LAMPUNG HANDWRITTEN CHARACTER RECOGNITION

Dissertation
zur Erlangung des Grades eines

D o k t o r s   d e r   N a t u r w i s s e n s c h a f t e n

der Universität Dortmund
am Fachbereich Informatik

von

AKMAL JUNAIDI

Dortmund

2016

Tag der mündlichen Prüfung: 19 October 2016

Dekan: Prof. Dr.-Ing. Gernot A. Fink

Gutachter:
Prof. Dr.-Ing. Gernot A. Fink
Prof. Dr. Heinrich Müller

ABSTRACT

Lampung script is a local script from Lampung province Indonesia. The script is a non-cursive script which is written from left to right. It consists of 20 characters. It also has 7 unique diacritics that can be put on top, bottom, or right of the character. Considering this position, the number of diacritics augments into 12 diacritics. This research is devoted to recognize Lampung characters along with diacritics. The research aim to attract more concern on this script especially from Indonesian researchers. Beside, it is also an endeavor to preserve the script from extinction. The work of recognition is administered by multi steps processing system the so called Lampung handwritten character recognition framework. It is started by a preprocessing of a document image as an input. In the preprocessing stage, the input should be distinguished between characters and diacritics. The character is classified by a multistage scheme. The first stage is to classify 18 character classes and the second stage is to classify special characters which consist of two components. The number of classes after the second stage classification becomes 20 class. The diacritic is classified into 7 classes. These diacritics should be associated to the characters to form compound characters. The association is performed in two steps. Firstly, the diacritic detects some characters nearby. The character with closest distance to that diacritic is selected as the association. This is completed until all diacritics get their characters. Since every diacritic already has one-to-one association to a character, the pivot element is switched to a character in the second step. Each character collects all its diacritics as a composition of the compound characters. This framework has been evaluated on Lampung dataset created and annotated during this work and is hosted at the Department of Computer Science, TU Dortmund, Germany. The proposed framework achieved 80.64% recognition rate on this data.

ACKNOWLEDGMENTS

I would like to express my deep gratitude to all the people who have supported me both professionally and personally during the development of this dissertation and without whom this work would not have been possible. First, I would like to thank my principal supervisor, Prof. Dr.-Ing. Gernot A. Fink, for his patient guidance, enthusiastic encouragement and useful critiques of this research work. I would also like to appreciate of his trust on the job of digitizing post cards of World War I, which is very convenient. Likewise, I thank Prof. Dr. Heinrich Müller as my co-examiner who always provided valuable advice for the research and dissertation improvements. I also thank Prof. Dr. Peter Buchholz and Dr.-Ing. Anas Toma for being the committee of my final defense. I also wish to thank Dr. Szilárd Vajda who always share some fresh ideas for enhancing the research quality and solutions when I found some obstacles during my technical and non-technical works.

My grateful thanks are also extended to my current and former colleagues in the research group Pattern Recognition in Embedded Systems, Dr.-Ing. Jan Richarz, Marius Hennecke, Leonard Rothacker, René Grzeszick, Axel Plinge, Irfan Ahmad, Sebastian Sudholt, Fabian Naße, Julian Kürby, and Eugen Rusakov for their valuable support and constructive recommendations in this work.

My thanks are also devoted to the administrative staff of our group, Claudia Graute, for all the help, from distributing office supplies and tools, the preparation of all administrative documents, up to guiding me on issues of my health insurance.

I would also like to thank the Directorate of Higher Education, Ministry of Education and Culture, Republic of Indonesia, for securing my scholarship, students of Grade 10^{th} and 11^{th} of the year $2010-2011$, SMKN 4 Bandar Lampung who have provided Lampung handwritten documents, and my bachelor students of the Mathematics Department and Computer Science Department in contribution of a new batch of Lampung handwritten documents.

A special thank goes to my "WG mates" a.k.a *mitbewohners* in Mülheim an der Ruhr, Joe, Reza, Vembi, Raffi, and Pak Iman, from whom I got encouragement during the writing of this dissertation. I would never forget the time together during watching movies, talking, shopping, cooking, eating, and all the fun we have had in the last two years. I greatly value the friendship of you guys.

In the work that spanned the years, the failures and sadness always appeared influencing my spirit. In this moment, the support of the family is really unchangeable. First and foremost, these were my parents and my parents in law who always devote their prayers for me. Likewise, my special thanks are dedicated to my beloved wife, Novilia, my brilliant sons Muhafiz Almas and Muhammad Fazli Haaziq, and my cute daughter Naureen Samara, who always shared the happy moments all the time. Thank you that you are all my inspiration in achieving this success.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

CC	Connected Component
CCs	Connected Components
DAR	Document Analysis and Recognition
EM	Expectation Maximization
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
LBP	Linear Binary Pattern
MST	Minimum Spanning Tree
NN	Neural Network
OCR	Optical Character Recognition
PCA	Principal Component Analysis
RBF	Radial Basis Function
SVM	Support Vector Machine
WR	Water Reservoir

1 INTRODUCTION

The invention of the writing system facilitated humanity to generate handwritten texts for storing and transmitting ideas or information. Handwritten text has been spread over the large geographical area spanning the whole part of continents. Some places have their own scripts while other places share the same script. The uniqueness of those scripts make them differentiable with each other. Some significant distinguished script can be observed among Roman, Cyrillic, Chinese, Kana/Kanji, Arabic, Devanagari, etc.

In the past, many handwritten manuscripts were created on a media like stones, leafs, woods, animal skins, animal bones, etc. Latterly the medium of handwriting was shifted to paper because it is more practical. These manuscripts were found in every part of the world as ancient relics. These pieces of old media with texts on it are considered as ancient relics which can reveal the history of mankind. Nowadays, it becomes a high concern to interpret ancient manuscripts for supporting historians.

Since a machine-print technology has been introduced, documents were not only written by hand but mostly produced by machines on paper. The production of text significantly increased because machines can perform a massive printing work in high speed process. As a result, the quality of text created by machine is more consistent in size and shape such that various types of printing media appeared for example books, magazines, newspapers, and others. However, printing machines can not replace handwriting completely. Therefore, the handwriting is still demanding because the usage of handwriting is simpler, more practical, and real time. In contrast to printing machine, it does need electrical power, machine operator, space for machines, etc.

With the huge activities in handwriting or printing texts, there is a need of automatic offline character recognition concerning those documents. The term offline character recognition refers to a recognition of text in which the text is captured from in a static digital image defined in a pixel or bit-map representation. In the 1960's, researchers started developing systems for analysis and recognition of the text in documents. The field is called Document Analysis and Recognition (DAR). The term DAR can be considered as one field of pattern recognition that focuses on the research of the document and its kind. This involves the analysis and recognition of texts and especially characters produced by humans or machines. With respect to this production, there are two major fields of research concern, the group of handwritten and printed character recognition. Both research directions have been developed for long time so that the state-of-the-arts of research in those two fields has been accelerating to advanced level.

The maturity of the research of offline handwritten character recognition is occurred to the scripts which are frequently used in the world like Roman, Chinese, and Arabic. Its success can improve many applications where large volumes of handwritten data are needed to be processed. For instance the recognition of addresses and postcodes or letters in a postal system, recognition of digits and handwriting on bank cheques, recognition of handwritten characters in form-filling sheet, are some of its application.

In contrary, a sparse-distributed script is less interesting to be explored because it will impact only to a few people in small area. It will automatically get no support in the research of handwritten character recognition if there is no awareness from researchers who are members of society that own the script. This is what has happened to Lampung script in Lampung province, Republic of Indonesia. Realizing this situation, the research that will be presented in this thesis can be considered as an initiative to increase the opportunity of Lampung script to be used as an object of research in DAR.

With the widespread of various approaches in handwritten character recognition nowadays, the research of Lampung handwritten character recognition can benefit from them. The existing approaches can be adopted or adapted for the purpose of Lampung handwritten character recognition.

1.1 OBJECTIVES AND MOTIVATIONS OF LAMPUNG HANDWRITTEN CHARACTER RESEARCH

Indonesians are familiar using Roman script because it is the official script of writing. However, some places in Indonesia like North Sumatera (Batak script), western part of South Sumatera (Rejang script), Lampung (Lampung script), West Java (Sundanese), Java (Hanacaraka), Bali (Balinese), South Sulawesi (Bugiesse) have their own scripts. Lampung script is not a popular ethnic script as other traditional Indonesian scripts like Javanese and Balinese. It is used by a limited number of natives in several areas in Lampung province. As other users of traditional scripts in Indonesia, the number of users of the Lampung script is small since the script was gradually abandoned since Roman script was introduced for writing.

With the decreasing number Lampung script users, the local government predicts that this script will be extinct in the near future. To raise more users, the local government have addressed the Lampung script as a lesson material of students in elementary and high school in Lampung province. The local government seems to start the effort in saving the script by educating the young students about the Lampung script. This is expected to increase the people who know and understand the script.

As another endeavor of preserving the Lampung script, the research of the Lampung characters had been introduced as an alternative support beside educating young students. The research aims to target the academic communities and to put the Lampung characters as the object of the research in the field of DAR. With the advances of computer technology, sooner or later the demand of recognition system for the non-cursive text like Lampung handwritten text will be more interesting and attractive. At this time, the Lampung handwritten text recognition has been initiated to get larger attention. It will drive many local or regional researchers to deal with the Lampung handwritten text into a broad scope of the research.

The motivations behind the research of Lampung handwritten character recognition are two-fold

- **Promotion of the Script**
  The research can be a way to start promoting the script into a larger area. By the research, dataset of Lampung script will be available to attract more researchers of the wider communities particularly from Indonesia.

- **The Heritage Preservation**
  The Lampung script is originated from the ancient Brahmi script [47] which is mostly used in India to establish the writing system of many languages in India. As part of the script inherited from an ancient script, the Lampung script now becomes a part of a cultural heritage of Lampung's society. This historical aspects can be considered an added value for conducting the research of Lampung script in the field of DAR.

The main objectives to be achieved from this research are to derive a framework of offline Lampung handwritten character recognition based on the general principle of handwritten character recognition. With a large set of developed approaches and methods in handwriting recognition research of Roman script, Chinese, Arabic, this research will exploit those approaches and methods to be a basis for the development of the framework. Through a complete chain of this framework, the Lampung handwritten character image can be transformed into a machine readable text.

This broad objective serves as a main focus of this research. It can be realized into some separated research goals as follows:

- To provide Lampung a dataset which can be freely downloaded for the purpose of the DAR research. A dataset is a basic resource to support research on handwritten character recognition. The datasets often become a big obstacle to start a handwritten character recognition because preparing them will require much time and cost. With the existence of such resources, researchers will have no initial barrier for conducting their research.

- To explore existing approaches on each milestone in a general handwritten character recognition to be adaptively applied for Lampung handwritten character recognition. The developed approaches will serve as a foundation of the framework that is explorable for further expansion.

- To investigate an approach which is capable to associate diacritics to a character. Lampung script is one example of a script with a rich set of diacritics. Although a main character can have no diacritic, most of the text documents contain diacritics. The diacritics in the text are important to change pronunciation of the basic character into a desired syllable. The presence of diacritics lets the research to study the coherence of diacritics toward a character during recognition. The big challenge is that the place and number of diacritics around the character are very sensitive to the composed syllable. Different position and number of diacritics will produce a different syllable. Therefore, determination of position and number of diacritics around a character must be done carefully.

1.2 RESEARCH METHODOLOGY

The research about Lampung handwritten character recognition is a big work which can not be conducted in one shot. It must be split into a set of tasks for the sake of simplicity and compatibility. Hence, this research can be broken into some sequential operations based on the basic methodology in the research of handwritten character recognition. These operations consist of four fundamental stages as follows:

1. Data preparation
   Handwritten documents are very crucial for conducting research of handwritten character recognition. Unfortunately, there is no former data of Lampung handwriting for the purpose of this research. Preparation consists of collecting, acquisition, and then storing the raw data. During the creation of this thesis, a dataset has been semi-supervised annotated and then visually checked for correctness of the annotation. To provide this data publicly available, the dataset is hosted on the website of Pattern Recognition in Embedded Systems Group, Department of Computer Science, TU-Dortmund, Germany [1].

2. Preprocessing
   After data preparation, some processes may be carried out in a consecutive order to transform the raw data into another form. Each process has a specific goal to prepare readiness of the data to some extent. These processes are the extraction of connected components, segmentation, noise reduction, labeling, categorization of connected components (into train, test and validation set), etc. Any other tasks may be involved in preprocessing if it is considered necessary.

3. Feature Extraction
   Feature representations are extracted from normalized connected components. The existing feature representations which have already been developed in other works for example chain-code, pixel densities, etc., or existing feature representations from other works that are adapted to Lampung script will be employed for Lampung handwritten character recognition. However, feature representations have to be carefully selected with a consideration that they can give a good impact in recognition.

4. Recognition
   Feature representations extracted from connected components are recognized to their classes by classifying them respectively based on their feature representations. Therefore, the role of a classifier is important such that each representation can be recognized correctly. Among many classifiers, NN and SVM are utilized in this research. Another statistical-based classifier like GMM is also applicable for recognition step.

5. Post Processing
   Post processing deals with injecting of an additional context into the processing chain during recognition. This is usually done after performing a complete

1 Available online on: http://patrec.cs.tu-dortmund.de/cms/en/home/Resources/index.html

recognition pipeline and evaluating the result. The aim of post processing is to improve the performance rate of the recognition or to provide a powerful approach to recognize more complex structures. The post processing of this research can be a context for assigning double-element characters and associating diacritics.

Another point of view that is also important in the field of DAR is about cursive and non cursive term. Both seem to create a dichotomy in the research, but they have the same level of importance. In case of the Roman-based text, the style of handwriting is most likely a cursive handwriting. Many researchers explore more on the cursive mode for handwritten recognition because the cursive text is more complex so it has more challenge rather than non-cursive text. In contrast to Roman-based script, Lampung script is not a cursive text but it still posses challenges in recognition. Beside to recognize Lampung characters, the recognition must also handle two other tasks, first recognizing diacritics and then associating them to the correct character.

1.3 OVERVIEW OF THE THESIS

This thesis mainly discusses about Lampung handwritten character recognition as a new challenging topic in DAR. Each chapter provides a comprehensive study of some aspects in Lampung script as well as some key components during the development of such a platform. The remaining thesis is organized as follows:

- Chapter 2 describes a simple document analysis pipeline which can also be regarded as a character recognition process, applicable either for machine-print or handwriting. In general there are five main processing steps in the pipeline,
  1. Image acquisition
  2. Preprocessing
  3. Segmentation
  4. Feature extraction
  5. Classification

  Each will be elaborated in a subsection. Although each part of this chapter only consists of a brief theoretical background, it covers the main methods and approaches in handwritten character recognition.

- Chapter 3 is focused on all information regarding the writing system of Lampung script. In the beginning, the utilization of Lampung script is explained and why its popularity is far behind the utilization of Roman script. Then, basic characters and their shapes are described. Lampung script also employs diacritics and those are explained in a separated subsection. As for the characters, this subsection describe the shape of diacritics and also their positions around a basic character. The last two parts contain discussions about punctuation marks and special attributes of Lampung script.

- Chapter 4 reviews related works which have strong relevance or are rather similar to main work of this research. Some of these works are the water reservoir approach for feature components which is recognized as a novelty approach in document analysis, identification of the writer based on diacritics only, recognition of some scripts where diacritics are part of these scripts, etc.
- Chapter 5 presents the core aspect of this research that develop Lampung handwritten character recognition. It is designed according to the process described in chapter 2. The topic encompasses from the preprocessing phase in the beginning, to the recognition phase at end. Each of them is explored in the context of Lampung handwritten character recognition. The discussion in this chapter also supplies some issues that were discovered during the work.
- Chapter 6 is devoted for data and facts of practical works. The source of the raw data is explained and how it is collected. The chapter also provides the result of each experiment proposed in chapter 5. The output of each approach is presented, discussed, and evaluated.
- Chapter 7 is the last chapter of the thesis summarizing all the works and discussing future directions and ideas for further research.

2 FOUNDATION OF A HANDWRITTEN CHARACTER RECOGNITION SYSTEM

Research and development of the first Optical Character Recognition (OCR) system has been introduced in early 1950s [8] with the introduction of a character reader device (scanner). In that period, the scanner processed documents slowly and was limited to one line at a time instead of the full page. With the development of technology, the OCR hardware was considerably improved. Nowadays, scanners, cameras, video recorders and any other optical devices are the common tools to capture data from a source with reasonable speed and capacity so that the output of those devices is more reliable for a recognition system.

In the domain of DAR, printed and handwritten text are the main object of the research. Unlike printed text, the challenge in handwritten text recognition is much higher due to fluctuation in the unconstrained handwritten style of the person who wrote the text. Some issues in this respect are the variation of skew angle, overlapping/touching lines, character size, variability of intra and inter-line.

The study of handwritten character recognition as a part of OCR research has grown progressively since the beginning of the 1990's [5]. Since that time, many publications in related journals focused on that field, for example International Journal of Document Analysis and Recognition (IJDAR), Pattern Analysis and Machine Intelligence (PAMI), etc. Furthermore, there was also an increasing demand of relevant workshops and conferences exploring various topics of handwritten character recognition. The high demand of these workshops and conferences indicates that the growth of handwriting recognition research is increasing as well.

Handwriting recognition is referred to a process of transformation a group of graphical marks of a particular language written on a spatial medium by hand into a set of defined symbols [46]. Most approaches of the character recognition refers to the traditional paradigm of pattern recognition [5]. It consists of a few stages which are data acquisition, feature extraction, and classification/recognition [30]. However in more refined paradigm, the framework may contain of more stages as can be seen in Fig. 1. Based on this framework, a handwritten text recognition system consists of several main stages:

- Image Acquisition
- Preprocessing
- Segmentation
- Feature Extraction
- Classification
- Post Processing

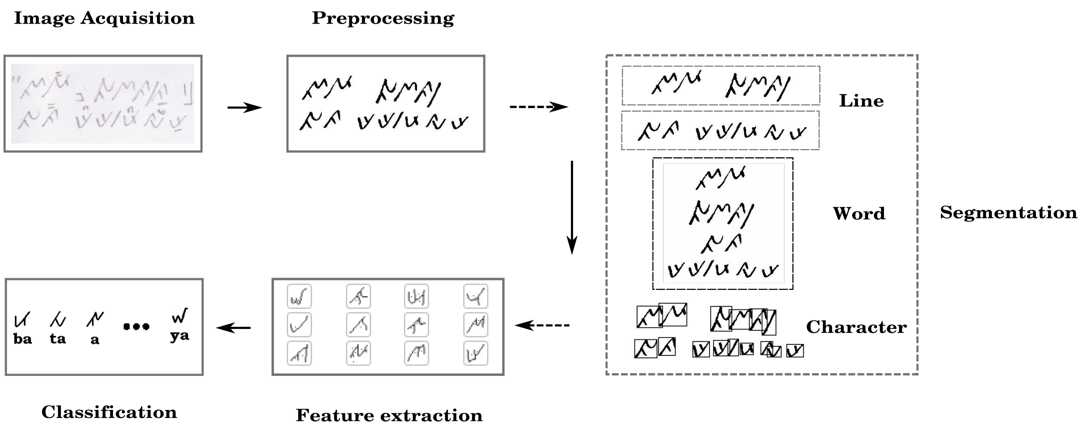


Figure 1: A simple document analysis processing. Each stage may consist of some sub-stages depending on the approach used within the stage. The segmentation stage is optional and it can be omitted for some circumstances.

This framework is basically inherited from the framework of the pattern recognition system. It is a sequential process which is not rigid to be refined. Each stage may be enhanced by sub-stages to commit some specific functions during processing. These sub-stages can be considered as an enrichment attempt to improve the performance of each stage. Beside multiple sub-stages, each stage may contain multiple options of algorithmic implementation [30]. This gives a freedom to researchers for selecting the best approach during accomplishment of the stage. To have an illustrative description of all stages, the following sections explain each stages in a sequential order.

2.1 IMAGE ACQUISITION

The first step in pipeline is the image acquisition. An image can be captured by an optical device via a sensor attached to the device. Based on the type of the sensor, there are three groups of image acquisition devices, with a single sensor, with sensor strips, and with sensor arrays [15].

A device with a single sensor has only one sensor that can scan the source document by moving the sensor head to the left and right before changing to next row by moving the sensor. Usually, this kind of device can produce high-resolution images with inexpensive costs because the mechanical motion can be controlled. However, since the device only has one sensor, the acquisition will be slow.

The second device employs many sensors that are set as an in-line strip. This strip is functioned as a receptor instrument of the device to capture the input image by moving the strip row-wised over the image source. A typical device which works in this manner is a flat bed scanner. Beside in-line arrangement, some devices have mounted the strip into a ring configuration. In this configuration, the output signal needs to be reconstructed by an algorithm to produce meaningful cross-sectional images. This is the basis of what so-called computerized axial tomography (CAT) imaging which is mainly used in the medical or industrial sectors.

The third class is an optical device using sensor arrays for image acquisition. Sensors are arranged inside the device in a certain rectangular dimension such that those sensors is run as an element of arrays. The dimension of the array can be in 4000 x 4000 elements or more. With this large array representation, the motion of the sensors during acquisition over the image source is not necessary because the array size will be large enough to cover the object. A digital camera is a typical device which operates these sensor arrays to produce images.

The use of the acquisition device depends on the target object. For example, if the acquisition is targeted to documents, a flat bed scanner is a good choice although a digital camera is also possible. While to capture landscape images and 3D objects, a digital camera will be an appropriate choice, whereas a scanner cannot do that.

The final concern in the sequence of image acquisition is compression format of the output acquisition. Whatever the device, the image must be represented in a format that preserves the original information of the image. The format using a lossy image compression must be avoided since it will degrade information of the image which will impact the performance of other stages hereinafter.

2.2 PREPROCESSING

A preliminary step in handwritten character recognition is called preprocessing. It is a series of operations performed after the image acquisition in order to achieve a certain level of image quality. The raw image of the handwritten document in this step is transformed into an intermediary image which minimizes variabilities that are not important for its recognition so that useful features can be extracted and the recognition is improved. According [2], the goals of preprocessing in general are:

1. Minimize the noise
2. Normalize the data
3. Compress the data

To accomplish these goals, preprocessing steps may involve a number of sub tasks such as noise removal, slant estimation and normalization, size normalization, binarization, thinning, etc. Since there are no specific standard methods for preprocessing, they will differ from one system to another system. Some of those methods are needed in one system while others are flexible to be used, depending on initial physical judgment of the documents that have to be processed or a prior knowledge of the data. Moreover, some of those sub tasks may be carried out simultaneously by a scheme or some may be overlapped with each other.

The following subsections describe some of those tasks that are often used during preprocessing, i.e. noise removal, normalization, and binarization. For additional schemes, the reader can explore further methods in references, like in [2], [8], [15].

2.2.1 *Noise Removal*

The quality of handwritten document input will affect the performance of handwritten character recognition outcome at the end of the process-chain. A bad quality handwriting will generate a low recognition rate but in contrast, a good quality handwritten source will achieve a better accuracy rate. In fact, the image enhancement by noise minimization of the input image or document would always be an advantage for the whole recognition system, whether or not the input image or document is in a bad quality.

The noise removal is necessary since many factors can degrade and distort handwritten documents. Some of degradation and distortion on documents may be caused by the quality of paper, aging of documents, quality of ink etc. which unintentionally generate artifacts in document images. This can produce imperfection in document images which are considered as a noise. The second type of noise that may be introduced is due to reproduction and transmission of image during its acquisition process by the hardware. The first type of noise is called high level noise while the second type that is a side effect of the acquisition hardware is called low level noise [16].

The methods to reduce the noise can be divided into three major groups, filtering, morphological operations, and noise modeling [2]. Filtering is done by performing a convolution between a filter mask (a convolution kernel) and the image to specify a value to a corresponding input pixel as a function of the values of its surrounding pixels. During convolution process, the mask will be moved like a sliding window from pixel to pixel on the image. For each pixel, a corresponding value will be calculated as a sum product between the mask and appropriate pixel with its neighborhood pixels. While morphological operations are a similar mechanism but the role of convolution is done by logical operators. Typical operators in this regard are adopted from the main operator in set theory i.e. AND, OR, and NOT [15]. Therefore, the morphological operation with operator AND and OR can only occur between two images in binary format. Whereas the operator NOT can be executed on a single binary image.

The last method, –a noise modeling–, portrays a different approach compared to the two previous schemes where their operations are explicitly applied to document images. The noise modeling scheme does not seem to be a direct approach for tackling the noise. In this regard, the noise is estimated by a mathematical formulation and with the help of this model, the image is improved. Some noise models are represented by probability density functions and are discussed in [15]. However, building a noise model does not always succeed. In many cases of handwritten character recognition applications, it is impossible to model the noise as noted in [2].

The noise removal process can be performed by a smoothing operation which is one of the filtering approaches. The idea is that the image is blurred to reduce the sharpness of the image as the random noise is indicated by sharp transitions in gray levels. This particularly impacts in reducing the noise. Nonetheless, at the same time smoothing has side effects. It will moderately diminish the detail of edges on image which is undesirable since edges are the most desired features. Another effect of smoothing is that it can bridge the gaps of a broken line or fill the empty spot. Bridging the gaps or filling the empty spot may be either desired or undesired features for the purpose of recognition. Therefore, the smoothing process must be considered before it is applied in the sequence of preprocessing. For more details, readers can refer to the noise removal in [15].

Some other methods of noise removal have been developed for specific noises such as a clutter, a large black area in binary image around document image which is dominantly generated during acquisition process like the scanning or photocopying. Some examples are a massive copier border produced during photocopying, output of scanning process between the gap of gutter and scanner, or output of scanning process due to different illuminations between paper edges and scanner bed. The other causes of a clutter are ink seeps, ink blobs, or punched holes. All those are considerably large compared to a text image.

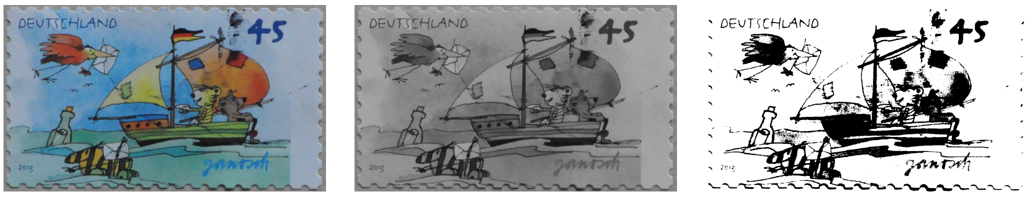
One simple approach to deal with this kind of noise was proposed in [51]. In this work, the removal process is targeted to large black borders of image documents. The approach uses projection profiles to estimate the location of massive black borders and cut them leaving only the text part. Initially, an image document must be binarized to get a binary image. Then, smoothing is executed along with horizontal and vertical direction by applying a smear method, Run Length Smoothing Algorithm/RLSA [24]. With threshold 4 pixel, the algorithm can fill 4 white pixel hole around foreground pixel into fully black pixel. Based on this result, the projection is calculated for horizontal and vertical direction. The massive black borders will be detected if the histogram is significantly large for several consecutive horizontal or vertical pixel. In this situation, this border will be cut. Another approach to get rid of clutter can be observed in [1]. Various other approaches are briefly reviewed in [13].

A more recent method of the noise removal can be found in [16]. In this work, the noise removal and recognition are combined as a single optimization problem and latent variables are incorporated into optimization process to store a priori knowledge of the noise. This optimization problem is then solved by employing Expectation Maximization (EM) algorithm in order to find the values of those variables. However, the usage of latent variables will impact on a longer processing time when the initial guess of these variables is not good enough. To accelerate the convergence solution, the initialization as well as improvement of those variables is estimated by fuzzy inference systems. The advantage of this method is a reduction in convergence time of the algorithm. Moreover, the applicability of the method is not only for French documents but it also flexible to other documents like Spanish, English, Arabic, etc., with no or little adaptation in the fuzzy inference systems.

2.2.2 *Binarization*

Within the preprocessing stage, it is often necessary to perform binarization that converts a raw image into black and white each for the object and background respectively. The goal of binarization is to sharpen the object as foreground against its background. The mechanism behind the normalization is a threshold value that becomes a parameter to group the pixel into either as foreground or background.

With respect to the threshold, binarization techniques can be distinguished between the global and the local binarization. The global binarization algorithms uses a single threshold value which is calculated based on the heuristics or statistical attribute of the entire image and then applied to the entire image. In contrast to global binarization, the local technique uses the neighborhood pixels attributes to compute the threshold and applies it only to the pixel where it was computed.



(a) color image (b) gray-scale image (c) binary image

Figure 2: An example of color image of a German stamp and its conversion to gray-scale and binary image.

To transform a raw image into binary image, the color raw image firstly has to be converted into grayscale and then continued by a binarization algorithm. Several algorithms that are most common used algorithm for binarization are Otsu [38], Niblack [37], and Sauvola [49]. They will be explained briefly in the following.

- **Otsu Algorithm**
  Otsu algorithm is one global technique for binarization. The threshold value is computed such that the sum of foreground and background spreads at its minimum. With another word the algorithm should compute the threshold that minimizes interclass variance.

$$\sigma^2 = \omega_b \cdot \sigma_b^2 + \omega_f \cdot \sigma_f^2 \tag{2.1}$$

Where:
ω_b is the weight of background pixel, which is computed as the probability of background pixel.
ω_f is the weight of foreground pixel, which is computed as the probability of foreground pixel.
σ_b^2 is the variance of background pixel.
σ_f^2 is the variance of foreground pixel.

However, in practice, the computation of the σ_b^2 and σ_f^2 is relatively slow. To handle this situation, the algorithm can simply use the mean without changing the decision by shifting the term *minimizing interclass variance* by *maximizing between class variance*. Then the formula 2.1 is adjusted as the following.

$$\sigma^2 = \omega_b \cdot \omega_f \cdot (\mu_b - \mu_f)^2 \tag{2.2}$$

Where:
μ_b is the mean of of background pixel.
μ_f is the mean of of foreground pixel.

The advantage of Otsu algorithm is quick processing because it works directly to the gray scale image. While the drawback is the poor result if it is applied on the image with unbalanced object against its background.

- **Niblack Algorithm**
  Niblack algorithm is a locally adaptive binarization method that computes the threshold value based on a local region on the image. The region chosen should be small enough to conserve the local attributes and at the same time should be large enough to remove the noise. The threshold computation moves over regions within the image like a sliding window. Then the mean and standard deviation are calculated for each local region with the center (x, y) and size $w \times w$. The threshold for this center is computed by the formula:

  $$T(x, y) = m(x, y) + k \cdot s(x, y) \tag{2.3}$$

  Where:
  $m(x, y)$ is the mean of the region with the center (x, y).
  $s(x, y)$ is the standard deviation of the region with the center (x, y).
  k is a user predefined constant which generally set to a negative value.

  This algorithm performs well to distinguish the text region as foreground. But in the meantime the algorithm can also generate an extreme amount of noise anywhere else, particularly whenever the background part contains light texture to some extent, such as gray zone, light spot, etc.

- **Sauvola Algorithm**
  Sauvola's algorithm [49] operates similar to the Niblack algorithm but with a little modification to handle the problem of Niblack algorithm. The value of k is still a fixed number, but it is set to a positive constant. In addition, the computation is modified such that it behaves more dynamically with respect to the each region. The former Niblack algorithm formula (equation 2.3) changes to be a new one as follows:

  $$T(x, y) = m(x, y) \cdot \left[1 + k \cdot \left(\frac{s(x, y)}{R} - 1\right)\right] \tag{2.4}$$

  Where:
  R is the dynamic range of standard deviation.
  k is a constant.

  By this new formula, the contribution of standard deviation becomes stronger in determining the threshold but more adaptive at the same time. The $m(x, y)$ coefficient in the new formula will downscale the threshold which in fact can diminish the noise produced on the background area by Niblack algorithm. Their experiment indicated that the optimum result was achieved with R set to 128 for 8-bit gray level images and k set to 0.5.

2.2.3 *Character Normalization*

As variability of handwritten characters is erratic in shape and size even handwritten characters from a single person, it can strongly distress the performance rate of the recognition process. To have a standard size, the character images need to be transformed such that all such character image instances are represented in the same size. This process is called a normalization. There are several normalization type such as skew normalization, slant normalization, and size normalization. The difference of them lay in the target of the normalization. The two first types will be briefly explained in the next two paragraphs and the latter will be covered more detail in the rest of this subsection.

The skew normalization is performed toward the baseline of handwriting. The baseline condition may be tilted during scanning of the document image. Another fact that can be found on the document image is the curve of the baseline. It is the nature that without line guides, human handwriting may turn up or down so the handwriting baseline may fluctuate. The task of detecting and correcting them can be accomplished by the skew normalization. A brief review of skew detection and correction can be observed in [13].

The slant normalization refers to the process of returning of characters to upright position. The tendency of most writers obliquely write their handwriting a little to right side. Hence characters will make a small angle between characters and the vertical direction. In this regard, a slant correction which is another term for the slant normalization need to be carried out. To get an basic illustration of the slant normalization, the reader can refer to [13].

The size normalization is also called the character normalization. The purpose of the character normalization is to reduce the arbitrary shape variation of character images by adjusting the original size into a predefined size, mostly into the same size of height and width. There are some functions to carry out a normalization. In a broad outlook, the normalization strategy can be distinguished into three categories [28]. These categories are grouped based on boundary alignment (conventional linear and nonlinear normalization), centroid alignment (moment normalization) and curve fitting (combination of both).

In practice, a pixel of original image will be mapped to normalization image by a certain function. For example, a simple mapping function of linear normalization can be formed by the ratio of respected size dimensionality between the normalized and original image. Let $f(x, y)$ denotes the original image with width W_1 and height H_1 respectively and $g(x', y')$ denotes the normalized image with width W_2 and height H_2 respectively, the transformation of an original image coordinate (x, y) to a normalized coordinate (x', y') can be done by forward mapping and backward mapping as follows:

$$x' = \alpha x \qquad y' = \beta y \tag{2.5}$$

and

$$x = x'/\alpha \qquad y = y'/\beta \tag{2.6}$$

where α and β represent transformation ratios, given by
$\alpha = W_2/W_1 \quad \beta = H_2/H_1$

An example of the normalization based on linear function is given in Fig. 3. In this example, an image of a Lampung character in a different width and height is normalized into three different copies each in a square size. The samples indicate that in those three normalized images, the basic shape of the original image is maintained similar to original one.

Beside a linear function applied to normalization, the method for normalization can also be either in non linear function or moment function. However, a comprehensive discussion of those methods beyond the scope of this research. The interested readers can refer to [8] and [30] to expose the detail of non linear or moment function normalization.

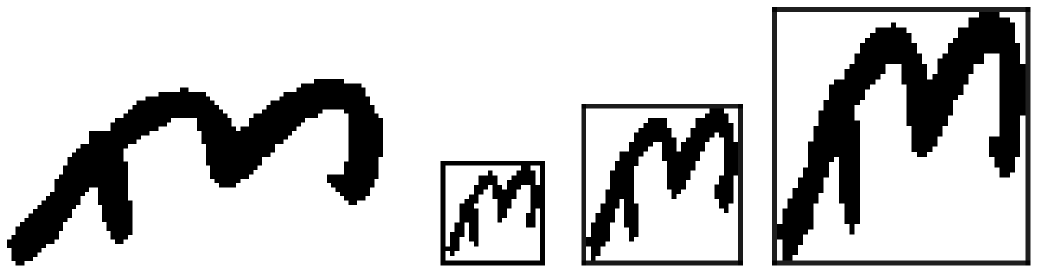

(a) original size 87x43 (b) normalized 20x20 (c) normalized 32x32 (d) normalized 48x48

Figure 3: Binary image of the Lampung character *Ja* in its original size and several normalized size.

As the normalization is executed based on the size of the image, a normalization process may be performed by a stretching or shrinking on width and height of character images. This does not only have a requisite impact but also introduce some negative effects like degradation of the shape, unbalance aspect ratio of the character, shifting the proper slant, etc.

In order to cope with these problems, some techniques have been developed. One of technique to solve those conventional problems is called aspect ratio adaptive normalization (ARAN) [29]. This technique controls the aspect ratio of the normalized image as a continuous function of aspect ratio of original image. Therefore, the original image aspect ratio is preserved into normalized image. In applying this strategy, the size of normalization image is not fixed but adaptively calculated based on the aspect ratio of original image via aspect ratio mapping function. If $W_1 < H_1$, H_2 has a fixed standard height whereas W_2 is centered and scaled according to aspect ratio of original image. In contrast, when $H_1 < W_1$, W_2 has a fixed standard width whereas H_2 is centered and scaled according to aspect ratio of original image.

Another technique is performing normalization by an ensemble process [28]. An example of this technique as presented in [28], fourteen basic normalization functions are chosen to build an ensemble normalization architecture and then doubled to twenty eight by switching on/off slant correction. From those outputs features are extracted and feed into a classifier. A decision combiner is employed at the end of pipeline to determine the class. To reduce the complexity of the normalization ensemble, a subset selection of the classifier is applied during the combination.

2.3 SEGMENTATION

Segmentation is a technique to decompose a document image into sub-images of individual symbols of a certain unit. In document analysis, the unit can be a paragraph which means segmenting the document image into units of paragraphs or a line segmentation which means extracting units of text lines, or on the level CCs where the output of the segmentation are units of connected pixels. Fig. 4 provides an example of the segmentation process toward a Lampung handwritten document. In this example, output of segmentation indicates a few lines text containing CCs of Lampung characters as well as the diacritics attached to them.

In the following subsections, the topic of the line and CCs segmentation will be covered concisely. In the first subsection, several approaches of the line segmentation are explained in general to provide a basic idea of the segmentation. While in the subsequent part, the segmentation in the level CCs is portrayed in brief.

2.3.1 *Line Segmentation*

Line segmentation can be considered as a middle stage process before segmenting the smaller units like words or characters. This means that the segmentation of the words or characters relies on the line segmentation because it keeps track of sequence of words and/or characters for each line. However not all text recognition process will perform word or character segmentation after a line segmentation, especially for cursive script. Beside, the usage of the classifier also clarifies whether the words and/or characters segmentation need to be done after the line segmentation or not. Even some techniques may require segmentation into unit smaller than characters.

Basically, there are three remarkable approaches for line segmentation. The approaches are the projection profiles, smear method, and Hough transform. Some other general, modified, or hybrid methods also exist although they are not so prominent. A few of them are the repulsive attractive network [39], or the minimum spanning tree [61], etc.

Projection profiles explores the line based on the total foreground pixel of the document image. The measurement is determined by counting the number of black pixel for each horizontal row. The counting will find the peaks and valleys of the foreground pixel. The peak indicates the massive foreground pixel which potentially represents a baseline while the valley is the blank space between the line.

The projection profiles is the easiest method to be implemented. However it is susceptible to curvilinear or oblique of text lines. Moreover, if the handwriting document contains such a thing, the projection may contain inconsistent peaks and valleys. It will consequently generate an incorrect segmentation. Touching and overlapping handwriting will also influence the performance of the projection profiles approach.

The smear method segments the lines by exploiting the local aggregate. In general it consists of two steps. First, each pixel in the row is scanned to localize two consecutive foreground pixels. Over those two consecutive pixels, the distance is measured. If the distance is less than a given threshold, the area between those two pixel must be switched to foreground. By this way, pixels are computed as a connected blob of foregrounds. However this task only computes the unconnected spot of baseline. Therefore the next step is to concatenate the bunch such that baseline can be generated completely.

