

APPLYING DOMAIN KNOWLEDGE TO THE RECOGNITION OF
HANDWRITTEN ZIP CODES

Ibrahim Chaaban

Submitted to the faculty of the University Graduate School
in partial fulfillment of the requirements
for the degree
Master of Sciences
in the Department of Computer and Information Sciences,
Indiana University South Bend

May 2007

ACKNOWLEDGEMENTS

A special thanks to Dr. Michael R. Scheessele for his time and valuable advice. I would also like to thank Dr. James Wolfer and Dr. Yi Cheng, for their time and useful comments.

TABLE OF CONTENTS

INTRODUCTION	1
EXPERIMENT 1	7
Method	
Subjects	7
Stimuli	8
Procedure	8
Results and Discussion	9
EXPERIMENT 2	9
Method	
Subjects	9
Stimuli	9
Procedure	10
Results and Discussion	10
MODEL	11
Segmentation	11
Separating connected digits in a ZIP.....	13
Joining segments of a digit	14
Feature Extraction	15
Recognition	16
Classification Method	17
Classifier structure	18
Training data	19
Testing data	20
Results and Discussion	20
CONCLUSION	23
REFERENCES	24
FIGURES	26
TABLES	40
APPENDIX	49

INTRODUCTION

The recognition of handwritten digits is very challenging and it has been the subject of much attention in the field of handwriting recognition. Recognizing digits is a problem that at first seems simple, but it is a very difficult task to program a computer to do it. The difficulties and complexity of this task lie in the fact that a computer program must be able to recognize handwritten digits produced by different people, using different instruments. The system has to deal with widely different sizes and slants, with different shapes and widths of the strokes. Handwritten digit recognition has been an especially active topic in the field of pattern classification and learning. Many approaches and methods have been proposed for pre-processing, feature extraction, classification and/or learning of handwritten digit images. Researchers have also compiled widely used standard image databases to evaluate the performance of these approaches and methods. The four most important and widely used databases are CENPARMI, CEDAR, MNIST, and the United States Postal Service (USPS) database.

The United States National Institute of Science and Technology (NIST) constructed their first database from NIST's Special Database 3 (SD-3) and Special Database 1 (SD-1). The training set of the modified database, MNIST, has 60,000 images of handwritten digits from approximately 250 writers and a testing set of 10,000 [17]. The Centre for Pattern Recognition and Machine Intelligence (CENPARMI) digit database contains 6,000 digit images collected from the envelope images obtained by the USPS. The digits are scanned in 166 DPI [16]. The CEDAR ZIP codes were scanned in 300 DPI from mail images obtained by USPS [3]. The USPS database contains 9298 handwritten digits (7291 for training, 2007 for testing) [18].

Handwritten digit recognition research concentrates on either individual digits or digit strings. With respect to machine recognition of individual handwritten digits, researchers have achieved an accuracy of 99.58% [8]. Such performance compares favorably to human performance. In fact, two experiments have been conducted to evaluate the human error rate on the USPS test data set. The first experiment reported a 2.5% error rate [2]¹, while the second experiment reported an error rate of 1.51% [4]. The first experiment was published as a proprietary report and is not readily available for public consumption. The second experiment suffered from two major methodological flaws. The experiment was conducted using four subjects and each was given 2007 test patterns which were printed on white paper. Each page had approximately 120 images of handwritten digits separated by white space. Each subject was asked to identify each pattern and then to clearly label the pattern on the paper. The results were “carefully” entered into an Excel file manually [4]. The first flaw of the experiment is what I call association. When a subject is looking at a sheet of paper that has 120 images on it, it is possible that when a subject encounters a difficult image to identify, this subject might associate this particular image with other images on the page in deciding on a response. The second flaw comes from the fact that the subjects are writing their responses on the papers they were given. How do we know that the person entering the results into Excel is reading the results (which are handwritten digits) correctly? It seems that the problem is being regenerated by the subjects, and now the person entering the results into Excel must solve the problem. Due to these two flaws in the design of the second experiment and the lack of availability of the report for the first experiment, I have run my own

¹ My advisor and I tried our best to locate a copy of this proprietary report, but we were unsuccessful. We contacted the authors, but neither were able to produce a copy of the report.

experiment to determine human performance in recognition of individual handwritten digits.

Many domains require recognition of digit strings, as opposed to individual digits. Some examples are automated sorting of mail by postal code [1], automated reading of checks [19] and tax returns, and data entry for hand-held computers. In these domains, handwritten digits rarely appear isolated. Instead they appear as part of a string of digits where some digits may touch and/or overlap. In many of these real world applications, the images are processed by human operators. However, automation may improve production and cut costs. For this to happen, performance of an automated system must compare favorably to human performance. Such comparison is an essential component in determining whether the problem has been solved or not. An automated system cannot be fully integrated into real world applications until the performance gap between humans and machines is sufficiently minimal. Morita and colleagues [11] developed a system that uses the Hidden Markov Model (HMM) and the Multilayer Perceptron (MLP) to segment and recognize unconstrained handwritten dates on Brazilian bank checks. The system processes the three subfields that make up the date (day, month, and year). In order to reduce the date lexicon size, Morita and colleagues used domain knowledge, which enabled them to reduce the complexity of the recognition process. This was possible because the lexicons for day and year are known. For example, in a two digit day the first digit can only be 0, 1, 2, or 3, while the second digit can range from 0 to 9. A similar approach was applied to two/four digit years. They restricted checks to just those written after 1990 and before 2029. In a two digit year the first digit can then be 0, 1, 2, or 9. The second digit can range from 0 to 9. In a four digit year, the first digit can only

be 1 or 2; similarly the second digit can only be 0 or 9. They used five MLP neural networks each using only one hidden layer to classify the different metaclasses of digits. For example, one MLP network classified the digits 0, 1, 2 and 3. It had one hidden layer with 70 hidden units. Another network classified the digits 0 through 9 using one hidden layer with 80 hidden units, and so forth. The hidden layer in each network contained an empirically determined number of hidden units, each of which connected to all input and output units. The performance of this system on four digit years was 100%. On two digit days performance was 93.2%, and on two digit years performance was 97.2%. The discrepancy between performance on two digit years and two digit days can also be explained by their use of domain knowledge. Specifically, they exploited the fact that year always appears at the end of a date. They used this knowledge to improve segmentation of the year from the rest of the date, and the improved segmentation led to improved recognition.

ZIP code recognition is another very interesting problem, due to the benefits of having an accurate automated system that can sort letters at a high rate. On average a postal worker can sort about 800 letters an hour. On the other hand, an automated sorting machine, reading printed ZIP codes with an optical scanner is estimated to process about 37 times more than the postal worker at a fraction of the cost [10]. Such performance is also desired for handwritten ZIP codes. Liu and colleagues [8] compared different classifiers and learning methods in the recognition of handwritten ZIP codes. The classifiers compared were single-layer perceptron (SLP), multi-layer perceptron (MLP), radial basis function classifier (RBF), polynomial classifier (PC), learning vector quantization (LVQ), modified quadratic discriminant function (MQDF), and learning

quadratic discriminant function (LQDF). Each classifier had two or three variations depending on the learning method, such as maximum likelihood estimation (MLE), discriminative learning (DL), or enhanced discriminative learning (EDL). The method of maximum likelihood is a general method of estimating parameters of a population by values that maximize the likelihood of a sample [9]. The discriminative learning method on the other hand, updates parameters iteratively to separate the patterns of different classes. The enhanced version of discriminative learning is equivalent to DL, except that in EDL the training is done with outliers [8]. The first classifier they tested was the SLP. A single layer perceptron has an input and output layer. Each neuron in the output layer of their network was connected to each input neuron. When trained with the EDL method and forced to make a decision without rejection, this network's correct recognition rate was 74.31%. This is probably because this type of network is limited to only a single layer. They also tested a MLP network. With one or two hidden layers, this network can approximate virtually any input-output map, by learning to transform input data into a desired response. The MLP's correct recognition was 89.22% without rejection using the same learning method. When using the EDL method, both the RBF and PC produced similar results to that of the MLP. The RBF correct rate without rejection was 87.84%, and the PC had a correct rate of 89.91% without rejection. The LVQ classifier was also tested. LVQ is a competitive learning algorithm, described sometimes as the supervised version of Kohonen's Self-Organizing Map [7]. Percent correct for the LVQ classifier was 87.61% with no rejections. The MQDF classifier described by Liu et al. [8] as the MLE version of LQDF was also tested, and it had a correct rate of 87.61% without

rejection. Finally, the LQDF classifier was tested to reveal a 90.37% correct rate without rejection.

