HANDWRITTEN DIGIT STRING SEGMENTATION AND RECOGNITION USING DEEP LEARNING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ORCUN ELITEZ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

DECEMBER 2015

Approval of the thesis:

HANDWRITTEN DIGIT STRING SEGMENTATION AND RECOGNITION USING DEEP LEARNING

Submitted by **ORÇUN ELİTEZ** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Gönül Turhan Sayan _____
Head of Department, **Electrical and Electronics Engineering**

Prof. Dr. Uğur Halıcı _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Aydın Alatan _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Uğur Halıcı _____
Electrical and Electronics Engineering Dept., METU

Prof. Dr. Gözde Bozdağı Akar _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Elif Vural _____
Electrical and Electronics Engineering Dept., METU

Assist. Prof. Dr. Tolga İnan _____
Electrical and Electronics Engineering Dept., TED University

Date: 08.11.2015

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: ORÇUN ELİTEZ

Signature:

ABSTRACT

HANDWRITTEN DIGIT STRING SEGMENTATION AND RECOGNITION USING DEEP LEARNING

Elitez, Orçun

M.S., Department of Electrical and Electronics Engineering

Supervisor: Prof. Dr. Uğur Halıcı

December 2015, 79 pages

The main purpose of this thesis is to build a reliable method for the recognition of handwritten digit strings. In order to accomplish the recognition task, first, the digit string is segmented into individual digits. Then, a digit recognition module is employed to classify each segmented digit completing the handwritten digit string recognition task.

In this study, a novel method, which uses deep belief networks architecture, is proposed in order to achieve high performance on the digit string segmentation problem. In the proposed method, images of digit strings are trained into a DBN structure by sliding a fixed size window through the images labelling each sub-image as a part of a digit or not. After the completion of the segmentation, in order to achieve the complete recognition of handwritten digit strings, the segmented digits are classified using both DBN algorithm and support vector machines and the results of these algorithms are compared over CVL Digit Strings Dataset. The result of the

segmentation which uses the proposed method is compared with the result of the segmentation algorithm using water reservoir concept. Moreover, the results of some benchmark algorithms which use the same database of handwritten digit strings are included in the comparison.

The proposed method outperformed the state of the art methods and also the baseline algorithm using water reservoir concept for digit segmentation on the CVL Digit Strings Dataset.

Keywords: Handwritten Digit String Segmentation, Touching Numeral Segmentation, Handwritten Digit Recognition, Deep Learning, Deep Belief Networks, Artificial Neural Networks, Pattern Recognition

ÖZ

DERİN ÖĞRENME YÖNTEMİ KULLANARAK EL YAZISI RAKAM DİZİLERİNİ BÖLÜTLEME VE TANIMA

Elitez, Orçun

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği

Tez Yöneticisi: Prof. Dr. Uğur Halıcı

Aralık 2015, 79 sayfa

Bu tezin amacı el yazısı rakam dizilerinin tanınabilmesi için güvenilir bir yöntem geliştirilmesidir. Tanımayı yapabilmek için öncelikle rakam dizileri bağımsız rakamlara ayrılmalıdır. Daha sonra bir rakam tanımlama yöntemi kullanılarak birbirinden ayrılmış rakamlar tanınır. Sonuç olarak el yazısı rakam dizisi tanıma işi gerçekleşmiş olur.

Bu çalışmada, rakamları bölütlemedeki başarıyı arttırmak için derin öğrenme metodu kullanılarak yeni bir yöntem geliştirilmiştir. Bu yöntemde, sabit büyüklükteki bir pencere rakam dizilerinin bulunduğu resim üzerinde kaydırılarak, bu pencerenin kapsadığı bütün parçalar derin yapay sinir ağına öğretilmektedir. Bu parçalar rakamın bir parçası veya rakamlar arasındaki geçişin bir parçası olarak sınıflandırılır. Bölütleme işlemi tamamlandıktan sonra ayrılmış rakamlar, rakam sınıflandırma

algoritmasına girdi olarak verilir. Rakam sınıflandırma işlemi için DBN (Deep Belief Networks) ve SVM (Support Vector Machines) yöntemleri uygulanmış ve bunların sonuçları karşılaştırılmıştır. Aynı zamanda önerilen bölütleme algoritması da başarısı kanıtlanmış başka algortmalar ile karşılaştırılmıştır. Bu karşılaştırmalara aynı veriseti üzerinde uygulanmış referans yöntemlerin sonuçları da eklenmiştir.

Önerilen yöntem, CVL veriseti üzerinde yapılan deneylerde, gelişmiş yöntemlere ve 'water reservoir' kavramını kullanan referans rakam dizisi bölütleme algoritmasına göre daha iyi sonuç vermiştir.

Anahtar Kelimeler: El Yazısı Rakam Dizisi Bölütleme, El Yazısı Rakam Tanıma, Derin Öğrenme, Yapay Sinir Ağları, Örüntü Tanıma

To Çilem

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor Prof. Dr. Uğur Halıcı for having provided me with her expertise and wonderful insights on this study. Her valuable supervision, advice and guidance made me a better researcher. I count myself lucky for having the opportunity to work under her supervision.

I am also grateful to my thesis committee members Prof. Dr. Aydın Alatan, Prof. Dr. Gözde Bozdağı Akar, Assist. Prof. Elif Vural and Assist. Prof. Tolga İnan for their helpful criticism and suggestions.

My warmest thanks to Çilem Çiçek, besides her contributions on proof-reading this thesis, for her unconditional love, patience, care, motivation and most importantly morale support at every stage of this study.

I would like to express my endless appreciation to my mother Aliye Elitez, my father Kamil Elitez and my sister Buse Elitez for their morale support throughout this study.

I am deeply grateful to Mr. Haluk Gökmen, the general manager of Enekom, for being understanding and supportive during this study. I am also grateful to my colleagues at Enekom for their morale support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

DBN	(Deep Belief Networks)
RBM	(Restricted Boltzmann Machine)
SVM	(Support Vector Machines)
CV	(Confidence Value)
TDNN	(Time Delay Neural Network)
RBF	(Radial Basis Function)
HNN	(Hopfield Neural Network)
MLP	(Multilayer Perceptron)
MDRNN	(Multidimensional Recurrent Neural Networks)
k-NN	(k-Nearest Neighbours)
CCA	(Connected Component Analysis)
SOM	(Self-Organizing Map)
BP	(Base Point)
IP	(Interconnection Point)
ANLD	(Average Normalized Levenshtein Distance)
NLD	(Normalized Levenshtein Distance)
KKT	(Karush-Kuhn-Tucker)

CHAPTER 1

INTRODUCTION

1.1. Motivation and Objective

The purpose of machine learning is to sense, remember, learn and recognize like a human being. One of the most in demand methods of machine learning is deep learning. The idea behind deep learning emerged back in 1950s with the definition of perceptron. Perceptron was the first machine that had the capability to sense and learn. Then, the multilayer perceptron structure with limited number of hidden layers was defined in 1980s. However, the learning capability of the perceptron was very limited. Thus, it was disregarded for years until the proposition of neural networks with many hidden layers in 2000s. The connectionism introduced by multiple levels of representation has brought better learning capability to the architecture. Actually, the main aspect behind deep learning is this multiple levels of representation phenomenon. It is originated by the theory that the brain processes information in multiple layers. As the deep learning algorithms have been improved, the resemblance of the architecture to the biological neural networks has been increased.

A complete document processing system should include handwritten text recognition, digit string recognition, signature recognition and localization of the handwritten inputs. Most of the existing systems have been privatized towards the application, and the main problem observed with such systems is that they define a wide variety of rules to the authors. These rules include respective boxes for each field, writing rules, restraining touching characters in a string, restraining slant angle of the characters etc. Our main purpose is to reduce the number of rules and to avoid application-specific methods so as to approach a universal document processing system.

The field studied in this thesis spans the concept of handwritten text recognition and all the steps from digitizing the text image to complete recognition of the intended meaning of the text. Since handwritten text recognition contains a wide variety of options that the text is written, the main concern of this study will be recognizing handwritten digit strings, such as the courtesy field of the bank cheques.

Although, the performance of machine learning algorithms on handwritten digits are very good when the digits are segmented well, the segmentation performance of the existing algorithms are poor, which in turn reduces the recognition performance when digit strings are considered. Therefore, reliable handwritten digit string recognition methods are necessary in order to increase the recognition rate of handwritten digit strings.

The success of deep neural networks in handwritten digit recognition motivated us to use them also to decide vertical cuts for segmentation of digit strings by training the regions between the digits.

1.2. Problem Definition and Our Approach

The recognition of the images of handwritten digit strings contains three main steps which are preprocessing, segmentation of image into individual digits and classification of each digit forming the final recognition result. In this study, deep learning will be used both in segmentation and classification steps and the results will be compared with the results of other well-known algorithms.

There are a wide variety of studies in the field of handwritten text recognition. Generally digit classification and preprocessing steps are achieved to a certain level of success, however the segmentation step is still a challenging task for this field. In order to overcome this challenge, some rules that restrict the style of handwriting are applied in order to increase the success rate of the segmentation. In this study, the handwritten digit strings independent of writings styles are examined.

As a baseline algorithm for comparison of the performance, water reservoir concept [42] and [43] will be used for segmentation and support vector machines for classification. The reason for choosing water reservoir algorithm for segmentation is that it is a high performance segmentation algorithm which does not use recognition implicitly and the reason for choosing support vector machines as classifier is that it is named as the best performing classifier for handwritten digit recognition after deep belief networks [18]. Then, deep belief nets (DBNs) will be used to segment the same images. In the novel approach proposed in this thesis that the digits and the gaps between the digits will be fed into the deep belief net and the success rate will be compared to the baseline algorithm. The algorithms will be tested on CVL Digit Strings [21] database which contains 1262 images each having 5 to 7 digits. The reason why CVL dataset is chosen is that it is a dataset containing a large number of images of connected handwritten digits.

In terms of classification of the digits, the segmented digits from the previous step will be classified by a pre-trained SVM and DBN classifiers. The classifiers will be trained using both MNIST database which contains already segmented handwritten digits and segmented digits from the CVL Strings database obtained in the previous step.

1.3. Contribution

In this study, we proposed a novel handwritten digit segmentation technique which is a recognition-based technique. Instead of focusing of extracting characters from the digit strings, we trained DBN to detect the gaps between the digits and used these gaps to segment the digits. An image of digit strings was divided into sub-images that contain the information that it is whether a gap or a part of a numeral. The sub-images then were used to train a classifier. Then the classifier was used to identify parts of an image.

The tricky part of this study is the behaviour of the touching pairs multiple times. Our approach extremely outperformed the baseline algorithm.

1.4. Organisation of the Thesis

This section, the outline of chapters and the topics to be covered will be given. In this chapter, the objectives of the thesis with our motivation and the contributions to the field of study has already been introduced.

Chapter 2 consists of the literature survey and background information about the main focus of this thesis. This chapter includes an overall review of the literature about handwritten digit string recognition and the techniques used in the steps of the handwritten digit string recognition system. The techniques used for preprocessing, segmentation and recognition parts are given respectively.

Chapter 3 includes the details of the proposed approach and experimental results. The proposed approach is introduced and the experiments done in order to observe the performance are explained. Then the results of the experiments are presented.

Finally, chapter 4 provides a conclusion of this thesis with a brief summary. The experimental results are discussed with some theoretical background and possible future works are explained.

CHAPTER 2

LITERATURE SURVEY AND BACKGROUND INFORMATION

2.1. Literature Survey

In this part, the solutions towards recognition of handwritten digit strings is examined. A critical analysis is carried out for the methods used solving this problem. A top-down research strategy will be adopted.

Automatic processing of handwritten text images is very crucial for creating document processing systems for bank cheques, government records etc. [1]. The main goal is to increase the reliability of the result of recognition process and to speed up the routine of collecting training and test data from handwritten digit strings.

Main focus of this study is recognition of handwritten digit sequences without restricting the writer with rules. The recognition of the segmented digits is an easier task compared to segmentation of the sequence. The overall recognition process consists of preprocessing, segmentation and classification modules.

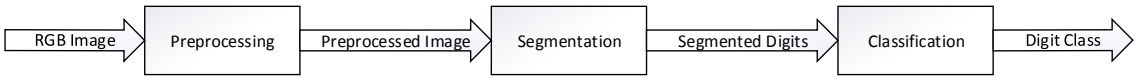


Figure 2.1: Modules of Handwritten Digit Recognition

2.1.1. Preprocessing

Figure 2.2: Steps of Preprocessing

The preprocessing could start with the binarization of the image if binary data is needed for the rest of the techniques used, as it is, in fact, needed in most of the cases. First the image must be converted to grayscale image if the image input is in RGB format. Hue and saturation are discarded and intensity is used to obtain grayscale image. Then the grayscale image should be turned into binary form. There are two main categories of binarization methods. These are global and local thresholding.

Global thresholding algorithms use a single threshold value for the overall image while the local thresholding algorithms use different threshold values for each pixel using their spatial information [1]. Twenty global thresholding techniques were compared by Sahoo et al. [2] based on uniformity and shape measures. According to the comparison, Otsu's thresholding method [3] gave the best performance. There are various local thresholding techniques as well. Trier and Jain [4], Sezgin and Sankur [5] surveyed and discussed these thresholding techniques.

After the binarization, skew correction could be done in order to correct the angle of the digits and the X-axis. Knerr et al. [6] determined the angle by computing the pixel densities between ±5 degrees with the help of horizontal guidelines. With the help of the histogram created using the pixel densities, the histogram with the longest peak is

chosen as the angle of the text in the image. The problem with this method is that it relies on the fact that there is a horizontal guideline to realize the actual angle.

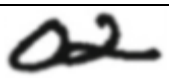
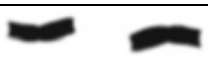
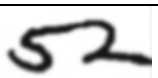
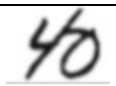
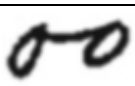
2.1.2. Segmentation

As stated earlier, segmentation of the handwritten digit strings is the most challenging problem of the recognition process. The problem arises from the fact that the number of digits, size of each digit and the gap between the digits are unknown [7].

A complete digit segmentation algorithm should be successful with all types of digit connections. It is easier to segment the string if there is a gap between the digits or there is a writing rule such as a box for each digit. These type of string could be separated by using a trivial algorithm such as applying connected component analysis after removing noises.

However, the intention of this study is to implement a complete handwritten digit string segmentation algorithm. Therefore, digit strings containing touching numbers should be considered as well. Chen and Wang [15] proposed that there are 5 types of touching digit strings that could be categorized as multiple-touching and single-touching digit strings. Examples of each type are shown in Table 2.1.

Table 2.1: Types of touching digit strings. (Adapted from [3])

Category	Type	Style of touching	Examples	
Single-touching	I.			
	II.			
	III.			
	IV.			
Multiple-touching	V.			

Segmentation algorithms could be categorized as segmentation-then recognition and recognition-based algorithms [8]. The segmentation-then recognition based algorithms, first, extract segmented images where each segmented region is assumed to contain a single character and then these regions are given to the classifier. The recognition-based algorithms provide a list of segmentation results. Then all of the results are submitted to the classifier. The computational cost of the recognition-based algorithms are very high, since all of the options from the segmentation process must be classified. However, they generate better results [7]. Additionally, the recognition module must classify fragments, isolated characters and connected characters.

Recognition-based segmentation can be categorized as explicit segmentation and implicit segmentation. In explicit segmentation, segmentation generates candidate

characters for the recognizer. In the implicit segmentation, segmentation and recognition are performed simultaneously.

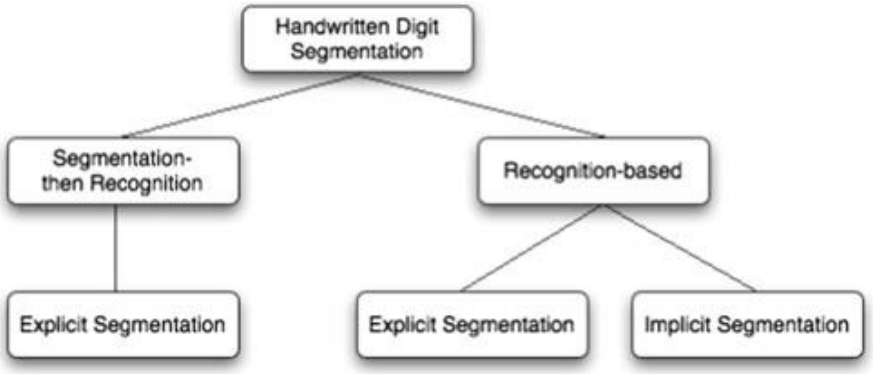


Figure 2.3: Methods of segmentation of digit strings (adapted from [9])

The literature shows that explicit segmentation have better results. The main problem with the implicit segmentation is that it is highly sensitive to the slanted images.

