Interactive Training of Pixel Classifiers Opens New Possibilities

Justus H. Piater, Edward M. Riseman, and Paul E. Utgoff

CMPSCI Technical Report 97-51

October 1997

Computer Science Department
Lederle Graduate Research Center
University of Massachusetts
Amherst, MA 01003-4601

lastname@cs.umass.edu

Interactive Training of Pixel Classifiers Opens New Possibilities

Justus H. Piater Edward M. Riseman Paul E. Utgoff

Abstract — We propose a novel interactive, incremental method for pixel classifier construction which yields very efficient decision tree classifiers, and allows extension to training of hierarchical classifiers for recognition of more complex objects.

Traditionally, image pixel classifiers are constructed off-line using pre-selected training instances, many of which may not be informative with respect to improving classification accuracy. Manually preparing these training instances is a costly process in most realistic tasks. We propose an incremental, interactive method, the "Learning Classifier", which relies on real-time feedback. It makes highly efficient use of human effort and yields decision tree classifiers with the desirable property of very few nodes. 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.

Furthermore, two novel concepts made possible by the Learning Classifier are demonstrated: (1) *Incremental training* of a decision tree classifier on a stream of images permits incremental, non-iterative improvement by dynamic addition of user-specified informative training pixels. The training process has the potential of fast convergence, and revisitation of training images is not necessary.

(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. From a label image created by a classifer, additional features are extracted and passed to the next level classifier. We propose a set of simple and generic feature functions that characterize spatial relationships between class labels. These can be used hierarchically to express potentially complex context-sensitive spatial structure. Thus, a human can convey more structural knowledge into the classifier construction process than any reasonable conventional feature set can express. This makes the Learning Classifier capable of solving object recognition tasks beyond the realm of traditional pixel classifier systems, which is exemplified by a seagull counting problem.

Keywords: interactive incremental image pixel classification, decision trees, hierarchical classification, image features, human effort

1 Introduction

Image pixel classification is a fundamental task in computer vision. Pixel classifers are an important component of many vision applications, e.g. texture-based segmentation (7), image understanding (1)(2)(15), object recognition (11)(17) and obstacle detection (16), medical image analysis (22), geoscience (3)(4), and agriculture (24).

Despite these 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.

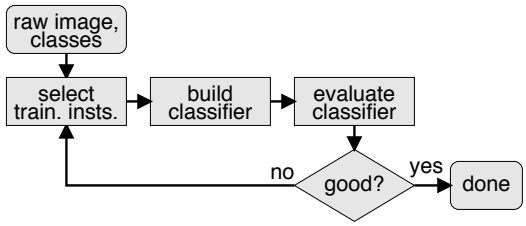

Figure 1: Traditional classifier construction.

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 (23)).

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 (19)(20). 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 the "Learning Classifier". The Teacher is a human domain expert who operates a graphical user interface to select images for training and, for any image, select and label 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.

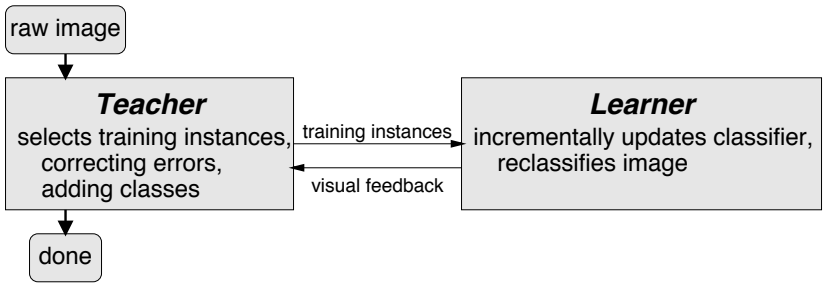

Figure 2: Interactive, incremental classifier construction.

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. Training is simple because the Teacher can produce a new training instance by a simple point-click of the mouse, using a flexible visualization interface. This interactive display-selection loop can operate quickly. In particular, this helps the Teacher to decide where additional or revised training instances are most needed, and when the set of accumulated training instances is sufficient.

This notion of incremental training makes the Learning Classifier suitable for training on image streams. 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. The Learning Classifier, in contrast, does not require fixed training and test sets: Training is started on the first image and continues until the result is satisfactory, and the image may 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 the training can be exploited to allow the user to convey – by by giving examples – more knowledge into the classifier construction process than any reasonable feature set can express. This idea led to the concept of hierarchical classification, which is presented in Section 4.

