INTERACTIVE TRAINING OF PIXEL CLASSIFIERS OPENS NEW POSSIBILITIES

Justus H. Piater, Edward M. Riseman, and Paul E. Utgoff
Computer Science Department
University of Massachusetts
Amherst, MA 01003
{piater|riseman|utgoff}@cs.umass.edu

Commision III, Working Group 5

KEY WORDS: On_line Pixel Classification, Incremental Decision Trees, Object Recognition

ABSTRACT

We propose a novel interactive incremental method for pixel classifier construction. It yields very efficient decision tree classifiers and allows training of hierarchical classifiers for recognition of simple structured objects. Two novel concepts made possible by our paradigm are demonstrated: (1) *Incremental training* of a decision tree classifier on a sequence of images permits incremental, non-iterative improvement by dynamic addition of user-specified informative training pixels, i.e. pixels that are currently misclassified and will thus change the classifier. In experiments on a realistic terrain classification task, the number of training instances involved in building a classifier was reduced by several orders of magnitude, at no perceivable loss of classification accuracy. (2) *Hierarchical classification* extends the concept of pixel classification from labeling pixels directly with their categories to utilizing these class labels to describe more complex objects. We propose a set of simple and generic feature extractors that characterize spatial relationships between class labels. This makes our system capable of solving object recognition tasks beyond the realm of traditional pixel classifier systems, which is illustrated by a wildlife survey example, a seagull counting problem.

KURZFASSUNG

Wir präsentieren eine neuartige Methode zur interaktiven, inkrementellen Erstellung von Pixel-Klassifikatoren. Sie ermöglicht die schnelle und einfache Erstellung sehr effizienter Entscheidungsbaum-Klassifikatoren, sowie die Konstruktion einer Hierarchie von Klassifikatoren für die Erkennung einfacher strukturierter Objekte. Wir demonstrieren zwei neuartige Konzepte: (1) Ein Entscheidungsbaum-Klassifikator kann durch *inkrementelles Training* auf Beispiel-Bildern, die in monotoner Folge präsentiert werden, konstruiert und verfeinert werden. Der Nutzer wählt dabei interaktiv informative Trainingspixel aus, d.h. solche Pixel, die vom aktuellen Klassifikator misklassifiziert werden. Experimente anhand eines realistischen Klassifikationsproblems ergaben eine drastische Reduktion der Anzahl von Trainingspixeln gegenüber einem herkömmlichen Trainingsverfahren, ohne erkennbaren Genauigkeitsverlust. (2) *Hierarchische Klassifikation* erweitert das Anwendungsgebiet von Pixel-Klassifikatoren: Das Ergebnis einer Klassifikation wird von einem weiteren Klassifikator verwendet, um komplizierte Klassifikationsprobleme zu lösen. Wir präsentieren drei einfache generische Funktionen zur Extraktion topologischer Relationen zwischen klassifizierten Pixeln. Diese können von einem hierarchischen Klassifikator genutzt werden, um einfache Objekte zu erkennen, was wir am Beispiel von Seemöwen demonstrieren.

1 INTRODUCTION

Image pixel classification is a fundamental task in computer vision. Despite abundant applications, the construction of high-performance pixel classifiers usually involves substantial cost in terms of human effort. A traditional procedure for classifier construction is illustrated in Figure 1: A number of training instances (i.e. completely or partially hand-labeled images) are selected and passed to a classifier construction system. The resulting classifier is then evaluated, typically by comparing its output with ground truth data and assessing its accuracy. If the performance is not satisfactory, some parameters of the system are changed, such as the feature set or the training set, or the classifier construction algorithm, and the entire procedure is repeated.

The training set has a great influence on the performance of a classifier. For this reason, great effort is traditionally put into the construction of the training set. This work is concerned with efficient selection of informative training instances. In the case of image pixel classification, substantial cost is incurred by the requirement to provide correct labels for the training pixels by hand. Therefore, one would like to be able to provide a *small number* of well chosen training instances relatively quickly, at no loss of classification accuracy (or even improved accuracy, Salzberg et al. 1995).

