

University of Montana

ScholarWorks at University of Montana

Graduate Student Theses, Dissertations, &
Professional Papers

Graduate School

2000

Comparison of three machine learning algorithms for automated feature extraction from digital images

William D. Bain

The University of Montana

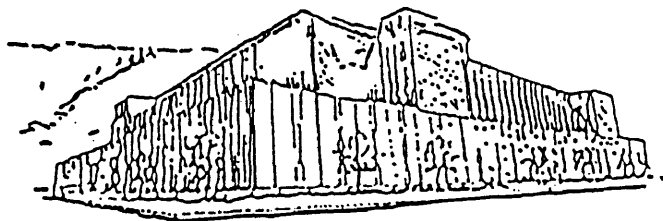
Follow this and additional works at: <https://scholarworks.umt.edu/etd>

Let us know how access to this document benefits you.

Recommended Citation

Bain, William D., "Comparison of three machine learning algorithms for automated feature extraction from digital images" (2000). *Graduate Student Theses, Dissertations, & Professional Papers*. 5101.
<https://scholarworks.umt.edu/etd/5101>

This Thesis is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact scholarworks@mso.umt.edu.



Maureen and Mike
MANSFIELD LIBRARY

The University of **MONTANA**

Permission is granted by the author to reproduce this material in its entirety,
provided that this material is used for scholarly purposes and is properly cited in
published works and reports.

*** Please check "Yes" or "No" and provide signature ***

Yes, I grant permission



No, I do not grant permission



Author's Signature William D. Pennington

Date 5/31/00

Any copying for commercial purposes or financial gain may be undertaken only with
the author's explicit consent.

**A comparison of three machine learning algorithms for automated
feature extraction from digital images**

by

William D. Bain, Jr.

BS, Mechanical Engineering, University of Washington

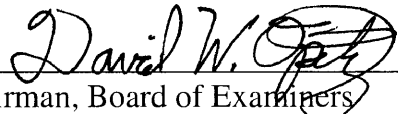
Presented in partial fulfillment of the requirements for the

Degree of Master of Science

The University of Montana

2000

Approved by:


Chairman, Board of Examiners


Dean of the Graduate School

6-1-2000
Date

UMI Number: EP40565

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI EP40565

Published by ProQuest LLC (2014). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code



ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

William D. Bain, Jr., MS 2000 Computer Science

DWJ

A comparison of three machine learning algorithms for automated feature extraction from digital images (pp. 48)

Director: David Opitz, Ph.D.

Extracting specific cartographic features such as roads or buildings from digital images has become an increasingly important problem. Traditional approaches are (1) to extract the features manually and (2) to create custom tailored computer programs for specific tasks. The problem with the former approach is that it is tedious and expensive, and the problem with the latter is that it is not flexible enough to apply to a variety of problems. The approach that we use in this thesis is to model the feature extraction process using statistical and machine learning techniques, specifically: artificial neural networks (ANN), K-nearest neighbor (KNN), and naive Bayesian (NB) classifiers. Each experiment compares a baseline input representation – a moving, square pixel window mapped directly to learning algorithm inputs – against a foveal representation – high pixel resolution at the center surrounded by regions of successively lower resolution. We apply these techniques to three different types of problems: (1) recognizing a specific linear feature (roads), (2) recognizing a landform object (volcanoes) and (3) ground cover classification.

Results indicate that machine learning can be used to model road extraction and classification of ground cover types effectively. It is not currently quite as effective at extracting large, complex objects such as volcanoes. In addition, we demonstrate that ANN and KNN classifiers are substantially more accurate for each type of problem than do NB classifiers, although ANNs are computationally more efficient than KNNs. Finally, our experiments show that foveal input representation provides results similar to those of baseline representation in most cases, while requiring less learning and classification time.

ACKNOWLEDGEMENTS

This work was supported in part by National Science Foundation grant IRI-9734419. I am extremely thankful for the help of Dr. David Opitz for providing funding and the inspiration for this study, the machine learning background, and for the countless hours of consultation and review of this document. I would like to thank Dr. Ray Ford for the Java software engineering background that I needed to implement VLS, and Dr. Hans Zuuring for his help with the statistical and GIS aspects of the project. Finally, I thank Wendy Parson for her endless patience and for motivating me to see this project through to completion.

TABLE OF CONTENTS

ABSTRACT	ii
ACKNOWLEDGEMENTS	iii
1. INTRODUCTION	1
2. BACKGROUND	4
2.1 <i>Learning Algorithms</i>	4
2.2 <i>Problem Representation</i>	7
3. OBJECTIVES	11
4. GENERAL METHODS.....	12
5. PROBLEM-SPECIFIC METHODS & RESULTS	16
5.1 <i>Helena Valley Experiment</i>	16
5.1.1 Helena Valley Methods	16
5.1.2 Helena Valley Results.....	18
5.2 <i>Presidio Experiment</i>	23
5.2.1 Presidio Methods	23
5.2.2 Presidio Results.....	26
5.3 <i>Magellan Venus Experiment</i>	30
5.3.1 Magellan Venus Methods	30
5.3.2 Magellan Venus Results	33
6. DISCUSSION & FUTURE WORK	41
7. CONCLUSIONS	46
8. REFERENCES	47
APPENDIX A: SOFTWARE DEVELOPMENT	A1
APPENDIX B: LEARNING ALGORITHMS.....	B1

1. INTRODUCTION

Automating the time-consuming task of extracting features from images has become an increasingly important problem. Researchers in diverse fields are finding that their ability to collect image data is rapidly outpacing their ability to analyze this data. For example, a major bottleneck in the data flow from digital images to geographic information system (GIS) applications is the extraction of specific cartographic features such as roads or buildings.

One traditional approach to feature extraction from images is to employ an expert to digitize images by hand. This poses many problems, including excessive cost, a considerable investment of time and inconsistent results. To catalog the one million or so small volcanoes pictured in synthetic aperture radar (SAR) images of Venus, for example, would require approximately ten years of mundane work by planetary geologists (Burl et al. 1994, Smyth et al. 1995). Another traditional approach is to employ a computer analyst to create a computer program that is custom tailored to the task at hand. However, this approach suffers from the fact that it is not flexible to changing conditions under which the imagery was acquired. For instance, one may need to rewrite a road finding algorithm that works well with images taken in springtime in order to find roads in exactly the same area in autumn.

An alternative and more recent approach is to model the feature extraction process using statistical and machine learning techniques rather than explicitly encoding the feature

concept (Maloof et al. 1998, Burl et al. 1998). The idea behind this approach is that the user provides to the learning algorithm (or classifier) a sample of extracted features from the image. The classifier then automatically develops a model that correlates known data (e.g., pixel values from images, stacked terrain data, etc.) with targeted outputs (i.e., the extracted features). The learned model then automatically classifies and extracts the remaining features. Of course, this relatively new approach to image recognition still requires the services of experts in the particular problem domain of interest. However, substantially less time is required of these experts when providing training examples than would be spent in manually classifying entire data sets. Researchers have used machine learning for many years for data mining and model induction in non-image-oriented problem domains, but its use for image recognition is new. This study presents an empirical comparison of three learning algorithms applied to automated image recognition – artificial neural networks (ANN), K-nearest neighbor (KNN), and naïve Bayesian (NB) classifiers (Rumelhart et al. 1986, Rumelhart et al. 1995, Mitchell 1996).

Concise problem representation for a learning algorithm is critical when creating accurate models. When learning to recognize objects in images, a typical difficulty with problem representation is taking into account spatial information without overwhelming the classifier. We address this challenge with the use of foveal input patterns. In foveal vision, there is high resolution at the center of the field of view and low resolution at the periphery. We test foveal input mappings with each of the three learning algorithms on three different problems – identifying volcanoes on Venus, roads in Montana’s Helena

Valley, and five mutually exclusive ground cover classes in San Francisco's Presidio military base. The results of this study demonstrate the potential of this new type of image learning representation.

2. BACKGROUND

Two central issues when applying machine learning to a problem domain are: (1) which type of learning algorithm provides the appropriate inductive bias for the learning task, and (2) what input representation is most appropriate for effective learning. This section introduces these concepts and discusses their importance in image learning.

2.1 Learning Algorithms

A system that learns from a set of labeled examples is an inductive learner. The set of labeled examples given to a learner is the training set. A teacher provides the output for each example. The goal of inductive learning is to generate from the training set a concept description that correctly predicts the output of all future examples, not just those from the training set. Many previous studies have shown the effectiveness of inductive learning algorithms, primarily for use in learning non-image-oriented concepts (Quinlan 1986, Rumelhart et al. 1986). These algorithms differ both in their concept representation language and in their method (or bias) for constructing a concept within this language. These differences are important since they determine which concepts a classifier will induce. For instance, some are more robust to noisy input data, some are well suited for real-valued inputs, and so on. The user should be able to choose the type of learning algorithm to match the task at hand. This thesis presents results from ANN, KNN and NB learning algorithms (refer to Appendix B).

Artificial, feed-forward neural networks consist of nodes interconnected by weighted links. These networks propagate a set of input signals (representing an example's feature values) forward into a set of output signals that serve as the prediction of the network. Nodes that are neither input, nor output nodes are called hidden nodes and their sole function is to help map input values to output values. The nodes in the network contain an activation function that allows the network to make a non-linear prediction. Training a network consists of modifying the interconnection weights via a learning algorithm such as backpropagation. Previous studies have successfully applied neural networks to a variety of real-world domains (e.g., Pomerleau 1991, Roscheisen et al. 1991), making neural nets a logical choice for consideration for the task of learning features in digital images.

A Bayesian classifier calculates probabilities of various classifications according to Bayes theorem. The calculation involves (1) the prior probability of each class, (2) the probability that each class occurs in conjunction with the observed data, and (3) the probability that the observed data occurs, independent of the classification. One can easily estimate the latter two quantities based on the measured frequency of similar instances in the training set.

Bayesian learning is computationally efficient at training and classification, but it has its limitations. The number of attributes is often quite high, and some combinations of attributes may be rare in the training data. Because of this, the population of similar training set examples is often not adequate to provide a good estimate of the probability

of each new instance. For example, if a particular combination of attributes never occurs in the training set, then a Bayesian classifier will return an estimated probability of zero for that instance.

Hence, in practice, a naïve Bayesian (NB) classifier – which makes the naïve assumption that attributes are probabilistically independent – is more useful. This means that the probabilities of each attribute value/class conjunction can be multiplied together (along with the prior probabilities of the attribute values) to estimate the probability that the entire set of attributes would lead to the given classification. The inductive model simply selects the most probable outcome. Despite these somewhat unrealistic assumptions, NB classifiers are quite effective for a variety of problems (Domingos & Pazzani 1996).

For many problems, it is not possible to calculate prior probabilities of the occurrence of each attribute value. In this case, one can make the simplifying assumption that the prior probabilities for each attribute value are uniform. In addition, one can significantly reduce the number of probability calculations using position-independent probabilities (Joachims 1996).

Nearest neighbor classifiers, unlike neural networks and Bayesian classifiers, do not create an explicit model of the function. Instead, they simply store the training instances and postpone generalization beyond these examples. The learning algorithm then classifies each new instance by examining its relationship to the stored training examples. For instance, K-nearest neighbor gives each new instance the most common classification

of the k nearest training instances, using a distance measurement such as Euclidean distance. Distance-weighted KNN is a special variant that weights the votes of the k nearest neighbors, usually in inverse proportion to their squared distance from the new input. This variant is robust to noisy training data and is effective for many practical problems (Duda & Hart 1973).

2.2 Problem Representation

For machine learning in general, representing the problem meaningfully to the learning algorithm is of the utmost importance. While an ANN, for example, can theoretically learn complex concepts, in practice only well-represented inputs provide useful results. The obvious approach of mapping individual pixel values to ANN input units might not be sufficient, especially where the objects to be recognized are of differing sizes and orientations. Volcanoes, for example, come in varying sizes and are recognized (at least by the human eye) by the shadows and highlights of not only the crater, but also of the cinder cone surrounding the crater. Real world images always contain noise and artifacts that can confuse the classifier, so in many cases, smoothing or blurring of portions of the image may be beneficial to the learning process. While it may take maximum resolution to recognize a small crater, the much larger surrounding cinder cone is more easily discernable at lower resolution; hence, a variable resolution or multi-scale approach to recognition seems appropriate.