In order to test these classification methods, Liu and colleagues [8] developed a new model. First, the model pre-processed the string image to prepare it for pre-segmentation. In the pre-segmentation stage, connected component labeling was applied. To handle the cases of touching digits, the model analyzed the upper and lower profile curves of any touching digits in order to generate a candidate cut. Heuristic rules were also applied to ensure that the candidate cut would not split a single digit. Their pre-segmentation stage was followed by an integrated segmentation and recognition (ISR) stage. In this stage Liu et al. [8] combined dynamic programming (DP) search and digit recognition. Each of the character classifiers described earlier was used to assign class scores to the candidate patterns generated in the previous stage. The optimal pattern was then found by DP search based on class scores given by the classifier. The ISR stage, as claimed by Liu and colleagues [8], is the most essential part of their system due to the variability in size, interval, and breaking/touching of digits in a ZIP code. Therefore, according to them, the digits in a ZIP code cannot be reliably segmented in a distinct stage prior to recognition. This implies that the problem can only be solved by integrated segmentation and recognition. The recognition portion of this integrated segmentation and recognition process tested the different types of classification methods described above. The LQDF classifier was chosen because it had the best performance. The system was tested on 436 5-digit ZIP code images from CEDAR CDROM-1. The ZIP code images were obtained by USPS from actual mail images. A number of these ZIP codes have digits which touch, making recognition more challenging. They reported a correct

rate of 90.37%. Unlike the system of Morita et al. [11] they did not use prior domain knowledge. So, incorporating prior domain knowledge into the system of Liu et al. [8] might have yielded an even better recognition rate. Still, their system has achieved the best performance to date. It is unknown how this system compares to human performance because no human recognition rate on handwritten ZIP codes has been established. Therefore, I have performed an experiment to establish human performance in recognizing handwritten ZIP codes. In the next two sections I describe the two human experiments conducted. In the section after that, I describe my model and its performance on the CEDAR CDROM-1 test database. In the final section I draw conclusions and make suggestions for future work.

EXPERIMENT 1

There have been two experiments conducted on the United States Postal Service (USPS) database to establish human performance on recognizing individual handwritten digits. Bromley and Sackinger [2] reported a 2.5% error rate, and Dong, Xiong, and Suen [4] reported a 1.51% error rate. However the former experiment is not in the public domain, and as mentioned, the latter experiment had two major flaws. The present experiment was performed to clearly establish human performance on recognition of individual handwritten digits.

Method

Subjects

Four undergraduate IUSB students participated in this experiment². Subjects were at least 18 years old and had normal (20/20) or corrected-to-normal vision in both eyes. For completing the experiment each subject was paid \$30.00.

Stimuli

As stimuli, I used the digits in the USPS database. This database was also used in the studies of Bromley and Sackinger [2] and Dong et al. [4]. The USPS database was originally collected by CEDAR. Then it was modified by LeCun's research group [18]. The binary patterns were transformed into a 16×16 pixel box that kept the same aspect ratio and centered the patterns. The resulting patterns were gray-level and scaled and translated to fall within the range from -1 to 1. When translating the vector into a 16×16 image I found that the digits were in white and the background in black. So I reversed the colors to match what would normally appear on an envelope (black text on white paper). The database contains 9298 handwritten digits 7291 for training and 2007 for testing. The test set of 2007 was used here (as in the previous studies). The files containing the vectors of these images can be found at [18].

Procedure

Each subject attended 3 sessions of approximately one hour each. In the first two sessions there were 700 trials, and in the third session there were 607 trials, for a total of 2007 trials. In each session the subject sat in front of a computer screen and used software especially created for this experiment (Figure 1). A subject's task was to identify a series of handwritten digits randomly presented on the computer screen, using the mouse to

² All human experiments reported here were first approved by the IUSB Institutional Review Board.

respond. Subjects were asked to respond even if a stimulus was ambiguous. At the end of each session the software calculated the percent correct.

Results and Discussion

As reported in the experiment of Dong et al. [4] there are four labeling errors in the USPS database (Table 1). Without taking those labeling errors into consideration, the average percent error was 2.57% (Table 2). After removing the four incorrectly labeled digits, the average percent error rate was 2.37% (Table 3). This is comparable to the error rate of 2.5% found by Bromley and Sackinger [2] and higher than the error rate of 1.51% reported by Dong et al. [4]. Since these two reports presented considerably different error rates, my experiment helps decide which is correct. The results of my experiment confirmed two things: Bromley and Sackinger's error rate is very close to the actual error rate and the error rate of Dong et al. is inaccurate (as expected) due to the methodological flaws in their experiment.

EXPERIMENT 2

In Experiment 1, I measured human performance in recognition of individual handwritten digits. For this experiment, I will establish human performance in recognition of handwritten ZIP codes. I expect that my results will serve as a benchmark for machine performance in recognition of handwritten ZIP codes.

Method

Subjects

Four undergraduate IUSB students participated in this experiment. Subjects were at least 18 years old and had normal (20/20) or corrected-to-normal vision in both eyes. For completing the experiment each subject was paid \$10.00.

Stimuli

For this experiment I used the 436 5-digit ZIP code images available from the CEDAR CDROM-1 in the testing folder under the BINZIPS directory. These ZIP codes were used in Liu's study to test various classification methods [8]. The ZIP codes were segmented from mail images obtained by the USPS, and they are in binary format. In this database, some ZIP codes have digits that are touching and/or overlapping. This makes the task of recognition more challenging. Figure 2 shows examples of handwritten ZIP codes used in this experiment.

Procedure

Each subject attended one session in which he/she sat in front of a computer screen and used software especially created for this experiment (Figure 3). A subject's task was to identify a series of handwritten ZIP codes randomly presented on the computer screen, by using a mouse/keyboard. For an ambiguous stimulus, a subject was instructed to do their best, but to still respond. At the end of the session the software calculated percent correct.

Results and Discussion

The average percent error of the four subjects was 3.44% (Table 4). Due to the ease of the task, the legibility of most ZIP codes, and the performance of subjects 1, 2, and 4, the result of subject 3 appears to be an outlier. Subject number 3 apparently had some lapses of attention because some of his/her errors were on easy ZIP codes. The average percent error with this outlier removed was 1.61% (Table 5). Until now, no human recognition rate on handwritten ZIP codes had been established. Therefore, this experiment is significant because it establishes human performance in recognizing handwritten ZIP codes. The results of this experiment will allow researchers to compare their models to the ultimate ZIP code recognition system – that of the human visual system. However,

due to the small number of subjects, one must be cautious about over generalizing the results.

MODEL

A unique feature of my model is that it incorporates domain knowledge. For instance, the structure of ZIP codes and the state of destination could be exploited to make the segmentation and recognition processes more accurate. Understanding the structure of ZIP codes can significantly reduce the range of possible classes to consider during the recognition process, thereby increasing accuracy. For example, in a five digit ZIP code, the first digit indicates one of ten large geographic areas in the country. It represents a certain group of U.S. states, ranging from zero in the Northeast to nine in the far West. The second and third digits indicate metropolitan areas and sectional centers. The fourth and fifth digits represent more specific areas such as local post offices or postal zones in larger cities [20]. Combining this knowledge with the state of destination radically reduces the number of ZIP codes to consider during classification.

My model consists of two stages: a segmentation stage and a recognition stage. In the segmentation stage the ZIP code patterns are segmented by applying a few simple techniques and by exploiting the fact that a ZIP code is composed of 5 digits. Utilizing the destination state, the recognition stage will try to recognize the five segments derived from the segmentation stage, as shown in Figure 4.