Besides the cost of manual labeling, there are other benefits to keeping the training set small. For example, a typical decision tree classifier will attempt to place training instances of different classes in separate leaf nodes, as long as they are discernible based on their feature vectors. However, in most practical applications the distributions of the different classes overlap in feature space, which leads to overly specialized and very complicated decision trees with poor generalization properties. This is typically addressed by elaborate pruning algorithms which try to detect overspecialization and simplify a tree, in essence trading classification accuracy on the training set for improved generalization. Other types of classifiers address this problem differently, e.g. by drawing maximum-likelihood boundaries between classes in feature space. To generate optimal classifiers, such algorithms require a sufficiently large number of training instances whose distributions in feature space meet the statistical assumptions made by the algorithm. In many practical applications this requirement cannot be met.

Consequently, it would be beneficial to select a small number of *informative* training instances that are known (or thought)

to be typical representatives of their class, rather than a large number from an unknown distribution. In the case of decision tree classifiers, such a procedure should ideally eliminate the need for pruning altogether.

This raises the question of what constitutes a well-chosen training instance. If one could know where the classifier makes mistakes, one could generate an informative instance by providing a correct label for a currently misclassified pixel. This suggests an interactive tool.

We propose an interactive system for efficient construction (in terms of human involvement) of pixel classifiers. In our system, the off-line iterative procedure (Figure 1) is replaced by an on-line Interactive Teacher/Learner paradigm (Figure 2), which we call ITL. The Teacher is a human domain expert who operates a graphical user interface to select images for training and, for any image, selects and labels small clusters of pixels. The Learner is a computer program that operates through a well-defined communication interface with the Teacher's interface. The Learner can receive images and training instances, and can produce a classifier, which in turn produces labels for the pixels of the current training image, according to the most recent classifier.

A fundamental aspect of this model is that it is incremental. The Teacher does not need to provide a large number of instances that may or may not be informative. Instead, each time the user provides a new instance, the Learner revises its classifier as necessary, and begins to relabel the current training image. This lets the user see the misclassified pixels immediately, and helps him decide where additional or revised training instances are most needed, or when the set of accumulated training instances is sufficient.

This notion of incremental training makes the ITL system suitable for training on image sequences. Traditionally, the classifier construction procedure is performed using some fixed set of training and test images. Once a good classifier is found, it is retained and expected to perform well on previously unseen images. With ITL, in contrast, fixed training and test sets are not required: Training is started on the first image and continues until the result is satisfactory, and the image might never be considered again. When the next image comes in, training resumes, until good classification has been achieved. This process is continued as necessary. Assuming that the images in the given class are sufficiently similar – as must be the case in any classification task – this training procedure is expected to converge in the sense that less and less training is required to achieve satisfactory performance, as more images are seen. An example of this procedure is presented and quantitatively evaluated in Section 3 below.

The interactive nature of training can be exploited to allow the user to convey – by giving examples – more knowledge into the classifier construction process than any reasonable

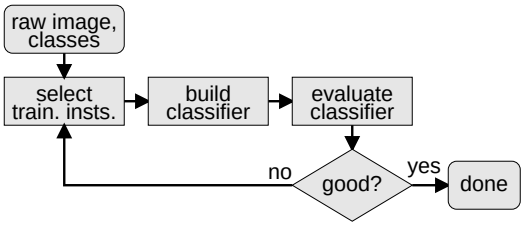

Figure 1: Traditional classifier construction.

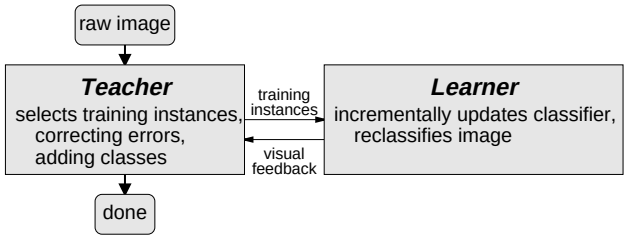

Figure 2: Interactive, incremental classifier construction.

feature set can express. This idea led to the concept of hierarchical classification, which is presented in Section 4.