The drawback of the smear method is that it is less robust to curved and large skew lines. Another shortcoming occurs whenever the document contains touching or overlapping lines. As the impact, those lines will be grouped together as one line although it apparently should be two different lines.

As its name, Hough transform performs the line segmentation by employing a transformation scheme. The image point in a Cartesian coordinate system is transformed into a polar coordinate system. The generic scheme of Hough transform works as follows. The coordinate of the points of edge segments (x_i, y_i) is used as a parameter to calculate a new parameters (r, θ) with $r > 0$ and $0 \leqslant \theta \leqslant 2\pi$. The transformation is done as follows. A tangent line is made through this point. Then distance from the line to the origin is measured and represented by r. The angle θ is the angle between the horizontal axes and the tangent line. The line equation that passes through this point can be represented by:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right) \cdot x + \left(\frac{r}{\sin\theta}\right)$$

The formula can be rearranged as:

$$r_i = x_i \cdot \cos\theta + y_i \cdot \sin\theta \tag{2.7}$$

For all the points along the pixels segment, the transformation is done which in fact generates many (r, θ) parameters. To detect the line, this approach uses what is called an accumulator which graphically is a representation of all (r, θ) in the Hough space. All the points along the pixels segment that forms a collinear line will have a peak point in the Hough space. With a threshold for the accumulator, a set of pixels points can be detected as a nearly collinear for the purpose of line segmentation.

2.3.2 *Connected Components (CCs)*

CC is defined as regions of adjacent pixels that have the same input value or label [53]. A set S of pixel is a CC if there is at least one path in S that joins every pair pixel (p,q) of pixel in S. The joining pair of pixels in the set can be ruled by a connectivity criteria to its neighbor pixels. There are two common connectivities for the neighbor i.e. 4-connectivity or 8-connectivity. The 4-connectivity rule determines a pixel connected if the neighbor pixel residing in one of four major compass points, i.e. north, south, west, and east. While 8-connectivity rule determines that two pixel connected if one pixel located exactly at the surrounding of the other pixel which is one of 8 closest side positions.

The CCs can be in any meaning depending on the sort of text in the document. In the cursive handwritten document, the isolation process into CC yields connected pixel as a single word, while in the printed document, the process yield the characters. As there is no perfect document, among generated CCs, there may also be unintended objects like noise specks, groups of touching rows, parts of a character that is broken, diacritics, etc.

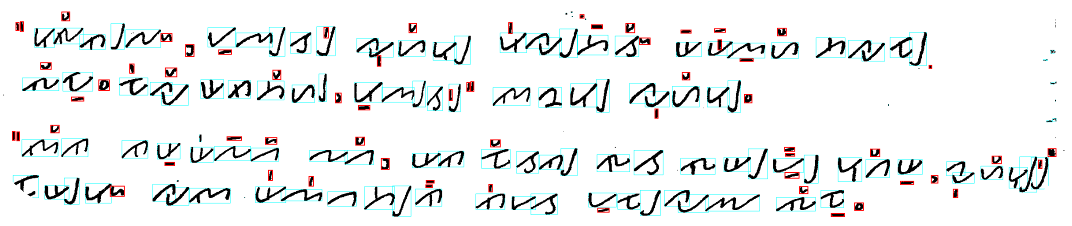


Figure 4: CCs of characters vs non-characters. CCs of characters are surrounded by cyan bounding boxes. Some of cyan boxes also contain unknown marks or noise like some on the right side. The small mark in red boxes indicate the CC of non-character symbols such as diacritics, unknown marks like double vertical strips in the beginning both sentences, or punctuation marks at the end of both sentences.

For particular situations, the CC can be directly passed through the next step in the handwritten character recognition pipeline, but in other different situations probably the CC still needs a further treatment before being processed by the next step.

To extract CCs from a binary image, two most common algorithms can be applied. The first algorithm is called one-pass algorithm which sometimes is also called flood-fill algorithm. In general, the algorithm extracts one CC at a time and continues to another CC until all CC in an image are completely extracted. The process is started by locating a foreground pixel of a component. This pixel is regarded as a seed point during extraction. Then from this point, neighbor pixels are traversed one by one to search connected foreground pixel based on the neighbor connectivity definition. If a connected foreground pixel found, it is labeled as the same label as the seed point. If it is not foreground, the process continues to another pixel around the current position until no other connected pixels found. This process is restarted by addressing the next seed point from another CC.

Beside one-pass algorithm, the extraction of CCs can also be extracted by the two-pass algorithm. This algorithm is easier to be implemented. As the name implies, this algorithm extracts CCs in two main steps which should be done consecutively. Both are illustrated in the following.

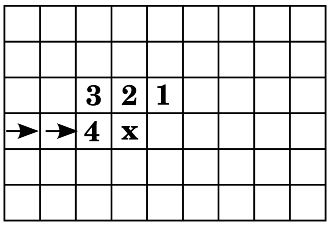
1. The first step is temporarily assigning the label for each pixel in the image. This labeling is started from the first pixel on the upper left and traversed to the next pixel on the right of current pixel in the first row. It scans the pixel from the left to the right and then continues to the next row until reaching the last pixel in the last row. During this step, a label is assigned to foreground pixels. To have a concise illustration, the pseudo code 1 explains the process of labeling in this first pass.

   Assume a single step in the first pass process is indicated in Fig. 5. During checking the label of neighbors in the first pass, there is a possibility that labels of neighbors can be more than one label. This causes a problem in decision of label to be set on the current pixel. To deal with this problem, the current pixel is labeled based on the pixel with the lowest label. Meanwhile, the structure of encountered neighbors depends on the definition of connectivity, either 4-connectivity or 8-connectivity. Let the current pixel is marked as a symbol

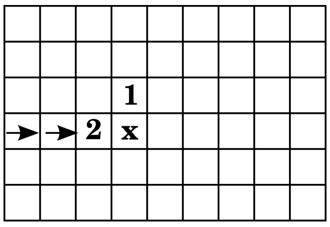
Algorithm 1 Labeling of the First Pass

```
1: scan pixel by pixel
2: while remaining pixel do
3:   if foreground pixel is found then
4:     check its direct neighbors
5:     if neighbor had been labeled then
6:       if All neighbors have a same label then
7:         assign foreground with this label
8:       else
9:         assign foreground with the lowest label
10:      end if
11:    else
12:      assign new label
13:    end if
14:  end if
15: end while
```

x. Referring to neighbors that already traversed and labeled around x, the number of neighbors of the current pixel in configuration of 8-connectivity is four pixels. Those neighbors are three pixels on the top and one pixel on the left of the current pixel (Fig. 5a). While in the structure of 4-connectivity, neighbors are only two pixel which are one on the top and one on the left of the current pixel (Fig. 5b).



(a) The composition of neighbor pixel in 8-connectivity



(b) The composition of neighbor pixel in 4-connectivity

Figure 5: The shape structure of encountered neighbors during checking of neighbors in the first pass of Connected Components (CCs) extraction.

2. As the first step only assigns labels temporarily, the second step must ensure that temporary labels for each same blob must have a single label. Each single blob that has been identified and labeled from the first step will be reset to the lowest label inherited from labels within that blob. This new label is mapped to all pixels belong to the blob as a final label. This relabeling process is done for every blob in the image until all blobs are completely reset.

2.4 FEATURE EXTRACTION

Features are the measurements or attributes extracted from image that are used for training (learning) and classifying this image into classes. This means that a feature can be regarded as a representation of the image itself. The process to generate features is called feature extraction. In this process, the input patterns of an image are mapped onto points in a feature space.

The role of features in a handwritten character recognition framework is very important because it will give a big impact in the overall performance of the recognition. Therefore, the algorithm to extract the feature from image should produce features which can group all images of the same class together while at the same time they can discriminate the images in the different classes. In addition, the algorithm should also be easily computable [20].

There are various features of handwritten character extracted by researchers. Most of those features can be grouped into two major types [22], [58], [32]:

- Statistical feature.
  The statistical feature is the feature that is generated as statistical measurements of the image or regions of the image [32]. The statistical measurements of this feature are usually derived from distributions (of the image attributes) like pixel point. Features in this category include pixel densities, projection profiles, histograms of chain code directions, image intensity, etc. A sample of projection profiles in a horizontal direction can be seen in Fig. 6. Beside in a horizontal direction, the projection profiles can also be extracted toward a vertical direction. The choice of both options is usually depend on the purpose of the projection profiles itself. The most application of the projection profiles, particularly across a horizontal direction, is to help a task of line extraction in handwritten character recognition.

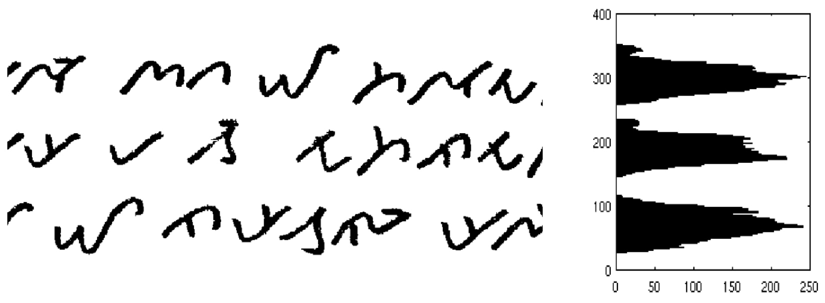


Figure 6: A projection profiles of the Lampung handwritten character text in horizontal direction.

- Structural feature.
  The structural feature of handwriting is in general reflecting intuitive aspects of writing. It can be generated from topological and geometrical properties of the character image. This means that it can be various elements such as maxima and minima, ascenders, descenders, cross points, branch points, end

points, dots, aspect ratio, loops, strokes and its direction, etc. The example of end points and branch points are given in the Fig. 7. In term of the graph theory, an vertex or node is called end point if its degree is one. A branch point is a vertex or node with the degree three and a cross point is a vertex or node which has degree more than three.

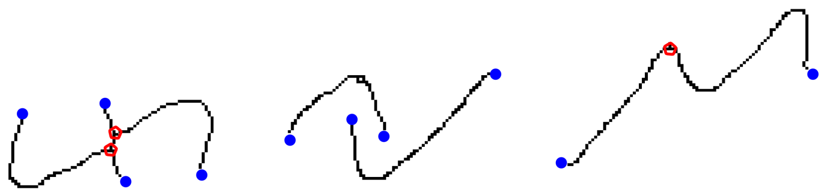


Figure 7: A sample of end points and branch point of the Lampung handwritten character text. The blue dots indicate end points while the red pentagons indicate branch points.

The crucial issue during feature extraction is how to generate an efficient feature representation. This issue is rather difficult to be solved since the features will much more depend on the image source that will not be identical case by case. Some feature extraction algorithms will generate the best suited features for one problem but those algorithms probably will not be appropriate for other problems. Here the a prior knowledge of the character image (candidate) will be meaningful before performing a feature extraction.

Another problem in handling the features is reducing dimensionality size of the feature vector. The feature extraction algorithms mostly extract big feature vectors as the impact of the operation on the pixel level. The big dimension of the feature vector is not always good for the handwritten character recognition framework. The bigger size of the feature vector, the longer time that will be needed to perform training and recognition. Therefore, it is an advantage to utilize a smaller dimensionality vector for representing the feature vectors. The way to reduce dimensionality is by transforming feature vectors to be less in dimensionality but at the same time still preserve the underlying structure of the original feature data. One approach to perform this task is the widely used technique called Principal Component Analysis (PCA). The reduction in PCA is done by transformation the basis of original data points into a few orthogonal linear combinations, which is called principle components (PCs), with the maximum variance. However, the maximum variance is no guarantee that the data point in a new space contains appropriate discriminative vectors. The new data space is built by a set of new orthogonal basis with less dimensionality such that all original data points can be represented with minimum loss of information. A detail description of PCA can be found in [4], [10].

2.5 CLASSIFICATION

As the final goal of a recognition process is assigning the class label, there is a task to group primitive candidates of an image to a predefined class based on their identical matching feature patterns. The group of candidates with identical feature patterns lay in the same class since this similarity represent characteristic of the same group. Then there is also a task to determine unknown input feature patterns into group class members. These processes in the context of a handwritten character recognition is called a classification. In general, the classification is defined as a work of deciding extracted feature patterns of primitive candidates from an image into one of a given set of classes.

The classification is performed by a classifier that uses a particular approach. This classifier is necessarily trained according to the presence of samples with predefined classes. This process where the samples are involved is called the classifier training.

The training phase aims at learning the nature of all character classes. Throughout this phase, the classifier will inspect all possible classes via feature patterns and collect some attributes or signatures that are owned by each class. Once the training phase succeeded, the classifier noted and recorded the feature patterns of each character class, which suppose to be unique for each class. However in a worst case, the feature patterns might be incomplete whenever the training process does not find some particular classes in the samples.

If the training phase has been completed to learn feature patterns of each class in samples, another phase that is called testing is taken into account to tackle the input features of unknown classes. The process is as follows. The classifier receives the input features, and then the classifier identifies and verifies these input features based on the information acquiring from the training phase. The classifier assigns the class for each input feature during the testing phase.

Concerning the type, Authors in [54] divide classifiers into the Bayes-based, linear, and non-linear classifier. However from a more general view point, the classifiers can be roughly distinguished into two groups based on the training approach. Both are the non-discriminative (statistical classifier) and discriminative classifier [27].

The key point of the non-discriminative classifier is the involvement of the statistical theory particularly the Bayes decision theory. In the non-discriminative group, the classifier initially inspects all the input features of the samples during training process. Then the classifier generates the model for each class from the training samples. In order to get a representative model, the classifier estimates parameters of classes in the samples by calculating underlying probabilities, for example, a posterior probability by Bayes formula, of the model when the training process takes place. The built model is then used to identify and verify a set of unknown input features into a class in the group. The classifier decides to which class the given input belongs. Several classifiers in this category are Gaussian Mixture Model [10] which can be used with a Bayes classifier and Hidden Markov Model (HMM) [11].

In contrast, the discriminative approach does not build a model for class determination of unknown input features. Instead of generating an explicit model for each character class, the classifier composes the decision boundary of each class from training samples during the training process. This boundary then becomes the basis to directly map the unknown input features into one of the class label. This approach does not depend on the probability counting from the training samples. The classifiers of this category are NN and SVM [4], [10] [8], [54].

The next subsections only discuss an overview of NN and SVM. Each will describe a brief introduction to the principle behind the approach as well as some important aspects within the concept.

2.5.1 *Neural Network*

The idea of the NN classifier adopts the work of the human brain system. The brain can accept some input signals simultaneously and then process them into an output as a specific information. The representing system of this occurrence can be seen as a simple neuronal network sketched in Fig. 8, which, in the simple case, represents the model of a binary classification problem.

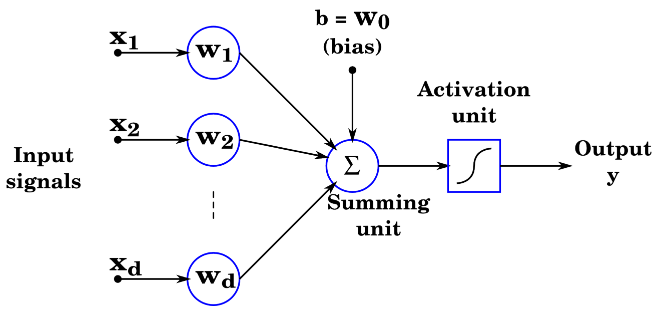


Figure 8: A basic neuronal model consists of three elements, synapses, summing unit, and activation unit. This simple model denotes a single layer neural network with a single output where the value of this output can classify inputs into a class among a limited number of classes.

In this basic form, a single neuron composes a simple neural network that models a single layer with single output as a three elements processing system. It comprises several synapses as input sensors, a summing unit, an activation function and one output. All input signals from synapses are processed by a summing unit after multiplying respectively by particular weights. The role of the summing unit is as a combiner of all weighted inputs from synapses such that it yields one scalar value which is called the net activation. However, the value of this net activation cannot be directly used as a criteria for classification of the input signal. To adjust this net activation being ready as a basis of a classification decision, another transformation must be carried out to control amplitude of this net activation value. The transformation is generated by an activation function which is suitably chosen based on the distribution of the target values. Hence, the final output of the network will depend on the selected activation function. For example if the network is dedicated for the system of a two classes classification problem, which can be modeled as the network in Fig. 8, the output value can be mapped to a zero or one. The output "zero" means that the input signal represents the first class, otherwise it represents the other class.

Suppose a feature representing the input signal consists of d numerical measurements $\{x_1, x_2, \dots, x_d\}$. This incoming input is accepted by the network via input unit and multiplied by particular weight w_i. The output of summing unit for this simple model is formulated as a weighted combination of incoming inputs by the following formula:

$$v = \sum_{i=1}^{d} w_i \cdot x_i + b = \sum_{i=1}^{d} w_i \cdot x_i + w_0 \cdot x_0 = \sum_{i=0}^{d} w_i \cdot x_i = \mathbf{w^t x} \tag{2.8}$$

where:
v is a net activation
w_i is the weight of input component i
$b = w_0 \cdot x_0$ is a bias which can be considered as an input with a fixed signal $x_0 = 1$
x is an input vector x with dimension (d+1)

This net unit value is fed into activation unit to delimit the output within a certain bounded amplitude. In this circumstance, the activation function, denoted by $\sigma(.)$, can be either linear or non linear. For a simple linear function, one of the possibility is using identity function as follows:

$$g(a) = \sigma(a) = a$$

Despite a linear function, the activation function can also be set to a non linear function. This function is selected to accommodate non linear inputs so that the function can manage a non-linear behavior. One simple example in this category is a piecewise-constant function that is constructed by two discrete values based on a certain threshold. This function is indeed employed to handle the classification problem with two outputs (binary). The function is in form,

$$g(a) = \begin{cases} 1, & \sigma(a) > t \\ 0, & \sigma(a) \leqslant t \end{cases}$$

This piecewise-constant function can be approximated and smoothed by a sigmoid function. The mathematical formula for sigmoid function is

$$g(a) = \sigma(a) = \frac{1}{1 + e^{-a}}$$

This sigmoid function is also common for activation function. The usage of a sigmoid function as the activation function encompasses many benefits. First, it is a non linear function so the network is capable to simulate a non linearity behavior of the input. The function also has a minimum and maximum output value so that the network can keep weights and activations bounded. Another advantage is that the function is differentiable which enables gradient learning during training of the classifier. The goal of this learning is to improve weight parameters such that the optimal net activation value can be estimated for each iteration. Thus at the end of iteration, a best net activation value can be obtained. With all those prominent properties, the usage of sigmoid function for activation function will obtain those advantages.

2.5.1.1 *Single Layer Neural Network*

In a realistic scenario, the multiclass classification problems seem to dominate over the binary classification problems. The model in Fig. 8 can be extended to handle multiple outputs as representation of multiple classification problems. In fact, the structure for new network becomes more complex than a simple model that was introduced before. The new network can be constructed by augmenting the only one output of the simple neuronal model in Fig. 8 to be multiple outputs as sketched in Fig. 9. This new system is still a single layer network but having multiple outputs as representation of multiple classes. All weighted input elements are combined to generate values of each possible outputs. Consequently, all inputs have a certain contribution in generating the value of outputs.

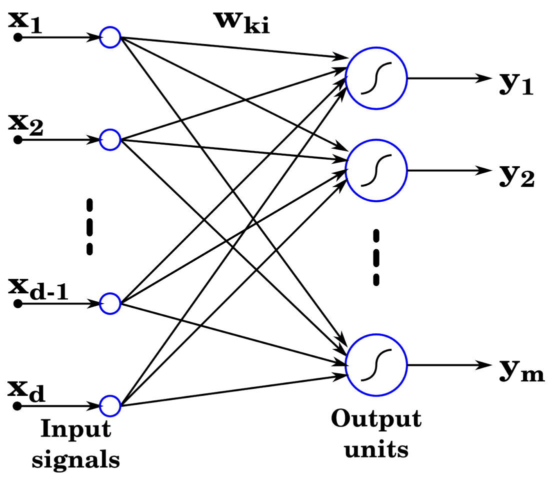


Figure 9: The model of a single layer neural network with multiple outputs. The single layer refers to the output layer which one and the only layer in the network. Multiple outputs indicate that the network serves as a processor of the input to assign one class among multiple classes possibility

Recall the equation 2.8, a corresponding formulation of the function for multiple outputs network is given as the following,

$$y_k(x) = \nu_k = \sum_{i=1}^{d} w_{ki} \cdot x_i + b = \sum_{i=1}^{d} w_{ki} \cdot x_i + w_{k0} \cdot x_0 = \sum_{i=0}^{d} w_{ki} \cdot x_i = \mathbf{w}_k^t \mathbf{x} \quad (2.9)$$

where:
y_k is a net activation delivered to output unit of k-th
$b = w_{k0} \cdot x_0$ is bias which can also be regarded as an input at fixed signal $x_0 = 1$

This value is fed into an activation function $\sigma(.)$ that is purposely selected to suit the criteria of the target. The net activation is then transformed to another scalar according to this function such that

$$g(y_k) = \sigma(y_k)$$

will fit to a certain distribution in a bounded scalar value as output of the network. The classification is eventually decided based on this final output.

Note that the term output unit is also equivalent to output layer. That is the term layer may replace the term unit and vice versa. They are exchangeable with each other but the term "layer" is much more popular. Therefore the term layer frequently appears in many discussions about NN.

2.5.1.2 *Multilayer Neural Network*

Considering a single layer neural network, the ability of the network to handle input features from arbitrary sample is rather limited. Disregard of its learning algorithm for counting weights, a single layer neural network always isolates any two classes via a linear hyperplane decision boundary [8]. The problems appear if the samples contain classes with a complicated distribution for which the decision boundary is most likely non linear. In fact, the samples are not linearly separable and a single layer neural network will be unable to separate classes of those samples.

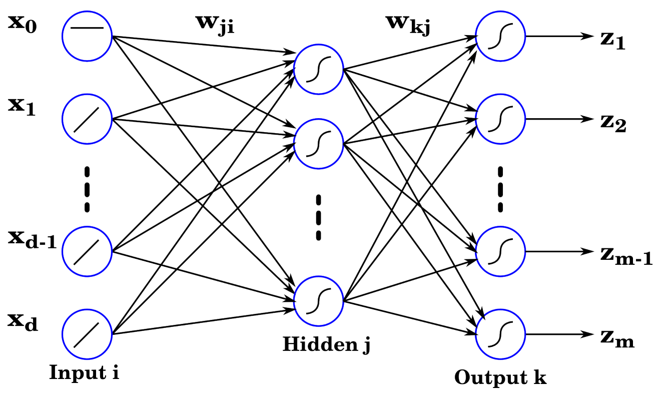


Figure 10: Multi layers neural network composed by three layers with multiple outputs. The layer between input layer and output layer is called hidden layer.

This problem can be definitely handled by employing non linear functions for representing pattern features. Thus the network can deliver a linear combination of a non linear function as the function of its original features inputs. However, to probe the weights in a more dynamical way, more layers can be added at preceding output layer. Note that the input unit is considered as an individual layer that is called input layer. In fact, additional layers are placed between input layer and output layer.

These additional layers upgrade the network to be a multilayer neural network and enhance the network operation to be more powerful for handling such a complicated sample. The explanation behind this fact is that a multilayer neural network is able to perform simple algorithms to learn non linearity of the training sample [10]. Hence, the usage of multilayer network can adequately afford non linearity of the sample.

As these additional layers laid between input and output layer, their existence look hidden from external view. Thus, the layers located between input and output layer are called hidden layers. The example of multi layer neural network modeled by three layers as an input layer, a hidden layer, and an output layer is shown in Fig. 10.

Hidden layers provide a more flexible way to generate a better net activation value for a subsequent layer. These hidden layers compute the weighted sum of its inputs signal by using a particular function. The function can be either adaptive or predefined. Suppose a three layer neural network in Fig. 10. The net activation value of input layer can be computed as the following:

$$a_j = \sum_{i=1}^{d} w_{ji} \cdot x_i + b = \sum_{i=1}^{d} w_{ji} \cdot x_i + w_{j0} \cdot x_0 = \sum_{i=0}^{d} w_{ji} \cdot x_i = \mathbf{w}_j^t \mathbf{x} \qquad (2.10)$$

where:
a_j is a net activation from input layer on output of j-th
w_{ji} is a weight of the input-to-hidden layer
$b = w_{j0} \cdot x_0$ is a bias which can be considered as an input with a fixed signal $x_0 = 1$

Each net activation generated from input layer is transformed by a differentiable, non linear activation function $f(.)$ to deliver output to hidden layer j:

$$y_j = f(a_j) \qquad (2.11)$$

Since these net activations fed into hidden layer, the next computation is a repetition of the same process as the beginning. Each net activation from input layer is linearly combined in hidden layer to produce net activation value of hidden layer:

$$a_k = \sum_{j=1}^{m} w_{kj} \cdot y_j + b = \sum_{j=1}^{m} w_{kj} \cdot y_j + w_{k0} \cdot y_0 = \sum_{j=0}^{m} w_{kj} \cdot y_j = \mathbf{w}_k^t \mathbf{y} \qquad (2.12)$$

The activation function to be used in output layer can be the same function as in hidden layer or a different function. Suppose the activation function on the output layer is $g(.)$. The final output of the network can be constructed by the following formula:

$$z_k = g(a_k) \qquad (2.13)$$

The output z_k can be indeed thought as the function of the input feature vector x. The overall network function after substitution a_k and a_j is given by

$$h_k(\mathbf{x}) \equiv z_k = g\left(\sum_{j=0}^{m} w_{kj} \cdot y_j\right) = g\left(\sum_{j=0}^{m} w_{kj} \cdot f\left(\sum_{i=0}^{d} w_{ji} \cdot x_i\right)\right) \qquad (2.14)$$

The neural network computes the output as a linear discriminant measurement. If there are c output units, the network computes c discriminant functions $z_k = h_k(\mathbf{x})$. The input is then classified based on the output from which discriminant function is maximum.

Note that a three layers neural network with a hidden layer between input and output layer is able to approximate any function from input to output [10]. A hidden layer play an important role during training since it provides an extended medium to two primary existing layers which accordingly extend the learning capability and capacity of such a network. Through hidden layer, the network accept incoming inputs to compose arbitrary linear combinations of input components from training data and sufficiently transfer them to output unit. However, the number of hidden neurons within hidden layer must be an important concern during designing of the network. Too many hidden neurons can impact the system to be over specified but in contrast, too few number of hidden neurons can potentially reduce network performance in fitting the input data into a representative model.

2.5.1.3 *Network Training*

Training of neural network aims to learn of weights according to input patterns with assigned labels on them such that the corresponding layer can generate the optimum net activation value. Technically speaking, the network learns weights based on inputs of the training sample that are iteratively accepted by the network and corresponding outputs as the response afterward. During training, the network performs a corrective procedure which is called *backpropagation* to optimized computation outcome. In this procedure, the weights are adjusted and updated each time the inputs come and outputs obtained.

In the beginning of the training, weights need to be initialized to guarantee that the training step will continuously move forward to the next iteration. However, the initialization by zeros will yield output values of the current layer to zeros which in fact will set the error to be zero as well. The problem of this situation is that the error will not impose the change of weights which is not desired. Therefore, initial weights are set to random values to ensure that each iteration continued with proper weights.

Assume the input $\mathbf{x}_n$ represents the feature vectors of the training sample with $n = 1, \dots, N$ indicate indexes of all data in training sample and desired output t_n represents corresponding target vectors. Suppose the output z_k is generated on output layer after an input sample x_n executed by the network. The difference between output z_k and target t_k is regarded as an error. The error function of the network for $\mathbf{x}^n$ can be computed as the total squared difference between the output z_k and the target t_k as written in the following,

$$E = \frac{1}{2} \sum_{k=1}^{M} (z_k - t_k)^2 \tag{2.15}$$

When the training is carried out, all data in training sample are passed through the network and adjustments of weights are iteratively made to reduce error. In this respect, the process to evaluate minimum error value is done by a procedure that is called *gradient descent* (see [4], [10]). On each new iteration p, weights of output unit will be adjusted while reducing the error in the same time by:

$$\Delta \boldsymbol{w}_{kj}(p) = -\eta \frac{\partial \mathbf{E}}{\partial \boldsymbol{w}} \tag{2.16}$$

where η is learning rate which controls the relative size of weight and bias changes during learning.

The main focus of Eq. 2.16 is to compute $\partial E/\partial \boldsymbol{w}$. This factor can be re-written in component form as $\partial E/\partial w_{kj}$ to refer the derivative of error with respect to weight of layer j to layer k. Due to E is not explicitly dependent to w_{kj}, evaluation of this factor must consider the error function E as a function of a_k and a_k as a function of w_{kj}. Thus, the differentiation can be derived by chain rule,

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial w_{kj}} \tag{2.17}$$

The factor $\partial E/\partial a_k$ indicates the change of error over net activation of unit k. This is called sensitivity of unit k and defined as

$$\delta_k = \frac{\partial E}{\partial a_k} \tag{2.18}$$

However sensitivity δ_k is not explicitly dependent to net activation a_k. Therefore, the differentiation can be examined by chain rule by regarding $\partial E/\partial a_k$ as a multiplication of $\partial E/\partial z_k$ and $\partial z_k/\partial a_k$. Then the overall multiplication can be solved by differentiation of Eq. 2.15 and Eq. 2.13 and the result is,

$$\delta_k = \frac{\partial E}{\partial a_k} = \frac{\partial E}{\partial z_k}\frac{\partial z_k}{\partial a_k} = (z_k - t_k)g'(a_k) \tag{2.19}$$

The formula of this sensitivity shows that the activation function g(.) is necessarily differentiable to enable backpropagation running properly.

Back to the main concern of Eq. 2.17, the only remaining part is the last factor $\partial a_k/\partial w_{kj}$ which can be obtained from Eq. 2.12. The evaluation of $\partial a_k/\partial w_{kj}$ yields y_j. Thereby adjustment rate of weights is given by

$$\Delta \boldsymbol{w}(p) = -\eta\delta_k y_j = -\eta(z_k - t_k)g'(a_k)y_j \tag{2.20}$$

Likewise $\partial E/\partial w_{kj}$, the term $\partial E/\partial w_{ji}$ occurred on hidden layer will be evaluated through a resemble process,

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} \tag{2.21}$$

The factor $\partial E/\partial a_j$ is considered as sensitivity of unit j on hidden layer. Since the error function E is not directly derivable over the a_j, the derivation of sensitivity $\partial E/\partial a_j$ must be solved by using chain rule as follows,

$$\delta_j = \frac{\partial E}{\partial a_j} = \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial a_j} \tag{2.22}$$

The first element of this sensitivity indicates that the differential of E must be evaluated with respect to y_j. The process to derive the solution of this part is as follows,

$$\begin{aligned}\frac{\partial E}{\partial y_j} &= \frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{M}(z_k - t_k)^2\right] \\ &= \sum_{k=1}^{M}(z_k - t_k)\frac{\partial z_k}{\partial y_j} \\ &= \sum_{k=1}^{M}(z_k - t_k)\frac{\partial z_k}{\partial a_k}\frac{\partial a_k}{\partial y_j} \\ &= \sum_{k=1}^{M}(z_k - t_k)g'(a_k)w_{kj}\end{aligned} \tag{2.23}$$

Since $\delta_k = (z_k - t_k)g'(a_k)$ as indicated in Eq. 2.19, the result of Eq. 2.23 can be written as the following

$$\frac{\partial E}{\partial y_j} = \sum_{k=1}^{M}\delta_k w_{kj} \tag{2.24}$$

The second part of Eq. 2.21 is $\partial y_j/\partial a_j$ which can be differentiated from Eq. 2.11 as $f'(a_j)$. Therefore, the sensitivity of the unit j is given as follows

$$\begin{aligned}\delta_j &= \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial a_j} \\ &= \left[\sum_{k=1}^{M}\delta_k w_{kj}\right] f'(a_j)\end{aligned} \tag{2.25}$$

To complete evaluation of Eq. 2.21, the last part $\partial a_j/\partial w_{ji}$ is solved as x_i according to Eq. 2.10. Put them together as one term provides a formula to adjust weights on hidden layer as,

$$\Delta w_{ji}(p) = -\eta\delta_j x_i = -\eta\left[\sum_{k=1}^{M}\delta_k w_{kj}\right] f'(a_j)x_i \tag{2.26}$$

By resolving the value of Δw on both hidden layer and output layer into current weights, the updating process will occur after iteration p by

$$\boldsymbol{w}(p+1) = \boldsymbol{w}(p) + \Delta w \tag{2.27}$$

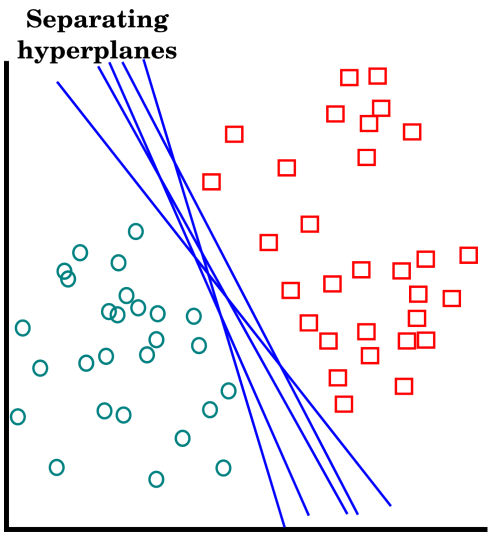
Note that the updating will in general be proportional to three factors which are the match between the target value (t_n) and the output value from the network, the differentiable function of the net activation, and input value in unit layer. If output value has been frequently matched to the target, there are no changes on the weight which is a sign that the iteration may be able to be stopped due to achieving an optimal result.

2.5.2 *Support Vector Machine*

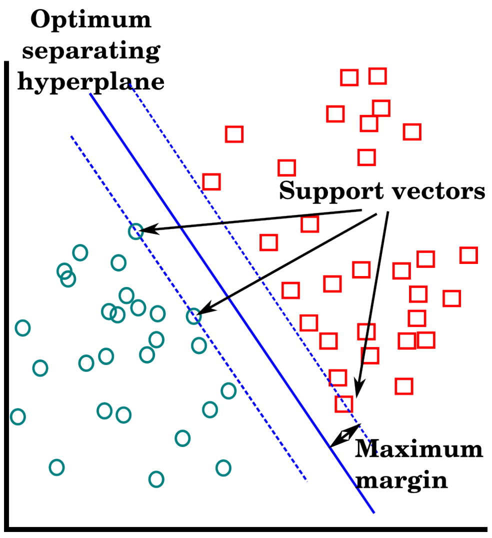
The foundation of SVM is firstly introduced in 1995 by Vapnik [59] as an application of statistical learning theory to solve classification problems. Several successful applications of SVM in classification, regression, and novelty detection [4] increased the popularity of SVM because it can effectively do these tasks and at the same time deliver a very promising accuracy. Regarding classification, its performance has been proved better in many cases compared to other classifiers.

2.5.2.1 *SVM Learning Algorithm*

Basically, the SVM was developed for the case of binary classification problems where the sample object is presumably linearly separable. The idea behind this classifier is to separate objects with a hyperplane that is constructed by maximizing the distance between a separator (separating hyperplane) and outermost boundaries of each class. To learn how SVM works, Fig. 30 provides a geometric picture to help understanding the approach behind the SVM in the case of a binary classification problem.



(a) Multiple separating hyperplanes.



(b) An optimum separating hyperplane.

Figure 11: SVM classifier for binary classification. Decision for a separating hyperplane is chosen such that the margin is maximum distance to the nearest data points.

Suppose all data points in Fig. 11 represent a problem that needs to be classified into two classes. Intuitively, the rough solution for this classification problem is to split the data points into two parts by composing a geometric line. This line in terms of the SVM is called a separating hyperplane. Note that in the context of the two dimensional domain, the separation can be handled by a line where the separator can be expressed as a line function $ax + by = c$. Whereas in a higher dimension, the separation will be handled by hyperplanes.

In general, there are a lot of separating hyperplanes as solution to this classification problem as sketched in Sub figure 11a. Then the question is how to decide for the best one among multiple existing lines. As explicitly described in the beginning of this sub section, SVM will probe the outermost data points around the separating hyperplane and compute their distances to the hyperplane. As there are many possible hyperplanes, this probing will apparently find many configurations of outermost data points. The separating hyperplane is selected in such a way so that the distance between outermost data points for each class respectively to this separating hyperplane is maximum. The scheme to solve this problem is called *Lagrange Optimization* [4], [8], [10]. After this separating hyperplane has been found, these outermost points are called ***support vectors*** while the distance from these support vectors perpendicular to separating hyperplane is called ***margin*** (see sub figure 11b for illustration).

The formation of the support vectors are used to define desired hyperplane. This specifies a decision boundary for the SVM. From this decision boundary, a decision function can be defined. It is characterized by two parameters, a weight vector $\boldsymbol{w}$ that is orthogonal against separating hyperplane and a constant b that regulates a bias or threshold. This decision function is represented in a pair $(\boldsymbol{w}, b)$ that is formulated as the following,

$$f(\mathbf{x}) = \boldsymbol{w}^T \cdot \phi(\mathbf{x}) + b \tag{2.28}$$

Where:
$\boldsymbol{w}$ is a weight vector
b is a bias or threshold parameter
$\phi(x)$ is a fixed feature-space transformation function

Note that each data point $\mathbf{x}_i, i = 1 \dots N$ in the sample has a corresponding target label $y_i \in \{-1, 1\}$. Given such a hyperplane $(\boldsymbol{w}, b)$, the classification process of new data points $\mathbf{x}$ are based on the sign of $f(x)$.

With the initial assumption that the classes are linearly separable, by definition there will be at least one pair $(\boldsymbol{w}, b)$ such that the equation 2.28 satisfies $f(x_i) > 0$ when $y_i = +1$ that indicates the points in the class $+1$ and $f(x_i) < 0$ when $y_i = -1$ that indicates the points in the class -1. In a compact form, for all training data points, both inequalities can be rewritten as,

$$y_i(\boldsymbol{w}^T \cdot \phi(\mathbf{x}_i) + b) > 0 \tag{2.29}$$

As the goal is to select the margin as wide as possible, the original problem is basically an optimization task by maximizing the distance of support vectors (the closest data points) to separating hyperplane. The geometric distance of the support vectors (data points) perpendicular to separating hyperplane can be computed by formula [4]:

$$\begin{aligned} d((\boldsymbol{w}, b), x_i) &= \frac{y_i(\boldsymbol{w}^T \cdot \phi(\mathbf{x}_i) + b)}{\|\boldsymbol{w}\|} \\ &= \frac{1}{\|\boldsymbol{w}\|} \left(y_i(\boldsymbol{w}^T \cdot \phi(\mathbf{x}_i) + b) \right) \end{aligned} \tag{2.30}$$

The margin, in this case represented by the distance d, will be maximum if the value of $\|\boldsymbol{w}\|$ is minimum. Unfortunately, a direct solution of this problem is not straight forward. The original problem should be transformed into another form that expresses the same problem with a much easier solution. This can be done by rescaling $\boldsymbol{w} \rightarrow \lambda \boldsymbol{w}$ and $b \rightarrow \lambda b$ which in principle will not change the distance d in formula 2.30. This rescaling form is called a canonical representation of the decision hyperplane. As this rescaling maintains a flexible way to assign value of the decision function, the closest data point can be set to a certain value to define the margin. Based on this freedom, the following equation

$$y_i(\boldsymbol{w}^T \cdot \phi(\boldsymbol{x_i}) + b) = 1 \tag{2.31}$$

is arranged for the points closest to the separating hyperplane. This explicitly selects the closest data points to be support vectors from any data points as long as the margin is equal to 1. Consequently, all data points $\boldsymbol{x_i}$ will satisfy the constraints,

$$y_i(\boldsymbol{w}^T \cdot \phi(\boldsymbol{x_i}) + b) \geqslant 1 \tag{2.32}$$

From equation 2.30, it can be noted that the problem need to maximizing the distance $\frac{1}{\|\boldsymbol{w}\|}$. This maximization is equivalent although not in general to minimizing $\|\boldsymbol{w}\|^2$. In fact the problem can be defined as,

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{subject to} \quad & y_i(\boldsymbol{w}^T \cdot \phi(\boldsymbol{x_i}) + b) \geqslant 1, \quad i = 1 \ldots N \end{aligned} \tag{2.33}$$

Note that the constant $\frac{1}{2}$ is added for normalization. This minimization problem is an example of ***quadratic programming***. To solve this problem, some techniques can be applied. The lagrange multiplier is an appropriate method to generate the solution of quadratic programming. The reader can refer to [4], [8], [10] for the detail of how this approach is used for solving the problem.