Segmentation

The main challenge of classifying handwritten ZIP codes is the fact that in real applications the image extracted from a piece of mail will not necessarily appear as five separated digits. This is due to imperfect handwriting, as shown in Figure 5. In addition,

extra noise, (such as bar codes, stamps and other markings made by the post office) is added to the image during processing. Further, moisture and handling may smear or smudge the handwriting on an envelope. To overcome these challenges and to achieve segmentation, a ZIP code pattern goes through three phases in my model (see Figure 4).

In the first phase, the ZIP code pattern is prepared for division into five separate patterns, each representing a digit in the ZIP code. First the pattern is converted into a binary image with a 0/1 representation (0 for white pixels and 1 for black). This allows for easy processing of patterns. Then, from the pattern a noise reduction algorithm locates and removes any set of connected pixels with an area-size less than an empirically determined threshold (3×3). This step is necessary because it helps to remove some of the added noise discussed earlier. Next, another algorithm is applied to the pattern to “close” any open gaps found in digits (see Figure 6) [6]. This step will enhance the shape of each digit and therefore allow for better classification later in the process [14].

The second phase of segmentation finds all connected components in a given ZIP code pattern. A connected component is a set of pixels sharing some feature where each pixel in the set neighbors at least one other pixel in the set. The purpose of this step is to isolate the digits which make up the ZIP code pattern. Two variations of the connected components algorithm were tested. The first algorithm checks four neighboring pixels to determine connectivity to other pixels, while the second algorithm checks all eight neighboring pixels, as shown in Figure 7 [15]. Since there was no distinguishable difference in the outcome of the two algorithms, I chose to use the algorithm which checks four neighboring pixels, because of execution time efficiency.

Because a ZIP code is composed of 5 digits, the second phase can have three possible outcomes. The first outcome would be to get five connected components with a relatively similar size. This implies that the pattern was successfully segmented into five digits. The second outcome would be to get less than five connected components. This could mean that two or more digits in the ZIP code string are touching or overlapping. If so, they must be separated. The third outcome would be to get more than five connected components. This particular outcome suggests that there may be one or more individual digits that are broken into multiple pieces and therefore must be joined together.

The third phase of segmentation involves separating connected digits (if necessary) or joining multiple segments of a digit (if necessary) found in a ZIP code string.

Separating connected digits in a ZIP. When two or more digits are touching or overlapping the recognition task becomes particularly challenging. An improper segmentation tends to leave some of the newly segmented digits with noise or parts of the neighboring digit(s). In the separation process a digit might lose a piece of a stroke to a neighboring digit or a digit might lose a chunk because of overlapping. These problems arise because finding the precise splitting path that separates touching digits is nontrivial (see Figure 8).

To split touching digits I tested two algorithms. Both algorithms start by finding the bounding box of the entire ZIP code in a given pattern. The bounding box is a rectangle with horizontal and vertical sides that encloses the ZIP code and touches its topmost, bottommost, leftmost, and rightmost points. The first algorithm would divide the area of the bounding box vertically into five equal segments. This algorithm tends to

leave segments with noise or parts of the neighboring digit(s). In order to filter out the noise found in each of the five segments, the largest component is located in each segment and everything else (noise or parts of neighboring digits) is discarded. The result of this algorithm is five noise-free segments each representing a single isolated digit ready to be converted into a feature vector. The second algorithm tested does the exact opposite of the first algorithm. Instead of segmenting the digits and then converting to vectors, this algorithm converts the entire ZIP code enclosed by the bounding box to one long vector and then divides it into five small vectors. The process of converting a ZIP code pattern into a vector is described in the Feature Extraction section.

When tested on training data the performance of the first algorithm was much better than the second algorithm, therefore, I chose to use the first algorithm in my model. *Joining segments of a digit.* Joining segments that belong to one digit is as challenging as separating touching digits. The challenges here are to determine which segment belongs to which digit and whether a segment is part of a digit or simply noise. Furthermore, using the improper joining algorithm may leave some of the digits with artifacts or noise. These problems arise because finding the precise technique to join segments is nontrivial. A digit may occur in pieces as a result of three things: not having the proper writing tool (dry pen), noise or extra markings added during processing, or the writer did not properly connect the digit. By analyzing various cases of broken digits, I found that the digit which often occurs in two segments is the digit 5. With the exception of the digit 4, the digit 5 is the only digit which often is written in two parts (see Figure 9). Therefore, the digit 5 is perhaps more likely to appear in two parts than any of the other digits.

To solve this problem I used the single line test algorithm. This algorithm is commonly used in mathematics [5]. It starts by drawing a vertical line and moving it across the entire ZIP code starting from the left-hand-side. If the line intersects the area of two of the connected components found earlier, then those two components are joined. In case of a failure to find two components to connect, the ZIP code is simply divided into five equal segments as described in the previous section.

Feature Extraction

Since different individuals can have various writing styles, the features extracted from each digit must be independent of size, width of the strokes, and writing styles of the individuals. To extract features independent of writing styles, my model samples a number of pixels from each pattern. The sampled pixels are then stored in a matrix structure, as shown in Figure 10. The matrix is then reduced to a vector by projecting it onto the X-axis, as shown in Figure 11. The resulting vector will contain values which represent the number of black pixels found in each column of the matrix. A similar technique was used by Rababaah [13] to reduce the size of asphalt pavement crack patterns.

The size of a feature vector is related to the issue of efficient data representation. Determining the proper sample size means extracting meaningful features using the smallest sample size possible. My initial sample size was 100 pixels (5×20). However, the sample size was not sufficient to represent complex digits. Careful evaluation of the size, width of the strokes, and the shapes of the training patterns indicated that a sample size of 208 pixels (13×16) would be better. The new sample size allowed extraction of meaningful features that better describe complex digits. However, for some small

extracted digits, the 13×16 sample size presented a problem. Because the sample size was greater than the size of the digit pattern in these cases, there were not enough pixels in the pattern from which to sample. To solve this problem I used all the pixels in the pattern and completed the rest of the vector with zeros. The Results and Discussion section describe the results obtained from using the two sample sizes.

Recognition

The task of the recognition stage is to recognize the string of individually segmented digits. Utilizing the destination state, the recognition stage tries to recognize each digit starting with the left-most digit, as shown in Figure 12. For the first two digits the system will make a decision on whether it is feasible to continue or not. For example, if the destination state is Indiana, then the first digit must be a 4, because all ZIP codes in the state of Indiana start with the digit 4. Before going any further, the model must verify that the first digit is a 4, as shown at the top of Figure 12. If it is not, the model will stop and reject the ZIP code. On the other hand, if the first digit is a 4, then the model will proceed to classify the next digit. In this example, the model now must determine if the second digit is a 6 or 7, because in the state of Indiana these are the only possible digits after the first one. If the model classifies the second digit as a 6 or 7, then it will continue classifying the rest of the digits. Otherwise it will stop and reject the ZIP code, as shown in Figure 12³. Recall that Morita and colleagues [11] used a similar strategy to classify handwritten dates (days/years) using domain knowledge. They were able to reduce the complexity of the recognition process by reducing the date lexicon size. This was possible because they knew the lexicons for day and year. For example, in a two digit day

³ Rejection does not stop the processing of a given ZIP code; instead the model passes the digits to another network to be classified as one of ten possible digits. This allows for seamless comparison to the results obtained by Liu et al. [8].

the first digit can only be 0, 1, 2 or 3 while the second digit can range from 0 to 9. A similar approach was applied to two and four digit years. For instance, in a two digit year, they only allowed the first digit to be 0, 1, 2 or 9. The second digit can range from 0 to 9. In their model, they identified five different metaclasses of digits: one metaclass for digits 0, 1, 2, and 3, one for digits 0, 1, 2, 9, one for digits 1, 2, one for digits 0, 9, and one for digits 0 through 9. In my model, knowledge of the destination state may help in the recognition of the first two digits, because only the first two digits uniquely identify a particular state. The other digits must be classified by the model without any assistance from domain knowledge.