2 CLASSIFIER ISSUES

The Interactive Teacher/Learner paradigm requires a learning algorithm that can classify pixels quickly (thousands of pixels per second), and can update its classifier quickly as new instances become available from the Teacher. If the classifier is updated slowly, the user will experience delay in seeing the effects of training, which slows down the process of correcting misclassified points, and basically defeats some of the advantages of the interactive process. More importantly, the various costs associated with a complex classification problem (time, human labor, psychological burden) are reduced when the user rapidly gets to see the effect of the training examples that are incrementally supplied to the classifier construction algorithm. Thus, reducing the feedback delay between selecting a training example and obtaining the classification results within a current region of interest is essential.

This work is primarily concerned with effective selection of training instances. Another important issue in classifier construction is the definition of a feature set. It is known that increasing the size of a feature set can adversely affect classifier performance (Devijver and Kittler 1982). Selection of an optimal feature subset from a given universe of features has been shown to be infeasible in practice (Ferri et al. 1994). Classifiers that utilize the entire feature set (such as neural networks, nearest-neighbor clusterers, linear machines) are particularly sensitive to redundant and noisy features. This motivates the use of a *decision tree* classifier which consults only a single feature at each decision node. Only *informative* features are incorporated into the tree, and features of little discriminative power are disregarded entirely. "Informative" here refers to the ability of the classifier to classify the training set correctly. One is still left with the problem of selecting representative training instances that will cause the tree induction algorithm to select those features that will result in good generalization. Thus, we have not solved the feature selection problem, but by employing an interactive decision tree paradigm we can address these issues in terms of training instance selection.

We employ the incremental decision tree inducer ITI (Utgoff 1994; Utgoff et al. 1997). ITI revises its tree incrementally, meaning that it can accept and incorporate training instances serially without needing to rebuild the tree repeatedly. The algorithm builds the same tree for the same accumulated set of training instances, regardless of the order in which they are received. Furthermore, the algorithm maintains data structures that allow it to revise its tree dynamically without reinspecting the training instances.

In this work, we are not concerned with some of the issues

that are important within the machine learning research community. Our goal is to allow the user to construct a reasonably small tree (to produce a fast classifier with good generalization properties) through an acceptably small amount of training. If a tree were produced that was a little larger than might otherwise be necessary, and required a few more training examples than might otherwise be necessary, but were produced much more quickly, then the overall task of producing the component pixel classifier nevertheless might still be accomplished more cheaply, due to less wasted time.

Whenever the Learner is done updating its tree, it begins to classify pixels of the current training image until it has classified the entire image, or until a new tree is available. To reduce feedback delay, this classification process begins at the location of the latest mouse click, and continues in the shape of a growing square centered at this location. This gives the user immediate feedback at the location which is likely to be most important to him. In practice, the user will have to wait only a few seconds until she or he detects new candidates for informative training instances. Our experiments suggest that a growing-radius scheme like this is the key to operating on images of virtually unlimited size in interactive time.

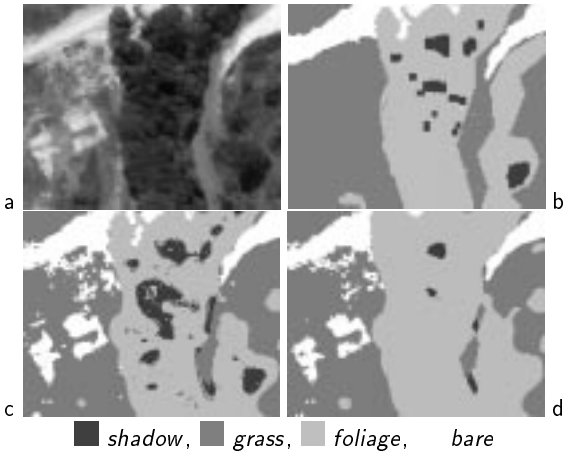

Figure 3: (a) Subimage (250×200 pixels) of Ft. Hood scene; (b) manually generated ground truth (note the difficulty of this task on this particular subimage); (c) classification results generated by an ITI classifier built in traditional batch mode using Wang et al.'s training data; (d) classification results generated by an interactively trained ITI classifier.

3 QUANTITATIVE RESULTS

In this section, we compare our ITL system with a previously published classification result by Wang et al. (1997). We chose this example because it uses state-of-the-art techniques, the task is realistic, and their data include ground truth.

