RECOGNITION AND VERIFICATION OF UNCONSTRAINED HANDWRITTEN NUMERALS

JIE ZHOU

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
CONCORDIA UNIVERSITY
MONTRÉAL, QUÉBEC, CANADA

NOVEMBER 1999

0-612-47716-9

Abstract

Recognition and Verification of Unconstrained Handwritten Numerals

Jie Zhou, Ph.D.
Concordia University, 1999

Despite the success of many recognition systems for handwritten numerals within constrained domains, the problem remains difficult when unconstrained inputs are involved. The gap between the state-of-the-art machine recognition reliability and high practical demand leads to this investigation of verification scheme in pattern recognition.

A pattern verifier is an expert specially trained to reliably confirm or negate a pattern identity from the General Purpose Recognizer (GPR), with the intention to significantly improve the class-specific *Precision Rates* of the system. The main goal of this thesis is to study the promising and critical role of a verifier in a recognition system. Theoretical aspects of a verifier including its unique task and functionality, inherent requirement, evaluation measurement, design concern and control strategy are discussed throughout the thesis, focusing on the problems of recognizing Unconstrained Isolated Handwritten Numerals (UIHN) and Unconstrained Touching Handwritten Numerals (UTHN). For each problem, an integrated recognition and verification system is designed and evaluated by incorporating together the GPR and the verifier.

The GPR for UIHN is a combination of three conventional neural approaches. In the design of class-specific verifier for UIHN, a new kind of neural network — Quantum Neural Network (QNN) — with better distinguishing ability along decision boundary, is embedded in an efficient way. Novel experiments have been designed for in-depth studies of applying the QNN to both real data and confusing images synthesized by morphing. CENPARMI database and MNIST database are used for evaluation.

UTHN recognition is an important component for automatic document processing in applications such as cheque processing. However, it is a more difficult problem that has attained less attention, reflected by the mediocre performance of current systems and lack of benchmarking databases. Two databases IRIS-Bell'98 and NIST for UTHN are newly built by the researchers at CENPARMI and the author. They are used in this research and are intended to serve as standard databases in this field. A novel graph-based combination of segmentation and recognition schemes is used in GPR for UTHN. Effective domain specific strategies making use of touching type, touching location and structural information are applied in the verifier for UTHN.

The recognition and verification system for UIHN achieved a precision rate of 99.1% on MNIST database while the one for UTHN reached a precision rate of 96.1% on NIST database. The two systems are also evaluated by hypothesis testing. The substantial improvement of system precision rates by verification scheme proves the effectiveness of the proposed systems and justifies the important role of verifiers in the OCR system.

Acknowledgments

First and foremost I would like to express my sincere gratitude to my supervisor Dr. Ching Y. Suen for offering me such a wonderful environment and for his assistance at so many levels. I also thank him for suggesting this topic and for carefully reviewing this thesis. It is his supervision and support that have made this work possible.

I am also grateful to my co-supervisor Dr. Adam Krzyzak for his guidance and encouragment during the course of this work while for his careful revisions of my publications.

Special thanks also extend to my previous director Dr. Qiang Gan for his suggestions and triggering ideas.

My thanks go to Dr. Ke Liu who provided precious discussions, to Nick Strathy who shared his experience with me in the field, and to Guiling Guo for her help during tagging the database for touching numerals. I am also indebted to Christine Nadal and Alceu S. De Britto for their help in building the database.

I would like to thank all the friends at CENPARMI for their help and friendship that enriched my study these years: Qizhi, Hao, Mary, Danny, Didier, Rita, Fady and Yousef. I also shared nice memory with visitors to CENPARMI, in particular with Zheng Lou, Dr. Il-Seok Oh, Dr. Mounim El Yacoubi and Dr. Yuan Y. Tang.

The love and support from my parents and my husband Jun Zhang are so crucial for the completion of this work and are hard to thank by words. Their care through phone lines and emails is the backbone of this work.

This research has been supported by research grants from the Natural Sciences and Engineering Research Council of Canada and the IRIS National Network of Centres of Excellence of Canada, and FCAR research program of the Ministry of Education of Quebec.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Would you tell me, please, which way I ought to go from here?"
"That depends a good deal on where you want to get to."
— Lewis Carroll, Alice in Wonderland

1.1 OCR: Motivations

Optical Character Recognition (OCR) is one of the most successful applications of automatic pattern recognition [94, 96, 70] . Research on OCR began in the mid 1950s. Today, reasonably good OCR systems are available for only a few hundred dollars. However, they are only good at recognizing high quality printed text documents or neatly written handprinted texts. The current research on OCR is addressing more diversified and sophisticated problems, including severely degraded, omnifont machine-printed texts and unconstrained handwritten texts, analysis and recognition of complete documents. Presently, OCR research is driven by the following needs:

1. Large volume document processing and data entry applications. Among these are: postal address recognition for mail sorting, financial document processing

(cheques, credit card slips) and form processing (tax forms, census forms etc.).

2. Paperless office for automated document inputting including OCR of existing texts or faxed information.

3. Digital library and other web-based OCR services along with the popularity of internet.

4. "Intelligent input" requirements for next generation computers.

Currently, the most diffuse data acquisition devices for off-line recognition systems are the optical scanners which can be of four types: flatbed, paper-fed, handheld and drum scanners.

In a typical OCR system, the input document is scanned by an optical scanner to produce a gray-level or binary bit-mapped image. Components of interest are then extracted and fed to recognition engines to get the final results, such as identities of characters, words or numeral amounts. Contextual postprocessing or language understanding modules may also be included.

1.2 Focus of Our Work

The focus of our work is recognition and verification of unconstrained handwritten numerals, which is an important aspect of OCR problem. However, the concept and theory of verification methods that we propose in the thesis are not restricted to the focused problem. Instead, they can be applied OCR systems of wider scope and even general pattern recognition problems.

Unconstrained handwritten numerals are handwritten numerals that are not written in separate boxes, nor written neatly, nor written with a specific type of pen. Hence, the goal is to recognize numbers written by people in real-life situations such as zip codes on envelopes and courtesy amounts on cheques.

Our work started with recognition of unconstrained isolated handwritten numerals (UIHN problem). Case study of our theory and methods have been conducted on standard databases of isolated numerals including CENPARMI database and NIST database. Driven by the inherent goal of recognizing numerals in real applications, our efforts were extended to the field of unconstrained touching handwritten numerals (UTHN problem), typically extracted from financial documents, such as courtesy amounts of English/French cheques and bill slips.

Recognition of touching numerals is generally considered as a more complicated problem due to many application-specific reasons (see Chapter 5). While examining the two problems in a consistent way, we can actually see several *common* reasons that contribute to the unsatisfactory performance of many handwritten numeral recognition algorithms:

1. Variety of handwritten numerals. Features used by recognition system may be character-intrinsic (having little relation with writer), or writer-dependent (varying with personal idiosyncrasies). Variety of these two kinds of features causes the diversity of writing style.

2. Ambiguity of handwritten numerals. Handwritten numerals tend to have ambiguous appearances which may cause problem to machines while humans can still recognize them by other information such as context, heuristic guess, etc..

3. High expectation of the industry. OCR applications are expected to be used in many fields with high demand for accuracy, e.g., cheque reading system or financial bill slip processing system.

The third aspect above deserves some further discussion. Although OCR systems are expected to improve the efficiency in many areas, current systems still incur lots of post-processing cost since people have to *verify* the recognition results by hand. In the case of an automatic form reading system, assuming a recognition rate of 90%, we will need 10 key-input corrections for each form containing 100 numerals. With 10,000 forms a day, which is a reasonable assumption in many industrial situations, at least 100,000 extra key entries will be needed. We also note that errors are much more intolerable than rejections since extra effort is necessary to detect errors. Thus very high reliability is expected from the system where reliability is defined as

$$reliability = \frac{correct\ recognition\ rate}{correct\ recognition\ rate\ +\ substitution\ rate}$$

.

That explains why we have made substantial effort towards improving the reliability of recognizing unconstrained isolated and touching handwritten numerals.

1.3 State of the Art

1.3.1 Isolated Numeral Recognition

There have been decades of intensive study on the topic of isolated handwritten numeral recognition. Many researchers tried to improve the reliability of recognition system by considering various features, classification methods and sophisticated rules of multi-experts. This section presents a concise summary of these endeavours.

Features

According to Devijiver & Kittler [16], features are "the information which is most relevant for classification purposes, in the sense of minimizing the within-class pattern variability while enhancing the between-class pattern variability".

- The first kind of features are the color, gray-level or binary values of all the pixels in an image, usually represented by a N-dimensional vector, where N is the number of pixels in the image. Since no abstraction is applied, all the variances among the patterns are to be handled by a classification procedure. The conventional way of using these pixel features is a direct comparison which is called template matching. Usually the comparison is based on selected pixels to avoid big complexity, such as the method described by Casey and Nagy [8]. Another fast evolving approach is using pixel features as direct input of artificial neural networks. Desirable results have been reported [57, 58]. The reason of success can be explained that the neural network also acts as a feature extractor during the learning and weight forming [101].

- Second type feature is the predefined combination of pixel value, or the values that are defined on a local scope centered at one pixel. Amit et al. [1] proposed a feature based on the topography in their immediate neighborhood, which is referred as "tags". The arrangements of the tags are used as inputs of a tree classifier. The difficulty in using this kind of features is in designing the arrangements that will be sufficient to distinguish among all classes. They are often purposely designed to represent shapes and bear a resemblance to the geometric features.

- Features of the third type are deformable prototypes. Hastie et al. [33] used a piecewise-linear curve which is fitted to a point-set representation of the image. Classification is based on the matching statistics. There may be multiple prototypes for each class. Similar idea is used by Revow et al. [84]. In Jain et al. [48], the characters are matched by deforming the contour to fit edge strengths and measure the amount of deformations. The difficulty is in designing enough prototypes for the problem and the computation cost in comparing each prototype.

- Features of the fourth type are the structural features of the image, which are typically perceptual entities of the character such as bends, end points, loops or joints. The scope of definition over the image area is usually flexible. They can be detected from contour or skeleton. The set of chosen features are designed based on the property of the problem. Most of the complexity in recognition is handled by the feature detection algorithm. A typical successful system has been developed by Suen et al. [96]. There are also other good results reported [49, 41]. The down side of this method is the usually labor-intensive and heuristic feature extraction. To measure the confidence level is also not an easy task.

- Features of the fifth type are the results of global transformations on an image. Typical mathematical transformations include moments [9], Fourier descriptors [90] and the Walsh transform [42]. The difficulty in using these features for recognition is usually in defining an effective similarity among feature sets, such as among sets of Fourier descriptors.

Classification Methods

The classifiers include nearest-neighbor classifiers, Bayesian classifiers, polynomial discriminant classifiers, neural network classifiers, tree classifiers, syntactic approaches and Hidden Markov Model(HMM). They typically use feature descriptors in the form of vectors and return a class identity. The first three are classic statistical classifiers and were described in books by Duda and Hart [19], Devijver and Kittler [16], and Devroye et al. [17]. Neural network classifiers are more often used by recent researchers. Their inherent relationship with statistical classifiers has also been studied [88].

- The nearest-neighbor classifier performs direct prototype matching using a predefined distance to measure the similarity between a pattern and those in a class. The distance function can be Euclidean or Hamming distance. The problem with the method is the high computation cost when classification is conducted [53]. They are many variants of this approach with the intention to reduce the complexity. A famous one is k-nearest neighbor which finds k closest matches and uses a voting scheme to decide on the class. When pixel value is used directly, the method is referred as template matching.

- The Bayesian classifier assigns a pattern to a class with the maximum *a posterior* probability. The class prototypes are used in a training stage to estimate the class-conditional probability density function for a feature vector [19, 72].

- The polynomial discriminant classifier [88] assigns a pattern to a class with the maximum discriminant value which is computed by a polynomial in the components of a feature vector. The class models are implicitly represented by the coefficients in the polynomial.

- Syntactic classifiers [26] use grammars at all levels in the Chomsky hierarchy to describe class models. These grammars take in a high level descriptor such as a symbol string instead of feature vectors. The class models are abstracted as grammatical rules that can be used to generate the prototypes.

- Tree classifiers are motivated by the need to reduce the complexity in prototype matching. They are many design strategies [87, 21] but generally it is difficult to control the growing and pruning of trees. Commonly used control methods are mutual information, probability models or entropy values. The most famous tree classifiers are CART [3] and C4.5 [79]. Ho [38] extends the work to C4.5 Decision Forests and reports good results. Refer to Chapter 3 for more information.

- Hidden Markov Model (HMM) [81] is a statistical framework for modelling sequential input by state transitions. It has been widely used in speech recognition and online handwritten recognition. Its applications to offline handwritten recognition have been growing. Cai et al. [5] define the state of a given observation in HMM as micro-state and the collections of individual micro-states as macro-states. The statistical information of handwritten numeral is represented by micro-states using HMMs and the structural information is modelled by relationships between macrostates. Park et al. [76] use a 2-D HMM for character recognition.

- In the past decade, there has been a tremendous increase of interest in artificial neural networks as a possible solution to the problem of recognizing handwritten numerals [34, 36]. This has been the preferred approach and good results have been reported. Typically, there are multiple layers of interconnected nodes used to imitate the organization of neurons in biological systems. In the back-propagation learning phase, which will be explained in Chapter 3, the weights associated with each neuron are modified using an updating algorithm until the desired outputs are achieved.

The primary advantage of neural networks is their ability to be trained automatically from examples. There are also other merits including their good performance with noisy or incomplete data and possible parallel implementation. Artificial neural networks are adopted as the major classification approach in our work.

Other methods include the perturbation method [31] which considers the distortion of handwriting habits, rule-based system [63] and random graph method [105] which often make use of structural features.

Combinations

Combination of several classification approaches has attracted theoretical and practical attention in the recent years [96, 52, 83]. It can be categorized as multistage strategy or multiexpert strategy.

The main reason for combining classifiers in a multistage way is efficiency. Objects are first preclassified using a small set of cheap features with a reject option. The rejected samples then are passed to more complex procedures to get an identity [67, 75]. Notice that it is inherently different from the verification scheme whose major goal is reliability and the role of verifier is reliable confirmation or negation of the conclusion from previous stage, which will be explained in following sections.

In a multiexpert scheme, the images of numerals can be fed to two or more recognition systems simultaneously. The combination occurs at the decision level on the outputs of all systems. By selecting an identity among the individual outputs, the multiexpert system becomes a more accurate system as a whole.

Some theoretical work has been reported. Representative results can be grouped as follows:

- Xu et al. [107] invesigated the combination of output information at an abstract level. They first consider several variants of the simple voting principle, then examine the combination of multiple classifiers in the Bayesian formalism and Dempster-Shafer formalism.

- Huang and Suen [44, 43] proposed the Behavior-Knowledge Space (BKS) method for the combination of multiple classifiers providing abstract level information. The BKS is a K-dimensional space which is constructed during a learning phase. This method does not assume classifier independence but suffers from high storage requirements. The storage problems can be alleviated with dynamic allocation schemes. Huge training sets are needed to take full advantage of the method.

- Lam and Suen [56, 55] conducted in-depth study of the simple majority vote and claimed it to be as effective as more complicated schemes in improving the recognition results. Particular attention was directed toward the changes of performance when classifiers are added.

- Ho [37, 100] presented a theory of decision combination scheme based on ranking of classes by each classifier. The ranks are comparable across different types of classifiers and can be combined by methods which either reduce or rerank a given set of classes.

- Woods et al. [106] presented a method using local accuracy estimates. The combination method uses estimates of each individual classifier's local accuracy in small regions of feature space surrounding an unknown test sample. Synthesized data sets are used for the experiments. Comparison was done with other four combination schemes. No real world data result is reported.

- Cho and Kim [13] experimented with network fusion using fuzzy integral. The fuzzy integral is a nonlinear function that is defined with respect to a fuzzy measure. The method nonlinearly combines objective evidence, in the form of a network output, with subjective evaluation of the importance of the individual neural networks.

- Lee and Srihari [61, 59] presented a combination scheme based on neural network approach. A scheme is presented to transform classifier outputs into vectors of values in the [0,1] range. These vectors have the same dimension as the number of classes. They are used as input to a multi-layer perceptron without a hidden layer which is named Decision Combination Neural Network (DCNN). By integrating a dynamic selection network, DCNN evolves into a Dynamic Selection Combination Network (DSCN). The main concern of this approach is network complexity. Some discussions are then made about the ability to eliminate redundant classifiers and the complexity control mechanism.

Some interesting results of combination scheme are also reported in practical applications:

- Suen et al. [96] described one of the earliest successful multi-expert system for handwritten numeral recognition. They presented the combination of four structural methods using variants of majority voting. High performance was reported on CENPARMI database.

- Xu et al. [107] used the same four experts and results are provided using Dempster-Shafter formalism and Bayesian formalism.

- Huang and Suen [44] also provided results on the same experts using Behaviour-Knowledge Space method. Because of insufficient learning data, the results were obtained by a mixed BKS-Bayesian method rather than pure BKS method.

- Combination classifiers in [100] was tested on degraded machine-printed characters and word recognition.

- Lam and Suen [55] applied weighted voting to handwritten numeral recognition using seven classifiers.

In Section 1.3.3, we will list the performances of the most reliable systems in the field of handwritten numeral recognition with detailed discussions. We can find that many up-to-date reliable systems for handwritten recognition actually contain some kind of embedded combination scheme.

1.3.2 Numeral String Recognition

Numeral String Recognition can be categorized as discrete approach and holistic approach. The philosophy shares some resemblance with word recognition. However, unlike word recognition, no contextual information is contained in a numeral string. There is usually no hypothesis of the length or next numeral identity in a string.

Discrete Approach

To recognize a numeral string, we try to decompose a numeral string into subimages of individual numerals. However, to successfully isolate the numerals from a string is not always an easy job, especially when the numerals overlap or touch each other. A survey of segmentation strategies is provided by Casey and Lecolinet [7]. Next we list the commonly used approaches for the recognition of handwritten numeral strings.

Discrete approach can be purely segmentation-based which depends on the segmentation method to get the correct subimages, or recognition-based which needs the confidence of a recognizer to help ensure segmentation. The latter is also mentioned in the literature as segmentation-free approach. Segmentation-based method requires a carefully designed segmentation strategy to get the correct subimages. Segmentation-free approach, while remedies this problem, usually results in computational complexity. So combinations of these two approaches are also reported.

- The simplest segmentation strategy is based on vertical projections. The projections are running counts of the black pixels in each column. By detecting the minimum of the absolute running value or the ratio of second derivative of the projection curve [2], the segmentation line candidate can be obtained. The obvious limit of this approach is that its weakness with poor quality handwritten numeral image, such as slanting and overlapping numerals.

- The second segmentation strategy uses structural features. Westall et al. [103] used vertex directed segmentation. The method is designed to segment touching numerals by identifying directly the point of connection through an analysis of the vertices of vertically oriented edges of the strokes. Yu and Yan [108] proposed a method of segmenting single touching numerals using contour features. Shi [89] detected potential segmentation regions by analysis of the trajectory of strokes. They are other variants of structural feature based segmentation. Cheriet et al. [11] detect the splitting paths by background analysis to extract the face-up and face-down valleys, strokes and loop regions of component image. A "marriage score matrix" is then used to determine which pair of valleys is the most appropriate.

- The third segmentation strategy for numeral string is element based. The method is called graph-based approach [23]. The segmentor constructs a graph of the input image based on handwriting elements, each of which can be transformed into both neighboring elements. An estimation of a match is then evaluated between the input graph and the prototype graph. This method follows the concept of recognition-free approach in the sense that the breaks are at the elementary stroke level, but it is less time-consuming since a thorough analysis of the segmentation variants is done first.

- The segmentation-free strategy can be based on sliding windows or Hidden Markov Models (HMM) [77]. The Markov model is based on state-to-state transitions within a character, which is different from those used in isolated character recognition that represents letter-to-letter variations. These transitions provide a sequence of observations on the numeral. Features are typically measured in the left-to-right direction. Segmentation is *implicitly* done when matching the model against a given sequence of feature values.

- The combination of segmentation-based and segmentation-free strategy is exemplified in Ha et al [31] which used a presegmentation module that divides the input numeral string into independent groups of digits. The touching numerals are then processed by a segmentation-free method which recognizes elementary strokes by a statistical recognizer.

Another example of segmentation-based method with recognizer assistance for touching numerals is presented by Strathy et al. [92]. They detect splitting paths by significant contour points (SCPs) method and seek for confidence of a recognizer. This method can handle single touching and multiple touching problems with reasonable computation cost. The details will be given in Chapter 5.

Holistic Approach

Holistic approach recognizes the whole string as a unit. Although the problem of segmentation can be avoided, a major limit of this method is that it usually works on a small and predefined lexicon which contains fixed number of possible strings .

One holistic approach is based on global structural features. Wang [102] used gradient, structrual and concavity (GSC) features to recognize touching numeral pairs as a whole. The features are symbolic and multi-resolutional, extracted from 4^*4 grid of subimage cells. A weighted summation of these GSC features is used for classification.

Hidden Markov Model (HMM) can also be used as a holistic handwritten numeral string recognizer. The model represents the state-to-state variations within a specific string, which is different from the models dedicated to isolated numerals or segmentation-free methods. It does not need either explicit or implicit segmentation. This approach is more commonly used in word recognition than in numeral recognition. The reason may be explained by the little contextual information within a numeral string.

1.3.3 Performance of the Current Systems

In this section, we summarize the best published performances of the systems in this field.

The results of isolated numeral recognition are listed in Tables 1 and 2, distinguished by the size of the test database. The results of numeral string recognition on databases are listed in Table 3. Table 4 presents performances of practical applications on numeral string recognition.

Table 1 gives the results on CENPARMI and CEDAR databases for isolated handwritten numerals. (Descriptions of databases that we mention in the thesis or established/used during our work are given in Appendix C.)

Cho [14] uses 3 combination methods based on 3 neural network classifiers. The first classifier is a fully connected one-hidden layer perceptron. The images are size-normalized to 16*16 then compressed to 4*4 feature vectors as input of the network. The second classifier has the same architecture and uses Kirsh masks to extract directional features, the input feature vector also has the size of 4*4. The third neural classifier uses 15 Fourier descriptors from the outer contours and simple topological features from the inner contours. The three combinations methods are majority voting, averaging of the separate networks and fuzzy integral. Among all the schemes, the best performance is 96.05% recognition rate and 3.95% substitution on CENPARMI database.

Lee [62] also uses a neural approach. He uses the similar feature vector as [14] based on Kirsh masks. The input layer contains 4*4*4 local feature maps. The hidden layer consists of five 4*4 feature maps. Genetic algorithm is used to optimize the initial weights of the networks. A reliability of 97.10% is reported on CENPARMI database.

AEG-CENPARMI [25] is a collaboration work of AEG researcher and CENPARMI. Good result is also reported based on CENPARMI database. Six experts are combined, among which E1-E4 are described in [96]. A reliability of 98.50% was achieved.

In [95] the handwritten numeral recognizer is a combination of three similar standard backpropagation networks. Network A uses the pixel distance features while network B and C use the size-normalized image pixels. The combined decision is obtained from the sum of outputs. By training on very big datasets and years of intensive work, the algorithm achieved a high reliability of 98.85%.

Suen et al. [96] described one of the most successful attempts to achieve high reliability. They combine four different experts developed by CENPARMI researchers and get the final conclusion using the consensus of the methods. Substitutions on CENPARMI database are avoided completely while a recognition rate above 90% is retained.

CEDAR database is another database commonly used to report test results of recognition algorithms for isolated handwritten numerals. Lee [60] presented the results of seven different classifiers and five combination methods. The combination methods are Bayesian, neural approach, logistic regression, fuzzy integral and majority voting. Fuzzy integral gives a performance of 98.87% on CEDAR dataset of isolated handwritten numerals.

Ha et al. [31] applied a perturbation method on handwritten numeral recognition. It is an approach that considers variety of distortions due to eccentric handwriting. 12 perturbation types are discussed including rotation, slant, perspective and shrinking in different directions. Experiments show a reliability of around 99% on CEDAR goodbs dataset.

Parascript [24] presents a system for handwritten postal address recognition named $AddressScript^{TM}$. The embedded handwritten numeral recognizer constructs a graph of the input image based on handwriting elements, each of which can transform into both neighboring elements. Two neural networks are responsible for numeral classification. The algorithm was tested on CEDAR goodbs database (a set of 2,213 well segmented samples extracted from CEDAR CDROM *bs* directory) and shows very high reliability.

Table 2 lists the result on NIST database for isolated handwritten numerals.

Researcher/System	Recognized	Substituted	Reliability	Database	Date
S. B. Cho [14]	96.05%	3.95%	96.05%	CENPARMI	1997
S. W. Lee [62]	97.10%	2.90%	97.10%	CENPARMI	1996
AEG-CENPARMI [25]	98.50%	1.50%	98.50%	CENPARMI	1993
CENPARMI [95]	98.85%	1.05%	98.85%	CENPARMI	1998
CENPARMI [96]	93.05%	0.00%	100.00%	CENPARMI	1992
D. S. Lee [60]	98.87%	1.13%	98.87%	CEDAR	1993
T. M. Ha [31]	99.09%	0.91%	99.09%	CEDAR	1997
Parascript [24]	99.54%	0.46%	99.54%	CEDAR	1998

Table 1: Some Reliable Systems for Isolated Handwritten Numeral Recognition on CENPARMI and CEDAR databases

NIST database for isolated handwritten numerals was originally designed for a competition on First Census Optical Character Recognition Systems Conference in May 1992 organized by the National Institute for Standards and Technology (U.S.). The event was for assessing the state of the art in OCR. Twenty nine groups from North America and Europe responded the call and twenty six finally attended the conference with their test result on NIST Test Data 1 (TD1) which included 58,646 numerals. The training set provided by NIST is called NIST Special Database 3 (SD3) with 223,122 numerals. About half of the systems recognized more than 95% of the test samples. The results of IBM, AEG and AT&T are listed in Table 2. Corresponding recognition rates when they try to obtain very low substitution rates are also listed. IBM Almaden research center used a neural approach. AEG used feature vector from KL-transform after preprocessing of the numeral image. AT&T used gray level image and a k-nearest neighbor classifier.

The NIST datasets are then used by other researchers to train and report algorithms for handwritten numerals. T.M. Ha [31] achieved a 97.10% reliability with the algorithm described above. Hewlett Packard [50] reported a 98.60% reliability using improved LDA (Learning by Discriminant Analysis) method.

NIST datasets are also used at our CENPARMI laboratory due to their large quantity of samples. With the similar methods reported upon CENPARMI database in Table 1, high reliability has been achieved recently, which is the highest reported in literature.

Methods	Recognized(%)	Substituted(%)	Reliability(%)	Date
IBM [104]	96.51	3.49	96.51	1992
IBM [104]	50.00	0.1	99.80	1992
AEG [104]	96.57	3.43	96.57	1992
AEG [104]	50.00	0.1	99.80	1992
AT&T [104]	96.84	3.16	96.84	1992
AT&T [104]	68.00	0.1	99.85	1992
T.M. Ha [31]	97.10	2.90	97.10	1997
Hewlett Packard [50]	98.60	1.40	98.60	1996
CENPARMI [95]	99.07	0.93	99.07	1998
CENPARMI [95]	94.45	0.1	99.89	1998

Table 2: Some Reliable Systems for Isolated Handwritten Numeral Recognition on NIST TD1 database

Table 3 lists the results of numeral string recognition as reported in the literature. There is no standard public database yet for numeral string recognition. The data sets often used in literature are from the zip codes database developed at CEDAR.

Researchers from CEDAR reported results with data under different directories. Shi et al. [89] used a segmentation based approach and achieved 80.8% recognition rate on numeral pairs under BU directory. Lower rates are reported from images of other directories. Wang et al. [102] used a holistic approach for numeral pairs and obtained 83.6% recognition rate as well as reliability, based on 2778 images from CEDAR CD-ROM. Ha et al. [32] uses the same source of data and reported 83.6% recognition rate on numeral string. No exact directory or amount of images was indicated.

Up-to-date practical system performances are listed in Table 4. Performance of Lucent system for cheque processing is obtained from advertisement. Parascript [20] used the cross validation of courtesy amount and legal amount fields after recognizing numeral strings on cheques. Parascript [24] reported the performance of the system *AddressScript*^{TM} when recognizing 5-digit zip code, with the method already described above in this section. The detailed modules of CENPARMI cheque reading system were described in [95] and [93]. The cross-validation of courtesy amount and legal amount is also part of the post-processing module.

Researcher	Recognized	Substituted	Reliability	Database	Date
Z. Shi [89]	80.8%	19.2%	80.8%	CEDAR (numeral pair)	1997
X. Wang [102]	85.1%	14.9%	85.1%	CEDAR (numeral pair)	1998
T. M. Ha [32]	83.6%	16.4%	83.6%	CEDAR	1998

Table 3: Performance of Numeral String Recognition based on Data in CEDAR CDROM

System	Recognized	Substituted	Application Domain	Date
Lucent (ad.)	44.0%	1.0%	cheques	1998
Parascript [20]	47.0%	1.0%	cheques	1997
Parascript [24]	64.0%	0.8%	5-digit zip codes	1998
Mitek [40]	55.0%	1.0%	cheques	1996
CENPARMI [95]	62.0%	1.0%	cheques and financial documents	1998

Table 4: Performance of Application Systems on Numeral String Recognition

We can summarize several key aspects of performances of the state- of-the-art systems.

Researchers are getting good results on relatively small databases of isolated handwritten numerals. For example, after years of intensive work, our group has achieved high reliability on CENPARMI database [95]. Good results are also reported on CEDAR *goodbs* database. Test sets of the two databases contain 2000 and 2213 digits respectively.

When the size of test database increases such as NIST TD1 which contains 58,646 images of digits, performance drops as represented by low reliabilities. To keep a high reliability is synonymous with a big sacrifice of recognition rate, which is obviously not ideal.

When the topic comes to numeral string recognition, performance figures go dramatically down to 80% level due to complicated touching cases and more diversified writing styles. Ha et al. [32] recently reported a recognition rate of 92.7% based on NIST. However, the result was attained on training set SD3 instead of test set and there was no description given in the paper about their methods of extracting the strings from NIST CDROM.

Practical systems usually report relatively low substitution rates which depend on other sources such as city names in postal codes and legal amount in check processing, but they give even lower correct rates with the introduction of problems occurred during other stages (such as item extraction) and stringent demand of reliability.

1.4 Verification and Our Challenges

As we see from the analysis of system performances in section 1.3.3, although some current methods work well in certain situations, few has achieved satisfactory performance in situations demanding very high reliability, such as financial document processing.

Consequently, a special engine geared *precisely* toward high reliability while maintaining reasonable recognition rate is in need. It is called Verifier and it forms the theoretical backbone of this thesis.

Verifier is not a newcomer in a sense that the concept has been adopted in other related domains. Verifier is a newcomer in a sense that its great potential for improving the reliability of handwritten recognition has never been addressed in literature. Its theoretical requirements and design strategy to explore this potential have also never been conducted.

The concept of the verifier has been used in *utterance verification* and *speaker verification* in the domain of speech recognition. Utterance verification [98, 82] is to verify whether or not the expected keywords are embedded in a spoken phrase or a sentence, while speaker verification distinguishes the speaker [10]. Signature verification is another well known field which deals with verification of the validity of a signature. With the possibility of getting some hints from these fields, verification problem in handwritten recognition has its distinguishing characteristics. Firstly handwritten recognition has totally different pattern properties and feature sets. Secondly the goal is to get high reliability of a complete recognition system, which implies dealing with a multi-class problem instead of a yes-no question.

The precise definition of a pattern verifier will be given in Chapter 2. Loosely speaking, the verifier is a specially trained expert to confirm or negate a classification result from a general purpose recognizer, so as to improve the system reliability. Up to now, verification in the domain of handwritten recognition has gained little attention and effort of researchers.

In Table 5, we list recent papers concerning the concept or usage of a verifier in OCR and their contributions. Takahashi [99] is among the earliest that mention the concept of verification in a document analysis domain. Chiang [12] did some work on verification of Chinese characters. Lee [51] incorporates a verifier into the system for handwritten numerals. However, it was not until our preliminary work in Zhou et al. [112] that the importance and functionality of a verifier is analysed and consciously emphasized.

We consider verifier as an important stage to improve the reliability of the recognition system by two rationales:

1. Whatever feature extracting method we use, we may lose some other features and may cause errors which are not understandable to human. In the case of a practical system, if the user cannot understand the reason which causes the error, the system tends to be turned down. Thus a reliable confirmation or rejection scheme,

ideally reflecting human referee view point, becomes essential to the success of a practical system.

2. Instead of being general OCR products, current systems applications tend to have a relatively fixed working environment, in which case, there are often extra relations among the recognized data. These rules can be applied in the verifier.

Correspondingly, there could be two levels of verifiers:

- low level: make use of internal knowledge of the recognition system.
- high level: cross check by user-specified rules.

Low level verifier is the topic of this thesis and will always be meant when using notion of a *verifier*. However, we also point out that making full use of the inherent high-level relations between recognized data is an important task when developing the practical system.

Author	Source	Application Background	Contribution
Takahashi[99]	ICDAR93	English alphabet	Early work on the subject.
Chiang[12]	ICPR96	Chinese Characters	Features of Chinese characters for verification purpose are analysed.
Lee[51]	ICDAR97	Handwritten Numerals	Use a verification engine in the system.
Zhou[112]	ICDAR97	Handwritten Numerals	Explicitly incorporate a verification module. Importance and functionality are analysed.

Table 5: Literature related to Verifier in OCR Domain

Conventionally, some classifiers unconsciously made efforts to fulfill the role of a preliminary verifier. One example is the feedforward artificial neural network applying a thresholding criterion to the difference of two largest output values, to draw the conclusion and expedite rejections. However, as we will see in Chapter 4, conventional neural networks are not adequate enough in a verification task, as reflected by the inability in ensuring the fairness of thresholding. Our work thus includes exploring the theoretical requirement of good verifiers and related design strategies.

Since only little advanced work on the verifier had been conducted in the research domain of handwriting recognition, the pioneer work on verification theory and its application in handwritten numeral recognition (especially on touching numeral recognition), combined with the goal to build highly reliable recognition system, constitute the challenges of our work.

1.5 Research Goals

Research goals of this thesis are twofold: theory and application. The theoretical part is focused on the original work about the verifier, while the application part deals with the practical problems of handwritten numeral recognition.

1.5.1 Theory

Developing theory of the verifier in the field of handwritten character recognition is a challenge. We will start with a definition. The definition and evaluation of verifier are based on the measurement of *Precision Rate*, which is a *a posteriori* point of view to look at the reliability of the system. It is aimed at ensuring the accuracy of a conclusion, thus perfectly matches the goal of a verifier. A significant improvement of precision rate can justify the effectiveness of a verifier, which will be evaluated by a statistical hypothesis testing.

Then we address several other issues:

1. To analyse the role of a verifier in a system from a theoretical point of view.

2. To evaluate a verifier and explore the inherent requirement of a good verifier.

3. To design the verifier from quantitative and qualitative points of views.

4. To control the power of verification in a desired way.

The discussions and results of these topics are scattered over Chapter 2, Chapter 4 and Chapter 6.

Theoretically speaking, we expect a good verifier possessing an inherent advantage in detecting the fuzziness along the decision boundary. This property is desired in verifying a confusing sample and reaching a reliable conclusion of confirmation or negation. A novel approach of Quantum Neural Network will be used to exemplify this property. Discussion of qualitative and domain-specific verification is done in Chapter 6.

We suggest a class-specific way of designing verifiers to maximize the ability of verification. The detailed architecture of the verifier is usually problem related. The verifiers proposed in this thesis for isolated handwritten numerals and touching handwritten numerals can be viewed as good examples of effective verifiers. In our verifiers, the power of verification is controlled by the inner layer components.

1.5.2 Applications

The primary goal in application aspect of our research is to develop Recognition and Verification (R&V) systems for the problems of two domains: Unconstrained Isolated Handwritten Numerals (UIHN) and Unconstrained Touching Handwritten Numerals (UTHN), to get reasonably high reliability for these two practical problems and to justify the effectiveness of verifiers based on the proposed systems. To accomplish this goal, efforts are geared towards the following:

1. To design and implement general purpose recognizer for UIHN.

2. To design and implement verifier for UIHN.

3. To incorporate recognizer and verifier into R&V system for UIHN, test the performance on standard databases. Evaluate the effectiveness of verifier for UIHN.

4. To design and implement general purpose recognizer for UTHN.

5. To design and implement verifier for UTHN.

6. To incorporate recognizer and verifier into a R&V system for UTHN, test the performance on databases. Evaluate the effectiveness of the verifier for UTHN.

Standard databases are important for building and evaluating systems. There exist standard databases for UIHN problem. However, we suggest that more thorough analyses of the properties of the databases be carried out as they may be very helpful for understanding the behaviour of the applications. At the current stage of UTHN problem research, lack of benchmarking databases is a big obstacle. Our efforts thus also extend to building good and standard databases for touching numerals, with the intention to benefit other researchers in this field. The design and building process of databases is described in Chapter 5.

