DIRECTION HISTOGRAM: NOVEL DISCRIMINATIVE GLOBAL FEATURE FOR THAI OFFLINE HANDWRITTEN OCR

EKAWAT CHAOWICHARAT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
(MATHEMATICS)
FACULTY OF GRADUATE STUDIES
MAHIDOL UNIVERSITY
2016

Thesis
entitled

DIRECTION HISTOGRAM: NOVEL DISCRIMINATIVE GLOBAL FEATURE FOR THAI OFFLINE HANDWRITTEN OCR

Ekawat Chaowicharat

Mr. Ekawat Chaowicharat
Candidate

Assoc. Prof. Kanlaya Naruedomkul,
Ph.D. (Computer Science)
Major advisor

(Deceased)

Prof. Nick Cercone,
Ph.D. (Computer Science)
Co-advisor

Lalita Narupiyakul

Lect. Lalita Narupiyakul,
Ph.D. (Computer Science)
Co-advisor

Prof. Patcharee Lertrit,
M.D., Ph.D. (Biochemistry)
Dean
Faculty of Graduate Studies
Mahidol University

Montip Tiensuwan

Assoc. Prof. Montip Tiensuwan,
Ph.D. (Applied Statistics)
Program Director
Doctor of Philosophy Program in
Mathematics,
Mahidol University

Thesis
entitled

DIRECTION HISTOGRAM: NOVEL DISCRIMINATIVE GLOBAL FEATURE FOR THAI OFFLINE HANDWRITTEN OCR

was submitted to the Faculty of Graduate Studies, Mahidol University
for the degree of Doctor of Philosophy (Mathematics)
on
March 22, 2015

Mr Ekawat Chaowicharat
Candidate

Assoc. Prof. Wanchai Rivepiboon,
Ph.D. (Computer Science)
Chair

Assoc. Prof. Kanlaya Naruedomkul,
Ph.D. (Computer Science)
Member

Lect. Lalita Narupiyakul,
Ph.D. (Computer Science)
Member

Lect. Wararat Wongkia,
Ph.D. (Science and Technology Education)
Member

Prof. Patcharee Lertrit,
M.D., Ph.D. (Biochemistry)
Dean
Faculty of Graduate Studies
Mahidol University

Assoc. Prof. Sittiwat Lertsiri,
Ph.D. (Agricultural Science)
Dean
Faculty of Science
Mahidol University

ACKNOWLEDGEMENTS

The success of this thesis can be succeeded by the attentive support from my advisors including Assoc.Prof. Kanlaya Naruedomkul, Prof. Nick Cercone, and Lect. Lalita Narupiyakul who help to teach, encourage me to do this research, and compiled them into this dissertation successfully.

I would like to thank the teachers, classmates, and officers in the Department of Mathematics. They also cheer and advise me to do paper works. They also gave me consultation time when I felt discouraged.

Thanks to the families and my beloved who are always beside me throughout the study. They supported me for everything. In particular, Miss Chatchadaporn Promnork, my girlfriend who always stands by me and goes through every difficulty together from the very first time until the present. I really appreciate the beauty and generosity, always and forever.

I would like to thank every participant who helps provide the handwritten dataset in this study. They helped to fulfil this thesis to complete and make valuable for this dissemination to society and those interested in further study.

Lastly, all of my studying budget, funding for my trip to Canada, my devices I used, came from the scholarship provided by Development and Promotion of Science and Technology Talents Project (DPST).

Ekawat Chaowicharat

DIRECTION HISTOGRAM: NOVEL DISCRIMINATIVE GLOBAL FEATURE FOR THAI OFFLINE HANDWRITTEN OCR

EKAWAT CHAOWICHARAT 5237083 SCMA/D

Ph.D. (MATHEMATICS)

THESIS ADVISORY COMMITTEE: KANLAYA NARUEDOMKUL, Ph.D.
NICK CERCONE, Ph.D., LALITA NARUPIYAKUL, Ph.D.

ABTRACT

The image feature used for classification is a crucial part of an optical character recognition system. To achieve a high accuracy for offline handwritten character recognition, the feature should capture and emphasize the differences between different characters classes and ignore the differences among the various drawings of the same character class. In this research, we present a novel image feature called direction histogram (DH) and a feature extraction algorithm called bag of histogram (BoH). Unlike the traditional image feature, DH is a global feature that describes pixel density in every direction around each center, which tolerates stroke thickness and variation, and omits the stroke connectivity (if any). BoH is the algorithm to count the proportion of distinct DHs in an unseen image and generate a one-dimensional feature vector. This vector is compressed by PCA and is classified by a neural network. Fifty-two datasets, each containing 30 drawings of 80 Thai characters, are used for training our OCR system, and the original, thick, and distorted handwritten datasets are used for testing. The recognition system with our proposed DH and BoH feature extraction yielded higher recognition accuracy when compared to the popular convolutional neural network.

KEY WORDS: OPTICAL CHARACTER RECOGNITION / DIRECTION HISTOGRAM / BAG OF HISTOGRAM / HANDWRITTEN / OFFLINE / NEURAL NETWORKS

107 pages

ฮิสโทแกรมทิศทาง: การหาคุณลักษณะแบบแยกแยะทั่วทั้งภาพชนิดใหม่ สำหรับระบบรู้จำตัวอักขระลายมือเขียนภาษาไทย

DIRECTION HISTOGRAM: NOVEL DISCRIMINATIVE GLOBAL FEATURE FOR THAI OFFLINE HANDWRITTEN OCR

เอกวัจน์ เชาว์วิชารัตน์ 5237083 SCMA/D

ปร.ด. (คณิตศาสตร์)

คณะกรรมการที่ปรึกษาวิทยานิพนธ์: กัลยา นฤดมกุล, Ph.D., NICK CERCONE, Ph.D., ลลิตา นฤปิยะกุล, Ph.D.

บทคัดย่อ

การหาคุณลักษณะของภาพเป็นส่วนสำคัญของการสร้างระบบรู้จำตัวอักขระให้มีความแม่นยำ คุณลักษณะของภาพที่ได้จะต้องแยกแยะความแตกต่างระหว่างตัวอักขระที่ต่างกันได้ แต่ต้องประนีประนอมต่อความผิดเพี้ยนที่ทำให้รูปร่างของตัวอักขระเดียวกันมีความแตกต่างกัน งานวิจัยนี้ได้นำเสนอคุณลักษณะของภาพที่ชื่อว่า Direction histogram (DH) และระเบียบวิธีหาคุณลักษณะของภาพที่เรียกว่า Bag of histogram (BoH) หลักการออกแบบ DH ใช้ความหนาแน่นของจุดภาพในแต่ละทิศทางรอบจุดศูนย์กลางแต่ละตำแหน่งมาสร้างเป็นเวกเตอร์ ค่าของ DH มีคุณสมบัติไม่เปลี่ยนแปลงอย่างมีนัยสำคัญเมื่อเปลี่ยนความหนาของเส้นที่ใช้เขียน เมื่อเกิดความบิดเบี้ยวของภาพตัวอักขระ และเมื่อเกิดการเชื่อมกันของจุดภาพ ส่วน BoH เป็นการสร้างเวกเตอร์ที่เป็นตัวแทนของภาพโดยใช้การนับ DH ที่ปรากฏซ้ำๆ กันในภาพหนึ่งๆ เวกเตอร์ที่ได้จะนำไปบีบอัดด้วย PCA และนำเข้าสู่ขั้นตอนการแยกแยะด้วย Neural Network ในการทดสอบได้ใช้ลายมือของกลุ่มตัวอย่างจำนวน 52 คน โดยเขียนชุดตัวอักขระภาษาไทยทั้ง 80 ตัว จำนวน 50 ครั้ง โดย 30 ครั้งแรกเพื่อเป็นข้อมูลที่ใช้ฝึกฝน Neural Network และอีก 20 ครั้งที่เหลือใช้เป็นลายมือสำหรับทดสอบระบบ ลายมือนี้ยังถูกทำให้หนาขึ้นและบิดเบี้ยวเพื่อใช้สำหรับทดสอบความสามารถในการรับมือกับความผันแปรของลายมือ ผลการทดสอบพบว่าระบบรู้จำตัวอักขระที่ใช้เวกเตอร์จาก BoH ให้ความแม่นยำสูงกว่าระบบที่สร้างจาก Convolutional neural networks

107 หน้า

CONTENTS

CONTENTS (cont.)

CONTENTS (cont.)

CONTENTS (cont.)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (cont.)

CHAPTER I
INTRODUCTION

1.1 Motivation

Life would be a lot easier if our mobile can capture and translate text of the unfamiliar languages when we are going abroad or reading the instruction manual in the foreign language. Visually impaired people will be able to access any ink-printed book by reading text on the paper if there is a “book reading machine” that helps them to read book out loud. Tedious work such as typing handwritten manuscripts through the keyboard can be replaced by a smart scanner that automatically transcribe paper documents into editable electronics files. License plate detection and recognition help polices to track and trace the criminal trail while they are trying to escape by car.

The key technology that underlies the examples above into the real world application is called Optical Character Recognition (OCR), which is the computer software that capable to digitize text image into editable format such as text file.

Some OCR-based applications are made available in the market, for example Word lens (Wikipedia, 2015), Google translate (with OCR) (Google Translate, 2015), book reading machine (Reading Devices and Machines, 2015). However, the low accuracy of OCR is the bottleneck that affects the overall accuracy. Although OCR has been in the active research field for a long time, their accuracy is not practicable.

The reliable Thai OCR has not so far been found available. The only existing OCR software is ArnThai from NECTEC released in 2004 (Thai OCR : Arnthai 2.5, 2015), which claim the 95% accuracy for printed characters in limited scripts. No other software is available since then although Thai OCR researches have been carried on over decades. As we look closer to OCR researches, it is a challenging problem because there are many difficult sub-problems such as the similar characters and the variation of shape, especially in handwritten domain.

One of the most important parts that influence recognition accuracy is the image feature extraction, i.e., how we represent or describe a given image so the recognition process can capture the common properties of the same character class and the discrimination between different character classes.

Handwritten OCR is quite complicated mainly because of the variety in writing styles and writing instruments, for example, uneven thickness, distortion, unwanted junction, lack of visual features.

1.2 Objective

Our main objective is to find an approach to handwritten Thai OCR that overcomes the problems caused by uneven thickness, distortion, unwanted junction, and lack of visual feature. The prototype of Thai OCR system is developed based on our designed approach.

1.3 Definition of terms

OCR (Optical Character Recognition) is the computer software that convert text image into the machine encoded editable text.

Handwritten character means individual character written by people, in contrast to the typeset character, which means the character created by printing machine.

Character set means the set of every character in a language. Each character set comprises the set of consonants, vowels, tone marks (in some languages), and digits.

Datasets means handwritten character sets collected from many writers. The datasets is divided into training set and test set.

CHAPTER II
LITERATURE REVIEWS

In this chapter, we explore the history, concept, relevant theory, and methods related to OCR. From the historical perspective, OCR is originated by mean of mechanical hardware with light-sensitive materials, and became the computer software when the computer technology has developed through time. OCR software itself has been developing continuously from the rigid process to the more flexible one. All the process underlying OCR is based on the developing image processing theory and the implementations.

2.1 What is OCR

Optical Character Recognition or OCR has many different definitions, for example, the definition by Longman is “computer software that recognizes numbers and letters of the alphabet which are written on paper, so that information from paper documents can be scanned into a computer” (Longman dictionary, OCR, 02). Merriam Webster defines OCR by “Scanning and comparison technique intended to identify printed text or numerical data” (Fink, 2008). Wikipedia also defines that OCR is the mechanical or electronic translation of scanned images of handwritten, typewritten or printed text into machine-encoded text (Wikipedia, Optical Character Recognition, 02). We can conclude that OCR is the computer software together with some specific hardware that converts a text image into a searchable and editable format.

The first OCR was invented by Gustav Tauschek (Tauschek, 1935). Then a similar OCR machine was proposed by Paul Handel (Handel, 1993). In the first era, OCR machines are mechanical devices that operate based on the principle of superposition of two geometric objects: if two objects in plane are identical, their borders are fit exactly when placed one of them on another. The machine consisted of a light source, a rotating circular plate holed into character patterns and a photoelectric

cell. When light is exposed to a character on paper, the reflection pass through the hole and hit the photoelectric cell. If the character fits exactly to the hole, black ink will absorb all of light in the character region so there is no light hit the cell. The switch corresponding to the angle of the circular plate (and also corresponding to the character) will be activated.

Modern OCRs on microcomputer are systems that consist of software modules that work together. Each module does a specific job. Until today, a number of approaches were proposed. Some approaches were applied repeatedly so there is much diversity in each process.

OCR can be classified by the input character styles into two types: typeset and handwritten OCR. Typeset OCR is designed to recognize character in various printed font styles where each identical character in the same font set are exactly the same shape. Handwritten OCR, on the other hand, is designed to recognize character written by hand. It must be capable to handle the variation of character shape even in the same handwriting.

OCR can also be classified by physical input source into online and offline. Online OCR requires the stylus hardware or touchable-screen to feed the sequence of writing coordinate to the computer (Thongkamwitoon, Asdornwised, & Aramvith, 2002; Budsayaplakorn, Asdornwised, & Jitapunkul, 2013). On the other hand, the offline OCR can read paper with written (or typed) text via a scanner (Airphaiboon, Sangworasil, & Kondo, 1994; Mitrpanont & Kiwprasopsak, 2002; Imprasert, 2009).

Obviously, off-line OCR is the more challenging problem than the on-line OCR because of the slighter provided data. Moreover, text on the off-line image needs to be extracted and segmented so that the factor of recognition error increases. We will explore the traditional methodologies used in off-line OCR in the next section.

2.2 Traditional steps in off-line OCR

Temporal data in the writing trace for on-line OCR is useful for character recognition because the writing trace is already one-dimensional data that can be transformed into a parametric curve in a real plane. Representing the curve as the

sequence of bending, character recognition process can be achieved by defining distance function between curves so that the similarity and dissimilarity can be extracted. Off-line OCR, on the other hand, cannot take advantage of the temporal data. Therefore the information used and the recognition process are different from that of the on-line OCR. There are many possible ways to cope with the offline data. However, there are mainly two approaches for off-line OCR: Template-based and Feature-based approach.

2.2.1 Template-based OCR

Template-based OCR is the matching between an unknown character image to the item in the set of template. Templates are usually in the form of pixel set. The OCR process is to find an item in the template set that best fit to the text image. There are mainly two types of templates: rigid and elastic templates.

Rigid template is the first idea of character recognition; it was invented and built mechanically before the beginning of digital computer (Handel, 1993). Rigid template is the set of character template the shape of character images are fixed. The concept of rigid template OCR is the superposition of two objects in plane: if an image can be translated until it fits perfectly to another one, we said that the two images are geometrically identical. Some OCRs allow affine transformation (rotation and scaling) of the template for better matching result. Matching process starts by finding the pixel cluster of a character in the document image. Then adjust the rotation and scaling parameter until the image pixel and the transformed template are best fit (Mori, 1992). The rigid template OCR is very limited in that it works with only one font style for each template set. For additional font style, more templates must be added. This method is suitable for typeset font with perfectly scanned text image.

The idea of elastic template comes from the property of elastic material like rubber. If we prepare a rubber template, it can be bent and stretch into a set of similar patterns by applying some amount of force to it.

Elastic template is a fixed set of character shapes together with the operator that can deform the template shape into variations of character style. So, one template can afford more than one font style. Recognition can be achieved based on

the matching under the cost of deformation. The less cost (or force) used in the deformation, the more likely the matching is correct.

2.2.2 Feature-based OCR

In feature-based OCR, the informative part of the image is not the image pixels, but the feature vector extracted from image (Mori, 1992). The feature vector will then be compared to those of the training image or classified by a certain algorithm. The principle to concern when design a feature-based OCR is to find the common characteristic among the variations of character shapes in the same character class, and find the difference between the distinct character classes.

Most of modern OCRs are feature-based or hybrid between template and feature-based. There are four traditional processes that we can find in feature-based OCRs: Pre-processing, feature extraction, categorization/ matching and post-processing.

Pre-processing

Pre-processing is the process to prepare the input image to be ready for the further process. The following tasks are found (but not necessary) in most of OCR pre-processing module. The process begins with repairing the noisy input image by the Noise Reduction. The noise affects the quality of image because it creates the unwanted pixels. The denoised text image will be segmented into a set of single character by using the character segmentation. Segmentation seems to be trivial if the characters were written separately, however, the characters can be connected (by nature of language), smeared or overlapped. Many OCRs use the thinning algorithm to extract the skeleton of each character image. In some systems, especially the handwritten OCR, the writing trace extraction is applied to recover the hand movement direction, which is useful in the recognition process. For multi-lingual OCRs, the language of the document must be determined by using the script identification before the recognition process.

Feature extraction

Image contains two-dimensional raw data with million greyscale pixels that contain little meaning in each pixel. Feature extraction is the process to convert image into the more informative, concised, lower-dimensional vector called feature. Feature is the representation that makes sense of human interpretation. The issue to be concerned when designing a feature for character recognition is: what is the common structure among the shape variation of the same character, and what is the discriminative structure between the different character classes. We will explore the feature used in OCR in a next few section.

Classification

Classification means finding the most appropriate class of the unknown object. An extracted feature can be considered as a point in the multi-dimensional space of all possible value of the feature. Mathematically, we can define a categorization function, to map the feature vector of the unseen image to the most suitable character class. The feature extraction and the categorization function are closely dependent. The commonly used classification techniques are neural networks, support vector machine, and hidden markov model.

Post-processing

Post-processing step is aimed to correct the error found in any steps of recognition. With the fact that the practical use of OCR is to recognize not only the single character but the entire paragraph of document, language model can be applied to detect and correct the recognition error in the recognized text.

2.3 Difficulty of handwritten Thai OCR

In this study, we focus on the handwritten Thai language OCR due to the absence of practical handwritten OCR software in Thai language. The difficulty comes from both “handwritten” and “Thai”. Handwritten OCR is the more difficult task than the typeset OCR since handwritten document itself contains variation from various sources caused by both nature of language and handwriting style as follows:

- Uneven thickness: thickness depends on pen width and pressure during the writing process. Unlike thick typeset characters, the thickness of handwritten character within the single character can change unevenly (Methasate, Marukatat, Sae-tang, & Theeramunkong, 2005).
- Distortion: character shape can be bent and stretched non-linearly, which makes unlimited variation of handwritten characters (Methasate, Marukatat, Sae-tang, & Theeramunkong, 2005).
- Unwanted intersection: due to the distortion and thickness, curve intersection can occur accidentally by the variation of shape itself. Unwanted intersection violates the shape topology and possibly results in recognition error.
- Lack of visual features: in the case that some features of a character are visually absent, the predefined feature fails to capture the key features of the character.

Examples of distortion, unwanted intersection and lack of features are shown in Figure 2.1.

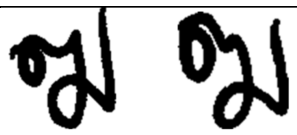
<table>
<tr><td> 
Typeset characters</td><td> 
Non-uniform bending and stretching</td></tr>
<tr><td> 
Unwanted intersection due to distortion</td><td> 
Lack of feature
Left: lack of zigzag stroke near the head.
Right: loop head and loop on bottom left are filled.</td></tr>
</table>

Figure 2.1 Variations of character ฆ (The 6th Thai character)

The difficulties of Thai character recognition are partly because of the nature of Thai character shapes. The most important issue is the similarity between

character pairs. For some character pairs, their dissimilarities are hardly identified. The lack or unclear of visual feature can possibly make one look like another. For example, the only different between ข and ฃ, the first letter pair shown in Figure 2.2, is the zigzag at the head of ฃ. The ฃ could possibly be recognized as 'ข' if its zigzag head is not noticeable.

Figure 2.2 shows Thai character pairs with four different discriminative visual features: zigzag stroke, head existence, head orientation, and tail existence and length. Head in Thai character is a small loop written as the starting point of the writing stroke. Head can be written in two different orientations: clockwise and counterclockwise. Tail, on the other hand, is the ending stroke, usually refers to ending part of character that goes beyond or below the level of the main line. Zigzag stroke is the single notch appearing at the head, the base, or the top of the character. Every character that contain zigzag stroke always have another similar character with no zigzag as shown in the first row of Figure 2.2.

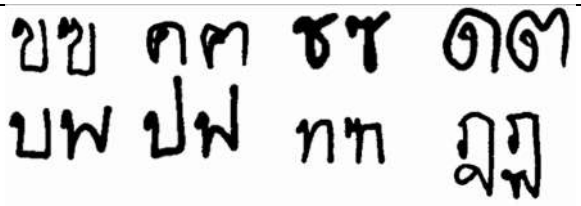
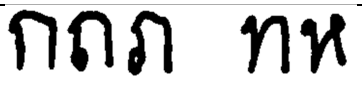
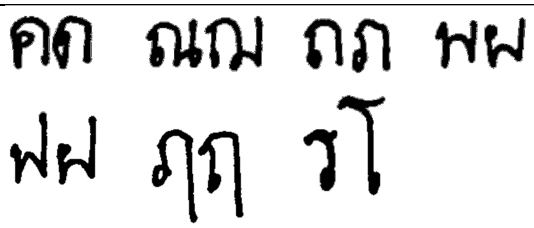
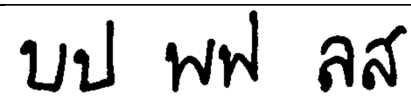
Zigzag stroke	
Head or loop existence	
Head orientation	
Tail existence and tail length	

Figure 2.2 Examples of similar characters categorized by their discriminative visual feature

If the characters are written clearly, these four visual features can be used to discriminate character classes. Unfortunately, these visual features can be easily violated by individual writing style.

2.4 Features used in Thai Handwritten OCR

Features used in Thai handwritten OCR can be classified into three groups, i.e., traditional global features, language specific features, and geometric/topological features. Different group of features usage has a trade-off between the existence, the between-class discriminability and the within-class similarity.

Traditional global feature such as width/height ratio, pixel density, number of components always exist in any shape of character. They can be measured objectively without any argument. However, the existence of these features does not directly identify the character identity. Usually, traditional global features are used in coarse level classification that divides the character set into many groups of characters.

Language specific feature such as beak, leg, head, zigzag are human interpretable. So it conveys high discrimination power if the feature can be extracted. However, it is not guaranteed that the language specific features exist due to the variation of handwritings that might violate the feature itself.

Geometrical/topological feature such as number of components when sliced (number of island), contour, gradient represent image locally by the pixel properties in each area. These features always exist and convey good discriminating power. Nevertheless, the locality of the feature makes it sensitive to the geometrical and topological variations of shape such as noises and unwanted intersection. The following discussion focuses on image features of Thai handwritten OCR.

