Int. J. Pattern Recognition and Artificial Intelligence 13(2), 1999. © World Scientific.
This is a draft version. Please consult the original publication for accurate reference.

Interactively Training Pixel Classifiers

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

January 19, 1999

Computer Science Department
University of Massachusetts
Amherst, MA 01003

Phone: (413) 545-3143
Fax: (413) 545-1249

Email: `piater@cs.umass.edu`

Abstract — For typical classification tasks, all training data are prepared in advance and are supplied to the classifier all at once. This is unnecessarily expensive and incurs overfitting problems, since the individual contributions of the training instances to the classifier are not known. We address this by proposing an interactive incremental framework for image classifier construction, where small numbers of training examples are supplied at each user interaction. After incorporating new training instances, the classifier immediately reclassifies the image to provide the user with instant feedback. This allows the user to choose additional *informative* training pixels from among the currently misclassified ones. Using a realistic terrain classification task, we demonstrate the potential of our method to generate small and accurate decision tree classifiers from surprisingly few training examples while avoiding overspecialization. We also briefly discuss the novel concept of *hierarchical classification*, where higher-level classifiers take as input the output of lower-level classifiers. We present preliminary results indicating that within our interactive framework, this is a practical approach to exploiting spatial relationships for classification.

Keywords: interactive image classifier construction, incremental decision trees, human effort, overfitting.

1 Introduction

As advances are made in technology for machine learning, one can expect to see this technology incorporated in tools for constructing decision making components of larger systems that non-specialists build. In particular, pixel classifiers are an important component of many vision applications, e.g. texture-based segmentation [9, 1], image understanding [3, 14], object recognition [12], obstacle detection [15], and geoscience [5, 4].

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 supplied 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. Then, the entire procedure is repeated — often a time-consuming and tedious task.

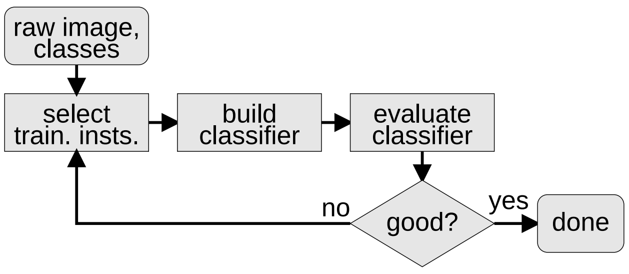

Figure 1: Traditional classifier construction.

It is well known that the appropriateness of the training set has a great influence on the performance of a classifier. For this reason, significant 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 by hand the correct labels for the training pixels. 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 performance [21].

There are other benefits of keeping the training set small. For example, a typical decision tree classifier will make every 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 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 that try to detect overspecialization and then simplify the decision tree. Such pruning reduces the classification accuracy on the training set to some degree, but in practice the accuracy on independent test data often increases. 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 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.

Intuitively, an informative training example is one that is of significance to the classification process. If a large number of training examples is chosen a priori, as is usually done, then the contribution of the individual instances is not known. Evidently, if one knew how to choose the instances appropriately, a very similar classification performance could be achieved with much fewer training examples.

In this paper, we are not addressing the general problem of selecting optimal sets of training instances. Instead, we propose to supply training instances incrementally, which is motivated by the following consideration: If one could know where the classifier currently makes mistakes, one could generate an informative instance by providing a correct label for a currently misclassified pixel. We arrive at 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 interactive incremental Teacher-Learner paradigm (Figure 2), which we call ITL. The Teacher is a human domain expert who operates a graphical user interface. He can 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 quickly produce a classifier and labels for the pixels of the current training image, according to the most recent classifier.

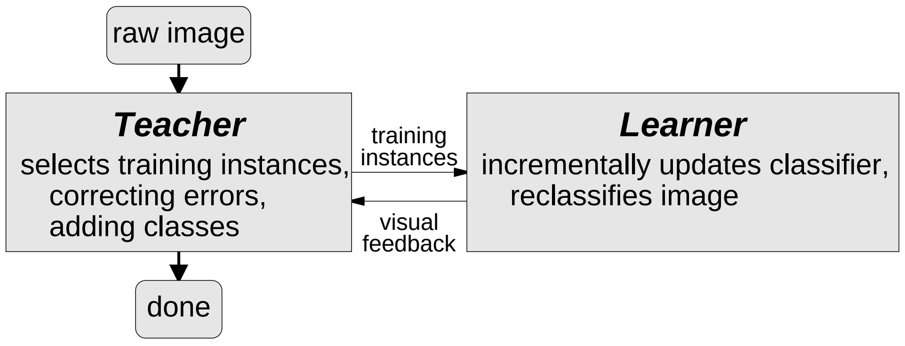
raw image
Teacher
selects training instances, correcting errors, adding classes
training instances
visual feedback
Learner
incrementally updates classifier, reclassifies image
done

Figure 2: The Incremental Teacher/Learner framework (ITL): 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 rapidly revises its classifier as necessary, and then recomputes the class labels for all pixels of the image. This lets the user see the misclassified pixels with almost no delay. He can immediately respond by providing correct labels for one or more of them, which are passed as new training examples to the classifier. In this sense, we call a newly supplied training instance informative if and only if it is misclassified by the current classifier.

2 Incremental Decision Trees

This work is primarily concerned with effective selection of training instances. Another important issue in classifier construction is the selection of a feature set. It is known that increasing the size of a feature set can adversely affect classifier performance [8]. Selection of an optimal feature subset from a given universe of features has been shown to be infeasible in practice [10]. Classifiers that utilize all available features (such as neural networks, nearest-neighbor clusterers, linear machines) are particularly sensitive to redundant and noisy features. Therefore, they prefer very large training sets. This motivates the use of a univariate *decision tree* classifier which consults only a single feature at each decision node. Only *useful* features are incorporated into the tree, and features of little discriminative power are disregarded entirely. "Useful" here refers to the ability of the classifier to classify the training set correctly. If overfitting is effectively avoided by proper selection of training instances, then the resulting decision tree may not require all available features. 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 it in terms of training instance selection.

With ITL, the user presents training instances sequentially to the classifier construction system, and expects the classifier to incorporate each new training example very quickly. Thus, the system requires a classifier that can be rebuilt or incrementally upgraded very quickly, without unlearning previously learned instances. This rules out many classifiers, e.g. neural networks which converge relatively slowly and require a large number of training example presentations. Decision trees, on the other hand, are known for their computational efficiency.

An early incremental decision tree algorithm was proposed by Crawford [7] based on CART [2]. When a new training instance would cause a new test to be picked at a decision node, the entire subtree rooted at this node is discarded and rebuilt based on the corresponding subset of the training examples. Lovell and Bradley [17] constructed another partially incremental decision tree algorithm, the "Multiscale Classifier". It works by adjusting decision thresholds and, if necessary, splitting leaves by introducing new decision nodes. Because all the data seen are not stored in the tree, these adjustments may cause previously processed instances to be classified incorrectly. Therefore, these instances must be presented to the decision tree again, which in turn may cause alterations of the tree. The method refines the tree incrementally, and the result is dependent on the order of the training instances.