Wang et al. considered a monochromatic aerial image of a rural area in Ft. Hood, Texas (Figure 3). The goal was to build a pixel classifier to recognize the four terrain classes *bare ground* (road, riverbed), *foliage* (trees, shrubs), *grass*, and *shadow*. Their most effective feature set consisted of 12 *co-occurrence features* (angular second moment, contrast, and entropy at four angular orientations each; Haralick et al. 1973), four *three-dimensional features* introduced by Wang et al., and the intensity. The co-occurrence features employed have previously been claimed to be highly effective for classification (Conners and Harlow 1980; du Buf et al. 1990; Ohanian and Dubes 1992; Weszka et al. 1976). The 3D features are generated during stereo processing of a calibrated image pair (Schultz 1995) and were recently shown to be highly discriminative in this task. The Foley-Sammon transform (FST; Foley and Sammon 1975) was employed as a classifier. FST is a linear discriminant method that is considered effective (Liu et al. 1993; Weszka et al. 1976).

As a training set, Wang et al. used four homogeneous square regions of different sizes: 99×99 (*foliage*), 75×75 (*grass*), 37×37 (*bare*), and 11×11 (*shadow*). This was one of their best training sets found after extensive experimentation. The 16916 training pixels constitute less than 1% of the entire image (1,936,789 pixels). Ground truth was generated by hand. For more details, refer to Wang et al. (1997).

We generated a baseline of the performance of ITI with respect to FST on this task by running ITI in conventional batch mode on the same input data as described above, using the full training set of 16916 pixels (Table 1). ITI achieved a classification accuracy of 85.9%, outperforming FST.

To give a first impression of an ITL classifier when trained on an image sequence, we split the image into ten subimages of size 400×400 which we shuffled to obtain a sequence. Since all of the subimages stem from the same image, they are likely to be more similar than images from a real sequence. On the other hand, local terrain characteristics vary enough across the large spatial extent to demonstrate our point.

A classifier was then interactively trained on one subimage at a time: Initially, the Learner knows nothing, i.e. the classifier is empty. Each mouse click creates a single training instance. The learner receives it, updates its classifier, and begins to reclassify the current image. As soon as the Teacher observes misclassifications, she/he can choose to provide another training example. This procedure is repeated until the Teacher is satisfied with the Learner's performance. Then, training continues with the next subimage.

For later analysis of the training process, each decision tree updated as the result of a mouse click was saved to a file. At the end of the session, the saved decision trees were used offline to evaluate the accuracy of the decision tree after each mouse click by comparing the classification results on the entire image with Wang et al.'s ground truth data. The results for two separate training sessions (using different subimage permutations) are plotted in Figure 4.

In both cases, excellent classifier accuracy was achieved after very few mouse clicks. Most training was performed on the first few images. In subsequent images, little or no corrections were necessary. Furthermore, the amount of change in the learning curve introduced by the application of a single training pixel decreases with training, as is to be expected. Both observations indicate well-behaved convergence of this incremental training procedure. This needs to be confirmed on actual video image sequences.

However, the accuracy did not increase monotonically, and training on an atypical subimage can decrease the overall accuracy. This is the case for one particular subimage which appears as #6 in Sequence 1 and as #2 in Sequence 2. Notice in Figure 4 that this subimage is the only one where training did not result in a local improvement of the full image classification accuracy.

FST:	*shadow*	*grass*	*foliage*	*bare*	total
gt-*shadow*	13.8	0.0	27.8	0.2	41.8
gt-*grass*	0.0	468.6	202.7	11.8	683.0
gt-*foliage*	2.0	18.0	995.7	2.8	1018.6
gt-*bare*	0.0	33.9	21.4	138.1	193.4
total	15.8	520.6	1247.5	152.9	1936.8
correctly classified pixel total:					1616.1
overall classification accuracy %:					83.4

ITI:	*shadow*	*grass*	*foliage*	*bare*	total
gt-*shadow*	31.6	0.0	9.8	0.0	41.4
gt-*grass*	1.8	565.4	82.7	27.5	677.4
gt-*foliage*	26.9	86.6	881.9	13.6	1009.0
gt-*bare*	0.5	18.1	3.9	169.6	192.1
total	60.8	670.0	978.4	210.7	1919.9
correctly classified pixel total:					1648.5
overall classification accuracy %:					85.9