Fujisawa et al. [10] proposed a recognition-based algorithm. According to the algorithm, first connected components are detected. Then a classifier classifies each segmented image as a digit or a string of digits. The classifier in this algorithm is depends on the horizontal length of the segmented images. In order to segment the string of digits that was not classified earlier, contour information is divided into upper and lower contours. Then an approximate vertical width is computed and candidate segmentation points are assigned to whose vertical length exceeds the threshold. Moreover, for the touching loops such as 8-8, 0-9 inner loops are divided into two groups and the distance between them is calculated. If the distance is greater than a threshold, segmentation occurs. Segmentation cuts are produced using line segments. Then the segmentation points are used in order to build a segmentation graph. Shortest

path of the graph gives the best option for the segmentation. This algorithm is tested by Fujisawa et al. [10] with 920 custom images. In these images, touching pairs are used. However, digits touching to each other multiple times were not used. 95% success rate in terms of segmented digits is calculated by the authors. The success rate is calculated by using a commercial optical character reader (OCR) for each segmented digit.

Fenrich and Krishnamoorthy [11] applied a trivial recognition-based method for the segmentation problem. Vertical histogram projection and valleys and peaks of the contour are used. First vertical histogram projection is applied to segment the digit string. The column with the minimal value is considered as a candidate for the vertical segmentation line. If this minimal value is larger than the minimum stroke width or multiple crosses occur through the segmentation line, the segmentation terminates.

If there is no candidate segmentation points from the first part of the algorithm where the vertical histogram projection is applied, upper and lower contours are used to define the segmentation points. If a peak of the lower contour and a valley of the upper contour could be connected with a line segment that satisfies slope thresholds, then a piecewise linear split is made. If they cannot be connected with a line segment that satisfies the slope threshold, then a straight line is used to generate the segmentation cut [7]. A success rate of 94.9% is calculated by the authors with 450 images of ZIP codes. Although solution is very trivial, it inspired many other researchers.

Yu and Yan proposed a method using morphological structuring technique for segmentation [12]. In the preprocessing step, smoothing and linearization are done and structural points of the image contours which are used in defining segmentation cuts are detected. If the digit string consists of more than two members, then the region containing left two numerals is determined and string of two numerals is separated. Then the same method is applied to the rest of the image until there is no region of two members left. The algorithm contains a lot of rules and heuristics. The success rate is calculated as between 85 to 95% on 3287 images which were extracted from NIST database depending on the length of the string.

Elnagar and Alhajjb [13] proposed a segmentation-recognition type algorithm to split pair of digits. Normalization and thinning are applied before segmentation. Although thinning is computationally expensive, according to authors thinning is necessary to get uniform stroke width to simplify detection of features. Since thinning process creates spurious points, a noise reduction algorithm is applied before the segmentation algorithm. For the segmentation, heuristics are defined to obtain most probable segmentation cuts. The success rate of the segmentation calculated by the authors was 96% with CEDAR and NIST datasets.

Suwa and Naoi proposed a segmentation-recognition based algorithm that uses the skeleton of the image as input [14]. Edges and vertices are extracted from this skeleton and a connected graph is formed. As in [11], possible segmentation points are found based on peaks and valleys of the skeleton. The segmentation path is computed using graph theory methods. The authors reported that the success rate is 88.75% with 2000 images form NIST SD19 dataset.

Sadri et al. combined a recognition-based algorithm with a genetic algorithm [9]. Before detecting the segmentation points, it uses a classifier to identify parts of digits, isolated digits and pairs of digits from the connected components. Segmentation paths are created using background and foreground skeleton. Then, all segmentation candidates are combined into a graph and a genetic algorithm searches all possible outputs. 96.5% success rate with 5000 touching pairs from NIST SD19 database.

Ribas et al. [7] observed that although some algorithms have lower performances compared to the others, they may be able to achieve segmentation with some samples that others cannot. These algorithms could be used developing more reliable segmentation models.

The segmentation algorithms using water reservoir concept proposed in [42] and [43] are the ones used as baseline in this thesis and they are explained in the next section.

2.1.2.1. Segmentation using Water Reservoir Concept

Pal et al. [42] proposed a new method for segmenting handwritten touching numerals based on a concept named water reservoir. The reservoir concept is used to identify the regions where digits touch each other.

The water reservoirs are obtained by dropping water from the top and bottom of an image and the locations where the water is accumulated are called the reservoir points. Using these reservoir points, the segmentation points are decided without any thinning and normalization processed done beforehand.

First connected components are extracted from the image, then it is decided whether a connected component is isolated or touching. Then the segmentation is applied to the touching numerals.

When two digits touch each, they create large reservoir space. The cutting points generally lies on the base of the reservoir. In order to get the best cutting scenario the attributes of the reservoir such as height and centre of gravity could be used.

Firstly, the reservoirs are obtained based on the structure of the image containing touching digits. These reservoirs are grouped into two types as top and bottom reservoirs. Observing the base positions of these reservoirs and according to the types of the reservoirs, touching position of the digits is decided [42]. Then the touching position, close loops, heights of the reservoirs and centre of gravity of the reservoirs help decide the best cutting point.

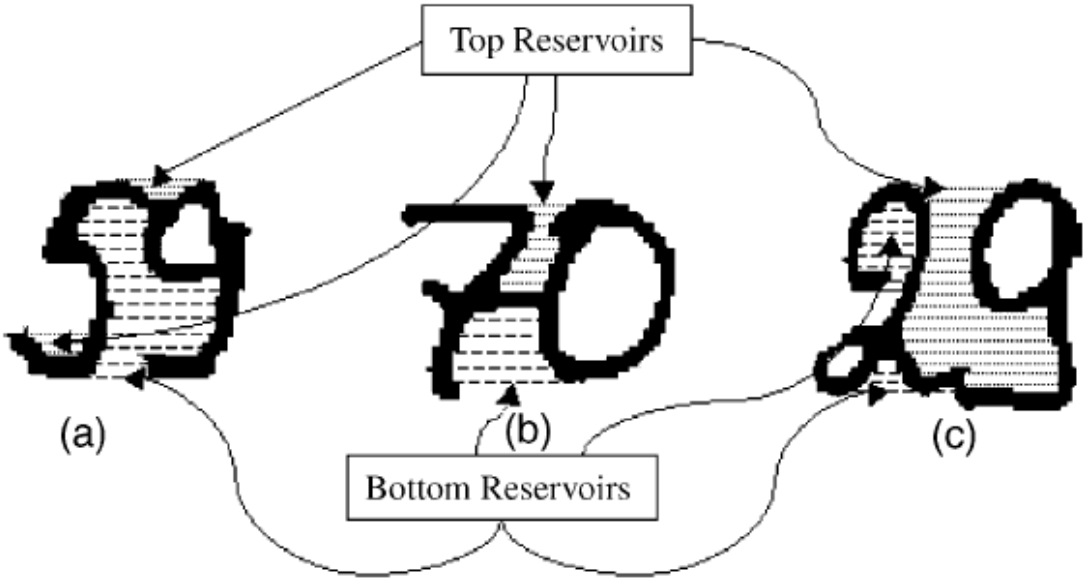


Figure 2.4: Examples of top and bottom reservoirs of different types of touching numerals (obtained from [42])

The reservoirs are formed by dropping water from the top of the image and filling the appropriate regions that could accumulate water and then the image is rotated upside down and water pouring is applied to the rotated image. A threshold, *T*, is assigned in order to limit the heights of the reservoirs appropriate for the segmentation process. In [42], the threshold value is chosen to be *1/8* of the height of the connected component.

The features chosen for segmentation by Pal et al. [42] are

1. The number of reservoirs
2. Location of reservoirs in the bounding box of the connected component
3. Size and shape of the reservoirs
4. Centre of gravity of reservoirs
5. Relative position of reservoirs
6. Number of closed loops

7. Locations of closed loops
8. Centre of gravity of the closed loops
9. The ratio of the height of the closed loops to the height of the connected component.

Before applying segmentation, the connected component is classified whether it is an isolated digit or a touching digit string.

Let

N = number of closed loops

$G_i(x,y) = Centre\ of\ gravity\ of\ i^{th}\ closed\ loop$

$W = total\ number\ of\ reservoirs$

$\overline{R_J} = j^{th}\ reservoir$

$\overline{G_J}(x,y) = Centre\ of\ gravity\ of\ j^{th}\ \overline{R_J}$

$\overline{R_{H_J}} = Height\ of\ \overline{R_J}$

$T = N + W$

$S_A = func_1\big(N, G_i(x,y)\big) = \begin{cases} 1, & N \geq 2\ and - 45° \leq \theta \leq 45° \\ 0, & otherwise \end{cases}$, where θ is the angle between the CGs of two closed loops.

$S_B = func_2\left(\overline{R_{H_l}}, \overline{G_l}(x,y)\right) = \begin{cases} 1, & \overline{R_{H_l}}\ \geq 75\%\ for\ \exists l\ and\ \overline{G_l}(x,y)\ \in h_m \\ 0, & otherwise \end{cases}$

Where h_m is the space between the horizontal lines that lies on 25% and 75% of the image. In other words, it is the middle portion of the image in the horizontal direction.

$S_C = func_1(W) = \begin{cases} 1, & W \geq 4 \\ 0, & otherwise \end{cases}$

The connected component is decided to be connected if S_A or S_C is 1. It is also determined as connected if S_B is 1 and T is greater than or equal to 3. If S_B is 1 and T is smaller than 3, then the connected component is decided to be confused. If none of the mentioned conditions are the case, then the connected component is isolated.

The flowchart for the algorithm determining the connected component type is given in Figure 2.5.

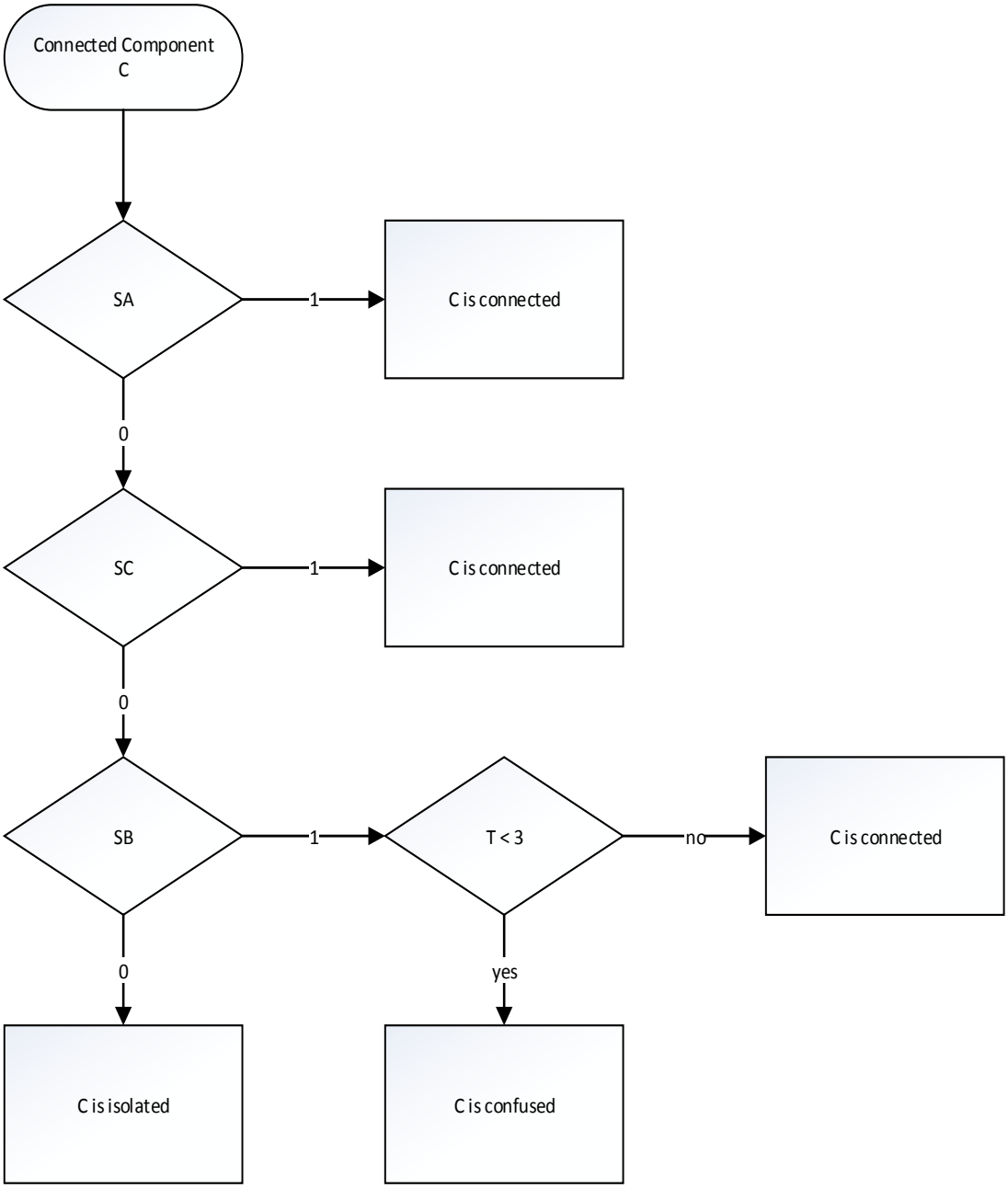


Figure 2.5: The classification algorithm deciding that a connected component is isolated, connected or confused.

Pal et al. [42] achieved 98.81% success rate using this method applied on 3800 components from French bank checks in determining whether connected components are connected or isolated digits.

After the classification of the component as isolated digit or touching digit string, touching numerals are segmented. In order to do so, the touching position of the digits is classified as top, middle or bottom. Then, the segmentation path is obtained based on closed loops, structural features and reservoir features.

In order to classify the touching position of the numerals, the bounding box of the connected component is divided into three regions both horizontally and vertically. Horizontal and vertical regions are obtained dividing the image into three regions in horizontal and vertical directions. The region are divided in 1:2:1 fashion. Figure 2.6 shows an example of the horizontal and vertical regions on a connected component.

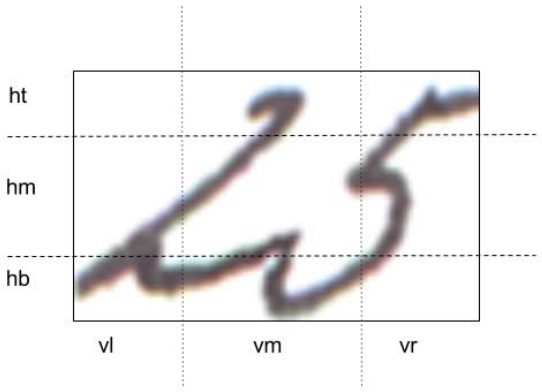


Figure 2.6: Horizontal and vertical regions of a connected component

From this point on, the largest reservoir whose centre of gravity lies in the v_m region is found. This reservoir is called the best reservoir [42]. The row corresponding to the bottom of the reservoir, named baseline, is found. The algorithm for the touching position is applied.

Let

$\delta = the\ number\ of\ boundary\ points\ of\ the\ best\ reservoir$

$S = \{f_R(x_i, y_i)\} = set\ of\ boundary\ points$

$I_P = ((x_1 + x_r)/2\,, (y_1 + y_r)/2$

For a top reservoir,

$x_1 = \max(x_i)$ and $y_1 = \min(y_i)$

$x_r = \max(x_i)$ and $y_r = \max(y_i)$

For the bottom reservoir,

$x_1 = \min(x_i)$ and $y_1 = \min(y_i)$

$x_r = \min(x_i)$ and $y_r = \max(y_i)$

Where $i = 1,2, \ldots, \delta$

From I_P the touching region is determined.