Utgoff's Incremental Tree Inducer ITI [25, 26] solves this problem by storing all data relevant for restructuring a decision tree within the nodes [22]. It can accept and incorporate training instances serially without needing to rebuild the tree repeatedly. Another desirable characteristic is that it produces the same tree for the same accumulated set of training instances, regardless of the order in which they are received.[1] It can also operate in conventional batch mode, where the full set of training instances is made available at once. The classification accuracy is statistically indistinguishable [26] from that of C4.5 [20], which is widely considered one of the leading decision tree algorithms.

Each decision node in ITI defines a cut point of a tested numeric variable, or a comparison of a symbolic variable. Each numeric decision node maintains a sorted list of all values of that variable encountered in the instances. Each of these values is tagged by the corresponding class label(s). For each pair of adjacent values of different classes, the midpoint of the two values defines a possible cut point. The legal cut points and the merit of each one can be computed efficiently during a single pass over the sorted list of tagged values. Similarly, each symbolic decision node keeps track of how many instances of each value of the respective variable are encountered.

When a new training instance is added, it works its way down the tree until it ends up in a leaf, storing its values of the variables tested at each decision node along the way. If the leaf has the correct class label, then the instance is simply added to the leaf node. Otherwise, this node is turned into a decision node that classifies all instances already stored here plus the new one correctly.

Then, each decision node encountered by this training instance is revisited for the purpose of making sure that the "best test" is installed at each node, according to some metric. This can be computed from the information stored at each decision node and its children. Proceeding from root to leaf ensures that the resulting tree is optimal with respect to the given metric.

One drawback of univariate decision trees like ITI is that decision boundaries best described by functions of multiple features must be approximated by multiple univariate decisions. Nevertheless, for ITL, the computational efficiency of ITI (in terms of tree revision and instance classification) and relatively good generalization properties make it an excellent system. It achieves a very quick feedback loop, consisting of receiving a new training instance, updating the classifier, and reclassifying the image. This allows ITI to function in close to real time, ensuring the effectiveness of the human in the loop.

[1]Note however that in this application the final decision tree will generally depend on the order in which the user supplied the training instances, since their selection by the user depends on the feedback received from the classifier.

To maximize the utility to the user, pixels near the location of the latest training pixel are (re)classified first and displayed by our graphic user interface. Note that this does not involve any distance computations, since the distances are implicitly given by the location of the pixels in the image. The user can select new training pixels at any time, without waiting for the classification process to complete, which facilitates very rapid training even on large images.

3 Qualitative Examples

This section walks through an example session. The initial training image is the one shown in Figure 10. The goal is to learn to classify pixels as one of SKY, ROOF, BRICK, or FOLIAGE. Pixels that belong to another region type, e.g. PAVEMENT, are not of interest. In this example, none of these pixels will be labeled by the teacher, and will therefore never serve as a training instance. Six features are used, which are the raw red/green/blue measurements of a pixel, and the variances of each in a 3×3 window centered around that pixel. Each mouse click of the teacher produces a 3×3 square of training instances of a specific class that are used to update the learner's decision tree. Immediately, the Learner begins to reclassify the image using the updated tree. Figure 11 shows various intermediate stages of training performed on a subimage.

When the teacher points to a pixel and labels it as a training instance, the eight adjoining pixels are also included as well, producing a total of nine training instances per click. Thus, the teacher clicks on a 3×3 'square' rather than a pixel, and nine training pixels are produced at once. The discussion below is in terms of clicking on a 'square', which is a simple way of saying that one clicks a 3×3 set of pixels.

Figure 11b shows the classification result after labeling just one square of each of two classes, SKY and ROOF. The sky is already almost perfectly separated from the rest of the image. In Figures 11c and d, one square of each of the remaining two classes is added. While the addition of BRICK (c) again results in good generalization, things become more complicated when a sample of the FOLIAGE class is added (d). This occurs in this image because FOLIAGE and ROOF are mainly characterized by large variances within the RGB intensities rather than the colors themselves, and thus hard to separate. The image in Figure 11i contains several contradictory training instances that belong to different classes (FOLIAGE and ROOF) but are characterized by identical feature vectors (RGB values and variances). Thus, they are indistinguishable using the given feature set, which is not surprising — even for a human, many low-contrast textured areas in roof and foliage are difficult to discern. Consequently, perfect classification is not achievable given these features, and classifier training is stopped at this point.

The results of Figure 11i on the entire image are shown in Figure 12a. Note that many areas outside the trained subimage are poorly classified. For example, large parts of the left of the house and the brick wall are labeled as ROOF, and most of the bush on the right is labeled as BRICK. However, very little additional training is needed to eliminate most misclassifications in all parts of the image, as shown in Figure 12b. This is a huge gain over exhaustive off-line labeling. In a typical application involving training on multiple images, only the first image will require substantial training. On subsequent images, the amount of training required will generally decline as illustrated here (see also Section 5.1).

Figure 13a shows an enlarged subimage of Figure 12b. Note that the lower part of the chimney is surrounded by a border belonging to the ROOF class, which is clearly a misclassification. This is because the system has learned that the roof is mainly characterized by its variance; therefore many cluttered and contrasted areas are classified as ROOF. However, after training on just one incorrect square of the SKY class as shown in Figure 13b, almost all of the border around the chimney becomes classified correctly. In the same way, other misclassifications can be corrected, as long as the features are sufficiently discriminant. This process is significantly faster than exhaustively applying the correct labels by hand.

How well does the system work? In terms of accuracy and efficient production of well chosen training instances, the system performs very nicely. Because the tree generalizes well, it is not necessary to make a large number of corrections. The error-driven point-click loop can be very productive. It is much more satisfying to build a pixel classification component by correcting its mistakes than it is to generate large amounts of training data without knowing which are informative.

In terms of computational efficiency, ITI produced small trees that were highly efficient classifiers. The learner classifies a small image of about $200 \times 200 = 40,000$ pixels in a few seconds. This rate is linear in the number of pixels in an image, and largely independent of the number of features (if they are pre-computed off-line).

In the examples above, a mouse-click by the teacher generated nine training instances (the pixels in a 3 by 3 square region). If these squares are bigger, fewer mouse-clicks will generally be needed unless the covered regions are very uniform. On the other hand, the learner will spend more time processing that many more training instances. This issue of how many training examples to generate per mouse click will be discussed in more detail in Section 4.

While learning is very fast in the early stages of training, the later stages usually involve much refinement. As errors are corrected in one part of the image, others appear in different locations. This usually goes on for some number of iterations. On the other hand, inspection of the decision tree will reveal when contradictory training instances indicate that the discriminatory power of the features is reached. This is the case when impure leaves occur, i.e. leaves which contain training examples which do not all belong to the same class.

By choosing appropriate training examples, one can, to some extent, bias the classifier to avoid certain types of errors while tolerating others. For example, in the illustrations above it was not possible to separate ROOF from FOLIAGE completely. If the teacher were more concerned about correct classification of the roof than the tree, he could give the learner more training examples of ROOF which would increase the accuracy on ROOF at the expense of more misclassifications of pixels belonging to the tree as ROOF.