Classification Method

The recognition portion of this model was implemented using multi-layer perceptron (MLP) neural networks. Each individual network classified a different metaclass of digits. This was inspired by the use of distinct classifiers for distinct metaclasses of digits in the study of Morita et al. [11]. One of ten MLP networks was used to classify the first digit of a ZIP code, depending on the state. One MLP network classified the digits 6 and 7 for the second digit for the state of Indiana. Another classified the digits 0, 1, and 2 for the second digit for the state of Illinois, and so forth for other states. This strategy required a total of twenty-six different MLP networks. Recall that in a five digit ZIP code the first digit corresponds to one of ten large geographic areas in the country. Therefore, I used ten networks to classify the first digit in a ZIP code. Also, by exploiting the structure and range of possible ZIP codes for each state, I was able to create a list of possible second digit(s) for each state, as shown in Table 6. This table shows how the second digit for some states can correspond to a single digit

(e.g. Montana – 9), two digits (e.g. Indiana – 6,7), three digits (e.g. Missouri – 3,4,5), four digits (e.g. Virginia – 0,2,3,4), five digits (e.g. Texas – 5,6,7,8,9) or seven digits (e.g. California – 0,1,2,3,4,5,6). Since some of these states share similar sets of possible second digit(s) - for example both Kansas and Indiana have the digits 6 and 7 as possible second digits - I was able to eliminate redundancies among metaclasses and compile a list of metaclasses needed for this model, as shown in Table 7. After analyzing the testing data found on CEDAR CDROM-1 [3] (the standard test database), I found that the metaclass needed to classify whether the second digit in a California ZIP code is 0, 1, 2, 3, 4, 5 or 6 can be eliminated due to insufficient testing data, as shown in Table 8⁴. Finally, to classify the third, fourth and fifth digits of a ZIP code, I used the last metaclass listed in Table 7. Recall that Morita and colleagues [11] used five MLP neural networks in their model to classify the five different metaclasses of digits, and they had good success. The performance of their model on four digit years was 100%. On two digit days performance was 93.2% and on two digit years the performance was 97.2%. (The discrepancy between performance on two digit years and two digit days was explained by the fact that year always appears at the end of a date on Brazilian bank checks, which led to an improvement in the segmentation and, thus, recognition of two digit years.) Liu and colleagues [8] also tested a MLP network as one of the classifiers in their model to recognize handwritten ZIP codes. Their model integrated segmentation and a single MLP for recognition and did not make use of domain knowledge. When trained without outliers, the model's performance was 63.99%. However, when trained with outliers the

⁴ The testing data found on CEDAR CDROM-1 [3] does not provide ZIP codes for California with 0, 1, 2, 3, or 4 as the second digit. Therefore the 26th metaclass listed in Table 7 is not needed and can be replaced with the one that classifies the digits 5 and 6 instead.

model's performance was 89.22%. This compares favorably to the best performance of 90.37% obtained using the LQDF classifier in their model.

Classifier structure. The input to each network used for classification is a vector of 13 features. The hidden layer in each network contains an empirically determined number of hidden units, each of which is connected to all input and output units. The number of hidden units for all of the networks is five, with the exception of the last two networks (26 , 27) shown in Table 7. The network numbered 26 requires seven hidden units while the one numbered 27 requires nine hidden units. The output layer for each of the MLP networks contains a number of output units corresponding to the number of digits in each metaclass. For instance, a MLP network intended to classify whether a given digit is a 4 must have two output nodes in the output layer, one for the digit 4 and one for any other digit. A MLP network intended to classify a particular digit as 6 or 7 must have three nodes in the output layer, one for the digit 6, one for the digit 7, and one for any other digit, for example.

Training data. To train the classifiers I used the same dataset used by Liu et al. [8]. The classifiers were trained on data compiled from the NIST Special Database 19 (SD19). I used 66,214 digit samples from the segmented hand-printed digits found in SD19. I also generated 16,000 outlier training patterns from the training digit data. The idea of combining outliers with the training data was introduced by Liu et al. [8]. They showed that it improves classification. I generated outlier patterns, using their technique, by merging and splitting training images. A pair of digit images generated four outlier patterns: full-full combination, full-half combination, half-full combination, and half-half combination as shown in Figure 13. The patterns were generated by first arbitrarily

selecting two digits (see top of Figure 13). The first digit is split vertically into two parts, A and B. The second digit is also split vertically into two parts, C and D. In order to generate four outliers, parts must be joined together as follows: A, B, C will make one outlier (Full, Half). Then, B, C, D will make another outlier (Half, Full). Then A, B, C, D combined together make another outlier (Full, Full). Finally, I combined B and C (Half, Half).

Testing data. The system was tested on 436 5-digit ZIP code images found on CEDAR CDROM-1 in the Binary ZIP code directory (BINZIP). These strings were used by Liu et al. [8] to test their classification methods. The string images were extracted from live mail images of USPS [18].

Results and Discussion

The performance of ZIP code recognition was evaluated using the same dataset used by Liu and colleagues [8] to test their system. Since my model relies on prior knowledge of destination state, this information had to be identified beforehand. To achieve recognition of a given ZIP code, the segmented vectors for a ZIP code, along with the destination state, are passed to a recognition script (see Appendix). The script classifies digits by calling the appropriate neural network for a given state and digit position.

The performance of my model has been evaluated over three different versions. Recall from the Feature Extraction section that meaningful features are vital to achieving high performance, and acquiring meaningful features depends primarily on the sample size. The first version of my model was tested using 5×20 sample size, while the second version used a sample size of 13×16 . The correct recognition rate achieved by the first

version was 65.6% with no rejection. The correct recognition rate achieved by the second version was 88.76%, also with no rejection. Clearly, using the larger sample size facilitated capturing the features necessary for recognition of handwritten ZIP codes.

The third version of my model was created to compare against the second version, in order to measure the effectiveness of applying domain knowledge to the recognition of handwritten ZIP codes. This third version applied domain knowledge in the segmentation stage only and ignored any information attained from the destination state or ZIP code structure in the recognition stage. The model's performance was 75.69% with no rejection. Compared to the second version, which had performance of 88.76% with no rejection, it is clear that domain knowledge improves the recognition of handwritten ZIP codes.

To my knowledge, the best recognition performance achieved on the CEDAR dataset was reported by Liu and colleagues [8]: 90.37% correct rate with no rejection using the LQDF classifier. More importantly for this study, Liu et al. [8] achieved 89.22% correct recognition with no rejection using the MLP classifier in their model. This compares favorably with the performance I found – 88.76% with no rejection. Despite our comparable performance, I believe that my model has revealed two interesting new concepts. First, a complex problem such as recognizing ZIP codes can have a simple, easy, and straightforward solution. Recall that Liu and colleagues [8] claimed that the problem could not be solved without having an integrated segmentation and recognition system. I have shown that this is not true. My model uses a distinct segmentation stage followed by a recognition stage, and it performed comparably to their model. Compared to their system, my model took a simpler approach to solve the

problem. My segmentation stage is straightforward, including some of the same elements of their “pre-segmentation” stage. In the recognition stage, even though I used 26 neural networks in my model, these networks are not complex and were easy to train.

The second interesting concept revealed by my model is that domain knowledge can aid in the classification of ZIP codes. That is, the technique of Morita et al. [11] seems to extend beyond the classification of Brazilian check dates. Perhaps it can be applied to solve other digit string problems such as dollar amount on checks, and social security number or driver license number on forms. By utilizing domain knowledge I was able to simplify the structure of many neural networks. For example, a neural network used to classify the first digit in a ZIP code is no longer classifying the digit among 10 classes (0 to 9), instead it is using 2 classes. An indication for the effectiveness of domain knowledge in recognizing handwritten ZIP codes was shown by comparing the second (main) version of my model to the third version (which did not use domain knowledge in the recognition stage). This third version simply used one MLP neural network to classify each digit in the ZIP code without taking into consideration the destination state and the structure of ZIP codes. To tie the two discoveries derived from my model together, the claim of Liu et al. [8] about the need for integrated segmentation and recognition seems reasonable if you are not using domain knowledge. A detailed comparison between the model of Liu et al. [8] and my third version shows that we both used MLP classifiers, and we both used the EDL (outlier) learning method. I trained the classifiers using the same MNIST dataset they did and generated outliers in the same way as they did. My performance of 75.69% was lower than their performance of 89.22%, probably due to the fact that I used a distinct segmentation stage followed by a recognition stage. However,

when domain knowledge was applied, as in the second (main) version of my model, there was no need for a complex model that integrates segmentation with recognition.