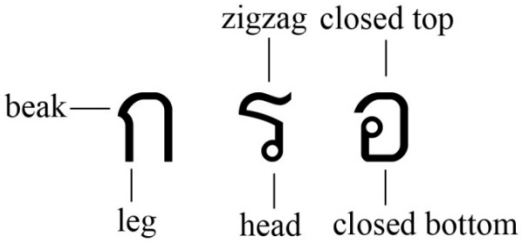


Figure 2.3 Example of language specific feature.

Airphaiboon et al. (Airphaiboon, Sangworasil, & Kondo, 1994) proposed Thai handwritten OCR based on pre-defined features such as end point (EP) and loop (LP) (Airphaiboon, Sangworasil, & Kondo, 1994). Features are extracted and formed into a set of rules called classification pattern (CP). Characters are divided into four groups based on their CP. In each group, sub-region is applied differently to the image in order to extract finer level features. At last, feature code (FC), loop type (LT), sub-subregions (SR) and width/height ratio (WH) are extracted. The classification tree can be built so that each character has distinct classification rule associated with these features.

Phokharatkul and Kimpan (Phokharatkul & Kimpan, 1998) proposed the method using features based on cavity in six types including north, east, south and west opening cavity, center cavity, and loop (Phokharatkul & Kimpan, 1998). Image is segmented into four pieces at the center of gravity. Number of each cavity type in every piece is counted. Then the counting will be the feature vectors for neural network classification. Finer classification is based on cavity analysis of head, i.e., discriminating head orientation based on the type of surrounding cavity.

In 2005, Sankhuangaw applied ant-miner algorithm to collect classification rule by using ant colony optimization (Sankhuangaw , 2005). Features are head zone, number of island, notch, end point, feature code (number of island the reference line pass), width/height ratio. Each training case creates a classification rule. Ant-miner algorithm selects the most optimal rule that describes every training case in each class by bringing forth the repeated rule that is found in the same character class.

In 2005, Methasate et al. proposed the feature combination technique (Methasate, Marukatat, Sae-tang, & Theeramunkong, 2005). The method exploits both global feature and local feature in the recognition process. Global feature such as pixel distribution and aspect ratio are used for classification into 20 groups of similar characters. Local feature such as loop, end point, junction and curl are used for fine classification.

In 2002, Mitrpanont proposed method based on extracting language specific feature at the appropriate region (Mitrpanont & Kiwprasopsak, 2002). The method automatically selects region to cover image pixels that represent analogous predefined features such as number of legs, existence of closed top and bottom,

existence of loop and zigzag stroke. Then the contour is extracted and classified into contour feature categories. The classified features will be put into 5x3-matrix and the matrix is classified to character classes by a set of rules.

In 2006, Limkonglap improved Mitrpanont's approach by using contours fitting of the extracted contour to quadratic or Gaussian function and classify into feature categories. This method helped feature categorization to be more accurate (Limkonglap, 2006).

In 2009, Imprasert proposed the improvement of Mitrpanont's approach as well by adding the set of rules to the feature conflict resolution and applying artificial neural network to the zigzag stroke extraction and the recognition process (Imprasert, 2009).

Theeramunkong proposed Island-based multidirection projection technique with Hidden Markov Model (HMM) in 2005 (Theeramunkong & Wongtapan, 2005). Island means the connected component of foreground pixels when the image is sliced in a specified direction. The method starts by slicing the image vertically, horizontally and diagonally. Each slice is segmented into six zones. The number of islands is counted in each zone and becomes sequential feature vector to train hidden markov model for each character.

Another HMM-related method proposed by Roongroj in 2003 is the Block-based PCA (Nopsuwanchai, 2003). The block-based PCA is obtained by segmenting the vertical and horizontal slices into blocks and take PCA to reduce the feature dimensions in each block. The HMM classifier is trained by maximum mutual information scheme. The result shows a significant improvement in accuracy by using maximum likelihood training.

The disadvantage of the approaches that uses classical feature, as in Airphaiboon, Phokharatkul, Sankhuangaw, Methasate, Mitrpanont, Limkonglap and Imprasert's work, is that they are sensitive to the absence of features. For example, character heads are often not closed or are written as blob, which is the cause of feature missing, and results in misrecognition (Saetang, 2011).

Using contour as in Mitrpanont, Limkonglap and Imprasert's work has a limitation because contour can be highly deformed from thickness and unwanted intersection. Moreover, these approaches rely on rules in the feature extraction process

that have to be constructed specifically on the Thai character shape. It is difficult to apply this technique to other languages.

Counting number of islands as in Theeramunkong's work is sensitive to unwanted intersection and degenerated loop feature. For example, filled-head stroke makes the number of island decreased from 2 to 1.

There is a limitation of feature that extracted based on sequence of sliced row and column as in Theeramunkong and Roongroj's work. The problematic situation occurs when an unseen input has the visual feature that is not aligned in the same way as in the training pattern. It can produce the unseen sequence of feature and affects the classification.

To design the methodology that is able to accurately recognize these characters, we should be aware of the following issues:

- Features should be transformation and morphological invariant, which include scale, translation, slant and thickness. This requirement is crucial for general handwritten OCR.
- Features should tolerate deformation, which allows the feature to exclude any insignificant change in curve shape.
- The connectedness of curve should be ignored in the feature extraction process. This requirement is to avoid error from the abnormal number of components. Moreover, it allows fanciful font style such as dash line or poor scanned document.
- Features should be independent of the fixed set of discriminative features, means that if the predefined features are missing, some remaining features can be used as discriminative features. This requirement makes generality to the system for it does not rely on specific shape, number of visual features, or the existence of features that is possibly missing.

In this research, we propose the novel feature that is free from the pre-defined visual feature, unaffected by curve intersection, and able to handle the variations of handwritten characters.

2.5 Classification

2.5.1 Support Vector Machine

Support Vector Machine (SVM) is a learning algorithm invented by Cortes and Vapnik in 1995 for binary classification (Scholkopf, 2002; Shih, 2010). It respects feature vector as a point in multi-dimensional space. The vectors that belong to the same class should be closed together, and far away from different classes. If so, a separator can be drawn between two classes.

The important concept of SVM is to build a hyperplane (generalization of straight line or plane in 3-dimension) to separate classes. The plane will separate the whole space as well. Then the unseen vector fall into one side of the plane, it will be classified. Hyperplane is defined by a normal vector w and a translation constant b.

$$f(x) = wx + b = k$$

If a point x is on the plane, then $f(x) = 0$. For other points, we can determine that x is on which side of the plane by checking the sign of $f(x)$. The parameter w and b must be well chosen for the best classification power. w is the direction of the normal vector of the hyperplane that maximize the margin, that is, the plane should be as far as possible from the boundary of each class (that is why the method is called support vector).

A vector in the training data $\{x_1, x_2, x_3, \dots, x_n\}$ must belongs to one of the two classes, they are marked by y_i where $y_i \in \{-1,1\}$. If the training data can be linearly separated, vectors in the set must satisfy:

$$w \cdot x_i + b \geq 1 \text{ if } y_i = 1$$
$$w \cdot x_1 + b \leq 1 \text{ if } y_i = -1 \quad (2.1)$$

Or it can be written as $y_i(w \cdot x_i + b) \geq 1$ for all x_i

Since the distance between a vector and the hyperplane is $\frac{|w.x_i+b|}{\|w\|}$, the closest vector is $\frac{1}{\|w\|}$ unit away from the hyperplane and $\frac{2}{\|w\|}$ unit from the other class. The optical hyperplane can be obtained by solving the optimization problem:

Minimize: $\frac{1}{2} w \cdot w$

Subject to: $y_i(w \cdot x_1 + b) \geq 1$ for all x_i

This problem can be solved by finding the saddle point of the Lagrange function

$$L(x, b, \alpha) = \frac{1}{2} w.w - \sum_{i=1}^{K} \alpha_i (y_i (w \cdot x_i + b) - 1) \quad (2.2)$$

where $\alpha_i > 0$ are the Lagrange multipliers.

Then the optimal hyperplane will be determined by

$$w_0 = \sum_{i=1}^{K} \alpha_i y_i x_i$$
$$b_0 = y_i - w_0 \cdot x_i \quad (2.3)$$

An unknown can be classified by calculating $f(x) = \text{sign}(w_0 \cdot x + b_0)$

2.5.2 Artificial Neural Networks

Artificial neural network (ANN) is computational technique inspired by the process of biological neurons in human brain to learn a specific task. Neuroscientist proposed that brain operates by transmitting the input signal via a set of connected neuron. The signal is processed through the transmission in every single neuron by taking weighted sum of the multiple inputs, applying a certain multivariate function, and obtaining the output signal for other neurons. Artificial neural network inherit the similar neurotransmission functioning by using the artificial neuron, which is an abstract object containing multiple input nodes, an output node, and an activating function. Output node of a neuron can be connected to the input nodes of other neurons. The connections allow the output signal to be transmitted to another neuron with weight that controls the strength of the signal. A certain neuron calculates the weighted sum of the signal and applies the activating function in order to get the output signal. Neural network learns by adjusting the set of weights in the connections (Zurada, 1992).

The traditional structure of the neuron network is known as the feed-forward neural network as shown is Figure 2.4. Feed-forward neural network can be created by constructing a bundle of neurons in the same level called layer. Each neuron in a layer is connected to every output of the neuron in the previous layer. The input of the first layer is called the input layer and the output of the last layer is called the output layer. Signal is processed through the input layer and is transmitted layer-by-layer to the output layer.

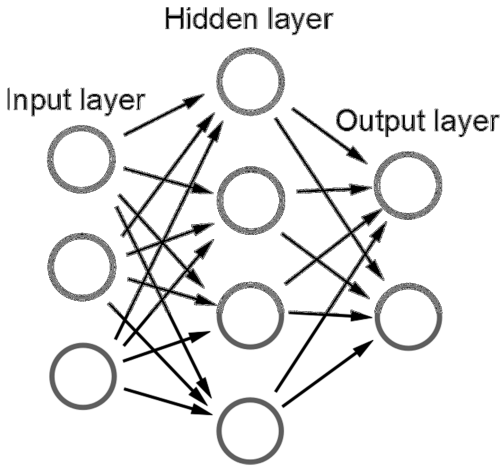


Figure 2.4 Structure of a multilayer feed-forward neural network.

Before the constructed neural network can operate a specific task, it must be trained. The traditional training scheme is known as the back-propagation training. Neural network itself is just a connection of abstract nodes between layers. The key concept that makes neural network works is the ability to learn the relation between the input and output. Neural network learns when its weight of the connection is changed. Since every single connection between nodes influence the output. There must be a good strategy for weight changing that leads to the desired input-output relation in the shortest training time.

The purpose of back propagation is to change the set of weights in order to minimize the difference between the actual output and the ideal output. The output difference is referred to as the error. To accomplish this purpose, the back propagation algorithm calculates the partial derivative of error with respect to each weight connection in every layer. The partial derivative conveys the clue to update the weight. Positive derivative means the error increases as the weight increase so the weight must be reduced in order to lower the error, and vice versa.

The error E, the value that we want to minimize, is equal to $E = \frac{1}{2}(t - y)^2$, where t is the ideal output of the training data and y is the actual output of the neural network at a certain epoch of training.

The output of the j^{th} neuron $\mathrm{o_j}$ is the weighted sum of the output of the previous layer $\mathrm{s_j}$ applied to the activation function ϕ.

$$o_j = \phi(s_j) = \phi\left(\sum_{k=1}^{n} w_{kj}\, x_k\right) \tag{2.4}$$

In general, ϕ is a non-linear, differentiable function. The commonly used ϕ is the logistic function defined by $\phi(z) = \frac{1}{1+e^{-z}}$. The number of the input to the neuron is n. The variable $w_{ij}$is the weight connecting between neuron i and j.

Back propagation is the process to reduce error E by changing the value of weight w_{ij}. To justify how each weight must be changed in order to reduce E, the partial derivative $\frac{\partial E}{\partial w_{ij}}$ is evaluated. By using chain rule twice, we get

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial s_j} \cdot \frac{\partial s_j}{\partial w_{ij}}$$

Each factor of the $\frac{\partial E}{\partial w_{ij}}$ above can be derived as follows.

$$\frac{\partial s_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left(\sum_{k=1}^{n} w_{kj}\, x_k \right) = x_i$$

$$\frac{\partial o_j}{\partial s_j} = \frac{\partial}{\partial s_j} \phi(s_j) = \phi(s_j)(1 - \phi(s_j))$$

$$\frac{\partial E}{\partial o_j} = \frac{\partial E}{\partial y} = \frac{\partial}{\partial y} \frac{1}{2} (t - y)^2 = y - t$$

The calculation above is applied to the output neuron in the final layer. For the inner layers, consider the error E is the function of the inputs of all neurons $L = u, v, \dots, w$ that receive input from neuron j.

$$\frac{\partial E(o_j)}{\partial o_j} = \frac{\partial E(s_u, s_v, \dots, s_w)}{\partial o_j}$$

$$= \sum_{l \in L} \left(\frac{\partial E}{\partial s_l} \frac{\partial s_l}{\partial o_j} \right) = \sum_{l \in L} \left(\frac{\partial E}{\partial o_l} \frac{\partial o_l}{\partial s_j}\, w_{jl} \right)$$

The derivation shows the recursion of $\frac{\partial E}{\partial o_j}$ that needs the such a derivative from the following layer. Therefore the calculation of backpropagation process starts at the output layer and trace back to the previous layers. By putting all derivation together, we get,

$$\frac{\partial E}{\partial w_{ij}} = \delta_j x_i \tag{2.5}$$

with

$$\delta_j = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial s_j} = \begin{cases} (o_j - t)\phi(s_j)\left(1 - \phi(s_j)\right) & \text{if } j \text{ is an output neuron} \\ \left(\sum_{l \in L} \delta_l w_{jl}\right)\phi(s_j)\left(1 - \phi(s_j)\right) & \text{if } j \text{ is an inner neuron} \end{cases} \quad (2.6)$$

Updating neuron can be performed by using gradient descent with the predefined value of learning rate α.

$$\Delta w_{ij} = -\alpha \frac{\partial E}{\partial w_{ij}} \quad (2.7)$$

A number of Thai OCR systems using artificial neural network has been proposed. Tanprasert and Koanantakool proposed Thai OCR system using artificial neural network. An input image is denoised, segmented, and feed to the normalization process that maps character pixels to a fixed dimensional matrix. Binary value of each matrix element will enter the multi-layered perception neural network with back-propagation learning algorithm. The experiment shows that the 8x8 matrix give the best result among all cases (8x8, 8x12, 8x16 and 8x23) (Tanprasert, 1996). Methasate and Sae-tang used confusion matrix to analyze the recognized result from neural network model (Methasate I. S.-t., 2004). The confusion matrix stores the frequency that each training character is recognized correctly or misread into another character. We can define coarse classification groups based on the frequently wrong recognition stored in the confusion matrix. P. Phokharatkul and C. Kimpan did the coarse classification of character groups by using number of heads, which is the number of inner contours, then feed the fourier descriptor of the outer contour into the Genetic Algorithm based Neural Network (GA-NN). Genetic algorithm is general purpose search technique, emulated from evolutionary process in living things via genetic substance. GA-NN makes use of the genetic algorithm to search for the most effective parameters (weight and bias) to be used in the neural network (Phokharatkul P. K., 2002).

2.5.3 Convolutional Neural Networks

Convolutional neural network is a state-of-the-art implementation of artificial neural network that gives a remarkable result on handwritten character recognition (O'Neill, 2006; Soman, Nandigam, & Chakravarthy, 2013; Graham, 2014; Yuan, Bai, Jiao, & Liu, 2012). As shown in Figure 2.5, CNN is a 5-layer, not-fully-

connected neural network. The networks receive the image pixel directly without extracting any feature. It is designed to learn feature by itself. Layers are connected by mapping sliding window kernel to each node in the next layer, called feature mapping. The weight of feature mapping will be learnt automatically without using any handcraft feature.

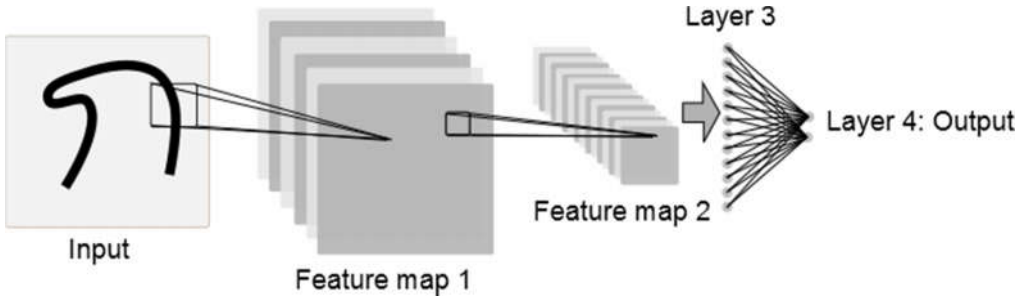


Figure 2.5 Structure of Convolutional neural networks

2.5.4 Hidden Markov Model

Hidden markov model is a generative statistical model designed to estimate the probability of the given pattern. The term "generative" refers to the concept that a particular pattern is a special case of the general model that create it. The goal of the estimation is to find the best model among the set of models that is likely to generate the given pattern. In character recognition, the goal is to find the most appropriate character class that is likely to generate the given character image.

Markov model contains states that interdependent in the sequential form. The characteristic of markov process is that the probability of a certain state in the sequence depends only on the previous state. This characteristic allows us to create a graph representing a markov process where each edge connected between two nodes is labeled by a constant probability where sum of the outgoing probability from every node sum up to one.

The term "hidden" refers to the invisibility in the process. In hidden markov model, the set of state is the set of abstract representation. Each hidden state has the "output probabilities" to every visible state in the given pattern. There are also the "transition probabilities" between every pair of hidden states. The main objective of HMM is to find out which sequence of the hidden states is most likely to generate

the given visible pattern. These hidden states are hidden in sense that one cannot know the exact sequence of these state that corresponds to the given visible pattern.

Probability of each link in the model is the parameter that needs to be evaluated before the model can be used. Finding these parameters can be done in the training process, which can be achieved by statistical parameter approximation from the collected data (Fink, 2008).

2.6 Other relevant issues

2.6.1 Morphology

As we referred to in the steps of OCR, thinning algorithm is one of the traditional process in pre-processing step.

The theoretical version of the thinning algorithm is called the Mathematical Morphology, which is the analysis of signal in term of shape (Jang, 1990; Shih, 2010). Written character and thinning algorithm can be described by two morphological functions called dilation and erosion, defined under set theory.

Considering an image as a set of coordinates in a plane, there are two sets in concern: one of them (A) is the image being morphed, and another (B) is called the structuring element, which is a modifier of A. Dilation and erosion of A by B are defined by

$$A + B = \{x | B_x \cap A \neq \phi\} = \bigcup_{x \in B} A_x = \{a + b | a \in A, b \in B\}$$
$$A - B = \{x | B_x \subset A\} = \bigcap_{x \in B} A_x = (A^c + B)^c \quad (2.8)$$

Intuitively, the idea of dilation of A by B is that we have a brush with the cross section image B centered at x. As x is moved over A, the brush trace expands the region at the border of A.

The erosion is the inverse operation of the dilation, A can be eroded by B by tracing only the point x while moving the brush tip of B inside A. Both the dilation and erosion result of A depends on the size and figure of B.

2.6.2 Connectivity

Connected component is simplest way to define character segmentation. Most of non-cursive script character can be separated by using space (background pixels) between characters. Moreover, connectivity is also used in feature extractions that rely on counting pixel components. The way to define connectivity is adapted from topology and graph theory.

In topology, a connected set is defined by the set that cannot be partitioned into two nonempty subsets which are open (McCleary, 2006). This concept gives the idea of connectivity to be the object that cannot be contained into two disjoint boxes. The definition is suitable for continuous space. In the discrete space like image pixels in plane, the more suitable definition is inherited from the connected graph in graph theory. A connected graph is the graph that we can find a path to connect any pair of nodes in that graph (W.D., 2007).

Discrete points connectivity is defined in a neighborhood and then uses the transitive property to link the connectivity in long distance. There are two types of point connectivity; 4-connect for background points and 8-connect for foreground points. A point in the neighborhood of a given point (center), indexed by the number 0,1,…,7 around the point x, is always said to be 8-connected to x, while it is said to be 4-connected to x if it is in one of the position 0, 2, 4 or 6 (Shih, 2010).

From the definition, it implies that foreground pixels can be connected via 45 degree direction (Figure 2. (b) and (c)), but the same pattern for background is said to be isolated background component (Figure 2. (e)). A hole (isolated background) can be surrounded by only four foreground pixels, but an isolated foreground must be surrounded by all eight pixels.

2.6.3 Hough transform

Hough transform in the conversion between the set of points in a pattern and the parametric description of the patterns (Duda, 1972). For example, a straight line can be described by a set of points lying on the same row, or it can be described another way as a pair of slope-intercept as in the equation $y = mx + c$ or in the distance-angle form $r = x\cos\theta + y\sin\theta$. A circle can be described as a 3-tuple

parameter: two components of center and one radius. Hough transform allow us to describe patterns in sense of their intuitive properties instead of pointwise method.

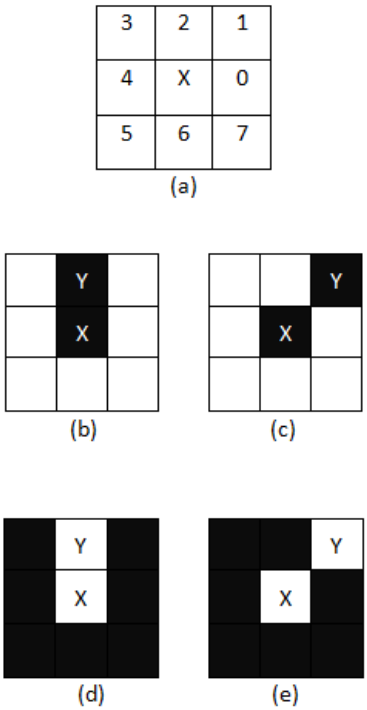


Figure 2.6 (a) indexing of neighborhood, (b) – (c) X and Y are 8-connected, (d) X and Y are 4-connected, (e) X and Y are not 4-connected.