Figure 1 illustrates our baseline approach for problem representation – directly mapping pixel values from a square, moving window on the image to classifier inputs. To train the classifier, the algorithm clips a square region surrounding an example feature from the image and scales the pixel values to an appropriate interval. The learning algorithm processes this scaled pixel data and compares the output against the appropriate pixel in a target image mask that reflects the locations of the features of interest. This baseline method is simple, but for large features (more than a few pixels across) the number of inputs required can overwhelm the classifier. In addition, this baseline representation does nothing to reduce the effect of noisy pixel data that can make the feature extraction concept difficult to induce.

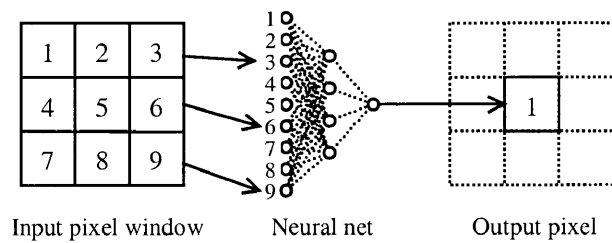


Figure 1: Example of baseline input representation

Figure 2 presents examples of our alternative input representation model: recursive, foveal window patterns. Foveal vision, exhibited by humans and other animals, involves the use of high densities of optic nerves at the center of the field of view, and lower densities at the periphery. Aside from the evolutionary advantages of this ocular arrangement, it seems ideally suited for image classifiers that must necessarily examine

small regions of an image at a time. The user is generally interested in whether a particular object is centered on the location in question. If the exact size of the feature is a priori unknown, then some blurring at the periphery of the window ought to make the recognition task easier.

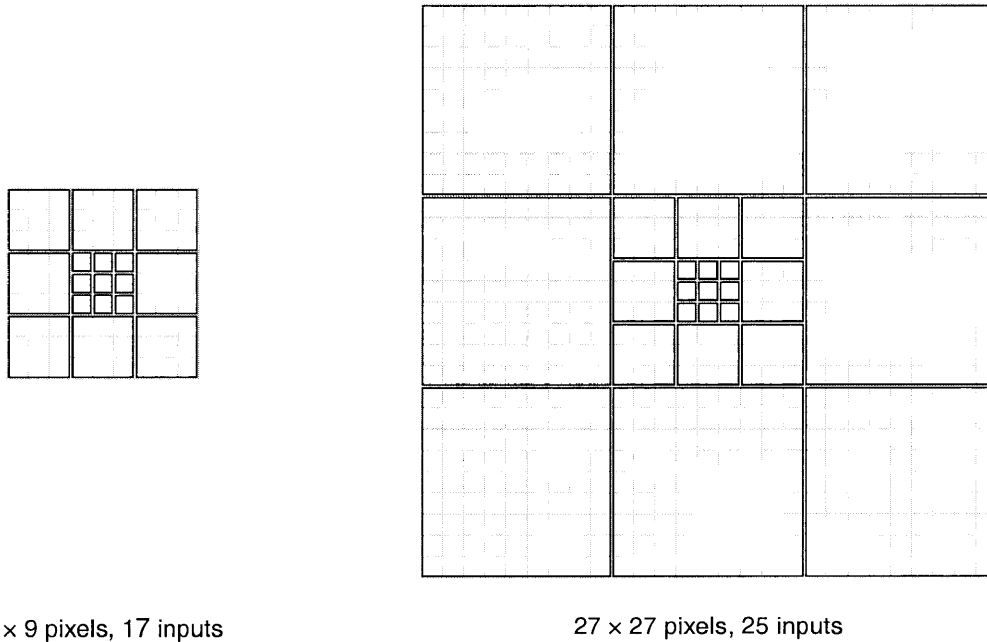


Figure 2: Example foveal input patterns

Foveal window representation is a type of multi-scale approach that offers numerous distinct advantages over the baseline method. First and foremost, foveal representations allow the classifier to take into account some spatial characteristics of the image without overloading it with too many irrelevant features. Second, any noise in an image at maximum resolution often disappears at lower resolution. Third, a classifier that recognizes small objects at one resolution ought to be able to recognize larger objects as

it zooms out on the image, hence this representation can be inherently more scale invariant.

3. OBJECTIVES

The general objective of this study is to demonstrate that machine learning can simplify feature extraction from images. By this, we simply mean that machine learning can provide classification accuracy comparable to that of human experts while requiring far less time and effort than traditional approaches. Extracting cartographic features from aerial or remotely sensed images ought to be generally amenable to learning by example, and if so, would certainly be much more easily and quickly accomplished with machine learning than by expert or custom software methods. This is not to suggest that all machine learning approaches can be successfully adapted to image recognition tasks. However, we presume that if the machine learning approach holds any promise, then at least one of the most common types of machine learning algorithms will probably succeed. Hence, our specific objectives are:

- Develop learning software to permit automated feature extraction from digital images using three learning algorithms, namely: artificial neural network (ANN), K-nearest neighbor (KNN) and Naïve Bayes (NB).
- Compare the classification accuracy of these three learning algorithms with two types of input representation on three different types of feature extraction problems: roads, volcanoes and complete, five-way ground cover classification.

4. GENERAL METHODS

The baseline input representation – mapping pixel values directly to learning algorithm inputs – is simple and sufficient for certain rudimentary image processing applications (e.g., edge detection). However, it is probably not sophisticated enough to be used for extracting large or complex features from images. Because of the possible shortcomings of the baseline approach, this study also investigated foveal patterns as a possible improvement to baseline representation. In creating foveal input patterns, the input value for each cell in the window was equal to the average of the pixel values in the cell. Since the sliding window scheme employed in the experiment necessitates repeatedly re-sampling the same image coordinates, we gained some speed (at the expense of increased memory requirements) by applying an area-averaging convolution filter to entire image rasters. We temporarily stored these filtered rasters so that entire foveal cells could be accessed with a single pixel index and without need of further arithmetic.

In each experimental treatment, we chose the input window size roughly to match the expected width of the target features. To assess the effectiveness of a foveal representation, we tested each foveal pattern against baseline patterns having (1) a similar number of inputs to the learning algorithm and (2) a similar size, wherever practicable.

Each of the three learning algorithms – ANN, KNN and NB – were tested on each of three diverse feature extraction problems – road detection, volcano detection, and ground cover classification. This study used typical machine learning algorithms almost

verbatim from the literature (e.g., Mitchell 1996), but for maximum efficiency in processing large images, we re-implemented each one in our Visual Learning System (VLS – refer to Appendix A).

The wide scope and inherent complexity of our experimental design complicated the determination of an optimal set of learning parameters. Three types of learning, two types of input representation and three problem domains require 18 separate experimental treatments. To optimize at least two parameters per classifier would have required multiplying the number of treatments by another factor of nine, which would have been prohibitively time and memory consuming. Because the purpose of this experiment is not to explore algorithm sensitivity to choice of learning parameters, we tried a range of parameters initially for each learning algorithm. We then used the apparent best ones for the remainder of the experiment.

ANNs used a learning rate = 0.05, momentum = 0.1 and a single hidden layer topology. The number of hidden nodes was equal to the square of the average of the square roots of the input and output layers. For binary classifications (e.g., road versus not road), each ANN used a single output node. For N-way classification (as in the Presidio experiment), the number of outputs was equal to the number of classes and the maximum output value determined the class. The number of epochs was initially set at 100 and increased until test set accuracy appeared to level off.

Each KNN classifier used $k = 5$ neighbors, with inverse squared distance weighting. We applied no scaling to the axes of the input space. Because KNNs do not derive a concise model of the learned concept from the training examples, they require more time than the other classifiers. By simply storing the entire training data set as the model, they are instantaneous to train, but slow to classify new instances. To classify each pixel, the learning algorithm compares distances from the input vector for that pixel to each of the training example input vectors, and then selects the closest K of these examples to classify the pixel. While this was not as time consuming as sorting the entire set of distances, it was still slow in comparison to ANN and NB classification. This lazy classification scheme was computationally intensive enough that the classifier could accommodate relatively few training examples. For this reason, we used no more than one percent of the pixel data for training.

For all but the NB classifiers, we sampled a small subset of the pixel data for training. In the Helena Valley road data set and the Magellan Venus data set, the targeted features comprised a small proportion (less than one percent) of the image. In these cases, we selected for training all of the available positive instances plus a randomly sampled one-percent of the negative instances in the training image tile set. In the case of complete classification of the Presidio data set, we trained on a randomly sampled one-percent of the available training data.

Due to the inherent efficiency of NB classifiers, they used a much larger training set than did the other learning algorithms. Physical memory was the only practical limit on the

size of the training data. A NB classifier can accommodate all the data in a mega-pixel image in about the same time required for an ANN to learn on one percent of the same image and then classify it. Even when training on entire images, there were usually so many attributes and possible attribute values that there was a high incidence of zeros among the calculated attribute value/target pairs, resulting in biased underestimates of the actual probabilities. For this reason, we clustered the attribute values linearly, from the original 256 values per band down to 16 values, and used an m-estimate of probability (with uniform prior probabilities). In each case, the equivalent sample size m was equal to the number of clustered attribute values.

Each image learning experiment involved a “leave one out” type of cross validation. For training, we left out each image tile in turn, and the resulting trained classifier was then used to classify the left-out tile. In this way, a classifier never classified its own training data.

5. PROBLEM-SPECIFIC METHODS & RESULTS

The three problem domains that this thesis explores differ from one another in the type of data available and the goal of the experiment. To elucidate these differences, the results comprise three sections, each of which describes a single problem domain. Each of these three sections begins with (1) a subsection describing the particular issues pertinent to the data set and ends with (2) a subsection that presents the accuracy and effectiveness of image learning in the problem domain.

5.1 Helena Valley Experiment

5.1.1 Helena Valley Methods

Figure 3 shows an eight-bit panchromatic image of the Scratch Gravel region of Montana's Helena Valley at 5-meter resolution. The goal is to extract the primary road network in this valley.



Figure 3: Excerpt from Helena Valley road data set

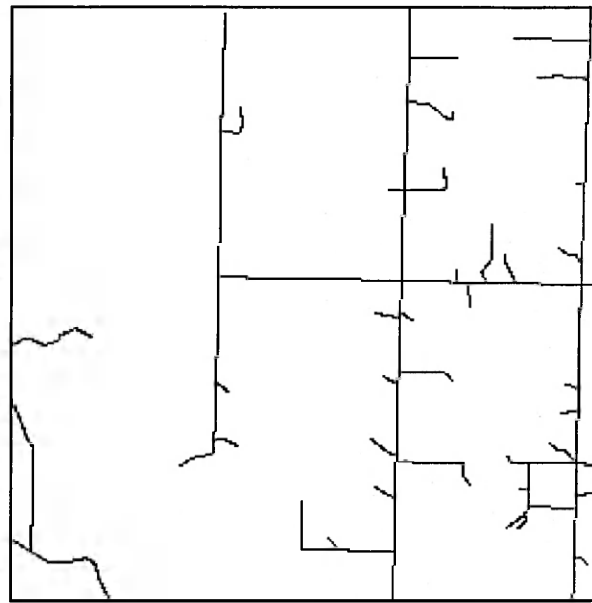


Figure 4: Nominal ground truth classification of road image

At the given resolution, most roads appeared only about three to five pixels wide. Hence we selected the smallest recursive foveal input window: a 3×3 square of single pixels surrounded by eight 3×3 pixel regions (17 inputs total – see Figure 2). The resulting pattern is nine pixels wide – more than wide enough to span the entire road and to include the edges of the road for context. For comparison, we chose the baseline window with the most similar number of inputs: 5×5 pixels (25 inputs). This choice of baseline pattern spans a smaller pixel width than the foveal pattern, yet results in a greater number of inputs to the learning algorithm. To control for effects of input window size, we also used a 9×9 baseline representation (81 inputs). We applied each of the three input representations in conjunction with each of the three learning algorithms. Finally, we filtered the output of each classifier using a popular hierarchical aggregation scheme known as Merge (Ford et al. 1997). We determined the optimum to-be-merged (TBM)

tolerance experimentally. For road detection, we found that a 20-pixel TBM tolerance was optimal.