Table 1: Contingency tables of classification results using Wang et al.'s training data. Numbers represent pixels in 1000's. Rows show ground-truth numbers, columns classifier results. Top: FST, bottom: ITI.

	interactive		*batch*
# Training instances:	14	22	16916
% correct:	85.6	85.2	85.9
# tree nodes:	9	13	181
max. tree depth:	3	3	23
# features used (of 17):	4	4	16

Table 2: Comparison of classification results by different ITI classifiers. The interactively trained classifiers are from Sequence 1 (14 training pixels) and Sequence 2 (22 training pixels). The batch-generated classifier used the training data from Wang et al.

Table 2 compares the two interactively trained ITI classifiers (sequences 1 and 2) with a traditionally constructed ITI classifier (using Wang's training data). All three perform equally well; the differences in accuracy are negligible. Striking, however, is the simplicity of the interactively trained classifiers: Both decision trees are extremely small and consult only four features out of the total library of 17 features. On the other hand, the batch-trained ITI classifier had to account for a large number of exceptions to the simple rules found by the simple classifiers. The effects of such overspecialization are illustrated in Figure 3, which shows subimage classifications created by the batch-generated ITI classifier and the (interactively trained) ITI classifier from Sequence 1 after 14 training examples. The batch-generated classifier, likely overtrained by the large number of training examples, produced a more cluttered result than the interactively trained classifier.

Overtraining is a general problem that affects most kinds of classifiers. In the context of decision trees, this problem is typically addressed in the machine learning research community by pruning algorithms that try to reverse the effects of overspecialization. Our experiments demonstrate that such overspecialization can be effectively avoided by selecting training instances *in an informed manner*. This is one example of dramatically increased classifier quality achieved by our Interactive Teacher/Learner approach.

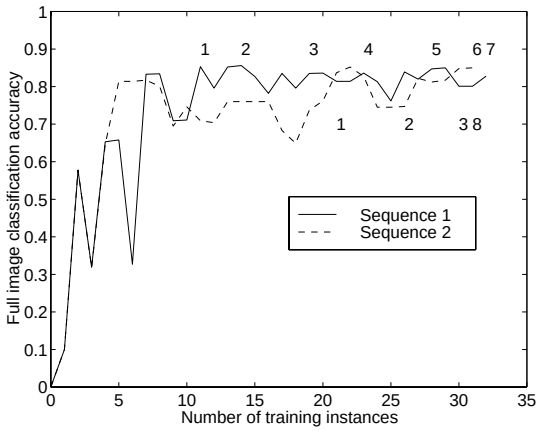

Figure 4: Classifier behavior during interactive training. Digit k next to a curve marks the last training pixel supplied while training on subimage k. In Sequence 1 (top digits), no training instances were supplied for subimages 8–10; in Sequence 2 (bottom digits), none were supplied for subimages 4–7 and 9–10.

4 HIERARCHICAL CLASSIFICATION

Traditionally, the utility of pixel classification is limited by the locality and simplicity of the underlying features. For example, it is not usually practical to cast an object recognition task purely as a classification task and train a classifier to mark all pixels which are part of a structured object, e.g. a building. This would either involve a small set of carefully designed features – which in effect implement the object recognition task – or a large set of simpler features powerful enough to express the context-sensitive spatial features that characterize a chair. The latter case constitutes a combinatorial search space too large to construct, let alone search.

However, in many applications of classification systems, the classes involve some known structure, such as the distribution, size, and/or shape of pixel classes. Such knowledge is often incorporated by hand into a classification system. Some of the most powerful expert systems for classification tasks were built this way (e.g. Slaymaker et al. 1996). In contrast, we now present an example-based method for user-defined incorporation of structural knowledge into classification-based systems trained interactively and incrementally.

Many objects can be characterized by a simple combination of straightforward structural features. For example, in a wildlife survey with our Forestry department, black-back seagulls on aerial images appear as adjacent white and dark blobs of certain sizes, representing the head and back of gulls. Often the white tip of the tail can be seen as well (Figure 5). If represented as a single feature set, these relationships would cause it to grow combinatorially, since the features required to recognize "white" and "dark" would have to be crossed with those for characterizing the sizes and spatial relationships.