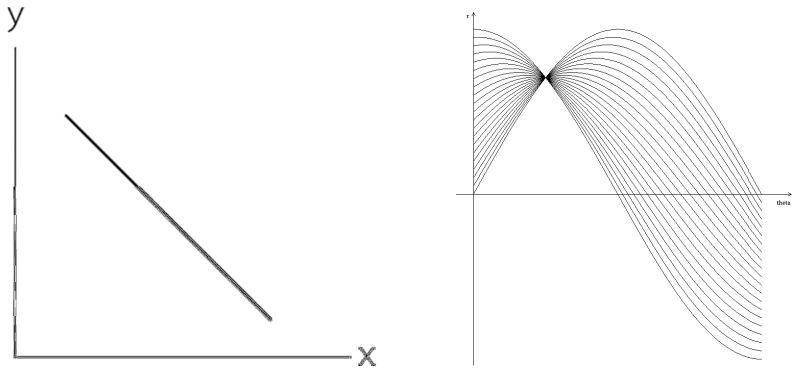


Figure 2.7 A line segment on x-y plane and its hough transformation into $r - \theta$ plane. The right hand side shows the intersection of transformed graph, which is where the line is located.

Detection of pattern in an image by using Hough transforms algorithm can be done by defining the parameter space with the dimension equals to the unknown

parameter, calculating all possible parameter points in the space for each image pixel in a neighborhood. Then vote for the maximum accumulation point in the parameter space and determine the parameter of the pattern.

2.6.4 Principal component analysis

Principal component analysis (PCA) is a process to transform a set of possibly correlated multidimensional vector into another orthogonal uncorrelated space. The term "principal component" conceptually refers to the "main axis" that lies across the highest variant direction of data points in the hyperspace. PCA manages to project the original vector into the new space with lower dimension of uncorrelated variables where the magnitude of variance of the data is sorted decreasingly.

PCA comprises the analysis and the transformation. Start with a set of N variable vectors, the first process is to normalize the data by subtracting the mean of each variable out of the original data. The purpose of mean extraction is to translate the data points to be centered at the origin.

After the normalized vector is obtained, then the next process is to calculate the covariant between every two variables by using the following equation.

$$cov(X,Y) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{(n-1)} \tag{2.9}$$

Since covariant describes the relationship between two variables at a time, the relationship between every pair of variables can be put into the covariant matrix where each entity of the matrix is a covariant corresponding to the variable at row i and column j.

Eigenvectors and eigenvalues of a square matrix A are the set of corresponding vector v and scalar λ that satisfy

$$Av = \lambda v \tag{2.10}$$

In the PCA calculation process, the eigenvectors and eigenvalues of the covariant matrix are then extracted and sorted. Each of the eigenvector represents a new axis passing through the data. The eigenvalue corresponding to the eigenvector indicates the distribution of data along the eigenvector direction. Any two eigenvectors are perpendicular to each other.

Sorting eigenvectors by their eigenvalues decreasingly is the way to arrange the significance of the axes. The highest variance axis is brought to the first axis in the transformed matrix.

Scalar product between the original data to each eigenvector gives the projection of the data along the chosen axis. Thus the projection to every axis returns the transformation of the data point into the new set of axes.

After the transformation, the first component of each data point is the value that best describe the characteristic of the data. Higher index components will be the lesser well-describer of the data. The component at the very end of the transformed data can be truncated with no significant effect to the overall data. This truncation makes the lossy data compression, which is one of the main purpose of principal component analysis.

2.6.5 Bag of Words

Bag of word (BoW) is a feature extraction technique that applied to text, image and video object for the classification purpose. The essential of BoW is based on the creation of abstract bag, putting elements into bags, and counting objects inside the bags. Each bag in BoW is defined to be a filter that determines whether a certain element can be in the bag or not, together with a counter that counts the number of element already in the bag. Element can be either linguistic word or visual word. BoW returns the vector of relative frequency that each word occurs in the object. For example, in text classification (Alahmadi, Joorabchi, & Mahdi, 2013) (such as spam message classification), the element is each particular word and bag is defined be a container labeled by each particular word. Since the relative frequency of each word derived from spam message and non-spam are different, so the classifier is capable to discriminate spam message out of normal message using feature from BoW.

Bag of word is also applied in image and video classification task, known as Bag of Visual Word (Xiong-wei, De-cai, Lu-ming, & Ai-jun, 2014; Wang & Huang, 2014; Li & Feng, 2013). The term visual refers to local feature or image descriptor such as SIFT (Scale-Invarient Feature Transform). BoVW perform the same process as BoW, i.e., to count the number of visual word in the video or image in order to get the relative frequency of each distinct visual word.

One of the most important characteristic of BoW, both in linguistic word and visual word, is that the algorithm omits the ordering of elements. The feature extraction concerns only the existence of word but not the semantic nor its location and context. This characteristic makes BoW works efficiently for offline data such as image.

CHAPTER III
METHODOLOGY

In this chapter, we present the constructional approach to create a novel feature named direction histogram and the feature extraction algorithm known as bag of histogram. At the very first of the chapter is located to introduction of direction histogram and its feasibility. Next, the algorithm of bag of histogram is proposed. We then also explore the parameters and properties of bag of histogram at the end of this chapter.

3.1 Direction Histogram

3.1.1 Inspiration

In the unconstrained handwritten character domain, an unlimited variation of writing style is possible happened. The feasible image feature for character recognition is supposed to capture the underlying structures behind the variation while discriminating the character from different classes. One of the feature is the relative position between every foreground pixel. From our observation through a number of handwritings, the relative position within the set of foreground pixel is roughly preserved. We pay attention about the position of each pixel as it is surrounded by other pixel.

3.1.2 Definition

We propose a feature named **direction histogram** (DH). Direction histogram is a global feature describing the distribution of pixel in the entire image around each pixel.

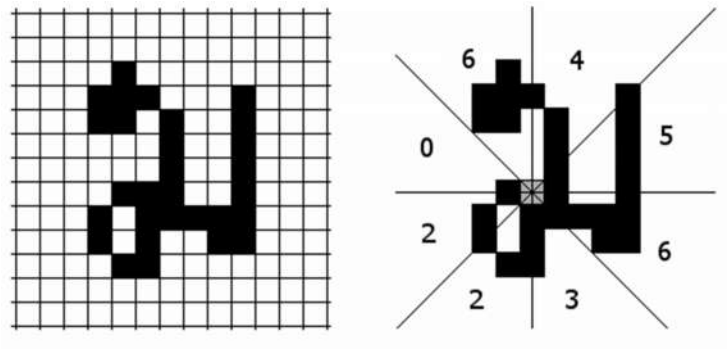


Figure 3.1 Image pixels and selected center for DH calculation

Consider an image as a collection of discrete pixels (Figure 3.1 left). If we select a particular pixel as a center (grey pixel on Figure 3.1 right), the remaining pixels are distributed around the center in the different directions. The number of pixels in each direction is counted, regardless of the distance, to get the discrete distribution of point in each direction. The normalization of the distribution is called "Direction Histogram" (DH). For example, the pixel counting around the center in the Figure 3.1 returns an 8-component vector $< 5, 4, 6, 0, 2, 2, 3, 6 >$ (started from positive x-axis and moved counterclockwise). The normalization can be done by dividing the sum of every component (5+4+6+0+2+2+3+6 = 28), which results the direction histogram equals to $< 0.179, 0.143, 0.214, 0, 0.071, 0.071, 0.107, 0.214 >$.

The formal definition of DH is described as follows. Given image *I* containing m pixels, each pixel has its own DH, so there are m direction histograms from this image. Suppose we segment the round angle into *N* direction intervals. For *j* runs from 1 to N, each direction interval is defined by semi-closed interval $D_j = [s_j, t_j)$ where $s_j = \frac{2(j-1)\pi}{N}$ and $t_j = \frac{2j\pi}{N}$

Direction histogram $DH_i[j]$ is a vector containing pixel density from i^{th} center in the direction interval j^{th}.

$$\theta_{ik} = arctan\left(\frac{y_k - y_i}{x_k - x_i}\right) \tag{3.1}$$

$$DH_i[j] = \frac{\sum_{k=1}^{m} f(\theta_{ik}, D_j)}{m} \text{ where } f(\theta_{ik}, D_j) = \begin{cases} 1 & if\ \theta \in D_j \\ 0 & otherwise \end{cases} \tag{3.2}$$

Distance between two of direction histogram $d(DH_1, DH_2)$ can be measured by using Euclidean distance.

$$d(DH_1, DH_2) = \sqrt{\sum_{j=1}^{N} (DH_1[j] - DH_2[j])^2} \quad (3.3)$$

Since various patterns of the same character share the same DH set, DH will be used in constructing our feature extraction method named “bag of histogram”. Direction histogram will be extracted and converted into image feature by using the bag of histogram algorithm presented in the next section.

3.1.3 DH on Continuous space

Other than the discrete space viewpoint, DH can be defined on a continuous plane. The concept of direction interval and direction between points are the same but the notation of image and $DH_i[j]$ is changed to be a double integral on plane. The definition of DH on continuous space will be useful for the proof of some statements of DH properties later.

In continuous space, image is defined as a bounded, finitely many connected component, open subset of the plane. The reason we define it to be open is to prevent the case that any part of the image is a thin line with zero stroke width so that its integral equals to zero.

Let S be the set of coordinate $(x, y) \in R^2$ representing an image. We define $DH_c[j]$ as the j^{th} component of the direction histogram centered at c as follows.

$$DH_C[j] = \frac{area\ of\ image\ in\ D_j}{area\ of\ the\ entire\ image} = \frac{\iint_{(x,y)\in S} f_j(x, y, c)dxdy}{\iint_{(x,y)\in S} dxdy}$$

$$\text{where } f_j(x, y, c) = \begin{cases} 1, & if\ arctan(\frac{y-y_c}{x-x_c}) \in D_j \\ 0, & otherwise \end{cases} \quad (3.4)$$

3.1.4 Visualization of DH

Figure 3.2 demonstrates three samples of character ค from different handwritings and a sample of character ค. Five points picking at different positions from each image at analogous position. At each point, DH of 40 bags corresponding to that point is calculated and presented. It can be interpret that DHs at analogous position among on the same character from different handwriting are also similar.

Visually, the surrounding pixels at analogous position are distributed in more or less the same way, therefore its DHs displays the similar pattern.

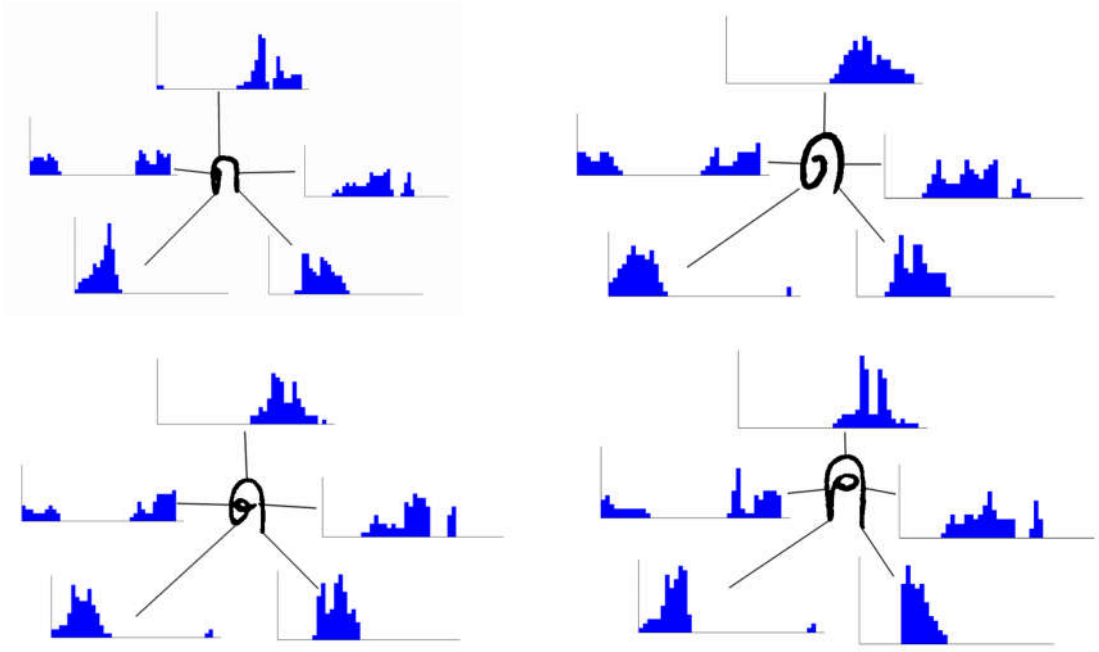


Figure 3.2 Direction histogram extracted at different position on different characters.

On the other hand, DHs extracted from ค, the character with similar shape to ด, represent the similar graphs of DH as well. The only different between ด and ค is the head orientation, i.e., head of ด is drawn clockwise and that of ค is drawn counterclockwise. However, there is a discriminative DH at the lower right of the figures that represent different shape of DHs between that of ด and ค. The lower right DH of ด consists of two lobes where that of ค has only on lobe. Discrimination like this can be found at every pair of characters that differ in their shapes.

3.2 Some proofs

In this section, we are to show some properties of direction histogram, i.e., translation invariance, scale invariance, uniqueness, and continuity.

3.2.1 Translation invariant

Let S be an image, c be a point on S, and p be an arbitrary point in the same space. To show that DH is translation invariant, we need to show that the direction histogram of S at c is equal to the direction histogram of $S - p$ at $c - p$.

$$DH_c[j] = \frac{\iint_{(x,y)\in S} f_j(x,y,c)dxdy}{\iint_{(x,y)\in S} dxdy}$$

$$\text{where } f_j(x,y,c) = \begin{cases} 1, & if\ arctan(\frac{y-y_c}{x-x_c}) \in D_j \\ 0, & otherwise \end{cases} \quad (3.5)$$

From the definition of DH in continuous space above, we exploit the fact that translating image on 2D Euclidean plane does not change its area. If we move the center to calculate DH sector of image $S - p$ centered at $c - p$, it equals to the area of the corresponding sector in the original image, i.e.,

$$\iint_{(x,y)\in S-p} f_j(x,y,c-p)dxdy = \iint_{(x,y)\in S} f_j(x,y,c)dxdy \quad (3.6)$$

Since the equality of sector area implies the equality of the entire image, then DH component, which its value is the fraction of sector area divided by the image area, is the same as that of the original image. It is proved that DH is translation invariant.

3.2.2 Scale invariance

A mathematical function is scale invariant if the values of the function when applied before scaling and after scaling are identical. Let S be an image and r be an arbitrary positive scalar. To show that DH is scale invariant, we need to show that DH centered at c of S ($DH_c(S)$) has the same value as DH centered at rc of the image rS ($DH_{rc}(rS)$).

The statement can be shown easily by using the fact that scaling the image by the factor r get the image area scaled by r^2. The area scaling factor r^2 is applied the same way on both DH sector and the entire image. Therefore

$$DH_{rc}(rS)[j] = \frac{area\ of\ rS\ in\ D_j}{area\ of\ the\ rS} = \frac{r^2(area\ of\ S\ in\ D_j)}{r^2(area\ of\ S)} = \frac{area\ of\ S\ in\ D_j}{area\ of\ S} = DH_c(S)[j] \quad (3.7)$$

The area factor r^2 cancelled and the original ratio remains, which is equal to the original DH value.

3.2.3 Uniqueness

Definition: Equality of DH

Two direction histograms are equal if and only if every component in the DHs are exactly matched component-by-component.

$$DH_1 = DH_2 \Leftrightarrow \forall i \in \{1,2,3,\dots,\mathrm{DH}_{size}\}[DH_1[i] = DH_2[i]] \quad (3.8)$$

Corollary: Two DHs are equal if the arbitrary sum of components from $DH_1$and DH_2 selected from the same set of index are equal.

$$DH_1 = DH_2 \Leftrightarrow \forall S \subset I\ [\textstyle\sum_{i\in S} DH_1[i] = \sum_{i\in S} DH_2[i]] \quad (3.9)$$

Proof

($\Rightarrow$) Since $DH_1 = DH_2$, $DH_1$and DH_2 values are equal componentwise. It obviously implies any sum of subcomponents of DH_1 and DH_2 to be equal.

($\Leftarrow$) Since sum of any subset between $DH_1$and DH_2 are equal, the sum of each single component will be equal as well. It implies DH equality directly from definition.

Statement: Uniqueness of DH

Let S be a connected thick image. c_1 and c_2 are distinct points on S. The vector from c_1 to c_2 must be on exactly one sector because the sectors of any DH are mutually exclusive. Moving the center from c_1 to c_2 also moves the sector from v to v^*. The openness of S claims that there is an open disk with positive radius, which has positive area that is contained in the difference $v - v^*$. At this step, it implies that the value of DH in sector v is greater than that of v^* because v^* is a proper subset of v and there is positive area of S in v that is not in v^*. There are at least one component of DH_1 that has different value to DH_2. From the contraposition of the corollary above, it implies that $DH_1 \neq DH_2$ if $c_1 \neq c_2$.

3.2.4 Continuity

We are to show that the value of DH changes continuously when the center moves. Let S be a thick image in plane bounded by a constant M, the continuity of DH means:

$$\forall p_0 \in S, \forall \epsilon > 0, \exists \delta > 0 \; such \; that \; \forall p \; with \; \|p - p_0\| < \delta \rightarrow \left\|DH_p - DH_{p_0}\right\| < \epsilon$$

In other word, DH is continuous if for an arbitrary constant, there is a radius of center movement that moving the new center inside this radius will change the DH value less than the constant we setup.

To show that the continuity of DH holds, we start by picking an arbitrary $p_0 \in S$ and an arbitrary $\epsilon > 0$. Then we choose $\delta > 0$ as a function of ϵ, D (dimension of DH) and M (bound constant of the image). After that, we show that $\|p - p_0\| < \delta$ implies $\left\|DH_p - DH_{p_0}\right\| < \epsilon$.

We construct an illustration to demonstrate the proof. There is an image S with two centers p_0 and p. Suppose $D = 8$ for the sake of less complexity. Drawing everything together looks like in Figure 3.3Figure.

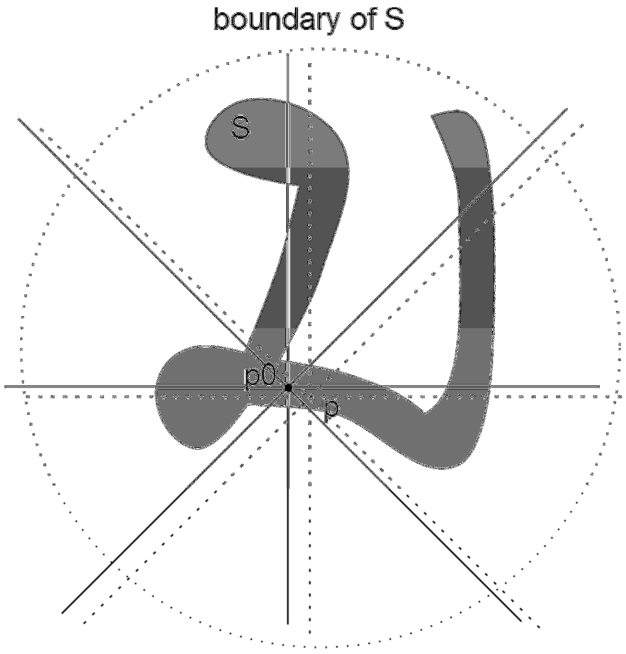


Figure 3.3 An image with two centers and their sector boundaries.

To prove the existence of δ, we choose $\delta < \frac{\epsilon\ A}{M\ D}$, where A is the area of the image, M is the bound of the image, and D is the dimension of the DH. Let $\|p - p_0\| < \delta$, we get $\|p - p_0\| < \frac{\epsilon\ A}{M\ D}$.

Next, we will show that, the difference between new DH_p and DH_{p_0} is bounded by ϵ. To achieve this goal, we define the "band" to be the area enclosed by

two parallel sector boundaries and the boundary of image S. The area of the band represent the "change" of the DH_p with respect to DH_{p_0}.

Since the band maximum length is M and the maximum width is $\|p - p_0\|$, which occurs when the vector $p - p_0$ is perpendicular to one of the sector separator, maximum area of the band is $M\|p - p_0\|$ and the change of DH value corresponding to this sector is bounded by $\frac{M\|p-p_0\|}{A}$ since DH value is area of foreground in each sector normalized by the image area. There are D sectors and each sector can change DH component less than $\frac{M\|p-p_0\|}{A}$, we get the absolute sum of DH change less than $\frac{MD\|p-p_0\|}{A} < \frac{MD\delta}{A}$. From our choice of $\delta < \frac{\epsilon\ A}{M\ D}$, it implies $\frac{MD\|p-p_0\|}{A} < \frac{MD\left(\frac{\epsilon A}{MD}\right)}{A} = \epsilon$.

Since we use Euclidean norm for $\|DH_p - DH_{p_0}\|$ and the Euclidean norm is less than or equal to L-1 norm, $\left\|DH_p - DH_{p_0}\right\| \leq \frac{MD\|p-p_0\|}{A} < \epsilon$ which is the result we want to prove. Therefore, the function DH applied on 2D Euclidean space is continuous.

3.3 Bag of histogram

In this section, "Bag of histogram", the main algorithm for our offline feature extraction is described. The fundamental concept of bag of histogram was inspired by bag of word (BoW).

Using bag is convenient to process offline data because feature extraction using bag of word need neither grammar rule, word order nor any temporal information in data. To take the advantages from bag of words, we design feature extraction algorithm for offline character recognition by using bag structure as in bag of words. Replacing "words" in the bags by our "direction histogram" feature, we name it "bag of histogram".

Bag of histogram is a collection of bags where each bag is labeled by a direction histogram. Bag of histogram is designed to represent the different zone of the character image. Character pixel zoning can be seen in Figure 3.6 (visualization of

BoH). After bags are created, feature vector will be calculated by counting the number of pixels in each zone.

Since direction histogram changes its value continuously when center moves, adjacent pixels in the same zone have similar direction histograms. We can count the number of pixel in each zone by alternatively counting the number of direction histogram that is "similar enough" to the label of the bag. The similarity of direction histogram is defined by the distance function in the previous section.

3.4 Feature extraction and bag of histogram

3.4.1 Bag preparation algorithm

In this process, a collection of empty bags is created. Each bag is labeled by a direction histogram to filter specific DH that can be put into the bag. The bag collection needs to cover every possible direction histogram in the recognition process. Therefore, the bag collection must be prepared from a sufficiently large training image set.