Ground truth was not available for the image data. Instead, we used our best guess at a true classification of road pixels, produced by hand-digitizing two tiles cropped from the original image, each approximately 1109 pixels wide by 778 pixels high (8.63×10^5 pixels). Since the edges of the roads appeared indistinct, we made no effort to digitize the full road width. Instead, ground truth consisted only of the centermost single pixels along each roadway (see Figure 4). To make the results unambiguous, we eliminated from consideration any positively classified pixels within a two-pixel region surrounding the centerline pixel set before analysis.

The road finding experiments used two-fold cross validation: a classifier trained on examples from the first of the two tiles, classified the second tile, then trained on the second and classified the first.

5.1.2 Helena Valley Results

Figure 5 through Figure 8 show examples of our best experimental results for locating roads. ANN and KNN results are similar; each type of learning algorithm correctly located the majority of the roads, with few false positives (i.e., reported road locations where there is no road). The ANN learning algorithm provided slightly better accuracy than did the KNN, and for both types of learning algorithms, foveal representation is comparable to the baseline representation. The NB classifier performed poorly. While it

has a low false negative rate (i.e., it did not miss very many of the roads), it has an extremely high false positive rate, erroneously reporting some cultivated areas as roads.



Figure 5: Typical ANN results on road image using 5×5 baseline representation

Figure 6: Typical ANN results on road image using 9×9 foveal representation.

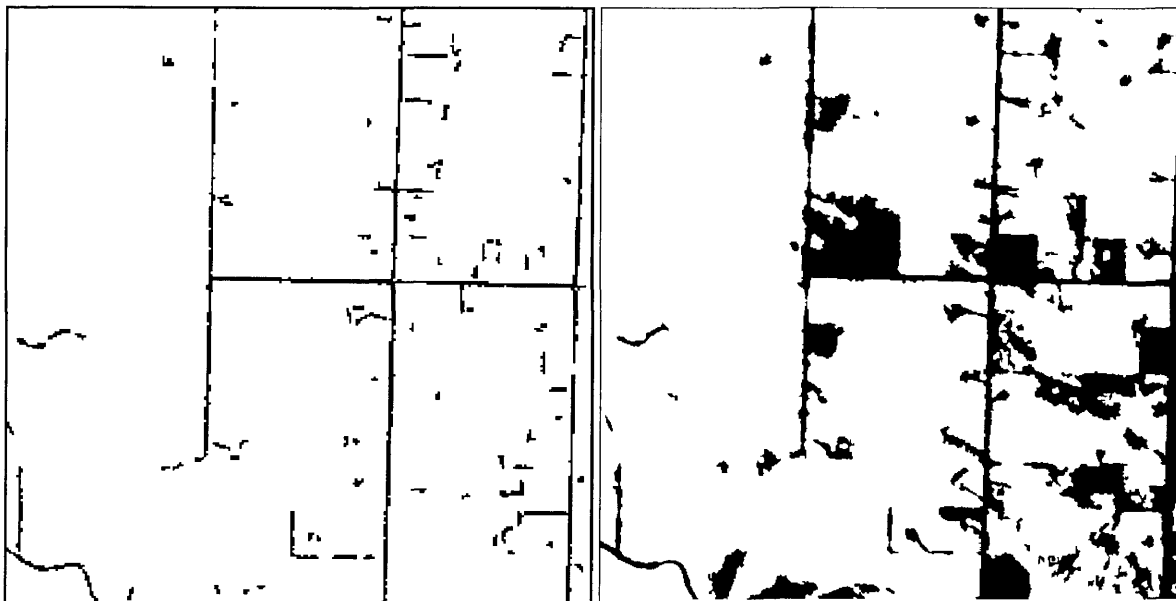


Figure 7: Typical KNN results on road image using 9×9 foveal representation.

Figure 8: Typical NB results on road image using 9×9 foveal representation.

Figure 9 compares overall accuracy of the various learning algorithms for road detection. Because the costs of misclassification are unknown, we present the results as Receiver Operating Characteristic (ROC) curves (Swets 1988). These curves show true positive rate versus false positive rate over a range of costs. True positive rate (TPR) is the number of true positives (found road centerline pixels) divided by number of true instances (all road centerline pixels). False positive rate (FPR) is the number of false positives (noise) divided by number of false instances (non-road pixels). The cost parameter in this case is the threshold applied to the outputs.

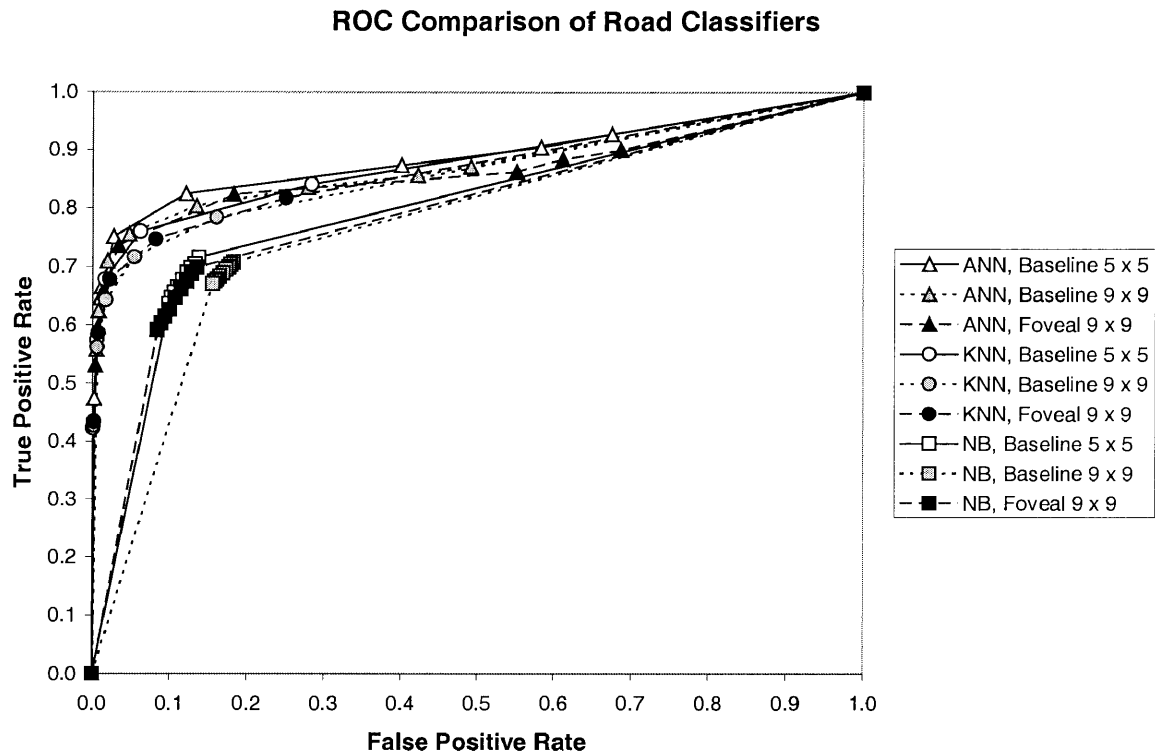


Figure 9: ROC comparison of road classifiers by learning algorithm and input representation

For the purposes of this experiment, a true positive is coincident with a roadway centerline pixel. Since most roadways appear to be about three to five pixels wide at the given resolution, any positives within two pixels of a road centerline pixel (but not on the centerline) were eliminated from consideration. Positives further than two pixels from a centerline pixel are false positives (errors of commission). Any centerline pixels that are classified negative are false negatives (errors of omission).

The ROC curve illustrates the inescapable tradeoff between effectiveness at locating a high proportion of the roads, and accuracy in reporting only road pixels with a minimum of false positives (noise). With a low threshold applied, a classifier generally returns a

large portion of the image as positive. While this reduces the chance of missing actual roads, it generally results in a high FPR. With a high threshold, almost nothing is classified positive. This gives a low FPR, but also a low TPR (i.e., misses many actual roads). An ideal classifier would find all the roads ($TPR = 1$) and nothing else ($FPR = 0$). Hence, the better classifiers are ones whose curves push toward the upper left corner of the chart. Since the cost of misclassification is unknown, the best way to compare classifiers across a range of costs is to compare their areas under the ROC curve, as shown in Figure 10.

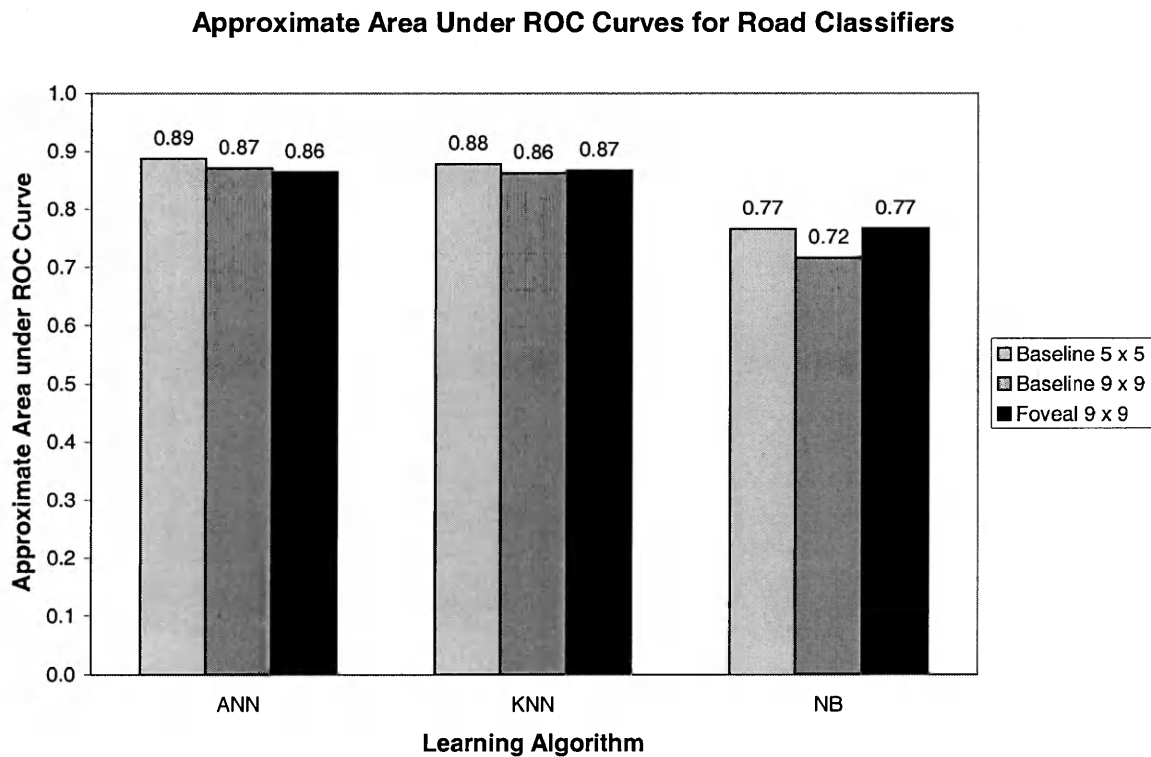


Figure 10: Approximate area under ROC curves for road classifiers by learning algorithm and input representation

Surprisingly, the small, 5×5 baseline representation, which barely spans the roadways, provides slightly better results than do the others. One possible reason for this may be that an overabundance of confounding data offset any potential value of spatial context surrounding the road. Machine learning researchers call this effect the curse of dimensionality, and nearest neighbor classifiers are especially susceptible to it (Mitchell 1996). The NB classifier scored lower than the ANN and KNN, primarily because of the high number of false positives. The false positive rates reported here are perhaps slightly higher than actual, since the nominal “ground truth” classification that we used is probably not quite complete; when manually digitizing questionable roads, we chose to err on the conservative side.

5.2 Presidio Experiment

5.2.1 Presidio Methods

Figure 11 is an excerpt from a mosaic of four-band images of the Presidio, a former military base in San Francisco. A near infrared (NIR) band was included along with the red, green and blue (RGB) image data (shown here in grayscale). For this image data set, the task is to classify each pixel as belonging to one of the following five classes:

- Low structure (roads and parking lots)
- High structure (buildings and bridges)
- Low vegetation (grass, dirt and sand)
- High vegetation (trees and shrubs)
- Water

Figure 12 illustrates the true classification provided by the NASA Jet Propulsion Laboratories (JPL) for the five classes listed above. There are obvious discrepancies between the image and the classification. For example, artifacts of the mosaic construction of the image make some features appear to have shear discontinuity in the original image (e.g., the large building, right side, just above center). Curiously, the classification shows no such artifacts. Regardless, the nominal ground truth provided is sufficiently accurate for training purposes. Opitz et al. estimated the error in this data set to be about 10%. Machine learning, by its very nature, is able to accommodate a certain degree of error in the training data. ANNs in particular are robust to noisy data.