2 Implementation 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 (6). Selection of an optimal feature subset from a given universe of features has been shown to be infeasible in practice (8). 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 selected the incremental decision tree inducer ITI (27)(28). 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.

3 Quantitative Results

In this section, we compare our Learning Classifier with a previously published classification result by Wang et al. (29). 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 an orthographic 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-occurence features* (angular second moment, contrast, and entropy at four angular orientations each (12)), four *three-dimensional features* introduced by Wang et al., and the intensity. The co-occurence features employed have previously been claimed to be highly effective for classification (5)(7)(21)(30). The 3D features are generated along with the orthographic image and a digital elevation map during stereo processing of a calibrated image pair (25) and were recently shown

to be highly discriminative in this task. The Foley-Sammon transform (FST (9)) was employed as a classifier. FST is a linear discriminant method that is considered effective (18)(30).

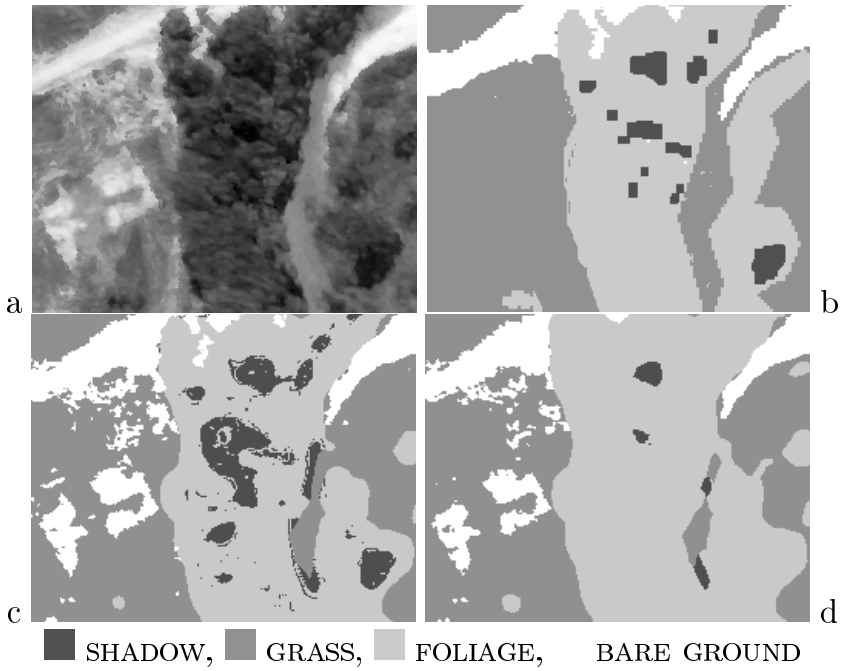

Figure 3: (a) Subimage (250 × 200 pixels) of Ft. Hood scene; (b) manually generated ground truth (note the difficulty of this task on this particular subimage); (c) classification results generated by an ITI classifier built in traditional batch mode using Wang et al.'s training data; (d) classification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

As a training set, Wang et al. used four homogeneous square regions of different sizes: 99×99 (FOLIAGE), 75×75 (GRASS), 37×37 (BARE GROUND), 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. The achieved classification accuracy is shown in Table 1. For more details, refer to Wang et al. (29).

To provide a baseline of the performance of ITI with respect to FST on this task, we ran ITI in batch mode on the same input data as described above (Table 2). Note that ITI here outperformed FST in terms of classification accuracy.[1]

To give a first impression of the Learning Classifier when trained on an image stream, 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

[1]Slight discrepancies in the totals arise because Wang et al. apparently included their training set in the test set. Assuming that the entire training set was correctly classified, the resulting overall classification accuracy is reduced from 83.4 to 83.3%.

FST results:	SHADOW	GRASS	FOLIAGE	BARE GRD	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 GRD	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

Table 1: Contingency table of classification results (from Wang et al.). Numbers represent pixels in 1000's. The first row means that of the 41800 ground truth SHADOW pixels, 13800 were classified as SHADOW, 27800 as FOLIAGE, and 200 as BARE GROUND.

ITI results:	SHADOW	GRASS	FOLIAGE	BARE GRD	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 GRD	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 2: Classification results using ITI in batch mode on the same training set as in Wang et al. Numbers represent pixels in 1000's.