The algorithm for bag preparation is named "first come- first add". In this algorithm, a new bag will be created when a new DH (calculated from the training set) does not fit any existing bag. The algorithm starts by calculating an arbitrary DH from the training set. The first DH always creates the first bag because there is no bag at the beginning. Then the next DH is iteratively calculated. The new DH get distance measured to each existing bag label. If the distance is greater than a predefined threshold (DH different threshold: T), a new bag will be created and labeled by the new DH. The process iterates over every center. The "first come- first add" algorithm for bag preparation is presented in Figure 3.4.

```
Get the list of foreground pixels
Choose set of centers {c_i}
Set parameter : DH different threshold
for( 1 ≤ i ≤ number of centers)
        Measure direction histogram dh_i centered at c_i
        If i = 1
        Then create the first bag b_1 = dh_1
        End if

        If ∀j, d(dh_i, b_j) > T
        Then create another bag b_{j+1} = dh_i
        End if
End for
//Get a collection of BoH
```

Figure 3.4 The "first come – first add" algorithm

3.4.2 Feature extraction algorithm

From the bag preparation process, each bag represents a certain group of direction histogram. If we count the number of direction histograms that fit into each bag, similar images are supposed to contain approximately the same proportion of direction histogram in each bag. Therefore, the proportion of direction histogram in each bag is the feature to be extracted. Each bag becomes a component in the feature vector, so the feature vector dimension is equal to the size of the bag collection.

To extract feature of an image, each DH from the image gets the distance measured to each bag in the bag collection. The feature vector get score count +1 for the bag which the distance is less than the DH different threshold. The process iterates until every DH is put into the appropriate bag. The normalization of score counting in each bag will be returned as the feature vector of the image. The pseudocode of the feature vector extraction algorithm is shown in Figure 3.5.

```
For each training image
        Get the list of foreground pixels
        Choose set of centers {c_i}
        For each center in {c_i}
                Calculate dh_i around the center c_i
                For each bag b_j
                        If d(dh_i, b_j) < T
                        Then, b_j.count+= 1
                        End if
                End for
        End for
        Normalize the counting to be the feature vector
        Return feature vector
End for
```

Figure 3.5 The feature vector extraction algorithm

3.4.3 Visualization of BoH

The pixel zone corresponding to the extracted feature vector can be visualized by measuring the difference between DH of every pixel against the DH of the selected bag center. Pixels that have DH difference from the bag center less than threshold *T* are put into the same bag and we can see so in the visualization. Figure 3.6 shows the pixel zones corresponding to the bag of histograms in the various places.

Figure 3.6 Pixel zones corresponding to various BoHs

BoH is not just labelling the disk around the center. It is used to identify similar "point environment" among different image. Once a bag is tagged by a DH, the bag is shared through the image set so that every image has a possibility to possess a DH. It is expected that the variation of character shapes in the same class share DH from the same bag, and the different character classes use DH from different bags.

3.4.4 Mutual features and discriminative features

Though the feature extracted from bag of histogram is abstract, we can visualize and analyze their patterns so we can understand the feasibility that the classifier can discriminate different characters. The graph in Figure 3.7 displays the first 50 components of feature vectors of ก, ด, and ต, a set of similar characters. Generally, these three graphs look the same. These feature components are refered to as "mutual features".

On the other hand, feature vector component 208-211 of the same set of character ด ก and ต in Figure 3.8 displays the area of dissimilar components. We refer to these feature components as "discriminative features" because it is the key feature to discriminate the character in different classes.

Mutual feature and discriminative feature are relative to each pair of character images. Any feature can be mutual or discriminative, depend on which pair. Both mutual and discriminative are important. Mutual feature confirm the validity of

membership in a certain class, while discriminative feature help separating the identity between different classes.

3.5 Parameters of BoH

We can create different BoH by varying its parameters. In this study, the effect of the dimension of bag (D) and the DH different threshold (T) were explored. Each parameter leads to different characteristics of BoH. The effects and the appropriate values of the parameters are studied in our experiments.

3.5.1 Dimension of bag

Dimension of bag (D) is the number of direction intervals in each direction histogram. Since the width of direction interval is $\frac{360}{D}$. Dimension of bag has to be small enough to handle the shape variation. Meanwhile, the dimension has to be big enough to discriminate different character classes apart.

3.5.2 DH different threshold

DH different threshold (T) determines whether any two DHs are considered as different DH or not. Since DH is rational-valued vector, there are infinite amount of different DHs. We create bag of histogram to quantize the similar DHs into the set of finite bag. The similarity is determined by the threshold T. If the distance $d(DH_1, DH_2) < T$, DH_1 and DH_2 are said to be similar. The value of T controls the resolution of bag collection. Smaller T creates higher number of bags, which represent finer details of shape.

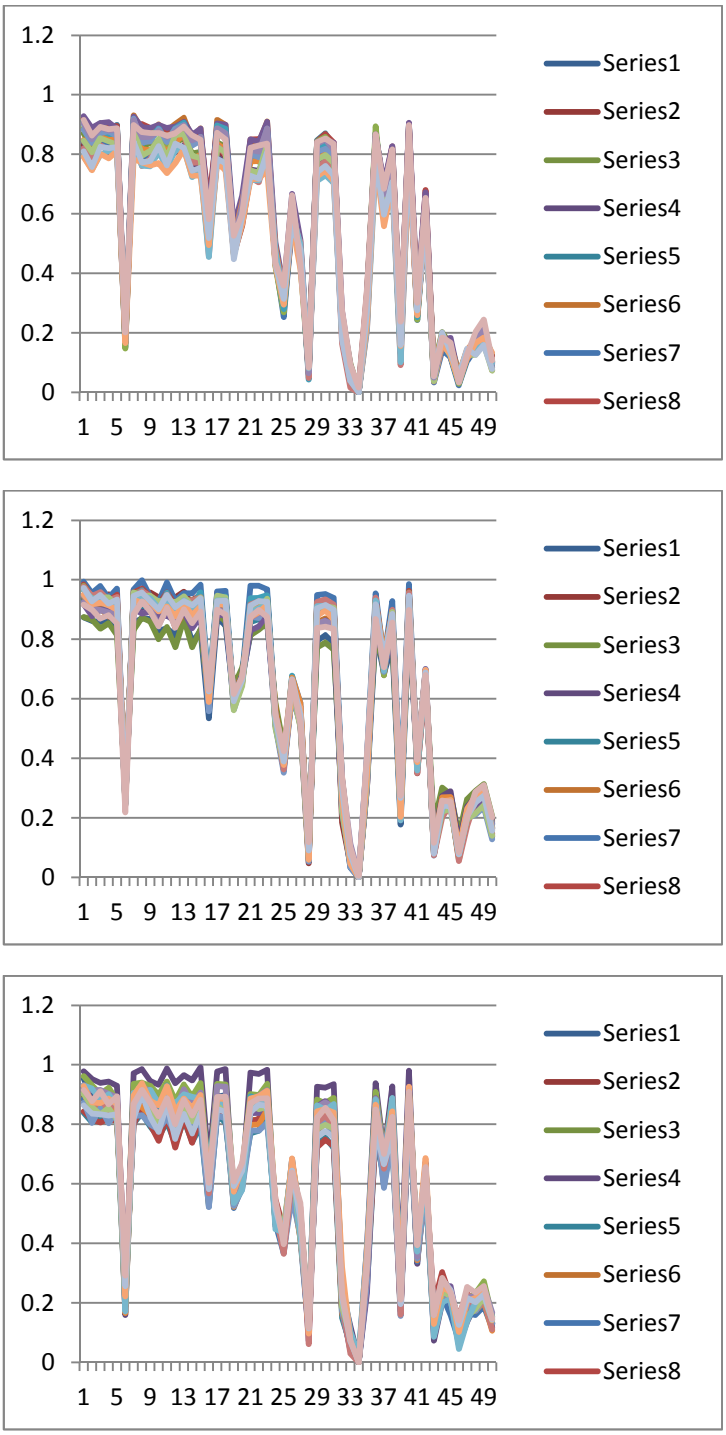


Figure 3.7 The first 50 components of feature vectors of ค, ด and ต respectively

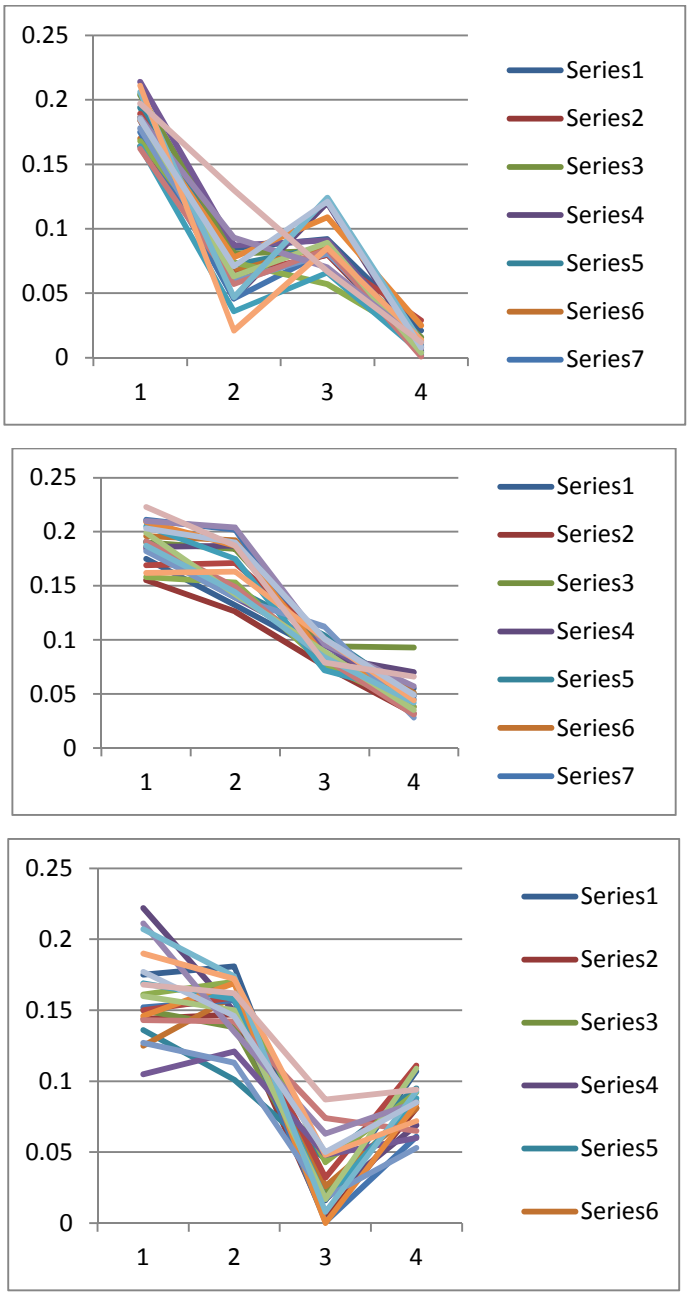


Figure 3.8 Component 208-211 of feature vectors of ค, ด and ต show the significantly different value between the three graphs

3.6 Properties of BoH

The benefits of using bag of histogram that inherited from bag of words are the simplicity of bag organization and the independence of feature order. Bag of histogram also give some additional benefits for character recognition, i.e., tolerate thickness, tolerate curve variation, ignore connectivity, and capture alternative feature as described below.

3.6.1 Tolerate thickness

Threshold T creates the region that similar DHs are put into the same bag. This region is as shown in the visualization of BoH.

Figure 3.9 shows the comparison of the sizes of region around some bag between thin and thick character image. The regions are about the same size, which cover about the same proportion of pixels in both image. Therefore the feature vector extracted using BoH in both image are about the same value. It means that the classifier trained by using BoH feature extracted from the thin images can recognize thick images as well.

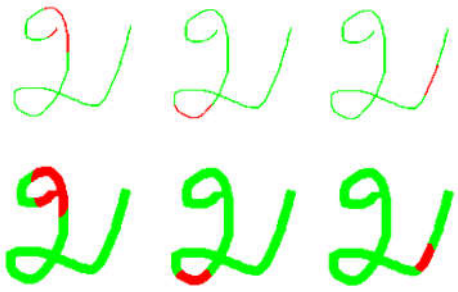


Figure 3.9 Comparison of region corresponding to BoH at different position in the thin and the thick characters

3.6.2 Tolerate curve variation

Image scaling does not affect direction between any two points. Since DH takes only direction in the calculation, the direction histogram is scale independent.

In DH calculation, each pixel is put into discrete direction interval. Discrete interval can tolerate pixel variations because the DH value still is the same as long as pixels in each direction interval move inside the interval. Straight lines

mapping between characters in Figure 3.10 indicates that the same DH can be found at the analogous position in the character images with various shape distortion.

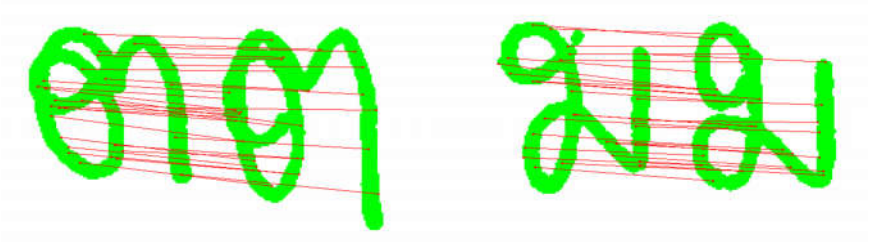


Figure 3.10 Mapping between pixels of two images that have the same value of DH

3.6.3 Ignore curve intersection

Feature constructed by BoH is not affected by the variation of unwanted curve intersection because the calculation of DH and BoH requires only direction between two pixels. The number of island or connected components is not involved in the feature calculation in the first place.

3.6.4 Capture alternative feature

For similar character classes, BoH brings out the difference in feature vector so that the classifier still is able to discriminate different character classes.

Figure3.11 indicates the pixel zones with same DH (green) and different DH (red) between each pair of similar characters. An interesting result is that the zones with different DH are not necessary to be located at the discriminative visual feature, but also be located around the visual feature in various directions. For example, although ฆ and ฅ are different in their head, different DH can be found both near the character head and somewhere else.

Figure 3.11 Green zone on each image represents pixels that have repeated DH in its pair. Red zone represents pixels that have different DH.

CHAPTER IV
OCR SYSTEM USING BAG OF HISTOGRAM FEATURE

To determine the potential of direction histogram and bag of histogram, we design an OCR system using bag of histogram. Our system comprises three modules, i.e., pre-processing, feature extraction, and classification as shown in Figure 4.1.

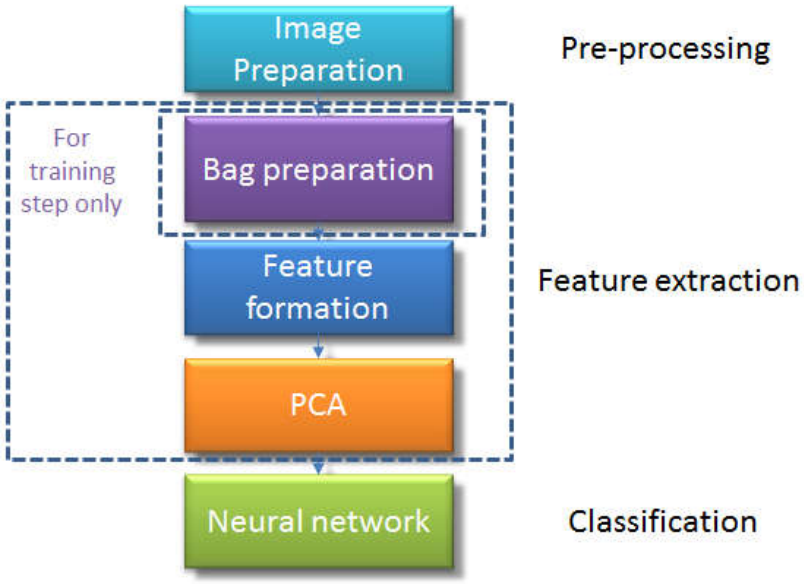


Figure 4.1 OCR system architecture

In training process, the set of training image enters the following steps, start with pre-processing, feature extraction, and classifier training. We will demonstrate the detailed process by using the example images in Figure 4.2 and Figure through every step.

Figure 4.2 Training set example

Figure 4.3 Test set example

4.1 Pre-processing

In pre-processing module, an input image is cropped out all white margins until foreground pixels reach the each foreground pixel on all four sides.

4.1.1 Binarization

Character image identity can be extracted by it shape, not its color and gradient. Therefore making binary image from color image reduces the complexity of the input. We use global binarization by taking min and max values of greyscale, and setting threshold at the middle of the range. Darker pixel will be black, brighter will be white.

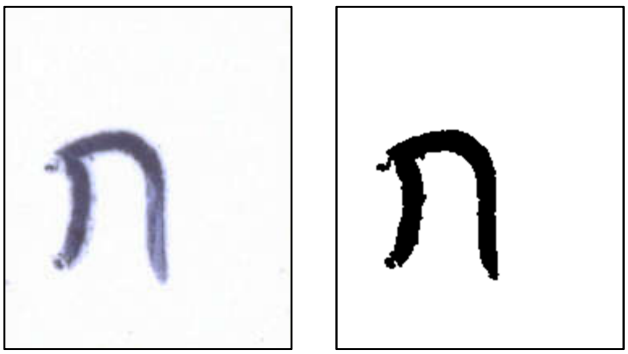


Figure 4.4 Left: original image, right: binarized image

4.1.2 Cropping

White margin does not affect the feature extraction by neither BoH nor in CNN OCR process. However, the dataset is prepared distribute to anyone who interested in handwritten character recognition in the future. The convenient format is the image with white margin cropped out until it reaches the foreground pixel on all four sides.

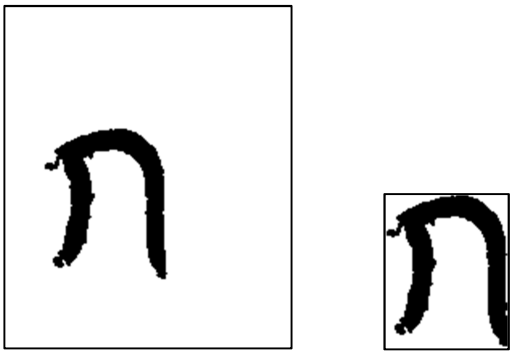


Figure 4.5 Left: binarized image, right: cropped image

The algorithm to crop white margin is simple. First, find horizontal and vertical projection. Then from the left most and right most, top most, and bottom most of each projection histogram, find the consecutive range of zero foregrounds, which locate the white margin of the image. Finally, image is cropped to the new frame.

4.2 Feature extraction

Feature extraction comprises three submodules: bag preparation, feature vector formation, and dimensionality reduction. First, bag preparation creates a set of bag that covers all possible DH in the training set. Then feature vector formation

begins the process by extracting DH of every foreground pixel and put into its appropriate bag. Each component of the feature vector is defined by the number of DH counted in each bag divided by the number of image pixels. Finally, Principal component analysis (PCA) is applied to compress the dimension of the feature vectors by removing vector components redundancy. The compressed feature vectors are the input for the neural network in the classification module.

4.2.1 Bag preparation

Each pixel on the image creates one DH. Therefore, pixels of the entire image create a bunch of different DHs (Figure 4.6). Each DH go through the "first come-first add" algorithm right after it is created. In this process, Bag of Histogram is gradually created as the new DH is calculated. As shown in Figure 4.7, the first DH always create the first bag. After that, there are two possibilities. If the new DH is different from every existing bag, this DH create a new bag (Figure 4.8). On the other hand, if the new DH fit to an existing bag, it is thrown away (Figure 4.9). At this step, the bag that is thrown away still leave a mark on the bag that fit to it. The reason is to get rid out of insignificant bag which get a few marks at the end of the bag preparation. Bag pruning process will be described later. Finally, the set of distinct bags are saved (Figure 4.10Figure) and feature formation slots corresponding to the bags are prepared for the counting in the feature formation process (Figure 4.11).

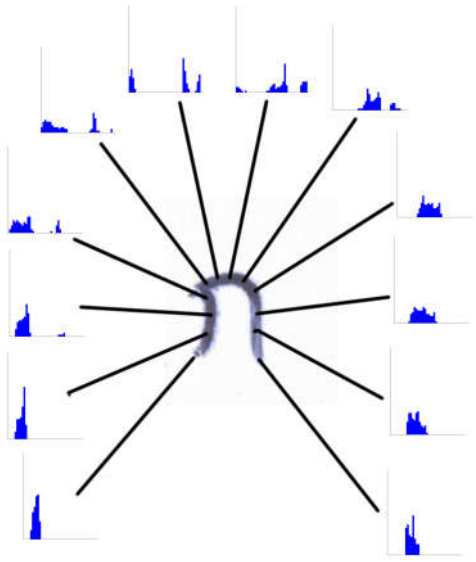


Figure 4.6 Different DHs from different position of ก

Figure 4.7 The first DH always create the first bag

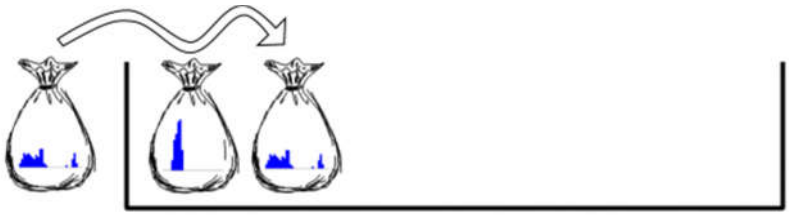


Figure 4.8 New DH that does not match existing bags creates a new bag

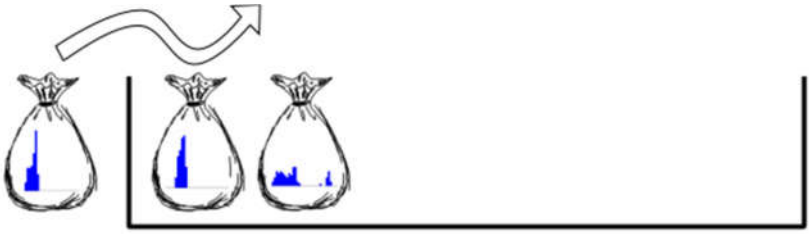


Figure 4.9 New DH that match an existing bag is thrown away

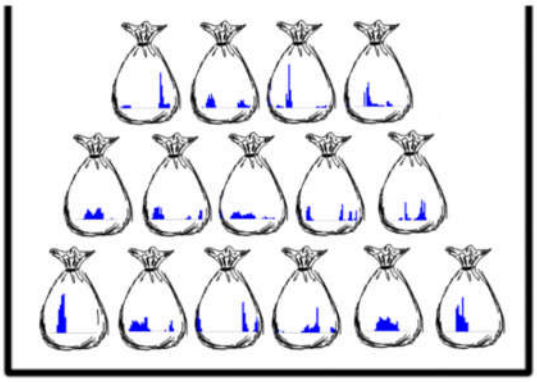


Figure 4.10 Bag of histogram after putting every DH from ก, ข and ค

The number of bags in the BoH can grow up to ten thousands, which take enormous memory. Before any other operation, insignificant bags are pruned out of the set of bags. During the bag preparation process, the repeating bag leave a mark on the existing bag to count how many DHs ever fit that bag. The bag that get the fewest

mark will be deleted first. The deletion iterates until the number of bags remains less than the number we set, which is 2000. The set of remaining bags becomes the real BoH that construct the slot array for DH counting.