For the Presidio images, we used relatively small input representations. The reason for this is that the image data pixel depth was greater here (red, green, blue, and near infrared) than in the other data sets (single band, grayscale). The greater pixel depth would have made larger input representations too computationally intensive for timely learning and classification.



Figure 11: Excerpt from Presidio image



Figure 12: Nominal ground truth classification

Based on the above consideration, the Presidio experimental treatments started with the smallest of the recursive foveal representations (9×9 pixels, 17 inputs). To control for number of inputs, a 5×5 pixel baseline representation (25 inputs) was used and to control for window size, a 9×9 pixel baseline representation (81 inputs) was included.

The Presidio image data originally comprised a single mosaic approximately 9000 pixels wide by 6000 pixels high (approximately 45 million pixels total). This volume of image data is about ten times greater than what VLS can accommodate at one time on a modern high-end desktop PC (e.g., a 500 MHz Pentium III with 256 MB RAM). For our experiment, we cropped out two adjacent mega-pixel image tiles, each containing representative proportions of the five classes, for two-fold cross validation. We then applied a range of thresholds to the output, and finally used the Merge algorithm with a 40-pixel TBM tolerance to filter noise from the resulting classification.

5.2.2 Presidio Results

Figure 13 through Figure 16 show typical results obtained from each of the three types of learning algorithms.



Figure 13: Typical Presidio ANN results using 9×9 foveal representation



Figure 14: Typical Presidio ANN results using 9×9 baseline representation



Figure 15: Typical Presidio KNN results using 9×9 foveal representation

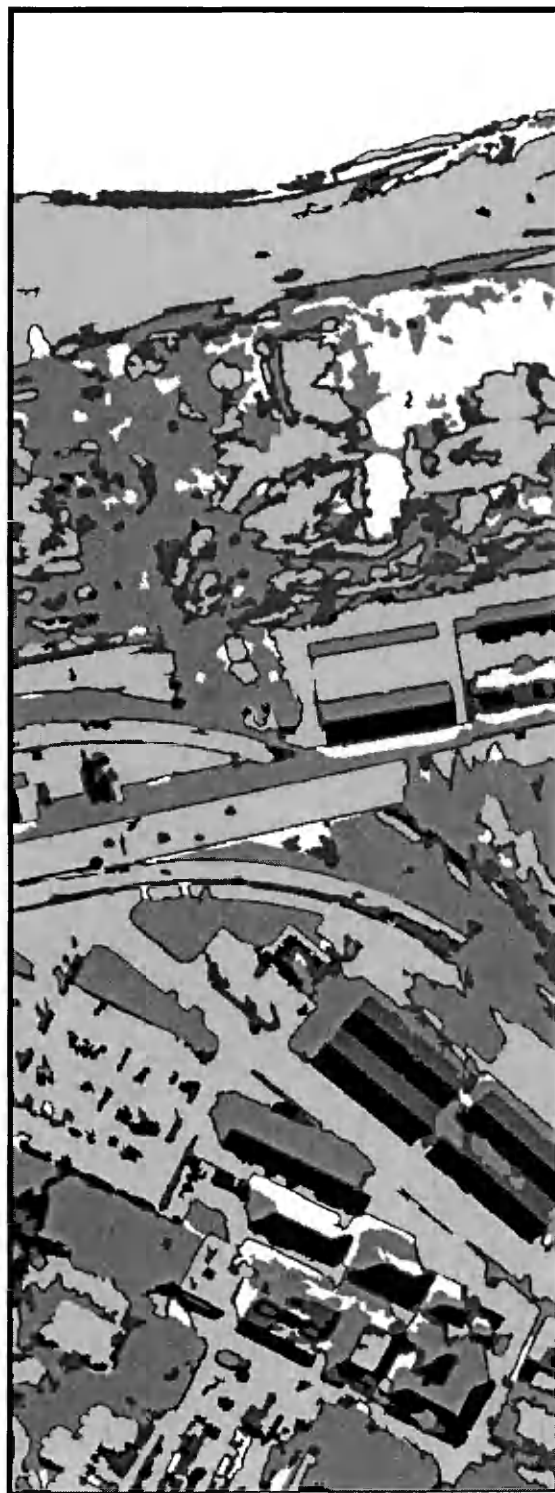


Figure 16: Typical Presidio NB results using 9×9 foveal representation

Figure 17 compares overall classification accuracy obtained from each of the three types of learning algorithm. We define accuracy as the proportion of the classification comprising the true positives of all five classes (i.e., the sum of the diagonals in a confusion matrix analysis).

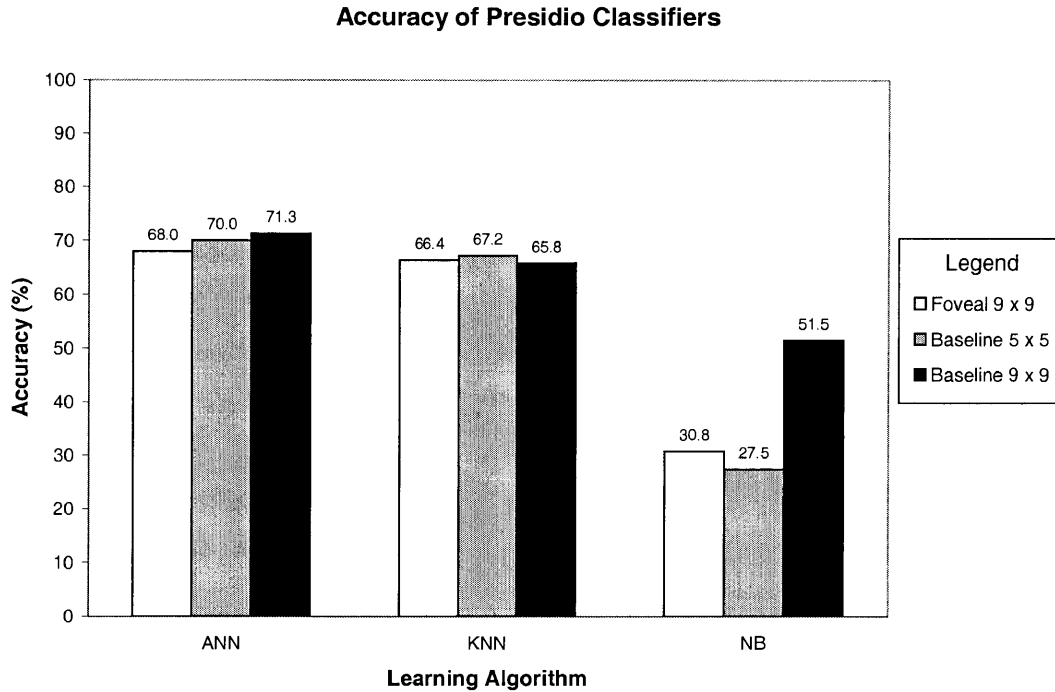


Figure 17: Overall accuracy of Presidio classification by learning algorithm and input representation

Results were far better for ANN and KNN than for NB classifiers. The ANNs performed better the more inputs they had, while the KNN classifiers were insensitive to the type of input representation. The erratic results from the NB classifiers are most likely due to insufficient training data to estimate probabilities accurately and imbalances in the data. Despite the use of equivalent sample sizes and m-estimates of probability, the NB classifiers would often completely ignore one or two of the five classes. If the NB

classifier happened to recognize one of the majority classes (e.g., low vegetation or low structure), then the overall accuracy increased, albeit still significantly lower than for the other two learning algorithms.

5.3 Magellan Venus Experiment

5.3.1 Magellan Venus Methods

Figure 18 shows an example of the grayscale, synthetic aperture radar (SAR) images of the surface of Venus that NASA's Magellan space probe collected as it circled the planet. The entire data set collected during the mission comprises approximately 30,000 images, each 1024 pixels square (approximately one million pixels) with a pixel resolution of 75 m. For the purposes of this experiment, NASA JPL provided only four images. The task is to be able to locate each of the volcanoes, but to reject the impact craters, which are superficially similar in appearance.

NASA JPL provided the coordinates and sizes for the circles shown in the figure, which represent the "ground truth" classification. The extent of each circle indicates the extent of the cinder cone surrounding each volcanic crater. Since actual ground truth on Venus may not be available for some time, this nominal ground truth is actually the consensus among planetary geologists who digitized these image tiles by hand. Although not visible in the figure, each of the examples were classified according to level of certainty: (1) definitely a volcano, (2) probably, (3) possibly, and (4) only a pit is visible. The last

class was used to label non-volcanic impact craters; hence, the specific learning goal is to locate all instances of the first three classes listed above.

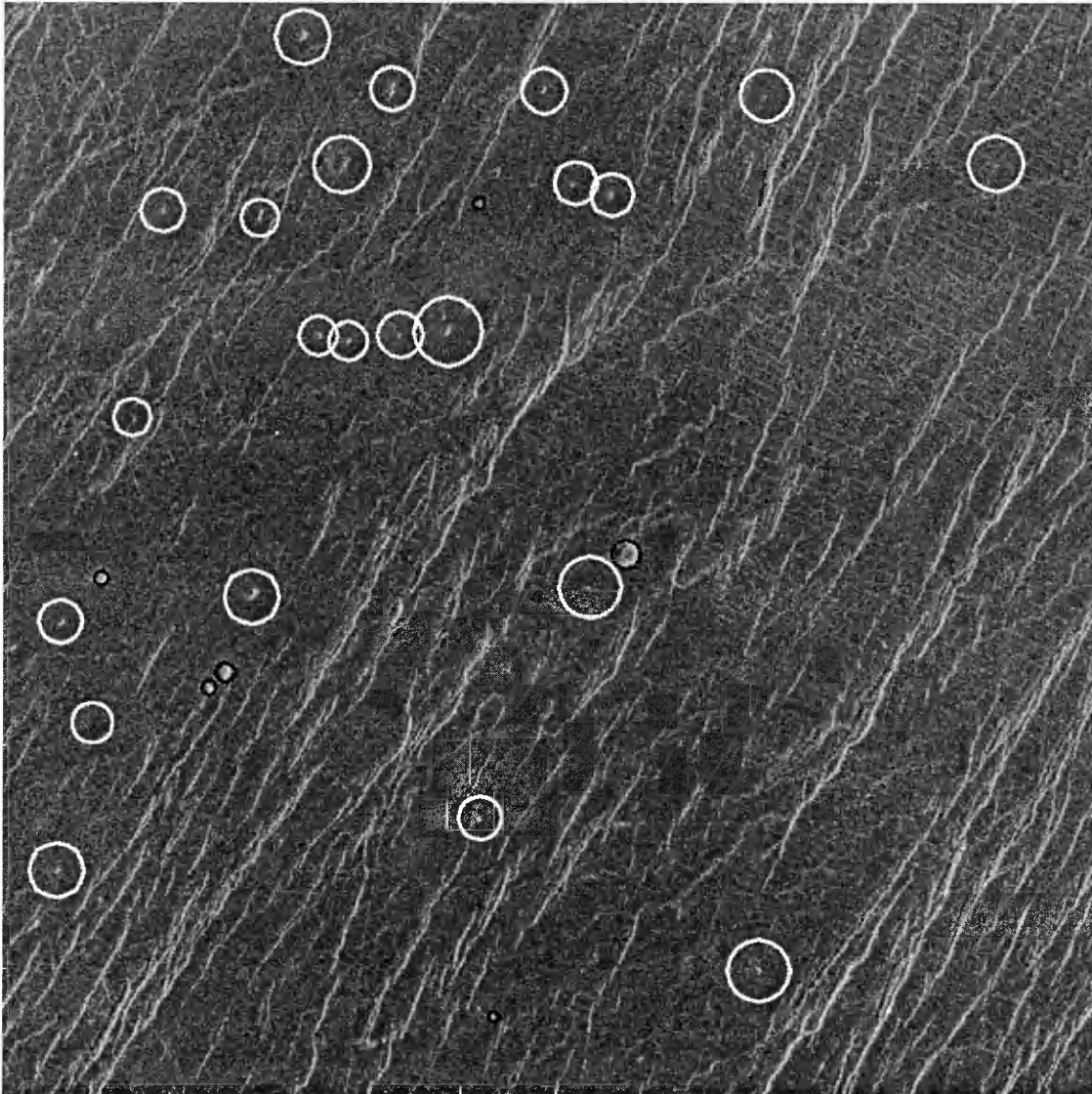


Figure 18: Excerpt from Magellan Venus data set, with volcanoes circled in white and impact craters (negative examples) circled in black

We initially chose a foveal window size of 81×81 pixels (33 inputs) to match the typical size of the targeted volcanoes. However, a baseline representation of this size would

have resulted in far too many inputs for practical experimentation, so we made a tradeoff between size and number of inputs. A 9×9 square window (with 81 inputs, about as large as practicable), although significantly smaller in area than the foveal window, still results in more than twice as many inputs to the learning algorithm. To control for the effect of window size, we also compared the 9×9 baseline representation with a 9×9 foveal representation (17 inputs). By comparing these three input representations, we hoped to explore the tradeoff between window area and number of inputs to the learning algorithm.