If $I_P \in h_t$, touching region = top region,

If $I_P \in h_m$, touching region = middle region,

If $I_P \in h_b$, touching region = bottom region,

$L = Euclidean\ distance\ of\ baseline\ (horizontally)$

$R = most\ frequent\ horizontal\ black\ run$

Pal et al. [42] proposed two segmentation algorithms for two cases. One of these cases is the connected components having two closed loops touching side by side and the other case is other touching components.

For the first case, the number of closed loops and their positions are taken into account. If two close loops are side by side and touching each other, curve segment is applied through the middle of their touching segment. A starting point is detected from the touching region. A segmentation path is produced moving up and also moving down from the starting point until it reaches to a reservoir or a boundary of a component. The joint paths form the segmentation path which is named as curve segmentation.

The starting point is found by drawing a line between centres of gravity of two close loops. The middle point of the points of this line lying in the touching region is assigned as starting point.

For the second case, the segmentation is done by tracing the boundary of the reservoir clockwise until an obstacle is reached. An obstacle is defined as the vertical black run whose length is greater than *3R/2*, where *R* is the most frequent black run. Then, the boundary is traced in the counter clockwise direction searching for another obstacle. The best obstacle should be chosen to determine the best segmentation path. If one of the obstacles is in the v_m region, then this obstacle is chosen. If both of them are in the v_m region, then the closest one to the largest close loop is chosen. If none of them is in the v_m region the segmentation path is chosen at the middle of the obstacles.

Pal et al. [42] named the proposed algorithm as NUM_SEGM algorithm which is defined as follows,

Step 1: Detect best reservoir and find the touching position based on this reservoir.

Step 2: Detect touching type. If the touching is loop–loop fashion, go to Step 5.

Step 3: Find initial points based on reservoir and find the best feature points based on confidence value (CV) calculated form the features extracted.

Step 4: Find the best obstacle point. For top and bottom touching straight line segmentation is done at the best node point. For the middle touching the best associated point of the best node point is detected. Segmentation is done by joining the best node point and its associate point. Stop.

Step 5: Detect start point. Segment according to the movement of starting point. Stop.

Pal et al. [42] tested the proposed algorithm with 2250 touching digit pairs extracted from the courtesy amount field of French bank cheques and they claimed 94.8% success rate which is verified manually.

Rui et al. [43] proposed a method merging drop-fall algorithms and the water reservoir concept explained earlier. The drop-fall algorithms creates a segmentation path by imitating an object falling or rolling in between two connected characters [44].

The drop-fall algorithms are proposed by Condego et al. [45]. The drop-fall algorithms are categorized in 4 types which are defined according to the starting point and the direction of the drop-fall which are ascending-left, ascending-right, descending-left and descending-right. Figure 2.7 illustrates 4 types of drop-fall algorithms.

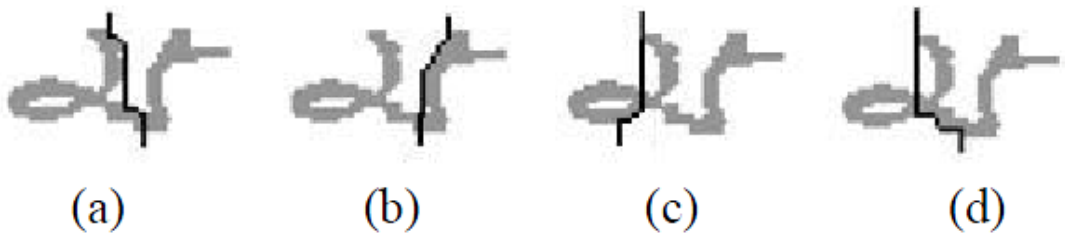


Figure 2.7: 4 Categories of drop-fall algorithms [43]

(a) Descend-left algorithm, (b) Descend-right algorithm, (c) Ascend-left algorithm, (d) Ascend-right algorithm

The drop-fall algorithms are based on a principle that a hypothetical marble falls in between two characters and applies the segmentation where the marble stops.

The important issue with the drop-fall algorithm is deciding where the marble start falling. Rui et al. [43] uses water reservoir concept in order to address this issue. They, first, divided touching numerals into categories. These categories are simple touching and multiple touching.

Simple touching is the case where two numerals touch at a single location. Simple touching could be divided into two sub-categories which are single-segment touching and single-point touching. Single-segment touching is the case where there the touch occurs in a long continuous stroke. Moreover, single-point touching happens where two digits connect at one point with the help of a ligature. This ligature could belong to a digit or not.

The single-point touching is also divided into 4 categories according to circumstances of ligatures. The ligature could belong to left, right, both or no digits.

The multiple-touching case is the case where the touching occurs in multiple locations.

The categories of the touching numerals based on their touching style are illustrated in Figure 2.8.

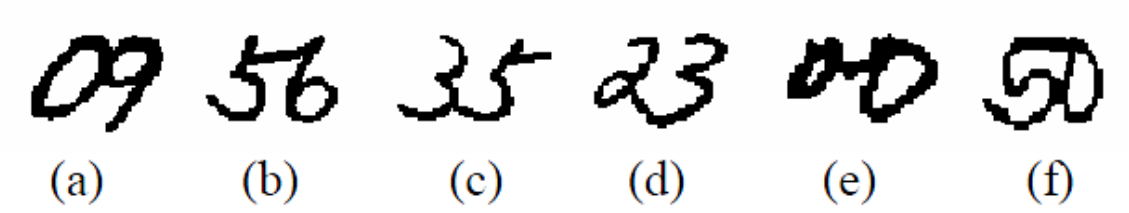


Figure 2.8: Different types of touching digits [43]

Single-segment touching (b)-(e) single-point touching (f) multiple touching

In [43], the structural features are examined in order to achieve accurate segmentation paths. They proposed a structural segmentation algorithms instead of recognition based algorithms in order to avoid recognition errors.

Rui et al. used the water reservoir concept defined in [42] deciding which drop-fall algorithm would be used to create the segmentation path. First, an algorithm is applied in order to classify the connected component whether it is isolated digit or connected digits. In the second case, touching style of the connected digits needs to be determined. The features used in the algorithm is as follows,

Let:

$NW_U = Number\ of\ top\ reservoirs$

$NW_D = Number\ of\ bottom\ reservoirs$

$NW = Number\ of\ reservoirs$

$GW_i = Centre\ of\ gravity\ of\ reservoir\ W_i$

$BW_i = Bounding\ box\ of\ reservoir\ W_i$

$NL = Number\ of\ close\ loops$

$GL_k = Centre\ of\ gravity\ of\ close\ loop\ L_k$

$BL_k = Bounding\ box\ of\ close\ loop\ L_k$

$f_{ho}(A, B) = horizontal\ overlap\ between\ rectange\ A\ and\ B$

$f_{vo}(A, B) = vertical\ overlap\ between\ rectange\ A\ and\ B$

$f_{hd}(A, B) = horizontal\ distance\ between\ rectange\ A\ and\ B$

$f_{vd}(A, B) = vertical\ distance\ between\ rectange\ A\ and\ B$

The pseudo code for the algorithm is given in Figure 2.9[43],

```
IF NL ≥ 2

        IF ( ∃ Lk,Ll) fvo (BLk,BLl) > 0
                IF ( ∃ Lk) GLk∈vm THEN s=3
                ELSE IF fhd (BLk,BLl ) >3·SW THEN s=2
                ELSE s=1
        ELSE IF ( ∃ Lk,Ll) fvd (BLk, BLl )>1.5·SW
                IF NW >0 OR fhd (BLk,BLl )>3·SW
                        THEN s=2
                ELSE s = 1
        ELSE IF (∀ Lk) width(BLk) < 2/3·CW
                THEN s=3
        ELSE s = 0
ELSE
        IF NW ≥ 3 THEN s=2
        ELSE IF ( ∃Wi ,Wj) fho (BWi , BWj ) > 0
                THEN s=2
        ELSE IF NWU = 2
                IF ( ∃Wi ,Wj) max(width(BWi),width(BWi)) < 2/3·CW
                        THEN s=2
                ELSE s = 0
        ELSE IF ( ∃Wi) height (BWi ) ≥3/4·CH ∧ GWi ∈vm
                THEN s=2
        ELSE IF NL=1
                IF GLk ∈hm
                        IF width(BLk)<2/3·CW
                                IF NW=0 THEN s=3
                                ELSE s=2
                        ELSE s = 1
                ELSE
                        IF NW > 0 THEN s=1
                        ELSE s=0
        ELSE s=0
Case s=0:               isolated
Case s=1:               single-segment touching
Case s=2:               single-point touching
Case s=3:               multiple-segment touching
```

Figure 2.9: Algorithm for touching style classification [43]

After the touching pattern is classified using this algorithm, the drop-fall algorithm is applied to the connected digits.

If the touching pattern is single-segment touching, then a descending-left algorithm is applied in the top reservoir or ascending-left algorithm is applied in the bottom reservoir.

If the touching pattern is classified as single-point touching, then the drop-fall algorithm is chosen according to the types of the single-point touching numerals described earlier. If the component has top reservoir and no bottom reservoir, then it is type-1, where the ligature belongs to the left digit. In this case descending-left algorithm is applied in the top reservoir.

If the component has bottom reservoir and no top reservoir, then it is type-2, where the ligature belongs to the right digit. In this case ascending-right algorithm is applied in the bottom reservoir.

If the component has both top and bottom reservoir and their bounding boxes overlap in both horizontal and vertical directions, then it is type-3, where the ligature belongs to the both digits. In this case there are two possibilities. According to the middle positions of the top and bottom reservoir baselines, if the middle point of the top reservoir's baselines lies at left side of the baseline of the bottom reservoir, then ascending-left algorithm is applied.

If the middle point of the top reservoir lies at the right side of the baseline of the bottom reservoir, then ascending-right algorithm is applied.

If the component has both top and bottom reservoirs and their bounding boxes overlap in the horizontal direction not in the vertical direction, then it is type-4, where the ligature belongs to no digit. In this case descending-left algorithm is applied in the top reservoir.

If the touching pattern is classified as multiple touching digits and only one close loop is obtained then descending-left or ascending-left algorithm could be applied from the

centre of the loop. If there are more than one loops, the loop whose centre is in v_m is chosen or the nearer to the centre of the touching component is chosen, then descending-left or ascending-left algorithm is applied.

Rui et al. [43] tested the proposed method with NJUST603HW dataset containing 623 binary images of digit strings, which are courtesy amount part of Chinese bank checks. The performance of the touching style classification algorithm was claimed to be 98.29% with 5374 sub-images. The proposed segmentation method was applied to 1557 sub-strings of touching numerals and 96.66% success rate was reached which is verified manually.

2.1.3. Handwritten Digit Recognition

After segmentation of the digit string is achieved successfully, separated digits must be classified in order to get the actual value of the digit string. Digit string recognition methods can be categorized into two sub-categories which are neural network based methods and other prominent methods. This thesis study is based on neural network methods.

Neural network is a group of interconnected artificial neurons which processes the data using its connections. The structure of the neural network is updated during the training process.

Leroux et al. [16] proposed combining a radial basis function and a time delay neural network (TDNN) for isolated digit classification. The radial basis function which is used in this study employed a concavity feature vector. In this context, the RBF is a neural network with three layers and the TDNN is a multilayer perceptron.

Liazaragga et al. [17] divided the set of features in two groups. The first group consists of holes and their relative locations, number of intersections with the principal and secondary axis and crossing sequences whereas the other group contains distribution of foreground pixels and the relative positions of intersecting strokes. The

classification procedure in this algorithm has two stages. First rule based classification is applied. Then Hopfield Neural Network (HNN) is used to finalize classification.

Hinton et al. [18] prosed a fast learning algorithm for deep belief nets. They showed that it is possible to train a deep, densely connected, belief network one layer at a time. After learning process is completed for each layer, weights are untied from the weights of the higher layers. They reported 98.75% recognition rate with MNIST database containing segmented and normalized handwritten digits. They argued that the only machine learning technique that come close to 1.25% error rate is a support vector machine (SVM) with a recognition rate of 98.6% [19]. It is also stated that this error rate could be improved by using weight sharing and sub-sampling.

2.1.4. Survey on Handwritten Digit String Segmentation Algorithms on CVL Strings Dataset

This section discusses different handwritten digit string recognition methods on CVL Strings dataset and their results. Moreover, this section includes the results of HDSRC competition of ICDAR on handwritten digit string recognition [29].

A-1. In the method proposed by Wu et al. [30], firstly, the input image is classified into two subclasses as simple or complex image using a neural network. Then, a simple image is thresholded using Otsu' method. While a complex image is thresholded using Sauvola Method after two Multilayer Perceptron neural networks are used to clean and enhance the image. After thresholding is done, connected components are extracted and slant-correction is done on the connected components. The connected components that are heavily overlapped in the horizontal direction are merged together. After that, according to the estimated string height, the connected components that have large width are considered as potential touching multiple numerals. These potential touching numerals are split using upper/lower profile curve analysis generating vertical cuts [31]. As a result, the digit string is split into primitive image segments. These segments are combined together in order to generate candidate digit

patterns and for a segmentation candidate lattice. All of the paths from the start node towards the end node represent a candidate segmentation. A polynomial classifier with gradient direction histogram feature is used for classification of the candidate patterns. The digit classifier and binary class-dependent geometric model are combined together using confidence transformation in order to evaluate the segmentation path [32]. This method has achieved 85.3% recognition rate on CVL Strings Database.

A-2. [33] Leite and Zanchettin designed a system combining multiple classifiers. First a preprocessing algorithm is applied in order to eliminate non-digit structures [34]. For the thresholding an MLP neural network is used to choose the best threshold for a set of pixels in the image [35]. In this method, a combination of two hybrid classifiers is used. One of them is a model which is a combination of k-NN and SVM that increases accuracy in highly similar situations. The other classifier is a combination of Multidimensional Recurrent Neural Networks (MDRNN) [36] and SVM [37]. Using this classifier, an improvement on the accuracy is achieved on highly connected digits without segmenting them. This algorithm achieved 58.6% recognition rate on CVL-Strings dataset.

A-3. Lu and Lu [38] proposed a method that applies the segmentation based on adaptive binarization and connected component analysis (CCA). Then character classifiers, such as SVM, self-organizing map (SOM) and backpropagation, are used to recognize segmented digits. The recognition rate of this method is 48.9% on CVL Strings dataset.

A-4. Gattal et al. [39] applied a forward segmentation strategy based on oriented sliding window [40] and [41]. This method uses contour and skeleton points as structural features. Radon transform is applied on the connected digits in order to get the orientation of the sliding window. In order to extract the digits that are not overlapped, the histogram of vertical projection is applied. If the extracted digit could not be classified, then it is considered that the image

contains multiple digits. In this case, oriented sliding window algorithm is applied. The algorithm is summarized as

a. Base points (BP) and Interconnection points (IP) are generated. BPs are extracted from the local extrema detected on the contour whereas IPs are calculated using Freeman code.
b. The sliding window is set to IPs positions.
c. Oriented window is crossed around IPs with an angle in order to find the cutting path.
d. All possible relationships of BPs and IPs are scanned. The possible relations are [41]
    i. If the Euclidean Distance between projection BP and IP is lower than a threshold, then the cut is between IP and upper BP and between IP and lower BP.
    ii. If the lower segment of IP is related to upper segment of IP and they are both close to a BP, the skeleton path linking IPs is used as a part of a segmentation cut.
    iii. If there is no IP on the skeleton path, although there is connection of digits, the segmentation path is placed based on the minimal Euclidean distance between upper and lower BP.
e. The correct segmentation path is evaluated using SVM.

This algorithm achieved 59.3% recognition rate on CVL Strings dataset.

2.1.5. Evaluation Metrics

Two evaluation metrics are used in ICDAR competitions. First of them is the hard metric, it basically measures the recognition rate which is the number of correctly recognized digit strings divided by total test samples.

Another metric is the soft metric. It is defined in order to find how close the misclassified digit string is to the actual value. This metric is Normalized Levenshtein Distance (NLD) [29]. It is the normalized version of the Levenshtein Distance (LD).