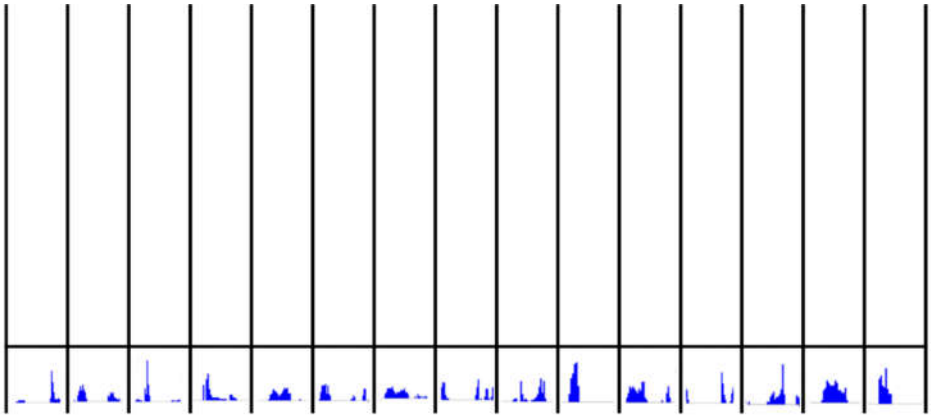


Figure 4.11 Slot array for counting DHs from training and unknown images

4.2.2 Feature formation

This is the essential of feature extraction where the operational feature starts to come out of the image. DH that we define and use in the bag preparation is acquired again as elements that we count into the slot we have prepared. The training image is digested into a set of DHs. These DHs are distinct (see the uniqueness statement in section **Error! Reference source not found.**) but some of them are similar to each other. Each DH must find at least one slot that fit to itself within range T, where T is the DH different threshold (described in section **Error! Reference source not found.**). Creating BoH using the same dataset as the training set guarantees that BoH cover every DH from every training image. DHs in each slot accumulate differently and become the identity of that character. Figure 4.12 demonstrates the different patterns of DH accumulation in ก and ข. We use this accumulation as the feature vector of the character image.

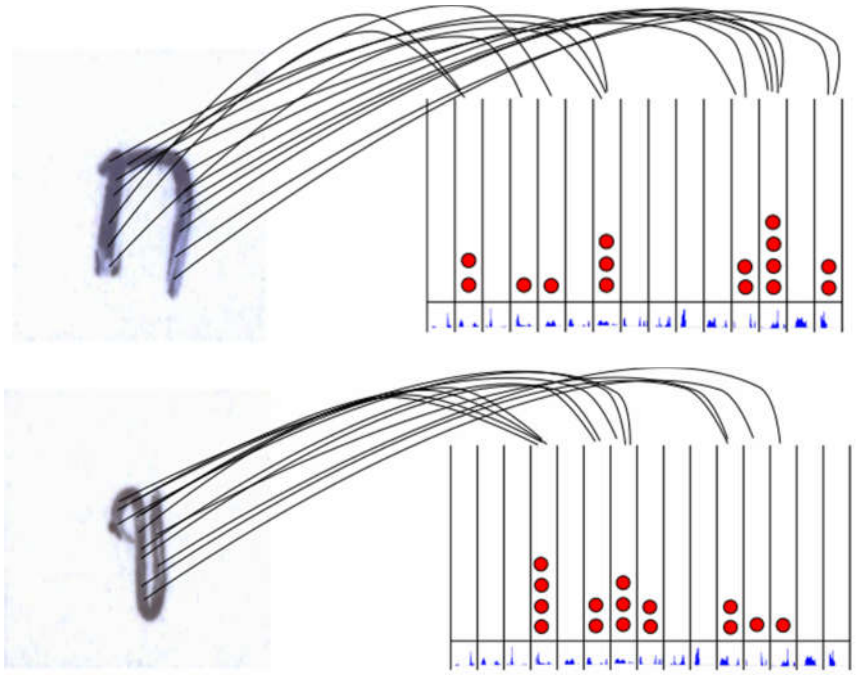


Figure 4.12 DH accumulation of ก and ข shows the different pattern of accumulation.

4.2.3 Principal component analysis

We exploit the PCA which is an efficient algorithm that create a new set of uncorrelated variables, and truncate data at the tail without losing useful information. To apply PCA, the set of feature vectors is imported directly to the PCA process. For the big training set, for example, the mixed dataset from multiple handwritings, the set of feature vector is randomly picked up as a sparse subset for the purpose of efficiency in PCA calculation. From the experiment, calculating PCA transformation matrix on the mixed dataset of 168,000 feature vectors run out of memory 8 GB on our desktop computer. The random sampling that picks up 3,300 vectors also create transformation matrix. We also set up the experiment on comparison of recognition accuracy by varying the PCA compression size from 50 to 500.

4.3 Classification

The regular 2-layer feed-forward neural network with logistic activation function is trained by the second-order back propagation for classifying 80 character

classes from the compressed feature vector. Artificial neural network is chosen because of the performance in multi-class classification. The experiment was done to choose the appropriate number of hidden node in range 500 to 1000.

4.4 Recognition process

In recognition process, an unseen input image is pre-processed by cropping and normalization as described in section 4.1. Then feature vector is extracted and compressed. Bag preparation is omitted because the feature vector formation uses the existing bag of histogram already created in the training process. Finally, the compressed feature vector is classified by the trained neural network. Everything in the recognition process follows the same pathway as in training process in Figure 4.1 except the bag preparation.

CHAPTER V
EVALUATION

In this chapter, we report and discuss on our experiment in bag of histogram-based recognition system. The evaluation process includes handwritten character data collection, benchmark of recognition accuracy on single handwriting, mixed handwriting, thick and distorted datasets. Experiment on parameters comparison (DH dimension, bag different threshold, PCA compression size, and NN hidden nodes) is also included in this chapter. We use the convolutional neural network recognition system as the baseline in our benchmark.

5.1 Dataset collection

Dataset is collected from 52 Thai peoples with the age ranges from 18-55 year-old and various occupations such as student, teacher, officer, doctor, pharmacist, and engineer. There are 80 Thai characters in a set including 44 consonants, 22 vowels, 4 tone marks, and 10 digits. Each participant was asked to write the natural handwriting for 50 sets. The entire dataset, therefore, contains 208000 characters in total.

Datasets in the collecting form were scanned and cut automatically. We designed the cutting method by using Hough transform to straight line of the grid table and construct the frame of each table cell to crop individual character separately.

5.1.1 Pilot study and sample size calculation

The recognition accuracy of each participant's handwriting is a data point. To make the result statistically significant, we want enough samples so that the sample mean represents the population mean. We set the value of sample mean of accuracy to deviate from the population mean less than 2% using the significant level at 99%. After investigated on the pilot study of 10 samples, it is found that the sample mean of

the pilot is 94.14% and the standard deviation is 4.72%. The sample size to achieve the desired statistical significant can be calculated from the formula

$$N = \frac{\sigma^2 Z_{1-\alpha}{}^2}{d^2},$$

where $Z_{1-\alpha} = 2.575,\ d = 2, \sigma = 4.72.$ Therefore, $N = \frac{4.72^2 2.575^2}{2^2} = 36.9,$ which means there must be at least 37 participants recruited in this research. In the real data collection, we have collected the handwritten data from 52 participants.

5.1.2 Collecting form

Datasets were written in 10-page-form per person. The example of filled form is as shown in Figure 5.1. There are 520 pages from 52 participants, so we have to scan and crop these of character automatically.

5.1.3 Ethical approval

The research conduct was approved by the Mahidol University Institutional Review Board (MU-IRB) with Code of Approval number: MU-IRB 2013/154.2911. The proposal passed the review board with the expedited review. The data collection might cause the risk on slightly hurt of the wrist or some stress when writing continuously for a long time. The approval cover the data collection dated from December 2013-December 2014.

แผ่นที่ 1/10

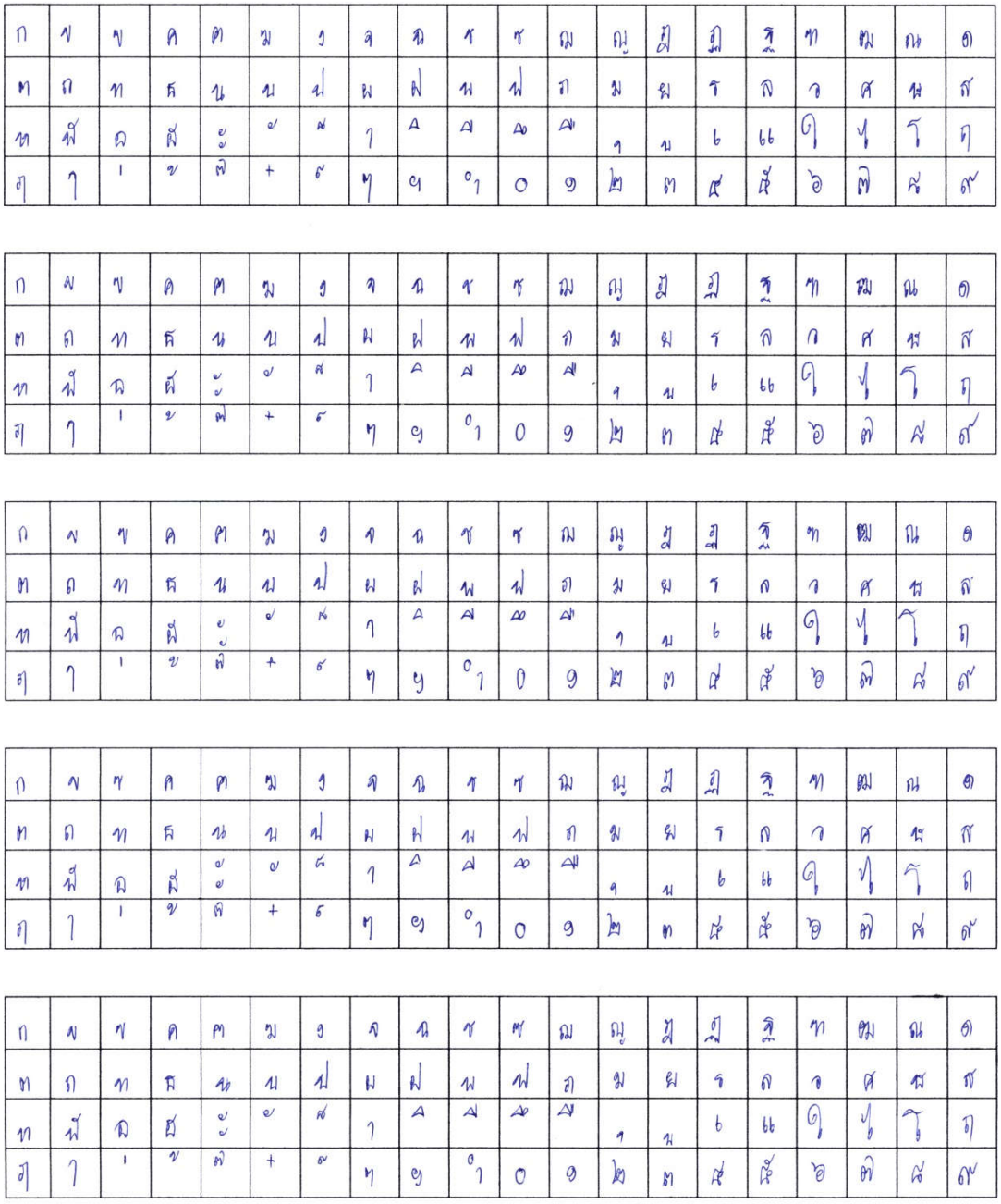


Figure 5.1 Raw datasets

Figure 5.2 Segmented dataset

5.1.4 Scanning and cropping

The dataset was scanned by using a flatbed scanner with resolution 600 DPI. Image was saved into a JPEG files for each page.

Method to crop the pages of data form into separated character consists of detecting the straight line of the table and constructing the frame of each table cell to crop each individual character separately.

We use Hough transformation to detect straight line. Hough transform projects each foreground pixel in the image into polar coordinate axis (r, θ). Local maxima in the polar coordinate axis represent the existence of straight lines.

Next, straight lines are arranged vertically and horizontally. There are 25 horizontal lines $(h_1 - h_{25})$ and 21 vertical lines $(v_1 - v_{21})$ in total. Four junctions between $h_i, h_{i+1}, v_j, v_{j+1}$ forms four corners of a character cell. Character image was cropped by these new corners and saved.

5.2 Implementation

Bag or histogram algorithm is implemented in Java. PCA java library is distributed by Mateusz Kobos. (2013). We use the convolutional neural network (CNN) with second order back-propagation distributed by Mike O'Neill (2006) as the baseline for accuracy comparison and as a normal feed-forward neural network in our BoH system. The entire system was written and built using Eclipse SDK.

Recognition accuracy is measured by the fraction of correctly recognizes character divided by the number of characters in the test set. For each single

handwriting experiment, the number of character in the test set is 1,600, and for the mixed handwriting experiment, the number of test set is 32,000.

5.3 Effect of parameters

There are four parameters that expected to have influence to the recognition accuracy. The first two parameters belongs to BoH configuration, i.e., the dimension of DH (D) and the bag different threshold (T). Another two belongs to PCA and NN in general, i.e., the PCA compression size (P) and the number of hidden nodes in the neural network (H). Testing by crossing every possible value of all four parameters will take time. So we choose the more efficient way which is to inspect the nature of these parameters in the wide range. At first, we try to cross all value of the four parameters by using $\mathrm{D} = \{20, 40, 60\}$, $\mathrm{T} = \{0.05,\ 0.1,\ 0.15\}$, $\mathrm{P} = \{400,\ 500,\ 600\}$, $\mathrm{H} = \{500,\ 700,\ 1000\}$. We selected the handwriting number 36 and applied every configuration among all four parameters.

Table 5.1 indicates that PCA compression size and NN hidden nodes do not clearly influence the recognition accuracy. Whereas the other two parameter T and D show significant effect to the accuracy. Firstly, we can tell from the table that smaller T gives better accuracy. The bigger D seems to be positively influence the accuracy but the data is slight different between $\mathrm{D} = 40$ and $\mathrm{D} = 60$ so it can be argued for the further investigation.

Table 5.1 Recognition accuracy (%) of handwriting index 36 using different values of parameters

D=20 T=0.05			
H	P		
	400	500	600
500	95.44	95.81	95.50
700	96.00	96.06	96.00
1000	96.13	96.13	96.00

D=40 T=0.05			
H	P		
	400	500	600
500	96.63	96.94	96.75
700	97.06	96.69	96.75
1000	96.94	96.75	96.75

D=60 T=0.05			
H	P		
	400	500	600
500	96.25	96.13	96.31
700	96.38	96.44	96.00
1000	96.25	96.56	96.06

D=20 T=0.1			
H	P		
	400	500	600
500	95.19	95.38	94.75
700	95.31	95.50	95.13
1000	95.56	95.38	95.13

D=40 T=0.1			
H	P		
	400	500	600
500	95.56	95.56	95.56
700	95.88	95.19	95.50
1000	95.94	94.93	95.50

D=60 T=0.1			
H	P		
	400	500	600
500	93.88	94.13	93.88
700	93.94	93.88	94.00
1000	94.31	94.00	93.88

D=20 T=0.15			
H	P		
	400	500	600
500	91.56	91.25	90.75
700	91.06	91.38	91.25
1000	91.81	91.56	91.31

D=40 T=0.15			
H	P		
	400	500	600
500	85.50	86.00	85.88
700	85.69	85.13	85.63
1000	85.06	85.44	85.25

D=60 T=0.15			
H	P		
	400	500	600
500	70.00	69.69	69.88
700	69.44	69.44	69.63
1000	68.44	68.25	68.19

We then setup the experiment to find the appropriate value of P and H. The data in Table 5.1 shows no difference when varying P from 400 to 600 and H from 500 to 1000. We extend these ranges to cover the lower end of what we have done. The purpose of this experiment is to find the optimal values of P and H that also safe the processing resource and give a noble result as using big P and H. The range of P and H used in this experiment is $\{50, 100, 300, 500\}$ and $\{100, 300, 500, 700\}$ respectively.

Table 5.2 Recognition accuracy of handwriting index 36 when varying PCA compression size and the number of NN hidden nodes.

	PCA50	PCA100	PCA300	PCA500
HD100	1402	1477	1521	1512
HD300	1436	1521	1550	1548
HD500	1426	1517	1547	1555
HD700	1424	1516	1555	1550
HD900	1412	1513	1548	1551

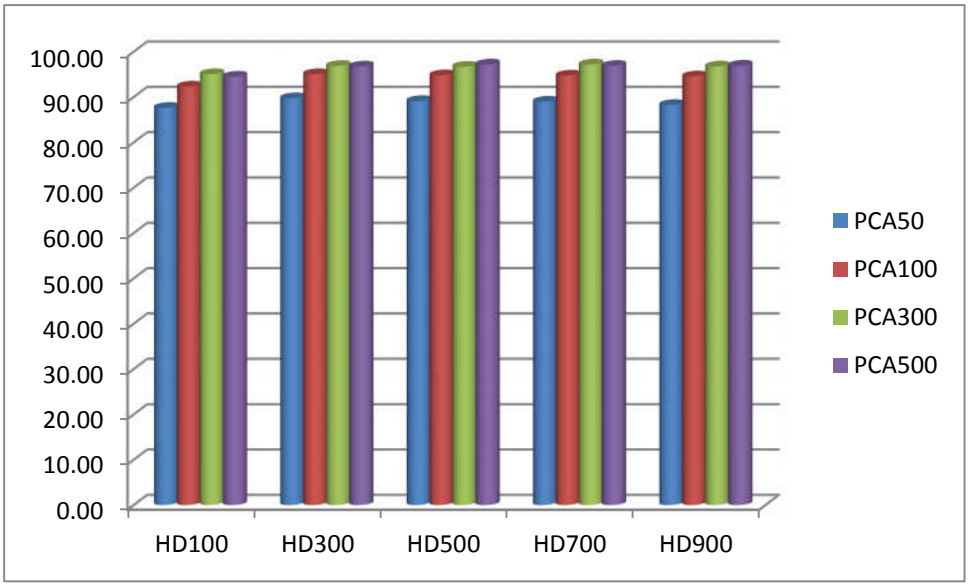


Figure 5.3 Clustered bar chart showing recognition accuracy comparison in Table 5.2

The result shows that bigger P and H contributes to the better accuracy until a certain point where the accuracy start to be stable. From Table 5.2, P that start to stabilize the accuracy is 300, where H is 300 as well. It follows our intuition that the less compression contains more useful information for recognition. At a point, adding more information does not gain any better recognition accuracy. Similarly, the hidden nodes of NN create its own abstract feature. The higher number of feature gain higher accuracy but when it reaches the sufficient point, too many nodes does not provide any better accuracy as well.

After the experiment on PCA and NN parameters, we then explored the effects of BoH parameters in finer scale by varying the dimension of direction

histogram from 10, 20, 30, 40, 50 and 60 and DH different threshold from 0.05, 0.1 and 0.15. Ten handwritings are randomly selected for training and testing with all 18 configurations. Average recognition accuracy (%) from the selected 10 handwritings on every pair of parameter is shown in Table 5.3. Yellow highlight cell indicates the highest accuracy corresponding to each value of T.

Figure 5.4 demonstrates the effect of bag dimensionality and threshold. Bag of histogram with 40 dimensions in each bag and $T = 0.05$ achieves the highest accuracy at 95.01%. The graph indicates that lower threshold gives the best accuracy at the higher dimension. The best recognition accuracy can be achieved at high dimension ($D = 40$) and low threshold ($T = 0.05$) as shown in Figure 5.4. Intuitively, low bag dimension provides too coarse information in the feature so that it cannot separate some similar pair of characters. Higher dimension gives finer information for discrimination, but it also needs high resolution of bag, which is the consequence of low threshold, in order to get better recognition accuracy when dimension increases.

Table 5.3 Average recognition accuracy corresponding to every pair of parameters in 10 handwritings

T	D					
	10	20	30	40	50	60
0.05	93.56%	94.44%	94.91%	95.02%	93.01%	92.01%
0.10	94.02%	94.59%	92.91%	87.29%	75.26%	76.49%
0.15	90.09%	89.09%	84.21%	77.04%	52.19%	39.86%

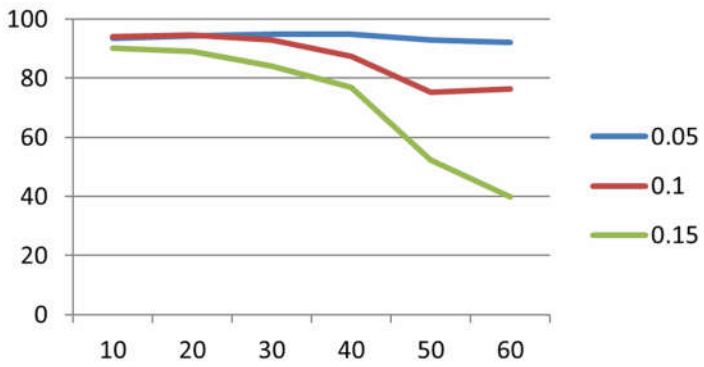


Figure 5.4 Effect of bag dimensionality and threshold to average recognition accuracy

If both high dimension and high threshold are used, high dimensional DHs that contain some discriminative features are eventually put into the same bag. It violates the information that is useful for discrimination, and leads to the drop of accuracy when T is higher (0.1 and 0.15).

5.4 Performance

5.4.1 Single handwriting test

Accuracy for each handwriting with the best parameter configuration ($\mathrm{T} = 0.05, \mathrm{d} = 40$) is examined. The accuracy is compared to the result from convolutional neural network (CNN).

CNN is chosen to be our baseline because of its ability to develop its own feature and the remarkable recognition accuracy. The accuracy of BoH and CNN recognition system tested with handwriting from every participant is shown in

Table 5.4 and the comparison graph is visualized in Figure 5.5. The average accuracy of every set yields 95.18%.