Ideally, a classifier would locate each volcano by its center and report the corresponding coordinates. To aid in learning this specific target concept, we provided only positive training examples in which a volcano is centered in the input pixel window. To this end, we began by scaling each target circle from the nominal “ground truth” classification down by a factor of eight. The resulting target regions are only a few pixels in diameter and correspond roughly to just the volcanic craters. We randomly sampled positive training examples from among these regions with equal probability. We randomly sampled negative training examples from the remainder of each image.

As with the other two problem domains, we found that different sampling rates were appropriate for the different learning algorithms. Selecting all of the positive examples and a random one-percent of the remainder of the images (negative examples) worked well for ANNs and trained the networks in a reasonable amount of time. Due to the inherent inefficiency of the KNN algorithm, we were only able to train using 10 percent

of the positives and 0.1 percent of the negatives. Although speed is not an issue for NB classifiers, physical memory constraints limited us to the same sampling rates as we used for the ANNs.

With the four images available to us, we conducted four-fold, leave-one-out cross validation experiments. As a final step, we applied the Merge algorithm with a 20-pixel TBM size to filter out noise in the resulting classification.

5.3.2 Magellan Venus Results

Figure 19 through Figure 22 show our results for locating volcanoes on Venus. In each of the figures, the dark regions indicate the reported positives, and the light background areas are the negative class. The black circles on the images indicate the consensus on where volcanoes are actually located in the images. The light gray circles mark the consensus location of impact craters. We did not specifically sample impact craters as negative examples, but left the classifier to deduce that impact craters are negative based only on the training data.

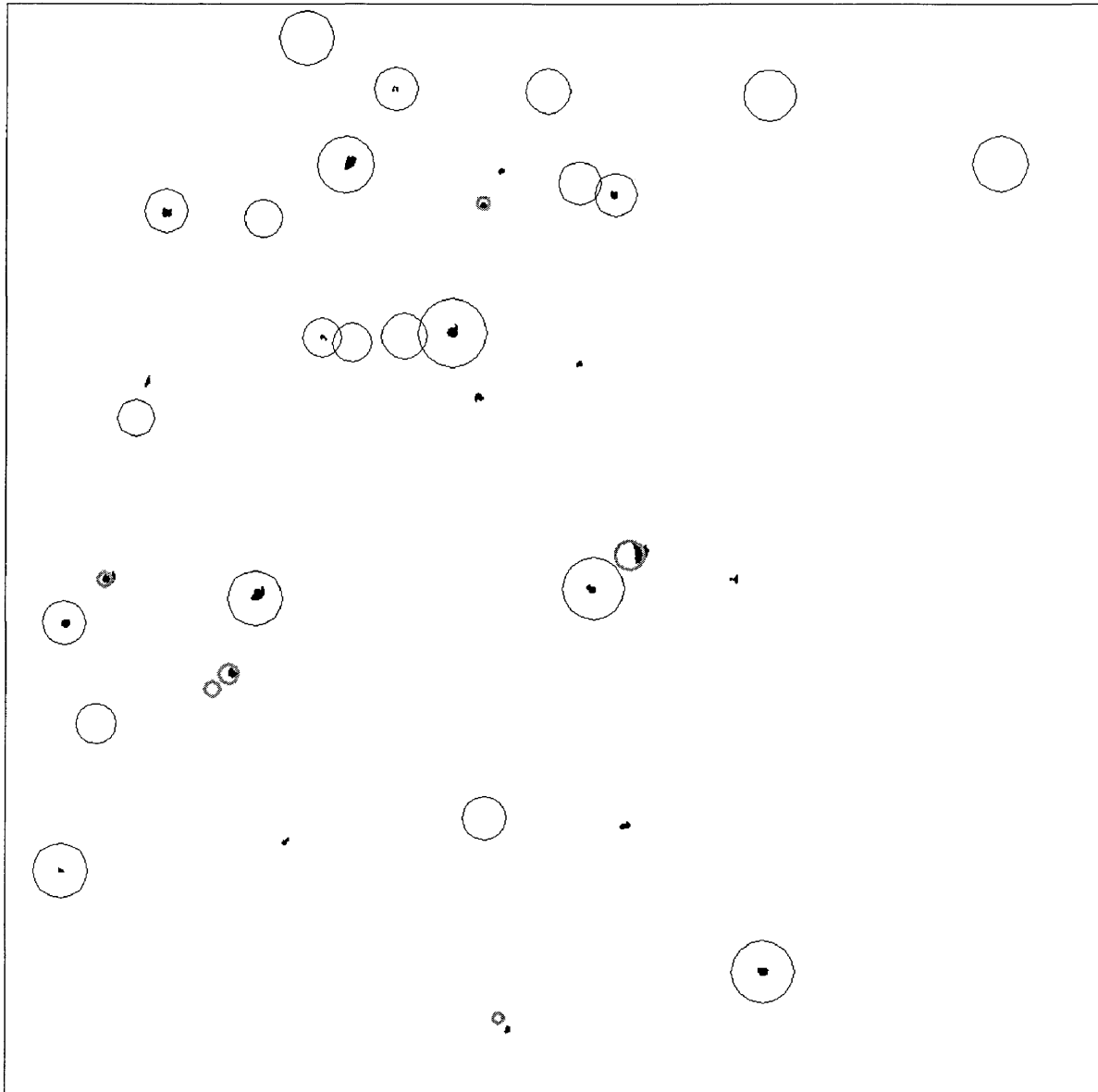


Figure 19: Typical ANN results using 81×81 foveal representation (33 inputs). Black circles mark consensus volcano locations, gray circles mark impact craters.

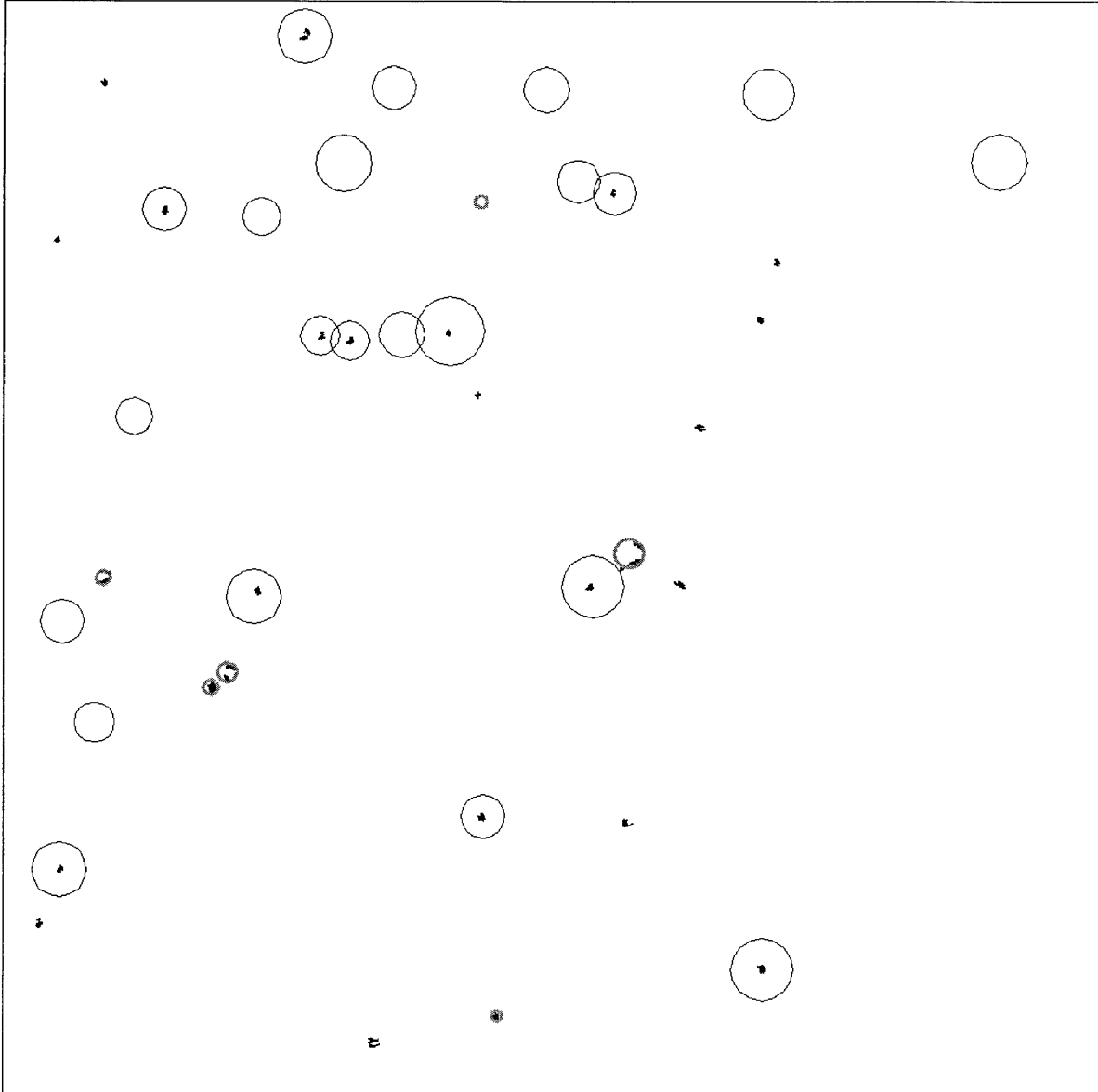


Figure 20: Typical ANN results using 9×9 baseline representation (81 inputs). Black circles mark consensus volcano locations, gray circles mark impact craters.