CONCLUSION

For the task of machine recognition of handwritten ZIP codes, I have shown that combined with simple algorithms, domain knowledge can be used to achieve good performance. In my Experiment 2, however, I showed that human performance in the task of recognizing handwritten ZIP codes is 98.39%. This indicates that there is still room for improvement in machine performance. In my model, improvements can be made in the following areas:

- Segmentation of digits can be improved to better handle touching and broken digits. Experimenting with different algorithms to isolate components and analyze digits by looking at their length, curvature, and area may help in identifying different digits.
 - Feature vectors can be improved by using a different sample size or by extracting different types of features that better represent the data such as curvature of digit, number of holes/corners in a digit, and so on.
 - Different classification algorithms can be tested to see if they provide better performance. Perhaps applying LQDF classification along with domain knowledge might yield better performance than that obtained by my model and by Liu et al. [8].
- The improvements suggested could bring the performance of my model closer to human performance and provide a solution which could be implemented in many real world applications.

REFERENCES

- [1] Bouchaffra, D., Govindaraju, V., & Srihari N. S. (1999). Recognition of Strings Using Nonstationary Markovian Models: An Application in ZIP Code Recognition. Conference on Computer Vision and Pattern Recognition (CVPR), 2174-2179.
- [2] Bromley, J., & Sackinger, E. (1991). Neural-network and k-nearest-neighbor classifiers. Tech. Rep. 11359-910819-16TM, AT&T.
- [3] CEDAR CDROM1 Specifications of the Databases. Retrieved September 8, 2005, from <http://www.cedar.buffalo.edu/Databases/CDROM1>
- [4] Dong, J., Xiong, K. A., & Suen, C. Y. (2002). Statistical Results of Human Performance on USPS Database. Retrieved April 8, 2005, from <http://www.cenparmi.concordia.ca/people/jdong>
- [5] Etgen, J.G. (1999). Salas and Hille's Calculus: One Variable. Eighth edition. John Wiley & Sons, INC.
- [6] Gonzalez, C., and Wood, E. (2002). Digital Image Processing. Second Edition, Prentice Hall, New Jersey.
- [7] Kohonen, T. (1990). The Self-Organizing Map, Proc. IEEE, vol. 78, PP . 1464-1480.
- [8] Liu, L. C., Nakashima, K., Sako H., & Fujisawa, H. (2002). Integrated Segmentation and Recognition of Handwritten Numerals: Comparison of Classification Algorithms. International Workshop on Frontiers in Handwritten Recognition (IWFHR), 8, 303-308.
- [9] Liu, L. C., Nakashima K., Sako H., & Fujisawa H. (2002). Handwritten Digit Recognition Using State-of-the-Art Techniques. Pattern Recognition, 36, 2271-2285.
- [10] Mead, Judson. (1991). Speed Reading. Journal of University of Buffalo Research, vol1.1 page 2 and 3. Retrieved April 8, 2005, from http://www.cedar.buffalo.edu/pub_docs/article41.html
- [11] Morita, M., Sabourin, R., Bortolozzi, F. & Suen, Y. C. (2004). Segmentation and recognition of handwritten dates: a HMM-MLP hybrid approach. International Journal on Document Analysis and Recognition (IJ DAR) 6, 248-262.
- [12] Qizhi, X., Louisa, L., & Suen, Y. C. (2003). Automatic Segmentation and Recognition system for Handwritten Dates on Canadian Bank Cheques.

International Conference on Document Analysis and Recognition (ICDAR), 7, 704 -712.

[13] Rababaah, H. (2005). Asphalt Pavement Crack Classification Using AI and Computer Vision (Masters Thesis, Indiana University, 2005).

[14] Russell, S. & Norvig, P. (2002). Artificial Intelligence: A Modern Approach. Second Edition, Prentice Hall, New Jersey.

[15] Shapiro, G.L., & Stockman, C.G. (2001). Computer Vision. Prentice Hall, New Jersey.

[16] The Centre for Pattern Recognition and Machine Intelligence (CENPARMI) Database. Retrieved September 8, 2005, from <http://www.cenparmi.concordia.ca/>

[17] The MNIST Database of Handwritten Digits. Retrieved September 10, 2005, from <http://yann.lecun.com/exdb/mnist>

[18] United States Postal Service Database. Retrieved September 8, 2005, from <http://www.kernel.org/data.html>

[19] Zhang, L. O., & Suen, Y. C. (2002). Recognition of Courtesy Amounts of Bank Checks based on a Segmentation Approach. International Workshop on Frontiers in Handwriting Recognition (IWFHR), 298-302.

[20] ZIP-Codes. Retrieved March 15, 2006, from <http://www.zip-codes.com>

FIGURES

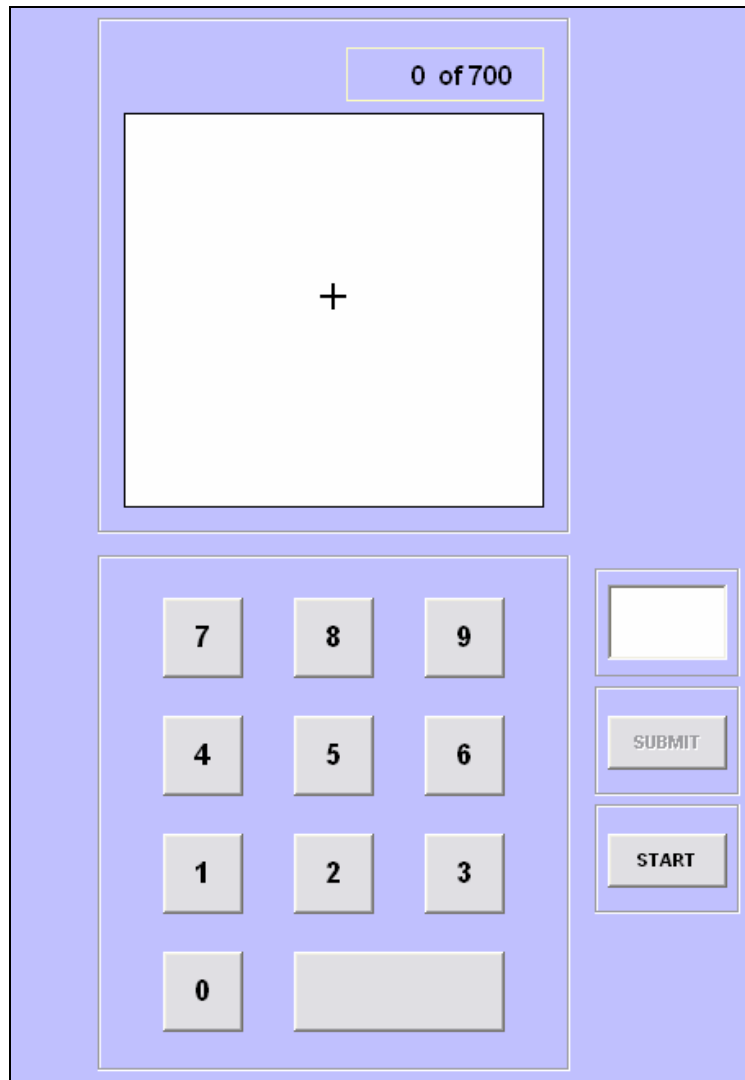
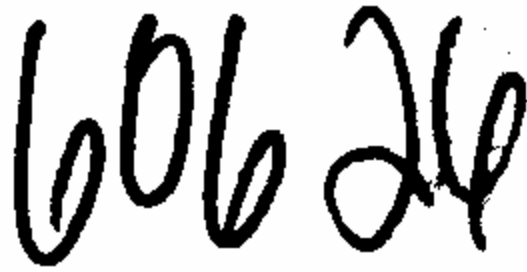


Figure 1. A screenshot of the software used in Experiment 1.