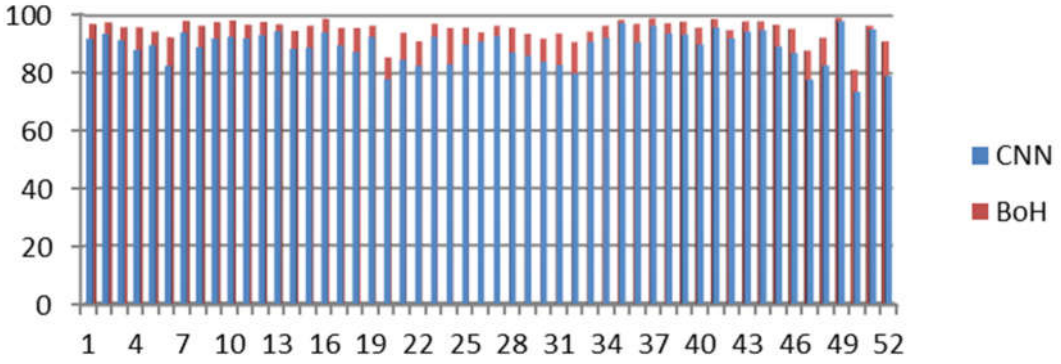


Figure 5.5 Recognition accuracy comparison between OCR system using CNN and BoH

Table 5.4 Recognition accuracy (%) comparison between OCR system using CNN and BoH

HW	CNN	BoH	HW	CNN	BoH	HW	CNN	BoH	HW	CNN	BoH
1	92.25	96.69	14	89.88	94.13	27	92.44	96.50	40	91.75	96.19
2	93.06	97.19	15	90.69	96.44	28	89.44	95.69	41	95.88	98.50
3	91.56	95.81	16	95.31	98.69	29	84.25	94.06	42	91.44	94.50
4	81.69	96.19	17	90.88	96.19	30	79.06	91.50	43	93.81	97.81
5	89.00	95.00	18	90.44	96.00	31	85.88	93.69	44	95.44	98.06
6	81.75	92.44	19	92.63	96.06	32	80.81	90.56	45	90.88	96.94
7	94.81	98.31	20	80.25	85.81	33	91.19	94.19	46	89.00	95.63
8	89.31	96.25	21	86.56	93.94	34	93.88	96.06	47	77.13	88.06
9	93.19	97.75	22	83.81	91.31	35	96.44	98.44	48	85.56	91.81
10	93.63	97.63	23	92.88	97.25	36	91.31	97.19	49	97.50	99.25
11	93.06	97.25	24	85.75	95.75	37	96.25	98.94	50	73.50	81.63
12	93.56	97.63	25	90.56	95.94	38	93.19	97.00	51	94.75	96.56
13	94.06	96.94	26	90.38	94.31	39	93.56	97.50	52	80.63	91.63

5.4.2 Mixed handwriting test

Practically, OCR operates on unseen data from unknown handwriting. To setup the experiment for testing totally unseen data, we divided the 52 sets of handwritten data into two parts. The first 40 handwriting is for training and the rest is for testing. Training data from 40 handwritings are shuffled so that the character index and set index are random.

Randomized set of feature vectors is used in PCA transformation for the efficiency purpose. Only 3,300 feature vectors out of 168,000 are randomly picked up from the set of all training feature vector to calculate the PCA transformation matrix. After that, the entire set of feature vector is transformed by using the transformation matrix.

Table 5.5 Recognition accuracy comparison between OCR system using CNN and BoH when tested with the mixed dataset

	CNN	BoH
Correct	28017	29970
Percent	87.55%	93.66%

The recognition accuracy of the mixed handwriting test, although drops around 1.5% compared to the average of every single handwriting test, still overcome the accuracy from CNN recognition.

5.4.3 Thick and distort dataset

To demonstrate the efficiency of BoH over distorted and thick character shape, the original dataset is modified by the dilation and mesh warp filter. Dilation is the process that expands each foreground pixels to its 8-neighborhood. We apply dilation that gives 20% increasing of foreground pixel area, so the character stroke is thicker than the original. Mesh warp is the process to distort image by creating a direction field for each pixel which is used for mapping it to the new position. Mesh warp starts by selecting a set of initial vectors, which is randomly selected in this study. The maximum amplitude of initial vectors is set to be 20% of the longest image side. The initial vectors then are smoothen by Gaussian function, which creates a smooth direction field for pixel mapping.

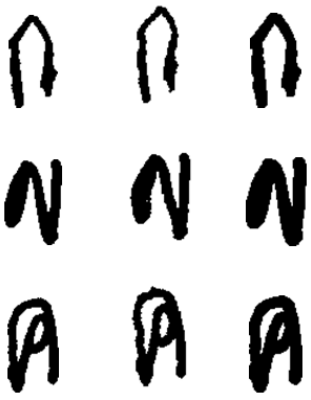


Figure 5.6 Examples of distorted and thick dataset. Left: original image. Center: distorted image. Right: thick image

Thick and distorted accuracy was tested by training the recognition system using the original training set and then measure the accuracy by using the thick and distorted test set. The accuracy is shown Table 5.6. The accuracy when tested on thick and distorted dataset is dropped by average 3.83 % and 6.44 % respectively, compared to that of CNN which is dropped by 1.75 % and 19.86 %. We can see that OCR system with BoH feature extraction can handle the unseen distortion much better than the CNN OCR system. However, CNN is more robust to the unseen thickness than BoH.

Table 5.6 Recognition accuracy of OCR system with CNN and BoH on thick and distorted datasets

	BoH	CNN
Original	95.18%	88.91%
Thick	91.35%	87.22%
Distorted	88.74%	69.05%

5.4.4 Similar character pairs

In this section, we pay attention on the set of similar Thai characters that we have raised in the introduction. Since the misrecognition can be recorded into the statistical data in the form of confusion matrix, we use the confusion matrix to compare the ability to discriminate similar pair of characters between BoH and CNN.

Each character class in the mixed test set contains 400 testing images. From the mixed handwriting test on BoH and CNN, the overall confusion in Table 5. shows that the confusing characters of CNN is in a wider range than that of BoH. We inspect the significant confusion made by both BoH and CNN, categorize the confusion due to its possible source of confusion and conclude the categories into Table 5.8 to 5.13. For the most part in every confusion category, BoH result shows the fewer number of confusion than CNN.

Table 5.7 Confusion table of every character class recognized by BoH and CNN

Char	Confusion by BoH	Confusion by CNN
ก	ค = 3 ฅ = 1 ญ = 2 ฑ = 2 ถ = 19 ท = 6 ภ = 1 ม = 1 ส = 2 ๙ = 1	ค = 5 ฑ = 2 ด = 2 ต = 20 ถ = 48 ท = 10 ภ = 19 ล = 7 ส = 1 อิ = 4 ฦ = 2 ๑ = 2 ๓ = 2
ข	ฃ = 25 จ = 1 ฉ = 1 ช = 7 ซ = 5 ธ = 28 น = 1 บ = 13 ผ = 1 พ = 1 ย = 7 ร = 4 อี = 9 อุ = 6 อู = 95 ไ = 4 อ้ = 24	ฃ = 11 จ = 1 ช = 18 ซ = 12 ญ = 2 ธ = 30 น = 3 บ = 12 พ = 1 ย = 19 ร = 4 ล = 1 ห = 1 ฮ = 1 อิ = 2 อี = 10 อึ = 1 อุ = 7 อู = 67 ใ = 1 ไ = 1 อ้ = 22 ฯ = 3
ฃ	ข = 5 ฆ = 4 ซ = 19 ฌ = 1 ท = 1 ธ = 5 น = 4 บ = 1 ร = 3 ษ = 1 ห = 1 อึ = 6 อู = 38 อ๊ = 1	ข = 23 ฆ = 3 จ = 2 ช = 2 ซ = 12 ญ = 1 ต = 1 ธ = 11 น = 5 พ = 3 ร = 4 อ = 1 อุ = 8 อู = 21 อ๊ = 4 ฯ = 2 ๖ = 4
ค	ฅ = 26 ฉ = 1 ด = 15 ถ = 1 ล = 1 อิ = 2	ฅ = 27 จ = 2 ฉ = 4 ด = 34 ต = 13 ถ = 9 ภ = 3 ล = 14 ศ = 1 อ = 7 แ = 1 ๓ = 6 ๕ = 4 ๙ = 1
ฅ	ค = 13 ต = 13 ท = 2 ๓ = 9	ก = 1 ฃ = 1 ค = 44 จ = 1 ต = 51 ท = 2 ล = 1 อึ = 1 อื = 1 อำ = 1 ๓ = 56 ๔ = 2 ๕ = 1
ฆ	ฌ = 1 ฎ = 1 ฏ = 1 ฆ = 2 น = 1 ม = 51 ว = 2 อี = 2 อึ = 2 ๖ = 1	ฉ = 2 ฌ = 16 ฆ = 9 ณ = 2 ธ = 1 น = 1 พ = 5 ม = 51 ย = 5 ร = 3 ล = 5 ว = 2 อ = 1 อี = 1 อู = 1 อ๊ = 4 ๙ = 1
ง	ฎ = 3 ย = 1 ร = 3 ฟ = 1 อ = 3 ฮ = 3 อุ = 2 อู = 1 ฯ = 1 ๑ = 23	ข = 3 จ = 3 ฉ = 1 ฎ = 13 ฏ = 1 ฐ = 2 ธ = 4 ย = 2 ร = 1 ฟ = 2 อ = 10 ฮ = 1 อิ = 1 อี = 1 อุ = 4 ฦ = 1 ฯ = 2 ๑ = 15 ๒ = 9

Table 5.7 Confusion table of every character class recognized by BoH and CNN (cont.)

Char	Confusion by BoH	Confusion by CNN
จ	ก = 2 ฉ = 1 ด = 2 ธ = 2 ล = 4 อิ = 4 อึ = 2 อื = 2 อุ = 25 อู = 2	ข = 1 ฉ = 2 ด = 8 ถ = 2 ล = 11 ว = 3 อ = 4 อิ = 1 อึ = 4 อื = 13 อุ = 25 ฤ = 4 ฦ = 2 ฯ = 2 อำ = 3 ๑ = 1 ๔ = 1
ฉ	ข = 1 ค = 1 จ = 2 ด = 2 ธ = 2 น = 2 ล = 5 อิ = 3 อึ = 2 ไ = 1	ก = 2 จ = 22 ด = 7 ต = 1 ธ = 3 น = 12 ร = 1 ล = 18 อ = 1 อี = 1 อิ = 10 อึ = 4 ไ = 1 ๑ = 2 ๖ = 1
ช	ข = 1 ซ = 29 ธ = 44 น = 1 บ = 1 ย = 17 ส = 2 ห = 5 ะ = 1 อี = 6 อึ = 5 อู = 4 โ = 1 อ์ = 2 ๔ = 1 ๘ = 1	ซ = 37 ธ = 36 น = 4 ร = 2 ย = 4 ส = 1 ห = 5 อ = 1 ฮ = 1 อี = 16 อู = 8 เ = 1 อ๊ = 3 อ์ = 2 ๕ = 7
ซ	ฃ = 4 ช = 16 ฒ = 2 ธ = 16 น = 2 ฝ = 1 ร = 3 ย = 12 อี = 4 อึ = 2 อู = 4 อ๊ = 3 ๘ = 3 ๕ = 2	ฃ = 8 ช = 38 ญ = 2 ฌ = 2 ธ = 15 ป = 1 ร = 2 ย = 7 ส = 1 ฮ = 1 อี = 10 อ้ = 2 อ๊ = 1 ฯ = 3 ๔ = 2 ๗ = 1 ๕ = 2
ฌ	ฒ = 6 ฌ = 6 อี = 1 อึ = 1 อ๊ = 4	ฉ = 1 ฒ = 2 ฌ = 8 พ = 2 อิ = 14 อื = 4 ๗ = 1 ๘ = 1
ญ	ฃ = 1 ซ = 1 ฒ = 3 ฌ = 7 ร = 2 อี = 1 ๕ = 2	ฃ = 3 ช = 1 ฌ = 2 น = 5 ย = 4 ว = 3 ะ = 1 อิ = 4 ฯ = 2
ฎ	ฉ = 1 ฎ = 64 ว = 3 อี = 1 ๖ = 1	ฉ = 11 ฎ = 50 ฐ = 1 ฒ = 1 ธ = 1 น = 1 บ = 2 ย = 2 ร = 1 ล = 2 ว = 4 ษ = 1 ส = 2 อี = 4 ๑ = 1 ๔ = 2 ๖ = 1 ๗ = 1 ๕ = 3
ฏ	ฎ = 45 ฐ = 6 อี = 3 อิ = 8 ๖ = 1	ฎ = 98 ฐ = 3 ธ = 1 ฝ = 3 อ = 2 อิ = 7 อำ = 1 ๘ = 1 ๕ = 3
ฐ	ฎ = 1	ฎ = 2 ฏ = 2 ส = 1 ห = 2 อี = 1 ๘ = 1
ฑ	ท = 9 พ = 1 อื = 1	ก = 1 ต = 8 ท = 15 น = 1 พ = 1

Table 5.7 Confusion table of every character class recognized by BoH and CNN (cont.)

Char	Confusion by BoH	Confusion by CNN
		ร = 2 อำ = 1
ฒ	ฒ = 10 ฌ = 4 อึ = 1	ฒ = 7 ฌ = 10 ณ = 1 อ = 3 อิ = 7 อึ = 1 อั๊ = 4 ๔ = 4
ณ	ฌ = 5 น = 1 ผ = 1 ม = 1	ฌ = 6 ต = 1 น = 5 พ = 1 อี = 4 อึ = 2 อู = 1 อั๊ = 1
ด	ก = 1 ค = 13 จ = 1 ต = 2 ถ = 2 ภ = 2 ๐ = 1 ๑ = 2 ๓ = 1	ก = 1 ข = 1 ค = 13 จ = 3 ฉ = 1 ต = 10 ถ = 5 ภ = 7 ย = 1 ล = 7 อ = 1 อิ = 1 อุ = 1 ใ = 2 ฤ = 1 ฦ = 3 ๑ = 4 ๓ = 2 ๙ = 1
ต	ศ = 5 ด = 1 ท = 2 ฮ = 1 ๓ = 20	ค = 2 ศ = 5 ฑ = 1 ด = 2 ท = 5 ส = 2 ฦ = 1 ๆ = 1 ๓ = 52
ถ	ก = 2 ค = 1 จ = 1 ด = 3 ภ = 5 ล = 1 ห = 1 ฤ = 3	ก = 7 จ = 2 ฉ = 4 ญ = 1 ด = 3 ต = 1 ท = 1 ภ = 9 ล = 4 แ = 1 ฤ = 4 ฦ = 4 ๐ = 1 ๑ = 1
ท	ก = 1 ฑ = 5 ด = 2 ถ = 1	ข = 1 ฉ = 1 ฑ = 6 ต = 10 ถ = 1 ห = 3 ฯ = 1 อำ = 4 ๓ = 2
ธ	ช = 5 ษ = 2 อ = 2 ฮ = 1 อี๋ = 1 อ์ = 1 ๒ = 1 ๕ = 1	ก = 3 ข = 1 จ = 2 ช = 11 ถ = 1 ย = 1 ร = 8 ส = 3 อ = 4 ฮ = 5 อี๋ = 1 อ์ = 2 ๒ = 1
น	ณ = 4 บ = 1 พ = 3 อิ = 1	ฉ = 1 ซ = 1 ณ = 3 ท = 1 บ = 4 พ = 3 ย = 1 ล = 2 ว = 1 ส = 1 อู = 1 ๖ = 2
บ	ข = 7 ท = 2 น = 2 พ = 1 ม = 1 ย = 2 อู = 10	ข = 8 ฒ = 2 น = 1 พ = 2 ม = 3 ย = 8 อี = 1 อู = 21 อ์ = 1
ป	ช = 2 บ = 2 ฝ = 5 ฟ = 1	ฒ = 2 ช = 3 บ = 1 ฝ = 9 ฟ = 8 ศ = 1 ส = 1 ฬ = 2 อ = 1 อั = 1 ๆ = 2
ผ	ณ = 1 ฝ = 1 พ = 3 ม = 2 อี = 1	ข = 5 ฒ = 5 ณ = 1 ต = 2 ธ = 3

Table 5.7 Confusion table of every character class recognized by BoH and CNN (cont.)

Char	Confusion by BoH	Confusion by CNN
	๒ = 1	น = 1 บ = 4 ฝ = 1 พ = 6 ม = 8 ย = 10 ล = 3 อ = 1 อิ = 1 อี = 3 อ๊ = 3 อ์ = 1 ๔ = 1 ๙ = 2
ฝ	ฟ = 1	ป = 3 ผ = 1 ฟ = 14 ฬ = 1 ๘ = 1
พ	ฒ = 1 บ = 3 ฟ = 5 ม = 1 ห = 2 อ๊ = 2	ฒ = 1 ต = 1 น = 3 บ = 1 ฟ = 8 ม = 3 ย = 3 อู = 9 อ้ = 3 อ๊ = 1 ฯ = 1 ๗ = 4 ๙ = 1
ฟ	ป = 5 ฝ = 1 พ = 8	ฏ = 1 ป = 7 ฝ = 13 พ = 9 ส = 1 ฬ = 2
ภ	ก = 2 ด = 5 ต = 1 ถ = 8 ล = 1 ฤ = 1	ก = 5 ค = 2 จ = 1 ฑ = 1 ด = 6 ต = 2 ถ = 7 ล = 2 ฬ = 1 อ = 1 ใ = 1 ฦ = 3 อ์ = 1 ๓ = 2
ม	ฒ = 2 ฌ = 1 ผ = 4 พ = 1 ร = 3	ฒ = 3 น = 2 บ = 2 ผ = 9 พ = 2 ย = 1 อ = 2 อู = 1 อ๊ = 4 ๔ = 2 ๗ = 1
ย	ข = 1 ง = 1 ธ = 1 บ = 1 ป = 1 ผ = 2 ม = 1 อ้ = 1	ก = 1 ข = 1 ฉ = 2 ธ = 1 บ = 7 ม = 3 ล = 1 ห = 1 ฬ = 1 อ = 1 ฮ = 2 อ็ = 1 อี = 1 อุ = 1 อู = 4 อ้ = 1 ๑ = 1 ๙ = 1
ร	ข = 3 ช = 2 ซ = 1 ธ = 3 ว = 2 ส = 1 ฮ = 4 อั = 1 า = 1 อ้ = 1 อ์ = 1 ๕ = 1	ข = 1 ฃ = 1 ฉ = 2 ช = 6 ซ = 1 ฐ = 1 ธ = 7 ภ = 1 ย = 1 ว = 2 ส = 4 อ = 1 ฮ = 2 อั = 1 ใ = 1 โ = 3 อ์ = 1 ฯ = 1 ๔ = 2 ๕ = 2 ๙ = 2
ล	ก = 1 ฉ = 1 ด = 1 ว = 3 อ = 1 ๗ = 2	ค = 2 จ = 5 ฉ = 2 ฏ = 1 ด = 1 ถ = 1 ภ = 1 ว = 2 อ = 2 อี = 1 อ๊ = 1 ๑ = 1 ๗ = 1 ๙ = 2
ว	ล = 1 อ่ = 1	จ = 1 ล = 6 อ = 5 า = 1 อิ = 1

Table 5.7 Confusion table of every character class recognized by BoH and CNN (cont.)

Char	Confusion by BoH	Confusion by CNN
		อุ = 1 ใ = 1 ๖ = 1
ศ	ค = 1 ช = 1 ธ = 2 ผ = 2 ส = 6 อี๋ = 1 ๔ = 1 ๗ = 1	ค = 1 ช = 3 ฐ = 1 ต = 2 ธ = 2 ส = 13 พ = 1 อี๋ = 10 อิ = 1 อึ = 1 อ๋ = 1 ๔ = 3 ๕ = 1 ๗ = 3 ๘ = 5 ๙ = 9
ษ	ช = 1 ซ = 1 ธ = 1 ห = 3	ข = 6 ช = 3 ซ = 2 ด = 1 ธ = 7 น = 1 บ = 2 พ = 2 ย = 1 ร = 1 ห = 8 ฟ = 3 อั = 2 อี๋ = 1 า = 1 อิ = 1 อู = 2 อ้ = 1 อ๊ = 1 อ๋ = 2 ๙ = 1
ส	ง = 1 ธ = 1 ศ = 2 ฮ = 2 อี๋ = 2 ๔ = 2 ๕ = 1 ๘ = 1 ๙ = 2	พ = 1 ร = 1 ศ = 9 ฟ = 5 ฮ = 2 อี๋ = 2 อ๋ = 3 ๔ = 4 ๕ = 7 ๗ = 1 ๘ = 6 ๙ = 11
ห	ข = 1 ฑ = 1 ษ = 1	ช = 2 ฑ = 2 ต = 4 ท = 2 พ = 1 ฟ = 3 ม = 1 ย = 2 ร = 1 ล = 1 ษ = 1 อี = 1 อู = 2 อ๊ = 1 ๔ = 1 ๙ = 2
ฟ	ป = 1 ษ = 1 ส = 2 ๕ = 1	ธ = 1 ฟ = 1 ษ = 1 ส = 3 ๕ = 3 ๙ = 2
อ	ด = 3 ล = 2 ว = 1 ๐ = 8 ๑ = 3	ค = 2 จ = 3 ฉ = 3 ฒ = 1 ด = 2 ถ = 1 ธ = 4 ย = 1 ล = 17 ว = 1 ห = 1 อั = 3 อึ = 1 ใ = 1 อ๊ = 2 ๐ = 4 ๑ = 7
ฮ	ข = 1 ช = 1 ธ = 6 ร = 6 ศ = 1 ฟ = 2 อ = 1	ช = 11 ธ = 10 ฝ = 1 ร = 5 ส = 12 ห = 1 ฟ = 8 อ = 4 อั = 1 อี๋ = 1 อู = 1 โ = 2 ๔ = 2 ๕ = 16 ๙ = 3
ะ	ช = 1 โ = 1	ข = 1 ช = 1 ย = 1 ร = 2 อ = 1 อ๋ = 1 ฯ = 2 ๕ = 1 ๙ = 1

Table 5.7 Confusion table of every character class recognized by BoH and CNN (cont.)