The idea of interactive hierarchical classification is to avoid these combinatorics by putting the burden of combining features on the human Teacher. While a human will not generally find an optimal solution, in many cases good feasible solutions are apparent. Consider the enlarged gull in Figure 5. The spatial relationships between blobs of colored pixels that characterize a gull can be exploited by hierarchical classifica-

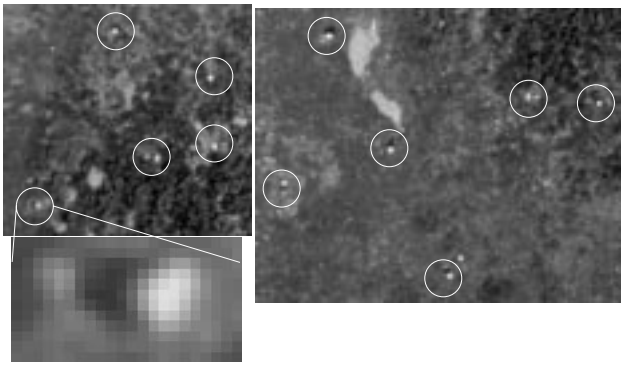

Figure 5: Aerial photographs showing sections of an island off the coast of Maine (original images in color). White circles indicate black-back seagulls. The left image (275×258 pixels) was used as a training image, the right one (405×328 pixels) as a test image. One gull is shown enlarged to reveal its white head (lower right), white tail (upper left) and black back.

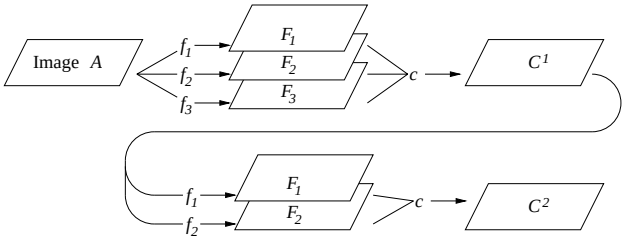

Figure 6: The concept of hierarchical classification: The output of one classification system serves as the input to another one. This can in principle be extended to any number of hierarchies. The feature extractors f typically differ from level to level, and a unique classifier c is trained at each level.

tion: First, a classifier is trained to recognize blobs of colored pixels, until all relevant blobs are recognized, even at the expense of many false positives. Then, a second classifier is trained to recognize spatial relationships between the blobs found by the first classifier. The input to the first classifier are features of the original image, and the input to the second classifier are features of the label image generated by the first classifier.

We now describe this procedure in detail (cf. Figure 6). To formalize traditional pixel classification, let an image be defined as a scalar function $A(i,j)$, mapping coordinates to intensities. Ignoring boundary problems, we can define a *feature image* as a function $F(A;i,j)$, where each element is computed as some function f of the gray values in a local square image region of radius k_f, centered at (i,j). Typically, many such feature images will be computed for a given image.

A pixel classifier c takes the l features $F_1,\ldots,F_l$ of a pixel at (i,j) and returns a class label. The resulting *label image* is denoted C with

$$C(F_1,\ldots,F_l;i,j) = c\big(F_1(A;i,j),\ldots,F_l(A;i,j)\big).$$

In hierarchical classification, we use the label image C as input to another set of feature extractors. A superscript from now on indicates the classification hierarchy: At the base level, we let $C^0 = A$, and if we provide a suitable set of feature sets – one for each hierarchy level – we can define hierarchical classification as

$$C^{h+1}(F_1^h,\ldots,F_{l^h}^h;i,j) = c^{h+1}\big(F_1^h(C^h;i,j),\ldots,F_{l^h}^h(C^h;i,j)\big)$$

for $h \geq 0$, which says that the features for c^{h+1} classifier are computed from the classification results of the c^h classifier. In general, c^{h+1} may depend on any c^m classifier with $0 \leq m \leq h$.

This hierarchy of classifiers is trained from the bottom up, yielding classifiers $c_1, c_2, \ldots$. Since the required characteristics of c^{h+1} depend on the exact qualities of c^h, this concept relies on interactive training: Informative training examples at one level are selected depending on the classification results from the previous level.