1.6 Outline of the Thesis

The first chapter outlined the challenges and goals. A review of the state of the art and system performances has also been given. We emphasize the importance of a verifier in improving the reliability of practical systems.

In Chapter 2, we present the definition of the verifier, the evaluation method and the architecture of a general R&V system.

Chapter 3 presents the extensive work on General Purpose Recognizer (GPR) for UIHN. Several recognizers are implemented and compared. A final GPR for UIHN is obtained by a combination scheme. Chapter 4 explores the design of a verifier for UIHN. Theoretical requirements are discussed. Experiments on synthetic and real world data are reported. The architecture of the verifier is proposed. Its effectiveness is evaluated.

Chapters 5 and 6 focus on the problem of UTHN. Chapter 5 provides the scheme of GPR while the following chapter presents the verifier. The problem background of financial documents is discussed briefly. The building of two databases IRIS-Bell'98 and NIST for UTHN are described in detail. Properties of the two databases are also studied. Databases are used to test final performance and to evaluate the verifier.

The thesis is concluded with Chapter 7 which summaries verifier theory and main thesis contributions. Avenues of future work are also outlined.

Appendices contain descriptions of databases used in this work, notations and abbreviations, as well as some proofs.

Chapter 2

Verifier and System Architecture

"Who am I, then? Tell me that first, and then, if I like being that person, I'll come up: if not, I'll stay down here till I'm somebody else."
— Lewis Caroll, Alice's Adventures in Wonderland

2.1 Definitions

The definition of pattern verifier is based on its quantitative measurements aiming at the goal of improving reliability. Precision Rate (PR) and Recall Rate (RR) as defined below will be used in the thesis since they represent the functionality of a verifier much more clearly than conventional correct rate, as we will explain.

Definition 2.1 Precision Rate (PR): *Consider pattern class C_m. Precision rate of class C_m is the measurement of how well ONLY the input patterns of class C_m are correctly classified.*

Consider subset S_m of the input patterns as the set of samples belonging to C_m. Let K_m be the number of input patterns classified as class C_m

by the system. Within K_m, there are X_m patterns *truly* belonging to S_m. The precision rate of C_m is then given by

$$PR_{C_m} = X_m / K_m \tag{1}$$

The precision rate of the system is given by

$$PR_{sys} = \frac{1}{n_0} \sum_{m=1}^{n_0} PR_{C_m} \tag{2}$$

where n_0 is the total number of classes.

Definition 2.2 Recall Rate (RR): *Consider pattern class C_m. Recall rate of class C_m is the measurement of how well ALL the input patterns of class C_m are correctly classified.*

Consider subset S_m of the input patterns as the set of samples belonging to C_m. Let K_m be the number of input patterns classified as class C_m by the system. Within K_m, there are X_m patterns *truly* belonging to S_m. The recall rate of C_m is then given by

$$RR_{C_m} = X_m / |S_m| \tag{3}$$

where $|S_m|$ is the cardinality of S_m.

The recall rate of the system is given by

$$RR_{sys} = \frac{1}{n_0} \sum_{m=1}^{n_0} RR_{C_m} \tag{4}$$

where n_0 is the number of total classes.

Definition 2.3 Pattern Verifier: *A pattern verifier is a specially trained expert to confirm or negate a preset pattern class, usually used after a general purpose recognizer, with the intention to significantly improve class-specific precision rates of the system.*

We can see from the Definition 2.3 that a pattern verifier focuses on PR, i.e., it emphasizes on getting the correct pattern, even to the extent that it skips some relevant patterns. The objective originates from the stringent demand of system reliability. We will also point out that PR is a different measurement than conventional correct rate and it is more suitable and effective for the verifier. Below we explain this from a statistical point of view.

Let v be the input sample and let r_v denote the classification result of the system for v. PR_{C_m} reflects the *a posteriori* probability of the system performance with respect to class C_m:

$$PR_{C_m} = P(v \in C_m | r_v = m). \tag{5}$$

With Bayes rule,

$$\begin{aligned} PR_{C_m} &= \frac{P(v \in C_m, r_v = m)}{P(r_v = m)} \\ &= \frac{P(v \in C_m, r_v = m)}{\sum_{i=1}^{n_o} P(r_v = m | v \in C_i) P(v \in C_i)} \end{aligned} \tag{6}$$

Examine the denominator of Equation (6). $P(v \in C_i)$ can be considered as constants assuming every numeral class has the same occurring probablity. In $\sum_{i=1}^{n_o} P(r_v = m | v \in C_i)$, $P(r_v = m | v \in C_m)$ (i=m) is the conventional "recognition rate", i.e., unlike PR, the conventional recognition rate ignores the effect of the remaining items:

$$M = \sum_{i \neq m} P(r_v = m | v \in C_i). \tag{7}$$

M represents the chance of mistakenly recognizing other samples as those of class C_m. Only when M is small, we can get a high precision rate of C_m, so as to ensure a lower error rate and a high reliability of the system.

We present an example in Table 6 to illustrate the computation of precision rate and recall rate with reference to the conventional recognition rate.

Consider recognition of class A in a test set. Assume there are 50 samples from it within 100 samples. We list two situations and the corresponding precision rate, recall rate and recognition rate.

In situation I, no $\bar{A}$ is recognized as A. In situation II, 3 $\bar{A}$ are recognized as A. We can see that although the two situations give the same recognition rate of class A (90%), situation I is more desirable due to its high precision rate which leads to a higher overall system reliability.

	Samples	Recognized as A	Precision Rate of A	Recall Rate of A	Recog. Rate of A
I	50 A	45	100% (45/45)	90% (45/50)	90% (45/50)
	50 $\bar{A}$	0			
II	50 A	45	93.7% (45/48)	90% (45/50)	90% (45/50)
	50 $\bar{A}$	3			

Table 6: Example for Computing Precision Rate and Recall Rate

Precision rate reflects the system reliability. It provides the definition in a class-specific matter, which distingishes itself from reliability and represents the characteristics of verifiers.

2.2 Statistical Evaluation of Verifier

According to Definition 2.3, the verifier aims to *significantly improve* the precision rates of the system. Evaluating the effectiveness of a verifier thus becomes a question of whether the adoption of the verifier has introduced significant performance difference to the system, which falls into the scope of a statistical inference method called "hypothesis testing" [6, 68].

Hypothesis testing is a statistical method to examine a population parameter or to examine the relationship of two populations. To see how it will fit into our problem of evaluating verifiers, we will start from the definitions of statistical hypothesis and complementary hypotheses in hypothesis testing.

Definition 2.4 Hypothesis: *A hypothesis is a statement about a population parameter.*

Definition 2.5 Complementary Hypotheses: *The two complementary hypotheses in a hypothesis testing problem are called the* null hypothesis *and the* alternative hypothesis. *They are denoted by H_0 and H_1, respectively.*

If θ denotes a population parameter, the general format of the null and alternative hypothesis is $H_0 : \theta \in \Theta_0$ and $H_1 : \theta \in \Theta_0^c$ where Θ_0 is a subset of the parameter space and Θ_0^c is its complement.

For example, if θ denotes the average change in a patient's blood pressure after taking a drug, an experimenter might be interested in testing $H_0 : \theta = 0$ versus $H_1 : \theta \neq 0$. The null hypothesis states that, on the average, the drug has no effect on blood pressure and the alternative hypothesis states that there is on effect. This common situation, in which H_0 states that a treatment has no effect, has led to the term "null" hypothesis. One reason for the negative aspect of this name is that it is common practice to set up such a hypothesis in the hope of rejecting it.

In our problem of evaluating the verifier, we can set two complementary hypotheses as:

H_0: Systems with or without verifier show no significant difference in precision rates;

H_1: Systems with verifier show a significant improvement in precision rates.

We can then test the hypotheses by observing the samples and decide either to accept H_0 as true or to reject H_0 as false and decide H_1 as true.

Definition 2.6 Hypothesis Test: *A* hypothesis testing procedure *or* hypothesis test *is a rule that decides:*

i For which sample values the decision is made to accept H_0 as true.

ii For which sample values H_0 is rejected and H_1 is accepted as true.

Definition 2.7 Critical Region: *The subset of the sample space for which H_0 will be rejected is called the critical region.*

Definition 2.8 Level of Significance: *The probability of rejecting a true hypothesis (type I error), denoted by α, is called the level of significance of the test.*

The meaning of the critical region is illustrated in Figure 1. If level of significance is $\alpha = 0.01$, then one percent of our possible sample will have means which fall in the critical region, provided that H_0 is true. The critical region and significance level are set according to the seriousness of making a type I error. The more serious the type I error could be, the smaller α will be set.

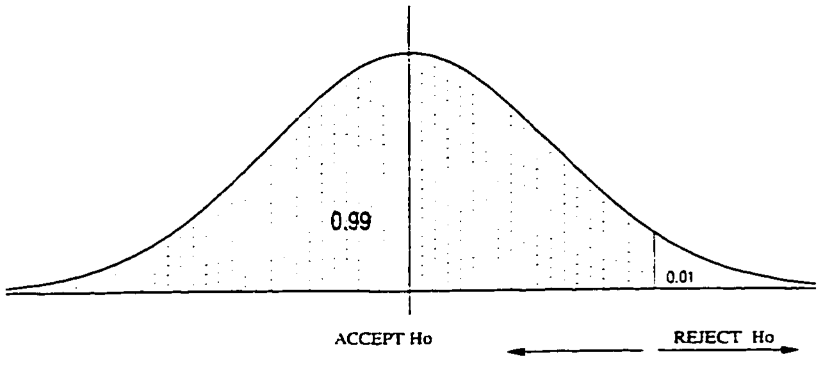


Figure 1: Critical Region of Hypothesis Testing

Hypothesis testing is a well-developed method rich in theoretical basis. It inherently takes many factors into consideration such as test set size and sample variations to come up with a much more confident conclusion compared with a simple two percentage figure comparison.

The procedure to solve a practical hypothesis testing problem is listed in Table 7.

In our problem of testing the effectiveness of a verifier, we will use $t - test$, α will be set as 0.01. Let A be the random variable that represents the precision rates getting from a system without verification module, let B represent the precision rates from a system with verification module. Experiments are done on randomly selected

Step 1:	Determine the null hypothesis H_0.
Step 2:	Select statistical variable and get its distribution.
Step 3:	Determine the corresponding test method, which can be $u-test$, $t-test$, $\chi^2 - test$ or $F - test$.
Step 4:	Determine the critical region of accepting H_0.
Step 5:	Compute the result of the test.
Step 6:	Compare results of Step 4 and Step 5, decide to accept H_0 or reject H_0.

Table 7: Procedure of Hypothesis Testing

sets of numbers n_1 and n_2 to get distributions. The test procedures are described below:

Step 1: Set H_0: Systems with or without verifier has no significant difference on precision rates. H_1: Systems with verifier have significant improvement on precision rates.

Step 2: Let $N(\mu_1, \sigma_1^2)$ and $N(\mu_2, \sigma_2^2)$ be the distributions of A and B respectively. Assume $\sigma_1^2 = \sigma_2^2$ (value unknown). The populations are assumed to be independent. Thus $H_0 : \mu_1 = \mu_2$. $H_1 : \mu_1 > \mu_2$.

Step 3: Use $t - test$. The appropriate statistic is,

$$\frac{(\overline{X} - \overline{Y}) - (\mu_1 - \mu_2)}{S(\frac{1}{n_1} + \frac{1}{n_2})^{1/2}} : t_{n_1+n_2-2}$$

where

$$S^2 = \frac{n_1 S_1^2 + n_2 S_2^2}{n_1 + n_2 - 2}$$

S_1 and S_2 are sample standard deviations of A & B, n_1 and n_2 are the numbers of samples of A & B.

Step 4: Test the hypothesis at 0.01 significance level. $\alpha = 0.01$, $t_{n_1+n_2-2,\alpha} = t_{n_1+n_2-2,0.01}$.

Step 5: Compute $t_{n_1+n_2-2}$ according to experimental results.

Step 6: We will reject H_0 if $t_{n_1+n_2-2} > t_{n_1+n_2-2,0.01}$; otherwise, we will accept H_0.

Based on the above procedures, Chapters 4 and 6 will give the evaluations of our verifiers according to the test results of the systems for UIHN and UTHN respectively.

The theoretical basis of hypothesis testing is the *Central Limit Theorem* and *Neyman_Pearson Lemma* [80]. An exposition of these theories and different test methods in their full generality would lead us rather far afield, and we would refer interested readers to the literature.

2.3 Recognition and Verification System

2.3.1 Architecture

Following the idea of the previous sections, which takes high reliability as the ultimate goal and verification as the important tool, we propose a Recognition & Verification System for unconstrained handwritten numeral recognition depicted in Figure 2.

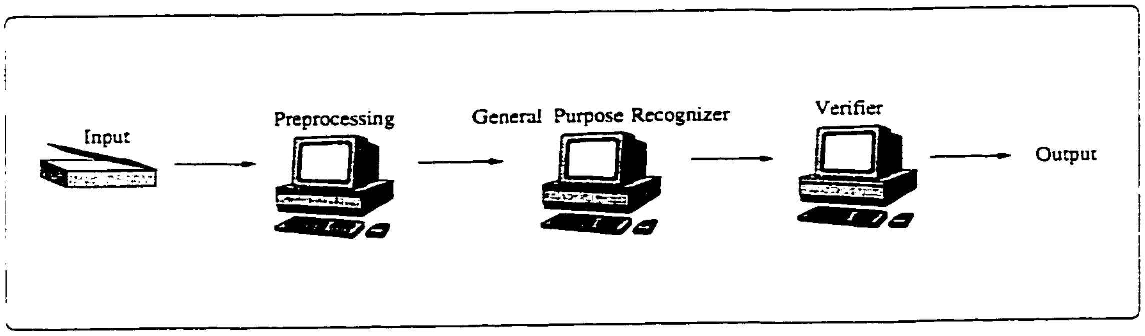


Figure 2: Diagram of the Recognition & Verification (R&V) System

The scheme looks straightforward, with a verification module embedded in the traditional classification system which only has a general purpose recognizer. Let us look at the rationale behind it.

We know that pattern classification is the task of assigning a class C_m to the input pattern v. It is equivalent to establishing a mapping of $V \rightarrow D$ from measure space V into the decision space D [88]. Since $V \rightarrow D$ is usually unknown, what we do is trying to recover it from sample sets. Let $P(C_m|v)$ denote the probability measure of $V \rightarrow D$ which states how probable the class C_m is after v has been observed. With our R&V system, the overall $P(C_m|v)$ is now expressed in the following way:

$$P_{R\&V}(C_m|v) = P_R(C_m|v) * P_V(C_m|v) \tag{8}$$

where $P_R(C_m|v)$ is the *a posteriori* measure of the general purpose recognizer. $P_V(C_m|v)$ is the probability that verifier will give a confirmatory answer.

We can see from Equation 8 that the final performance will now depend on two modules jointly. The recognizer is a general purpose classifier working fairly well for a broad class of problems, without special focus on reliabilities. The verifier will take the role of an expert to fairly and precisely evaluate the result of recognizer, to compensate its weakness by particular training, and to make the whole system more reliable.

2.3.2 Usage of a Verifier

The verifier is applied after a general purpose recognizer and designed to "plug and play", i.e., it is used without knowing the implementation details of the recognition modules. A verifier for Class C_m can give us three possible conclusions:

1. The class label m for sample v from the general purpose recognizer is confirmed. Conclusion: $v \in C_m$.

2. The class label m for sample v from the general purpose recognizer is negated. Conclusion: $v \notin C_m$, which may lead to a rejection of the input pattern.

3. There is not enough knowledge to give a precise verification. Conclusion: uncertainty, which may lead to a rejection of the input pattern.

A good verifier is required to give reliable confirmation and negation, while not causing many extra rejections. Using a verifier this way, errors are minimized and the reliability of the whole R&V system is expected to be highly enhanced.

2.3.3 Class-specific Verifier

There are four types of verifiers according to the number of classes they work on. Let Ω denote the working space of a verifier, and let $|\Omega|$ be the dimension of the space.

- $|\Omega| = n_0$: General verifier, working on all classes in the problem.
- $0 < |\Omega| < n_0$: Cluster verifier, verification in clustered categories (Is it a "0", "6", or "9"?).
- $|\Omega| = 2$: Pairwise verifier, verification between two categories (Is it a "0" or "6"?).
- $|\Omega| = 1$: Class-specific verifier. Working on one candidate class(Is it a "1"?).

Among different types of verifiers, class-specific verifier needs some extra attention. It basically divides the verification problem into n_0 subtasks, each verifier explicitly indicates a pre-known focus — the candidate class. The scheme has several special merits:

1. The underlying principle of "divide and conquer" makes it easier to get optimal solutions for subproblems.

   See an illustrative example of 9 classes in the sample space shown in Figure 3. Each class is clustered as a circle. If we want to develop a single shot discriminant function to distinguish all the classes, the decision space will be unavoidably complicated. On the other hand, if we want to separate one cluster from the others, a circle-shape discriminant function surrounding the candidate class will be an ideal solution(as shown in Figure 3). We also know that certain classifiers such as RBF neural network are good at this kind of functional approximation [47].

2. From software engineering point of view [74], the system can be separated into sub-modules and be developed without interference among others. For each module, optimal features and classification method will be exploited.

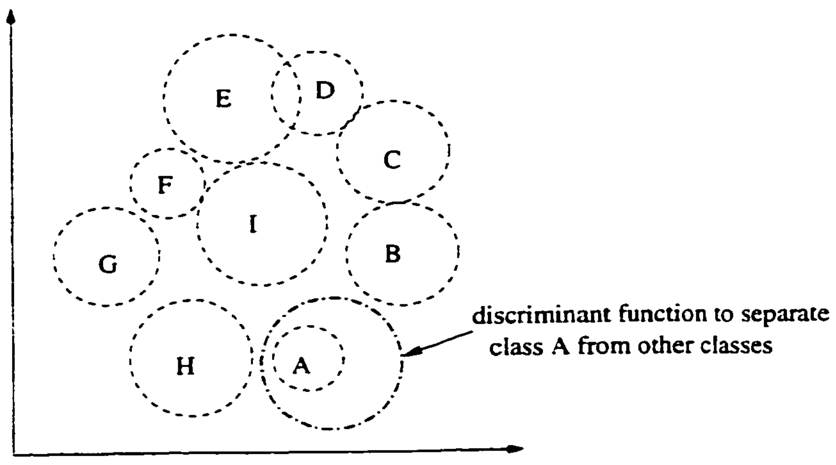

E
D
C
F
I
G
B
H
A
discriminant function to separate class A from other classes

Figure 3: Illustrative Example of Class-specific Solution

Adoption of class-specific verifiers in handwritten numeral classification problem results in system architecture having a "sun-planet" look, as in Figure 4.

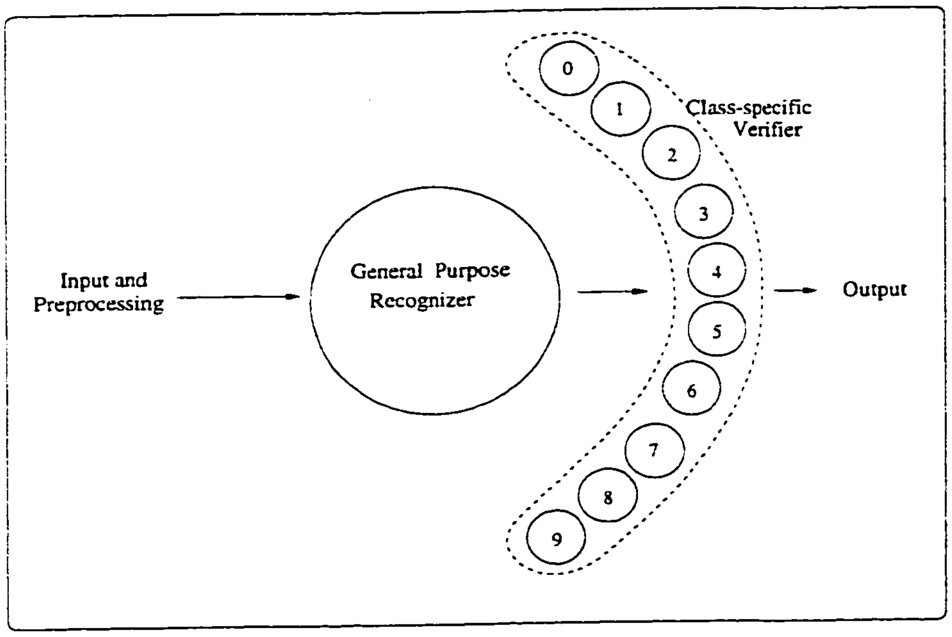


Figure 4: System Architecture with Class-specific Verifiers

Chapter 3

General Purpose Recognizer for Unconstrained Isolated Handwritten Numerals

"Now, I know that there are seventeen steps, because I have both seen and observed."

— Sherlock Holmes, A Scandal in Bohemia

Designing a general purpose recognizer for unconstrained isolated handwritten numerals is a topic studied by many researchers. To get a high performance recognition module in our R&V system, different types of classifiers are experimented with the focus on artificial neural networks and tree classifiers. Performances of four classifiers: Multilayer Perceptron (MLP), Convolutional Networks, Radial Basis Function (RBF) network and Tree Classifier C4.5 are discussed in this chapter. A general purpose recognizer combined from the discussed classifiers is given with good performance on standard databases. Precision rates are computed for further improvement and effectiveness test of verifiers.

3.1 Multilayer Perceptron

Work on artificial neural networks began more than 50 years ago with the efforts of McCulloch and Pitts [69], Hebb [35], Rosenblatt [85] et al.. More recent work by Hopfield [39], Rumelhart and McClelland [86], Feldman [22] and others has led new interests to the field, due to the development of new network topologies and algorithms, new analog VLSI implementation techniques, intriguing demonstrations in the fields of speech and image recognition as well as by a growing fascination with the functioning of the human brain. Good reviews of various artificial neural networks can be found in [65] and [47].

Among various neural net models, Multilayer Perceptron (MLP) is the most widely used, especially in the problem of pattern classification.

3.1.1 Network Topology

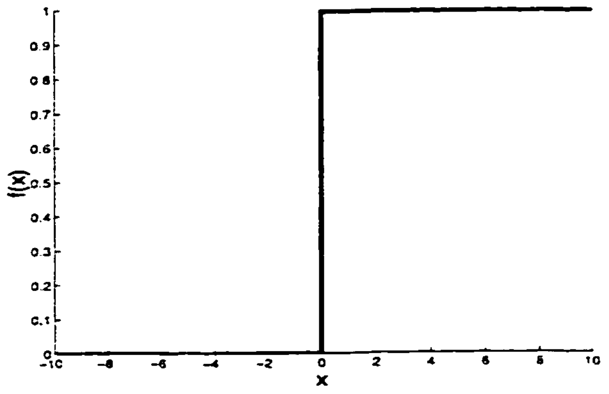
The perceptron was conceived by Rosenblatt in 1959 [85]. The perceptron, as the building block of MLP network, forms a weighted sum of n components of the input vector and adds a bias value, θ. The result is then passed through a nonlinearity (See Figure 5).

Rosenblatt's original model used the hard-limiting nonlinearity. When perceptrons are cascaded together in layers, it is more common to use the sigmoid nonlinearity:

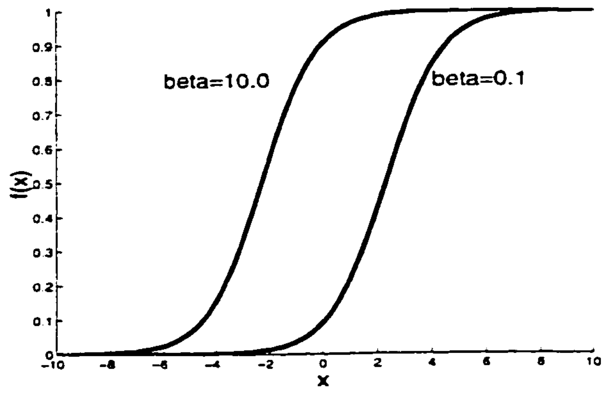
$$sig(x) = (1 + e^{-\beta x})^{-1} \tag{9}$$

where slope factor β determines the steepness of the transition region. One of the advantages of the sigmoid compared with hard-limiting is that it is differentiable, which makes it possible to derive a gradient search learning algorithm for the multilayer network.

By cascading perceptrons in layers, multilayer perceptron overcomes single perceptron's limitation of linear decision boundary and approximation of a simple logic



(a) Hard-limiting



(b) Sigmoid function

Figure 5: Activation Functions of Conventional Perceptron

function. The individual perceptrons in the network are called neurons or nodes. Nodes in adjacent layers are connected through links whose associated weights determine the contribution of nodes on one end to the overall activation of nodes on the other end.

There are generally three types of layers. Nodes in the input layer are connected to the elements in the input vector. The multi-nodes in the output layer typically represent different classes in the multi-class pattern recognition problem. An arbitrary number of hidden layers may be used depending on the desired complexity. A diagram of a standard MLP with one hidden layer is shown in Figure 6.

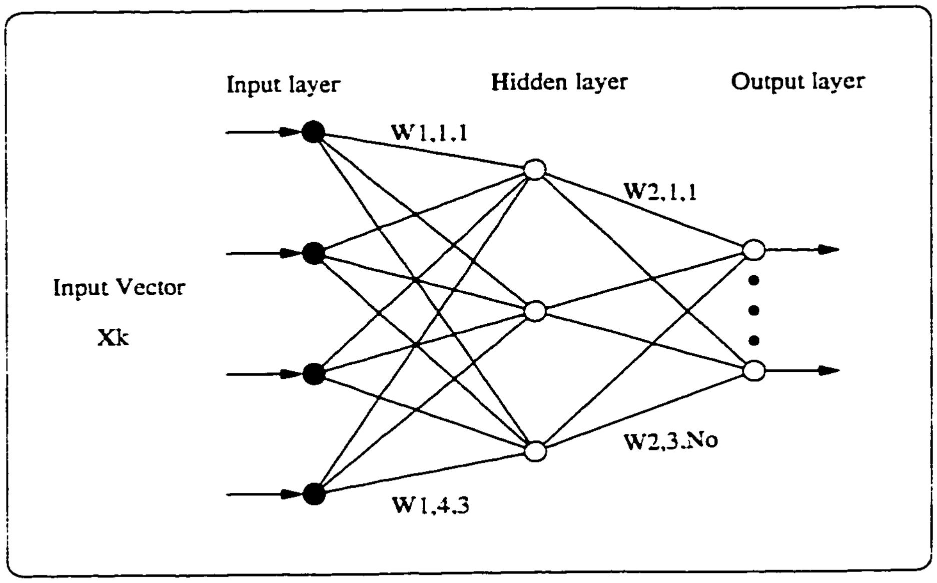


Figure 6: Diagram of One Hidden Layer MLP

3.1.2 Learning Algorithm

Conventionally, error back-propagation (BP) algorithm is used as the learning algorithm. The "Standard Backpropagation" is an euphemism for the generalized delta rule, the training algorithm that was popularized by Rumelhart, Hinton and William [86]. By extension, the term *BPNet* is also used to refer a feedforward MLP trained by backpropagation, which is adopted in this thesis.

With the notations as in Table 8, as well as the one hidden layer topology of Figure 6 as the illustrative example, we can derive the learning algorithm:

$w_{l,i,j}$	Connecting weight between *ith* node in l layer and *jth* node in l+1 layer
$d_{i,k}$	Desired output of ith output node for input sample x_k
$u_{i,k}$	Real output of ith output node for input sample x_k
$o_{j,k}$	Output of jth hidden layer node for input sample x_k
$x_{n,k}$	*nth* component for input sample x_k
P	Number of training patterns
n_h	Number of hidden layer nodes
n_i	Number of input layer nodes
n_o	Number of output layer nodes

Table 8: Notations Used in BP Training Algorithm

$$\begin{aligned} u_{i,k} &= sig(\sum_{j=0}^{n_h} w_{2,i,j} o_{j,k}) \\ &= sig(\sum_{j=0}^{n_h} w_{2,i,j} sig(\sum_{n=0}^{n_i} w_{1,j,n} x_{n,k})) \end{aligned} \tag{10}$$

A gradient search technique is used to minimize the mean-square error on training data:

$$E_k = \frac{1}{2}\sum_{i=0}^{n_O}(d_{i,k} - u_{i,k})^2 \tag{11}$$

$$E = \sum_{k=0}^{P} E_k \tag{12}$$

During the training process, the connecting weights are updated by gradient descent, where the derivatives with respect to the weights are computed as

$$\frac{\partial E_k}{\partial w_{2,i,j}} = \delta_i^{(2)} o_j \tag{13}$$

$$\delta_i^{(2)} = (u_k - d_k)u_k(1 - u_k) \tag{14}$$

$$\frac{\partial E_k}{\partial w_{1,j,n}} = \delta_j^{(1)} x_n \tag{15}$$

$$\delta_j^{(1)} = o_j(1 - o_j)\sum_{i=0}^{n_0} w_{2,i,j}\delta_i(2) \tag{16}$$

The weights are incremented towards the negative direction of gradient and momentum, η, may be used to improve the convergence:

$$\begin{cases} \Delta w_{2,i,j}(k+1) = -\eta\frac{\delta E_k}{\delta w_{2,i,j}} + \alpha\Delta w_{2,i,j}(k) \\ \Delta w_{1,j,n}(k+1) = -\eta\frac{\delta E_k}{\delta w_{1,j,n}} + \alpha\Delta w_{1,j,n}(k) \end{cases} \tag{17}$$

where η is the learning rate and α is the momentum rate. To guarantee the convergence, the learning rate should be small enough and may be decreased with time as we will show in the experiments.

3.1.3 Experiments

Applying BPNet as the 10-ary classifier starts our task to develop the general purpose recognizer for Unconstrained Isolated Handwritten Numerals (UIHN).

Experiments are based on two standard databases:

1. CENPARMI database of UIHN.

   This widely used standard database of UIHN is collected from the dead letter envelopes by the U.S. Postal Service at various locations in the U.S., with equal number of 600 binarized samples in each of 10 digit classes. 4000 digits are specified as training set, other 2000 digits are used for testing.

2. MNIST database of UIHN. MNIST is the acronym for Modified NIST. NIST originally designated SD-3 as their training set and TD-1 as their test set. However, SD-3 is much cleaner and easier to recognize than TD-1. The reason for

this can be attributed to the fact that SD-3 was collected from Census Bureau employees, while TD-1 was collected from high-school students. Drawing sensible conclusion from learning experiments requires the result to be independent of the choice of training and test data among the complete set of samples. So building new sets by mixing NIST database becomes a shared view[57].

TD-1 contains 58646 numeral images. We used a subset of 10000 test images plus 5000 images from SD-3 as the test set. We combine the remaining 48646 in TD-1 with another 20000 numerals from SD-3 as the training set. After shuffling, we got a training set of 68646 images and a test set of 15000 images.

A detailed description of the two databases is given in Appendix C.

Size-normalized raw images of the numerals are used as the input vector of BPNet based on the consideration of avoiding distortions that may be introduced by thinning etc.. This method of direct regression on image is also justified by the competitive results obtained by us and other researchers [112, 58, 57].

The input vector is 22*15, which comes to 330D and thus defines the number of nodes at the input layer. We used one hidden layer with 50 nodes and 10 nodes at the output layer corresponding to the 10 classes of numerals.

During the training by BP algorithm, we initialize all the connecting weights to random small values. 4000 training samples are repeatedly fed into the network and the weights are updated according to Equation (17).

To define the stopping strategy of training process, we compute $flag(k)$ for each sample: $flag(k)$ is set as 1 if $E_k \geq 0.05$ or the class conclusion for the training sample does not agree with the real identity of the sample. A sum of $flag(k)$ for all training samples is then obtained

$$F = \sum_{k=0}^{P} flag(k) \tag{18}$$

The training process stops if one of the following conditions is satisfied:

1. $F \leq 0.003 * P$.

2. Number of training epochs exceeds a threshold of 500.

The rationale behind this strategy is to combine the training with cross-validation to ensure the generalization ability of the network.

The parameters of our BPNet are set as follows:

- The learning rate is set to be $\eta = 0.3$. When the number of epochs is greater than 300 or E is smaller than 5.0, η is changed to 0.2.
- The momentum of training is set to be $\alpha = 0.7$. When the number of epochs is greater than 300 or E is smaller than 5.0, α is changed to 0.5.
- Slope factor β is set as 1.0.

Figure 7 shows the diagram of E with respect to the number of training epochs on MNIST database. We can see from Figure 7 that E drops very fast at the first several epochs, then goes down at much smaller steps.

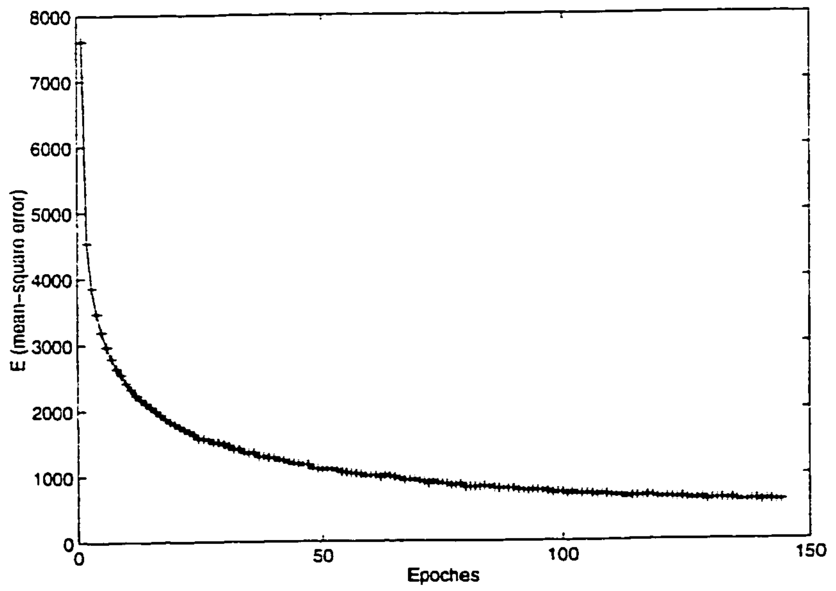


Figure 7: Learning in BPNet (On MNIST Databases)

We test error notes with respect to the number of different training epochs and get the diagram in Figure 8. It is observed that after 50 epochs, the test error starts increasing, while the training error still decreasing over time, this is when the so-called "over-training" occurs.

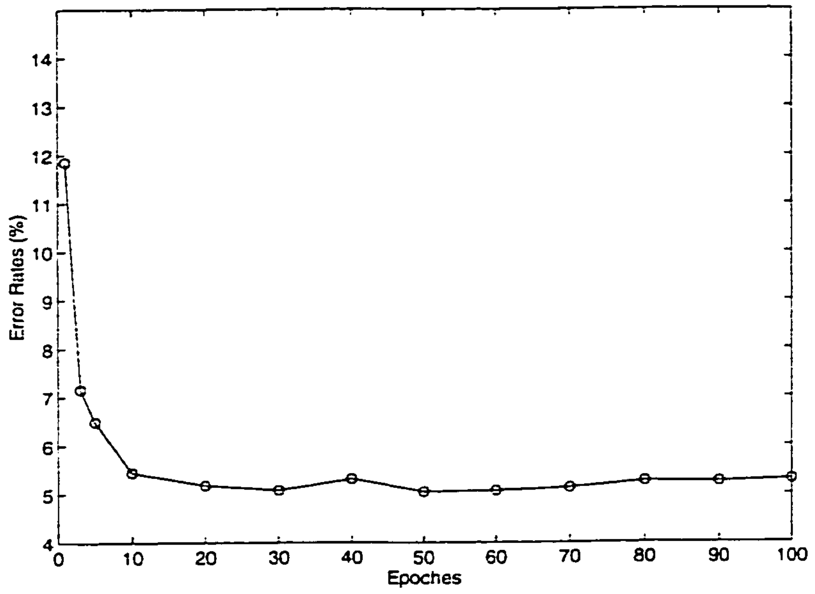


Figure 8: Error Rates of BPNet vs. Training Epochs (On MNIST Databases)

The final performance results are listed in Table 9.

Database	Correct(%)	Substitution(%)	Precision(%)
CENPARMI	93.56	6.44	93.74
MNIST	94.95	5.05	94.93

Table 9: Test Results of BPNet

As a single recognition method before any combination, the result of BPNet is satisfactory. Performance on CENPARMI database is slightly lower than on MNIST database, which may be explained by the smaller training set size of the former. Among the errors, most common ones are the confusions between numerals 3 and 5, or between 4 and 9.

Table 9 presents the results without rejection. One reason is that a combination will be done after. Another more important consideration is that rejection criterion is essentially an evaluation and verification of results and we will leave the task to our more capable verifiers in the following chapters. However, we give out precision rates to provide an understanding of this measurement and for later examining the effectiveness of our verifiers.