Char	Confusion by BoH	Confusion by CNN
อั	อ้ = 2 อ๊ = 2 อ์ = 1	ๅ = 1 ย = 1 อี่ = 1 ไ = 2 อ้ = 3 อ๊ = 1 อ์ = 4 ๔ = 1
อี๋	จ = 2 ช = 2 ศ = 3 ษ = 1 ส = 4 อั = 2 อิ = 2 อี = 1 อึ = 1 อ์ = 2 ๔ = 2 ๕ = 1 ๘ = 32 ๙ = 1	ข = 1 จ = 1 ช = 3 ด = 1 น = 1 ศ = 4 ส = 8 ห = 3 ฮ = 1 อั = 1 อิ = 2 อี = 3 อื = 2 อู = 2 อ๊ = 3 อ์ = 1 ๑ = 2 ๔ = 2 ๕ = 1 ๘ = 25 ๙ = 6
า	อุ = 5 ไ = 1 ไ = 2 ฦ = 1 ๅ = 53 ๆ = 3	ก = 1 ร = 2 อุ = 7 ไ = 1 ไ = 1 ฤ = 1 ๅ = 92 ๆ = 7 ๙ = 2
อิ	อี๋ = 1 ๐ = 2	ค = 1 ธ = 1 อี่ = 1 อึ = 2 อำ = 1 ๐ = 2
อี	อิ = 1 อึ = 4 อื = 4 ๐ = 1	ฆ = 4 จ = 1 ผ = 1 พ = 1 ล = 2 อิ = 5 อื = 2 อ้ = 1 ๗ = 1 ๙ = 1
อึ	อื = 6	จ = 4 อี่ = 1 อิ = 2 อี = 2 อื = 13 ไ = 1 อ์ = 2 อำ = 2 ๒ = 1 ๙ = 1
อื	อี = 4 อึ = 8	จ = 4 ณ = 1 พ = 2 อิ = 5 อี = 9 อึ = 8 อำ = 1 ๗ = 1 ๙ = 1
อุ	จ = 7 ด = 2 า = 3 อ่ = 4 ฯ = 2 อำ = 1	ข = 1 จ = 3 ช = 1 า = 3 อู = 2 ไ = 1 ฤ = 2 อ่ = 3 อ๋ = 1 ๆ = 1 ฯ = 2 ๖ = 1
อู	ข = 37 ฃ = 2 ฆ = 1 จ = 1 ช = 3 บ = 21 ผ = 1 ย = 2 ร = 1 ษ = 1 ส = 1 อ = 1 อื = 1 อุ = 1 อ้ = 9	ข = 62 ฃ = 4 ฆ = 3 ช = 1 ธ = 3 น = 1 บ = 33 ผ = 1 ฝ = 1 ฟ = 2 ม = 2 ย = 7 ร = 1 ว = 1 ห = 3 อุ = 2 เ = 1 อ้ = 9 อ์ = 1 ฯ = 1 ๑ = 2 ๙ = 1
เ	อั = 2 อ่ = 1 อ์ = 2	ช = 1 อิ = 2 อ่ = 1 ๖ = 1
แ		
ไ	อุ = 1 ไ = 2 โ = 12 ฦ = 1 ๅ = 4	ง = 1 จ = 1 า = 1 ไ = 3 โ = 5

Table 5.7 Confusion table of every character class recognized by BoH and CNN (cont.)

Char	Confusion by BoH	Confusion by CNN
		ฤ = 1 ฦ = 1 า = 12 อ่ = 1 ๆ = 1
ไ	เ = 1 า = 2 อ่ = 2	ฃ = 1 เ = 1 ใ = 3 โ = 4 ๆ = 2 ๖ = 1
โ	ฃ = 1 เ = 1 ใ = 4 ไ = 2 า = 1 อ่ = 2 ๘ = 1	ช = 3 ญ = 1 ใ = 7 ไ = 6 า = 3 ๆ = 1 ๕ = 1
ฤ	ก = 1 จ = 3 ถ = 6 อุ = 2 ฦ = 15 ๆ = 1 ๖ = 1	ก = 1 จ = 2 ซ = 1 ถ = 6 บ = 1 ล = 1 อ = 1 า = 1 อุ = 4 ใ = 10 ฦ = 34 า = 1 ๆ = 2 ฯ = 1
ฦ	ด = 1 ภ = 3 ฤ = 6 า = 1 ๆ = 4	ก = 1 ด = 1 ถ = 3 ภ = 4 า = 3 ใ = 4 ฤ = 18 า = 2 ๆ = 5
า	า = 84 ฤ = 2 ๆ = 5	ฝ = 1 ว = 1 า = 54 ใ = 2 โ = 2 ฦ = 2 ๆ = 10
อ่	อุ = 3 ไ = 1 อ์ = 1	ล = 1 เ = 1 ๔ = 1
อ้	ฆ = 2 ย = 1 อั = 6 อู = 2 เ = 1 อ๊ = 1 ๔ = 1	ช = 2 ฟ = 1 ม = 2 ย = 3 ล = 1 ะ = 2 อั = 7 อ็ = 1 อู = 2 เ = 1 อ์ = 3 ๗ = 1 ๘ = 1
อ๊	ต = 1 ฝ = 1 ฟ = 1 ล = 1 อี = 1 อึ = 1 ๓ = 1 ๗ = 19 ๘ = 3	ฆ = 2 ด = 1 ต = 1 ฝ = 1 พ = 2 ฟ = 1 ล = 1 ห = 2 อั = 1 อี = 4 อื = 1 อ้ = 1 อ์ = 2 ๗ = 37 ๘ = 4
อ๋		อื = 1 เ = 1 แ = 1 ไ = 1
อ์	ร = 1 ฮ = 1 อั = 1 อ็ = 1 เ = 6 อ๊ = 1 ๘ = 2	ช = 3 ธ = 1 ย = 2 ส = 2 ห = 1 อั = 3 อี = 2 เ = 2 อ๊ = 3 ๔ = 1 ๘ = 9
ๆ	า = 3 า = 1 ฯ = 5 ๖ = 1	จ = 1 อุ = 1 ใ = 3 โ = 3 ฤ = 4 ฦ = 2 า = 3 ฯ = 11 อำ = 1
ฯ	ผ = 1 ห = 1 อุ = 2 อู = 1 ฦ = 1 า = 1 ๆ = 16 ๑ = 1	ข = 3 จ = 2 ธ = 1 อี = 1 อุ = 3 อู = 2 ใ = 2 ไ = 6 ฤ = 1 ฦ = 2 า = 1 ๆ = 20 อำ = 4

Table 5.7 Confusion table of every character class recognized by BoH and CNN (cont.)

Char	Confusion by BoH	Confusion by CNN
อำ	อ่ = 1 ๆ = 1 ฯ = 5	ไ = 1 ฦ = 1 ๆ = 2 ฯ = 7 ๖ = 1
๐	๐ = 3 ๑ = 1	ฐ = 2 ๐ = 2 ๔ = 1
๑	ง = 1 จ = 1 ด = 5 ๐ = 1 ฯ = 1	ค = 2 ง = 4 จ = 2 ด = 5 ๐ = 9 ๐ = 1 ๓ = 1
๒	ษ = 1 ๐ = 1 เ = 1 ๖ = 1 ๘ = 1	ก = 1 ง = 1 ญ = 1 ฬ = 1 อิ = 2 เ = 1 โ = 1 ๕ = 1 ๖ = 7 ๙ = 1
๓	ค = 1 ฅ = 3 ต = 25 ท = 1 อั = 1 ฤ = 1 อำ = 1	ฅ = 2 ณ = 1 ด = 1 ต = 27 ถ = 1 ท = 1 พ = 1 ภ = 1
๔	ฮ = 1 อี๋ = 1 ๕ = 1 ๘ = 4	ช = 3 ศ = 2 ส = 3 ฬ = 2 อั = 1 อี๋ = 8 อี = 1 โ = 1 อ์ = 1 ๕ = 5 ๘ = 13 ๙ = 7
๕	ษ = 1 อ์ = 1 ๔ = 3 ๙ = 1	ช = 1 ฐ = 1 ฐ = 1 ร = 2 ศ = 3 ส = 8 ฬ = 1 ฮ = 2 ะ = 1 อี๋ = 1 อู = 1 โ = 1 ๆ = 1 ๔ = 3 ๖ = 1 ๘ = 1 ๙ = 11
๖	ฆ = 1 ว = 1 อึ = 1 เ = 2 อ่ = 1 ๒ = 3	ฐ = 1 ร = 1 ล = 3 ๐ = 2 ฮ = 2 เ = 2 ใ = 3 ไ = 2 โ = 1 ๅ = 2 อำ = 1 ๒ = 9 ๓ = 1
๗	อ๊ = 22 ๙ = 1	ซ = 1 ฆ = 1 ล = 1 ส = 1 อี๋ = 1 อ๊ = 28 ๔ = 3 ๘ = 4 ๙ = 4
๘	ช = 1 ณ = 1 ด = 1 ศ = 1 ส = 6 ฬ = 1 ฮ = 1 อี๋ = 32 ๔ = 1 ๙ = 1	ช = 3 ซ = 1 ญ = 1 ฟ = 1 ศ = 9 ส = 17 อี๋ = 41 ใ = 1 อ์ = 1 ๔ = 10 ๕ = 1 ๗ = 2 ๙ = 14
๙	ศ = 1 ส = 3 ห = 1 อ์ = 2 ๔ = 2 ๘ = 2	ศ = 1 ส = 4 ฬ = 2 อี๋ = 3 อ๊ = 1 อ์ = 4 ๔ = 2 ๗ = 1 ๘ = 2

Table 5.8 Confusion of character pair from zigzag stroke issue

Expected	Actual	BoH	CNN
ข	ฃ	25	11
ฃ	ข	5	23
ค	ฅ	26	27
ฅ	ค	13	44
ฆ	ม	51	51
ซ	ช	16	38
ฎ	ฏ	64	50
ฏ	ฎ	45	98
ฑ	ท	9	15
ท	ฑ	5	6

Table 5.9 Confusion of character pair from Head existence issue

Expected	Actual	BoH	CNN
ก	ถ	19	48
ก	ภ	1	19
ญ	ณ	7	2
อึ	อื	6	13
อื	อึ	8	8

Table 5.10 Confusion of character pair from head orientation issue

Expected	Actual	BoH	CNN
ค	ด	15	34
ศ	ต	13	51
ณ	ฌ	5	6
ด	ค	13	13
ถ	ภ	5	9
ฝ	ฟ	1	14
ฟ	ฝ	1	13
ภ	ถ	8	7
ฤ	ภ	15	34
ภ	ฤ	6	18
ๆ	ฯ	5	11
ฯ	ๆ	16	20

Table 5.11 Confusion of character pair from tail existence issue

Expected	Actual	BoH	CNN
ข	ช	7	18
ฃ	ซ	19	12
พ	ฟ	5	8
ฟ	พ	8	9
า	ไ	53	92
ไ	า	84	54

Table 5.12 Confusion of character pair from identical shape issue

Expected	Actual	BoH	CNN
ข	อู	95	67
บ	อู	10	21
อี๋	๘	32	25
อู	ข	37	62
อู	บ	21	33
อี๊	๗	19	37
๗	อี๊	22	28
๘	อี๋	32	41

Table 5.13 Confusion of character pair from other issues

Expected	Actual	BoH	CNN
ข	ธ	28	30
ข	อ้	24	22
ฃ	อู	38	21
ศ	๓	9	56
ฆ	ฌ	1	16
ง	ฎ	3	13
ง	๑	23	15
จ	อุ	25	25
ฉ	จ	2	22
ฉ	ล	5	18
ช	ธ	44	36
ช	ษ	17	4
ซ	ธ	16	15
ฌ	อิ	0	14
ฒ	ฆ	10	7
ฒ	ฌ	4	10
ต	๓	20	52

Table 5.13 Confusion of character pair from other issues (cont.)

Expected	Actual	BoH	CNN
ท	ต	0	10
ธ	ช	5	11
ป	ฝ	5	9
ผ	ย	0	10
ม	ผ	4	9
ย	บ	1	7
ร	ฮ	4	2
ล	จ	0	5
ว	ล	1	6
ศ	ส	6	13
ษ	ห	3	8
ส	ธ	2	11
อ	ล	2	17
ฮ	ส	0	12
ฮ	๕	0	16
ใ	โ	12	5
ใ	ไ	4	12
อ้	อั	6	7
อ์	เ	6	2
อ์	ธ	2	9
อำ	ฯ	5	7
๑	อ	1	9
๒	๖	1	7
๓	ต	25	27
๔	๘	4	13
๕	ธ	1	11
๖	๒	3	9

5.5 Discussion

The experimental results confirm our design principle: character shape recognition can be efficiently performed by using only one image feature that describes the relative position between its subcomponents. The feature shows the mutual information and discriminative feature even in the abstract level. The algorithm choose feature vector components by itself. If the information is too sparse, PCA manages to bring forth the useful components of it and leave the sparsity out. CNN on the other hand, is designed to extract feature from region of pixel. It cannot take the relationship of remote pixels as in DH, which is the strength and the weakness at the same time. CNN is more flexible in its application. Having self-arranged feature such as edge feature and gradient feature in the higher level allows CNN to recognize general images including real scenes for example. Designing BoH to be application-specific is a trade-off that we pay to gain the better performance.

We expect that higher number of components of PCA compression and NN hidden nodes give the better accuracy. The experimental result shows the consistent evidence to the expectation. However, if too many components or hidden nodes are provided, there is no more improvement of accuracy. For PCA issue, it can be explained in term of information provided in the PCA output. Since PCA arrange the transformation in the way that the most "significant" variables are moved to the first component, what is remaining in the tail components are noises or useless information for the classification. While keeping the first few components is enough for the coarse recognition task, more components provide more information for finer recognition. But too many components include noises and junks so that the accuracy cannot be improved any more.

For the thick and distorted dataset issue, BoH has done a good job on thick dataset because thick character still have about the same value of BoH component. Since a DH bag on thick and thin curve do not change size, proportion of area of foreground pixel in the bag is preserved. However, BoH is sensitive to curve distortion because distortion causes shifting in BoH component. The feature formation cannot put the DH from the distorted image in the same bag as that of the undistorted image. Having too many components in BoH even make the worse distortion affect to the accuracy because feature shifting is more likely to happen.

CHAPTER VI
CONCLUSION

In this research, we introduce a novel image feature called direction histogram (DH) and feature extraction algorithm called bag of histogram (BoH). In this concluding chapter, the design, implementation, and performance of the bag of histogram are summarized. We also add the critiques, suggestion, and future works for researchers that might be interested in our work.

6.1 Design

Bag of histogram is inspired by the well-known algorithm known as bag of word, which defines word as element in the object that we want to classify. Bag of word generates feature from counting the number of each word in the object with no need of ordering or grammatical rules, which is an efficient way to gather the feature from offline object.

We define an image feature namely “direction histogram” (DH) which describes the image pixel distribution in every direction around each pixel. A strength of DH is that it is a global feature. It takes every part of the image to the calculation, so it is able to capture the detailed structure of visual feature as well as the coarse structure of the image.

The observation that proves the feasibility of DH is that the various writing style of the same character class share the same DH set, while different character classes still have discriminable identity that can be captured in different DH set. This observation is the key to design feature extraction without specifying a fixed set of features. We instead represent images using neutral features and allow the classifier to automatically choose the useful features for discrimination.

Bag of histogram is designed to manage set of DH and form feature vector of each image. Bag of histogram starts the process by creating a set of bags that covers

every possible value of DH in the training set using the algorithm called first come-first add. Each bag consists of a DH center and a counter. When a new DH is calculated from a training image, it will be compared to every existing bag. If the exact bag or similar bag already exists in the BoH, the counter adds a mark to that bag. If not, BoH add the new DH to the bag collection.

Using the bag preparation as stated above, all DHs within the same range will be put into the same bag. Each bag in the BoH represents different character pixel zones. The idea that makes BoH works is that the same characters, although contain variations of shape, have about the same proportion of pixel in each analogous zone. Feature vector of an image is defined to be proportion of pixels that fits to each bag. Since different character comprises distinct set of pixel zones, feature vector from BoH can discriminate different shape between character classes.

6.2 Implementation

Bag of histogram is in the feature extraction module. To demonstrate the performance of BoH, we design a handwritten character recognition system using BoH as the feature extraction. The system consists of the preprocessing, feature extraction, and classification.

In the preprocessing step, an image is binarized and cropped into black and white, no white margin left on all four sides. In the feature extraction, the prepared BoH operates as a slot to count DHs that is digested from the preprocessed image. Feature vector is the number of mark counted in each slot, normalized by the number of pixel in the image. This feature vector is then compressed by the principal component analysis (PCA) transformation. The transformed feature vector will be truncated and transfered to the classifier. The feed-forward neural network that was trained by a bunch of feature vector is chosen to be the classifier with 80 output nodes corresponding to the 80 characters.

6.3 Performance

There are four parameters that we expected influencing the recognition accuracy: The bag dimension (D), DH different threshold (T), PCA compression size (P) and neural network hidden nodes (H).

It is tested that P and H do not cause any effect to the recognition accuracy. The best configuration of the remaining two parameters was investigated. The experiment shows that BoH with D = 40 and T= 0.05 gives the highest recognition accuracy at average 95.18 %.

The evaluation shows that the recognition system with BoH feature extraction performs better accuracy than the popular convolutional neural network, which is the state of the art of handwritten character recognition system, when tested with individual handwriting of 52 Thai peoples in the original, thick and distorted datasets.

The mixed handwriting test was investigated on the same collected data, but they were mixed together into one set. Mixed handwriting is created in two parts, training and testing set. The training set of the mixed handwriting is randomly picked up from the handwriting index 1-40, where the testing set is from the handwriting index 41-52. The performance of the classifier trained by feature vector from BoH of the mixed handwriting is 93.66% accuracy, which is, dropped around 1.5% from the average of the single handwriting test because the single classifier has to perform the more general task. However, it still performs better than the CNN that was trained by the same dataset, which perform 87.51% accuracy.

6.4 Further research

Local DH

This version of DH requires perfectly segmented character image. It is yet impractical for natural handwriting that might not be perfectly segmented into single character. A possibility to solve this weakness is to define local DH that is calculated from a neighborhood of the character image at a time so that the entire image does not

need to be segmented. This might lead to the more natural way of handwritten character recognition.

Multi-language recognition

Since the algorithm of DH and BoH feature extraction does not depend on any language specific feature, it is possible to use DH and BoH in other languages. Moreover, there is a possibility for DH and BoH to work on more than one language in the single system, i.e., multi-language character recognition, because there is no need for language selection step. The system can be trained to recognize mixed character set including two or more languages. If there are similar character shape across language, for example, “a” and “ล”, “b” and “เ”, the post-processing can use language redundancy to resolve the correct character classes.

Different distance function

In this research, we use Euclidean distance as the distance between any two DHs. There are a number of other distance functions that we can apply in Euclidean space, for example, L_p norms, and taxicab distance. We are able to add a non-linear function upon the calculated distance. These possibilities lead to many different ways to redefine DH similarity, which will change the characteristic of DH when used for OCR.

Less computational complexity

Calculating a set of DH out of a given image containing N pixels takes $O(N^2)$ time complexity because the algorithm needs to measure relative direction between any two pixels. Even we cannot reduce the order of complexity, we can reduce the process in each iteration. For example, the function used to measure relative direction is $\arctan\left(\frac{\Delta y}{\Delta X}\right)$, which is a transcendental function that need to be approximated by power series. We can replace it to $tan\theta = \frac{\Delta y}{\Delta X}$ directly and change the way to check the membership of relative direction in each direction interval. Moreover, the relative direction between each pair of pixel should be calculated just once and use twice in two DHs. For example, DH around p_1 need direction from p_1 to

p_2. After calculating this direction, the negative of this direction can be used again for DH around p_2 without recalculation.

REFERENCES

(2015, 02). Retrieved from Longman dictionary, OCR: http://www.ldoceonline.com /dictionary/OCR.

(2015, 02). Retrieved from Wikipedia, Optical Character Recognition: http://en. wikipedia.org/wiki/Optical_character_recognition

(2015, 02). Retrieved from Google Translate: https://play.google.com/store/apps /details?id=com.google.android.apps.translate&hl=th

(2015, 02). Retrieved from Reading Devices and Machines:http://www.adaptivetr.com /blindness/reading-devices-machines

(2015, 02). Retrieved from Thai OCR : Arnthai 2.5: http://www.nectec.or.th/img /index.php?option=com_content&task=view&id=25&Itemid=39&lang=en

Airphaiboon, S., Sangworasil, M., & Kondo, S. (1994). Off-line handwritten Thai characters from word script. Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on, (pp. 445-449).

Alahmadi, A., Joorabchi, A., & Mahdi, A. E. (2013). A new text representation scheme combining Bag-of-Words and Bag-of-Concepts approaches for automatic text classification. GCC Conference and Exhibition (GCC), 2013 7th IEEE, (pp. 108-113).

Budsayaplakorn, R., Asdornwised, W., & Jitapunkul, S. (2013). On-line Thai handwritten character recognition using hidden Markov model and fuzzy logic. Proceeding of the IEEE 13th Workshop on Neural Networks for Signal Processing, 2003. (NNSP'03).

Duda, R. O. (1972). Use of the Hough Transformation to Detect Lines and Curves in Pictures. Communication of the ACM, 15, 11–15.

Fink, G. (2008, 2015). Markov Models for Pattern Recognition from Theory to Applications. Springer-Verlag Berlin Heidelberg.

Graham, B. (2014, 09). Spatially-sparse convolutional neural networks. Computer Vision and Pattern Recognition.

Handel, P. W. (1993). United State Patent Office.

Harris, Z. (1954). Distributional structure. Word 10 (23), (pp. 146-162).

Imprasert, Y. (2009). Off-line Thai Handwritten Character Recognition Using Heuristic Rules and Neural Network. Thailand: Mahidol University.

Jang, B.-K. C. (1990, June). Analysis of thinning algorithms using mathematical morphology. IEEE Transactions on Pattern Analysis and Machine Intelligence, 12(6), 541-551.

Kobos, M. (2013). Retrieved June 1, 2014, from https://github.com/mkobos /pca_transform

Latha, J., & Devarajan, N. (2013). Feature extraction of handwritten numeric characters and recognition using an artificial neural network: anew approach. Journal of Vibration and Control, 1869-1876.

LeCun, Y., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE, (pp. 2278-2324).

Li, Z., & Feng, X. (2013, 10). Near Duplicate Image Detecting Algorithm based on Bag of Visual Word Model. Journal of Multimedia, 8(5), 557-564.

Limkonglap, U. (2006). Thai Handwritten Character Recognition System (THW-CR) Improving Feature Extraction Process by the Analysis of Contour Characteristics. Thailand: Mahidol University.