4 Quantitative Results

We now compare the performance of our ITL classifier with a previously published classification result by Wang et al. [27]. 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 (1,936,789 pixels) of a rural area in Ft. Hood, Texas (Figure 14a). The goal was to build a pixel classifier to recognize the four terrain classes BARE GROUND (road, river bed), 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 [13]), four *three-dimensional features* [27], and the gray value. The co-occurrence features employed have previously been claimed to be highly effective for classification [6, 9, 18, 28]. The 3D features are generated during stereo processing of a calibrated image pair [23] and were recently shown to be highly discriminative in this task [27]. The Foley-Sammon Transform (FST [11]) was employed as a classifier. FST is a linear discriminant method that is considered effective [16, 28].

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. Ground truth was generated by hand. The achieved classification accuracy was 83.4% [27].

To provide a baseline of the performance of ITI with respect to FST on this task, we ran ITI in conventional batch mode on the same input data as described above, using the full training set of 16916 pixels. ITI achieved a classification accuracy of 86.3% (86.4% using ITI's pruning mechanism), outperforming FST.

We then trained various classifiers interactively on these data using ITL. The intermediate decision trees resulting from each mouse click were saved and subsequently used to generate performance curves. The training sessions differed in the number of training examples generated by each mouse click. We will discuss cases where each mouse click generated either 1, 9, or 61 training pixels (arranged roughly in the shape of a circular disc centered at the mouse pointer), and one case where the user applied broad strokes to "paint" larger areas with training pixels, thus generating a variable number of training examples per interaction (about 1000 on average).

Figure 3 shows learning curves for example sessions of 1 and 9 training pixels per mouse click. Good classification accuracy was achieved after very few mouse clicks. For the cases of 61 pixels per mouse click and the "paint" mode, learning curves are depicted in Figure 4. Our initial results indicate that higher accuracy can be achieved sooner if more training pixels are supplied per user interaction. On the downside, the classification algorithm needs to keep around this increasing number of training examples if they are not to be unlearned (which is the case here). This sets a limit to the number of training examples that can be managed fast enough for productive user interaction. Because of this limit, the "paint" mode learning curve shown in Figure 4 was discontinued after 16 user interactions.

None of the interactively generated classifiers quite reached or exceeded the accuracy of the batch-generated ITI classifier using Wang et al.'s training set of 16916 pixels. This shows that continued training should yield further improvement in the long run. In the light of the "paint" mode run which

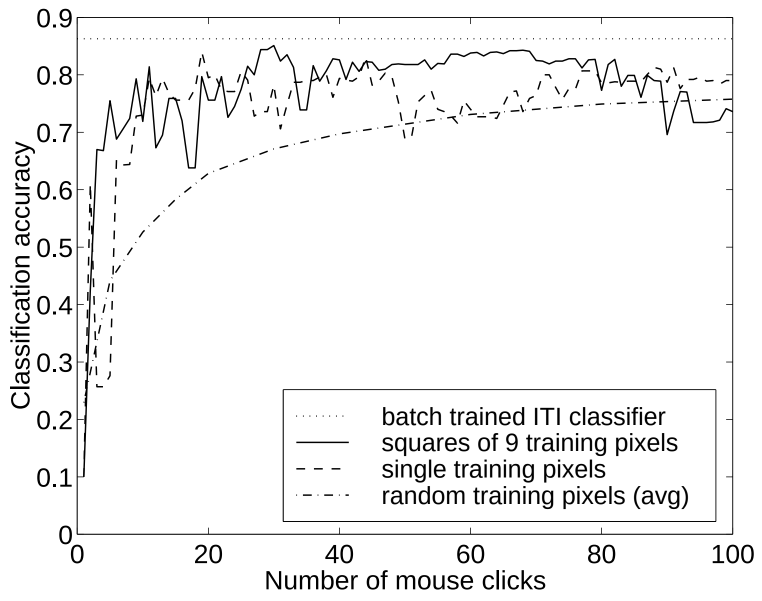

Figure 3: Learning curves obtained during interactive training of a classifier. Each mouse click generated either a 3×3 square of training pixels, or a single training pixel. The ITI batch classifier trained on Wang et al.'s 16916 training pixels and an average of 100 runs with randomly selected training pixels are also shown. For the latter curve, each training pixel was picked from each class with equal probability as a human might do, even if a class is rare.

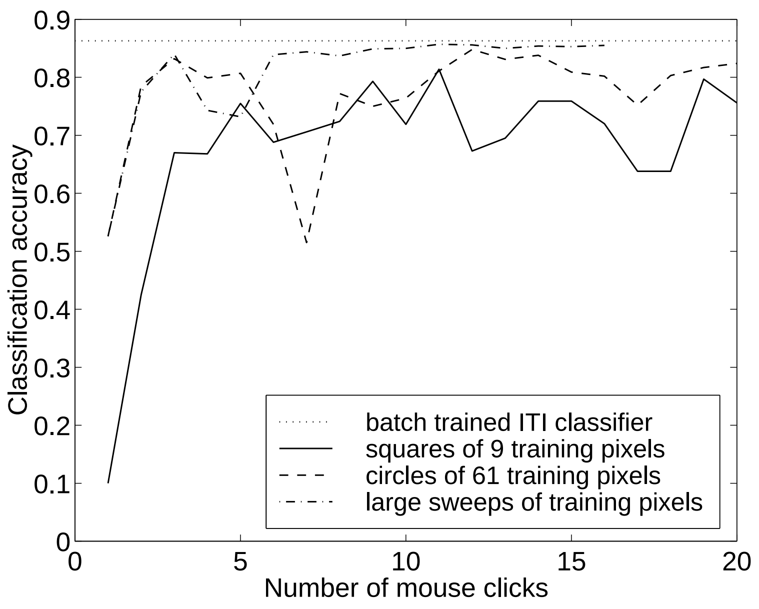

Figure 4: Learning curves obtained during interactive training of a classifier. In one case, each user interaction consisted of a single mouse click generating a patch of 61 training pixels; in another case, the user applied broad strokes generating large numbers of training pixels at once. The first section of the 9-pixel curve from Figure 3 is reproduced here for comparison; recall that it attained its maximum at 30 mouse clicks.

resulted in a similar number of training instances (15497 at the end of the run), it also gives evidence that Wang et al.'s training data were carefully selected for high accuracy.

However, one must bear in mind that the evaluation is done on a single image which was used for both training and testing. Continuing to select training instances from this image will presumably lead to a very specialized classifier with poor generalization properties for other images. This is likely to be the case with the preselected training set. We have not been able to verify this presumption for lack of suitable classification data with ground truth. We shall discuss a related experiment below in Section 5.1.

This specialization/generalization conflict is well illustrated by a brief analysis of some of the resulting decision trees. Table 1 shows that batch training on the large preselected training set produced a large tree which employed nearly all available features, even after pruning. In comparison, the interactively trained classifiers were all very small, with the exception of the "paint" mode run, and used only small subsets of the available features, at very little loss in classification accuracy (see also Figure 14). Undoubtedly the complex trees achieved a marginally greater accuracy by accounting for a large number of exceptions that cannot be expected to generalize to other (similar) images. Therefore it would be better to ignore these exceptions during training. In contrast, the simple classifiers achieved good results because their *very few* training pixels were *selected in an informed manner.* Pruning of the batch-generated classifier managed to cut the number of tree nodes in half, but the the resulting tree is still much larger than the smaller interactively trained classifiers, while only marginally more accurate.

	interactive					*batch*	
				full,	*pruned*	*full,*	*pruned*
# pixels / click:	1	9	61	"paint"			
# mouse clicks:	19	30	12	11			
# training instances:	19	270	732	11394		16916	
% correct:	84.0	85.1	84.8	85.7	85.7	86.3	86.4
# tree nodes:	9	25	13	79	73	143	71
# features used (of 17):	3	7	5	15	15	15	15

Table 1: Summary of classification results using ITI.