The base level (c^1) classification process is conventional in that its input features are computed from an ordinary image. The higher level classifications are different in that their inputs consist of label images generated by lower-level classifiers. The pixel values of label images do not represent cardinalities but merely categories. It is common to compute features – numerical or categorical – from numerical data for classification. Here, however, we want to express spatial relationships between categorical labels in terms of features for classification. This requires the construction of special features. To develop features capturing spatial context, we introduce the following window functions:

- To measure the *size* of a blob of labels, we count the number of occurrences of a given label M:
$$F_M(i,j) = |\{(x,y) \mid C(i+x, j+y) = M\}|$$
- To assess *adjacency* of two blobs, use the product of the number of occurrences of two given labels M and N, i.e. $F_{M,N} = F_M F_N$. This returns a high value if and only if large numbers of both class labels are present near the current pixel (i,j).
- The previous two features involve just counts and ignore spatial coherence. The feature F_c provides a measure of *clutter* by counting the number of label discontinuities between horizontally and vertically adjacent pixels within a window.

To illustrate an application of hierarchical classification, we trained a classifier to recognize black-back seagulls on color aerial images (Figure 5). At the base level, we interactively trained the three classes *light-gull* (head), *dark-gull* (back), and *not-gull*. Our 12 features included each of the HSV bands (Foley and van Dam 1982), and variance, contrast and dispersion (Jain 1989) within a 3×3 pixel window for each of these bands. Each mouse click produced a 3×3 blob of training pixels of the same class. It was not necessary (nor possible) to identify light and dark parts of gulls uniquely. Rather, base-level training was stopped (this was after just 10 mouse clicks) when gulls appeared reasonably well *characterized by adjacent blobs* of *light-gull* and *dark-gull* labels. A necessary condition is that no relevant blobs are missed by the classifier. Notice however the large number of false positives of both *light-gull* and *dark-gull* labels (Figure 7).

We then trained the c^2 classifier to discriminate *gull* and *not-gull*. Eight features were computed on C^1, namely *adjacency* ($F_{\textit{light-gull,dark-gull}}$) and *clutter* ($F_c$) as described above,

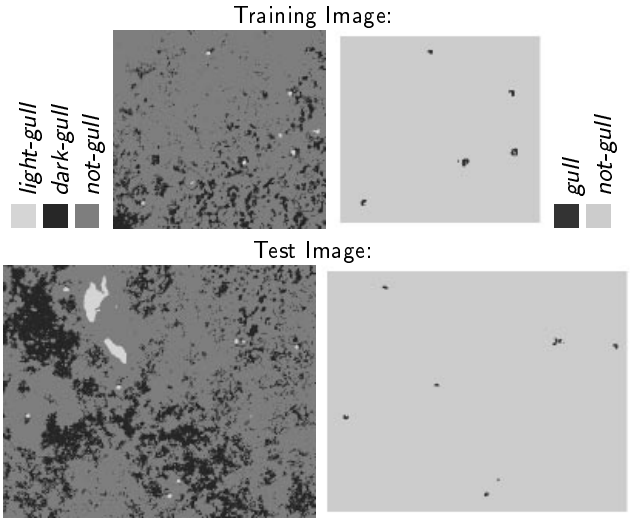

Figure 7: Results of hierarchical classification for black-back seagull localization (cf. Figure 5). **Left:** Base-level (C^1) classification results; **right:** hierarchical (C^2) results.

with window radii $k = 1, 2, 4, 8$ pixels. Training pixels for class *gull* were selected near the junction of head and body of some gulls. After 6 mouse clicks, the tree generated by ITI consisted of a single decision node testing a threshold of $F_{\textit{light-gull,dark-gull}}$ (window radius 2), and achieved perfect accuracy on the training image and near perfect accuracy on the test image (Figure 7). Accuracy is measured as the number of *gull*-labeled blobs that correspond to actual gulls. In the test image, there were only a few minor false positives, which were substantially smaller than the true positives and thus easily removed by standard noise reduction techniques (e.g. replacing each label with its most common neighbor).

5 CONCLUSION

We have demonstrated a new methodology for interactive training of pixel classifiers. It is a very effective tool for selecting few but informative training instances, resulting in great reduction of human labor and dramatically simplified decision tree classifiers. A simple user interface allows training with useful real-time feedback, regardless of the size of the image.