A handwritten ZIP code '60626' in black ink on a white background. The digits are somewhat irregular and cursive.

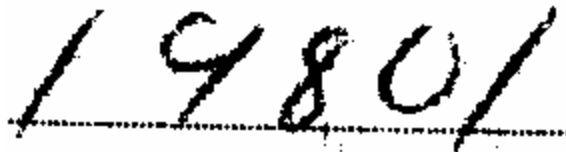
86 - (60626)

A handwritten ZIP code '97222' in black ink on a white background. The digits are bold and somewhat irregular.

92 - (97222)

A handwritten ZIP code '82071' in black ink on a white background. The digits are somewhat irregular and cursive.

119 - (82071)

A handwritten ZIP code '19801' in black ink on a white background. The digits are somewhat irregular and cursive. A horizontal dashed line is drawn below the digits.

469 - (19801)

Figure 2. Sample patterns from the CEDAR CDROM-1 database. The number at left represents the pattern number. The number at right is the correct classification of the ZIP code.

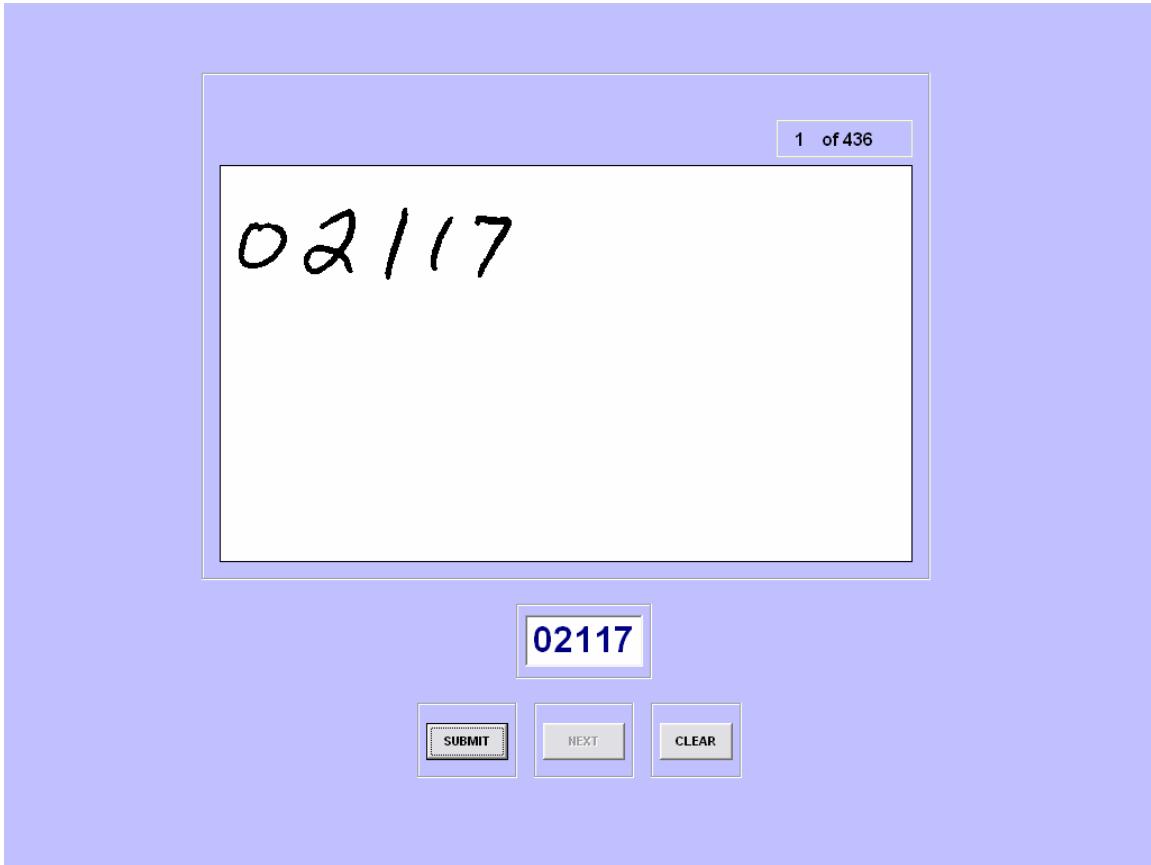


Figure 3. A screenshot of the software used in Experiment 2.

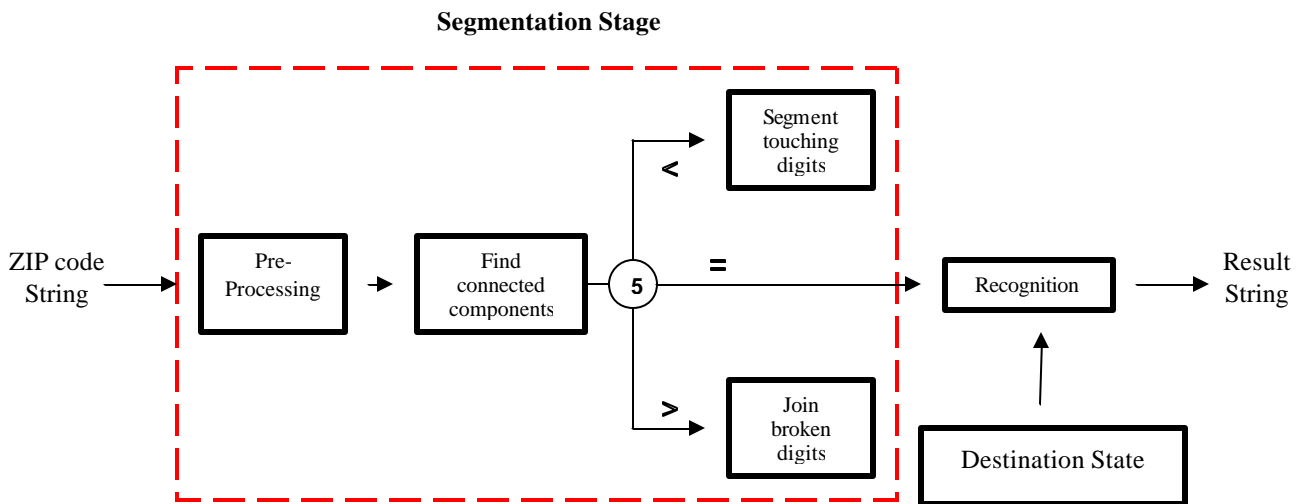
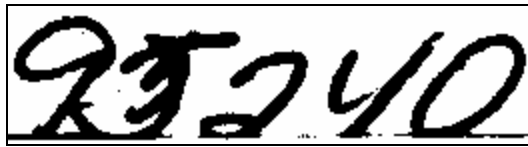


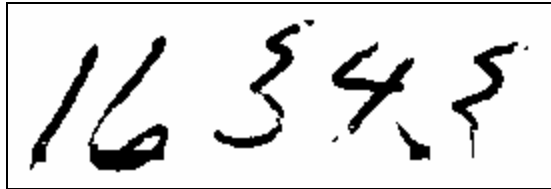
Figure 4. The two stages (segmentation & recognition) of my model.

A rectangular box containing the handwritten number '48708' in a cursive style. The '4' is formed with a single stroke, and the '8' is a closed loop.

70

A rectangular box containing the handwritten number '95240' in a cursive style. The '9' is a closed loop, and the '0' is a closed loop.

105

A rectangular box containing the handwritten number '16345' in a cursive style. The '1' is a simple vertical stroke, and the '5' has a small hook at the end.

172

Figure 5. Sample patterns from the CEDAR CDROM-1 database.