In all of our experiments where few (one or nine) training pixels were generated per mouse click, ITI's pruning algorithm left the decision trees unchanged. This is in agreement with our intuition that the training examples are selected because they are *informative* and, by choice of the user, *typical* representatives of their class. On the other hand, if larger numbers of training pixels are selected at each user interaction, redundant and atypical examples are inadvertently included. This leads to larger decision trees, and pruning becomes an issue. We found that in the 61-pixel and "paint" modes, pruning began to take effect in cutting down the size of the trees (Figure 5). However, the classification accuracy was in no case noticeably affected by pruning.

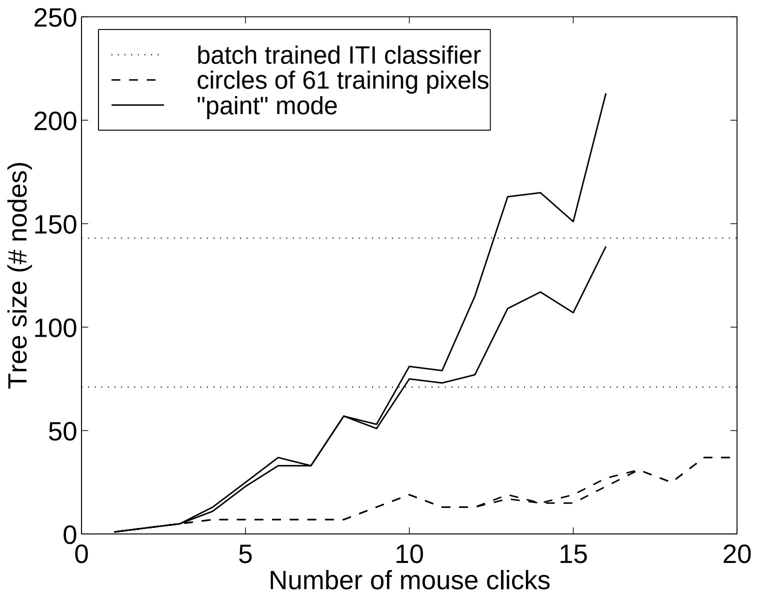

Figure 5: Decision tree sizes during interactive training. The upper curve of a corresponding pair of curves shows the fully grown tree, whereas the lower curve shows its pruned version.

Figure 5 shows how the decision trees, pruned and unpruned, change in size during these two training sessions. As just discussed, pruning starts to have a noticeable effect when the number of training examples grows large. A comparison of the "paint"-mode classifier with the batch-trained classifier is interesting: After 11 user interactions, both classifiers are about equally accurate (Table 1). At this point, they have seen a comparable number of training examples (11394 vs. 16916), and the pruned versions of their trees are of about the same size. However, in contrast to the batch trained tree, the unpruned version of the interactively trained tree is only marginally larger than the pruned one. The presumed reason is that this classifier was trained on examples which were carefully chosen to be informative and representative. As training is continued even further, an increasing number of less typical examples is introduced because those are the ones that are left misclassified. Consequently, the sizes of the pruned and unpruned trees diverge, yet their accuracy ceases to increase.

The preceding paragraphs emphasized the importance of informative training examples. For comparison with uninformed selection of training pixels, Figure 3 includes a learning curve of a classifier trained by randomly selected training pixels, regardless of whether a newly chosen training pixel is misclassified by the current classifier. This curve rises much more slowly than the interactively built classifiers. Clearly, informed selection of training examples can facilitate the rapid construction of simple decision tree classifiers.

It is also interesting to note that there is a component of human skill in selecting useful training examples. Figure 6 depicts a learning curve created by selecting training pixels at random from among currently misclassified pixels only. This implies that each new training pixel alters the classifier and is therefore informative according to our definition. In fact, this learning curve rises somewhat faster than if pixels are selected purely at random. However, it still does not even come close to a learning curve trained by a human teacher. At some point – after about 40 training pixels – the curves cross. (Even though there is much variability in the random learning curves, this phenomenon is statistically significant.) A plausible explanation for this is that after this point, most "typical" representatives of a class are already classified correctly, and forcing the algorithm to select a currently misclassified pixel causes overspecialization by including atypical "exceptions" into the tree.

This explanation carries over to the interactively trained classifiers. Note in Figures 3 and 4 that after a certain number of user interactions, the accuracy ceases to increase or even decreases with further training. We suggest that this is probably significant and is due to the fact that since the user will predominantly select training examples from among currently misclassified pixels, an increasing number of atypical examples will be selected.

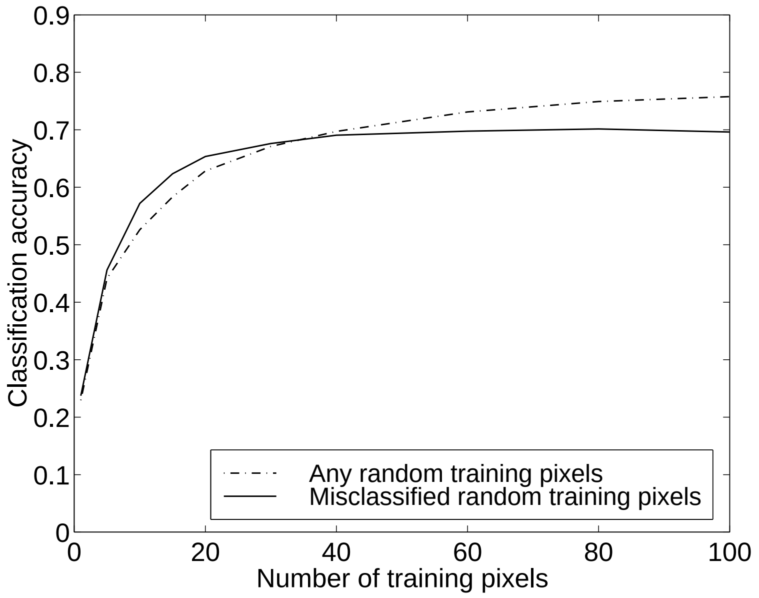

Figure 6: Learning curves for incrementally trained classifiers using randomly selected training pixels. Each curve represents an average of 100 individual runs. The dash-dotted curve is identical to the one shown in Figure 3. The 95% confidence intervals are around 0.02, the standard deviations around 0.1.