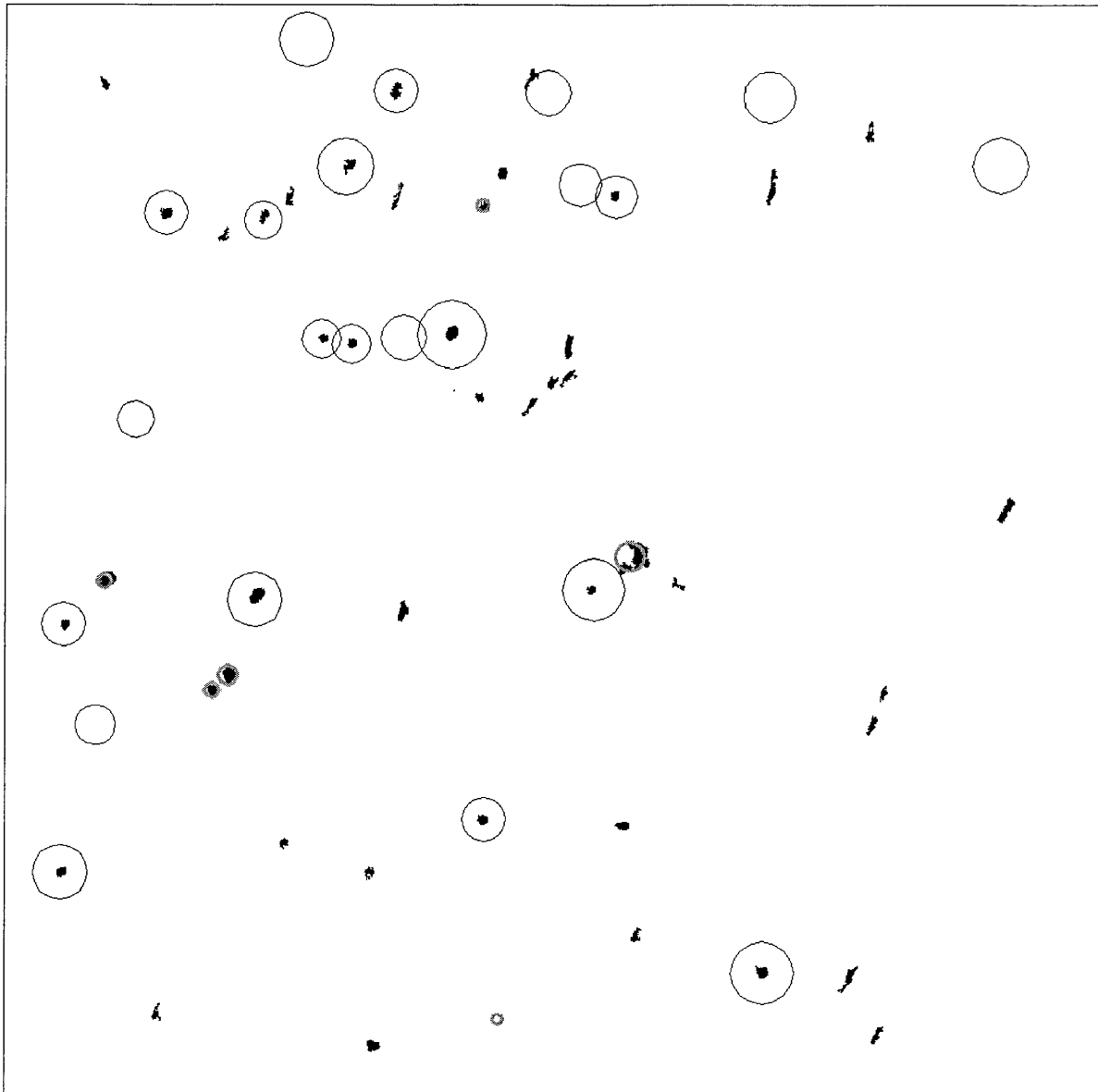


Figure 21: Typical KNN results using 81×81 foveal representation (33 inputs). Black circles mark consensus volcano locations, gray circles mark impact craters.

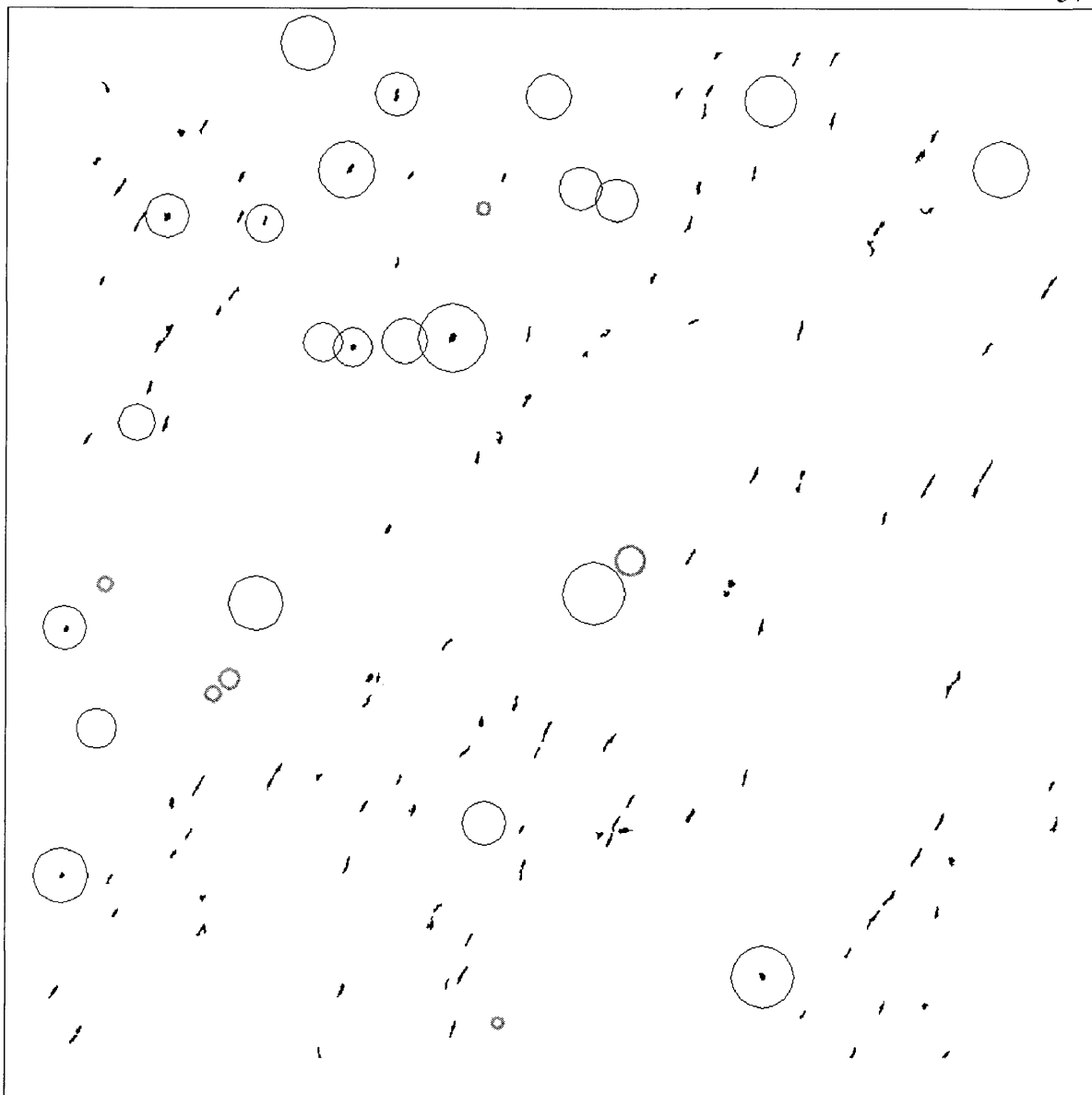


Figure 22: Typical NB results using 81×81 foveal representation (33 inputs). Black circles mark consensus volcano locations, gray circles mark impact craters.

In most cases, the various learning algorithms were unable to discern the difference between volcanoes and impact craters effectively. In fact, the two types of craters are difficult to distinguish in these images by the human eye as well. Typically, a planetary geologist would distinguish between them by the presence or absence of a cinder cone

surrounding the crater. Due primarily to computer memory and processor power constraints, only the larger of the two foveal input patterns in this experiment provided enough spatial context to detect the surrounding cone.

Figure 23 compares the accuracy and effectiveness of the various learning algorithms and input representations. Again, we use ROC curves to illustrate the tradeoff between locating a high proportion of the true targets and keeping the number of false positives to a minimum over a range of costs.

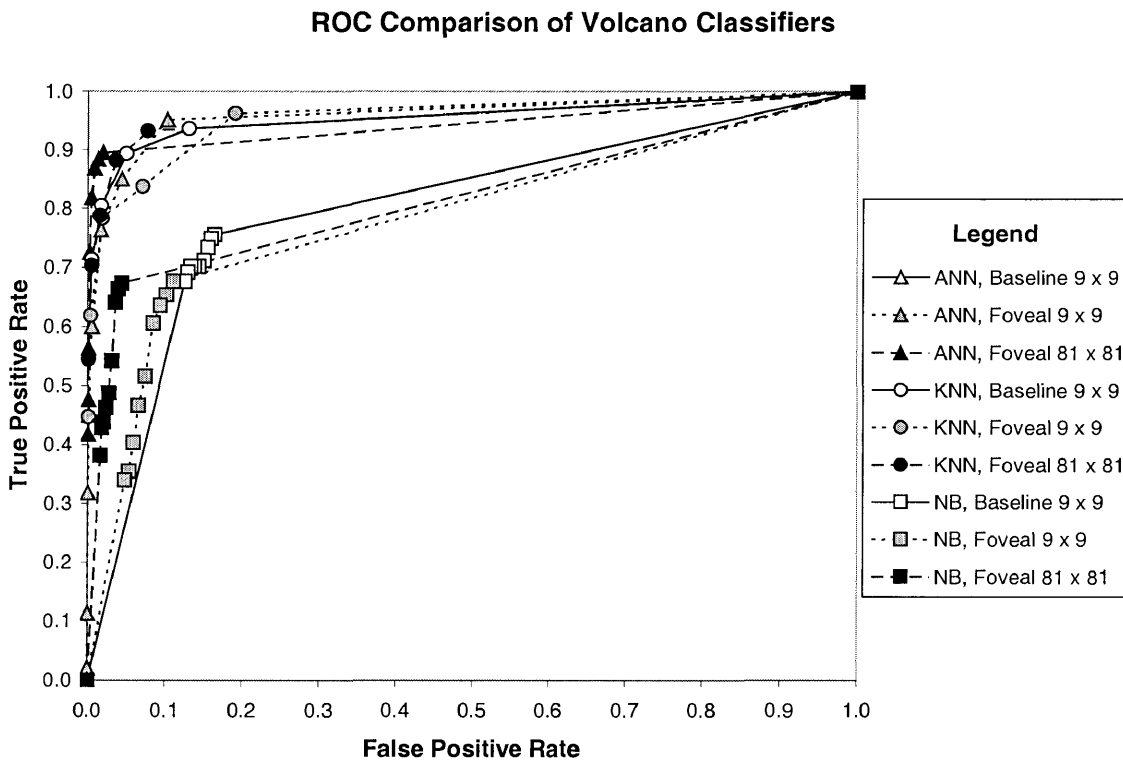


Figure 23: ROC comparison of volcano classifiers by learning algorithm and input representation

As explained above, each ROC curve represents a series of thresholds applied to the output of the learning algorithm in a particular experimental treatment. The better classifiers are ones whose curves push toward the upper left corner of the chart. Again, the cost of misclassification is unknown, so the best way to compare classifiers across a range of costs is to compare their areas under the ROC curve. Figure 24 summarizes the effectiveness of each type of learning algorithm and input representation by comparing the areas under their ROC curves.

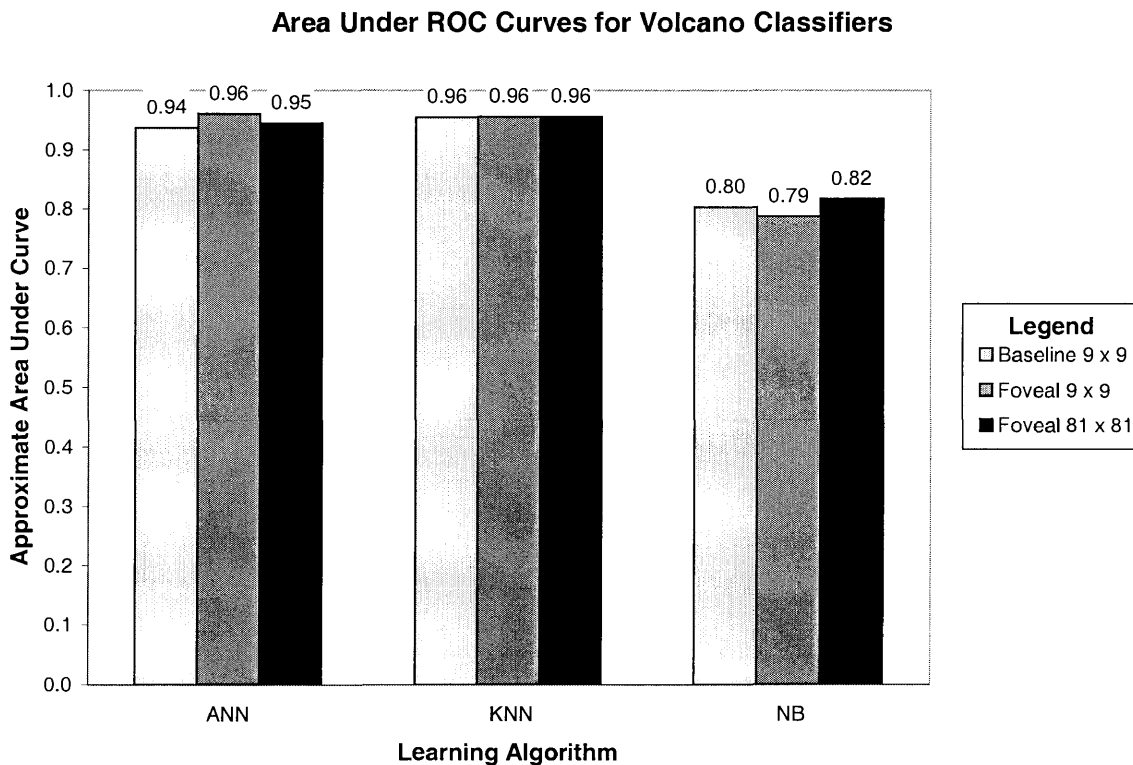


Figure 24: Approximate area under ROC curve by learning algorithm and input representation.

