbor*, 1001(48109):1612, 2006.

10. Appendix

1. All components of this project are computer vision components. Some of the techniques used may be generalizable to other areas of ML, but in my mind, that's an advantage.

2. Contributions in this project:

   (a) Ideas: The idea of classification with accuracy guarantees was not originally thought of by me, and I don't make that claim. Jia Deng urged me to consider boosting and focus on improving performance (which was an idea for the focus of the paper that I was already considering), but that's nothing particularly new, just an idea for the project. Andrej Karpathy mentioned BRIEF descriptors to me in passing, which made me check out the BRIEF paper and try it out.

   (b) Formulations of algorithms: The DARTS algorithm was mostly-entirely thought of by Jia Deng, who consulted me during its development. I also had several discussions about boosting with Ben Poole, though I don't think we both ended up implementing the same thing at all.

   (c) Software and coding: There was some amount of code already written (presumably by Jia) for the purpose of testing on the toy dataset. As time went on a larger and larger fraction of all the code was written by me. In terms of feature extraction, that had been entirely done by Jia until I started this project (i.e. post-CVPR paper). So there was a bit to learn there, and I used some code that he had written for extracting dense SIFT features, though that all had to be changed for each of the different features. The boosting methods were coded entirely by me. I've also indicated where I got each of the feature implementations from in the Features section of the paper.

   (d) Designs of experiments: Not too much to say here, the design is pretty straightforward – try some methods, then combine them. To that simple extent, I've done all of the work.

   (e) Analysis of experiments: Again, this has been only me. Admittedly I wish I could give them a more scientific treatment, optimizing over all parameters, etc., etc., but that would take an incredible long amount of time (see the portion of the Experiments section where I give running time estimates). On a similar note, there are always technical difficulties that have nothing to do with the experiments explicitly but always slow things down (e.g. nodes in the cluster running out of disk space, cryptic Hadoop error messages, etc.).

3. Verify and confirm that you (and your partner currently taking CS231A) are the sole author(s) of the writeup.

I'd cite the paper Jia and I worked on for CVPR, but that hasn't been published, or even accepted, technically, so I can't. It's available upon request. The primary part of that paper that I took was the DARTS algorithm, which I think is reasonable. I also copied over Figure 3, which I figure is ok since I made it in the first place.

Future Distribution Permission

The author(s) of this report give permission for this document to be distributed to Stanford-affiliated students taking future courses.