5 Discussion and Future Work

The results presented in the preceding section suggest that incremental selection of informative training instances has great potential to create small decision trees from relatively few training examples. The availability of ground truth permitted monitoring of the training process and the selection of the best classifier. In most applications of pixel classifiers, ground truth is not available and must be manually prepared for the training of classifiers. As mentioned in the Introduction, the reduction of this expensive process was one of the motivations for our interactive approach. Without ground truth available, it is not straightforward to monitor the training process and evaluate classification accuracy as described in the previous section. Therefore, applications requiring accurate assessment of classifier performance on large amounts of test data do not take full advantage of the ITL approach, unless ground truth data are already available — but this is true of any classifier. The advantages of ITL come fully into play when quick classification results are desired, and subjective visual assessment of classification accuracy is sufficient. In such situations, the cost associated with classifier construction is greatly reduced because ground truth only needs to be specified for a small number of training instances.

An open issue is when to stop training. The results discussed above indicated that the best performing classifiers are obtained in early stages of training, before overfitting becomes an issue, and that with much further training the overall accuracy begins to deteriorate. Unfortunately, detecting this requires the availability of ground truth. In the preceding section, we identified several related cues to the issues associated with overfitting that do not rely on ground truth. For example, by monitoring the behavior of various classifier characteristics such as tree growth rate, the occurrence of impure leaves, the impact of pruning on various tree characteristics, or the number of pixels that change their class label as a result of applying an additional set of training examples, it should be possible to derive a recommendation of when to stop training. This problem deserves further investigation.

The present discussion has been centered at the intuitive notion of the user training the classifier by correcting its mistakes. Here, training pixels are selected from among currently misclassified pixels. However, ITI may also alter a decision tree in response to a training pixel that is currently classified correctly. This may increase or decrease the overall accuracy of the classifier. It may well be that such training pixels are required in order to exceed the baseline performance of the batch-trained ITI classifier in the previous section. This issue has not been investigated yet. It is unclear how to select "good" training instances that are not currently misclassified.

This error-correcting training paradigm also raises a theoretical question in pattern classification: Classifiers usually make the assumption that training and test instances are drawn from the same distribution. This is generally not true if training examples are selected in the way we propose: Training pixels are *not randomly* chosen from the population of pixels in an image. On the other hand, since the user will select training examples with the aim to improve overall performance, it may be that with continuing training the distribution of training pixels will begin to resemble a random sample of the population of image pixels. We do not know how serious this discrepancy is, and what its implications on the training process are. This is a subject of future study.

Together with a suitable user interface, the ITL framework achieves very fast turnaround times between labeling training pixels and feedback from the updated classifier. According to our experience, the constraining factor now is the ability of the user to discern how individual pixels should be labeled. This is a matter of visualization. To aid the user, we are currently experimenting with animated 3-D fly-through visualization, which provides the user with terrain relief in addition to color information.

We have already indicated several open questions introduced by the ITL paradigm. To give a flavor of the variety of novel possibilities opened up by ITL, we will now discuss two examples of our current work in some detail.

5.1 Sequential Training on a Set of Images

In many applications, one wants to consider a set of training images in sequence, without going back to images considered earlier. For example, to classify a video stream, one would train a classifier on one frame at a time, one after the other. The ITL framework suggests itself as a useful tool for such a purpose, since it allows the user to supply training instances one at a time, correcting errors and refining the classifier. When classifier performance is satisfactory on one frame, the next frame is displayed, and training resumes. For reasonably constrained video data, we expect the training process to converge in the sense that as more frames are processed, the amount of training required on a frame will decrease. In the limit, no corrections are necessary at all, and the classifier can run stand-alone.

For lack of appropriate video data with ground truth, we did a preliminary experiment using the same dataset as in Section 4. Here, the large image was split into 10 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 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 7.

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 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 7 that this subimage is the only one where training did not result in a local improvement of the full image classification accuracy.

Table 2 summarizes some qualities of the best and last classifiers obtained during these two training sessions. Like in the previous section (cf. Table 1), the decision trees were very small yet accurate.

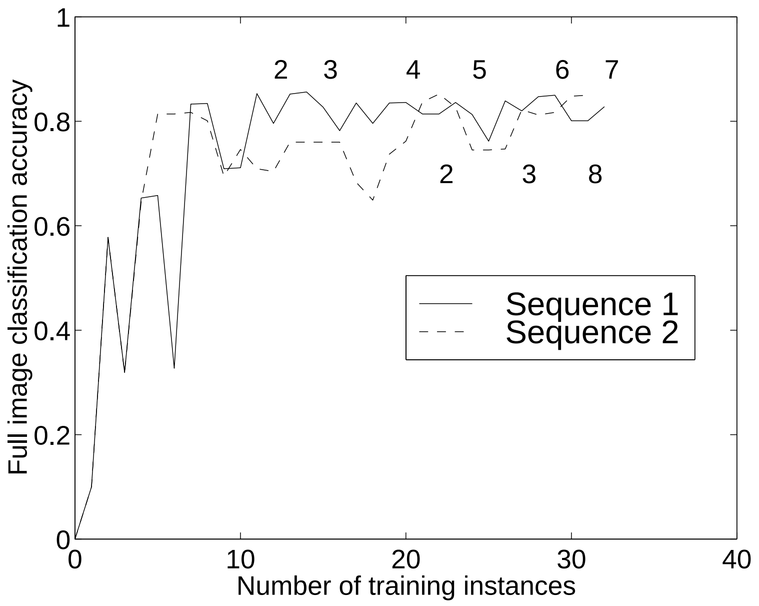

Figure 7: Classifier behavior during interactive training. The first part of each curve corresponds to the initial training, which is done on image 1 within each sequence. The numbers next to the graph mark new training images: Digit k next to a curve indicates that the user switched to subimage k prior to selecting this training instance. 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.

	Sequence 1		*Sequence 2*	
Classifier:	*best*	*final*	*best*	*final*
# Training instances:	14	32	22	31
% correct:	85.6	82.8	85.2	85.0
# tree nodes:	9	17	13	15
# features used (of 17):	4	7	4	5

Table 2: Summary of classification results after training on two image streams. In both sequences, the best classifier was attained during training on the second image in the sequence (cf. Figure 7).

5.2 Recognition of Simple Objects

Beyond the traditional classification problem described so far, our interactive method opens up additional possibilities. For example, we have introduced the concept of *hierarchical classification* [19]. The idea is that classification can be applied to the labels produced by lower-level classifiers. The classifiers are trained from the bottom up: First, an ordinary pixel classifier c^1 is trained, and produces a label image C^1. Then, a classifier c^2 is trained to classify C^1 to produce C^2 (Figure 8). In general, a classifier c^n is trained on any features computed of classification results output by any lower-level classifier. This is useful, for example, to extract spatial information from an image.

We obtained initial results on a task involving the recognition of black-back sea gulls on aerial photographs (Figure 15). On these images, the gulls are characterized by the *adjacency* of black and white blobs of characteristic sizes. Thus, the c^1 classifier is trained to label the appropriate black and white areas, such that the proportion of false negatives is small, even at the expense of many false positives. Then, a c^2 classifier is trained to recognize adjacent black and white blobs of the right sizes.

An example result [19] is shown in Figure 9. Here, the base level classifier c^1 employed commonly used color-based and statistical texture features. The c^2 classifier operated on two types of features computed on C^1, both of which consider a window centered at the pixel in question:

- The *adjacency* feature counts the numbers of occurrences of two given class labels, and returns their product. This returns a large value if and only if both labels are abundantly present near the current pixel.
- The *clutter* feature counts the number of vertical and horizontal label discontinuities between adjacent pixels. This returns small values only for rather uniformly classified regions.