The large size and complexity of the volcano features required a large baseline representation (9×9 pixels, 81 inputs) resulting in very time consuming training and

classification. Even with so many inputs though, the baseline representation was slightly less effective than the foveal. The KNN appeared to perform a little better overall than the ANN. Although the NB was by far the fastest of the three types of learning algorithms, it was able to locate only a few of the volcanoes successfully and returned many spurious non-volcanic features, including impact craters.

6. DISCUSSION & FUTURE WORK

In the absence of ground truth, planetary geologists agree approximately 80% in their expert classification of volcanoes on Venus (Burl et al. 1994). Our automated classification system shows promise of achieving that level of accuracy. When results are based on flawed ground truth classifications – as in the case of the Presidio data and, to a lesser extent, the Helena Valley data – the accuracy and effectiveness metrics appear less impressive than they otherwise would. For example, Opitz et al. found that when learning results are compared against a carefully hand-digitized ground truth classification instead of the noisy version used in this study, accuracy improved by about 10%. This suggests that machine learning may achieve better accuracy in some cases than can humans.

Our results indicate that machine learning can be used to model road extraction and classification of ground cover types effectively. However, in its present form our method is less effective at extracting large, complex objects such as volcanoes. In addition, we demonstrate that ANN and KNN classifiers perform substantially better for each type of problem than do NB classifiers. The use of attribute position independence, better estimates of prior probabilities, etc. may improve NB performance in the future. Finally, our experiments show that foveal input representation performs equal to or slightly better than baseline representation in most cases, while requiring less learning and classification time.

This study demonstrates the possibilities for applying machine learning to automatic image recognition, but we need to refine our approach further. The remainder of this section briefly describes some alternative approaches.

Interactive learning: Due to the barely discernable distinction (even by the human eye) between impact craters and volcanic craters, we expected that learning to automatically differentiate between them will require multiple, interactive learning iterations. Iterative training set refinement would be particularly valuable, because the user generally does not know a priori which examples will prove most useful to the classifier. By interactively augmenting the training set, the user would be able to nudge the classifier toward increasingly accurate concepts.

Hierarchical learning: Recognition tasks that do not yield to the simple representations may succumb to a hierarchical approach. Volcanoes come in a wide range of sizes, and rather than relying on the classifier to learn scale invariance implicitly, we could instead start by scaling the images by various factors. We would use these scaled images to train the classifier, and then use the outputs of these initial passes as inputs to a next layer of learning.

Concept Drift and “Local Mode” Classification: Learning to classify homogeneous images taken from near the training data source is generally more successful than trying to classify heterogeneous images taken from some distance away. This is known in the machine learning literature as concept drift and is one of the motivating influences for

image learning by example. Figure 25 compares overall accuracy obtained from the various learning algorithms when classifying image data located very near to the training examples (“Near”) versus image data sampled from the far side of the image data set (“Far”). The ANN and KNN results show the expected drop in accuracy when the training data is spatially further removed from the new instance to classify. In such cases, the classifier is suited only to use in what Burl et al. refer to as “local mode,” and will need to be retrained for each new region of interest. In future studies, we need to explore further the robustness of various image learning approaches to concept drift.

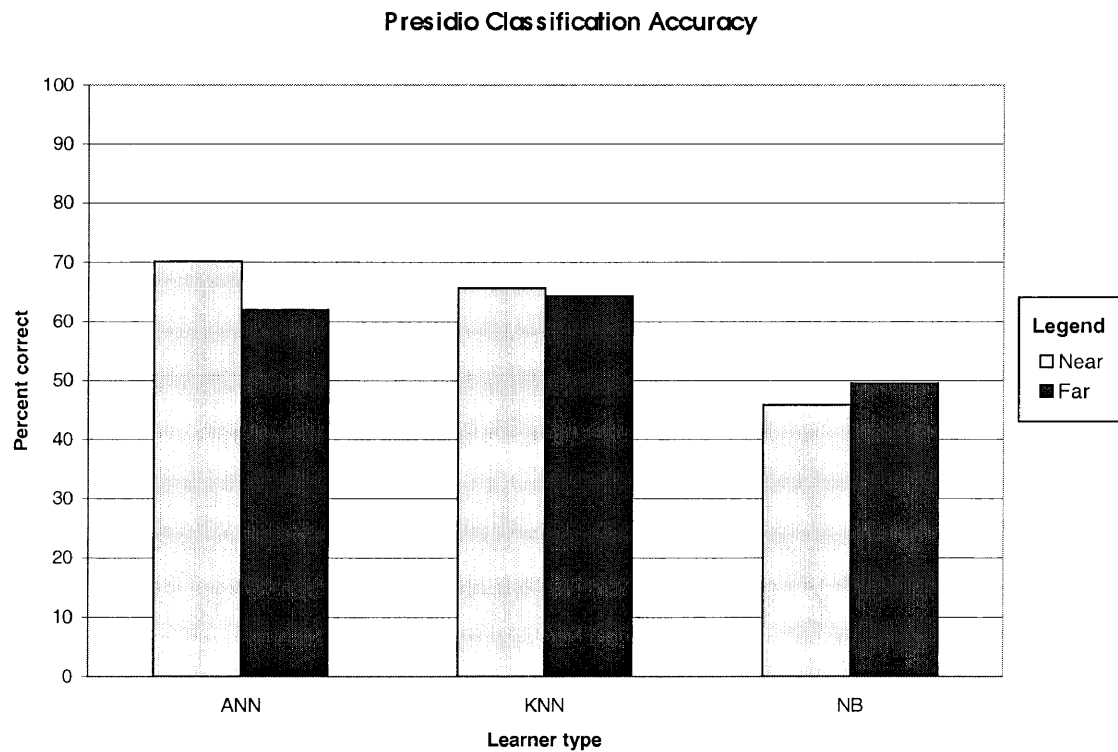


Figure 25: Accuracy of Presidio classification by learning algorithm and training/classification proximity.

N-Way versus Binary Classification: Some classifiers learn a binary concept (e.g., road versus not road, or volcano versus not volcano), while other classification problems involve an N-way classification (e.g., structure versus water versus vegetation). In this latter type of problem – of which the Presidio problem is a good example – the multitude of simultaneous decisions that it must make overwhelm the classifier. Hence, a more effective approach to learning N-way concepts might be to learn each class separately, as a binary concept, and then to combine the results.

Code Optimization: If the VLS software were ever to be adapted for a production environment, some optimization would be appropriate. For example, the KNN classification scheme might benefit from the use of a modified Quick Sort to partition the k nearest neighbors from the more distant ones. The current version of the software employs an inefficient selection sort. In addition, VLS could accommodate larger images via the following:

- A non-graphical, batch classification module
- The new Java Advanced Imaging API (in addition to the Java 2D API that is currently used)
- Optional tradeoffs between speed and memory requirements (instead of always being set for speed)

On-the-Fly Learning: The user interface might benefit, too, by inclusion of interactive features to make training set selection easier and faster. It could incorporate on-the-fly

learning that could predict obvious training examples based on the user's recent actions, allowing the user to concentrate on determination of the more subtle examples.

7. CONCLUSIONS

In this thesis, we looked at the effectiveness of a variety of machine learning techniques in extracting features from various types of digital images.

We found that machine learning can be practical for road extraction and ground cover classification. At this point, it is less effective at locating large, complex objects like volcanoes. We showed that ANN and KNN classifiers achieve higher accuracy for each type of problem than do NB classifiers, and that ANNs are computationally more efficient than KNNs. Finally, our experiments show that foveal input representation is about as effective as baseline representation, while requiring less learning and classification time.

In summary, this study takes an important step forward by demonstrating the promise of machine learning for image recognition.

8. REFERENCES

- Burl, M. et al. (1998). Learning to recognize volcanoes on Venus. In *Machine Learning Journal Special Issue on Applications and Knowledge Discovery*, 30:165-194.
- Burl, M.C., Fayyad, U., Smyth, P., Perona, P. & Burl, M.P. (1994). Automating the hunt for volcanoes on Venus. In *Proceedings of the 1994 Conference on Computer Vision and Pattern Recognition*.
- Domingos, P. & Pazzani, M. (1996). Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. In *Proceedings of the Thirteenth International Conference on Machine Learning*.
- Duda, R., & Hart, P. (1973). *Pattern classification and scene analysis*. New York: John Wiley & Sons.
- Ford, R., Ma, Z., Barsness, S. & Redmond, R. (1997). Rule-Based Aggregation for Region Identification. In *Proceedings of the 1997 American Society for Photogrammetry and Remote Sensing Annual Conference*, Seattle, WA.
- Joachims, T. (1996). *A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization*, (Computer Science Technical Report CMU-CS-96-118). Carnegie Mellon University.
- Maloof, M. et al. (1998). Learning to detect rooftops in aerial images. In *Proceedings of the Image Understanding Workshop*, (pp. 835-845).
- Mitchell, T. (1996). *Machine Learning*. MIT Press, Boston, MA.
- Quinlan, J. (1986). Induction of Decision Trees. In *Machine Learning*, 1:81-106.
- Roshceisen, M., Hofmann, R. & Tresp, V. (1991). Neural control for rolling mills: Incorporating domain theories to overcome data deficiency. In Moody, J., Hanson, S. & Lippmann, R., editors, *Advances in Neural Information Processing Systems (volume 4)*, (pp. 659-666). Morgan Kaufmann, San Mateo, CA.
- Rumelhart, D., Durbin, D., Golden, R. & Chauvin, Y. (1995). Backpropagation: The basic theory. In *Backpropagation: Theory, Architectures, and Applications*, (pp. 135-143).

Rumelhart, D., Hinton, G. & Williams, R. (1986). Learning internal representations by error propagation. In *Parallel Distributed Processing: Exploration in the microstructure of cognition*, Volume 1: Foundations, MIT Press, Cambridge, MA, (pp. 135-143).

Smyth, P., Fayyad, U., Burl, M., Perona, P. & Baldi, P. (1995). Inferring Ground Truth from Subjective Labeling of Venus Images. In *Advances in Neural Information Processing Systems 7*, MIT Press, Cambridge, MA.

Swets, J. (1988). Measuring the accuracy of diagnostic terms. *Science* 240:1285-1293.

APPENDIX A: SOFTWARE DEVELOPMENT

The sheer volume of data involved with digital images suggested that a specialized graphical user interface (GUI) would be helpful in setting up the learning problems. It does not suffice to define a particular combination of pixel values as positive (e.g., a volcano) or a negative (e.g., an impact crater) numerically. The user needs to be able to specify with the click and drag of a mouse the approximate extent of the geographical feature of interest. By selecting entire regions of an image at once, one can generate a multitude of training examples with minimal time and effort. In addition, we perceived a need for a suite of image transformation utilities for the purposes of pre-processing the images before learning, and post-processing the output of the classifiers.

To complicate matters, many available satellite images data sets are multi-spectral – that is, they include information from parts of the radiation spectrum that are invisible to the human eye. In addition to the red/green/blue (RGB) or hue/saturation/brightness (HSB) color components typically used in digital imaging, some satellite images include infrared, ultraviolet and even ancillary data such as digital elevation values. Typical graphics software does not provide adequate support for visualizing and manipulating image data of such pixel depth. In addition, highly optimized code could rapidly generate the millions of input patterns and provide them to the learning algorithms. For all the above reasons, we created a software package known as Visual Learning System (VLS).