3.2 Convolutional Network

3.2.1 Network Topology and Learning Algorithm

Convolutional networks combine three architectural ideas to ensure some degree of shift, scale and distortion invariance: local receptive fields, shared weights (or weight replications), and spatial or temporal sub-sampling.

The neocognitron[27] can be considered as the first realization of convolutional networks. Hubel and Wiesel's discovery of locally-sensitive, orientation-selective neurons in the cat's visual system[45] leads to the first extensive use of receptive field in artificial neural networks. Later variants of convolutional networks have actively been applied in the field of image analysis [57, 73, 66].

We adopted a typical convolutional network called LeNet[58]. The architecture is shown in Figure 9.

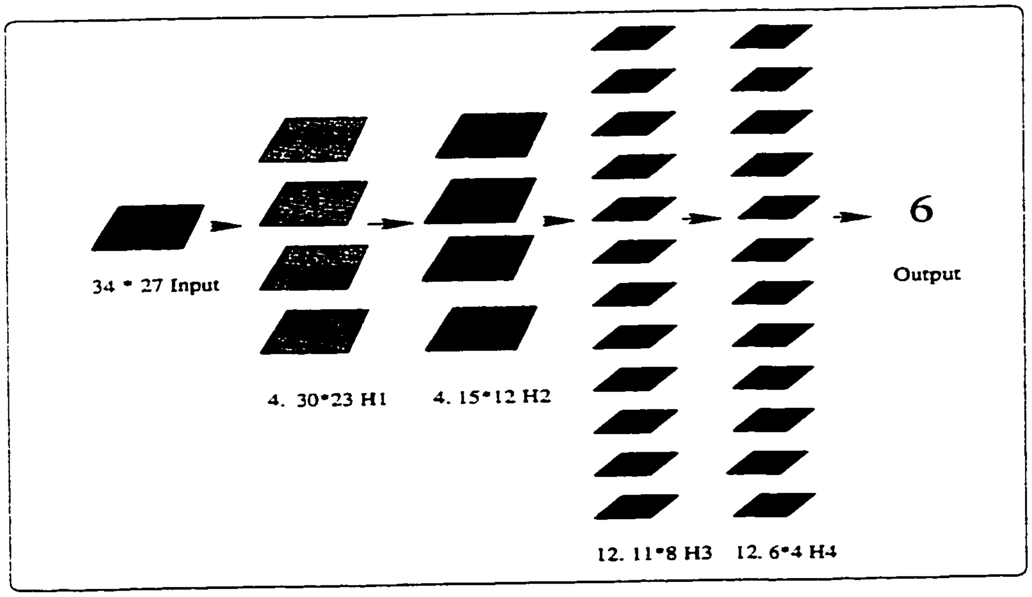


Figure 9: Architecture of LeNet

LeNet consists of four hidden layers named H1, H2, H3 and H4. The dimension of the input patterns is 34*27. It is bigger than 22*15 to avoid boundary overlaps. Units in each layer are arranged in two-dimensional planes except for the output layer.

Hidden layer H1 is the feature extractor. It is composed of four groups of feature maps of 30*23 units. Each unit takes its input from a 5*5 neighbourhood on the input plane. The receptive fields of units in this layer overlap. All the units in a map use the same set of 26 weights (including the bias). The units in another map share another set of 26 weights. There are 100 independent weights and 4 independent biases in H1.

Hidden layer H2 does averaging/subsampling. It is composed of four groups of feature maps of 15*12 units. Each unit takes its input from a 2*2 unit on the corresponding plane in H1. All the units in a map have the same weight value and the same bias. Therefore, H2 forms a local averaging and a 2:1 subsampling of H1 in each direction. There are four independent weights and four independent biases in H2.

Hidden layer H3 is the feature extractor. It is composed of 12 feature maps. Each map consists of 11*8 units. The connection scheme between H2 and H3 is quite similar to the one between the input plane and H1, but slightly more complicated because each unit receptive field is composed of one or two 5*5 neighbourhoods.

Hidden layer H4 does averaging/subsampling. It plays the same role as H2. The output layer is fully connected to layer H4.

We can see from the structure that a sequential implementation of a feature map would scan the input image with a single unit that has a local receptive field, and store the states of this unit at the corresponding location in the feature map. This operation is equivalent to a convolution, followed by an additive bias and squashing function, hence the name "convolutional networks".

In LeNet, the result of the sub-sampling layer is passed through a sigmoidal function. Weights are learned by back propagation algorithm.

3.2.2 Experiments

Experiments for UIHN involve CENPARMI and MNIST datasets. Size-normalized raw images of the numerals are used as the input vector of the dimension 34*27.

$flag(k)$ is set to 1 if $E_k \geq 0.10$ or the class conclusion for the training sample does not agree with the real identification of the sample. The stopping criterion is either of the following, with the same notation as in BPNet:

1. $F \leq 0.005 * P$.
2. Number of training epoch exceeds a threshold: 500.

The parameters of our LeNet are set as follows:

- The learning rate is set to $\eta = 0.05$ when training on CENPARMI database. With MNIST database, η is set to 0.001 due to much bigger number of samples. After the number of epochs exceeds 10, η is decreased to 0.0005.
- The momentum of the training is set to $\alpha = 0.5$.
- Slope factor β is set to 1.0.

Figure 10 shows the diagram of E with respect to the number of training epochs based on MNIST database.

We test the data with different training epochs and get the diagram in Figure 11.

The convolutional networks, represented by LeNet in our experiment, have better performance compared with BPNet. We did not see "over-training" phenomenon in the experiments which may be explained by the small learning rates. However, the training time is much longer. Training LeNet for 100 epochs on MNIST database needs 4 days, while BPNet just needs several hours (both on Sun Ultra60 Workstation).

The final performance results are listed in Table 10. The general performance of LeNet is higher than BPNet. The substitutions still occur on confusing numerals

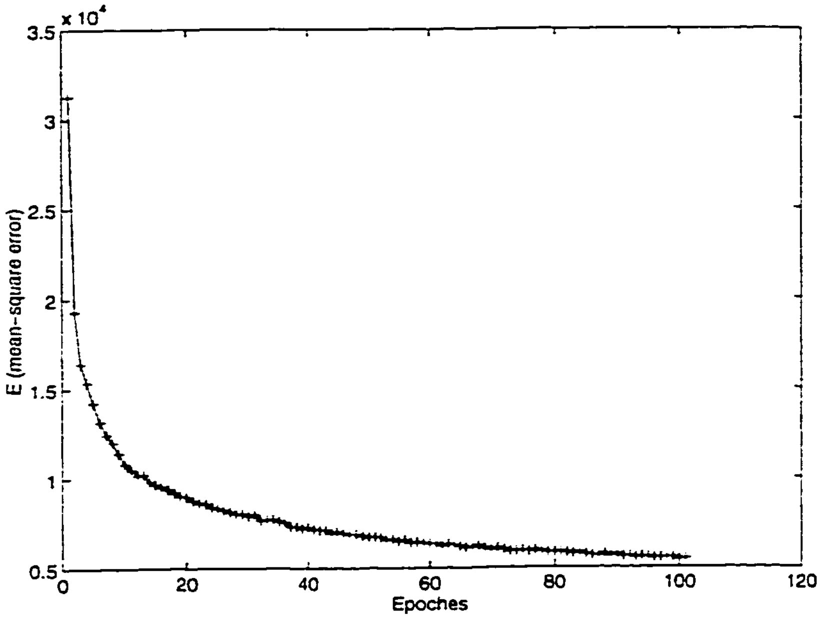


Figure 10: Learning in LeNet (On MNIST Databases)

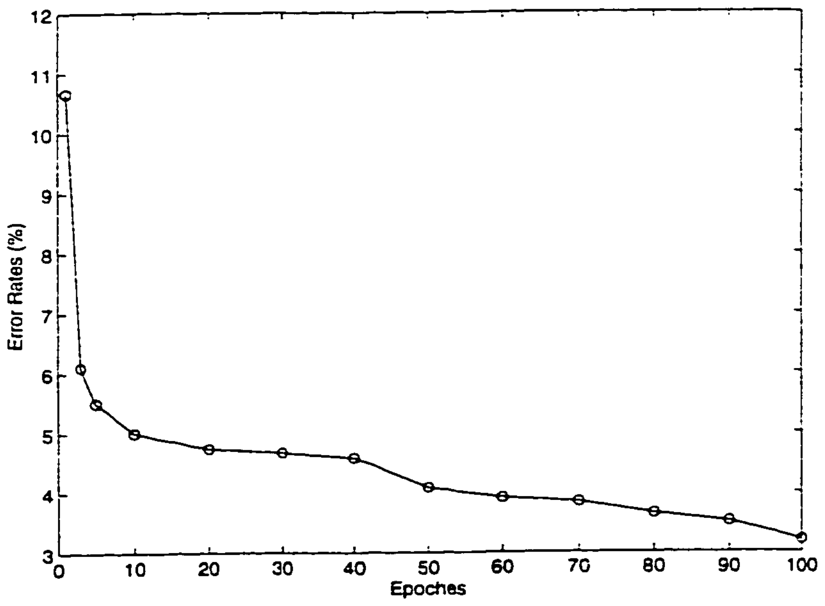


Figure 11: Error Rates of LeNet vs. Training Epochs (On MNIST Databases)

including (4,9) and (0,8). Results also show that the precision rate of numeral 0 is low. So we expect that the combination of several conventional approaches will lead to a better General Purpose Recognizer. In the following sections, we will examine two more methods for the purpose of building GPR.

Database	Correct(%)	Substitution(%)	Precision(%)
CENPARMI	95.69	4.31	95.43
MNIST	96.80	3.20	96.94

Table 10: Test Results of LeNet

3.3 Radial Basis Function Network

3.3.1 Network Topology and Learning Algorithm

Broomhead and Lowe [4] were among the first users of the RBF technique to provide an alternative to learning in artificial neural networks [71].

The name "Radial Basis Function" (RBF) comes from the fact that kernel functions are usually radially symmetric. These functions are placed at key locations in the input space. A map based on a weighted-sum interpolation of the kernel functions is then formed.

Typically, RBF network has one hidden layer of radial function nodes and an output layer of linear nodes, with full inter-layer connections, as illustrated in Figure 12. Weights of hidden nodes encode basis function centers and variances. Each hidden node computes the output based on the Euclidean distance between the input and the basis function center. Each output node's result is then a linear weighted sum of the hidden node outputs.

We adopt Gaussian kernel as the basis function of the form:

$$v_j = exp\left[-\frac{(X - W_{1,j})^T(X - W_{1,j})}{2\sigma_j^2}\right] \tag{19}$$

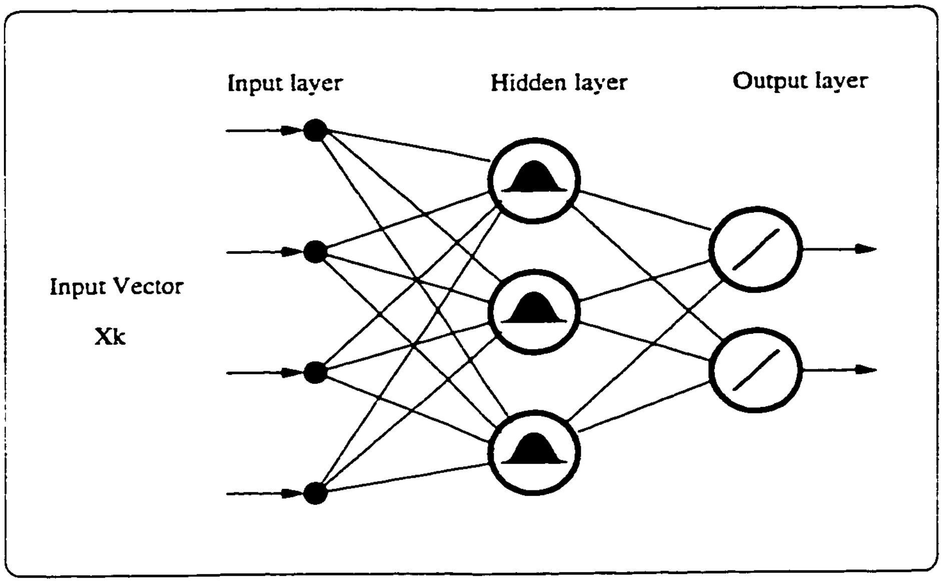


Figure 12: Radial Basis Function Network

$$j = 1, 2, ..n_h$$

where v_j is the output of the jth node in the hidden layer, X is the input pattern, $W_{1,j}$ is the weight vector for the jth node in the hidden layer, i.e., the center of the Gaussian function for node j; σ_j^2 is the normalization parameter for the jth node, i.e., the variance of the Gaussian function for node j. n_h is the number of nodes in the hidden layer.

The node equations of the output layer are given by:

$$u_i = W_{2,i}^T V, i = 1, 2, ..n_o \tag{20}$$

where u_i is the output of the *ith* node in output layer, $W_{2,i}$ is the weight vector for this node, and V is the vector of outputs from the hidden layer. n_o as before denotes the number of output nodes.

We can see that a fundamental difference between MLP and RBF nets is the way in which hidden units combine values coming from preceding layers in the network–MLP use inner products, while RBF use weighted Euclidean distance. There are also differences in the customary methods for training MLPs and RBF networks. Most RBF learning algorithms separate the learning of RBF into two stages:

- Learning in the hidden layer — get the centers and variances of the Gaussian Functions.
- Learning in the output layer — get the weights between the hidden layer and output layer.

In our experiment, an unsupervised method (K-means clustering algorithm) is used in first-stage learning while a supervised method (Least Mean Square algorithm) is used in the learning process of the output layer. The two algorithms are listed in Table 11 and Table 12 respectively.

```
For j = 1, ..., $n_h$
    Initialize the cluster center $W_{1,j}$
End
/* Group patterns */
Repeat
    For k = 1, ..., P
        Assign $X_k \rightarrow \Theta_{j*}$, if j* achieves $\min_j ||X_k - W_{1,j}||$;
    End
    /* compute the sample means */
    For j = 1, .., $n_h$
        $W_j = \frac{1}{|\Theta_j|} \sum_{X_k \in \Theta_j} X_k$
    End
Until there is no change in cluster assignments from one iteration to the next.
```

Table 11: K_Means Algorithm (Θ_j denotes cluster j, $|\Theta_j|$ denotes the cardinality of Θ_j.)

3.3.2 Experiments

The experiments are still based on CENPARMI and MNIST databases.

The architecture of our RBF networks is designed as follows:

```
For i = 1, ..., $n_o$
    Initialize the weights $W_{2,i}$
End
Repeat
    /* Compute outputs */
    For i = 1, ..., $n_o$
        Compute $u_i$ as $u_i = W_{2,i}^T V$,
            where V is the output vector of the hidden layer.
    End
    For i = 1, ..., $n_o$
        $e_i = u_i - d_i$, where $d_i$ is the desired output of $ith$ output node.
    End
    /* Update weights */
    For i = 1, ..., $n_o$
        $\Delta W_i(k+1) = -\eta e_i V + \alpha \Delta W_i(k)$
    End
Until the termination condition is reached.
```

Table 12: LMS Algorithm

- Input Layer: 330 nodes;
- Hidden Layer: 20 nodes;
- Output Layer: 10 nodes.

Training algorithms have been described in the above section. As in MLP and convolutional networks, learning rate η and momentum α are important parameters during training. However, less attention has been paid to RBF network in the task of recognizing UIHN. What are the roles of these parameters in RBF network? Under which condition can we get a set of good-performance weights without paying too much time on training? Do they have the behaviour similar to MLP network training? These are some questions deserving further investigation. Table 13 gives the numbers of training epochs needed and the corresponding recognition rates on CENPARMI database versus α and η,.

$\alpha \downarrow \rightarrow \mu$	0.1	0.3	0.5	0.7
0.0	456 (91.8%)	76 (88.8%)	97(86.2%)	26(81.8%)
0.3	223 (90.0%)	74 (87.4%)	69(85.0%)	11(81.9%)
0.5	116 (89.8%)	70 (87.0%)	7 (79.6%)	11(81.8%)
0.7	32 (82.6%)	17 (78.8%)	11(80.2%)	7 (77.2%)

Table 13: Training Pattern in Terms of α and η in RBF Networks

We can draw the following conclusions on the effects of learning rate and momentum factor on the convergence speed of training process and performance:

1. Adding and increasing of momentum factor (α) dramatically increases the convergence speed. It also introduces some negative effect on the final performance. In Table 13, when the value of α is less than 0.5, we can see that the effect on performance is slight while the time needed in training decreases a lot (results in the same column). The only exception is the (0.5, 0.5) pair in Table 13, which converges only in iteration 7 and gives a poor performance of 79.6%. One

possible explanation is that it reaches a local minimal soon after the training begins.

2. A larger learning factor (μ) also leads to a faster convergence. But its effect on performance seems bigger even when the value is small. We can see this from Table 13: When μ increases from 0.1 to 0.5, the performance (rates in the same row) drops several percents.

Performance of RBF network is given in Table 14, which is comparable to BPNet but lower than LeNet. More discussions of RBF properties will be given in next section in which we compare it to tree classifier.

Database	Correct(%)	Substitution(%)	Precision(%)
CENPARMI	91.80	8.20	91.32
MNIST	94.46	5.54	94.57

Table 14: Test Results of RBF Network

3.4 Tree Classifier

3.4.1 Learning Algorithm

The use of automatically trained tree classifiers has been studied by many researchers. The famous CART (classification and regression trees) [3] system was used successfully in areas like medical diagnosis and signal classification. The same algorithm lays the foundation of Quinlan's ID3 and the recent C4.5, although some details in splitting and pruning are different [79].

A trained tree is shown in Figure 13. Among the various methods for designing a decision tree, the best known is the top-down method. There are three main problems in top-down induction of decision trees:

1. How to classify new observations, given a decision tree. The most common approach associates each leaf with a single class and then assign that class to

all new observations which reach that leaf. Typically, the associated class is the one with the largest number of examples reaching the leaf.

2. How to determine the test to associate with each node of the tree, i.e., how to split and construct the tree.

3. How to determine the leaves. It can be solved by stopping the growth of a tree (pre-pruning) or retrospectively reducing the size of a fully expanded tree by pruning some branches (post-pruning).

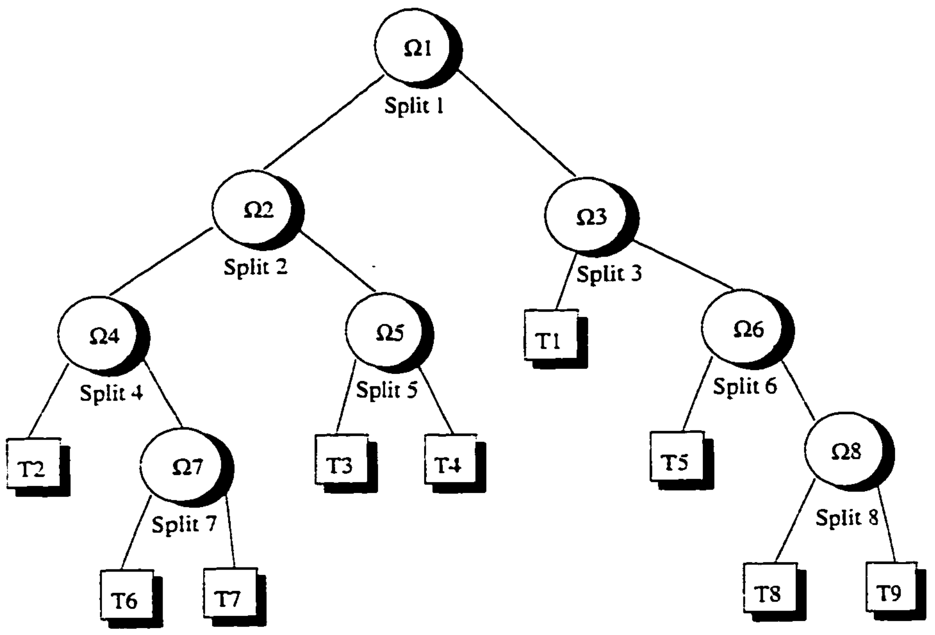


Figure 13: A Trained Tree Classifier. Ω stands for the sample space associated with the internal nodes, upon which the split rules are applied. T stands for terminal nodes. Each terminal node is assigned to a class with the largest number of examples in the node.

C4.5 classifier, as what we use in our experiments, constructs the tree by recursively splitting the input space into polygons until each subset in the partition contains cases of a single class or until no test offers any improvement. Data could be quantitative and/or qualitative. Each path in the tree that starts at the root and terminates at a leaf node represents a rule that has been "learned" from the training set.

During learning, the *gain ratio criterion* is used to maximize the information gain. which expresses the useful proportion of information generated by the split on test X:

$$\text{gain ratio}(X) = \text{gain}(X)/\text{split info}(X)$$

Entropy information is included in *gain(X)* and *split info(X)*. It is a modification of ID3 which only used gain(X) as the criterion. The result is often a very complex tree that "overfits the data". Pruning is then done to trim the tree to the right size.

The tree built this way can be used for a classification task including handwritten digits. If a decision node is encountered at which the relevant attribute value is unknown, so that the outcome of the test cannot be determined, all possible outcomes are explored and the class with the highest probability is assigned as the predicted class.

3.4.2 Experiments and Comparisons

Experiments are conducted on MNIST database. To study the behaviour of a tree classifier in the task of recognizing UIHN, a comparative study is also done between tree classifier and neural network classifier in terms of performance, speed and memory requirement, using C4.5 and RBF network as illustrative examples.

First of all, we notice that there are some inherent difference between C4.5 and RBF, which may reflect in the experimental results:

- RBF has a much smoother boundary because of its radial basis function, while C4.5's decision boundaries are typically straight lines and crisp corners caused by the recursive partitions.
- Number of hidden layers in RBF could be chosen manually so as to avoid the overfitting or underfitting, since we are using the K-means algorithm in the first stage of learning. But tree pruning is a completely automatic method.

- The weights of RBF are learned adaptively. Due to its distributed structure, the small change of a single weight may affect all other units and the final output, but only in a minor way. C4.5 is not easy to adapt. Also, a suboptimal partition may accumulate the error along the tree.

C4.5 gives a recognition rate of 85.27% on MNIST test set. Table 15 lists the time needed for training and classification of the two methods. The learning rate and momentum factor are both 0.1. All experiments are conducted on a Sun Ultral workstation. The classification time is for recognizing a single digit. Table 16 lists the memory requirement during training and classification. From the experimental results in Tables 15, 16 and Table 14 in the previous section, we can observe the following:

- Performance: Although the performance of the C4.5 shows that it is also feasible for the complicated problem of recognizing UIHN, RBF network has a better performance than C4.5. The reason may be attributed to RBF's smoother decision boundary and ease in adaptation.

- Speed: RBF's training time is much longer than C4.5. With the given training set, it took several days for the RBF network to converge while the tree classifier only uses half an hour. But their classification speeds are comparable and both are fast enough for practical use.

- Memory Requirement: Both classifiers use tolerable memory resources. In the experiments, we did not put training examples into the memory due to the large number of samples which actually implies a slower training speed.

Considering the tolerable training time and memory expense, a higher performance becomes more desirable, which contributes to the decision that we will use neural networks to build the general purpose recognizer for our R&V system in this thesis. Nevertheless, it is worthwhile to point out that our work for the first time proves the feasibility of applying tree classifiers to UIHN problem and some recent improvement of tree classifiers[38] may increase the classification accuracy of the method.

Classifiers	RBF network			C4.5 Classifier
Training	1st Stage Training	2nd Stage Training	Total	28 minutes
	2.5 hrs	50 hrs	52.5 hrs	
Classification	0.01 seconds			0.005 seconds

Table 15: Training Time and Classification Time of Neural and Tree Classifiers

Classifiers	RBF network	C4.5 Classifier
Memory Usage	0.44 Megabytes	0.53 Megabytes

Table 16: Memory Requirements of Neural and Tree Classifiers

3.5 Combination and Overall Performance of General Purpose Recognizer for UIHN

In this section, we combine the decisions of the three neural classifiers to produce the recognition result of the General Purpose Recognizer (GPR).

The combination scheme is a consensus of the methods that compensates for the individual weaknesses while perserving the strengths.

The class identity of the choices is denoted as $I_{i,j}$, the output value of the choices is denoted as $U_{i,j}$, where $i = 1, ..., L$, L is the number of possible choices of each classifier taken into consideration; $j = 1, ..., N$, N is the number of classifiers. Here $i = 1$ indicates the top choice of the neural classifier, $i = 2$ is the second choice. $j = 1$ represents the convolutional network, $j = 2, 3$ represent the BPNet and RBF network respectively.

The combination is conducted as follows:

$$\begin{cases} I_{1,j_1} & \text{If } \exists j_1, j_2 \in \{1, ..., N\}, I_{1,j_1} = I_{1,j_2}; \\ Rejection & \text{If } \forall i_1, i_2 \in \{1, ..., L\}, j_1, j_2 \in \{1, ..., N\}, I_{i_1,j_1} \neq I_{i_2,j_2}; \\ Voting & \text{Otherwise.} \end{cases} \tag{21}$$

```
Initialize SUM(k) = 0, k=0,..,9
For i=1,2
    For j=1,2,3
        SUM($I_{i,j}$) = SUM($I_{i,j}$) + $V(I_{i,j})$;
    End
End
```

Table 17: Combined Vote

In this way, if at least any of the two classifiers agree on the pattern identity as their top choice, their conclusion is considered as that of GPR. If there is no common conclusion among the classifiers, a rejection is given. Otherwise, a voting among the possible choices is done:

The vote of each identity $I_{i,j}$: $V(I_{i,j})$ is set as $\frac{1}{L*N}$, i.e., $\frac{1}{6}$. The sum of the vote on each $I_{i,j}$ is computed as presented in Table 17.

We can then get the final decision:

$$\begin{cases} I_{i,j} & \text{If } SUM(I_{i,j}) \text{ gives the largest total vote;} \\ I_{i,1} & \text{If } I_{i,1} \text{ gets the same vote as another identity and } U_{i,1} \geq \text{threshold;} \\ Rejection & \text{Otherwise.} \end{cases} \tag{22}$$

We can see that the conclusion of GPR is the $I_{i,j}$ which gives out the highest SUM. If there exists a tie, the conclusion from the best performing classifier is considered. To ensure that such a decision is reliable, the output value should exceed a threshold t. Otherwise, a rejection is given.

The combination scheme is an expansion of the majority rule and it takes more factors into consideration while keeping its simplicity. The results on CENPARMI and MNIST databases are presented in Table 18.

With this scheme, the performance in terms of precision rates is slightly improved due to respecting and selecting the decisions of individual classifiers. However, it is instructive to point out that, unlike the verifiers in following chapters, no inherent ability of distinguishing different classes is embedded in any combination scheme. As we shall see, specially trained verifiers are expected to enhance the precision rates of the system in a drastic way due to their inherent properties, so as to highly improve the reliability of the system.

Database	Correct(%)	Substitution(%)	Rejection(%)	Precision(%)
CENPARMI	94.83	3.32	1.85	96.66
MNIST	95.97	2.10	1.93	97.89

Table 18: Test Results of General Purpose Recognizer for UIHN

Chapter 4

Verifier for Unconstrained Isolated Handwritten Numerals

"Perhaps I've trained myself to see what others overlook. If not, why should you come to consult me?"
— Sherlock Holmes, A Case of Identity

In this Chapter, we present the verifier for UIHN. We first analyse the general need for a good verifier. The property is then exemplified by a novel neural approach of Quantum Neural Network. In-depth experiments and discussions are presented for both synthetic and real world data. The verifier for UIHN is then proposed in a class-specific way. The effectiveness of R&V system for UIHN is studied and evaluated by hypothesis testing based on Precision Rates.

4.1 Theoretical Requirements

A fundamental requirement for a valid pattern verifier is that it is expected to perform a reliable confirmation or negation of the pattern.

From the human point of view, satisfactory verification of uncertain factors will employ linguistic hedges, rules-of-thumb experience, intuition and other heuristics. Our mission is thus to incorporate human reasoning into the *quantitative* and/or *qualitative* aspects of machine intelligence.

Here we examine the decision boundary of pattern space to extract general *quantitative* features we expect from a good verifier. *Qualitative* features will be evaluated by taking into account the requirement of the specific application domain — handwritten numeral recognition.

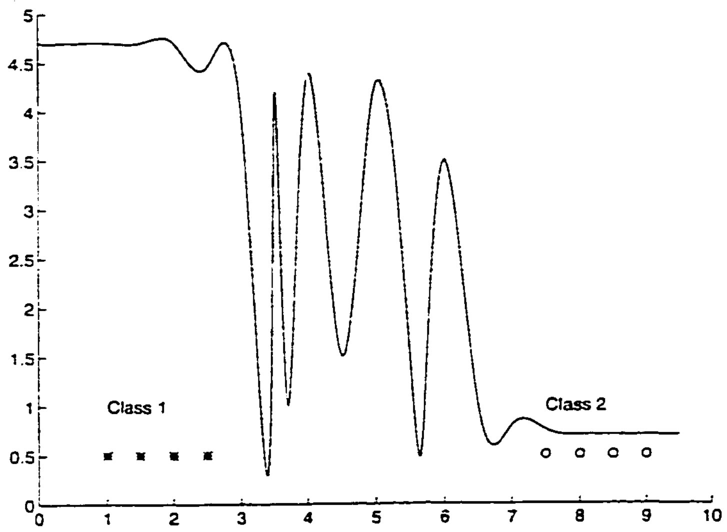
Consider the decision space of a simple example as in Figure 14. Let two classes of data sets be represented by "stars" and "circles". If the two classes are by nature well clustered and separated as in Figure 14(a), a classifier with rough and crisp decision boundary may be adequate for the classification problem.

For a complex situation that contains overlapping and confusing patterns which is the common case of practical classification problem, including handwritten numeral recognition, the classifier in Figure 14(a) will give unreliable decisions, especially along the decision boundary. To avoid the consequence of either very low reliability or very high rejection rate, the ability of detecting and identifying the uncertainty within the data becomes essential. The property illustrated in Figure 14(b) which encodes the structure of feature space into the membership function is therefore ideal for a good verifier.

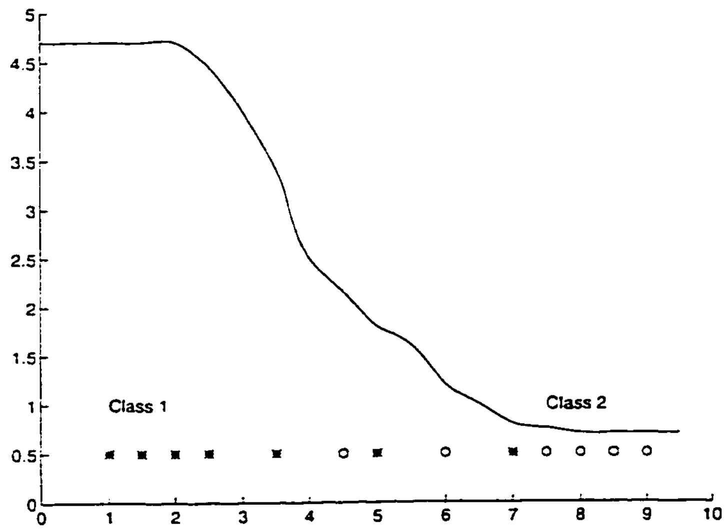
Unfortunately, some classifiers lack this kind of inherent fuzzy feature. Their crisp decision boundary makes them inadequate for the pattern recognition task, and definitely not good for the pattern verification task. For example, there are clear arguments for believing that the conventional feedforward network is not very well suited for general problems of pattern verification [29, 78].

Conventional feedforward neural network's classification ability is influenced by the topology of the feature space in the following manner:

Consider the problem of classifying objects described by feature vectors $x \in R^d$ to one of n_o classes $C_1, ..., C_{n_o}$. Let feature vector p belong to the ith class and let vector q not belong to the ith class. Let G_i be the gradient of the response of the ith



(a) Well separated classes with crisp decision boundary.



(b) Overlapping class with ideal decision boundary.

Figure 14: Two Class Example of Requirement on Decision Boundary

output unit evaluated at some point between p and q. Then it can be shown [78] that

$$||G_i|| \geq c\frac{\gamma}{||p-q||} - M \tag{23}$$

where $||\cdot||$ denotes L_2 norm on R^d, c and M are positive constants and γ has a value between 0 and 1. This formula provides the theoretical lower bound of $||G_i||$. As $||p-q|| \to 0$, i.e., the classes are coming closer together or their overlap increases, $||G_i||$ will increase. We can see that the network forms an increasingly sharp boundary as $||p-q|| \to 0$, which implies that conventional feedforward neural network loses its ability to function as a fuzzy classifier for data sets of overlapping or closely spaced feature vectors.

Comparatively, a novel neural network — Quantum Neural Network overcomes this weakness by combining the advantages of neural modelling and fuzzy theoretic principles, which make it a good candidate for verification purpose.

4.2 Quantum Neural Network

The idea of applying Quantum Neural Network originates from exploiting fuzzy feedforward neural network with the intention to improve the ability of recognizing confusing characters, after realizing the limitation of conventional neural network with regards to crisp boundary. Nevertheless, experiments reveal its capability of handling more general verification problem.

4.2.1 Quantum Neuron

Quantum Neural Network has an inherently fuzzy architecture which can encode the sample information into discrete levels of certainty/uncertainty. The goal is accomplished by using *quantum neurons* in the hidden layer of the network. The transfer function (activation function) of quantum neuron has the ability to form *graded* partitions instead of crisp linear partitions of the feature space. One possibility of obtaining this kind of transfer function is to take the superposition of *ns* sigmoidal functions,

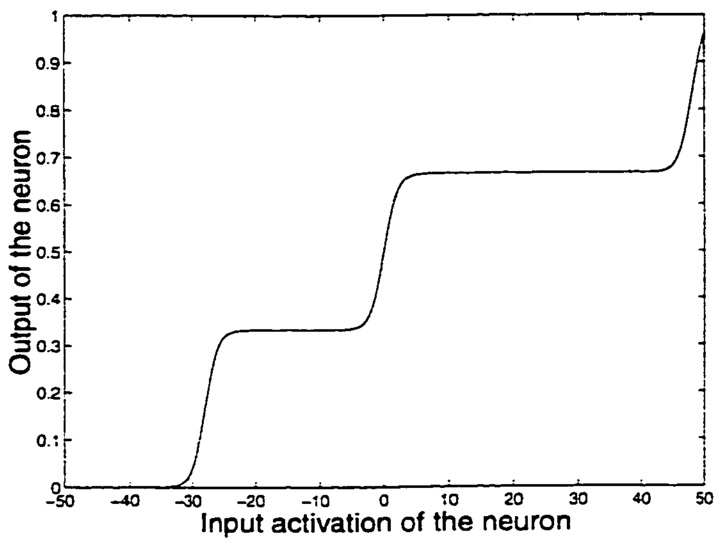


Figure 15: Output of a 3-level Transfer Function in Quantum Neuron ($\theta_1 = 28.0, \theta_2 = 0.0, \theta_3 = -48.0, \beta = 1.0$)

each shifted by quantum interval θ_s (s=1, ..., ns), where ns is called the total number of quantum levels.

The output of quantum neuron can be written as

$$\frac{1}{ns}\sum_{s=1}^{ns} sig(\beta * (W^T X - \theta_s)) \tag{24}$$

where $sig(x) = 1/(1 + exp(-x))$, W is the weight vector, X is the input vector, β is a slope factor. Defining $W^T X$ as the input activation of the neuron, visually, we can get the output of a 3-level (ns=3) transfer function as shown in Figure 15.

The quantum intervals of QNN will be determined by training. No *a priori* fuzzy measure needs to be pre-defined. Given a suitable training algorithm, the uncertainty in the sample data will be adaptively learned and quantified. If the feature vector lies at the boundary between overlapping classes, the QNN will assign it partially to all related classes. If no uncertainty exists, QNN will assign it to the corresponding class.

4.2.2 Training Algorithms

The gradient-descent-based algorithm is used to train the QNN. In each training epoch, the training algorithm updates both connectivity weights among different layers and quantum intervals of the hidden layer.

Weight updating is carried out by the standard backpropagation algorithm . Once the synaptic weights have been obtained, quantum intervals can be learned by minimizing the class-conditional variances [19] at the outputs of the hidden units.

The variance of the output of the *ith* hidden unit for class C_m is

$$\sigma_{i,m}^2 = \sum_{x_k : x_k \in C_m} (\langle O_{i,m} \rangle - O_{i,k})^2 \tag{25}$$

where $O_{i,k}$ is the output of the *ith* hidden unit with input vector x_k, $\langle O_{i,m} \rangle = \frac{1}{|C_m|} \sum_{x_k : x_k \in C_m} O_{i,k}$, $|C_m|$ denotes the cardinality of C_m.