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 off-line 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

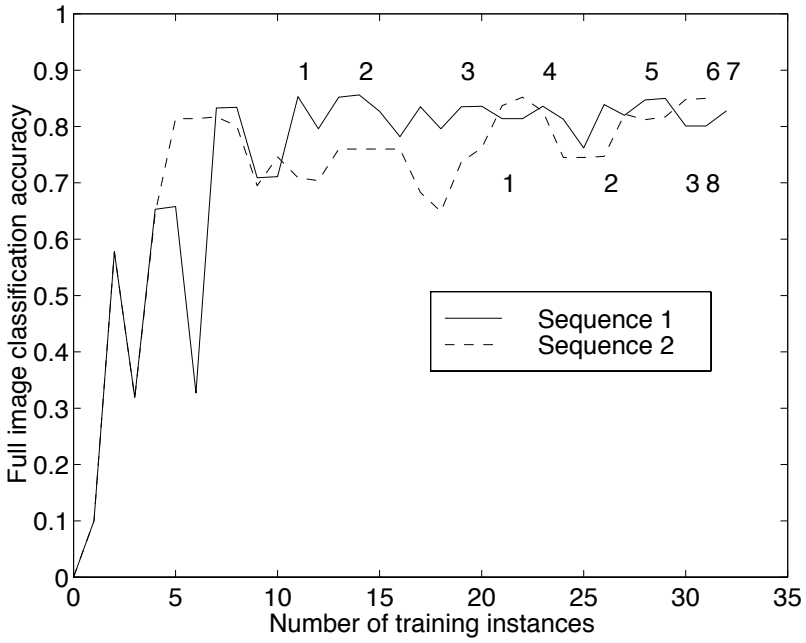

Figure 4: Classifier behavior during interactive training. Digit k next to a curve marks the last training pixel supplied while training on subimage k. In Sequence 1 (top digits), no training instances were supplied for subimages 8–10; in Sequence 2 (bottom digits), none were supplied for subimages 4–7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

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.

Table 3 compares two interactively trained ITI classifiers with a traditionally constructed ITI classifier (using Wang's training data). All three perform equally well; the differences in accuracy are neglegible. 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 14th (interactively trained) ITI classifier from Sequence 1. 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 a striking example of the dramatically increased classifier quality achievable by our Learning Classifier approach.

	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 3: 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.

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 chair. 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 (26). 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 simple 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 head and back. Often the white tip of the tail can be seen as well (Figures 5 and 6). 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.

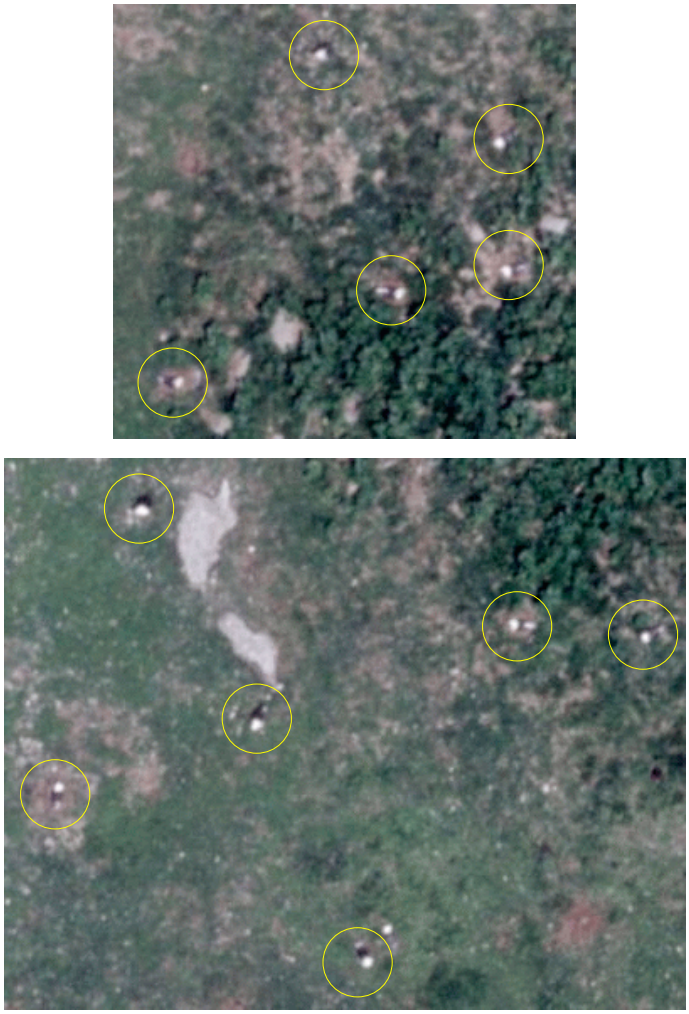

Figure 5: Aerial photographs showing sections of an island off the coast of Maine. Yellow circles show black-back seagulls. The top image (275×258 pixels) was used as a training image, the bottom one (405×328 pixels) as a test image.