Note that the window must be somewhat larger than the size of the structure sought after, since the maximum value returned by a feature is bounded by this window size.

In our example, the most important feature turned out to be an adjacency feature that returned the product of the numbers of pixels labeled by c^1 as "black" and "white" within a 5×5 window. This intuitively expresses the fact that, for our purposes, gulls are composed of black backs and white heads.

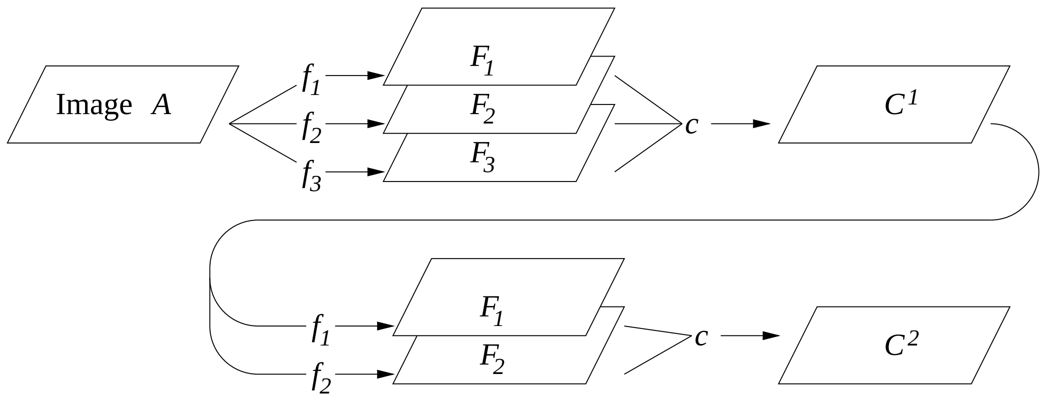

Figure 8: The concept of hierarchical classification: The output of one classification system serves as the input to another one. This can in principle be extended to any number of hierarchies. The feature extractors f typically differ from level to level, and generate feature planes F. A unique classifier c is trained at each level, generating a label image C, which is one of the inputs at the next level.

Many other nonstandard features in the hierarchy of classifiers are possible. Besides spatial relationships, they may express a variety of other characteristics. For instance, we are experimenting with features expressing the confidence in labels assigned at lower hierarchies.

We believe that this is only the tip of the iceberg of applications of interactive image classification. Many of these concepts are currently being applied in a large-scale environmental monitoring project in cooperation with the Department of Forestry of Wildlife Management at the University of Massachusetts at Amherst, which attempts to reproduce, extend, and simplify existing work in ground use mapping [24].

6 Conclusion

We have demonstrated a new interactive methodology for training of pixel classifiers. It is a very effective tool for selecting few but informative training instances, which helps reduce human labor and produces simple and accurate classifiers, while reducing the overfitting problem. These findings need to be further evaluated on more data. We also indicated further perspectives

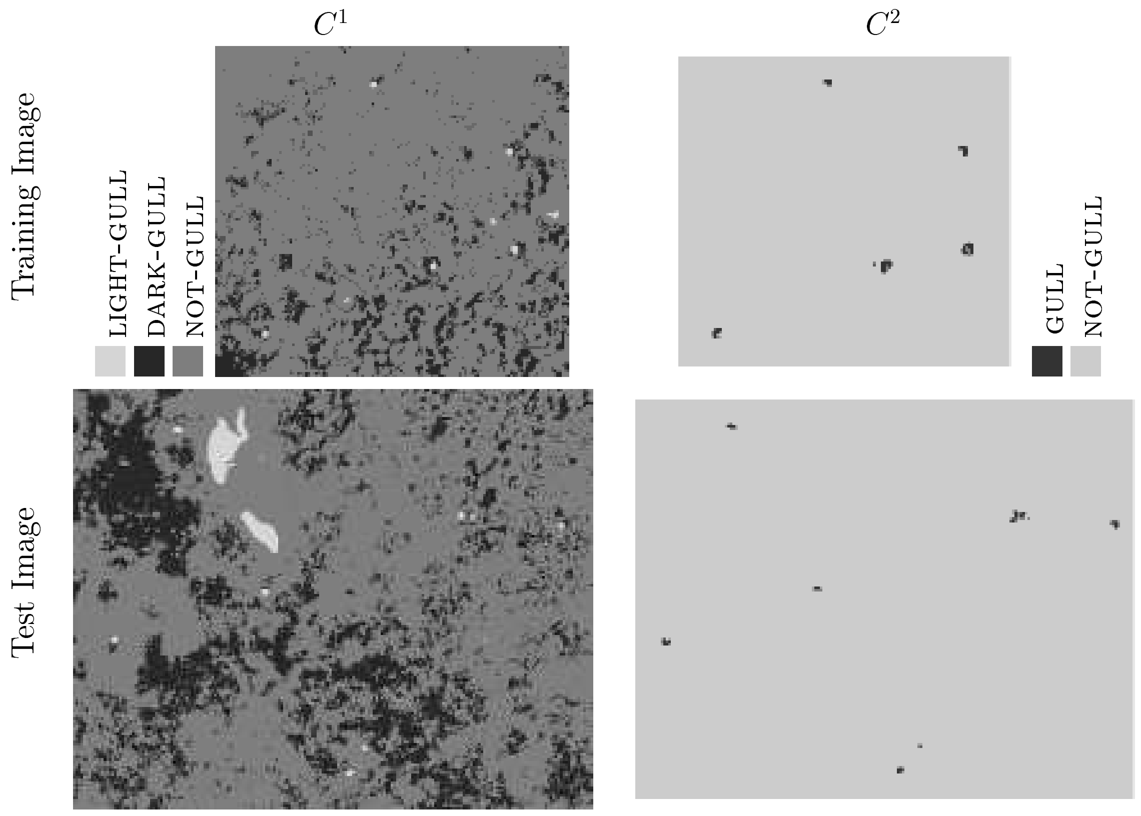

Figure 9: Results of hierarchical classification for black-back seagull localization (cf. Figure 15). Shown are the label images C^1 and C^2 for the disjoint training and test images. After cleaning up the C^2 results shown using a straightforward morphological opening operation, all gulls are found in both images with no false positives.

opened up by the ITL paradigm. We believe that the seagull detection example barely scratches the surface of potential applications of hierarchical classification. This is an area of continuing research in our laboratory.

Real-time feedback provided through a suitable user interface allows rapid training, on the order of a few minutes in our realistic examples. The efficiency of the feedback loop is not limited by the size of the training image.

To build interactive learning systems that update their parameters in real time, incremental learning algorithms are beneficial. The ITI classifier was chosen because of its capability to incorporate training instances incrementally, and because of the implicit feature selection property of decision trees. While it works well in our applications, more experiments with this and other classification algorithms will be performed on more complex tasks. Fast incremental learning algorithms are an open area of research with many potential applications for interactive learning systems.