By minimizing $\sigma_{i,m}^2$, we can get the update equation for $\theta_{i,s}$ as follows [78]: for each hidden unit i and its *sth* quantum level ($s = 1, \ldots, ns$),

$$\Delta\theta_{i,s} = \eta \frac{\beta}{ns} \sum_{m=1}^{n_o} \sum_{x_k : x_k \in C_m} (\langle O_{i,m} \rangle - O_{i,k}) * (\langle \nu_{i,m,s} \rangle - \nu_{i,k,s}) \tag{26}$$

where η is the learning rate; n_o is the number of output nodes, i.e., the number of classes.

$$\langle \nu_{i,m,s} \rangle = \frac{1}{|C_m|} \sum_{x_k : x_k \in C_m} \nu_{i,k,s} \tag{27}$$

$$\nu_{i,k,s} = O_{i,k,s}(1 - O_{i,k,s}) \tag{28}$$

$O_{i,k,s}$ is the output of the *sth* quantum level of the *ith* hidden unit with input vector x_k.

The above procedures are summarized in Table 19.

Table 19: Training Algorithm of Quantum Intervals

```
For k= 1, ..., number of input samples
    For i = 1, ..., number of nodes in the hidden layer
        For s = 1, ..., ns
        Compute $O_{i,k,s}$
        Compute $\nu_{i,k,s}$
        End
    Compute $O_{i,k}$ for each hidden layer node
    End
End

For m = 1, ..., number of classes
    For i = 1. ..., number of of nodes in the hidden layer
        For s = 1, ..., ns
        Compute $\langle \nu_{i,m,s} \rangle$
        End
    Compute $\langle O_{i,m} \rangle$
    End
End

For k = 1, ..., number of input samples
    For i = 1, ..., number of nodes in the hidden layer
        For s = 1, ..., ns
            $\theta_{i,s} = \theta_{i,s} - \eta \frac{\beta}{ns} \sum_{m=1}^{n_o} \sum_{x_k : x_k \in C_m} (\langle O_{i,m} \rangle - O_{i,k}) * (\langle \nu_{i,m,s} \rangle - \nu_{i,k,s})$
        End
    End
End
```

4.3 Why Quantum Neural Network is Good for Verification?

This section presents experiments designed to evaluate QNN's fuzzy decision boundary. Synthetic numeral patterns are created as data sets for this purpose. Decision spaces of QNN on these input patterns are studied in terms of different ns (total number of quantum levels). Comparison with BP network is also done. Experimental results clearly demonstrate that QNN provides smoother multilevel partition of feature space while standard BP net exhibits a much sharper response. Decision boundary's fuzziness also turned out to differ according to ns and an optimal value exists for a certain classification problem. Results imply that QNN satisfies the requirement of a valid verifier as defined in Section 4.1. Details of this work can also be found in Zhou et al. [110] and [111].

4.3.1 Generation of Synthesized Data Sets

Both the Quantum Neural Network and standard BP used in our experiments have one hidden layer of 50 nodes, one input layer with $22^{*}15$ nodes and one output layer with 10 nodes. Binary raw images were normalized and fed to the network.

To study the response of networks to overlapping input patterns, we adopted a morphing technique of image processing to generate the synthetic confusing numeral patterns [28, 97]. We used *transition morph* which smoothly transforms one source image into another by creating intermediate transition frames to test the network's response. In our experiment, we used one numeral image as the source and its easily confused counterpart as the destination. (4,9), (3,5) or (0,6) can be such candidate pairs for experimental purpose. Here we use pair (4,9) as an illustrative example.

The skeletons of the images were interpolated by a set of control points using spline function. The source and destination arrays of the control points must have the same dimension to provide the one-to-one mapping. The control points within the arrays consist of x and y coordinates. Cross-dissolving was used to get the intermediate images. Instead of interpolating linearly between source and destination control points, we adopted tangent function to assign the weights. With the function's flat gradient near the origin, we can get more details in the middle region of transition, which is the most overlapped region of the feature space, i.e., the focus of our study.

The skeletons of all the images (source image, destination image, intermediate images) were dilated to produce a stroke width of 6 pixels. The images were then used as input patterns to the neural networks.

Figure 16 (a) shows the source image of numeral '4' with its control points. Figure 16 (f) shows the synthetic numeral '9'. Figures 16 (b)-(e) show several intermediate confusing images, with gradual transition from '4' to '9'.

In the experiment, 300 intermediate images were used to test the output of the neural networks.

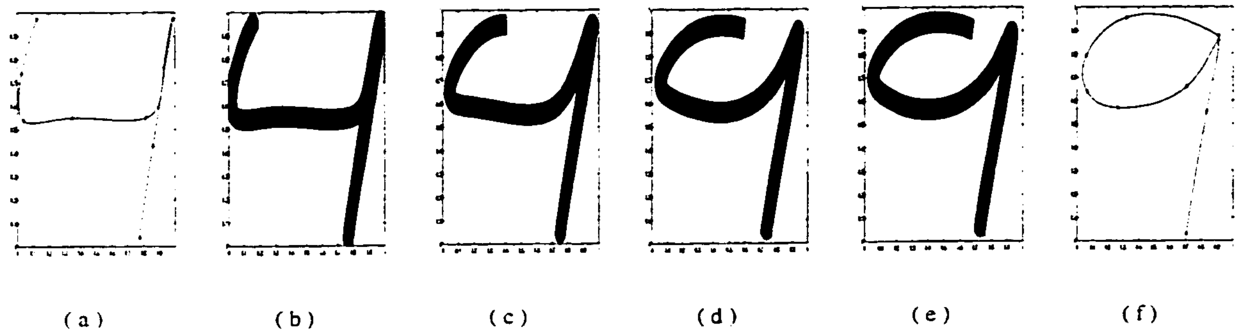


Figure 16: Synthesizing Image Patterns by Morphing

4.3.2 Experiments and Analysis On Synthesized Data Sets

Two-class training was done before we examined NN's output. 100 images of '4' with small variations from the source '4' plus 100 images of '9' with small variations from the destination '9' were used as the training set. Typically, training finishes within 2-3 epochs for both QNN and BP. The training set consists of only two numerals in order to avoid the complexity that may be introduced by other numerals. For example, some ambiguous patterns between '3' and '5' can look very much like a '6'.

The synthetic numeral patterns generated in Section 4.3.1 were then fed to the networks to examine the responses of the *4th* and *9th* output nodes. Ideally, those patterns similar to '4' will yield a strong response at the *4th* node and lower response at the *9th* and vice versa. The ambiguous patterns will yield similar outputs on both units (can be both high or low).

The difference between the output values of the 4th and 9th nodes (Y axis) in terms of input patterns (X axis) is depicted in Figure 17. On the X axis, 300 synthetic input patterns are indicated with the gradual transition from numeral 4 to 9. Note that in Figure 17, for ease of comparison, graphs A-E show only the approximations while graph F gives the detailed graph of QNN when ns is 12.

From Figure 17, we can observe the following:

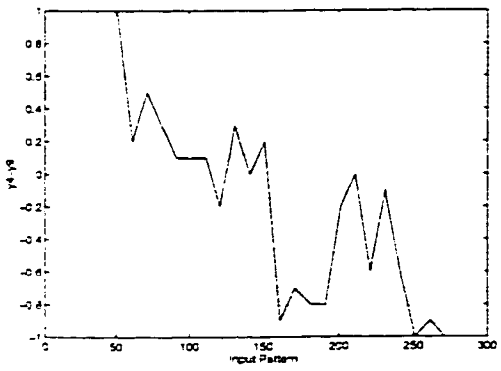
1. When the inputs are gradually morphed from 4 to 9, QNN provides a graded response for the overlapping input patterns, which implies the multilevel and smoother partition of the feature space. QNN response function is fuzzier and more intuitive than BP response function which exhibits sharp transitions, abrupt jumps and valleys as shown in Figure 17 (A) .

2. Response smoothness of QNN differs according to the number of quantum levels ns. For the example pair (4,9), $ns = 12$ turns out to be the optimal value. For ns greater than the optimal value, the fuzziness of response decreases and the coarseness increases.

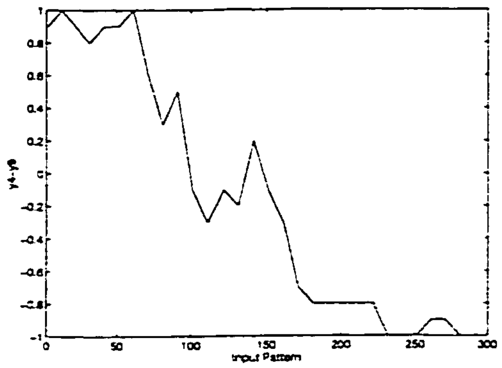
The above observations clearly indicate that QNN yields fuzzier decision boundary compared with conventional neural networks. This property can help QNN avoid unreliable conclusion for uncertain samples, which is ideal for verification purpose.

4.3.3 Experiments on Real Data and Concern of Confusing Pairs

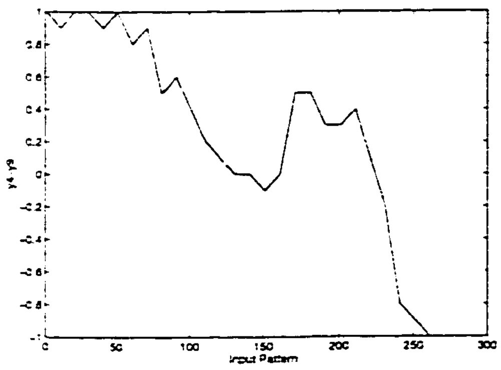
Recognition experiments conducted on CENPARMI database, with the results in Table 20, show that when the rejection rates are the same, QNN has a higher reliability



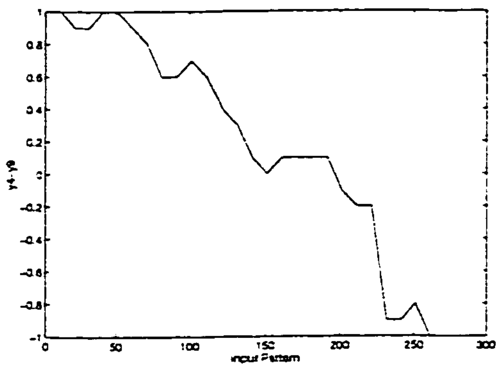
A: Standard BP network



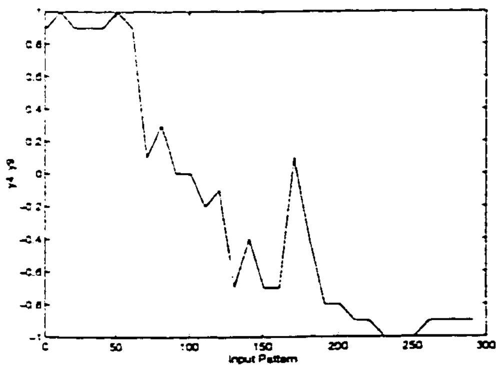
B: QNN with ns=3



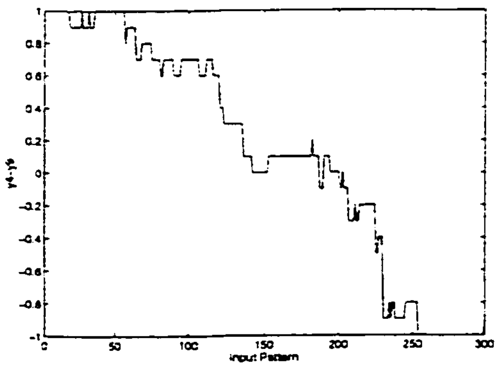
C: QNN with ns=6



D: QNN with ns=12



E: QNN with ns=15



F: Raw Output of QNN with ns=12

Figure 17: Outputs of BPNet and QNN with Different ns

than BP net, when we reduce the error rates of neural networks, BP gives a higher rejection rate than QNN.

However, instead of being used as a slightly better classifier, the main advantage of QNN network lies in its strong ability of recognizing confusing characters, which is suitable for more general verification purpose. Figure 18 shows the performance of BP and QNN in terms of confusing pairs. 22 confusing pairs for BP network are given along the X axis and (m,n) means n is misrecognized as m. Corresponding numbers of errors by QNN are also shown. Although QNN does not perform better on every pair, its improvement on those most confusing pairs is obvious. This pattern will be utilized in the following design of our verifier for UIHN.

Classifiers	BP Net				QNN (ns=6)			
Rates(%)	Recog.	Err.	Rej.	Reliability	Recog.	Err.	Rej.	Reliability
Zero Rej. Rate	93.56	6.44	0	93.56	95.65	4.35	0	95.65
Fixed Rej. Rate	87.44	3.58	8.98	96.06	90.02	1	8.98	98.90
Fixed Err. Rate	84.22	1	15.78	98.83	90.02	1	8.98	98.90

Table 20: Performance Comparison of BPNet and QNN. The error rate is controlled by thresholding the difference between two highest outputs.

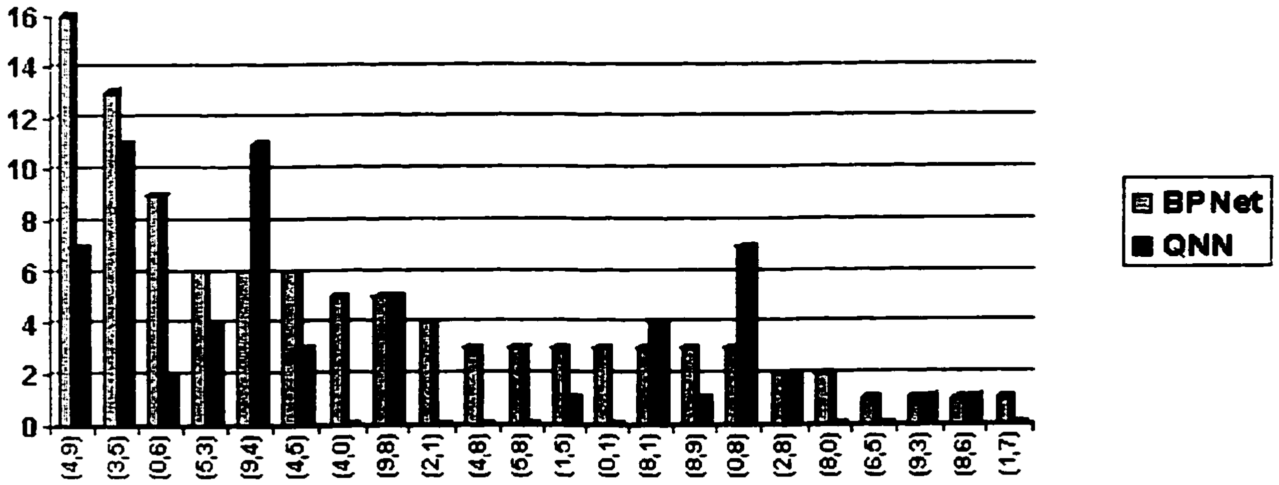


Figure 18: Most Confusing Numeral Pairs of BP and QNN

4.4 Architecture and Experimental Results

Based on the discussion of previous sections about the properties of Quantum Neural Network, our task now is to design verifiers for UIHN making use of QNN.

We proposed a two-layer module utilizing 10 QNN class-specific verifier and 18 QNN pairwise verifier. The architecture of these two kinds of verifier are shown in Figure 19.

Thresholding is still used as the rejection criterion. A rejection is given when the following condition is satisfied:

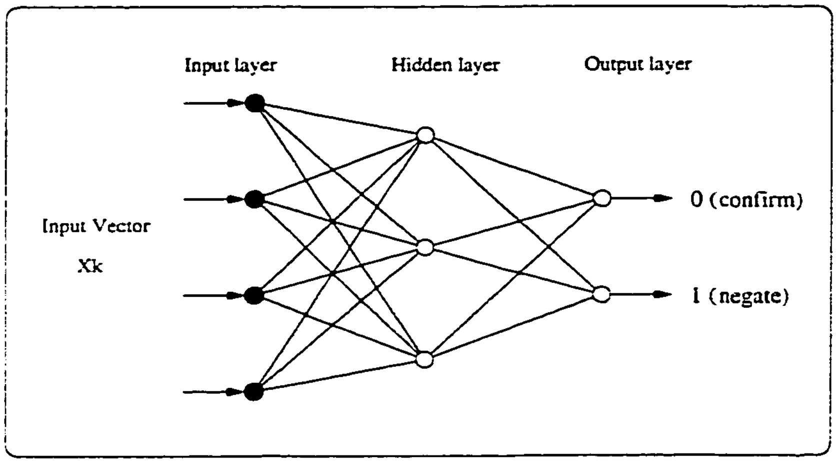
$$|O_1 - O2| < \theta_v \tag{29}$$

where O_1 and O_2 are the two maximum outputs of the nodes. θ_v is the threshold chosen for the particular class-specific verifier or pairwise verifier. It is the same criterion widely used in conventional neural networks. However, the inherent multilevel decision boundary and the ability to represent fuzzy information along overlapping classes are expected to make thresholding a much more reliable criterion than in the conventional neural networks.

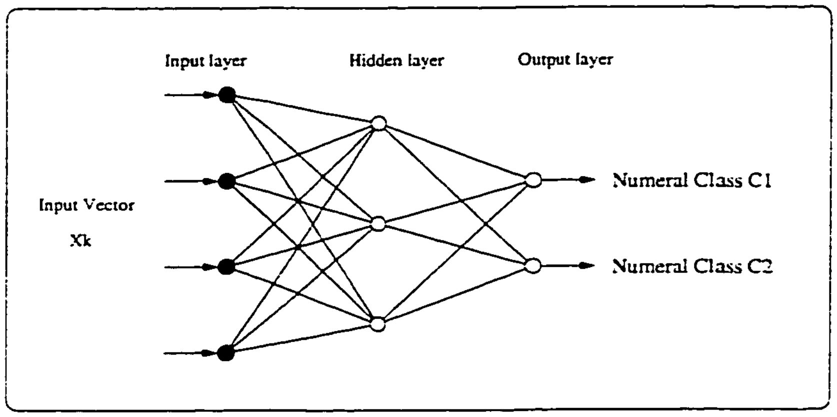
Each QNN class-specific verifier is trained independently by using the algorithm described in Section 4.2.2. To train the verifier V(i), all the sample of class i in MNIST training set are fed with the output (1,0), 10,000 other patterns are fed with the desired value (0,1). The training parameters are set as follows:

- The quantum level ns is set to 3;
- The learning rate is set to $\eta = 0.3$.
- The momentum of training is set to $\alpha = 0.3$.
- The number of hidden nodes is set to 20.

Each QNN pairwise verifier is trained by feeding the samples of the two related classes only, dedicated to verify the conclusions within a confusing pair. All training



A: Class specific QNN verifier, one for each digit class, denoted as V_0, V_1, ..., V_9. Output node 0 confirms the digit while output node 1 negates it.



B: Pairwise QNN verifier. 18 mostly confusing pairs are considered: (4,9), (3,5), (0,6), (4,5), (4,0), (9,8), (2,1), (4,8), (5,8), (1,5), (0,1), (8,1), (0,8), (2,8), (6,5), (9,3), (8,6), (1,7). Two output nodes represent the pair digits.

Figure 19: QNN Verifiers

samples of the two classes in MNIST training set are fed into the QNN verifier with the desired output of (1,0) and (0,1) respectively. The training parameters are set as follows:

- The quantum level ns is set to 3.
- The learning rate η is set to 0.7.
- The momentum of training α is set to 0.2.
- The number of hidden nodes is set to 10.

The diagram of the verifier for UIHN used in our R&V system is shown in Figure 20. The underlying rationale is to make use of the distinguishing ability of QNN on overlapping decision boundaries. The two layer scheme guarantees the full use of the verifier and provides a chance to get a correct recognition result in case the first layer verifier gives a negation. Figures 21, 22 and 23 give three typical dataflows of the verifier.

Ideally, we want the verifier to confirm all the true cases and negate all the wrong cases. Except in trivial cases, this ideal cannot be attained. Like all verification schemes, the verifier for UIHN has a problem of dealing with the balance between false positive and false negative. By false positive we mean irresponsible confirmation of a wrong conclusion from GPR. By false negative we mean negating a conclusion which is actually correct. For a fixed problem, it is usually impossible to make both types of errors arbitrarily small[6].

If we denote the probability of false positive as β, then we can define the *Power of Verification* as $1 - \beta$. It is common that the higher the power of verification, the higher the possibility of false negative. How to control the power of verification so that we can achieve high reliability while maintaining a reasonable high recognition rate is an important problem-related issue. In our verifier for UIHN, the control is represented by the ease of entering into second-layer verification, which decreases the probability of false negative. Threshold control is applied in such a way that the

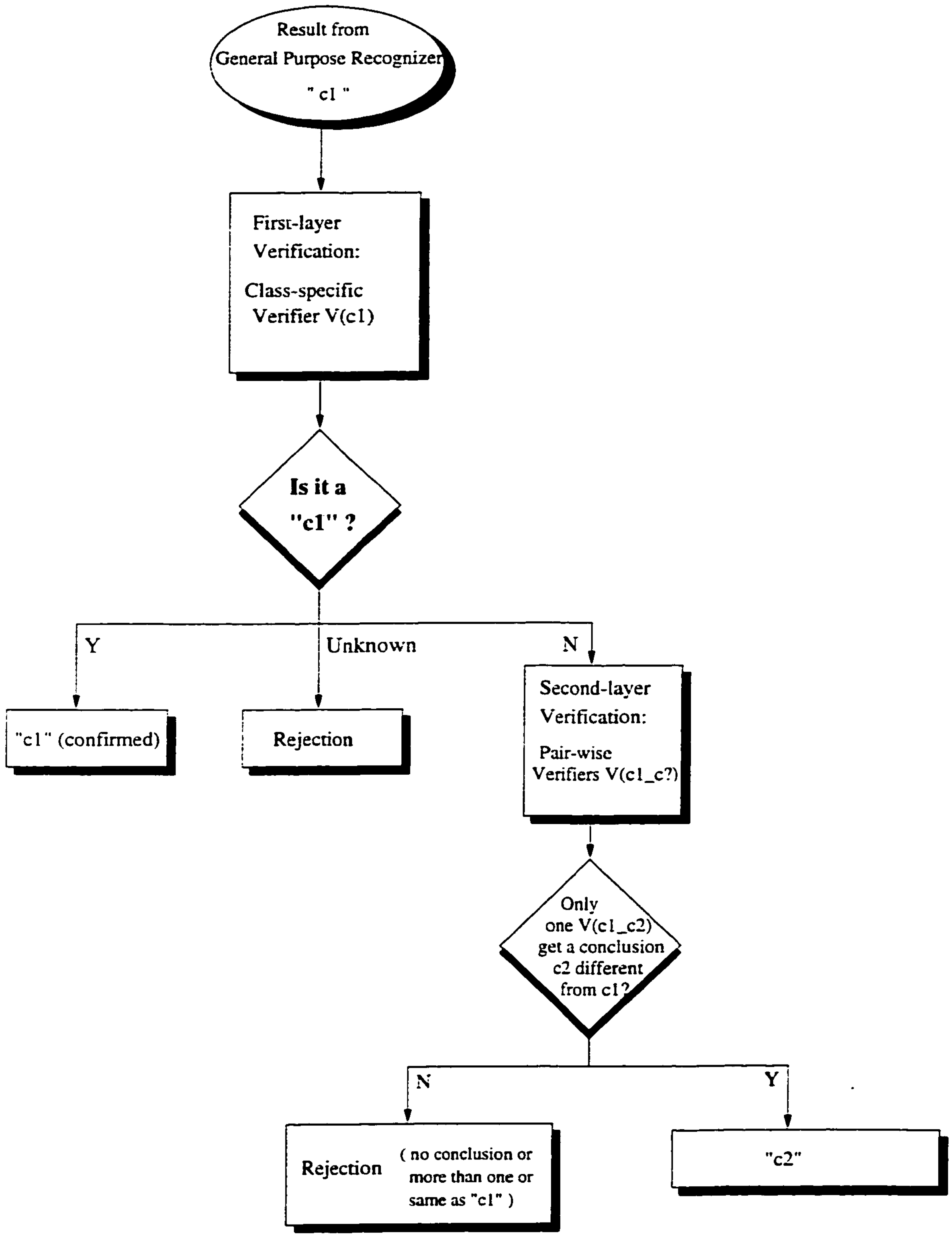


Figure 20: Diagram of Verifier in R&V system for UIHN

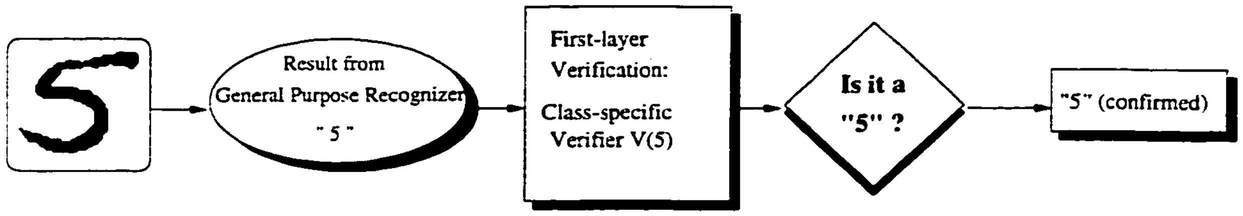


Figure 21: Example One of Dataflow in Verifier: Confirmation of a Good Sample

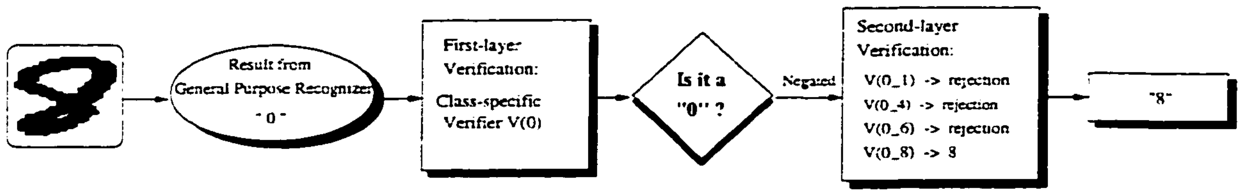


Figure 22: Example Two of Dataflow in Verifier: Negation and Second-layer Verification – Resulting in a New Result

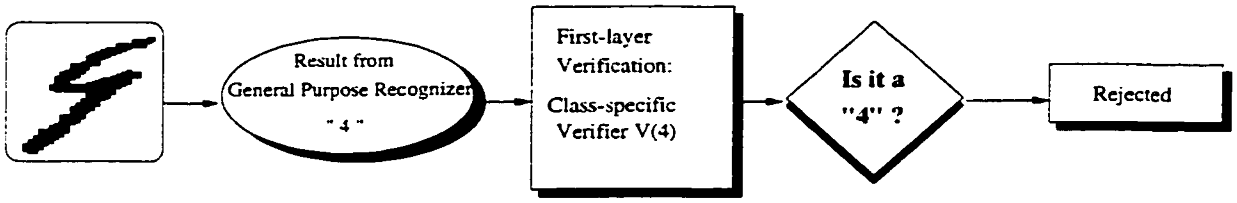


Figure 23: Example Three of Dataflow in Verifier: Rejection of a Confusing Example

pairwise verifier has more strict threshold (big value of θ) than class-specific verifiers since there is a possibility that the sample goes to the pairwise verifier does not belong to either candidate class.

Figure 24 gives the samples misclassified by the General Purpose Recognizer and negated or corrected by Verifier. The results demonstrate the ability of the verifier to detect erroneous conclusions. Negation of some very ambiguous samples implies the verifier's ability to deal with garbage samples.

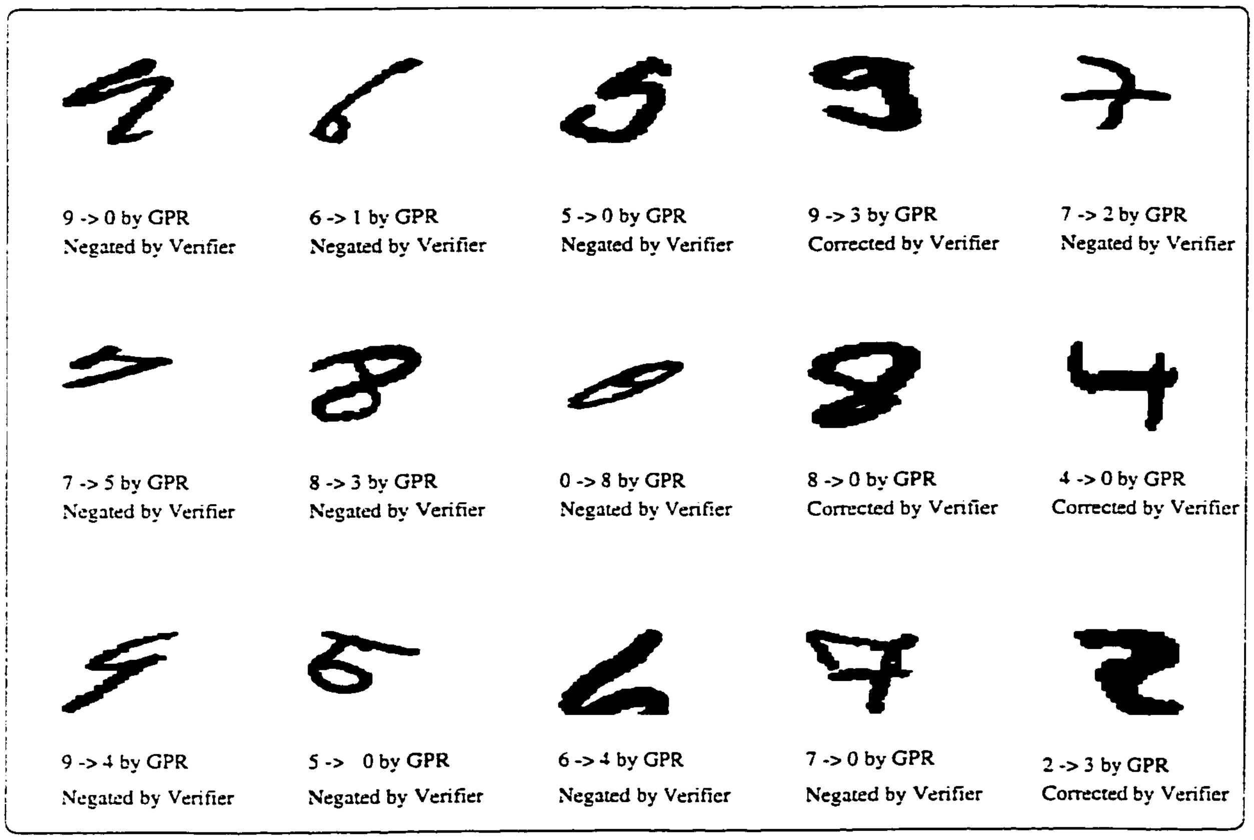


Figure 24: Confusing Samples in MNIST Dataset Negated or Corrected by Verifier

The test results on standard databases are given in Table 21. We compare the results with the performance of the most reliable systems in Tables 1 and 2. Recall that a good system should have a high reliability with a reasonable recognition rate. We can see that our R&V system is among the best three when both factors are considered.

Besides building a high performance system, this work focuses on the exploration

Database	Correct(%)	Substitution(%)	Rejection(%)	Precision(%)
CENPARMI	94.15	1.46	4.39	98.52
MNIST	95.05	0.92	4.03	99.11

Table 21: Test Results of Verifier for UIHN

of the role of verifier in an OCR system. We thus list in Table 22 the performances of the individual classifier, General Purpose Recognizer and Verifier. We can see that the performance of GPR is acceptable as a well performing recognizer. However, our goal is to attain a high precision rate of a system to satisfy situations demanding high reliability. This objective is achieved by applying the verifier which increases the precision rate of the system by its inherent characteristics and the class-specific design of the system. The results show that our verifier is capable of attaining very high precision rates without sacrificing the recognition rate much.

Further to an analysis by performance figures, the next section will use hypothesis testing to evaluate the effectiveness of the verifier from a statistical point of view.

Database	Method	Correct(%)	Substitution(%)	Rejection(%)	Precision(%)
CENPARMI	BPNet	93.56	6.44	0.0	93.74
	LeNet	95.69	4.31	0.0	95.43
	RBFNet	91.80	8.20	0.0	91.32
	GPR	94.83	3.32	1.85	96.66
	Verifier	94.15	1.46	4.39	98.52
MNIST	BPNet	94.95	5.05	0.0	94.93
	LeNet	96.80	3.20	0.0	96.94
	RBFNet	94.46	5.54	0.0	94.57
	GPR	95.97	2.10	1.93	97.89
	Verifier	95.05	0.92	4.03	99.11

Table 22: Test Results for UIHN

4.5 Evaluation of Recognition and Verification System for UIHN

As explained in Chapter 2, the statistical hypothesis testing is used to evaluate the effectiveness of the verifier.

We consider hypothesis testing a more meaningful and powerful evaluation tool than a simple two percentage figure comparison, because it views the test results as samples from a distribution. As a mature testing philosophy, it is rich in mathematical structure which takes deviations, sample size and result confidence into theoretical consideration. The adoption of hypothesis testing can partially solve the question whether a single result figure can be representative of the algorithm.

Let A be a random variable that represents the precision rates obtaining from a system without verification module, let B represent the precision rates from a system with verification module. We randomly selected 20 sets of samples from the test database of UIHN (MNIST) so that the number of the two sets for hypothesis testing are $n_1 = n_2 = 20$. On each set, we get a precision rate as presented in Table 23.

From Table 23, we can get sample averages of A & B as $\overline{X_1} = 97.84$, $\overline{X_2} = 99.07$ respectively. We then compute sample standard deviations S_1 and S_2 of A & B as square root of $E(X - E(X))^2$, where X is A or B. We get $S_1 = 0.63$ and $S_2 = 0.49$, respectively. With the definition

$$S^2 = \frac{n_1 S_1^2 + n_2 S_2^2}{n_1 + n_2 - 2}$$

we get $S = 0.58$.

The test starts by setting the hypotheses:

Step 1: Set H_0: Systems with or without verifier have no significant differences on precision rates. H_1: Systems with verifier have significant improvement on precision rates.

Step 2: Let $N(\mu_1, \sigma_1^2)$ and $N(\mu_2, \sigma_2^2)$ be the distributions of A and B, respectively. Assume $\sigma_1^2 = \sigma_2^2$ (value unknown). The populations are assumed to be independent.

Dataset	Precision Rate for A (%)	Precision Rate for B(%)
1	99.01	99.58
2	98.06	98.87
3	96.81	98.91
4	97.03	98.43
5	98.05	99.58
6	98.31	99.61
7	98.37	99.15
8	98.07	98.78
9	97.31	98.81
10	97.77	99.01
11	98.13	99.18
12	96.82	97.70
13	97.00	98.36
14	97.33	99.21
15	98.19	99.65
16	97.35	98.91
17	97.78	99.62
18	98.96	99.56
19	98.28	98.95
20	98.24	99.35

Table 23: Data Sets for Hypothesis Testing to Evaluate Verifier for UIHN

Thus $H_0 : \mu_1 = \mu_2$. $H_1 : \mu_1 > \mu_2$.

Step 3: Use $t - test$. The appropriate statistic under H_0 is,

$$\frac{(\overline{X_1} - \overline{X_2}) - (\mu_1 - \mu_2)}{S(\frac{1}{n_1} + \frac{1}{n_2})^{1/2}} = 7.24$$

Step 4: Test the hypothesis at 0.01 significance level. $\alpha = 0.01$, $t_{n_1+n_2-2,\alpha} = t_{38,0.01}$.

Step 5: Compute $t_{38,0.01}$. Since $38 > 30$, we get a good normal approximation. By looking into a standard normal table, corresponding to $\alpha = 0.01$, we have

$$P\{Z \geq 2.33\} = 0.01$$

.

Step 6: Since $7.24 > 2.33$, the observed value of Z lies in the critical region, hence we reject $H_0 : \mu_1 = \mu_2$ in favor of $H_1 : \mu_1 > \mu_2$.

By hypothesis testing, we get the conclusion that the R&V system for UIHN has significantly improved the precision rates, thus our proposed verification scheme properly suits our definition of a good verifier.

Chapter 5

General Purpose Recognizer for Unconstrained Touching Handwritten Numerals

"Data! Data! Data!", he cried impatiently. "I can't make bricks without clay."