During construction of the solution, Lagrange multiplier introduces a constant α and modifies the decision function to be:

$$f(x_i) = \sum_{i}^{N} \alpha_i y_i k(\boldsymbol{x}, \boldsymbol{x_i}) + b \tag{2.34}$$

Here $k(x, x')$ that is called a kernel function, is introduced by the Lagrange multiplier during the construction of the quadratic programming solution. The term $k(x, x')$ is defined as a dot product $\phi(x)^T \phi(x')$. The classification of new data points is based on the model can be examined according to the sign of the decision function as formulated in Eq. 2.34.

As the quadratic problem has been solved by obtaining α, the parameter b is computed by considering all support vectors x_n that satisfy the constraints $y_i f(x_i) = 1$ (constraint in Eq. 2.31). By substitution $f(x_i)$ from Eq. 2.34, the constraint turns to be:

$$y(x_i)\left(\sum_{j \in \mathcal{S}} \alpha_i y_i k(\boldsymbol{x}, \boldsymbol{x_i}) + b\right) = 1 \tag{2.35}$$

Where:
$\mathcal{S}$ indicates the set of indices of the support vectors

To obtain parameter b, this equation can be solved by using an arbitrary support vector x_i. However, numerical computation by considering all support vectors and then averaging them is much more stable [4] than a solution by only considering a single support vector. With this rationale, the formulation of b can be resolved by

$$b = \frac{1}{N_{\mathcal{S}}} \sum_{i \in \mathcal{S}} \left(y(x_i) - \sum_{j \in \mathcal{S}} \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) \right) \tag{2.36}$$

Where:
$N_{\mathcal{S}}$ is the total number of support vectors

Note that the optimality of SVM is influenced by points close to separating hyperplane. These points, as aforementioned explanation, called support vectors. This is a great strategy to solve the problem without involving the massive state space searching. The solution is mainly derived from the contribution of those support vectors only so that it can save processing time. This is one prominence of SVM among other classifiers.

2.5.2.2 *Non-Linear Data SVM*

As mentioned previously, the basic assumption of applying an SVM for classification is that the classes are linearly separable. Whereas in practice, most of the data samples are far from linear and this can overturn the concept of SVM. In this circumstance, the kernel trick has been introduced to deal with non-linear classification.

The indigenous principle of linearly separable in SVM is retained but the original data points are transformed in such a way that the separation can be achieved linearly in a new space. This can be done by exchanging the kernel function from a linear function into a non-linear function. The most common kernel functions for overcoming non-linearity data sample are:

- Sigmoid function

  $$k(\mathbf{x}, \mathbf{x}_i) = \tanh\left(C \cdot (\mathbf{x} \cdot \mathbf{x}_i) + \theta\right)$$

  Note that the SVM approach with a Sigmoid kernel function is essentially similar to a NN with a Sigmoid function as the activation function.

- Polynomial function

  $$k(\mathbf{x}, \mathbf{x}_i) = (C \cdot (\mathbf{x} \cdot \mathbf{x}_i) + 1)^p$$

- Radial Basis Function (RBF)

  $$k(\mathbf{x}, \mathbf{x}_i) = \exp\left(- \frac{\|(\mathbf{x} - \mathbf{x}_i)\|^2}{2\sigma^2} \right)$$

Each function rather works differently and may be problem dependent. A prudent selection based on a few insights of samples prior to classification will help choosing an appropriate kernel function.

Finally, some key points can be identified as characteristic of the SVM. One of them is that the complexity of the SVM approach is defined by the number of the support vectors rather than the dimensionality of the feature space. More dimensions imply a higher dimension hyperplane and thus higher complexity due to the need of more support vectors.

Another prominent bottom line of the SVM is because the solution of the classification problem is reduced to design of a hyperplane. This will swap the model of the objective function into a convex problems so that the solution can be generated in a direct manner [4].

The obstacle of non-linearity is also anticipated moderately. By exchanging the kernel function from linear to non-linear, this issue has been handled wisely without sacrificing too much in the accuracy of the classifier.

The SVM is basically considered for a binary classification. While many problems are also non-linear. To cope with multi class classification problems, it can be extended by regarding the problem as multiple binary classification problem. This multiple binary classification problem can be solved by a particular technique such as one-versus-all or one-versus-one. More detail about these techniques, readers can refer to [4] or [8].

2.5.3 *Gaussian Mixture Model*

In the Gaussian Mixture Model (GMM) classification approach [18, p. 188–190], a density $P(x|\lambda_j)$ is estimated for each class. The classification is done by selecting the class with the highest score,

$$j = \arg\max_j \left(P(x|\lambda_j)\right) \tag{2.37}$$

Where:
x: the feature vector
λ_j: class j

The GMM [4, p. 430–439] is a density model using a weighted sum of Gaussians. The classification uses such a model for each of the N classes. The probability of the D-dimensional feature vector x given the class λ_j is defined as the mixture density by,

$$P(x|\lambda_j) = \sum_{i=1}^{M} w_{i,j}\mathcal{N}(x|\mu_{i,j}, \Sigma_{i,j}) \tag{2.38}$$

Where:
M: is a number of components
$w_{i,j}$: the weight of component i
$\mathcal{N}$: the Gaussian normal distribution
$\mu_{i,j}$: the mean of the component i
$\Sigma_{i,j}$: the covariance of the component i.

Since the components are normally distributed, parameters of each component are characterized by the means μ and covariances Σ as indicated in Eq. 2.38. The dimension of the mean $\mu_{i,j}$ is Dx1 and the dimension of the covariance $\Sigma_{i,j}$ is DxD. Note that the weight of $w_{i,j}$ satisfies the constraint $\sum_{i=1}^{M} w_{i,j} = 1$ and $0 \leqslant w_{i,j} \leqslant 1$ for each class index j.

A classification based on GMM is a classification by modeling the classification problem using GMM approach. During training phase, the component parameters of GMM are estimated from train dataset. The estimation of the parameters of a mixture can be handled by various techniques. The Expectation Maximization (EM)-algorithm [9] is a well established approach for this estimation. This algorithm is an iterative method for calculating maximum likelihood distribution parameters so that the best match parameters can be obtained.

2.5.4 *Multistage Classification*

In ideal situation, the task of classification can be finished at once which means that objects can be recognized at the end of the classification step. However, some cases of classifications do not directly classify complete characters in one step but only subset of character instead. Then, a further step is needed to refine each subset. By performing the latter step, the overall classification task can be completed. The scheme which consists of a classification in the beginning followed by further classifications is categorized as a multistage classification scheme.

The multistage classification scheme is mostly applied to target objects with the high complexity formation. Such objects need a great requirement for both computation and storage if a single stage classification is performed. By splitting object targets into some classification subsets, the whole classification can be broken into several consecutive classifications. As the classification scope in each subset becomes less complex, the cost of computation and storage would consequently be reduced.

The multistage classification has been applied in classification of several scripts in the field of Document Analysis and Recognition (DAR). Some of them are to recognize the Roman script, Chinese, and Marathi script. The texts of these scripts have complex structures in various different level. The last two scripts apparently consist of many combination texts in very complicated structure.

The usage of multistage classification in Marathi script is intended to recognize compound characters of handwritten Marathi [50]. The complexity of generating compound characters become higher due to the combination of consonants and vowels or consonants and consonants forming a new symbol. Based on the report in [50], the compound character recognition can be improved by applying multistage classification.

In the work of handwriting Chinese character recognition [60], the multistage classification consists of three stages. The whole classes of Chinese handwriting are divided into a set of subsets which are called groups. The first step is to search the most representative prototype to globally initialized the desired groups. The second step is performing optimization of the groups centroid. Finally, the fine classifiers are trained by using local features after all groups have been decided. The performance of this approach is claimed better in term of the recognition accuracy and the time of processing.

Another work of multistage is also applied to the recognition of the Roman characters. The idea in the work of multistage classification in [17] is to split the overall classification into several tasks with the goal to reduce the complexity. The task of classification is broken into three smaller tasks. The role of the first task is to classify the instance into upper and lower case. The second task is then to classify instances from the first task into 15 cluster of characters. Each cluster in the second task is designed to group Roman characters that are similar in shape as a strategy for simplifying the complexity of the classification process. Then, the final task is to classify the instance from second task into the complete Roman characters. This means that the final classification refines 15 character classes into 52 character classes.

3 PROPERTIES OF LAMPUNG SCRIPT

It is not surprising that most of the scripts in the region of Southeast Asia like Javanese, Balinese, Thai, Lao, and Burmese are descended of the same ancestor script in India [14]. This is also true for *Lampung script*, the script used by indigenous of Lampung in Lampung province, Indonesia. This script is originally derived from the cluster of *Brahmic script*, an ancient script from South India.

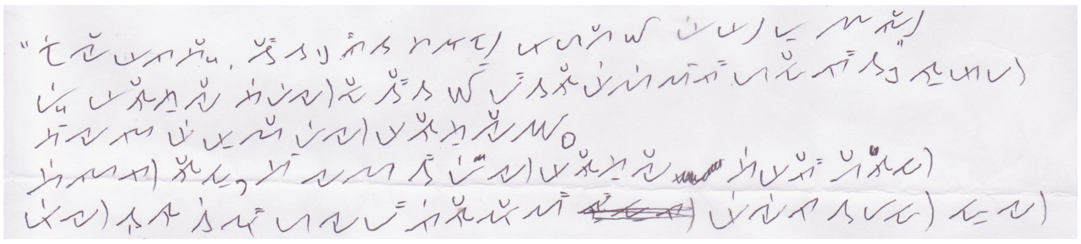
It is believed that Lampung script had been used by native tribes in Lampung provincial area since a long time ago. This is reflected by some ancient manuscripts collected by individual, local museum, and also international museums. A few of them are in Museum of Ruwa Jurai in Bandar Lampung, Indonesia, the National Library in Jakarta, Indonesia, University of Leiden in Netherlands, the School of Oriental and African Studies in London, United Kingdom, and the National Library of India [47]. However, there is no certain information that the script ancestor, Brahmic script, started reaching Lampung region, how it spread through out the Lampung area, by whom it was delivered, and how the evolution occurred from its ancestor. The only notable historical information about the script is its origin, as stated in the first paragraph.

The Brahmic script family falls into *the abugida* [14] writing system class. Lampung script is consequently categorized as the abugida class as well. In this type, each character of the script indicates a particular syllable constructed by the consonant-vowel composition. The vowel is inherently associated to the consonant unless it is overridden by a sound modifier.

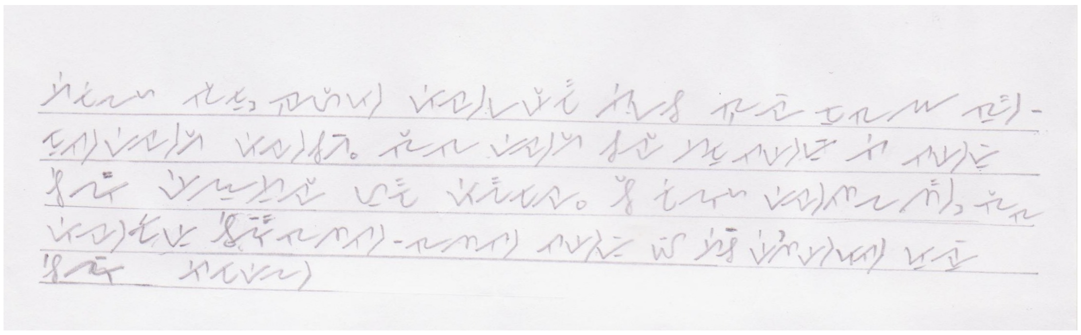
Lampung script is a non-cursive script which is written from left to right. The characters suppose to be distinguished each others in both printed and handwritten texts. Unlike the Roman script which can be written either in cursive or non-cursive style, it is impossible to join two adjacent Lampung characters because the combination will exchange characters to be a non-character symbol. Hence, Lampung script is permanently a non-cursive script without a possibility to be written in a cursive style.

3.1 SCRIPT UTILIZATION

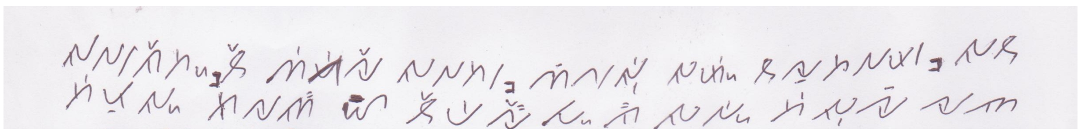
Lampung script is not a complicated and difficult script to be learned and used. The local inhabitants in Lampung can easily write the script to produce some texts. This easiness still does not encourage inhabitants to frequently use Lampung script because Roman script is too dominant in their writings. As it is simply understandable and applicable for writing, it can be dedicated to compose texts not only in Lampungnes but also Bahasa Indonesia, the Indonesian language. The sample of texts in Bahasa Indonesia written by using Lampung script can be seen in Fig. 12. In this research, all handwritten texts are fully in Bahasa Indonesia and written by using Lampung script.



(a) with folded artifact



(b) with guiding line



(c) with skewed line

Figure 12: Sample of the texts in Bahasa Indonesia transcribed using Lampung script. The texts consist of the basic characters and particular marks around this character that so-called diacritics.

At the recent time, the usage of Lampung script in writing nearly vanishes in society. Each documented manuscript was found containing the Roman script instead of Lampung script. The reason is neither because of Lampung inhabitants is illiterate of the script nor ignorant to the script but rather inhabitants use a formal script –Roman script– in their writing to communicate to other inhabitants. This is the way of Lampungnes to respect other ethnic groups that live together in Lampung. As information, Lampung provincial area had become one specific-purpose territory of a local migration (transmigration) especially from Java and Bali island since the Netherlands colonization until the Soeharto regime.

Looking at the current situation, the utilization of Lampung script is not really significant and the script has to be protected from extinction. The threat is becoming bigger and the script will probably not survive in the future if the users decrease. The low frequent usage of Lampung script eventually tends to decrease the spirit of preserving the script.

This fact alarms the local authorities –Lampung provincial government– to concern about the script preservation. Although the Lampung provincial government does not have outstanding program to revive the script, the government initiated a smart endeavor to cope with that problem. The Lampung script learning was integrated as a course for local curriculum of the elementary school and junior high school in Lampung. With this effort, the script will regularly be learned by students and it will get much attention by young people.

3.2 CHARACTERS

Although Lampung script descended of the Brahmic script family, all characters of Lampung script are not as complex as their origin. The characters are much more simpler in shape than its genuine characters. The recent script is the result of the evolution of the original raw script during a long time period of its engagement in the society. The list of characters in Lampung script is completely presented in Fig. 13.

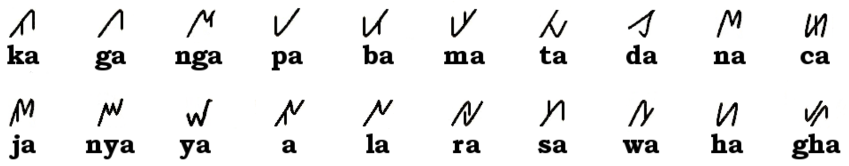


Figure 13: Lampung script consist of 20 basic characters. The character name is taken from the syllabic pronunciation of the character itself.

Lampung script only comprises 20 basic characters. Each of them corresponds to a consonant-vowel syllable, except the character *a* (𑴀) that purely represents a single vowel. The major shape of all characters is the curvature. More precisely, each character contains at least one cavity which can face up and/or down. These cavities are not symmetrical so as the main orientation of characters seems not upright. Yet every single character tends to have a backbone across the bottom left to upper right side.

As clarified toward the abugida class in the beginning of this chapter, the basic character transcribes a consonant with an inherent vowel and it eventually generates a syllable with a specific pronunciation. In this context, all basic characters excluding the single character *a*, are pronounced as the respected consonant with an inherent vowel "*a*". In addition, the character pronunciation also serves as the name of the character (see Fig. 13 for the detail).

Note that number of characters in Lampung script is less than characters in Roman script. Thus Lampung script does not encompass all characters of the Roman script. Some characters have never existed in Lampung script such as f, q, v, x and z because those characters are not recognized in the writing of Lampungnes. If the texts contain one of those characters, it can be replaced by a character that resembles to it. For example, character f and v can be substituted by the consonant p from the character *pa*, the character q can be substituted by the consonant k from the character *ka*, and the character z can be substituted by the consonant j from the character *ja*. The character x never exists both in either Lampungnes or Bahasa Indonesia. In case the character is needed in the text, its role can be played by a combination of the sound k and s from character *ka* and *sa*, of course after their vowel are muted.

For a frequent use of nasal voice, Lampung script provides two characters. The voice of nasal palatal is represented by the character *nga* and nasal velar is represented by the character *nya*.

3.3 DIACRITICS

As each basic character of Lampung script always transcribes a consonant with an inherent vowel *a*, the syllable pronunciation of it will always end with the vowel *a*, likewise as the pronunciation of the vowel *a* in word "but". Beside the vowel *a*, other ending vowels frequently appear in the text during the writing. To set other vowels, the Lampung writing system employs diacritics along the basic characters. The presence of diacritics is essentially needed to override the inherent vowel of the basic characters into another vowel. Thus diacritics play an important role as a vowel sound modifier of the syllable formed by the basic character.

The appearance of diacritics along the character scatter in various positions, close to the basic character. In the sample of the Lampung texts provided in Fig. 12, diacritics can be found nearby the basic character on the top, the bottom, or the right position solely. However, in some parts of the document , two or three diacritics may simultaneously emerge in one character in a certain combination of positions. This is an allowed operation in the Lampung writing system during the composition of texts to form a certain syllable pronunciation. Having diacritics surrounding the basic character will not affect the shape of the character itself since diacritics are located narrow but not attached to the basic character. The addition of diacritics will only control the modification of the vowel.

The overall diacritics in the Lampung writing system consist of seven shapes regardless of their position. Each of them is geometrically unique so they are clearly distinguishable from each other. These unique shape of diacritics can be seen in Fig. 14.

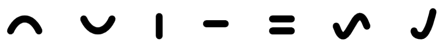
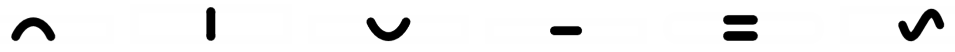


Figure 14: All unique diacritics of the Lampung writing system.

Moreover, the specific functionality of diacritics for overriding the vowel of the basic character can be explored further according to their place around the basic character. Among of seven diacritics, some of diacritics can appear only on one side or a few of them can appear in two sides or three sides. If the diacritic is grouped by considering these three positions, diacritics enlarge to be twelve diacritics due to some shapes may appear on two or three different sides. The detail on which they are distributed on each position is explained in the following.

3.3.1 *Top diacritics*

The majority of those unique diacritics as viewed in Fig. 14 are positioned on the top of the character. In total six diacritics among them can occupy this position. Each of them is named by a particular term followed by an explicit vowel they generate. All these diacritics along with their names are depicted in Fig. 15.



(a) ulan é (b) bicek e (c) ulan i (d) tekelubang ang (e) datas an (f) rejenjung ar

Figure 15: The set of diacritics that can be placed on the top of the character.

The first two diacritics, "ulan é" (⌒) and "bicek e" (|) can override the inherent vowel of a basic character into the vowel *e*. Therefore, the differences of both can not be inspected directly on the text but they can be detected in pronunciation of the vowel *e*. For "ulan é", the pronunciation is like the vowel *e* in word *dosen* (English: lecturer) while for "bicek e", the pronunciation is like the vowel *e* in *sekarang* (English: now).

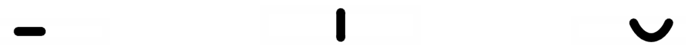
The diacritic "ulan i" (⌣), as indicated at the name, can exchange the inherent vowel of a basic character into the vowel *i*.

The last three diacritics in this category have a little specialty in their function. They do override the inherent vowel of the basic character and at the same time add an expansion at the end of the vowel by using a consonant or a nasal so as producing a particular string. Those strings are *ang* for diacritic "tekelubang ang" (‒), *an* for diacritic "datas an" (=), and *ar* for diacritic "rejenjung ar" (∿). All those strings frequently occur in the text of Lampungnes or Bahasa Indonesia.

Note that a special attention should be made about the top position. It can hold a pair diacritics at once from the set of diacritics in Fig. 15 depending on the syllable to be created. A few examples of this composition can be observed in Subsection 3.4.

3.3.2 *Bottom diacritic*

Diacritics which can be put beneath the character consist of three out of the seven unique shapes. During the syllable construction, the occurrence of the diacritics in pair at once on the bottom of the character is also possible. A proper illustration can be observed in the Section 3.4.



(a) bitan u (b) bitan o (c) tekelungau au

Figure 16: The set of diacritics that can be placed on the bottom of the character.

The diacritic "bitan u" (‒) changes the inherent vowel of a character into the vowel *u*. Meanwhile the diacritic "bitan o" (|) transforms the inherent vowel of a character into the vowel *o*.

The role of the third diacritic "tekelungau au" (⌣) is different compared to the other two diacritics in this category. Instead of one vowel replacement in a syllable, the diacritic causes one vowel of the character to be converted into two vowels a so-called diphthong. As indicated by its name, this diacritic change the vowel into the diphthong *au*. This diphthong is often found within a syllable of Lampungnes or Bahasa Indonesia respectively. To discover some real examples in texts, the reader can refer to Section 3.4.

3.3.3 *Right diacritic*

The member of diacritics which can be situated at the right side of the character comprise of three unique shapes. The right position of the basic character can only be occupied by one right diacritic at once. The list of all these diacritics are pointed out in Fig. 17.

The diacritic "tekelingai ai" ( ) is also a diacritic for generating a diphthong. It will substitute the inherent vowel of the basic character into the diphthong *ai*.

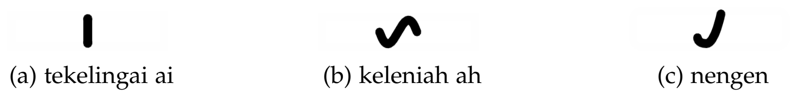

(a) tekelingai ai (b) keleniah ah (c) nengen

Figure 17: The set of diacritics that can be placed on the right of the character.

The diacritic "keleniah ah" ( ) has the same role as the two mentioned diacritics on the top, "datas an" and "rejenjung ar". It can exchange the vowel of the basic character into the vowel *a* and at the same time add the consonant *h* at the end of the vowel so that the whole composition turns to be string *ah*. This arrangement is used to handle the tail part of the syllable in form of string *ah* which frequently occurs in Lampungnes or Bahasa Indonesia.

Regarding the last mark, "nengen", there are two different point of views discussing about what kind of mark this symbol belong to. In [47], the author specifies the mark as a punctuation mark, while in [48] the author characterizes it as being a diacritic. Since its functionality is related to the alteration of the vowel of the basic character, the role of this diacritic is closer to a diacritic rather than a punctuation mark. Hence in this study, the mark is consequently considered as a diacritic.

The diacritic "nengen" ( ) is categorized as one special diacritic in Lampung writing system. This diacritic is used to tackle the inherent vowel of a basic character as well but in a different way as other diacritics are. It is used to mute the inherent vowel of the basic character so that the remaining part is only the consonant of the basic character. Since its function is omitting the vowel, the diacritic "nengen" is never used with other diacritics simultaneously.

The consonant as a result of this operation is not an independent syllable anymore. According to the rule in Lampung or Bahasa Indonesia writing system, it must be incorporated to the predecessor syllable. For example, the word *bahasa* (English: language) consists of three syllables, *ba*, *ha*, and *sa*. The transcription of the word in Lampung script is . With a diacritic "nengen" on the right end of the character *sa* in this word, the transcription becomes which forms the word *bahas* (English: discuss). If this word is separated based on its syllable, it constitutes of two syllables, *ba* and *has*, where the consonant formed by adding the diacritic "nengen" is merged to predecessor syllable.

3.4 COMPOUND CHARACTER

Before discussing the text constructed by using Lampung script, it is important to give a succinct explanation regarding the syllable construction in Bahasa Indonesia by using the Roman script. This is essentially needed because all Lampung text materials in this study are transcribed from documents in Bahasa by using the Roman script.

Suppose that a vowel is shortly represented by V and a consonant by C. In general, there are three most common syllable patterns in Bahasa Indonesia. Those are in form of V, or CV, or CVC which dominate the used syllable in Bahasa Indonesia. Note that the last pattern –CVC– can be extended into pattern like CCVC, CVCC, and CCVCC. In some rare cases, the number of consonants can be three letters in the left side. The words in Bahasa Indonesia can then be built by a combination some of these syllables. The following are a few example of words in Bahasa Indonesia that are separated into appropriate syllables according to those patterns. The syllable in the words *si-a-pa*, *i-ta-li-a*, *men-da-hu-lui*, *u-lang*, *e-mas*, and *ra-di-o* complies to the rule of character-separating in the writing system of Bahasa Indonesia. Nevertheless, the character-separating manner will be different if the text will be transliterated by using Lampung script. One general guidance is to split the text in Roman characters in Bahasa Indonesia into sequences of consonants followed respectively by a vowel if the vowel exists.

Likewise to Roman script, Lampung script is also able to transcribe the text in Bahasa Indonesia. In a similar way as the Roman script composing the text by using Roman characters, the Lampung writing system also employs Lampung characters as a basis to generate texts. However, as noted before, texts written in Lampung script will also be stipulated by diacritics around the basic character. In fact, the most relevant thing to be investigated is how the combination of the character and diacritics of Lampung script supports the transcription of the text in Bahasa Indonesia.

The text written in Lampung script can be established by a sequence of basic characters with or without the presence of diacritics nearby. The sequence of one word may contain various combinations of the character and/or character-diacritics. In addition, multiple diacritics in various positions lying around the character may also occur during the production of the text. The composition of a character with or without diacritics is called compound character. A compound character defines a particular syllable required to build the text. Therefore, the syllable will always depend on the character and certain diacritics with their positions. A different composition will accordingly deliver a different syllable. The following illustration provides some configurations of the compound character that can be interpreted as a new string for replacement of the inherent vowel of the basic character.

- A Basic Character
  The most simple form of the unit text in Lampung script is composed by only a character without any diacritics. This unit text can be transcribed into Roman script in form of string CV where the V is always interpreted by the vowel *a*. Hence the sequence of Lampung characters will construct the text in sequence

of the string CVCV...CV. For example the text in Table 1 no. 1 consists of three basic characters composing the word *ca-ha-ya*.

Among Lampung script characters, there are three characters that have the pattern CCV while one character represents a pure vowel. Those three characters are *nga* ( ), *nya* ( ), and *gha* ( ). One example of the word constructed by this character is showed in Table 1 no. 2. The word consist of two character forming the word *nya-ta*.

The only character representing a vowel in Lampung script is character *a* ( ). The existence of this vowel is to afford the need of the vowel syllable in Lampungnes which frequently occurred in texts. For example, the usage of character *a* in the word *a-sa* can be transcribed as the text indicated in Table 1 no. 3. Note that as the single independent character symbol, the character *a* has never been used as the inherent vowel part of other characters or other strings

- A Basic Character and A Single Diacritic
  Each diacritic on each position can be used as a single diacritic at the position it supposes to be. Therefore, in general there are three group configurations according to their basic positions and the overall configuration consist of twelve particular forms distributed over individual positions. All samples of the text with one surrounded diacritic are supplied by Table 1 no. 4-15.

  The top group consist of six diacritics. The particular string forms produced by this group are *é*, *e*, *i*, *ang*, *an*, and *ar*. The usage of this diacritic around the basic character will override the vowel *a* of basic character into one of those strings. For example in Table 1 no. 4, the basic character *ma* ( ) changed to be string *me* whenever the diacritic "ulan é" ( ) appears on the top of the character. Another example is the syllable *kar* in no. 9 of Table 1 which can be formed by placing the diacritic "rejenjung ar" ( ) on the top of character *ka* ( ). Other examples with the rest of diacritics can be observed in Table 1 no. 5-8.

  With the bottom diacritic, a vowel *a* of the basic character can be changed into vowel *u*, vowel *o*, and diphtong *au*. Table 1 item no. 10-12 provide examples of texts that use bottom diacritics. The character *ca* ( ) can be converted to be *cu* by adding the diacritic "bitan u" ( ) on the bottom of the character. The same way also holds to generate the vowel *o* of the basic character. This is done by putting the diacritic "bitan o" ( ) on the bottom of the basic character as seen in the word *kado* in no. 11. The remaining examples indicate the usage of the diacritic "tekelungau au" ( ) to compose a special form consists of vowel *a* and *u* as one element. In no. 12, the character *nga* is switched to be *ngau* after the addition of the mark on the bottom of the character *nga* ( ).

  Two out of three right diacritics can establish the string *ai* and *ah* as a part of the whole syllable. The basic character *la* ( ) with the diacritics "tekelingai ai" ( ) in the right position as presented in the Table 1 no. 13 is switched to be the string *lai*. The second example in no. 14, the formation of string *rah* is composed from character *ra* ( ) after positioning the diacritic "keleniah ah" ( ) on the right side. The last diacritic on the right side, the diacritic "nengen"

(✓), as mentioned before, is functioned to eliminate the inherent vowel of the basic character. The usage of diacritic "nengen" is exemplified in Table 1 no. 15. It eliminates the vowel *a* of the character *la* (ʌ/) to be a fully consonant *l* in the word of *halal*.

- A Basic Character with Two Diacritics on The Top
  In the Lampung writing system, there exists a consensus that the vowel *a* in each possible string is always replaceable by another vowel represented by a particular vowel diacritic not containing the vowel *a*. In fact, the typical composition of two diacritics on the top of the character is to make the portion of the string by overriding the vowel *a* in the string *ang*, *an*, and *ar* by the vowel *é*, *e*, and *i*. The new created strings can be *eng*, *ing*, *en*, *in*, *er*, and *ir* where one slot of the two places on the top of the basic character may be filled by one of the first three of the top diacritics (⌃ ı ◡) and another slot by one of the rest (_ = ∿).

  For example, the diacritic "datas an" (=) generated the string *an* if it is solely used. However, if another vowel diacritic on the top accompanies it, the vowel *a* in string *an* may change to the vowel that represented by this new diacritic. One example is presented in Table 1 no. 16 which uses a combination of diacritics "ulan i" and "datas an" (◡ =) on the top of a character to construct the string *in*. Whenever both diacritics are placed on the top of the character *pa* (ʌ/), they form a string *pin*. Another example presents the string *ner* as a combination of the character *na* (ʌʌ), the diacritics "ulan é", and "rejenjung ar" (⌃ ∿). This can be seen in the Table 1 no. 17.

- A Basic Character with Two Diacritics on The Bottom
  There is only one possible combination for this composition by assigning the diacritic "bitan o" (ı) and "tekelungau au" (◡) side by side on the bottom of the character. This configuration will establish the string *ou* as a resultant of the string *au* coming from the diacritic "tekelungau au" and the vowel *o* coming from the diacritic "bitan o". The example on no. 18 of the Table 1 indicates the character *ca* (ʋʌ) with both diacritics generating the string *cou*.

- A Basic Character with Diacritics on The Top and The Bottom
  The new string formed by these two diacritics is a new string as a combination of the string from the top diacritic and string from the bottom diacritic. The way of these diacritics combined is, as explained aforementioned, following the consensus that the vowel *a* can always be overridden by other diacritics containing the vowel except *a*. However, a combination can not be imposed if both sides contain a diacritic represented a single vowel diacritic. Therefore diacritic on the top representing the vowel *é*, *e*, and *i* can not be paired to a single vowel diacritic on the bottom representing the vowel *u* and *o*.

  With all possible combination diacritics, the new composed strings can be *eu*, *iu*, *ung*, *ong*, *aung*, *un*, *on*, *aun*, *ur*, *or*, and *aur*. Two examples are performed by Table 1 no. 19 and 20. The first one represents the string *kor* as the combination of the character *ka* (ʌ) and the string *or*. This rear string can be created by joining the diacritic "rejenjung ar" (∿), and diacritic "bitan o" (ı). While the second example indicates the use of diacritic "tekelubang ang" ( ) and diacritic "bitan u" ( ) which generates string *ung* over the character *ra* ( ) such that the outcome string is *rung*.

- A Basic Character with Diacritics on The Top and The Right
  Although there are six diacritics on the top, only a part of them can be applied together with the right diacritics around the character as a pair. The combinations produce new strings containing *ei*, *éi*, *eh*, *éh*, and *ih*. For example in the word *arbei* in Table 1 no. 21, there is a string *ei* which can be created by combining two strings *ai* + *e* from the diacritic "tekelingai ai" ( ) on the right and "bicek e" ( ) on the top of the character ba ( ) such that it yields the string *bei*. In the second example in Table 1 no. 22, the string *nih* is composition of the character na ( ) and the string *ih*. Diacritics configuration for this string are the diacritic "ulan i" ( ) positioned on the top, and the diacritic "keleniah ah" ( ) positioned on the right of the character.

  The set of diacritics that can not be joined is the string *ang*, *an*, *ar* versus *ah*, *ai*. The reason is that both sides contain consonants such that the combination is not feasible to be part of a syllable. Another impossible combination is the vowel *i* as the diacritic "ulan i" ( ) versus string *ai* as the diacritic "tekelingai ai" ( ). This is not possible to happen because the final result will have two vowel *i* which is not a valid syllable in either Lampungnes or Bahasa Indonesia.

- A Basic Character with Diacritics on The Bottom and The Right
  In this arrangement, a few restrictions must be noticed during the pairing of diacritics. The first one is that it is impossible to arrange two diphthong diacritics concurrently in a pair. In fact, the diacritic "tekelungau au" ( ) on the bottom and "tekelingai ai" ( ) on the right have never been joined together at once. Secondly, the diacritic "nengen" is employed to eliminate the inherent vowel of the character which exclusively returning a pure consonant. The role of this diacritic is likely the opposite task of the other diacritics which controlling the vowel of the basic character. Consequently, the diacritic "nengen" can never been paired to any other diacritics at the same time.

  The rest of the combination comprises of four forms i.e. the string *ui*, *oi*, *uh*, and *oh*. Two examples are given in Table 1 no. 23 and 24. The string *luh* is constructed by the character *la* ( ) and string *uh* which is constructed by the diacritic "bitan u" ( ) on bottom of the character, and the diacritic "keleniah ah" ( ) on the right of the character. Meanwhile, the second example shows the string *toh* that consists of the character *ta* ( ) and the rear string *oh*. The latter is developed by the combination *ah* + *o* which basically formed by the diacritic "bitan o" ( ) on the bottom and "keleniah ah" ( ) on the right.

3.5 PUNCTUATION MARKS

Beside characters and diacritics, the Lampung writing system also employs punctuation marks. Compared to the Roman-based writing system, the Lampung writing system only has a few marks [48]. The total number of the punctuation marks consist of five marks. The list of these punctuation marks can be seen in Fig. 18.

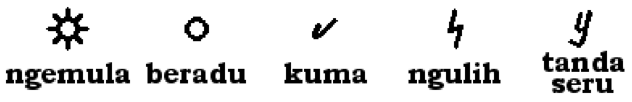


Figure 18: Punctuation marks in Lampung writing system. Ngemula is a mark to start a sentence. Beradu is equal to full stop. Kuma represents the comma. Ngulih is a question mark. And tanda seru is an exclamation mark.

Only the main punctuation marks, like a full stop, a comma, a question mark, an exclamation mark, and a unique mark for starting a sentence are available in the Lampung writing system. It does not recognize other marks like colon, semicolon, apostrophe, double apostrophe, slash, hyphen, and brackets.

1. Ngemula
   Ngemula is a special and unique mark in the Lampung writing system which most likely cannot be found in other writing systems. Its function is to commence a sentence. That is why the symbol to represent this mark like a shining sun because it reflects the philosophy of the sun starting the day by shining its light in the morning.

   The functionality of ngemula is well defined and understandable. Nevertheless, based on the observation in our original data collection (partially explained in section 6.1), nobody used this punctuation mark to start a sentence. This makes sense since the contributors have been custom with their daily writing system, the Roman-based writing system that does not have this kind of mark.

2. Beradu
   The function of the mark beradu is the opposite function of the mark ngemula. It is put at the end of a sentence to complete it. The symbol of this mark is a small circle with symmetrical shape in height and width. In practical, the size of this mark is around half the height of the basic character.

   It is unclear whether the mark beradu can also be used to mark abbreviation. Both literature sources in [47] and [48] do not summarize this issue because Lampungnes does not have particular abbreviation.

3. Kuma
   The mark kuma is equivalent to a comma. As the function of the punctuation mark comma, it is used to pause the sentence (somewhere in the middle) or to separate the elements in a series of three or more things in one sentence. In this context, the purpose of the mark kuma in a sentence by pausing or separating is to avoid confusion or emphasize some important things.

4. Ngulih
   The Lampung writing system also supports questions by supplying the mark ngulih as a question mark. In its role as a question mark, it can be put at

the end of a sentence as a mark in this sentence containing a question about something.

5. Tanda Seru
Tanda seru is a punctuation mark for expressing that a sentence contains an interjection, a command or an emphatic declaration. This is the same function as the exclamation mark in the Roman-based writing system. The mark is also placed at the end of the such sentences. Note that although the mark consists of two separated components, it is considered as one mark.

3.6 SPECIAL ATTRIBUTES OF LAMPUNG SCRIPT

An early observation of the Lampung handwritten character is worth to be addressed prior to the development of the handwritten character recognition system. This analysis can notice problems before the development phase. Those potential problems can be mapped onto appropriate solutions during the development process. The solutions will subsequently lead to a better design and the overall system can hopefully reduce the possibility of failure during the operation of the system.

The following analysis emphasizes some important facts from observations regarding the nature of Lampung script that can influence the design and development of the Lampung character recognition system. The particular handling may be prepared for such attributes in advance before the development of the system.

3.6.1 *Non-cursive*

As indicated in Fig. 12, all characters are separated from each other so that they have their own visible boundaries. In a closer look, two adjacent characters of the Lampung text are clearly unconnected. In the context of handwriting, this property is called a non-cursive script.