It is used in order to eliminate the effect of the string length to the performance. NLD is defined by the following equation,

$$NLD(a_T, a_R) = \frac{LD(a_T, a_R)}{|a_T|}$$

Where $|a_T|$ is the length of the string a_T, $|a_R|$ is the length of the string a_R and *LD* is the Levenshtein distance. Levenshtein distance is a string metric measuring the difference between two strings. It is defined as minimum number of single character edits required to convert one string to another.

After NLD is calculated, average NLD is calculated in order to reliably compare different methods.

$$ANLD = \frac{\sum_{i=1}^{T} NLD(a_T^i, a_R^i)}{T}$$

Where T is the number of test samples. Since ANLD is an inverse performance metric, low values indicate better performance whereas high values are worse.

2.1.6. Comparison of Algorithms

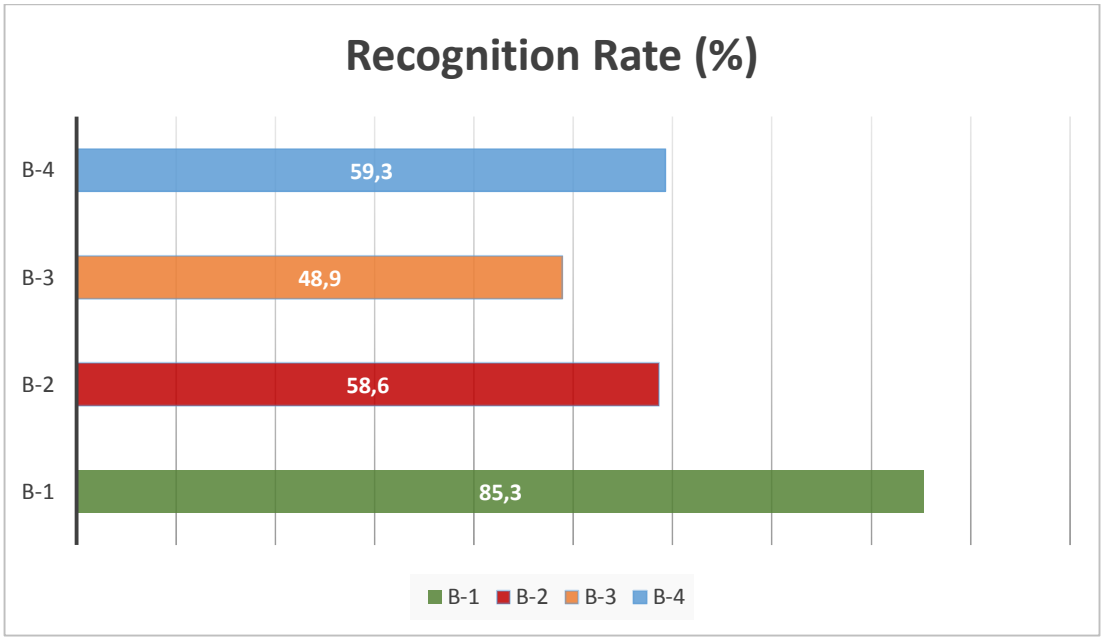


Figure 2.10: The Recognition Rate of Described Algorithms on CVL Database

B1: Wu et al. [30], B2: Leite & Zanchettin [33],

B3: Lu & Lu [38], B4: Gattal et al. [39]

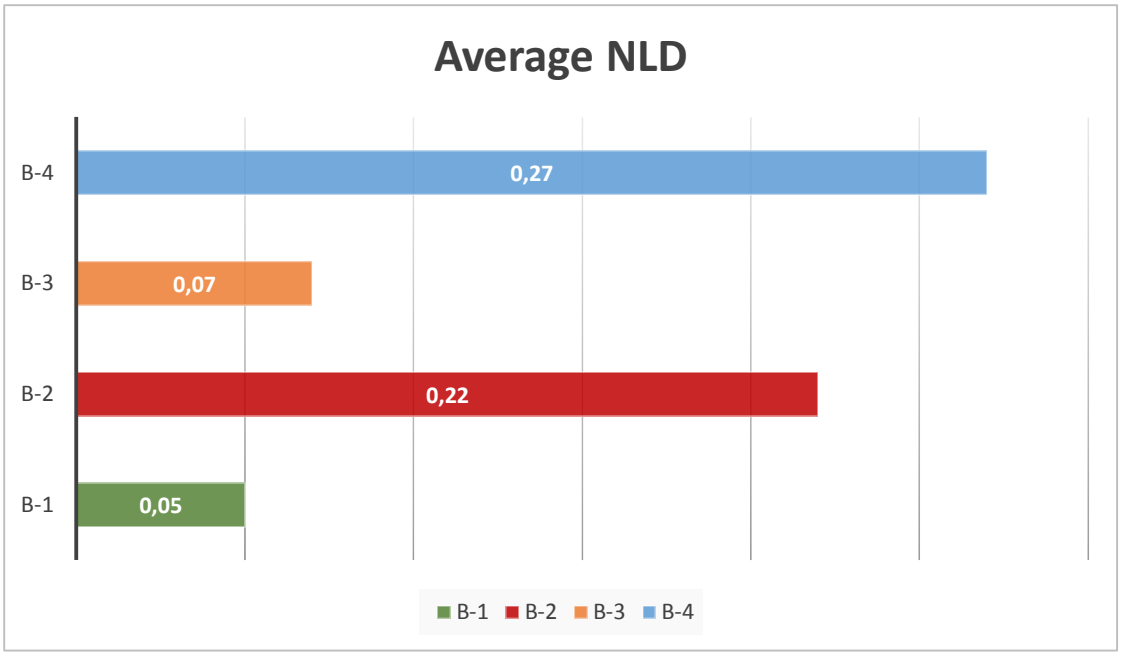


Figure 2.11: The Average NLD of Described Algorithms on CVL Database

B1: Wu et al. [30], B2: Leite & Zanchettin [33],

B3: Lu & Lu [38], B4: Gattal et al. [39]

2.2. Datasets

In this study, two datasets are used. MNIST database is used for the training set of the handwritten digit recognition algorithms. CVL-Strings databases is used for the segmentation algorithms.

2.2.1. MNIST Dataset

This database contains 60 thousand training samples and 10 thousand test samples of handwritten digits from 0 to 9. Each image on this database is 28x28 pixels. The digits in this database are size-normalized and centred [20].

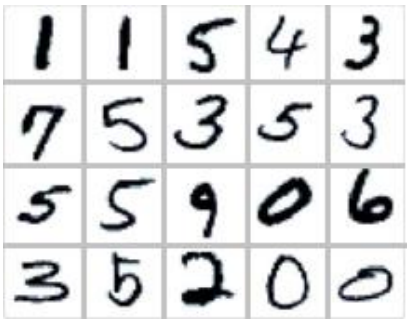
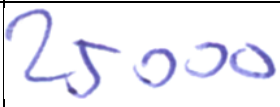
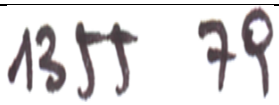
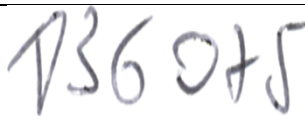
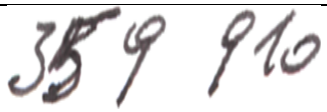
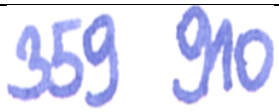
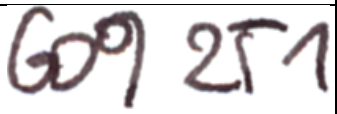
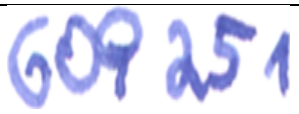
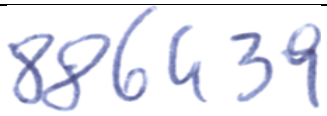
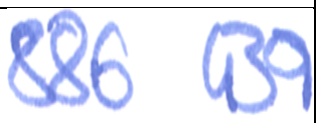
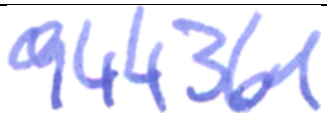
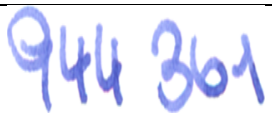
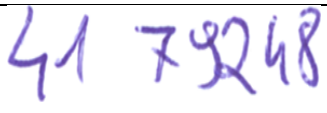
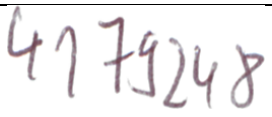
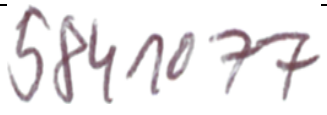
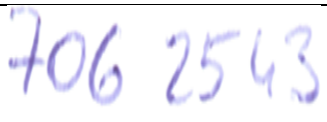
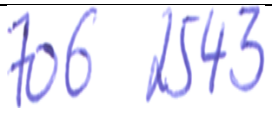


Figure 2.12: Examples of digits in MNIST database.

2.2.2. CVL-Strings Dataset

This database contains 10 different digit strings from about 120 writers resulting in 1262 training images and 6698 test samples [21]. The lengths of the strings change from 5 to 7 averaging 6.1 digit length.

Table 2.2: Some examples of 10 different strings from CVL-strings database

Strings	**Value**	**Sample 1**	**Sample 2**
String 1	2500		
String 2	135579		
String 3	136075		
String 4	359910		
String 5	609251		
String 6	886439		
String 7	944361		
String 8	4179248		
String 9	5841077		
String 10	7062543		

2.3. Background Information

In this part, background information about the methods which are implemented in this thesis will be given.

2.3.1. Otsu Thresholding

In Otsu thresholding [3], the goal is to find a threshold value to cluster data into two groups minimizing within cluster variances, thus maximizing between cluster variances. It can be used to convert grayscale images to binary images.

The weighted sum of within cluster variance is defined as:

$$\sigma_w^2(t) = w_1(t)\sigma_1^2 + w_2(t)\sigma_2^2 \tag{2.1}$$

Where, ω_i is the probability of occurrence for class *i* that is separated by threshold t.

$$\omega_1(t) = \sum_{i=1}^{t} P(i), \quad \omega_2(t) = \sum_{i=t+1}^{I} P(i) \tag{2.2}$$

Therefore, mean of each cluster is calculated by

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{\omega_1(t)}, \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{\omega_2(t)} \tag{2.3}$$

Moreover, individual class variances are defined as

$$\sigma_1(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{\omega_1(t)} \tag{2.4}$$

$$\sigma_2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P(i)}{\omega_2(t)} \tag{2.5}$$

When all possible threshold, t, values are attempted, the threshold value that minimizes $\sigma_w^2(t)$ could be found.

In order to decrease the computational cost, a faster method could be generated using recursive operations. This calculation exploits the definition that for any threshold, the total variance is the weighted sum of within cluster variances and the between cluster variance (sum of weighted squared distances between the class means and the overall mean). The total variance can be written as,

$$\sigma^2 = \sigma_w^2(2) + \omega_1(t)[1 - \omega_1(t)][\mu_1(t) - \mu_2(t)]^2 \qquad (2.6)$$

Since the total variance is constant, maximizing the between cluster variance could be used instead of minimizing within cluster variances. This yields to the recursive computation of between cluster variance.

The algorithm is as follows,

- Compute the normalized histogram of the input image
- Compute the global mean
- For each possible threshold value compute the between cluster variance
- Choose the threshold value that maximizes the between cluster variance.

2.3.2. Deep Learning

Deep learning is a set of algorithms in machine learning that attempt to learn multiple levels of representation, corresponding to different layers of abstraction. It typically uses artificial neural networks. Main interest of this thesis on this concept is Deep Belief Networks.

Deep belief networks (DBN) is a probabilistic model with multiple layers of stochastic of variables. Each connected pair of DBN forms directed sigmoid belief network except for the top two layers that form an undirected graph. Each layer of the DBN contains high order relations of the features in the preceding layer.

Hinton et al. [18] introduced an unsupervised learning algorithm for this structure utilizing greedy layer wise training to learn a deep network.

DBNs are composed of energy based models which are mostly Restricted Boltzmann Machines (RBM).

2.3.2.1. Restricted Boltzmann Machines

Restricted Boltzmann Machine (RBM) is a Boltzmann Machine with restricted connectivity. In RBM, there are two layers called as visible (v) and hidden (h) layers. While the units in one layer are fully connected to the units in the other layer, the units in the same layer are not connected to each other. The connection in the RBM between visible and hidden layers are symmetric.

$$W_{ij} = W_{ji} \tag{2.7}$$

Where,

$i \in visible\ layer,$

$j \in hidden\ layer$

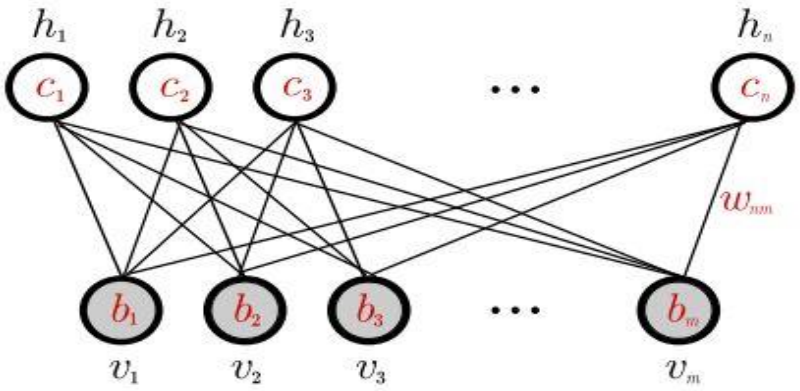


Figure 2.13: RBM Structure

Supposing the RBM consists of visible units, $v \in \{0,1\}^D$, and hidden units, $h \in \{0,1\}^F$. The energy function is defined as:

$$E(v,h;\theta) = -v^T W h - b^T v - a^T h \quad (2.8)$$

$$E(v,h;\theta) = -\sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij}\, h_i v_j - \sum_{j=1}^{D} b_j\, v_j - \sum_{i=1}^{F} a_i h_i \quad (2.9)$$

Where $\theta = \{W, b, a\}$ are the network parameters.

$\boldsymbol{W_{ij}}$ represents weights between each visible and hidden unit pair,

$\boldsymbol{b_j}$ and $\boldsymbol{a_i}$ represent bias terms.

The joint probabilistic distribution $P(v,h;\theta)$ of the RBM states (v, h) depends on the energy of the state.

$$P(v,h;\theta) = \frac{1}{Z(\theta)} e^{-E(v,h,\theta)} \quad (2.10)$$

Where,

$$Z(\theta) = \sum_v \sum_h e^{-E(v,h,\theta)} \quad (2.11)$$

$Z(\theta)$ is the normalizing constant. Then the probability of the vector of the visible layer is

$$P(v;\theta) = \frac{1}{Z(\theta)} \sum_h e^{-E(v,h,\theta)} \quad (2.12)$$

$$P(v;\theta) = \frac{1}{Z(\theta)} \sum_h e^{v^T W h + b^T v + a^T h} \quad (2.13)$$

$$P(v;\theta) = \frac{1}{Z(\theta)} e^{b^T v} \prod_{j=1}^{F} \sum_{h_j \in \{0,1\}} e^{a_j h_j + \sum_{i=1}^{D} W_{ij} v_i h_j} \quad (2.14)$$

$$P(v;\theta) = \frac{1}{Z(\theta)} e^{b^T v} \prod_{j=1}^{F} (1 + e^{a_j + \sum_{i=1}^{D} W_{ij} v_i}) \quad (2.15)$$

Thus, the effect of hidden units is marginalized out.