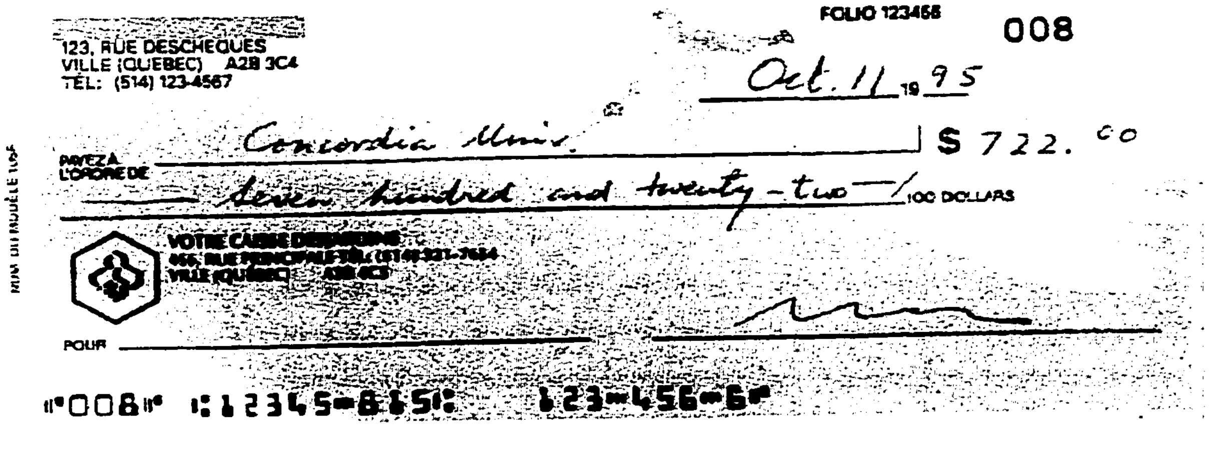
— Sherlock Holmes, The Adventure of the Copper Beeches

Before tackling the problem of UTHN, we will first discuss the practical problem of automatic processing of financial documents, which pushes the UTHN recognition to the front stage. Lack of benchmarking data, a serious obstacle in research work for on UTHN, is solved by two newly built databases, with the intention to serve as standard databases for upcomers in this field. Building of the databases is described in Section 5.2. A GPR for UTHN is then presented in the following sections. It is a novel graph-based combination of segmentation-based and segmentation-free approach. Experimental results based on the databases are also given.

5.1 Problem Background

In an information society of today, an immense number of business and financial documents are being processed, including bank cheques, payment slips, tax forms, maps etc.. Automation of these processes with document analysis and recognition technology is in strong demand. Take processing of bank cheques as an example. According to recent survey in [95], over 55 billion cheques are processed annually in North America, at a cost of 25 billion dollars, where the amounts are mostly keyed in manually. Recent developments in the recognition technology have allowed the development of automatic processing systems [18, 93]. However, as we will analyze, many challenges still hinder the emergence of successful systems.

A typical bank cheque is given in Figure 25. The textual format of the amount is called the *legal amount*, while the numerical format is called the *courtesy amount*. Automatic cheque processing is a multi-faceted subject that integrates many modules such as item extraction, courtesy amount recognition, legal amount recognition, amount validation, and in some cases date zone recognition and signature verification. Among these tasks, courtesy amount recognition is the problem of recognizing unconstrained handwritten numeral strings. Data zone recognition also utilizes numeral recognition methods.



FOLIO 123468 008

123, RUE DESCHEQUES
VILLE (QUEBEC) A2B 3C4
TÉL: (514) 123-4567

Oct. 11 19 95

PAYEZ À L'ORDRE DE Concordia Univ. $ 722. co

seven hundred and twenty-two — /100 DOLLARS

VOTRE CAISSE [illegible]
466, RUE PRINCIPALE TÉL: [illegible]
VILLE (QUÉBEC) [illegible]

POUR

[illegible]

⑈008⑈ ⑆12345-815⑆ 123-456-6⑈

Figure 25: A Typical Bank Cheque

Research work of recognizing handwritten numeral strings is naturally focused on touching cases, which is the core problem of string recognition [32, 89, 102]. Here we refer to the problem as the recognition of Unconstrained Touching Handwritten Numeral (UTHN).

Compared with the recognition of UIHN, literature indicates that the problem of UTHN is less studied and more difficult. However, it has been gaining attention of researchers due to its important role in practical systems. We can see from Tables 3 and 4 in Chapter 1 that the performances of UTHN recognizers are still mediocre, especially when the practical systems are concerned. This situation is related to many intrinsic and application specific reasons. In this thesis, our focus is UTHN recognition in cheque or financial document processing systems. Some practical problems arising in this domain are

1. In practical systems, the preprocessing stage may interfere with the subsequent recognition. Remaining noise, broken strokes or incomplete string images may present major problems to the recognizers.

2. The recognition problem of UTHN extracted from cheques and financial documents differs from other domain problems such as zip code recognition. Unlimited amount, free style touching "00" in the cents part (Figure 26(a-b)) and unforeseen garbage symbols (Figure 26(c-d)) all contribute to the low reliability of the system. Moreover, since the systems are typically used in financial environments such as banks, revenue departments or other financial institutions, the reliability demands of users is higher.

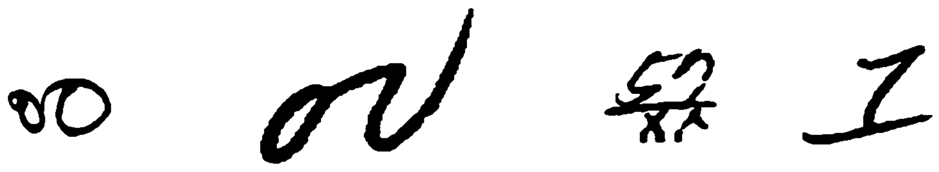

(a) (b) (c) (d)

Figure 26: Touching Numerals in Courtesy Amount of Cheques

3. Last but not least, up to now, there is no public-available standard database for the recognition of UTHN extracted from cheques and financial documents.

Lack of a standard database for recognizing UTHN is the first challenge we met in our work.

Although CEDAR CD-ROM database contains some images of touching numerals (provided as mixture with isolated numerals), they are collected from zip codes instead of financial domains. The size of such sets is at the level of several hundreds which is small.

Constructing good data sets for recognizing UTHN with financial document background thus became a necessity. We need standard databases for UTHN from financial domains, as well as a bigger size for training and testing. In the next section, we will describe the construction of the databases toward these two goals.

With the intention to make them standard training and testing sets, we consider building databases an important contribution toward benchmarking and evaluating methods of recognizing UTHN.

5.2 Building Databases

5.2.1 IRIS-Bell'98 for UTHN

The IRIS-Bell'98 database is collected and established by CENPARMI.

IRIS-Bell'98 actually consists of 2 databases referred as IRIS-Cheque database and Bell'98 database. The IRIS-cheque includes samples of Canadian personal cheques written by employees/students of Concordia University and employees of Bell Canada, financially supported by the Institute for Robotics and Intelligent Systems (IRIS) in Canada. Bell'98 includes samples of phone bills written by the general public. Consequently, all the data in IRIS-Bell'98 have real-life financial document backgrounds.

The document is first scanned with a resolution of 300 dots per inch, then the area of courtesy amount is extracted. Isolated and touching numerals are distinguished automatically or manually. They are tagged and stored as Cenparmi Binary Code (cbc) format.

The current IRIS-Bell'98 for UTHN contains 4059 touching numeral strings, among which 3731 are touching numeral pairs, i.e., 92% of touching numeral strings are actually touching numeral pairs. It is a shared phenomenon with other domains[102] that most UTHN are touching pairs. Considering that a segmentation based method can solve the touching numeral strings of 3 or more numerals by recursively adopting the algorithm for touching pairs, we will focus on the problem of recognizing touching numeral pairs in the following sections.

IRIS-Bell'98 for UTHN contains a training set of 2538 touching numeral pairs and a test set of 1193 pairs. One obvious feature of this database is its very free style. Although small, the data set represents some difficulties of real situation for financial document processing system.

To have an idea of the characteristics of this database, we analysed different touching situations of IRIS-Bell'98. We categorized the situation as:

1. Single Point Touching: The simplest touching case that two numerals have only one touching point between the numeral strokes. No ligature is involved.

2. Ligature Touching: The touching is caused by extra ligature stroke between two numerals. After segmentation,the subimage may contain unwanted noises produced by the ligature.

3. Ligature Overlap: The ligature not only touches a neighbouring numeral but also overlaps with it.

4. Multiple Point Touching: The number of touching points ≥ 2.

5. Overlap: Two numerals overlap each other. A segmented subimage may contain part of another numeral.

6. Noise: Useless (usually harmful) pattern in the numeral image caused by extra symbols or poor writing conditions and styles.

7. Broken: Fragmented images which may confuse the segmentor and some classifiers. Can be caused by writing condition, scanning resolution or binarization process.

Table 24 presents frequencies and examples of touching situation in IRIS-Bell'98 test set. High percentages of ligature touching, ligature overlapping and noise introduce the difficulties of this database. It includes problems that a recognizer may meet in real situation of processing financial documents. Nevertheless, the size of this data set is considered to be small, which will be compensated by the data set for UTHN extracted from NIST database.

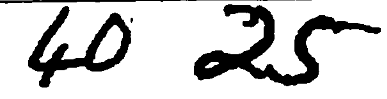
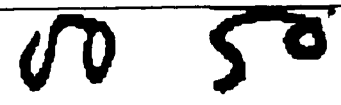
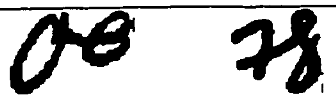
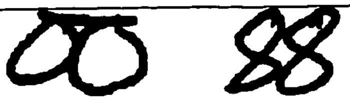
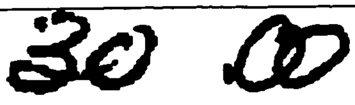
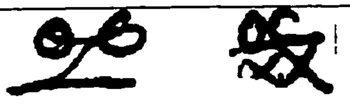
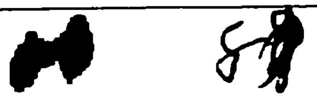
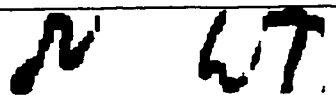
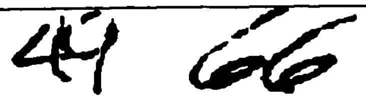
Touching Types	Frequency	Examples
Single Point Touching	50%	
Ligature Touching	23%	
Ligature Overlap	11%	
Multiple Point Touching	2.5%	
Overlap	1.4%	
Noise_1 (underline + "xx")	12.6%	
Noise_2 (ink or others)	4.0%	
Cursive	2.4%	
Broken	1.4%	

Table 24: Analysis of Touching Situations in IRIS-Bell'98 Test Set

5.2.2 NIST for UTHN

The NIST for UTHN database is built on NIST CD ROM Special Database 19. SD 19 contains the full page of 3699 binary images of handwritten data. It represents NIST's most comprehensive and probably final release of class referenced images for OCR problem. It contains all the data of Special Database 3 and 7 (SD7 is also known as TD1 in literature) which it supersedes. The form page contains 34 fields: name and date entries, a city/state field, 28 digit fields, one upper-case field, one lower-case field and an unconstrained text paragraph. They are filled by Census field workers or high school students and scanned with a resolution of 300 dots per inch.

In SD19, isolated numeral and upper or lower case characters are stored in a well organized way according to the writer or the class (See details in [30]). As we mentioned in Chapter 3, the database of isolated numerals has been used in our work for recognizing UIHN. However, when the problem comes to UTHN, no organized database exists. With NIST SD19 as a good resource, we decided to build a standard database for the research work of UTHN. Considering the building of the database a necessity of our own work, we also believe that it will benefit other researchers in this field.

The work was separated into two stages:

1. Extract touching numerals from form images.

2. Tag the touching numerals with their identities and organize them as databases.

We summarize these two steps in this section. More details can be found in [15], [109] and Appendix C.

Extracting Touching Numerals

Extracting touching numerals is a semi-automatic process. First of all, the numeric fields are extracted from the form image making use of the surrounding boxes. Given an input numeral string, we detected all the connected components and then fed each component into the recognizer for UIHN. If it is rejected by the recognizer, the number of connected components (NCC) is compared with the known number of digits (ND) of the string. If NCC is smaller than ND, this component is considered as a sample of touching numerals. Otherwise, it is considered as *garbage*.

The garbage directory is manually checked afterwards to get the missing touching numerals, which is possible when the images are fragmented. Fortunately, this situation does not occur often. The frequency is lower than 1%.

The manual check of extracted samples is done at the next stage: tagging.

Tagging Touching Numerals

Tagging touching numerals is also a semi-automatic process. The extracted touching numerals are fed into a preliminary recognizer for UTHN, the resulting labels are saved in a file. Another program binds the image and the corresponding label into cbc (Cenparmi Binary Code) format which puts many images into a single file.

The tagged cbc files are manually checked afterwards. Errors are corrected and rebound into the database.

Touching numerals are categorized according to the number of their numerals. At the time of writing the thesis, the extracting and tagging of touching numeral pairs have been finished, which represents the majority of all touching strings.

The database is separated into training and testing samples according to the original partition of NIST SD 19. Consequently, NIST for UTHN contains a training set of 4252 touching numeral pairs and a test set of 4395 touching numeral pairs. The phenomenon that testing partition contains more touching cases also confirms the observation we mentioned in Chapter 3 that the test set of NIST is more difficult than the training set.

We can see that the size of NIST for UTHN is much bigger than IRIS-Bell'98, which is a good feature for a benchmarking database. On other other hand, the percentage of difficult situations such as garbage symbol and cursive writings are smaller than IRIS-Bell'98. Table 25 shows the result of touching situation analysis on the test set of NIST for UTHN.

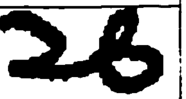
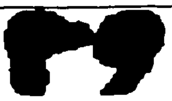
Touching Types	Frequency	Examples
Single Point Touching	75%	
Ligature Touching	9.4%	
Ligature Overlap	2.5%	
Multiple Point Touching	7.5%	
Overlap	1%	
Noise and Cursive	4.4%	

Table 25: Analysis of Touching Situations in NIST for UTHN Test Set

The establishment of the above two databases for UTHN are original and important work for the research on touching numerals. The recognition and verification of UTHN in this thesis are conducted on these two data sets.

5.3 Segmentation

As we discussed in Chapter 1, the recognition of UTHN can be categorized as segmentation-based, segmentation-free or holistic. What we used to tackle the touching numerals is a graph-based combination of segmentation-based and segmentation-free methods. This scheme solves slanting and some overlapping problems by contour tracing in the segmentation stage. It also ensures efficient computation by a filtering stage before recognition.

In this section, we will present the segmentation-based module, which provides the candidates of cutting points. The method was proposed in Strathy et al. [92]

with minor modification of the author of this thesis. Here for completeness we give a brief description of it.

The segmentation algorithm is based on contour features. It consists of four steps:

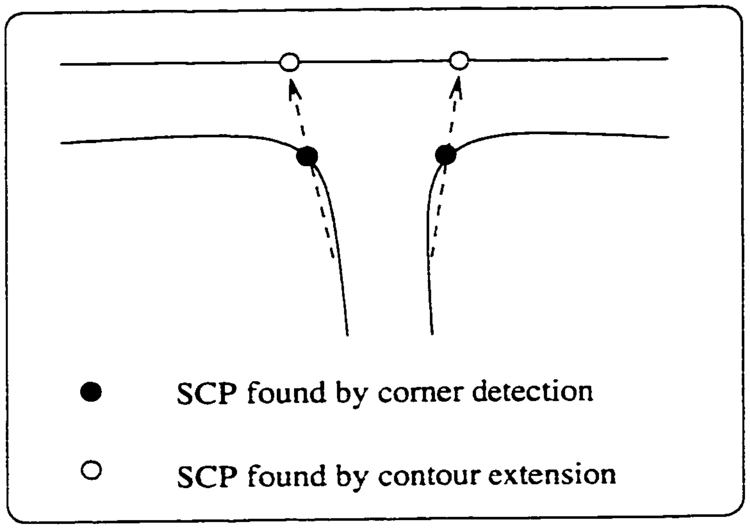
1. Preprocessing to remove noise and normalize height.
2. Contour analysis to determine significant contour points (SCPs).
3. Sorting of SCP pairs.
4. Guiding the cut from the entry SCP to the exit SCP.

Simple preprocessing is first done to smooth the edges and get the contour information. A set P of significant contour points (SCPs) is then created which contains corner points in mountain, valley and open regions such as:

- Minimum point of each valley.
- Maximum point of each mountain.
- Exit points of the imaginary straight lines formed by, wherever possible, extending the contour through the stroke at concave corners in mountains and valleys (Figure 27).

To get the pair of SCP between which to pass the cutting path, we sort the SCPs in P according to 9 measures of each pair:

1. A fixed number of credits for each mountain/valley pair;
2. A fixed number of credits for each SCP that was found by corner detection;
3. Credits for the proximity to each other of the points in the pair;
4. Credits for the sharpness of the concavity at each of the SCPs.
5. Credits for the proximity, where applicable, of an SCP's valley (mountain) to the top (bottom) of the image;



● SCP found by corner detection
○ SCP found by contour extension

Figure 27: Get SCP by Extending Contours Through a Stroke (From [92])

6. Credits, where applicable, for the distance of an SCP from the bottom (top) of its mountain(valley);
7. Credits, where applicable, for the degree to which a valley corner is above a mountain corner in the image;
8. Credits for the degree to which stroke pixels outnumber background pixels in an imaginary straight line drawn between the pair;
9. Credits for the nearness of the pair to the left hand side of the image.

Credit values, including fixed ones, are normalized according to the height of the bounding box of the given numeral string. After the score for each pair has been computed the pairs are sorted from the highest score to the lowest with, the more favourable cuts appearing towards the front of the list.

Once the cutting path has been determined, we must split the single connected stroke component into two new components. We trace the cutting path and the outer contour of the numeral string onto a blank image, then follow the lefthand region of the contour chain counterclockwise and copy the contents of each corresponding row from the original image into the left image. The right hand component is extracted in a similar fashion.

The method in [92] has some limitations, one of which is that it discards small components when getting the two new subimages. Consequently, some fragmented '5's that have a separate above line will be incomplete. The problem is solved in this thesis work by considering all components and pasting them back to either left or right image according to their relative positions.

5.4 Candidate Filtering

After segmentation described in the previous section, we get a list of possible cuts in a sorted sequence. On the average, the cardinality of P is around 20, which is the number of possible split paths.

In a classic over-segmentation method, each segmentation candidate will be applied. The lefthand and righthand images are fed into a single-digit recognizer. If the recognition is successful, e.g., a high confidence level is achieved, the result will be returned [57][7].

The obvious disadvantage of such a scheme is the intensive computation. Alternative methods such as "split-and-merge" [32] at the stroke level also entail high computational complexity. On the other hand, simply taking only the first several split candidates may miss some proper cuts.

A candidate filter is thus introduced after our segmentation phase. After candidate filtering, 3 criteria must be fulfilled:

1. The size of segmentation candidate set is significantly minimized.

2. High probability of keeping the most appropriate candidate within the new set.

3. Moderate computation complexity.

The filtering rule is described in Table 26.

The underlying hypothesis in the filtering rule is that the most appropriate cut point for a touching numeral pair occurs around the middle zone of the image, except

```
For each segmentation candidate $sc$,
    If $|BW_{left} - BW_{right}| < T_{split}$, Then
        Put $sc$ in the new set
    Else if $BW_{left} - BW_{right} > T_{split}$ and lefthand image is recognized as '2' or '5'
        Put $sc$ in the new set
    Else if $BW_{right} - BW_{left} > T_{split}$ and righthand image is recognized as '2' or '5'
        Put $sc$ in the new set
    Else if $BW_{left} - BW_{right} > T_{split}$ and lefthand image is recognized as '1'
        Put $sc$ in the new set
    Else if $BW_{right} - BW_{left} > T_{split}$ and righthand image is recognized as '1'
        Put $sc$ in the new set
    End
End
```

Table 26: Filtering Rule (BW_{left} denotes the width of lefthand component, T_{split} is a pre-defined threshold.)

images with '2','5' and '1'. A statistical method to test and prove the assumption is described in Appendix D.

After filtering, the average number of segmentation candidates drops to 8. The candidates are still kept in a sorted sequence provided by the segmentation method and is ready for further recognition and verification.

5.5 Segmentation Recognition Cost Graph and Double Zero Problem

In this section a graph-based segmentation-recognition scheme is introduced with reference to the set of segmentation candidates from the previous section. A holistic solution for touching double zero is then described due to its high frequency in IRIS-Bell'98 database.

Segmentation Recognition Cost Graph

Segmentation Recognition Cost Graph (SRCG) is a scheme to combine segmentation module and dual-recognizer into an overall consideration. It is a novel way to combine segmentation-based and recognition-based approaches by converting many *hard constraints* into *soft decisions* in the form of costs.

Figure 28 illustrates the recognition of a touching digit pair using SRCG. Results and confidence values of the segmentation module and recognizers are put on the arches of the graph. The path from "start" to "end" with the highest combined value (lowest cost) is selected to give the best result.

To get normalized score values, the segmentation candidates are rescored according to heuristic Gaussian like formula:

$$S_i = e^{-\frac{(ai)^2}{b}} \qquad (30)$$

where i=1,2,.., are numbers of available candidates; a=0.4, b=10.

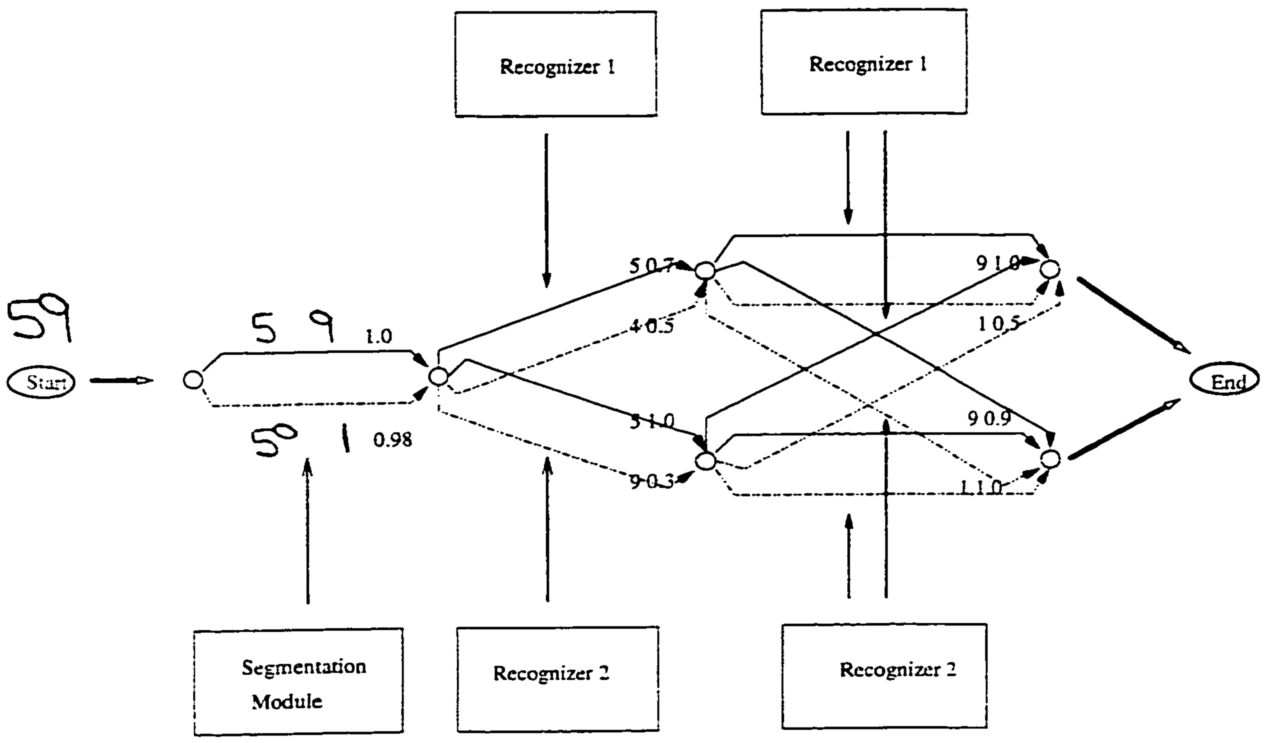


Figure 28: Segmentation Recognition Cost Graph

The two recognizers used in SRCG are the neural approaches described in Chapter 3 and another recognizer used in CENPARMI[93]. The scores for these recognizers are the confidence values of the output. In our recognizer, the confidence value is an average of the three normalized neural network outputs.

In the illustrative example of Figure 28, a touching "59" goes through the segmentor. After segmentation candidate filtering, we got 2 possible candidate cuts. We fed the left and right subimage of each possible segmentation into two recognizers and got 4 decisions with confidence values on the graph edges. Only the lines with the same style (solid or dotted) can be a possible combination. We can obtain 8 possible paths. The highest is the conclusion of '5' and '9' with score 1.0 for segmentation, score 1.0 for recognizer two for '5', and score 1.0 for recognizer one for '9', which give a total sum of 3.0.

Figures 29 and 30 are two other examples of the way the SRCG recognizes touching numerals. Figure 29 shows successful recognition of touching '78'. Figure 29 is an example that contains a confusing numeral '9'. The scores of different recognition decision are listed in the figures, while the scores of segmentation for simplicity are not shown.

Concern of Double Zero

Before giving out the performance of our GPR on the built databases, a remaining problem which needs particular notice is the frequency and poor quality of the double zero in IRIS-Bell'98 database.

There is a total of 600 touching "00"s in IRIS-Bell'98 test set, whose total size is 1193. Moreover, most of these touching double zeros are cursively written. Ligature and overlapping are common, and cause serious difficulty in segmentation. To tackle this problem, we developed a holistic recognizer using QNN neural network. This network is dedicated to improve the recognition rate of "00" sequence in IRIS-Bell'98 database.

The training set of this network is extracted from MNIST database. 267 samples of "00" are fed with the output (1,0), 400 other patterns are fed with the desired value (0,1). The training parameters are set as follows:

- The quantum level ns is set to 3;
- The learning rate is set to $\eta = 0.7$.
- The momentum of training is set to $\alpha = 0.2$.
- The number of hidden nodes are set to 20.

The network is applied after SRCG for IRIS-Bell'98 database, under the condition that the conclusion from GPR contains at least one zero. The threshold of this recognizer is set to 0.5, which is relatively big, to prevent a decrease in the precision rate of "00". The application of this recognizer increases about 10 percentage the recognition rate of "00". However, the recognition rate, as we will report in next

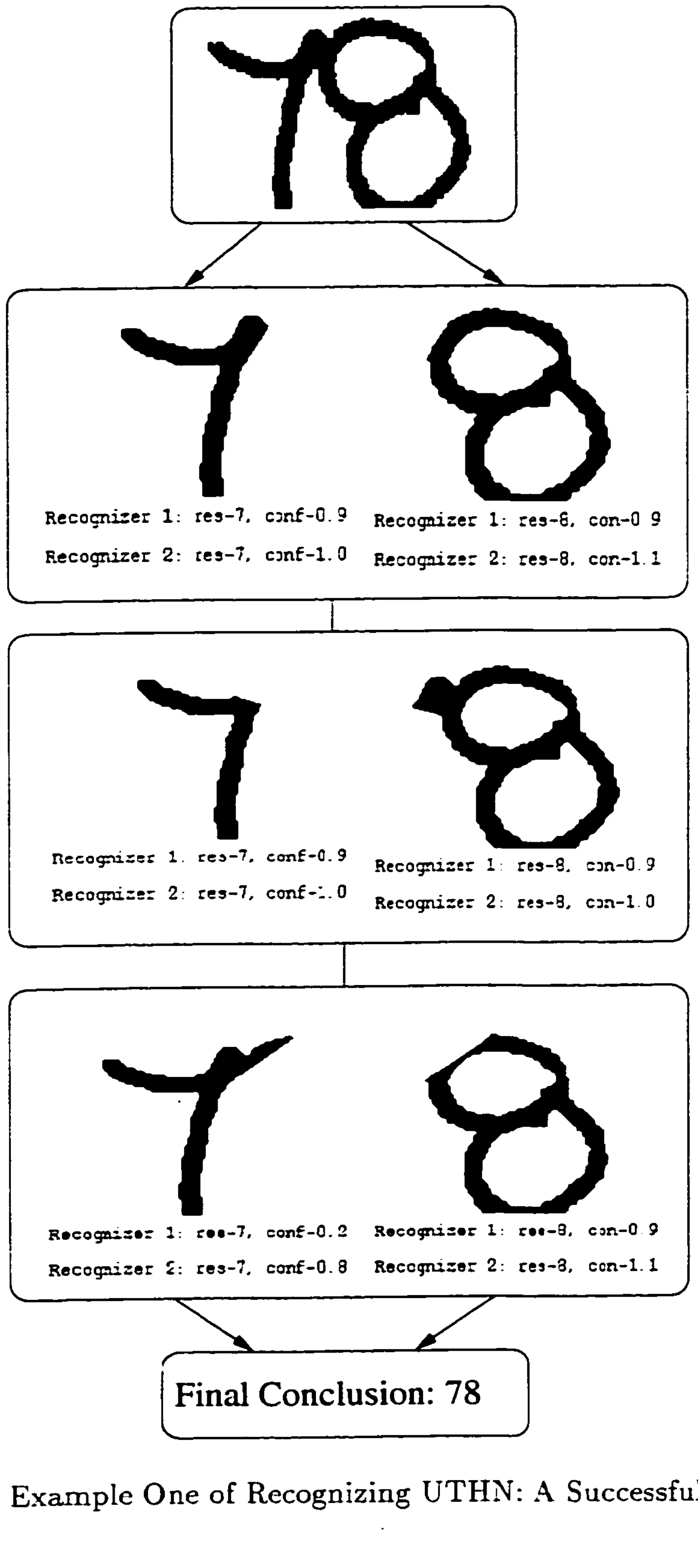

Recognizer 1: res-7, conf-0.9
Recognizer 2: res-7, conf-1.0
Recognizer 1: res-8, con-0.9
Recognizer 2: res-8, con-1.1
Recognizer 1: res-7, conf-0.9
Recognizer 2: res-7, conf-1.0
Recognizer 1: res-8, con-0.9
Recognizer 2: res-8, con-1.0
Recognizer 1: res-7, conf-0.2
Recognizer 2: res-7, conf-0.8
Recognizer 1: res-8, con-0.9
Recognizer 2: res-8, con-1.1
Final Conclusion: 78

Figure 29: Example One of Recognizing UTHN: A Successful Example

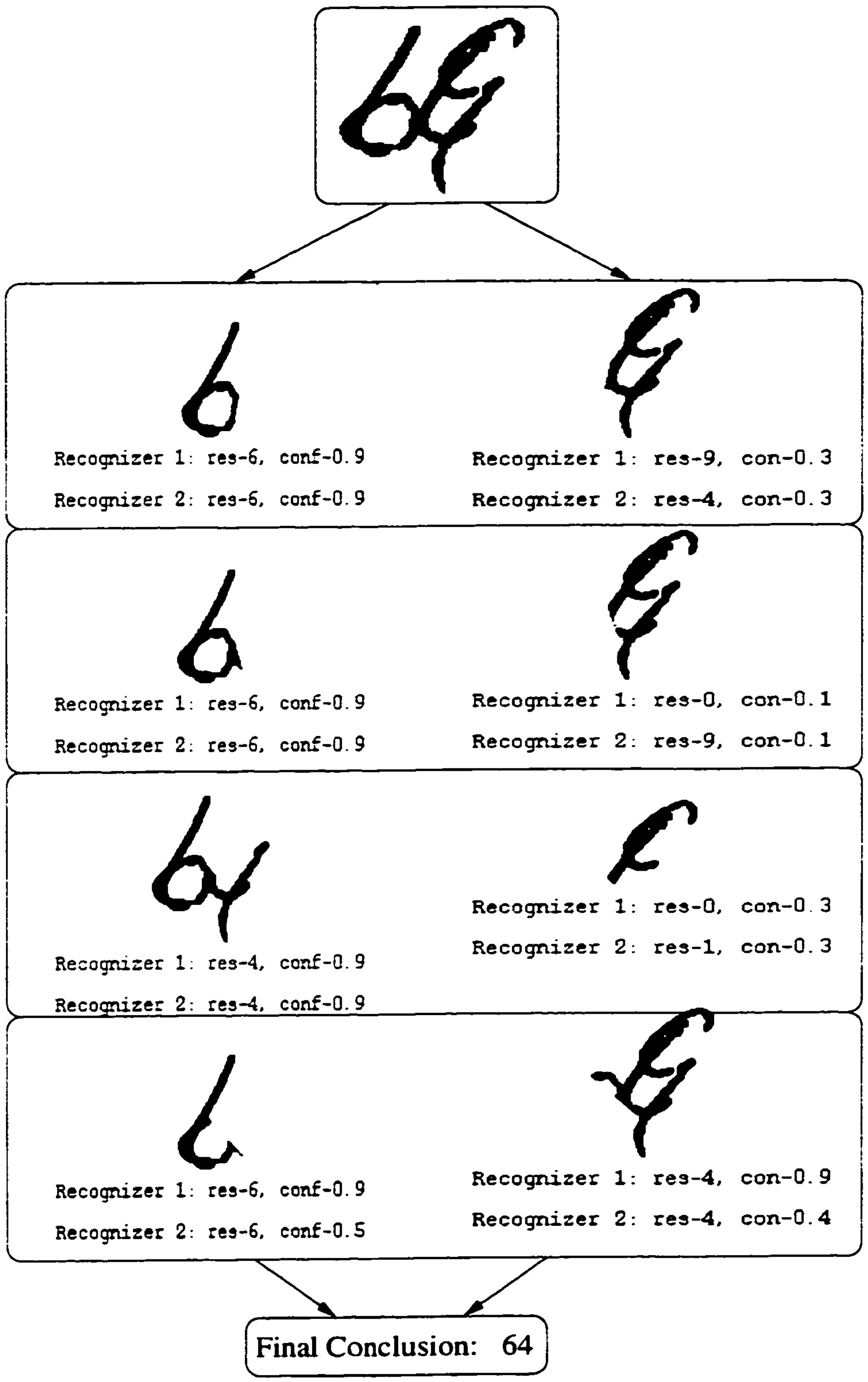

Recognizer 1: res-6, conf-0.9
Recognizer 2: res-6, conf-0.9
Recognizer 1: res-9, con-0.3
Recognizer 2: res-4, con-0.3
Recognizer 1: res-6, conf-0.9
Recognizer 2: res-6, conf-0.9
Recognizer 1: res-0, con-0.1
Recognizer 2: res-9, con-0.1
Recognizer 1: res-4, conf-0.9
Recognizer 2: res-4, conf-0.9
Recognizer 1: res-0, con-0.3
Recognizer 2: res-1, con-0.3
Recognizer 1: res-6, conf-0.9
Recognizer 2: res-6, conf-0.5
Recognizer 1: res-4, con-0.9
Recognizer 2: res-4, con-0.4
Final Conclusion: 64

Figure 30: Example Two of Recognizing UTHN: A Confusing Example

section, is still low. Fortunately, the cents part in financial documents is usually less important, which may also explain that they are written in such a careless way.

Up to now, we have our General Purpose Recognizer handy for UTHN problem. For NIST database, it is actually the SRCG.

The system rejects a decision when both of the following conditions are satisfied: (1) two top scores with different conclusions are smaller than a threshold; (2) for the best path, the difference between two recognizers gives different conclusions.

The first condition is easy to understand. However, in practice it is possible that two ways of segmentation yield two highly confidence conclusions. Although the difference between two scores is not big, the top one is actually correct. The probability of a wrong segmentation increases if for a single path there is an opinion discreteness between the two recognizers. On the other hand, due to different characteristics of the two recognizers, only considering the same opinion of both engines will decrease the recognition rate too much. Thus the combination of the two rules are used as the rejection criterion for the GPR.

5.6 Overall Performance of General Purpose Recognizer for UTHN

Using the proposed GPR proposed in the previous section, we get the performance presented in Table 30, based on IRIS-Bell'98 and NIST test sets for touching numeral pairs. Tables 28 and 29 list the detailed correct rate of each numeral pair.

We can see that the rates on IRIS-Bell'98 is lower than what we get from NIST, which is consistent with the analysis of the two datasets in Section 5.2. The distribution of the numeral pairs in IRIS-Bell'98 is very unbalanced due to many "00" in the cents part of the courtesy amount.

By examining the frequency of occurrences of touching numeral pairs in NIST set, we can observe that the 5 most frequent touching pairs are "89", "56", "00", "50" and "20". The least frequent ones are "61", "91", "14", "41" and "71". We can see that touching occurs commonly when numeral '0' or '5' show up or '8' and '9' are together, while '1' and '7' tend to stand alone. Although we can not yet conclude that this is the general pattern of human writing, the information can be viewed as important hints for developing a R&V system for UTHN as we will see in the next chapter.

Due to lack of benchmarking data for comparison, it is not easy to compare the results directly with other algorithms for UTHN. However, the recognition results are already comparable to those reported on data from CEDAR CDROM. On NIST database, we get an error rate less than 9%, which is much better than the results of [89] and [102]. We also see that precision rates are not yet good enough to reach our high reliability goal. The errors come from both the segmentation and recognition stages. We will improve the precision rates at the verifier stage in the next chapter by overcoming the problems from both sources.

Some may argue that "the precision rate of GPR can also be improved by adopting more strict thresholding, instead of rejecting only the images that GPR is obviously unable to handle". The answer comes from the overall goal of R&V system. Recall that the very purpose of the verifier is to make *reliable* confirmation or negation. If a conclusion is irresponsibly rejected by the GPR, the verifier will never meet the question. Between the choices of applying overstrict thresholding in GPR and leaving the problem to the verifier, we are naturally prone to the latter. Inappropriate use of the power of verification will have negative impact on our overall goal, which is to ensure a high reliability while maintaining a reasonable recognition rate for the whole R&V system.