This property has a positive as well as a negative impact on the design and development of the Lampung handwritten character recognition system. The positive impact is that the character segmentation will not be a big issue since the extraction of the connected components (CCs) in the text (see chapter ..) will handle the segmentation and result in entities which can be considered as characters to some extent. A further evaluation and correction needs to be taken toward these entities to acquire a final character segmentation output.

However, this property also introduces a drawback. The distance among two adjacent characters, even in handwritten text, is equally uniform in length. It is therefore unclear where the border of words is, which in fact complicates to separate the words.

3.6.2 *No Uppercase*

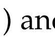
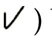
Lampung script comprises of only a single shape for each character. The script does not recognize the concept of the upper and lowercase characters like the most scripts in Asia. Thus all Lampung characters appear in the text with the same role. Probably the use of the punctuation mark ngemula as a sentence starter is to emphasize the mark of the first character in the sentence.

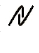
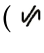
From the perspective of character recognition, this property is a benefit. First it only has to recognize one type of character, so that the design of the recognition system will be less complex than one that must recognize lower and uppercase character. Second, the fact that the number of character in Lampung script is only 20 characters and consists of one character type –no lower and uppercase character–, are also an advantage. The exploration time of the character domain will become lower compared to a recognizer of a script with both character case types.

3.6.3 *Character with Two Unconnected Components*

The basic characters in Lampung script generally consist of one component. However, two of the characters are respectively composed by components. Both components are separated from each other although they represent one character. Each component apparently comes from another basic character with a single component which are the characters *ga* ( ) and *pa* ( ). For the sake of simplification, both characters are respectively called the constructor character. Character with two components is formed by mutually interrelating those two constructor characters such that they are close each other.

The first character with two components is the character *ra* ( ). It is composed by putting the line tip of the right end of the character *ga* ( ) into the cavity of the character *pa* ( ) such that both constructor characters lay in parallel side by side without touching each other.

The second character is the character *gha* ( ) that is also constructed by the character *ga* () and the character *pa* () but in a different way of placement. Both constructor characters are formed by a line with two different stroke orientations forming the character cavity. One stroke is a short line and another stroke is longer. The longer stroke is skew with the slope orientation from the bottom-left to the top-right direction. To form the character *gha*, the longer stroke of each the character is put together such that the character *pa* ( ) is positioned on top of the character *ga* ( ).

Due to two component characters existing in Lampung script, a specific handling must be carried out prior to the character recognition phase. One is the detection of closeness of those two consecutive constructors. If their distance is within a certain threshold, then both can be considered as one character. Another treatment is the check of position of a constructor character relative to another constructor. The configuration of this position will determine which character is represented by both constructors, whether it is the character *ra* () or the character *gha* ().

3.6.4 *Diacritic with Two Unconnected Components*

The diacritic "datas an" (=) on the top position consists of two unconnected components. The single component of this diacritic is also a diacritic with the shape a horizontal line or a dash sign. The diacritic "datas an" can be formed by two copies of this horizontal line diacritic. One copy is arranged above of the other such that the shape of the diacritic "datas an" is similar to the symbol of equal sign in mathematics.

The potential ambiguity on the recognition of this diacritic is whether the both are together as one diacritic or respectively two separated diacritics. This will mainly occur whenever the diacritic "datas an" is located between two character baselines. To handle this double components diacritic, it first needs to be checked with specific distance threshold. Then if it is in the range, both components first need to be bound.

3.6.5 *Diacritic Resembles Character*

In the Lampung writing system, the characters of Lampung script are unique as well as the diacritics. However, the comparison among characters and diacritics signify some almost similar instances between the basic character shape and the diacritic shape. The following list denotes this resemblance:

1. The diacritic resembles to the character *ga* ( ).
2. The diacritic resembles to the character *pa* ( ).
3. The diacritic resembles to the character *ha* ( ).

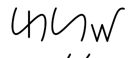
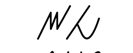
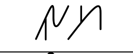
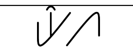
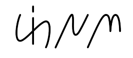
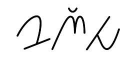
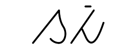
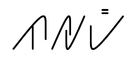
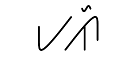
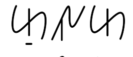
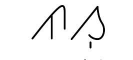
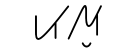
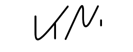
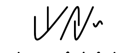
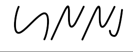
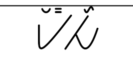
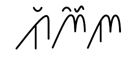
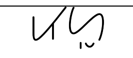
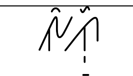
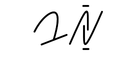
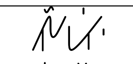
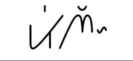
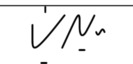
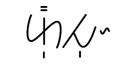
As explained in the beginning of this chapter, the size of a diacritic is smaller than the size of a character. Nonetheless, since the human handwriting is often fluctuating and it cannot be controlled, even by the writer of handwriting, there is always a likelihood that the a handwritten diacritic and character are nearly the same in size and shape. Since the detection of character candidates is run automatically, a big size diacritic in above list will be grouped as a character rather than a diacritic. This fact is indeed difficult to be avoided.

Another potential problem between a character and a diacritic occurs during the process of pairing both. The pairing of a character and a diacritic may lead to another character. The following configurations of a character and a diacritic indicate this possibility especially when the size of the diacritic nearly as big as the size of the character:

1. The character *pa* ( ) and the diacritic "ulan é" ( ) on the top can generate the character *ra* ( ).
2. The character *ga* ( ) and the diacritic "tekelungau au" ( ) on the bottom can also generate the character *ra* ( ).
3. The character *ga* ( ) and the diacritic "ulan i" ( ) on the top can generate the character *gha* ( ).

The recognition phase becomes more sensitive to errors due to all these problems. For the purpose of the design and development of the Lampung handwritten character recognition system, a particular concern on these problem solutions can help overcoming these problems.

Table 1: The usage of diacritics on the top, the bottom, the right, or combinations of them around the character. The table contains some examples of words in Bahasa Indonesia (except item no. 18 that is in Lampungnes) which are written in Lampung script.

No.	Diacritics	Position	String	Example	Transcription	English
1.	no diacritic	-	a		ca-ha-ya	light
2.	no diacritic	-	a		nya-ta	real or fact
3.	no diacritic	-	a		a-sa	hope
4.	⌒	top	é		*me*-ga	cloud
5.	ı	top	e		*ce*-la-na	pant
6.	⌣	top	i		wa-*ni*-ta	lady
7.	‒	top	ang		da-*tang*	come
8.	=	top	an		ka-ra-*pan*	bull race
9.	∿	top	ar		pa-*kar*	expert
10.	‒	bottom	u		*cu*-a-ca	weather
11.	ı	bottom	o		ka-*do*	gift or present
12.	⌣	bottom	au		ba-*ngau*	stork
13.	ı	right	ai		ba-*lai*	hall
14.	∿	right	ah		ma-*rah*	angry
15.	J	right	(muted)		ha-la*l*	halal
16.	⌣ =	top-top	in		*pin*-tar	clever or smart
17.	⌒ ∿	top-top	er		ki-*ner*-ja	performance
18.	ı ⌣	bottom-bottom	ou		ba-*cou*	read
19.	∿ ı	top-bottom	or		e-*kor*	tail
20.	‒ ‒	top-bottom	ung		wa-*rung*	stall
21.	ı ı	top-right	ei		ar-*bei*	strawberry
22.	⌣ ∿	top-right	ih		be-*nih*	seed
23.	‒ ∿	bottom-right	uh		pe-*luh*	sweat
24.	ı ∿	bottom-right	oh		con-*toh*	example

4 SURVEY OF RELATED WORKS

Recently, many approaches have been developed to solve many different tasks in the field of Document Analysis and Recognition (DAR). Some of those approaches are applicable for various scripts but others are only applied for specific scripts.

This chapter emphasizes several important approaches which are important for development of the Lampung handwritten character recognition framework. These approaches can be applied or modified as a preliminary foundation in the framework as they are compatible to the characteristic of Lampung script. Each of approach is concisely reviewed to introduce the basic idea of the methods along with the existing work for dealing with handwritten character input. The discussion comprises a topic about the feature vector, diacritics works, and the multistage classification of handwritten character inputs. These subjects are illustrated in the following sections.

4.1 WATER RESERVOIR FEATURE

As handwritten character recognition requires appropriate feature representation, various feature representations have been invented for dealing with recognition. However, some of those feature representations are meaningful for recognition of particular characters but some other are not. Therefore, feature extraction must be compromised to the nature of the character. The following subsections describes the Water Reservoir (WR) feature which is used in the first recognition of the Lampung handwritten character.

4.1.1 *Water Reservoir (WR) Principle*

Water Reservoir (WR) is not a pure terminology in DAR field but considered as a principle in the mechanical world. The idea behind the principle is that reservoirs are used to store water by pouring it into them.

The principle of a water reservoir can be adopted into DAR research particularly in the handwritten character recognition. With respect to this adaption, the research essentially uses of the main characteristic of the reservoir which is the bin of the reservoir itself. The bin is then translated as a cavity in the field of handwritten recognition research. Each cavity has some attributes like area size, center of gravity, the depth, and extension of them for example total number of reservoirs, type of the cavity, etc.

The strategy for applying this principle into handwritten recognition research is that the reservoir (also called **the cavity** as the same terminology and they are replaceable with each other) is filled by the water until fully loaded. Whenever it has been fully loaded, the volume capacity of the reservoir can be defined as the area size, the center of mass of the reservoir can be defined as the gravity center and the depth of the reservoir can be defined as the height. All these measurements can be exported as features needed for recognition scheme.

4.1.2 *Some Applications of WR principle*

The principle of the WR in the field of DAR was firstly introduced by Pal et al. in 2001 [41]. In that work, the WR-based feature was used for a segmentation task of the touching numerals. This approach is effectively applicable for segmentation due to the property of WR principle that is producing a large cavity whenever two numerals come into contact with. Therefore, the first step of segmentation task is the detection of large cavities as an indication of touching numerals. If it was found, then the next step is determining position of the cutting edge. After cutting, the segmentation process is completed.

Beside a large cavity, touching numerals will also have more reservoirs than isolated numerals. If the number of reservoir exceeds three, it can be concluded that the component is a touching numerals. Then the segmentation should be done over the component.

The WR approach is a convenient way to alert the touching numerals since it does not need the thinning and normalization phase prior to the segmentation. In their experiments, 94.35% of connected numerals are correctly segmented. The only drawback of this approach is miss-segmentation. It occured when the proposed method found a point break on the contour used as the boundary of the reservoir.

Since the usage of WR gives a good contribution in the field of DAR, the author emphasized the prominent of this WR-based approach in the field of DAR by publishing it into a journal in 2003 [42]. The authors encourage that the WR based concept will offer a potential benefit for pattern recognition community.

The application of WR-based approach was also applied for Bangla [40]. In this work, the WR-based approach was used for the segmentation as well. The task was appropriate since the Bangla handwritten texts particularly the words contain many touching characters. The connected part of the touching character is mostly occurred through the head-line hence two closest characters will generate a large bottom reservoir (reservoir with the open part face to down). In the first round, their work aimed to segment the line which did not use of WR-based approach. In the second round, the work was carried out to determine the isolated and touching characters. Finally, the last round was dedicated to split the touching characters using WR-based approach. Among of 1430 Bangla touching characters, 95.97% of them are correctly segmented. The rest are errors due to the touching characters have multi-touching points.

The different purpose of WR-based approach other than segmentation had been utilized in Malayalam handwritten Numeral [43]. In this work, WR-based features act as part of the features for recognition of the unconstrained Malayalam handwritten Numeral. Some WR-based characteristics like number of reservoirs, size and positions, water flow direction, ratio of the reservoir height to numeral height were chosen to be the features of the recognition scheme. However, the authors built a binary tree classifier to recognition the numerals which restrict their proposed approach to be character specific rather than more general characters.

The WR-based approach raise more and more attention in various fields in DAR. Its usage started to cover the problem in the postal automation [44]. In this work, the WR-based approach handled the pre-segmentation task of the touching digits in a postal document that contain multiple languages and multiple scripts. The idea of segmentation process remains the same as previous works, by getting benefit of the big cavity whenever the digits touched each other. In this way, the WR-based approach was applied for pre-segmentation into components regarded as the primitives of the candidate of the digits. The primitive components was merged into digit of possible pin-code (post code). To obtain the optimized segmentation, the Dynamic Programming was employed.

The applications of the WR-based approach keep moving forward into various purposes of the document processing. One of the notable application is focused on the orientation detection of the major Indian scripts [7]. The proposed scheme was executed for detection of the text line of 11 different scripts. Initially, the authors employed various features to detect orientation of the handwritten text including WR-based feature. Each of such a feature had been evaluated and tested. The conclusion indicates that the features generated from the WR concept can uniformly work out for any major Indian scripts.

The WR principle is potential to be applied in various fields of DAR. However, there is only a little works with respect to the application of this principle. Although not all fields can engaged this principle, the chance to be involved in the field of DAR is still opened.

4.2 DIACRITIC-BASED WORKS

In the world of writing, some scripts may have diacritics. These diacritics can be found in some script for example French, Greek, German, Czech, Hungarian, Spanish, Portuguese and Turkish from Europe, Arabic from the Middle East or Indic scripts like Vietnamese and Lampung from Asia. However the development of the handwritten recognition system concentrated more on the character rather than the diacritic. In our best knowledge, only a few works were dedicated for handling the diacritic.

4.2.1 *French*

A work on diacritic the French handwriting had been proposed in 2010 [55]. In general, the idea is to split the system into several HCR systems with smaller amount of the class member rather than only one system with the whole classes. By this manner, the complexity of such a system will be less than the one with all class members. Therefore, in this work the French handwritten characters were firstly processed into two groups, the non diacritic characters class and the characters with diacritic class. The further processing was done for the characters with diacritic. In this regard, those characters can be seen as a composition of two parts i.e. the character and the diacritic. Both were recognized separately in the beginning and at the end both would be checked whether the character part and the diacritic part could be constructed together or not. If it could, the composition character proposed as character with diacritic. Otherwise it would be recognized as the character without a diacritic.

4.2.2 *Vietnamese*

Vietnamese alphabet is basically compiled by the Latin alphabet with several additional small marks employed as diacritics. There are 9 diacritics in Vietnamese with two functionalities. One group comprises of four diacritics is used for producing an additional sound and another group consists of five diacritics is employed for controlling the tone of each word. The tone in Vietnamese like low, high, sharp, fall, or rise in tone is crucial to distinguish the meaning of the words.

The recognition of Vietnamese with their diacritics had been investigated for online handwritten character in 2008 [36]. The main work focused on the design of an input descriptor for Vietnamese recognition system. The descriptor was built based on the optimized cosine descriptor with a modification at the level of character strokes. Instead of using a vector with a small number of features, the proposed method regenerated the vector by re-sampling points over all strokes of a handwritten character and represented all of them in a single set of features. This input vector is then delivered to a recognition system that consists of three different layers. The first layer is designed for classifying of the main character. The second layer is for classifying the circumflex diacritics. The last layer is to identify the tonal diacritics.

4.2.3 *Arabic*

The most specific work on diacritics, dedicated in Arabic can be found in [33]. The work had shown a different perspective on handling of the document that consists both of the character and diacritics. The usage of the diacritic without involvement of the character was applied for identification of the writers. The features were solely extracted from diacritics by calculating the Linear Binary Pattern (LBP) histogram. The writer will be identified out of database whenever the distance between LBP histogram of the unknown writer and the known writer in the database is minimum. The proposed approach had been tested on the IFN/ENIT database [45] with performance rate 97.56% from total 287 writers.

4.3 MULTISTAGE CLASSIFICATION

A typical script that require a multi-stage classification is a script which containing complex structures or particular marks i.e. diacritics. But this complexity can not be generalized for all cases. Some complex scripts can be principally classified by a single classification task but some cannot. The example of the script with high complexity is the group of Indic scripts. This group consists of various scripts which are used on the Indian mainland such as Bengali, Devanagari, Gujarati, Gurmukhi, Kannada, Malayalam, Oriya, Tamil, Marathi, and Telugu. Characters of those scripts has a lot of variation with some curves as a dominant shape. With a possible

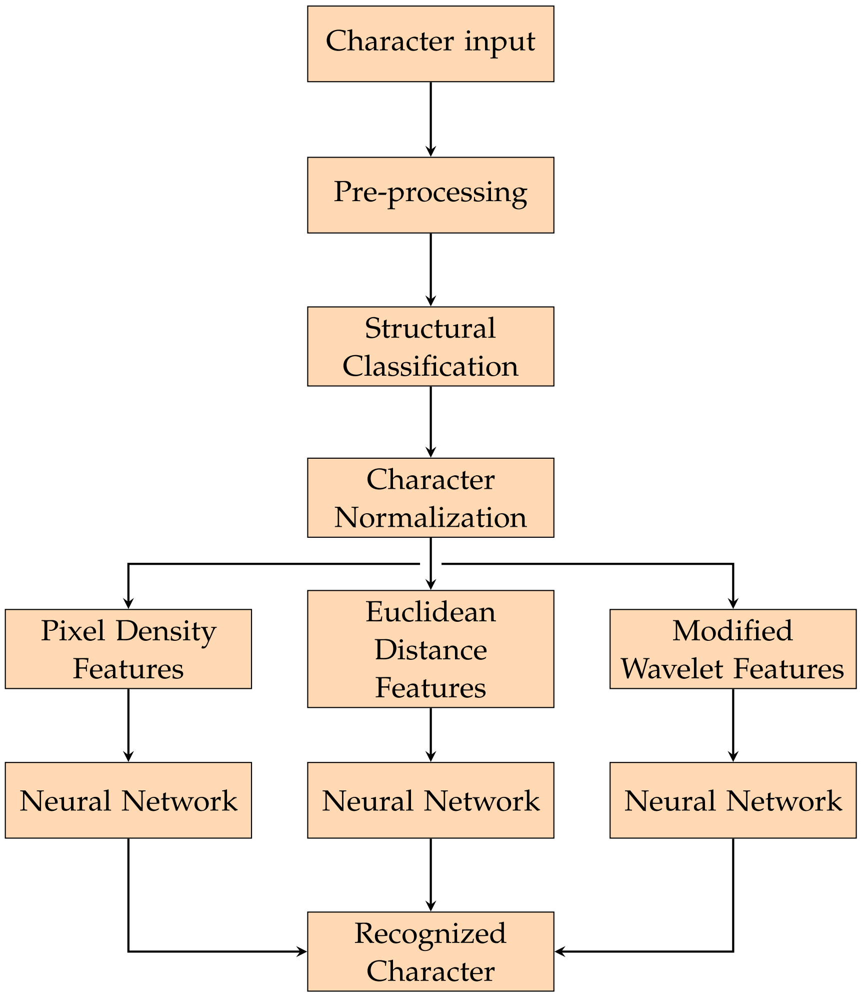


Figure 19: The design of multistage classification for Marathi compound characters [50].

combination among characters, the task of classification become more complex so that one-level classification becomes difficult. Therefore, a multistage classification can provide a feasible solution for this complexity problem.

One work of multistage classification had been done for Marathi script [50]. The script consist of 52 characters with 36 consonants and 16 vowels. Each character has a horizontal line on the top of each character. Characters are connected with each other to form a word by joining their header lines. A consonant can be connected by a vowel with a help of particular marks that can be located in line, at the top, or at the bottom of a character in a word. Moreover, its complex writing system enable to form a new specific symbol by combining two or more consonants. The last case is then called a compound characters in Marathi script. This compound character can be formed in several ways. The most common way is by removing header line and connecting it on the right side of another character. Another way of joining characters to produce a compound character is by joining both characters side by side or one on the bottom of another character. This circumstance may impact a low accuracy performed by a single level classification only. Therefore, to deal with this complexity as proposed in [50], the classification of this compound character was done as a multistage classification.

The idea of this multistage classification is explained in Fig. 19. There are two main stages for the classification of Marathi compound character. The first stage is called pre-classification by employing structural features. The use of structural features is demanded as Marathi compound characters comprise of many structure features such as vertical line, horizontal line, enclosed regions, end points, junction points etc. To efficiently performing classification in this first stage, all those features are initially grouped in two different types, the global and local features. The group of global features consists of the presence of vertical line and its position in the character, and the presence of enclosed regions in the character, while the group of local features consists of end points and their position in the character. Both groups are extracted as two consecutive sub-stages based on these groups. The first sub-stage extracts global features followed by a classification. The results from this sub-stage are then classified in the second sub-stage by using local features.

The second main stage is started by normalization the outcome of the first main stage in a fixed size. The feature of the second main stage is extracted from this normalized entity into three different features. Those three features are the pixel density, Euclidean distance and modified approximation wavelet. All three features vector are respectively fed into Neural Network (NN) resulting in three different outcomes. A final decision is made based on the majority voting of those three outcomes. In the case of all three outputs from networks are different, the decision is made according to output from the network with modified approximation wavelet. The accuracy of handwritten Marathi compound character by using this multistage classification is 97.95%. For a further information, the reader can refer to [50].

5 LAMPUNG HANDWRITTEN CHARACTER RECOGNITION

The idea of conducting research of the Lampung handwriting is encouraged by the fact that the research will open a preliminary development of a Lampung handwritten character recognition framework. The research introduces a basic framework containing fundamental approaches as pillars of the framework which may be enhanced in the future to be more powerful or extended to handle many problems or even exchanged to provide flexibility.

As described in Chapter 3, the Lampung text is not cursive script, approaches and methods from a general handwritten character recognition framework are not directly applicable to Lampung handwriting recognition. The reason is that the most of recent developments of offline handwritten character recognition is concentrated on cursive handwriting rather than non-cursive text. This can be a merit on one side but can also be a drawback on the other side. Hence, it is necessary to analyze and modify these approaches or methods to fit Lampung script. Another concern is that the Lampung characters are also accompanied by various diacritics. Each diacritic plays an important role for composing overall texts. Thereby, the presence of these diacritics should be modeled in the framework.

The following subsections describe a processing chain of Lampung handwritten character recognition in this framework. In each stage, specific methods or approaches are given and intensively discussed to cope with the task in the stage.

5.1 PREPROCESSING

The primary preprocessing tasks of the Lampung handwritten documents are a binarization, a Connected Component (CC) generation, a grouping, and size normalization. These four tasks can provide basic usable instances for the next stage in the handwritten character recognition pipeline. Other tasks might be needed as long as they support the goal of the current preprocessing task or they can give a significant contribution to further stage of the handwritten character recognition.

However, due to the nature of Lampung script, some tasks that are often applied to cursive script are not urgently done during preprocessing. For example, a slant normalization is not needed because Lampung script is a typical script without tendency of the slant. The Lampung character orientation mainly directs from left-bottom to top-right sideways (see Fig. 13 of Lampung characters in Chapter 3). Nevertheless, if someone writes Lampung texts with a slant handwriting style, his or her handwriting would not significantly differ to a common handwriting. Another task which can be switched on and off during preprocessing is the smoothing and sharpening. As one goal of the smoothing and sharpening is to remove a noise especially small spots, the smoothing and sharpening should not be executed when the goal of preprocessing is also to extract diacritics not only characters. The reason behind this idea is that the smoothing and sharpening will potentially remove diacritics as their shape is small.

The major preprocessing tasks of the Lampung handwritten document will be explained in detail in the following subsections. The order of tasks as explained in this subsections indicates the most feasible order for preparing better primitives to be fed into recognition.

5.1.1 *Binarization*

The raw image data is originally stored in RGB format. Thereby, the first step to be done is a binarization. In order to perform this binarization task, the process after the image acquisition is converting the raw image into gray scale and then it can be continued by a binarization.

Some algorithms to accomplish a binarization task like Otsu [38], Niblack ([37], cf. [23]), and Sauvola ([49], cf. [23]) are among the popular algorithms. The Niblack algorithm is chosen with a consideration that it is more adaptive to the local pixel. As explained in Subsection 2.2.2, the Niblack algorithm is a binarization algorithm with locally calculated threshold based on surrounding pixels in the window during computing operation. With this manner, binarization is expected to be more representative according to local pixel and at the end producing the best result of these algorithms. The realization of binarization in this work was done by utilizing the algorithm offered by the ESMERALDA tool [12] that provides various approaches for binarization. Among of them, the modified Niblack algorithm from this package mainly produced the best result. Results of binarization are shown in subsection 6.2.1.

5.1.2 *Connected Components*

The lampung script is a non cursive writing system. Hence, from the source of the handwritten document, each character as well as each diacritic can be contrasted to its background as single components. In fact, the extraction of the Connected Component (CC) from the document will implicitly complete the segmentation task of the characters and diacritics.

However, there are exception for some cases. For example, the segmentation fails if deformations occur such as two or more characters touch with each other, two or more diacritics touch with each other, diacritics are connected to a character, the noise is connected to characters or to diacritic, etc. In this case, an additional effort is needed to separate those touching objects. Since the occurrence of this case in Lampung handwritten document was presumably low, there was no extra effort on behalf of separation after the generation of connected components. Those deformation objects will be considered as the noise.

The extraction of CCs can be accomplished by two algorithms. First is called the one-pass algorithm and the second algorithm is called two-pass algorithm. In this work, CCs as representation of character or diacritic primitives including the noise had been extracted by applying two-pass algorithm which has been discussed in Sub Section 2.3.2.

All produced CCs were not altered. They were stored in their original shape and size, just as they were obtained after generation from original document images. In this form, CCs are flexible to be transformed in any other forms based on the needs of the next step.

5.1.3 *Separation of Connected Component (CC)*

As the segmentation has been done at the level of CC's extraction, the resulting CCs would consist of two type of instances along with unwanted instances. Those two instances are regarded as characters or diacritics along with unwanted instances as noise. To distinguish these prospected instances, a separation procedure on all of CCs is applied. This separation is accomplished for characters and diacritic instances respectively through two independent procedures.

In the first turn, a separation scheme was applied to obtain the instance of characters and drop others. The character and other CCs can be distinguished based on its size, aspect ratio, and pixel density. Therefore those three parameters were tuned to control separation process. To get a complete illustration regarding this tuning, the reader can refer to Subsection 6.2.2.

The second round of separation process was run to discriminate diacritics and discard others. The carefulness of this separation procedure becomes a big concern in Lampung handwritten character recognition since the size of diacritics is relatively small. Because of their size, diacritics potentially resemble noise and they may be removed during this process. Another problem is that the separation could not be run straightforward at once since there is one diacritic class which significantly differs to other diacritic class. The distinction occurred among diacritic nengen (J) and six other diacritics, particularly the difference in aspect ratio. The height of a diacritic nengen is like the height of a character but its width is like an ordinary diacritic. Whereas the height and width of an ordinary diacritic is much shorter than the height and width of a character. With this nature, the separation of diacritics can not be finished all at once. Therefore, the task was necessarily run twice for each possibility. The separation step along with parameters tuning are also provided in Subsection 6.2.2.

5.1.4 *Normalization*

The outcome of the grouping is CCs with different height and width. All those CCs have been stored in their original size and shape so that they are in the state of "*ready to use*" or "*ready to modify*". If they need to be modified into specific dimension prior to feature extraction, they were mapped into particular dimension by applying a linear normalization. Since CCs indirectly represent character and diacritic instances, the normalization must be targeted for both.

Concerning character, initial analysis of bounding boxes of some CCs from every document prior to normalization process had been conducted and noted. It could be highlighted that sometimes the height is longer than the width and vice versa but majority they are approximately close with each other. In other word, the aspect ratio of bounding box is almost one. Based on this fact, it would be better to normalize CCs by imposing the same length for height and width for output of normalization. It can preserve the shape details as much as possible. Therefore, the process of normalization reproduced all CC's character bounding boxes into a square.

Similar to character instances, diacritic instances as the second instance within the set of CCs also encountered a normalization process. In this turn, a visual reasoning on CCs of diacritic instances indicated a risk of significant distortions after normalization due to a tiny size of the original CCs and variability in aspect ratio. To reduce this drawback, each CC's bounding box was firstly extended by circling its bounding box with one-extra pixel perimeter. The normalization of diacritic instances were then applied over this new size CC's bounding box.

The normalization purely relied on a linear function to map the pixel by using formula 2.5 and 2.6 in subsection 2.2.3. After normalization, the character and diacritic are in a fixed size determined prior to normalization. The size of normalization output for character bounding boxes was estimated from the average size of all original bounding boxes, while the size of normalization output for diacritic bounding boxes was initially set to a fixed size from the beginning of normalization.

5.2 LABELING CHARACTERS

After the preprocessing the Lampung documents had been completed, a new collection of the Lampung handwritten characters has been documented for the purpose of research. But the classic problem appears for new introduced character sets as there are no labels for such collections while labels are needed for training and testing recognizers. Hence, the labeling task has been addressed for the Lampung dataset collection.

There is no fully automatic method for labeling but on the other hand it is too naive if all character sets in the collection are labeled manually. Many datasets that are publicly available for example in [3] [26] [35] mainly set the labels manually which is very time consuming, tedious, and costly.

To reduce human involvement in the labeling task while keeping a reasonable speed and cost as noted in [52], a semi supervised approach for character labeling of the Lampung handwritten character was proposed in [57]. The main concept behind the approach is to give the label for each cluster of each data representation and then determine the label by voting to have a final label. By handling this way, the human effort will be minimized during the labeling process. The complete process of the approach consists of three consecutive stages as follows:

1. Compute different feature representations.
2. Cluster and label the sample in each representation.
3. Vote the label.

The general overview of the approach can be observed in Fig. 20 and the following subsections describe the approach in more detail.

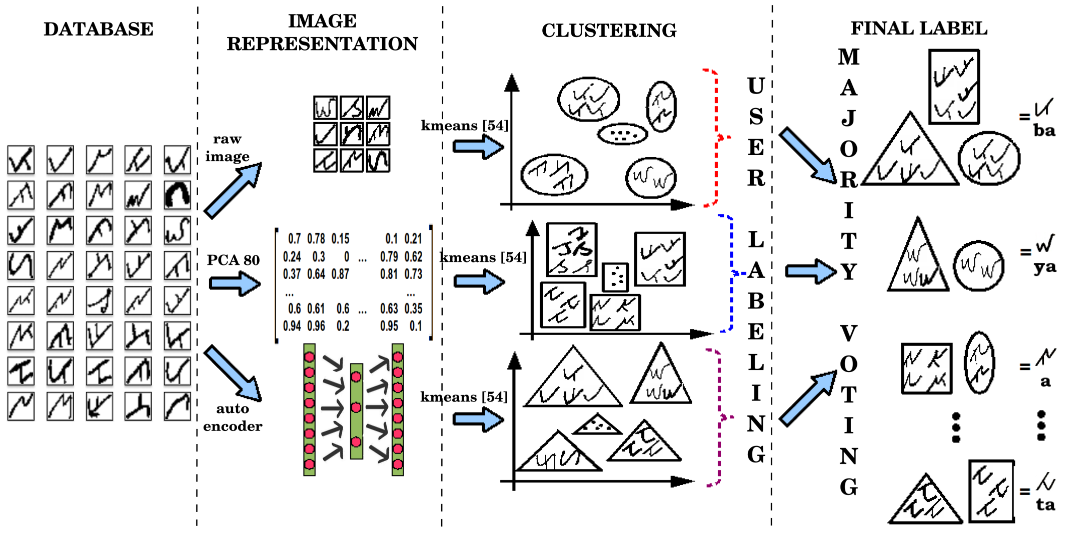


Figure 20: General view of Semi-automatic Labeling of the Lampung character (Taken from [57]).

5.2.1 *Data Abstraction*

The initial step of the system is to compute some feature representations to get different representations of the data. This strategy is implemented to provide diverse input for a multi-view voting scheme so that complementary representations [25] and classifiers can be ideally combined in a labeling system.

The number of feature representation is not restricted but it is clear that more than one representation will be needed. The more the representation, the more complementarity can be achieved during the labeling process.

As illustrated in Fig. 20, three kinds of representations are considered for labeling the Lampung script. Besides to label as little as possible, the number of representation was chosen three because to enable a simple majority voting scheme with a minimum number of representations.

The first feature representation is using pixel values which explicitly represents the value of foreground and background pixel of a binary image. This pixel value is extracted after a binarization process on the original image following by a normalization to 20x20 pixels. All pixel values of the image were concatenated forming the series of 400 binary values. Although this representation looks very basic way, it was inspired by successful works in digit recognition in [26] [56].

The second feature representation uses a reduction approach over the original observation. In this regard, a simple and widely used method, PCA (cf. [4, p. 559-570], cf. [10, p. 115-117] was chosen to transform the original pixel data such that the dimensionality reduces and the first principal component preserves the maximum variance.

The last feature representation also uses another reduction scheme that is called autoencoder network [19]. The reduction strategy is based on a multilayer neural network with the ability to reconstruct the original input during training. And at the end of the operation, the overall procedure will generate a vector with small dimensionality but still inherit the properties of the original pixel data.The reader can go into detail about this scheme by referring the article [19].

Among those three representations, the pixel value representation indicates a very raw image representation while the last two representations characterize two different reduction strategies. All of them define three different type of character representations of Lampung handwritten data. In fact, the certain level of complementarity in those representations can be assumed from the fact of those differences.

5.2.2 *Clustering and Labeling*

After creating the multiple representations, the process will be continued by a clustering to get agglomerations of the Lampung character candidates (see the third column of Fig. 20). To facilitate this task without human involvement, each representation from the first stage is agglomerated by using an unsupervised clustering method, Lloyd algorithm [31], which is often also referred to k-means. The easiness and simplicity are the spirit of use this algorithm instead of other algorithm for clustering. The parameter k of k-means indicates the number of the clusters or agglomerations for which the data representation need to be partitioned. The higher of k, the more refined agglomerations can be reached.

Once the clustering has been finished, each sample data in each cluster will gain the verdict of a character derived from the cluster centroid. However assigning the Lampung character label to them can not be achieved during the clustering process but it needs the human intervention because it is related to an expert that can interpret each cluster by a visual examination as being a Lampung character. In another word, the label of the cluster must be done manually by an expert for total number of clusters indicated by parameter k of term k-means. In the case of this work, the Lampung handwritten characters were labeled in 11 classes.

The overall process in this stage consisting of an unsupervised clustering and a manual labeling is considered as a semi automatic process. The human effort in labeling task is reduced to label the centroid of the cluster. The number of labeling operations for each representation is only k which is insignificant compared to the total Lampung data sample that might be thousands. Since there are 3 representations in this work, there will be in total $3k$ labeling operations for the Lampung handwritten data.

5.2.3 *Voting*

The previous stages, as depicted in Fig. 20 at the second and third column, generated three labels for each Lampung data sample. Considering those labels, a decision must be done at the last stage (see last column in Fig. 20) to determine a final label for each Lampung data sample by a voting scheme [25]. The voting output would be accepted as the label for each data sample.

Let the label be denoted as a d-dimensional binary vector $[l_{i,1}, \dots, l_{i,d}]^T \in \{0,1\}^d$, $i = 1, \dots C$, where $l_{i,j} = 1$ if classifier C_i labels a samples p in class ω_j and 0 otherwise.

The ensemble decision could be based on unanimity vote where the label will fall to class ω_k if all classifiers decided to class ω_k. This decision is formulated by,

$$\sum_{i=1}^{C} l_{i,k} = C. \tag{5.1}$$

However, it might necessary to adopt another scenario as a second choice for an ensemble decision such as *simple majority vote*. In this scenario, the label of a cluster can be decided whenever the majority classifiers choose the same label. The formula for this decision is in the form,

$$\sum_{i=1}^{C} l_{i,k} \geqslant \lfloor \frac{C}{2} \rfloor + 1 \tag{5.2}$$

Since this procedure use three different representations, those are regarded as three different classifiers during labeling process. With the unanimity vote, a selected label can be chosen if all those classifiers vote this label. Meanwhile, the simple majority vote will consider the label if at least two classifiers have the same vote as shown by equation 5.2.

Although the ensemble decisions in this work seem to be very common tasks but according to the approach explained above, there is a fundamental distinction between this strategy and the other ensemble learning strategies. The difference between other strategies and this current solution is on the purpose of the voting scheme. Here, the voting scheme is actualized only to label the training data and a classifier is built on top of this label information. Otherwise, voting schemes are often used in classification ensemble. In summary, this method can be considered as a novelty approach for semi-supervised labeling with less human involvement. The analysis and evaluation of the result concerning this approach is discussed in section 6.3.

5.3 RECOGNITION OF THE TEXT

The Lampung handwritten character recognition is still in the beginning state of the research. It is still a long way to reach mature state-of-the-art like the Roman-based character recognition. However, it is undeniable that the Roman-based recognizer may also impact the development of recognizer for Lampung handwritten character.

As explained in the beginning of this thesis, Lampung handwritten character is non-cursive character where each Lampung character separately stands as a single element in the character formation. There is no way to make them cursive like Roman-based handwriting. This is indeed a positive circumstance during development of the recognizer because the task of character segmentation from a bigger blob composition at least do not have to be deployed. Therefore, it can reduce one work. Nonetheless, the real challenge in development of Lampung handwritten character recognizer is the presence of a tremendous amount of diacritics. They must be attached to their respective characters which are not a simple task indeed.

The following subsections explains the works on recognition of the Lampung text, particularly three independent tasks. The first one is the recognition of the basic character with special feature representations. In this step, the discussion comprises of the procedure of feature extraction, the chosen classifier with experiment setup and the recognition. In the second step, the subsection discusses about the association between characters and diacritics. The idea of this work starts by choosing diacritics and selecting one character over some possible characters nearby. Then an approach to associate a diacritic to a character is presented. The last step is focused on the topic of building a recognizer for the complete Lampung handwritten text. In this step, a basic character is associated to all possible diacritics nearby instead of only one-to-one association as given in the second step. Accordingly, the product of this association represents a complete model of the text composition in the Lampung writing system. Therefore, the result of the last step plays an important role in Lampung handwritten character recognition.

5.3.1 *Basic Character*

The recognition of the basic character of Lampung handwriting [20] is the second milestone in the research on Lampung handwritten character recognition beside the labeling work on Lampung Connected Components (CCs) [57] as the first one. The success of this recognition had brought achievements on two aspects which are the introduction of a novel feature representation for Lampung handwritten character recognition and supplying the Lampung dataset for various research of Lampung handwritten character recognition.

As described in section 3.2, Lampung script consists of 20 basic characters. Therefore, the recognition of the Lampung handwritten text should be addressed by identifying 20 character classes. Nevertheless, as illustrated in the early work of Lampung character labeling in [57], some characters have only a tiny difference between each other and for this reason, the labeling was not directly done for those 20 character classes but instead 11 character classes. The idea of this simplificatin as reported in [57], was to group some resemblance characters as one class so that the number of character classes to be recognized was reduced.

This recognition task would consider the same number of classes as used in that work. Hence, the recognition of the Lampung handwritten text in this work has been focused on identifying 11 character classes. The recognition of these character classes as illustrated in [20] is explained in following subsections.

5.3.1.1 *Feature Representation*

Feature extraction is one of the important steps during the recognition scheme because it generates feature representations which denote the character itself in the form of a numerical pattern. During recognition, characters will be represented by feature representations. Thus, feature representations become a critical point in a handwritten character recognition pipeline since it will affect the performance of the overall recognition.

Feature representation in a recognition can be generated from well-developed feature extractors by other researchers or invented as a new feature representation or even combinations of both. An important thought when dealing with feature representations is that they must be relevant as much as possible to the nature of the character so that they can positively impact the performance in recognition. For Lampung handwritten character recognition, the use of existing feature representations is more reasonable to be applied.