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/`. 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, by the Army Research Office (via ARL) under grant number DAAG55-97-1-0026, by the National Science Foundation under grants IRI-9222766 and EIA-9726401, and by the National Fish and Wildlife Foundation under project #98-089. We would also like to thank two anonymous reviewers for their thoughtful comments that helped improve the paper.

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] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*, Wadsworth&Brooks, Pacific Grove, CA, 1984.

[3] N. W. Campbell, W. P. J. Mackeown, B. T. Thomas, and T. Troscianko, "Interpreting image databases by region classification," *Pattern Recognition* **30** 555–563, April 1997.

[4] 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** 308–325, March 1997.

[5] J. R. Carr, "Spectral and textural classification of single and multiple band digital images," *Computers & Geosciences* **22** 849–865, October 1996.

[6] R. Conners and C. Harlow, "A theoretical comparison of texture algorithms," *IEEE Trans. Pattern Anal. Machine Intell.* **2** 204–222, 1980.

[7] S. L. Crawford, "Extensions to the CART algorithm," *Int. J. Man-Machine Studies* **31** 197–217, 1989.

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

[9] J. du Buf, M. Kardan, and M. Spann, "Texture feature performance for image segmentation," *Pattern Recognition* **23** 291–309, 1990.

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

[11] J. D. Foley and J. Sammon, Jr., "An optimal set of discriminant vectors," *IEEE Trans. on Computers* **24** 281–289, 1975.

[12] W. Hafner and O. Munkelt, "Using color for detecting persons in image sequences," *Pattern Recognition and Image Analysis* **7** 47–52, 1997.

[13] R. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Trans. Systems, Man, and Cybernetics* **3** 610–621, 1973.

[14] 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*, 1996, pp. 2–7.

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

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

[17] B. C. Lovell and A. P. Bradley, "The multiscale classifier," *IEEE Trans. Pattern Anal. Machine Intell.* **18** 124–137, 1996.

[18] P. Ohanian and R. Dubes, "Performance evaluation for four classes of textural features," *Pattern Recognition* **25** 819–833, 1992.

[19] J. H. Piater, E. M. Riseman, and P. E. Utgoff, "Interactive training of pixel classifiers opens new possibilities," in *ISPRS ComIII Symposium on Object Recognition and Scene Classification from Multispectral and Multisensor Pixels*, Columbus, OH, July 1998, International Archives of Photogrammetry and Remote Sensing.

[20] J. R. Quinlan, *Programs for machine learning*, Morgan Kaufmann, 1993.

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

[22] J. C. Schlimmer and D. Fisher, "A case study of incremental concept induction," in *Proc. Fifth Nat. Conf. on Artificial Intelligence*, Morgan Kaufmann, Philadelphia, PA, 1986, pp. 496–501.

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

[24] 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*, American

Society for Photogrammetry and Remote Sensing, Bethesda, Maryland, pp. 87–101 and 308–312, 1996.

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

[26] P. E. Utgoff, N. C. Berkman, and J. A. Clouse, "Decision tree induction based on efficient tree restructuring," *Machine Learning* **29** 5–44, October 1997.

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

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

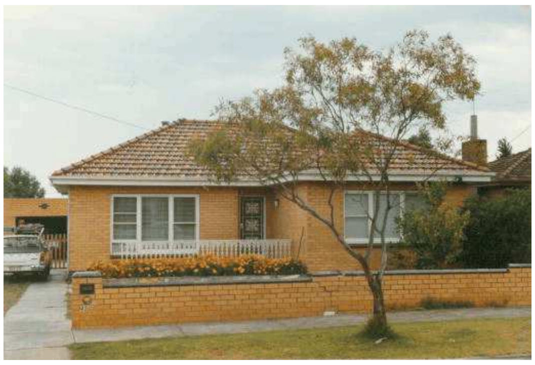

Figure 10: Example image (515 by 346 pixels)

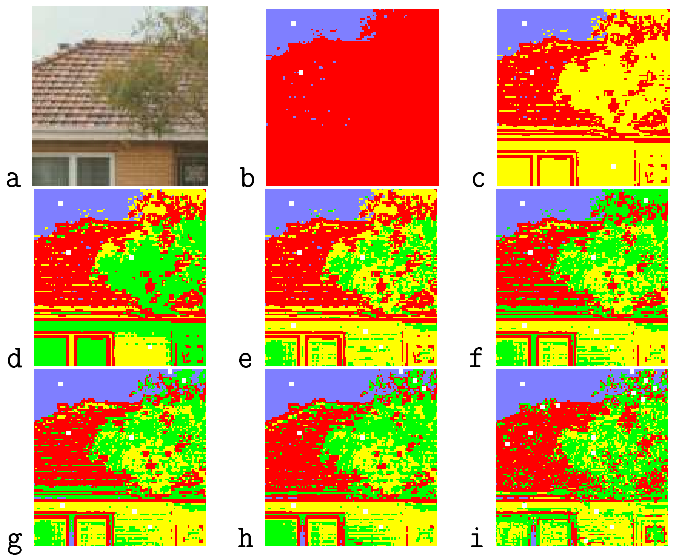

Figure 11: An example training session on a subimage (112 by 115 pixels) of Figure 10. Image (a) shows the original image. In the following images, the class labels are represented by mnemonic (but otherwise meaningless) colors: SKY blue, ROOF red, BRICK yellow, and FOLIAGE green. The tiny white squares represent 3 × 3 blobs of training instances provided by the Teacher's mouse clicks. Images (b)–(d) show the results after adding one set of training instances for each class, and (e)–(i) are snapshots during some refining.

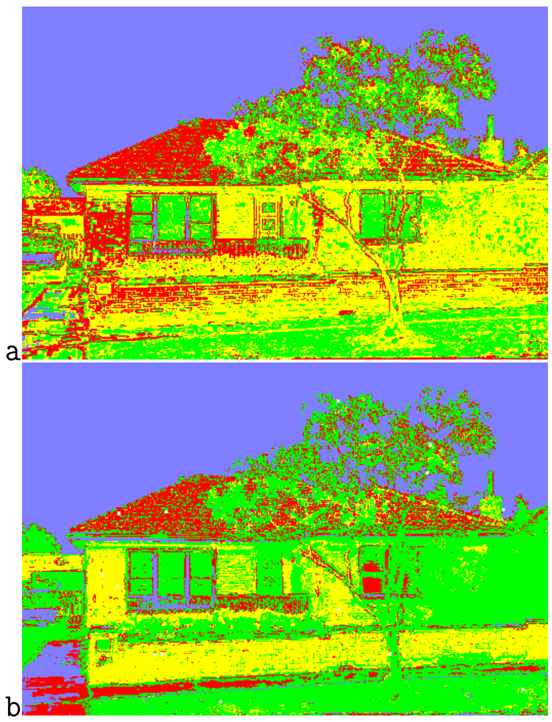

Figure 12: Results on the entire image (Fig. 10): (a) using the classifier from Figure 11i, (b) after adding a few additional squares of training instances. Note that the tree trunk was labeled as FOLIAGE, and that several areas were ignored during training, e.g. the windows and the pavement.