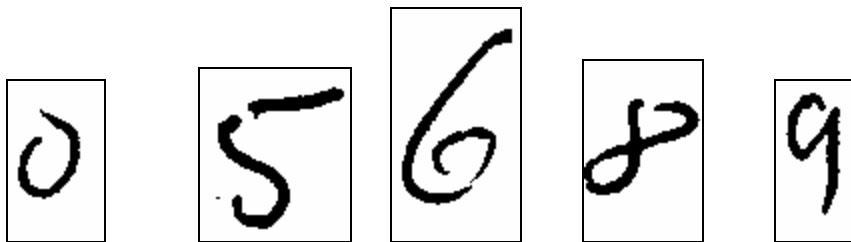


Figure 6. Sample patterns extracted from ZIP codes found on the CEDAR CDROM-1 database. The patterns show how some digits are incomplete because of gaps.

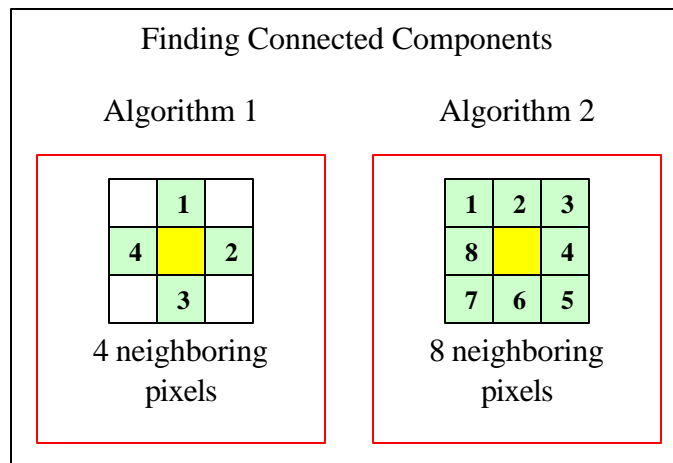
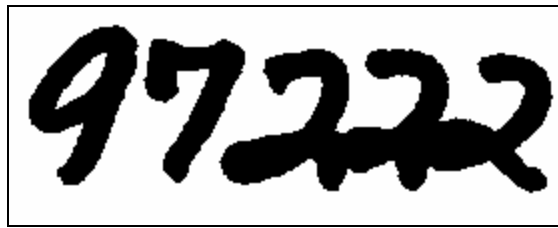
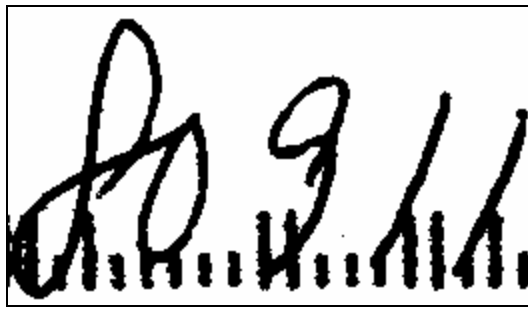


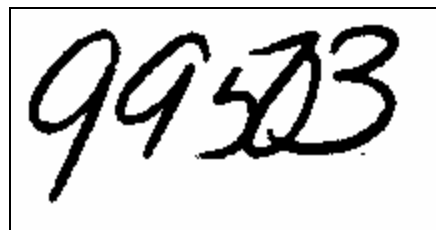
Figure 7. The two algorithms tested for finding the connected components in a ZIP code string.



81 - 97222



163 - 80911



216 - 99503

Figure 8. A sample of ZIP codes from CEDAR-CD-ROM 1. The samples show that finding the precise splitting path is nontrivial.

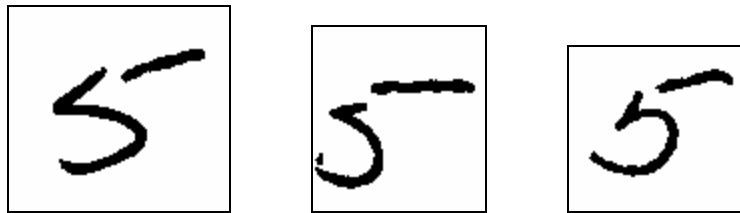
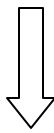


Figure 9. Sample patterns extracted from ZIP codes found on the CEDAR CDROM-1 database. The patterns show how the digit 5 is sometimes presented in two segments.

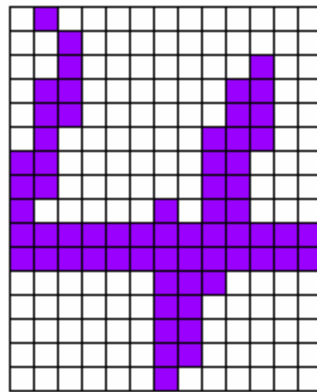
Original Segmented Pattern



Pattern size 37×40



Normalize the pattern by sampling:
Sample Size 13×16



13×16 Matrix

Figure 10. Normalizing the segmented digits by sampling.

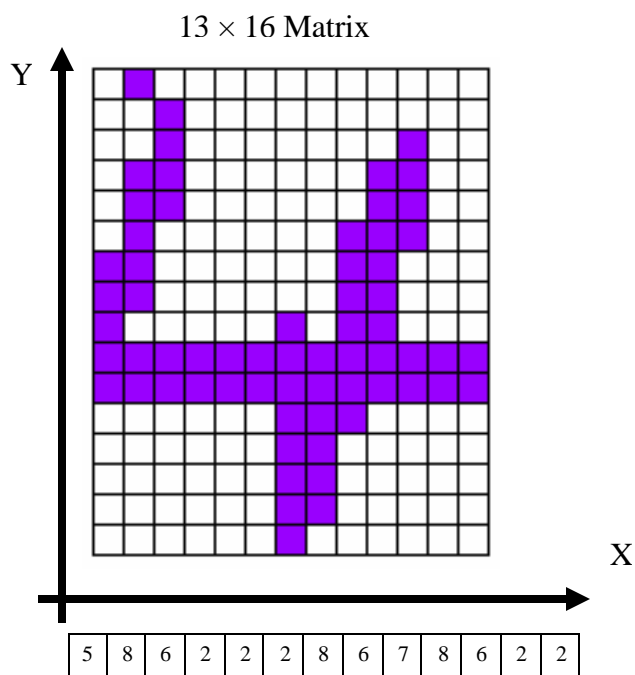
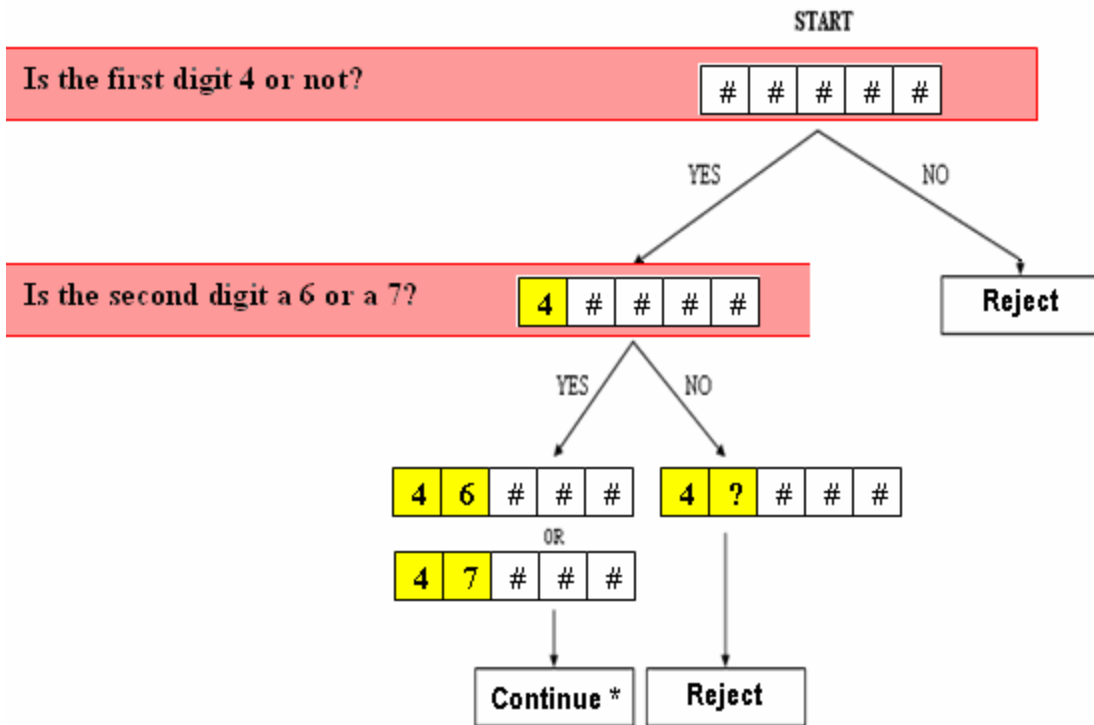


Figure 11. Converting the 13×16 matrix into a vector by projecting onto the X-axis.