Recently, various well-defined feature representations were introduced that can be applied to the recognition task. From many kind of feature representations in literature, four of them were selected for the recognition of the Lampung handwritten text in this work. These feature representations are *branch points* [8], *end points* [8], *pixel densities*, and *the Water Reservoir (WR)-based* [7], [40], [41], [42], [43], [44] feature. The reason behind the decision of using the selected features are because a strong correlation between the feature representations and the nature of the Lampung character. In another word, the characteristic of selected features reflect the most-related attributes of Lampung characters.

The branch point is good for representing the branch line in the body of the Lampung character stroke while the end point can notice the end line of the Lampung character which basically the effect of non cursiveness. Branch points or an end points can be identified after converting image into a skeleton image. A pixel on the skeleton is called a branch point if it surrounded by three pixel neighbors. While an end point is defined as a pixel along the skeleton having only a single pixel neighbor. In term of graph theory, a pixel or a vertex is called a branch if it has degree of three while an end if it has degree of one. The pixel density would provide information about the general concentration of the foreground pixel in some identical zones within the character bounding box. And finally, the WR-based feature would be a special feature due to each Lampung character contains at least one cavity that resembles to that reservoir.

In order to extract the features of branch points, end points, pixel densities and the WR-based features, each normalized CC needs to be transformed into a skeleton before extraction. Then, the square area of each CC is partitioned into some smaller zones to shift the level of computational complexity from a complete area into a smaller scale zone which accordingly can simplify feature extraction procedures. This mechanism is particularly applied for feature representations of branch points, end points, and pixel densities.

Feature Extraction of Branch Point, End Point, and Pixel Density

Concerning the zone, a full area of the CC image with size 20x20 pixel was broken into small zones with size 4x4 pixels. Hence, one CC has 25 identical zones which each zone contains 16 pixels.

Fig. 21 shows a CC bounding box of the Lampung character *"a"* along with its 25 zones. The writing order of the feature values is aligned to the direction starting on the topmost level from left to right. After one level finished, this procedure is

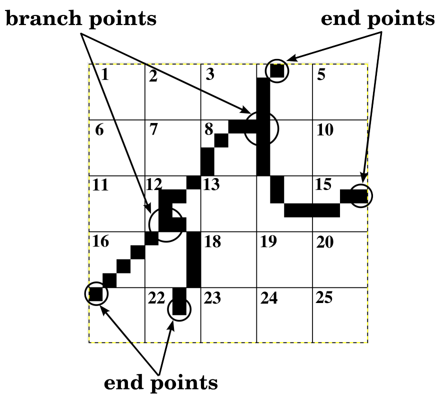


Figure 21: The sample of branch points and end points in zoning areas on the image skeleton of character *a*.

repeated to one level on the below until the last level on the bottom. From each zone, the number of branch points, end points, and pixel densities were respectively counted and then concatenated into a series of feature values. Take an example of branch points in this figure. One branch point is located on the segment 8 and 12. Consequently, the value on those position will be set to one. However, to provide a scale invariant feature representations, those values were normalized by the total number of pixels in each zone, which is 16 pixels. This normalization results the values between 0 and 1. Moreover, end points in this figure can be found in four segments, 4, 15, 21, and 22 which respectively contain only one end point. This is also normalized with respect to total number of pixel in each zone. The feature of pixel densities is counted and normalized in the same manner. Since each representation has 25 values obtaining from each zone, the concatenation of three representations yields 75 values.

This feature representation was then used for the recognition experiment. The experiment of using solely this representations is interesting since those features are relatively simple to be extracted.

Feature Extraction of Water Reservoir (WR)

The idea of imitating the Water Reservoir (WR) principle in handwriting character recognition is not a new approach. Some applications of WR principle can be noticed from successful works in [7], [40], [41], [42], [43], [44]. As explained in those papers, it was used as a segmentation method. In this work, the WR principle was used in a fundamentally different manner. It is applied to Lampung handwritten character recognition for serving the feature representation instead of segmentation.

To extract the WR-based feature, the zoning areas are not needed. The feature representation can be extracted directly from the normalized CCs by applying an invented algorithm that is named *cavity-searching*. The work of this method in general is given in Algorithm 2. This cavity-searching works by tracking the skeleton of character image pixel by pixel. More explanations about the algorithm are given in the next paragraph with Fig. 22 illustrating how the algorithm accomplishes this task.

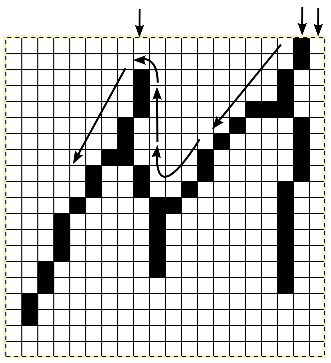


Figure 22: The algorithm of cavities searching on the image skeleton of character *na* to be assigned for the WR-based feature representation.

Fig. 22 illustrates how the algorithm accomplishes this task.

- From the top-right cell of a character bounding box, the tracking is started. It goes to downward unless it found a foreground pixel. In the case of a foreground pixel is found, it will continue to inspect foreground pixels on its neighbors until the last pixel on the character skeleton.
- The tracking process will record transitions of foreground pixels. Transitions are grouped based on their direction which is downward, upward, and horizontal.
- Set the pointer to upper right corner, and select a transition to downward from the record buffer. The transition to downward can potentially be a candidate of a cavity.
- A cavity is identified if the next transition is upward and the algorithm will repeat the same process to the next processed pixels. While if the next transition is a horizontal line, the algorithm will start identifying the change of transition again. However, if the next process does not find a further foreground pixel, the algorithm end.

In the analogy of a Water Reservoir (WR), each cavity is poured by water until the water level reaches the lowest end point among of two end points in one cavity. The area on the character skeleton that is full of water is then the so-called water reservoir. To obtain a best understanding, Figures 23 provides a proper visual illustration.

Some measurements for feature representations were calculated during the inspection of all cavities in a character skeleton image. As indicated in aforementioned algorithm, the height or the depth of the reservoir are noted during the inspection of a cavity. The inspection was supposed to measure the width as well. Nevertheless, the shape of reservoir unfortunately does not allow to measure it in straightforward manner. To overcome this problem, alternatively the volume of each reservoir is counted and then the width can be estimated by a division between the volume and the height. And finally, the gravity center of the reservoir is also identified during

Algorithm 2 Cavity-Searching Algorithm

```
 1: Put pointer to upper right corner
 2: Track all pixels of skeleton
 3: Record transitions
 4: Set pointer to upper right corner
 5: if no transition to downward left? then
 6:     stop
 7: end if
 8: Select a transition to downward
 9: Identify the transition change
10: if upward then
11:     cavity is found
12:     reset parameter and go to line 5
13: else if horizontal then
14:     go to line 9
15: else
16:     reach the last pixel, no cavity and stop
17: end if
```

this inspection. The gravity center (x_0, y_0) of area in a binary image is computed based on the formula,

$$x_0 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} jB[i,j]}{A} \tag{5.3}$$

and

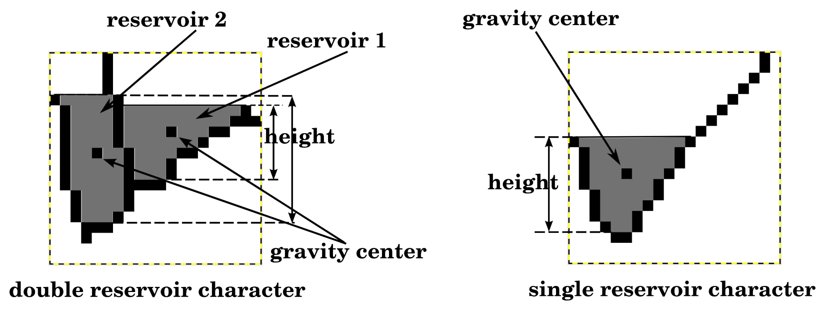
$$y_0 = \frac{\sum_{i=1}^{N}\sum_{j=1}^{M} iB[i,j]}{A} \tag{5.4}$$

Where A is the area of the region which can be computed by the following,

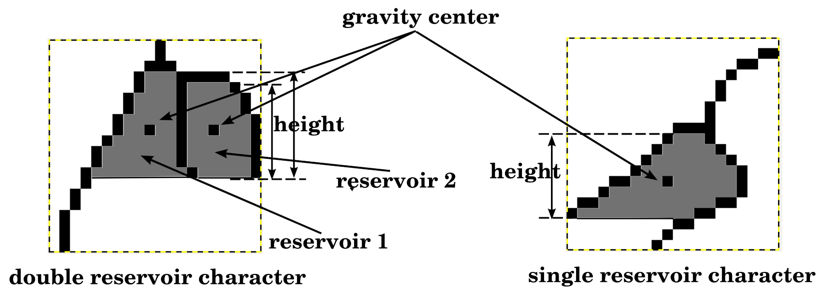
$$A = \sum_{i=1}^{N}\sum_{j=1}^{M} B[i,j] \tag{5.5}$$

Take a closer look on the skeleton character image with their reservoir(s) in Fig. 23, it can be concluded that there are two kinds of reservoir, the top and bottom reservoir. A top reservoir, as shown in sub figure 23a, is opened to the top so the water can be filled from upward. While a bottom reservoir, as indicated in sub figure 23b, is opened down enabling the pouring of water after rotating the reservoir to 180^o. In feature representations, both types can be discriminated by a positive one (1) for the top reservoir and a negative one (-1) for the bottom reservoir.

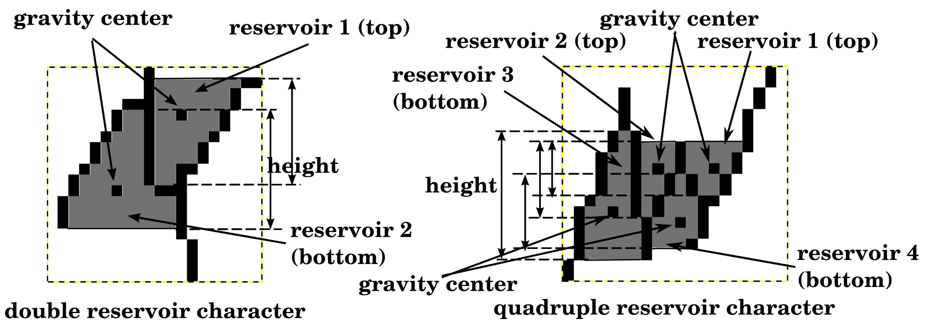
As all needed measurements had been gathered during the cavity-search algorithm, the next step was to set them into a feature representation. To express a feature representation of a reservoir, an arrangement of all measurements was formulated as 6 consecutive numbers comprising of:



(a) Top Reservoir



(b) Bottom Reservoir



(c) Top & Bottom Reservoir

Figure 23: Different types of reservoirs in some samples of characters [20].

- The first value symbolizes the type of the reservoir. As explained the type of a reservoir can be either reservoir with the open part faces up represented by 1 or with faces down represented by -1.
- The second and third values are a pair x and y indicate the coordinate of the reservoir's gravity center after normalization with respect to the character height and width. This normalization is to transform the coordinate into a certain range value which represents a uniform measurement. This way consequently change coordinates into a scale invariant value.
- The fourth value denotes the volume of the reservoir. Since the reservoir in this case is only a 2-D object, the volume is represented by the number of pixels inside the cavity.
- The last two values are assigned for the height and width of the reservoir.

Note that all those consecutive numbers are integers except the reservoir's volume. The value of reservoir volume is a floating point because the value of size is a result of a normalization by the total pixels in the image.

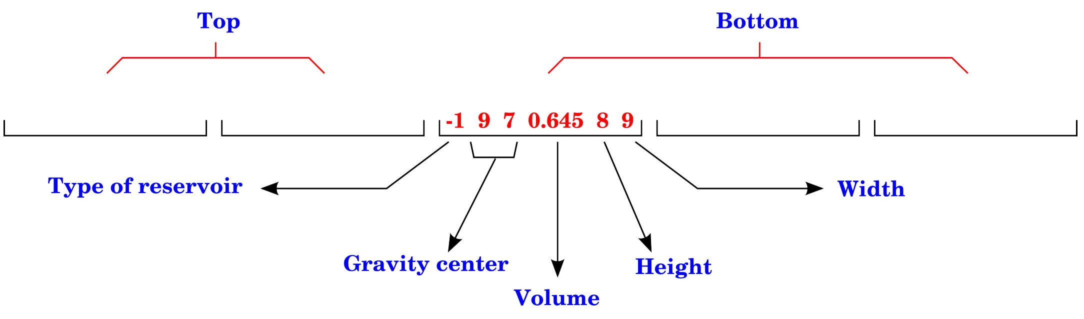


Figure 24: Feature representation of a Water Reservoir (WR) with five tuples for a Lampung character.

This integer series only represent one reservoir. Whereas Lampung script mainly has more than one reservoir. Only three characters (ga( ), pa( ), and da( )) have one reservoir and the rest have more. The observation on Lampung characters shows that a character can have a maximum of 2 top and 3 bottom reservoirs. Based on this fact, the feature representation must be constructed by 5 tuples as a concatenation of those kind of reservoirs where each tuple consist of 6 values representing the characteristic of one reservoir. In the overall series, the appearance of top reservoirs precede the appearance bottom reservoirs. The total length of the feature representation for the WR-based feature of each Lampung character in this series is therefore 30 values. See Fig. 24 for the composition detail of the feature representation.

Nevertheless, a further observation on characters in dataset pointed out that the number of reservoir could be more than five. Reasons of this occurrence are due to variation of personal writing styles, implication of normalization process, effect of noise, etc. Since the tuples for feature representation are only 2 for the top reservoir and 3 for the bottom reservoir, those are selected by their volume. The maximum volume will be considered as reservoir to be inscribed in tuples. The argument for this rule is that the big volume reservoirs really belong to character while a small volume can be effect of aforementioned factors. Hence, small volume reservoirs will be ignored whenever the tuples are already occupied. On the other hand, if a character contains less reservoirs than the maximum number of provided tuples, the remaining tuples can be set to zero. In a very bad situation, sometimes the feature extraction procedure failed due to unconnected components as the result of size normalization. In this case, a character does not have reservoir at all. To avoid a distortion during recognition, that respective feature representation will be zero.

The feature of WR is also included in the recognition experiment as a complement for the feature of branch points, end points, and pixel densities.

5.3.1.2 *Character Classification*

From the feature extraction step, there were two group feature representations which accordingly led to two experiment series, one for each representation. Moreover, a new feature representation could be composed by concatenation both feature representations. This new concatenated feature representation also led to the third series of experiments.

To perform experiments, a multilayer perceptron Neural Network (NN) [4], [10] [8], [54] was applied to train the different classifiers. The architecture of this NN was organized as three layers network consisting of input layers, hidden layers, and output layers. The algorithm for training was driven by the resilient back propagation. While neurons use sigmoid function to handle the activation process.

The first experiment involved the first group of feature representations i.e. branch points, end points, and pixel densities. In this regards, the input layer was set to 75 according to dimension of this feature representation. While the output was set to 11 since the recognition was done for only 11 character classes, simplified from the total of 20 character classes of Lampung script. The reason of this simplification is to combine some resembling characters into one class to reduce the complexity of recognition tasks. For the purpose of informal experiment runs, the hidden layer was set to several configurations as desired but at least equal to the number of input layer.

In the second round, the WR-based feature representation became the target of experiments. The input layer was assigned 30 neurons, equal to dimension of the WR-based feature representation. The output of the network was still 11.

In the last configuration, both feature representations were merged into one piece with a total of 105 values. Based on this dimension, the input layer was specified to 105. This combination scheme aimed at observing how well the combination of statistical features (pixel densities) and structural features (end points, branch points and water reservoirs) could impact the performance of the recognition.

5.3.2 *Character-Diacritic Pair*

One critical challenge in a Lampung handwritten character recognition is to associate a single diacritic or multiple diacritics onto a character. The problem in an unsupervised character recognition is that any diacritics may be surrounded by some characters which makes the binding of a character and diacritics a difficult task. Consequently, one indispensable concern in the Lampung recognition is to handle the association of the character and any dedicated diacritics as one compound character in a complete recognition. However, a complete association of the character and any diacritics as a compound character is not a simple task to be done directly at once. Therefore, it is necessary to perform a less complicated association work to bridge the task or at least to know the appearance of obstacles during the association process such that a further process for handling a compound character can be accomplished properly. In the following subsection, a simple association model between a character and any diacritics is described as a pairwise instance. The model is generally built as a one-to-one relation between the character and a diacritic where each pairing is determined based on a statistical measurement computed among the character and diacritic [21].

5.3.2.1 *Feature Representation of Pairing*

With respect to the association process as indicated in [21], the main point of view has been shifted from a character-wise to a diacritic-wise. Thus the process is inverted by firstly looking at the diacritic and then identifying the character as the companion of this diacritic. The important issue during this association process is that there are composition of characters with more than one diacritic nearby, but no diacritics associated to more than one character. In this case, each diacritic will be handled separately as one independent instances so that the number of independent instance will be equal to the number of diacritics. As the pairwise instance always consists of a single character and a diacritic, the character without any diacritics can be considered as out of discussion for this subsection. Thus it can be excluded in this analysis.

A pairwise instance can be represented by a feature vector which represents the relation of a character and a diacritic in form of a numerical value. To assist of the pairing process, the paired value can be obtained by the following procedures.

For the purpose of technical illustration, Fig. 25 will be treated as a visual aid of the following explanation. A diacritic under consideration is selected and its geometric center is computed.

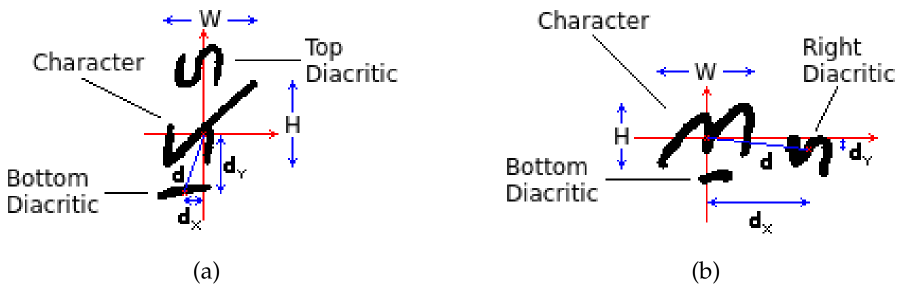


Figure 25: Sample of two compound characters of Lampung handwriting [21] (a) the compound character *bur* built by the basic character *ba* and a top and a bottom diacritic and (b) the compound character *nuh* formed by the basic character *na* with a bottom and a right diacritic.

After localizing geometric center of the diacritic, the next step is switched to the character. As for the diacritic, the center of this character is also examined and set as a Cartesian coordinate $(0,0)$ to be an anchor of the character. Then a point to point distance is computed between the center of the diacritic and the character. This distance is projected along the X (d_x) and Y (d_y) axis. Each projected distance is respectively normalized by dividing to the side length of parallel dimension of the character. Hence the projection along the X (d_x) axis is divided by the width W and the projection along the Y (d_y) axis is divided by the height H of the character. The mathematical formula produced by this procedure is represented as:

$$x = \frac{d_x}{W}, \quad y = \frac{d_y}{H} \tag{5.6}$$

Both values in Eq. 5.6 can be rewritten in form of a vector $v = [x, y]$ which implicitly represent the coordinate of the diacritic over the character in a two dimensional normalized form. This vector is set as a feature representation of the character-diacritic relation and becomes a basis of a further exploration to determine the desired association.

5.3.2.2 *The Association Model*

The vector as described in the aforementioned subsection only represents one character-diacritic relation. While near a diacritic, some characters may be close by and among those only one should get associated to the diacritic. Hence the process of pairing must consider all respective characters nearby the diacritic and compute all vectors to them one by one. The role of the vectors is to indicate a relation with the diacritics as a central point of inspection.

As not only one character but some characters are considered to be evaluated, this will lead to a particular approach by involvement of those vectors to decide one character over the other. For each candidate character c_j, the probability of pairing is computed in terms of a pairing probability by applying a Gaussian mixture model:

$$P(v|c_j) = \sum_{i=1}^{k_j} w_{i,j} \mathcal{N}(v|\mu_{i,j}, \Sigma_{i,j}) \tag{5.7}$$

Where:
k_j: the number of components for character c_j
$w_{i,j}$: the weight of component i
$\mathcal{N}$: the Gaussian normal distribution
$\mu_{i,j}$: the mean of the component i
$\Sigma_{i,j}$: the covariance of the component i.

These elements are estimated from a training dataset during the training phase. For an initial process, the training dataset is clustered by applying k-Means [34] and then means and covariances are computed. To improve these parameters, an additional optimization step is carried out with respect to the different character distribution by applying EM-algorithm [9]. The usage of EM-algorithm is expected to generate means and covariances that are much more representative of the data.

Since the diacritic is surrounded by many characters, only s characters are chosen to be reviewed. The probability of pairing those characters with the diacritic are respectively computed. All possible pairing probabilities are then examined with each others and a decision is made by selecting the pair with the maximum conditional likelihood probability,

$$s = \arg\max_{s} \left(P(v_s|c_s)\right) \tag{5.8}$$

Where:
v_s: feature vector derived from the pair of a diacritic and the character candidate c_s: character candidate to be associated with the diacritic

This pair is considered as a correct association between character and diacritic under an assumption that the maximum likelihood probability will lead to a minimum error rate.

However, this approach can trigger a technical problem during parameter computation if the sample of a particular component is extremely small. In this circumstance, such a computation can only generate parameters rather locally for that small sample. While during the estimation of unknown data, the usage of Gaussian mixture with these parameters can introduce a bias which result a significant error of the estimation. To cope with this situation, parameters of Gaussian mixture can be approximated by computing complete training set. This can be formulated as the marginal density of $P(v_j, c_j)$ or approximated by estimating the model parameters character independently:

$$P(v) = \sum_{j=1}^{|c|} P(c_j)P(v|c_j) \approx \sum_{i=1}^{n} w_i \mathcal{N}(v|\mu_i, \Sigma_i) \tag{5.9}$$

Here n denotes the number of mixture components computed on the complete training set and $|c|$ denotes the set of characters.

5.3.3 *Syllable Level*

The work of association character-diacritic of Lampung handwriting was previously designed only for a simple composition as one-to-one relation between a character and a diacritic. This association apparently does not fully characterize all complete text units in Lampung writing system. A complete text unit in Lampung writing system is in form of a syllable which can not be composed by only a character and a diacritic. There are several possible compositions to form this syllable for example a single character only, a character and a diacritic, a character and two diacritics, or a character and three diacritics. In fact, the main topic of Lampung handwritten character recognition is not only recognition of these two involved elements, i.e. characters or diacritics, but also the most challenging task in this recognition is to recognize syllables based on their building blocks as a representation of a complete unit model.

As a unit model consists of some components, it can be considered as a compound character which can be formed by one model among several compositions as mentioned in previous paragraph. To recognize this compound character, several elementary tasks must be performed one by one in a consecutive order. Each elementary task handles a specific target to simultaneously establish a recognizer of those syllables. The following parts comprehensively explain each of those elementary tasks with a necessary analysis and discussion.

5.3.3.1 *Recognition of Basic Components*

In order to recognize a complete composition of characters and diacritics as a building block of syllables, the recognition of each basic component is needed as a baseline for the performance of the combination. The recognition of individual component encompasses the recognition of characters, diacritics, and one-to-one association of character-diacritic. Concerning this baseline, results from existing work are reused during this sub-step as long as it meets the requirement.

1. Recognition of the Characters
   Formerly, recognition of Lampung character had been done as reported in sub section 5.3.1. However, this result could not be used as a baseline indicator in this step since the recognition was counted only for 11 classes. While in this task, the recognition needed to cover all 20 characters as many as total characters in Lampung script. In practice, the recognition of character could only be executed for 18 character classes since two of them were not counted as basic characters so as both characters were excluded. Such a problem occurred because both characters consist of two separated components and the recognition task had never recognized both components in one piece. Instead, the recognition regarded both components as two distinctive components where each component coincidently resemble to another single-component character. These characters are character "ra" ( ) and "gha" ( ) that are formed by concatenation of character "ga" ( ) and "pa" ( ). To deal with this circumstance, a post-processing step is absolutely needed to link both components. This is discussed in part 5.3.3.2 within this chapter.

   The recognition of 18 character classes will re-apply the same characteristic of the recognition of 11 character classes. Although a recognition without enhancing features and with the expansion of classes from 11 to 18 may deteriorate the performance, it is still worth to check its performance. Therefore, the feature of Water Reservoir (WR) and branch points, end points, pixel densities will be included as well.

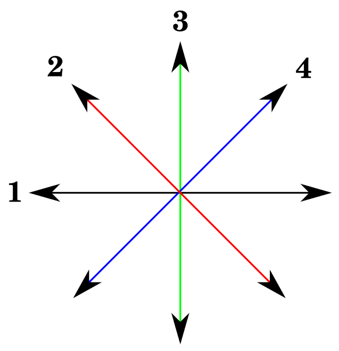


Figure 26: Integer codes for each direction in a chain code. Left and right direction are represented by code 1, diagonal of 45° and 225° direction are represented by code 2, upper and lower direction are represented by code 3, and diagonal of 135° and 315° direction are represented by code 4

In order to anticipate the degradation of the recognition performance, the feature representation for this recognition also uses the chain codes. They are extracted from the contour of the normalized binary image. The contour represents the outermost border of the character shape. The chain codes itself is derived from the direction of the border edge of the contour in each their pixel coordinate. Directions are grouped according to 4 or 8 directions. Each of these directions is encoded by a number to represent a unique direction.

With respect to this work, chain codes with 4 directions are used. The chain codes which are discovered from a Lampung character image can reflect the nature of the character. It can transform the pixel body of the character into values that identically represent the direction of the character shape boundary. This characteristic can cause two consequences during the character recognition. It enables a high accuracy on recognition, but on the other circumstance, it can distort the performance when noise involved. The source of noise can be originated from the raw image source or the effect of preprocessing.

The image sources for feature extraction are taken from the CC of the binary images that are normalized with the dimension 32x32 pixels. Each of them was sub-sampled from its original size into a small grid area with dimension 4x4 pixels. Hence, the total areas for a single image source is 64 zones with the size 16 pixels.

As the type of chain codes is defined by its directions, the feature representation in this work employs the chain codes of 4-directions with the codes and directions are indicated in Fig. 26. Since there were 4 directions, the feature representation is set in 4 consecutive parts. Each direction of the chain codes will be counted from each small area over all 64 grid areas. Therefore, the total length of the feature representation is $4x64 = 256$. The first part will be filled by the number of code 1 in all area $1 - 64$ and put on the first $1 - 64$ segment of the representation. The second part will be filled by the number of code 2 in all area $1 - 64$ and concatenated to the first representation in position $65 - 128$. The same things are applied to the code 3 and 4 to fill the position of $129 - 192$ and $193 - 256$.

The recognition of the basic character is committed by tool LIBSVM [6]. Some experiments with different categories were executed to provide some comparable results. The complete settings, processes, and results are described in sub section 6.4.2.

2. Recognition of the Diacritics
   Lampung script has several diacritics as readers can see in Sec. 3.3. Although there are 12 kinds of diacritics, but basically they only need to be recognized as 7 classes. This occurs since a particular diacritic can be found in two or three different positions. In term of its position around characters, one diacritic glyph can be considered as two or three distinctive diacritics according to its position, while in term of its geometric shape, the diacritic is regarded as one identical diacritic.

The feature to be used in the diacritic recognition should be selected in such a way that the feature should contain the characteristics of this diacritic like; the small size, variability in their dimension, less variation in shape, and usually fewer classes as compared to characters. Concerning these constraints, two binary image sources are considered to provide representative feature during feature extraction. One part of feature was extracted from normalized CC images while another part was extracted from original size CC images.

Feature representations that were merely extracted from normalized CC images often got failed to classify some diacritics during recognition. Thereby, the feature extraction does not only rely on features from normalized CC images but also original size CC images. From normalized CC images, pixel densities was extracted. While From a binary image with original size, some characteristic measurements of the diacritic in their realistic shape can be explored and used for feature like major axis length, the minor axis length, orientation, aspect ratio, and eccentricity.

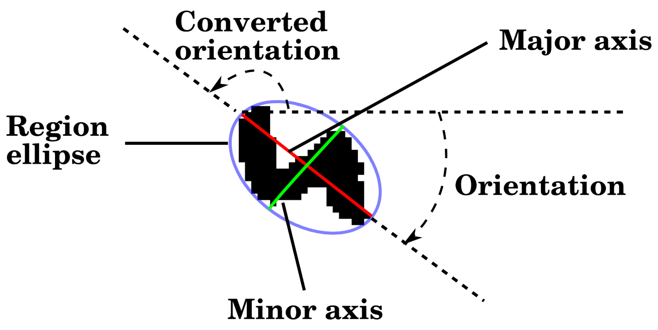


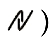
Figure 27: A sample of the diacritic in its original size with the definition of some characteristics. Those characteristics are set to be the feature representation of the diacritic.

Figure 27 illustrates the representation of those characteristics for a diacritic sample. Major axis length is the length of the major axis of the ellipse of the diacritical circumscribing. Whereas, the minor axis length is the length of its minor axis of the same region. Both values are necessary to estimate the magnitude of the diacritical region which is measured along its own axes. The orientation is a scalar that indicates the angle between the x-axis and the major axis of the diacritical region. The range value of this parameter is between 90° to -90°. However, in this representation, the value is converted into range from 0° to 180°. This parameter along with the aspect ratio can detect the diacritic orientation relative to the horizontal axis. This means that both orientation and aspect ratio can discriminate diacritics in form of horizontal-shape and vertical-shape, that are frequently used in the text. The eccentricity feature is defined as the ratio between the distance of two foci and major axis of the circumscribing ellipse of a diacritic. The range value of the eccentricity is between 0 and 1. The value of 0 for eccentricity means that the ellipse is a circle, while the value 1 means that the ellipse represents a line segment. By using this characteristic in the feature representation, diacritics with the shape of proportional dimension or in form of a line segment regardless in horizontal or vertical direction can be appropriately addressed.

To provide a variation of the feature representation, the pixel density feature is also extracted from binary image with a normalized size of 20x20 pixels. Before extraction, the bounding box area of a diacritic is sub sampled into smaller areas with size 4x4 pixels resulting in 25 connected areas. From each area, the pixel density is counted and then normalized with the total pixel in each area.

The classifier for these experiments is SVM. The feature vectors are arranged into the format of one SVM tool i.e. LIBSVM [6]. To generate several results, some SVM executions are accomplished with different kernel types. The detail of this execution can be found in sub section 6.4.3.

5.3.3.2 *Recognition of Two-components Character*

In recognition of the basic characters, the whole target of recognition were only addressed for 18 characters with a single component blob. However, the official Lampung character totally consists of 20 characters where two of them are respectively assembled by two other single component characters in a specific position. These characters are ″ra″ () and ″gha″ ( ). To recognize complete characters, a classification should be performed in two stage. The first stage should deal with the recognition of single component characters while the next stage should handle characters with two separated components. The first stage has been discussed within part 5.3.3.1 and this part concerns about the use of its results to classify two components character.

As characters ″ra″ ( ) and ″gha″ ( ) are created by concatenation of character ″ga″ ( ) and ″pa″ ( ), a general strategy of this step is to search both building block characters from the output of the first classification stage (single blob classification) and examine whether its neighbor characters should be attached or not. A general procedure to handle two components character, as a part of the framework of Lampung handwritten character recognition, is done as the following:

1. Classification of the single component character as preliminary step to get all one component nominee. This step is previously explained in subsection 5.3.3.1.

2. Scanning of the component of character ″Ra″ and ″Gha″. Both components are respectively single blob of class 2 and 4. Therefore, this step is to isolate all class 2 and class 4 among others from the recognition of the first step.

3. For each class 2 and 4 in the second step, identify the closest neighbors for inspection and consider only class 2 and 4 as potential pair entity of character ″Ra″ and ″Gha″.

4. Between each pairing entity, particular features that can reflect character ″Ra″ and ″Gha″ as one unit character are extracted.

5. With generated features, the recognition is performed by classifier and the results are documented to be analyzed further.

This second stage experiments were also performed by SVM through LIBSVM tool [6]. The task in this stage is intended to recognize a connected or unconnected type between two CCs under consideration. The connected indicate that both components represent a character and an unconnected type refers to a single character for both components. To measure the performance, the outcome of classifier should be arranged into table 2 as follows:

Table 2: The extracted values for computing two-components character performance

	Classified pairing	Classified nonpairing
Pairing class	True Positive (TP)	False Negative (FN)
Nonpairing class	False Positive (FP)	True Negative (TN)

This measurement is actually used for binary classification. The use of it in the recognition of two-components character is still acceptable because recognition outcomes can be represented as interrelation of those components. In a practical context, the result can be mapped into "correlated" for a decision of pairing and "incorrelated" for those which are not pairing.

From the values in the table 2, the performance rate can highlighted in term of precision, recall, and accuracy. The formula to compute those metrics are given in the formula 5.10.

$$\text{Precision} = \frac{TP}{TP+FP}; \text{Recall} = \frac{TP}{TP+FN}; \text{Accuracy} = \frac{TP+TN}{TP+FN+FP+TN} \quad (5.10)$$

Results of the experiment and discussion regarding two-components characters are comprehensively described in section 6.4.4.

5.3.3.3 *Association Scenarios*

As Lampung writing system employs many diacritics, an association strategy between the character and diacritics must be addressed before the final recognition. Although a preliminary association between a character and a diacritic had been exposed in subsection 5.3.2, it is not enough to cover the complete model of the Lampung handwriting. That work only assigns one-to-one associations between a character and a diacritic which means a diacritic only associated to a character. While the association may also appear in more than one relation.

To handle this problem, it is firstly important to understand the principle of associating characters and diacritics. The task of associating is always initiated from the diacritic side. This principle stems from the fact that a diacritic, if exists, is always attached to a character but this is not applied in the opposite way. A character does not always have one or more diacritic assigned to it. By using this principle, the possibility of false association can be avoided as much as possible during the process of association.

The association itself essentially consists of the pairing and combining task. The pairing is the task of searching which character has to be selected to make diacritic-character pair. The combining is the task of incorporating all diacritics that belong to a character. The task of pairing had been introduced in sub section 5.3.2. In this task, two schemes of pairing have been proposed to prepare a relation between a diacritic and a character. The first scheme is relying on the closest distance between a diacritic and a character as a parameter to connect them. The second scheme is using Gaussian Mixture Model (GMM) to detect the connection between a diacritic and a character. Results from this work can be served as a foundation of the combining task. They can be fed into the combining task to produce a complete model of characters and diacritics.

By the nature, diacritics from the searching task are already paired to characters. To follow up this result, the task that left is to identify all diacritics to their appropriate character and join them. At the end, a complete model of Lampung handwriting can be obtained based on characters that are completely joined. Regarding this task, three scenarios can be applied to form a complete model,

- Join all diacritics from the closest distance to characters they belong to.
- Identify all diacritics from the experiment of GMM and attach them to their character pair.
- Attach all diacritics from the second scenario with some additional rules taken from Lampung writing system particularly rules regarding the use of diacritics around characters.

Those scenarios can be considered as the final line in the recognition chain since the overall level of the Lampung handwritten character recognition will enter to this phase. All performance of previous works will consequently contribute to the performance of this work. A comprehensive outcome as well as the discussion of this task are reported in sub section 6.5.2.

5.3.4 *Remarks*

The complete framework for Lampung handwritten character recognition is already designed from this work. The target of this framework is to process the input of the Lampung handwritten character images such that the compound characters as the smallest unit can be recognized. The overall process in the framework can be observed in Fig. 28.

This framework is still an underlying foundation so that it may offer a broad opportunity to be explored. There are much open problems in this framework to be solved by new approaches for example, separation of touching components, reconstruction of the break-up components, baseline normalization, etc. There are also some topics that has not been touched like writer identification, line segmentation, skew detection, touching character, etc. A final procedure like the one occurs in the work of composing the complete model also needs a further investigation especially revert back to some former tasks which can affect the current performance. All

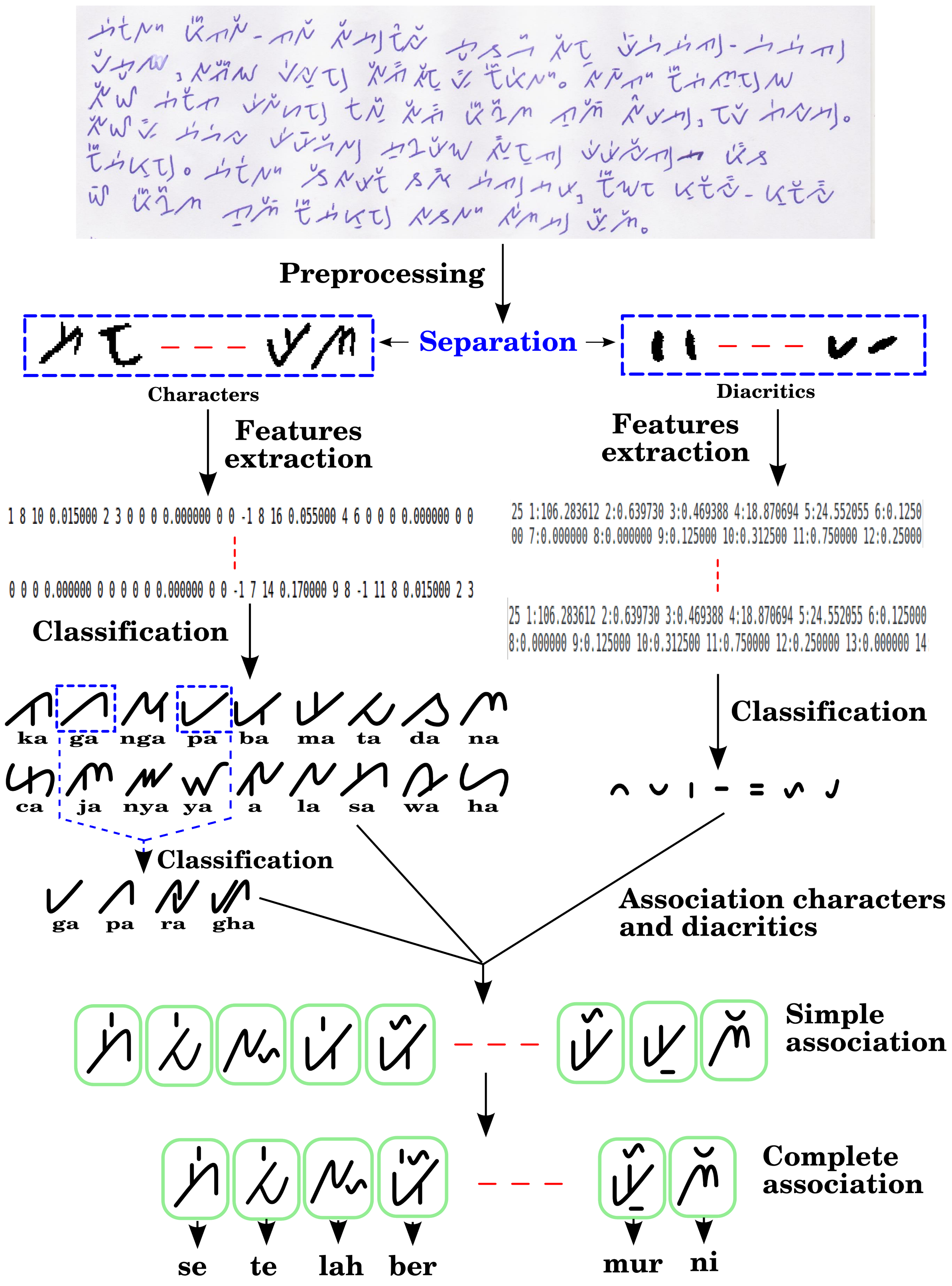


Figure 28: The Lampung handwritten character recognition framework.

those are the prospective challenge for the future research of Lampung handwritten character recognition.