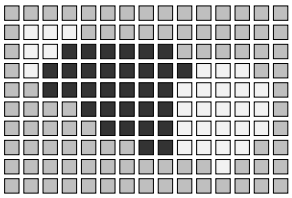

Figure 6: Sketch of a typical black-back gull as it appears in Figure 5. The white head (lower right), white tail (upper left) and black back are represented by adjacent blobs of pixels.

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 gull in Figure 6. It is not generally practical to design a generic feature set that expresses spatial relationships between blobs of colored pixels. Observe, however, the hierarchy inherent in the previous formulation. This is what hierarchical classification can exploit: First, a classifier is trained to recognize blobs of colored pixels. 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.

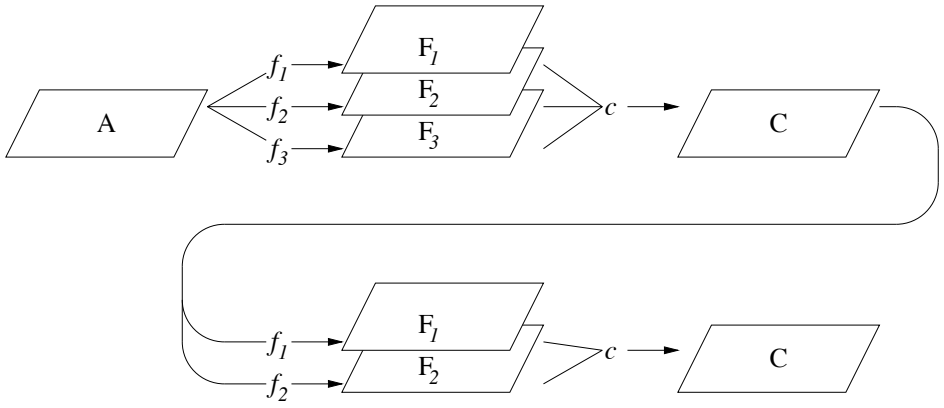

Figure 7: The concept of hierarchical classification: The output of one classification system serves as the input to another one. This can in principle be extended to any number of hierarchies. The feature extractors f typically differ from level to level, and a unique classifier c is trained at each level.

We now describe this procedure in detail (cf. Figure 7). 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):

$$F(A;i,j) = f\ \Big(\ \begin{array}{l} A(i-k_f,j-k_f),\ldots,A(i+k_f,j-k_f)\ \ ,\ldots, \\ A(i-k_f,j+k_f),\ldots,A(i+k_f,j+k_f)\end{array}\ \Big)$$

A pixel classifier c takes the features $F_1,\ldots,F_l$ of a pixel at (i,j) and returns a class label. The resulting *label image* is denoted $C(F_1,\ldots,F_l;i,j)$ 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 features $f^1_1,\ldots,f^1_{l^1}$. 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 effects of C^h on C^{h+1} are hardly forseeable, this concept relies on interactive training: Informative training examples at one level are selected depending on the classification results from the previous level.

There is nothing special about the base level (c^1) classifier. The input data to a higher level feature function f, however, will generally be a label image, i.e. the pixel values do not represent cardinalities but merely categories. This requires the construction of a special feature set. For example, we suggest the following window functions ($-k \leq x, y \leq k$) for capturing simple spatial features:

- To measure the *size* of an blob of labels, use the number of occurrences of a given label M:
$$F_s(i, j) = |\{(x, y) \mid C(i + x, j + y) = M\}|$$
- To assess *adjacency* of two blobs, use the product of the number of occurrences of two given labels M and N:
$$F_a(i, j) = |\{(x, y) \mid C(i + x, j + y) = M\}| \cdot |\{(x, y) \mid C(i + x, j + y) = N\}|$$
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 following feature provides a measure of *clutter* by counting the number of label discontinuities between neighboring pixels:
$$\begin{aligned} F_c(i, j) = \; & |\{(x, y) \mid C(i + x, j + y) \neq C(i + x + 1, j + y)\}| \; + \\ & |\{(x, y) \mid C(i + x, j + y) \neq C(i + x, j + y + 1)\}| \end{aligned}$$
A generalization of this was proposed by Jarvis (14) as a sharpness criterion for image processing.