Database	Correct	Rejection	Precision for Strings	Precision for Numerals
IRIS-Bell'98	65.5%	16.3%	78.5%	84.3%
NIST	88.6%	3.1%	90.1%	94.2%

Table 27: Test Results of General Purpose Recognizer for UTHN

	occur.	correct		occur.	correct		occur.	correct		occur.	correct
00	600	365	25	9	9	50	18	14	75	6	4
01	1	1	26	4	4	51	5	3	76	8	5
02	1	1	27	5	2	52	6	5	77	3	3
03	1	1	28	6	5	53	7	5	78	7	6
04	3	1	29	6	5	54	7	4	79	12	6
05	3	3	30	13	11	55	8	7	80	12	9
06	1	1	31	2	2	56	9	8	81	1	1
07	2	1	32	5	5	57	6	2	82	2	2
08	5	2	33	2	2	58	11	9	83	4	3
09	5	4	34	3	2	59	12	9	84	3	2
10	1	1	35	6	5	60	4	3	85	4	2
11	1	1	36	5	5	61	2	2	86	10	9
12	3	3	37	3	3	62	2	2	87	3	2
13	0	0	38	5	5	63	5	5	88	15	9
14	2	1	39	7	6	64	2	0	89	13	11
15	2	2	40	6	5	65	4	3	90	3	1
16	1	0	41	4	2	66	7	5	91	0	0
17	1	1	42	3	2	67	0	0	92	1	0
18	2	1	43	2	1	68	5	5	93	1	1
19	2	1	44	5	4	69	5	4	94	0	0
20	25	21	45	6	6	70	13	10	95	2	2
21	3	2	46	5	5	71	5	4	96	3	1
22	5	4	47	1	1	72	3	2	97	2	2
23	5	5	48	4	4	73	3	2	98	8	8
24	0	0	49	7	4	74	1	1	99	3	3

Table 28: Performance of GPR for UTHN on IRIS-Bell'98

	occur.	correct		occur.	correct		occur.	correct		occur.	correct
00	162	138	25	111	106	50	161	141	75	38	35
01	8	8	26	93	88	51	26	24	76	31	29
02	26	22	27	6	6	52	44	40	77	17	12
03	17	14	28	84	77	53	39	34	78	76	70
04	39	31	29	47	40	54	91	77	79	60	52
05	33	32	30	98	88	55	56	48	80	74	62
06	41	40	31	7	6	56	202	184	81	8	7
07	18	14	32	42	35	57	46	33	82	23	19
08	36	32	33	47	44	58	104	101	83	38	37
09	56	53	34	68	61	59	112	89	84	37	32
10	3	3	35	46	43	60	62	53	85	62	59
11	2	2	36	68	62	61	1	1	86	58	50
12	5	5	37	10	8	62	17	15	87	34	22
13	10	8	38	40	35	63	27	24	88	75	66
14	1	1	39	30	28	64	25	18	89	279	255
15	14	14	40	89	81	65	58	50	90	15	14
16	2	2	41	2	2	66	33	27	91	2	1
17	4	2	42	20	18	67	5	4	92	14	12
18	7	7	43	26	21	68	28	23	93	14	11
19	6	6	44	13	13	69	30	25	94	11	10
20	135	128	45	78	71	70	46	41	95	16	13
21	33	30	46	60	55	71	3	3	96	11	5
22	42	38	47	12	11	72	3	3	97	11	9
23	99	95	48	54	51	73	24	19	98	21	18
24	32	28	49	64	56	74	34	27	99	36	28

Table 29: Performance of GPR for UTHN on NIST

Chapter 6

Verifier for Unconstrained Touching Handwritten Numerals

"Take time to consider. The smallest thing may be the most essential."

— Sherlock Holmes, The Adventure of the Red Circle

6.1 Incorporating Domain Specific Verification

In Chapter 4, theoretical requirement for the verifier has been discussed. Experiments with Quantum Neural Network further addressed the property of a good verifier. Our verification scheme based on QNN performs effectively for UIHN.

With our discrete strategy for touching numerals, it is clear that the verifier for UIHN can be applied to improve upon the results of GPR for UTHN. Although reasonable and feasible, this scheme is not enough. The reason is that GPR of UTHN suffers from specific weaknesses arising from the domain of touching numerals. Understanding and overcoming these difficulties, developing specially trained domain specific verification schemes, became the key to success of solving UTHN problem. In this chapter, we present our work on this aspect. We deal with the following domain specific problems:

1. Wrong segmentations which can be detected by Touching Type & Location verification.

2. Errors of GPR caused by ligatures and/or weakness of recognizers which can be solved by the structural the verifier making use of structural features.

3. Dummy symbols which can be detected by our domain specific dummy symbol detector.

The above schemes are addressed in the following three sections respectively. In each section, the problem is first exemplified and analysed. The corresponding approach is then proposed in detail. The architecture of Verifier for UTHN is described in Section 6.5, which combines the domain specific schemes with the verifier for UIHN. Experimental results are given on IRIS-Bell'98 and NIST database for UTHN. Finally, a statistical evaluation of R&V system for UTHN is conducted using hypothesis testing.

6.2 Touching Type & Location Verifier

As discussed in Chapter 1, the discrete solution and holistic solution for UTHN problem have their respective advantages and disadvantages. In discrete approach the proper segmentation is very important. If a wrong segmentation occurs, even a perfect recognizer won't help. Our GPR for UTHN, which considers multiple splitting possibilities with recognizer assistance in an efficient way, partially relieves the headache of pure segmentation-based algorithm. However, it is far from worry-free. In the example presented in Figure 31 the segmentor provides three choices according to the scores of SCP pairs. Due to the smooth connection of the loop in '6' and the stroke in '5', none of the three segmentation paths separates the left '6' from the '5'.

Although GPR may still recognize the right hand image as '5', it seems inevitable to get a conclusion of '1' for the left hand image. "15" is actually the conclusion of GPR for this pair of numerals. To solve this problem, we propose Touching Type & Location Verifier.

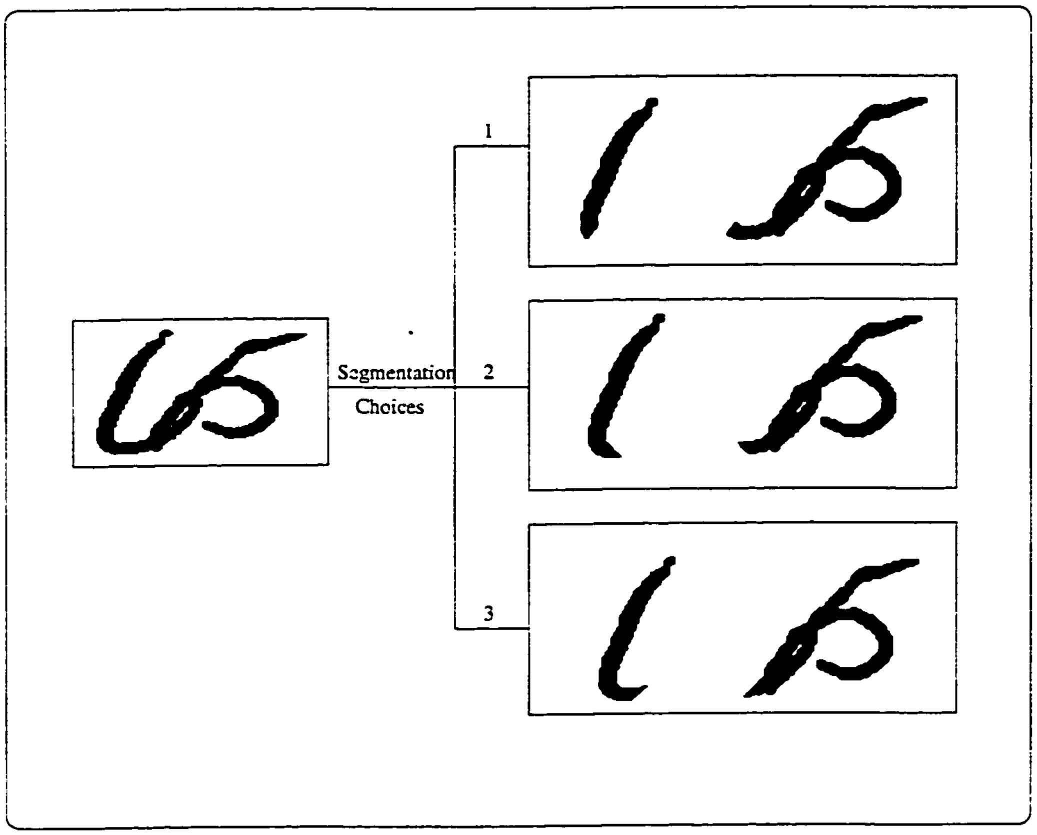


Figure 31: Wrong Segmentation of Touching '6' and '5'

Touching Type & Location Verifier is based on the possible touching types between 2 numerals and location of the touching zone. Given a segmented numeral, if its touching location and touching type are impossible to occur together, the decision of GPR is negated.

Before giving the verification details, we will first explain different kinds of touching types.

6.2.1 Touching Types and Locations

Touching types can be categorized according to the stroke type we get after segmentation. If the cutting points create an end of a stroke, we call it an End Point (EP). Otherwise, it is a Non-end Point (NEP), which can be the side of a stroke or a bend area. We define 4 touching types for a touching numeral pair, which will be used in Touching Type & Location Verification:

1. EP-NEP or NEP-EP

2. EP-EP

3. NEP-NEP

4. MULTIPLE

Figure 32 exemplifies four touching types.

Touching location is defined for a segmented numeral from touching pairs. It is divided into two kinds: uppertouching and lowertouching. If the touching occurs within the upper half part of the segmented numeral, we say it is a UpperTouching. If it occurs in the lower half, LowerTouching is the conclusion. Let Y be the coordinate in the height direction, A and B be the SCP pairs of contour pixels between which the cut path is obtained. H is the height of the numeral image. Touching location can be simply detected according to the coordinates of the touching points:

```
If $Y_A < H/2$ and $Y_B < H/2$
    Then LowerTouching;
Else if $Y_A > H/2$ and $Y_B > H/2$
    Then UpperTouching;
Otherwise
    Unknown touching location.
```

Touching location applies to a single segmented digit. It is only related to Touching Type 1-3. It is possible for two touching numerals to have different touching locations,

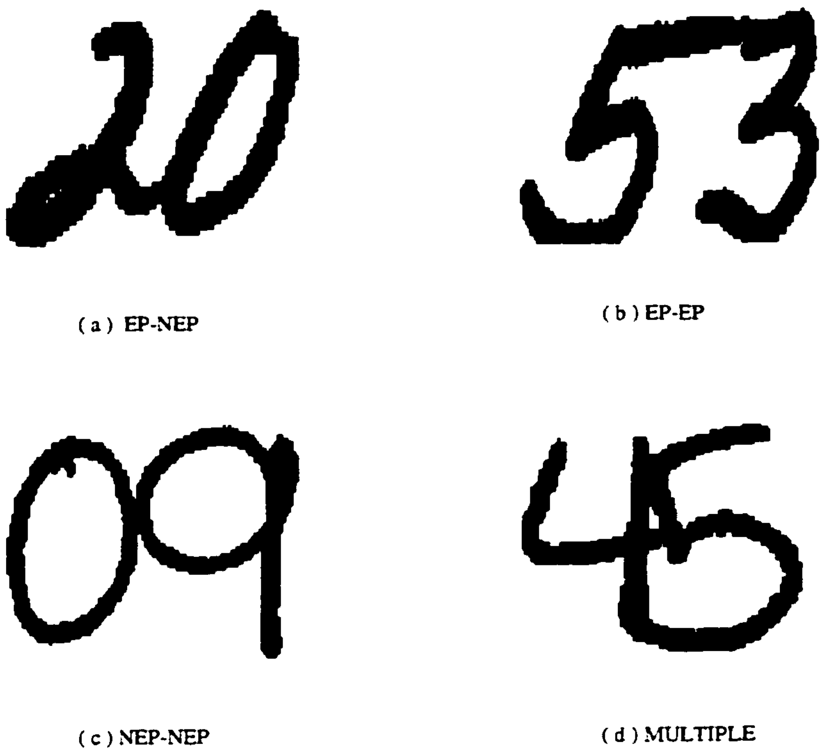


Figure 32: Touching Types and Locations Used in Verification. EP represents End Point. NEP represents Non-end Point. (a) is also an example of LowerTouching. (b)(c) are UpperTouching.

since the bounding box of left-hand and right-hand images may be changed after splitting. Figure 32 also exemplifies different touching locations.

6.2.2 Detection of Touching Types

The key point of detecting touching type is to decide whether the touching point is an end point.

There are many existing ways to accomplish this [54, 64]. We proposed a method based on contour pixels. It is designed according to the criteria of quick computation and easy implementation, with the consideration of using the similar data structures of our other algorithms for segmentation and verification. Our experiments proved the effectiveness of the algorithm. The uniqueness and problems will also be discussed in this section.

The algorithm consists of two phases.

In the first phase, the direction of every contour pixel to the next 8-connected pixel is computed using Freeman codes as in Figure 33. The values are stored as a chain along the contour.

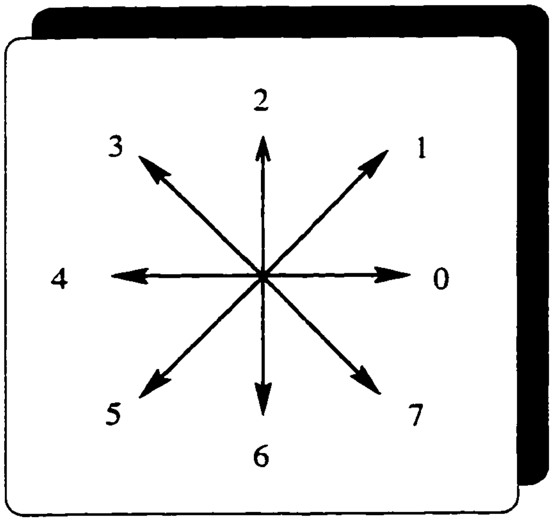


Figure 33: Freeman Codes of 8 Directions

In the second phase, after the segmentation has been done, we get a pair of SCP pixels on the contour as cutting points. We examine the chain code and relative positions of two pixels to determine whether they compose an end point of the numeral after being split from the original touching situation.

Ideally, the strokes leading to end-regions have relative stable width. Their edges are parallel, the end regions — we are actually considering an open end — are expected to have a close to 180 degree direction change, represented by a difference of 4 in Freeman codes. Let A, B be the SCP pair, the criterion of checking end-point can be summarized as:

1. $||A - B|| < 0.3 * max(width, height)$, $||\cdot||$ is the Euclidean distance.

2. $|C_A - C_B| \in [3, 5]$, C represents the Freeman code.

The algorithm is exemplified in Figure 34, which presents the chain codes of the touching "65" in Figure 31 after segmentation. Examinination of the left image discovers that the two cutting points have opposite directions represented by a code difference of '5'. The distance between the two points is also close. We can then decide that after splitting, the left image gets an end-point due to the cutting.

The algorithm has its merits of rotation invariance, scale invariance and speed. Another unique advantage of our situation is that it avoids the problems inherent in many other algorithms — to decide the locations of an extremity point or the pair points to examine the direction changes[63]. These problems can be difficult in the situation of circular end region, as explained in Figure 35.

However, there exist variabilities that cause difficulties in our task of detecting end points.

One difficulty is due to the unsmooth direction transitions on the contour sometimes. Smoothing has been done before segmentation (See details in [91]). Nevertheless, it is still possible that one cutting point does not have the same tendency of the stroke edge represented by small bumps and direction variations on the contour. Re-examining the '1' like stroke in Figure 34, we can see that although the major direction code of the left edge is 6, there are some pixels with directions 5s and 7s, and the code of the cut point is 7. This problem can be solved by averaging the codes of

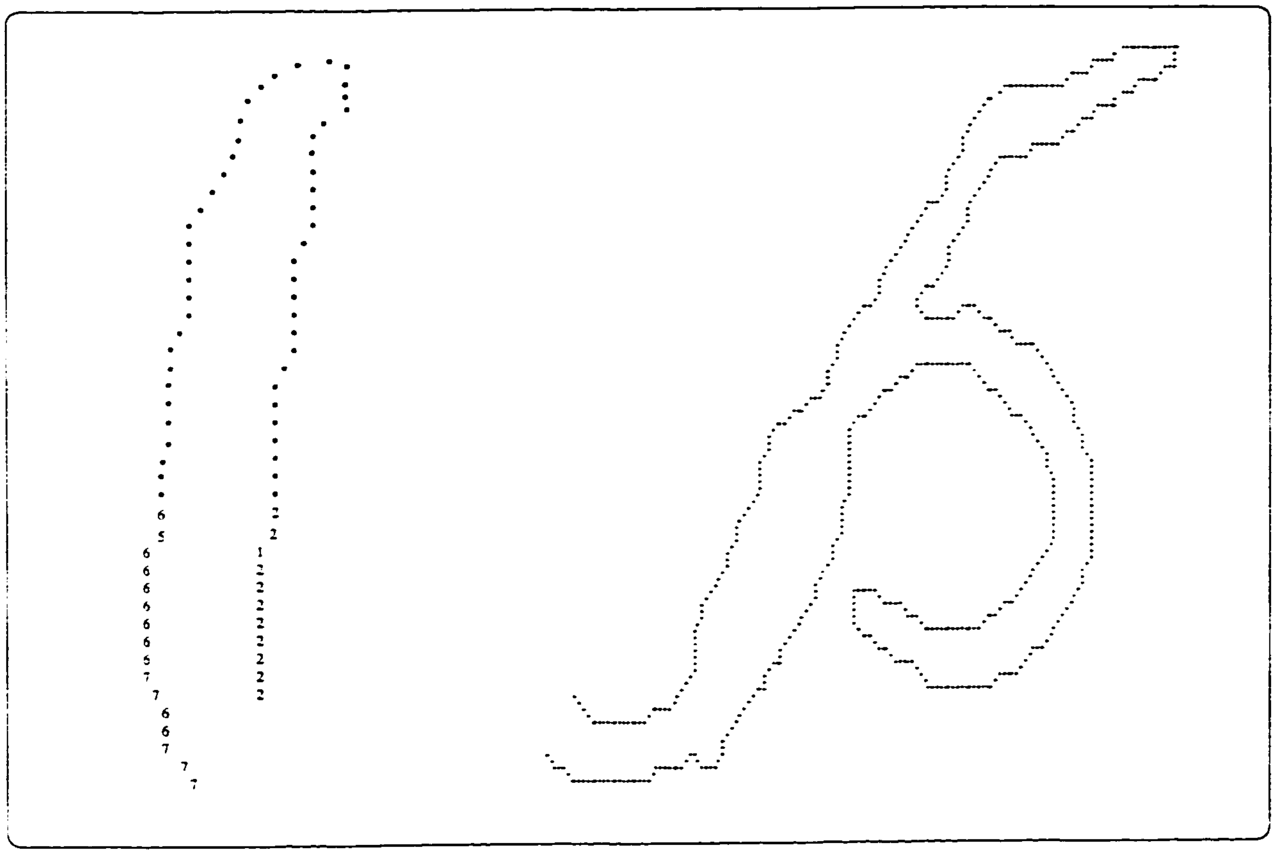


Figure 34: The Outer Contours and Freeman Codes After Segmentation of Touching '65'. Only the chain codes close to left cutting points are shown to highlight the region of interest.

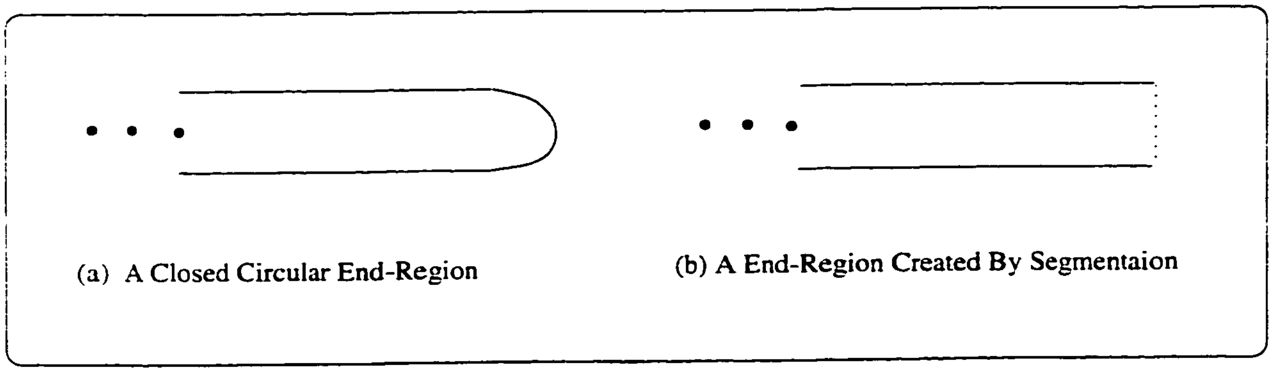

(a) A Closed Circular End-Region
(b) A End-Region Created By Segmentaion

Figure 35: Two End Regions. On the left is the circular region that may cause problems in other end-point extraction methods. On the right is our situation which is dealing with an open contour after segmentation. It implies a straight-angled end region.

several contour pixels near the cut point, which will solve most cases. However, when averaging, we have to be careful about directions 7 and 0. These two neighbouring directions will give a very different direction after being averaged. A solution is to replace 7 by -1 when they are found near 0s.

One may ask that "why not rely on skeletons for endpoint extraction?". Besides the consideration of using available data structures, a statement with similar problem background can be a brief answer:

> "Let us first recall that for the great majority of samples, none of these problems exist. Also, our work indicates that it is possible to deal with most difficulties in an efficient and logical manner. Furthermore, skeletonization is a time-consuming process which has its own shortcomings, even for endpoint detection, such as the creation of spurious branches and endpoint erosion, especially for sample with wide strokes." (See [63], page 168)

6.2.3 Touching Type & Location Verification

Touching Type & Location Verification is a negation scheme which examines the impossible combination of touching type and touching location between two numerals. Class-specific verification rules are defined for this purpose.

Let us take Figure 31 again as the illustrative example. For real numeral '1', the touching with another numeral usually occurs along the side of the stroke, i.e., it is a NEP type touching. If a segmentation creates an end point in the lower part, it is highly possible that the segmentor mistakenly cut one numeral into two and left a '1' like segment as the left image. Represented by Touching Type & Location rule for numeral '1', EP-EP touching type and LowerTouching Location will create a negation. Thus we are able to avoid the error in Figure 31.

Figure 36 gives another example of touching type verification. There are 3 candidate cutting paths. The first candidate gives out high scores in SRCG, which may conclude the string as "75". However, the conclusion of the left image has to pass the verification rules of '7', which include the impossible case of "EP-EP + Lower-Touching". This candidate conclusion is thus rejected and the chance will be given to other cutting possibilities.

The Touching Type & Location verification rules are "hand-picked" in our current system according to the samples in the training set. The targets of verification are mostly '1' and '7', while the original sample classes are '0', '2', '6" and '9'. This phenomenon matches our observation of occurrence frequency as analysed in the Chapter 5. 3.5% samples in IRIS-Bell'98 training set are affected positively by this verifier.

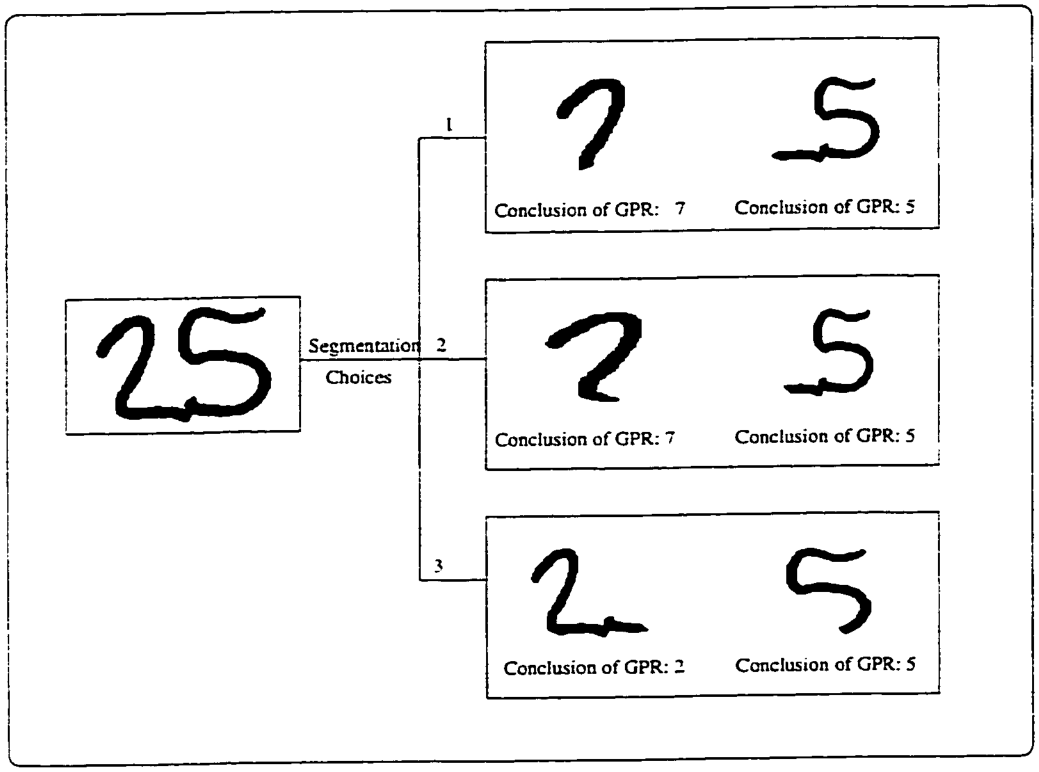


Figure 36: Example of Touching Type & Location Verification. The first two cutting possibilities are rejected by the verification rules. The final conclusion is '25'.

6.3 Structural Verifier

Touching numerals tend to have more structural variations compared with isolated numerals. These variations can be introduced by ligatures (including ligature touching and ligature overlapping, see definitions in Section 5.2.1), overlaps and some degree of cursive styles. The situation can be worsened by inappropriate segmentation.

Figure 37 gives an example when cursive style and the ligature cause some distortion of the numeral '0' in pair "80". Examination of the first segmentation possibility reveals that the recognizers used in GPR for UTHN caught only the global shape of the right hand image which is similar to '7'. Although the confidence value is not high (0.6), it is higher than the confidence value of '0' in the second segmentation possibility (0.3). It gives a final conclusion of "87".

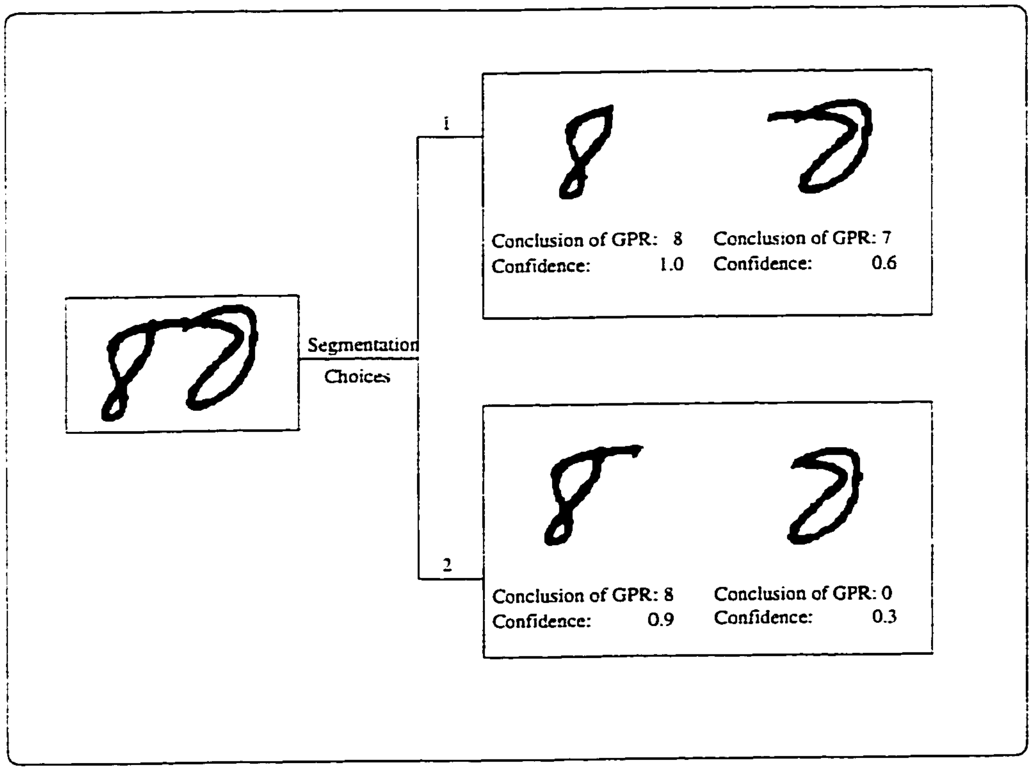


Figure 37: Example of Structural Distortion. A '0' with a ligature on top becomes a '7', which will be negated by the structural verifier.

The problem is not as fatal as errors of segmentation and it is likely to be negated by the verifier for UIHN as described in Chapter 4. However, we can see that despite distortion, the basic structural feature of numeral "0" — the big loop — persists in the image, which can be used for verification purpose.

Before we discuss the detailed features we use in the structural verifier for UTHN, we first emphasize the reason of using them as part of our verification scheme.

In domain specific verification, structural feature plays an important role. It serves as a complement for other verifiers including QNN verifier for UIHN. We prefer to use the structural features in the verification part instead of our GPR, due to several considerations:

1. Structural features are intuitive and explicit observations which match human expect verification angle. As mentioned in Chapter 4, our mission of developing the verifier is to incorporate human reasoning into the *quantitative* and/or *qualitative* aspects of the machine intelligence. We may try all kinds of feature extraction or classification methods in GPR, nevertheless, there is still a possibility of making errors that are not understandable to humans. In a practical system, if the user can not understand the cause of the remaining errors from her/his point of view, the system tends to be rejected. Thus a highlight of the verifier is concern about the needs of the real users. The fact that the verifier is in great demand for practical system justifies our decision of putting structural features in the domain specific verification schemes.

2. Structural verifiers are class-specific. They have a pre-determined target, which makes the scheme computationally effective. If a structural feature is put into GPR, we have to filter every possible identity by hierarchically applying all features before we know which features are essential for the input image to finally determine its class identity.

3. Structural feature is very effective in indicating some dominant features for confirmation or negation as verification purpose. However, it would be a painstaking process to develop complete sets of rules for a ten-class GPR. The manual way of generating good classification rules are very labour-intensive. On the

other hand, automatic rule generation can hardly guarantee the reliability, especially for unconstrained touching or cursive cases, since a very broad range of information will be needed for complete classification rules.

In our structural verifier, holes and concavities are dominant features used for verification purpose. Next section will explain the process of detecting these features.

6.3.1 Detecting Structural Features

Our goal is to detect significant concavities, holes or other dominant features in the numeral image.

Concavity Detection

Concavity detection algorithm is based on contours. The algorithm consists of three steps:

1. Compute the "close-rate" of contour pixels. "close-rate" is the measurement of contour pixel that contains the information of gradient magnitude and gradient direction. So we can hypothesize whether strokes are concave or convex, as well as estimate their dimensions and centroids.

   Definition 6.1 Close-Rate of Contour Pixel (i,j): *Reset counter to 0. Scan from the center pixel (i,j). The eight directions of the scanning are $45°$, ... , $360°$. At each direction, if at least one non-contour black pixel is reached, the counter adds 1 and the scanning of this direction is finished. The final number of the counter is the close-rate of the pixel (i,j).*

   From the definition, we can see that the close-rate has the range of $[0, 8]$. The bigger the rate, the bigger the possibility that the pixel stays at the deep bottom of a concave stroke. Figure 38 is an example of close-rate computation. It provides the close rates of a sample '2'.

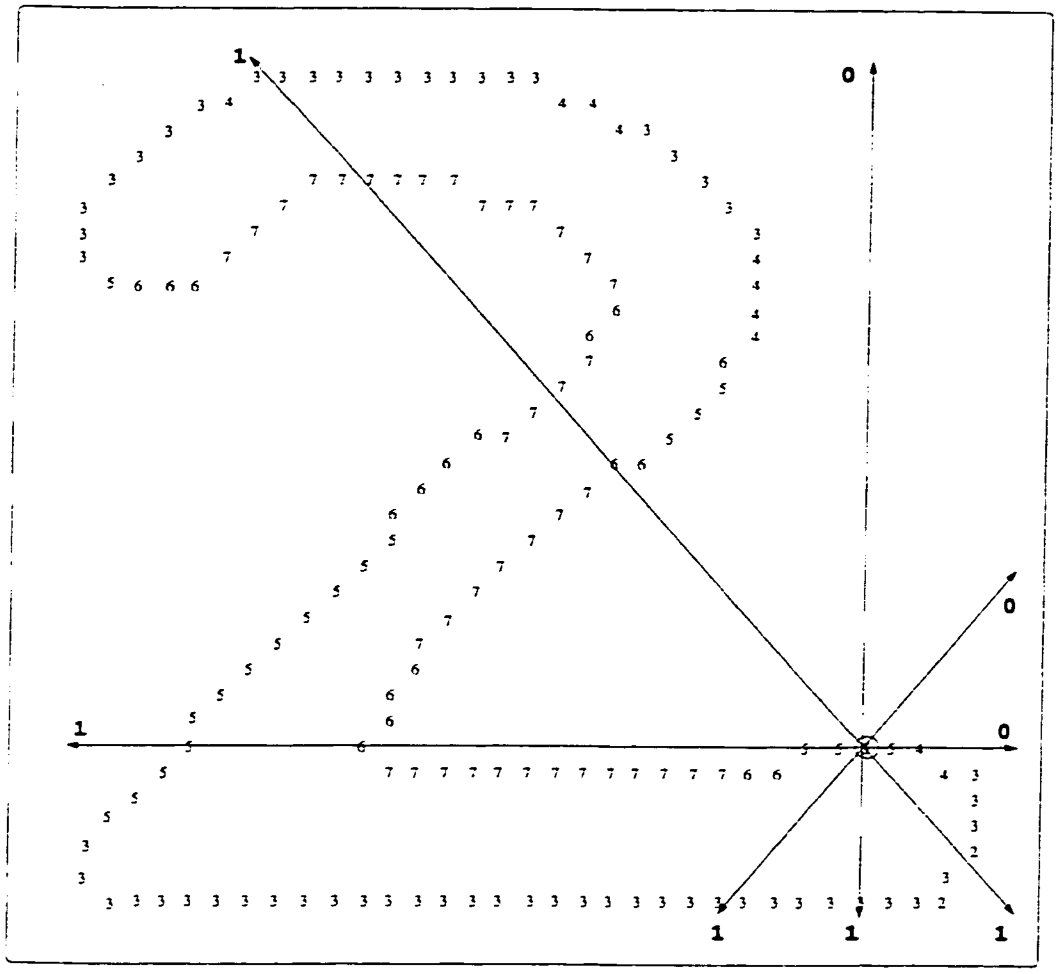


Figure 38: Computation of Contour Pixel Close Rate. The numerals shown along the contour are the close rates of the corresponding pixel. Each of them is obtained by adding the numbers on eight directions according to Definition 6.1.

2. Determine two thresholds LIM1 and LIM2 according to the image dimension. LIM1 is the minimum number of the continuous pixels in a concave stroke with close-rate ≥ 6. LIM2 is the upper limit of the number of broken pixels in a stroke.

3. We search the entire contour to find out all the concave strokes around the contour. We record the detailed structural information of every concave stroke including centroids, dimension, orientation range and other special features. By combining the information, every numeral is related to a set of concavities with the information data.

For every numeral, the concavities found are stored in the data structure "Concave" with all the related information. If the concavity is longer than a threshold, a dominant feature of numeral '3' is examined within the concavity, which is the "chin" in the middle directing towards the left.

Hole Detection

Holes are intrinsic structural features of numerals 0, 6, 8, 9. We scan the original image from vertical and horizontal directions respectively. If in both directions we find the candidate of hole feature, we combine the two dimensional information to confirm the existence of a hole. The algorithm can tolerate slightly broken strokes. It can also prevent insignificant noise in the image.

The algorithm has three steps with Figure 39 as an illustrative example:

1. Take $C(i)$ as the number of strokes which are crossed by the horizontal scanning line $Y = i$. Scan from top to bottom, until $i1$ such as $C(i) == 1$ and $C(i+1) >= 1$;

2. Continue scanning, until $i2$ such that $C(i_2) >= 2$ and $C(i_2 + 1) == 1$. To prevent broken strokes, we continue to scan to see if $C(i_2 + 2) == 1, .., C(i_2 + k) == 1$, where k is a small integer.