VLS allows non-technical computer users to view and manipulate image data of arbitrary pixel depth; more importantly, it allows point-and-click selection of targeted features and assignment of them to arbitrary feature type classes. A user may select and each machine learning algorithm from a drop-down menu and configure it using standard graphical controls.

From the user standpoint, a typical image learning session consists of:

- Drawing shapes on a few images (or portions thereof) to indicate examples of the target concept
- Specifying learning parameters
- Initiating training
- Using the trained classifier to classify other images

If the classifier does not adequately classify the new instances, then the user may provide more examples and re-train the classifier. When the classifier achieves adequate classification accuracy, the user can save its state for later classification of entire sets of images. In this way, one can classify large volumes of image data without further user intervention, presumably much more quickly than could be done by digitizing.

Figure A1 illustrates the fundamental class structure of VLS. The VLS object represents the application window and is the executable object in the package. The VLS object contains a set of DocumentPane objects, each displayable in an internal frame. The name derives from the fact that each contains an image document, and each is a lightweight

graphical component. The DocumentPane data structure consists of a series of image data rasters, one byte per pixel, called bands. Each pixel in an image document consists of a vector of reflectance values, one for each band, and can be arbitrarily deep.

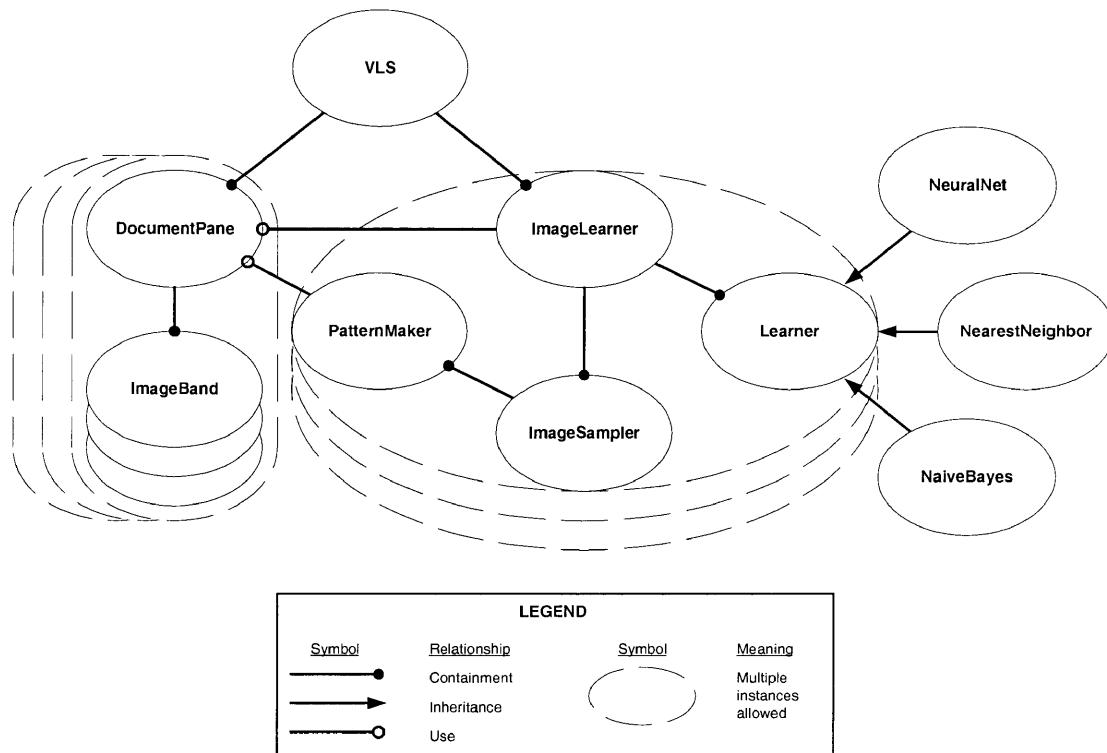


Figure A1: Visual Learning System class structure

VLS can accommodate many ImageLearner objects at a time. Each one encapsulates a learning algorithm (a Learner object) and an input representation (a PatternMaker object) along with methods and objects (e.g. an ImageSampler object) for training on and classifying sets of image documents.

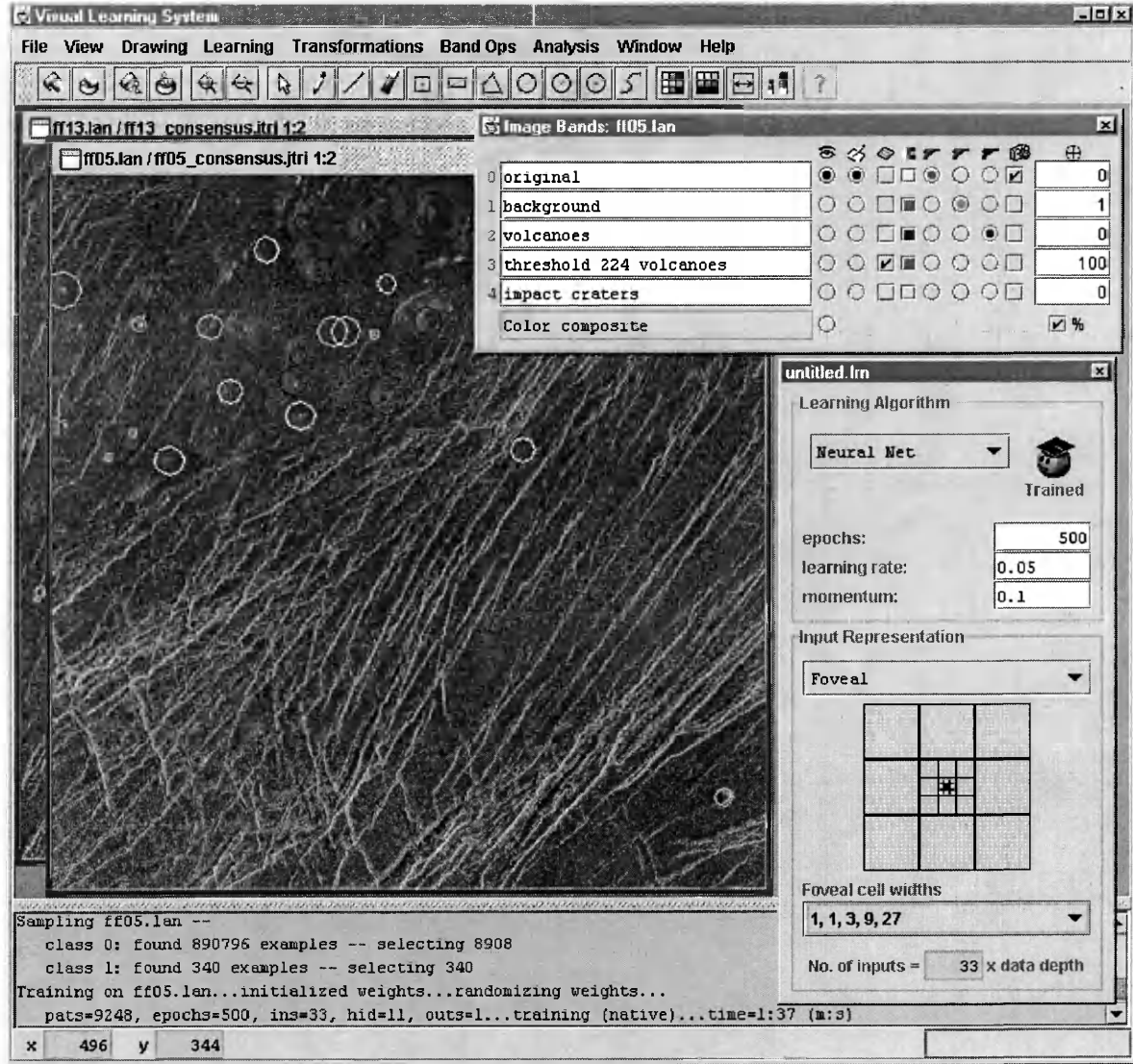


Figure A2: Visual Learning System graphical user interface

Figure A2 shows an example of the VLS graphical user interface during a typical interactive training session. Features include a multiple internal frame layout, textual status pane, progress bar, floating tool bar, image band visualization panel, and image learner panel, each an extension of the Java Swing component library. VLS graphically

represents both of the basic objects involved in image learning – the image document and the learner (or classifier) –for easy manipulation.

VLS includes tools to support the hierarchical nature of image data. Images typically comprise rasters (two-dimensional arrays of data), often include multiple bands (a third dimension), and sometimes contain a color palette for visual interpretation. Images may of course be subsets of other images, cropped from a larger version or otherwise spatially related. Even when the image data of interest is provided in one large raster (as in the case of the Presidio) it is often more convenient or even necessary to tile the image into smaller, more manageable rasters. In such cases, one usually manipulates images in sets – perhaps one small set for training, another for testing classification accuracy, and the remainder for bulk classification. For the above reasons, VLS implements a multiple-document interface (MDI) and images of arbitrary pixel depth in the popular ERDAS/LAN image format (widely used for GIS applications). Wherever possible, the image processing and learning tools are configurable to operate uniformly on multiple bands within multiple images at once.

To complicate matters, images are not the only type of document needed for image learning. The state of each instance of a learning algorithm, configured and trained for a particular task, may be stored for later retrieval, either for classifying more images or for further training. Since the typical user only deals with one classifier at a time, the graphical representation of the classifier does not appear in the same internal frame layout as is used for the images. Instead, a separate, modeless dialog panel represents the

state of the classifier and its corresponding input pattern. Here, the user can directly manipulate the learning parameters and see the input pattern just as the classifier sees it, with a dynamic grid of boxes representing the sliding pixel window on the image.

VLS requires not only a complex GUI, but also efficient implementation of some computationally intensive machine learning algorithms. While Java is known for its strengths in rapid GUI prototyping, it is not particularly fast for extensive computation. When Java prototype implementations of the machine learning algorithms ended up running too slowly, it became necessary for us to use the C programming language to handle the image learning and other image processing tasks. In the end, all GUI code remained in Java, while we ported the computationally intensive modules to ANSI C, using the Java Native Interface (JNI) bridging the gap. Since both Java and C are available on most PC and workstation platforms, VLS sacrificed very little portability to realize the benefits of both programming languages.

APPENDIX B: LEARNING ALGORITHMS

Artificial neural network

Each artificial neural network (ANN) used in this study employed standard backpropagation (Mitchell 1996) and the logistic (a.k.a. sigmoid) function to squash the output of each hidden and output node:

$$\sigma(net) = \frac{1}{1 + e^{-net}} \quad (\text{Eq. 1})$$

K-Nearest Neighbor

The k-nearest neighbor (KNN) classifiers used a Euclidean distance metric and weighted the contribution of each of the k nearest targets in inverse proportion to the distance:

$$w_i \equiv \frac{1}{d(x_q, x_i)^2} \quad (\text{Eq. 2})$$

Naïve Bayes

A Bayes classifier returns the maximum a posteriori (MAP) classification given the attributes that describe the instance:

$$v_{MAP} = \arg \max_{v_j \in V} P(v_j | a_1, a_2 \dots a_n) \quad (\text{Eq. 3})$$

We can express this in terms that are more useful by applying Bayes Theorem:

$$v_{MAP} = \arg \max_{v_j \in V} \frac{P(a_1, a_2 \dots a_n | v_j) P(v_j)}{P(a_1, a_2 \dots a_n)} \quad (\text{Eq. 4})$$

Because the prior probability of the attribute vector is independent of the classification, the term in the denominator is superfluous:

$$v_{MAP} = \arg \max_{v_j \in V} P(a_1, a_2 \dots a_n | v_j) P(v_j) \quad (\text{Eq. 5})$$

The prior probability of each class, $P(v_j)$, is easily estimated based on the occurrence of each class in the training set. However, accurately estimating the conditional probabilities in Equation 5 typically would require much larger training sets than are practicable. Hence, in practice, it is often more useful to naïvely assume that the attributes are conditionally independent. This assumption gives us the Naïve Bayes classifier:

$$v_{NB} = \arg \max_{v_j \in V} P(v_j) \prod_i P(a_i | v_j) \quad (\text{Eq. 6})$$

The Naïve Bayes classifier is often accurate even in cases where the attributes are not actually conditionally independent.