The knowledge from this research can hopefully be useful as a learning source with respect to framework of Lampung handwritten character recognition for future researchers. The framework can be adapted, extended, or further developed for improvement. This may ultimately bring a success for establishing the system for recognizing of Lampung character.

6 EVALUATION

The purpose of this chapter is to provide a comprehensive assessment concerning the proposed approaches in the framework of Lampung handwritten character recognition as presented in Chapter 5. The performance of each approach was characterized by experiments. The discussion may involve a necessary analysis of some quantitative and qualitative results from those experiments.

The chapter is started by brief information of the primary data used in this work. Then, the process of each part of the framework are described. Measurements were documented and analyzed to know merits and drawbacks of respective approaches as well as the solution proposed for the appearing drawbacks. Finally, experiments has also listed some substantial hints for improvement in future works.

6.1 DATASET

The primary data of this research had been prepared prior to research work in form of handwriting scanned images. The raw Lampung handwritten text data had been acquired and collected from local contributors in Bandar Lampung city, Indonesia. All contributors are 10^{th} and 11^{th} grade students of a senior high school in Bandar Lampung.

The sources of the texts were taken from Indonesian fairy tales written in Roman characters. Then contributors had to transcribe these Roman texts into Lampung handwriting on a sheet of A4 paper. To have proportional handwriting data in one page, each page of source texts were controlled no more than 200 Indonesian words. This constraint became a concern because the size of a single handwriting Lampung character is usually bigger than a single printed Latin character. This would consequently enable contributors to have enough space for transcribing of fairy tales texts because a single A4 page approximately can conceive all characters of those words. Fig. 29 shows a snippet of a document image from data collection.

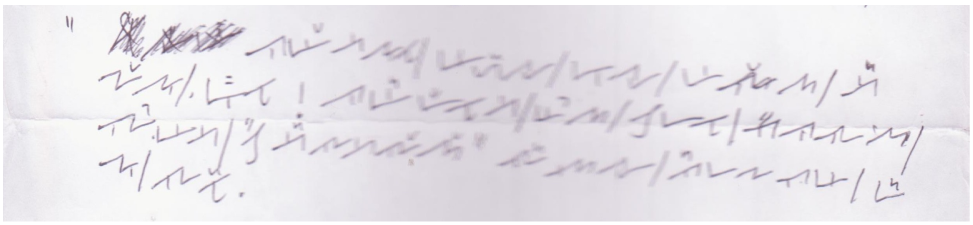


Figure 29: Sample of a Lampung document image, containing degraded illumination, folded track, and noise by overwriting.

Table 3 exhibits brief information regarding raw data. This list can implicitly portray the actual fact of Lampung handwriting documents to be dealt with.

Table 3: Statistical summary of raw data

Attributes	Remark
Male Contributors	20 persons
Female Contributors	62 persons
Number of page samples	82 pages
Number of words	11,722
Collection Period	December 2010

In general the Lampung document images are in good condition. However, the quality of the documents is not entirely uniform. A few documents contain dirty spots, typos, overwritten character, folded marks, and other type of noise. There is a document with different sizes of characters i.e. normal size in several lines of the beginning but smaller for the rest. In other documents, contributors made guiding lines before start writing Lampung handwriting. A little example of this artifacts has been given in Fig. 29 and others can be explored in Chapter 3 as can be seen in Fig. 12.

6.1.1 *Dataset of Initial Labeling*

The existence of labeled data is highly important for running this research experiment as well as the recognition and evaluation. The initial labeling of this dataset is the first endeavor to realize the framework. The process was accomplished by applying semi-supervised labeling as proposed in [57]. Then, the labeling results were inspected to ensure the correctness of the given label from this approach by displaying them for a rapid visual check by a human expert. If there are CCs which do not belong to the current class, then those CCs must be split and re-labeled as it is. The visual check was done until all members of all classes had been completely checked. Hence, this preparation had ensured that the dataset for experiments in this work has a set of correct labels.

Note that experiments on labeling utilized data samples in 11 character classes. The distribution of each character in samples is arbitrarily unbalanced. Some character classes have a big proportion of at least 8000 samples while one of them has only less than 300 samples. According to this distribution, the biggest proportion is distributed in character class *pa** with composition 24.52% (8629 samples) and the lowest number of samples is distributed in character class *wa* which containing only 0.72% (254 samples). The overall distribution of the samples based on 11 character classes is presented in Appendix A.1.

6.1.2 *Dataset of 11 Character Classes*

The total number of labeled data of characters to be utilized in this experiment is 35193. The classifier for this classification is Neural Network (NN). To provide an appropriate data samples for NN, the composition of data were divided into 3 different parts for train, test, and validation set respectively. The proportion of data samples are respectively 60%, 30%, and 10% from total number of data respectively for training, testing, and validation set. When these proportions are converted to the number of samples, each of them are respectively 21122 samples dedicated for the training set, 10547 samples dedicated for the test set and 3524 samples dedicated for the validation set. As the whole data is not equally distributed into all classes, the proportion of each character in this composition is also not equally distributed. Some characters have a large samples while other has only a few samples, in particular character ″wa″ that falls below 1%.

6.1.3 *Dataset of 18 Character Classes*

Beside the composition of 11 classes, the more refined data samples were grouped into 18 character classes. This composition was basically built from composition of 11 character classes by discriminating inner characters of classes with sign ″*″ to be some other classes. Within the class with this sign, there exists 2 or 3 resemble character classes so that the character classes could be extended. However, the class ne* in distribution of 11 character classes is not part of the character as indicated in subsection 6.3.2 because this class collects all kind of noise generated by unwanted conditions, for example touching characters, broken characters, some diacritics with the same size as characters, etc. In fact, it was excluded in the group of 18 character classes so that the total number of samples is 32140. The whole distribution of this data sample extension can be observed in Appendix A.2.

The distribution of data samples in 18 character classes is also not equally distributed. The number of character class of *ca* and *wa* in this composition are less than 300 samples. This is consequently impact of the nature of *Bahasa* in which the transcription was written. In general, *Bahasa* contain less use of the character ″c″ and ″w″ compared to other characters.

The dataset for these experiments were arranged in document-based mode which means that the sample distribution was grouped in a document-wise basis. Among all 82 documents in the dataset, 52 documents were collected for training samples, 10 documents were used for validation sample, and the remaining 20 documents were assigned for testing samples. The training set of 52 documents consists of 20141 samples. The validation set of 10 documents consists of 4146 samples. Finally, the testing set of 20 documents consists of 7853 samples. The percentage of samples in the training set contains 62.67%, validation set contains 12.90% and testing set contains 24.43% of the whole samples in dataset.

6.1.4 *Dataset of 7 Diacritic Classes*

The total number of diacritics in dataset is 24775 samples. They consist of 7 classes only with the class 6 (=) at the lowest rank. The sample distribution of diacritics is shown in Appendix A.3.

Note that the class 6 probably will not appear in Appendix A.3 if contributors perfectly wrote them as two components diacritics. Instead, it will be grouped as two separated components in class 5. Nevertheless, the class 6 still appears in Appendix A.3 which indicates that both components of class 6 were somehow connected by a tiny pixel so that during clustering, they are regarded as one class.

The dataset of the diacritic, as the experiment of character recognition in 18 classes, was arranged in a document-wise basis. The training set consists of diacritics from 52 documents, the validation set consists of 10 and testing set consists of 20 documents. From the total 24775 diacritic samples in dataset, there are 15516 samples derived from training set, 3201 samples derived from validation set and 6058 samples derived from testing set.

The distribution of diacritics for each class is not equally balanced. The class 5 occupied 34.73% of the testing data or 2104 samples as the biggest class. Whereas, the class 6 is at the last position with 108 samples. The distribution number of each classes in the testing set can be observed in Appendix A.3

6.2 PREPROCESSING

As indicated in Chapter 5, the ESMERALDA tool was involved in the preprocessing stage. Specifically, binarization and Connected Component (CC) extraction have been performed as described in Section 5.1. A complete evaluation of these activities is described in the following.

6.2.1 *Binarization*

The modeling of the document foreground over its background is difficult due to various types of document degradation such as uneven illumination, image contrast variation, bleeding-through, and smear. Some raw image data were chosen to be used in initial binarization for the purpose of a qualitative evaluation. The images were selected in such a way that all types of quality are included i.e. from the worst upto the best image. This is important to enable performance comparison among various algorithms of binarization.

During practical works, the Otsu [38], Niblack [37], and modified Niblack algorithm of the ESMERALDA tool [12] were respectively executed to all chosen images. The results of different algorithms for each image were compared to each other and also to the original image data. These observations were only visual checks of the result produced by two algorithms to ensure which algorithm would be used for the rest of the raw image data which could generate the best quality.

From experiments, binarization of the best quality images returned almost the same quality binary images from all algorithms. No significant differences appeared between output images. However, for the worst images, all algorithms generated a different level of quality. Results of the Otsu algorithm degraded in many parts of the image. The Niblack algorithm also generated binary image with many noise spots around a large empty area with a certain level of gray intensity according to evaluation in [23]. The modified Niblack algorithm gave a best result for binarization compared to other two algorithms. The visual output from those algorithm can be observed in Fig. 30.

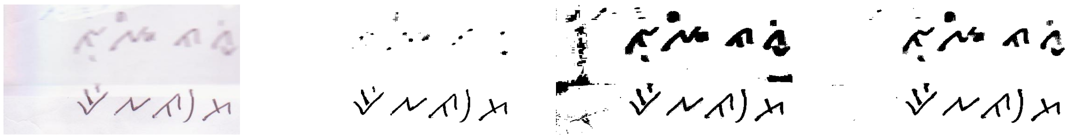

(a) Gray image sample
(b) Binary image generated by Otsu algorithm
(c) Binary image generated by Niblack algorithm
(d) Binary image generated by modified Niblack algoritm from ESMERALDA tool

Figure 30: Binary images produced by performing Otsu, Niblack, and modified Niblack algorithm.

As there is no perfect binarization for all kind of image quality, the best result will still depend on the original document. For the dataset of the Lampung handwritten character recognition, binarization results produced by modified Niblack algorithm generated the most precise binary images reflecting the original images.

6.2.2 *Separation of Connected Components (CCs)*

After collecting all CCs, measurements were carried out to them to obtain some basic characteristics of CCs to be used for separation thresholds. These thresholds encompass from the size, pixel density of bounding boxes, and aspect ratio of the width and height. They are specifically computed for each document image. Since those characteristics were solely computed for each document image, threshold values differ from one document to another document. These thresholds were ultimately used to perform a fully automatic CCs separation for each document image so that all unwanted objects can be minimized.

As the target of separation is to obtain character primitives over diacritic primitives and vice versa, while dropping other components like noise, the separation consists of two different tasks. The first one is for retrieving characters and another one for diacritics.

For a general treatment of both type of CCs, a fixed global threshold had been applied for an initial selection of CCs. Threshold values are selected based on the size of CC relatively to the size of respected document. In this regard, the accepted CC for preliminary measurement must have the dimension between 5x5 pixels to 75% of the width and height of the document size. Other CCs that did not meet this requirement were not considered for character or diacritic candidates.

6.2.2.1 *Character Separation*

Preparation and execution of the character separation respectively comprise of two steps. In the first step, an elaboration of threshold values should be done through a preliminary process. The threshold was computed for each document with respect to minimum, maximum, and average of the width and height of all CCs. Since there are three thresholds and each of them spans in a specific range such that a single threshold value is within a range, a minimum and maximum bound, all those three thresholds will have six setting values for lower and upper bound. Then in the second step, those thresholds were used to split CCs of characters among others. Sometimes, a necessary adaptation to these thresholds is needed prior to separation process to get more primitives rather than relies only on those thresholds without any modification.

The first threshold value to be explored is the size of the bounding box which is defined by the area of the components. The value of large area for this separation was represented as an interval with a lower and upper bound value. To get a lower and upper bound, a qualitative evaluation estimated the weight threshold for both bounded limits respectively. The separation is specified by the following formula,

$$large\_area = w \cdot ave\_width \cdot ave\_height \tag{6.1}$$

where:
w represents the weight for large area, $w_{min} = 0.5$ and $w_{max} = 9.0$
ave_width represents the average of width of the bounding box
ave_height represents the average of height of the bounding box

Based on the qualitative evaluation, the weight threshold w of the lower bound for large area is set to 0.5 while for the upper bound, the threshold is set to 9.0. The weight factor for the upper bound seems high, but this factor is realistic since it compensates the small size of diacritic primitives which contributes in reducing the whole average of the width and height. With a high weight, the average is presumably approximated to the real character primitives average.

Among extracted CCs based on separation by applying large area, there are also possibilities that some of them can be considered as noise. For example, CCs with either less or too much black pixels will likely be a noise rather than character instances. To explore this characteristic, the pixel density of the bounding box is employed. The pixel density is counted for the bounding box which defined as,

$$pixel\_dens = \frac{foreground\ pixel}{total\ pixel} \tag{6.2}$$

$foreground\ pixel$ represents the number of foreground pixels
$total\ pixel$ represents the total pixel in bounding box

Based on the computation of the density during a qualitative evaluation, a character might have pixel density around $20 - 30\%$ of the large area. While the threshold of large area cannot handle this situation since it only consider the size. Therefore the pixel density was included as a separation threshold as well. To define a representative pixel density threshold, the minimum, maximum, and average of pixel density were counted and stored.

The lower bound of the pixel density can be computed as a composition of the average and minimum pixel density with a certain weight for each of them. The determination of the weight of each component was done via trial runs in informal experiments. The formula for lower bound of the pixel density is represented as,

$$\begin{aligned} pix\_dens_{min} &= ave\_dens - \frac{1}{20}(ave\_dens - min\_dens) \\ &= \frac{19}{20}ave\_dens + \frac{1}{20}min\_dens \end{aligned} \tag{6.3}$$

where:
ave_dens represents the average pixel density of all CCs
min_dens represents a minimum pixel density from the set of CCs

Moreover, the weight for the upper bound of pixel density was also derived from such experiments. The formula is composed by the same manner as formula 6.3 with involvement of the average of pixel density. While for another component, this upper bound needs the value of the maximum pixel density instead of the minimum pixel density. The formula of this upper bound is given by,

$$\begin{aligned} pix\_dens_{max} &= ave\_dens + \frac{1}{4}(max\_dens - ave\_dens) \\ &= \frac{3}{4}ave\_dens + \frac{1}{4}max\_dens \end{aligned} \tag{6.4}$$

where:
ave_dens represents the average pixel density of all CCs
max_dens represents the maximum pixel density from the set of CCs

The minimum value in formula 6.3 and maximum value in formula 6.4 represent the value of a single CC over the whole extracted CCs. Both values are used as marks for computing a valid range of the pixel density. An actual value of lower and upper bound density must fall within this range. It is impossible for the density of primitives to be less than the minimum density or more than the maximum density.

The factor $\frac{1}{20}$ in lower bound as shown in formula 6.3 and $\frac{1}{4}$ in upper bound as shown in formula 6.4 are purely obtained after some trial runs of informal experiments. These constants are the best approximation for all image documents in this work. They will probably not be ideal to other documents since other documents can contain different composition of diacritic and character primitives.

The third threshold for separation is aspect ratio. This threshold is defined as the division of the bounding box width and height as given in the following,

$$aspect\_ratio = \frac{width}{height} \tag{6.5}$$

where:
$width$ represents the width of the bounding box
$height$ represents the height of the bounding box

In contrast to the two previous thresholds, aspect ratio was not directly computed from document processing because the aspect ratio of characters mainly approximate to one as indicated in subsection 5.1.4. Consequently, the range between the lower and upper bound of the aspect ratio should have the value one in between. Preliminary option for this interval were set to 0.5 and 2.0, respectively. However, some handwritten documents contained character bounding box with ratio up to 4.0. Finally, the interval for aspect ratio became the following [20],

$$0.5 \leqslant aspect\_ratio \leqslant 4.0 \tag{6.6}$$

These interval could distinguish the noise, which resembled a long vertical line if its ratio was less than 0.5 or a long horizontal line if its ratio was greater than 4.0. This threshold was effective to remove some artifacts coming from folding, guiding line, massive touching components, etc.

As a final remark, it can be said that separation is not always working perfectly. As separation is fully executed in automatic mode, it still cannot split all kind of noise and may possible remove character instances. As long as missing instances is only a little, the output of this separation task is still tolerable.

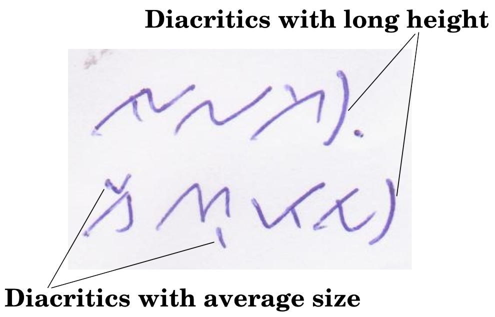
Table 4: Connected Components of Character

Attributes	Quantity
CCs of character primitives	35,193
Character primitives per page	429

Table 4 shows the statistic summary of the CC's character after separation process with aforementioned thresholds.

6.2.2.2 *Diacritic separation*

The procedure of diacritic separation was similar to procedure of character separation with a little adjustment to cope with diacritic matters. In this procedure, the preliminary measurement considered the range from 5x5 pixels to 25% of the width and height as CCs to be used for calculation of the threshold. Beyond that range, instances were not accounted for calculation of the threshold.



(a)

Figure 31: Comparison of the average diacritic and diacritic with the same height as the height of the character.

In general, the size of diacritics is smaller than characters. However, they can essentially be distinguished into the following characteristics,

- Ordinary diacritics which are the most common diacritics. They have a proportional size of width and height such that their aspect ratio are close to one.
- The diacritic with specific dimension where the height is longer than its width. The height is usually the same as the height of the character but the width is like the width of the diacritic.

Fig. 31 shows samples of both types and confirms this indication. In this example, two sample diacritics on the right side of characters have the same height as the height of those characters but their width are equal to the common size of ordinary diacritic. In other samples, two other diacritics indicate the sample of ordinary diacritics. To explore more evidences, Fig. 12 in Chapter 3 provide three fragments of three document images. Each of them displays many other samples of both diacritic types coming from three different styles of writing.

To handle separation of each diacritic, parameters remain the same as ones applied in character separation. They consist of the large area, pixel density, and aspect ratio which need to be computed based on local characteristic of document images.

The weight formulation for diacritic separation is computed with the same manner as the formula in character separation defined in Eq. 6.1, 6.2, and 6.5.

To obtain diacritics, settings are applied to all CCs of diacritics. Each setting is usually built by at least two thresholds with a particular range. These thresholds are described in the following.

1. The ordinary diacritic type can be grouped based on the combination of the large area and aspect ratio. The setting of the lower and upper bound w in the formula of large area as denoted in 6.1 are,

   $$w_{min} = 0.1 \qquad w_{max} = 0.7 \tag{6.7}$$

   Those constants are still combined with the aspect ratio by the following interval,

   $$0.2 \leqslant aspect\_ratio \leqslant 4.0 \tag{6.8}$$

2. The pixel concentration of some ordinary diacritics is higher than characters. For specific diacritics, it can contain almost $80 - 90\%$ of their bounding box. With only above threshold settings, those diacritics will be dropped as being not a diacritic. Therefore, the second option to handle this condition is to set other thresholds by involving the pixel density. After analyzing the pixel density of diacritic with massive concentration, the minimum pixel density level was set to 0.5. Along with the previous aspect ratio, the threshold is given as follow

   $$aspect\_ratio \leqslant 4.0 \qquad pixel\_dens \geqslant 0.5 \tag{6.9}$$

3. The second type of diacritic is only diacritic *nengen* ("**J**"). Due to handwriting style of contributors in transcription, this diacritic may be seen in various appearances. The aspect ratio of this diacritic is roughly between 0.2 and 0.6. This threshold has been covered in formula 6.9. However, the pixel density threshold in this formula only applied for a few of these diacritics. Since, the this diacritic does not contain massive pixels, another threshold must be set. By reviewing many samples of this diacritic type, the pixel density is set between 0.1 and 0.3. Therefore, another threshold of this diacritic is

$$0.2 \leqslant \mathtt{aspect\_ratio} \leqslant 0.6 \qquad 0.1 \leqslant \mathtt{pixel\_dens} \leqslant 0.3 \tag{6.10}$$

The separation of diacritics based on thresholds setting also has a weakness like the separation of characters. There is a possibility that a small number of diacritics will be discarded during the separation process because of a high variability in writing style of the contributors. However, this is assumed really small compared to the total samples of diacritics. The summary output generated by all settings is outlined in Table 5.

Table 5: Connected Components of Diacritic

Attributes	Quantity
CCs of diacritic primitives	23,534
Diacritic primitives per page	287

6.2.3 *Normalization*

A normalization was conducted to transform all CCs into a certain level of uniformity so that features can be extracted easily. The CCs of Lampung characters and diacritics were linearly normalized into a fixed square dimension.

The character CCs were respectively normalized into two different square size, the small size with dimension 20x20 pixels and the moderate size with dimension 32x32 pixels. The first size of normalization is similar to normalization size of MNIST dataset [26]. However, there is a little difference. The MNIST 20x20 images is a result of normalization by preserving the aspect ratio, while in this work, the image 20x20 is a result normalized without preserving aspect ratio.

The MNIST bigger size is set to 28x28 with a reposition of the normalized image such that the center of mass of the pixels positioned on the center of the bounding box. Whereas in this work, the normalization size is addressed for 32x32 without such a reposition.

Concerning the small size for normalized images, it is encouraged by two reasons. First, the small size of normalized image will relatively require a little cost and time during image operations. The small size means a small amount of pixel leading to a small processing time. Second, the small size normalization was also desired because it is closer to the actual size of the real character bounding box. The moderate size normalization was merely chosen to provide a larger detail of character blobs. Another component need to be normalized is the diacritics. In this case, the diacritic CCs were only normalized to 20x20 pixels through linear normalization. This dimension is sufficient as the real size of diacritic is small and less in variation. As described in subsection 5.1.4, the process of the diacritic normalization was begun by adding one-pixel perimeter surrounding to the raw CCs bounding box as an effort to keep the original characteristic during the normalization process. Consequently, normalized diacritic images were surrounded by one-pixel frame of background.

6.3 annotation

The labeling process of Lampung handwritten character dataset had been done through the work of semi-supervised labeling [57] as reported in sub section 5.2. The labeling strategy in this work initially covered 20 classes of Lampung characters. From a description in part 5.3.3.1, there are essentially 18 basic characters, while the remaining two classes in this composition define a ”miscellaneous classes”. Both respectively represent a diacritic (class 19) and a collection of noise (class 20). However, due to a tiny difference between some characters, the almost-similar shape characters were merged in a class so that the total becomes 11 classes at the second round of labeling. The concern of the labeling strategy in this work is not to achieve a high score, but to emphasize potential benefits from this semi-automatic labeling approach with less work and cost. This section discusses about some results and the evaluation of this labeling procedure.

6.3.1 *Initial Experiment*

To assure such a labeling approach can work appropriately and show the potency, the training and testing had been executed on the Lampung dataset. These data consist of 35193 CC images of characters extracted from 82 Lampung handwritten documents. About 20 characters from each document were labeled manually for the purpose of testing resulting 1640 characters in total. The rest of 33553 CCs were set for the purpose of training without any label attached to them. To easily check this information, Table 6 provides a summary regarding this statistical data.

Table 6: Summary of Dataset for Labeling Works

Attributes	Quantity
Number of document	82
Manual labeling	20 characters/document
Number of CCs images	35193
Total labeled samples	1640
Unlabeled samples	33553

The implicit labels for the training set were inferred during the training process by aforementioned strategy as illustrated in section 5.2 which is considered as a semi-automatic labeling approach. After the training process, output of the final voting were 45.44% sample agreed upon two classifiers and 45.99% sample agreed upon all three classifiers. The rest of 8.57% samples were undecided due to all classifiers voting differently. Supplying only the sample acquiring from the unanimity vote (Eq. 5.1) over the testing, the accuracy of K-nearest neighbor classifier was 60%. Performance of the approach was rather low rate because of the fact of the sample agreed for unanimity vote on the training set less than half (45.99%) of its total. In addition, the K-nearest neighbor is susceptible to small dissimilarities so that a small disparity on the character shapes would not be enough to discriminate the characters correctly.

6.3.2 *Result Analysis and Further Experiment*

Sensitivity of K-nearest neighbor on the distortion was visible with a closer look at the test set after testing process. In this inspection, based on the confusion matrix many character shapes of the CCs of the test set were really similar. The small distinction among the shapes was just a short stroke, stripe or strip that frequently occurred in Lampung handwritten texts. Realizing this fact, it is reasonable to merge any resembling characters into a single class and then re-labeled them. The following list is the group of characters (which contain the set of nearly-similar characters) and their new classes,

- The characters ka( ), ga( ) and sa( ) were merged into class ka*.
- The characters nga( ), a( ) and la( ) were merged into class nga*.
- The characters pa( ), ba( ) and ma( ) were merged into class pa*.
- The characters na( ) and ja( ) were merged into class na*.
- The characters ca( ) and ha( ) were merged into class ca*.
- The diacritic nengen( ) and noises were merged into class ne*.

Note that in this list, classes sometimes consist of two actual characters while sometimes can be three actual characters. With this re-arrangement of classes, the overall class reduced from 20 to 11 classes.

This relaxing technique by merging some similar characters indeed implied the better performance than the one that reported from the initial experiment. After further training with the new merged classes, the number of votes on the training set with all classifiers agreeing (unanimity vote on formula 5.1) rose to 75.04% while the vote for 2 classifiers (simple majority on formula 5.2) fell to 22.27%. The rest 2.33% were undecidable as all classifiers respectively chose a different label. The detailed classification results on the test set with confusion can be observed in Table 7.

The improvement of the rate in the latest training also influenced positively to the performance of the recognition rate by k-nearest neighbor over the test set. Relying only the training sample with all classes agreed (unanimity vote with formula 5.1), the recognition performance of the k-nearest neighbor classifier ($k = 1$) for 11 character classes improved to 86.21%. The rate is very promising compared to the

	ka*	nga*	pa*	ta	da	na*	ca*	nya	ya	wa	ne.*
ka*	360	0	0	2	1	0	0	0	0	0	4
nga*	3	256	1	8	0	4	0	0	0	0	0
pa*	1	0	373	1	0	0	0	0	0	0	3
ta	9	14	0	133	2	0	0	0	0	0	3
da	8	1	1	19	66	1	0	0	0	0	8
na*	6	43	0	0	0	46	0	0	0	0	0
ca*	2	0	6	0	0	0	46	0	0	0	0
nya	0	13	3	2	0	2	0	0	6	0	0
ya	1	1	3	0	0	0	1	0	33	0	0
wa	1	5	2	5	0	0	0	0	0	0	0
ne*	10	6	6	5	1	0	1	0	1	0	93

Table 7: Confusion matrix for Lampung using a K-nearest neighbor (K = 1)

effort of manual labeling by an expert which were only 162 (3 representations * 54 clusters) instead of thousands of all data samples. The result table with a confusion matrix can be seen in Table 7 [57].

In this table, the biggest portion of error is 43 on class na*. The confusion occurred between class na* (na ( ) and ja( )) and class nga* (nga( ), a( ) and la( )). With a detail observation on both classes, the basic shape of both resemble with each other. The main stroke of both classes share the same drift so that by sensitivity of K-nearest neighbor, both character classes were recognized as being the same class.

The rationale to use unanimity vote instead of simple majority vote for this labeling is indeed to ascertain a feasible guarantee toward the result. This labeling strategy has been proven to be comparable to another dataset as reported in [57].

6.4 recognition of basic elements

After labeling work had been reported, the next task to be evaluated is the recognition stage as had been explained in section 5.3. This discussions reviewed the result of main recognition tasks with respect to characters, diacritics, and association of both. Each topic was comprehensively elaborated to some extent to assess proposed approaches or methods.

The subsection is started by a summary of the work, reviewed based on the number of character classes. Then it is followed by the discussion of two recognition works of Lampung handwritten characters with different focus on the total character classes to be recognized. In the first part, recognition was targeted for 11 character classes while another part targeted for 18 character classes. In addition, the recognition of diacritics is also covered in this section.

6.4.1 *Recognition of 11 Character Classes*

The first recognition task of the Lampung handwritten character was focused on the set of character group which consists of 11 different classes. These groups are shrunk from the complete character set since some of them resemble with each other (see subsection 6.3.2).

This work is the first effort to build a preliminary rate for the Lampung handwritten character recognition. Since this is an early attempt, a small number of classes is taken to simplify the whole process of recognition and be focused on the main task with less problems. The result of this recognition provides a baseline performance for Lampung handwritten character recognition. With this baseline performance, it is expected that new methods or approaches can improve this performance.

6.4.1.1 *Experiment*

The classification task for 11 character classes in this part uses two group of features as explained in sub subsection 5.3.1.1. These are branch points [8], end points [8], pixel densities in the first group and the Water Reservoir (WR) [7], [40], [41], [42], [43], [44] in the second place.

The Neural Network (NN) [4], [10] [8], [54] was chosen to perform classification because it uses less threshold, less storage, and less computation so that it is easy to be maintained.

Table 8: Confusion results for branch points, end points and pixel densities

	ka*	nga*	pa*	ta	da	na*	ca*	nya	ya	wa	ne.*
ka*	2334	11	2	5	28	17	9	3	0	0	13
nga*	3	1500	3	20	2	17	0	35	2	2	20
pa*	1	4	2555	4	1	0	5	1	4	1	12
ta	1	33	4	857	6	2	0	2	1	3	17
da	12	1	1	4	611	0	0	0	1	3	13
na*	14	20	0	2	3	480	0	2	1	0	4
ca*	4	0	4	0	0	2	402	0	1	0	4
nya	1	31	0	2	2	5	0	170	9	0	11
ya	0	0	11	0	0	0	2	2	178	0	5
wa	4	19	1	5	3	1	0	1	0	36	5
ne.*	31	47	33	36	17	12	12	15	0	4	707

The architecture of NN used in this work is multi-layer perceptron which consist of three layers i.e. the input layer, hidden layer, and output layer. The input layer always refers to the the size of feature vectors, while the output layer refers to the number of target classes. Setting of hidden layers is more flexible compared to both layers. Logically, the size of hidden layer can be assigned to a value in the range of input layer and output layer. This consequently allows the NN to manage a various combination values from input layer to output layer. However, assigning to another values outside of the interval input-output is still possible but it may impact to the performance of the NN. If the size of hidden layer is below the lowest size, the NN may lose some input combinations for supplying the output layer. In contrast, if the size of hidden layer exceeds the biggest size, the NN may trigger an over fitting before processing by the output layer.

Experiments were done at least three times by executing the NN in several network setups to obtain robust results. The configuration setup comprises of adapting the size of hidden layer and changing of learning rate. The setting of the value of the hidden layer depends on the size of the input and output layer, while the learning rate is set in the range $0.05 - 0.3$.

Table 9: Confusion results using water reservoir based descriptors

	ka*	nga*	pa*	ta	da	na*	ca*	nya	ya	wa	ne.*
ka*	2338	2	0	1	43	13	9	1	0	1	14
nga*	13	1461	6	42	0	31	2	14	0	4	31
pa*	0	2	2546	10	0	0	12	0	2	0	16
ta	1	51	14	810	10	1	2	1	1	0	35
da	61	3	1	17	536	2	2	0	0	4	20
na*	21	21	1	1	0	467	0	10	0	1	4
ca*	10	0	7	0	0	0	397	0	3	0	0
nya	0	20	1	3	0	6	0	185	7	0	9
ya	1	1	2	6	0	0	4	3	177	0	4
wa	5	17	0	1	9	0	0	0	0	34	9
ne.*	32	48	29	66	13	9	13	8	2	13	681

The first configuration deals with the feature representation of branch points, end points, and pixel densities that are extracted from each CC with small grid areas. These feature representations consist of a total of 75 values. Hence, the size of the input layer is 75. A NN pattern for a complete layer configuration can be in the form of $75 - h - 11$ where the variable of h indicates the size of hidden layer. During the training of the NN, the variable h is set to some arbitrary numbers to experience various results. Testing results indicate that the best rate can be achieved at the level 93.20% with the best parameter $h = 75$. The performance is reasonably accurate and acceptable for easy-extracted features like branch points, end points, and pixel densities. Therefore, the feature of branch points, end points, and pixel densities is very potential for Lampung handwritten character recognition. The detailed of confusion matrix is presented in Table 8.

The second round of this 11 classes recognition relies on the Water Reservoir (WR) feature. This feature vector consists of 30 values as indicated at the structure in Fig. 24. Hence, the size of input layer of the NN is 30. The structure of the NN layering for this training is $30 - h - 11$ with the variable h also representing the size of hidden layers. The testing result by using this feature yields the rate 91.32% with the best parameter $h = 30$. This rate is approaching the previous result with the feature

Table 10: Confusion results for branch points, end points, pixel density and water reservoirs [20]

	ka*	nga*	pa*	ta	da	na*	ca*	nya	ya	wa	ne.*
ka*	2358	11	1	0	18	17	6	1	2	0	8
nga*	5	1528	5	13	2	16	1	9	0	4	21
pa*	0	3	2559	3	0	0	4	3	3	2	11
ta	1	26	4	854	6	0	0	1	1	2	31
da	20	4	0	7	598	2	0	1	1	1	12
na*	18	15	0	0	0	484	0	4	0	1	4
ca*	6	0	4	1	0	1	397	0	3	1	4
nya	0	12	2	0	0	6	0	198	6	0	7
ya	2	0	6	2	1	0	1	2	182	0	2
wa	5	8	0	1	5	0	1	1	0	47	7
ne.*	25	39	19	37	18	12	5	12	3	6	738

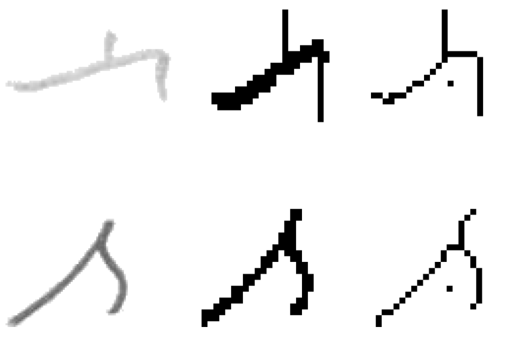
branch points, end points, and pixel densities. To analyze the result, Table 9 shows the confusion matrix with WR features.

For an extended experiment, both features are merged as one integrated feature for the recognition task. With the size 75 from the first feature set and 30 from the second one, the total dimension of this new feature is 105, which also indicates the size of the input layer for the NN. Thereby, the complete structure of this NN layers is $105 - h - 11$. This experiment is developed based on the two former experiments. As for the two former experiments, this experiment round also executed several configurations to see the performance of this combined features. These configurations are generated by modifying the NN learning rate as ruled in the beginning of this subsection and the size of hidden layer with value in between the value input-output layer. The recognition rate of the NN with this concatenated features is 94.27% with the best parameter $h = 105$. This achievement answers the assumption that the merging of these features can improve their original performances. The confusion matrix of the recognition with this merging features can be seen in Table 10.

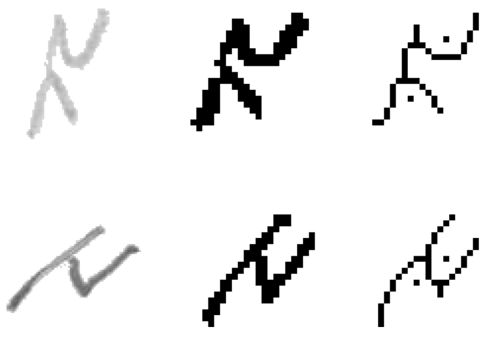
All setting of those three experiments remain the same. The difference only occurs at the size of input layer which is inferred from the dimension of the respected feature vector.

6.4.1.2 *Discussion of the Result*

During the recognition of 11 character classes of Lampung handwriting, three feature components i.e. branch points, end points, and pixel densities are used in the first experiment. A combination of those three features achieved the rate 93.20% accuracy [20] which indicates a proper feature representation to the nature of the character. This appropriateness can be testified from the fact that Lampung characters are non cursive characters so each character will have at least one end point. Moreover, branch points that can be identified from skeletonized character image are also frequently found in Lampung characters due to small strokes, blind bows, intersection, etc. Both strong points are still enhanced by pixel densities that represent local measurement so that all these representations together generate a very promising recognition rate.



(a) The confusion between character *ka** and *da*. Character *ka** is recognized as character *da* (top) and vice versa (bottom)



(b) The confusion between character *nga** and *ta*. Character *nga** is recognized as character *ta* (top) and vice versa (bottom)

Figure 32: Samples of confused characters during handwritten character recognition by using the feature of Water Reservoir (WR). Each sample consist of three images, on the left is gray scale in original size, on the center is binarized image in normalized size, and on the right is skeletonized image in normalized size.

From the confusion matrix of this recognition on table 8, it seems that the most significant problem occurs for recognition of the character of the class *nga** to the class *nya* and vice versa. The number of the character of the class *nga** which are recognized as the character of the class *nya* is 35 and there are 31 confusions for the other way around. Analyzing from the shape of both classes indicated that their skeleton image contain the same amount of end points on the top and bottom side. Those end points for both classes also appear in the same zone and this strongly indicates the confusion between both classes. The same situation also appears for the class *nga** and the class *ta*. In this regard, both characters also have the same situation as class *nga** and class *nya*.

Meanwhile, Table 9 indicates the confusion of the recognition by using the feature of the WR. The distribution of non-zero values in this table and the previous one are relatively similar with each other. However, the source of the major problem comes from recognition of two pair of classes, between the class *ka** and the class *da* and between the class *nga** and the class *ta*. The number of the character of the class *ka** that are recognized as the character of class *da* is 43 and 61 characters of the class *da* are recognized as the character of the class *ka**. Moreover, the number of the character *nga** recognized as *ta* is 42 and there are 51 character *ta* recognized as *nga**.

To see the confusion of two pair of classes, Fig. 32 shows four samples of the major problem during the process of recognition by using the feature of WR. On the left images, the confusion occurs between the character *ka** and character *da*, and on the right side the confusion occurs between character *nga** and character *ta*.

In figure 32a, both characters in their original samples on the left part are clearly distinguishable. However, the process of binarization, normalization, and skele-

Table 11: Recognition improvement for the recognition by using the feature representation of [1]Branch point, end points, pixel density. [2]Water reservoir. [3]Concatenation of 1 & 2

Character	Correct recognition by		
Class	BED[1]	WR[2]	BED-WR[3]
ka*	2334	2338	2358
nga*	1500	1461	1528
pa*	2555	2546	2559
ta	857	810	854
da	611	536	598
na*	480	467	484
ca*	402	397	397
nya	170	185	198
ya	178	177	182
wa	36	34	47
ne.*	707	681	738

tonization had transformed each of them into the shape that are similar with each other. Therefore, the feature of WR of both characters is the same for their number of reservoirs and type of reservoir. While their gravity center (see the dot in the center of WR in Sub fig. 32a), volume, and width-height of the WR are nearly the same. And finally, the complete feature representations of the character *ka** and *da* are identical. This cause a confusion during recognition.