3. Given i_1 and i_2, begin to confirm the hole. Assume the internal white region at $Y = i_1 + 1$ is $[l_1, l_2]$. Let $B(i_1 + 1) = |l_2 - l_1|$. Let $D(i_1)$ be the length of the internal black region at $Y = i_1$. Similarly, we can get $B(i_2), D(i_2 + 1), D(j_1), B(j_1 + 1), D(j_2 + 1), B(j_2)$. Compute:

   (a) $r_1 = D(i_1)/B(i_1 + 1)$

   (b) $r_2 = D(i_2 + 1)/B(i_2)$

   (c) $r_3 = D(j_1)/B(j_1 + 1)$

   (d) $r_4 = D(j_2 + 1)/B(j_2)$

   If $(r1 > 0.6\ and r2 > 0.6)$ or $(r3 > 0.6 \&\& r4 > 0.6)$, return TRUE (the hole is confirmed). Otherwise, return FALSE.

6.3.2 Structural Verification

Structural verification for UTHN is a negation scheme achieved by detecting the impossible structural features of a class candidate. Class-specific verification rules are defined for this purpose. For example, numeral "1" will get a negation if the following conditions are satisfied:

1. There exist one or more loops.

2. The height of at least one loop is greater than a threshold θ, which is determined according to the height of the numeral image.

This rule is created to prevent long ligatures from confusing the image identity.

Figure 40 presents another example of structural verification. It is a typical case that a neural-based classifier makes a mistake unlikely understandable to human – an '8' with a long tail becomes a '3' – which can be easily solved by a structural verifier.

The structural verification rules are handcrafted in our system with the efforts to ensure the reliability. Except the time constraint, we have designed the rules carefully in order to improve the reliability while not increasing the rate of false negative. About 3% of the cases in training set of IRIS-Bell'98 are positively affected by this verifier.

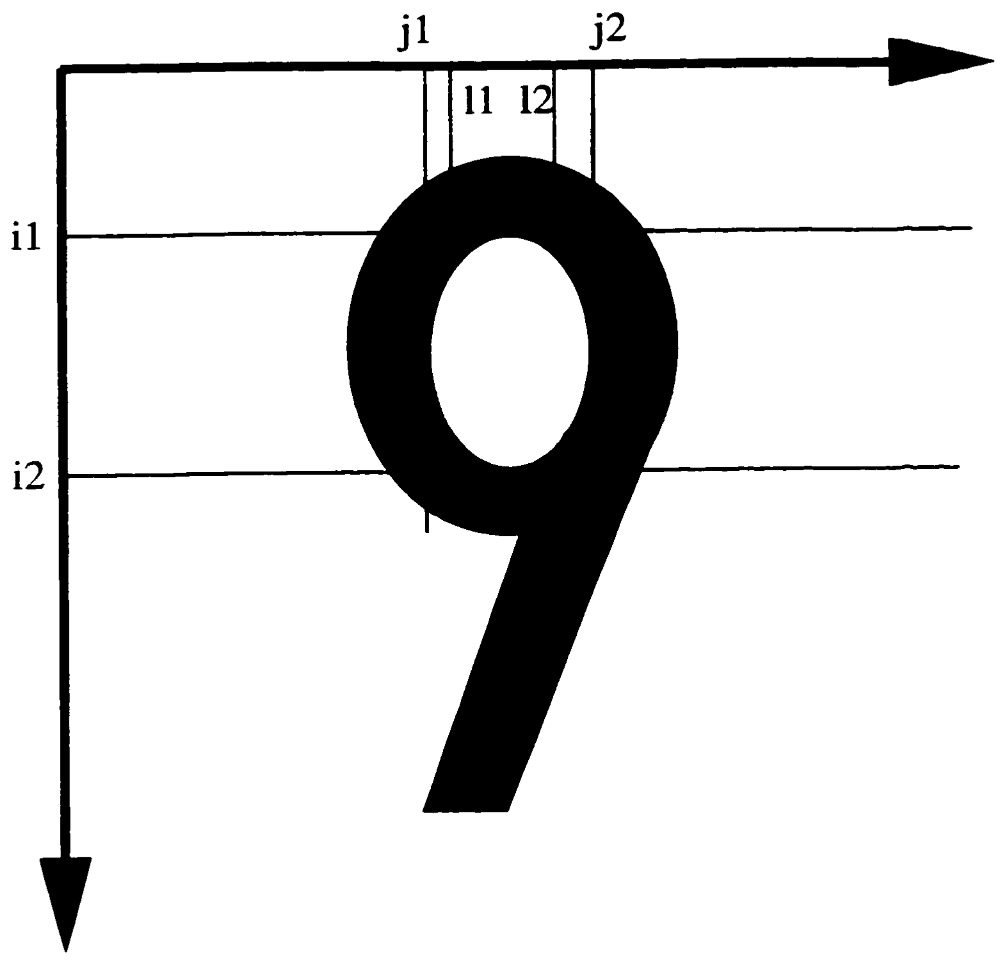


Figure 39: Algorithm of Hole Detection

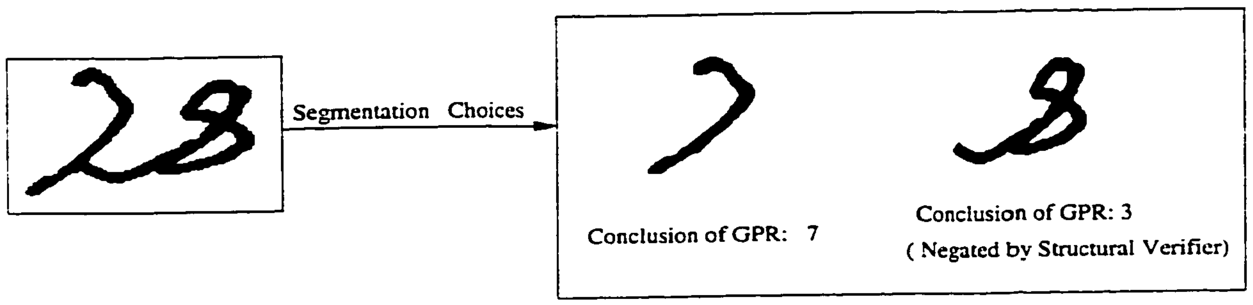


Figure 40: Example of Negation by Structural Verification

6.4 Dummy Symbol Detector

The goal of detecting dummy symbol is to introduce a systematic way to handle the "noisy" parts of the image, which may not be handled easily by normal methods. By normal methods we mean the GPRs and Verifiers that handle the standard 10 numeral classes or their combinations.

Dummy symbol detector is a typical domain-specific problem. In the recognition of UTHN with financial document background, common dummy symbols can be underscores, "xx" in the cents part of the courtesy amount or other unexpected noises. While in other application domains such as postal code processing, hyphen sign is a typical dummy symbol. The dummy symbols are also affected by algorithms. In some stroke based segmentation approach, ligature may be separated as a single component and becomes a dummy symbol. In this thesis, we limit our discussion to underscores. The detector works under the condition that the segmentor can successfully separate the dummy symbol from the numeral.

Figure 41 provides a sample of dummy symbol detection. When the identity conclusion comes from GPR, it should first pass the dummy symbol detector, if the detector considered it as a potential dummy, it will not go to further verification stages for numeral classes.

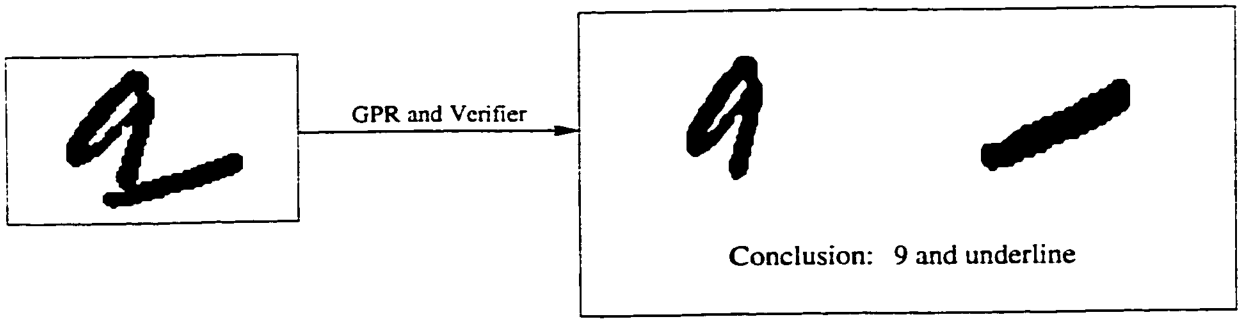


Figure 41: Example of Dummy Symbol Detection: '9' touches a dummy symbol. It is successfully segmented and detected as '9' and an underline.

An image is considered dummy if the following 3 conditions are satisfied:

1. The segmented image locates at the bottom part of the original image of touching numerals.

2. $\frac{Width}{Height} > \theta_1$

3. $Height < \theta_2$

Condition 3 is proposed to prevent cursive numerals that have a high Width/Height ratio. θ_1 is a threshold in the range of $[2, 4]$. θ_2 is a value in the range of $[30, 45]$.

The above 3 conditions are sufficiently strict that no valid numeral can be detected as dummy. However, there is still a possibility of missing some patterns which are actually dummies.

In IRIS-Bell'98, above 60% dummy occurrences in training set are solved by this detector. Among the misses, most are due to the segmentor which cannot separate the touching numeral from the dummy symbol. There is no such dummy symbol in the NIST for UTHN database.

6.5 Architecture and Experimental Results

In the previous several sections, we discussed domain specific verification schemes of Touching Type & Location Verifier, Structural Verifier and Dummy Detector. By incorporating these schemes into the QNN verifier we discussed in Chapter 4, we propose the complete architecture of verifier for UTHN depicted in Figure 42.

The verifier has a hierarchical architecture. When an identity conclusion of a segmented image comes from GPR for UTHN, it first goes through dummy symbol checking. If it is considered as a dummy, further verification dedicated for numeral classes becomes unnecessary. Otherwise, class specific Touching Type & Location Verification and Structural Verification are offered. If the conclusion is negated by either scheme, the system will go back to the next segmentation candidate of different conclusion for a second chance. If both schemes keep silent, the candidate goes

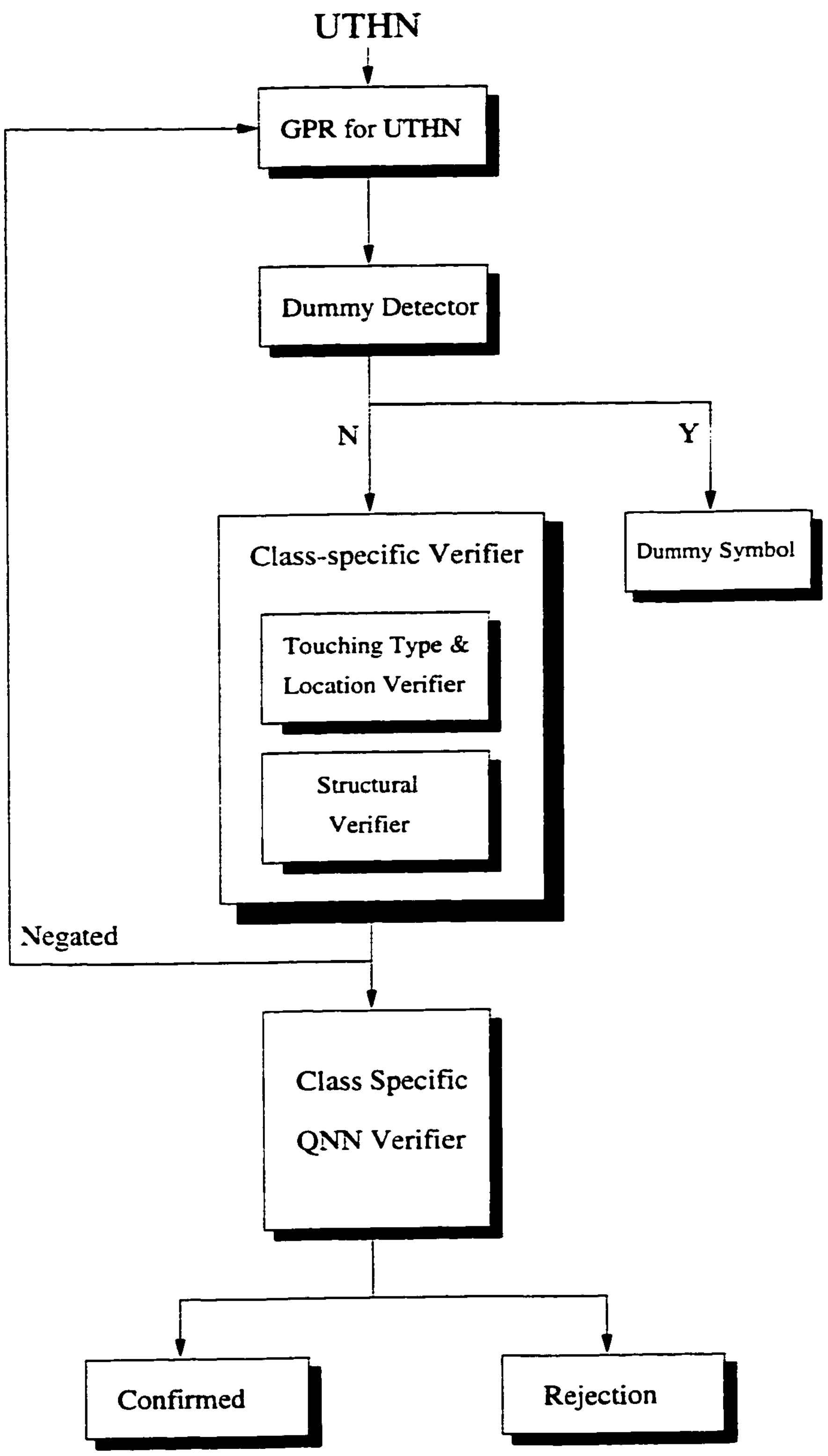


Figure 42: Diagram of Verifier in R&V system for UTHN

straight to the QNN Verifier for a final verification. The result will be either a confirmation or a rejection.

The experiments are conducted on IRIS-Bell'98 database and NIST for UTHN database. The complete R&V system produces the performance shown in Table 30. It is one of the common strategies when dealing with two measurements of one system which may not go toward the same direction: Try to restrict one factor at a specified level then control the other (which is usually of more interest to us) to the goal as desirable as possible. Compared with the results from GPR for UTHN, the precision improvement is obvious, while the recognition rate does not drop much. It can be explained by the reliable negation scheme with a low false negative rate, and the helpful second chance loop.

For NIST, the drop of recognition rate is less than 3% while the improvement of precision rate is much bigger. For IRIS-Bell'98, the drop of recognition rate is smaller since the dummy detector also corrected some errors. However, as we emphasized in Section 4.5, instead of simply comparing two figures, an evaluation based on hypothesis testing is more accurate and convincing, as we will do in the next section.

Compare Table 30 with Table 3 in Chapter 1, we can see that the relibilities of our system, reflected by precision rates, is much better than those reported on data extracted from CEDAR CDROM [89, 102, 32]. The published results were not based on a well known standard set, which limits the possibility of further comparison. We expect that the building of IRIS-Bell'98 and NIST for UTHN databases will provide an open and standard base for researchers to exploit and compare techniques of UTHN problem.

Database	Correct(%)	Substitution(%)	Precision for Strings(%)
IRIS-Bell'98	65.2	9.2	89.2
NIST	85.7	3.5	96.1

Table 30: Test Results of R&V System for UTHN

6.6 Evaluation of Recognition and Verification System for UTHN

As in Section 4.5, the effectiveness of our verifier and R&V system for UTHN is subject to the evaluation of statistical hypothesis testing. The test is based on NIST for UTHN which is a bigger database than IRIS-Bell'98.

Let A be the random variable that represents the precision rates of a system without the verification module, let B represent the precision rates of a system with verification module. We randomly selected 20 sets of samples from the test database of NIST for UTHN. So the number of the two sets for hypothesis testing are $n_1 = n_2 = 20$. For each set, we get a precision rate as shown in Table 31.

From Table 31, we can get the sample averages of A & B as $\overline{X_1} = 90.21$, $\overline{X_2} = 96.28$ respectively and compute the sample standard deviations of A & B S_1 and S_2 as square root of $E(X - E(X))^2$, where X is A or B. We get $S_1 = 8.19$ and $S_2 = 3.74$ respectively. With the definition of

$$S^2 = \frac{n_1 S_1^2 + n_2 S_2^2}{n_1 + n_2 - 2}$$

we get $S = 2.5$.

The test starts by setting the hypotheses:

Step 1: Set H_0: Systems with or without verifier have no significant difference on precision rates. H_1: Systems with verifier have significant improvement on precision rates.

Step 2: Let $N(\mu_1, \sigma_1^2)$ and $N(\mu_2, \sigma_2^2)$ be the distributions of A and B respectively. Assume $\sigma_1^2 = \sigma_2^2$ (value unknown). The populations are assumed to be independent. Thus $H_0 : \mu_1 = \mu_2$. $H_1 : \mu_1 > \mu_2$.

Step 3: Use $t - test$, the appropriate statistic under H_0 is,

$$\frac{(\overline{X_1} - \overline{X_2}) - (\mu_1 - \mu_2)}{S(\frac{1}{n_1} + \frac{1}{n_2})^{1/2}} = 7.58$$

Step 4: Test the hypothesis at 0.01 significance level. $\alpha = 0.01$, $t_{n_1+n_2-2,\alpha} = t_{38,0.01}$.

Dataset	Precision Rate for A (%)	Precision Rate for B(%)
1	85.82	91.23
2	92.69	95.21
3	89.27	97.03
4	92.21	94.25
5	91.15	97.64
6	90.07	96.25
7	86.69	93.29
8	87.09	95.24
9	86.48	97.56
10	90.65	97.34
11	86.51	95.51
12	93.46	98.71
13	93.77	98.29
14	92.32	97.63
15	93.15	97.47
16	96.03	99.06
17	91.18	96.32
18	88.40	94.14
19	90.03	97.82
20	87.31	95.66

Table 31: Data Sets for Hypothesis Testing to Evaluate Verifier for UTHN

Step 5: Compute $t_{38,0.01}$. Since 38 > 30, we get a good normal approximation. By looking into a standard normal table, corresponding to $\alpha = 0.01$, we have

$$P\{Z \geq 2.33\} = 0.01$$

.

Step 6: Since 7.58 > 2.33, hence the observed value of Z lies in the critical region. Hence we reject $H_0 : \mu_1 = \mu_2$ in favour of $H_1 : \mu_1 > \mu_2$.

By hypothesis testing, we get the conclusion that R&V system for UTHN has significantly improved the precision rates, thus our proposed verification scheme properly suits our definition of a good verifier.

Chapter 7

Conclusion

"The method employed I would gladly explain, ...,
But much yet remains to be said."

— Lewis Carroll, Alice in Wonderland

For decades, researchers worked on the topic of handwriting recognition, with two ultimate goals in mind:

- to provide deeper understanding of the essence of pattern recognition;
- to exploit techniques for OCR and document analysis systems for market needs.

In this thesis, tremendous efforts have been made with these two inherent underlying directions, reflected by the theory of pattern verifier, and the two proposed R&V systems for handwritten numerals.

The concepts of verifier are highlighted in this chapter. A summary of contributions and discussions about future directions is also provided.

7.1 Summary of Verifier

A R&V system contains two modules: General Purpose Recognizer and Verifier. Essentially, the role of a general purpose recognizer is indexing, by which we mean labelling a class identity on an input sample. The task of classification thus can be viewed as two-stage process: indexing and verification. Indexing has received much attention in the literature with various methods developed. The scheme of verification, although essential for systems that demand high reliability, has attained little attention and is generally used in its simplest form.

In this thesis, the concept of a verifier has been proposed in the domain of handwritten numeral recognition system. Despite that all the related topics have been addressed in the previous chapters, we think it is worthwhile to highlight its difference from a general purpose recognizer in the concluding remarks, so that some ambiguities can be resolved and deep insights can be brought out.

The comparison is based on aspects of basic task goal, measurement, target range and methodology. The highlights are listed in Table 32.

	Verifier	Recognizer
Task Goal	Reliability	Indexing Ability
Measurement	Precision Rate	Recognition Rate
Target Range	Focus on a pre-defined identity (class-specific). Give a confirmation or negation.	Choose from a set of identities.
Methodology	Advanced requirements such as better discriminant power along decision boundary or other specially tuned ad hoc abilities.	No such requirements.

Table 32: Summary of Differences between Recognizer and Verifier

7.2 Contributions

We view the introduction of verifier concept and its role in handwritten numeral recognition systems as original contribution of this thesis.

As a pioneering work, the definitions of the role and measurement for a verifier have been carefully examined to match the goal and functionality of the verification scheme. Precision rate has been finally chosen due to its precise performance representation from the verifier's point of view. It is a novel angle to look at a pattern recognition system. The usage and types of verifiers are also discussed for the purpose of incorporating it into a practical R&V system in the most effective way.

Analysis of theoretical requirement for a good verifier is conducted. A better discriminant power along a decision boundary to separate confusing patterns is desired in our verifier for handwritten numerals. A class-specific design strategy is suggested for UIHN and UTHN problems. The power of verification is pointed out for the control policy of verifier in practical systems.

Using hypothesis testing to justify the effectiveness of an algorithm is a new alternative testing methodology for OCR problems, which considers sample size and variance of data properties. It raises the logical question whether the comparision between two percentage figures is enough to prove the superiority of an algorithm and provides a possible answer.

From a practical perspective, the contributions can be summarized as follows:

- A complete neural-based general purpose recognizer for UIHN is provided with in-depth discussions of three neural approaches and performance optimization. For the first time, multilayer perceptron, convolutional network and radial basis function network are combined together to produce a well performing recognizer for UIHN.

- Tree classifier, a method popular for medical diagnosis and signal classification but seldom seen in the field of handwriting recognition, is applied to UIHN problem and proved feasible. A comparison between neural approach and tree classifier is also conducted for a deeper understanding of both schemes.

- In designing the verifier for UIHN, a novel approach of Quantum Neural Network is provided. The application of QNN to handwritten numeral recognition is done the first time in OCR literature. The advantageous characteristics of QNN is applied in an effective and computationally efficient way to fulfill its role as a verifier.

- A graph-based combination scheme of segmentation-based and segmentation-free approaches is proposed in general purpose recognizer for UTHN. It takes the segmentor and dual-recognizers into an overall consideration and converts many *hard constraints* into *soft decisions* expressed in terms of costs.

- Touching Type & Location analysis Structural Verification are embedded in Verifier for UTHN. Effective methods are proposed in end-region detection and structural feature extraction. A verifier for UTHN is designed by combining domain-specific verification scheme and QNN verifier effectively.

- With the intention to benchmark the research work of UTHN problem, two databases are built with the efforts of researchers in CENPARMI and the author. They are IRIS-Bell'98 and NIST for UTHN. The availability of standard databases is an important element and will benefit newcomers in this field. An analysis of the properties of these two databases in terms of touching types has also been conducted to provide deep understandings of the UTHN problem.

7.3 Future Directions

This thesis provides pioneering thoughts and basic strategies for verifiers. As a promising and important scheme, it must be of great interest to extend the work of verifier into more concrete theories and practical engines.

The theory can advance toward the general design strategy by examining pattern recognition problems other than recognizing the UIHN and UTH. In our current systems, the architectures, although effective, are somehow domain-specific. A systematic study of design strategies will be desirable for further theoretical work.

Another open problem is the control strategy. Preliminary work has been included in the thesis such as the definition of power of verification and ad hoc policies. However, we can expect that the theoretical study of this topic has deep meaning in understanding the relationship between the measurements of a practical system and providing the bounds of performance measurements for a specific application. Hints may be obtained from other fields of information technology or some statistical theories.

For the existing practical R&V systems for UIHN and UTHN, there are also avenues to be followed.

The performances of the systems have not achieved their upper limit. Due to the large number of engines and the time-consuming neural training, some training phases had to be stopped before they reached the best capability. Parameters within verifier architecture and thresholds can be further tuned.

For UTHN problem, the segmentor works fairly well for simple touching situations, but many errors occur when complicated writing styles are encountered. Although segmentation is not the focus of this thesis, it is essential for the performance of practical UTHN systems and more effort should be expended in this area if a commercial system is envisaged. Garbage detector is another component needed to be strengthened.

Finally, it might be interesting to automate the rules of Touching Type & Location Verification and Structural Verification for UTHN. Nevertheless, we shall be careful in exploring this direction. The reason has been indicated in [63]: although some global attempts could be made on automation of part (or all) of the process, the high reliability of the rules would most likely disappear and that will defect the whole purpose of our work.

Bibliography

[1] Y. Amit, D. Geman, and K. Wilder. Joint introduction of shape features and tree classifiers. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 19(11):1300–1305, November 1997.

[2] H. S. Baird, S. Kahan, and T. Pavlidis. Components of an omni-font page reader. In *Proceedings of Eighth International Conference on Pattern Recognition*, pages 344–348, Paris, 1986.

[3] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International, 1984.

[4] D. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex System*, 2:321–323, 1988.

[5] J. Cai and Z.-Q. Liu. Integration of structural and statistical information for unconstrained handwritten numeral recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(3), March 1999.

[6] G. Casella and R. L. Berger. *Statistical Inference*. Wadsworth & Books, 1990.

[7] R. G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 18(7):690–706, July 1996.

[8] R. G. Casey and G. Nagy. Decision tree design using a probabilistic model. *IEEE Transactions on Information Theory*, pages 93–99, January 1984.

[9] G. L. Cash and M. Hatamain. Optical character recognition by the method of moments. *Computer Vision, Graphics, and Image Processing*, 39:93–99, 1987.

[10] C. Che, Q. Lin, and D. suk Yuk. An HMM aproach to text-prompted speaker verification. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing*, pages 673–676, Atlanta, USA, May 1996.

[11] M. Cheriet, Y. S. Huang, and C. Y. Suen. Background region-based algorithm for the segmentation of connected digits. In *Proceedings of 11th International Conference for Pattern Recognition*, volume 2, page 619, Sept. 1992.

[12] C. Chiang and S. Yu. A method for improving the machine recognition of confusing Chinese characters. In *Proceeding of ICPR96*, pages 79–83, 1996.

[13] S. Cho and J. H. Kim. Multiple network fusion using fuzzy logic. *IEEE Transactions on Neural NEtworks*, 6(2), 1995.

[14] S.-B. Cho. Neural-network classifiers for recognizing totally unconstrained handwritten numerals. *IEEE Transactions on Neural Networks*, 8(1):43–53, January 1997.

[15] A. de Souza Britto Jr. Proposal of a new method for recognition of handwritten numeral strings. Technical report, ETS- Ecole Technologie Superieure, March 1999.

[16] P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prentice-Hall, London, 1982.

[17] L. Devroye, L. Gyorfi, and G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, 1996.

[18] G. Dimauro, S. Impedovo, G. Pirlo, and A. Salzo. Handwriting recognition: State of the art and future trends. In N. A. Murshed and F. Bortolozzi, editors, *Advances in Document Image Analysis*, pages 1–18. Springer, Heidelberg, Germany, November 1997.

[19] R. O. Duda and P. E. Hart. *Pattern classification and scene analysis.* John Wiley & Sons, New York, 1973.

[20] G. Dzuba, A. Filatov, D. Gershuny, I. Kil, and V. Nikitin. Check amount recognition based on the cross validation of courtesy and legal amount fields. *International Journal of Pattern Recognition and Artificial Intelligence*, 11(4):639–655, 1997.

[21] F. Espositio, D. Malerba, and G. Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and machine intelligence*, 19(5):476–491, May 1997.

[22] J. A. Feldman and D. H. Ballard. Connectionist models and their properties. *Cognitive Science*, 6:205–254, 1982.

[23] A. Filatov, A. Gitis, and I. Kil. Graph-based handwritten digit string recognition. In *Proceedings of ICDAR'95*, pages 845–848, Montreal, 1995.

[24] A. Filatov, V. Nikitin, A. Volgunin, and P. Zelinsky. The *AddressScript*^{TM} recognition system for handwritten envelopes. In *Proc. of the Third International Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 222–236, Nagano, Japan, November 1998.

[25] J. Franke, L. Lam, R. Legault, C. Nadal, and C. Y. Suen. Experiments with the cenparmi database combining different classification approaches. In *Proceedings of IWFHR'93*, pages 305–311, Buffalo, New York, May 1993.

[26] K. S. Fu. *Syntactic Pattern Recognition and Applications.* Prentice-Hall, 1982.

[27] K. Fukushima. Neocognitron: A self-organizing neural-network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 1980.

[28] J. Gomes and L. Velho. *Image Processing For Computer Graphics.* Springer-Verlag, New York, 1997.

[29] M. Gori and F. Scarselli. Are multilayer perceptrons adequate for pattern recognition and verification? *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 20(11):1121–1132, 1998.

[30] P. J. Grother. NIST special database 19 handprinted forms and characters database. Technical report, National Institute of Standards and Technology, March 16 1995. Postscript file sd19.cd/doc/doc.ps on Special Database 19 CD ROM.

[31] T. M. Ha and H. Bunke. Off-line handwritten numeral recognition by perturbation method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):535–539, May 1997.

[32] T. M. Ha, M. Zimmermann, and H. Bunke. Off-line handwritten numeral string recognition by combining segmentation-based and segmentation-free methods. *Pattern Recognition*, 31(3):257–272, 1998.

[33] T. Hastie et al. Learning prototype models for tangent distance. In *Proceedings of Neural Information Processing Systems Conference*, 1994.

[34] S. Haykin. *Neural Networks: A Comprehensive Foundation.* Prentice Hall, 1998.

[35] D. O. Hebb. *The Organization of Behavior.* John Wiley & Sons, New York, 1949.

[36] J. Hertz, A. Krogh, and R. Palmer. *Introduction to the theory of neural computation.* Addison-Wesley, 1991.

[37] T. K. Ho. *A Theory of Multiple Classifier Systems And Its Application to Visual Word Recognition.* Ph.D. thesis, State University of New York at Buffalo, 1992.

[38] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:832–844, August 1998.

[39] J. J. Hopfield and D. W. Tank. Computing with neural circuits: A model. *Science*, 233:625–633, 1986.

[40] G. F. Houle, D. B. Aragon, and R. W. Smith. A multi-layered corroboration-based check reader. In *Proc. of IAPR Workshop on Document Analysis Systems*, pages 495–543, 1996.

[41] J. Hu and H. Yan. Structural primitive extraction and coding for handwritten numeral recognition. *Pattern Recognition*, 31(5):493–509, 1998.

[42] J. S. Huang and M. L. Chung. Separating similar complex Chinese characters by walsh transform. *Pattern Recognition*, 20(4):425–428, 1987.

[43] Y. S. Huang and C. Y. Suen. An optimal method of combining multiple classifiers for unconstrained handwritten numeral recognitio. In *Proceedings of 3rd International Workshop on Frontiers in Handwriting Recognition*, May 1993.

[44] Y. S. Huang and C. Y. Suen. A method of combining multiple experts for the recognition of unconstrained handwritten numerals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(1):90–94, 1995.

[45] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology (London)*, 160:106–154, 1962.

[46] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Recognition and machine intelligence*, 16(5):550–554, May 1994.

[47] D. R. Hush and B. G. Horne. Progress in supervised neural networks. *IEEE Signal Processing Magazine*, pages 8–39, 1993.

[48] A. K. Jain and D. Zongker. Representation and recognition of handwritten digits using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1386–1391, December 1997.

[49] S. Kahan, T. Pavlidis, and H. S. Baird. On the recognition of printed characters of any font and size. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(2):274–288, March 1987.

[50] T. Kawatani, H. Shimizu, and M. McEachern. Handwritten numeral recognition with the improved LDA method. In *Proceedings of International Conference on Pattern Recognition*, pages 441–446, 1996.

[51] W. Kim, J. Paik, K. Lee, H. Byun, and Y. Lee. handwritten digit verifier for improving recognition error. In *Proceedings of ICDAR97*, 1997.

[52] J. Kittler, M. Hatef, R. P. Duin, and J. Matas. On combining classifiers. *IEEE transactions on pattern recognition and machine intelligence*, 20(3):226–239, march 1998.

[53] M. A. Kraaijveld. An experimental comparison of non-parametric classifiers for time-constrained classification tasks. In *Proceedings of International Conference on Pattern Recognition*, pages 428–435, 1998.

[54] L. Lam and C. Y. Suen. Structural classification and relaxation matching of totally unconstrained handwritten zip-code numbers. *Pattern Recognition*, 21(1):19–31, 1988.

[55] L. Lam and C. Y. Suen. Optimal combinations of pattern recognition. *Pattern Recognition Letters*, 16:945–954, 1995.

[56] L. Lam and C. Y. Suen. Application of majority voting to pattern recognition: An analysis of its behavior and performance. *IEEE Transactions on Systems, man and Cybernetics*, 27(5):553–568, September 1997.

[57] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[58] Y. LeCun et al. Constrained neural network for unconstrained handwritten digit recognition. In *Proceedings of the International Workshop on Frontiers in Handwriting Recognition, Montreal*, pages 145–152, 1990.

[59] D. Lee. *A Theory of Classifier Combination: The neural network approach.* PhD thesis, State University of New York at Buffalo, 1995.

[60] D. S. Lee and S. N. Srihari. Handprinted digit recognition: A comparision of algorithms. In *Proc. of the third International Workshop on Frontiers in Handwritting Recognition*, pages 153–162, Buffalo, New York, USA, May 1993.

[61] D. S. Lee and S. N. Srihari. A theory of classifier combination: the neural network approach. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, pages 42–45, Montreal, August 1995.

[62] S.-W. Lee. Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural network. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, 18(6):648–650, 1996.

[63] R. Legault. *Unconstrained Handwritten Numeral Recognition: A Contribution towards matching human performance.* Ph.D. thesis, Concordia University, 1997.

[64] R. Legault and C. Y. Suen. A comparison of methods of extracting curvature features. In *Proceedings of 11th International Conference on Pattern Recognition*, pages 134–138, The Hague, The Netherlands, Aug.-Sept. 1992.

[65] R. P. Lippman. In introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–14, April 1987.

[66] D. R. Lovell, T. Downs, and A. C. Tsio. An evaluation of the neocognitron. *IEEE Transactions on Neural Networks*, 8(5):1090–1105, 1997.

[67] J. Mao, K. Mohiuddin, and T. Fujisaki. A two-stage multi-network ocr system with a soft pre-classifier and a network selector. In *Proceedings of ICDAR*, volume 1, pages 78–80, 1995.

[68] A. M. Mathai and P. N. Rathie. *Probability and Statistics*. Macmillan India Press, 1977.

[69] W. S. McCulloch and W. Pittts. A logical calculus of the ideas imminent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.

[70] S. Mori, C. Y. Suen, and K. Yamamoto. Historical review of ocr research and development. *Proceedings of the IEEE*, 80:1029–1058, 1992.

[71] B. Mulgrew. Applying radial basis functions. *IEEE Signal Processing Magazine*, pages 50–65, 1996.

[72] G. Nagy. State of the art in pattern recognition. *Proceedings of the IEEE*, 56(836), 1968.

[73] C. Neubauer. Evaluation of convolutional neural networks for visual recognition. *IEEE Transactions on Neural Networks*, 9(4):685–696, 1998.

[74] I. Oh, J. Lee, K.-C. Hong, and S. Choi. Class-expert approach to unconstrained handwritten numeral recognition. In *Proceedings of IWFHR'96*, pages 35–40, Colchester, UK, September 1996.

[75] J. Paik, S. Cho, K. Lee, and Y. Lee. Multiple recognizers system using two-stage combination. In *Proceedings of International Conference on Pattern Recognition*, volume IV, pages 581–585, August 1996.

[76] H.-S. Park and S.-W. Lee. A truly 2-d hidden markov model for off-line handwritten character recognition. *Pattern Recognition*, 31(12):1849–1864, 1998.

[77] S. Procter, J. Illingworth, and A. J. Elms. The recognition of handwritten digit strings of unknown length using hidden markov models. In *Proceedings of*

Fourteenth International Conference on Pattern Recognition, pages 1515–1517, August 1998.

[78] G. Purushothaman and N. B. Karayiannis. Quantum neural networks (QNN's): Inherently fuzzy feedforward neural networks. *IEEE Transactions on Neural Networks*, 8(3):679–693, May 1997.

[79] J. R. Quinlan. *C4.5: Programming For Machine Learning.* Morgan Kaufmann, San Mateo, California, 1993.

[80] W. L. Quirin. *Probability and Statistics.* Harper & Row, 1978.

[81] L. R. Rabiner. A tutorial on hidden markov models and selected application in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[82] M. G. Rahim, C. Lee, and B. Juang. Discriminative utterance verification for connected digits recognition. *IEEE transactions on Speech and Audio Processing*, 5(3):266–277, 1997.

[83] A. F. R. Rahman and M. C. Fairhurst. An evaluation of multi-expert configurations for the recognition of handwritten numerals. *Pattern Recognition*, 31(9):1255–1273, 1998.

[84] M. Revow, C. K. Williams, and G. E. Hinton. Using generative models for handwritten digit recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligene*, 18(6):592–606, June 1996.