The RBM can be interpreted as stochastic neural network. The conditional probability of a single variable being 1 can be explained as the firing rate of a neuron with sigmoid activation function.

$$\sigma(x) = {}^{1}/_{1 + e^{-x}} \quad (2.16)$$

The conditional probabilities over **h** and **v** can be derived using the joint distribution.

$$P(h|v;\theta) = \prod_j p(v_i|h) \quad (2.17)$$

$$P(v|h;\theta) = \prod_i p(h_j|v) \quad (2.18)$$

$$p(h_j = 1|v) = \sigma\left(\sum_i W_{ij} v_i + a_j\right) \quad (2.19)$$

$$p(v_i = 1|h) = \sigma\left(\sum_j W_{ij} h_j + b_i\right) \quad (2.20)$$

The derivative of the probability function of visible layer with respect to θ is

$$\frac{\partial \log(p;\theta)}{\partial W} = E_{P_{data}}[vh^T] - E_{P_{Model}}[vh^T] \quad (2.21)$$

$$\frac{\partial \log(p;\theta)}{\partial a} = E_{P_{data}}[h] - E_{P_{Model}}[h] \quad (2.22)$$

$$\frac{\partial \log(p;\theta)}{\partial b} = E_{P_{data}}[v] - E_{P_{Model}}[v] \quad (2.23)$$

where $E_{P_{data}}$ is the expectation of the data distribution $P_{data}(h, v; \theta)$ and $E_{P_{Model}}$ is the expectation of the distribution of the model. In training RBM, θ is updated so that the expectation of the model distribution $E_{P_{Model}}$ gets closer to the expectation of the data distribution $E_{P_{data}}$. In other words, θ is updated in the direction of gradient of the objective function. However, the computational cost of the calculation of the $E_{P_{Model}}$ is quite large as it increases exponentially with respect to total number units in the RBM. Therefore, Hinton [24] introduced "Contrastive Divergence" function which is an approximation to the gradient of objective function [44]

$$\Delta W = \alpha(E_{P_{data}}[vh^T] - E_{P_T}[vh^T]) \quad (2.24)$$

where α is the learning rate and P_T is a distribution defined by running Gibbs chain for T steps initializing at the data. This contrastive divergence function converges to maximum likelihood when T goes to infinity.

When an input is given to the visible units of the RBM, a sample for the hidden layer according to the probability distribution is chosen. Then reconstruction is done using the chosen sample from the hidden unit, and using the contrastive divergence method the weight matrix is updated.

2.3.2.2. Training Deep Belief Networks

Deep belief networks (DBN) are composed of one visible layer which is the input layer and multiple hidden layers. DBNs are formed by stacking multiple layers of RBMs. The top two layers of the DBN form an RBM and lower layers form a directed sigmoid belief network.

A greedy learning algorithm is applied to the deep belief networks. Considering a DBN with two hidden layers (h^1 and h^2) of the same size and a visible layer (v). The joint distribution of this model is given as

$$P(v, h^1, h^2; \theta) = P(v|h^1; W^1)P(h^1, h^2; W^2) \quad (2.25)$$

Where

$$\theta = \{W^1, W^2\} \quad (2.26)$$

$$P(v|h^1; W^1) = \prod_i p(v_i|h^1; W^1) \quad (2.27)$$

$$p(v_i = 1|h^1; W^1) = g\left(\sum_j W_{ij}^1 h_j^1\right) \quad (2.28)$$

$$P(h^1, h^2; W^2) = \frac{1}{Z(W^2)} e^{{h^1}^T W^2 h^2} \quad (2.29)$$

Since $W^2 = {W^1}^T$, DBN's joint distribution is equal to RBM's joint distribution.

$$P(v, h^1; \theta) = P(v|h^1; W^1) x \sum_{h^2} P(h^1, h^2; W^2)$$

$$P(v, h^1; \theta) = \frac{1}{Z(W^1)} e^{W_{ij}^1 v_i h_j^1} \quad (2.30)$$

The greedy algorithm explained here takes the DBN as a stack of RBMs. First, the bottom RBM is trained and W^1 is computed. Then W^2 is initialized as $W^2 = {W^1}^T$. Let $Q(h^1|v)$ be any approximating distribution, then the log-likelihood of the DBN model has lower bound.

$$\log P(v; \theta) \geq \sum_{h^1} Q(h^1|v)[\log P(h^1; W^2) + logP(v|h^1; W^1)] + H(Q(h^1|v))$$

Where H is the entropy function. Then $Q(h^1|v)$ is set to $P(v|h^1; W^1)$ defined by the bottom RBM. Then W^1 is frozen and better model for $P(h^1; W^2)$ is searched maximizing lower bound defined previously.

$$\sum_{h^1} Q(h^1|v) log P(h^1; W^2) \qquad (2.31)$$

This idea could be improved to training DBN with more hidden layers. The joint distribution and approximate posterior of the DBN with L layers are given by

$$P(v, h^1, \dots, h^L) = P(v|h^2) \dots P(h^{L-2}|h^{L-1}) P(h^{L-1}|h^L)$$

$$Q(h^1, \dots, h^L|v) = Q(h^1|v) Q(h^2|h^1) \dots Q(h^L|h^{L-1}) \qquad (2.32)$$

In order to produce approximate sample from the DBN, a Gibbs Sampler could be done generating h^{L-1} from $P(h^{L-1}, h^L)$ [25]. Then a top-down approach is applied through the DBN activating each layer. In order to produce an exact sample from *Q*, approximate posterior distribution, a bottom-up approach is applied activating each layer.

Increasing number of layers, adding a layer of features, improves variational lower bound on the log probability of the training data. Each RBM converts its data distribution into an aggregated posterior distribution over its hidden layer. Firstly, generative weights are learned so as to convert aggregated posterior distribution over hidden layer into the data distribution. Then, aggregated posterior distribution is modelled.

Training is done using an unsupervised greedy manner. More precisely, each layer is treated separately and successively trained in a greedy manner. Then a supervised fine-tuning is applied to the whole network. The algorithm is as follows:

- Construct an RBM with an input and a hidden layer.
- Train the constructed RBM
- Stack another hidden layer on top of the constructed RBM and form a new RBM.
- Fix the weights of the first RBM, take a sample from the hidden layer of the first RBM using the probability distribution and use this sample as an input to the new RBM.
- Continue to stack layers on top of the network and train in the same manner.

- Finally, do the supervised fine-tuning to the constructed neural network.

Different fine-tuning algorithms are proposed for deep belief nets. In this study fine-tuning with back propagation is used. For this purpose, the resulting DBN is considered as multilayer neural network and conventional back propagation algorithm is applied with the calculated connection weights in the unsupervised training. This is done in a supervised manner using labelled data. The advantage of using DBN and applying back propagation afterwards is a good and fast performance could be achieved with a very few labelled data.

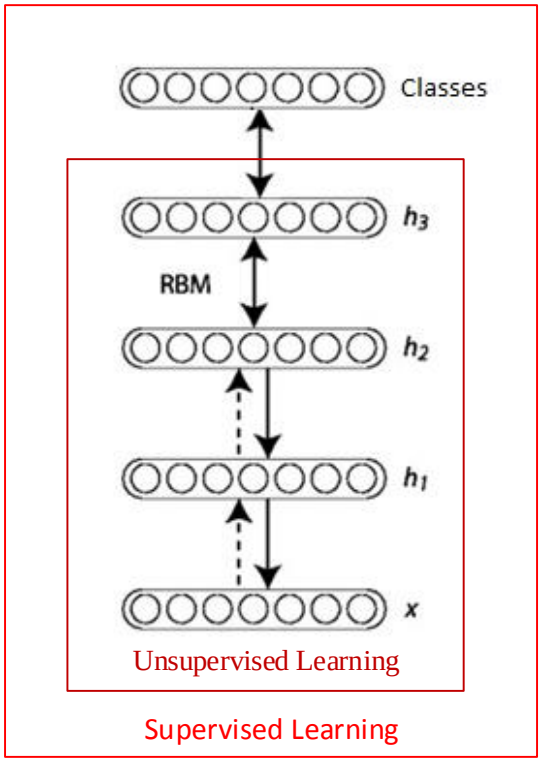


Figure 2.14: DBN Structure

2.3.3. Support Vector Machines

Support Vector Machines (SVM) performs classification between two classes by finding a decision surface that is based on the most informative points on the training set. The aim of this approach is to find a separating hyperplane that classifies two classes of data with the largest margin. The shortest distance between samples of two classes that are closest to the separating hyperplane is called the margin.

The theory behind support vector machines claims that the distance of all samples, x_i, to the decision boundary should be as large as possible. In this way, the decision boundary becomes more robust. Support vector machines maximizes the distance between the decision boundary and the closest training samples.

These constraints could be defined for the separating hyperplane of two linearly separable classes.

$$w.x_i + b \geq +1, \qquad y_i = +1 \tag{2.33}$$

$$w.x_i + b \leq -1, \qquad y_i = -1 \tag{2.34}$$

The constraints defined in equations 2.33 and 2.34 can be combined as follows,

$$y_i(w.x_i + b) \geq +1, \forall i \tag{2.35}$$

Where $\boldsymbol{y_i}$ is the label of each sample, $\boldsymbol{x_i}$ is the feature vector of each sample and $\boldsymbol{w}$ is the normal to the separating hyperplane. From these equations, $\frac{\boldsymbol{b}}{||\boldsymbol{w}||}$ is the distance from the hyperplane to the origin and $||\boldsymbol{w}||$ is the Euclidean norm of the surface.

The margin, that is the distance between the hyperplanes having the same normal passing through the closest samples from each class, is $\frac{\boldsymbol{2}}{||\boldsymbol{w}||}$.

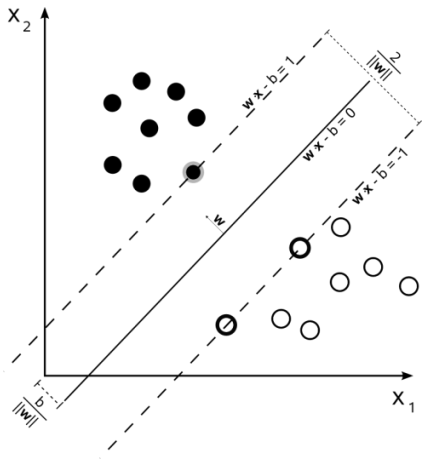


Figure 2.15: SVM with two classes

In order to find the separating hyperplane with maximum margin, we need to solve the optimization problem maximizing the calculated distance $\frac{\mathbf{2}}{||\boldsymbol{w}||}$, thus minimizing

$\frac{\mathbf{1}}{\mathbf{2}}||\boldsymbol{w}||^{\mathbf{2}}$ under constraints $\boldsymbol{y_i}(\boldsymbol{w}.\boldsymbol{x_i} + \boldsymbol{b}) \geq +\mathbf{1}, \forall \boldsymbol{i}$.

Using the method of Langrange Multipliers, this optimization problem is expressed as,

$$\min L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i w + b) + \sum_{i=1}^{l} \alpha_i \qquad (2.36)$$

Where $\boldsymbol{\alpha_i}$'s are the Lagrange multipliers. For minimizing this equation with respect to **w** and **b**, the Karush-Kuhn-Tucker conditions are used. The KKT conditions state that **w**, **b** and $\boldsymbol{\alpha}$ should satisfy the following conditions.

$$\frac{\partial L(w,b,a)}{\partial w} = w_v - \sum_i \alpha_i y_i x_{iv} = 0 \ , v = 1, \dots , d \qquad (2.37)$$

$$\frac{\partial L(w,b,a)}{\partial b} = -\sum_i \alpha_i y_i = 0 \qquad (2.38)$$

$$y_i(x_i.w + b) - 1 \geq 0, \forall i \quad (2.39)$$

$$\alpha_i \geq 0, \forall i \quad (2.40)$$

$$\alpha_i(y_i(x_i.w + b) - 1) = 0, \forall i \quad (2.41)$$

The first KKT condition defines the optimal hyperplane that separates individual classes

$$w^* = \sum_i \alpha_i^* y_i x_i$$

whereas the second condition defines a requirement for $\boldsymbol{\alpha_i}$ coefficients.

$$\sum_{i=1}^{n} \alpha_i^* y_i = 0, \qquad \alpha_i^* \geq 0$$

At this point, the derivatives of **L** with respect to **w** and **b** must vanish the following KKT conditions. By substituting equations 2.37 and 2.38 into equation 2.16, the following formulation is obtained.

$$\max_{\alpha} L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i.x_j \qquad subject\ to \begin{cases} \sum_i \alpha_i y_i = 0 \\ \alpha_i \geq 0 \end{cases}$$

Therefore solving this optimization problem, α_i coefficients are obtained. The samples that yield $\alpha_i > 0$ are called the support vectors. The support vectors lie on the hyperplanes that defines the closest hyperplane consisting samples to the separating hyperplane. In order to find the solution for **w**, the normal to the optimal separating hyperplane, only the support vectors are needed. Therefore, the decision function for SVM becomes,

$$f(x) = w^T x_i + b = \sum_{i=1}^{M} y_i \alpha_i (x_i^T x) + b$$

$sgn(f(x))$ determines the class of a new sample, **x**.

Moreover, positive slack variables, ξ_i, are introduced in order to relax the constraints defined previously. This is needed when the training samples cannot be seperated by a hyperplane. Even if the data is linearly seperable, definition pf positive slack variables prevents overfitting of the separating hyperplane. Basically, by defining ξ_i, we permit a number of instances to be inside the margin or in the area of other class. The number of these instances depends on ξ_i value. As ξ_i increases, also the number of instances being in the other side of the marginal hyperplane also increases.

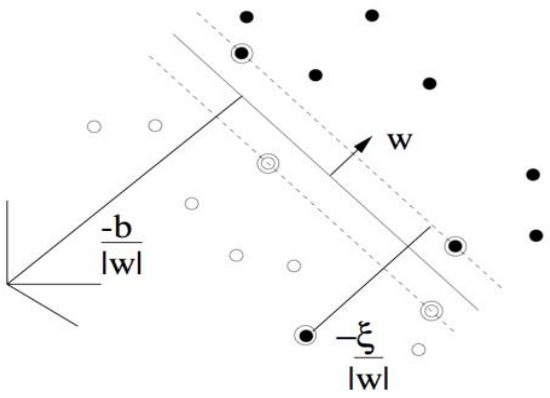


Figure 2.16: Definition of slack variables

In addition to that, using 'kernel trick', mapping the samples into higher dimensional feature space, SVM can perform nonlinear classification. The training data is mapped to another Euclidean space by Φ. The only difference that this mapping would yield is

that instead of computing dot products $\boldsymbol{x_i}.\boldsymbol{x_j}$, the dot products $\boldsymbol{\Phi}(\boldsymbol{x_i}).\boldsymbol{\Phi}(\boldsymbol{x_j})$ are taken into account. Mapping all the samples by $\boldsymbol{\Phi}$ would be computationally expensive. However, using kernel function corresponding to the dot product, $K(x_i, x_j) = \Phi(x_i).\Phi(x_j)$, decreases the computational cost. Therefore, the optimization problem can be expressed as,

$$\max_{\alpha} L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \;\; subject\ to \begin{cases} \sum_i \alpha_i y_i = 0 \\ \alpha_i \geq 0 \end{cases} \quad (2.42)$$

The kernel functions used in the experiments are listed below.

- RBF kernel: $K(x_i, x_j) = \exp(\frac{-||x_i - x_j||^2}{2\sigma^2})$
- Polynomial kernel: $K(x_i, x_j) = (\langle x, y \rangle)^d$
- Linear kernel: $K(x_i, x_j) = \langle x, y \rangle$
- Sigmoid Kernel: $K(x_i, x_j) = \tanh(\gamma\langle x, y \rangle - \theta)$

  Where < a, b> denotes the inner product of a and b.

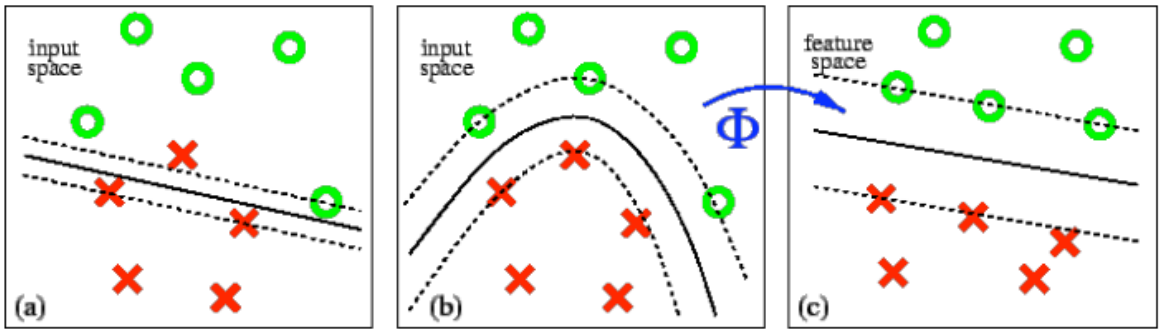


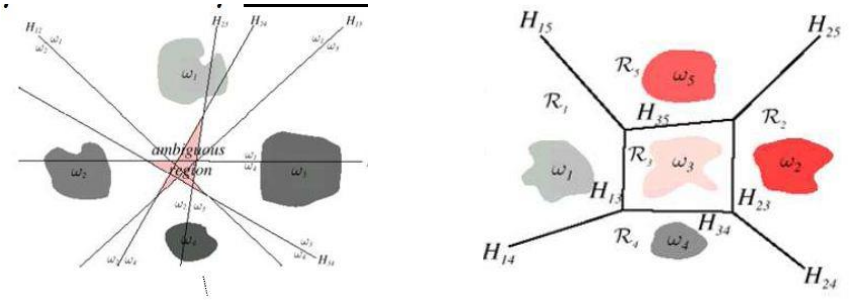
Figure 2.17: The Kernel Trick (a): data is not linearly separable, (b): nonlinear surface is defined for separation, (c): linear surface for mapped samples.