Table 12: The sample of incorrect characters and their reduction for the feature representation of [1]branch point, end points, pixel density. [2]Water reservoir. [3]Concatenation of 1 & 2

Character	Recognized as	Incorrect recognition by		
		BED[1]	WR[2]	BED-WR[3]
nga*	nya	35	14	9
nya	nga*	31	20	12
nga*	ta	20	42	13
ta	nga*	33	51	26
ka*	da	28	43	18
da	ka*	12	61	20

The second sample of confusion is given in Sub fig. 32b. In the final image on the right side, the shape of both characters are totally different. Character *nga** (the image on the top in Sub fig. 32b) has two reservoirs with type the top and the bottom. This also occurs to character *ta* which also has the top and the bottom reservoirs. With almost the same size and position in their water reservoirs, the probability of a misinterpretation is increasing. Hence, this occurrence becomes the source of a confusion between the characters *nga** and *ta*.

With a low size of input vector from the WR feature compared to the size of branch points, end points, and pixel densities, the result of the NN with WR as the feature is still competitive to the performance of the NN with features of branch points, end points, and pixel densities. Although the performance of recognition by using the WR feature is lower compared to performance of recognition by using the features of branch points, end points, and pixel densities, the WR characteristic provides a significant effectiveness during recognition of Lampung handwritten characters. It is able to discriminate most of the characters in Lampung dataset although the size of feature is only 30. This feature can be categorized as a distinctive feature for Document Analysis and Recognition (DAR) of Lampung handwritten characters.

The confusion of each feature representation has been discussed previously. Each feature representation brings some drawbacks as the impact of constructing the feature from the original characters. The concatenation of both groups of features is also used for the character recognition. The output of this recognition is given in Table 10. From this table, the concatenation feature representation indicates some improvements of the drawbacks that previously appeared in the output of a single group of feature. It can be noted that the majority of the characters are improved. A comparative result of the recognition between three feature representations can be observed in Table 11. This table can be read as the number of correct characters recognized from the perspective of each feature representation.

Table 13: Summary of the NN experiment for Lampung handwritten character recognition for 11 character classes. The feature representation is [1]Branch point, end points, pixel density. [2]Water reservoir. [3]Concatenation of 1 & 2

Features	Dimension	Layer size		Learning Rate	Performance of Test(%)
		Hidden	Output		
BED[1]	75	75	11	0.1	93.20
WR[2]	30	30	11	0.1	91.32
BED-WR[3]	105	105	11	0.2	94.27

Beside measuring the correct recognition from each feature representation, an alternative evaluation can also be investigated from the incorrect recognition perspective, particularly the classes with significant mis-classification as previously discussed. The concatenation of the feature of branch points, end points, pixel density, and WR contributes to a better recognition output. As expressed in Table 12, some samples of misclassification characters can be reduced after a recognition by using a concatenation of the feature of branch point, end points, pixel density and WR.

This reduction stems from the fact that both feature representations concurrently support each other during recognition. One feature representation capable to depreciate some drawbacks of another feature representation and vice versa. For example, by only using the feature of WR as presented in Figure 32b, the recognizer had misclassified those characters. By adding the feature of branch point, end points, and pixel density, the position of branch points and end points along with pixel densities can precisely be discriminated for both character. This can drive the classifier to decide a correct character during recognition process.

To summarize of Lampung handwritten character recognition for 11 character classes, Table 13 compares the performance of the recognition based on each feature representation. A remark from this table is that the combination feature between branch point, end points, pixel density with WR is able to upgrade the final performance although the improvement is not large. Therefore, those feature representations can be a suitable choice for recognition of Lampung handwritten character recognition.

6.4.2 *Recognition of 18 Character Classes*

The previous recognition task that focuses on 11 character classes is a preliminary recognition task of the Lampung handwriting in a basic level. The outcome is usable but there is a lack of flexibility and may not be detailed enough for further application. Therefore, a more refined character recognition is necessary to provide an appropriate entity for Lampung handwritten character recognition.

The following subsection address the most complete recognition of the basic character of Lampung. The recognition is done for 18 character classes. Besides providing a first baseline for 18 character classes, this recognition also prepares a necessary foundation for the next stage which is a task with a more complex structure including the diacritics. With respect to this issue, the accuracy of the recognition in this stage become very important for the next stage since it will directly affect the accuracy of the next stage and it will ultimately influence the overall accuracy of the framework. Therefore, it is necessary to strive with all efforts to achieve the best accuracy for the recognition.

6.4.2.1 *Experiment*

Due to the classification of 18 character classes has to achieve the accuracy as high as possible, some strategies have been applied to produce some results which hopefully one of them has small error rate. The strategies are arranged with respect to classifiers or features.

The first strategy reuses the former features i.e. the feature of BED-WR for a classification of 18 character classes and keeps the use of Neural Network (NN) as the classifier. The configuration of the NN is three layers with some variation of its hidden layers. There are five settings of hidden layer respectively for each separated classification with 85, 95, 105, 120, and 130 nodes. These five composition is respectively evaluated over the validation set and the optimum one is then applied to the testing set. Tabel 14 provides the evaluation of the NN with feature BED-WR for various settings of the network configuration. The evaluation of the NN toward the validation set returns the optimum performance at 92.57% with the layer composition of $105 - 95 - 18$. The rate of testing set by using this network configuration produce the accuracy 94.50%. This rate serves as the baseline accuracy of the 18 character classes.

Table 14: The performance of Neural Network (NN) classification with the feature combination of the branch points, end points, pixel densities, and water reservoir (BED-WR) for 18 character classes.

Features	Layer size			Learning	Accuracy(%)	
	Input	Hidden	Output	rate	Validation Set	Testing Set
BED-WR	105	85	18	0.1	90.98%	
BED-WR	105	95	18	0.1	91.56%	
BED-WR	105	105	18	0.1	91.41%	
BED-WR	105	120	18	0.1	91.61%	94.18%
BED-WR	105	130	18	0.1	91.41%	
BED-WR	105	95	18	0.2	92.57%	94.50%
BED-WR	105	105	18	0.2	91.32%	
BED-WR	105	120	18	0.2	91.77%	
BED-WR	105	130	18	0.2	91.99%	

The first effort achieved the accuracy at **94.50%** which still remains the range around 5% for an improvement. Another strategy could be applied to reduce the gap by considering another classifier while keeping the use of BED-WR features as done previously. The option for classifier is the use of SVM classifier which better than NN in the case of input with high dimensional spaces. As some results should be produced, various kernel functions in the SVM are applied during classification. There are four widely used types of function to be employed as the kernel of the SVM i.e. the linear, polynomial, radial basis, or sigmoid function [6]. All those functions were involved in experiments with or without a scaling of the input. Scaling is basically a process to convert data values of the feature vector into a particular interval that are shrunk from its original values. With such an interval, some negative impact of the original data processing can be minimized. For example, dominating the big numerical ranges over smaller numerical ranges can be avoided by this restriction. Also, a large calculation problem during inner product of the kernel from original data can also be reduced. The output of this classification can be observed in Table 15.

The usage of a different classifier for classification positively contributes to the classification accuracy although the improvement is not really significant. This second effort indeed increases the accuracy from 94.50% to be **95.40%**. A further strategy is needed to raise a better accuracy for the next classification. Since the switch of the classifier has been applied previously, the switch of features should be the next strategy. In this turn, the feature of BED-WR is replaced by the feature of chain codes of the contour. The extracted features consist of 256 values as the input to the SVM classifier. Detail of the feature extraction can be found in part 5.3.3.1.

During practical works, several experiment runs were executed based on some predefined settings of SVM classifier. This configuration deal with the setting of the kernel functions and scaling of inputs. The performance of those results over validation set were respectively noted and then compared with each other in order

Table 15: The performance of Support Vector Machine (SVM) classification with the feature combination of the branch points, end points, pixel densities, and water reservoir (BED-WR) for 18 character classes.

Features	Kernel function	Scaling input	Accuracy (%) Validation set	Accuracy (%) Testing set
BED-WR	Linear	No scaling	92.91%	
BED-WR	Polynomial	No scaling	92.96%	94.78%
BED-WR	RBF	No scaling	92.02%	
BED-WR	Sigmoid	No scaling	44.36%	
BED-WR	Linear	Scaling	93.46%	95.40%
BED-WR	Polynomial	Scaling	82.51%	
BED-WR	RBF	Scaling	88.76%	
BED-WR	Sigmoid	Scaling	87.89%	

to find the best configuration. The evaluation on the validation set and testing set is presented in Table 16.

Table 16: The performance of Support Vector Machine (SVM) classification with the feature the chain codes for 18 character classes.

Features	Kernel function	Scaling input	Accuracy (%) Validation set	Accuracy (%) Testing set
Chain code	Linear	No scaling	93.39%	
Chain code	Polynomial	No scaling	95.39%	
Chain code	RBF	No scaling	95.73%	97.38%
Chain code	Sigmoid	No scaling	83.24%	
Chain code	Linear	Scaling	94.60%	96.47%
Chain code	Polynomial	Scaling	84.11%	
Chain code	RBF	Scaling	91.44%	
Chain code	Sigmoid	Scaling	89.56%	

The best rate of the recognition in these settings appeared from an experiment by using the RBF kernel. The LIBSVM was run over the samples without performing a scaling procedure to the input vectors. The best performance from experiments with those configuration is **97.38%**. A complete result of the character recognition and their confusions are given in the Table 17.

The accuracy from the last effort indicates a good performance of the last strategy for the classification of 18 character classes. The switch of the classifier from NN to SVM and features from BED-WR to chain codes are success to generate an acceptable accuracy for character classification as the foundation of the next stage in the framework.

Table 17: Confusion matrix of basic character recognition by SVM for 18 classes

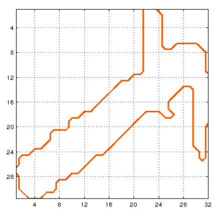
	ka	ga	nga	pa	ba	ma	ta	da	na	ca	ja	nya	ya	a	la	sa	wa	ha
ka	743	4	0	0	0	0	1	3	2	0	0	0	0	0	0	4	0	0
ga	2	624	0	0	0	0	0	3	0	0	0	0	0	0	0	4	2	2
nga	1	3	161	0	0	0	0	0	0	0	0	0	1	0	4	0	0	1
pa	0	1	0	911	0	7	1	0	0	0	0	0	0	0	0	0	0	0
ba	0	0	0	5	451	0	0	0	0	0	0	0	0	0	0	0	0	2
ma	0	0	0	4	1	727	0	0	0	0	0	3	0	0	0	0	0	0
ta	0	0	0	0	0	0	771	0	0	0	1	0	0	3	3	0	0	0
da	2	7	1	0	0	0	2	511	0	0	0	0	0	1	0	1	1	0
na	1	0	2	0	0	0	1	2	280	0	5	3	0	1	1	1	0	0
ca	0	0	0	0	0	0	0	0	0	42	2	1	0	0	0	0	0	2
ja	1	0	0	0	0	0	0	0	2	0	141	1	0	3	0	0	0	0
nya	0	0	0	0	0	0	0	0	2	0	2	171	2	1	0	0	0	0
ya	1	0	0	0	0	0	0	0	1	0	2	3	164	0	1	0	0	1
a	5	0	1	0	0	0	0	0	0	0	4	0	0	670	5	0	0	0
la	0	1	3	0	0	1	2	1	3	0	0	1	0	2	385	0	0	0
sa	8	12	0	0	0	0	0	3	1	0	0	0	0	0	0	566	0	1
wa	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	70	0
ha	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	1	1	275

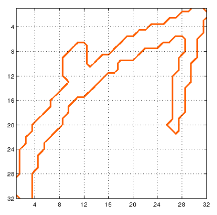
6.4.2.2 *Discussion of the Result*

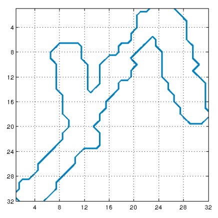
The chain codes work very well for recognition of the basic Lampung handwritten characters in 18 classes. The majority of handwritten characters can be accurately recognized. As stated previously, the performance of the SVM by employing the chain code features obtains the accuracy of 97.38% which is the best result that had ever been produced.

Although the majority of characters were successfully recognized, some small misclassification characters still exist. According to confusion matrix in Table 17, some problem occurred on the recognition of character *sa* ( ), *da* ( ), and *pa* ( ). Character *sa* ( ) confuses with character *ka* ( ) in 8 samples. In addition, character *sa* ( ) also confuses with character *ga* ( ) in 12 samples. This problem happens because the main shape of those three characters is relatively identical which are formed by character *ga* as the main shape. The only difference among of them is a small tip attached in the middle of the body as can be seen in Fig. 33.

The confusion of the character *sa* to be character *ga* in this example tends to occur because the small vertical tip on the top of the character *sa* is not identified as a branch of the character body but as an integral unit of the shape in the upper right zone of the character *sa* as can be seen in Subfig. 33a. In its feature representation, the chain codes of the tip will appear in the upper right area and then by the SVM classifier, those codes will be considered as the chain codes of the main body in the upper right as well. Meanwhile, the chain codes in the middle area of character *sa* where the tip should be located as can be observed in Subfig. 33c, consequently become blank. As the impact, the likelihood of the chain codes representation of character *sa* and *ga* being similar becomes increasing. As a consequence, the SVM


(a) The character *sa* confuses to character *ga* due to the position of its vertical tip


(b) The character *sa* confuses to character *ga* due to the tip is too small


(c) A successful recognition of character *sa*

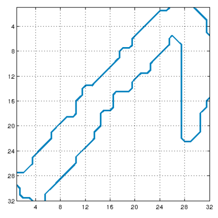
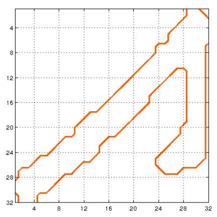

(d) A successful recognition of character *ga*

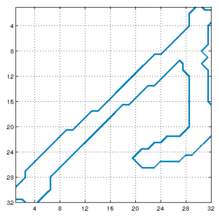
Figure 33: Some samples of character *sa* confuse to be character *ga* and a sample of character *sa* and *ga* which is correctly recognized.

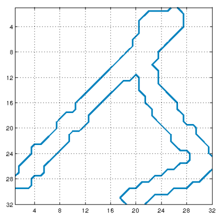
classifier can confuse the character *sa* in Subfig. 33a with the character *ga* in Subfig. 33d.

Another case of the confusion of character *sa* to be character *ga* is depicted in Subfig. 33b. In this example, the vertical tip of the character *sa* is tiny so that the edge boundary of this tip is also small. As the edge boundary of the tip is considerably small, the chain code representation of such a tip will not provide sufficient information to discriminate the character *sa* from *ga*. Thus, the character *sa* is often recognized as *ga*.

The same phenomenon is also applied to the confusion of the character *pa* ( ) as *ma* ( ). Both characters also have the same main shape which is formed by character *pa*. The difference with the previous confusion is only the curve orientation of characters. If the curve of the character *ga* in the previous analysis faces down, the character *pa* has a curve in the opposite orientation: its curve faces up. Thus, the problem of confusion between the character *pa* and *ma* has also happened in the same manner as the confusion of *sa* and *ga*.


(a) The confusion of character *da* as character *ga*


(b) A successful recognition of character *da*


(c) A correct recognized character *da* that is properly written

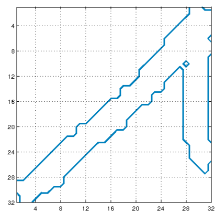

(d) A successful recognition of character *ga*

Figure 34: The sample of character *da* confuse to be character *ga* and its comparison to a correct recognition of character *da* and *ga*

The problem of confusion is also caused by the similarity of both characters in the shape of their contour images. This case can be found between character *da* ( ) and character *ga* ( ). One example of this confusion is shown in Subfig. 34a. If the contour of *da* in Subfig. 34a is compared to the contour of *ga* in Subfig. 34d, there is a high level similarity between both characters. A little difference can be observed on the right tail of the character. In the contour of the character *ga* in 34d, the right tail is only in form of a straight vertical line. While in the contour of character *da* in Subfig. 34a, the right tail is not a straight line but the tail is shifted a little to left. A proper character *da* has the right tail with a left-shifted stroke shich is much longer as can be seen in Subfig. 34b than the one which is shown in Subfig. 34a. Basically, the character *da* can be distinguished to *ga* if some contributors of handwritings wrote them properly. The difference between *da* and *ga* is really clear as can be compared between Subfig. 34c and Subfig. 34d. However, the writing style of some contributors, especially as shown by the sample in Subfig. 34a, enable them to be confused.

6.4.3 *Recognition of Diacritics*

Diacritics are a special element in the Lampung writing system because they are employed to accomplish a specific function. Particularly, they have an important relation to characters in composing syllables of a word as illustrated in Section 3.4. In fact, recognition of diacritics is also an essential task in Lampung handwritten character recognition framework. Although diacritics have particular association to characters, it is necessary to recognize them independently in one step to provide prior information for the association step.

Since the recognition of diacritics in this work is the first recognition of Lampung diacritics, no baseline of the diacritic recognition has been documented.

Note that as there is no specific name for diacritics in the absence of its position to the character, the name of each diacritic glyph would be mentioned as class 1 to class 7 as shown in Table 20.

6.4.3.1 *Experiment*

Experiments were performed using LIBSVM [6] to recognize 7 diacritic classes. Trials had been executed several times based on some predefined combination of SVM parameters as well as feature representations. The setting in this experiment consists of various type of kernel functions and the scaling process for inputs.

The feature representation is arranged in two groups of feature. The first group consists of the feature representation of the major axis length, minor axis length, orientation, aspect ratio, and eccentricity (F1) generated from original size Connected Component (CC). Then, the second one is composed by pixel densities only (F2) extracted from normalized CC. Each group is executed by applying four kernel functions and with or without scaling process of input samples. The kernel function RBF dominate the optimum classification in both features. The best accuracy in this configuration is performed by the classification with feature F2. The results of these experiments are given in Table 18.

Due to a necessary improvement of the accuracy, the last scheme is to concatenate the feature of F1 and F2 to be one single feature for classification. The configuration setting with respect to kernel functions and scaling remains the same as previously.

Table 18: The performance of Support Vector Machine (SVM) classification of each feature F1 and F2 for 7 diacritic classes.

Features	feature size	Kernel function	Scaling input	Accuracy (%)	
				Validation set	Testing set
F1	5	Linear	No scaling	67.29%	
F1	5	Polynomial	No scaling	78.07%	
F1	5	RBF	No scaling	81.10%	83.18%
F1	5	Sigmoid	No scaling	37.49%	
F1	5	Linear	Scaling	65.98%	
F1	5	Polynomial	Scaling	82.91%	
F1	5	RBF	Scaling	83.72%	85.96%
F1	5	Sigmoid	Scaling	73.51%	
F2	25	Linear	No scaling	94.72%	
F2	25	Polynomial	No scaling	93.00%	
F2	25	RBF	No scaling	95.25%	96.47%
F2	25	Sigmoid	No scaling	94.63%	
F2	25	Linear	scaling	94.91%	
F2	25	Polynomial	scaling	93.00%	
F2	25	RBF	scaling	95.47%	96.43%
F2	25	Sigmoid	scaling	94.41%	

With this configuration where both features are concatenated, the classification produces some better results. The use of the multi groups of feature shows improvement of classification rates compared to the rate of the single group of feature. The best rate was achieved by a concatenation of F1 and F2 by employing linear function without scaling its inputs. The accuracy 97.61% of the diacritic classification is sufficient to be used in the next stage of the framework. The detail of confusion matrix of this classification can be observed in the Table 20.

6.4.3.2 *Discussion of the Result*

The difference accuracy between classification of the diacritic by single group of feature F1 and F2 is rather big. The classification by using the feature F1 is not more than **86%**, while the classification by using the feature F2 can reach accuracy around **96%**. A reason of the big gap between the accuracy of the classification by F1 and F2 is that the size of the feature F1 is much smaller than the feature F2. A small size of the feature representation means that the feature representation contains a limited information about the characteristic of the diacritic. As the consequence, the performance will not be significant as shown in table 18. The maximum performance in this respect is 96.47%, achieved by the classification by using the feature F2. Although this maximal accuracy is considered as a good result, the prerequisite

Table 19: The performance of Support Vector Machine (SVM) classification with concatenation of the feature F1 and F2 for 7 diacritic classes.

Features	feature size	Kernel function	Scaling input	Accuracy (%) Validation set	Accuracy (%) Testing set
F1 and F2	30	Linear	No scaling	96.13%	97.61%
F1 and F2	30	Polynomial	No scaling	93.97%	
F1 and F2	30	RBF	No scaling	91.16%	
F1 and F2	30	Sigmoid	No scaling	37.77%	
F1 and F2	30	Linear	scaling	95.97%	
F1 and F2	30	Polynomial	scaling	94.16%	
F1 and F2	30	RBF	scaling	96.38%	97.31%
F1 and F2	30	Sigmoid	scaling	95.50%	

Table 20: Confusion matrix of diacritic recognition in 7 classes by SVM

Diacritic class	1	2	3	4	5	6	7
1 (ᒍ)	836	0	8	0	0	0	0
2 (⌒)	0	493	1	0	2	2	3
3 (ǀ)	3	2	1014	0	31	2	0
4 (◡)	0	0	1	1003	2	4	4
5 (–)	9	2	3	5	2083	0	2
6 (=)	0	0	2	1	5	96	4
7 (✓)	0	5	2	6	0	4	418

of the framework should achieve the accuracy as high as possible. Therefore, the accuracy should be improved in the further experiment because the accuracy can be still improved in the range of 1%-3%.

Concerning the usage of kernel functions during classification, the effect on the performance is relatively equal with each other except a few cases in the usage of the sigmoid function. In the classification by the feature of F1 without scaling, the performance is only 37.49%. The similar result also occurred in the classification by concatenation of feature F1 and F2 without scaling. The performance of classification only reach accuracy 37.77%.

The use of multi groups of feature with a proportional input size also enable the feature representation to contain more relevant information so that the classifier can recognize the diacritic in more accurate. This phenomenon can be observed from the fact that combination of the F1 and F2 as shown in table 19 contributes to the improvement of the performance. In the single group of feature, the feature F1 classified the diacritic with the performance of 85.96% and the feature F2 classified the diacritic with the performance of 96.47%. When both feature representations are combined, the performance of recognition with the RBF kernel increases to 97.61% which is the best rate in the classification of the diacritic.

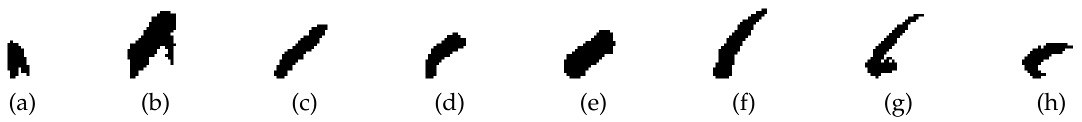


Figure 35: The sample of confusion among the diacritic class 3 and 5

The rate 98.10% of the diacritic recognition can be categorized as a high recognition rate. However, there are still some error cases that need to be examined regarding the occurrence of confusions. The Table 20 shows the detailed distribution of the confusion among diacritic classes after recognition process. The first concern about this table can be started from the highest confusion. There are 31 samples of the diacritic class 3 (ı) that is confused to be the class 5 (_). Meanwhile, there are 3 samples of the class 5 confused to be the class 3. Some samples of these confusions can be observed in the Fig. 35 where sub fig. 35a - 35e represent the confusion of class 3 to be class 5 and sub fig. 35f - 35h represent the confusion of class 5 to be class 3.

The first case of confusion occurred by the effect of noise on the body of the diacritic. As shown in sub fig. 35a and 35b, both confusion occurred due to an excess of tiny line at the right position of the main body of the diacritic. This tiny line consequently ruin the proper feature representation of the class 3. The SVM classifier recognized these samples as the class 5.

The next confusions in these samples is influenced by the writing style of contributors. The proper shape of class 3 is a small vertical line but many confusion samples of this class are written slightly to the right. In fact, the stroke of the diacritic is no longer vertical but it instead tends to form an angle around 45^o relatively to vertical axis or horizontal axis. As the classifier failed to identify the diacritic of class 3, then the classifier consider them as closer to class 5. The sample of this confusion can be observed in sub fig. 35c - 35e. This tendency also occurred during the recognition of the class 5. Samples in sub fig. 35f - 35h indicate three samples of the class 5 that are incorrectly identified as the class 3. The trigger for these confusions is similar to the reason of previous confusion. The writing style of contributors cause the change the correct stroke direction of diacritics. Instead of composing a small horizontal line as the shape of the class 5 or a vertical line as the shape of class 3, contributors made an angle around $30^o - 45^o$ to the horizontal axis which should not be happened. Since both class 3 and 5 might be written with the same angle which eventually form similar shapes as seen in Fig. 35, the classifier can recognize it as one of both classes.

6.4.4 *Recognition of Two-components Character*

The pair component of character 'ra' and 'gha' is obtained from classification of basic components. Therefore, combining of these components is an intermediate step after classification of basic components before performing classification of the Lampung compound character. The procedure of construction for the pair has been explained in part 5.3.3.2. However, that procedure only describes the task in a normative way. Therefore, this sub section will describe this normative way into practical aspect so that the execution can be carried out. In addition, the evaluation of classification is also provided.

6.4.4.1 *Experiment*

The experiment is started by isolating class 2 ( ) or 4 ( ) that are a potential component of character ”ra” ( ) and ”gha” ( ). Then, a process to search another pair entity from its surrounding is carried out. The first aspect to be concern in this regard is the number of neighbor to be examined prior to feature extraction. In this experiment, the number of the surrounding neighbors to be checked for pairing is restricted to four closest neighbors. This number is determined according to trial runs with the fact that execution of too many neighbors can be ineffective since the target of searching is only one class. In fact, four neighbors are adequate to provide a nominee for further pairing inspection.

The second aspect is the selection of feature representation of the pair. Since there are two components to be classified as one character, the feature representation should be a measurement such that it can represent a strong relation between two components. Features that suit to this criteria are the distance of gravity center between both components and the overlapping area of both bounding boxes. Although only two values, both values are powerful enough to determine the pairing.

The classification is addressed for three different classes. The first two classes represent a positive pairing, which means that both components unite as a two-components character. In this context, those relate to class 21 for character ”ra” and class 22 for character ”gha”. The remaining class represents other than aforementioned classes which explicitly indicates two independent components. This means that each component respectively represents a single component character.

A learning process of the pairing was run from training data with total sample 2377 pairs. Unfortunately, the number of character ”gha” is only 5 samples over total samples which is only 0.2%. In fact, the classifier has less information to model the character ”gha” during training phase. This condition is worse in the validation set. Among of 455 samples in the validation set, none of them is coming from character ”gha”.

For the purpose of testing, pairs are composed from the basic character of class 2 and 4. The number of pairing nominee depend on how many character of class 2 and 4 detected during the basic character recognition. By considering these classes as pivot, the pair is constructed from surrounding character classes. And the result is compared to real pairs from ground truth.

The experiment utilizes Support Vector Machine (SVM) for classification. To provide some comparative outputs, the dataset is run using several kernel function i.e. linear, polynomial, Radial Basis Function (RBF), and sigmoid function. The performance and analysis of the result is illustrated in the following subsection.

6.4.4.2 *Discussion of the Result*

Outcomes after experiments are collected and recorded to measure the performance of classification. The optimum outputs are generated by SVM classifier with linear kernel function. The result of classification for two-components character can be organized as illustrated in the table 21.

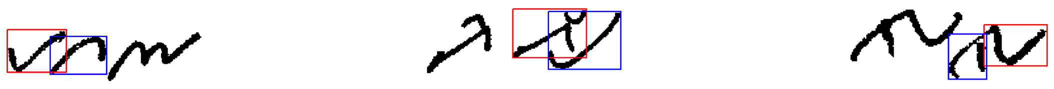
Table 21: The experiment outcomes of two-components character

	Classified pairing	Classified nonpairing
Pairing class	428 (True Positive/TP)	5 (False Negative/FN)
Nonpairing class	9 (False Positive/FP)	269 (True Negative/TN)

The performance is measured based on values in table 21. By applying formula 5.10 and the value in this table, the performance of two-components character recognition can be computed as "precision" and "recall". The computations are given in the following.

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} * 100\% = 97.94\%$$

This "precision" describes the performance of the recognition proportional to classifier outcomes. In this case, it shows that from the total pairing outcomes, 97.94% of them are correct pairings. The remaining pairing outcomes i.e. 9 pairs are misclassification. This misclassification occurred when a non-pairing class is classified as a pairing class. The trigger of this pairing is that both components are nearby and have an overlapping area as requiring by a two-components character. To see this misclassification, some samples are given in Fig. 36.



(a) (b) (c)

Figure 36: Samples of 2-components characters which are incorrectly recognized as 2-components characters by classifier

Subfig. 36a indicates two independent components, class 2 (character ga) and class 4 (character pa), which are correctly classified during single-component classification and identified as non-pairing in ground truth. However, during classification of pairing, classifier recognized them as a pair due to both are close with a sufficient overlapping area.

The second sample in subfig. 36b is classified as a pairing. One of its component is incorrectly recognized due to a touching diacritic on the top of class 2 (character ga). This component is recognized as class 16 (character sa) which is consequently impossible to construct a pair. Therefore, both components are regarded as two independent classes in the ground truth.

The last sample in subfig. 36c shows a broken component of character a ( ) into two pieces. One piece is similar to class 2 and another piece is similar to class 4. Since both components are close each other and having overlapping area, the classifier identifies them as a pair.

The next measurement is the ratio of the correct unit over the ground truth which is called "recall". By supplying the values from table 21, this measurement can be computing as,

$$\mathrm{Recall} = \frac{TP}{TP + FN} * 100\% = 98.85\%$$

By examining this formula, the correct pairing of recall is viewed as the correctness from perspective of the total ground truth. From this context, there are 5 pairs of misclassification and there is no further information from generated list by classifier. This indicate that the classifier has been failed to establish pairing nominees by particular reasons. To inspect them, Fig. 37 provide all pairs in this misclassification

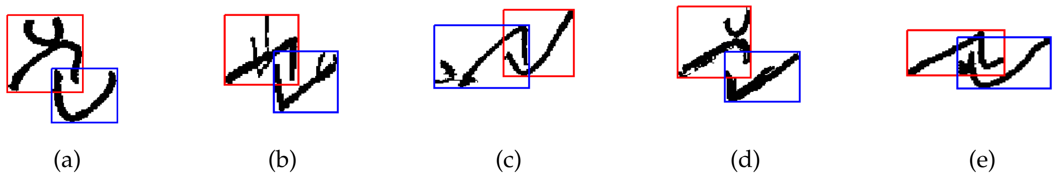


Figure 37: Samples of 2-components characters which are unknown after classification of two-components characters by classifier

The major problem in this situation is that one component has been detected as other classes than class 2 and 4. In fact, the remaining component does not have a proper pair. Therefore, the classifier considers the both components as two independent components. This fact can be observed from samples in Fig. 37. In this example, all cases of misclassification of pairing are caused by misclassification of components class 2 as another classes i.e. class 8, 16, 17, and 18 due to the touching of diacritic or noise.

Two aforementioned performance rates only rely on the correctness from the side of true pairings. While, the element of true non-pairing nominees also involved in classification process. As this component also contributes in overall classification process, true non-pairing nominees can not be just ignored. It should be incorporated in the measurement of the performance. The term of "accuracy" is used to rate the magnitude the overall of classification performance of two-component character. The rate is compute as follows.

$$\mathrm{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} * 100\% = 98.03\%$$

The achievement rate is 98.03% with respect to classification of two-components character. At this level, the performance can be regarded as the overall performance.

Finally, misclassification always emerge during any classification including in this work. Not much effort can be done to improve the rate since the classification in this phase always depend on the previous classification. One possible solution of the major problem in this case is to expand more features for pairing representation.

6.5 RECOGNITION OF COMPOUND CHARACTERS

Since the Lampung writing system also contain diacritics, a recognition of its basic form either the character or diacritic is only one part in the overall framework. The final goal is to recognize the complex structure which is composed by the character with or without diacritics. In more specific term, the final structure which contain character, with or without diacritics as one unit is called a compound character.

The association process of the character and diacritics are elaborated in the following subsections. The discussion concerns about two major topics regarding this association work. First, one diacritic is assigned to a character and this is called *a simple association*. Second, a compound character is specified from a character with or without diacritics and this work is called *a complete association*. The explanation covers experiment design as well as its process followed by a brief analysis of misclassification.

6.5.1 *Simple Association*

This simple association can be considered as an attempt to provide preliminary keys for handling a compound character because the structure of character and diacritic from this simple association can be considered as a subset of a compound character.

With the respect to this association, the way to solve the problem is started from a diacritic and then terminated to a character. This diacritic wise approach is more appropriate than a character wise one since a diacritic can not be independent and it is always issued by a character, while a character can stand independently without a diacritic. Therefore, the association step will always be initiated from diacritic side.

6.5.1.1 *Experiment*

There are two approaches to establish a simple association of a diacritic to a character. The first approach uses the nearest distance of a diacritic to a character as a basis of the association. The process of association firstly identifies the geometric center of a diacritic. From this point, the distance to the center of characters in proximity are measured. The closest distance is then selected and set as the association between a diacritic and a character. This process is completed until all diacritics get their companion. The performance of this approach serves as a baseline indicator for the simple association.

The second approach for this association is developed by a statistical method. From observation of diacritics around characters, the distribution of all diacritics of training data regardless of their classes is mainly accumulated over three different areas. This is reasonable since the diacritic can be positioned over three positions around a character. Figure 38 clarifies this distribution fact.

As the accumulation of diacritics distributes in three main areas, the number of components for GMM is ideally set to three as well. Nonetheless, this consideration will not always guarantee that an optimum solution could be achieved. Diacritics may spread over the area probably with more than three spots of accumulation. Therefore, the number of component for GMM can be still increased. For this reason,

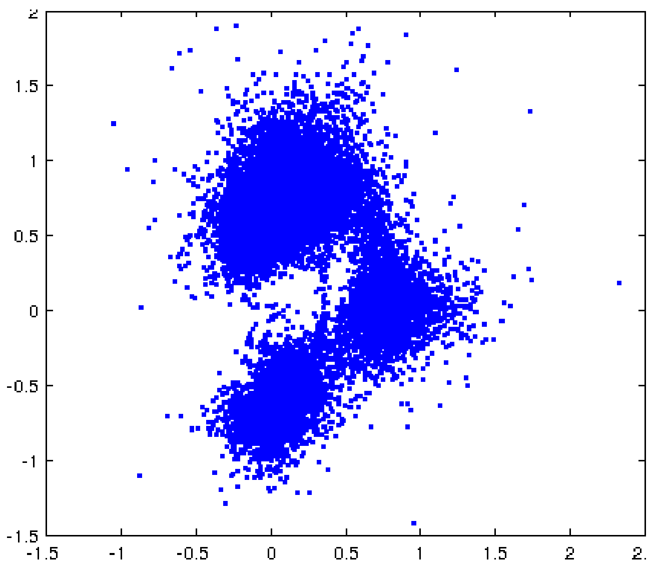


Figure 38: Distribution of diacritics around character of training set where each dot indicates coordinate of a diacritic over the character. The geometric center of the character lies at the coordinate of origin [21].

the number of component has been set to 5, 10, and 20 along with 3 components to provide more alternative outcomes for the best result.

To have an association, a diacritic as an input item should be paired to a character. However, a diacritic is regularly surrounded by many characters and each of them has a same chance to be paired. To minimize processing time, they are restricted to 6 closest nominees only. The chosen number is based on trial runs.

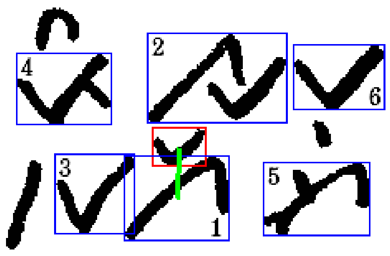


Figure 39: Association process of a diacritic around characters by applying Gaussian Mixture Model (GMM) [21].

For each character among of 6 nominees, a different feature vector of pairing with a diacritic as pivot is computed according to formula 5.6 of part 5.3.2.1. Fig. 39 illustrates a single formation between a character and a diacritic among of total 6 different pairings to be computed. Then the probability of a pairing can be estimated by a GMM with formula 5.7. As there are 6 possible pairings, the decision for association is taken based on a maximum probability by applying formula 5.8.

The experiment setup consists of 4 different composition with respect to the usage of parameters of components. This composition is illustrated as follows.

1. Parameters of the mixture model for each 3, 5, 10, and 20 densities are obtained from training sample by K-Means clustering regardless of diacritic and character classes. These parameters play the role of global parameter during experiment.

2. Parameters are generated by the same configuration and method as in the first setup, with an additional optimization technique by using Expectation Maximization (EM) algorithm. Parameters from this setup are also regarded as the global parameter.

3. The third scheme computes all parameters for each character specific distribution of training samples. Since there are 20 characters, 20 set of parameters are accordingly produced. In this case, parameters are considered as local parameters.

4. The last scenario is duplication of the third scheme with a combination of the global parameter. The procedure is to replace of the local component parameters by the global parameter. The idea of this replacement is based on the fact that some characters only have small samples. This may deteriorate the value of parameters during computation. By replacement with the global parameter, the risk of distortion for the lack of samples can be minimized. In practical, the replacement is administered such that the first replacement is zero parameter. This means that no parameter is replaced which is equivalent to use full local parameters. In the next step, one local parameter is replaced by the global parameter. The parameter to be replaced is from the component with the lowest sample, while the rest of 19 parameters are kept unchanging. The next replacement is a replacement of parameters from the first and second lowest samples by the global parameter. Then, the next is three parameters replacement from the first, second, and third lowest samples. This is done until all parameters replaced. In case of all local parameters has been replaced by the global parameter, the system configuration is equivalent to association with the global parameter as one described at composition 2.

6.5.1.2 *Discussion of the Result*

The number of correct association pairs in the simple association with the nearest distance is 5481 samples out of the total 6058 samples. Thus, this simple association achieves the accuracy **90.5%**. Since the rate is the first result, the rate is acknowledged as a baseline indicator for simple association or one-to-one association of a diacritic and a character.

Table 22 shows the output performance of the global model of the first and the second composition. In the first part of the table, the clustering is purely done without any optimization algorithms and the best performance is 91.9%. This performance is generated from the clustering with 10 components.

The rate is a little better for simple association with GMM after applying EM algorithm. This is given in the second part of the Table 22. The performance of the simple association with clustering and EM algorithm achieve 92.1% accuracy. This rate is recorded from the cluster with 20 components. However, the best trade-off

Table 22: Experiment of mixture model with the global parameters.

Number of density	Clustering method	Correct association (%)
3	K-Means	91.5
5	K-Means	91.5
10	K-Means	91.9
20	K-Means	91.8
3	K-Means with EM	91.6
5	K-Means with EM	92.0
10	K-Means with EM	91.9
20	K-Means with EM	92.1

between association performance and model complexity occurred at 5 components from 91.5% (without EM) to be 92.0% (with EM).