McCleary, J. (2006). A First Course in Topology, Continuity and Dimension. American Mathematical Society.

Methasate, I. S.-t. (2004). The clustering technique for Thai handwritten recognition . Ninth International Workshop on Frontiers in Handwriting Recognition (IWFHR-9), (pp. 450- 454).

Methasate, I., Marukatat, S., Sae-tang, S., & Theeramunkong, T. (2005). The feature combination technique for off-line Thai character recognition system. Proceedings of the 2005 Eight international conference on document analysis and recognition.

Mitrpanont, J., & Kiwprasopsak, S. (2002). The Development of the Feature Extraction Algorithms for Thai Handwritten Character Recognition System. In T. Hendtlass, Lecture Notes in Computer Science (pp. 536-546). Springer Berlin Heidelberg.

Mori, S. S. (1992). Historical review of OCR research and development. Proceedings of the IEEE, (pp. 1029-1058).

Nopsuwanchai, R. (2003). Discriminative training for HMM-based offline handwritten character recognition. Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on, (pp. 114-118).

O'Neill, M. (2006). Retrieved June 1, 2014, from [http://www.codeproject.com /Articles/16650/Neural-Network-for-Recognition-of-Handwritten-Digi].

Phokharatkul, P. K. (2002). Handwritten Thai Character Recognition Using Fourier Descriptors and Genetic Neural Networks. Computational Intelligence, 18(3), 270-293.

Phokharatkul, P., & Kimpan, C. (1998). Recognition of handprinted Thai characters using the cavity feature of character based on neural network. Circuits and Systems, 1998. IEEE APCCAS 1998. The 1998 IEEE Asia-Pacific Conference on, (pp. 149-152).

Saetang, S. (2011). A systematic study of offline recognition of Thai printed and handwritten characters. University of Southamton.

Sankhuangaw , K. (2005). Off-line handwritten character recognition using ant miner algorithm. Mahidol university.

Scholkopf, B. S. (2002). Learning with Kernel. MIT Press.

Shih, F. Y. (2010). Image processing and pattern recognition. New Jersey: John Wiley & Sons, Inc.

Soman, S. T., Nandigam, A., & Chakravarthy, V. S. (2013). An Efficient Multiclassifier System Based on Convolutional Neural Network for Offline Handwritten Telugu Character Recognition. 2013 National Conference on Communications. Delhi .

Tanprasert, C. K. (1996). Thai OCR: a neural network application. Proceedings of the 1996 IEEE TENCON. Digital Signal Processing Applications, (pp. 90-95).

Tauschek, G. (1935, 12 31). Patent No. US2026330 A. US.

Theeramunkong, T., & Wongtapan, C. (2005). Off-line isolated handwritten Thai OCR using island-based projection with n-gram model and hidden Markov models. Information Processing and Management: an International Journal - Special issue: An Asian digital libraries perspective, 41(1), 139 - 160 .

Thongkamwitoon, T., Asdornwised, W., & Aramvith, S. (2002). On-line Thai-English handwritten character recognition using distinctive features. Asia-Pacific Conference on Circuits and Systems (APCCAS '02), (pp. 259- 264).

W.D., W. (2007). A Beginner's Guide to Graph Theory (2 ed.). Birkhauser Boston.

Wang, C., & Huang, K. (2014, 12). How to Use Bag-of-Words Model Better for Image Classification. Image and Vision Computing.

Wikipedia, W. L. (2015, 02). Retrieved from http://en.wikipedia.org/wiki/Word_Lens

Xiong-wei, L., De-cai, H., Lu-ming, F., & Ai-jun, X. (2014, 2). An Image Classification Algorithm Based on Bag of Visual Words and Multi-kernel Learning . Journal of Multimedia, 9(2), 269-277.

Yuan, A., Bai, G., Jiao, L., & Liu, Y. (2012). Offline Handwritten English Character Recognition based on Convolutional Neural Network. 10th IAPR International Workshop on Document Analysis Systems, (pp. 125-129).

Zurada, J. M. (1992). Introduction to Artificial Neural System. PWS Publishing Company.

APPENDIX

Source code: BoH.java

```
import java.util.ArrayList;
import Shared.Coordinate;

public class BoH {
	int dim;
	double bagCreationThr;
	ArrayList<Histogram> bagList = new ArrayList<Histogram>();
	public BoH(int dim, double bagCreationThr) {
		this.dim = dim;
		this.bagCreationThr = bagCreationThr;
	}

	// add arrayList of foreground pixels to BoH
	public void addForeground(ArrayList<Coordinate> img) {
		double[] countH = new double[dim];
		System.out.println("image size"+ img.size());
		// create a histogram from foreground pixels
		for (int i = 0; i < img.size(); i++) {
			// coordinate of point i
			Coordinate ci = img.get(i);
			// reset countH for every new point i
			countH = new double[dim];

			for (int j = 0; j < img.size(); j++) {
				if (i != j) {
					Coordinate cj = img.get(j);
					// calculate direction from ci to cj
					double direction = Math.atan2(cj.y - ci.y, cj.x -
ci.x);
					int directionIndex = classifyDirection(direction);
```

```
                        countH[directionIndex] += 1;
                    }
                }

                // normalize countH into a histogram
                int sum = 0;
                for (int k = 0; k < dim; k++) {
                    sum += countH[k];
                }
                for (int m = 0; m < dim; m++) {
                    countH[m] = countH[m] / sum;
                }
                // add histogram into BoH
                Boolean addSuccess = false;
                for (int n = 0; n < bagList.size(); n++) {
                    if (bagList.get(n).addHistogram(countH)) {
                        addSuccess = true;
                        n = bagList.size();
                    }
                }

                // if countH can't fit any bag, create a new bag
                if (!addSuccess) {
                    Histogram h = new Histogram(dim, bagCreationThr);
                    h.initHistogram(countH);
                    bagList.add(h);
                }
                // System.out.println("end one point");
            }
        }

        public void prune(int pruneThreshold){
```

```
            for(int i=0; i<bagList.size(); i++){
                if(bagList.get(i).bagSize < pruneThreshold){
                    bagList.remove(i);
                    i-=1;
                }
            }
        }

        private int classifyDirection(double direction) {
            int directionIndex = (int) Math.floor((direction + Math.PI)
                    * dim / (2 * Math.PI))
                    % dim;

            // System.out.println(directionIndex);
            return directionIndex;
        }

        // get BoH size
        public int getBohSize() {
            System.out.println("Size of BoH: "+ bagList.size());
            return bagList.size();
        }

        public void printBoH() {
            for (int i = 0; i < bagList.size(); i++) {
                System.out.print("Size = " + bagList.get(i).bagSize + " ");
                for (int j = 0; j < dim; j++) {
                    System.out.print(bagList.get(i).h[j] + " ");
                }
                System.out.println("");
            }
        }
```

```
// create training/testing data from given img
public double[] getTrainingData(ArrayList<Coordinate> img) {
        double[] countBag = new double[bagList.size()];
        double[] countH = new double[dim];
        System.out.println("image size"+ img.size());
        for (int i = 0; i < img.size(); i++) {

                // reset countH for every new point i
                countH = new double[dim];

                // coordinate of point i
                Coordinate ci = img.get(i);
                for (int j = 0; j < img.size(); j++) {
                        if (i != j) {
                                Coordinate cj = img.get(j);
                                // calculate direction from ci to cj
                                double direction = Math.atan2(cj.y - ci.y, cj.x -
ci.x);
                                int directionIndex = classifyDirection(direction);
                                countH[directionIndex] += 1;
                        }
                }
                // normalize countH into a histogram
                int sum = 0;
                for (int k = 0; k < dim; k++) {
                        sum += countH[k];
                }
                for (int m = 0; m < dim; m++) {
                        countH[m] = countH[m] / sum;
                }
                // Match DH of this point into the closest bag
```

```
                    for (int j = 0; j < bagList.size(); j++) {
                        double distance = bagList.get(j).histogramDistance(
                                bagList.get(j).h, countH, 0.12);

                        if(distance < Double.POSITIVE_INFINITY){
                            countBag[j] += 1/ (double) (img.size());
                        }
                    }

                    // add 1 to bag[closestIndex]
                    //countBag[closestIndex] += 1 / (double) (img.size());
                }
                System.out.println("Direction histogram is created");
                return countBag;
        }
}
```

Source code: RunBoH.java

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.ArrayList;

import convolutional.Convolutional;

import pca_transform.CompressDataSet;

import Shared.Coordinate;
import Shared.Parameters;
import Shared.ReadImg;

public class RunBoH {
	int numChar = 80;
	int bohDim;
	double bagCreationThr;

	BoH boh;
	ReadImg readImage;
	int hwIndex;

	// static String subFolder;
	String subFolderName;
	CompressDataSet cd = new CompressDataSet();
	Parameters param = new Parameters();
```

```
	Convolutional cn;
	public static void main(String args[]) throws IOException {
		RunBoH runBoh = new RunBoH();
	}

	public RunBoH() throws IOException {
		readImage = new ReadImg();
		int[] hwIndexSet = {};
		int[] dimSet = { 40 };
		double[] thr1Set = { 0.05 };
		int[] pcaSizeSet = { 500 };
		int[] hiddenSizeSet = { 500 };
		// loop over every parameters
		for (int hwIndex = 1; hwIndex <= 52; hwIndex++) {
			for (int dim : dimSet) {
				for (double thr1 : thr1Set) {
					runTrainAndTestOnSet(hwIndex, dim, thr1, 0,
0);
					// buildEverything();
					String subFolderName = dim + "_" + thr1;
					cd.compress(hwIndex, dim, thr1,
subFolderName);
					for (int hiddenSize : hiddenSizeSet) {
						for (int pcaSize : pcaSizeSet) {
							param.setParameters(hwIndex,
							dim, thr1, pcaSize, hiddenSize);

							String trainFileName =
							param.currentFolder +
							param.pcaTrainFileName;
```

```
                                        String testFileName =
                                        param.currentFolder +
                                        param.pcaTestFileName;

                                        // find out how big is the input
                                        size
                                        int inputSize =
                                        inputSize(trainFileName);

                                        cn = new
                                        Convolutional(inputSize,
                                        hiddenSize, 80);

                                        test(cn, dim, thr1, pcaSize,
                                        hiddenSize, testFileName,
                                        param.currentFolder +
                                        param.confusionFileName);

                                        for (int i = 0; i < 35; i++) {


                                        cn.CalculateHessian(trainFileNam
                                    e);
                                        train(cn, dim, thr1, pcaSize,
                                        hiddenSize, trainFileName,
                                        testFileName, i);
                                        }
                                    }
                                }
                            }
                        }
                    }
```

```
}

public void runTrainAndTestOnSet(int hwIndex, int dim, double thr1, int
pcaSize, int hiddenSize) {
        param.setParameters(hwIndex, dim, thr1, pcaSize, hiddenSize);
        param.setFileNamesAll();
        cd.setParam(param);
        File createSubFolder = new File(param.currentFolder);
        createSubFolder.mkdir();
        this.hwIndex = hwIndex;
        bohDim = dim;
        bagCreationThr = thr1;
        boh = new BoH(bohDim, bagCreationThr);
}

public int inputSize(String fileName) throws IOException {
        FileReader freader = new FileReader(fileName);
        BufferedReader inputFile = new BufferedReader(freader);
        String str = inputFile.readLine();
        String[] tempString;
        tempString = str.split(",");
        // System.out.println(k+" "+j);
        double[] input = new double[tempString.length - numChar];
        return input.length;
}

public void createBoH(String fileName) throws IOException {
        ArrayList<Coordinate> img = new ArrayList<Coordinate>();
        // read an image file
        readImage.readBMP(fileName);

        // extract foreground pixel, put into an ArrayList
```

```
            for (int i = 0; i < readImage.binary.length; i++) {
                for (int j = 0; j < readImage.binary[0].length; j++) {

                    // put all black pixel into an ArrayList
                    if (readImage.binary[i][j]) {
                        img.add(new Coordinate(i, j));
                    }
                }
            }

            img = samplingImage(img);

            // put into BoH
            boh.addForeground(img);

            // show BoH we just created
            // boh.printBoH();
            boh.getBohSize();

    }

    public ArrayList<Coordinate> samplingImage(ArrayList<Coordinate> list) {
            // get size of the list
            int listSize = list.size();

            // set output size to 100 pixels
            int outputSize = 100;
            outputSize = (int) (listSize);
            // outputSize = min (100, list.size)
            outputSize = Math.min(outputSize, listSize);

            // uniform random index from list
```

```
        ArrayList<Coordinate> outputArray = new ArrayList<Coordinate>();

        for (int i = 0; i < outputSize; i++) {
            // random a number n where 0 <= n <= list.size()
            int n = (int) (Math.random() * (list.size() - 1));

            if (i % 2 == 0) {
                // get component n from list, and delete it from list
                Coordinate c = list.get(i);

                // add to output
                outputArray.add(new Coordinate(c.x, c.y));
            }
        }
        // generate output
        return outputArray;
    }

    public double[] createDataSet(String fileName) throws IOException {
        ArrayList<Coordinate> img = new ArrayList<Coordinate>();
        // read an image file
        readImage.readBMP(fileName);

        // extract foreground pixel, put into an ArrayList
        for (int i = 0; i < readImage.binary.length; i++) {
            for (int j = 0; j < readImage.binary[0].length; j++) {

                // put all black pixel into an ArrayList
                if (readImage.binary[i][j]) {
                    img.add(new Coordinate(i, j));
                }
            }
```

```
            }

            img = samplingImage(img);

            // get feature vector with label

            double[] output = boh.getTrainingData(img);
            System.out.println("dim of feature vector" + output.length);
            System.out.println("dim of boh" + boh.bagList.size());
            return output;
    }

    public void saveBoH() throws IOException {
            DecimalFormat df = new DecimalFormat("#.###");
            String fileNameBoH = param.currentFolder + param.bohFileName;

            FileWriter fwriter = new FileWriter(fileNameBoH);

            for (int i = 0; i < boh.getBohSize(); i++) {
                    // number of dh in this bag
                    fwriter.write(Integer.toString(boh.bagList.get(i).bagSize));
                    fwriter.write(",");
                    // write boh data for bag i
                    for (int j = 0; j < boh.dim; j++) {
                            fwriter.write(df.format(boh.bagList.get(i).h[j]));
                            fwriter.write(",");
                    }
                    fwriter.write("\r\n");
            }
            fwriter.close();
    }
```

```
public void buildEverything() throws IOException {
        int numTrainingSet = 30;
        // create BoH
        // Batch process for every datasets at the same time

        boh = new BoH(bohDim, bagCreationThr);
        for (int i = 1; i <= numTrainingSet; i++) {
                for (int j = 1; j <= 80; j++) {
                        String fileName = "charImages/set_" + i + "/char_" + j +
                        ".png";

                        fileName = "DistortedDataset/HW" + hwIndex + "/" +
                        "char_" + j + "_set_" + i + ".png";
                        createBoH(fileName);

                }
        }

        // prune small bags if number of bag >2000
        int pruneSize = 0;
        while (boh.bagList.size() > 2000) {
                pruneSize += 1;
                boh.prune(pruneSize);
        }

        saveBoH();

        DecimalFormat df = new DecimalFormat("#.###");
        String fileNameTrain = param.currentFolder + param.trainFileName;

        FileWriter fwriter = new FileWriter(fileNameTrain);
```

```
String fileNameTest = param.currentFolder + param.testFileName;
FileWriter fwriter2 = new FileWriter(fileNameTest);

// create dataset from BoH

for (int charIndex = 1; charIndex <= 80; charIndex++) {
        for (int setIndex = 1; setIndex <= 50; setIndex++) {
                String fileName = "DistortedDataset/HW" + hwIndex +
                "/" + "char_" + charIndex + "_set_" + setIndex
                                + ".png";
                double[] trainingData = createDataSet(fileName);

                if (setIndex <= numTrainingSet) {
                        System.out.println("add training data");
                        for (int i = 0; i < trainingData.length; i++) {
                                // System.out.println(trainingData[i]);
                                String data = (df.format(trainingData[i]));
                                fwriter.write(data);
                                fwriter.write(",");

                        }

                        for (int i = 0; i < 80; i++) {
                                if (i != (charIndex - 1)) {
                                        fwriter.write(Integer.toString(0));
                                } else {
                                        fwriter.write(Integer.toString(1));
                                }
                                fwriter.write(",");
                        }
                        fwriter.write("\r\n");
                } else {
```

```
                                System.out.println("add testing data");
                                for (int i = 0; i < trainingData.length; i++) {
                                        // System.out.println(trainingData[i]);
                                        String data = (df.format(trainingData[i]));
                                        fwriter2.write(data);
                                        fwriter2.write(",");
                                }

                                for (int i = 0; i < 80; i++) {
                                        if (i != (charIndex - 1)) {

fwriter2.write(Integer.toString(0));
                                        } else {

fwriter2.write(Integer.toString(1));
                                        }
                                        fwriter2.write(",");
                                }
                                fwriter2.write("\r\n");
                        }
                }
        }
        fwriter.close();
        fwriter2.close();
        boh = null;
}

public void train(Convolutional cn, int dim, double thr, int pcaSize, int
hiddenSize, String trainFileName,
                String testFileName, int epch) {
        try {
```

```
double dTotalMSE = 0;
double[] input;
double[] targetOutputVector = new double[numChar];
int i = 0;
FileReader freader = new FileReader(trainFileName);
BufferedReader inputFile = new BufferedReader(freader);
String str = inputFile.readLine();
String[] tempString;
int sizeInput;
while (str != null) {
        // for each row
        tempString = str.split(",");
        input = new double[tempString.length - numChar];
        targetOutputVector = new double[numChar];

        sizeInput = input.length;
        // for each column
        for (int j = 0; j < tempString.length - numChar; j++) {
                input[j] = Double.parseDouble(tempString[j]);
        }

        for (int j = 0; j < numChar; j++) {
                targetOutputVector[j] =
                Double.parseDouble(tempString[tempString.len
                gth + j - numChar]);

                if (targetOutputVector[j] > 0.5) {
                        // extract output argument into int
                        // idealOutput = j;
                } else {
                        // make 0 becomes -1
                        targetOutputVector[j] = -1;
```

```
                }
            }

            // // now train here
            double[][] memorizedNeuronOutputs = null;

            double dPatternMSE = 0.0;
            // do {
            dPatternMSE = 0.0;
            double[] actualOutputVector = cn.calculate(input);
            cn.backpropagate(actualOutputVector,
targetOutputVector,                    numChar,
memorizedNeuronOutputs);

            for (int ii = 0; ii < numChar; ++ii) {
                dPatternMSE += (actualOutputVector[ii] -
                targetOutputVector[ii])* (actualOutputVector[ii]
                - targetOutputVector[ii]);
            }
            dPatternMSE /= 2.0;

            dTotalMSE += dPatternMSE;

            System.out.println("Pattern " + i + ", mse = " +
                dPatternMSE);

            // // done training for this pattern

            System.out.println("");

            str = inputFile.readLine();
            i += 1;
```

```
                }
                cn.save(param.currentFolder + param.nnFileName);
                cn.load(param.currentFolder + param.nnFileName);
                test(cn, dim, thr, pcaSize, hiddenSize, testFileName,
                param.currentFolder + param.confusionFileName);

                cn.reduceEta();

        } catch (FileNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}

public void test(Convolutional cn, int dim, double thr, int pcaSize, int
hiddenSize, String testFileName,
                String confusionFileName) {
        try {
                double[] input;
                double[] targetOutputVector = new double[numChar];
                int i = 0;
                FileReader freader = new FileReader(testFileName);
                BufferedReader inputFile = new BufferedReader(freader);
                FileWriter fwriter2 = new FileWriter(confusionFileName);
                String str = inputFile.readLine();
                String[] tempString;
                // double[] output;
                int sizeInput = 500;
```

```
int countCorrect = 0;
while (str != null) {
        // for each row
        tempString = str.split(",");
        // System.out.println(k+" "+j);
        input = new double[tempString.length - numChar];
        targetOutputVector = new double[numChar];
        sizeInput = input.length;
        // for each column
        // System.out.println("Input");
        for (int j = 0; j < tempString.length - numChar; j++) {
                input[j] = Double.parseDouble(tempString[j]);
                // System.out.print(input[j] + ",");
        }

        // System.out.println("Output");
        int idealOutput = 0;

        for (int j = 0; j < numChar; j++) {
                targetOutputVector[j] =
                Double.parseDouble(tempString[tempString.len
                gth + j - numChar]);
                // System.out.print(targetOutputVector[j] + ",");

                if (targetOutputVector[j] > 0.5) {
                        // extract output argument into int
                        idealOutput = j;
                } else {
                        // make 0 becomes -1
                        targetOutputVector[j] = -1;
                }
        }
```

```
                        double[] actualOutputVector = cn.calculate(input);
                        // manipulate actualOutputVector to find max
                        int maxArgument = 0;
                        double maxOutput = -1;
                        for (int k = 0; k < actualOutputVector.length; k++) {

                                if (actualOutputVector[k] > maxOutput) {
                                        maxOutput = actualOutputVector[k];
                                        maxArgument = k;
                                }
                        }
                        // if actual output == ideal output, then countCorrect+=1
                        if (maxArgument == idealOutput) {
                                countCorrect += 1;
                        }
                        // print both ideal and actual output together in the file
                        fwriter2.write(idealOutput + "," + maxArgument +
"\r\n");
                        // System.out.println("maxAr");
                        str = inputFile.readLine();
                        i += 1;
                        System.out.println("reading line " + i + " from the file.
                                Size of line is " + tempString.length);
                }
                fwriter2.close();
                System.out.println(countCorrect);
                FileWriter fwriter = new FileWriter(param.currentFolder +
                param.accuracyFileName, true);
                fwriter.write(countCorrect + ",\r\n");
                fwriter.close();
        } catch (FileNotFoundException e) {
                // TODO Auto-generated catch block
```

```
                    e.printStackTrace();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
        }
}
```

BIOGRAPHY

NAME	Ekawat Chaowicharat
DATE OF BIRTH	7 December 1986
PLACE OF BIRTH	Chachoengsao, Thailand
INSTITUTIONS ATTENDED	Mahidol University, 2005-2009 Bachelor of Science (Mathematics) Mahidol University, 2009-2016 Doctor of Philosophy (Mathematics)
RESEARCH GRANTS	Development and Promotion of Science and Technology Talents Project (DPST)
HOME ADDRESS	3/18 Srisothorn Road, Tambon Namuang Muang District, Chachoengsao Province Thailand 24000 Tel. 081-298-5882 E-mail: ekawatchaow@gmail.com