Although, SVM is inherently a two-class classifier, in order to classify datasets having multiple classes, a multiclass SVM method is needed. The most common thing to do so is building a one-versus-rest classifier. It is done by generating N different classifiers for each class.

Another method is to build N(N-1) one-versus-one classifiers to separate each pair of classes, but there is a possibility of an ambiguous region in this method.

There are great number of techniques solving multiclass classification problem, but these are the simplest ones and they work good enough with properly tuned classifiers. The choice between two options mainly depends on the computational complexity.

One-versus-rest method requires O(N) classifiers whereas one-versus-one method requires $O(N^2)$ classifier. Although, one-versus-one method require more classifiers, the size of the subset of dataset containing related exemplars related to each classifier is a lot smaller. Thus, it becomes faster and more memory efficient.



(a) (b)

Figure 2.18: Multiclass SVM algorithms [46]

(a: one-versus-one classifier, b: one-versus-rest classifier)

CHAPTER 3

PROPOSED APPROACH AND EXPERIMENTAL RESULTS

The block diagram of the approach followed in order to solve the handwritten digit recognition problem is given in Figure 3.1.

First a preprocessing part is applied. This part contains all the algorithms applied from the digitization of the image to segmentation. After the digitization, grayscale image is calculated. Then thresholding is applied using Otsu's method to obtain binary images and it is resized keeping the aspect ratio fixed so that all images have the same height. Finally, morphological operators are used to remove the noise in the image.

After the preprocessing is completed segmentation part is applied. Here, two algorithms are applied for comparison. First, the segmentation algorithm that merges water reservoir concept with the drop-fall algorithms [43] is applied as a baseline for comparison purpose. Then the segmentation using deep belief nets which is the segmentation approach proposed in this study is applied.

Before the digit recognition module is used, another preprocessing module is needed in order to get compatible images for the classifier. This module mainly contains centre of mass extraction and normalization.

Finally the digit classifier module is used to recognize segmented digits. In this part, support vector machines and deep belief nets are used and compared as they are considered as competing algorithms with high success rate by Hinton et al. [18].

In this chapter, the modules mentioned earlier will be described step by step until the conclusion of the experiments.

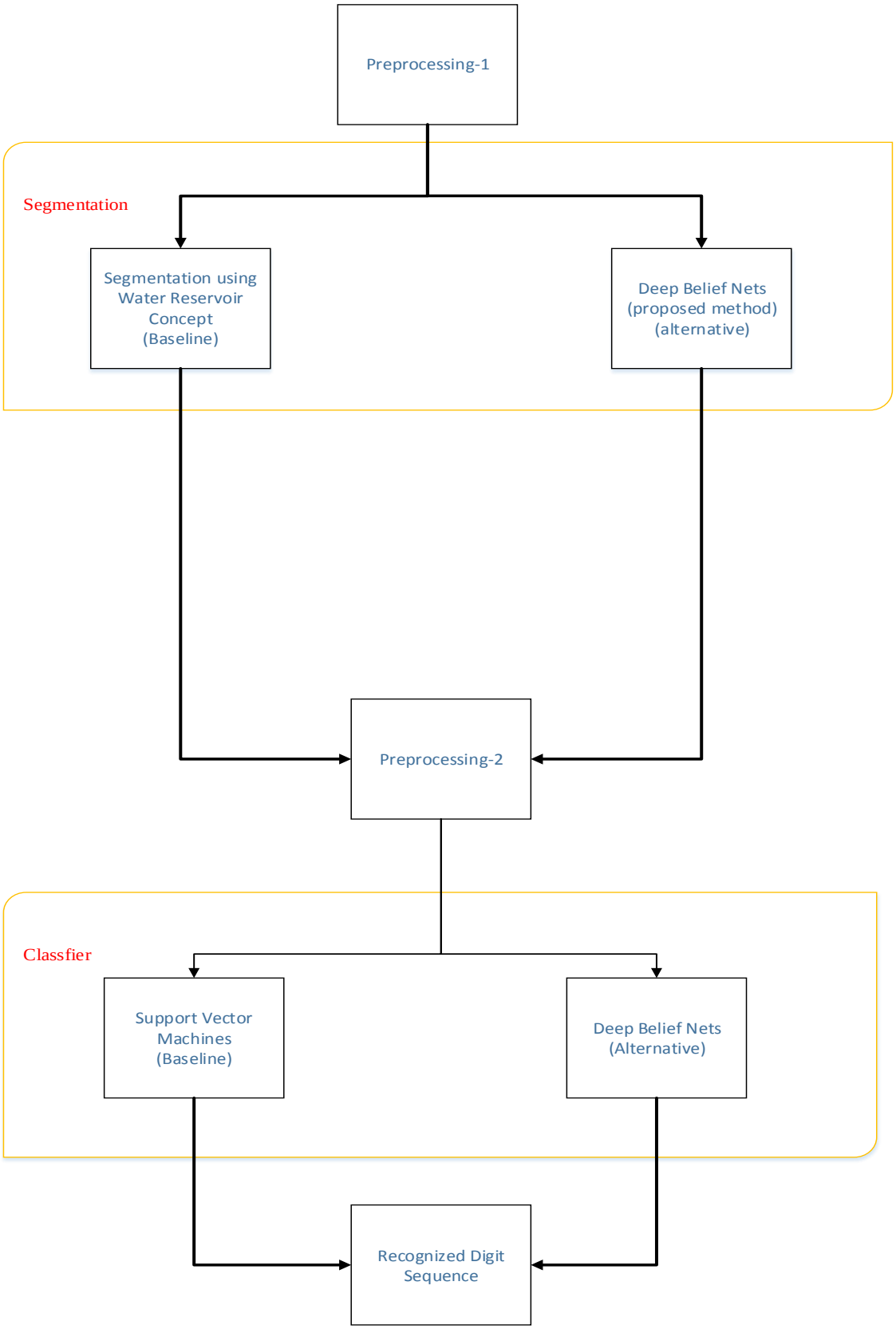


Figure 3.1: The block diagram of the proposed approach

3.1. Custom Dataset for Digit String Segmentation

For the algorithm employed in this thesis, that is segmentation using deep belief networks, and also for the validation of the segmentation algorithms, a dataset of digit and non-digit members are needed. Therefore, an extraction of sample images with these labels from the strings is needed. CVL-Strings library is used for establishing this custom dataset.

For this reason a MATLAB script is written in order to make this process less time consuming. This module takes the input image and applies all of the preprocessing algorithms that will be applied in the main segmentation and recognition module. Then a window which has 10 pixels horizontally slides through the image. Therefore a window with h=100 pixels and w=10 pixels is used, since all the digit string images are normalized to height h=100. While sliding, it asks the user the label of the image that is spanned by the window. Finally it saves the data to a .mat file. There are 36442 samples containing digit and non-digit images in this dataset.

3.2. Custom Dataset for Digit Recognition

For the digit recognition algorithms, MNIST database is used for training purposes. However, since the tests are done with CVL Strings database, the recognition algorithms are decided to be tested with also classifiers trained with the individual digits from the CVL Strings database. This could constitute a true comparison between the benchmark algorithms with our results.

Since the main database is a digit string library, correctly segmented digits are needed for creating this database. Therefore, the results from the segmentation module is passed to the digit recognition module trained with the MNIST database and classified the individual digits. The correctly classified digits are, then, saved to a .mat file forming a new custom set of digits.

For digit recognition module, DBN is used, since it has the highest recognition rate. As a result, a dataset is created containing 7344 digits.

The digits in the resulting dataset are saved in the same format as MNIST in order to avoid any inconsistency.

3.3. Preprocessing

When the coloured input image comes to the handwritten digit string recognizer, first the image is converted to a grayscale image.

Then the image is resized keeping the aspect ratio. Because, input images varies in vertical length and all the images must be in the same height for the segmentation and classification modules to work correctly, since the segmentation module creates a sequence of images by windows of the same size. Equalizing only the heights to same size would be enough. Therefore, height is fixed as h=100 and horizontal length is adjusted according to the aspect ratio of the image.

From this grayscale image thresholding is applied based on Otsu Thresholding [3].

Finally, morphological operations are applied to remove noise.

3.4. Segmentation

In this step two alternative segmentation algorithms are applied and compared. One is the segmentation combining water reservoir concept and drop-fall algorithms [43] explained in chapter 2 and the other one is the segmentation using deep belief network which is a novel approach proposed in this study.

3.4.1. Segmentation Using Water Reservoir Concept

The algorithm described in chapter 2 is employed in this chapter. Since the algorithm described before is for segmenting connected pairs of the digits, first connected component analysis is applied to the input digit string and connected components are extracted.

After connected components are extracted, the bounding boxes of these components are calculated. The components whose height is smaller than 10 pixels are eliminated in a noise removal step. Moreover, the bounding boxes are normalized according to a constant column length which is equal to 100 pixels in this case.

Then top and bottom reservoirs are obtained as described in [42]. Figure 3.2 illustrates an example of reservoirs obtained from respected connected component. The illustrated example is a connected digit pair of 55 and in this case there is no top reservoir for this image. Only the bottom reservoir help decide the segmentation path. Moreover, corresponding segmentation result of this example is shown in Figure 3.3.

(a) (b)

Figure 3.2: Water reservoir example

(a) Connected component (b) Corresponding bottom reservoir

Figure 3.3: Segmentation of touching numerals

After finding top and bottom reservoirs the classification algorithm described in [43] is applied deciding whether a component is isolated or not and the touching types of the connected numerals. Then, the selected drop-fall algorithm is applied on the touching numeral segmenting the pair into two isolated digits.

Some drawbacks of this algorithm are observed. One of the drawbacks is that when an isolated digit is given into the algorithm, an unnecessary segmentation points are sometimes found if a single digit is written in a style that it creates large reservoirs. One unnecessary segmentation results in wrong prediction on the value of the string.

One other drawback of this algorithm is that when more than two digits are connected to each other, it fails to correctly segment all the digits. However, since it is a rare possibility, the effect of this error is expected to be quite small in this thesis.

After the segmentation is completed. 40142 components are obtained and they are given to the digit recognition module as inputs.

3.4.2. Segmentation Using Deep Belief Networks

In order to segment handwritten digit strings, we proposed a recognition-based algorithm using deep belief nets. In this algorithm, we treated each image of digit strings as a concatenation of samples. These samples are formed by sliding a window

with a fixed size and their labels are assigned semi automatically as described previously.

The idea behind this algorithm is to classify each sample if it is a part of a number or it is in the gap between two digits. It is expected to give better results with touching digit pairs, since the recognition algorithm could be trained accordingly.

DBN structure is used as the classifier in this algorithm. The custom database mentioned in section 3.1.4. is used as training samples of the DBN. The input images are 100x10 pixels resulting in 1000 units in the input layer.

Test are done changing the number of epochs, batch size, learning rate and momentum of the DBN structure. Hinton's practical guide for training RBM [23] is used obtaining these values.

The best result was obtained with:

Hidden layer sizes = [500 500 2000]

Number of epochs = 5

Batch size = 10

Momentum = 0

Learning rate = 1

The success rate of the classifier under these conditions was 92%. In other words, 92% of the sub-images with size of 100x10 pixels are predicted correctly whether each sub-image belongs to a digit or a non-digit field.

Although a numerical success rate of segmentation is reached with this approach, it is observed that most of the errors occurred on the boundary of gap regions. Therefore, using these results, the neighbour windows with the same label are merged and segmentation cuts are placed at the centre of the merged gap regions. In this way, the

success rate for the segmentation increases. The increase in the segmentation is understood after applying handwritten digit recognition module which is mentioned in the next section.

After merging is completed. We reached 40198 images containing individual digits.

3.5. Handwritten Digit Classifier

This is the final step of the designed handwritten digit segment recognition system. The deep belief network structure used in the segmentation process is modified and used for the digit classifier. Support vector machines (SVM) is also used a baseline for comparison.

Before starting this step, another preprocessing part should be applied. This part should move the digit to the centre of mass of the segmented digits and normalize the resulting image into 28x28 pixel size.

Firstly, MNIST database is used for training and testing of both classifiers. Secondly, both classifiers are trained with the digits which are successfully segmented in the earlier step, and tested with the result of segmentation algorithm whose result is unknown whether it is successful or not.

3.5.1. Support Vector Machines

Multiclass SVM classifier is formed using one versus rest relation. Training is done by finding a decision boundary for each class separating it from the rest of the classes. Different kernel functions are tested with multiclass SVM.

The results for MNIST database:

- Quadratic Kernel:

Recognition rate: 92% with 5000 training and 100 test samples

Recognition rate: 93.6% with 20000 training and 1000 test samples

- Polynomial Kernel:

Recognition rate: 73% with 5000 training and 100 test samples

Recognition rate: 82% with 20000 training and 1000 test samples

- Linear Kernel:

Recognition rate: 81% with 5000 training and 100 test samples

The best performing kernel on MNIST dataset is the quadratic kernel, thus quadratic kernel is considered in the rest of the experiments.

After testing the SVM module with MNIST dataset. The segmented digits from the previous step is fed into the SVM algorithm. Segmented digits from previous step are tested only with Quadratic kernel.

First, the digits that are segmented using baseline segmentation algorithm, that is the one based on the water reservoir concept, is tested with this module trained with MNIST database. 40142 digits are tested with the quadratic kernel. As a result, 88.4% recognition rate of digits is achieved.

Then, the segmented digits from the proposed segmentation algorithm that is the one based on deep belief nets, is fed into the SVM module trained with MNIST database. 40198 digits are tested and 90.2% recognition rate is achieved.

Moreover, SVM module trained using the custom made dataset containing 7344 digits from CVL Strings training samples. Then, the same tests are repeated and the following results were obtained.

Recognition rate of the digits from the baseline segmentation algorithm: 80.2%

Recognition rate of the digits from the proposed segmentation algorithm: 82.1%

3.5.2. Deep Belief Networks

Tests on DBN are done forming a four layer DBN structure with three hidden layers. Training is done one layer at a time and mean square error is used to calculate the average reconstruction error. After training is completed, resulting DBN is unfolded into a multilayer neural network and back propagation is applied using labelled data. Then tests are done using the test sample set and error rate is calculated.

Hinton's practical guide for training RBM [23] is used for obtaining parameters. The average reconstruction error helped monitoring the learning process. Optimization of the learning rate, batch sizes are done using the insight that this parameter brings. Average reconstruction error is defined as the average of the sum of the squared distance between input data in the RBM and the reconstructed data in the input layer with respect to the hidden layer and the weights.

Moreover, the RBMs in the DBN are trained passing all the training samples simultaneously through the RBM at once, then the weights are updated. This procedure is named as epoch. In general, it is defined as completed iteration of the training procedure. Increasing the number of epochs improves the performance of the DBM in expense of computational time.

In the constructed DBN, input layer consists of 784 units, since the input images are 28x28 pixels. Tests are done changing the number units ($[h_1\ h_2\ h_3]$) in the hidden layers and changing the number of epochs for training each RBM in this structure.

All the tests are done with 60000 training and 10000 test samples and the following cases are examined.

Case 1:

When $[h_1\ h_2\ h_3]$ = [100 100 100] with 5 epochs while training each RBM:

Recognition Rate = 93.61%

The reconstruction error versus the number of total epochs for this case is illustrated in Figure 3.4.