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 (10), and variance, contrast and dispersion (13) 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 after 10 mouse clicks when there were still false positives of both LIGHT-GULL and DARK-GULL, but gulls appeared reasonably well *characterized by adjacent blobs* of LIGHT-GULL and DARK-GULL labels (Figure 9). The trained decision tree is shown in Figure 8.

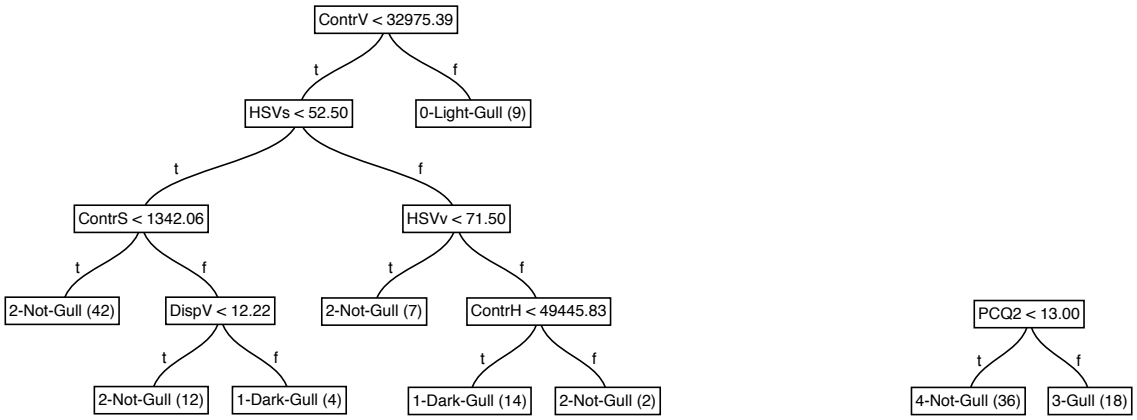

Figure 8: Decision trees trained for black-back seagull localization (cf. Figure 5). In the leaf nodes, the numbers of associated training pixels are given in parentheses. **Left:** Base-level classifier c^1. In the decision nodes, the trailing letters of the feature names refer to the respective HSV band. **Right**: Classifier c^2. Only the *adjacency* feature (F_a) with radius 2 was employed.

We then trained the c^2 classifier to discriminate GULL and NOT-GULL. Eight features were computed on C^1, namely *adjacency* (F_a) and *clutter* (F_c) as described above, 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. An extremely simple tree was generated (Figure 8) after 6 mouse clicks, which nevertheless achieved perfect accuracy on the training image and near perfect accuracy on the test image (Figure 9). 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 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 classifiers. A simple user