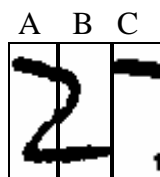
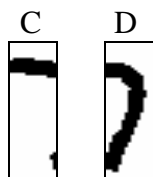
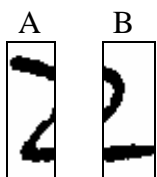
Continue* - The model will classify the three remaining digits.

Figure 12. The knowledge-based recognition model starts with five unknown segmented digits. The model will first attempt to classify the left-most digit. If unsuccessful the ZIP code is rejected, otherwise it will go on to classify the second digit. If the second digit is classified correctly the model will classify the remaining three digits, otherwise it will stop. (Indiana is the state assumed in this example.)

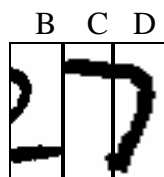
First Digit



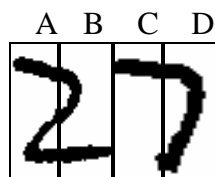
Second Digit



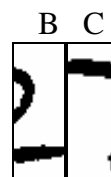
Full - Half



Half - Full



Full - Full



Half - Half

Figure 13. A sample of outliers generated from the NIST SD19 dataset.

TABLES

PATTERN NUMBER	16 × 16 IMAGE	USPS LABEL
234	4	1
971	1	4
994	0	5
1978	3	5

Table 1. This table shows the USPS misclassified patterns. The first column (leftmost) shows the pattern numbers, the middle column shows the actual 16×16 patterns, and the third column (rightmost) shows the USPS labels of the patterns. In each case, note that the USPS labeling appears to be incorrect.

Subject #	Session 1 Correct Responses	Session 2 Correct Responses	Session 3 Correct Responses	Total Correct Responses	Percent Error
1	688	682	590	1960	2.34%
2	685	683	592	1960	2.34%
3	685	681	594	1960	2.34%
4	681	676	585	1942	3.24%
Total Number of Trials	700	700	607	2007	
Average Percent Error					2.57%

Table 2. Results obtained from Experiment 1. The average percent error was 2.57%.

Subject #	Session 1 Correct Responses	Session 2 Correct Responses	Session 3 Correct Responses	Total Correct Responses	Percent Error
1	688	682	590	1960	2.15%
2	685	683	592	1960	2.15%
3	685	681	594	1960	2.15%
4	681	676	585	1942	3.05%
Total Number of Trials	699	698	606	2003	
Average Percent Error					2.37%

Table 3. Results obtained from Experiment 1. After removing the four trials with mislabeled images, the average percent error was 2.37%.

Subject #	Errors	Percent Error
1	10	2.29%
2	8	1.83%
3	39	8.94%
4	3	0.69%

Average Percent Error	3.44%
------------------------------	--------------

Table 4. Results from Experiment 2. The average percent error was 3.44%.

Subject #	Errors	Percent Error
1	10	2.29%
2	8	1.83%
4	3	0.69%

Average Percent Error	1.61%
------------------------------	--------------

Table 5. Results from Experiment 2, with the result of subject 3 omitted. The average percent error after removing this outlier was 1.61%.

States (ZIP code starts with 0)		Possible Second digit	States (ZIP code starts with 5)		Possible Second digit
CT	Connecticut	6	IA	Iowa	0 - 1 - 2
MA	Massachusetts	1 - 2	MN	Minnesota	5 - 6
ME	Maine	3 - 4	MT	Montana	9
NH	New Hampshire	3	ND	North Dakota	8
NJ	New Jersey	7 - 8	SD	South Dakota	7
RI	Rhode Island	2	WI	Wisconsin	3 - 4
VT	Vermont	5			
States (ZIP code starts with 1)		Possible Second digit	States (ZIP code starts with 6)		Possible Second digit
DE	Delaware	9	IL	Illinois	0 - 1 - 2
NY	New York	0 - 1 - 2 - 3 - 4	KS	Kansas	6 - 7
PA	Pennsylvania	5 - 6 - 7 - 8 - 9	MO	Missouri	3 - 4 - 5
			NE	Nebraska	8 - 9
States (ZIP code starts with 2)		Possible Second digit	States (ZIP code starts with 7)		Possible Second digit
DC	District of Columbia	0	AR	Arkansas	1 - 2
MD	Maryland	0 - 1	LA	Louisiana	0 - 1
NC	North Carolina	7 - 8	OK	Oklahoma	3 - 4
SC	South Carolina	9	TX	Texas	5 - 6 - 7 - 8 - 9
VA	Virginia	0 - 2 - 3 - 4			
WV	West Virginia	4 - 5 - 6	States (ZIP code starts with 8)		Possible Second digit
States (ZIP code starts with 3)		Possible Second digit	AZ	Arizona	5 - 6
AL	Alabama	5 - 6	CO	Colorado	0 - 1
FL	Florida	2 - 3 - 4	ID	Idaho	3
GA	Georgia	0 - 1	NM	New Mexico	7 - 8
MS	Mississippi	8 - 9	NV	Nevada	8 - 9
TN	Tennessee	7 - 8	UT	Utah	4
			WY	Wyoming	2 - 3
States (ZIP code starts with 4)		Possible Second digit	States (ZIP code starts with 9)		Possible Second digit
IN	Indiana	6 - 7	AK	Alaska	9
KY	Kentucky	0 - 1 - 2	CA	California	0 - 1 - 2 - 3 - 4 - 5 - 6
MI	Michigan	8 - 9	HI	Hawaii	6
OH	Ohio	3 - 4 - 5	OR	Oregon	7
			WA	Washington	8 - 9

Table 6. A grouping of states with same first ZIP code digit and all possible second digit(s) for each state [18].

#	MLP Metaclass
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	0
11	0 - 1
12	1 - 2
13	2 - 3
14	3 - 4
15	5 - 6
16	6 - 7
17	7 - 8
18	8 - 9
19	0 - 1 - 2
20	2 - 3 - 4
21	3 - 4 - 5
22	4 - 5 - 6
23	0 - 2 - 3 - 4
24	0 - 1 - 2 - 3 - 4
25	5 - 6 - 7 - 8 - 9
26	0 - 1 - 2 - 3 - 4 - 5 - 6
27	0 to 9

Table 7. A list of all MLP metaclasses needed for the model, including the not needed metaclass numbered as 26.

ZIP Codes		First Digit									
Second Digit		0	1	2	3	4	5	6	7	8	9
	0	5	0	7	10	7	1	9	10	5	0
	1	1	7	8	0	1	0	0	1	0	0
	2	8	0	8	2	2	7	1	11	7	0
	3	6	7	0	0	13	7	11	8	6	0
	4	0	5	0	1	0	1	0	3	6	0
	5	6	0	4	7	1	7	1	0	6	8
	6	5	15	1	2	11	0	13	0	1	10
	7	5	1	2	10	0	3	0	12	6	11
	8	1	0	5	0	12	6	12	0	2	7
	9	1	8	7	9	4	2	1	0	6	13

Table 8. Distribution of testing dataset found on CEDAR CDROM-1 [3], according to the first and second digits. A value of 0 represents 0 ZIP codes found in the testing data with the corresponding column and row numbers as the first and second digits in the ZIP code.

APPENDIX