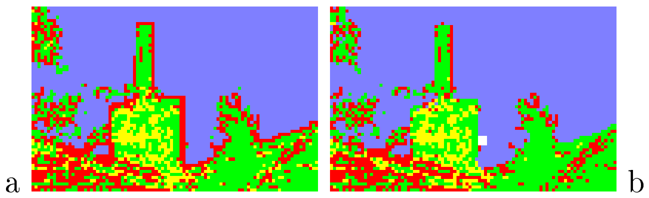

Figure 13: Generalization example. The border of misclassified pixels around the chimney (a) is almost entirely removed by adding one training square of the SKY class (b).

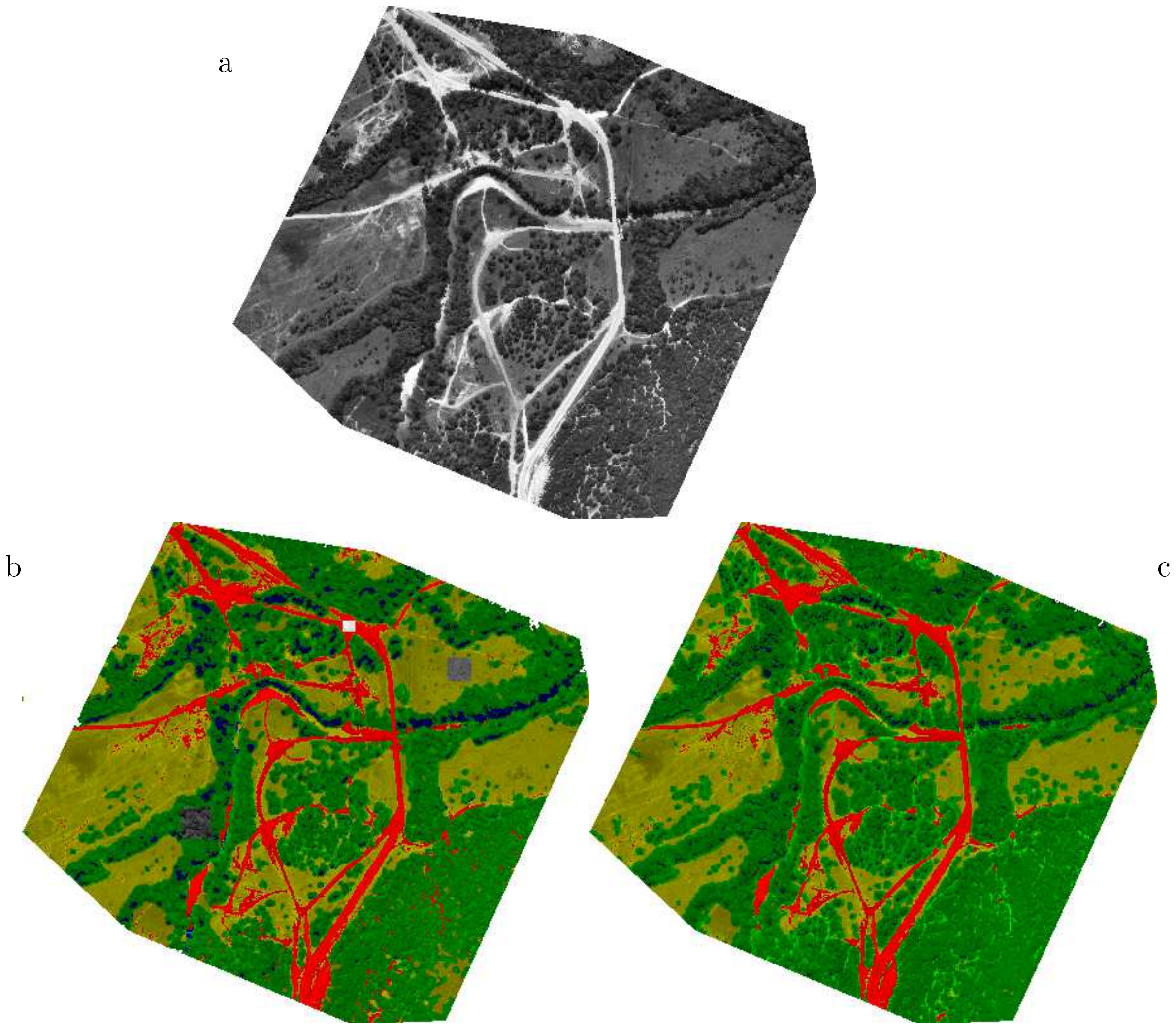

Figure 14: Ft. Hood scene (**a**, 1803 × 1591 pixels) and classification results using ITI, overlaid on the gray scale image: (**b**) Batch training on the same training set as in Wang et al.; (**c**) interactive incremental training. The class label of a pixel is visualized by generating a false color, where the hue corresponds to the class, and the intensity corresponds to the gray scale intensity of the underlying image. No classification results are provided for the training pixels, which in image b appear as gray squares. The two classification results are very similar; differences mainly occur in regions that are ambiguous even to a human.

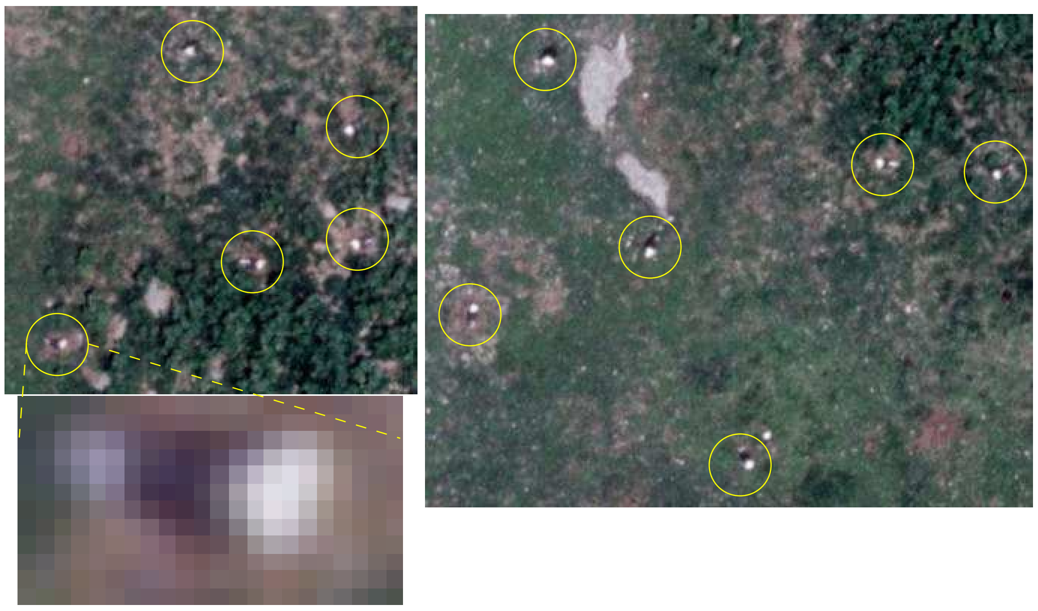

Figure 15: Aerial photographs showing sections of an island off the coast of Maine. Yellow circles indicate black-back seagulls. The left image (275×258 pixels) was used as a training image, the right one (405×328 pixels) as a disjoint test image. One gull is shown enlarged to reveal its white head (right), white tail (left) and black back (center).