In the scheme where local parameters involved, samples are spread over 20 character classes. Concerning this situation, the number of sample in each GMM component may vary. There is a possibility that several characters will not have enough samples for each component. Therefore, the number of components for experiments with local parameters is restricted to 3 and 5.

Table 23 provides the result of the simple association by using the local parameter and a local parameter with specific replacement. In the first row, the computation is totally generated by local parameters with respect to each character class. The best accuracy of the association is 91.7% with 3 components. This performance is little lower than the best result of the association by using the global parameter.

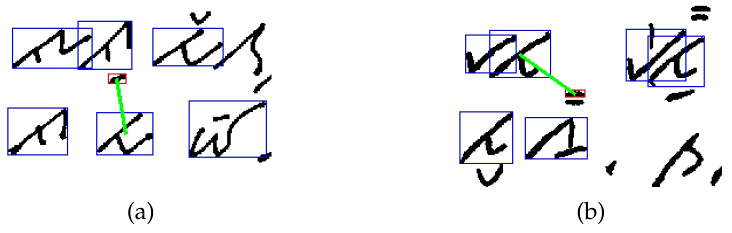


Figure 40: Incorrect association of a diacritic to the character: (a) due to domination of the diacritic position, (b) due to a less data sample.

As explained previously, the replacement is applied due to several classes only have very less samples. The replacement can lower the risk of infeasible parameters. The result of replacement can be observed in table 23. The term "number of character specific models" means the number of local parameters to be kept. For example, "the number of character specific models 18", means that local parameters to be kept is 18 parameters and the remaining 2 parameters are replaced by the global parameter. Accuracy of association with 3 components are gradually decreasing from 91.7% to 78.0%. None of them exceed the maximum accuracy of the association with the global parameter. A stable performance is demonstrated by the association

Table 23: Experiment of mixture model with replacements the local to global parameters

Number of character	Association rate of	
specific models	3 densities	5 densities
20 (fully local parameter)	91.7	91.5
19	91.9	92.0
18	91.8	**92.2**
17	91.8	**92.2**
16	91.7	**92.2**
15	91.5	**92.2**
14	91.3	**92.2**
13	91.0	92.1
12	90.3	**92.2**
11	90.0	92.1
10	89.7	92.1
9	88.9	92.1
8	88.1	92.1
7	87.4	92.0
6	85.6	92.0
5	84.3	91.9
4	83.4	91.9
3	82.0	91.8
2	79.6	91.7
1	78.0	91.7
Global model	91.6	92.0

with 5 components. The performance rate of this association fluctuates in the range 91.5% and 92.2%. The maximum accuracy is generated by several replacement compositions. They can be observed in the last column of the Table 23.

Two samples of incorrect association from the experiment are given in Fig. 40. These samples identified from association configuration with five components.

The first sample in fig. 40a indicates that the diacritic should belong to character *ka* ( ), but the classifier assigned this diacritic to character *ta* ( ). Misclassification of this type will occurred between two characters where the diacritic is located between both characters. Due to the concentration of diacritics in general is much more on the top of the character (see the distribution in Fig. 38), a diacritic is dominantly set as top diacritic of a character rather than other position. In this example, character *ta* will be powerful then character *ka* since this diacritic laid on the bottom of the character *ka*, while located on top for character *ta*. Thus the power of character *ta* to admit the diacritic is stronger than character *ka*.

Contrary to the first sample, the configuration of a character with a top diacritic in fig. 40b is not superior. This situation will happen whenever a character in the training set has very small number of samples compared to other characters. For this case, the occurrence of character *wa* ( ) is very low. This statistically means that the probability of character *wa* is very small (converge to zero). Consequently, the competition between character *wa* and character *ta* ( ) to be paired on this diacritic has been resolved to character *ta*.

6.5.2 *Complete Association*

The complete association is considered as the final stage in the Lampung handwritten character recognition framework. In this association, the smallest unit under examination is called a compound character. The structure of a compound character can be purely a character or a character with some diacritics. Both will be discussed in the following based on the context of a complete association.

6.5.2.1 *Experiment*

The experiment of the complete association is a task of building compound characters from basic characters, diacritics, and their associations. Based on the presence of its diacritics, the structure of a compound character can be distinguished as,

- A character.
- A character with a diacritic.
- A character with two diacritics.
- A character with three diacritics.
- A character with four diacritics.

To have compound characters, the nearest distance is applied as the foundation of the association between character and diacritic. This technique is chosen due to its simplicity during association process. The outcome of the nearest distance has been confirmed from the simple association as described in Subsection 6.5.1 and the current phase can benefit from it. Therefore, the current experiment is just the same step as in the simple association with an additional task for grouping characters with or without diacritics. In this grouping, the pivot of the process is switched from a diacritic wise approach in the previous association to a character wise approach in the current association. Each character is then examined whether having diacritics or not. If it has no diacritic, this character is solely acknowledged as a compound character. If it has diacritics, the algorithm marks all diacritics attached to this character and unites them as a compound character.

The number of diacritics may vary from one compound character to another compound character. The lack or excess of diacritics in a compound character could still occur after the association process.

6.5.2.2 *Discussion of the Result*

As the task in this stage is apparently a final step from successive tasks, the performance of this work is consequently an accumulation of the previous performances. Therefore, the performance is concurrently derived by elements such as the basic character classification, diacritic classification, two-components character classification, and the simple association of the character and diacritics. To review those works, table 24 summarizes the performance of each task.

Table 24: Performance of consecutive works prior to complete association

Order of the work	Recognition Target	Performance
1	Basic Characters	97.38%
2	Diacritics	97.61%
3	Two-components Character	98.03%
4	Simple Association of Character-Diacritic	90.50%

With respect to complete association, the number of compound characters in the test set is 7568 samples and with the nearest distance scheme for the complete association, the number of correct compound characters is 6103. Hence, the performance of the complete pipeline is 80.64%.

Table 25: Detail of diacritic number in compound characters of the test set for ground truth and outcome of the classifier

Source	Character type	Number of diacritics					Total	Percentage
		0	1	2	3	4		
Ground truth	Single component	2773	3611	633	118	5	7140	94.34%
	Two components	136	262	23	7	0	428	5.56%
	Total	2909	3873	656	125	5	7568	
Classifier (correct)	Single component	2054	3080	507	90	5	5736	80.34%
	Two components	125	219	18	5	0	367	85.75%
	Total	2179	3299	525	95	5	6103	80.64%
Incorrect	Single component	719	531	126	28	0		
	Two components	11	43	5	2	0		

Since the complete association is composed by some elements as stated previously, the accuracy can be refined by considering individual elements with their accuracy as indicated in Table 24. In a coarse category, there are two groups of compound character which are respectively composed by the single component and two components characters. Then, each group can be refined according to the number of its diacritics. By using the result from the table 25, the accuracy of each specific part is computed. All performance results are presented in table 26. It seems that the accuracies are a lower than expected due to many incorrect compound characters during the association process. A concise analysis of the reason for this problem is explained by guidance of Fig. 41 and 42.

Table 26: The accuracy of compound characters based on the group of specific element

Character type	Number of diacritics				
	0	1	2	3	4
Single component	74.07%	85.29%	80.09%	76.27%	100%
Two components	91.91%	83.59%	78.26%	71.43%	-

The figures show two snippets of document images containing compound characters resulting of the complete association. In this sample, the bounding box indicates a compound character. The line indicates an existing pair between a diacritic and a character. The symbol "T" (true) and "F" (false) in the upper left of bounding box respectively indicate the correct and incorrect association based on the ground truth. In general, an incorrect association can be caused by three issues as follows:

1. *Incorrect association due to a misclassification of the basic character:*
   The main element in a compound character is built by a basic character. If the involved character in a compound character has been stated as a misclassification from the former stage, the complete association will be incorrect. This is a direct consequence. A misclassification of the character automatically forwards the error to the compound character. The discussion regarding this topic has been addressed in subsection 6.4.2.

2. *Incorrect association due to a misclassification of the diacritic:*
   A typical similar occurrence can also be driven by a diacritic. A compound character with a misclassified diacritic will be interpreted as a different pattern compared to the one from the ground truth. As a distinction pattern indicates an incorrect formation, the result of the complete association automatically becomes incorrect as well. To review the result of this classification, the reader can refer to subsection 6.4.3.

3. *Incorrect association as a result of an incorrect assignment of a diacritic to a character:*
   The most sensitive problem of the complete association is the correlation of diacritics around the character. The incorrect association of a diacritic has a multiplier effect for the performance of the complete association. It can contribute to misclassification **multiple times**, proportionally to the number of diacritics. This can happen as follows. Assume there is a character with two diacritics and they form a compound character based on the ground truth. During basic recognition, all of the individual entity are correctly classified. However, during process of the association, one diacritic is assigned to another character close by. As a consequence, this association triggers two mistakes at once. First, the character under inspection lost one of its diacritic and in turn becomes incorrect. Second, this diacritic will be reassigned to another compound character. When this diacritic is attached to a correct compound character, the resolved compound character turns to be incorrect because it

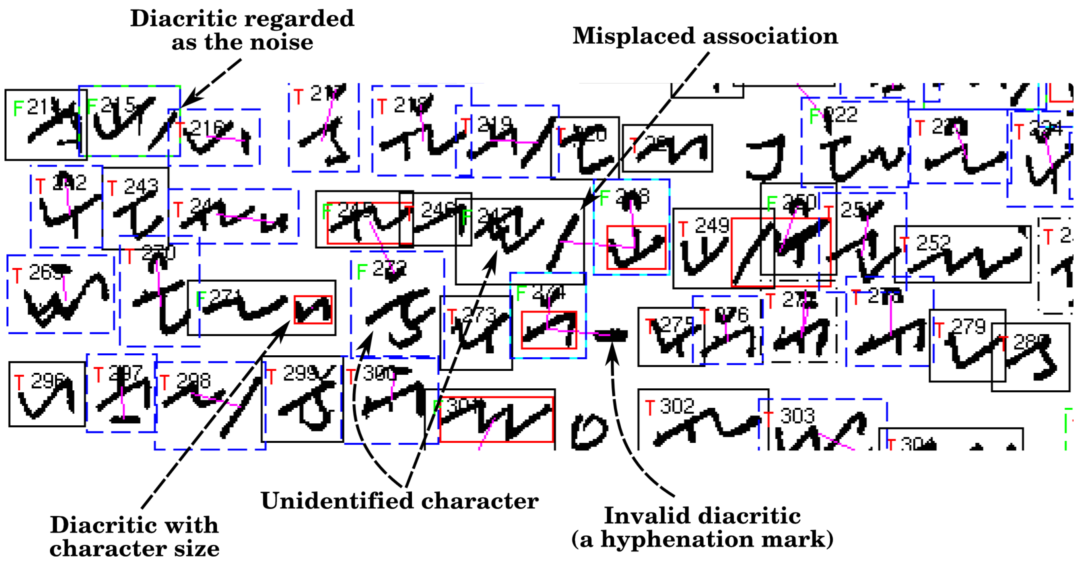

Diacritic regarded as the noise
Misplaced association
Diacritic with character size
Unidentified character
Invalid diacritic (a hyphenation mark)

Figure 41: The first snippet of a document image indicates various types of incorrect associations of diacritics and characters

should not have an additional diacritic. Therefore, the more diacritics in a compound character, the higher probability of incorrect association may occur. These errors are imposed by the following two common problems and both may influence each other.

a) The first problem is characterized by an inappropriate number of diacritics in a compound character. Typically, this case comprises of the loss of any diacritics, less or more assigned diacritics.

   The reason of lost diacritics is because this diacritic is regarded as noise so it will never be found during the association process. One sample of this case can be observed in Fig. 41 at bounding box 215. The diacritic on the right of the character is detected as noise. During the association process, it cannot be found and no association can be made. Therefore, the character is solely left unrelated. Another possibility is caused by the size of the diacritic. In some cases, the diacritic may almost be similar to the size of a character due to the writing style of contributors. In a condition where the shape of a diacritic resembles to a character, these diacritics are further grouped as the instance of characters. In fact, it does not present as a diacritic during the association process. The sample of this compound character can be observed in Fig. 41 at bounding box 271. An *s-shape* diacritic is written on the right of the character *la* (ꦭ) almost as big as the character. It also resembles to character *ha* (ꦲ). Due to both facts, the diacritic is not grouped as a diacritic but instead is grouped as character *ha*.

   The association of less or more diacritics usually occurs because the diacritic is shifted to a closed compound character during the association process. In fact, the holding compound character of this diacritic lost

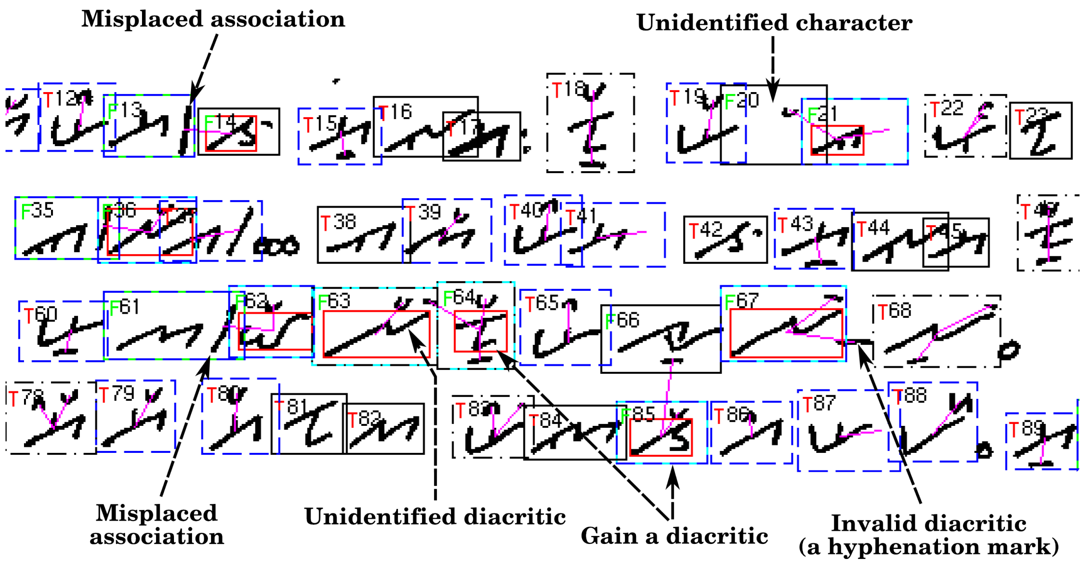


Figure 42: The second snippet of an image document indicates various type of incorrect association of diacritics and characters

its diacritic and the neighbor compound character got more diacritics. Figure 42 at bounding box 63 and 64 shows samples of this association. A *horizontal line* diacritic in bounding box 63 is incorrectly associated to the character in bounding box 64. Hence, the holding character in bounding box 63 lost its *horizontal line* diacritic while the character in bounding box 64 gained one extra diacritic. This loss and gain occurrence is the main source of a significant error illustrated by table 25. Both compound characters, the one that lost as well as the one that gained the diacritic will turn into incorrect. This means that the number of errors will be twice the error for each incorrect association. Note that this occurrence will be called "loss and gain" hereinafter.

Another specific case of surplus the diacritic could also arise with respect to the noise. The surplus occurs whenever the noise is detected as a diacritic and attached to a compound character during the association process. Hence, this compound character accepts a new diacritic which should never exist in the ground truth. The sample of this association can be seen in Fig. 42 at the character in bounding box 67. In this sample, the noise was derived by a hyphenation mark which is similar in shape to a diacritic and resides on the right position of the character.

b) The second issue is an incorrect association due to a misplaced position of diacritics. Some particular diacritics can only be put on a specific position around the character. Whereas the complete association with the nearest distance scheme only assures an association to a character and there is no concern with respect to the position of those diacritics. Thus, the position of the diacritic could be anywhere in the proximity of the character. A typical case of this error can be observed in Fig. 42 at bounding box 13 and 14. The diacritic *nengen* (ɹ) should be located on the right of the character. In this sample, the diacritic *nengen* is assigned to the character in bounding box 14. This means that the diacritic is positioned on the left of the character in bounding box 14 which is not valid position for diacritic *nengen*. Whereas the character in bounding box 13 lost its diacritic. This situation also occurred between characters in bounding box 35 and 36 and bounding box 61 and 62.

The big part of incorrect associations are produced by the single component characters without diacritics and single characters with one diacritic with an accuracy 74.07% and 85.29% respectively. According to table 25, the number of incorrect associations for both are 719 and 531, while the rest are respectively 156. The accuracy of the single component character without diacritic should not be far from the accuracy of the basic character recognition i.e. 97.38% (see table 24). Likewise, the accuracy of the single component character with one diacritic should not be far from the accuracy of the simple association of character and diacritic i.e. 90.50% (see table 24).

Based on the inspection on 4 document images of the test set as samples, the reason of this situation is dominantly caused by the occurrence of the "loss and gain" of the diacritics. To support this rationale, a simulation of the "loss and gain" of the diacritics is performed. Assume that all loss occurrences contribute to incorrect association of the single component characters without diacritic. It also presumes that the incorrect single component characters with 1,2,3, and 4 diacritics (see table 25) only lost one diacritic such that those single component characters only cause one incorrect factor toward the single component characters without diacritic. Therefore, the total incorrect gain for single component character without diacritic is $531 + 128 + 28 + 0 = 685$. Thereby, the source of 719 incorrect derives from 685 of single character with one or more diacritics. There is only 34 remaining.

The reason of this significant deviation is not only the "loss and gain" of the diacritics. Another source of the incorrect association is the unidentified characters during preprocessing phase. The sample of these unidentified characters can be observed in Fig. 41 at bounding box 247 and 272 and Fig. 42 at bounding box 20. During inspecting the test set, these unidentified characters are triggered by broken characters, touching characters, or character with diacritics. All these factors apparently contribute to degradation of the accuracy in the association diacritics to characters. From total 7568 compound characters in the test set, there are 418 unidentified characters or 5.52%. Among of those 418, only 94.34% (see table 25) or around 394 samples belong to single component character. Since there are 5 parts (comprising of $0, 1, \ldots, 4$ diacritics) and only 4 of them contain incorrect associations (compound characters with 4 diacritics are 100% correct), the average loss of characters during preprocessing for each part is $394/4 \approx 98$. In fact, it is most likely that the remaining 34 of incorrect compound character derives from these 98. The excess 64 of incorrect from unidentified characters could already be scattered in 685 of the "loss and gain".

Considering the problem of incorrect association, some alternative solutions can be proposed for improvement of the association performance. One of them is by applying additional rules during the association of a diacritic to a character. Rules can be established based on association rules in the Lampung writing system to deal with the legitimate association and the evidence of the samples in dataset to notice the actual characteristic of the data.

6.5.3 *Remark*

The discussion of the complete process of the Lampung handwritten character recognition framework has been finished. The results and assessments of each stage have been given. Finally, the completed framework has become a substantive groundwork to guide further developments in Lampung handwritten character recognition.

7 CONCLUSION

The accomplishment of the Lampung handwritten character recognition framework is an important milestone for the beginning of Lampung handwritten character recognition research. The opportunity has been opened for everyone who is interested to do research in this area. The researchers will have chances to improve the framework.

The following sections highlight some important issues inferred from the work of Lampung handwritten character recognition research. Each section highlights several results or emphasizes some aspects need to be paid attention. Section summary contains outcomes or remarks on the research. While, section outlook contains statement on some future works and potential development of the framework in the future.

7.1 SUMMARIES

Social and Culture Impacts: Lampung script emerged since long time ago. The existence of Lampung script is a proof that the Lampung people in the past have been living with a tradition of writing. Some old manuscripts have been discovered. Some of them were stored in museums and some others were owned by people. Those manuscripts indicate that many writing activities have been done in the past. Lampung script is rarely used recently, especially after Roman script was introduced as the official script. The Lampung script is just regarded as an ornament with less applying in writings. This research can be considered as one endeavor to emphasize the importance of the script for society and also an effort to save the script from extinction.

In the last decade, there were very limited research regarding Lampung script with respect to Document Analysis and Recognition (DAR). Such a research only relies on a limited number of datasets and the results have never been successfully published for the international community. In fact, it is hard to carry out the research since there is no preceding knowledge or dataset available to support the research on this Indic related script. This work, beside publishing some results, has also pioneered an initial dataset for the purpose of the research in the area of DAR. This dataset is stored at the website of LS XII - Department of Computer Science, TU Dortmund, Germany[1]). With the existence of this dataset, it is expected that more researchers will be attracted and more researches will be triggered.

The Framework: The complete framework of Lampung handwritten character recognition has been defined. The main structure of the framework is characterized by three tasks, annotation, recognition of basic elements, and recognition of compound characters.

1 http://patrec.cs.tu-dortmund.de/cms/en/home/Resources/index.html

Annotation with semi-supervised approach in this work is very promising for the labeling task. The role of human expert in this labeling can be reduced significantly without sacrificing much of the quality. The complexity of the approach much more depends on the number of clusters during clustering process rather than on number of samples. By labeling only 0.48% of the samples represented by cluster, the accuracy can reach the rate 86.21%. The achievement rate is considered to be reasonable especially with respect to the number of samples to be labeled.

The basic elements in this work are the character and diacritic recognition. The characters are initially recognized into 18 classes and then completed to be 20 classes by post-recognizing of two-components characters from those 18 classes. The chain code directions of the character contour were used as features and the SVM was employed in classification of 18 classes. The recognition rate is 97.38% for 18 classes recognition task. Having the result of 18 classes, a further recognition is then performed to identify two-components characters. The usage of the distance of the gravity-center between both components and the overlapping area of both bounding boxes as the feature of two-components character classification gives the recognition rate of 98.03%. Meanwhile, the diacritic classification uses combination features which are extracted from diacritic in original and in normalized size. The performance of diacritic classification into 7 classes, regardless of its position around the character, is 97.61%.

The final task in the framework is to compose a compound character by association of the character with the diacritics. The task is started by a one-to-one association between a diacritic and a character. Then in the second round, every character is reviewed for having diacritics nearby or not. With the closest distance between diacritics and a character as the association criteria, the forming of compound characters achieves performance of 80.64%.

7.2 OUTLOOK

During completion of the framework, some new challenges arose in many levels of the framework. The framework could be enhanced by resolving some of these issues. Among those issues are the following:

1. Line extraction is not conducted in this work due to preventing a higher complexity of the work. One advantage of supplying definitive line separations for character is that it can assist the location of diacritics. There are many approaches to perform line extraction. One possible approach to fulfill the line extraction task is by applying the Minimum Spanning Tree (MST) concept [61] from graph theory. The concept is appropriate because Lampung character is a type of non-cursive script.

2. A character bounding box can be enlarged vertically to cover the area on the top and the bottom of the character. It can then partitioned horizontally into 3 regions as a new strategy for handling diacritics. This way could probably help to identify the diacritic around the character. The top diacritic can be found in the upper baseline region and the bottom diacritic can be found in the lower baseline region. The right diacritic can be located within the baseline.

3. Features used for identifying two-component characters in the present work are only the distance of gravity-center and the overlapping area. Other features can be added for example, the position of one component relatively to another. This feature can be used to discriminate the left and right component of the character so that every two-components class can be classified accurately.

4. This work employs Neural Network (NN) and Support Vector Machine (SVM) as the single classifier systems. Those utilized classifiers, as well as other classifiers, can be combined as multi-classifier system for this framework. The multi-classifier system can lead to a new results and is expected to be a potential improvement of the single-classifier classification.

5. The most challenging task in this framework is to recognize the final compound character that consists of a character with or without diacritics. In the framework, the task is divided into two steps. First, the association of a diacritic and a character is assigned as one-to-one mapping by using closest distance. Later, the pivot is shifted to the character and associate to all nearby diacritics. The closest distance can be combined with additional rules for preventing any improper association which can improve the performance. Beside adding rules, the association criteria can also be altered by using Gaussian Mixture Model (GMM).

BIBLIOGRAPHY

[1] M. Agrawal and D. S. Doermann. Clutter Noise Removal in Binary Document Images. In *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, pages 556–560. IEEE Computer Society, 2009. (Cited on page 11.)

[2] N. Arica and F. T. Yarman-Vural. An Overview of Character Recognition Focused on Off-line Handwriting. *Transactions on Systems, Man and Cybernetics – Part C*, 31(2):216–233, May 2001. (Cited on pages 9 and 10.)

[3] U. Bhattacharya and B. B. Chaudhuri. Databases for Research on Recognition of Handwritten Characters of Indian Scripts. In *Proceedings of the 2005 8th International Conference on Document Analysis and Recognition*, volume 2, pages 789 – 793, 2005. (Cited on page 64.)

[4] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 223 Spring Street, NY, USA, 2006. (Cited on pages 21, 23, 28, 31, 32, 33, 34, 35, 65, 75, and 100.)

[5] H. Bunke. Recognition of Cursive Roman Handwriting - Past, Present and Future. In *Proceedings of the 2003 7th International Conference on Document Analysis and Recognition*, volume 1, pages 448–459. IEEE Computer Society, 2003. (Cited on page 7.)

[6] C. -C Chang and C. -J Lin. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. (Cited on pages 80, 82, 83, 107, and 111.)

[7] B. B. Chaudhuri and S. Ghosh. Orientation Detection of Major Indian Scripts. In *Proceedings of the International Workshop on Multilingual OCR*, MOCR '09, pages 8:1–8:7, New York, NY, USA, 2009. ACM. (Cited on pages 57, 69, 70, and 100.)

[8] M. Cheriet, N. Kharma, C. -L. Liu, and C. Y. Suen. *Character Recognition Systems: A Guide for Students and Practitioners*. Wiley-Interscience, 2007. (Cited on pages 7, 9, 15, 23, 26, 32, 33, 35, 69, 75, and 100.)

[9] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–22, 1977. (Cited on pages 36 and 77.)

[10] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley & Sons, Inc., New York, NY, USA, 2nd edition, 2001. (Cited on pages 21, 22, 23, 26, 28, 32, 33, 65, 75, and 100.)

[11] G. A. Fink. *Markov Models for Pattern Recognition, From Theory to Applications*. Advances in Computer Vision and Pattern Recognition. Springer, London, 2 edition, 2014. (Cited on page 22.)

[12] G. A. Fink and T. Plötz. Developing Pattern Recognition Systems Based on Markov Models: The ESMERALDA Framework. *Pattern Recognition and Image Analysis*, 18(2):207–215, June 2008. (Cited on pages 62 and 90.)

[13] B. G. Gatos. Imaging Techniques in Document Analysis Processes. In D. Doermann and K. Tombre, editors, *Handbook of Document Image Processing and Recognition*, pages 73–131. Springer London, 2014. (Cited on pages 11 and 14.)

[14] D. Ghosh, T. Dube, and A. P. Shivaprasad. Script Recognition – A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2142 –2161, dec. 2010. (Cited on page 39.)

[15] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2nd edition, 2002. (Cited on pages 8, 9, and 10.)

[16] M. Haji, T. D. Bui, and C. Y. Suen. Removal of Noise Patterns in Handwritten Images Using Expectation Maximization and Fuzzy Inference Systems. *Pattern Recognition*, 45(12):4237–4249, December 2012. (Cited on pages 10 and 11.)

[17] N. Hajj and M. Awad. Isolated Handwriting Recognition via Multi-stage Support Vector Machines. In *6th IEEE International Conference on Intelligent Systems, IS 2012, Sofia, Bulgaria, September 6-8, 2012*, pages 152–157, 2012. (Cited on page 37.)

[18] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, New York, 2001. (Cited on page 35.)

[19] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006. (Cited on pages 65 and 66.)

[20] A. Junaidi, S. Vajda, and G. A. Fink. Lampung - A New Handwritten Character Benchmark: Database, Labeling and Recognition. In *Proceeding of the Joint Workshop on Multilingual OCR and Analytics for Noisy Unstructured Text Data*, pages 105–112, Beijing, China, 2011. ACM Press. (Cited on pages xi, xiii, 20, 68, 73, 94, and 102.)

[21] A. Junaidi, R. Grzeszick, S. Vadja, and G. A. Fink. Statistical Modeling of the Relation between Characters and Diacritics in Lampung Script. In *Proceedings of the 2013 12th International Conference on Document Analysis and Recognition*, pages 663–667, Washington DC, USA, August 2013. IAPR, IEEE Computer Society. (Cited on pages xi, xii, 76, and 119.)

[22] A. Kacem, N. Aouïti, and A. Belaïd. Structural Features Extraction for Handwritten Arabic Personal Names Recognition. In *ICFHR - 13th International Conference on Frontiers in Handwriting Recognition - 2012*, pages 268–273, Bari, Italy, September 2012. IEEE. (Cited on page 20.)

[23] K. Khurshid, I. Siddiqi, C. Faure, and N. Vincent. Comparison of Niblack Inspired Binarization Methods for Ancient Documents. In K. Berkner and L. Likforman-Sulem, editors, *DRR*, volume 7247 of *SPIE Proceedings*, pages 1–10. SPIE, 2009. (Cited on pages 62 and 90.)

[24] K. Kise. Page Segmentation Techniques in Document Analysis. In D. Doermann and K. Tombre, editors, *Handbook of Document Image Processing and Recognition*, pages 135–175. Springer London, 2014. (Cited on page 11.)

[25] L. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004. (Cited on pages 65 and 66.)

[26] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. In *Intelligent Signal Processing*, pages 306–351. IEEE Press, 2001. (Cited on pages 64, 65, and 96.)

[27] C. -L. Liu and H. Fujisawa. Classification and Learning for Character Recognition: Comparison of Methods and Remaining Problems. In *International Workshop on Neural Networks and Learning in Document Analysis and Recognition*, 2005. (Cited on page 22.)

[28] C. -L Liu and K. Marukawa. Normalization Ensemble for Handwritten Character Recognition. In *Ninth International Workshop on Frontiers in Handwriting Recognition*, volume 0, pages 69–74, Los Alamitos, CA, USA, 2004. IEEE Computer Society. (Cited on pages 14 and 15.)

[29] C. -L. Liu, M. Koga, H. Sako, and H. Fujisawa. Aspect Ratio Adaptive Normalization for Handwritten Character Recognition. In T. Tan, Y. Shi, and W. Gao, editors, *ICMI*, volume 1948 of *Lecture Notes in Computer Science*, pages 418–425. Springer, 2000. (Cited on page 15.)

[30] C. -L. Liu, K. Nakashima, H. Sako, and H. Fujisawa. Handwritten Digit Recognition: Investigation of Normalization and Feature Extraction Techniques. *Pattern Recognition*, 37(2):265–279, 2004. (Cited on pages 7, 8, and 15.)

[31] S. P. Lloyd. Least Squares Quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. (Cited on page 66.)

[32] L. M. Lorigo and V. Govindaraju. Offline Arabic Handwriting Recognition: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28:712–724, May 2006. (Cited on page 20.)

[33] M. Lutf, X. You, and H. Li. Offline Arabic Handwriting Identification Using Language Diacritics. In *20th International Conference on Pattern Recognition*, pages 1912 –1915, August 2010. (Cited on page 58.)

[34] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In L. M. L. Cam and J Neyman, editors, *Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–296, 1967. (Cited on page 77.)

[35] S. Mozaffari, K. Faez, F. Faradji, M. Ziaratban, and S. M. Golzan. A Comprehensive Isolated Farsi/Arabic Character Database for Handwritten OCR Research. In *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule (France), 2006. (Cited on page 64.)

[36] D. K. Nguyen and T. D. Bui. Recognizing Vietnamese Online Handwritten Separated Characters. In *International Conference on Advanced Language Processing and Web Information Technology*, volume 0, pages 279–284, Los Alamitos, CA, USA, 2008. IEEE Computer Society. (Cited on page 58.)

[37] W. Niblack. *An Introduction to Digital Image Processing*. Strandberg Publishing Company, Birkeroed, Denmark, 1985. (Cited on pages 12, 62, and 90.)

[38] N. Otsu. A Threshold Selection Method from Gray-level Histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, January 1979. (Cited on pages 12, 62, and 90.)

[39] E. Öztop, A. Y. Mülayim, V. Atalay, and F. Yarman-Vural. Repulsive Attractive Network for Baseline Extraction on Document Images. *Signal Process.*, 75:1–10, May 1999. (Cited on page 16.)

[40] U. Pal and S. Datta. Segmentation of Bangla Unconstrained Handwritten Text. In *Proceedings of the 2003 7th International Conference on Document Analysis and Recognition*, volume 2, pages 1128–1132, Washington, DC, USA, 2003. IEEE Computer Society. (Cited on pages 56, 69, 70, and 100.)

[41] U. Pal, A. Belaïd, and C. Choisy. Water Reservoir Based Approach for Touching Numeral Segmentation. In *Proceedings of the 2001 6th International Conference on Document Analysis and Recognition*, ICDAR '01, pages 892–896. IEEE Computer Society, September 2001. (Cited on pages 56, 69, 70, and 100.)

[42] U. Pal, A. Belaïd, and C. Choisy. Touching Numeral Segmentation Using Water Reservoir Concept. *Pattern Recognition Letters*, 24(1-3):261–272, January 2003. (Cited on pages 56, 69, 70, and 100.)

[43] U. Pal, S. Kundu, Y. Ali, H. Islam, and N. Tripathy. Recognition of Unconstrained Malayalam Handwritten Numeral. In *ICVGIP*, pages 423–428, 2004. (Cited on pages 56, 69, 70, and 100.)

[44] U. Pal, R. K. Roy, K. Roy, and F. Kimura. Indian Multi-Script Full Pin-code String Recognition for Postal Automation. In *Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, ICDAR '09, pages 456–460, Washington, DC, USA, 2009. IEEE Computer Society. (Cited on pages 57, 69, 70, and 100.)

[45] M. Pechwitz, S. S. Maddouri, V. Märgner, N. Ellouze, and H. Amiri. IFN/ENIT - Database of Handwritten Arabic Words. In *Proc. of CIFED 2002*, pages 129–136, October 2002. (Cited on page 58.)

[46] R. Plamondon and S. N. Srihari. On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, January 2000. (Cited on page 7.)

[47] T. Pudjiastuti. *Aksara dan Naskah Kuno Lampung dalam Pandangan Masyarakat Lampung Kini*. Department of Education and Culture, Republik of Indonesia, Jakarta, 1997. (Cited on pages 3, 39, 44, and 49.)

[48] S. Sa. *Lampung Pepadun dan Saibatin/Pesisir – Dialek O/Nyow dan Dialek A/Api*. Buletin Way Lima Manjau, Jakarta, 2012. (Cited on pages 44 and 49.)

[49] J. Sauvola, T. Seppänen, S. Haapakoski, and M. Pietikäinen. Adaptive Document Binarization. In *Proceedings of the 1997 4th International Conference on Document Analysis and Recognition*, volume 1, pages 147–152. IEEE Computer Society, August 1997. (Cited on pages 12, 13, and 62.)

[50] S. Shelke and S. Apte. Multistage Handwritten Marathi Compound Character Multistage Handwritten Marathi Compound Character. *Journal of Pattern Recognition Research*, 6(2):253–268, 2011. (Cited on pages xi, 36, 59, and 60.)

[51] N. Stamatopoulos, B. Gatos, and A. Kesidis. Automatic Borders Detection of Camera Document Images. In *2nd International Workshop on Camera-Based Document Analysis and Recognition, Curitiba, Brazil*, pages 71–78, 2007. (Cited on page 11.)

[52] N. Stamatopoulos, G. Louloudis, and B. Gatos. Efficient Transcript Mapping to Ease the Creation of Document Image Segmentation Ground Truth with Text-Image Alignment. In *Proceedings of the 2010 12th International Conference on Frontiers in Handwriting Recognition*, pages 226–231, Washington, DC, USA, November 2010. IEEE Computer Society. (Cited on page 64.)

[53] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010. (Cited on page 17.)

[54] S. Theodoridis and K. Koutroumbas. *Pattern Recognition*. Academic Press, Inc., Sand Diego, CA, USA, third edition edition, 2006. (Cited on pages 22, 23, 75, and 100.)

[55] D. C. Tran, P. Franco, and J. Ogier. Accented Handwritten Character Recognition Using SVM - Application to French. In *Proceedings of the 2010 12th International Conference on Frontiers in Handwriting Recognition*, pages 65 –71, November 2010. (Cited on page 57.)

[56] S. Vajda and G. A. Fink. Exploring Pattern Selection Strategies for Fast Neural Network Training. In *2010 20th International Conference on Pattern Recognition*, pages 2913 –2916, August 2010. (Cited on page 65.)

[57] S. Vajda, A. Junaidi, and G. A. Fink. A Semi-Supervised Ensemble Learning Approach for Character Labeling with Minimal Human Effort. In *Proceedings of the 2011 11th International Conference on Document Analysis and Recognition*, pages

259–263, Beijing, China, September 2011. IAPR, IEEE Computer Society. (Cited on pages xi, 64, 65, 68, 88, 97, and 99.)

[58] G. Vamvakas, B. Gatos, and S. J. Perantonis. Handwritten Character Recognition Through Two-stage Foreground Sub-sampling. *Pattern Recognition*, 43(8):2807–2816, August 2010. (Cited on page 20.)

[59] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995. (Cited on page 31.)

[60] L. Xu, B. Xiao, C. Wang, and R. Dai. *Neural Information Processing: 13th International Conference, ICONIP 2006, Hong Kong, China, October 3-6, 2006. Proceedings, Part II*, chapter A Novel Multistage Classification Strategy for Handwriting Chinese Character Recognition Using Local Linear Discriminant Analysis, pages 31–39. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. (Cited on page 36.)

[61] F. Yin and C. -L. Liu. Handwritten Text Line Extraction based on Minimum Spanning Tree Clustering. In *International Conference on Wavelet Analysis and Pattern Recognition, 2007. ICWAPR '07.*, volume 3, pages 1123 – 1128, November 2007. (Cited on pages 16 and 132.)

A APPENDICES

A.1 CHARACTER DISTRIBUTION OF 11 CLASSES

Table 27: Character distribution in 11 classes

Class	Number of Samples	% of distribution
ka*	8077	22.95%
nga*	5352	15.21%
pa*	8629	24.52%
ta	3092	8.79%
da	2157	6.13%
na*	1756	4.99%
ca*	1394	3.96%
nya	773	2.2%
ya	660	1.88%
wa	254	0.72%
ne.*	3049	8.66%
Total	35193	100%

A.2 CHARACTER DISTRIBUTION OF 18 CLASSES

Table 28: Character distribution in 18 classes

Class	Number of Samples	% of distribution
ka	3131	9.74%
ga	2633	8.19%
nga	695	2.16%
pa	3802	11.83%
ba	1957	6.09%
ma	2874	8.94%
ta	3093	9.62%
da	2164	6.73%
na	1201	3.74%
ca	238	0.74%
ja	563	1.75%
nya	772	2.4%
ya	660	2.05%
a	2928	9.11%
la	1715	5.34%
sa	2305	7.17%
wa	254	0.79%
ha	1155	3.59%
Total	32140	100%

A.3 DIACRITIC DISTRIBUTION OF 7 CLASSES

Table 29: Diacritics distribution in 7 classes

Class	Number of Samples	% of distribtuion
1 (✓)	3470	14.01%
2 (∧)	1548	6.25%
3 (|)	4763	19.23%
4 (◡)	4145	16.73%
5 (–)	8607	34.74%
6 (=)	465	1.88%
7 (∽)	1777	7.17%
Total	24775	100%