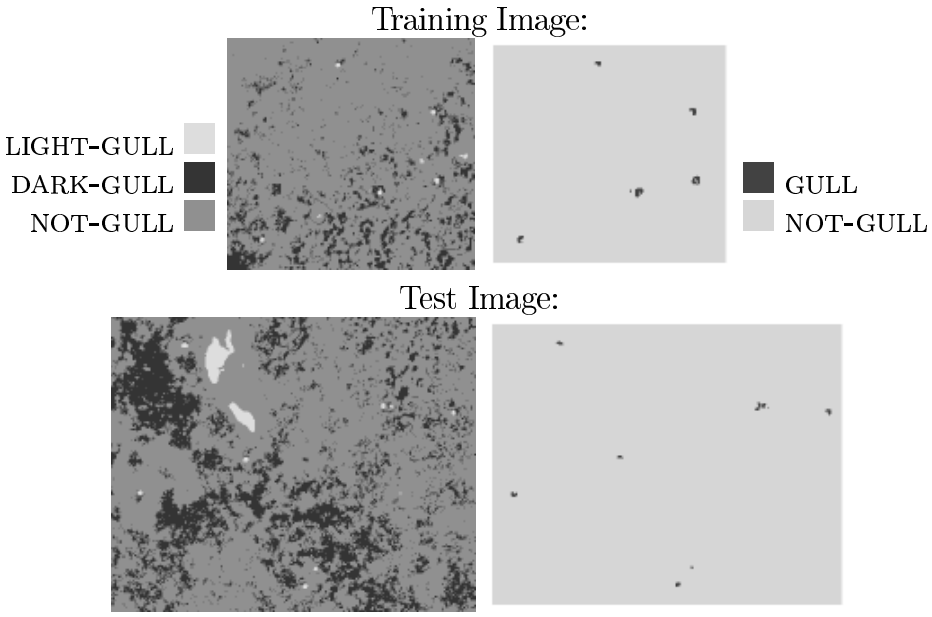

Figure 9: Results of hierarchical classification for black-back seagull localization (cf. Figure 5). **Left:** Base-level (C^1) classification results; **right:** hierarchical (C^2) results.

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.

Acknowledgements

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`. The decision tree illustrations were also generated using this package. We thank X. Wang for providing the feature files and ground truth data for the Ft. Hood imagery.

References

[1] M. Blume and D. R. Ballard. Image annotation based on learning vector quantization and localized Haar wavelet transform features. *Proc. SPIE*, 3077:181–190, 1997.

[2] N. W. Campbell, W. P. J. Mackeown, B. T. Thomas, and T. Troscianko. Interpreting image databases by region classification. *Pattern Recognition*, 30(4):555–563, 1997.

[3] G. A. Carpenter, M. N. Gjaja, S. Gopal, and C. E. Woodcock. ART neural networks for remote sensing: vegetation classification from Landsat TM and terrain data. *IEEE Trans. Geoscience and Remote Sensing*, 35(2):308–325, 1997.

[4] J. R. Carr. Spectral and textural classification of single and multiple band digital images. *Computers & Geosciences*, 22(8):849–865, 1996.

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

[6] P. A. Devijver and J. Kittler. *Pattern recognition: a statistical approach*. Prentice-Hall, 1982.

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

[8] J. J. Ferri, P. Pudil, M. Hatef, and J. Kittler. Comparative study of techniques for large-scale feature selection. In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice IV*, pp. 403–413. 1994.

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

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

[11] W. Hafner and O. Munkelt. Using color for detecting persons in image sequences. *Pattern Recognition and Image Analysis*, 7(1):47–52, 1997.

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

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

[14] R. A. Jarvis. Focus optimisation criteria for computer image processing. *Microscope*, 24(2):163–180, 1976.

[15] M.-P. D. Jolly and A. Gupta. Color and texture fusion: application to aerial image segmentation and GIS updating. In *IEEE Workshop on Applications of Computer Vision*, pp. 2–7, 1996.

[16] D. Langer and T. Jochem. Fusing radar and vision for detecting, classifying and avoiding roadway obstacles. In *Proc. IEEE Intelligent Vehicles Symposium*, pp. 333–338, 1996.

[17] E. Littmann and H. Ritter. Adaptive color segmentation – a comparison of neural and statistical methods. *IEEE Trans. Neural Networks*, 8(1):175–185, 1997.

[18] K. Liu, Y. Cheng, and J. Yang. Algebraic feature extraction for image recognition

based on an optimal discriminant criterion. *Pattern Recognition*, 26(6):903–911, 1993.

[19] T. M. Mitchell. *Version spaces: An approach to concept learning*. PhD thesis, Department of Electrical Engineering, Stanford University, Palo Alto, CA, 1978.

[20] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

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

[22] C. Rossmanith, H. Handels, S. J. Poppl, and E. Rinast. Characterisaztion and classification of brain tumours in three-dimensional MR image sequences. In K. H. Hohne and R. Kikinis, editors, *Proc. Visualization in Biomedical Computing*, pp. 429–438. 1996.

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

[24] D. L. Schmoldt, P. Li, and A. L. Abbott. Machine vision using artificial neural networks with local 3D neighborhoods. *Computers and Electronics in Agriculture*, 16(3):255–271, 1997.

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

[26] D. M. Slaymaker, K. M. L. Jones, C. R. Griffin, and J. T. Finn. 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. 1996.

[27] P. E. Utgoff. An improved algorithm for incremental induction of decision trees. In *Machine Learning: Proceedings of the Eleventh International Conference*, pp. 318–325. 1994.

[28] P. E. Utgoff, N. C. Berkman, and J. A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, in press.

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

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