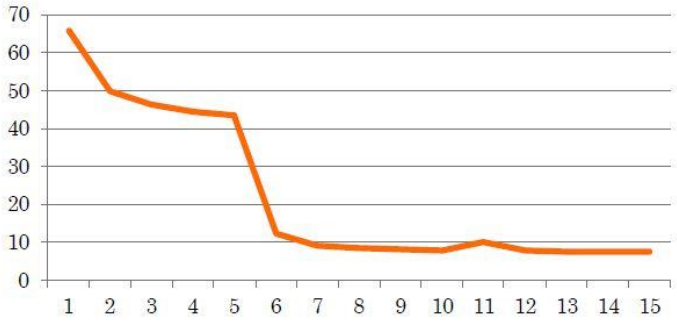


Figure 3.4: Average reconstruction error per epoch in case 1 for MNIST

Case 2:

When $[h_1\ h_2\ h_3] = [500\ 500\ 2000]$ with 5 epoch while training each RBM:

Recognition Rate = 94.81%

The reconstruction error versus the number of total epochs for this case is illustrated in Figure 3.5.

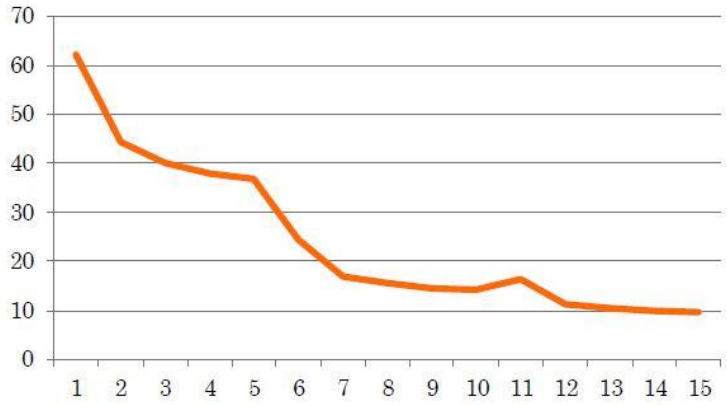


Figure 3.5: Average reconstruction error per epoch in case 2 for MNIST

Case 3:

When [h_1 h_2 h_3] = [500 500 2000] with 50 epochs while training each RBM:

Recognition Rate = 98.08%

The reconstruction error versus the number of total epochs for this case is illustrated in Figure 3.6.

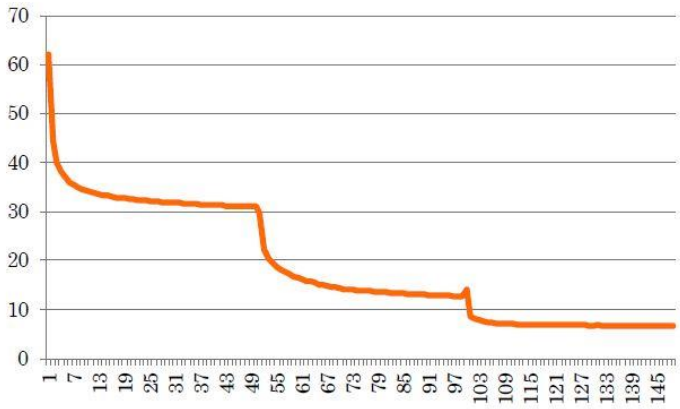


Figure 3.6: Average reconstruction error per epoch in case 3 for MNIST

The digits segmented from earlier segmentation step are only tested with case 3, since it has the best result with MNIST. First, the tests are done with using MNIST as the training database. Then, the custom digit dataset is used for training the DBN and the segmented digits from the earlier segmentation modules are tested.

3.6. Experimental Results

In this chapter, the overall results of the experiments done on CVL-Strings database. For the performance metrics, the recognition rate of segmented digits, the overall recognition rate on the digit strings and average normalized Levenshtein Distance is used for comparison of the applied methods.

The experiments varies according to the segmentation algorithm, the digit recognition module and the training dataset of the recognition module (MNIST or digits from the CVL-Strings).

For the segmentation module, two approaches are implemented. First the drop-fall algorithms combined with water reservoir concept is implemented as a baseline

algorithm. Then, the novel approach introduced before, segmentation using DBN, is applied.

The results of the both segmentation algorithms are used in the digit recognition modules which are support vector machines and deep belief networks. These recognition modules are trained with MNIST database. The digit recognition modules are also implemented using correctly extracted digits from CVL-Strings dataset as their training samples.

Table 3.1 lists the experiments conducted using CVL-Strings dataset. In the experiments, there are 8 different combinations of segmentation module, digit recognition module and the training dataset.

Since there are a number of parameters of DBN and SVM modules that could be carried out, the overall tests are implemented using the parameters that have given the best results in the previous sections.

Table 3.2 shows the results of the listed experiments according to the defined performance metrics. The results are compared according to the recognition rate of segmented digits, the recognition rate of strings and average normalized Levenshtein Distance.

Moreover, Figure 3.7-Figure 3.9 demonstrates the comparison of results graphically in detail.

Table 3.1: List of Conducted Experiments and Their Abbreviations

Abbrv.	**Segmentation Module**	**Digit Recognition Module**	**Training Dataset**
CVL-1	Segmentation using water reservoir concept	SVM	MNIST
CVL-2	Segmentation using water reservoir concept	SVM	CVL-Strings
CVL-3	Segmentation using water reservoir concept	Deep Belief Networks	MNIST
CVL-4	Segmentation using water reservoir concept	Deep Belief Networks	CVL-Strings
CVL-5	Deep Belief Networks	SVM	MNIST
CVL-6	Deep Belief Networks	SVM	CVL-Strings
CVL-7	Deep Belief Networks	Deep Belief Networks	MNIST
CVL-8	Deep Belief Networks	Deep Belief Networks	CVL-Strings

Table 3.2: Comparison of Results of Conducted Experiments

Abbreviation	**Recognition Rate of Segmented Digits**	**Overall Recognition Rate Of Digit Strings**	**ANLD**
CVL-1	88.4%	76.2%	0.15
CVL-2	80.2%	69.8%	0.19
CVL-3	89.7%	77.8%	0.10
CVL-4	82.5%	72.8%	0.13
CVL-5	90.2%	78.2%	0.09
CVL-6	82.1%	73.0%	0.14
CVL-7	97.0%	89.1%	0.06
CVL-8	91.1%	86.2%	0.07

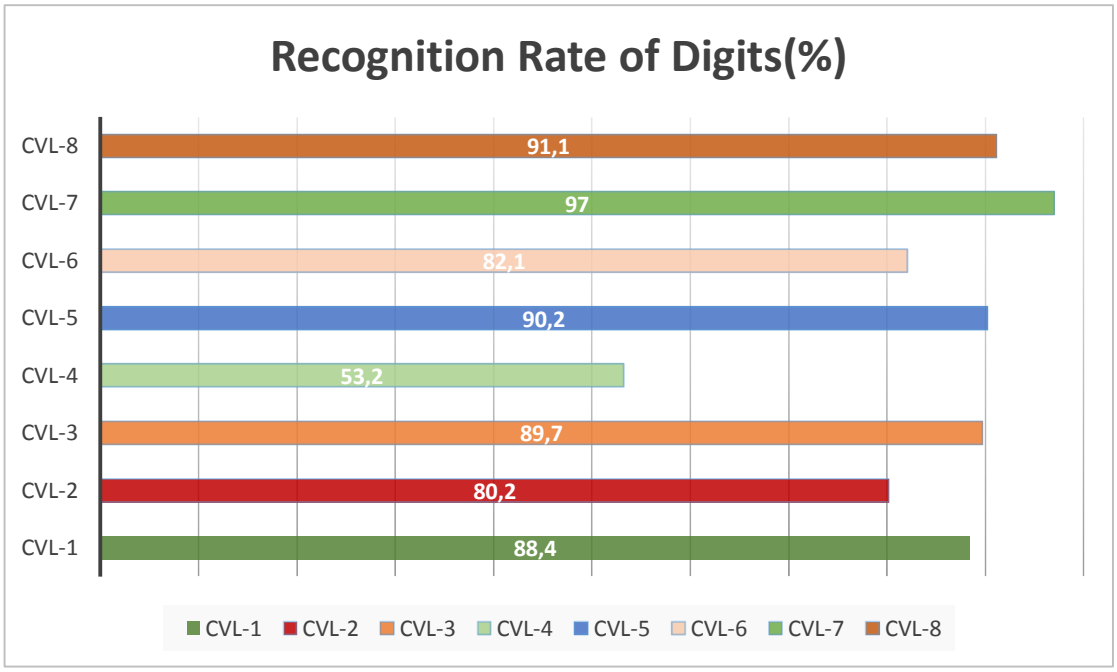


Figure 3.7: Recognition Rate of Segmented Digits

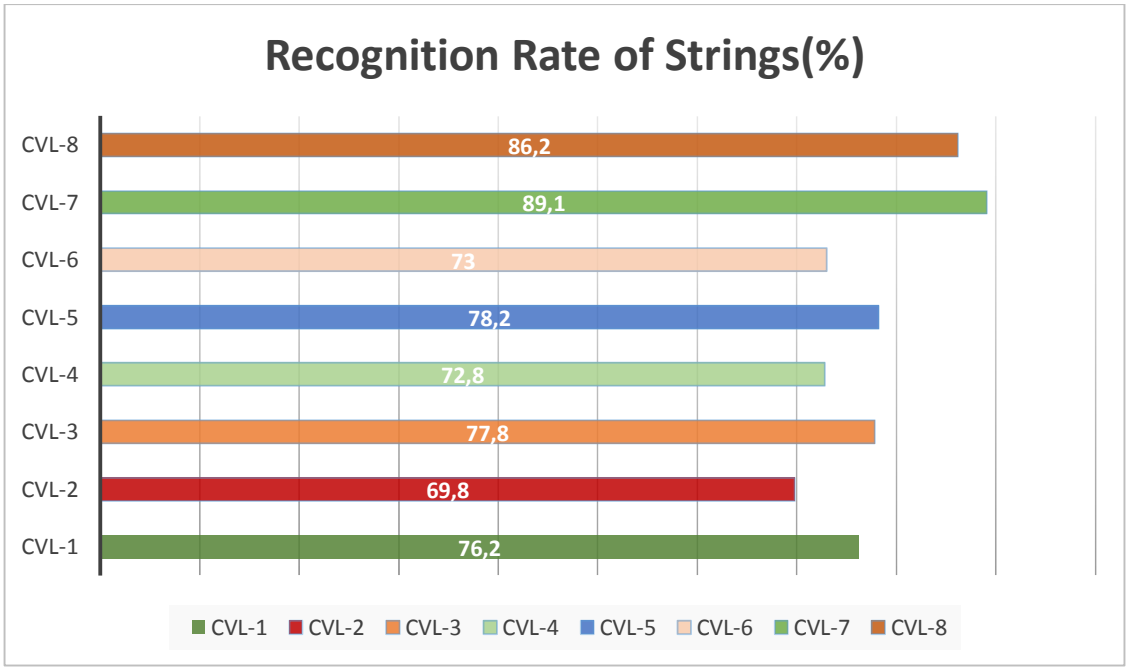


Figure 3.8: Recognition Rate of Overall Strings

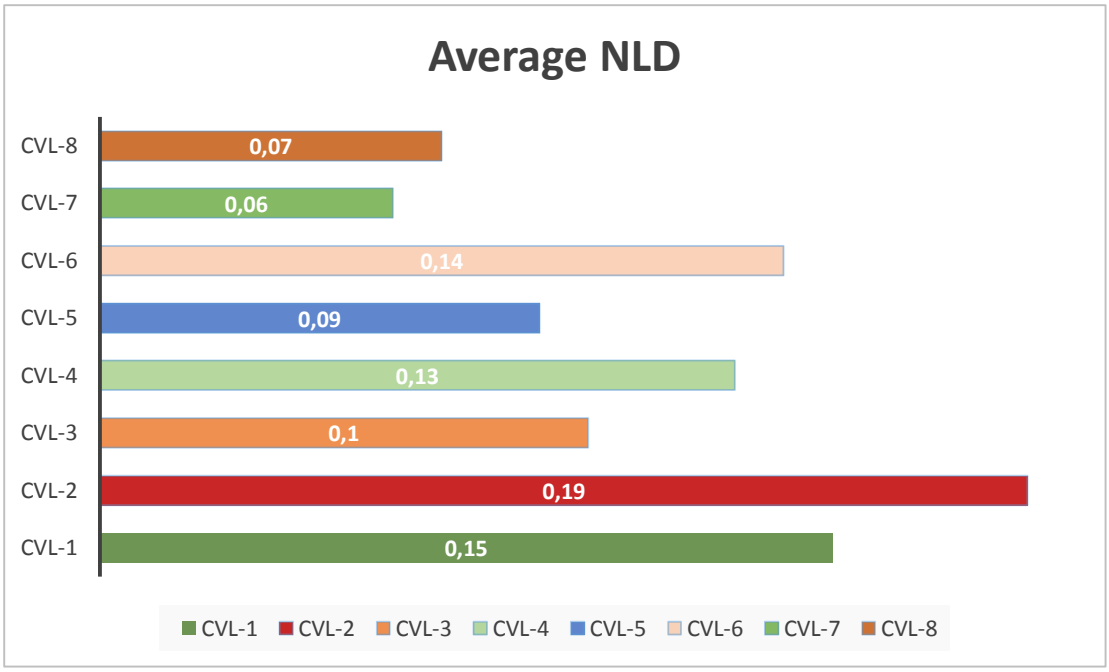


Figure 3.9: Average Normalized Levenshtein Distance

From the results, it is seen that CVL-7 case significantly outperformed other cases. Since the custom dataset including digits from correctly segmented images from CVL-Strings database have less training samples for each label, the digit recognition algorithms trained with this dataset performed worse than the ones trained with MNIST dataset.

Moreover, ANLD result is included in order to compare the distance between wrong classification result and the target digit string. In other words, our aim is to analyse the performance of the segmentation module. If a digit string is poorly segmented, then the ANLD result goes higher showing that the predicted digit string is a lot further away from the actual digit string.

Therefore, it is seen that the cases using the baseline segmentation algorithm have higher ANLD values showing that the reason of misclassification of digit strings is mostly the segmentation errors.

3.7. Comparison of the Results with Benchmark Algorithms

In this section, the results obtained from the experiments are compared with the benchmark algorithms explained earlier. Figure 3.10 and Figure 3.11 illustrate the comparison of the results.