Incremental update of a classifier allows training on an image sequence without a static training set. Our results suggest that the training procedure converges rapidly, but further experiments on real image sequences are necessary.

We introduced the new concept of hierarchical classification and demonstrated its usefulness on a simple seagull recognition problem. The system will subsequently be used for vegetal classification of ground cover in environmental monitoring via forestry aerial images. Future research will apply this technique to other tasks and investigate the behavior of the training process when more classes and more hierarchies are involved in order to express more complex object characteristics.

ACKNOWLEDGMENTS

The sample implementation makes use of the official ITI distribution which is accessible over the Internet at `http://www-ml.cs.umass.edu/iti/index.html`. We thank X. Wang for providing the feature files and ground truth data for the Ft. Hood imagery. This research has been supported in part by the Advanced Research Projects Agency (via Army Research Labs) under contracts DAAL02-91-K-0047 and DACA76-97-K-0005, and by the National Science Foundation under grant IRI-9222766.

REFERENCES

du Buf, J., M. Kardan, and M. Spann 1990. Texture feature performance for image segmentation. *Pattern Recognition 23*(3/4), 291–309.

Conners, R. and C. Harlow 1980. A theoretical comparison of texture algorithms. *IEEE Trans. Pattern Anal. Machine Intell. 2*(3), 204–222.

Devijver, P. A. and J. Kittler 1982. *Pattern recognition: a statistical approach*. Englewood Cliffs: Prentice-Hall.

Ferri, J. J., P. Pudil, M. Hatef, and J. Kittler 1994. Comparative study of techniques for large-scale feature selection. In E. S. Gelsema and L. N. Kanal (Eds.), *Pattern Recognition in Practice IV*, pp. 403–413. Elsevier Science B.V.

Foley, J. D. and A. van Dam 1982. *Fundamentals of interactive computer graphics*. Addision-Wesley.

Foley, J. D. and J. Sammon, Jr. 1975. An optimal set of discriminant vectors. *IEEE Trans. on Computers 24*(3), 281–289.

Haralick, R., K. Shanmugam, and I. Dinstein 1973. Textural features for image classification. *IEEE Trans. Systems, Man, and Cybernetics 3*(6), 610–621.

Jain, A. K. 1989. *Fundamentals of digital image processing*. Prentice-Hall.

Liu, K., Y. Cheng, and J. Yang 1993. Algebraic feature extraction for image recognition based on an optimal discriminant criterion. *Pattern Recognition 26*(6), 903–911.

Ohanian, P. and R. Dubes 1992. Performance evaluation for four classes of textural features. *Pattern Recognition 25*(8), 819–833.

Salzberg, S., A. Delcher, D. Heath, and S. Kasif 1995. Best-case results for nearest-neighbor learning. *IEEE Trans. Pattern Anal. Machine Intell. 17*(6), 599–608.

Schultz, H. 1995. Terrain reconstruction from widely separated images. *Proc. SPIE 2486*, 113–123.

Slaymaker, D. M., K. M. L. Jones, C. R. Griffin, and J. T. Finn 1996. Mapping deciduous forests in southern New England using aerial videography and hyperclustered multi-temporal Landsat TM imagery. In *Gap Analysis, A Landscape Approach to Biodiversity Planning*, pp. 87–101 and 308–312. Bethesda, Maryland: American Society for Photogrammetry and Remote Sensing.

Utgoff, P. E. 1994. An improved algorithm for incremental induction of decision trees. In *Machine Learning: Proc. 11th Int. Conf.*, pp. 318–325. Morgan Kaufmann.

Utgoff, P. E., N. C. Berkman, and J. A. Clouse 1997. Decision tree induction based on efficient tree restructuring. *Machine Learning 29*(1), 5–44.

Wang, X., F. Stolle, H. Schultz, E. M. Riseman, and A. R. Hanson 1997. Using three-dimensional features to improve terrain clasification. In *Proc. Computer Vision and Pattern Recognition*, pp. 915–920.

Weszka, J., C. Dyer, and A. Rosenfeld 1976. A comparative study of texture measures for terrain classification. *IEEE Trans. Systems, Man, and Cybernetics 6*(4), 269–285.