[85] R. Rosenblatt. *Principles of Neurodynamics.* Spartan Books, New York, 1959.

[86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of CCognition*, volume 1. MIT Press,Cambridge, Massachusetts, 1986.

[87] S. R. Safavian and D. Landgrebe. A survey of decision tree classifer methodology. *IEEE Transactions on System, Man, and Cybernetics*, 21(3):660–674, 1991.

[88] J. Schurmann. *Pattern Classification — A unified view of statistical and neural approaches.* Wiley-interscience, 1996.

[89] Z. Shi and V. Govindaraju. Segmentation and recognition of connected handwritten numeral strings. *Pattern Recognition*, 30(9):1501–1504, 1997.

[90] M. Shridhar and A. Badreldin. High accuracy character recognition algorithm using fourier and topological descriptors. *Pattern Recognition*, 17(5):515–524, 1984.

[91] N. W. Strathy. A method for segmentation of touching handwritten numerals. Master's thesis, Dept. of Computer Science, Concordia University, 1993.

[92] N. W. Strathy, C. Y. Suen, and A. Krzyzak. Segmentation of handwritten digits using contour features. In *Proc. of the Second International Conference on Document Analysis and Recognition*, pages 577–580, 1993.

[93] C. Y. Suen, L. Lam, D. Guillevic, N. W. Strathy, M. Cheriet, J. N. Said, and R. Fan. Bank check processing system. *International Journal of Imaging Systems and Technology*, 7:392–403, 1996.

[94] C. Y. Suen, R. Legault, C. Nadal, M. Cheriet, and L. Lam. Buildling a new generation of handwriting recognition systems. *Pattern Recognition Letters*, 14:303–315, 1993.

[95] C. Y. Suen, K. Liu, and N. W. Strathy. Sorting and recognizing cheques and financial documents. In *Proc. of the Third International Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 1–18, 1998.

[96] C. Y. Suen, C. Nadal, R. Legault, T. A. Mai, and L. Lam. Computer recognition of unconstrained handwritten numerals. *Proceedings of the IEEE*, 80:1162–1180, 1992.

[97] C. Y. Suen and R. J. Shillman. Low error rate optical character recognition of unconstrained handprinted letters based on a model of human perception. *IEEE Transactions on Systems, Man and Cybernetics*, pages 491–495, June 1977.

[98] R. A. Sukkar and C. Lee. Vocabulary independent discriminative utterance verification for nonkeyword rejection in subword based speech recognition. *IEEE transactions on Speech and Audio Processing*, 4(6):420–429, 1996.

[99] H. Takahashi and T. D. Griffin. Recognition enhancement by linear tournament verification. In *Proceedings of ICDAR93*, pages 585–588, 1993.

[100] S. N. S. Tin Kam Ho, Jonathan J. Hull. Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):66–75, January 1994.

[101] O. D. Trier, A. k. Jain, and T. Taxt. Feature extraction methods for character recognition — a survey. *Pattern Recognition*, 29(4):641–662, April 1996.

[102] X. Wang, V. Govindaraju, and S. Srihari. Holistic recognition of touching digits. In *Proceedings of Sixth International Workshop on Frontiers in Handwriting Recognition*, pages 295–303, Taejon, Korea, August 1998.

[103] J. M. Westall and M. S. Narasimha. Vertex directed segmentation of handwritten numerals. *Pattern Recognition*, 26(10):1473–1486, 1993.

[104] R. Wilkinson, J. Geist, S. Janet, P. Grother, and C. Burges. The first census optical character recognition systems conference. Technical report, National Inst. of Standards and Technology, Gaithersburg, MD, August 1992.

[105] K. Y. Wong, R. G. Casey, and F. M. Wahl. Document analysis system. *IBM Journal of Research and Development*, 26(6):647–656, 1982.

[106] K. Woods, W. P. K. Jr., and K. Bowyer. Combination of multiple classifier using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410, April 1997.

[107] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):418–435, 1992.

[108] D. Yu and H. Yan. Separation of single-touching handwritten numeral strings based on structural features. *Pattern Recognition*, 31(12):1835–1847, 1998.

[109] J. Zhou. Verifier of unconstrained handwritten numerals. Technical report, Concordia University, April 1999.

[110] J. Zhou, Q. Gan, A. Krzyzak, and C. Y. Suen. Quantum neural network in recognition of handwritten numerals. In *Proceedings of Sixth International Workshop on Frontiers in Handwriting Recognition*, pages 305–314, Taejon, Korea, August 1998.

[111] J. Zhou, Q. Gan, A. Krzyzak, and C. Y. Suen. Recognition of handwritten numerals by quantum neural network with fuzzy features. *International Journal on Document Analysis and Recognition*, 2(1):30–36, 1999.

[112] J. Zhou, Q. Gan, and C. Y. Suen. A high performance hand-printed numeral recognition system with verification module. In *Proceedings of ICDAR'97*, volume 1, pages 293–297. IEEE Computer Society, 1997.

Appendix A

Notations

C_m	pattern class m
S_m	the set of samples belonging to C_m within the input patterns
K_m	number of input patterns classified as class C_m by the system
X_m	number of patterns that *truly* belong to S_m within K_m
H_0	null hypothesis
H_1	alternative hypothesis
$w_{l,i,j}$	connecting weight between ith node in l layer and jth node in l+1 layer in a neural network
$d_{i,k}$	desired output of ith output node for input sample x_k
$u_{i,k}$	real output of ith output node for input sample x_k
$o_{i,k}$	output of ith hidden layer node for input sample x_k
$x_{n,k}$	nth component for input sample x_k
P	number of training patterns
n_h	number of hidden layer nodes
n_i	number of input layer nodes
n_o	number of output layer nodes
η	learning rate of training algorithm
α	momentum rate of the training algorithm
α in hypothesis testing	level of significance

G_i	gradient of the *ith* output of conventional neural networks
C, M	positive constants
γ	real number $\in (0, 1)$
p, q	feature vectors
θ_s	quantum interval of quantum level s in QNN
ns	total number of quantum levels in QNN
β	slope factor of activation function
W	connectivity weight vector
X	input vector of quantum neuron
$\sigma^2_{i,m}$	variance of the output of *ith* hidden unit for class C_m
$\|C_m\|$	cardinality of C_m
$O_{i,k}$	output of the *ith* hidden unit with input vector x_k
$\langle O_{i,m} \rangle$	$\frac{1}{\|C_m\|} \sum_{x_k : x_k \in C_m} O_{i,k}$
$O_{i,k,s}$	output of the *sth* quantum level of the *ith* hidden unit with input vector x_k
$\nu_{i,k,s}$	$O_{i,k,s}(1 - O_{i,k,s})$
$\langle \nu_{i,m,s} \rangle$	$\frac{1}{\|C_m\|} \sum_{x_k : x_k \in C_m} \nu_{i,k,s}$
$\Delta\theta_{i,s}$	adjustment of quantum interval for each hidden unit i and its *sth* quantum level (QNN)

Appendix B

Glossary

BP	Back Propagation
CENPARMI	Center for Pattern Recognition and Machine Intelligence
CEDAR	Center of Excellence for Document Analysis and Recognition
GPR	General Purpose Recognizer
IRIS	Institute for Robotics and Intelligent Systems
MLP	Multilayer Perceptron
MNIST	Modified NIST
NIST	National Institute of Standards and Technology
OCR	Optical Character Recognition
PR	Precision Rate
QNN	Quantum Neural Network
R&V	Recognition & Verification
RBF	Radial Basis Function
RR	Recall Rate
SCP	Significant Contour Point
SRCG	Segmentation Recognition Cost Graph
UIHN	Unconstrained Isolated Handwritten Numerals
UTHN	Unconstrained Touching Handwritten Numerals

Appendix C

Databases

C.1 CENPARMI Database for Isolated Numerals

CENPARMI database for isolated numerals is one of the first databases that has been used as a standard for UIHN problem. The descriptions can be found in [96] and [63]. Here we give a brief introduction of the database.

The database consists of about 17,000 run-length coded binarized numerals. The samples were collected from the zip codes of dead letter envelopes provided by the U.S. Postal Service at different locations in the U.S. Assuming one ZIP code of five digits per writer, there are around 3400 writers.

The data were digitized in 16 grey levels on 64*224 grid of 0.153mm square elements, giving a resolution of 166 dots per inch. Each digitized zipcode was enhanced, binarized and segmented, trying to obtain no more than five body regions per ZIP code. When successful, the resulting binary images of individual digits were run-length encoded.

The samples in the whole data base are unevenly distributed across the ten numeral classes. So the finally constituted database consists totally 6000 numerals — Two training sets A and B and a testing set T, each consisting of 2,000 samples, with

200 of each class.

Figure 43 gives some representative samples of the database.

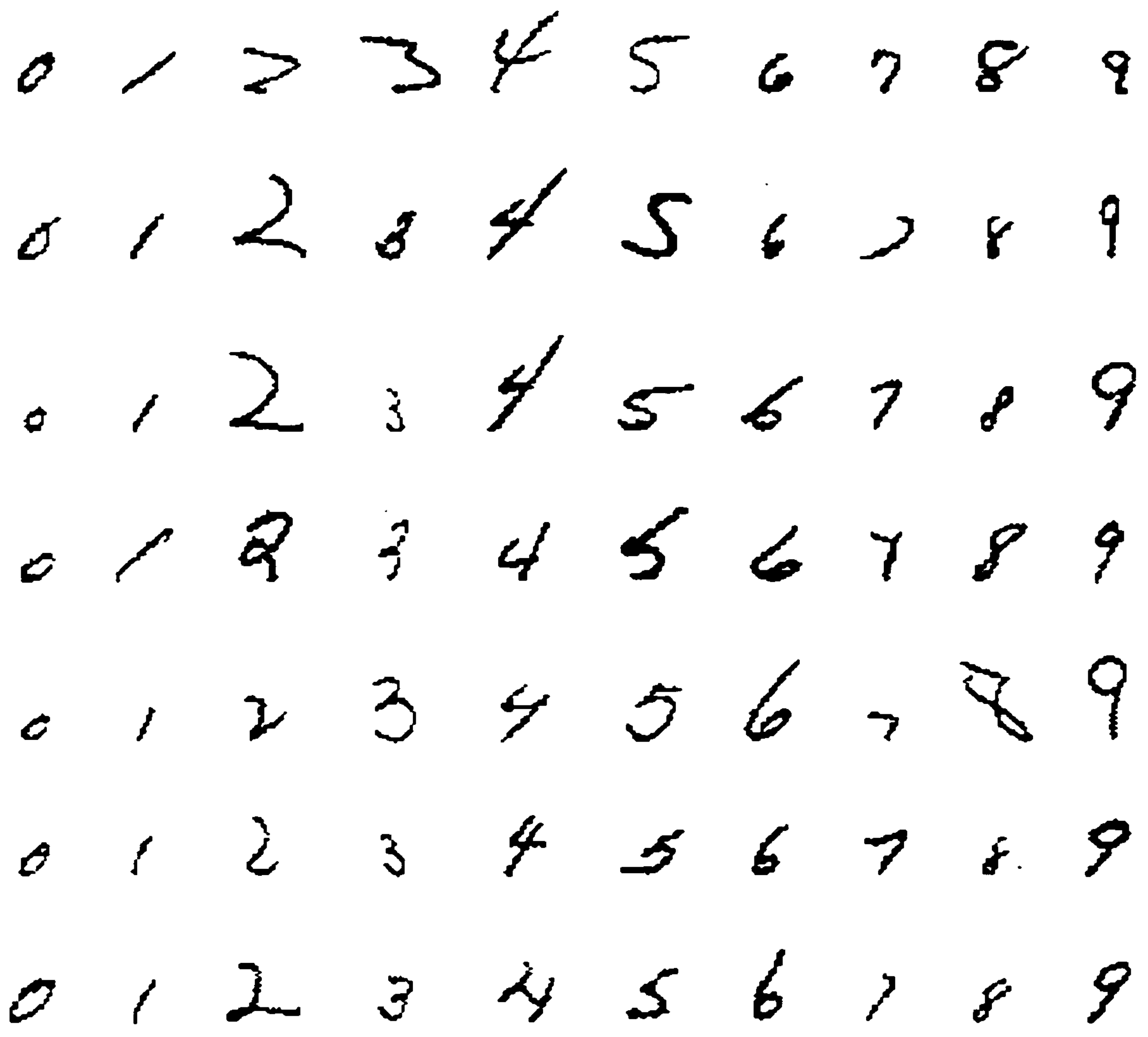


Figure 43: Representative Samples of CENPARMI Database for Isolated Numerals

C.2 NIST and MNIST Databases for Isolated Numerals

NIST databases for isolated numerals come from the NIST Special Database 19 provided by National Institute of Standards and Technology in U.S.A. The CD ROM contains 3,699 full page binary images and 814,255 handwritten numerals and alphabetic characters from those forms. A sample image is provided in Figure 44.

The commonly used databases for comparison of algorithms for UIHN are the training set Special Database 3 (SD3) and test set Special Database 7 (SD7). SD7 is also known as Test Database 1 (TD1) in the literature. Both databases were originally for a competition on First Census Optical Character Recognition Systems Conference in May 1992 organized by NIST. The event was for assessing the state-of-the-art in OCR. The SD3 has 223,122 numerals while the TD1 has 58,646 numerals.

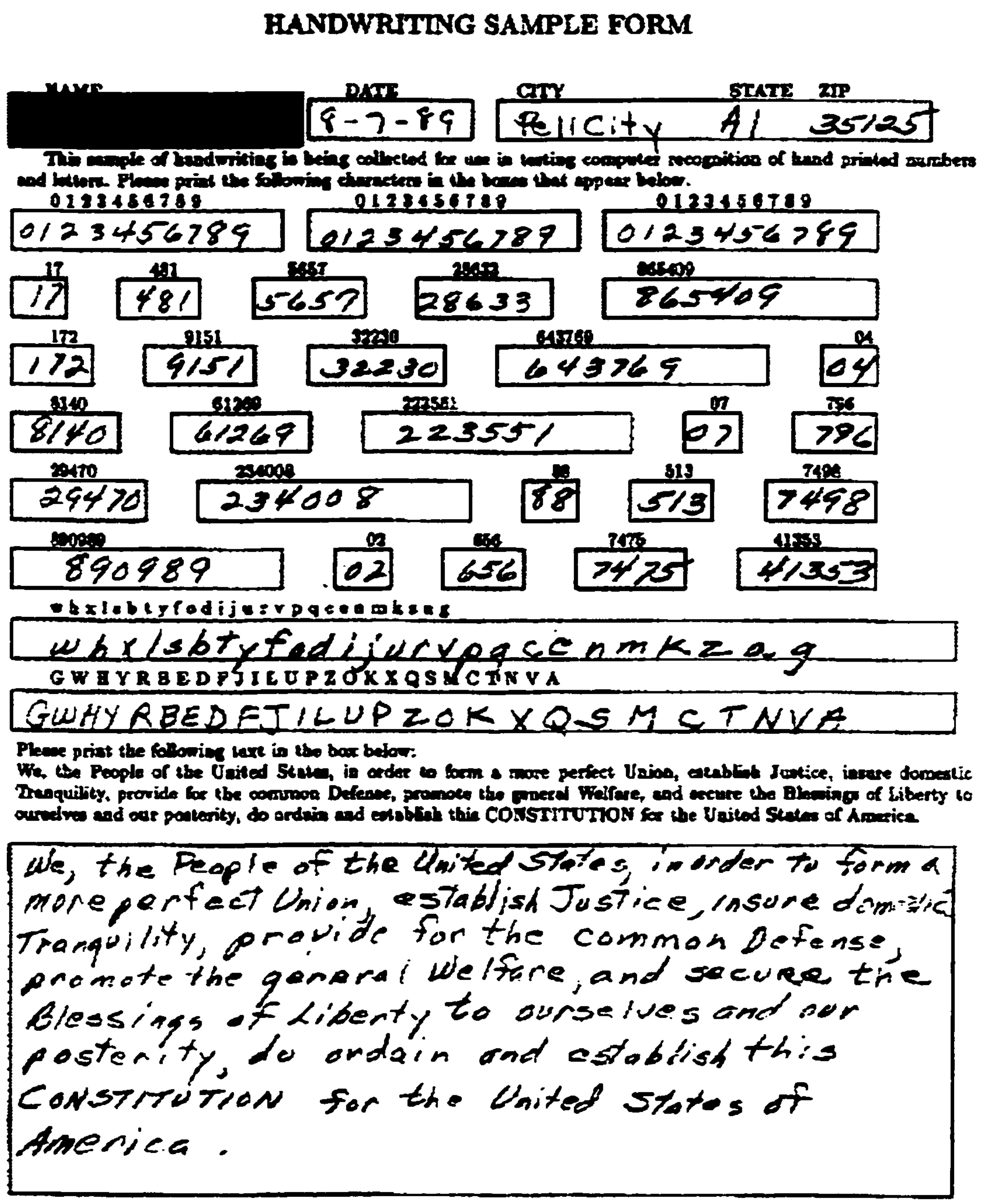
The problem with these two databases is that SD3 did not constitute very appropriate training material for TD1 test set. Indeed, cross validation studies performed after the competition indicated that TD1 was more diverse and more general. The observation is confirmed by the fact that the top entry of the NIST competition was not trained with SD3. These two databases differ due to the fact that SD3 was collected among Census Bureau employees, while TD1 was collected among high-school students.

This motivated researchers to build a Modified NIST database (MNIST) based on the large amount of data in SD3 and SD7 by mixing the two.

In this thesis, the training set of MNIST contains 20,000 numeral from SD3 and 48,646 from TD1. The test set has 5,000 images from SD3 and 10,000 from TD1. After shuffling, the MNIST for UIHN has a training set of 68646 images and a test set of 15000 images.

C.3 CEDAR Database for Isolated Numerals

The CEDAR database for isolated numerals is available on CEDAR CDROM 1. It consists of 21,179 binary images of isolated numerals. The resolution is 300 dots per inch. Training data is in *br* directory and includes a total of 18,468 samples. Balanced training sets, of 200 samples per class, are labelled as *cedar 1, cedar 2, ... cedar 7.* Testing data is in the *bs* directory and contains 2711 samples. A subset of 2,213 well



HANDWRITING SAMPLE FORM

NAME	DATE	CITY	STATE	ZIP
[redacted]	9-7-89	Fell City	Al	35125

This sample of handwriting is being collected for use in testing computer recognition of hand printed numbers and letters. Please print the following characters in the boxes that appear below.

0123456789	0123456789	0123456789		
0123456789	0123456789	0123456789		
17	481	5657	28633	865409
17	481	5657	28633	865409
172	9151	32230	643769	04
172	9151	32230	643769	04
8140	61269	223551	07	796
8140	61269	223551	07	796
29470	234008	88	513	7498
29470	234008	88	513	7498
890989	02	656	7475	41353
890989	02	656	7475	41353

w b x l s b t y f o d i j u r v p q c e n m k z a g

wbxlsbtyfodijurvpqcenmkzag

G W H Y R B E D F J I L U P Z O K X Q S M C T N V A

GWHYRBEDFJILUPZOKXQSMCTNVA

Please print the following text in the box below:
We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

We, the People of the United States, in order to form a more perfect Union, establish Justice, insure domestic Tranquility, provide for the common Defense, promote the general Welfare, and secure the Blessings of Liberty to ourselves and our posterity, do ordain and establish this CONSTITUTION for the United States of America.

Figure 44: Example Form Image in NIST SD 19 CDROM. This is the file *hsf_page/hsf_0/f0165_04.pct*.

segmented samples were extracted from the *bs* directory and are known as CEDAR *goodbs* database. A detailed description can be found in [46].

C.4 IRIS-BELL'98 Database

The IRIS-Bell'98 database is the result of efforts of many researchers in CENPARMI. It consists of two projects: IRIS-cheque and Bell'98 which are images of cheques and bill documents respectively.

The IRIS-cheque includes samples of Canadian personal cheques gathered as a project financially sponsored by IRIS (Institute for Robotics and Intelligent Systems), with the courtesy of Interchèques Inc. and Bell Canada. Many employees at Concordia University also helped collecting the data images. Bell'98 includes samples of phone bills written by Bell employees and the general public. The number of documents are listed in Table 33.

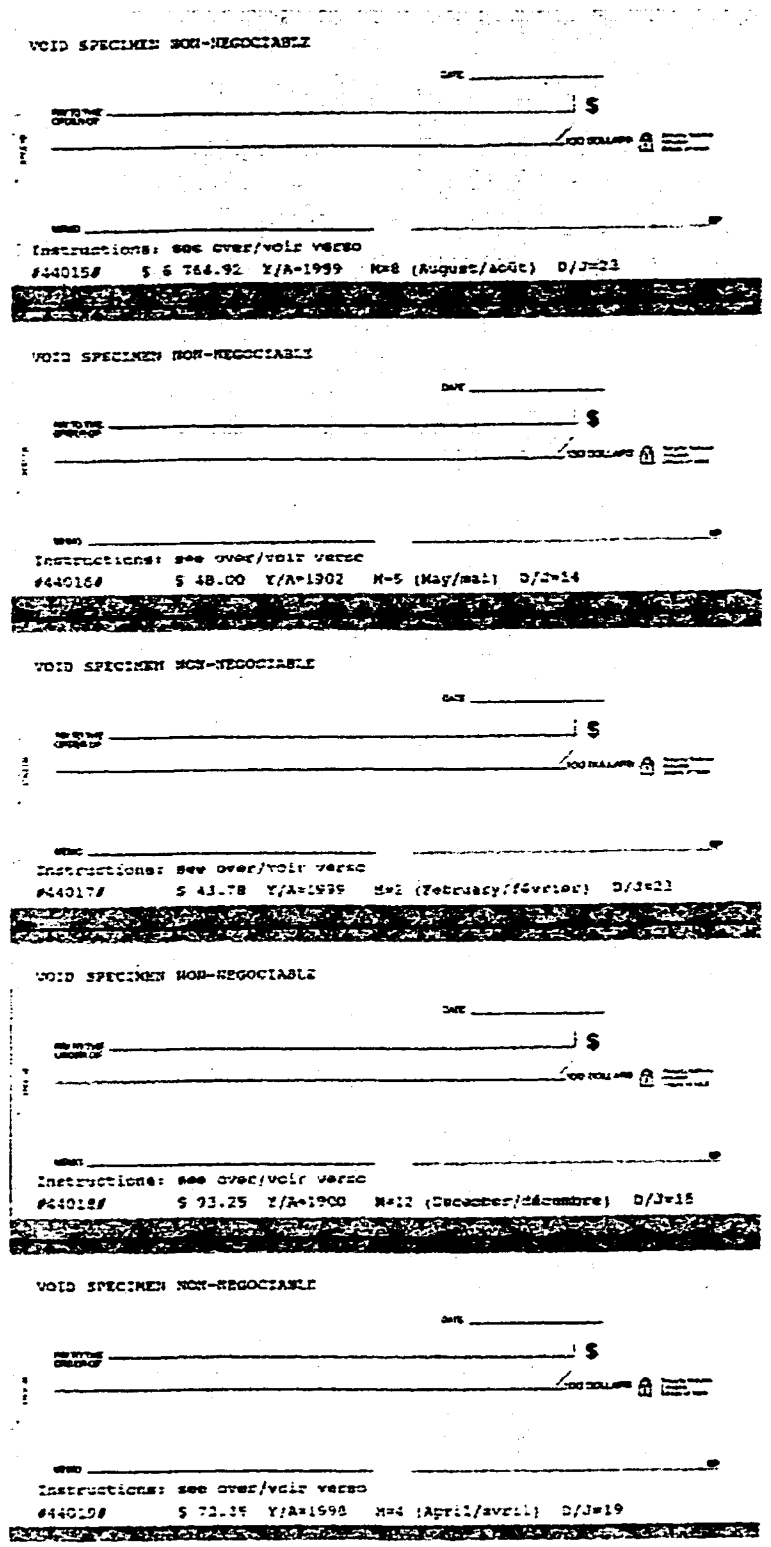
Databases	IRIS Cheque		Bell'98
Related Facility	Concordia Univ.	Bell Canada	Bell Canada
No. of Cheques/Bills	6,284	7,070	5,932
Source of Writers	Employees/Students	Employees	General Public

Table 33: IRIS Cheque Database and Bell'98 Database

To describe the whole database is not an easy task. Here I will focus on the building of IRIS-cheque database from several stages including cheque preparation, collection, scaning and tagging.

Cheque Preparation

Every sheet for collecting data consists of a series of 5 cheques. A sample of such data sheet is given in Figure 45. The sheets were provided by Interchèques Inc. which prints real cheques for banks, so the cheques actually have the same format and pattern as real ones. Totally 50,000 cheques were printed.



VOID SPECIMEN NON-NEGOCIABLE

$

/100 DOLLARS

Instructions: see over/voir verso

#44015# $ 6 764.92 Y/A=1999 M=8 (August/août) D/J=23

[redacted]

VOID SPECIMEN NON-NEGOCIABLE

DATE

$

/100 DOLLARS

Instructions: see over/voir verso

#44016# $ 48.00 Y/A=1902 M=5 (May/mai) D/J=14

[redacted]

VOID SPECIMEN NON-NEGOCIABLE

DATE

$

/100 DOLLARS

Instructions: see over/voir verso

#44017# $ 43.78 Y/A=1939 M=2 (February/février) D/J=22

[redacted]

VOID SPECIMEN NON-NEGOCIABLE

DATE

$

/100 DOLLARS

Instructions: see over/voir verso

#44018# $ 93.25 Y/A=1900 M=12 (December/décembre) D/J=15

[redacted]

VOID SPECIMEN NON-NEGOCIABLE

DATE

$

/100 DOLLARS

Instructions: see over/voir verso

#44019# $ 72.35 Y/A=1998 M=4 (April/avril) D/J=19

[redacted]

Figure 45: Sample Sheet of IRIS-Bell'98 Data Collection

The instructions for filling the cheques were then printed on the back of each sheet in English and French. Information lines were printed at the bottom of each cheque. As we can see from the the sheet sample, information lines contain the following:

- The line of "VOID SPECIMAN NON-NEGOTIABLE" at the left top of each cheque.
- Cheque sequence number such as "#44015#", is used to keep track of the data collection and image storage.
- Desired amount. The amounts are generated as a controlled random sequence. The control pattern is designed in such a way that (1) Each digit can appear at different positions of a numeral string. (2) The bigger the amount (longer numeral string), the smaller is its occurrence probability. All amounts are less than one million.
- Desired date zone that contains year, month and day.

Each writer fills two sheets, i.e., 10 cheques. The cheques are filled by ink. Languages can be English or French. The amount and date in the information line are supposed to be used to fill the cheque. To make the resulting image closer to a real cheque, each region is to be filled. The cheque can be made payable to person or organization of any choice. It is also required to write something on the signature line (not necessarily the real signature).

Data Collection

The data collection was done at two places: Concordia University and Bell Canada.

Cheques with sequence numbers from 0 to 27,000 were filled by Concordia University employees with different backgrounds, including faculty and staff members at different departments with various levels of education.

An explanation sheet was given to each writer, explaining the purpose of this collection — we are just collecting handwriting samples. There was a small gift draw to encourage more writers. Not every cheque sent out came back. There are people who feel it would be related with personal information and are reluctant in filling the cheques. As a result, 6,284 cheques were filled and returned out of the total number of 20,000 we sent out in Concordia University. Most of the cheques are filled in English.

Cheques with sequence numbers from 30,000 to 66,000 were filled by employees of Bell Canada in Montreal (some cheque writers are out of the region of Montreal). 7,070 cheques were returned and we got a majority of French cheques.

Scanning & Tagging

The filled cheque sheets are scanned at 300 dots per inch in gray level. The sheet image is automatically separated into 5 sub cheque images. The file name of the saved cheque image corresponds to the cheque sequence number. The image files are stored in an English or French directory according to the language of legal amount and date.

The courtesy amounts of the cheques are then extracted automatically and saved as binary images.

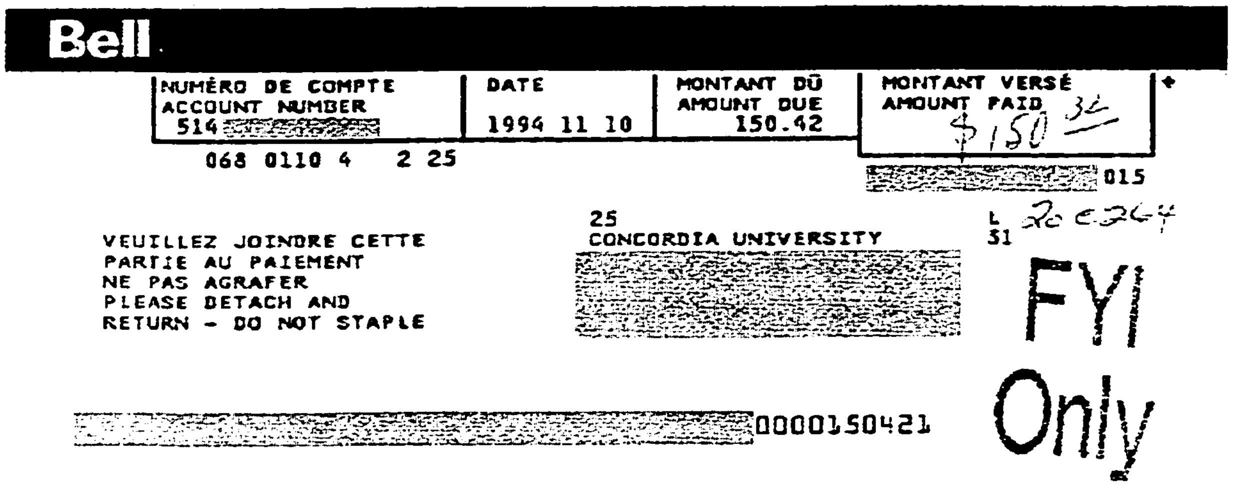
The tagging of courtesy amounts is a semi-automatic process with the assistance of a recognizer for UTHN and a known truth file created from the information lines of the cheques (the truth file may be wrong since it is possible that people didn't fill the desired contents.)

The tasks of tagging the courtesy amount include (1) separate the numeral string into isolated and touching numerals; (2) label the isolated or touching numerals with correct identities either from truth file or from recognizer, whichever is right; (3) put them into different directories.

When the conclusion of a numeral string comes out from a recognizer, it is compared with the truth file. If they agree with each other, the tagging and file storage are automatically done by the program. If they do not agree, the error may come from truth file or recognizer or both. Then the operator manually looks at the cheque image. If the computer recognition results are correct, just update the truth file since the correct tagging has been done by the program. If the recognition results are wrong, another tagging GUI program is used to manually tag the courtesy amount of the cheque image and save the isolated and touching numerals into different directories.

The legal amount and date zone are also automatically extracted from individual gray cheque images with background and noise removed, and saved as binary images. Each legal amount and date image is manually tagged. At the right end of legal amount image, there are often cents part written in numerals (usually touching pairs) which can be solved by the schemes proposed in this thesis for UIHN and UTHN, so are some numeral formats in date zones.

Another database — Bell'98 is the result of a contract between Bell Canada and CENPARMI. A sample of Bell'98 bill is given in Figure 46. Courtesy amounts are extracted and tagged in a similar way as in IRIS-Cheque project.



Bell

NUMÉRO DE COMPTE ACCOUNT NUMBER	DATE	MONTANT DÛ AMOUNT DUE	MONTANT VERSÉ AMOUNT PAID
514 [redacted]	1994 11 10	150.42	$150 $\frac{32}{}$

068 0110 4 2 25

[redacted] 015

VEUILLEZ JOINDRE CETTE
PARTIE AU PAIEMENT
NE PAS AGRAFER
PLEASE DETACH AND
RETURN - DO NOT STAPLE

25
CONCORDIA UNIVERSITY
[redacted]

L 20 e 264
51

FYI
Only

[redacted] 000015042l

Figure 46: Sample of Phone Bill Form. Notice that some fields have been intentionally occluded.

Conclusively, the IRIS-Bell'98 database consists of numeral strings from courtesy amounts, words from legal amounts and date zone images. The touching numeral pairs are being used in this thesis work. The IRIS-Bell'98 for UTHN has a training set of 2538 touching numeral pairs and a test set of 1193 pairs.

C.5 NIST Database for Touching Numerals

Building the NIST database for touching numerals is one of the contributions of this thesis. The building process has been described in Chapter 5. Here I summarize the property of this database concisely.

The NIST for UTHN database is built on NIST CD ROM Special Database 19. Unlike isolated numerals, there is no well-organized UTHN images on the CD ROM. We first extract the numeral strings from the form images, then separate them into isolated and touching numerals. We are only interested in touching cases. The identities of the touching numerals are tagged semi-automatically, then manually checked and stored in Cenparmi Binary Code (cbc) format.

The training set for touching numeral pairs has 4,252 images. The test set consists of 4,395 images.

Appendix D

Filtering Rule

The Beta-Binomial Model is used. Let P be the probability of getting correct recognition results using middle zone candidate as the cutting point. Assume *a priori* distribution of P as Beta distribution Beta(a,b). The beta family of distribution is a continuous family on (0,1) indexed by two parameters a & b, which decide the final shape of the distribution. *pdf* is

$$f(p) = p^{a-1}(1-p)^{b-1}/B(a,b) \tag{31}$$

where

$$0 < p < 1$$

$$B(a,b) = A(a)A(b)/A(a+b) \tag{32}$$

$$A(a) = \int_0^\infty x^{a-1}e^{-x}dx \tag{33}$$

Let *a priori* distribution be a=b=1, which is a uniform distribution, indicating no advantage of adopting middle zone cutting points. It is then modified by Bayes inference:

For a fixed p, we have p chance to get the correct result when using the middle zone cutting point, and 1-p to lose. Let n be the total number of experiments and Y the number of successes. Y is a Binomial distribution:

$$f(Y|n,p) = \binom{n}{y} p^y (1-p)^{n-y} \tag{34}$$

We can derive *posteriori* distribution:

$$\begin{aligned}
f(p|Y=y) &= \frac{f(Y|n,p) * f(p)}{f(Y)} \\
&= \frac{\binom{n}{y} p^y (1-p)^{n-y} p^{a-1} (1-p)^{b-1} / B(a,b)}{\int_0^1 \binom{n}{y} p^y (1-p)^{n-y} p^{a-1} (1-p)^{b-1} / B(a,b) dp} \\
&= \frac{p^{y+a-1} (1-p)^{n-y+b-1}}{B(a+y, b+n-y)} \\
&= Beta(a+y, b+n-y) && (35) \\
E(p|y) &= \frac{a+y}{a+b+n} && (36)
\end{aligned}$$

We can see that prior information (a,b) is now combined with test data (y,n). So we can compute $E(p|y)$ using the experimental data. High probability of getting correct result using middle zone cutting points can justify our rule of adopting cutting points close to the middle zone of touching pairs. In the training set of NIST for UTHN, we get a result of 0.91, which is adequate to prove the validity of the rule.

Appendix E

Confusion Matrix of UIHN Recognition

	0	1	2	3	4	5	6	7	8	9	Total	Error
0	973	0	0	2	0	0	3	0	2	1	982	8
1	2	1108	1	3	0	1	0	0	3	1	1126	11
2	0	9	964	1	0	0	0	1	1	1	990	13
3	1	3	3	993	0	5	0	0	0	2	1014	14
4	0	2	0	0	932	0	4	2	0	7	952	15
5	5	2	0	6	0	895	3	0	3	0	925	19
6	4	5	0	0	6	4	979	0	1	0	1004	20
7	0	5	8	5	11	1	0	959	2	11	1028	43
8	1	2	7	3	3	3	7	1	947	8	997	35
9	2	3	0	4	7	3	1	2	4	942	982	26
Total	988	1139	983	1015	959	912	997	965	963	973	10000	-
Error	15	31	19	24	27	17	18	6	16	31	-	204

Table 34: Confusion Matrix of General Purpose Recognizer for UIHN, for a total of 10,000 MNIST samples (including rejection). The first column shows the real identity of the sample while the top row is the result identity.

	0	1	2	3	4	5	6	7	8	9	Total	Error
0	948	0	0	0	0	0	2	0	2	0	982	4
1	0	1086	1	3	0	0	0	0	1	0	1126	5
2	0	3	938	0	0	0	0	1	1	0	990	5
3	0	2	2	953	0	3	0	0	0	1	1014	8
4	0	1	0	0	925	0	0	2	0	6	952	9
5	2	1	0	5	0	873	1	0	3	0	925	12
6	3	1	0	0	3	2	967	0	1	0	1004	10
7	0	1	4	2	6	0	0	950	0	7	1028	20
8	0	2	3	0	1	3	1	0	933	2	997	12
9	1	1	0	3	5	1	1	2	3	927	982	17
Total	954	1098	948	966	940	882	972	955	944	943	10000	-
Error	6	12	10	13	15	9	5	5	11	16	-	102

Table 35: Confusion Matrix of Verifier for UIHN, for the same 10,000 MNIST samples. We can see that the verifier reduces the errors substantially while maintaining a reasonable recognition rate.