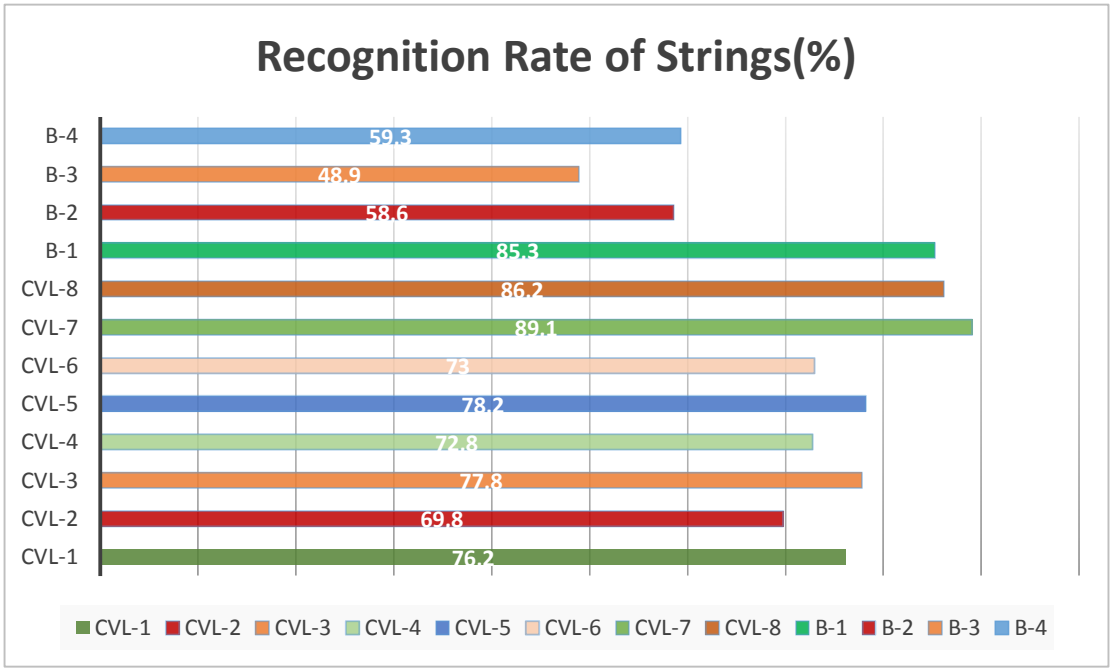


Figure 3.10: Comparison of Recognition Rates between Experiments and Benchmark Algorithms

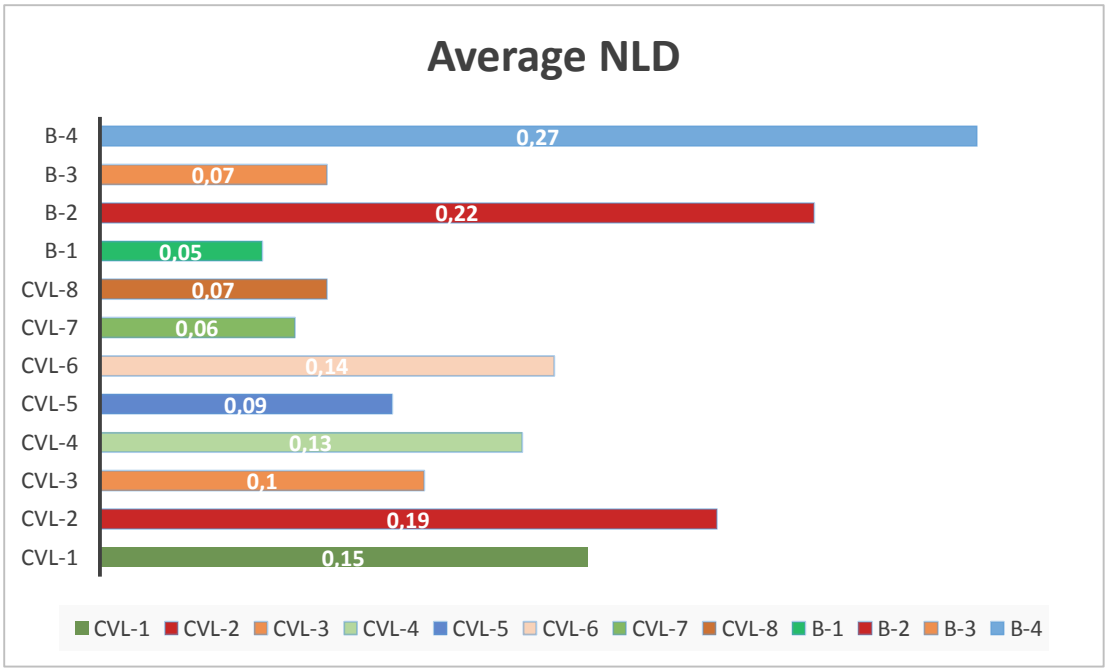


Figure 3.11: Comparison of ANLD between Experiments and Benchmark Algorithms

From the results, it is observed that the closest performance of the benchmark algorithms to our results is the first benchmark algorithm. CVL-7 and CVL-8 outperformed all the benchmark algorithms in terms of recognition rate.

However, B-1 performed slightly better in terms of average normalized Levenshtein distance.

CHAPTER 4

CONCLUSION AND FUTURE WORKS

Handwritten text recognition is a simple process for human beings. The complex interactions that the neurons in the human brain accomplish are often taken for granted. Human brain has the ability to recognize the written text simultaneously using the features of the text and knowledge that had been learned during life time. From these clues, it is understood that systems with higher performance for handwritten digit string recognition task or any other pattern recognition problem could be built using models resembling human brain.

The main concern of this study was recognition of handwritten digit strings which could be divided into 3 steps which are preprocessing, segmentation and recognition steps. There are various studies that have been conducted in this field. However, handwritten digit strings might differ in many ways. Writing styles and origin of the authors are some of the sources of the differences in these strings. Moreover, the application that the hand writing is needed is also effective on these differences. For instance, there could be some guidelines for some applications such as courtesy fields of the bank checks, making easier to horizontal and vertical orientation of the string.

Most of the studies conducted in this field take the differences mentioned earlier take into consideration in their proposed methods. In other words, different heuristics are defined according the dataset that the algorithms are built for. Moreover, the preprocessing step is another thing that differs depending on the dataset used.

In order to build a universal algorithm for the handwritten digit string recognition task, the heuristics should not depend on the context that could differ from source to source. The preprocessing applied before the segmentation step should also be adaptive to the

data used or more general techniques could be used. Moreover, the recognition modules that could be used, should be trained with various datasets from different sources in order to make the recognition part more universal.

The trickiest part of the handwritten digit string recognition task is the segmentation module which is responsible for segmenting the digit string into individual characters. This is actually a simple task if the digits are well separated. However, this is mostly not the case. There are various types of touching numerals. Some of the methods described in the proposed algorithms are good at some of the touching styles. The digit pairs that touch each other in multiple points or overlapping digits are the most difficult ones to separate.

In this study, a more universal technique in handwritten digit string segmentation is proposed. For the segmentation, a recognition based algorithm is applied, which is free of any heuristics, in expense of some recognition error. In this method, a DBN classifier is used for classifying regions in an image of a digit string as a digit or the transition region between two digits. Segmentation is completed after merging neighbor regions which have the same label.

In this thesis, the algorithm, which Rui et al. [43] proposed, was also implemented and tested with the same dataset. This algorithm was chosen for implementation, since the success rate claimed by the authors was quite high and it does not include any training, but heuristics.

After the segmentation is completed, resulting individual digits were given as inputs to the digit recognition algorithms. For the digit recognition algorithms, deep belief networks and support vector machines were chosen. The recognizers were trained with MNIST database and the custom dataset formed using the accurately segmented digits from the CVL-Strings dataset.

After the recognition of individual digits was achieved, the prediction of the digits in each strings were combined together and compared with the ground truth of the corresponding digit string. Hence, the overall recognition rates were achieved.

Moreover, average normalized Levenshtein distance (ANLD) parameter was also employed as performance metric.

Overall results were calculated with each segmentation and recognition algorithms, which trained with both dataset separately, resulting in 8 different combination of experiments. The overall results were compared for each combination. The combination of the proposed recognition based segmentation algorithm with DBN digit classifier trained with the MNIST database has outperformed other combinations.

Moreover, 4 benchmark algorithms that were tested with the CVL-Strings database, the same database we have used in our experiments, are chosen for comparison of our experiments. This algorithms are chosen because of the fact that they were chosen as top ranked methods in ICFHR 2014. The results showed that the proposed method here had better performances according to the recognition rates.

When compared to the benchmark algorithms, the segmentation algorithm which uses the water reservoir concept [43] had very competitive results with the benchmark algorithms.

In addition to the recognition rates, the ANLD parameter was also calculated to compare the handwritten digit string recognition algorithms. The definition of this parameter helped to decide how close a misclassified digit string is to the actual value of the string. From this point on, it is observed that the segmentation using water reservoir concept has a promising result in terms of this parameter. In other words, it is observed that even though it has lower recognition rates, the distance of the misclassified samples to the actual values are not very large.

In the future, the proposed technique can be experimented with various datasets in order to become a candidate to be a universal technique solving this problem. Moreover, other recognition algorithms i.e. convolutional neural networks can be experienced and compared their results to the existing ones.

REFERENCES

[1] R. Jajadevan, S.R. Kolhe, P.M. Patill and U. Pal, "Automatic processing of handwritten bank cheque images: a survey", in IJDAR, 2012 © Springer-Verlag, doi: 10.1007/s10032-011-0170-8

[2] P.K. Sahoo, S. Soltani and A.K.C. Wong, "A survey of thresholding techniques", Comput. Vis. Graph. Image Process, 1988, 41(2), pp. 233-260

[3] N. Otsu, "A threshold selection method from gray-scale histogram", IEEE Trans. Syst. Man Cybern, 8, pp. 62-66, 1978

[4] O. D. Trier and A. K. Jain, "Goal-directed evaluation of binarization methods", IEEE Trans. Pattern Anal. Mach. Intell., 17(12), pp. 1191-1201, 1995

[5] M. Sezgin and B. Sankur, "Survey over image thresholding techniques and quantitative performance evaluation", J. Electron. Imaging, 13(1), pp. 317-327, 2004

[6] S. Knerr, V. Anisimov, O. Baret, N. Gorski, D. Price and J. Simon, "The A2iA intercheque system: courtesy amount and legal amount recognition for French checks", Int. J. Pattern Recognit. Artif. Intell., 11(4), pp. 505-548, 1997

[7] F. C. Ribas, L.S. Oliveira, A.S. Britto Jr., and R. Sabourin, "Handwritten digit segmentation: a comparative study", Springer-Verlag, 2012

[8] R. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation", IEEE Trans., PAMI 18(7), pp. 690-706(1996)

[9] J. Sadri, C.Y. Suen and T.D. Bui, "A generic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings", Pattern Recog. 40, pp.898-919, 2007

[10] H. Fujisawa, Y. Nakano and K. Kurino, “Segmentation methods for character recognition: from segmentation to document structure analysis”, Proc. IEEE 80, pp. 1079-1092, 1992

[11] R. Fenrich and S. Krishnamoorthy, “Segmenting diverse quality handwritten digit strings in near real-time”, Proceedings of 5th USPS Advanced Technology Conference, pp. 523-537, 1990

[12] D. Yu and H. Yan, “Separation of touching handwritten multi-numeral strings based on morphological structural features”, Pattern Recognit. 34(3), pp. 587-598, 2001

[13] A. Elnagar and R. Alhajj, “Segmentation of connected handwritten numeral strings”, Pattern Recognit., 36(3), pp. 625-634, 2003

[14] M. Suwa and S. Naoi, “Segmentation of handwritten numerals by graph representation”, Proceedings of 9th International Workshop on Frontiers of Handwriting Recognition (IWFHR), Tokyo, 2004

[15] Y.K. Chen and J.F. Wang, “Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis”, IEEE Trans. Pattern Anal. Mach. Intell. 22(11), pp.1304-1317, 2000

[16] M. Leroux et al., “Automatic reading of handwritten amounts on French checks”, Int. J. Pattern Recognit. Artif. Intell. 11(4), pp. 619-638, 1197

[17] L. L. Lee at al.,” A prototype for Brazilian bank check recognition”, Int. J. Pattern Recognit. Artif. Intell. 11(4), pp. 549-579, 1997

[18] G. E. Hinton et al., “A fast learning algorithm for deep belief nets”, Neural Computation, 2006

[19] D. Decoste and B. Schoelkopf, “Training invariant support vector machines”, Machine Learning, 46, pp. 161-190, 2002

[20] Y. LeCun, “The MNIST Database”, Available:
http://yann.lecun.com/exdb/mnist/

[21] Markus Diem, Stefan Fiel, Angelika Garz, Manuel Keglevic, Florian Kleber and Robert Sablatnig, ICDAR 2013 Competition on Handwritten Digit Recognition (HDRC 2013), In Proc. of the 12th Int. Conference on Document Analysis and Recognition (ICDAR) 2013, pp. 1454-1459, 2013

[22] F. Meyer, "Topographic distance and watershed lines," Signal Processing, Vol. 38, July 1994, pp. 113-125.

[23] G. Hinton, “A practical guide to training Restricted Boltzmann Machines”.

[24] G. E. Hinton. Training products of experts by minimizing contrastive divergence. Neural Computation, 14(8):1711–1800, 2002.

[25] R. Salakhutdinov, “Learning Deep Generative Models”, Ph.D dissertation, Department of Computer Science, University of Toronto, 2009.

[26] A. Kaur and Aayushi, “Image Segmentation Using Watershed Transform”, International Journal of Soft Computing and Engineering (IJSCE), ISSN: 2231-2307, Volume-4, Issue-1, March 2014.

[27] Tutorial: Image Segmentation Yu-Hsiang Wang Institute of Communication Engineering National Taiwan University, Taipei, Taiwan, ROC

[28] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 3rd ed., Prentice Hall, New Jersey 2008.

[29] M. Diem, S. Fiel, F. Kleber, R Sablatnig, J.M. Saavedra1, D. Contreras, J.M. Barrios and L.S. Oliveira, “ICFHR 2014 Competition on Handwritten Digit String Recognition in Challenging Datasets (HDSRC 2014)”

[30]Yi-Chao Wu, Fei Yin, Chang Zhong, Cheng-Lin Liu ICFHR Competition on Handwritten Digit String Recognition, 2014

[31] C.-L. Liu, H. Sako, and H. Fujisawa, “Effects of Classifier Structures and Training Regimes on Integrated Segmentation and Recognition of Handwritten Numeral Strings,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 26, no. 11, pp. 1395– 1407, Nov. 2004.

[32]F. Yin, Q.-F. Wang, and C.-L. Liu, “Transcript Mapping for Handwritten Chinese Documents by Integrating Character Recognition Model and Geometric Context,” Pattern Recogn., vol. 46, no. 10, pp. 2807–2818, Oct. 2013.

[33]Pernambuco: Universidade de Pernambuco, Santo Amaro, Brasil (Byron Leite and Cleber Zanchettin)

[34]B. L. D. Bezerra, G. D. C. Cavalcanti, Z. C., and J. C. B. Rabelo, “Detecting and Treating Invasion in the Courtesy Amount Field on Bank Checks,” in 11th International Conference on Frontiers in Handwriting Recognition (ICFHR 2008), 2008.

[35] J. Rabelo, C. Zanchettin, C. A. B. Mello, and B. L. D. Bezerra, “A Multi-Layer Perceptron Approach to Threshold Documents With Complex Background,” in 2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2011, pp. 2523–2530.

[36]A. Graves and J. Schmidhuber, “Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks,” in Advances in Neural Information Processing Systems 21, 2008, pp. 545–552.

[37] B. L. D. Bezerra, C. Zanchettin, and V. B. de Andrade, “A MDRNN-SVM Hybrid Model for Cursive Offline Handwriting Recognition,” in Proceedings of the 22nd International Conference on Artificial Neural Networks and Machine Learning - Volume Part II, ser. ICANN’12, 2012, pp. 246– 254.

[38] Shanghai: ECNU-SRI Joint Lab for Pattern Analysis and Intelligent System, Shanghai Research Institute of China Post, Shanghai, China (Shujing Lu and Yue Lu)

[39]Tebessa I: LAMIS Laboratory University of Tebessa, Algeria, (Abdeljalil Gattal) and Speech Communication and Signal Processing Laboratory, Faculty of Electronics and Computer Science, University of Sciences and Technology Houari Boumedienne, Bab-Ezzouar, Algiers, Algeria (Youcef Chibani)

[40]A. Gattal and Y. Chibani, "Segmentation Strategy of Handwritten Connected Digits (SSHCD)," in Image Analysis and Processing – ICIAP 2011, ser. Lecture Notes in Computer Science, G. Maino and G. Foresti, Eds., vol. 6979. Springer Berlin Heidelberg, 2011, pp. 248–254.

[41] A. Gattal and Y. Chibani, "Segmentation and Recognition Strategy of Handwritten Connected Digits Based on the Oriented Sliding Window," in 2012 International Conference on Frontiers in Handwriting Recognition (ICFHR), 2012, pp. 297–301.

[42] U.Pal, A. Belaid, and Ch. Choisy, "Touching Numeral Segmentation Using Water Reservoir Concept," Pattern Recognition Letters, 2003, Vol. 24, pp. 261-272.

[43] M. Rui, Z. Yingman, X. Yongquan and Y. Yunyang, "A Touching Pattern-oriented Strategy for Handwritten Digits Segmentation", International Conference on Computational Intelligence and Security, 2008

[44] S. A. Khan, "Character Segmentation Heuristics for Check Amount Verification", MsC Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 1998

[45] G. Congedo, G. Dimauro, S. Impedovo, and G. Pirlo, "Segmentation of Numeric Strings", Proc. of Third Int. Conf. on Document Analysis and Recognition, Montreal, August 14-16, 1995, pp.1038-1041

[46] A. Alatan, "EE 583 Lecture Notes", 2013